Eletrônica Digital
Práticas de Laboratório
Volume 1

Rafael Manfrin Mendes

EDITORA SCIENZA

Eletrônica Digital

Práticas de Laboratório

Volume 1

Rafael Manfrin Mendes

EDITORA SCIENZA

2023

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

M5226e Mendes, Rafael Manfrin

Eletrônica Digital - Práticas de Laboratório - Volume 1 / Rafael Manfrin Mendes : São Carlos, 2023.

500 p. il. cor

ISBN – 978-65-5668-111-5

DOI - http://dx.doi.org/10.26626/9786556681115.2023B0001

1. Eletrônica Digital. 2. Práticas de Laboratório. 3. Circuitos Lógicos. 4. Técnicas de implementação. 5. Programação de microcontrolador. I. Autor. II. Título.

CDD 600
Tecnologia - ciências aplicadas

Elaborado por Editora Scienza
Índice para catálogo sistemático:
1. tecnologia - ciências aplicadas. 600

Revisão, Editoração e E-book:
EDITORA SCIENZA

Rua Juca Sabino, 21 – São Carlos, SP | (16) 9 9285-3689
www.editorascienza.com.br | gustavo@editorascienza.com

Prefácio

Este livro se destina ao uso em aulas práticas de circuitos eletrônicos digitais e é apropriado para as disciplinas básicas de cursos técnicos, tecnologia ou bacharelado em engenharia elétrica. O livro contém exercícios para serem utilizados pelo estudante em 3 modalidades diferentes de aprendizado: simulador digital em computador, montagem de componentes eletrônicos físicos em placas de furos com fios e outros componentes e programação de eventos digitais em um microcontrolador Arduíno.

A simulação digital em computador pode ser feita por uma grande quantidade de programas, nas diversas plataformas disponíveis. Em ambiente Android, para celular ou tablet, existem vários aplicativos para essa função. Por exemplo o programa "EveryCircuit" de MuseMaze. Este livro utiliza o programa SmartSim disponível em "https://smartsim.org.uk/", por ser gratuito e disponível para plataformas Windows e Linux. É necessário o uso de um computador com o sistema operacional instalado. Recomenda-se a aprendizagem do uso do programa através do manual disponibilizado.

O aprendizado com os componentes eletrônicos físicos é realizado com o auxílio de uma placa de montagem sem solda e de outros componentes como fios de ligação, resistores, leds e chaves de contato momentâneo. Os componentes eletrônicos são da série TTL 74XX e da série CMOS 40XX. Para o uso dos circuitos da série TTL é necessário o uso de uma fonte de tensão estabilizada em 5 Volts. Para o uso dos circuitos da série CMOS é necessário o uso de uma fonte de tensão estabilizada com mínimo de 3 Volts e máximo de 18 Volts. Recomenda-se o uso da fonte de 5 Volts também para os circuitos da série CMOS. O uso de um multímetro digital é opcional.

Todos os exercícios propostos também são realizados em ambiente de programação do Arduíno. É necessário a instalação do programa de edição disponível em "https://www.arduino.cc/". Instale a versão de acordo com o sistema operacional do computador a ser utilizado. Também é necessário a utilização de uma placa básica de Arduíno. Recomenda-se o uso do Arduíno Uno Rev 3 ou Arduíno Mega 2560 Rev 3. Em alguns exercícios práticos é necessário a integração entre a placa do Arduíno e a placa de montagem sem solda com alguns componentes eletrônicos.

Os conteúdos de eletrônica digital que são abordados neste livro são: noções básicas de codificação numérica, noções básicas de álgebra booleana, portas lógicas digitais, circuitos básicos combinacionais e circuitos básicos sequenciais.

Os conteúdos de eletrônica digital que não são abordados neste livro são: memórias, unidade lógica aritmética, conversores analógicos para digitais, dispositivos programáveis (PAL, CPLD e FPGA), microprocessadores e microcontroladores.

Alguns apêndices foram adicionados para aprendizado complementar como dicas para confecção de um relatório, informações sobre circuitos integrados TTL e CMOS, e técnicas para uso das placas de furos para montagem física dos circuitos eletrônicos.

Recomenda-se que o professor orientador solicite que os alunos implementem, em simulador digital ou ambiente físico ou microcontrolador Arduíno, um projeto de final de curso que consiste de um circuito digital para aplicação de um evento prático.

Sumário

3.4.3. Experimento 1 _ _ _

Capítulo 1. Conversão de Numeração

1.1. Objetivos

A conversão de números para outra base de representação é investigada neste exercício. Os sistemas investigados são o binário, o octal e o hexadecimal. É analisado a conversão de base de entre uma representação e outra. O uso de calculadora de conversão é incentivada. Também é feita a representação com Arduíno.

1.2. Informação preliminar

O modo mais comum de representar números decimais é com o código binário sequencial, também conhecido como código posicional, código binário regular ou simplesmente código binário (é necessário ter cuidado com este tipo de designação porque todos os códigos que empregam apenas símbolos de dois valores são de fato binários). Mesmo que um código diferente seja normalmente usado para representar as teclas no teclado (por exemplo, no caso de computadores com um teclado PS / 2, um código chamado Conjunto de Códigos de Varredura 2 é empregado), o vetor que realmente entra no processador (para executar uma soma, por exemplo) normalmente emprega codificação binária sequencial. Esse tipo de codificação consiste em usar um vetor de bits em que cada bit tem um peso diferente, dado por $2^k – 1$, onde k é a posição do bit na palavra binária da direita para a esquerda. Consequentemente, se 8 bits (um byte) são usados para representar o número 5, então seu valor binário equivalente é “00000101” porque $0·2^7 + 0·2^6 + 0·2^5 + 0·2^4 + 0·2^3 + 1·2^2 + 0·2^1 + 1·2^0 = 5$. Por razões óbvias, o bit mais à esquerda chama-se MSB (“most significant bit” ou bit mais significativo), enquanto o mais à direita chama-se LSB (“least significant bit” ou bit menos significativo). Esta é a técnica de conversão de binário para decimal.

1.3. Procedimento

Execute as operações pedidas em uma folha de rascunho utilizando o algoritmo apresentado nos livros de teoria e escreva o resultado no quadro em branco correspondente. Compare os valores dos exercícios de com os valores realizados pela calculadora. Utilize uma calculadora científica com opção de conversão de base. As calculadoras mais comuns para esta tarefa são as da marca Truly com os modelos SC-103 e SC-107. A calculadora Casio FX-570MS também executa essa tarefa. A calculadora FX-82MS executa esta tarefa somente acessando o menu escondido. Outra opção é utilizar calculadoras para Android, como a calculadora CalcTastic.

1. Conversão de binário para decimal #1

Escreva os números decimais correspondentes às seguintes representações binárias sem sinal (o espaço em branco é apenas para facilitar a leitura):

Binário	Decimal
“0000 1111”	
“0000 1111 0010”	
“1000 1000 0001 0001”	

2. Conversão de binário para decimal #2

Escreva os números decimais correspondentes às seguintes representações binárias sem sinal:

Binário	Decimal
"1000 1001"	
"1000 1111 0000"	
"0010 1000 0000 0001"	

3. Conversão de binário para hexadecimal #1

Escreva os números hexadecimais correspondentes às seguintes representações binárias sem sinal:

Binário	Hexadecimal
"0000 1110"	
"0000 1111 0010"	
"1000 1010 0001 0011"	

4. Conversão de binário para hexadecimal #2

Escreva os números hexadecimais correspondentes às seguintes representações binárias sem sinal:

Binário	Hexadecimal
"0100 1110"	
"0011 1111 0010"	
"1111 1010 0001 1001"	

5. Conversão de decimal para binário #1

Determine o número mínimo de bits necessários para representar os seguintes números decimais e escreva os vetores binários correspondentes:

Decimal	Número de bits	Binário
"15"		
"16"		
"511"		
"12.345"		
"49.999"		

6. Conversão de decimal para binário #2

Determine o número mínimo de bits necessários para representar os seguintes números decimais e escreva os vetores binários correspondentes:

Decimal	Número de bits	Binário
"63"		
"64"		
"512"		
"2007"		
"99.999"		

7. Conversão de decimal para hexadecimal # 1

Usando 4 (quatro) dígitos hexadecimais, escreva os números hexadecimais correspondentes aos seguintes números decimais:

Decimal	Hexadecimal
"63"	
"64"	
"512"	
"2007"	
"49.999"	

8. Conversão de decimal para hexadecimal #2

Usando o número mínimo possível de dígitos hexadecimais, escreva os números hexadecimais correspondentes aos seguintes números decimais:

Decimal	Número de dígitos	Hexadecimal
"255"		
"256"		
"4096"		
"12.345"		

9. Conversão de hexadecimal para binário # 1

Usando N = 16 bits, escreva as cadeias binárias correspondentes aos seguintes números hexadecimais:

Hexadecimal	Binário
"AA"	
"99C"	
"000F"	
"FF7F"	

10. Conversão de hexadecimal para binário # 2

Usando o número mínimo possível de bits, escreva as cadeias binárias correspondentes aos seguintes números hexadecimais:

Hexadecimal	Binário
"D"	
"29C"	
"F000"	
"13FF"	

11. Conversão de hexadecimal para decimal # 1

Converta os seguintes números hexadecimais em decimais:

Hexadecimal	Decimal
"D"	
"99C"	
"000F"	
"1FF7F"	

12. Conversão de hexadecimal para decimal # 2

Converta os seguintes números hexadecimais em decimais:

Hexadecimal	Decimal
"AA"	
"990"	
"7001"	
"FF007"	

13. Conversão de octal para decimal

Converta os seguintes números octais em decimais:

Octal	Decimal
“3”	
“77”	
“0011”	
“2222”	

14. Conversão de decimal para octal

Escreva a representação octal para as seguintes números decimais:

Decimal	Octal
“3”	
“77”	
“0011”	
“2222”	

1.4. Programa para o Arduíno

Execute o programa para o Arduíno e compare os resultados com os valores que você encontrou.

```
// Conversão de numeração

unsigned long a;

// ----------------------------------------------------
// Imprime os bits 0 e 1 de todo o vetor binário
// ----------------------------------------------------
void printBits(unsigned long x, unsigned long m1){
  byte i = 0;
  for(unsigned long mascara = m1; mascara; mascara >>= 1){
    i++;
    if(mascara & x)
    Serial.print(“1”);
    else
    Serial.print(“0”);
    if (i == 4){
      Serial.print(“ “);
```

```
      i = 0;
    }
  }
}
// ----------------------------------------------------
// Imprime binários 8 ou 16 ou 32 bits
// ----------------------------------------------------
void printbin(unsigned long x, byte i){
  Serial.print("Binário = ");
  Serial.print(x, BIN);
  Serial.print(" ( ");
  if (i == 8) printBits(x, 0x80);
  if (i == 16) printBits(x,0x8000);
  if (i == 32) printBits(x,0x80000000);
  Serial.println(")");
}
// ----------------------------------------------------
// Imprime decimal
// ----------------------------------------------------
void printdec(unsigned long x){
  Serial.print("Decimal = ");
  Serial.println(x);
}
// ----------------------------------------------------
// Imprime hexadecimal
// ----------------------------------------------------
void printhex(unsigned long x){
    Serial.print("Hexadecimal = ");
    Serial.println(x, HEX);
}
// ----------------------------------------------------
// Imprime octal
// ----------------------------------------------------
void printoct(unsigned long x){
    Serial.print("Octal = ");
    Serial.println(x, OCT);
}
// ----------------------------------------------------
// Converte de binário 8 ou 16 ou 32 bits para decimal
// ----------------------------------------------------
void bin_dec(unsigned long x, byte i){
  printbin(x,i);
  printdec(x);
```

```
  Serial.println("");
}
// ----------------------------------------------------
// Converte de binário 8 ou 16 ou 32 bits para hexadecimal
// ----------------------------------------------------
void bin_hex(unsigned long x, byte i){
  printbin(x,i);
  printhex(x);
  Serial.println("");
}
// ----------------------------------------------------
// Converte de decimal para binário 8 ou 16 ou 32 bits
// ----------------------------------------------------
void dec_bin(unsigned long x, byte i){
  printdec(x);
  printbin(x,i);
  Serial.println("");
}
// ----------------------------------------------------
// Converte de hexadecimal para binário 8 ou 16 ou 32 bits
// ----------------------------------------------------
void hex_bin(unsigned long x, byte i){
  printhex(x);
  printbin(x,i);
  Serial.println("");
}
// ----------------------------------------------------
// Converte de decimal para hexadecimal
// ----------------------------------------------------
void dec_hex(unsigned long x){
  printdec(x);
  printhex(x);
  Serial.println("");
}
// ----------------------------------------------------
// Converte de hexadecimal para decimal
// ----------------------------------------------------
void hex_dec(unsigned long x){
  printhex(x);
  printdec(x);
  Serial.println("");
}
// ----------------------------------------------------
```

```
// Converte de decimal para octal
// ----------------------------------------------------

void dec_oct(unsigned long x){
  printdec(x);
  printoct(x);
  Serial.println("");
}
// ----------------------------------------------------
// Converte de octal para decimal
// ----------------------------------------------------
void oct_dec(unsigned long x){
  printoct(x);
  printdec(x);
  Serial.println("");
}

// ----------------------------------------------------
// Programa do principal
// ----------------------------------------------------
void setup(){
  Serial.begin(9600);

  Serial.println("1. Conversão de binário para decimal #1");
  Serial.println("");
  a = B00001111; bin_dec(a,8);
  a = B00000000; a = a << 8; a = a | B11110010; bin_dec(a,16);
  a = B10001000; a = a << 8; a = a | B00010001; bin_dec(a,16);

  Serial.println("2. Conversão de binário para decimal #2");
  Serial.println("");
  a = B10001001; bin_dec(a,8);
  a = B00001000; a = a << 8; a = a | B11110000; bin_dec(a,16);
  a = B00101000; a = a << 8; a = a | B00000001; bin_dec(a,16);

  Serial.println("3. Conversão de binário para hexadecimal #1");
  Serial.println("");
  a = B00001110; bin_hex(a,8);
  a = B00000000; a = a << 8; a = a | B11110010; bin_hex(a,16);
  a = B10001010; a = a << 8; a = a | B00010011; bin_hex(a,16);

  Serial.println("4. Conversão de binário para hexadecimal #2");
  Serial.println("");
  a = B01001110; bin_hex(a,8);
```

```
a = B00000011; a = a << 8; a = a | B11110010; bin_hex(a,16);
a = B11111010; a = a << 8; a = a | B00011001; bin_hex(a,16);

Serial.println("5. Conversão de decimal para binário #1");
Serial.println("");
a = 15; dec_bin(a,8);
a = 16; dec_bin(a,8);
a = 511; dec_bin(a,16);
a = 12345; dec_bin(a,16);
a = 49999; dec_bin(a,16);

Serial.println("6. Conversão de decimal para binário #2");
Serial.println("");
a = 63; dec_bin(a,8);
a = 64; dec_bin(a,8);
a = 512; dec_bin(a,16);
a = 2007; dec_bin(a,16);
a = 99999; dec_bin(a,32);

Serial.println("7. Conversão de decimal para hexadecimal #1");
Serial.println("");
a = 63; dec_hex(a);
a = 64; dec_hex(a);
a = 512; dec_hex(a);
a = 2007; dec_hex(a);
a = 99999; dec_hex(a);

Serial.println("8. Conversão de decimal para hexadecimal #2");
Serial.println("");
a = 255; dec_hex(a);
a = 256; dec_hex(a);
a = 4096; dec_hex(a);
a = 12345; dec_hex(a);

Serial.println("9. Conversão de hexadecimal para binário #1");
Serial.println("");
a = 0xAA; hex_bin(a,16);
a = 0x99C; hex_bin(a,16);
a = 0x000F; hex_bin(a,16);
a = 0xFF7F; hex_bin(a,16);

Serial.println("10. Conversão de hexadecimal para binário #2");
Serial.println("");
a = 0xD; hex_bin(a,16);
```

```
  a = 0x29C; hex_bin(a,16);
  a = 0xF000; hex_bin(a,16);
  a = 0x13FF; hex_bin(a,16);

  Serial.println("11. Conversão de hexadecimal para decimal #1");
  Serial.println("");
  a = 0xD; hex_dec(a);
  a = 0x99C; hex_dec(a);
  a = 0x000F; hex_dec(a);
  a = 0x1FF7F; hex_dec(a);

  Serial.println("12. Conversão de hexadecimal para decimal #2");
  Serial.println("");
  a = 0xAA; hex_dec(a);
  a = 0x990; hex_dec(a);
  a = 0x7001; hex_dec(a);
  a = 0xFF007; hex_dec(a);

  Serial.println("13. Conversão de octal para decimal");
  Serial.println("");
  a = 03; oct_dec(a);
  a = 077; oct_dec(a);
  a = 00011; oct_dec(a);
  a = 02222; oct_dec(a);

  Serial.println("14. Conversão de decimal para octal");
  Serial.println("");
  a = 3; dec_oct(a);
  a = 77; dec_oct(a);
  a = 111; dec_oct(a);
  a = 2222; dec_oct(a);
}

void loop(){
  // nada a fazer aqui.
}

// fim do programa.
```

Capítulo 2. Aritmética binária

2.1. Objetivos

Os humanos estão acostumados a fazer operações aritméticas com números decimais, enquanto os computadores executam operações aritméticas semelhantes, mas usam o sistema binário de ‘0’s e ‘1’s. O objetivo deste exercício é mostrar como o último ocorre. A análise inclui valores sem sinal e com sinal, dos tipos inteiro e valor real. Como as operações de deslocamento também podem implementar certas funções aritméticas, elas também são incluídas neste exercício.

2.2. Informação preliminar

Operações matemáticas básicas com números binários funcionam de forma semelhante ao sistema decimal. No entanto, existem algumas regras específicas para o sistema binário. Vamos olhar para cada um deles individualmente. Existem 3 regras básicas para adicionar números binários: 0 + 0 = 0, 0 + 1 = 1, 1 + 1 = 10. Se a soma de 2 bits for maior que 1, precisamos mudar uma coluna à esquerda. Em sistema decimal, 1 + 1 = 2. A notação binária de 2 é 10 (1*2^1+0*2^0). Então, mantemos 0 na coluna 1 e deslocamos 1 para a coluna 2. A multiplicação em binário é exatamente como é em decimal, ou seja, multiplique os números da direita para a esquerda e multiplique cada dígito de um número para cada dígito do outro número, então some tudo. As 3 regras básicas de multiplicação binária também são semelhantes às decimais: 1 * 1 = 1, 1 * 0 = 0 * 1 = 0, 0 * 0 = 0. Além disso, lembre-se de que, para cada deslocamento à esquerda do dígito do multiplicador, um zero extra precisa ser anexado ao produto. Isso é semelhante ao sistema decimal também. Antes de tentar a subtração, precisamos entender como os números negativos são representados no binário. Seja qual for o sistema utilizado (isto é, 4 bits, 8 bits, 16 bits, etc.), o número com sinal deve ter todos o mesmo número de bits. 0s são usados para preencher os bits vazios. Vamos usar 8 bits para este tutorial. Existem 3 padrões básicos para anotar números negativos. Magnitude com sinal, um bit extra é adicionado à esquerda do número para anotar seu sinal. 0 indica (+) e 1 indica (-). No complemento de 1 os números positivos são representados exatamente como números binários regulares. Os números negativos são representados simplesmente invertendo o bit, ou seja, 0 se torna 1 e 1 se torna 0. Na subtração usando o complemento de 1, o número a ser subtraído deve ser negado usando o complemento de 1 e depois adicionado (não subtraído) ao outro número. Como o número com sinal deve ter o mesmo número de bits, qualquer bit de “estouro” deve ser adicionado ao restante do resultado. No complemento de 2, um número binário de n bits é definido como o complemento em relação a 2^n ou, simplesmente, o resultado da subtração do número de 2n. Neste método, um número negativo é anotado, primeiro determinando o complemento de 1 do número positivo e, em seguida, adicionando 1 a ele. Ao executar operações binárias, é importante conhecer a convenção que está sendo usada para executar a operação seguindo as regras aplicáveis. A divisão binária é semelhante à divisão decimal. A única diferença é que no sistema decimal, uma vez que estamos dividindo números tradicionais, o dividendo (ou parte dele) pode ser 0, 1 ou mais de 1 vez do divisor. No entanto, em binário, só pode ser 0 ou 1 vezes, isto é, o dividendo (ou parte dele) é >= ou < que o divisor.

2.3. Procedimento

Execute as operações pedidas utilizando o algoritmo apresentado nos livros de teoria. Preencha os quadrados com os valores correspondentes de 0 (zero) ou 1 (um). Anote o valor decimal correspondente. Compare os valores dos exercícios de com os valores realizados pela calculadora.

Utilize uma calculadora científica com opção de conversão de base. As calculadoras mais comuns para esta tarefa são as da marca Truly com os modelos SC-103 e SC-107. A calculadora Casio FX-570MS também executa essa tarefa. A calculadora FX-82MS executa esta tarefa somente acessando o menu escondido. Outra opção é utilizar calculadoras para Android, como a calculadora CalcTastic.

1. Adição sem sinal # 1

Considere um somador com entradas de 5 bits sem signal e saída de 5 bits.

a) Determine o intervalo (decimal) de cada entrada e saída.

	b4	b3	b2	b1	b0	decimal		b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	_____		☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	_____	+	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	_____	=	☐	☐	☐	☐	☐	_____

b) Representando todos os sinais usando código binário regular, calcule 16 + 15 e verifique se o resultado é válido.

	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	_____

c) Repita a parte (b) acima para 16 + 16.

	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	_____

2. Adição sem sinal # 2

Considere um somador com entradas de 6 bits sem signal e saída de 6 bits.

a) Determine o intervalo (decimal) de cada entrada e saída.

	b5	b4	b3	b2	b1	b0	decimal		b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____		☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____	+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____	=	☐	☐	☐	☐	☐	☐	_____

b) Representando todos os sinais usando código binário regular, calcule 16 + 15 e verifique se o resultado é válido.

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

c) Repita a parte (b) acima para 16 + 16.

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

3. Adição sem sinal # 3

Dados os valores sem sinal a = “111101”, b = “000001” e c = “100001”, e sabendo que x, y e z também são números de 6 bits sem sinal, calcule e verifique o resultado de:

a) x = a + b

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

b) y = a + c

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

c) z = b + c

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

4. Adição com sinal # 1

Considere um somador com entradas de 5 bits com sinal e saída de 5 bits.

a) Determine o intervalo (decimal) de cada entrada e saída.

	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	_____

	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	_____

b) Usando vetores binários, com números negativos no formato de complemento de 2, calcule - 8 - 8 e verifique se o resultado é válido.

	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	_____

c) Repita a parte (b) acima para - 8 - 9.

	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	_____

d) Repita a parte (b) acima para + 8 - 9.

	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	_____

e) Repita a parte (b) acima para + 8 + 8.

	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	_____

5. Adição com sinal # 2

Considere um somador com entradas de 6 bits com sinal e saída de 6 bits.

a) Determine o intervalo (decimal) de cada entrada e saída.

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

b) Usando vetores binários, com números negativos no formato de complemento de 2, calcule - 8 - 8 e verifique se o resultado é válido.

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

c) Repita a parte (b) acima para - 8 - 9.

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

d) Repita a parte (b) acima para + 8 - 9.

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

e) Repita a parte (b) acima para + 8 + 8.

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

6. Adição com sinal # 3

Dados os valores com sinal a = “111101”, b = “000001” e c = “100001”, e sabendo que x, y e z também são valores de 6 bits com sinal, calcule e verifique o resultado de:

a) x = a + b

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

b) y = a + c

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

c) z = b + c

	b5	b4	b3	b2	b1	b0	decimal
	☐	☐	☐	☐	☐	☐	_____
+	☐	☐	☐	☐	☐	☐	_____
=	☐	☐	☐	☐	☐	☐	_____

7. Deslocamento Lógico #1

Escreva os vetores resultantes quando as operações de deslocamento lógico abaixo forem executadas. SRL n = deslocamento lógico para direita de n bits (shift right logical n positions), SLL n = deslocamento lógico para esquerda de n bits (shift left logical n positions).

a) “11010” SRL 2

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

b) “01011” SRL −3

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

c) “10010” SLL 2

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

d) “01011” SLL 3

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

8. Deslocamento Lógico # 2

Escreva os vetores resultantes quando as operações de deslocamento lógico abaixo forem executadas.

a) “111100” SRL 2

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

b) “010000” SRL 3

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

c) “111100” SLL −2

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

d) “010011” SLL 3

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

9. Deslocamento Aritmético # 1

Escreva os vetores resultantes quando as operações de deslocamento aritmético abaixo forem executadas. SRA n = deslocamento aritmético para direita de n bits (shift right arithmetic n positions), SRA n = deslocamento aritmético para esquerda de n bits (shift left arithmetic n positions).

a) “11010” SRA 2

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

b) “01011” SRA −3

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

c) “10010” SLA 2

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

d) “01011” SLA 3

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

10. Deslocamento Aritmético # 2

Escreva os vetores resultantes quando as operações de mudança aritmética abaixo forem executadas.

a) “111100” SRA 2

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

b) “010000” SRA 3

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

c) “111100” SLA −2

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

d) “010011” SLA 1

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

11. Deslocamento Circular # 1

Escreva os vetores resultantes quando as operações de deslocamento circular abaixo forem executadas. ROR n = deslocamento circular para direita de n bits (rotate right n positions), ROL n = deslocamento circular para esquerda de n bits (rotate left n positions)

a) “11010” ROL 1

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

b) “01011” ROL −3

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

c) “10010” ROR 2

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

d) “01011” ROR 3

original ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐

12. Deslocamento Circular # 2

Escreva os vetores resultantes quando as operações de deslocamento circular abaixo forem executadas.

a) “111100” ROL 2

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

b) “010000” ROL 3

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

c) “111100” ROR-2

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

d) “010011” ROR 3

original ☐ ☐ ☐ ☐ ☐ ☐ ☐ deslocado ☐ ☐ ☐ ☐ ☐ ☐ ☐

2.4. Programa para o Arduíno

Execute o programa para o Arduíno e compare os resultados com os valores que você encontrou.

```
// Aritmética binária

unsigned int a; unsigned int b; unsigned int c;
unsigned int x; unsigned int y; unsigned int z;
int a1; int b1; int c1;
int x1; int y1; int z1;
byte s; byte td; byte tx; byte i;
byte pos[9] = {0x00,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};

// ----------------------------------------------------
// Imprime os bits 0 e 1 de todo o vetor binário
// ----------------------------------------------------
void printBits(unsigned int x, unsigned int m1){
  for(unsigned int mascara = m1; mascara; mascara >>= 1){
    if(mascara & x)
      Serial.print("1 ");
    else
      Serial.print("0 ");
  }
}

// ----------------------------------------------------
// Deslocamento lógico para direita
// ----------------------------------------------------
void srl(byte b, byte t, int q){ // byte, tamanho, quantidade
  if(q<0) sll(b,t,-q);
  else s = b >> q;
}

// ----------------------------------------------------
// Deslocamento lógico para esquerda
// ----------------------------------------------------
void sll(byte b, byte t, int q){ // byte, tamanho, quantidade
  if(q<0) srl(b,t,-q);
  else {
    b = b << (8-t);
    s = b << q;
    s = s >> (8-t);
  }
}
```

```
// ----------------------------------------------------
// Deslocamento aritmético para direita
// ----------------------------------------------------
void sra(byte b, byte t, int q){ // byte, tamanho, quantidade
  if(q<0) sla(b,t,-q);
  else {
    s = b;
    td = B00000001 << (t-1);
    for (i=0; i<q; i++){
      s = s >> 1;
      s = s | td;
    }
  }
}

// ----------------------------------------------------
// Deslocamento aritmético para esquerda
// ----------------------------------------------------
void sla(byte b, byte t, int q){ // byte, tamanho, quantidade
  if(q<0) sra(b,t,-q);
  else {
    s = b << (8-t);
    td = B00000001 << (8-t);
    for (i=0; i<q; i++){
      s = s << 1;
      s = s | td;
    }
    s = s >> (8-t);
  }
}

// ----------------------------------------------------
// Rotação de bits para direita
// ----------------------------------------------------
void ror(byte b, byte t, int q){ // byte, tamanho, quantidade
  if(q<0) rol(b,t,-q);
  else {
    s = b;
    for (i=0; i<q; i++){
      td = s & B00000001;
      td = td << (t-1);
      s = s >> 1;
      s = s | td;
    }
  }
}
```

```
// ----------------------------------------------------
// Rotação de bits para esquerda
// ----------------------------------------------------
void rol(byte b, byte t, int q){ // byte, tamanho, quantidade
  if(q<0) ror(b,t,-q);
  else {
    s = b << (8-t);
    for (i=0; i<q; i++){
      td = s & B10000000;
      td = td >> (t-1);
      s = s << 1;
      s = s | td;
    }
    s = s >> (8-t);
  }
}

// ----------------------------------------------------
// Mostra soma a + b = x
// ----------------------------------------------------
void mostrasoma(int a, int b, int x, byte t){
  printBits(a, t);
  Serial.print("+ ");
  printBits(b, t);
  Serial.print("= ");
  printBits(x, t);
  Serial.print(" ( ");
  Serial.print(a);
  Serial.print(" + ");
  Serial.print(b);
  Serial.print(" = ");
  Serial.print(x);
  Serial.println(" )");
}

// ----------------------------------------------------
// Mostra valor
// ----------------------------------------------------
void mostravalor(int a, byte t){
  Serial.print("Valor = ");
  printBits(a,t);
  Serial.print("( ");
  Serial.print(a);
  Serial.println(" )");
}
```

```
// ----------------------------------------------------
// Mostra byte
// ----------------------------------------------------
void mostrabyte(int a, byte t){
  printBits(a,t);
  Serial.print("( ");
  Serial.print(a);
  Serial.print(" )");
}
// ----------------------------------------------------
// Deslocamento
// ----------------------------------------------------

void desloca(int a, byte t, byte tipo, int q){
  printBits(a, pos[t]);
  Serial.print("-> ");
  if(tipo == 1){ Serial.print("SLL "); Serial.print(q); sll(a,t,q); }
  if(tipo == 2){ Serial.print("SRL "); Serial.print(q); srl(a,t,q); }
  if(tipo == 3){ Serial.print("SLA "); Serial.print(q); sla(a,t,q); }
  if(tipo == 4){ Serial.print("SRA "); Serial.print(q); sra(a,t,q); }
  if(tipo == 5){ Serial.print("ROL "); Serial.print(q); rol(a,t,q); }
  if(tipo == 6){ Serial.print("ROR "); Serial.print(q); ror(a,t,q); }
  Serial.print(" = ");
  printBits(s, pos[t]);
  Serial.println("");
}

// ----------------------------------------------------
// Programa principal
// ----------------------------------------------------
void setup(){
  Serial.begin(9600);

  Serial.println("1. Adição sem sinal #1");
  a = 16; b = 15; x = a + b; x = x & B00011111; mostrasoma(a,b,x,0x10);
  a = 16; b = 16; x = a + b; x = x & B00011111; mostrasoma(a,b,x,0x10);

  Serial.println(""); Serial.println("2. Adição sem sinal #2");
  a = 16; b = 15; x = a + b; x = x & B00111111; mostrasoma(a,b,x,0x20);
  a = 16; b = 16; x = a + b; x = x & B00111111; mostrasoma(a,b,x,0x20);
```

```
Serial.println(""); Serial.println("3. Adição sem sinal #3");
a = B111101; mostravalor(a, 0x20);
b = B000001; mostravalor(b, 0x20);
c = B100001; mostravalor(c, 0x20);
x = a + b; x = x & B00111111; mostrasoma(a,b,x,0x20);
y = a + c; y = y & B00111111; mostrasoma(a,c,y,0x20);
z = b + c; z = z & B00111111; mostrasoma(b,c,z,0x20);

Serial.println(""); Serial.println("4. Adição com sinal #1");
a1 = -8; b1 = -8; x1 = a1 + b1; mostrasoma(a1,b1,x1,0x10);
a1 = -8; b1 = -9; x1 = a1 + b1; mostrasoma(a1,b1,x1,0x10);
a1 = 8; b1 = -9; x1 = a1 + b1; mostrasoma(a1,b1,x1,0x10);
a1 = 8; b1 = 8; x1 = a1 + b1; mostrasoma(a1,b1,x1,0x10);

Serial.println(""); Serial.println("5. Adição com sinal #2");
a1 = -8; b1 = -8; x1 = a1 + b1; mostrasoma(a1,b1,x1,0x20);
a1 = -8; b1 = -9; x1 = a1 + b1; mostrasoma(a1,b1,x1,0x20);
a1 = 8; b1 = -9; x1 = a1 + b1; mostrasoma(a1,b1,x1,0x20);
a1 = 8; b1 = 8; x1 = a1 + b1; mostrasoma(a1,b1,x1,0x20);

Serial.println(""); Serial.println("6. Adição com sinal #3");
a1 = B11111111; a1 = a1 << 8; a1 = a1 | B11111101; mostravalor(a1,
0x20);
b1 = B00000000; b1 = b1 << 8; b1 = b1 | B00000001; mostravalor(b1,
0x20);
c1 = B11111111; c1 = c1 << 8; c1 = c1 | B11100001; mostravalor(c1,
0x20);
x1 = a1 + b1; mostrasoma(a1,b1,x1,0x20);
y1 = a1 + c1; mostrasoma(a1,c1,y1,0x20);
z1 = b1 + c1; mostrasoma(b1,c1,z1,0x20);

Serial.println(""); Serial.println("7. Deslocamento lógico #1");
a = B11010; desloca(a,5,2,2); // srl 2
a = B01011; desloca(a,5,2,-3); // srl -3
a = B10010; desloca(a,5,1,2); // sll 2
a = B01011; desloca(a,5,1,3); // sll 3

Serial.println(""); Serial.println("8. Deslocamento lógico #2");
a = B111100; desloca(a,6,2,2); // srl 2
a = B010000; desloca(a,6,2,3); // srl 3
a = B111100; desloca(a,6,1,-2); // sll -2
a = B010011; desloca(a,6,1,3); // sll 3
```

```
  Serial.println(“”); Serial.println(“9. Deslocamento aritmético #1”);
  a = B11010; desloca(a,5,4,2); // sra 2
  a = B01011; desloca(a,5,4,-3); // sra -3
  a = B10010; desloca(a,5,3,2); // sla 2
  a = B01011; desloca(a,5,3,3); // sla 3

  Serial.println(“”); Serial.println(“10. Deslocamento aritmético #2”);
  a = B111100; desloca(a,6,4,2); // sra 2
  a = B010000; desloca(a,6,4,3); // sra 3
  a = B111100; desloca(a,6,3,-2); // sla -2
  a = B010011; desloca(a,6,3,1); // sla 1

  Serial.println(“”); Serial.println(“11. Deslocamento circular #1”);
  a = B11010; desloca(a,5,5,1); // rol 1
  a = B01011; desloca(a,5,5,-3); // rol -3
  a = B10010; desloca(a,5,6,2); // ror 2
  a = B01011; desloca(a,5,6,3); // ror 3

  Serial.println(“”); Serial.println(“12. Deslocamento circular #2”);
  a = B111100; desloca(a,6,5,2); // rol 2
  a = B010000; desloca(a,6,5,3); // rol 3
  a = B111100; desloca(a,6,6,-2); // ror -2
  a = B010011; desloca(a,6,6,3); // ror 3
}
  void loop(){
    // Nada a fazer aqui!
}
```

Capítulo 3. Portas Lógicas

3.1. Objetivos

Conhecer o funcionamento das portas lógicas básicas é de primeira importância para a compreensão dos circuitos lógicos mais complexos. Neste exercício será feito o levantamento da tabela verdade do funcionamento das portas E (AND), NE (NAND), OU (OR), NOU (NOR), Não (Inv), OU-EXC (EX-OR) e NOU-EXC (EX-NOR). Será estudado também algumas possibilidades de conexão para obtenção de outras portas com mais entradas e uso de determinadas portas no lugar de outras.

3.2. Informação preliminar

As portas lógicas são componentes básicos da eletrônica digital. Elas são usadas para criar circuitos digitais e até mesmo circuitos integrados complexos. Em eletrônica digital apenas dois níveis são permitidos, “0” e “1”. Zero representa tensão de 0 V, enquanto que “1” representa uma tensão de 5 V no padrão TTL. Assim as portas lógicas são capazes de realizar diversas operações matemáticas, para desenvolvimento da lógica digital.

As portas lógicas básicas são:

NÃO ou INVERSORA (NOT): Como o próprio nome já sugere, o inversor irá inverter o estado da entrada. Se você entrar o número “0” em um circuito inversor, você obterá na saída o número “1”, e vice-versa. A porta inversora é mais conhecida como NOT e sua saída é Y =/A. Os circuitos integrados mais comum são:

Nº de entradas por porta	Nº de portas por chip	Família TTL	Família CMOS
1	6	7404	4069

E (AND): Uma porta lógica E (AND) realiza uma operação lógica “E”. Ela possui pelo menos duas entradas. Por isso, se A e B são suas entradas, na saída teremos o resultado de A x B (também representado como A · B). Os circuitos integrados mais comum são:

Nº de entradas por porta	Nº de portas por chip	Família TTL	Família CMOS
2	4	7408	4081
3	3	7411	4073
4	2	7421	4082

OU (OR): A porta lógica OU (OR) realiza uma operação lógica “OU”. Ela possui pelo menos duas entradas. Por isso, se A e B são suas entradas, na saída teremos o resultado de A + B. Os circuitos integrados mais comum são:

Nº de entradas por porta	Nº de portas por chip	Família TTL	Família CMOS
2	4	7432	4071
3	3	-	4075
4	2	-	4072

Portas Lógicas Derivadas:

Não-E (NAND): Esta porta nada mais é do que uma porta E (AND) com um inversor acoplado na saída. Por isso, sua saída é o oposto da E (AND). Você pode construir uma porta Não-E (NAND) conectando uma porta E (AND) a um inversor. Os circuitos integrados mais comum são:

Nº de entradas por porta	Nº de portas por chip	Família TTL	Família CMOS
2	4	7400	4011
3	3	7410	4013
4	2	7420	4012
8	1	7430	4068
12	1	74134	-
13	1	74133	-

Não-OU (NOR): Esta é uma porta OU (OR) com um inversor acoplado. Por isso, sua saída é o oposto da porta OU (OR). Você pode construir uma porta Não-OU (NOR) conectando uma porta OU (OR) a um inversor. Os circuitos integrados mais comum são:

Nº de entradas por porta	Nº de portas por chip	Família TTL	Família CMOS
2	4	7402	4001
3	3	7427	4025
4	2	7425	4002
5	2	74260	-
8	1	-	4078

OU-Exclusivo (XOR): A porta lógica OU-Exclusivo (XOR) compara dois valores de entrada e se eles forem diferentes a saída será “1”. Os circuitos integrados mais comum são:

Nº de entradas por porta	Nº de portas por chip	Família TTL	Família CMOS
2	4	7486	4070

Não-OU-Exclusivo (XNOR): Esta é uma porta OU-Exclusivo (XOR) com sua saída invertida. Dessa forma, sua saída será igual a “1” quando suas entradas possuírem o mesmo valor e “0” quando elas forem diferentes. Os circuitos integrados mais comum são:

Nº de entradas por porta	Nº de portas por chip	Família TTL	Família CMOS
2	4	74266	4077

3.3. Exame da porta E (AND)

3.3.1. Objetivos

1. Conhecer a lógica digital da porta E (AND) e verificar sua operação lógica,
2. Obter uma porta E (AND) de 3 entradas e 4 entradas usando portas de 2 entradas.

3.3.2. Informação preliminar

1. Em uma operação lógica, o nível ALTO (HIGH, H) é representado pelo número “1” e o nível BAIXO (LOW, L) é representado pelo número “0”.
2. Para CIs TTL, H (1) representa níveis de tensão entre 2,4V e 5V. Da mesma forma, L (0) representa níveis de tensão entre 0 V e 0,4 V.
3. Para CIs CMOS, a tensão equivalente de H (1) é aproximadamente a tensão de alimentação e L (0) representa níveis de tensão entre 0V e 0,5 V.
4. A porta E (AND) é uma porta de multiplicação. Tem pelo menos duas entradas. Pelo menos uma entrada “0” faz a saída “0”. A saída é “1” somente quando todas as entradas são “1” (Tabela 3.1).
5. A saída de uma porta E (AND) de 2 entradas (figura 3.1) é Y = A. B (tabela 3.1).
6. A saída de uma porta E (AND) de 3 entradas (figura 3.2) é Y = A.B.C (tabela 3.2).
7. Como pode ser visto na figura 3.3, tanto o CI TTL 7408 quanto o CI CMOS 4081 contêm 4 portas E (AND), mas a disposição interna das portas é diferente.

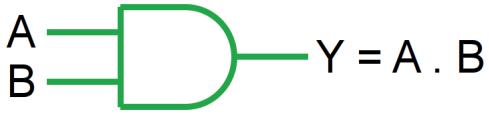


Figura 3.1 – Porta E (AND) de 2 entradas.

A	B	Y = A . B
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 3.1 – Tabela verdade de uma porta E (AND) de 2 entradas.

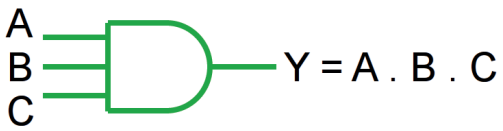


Figura 3.2 – Porta E (AND) de 3 entradas.

A	B	C	Y = A . B . C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tabela 3.2 – Tabela verdade de uma porta E (AND) de 3 entradas.

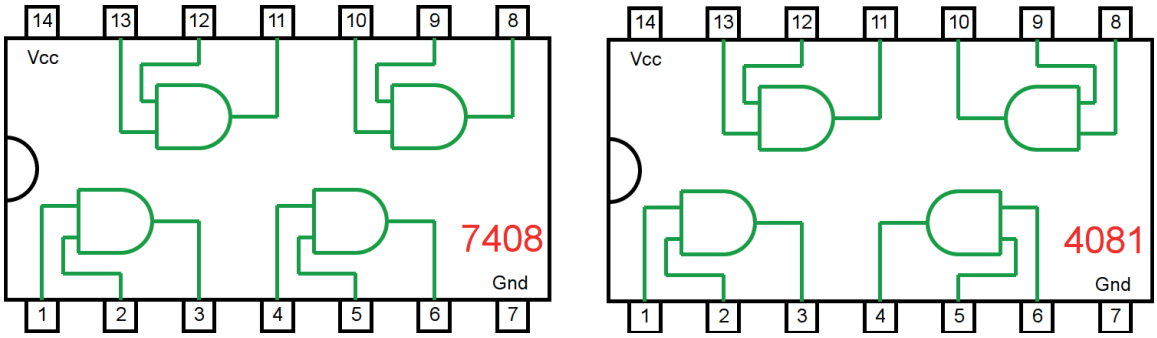


Figura 3.3 – Circuito Integrado TTL e CMOS.

3.3.3. Experimento 1

Obter a tabela verdade para uma porta E de duas entradas.

Esquemas:

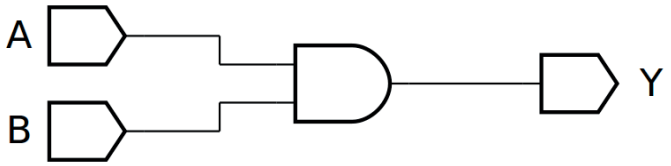


Figura 3.4 – Diagrama esquemático.

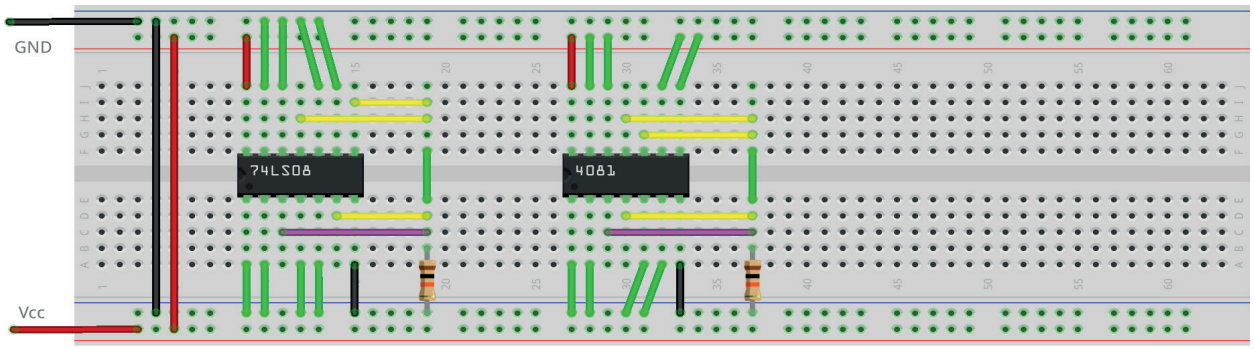


Figura 3.5 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.4, conforme mostrado na figura 3.5, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo. Não conecte os fios amarelos!
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.

   Os procedimentos a seguir se referem ao teste da porta 1-2-3.
3. Conecte o cabo vermelho na fonte de energia.
4. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.3a, na coluna correspondente ao CI e linha corresponde a A =0 e B=0.
5. Desconecte o cabo vermelho na fonte de energia.
6. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
7. Conecte o cabo vermelho na fonte de energia.

8. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.3a, na coluna correspondente ao CI e linha corresponde a A =0 e B=1.
9. Desconecte o cabo vermelho na fonte de energia.
10. Mude o fio verde do terminal 1 do CI do barramento de energia (vermelho, 1) para o barramento de terra (preto, 0). Mude o fio verde do terminal 2 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
11. Conecte o cabo vermelho na fonte de energia.
12. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.3a, na coluna correspondente ao CI e linha corresponde a A =1 e B=0.
13. Desconecte o cabo vermelho na fonte de energia.
14. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
15. Conecte o cabo vermelho na fonte de energia.
16. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.3a, na coluna correspondente ao CI e linha corresponde a A =1 e B=1.
17. Desconecte o cabo vermelho na fonte de energia.
18. Preencha a 3ª coluna da tabela 3.3a com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.

    Os procedimentos a seguir são opcionais e se referem aos testes das outras portas e outro CI.

19. Remova o fio roxo do pino 3 do CI e conecte no pino 6.
20. Repita os procedimentos semelhantes de 3 a 18 para a porta 4-5-6 e anote os valores na tabela 3.3b. Onde for terminal 1 use terminal 4. Onde for terminal 2 use terminal 5.
21. Remova o fio roxo do pino 6 do CI e conecte no pino 8.
22. Repita os procedimentos semelhantes de 3 a 18 para a porta 8-9-10 e anote os valores na tabela 3.3c. Onde for terminal 1 use terminal 9. Onde for terminal 2 use terminal 10.
23. Remova o fio roxo do pino 8 do CI e conecte no pino 11.
24. Repita os procedimentos semelhantes de 3 a 18 para a porta 11-12-13 e anote os valores na tabela 3.3d. Onde for terminal 1 use terminal 12. Onde for terminal 2 use terminal 13.
25. Repita os procedimentos semelhantes de 3 a 24 para o CI CMOS 4081. Lembre-se que as conexões internas são diferentes do TTL testado.

    Os procedimentos a seguir são opcionais e se referem à simulação em computador.

26. Utilizando o software de simulação SmartSim, simule a porta E (AND) de acordo com a figura 3.4. Compare a simulação com a tabela 3.1.
27. Utilizando o software de simulação Atanua, simule o CI 74LS08, conforme a figura 3.6.
28. Utilizando o software de simulação Qucs, simule a porta E (AND) e construa a tabela verdade de acordo com a figura 3.7. Compare os dados da tabela do Qucs com a tabela 3.1.

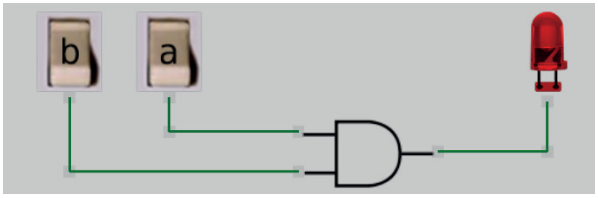


Figura 3.6 – Simulação da porta E de duas entradas no Atanua.

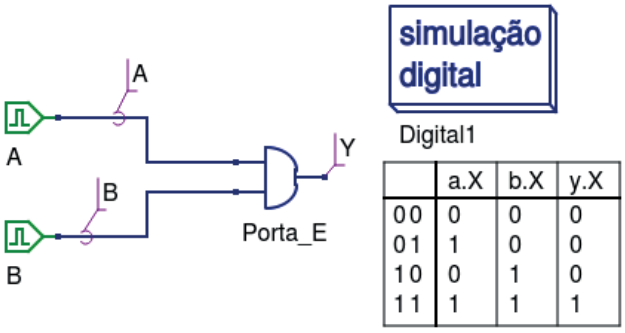


Figura 3.7 – Simulação da porta E de duas entradas no Qucs.

Tabelas de dados:

A	B	Y = A.B	Valor de tensão medido pelo voltímetro	
			Para o 74LS08	Para o 4081
0	0			
0	1			
1	0			
1	1			

Tabela 3.3a – Valores medidos na porta 1-2-3.

A	B	Y = A.B	Valor de tensão medido pelo voltímetro	
			Para o 74LS08	Para o 4081
0	0			
0	1			
1	0			
1	1			

Tabela 3.3b – Valores medidos na porta 4-5-6.

A	B	Y = A.B	Valor de tensão medido pelo voltímetro	
			Para o 74LS08	Para o 4081
0	0			
0	1			
1	0			
1	1			

Tabela 3.3c – Valores medidos na porta 8-9-10.

A	B	Y = A.B	Valor de tensão medido pelo voltímetro	
			Para o 74LS08	Para o 4081
0	0			
0	1			
1	0			
1	1			

Tabela 3.3d – Valores medidos na porta 11-12-13.

3.3.4. Experimento 2

Obter a tabela verdade para uma porta E de três entradas, construída a partir de duas portas E de duas entradas.

Esquemas:

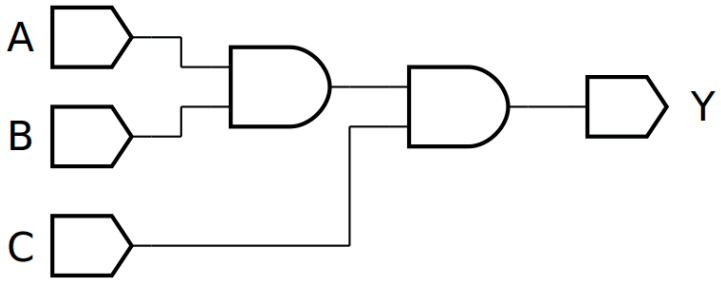


Figura 3.8 – Diagrama esquemático.

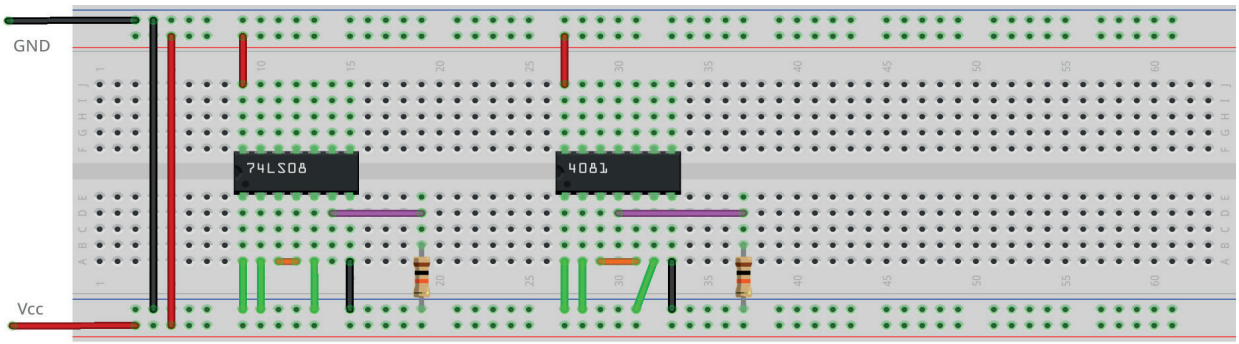


Figura 3.9 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.8, conforme mostrado na figura 3.9, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   s) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo.
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Na tabela 3.4 as colunas A, B e C são as entradas. “0” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de terra (gnd, preto) do protoboard. “1” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de energia (Vcc, vermelho) do protoboard. Antes de cada mudança nos fios verdes, o cabo de energia (vermelho) deve ser desconectado da fonte de energia. A conexão do cabo na fonte de energia somente deverá ser feita após a mudança dos fios verdes.

4. Meça o valor de tensão sobre o resistor e registre na coluna adequada da tabela 3.4.
5. Preencha a coluna Y da tabela 3.4 com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.
6. Repita os itens 3 a 5 para todas as combinações na tabela 3.4.
7. Repita os itens 3 a 6 para o CI 4081. Lembre-se que as conexões internas são diferentes do TTL testado.
8. Simule, no computador, o circuito da figura 3.8 utilizando os programas SmartSim e Atanua (figura 3.10) e Qucs (figura 3.11). Compare os resultados com a tabela 3.2 e 3.4.

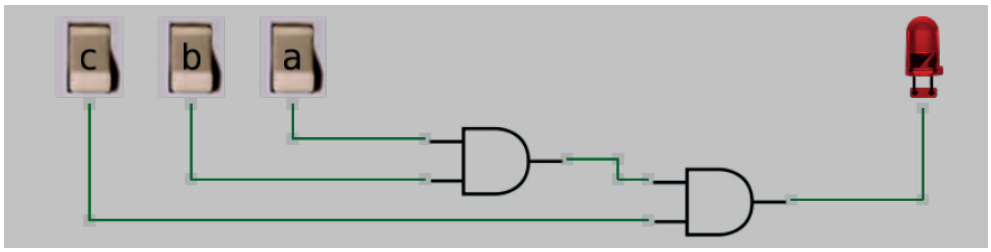
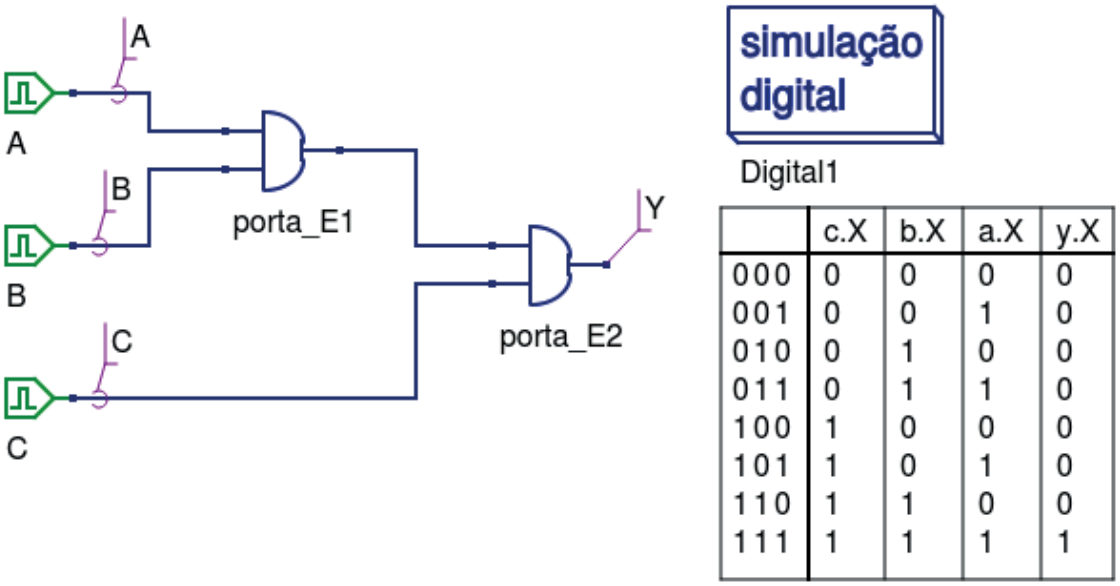


Figura 3.10 – Simulação da porta E com 3 entradas no Atanua.



	c.X	b.X	a.X	y.X
000	0	0	0	0
001	0	0	1	0
010	0	1	0	0
011	0	1	1	0
100	1	0	0	0
101	1	0	1	0
110	1	1	0	0
111	1	1	1	1

Figura 3.11 – Simulação da porta E de 3 entradas no Qucs.

Tabelas de dados:

A	B	C	Y = A.B.C	Valor de tensão medido pelo voltímetro	
				Para o 74LS08	Para o 4081
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 3.4 – Tabela verdade para porta E de 3 entradas.

3.4. Exame da porta NE (NAND) ou porta E invertida

3.4.1. Objetivos

1. Conhecer a lógica digital da porta NE (NAND) e verificar sua operação lógica,
2. Derivar sua tabela de verdade e aprender algumas propriedades dela.
3. Familiarizar-se com a porta TTL 74LS00 NE (NAND).
4. Analisar as propriedades dos circuitos compostos por 3 ou mais portas NE (NAND).

3.4.2. Informação preliminar

1. A porta NE (NAND) fornece a saída exatamente como forma invertida de uma porta E (AND).
2. Simbolicamente, esta situação é mostrada com uma porta E (AND) tendo um pequeno círculo na saída conforma a figura 3.12.
3. O princípio de funcionamento de uma porta NE (NAND) pode ser resumido como: Se todas as entradas forem 1 (H), a saída será 0 (L) e se houver apenas um nível 0 (L) em qualquer uma das entradas, a saída será 1 (H). Isso pode ser visto na tabela 3.5.
4. Como pode ser visto na figura 3.13, tanto o CI TTL 7400 quanto o CI CMOS 4011 contêm 4 portas NE (NAND), mas a disposição interna das portas são diferentes.
5. Se todas as entradas de uma porta NE (NAND) estiverem conectadas, ela pode operar como um inversor.

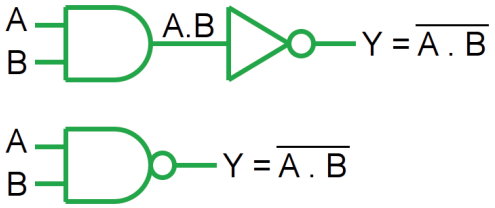


Figura 3.12 – Porta NE (NAND) de 2 entradas.

A	B	A.B	Y=(A.B)'
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Tabela 3.5 – Tabela verdade da portaNE (NAND) de 2 entradas.

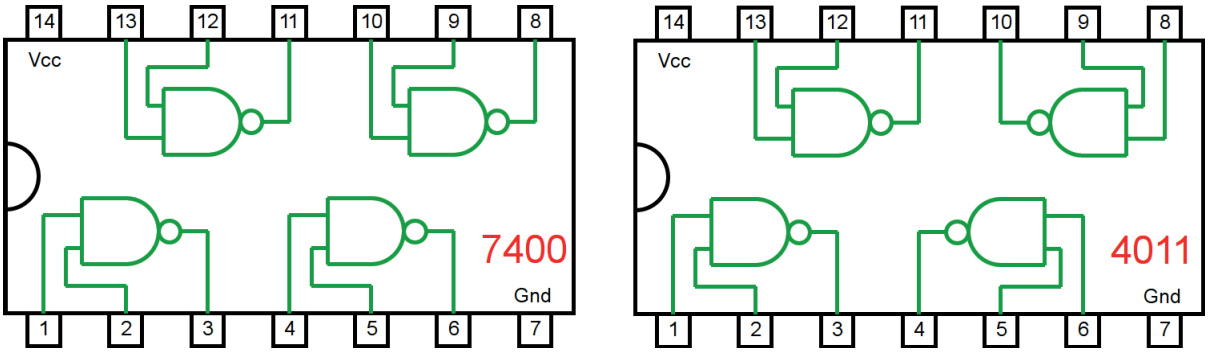


Figura 3.13 – Circuito Integrado TTL e CMOS.

3.4.3. Experimento 1

Obter a tabela verdade para uma porta NE de duas entradas.

Esquemas:

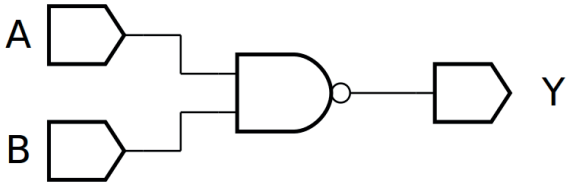


Figura 3.14 – Diagrama esquemático.

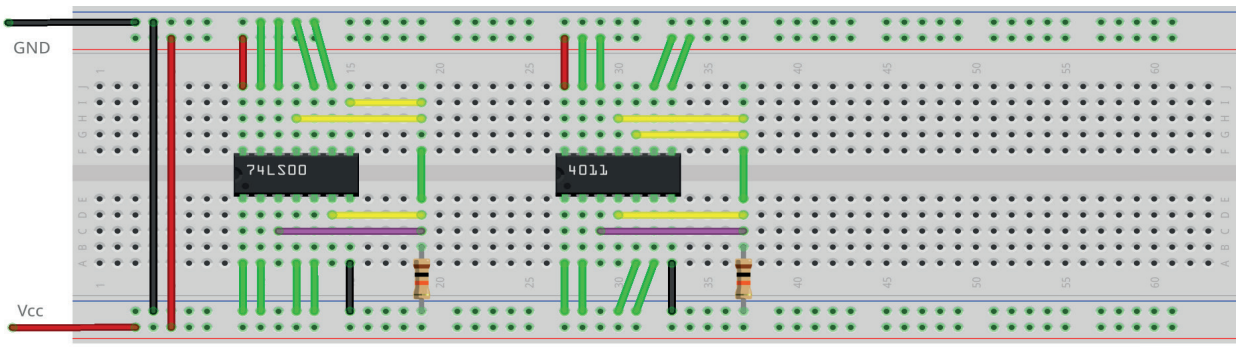


Figura 3.15 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.14, conforme mostrado na figura 3.15, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo. Não conecte os fios amarelos!
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.

   Os procedimentos a seguir se referem ao teste da porta 1-2-3.
3. Conecte o cabo vermelho na fonte de energia.
4. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.6a, na coluna correspondente ao CI e linha corresponde a A =0 e B=0.
5. Desconecte o cabo vermelho na fonte de energia.
6. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).

7. Conecte o cabo vermelho na fonte de energia.
8. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.6a, na coluna correspondente ao CI e linha corresponde a A =0 e B=1.
9. Desconecte o cabo vermelho na fonte de energia.
10. Mude o fio verde do terminal 1 do CI do barramento de energia (vermelho, 1) para o barramento de terra (preto, 0). Mude o fio verde do terminal 2 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
11. Conecte o cabo vermelho na fonte de energia.
12. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.6a, na coluna correspondente ao CI e linha corresponde a A =1 e B=0.
13. Desconecte o cabo vermelho na fonte de energia.
14. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
15. Conecte o cabo vermelho na fonte de energia.
16. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.6a, na coluna correspondente ao CI e linha corresponde a A =1 e B=1.
17. Desconecte o cabo vermelho na fonte de energia.
18. Preencha a 3ª coluna da tabela 3.6a com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.

    Os procedimentos a seguir são opcionais e se referem aos testes das outras portas e outro CI.

19. Remova o fio roxo do pino 3 do CI e conecte no pino 6.
20. Repita os procedimentos semelhantes de 3 a 18 para a porta 4-5-6 e anote os valores na tabela 3.6b. Onde for terminal 1 use terminal 4. Onde for terminal 2 use terminal 5.
21. Remova o fio roxo do pino 6 do CI e conecte no pino 8.
22. Repita os procedimentos semelhantes de 3 a 18 para a porta 8-9-10 e anote os valores na tabela 3.6c. Onde for terminal 1 use terminal 9. Onde for terminal 2 use terminal 10.
23. Remova o fio roxo do pino 8 do CI e conecte no pino 11.
24. Repita os procedimentos semelhantes de 3 a 18 para a porta 11-12-13 e anote os valores na tabela 3.6d. Onde for terminal 1 use terminal 12. Onde for terminal 2 use terminal 13.
25. Repita os procedimentos semelhantes de 3 a 24 para o CI CMOS 4011. Lembre-se que as conexões internas são diferentes do TTL testado.

    Os procedimentos a seguir são opcionais e se referem à simulação em computador.

26. Utilizando o software de simulação SmartSim, simule a porta NE (NAND) de acordo com a figura 3.14. Compare a simulação com a tabela 3.5.
27. Utilizando o software de simulação Atanua, simule a porta NE (NAND) conforme a figura 3.16. Compare a simulação com a tabela 3.5.
28. Utilizando o software de simulação Qucs, simule a porta NE (NAND) de modo semelhante ao simulado anteriormente para a porta E (AND) conforme a figura 3.17. Compare os dados da tabela do Qucs com a tabela 3.5.

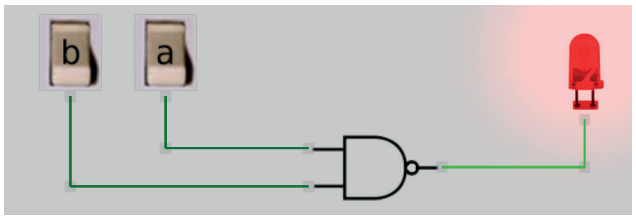


Figura 3.16 – Simulação da porta NE de 2 entradas no Atanua.

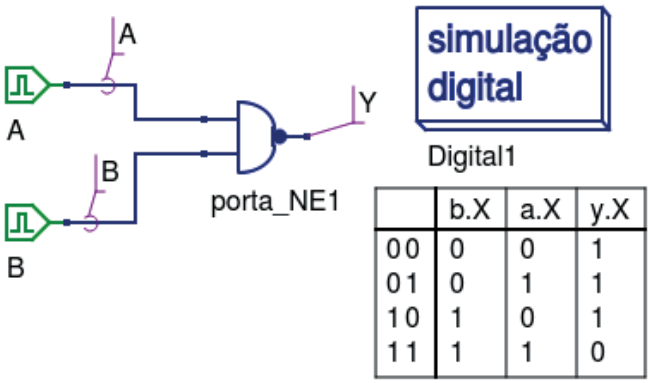


Figura 3.17 – Simulação da porta NE de 2 entradas no Qucs.

Tabelas de dados:

A	B	Y = (A.B)'	Valor de tensão medido pelo voltímetro	
			Para o 74LS00	Para o 4011
0	0			
0	1			
1	0			
1	1			

Tabela 3.6a – Valores medidos na porta 1-2-3.

A	B	Y = (A.B)'	Valor de tensão medido pelo voltímetro	
			Para o 74LS00	Para o 4011
0	0			
0	1			
1	0			
1	1			

Tabela 3.6b – Valores medidos na porta 4-5-6.

A	B	Y = (A.B)'	Valor de tensão medido pelo voltímetro	
			Para o 74LS00	Para o 4011
0	0			
0	1			
1	0			
1	1			

Tabela 3.6c – Valores medidos na porta 8-9-10.

A	B	Y = (A.B)'	Valor de tensão medido pelo voltímetro	
			Para o 74LS00	Para o 4011
0	0			
0	1			
1	0			
1	1			

Tabela 3.6d – Valores medidos na porta 11-12-13.

3.4.4. Experimento 2

Usando a porta NE como porta inversora.

Esquemas:

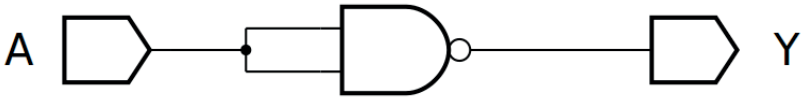


Figura 3.18 – Diagrama esquemático.

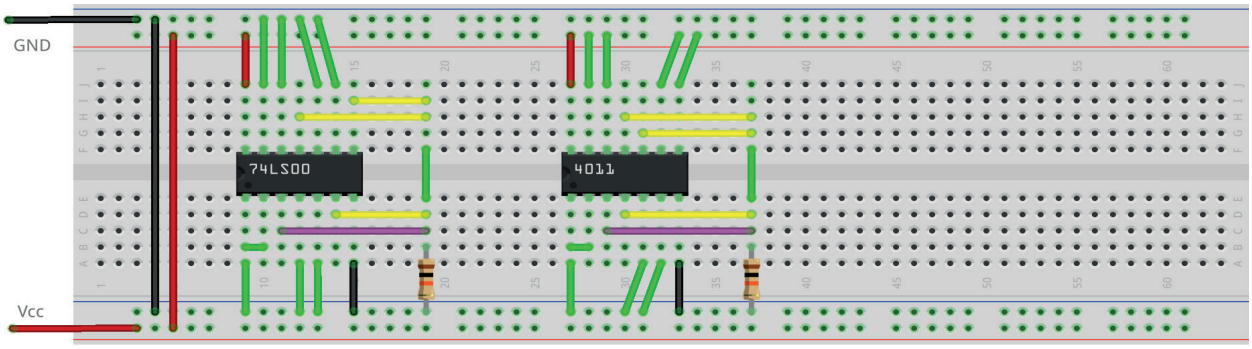


Figura 3.19 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.18, conforme mostrado na figura 3.19, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo. Não conecte os fios amarelos!
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho na fonte de energia.
4. Faça uma medida de tensão sobre o resistor. Se o valor de tensão for próximo de Vcc registre “1” linha corresponde a A =0, caso contrário se a tensão for próximo de Gnd, registre “0”.
4. Desconecte o cabo vermelho na fonte de energia.
6. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
7. Conecte o cabo vermelho na fonte de energia.

8. Faça uma medida de tensão sobre o resistor. Se o valor de tensão for próximo de Vcc registre “1” linha corresponde a A =1, caso contrário se a tensão for próximo de Gnd, registre “0”.
9. Desconecte o cabo vermelho na fonte de energia.
10. Repita os procedimentos anteriores para o CI 4011. Lembre-se que as conexões internas são diferentes do TTL testado.
11. Utilizando o software de simulação SmartSim, simule a porta NE (NAND) de acordo com a figura 3.18. Compare a simulação com a tabela 3.7.
12. Utilizando o software de simulação Atanua, simule a porta NE (NAND) conforme a figura 3.20. Compare a simulação com a tabela 3.7.
13. Utilizando o software de simulação Qucs, simule a porta NE (NAND) conforme a figura 3.21. Compare os dados da tabela do Qucs com a tabela 3.7.

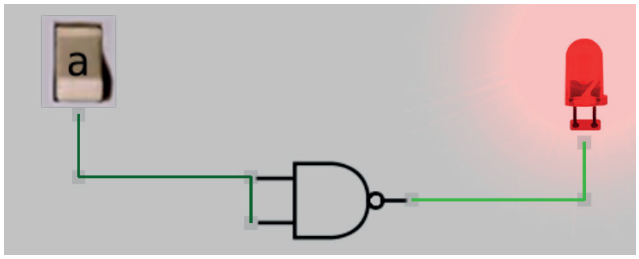


Figura 3.20 – Simulação da porta NE como inversor no Atanua.

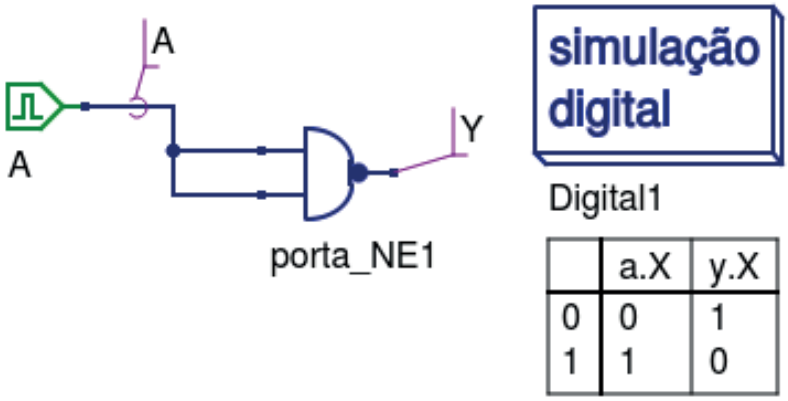


Figura 3.21 – Simulação da porta NE como inversor no Qucs.

Tabelas de dados:

A	Y = (A)’
0	
1	

Tabela 3.7 – Tabela verdade da porta NE atuando como porta inversora.

3.4.5. Experimento 3

Obtendo uma porta NE de três entradas, construída a partir de portas NE de duas entradas.

Esquemas:

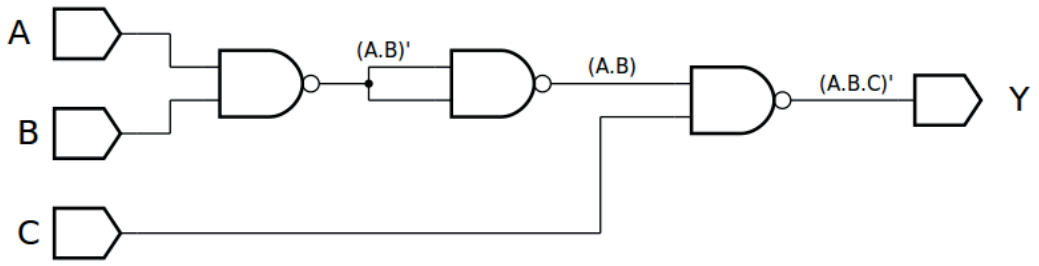


Figura 3.22 – Diagrama esquemático.

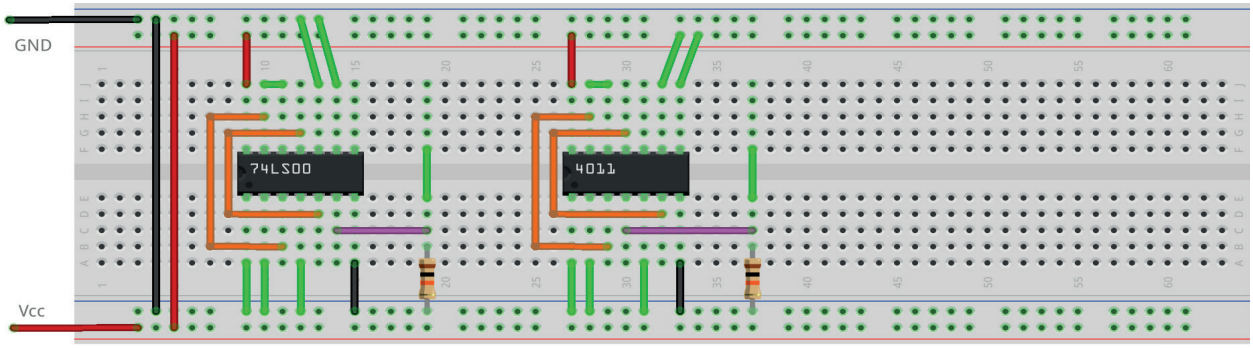


Figura 3.23 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.22, conforme mostrado na figura 3.23, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo.
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Na tabela 3.8 as colunas A, B e C são as entradas. “0” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de terra (gnd, preto) do protoboard. “1” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de energia (Vcc, vermelho) do protoboard. Antes de cada mudança nos fios verdes, o cabo de energia (vermelho) deve ser desconectado da fonte de energia. A conexão do cabo na fonte de energia somente deverá ser feita após a mudança dos fios verdes.
4. Meça o valor de tensão sobre o resistor e registre na coluna adequada da tabela 3.8.

5. Preencha a coluna Y da tabela 3.8 com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.
6. Repita os procedimentos de 3 a 5 para todas as combinações da tabela 3.8.
7. Repita os itens 3 a 6 para o CI 4011. Lembre-se que as conexões internas são diferentes do TTL testado.
8. Simule, no computador, o circuito da figura 3.22 utilizando os programas SmartSim e Atanua (figura 3.24) e Qucs (figura 3.25). Compare os resultados com a tabela 3.8.

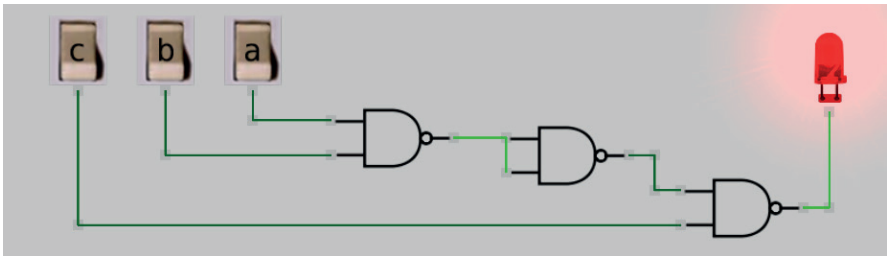


Figura 3.24 – Simulação da porta NE com 3 entradas no Atanua.

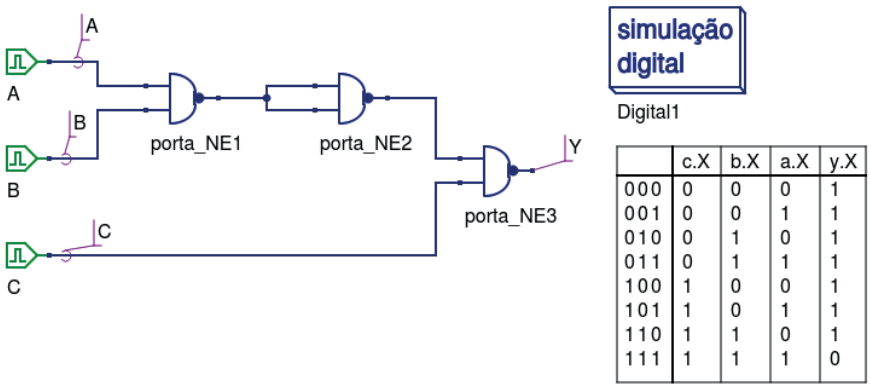


Figura 3.25 – Simulação da porta NE de 3 entradas no Qucs.

Tabelas de dados:

A	B	C	Y = (A.B.C)’	Valor de tensão medido pelo voltímetro	
				Para o 74LS08	Para o 4081
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 3.8 – Tabela verdade para porta NE de 3 entradas.

3.5. Exame da porta INVERSORA (NOT)

3.5.1. Objetivos

1. Conhecer a lógica digital da porta INVERSORA e verificar a sua operação lógica,
2. Derivar a sua tabela de verdade.
3. Familiarizar-se com o CI 74LS04 INVERSOR.
4. Aprender a converter "uma porta E em uma porta OU" e converter "uma porta OU em uma porta E" usando INVERSORES.

3.5.2. Informação preliminar

1. Um inversor inverte o nível lógico na entrada (se entrada = 0 então saída = 1; se entrada = 1 então saída = 0) conforme a tabela 3.9.
2. Simbolicamente, essa situação é denotada com um pequeno círculo nas entradas ou saídas conforme a figura 3.26.
3. Colocar uma pequena barra no topo da letra representando a entrada ou saída significa lógica invertida.
4. Esta situação é simplesmente denominada "NÃO"(NOT).
5. Como pode ser visto na Figura 3.27, tanto o CI TTL 7404 quanto o CI CMOS 4063 contêm 6 portas INVERSORAS. Observe que a disposição interna dos inversores é a mesma para os dois tipos de CI.

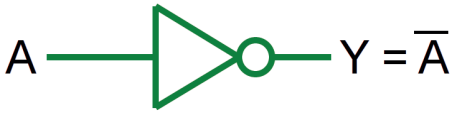


Figura 3.26 – Porta INVERSORA.

A	Y = A'
0	1
1	0

Tabela 3.9 – Tabela verdade da porta INVERSORA.

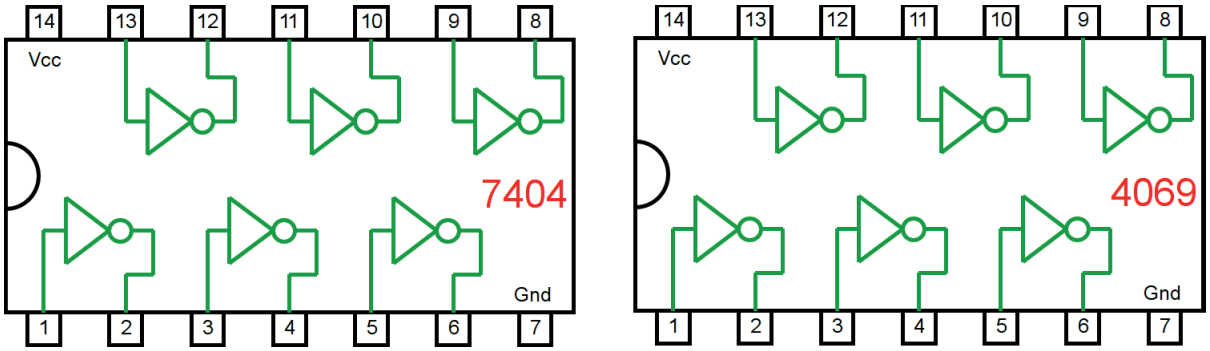


Figura 3.27 – Circuito Integrado TTL e CMOS.

3.5.3. Convertendo uma porta E em uma porta OU usando inversores

De acordo com a lei de De Morgan, a fim de alterar a multiplicação em adição

a) As variáveis de entrada são invertidas logicamente.

b) O sinal de multiplicação é substituído por sinal de adição.

c) Todo o processo é invertido.

O diagrama de conexão para converter uma porta E em uma porta OU usando inversores é mostrado na Figura 3.28. A tabela verdade correspondente é fornecida na tabela 3.10.

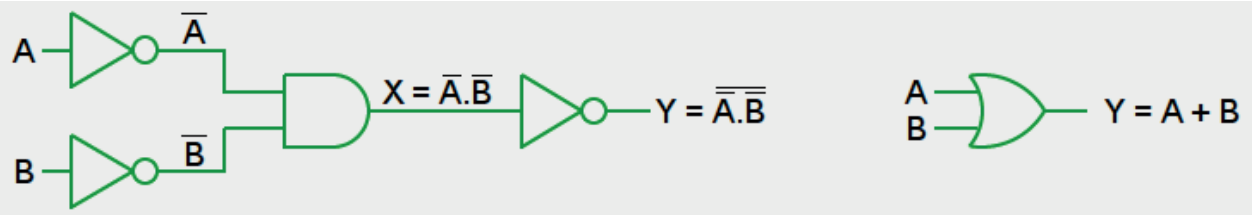


Figura 3.28 – Convertendo uma porta E em uma porta OU usando inversores.

Entradas				Saída	
A	B	Inversão		X = A' . B'	Y = A + B
		A'	B'		
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

Tabela 3.10 – Tabela verdade da porta OU convertida de uma porta E.

3.5.4. Convertendo uma porta OU em uma porta E usando inversores

De acordo com a lei de De Morgan, a fim de alterar a adição em multiplicação:

a) As variáveis de entrada são invertidas logicamente.

b) O sinal de adição é substituído por um sinal de multiplicação.

c) Todo o processo é invertido.

O diagrama de conexão para converter uma porta OU em uma porta E usando inversores é mostrado na figura 3.29. A tabela verdade correspondente é fornecida na Tabela 3.11.

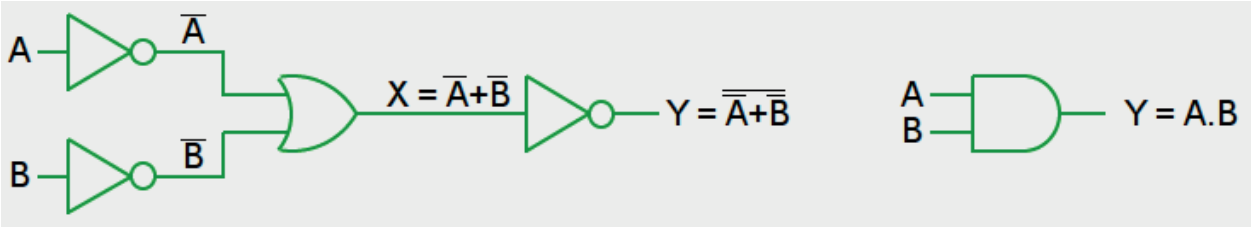


Figura 3.29 – Convertendo uma porta OU em uma porta E usando inversores.

Entradas				Saída	
A	B	Inversão		X = A' + B'	Y = A . B
		A'	B'		
0	0	1	1	1	0
0	1	1	0	1	0
1	0	0	1	1	0
1	1	0	0	0	1

Tabela 3.11 – Tabela verdade da porta E convertida de uma porta OU.

3.5.5. Experimento 1

Obter a tabela verdade da porta INVERSORA.

Esquemas:

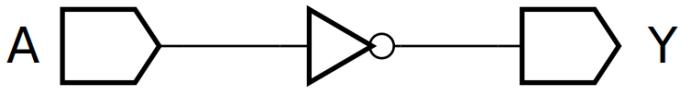


Figura 3.30 – Diagrama esquemático.

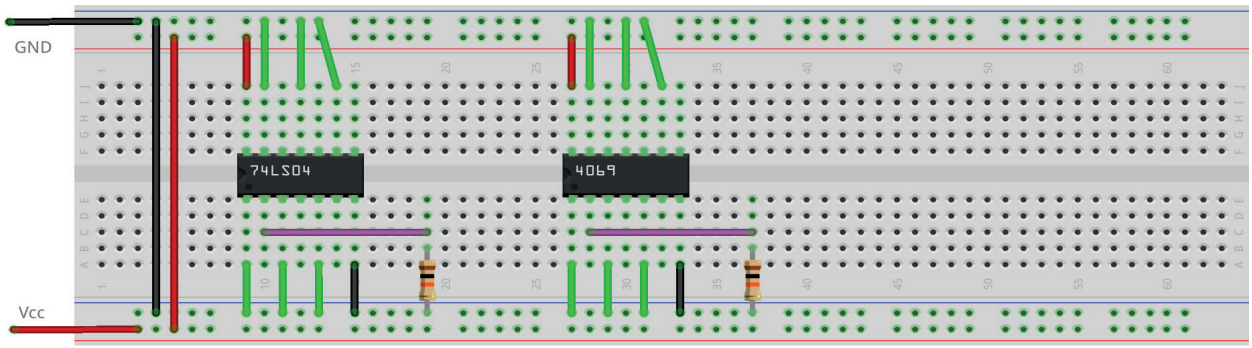


Figura 3.31 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.30, conforme mostrado na figura 3.31, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo.
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Na tabela 3.12 a coluna A são as entradas. “0” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de terra (gnd, preto) do protoboard. “1” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de energia (Vcc, vermelho) do protoboard. Antes de cada mudança nos fios verdes, o cabo de energia (vermelho) deve ser desconectado da fonte de energia. A conexão do cabo na fonte de energia somente deverá ser feita após a mudança dos fios verdes.
4. Meça o valor de tensão sobre o resistor e registre na coluna adequada da tabela 3.12.
5. Preencha a coluna Y da tabela 3.12 com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.
6. Repita o procedimento de medição de tensão, itens 4 e 5, para cada uma das 6 portas. Desconecte o cabo de energia da fonte e mude o fio roxo de acordo com a saída desejada (pinos 1, 3, 5, 9, 11,13). Reconecte o cabo de energia na fonte.
7. Repita os itens 4 a 6 para o CI 4069. Lembre-se que as conexões internas são as mesmas do TTL testado.

8. Simule, no computador, o circuito da figura 3.30 utilizando os programas SmartSim e Atanua (figura 3.32) e Qucs (figura 3.33). Compare os resultados com a tabela 3.12.

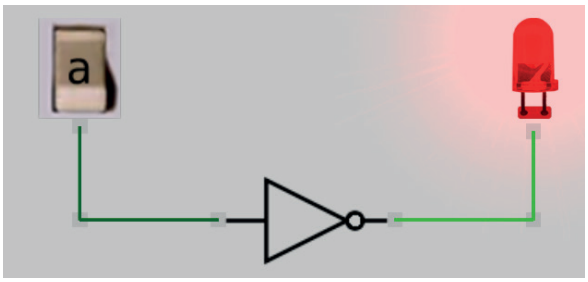


Figura 3.32 – Simulação da porta INVERSORA no Atanua.

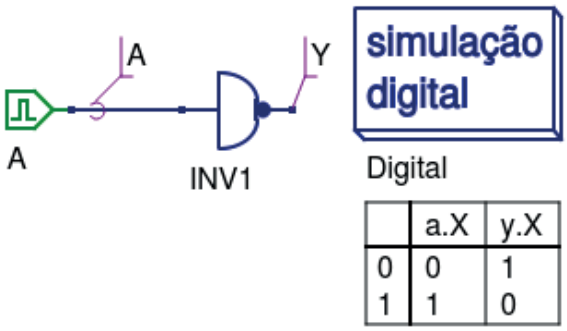


Figura 3.33 – Simulação da porta INVERSORA no Qucs.

Tabelas de dados:

Pinos	A (entrada)	Y (saída)	Valor de tensão medido pelo voltímetro	
			Para o 74LS08	Para o 4081
1-2	0			
	1			
3-4	0			
	1			
5-6	0			
	1			
8-9	0			
	1			
10-11	0			
	1			
12-13	0			
	1			

Tabela 3.12 – Tabela verdade para porta INVERSORA.

3.5.6. Experimento 2

Converter uma porta E em uma porta OU usando inversores.

Esquemas:

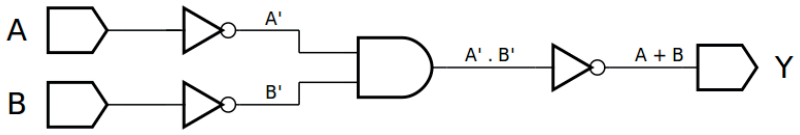


Figura 3.34 – Diagrama esquemático.

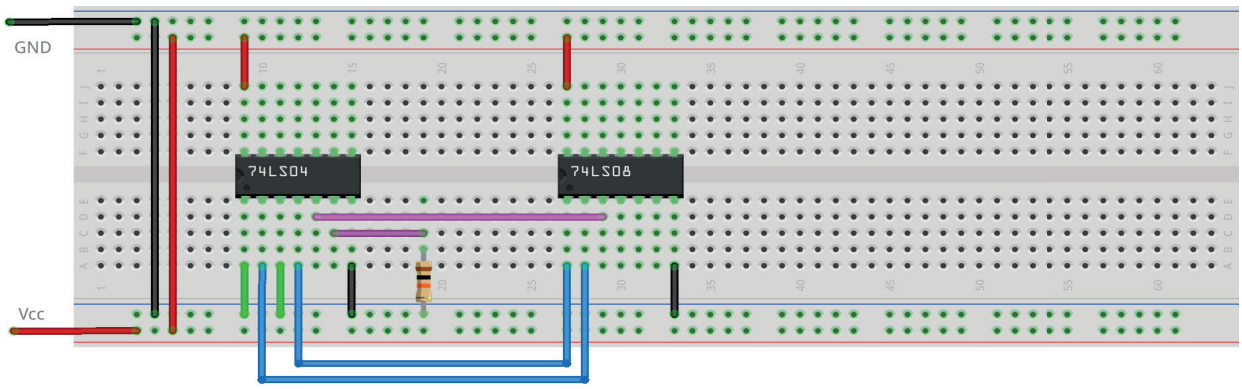


Figura 3.35 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.34, conforme mostrado na figura 3.35, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios azuis. Eles estão nas saídas dos inversores e fornecem A' e B'.
   f) Conecte o fio rosa. Ele está conectado na saída da porta E e fornece o produto (A' . B').
   g) Conecte o fio roxo. Ele está conectado na saída de um inversor e fornece a soma (A + B).
   h) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   i) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Na tabela 3.13 as colunas A e B são as entradas. "0" indica que o fio verde conectado nesta entrada deve ser conectado no barramento de terra (gnd, preto) do protoboard. "1" indica que o fio verde conectado nesta entrada deve ser conectado no barramento de energia (Vcc, vermelho) do protoboard. Antes de cada mudança nos fios verdes, o cabo de energia (ver-

melho) deve ser desconectado da fonte de energia. A conexão do cabo na fonte de energia somente deverá ser feita após a mudança dos fios verdes.

4. Meça o valor de tensão sobre o resistor. Preencha a coluna Y da tabela 3.13 com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.
5. Repita os itens 3 e 4 para cada uma das linhas da tabela 3.13.
6. Simule, no computador, o circuito da figura 3.34 utilizando os programas SmartSim e Atanua (figura 3.36) e Qucs (figura 3.37). Compare os resultados com a tabela 3.13.

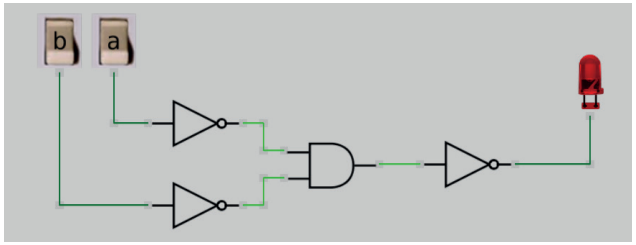


Figura 3.36 – Simulação da porta OU convertida de porta E no Atanua.

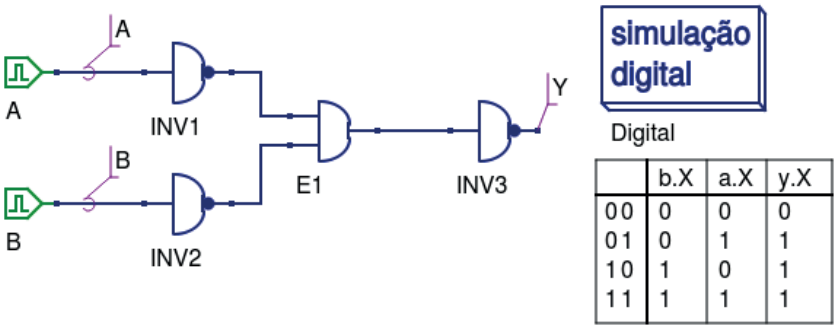


Figura 3.37 – Simulação da porta OU convertida de porta E no Qucs.

Tabelas de dados:

Entradas				Saída
A	B	INVERSA		Y = A + B
		A’	B’	
0	0	1	1	
0	1	1	0	
1	0	0	1	
1	1	0	0	

Tabela 3.13 – Tabela verdade para porta OU convertida de porta E.

3.5.7. Experimento 3

Converter uma porta OU em uma porta E usando inversores.

Esquemas:

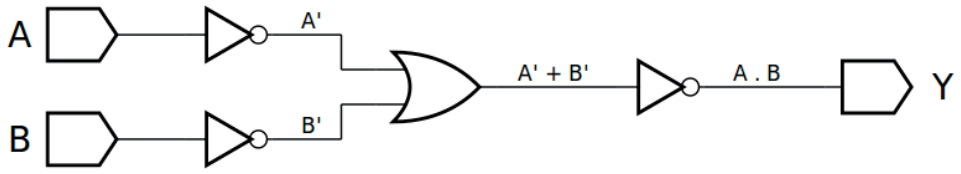


Figura 3.38 – Diagrama esquemático.

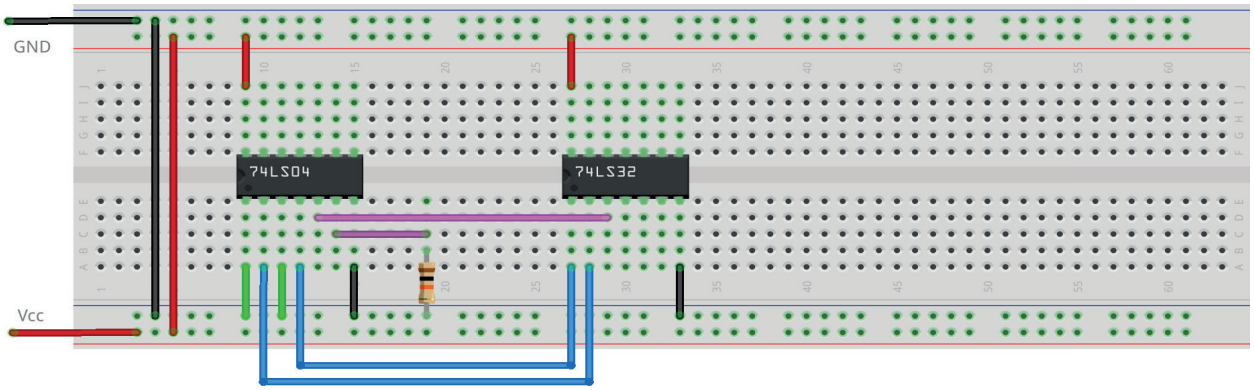


Figura 3.39 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.38, conforme mostrado na figura 3.39, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios azuis. Eles estão nas saídas dos inversores e fornecem A' e B'.
   f) Conecte o fio rosa. Ele está conectado na saída da porta E e fornece o produto (A' . B').
   g) Conecte o fio roxo. Ele está conectado na saída de um inversor e fornece a soma (A + B).
   h) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   i) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Na tabela 3.14 as colunas A e B são as entradas. "0" indica que o fio verde conectado nesta entrada deve ser conectado no barramento de terra (gnd, preto) do protoboard. "1" indica que o fio verde conectado nesta entrada deve ser conectado no barramento de energia (Vcc,

vermelho) do protoboard. Antes de cada mudança nos fios verdes, o cabo de energia (vermelho) deve ser desconectado da fonte de energia. A conexão do cabo na fonte de energia somente deverá ser feita após a mudança dos fios verdes.

4. Meça o valor de tensão sobre o resistor. Preencha a coluna Y da tabela 3.14 com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.
5. Repita os itens 3 e 4 para cada uma das linhas da tabela 3.14.
6. Simule, no computador, o circuito da figura 3.38 utilizando os programas SmartSim e Atanua (figura 3.40) e Qucs (figura 3.41). Compare os resultados com a tabela 3.14.

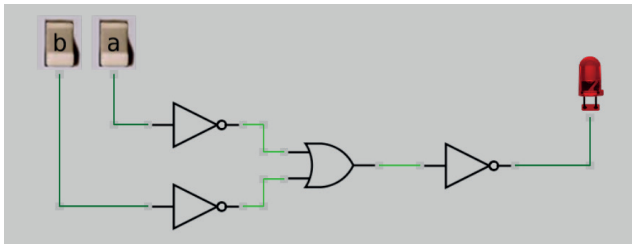


Figura 3.40 – Simulação da porta E convertida de porta OU no Atanua.

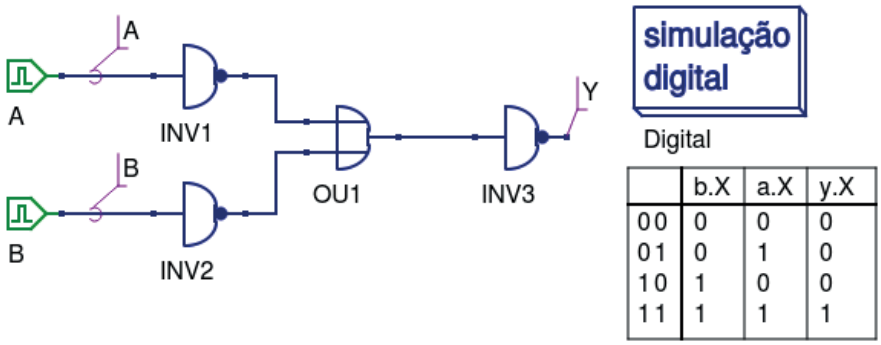


Figura 3.41 – Simulação da porta E convertida de porta OU no Qucs.

Tabelas de dados:

Entradas				Saída
A	B	INVERSA		Y = A . B
		A’	B’	
0	0	1	1	
0	1	1	0	
1	0	0	1	
1	1	0	0	

Tabela 3.14 – Tabela verdade para porta E convertida de porta OU.

3.6. Exame da porta OU (OR)

3.6.1. Objetivos

1. Conhecer a lógica digital da porta OU e verificar sua operação lógica.
2. Obter uma porta OU de 3 entradas ou 4 entradas usando várias portas OU de 2 entradas.

3.6.2. Informação preliminar

1. A porta OU é uma porta de adição. Tem pelo menos duas entradas. Para qualquer condição das entradas 1 (H), a saída estará no nível 1 (H). Apenas para todas as entradas serem 0 (L), a saída será 0 (L) conforme a tabela 3.15.
2. A saída de uma porta OU de 2 entradas é Y = A + B, figura 3.42. A tabela verdade da porta OU de 2 entradas é dada na tabela 3.15.
3. A saída de uma porta OU de 3 entradas é Y = A + B + C, figura 3.43. A tabela verdade da porta OU de 3 entradas é dada na Tabela 3.16.
4. O CI TTL 74LS32 contém 4 portas OU de duas entradas, figura 3.44.

Figura 3.42 – Porta OU de duas entradas.

A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 3.15 – Tabela verdade para uma porta OU de duas entradas.

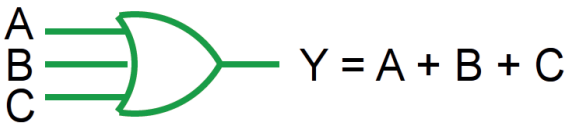


Figura 3.43 – Porta OU de três entradas.

A	B	C	Y = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Tabela 3.16 – Tabela verdade para uma porta OU de três entradas.

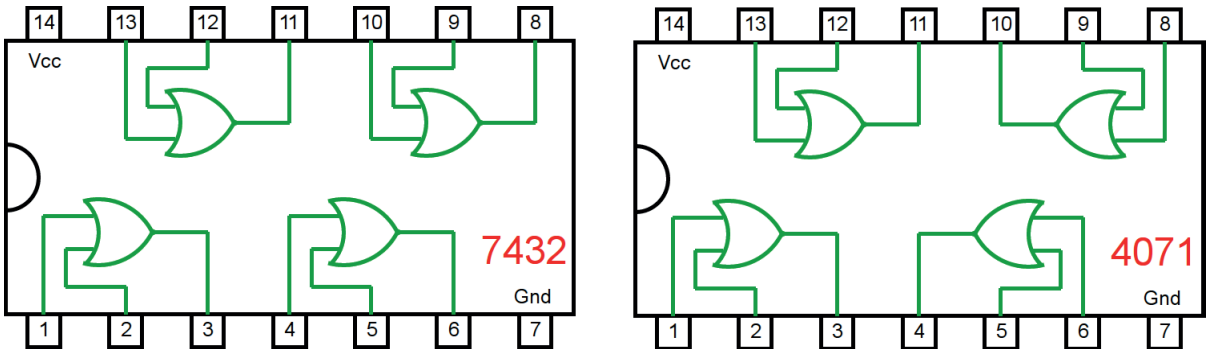


Figura 3.44 – Circuito integrado TTL e CMOS.

3.6.3. Experimento 1

Obter a tabela verdade da porta OU de duas entradas.

Esquemas:

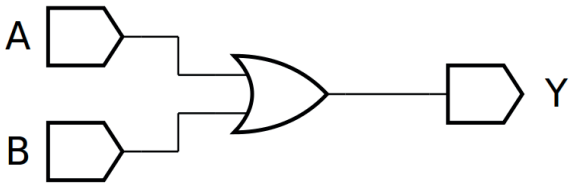


Figura 3.45 – Diagrama esquemático.7

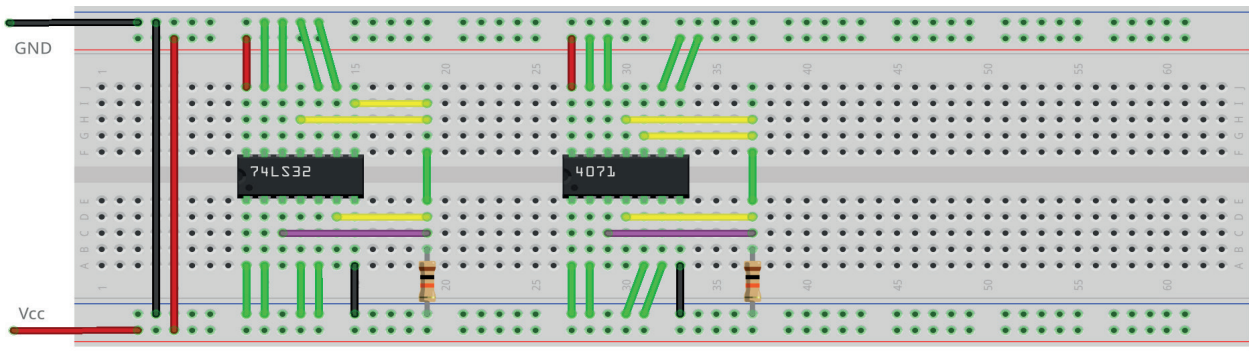


Figura 3.46 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.45, conforme mostrado na figura 3.46, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo. Não conecte os fios amarelos!
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.

   Os procedimentos a seguir se referem ao teste da porta 1-2-3.
3. Conecte o cabo vermelho na fonte de energia.
4. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.17a, na coluna correspondente ao CI e linha corresponde a A =0 e B=0.
5. Desconecte o cabo vermelho na fonte de energia.
6. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
7. Conecte o cabo vermelho na fonte de energia.

8. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.17a, na coluna correspondente ao CI e linha corresponde a A =0 e B=1.
9. Desconecte o cabo vermelho na fonte de energia.
10. Mude o fio verde do terminal 1 do CI do barramento de energia (vermelho, 1) para o barramento de terra (preto, 0). Mude o fio verde do terminal 2 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
11. Conecte o cabo vermelho na fonte de energia.
12. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.17a, na coluna correspondente ao CI e linha corresponde a A =1 e B=0.
13. Desconecte o cabo vermelho na fonte de energia.
14. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
15. Conecte o cabo vermelho na fonte de energia.
16. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.17a, na coluna correspondente ao CI e linha corresponde a A =1 e B=1.
17. Desconecte o cabo vermelho na fonte de energia.
18. Preencha a 3ª coluna da tabela 3.17a com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.

Os procedimentos a seguir são opcionais e se referem aos testes das outras portas e outro CI.

19. Remova o fio roxo do pino 3 do CI e conecte no pino 6.
20. Repita os procedimentos semelhantes de 3 a 18 para a porta 4-5-6 e anote os valores na tabela 3.17b. Onde for terminal 1 use terminal 4. Onde for terminal 2 use terminal 5.
21. Remova o fio roxo do pino 6 do CI e conecte no pino 8.
22. Repita os procedimentos semelhantes de 3 a 18 para a porta 8-9-10 e anote os valores na tabela 3.17c. Onde for terminal 1 use terminal 9. Onde for terminal 2 use terminal 10.
23. Remova o fio roxo do pino 8 do CI e conecte no pino 11.
24. Repita os procedimentos semelhantes de 3 a 18 para a porta 11-12-13 e anote os valores na tabela 3.17d. Onde for terminal 1 use terminal 12. Onde for terminal 2 use terminal 13.
25. Repita os procedimentos semelhantes de 3 a 24 para o CI CMOS 4071. Lembre-se que as conexões internas são diferentes do TTL testado.

Os procedimentos a seguir são opcionais e se referem à simulação em computador.

26. Utilizando o software de simulação SmartSim, simule a porta E (AND) de acordo com a figura 3.45. Compare a simulação com a tabela 3.15.
27. Utilizando o software de simulação Atanua, simule a porta OU, conforme a figura 3.47.
28. Utilizando o software de simulação Qucs, simule a porta OU (OR) e construa a tabela verdade de acordo com a figura 3.48. Compare os dados da tabela do Qucs com a tabela 3.15.

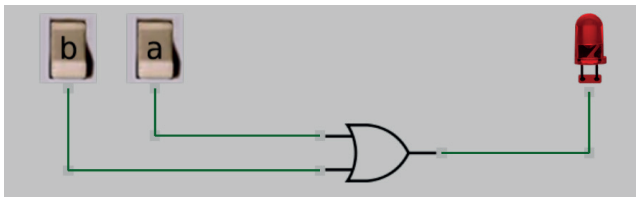


Figura 3.47 – Simulação da porta OU no Atanua.

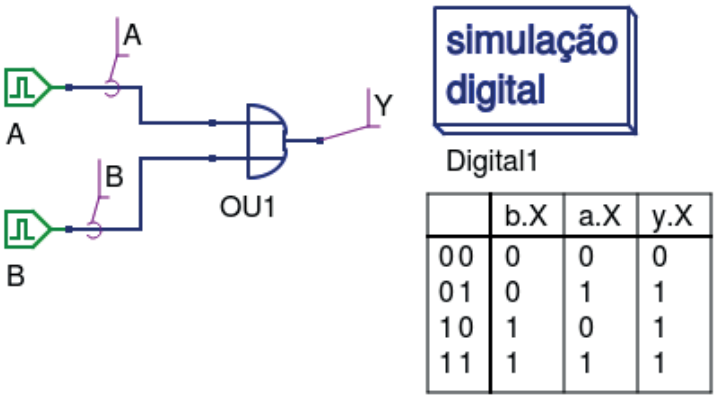


Figura 3.48 – Simulação da porta OU no Qucs.

Tabelas de dados:

A	B	Y = A + B	Valor de tensão medido pelo voltímetro	
			Para o 74LS32	Para o 4071
0	0			
0	1			
1	0			
1	1			

Tabela 3.17a – Valores medidos na porta 1-2-3.

A	B	Y = A + B	Valor de tensão medido pelo voltímetro	
			Para o 74LS32	Para o 4071
0	0			
0	1			
1	0			
1	1			

Tabela 3.17b – Valores medidos na porta 4-5-6.

A	B	Y = A + B	Valor de tensão medido pelo voltímetro	
			Para o 74LS32	Para o 4071
0	0			
0	1			
1	0			
1	1			

Tabela 3.17c – Valores medidos na porta 8-9-10.

A	B	Y = A + B	Valor de tensão medido pelo voltímetro	
			Para o 74LS32	Para o 4071
0	0			
0	1			
1	0			
1	1			

Tabela 3.17d – Valores medidos na porta 11-12-13.

3.6.4. Experimento 2

Obter a tabela verdade da porta OU de três entradas, construída a partir de duas portas OU de duas entradas.

Esquemas:

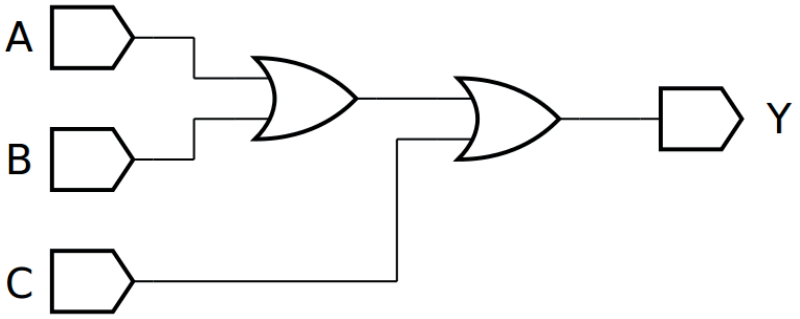


Figura 3.49 – Diagrama esquemático.

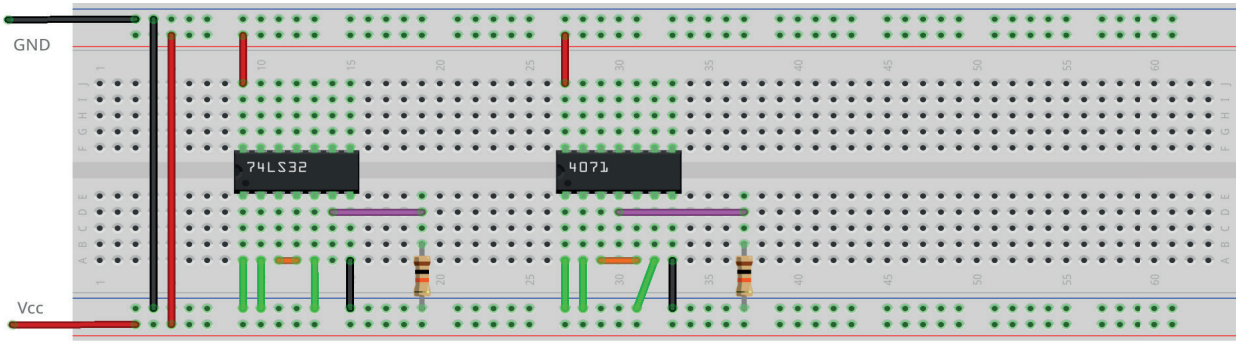


Figura 3.50 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.49, conforme mostrado na figura 3.50, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo.
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Na tabela 3.18 as colunas A, B e C são as entradas. “0” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de terra (gnd, preto) do protoboard. “1” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de energia (Vcc, vermelho) do protoboard. Antes de cada mudança nos fios verdes, o cabo de

energia (vermelho) deve ser desconectado da fonte de energia. A conexão do cabo na fonte de energia somente deverá ser feita após a mudança dos fios verdes.

4. Meça o valor de tensão sobre o resistor e registre na coluna adequada da tabela 3.18.
5. Preencha a coluna Y da tabela 3.18 com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.
6. Repita os itens 3 a 5 para todas as combinações na tabela 3.18.
7. Repita os itens 3 a 6 para o CI 4071. Lembre-se que as conexões internas são diferentes do TTL testado.
8. Simule, no computador, o circuito da figura 3.49 utilizando os programas SmartSim e Atanua (figura 3.51) e Qucs (figura 3.52). Compare os resultados com a tabela 3.16 e 3.18.

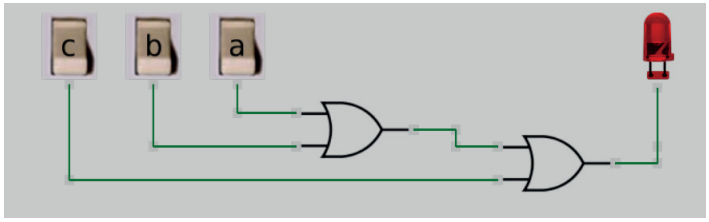
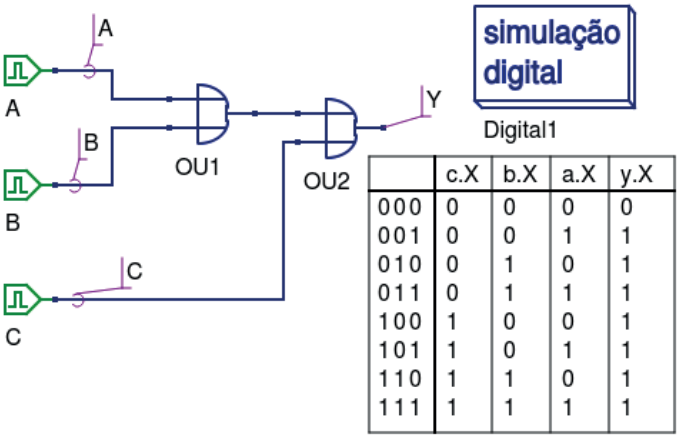


Figura 3.51 – Simulação da porta OU de 3 entradas no Atanua.



	c.X	b.X	a.X	y.X
000	0	0	0	0
001	0	0	1	1
010	0	1	0	1
011	0	1	1	1
100	1	0	0	1
101	1	0	1	1
110	1	1	0	1
111	1	1	1	1

Figura 3.52 – Simulação da porta OU de 3 entradas no Qucs.

Tabelas de dados:

A	B	C	Y = A + B +C	Valor de tensão medido pelo voltímetro	
				Para o 74LS32	Para o 4071
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 3.18 – Tabela verdade para porta OU de 3 entradas.

3.7. Exame da porta NOU (NOR) ou porta OU invertida

3.7.1. Objetivos

1. Conhecer a lógica digital da porta NOU (NOR) e verificar sua operação lógica,
2. Obter a tabela verdade e analisar algumas propriedades dela.
3. Familiarizar-se com o Circuito Integrado TTL 74LS02.
4. Obtenção de uma porta NOU (NOR) de três entradas usando 3 portas NOU de duas entradas.

3.7.2. Informação preliminar

1. A porta NOU (NOR) fornece a saída exatamente como forma invertida de uma porta OU.
2. Simbolicamente, esta situação é mostrada com uma porta OU tendo um pequeno círculo na saída conforma a figura 3.51.
3. Se alguma das entradas for 1 (H), a saída será 0 (L) e, quando todas as entradas forem 0 (L), a saída será 1 (H) conforme a tabela 3.19.
4. Se as entradas da porta NOU estiverem conectadas, ela pode operar como um inversor, conforme a figura 3.52 e tabela 3.20.
5. O CI TTL 74LS02 e o CMOS 4001 contém 4 portas NOU de 2 entradas conforme a figura 3.53. Observe que a disposição interna das portas do CI CMOS é diferente do CI TTL.

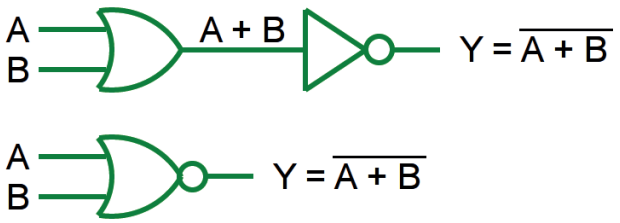


Figura 3.53 – Porta NOU de duas entradas.

Entradas		Saída	
A	B	A + B	(A + B)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Tabela 3.19 – Tabela verdade para a porta NOU de duas entradas.

Figura 3.54 – Porta NOU atuando como INVERSORA.

A	A'
0	1
1	0

Tabela 3.20 – Tabela verdade para a porta INVERSORA.

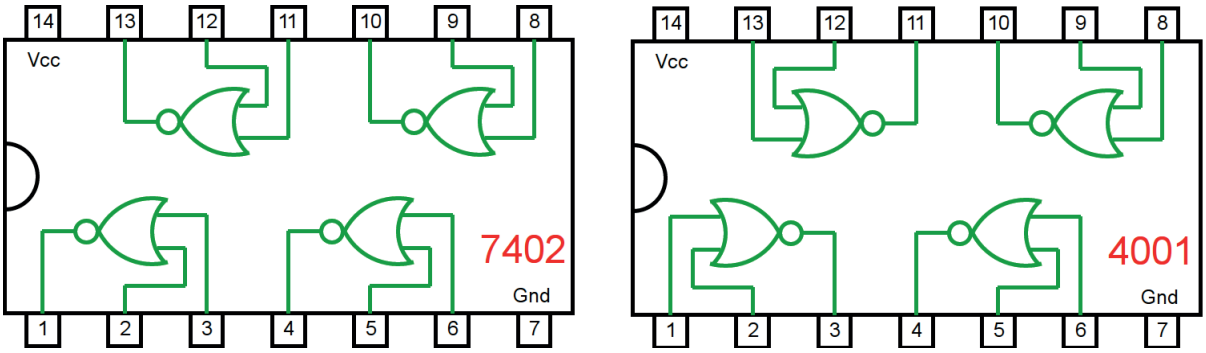


Figura 3.55 – Circuito integrado TTL e CMOS.

3.7.3. Experimento 1

Obter a tabela verdade da porta NOU de duas entradas.

Esquemas:

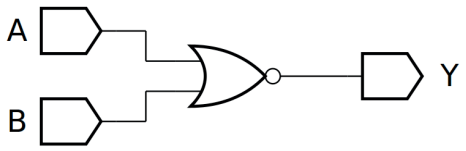


Figura 3.56 – Diagrama esquemático.

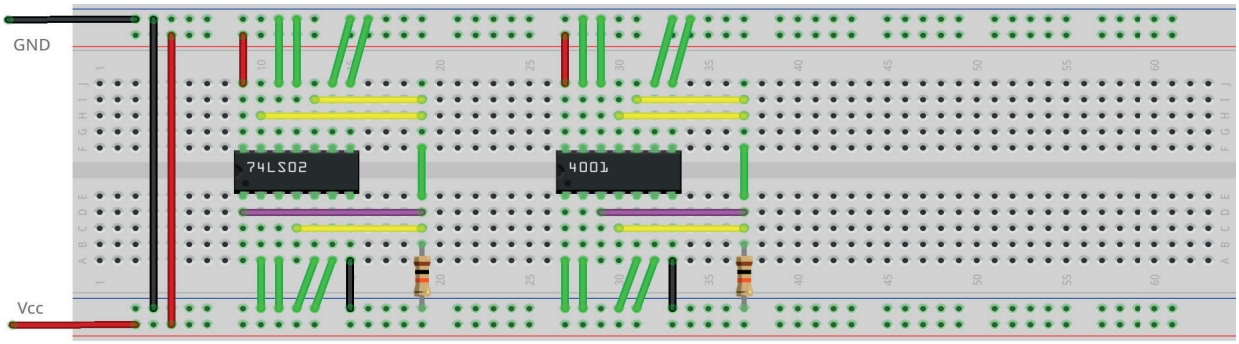


Figura 3.57 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.56, conforme mostrado na figura 3.57, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo. Não conecte os fios amarelos!
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.

Os procedimentos a seguir se referem ao teste da porta 1-2-3.

3. Conecte o cabo vermelho na fonte de energia.
4. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.21a, na coluna correspondente ao CI e linha corresponde a A =0 e B=0.
5. Desconecte o cabo vermelho na fonte de energia.
6. Mude o fio verde do terminal 2 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
7. Conecte o cabo vermelho na fonte de energia.

8. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.21a, na coluna correspondente ao CI e linha corresponde a A =0 e B=1.
9. Desconecte o cabo vermelho na fonte de energia.
10. Mude o fio verde do terminal 2 do CI do barramento de energia (vermelho, 1) para o barramento de terra (preto, 0). Mude o fio verde do terminal 3 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
11. Conecte o cabo vermelho na fonte de energia.
12. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.21a, na coluna correspondente ao CI e linha corresponde a A =1 e B=0.
13. Desconecte o cabo vermelho na fonte de energia.
14. Mude o fio verde do terminal 2 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
15. Conecte o cabo vermelho na fonte de energia.
16. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.21a, na coluna correspondente ao CI e linha corresponde a A =1 e B=1.
17. Desconecte o cabo vermelho na fonte de energia.
18. Preencha a 3ª coluna da tabela 3.21a com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.

Os procedimentos a seguir são opcionais e se referem aos testes das outras portas e outro CI.

19. Remova o fio roxo do pino 1 do CI e conecte no pino 4.
20. Repita os procedimentos semelhantes de 3 a 17 para a porta 4-5-6 e anote os valores na tabela 3.21b. Onde for terminal 2 use terminal 5. Onde for terminal 3 use terminal 6.
21. Remova o fio roxo do pino 4 do CI e conecte no pino 10.
22. Repita os procedimentos semelhantes de 3 a 17 para a porta 8-9-10 e anote os valores na tabela 3.21c. Onde for terminal 1 use terminal 8. Onde for terminal 3 use terminal 9.
23. Remova o fio roxo do pino 10 do CI e conecte no pino 13.
24. Repita os procedimentos semelhantes de 3 a 17 para a porta 11-12-13 e anote os valores na tabela 3.21d. Onde for terminal 2 use terminal 11. Onde for terminal 3 use terminal 12.
25. Repita os procedimentos semelhantes de 3 a 24 para o CI CMOS 4001. Lembre-se que as conexões internas são diferentes do TTL testado.

Os procedimentos a seguir são opcionais e se referem à simulação em computador.

26. Utilizando o software de simulação SmartSim, simule a porta NOU (NOR) de acordo com a figura 3.56. Compare a simulação com a tabela 3.19.
27. Utilizando o software de simulação Atanua, simule a porta NOU (NOR) conforme a figura 3.58. Compare a simulação com a tabela 3.19.
28. Utilizando o software de simulação Qucs, simule a porta NOU (NOR) conforme a figura 3.59. Compare os dados da tabela do Qucs com a tabela 3.19.

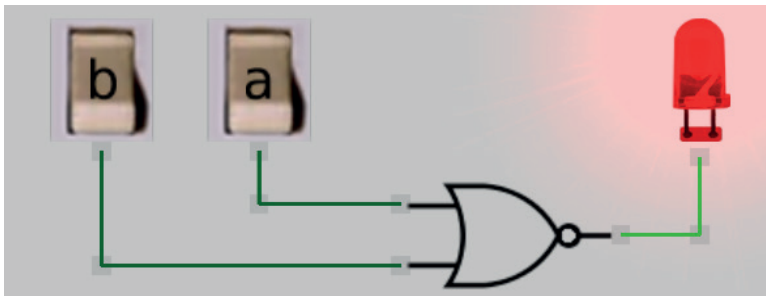


Figura 3.58 – Simulação da porta NOU de 2 entradas no Atanua.

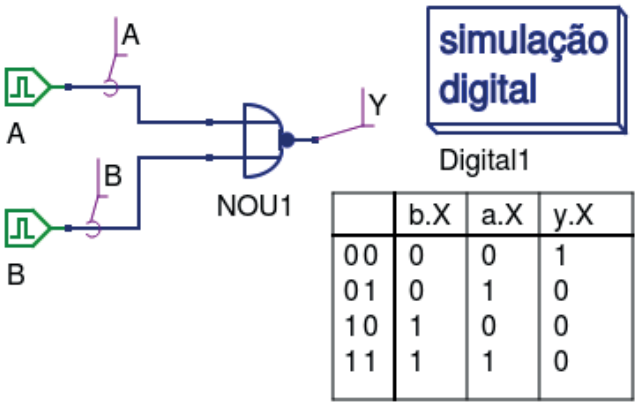


Figura 3.59 – Simulação da porta NOU de 2 entradas no Qucs.

Tabelas de dados:

A	B	Y = (A + B)'	Valor de tensão medido pelo voltímetro	
			Para o 74LS02	Para o 4001
0	0			
0	1			
1	0			
1	1			

Tabela 3.21a – Valores medidos na porta 1-2-3.

A	B	Y = (A + B)'	Valor de tensão medido pelo voltímetro	
			Para o 74LS02	Para o 4001
0	0			
0	1			
1	0			
1	1			

Tabela 3.21b – Valores medidos na porta 4-5-6.

A	B	Y = (A + B)'	Valor de tensão medido pelo voltímetro	
			Para o 74LS02	Para o 4001
0	0			
0	1			
1	0			
1	1			

Tabela 3.21c – Valores medidos na porta 8-9-10.

A	B	Y = (A + B)'	Valor de tensão medido pelo voltímetro	
			Para o 74LS02	Para o 4001
0	0			
0	1			
1	0			
1	1			

Tabela 3.21d – Valores medidos na porta 11-12-13.

3.7.4. Experimento 2

Usando a porta NOU como porta inversora.

Esquemas:

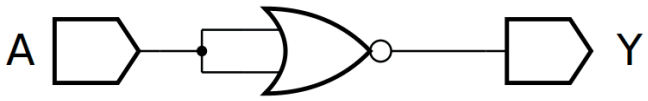


Figura 3.60 – Diagrama esquemático.

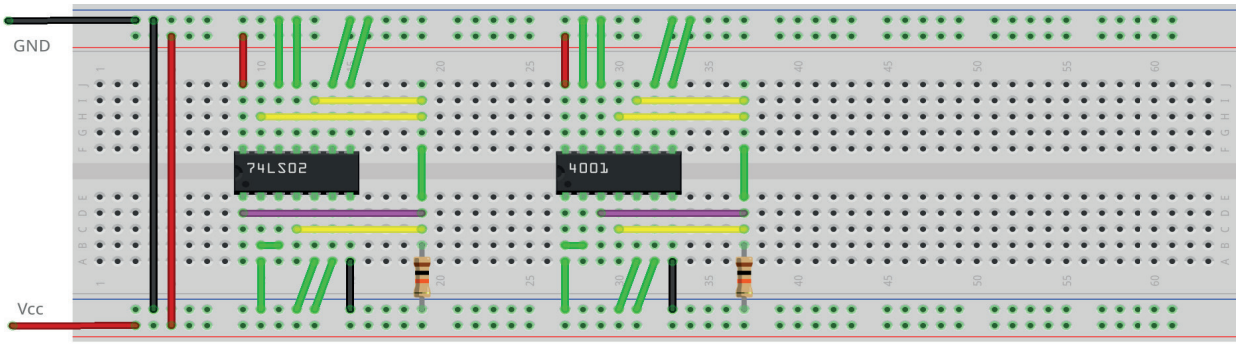


Figura 3.61 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.60, conforme mostrado na figura 3.61, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo. Não conecte os fios amarelos!
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho na fonte de energia.
4. Faça uma medida de tensão sobre o resistor. Registre na tabela 3.22: Se o valor de tensão for próximo de Vcc registre “1” na linha corresponde a A =0, caso contrário se a tensão for próximo de Gnd, registre “0”.
5. Desconecte o cabo vermelho na fonte de energia.
6. Mude o fio verde do terminal 2 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
7. Conecte o cabo vermelho na fonte de energia.

8. Faça uma medida de tensão sobre o resistor. Registre na tabela 3.22: Se o valor de tensão for próximo de Vcc registre “1” na linha corresponde a A =1, caso contrário se a tensão for próximo de Gnd, registre “0”.
9. Desconecte o cabo vermelho na fonte de energia.
10. Repita os procedimentos anteriores para o CI 4011. Lembre-se que as conexões internas são diferentes do TTL testado.
11. Utilizando o software de simulação SmartSim, simule a porta NOU (NOR) de acordo com a figura 3.60. Compare a simulação com a tabela 3.7.
12. Utilizando o software de simulação Atanua, simule a porta NOU (NOR) conforme a figura 3.62. Compare a simulação com a tabela 3.20 e 3.22.
13. Utilizando o software de simulação Qucs, simule a porta NOU (NOR) conforme a figura 3.63. Compare os dados da tabela do Qucs com a tabela 3.20 e 3.22.

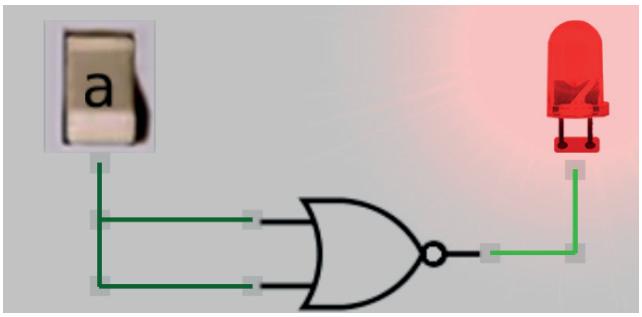


Figura 3.62 – Simulação da porta NOU como inversor no Atanua.

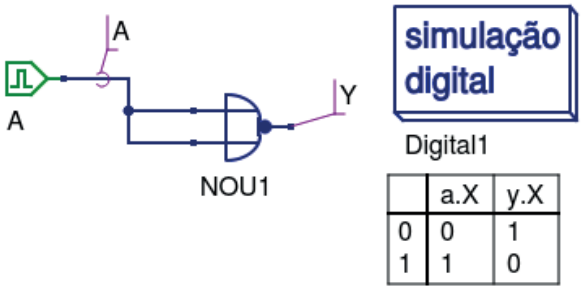


Figura 3.63 – Simulação da porta NOU como inversor no Qucs.

Tabelas de dados:

A	Y = (A)’
0	
1	

Tabela 3.22 – Tabela verdade da porta NOU atuando como porta inversora.

3.7.5. Experimento 3

Obter a tabela verdade da porta NOU de três entradas, construída a partir de portas NOU de duas entradas.

Esquemas:

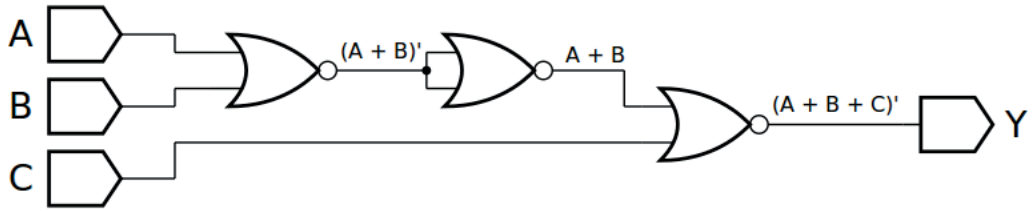


Figura 3.64 – Diagrama esquemático.

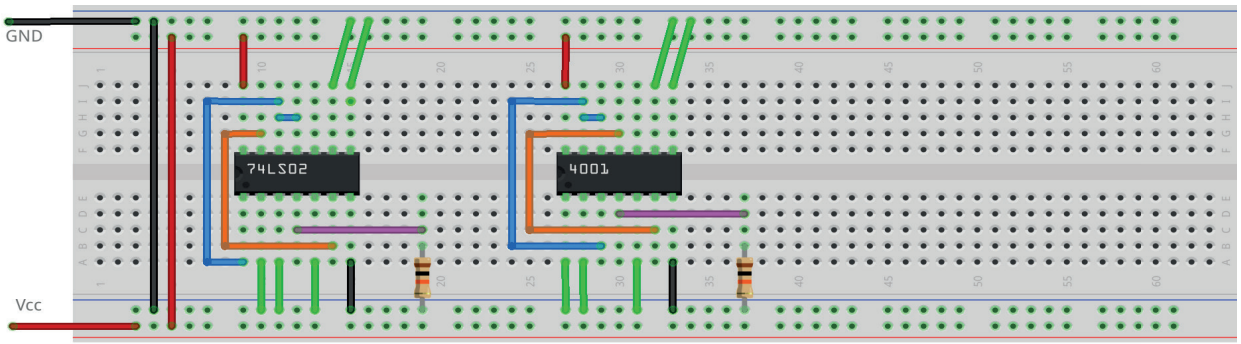


Figura 3.65 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.64, conforme mostrado na figura 3.65, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo.
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Na tabela 3.23 as colunas A, B e C são as entradas. “0” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de terra (gnd, preto) do protoboard. “1” indica que o fio verde conectado nesta entrada deve ser conectado no barramento de energia (Vcc, vermelho) do protoboard. Antes de cada mudança nos fios verdes, o cabo de energia (vermelho) deve ser desconectado da fonte de energia. A conexão do cabo na fonte de energia somente deverá ser feita após a mudança dos fios verdes.
4. Meça o valor de tensão sobre o resistor e registre na coluna adequada da tabela 3.23.

5. Preencha a coluna Y da tabela 3.23 com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.
6. Repita os itens 3 a 5 para todas as combinações na tabela 3.18.
7. Repita os itens 3 a 6 para o CI 4001. Lembre-se que as conexões internas são diferentes do TTL testado.
8. Simule, no computador, o circuito da figura 3.62 utilizando os programas SmartSim e Atanua (figura 3.66) e Qucs (figura 3.67). Compare os resultados com a tabela 3.23.

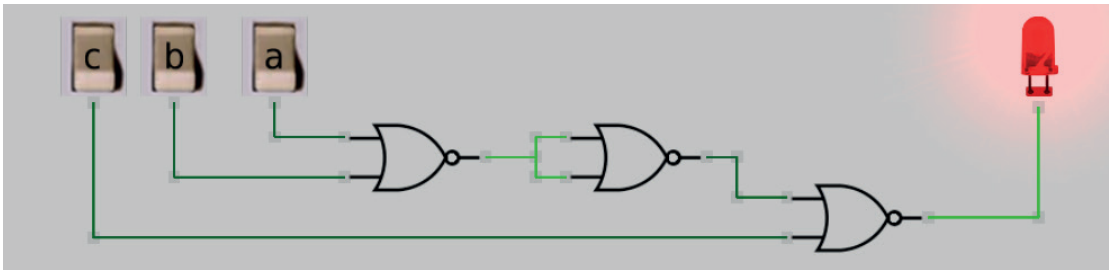
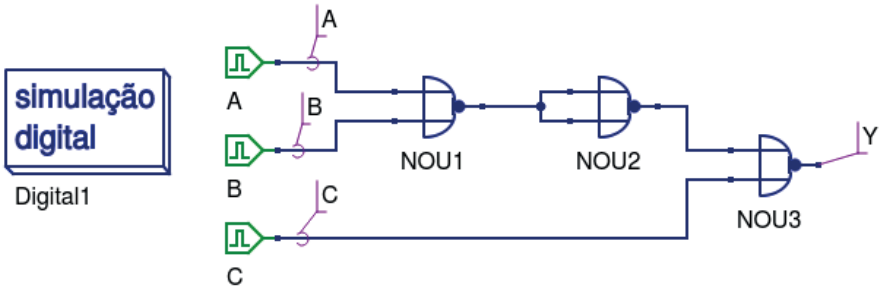


Figura 3.66 – Simulação da porta NOU de 3 entradas no Atanua.



	c.X	b.X	a.X	y.X
000	0	0	0	1
001	0	0	1	0
010	0	1	0	0
011	0	1	1	0
100	1	0	0	0
101	1	0	1	0
110	1	1	0	0
111	1	1	1	0

Figura 3.67 – Simulação da porta NOU de 3 entradas no Qucs.

Tabelas de dados:

A	B	C	Y = (A + B +C)’	Valor de tensão medido pelo voltímetro	
				Para o 74LS02	Para o 4001
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 3.23 – Tabela verdade para porta NOU de 3 entradas.

3.8. Exame da porta OU-EX (XOR) ou OU-Exclusivo

3.8.1. Objetivos

1. Conhecer a lógica digital da porta OU-EX (XOR) e verificar sua operação lógica.
2. Familiarizar-se com os circuitos integrados TTL 7486 e CMOS 4030.

3.8.2. Informação preliminar

1. A porta OU-EX (XOR) de 2-entradas, figura 3.65, compara dois bits. Se os bits forem diferentes um do outro, a saída será 1 (H). Se os bits forem os mesmos, a saída será 0 (L), conforme a tabela 3.24.
2. A porta OU-EX (XOR) pode ser formada não apenas pela ajuda de outras portas, mas também por circuitos integrados padrão.
3. Esses circuitos também são chamados de Comparadores de Iniquidade.
4. O código de paridade, que é um código de igualdade, é verificado por uma porta OU-EX (XOR).
5. O CI TTL7486 e o CI CMOS 4070 contêm quatro portas OU-EX (XOR) de 2 entradas, conforme a figura 3.66.

Figura 3.68 – Porta OU-Exclusivo de duas entradas.

A	B	Y = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 3.24 – Tabela verdade para uma porta OU-Exclusivo de duas entradas.

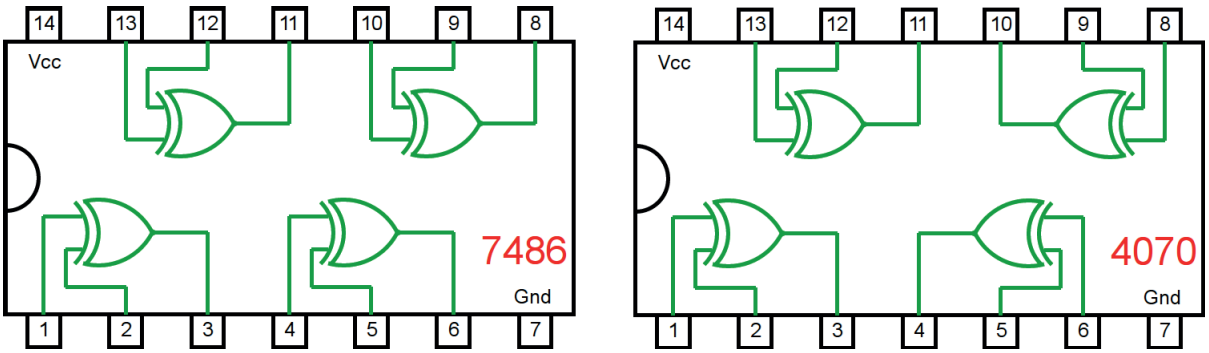


Figura 3.69 – Circuito integrado TTL e CMOS.

3.8.3. Experimento 1

Obter a tabela verdade da porta OU-Exclusivo de duas entradas.

Esquemas:

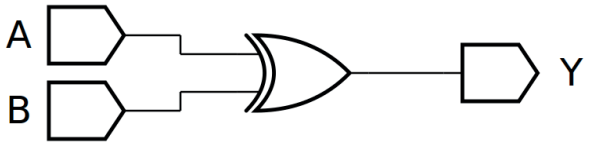


Figura 3.70 – Diagrama esquemático.

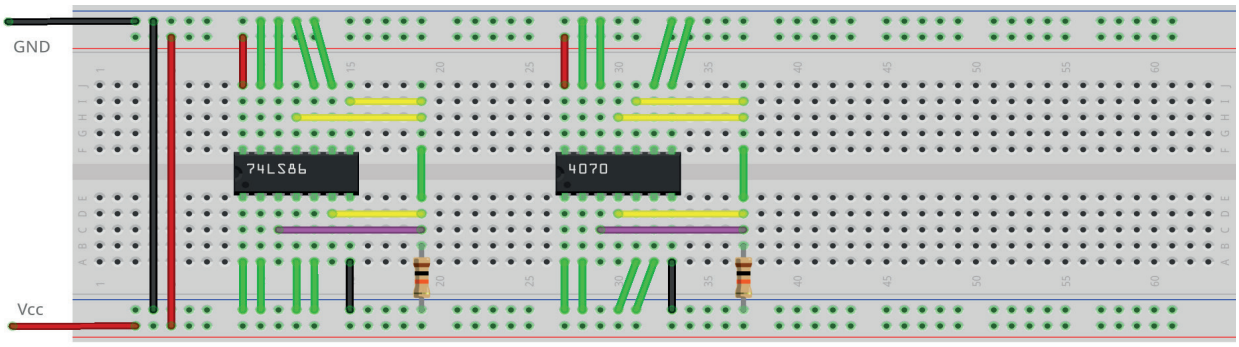


Figura 3.71 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.70, conforme mostrado na figura 3.71, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo. Não conecte os fios amarelos!
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.

Os procedimentos a seguir se referem ao teste da porta 1-2-3.

3. Conecte o cabo vermelho na fonte de energia.
4. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.25a, na coluna correspondente ao CI e linha corresponde a A =0 e B=0.
5. Desconecte o cabo vermelho na fonte de energia.
6. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
7. Conecte o cabo vermelho na fonte de energia.

8. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.25a, na coluna correspondente ao CI e linha corresponde a A =0 e B=1.
9. Desconecte o cabo vermelho na fonte de energia.
10. Mude o fio verde do terminal 1 do CI do barramento de energia (vermelho, 1) para o barramento de terra (preto, 0). Mude o fio verde do terminal 2 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
11. Conecte o cabo vermelho na fonte de energia.
12. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.25a, na coluna correspondente ao CI e linha corresponde a A =1 e B=0.
13. Desconecte o cabo vermelho na fonte de energia.
14. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
15. Conecte o cabo vermelho na fonte de energia.
16. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.25a, na coluna correspondente ao CI e linha corresponde a A =1 e B=1.
17. Desconecte o cabo vermelho na fonte de energia.
18. Preencha a 3ª coluna da tabela 3.24a com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.

Os procedimentos a seguir são opcionais e se referem aos testes das outras portas e outro CI.

19. Remova o fio roxo do pino 3 do CI e conecte no pino 6.
20. Repita os procedimentos semelhantes de 3 a 18 para a porta 4-5-6 e anote os valores na tabela 3.25b. Onde for terminal 1 use terminal 4. Onde for terminal 2 use terminal 5.
21. Remova o fio roxo do pino 6 do CI e conecte no pino 8.
22. Repita os procedimentos semelhantes de 3 a 18 para a porta 8-9-10 e anote os valores na tabela 3.25c. Onde for terminal 1 use terminal 9. Onde for terminal 2 use terminal 10.
23. Remova o fio roxo do pino 8 do CI e conecte no pino 11.
24. Repita os procedimentos semelhantes de 3 a 18 para a porta 11-12-13 e anote os valores na tabela 3.25. Onde for terminal 1 use terminal 12. Onde for terminal 2 use terminal 13.
25. Repita os procedimentos semelhantes de 3 a 24 para o CI CMOS 4070. Lembre-se que as conexões internas são diferentes do TTL testado.

Os procedimentos a seguir são opcionais e se referem à simulação em computador.

26. Utilizando o software de simulação SmartSim, simule a porta OU-Ex (XOR) de acordo com a figura 3.70. Compare a simulação com a tabela 3.24.
27. Utilizando o software de simulação Atanua, simule a porta OU-Ex (XOR) de acordo com a figura 3.72.
28. Utilizando o software de simulação Qucs, simule a porta OU-Ex (XOR) e construa a tabela verdade de acordo com a figura 3.73. Compare os dados da tabela do Qucs com a tabela 3.24.

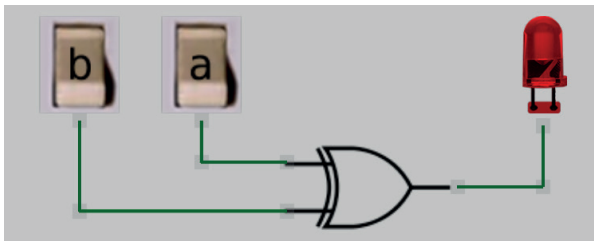


Figura 3.72 – Simulação da porta OU-Ex de duas entradas no Atanua.

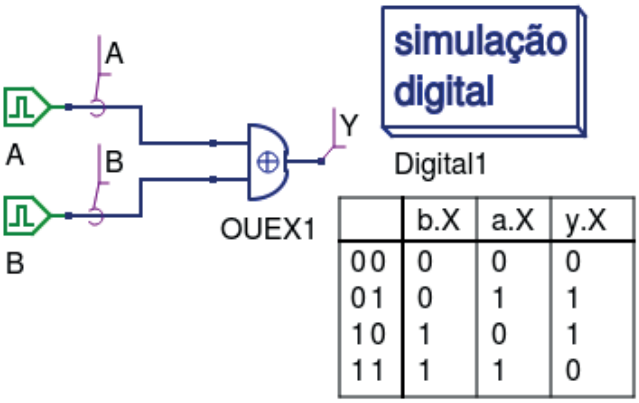


Figura 3.73 – Simulação da porta OU-Ex de duas entradas no Qucs.

Tabelas de dados:

A	B	Y = A⊕B	Valor de tensão medido pelo voltímetro	
			Para o 74LS86	Para o 4070
0	0			
0	1			
1	0			
1	1			

Tabela 3.25a – Valores medidos na porta 1-2-3.

A	B	Y = A⊕B	Valor de tensão medido pelo voltímetro	
			Para o 74LS86	Para o 4070
0	0			
0	1			
1	0			
1	1			

Tabela 3.25b – Valores medidos na porta 4-5-6.

A	B	Y = A⊕B	Valor de tensão medido pelo voltímetro	
			Para o 74LS86	Para o 4070
0	0			
0	1			
1	0			
1	1			

Tabela 3.25c – Valores medidos na porta 8-9-10.

A	B	Y = A⊕B	Valor de tensão medido pelo voltímetro	
			Para o 74LS86	Para o 4070
0	0			
0	1			
1	0			
1	1			

Tabela 3.25d – Valores medidos na porta 11-12-13.

3.9. Exame da porta NOU-EX (XNOR) ou Não-OU-Exclusivo

3.9.1. Objetivos

1. Conhecer a lógica digital da porta NOU-EX (XNOR) e verificar sua operação lógica.
2. Observar o circuito gerador de bit de paridade e derivar sua tabela de verdade.
3. Formar uma porta NOU-EX usando uma porta OU-EX e uma porta INVERSORA.

3.9.2. Informação preliminar

1. A porta NOU-EX (XNOR) de 2 entradas, figura 3.71, compara dois bits. Se os bits forem iguais, a saída será 1 (H). Se os bits forem diferentes um do outro, a saída será 0 (L), conforme a tabela 3.26.
2. A porta NOU-EX (XNOR) é a forma invertida da porta OU-EX (XOR).
3. A porta NOU-EX pode ser formado não apenas pela ajuda de outras portas, mas também por circuitos integrados padrão.
4. O código de paridade, ou seja, o código de igualdade, é produzido com uma porta NOU-EX.
5. O CI TTL 74HC266 e o CI CMOS 4077 contêm quatro portas NOU-EX de duas entradas, figura 3.75. Como as saídas do CI 74HC266 são de dreno aberto, para observar um sinal lógico de uma saída, é necessário conectar um resistor de pull-up da saída à fonte de alimentação.

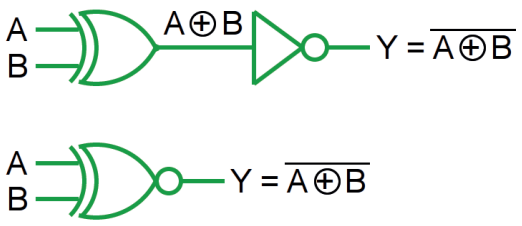


Figura 3.74 – Porta NOU-Ex de duas entradas.

A	B	Y = A ⊕ B	Y = (A ⊕ B)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Tabela 3.26 – Tabela verdade da porta NOU-Ex de duas entradas.

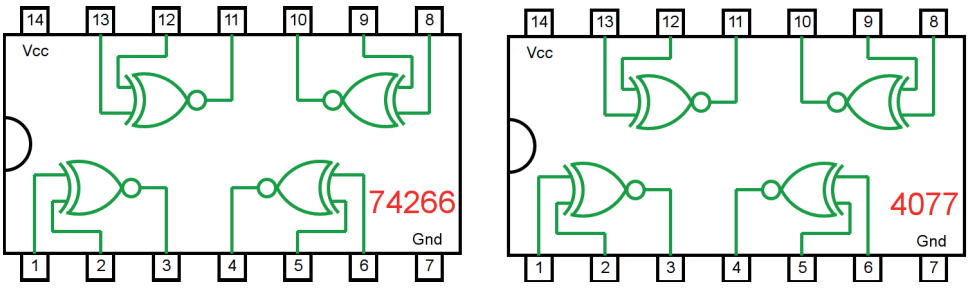


Figura 3.75 – Circuito integrado TTL e CMOS.

3.9.3. Experimento 1

Obter a tabela verdade da porta NOU-Exclusivo de duas entradas.

Esquemas:

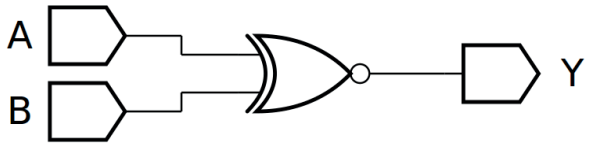


Figura 3.76 – Diagrama esquemático.

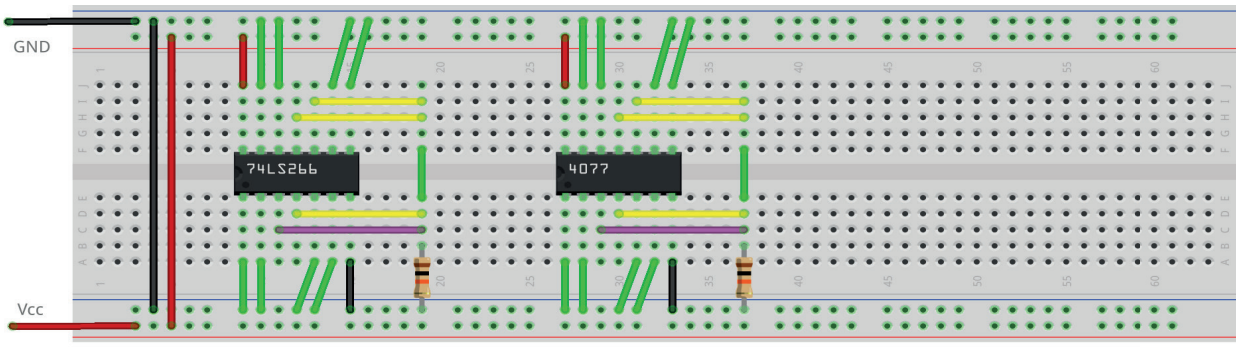


Figura 3.77 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 3.76, conforme mostrado na figura 3.77, na placa de montagem.
   a) Coloque o circuito integrado na posição indicada.
   b) Coloque o resistor na posição indicada.
   c) Conecte os fios vermelhos e pretos.
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo. Não conecte os fios amarelos!
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.

Os procedimentos a seguir se referem ao teste da porta 1-2-3.

3. Conecte o cabo vermelho na fonte de energia.
4. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.27a, na coluna correspondente ao CI e linha corresponde a A =0 e B=0.
5. Desconecte o cabo vermelho na fonte de energia.
6. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
7. Conecte o cabo vermelho na fonte de energia.

8. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.27a, na coluna correspondente ao CI e linha corresponde a A =0 e B=1.
9. Desconecte o cabo vermelho na fonte de energia.
10. Mude o fio verde do terminal 1 do CI do barramento de energia (vermelho, 1) para o barramento de terra (preto, 0). Mude o fio verde do terminal 2 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
11. Conecte o cabo vermelho na fonte de energia.
12. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.27a, na coluna correspondente ao CI e linha corresponde a A =1 e B=0.
13. Desconecte o cabo vermelho na fonte de energia.
14. Mude o fio verde do terminal 1 do CI do barramento de terra (preto, 0) para o barramento de energia (vermelho, 1).
15. Conecte o cabo vermelho na fonte de energia.
16. Faça uma medida de tensão sobre o resistor. Anote o valor na tabela 3.27a, na coluna correspondente ao CI e linha corresponde a A =1 e B=1.
17. Desconecte o cabo vermelho na fonte de energia.
18. Preencha a 3ª coluna da tabela 3.27a com “1” se o valor de tensão medido for próximo de +5Vcc, preencha com “0” se o valor de tensão medido for próximo de 0V.

Os procedimentos a seguir são opcionais e se referem aos testes das outras portas e outro CI.

19. Remova o fio roxo do pino 3 do CI e conecte no pino 4.
20. Repita os procedimentos semelhantes de 3 a 18 para a porta 4-5-6 e anote os valores na tabela 3.27b. Onde for terminal 1 use terminal 5. Onde for terminal 2 use terminal 5.
21. Remova o fio roxo do pino 6 do CI e conecte no pino 10.
22. Repita os procedimentos semelhantes de 3 a 18 para a porta 8-9-10 e anote os valores na tabela 3.27c. Onde for terminal 1 use terminal 8. Onde for terminal 2 use terminal 9.
23. Remova o fio roxo do pino 10 do CI e conecte no pino 11.
24. Repita os procedimentos semelhantes de 3 a 18 para a porta 11-12-13 e anote os valores na tabela 3.27d. Onde for terminal 1 use terminal 12. Onde for terminal 2 use terminal 13.
25. Repita os procedimentos semelhantes de 3 a 24 para o CI CMOS 4077. Lembre-se que as conexões internas são as mesmas do TTL testado.

Os procedimentos a seguir são opcionais e se referem à simulação em computador.

26. Utilizando o software de simulação SmartSim, simule a porta NOU-Ex (XNOR) de acordo com a figura 3.76. Compare a simulação com a tabela 3.26.
27. Utilizando o software de simulação Atanua, simule a porta NOU-Ex (XNOR) de acordo com a figura 3.78.
28. Utilizando o software de simulação Qucs, simule a porta NOU-Ex (XNOR) e construa a tabela verdade de acordo com a figura 3.79. Compare os dados da tabela do Qucs com a tabela 3.26.

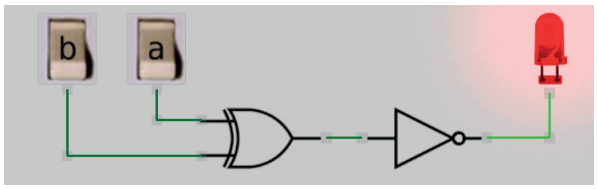


Figura 3.78 – Simulação da porta NOU-Ex de duas entradas no Atanua.

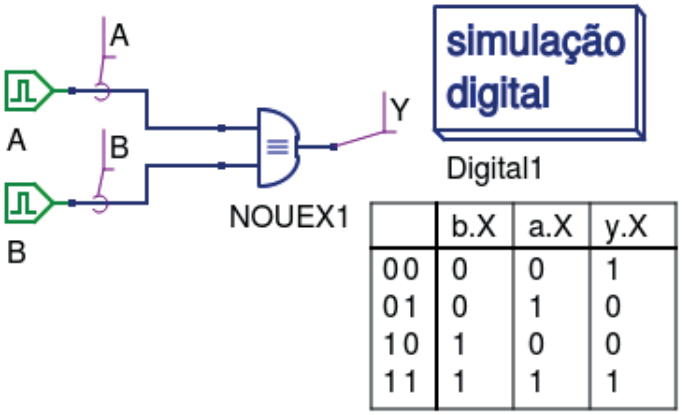


Figura 3.79 – Simulação da porta NOU-Ex de duas entradas no Qucs.

Tabelas de dados:

A	B	Y = A⊕B	Valor de tensão medido pelo voltímetro	
			Para o 74LS266	Para o 4077
0	0			
0	1			
1	0			
1	1			

Tabela 3.27a – Valores medidos na porta 1-2-3.

A	B	Y = A⊕B	Valor de tensão medido pelo voltímetro	
			Para o 74LS266	Para o 4077
0	0			
0	1			
1	0			
1	1			

Tabela 3.27b – Valores medidos na porta 4-5-6.

A	B	Y = A⊕B	Valor de tensão medido pelo voltímetro	
			Para o 74LS26	Para o 4077
0	0			
0	1			
1	0			
1	1			

Tabela 3.27c – Valores medidos na porta 8-9-10.

A	B	Y = A⊕B	Valor de tensão medido pelo voltímetro	
			Para o 74LS266	Para o 4077
0	0			
0	1			
1	0			
1	1			

Tabela 3.27d – Valores medidos na porta 11-12-13.

3.10. Simulação das portas lógicas em Arduíno

Esquemas:

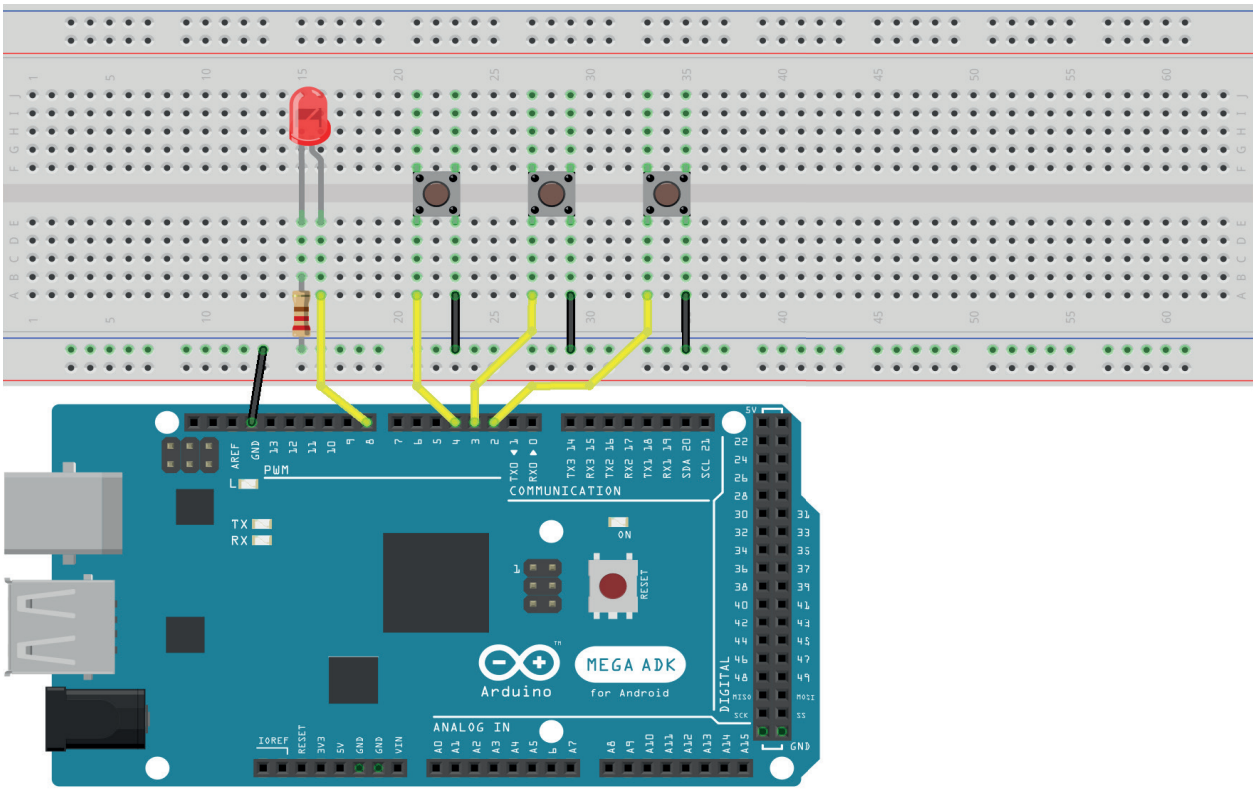


Figura 3.80 – Placa de montagem com componentes e placa do Arduíno Mega.

Procedimento:

1. Faça a montagem dos componentes conforme a figura 3.77.
   a) Insira os botões, o led e o resistor.
   b) Conecte os fios de terra (preto).
   c). Conecte os fios de entrada e saída (amarelos).
2. Digite o programa na interface do Arduíno.
3. Transfira o programa para o Arduíno.
4. Faça a simulação da porta E de 2 entradas pressionando os botões conectados aos pinos 2 (A) e ao pino 3 (B). A saída é dado pelo LED: aceso representa saída “1”, apagado representa a saída “0”. Compare com a tabela 3.1.
5. Observe a linha da instrução if. Ela contém a lógica a executar (destacado em negrito). Para simular uma porta E de 3 entradas utilize a seguinte lógica: ((A&&B)&&C)).
6. Faça a simulação da porta E de 3 entradas pressionando os botões conectados aos pinos 2 (A), ao pino 3 (B) e ao pino 4 (C). A saída é dado pelo LED: aceso representa saída “1”, apagado representa a saída “0”. Compare com a tabela 3.2.
7. Faça a simulação para as portas OU, NE, NOU, OU-Ex e NOU-Ex para duas e três entradas e compare com as respectivas tabelas verdades dos experimentos anteriores.

Programa para o Arduíno:

```
/* ---------------------------------------------
Universal Logic Gates Implementer with Arduino
Created by Panos Agiakatsikas
Date: 22/1/2017
-----------------------------------------------
The logic gates
Buffer     (A) (Also(!A&&!A))    output [0]
NOT        (!A)                  output [1]
AND        (A&&B)                output [0001]
OR         (A||B)                output [0111]
NAND       (!(A&&B))             output [1110]
NOR        (!(A||B))             output [1000]
XOR        (!A&&B)||(A&&!B)      output [0110]
XNOR       (A&&B)||(!A&&!B)      output [1001]
-----------------------------------------------*/

int buttonPin1 = 2; // Button 1
int buttonPin2 = 3; // Button 2
int buttonPin3 = 4; // Button 3
int LEDred = 8; // Out to pin
int A; // variable for input state of the button 1
int B; // variable for input state of the button 2
int C; // variable for input state of the button 3

void setup() {
  pinMode(LEDred, OUTPUT); // set led as output
// set pins as inputs with internal pull-up resistor,
// the open button logic state for all will be HIGH
  pinMode(buttonPin1, INPUT_PULLUP);
  pinMode(buttonPin2, INPUT_PULLUP);
  pinMode(buttonPin3, INPUT_PULLUP);
}

void loop(){
  A = digitalRead(buttonPin1);
  B = digitalRead(buttonPin2);
  C = digitalRead(buttonPin3);
  A=!A; // inverse the HIGH to LOW from the pull-up resistor
  B=!B; // inverse the HIGH to LOW from the pull-up resistor
```

```
  C=!C; // inverse the HIGH to LOW from the pull-up resistor
  if (A&&B){ // put here your logic statement
    digitalWrite(LEDred, HIGH);
  } else {
    digitalWrite(LEDred, LOW);
  }
}
```

Capítulo 4.
Leis e teorias da álgebra boolena

4.1. Objetivos

1. Analisar as relações entre operações lógicas,
2. Verificar as leis e teoremas da álgebra booleana.

4.2. Informação preliminar

1. O matemático inglês George BOOLE (1815-1864) analisou as operações lógicas em matemática, apresentou e comprovou algumas leis e teoremas que são chamados pelo seu nome.
2. Em uma operação lógica, VERDADEIRO ou seu equivalente em tensão H (“HIGH”, alto) é representado por “1” e FALSO ou seu equivalente em tensão L (“LOW”, baixo) é representado por “0”.
3. Para CIs TTL, o nível alto H (1) representa tensões entre 2,4 V e 5 V, e o nível baixo L (0) representa tensões entre 0 V e 0,4 V.
4. Lei Comutativa: $A + B = B + A$
   $A . B = B . A$
5. Lei Associativa: $A + B + C = (A + B) + C = A + (B + C)$
   $A . B . C = (A . B) . C = A . (B . C)$
6. Lei Distributiva: $A . (B + C) = (A . B) + (A . C)$
   $A + (B . C) = (A + B) . (A + C)$
7. Lei Idempotente: $A + A = A$
   $A . A = A$
8. Lei E $A . 1 = A$
   $A . 0 = 0$
9. Lei OU: $A + 0 = A$
   $A + 1 = 1$
10. Complementos: $A + A' = 1$
    $A . A' = 0$
11. Lei da Involução: $(A')' = A$
    $((A + B)')' = A + B$
    $((A . B)')' = A . B$
12. Lei de Absorção: $A + (A . B) = A$
    $A . (A + B) = A$
13. Teoremas de De Morgan $(A . B)' = A' + B'$
    $(A + B)' = A' . B'$

4.3. Exame da Lei Comutativa

4.3.1. Experimento 1

Obter a tabela verdade para a equação: A + B = B + A

Esquemas:

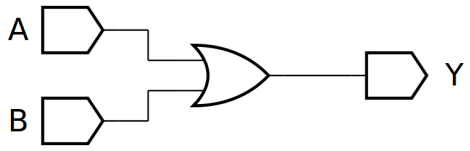


Figura 4.1 – Diagrama esquemático.

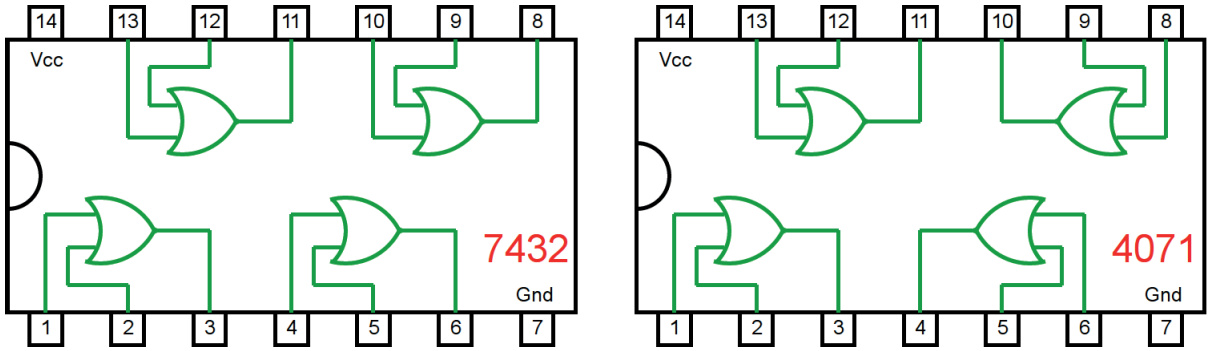


Figura 4.2 – Circuitos integrados TTL e CMOS.

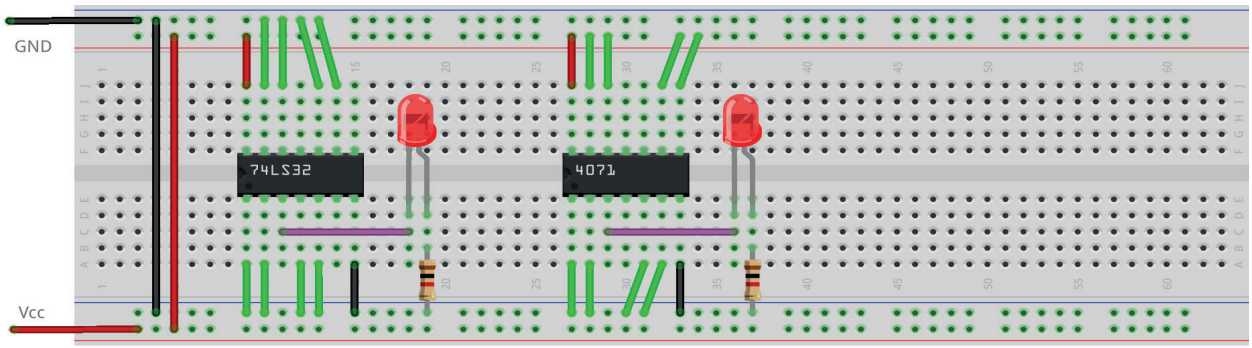


Figura 4.3 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.1, conforme mostrado na figura 4.3, na placa de montagem.
   a) Coloque o circuito integrado (TTL e/ou CMOS da figura 4.2) na posição indicada.
   b) Coloque o led e o resistor de 1KΩ na posição indicada.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo.
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.1 considere:
   a) Para a lógica “0” da variável A ou B, o fio verde deve estar conectado ao terminal do CI (1 ou 2) e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A ou B, o fio verde deve estar conectado ao terminal do CI (1 ou 2) e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Considere o pino 1 como entrada A e o pino 2 como entrada B.
6. Preencha a coluna A + B da tabela 4.1, para todos as combinações descritas.
7. Considere o pino 1 como entrada B e o pino 2 como entrada A.
8. Preencha a coluna B + A da tabela 4.1, para todos as combinações descritas.
9. Simule o circuito da figura 4.1 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.1.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	A + B	B + A
0	0		
0	1		
1	0		
1	1		

Tabela 4.1

4.3.2. Experimento 2

Obter a tabela verdade para a equação: A . B = B . A

Esquemas:

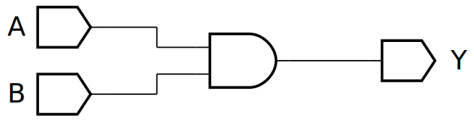


Figura 4.4 – Diagrama esquemático.

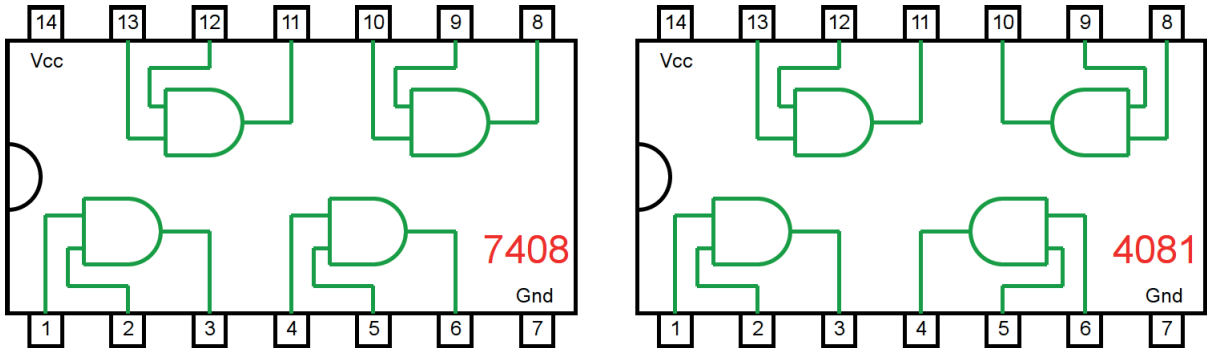


Figura 4.5 – Circuitos integrados TTL e CMOS.

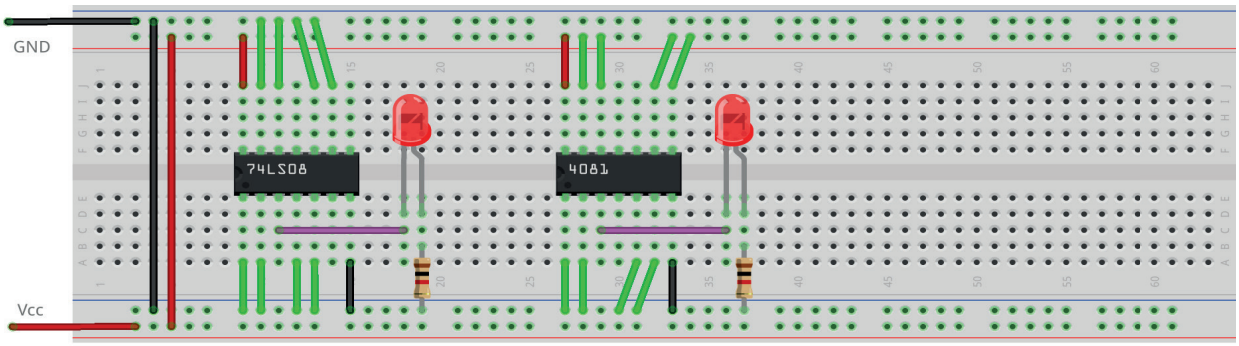


Figura 4.6 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.4, conforme mostrado na figura 4.6, na placa de montagem.
   a) Coloque o circuito integrado (TTL e/ou CMOS da figura 4.5) na posição indicada.
   b) Coloque o led e o resistor de 1KΩ na posição indicada.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo.
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.2 considere:
   a) Para a lógica “0” da variável A ou B, o fio verde deve estar conectado ao terminal do CI (1 ou 2) e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A ou B, o fio verde deve estar conectado ao terminal do CI (1 ou 2) e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Considere o pino 1 como entrada A e o pino 2 como entrada B.
6. Preencha a coluna A . B da tabela 4.2, para todos as combinações descritas.
7. Considere o pino 1 como entrada B e o pino 2 como entrada A.
8. Preencha a coluna B . A da tabela 4.2, para todos as combinações descritas.
9. Simule o circuito da figura 4.4 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.2.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	A . B	B . A
0	0		
0	1		
1	0		
1	1		

Tabela 4.2

4.4. Exame da Lei Associativa

4.4.1. Experimento 1

Obter a tabela verdade para a equação: A + B + C = (A + B) + C

Esquemas:

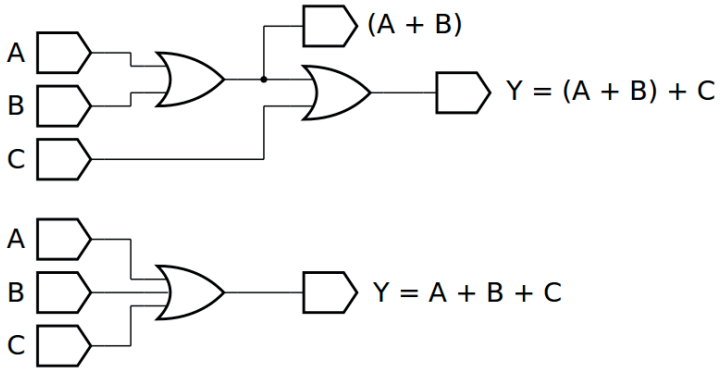


Figura 4.7 – Diagrama esquemático.

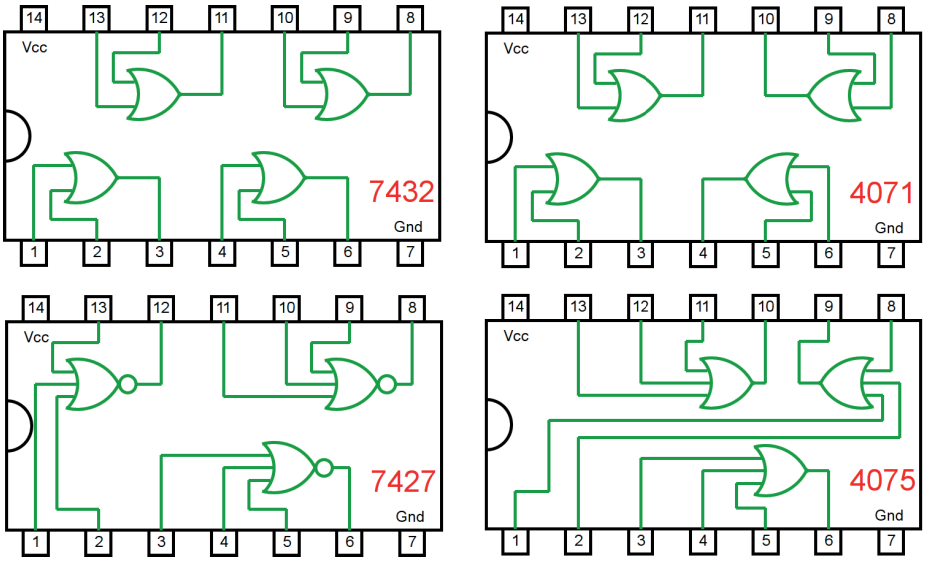


Figura 4.8 – Circuitos integrados TTL e CMOS.

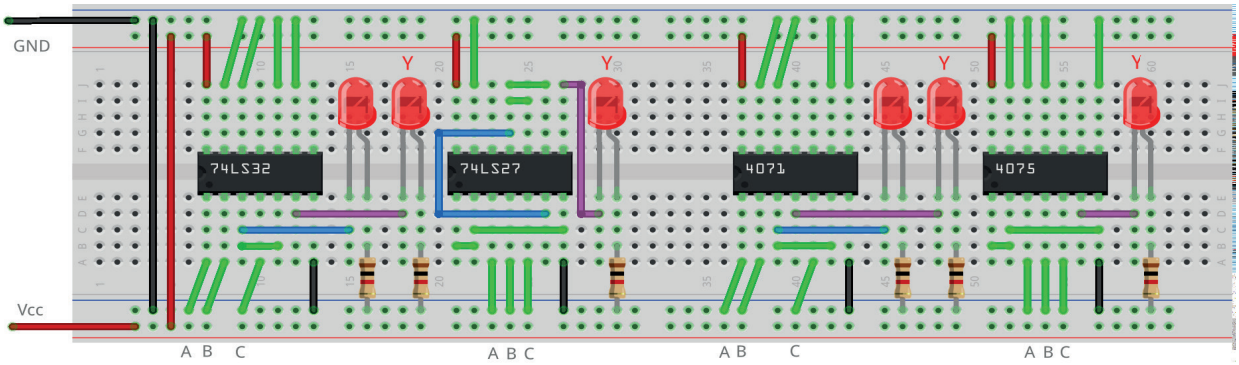


Figura 4.9 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.7, conforme mostrado na figura 4.9, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.8) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saída Y) e o fio azul (valor intermediário A + B).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.3 considere:
   a) Para a lógica "0" da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica "1" da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a "1", caso contrário, se o LED estiver apagado então a saída é igual a "0".
5. Preencha a tabela 4.3 para todas as combinações descritas.
6. Simule o circuito da figura 4.7 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.3.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	C	A + B	Y = (A + B) + C	Y = A + B + C
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 4.3

4.4.2. Experimento 2

Obter a tabela verdade para a equação: A + B + C = A + (B + C)

Esquemas:

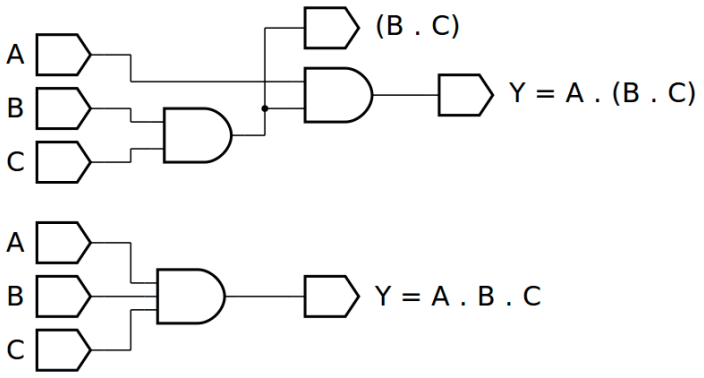


Figura 4.10 – Diagrama esquemático.

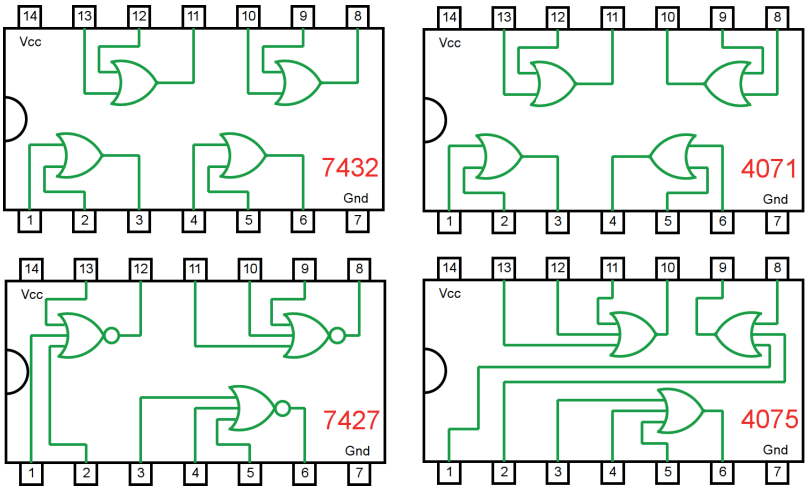


Figura 4.11 – Circuitos integrados TTL e CMOS.

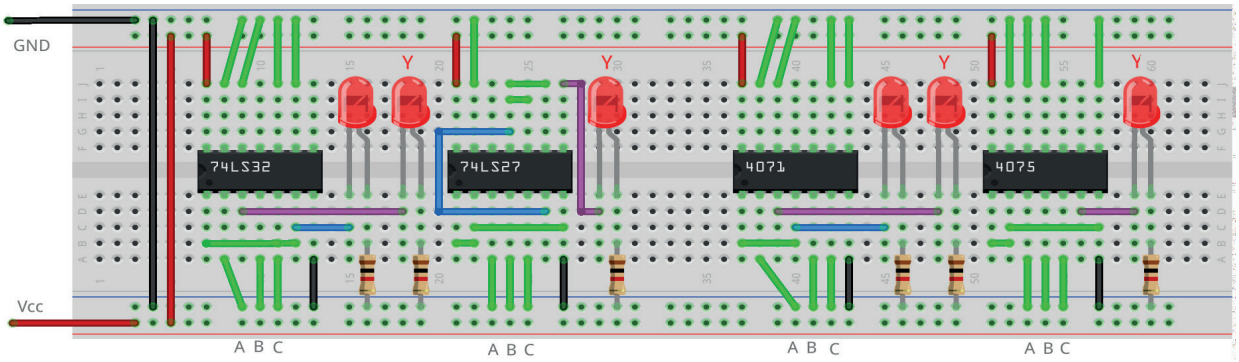


Figura 4.12 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.10, conforme mostrado na figura 4.12, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.11) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saída Y) e o fio azul (valor intermediário A + B).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.4 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.4 para todos as combinações descritas.
6. Simule o circuito da figura 4.10 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.4.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	C	B + C	Y = A + (B + C)	Y = A + B + C
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 4.4

4.4.3. Experimento 3

Obter a tabela verdade para a equação: A . B . C = (A . B) . C

Esquemas:

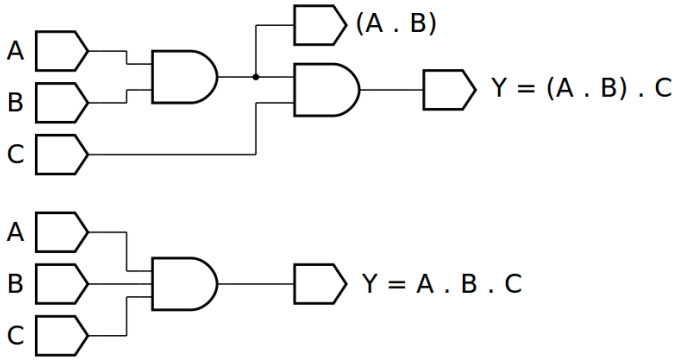


Figura 4.13 – Diagrama esquemático.

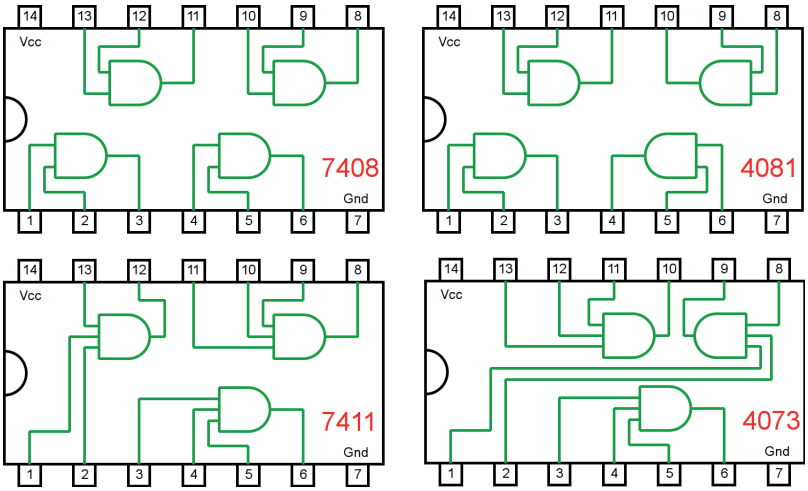


Figura 4.14 – Circuitos integrados TTL e CMOS.

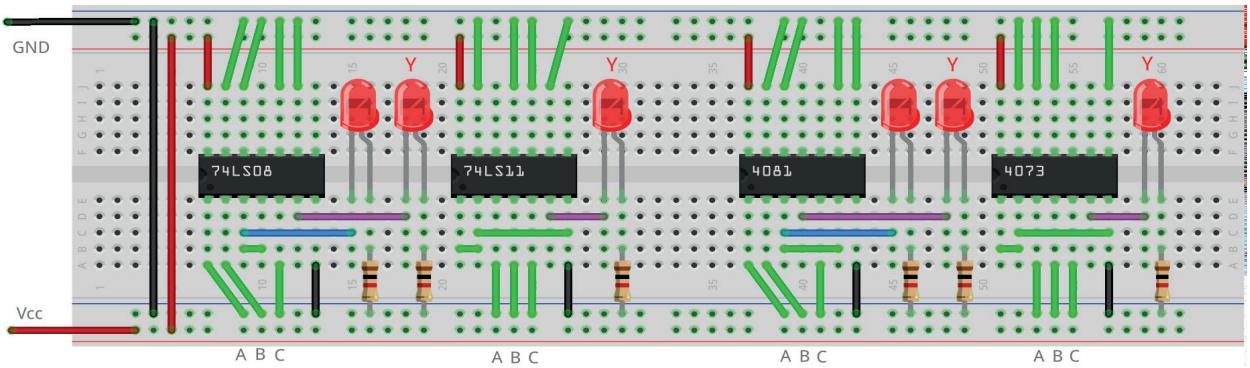


Figura 4.15 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.13, conforme mostrado na figura 4.15, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.14) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saída Y) e o fio azul (valor intermediário A + B).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.5 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.5 para todos as combinações descritas.
6. Simule o circuito da figura 4.13 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.5.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	C	A . B	Y = (A . B) . C)	Y = A . B . C
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 4.5

4.4.4. Experimento 4

Obter a tabela verdade para a equação: A . B . C = A . (B . C)

Esquemas:

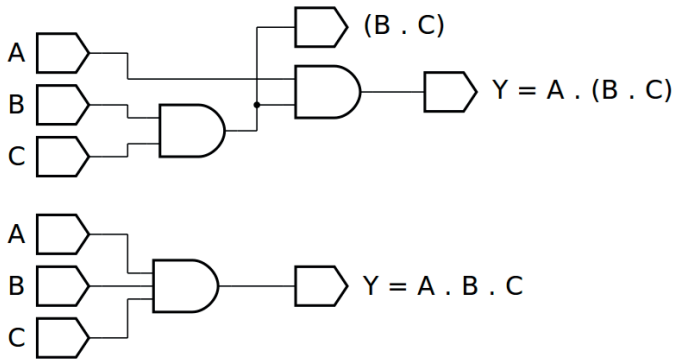


Figura 4.16 – Diagrama esquemático.

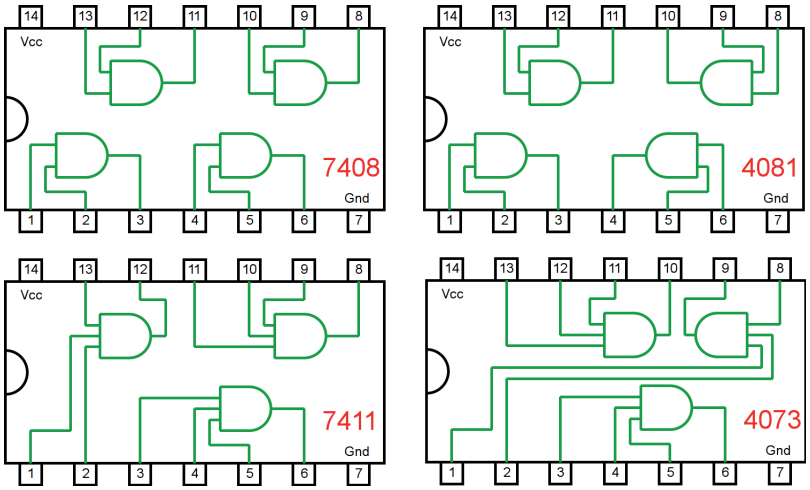


Figura 4.17 – Circuitos integrados TTL e CMOS.

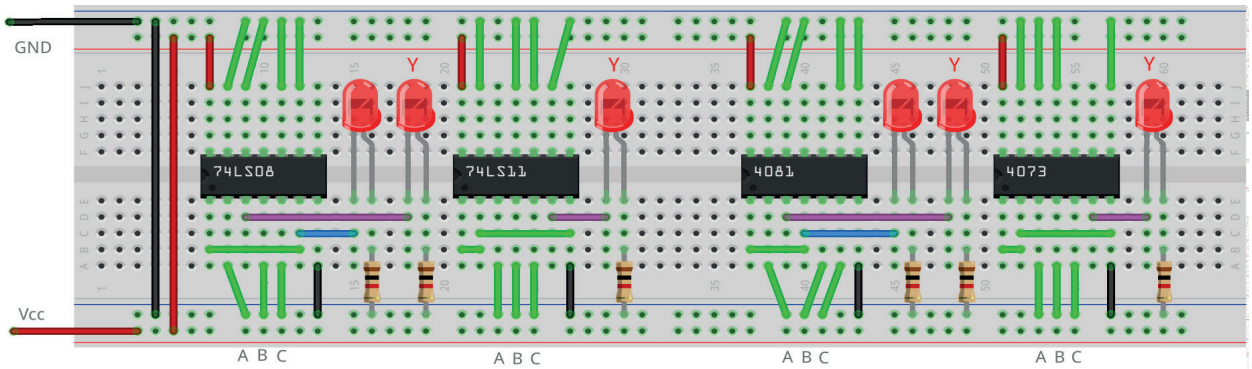


Figura 4.18 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.16, conforme mostrado na figura 4.18, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.17) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saída Y) e o fio azul (valor intermediário A + B).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.6 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.6 para todos as combinações descritas.
6. Simule o circuito da figura 4.16 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.6.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	C	B . C	Y = A . (B . C)	Y = A . B . C
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 4.6

4.5. Exame da Lei Distributiva

4.5.1. Experimento 1

Obter a tabela verdade para a equação: A . (B + C)

Esquemas:

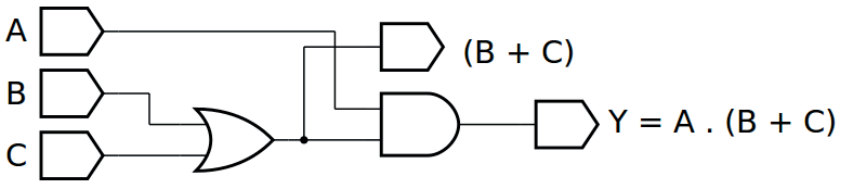


Figura 4.19 – Diagrama esquemático.

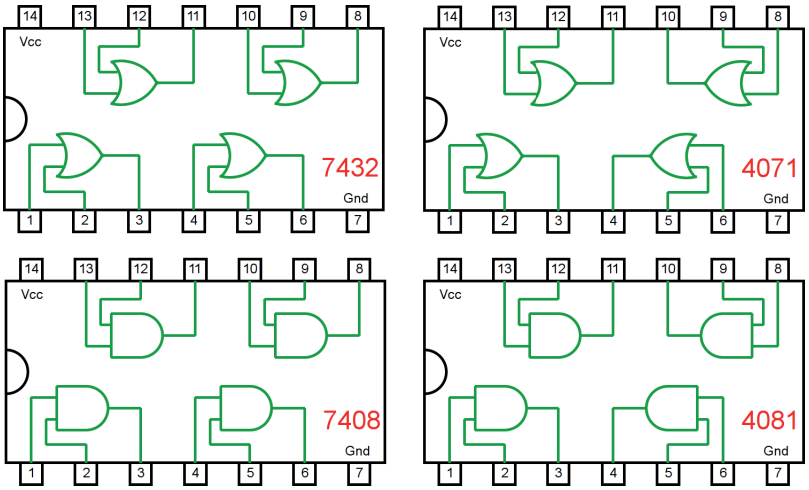


Figura 4.20 – Circuitos integrados TTL e CMOS.

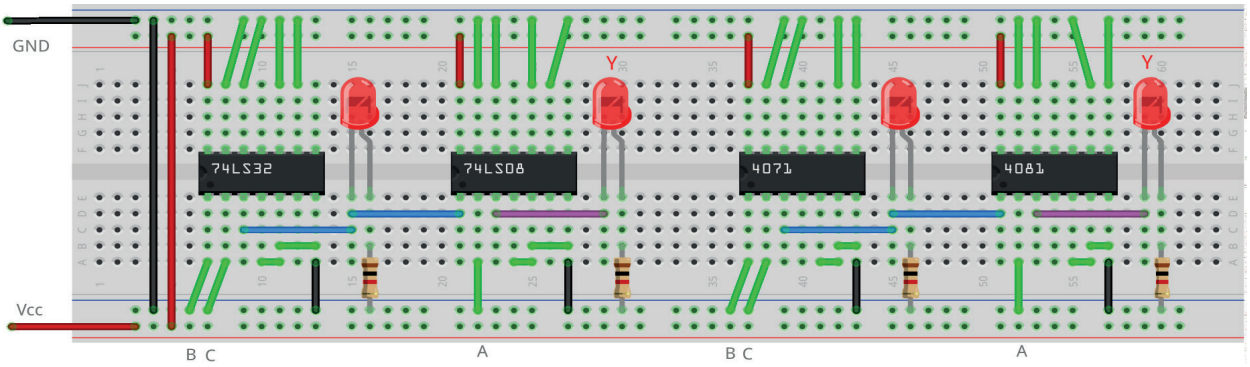


Figura 4.21 – Placa de montagem com componentes.

Procedimento

1. Monte o circuito da figura 4.19, conforme mostrado na figura 4.21, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.20) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saída Y) e o fio azul (valor intermediário B + C).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.7 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.7 para todos as combinações descritas.
6. Simule o circuito da figura 4.19 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.7.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	C	B + C	Y = A . (B + C)
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Tabela 4.7

4.5.2. Experimento 2

Obter a tabela verdade para a equação: (A . B) + (A . C)

Esquemas:

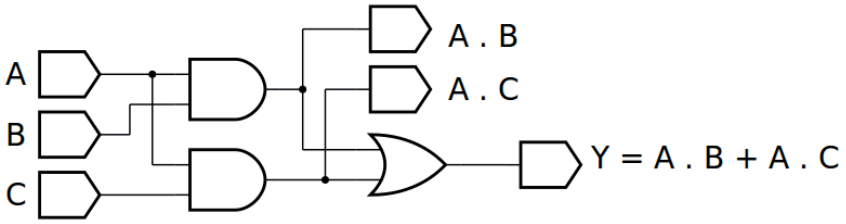


Figura 4.22 – Diagrama esquemático.

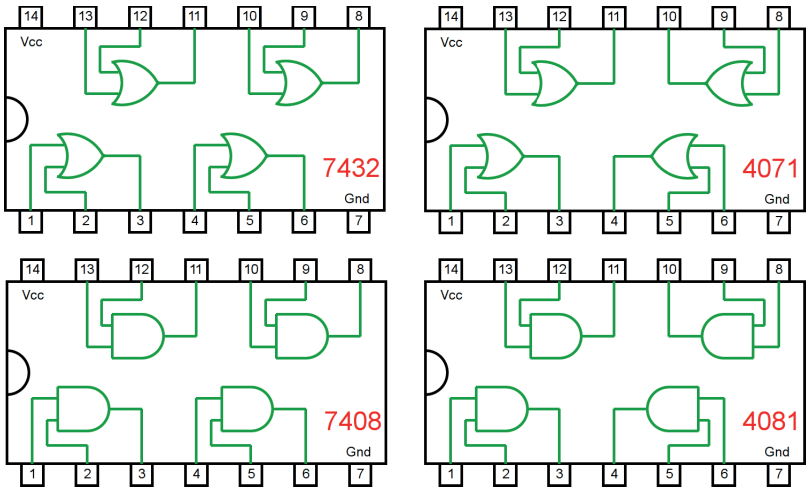


Figura 4.23 – Circuitos integrados TTL e CMOS.

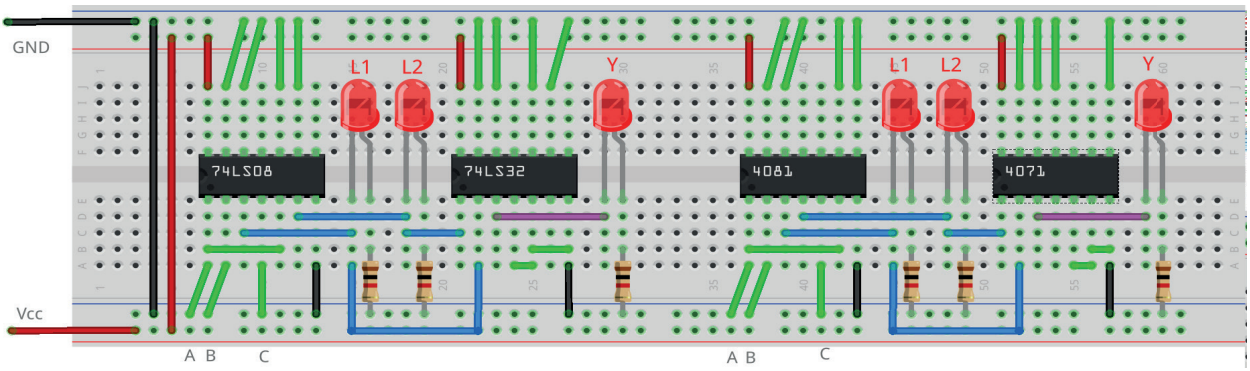


Figura 4.24 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.22, conforme mostrado na figura 4.24, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.23) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saída Y) e os fios azuis (valor intermediário L1 = A.B e L2 = B.C).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.8 considere:
   a) Para a lógica "0" da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica "1" da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a "1", caso contrário, se o LED estiver apagado então a saída é igual a "0".
5. Preencha a tabela 4.8 para todos as combinações descritas.
6. Simule o circuito da figura 4.22 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.8.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	C	A . B	A . C	Y = (A . B) + (B . C)
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 4.8

4.5.3. Experimento 3

Obter a tabela verdade para a equação: A + (B . C)

Esquemas:

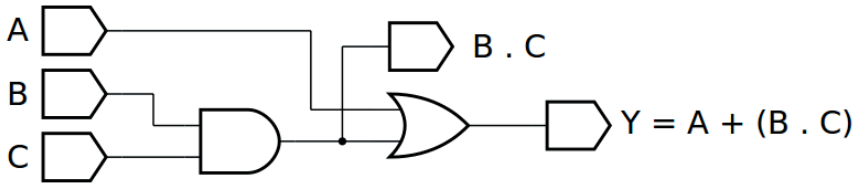


Figura 4.25 – Diagrama esquemático.

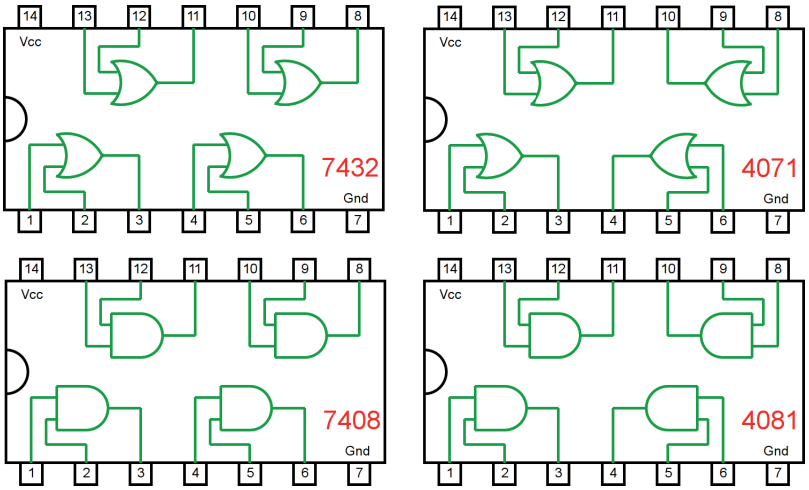


Figura 4.26 – Circuitos integrados TTL e CMOS.

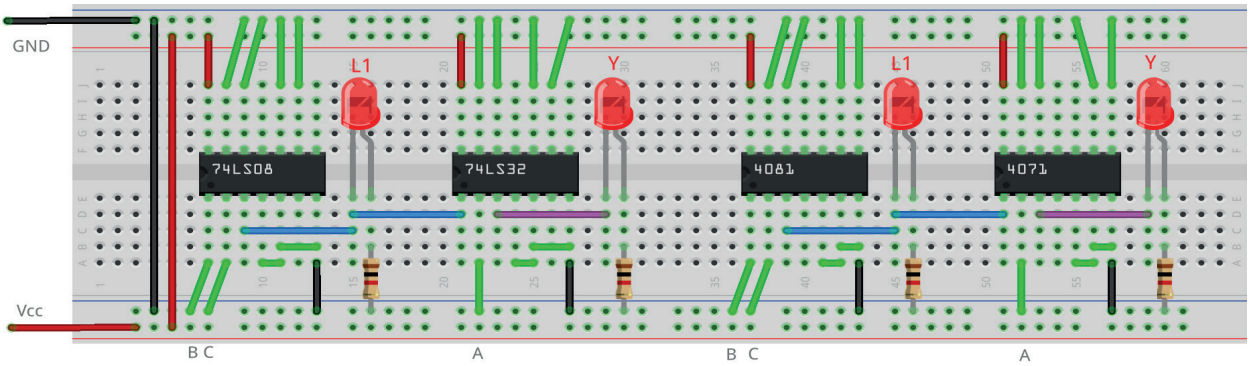


Figura 4.27 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.25 conforme mostrado na figura 4.27, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.26) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saída Y) e os fios azuis (valor intermediário L1 = B.C).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.8 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.9 para todos as combinações descritas.
6. Simule o circuito da figura 4.25 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.9.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	C	B . C	Y = A + (B . C)
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Tabela 4.9

4.5.4. Experimento 4

Obter a tabela verdade para a equação: (A + B) . (A + C)

Esquemas:

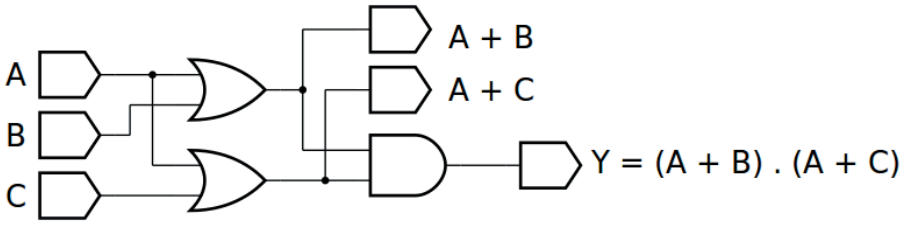


Figura 4.28 – Diagrama esquemático.

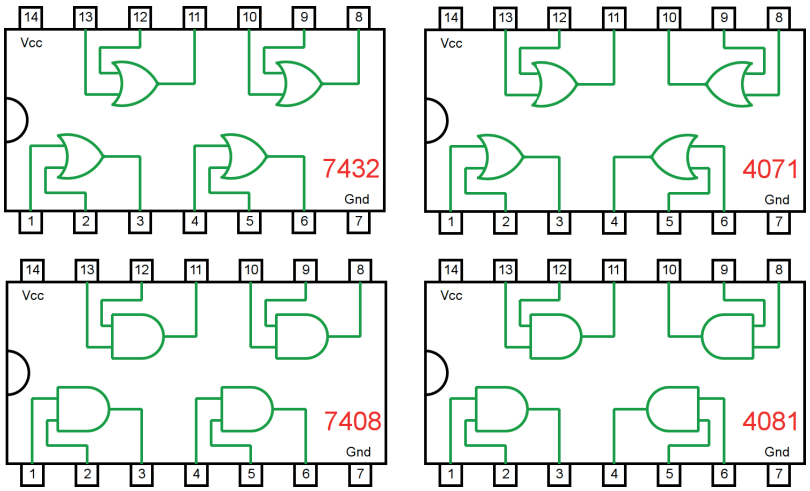


Figura 4.29 – Circuitos integrados TTL e CMOS.

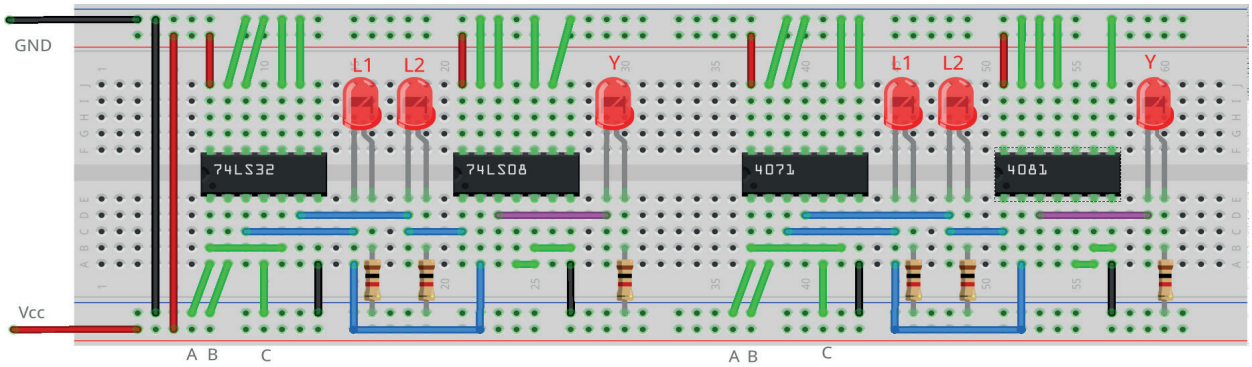


Figura 4.30 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.28, conforme mostrado na figura 4.30, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.29) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saída Y) e os fios azuis (valor intermediário L1 = A+B e L2 = B+C).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.10 considere:
   a) Para a lógica "0" da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica "1" da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a "1", caso contrário, se o LED estiver apagado então a saída é igual a "0".
5. Preencha a tabela 4.10 para todos as combinações descritas.
6. Simule o circuito da figura 4.28 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.10.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	C	A + B	A + C	Y = (A + B) . (B + C)
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Tabela 4.10

4.6. Exame da Lei Idempotente

4.6.1. Experimento 1

Obter a tabela verdade para as equações: A + A = A e A . A = A

Esquemas:

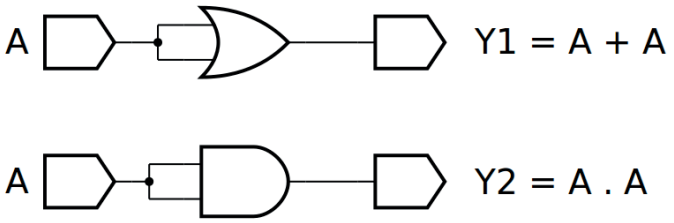


Figura 4.31 – Diagrama esquemático.

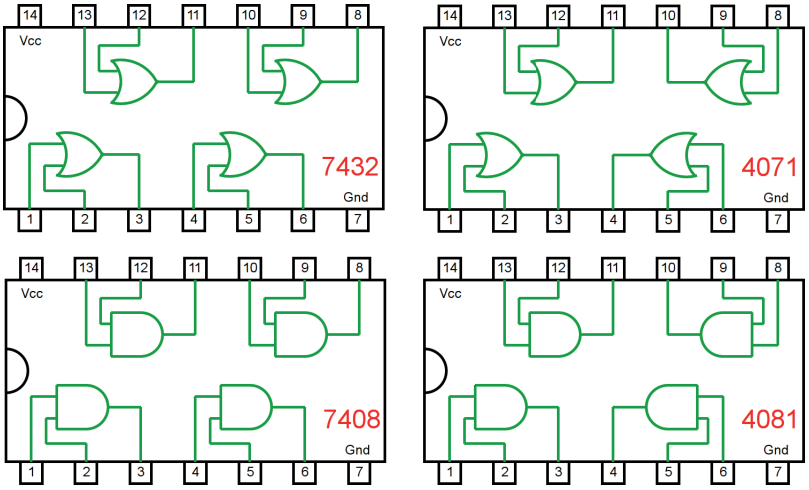


Figura 4.32 – Circuitos integrados TTL e CMOS.

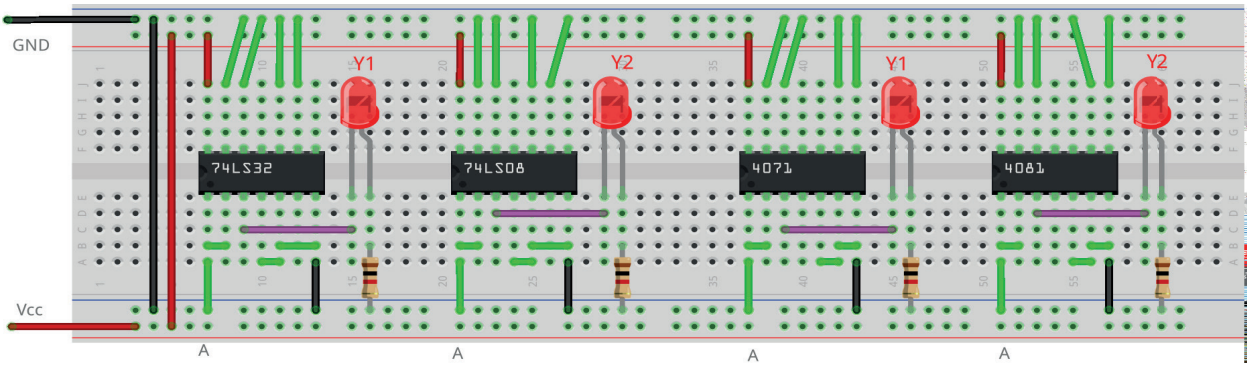


Figura 4.33 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.31, conforme mostrado na figura 4.33, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.32) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saídas Y1 e Y2).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.11 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.11 para todos as combinações descritas.
6. Simule o circuito da figura 4.31 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.11.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	A +A	A . A
0		
1		

Tabela 4.11

4.7. Exame da Lei do “E”

4.7.1. Experimento 1

Obter a tabela verdade para as equações: A . 1 = A e A . 0 = 0

Esquemas:

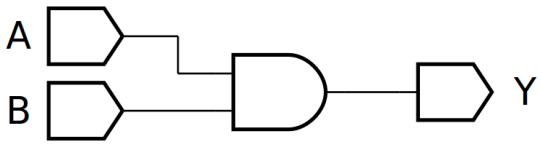


Figura 4.34 – Diagrama esquemático.

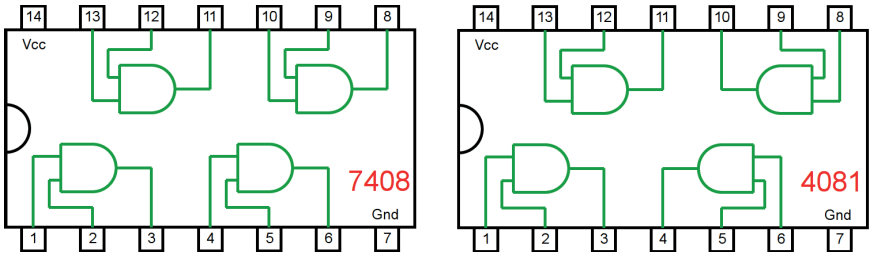


Figura 4.35 – Circuitos integrados TTL e CMOS.

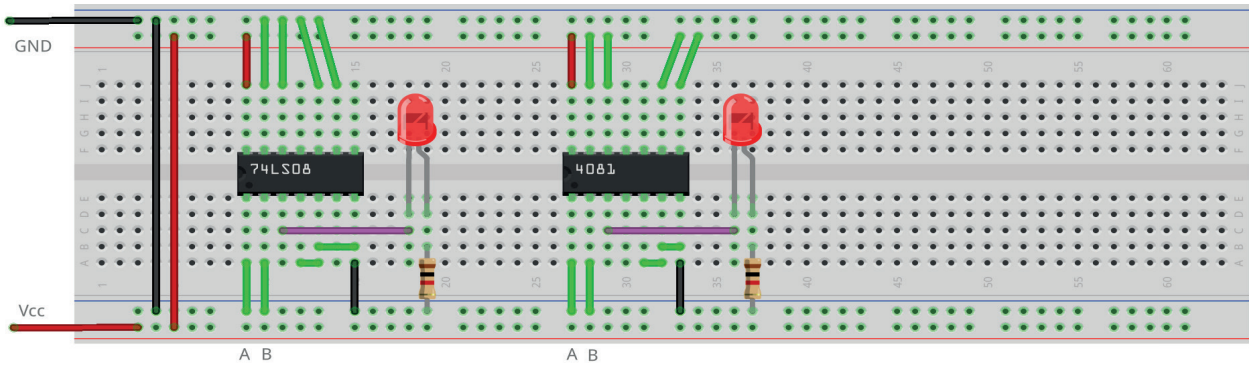


Figura 4.36 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.34, conforme mostrado na figura 4.36, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.35) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo (saída Y).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.12 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   d) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.12 para todos as combinações descritas.
6. Simule o circuito da figura 4.34 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.12.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	A . 0 (B = 0)	A . 1 (B = 1)
0		
1		

Tabela 4.12

4.8. Exame da Lei do “OU”

4.8.1. Experimento 1

Obter a tabela verdade para as equações: A + 0 = A e A + 1 = 1

Esquemas:

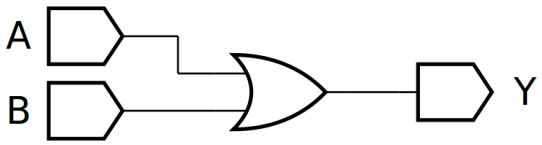


Figura 4.37 – Diagrama esquemático.

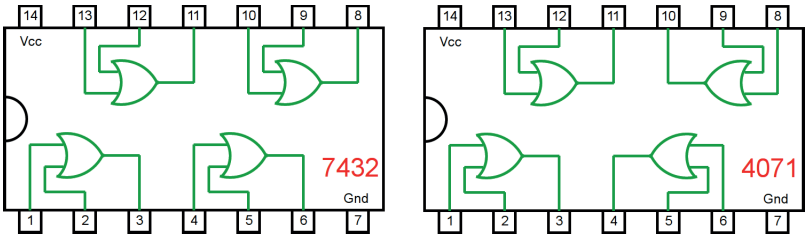


Figura 4.38 – Circuitos integrados TTL e CMOS.

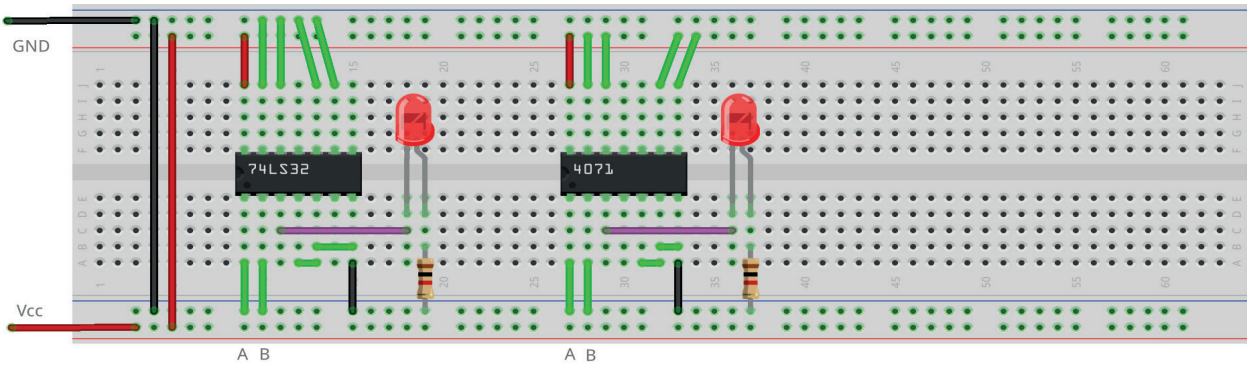


Figura 4.39 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.37, conforme mostrado na figura 4.39, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.38) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte o fio roxo (saída Y).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.13 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.13 para todos as combinações descritas.
6. Simule o circuito da figura 4.37 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.13.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	A + 0 (B = 0)	A + 1 (B = 1)
0		
1		

Tabela 4.13

4.9. Exame dos Complementos

4.9.1. Experimento 1

Obter a tabela verdade para as equações: A + A’ = 1 e A . A’ = 0

Esquemas:

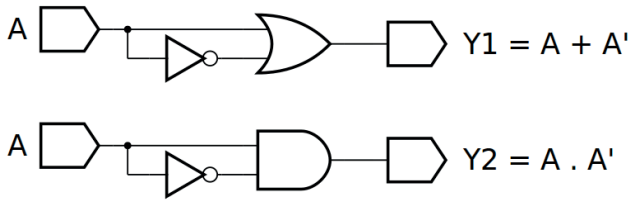


Figura 4.40 – Diagrama esquemático.

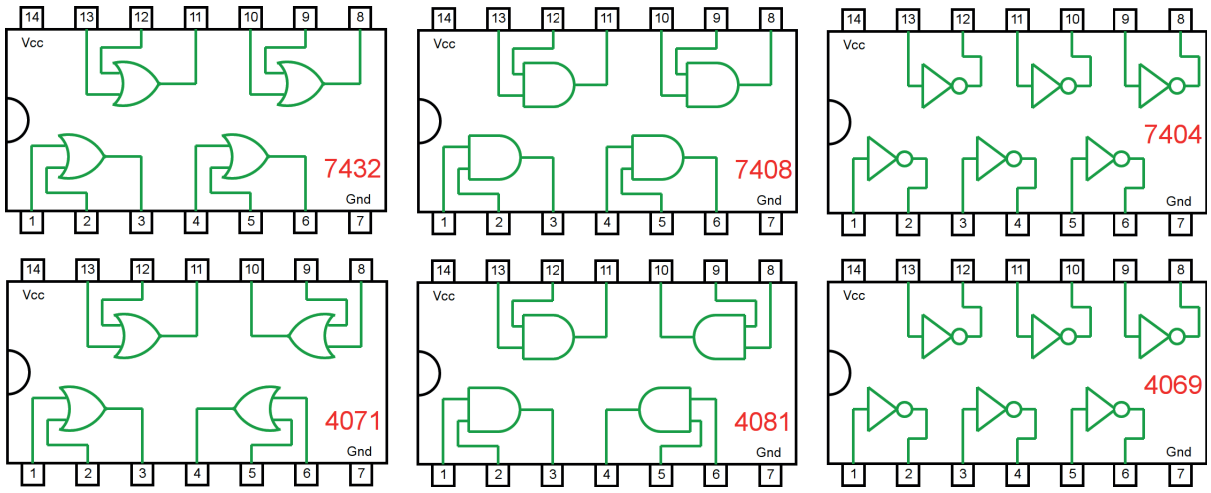


Figura 4.41 – Circuitos integrados TTL e CMOS.

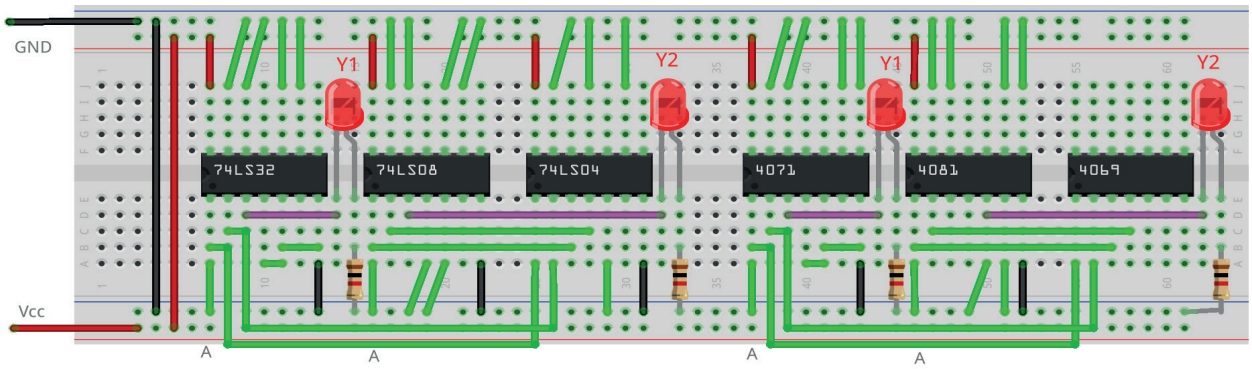


Figura 4.42 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.40, conforme mostrado na figura 4.42, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.41) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios roxos (saídas Y1 e Y2).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.14 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.14 para todos as combinações descritas.
6. Simule o circuito da figura 4.40 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.14.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	A + A’	A . A’
0		
1		

Tabela 4.14

4.10. Exame da Lei da Involução

4.10.1. Experimento 1

Obter a tabela verdade para as equações: (A')' = A e ((A + B)')' = A + B e ((A . B)')' = A . B

Esquemas:

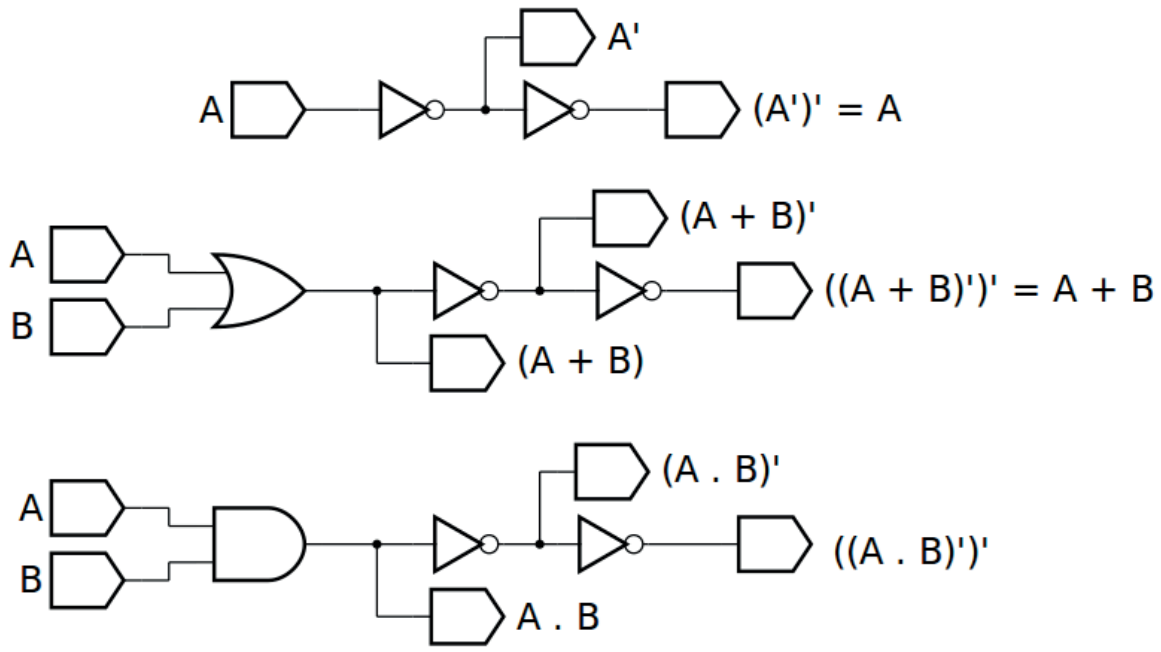


Figura 4.43 – Diagrama esquemático.

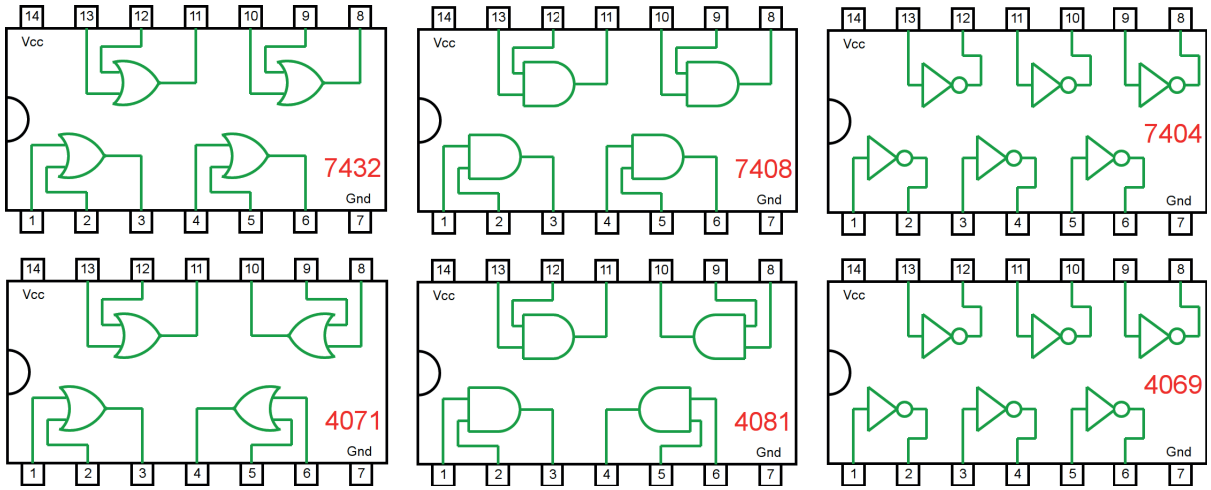


Figura 4.44 – Circuitos integrados TTL e CMOS.

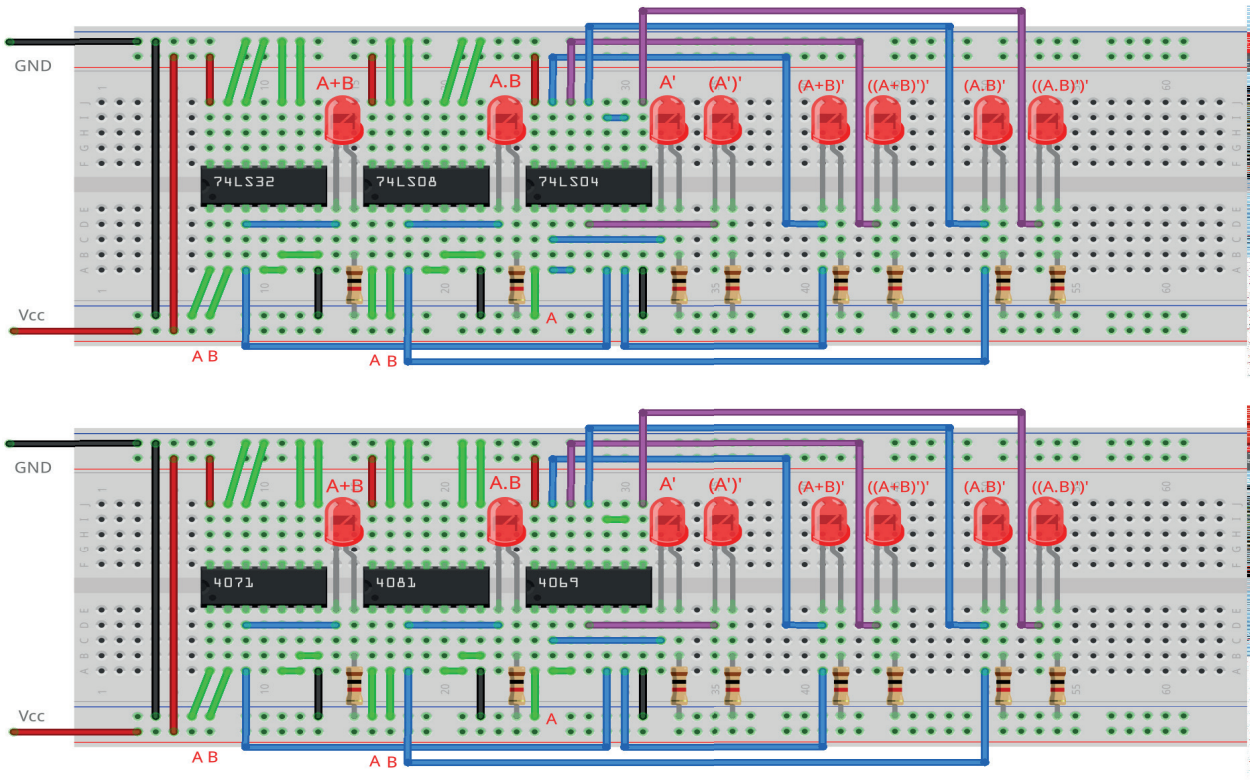


Figura 4.45 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.43, conforme mostrado na figura 4.45, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.44) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios azuis (valores intermediários) e os fios roxos (saídas (A')' , ((A+B)')' e ((A.B)')').
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.15 considere:
   a) Para a lógica "0" da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica "1" da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a "1", caso contrário, se o LED estiver apagado então a saída é igual a "0".

5. Preencha a tabela 4.15 para todos as combinações descritas.
6. Simule o circuito da figura 4.43 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.15.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	A'	(A+B)	(A+B)'	((A+B)')'	(A.B)	(A.B)'	((A.B)')'
0	0							
0	1							
1	0							
1	1							

Tabela 4.15

4.11. Exame da Lei da Absorção

4.11.1. Experimento 1

Obter a tabela verdade para as equações: A + (A . B) = A e A . (A + B) = A

Esquemas:

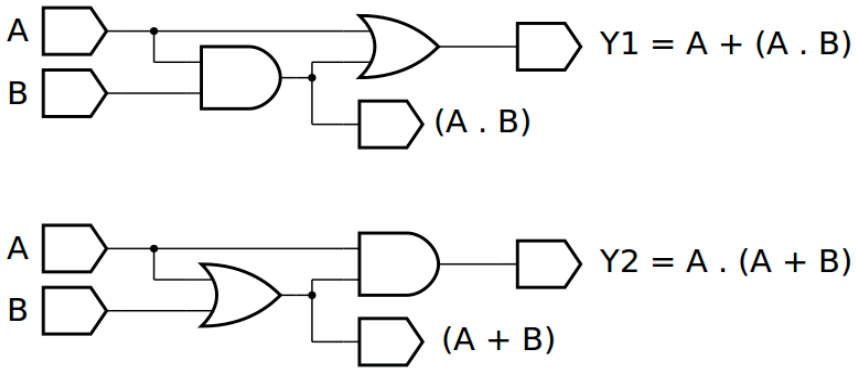


Figura 4.46 – Diagrama esquemático.

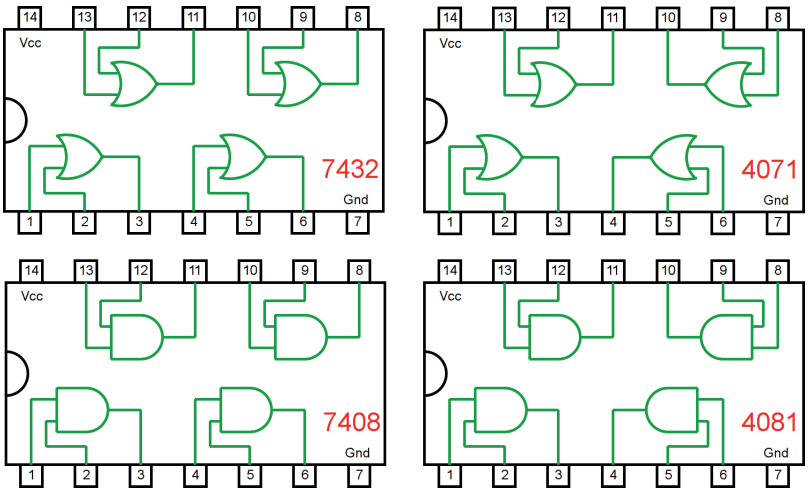


Figura 4.47 – Circuitos integrados TTL e CMOS.

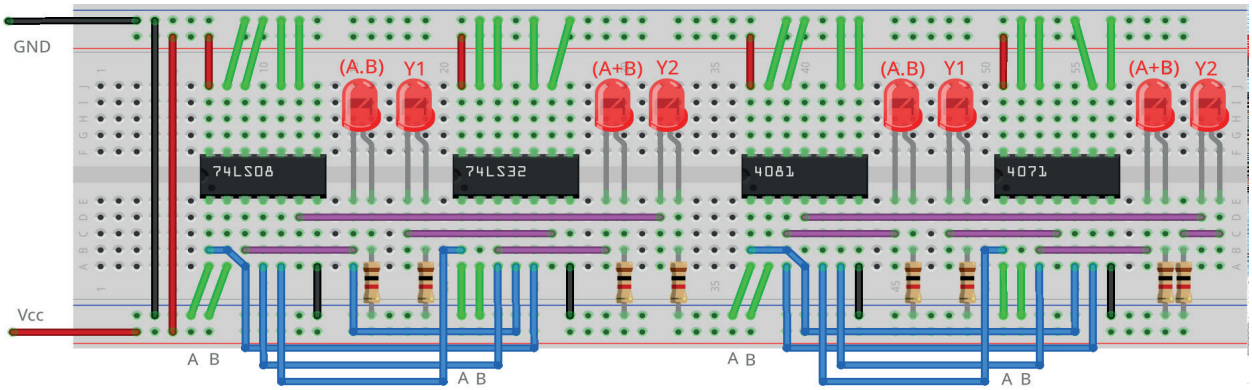


Figura 4.48 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.46, conforme mostrado na figura 4.48, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.47) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios azuis e os fios roxos (saídas).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.16 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.16 para todos as combinações descritas.
6. Simule o circuito da figura 4.46 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.16.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	(A.B)	(A+B)	A+(A.B)	A.(A+B)
0	0				
0	1				
1	0				
1	1				

Tabela 4.16

4.12. Exame da Lei de De Morgan

4.12.1. Experimento 1

Obter a tabela verdade para a equação: (A . B)’ = A’ + B’

Esquemas:

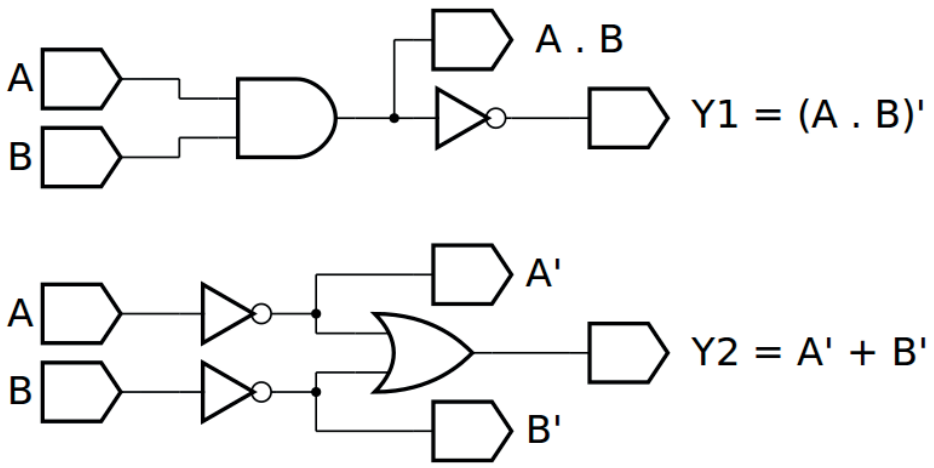


Figura 4.49 – Diagrama esquemático.

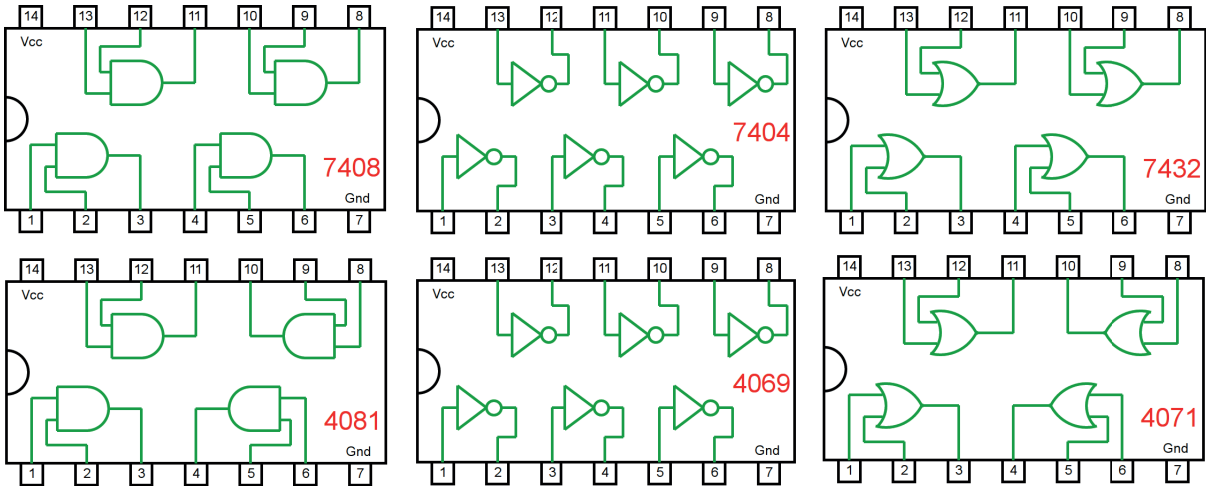


Figura 4.50 – Circuitos integrados TTL e CMOS.

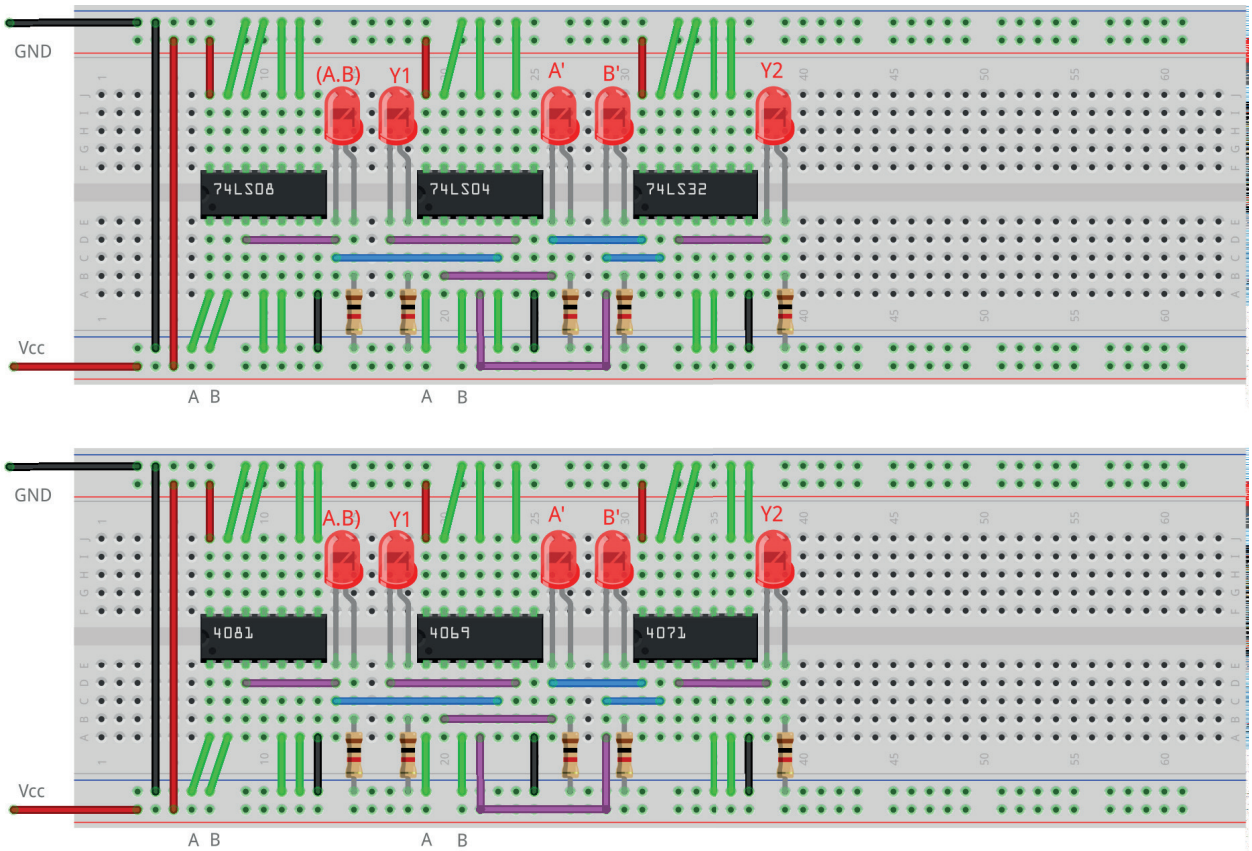


Figura 4.51 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.49, conforme mostrado na figura 4.51, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.50) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios azuis e os fios roxos (saídas).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.17 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).
   c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.
5. Preencha a tabela 4.17 para todos as combinações descritas.
6. Simule o circuito da figura 4.49 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.17.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	A’	B’	A . B	Y1 = (A . B)’	Y2 = A’ + B’
0	0					
0	1					
1	0					
1	1					

Tabela 4.17

4.12.2. Experimento 2

Obter a tabela verdade para a equação: (A + B)’ = A’ . B’

Esquemas:

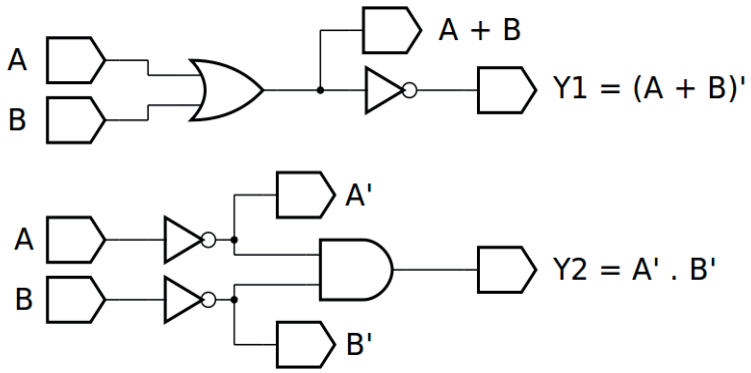


Figura 4.52 – Diagrama esquemático.

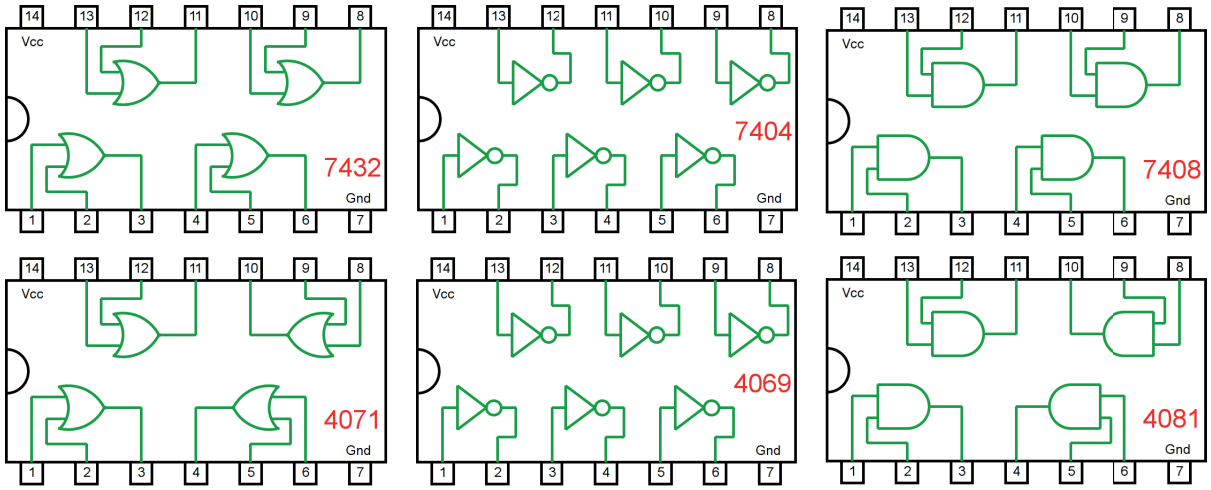


Figura 4.53 – Circuitos integrados TTL e CMOS.

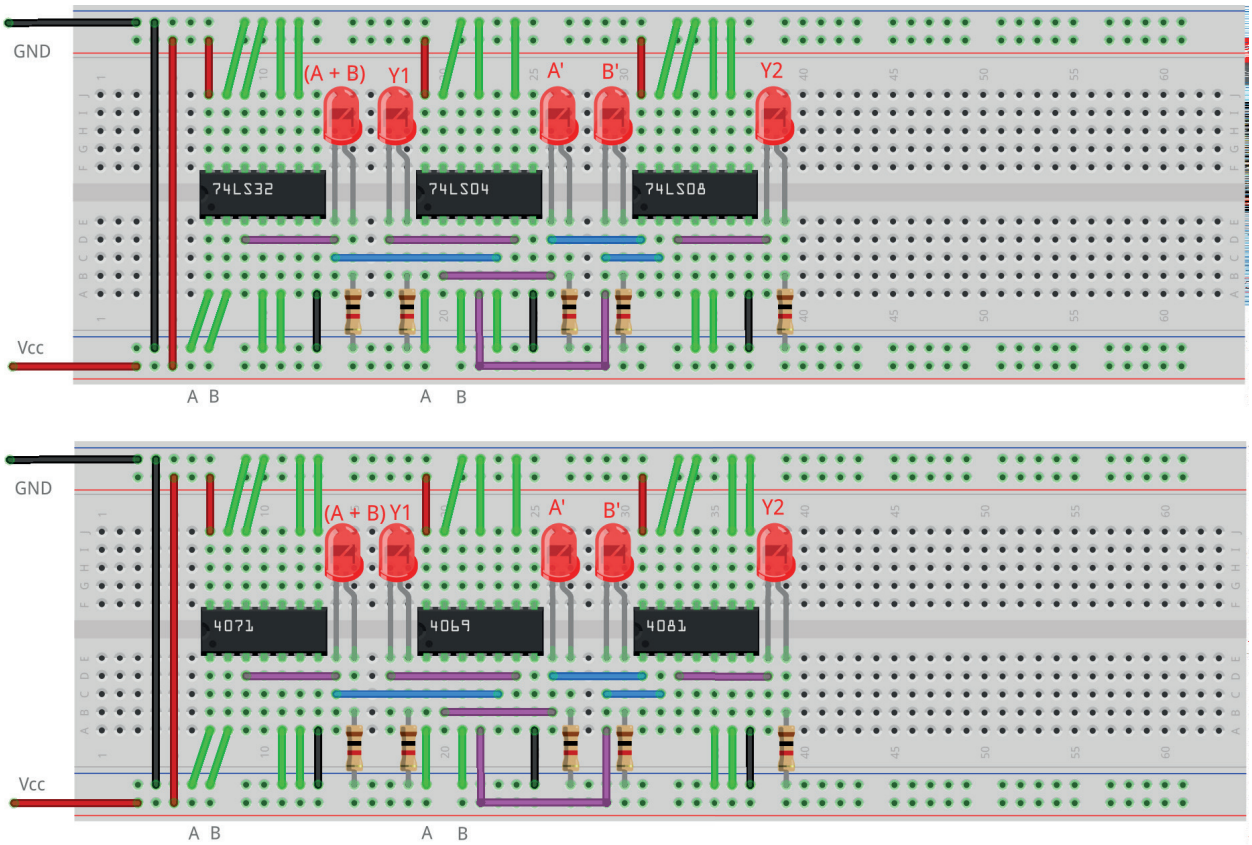


Figura 4.54 – Placa de montagem com componentes.

Procedimento:

1. Monte o circuito da figura 4.52, conforme mostrado na figura 4.54, na placa de montagem.
   a) Coloque os circuitos integrados (TTL e/ou CMOS da figura 4.53) nas posições indicadas.
   b) Coloque os LEDs e os resistores de 1KΩ nas posições indicadas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte os fios verdes. Os fios verdes estão conectados ao terra e portanto fornecendo valor lógico (0) às entradas de todas as portas do CI.
   e) Conecte os fios azuis e os fios roxos (saídas).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc). Não conecte o pino banana vermelho na fonte de energia.
2. Ligue a fonte de energia.
3. Conecte o cabo vermelho ao borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 4.18 considere:
   a) Para a lógica “0” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de terra (Gnd, preto).
   b) Para a lógica “1” da variável A, B ou C, o fio verde deve estar conectado ao terminal do CI correspondente e conectado ao barramento de energia (Vcc, vermelho).

c) Se o LED estiver aceso então a saída é igual a “1”, caso contrário, se o LED estiver apagado então a saída é igual a “0”.

5. Preencha a tabela 4.18 para todos as combinações descritas.
6. Simule o circuito da figura 4.52 em um software de simulação para computador (SmartSim, Atanua ou Qucs) e compare com a tabela 4.18.

Observação importante: Qualquer alteração de fios na placa de montagem somente deve ser feita com o cabo vermelho DESCONECTADO da fonte de energia.

Tabelas de dados:

A	B	A’	B’	A + B	Y1 = (A + B)’	Y2 = A’ . B’
0	0					
0	1					
1	0					
1	1					

Tabela 4.18

4.13. Simulação das identidades booleanas em Arduíno

Utilizando o programa do Capítulo 3, simule algumas identidades booleanas e compare com a tabela apropriada. Substitua a lógica destacada em negrito pela lógica da identidade booleana a simular.

Lei Distributiva:

Parte 1: A. (B + C)	Lógica: A&&(B||C)	Tabela: 4.7
Parte 2: (A.B) + (A.C)	Lógica: (A&&B)||(A&&C)	Tabela: 4.8
Parte 3: A + (B.C)	Lógica: A||(B&&C)	Tabela: 4.9
Parte 4: (A + B). (A + C)	Lógica: (A||B)&&(A||C)	Tabela: 4.10

Lei de Absorção:

Parte 1: A + A.B = A	Lógica: A||A&&B	Tabela: 4.16
Parte 2: A. (A + B) = A	Lógica: A&&(A||B)	Tabela:4.16

Teoremas de De Morgan:

Parte 1: (A.B)’	Lógica: !(A&&B)	Tabela: 4.17
Parte 2: A’ + B’	Lógica: !A||!B	Tabela: 4.17
Parte 3: (A + B)	Lógica: A||B	Tabela: 4.18
Parte 4: A’.B’	Lógica: !A&&!B	Tabela: 4.18

Capítulo 5. Implementação de funções booleanas com portas lógicas

5.1. Objetivos

1. Simplificação de funções booleanas por meio de álgebra booleana e mapas de Karnough
2. Implementação de funções booleanas simplificadas com diferentes combinações de portas lógicas

5.2. Informação preliminar

Nestes experimentos, a realização (implementação) de funções booleanas (expressões) usando circuitos integrados (CIs) de portas lógicas são investigadas. Antes da implementação de funções booleanas, é necessário simplificar essas funções, porque funções simples significam o seguinte: a implementação é mais econômica, o espaço de implementação será menor e você economizará energia. Portanto, é desejável simplificar as funções booleanas. Em segundo lugar, a expressão lógica simplificada é implementada usando CIs de portas lógicas. Finalmente, todas as combinações possíveis das entradas são aplicadas como lógica 0 e lógica 1 e os valores de saída são obtidos experimentalmente.

5.2.1. Simplificação de funções booleanas

Existem alguns métodos bem conhecidos para simplificar as funções booleanas, usando álgebra booleana, mapas de Karnough (K-map) e método de Quine McCluskey. É muito difícil simplificar as funções booleanas usando a álgebra booleana. Os mapas de Karnough resolvem este problema de maneira fácil, mas são práticos para simplificar funções booleanas de até 4 variáveis. Se houver mais de 4 variáveis nesse caso, o método Quine McCluskey é preferido para simplificar as funções booleanas.

5.2.2. Mapas de Karnough de 3 Variáveis

Como pode ser visto nas Tabelas 5.1a e 5.1b, em um mapa de Karnough com três variáveis, há oito mintermos (ou maxtermos), cada um dos quais é representado por uma célula dentro do mapa. A ordem dos números 00, 01, 11 e 10 na linha (esquerda) e coluna (direita) é do tipo de código Gray, e não da forma binária. A simplificação de uma função em um K-map de 3 variáveis é mostrada na Tabela 5.2.

	a	
bc	**0**	**1**
00	m0	m4
01	m1	m5
11	m3	m7
10	m2	m6

(modo vertical)

	bc			
a	**00**	**01**	**11**	**10**
0	m0	m1	m3	m2
1	m4	m5	m7	m6

(modo horizontal)

Tabela 5.1a – Mapas de Karnough de 3 variáveis (versão 1).

	a'	a	
b'	m0	m4	**c'**
	m1	m5	**c**
b	m3	m7	
	m2	m6	**c'**

(modo vertical)

	b'		b	
a'	m0	m1	m3	m2
a	m4	m5	m7	m6
	c'	**c**		**c'**

(modo horizontal)

Tabela 5.1b – Mapas de Karnough de 3 variáveis (versão 2).

Este é um exemplo de como simplificar uma função de 3 variáveis:

f(a,b,c) = Σm(0,2,3,4,6,7).

A região em vermelho agrega os mintermos m3, m2, m7 e m6. Isso anula os bits (a) (a + a' = 1) e (c) (c + c' = 1). Sobra o bit (b).

A região em verde agrega os mintermos m0, m1, m4 e m5. Isso anula os bits (a) (a + a' = 1) e (b) (b + b' = 1). Sobra o bit (c').

A equação final é: **f(a,b,c) = b + c'**

	a	
bc	**0**	**1**
00	1	1
01		
11	1	1
10	1	1

Tabela 5.2 – Simplificação de uma função em um K-map de 3 variáveis.

5.2.2. Mapas de Karnough de 4 Variáveis

Como pode ser visto na Tabelas 5.3a e 5.3b, em um mapa Karnough de 4 variáveis, há dezesseis mintermos (ou maxtermos), cada um dos quais é representado por uma célula dentro do mapa. A ordem dos números 00, 01, 11 e 10 nas linhas e colunas é do tipo de código Gray em vez de binário. A simplificação de duas funções em K-map de 4 variáveis é mostrada na Tabela 5.4. À esquerda (respectivamente à direita), uma função booleana de 4 variáveis representada por uma soma de mintermos (respectivamente produto de maxterms) é simplificada por um K-map de 4 variáveis.

	ab			
cd	**00**	**01**	**11**	**10**
00	m0	m4	m12	m8
01	m1	m5	m13	m9
11	m3	m7	m15	m11
10	m2	m6	m14	m10

(modo vertical)

	cd			
ab	**00**	**01**	**11**	**10**
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10

(modo horizontal)

Tabela 5.3a – Mapas de Karnough de 4 variáveis (versão 1).

	a'		**a**		
c'	m0	m4	m12	m8	**d'**
	m1	m5	m13	m9	**d**
c	m3	m7	m15	m11	
	m2	m6	m14	m10	**d'**
	b'	**b**		**b'**	

(modo vertical)

	c'		**c**		
a'	m0	m1	m3	m2	**b'**
	m4	m5	m7	m6	**b**
a	m12	m13	m15	m14	
	m8	m9	m11	m10	**b'**
	d'	**d**		**d'**	

(modo horizontal)

Tabela 5.3b – Mapas de Karnough de 4 variáveis (versão 2).

Este é um exemplo de como simplificar uma função de 4 variáveis:

Usando mintermos: $f1(a,b,c,d) = \Sigma_m(0,1,2,4,5,6,8,10,12,13,14,15)$

Usando maxtermos: $f2(a,b,c,d) = \Pi_M(0,1,2,4,5,6,8,10,12,13,14,15)$

	ab			
cd	00	01	11	10
00	1	1	1	1
01	1	1	1	
11			1	
10	1	1	1	1

f1(a,b,c,d) = d'+a'c'+ab

	ab			
cd	00	01	11	10
00	0	0	0	0
01	0	0	0	
11			0	
10	0	0	0	0

f2(a,b,c,d) = d.(a+c).(a'+b')

Tabela 5.4 – Simplificação de duas funções em K-map de 4 variáveis.

5.2.3. Condições de "não importa"

As condições de "não importa" estão relacionadas a funções especificadas de forma incompleta. Neste caso, a função não é especificada para certas combinações das variáveis. Essas combinações podem ser designadas como 0 ou 1 e são representadas por "x" em um K-map. Segue-se uma função de exemplo, em que os termos 0, 2 e 5 (mintermos) são as combinações de variáveis que tornam a função igual a 1. Os termos de referência das condições 1, 3 e 7 podem ser atribuídos 0 ou 1.

$$f(a,b,c) = \Sigma_m(0,2,5) + \Sigma_d(1,3,7)$$

Podemos simplificar a função dada como mostrado na Tabela 5.5. Neste exemplo, as condições de não importa, representadas por "x", são todas atribuídas 1.

	bc			
a	00	01	11	10
0	1	x	x	1
1		1	x	

f(a,b,c) = a' + c

Tabela 5.5 – A simplificação da função
f(a,b,c) = $\Sigma_m(0,2,5) + \Sigma_d(1,3,7)$ em um K-map de 3 variáveis.

A função f(a,b,c) = $\Sigma_m(0,2,5) + \Sigma_d(1,3,7)$ pode ser representado na forma de mantermos:

$$f(a,b,c) = \Pi_M(4,6) \,.\, \Pi_D(1,3,7)$$

Podemos simplificar a função dada como mostrado na Tabela 5.6. Neste exemplo, as condições de "não importa", representadas por "x", são todas atribuídas 1.

	bc			
a	**00**	**01**	**11**	**10**
0		x	x	
1	0		x	0

f(a,b,c) = a' + c

Tabela 5.6 – A simplificação da função
f(a,b,c) = $\Pi_M(4,6) \cdot \Pi_D(1,3,7)$ em um K-map de 3 variáveis.

5.2.4. Soma de Mintermos

Para as "n" variáveis booleanas, pode haver 2^n mintermos diferentes. Cada função booleana pode ser representada como soma de mintermos. Os números binários de 0 a 2^{n-1} estão listados sob as "n" variáveis. Cada mintermo é obtido a partir de um termo E de "n" variáveis, com cada variável sendo iniciada se o bit correspondente do número binário for um 0 e não aplicado se for 1. Às vezes, é mais conveniente expressar uma função booleana como soma de mintermos. Se uma função não estiver na soma padrão da forma de mintermos, ela poderá ser convertida na forma padrão. O primeiro passo é representar a função como soma de produtos. Em segundo lugar, neste caso, cada forma do produto (se não for um produto padrão) é convertido em produto padrão. Por exemplo, se um termo do produto estiver sem a variável 'a' ou seu complemento, esse termo será convertido em E com o termo (a + a'). Dezesseis mintermos para quatro variáveis, juntamente com sua designação simbólica, estão listados na tabela 5.7.

Variáveis				Mintermos		Maxtermos	
a	b	c	d	termo	nome	termo	nome
0	0	0	0	a'b'c'd'	m0	a+b+c+d	M0
0	0	0	1	a'b'c'd	m1	a+b+c+d'	M1
0	0	1	0	a'b'cd'	m2	a+b+c'+d	M2
0	0	1	1	a'b'cd	m3	a+b+c'+d'	M3
0	1	0	0	a'bc'd'	m4	a+b'+c+d	M4
0	1	0	1	a'bc'd	m5	a+b'+c+d'	M5
0	1	1	0	a'bcd'	m6	a+b'+c'+d	M6
0	1	1	1	a'bcd	m7	a+b'+c'+d'	M7
1	0	0	0	ab'c'd'	m8	a'+b+c+d	M8
1	0	0	1	ab'c'd	m9	a'+b+c+d'	M9
1	0	1	0	ab'cd'	m10	a'+b+c'+d	M10
1	0	1	1	ab'cd	m11	a'+b+c'+d'	M11
1	1	0	0	abc'd'	m12	a'+b'+c+d	M12
1	1	0	1	abc'd	m13	a'+b'+c+d'	M13
1	1	1	0	abcd'	m14	a'+b'+c'+d	M14
1	1	1	1	abcd	m15	a'+b'+c'+d'	M15

Tabela 5.7 – Mintermos e maxtermos para quatro variáveis binárias.

5.2.5. Produto de Maxtermos

Para as "n" variáveis booleanas, pode haver 2^n maxterms diferentes. Cada função booleana pode ser representada como produto de maxtermos. Os números binários de 0 a 2^{n-1} estão listados sob as "n" variáveis. Cada maxtermo é obtido de um termo OU de "n" variáveis, com cada variável sendo não iniciada se o bit correspondente do número binário é um 0 e iniciado se for 1. Às vezes é mais conveniente expressar uma função booleana como produto de maxtermos. Se uma função não estiver na forma padrão de produto de maxterms, ela poderá ser convertida na forma padrão. O primeiro passo é representar a função como produto de somas. Em segundo lugar, neste caso, cada forma de soma (se não for uma soma padrão) é convertida em uma soma padrão. Por exemplo, se um termo de soma tiver a variável "a" ou seu complemento, então este termo será convertido em OU com o termo (a.a'). Dezesseis maxtermos para quatro variáveis, juntamente com sua designação simbólica, são listados na Tabela 5.7. Uma função dada na forma de uma soma de mintermos pode ser convertida na forma de produto de maxtermos e vice-versa. Por exemplo, a função $f(a,b,c,d) = \Sigma_m(0,2,10,11,12,14)$ é dada na forma de soma de mintermos. Esta função é convertida na forma de produto de maxtermos da seguinte forma: $f(a,b,c,d) = \Pi_M(1,3,4,5,6,7,8,9,13,15)$. Como pode ser visto, o número total de termos distintos (mintermos e maxtermos) para quatro variáveis é 16 para uma dada função quando se considera a forma de soma de mintermos e a forma de produto de maxtermos.

5.2.6. Formas de implementação possíveis de uma função booleana

É desejável simplificar uma função booleana antes de sua implementação com portas lógicas. Uma função booleana pode ser implementada em dez formas diferentes, como pode ser visto nesta seção. Primeiramente, oito dessas formas são consideradas. As duas primeiras formas são a soma de produtos (forma E-OU) e o produto de somas (forma OU-E). Usando a lei da involução ((x')' = x) e os teoremas de De Morgan ((x + y)' = x' . y' ; (x . y)' = x' + y') da álgebra booleana, a forma E-OU pode ser convertida em formas NE-NE, OU-NE e NOU-OU como pode ser visto na Figura 5.1. Da mesma forma, a forma OU-E pode ser convertido nas formas NOU-NOU, E-NOU e NE-E. Também é possível converter a forma E-OU em forma OU-E e vice-versa. Finalmente, como visto na Figura 5.1, é possível implementar uma função booleana nessas oito formas diferentes. Agora vamos considerar essas formas com mais detalhes.

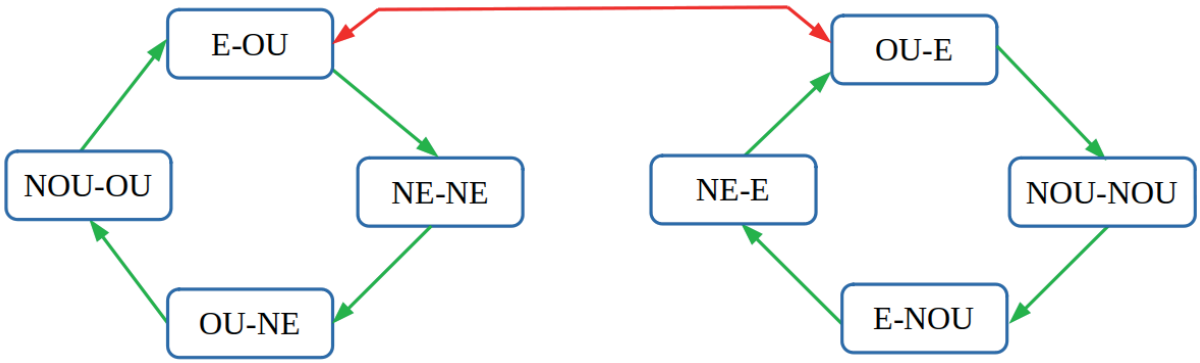


Figura 5.1 – 8 formas possíveis de implementação de uma função booleana.

Exemplo: Para uma função booleana dada como a soma dos mintermos

$$f(a,b,c) = (2,3,4,6) = m2 + m3 + m4 + m6 = a'bc' + a'bc + ab'c' + abc':$$

a) Simplifique essa função usando a álgebra booleana.
b) Implemente a função soma de produtos simplificada na forma E-OU.
c) Implemente a função soma de produtos simplificada na forma NE-NE.
d) Implemente a função soma de produtos simplificada na forma OU-NE.
e) Implemente a função soma de produtos simplificada na forma NOU-OU.
f) Expresse a função dada como produto de maxtermos.
g) Simplifique a função "f" expressa como produto de maxtermos na etapa anterior usando a álgebra booleana.
h) Implemente a função produto de somas simplificada na forma OU-E.
i) Implemente a função produto de somas simplificada na forma NOU-NOU.
j) Implemente a função produto de somas simplificada na forma E-NOU.
k) Implemente a função produto de somas simplificada na forma NE-E.

Respostas:

a) Primeiramente vamos simplificar a função booleana dada como a soma dos mintermos usando álgebra booleana:

Forma original.

$$f(a,b,c) = a'bc' + a'bc + ab'c' + abc'$$

Aplicando a lei distributiva. Colocando em evidência (a'b) para o primeiro e segundo termos e (ac') para o terceiro e quarto termos.

$$f(a,b,c) = a'b(c'+c) + ac'(b'+b)$$

Aplicando a lei dos complementos. (c'+c) = 1 e (b'+b) = 1.

$$f(a,b,c) = a'b + ac'$$

Esta é a forma simplificada da função no formato E-OU.

b) A forma simplificada da função como soma dos produtos é a seguinte: f(a,b,c) = a'b + ac'. Nesta forma, esta função pode ser diretamente implementada por dois níveis de portas lógicas E-OU. A implementação desta função simplificada na forma E-OU é mostrada na Figura 5.2. A tabela 5.8 mostra a tabela verdade da função f(a,b,c) = (2,3,4,6).

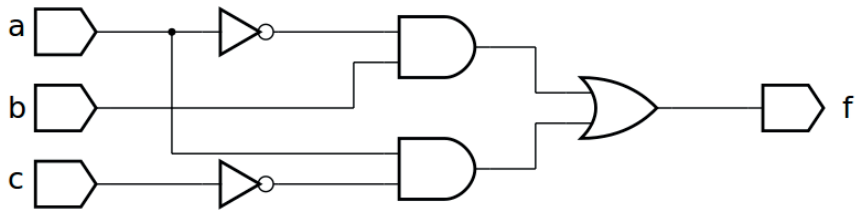


Figura 5.2 – A implementação da função simplificada f(a,b,c) = (2,3,4,6) = a'b + ac' na forma E-OU.

mintermo	a	b	c	f
m0	0	0	0	**0**
m1	0	0	1	**0**
m2	0	1	0	**1**
m3	0	1	1	**1**
m4	1	0	0	**1**
m5	1	0	1	**0**
m6	1	1	0	**1**
m7	1	1	1	**0**

Tabela 5.8 – A tabela verdade da função f(a,b,c) = (2,3,4,6).

c) Para obter a forma de dois níveis de portas lógicas NE-NE para a função f(a,b,c) = a’b + ac’, primeiro usamos a lei de involução ((x’)’ = x) como segue:

f (a,b,c) = a’b + ac’ => (f(a,b,c)’)’ = ((a’b + ac’)’)’

Então aplicamos o teorema de De Morgan ((x + y)’ = x’ . y’):

f (a,b,c) = ((a’b + ac’)’)’ = ((a’b)’. (ac’)’)’ => NE-NE

Nesta forma, esta função pode ser implementada por dois níveis de portas lógicas NE-NE. A implementação desta função simplificada na forma NE-NE é mostrada na Figura 5.3.

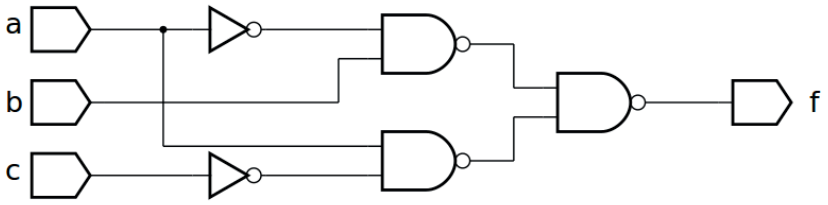


Figura 5.3 – A implementação da função simplificada f(a,b,c) = (2,3,4,6) = ((a’b)’. (ac’)’)’ na forma NE-NE.

d) Para obter a forma de dois níveis de portas lógicas OU-NE para a função, aplicamos o teorema de De Morgan ((x . y)’ = x’ + y’): f(a,b,c) = ((a’b)’. (ac’)’)’ = ((a + b’) . (a’ + c))’ => OU-NE

Nesta forma, esta função pode ser implementada por dois níveis de portas lógicas OU-NE. A implementação desta função simplificada na forma OU-NE é mostrada na Figura 5.4.

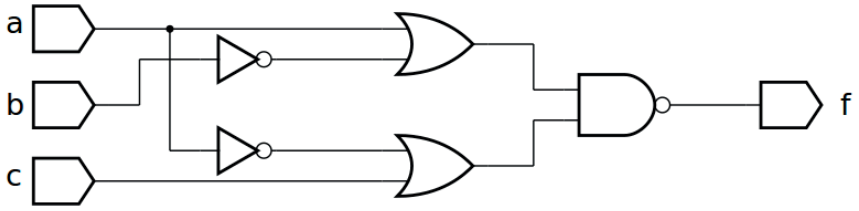


Figura 5.4 – A implementação da função simplificada f(a,b,c) = (2,3,4,6) = ((a + b’) . (a’ + c))’ na forma OU-NE.

e) Para obter a forma de dois níveis de portas lógicas NOU-OU para a função, aplicamos o teorema de De Morgan ((x . y)’ = x’+ y’):

f(a,b,c) = ((a + b’) . (a’ + c))’ = (a + b’)’ . (a’ + c)’ => NOU-OU

Nesta forma, esta função pode ser implementada por dois níveis de portas lógicas NOU-OU. A implementação desta função simplificada na forma NOU-OU é mostrada na Figura 5.5.

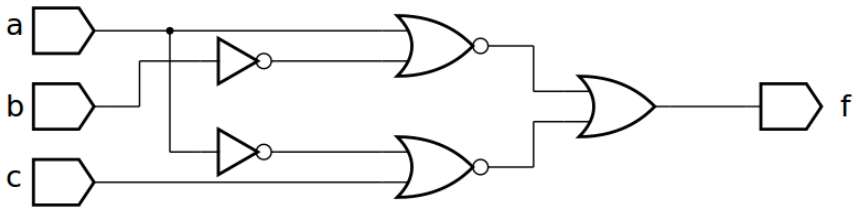


Figura 5.5 – A implementação da função simplificada f(a,b,c) = (2,3,4,6) = (a + b’)’ . (a’ + c)’ na forma NOU-OU.

f) A função booleana $f(a,b,c) = \Sigma_m(2,3,4,6) = m2 + m3 + m4 + m6 = a'bc' + a'bc + ab'c' + abc'$ é dada como soma de mintermos. Agora vamos expressá-lo como produto de somas:

$$f(a,b,c) = \Sigma_m(2,3,4,6) = \Pi_M(0,1,5,7) = M0 \,.\, M1 \,.\, M5 \,.\, M7$$

$$f(a,b,c) = \Pi_M(0,1,5,7) = (a+b+c).(a+b+c').(a'+b+c').(a'+b'+c')$$

g) A função booleana $f(a,b,c) = \Pi_M(0,1,5,7)$ expressa como produto de somas é simplificada usando a álgebra booleana como segue:

$$f(a,b,c) = (a+b+c).(a+b+c').(a'+b+c').(a'+b'+c')$$

e:

$$a+b = a+b+c.c' = (a+b+c).(a+b+c')$$

$$a'+c' = a'+c'+b.b' = (a'+b+c').(a'+b'+c')$$

então:

$$f(a,b,c) = (a+b).(a'+c')$$

Esta é a forma simplificada da função no formato OU-E.

h) A forma simplificada da função como produto de somas é a seguinte: $f(a,b,c) = (a+b).(a'+c')$. Nesta forma, esta função pode ser diretamente implementada por dois níveis de portas lógicas OU-E. A implementação desta função simplificada na forma OU-E é mostrada na Figura 5.6.

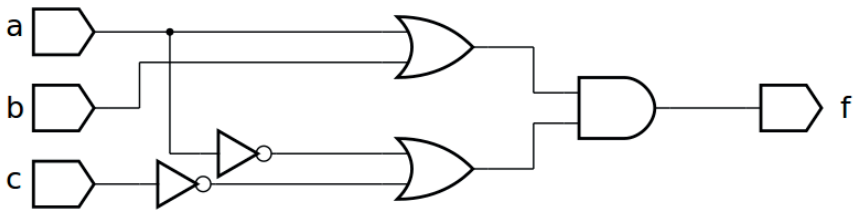


Figura 5.6 – A implementação da função simplificada $f(a,b,c) = \Sigma_m(2,3,4,6) = \Pi_M(0,1,5,7) = (a+b).(a'+c')$ na forma OU-E.

i) Para obter a forma de dois níveis de portas lógicas NOR-NOR para a função $f(a,b,c) = (a+b).(a'+c')$, primeiro usamos a lei de involução ($(x')' = x$) como segue:

$$f(a,b,c) = ((f(a,b,c))')' = (((a+b).(a'+c'))')'$$

Então aplicamos o teorema de De Morgan ($(x \,.\, y)' = x' + y'$):

$$f(a,b,c) = (((a+b).(a'+c'))')' = ((a+b)'+(a'+c')')' \Rightarrow \text{NOU-NOU}$$

Nesta forma, esta função pode ser implementada por dois níveis de portas lógicas NOU--NOU. A implementação desta função simplificada na forma NOU-NOU é mostrada na Figura 5.7.

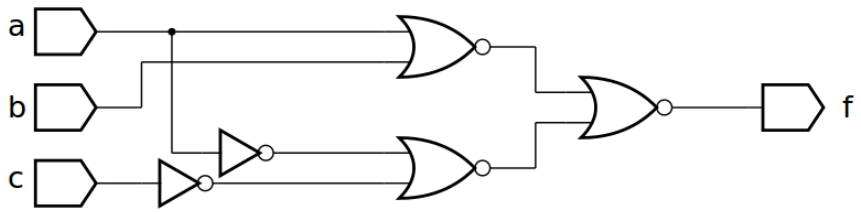


Figura 5.7 – A implementação da função simplificada
f(a,b,c) = Σ_m(2,3,4,6) = Π_M(0,1,5,7) = ((a+b)'+(a'+c')')' na forma NOU-NOU.

j) Para obter dois níveis de portas lógicas E-NOU para a função, aplicamos o teorema de De Morgan ((x + y)' = x' . y'): f(a,b,c) = ((a+b)'+(a'+c')')' = ((a' . b') + (a . c))' => E-NOU

Nesta forma, esta função pode ser implementada por dois níveis de portas lógicas E-NOU. A implementação desta função simplificada na forma E-NOU é mostrada na Figura 5.8.

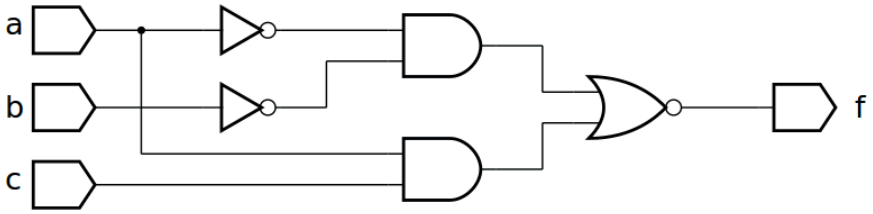


Figura 5.8 – A implementação da função simplificada
f(a,b,c) = Σ_m(2,3,4,6) = Π_M(0,1,5,7) = ((a' . b') + (a . c))' na forma E-NOU.

k) Para obter a forma de dois níveis de portas lógicas NE-E para a função, aplicamos o teorema de De Morgan ((x + y)' = x' . y'):

f(a,b,c) = ((a' . b') + (a . c))' = (a' . b')' . (a . c)' => NE-E

Nesta forma, esta função pode ser implementada por dois níveis de portas lógicas NE-E. A implementação desta função simplificada na forma NE-E é mostrada na Figura 5.9.

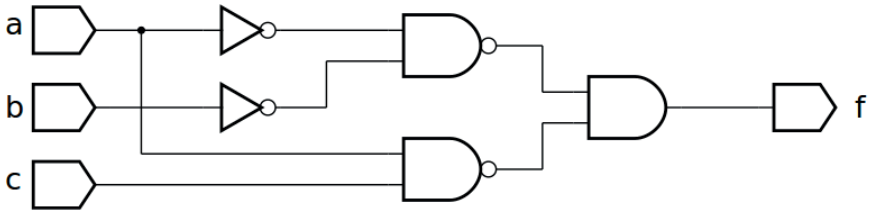


Figura 5.9 – A implementação da função simplificada
f(a,b,c) = (2,3,4,6) = (0,1,5,7) = (a' . b')' . (a . c)' na forma NE-E.

5.2.7. Equivalentes de portas NE e NOU

Na seção anterior, oito formas de implementação de funções booleanas usando portas lógicas foram consideradas. Além disso, existem mais dois métodos considerados aqui. Esses métodos são baseados em equivalentes de portas NE e NOU. As portas NE são uma das duas portas lógicas básicas (junto com as portas NOU) das quais qualquer outra porta lógica pode ser construída, como pode ser visto na Tabela 5.9. Esses métodos podem ser chamados como:

1. A implementação de uma função booleana usando portas Equivalente NE.
2. A implementação de uma função booleana usando portas Equivalente NOU.

Assume-se que a função a ser implementada é dada de forma simplificada. No primeiro método, a função é implementada na forma simplificada da soma dos produtos com portas E-OU (mais inversor – porta INV) ou na forma simplificado de produto de somas com portas OU-E (mais inversor – porta INV). Então, as portas E, OU e INV usadas na implementação da função são substituídas por seus equivalentes NE, conforme mostrado na Tabela 5.9. Como resultado, obtemos a implementação de uma função booleana usando Equivalentes NE. O segundo método é semelhante, no qual portas E, OU e INV usadas na implementação da função são substituídas por seus equivalentes NOU. Quando obtivermos a implementação de uma função com equivalentes NE ou NOU, na etapa final, sucessivamente conectados, duas portas INV podem ser excluídas para simplificar ainda mais a implementação.

Porta	Símbolo	Equivalente NE	Equivalente NOU
INV			
E			
OU			
NE			
NOU			

Tabela 5.9 – Equivalentes de portas NE e NOU.

Vamos agora considerar um exemplo de implementação usando apenas portas NE e NOU.

Exemplo: Para uma função booleana dada como a soma dos produtos f(a,b,c) = a'b + ac':

a) Implemente essa função usando equivalentes NE.

b) Implemente essa função usando equivalentes NOU.

a) A função f(a,b,c) = a'b + ac' pode ser diretamente implementada por dois níveis de portas lógicas E-OU. A implementação desta função na forma E-OU é mostrada na Figura 5.10.

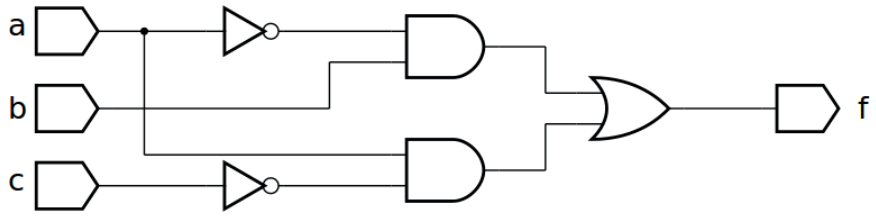


Figura 5.10 – A implementação da função f(a,b,c) = a'b + ac' na forma E-OU.

Quando as portas E, OU e INV usadas na implementação da função são substituídas por seus equivalentes NE, como mostrado na Tabela 5.9, obtemos a implementação da função booleana usando equivalentes NE como mostrado na Figura 5.11. Pode-se ver que existem dois conjuntos de duas portas INV conectadas sucessivamente, ou seja, as portas 5 e 7 e as portas 6 e 8. Isso significa que essas portas podem ser excluídas para simplificar ainda mais a implementação.

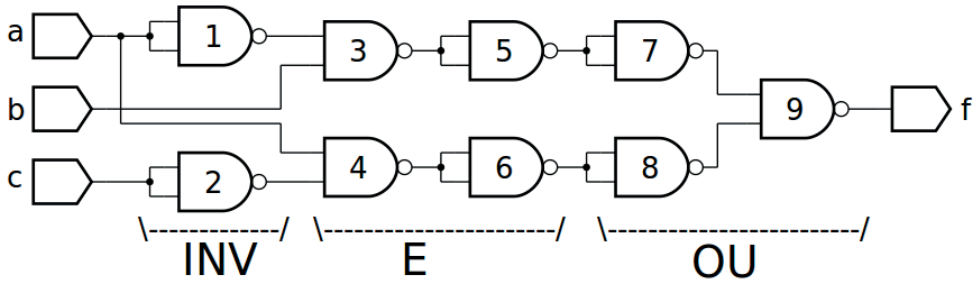


Figura 5.11 – A implementação da função f(a,b,c) = a'b + ac' usando equivalentes NE.

Quando as portas 5, 6, 7 e 8, mostradas na Figura 5.11, são deletadas, obtemos a implementação da função f(a,b,c) = a'b + ac' usando equivalentes NE como mostrado na Figura 5.12.

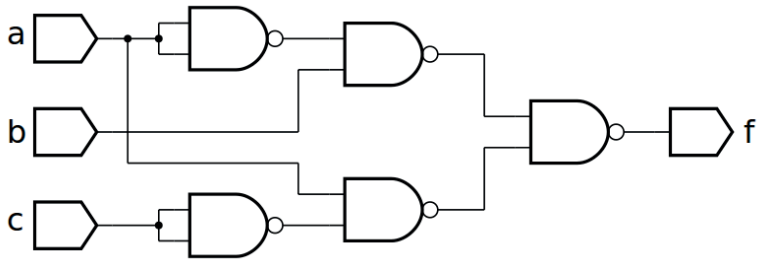


Figura 5.12 – A implementação da função f(a,b,c) = a'b + ac' usando equivalentes NE (mais simplificado)

b) A função f(a,b,c) = a’b + ac’ pode ser implementada por dois níveis de portas lógicas E-OU. A implementação desta função na forma E-OU é mostrada na Figura 5.13.

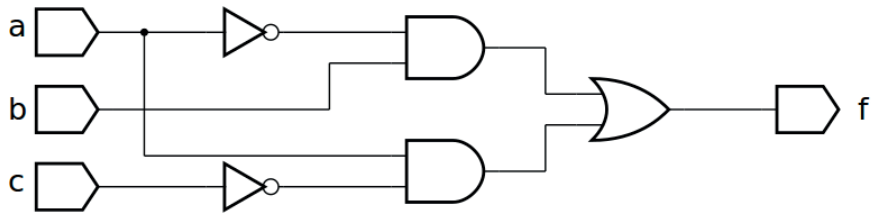


Figura 5.13 – A implementação da função f(a,b,c) = a’b + ac’ na forma E-OU.

Quando as portas E, OU e INV usadas na implementação da função são substituídas por seus equivalentes NOU, como mostrado na Tabela 5.9, obtemos a implementação da função booleana usando equivalentes de NOU, como mostrado na Figura 5.14. Pode-se ver que há dois conjuntos de duas portas INV conectadas sucessivamente, ou seja, as portas 1 e 3 e as portas 2 e 6. Isso significa que essas portas podem ser excluídas para simplificar ainda mais a implementação.

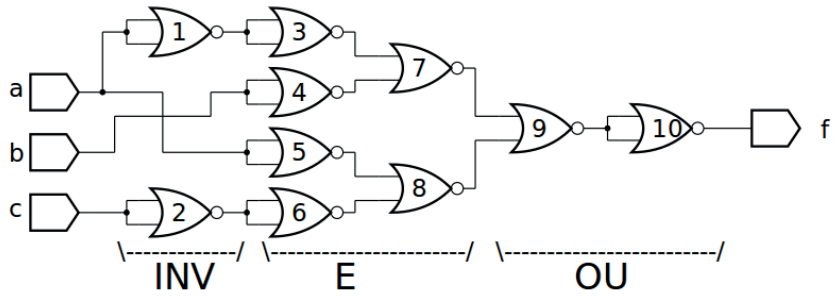


Figura 5.14 – A implementação da função f(a,b,c) = a’b + ac’ usando equivalentes do NOU.

Quando as portas 1, 2, 3 e 6, mostradas na Figura 5.14, são deletadas, obtemos a implementação da função f(a,b,c) = a’b + ac’ usando equivalentes de NOU como mostrado na Figura 5.15.

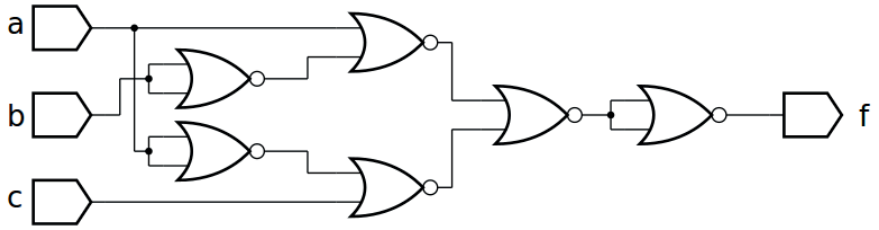


Figura 5.15 – A implementação da função f(a,b,c) = a’b + ac’ usando equivalentes NOU (mais simplificado).

5.3. Atividades experimentais

5.3.1. Disposição padronizada na placa de contatos

Para os experimentos a seguir utilize a placa de contatos com os componentes dispostos de acordo com a Figura 5.16. Utilize um LED vermelho com um resistor de 1KΩ para indicar a saída "f" do circuito. Utilize 4 chaves do tipo push-button com um resistor de 10KΩ para representar as entradas "a", "b", "c" e "d". Utilize os fios verdes como entradas e o fio roxo como saída. Coloque os CIs nas posições indicadas de acordo com o tipo e quantidade indicada em cada experimento.

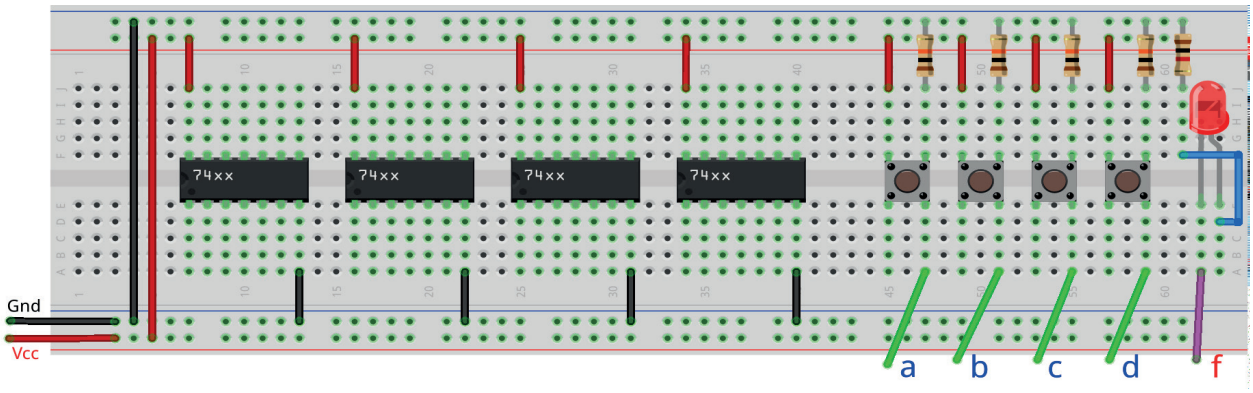


Figura 5.16 – Montagem inicial dos componentes na placa de contatos.

5.3.2. Diagrama de interligações dos circuitos integrados

Cada experimento contém uma disposição prévia dos circuitos integrados para você conectar antes da montagem na placa de contatos. Na figura 5.17, os fios verdes são as entradas "a", "b" e "c" provenientes dos botões, o fio roxo "f" vai para o LED e os fios azuis são de ligações internas entre as portas. É conveniente, mas não obrigatório, que os terminais de entrada das portas lógicas não utilizadas sejam conectadas em "Gnd".

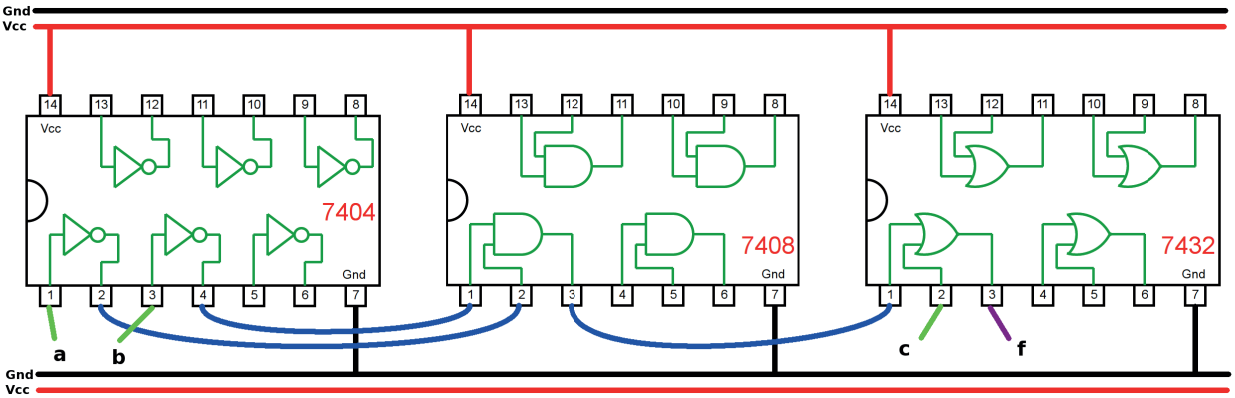


Figura 5.17 – Exemplo de ligações dos fios verdes, roxo e azuis.

Programas para o Arduíno:

Para simulação em Arduino, utilize os programas abaixo, substituindo a linha da função booleana, destacado em negrito, pela equação do experimento corrente. Use o programa de acordo com a quantidade de variáveis: 4 variáveis ou 3 variáveis.

```
// Tabela verdade para função de 4 variáveis

byte a;
byte b;
byte c;
byte d;
byte f;

void setup(){

  Serial.begin(9600);

  Serial.println("| a | b | c | d || f |");
  Serial.println("----------------------");
  for (a = 0; a <= 1; a++){
    for (b = 0; b <= 1; b++){
      for (c = 0; c <= 1; c++){
        for (d = 0; d <= 1; d++){
        f = (a && b) || (c && d); // função booleana!
        Serial.print("| ");
        Serial.print(a);
        Serial.print(" | ");
        Serial.print(b);
        Serial.print(" | ");
        Serial.print(c);
        Serial.print(" | ");
        Serial.print(d);
        Serial.print(" || ");
        Serial.print(f);
        Serial.println(" |");
      }
    }
  }
}
} // fim do 'setup'
void loop(){
// nada a fazer aqui!
} // fim do 'loop'
```

```
// Tabela verdade para função de 3 variáveis
byte a;
byte b;
byte c;
byte f;
void setup(){
  Serial.begin(9600);
  Serial.println("| a | b | c || f |");
  Serial.println("------------------");
  for (a = 0; a <= 1; a++){
    for (b = 0; b <= 1; b++){
        for (c = 0; c <= 1; c++){
        f = (a && b) || (c); // função booleana!
        Serial.print("| ");
        Serial.print(a);
        Serial.print(" | ");
        Serial.print(b);
        Serial.print(" | ");
        Serial.print(c);
        Serial.print(" || ");
        Serial.print(f);
        Serial.println(" |");
      }
    }
  }
} // fim do 'setup'

void loop(){
// nada a fazer aqui!
} // fim do 'loop'
```

5.3.4. Implementação de uma função booleana na forma E-OU

Esquemas:

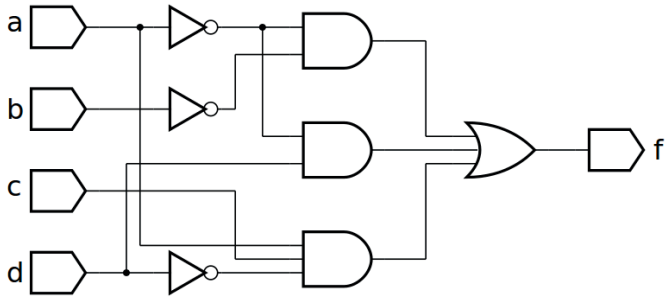


Figura 5.18 – Diagrama esquemático.

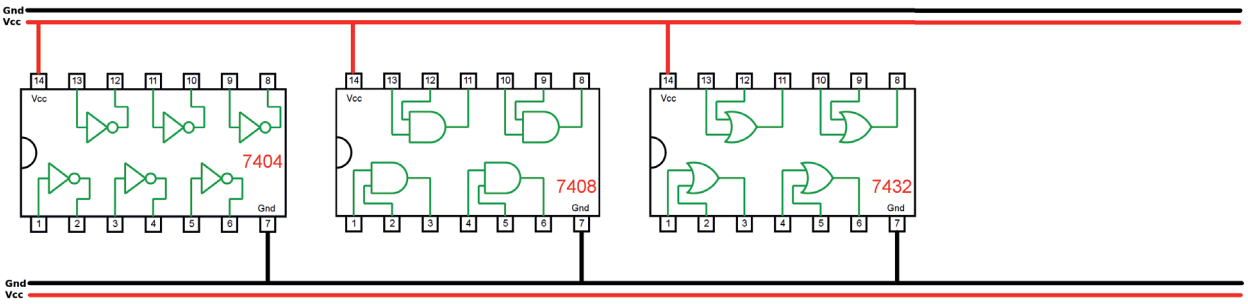


Figura 5.19 – Disposição dos componentes (TTL) para a placa de montagem.

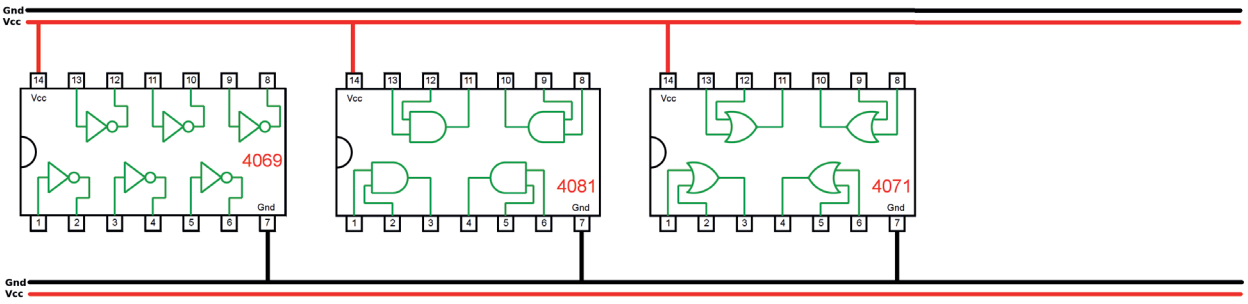


Figura 5.20 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c,d) = _____

Versão para o Arduíno:

`f =` _____

Procedimento:

1. Complete as ligações na figura 5.19 ou 5.20 de acordo com a figura 5.18.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.

2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.10, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.10, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.18. Compare os resultados com a tabela 5.10.
6. Escreva a equação booleana do circuito da figura 5.18 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	d	f
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Tabela 5.10

5.3.5. Implementação de uma função booleana na forma OU-E

Esquemas:

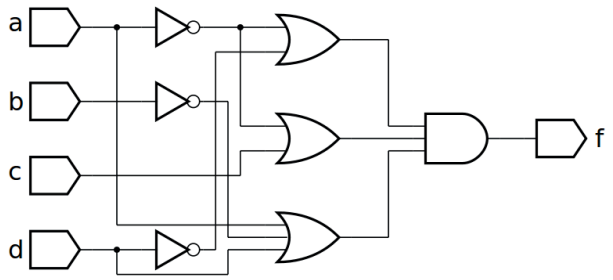


Figura 5.21 – Diagrama esquemático.

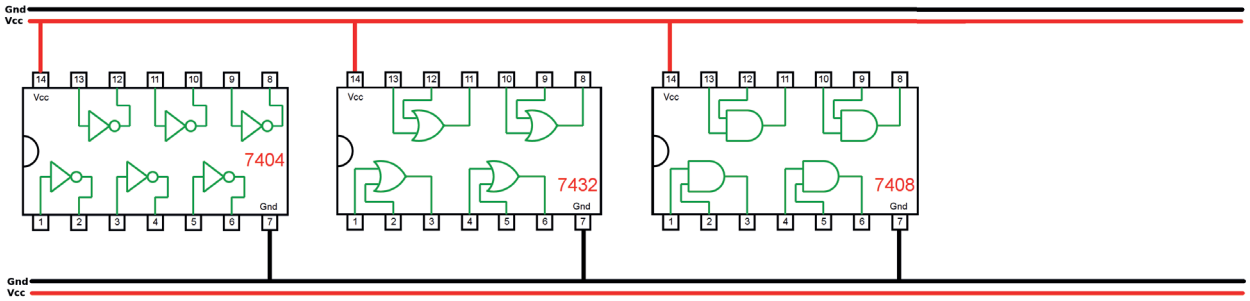


Figura 5.22 – Disposição dos componentes (TTL) para a placa de montagem.

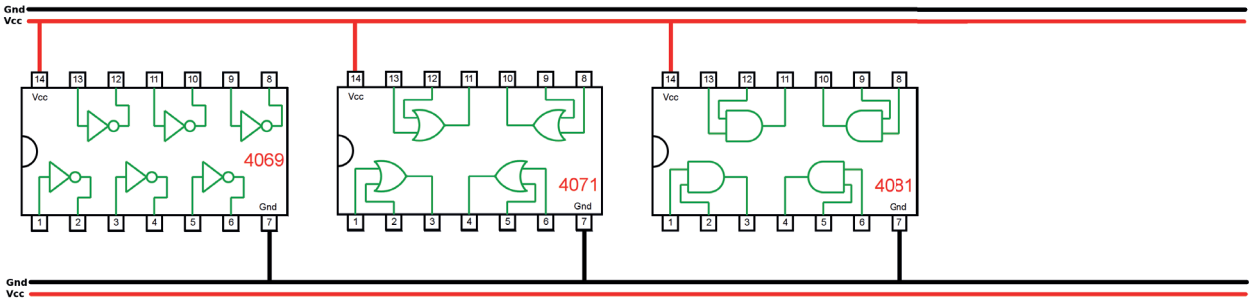


Figura 5.23 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c,d) = _____

Versão para o Arduíno:

`f =` _____

Procedimento:

1. Complete as ligações na figura 5.22 ou 5.23 de acordo com a figura 5.21.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.

2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da Tabela 5.11, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.11, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.21. Compare os resultados com a tabela 5.5.
6. Escreva a equação booleana do circuito da figura 5.21 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	d	f
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Tabela 5.11

5.3.6. Implementação de uma função booleana na forma E-OU (1)

Esquemas:

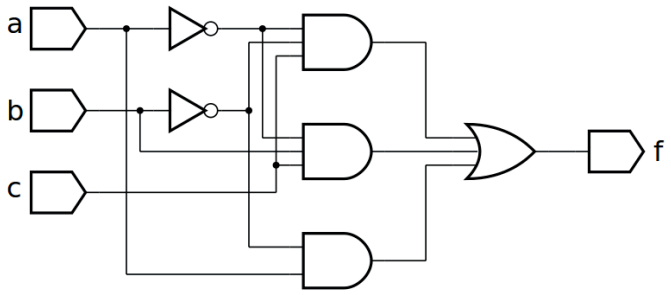


Figura 5.24 – Diagrama esquemático.

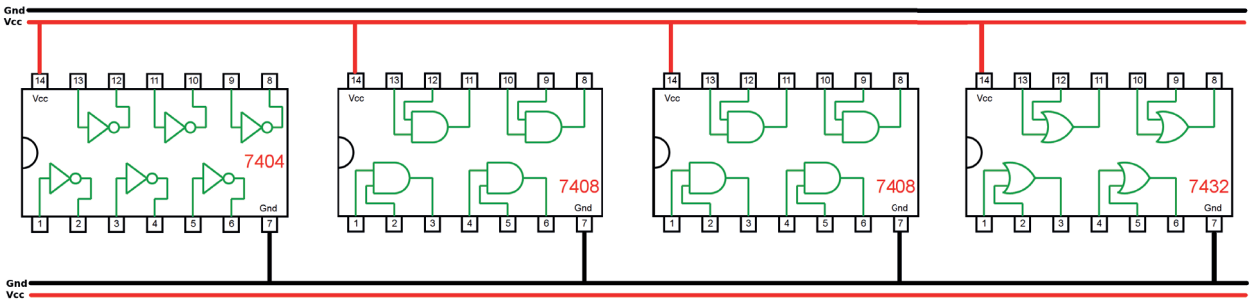


Figura 5.25 – Disposição dos componentes (TTL) para a placa de montagem.

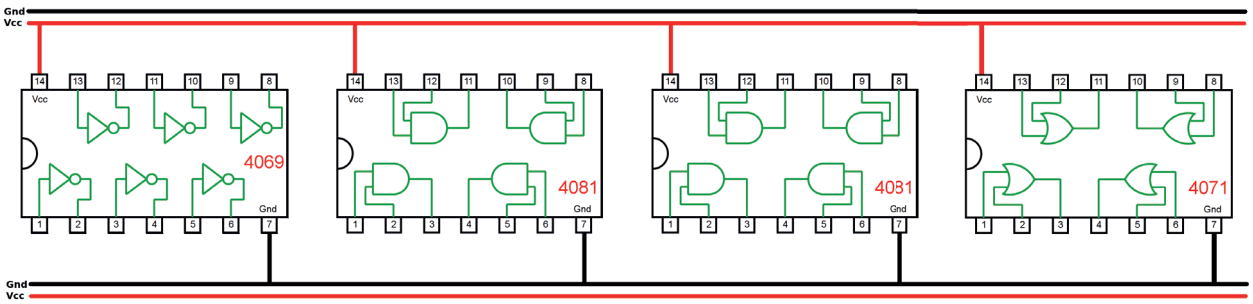


Figura 5.26 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.25 ou 5.26 de acordo com a figura 5.24.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da Tabela 5.12, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na Tabela 5.12, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.24. Compare os resultados com a Tabela 5.12.
6. Escreva a equação booleana do circuito da figura 5.24 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.12

5.3.7. Implementação de uma função booleana na forma E-OU (2)

Esquemas:

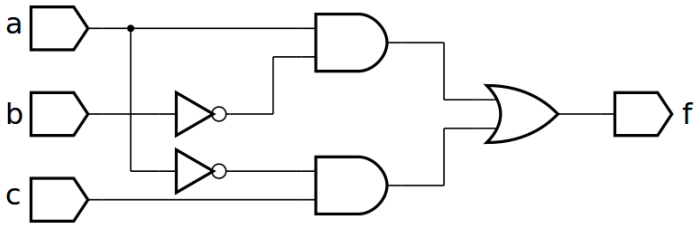


Figura 5.27 – Diagrama esquemático.

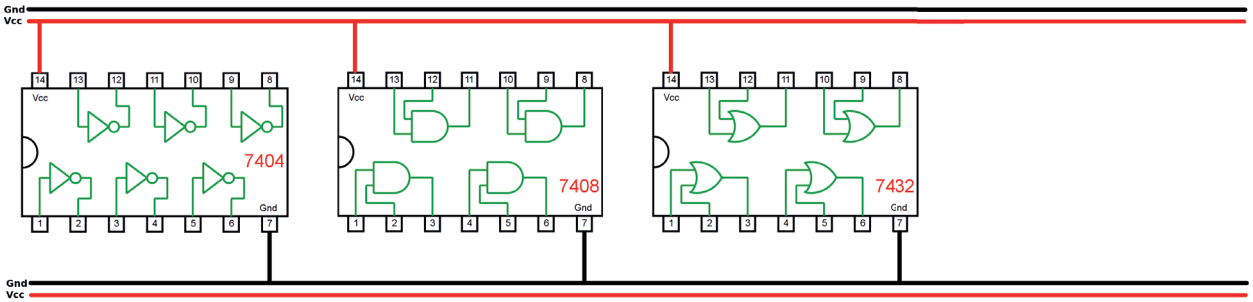


Figura 5.28 – Disposição dos componentes (TTL) para a placa de montagem.

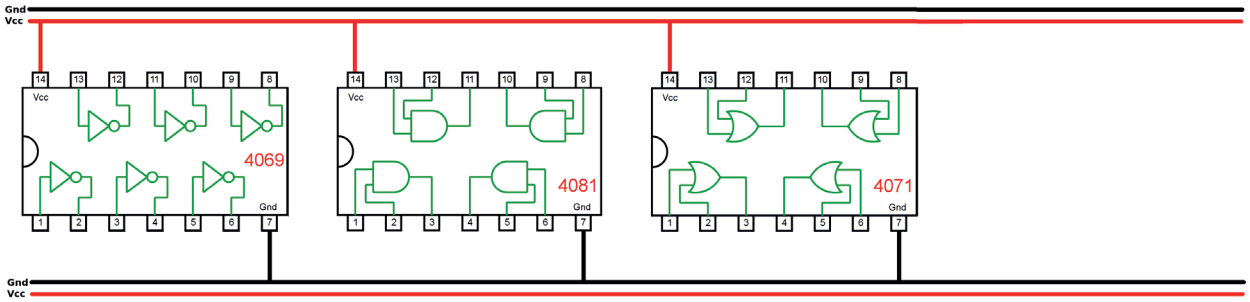


Figura 5.29 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.28 ou 5.29 de acordo com a figura 5.27.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da Tabela 5.13, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na Tabela 5.13, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.27. Compare os resultados com a Tabela 5.13.
6. Escreva a equação booleana do circuito da figura 5.27 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.13

5.3.8. Implementação de uma função booleana na forma E-OU (3)

Esquemas:

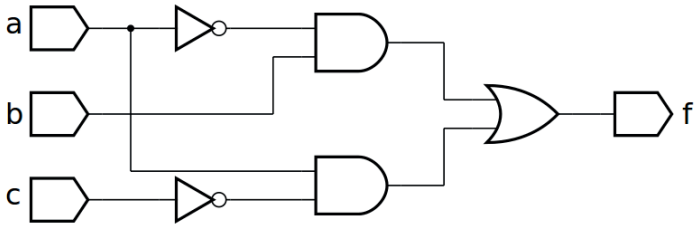


Figura 5.30 – Diagrama esquemático.

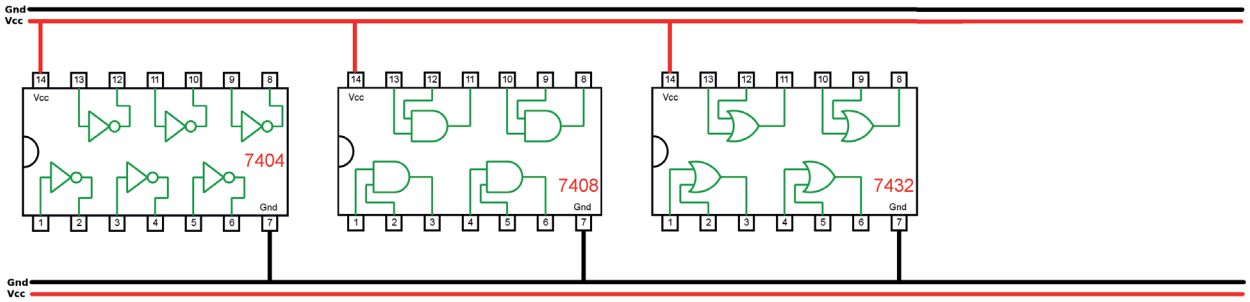


Figura 5.31 – Disposição dos componentes (TTL) para a placa de montagem.

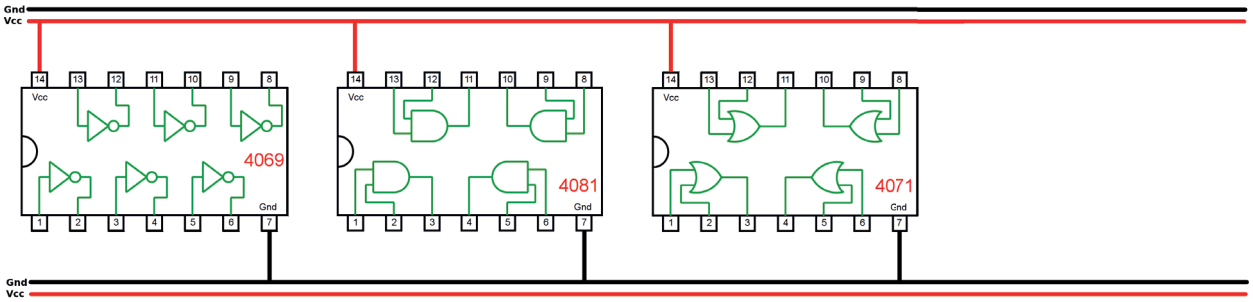


Figura 5.32 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.31 ou 5.32 de acordo com a figura 5.30.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da Tabela 5.14, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na Tabela 5.14, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.30. Compare os resultados com a Tabela 5.14.
6. Escreva a equação booleana do circuito da figura 5.30 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.14

5.3.9. Implementação de uma função booleana na forma NE-NE

Esquemas:

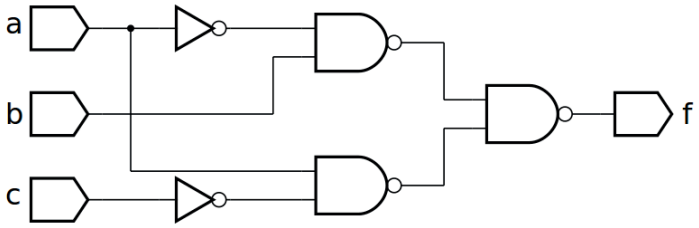


Figura 5.33 – Diagrama esquemático.

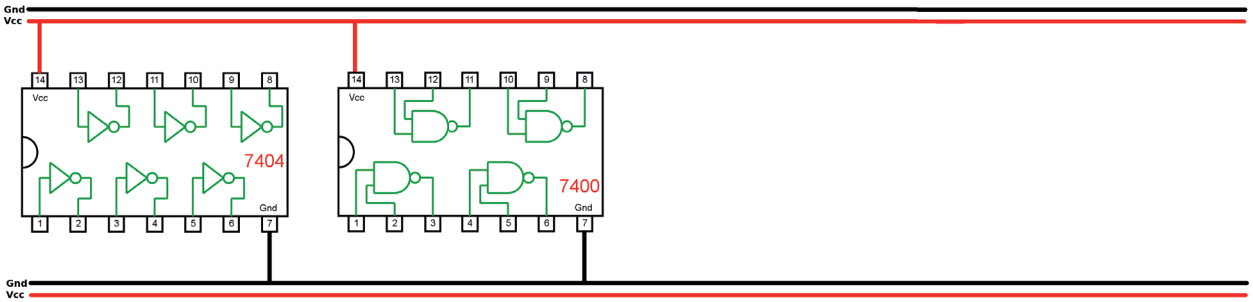


Figura 5.34 – Disposição dos componentes (TTL) para a placa de montagem.

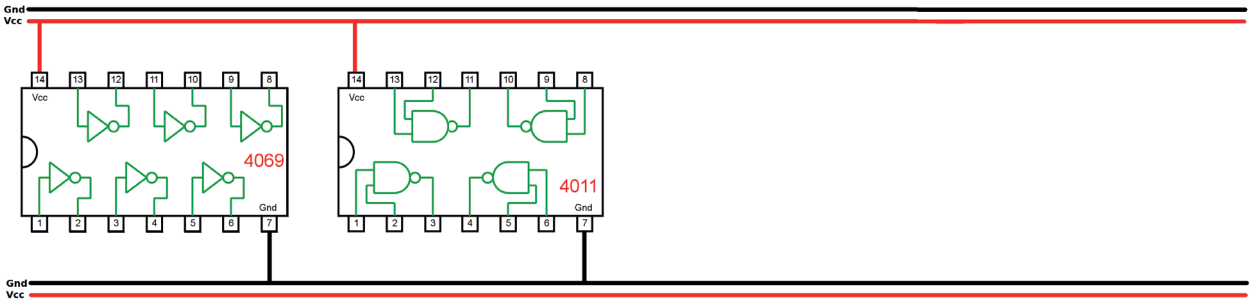


Figura 5.35 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.34 ou 5.35 de acordo com a figura 5.33.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.15, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.15, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.33. Compare os resultados com a tabela 5.15.
6. Escreva a equação booleana do circuito da figura 5.33 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.15

5.3.10. Implementação de uma função booleana na forma OU-NE

Esquemas:

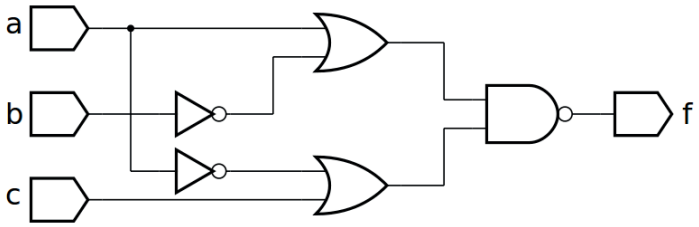


Figura 5.36 – Diagrama esquemático.

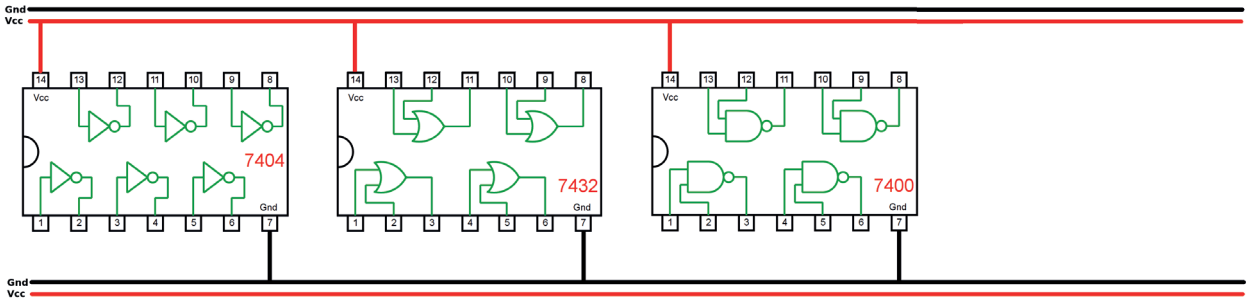


Figura 5.37 – Disposição dos componentes (TTL) para a placa de montagem.

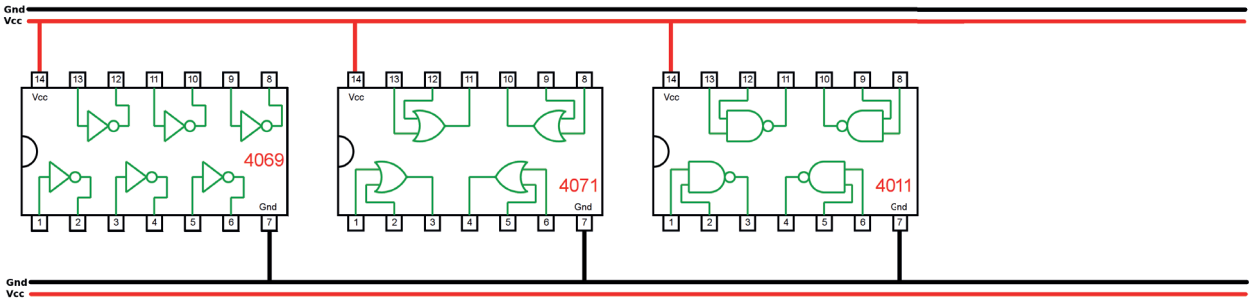


Figura 5.38 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.37 ou 5.38 de acordo com a figura 5.36.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhs azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.16, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.16, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.36. Compare os resultados com a tabela 5.16.
6. Escreva a equação booleana do circuito da figura 5.36 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.16

5.3.11. Implementação de uma função booleana na forma NOU-OU

Esquemas:

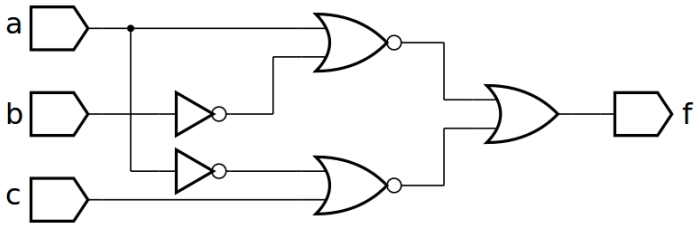


Figura 5.39 – Diagrama esquemático.

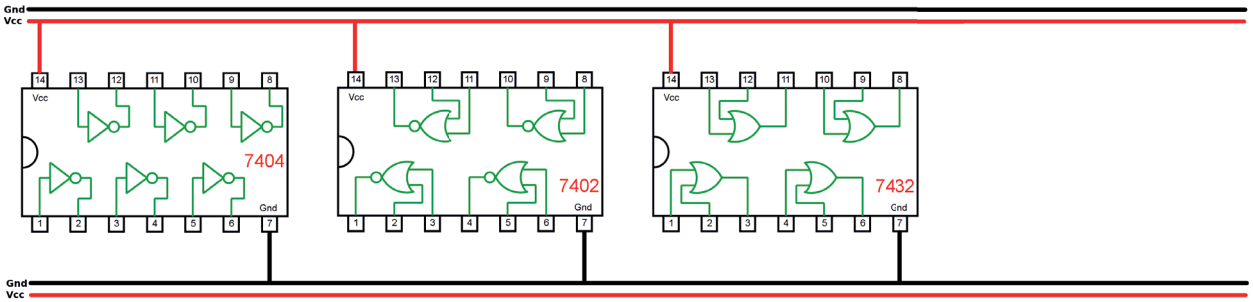


Figura 5.40 – Disposição dos componentes (TTL) para a placa de montagem.

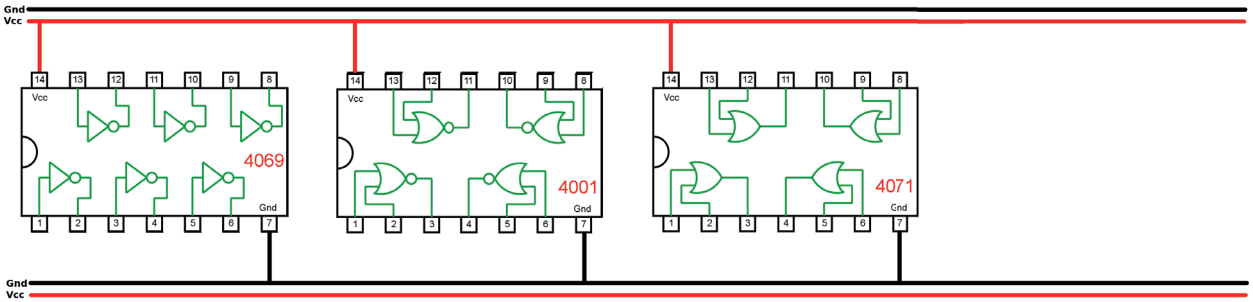


Figura 5.41 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.40 ou 5.41 de acordo com a figura 5.39.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.17, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.17, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.39. Compare os resultados com a tabela 5.17.
6. Escreva a equação booleana do circuito da figura 5.39 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.17

5.3.12. Implementação de uma função booleana na forma OU-E

Esquemas:

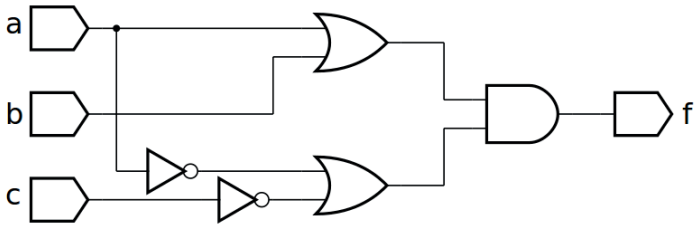


Figura 5.42 – Diagrama esquemático.

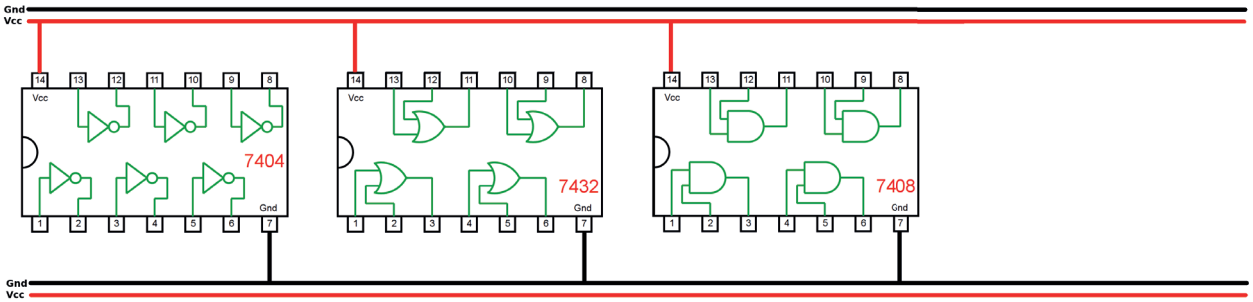


Figura 5.43 – Disposição dos componentes (TTL) para a placa de montagem.

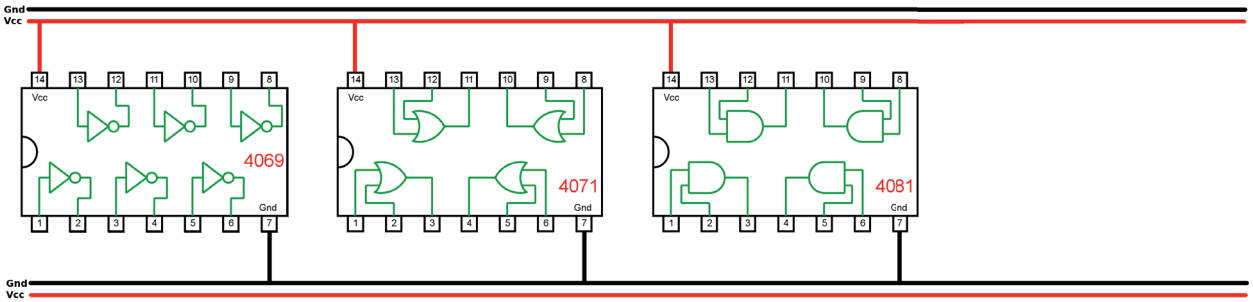


Figura 5.44 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.43 ou 5.44 de acordo com a figura 5.42.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.18, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.18, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.42. Compare os resultados com a tabela 5.18.
6. Escreva a equação booleana do circuito da figura 5.42 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.18

5.3.13. Implementação de uma função booleana na forma NOU-NOU

Esquemas:

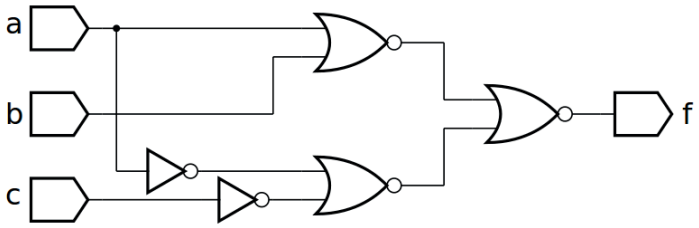


Figura 5.45 – Diagrama esquemático.

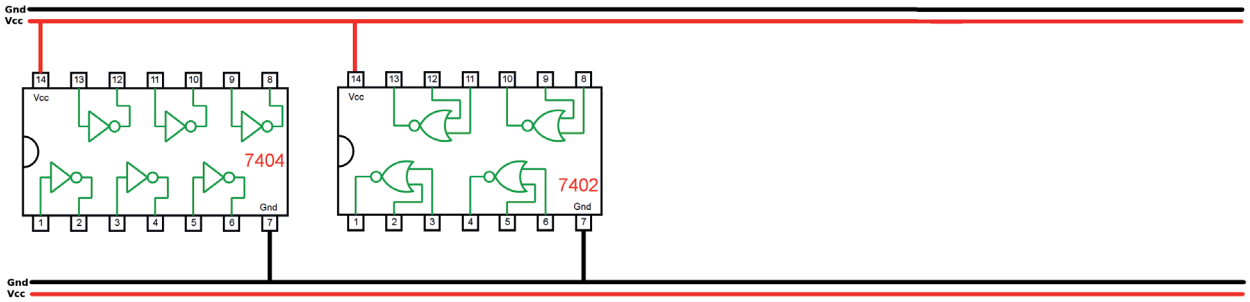


Figura 5.46 – Disposição dos componentes (TTL) para a placa de montagem.

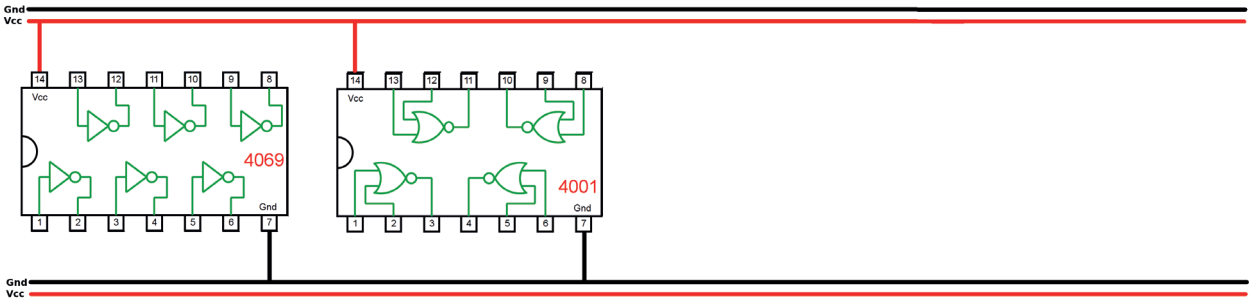


Figura 5.47 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.46 ou 5.47 de acordo com a figura 5.45.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.19, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.19, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.45. Compare os resultados com a tabela 5.19.
6. Escreva a equação booleana do circuito da figura 5.45 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.19

5.3.14. Implementação de uma função booleana na forma E-NOU

Esquemas:

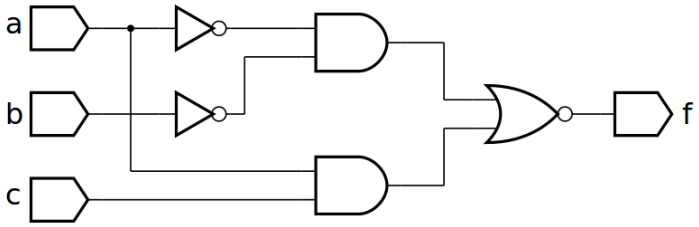


Figura 5.48 – Diagrama esquemático.

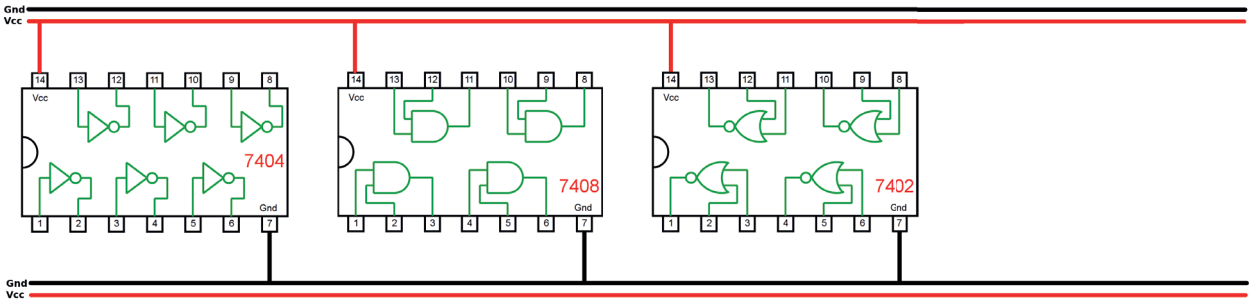


Figura 5.49 – Disposição dos componentes (TTL) para a placa de montagem.

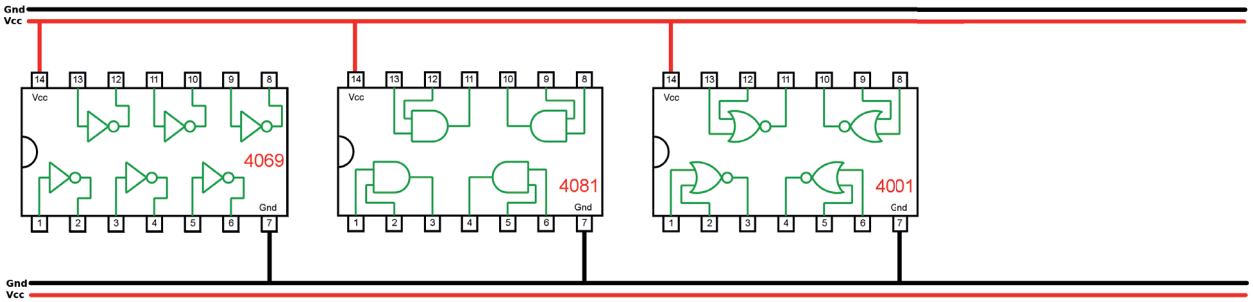


Figura 5.50 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.49 ou 5.50 de acordo com a figura 5.48.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.20, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.20, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.48. Compare os resultados com a tabela 5.20.
6. Escreva a equação booleana do circuito da figura 5.48 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.20

5.3.15. Implementação de uma função booleana na forma NE-E

Esquemas:

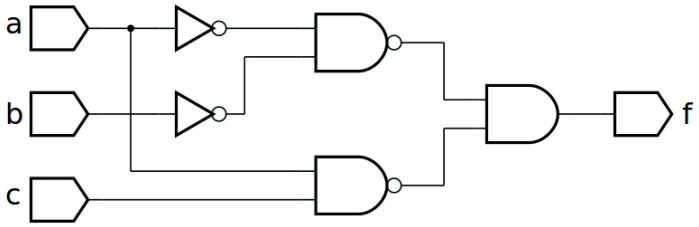


Figura 5.51 – Diagrama esquemático.

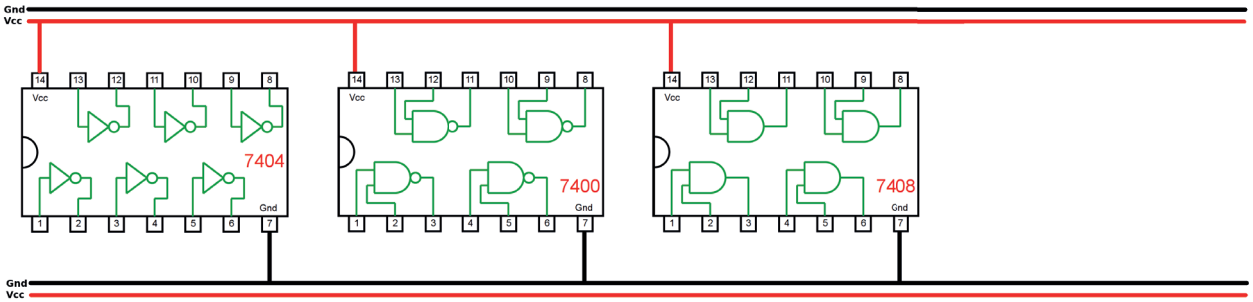


Figura 5.52 – Disposição dos componentes (TTL) para a placa de montagem.

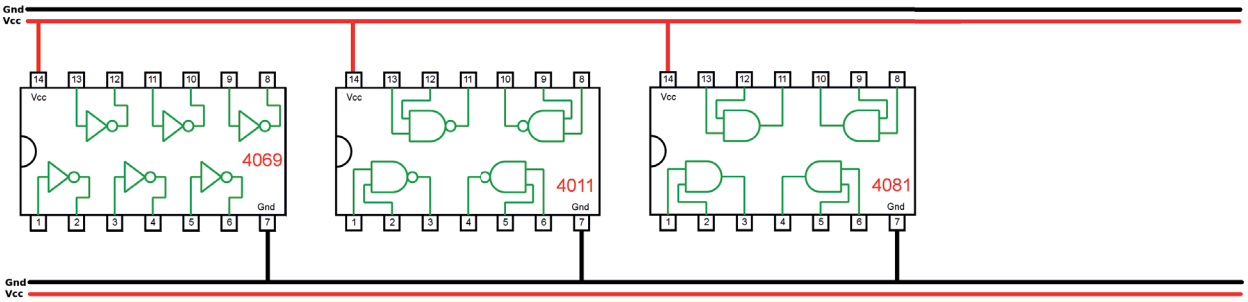


Figura 5.53 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.52 ou 5.53 de acordo com a figura 5.51.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.21, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.21, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.51. Compare os resultados com a tabela 5.21.
6. Escreva a equação booleana do circuito da figura 5.51 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.21

5.3.16. Implementação de uma função booleana usando equivalentes NE

Esquemas:

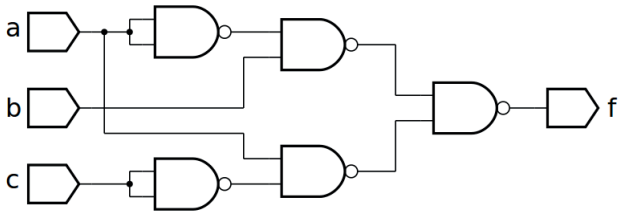


Figura 5.54 – Diagrama esquemático.

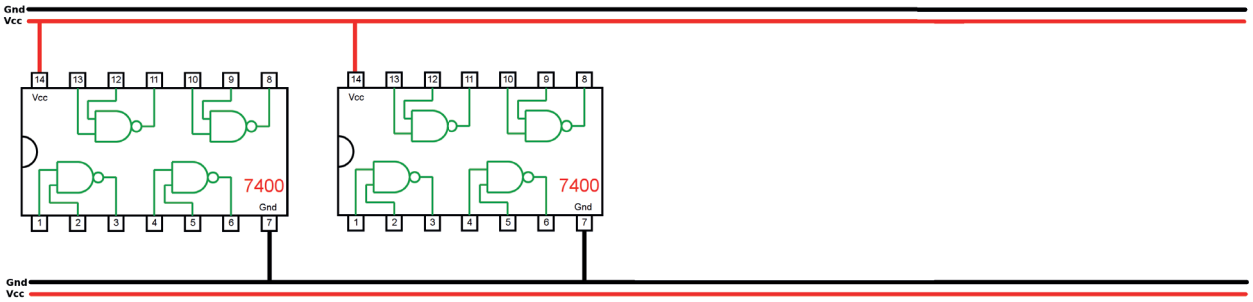


Figura 5.55 – Disposição dos componentes (TTL) para a placa de montagem.

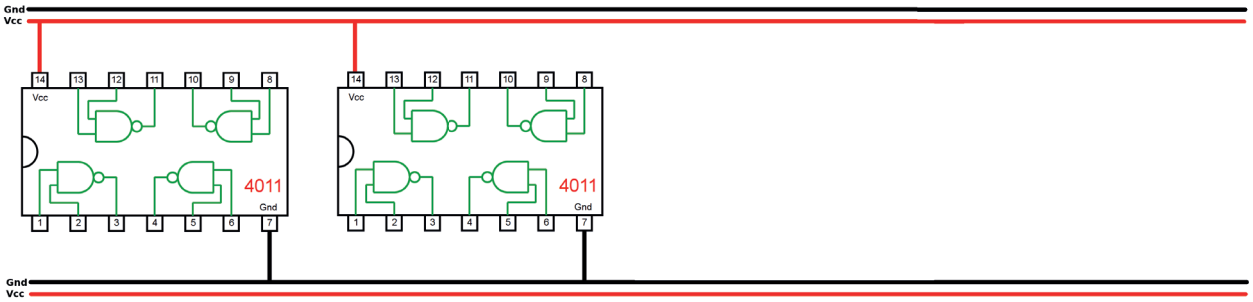


Figura 5.56 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.55 ou 5.56 de acordo com a figura 5.54.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.22, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.22, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.54. Compare os resultados com a tabela 5.22.
6. Escreva a equação booleana do circuito da figura 5.54 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.22

5.3.17. Implementação de uma função booleana usando equivalentes NOU

Esquemas:

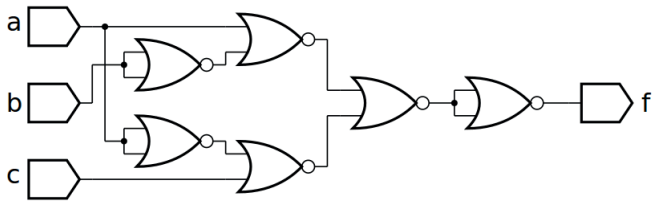


Figura 5.57 – Diagrama esquemático.

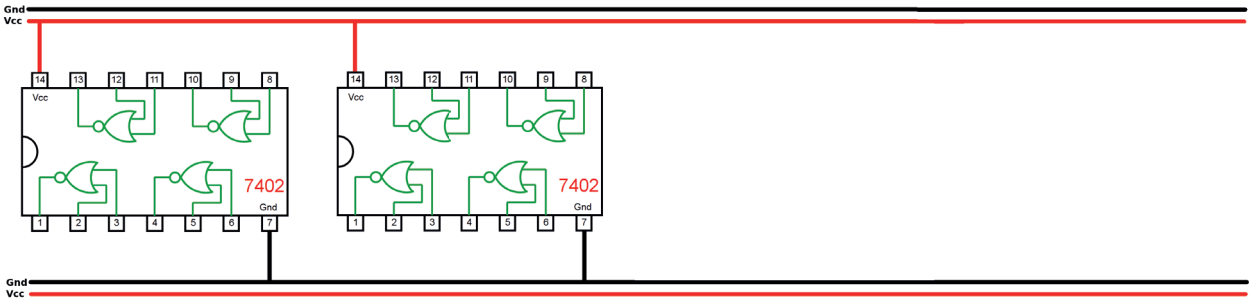


Figura 5.58 – Disposição dos componentes (TTL) para a placa de montagem.

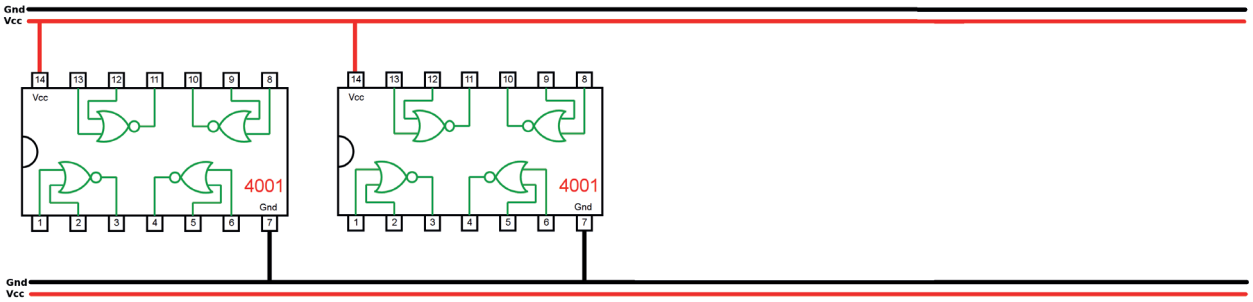


Figura 5.59 – Disposição dos componentes (CMOS) para a placa de montagem.

Equação booleana:

f(a,b,c) = _____

Versão para o Arduíno:

f = _____

Procedimento:

1. Complete as ligações na figura 5.58 ou 5.59 de acordo com a figura 5.57.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “a”, b”, “c” e “d”.
   c) Identifique o terminal do CI que receberá o fio roxo da saída com a letra “f”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem.
   a) Conecte os fios de entrada (verdes) e de saída (roxo).
   b) Conecte os fios de conexão entre as portas lógicas (azuis).
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 5.23, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “f” na tabela 5.23, considere:
   a) Se o LED estiver apagado então f = 0.
   b) Se o LED estiver aceso então f = 1.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 5.57. Compare os resultados com a tabela 5.23.
6. Escreva a equação booleana do circuito da figura 5.57 e simule no programa para o Arduíno.

Tabelas de dados:

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tabela 5.23

Capítulo 6. Circuitos lógicos combinacionais aritméticos

6.1. Somadores binários

6.1.1. Objetivos

1. Investigar os circuitos somadores binários,
2. Observar seu funcionamento e obter suas tabelas verdade,
3. Conhecer os CIs de circuitos somadores binários.

6.1.2. Informação preliminar

Circuitos que realizam adição, multiplicação de subtração e divisão com números binários são chamados de circuitos aritméticos. Parece que há quatro operações, mas na verdade, adições consecutivas são realizadas para multiplicação, e subtrações consecutivas são realizadas para divisão. Existem basicamente dois tipos de adição em circuitos lógicos. Os circuitos que executam a adição de dois bits são chamados de "meio somadores", circuitos que executam a adição de três bits são chamados de "somadores completos".

As regras a seguir são aplicáveis para somadores.

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 1 0 (Soma = 0, Transporte = 1)

O circuito meio somador é mostrado na Figura 6.1, e sua tabela verdade é fornecida na Tabela 6.1.

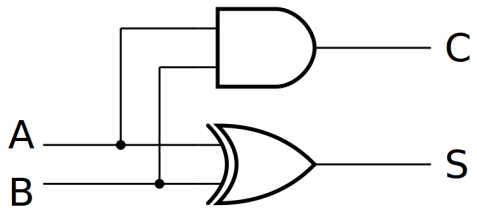


Figura 6.1 – O circuito do meio somador.

Entradas		Saídas	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabela 6.1 – A tabela verdade do meio somador.

6.1.3. Exame do meio somador

Esquemas:

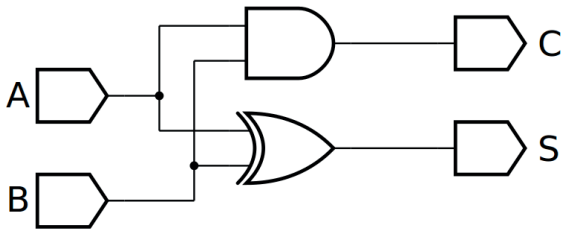


Figura 6.2 – Diagrama esquemático.

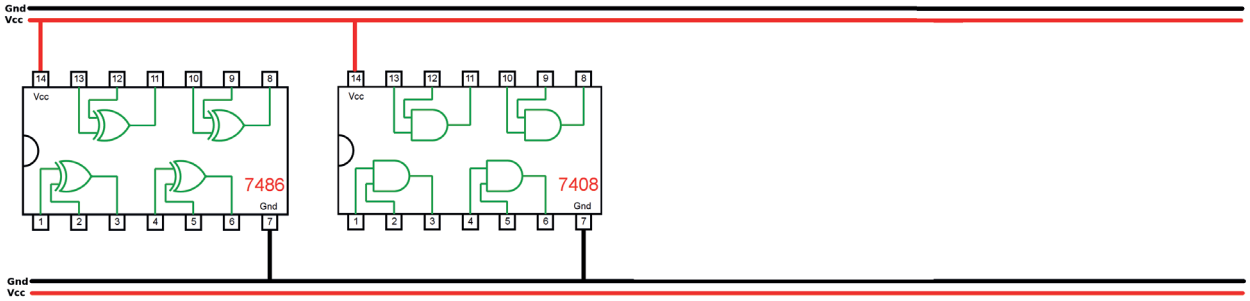


Figura 6.3 – Disposição dos componentes (TTL) para a placa de montagem.

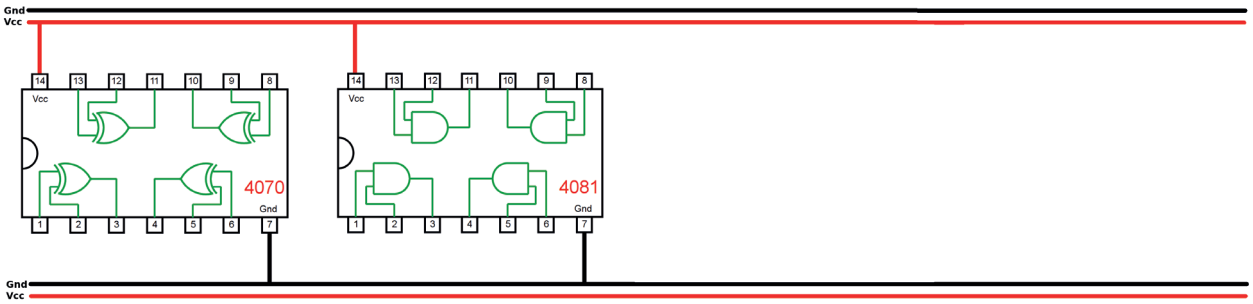


Figura 6.4 – Disposição dos componentes (CMOS) para a placa de montagem.

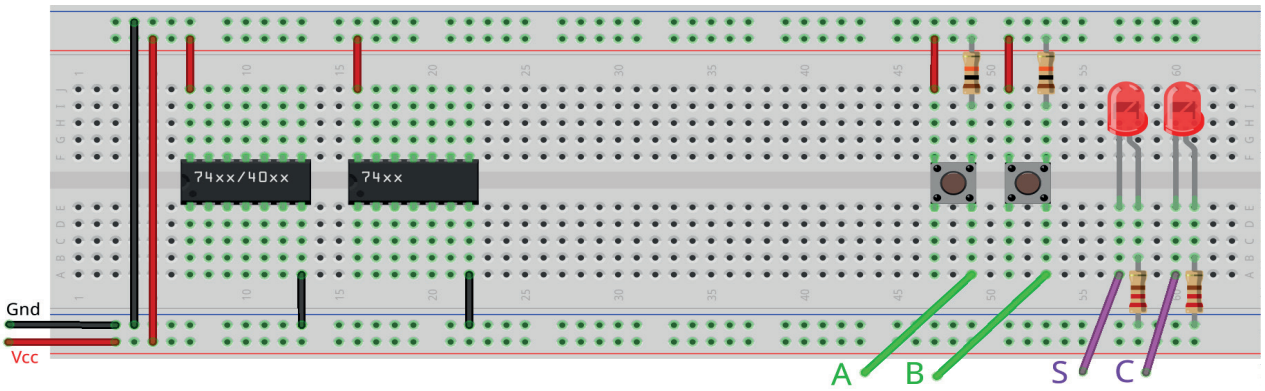


Figura 6.5 – Disposição dos componentes na placa de montagem.

Equação booleana:

S(A,B) = _____ C(A,B) = _____

Versão para o Arduíno:

s = _____ c = _____

Procedimento:

1. Complete as ligações na figura 6.3 ou 6.4 de acordo com a figura 6.2.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “A” e “B”.
   c) Identifique os terminais dos CIs que receberão os fios roxos das saídas com as letras “S” e “C”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem, conforme figura 6.5.
   a) Conecte os fios de entrada (verdes) aos botões e de saída (roxos) aos LEDs.
   b) Conecte os fios de conexão (azuis) entre as portas lógicas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 6.2, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “S” e “C” na tabela 6.2, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 6.2. Compare os resultados com a tabela 6.2.
6. Escreva a equação booleana do circuito da figura 6.2 e simule no programa para o Arduíno.

Tabelas de dados:

Entradas		Saídas	
A	B	S	C
0	0		
0	1		
1	0		
1	1		

Tabela 6.2

Programa para o Arduíno:

```
// Meio Somador

byte a; byte b; byte s; byte c;

void meio_somador(byte a, byte b){
  s = (!a && b) || (a && !b); // função booleana!
  c = (a && b); // função booleana!
}

void mostra4(){
  Serial.print("| ");
  Serial.print(a);
  Serial.print(" | ");
  Serial.print(b);
  Serial.print(" || ");
  Serial.print(s);
  Serial.print(" | ");
  Serial.print(c);
  Serial.println(" |");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Meio Somador");
  Serial.println("| a | b || s | c |");
  Serial.println("------------------");
  for (a = 0; a <= 1; a++){
    for (b = 0; b <= 1; b++){
        meio_somador(a, b);
        mostra4();
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

6.2. Somador completo

6.2.1. Objetivos

1. Exame do circuito do somador completo,
2. Observar sua operação e obter sua tabela verdade,
3. Conhecer circuitos somadores.

6.2.2. Informação preliminar

Um somador completo é um circuito combinacional que executa a soma aritmética de três bits de entrada. O símbolo de um somador completo é mostrado na Figura 6.7. Consiste em três entradas e duas saídas. Duas das variáveis de entrada, denotadas por “A” e “B”, representam os dois bits significativos a serem adicionados. A terceira entrada, “Ci” (input carry), representa o transporte da posição anterior mais baixa significativa. As duas saídas são “S” (Soma) e “Co” (transporte de saída). Um somador completo pode ser obtido conectando dois meios somadores, como visto na Figura 6.6. A tabela verdade do somador completo é fornecida na Tabela 6.3.

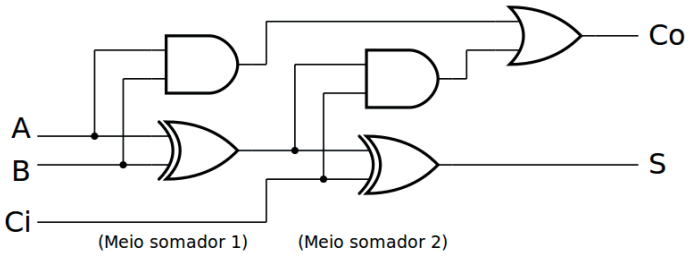


Figura 6.6 – O circuito completo somador.

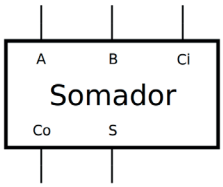


Figura 6.7 – O símbolo do somador completo.

Entradas			Saídas	
A	B	Ci	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabela 6.3 – A tabela verdade do somador completo.

6.2.3. Exame do somador completo

Esquemas:

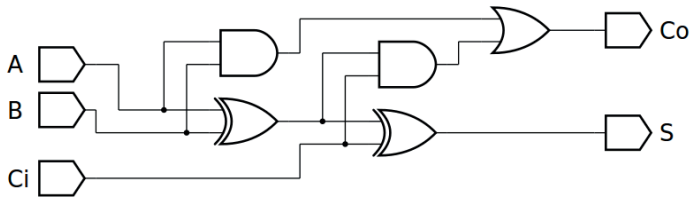


Figura 6.8 – Diagrama esquemático.

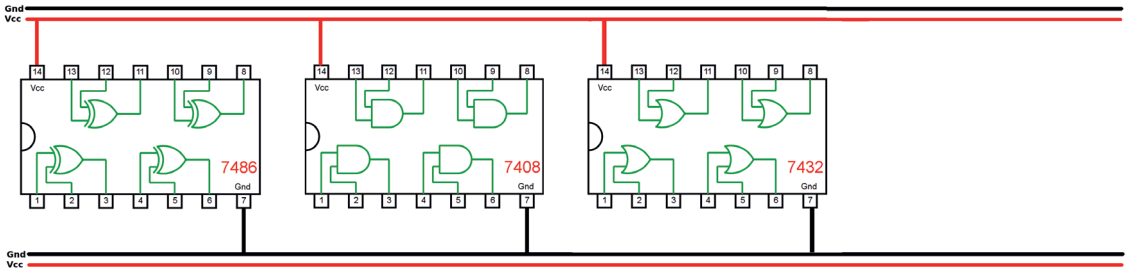


Figura 6.9 – Disposição dos componentes (TTL) para a placa de montagem.

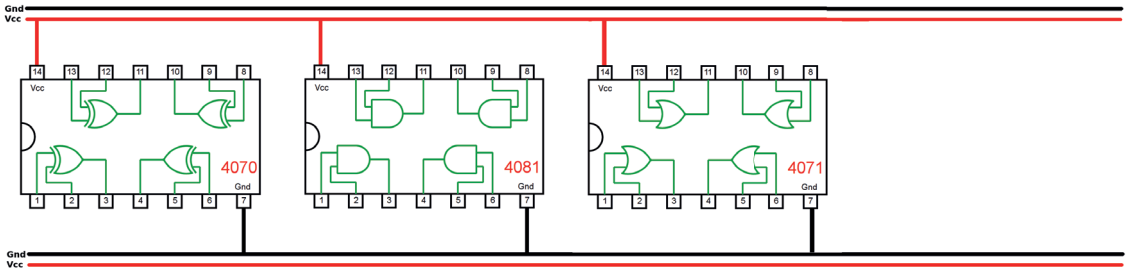


Figura 6.10 – Disposição dos componentes (CMOS) para a placa de montagem.

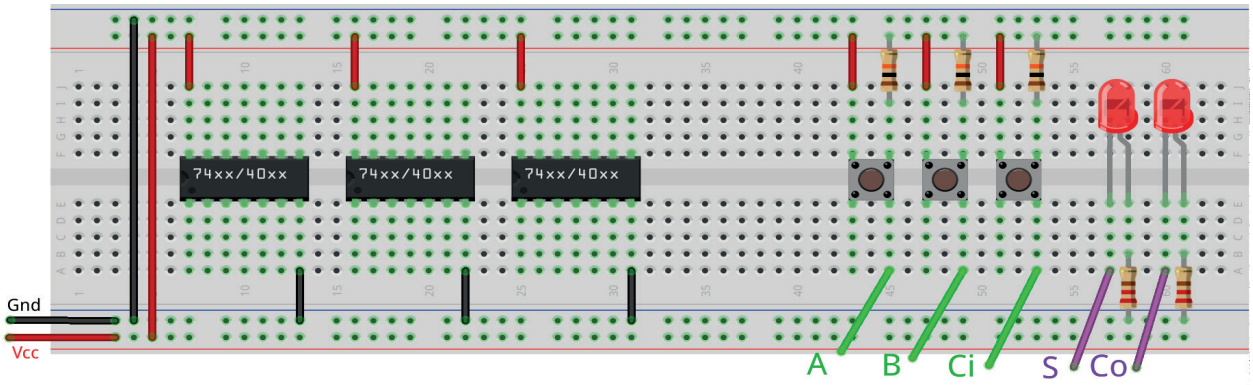


Figura 6.11 – Disposição dos componentes na placa de montagem.

Equação booleana:

S(A,B,Ci) = _____ Co(A,B,Ci) = _____

Versão para o Arduíno:

s = _____ co = _____

Procedimento:

1. Complete as ligações na figura 6.9 ou 6.10 de acordo com a figura 6.8.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “A”, “B” e “Ci”.
   c) Identifique os terminais dos CIs que receberão os fios roxos das saídas com as letras “S” e “Co”.
   d) Faça as conexões entre as portas lógicas dos CIs com linhas azuis.
2. Monte o circuito na placa de montagem, conforme figura 6.11.
   a) Conecte os fios de entrada (verdes) aos botões e de saída (roxos) aos LEDs.
   b) Conecte os fios de conexão (azuis) entre as portas lógicas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 6.4, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “S” e “Co” na tabela 6.4, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 6.8. Compare os resultados com a tabela 6.4.
6. Escreva a equação booleana do circuito da figura 6.8 e simule no programa para o Arduíno.

Tabelas de dados:

Entradas			Saídas	
A	B	Ci	S	Co
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Tabela 6.4

Programa para o Arduíno:

```
// Somador Completo

// valores de entrada
byte a; byte b; byte ci;

// valores intermediários
byte is; byte ic;

// valores de saída
byte s; byte c; byte co;

void meio_somador(byte a, byte b){
  s = (!a && b) || (a && !b); // função booleana!
  c = (a && b); // função booleana!
}

void somador_completo(byte a, byte b, byte ci){
  meio_somador(a, b);
  is = s;
  ic = c;
  meio_somador(is, ci);
  co = ic + c;
}

void mostra5(){
  Serial.print("| ");
  Serial.print(a);
  Serial.print(" | ");
  Serial.print(b);
  Serial.print(" | ");
  Serial.print(ci);
  Serial.print(" || ");
  Serial.print(s);
  Serial.print(" | ");
  Serial.print(co);
  Serial.println(" |");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Somador Completo");
  Serial.println("| a | b | ci || s | co |");
  Serial.println("------------------------");
  for (a = 0; a <= 1; a++){
    for (b = 0; b <= 1; b++){
      for (ci = 0; ci <= 1; ci++){
        somador_completo(a, b, ci);
        mostra5();
      }
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

6.3. Somador paralelo de 4 bits

6.3.1. Objetivos

1. Investigar o circuito somador paralelo de 4 bits,
2. Observar sua operação e obter sua tabela verdade,
3. Conhecer o somador paralelo de 4 bit 74LS83 (TTL) ou 4008 (CMOS).

6.3.2. Informação preliminar

A Figura 6.12 mostra um exemplo de um somador paralelo: um somador de 4 bits com bit de transporte acoplado. É composto por quatro somadores completos. Os bits de “B” são adicionados aos bits de “A” respeitando sua posição binária. Cada adição de bits produz uma soma (“S”) e um transporte (“Co”). O transporte de saída é então transmitida para o transporte de entrada (“Ci”) do próximo bit de ordem superior. O resultado final produz uma soma de quatro bits “S” (S4 S3 S2 S1) mais um bit de transporte (“Co”).

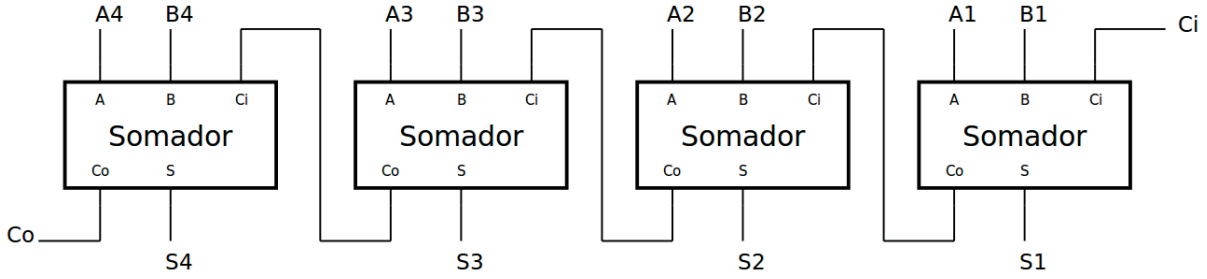


Figura 6.12 – Circuito somador paralelo de 4 bits.

O CI TTL 74LS83 é um somador paralelo. Ele adiciona dois números binários de 4 bits [A (A4 A3 A2 A1) e B (B4 B3 B2 B1)] e um transporte de um bit (“Ci”). A soma é expressa em forma binária como “Co” (transporte de saída), S4, S3, S2 e S1. É possível adicionar números BCD ou números em base 16 com o CI 74LS83 usando números de 4 bits. A tabela verdade do somador completo do CI 74LS83 é dada na Figura 6.13. O diagrama lógico do somador paralelo 74LS83 é mostrado na Figura 6.14. A Tabela 6.5 fornece alguns exemplos de operações para o somador paralelo 74LS83.

Número 1					Número 2				Ci		Saída				
A4	A3	A2	A1		B4	B3	B2	B1			Co	S4	S3	S2	S1
0	0	1	1	+	1	0	0	0	0	=	0	1	0	1	1
1	1	1	1	+	1	1	1	1	1	=	1	1	1	1	1
0	0	0	1	+	1	1	0	0	0	=	0	1	1	0	1
1	0	1	0	+	1	0	1	1	1	=	1	0	1	1	0
1	1	0	1	+	0	1	1	1	1	=	1	0	1	0	1
0	1	0	1	+	0	1	1	1	0	=	0	1	1	0	0

Tabela 6.5 – Alguns exemplos de operações para o somador paralelo 74LS83.

Inputs				Outputs					
				When C0 = L			When C0 = H		
				When C2 = L			When C2 = H		
A1 / A3	B1 / B3	A2 / A4	B2 / B4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

H = HIGH Level, L = LOW Level

Input conditions at A1, B1, A2, B2, and C0 are used to determine outputs ∑1 and ∑2 and the value of the internal carry C2. The values at C2, A3, B3, A4, and B4 are then used to determine outputs ∑3, ∑4, and C4.

Figura 6.13 – A tabela verdade do somador paralelo 74LS83.

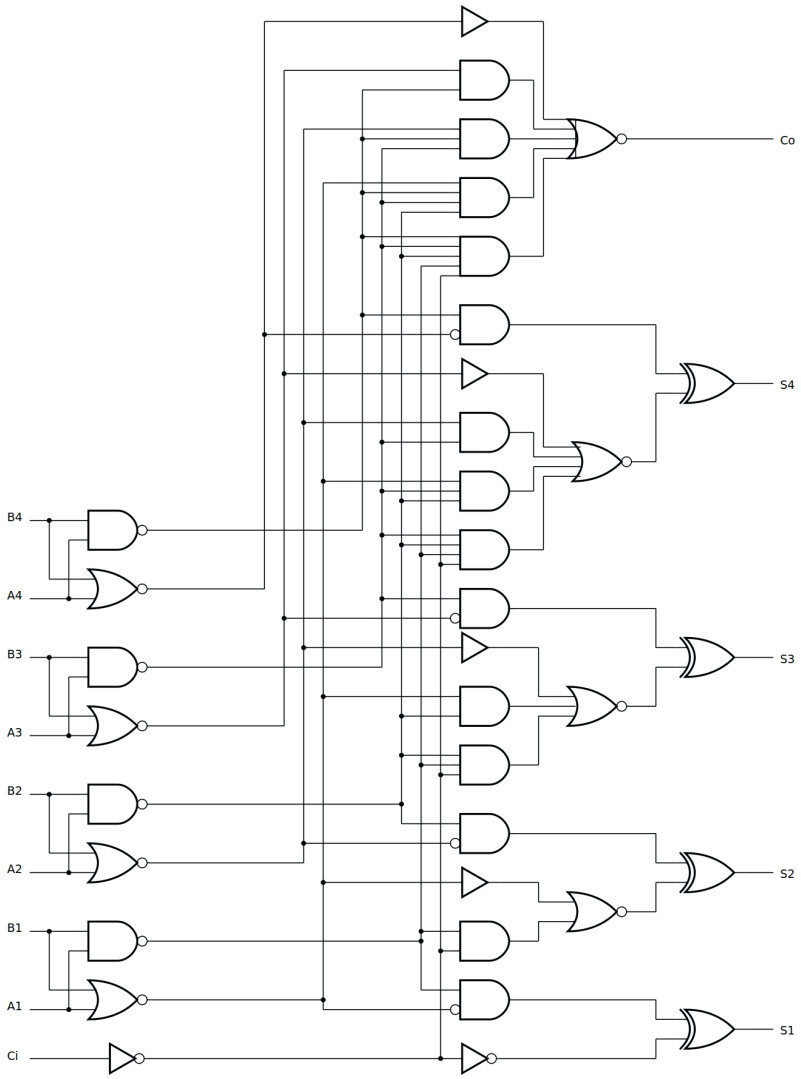


Figura 6.14 – Diagrama lógico do somador paralelo 74LS83.

6.3.3. Exame do somador paralelo de 4 bits

Esquemas:

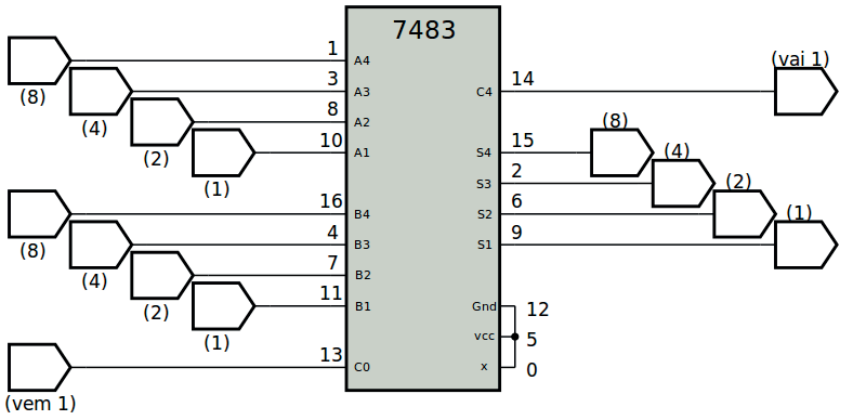


Figura 6.15 – Diagrama esquemático.

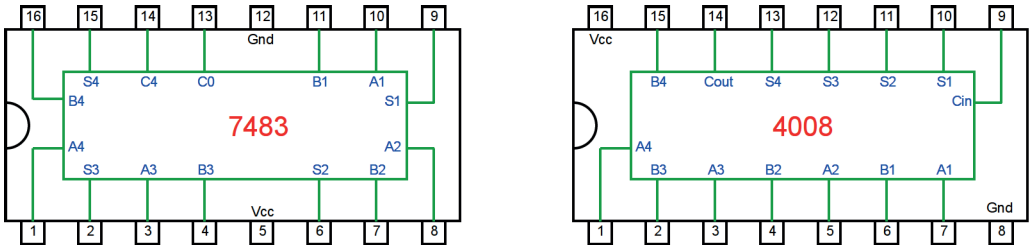


Figura 6.16 – Pinagem dos componentes (TTL e CMOS) para a placa de montagem.

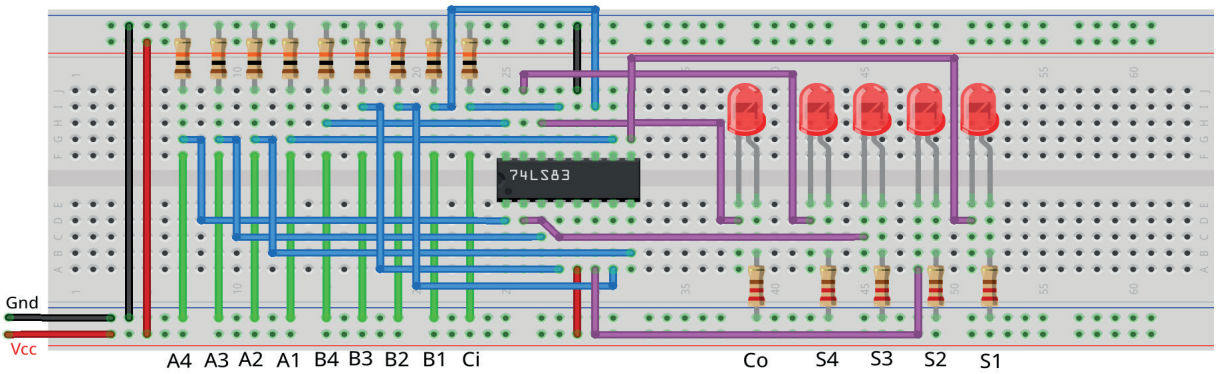


Figura 6.17 – Disposição dos componentes (TTL) na placa de montagem.

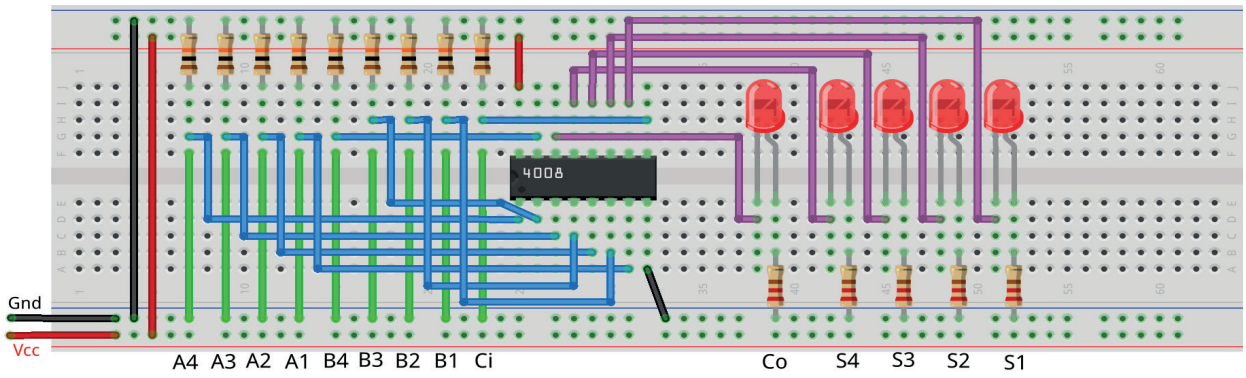


Figura 6.18 – Disposição dos componentes (CMOS) na placa de montagem.

Procedimento:

1. Monte o circuito da figura 6.15 na placa de montagem, conforme figura 6.17 (versão TTL) ou 6.18 (versão CMOS).
   a) Conecte os fios de entrada (verdes) à barra de terra (Gnd) e os fios de saída (roxos) aos LEDs.
   b) Conecte os fios de conexão (azuis) entre os resistores e as entradas do CI.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 6.6, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado ao “Gnd”.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado ao “Vcc”.
3. Para o preenchimento das colunas de saída na tabela 6.6, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 6.15. Compare os resultados com a tabela 6.6.
5. Simule o programa do somador para o Arduíno.

Tabelas de dados:

Número 1						Número 2							Saída					
dec	A4	A3	A2	A1		dec	B4	B3	B2	B1	Ci		Co	S4	S3	S2	S1	dec
10	1	0	1	0	+	11	1	0	1	1	0	=						
14	1	1	1	0	+	10	1	0	1	0	0	=						
12	1	1	0	0	+	4	0	1	0	0	0	=						
5	0	1	0	1	+	3	0	0	1	1	0	=						
4	0	1	0	0	+	5	0	1	0	1	0	=						
8	1	0	0	0	+	13	1	1	0	1	0	=						
15	1	1	1	1	+	9	1	0	0	1	1	=						
2	0	0	1	0	+	15	1	1	1	1	1	=						
13	1	1	0	1	+	2	0	0	1	0	1	=						
6	0	1	1	0	+	14	1	1	1	0	1	=						
7	0	1	1	1	+	7	0	1	1	1	1	=						
9	1	0	0	1	+	6	0	1	1	0	1	=						

Tabela 6.6

Programa para o Arduíno:

```
// Somador paralelo de 4 bits

// valores de entrada
byte q = 12;
byte x1[] = {10,14,12,5,4,8,15,2,13,6,7,9};
byte y1[] = {11,10,4,3,5,13,9,15,2,14,7,6};
byte ci1[] = {0,0,0,0,0,0,1,1,1,1,1,1};
byte x; byte y; byte a; byte b; byte ci;

// valores intermediários
byte is; byte ic; byte i;

// valores de saída
byte s; byte c; byte sx; byte co;
void meio_somador(byte a, byte b){
  s = (!a && b) || (a && !b); // função booleana!
  c = (a && b); // função booleana!
}

void somador_completo(byte a, byte b, byte ci){
  meio_somador(a, b);
  is = s;
  ic = c;
  meio_somador(is, ci);
  co = ic + c;
}

void printBits(byte myByte, byte mymask){
  for(byte mask = mymask; mask; mask >>= 1){
    if(mask & myByte)
      Serial.print("1 ");
    else
      Serial.print("0 ");
  }
}

void soma4bits(byte x, byte y, byte ci){
  sx = B00000000; // limpa vetor de saída
```

```
  // primeira soma A1 + B1 + Ci
  a = x & B00000001; // captura bit A1
  b = y & B00000001; // captura bit B1
  somador_completo(a,b,ci);
  sx = sx | s; // armazena no vetor de saída
  // segunda soma A2 + B2 + Co(1)
  a = x & B00000010; // captura bit A2
  b = y & B00000010; // captura bit B2
  somador_completo(a,b,co);
  s = s << 1; // desloca bit de saída
  sx = sx | s; // armazena no vetor de saída

  // terceira soma A3 + B3 + Co(2)
  a = x & B00000100; // captura bit A3
  b = y & B00000100; // captura bit B3
  somador_completo(a,b,co);
  s = s << 2; // desloca bit de saída
  sx = sx | s; // armazena no vetor de saída

  // quarta soma A5 + B5 + Co(3)
  a = x & B00001000; // captura bit A4
  b = y & B00001000; // captura bit B4
  somador_completo(a,b,co);
  s = s << 3; // desloca bit de saída
  sx = sx | s; // armazena no vetor de saída

  co = co << 4; // desloca bit de saída
  sx = sx | co; // armazena no vetor de saída
  co = co >> 4;
}

void mostra_soma4(){
  printBits(x, 0x08); Serial.print("+ ");
  printBits(y, 0x08); Serial.print("+ ");
  Serial.print(ci); Serial.print(" = ");
  Serial.print(co); Serial.print(" . ");
  printBits(sx, 0x08); Serial.print("-- ");
  Serial.print(x); Serial.print(" + ");
  Serial.print(y); Serial.print(" + ");
  Serial.print(ci); Serial.print(" = ");
  Serial.print(sx); Serial.println();
}
```

```
void setup(){
  Serial.begin(9600);
  Serial.println("Somador Paralelo de 4 bits");
  Serial.println("A4 A3 A2 A1 + B4 B3 B2 B1 + Ci = Co. S4 S3 S2 S1 --
  Decimal");
Serial.println("-------------------------------------------------------
---");

  for (i = 0; i < q; i++){
    x = x1[i]; y = y1[i]; ci = ci1[i];
    soma4bits(x, y, ci);
    mostra_soma4();
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

6.4. Subtratores binários

6.4.1. Objetivos

1. Investigar os circuitos subtratores binários,
2. Observar seu funcionamento e obter suas tabelas verdade,
3. Conhecer os CIs de circuitos subtratores binários.

6.4.2. Informação preliminar

Um meio-subtrator é um circuito combinacional que subtrai dois bits (x - minuendo, y - subtraendo) e produz sua diferença (D - diferença). Também possui uma saída para especificar se um 1 foi emprestado (B - emprestado). As regras a seguir são aplicáveis para subtratores.

x y B D

0 - 0 = 0

0 - 1 = 1 1 (Emprestado = 1, Diferença = 1)

1 - 0 = 1 (Diferença = 1)

1 - 1 = 0

O circuito do meio subtrator é mostrado na Figura 6.19, e sua tabela verdade é fornecida na Tabela 6.7.

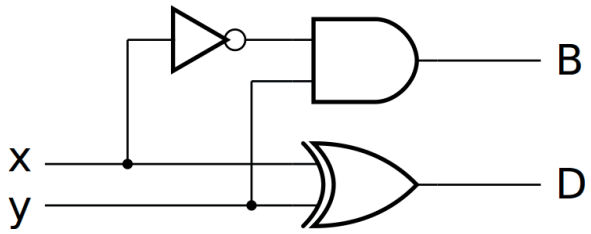


Figura 6.19 – O circuito meio subtrator.

Entradas		Saídas	
x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Tabela 6.7 – A tabela verdade do meio subtrator.

6.4.3. Exame do meio subtrator

Esquemas:

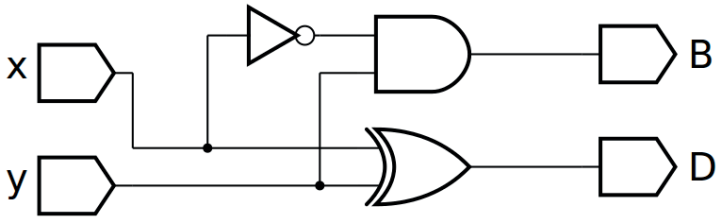


Figura 6.20 – Diagrama esquemático.

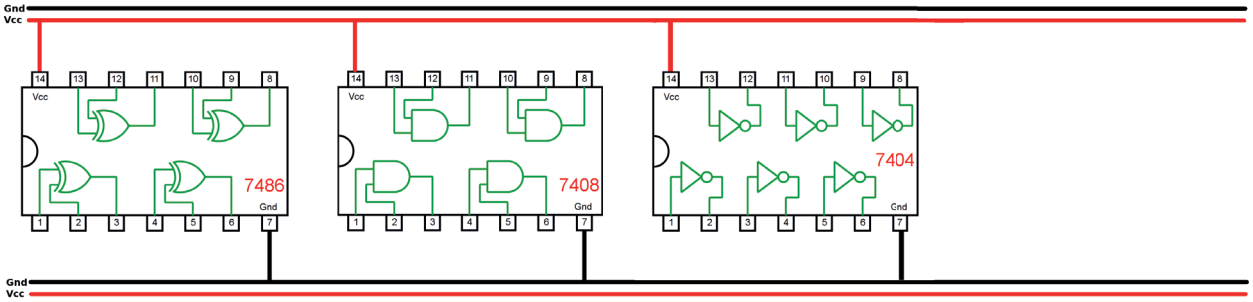


Figura 6.21 – Disposição dos componentes (TTL) para a placa de montagem.

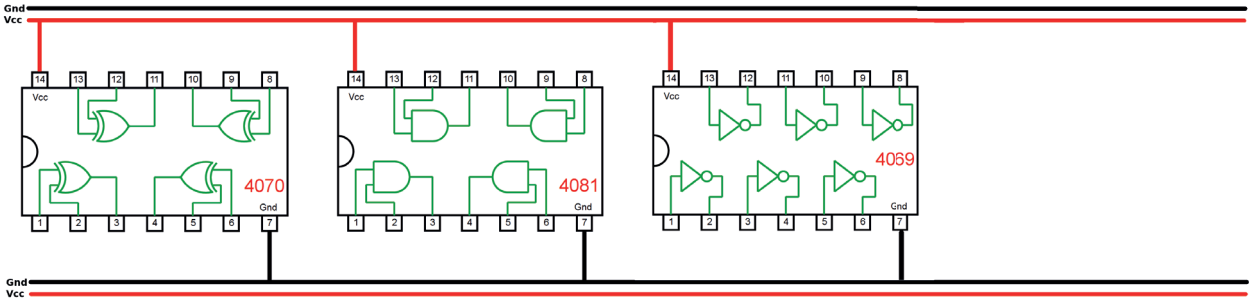


Figura 6.22 – Disposição dos componentes (CMOS) para a placa de montagem.

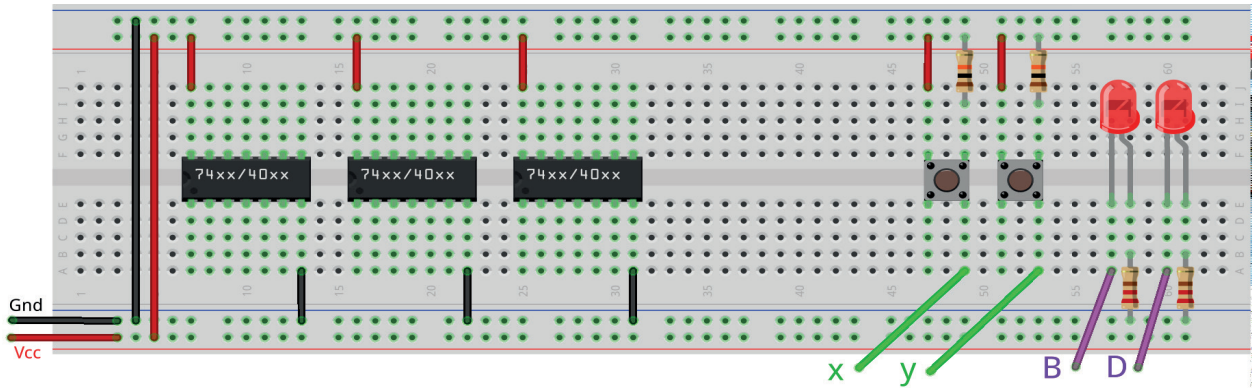


Figura 6.23 – Disposição dos componentes na placa de montagem.

Equação booleana:

D(x,y) = _____ B(x,y) = _____

Versão para o Arduíno:

D = _____ B = _____

Procedimento:

1. Complete as ligações na figura 6.21 ou 6.22 de acordo com a figura 6.20.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “x” e “y”.
   c) Identifique os terminais dos CIs que receberão os fios roxos das saídas com as letras “D” e “B”.
   d) Faça as conexões entre as portas lógicas dos CIs com fios azuis.
2. Monte o circuito na placa de montagem, conforme figura 6.23.
   a) Conecte os fios de entrada (verdes) aos botões e de saída (roxos) aos LEDs.
   b) Conecte os fios de conexão (azuis) entre as portas lógicas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 6.8, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “D” e “B” na tabela 6.2, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 6.20. Compare os resultados com a tabela 6.8.
6. Escreva a equação booleana do circuito da figura 6.20 e simule no programa para o Arduíno.

Tabelas de dados:

Entradas		Saídas	
x	y	B	D
0	0		
0	1		
1	0		
1	1		

Tabela 6.8

Programa para o Arduíno:

```
// Meio Subtrator

byte x; byte y; byte D; byte B;

void meio_subtrator(byte x, byte y){
  D = (!x && y) || (x && !y); // função booleana!
  B = (!x && y); // função booleana!
}

void mostra4(){
  Serial.print("| ");
  Serial.print(x);
  Serial.print(" | ");
  Serial.print(y);
  Serial.print(" || ");
  Serial.print(B);
  Serial.print(" | ");
  Serial.print(D);
  Serial.println(" |");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Meio Subtrator");
  Serial.println("| x | y || B | D |");
  Serial.println("------------------");
  for (x = 0; x <= 1; x++){
    for (y = 0; y <= 1; y++){
      meio_subtrator(x, y);
      mostra4();
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

6.5. Subtrator completo

6.5.1. Objetivos

1. Exame do circuito do subtrator completo,
2. Observar sua operação e obter sua tabela verdade,
3. Conhecer os circuitos subtratores.

6.5.2. Informação preliminar

Um subtrator completo é um circuito combinacional que realiza subtração entre dois bits levando em conta que um 1 pode ter sido emprestado por um estágio significativo inferior. As três entradas “x”, “y” e “z” indicam o minuendo, o subtraendo e o empréstimo anterior, respectivamente. As duas saídas D e B representam a diferença e a saída emprestada, respectivamente. Um subtrator completo pode ser obtido conectando dois meios subtratores, como visto na Figura 6.24. A tabela verdade do subtrator completo é fornecida na Tabela 6.9.

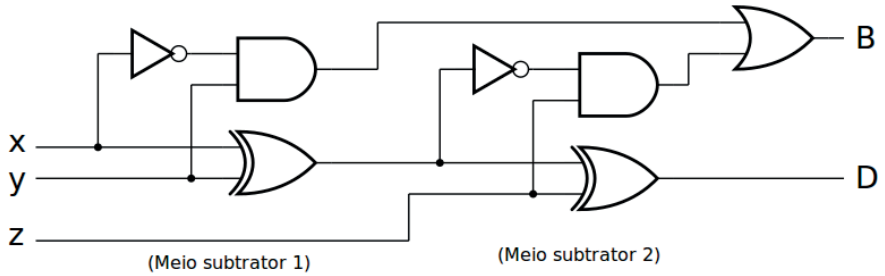


Figura 6.24 – O circuito subtrator completo.

Entradas			Saídas	
x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Tabela 6.9 – A tabela verdade do subtrator completo.

6.5.3. Exame do subtrator completo

Esquemas:

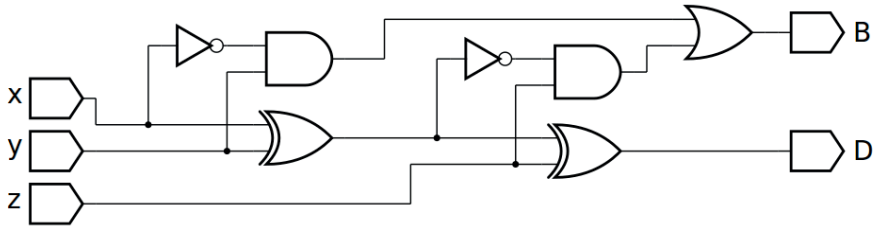


Figura 6.25 – Diagrama esquemático.

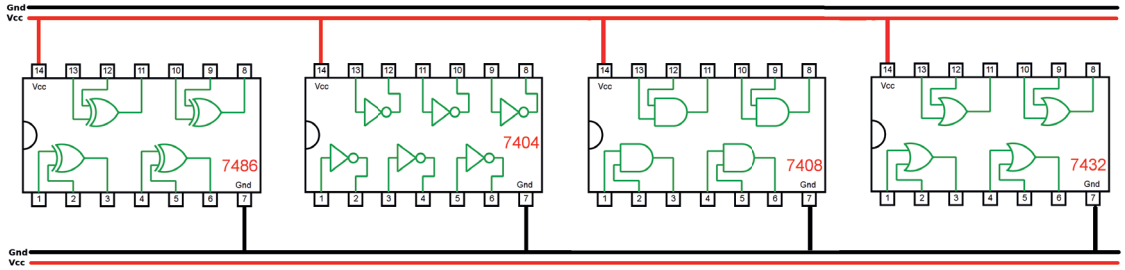


Figura 6.26 – Disposição dos componentes (TTL) para a placa de montagem.

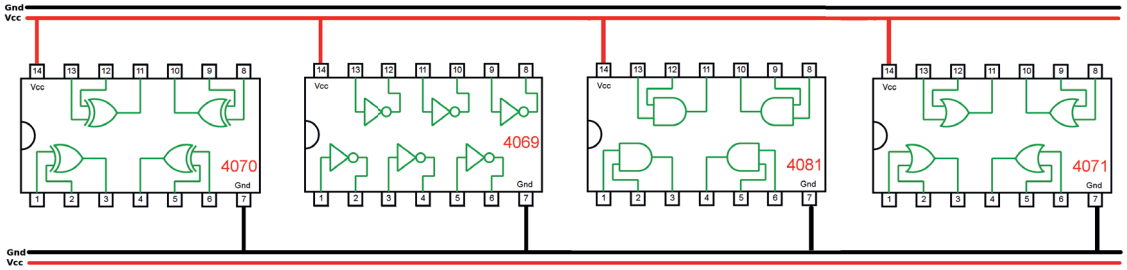


Figura 6.27 – Disposição dos componentes (CMOS) para a placa de montagem.

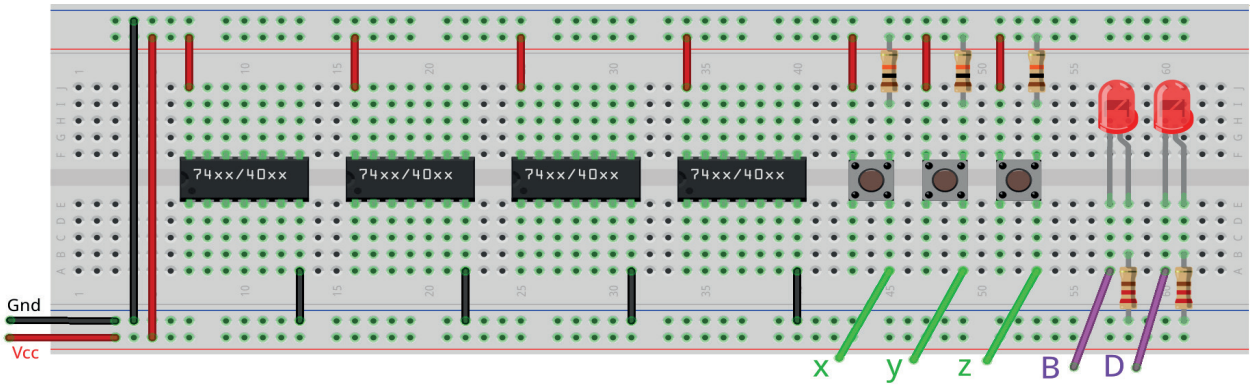


Figura 6.28 – Disposição dos componentes na placa de montagem.

Equação booleana:

B(x,y,z) = _____ D(x,y,z) = _____

Versão para o Arduíno:

B = _____ D = _____

Procedimento:

1. Complete as ligações na figura 6.26 ou 6.27 de acordo com a figura 6.25.
   a) Coloque as ligações de Vcc (vermelho) e Gnd (preto) em cada CI.
   b) Identifique os terminais dos CI que receberão os fios verdes das entradas com as letras correspondentes, “x”, “y” e “z”.
   c) Identifique os terminais dos CIs que receberão os fios roxos das saídas com as letras “B” e “D”.
   d) Faça as conexões entre as portas lógicas dos CIs com fios azuis.
2. Monte o circuito na placa de montagem, conforme figura 6.28.
   a) Conecte os fios de entrada (verdes) aos botões e de saída (roxos) aos LEDs.
   b) Conecte os fios de conexão (azuis) entre as portas lógicas.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 6.10, use os botões como segue:
   a) Se o valor da variável for “0” o botão correspondente NÃO DEVE ser pressionado.
   b) Se o valor da variável for “1” o botão corresponde DEVE ser pressionado.
4. Para o preenchimento da coluna “B” e “D” na tabela 6.10, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 6.25. Compare os resultados com a tabela 6.10.
6. Escreva a equação booleana do circuito da figura 6.25 e simule no programa para o Arduíno.

Tabelas de dados:

Entradas			Saídas	
x	y	z	B	D
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Tabela 6.10

Programa para o Arduíno:

```
// Subtrator Completo

// valores de entrada
byte x; byte y; byte z;

// valores intermediários
byte id; byte ib;

// valores de saída
byte D; byte B;

void meio_subtrator(byte x, byte y){
  D = (!x && y) || (x && !y); // função booleana!
  B = (!x && y); // função booleana!
}

void subtrator_completo(byte x, byte y, byte z){
  meio_subtrator(x, y);
  id = D;
  ib = B;
  meio_subtrator(id, z);
  B = ib + B;
}

void mostra5(){
  Serial.print("| ");
  Serial.print(x);
  Serial.print(" | ");
  Serial.print(y);
  Serial.print(" | ");
  Serial.print(z);
  Serial.print(" || ");
  Serial.print(B);
  Serial.print(" | ");
  Serial.print(D);
  Serial.println(" |");
}

void setup(){
```

```
  Serial.begin(9600);
  Serial.println("Subtrator Completo");
  Serial.println("| x | y | z || B | D |");
  Serial.println("------------------------");
  for (x = 0; x <= 1; x++){
    for (y = 0; y <= 1; y++){
      for (z = 0; z <= 1; z++){
        subtrator_completo(x, y, z);
        mostra5();
      }
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

6.6. Aplicações especiais

6.6.1. Exame do somador / subtrator paralelo de 4 bits

Esquemas:

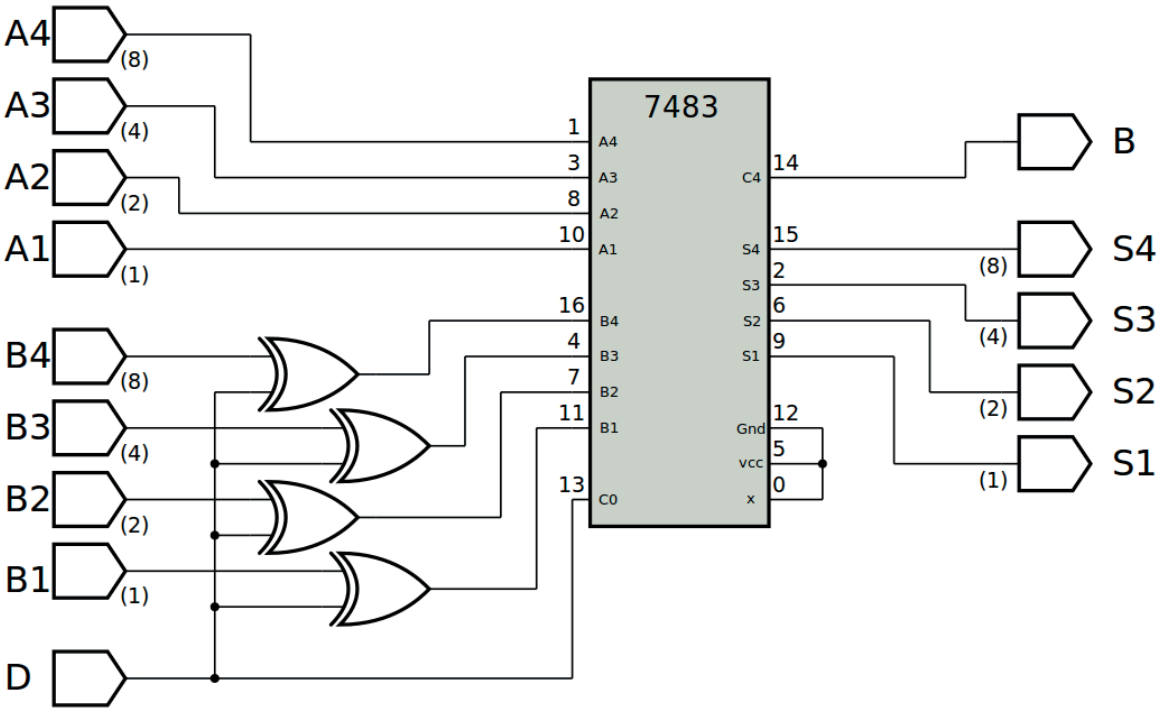


Figura 6.29 – Diagrama esquemático.

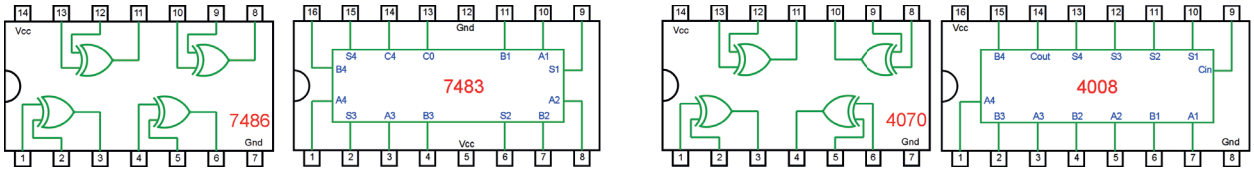


Figura 6.30 – Pinagem dos componentes (TTL e CMOS) para a placa de montagem.

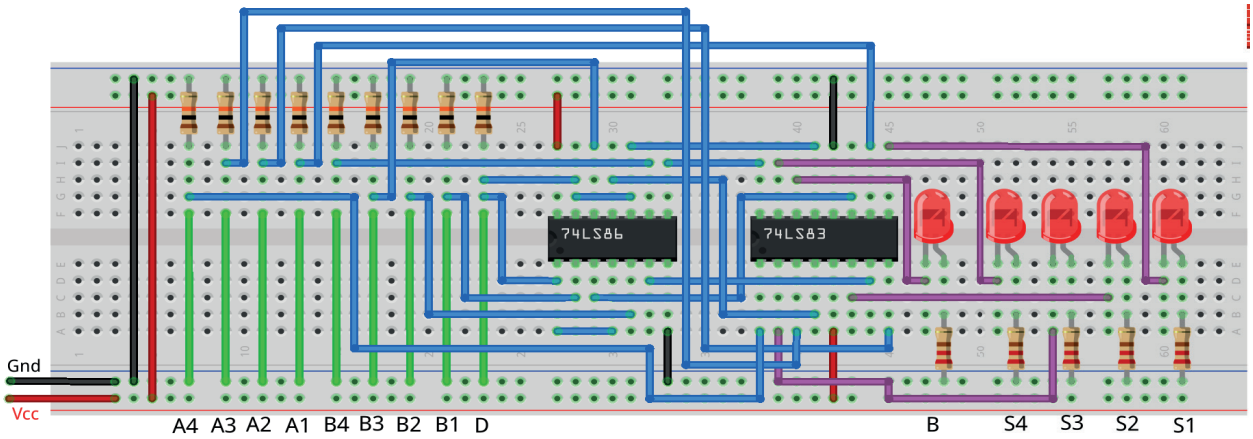


Figura 6.31 – Disposição dos componentes (TTL) na placa de montagem.

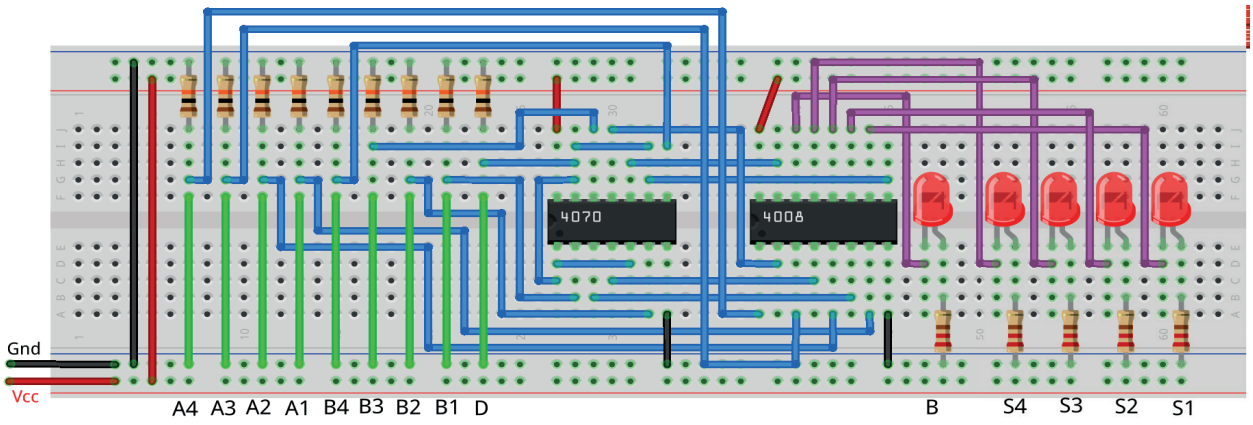


Figura 6.32 – Disposição dos componentes (CMOS) na placa de montagem.

Descrição do circuito:

A Figura 6.29 mostra um circuito somador / subtrator completo de 4 bits. Existem duas entradas de 4 bits A (A4 A3 A2 A1) e B (B4 B3 B2 B1). As saídas são B, S4, S3, S2, S1. Neste circuito, se D = 0 então as duas entradas de 4 bits A e B são adicionadas: A + B. Neste caso, as portas OU-Ex atuam como "buffers" não-inversores, e o transporte de entrada (Cin) para o somador é 0. Portanto, o somador calcula uma soma de quatro bits com transporte de saída: (B, S4, S3, A2, S1) = (A4, A3, A2, A1) + (B4, B3, B2, B1). Se D = 1, a subtração é aplicada com base no complemento de dois da seguinte forma: A + B '+ 1. Neste caso, as portas OU-Ex atuam como "buffers" de inversão, e o transporte de entrada (Cin) para o somador é 1. Portanto, o circuito calcula uma subtração de quatro bits com base no complemento de dois: (B, S4, S3, A2, S1) = (A4, A3, A2, A1) - (B4, B3, B2, B1)

Procedimento:

1. Monte o circuito da figura 6.29 na placa de montagem, conforme figura 6.31 (versão TTL) ou 6.32 (versão CMOS).
   a) Conecte os fios de entrada (verdes) à barra de terra (Gnd) e os fios de saída (roxos) aos LEDs.
   b) Conecte os fios de conexão (azuis) entre os resistores e as entradas do CI.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 6.11, use os fios verdes como segue:
   a) Se o valor da variável for "0" o fio verde correspondente deve ser conectado ao "Gnd".
   b) Se o valor da variável for "1" o fio verde correspondente deve ser conectado ao "Vcc".
3. Para o preenchimento das colunas de saída na tabela 6.11, considere:
   a) Se o LED estiver apagado então preencher com "0".
   b) Se o LED estiver aceso então preencher com "1".
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 6.29. Compare os resultados com a tabela 6.11.
5. Simule o programa do somador/subtrator para o Arduíno.

Tabelas de dados:

Número A					D		Número B						Saída					
dec	A4	A3	A2	A1			dec	B4	B3	B2	B1		B	S4	S3	S2	S1	dec
10	1	0	1	0	0	+	11	1	0	1	1	=						
14	1	1	1	0	0	+	10	1	0	1	0	=						
12	1	1	0	0	0	+	4	0	1	0	0	=						
5	0	1	0	1	0	+	3	0	0	1	1	=						
4	0	1	0	0	0	+	5	0	1	0	1	=						
13	1	1	0	1	0	+	13	1	1	0	1	=						
6	0	1	1	0	0	+	9	1	0	0	1	=						
2	0	0	1	0	0	+	8	1	0	0	0	=						
1	0	0	0	1	0	+	2	0	0	1	0	=						
7	0	1	1	1	0	+	1	0	0	0	1	=						
15	1	1	1	1	0	+	6	0	1	1	0	=						
9	1	0	0	1	0	+	15	1	1	1	1	=						
10	1	0	1	0	1	-	11	1	0	1	1	=						
14	1	1	1	0	1	-	10	1	0	1	0	=						
12	1	1	0	0	1	-	4	0	1	0	0	=						
5	0	1	0	1	1	-	3	0	0	1	1	=						
4	1	0	0	0	1	-	5	0	1	0	1	=						
13	1	1	0	1	1	-	13	1	1	0	1	=						
6	0	1	1	0	1	-	9	1	0	0	1	=						
2	0	0	1	0	1	-	8	1	0	0	1	=						
1	0	0	0	1	1	-	2	0	0	1	0	=						
7	0	1	1	1	1	-	1	0	0	0	1	=						
15	1	1	1	1	1	-	6	0	1	1	0	=						
9	1	0	0	1	1	-	15	1	1	1	1	=						

Tabela 6.11

Programa para o Arduíno:

```
// Somador / Subtrator paralelo de 4 bits

// valores de entrada
byte q = 24;
byte Ax[] = {10,14,12,5,4,13,6,2,1,7,15,9,10,14,12,5,4,13,6,2,1,7,15,9
};
byte Bx[] = {11,10,4,3,5,13,9,8,2,1,6,15,11,10,4,3,5,13,9,8,2,1,6,15};
byte Dx[] = {0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1};
byte A; byte B; byte D;
byte x; byte y; byte a; byte b; byte ci;

// valores intermediários
byte Bc; byte is; byte ic; byte i;

// valores de saída
byte s; byte c; byte sx; byte co;
void meio_somador(byte a, byte b){
  s = (!a && b) || (a && !b); // função booleana!
  c = (a && b); // função booleana!
}

void somador_completo(byte a, byte b, byte ci){
  meio_somador(a, b);
  is = s;
  ic = c;
  meio_somador(is, ci);
  co = ic + c;
}

void soma4bits(byte x, byte y, byte ci){
  sx = B00000000; // limpa vetor de saída

  // primeira soma A1 + B1 + Ci
  a = x & B00000001; // captura bit A1
  b = y & B00000001; // captura bit B1
  somador_completo(a,b,ci);
  sx = sx | s; // armazena no vetor de saída

  // segunda soma A2 + B2 + Co(1)
  a = x & B00000010; // captura bit A2
```

```
  b = y & B00000010; // captura bit B2
  somador_completo(a,b,co);
  s = s << 1; // desloca bit de saída
  sx = sx | s; // armazena no vetor de saída

  // terceira soma A3 + B3 + Co(2)
  a = x & B00000100; // captura bit A3
  b = y & B00000100; // captura bit B3
  somador_completo(a,b,co);
  s = s << 2; // desloca bit de saída
  sx = sx | s; // armazena no vetor de saída
  // quarta soma A5 + B5 + Co(3)
  a = x & B00001000; // captura bit A4
  b = y & B00001000; // captura bit B4
  somador_completo(a,b,co);
  s = s << 3; // desloca bit de saída
  sx = sx | s; // armazena no vetor de saída
  co = co << 4; // desloca bit de saída
  sx = sx | co; // armazena no vetor de saída
  co = co >> 4;
}

void printBits(byte myByte, byte mymask){
  for(byte mask = mymask; mask; mask >>= 1){
    if(mask & myByte)
      Serial.print("1 ");
        else
      Serial.print("0 ");
  }
}

void complemento(byte D, byte B){
  if (D){
    Bc = ~B;
    Bc = Bc & 0x0F;
  }else{
    Bc = B;
  }
}

void mostra_soma4(){
```

```
  printBits(A, 0x08);
  if (D) Serial.print(“. 1 - “);
  else Serial.print(“. 0 + “);
  printBits(B, 0x08);
  Serial.print(“= “);
  Serial.print(co);
  Serial.print(“ . “);
  printBits(sx, 0x08);
  Serial.print(“-- “);
  Serial.print(A);
  if (D) Serial.print(“ - “);
  else Serial.print(“ + “);
  Serial.print(B);
  Serial.print(“ = “);
  Serial.print(sx);
  Serial.println();
}

void setup(){
  Serial.begin(9600);
  Serial.println(“Somador Paralelo de 4 bits”);
  Serial.println(“A4 A3 A2 A1 . D +/- B4 B3 B2 B1 = B . S4 S3 S2 S1
  -- Decimal”);

  Serial.println(“-------------------------------------------------------
  -------”);

  for (i = 0; i < q; i++){
    A = Ax[i]; B = Bx[i]; D = Dx[i];
    complemento(D, B);
    soma4bits(A, Bc, D);
    mostra_soma4();
  }
} // fim do ‘setup’

void loop(){
  // nada a fazer aqui!
} // fim do ‘loop’
```

6.6.3. Exame do somador BCD de 4 bits

Esquemas:

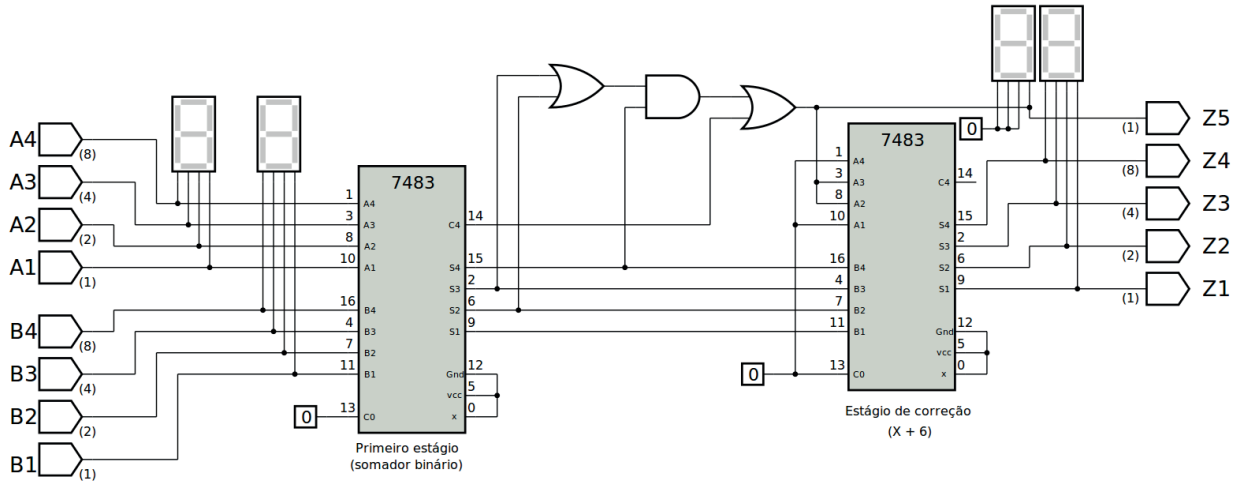


Figura 6.33 – Diagrama esquemático.

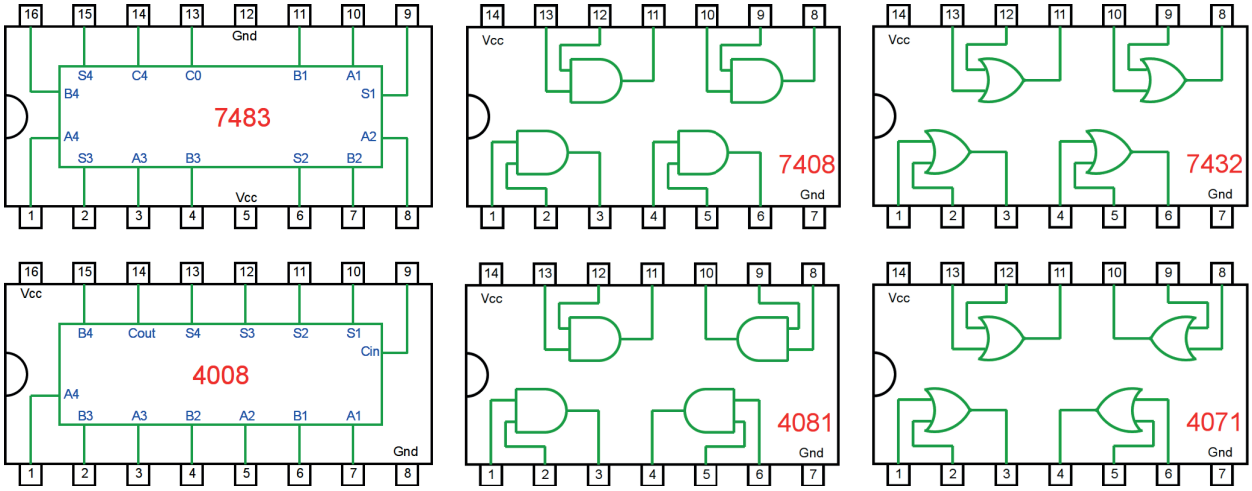


Figura 6.34 – Pinagem dos componentes (TTL e CMOS) para a placa de montagem.

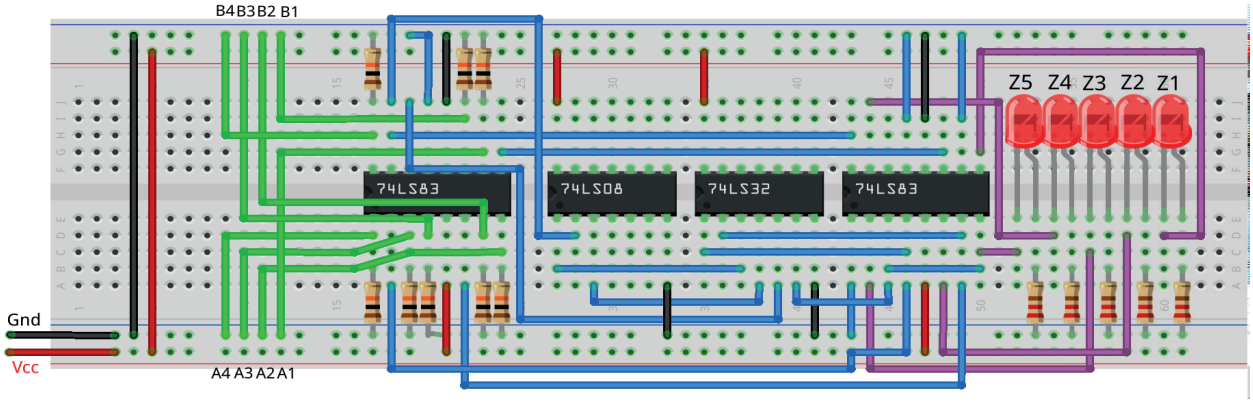


Figura 6.35 – Disposição dos componentes (TTL) na placa de montagem.

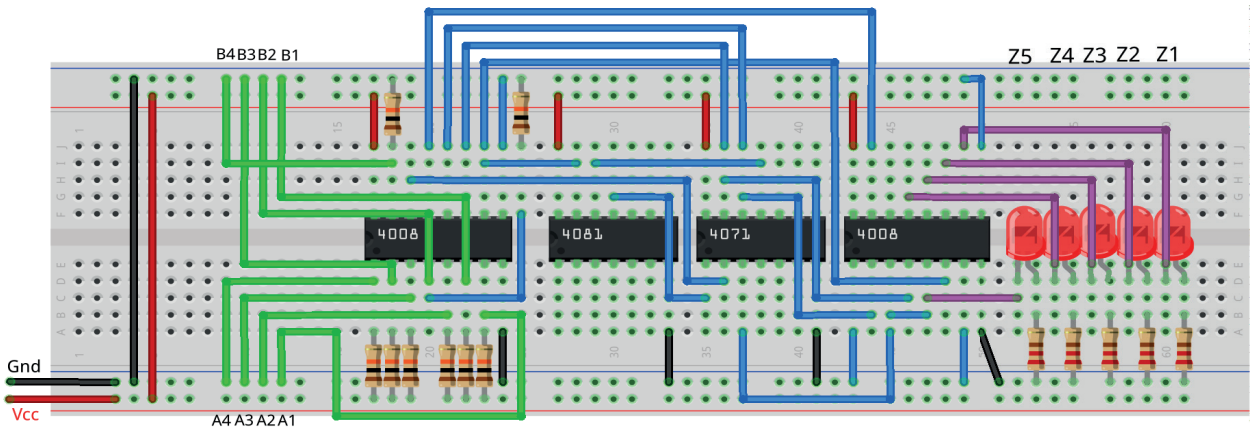


Figura 6.36 – Disposição dos componentes (CMOS) na placa de montagem.

Descrição do circuito:

A Figura 6.33 mostra um circuito somador de 4 bits BCD (Binary Coded Decimal). Neste circuito, são adicionadas as entradas BCD A (A4 A3 A2 A1) e B (B4 B3 B2 B1). Como as entradas A e B são definidas como BCD, espera-se que sejam aplicadas como um dos seguintes valores: 0, 1, 2, ..., 9, porque os valores restantes, a saber 10, 11, 12, 13, 14, 15 (valores hexadecimais A, B, C, D, E e F) são indefinidos para a aritmética BCD. Naturalmente, seria fácil projetar um circuito especial para a aritmética decimal com código binário. No entanto, isso raramente é feito. O circuito mostrado aqui baseia-se no mesmo truque que é frequentemente usado em microprocessadores para instruções aritméticas BCD. Por exemplo, muitos microprocessadores, incluindo as famílias Intel 808x e Motorola 68xx, fornecem uma instrução especial de acumulador de ajuste decimal (DAA). Uma adição de BCD é então executada em duas etapas, ou seja, uma adição padrão seguida pela instrução DAA. A operação básica executada pelo DAA é adicionar um valor constante de 6 para cada dígito BCD que transbordou durante a primeira adição. Apenas muito pouca lógica é necessária para implementar esta operação. Para tornar esse comportamento explícito, o circuito mostrado na Figura 6.33 usa dois estágios de somadores binários, cada um construído com um único somador de 4 bits 74LS83 ou 4008. O primeiro estágio consiste apenas no somador binário. O segundo estágio usa algumas portas para verificar um estouro decimal, ou seja, valores de saída maiores que 9. Se um estouro é detectado, o segundo somador é conectado para adicionar o valor 6 (0110) à saída do primeiro somador - que é equivalente a uma subtração de 10, desfazendo assim o estouro do primeiro estágio. O valor de saída de 4 bits resultante (Z4 Z3 Z2 Z1) e o transporte de 1 bit (Z5) são a soma correta na aritmética do BCD.

Procedimento:

1. Monte o circuito da figura 6.33 na placa de montagem, conforme figura 6.34 (versão TTL) ou 6.35 (versão CMOS).
   a) Conecte os fios de entrada (verdes) à barra de terra (Gnd) e os fios de saída (roxos) aos LEDs.
   b) Conecte os fios de conexão (azuis) entre os resistores e as entradas do CI.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 6.12, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado ao “Gnd”.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado ao “Vcc”.
3. Para o preenchimento das colunas de saída na tabela 6.1, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 6.33. Compare os resultados com a tabela 6.12.
5. Simule o programa do somador/subtrator para o Arduíno.

Tabelas de dados:

Número A						Número B						Saída					
dec	A4	A3	A2	A1		dec	B4	B3	B2	B1		dec	Z5	Z4	Z3	Z2	Z1
5	0	1	0	1	+	4	0	1	0	0	=						
9	1	0	0	1	+	9	1	0	0	1	=						
3	0	0	1	1	+	7	0	1	1	1	=						
6	0	1	1	0	+	2	0	0	1	0	=						
8	1	0	0	0	+	2	0	0	1	0	=						
6	0	1	1	0	+	6	0	1	1	0	=						
2	0	0	1	0	+	8	1	0	0	0	=						
1	0	0	0	1	+	7	0	1	1	1	=						
9	1	0	0	1	+	7	0	1	1	1	=						
10	1	0	1	0	+	9	1	0	0	1	=						
11	1	0	1	1	+	6	0	1	1	0	=						
12	1	1	0	0	+	7	0	1	1	1	=						
9	1	0	0	1	+	10	1	0	1	0	=						
3	0	0	1	1	+	11	1	0	1	1	=						
6	0	1	1	0	+	12	1	1	0	0	=						
10	1	0	1	0	+	13	1	1	0	1	=						
11	1	0	1	1	+	14	1	1	1	0	=						
12	1	1	0	0	+	15	1	1	1	1	=						
13	1	1	0	1	+	10	1	0	1	0	=						
14	1	1	1	0	+	11	1	0	1	1	=						
15	1	1	1	1	+	12	1	1	0	0	=						

Tabela 6.12

Programa para o Arduíno:

```
// Somador BCD de 4 bits

// valores de entrada
byte q = 21;
byte Ax[] = {5,9,3,6,8,6,2,1,9,10,11,12,9,3,6,10,11,12,13,14,15};
byte Bx[] = {4,9,7,2,2,6,8,7,7,9,6,7,10,11,12,13,14,15,10,11,12};
byte x; byte y; byte a; byte b; byte ci;

// valores intermediários
byte is; byte ic; byte i; byte est1; byte est2; byte est3;
byte z2; byte z3; byte z4;

// valores de saída
byte s; byte c; byte sx; byte co;
void meio_somador(byte a, byte b){
  s = (!a && b) || (a && !b); // função booleana!
  c = (a && b); // função booleana!
}

void somador_completo(byte a, byte b, byte ci){
  meio_somador(a, b);
  is = s;
  ic = c;
  meio_somador(is, ci);
  co = ic + c;
}

void printBits(byte myByte, byte mymask){
  for(byte mask = mymask; mask; mask >>= 1){
    if(mask & myByte)
      Serial.print("1 ");
    else
      Serial.print("0 ");
  }
  }

  void soma4bits(byte x, byte y, byte ci){
  sx = B00000000; // limpa vetor de saída
```

```
  // primeira soma A1 + B1 + Ci
  a = x & B00000001; // captura bit A1
  b = y & B00000001; // captura bit B1
  somador_completo(a,b,ci);
  sx = sx | s; // armazena no vetor de saída

  // segunda soma A2 + B2 + Co(1)
  a = x & B00000010; // captura bit A2
  b = y & B00000010; // captura bit B2
  somador_completo(a,b,co);
  s = s << 1; // desloca bit de saída
  sx = sx | s; // armazena no vetor de saída

  // terceira soma A3 + B3 + Co(2)
  a = x & B00000100; // captura bit A3
  b = y & B00000100; // captura bit B3
  somador_completo(a,b,co);
  s = s << 2; // desloca bit de saída
  sx = sx | s; // armazena no vetor de saída

  // quarta soma A5 + B5 + Co(3)
  a = x & B00001000; // captura bit A4
  b = y & B00001000; // captura bit B4
  somador_completo(a,b,co);
  s = s << 3; // desloca bit de saída
  sx = sx | s; // armazena no vetor de saída

  // co = co << 4; // desloca bit de saída
  // sx = sx | co; // armazena no vetor de saída
  // co = co >> 4;
  }

void estouro(){
  z2 = sx & B00000010;
  z3 = sx & B00000100;
  z4 = sx & B00001000;
  est1 = z2 || z3;
  est2 = est1 && z4;
  est3 = est2 || co;
}
```

```
void mostra_soma4(){
  printBits(x, 0x08); Serial.print("+ ");
  printBits(y, 0x08); Serial.print("= ");
  Serial.print(est3); Serial.print(" ");
  printBits(sx, 0x08); Serial.print("-- ");
  Serial.print(x); Serial.print(" + ");
  Serial.print(y); Serial.print(" = ");
  Serial.print(est3); Serial.print(" ");
  Serial.print(sx); Serial.println();
}

void setup(){
  Serial.begin(9600);
  Serial.println("Somador BCD de 4 bits");
  Serial.println("A4 A3 A2 A1 + B4 B3 B2 B1 = Z5 Z4 Z3 Z2 Z1 -- Deci-
  mal");
  Serial.println("-----------------------------------------------------");
  for (i = 0; i < q; i++){
    x = Ax[i]; y = Bx[i];
    soma4bits(x,y,0);
    estouro();
    if (est3)
      soma4bits(6,sx,0);
    else
      soma4bits(0,sx,0);
    mostra_soma4();
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

6.7. Comparadores binários

6.7.1. Objetivos

1. Investigar os circuitos comparadores binários,
2. Observar seu funcionamento e obter suas tabelas verdade,
3. Conhecer os CIs de circuitos comparadores binários.

6.7.2. Informação preliminar

Um comparador digital é um circuito lógico que compara dois números binários. O comparador pode ser usado para determinar se um determinado número é menor que, igual ou maior que outro número. Um comparador de 1 bit tem duas entradas (*A*, *B*) e três saídas ($O_{A<B}$, $O_{A=B}$, $O_{A>B}$). As equações lógicas de saída podem ser escritas como:

$$O_{(A>B)} = A \cdot B'$$

$$O_{(A=B)} = (A \cdot B) + (A' \cdot B') = (A \oplus B)' = A \odot B$$

$$O_{(A<B)} = A' \cdot B$$

O circuito lógico de um comparador de 1 bit é representado na Figura 6.37. A tabela verdade descrevendo a operação do comparador é mostrada na Tabela 6.13.

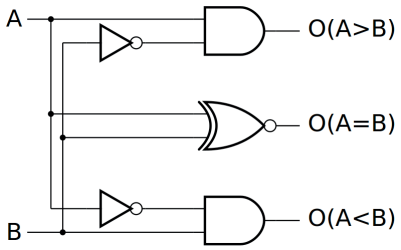


Figura 6.37 – Circuito lógico de um comparador de 1 bit

A	B	$O_{A>B}$	$O_{A=B}$	$O_{A<B}$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Tabela 6.13 – Tabela verdade para um comparador de 1 bit

Mais entradas ($I_{A<B}$, $I_{A=B}$, $I_{A>B}$) podem ser adicionadas a um comparador de 1 bit para permitir a conexão em cascata de vários circuitos do mesmo tipo. As expressões da lógica de saída são fatoradas em vez de simplificadas, de forma que a porta lógica XOR que executa a função $(A \oplus B)'$ pode ser compartilhada pelas saídas. Portanto:

$$O_{(A>B)} = (A + B') \cdot I_{(A>B)} + A \cdot B' = (A \oplus B)' \cdot I_{(A>B)} + A \cdot B'$$

$$O_{(A=B)} = (A \oplus B)' \cdot I_{(A=B)}$$

$$O_{(A<B)} = (A' + B) \cdot I_{(A<B)} + A' \cdot B = (A \oplus B)' \cdot I_{(A<B)} + A' \cdot B$$

O circuito lógico para o comparador em cascata de 1 bit é mostrado na Figura 6.38. A tabela verdade do comparador de 1 bit em cascata é mostrada na Tabela 6.14.

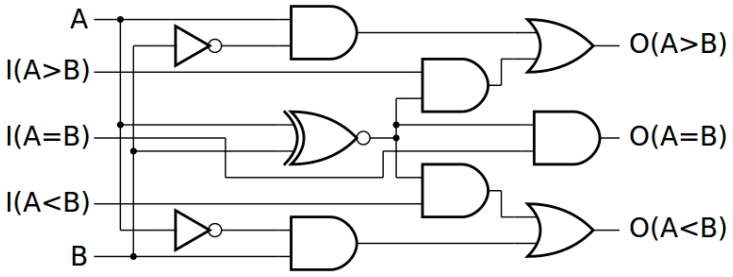


Figura 6.38 – Circuito lógico de um comparador de 1 bit em cascata

A	B	$O_{A>B}$	$O_{A=B}$	$O_{A<B}$
0	0	$I_{A>B}$	$I_{A=B}$	$I_{A<B}$
0	1	0	0	1
1	0	1	0	0
1	1	$I_{A>B}$	$I_{A=B}$	$I_{A<B}$

Tabela 6.14 – Tabela Verdade de um comparador de 1 bit em cascata

O comparador de 4 bits mostrado na Figura 6.39 é implementado por comparadores de 1 bit em cascata. A comparação é realizada de maneira iterativa, começando pelos bits menos significativos dos dados de entrada. O estado lógico 1 para as saídas, $O_{A>B}$ ou $O_{A<B}$, gerado por um determinado estágio é propagado para os estágios seguintes, enquanto se a saída $O_{A=B}$ de um determinado estágio é definida como 1, a mesma operação de comparação é repetida no próximo estágio.

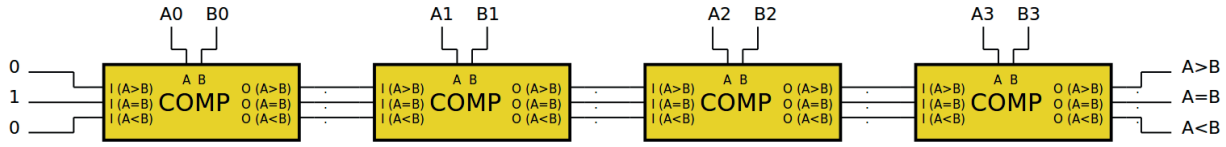


Figura 6.39 – Estrutura do comparador de 4 bits

O CI 74LS85 é um comparador de magnitude de 4 bits. Ele compara dois números binários de 4 bits A (A3 A2 A1 A0) e B (B3 B2 B1 B0) com valores de comparação anteriores $I_{A>B}$, $I_{A=B}$, $I_{A<B}$. A comparação é expressa na forma de uma única saída válida $O_{A>B}$ ou $O_{A=B}$ ou $O_{A<B}$. É possível comparar números com mais de 4 bits, interligando as saídas O's com as entradas I's de outro CI idêntico. O diagrama lógico do comparador de magnitude 74LS85 é mostrado na Figura 6.40.

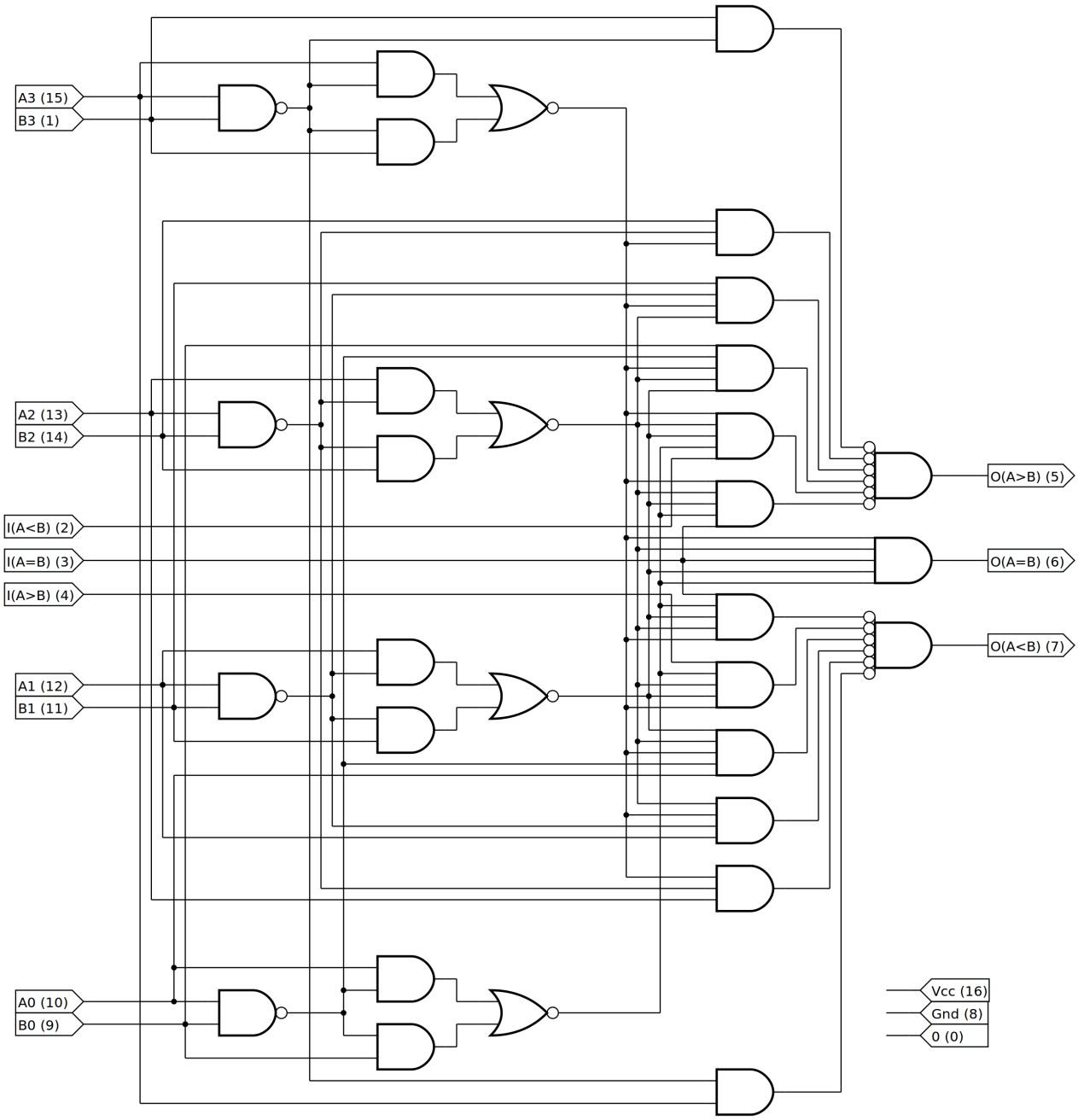


Figura 6.40 – Diagrama lógico do comparador de magnitude de 4 bits 74LS85.

6.7.3. Exame do comparador binário de 4 bits

Esquemas:

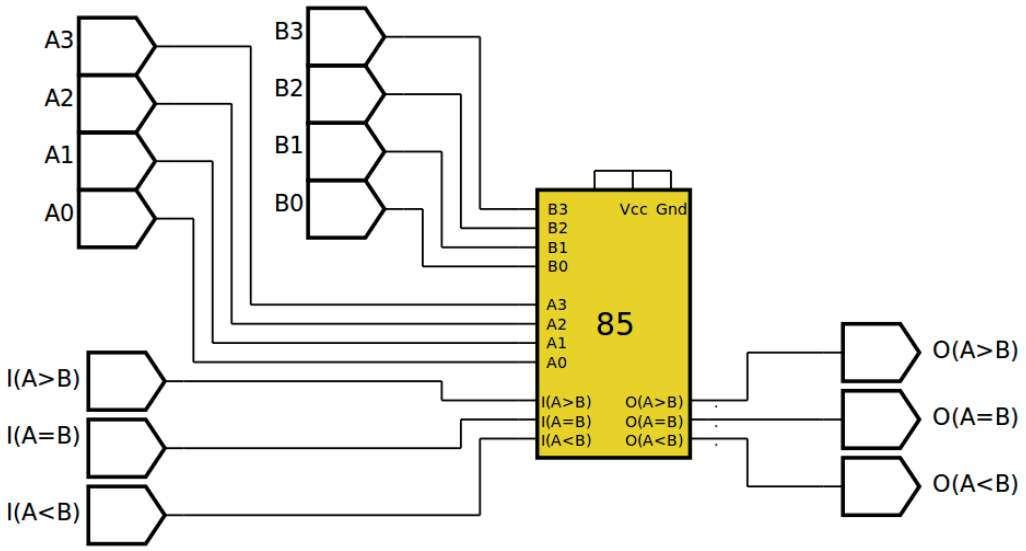


Figura 6.41 – Diagrama esquemático.

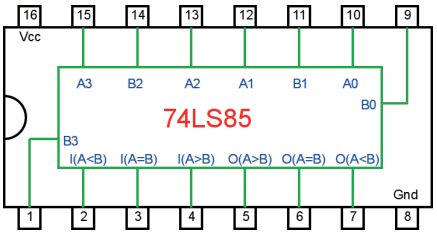


Figura 6.42 – Pinagem do CI TTL 74LS85 para a placa de montagem.

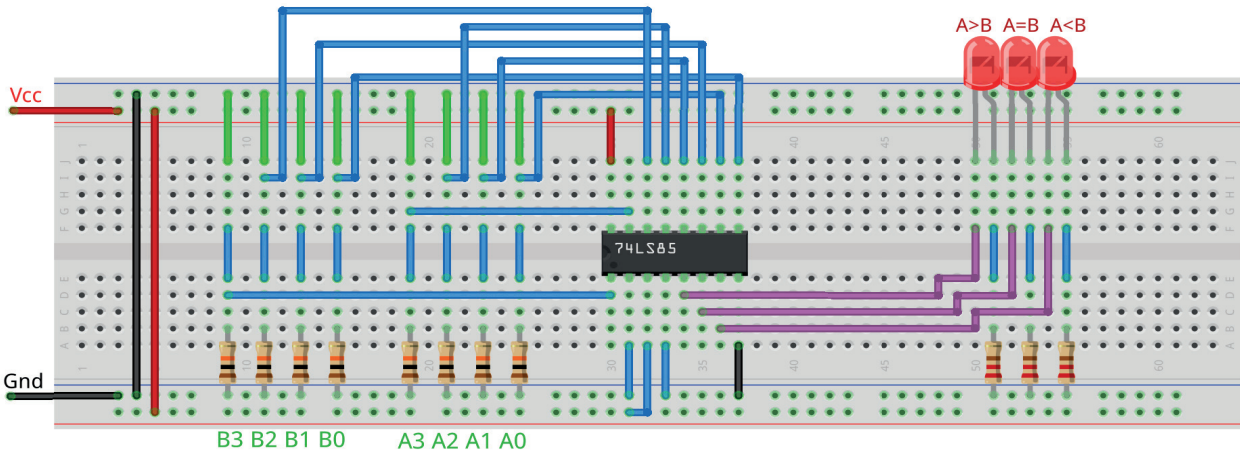


Figura 6.43 – Disposição dos componentes (TTL) na placa de montagem.

Procedimento:

1. Monte o circuito da figura 6.43 na placa de montagem, conforme figura 6.41 (versão TTL).
   a) Conecte os fios de entrada (verdes) à barra de terra (Gnd) e os fios de saída (roxos) aos LEDs.
   b) Conecte os fios de conexão (azuis) entre os resistores e as entradas do CI.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 6.15, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado ao “Gnd”.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado ao “Vcc”.
3. Para o preenchimento das colunas de saída na tabela 6.15, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 6.41. Compare os resultados com a tabela 6.15.
5. Simule o programa do comparador binário para o Arduíno.

Tabelas de dados:

Número A					Número B					Comparação		
dec	A3	A2	A1	A0	dec	B3	B2	B1	B0	A<B	A=B	A>B
5	0	1	0	1	4	0	1	0	0			
9	1	0	0	1	9	1	0	0	1			
3	0	0	1	1	7	0	1	1	1			
6	0	1	1	0	2	0	0	1	0			
8	1	0	0	0	2	0	0	1	0			
6	0	1	1	0	6	0	1	1	0			
2	0	0	1	0	8	1	0	0	0			
1	0	0	0	1	7	0	1	1	1			
9	1	0	0	1	7	0	1	1	1			
10	1	0	1	0	9	1	0	0	1			
11	1	0	1	1	6	0	1	1	0			
12	1	1	0	0	7	0	1	1	1			
9	1	0	0	1	10	1	0	1	0			
3	0	0	1	1	11	1	0	1	1			
6	0	1	1	0	12	1	1	0	0			
10	1	0	1	0	13	1	1	0	1			
11	1	0	1	1	14	1	1	1	0			
12	1	1	0	0	15	1	1	1	1			
10	1	0	1	0	10	1	0	1	0			
14	1	1	1	0	11	1	0	1	1			
15	1	1	1	1	12	1	1	0	0			

Tabela 6.15

Programa para o Arduíno:

```
// Comparador binário de 4 bits

// valores de entrada
byte q = 21;
byte Ax[] = {5,9,3,6,8,6,2,1,9,10,11,12,9,3,6,10,11,12,10,14,15};
byte Bx[] = {4,9,7,2,2,6,8,7,7,9,6,7,10,11,12,13,14,15,10,11,12};

// valores intermediários
byte x; byte y; byte a0; byte a1; byte a2; byte a3;
byte b0; byte b1; byte b2; byte b3;
byte i; byte s1; byte s2; byte s3; byte s4; byte s5;

// valores de saída
byte s_menor; byte s_igual; byte s_maior;

// se os bits do byte de entrada
void separa_bits(){
  a0 = x & B00000001;
  a1 = x & B00000010; a1 = a1 >> 1;
  a2 = x & B00000100; a2 = a2 >> 2;
  a3 = x & B00001000; a3 = a3 >> 3;
  b0 = y & B00000001;
  b1 = y & B00000010; b1 = b1 >> 1;
  b2 = y & B00000100; b2 = b2 >> 2;
  b3 = y & B00001000; b3 = b3 >> 3;
}

// comparador de 1 bit
void compara_bits(byte a, byte b, byte i_maior, byte i_igual, byte
i_menor){
  s1 = a && !b;
  s2 = !(( !a && b ) || ( a && !b ));
  s3 = !a && b;
  s4 = i_maior && s2;
  s5 = s2 && i_menor;
  s_maior = s1 || s4;
  s_igual = s2 && i_igual;
  s_menor = s5 || s3;
}
```

```
// mostra o resultado da comparação dos 4 bits
void mostra_resultado(){
  Serial.print(x);
  if(a3 == 0) Serial.print("\t0"); else Serial.print("\t1");
  if(a2 == 0) Serial.print("\t0"); else Serial.print("\t1");
  if(a1 == 0) Serial.print("\t0"); else Serial.print("\t1");
  if(a0 == 0) Serial.print("\t0"); else Serial.print("\t1");
  Serial.print("\t|\t");
  Serial.print(y);
  if(b3 == 0) Serial.print("\t0"); else Serial.print("\t1");
  if(b2 == 0) Serial.print("\t0"); else Serial.print("\t1");
  if(b1 == 0) Serial.print("\t0"); else Serial.print("\t1");
  if(b0 == 0) Serial.print("\t0"); else Serial.print("\t1");
  Serial.print("\t|\t");
  Serial.print(s_maior);
  Serial.print("\t");
  Serial.print(s_igual);
  Serial.print("\t");
  Serial.print(s_menor);
  Serial.println("");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Comparador binario de 4 bits");
  Serial.println("dec\tA3\tA2\tA1\tA0\t|\tdec\tB3\tB2\tB1\tB0\t|\tA>B\
  tA=B\tA<B");
  Serial.println("");
  for (i = 0; i < q; i++){
    x = Ax[i]; y = Bx[i];
    separa_bits();
    compara_bits(a0,b0,0,1,0);
    compara_bits(a1,b1,s_maior,s_igual,s_menor);
    compara_bits(a2,b2,s_maior,s_igual,s_menor);
    compara_bits(a3,b3,s_maior,s_igual,s_menor);
    mostra_resultado();
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

6.8. Geradores de paridade binários

6.8.1. Objetivos

1. Investigar os circuitos geradores de paridade binários,
2. Observar seu funcionamento e obter suas tabelas verdade,
3. Conhecer os CIs de circuitos geradores de paridade binários.

6.8.2. Informação preliminar

Os bits de paridade são bits que são adicionados aos dados a serem transmitidos para fins de verificação de erros. Existem dois tipos de paridade: par e ímpar. A saída de um gerador de paridade par (ímpar) é ajustada para 1 (0) se o número de bits configurado no nível lógico alto ou 1 na palavra de entrada for ímpar, ou para 0 (1) se esse número for par.

A concatenação de um bit de paridade com um dado resulta em uma palavra que pode ter um número ímpar de bits no nível lógico 1 (ou uma paridade par) ou um número par de bits no nível lógico 1 (ou uma paridade ímpar). Assim, para gerar a paridade, o número de bits no nível lógico 1 leva em consideração o bit de paridade, enquanto para checar a paridade, o bit de paridade é excluído do número total de bits no nível lógico 1 e é usado apenas para identificar tipo de paridade. A Tabela 6.16 apresenta três palavras de 8 bits com os bits de paridade correspondentes.

Palavra com	paridade	
8 bits	par	ímpar
00000000	0	1
00101000	0	1
01001100	1	0

Tabela 6.16 – Exemplo de três palavras de 8 bits com bits de paridade

A paridade é usada em sistemas de comunicação em série e memórias para detectar erros de transmissão devido a ruído. Depois que os erros que resultam na modificação do nível lógico do bit de paridade são detectados por um verificador, é possível prever uma correção pela retransmissão dos dados. A Tabela 6.17 fornece a tabela verdade de um gerador de paridade para palavras de 4 bits. Os dados de entrada são da forma $D_3D_2D_1D_0$ e a variável de saída é P_E (paridade par) ou P_O (paridade ímpar). O mapa de Karnaugh construído com base na tabela verdade, permite a determinação da expressão lógica para P_E. Nós temos assim:

$$PE = (D3' \cdot D2 \cdot D1' \cdot D0') + (D3 \cdot D2' \cdot D1' \cdot D0') + (D3' \cdot D2' \cdot D1' \cdot D0) + (D3 \cdot D2 \cdot D1' \cdot D0)$$

$$+ (D3' \cdot D2 \cdot D1 \cdot D0\,) + (D3 \cdot D2' \cdot D1 \cdot D0) + (D3' \cdot D2' \cdot D1 \cdot D0) + D(3 \cdot D2 \cdot D1 \cdot D0)$$

ou equivalente:

$$P_E = (D_3' \cdot D_2 + D_3 \cdot D_2') D_1' \cdot D_0' + (D_3' \cdot D_2' + D_3 \cdot D_2) D_1' \cdot D_0$$

$$+ (D_3' \cdot D_2 + D_3 \cdot D_2') D_1 \cdot D_0 + (D_3' \cdot D_2' + D_3 \cdot D_2) D_1 \cdot D_0'$$

$$= (D_3 \oplus D_2) . (D_1 \oplus D_0)' + (D_3 \oplus D_2)' . (D_1 \oplus D_0)$$

$$= D_3 \oplus D_2 \oplus D_1 \oplus D_0$$

Como as duas saídas P_E e P_O são complementares, segue-se que: $P_O = P_E'$

Usando portas lógicas OU-Exclusivo e um inversor, o circuito lógico de um gerador de paridade para palavras de 4 bits é implementado como mostrado na Figura 6.44(a). Outra versão do gerador de paridade é dada na Figura 6.44(b). A porta de saída OU-EX é configurada como um inversor que pode ser programado pelo sinal de seleção E/S; isto é útil para verificar a paridade par, quando E'/O é ajustado para 0, ou a paridade ímpar quando E'/O leva o nível lógico 1.

D3	D2	D1	D0	P_E	P_O
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	1

Tabela 6.17 – Tabela verdade para um gerador de paridade para palavras de 4 bits.

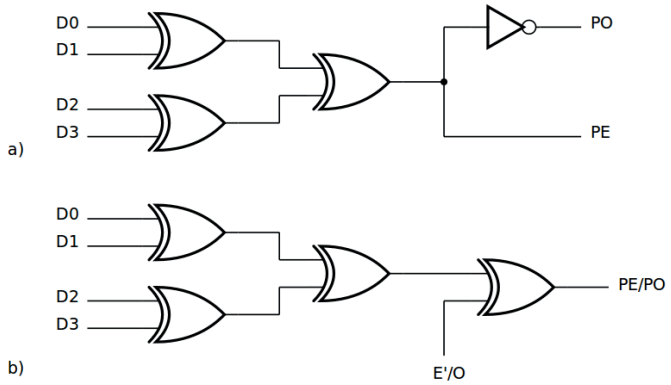


Figura 6.44 – Gerador de paridade a) com ou b) sem sinal de seleção

O circuito lógico mostrado na Figura 6.45 é um gerador de paridade ou verificador que seleciona entradas para escolher o tipo de paridade desejado. O sinal P toma o nível lógico 1 se a palavra de entrada contiver um número par de bits ajustado em 1. Uma análise da tabela de funções dada na Tabela 6.18, onde x denota o termo não importa, mostra que cada sinal de saída (ΣE ou ΣO) pode ser ativo alto ou baixo dependendo da combinação de níveis lógicos nas entradas selecionadas.

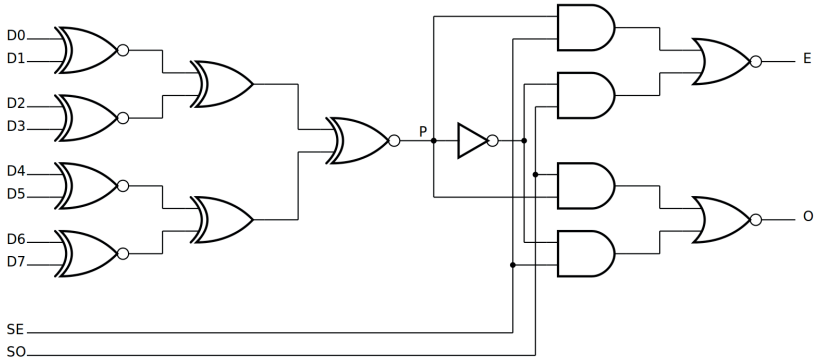


Figura 6.45 – Gerador de paridade / verificador 74LS180

Entradas			Saídas	
Numero de bits de D0 a D7 em "1"	SE	SO	E	O
par	1	0	1	0
impar	1	0	0	1
par	0	1	0	1
impar	0	1	1	0
x	1	1	0	0
x	0	0	1	1

Tabela 6.18 – Tabela que ilustra o funcionamento do gerador / verificador de paridade

A configuração mostrada na Figura 6.46(a), onde a entrada SE é ajustada ao nível lógico 1 e conectada por um inversor à entrada SO, permite a geração de um bit de paridade par. As palavras de 9 bits são formadas pela concatenação do bit de paridade e da palavra de entrada. O verificador de paridade par para palavras de 9 bits é representado na Figura 6.46(b), onde o bit de paridade e o complemento do bit de paridade são conectados às entradas SE e SO, respectivamente. Se existir uma paridade par, a saída E vai para o nível lógico “1” enquanto a saída O é definida como “0”.

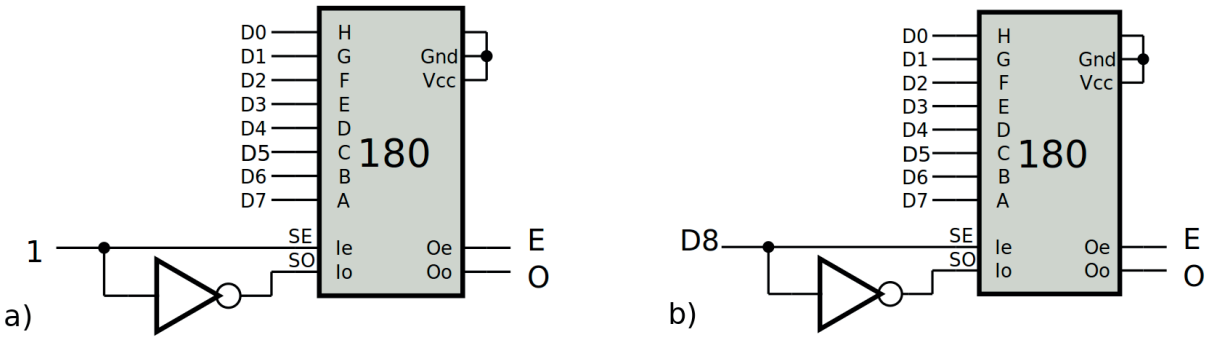


Figura 6.46 – a) Gerador de paridade; b) verificador de paridade para palavras de 9 bits

6.8.3. Exame do gerador de paridade de 8 bits

Esquemas:

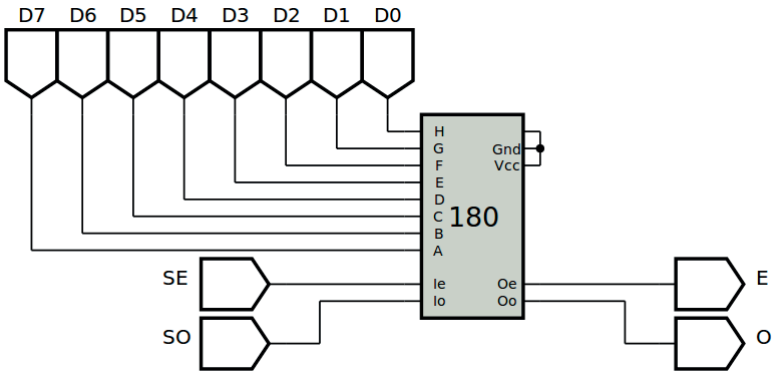


Figura 6.47 – Diagrama esquemático.

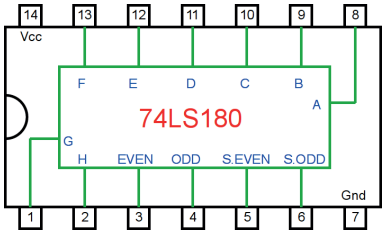


Figura 6.48 – Pinagem do CI TTL 74LS180 para a placa de montagem.

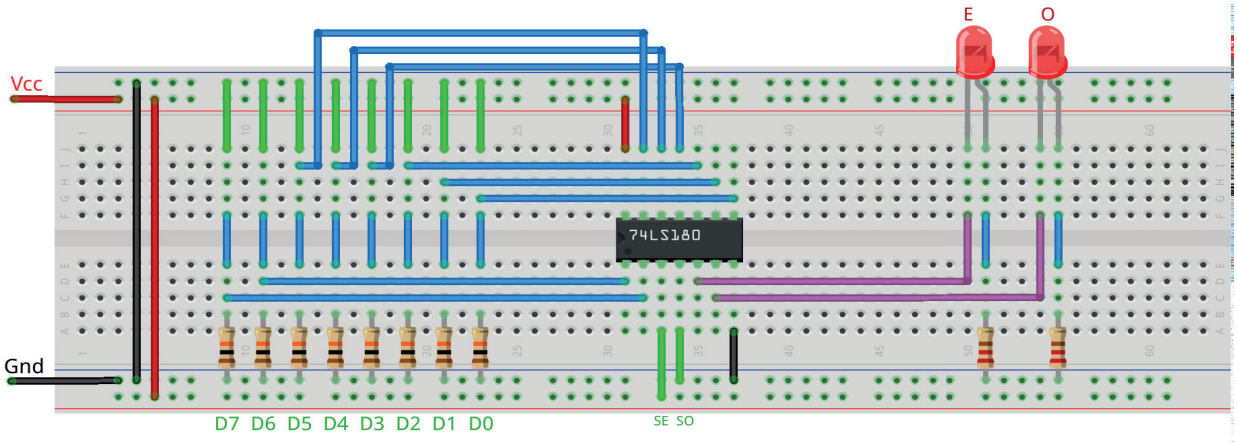


Figura 6.49 – Disposição dos componentes (TTL) na placa de montagem.

Procedimento:

1. Monte o circuito da figura 6.49 na placa de montagem, conforme figura 6.47 (versão TTL).
   a) Conecte os fios de entrada (verdes) à barra de terra (Gnd) e os fios de saída (roxos) aos LEDs.
   b) Conecte os fios de conexão (azuis) entre os resistores e as entradas do CI.
   c) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   d) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   e) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 6.19, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado ao “Gnd”.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado ao “Vcc”.
3. Para o preenchimento das colunas de saída na tabela 6.19, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 6.47. Compare os resultados com a tabela 6.19.
5. Simule o programa do gerador de paridade binário para o Arduíno.

Tabelas de dados:

D7	D6	D5	D4	D3	D2	D1	D0	SE	SO	E	O
0	0	0	0	0	0	0	1	1	0		
0	1	0	1	0	0	1	0	1	0		
0	0	1	1	1	1	0	0	1	0		
0	1	1	0	0	0	1	1	1	0		
1	1	1	0	0	1	1	0	1	0		
0	1	0	1	0	1	0	1	0	1		
1	1	0	1	1	1	0	0	0	1		
1	1	0	0	0	1	1	1	0	1		
0	0	0	1	1	1	0	0	0	1		
1	1	1	0	0	0	1	0	0	0		
1	0	0	1	1	1	0	0	1	1		

Tabela 6.19

Programa para o Arduíno:

```
// Gerador de paridade binário de 8 bits

// valores de entrada
byte q = 11;
byte Dx[] = {B00000001,B01010010,B00111100,B01100011,B11100110,B010101
01,B11011100,B11000111,B00011100,B11100010,B10011100};
byte SEx[] = {1,1,1,1,1,0,0,0,0,0,1};
byte SOx[] = {0,0,0,0,0,1,1,1,1,0,1};

// valores intermediários
byte d; byte se; byte so;
byte d7; byte d6; byte d5; byte d4; byte d3; byte d2; byte d1; byte d0;
byte s1; byte s2; byte s3; byte s4; byte s5; byte s6;
byte s7; byte s8; byte s9; byte s10; byte s11; byte i;

// valores de saída
byte e; byte o;

void separa_bits(){
  d0 = d & B00000001;
  d1 = d & B00000010; d1 = d1 >> 1;
  d2 = d & B00000100; d2 = d2 >> 2;
  d3 = d & B00001000; d3 = d3 >> 3;
  d4 = d & B00010000; d4 = d4 >> 4;
  d5 = d & B00100000; d5 = d5 >> 5;
  d6 = d & B01000000; d6 = d6 >> 6;
  d7 = d & B10000000; d7 = d7 >> 7;
}

void calcula_paridade(){
  s1 = !((!d0&&d1)||(d0&&!d1));
  s2 = !((!d2&&d3)||(d2&&!d3));
  s3 = !((!d4&&d5)||(d4&&!d5));
  s4 = !((!d6&&d7)||(d6&&!d7));
  s5 = (!s1&&s2)||(s1&&!s2);
  s6 = (!s3&&s4)||(s3&&!s4);
  s7 = !((!s5&&s6)||(s5&&!s6));
  s8 = s7&&se;
  s9 = !s7&&so;
  s10 = s7&&so;
```

```
  s11 = !s7&&se;
  e = !(s8||s9);
  o = !(s10||s11);
}

void mostra_resultado(){
  if (d7) Serial.print("1 "); else Serial.print("0 ");
  if (d6) Serial.print("1 "); else Serial.print("0 ");
  if (d5) Serial.print("1 "); else Serial.print("0 ");
  if (d4) Serial.print("1 "); else Serial.print("0 ");
  if (d3) Serial.print("1 "); else Serial.print("0 ");
  if (d2) Serial.print("1 "); else Serial.print("0 ");
  if (d1) Serial.print("1 "); else Serial.print("0 ");
  if (d0) Serial.print("1 "); else Serial.print("0 ");
  if (se) Serial.print("1 "); else Serial.print("0 ");
  if (so) Serial.print("1 "); else Serial.print("0 ");
  if (e) Serial.print("1 "); else Serial.print("0 ");
  if (o) Serial.print("1 "); else Serial.print("0 ");
  Serial.println("");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Gerador de paridade binário de 8 bits");
  Serial.println("D7 D6 D5 D4 D3 D2 D1 D0 SE SO E O");
  Serial.println("-----------------------------------");

  for (i = 0; i < q; i++){
    d = Dx[i];
    se = SEx[i];
    so = SOx[i];
    separa_bits();
    calcula_paridade();
    mostra_resultado();
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

Capítulo 7. Circuitos lógicos combinacionais: decodificadores e codificadores

7.1. Decodificadores

7.1.1. Objetivos

1. Investigar os circuitos de decodificadores,
2. Observar seu funcionamento e obter suas tabelas verdade,
3. Conhecer os circuitos de decodificadores.

7.1.2. Informação preliminar

Na teoria de circuitos digitais, a lógica combinacional (às vezes também chamada de lógica combinatória) é um tipo de lógica digital que é implementada por circuitos booleanos, onde a saída é uma função pura da entrada atual apenas. Isso está em contraste com a lógica sequencial, na qual a saída depende não apenas da entrada atual, mas também do histórico da entrada. Em outras palavras, a lógica sequencial tem memória, enquanto a lógica combinacional não. A lógica combinatória é usada em circuitos de computador para fazer álgebra booleana em sinais de entrada e em dados armazenados. Circuitos de computador práticos normalmente contêm uma mistura de lógica combinacional e sequencial. Por exemplo, a parte de uma unidade lógica aritmética, ou ALU, que faz cálculos matemáticos é construída usando lógica combinacional. Outros circuitos usados em computadores, como meio somadores, somadores completos, meio subtratores, subtratores completos, multiplexadores, demultiplexadores, codificadores e decodificadores também são feitos usando lógica combinacional. Estes experimentos tratam de decodificadores e codificadores.

Um decodificador é um circuito que altera um código em um conjunto de sinais. É chamado de decodificador porque faz o contrário da codificação. Um tipo comum de decodificador é o decodificador de linha que pega um dado binário de entrada de "*m*" bits e o decodifica em linhas de dados de "2^m". Como um componente combinacional padrão, um decodificador, ativa um de "*n*" linhas de saída, dependendo do valor de um dado binário de entrada de "*m*" bits. As saídas de um decodificador podem ser ativa em baixa ou ativa em alta. Quando as saídas estão ativas em alta (respectivamente ativa em baixa), a saída declarada é alta (respectivamente baixa) e o restante das outras saídas são baixas (respectivamente altas). A forma geral de um decodificador *m*-para-*n* pode ser vista na Figura 7.1. Em geral, um decodificador *m*-para-*n* tem *m* linhas de entrada, i_{m-1}, ..., i_1, i_0 e *n* linhas de saída, d_{n-1}, ..., d_1, d_0, onde $n = 2^m$. Como mostrado na Figura 7.1, além de linhas de entrada e linhas de saída, um decodificador tem uma linha de habilitação, "E", para habilitar o decodificador. Quando o decodificador é desabilitado com "E" definido como 0 (para um alto ativo habilita a entrada E), todas as linhas de saída são desabilitadas. Quando o decodificador é habilitado, a linha de saída cujo índice é igual ao valor dos dados binários de entrada é ativada (definida como 1 para um alto ativo), enquanto o restante das linhas de saída são desativadas (definido como 0 para alto ativo). Um decodificador é usado em um sistema com vários componentes e queremos que apenas um componente seja selecionado ou ativado a qualquer momento.

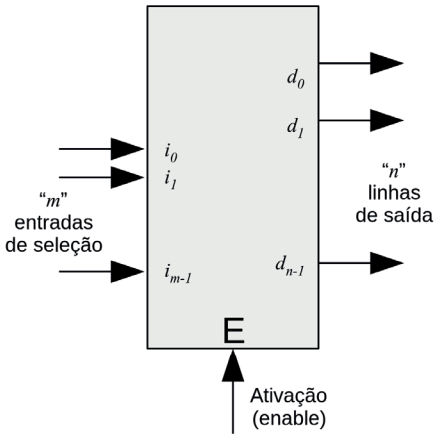
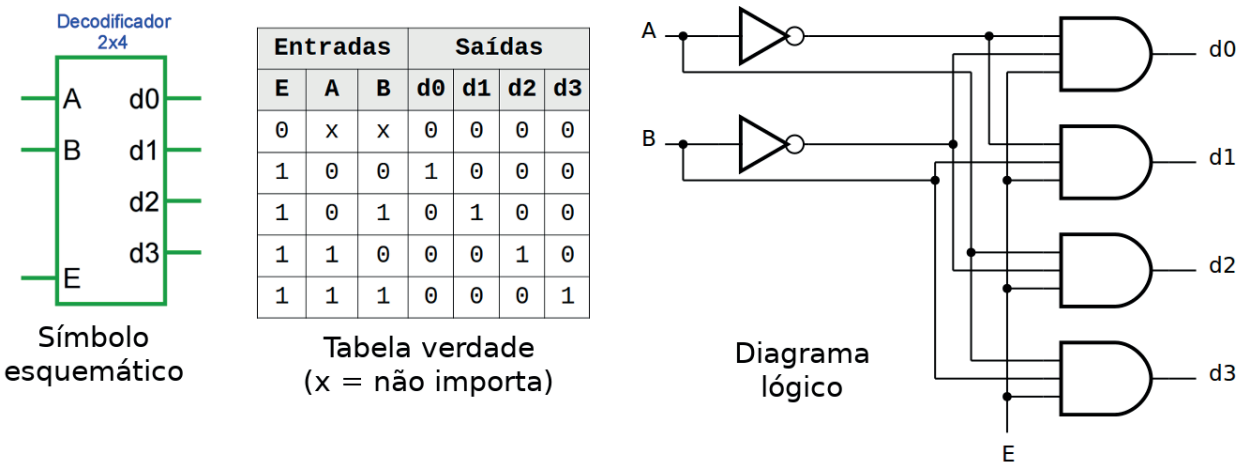


Figura 7.1 – A forma geral de um decodificador m-para-n, onde n = 2^m.

Em geral, os decodificadores são produzidos em circuitos integrados como decodificadores 2x4, 3x8 e 4x16. O símbolo esquemático, a tabela verdade e o diagrama lógico de um decodificador 2x4 com entrada de habilitação alta ativa (E) e saídas ativas altas (d0, d1, d2 e d3) são fornecidos na Figura 7.2. Neste decodificador, um alto ativo na entrada de habilitação E, seleciona as entradas A e B, e ativa um alto nos sinais de saída "d0", "d1", "d2" e "d3" que são todas variáveis booleanas. Quando este decodificador é desativado com E definido como 0, todas as linhas de saída ativas em alto são desativadas (definidas como 0). Quando este decodificador é ativado com E definido como 1: se as entradas de seleção forem AB = 00, (respectivamente, 01, 10, 11), a linha de saída, d0 (respectivamente, d1, d2, d3), é ativada (definido como 1) e todas as outras linhas de saída são desativadas (definidas como 0).

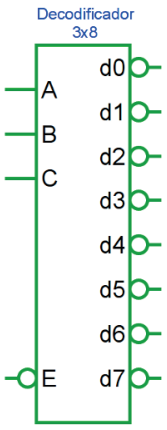


Entradas			Saídas			
E	A	B	d0	d1	d2	d3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Figura 7.2 – O símbolo esquemático, a tabela de verdade e o diagrama lógico de um decodificador 2x4 com entrada de habilitação em alto ativo e com alto ativo nas saídas.

O símbolo esquemático, a tabela verdade e o diagrama lógico de um decodificador 3x8 com entrada de habilitação em baixo ativo (E) e fornece um baixo ativo para saídas (d0, d1, d2, d3, d4, d5, d6 e d7) são fornecidos na Figura 7.3. Neste decodificador, um baixo ativo na entrada de habilitação E, seleciona as entradas A, B e C, e fornece um baixo ativo para as saídas "d0", "d1", "d2", "d3", "d4", "d5", "d6" e "d7" que são todas variáveis booleanas. Quando este decodificador é desativado com E definido como 1, todas as linhas de saída com baixo ativo são desativadas (definidas como 1). Quando este decodificador é habilitado com E definido como 0, ou seja, quando E = 0: se as entradas de seleção forem ABC = 000, (respectivamente, 001, 010, 011, 100,

101, 110, 111), a linha de saída, d_0 (respectivamente, d_1, d_2, d_3, d_4, d_5, d_6, d_7), é ativada (definido para 0) e todas as outras linhas de saída são desativadas (definidas para 1).



Símbolo esquemático

Entradas				Saídas							
E	A	B	D	d0	d1	d2	d3	d4	d5	d6	d7
1	x	x	x	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	0

Tabela verdade
(x = não importa)

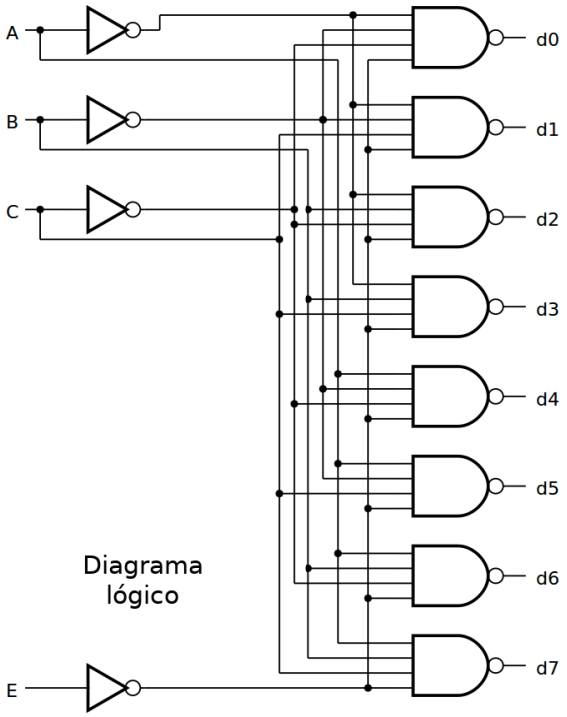


Figura 7.3 – O símbolo esquemático, a tabela verdade e o diagrama lógico de um decodificador 3x8 com entrada de habilitação em baixo ativo e baixo ativo para saídas.

Dois ou mais decodificadores pequenos, como 2x4, 3x8, 4x16 com entradas de habilitação, podem ser combinados para formar um decodificador maior. Por exemplo, um decodificador 4x16 é construído a partir de dois decodificadores 3x8, como mostra a Figura 7.4. Neste circuito, quando $A_3 = 0$, o decodificador 3x8 superior é ativado e baseado nos valores lógicos aplicados às entradas A_2, A_1, A_0, uma das saídas (D0, D1, D2, D3, D4, D5, D6 ou D7) é ativado e definido como 1. Quando $A_3 = 0$, o decodificador 3x8 inferior é desativado e todas as suas saídas, a saber, D8, D9, D10, D11, D12, D13, D14 e D15 são desativadas. Quando $A_3 = 1$, o decodificador 3x8 superior é desativado e todas as saídas, D0, D1, D2, D3, D4, D5, D6 e D7 são desativadas. Quando $A_3 = 1$, o decodificador 3x8 inferior é habilitado e baseado nos valores lógicos aplicados às entradas A_2 , A_1, A_0, uma das saídas (D8, D9, D10, D11, D12, D13, D14 ou D15) é ativada e definida como 1.

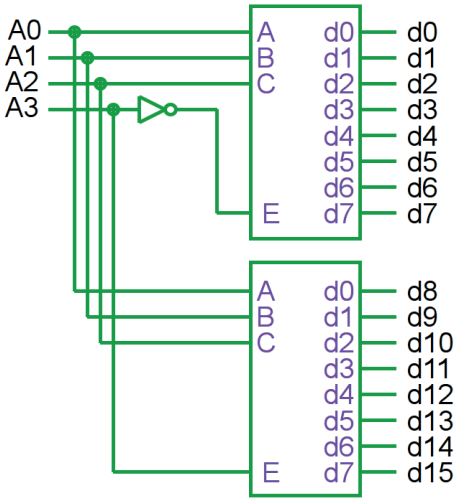
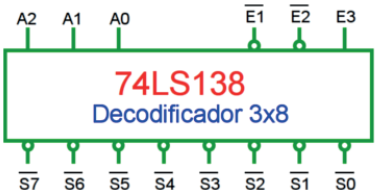


Figura 7.4 – Construção de um decodificador 4x16 a partir de dois decodificadores 3x8.

7.1.3. Circuito Integrado TTL 74LS138 – Decodificador 3x8

A Figura 7.5 mostra o símbolo esquemático, o diagrama lógico e a tabela verdade do CI decodificador 3x8 74LS138. O 74LS138 decodifica uma das oito linhas (S7’, S6’, S5’, S4’, S3’, S2’, S1’ e S0’), com base nas condições nas três entradas de seleção binárias (A2, A1 e A0) e as três entradas de habilitação (E1’, E2’ e E3). As saídas do 74LS138 são em baixo ativo. Duas entradas de habilitação em baixo ativo (E1’, E2’) e uma em alto ativo (E3) reduzem a necessidade de portas ou inversores externos quando em expansão.



Símbolo esquemático

Entradas						Saídas							
E1	E2	E3	A2	A1	A0	S7	S6	S5	S4	S3	S2	S1	S0
1	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	0	x	x	x	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1

Tabela verdade (x= não importa)

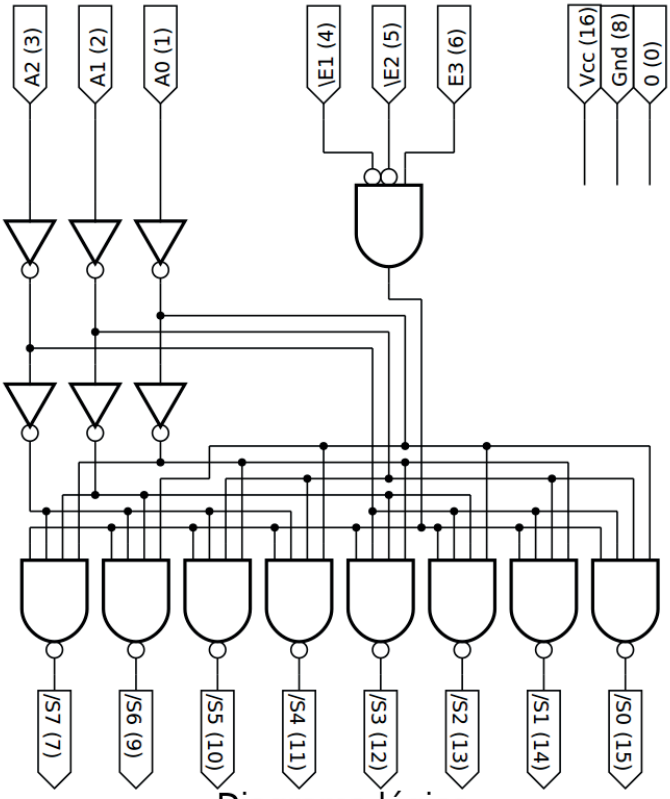


Diagrama lógico

Figura 7.5 – O símbolo esquemático, o diagrama lógico e a tabela verdade do CI decodificador 3x8 74LS138.

7.1.4. Exame do circuito decodificador 3x8 74LS138

Esquemas:

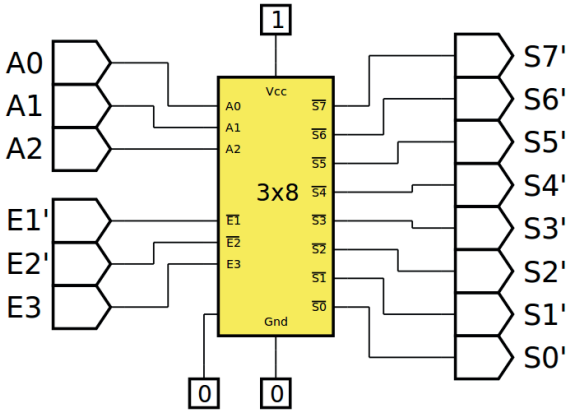


Figura 7.6 – Diagrama esquemático.

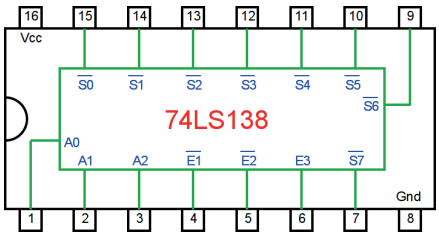


Figura 7.7– Circuito integrado TTL.

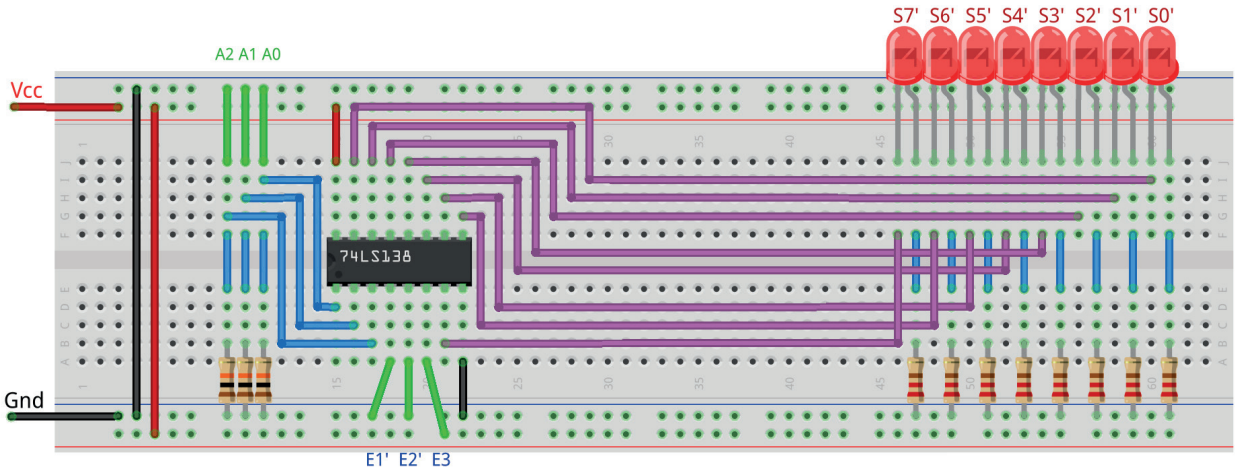


Figura 7.8 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 7.8.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 7.1, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 7.1, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 7.6. Compare os resultados com a tabela 7.1.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 7.1.

Tabelas de dados:

Entradas						Saídas							
E1	E2	E3	A2	A1	A0	S7	S6	S5	S4	S3	S2	S1	S0
1	x	x	x	x	x								
x	1	x	x	x	x								
x	x	0	x	x	x								
0	0	1	0	0	0								
0	0	1	0	0	1								
0	0	1	0	1	0								
0	0	1	0	1	1								
0	0	1	1	0	0								
0	0	1	1	0	1								
0	0	1	1	1	0								
0	0	1	1	1	1								

Tabela 7.1

Programa para o Arduíno:

```
// Decodificador 3x8

byte a2; byte a1; byte a0;
byte e1; byte e2; byte e3; byte en;
byte s0; byte s1; byte s2; byte s3; byte s4; byte s5; byte s6; byte s7;

void decod3x8(byte c, byte b, byte a, byte e1x, byte e2x, byte e3x){
  en = ((!e1x && !e2x) && e3x); // enable
  s0 = !(( en && !c ) && !b ) && !a ); // 000
  s1 = !(( en && !c ) && !b ) && a ); // 001
  s2 = !(( en && !c ) && b ) && !a ); // 010
  s3 = !(( en && !c ) && b ) && a ); // 011
  s4 = !(( en && c ) && !b ) && !a ); // 100
  s5 = !(( en && c ) && !b ) && a ); // 101
  s6 = !(( en && c ) && b ) && !a ); // 110
  s7 = !(( en && c ) && b ) && a ); // 111
}

void mostra_decod3x8(){
  if (a2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (s7) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s6) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s5) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s4) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Decodificador 3x8");
  Serial.println("| A2 | A1 | A0 || S7 | S6 | S5 | S4 | S3 | S2 | S1 |
  S0 |");
```

```
  Serial.println("-------------------------------------------------------
  ----");
  e1 = 0;
  e2 = 0;
  e3 = 1;
  for (a2 = 0; a2 <= 1; a2++){
    for (a1 = 0; a1 <= 1; a1++){
      for (a0 = 0; a0 <= 1; a0++){
        decod3x8(a2,a1,a0,e1,e2,e3);
        mostra_decod3x8();
      }
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

7.1.5. Implementação de uma função booleana usando um decodificador 3x8

Esquemas:

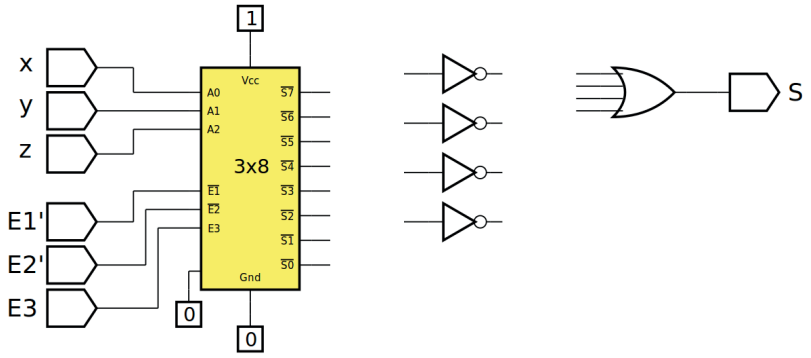


Figura 7.9 – Diagrama esquemático.

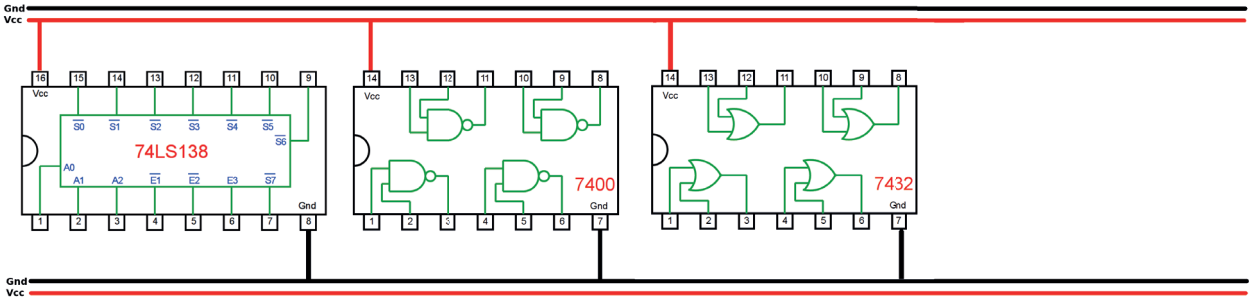


Figura 7.10 – Disposição dos componentes (TTL) para a placa de montagem.

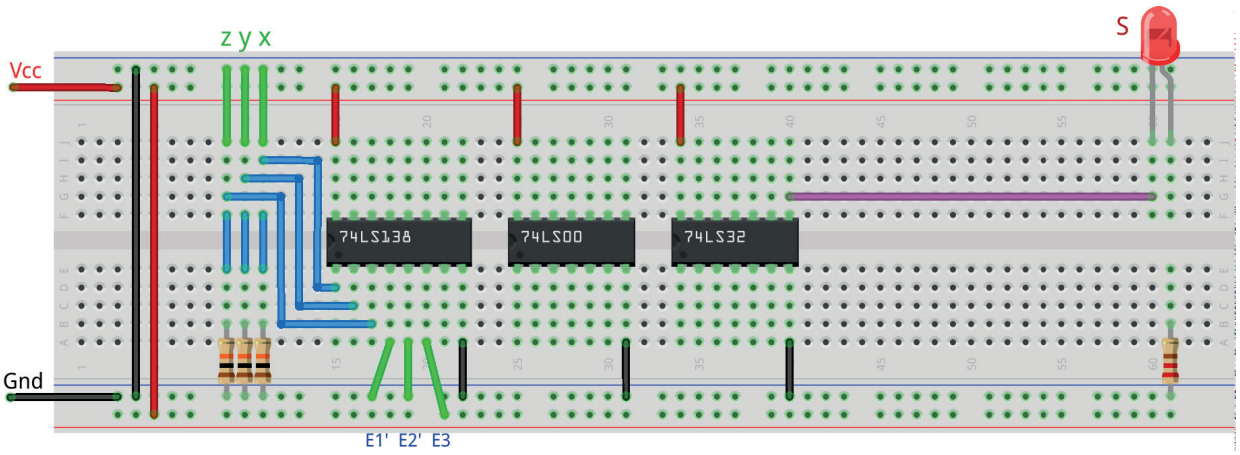


Figura 7.11 – Disposição dos componentes na placa de montagem.

Equação booleana:

S(x,y,z) = _____

Procedimento:

1. Complete a equação booleana para implementar a função: $S(x,y,z) = \Sigma_m(1,2,4,7)$.
2. Complete o desenho da figura 7.9, 7.10 e 7.11, conforme a equação booleana encontrada. Utilize as portas NE, do CI TTL 74LS00, como inversores e utilize duas portas OU, do CI TTL 74LS32, como uma porta OU de quatro entradas.
3. Monte o circuito na placa de montagem, conforme figura 7.11.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI 74LS138.
   d) Utilize fios azuis para conectar entre o CI 74LS138 e os CIs de portas lógicas.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 7.2, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
5. Para o preenchimento das colunas de saída na tabela 7.2, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
6. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 7.9. Compare os resultados com a tabela 7.2.
7. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 7.2.

Tabelas de dados:

Entradas						Saída
E1	E2	E3	z	y	x	S
0	0	1	0	0	0	
0	0	1	0	0	1	
0	0	1	0	1	0	
0	0	1	0	1	1	
0	0	1	1	0	0	
0	0	1	1	0	1	
0	0	1	1	1	0	
0	0	1	1	1	1	

Tabela 7.2

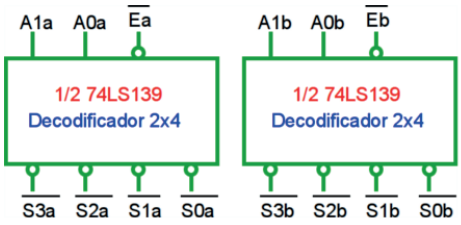
Programa para o Arduíno:

```
// Implementação de uma função booleana usando um decodificador 3x8
// S(x,y,z) = Σm(1,2,4,7).

byte a2; byte a1; byte a0;
byte e1; byte e2; byte e3; byte en;
byte s0; byte s1; byte s2; byte s3; byte s4; byte s5; byte s6; byte s7;
byte sx;

void decod3x8(byte c, byte b, byte a, byte e1x, byte e2x, byte e3x){
  en = ((!e1x && !e2x) && e3x); // enable
  s0 = !(( en && !c ) && !b ) && !a ); // 000
  s1 = !(( en && !c ) && !b ) && a ); // 001
  s2 = !(( en && !c ) && b ) && !a ); // 010
  s3 = !(( en && !c ) && b ) && a ); // 011
  s4 = !(( en && c ) && !b ) && !a ); // 100
  s5 = !(( en && c ) && !b ) && a ); // 101
  s6 = !(( en && c ) && b ) && !a ); // 110
  s7 = !(( en && c ) && b ) && a ); // 111
}

void mostra_decod3x8a(){
  if (a2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (s7) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s6) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s5) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s4) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (sx) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}
```

```
void setup(){
  Serial.begin(9600);
  Serial.println("Implementação de uma função booleana usando um deco-
  dificador 3x8");
  Serial.println("| A2 | A1 | A0 || S7 | S6 | S5 | S4 | S3 | S2 | S1 |
  S0 || Sx |");

  Serial.println("-----------------------------------------------------
  ------------");
  e1 = 0;
  e2 = 0;
  e3 = 1;

  for (a2 = 0; a2 <= 1; a2++){
    for (a1 = 0; a1 <= 1; a1++){
      for (a0 = 0; a0 <= 1; a0++){
        decod3x8(a2,a1,a0,e1,e2,e3);
        sx = ((( !s1 || !s2 ) || !s4 ) || !s7 );
        mostra_decod3x8a();
      }
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

7.1.6. Circuito Integrado TTL 74LS139 – 2 Decodificadores 2x4

A Figura 7.12 mostra o símbolo esquemático, o diagrama lógico e a tabela verdade do CI 74LS139 com dois decodificadores independentes 2x4. Como pode ser visto no diagrama lógico, o 74LS139 contém dois decodificadores 2x4 totalmente independentes. Cada decodificador 2x4 do 74LS139 decodifica uma de quatro linhas (S3’, S2’, S1’ e S0’), com base nas condições nas duas entradas de seleção binária (A1 e A0) e na entrada de habilitação em baixo ativo (E’). As saídas de cada decodificador 2x4 do 74LS139 são em baixo ativo.



Símbolo esquemático

Entradas			Saídas			
$\overline{E}$	A1	A0	$\overline{S3}$	$\overline{S2}$	$\overline{S1}$	$\overline{S0}$
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Tabela verdade
(x = não importa)

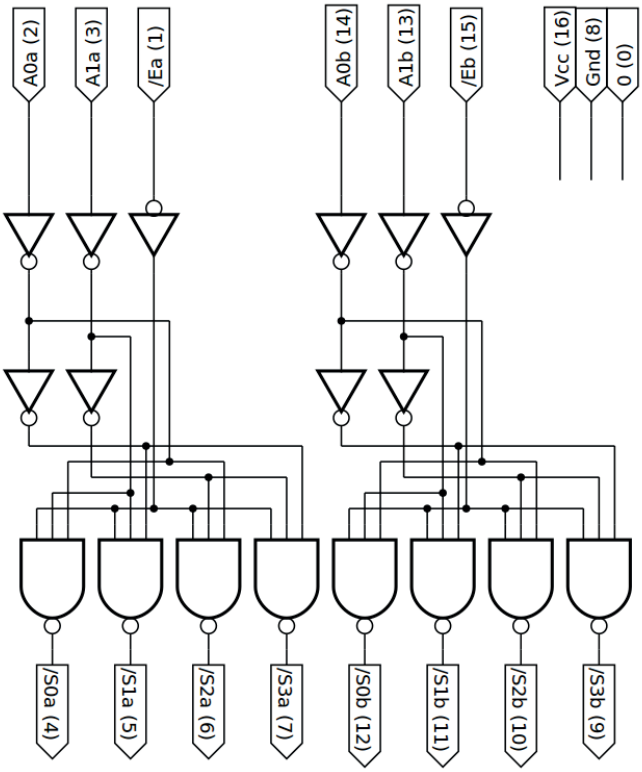


Diagrama lógico

Figura 7.12 – O símbolo esquemático, o diagrama lógico e a tabela verdade do CI 74LS139 com dois decodificadores 2x4.

7.1.7. Exame do circuito decodificador duplo 2x4 74LS139

Esquemas:

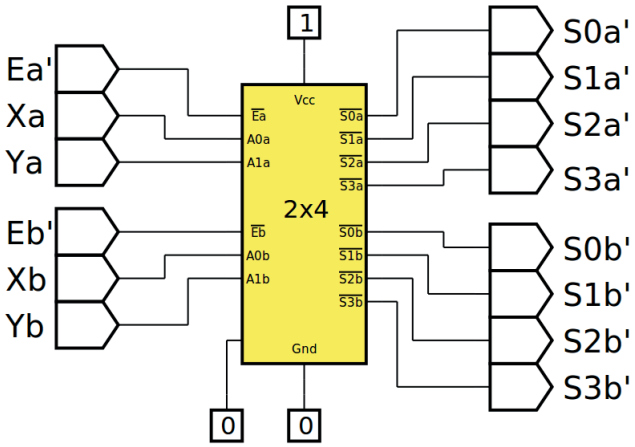


Figura 7.13 – Diagrama esquemático.

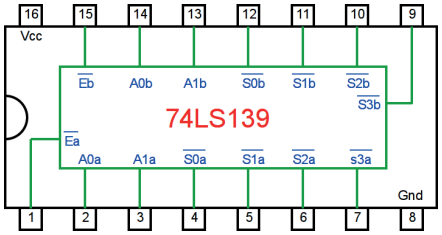


Figura 7.14 – Circuito integrado TTL.

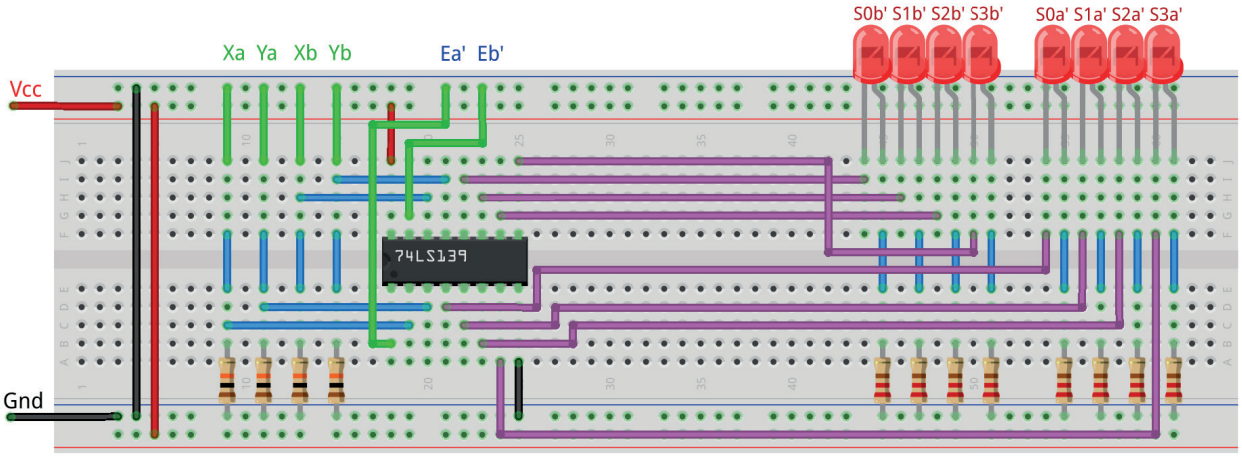


Figura 7.15 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 7.15.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 7.3, use os fios verdes como segue:
   a) Se o valor da variável for "0" o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for "1" o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 7.3, considere:
   a) Se o LED estiver apagado então preencher com "0".
   b) Se o LED estiver aceso então preencher com "1".
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 7.13. Compare os resultados com a tabela 7.3.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 7.3.

Tabelas de dados:

Entradas			Saídas			
Ea'	Ya	Xa	S3a'	S2a'	S1a'	S0a'
1	x	x				
0	0	0				
0	0	1				
0	1	0				
0	1	1				

Grupo "a"

Entradas			Saídas			
Eb'	Yb	Xb	S3b'	S2b'	S1b'	S0b'
1	x	x				
0	0	0				
0	0	1				
0	1	0				
0	1	1				

Grupo "b"

Tabela 7.3 (x = não importa)

Programa para o Arduíno:

```
// Decodificador 2x4
byte a1; byte a0; byte e; byte s0; byte s1; byte s2; byte s3;

void decod2x4(byte b, byte a, byte en){
  s0 = !(( !en && !b ) && !a ); // 00
  s1 = !(( !en && !b ) && a ); // 01
  s2 = !(( !en && b ) && !a ); // 10
  s3 = !(( !en && b ) && a ); // 11
}

void mostra_decod2x4(){
  if (a1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (s3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Decodificador 2x4");
  Serial.println("| A1 | A0 || S3 | S2 | S1 | S0 |");
  Serial.println("--------------------------------");
  e = 0;
  for (a1 = 0; a1 <= 1; a1++){
    for (a0 = 0; a0 <= 1; a0++){
      decod2x4(a1,a0,e);
      mostra_decod2x4();
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

7.1.8. Construção de um decodificador 3x8 usando decodificadores 2x4

Esquemas:

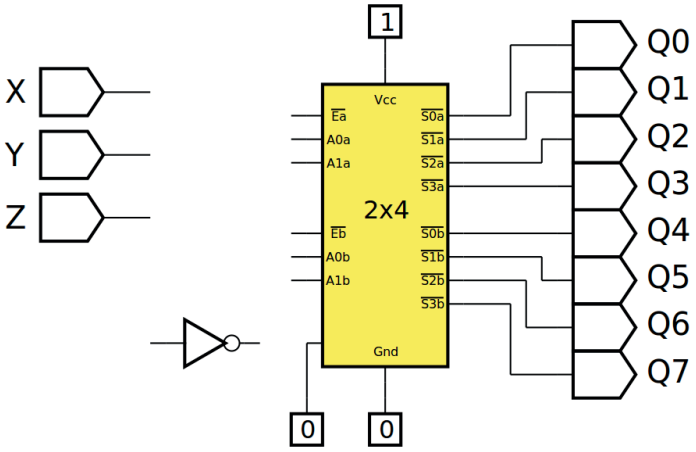


Figura 7.16 – Diagrama esquemático.

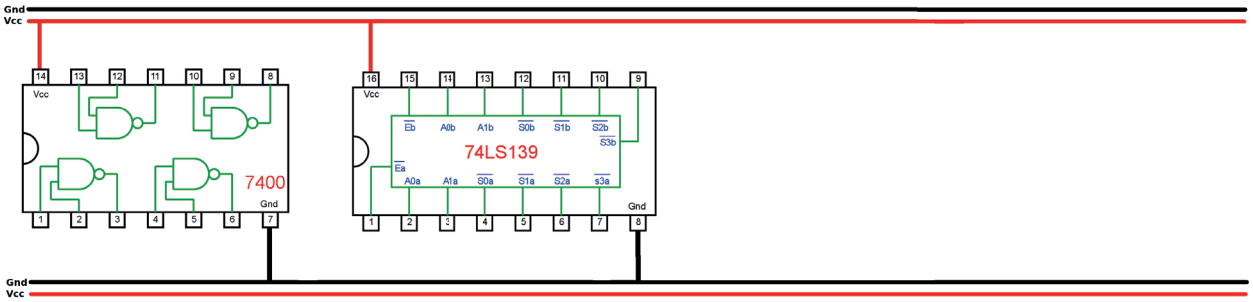


Figura 7.17 – Disposição dos componentes (TTL) para a placa de montagem.

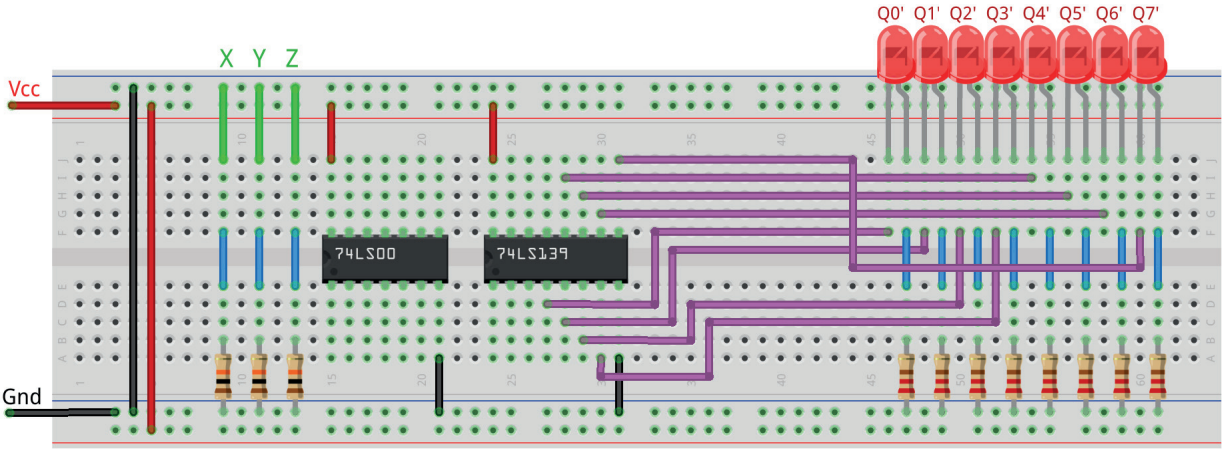


Figura 7.18 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Complete o desenho da figura 7.16, 7.17 e 7.18. Utilize as portas NE, do CI TTL 74LS00, como inversores.
2. Monte o circuito na placa de montagem, conforme figura 7.18.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI 74LS139.
   d0 Utilize fios azuis para conectar entre o CI 74LS139 e os CIs de portas lógicas.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Para o preenchimento da tabela 7.4, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
4. Para o preenchimento das colunas de saída na tabela 7.4, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 7.16. Compare os resultados com a tabela 7.4.
6. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 7.4.

Tabelas de dados:

Entradas			Saídas							
Z	Y	X	Q7’	Q6’	Q5’	Q4’	Q3’	Q2’	Q1’	Q0’
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

Tabela 7.4

Programa para o Arduíno:

```
// Construção de um decodificador 3x8 usando decodificadores 2x4

byte x; byte y; byte z;
byte s0; byte s1; byte s2; byte s3;
byte z0; byte z1; byte z2; byte z3; byte z4; byte z5; byte z6; byte z7;

void decod2x4(byte b, byte a, byte en){
  s0 = !(( !en && !b ) && !a ); // 00
  s1 = !(( !en && !b ) && a ); // 01
  s2 = !(( !en && b ) && !a ); // 10
  s3 = !(( !en && b ) && a ); // 11
}

void mostra_decod3x8b(){
  if (z) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (y) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (x) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (z7) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (z6) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (z5) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (z4) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (z3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (z2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (z1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (z0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Decodificador 3x8 com decodificadores 2x4");
  Serial.println("| Z | Y | X || Z7 | Z6 | Z5 | Z4 | Z3 | Z2 | Z1 | Z0
  |");

  Serial.println("-----------------------------------------------------
  ------");
  for (z = 0; z <= 1; z++){
    for (y = 0; y <= 1; y++){
      for (x = 0; x <= 1; x++){
      decod2x4(y,x,z);
```

```
      z0 = s0;
      z1 = s1;
      z2 = s2;
      z3 = s3;
      decod2x4(y,x,!z);
      z4 = s0;
      z5 = s1;
      z6 = s2;
      z7 = s3;
      mostra_decod3x8b();
    }
  }
}

} // fim do ‘setup’

void loop(){
  // nada a fazer aqui!
} // fim do ‘loop’
```

7.2. Codificadores

7.2.1. Objetivos

1. Investigar os circuitos de codificadores,
2. Observar seu funcionamento e obter sua tabela verdade,
3. Conhecer os circuitos de codificadores.

7.2.2. Informação preliminar

Um codificador é um circuito que altera um conjunto de sinais em um código. Como um componente combinacional padrão, um codificador é quase como o inverso de um decodificador onde ele codifica um dado de entrada de 2^n bits em um código de *n* bits. Como mostrado pela forma geral de um codificador *m*-para-*n* na Figura 7.11, o codificador possui $m = 2^n$ linhas de entrada e *n* linhas de saída. Para entradas ativas em alto, a operação do codificador é tal que exatamente uma das linhas de entrada deve ter um "1", enquanto as linhas de entrada restantes devem ter "0". A saída é o valor binário do índice da linha de entrada que possui "1". É assumido que apenas uma linha de entrada pode ser "1". Os codificadores são usados para reduzir o número de bits necessários para representar alguns dados como no armazenamento de dados ou na transmissão de dados. Os codificadores também são usados em um sistema com 2^n dispositivos de entrada, cada um dos quais pode precisar solicitar serviço. Uma linha de entrada está conectada a um dispositivo de entrada. O dispositivo de entrada que solicita o serviço declarará a linha de entrada que está conectada a ele. O valor de saída correspondente de n bits indicará ao sistema qual dos 2 dispositivos está solicitando serviço. No entanto, isso só funciona corretamente se for garantido que apenas um dos 2^n dispositivos solicitará o serviço a qualquer momento. Se dois ou mais dispositivos solicitarem serviço ao mesmo tempo, a saída estará incorreta. Para resolver esse problema, uma prioridade é atribuída a cada uma das linhas de entrada para que, quando várias solicitações forem feitas, o codificador produza o valor de índice da linha de entrada com a prioridade mais alta. Este codificador modificado é conhecido como um codificador de prioridade. Neste experimento, estamos preocupados com os codificadores de prioridade. Embora, não mostrado na Figura 7.19, o codificador de prioridade pode ter uma linha de habilitação, "E", para habilitá-lo. Quando o codificador de prioridade é desabilitado com "E" definido como "0" (para entrada de habilitação ativa em alto "E"), todas as linhas de saída terão "0" (para saídas ativas em alto). Quando o codificador de prioridade está habilitado, as linhas de saída emitem a representação de dados binários do sinal de entrada de prioridade mais alta ativada (definido como 1 para ativo em alto).

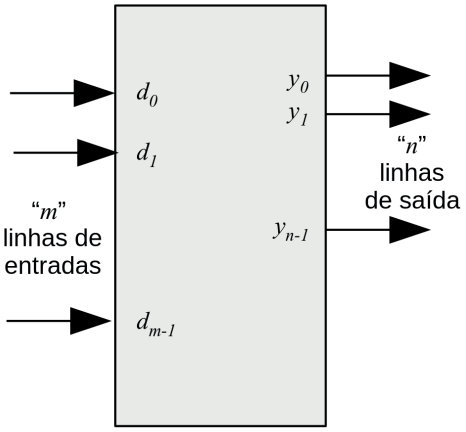


Figura 7.19 – A forma geral de um codificador *m*-para-*n*, onde $m = 2^n$.

Como exemplo, vamos considerar um codificador 8x3. A Tabela 7.5 mostra a tabela verdade de um codificador 8x3, enquanto a Figura 7.20 mostra o diagrama lógico deste codificador. Neste codificador existem oito entradas ativas em alto, a saber, D0, D1, D2, D3, D4, D5, D6 e D7, e há três saídas ativas em alto, a saber, X, Y e Z. Você deve notar que a tabela de verdade não é uma tabela completa. Como temos oito entradas, uma tabela completa deve ter 2^8 ou 256 entradas. Essas outras combinações correspondem a todas as entradas ou entradas zero com mais de uma entrada definida como 1. Dizemos que essas outras combinações são "Não permitidas". Se mais de uma única entrada for definida como "1", o codificador não poderá produzir saídas corretas. Como discutido acima, este problema é resolvido por meio de codificadores de prioridade. Portanto, vamos nos concentrar em codificadores de prioridade. Vamos considerar um tal codificador IC: 74LS148.

Entradas								Saídas		
D0	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**X**	**Y**	**Z**
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Tabela 7.5 – A tabela verdade de um codificador 8x3 com entradas ativas em alto e saídas ativas em alto.

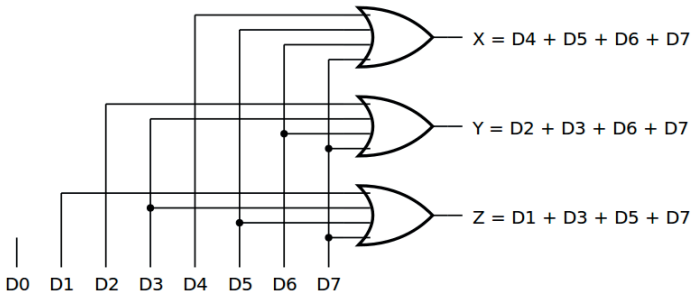
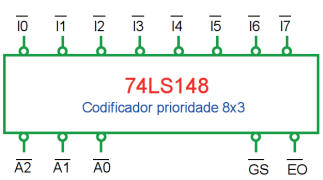


Figura 7.20 – O diagrama lógico de um codificador 8x3 com entradas ativas em alto e saídas ativas em alto.

7.2.3. Circuito Integrado TTL 74LS148 – Codificador de prioridade 8x3

A Figura 7.21 mostra o símbolo esquemático, o diagrama lógico e a tabela funcional do codificador de prioridade 8x3 CI 74LS148. O 74LS148 fornece codificação prioritária das entradas para garantir que apenas a linha de dados de maior ordem seja codificada. O 74LS148 codifica oito linhas de dados (I7’, I6’, I5 ‘, I4’, I3’, I2’, I1 ‘e I0’) para três linhas (A2’, A1’ e A0 ‘) (4-2-1) binário (octal). Todas as entradas e saídas do 74LS148 são organizadas como ativas em baixo. A entrada I7’ (respectivamente, I0’) tem a prioridade mais alta (mais baixa). Além das saídas ativas em baixo A2 (MSB), A1 e A0 (LSB), há saídas ativas em baixo GS’ (saída de seleção de grupo) e EO’ (habilitação de saídas). Se uma das entradas (I7’, I6’, I5’, I4’, I3’, I2’, I1’ou I0’) for baixa, então GS’ = 0. Isso indica que pelo menos uma das entradas está ativa. Se nenhuma das entradas estiver ativa, então EO’= 0. Isso indica que nenhuma das entradas está ativa. Ao fornecer circuitos em cascata (habilitação de entrada EI e habilitação de saída EO), a expansão octal é permitida sem a necessidade de circuitos externos.



Símbolo esquemático

Entradas									Saídas				
EI	I 0	I 1	I 2	I 3	I 4	I 5	I 6	I 7	A2	A1	A0	GS	EO
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
0	x	x	x	x	x	0	1	1	0	1	0	0	1
0	x	x	x	x	0	1	1	1	0	1	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	0	1
0	x	x	0	1	1	1	1	1	1	0	1	0	1
0	x	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

Tabela verdade
(x = não importa)

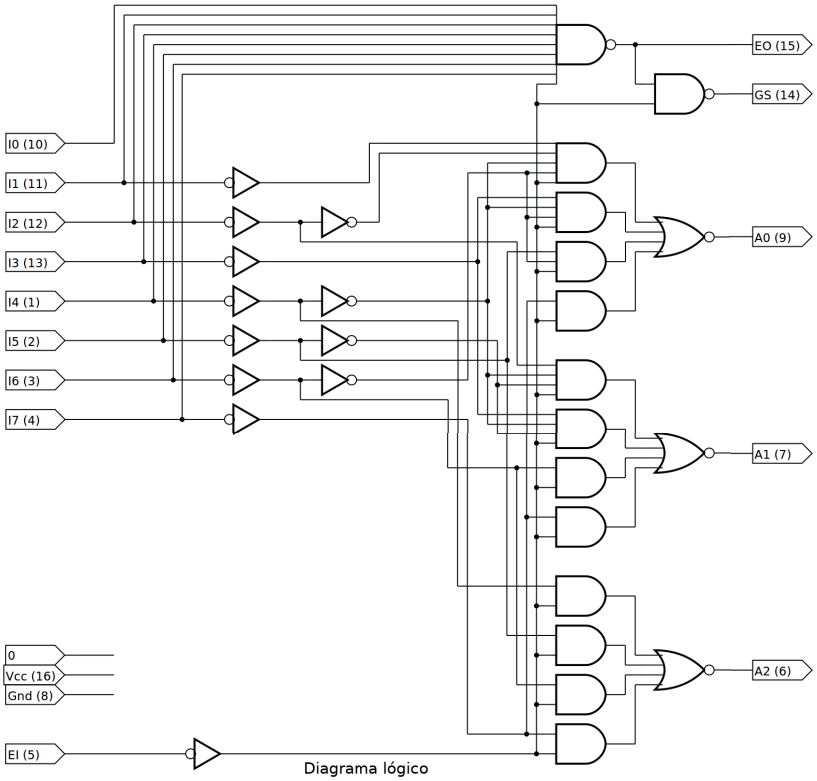


Figura 7.21 – O diagrama lógico, a tabela funcional e o símbolo esquemático do codificador de prioridade 8x3 CI 74LS148.

7.2.4. Exame do circuito codificador de prioridade 8x3 74LS148

Esquemas:

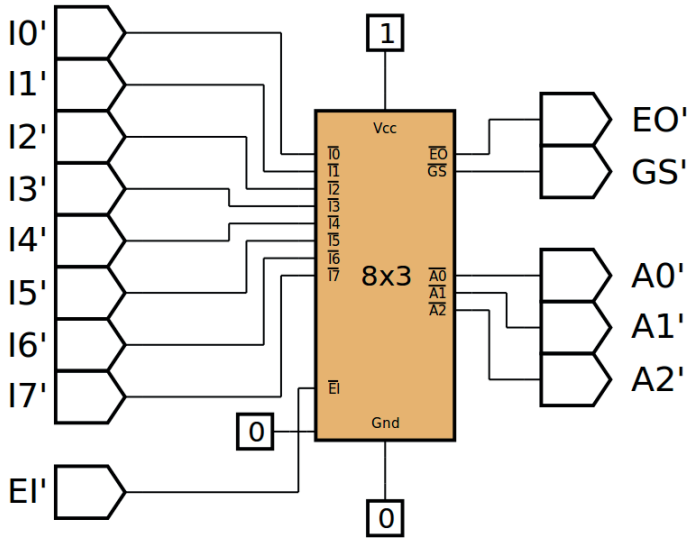


Figura 7.22 – Diagrama esquemático.

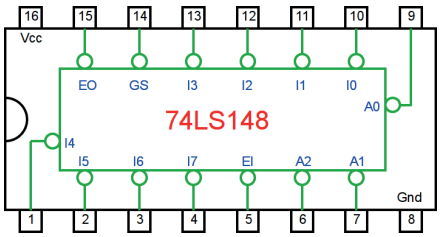


Figura 7.23– Circuito integrado TTL.

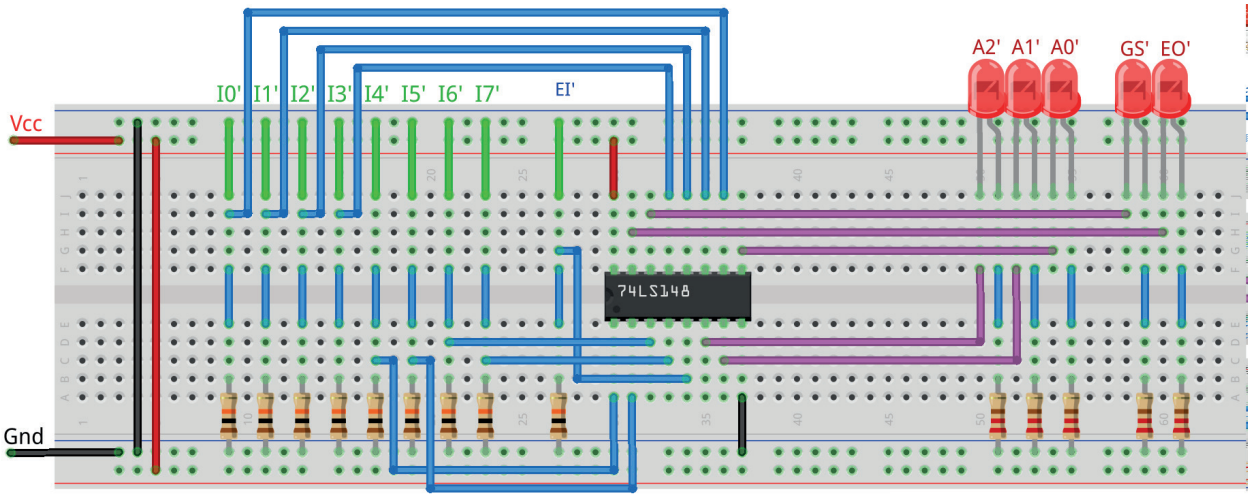


Figura 7.24 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 7.24.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 7.6, use os fios verdes como segue:
   a) Se o valor da variável for "0" o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for "1" o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 7.6, considere:
   a) Se o LED estiver apagado então preencher com "0".
   b) Se o LED estiver aceso então preencher com "1".
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 7.22. Compare os resultados com a tabela 7.6.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 7.6.

Tabelas de dados:

Entradas									Saídas				
EI	I0	I1	I2	I3	I4	I5	I6	I7	A2	A1	A0	GS	EO
1	x	x	x	x	x	x	x	x					
0	1	1	1	1	1	1	1	1					
0	x	x	x	x	x	x	x	0					
0	x	x	x	x	x	x	0	1					
0	x	x	x	x	x	0	1	1					
0	x	x	x	x	0	1	1	1					
0	x	x	x	0	1	1	1	1					
0	x	x	0	1	1	1	1	1					
0	x	0	1	1	1	1	1	1					
0	0	1	1	1	1	1	1	1					

Tabela 7.6 (X = não importa)

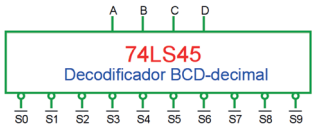
Programa para o Arduíno:

```
// Codificador de prioridade 8x3

int i; byte en;
byte p0; byte p1; byte p2; byte p3; byte p4; byte p5; byte p6; byte p7;
byte a2; byte a1; byte a0; byte gs; byte eo;
byte x0; byte x1; byte x2; byte x3; byte x4; byte x5; byte x6; byte x7;
byte x8;

void ativa_entrada(byte x){
  p0 = 1; p1 = 1; p2 = 1; p3 = 1; p4 = 1; p5 = 1; p6 = 1; p7 = 1;
  switch (x){
    case 0: p0 = 0; break;
    case 1: p1 = 0; break;
    case 2: p2 = 0; break;
    case 3: p3 = 0; break;
    case 4: p4 = 0; break;
    case 5: p5 = 0; break;
    case 6: p6 = 0; break;
    case 7: p7 = 0; break;
    defaut: break;
  }
}

void codec8x3(byte ei, byte i0, byte i1, byte i2, byte i3, byte i4,
byte i5, byte i6, byte i7){
  x0 = ((((((((!ei&&!i0)&&i1)&&i2)&&i3)&&i4)&&i5)&&i6)&&i7);
  x1 = (((((((!ei&&!i1)&&i2)&&i3)&&i4)&&i5)&&i6)&&i7);
  x2 = ((((((!ei&&!i2)&&i3)&&i4)&&i5)&&i6)&&i7);
  x3 = (((((!ei&&!i3)&&i4)&&i5)&&i6)&&i7);
  x4 = ((((!ei&&!i4)&&i5)&&i6)&&i7);
  x5 = (((!ei&&!i5)&&i6)&&i7);
  x6 = ((!ei&&!i6)&&i7);
  x7 = (!ei&&!i7);
  x8 = ((((((((!ei&&i0)&&i1)&&i2)&&i3)&&i4)&&i5)&&i6)&&i7);
  a2 = (((((ei||x8)||x3)||x2)||x1)||x0);
  a1 = (((((ei||x8)||x5)||x4)||x1)||x0);
  a0 = (((((ei||x8)||x6)||x4)||x2)||x0);
  gs = (ei||x8);
  eo = !x8;
}

void mostra_codec8x3(){
  if (en) Serial.print("| 1 "); else Serial.print("| 0 ");
```

```
  Serial.print("|");
  if (p0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (p1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (p2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (p3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (p4) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (p5) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (p6) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (p7) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (a2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (gs) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (eo) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void codecx(){
  for (i = 8; i >= 0; i--){
    ativa_entrada(i);
    codec8x3(en,p0,p1,p2,p3,p4,p5,p6,p7);
    mostra_codec8x3();
  }
}
void setup(){
  Serial.begin(9600);
  Serial.println("Codificador de prioridade 8x3");
  Serial.println("| EI || I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7 || A2 |
  A1 | A0 || GS | EO |");

  Serial.println("-------------------------------------------------------
  ----------------------");
  en = 0;
  codecx();
  Serial.println("");
  en = 1;
  codecx();
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

7.2.5. Circuito Integrado TTL 74LS45 – Decodificador BCD para decimal

A Figura 7.25 mostra o símbolo esquemático, o diagrama lógico e a tabela verdade do CI decodificador BCD para decimal 74LS45. O 74LS45 decodifica uma das 10 linhas de saída (S0, S1, S2, S3, S4, S5, S6, S7, S8, S9), com base nas condições nas quatro entradas de seleção binárias (D, C, B, A). As saídas do 74LS45 são em baixo ativo. Esse circuito não possui entradas de habilitação.



Símbolo esquemático

Entradas				Saídas									
A	B	C	D	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabela verdade

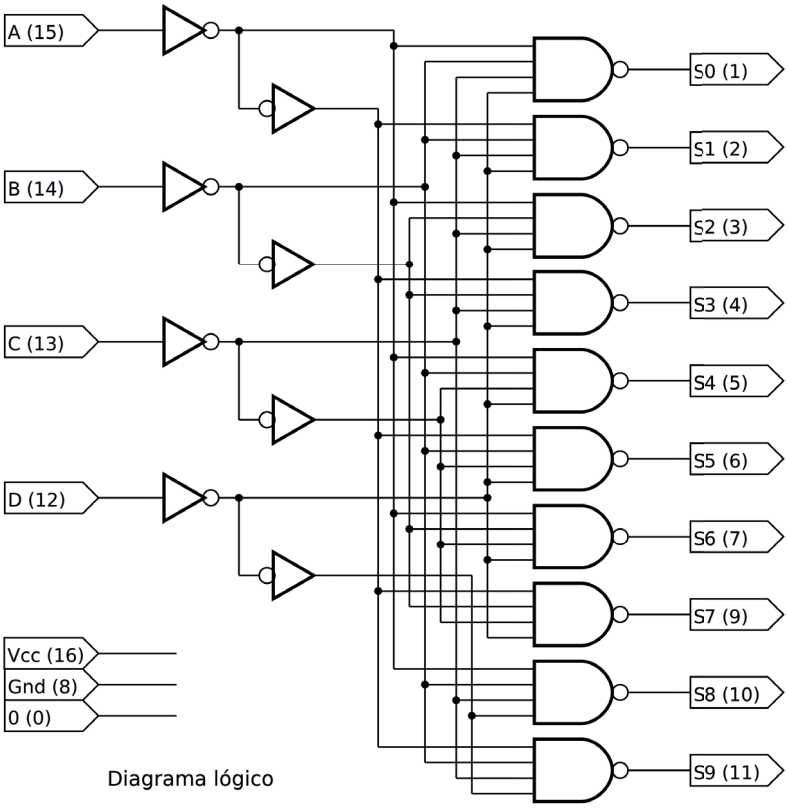


Diagrama lógico

Figura 7.25 – O símbolo esquemático, o diagrama lógico e a tabela verdade do CI decodificador BCD para decimal 74LS45.

O circuito integrado da linha CMOS equivalente é o 4028, mas com pinagem diferente. As saídas ativas são em nível lógico “1” (alto ativo).

7.2.6. Exame do circuito decodificador BCD para decimal 74LS45

Esquemas:

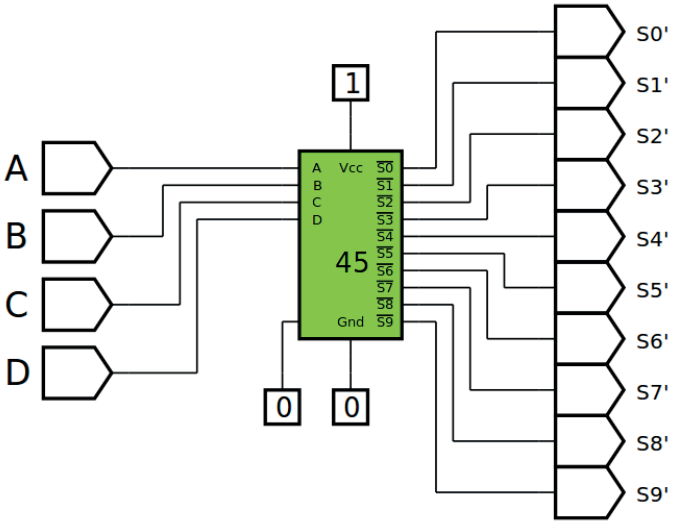


Figura 7.26 – Diagrama esquemático.

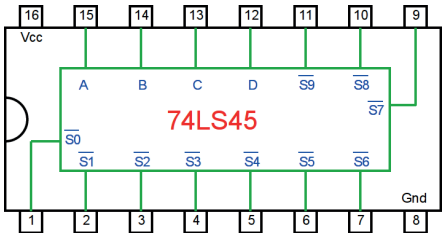


Figura 7.27– Circuito integrado TTL.

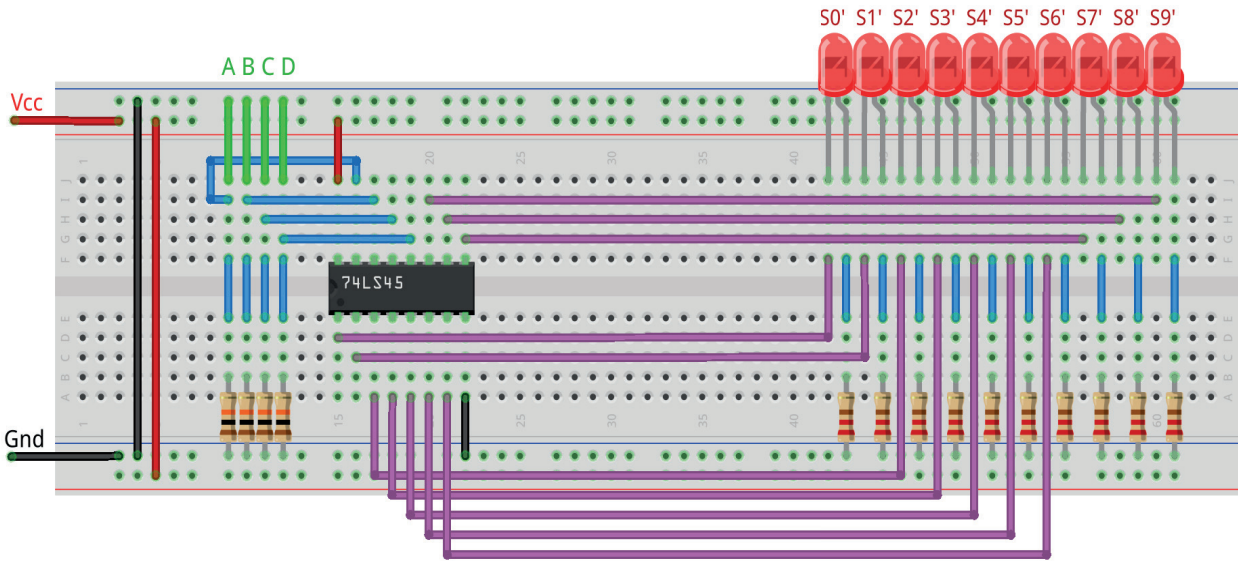


Figura 7.28 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 7.28.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 7.7, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 7.7, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 7.26. Compare os resultados com a tabela 7.7.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 7.7.

Tabelas de dados:

Entradas				Saídas									
A	B	C	D	S0’	S1’	S2’	S3’	S4’	S5’	S6’	S7’	S8’	S9’
0	0	0	0										
0	0	0	1										
0	0	1	0										
0	0	1	1										
0	1	0	0										
0	1	0	1										
0	1	1	0										
0	1	1	1										
1	0	0	0										
1	0	0	1										
1	0	1	0										
1	0	1	1										
1	1	0	0										
1	1	0	1										
1	1	1	0										
1	1	1	1										

Tabela 7.7

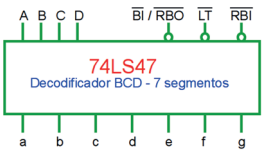
Programa para o Arduíno:

```
// Decodificador BCD para decimal

byte a; byte b; byte c; byte d;
byte s0; byte s1; byte s2; byte s3; byte s4;
byte s5; byte s6; byte s7; byte s8; byte s9;

void decod_bcd_dec(byte aa, byte bb, byte cc, byte dd){
  s0 = !(((  !aa && !bb ) && !cc ) && !dd ); // 0 - 0000
  s1 = !(((  !aa && !bb ) && !cc ) && dd ); // 1 - 0001
  s2 = !(((  !aa && !bb ) && cc ) && !dd ); // 2 - 0010
  s3 = !(((  !aa && !bb ) && cc ) && dd ); // 3 - 0011
  s4 = !(((  !aa && bb ) && !cc ) && !dd ); // 4 - 0100
  s5 = !(((  !aa && bb ) && !cc ) && dd ); // 5 - 0101
  s6 = !(((  !aa && bb ) && cc ) && !dd ); // 6 - 0110
  s7 = !(((  !aa && bb ) && cc ) && dd ); // 7 - 0111
  s8 = !(((  aa && !bb ) && !cc ) && !dd ); // 8 - 1000
  s9 = !(((  aa && !bb ) && !cc ) && dd ); // 9 - 1001
}

void mostra_bcd_dec(){
  if (a) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (b) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (c) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (d) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (s0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s4) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s5) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s6) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s7) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s8) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s9) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
```

```
  Serial.begin(9600);
  Serial.println("Decodificador BCD para decimal");

  Serial.println("| A | B | C | D || S0 | S1 | S2 | S3 | S4 | S5 | S6 |
  S7 | S8 | S9 |");
  Serial.println("-----------------------------------------------------
  -----------------------");
  for (a = 0; a <= 1; a++){
    for (b = 0; b <= 1; b++){
      for (c = 0; c <= 1; c++){
        for (d = 0; d <= 1; d++){
          decod_bcd_dec(a,b,c,d);
          mostra_bcd_dec();
      }
    }
  }
}
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

7.2.7. Circuito Integrado TTL 74LS47 – Decodificador BCD para sete segmentos

A Figura 7.29 mostra o símbolo esquemático, o diagrama lógico e a tabela verdade do CI decodificador BCD para sete segmentos 74LS47. O 74LS47 decodifica todas as 7 linhas de saída (a, b, c, d, e, f, g), com base nas condições nas quatro entradas de seleção binárias (D, C, B, A). As saídas do 74LS47 são em baixo ativo. O sinal de entrada LT' serve para teste dos elementos de saída. Quando em nível lógico ativo "0" todas as saídas são colocadas em "1" ativo. O sinal de entrada RBI' serve para desligar todos os elementos de saída, somente se todos os sinais de entrada forem "0". Quando em nível lógico ativo "0" todas as saídas são colocadas em "0". O pino BI' / RBO' funciona como entrada (BI') ou como saída (RBO'). Quando um "0" ativo é colocado em BI', todas as saídas são colocadas em "0". RBO' copia o sinal de RBI'.



Símbolo esquemático

Entradas				Saídas						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0
1	0	1	0	1	1	1	0	0	0	1
1	0	1	1	0	0	1	1	0	0	1
1	1	0	0	1	0	1	1	1	0	0
1	1	0	1	0	1	1	0	1	0	0
1	1	1	0	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1

Tabela verdade

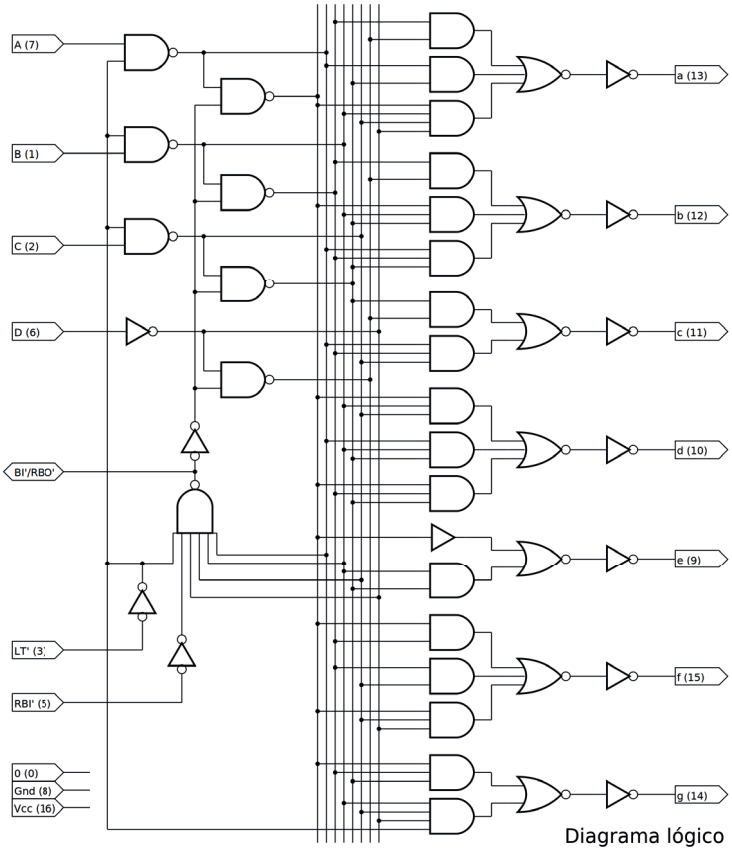


Diagrama lógico

Figura 7.29 – O símbolo esquemático, o diagrama lógico e a tabela verdade do CI decodificador BCD para sete segmentos 74LS47.

O circuito integrado da linha CMOS equivalente é o 4511, com pinagem semelhante, a exceção dos pinos 4 com a função de BI (blank input) e o pino 5 com a função de LE (latch enable). Quando LE é colocado em nível lógico "1" as saídas mantém o último valor, independente das entradas. As saídas ativas são em nível lógico "1" (alto ativo). Outros CI CMOS com função semelhante são o 4513 e o 4543.

7.2.8. Exame do circuito decodificador BCD para sete segmentos 74LS47

Esquemas:

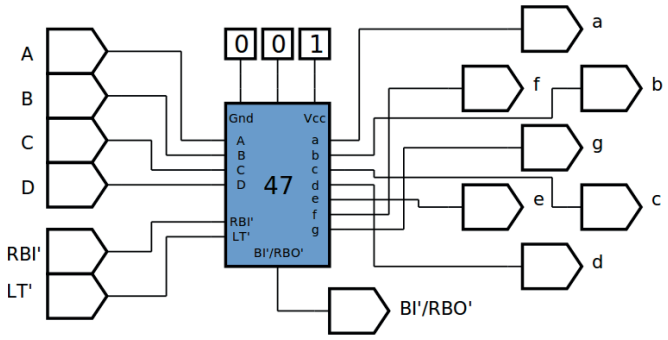


Figura 7.30 – Diagrama esquemático.

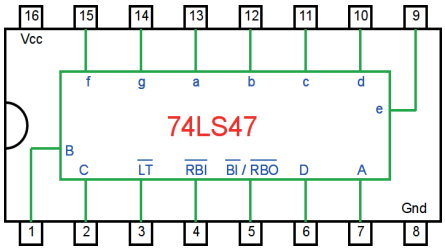


Figura 7.31– Circuito integrado TTL.

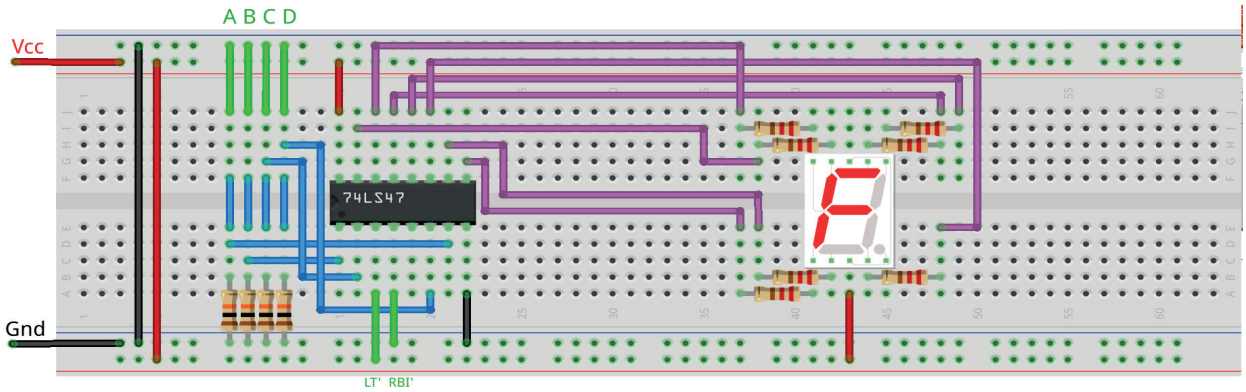


Figura 7.32 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 7.32.
   a) Coloque os componentes (resistores, “display” e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) ao display.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 7.8, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 7.8, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 7.30. Compare os resultados com a tabela 7.8.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 7.8.

Tabelas de dados:

Entradas				Saídas							
A	B	C	D	a	b	c	d	e	f	g	disp
0	0	0	0								
0	0	0	1								
0	0	1	0								
0	0	1	1								
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	1	1								
1	0	0	0								
1	0	0	1								
1	0	1	0								
1	0	1	1								
1	1	0	0								
1	1	0	1								
1	1	1	0								
1	1	1	1								

Tabela 7.8

Programa para o Arduíno:

```
// Decodificador BCD para 7 segmentos

byte a; byte b; byte c; byte d;
byte m0; byte m1; byte m2; byte m3; byte m4; byte m5; byte m6; byte m7;
byte m8; byte m9; byte m10; byte m11; byte m12; byte m13; byte m14;
byte m15;
byte a7; byte b7; byte c7; byte d7; byte e7; byte f7; byte g7;

void decod_bcd_7seg(byte aa, byte bb, byte cc, byte dd){
  m0 = ((( !aa && !bb ) && !cc ) && !dd ); // 0 - 0000
  m1 = ((( !aa && !bb ) && !cc ) && dd ); // 1 - 0001
  m2 = ((( !aa && !bb ) && cc ) && !dd ); // 2 - 0010
  m3 = ((( !aa && !bb ) && cc ) && dd ); // 3 - 0011
  m4 = ((( !aa && bb ) && !cc ) && !dd ); // 4 - 0100
  m5 = ((( !aa && bb ) && !cc ) && dd ); // 5 - 0101
  m6 = ((( !aa && bb ) && cc ) && !dd ); // 6 - 0110
  m7 = ((( !aa && bb ) && cc ) && dd ); // 7 - 0111
  m8 = ((( aa && !bb ) && !cc ) && !dd ); // 8 - 1000
  m9 = ((( aa && !bb ) && !cc ) && dd ); // 9 - 1001
  m10 = ((( aa && !bb ) && cc ) && !dd ); // 10 - 1010
  m11 = ((( aa && !bb ) && cc ) && dd ); // 11 - 1011
  m12 = ((( aa && bb ) && !cc ) && !dd ); // 12 - 1100
  m13 = ((( aa && bb ) && !cc ) && dd ); // 13 - 1101
  m14 = ((( aa && bb ) && cc ) && !dd ); // 14 - 1110
  m15 = ((( aa && bb ) && cc ) && dd ); // 15 - 1111
  a7 = ((((((m1||m4)||m6)||m10)||m12)||m14)||m15);
  b7 = (((((m5||m6)||m10)||m13)||m14)||m15);
  c7 = ((((((m2||m10)||m11)||m12)||m13)||m14)||m15);
  d7 = ((((((m1||m4)||m7)||m9)||m11)||m12)||m15);
  e7 = ((((((((m1||m3)||m4)||m5)||m7)||m9)||m12)||m13)||m15);
  f7 = ((((m1||m2)||m3)||m7)||m15);
  g7 = (((((m0||m1)||m7)||m10)||m11)||m15);
}

void mostra_bcd_7seg(){
if (a) Serial.print("| 1 "); else Serial.print("| 0 ");
if (b) Serial.print("| 1 "); else Serial.print("| 0 ");
if (c) Serial.print("| 1 "); else Serial.print("| 0 ");
if (d) Serial.print("| 1 "); else Serial.print("| 0 ");
```

```
Serial.print("|");
if (a7) Serial.print("| 1 "); else Serial.print("| 0 ");
if (b7) Serial.print("| 1 "); else Serial.print("| 0 ");
if (c7) Serial.print("| 1 "); else Serial.print("| 0 ");
if (d7) Serial.print("| 1 "); else Serial.print("| 0 ");
if (e7) Serial.print("| 1 "); else Serial.print("| 0 ");
if (f7) Serial.print("| 1 "); else Serial.print("| 0 ");
if (g7) Serial.print("| 1 "); else Serial.print("| 0 ");
Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Decodificador BCD para 7 segmentos");
  Serial.println("| A | B | C | D || a | b | c | d | e | f | g |");

  Serial.println("-----------------------------------------------------
  -----");
  for (a = 0; a <= 1; a++){
    for (b = 0; b <= 1; b++){
      for (c = 0; c <= 1; c++){
        for (d = 0; d <= 1; d++){
          decod_bcd_7seg(a,b,c,d);
          mostra_bcd_7seg();
        }
      }
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

Capítulo 8. Circuitos lógicos combinacionais: multiplexadores e demultiplexadores

8.1. Multiplexadores

8.1.1. Objetivos

1. Investigar os circuitos multiplexadores (MUX),
2. Observar seu funcionamento e obter suas tabelas verdade,
3. Conhecer os circuitos integrados multiplexadores (CIs).

8.1.2. Informação preliminar

Como um componente combinacional padrão, o multiplexador, abreviado como MUX, permite a seleção de um sinal de entrada entre *n* sinais, onde $n>1$, e é uma potência de dois. Selecionar linhas conectadas ao multiplexador determinam qual sinal de entrada é selecionado e passado para a saída do multiplexador. Como pode ser visto na figura 8.1, em geral, um multiplexador *n-para-1* tem *n* linhas de entrada de dados, *m* linhas de seleção onde $m=log_2 n$, ou seja, $2^m=n$, e uma linha de saída. Como mostrado na figura 8.1, além das outras entradas, o multiplexador possui uma linha de habilitação, *E*, para habilitá-lo. Quando o multiplexador é desativado com *E* definido como *"0"* (para entrada de habilitação *E* ativa em alto), nenhum sinal de entrada é selecionado e passado para a saída. Quando o multiplexador é ativado com *E* definido como *"1"* (para entrada de habilitação *E* ativa em alto), um sinal de entrada é selecionado e passado para a saída com base nos valores lógicos aplicados às entradas de seleção.

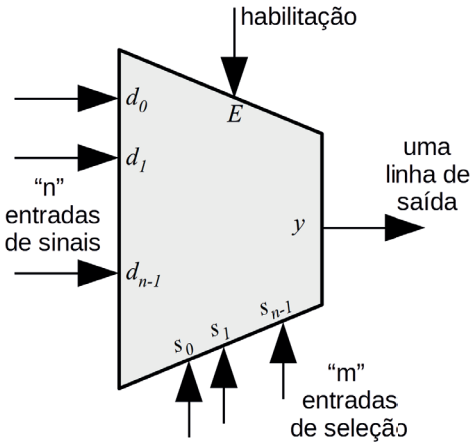
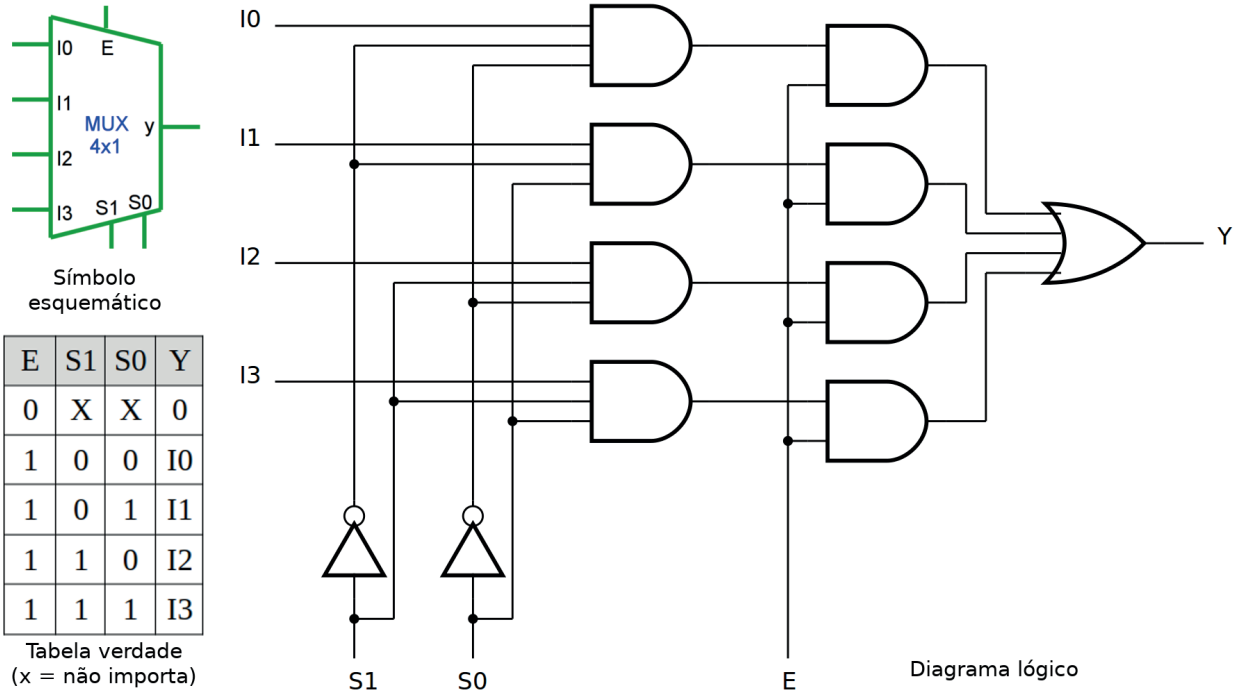


Figura 8.1 – A forma geral de um multiplexador de *n-para-1*, onde $n=2^m$.

Em geral, os multiplexadores são produzidos em circuitos integrados como multiplexadores 2x1, 4x1, 8x1 e 16x1. No caso de um multiplexador de *2-para-1* (2x1), um valor lógico de "0" aplicado à entrada de seleção s_0 conectaria d_0 à saída *y*, enquanto um valor lógico de "1" aplicado à entrada de seleção s_0 se conectaria d_1 para a saída *y*. Em outras palavras, o valor binário expresso nos pinos do seletor determina o pino de entrada selecionado. Em multiplexadores maiores, o

número de pinos do seletor é igual a [$log_2(n)$], onde n é o número de entradas. Portanto, para um multiplexador 4x1 (respectivamente 8x1, 16x1), o número de entradas de seleção é 2 (respectivamente 3, 4).

A figura 8.2 mostra o símbolo esquemático, o diagrama lógico e a tabela verdade de um MUX 4x1 com entrada de habilitação (E) ativa em alto. Neste multiplexador, a entrada de habilitação E ativa em alto, entradas de seleção S_1, S_0, os sinais de entrada I_0, I_1, I_2 e I_3, e a saída Y são todas variáveis booleanas. Quando este multiplexador é desativado com E definido como 0, nenhum sinal de entrada é selecionado e passado para a saída. Quando este multiplexador é habilitado com E ajustado para 1: se $S_1S_0 = 00$, (respectivamente, 01, 10, 11), então o sinal de entrada I_0 (respectivamente, I_1, I_2, I_3) é selecionado e passado para a saída Y.



E	S1	S0	Y
0	X	X	0
1	0	0	I0
1	0	1	I1
1	1	0	I2
1	1	1	I3

Tabela verdade
(x = não importa)

Figura 8.2 – O símbolo esquemático, o diagrama lógico e a tabela verdade de um MUX 4x1 com entrada de habilitação (E) ativa em alto.

Os multiplexadores também são chamados de seletores de dados. Quando os dados a serem selecionados têm mais de um bit, podemos construir um multiplexador para selecionar esses dados. Por exemplo, se tivermos dois dados de quatro bits A ($A_4A_3A_2A_1$) e B ($B_4B_3B_2B_1$) a serem selecionados, podemos usar um multiplexador como mostrado na figura 8.3. Esse multiplexador é chamado de multiplexador 2x1 quádruplo com entrada de habilitação (E) ativa em baixo. Quando este multiplexador está habilitado (isto é, quando $E = 0$) se $S = 0$ então a saída Y ($Y_4Y_3Y_2Y_1$) = A ($A_4A_3A_2A_1$) e se $S = 1$ então a saída Y ($Y_4Y_3Y_2Y_1$) = B ($B_4B_3B_2B_1$).

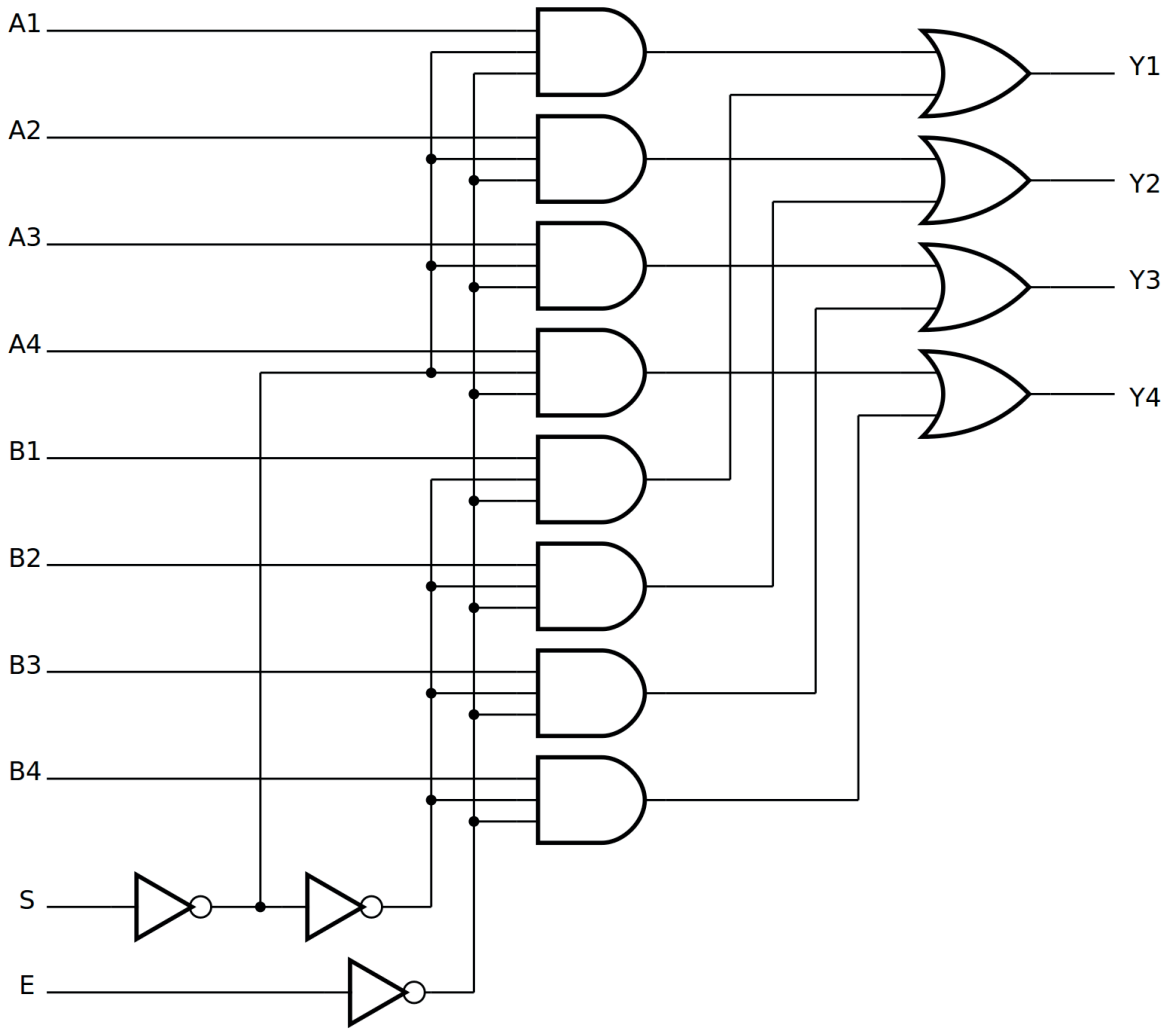
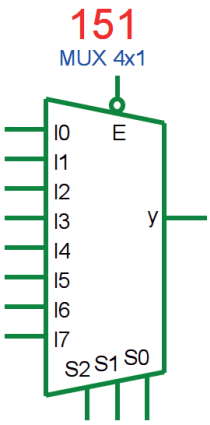


Figura 8.3 – Um multiplexador 2x1 quádruplo com entrada de habilitação (*E*) ativa em baixo.

8.1.3. Circuito Integrado TTL 74LS151 – Multiplexador 8x1

A figura 8.4 mostra o símbolo esquemático, o diagrama lógico e a tabela verdade do CI 74LS151 MUX 8x1. Neste multiplexador, a entrada de habilitação E ativa em baixo, as entradas de seleção S_2, S_1, S_0, os sinais de entrada *"I0"*, *"I1"*, *"I2"*, *"I3"*, *"I4"*, *"I5"*, *"I6"* e *"I7"*, e a saída *Y* (e seu complemento *Y*) são todas variáveis booleanas.



Símbolo esquemático

$\overline{E}$	S2	S1	S0	I0	I1	I2	I3	I4	I5	I6	I7	$\overline{Y}$	Y
1	X	X	X	X	X	X	X	X	X	X	X	1	0
0	0	0	0	0	X	X	X	X	X	X	X	1	0
0	0	0	0	1	X	X	X	X	X	X	X	0	1
0	0	0	1	X	0	X	X	X	X	X	X	1	0
0	0	0	1	X	1	X	X	X	X	X	X	0	1
0	0	1	0	X	X	0	X	X	X	X	X	1	0
0	0	1	0	X	X	1	X	X	X	X	X	0	1
0	0	1	1	X	X	X	0	X	X	X	X	1	0
0	0	1	1	X	X	X	1	X	X	X	X	0	1
0	1	0	0	X	X	X	X	0	X	X	X	1	0
0	1	0	0	X	X	X	X	1	X	X	X	0	1
0	1	0	1	X	X	X	X	X	0	X	X	1	0
0	1	0	1	X	X	X	X	X	1	X	X	0	1
0	1	1	0	X	X	X	X	X	X	0	X	1	0
0	1	1	0	X	X	X	X	X	X	1	X	0	1
0	1	1	1	X	X	X	X	X	X	X	0	1	0
0	1	1	1	X	X	X	X	X	X	X	1	0	1

Tabela verdade (x = não importa)

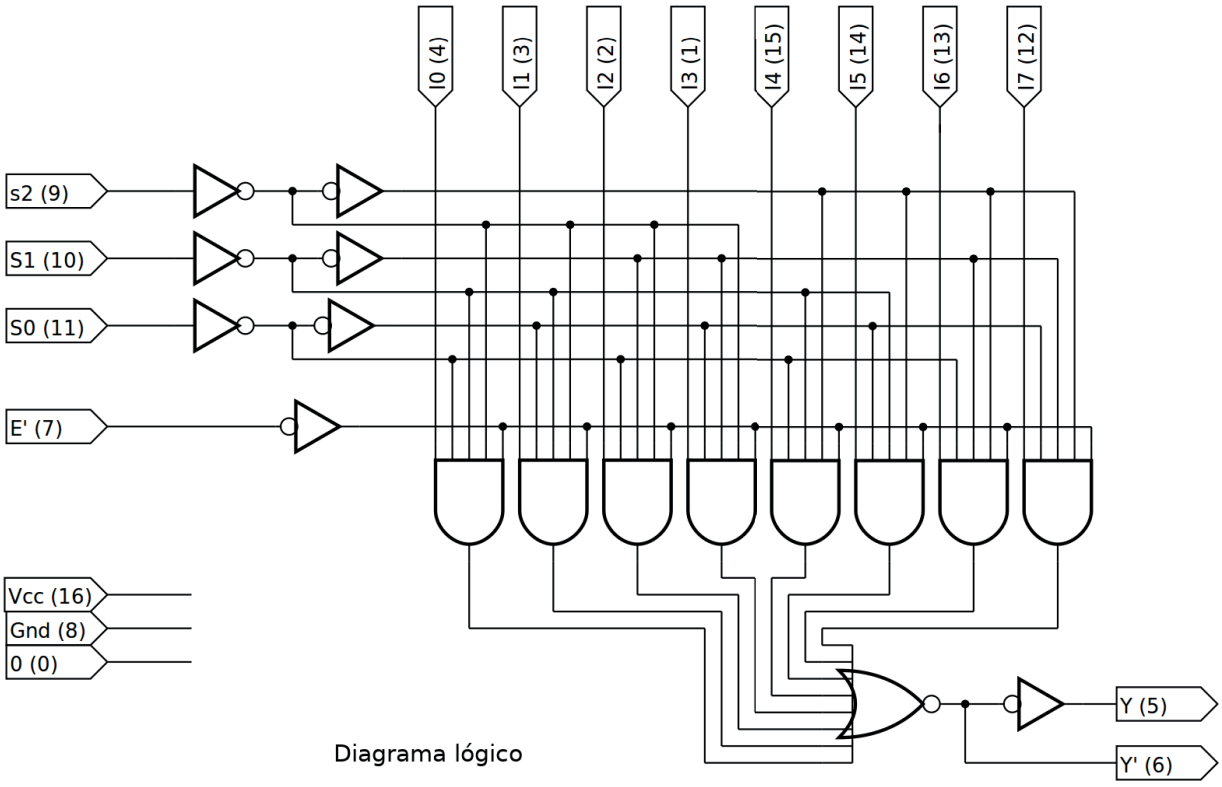


Diagrama lógico

Figura 8.4 – O diagrama lógico, o símbolo esquemático e a tabela verdade do multiplexador 74LS151 8x1.

8.1.4. Exame do CI 74LS151 – Multiplexador 8x1

Esquemas:

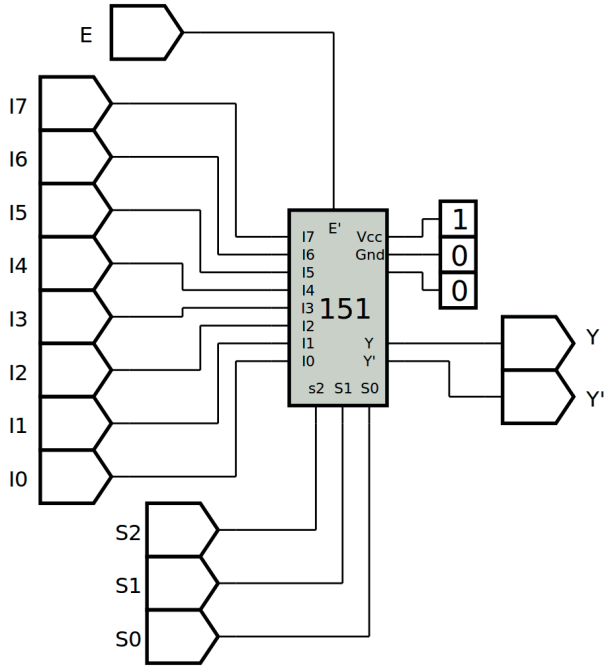


Figura 8.5 – Diagrama esquemático.

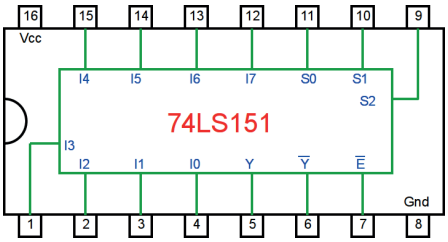


Figura 8.6 – Circuito integrado TTL.

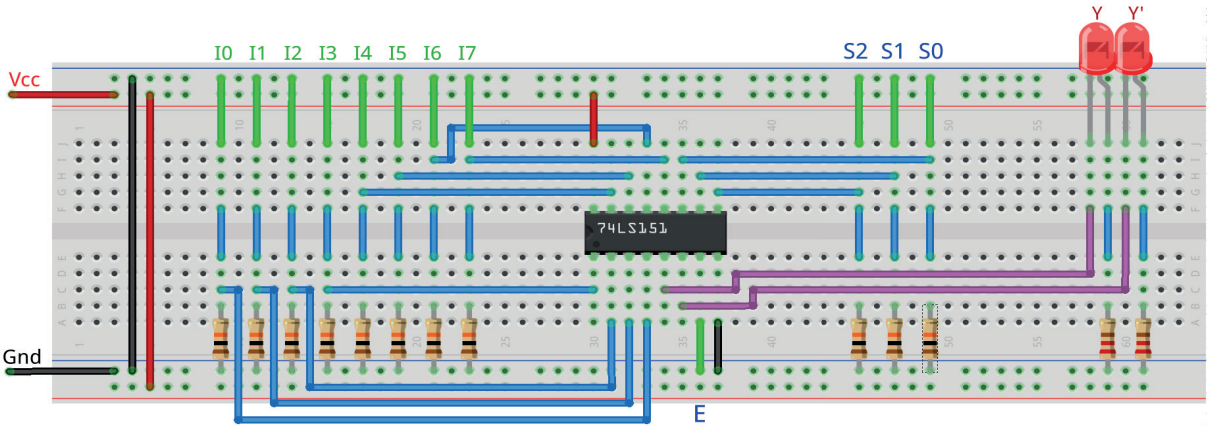


Figura 8.7 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 8.7.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 8.1, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 8.1, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 8.5. Compare os resultados com a tabela 8.1.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 8.1.

Tabelas de dados:

Seleção			Entradas								E'	Saídas			
S2	S1	S0	I0	I1	I2	I3	I4	I5	I6	I7		Y		Y'	
X	X	X	1	1	1	1	1	1	1	1	1				
0	0	0	0	0	1	1	1	0	1	1	0	I0		I0'	
0	0	0	1	0	1	1	1	0	1	1	0	I0		I0'	
0	0	1	0	1	0	1	0	1	0	1	0	I1		I1'	
0	0	1	0	0	0	1	0	1	0	1	0	I1		I1'	
0	1	0	0	0	1	0	0	0	0	0	0	I2		I2'	
0	1	0	0	0	0	0	0	0	0	0	0	I2		I2'	
0	1	1	0	0	1	0	0	0	1	1	0	I3		I3'	
0	1	1	0	0	1	1	0	0	1	1	0	I3		I3'	
1	0	0	0	1	1	1	1	1	1	1	0	I4		I4'	
1	0	0	0	1	1	1	0	1	1	1	0	I4		I4'	
1	0	1	1	1	1	0	1	1	1	0	0	I5		I5'	
1	0	1	1	1	1	0	1	0	1	0	0	I5		I5'	
1	1	0	1	1	0	0	1	1	1	0	0	I6		I6'	
1	1	0	1	1	0	0	1	1	0	0	0	I6		I6'	
1	1	1	1	0	1	0	1	0	1	0	0	I7		I7'	
1	1	1	1	0	1	0	1	0	1	1	0	I7		I7'	

Tabela 8.1

Programa para o Arduíno:

```
// multiplexador 8x1

byte m0; byte m1; byte m2; byte m3; byte m4; byte m5; byte m6; byte m7;
byte i0; byte i1; byte i2; byte i3; byte i4; byte i5; byte i6; byte i7;
byte p[] = {B00111011, B10111011, B01010101, B00010101, B00100000,
B00000000, B00100011, B00110011, B01111111, B01110111, B11101110,
B11101010, B11001110, B11001100, B10101010, B10101011};
byte y; byte q = 16; byte pi; byte a2; byte a1; byte a0; byte ax; byte
i;

void separa_bits(){
  i7 = pi & B00000001;
  i6 = pi & B00000010; i6 = i6>>1;
  i5 = pi & B00000100; i5 = i5>>2;
  i4 = pi & B00001000; i4 = i4>>3;
  i3 = pi & B00010000; i3 = i3>>4;
  i2 = pi & B00100000; i2 = i2>>5;
  i1 = pi & B01000000; i1 = i1>>6;
  i0 = pi & B10000000; i0 = i0>>7;
}

void mux8x1(byte s2, byte s1, byte s0){
  m0 = ((!s2&&!s1)&&!s0)&&i0);
  m1 = ((!s2&&!s1)&&s0)&&i1);
  m2 = ((!s2&&s1)&&!s0)&&i2);
  m3 = ((!s2&&s1)&&s0)&&i3);
  m4 = ((s2&&!s1)&&!s0)&&i4);
  m5 = ((s2&&!s1)&&s0)&&i5);
  m6 = ((s2&&s1)&&!s0)&&i6);
  m7 = ((s2&&s1)&&s0)&&i7);
  y = (((((((m0||m1)||m2)||m3)||m4)||m5)||m6)||m7);
}

void mostra_mux8x1(){
  if (a2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (i0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i1) Serial.print("| 1 "); else Serial.print("| 0 ");
```

```
  if (i2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i4) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i5) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i6) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i7) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (y) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Multiplexador 8x1");
  Serial.println("| S2 | S1 | S0 || I0 | I1 | I2 | I3 | I4 | I5 | I6 |
  I7 || Y |");

  Serial.println("-----------------------------------------------------
  ------------");
  i = 0;
  for (a2 = 0; a2 <= 1; a2++){
    for (a1 = 0; a1 <= 1; a1++){
      for (a0 = 0; a0 <= 1; a0++){
        for (ax = 0; ax <= 1; ax++){
          pi = p[i];
          separa_bits();
          mux8x1(a2,a1,a0);
          mostra_mux8x1();
          i = i + 1;
        }
    }
  }
}
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

8.1.5. Implementação de uma função booleana utilizando o CI 74LS151 (parte 1)

Esquemas:

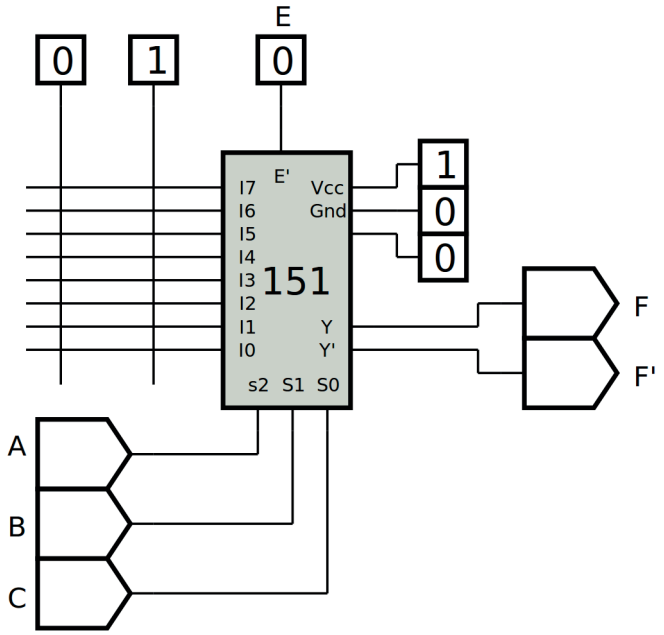


Figura 8.8 – Diagrama esquemático.

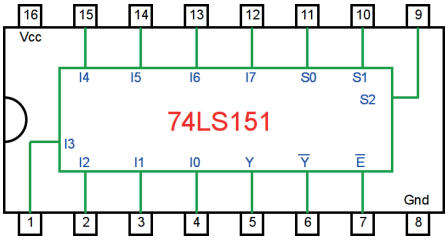


Figura 8.9 – Circuito integrado TTL.

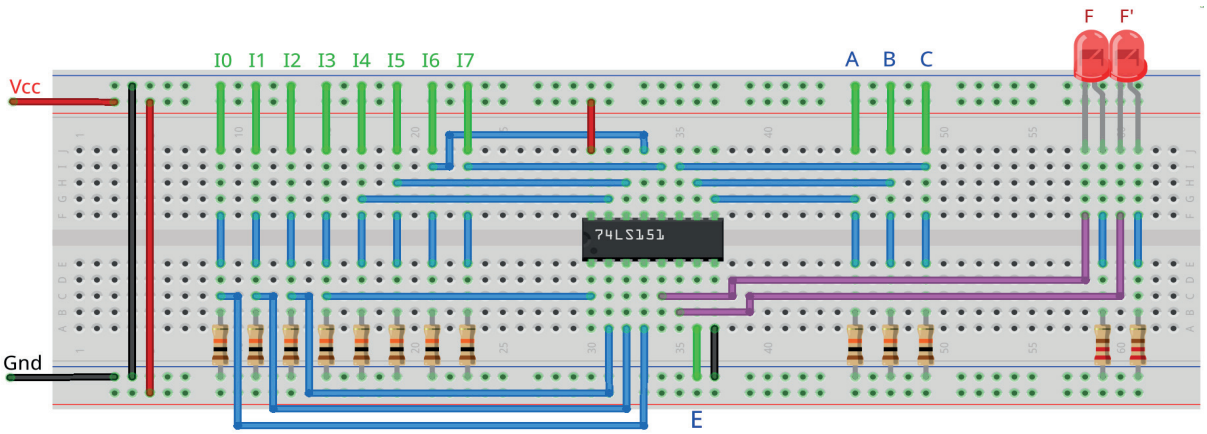


Figura 8.10 – Disposição dos componentes na placa de montagem.

Equação booleana:

F(A,B,C) = _____

Procedimento:

1. Complete a equação booleana para implementar a função: $F(A,B,C) = \Sigma_m(1,3,5,6)$.
2. Monte o circuito na placa de montagem, conforme figura 8.10.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
3. Ajuste as entradas de I0 até I7 conectando os fios verdes apropriadamente ou no barramento Vcc (1) ou no barramento Gnd (0) de acordo com a função booleana F(A,B,C).
4. Para o preenchimento da tabela 8.2, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
5. Para o preenchimento das colunas de saída na tabela 8.2, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
6. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 8.8. Compare os resultados com a tabela 8.2.
7. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 8.2.

Tabelas de dados:

Seleção			Entradas								E'	Saídas	
A	B	C	I0	I1	I2	I3	I4	I5	I6	I7		F	F'
0	0	0									0		
0	0	1									0		
0	1	0									0		
0	1	1									0		
1	0	0									0		
1	0	1									0		
1	1	0									0		
1	1	1									0		

Tabela 8.2

Programa para o Arduíno:

```
// implementação de função booleana com multiplexador 8x1 (parte 1)

byte m0; byte m1; byte m2; byte m3; byte m4; byte m5; byte m6; byte m7;
byte i0; byte i1; byte i2; byte i3; byte i4; byte i5; byte i6; byte i7;
byte f; byte pi; byte aa; byte bb; byte cc;

void separa_bits(){
  i7 = pi & B00000001;
  i6 = pi & B00000010; i6 = i6>>1;
  i5 = pi & B00000100; i5 = i5>>2;
  i4 = pi & B00001000; i4 = i4>>3;
  i3 = pi & B00010000; i3 = i3>>4;
  i2 = pi & B00100000; i2 = i2>>5;
  i1 = pi & B01000000; i1 = i1>>6;
  i0 = pi & B10000000; i0 = i0>>7;
}

void mux8x1(byte ax, byte bx, byte cx){
  m0 = ((!ax&&!bx)&&!cx)&&i0);
  m1 = ((!ax&&!bx)&&cx)&&i1);
  m2 = ((!ax&&bx)&&!cx)&&i2);
  m3 = ((!ax&&bx)&&cx)&&i3);
  m4 = ((ax&&!bx)&&!cx)&&i4);
  m5 = ((ax&&!bx)&&cx)&&i5);
  m6 = ((ax&&bx)&&!cx)&&i6);
```

```
  m7 = (((ax&&bx)&&cx)&&i7);
  f = (((((((m0||m1)||m2)||m3)||m4)||m5)||m6)||m7);
}

void mostra_mux8x1(){
  if (aa) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (bb) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (cc) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (i0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i4) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i5) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i6) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i7) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (f) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Implemantação de função booleana com multiplexador
  8x1");
  Serial.println("| A | B | C || I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7
  || F |");

  Serial.println("-----------------------------------------------------
  -------------");
  for (aa = 0; aa <= 1; aa++){
    for (bb = 0; bb <= 1; bb++){
      for (cc = 0; cc <= 1; cc++){
          pi = B01010110; // função booleana F(A,B,C) = M(1,3,5,6)
          separa_bits();
          mux8x1(aa,bb,cc);
          mostra_mux8x1();
      }
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

8.1.6. Implementação de uma função booleana utilizando o CI 74LS151 (parte 2)

Esquemas:

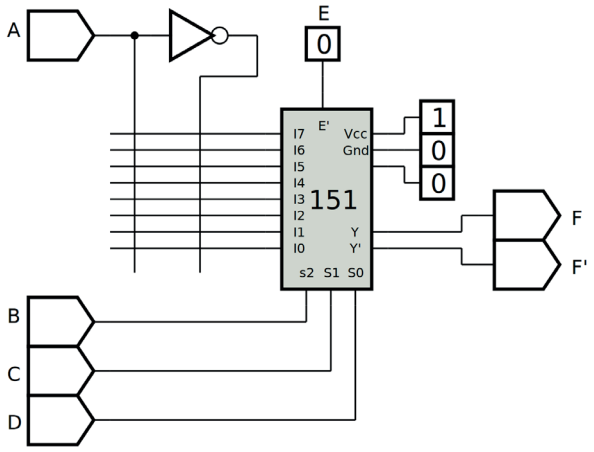


Figura 8.11 – Diagrama esquemático.

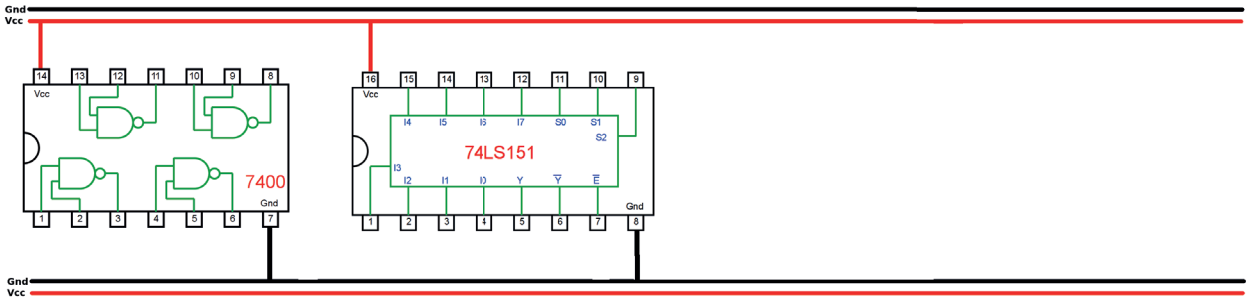


Figura 8.12 – Disposição dos componentes (TTL) para a placa de montagem.

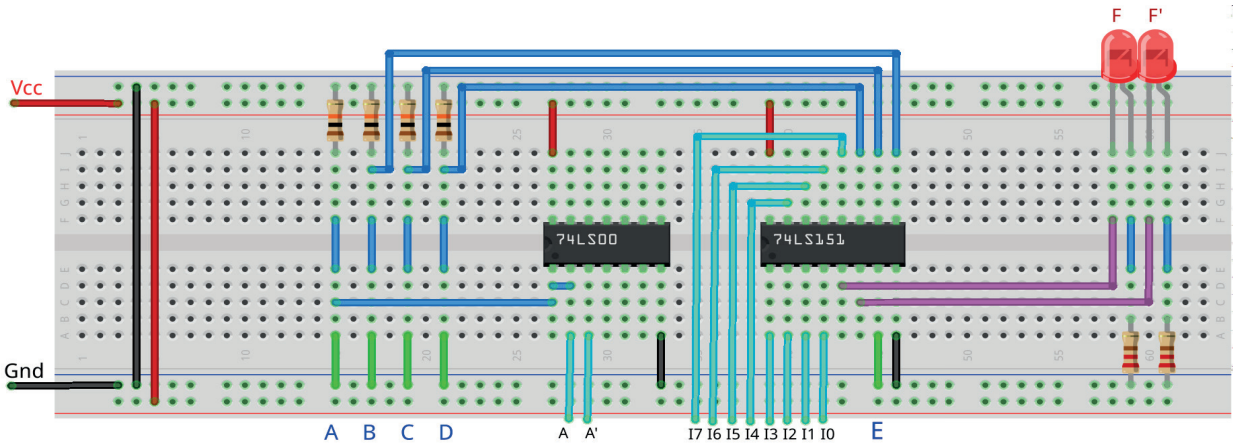


Figura 8.13 – Disposição dos componentes na placa de montagem.

Equação booleana:

F(A,B,C,D) = _____

Procedimento:

1. Complete a equação booleana para implementar a função: F(A,B,C,D) = Σ_m(0,1,5,7,9,10,11,15).
2. Faça as interligações necessárias para completar as figuras 8.11 e 8.12.
3 Monte o circuito na placa de montagem, conforme figura 8.13.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte os fios de projeto (azul claro) de acordo com o desenho da figura 8.12.
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
4. Para o preenchimento da tabela 8.3, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
5. Para o preenchimento das colunas de saída na tabela 8.3 considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
6. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 8.11. Compare os resultados com a tabela 8.3.
7. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 8.3.

Tabelas de dados:

Entradas				E'	Saídas	
A	B	C	D		F	F'
0	0	0	0	0		
0	0	0	1	0		
0	0	1	0	0		
0	0	1	1	0		
0	1	0	0	0		
0	1	0	1	0		
0	1	1	0	0		
0	1	1	1	0		
1	0	0	0	0		
1	0	0	1	0		
1	0	1	0	0		
1	0	1	1	0		
1	1	0	0	0		
1	1	0	1	0		
1	1	1	0	0		
1	1	1	1	0		

Tabela 8.3

Programa para o Arduíno:

```
// implementação de função booleana com multiplexador 8x1 (parte 2)

byte m0; byte m1; byte m2; byte m3; byte m4; byte m5; byte m6; byte m7;
byte i0; byte i1; byte i2; byte i3; byte i4; byte i5; byte i6; byte i7;
byte f; byte pi; byte aa; byte bb; byte cc; byte dd;

void separa_bits(){
  i7 = pi & B00000001;
  i6 = pi & B00000010; i6 = i6>>1;
  i5 = pi & B00000100; i5 = i5>>2;
  i4 = pi & B00001000; i4 = i4>>3;
  i3 = pi & B00010000; i3 = i3>>4;
  i2 = pi & B00100000; i2 = i2>>5;
  i1 = pi & B01000000; i1 = i1>>6;
```

```
  i0 = pi & B10000000; i0 = i0>>7;
}

void mux8x1(byte ax, byte bx, byte cx){
  m0 = ((!ax&&!bx)&&!cx)&&i0);
  m1 = ((!ax&&!bx)&&cx)&&i1);
  m2 = ((!ax&&bx)&&!cx)&&i2);
  m3 = ((!ax&&bx)&&cx)&&i3);
  m4 = ((ax&&!bx)&&!cx)&&i4);
  m5 = ((ax&&!bx)&&cx)&&i5);
  m6 = ((ax&&bx)&&!cx)&&i6);
  m7 = ((ax&&bx)&&cx)&&i7);
  f = (((((((m0||m1)||m2)||m3)||m4)||m5)||m6)||m7);
}

void mostra_mux8x1a(){
  if (aa) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (bb) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (cc) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (i0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i4) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i5) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i6) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i7) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (f) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Implementação de função booleana com multiplexador
  8x1");
  Serial.println("| A | B | C | D || I0 | I1 | I2 | I3 | I4 | I5 | I6 |
  I7 || F |");
  Serial.println("-----------------------------------------------------
  -----------");
```

```
  for (aa = 0; aa <= 1; aa++){
    for (bb = 0; bb <= 1; bb++){
      for (cc = 0; cc <= 1; cc++){
          for (dd = 0; dd <= 1; dd++){
            // função booleana F(A,B,C,D) = M(0,1,5,7,9,10,11,15)
            if (!aa) pi = B11000101; // A = 0 => 0,1,5,7
            else pi = B01110001; // A = 1 => 9,10,11,15
            separa_bits();
            mux8x1(bb,cc,dd);
            mostra_mux8x1a();
          }
        }
      }
    }
} // fim do ‘setup’

void loop(){
  // nada a fazer aqui!
} // fim do ‘loop’
```

8.2. Demultiplexadores

8.2.1. Objetivos

1. Investigar os circuitos do demultiplexador (DEMUX),
2. Observar seu funcionamento e obter suas tabelas verdade,
3. Conhecer o CI 74HC237 demultiplexador 1x8.

8.2.2. Informação preliminar

Um demultiplexador, abreviado como DEMUX, é usado quando um circuito envia um sinal para um dos muitos dispositivos. Esta descrição soa semelhante à descrição dada para um decodificador, mas um decodificador é usado para selecionar entre muitos dispositivos, enquanto um demultiplexador é usado para enviar um sinal entre muitos dispositivos. No entanto, qualquer decodificador que tenha uma linha de habilitação pode funcionar como demultiplexador. Se a linha de habilitação de um decodificador for usada como uma entrada de dados, então os dados podem ser roteados para qualquer uma das saídas e, portanto, nesse caso, o decodificador pode ser usado como um demultiplexador. Como o nome infere, um demultiplexador executa a função oposta à de um multiplexador. Um único sinal de entrada pode ser conectado a qualquer uma das linhas de saída fornecidas pela escolha de um sinal de seleção apropriado. A forma geral de um demultiplexador de *1-para-n* pode ser vista na figura 8.14. Se houver entradas de seleção "*m*", o número de linhas de saída para as quais os dados podem ser roteados é $n = 2^m$. Como pode ser visto na figura 8.14, além das outras entradas, o demultiplexador pode ter uma linha de habilitação, *E*, para habilitá-lo. O demultiplexador é habilitado com *E* definido como "1" (para entrada de habilitação E ativa em alto). Quando o demultiplexador é desativado com *E* definido como "0" (para entrada de habilitação E ativa em alto), nenhuma linha de saída é selecionada e, portanto, o sinal de entrada não é passado para nenhuma linha de saída.

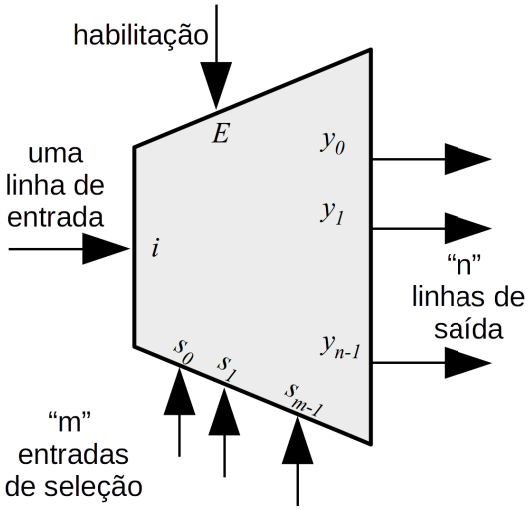
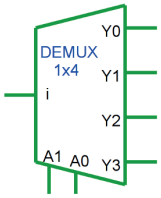


Figura 8.14 – A forma geral de um demultiplexador de *1-para-n*, onde $n = 2^m$.

O símbolo esquemático, o diagrama lógico e a tabela verdade de um demultiplexador 1x4 são fornecidos na figura 8.15. Quando a entrada *i* é considerada como uma entrada de habilitação ativa em alto, este circuito pode ser usado como um decodificador 2x4. Como existem 4 linhas de saída para um DEMUX 1x4, existem 2 entradas de seleção (quando $2^n=4$, $n=2$). Da mesma forma que existem 8 linhas de saída para um DEMUX 1x8, existem 3 entradas de seleção (quando $2^n=8$, $n=3$).



Símbolo esquemático

Entradas		Saídas			
A1	A0	Y0	Y1	Y2	Y3
0	0	i	0	0	0
0	1	0	i	0	0
1	0	0	0	i	0
1	1	0	0	0	i

Tabela verdade

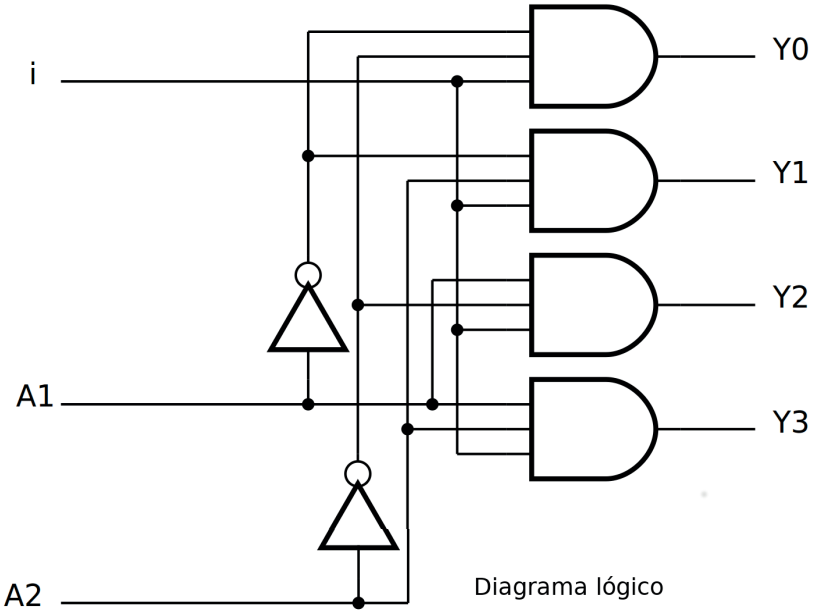
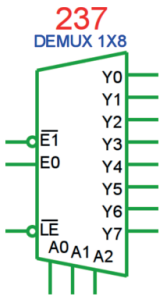


Diagrama lógico

Figura 8.15 – O símbolo esquemático, o diagrama lógico e a tabela verdade de um demultiplexador 1x4.

8.2.3. Circuito Integrado TTL 74HC237 – Decodificador 3x8 / demultiplexador 1x8

A figura 8.16 mostra o símbolo esquemático, o diagrama lógico e a tabela verdade do CI 74HC237. O 74HC237 pode ser usado como um decodificador de 3-para-8 linhas ou um demultiplexador de 1-para-8 com memórias nas três entradas de endereço (A2, A1, A0). O 74HC237 essencialmente combina a função de decodificador de 3-para-8 com uma memória de armazenamento de 3 bits. Quando a memória está ativada (LE'=0), o 74HC237 atua como um decodificador de 3-para-8 ativo em baixo. Quando a habilitação de memória (LE') passa de baixo (0) para alto (1), os últimos dados presentes nas entradas antes dessa transição são armazenados nas memórias. Outras alterações de endereço são ignoradas, desde que o LE' permaneça alto (1). As entradas de habilitação de saída (E1' e E0) controlam o estado das saídas independentemente das entradas de endereço ou da operação de memória. Todas as saídas são altas (1), a menos que E1' seja baixo (0) e E0 seja alto (1). O 74HC237 é ideal para a implementação de decodificadores não sobrepostos em sistemas de 3 estados e aplicações estroboscópicas (endereço armazenado) em sistemas orientados a barramento. Duas entradas de habilitação de saída (E1' e E0) são fornecidas para simplificar a conexão em cascata e facilitar a demultiplexagem. A função de demultiplexação é realizada usando as entradas de endereço A2, A1, A0 para selecionar a saída desejada e usando uma das outras entradas de habilitação de saída como entrada de dados enquanto mantém a outra entrada de habilitação de saída em seu estado ativo. Isto é para dizer que quando se utiliza o 74HC237 como um demultiplexador de 1 para 8, as entradas de endereço A2, A1, A0 são usadas para selecionar a saída desejada Y7, Y6, Y5, Y4, Y3, Y2, Y1 ou Y0. Nesse caso, se quisermos usar o sinal de entrada ativo em alto, E1' é definido como 0 e E0 é usado como a entrada ativa em alto. Da mesma forma, se gostaríamos de usar o sinal de entrada ativo em baixo, então E0 é definido como 1 e E1' é usado como a entrada ativa em baixo.



Símbolo esquemático

Entradas						Saídas							
LE	E0	E1	A2	A1	A0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	X	1	X	X	X	0	0	0	0	0	0	0	0
X	0	X	X	X	X	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0	1	0	0	0	0	0
0	1	0	0	1	1	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	1	0	0	0
0	1	0	1	0	1	0	0	0	0	0	1	0	0
0	1	0	1	1	0	0	0	0	0	0	0	1	0
0	1	0	1	1	1	0	0	0	0	0	0	0	1
1	1	0	X	X	X	Mantém a última condição de A2 A1 A0.							

Tabela verdade (x = não importa)

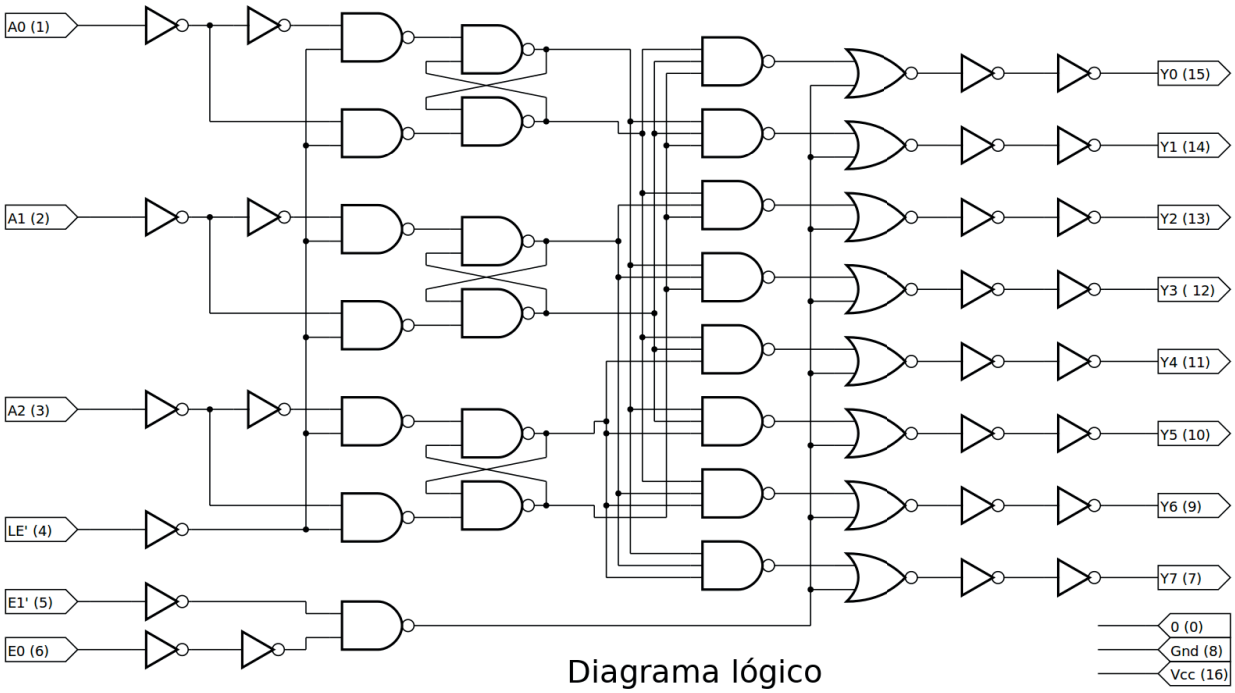


Diagrama lógico

Figura 8.16 – O diagrama lógico, o símbolo esquemático, e a tabela verdade do CI 74HC237 demultiplexador 1x8.

8.2.4. Exame do CI 74LS139 atuando como demultiplexador 1x4

Utilização do CI TTL 74LS139, que possui dois decodificadores 2x4, atuando como demultiplexador 1x4 com entrada ativa em baixo “0” e saídas ativas em baixo “0”.

Esquemas:

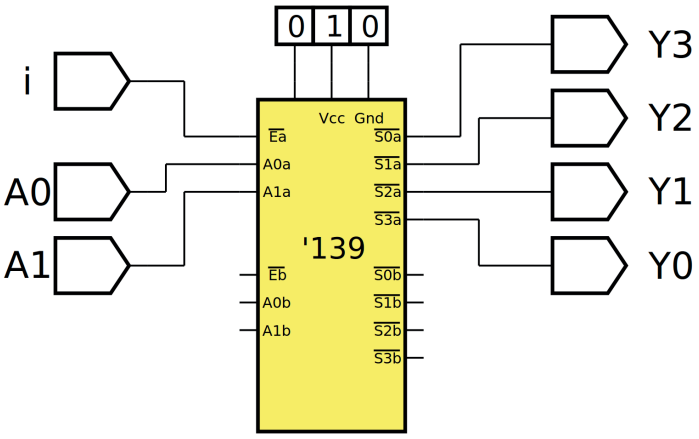


Figura 8.17 – Diagrama esquemático.

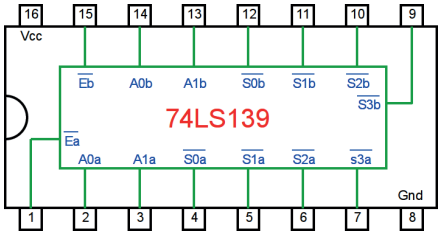


Figura 8.18 – Circuito integrado TTL.

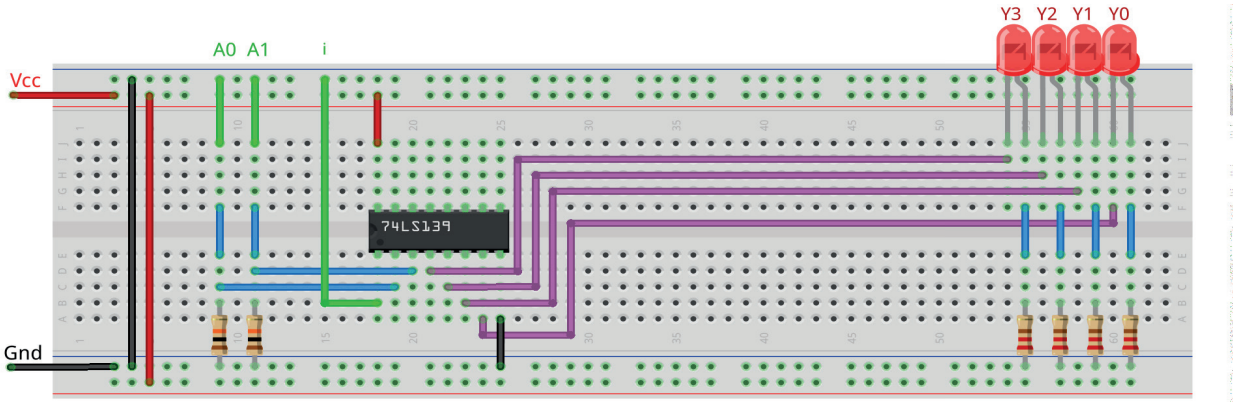


Figura 8.19 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 8.19.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 8.4, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 8.4, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 8.17. Compare os resultados com a tabela 8.4.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 8.4.

Tabelas de dados:

Entradas			Saídas			
A1	A0	i	Y3	Y2	Y1	Y0
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Tabela 8.4

Programa para o Arduíno:

```
// Implementação de um demultiplexador 1x4 usando um codificador 2x4
com 'enable'

byte s0; byte s1; byte s2; byte s3; byte a1; byte a0; byte i;

void decod2x4(byte b, byte a, byte en){
  s0 = !(( !en && !b ) && !a ); // 00
  s1 = !(( !en && !b ) && a ); // 01
  s2 = !(( !en && b ) && !a ); // 10
  s3 = !(( !en && b ) && a ); // 11
}

void mostra_demux1x4(){
  if (a1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (a0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (i) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (s3) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s2) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s1) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (s0) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Implementação de um demultiplexador 1x4 usando um
  codificador 2x4");
  Serial.println("Entrada ativa em baixo (0) e saídas ativas em baixo
  (0)");
  Serial.println("| A1 | A0 | i || Y3 | Y2 | Y1 | Y0 |");

  Serial.println("-------------------------------------");
  for (a1 = 0; a1 <= 1; a1++){
    for (a0 = 0; a0 <= 1; a0++){
      for (i = 0; i <= 1; i++){
        decod2x4(a1,a0,i);
        mostra_demux1x4();
      }
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

8.2.5. Exame do CI 74LS139 atuando como demultiplexador 1x8

Utilização do CI TTL 74LS139, que possui dois decodificadores 2x4, atuando como demultiplexador 1x8 com entrada ativa em baixo “0” e saídas ativas em baixo “0”.

Esquemas:

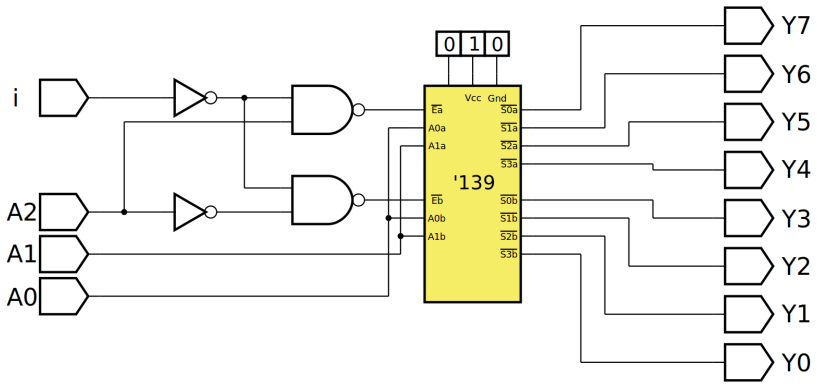


Figura 8.20 – Diagrama esquemático.

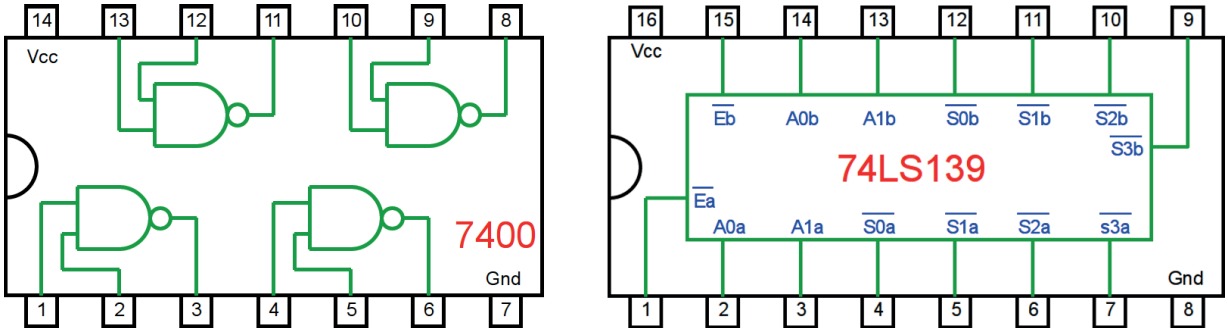


Figura 8.21 – Circuito integrado TTL.

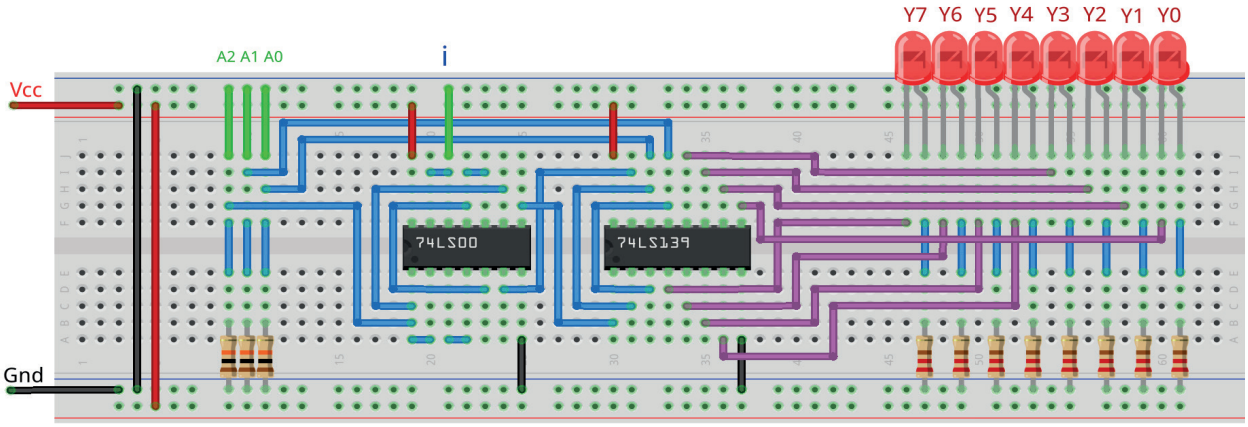


Figura 8.22 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 8.22.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 8.5, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 8.5, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 8.20. Compare os resultados com a tabela 8.5.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 8.5.

Tabelas de dados:

Entradas				Saídas							
A2	A1	A0	i	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0								
0	0	0	1								
0	0	1	0								
0	0	1	1								
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	1	1								
1	0	0	0								
1	0	0	1								
1	0	1	0								
1	0	1	1								
1	1	0	0								
1	1	0	1								
1	1	1	0								
1	1	1	1								

Tabela 8.5

Programa para o Arduíno:

```
// Implementação de um demultiplexador 1x8 usando dois codificadores
2x4 com ‘enable’

byte s0; byte s1; byte s2; byte s3;
byte y0; byte y1; byte y2; byte y3; byte y4; byte y5; byte y6; byte y7;
byte a2; byte a1; byte a0; byte i; byte en1; byte en2;

void decod2x4(byte b, byte a, byte en){
  s0 = !(( !en && !b ) && !a ); // 00
  s1 = !(( !en && !b ) && a ); // 01
  s2 = !(( !en && b ) && !a ); // 10
  s3 = !(( !en && b ) && a ); // 11
}

void mostra_demux1x8(){
  if (a2) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (a1) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (a0) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (i) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  Serial.print(“|”);
  if (y7) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y6) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y5) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y4) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y3) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y2) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y1) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y0) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  Serial.println(“|”);
}

void setup(){
  Serial.begin(9600);
  Serial.println(“Implementação de um demultiplexador 1x8 usando dois
  codificadores 2x4”);
  Serial.println(“Entrada ativa em baixo (0) e saídas ativas em baixo
  (0)”);

  Serial.println(“| A2 | A1 | A0 | i || Y7 | Y6 | Y5 | Y4 | Y3 | Y2 |
  Y1 | Y0 |”);
```

```
  Serial.println("-----------------------------------------------------
  -----------");
  for (a2 = 0; a2 <= 1; a2++){
    for (a1 = 0; a1 <= 1; a1++){
      for (a0 = 0; a0 <= 1; a0++){
        for (i = 0; i <= 1; i++){
          en1 = !(!i && !a2); // lógica do enable 1 (Y0,Y1,Y2,Y3)
          decod2x4(a1,a0,en1);
          y0 = s0;
          y1 = s1;
          y2 = s2;
          y3 = s3;
          en2 = !(!i && a2); // lógica do enable 2 (Y4,Y5,Y6,Y7)
          decod2x4(a1,a0,en2);
          y4 = s0;
          y5 = s1;
          y6 = s2;
          y7 = s3;
          mostra_demux1x8();
        }
      }
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

8.2.6. Exame do CI demultiplexador 1x8 74HC237

Esquemas:

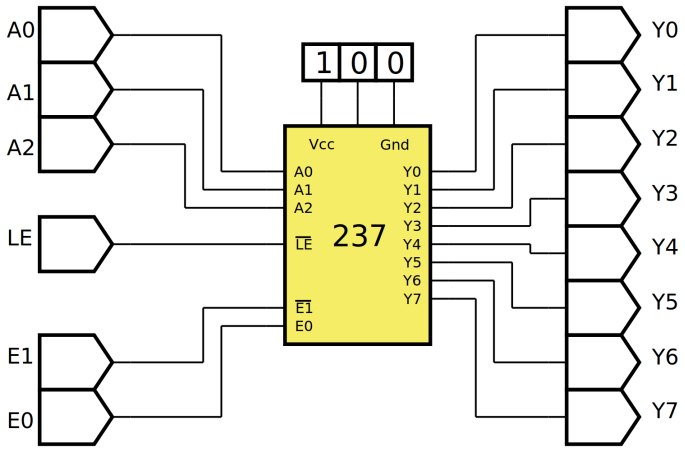


Figura 8.23 – Diagrama esquemático.

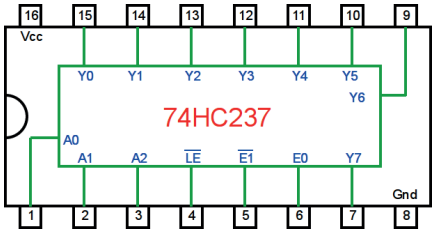


Figura 8.24 – Circuito integrado TTL.

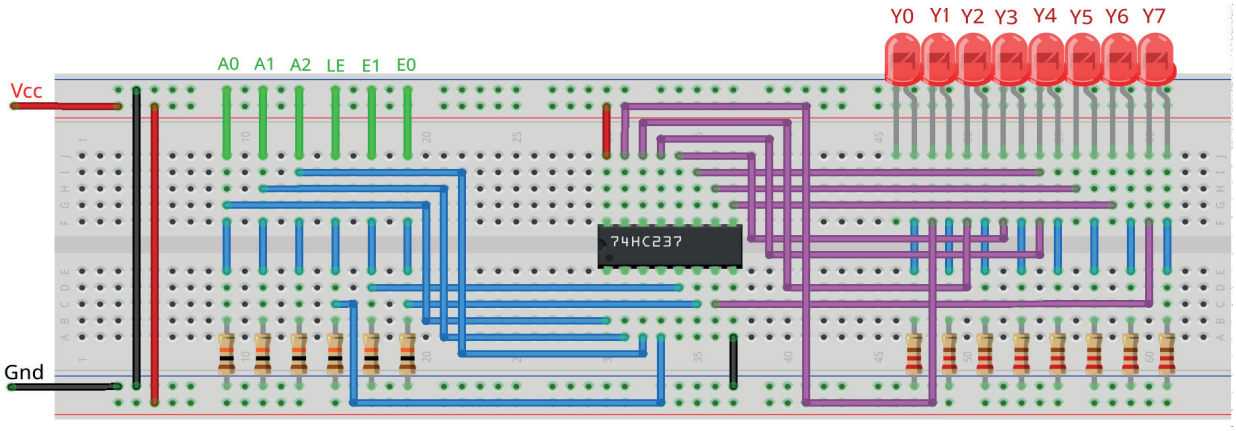


Figura 8.25 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 8.25.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 8.6, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 8.6, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 8.23. Compare os resultados com a tabela 8.6.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 8.6.

Tabelas de dados:

Entradas						Saídas							
A2	A1	A0	LE	E1	E0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	0	0	1								
0	0	0	0	1	1								
0	0	1	0	0	1								
0	0	1	0	1	1								
0	1	0	0	0	1								
0	1	0	0	1	1								
0	1	1	0	0	1								
0	1	1	0	1	1								
1	0	0	0	0	1								
1	0	0	0	1	1								
1	0	1	0	0	1								
1	0	1	0	1	1								
1	1	0	0	0	1								
1	1	0	0	1	1								
1	1	1	0	0	1								
1	1	1	0	1	1								
0	0	0	0	0	0								
0	0	0	0	0	1								
0	0	1	0	0	0								
0	0	1	0	0	1								
0	1	0	0	0	0								
0	1	0	0	0	1								
0	1	1	0	0	0								
0	1	1	0	0	1								
1	0	0	0	0	0								
1	0	0	0	0	1								
1	0	1	0	0	0								
1	0	1	0	0	1								
1	1	0	0	0	0								
1	1	0	0	0	1								
1	1	1	0	0	0								
1	1	1	0	0	1								

Tabela 8.6

Programa para o Arduíno:

```
// demultiplexador 1x8 usando dois ‘enable’

byte y0; byte y1; byte y2; byte y3; byte y4; byte y5; byte y6; byte y7;
byte a2; byte a1; byte a0; byte e1; byte e0;

void demux1x8(byte c, byte b, byte a, byte en1, byte en0){
  y0 = ((((!c && !b) && !a) && !en1) && en0); // 000
  y1 = ((((!c && !b) && a) && !en1) && en0); // 001
  y2 = ((((!c && b) && !a) && !en1) && en0); // 010
  y3 = ((((!c && b) && a) && !en1) && en0); // 011
  y4 = ((((c && !b) && !a) && !en1) && en0); // 100
  y5 = ((((c && !b) && a) && !en1) && en0); // 101
  y6 = ((((c && b) && !a) && !en1) && en0); // 110
  y7 = ((((c && b) && a) && !en1) && en0); // 111
}

void mostra_demux1x8(){
  if (a2) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (a1) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (a0) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (e1) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (e0) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  Serial.print(“|”);
  if (y0) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y1) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y2) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y3) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y4) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y5) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y6) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  if (y7) Serial.print(“| 1 “); else Serial.print(“| 0 “);
  Serial.println(“|”);
}

void setup(){
  Serial.begin(9600);
  Serial.println(“Demultiplexador 1x8”);
  Serial.println(“Entrada E1 ativa em baixo (0), entrada E0 ativa em
  alta (1)”);
  Serial.println(“| A2 | A1 | A0 | E1 | E0 || Y0 | Y1 | Y2 | Y3 | Y4 |
  Y5 | Y6 | Y7 |”);
```

```
  Serial.println("-----------------------------------------------------
  ----------------");

  for (a2 = 0; a2 <= 1; a2++){
    for (a1 = 0; a1 <= 1; a1++){
      for (a0 = 0; a0 <= 1; a0++){
        e0 = 1;
        for (e1 = 0; e1 <= 1; e1++){
          demux1x8(a2,a1,a0,e1,e0);
          mostra_demux1x8();
      }
    }
  }
}
Serial.println("");
for (a2 = 0; a2 <= 1; a2++){
  for (a1 = 0; a1 <= 1; a1++){
    for (a0 = 0; a0 <= 1; a0++){
      e1 = 0;
      for (e0 = 0; e0 <= 1; e0++){
        demux1x8(a2,a1,a0,e1,e0);
        mostra_demux1x8();
      }
    }
  }
}
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

8.3. Transferência de dados digitais usando uma combinação de mux-demux

8.3.1. Objetivos

1. Analisar a conexão do circuito MUX-DEMUX utilizado para transferência de dados digitais,
2. Observar seu funcionamento.

8.3.2. Informação preliminar

A transferência de dados é um tópico muito importante no mundo da eletrônica. Diferentes métodos são usados para esta operação. Principalmente, os dados são transferidos de duas maneiras:

a) Sistemas com fio

b) Sistemas sem fio.

Existem muitos tipos de sistemas com fio. Os mais importantes são sistemas digitais e analógicos. A mesma variação também é válida para sistemas sem fio. Hoje, os sistemas digitais são mais populares. Um método usado para transferência de dados através de uma única linha (com ou sem fio) é o método de multiplexação. Para explicar esse método, vamos considerar o sistema que foi usado para comunicação telefônica analógica no passado. Na figura 8.26, como os dados são transferidos através de uma única linha por dois comutadores analógicos que operam de forma síncrona são representados basicamente.

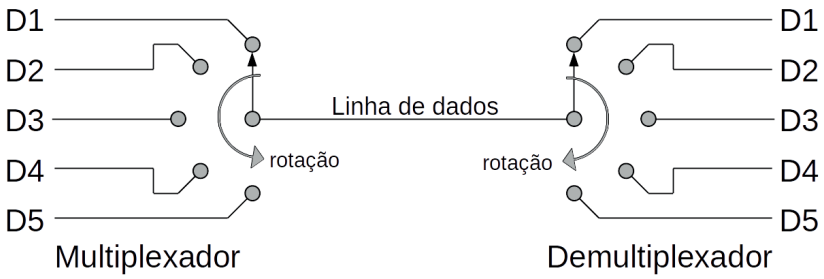


Figura 8.26 – Transferência de dados através de uma única linha por dois comutadores analógicos que operam de forma síncrona.

Como cinco transferências diferentes podem ser obtidas através de uma única linha são mostradas basicamente na figura 8.26. Como os comutadores operam de maneira síncrona, as mesmas posições correspondem a cada vez, como D1-D1 ou D2-D2. No entanto, este sistema tem as seguintes desvantagens: é mecânico, propenso a falhas, lento e a capacidade de transferência de dados é baixa. Para superar esses problemas, os sistemas digitais são usados. A estrutura principal disto é mostrada na figura 8.27.

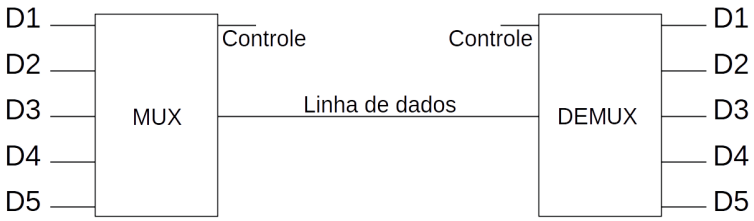


Figura 8.27 – A conexão do circuito MUX-DEMUX usado para transferência de dados digitais.

8.3.3. Exame da transferência de dados digitais usando uma combinação de mux-demux.

Esquemas:

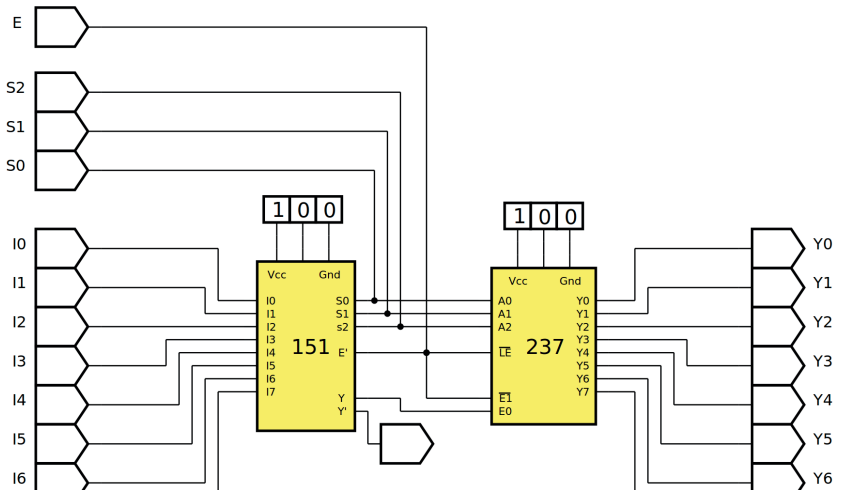


Figura 8.28 – Diagrama esquemático.

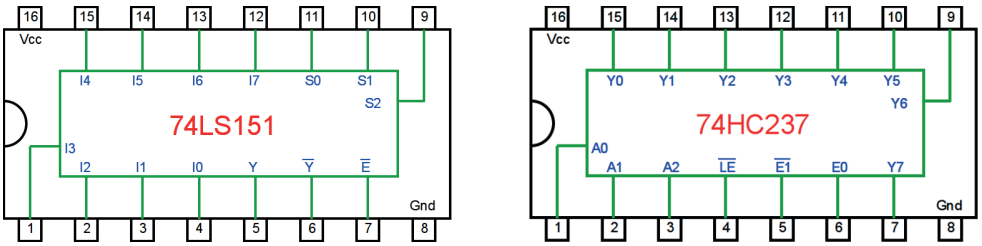


Figura 8.29 – Circuitos integrados TTL.

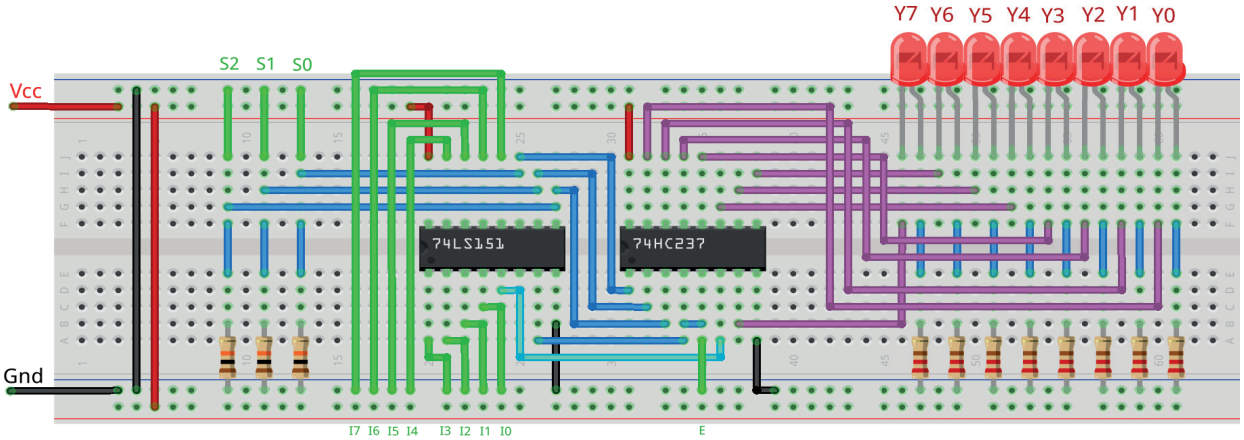


Figura 8.30 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 8.30.
   a) Coloque os componentes (resistores, LEDs e CIs) de acordo com a disposição mostrada.
   b) Conecte os fios de entrada (verdes) e os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de conexão (azuis) entre os resistores e o CI e os LEDs.
   d) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   e) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   f) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 8.7, use os fios verdes como segue:
   a) Se o valor da variável for “0” o fio verde correspondente deve ser conectado à barra Gnd.
   b) Se o valor da variável for “1” o fio verde correspondente deve ser conectado à barra Vcc.
3. Para o preenchimento das colunas de saída na tabela 8.7, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 8.28. Compare os resultados com a tabela 8.7.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 8.7.

Tabelas de dados:

Seleção				Entradas (74LS151)								Saídas (74HC237)							
S2	S1	S0	E	I7	I6	I5	I4	I3	I2	I1	I0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
X	X	X	1	X	X	X	X	X	X	X	X								
0	0	0	0	0	0	0	0	0	0	0	0								
0	0	0	0	0	0	0	0	0	0	0	1								
0	0	1	0	0	0	0	0	0	0	0	0								
0	0	1	0	0	0	0	0	0	0	1	0								
0	1	0	0	0	0	0	0	0	0	0	0								
0	1	0	0	0	0	0	0	0	1	0	0								
0	1	1	0	0	0	0	0	0	0	0	0								
0	1	1	0	0	0	0	0	1	0	0	0								
1	0	0	0	0	0	0	0	0	0	0	0								
1	0	0	0	0	0	0	1	0	0	0	0								
1	0	1	0	0	0	0	0	0	0	0	0								
1	0	1	0	0	0	1	0	0	0	0	0								
1	1	0	0	0	0	0	0	0	0	0	0								
1	1	0	0	0	1	0	0	0	0	0	0								
1	1	1	0	0	0	0	0	0	0	0	0								
1	1	1	0	1	0	0	0	0	0	0	0								

Tabela 8.7 (X = não importa)

Programa para o Arduíno:

Para a realização desta experiência, é necessário fazer a montagem de dois Arduínos com alguns componentes eletrônicos, de acordo com as figuras 8.31 e 8.32. Os dois conjuntos devem ficar próximos e interligados por 5 (cinco) fios: 3 (três) fios azuis para transferência do código binário, 1 (um) fio roxo para transferência dos dados e 1 (um) fio preto para interligação dos sinais de terra (Gnd) das duas placas de montagem.

Identificação	Fio	Pino no transmissor	Pino no receptor
Binário S0	Azul	10	2
Binário S1	Azul	11	3
Binário S2	Azul	12	4
Binário E	Roxo	13	5
Referência	Preto	Gnd	Gnd

Tabela 8.8 – Interligação dos fios entre o Arduíno Transmissor e o Arduíno Receptor.

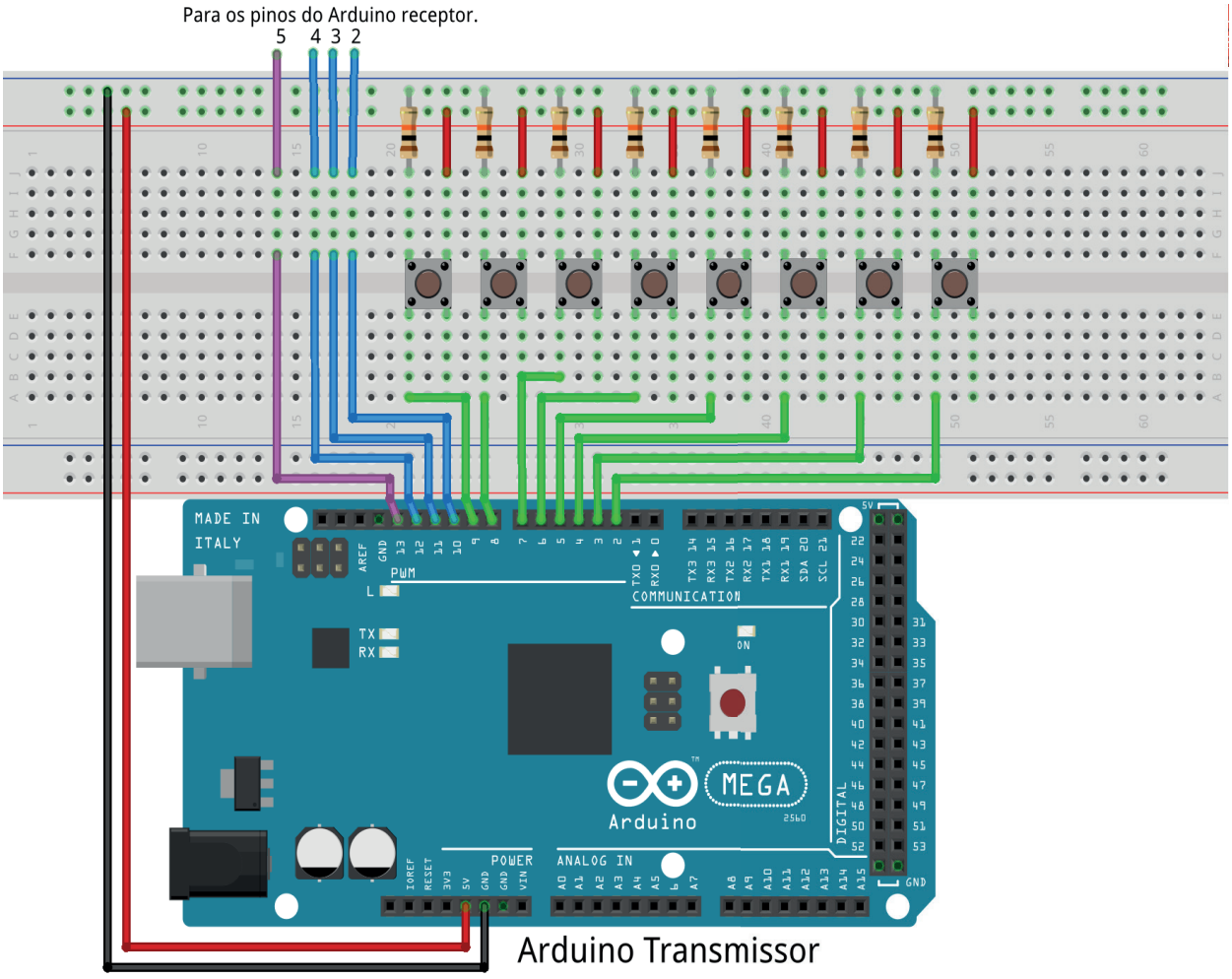


Figura 8.31 – Disposição dos componentes na placa de montagem (transmissor).

Programa para o Arduíno transmissor:

```
// multiplexador 8x1 (transmissor)

// pinos dos botões de entrada
  #define pino_i0 2
  #define pino_i1 3
  #define pino_i2 4
  #define pino_i3 5
  #define pino_i4 6
  #define pino_i5 7
  #define pino_i6 8
  #define pino_i7 9

// pinos dos fios de saída
  #define pino_a0 10
  #define pino_a1 11
  #define pino_a2 12
  #define pino_y 13

// variáveis do multiplex
  byte m0; byte m1; byte m2; byte m3; byte m4; byte m5; byte m6; byte m7;
  byte y;

// variáveis de entrada
  byte i0; byte i1; byte i2; byte i3; byte i4; byte i5; byte i6; byte i7;

// variáveis do seletor rotativo
  byte a2; byte a1; byte a0;
  void mux8x1(byte s2, byte s1, byte s0){
  m0 = ((!s2&&!s1)&&!s0)&&i0);
  m1 = ((!s2&&!s1)&&s0)&&i1);
  m2 = ((!s2&&s1)&&!s0)&&i2);
  m3 = ((!s2&&s1)&&s0)&&i3);
  m4 = ((s2&&!s1)&&!s0)&&i4);
  m5 = ((s2&&!s1)&&s0)&&i5);
  m6 = ((s2&&s1)&&!s0)&&i6);
  m7 = ((s2&&s1)&&s0)&&i7);
  y = (((((((m0||m1)||m2)||m3)||m4)||m5)||m6)||m7);
}

void setup(){
  pinMode(pino_i0, INPUT);
  pinMode(pino_i1, INPUT);
```

```
  pinMode(pino_i2, INPUT);
  pinMode(pino_i3, INPUT);
  pinMode(pino_i4, INPUT);
  pinMode(pino_i5, INPUT);
  pinMode(pino_i6, INPUT);
  pinMode(pino_i7, INPUT);
  pinMode(pino_a0, OUTPUT);
  pinMode(pino_a1, OUTPUT);
  pinMode(pino_a2, OUTPUT);
  pinMode(pino_y, OUTPUT);
} // fim do 'setup'

void loop(){

// leitura das entradas
  i0 = digitalRead(pino_i0);
  i1 = digitalRead(pino_i1);
  i2 = digitalRead(pino_i2);
  i3 = digitalRead(pino_i3);
  i4 = digitalRead(pino_i4);
  i5 = digitalRead(pino_i5);
  i6 = digitalRead(pino_i6);
  i7 = digitalRead(pino_i7);

// multiplex 8x1
  for (a2 = 0; a2 <= 1; a2++){
    for (a1 = 0; a1 <= 1; a1++){
      for (a0 = 0; a0 <= 1; a0++){
        // seleciona entrada
        mux8x1(a2,a1,a0);
        // transmite dados para o outro Arduino
        digitalWrite(pino_a0,a0);
        digitalWrite(pino_a1,a1);
        digitalWrite(pino_a2,a2);
        digitalWrite(pino_y,y);
        // aguarda um tempo
        delay(500);
      }
    }
  }
} // fim do 'loop'
```

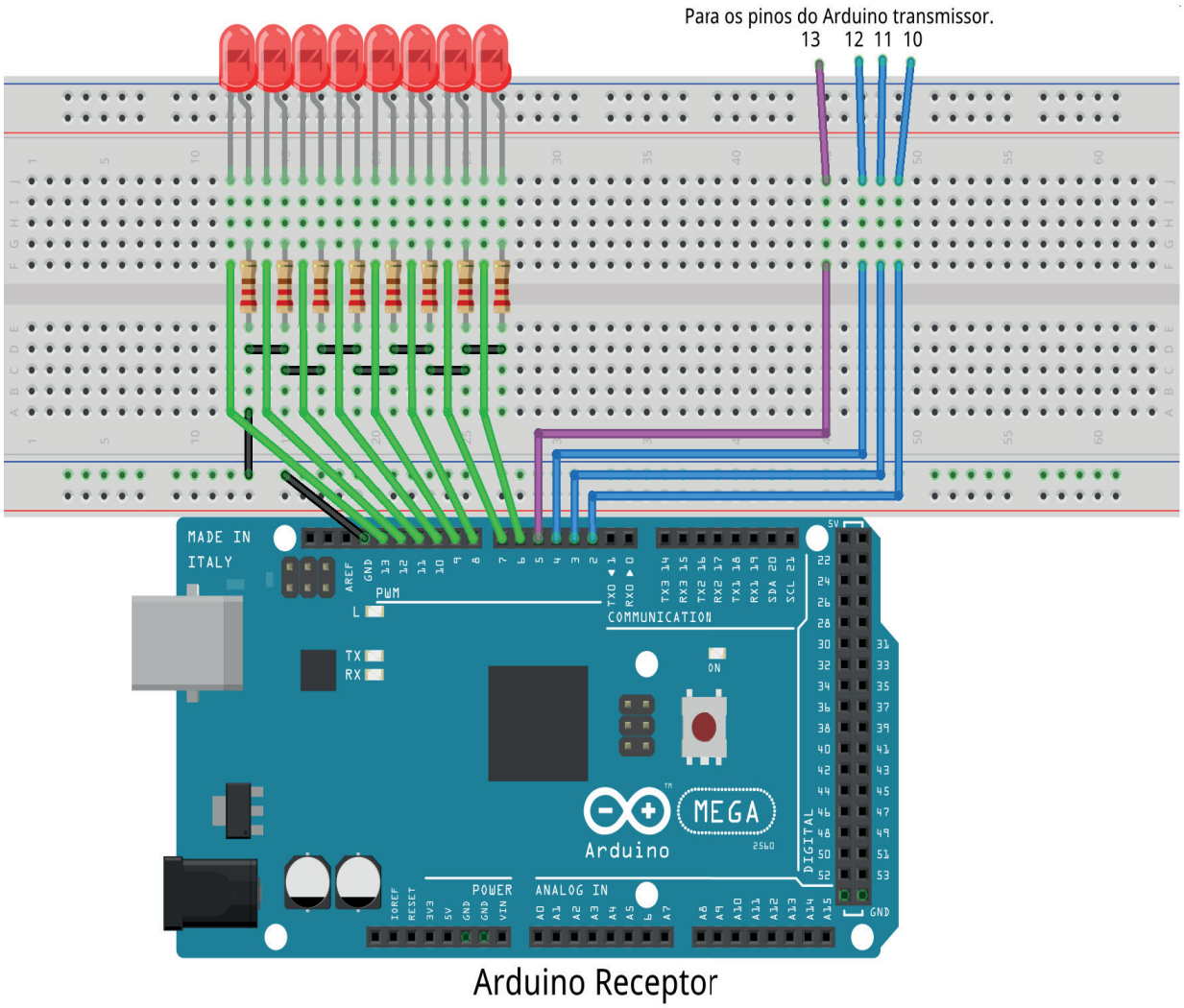


Figura 8.32 – Disposição dos componentes na placa de montagem (receptor).

Programa para o Arduíno receptor:

```
// demultiplexador 1x8 (receptor)

// pinos dos fios de entrada
  #define pino_d0 2
  #define pino_d1 3
  #define pino_d2 4
  #define pino_i 5

// pinos dos LEDs de saída
  #define pino_y0 6
  #define pino_y1 7
  #define pino_y2 8
  #define pino_y3 9
  #define pino_y4 10
  #define pino_y5 11
  #define pino_y6 12
  #define pino_y7 13

// variáveis do demultiplex
  byte y0; byte y1; byte y2; byte y3; byte y4; byte y5; byte y6; byte y7;

// variáveis de entrada
  byte d2; byte d1; byte d0; byte i;

void demux1x8(byte c, byte b, byte a, byte en1, byte en0){
  y0 = ((((!c && !b) && !a) && !en1) && en0); // 000
  y1 = ((((!c && !b) && a) && !en1) && en0); // 001
  y2 = ((((!c && b) && !a) && !en1) && en0); // 010
  y3 = ((((!c && b) && a) && !en1) && en0); // 011
  y4 = ((((c && !b) && !a) && !en1) && en0); // 100
  y5 = ((((c && !b) && a) && !en1) && en0); // 101
  y6 = ((((c && b) && !a) && !en1) && en0); // 110
  y7 = ((((c && b) && a) && !en1) && en0); // 111
}

void setup(){
  pinMode(pino_y0, OUTPUT);
  pinMode(pino_y1, OUTPUT);
  pinMode(pino_y2, OUTPUT);
  pinMode(pino_y3, OUTPUT);
```

```
  pinMode(pino_y4, OUTPUT);
  pinMode(pino_y5, OUTPUT);
  pinMode(pino_y6, OUTPUT);
  pinMode(pino_y7, OUTPUT);
  pinMode(pino_d0, INPUT);
  pinMode(pino_d1, INPUT);
  pinMode(pino_d2, INPUT);
  pinMode(pino_i, INPUT);
} // fim do ‘setup’

void loop(){
// leitura das entradas
  d0 = digitalRead(pino_d0);
  d1 = digitalRead(pino_d1);
  d2 = digitalRead(pino_d2);
  i = digitalRead(pino_i);

// demultiplex 1x8
  // seleciona saída
  demux1x8(d2,d1,d0,0,i);
  // mostra valor nos LEDs
  digitalWrite(pino_y0,y0);
  digitalWrite(pino_y1,y1);
  digitalWrite(pino_y2,y2);
  digitalWrite(pino_y3,y3);
  digitalWrite(pino_y4,y4);
  digitalWrite(pino_y5,y5);
  digitalWrite(pino_y6,y6);
  digitalWrite(pino_y7,y7);

} // fim do ‘loop’
```

Capítulo 9. Circuitos lógicos sequenciais: flip-flops

9.1. Flip-flop SR (trava, “latch”)

9.1.1. Objetivos

1. Conhecer o flip-flop RS e aprender sua operação,
2. Obter sua tabela verdade,
3. Construir o flip-flop SR usando CIs com portas lógicas NOU e NE,
4. Conhecer o flip-flop SR gatilhável e aprender sua operação.

9.1.2. Informação preliminar

Nos experimentos anteriores, circuitos lógicos combinacionais como decodificadores, codificadores, multiplexadores e demultiplexadores foram considerados. Por outro lado, em um circuito lógico sequencial; além das portas lógicas, também há elementos de memória. Os flip-flops são elementos básicos de memória usados em um circuito lógico sequencial. Quando os elementos de memória são removidos de um circuito lógico sequencial, a parte restante do circuito é apenas a parte combinacional. Neste experimento, os flip-flops são considerados. Um dos dispositivos básicos que armazenam e processam os dados digitais (1s e 0s) é o flip-flop. Existem basicamente quatro tipos de flip-flops:

1. Flip-flop SR (trava)
2. Flip-flop D
3. Flip-flop JK
4. Flip-flop T

Primeiro de tudo, vamos considerar os flip-flops SR (“Set-Reset”). Existem dois tipos de flip-flops SR: Flip-flop SR (FF SR) com entrada ativa em alto “1”, e Flip-flop SR (FF SR) com entrada ativa em baixo “0”. Observe que um flip-flop SR também é chamado de trava SR. Um FF com entrada ativa em alto é formada com duas portas NOU acopladas cruzadas, como mostrado na figura 9.1. As entradas são S (liga, “set”) e R (desliga, “reset”), enquanto a saída é Q e seu complemento é Q’. A tabela verdade de um FF SR com entrada ativa em alto é fornecida na tabela 9.1. A sua operação é explicada a seguir:

Se S = 0 e R = 0, então o FF SR preservará sua saída anterior.

Se S = 0 e R = 1, então o FF SR é desativado, ou Q = 0.

Se S = 1 e R = 0, então o SR FF é ativado, ou Q = 1.

Se S = 1 e R = 1, então Q e Q’ são 0. Isso é chamado de operação inválida para a entrada ativa em alto do FF SR.

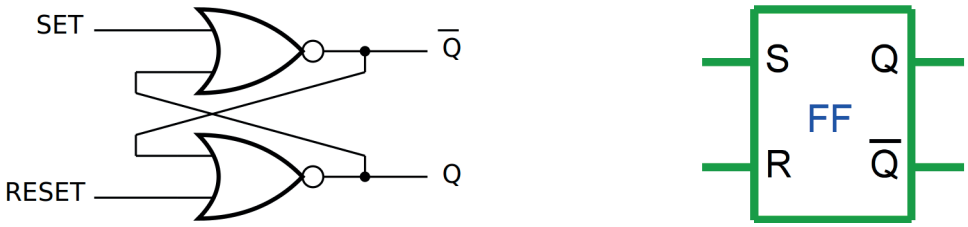


Figura 9.1 – O diagrama esquemático e símbolo do flip-flop SR com entrada ativa em alto.

Entradas		Saídas		Comentário
S	R	Q	Q'	
0	0	Q_0	Q_0'	Sem mudança.
0	1	0	1	Desativação.
1	0	1	0	Ativação.
1	1	0	0	Inválido.

Tabela 9.1 – A tabela verdade do flip-flop SR com entrada ativa em alto.

Um FF SR com entrada ativa em baixo é formada com duas portas NE acopladas cruzadas, como mostrado na figura 9.2. As entradas são S'(liga, "set") e R'(desliga, "reset"), enquanto a saída é Q e seu complemento é Q'. A tabela verdade do FF SR com entrada ativa em baixo é fornecida na tabela 9.2. O seu funcionamento é explicado a seguir:

Se S = 0 e R = 0, então ambos Q e Q' são 1. Isso é chamado de operação inválida para o FF SR de entrada ativa em baixo.

Se S = 0 e R = 1, então o FF SR é ativado, ou Q = 1.

Se S = 1 e R = 0, então o FF SR é desativado, ou Q = 0.

Se S = 1 e R = 1, o FF SR preserva sua saída anterior.

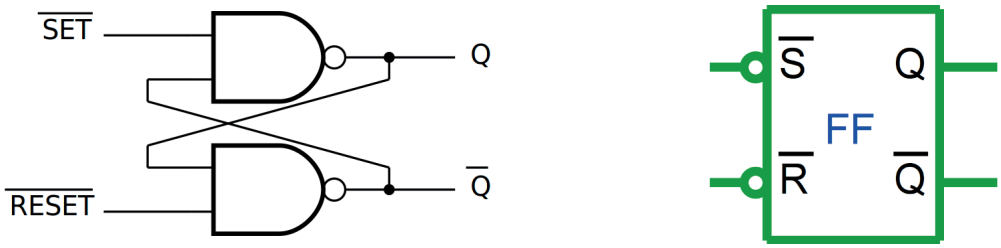


Figura 9.2 – O diagrama esquemático e símbolo do flip-flop SR com entrada ativa em baixo.

Entradas		Saídas		Comentário
S	R	Q	Q'	
0	0	1	1	Inválido.
0	1	1	0	Ativação.
1	0	0	1	Desativação.
1	1	Q_0	Q_0'	Sem mudança.

Tabela 9.2 – A tabela verdade do flip-flop SR com entrada ativa em baixo.

Como pode ser visto nas explicações anteriores, há diferenças entre um FF SR de entrada ativa em alto e um FF SR de entrada ativa em baixo. Esses flip-flops SR funcionam de maneira assíncrona. Isso significa que, quando os valores lógicos aplicados às entradas S e R forem alterados, as saídas serão alteradas de acordo. É possível obter um FF SR trabalhando de maneira síncrona. Isto é conseguido introduzindo uma entrada de controle adicional para um FF SR. Esse FF SR é mostrado na figura 9.3. A tabela verdade deste flip-flop é fornecida na tabela 9.3. Esse flip-flop é chamado de flip-flop SR gatilhável, que requer uma entrada de habilitação ("clock"). As entradas S e R controlam o estado para o qual o FF irá quando um nível alto (lógico 1) é aplicado à entrada CLK. O FF não mudará até CLK = 1; mas enquanto permanecer alto, a saída é controlada pelo estado das entradas S e R. Neste circuito, o estado inválido ocorre quando ambos S e R são simultaneamente altos.

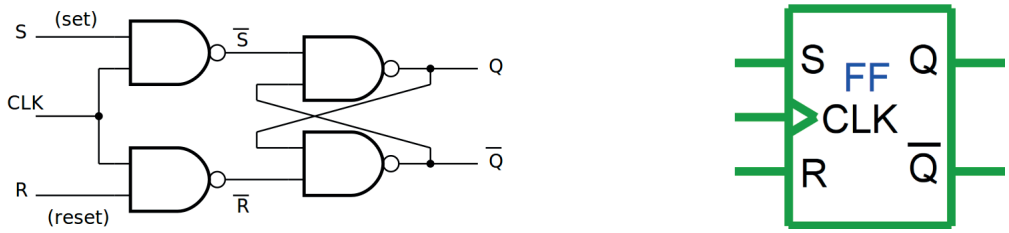


Figura 9.3 – O diagrama esquemático e símbolo do flip-flop SR gatilhável.

Entradas			Saídas		Comentário
CLK	S	R	Q	Q'	
0	X	X	Q_0	Q_0'	Sem mudança.
1	0	0	Q_0	Q_0'	Sem mudança.
1	0	1	0	1	Desativação.
1	1	0	1	0	Ativação.
1	1	1	1	1	Inválido.

Tabela 9.3 – A tabela verdade do flip-flop SR gatilhável.

9.1.3. Exame do flip-flop SR com entradas ativas em alto

Esquemas:

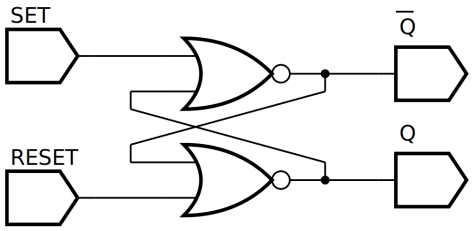


Figura 9.4 – Diagrama esquemático.

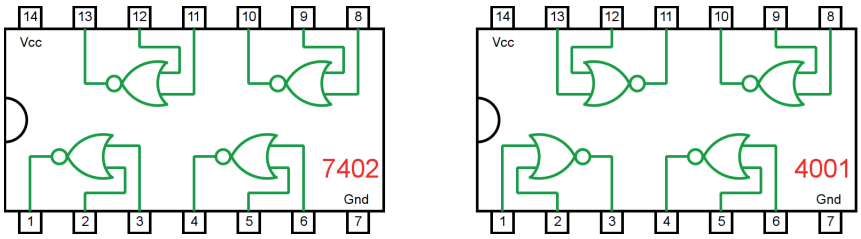


Figura 9.5 – Circuitos integrados TTL e CMOS.

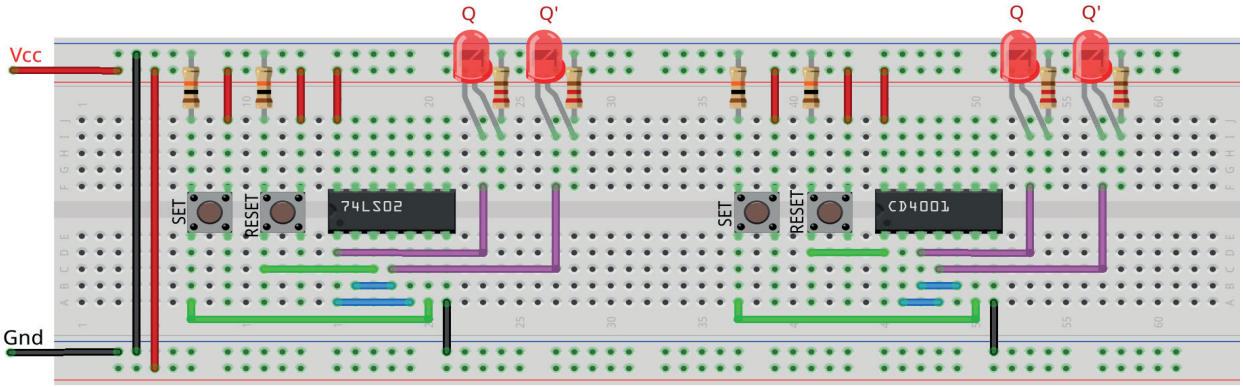


Figura 9.6 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 9.6.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes) entre os botões e o CI
   d) Conecte os fios de conexão (azuis) entre o CI.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).

f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.

g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.

2. Para o preenchimento das tabelas 9.4 e 9.5, use os botões como segue:
   a) Se o valor da variável for “0” NÃO pressione o botão.
   b) Se o valor da variável for “1” pressione o botão.
3. Para o preenchimento das colunas de saída nas tabelas 9.4 e 9.5, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 9.4. Compare os resultados com a tabela 9.4 ou 9.5.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 9.4 ou 9.5.

Tabelas de dados:

Passo	Entradas		Saídas	
	S	R	Q	Q’
1	0	1		
2	0	0		
3	1	0		
4	0	0		
5	1	1		

Tabela 9.4 – Flip-flop RS com CI TTL.

Passo	Entradas		Saídas	
	S	R	Q	Q’
1	0	1		
2	0	0		
3	1	0		
4	0	0		
5	1	1		

Tabela 9.5 – Flip-flop RS com CI CMOS.

Programa para o Arduíno:

```
// teste do flip-flop SR com portas NOU

byte q0; byte q1;
byte ds[] = {0,0,1,0,0,0,1,0,1,0};
byte dr[] = {1,0,0,0,1,0,1,0,0,1};
byte q = 10;
byte i;

void ffsr_nou(byte s, byte r, byte q0i, byte q1i){
  q1 = !(q0i||s); if (q1i != q1) q1i = q1;
  q0 = !(q1i||r); if (q0i != q0) q0i = q0;
  q1 = !(q0i||s);
}

void mostra_ffsr_nou(){
  if (ds[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dr[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (q0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Flip-flop SR com portas NOU");
  Serial.println("| S | R || Q | Q' |");
  Serial.println("----------------------");
  q0=0;
  q1=1;
  for (i=0;i<q;i++){
    ffsr_nou(ds[i],dr[i],q0,q1);
    mostra_ffsr_nou();
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

9.1.4. Exame do flip-flop SR com entradas ativas em baixo

Esquemas:

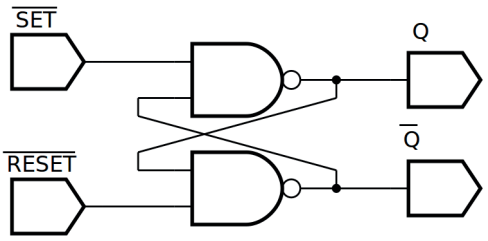


Figura 9.7 – Diagrama esquemático.

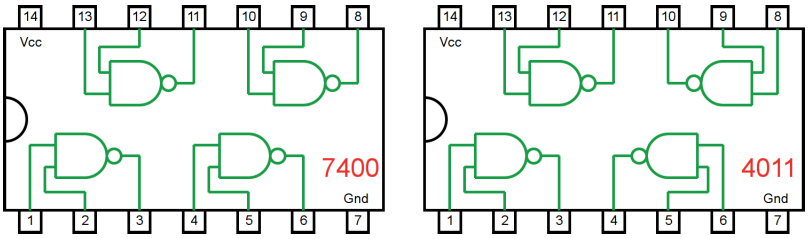


Figura 9.8 – Circuitos integrados TTL e CMOS.

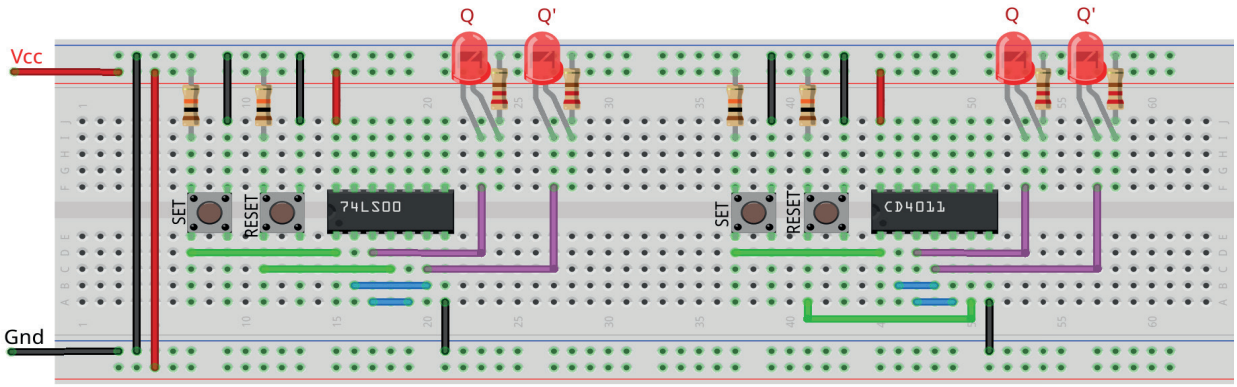


Figura 9.9 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 9.9.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes) entre os botões e o CI
   d) Conecte os fios de conexão (azuis) entre o CI.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).

f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.

g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.

2. Para o preenchimento das tabelas 9.6 e 9.7, use os botões como segue:

a) Se o valor da variável for “0” NÃO pressione o botão.

b) Se o valor da variável for “1” pressione o botão.

3. Para o preenchimento das colunas de saída nas tabelas 9.6 e 9.7, considere:

a) Se o LED estiver apagado então preencher com “0”.

b) Se o LED estiver aceso então preencher com “1”.

4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 9.7. Compare os resultados com a tabela 9.6 ou 9.7.

5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 9.6 ou 9.7.

Tabelas de dados:

Passo	Entradas		Saídas	
	S’	R’	Q	Q’
1	0	1		
2	1	1		
3	1	0		
4	1	1		
5	0	0		

Tabela 9.6 – Flip-flop RS com CI TTL.

Passo	Entradas		Saídas	
	S’	R’	Q	Q’
1	0	1		
2	1	1		
3	1	0		
4	1	1		
5	0	0		

Tabela 9.7 – Flip-flop RS com CI CMOS.

Programa para o Arduíno:

```
// teste do flip-flop SR com portas NE

byte q0; byte q1;
byte ds[] = {0,1,1,1,0,1,1,1,1,0};
byte dr[] = {1,1,0,1,0,1,0,1,0,1};
byte q = 10;
byte i;

void ffsr_ne(byte s, byte r, byte q0i, byte q1i){
  q1 = !(q0i&&r); if (q1i != q1) q1i = q1;
  q0 = !(q1i&&s); if (q0i != q0) q0i = q0;
  q1 = !(q0i&&r);
}

void mostra_ffsr_ne(){
  if (ds[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dr[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (q0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Flip-flop SR com portas NE");
  Serial.println("| S' | R' || Q | Q' |");
  Serial.println("----------------------");
  q0=0;
  q1=1;
  for (i=0;i<q;i++){
    ffsr_ne(ds[i],dr[i],q0,q1);
    mostra_ffsr_ne();
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

9.1.5. Exame do flip-flop SR gatilhável com entradas ativas em alto

Esquemas:

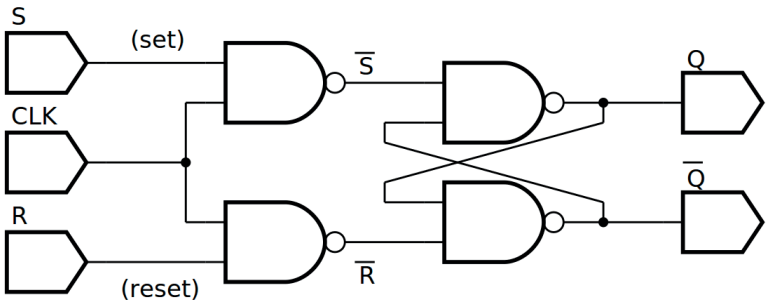


Figura 9.10 – Diagrama esquemático.

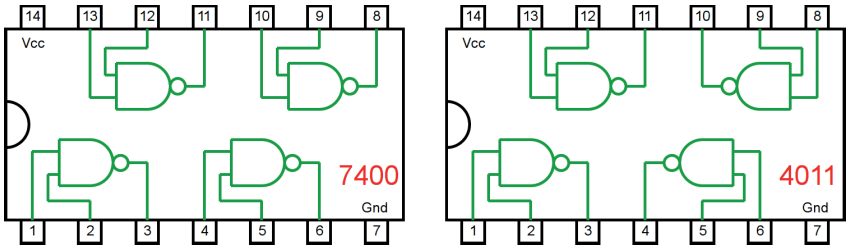


Figura 9.11 – Circuitos integrados TTL e CMOS.

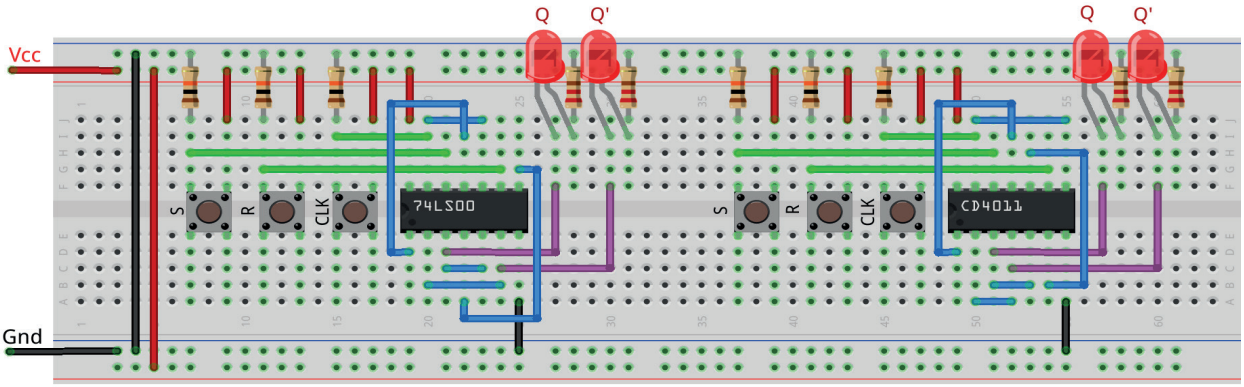


Figura 9.12 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 9.12.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes) entre os botões e o CI
   d) Conecte os fios de conexão (azuis) entre o CI.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento das tabelas 9.8 e 9.9, use os botões como segue:
   a) Se o valor da variável for "0" NÃO pressione o botão.
   b) Se o valor da variável for "1" pressione o botão.
3. Para o preenchimento das colunas de saída nas tabelas 9.8 e 9.9, considere:
   a) Se o LED estiver apagado então preencher com "0".
   b) Se o LED estiver aceso então preencher com "1".
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 9.10. Compare os resultados com a tabela 9.8 ou 9.9.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 9.8 ou 9.9.

Tabelas de dados:

Passo	Entradas			Saídas	
	S	R	CLK	Q	Q'
1	1	0	1		
2	0	0	1		
3	0	1	1		
4	0	0	1		
5	1	1	1		
6	1	0	1		
7	1	0	0		
8	0	0	0		
9	0	1	0		
10	0	0	0		
11	1	1	0		

Tabela 9.8 – Flip-flop RS gatilhável com CI TTL.

Passo	Entradas			Saídas	
	S	R	CLK	Q	Q’
1	1	0	1		
2	0	0	1		
3	0	1	1		
4	0	0	1		
5	1	1	1		
6	1	0	1		
7	1	0	0		
8	0	0	0		
9	0	1	0		
10	0	0	0		
11	1	1	0		

Tabela 9.9 – Flip-flop RS gatilhável com CI CMOS.

Programa para o Arduíno:

```
// teste do flip-flop SR + clock com portas NE

byte q0; byte q1; byte si; byte ri; byte clock;
byte ds[] = {1,0,0,0,1,1,1,0,0,0,1};
byte dr[] = {0,0,1,0,1,0,0,0,1,0,1};
byte cl[] = {1,1,1,1,1,1,0,0,0,0,0};
byte q = 11;
byte i;

void ffsrc_ne(byte s, byte r, byte clk, byte q0i, byte q1i){
  si = !(s&&clk);
  ri = !(r&&clk);
  q1 = !(q0i&&ri); if (q1i != q1) q1i = q1;
  q0 = !(q1i&&si); if (q0i != q0) q0i = q0;
  q1 = !(q0i&&ri);
}

void mostra_ffsrc_ne(){
  if (ds[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dr[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (cl[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (q0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Flip-flop SR + clock com portas NE");
  Serial.println("| S | R | CL || Q | Q' |");
  Serial.println("---------------------------");
  q0=0;
  q1=1;
  for (i=0;i<q;i++){
    ffsrc_ne(ds[i],dr[i],cl[i],q0,q1);
    mostra_ffsrc_ne();
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

9.2. Flip-flop JK

9.2.1. Objetivos

1. Conhecer o flip-flop JK, verificar sua operação lógica e obter sua tabela verdade,
2. Familiarizar-se com o flip-flop JK TTL 74LS76.

9.2.2. Informação preliminar

O flip-flop JK é versátil e é um tipo de flip-flop amplamente utilizado. O funcionamento do FF JK é idêntico ao do FF SR nas condições de operação de ativar (SET), desativar (RESET) e sem alteração. A diferença é que o FF JK não tem um estado inválido como o FF SR. As entradas J e K se comportam como entradas S e R para ativar e limpar (desativar) o flip-flop, respectivamente. A entrada marcada com J é para ativar ("SET") e a entrada marcada com K é para desativar ("RESET"). Quando ambas as entradas J e K são iguais a 1, o flip-flop muda para o seu estado de complemento, isto é, se Q = 1, muda para Q = 0 e vice-versa. Existem diferentes tipos de FF JK. Os flip-flops podem ser classificados em dois grupos com base em como eles são acionados: acionados por borda e acionados por nível. O primeiro grupo pode ser dividido em dois subgrupos, como flip-flops gatilháveis por borda ascendente (ou borda positiva) e borda descendente (ou borda negativa). Este fato é tornado visível com o símbolo " > " em seu símbolo esquemático. O gatilho, ou disparo, por borda ascendente (ou borda positiva) é simbolizado por " ⌋" ou " ↥ ". Os flip-flops gatilháveis pela borda ascendente mudam seus estados quando o estado do pulso de gatilho ("clock" ou CLK) é alterado de desligado ("OFF") para ligado ("ON"). Da mesma forma, o gatilho de borda descendente é simbolizado por "⌊" ou " ↧ ". Os flip-flops acionados pela borda descendente mudam seus estados quando o estado do pulso de gatilho ("clock" ou CLK) é alterado de ligado ("ON") para desligado ("OFF"). O segundo grupo pode ser dividido em dois subgrupos como gatilho de nível lógico 1 (⊓) e gatilho de nível lógico 0 (⊔). Os flip-flops acionados no nível lógico "1" mudam seus estados quando o pulso de gatilho ("clock" ou CLK) = 1. Da mesma forma, os flip-flops acionados no nível lógico "0" mudam seus estados quando o pulso de gatilho ("clock" ou CLK) = 0. A figura 9.13 mostra o diagrama lógico e o símbolo do flip-flop JK gatilhável por borda ascendente. A tabela verdade do flip-flop JK gatilhável por borda ascendente é fornecida na tabela 9.10. Existem vários circuitos esquemáticos possíveis para implementação do flip-flop JK. Uma versão mais compacta pode ser vista na figura 9.14. Normalmente, os sistemas de geração de sinal de relógio é uma forma de onda quadrada, com o tempo em alto igual ao tempo em baixo. Para garantir que o pulso de gatilho seja rápido o suficiente para mudar o estado do flip-flop e prevenir mudanças dos sinais de J e K durante o nível de habilitação, um circuito adicional produz um disparo de curta duração utilizando o atraso de propagação das portas lógicas. A figura 9.15 mostra os circuitos para gatilho de borda ascendente e descendente.

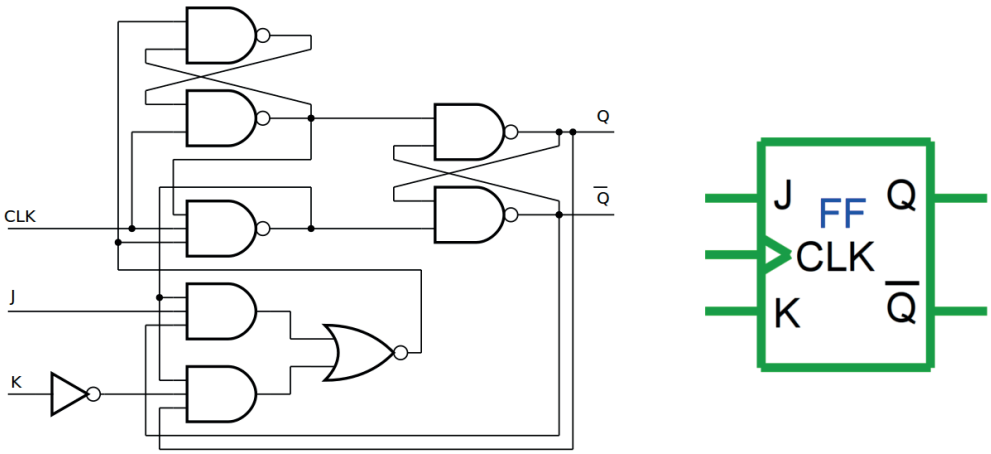


Figura 9.13 – O diagrama esquemático e o símbolo do flip-flop JK gatilhável por borda ascendente.

Entradas			Saídas		Comentário
CLK	J	K	Q	Q'	
0	X	X	Q_0	Q_0'	Sem mudança.
1	X	X	Q_0	Q_0'	Sem mudança.
↓	X	X	Q_0	Q_0'	Sem mudança.
↑	0	0	Q_0	Q_0'	Sem mudança.
↑	0	1	0	1	Desativação.
↑	1	0	1	0	Ativação.
↑	1	1	Q_0'	Q_0	Troca.

Tabela 9.10 – A tabela verdade do flip-flop de JK gatilhável por borda ascendente.

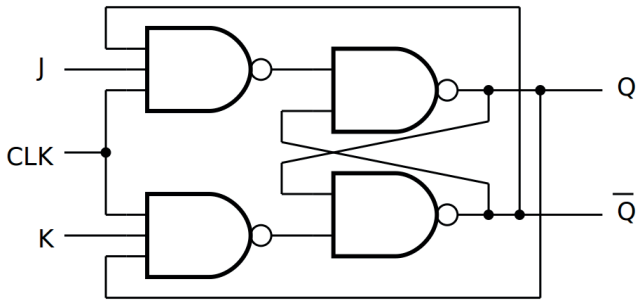


Figura 9.14 – Outra forma para o diagrama esquemático do flip-flop JK gatilhável por borda ascendente.

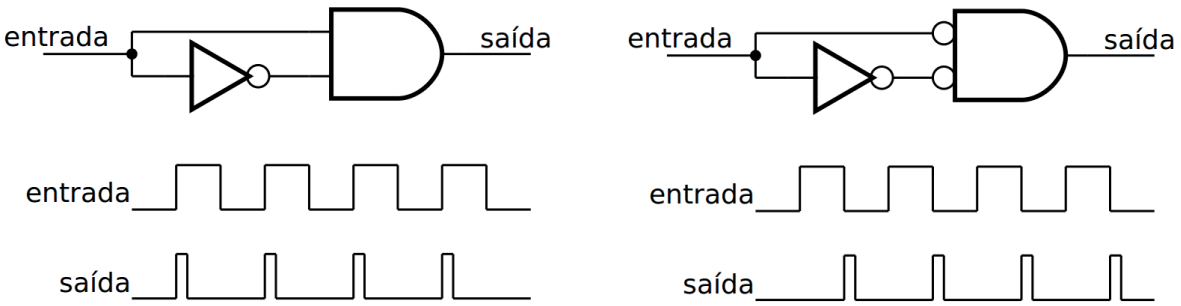


Figura 9.15 – Circuitos de gatilho para borda ascendente e borda descendente.

Pode ser visto que as entradas J e K são efetivas nas bordas crescentes da entrada CLK. Isso significa que as entradas J e K são síncronas com a entrada CLK. Portanto, essas entradas são chamadas entradas síncronas, porque os dados nessas entradas são transferidos para a saída do flip-flop na borda de disparo do pulso de clock; isto é, os dados são transferidos em sincronia com o relógio. A maioria dos circuitos integrados com flip-flops também possuem entradas assíncronas. Estas são entradas que afetam o estado do flip-flop independente do relógio. Elas são normalmente rotulados como PRESET (ou PRE ou S) e CLEAR (CLR ou R). Um nível ativo (alto ou baixo) na entrada PRESET ativa o flip-flop (Q = 1), e um nível ativo (alto ou baixo) na entrada CLEAR desativa o flip-flop (Q=0). A figura 9.16 mostra o flip-flop JK gatilhável por borda descendente com entradas assíncronas ativas em baixo. A tabela verdade deste flip-flop é fornecida na tabela 9.11.

Neste flip-flop: quando S’ = 0 e R’ = 1 a saída é ativada (Q = 1). Quando S’ = 1 e R’ = 0 a saída é desativada (Q = 0). Quando S’= 0 e R’ = 0, tanto a saída como seu complemento são ativados (Q = 1 e Q’ = 1). Esta é uma operação inválida. Quando S’ = 1 e R’ = 1, ambas as entradas assíncronas não estão ativas. Nesse caso, quando o estado do sinal de relógio (“clock” ou CLK) é alterado de ligado (“ON”) para desligado (“OFF”) (↴): se JK = 00, nenhuma alteração de estado é emitida; se JK = 01, então Q é desativado; se JK = 10, então Q é ativado; e finalmente se JK = 11, então Q é alternado. No modo de alternância, a frequência do sinal de entrada é dividida por dois. Devido a esse recurso, os flip-flops JK são usados para obter contadores.

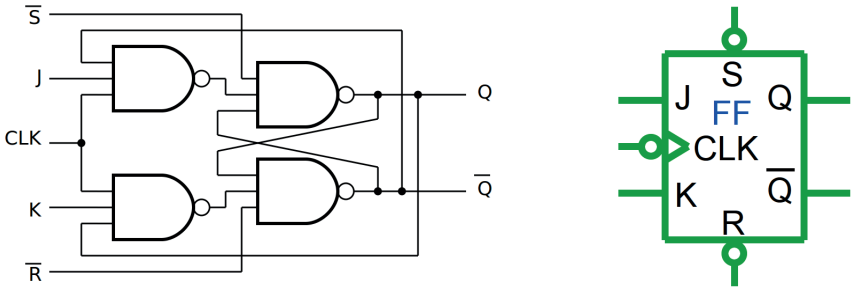


Figura 9.16 – O diagrama esquemático e símbolo do flip-flop JK gatilhável por borda descendente com entradas assíncronas ativas em baixo.

Entradas					Saídas		Comentário
S’	R’	CLK	J	K	Q	Q’	
0	1	X	X	X	1	0	Ativação.
1	0	X	X	X	0	1	Desativação.
0	0	X	X	X	1	1	Inválido.
1	1	0	X	X	Q_0	Q_0’	Sem mudança.
1	1	1	X	X	Q_0	Q_0’	Sem mudança.
1	1	↑	X	X	Q_0	Q_0’	Sem mudança.
1	1	↓	0	0	Q_0	Q_0’	Sem mudança.
1	1	↓	0	1	0	1	Desativação.
1	1	↓	1	0	1	0	Ativação.
1	1	↓	1	1	Q_0’	Q_0	Troca.

Tabela 9.11 – A tabela verdade do flip-flop JK gatilhável por borda descendente com entradas assíncronas ativas em baixo.

Além dos flip-flops JK gatilháveis por borda ascendente, borda descendente, nível lógico 1 e nível lógico 0, também existem os flip-flops JK mestre-escravo. Eles podem ser flip-flops JK mestre-escravo gatilháveis por pulso positivo ou por pulso negativo. Os dados J e K são processados pelo flip-flop após um pulso de relógio (“clock”) completo. Enquanto o relógio está BAIXO, o escravo é isolado do mestre. Na transição positiva do relógio (↑), os dados das entradas J e K são transferidos para o mestre. Enquanto o relógio estiver ALTO (⊓), as entradas J e K estarão desativadas. Na transição negativa do relógio (↓), os dados do mestre são transferidos para o escravo. O estado lógico das entradas J e K não podem mudar enquanto o relógio estiver ALTO (⊓). Os dados são transferidos para as saídas na borda descendente do pulso de relógio (“clock”). Um nível lógico BAIXO nas entradas PR ou CLR irá ativar ou desativar as saídas, independentemente dos níveis lógicos das outras entradas. A figura 9.17 mostra o diagrama esquemático de um flip-flop JK mestre-escravo com entradas assíncronas de Preset (PR’) e Clear (CLR’) ativas em baixo. A tabela 9.12 mostra as funções válidas para o flip-flop JK mestre-escravo com entradas assincronas.

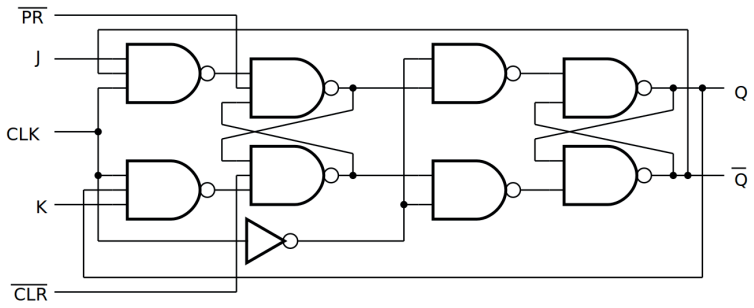


Figura 9.17 – O diagrama esquemático de um flip-flop JK mestre-escravo.

Entradas					Saídas		Comentário
PR'	CLR'	CLK	J	K	Q	Q'	
0	1	X	X	X	1	0	Ativação.
1	0	X	X	X	0	1	Desativação.
0	0	X	X	X	1	1	Inválido.
1	1	↥↧	0	0	Q_0	Q_0'	Sem mudança.
1	1	↥↧	0	1	0	1	Desativação.
1	1	↥↧	1	0	1	0	Ativação.
1	1	↥↧	1	1	Q_0'	Q_0	Troca.

Tabela 9.12 – A tabela de funções de um flip-flop JK mestre-escravo.

9.2.3. Circuito Integrado TTL 74LS76

Como exemplo, a figura 9.18 mostra o diagrama esquemático e pinagem do CI TTL 74LS76 com dois flip-flops JK mestre-escravo acionados por pulso positivo. Este dispositivo contém dois flip-flops J-K independentes acionados por impulso positivo (↥↧) com saídas complementares.

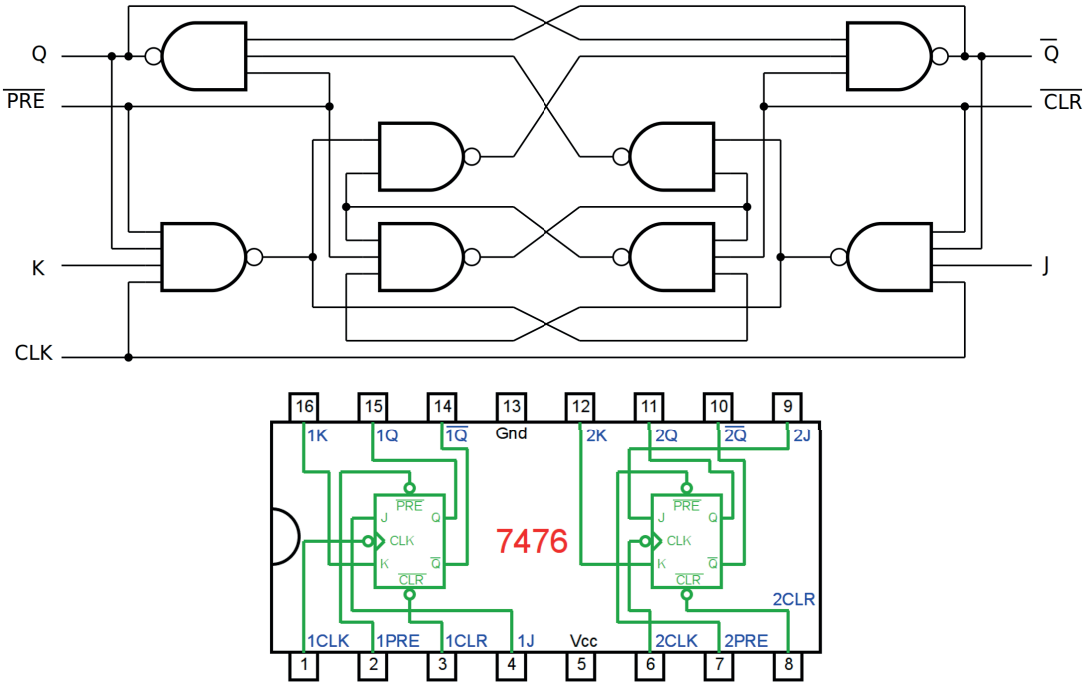


Figura 9.18 – O diagrama esquemático e pinagem do CI TTL 74LS76 com dois flip-flop JK mestre-escravo.

9.2.4. Exame do CI TTL 74LS76 com dois flip-flop JK mestre-escravo

Esquemas:

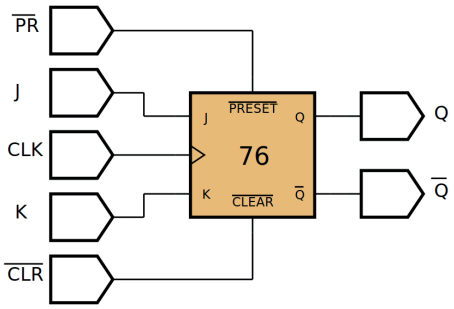


Figura 9.19 – Diagrama esquemático.

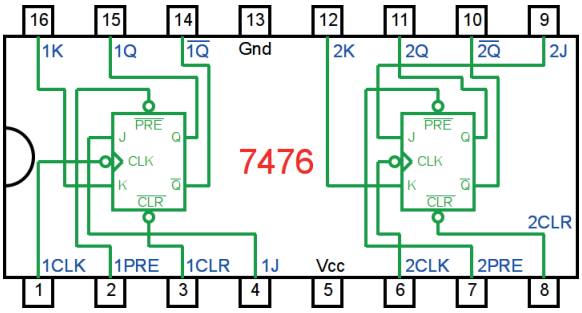


Figura 9.20 – Circuito integrado TTL.

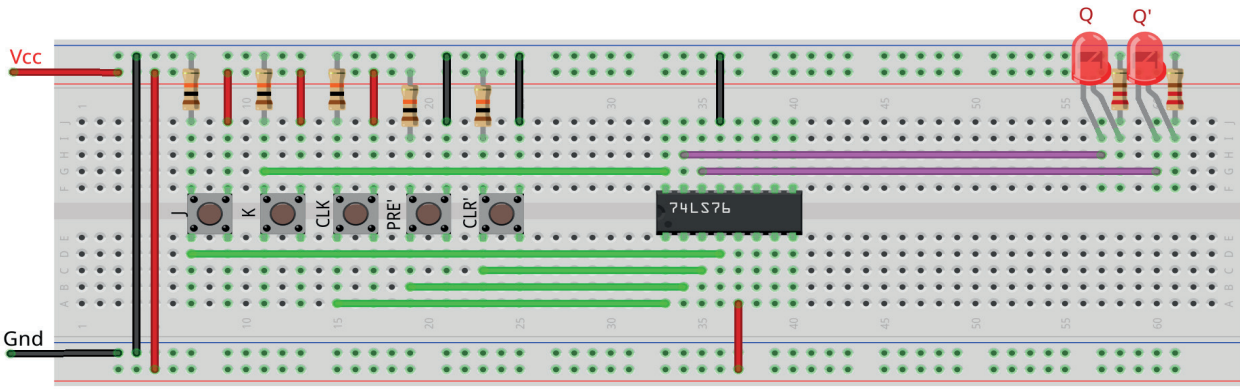


Figura 9.21 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 9.21.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes) entre os botões e o CI
   d) Conecte os fios de conexão (azuis) entre o CI.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 9.13, use os botões como segue:
   a) Para as variáveis PRE' e CLR' um "0" significa que o botão deve ser pressionado.
   b) Para as variáveis J e K um "1" significa que o botão deve ser pressionado.
   c) Para a variável CLK o sinal "↥↧" significa que o botão deve ser pressionado e solto após os botões J e K serem pressionados.
3. Para o preenchimento das colunas de saída na tabela 9.13, considere:
   a) Se o LED estiver apagado então preencher com "0".
   b) Se o LED estiver aceso então preencher com "1".
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 9.19. Compare os resultados com a tabela 9.13.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 9.13.

Tabelas de dados:

Início		Entradas			Saídas		
PRE'	CLR'	J	K	CLK	Q	Q'	Comentários
1	0	X	X	X			
0	1	X	X	X			
0	0	X	X	X			
1	1	0	1	↑↓			
1	1	0	0	↑↓			
1	1	1	0	↑↓			
1	1	0	0	↑↓			
1	1	1	1	↑↓			

Tabela 9.13 – Flip-flop JK com entradas assíncronas.

Programa 1 para o Arduíno:

```
// teste do flip-flop JK mestre escravo + clock com portas NE

byte p1 = 1; byte p2 = 1; byte p3 = 1; byte p4 = 1;
byte p5 = 0; byte p6 = 0; byte p7 = 1; byte p8 = 1;
byte q0; byte q1; byte i; byte cl;
byte p3i; byte p4i; byte p7i; byte p8i;
byte ds[] = {0,0,1,1,1,1,1,1,1,1,1,1,1};
byte dr[] = {0,1,1,0,1,1,1,1,1,1,1,1,1};
byte dj[] = {0,0,0,0,0,1,0,0,0,1,0,1,1};
byte dk[] = {0,0,0,0,0,0,0,1,0,1,0,1,1};
byte q = 13;

void ffjkmec_ne(byte pre, byte clr, byte j, byte k, byte clk){
  p1 = !((j&&clk)&&p8);
  p2 = !((k&&clk)&&p7);
  p3i = p3; p4i = p4;
  p3 = !((p4i&&p1)&&pre); if(p3i != p3) p3i = p3;
  p4 = !((p3i&&p2)&&clr); if(p4i != p4) p4i = p4;
  p3 = !((p4i&&p1)&&pre);
  p5 = !(p3&&!clk);
  p6 = !(p4&&!clk);
  p7i = p7; p8i = p8;
  p7 = !(p5&&p8i); if(p7i != p7) p7i = p7;
  p8 = !(p6&&p7i); if(p8i != p8) p8i = p8;
```

```
  p7 = !(p5&&p8i);
  q0 = p7;
  q1 = p8;
}

void mostra_ffjkmec_ne(){
  if (ds[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dr[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dj[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dk[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (!cl) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (q0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Flip-flop JK mestre-escravo + clock com portas NE");
  Serial.println("| S' | R' | J | K | CL || Q | Q' |");
  Serial.println("-------------------------------------");
  for (i=0;i<q;i++){
    for (cl = 0; cl <= 1; cl++){
      ffjkmec_ne(ds[i],dr[i],dj[i],dk[i],!cl);
      mostra_ffjkmec_ne();
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

Programa 2 para o Arduíno:

```
// teste do flip-flop JK mestre escravo sem portas lógicas
byte q0; byte q1; byte i; byte clock; byte cla; byte cld;
byte dpre[] = {0,0,1,1,1,1,1,1,1,1,1,1,1};
byte dcle[] = {0,1,1,0,1,1,1,1,1,1,1,1,1};
byte dj[] = {0,0,0,0,0,1,0,0,0,1,0,1,1};
byte dk[] = {0,0,0,0,0,0,0,1,0,1,0,1,1};
byte q = 13;
```

```
void ffjk(byte pre, byte clr, byte j, byte k, byte clka, byte clkd,
byte q0i, byte q1i){
  if(!pre && !clr){q0 = 1; q1 = 1; return;}
  if(!pre && clr){q0 = 1; q1 = 0; return;}
  if( pre && !clr){q0 = 0; q1 = 1; return;}
  if( pre && clr){
  if(clka && !clkd){
    if( j && k){q0 = q1i; q1 = q0i; return;}
      if( j && !k){q0 = 1; q1 = 0; return;}
      if(!j && k){q0 = 0; q1 = 1; return;}
    }
  }
}

void mostra_ffjk(){
  if (dpre[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dcle[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dj[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dk[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (cld) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (q0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Flip-flop JK mestre-escravo + clock sem portas lógi-
  cas");
  Serial.println("|PRE'|CLE'| J | K |CLK || Q | Q' |");
  Serial.println("-------------------------------------");
  cld = 0; q0 = 0; q1 = 1;
  for (i=0;i<q;i++){
    for (clock = 0; clock <= 1; clock++){
      cla = cld; cld = !clock;
      ffjk(dpre[i],dcle[i],dj[i],dk[i],cla,cld,q0,q1);
      mostra_ffjk();
    }
  }
} // fim do 'setup'
void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

9.3. Flip-flop D

9.3.1. Objetivos

1. Conhecer o flip-flop D, verificar sua operação lógica e obter sua tabela verdade,
2. Construir um flip-flop D usando um FF JK e um INVERSOR.

9.3.2. Informação preliminar

O flip-flop D é amplamente utilizado. Também é conhecido como um flip-flop de dados. O flip-flop D captura o valor da entrada D em uma parte definida do ciclo do relógio (como a borda de subida do relógio). Esse valor capturado se torna a saída Q. Em outros momentos, a saída Q não muda. O flip-flop D pode ser visto como uma célula de memória. A figura 9.22 mostra o flip--flop D (trava, "latch") gatilhável por nível lógico "1". A tabela verdade deste flip-flop é fornecida na tabela 9.14. A tabela verdade mostra que quando a entrada de relógio ("clock") é baixo "0", a entrada D não tem efeito na saída. Quando a entrada do relógio é alto "1", a saída é igual a D. Um flip-flop do tipo D pode ser obtido facilmente usando um flip-flop SR e um inversor. A figura 9.23 mostra como obter um flip-flop D gatilhável por nível lógico alto "1" usando um flip-flop SR gatilhável.

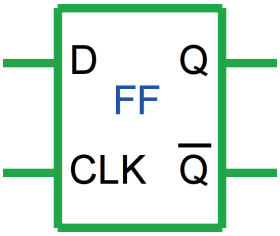


Figura 9.22 – O símbolo do flip-flop D (trava) gatilhável por nível lógico alto "1".

Entradas		Saídas		Comentário
CLK	D	Q	Q'	
0	X	Q_0	Q_0'	Sem mudança.
1	1	1	0	Ativação.
1	0	0	1	Desativação.

Tabela 9.14 – A tabela verdade do flip-flop D (trava) gatilhável por nível lógico alto "1".

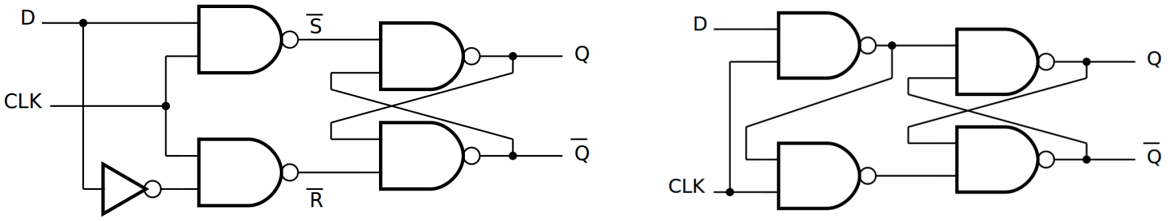


Figura 9.23 – O diagrama esquemático do flip-flop D (trava) gatilhável de nível lógico alto "1" obtido usando um flip-flop SR gatilhável.

Os flip-flops D também podem ser classificados como flip-flops D gatilháveis por borda ascendente (↥), borda descendente (↧), nível lógico 1 (⊓) e nível lógico 0 (⊔). Em geral, os flip-flops D gatilháveis por nível são chamados de travas (“latch”). Os flip-flops D gatilháveis por borda são construídos com o símbolo “ > “ em seu símbolo esquemático. Como exemplo, a figura 9.24 mostra o símbolo do flip-flop D gatilhável pela borda descendente. A tabela verdade deste flip-flop é fornecida na tabela 9.15.

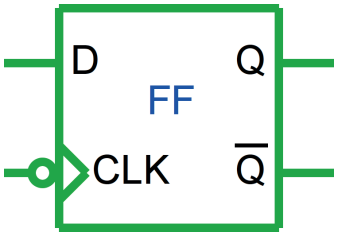


Figura 9.24 – O símbolo do flip-flop D gatilhável por borda descendente.

Entradas		Saídas		Comentário
CLK	D	Q	Q’	
1	X	Q_0	Q_0’	Sem mudança.
0	X	Q_0	Q_0’	Sem mudança.
↥	X	Q_0	Q_0’	Sem mudança.
↧	1	1	0	Ativação.
↧	0	0	1	Desativação.

Tabela 9.15 – A tabela verdade do flip-flop D gatilhável por borda descendente.

Um exemplo de diagrama de tempo para um flip-flop D gatilhável por borda ascendente é dado na figura 9.25. Aqui, a saída Q mantém seu estado atual até a borda de subida do sinal de entrada CLK. Quando o estado do sinal de entrada de relógio CLK é alterado de desligado (“OFF”) para ligado (“ON”) (↥), a saída Q é carregada com o estado da entrada D.

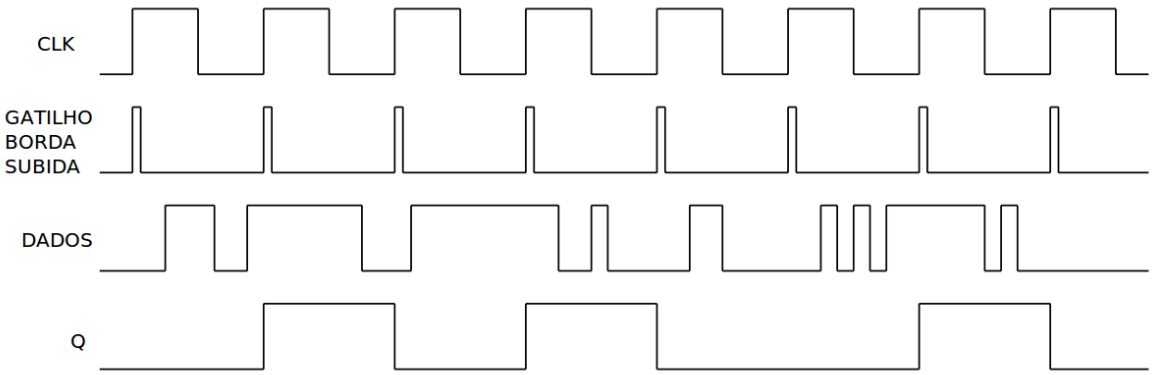


Figura 9.25 – Um exemplo de diagrama de tempo para um flip-flop D gatilhável por borda ascendente.

Um exemplo de diagrama de tempo para um flip-flop D gatilhável por nível lógico alto “1” é dado na figura 9.26. Como pode ser visto na figura quando o sinal de entrada CLK está no nível lógico 1, a saída Q copia e armazena o estado da entrada D. Quando o estado do sinal de entrada do relógio CLK é alterado de ligado (“ON”) para desligado (“OFF”) (↴), a saída Q é travada com o estado mais recente da entrada D e esse valor será mantido até o próximo nível lógico 1 do sinal de entrada CLK.

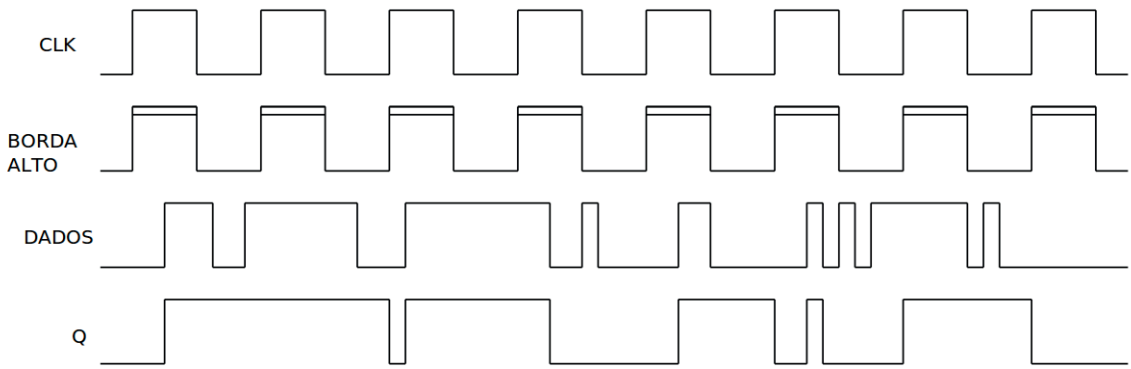


Figura 9.26 – Um exemplo de diagrama de tempo para um flip-flop D gatilhável por nível lógico alto “1”.

A entrada D é síncrona com a entrada CLK. Como explicado para os flip-flops JK, é possível introduzir entradas assíncronas PRESET (ou PRE ou S) e CLEAR (CLR ou R) para flip-flops do tipo D. Elas podem ser ativas em alto ou baixo. Um nível ativo (alto ou baixo) na entrada PRESET irá ativar o flip-flop, e um nível ativo (alto ou baixo) na entrada CLEAR irá desativar o flip-flop. A figura 9.27 mostra o símbolo do flip-flop D com gatilho de borda ascendente com entradas assíncronas ativas em baixo. A tabela verdade deste flip-flop é fornecida na Tabela 9.16. Neste flip-flop: quando S’ = 0 e R’ = 1 a saída é ativada (Q = 1). Quando S’ = 1 e R’ = 0, a saída é desativada (Q = 0). Quando S’ = 0 e R’ = 0, tanto a saída como seu complemento são ativados (Q = 1 e Q’ = 1). Esta é uma operação inválida. Quando S’ = 1 e R’ = 1, ambas as entradas assíncronas não estão ativas. Neste caso, quando o estado do sinal de relógio (“CLK”) é alterado de desligado (“OFF”) para ligado (“ON”) (↱), a saída Q é carregada com o estado da entrada D.

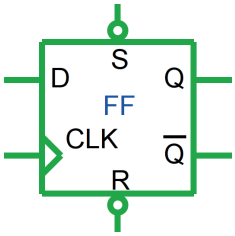


Figura 9.27 – O símbolo do flip-flop D com gatilho de borda ascendente com entradas assíncronas ativas em baixo.

Entradas				Saídas		Comentários
S’	R’	CLK	D	Q	Q’	
0	1	X	X	1	0	Ativação.
1	0	X	X	0	1	Desativação.
0	0	X	X	1	1	Inválido.
1	1	0	X	Q_0	Q_0’	Sem mudança.
1	1	1	X	Q_0	Q_0’	Sem mudança.
1	1	↓	X	Q_0	Q_0’	Sem mudança.
1	1	↑	0	0	1	Desativação.
1	1	↑	1	1	0	Ativação.

Tabela 9.16 – A tabela verdade do flip-flop D com gatilho de borda ascendente com entradas assíncronas ativas em baixo.

9.3.3. Circuito Integrado TTL 74LS74

O circuito integrado TTL 74LS74 possui dois flip-flops D gatilháveis por borda ascendente. O símbolo esquemático e a tabela de funções do 74LS74 são mostrados na figura 9.28. Este dispositivo contém dois independentes flip-flops D acionados pela borda ascendente com saídas complementares. As informações na entrada D são aceitas pelos flip-flops na borda positiva do pulso de relógio ("clock"). O disparo ocorre em um nível de tensão e não está diretamente relacionado ao tempo de transição da borda de subida do relógio. Os dados na entrada D podem ser alterados enquanto o relógio estiver baixo ou alto sem afetar as saídas, desde que a configuração de dados e os tempos de espera não sejam violados. Um nível lógico baixo nas entradas PR ou CLR vai ativar ou desativar as saídas, independentemente dos níveis lógicos das outras entradas.

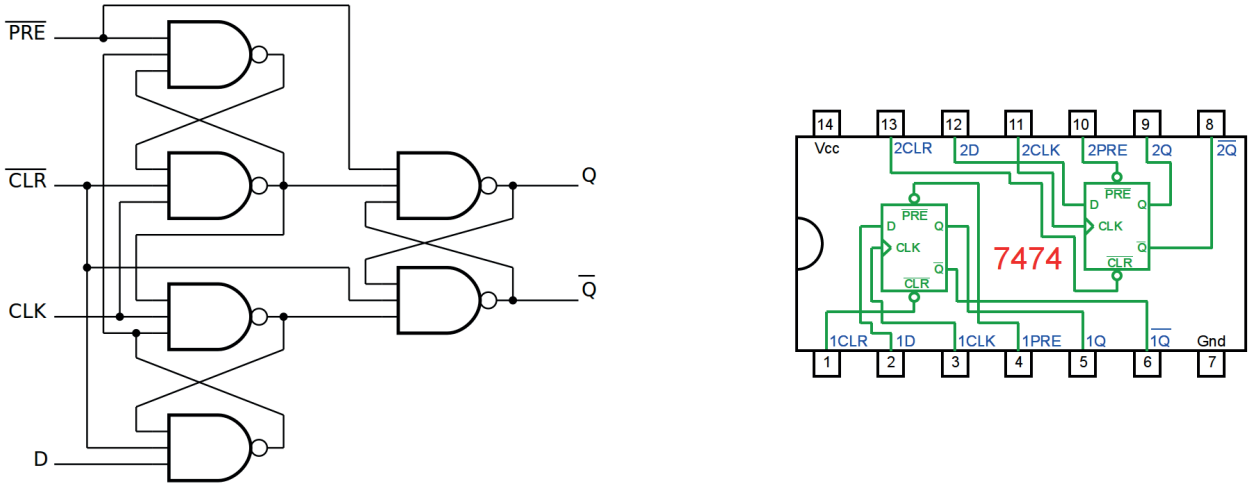


Figura 9.28 – O diagrama esquemático e a pinagem do CI 74LS74 com dois flip-flops D gatilháveis por borda ascendente.

9.3.4. Exame do CI TTL 74LS74 com dois flip-flop D

Esquemas:

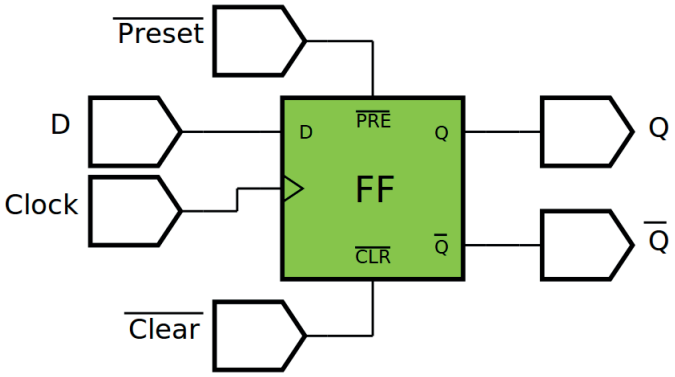


Figura 9.29 – Diagrama esquemático.

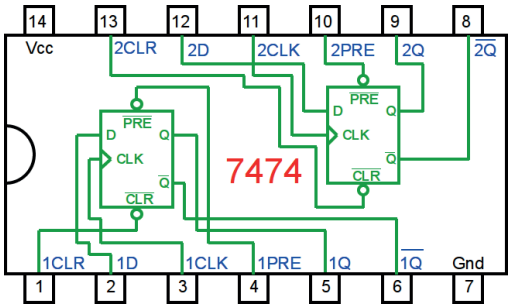


Figura 9.30 – Circuito integrado TTL.

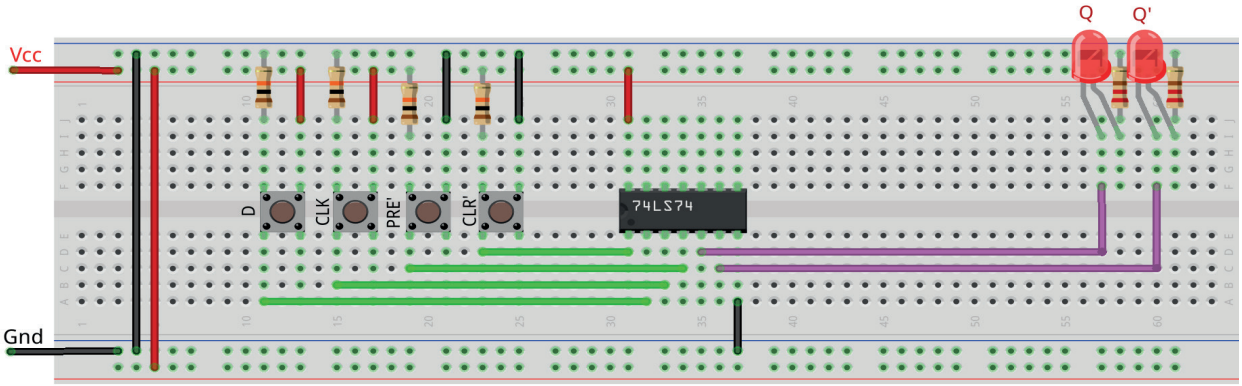


Figura 9.31 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 9.31.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes) entre os botões e o CI
   d) Conecte os fios de conexão (azuis) entre o CI.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 9.17, use os botões como segue:
   a) Para as variáveis PRE' e CLR' um "0" significa que o botão deve ser pressionado.
   b) Para a variável D um "1" significa que o botão deve ser pressionado.
   c) Para a variável CLK o sinal "↥" significa que o botão deve ser pressionado e a análise da saída deve ser executada. Com o sinal "↧" significa que o botão deve ser solto e a análise da saída deve ser executada.
3. Para o preenchimento das colunas de saída na tabela 9.17, considere:
   a) Se o LED estiver apagado então preencher com "0".
   b) Se o LED estiver aceso então preencher com "1".
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 9.29. Compare os resultados com a tabela 9.17.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 9.17.

Tabelas de dados:

Início		Entradas		Saídas		
PRE'	CLR'	D	CLK	Q	Q'	Comentários
0	0	X	X			
1	0	X	X			
0	1	X	X			
1	1	0	0			
1	1	1	0			
1	1	0	1			
1	1	1	1			
1	1	0	↧			
1	1	1	↧			
1	1	0	↥			
1	1	1	↥			

Tabela 9.17 – Flip-flop D com entradas assíncronas.

Programa para o Arduíno:

```
// teste do flip-flop D (sem usar portas lógicas!)

byte i; byte q0; byte q1; byte cl;
byte dpre[] = {0,1,1,0,1,1,1,1,1,1,1,1};
byte dclr[] = {0,0,1,1,1,1,1,1,1,1,1,1};
byte dd[] = {0,0,0,0,0,1,0,1,0,1,0,1};
byte q = 12;

void ffd(byte pre, byte clr, byte d, byte clk){
  if(!pre && !clr){q0 = 1; q1 = 1; return;}
  if(!pre && clr){q0 = 1; q1 = 0; return;}
  if( pre && !clr){q0 = 0; q1 = 1; return;}
  if(clk){q0 = d; q1 = !d;}
}

void mostra_ffd(){
  if (dpre[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dclr[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dd[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (!cl) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (q0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
  Serial.begin(9600);
  Serial.println("Flip-flop D");
  Serial.println("|PRE'|CLR'| D |CLK || Q | Q' |");
  Serial.println("--------------------------------");
  for (i=0;i<q;i++){
    for (cl = 0; cl <= 1; cl++){
      ffd(dpre[i],dclr[i],dd[i],!cl);
      mostra_ffd();
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

9.3.5. Converter um flip-flop JK em um flip-flop D

Esquemas:

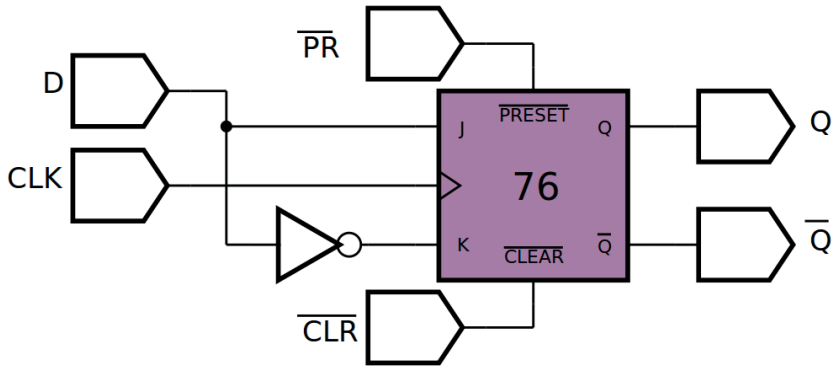


Figura 9.32 – Diagrama esquemático.

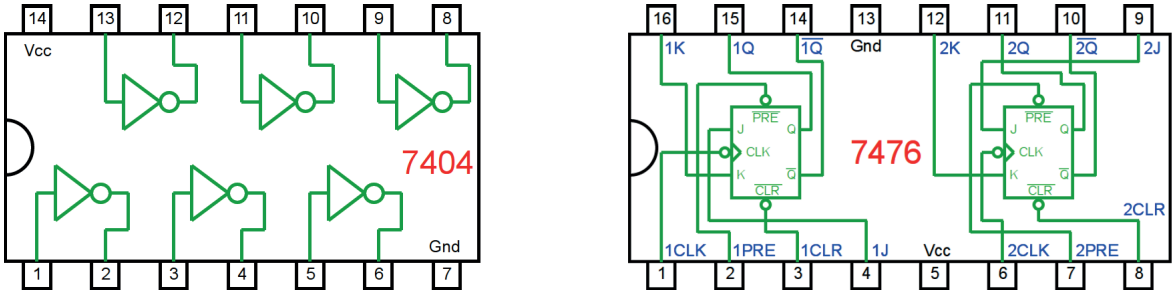


Figura 9.33 – Circuitos integrados TTL.

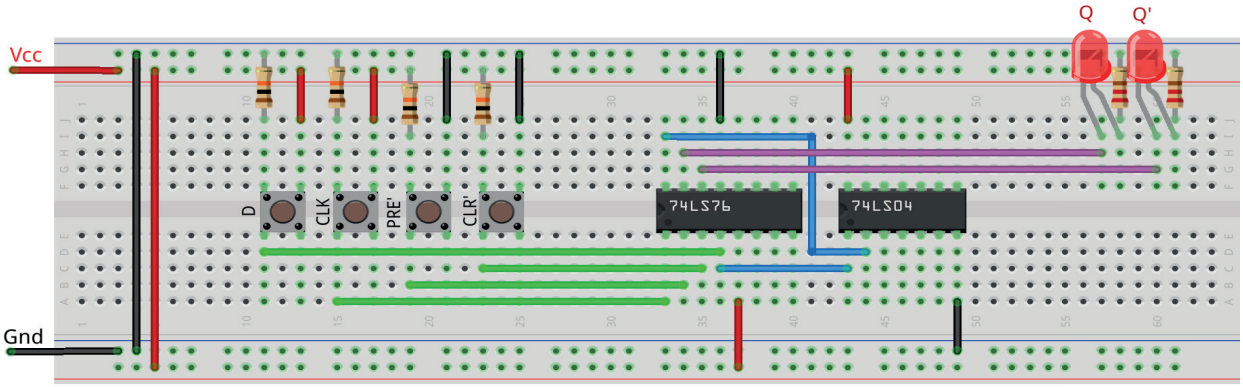


Figura 9.34 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 9.34.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes) entre os botões e o CI
   d) Conecte os fios de conexão (azuis) entre o CI.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 9.18, use os botões como segue:
   a) Para as variáveis PRE' e CLR' um "0" significa que o botão deve ser pressionado.
   b) Para a variável D um "1" significa que o botão deve ser pressionado.
   c) Para a variável CLK o sinal "↥↧" significa que o botão deve ser pressionado e solto. A entrada D não pode variar durante a execução do sinal de CLK.
3. Para o preenchimento das colunas de saída na tabela 9.18, considere:
   a) Se o LED estiver apagado então preencher com "0".
   b) Se o LED estiver aceso então preencher com "1".
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 9.32. Compare os resultados com a tabela 9.18.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 9.18.

Tabelas de dados:

Início		Entradas		Saídas		
PRE'	CLR'	D	CLK	Q	Q'	Comentários
0	0	X	X			
0	1	X	X			
1	0	X	X			
1	1	0	↥↧			
1	1	1	↥↧			

Tabela 9.18 – Flip-flop D com entradas assíncronas.

Programa para o Arduíno:

```
// teste do flip-flop D com flip-flop JK mestre escravo + clock com
portas NE

byte p1 = 1; byte p2 = 1; byte p3 = 1; byte p4 = 1;
byte p5 = 0; byte p6 = 0; byte p7 = 1; byte p8 = 1;
byte q0; byte q1; byte i; byte cl;
byte p3i; byte p4i; byte p7i; byte p8i;
byte ds[] = {0,0,1,1,1,1,1,1,1,1,1,1,1};
byte dr[] = {0,1,1,0,1,1,1,1,1,1,1,1,1};
byte dd[] = {0,0,0,0,0,1,0,1,0,1,0,1,0};
byte q = 13;

void ffjkmec_ne(byte pre, byte clr, byte j, byte k, byte clk){
  p1 = !((j&&clk)&&p8);
  p2 = !((k&&clk)&&p7);
  p3i = p3; p4i = p4;
  p3 = !((p4i&&p1)&&pre); if(p3i != p3) p3i = p3;
  p4 = !((p3i&&p2)&&clr); if(p4i != p4) p4i = p4;
  p3 = !((p4i&&p1)&&pre);
  p5 = !(p3&&!clk);
  p6 = !(p4&&!clk);
  p7i = p7; p8i = p8;
  p7 = !(p5&&p8i); if(p7i != p7) p7i = p7;
  p8 = !(p6&&p7i); if(p8i != p8) p8i = p8;
  p7 = !(p5&&p8i);
  q0 = p7;
  q1 = p8;
}

void mostra_ffjkmec_ne(){
  if (ds[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dr[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dd[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (!cl) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (q0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}
```

```
void setup(){
  Serial.begin(9600);
  Serial.println("Flip-flop D com flip-flop JK mestre-escravo + clock
  com portas NE");
  Serial.println("|PRE'|CLR'| D |CLK || Q | Q' |");
  Serial.println("--------------------------------");
  for (i=0;i<q;i++){
    for (cl = 0; cl <= 1; cl++){
      ffjkmec_ne(ds[i],dr[i],dd[i],!dd[i],!cl);
      mostra_ffjkmec_ne();
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

9.4. Flip-flop T

9.4.1. Objetivos

1. Conhecer o flip-flop T,
2. Verificar sua operação lógica e obter sua tabela verdade.

9.4.2. Informação preliminar

O flip-flop T é uma versão de entrada única do flip-flop JK. O flip-flop T é obtido do tipo JK se ambas as entradas estiverem conectadas juntas. Portanto, o flip-flop T não possui um estado inválido. Um flip-flop T possui duas entradas (T e CLK) e duas saídas (Q e seu complemento Q'). Se a entrada T for alta "1", o flip-flop T muda de estado ("alterna") sempre que é acionado. Se a entrada T for baixa "0", o flip-flop mantém o valor de saída anterior. Como explicado para os flip-flops JK e D, os flip-flops T também podem ser classificados como flip-flops T gatilhável por borda ascendente (↑), borda descendente (↓), nível lógico 1 (⊓) e nível lógico 0 (⊔). Também é possível introduzir entradas assíncronas PRESET (ou PRE ou S) e CLEAR (CLR ou R) para flip-flops tipo T. Como exemplo, a figura 9.35 mostra o símbolo do flip-flop gatilhável por borda ascendente. A tabela verdade deste flip-flop é dada na tabela 9.19. Quando o sinal de entrada de relógio CLK está ligadop ("ON", 1), ou desligado ("OFF", 0), ou muda seu estado de ligado ("ON") para desligado ("OFF") (↓), nenhuma mudança de estado é emitida para a saída Q e ela mantém seu estado atual. Quando o estado do sinal de entrada de relógio CLK é alterado de desligado ("OFF") para ligado ("ON") (↑), se T = 0, nenhuma alteração de estado é emitida para a saída Q e ela mantém seu estado atual. Quando o estado do sinal de entrada do relógio CLK é alterado de desligado ("OFF") para ligado ("ON") (↑), se T = 1, então a saída Q é alternada.

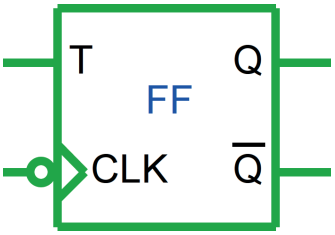


Figura 9.35 – O flip-flop T gatilhável por borda de subida.

Entradas		Saídas		Comentários
CLK	T	Q	Q'	
1	X	Q_0	Q_0'	Sem mudança.
0	X	Q_0	Q_0'	Sem mudança.
↓	X	Q_0	Q_0'	Sem mudança.
↑	0	Q_0	Q_0'	Sem mudança.
↑	1	Q_0'	Q_0	Troca.

Tabela 9.19 – A tabela verdade do flip-flop T gatilhável por borda de subida.

Na figura 9.36 e na tabela 9.20, o símbolo do flip-flop T gatilhável por borda de subida com entradas assíncronas ativas em baixo e sua tabela verdade são mostradas respectivamente. Neste flip-flop: quando S' = 0 e R' = 1 a saída é ativada (Q = 1). Quando S' = 1 e R' = 0, a saída é desativada (Q = 0). Quando S' = 0 e R' = 0, tanto a saída como seu complemento são ativados (Q = 1 e Q' = 1). Esta é uma operação inválida. Quando S' = 1 e R' = 1, ambas as entradas assíncronas não estão ativas. Neste caso, quando o estado do sinal de relógio CLK é alterado de ligado ("ON") para desligado ("OFF") (↴), se T = 0, nenhuma alteração de estado é emitida para a saída Q e ela mantém seu estado atual; se T = 1, então a saída Q é alternada.

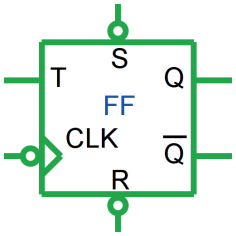


Figura 9.36 – O flip-flop T gatilhável por borda de subida com entradas assíncronas ativas em baixo.

Entradas				Saídas		Comentários
S'	R'	CLK	T	Q	Q'	
0	1	X	X	1	0	Ativação.
1	0	X	X	0	1	Desativação.
0	0	X	X	1	1	Inválido.
1	1	0	X	Q_0	Q_0'	Sem mudança.
1	1	1	X	Q_0	Q_0'	Sem mudança.
1	1	↥	X	Q_0	Q_0'	Sem mudança.
1	1	↴	0	Q_0	Q_0'	Sem mudança.
1	1	↴	1	Q_0'	Q_0	Troca.

Tabela 9.20 – A tabela verdade do flip-flop T gatilhável por borda de subida com entradas assíncronas ativas em baixo.

9.4.3. Exame do flip-flop T montado a partir de um flip-flop JK

Esquemas:

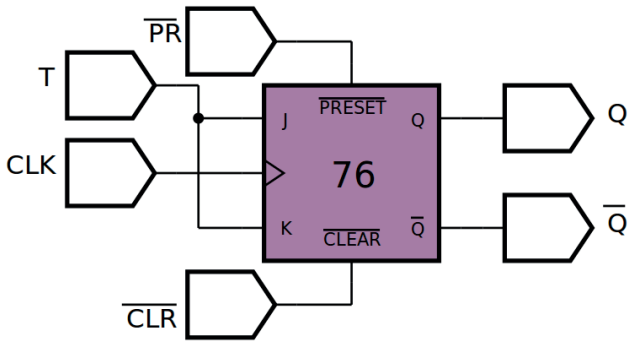


Figura 9.37 – Diagrama esquemático.

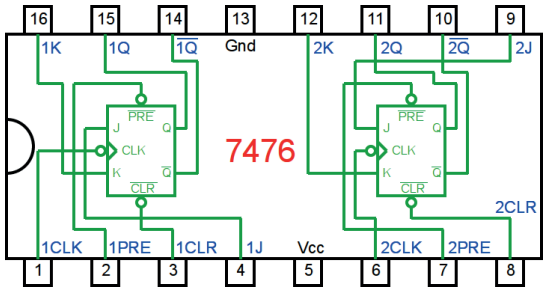


Figura 9.38 – Circuito integrado TTL.

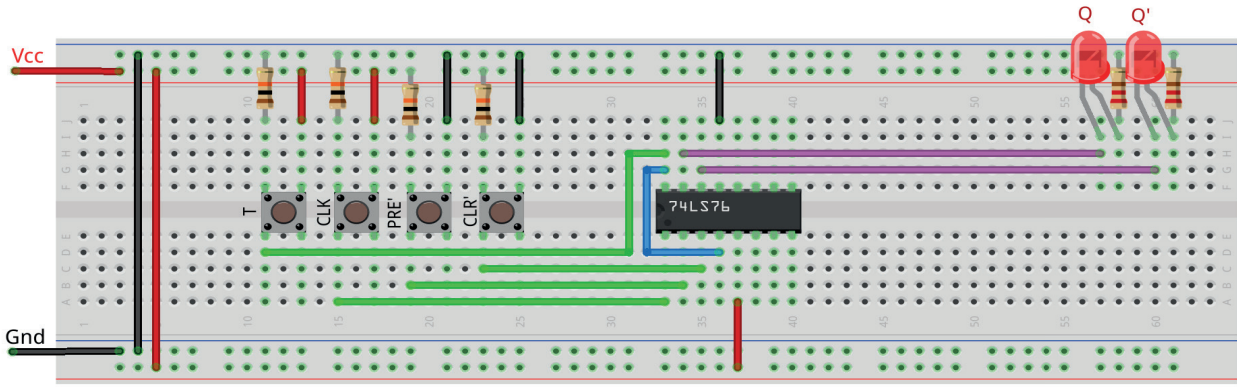


Figura 9.39 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 9.39.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes) entre os botões e o CI
   d) Conecte os fios de conexão (azuis) entre o CI.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 9.21, use os botões como segue:
   a) Para as variáveis PRE' e CLR' um "0" significa que o botão deve ser pressionado.
   b) Para a variável D um "1" significa que o botão deve ser pressionado.
   c) Para a variável CLK o sinal "↥↧" significa que o botão deve ser pressionado e solto. A entrada T não pode variar durante a execução do sinal de CLK.
3. Para o preenchimento das colunas de saída na tabela 9.21, considere:
   a) Se o LED estiver apagado então preencher com "0".
   b) Se o LED estiver aceso então preencher com "1".
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 9.37. Compare os resultados com a tabela 9.21.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 9.21.

Tabelas de dados:

Início		Entradas		Saídas		
PRE'	CLR'	T	CLK	Q	Q'	Comentários
0	0	X	X			
0	1	X	X			
1	0	X	X			
1	1	0	↥↧			
1	1	1	↥↧			

Tabela 9.21 – Flip-flop D com entradas assíncronas.

Programa para o Arduíno:

```
// teste do flip-flop T com flip-flop JK mestre escravo + clock com
portas NE

byte p1 = 1; byte p2 = 1; byte p3 = 1; byte p4 = 1;
byte p5 = 0; byte p6 = 0; byte p7 = 1; byte p8 = 1;
byte q0; byte q1; byte i; byte cl;
byte p3i; byte p4i; byte p7i; byte p8i;
byte ds[] = {0,0,1,1,1,1,1,1,1,1,1,1,1};
byte dr[] = {0,1,1,0,1,1,1,1,1,1,1,1,1};
byte dt[] = {0,0,0,0,0,1,0,1,0,1,1,1,1};
byte q = 13;

void ffjkmec_ne(byte pre, byte clr, byte j, byte k, byte clk){
  p1 = !((j&&clk)&&p8);
  p2 = !((k&&clk)&&p7);
  p3i = p3; p4i = p4;
  p3 = !((p4i&&p1)&&pre); if(p3i != p3) p3i = p3;
  p4 = !((p3i&&p2)&&clr); if(p4i != p4) p4i = p4;
  p3 = !((p4i&&p1)&&pre);
  p5 = !(p3&&!clk);
  p6 = !(p4&&!clk);
  p7i = p7; p8i = p8;
  p7 = !(p5&&p8i); if(p7i != p7) p7i = p7;
  p8 = !(p6&&p7i); if(p8i != p8) p8i = p8;
  p7 = !(p5&&p8i);
  q0 = p7;
  q1 = p8;
}

void mostra_ffjkmec_ne(){
  if (ds[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dr[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (dt[i]) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (!cl) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print("|");
  if (q0) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.println("|");
}

void setup(){
```

```
  Serial.begin(9600);
  Serial.println("Flip-flop T com flip-flop JK mestre-escravo + clock
  com portas NE");
  Serial.println("|PRE'|CLR'| T |CLK || Q | Q' |");
  Serial.println("--------------------------------");
  for (i=0;i<q;i++){
    for (cl = 0; cl <= 1; cl++){
      ffjkmec_ne(ds[i],dr[i],dt[i],dt[i],!cl);
      mostra_ffjkmec_ne();
    }
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

9.4.4. Divsor de frequência com flip-flop T montado a partir de um flip-flop JK.

Esquemas:

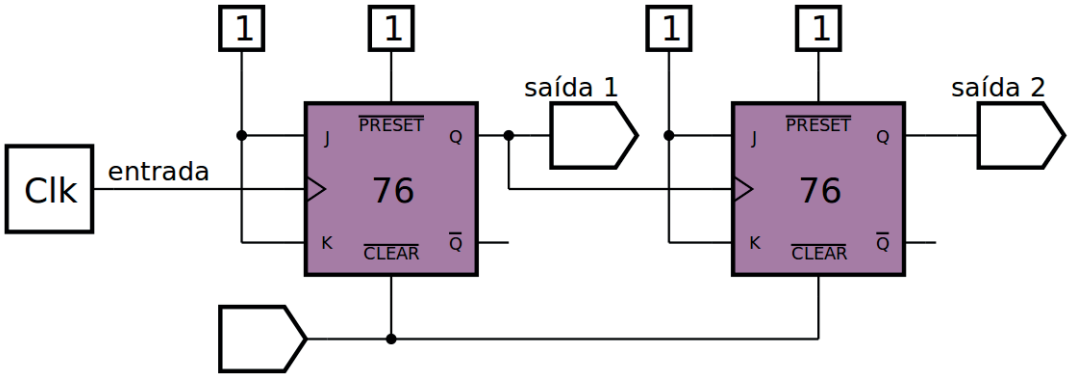


Figura 9.40 – Diagrama esquemático.

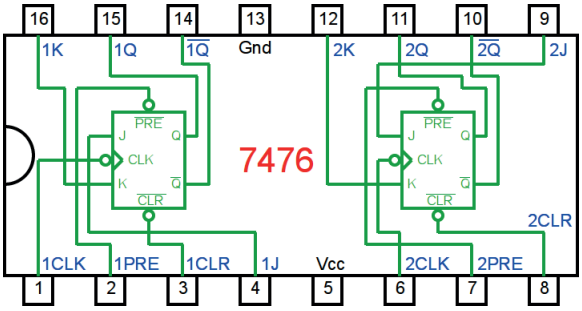


Figura 9.41 – Circuito integrado TTL.

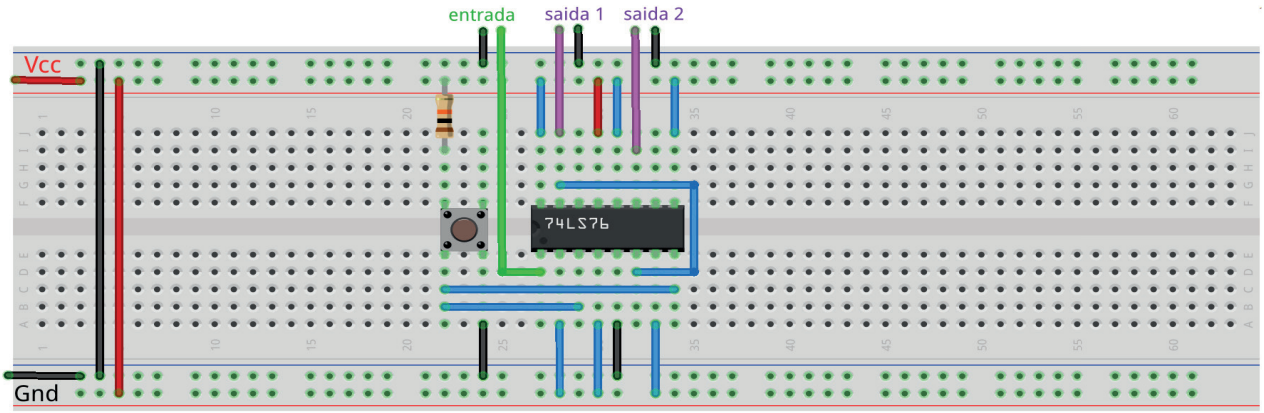


Figura 9.42 – Disposição dos componentes na placa de montagem.

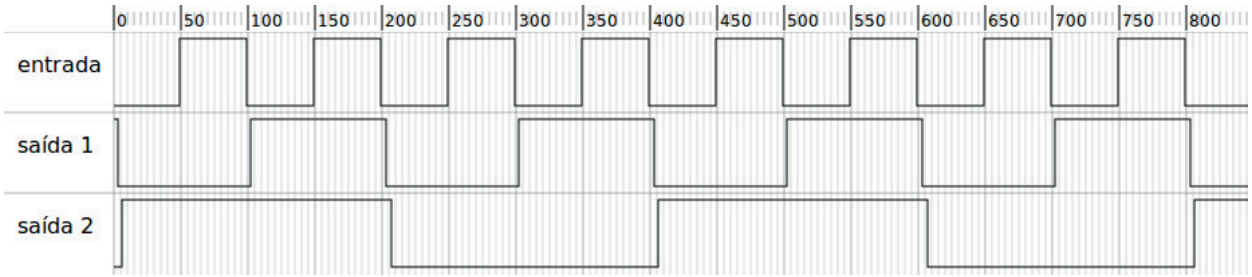


Figura 9.43 – Formas de onda esperadas.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 9.42.
   a) Coloque os componentes (resistores, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos).
   c) Conecte o fio de entrada (verde).
   d) Conecte os fios de conexão (azuis) entre os botões e o CI.
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Pressione uma vez o botão para iniciar os flip-flops.
3. Conecte um Gerador de Funções ao fio verde, com as seguintes configurações de saída: conector TTL, frequência de 500 Hertz, onda quadrada, valor alto de 5V e valor mínimo de 0V.
4. Conecte um osciloscópio com o canal X (ou A) na saída do gerador de funções (fio verde), conecte o canal Y (ou B) na saída do primeiro divisor de frequência, “saída 1” (fio roxo).
5. Tire uma foto da tela do osciloscópio.
6. Mude o canal Y (ou B) do osciloscópio do fio indicado como “saída 1” para o fio indicado como “saída 2”.
7. Tire uma foto da tela do osciloscópio.
8. Compare as fotos tiradas com o diagrama mostrado na figura 9.43.

Capítulo 10. Contadores

10.1. Objetivos

1. Investigar os contadores binários,
2. Observar seus princípios de operação,
3. Conhecer os CIs de contador binário 74LS93 e 4024.

10.2. Informação preliminar

Na lógica digital e computação, um contador é um dispositivo que armazena (e às vezes exibe) o número de vezes que um evento ou processo específico ocorreu, muitas vezes em relação a um sinal de relógio. Contadores são os circuitos lógicos que fazem um estado específico com os pulsos de relógio aplicados às suas entradas. Eles são amplamente utilizados na área de eletrônica digital. Algumas dessas áreas de aplicação são: Relógios Digitais, Contadores de Frequência, Decodificadores, Alarmes Digitais, Semáforos, etc.

A base dos contadores são circuitos lógicos e flip-flops. Geralmente, os contadores são obtidos com conexão em cascata de flip-flops em uma regra específica. Com cada pulso de relógio, o contador muda de estado. Um contador composto de *n* flip-flops sem realimentação pode ter 2^n estados diferentes, dependendo do número de pulsos de relógio. Por exemplo, se 4 flip-flops forem usados na estrutura do contador, haverá totalmente $2^4 = 16$ estados diferentes. Assim, o contador pode contar de 0 a 15. O número total de contagens ou estados estáveis que um contador pode indicar é chamado de MÓDULO (MOD). Por exemplo, o módulo de um contador de quatro estágios seria 16_{10}, já que é capaz de indicar 0000_2 a 1111_2. O termo módulo é usado para descrever a capacidade de contagem de contadores; ou seja, módulo-16 (MOD16) para um contador binário de quatro estágios, módulo-10 (MOD10) para um contador de década, módulo-8 (MOD8) para um contador binário de três estágios e assim por diante. Contadores podem ser contadores progressivos, cujo valor de contagem aumenta e contadores regressivos, cujo valor de contagem diminui. Um contador é geralmente considerado em conjunto com uma máquina de estados finitos (em inglês: FSM = "Finite State Machine"). A Fig. 10.1 mostra uma FSM para um contador binário de 3 bits. Nesta figura, cada estado (círculo) representa um valor de contagem diferente. O contador se moverá de um estado para o próximo com um sinal de relógio. Quando o valor da contagem é 111, o contador está em seu maior valor e, com o próximo sinal de relógio, passa do valor 111 para o valor inicial, a saber, 000. Os contadores podem ser divididos em dois grupos:

1. Contadores assíncronos (ondulação);
2. Contadores síncronos.

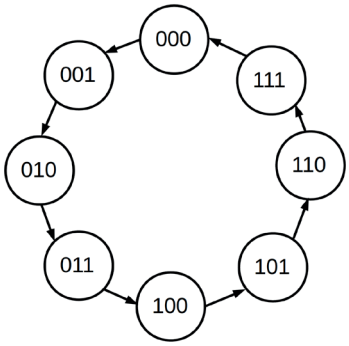


Figura 10.1 – Uma FSM para um contador binário de 3 bits.

10.3. Contadores assíncronos (ondulação)

10.3.1. Objetivos

1. Observar os princípios de operação dos contadores assíncronos,
2. Analisar os contadores progressivos e regressivos,
3. Definir os limites de contagem.

10.3.2. Informação preliminar

Os contadores assíncronos são contadores configurados de tal forma que todos os flip-flops não são acionados simultaneamente por um relógio comum. Como cada flip-flop no contador é acionado pelo flip-flop em série antes dele, esses contadores também são chamados de contadores de ondulação. Existem muitos tipos de contadores assíncronos. Um contador progressivo (UP) conta em uma sequência ascendente enquanto um contador regressivo (DOWN) conta em uma sequência descendente. Um contador também pode ter um comando para contagem progressiva e regressiva; tal contador é conhecido como um contador progressivo / regressivo (UP/DOWN).

Contadores assíncronos são limitados em velocidade, já que todos os flip-flops não são sincronizados pelo mesmo pulso de relógio. Portanto, o atraso de propagação em cada flip-flop frequentemente afeta a sequência de contagem em frequências de operação muito altas. Os flip-flops usados nos contadores assíncronos são geralmente flip-flops tipo “T” ou JK ou D que foram configurados como flip-flops T.

A decodificação do contador é uma técnica usada com contadores assíncronos para interromper ou reciclar a sequência de contagem em uma determinada contagem. Isso envolve um circuito conhecido como um decodificador de contagem que monitora a saída do contador para uma contagem específica e redefine o contador quando essa contagem é detectada.

Os contadores assíncronos podem ser construídos a partir de flip-flops discretos ou estão prontamente disponíveis na forma de CIs. As implementações de CI são projetadas para que os contadores possam ser configurados para uma ampla variedade de aplicações, desde a simples contagem até a divisão de frequência.

A saída de um flip-flop, em um contador assíncrono, é usada para acionar o próximo flip-flop. Em outras palavras, todos os flip-flops, exceto o primeiro, são acionados com a transição de estado dos flip-flops anteriores. No entanto, nos contadores síncronos, os pulsos de relógio de entrada são aplicados a todas as entradas CLK dos flip-flops ao mesmo tempo. O fato de um flip-flop mudar de estado depende dos estados de outros flip-flops. Todos os flip-flops funcionam no modo de alternância em um contador assíncrono.

Um contador binário assíncrono de 4 bits é mostrado na Fig. 10.2. Pode ser visto que este contador assíncrono é composto por quatro flip-flops JK. Todos os flip-flops JK estão trabalhando no modo de alternância. Quando as entradas J e K são iguais a “1”, com cada sinal de relógio a saída é alternada. O flip-flop que armazena o Bit Menos Significativo (em inglês: LSB = “Least Significant Bit”) recebe os pulsos do relógio. Os flip-flops são do tipo gatilho de borda descendente, então os flip-flops mudam seus estados com transição de alta para baixa (↴) do sinal em sua entrada CLK.

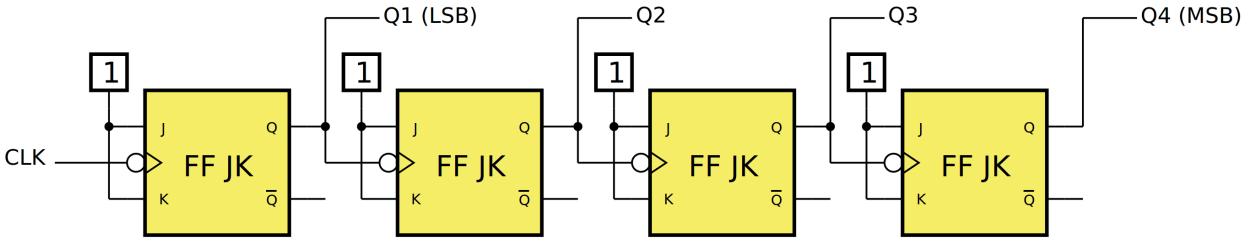


Figura 10.2 – Contador binário assíncrono de 4 bits.

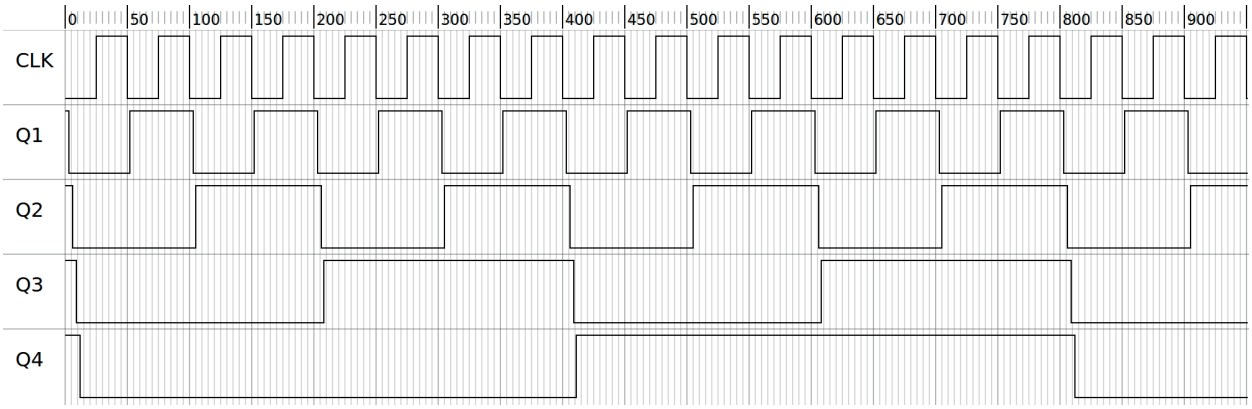


Figura 10.3 – Exemplo de diagrama de tempo para um contador binário assíncrono de 4 bits.

Um exemplo de diagrama de tempo para este contador é fornecido na figura 10.3. Observe as formas de onda na figura 10.3 para entender completamente o princípio de operação do contador binário assíncrono de 4 bits. Inicialmente, todas as saídas dos flip-flop são zero (0): Q1 = 0, Q2 = 0, Q3 = 0, Q4 = 0

Quando o pulso 0 (marca 50) do relógio CLK vai de “1” para “0”, o FF1 é acionado e sua saída torna-se “1”. As saídas Q dos outros flip-flops estão todas no nível “0” porque os pulsos de relógio adequados ainda não foram aplicados às suas entradas CLK. Então, após o pulso 0 do CLK, as saídas dos flip-flops são as seguintes: Q1 = 1, Q2 = 0, Q3 = 0, Q4 = 0

Quando o pulso 1 (marca 100) do relógio CLK vai de “1” para “0”, o FF1 é acionado novamente e sua saída Q1 vai de “1” para “0”. Como a saída Q1 está conectada à entrada CLK do FF2, essa entrada verá uma transição de alta para baixa, que aciona o FF2 e faz a saída Q2 ser “1”. (Saída Q1 é a entrada de disparo do FF2). Assim, com o pulso 1 de relógio, a saída do FF1 se torna “0” e a saída do FF2 se torna “1”. Finalmente, após o pulso 1 do CLK, as saídas dos flip-flops são as seguintes: Q1=0, Q2=1, Q3=0, Q4=0

Com o pulso 2 (marca 150) do relógio CLK, o FF1 é mais uma vez acionado e tem a saída Q1 como “1”. Os outros flip-flops conservam seus estados. Então, após o pulso 2 de CLK, as saídas dos flip-flops são as seguintes: Q1 = 1, Q2 = 1, Q3 = 0, Q4 = 0

Com o pulso 3 (marca 200) do relógio CLK, o FF1 é novamente acionado e sua saída Q1 vai de “1” para “0”. Portanto, esta transição negativa (borda descendente) irá acionar o FF2 e sua saída Q2 também irá de “1” para “0”. Com essa transição negativa, o FF3 também é acionado e a saída Q3 sobe para “1”. Então, após o pulso 3 do CLK, as saídas dos flip-flops são as seguintes: Q1 = 0, Q2 = 0, Q3 = 1, Q4 = 0

E o resto da operação de contagem é realizada de forma semelhante. Com o pulso 15 (marca 800) do relógio, todos os flip-flops são desativados (apagados): Q1 = 0, Q2 = 0, Q3 = 0, Q = 0

Quando as formas de onda da figura 10.3 são examinadas, pode ser visto que a frequência do sinal da saída do FF1 é 1/2, do FF2 é 1/4, do FF3 é 1/8 e do FF4 é 1/16 do pulso do relógio de entrada. Consequentemente, os contadores podem ser usados como divisores de frequência. Observe também o atraso de tempo na resposta dos flip-flop em relação à mudança de pulso de relógio. A saída Q1 está atrasada em relação ao pulso de relógio CLK. O mesmo acontece com a saída Q2 que está atrasada em relação ao pulso de relógio (Q1).

Nos contadores binários assíncronos regressivos, com cada pulso de relógio o valor da contagem é decrementado por 1. Por exemplo, se um contador binário regressivo de 4 bits começa a contar a partir de 15, então ele vai para 14, 13, 12, … ,1 ,0 com cada pulso de relógio e depois volta para 15. No circuito da figura 10.2, se conectarmos as entradas CLK dos flip-flops (exceto o primeiro) da saída Q' do flip-flop anterior e lermos o valor de contagem das saídas Q', o contador se torna um contador binário regressivo de 4 bits.

10.3.3. Contadores de módulos com reciclagem assíncrona

Os contadores podem ser projetados para ter um número de estados em sua sequência que seja menor que o máximo de 2^n. Esse tipo de sequência é chamado de *sequência truncada*. Para obter tal contador, um dos métodos que podem ser usados é a técnica de "reciclagem em relação ao módulo". Um módulo comum para contadores com sequências truncadas é dez (chamado MOD10). Contadores com dez estados em sua sequência são chamados de contadores de década. Um contador de uma década com uma sequência de contagem de zero (0000) a nove (1001) é um contador de décadas BCD porque a sua sequência de dez estados produz o código BCD. Esse tipo de contador é útil em aplicativos de exibição nos quais o BCD é necessário para conversão em uma leitura decimal. Para obter uma sequência truncada, é necessário forçar o contador a reciclar antes de passar por todos os seus possíveis estados. Por exemplo, o contador de décadas BCD deve reciclar de volta ao estado 0000 após o estado 1001. Um contador de uma década requer quatro flip-flops (três flip-flops são insuficientes porque $2^3 = 8$). Uma maneira de fazer um contador BCD a partir de um contador assíncrono de 4 bits é modificar sua sequência como mostrado na figura 10.4, que mostra um circuito contador ascendente assíncrono Mod10 (década ou BCD) obtido pelos flip-flops JK. Para forçar este contador a reciclar após a contagem de nove (1001) é decodificar a contagem em dez (1010) com uma porta NE e conectar a saída da porta NE às entradas clear (R) dos flip-flops, como mostrado na figura 10.4. Observe na figura 10.4 que apenas Q4 e Q2 estão conectados às entradas da porta NE. Esse arranjo é um exemplo de decodificação parcial, em que os dois estados únicos (Q4 = 1 e Q2 = 1) são suficientes para decodificar a contagem de dez porque nenhum dos outros estados (zero a nove) tem Q4 e Q2 em alto "1" ao mesmo tempo. Quando o contador entra em contagem de dez (1010), a saída da porta de decodificação fica em baixo "0" (ambas as entradas estão em alto "1") e desativa de forma assíncrona todos os flip-flops. Então a sequência de contagem começa no estado 0000 com o próximo sinal de relógio. O diagrama de tempos é mostrado na figura 10.5.

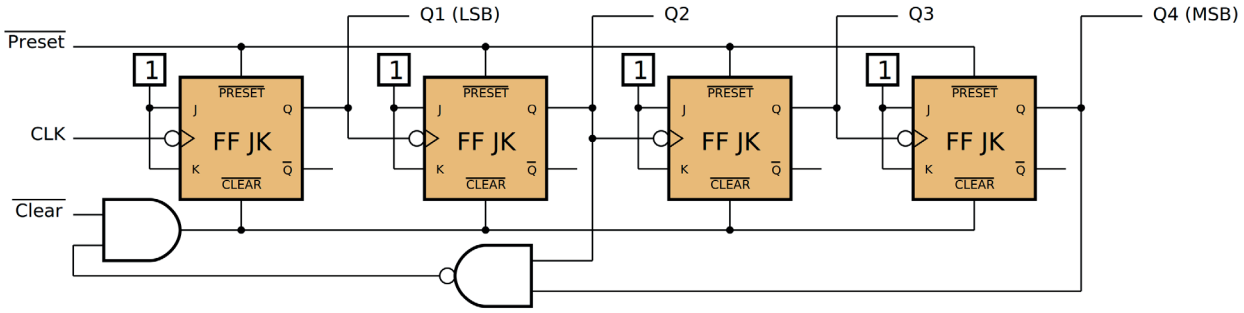


Figura 10.4 – Circuito contador ascendente assíncrono MOD10 (década ou BCD).

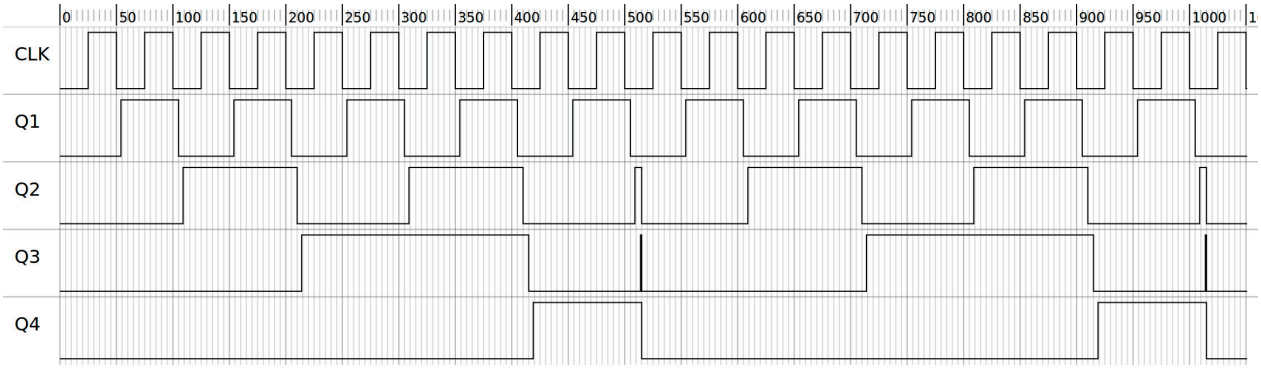


Figura 10.5 – Diagrama de tempo do contador ascendente assíncrono MOD10 (década ou BCD).

10.3.4. Exame do contador binário assíncrono de 4 bits crescente

Esquemas:

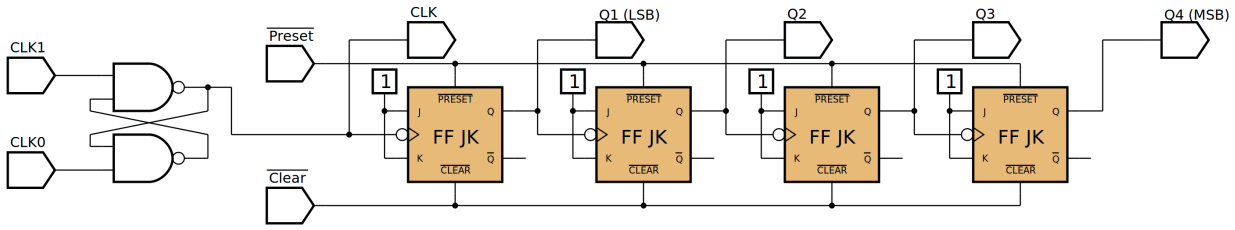


Figura 10.6 – Diagrama esquemático.

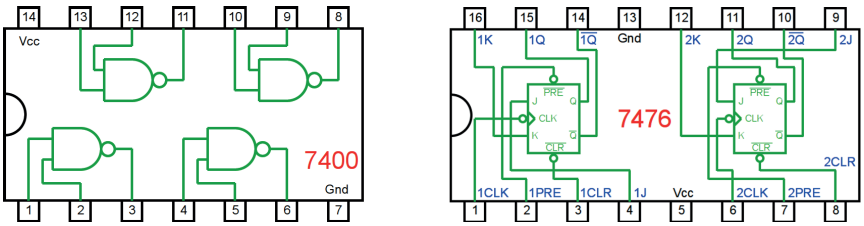


Figura 10.7 – Circuitos integrados TTL.

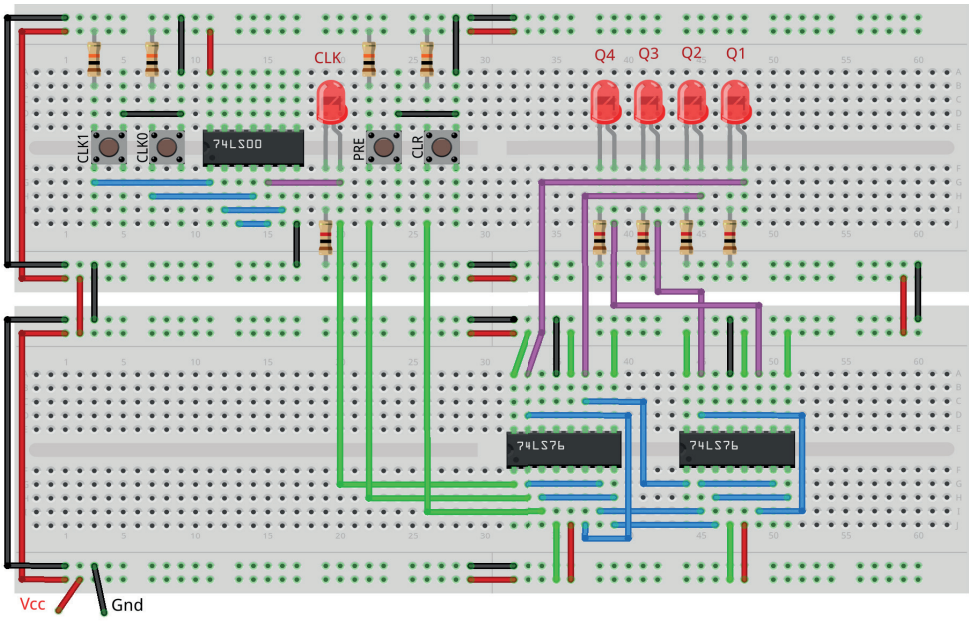


Figura 10.8 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 10.8.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 10.1, use os botões como segue:

a) Pressionando o botão PRE acende todos os leds.
b) Pressionando o botão CLR apaga todos os leds.
c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
e) Um pulso de relógio completo é conseguido pressionando o botão CLK1 e depois pressionando o botão CLK0.

3. Para o preenchimento das colunas de saída na tabela 10.1, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 10.6. Compare os resultados com a tabela 10.1.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 10.1.

Tabelas de dados:

	Saída				
Pulso	**Q4**	**Q3**	**Q2**	**Q1**	**Valor**
CLR					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

Tabela 10.1

Programa para o Arduíno:

```
// contador binário assincrono de 4 bits crescente

byte clka; byte clkd;
byte q1a; byte q2a; byte q3a; byte q4a;
byte q1d; byte q2d; byte q3d; byte q4d;
int intervalo;
byte i; byte q;
byte decimal;

void gera_clock(){
  delay(intervalo);
  clkd = !clka;
}

void contador4bits(){
  if (clka == 1 && clkd == 0) q1d = !q1a;
  if (q1a == 1 && q1d == 0) q2d = !q2a;
  if (q2a == 1 && q2d == 0) q3d = !q3a;
  if (q3a == 1 && q3d == 0) q4d = !q4a;
  q1a = q1d;
  q2a = q2d;
  q3a = q3d;
  q4a = q4d;
  decimal = 8*q4d + 4*q3d + 2*q2d + q1d;
}

void mostra_contador4bits(){
Serial.print("| "); if (i < 10) Serial.print("0");
  Serial.print(i); Serial.print(" ");
  if (q4d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q3d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q2d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (clkd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print(" | ");
  Serial.println(decimal);
}

void setup(){
  Serial.begin(9600);
```

```
  Serial.println("Contador binário assincrono de 4 bits crescente");
  Serial.println("| Pulso | Q4 | Q3 | Q2 | Q1 | Clk | Decimal");
  Serial.println("-------------------------------------------");

  q = 50;
  clka = 0;
  intervalo = 100;
  q1a = 0; q1d = 0;
  q2a = 0; q2d = 0;
  q3a = 0; q3d = 0;
  q4a = 0; q4d = 0;
  i = 0;
  mostra_contador4bits();

  for (i=1;i<q;i++){
    gera_clock();
    contador4bits();
    mostra_contador4bits();
    clka = clkd;
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

10.3.5. Exame do contador binário MOD10 assíncrono de 4 bits crescente

Esquemas:

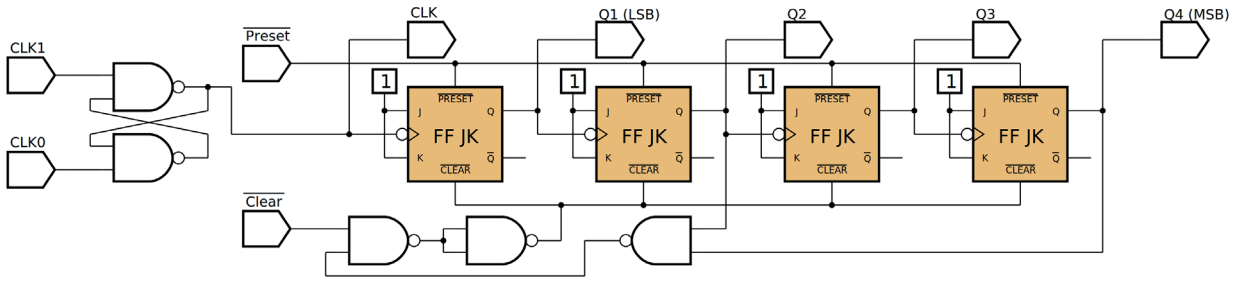


Figura 10.9 – Diagrama esquemático.

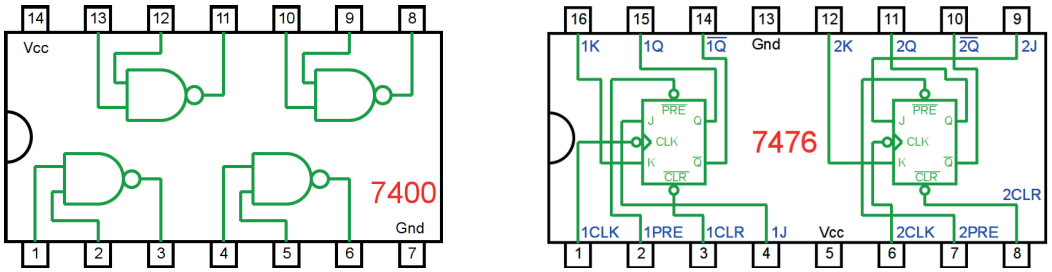


Figura 10.10 – Circuitos integrados TTL.

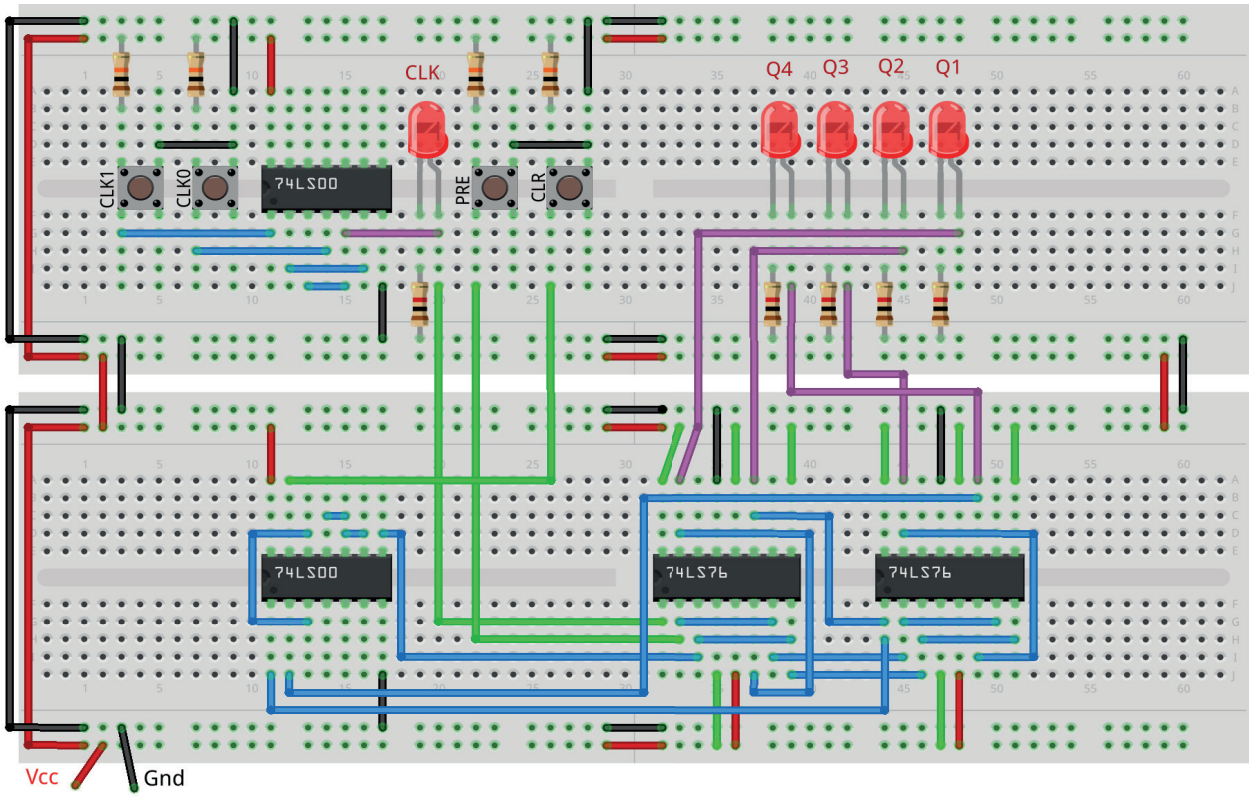


Figura 10.11 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 10.11.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 10.2, use os botões como segue:
   a) Pressionando o botão PRE acende todos os leds. (Observe quando retira o dedo do botão!)
   b) Pressionando o botão CLR apaga todos os leds.
   c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   e) Um pulso de relógio completo é conseguido pressionando o botão CLK1 e depois pressionando o botão CLK0.
3. Para o preenchimento das colunas de saída na tabela 10.2, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 10.9. Compare os resultados com a tabela 10.2.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 10.2.

Tabelas de dados:

	Saída				
Pulso	**Q4**	**Q3**	**Q2**	**Q1**	**Valor**
CLR					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

Tabela 10.2

Programa para o Arduíno:

```
// contador binário MOD10 assincrono de 4 bits crescente

byte clka; byte clkd;
byte q1a; byte q2a; byte q3a; byte q4a;
byte q1d; byte q2d; byte q3d; byte q4d;
int intervalo;
byte i; byte q;
byte decimal;

void gera_clock(){
  delay(intervalo);
  clkd = !clka;
}

void contador4bits(){
  if (clka == 1 && clkd == 0) q1d = !q1a;
  if (q1a == 1 && q1d == 0) q2d = !q2a;
  if (q2a == 1 && q2d == 0) q3d = !q3a;
  if (q3a == 1 && q3d == 0) q4d = !q4a;
  q1a = q1d;
  q2a = q2d;
  q3a = q3d;
  q4a = q4d;
    if(q4a == 1 && q2a == 1){
    q1a = 0; q1d = 0;
    q2a = 0; q2d = 0;
    q3a = 0; q3d = 0;
    q4a = 0; q4d = 0;
  }
  decimal = 8*q4d + 4*q3d + 2*q2d + q1d;
}

void mostra_contador4bits(){
  Serial.print("| "); if (i < 10) Serial.print("0");
  Serial.print(i); Serial.print(" ");
  if (q4d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q3d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q2d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1d) Serial.print("| 1 "); else Serial.print("| 0 ");
```

```
  if (clkd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print(" | ");
  Serial.println(decimal);
}

void setup(){
  Serial.begin(9600);
  Serial.println("Contador binário MOD10 assincrono de 4 bits
  crescente");
  Serial.println("| Pulso | Q4 | Q3 | Q2 | Q1 | Clk | Decimal");
  Serial.println("-------------------------------------------");
  q = 50;
  clka = 0;
  intervalo = 100;
  q1a = 0; q1d = 0;
  q2a = 0; q2d = 0;
  q3a = 0; q3d = 0;
  q4a = 0; q4d = 0;

  i = 0;
  mostra_contador4bits();
  for (i=1;i<q;i++){
    gera_clock();
    contador4bits();
    mostra_contador4bits();
    clka = clkd;
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

10.3.6. Exame do contador binário assíncrono de 4 bits decrescente

Esquemas:

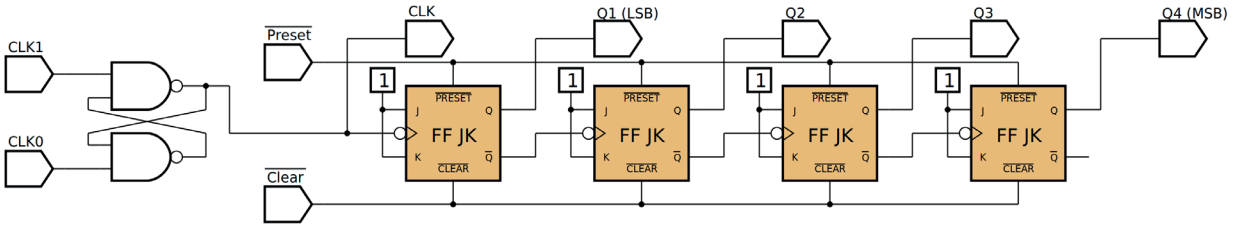


Figura 10.12 – Diagrama esquemático.

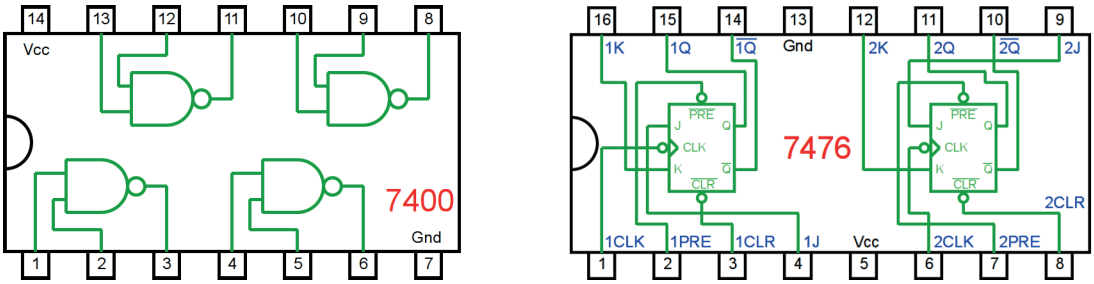


Figura 10.13 – Circuitos integrados TTL.

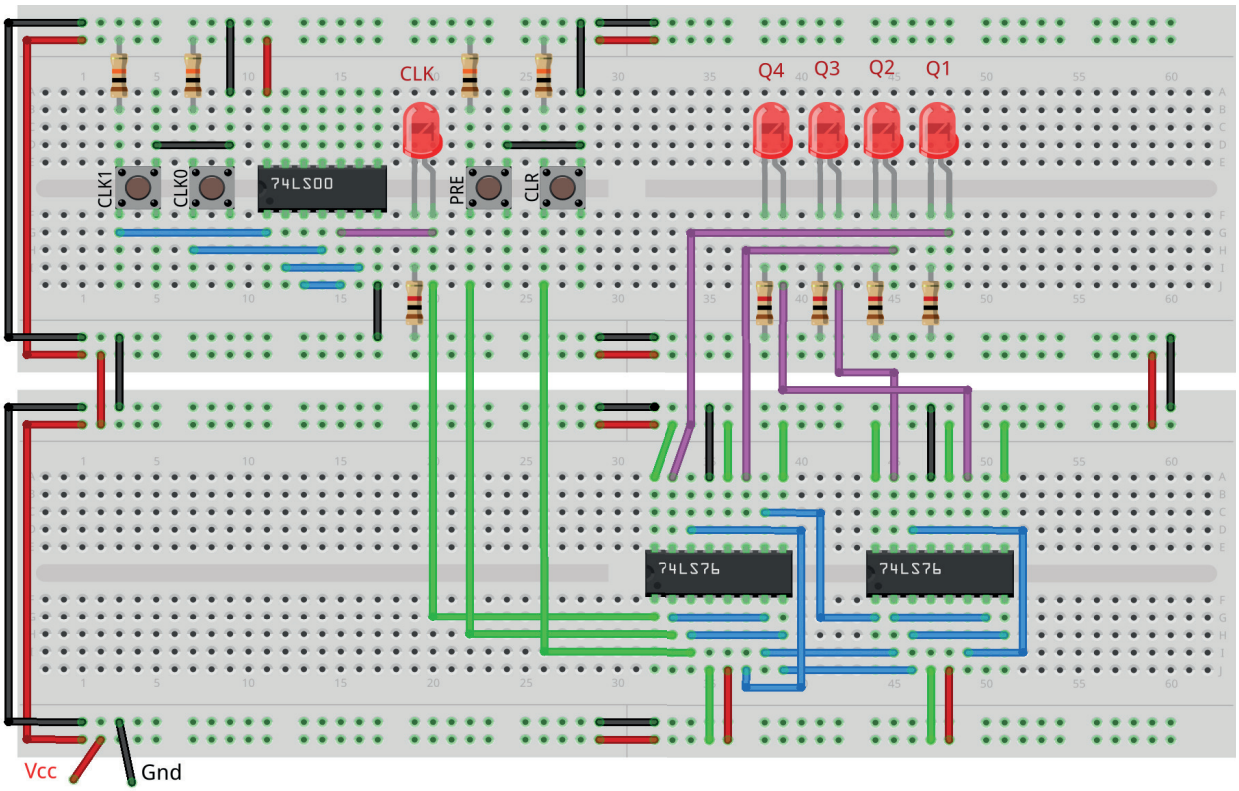


Figura 10.14 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 10.17.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 10.3, use os botões como segue:
   a) Pressionando o botão PRE acende todos os leds.
   b) Pressionando o botão CLR apaga todos os leds.
   c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   e) Um pulso de relógio completo é conseguido pressionando o botão CLK1 e depois pressionando o botão CLK0.
3. Para o preenchimento das colunas de saída na tabela 10.3, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 10.12. Compare os resultados com a tabela 10.3.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 10.3.

Tabelas de dados:

	Saída				
Pulso	**Q4**	**Q3**	**Q2**	**Q1**	**Valor**
PRE					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

Tabela 10.3

Programa para o Arduíno:

```
// contador binário assíncrono de 4 bits decrescente

byte clka; byte clkd;
byte q1a; byte q2a; byte q3a; byte q4a;
byte q1d; byte q2d; byte q3d; byte q4d;
byte q1ia; byte q2ia; byte q3ia; byte q4ia;
byte q1id; byte q2id; byte q3id; byte q4id;
int intervalo;
byte i; byte q;
byte decimal;

void gera_clock(){
  delay(intervalo);
  clkd = !clka;
}

void contador4bits(){
  if (clka == 1 && clkd == 0) {q1d = !q1a; q1id = q1a;}
  if (q1ia == 1 && q1id == 0) {q2d = !q2a; q2id = q2a;}
  if (q2ia == 1 && q2id == 0) {q3d = !q3a; q3id = q3a;}
  if (q3ia == 1 && q3id == 0) {q4d = !q4a; q4id = q4a;}
  q1a = q1d; q2a = q2d; q3a = q3d; q4a = q4d;
  q1ia = q1id; q2ia = q2id; q3ia = q3id; q4ia = q4id;
  decimal = 8*q4d + 4*q3d + 2*q2d + q1d;
}

void mostra_contador4bits(){
  Serial.print("| "); if (i < 10) Serial.print("0");
  Serial.print(i); Serial.print(" ");
  if (q4d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q3d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q2d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (clkd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print(" | ");
  Serial.println(decimal);
}
```

```
void setup(){
  Serial.begin(9600);
  Serial.println("Contador binário assincrono de 4 bits decrescente");
  Serial.println("| Pulso | Q4 | Q3 | Q2 | Q1 | Clk | Decimal");
  Serial.println("--------------------------------------------");

  q = 50;
  clka = 0;
  intervalo = 100;
  q1a = 0; q1d = 0;
  q2a = 0; q2d = 0;
  q3a = 0; q3d = 0;
  q4a = 0; q4d = 0;
  q1ia = 1; q1id = 1;
  q2ia = 1; q2id = 1;
  q3ia = 1; q3id = 1;
  q4ia = 1; q4id = 1;

  i = 0;
  mostra_contador4bits();
  for (i=1;i<q;i++){
    gera_clock();
    contador4bits();
    mostra_contador4bits();
    clka = clkd;
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

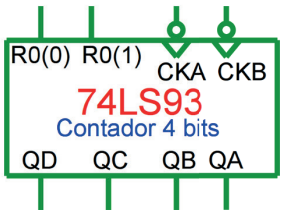
10.3.7. CI TTL 74LS93 – Contador binário assíncrono de 4 bits

O 74LS93 é um exemplo de um CI TTL específico de contador assíncrono. A figura 10.15 mostra o símbolo lógico, pinagem, o diagrama esquemático, a tabela verdade de reconfiguração / contagem e a sequência de contagem do CI 74LS93 contador crescente assíncrono. Como mostra o diagrama lógico na figura 10.15, este dispositivo consiste, na verdade, em um único flip-flop e um contador assíncrono de 3 bits. Esse arranjo é para flexibilidade. Ele pode ser usado como um dispositivo de divisão por 2 se apenas o flip-flop único for usado, ou pode ser usado como um contador MOD8 se apenas um contador de 3 bits for usado. Este dispositivo também fornece entradas de reinicialização controladas, R0(0) e R0(1). Quando essas duas entradas estão em ALTO, o contador é redefinido para o estado 0000. O 74LS93 pode ser usado como um contador MOD16 de 4 bits (contagens de 0 a 15). Ele também pode ser configurado como um contador de décadas (contagens de 0 a 9) com reciclagem assíncrona usando as entradas de reinicialização R0(0) e R0(1).

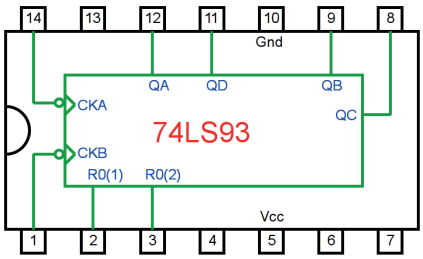
Reset		Saídas			
R0(0)	R0(1)	QD	QC	QB	QA
1	1	0	0	0	0
0	X	Contagem			
X	0	Contagem			

Contagem	Saídas			
	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Tabela verdade



Símbolo lógico



Pinagem

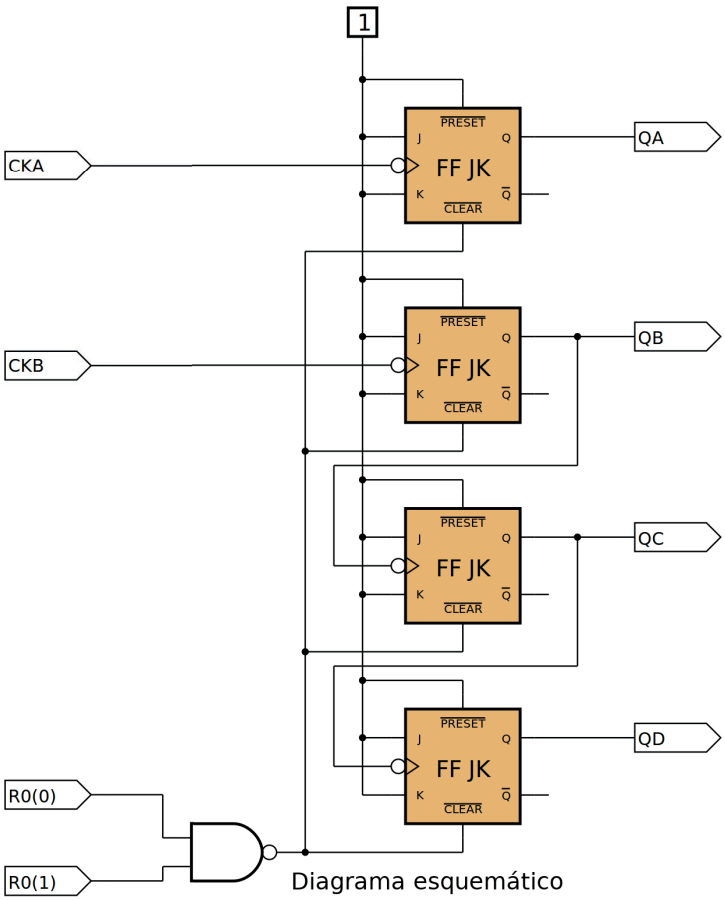


Diagrama esquemático

Figura 10.15 – O símbolo lógico, pinagem, diagrama esquemático, tabela verdade de reset/ contagem e a sequência de contagem do CI 74LS93 contador crescente assíncrono.

10.3.8. Exame do CI TTL 74LS93 contador binário assíncrono de 4 bits

Esquemas:

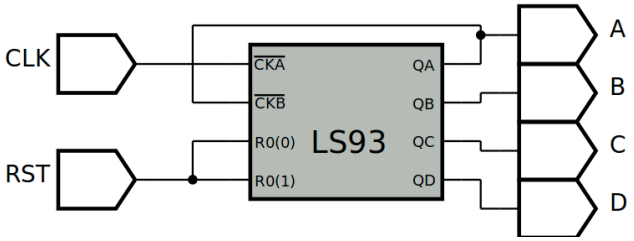


Figura 10.16 – Diagrama esquemático.

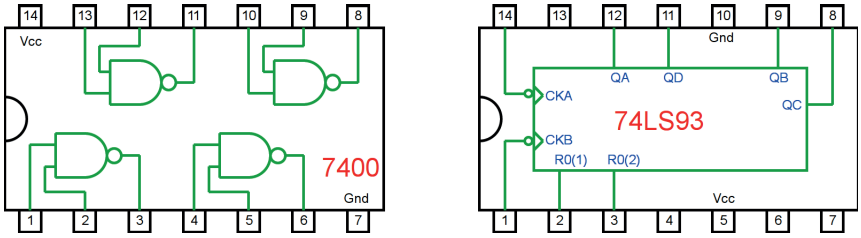


Figura 10.17 – Circuitos integrados TTL.

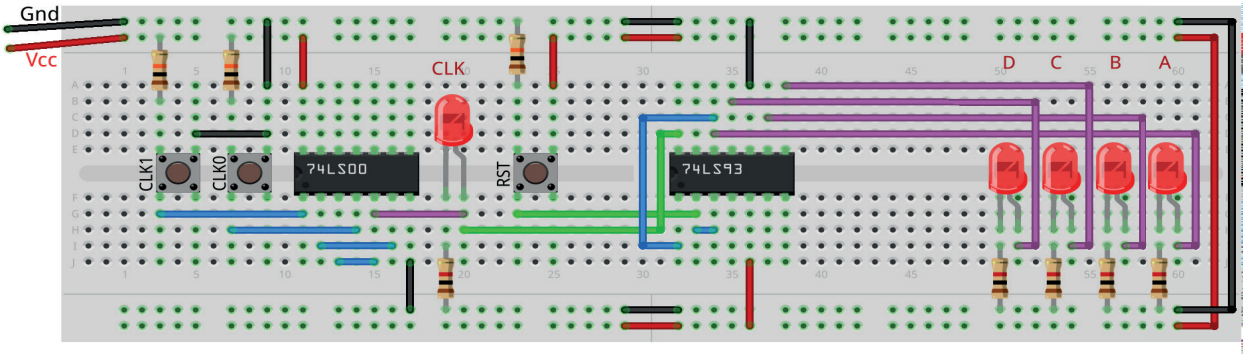


Figura 10.18 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 10.18.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 10.4, use os botões como segue:
   a) Pressionando o botão RST apaga todos os leds.

b) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
c) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
d) Um pulso de relógio completo é conseguido pressionando o botão CLK1 e depois pressionando o botão CLK0.

3. Para o preenchimento das colunas de saída na tabela 10.4, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 10.16. Compare os resultados com a tabela 10.4.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 10.4.

Tabelas de dados:

	Saída				
Pulso	**D**	**C**	**B**	**A**	**Valor**
RST					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

Tabela 10.4

Programa para o Arduíno:

Este programa necessita de 5 leds e 1 botão conectados numa placa de montagem com resistores, conforme mostrado na figura 10.19. Os resistores dos leds são de 1K e o resistor do botão é de 10K.

Ao pressionar o botão, um pulso de relógio é mostrado no led “clock” e o contador executa a contagem. Os leds Q1, Q2, Q3 e Q4 indicam o estado de saída do contador.

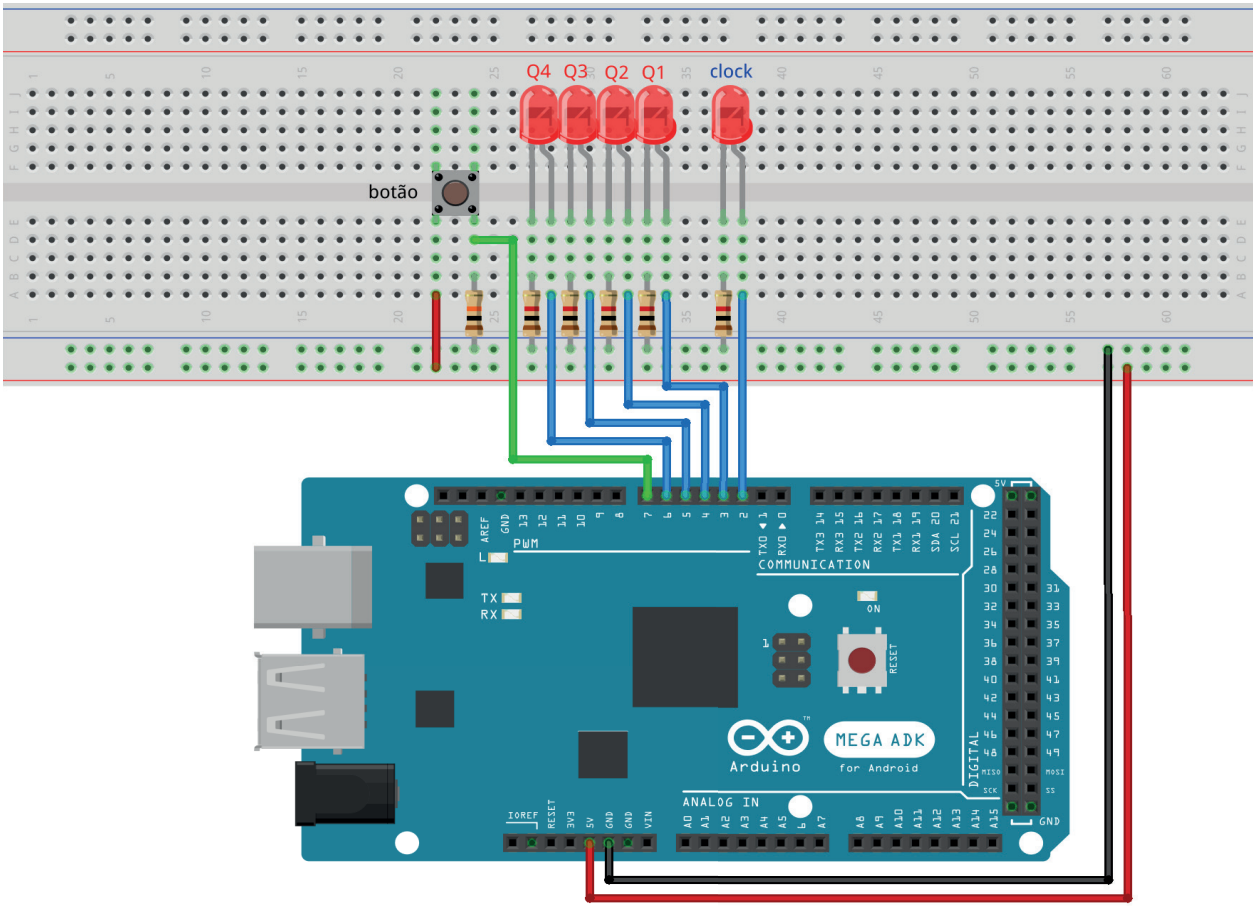


Figura 10.19 – Disposição dos componentes na placa de montagem e conexão com Arduíno.

```
// contador binário assíncrono de 4 bits crescente com leds

#define led_clk 2
#define led_q1 3
#define led_q2 4
#define led_q3 5
#define led_q4 6
#define botao 7

byte clka; byte clkd;
byte q1a; byte q2a; byte q3a; byte q4a;
byte q1d; byte q2d; byte q3d; byte q4d;
int intervalo;
byte i; byte q;
byte decimal;
byte vba; byte vbd;

void gera_clock(){
  delay(intervalo);
  clkd = !clka;
}

void contador4bits(){
  if (clka == 1 && clkd == 0) q1d = !q1a;
  if (q1a == 1 && q1d == 0) q2d = !q2a;
  if (q2a == 1 && q2d == 0) q3d = !q3a;
  if (q3a == 1 && q3d == 0) q4d = !q4a;
  q1a = q1d;
  q2a = q2d;
  q3a = q3d;
  q4a = q4d;
  decimal = 8*q4d + 4*q3d + 2*q2d + q1d;
}

void mostra_contador4bits(){
  Serial.print("| "); if (i < 10) Serial.print("0");
  Serial.print(i); Serial.print(" ");
  if (q4d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q3d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q2d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (clkd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print(" | ");
  Serial.println(decimal);
}
```

```
void leds_contador4bits(){
  digitalWrite(led_clk, clkd);
  digitalWrite(led_q1, q1d);
  digitalWrite(led_q2, q2d);
  digitalWrite(led_q3, q3d);
  digitalWrite(led_q4, q4d);
}

  void setup(){
  Serial.begin(9600);
  Serial.println("Contador binário assincrono de 4 bits crescente");
  Serial.println("| Pulso | Q4 | Q3 | Q2 | Q1 | Clk | Decimal");
  Serial.println("--------------------------------------------");

  pinMode(botao, INPUT);
  pinMode(led_clk, OUTPUT);
  pinMode(led_q1, OUTPUT);
  pinMode(led_q2, OUTPUT);
  pinMode(led_q3, OUTPUT);
  pinMode(led_q4, OUTPUT);

  vba = 0;
  clka = 0;

  q1a = 0; q1d = 0;
  q2a = 0; q2d = 0;
  q3a = 0; q3d = 0;
  q4a = 0; q4d = 0;
} // fim do 'setup'

void loop(){
  vbd = digitalRead(botao);
  if(vbd != vba){
    delay(100);
    clkd = vbd;
    contador4bits();
    mostra_contador4bits();
    leds_contador4bits();
    clka = clkd;
    vba = vbd;
  }
} // fim do 'loop'
```

10.3.9. Exame do CI TTL 74LS93 usado como contador BCD (MOD10)

Esquemas:

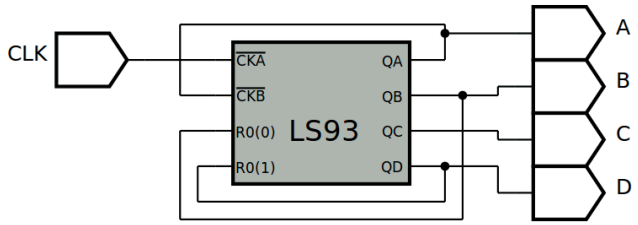


Figura 10.20 – Diagrama esquemático.

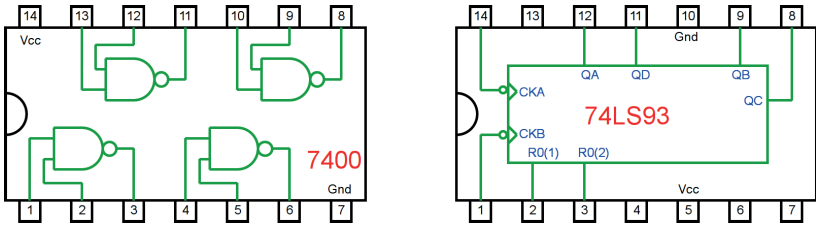


Figura 10.21 – Circuitos integrados TTL.

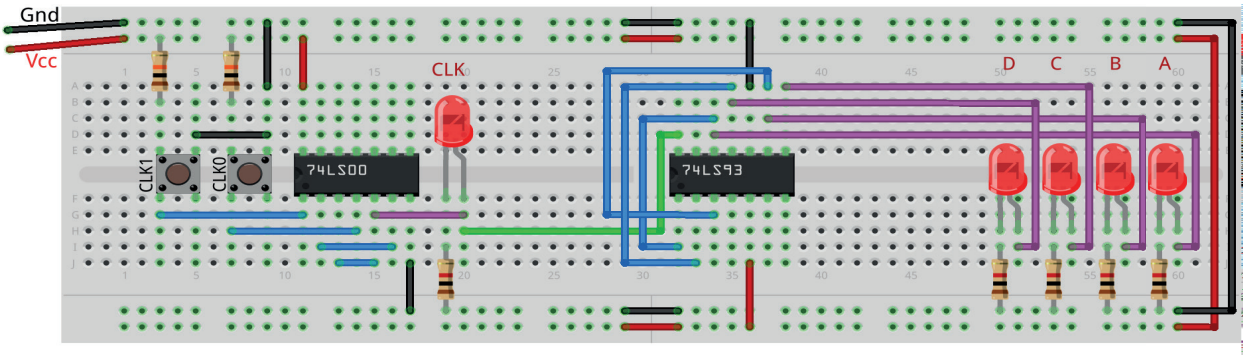


Figura 10.22 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 10.22.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 10.5, use os botões como segue:

a) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
b) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
c) Um pulso de relógio completo é conseguido pressionando o botão CLK1 e depois pressionando o botão CLK0.

3. Para o preenchimento das colunas de saída na tabela 10.5, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Altere as configurações para conseguir um contador MOD8 e um contador MOD12.
5. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 10.20. Compare os resultados com a tabela 10.5.
6. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 10.5.

Tabelas de dados:

	Saída				
Pulso	**D**	**C**	**B**	**A**	**Valor**
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					

Tabela 10.5

Programa para o Arduíno:

Este programa necessita da mesma montagem mostrada na figura 10.19.
Este programa está ajustado para MOD 10.
Para fazer o programa funcionar em MOD 8, remova as barras de comentário (//) no início da linha:

```
//if(q4a == 1){ // MOD 8
```

para:

```
if(q4a == 1){ // MOD 8
```

Para fazer o programa funcionar em MOD 12, remova as barras de comentário (//) no início da linha:

```
//if(q4a == 1 && q3a == 1){ // MOD 12
```

para:

```
if(q4a == 1 && q3a == 1){ // MOD 12
```

Depois de mudar para MOD8 ou MOD 12, coloque as barras de comentário (//) no início da linha para MOD10:

```
if(q4a == 1 && q2a == 1){ // MOD 10
```

para

```
//if(q4a == 1 && q2a == 1){ // MOD 10
```

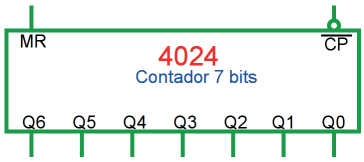
```
// contador binário assincrono MOD10 de 4 bits crescente com leds +
MOD8 e MOD12

#define led_clk 2
#define led_q1 3
#define led_q2 4
#define led_q3 5
#define led_q4 6
#define botao 7
byte clka; byte clkd;
byte q1a; byte q2a; byte q3a; byte q4a;
byte q1d; byte q2d; byte q3d; byte q4d;
```

```
int intervalo;
byte i; byte q;
byte decimal;
byte vba; byte vbd;
void gera_clock(){
  delay(intervalo);
  clkd = !clka;
}

void contador4bits(){
  if (clka == 1 && clkd == 0) q1d = !q1a;
  if (q1a == 1 && q1d == 0) q2d = !q2a;
  if (q2a == 1 && q2d == 0) q3d = !q3a;
  if (q3a == 1 && q3d == 0) q4d = !q4a;
  q1a = q1d;
  q2a = q2d;
  q3a = q3d;
  q4a = q4d;

//if(q4a == 1){ // MOD 8
//if(q4a == 1 && q3a == 1){ // MOD 12
  if(q4a == 1 && q2a == 1){ // MOD 10
    q1a = 0; q1d = 0;
    q2a = 0; q2d = 0;
    q3a = 0; q3d = 0;
    q4a = 0; q4d = 0;
  }
  decimal = 8*q4d + 4*q3d + 2*q2d + q1d;
}

void mostra_contador4bits(){
  Serial.print("| "); if (i < 10) Serial.print("0");
  Serial.print(i); Serial.print(" ");
  if (q4d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q3d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q2d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (clkd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print(" | ");
  Serial.println(decimal);
}
```

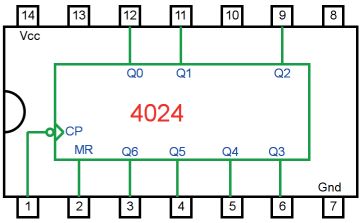
```
void leds_contador4bits(){
  digitalWrite(led_clk, clkd);
  digitalWrite(led_q1, q1d);
  digitalWrite(led_q2, q2d);
  digitalWrite(led_q3, q3d);
  digitalWrite(led_q4, q4d);
}

void setup(){
  Serial.begin(9600);
  Serial.println("Contador binário assincrono de 4 bits crescente");
  Serial.println("| Pulso | Q4 | Q3 | Q2 | Q1 | Clk | Decimal");
  Serial.println("---------------------------------------------");

  pinMode(botao, INPUT);
  pinMode(led_clk, OUTPUT);
  pinMode(led_q1, OUTPUT);
  pinMode(led_q2, OUTPUT);
  pinMode(led_q3, OUTPUT);
  pinMode(led_q4, OUTPUT);

  vba = 0;
  clka = 0;

  q1a = 0; q1d = 0;
  q2a = 0; q2d = 0;
  q3a = 0; q3d = 0;
  q4a = 0; q4d = 0;
} // fim do 'setup'

void loop(){
  vbd = digitalRead(botao);
  if(vbd != vba){
    delay(100);
    clkd = vbd;
    contador4bits();
    mostra_contador4bits();
    leds_contador4bits();
    clka = clkd;
    vba = vbd;
  }
} // fim do 'loop'
```

10.3.10. Circuito Integrado CMOS 4024 – Contador assíncrono de 7 bits crescente

O CI CMOS 4024 é um contador binário assíncrono de 7 bits. A figura 10.23 mostra o símbolo lógico, a pinagem e a tabela de funções do CI 4024 contador binário assíncrono de 7 bits crescente. O 4024 possui uma entrada de reset mestre (MR) assíncrona ativa em alto, uma entrada de relógio (CP') e sete saídas paralelas totalmente amplificadas (Q6, Q5, ..., Q0). O contador avança (conta de 0000000 até 1111111) na transição do pulso de relógio de alto "1" para baixo "0" (↴). Um sinal alto "1" na entrada MR limpa todos os estágios do contador e força todas as saídas para baixo "0", independentemente do pulso de relógio CP'. Cada estágio de contagem é um flip-flop estático. Na figura 10.24 pode ser vista a disposição interna dos flip-flops.



Símbolo lógico



Pinagem

Entradas		Saídas
MR	$\overline{CP}$	Qn
1	x	0
0	↴	Não altera
0	↴	Contagem

Tabela de funções

Figura 10.23 – O símbolo lógico, a pinagem e a tabela de funções do CI 4024 contador binário assíncrono de 7 bits crescente.

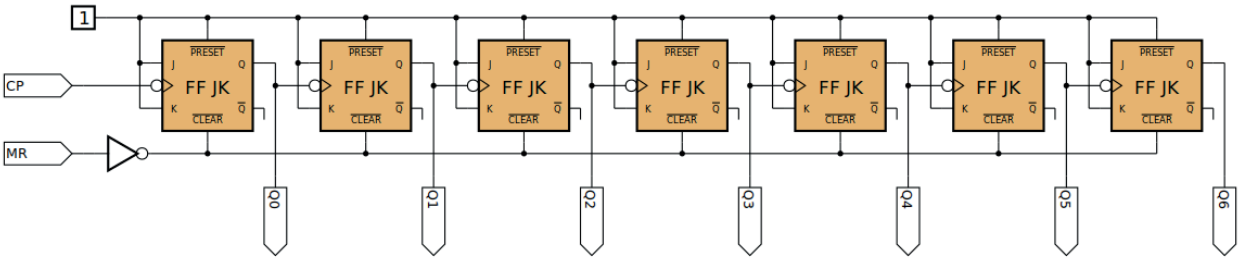


Figura 10.24 – Disposição interna dos flip-flops do CI 4024.

10.3.11. Exame do CI CMOS 4024 contador assíncrono 7 bits crescente

Esquemas:

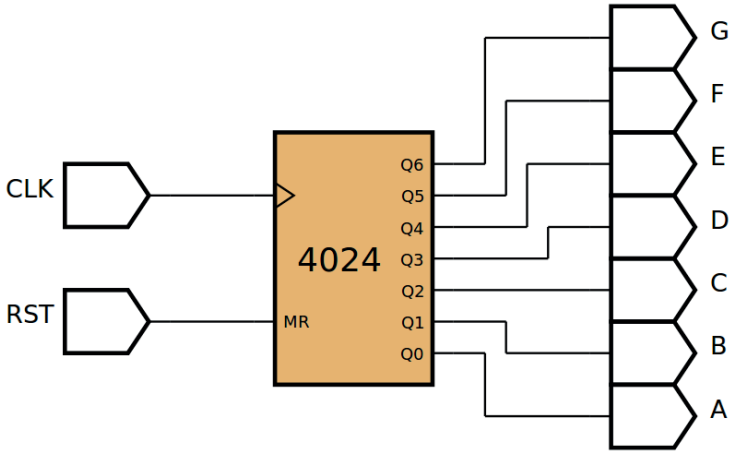


Figura 10.25 – Diagrama esquemático.

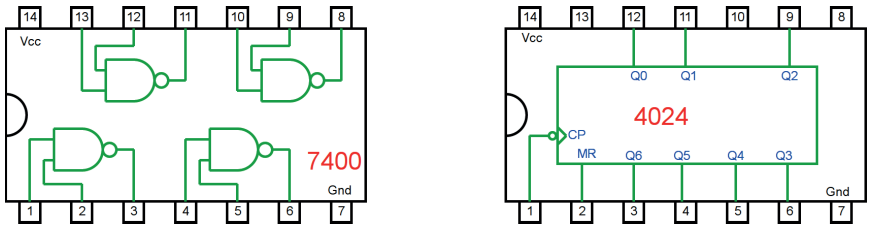


Figura 10.26 – Circuitos integrados TTL e CMOS.

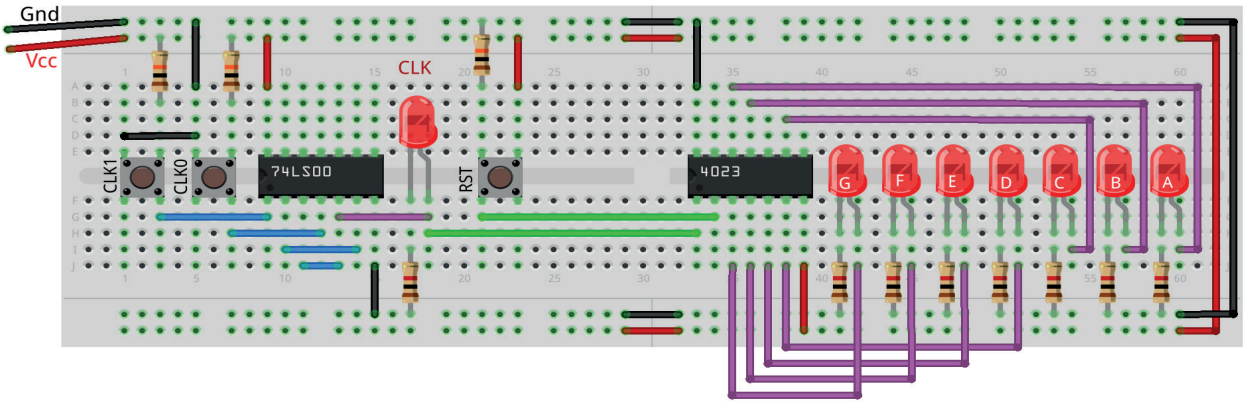


Figura 10.27 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 10.27.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 10.6, use os botões como segue:
   a) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   b) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   c) Um pulso de relógio completo é conseguido pressionando o botão CLK1 e depois pressionando o botão CLK0.
3. Para o preenchimento das colunas de saída na tabela 10.6, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 10.25. Compare os resultados com a tabela 10.6.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 10.6.

Tabelas de dados:

	Saída							
Pulso	G	F	E	D	C	B	A	Valor
RST								
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								

.

	Saída							
Pulso	G	F	E	D	C	B	A	Valor
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								

Tabela 10.6a

	Saída							
Pulso	G	F	E	D	C	B	A	Valor
32								
33								
34								
35								
36								
37								
38								
39								
40								
41								
42								
43								
44								
45								
46								
47								

.

	Saída							
Pulso	G	F	E	D	C	B	A	Valor
48								
49								
50								
51								
52								
53								
54								
55								
56								
57								
58								
59								
60								
61								
62								
63								

Tabela 10.6b

	Saída							
Pulso	G	F	E	D	C	B	A	Valor
64								
65								
66								
67								
68								
69								
70								
71								
72								
73								
74								
75								
76								
77								
78								
79								

.

	Saída							
Pulso	G	F	E	D	C	B	A	Valor
80								
81								
82								
83								
84								
85								
86								
87								
88								
89								
90								
91								
92								
93								
94								
95								

Tabela 10.6c

	Saída							
Pulso	G	F	E	D	C	B	A	Valor
96								
97								
98								
99								
100								
101								
102								
103								
104								
105								
106								
107								
108								
109								
110								
111								

.

	Saída							
Pulso	G	F	E	D	C	B	A	Valor
112								
113								
114								
115								
116								
117								
118								
119								
120								
121								
122								
123								
124								
125								
126								
127								

Tabela 10.6d

Programa para o Arduíno:

```
// contador binário assíncrono de 7 bits crescente

byte clka; byte clkd;
byte q1a; byte q2a; byte q3a; byte q4a; byte q5a; byte q6a; byte q7a;
byte q1d; byte q2d; byte q3d; byte q4d; byte q5d; byte q6d; byte q7d;

int intervalo;
int i; int q;
byte decimal;

void gera_clock(){
  delay(intervalo);
  clkd = !clka;
}

void contador7bits(){
  if (clka == 1 && clkd == 0) q1d = !q1a;
  if (q1a == 1 && q1d == 0) q2d = !q2a;
  if (q2a == 1 && q2d == 0) q3d = !q3a;
  if (q3a == 1 && q3d == 0) q4d = !q4a;
  if (q4a == 1 && q4d == 0) q5d = !q5a;
  if (q5a == 1 && q5d == 0) q6d = !q6a;
  if (q6a == 1 && q6d == 0) q7d = !q7a;
  q1a = q1d;
  q2a = q2d;
  q3a = q3d;
  q4a = q4d;
  q5a = q5d;
  q6a = q6d;
  q7a = q7d;
  decimal = 64*q7d + 32*q6d + 16*q5d + 8*q4d + 4*q3d + 2*q2d + q1d;
}

void mostra_contador7bits(){
  Serial.print("| ");
  if (i < 100) Serial.print("0");if (i < 10) Serial.print("0");
  Serial.print(i); Serial.print(" ");
  if (q7d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q6d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q5d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q4d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q3d) Serial.print("| 1 "); else Serial.print("| 0 ");
```

```
  if (q2d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (clkd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print(" | ");
  Serial.println(decimal);
}
void setup(){
  Serial.begin(9600);
  Serial.println("Contador binário assincrono de 7 bits crescente");
  Serial.println("| Pulso | Q7 | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Clk |
  Decimal");
  Serial.println("-----------------------------------------------------
  -------");

  q = 300;
  clka = 0;
  intervalo = 100;
  q1a = 0; q1d = 0; q2a = 0; q2d = 0;
  q3a = 0; q3d = 0; q4a = 0; q4d = 0;
  i = 0;

  mostra_contador7bits();

  for (i=1;i<q;i++){
    gera_clock();
    contador7bits();
    mostra_contador7bits();
    clka = clkd;
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

10.4. Contadores síncronos

10.4.1. Objetivos

1. Observar os princípios de operação dos contadores síncronos,
2. Estudar a concepção e construção de contadores síncronos,
3. Estudar diferentes exemplos de contador síncrono.

10.4.2. Informação preliminar

Contadores síncronos são contadores configurados de tal forma que todos os flip-flops são acionados simultaneamente por um relógio comum. Todos os flip-flops são, portanto, "sincronizados" pelo mesmo relógio. Como os contadores assíncronos, existem muitos tipos de contadores síncronos. Eles podem ser projetados para fornecer as mesmas funções que contadores assíncronos. Portanto, muitos aplicativos que exigem contadores podem ter contadores assíncronos ou contadores síncronos.

Diferentemente dos contadores assíncronos, os contadores síncronos não são limitados em velocidade, já que todos os flip-flops são sincronizados pelo mesmo pulso de relógio. Como o relógio de cada flip-flop não é afetado por atrasos de propagação, os contadores síncronos não são suscetíveis a efeitos adversos de operações de alta frequência. Portanto, os contadores síncronos são frequentemente usados em aplicativos que exigem que o contador seja operado em frequências além daquelas que um contador assíncrono pode manipular. Num contador assíncrono, a sequência de contagem deve seguir a sequência ascendente ou descendente regular, mas num contador síncrono, para além das sequências ascendentes ou descendentes regulares, podem também ser obtidos diferentes tipos de sequências de contagens irregulares.

Contadores síncronos podem ser construídos a partir de flip-flops discretos ou estão prontamente disponíveis na forma de CIs. As implementações de CI são projetadas para que os contadores possam ser configurados para uma ampla variedade de aplicações, desde a simples contagem até a divisão de frequência.

Etapas de projeto de contadores síncronos:

1. Especifique a sequência do contador e desenhe um diagrama de estados que descreva a operação do contador.
2. Defina o tipo e o número de flip-flops a serem usados.
3. Desenvolva uma tabela de "próximo estado" a partir do diagrama de estados.
4. Desenvolva uma tabela de transição, mostrando as entradas de flip-flop necessárias para cada transição usando a tabela de excitação do flip-flop escolhido.
5. Obter funções de entrada de flip-flop com base nos estados atuais e depois simplificá-las. Se você usar os mapas de Karnough, não esqueça de colocar "X" (não importa) nos mapas de Karnough para os valores de contagem não usados.
6. Desenhe o diagrama esquemático do contador.

Exemplo de projeto de contador síncrono:

Projetar um contador síncrono para contar na seguinte sequência

0→2→4→6→1→3→5→7→0→2 ...

Vamos agora seguir as etapas de projeto:

1 – Para este contador síncrono, a sequência do contador 0→2→4→6→1→3→5→7→0→2 … é desenhada como um diagrama de estado como mostrado na figura 10.13.

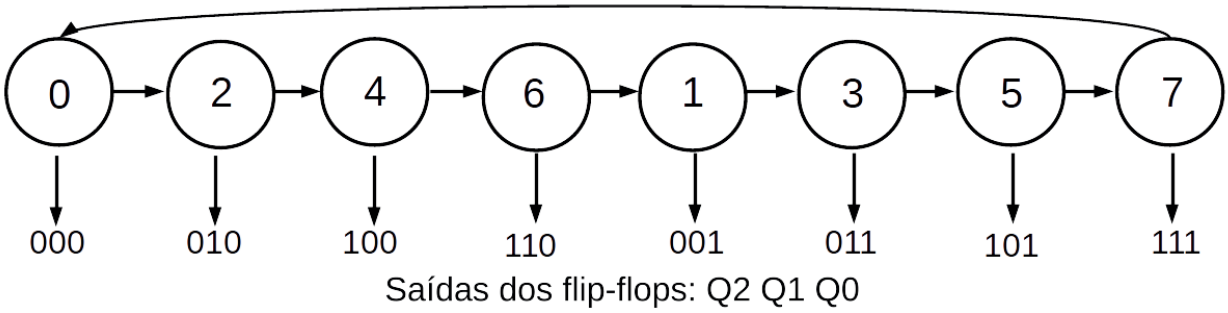


Figura 10.28 – O diagrama de estados (máquina de Moore) para o contador síncrono contar na sequência de contagem de 0→2→4→6→1→3→5→7→0→2 … e saídas dos flip-flop atribuídas a cada estado.

2 – O tipo de flip-flop para projetar este contador é escolhido como o flip-flop JK. O maior número dentro dos valores de contagem é 7, portanto, 3 flip-flops são suficientes para projetar este contador.

3 – A tabela de "próximo estado" para o contador síncrono contar na sequência de contagem de 0→2→4→6→1→3→5→7→0→2 … é executada a partir do diagrama de estados da figura 10.13 como mostrado na tabela 10.7.

Estado atual (t_n)		Próximo estado (t_{n+1})	
Binário Q2 Q1 Q0	Número decimal	Binário Q2 Q1 Q0	Número decimal
000	0	010	2
010	2	100	4
100	4	110	6
110	6	001	1
001	1	011	3
011	3	101	5
101	5	111	7
111	7	000	0

Tabela 10.7 – Próximo estado para o contador síncrono.

4 – A tabela de transição (tabela 10.8) para o contador síncrono contar na sequência de contagem de 0→2→4→6→1→3→5→7→0→2 … executada a partir da tabela de "próximo estado" da tabela 10.7 usando a tabela de excitação dada na tabela 10.9.

CP	Saídas		Transição de saída			Entradas					
	Q2Q1Q0	Q2Q1Q0	Q2	Q1	Q0	J2	K2	J1	K1	J0	K0
t_n	t_n	t_{n+1}	$t_n \rightarrow tn_{+1}$	$t_n \rightarrow t_{n+1}$	$t_n \rightarrow t_{n+1}$	t_n	t_n	t_n	t_n	t_n	t_n
0	000	010	0→0	0→1	0→0	0	x	1	x	0	x
1	010	100	0→1	1→0	0→0	1	x	x	1	0	x
2	100	110	1→1	0→1	0→0	x	0	1	x	0	x
3	110	001	1→0	1→0	0→1	x	1	x	1	1	x
4	001	011	0→0	0→1	1→1	0	x	1	x	x	0
5	011	101	0→1	1→0	1→1	1	x	x	1	x	0
6	101	111	1→1	0→1	1→1	x	0	1	x	x	0
7	111	000	1→0	1→0	1→0	x	1	x	1	x	1

Tabela 10.8 – Transição de saída para o contador síncrono.

$Qt_n \rightarrow \theta t_{n+1}$	J	K
0→0	0	x
0→1	1	x
1→0	x	1
1→1	x	0

Tabela 10.9 – Excitação do flip-flop JK.

5 – As funções de entrada dos flip-flop baseadas nos estados presentes são obtidas como segue da tabela de transição dada na tabela 10.8 (Σm = mintermos “1”, Σd = não importa “x”):

$$J2\ (Q2, Q1, Q0) = \Sigma m\ (2, 3) + \Sigma d\ (4, 5, 6, 7),$$

$$K2\ (Q2, Q1, Q0) = \Sigma m\ (6, 7) + \Sigma d\ (0, 1, 2, 3),$$

$$J1\ (Q2, Q1, Q0) = \Sigma m\ (0, 1, 4, 5) + \Sigma d\ (2, 3, 6, 7),$$

$$K1\ (Q2, Q1, Q0) = \Sigma m\ (2, 3, 6, 7) + \Sigma d\ (0, 1, 4, 5),$$

$$J0\ (Q2, Q1, Q0) = \Sigma m\ (6) + \Sigma d\ (1, 3, 5, 7),$$

$$K0\ (Q2, Q1, Q0) = \Sigma m\ (7) + \Sigma d\ (0, 2, 4, 6).$$

Essas funções de entrada são simplificadas usando mapas de Karnough como mostrado na figura 10.29.

	Q2	
Q1 Q0	0	1
0 0	0	x
0 1	0	x
1 1	1	x
1 0	1	x

J2 = Q1

	Q2	
Q1 Q0	0	1
0 0	x	0
0 1	x	0
1 1	x	1
1 0	x	1

K2 = Q1

	Q2	
Q1 Q0	0	1
0 0	1	1
0 1	1	1
1 1	x	x
1 0	x	x

J1 = 1

	Q2	
Q1 Q0	0	1
0 0	x	x
0 1	x	x
1 1	1	1
1 0	1	1

K1 = 1

	Q2	
Q1 Q0	0	1
0 0	0	0
0 1	x	x
1 1	x	x
1 0	0	1

J0 = Q2.Q1

	Q2	
Q1 Q0	0	1
0 0	x	x
0 1	0	0
1 1	0	1
1 0	x	x

K0 = Q2.Q1

Figura 10.29 – Simplificação das funções de entrada usando mapas de Karnough.

6 – A implementação do contador síncrono para contar na seguinte sequência 0→2→4→6→1→3→5→7→0→2 … usando os flip-flops JK é mostrada na figura 10.30.

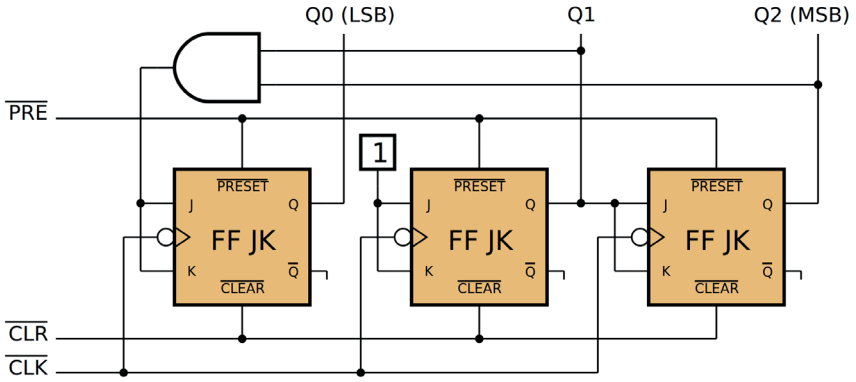


Figura 10.30 – Implementação do contador síncrono usando JK flip-flops.

Contador Síncrono de 4 Bits Crescente: O contador de circuito síncrono de 4 bits construído com flip-flops T é mostrado na figura 10.31. As funções de entrada do flip-flop T baseadas nos estados atuais são obtidas da seguinte forma:

T3 = Q2.Q1.Q0, T2 = Q1.Q0, T1 = Q0 , T0 = 1.

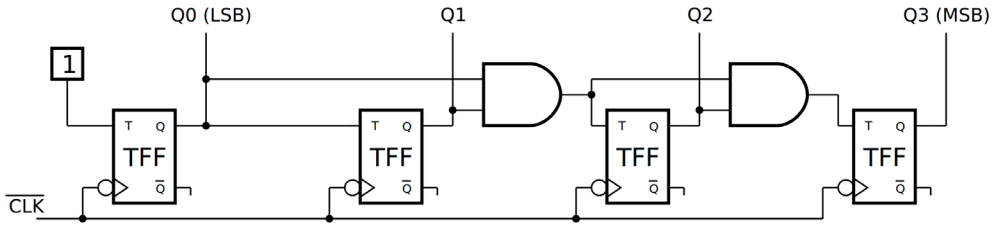


Figura 10.31 – Circuito contador síncrono de 4 bits construído com flip-flops T.

Contador Síncrono BCD (MOD10) Crescente: O circuito do contador síncrono BCD (MOD10) construído com flip-flops D é mostrado na figura 10.32. As funções de entrada dos flip-flops D baseadas nos estados atuais são obtidas da seguinte forma:

D3 = Q3.Q0’ + Q2.Q1.Q0, D2 = Q2.Q1’ + Q2.Q0’ + Q2’.Q1.Q0,

D1 = Q1.Q0’ + Q3’.Q1’.Q0, D0 = Q0

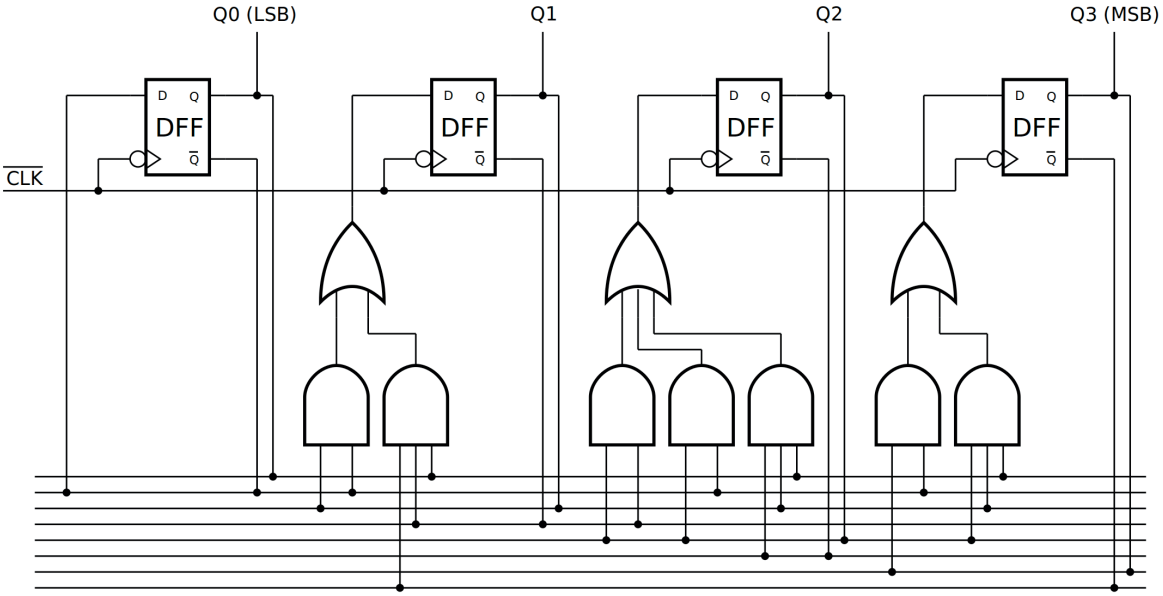


Figura 10.32 – Circuito do contador síncrono BCD (MOD10) construído com flip-flops D.

Contador Síncrono de 4 Bit Crescente/Decrescente: O circuito do contador síncrono de 4 bit crescente/decrescente construído com flip-flops T é mostrado na figura 10.33. As funções de entrada dos flip-flop T baseadas nos estados atuais são obtidas da seguinte forma:

T3 = K.Q2.Q1.Q0 + K’.Q2’.Q1’.Q0’, T2 = K.Q1.Q0 + K’.Q1’.Q0’,

T1 = K.Q0 + K’.Q0’, T0 = 1

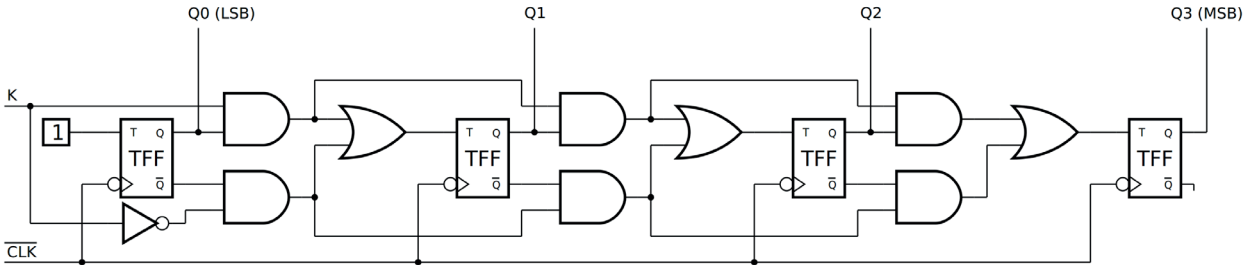


Figura 10.33 – Circuito do contador síncrono de 4 bits crescente/decrescente construído com flip-flops T.

10.4.3. Exame do contador síncrono para uma sequência específica

Montagem de um contador síncrono para contar na seguinte sequência 0→2→4→6→1→3→5→7→0→2 … usando flip-flops JK.

Esquemas:

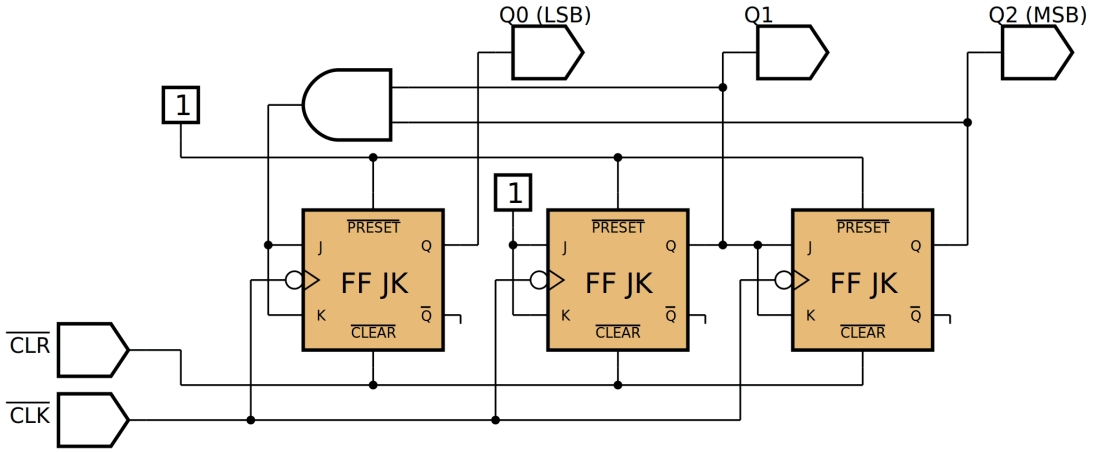


Figura 10.34 – Diagrama esquemático.

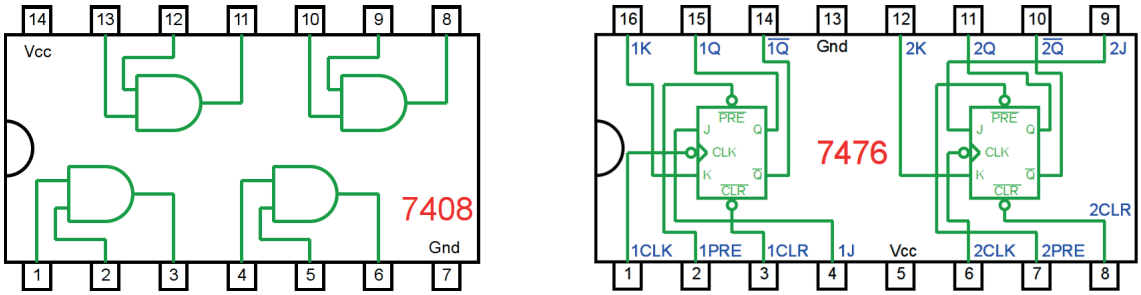


Figura 10.35 – Circuitos integrados TTL.

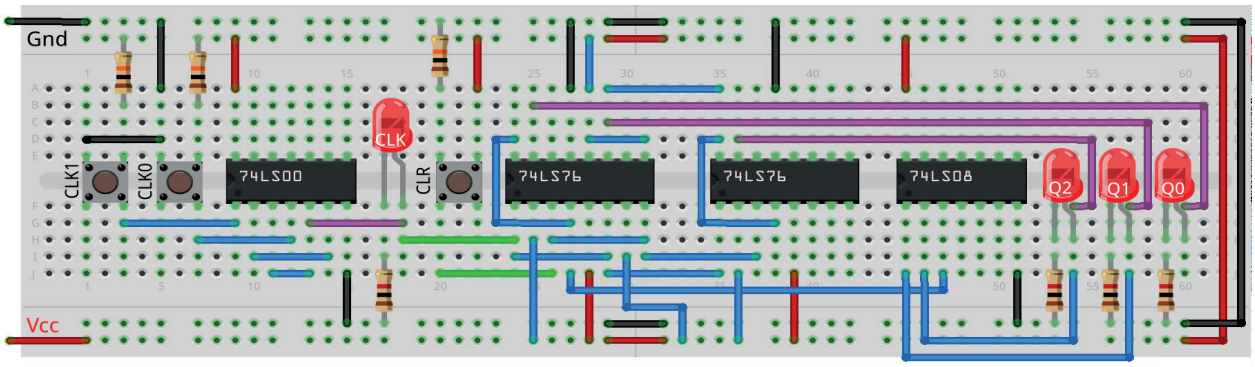


Figura 10.36 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 10.36.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 10.10, use os botões como segue:
   a) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   b) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   c) Um pulso de relógio completo é conseguido pressionando o botão CLK1 e depois pressionando o botão CLK0.
3. Para o preenchimento das colunas de saída na tabela 10.10, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 10.34. Compare os resultados com a tabela 10.10.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 10.10.

Tabelas de dados:

Pulso	Saídas			Valor decimal
	Q2	Q1	Q0	
CLR				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

Tabela 10.10

Programa para o Arduíno:

```
// gerador de sequencia binária de 3 bits

byte q1[] = {0,1,0,1,0,1,0,1};
byte q2[] = {0,0,1,1,0,0,1,1};
byte q3[] = {0,0,0,0,1,1,1,1};
byte sq[] = {0,2,4,6,1,3,5,7};
int intervalo;
byte i; byte j; byte q;
byte clka; byte clkd;
byte q1d; byte q2d; byte q3d; byte decimal;

void gera_clock(){
  delay(intervalo);
  clkd = !clka;
}

void gerador3bits(){
  if(clka == 1 && clkd == 0){
    q1d = q1[sq[j]];
    q2d = q2[sq[j]];
    q3d = q3[sq[j]];
    j = j + 1; if(j > 7) j = 0;
    decimal = 4*q3d + 2*q2d + q1d;
  }
}

void mostra_gerador3bits(){
  Serial.print("| "); if (i < 10) Serial.print("0");
  Serial.print(i); Serial.print(" ");
  if (q3d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q2d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (clkd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print(" | ");
  Serial.println(decimal);
}
```

```
void setup(){
  Serial.begin(9600);
  Serial.println("Gerador de sequencia binária de 3 bits");
  Serial.println("| Pulso | Q3 | Q2 | Q1 | Clk | Decimal");
  Serial.println("----------------------------------------");

  q = 36;
  j = 0;
  clka = 0;
  intervalo = 100;

  for (i=1;i<q;i++){
    gera_clock();
    gerador3bits();
    mostra_gerador3bits();
    clka = clkd;
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

10.4.4. Exame do contador síncrono de 4 bits crescente usando flip-flops JK.

Esquemas:

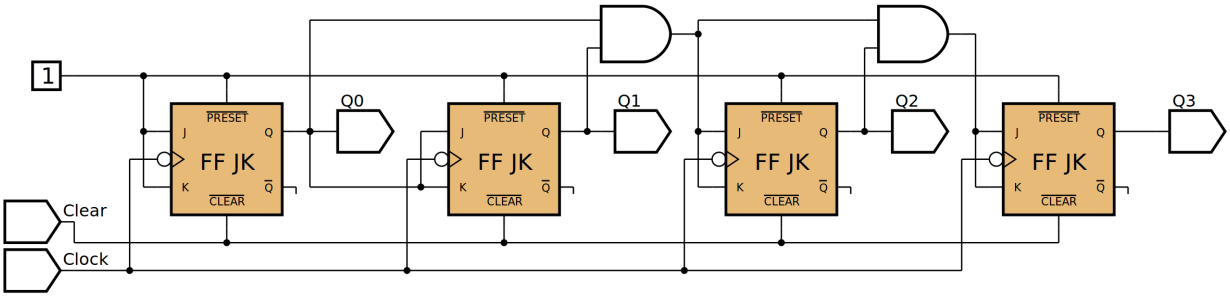


Figura 10.37 – Diagrama esquemático.

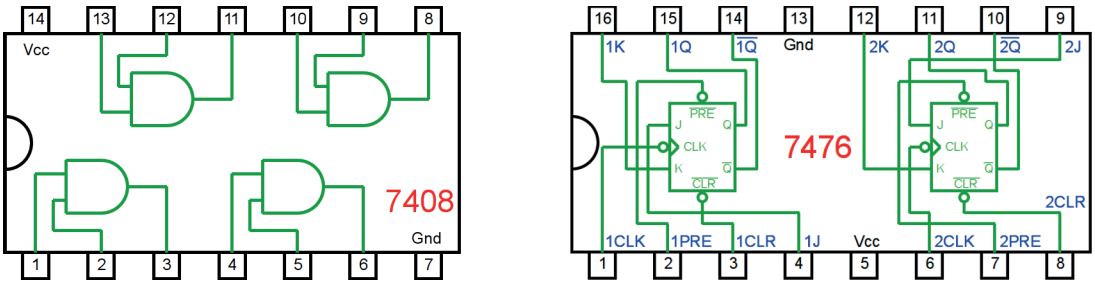


Figura 10.38 – Circuitos integrados TTL.

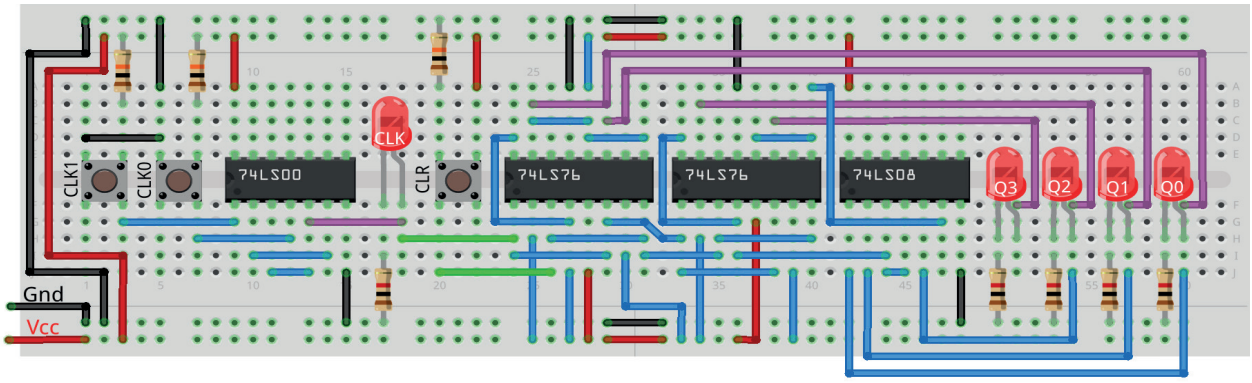


Figura 10.39 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 10.39.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 10.11, use os botões como segue:
   a) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   b) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   c) Um pulso de relógio completo é conseguido pressionando o botão CLK1 e depois pressionando o botão CLK0.
3. Para o preenchimento das colunas de saída na tabela 10.11, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 10.37. Compare os resultados com a tabela 10.11.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 10.11.

Tabelas de dados:

Pulso	Saídas				Valor decimal
	Q3	Q2	Q1	Q0	
CLR					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

Tabela 10.11

Programa para o Arduíno:

```
// gerador de sequencia binária de 4 bits

byte q1[] = {0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1};
byte q2[] = {0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1};
byte q3[] = {0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1};
byte q4[] = {0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1};
byte sq[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};

int intervalo;
byte i; byte j; byte q;
byte clka; byte clkd;
byte q1d; byte q2d; byte q3d; byte q4d; byte decimal;

void gera_clock(){
  delay(intervalo);
  clkd = !clka;
}

void gerador4bits(){
  if(clka == 1 && clkd == 0){
    q1d = q1[sq[j]];
    q2d = q2[sq[j]];
    q3d = q3[sq[j]];
    q4d = q4[sq[j]];
    j = j + 1; if(j > 15) j = 0;
    decimal = 8*q4d + 4*q3d + 2*q2d + q1d;
  }
}

void mostra_gerador4bits(){
  Serial.print("| "); if (i < 10) Serial.print("0");
  Serial.print(i); Serial.print(" ");
  if (q4d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q3d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q2d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (clkd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print(" | ");
  Serial.println(decimal);
}
```

```
void setup(){
  Serial.begin(9600);
  Serial.println("Gerador de sequencia binária de 4 bits");
  Serial.println("| Pulso | Q4 | Q3 | Q2 | Q1 | Clk | Decimal");
  Serial.println("--------------------------------------------");

  q = 50; j = 0; clka = 0; intervalo = 100;

  for (i=1;i<q;i++){
    gera_clock();
    gerador4bits();
    mostra_gerador4bits();
    clka = clkd;
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

10.4.5. Exame do contador síncrono BCD crescente usando flip-flops JK

Esquemas:

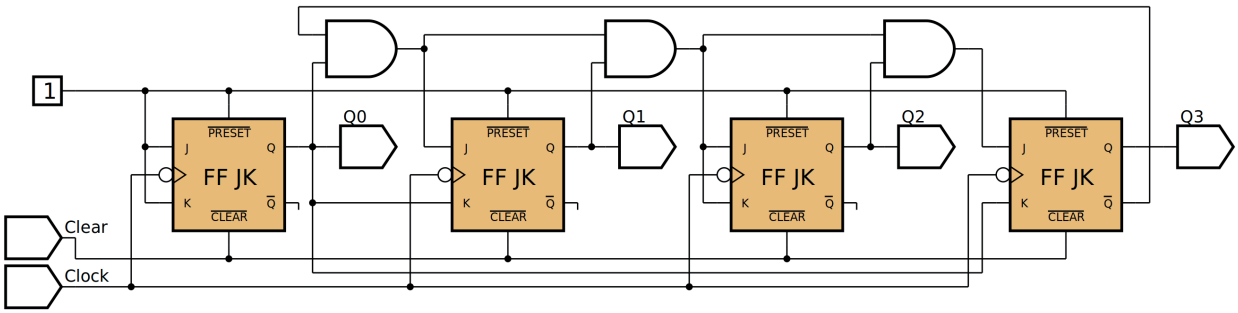


Figura 10.40 – Diagrama esquemático.

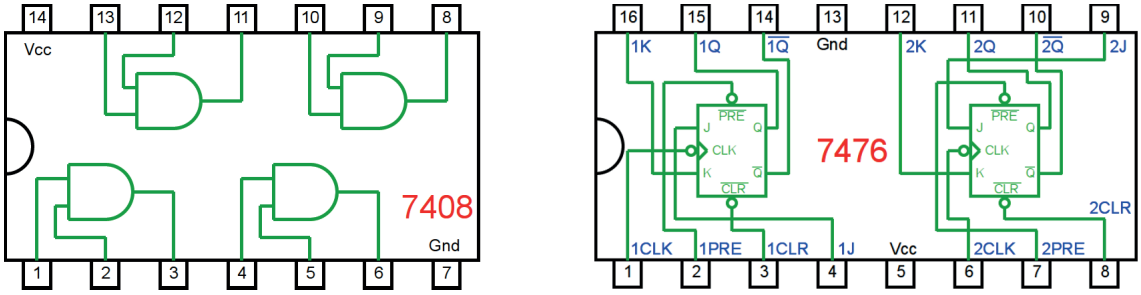


Figura 10.41 – Circuitos integrados TTL.

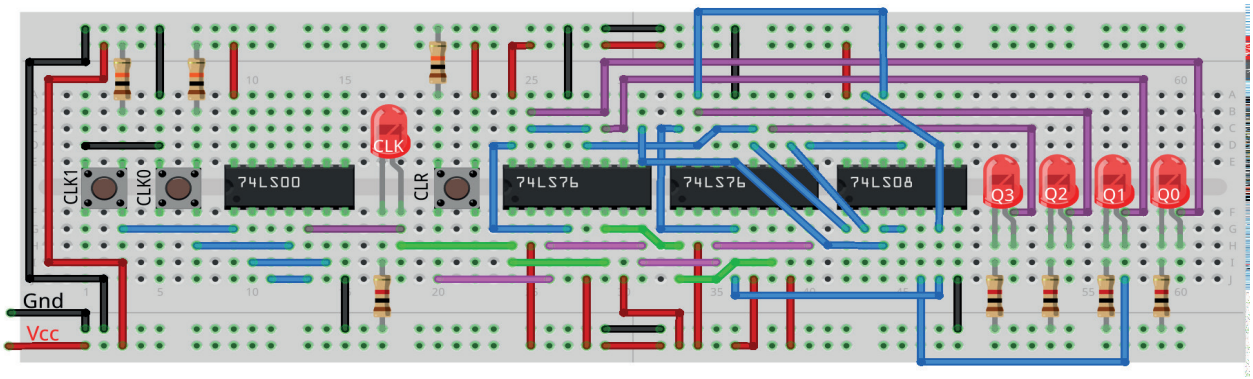


Figura 10.42 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 10.42.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada de relógio (verdes).
   d) Conecte os fios de entrada de “clear” (rosa)
   e) Conecte os fios de conexão (azuis).
   f) Conecte os fios vermelhos (Vcc) e pretos (Gnd).

g) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
h) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.

2. Para o preenchimento da tabela 10.12, use os botões como segue:
   a) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   b) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   c) Um pulso de relógio completo é conseguido pressionando o botão CLK1 e depois pressionando o botão CLK0.
3. Para o preenchimento das colunas de saída na tabela 10.12, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 10.40. Compare os resultados com a tabela 10.12.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 10.12.

Tabelas de dados:

Pulso	Saídas				Valor decimal
	Q3	Q2	Q1	Q0	
CLR					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

Tabela 10.12

Programa para o Arduíno:

```
// gerador de sequencia binária de 4 bits (BCD)

byte q1[] = {0,1,0,1,0,1,0,1,0,1};
byte q2[] = {0,0,1,1,0,0,1,1,0,0};
byte q3[] = {0,0,0,0,1,1,1,1,0,0};
byte q4[] = {0,0,0,0,0,0,0,0,1,1};
byte sq[] = {0,1,2,3,4,5,6,7,8,9};

int intervalo;
byte i; byte j; byte q;
byte clka; byte clkd;
byte q1d; byte q2d; byte q3d; byte q4d; byte decimal;

void gera_clock(){
  delay(intervalo);
  clkd = !clka;
}

void gerador4bits(){
  if(clka == 1 && clkd == 0){
    q1d = q1[sq[j]];
    q2d = q2[sq[j]];
    q3d = q3[sq[j]];
    q4d = q4[sq[j]];
    j = j + 1; if(j > 9) j = 0;
    decimal = 8*q4d + 4*q3d + 2*q2d + q1d;
  }
}

void mostra_gerador4bits(){
  Serial.print("| "); if (i < 10) Serial.print("0");
  Serial.print(i); Serial.print(" ");
  if (q4d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q3d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q2d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (q1d) Serial.print("| 1 "); else Serial.print("| 0 ");
  if (clkd) Serial.print("| 1 "); else Serial.print("| 0 ");
  Serial.print(" | ");
  Serial.println(decimal);
  }
```

```
  void setup(){
  Serial.begin(9600);
  Serial.println("Gerador de sequencia binária de 4 bits BCD");
  Serial.println("| Pulso | Q4 | Q3 | Q2 | Q1 | Clk | Decimal");
  Serial.println("--------------------------------------------");

  q = 50;
  j = 0;
  clka = 0;
  intervalo = 100;

  for (i=1;i<q;i++){
    gera_clock();
    gerador4bits();
    mostra_gerador4bits();
    clka = clkd;
  }
} // fim do 'setup'

void loop(){
  // nada a fazer aqui!
} // fim do 'loop'
```

Capítulo 11. Registradores de deslocamento

11.1. Objetivos

1. Aprender os princípios da operação de registradores de deslocamento.
2. Aprender a usar registradores de deslocamento como registradores de deslocamento à direita.
3. Aprender a usar registradores de deslocamento como registradores de deslocamento à esquerda.
4. Aprender os princípios de "Entrada Paralela, Saída Paralela, Entrada Serial, Saída Serial".
5. Familiarizar-se com os CIs 74LS174, 74LS194, 74LS195, 74LS165 e 74LS164.

11.2. Informação preliminar

Registradores de deslocamento podem armazenar e deslocar dados binários. Devido a essa propriedade, eles são usados em processos como registro de dados, subtração de dados binários e transferências de dados no computador. Eles também são usados na multiplicação e divisão binárias. Multiplicação e divisão são outra forma de mudança nos números binários. Por exemplo, se um número binário for deslocado para a esquerda, o valor decimal do número será multiplicado por 2. Da mesma forma, se um número binário for deslocado para a esquerda, o valor decimal dele será dividido por 2. Registradores de deslocamento são compostos de vários flip-flops conectados um após o outro, ou seja, a saída de um está conectada à entrada do próximo. Todos os flip-flops têm o mesmo relógio, portanto a transferência de dados é feita simultaneamente com os pulsos de relógio. A razão pela qual eles operam com um relógio comum é garantir a transferência síncrona de dados de um flip-flop para o outro. A transferência de dados ocorre com a borda descendente ou a borda de subida do relógio de acordo com o tipo de flip-flop. Os registradores de deslocamento podem ser classificados com o número de bits processados, tipo de entrada e saída e direção de deslocamento dos bits. O número de flip-flops em um registrador de deslocamento depende do número de bits armazenados ou processados (cada flip-flop armazena um bit de dados). Existem registradores do tipo de 4 bits, 8 bits e 16 bits.

O movimento de dados básico em um registrador pode ser um dos seguintes:

- Entrada serial / deslocamento para direita / saída serial.
- Entrada serial / deslocamento para esquerda / saída serial.
- Entrada paralela / saída serial.
- Entrada serial / saída paralela.
- Entrada paralela / saída paralela.
- Rotação à direita.
- Rotação à esquerda.

11.3. Registradores de deslocamento compostos por flip-flops

11.3.1. Objetivos

1. Analisar os registradores de deslocamento compostos de flip-flops,
2. Observar sua operação.

11.3.2. Informação preliminar

Registradores de deslocamento consistem em flip-flops e são importantes em aplicações que envolvem o armazenamento de dados em um sistema digital. Um registrador, ao contrário de um contador, não possui uma sequência especificada de estados, exceto em certas aplicações especializadas. Um registrador, em geral, é usado somente para armazenar e transferir dados (1s e 0s) inseridos a partir de uma fonte externa. A figura 11.1 mostra um registrador de deslocamento de 4 bits com entrada serial e saída paralelo composto de 4 flip-flops D. Uma amostra do diagrama de tempos e uma amostra da tabela de saídas para um registrador de deslocamento de 4 bits com entrada serial e saída paralelo são fornecidas na figura 11.2 e na tabela 11.1, respectivamente.

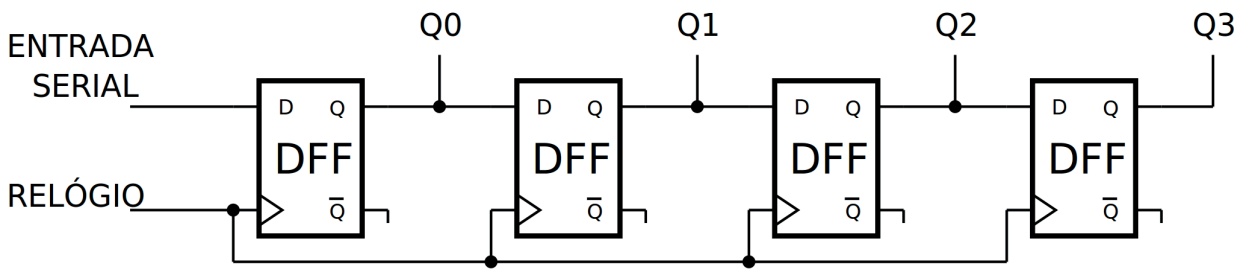


Figura 11.1 – Diagrama esquemático para um registrador de deslocamento de 4 bits com entrada em série e saída em paralelo composto de 4 flip-flops D.

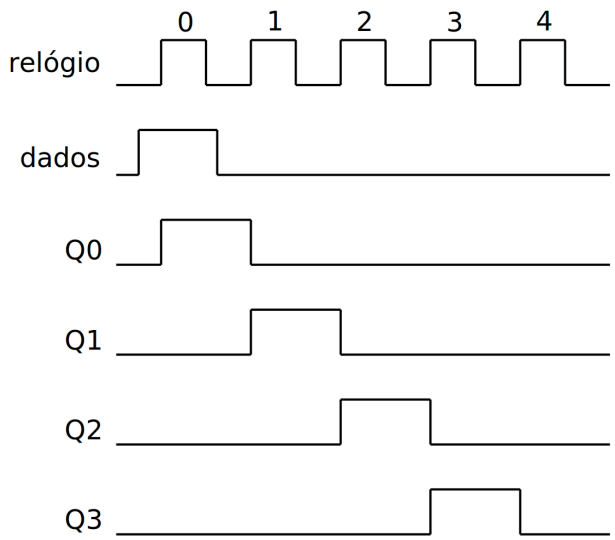


Figura 11.2 – Diagrama de tempos para um registrador de deslocamento de 4 bits com entrada em série e saída em paralelo.

Entradas		Saídas			
RELÓGIO	DADOS	Q0	Q1	Q2	Q3
0	1	1	0	0	0
1	0	0	1	0	0
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0

Tabela 11.1 – Tabela de saída para um registrador de deslocamento de 4 bits com entrada em série e saída em paralelo.

11.3.3. Circuito Integrado TTL 74LS174 – 6 flip-flops tipo D

O 74LS174 inclui seis flip-flops do tipo D. Estes flip-flops com gatilho por borda positiva (borda ascendente) utilizam circuitos TTL para implementar a lógica de flip-flop tipo D. Todos têm uma entrada de desativação (clear) comum. As informações nas entradas D que atendem aos requisitos de tempo de configuração são transferidas para as saídas Q na borda de subida positiva do pulso de relógio. O gatilho do relógio ocorre em um determinado nível de tensão e não está diretamente relacionado ao tempo de transição do pulso de baixo para alto. Quando a entrada do relógio está no nível ALTO ou BAIXO, o sinal de entrada D não tem efeito na saída. A figura 11.3 mostra a pinagem, o símbolo lógico, a tabela de funções e o diagrama esquemático do CI 74LS174 com 6 flip-flops D gatilháveis por borda ascendente e com “clear”. O 74LS174 pode ser facilmente usado para implementar registradores de deslocamento. Portanto, os três primeiros experimentos incluem registradores compostos com CI 74LS174.

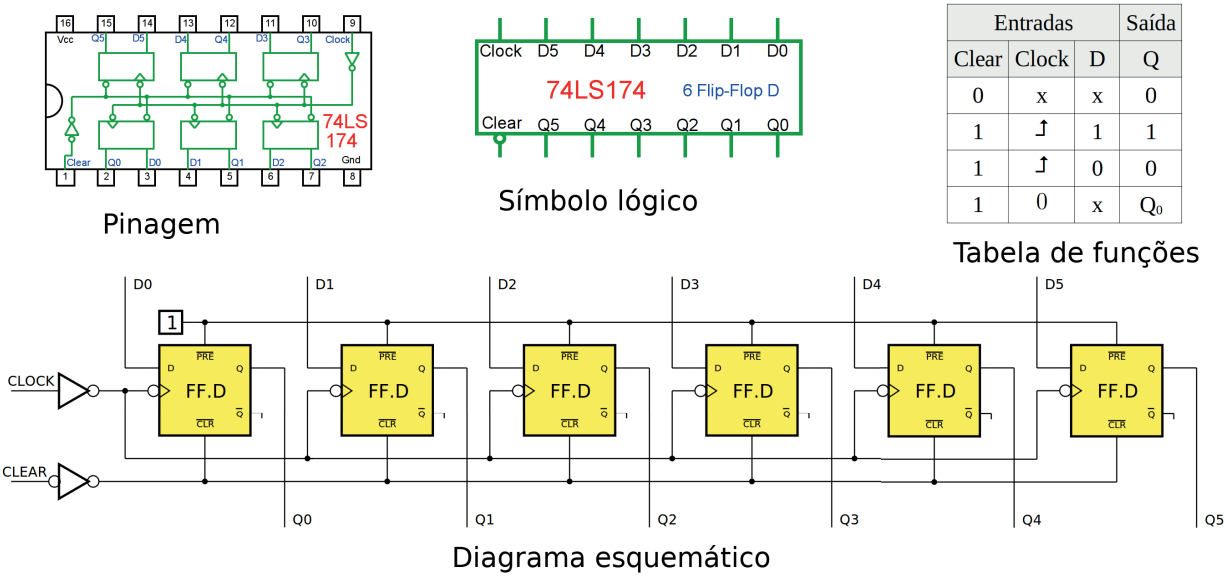


Figura 11.3 – A pinagem, o símbolo lógico, a tabela de funções e o diagrama esquemático do CI 74LS174 com 6 flip-flops D gatilháveis por borda ascendente e com “clear”.

11.3.4. Exame do registrador de 4 bits 74LS174 (direita)

Configuração do registrador de deslocamento para direita (Q0 → Q1 → Q2 → Q3), com entrada serial e saída paralela, composto por flip-flops.

Esquemas:

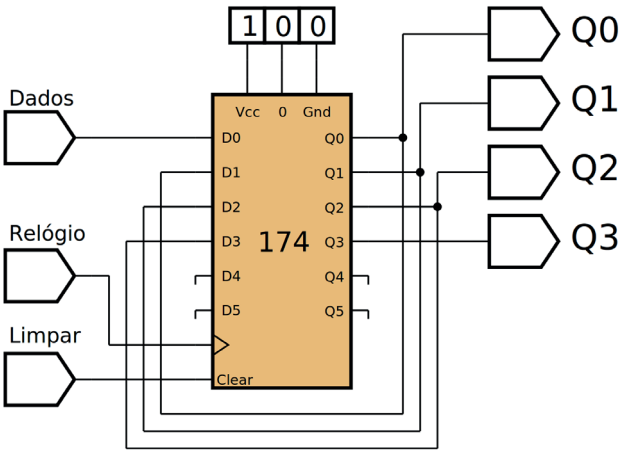


Figura 11.4 – Diagrama esquemático.

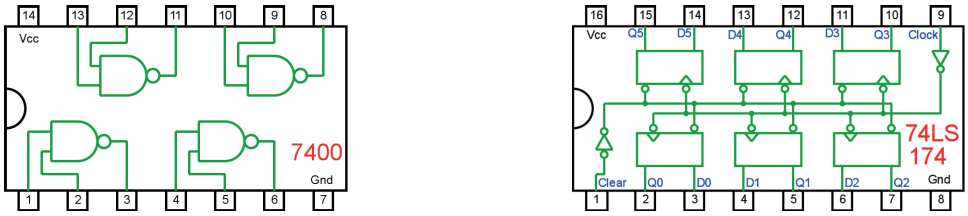


Figura 11.5 – Circuitos integrados TTL.

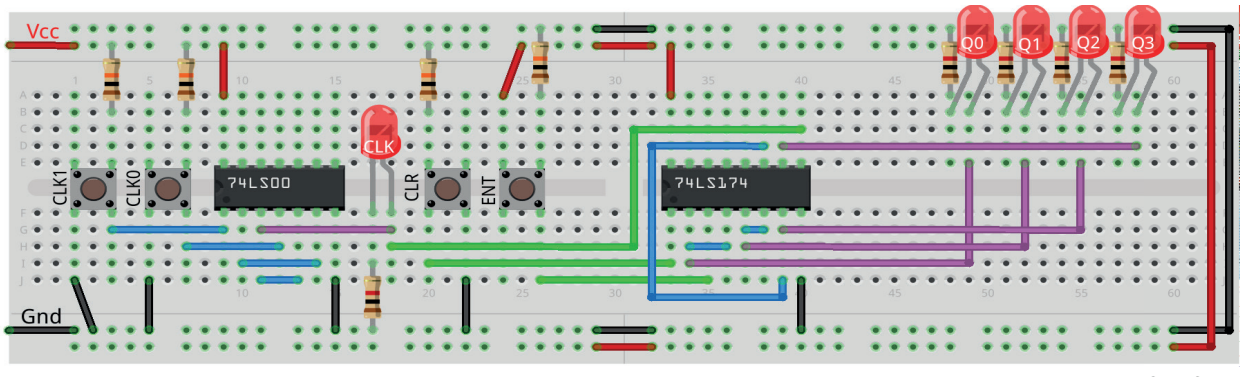


Figura 11.6 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 11.6.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 11.2, use os botões como segue:
   a) O botão CLR pressionado envia um ativo baixo “0” e apaga todos os leds (Limpar).
   b) O botão ENT pressionado envia um ativo alto “1” para a entrada do registrador (Dados). Mantenha pressionado “1”, ou não pressionado “0”, antes de executar o pulso de relógio.
   c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   e) Um pulso de relógio completo é conseguido pressionando primeiro o botão CLK1 e depois pressionando o botão CLK0 (Relógio).
3. Para o preenchimento das colunas de saída na tabela 11.2, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 11.4. Compare os resultados com a tabela 11.2.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 11.2.

Tabelas de dados:

Entradas			Saída			
Pulso de relógio	Limpar (CLR)	Dados (ENT)	Q0	Q1	Q2	Q3
0	0	1				
1	1	1				
2	1	0				
3	1	0				
4	1	0				
5	1	0				

Tabela 11.2

Programa para o Arduíno:

Monte o circuito na placa de montagem, conforme figura 11.7. Conecte os fios azuis e roxos aos respectivos pinos do Arduíno. Conecte os fios vermelho (Vcc) e preto (Gnd) nos terminais apropriados do Arduíno.

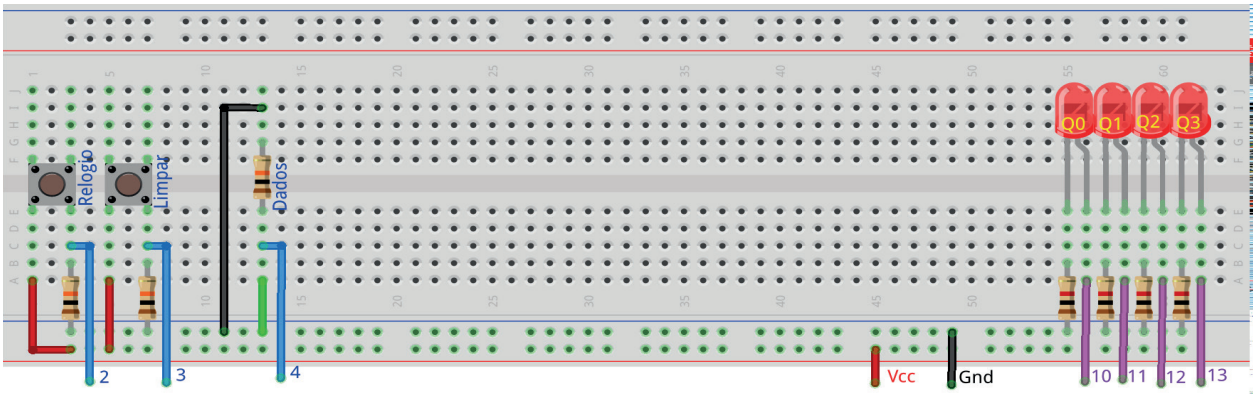


Figura 11.7 – Disposição dos componentes na placa de montagem.

Uso:

- O Botão "Limpar" apaga todos os LEDs, independente do pulso de relógio.
- Os dados de entrada devem ser configurados como: se for "0" o fio verde deve ser conectado na barra Gnd, se for "1" o fio verde deve ser conectado na barra Vcc. Esta operação deve ser feita antes do pulso de relógio.
- O botão "Relógio" transfere o valor de "Dados" para a posição inicial à esquerda do registrador, deslocando os outros valores: "Dados → Q0 → Q1 → Q2 → Q3 → Perde"

```
// Emulador do 74LS174 - 6 flip-flop D com Clear
// configurado para registrador de deslocamento para direita de 4 bits
// entrada serial e saída paralela

// pinos de entrada
  #define pino_relogio 2
  #define pino_limpar 3
  #define pino_dados 4

// pinos de saída
  #define pino_ledq0 10
  #define pino_ledq1 11
  #define pino_ledq2 12
  #define pino_ledq3 13

// valores do 74LS174
  byte d0; byte d1; byte d2; byte d3; byte d4; byte d5;
```

```
  byte q0; byte q1; byte q2; byte q3; byte q4; byte q5;
  byte clk; byte clk_old; byte clr;

// valores de entrada
  byte relogio; byte limpar; byte dados;
  byte relogio_antes; byte limpar_antes; byte dados_antes;

void ci_74LS174(){
  if (clk && !clk_old){
    q0 = d0; q1 = d1; q2 = d2; q3 = d3; q4 = d4; q5 = d5;
    clk_old = clk;}
  if (!clr){q0 = 0; q1 = 0; q2 = 0; q3 = 0; q4 = 0; q5 = 0;}
} // fim do ‘ci_74LS174’

void setup(){

// inicialização dos pinos de entrada
  pinMode(pino_relogio, INPUT);
  pinMode(pino_limpar, INPUT);
  pinMode(pino_dados, INPUT);

// inicialização dos pinos de saída
  pinMode(pino_ledq0, OUTPUT);
  pinMode(pino_ledq1, OUTPUT);
  pinMode(pino_ledq2, OUTPUT);
  pinMode(pino_ledq3, OUTPUT);

// inicialização das variáveis
  d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0;
  q0 = 0; q1 = 0; q2 = 0; q3 = 0; q4 = 0; q5 = 0;
  relogio_antes = 0;
  limpar_antes = 0;
  dados_antes = 0;
  clk_old = 0;
} // fim do ‘setup’

void loop(){

// leitura das entradas
  relogio = digitalRead(pino_relogio);
  if (relogio != relogio_antes) delay(100);
```

```
  limpar = digitalRead(pino_limpar);
  if (limpar != limpar_antes) delay(100);
  dados = digitalRead(pino_dados);
  if (dados !=dados_antes) delay (100);

// configuração para registrador de deslocamento
  clk = relogio;
  clr = !limpar;
  d0 = dados;
  d1 = q0;
  d2 = q1;
  d3 = q2;

// aplica valores ao CI virtual
  ci_74LS174();

// mostra saídas
  digitalWrite(pino_ledq0,q0);
  digitalWrite(pino_ledq1,q1);
  digitalWrite(pino_ledq2,q2);
  digitalWrite(pino_ledq3,q3);

// reinicia variáveis
  relogio_antes = relogio;
  limpar_antes = limpar;
  dados_antes = dados;

} // fim do ‘loop’
```

11.3.5. Exame do registrador de 4 bits 74LS174 (esquerda)

Configuração do registrador de deslocamento para esquerda (Q0 ← Q1 ← Q2 ← Q3), com entrada serial e saída paralela, composto por flip-flops.

Esquemas:

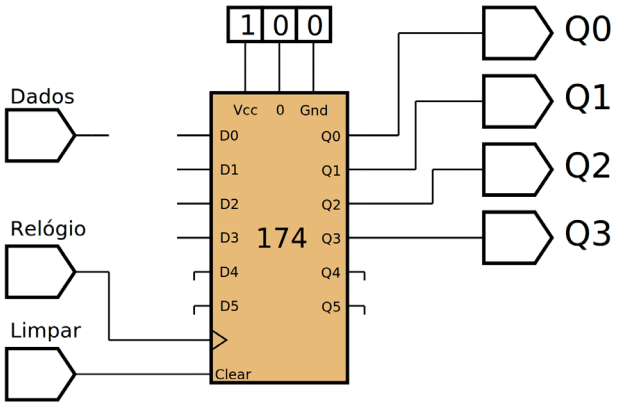


Figura 11.8 – Diagrama esquemático.

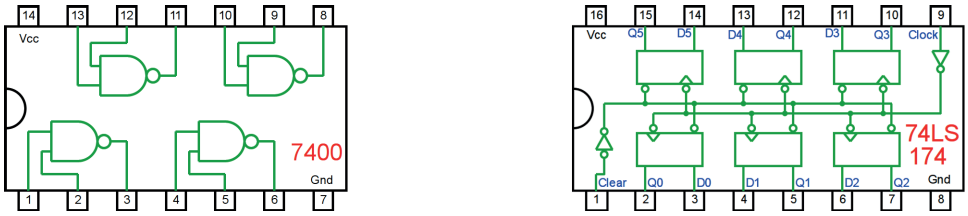


Figura 11.9 – Circuitos integrados TTL.

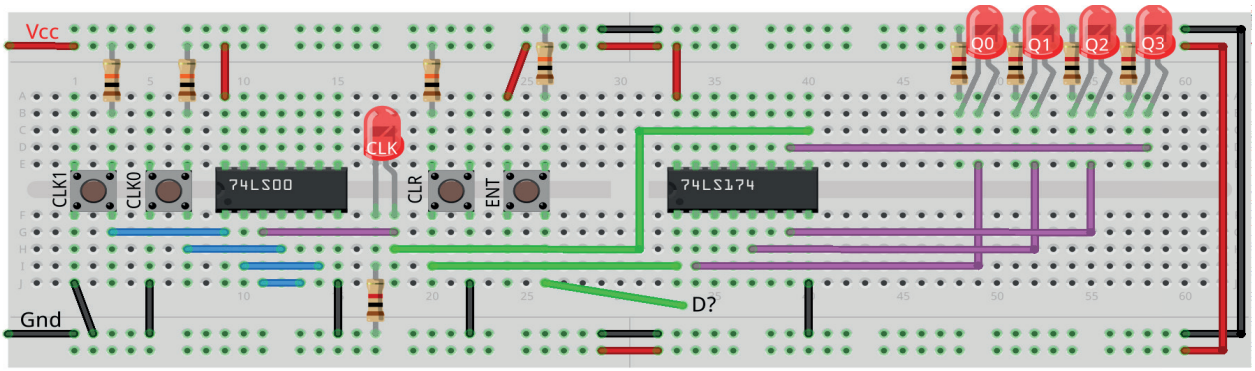


Figura 11.10 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Complete o desenho da figura 11.8 para implementar um registrador de deslocamento para esquerda (Q0 ← Q1 ← Q2 ← Q3). Complete as conexões na placa de montagem da figura 11.10.
2. Monte o circuito na placa de montagem, conforme figura 11.10.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 11.3, use os botões como segue:
   a) O botão CLR pressionado envia um ativo baixo “0” e apaga todos os leds (Limpar).
   b) O botão ENT pressionado envia um ativo alto “1” para a entrada do registrador (Dados). Mantenha pressionado “1”, ou não pressionado “0”, antes de executar o pulso de relógio.
   c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   e) Um pulso de relógio completo é conseguido pressionando primeiro o botão CLK1 e depois pressionando o botão CLK0 (Relógio).
3. Para o preenchimento das colunas de saída na tabela 11.3, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 11.8. Compare os resultados com a tabela 11.3.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 11.3.

Tabelas de dados:

Entradas			Saída			
Pulso de relógio	Limpar (CLR)	Dados (ENT)	Q0	Q1	Q2	Q3
0	0	1				
1	1	1				
2	1	0				
3	1	0				
4	1	0				
5	1	0				

Tabela 11.3

Programa para o Arduíno:

Considere a mesma montagem e o mesmo programa do tópico 11.3.4. mas faça modificações nas seguintes linhas, alterando o nome da variável necessária:

```
// configuração para registrador de deslocamento
  clk = relogio;
  clr = !limpar;
  d0 = _______;
  d1 = _______;
  d2 = _______;
  d3 = _______;
```

11.3.6. Exame do registrador de 4 bits 74LS174 com entrada e saída paralela

Esquemas:

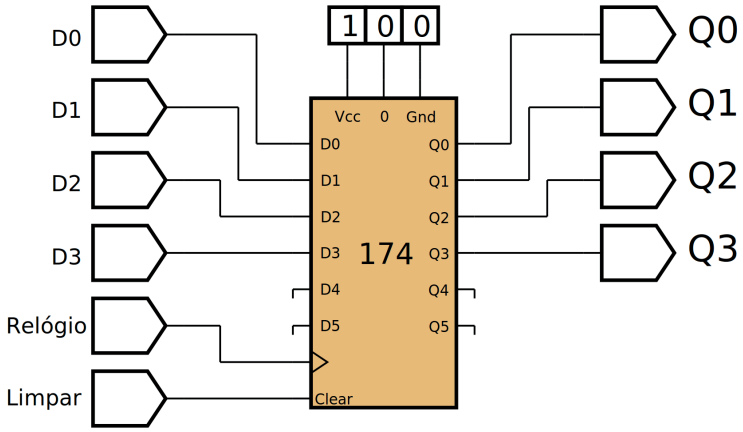


Figura 11.11 – Diagrama esquemático.

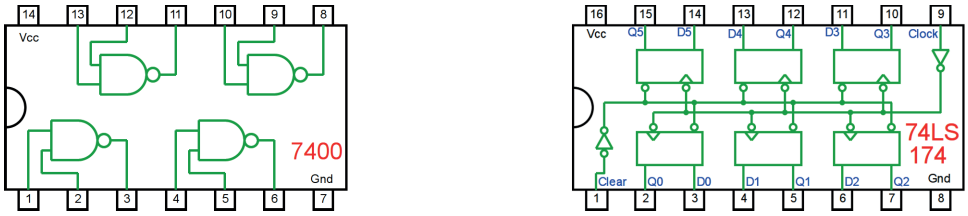


Figura 11.12 – Circuitos integrados TTL.

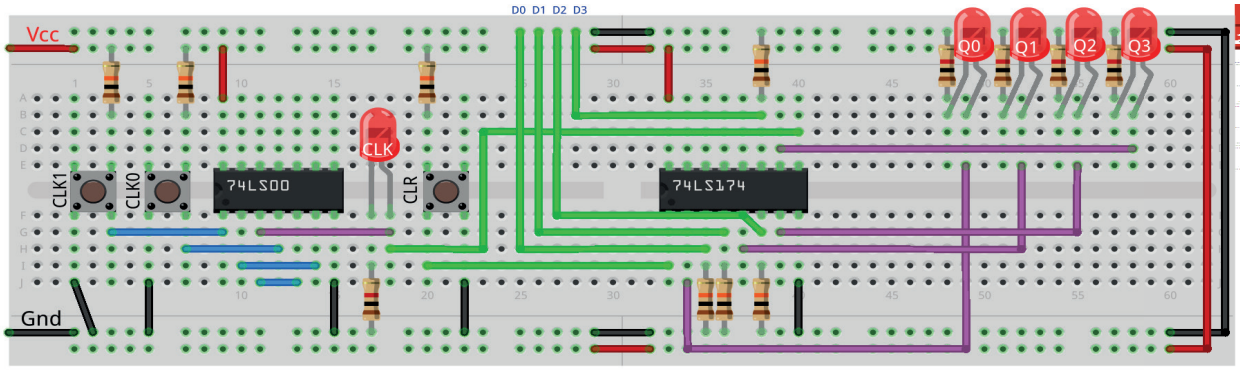


Figura 11.13 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 11.13.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 11.4, use os botões como segue:
   a) O botão CLR pressionado envia um ativo baixo “0” e apaga todos os leds (Limpar).
   b) Para configurar os dados de entrada, conecte os fios D0, D1, D2 e D3 na barra Gnd para “0” e na barra Vcc para “1”.
   c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   e) Um pulso de relógio completo é conseguido pressionando primeiro o botão CLK1 e depois pressionando o botão CLK0 (Relógio).
3. Para o preenchimento das colunas de saída na tabela 11.4, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 11.11. Compare os resultados com a tabela 11.4.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 11.4.

Tabelas de dados:

Entradas						Saída			
Pulso de relógio	Limpar (CLR)	D0	D1	D2	D3	Q0	Q1	Q2	Q3
0	0	0	1	0	1				
1	1	1	1	1	0				
2	1	0	1	1	0				
3	1	1	0	0	1				
4	1	1	0	1	0				
5	1	1	1	1	1				

Tabela 11.4

Programa para o Arduíno:

Monte o circuito na placa de montagem, conforme figura 11.14. Conecte os fios azuis e roxos aos respectivos pinos do Arduíno. Conecte os fios vermelho (Vcc) e preto (Gnd) nos terminais apropriados do Arduíno.

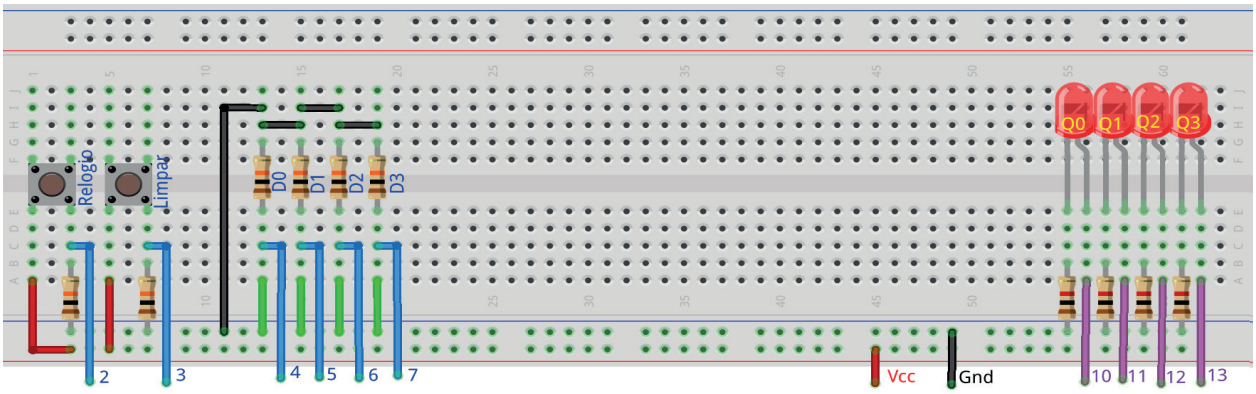


Figura 11.14 – Disposição dos componentes na placa de montagem.

Uso:

- O Botão "Limpar" apaga todos os LEDs, independente do pulso de relógio.
- Os dados de entrada ("D0 D1 D2 D3") devem ser configurados como: se for "0" o fio verde deve ser conectado na barra "Gnd"; se for "1" o fio verde deve ser conectado na barra "Vcc". Esta operação deve ser feita antes do pulso de relógio.
- O botão "Relógio" transfere o valor de "D0 D1 D2 D3" para "Q0 Q1 Q2 Q3" respectivamente.

```
// Emulador do 74LS174 - 6 flip-flop D com Clear
// configurado para registrador de carregamento paralelo de 4 bits
// entrada paralela e saída paralela

// pinos de entrada
  #define pino_relogio 2
  #define pino_limpar 3
  #define pino_dado0 4
  #define pino_dado1 5
  #define pino_dado2 6
  #define pino_dado3 7

// pinos de saída
  #define pino_ledq0 10
  #define pino_ledq1 11
  #define pino_ledq2 12
  #define pino_ledq3 13
```

```
// valores do 74LS174
  byte d0; byte d1; byte d2; byte d3; byte d4; byte d5;
  byte q0; byte q1; byte q2; byte q3; byte q4; byte q5;
  byte clk; byte clk_old; byte clr;

// valores de entrada
  byte relogio; byte limpar; byte dado0; byte dado1; byte dado2; byte
  dado3;
  byte relogio_antes; byte limpar_antes;

  void ci_74LS174(){
  if (clk && !clk_old){
  q0 = d0; q1 = d1; q2 = d2; q3 = d3; q4 = d4; q5 = d5;
  clk_old = clk;}
  if (!clr){q0 = 0; q1 = 0; q2 = 0; q3 = 0; q4 = 0; q5 = 0;}
} // fim do ‘ci_74LS174’

void setup(){

// inicialização dos pinos de entrada
  pinMode(pino_relogio, INPUT);
  pinMode(pino_limpar, INPUT);
  pinMode(pino_dado0, INPUT);
  pinMode(pino_dado1, INPUT);
  pinMode(pino_dado2, INPUT);
  pinMode(pino_dado3, INPUT);

// inicialização dos pinos de saída
  pinMode(pino_ledq0, OUTPUT);
  pinMode(pino_ledq1, OUTPUT);
  pinMode(pino_ledq2, OUTPUT);
  pinMode(pino_ledq3, OUTPUT);

// inicialização das variáveis
  d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 0; d5 = 0;
  q0 = 0; q1 = 0; q2 = 0; q3 = 0; q4 = 0; q5 = 0;
  relogio_antes = 0;
  limpar_antes = 0;
  clk_old = 0;

} // fim do ‘setup’
```

```
void loop(){

// leitura das entradas
  relogio = digitalRead(pino_relogio);
  if (relogio != relogio_antes) delay(100);
  limpar = digitalRead(pino_limpar);
  if (limpar != limpar_antes) delay(100);
  dado0 = digitalRead(pino_dado0);
  dado1 = digitalRead(pino_dado1);
  dado2 = digitalRead(pino_dado2);
  dado3 = digitalRead(pino_dado3);


// configuração para registrador de carregamento paralelo
  clk = relogio;
  clr = !limpar;
  d0 = dado0;
  d1 = dado1;
  d2 = dado2;
  d3 = dado3;

// aplica valores ao CI virtual
  ci_74LS174();

// mostra saídas
  digitalWrite(pino_ledq0,q0);
  digitalWrite(pino_ledq1,q1);
  digitalWrite(pino_ledq2,q2);
  digitalWrite(pino_ledq3,q3);

// reinicia variáveis
  relogio_antes = relogio;
  limpar_antes = limpar;

} // fim do ‘loop’
```

11.4. Circuito Integrado TTL 74LS194 – Registrador universal de deslocamento

11.4.1. Objetivos

1. Aprender diferentes princípios de operação de registradores de deslocamento.
2. Conhecer o CI registrador de deslocamento 74LS194.
3. Observar seu funcionamento e obter sua tabela verdade.

11.4.2. Informação preliminar

Os registradores de deslocamento podem ser compostos por CIs separados ou podem ser implementados em um único CI. O 74LS194 é um registrador de deslocamento universal de CI único. A figura 11.15 mostra as definições de pinos do 74LS194. O diagrama esquemático e a tabela verdade do registrador de deslocamento 74LS194 são mostrados na figura 11.16 e na tabela 11.5, respectivamente. Quando MR = 0, independentemente de todas as entradas, o registrador é limpo e, portanto, todas as saídas são forçadas a 0. Quando a entrada de "reset" mestre não está ativa, ou seja, quando MR = 1, entradas S_1 e S_0 são usadas para selecionar o modo de operação. Como pode ser visto na tabela 11.5, existem 4 tipos de modos: armazenamento ("hold"), deslocamento para esquerda ("shift left"), deslocamento para direita ("shift right") e carregamento paralelo ("parallel load"). Quando as entradas S_1 e S_0 são 00, as saídas (Q0, Q1, Q2 e Q3) mantêm seus valores anteriores ("hold").

Quando as entradas S_1 e S_0 são 10, os dados armazenados no registrador são deslocados para a esquerda com a borda de subida do sinal do relógio ("shift left"). Neste caso, a entrada serial é obtida da entrada DSL.

Quando as entradas S_1 e S_0 são 01, os dados armazenados no registrador são deslocados para a direita com a borda de subida do sinal do relógio ("shift right"). Neste caso, a entrada serial é obtida da entrada DSR.

Quando as entradas S_1 e S_0 são 11, os dados nas entradas paralelas P0, P1, P2 e P3 são carregados no registrador (ou seja, nos flip-flops Q0, Q1, Q2 e Q3, respectivamente) com a borda de subida do sinal de relógio ("parallel load").

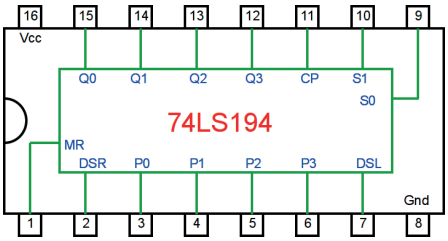


Figura 11.15 – Definições de pinos do registrador de deslocamento 74LS194.

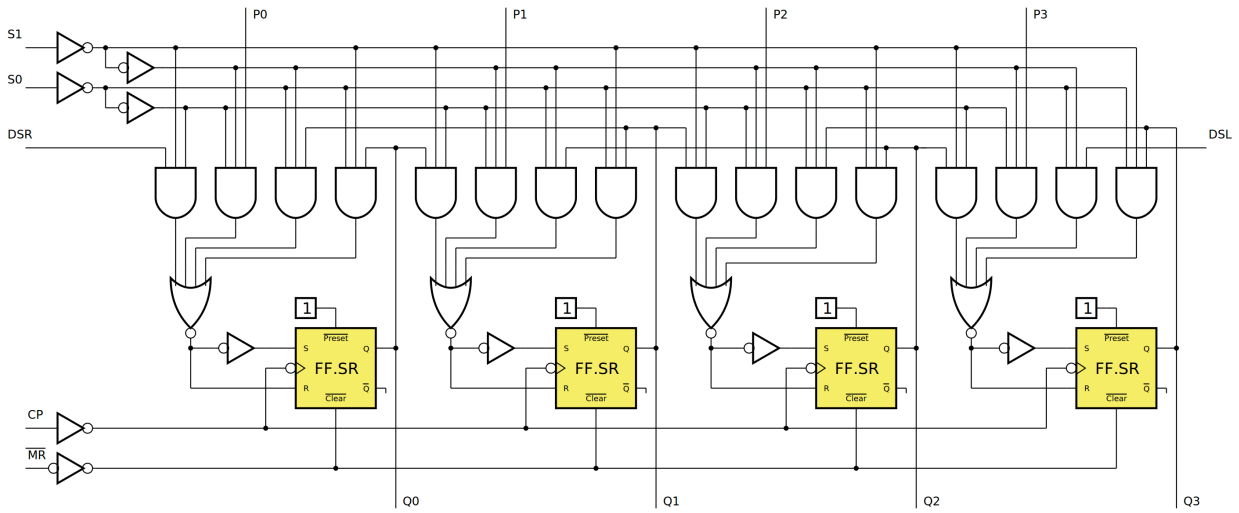


Figura 11.16 – Diagrama esquemático do registrador de deslocamento 74LS194.

Modo de operação	Entradas						Saídas			
	MR	S1	S0	DSR	DSL	Pn	Q0	Q1	Q2	Q3
Limpar (reset)	0	x	x	x	x	x	0	0	0	0
Armazenar	1	0	0	x	x	x	q0	q1	q2	q3
Deslocar para esquerda	1	1	0	x	0	x	q1	q2	q3	0
	1	1	0	x	1	x	q1	q2	q3	1
Deslocar para direita	1	0	1	0	x	x	0	q0	q1	q2
	1	0	1	1	x	x	1	q0	q1	q2
Carga paralela	1	1	1	x	x	Pn	P0	P1	P2	P3

Tabela 11.5 – Tabela verdade do registrador de deslocamento 74LS194.

11.4.3. Exame do CI 74LS194, registrador universal de 4 bits

Esquemas:

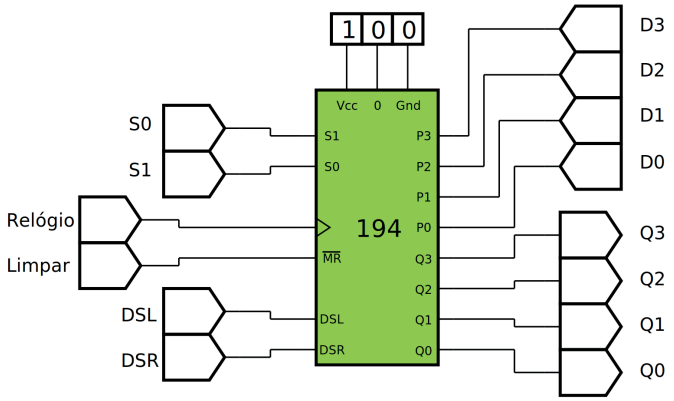


Figura 11.17 – Diagrama esquemático.

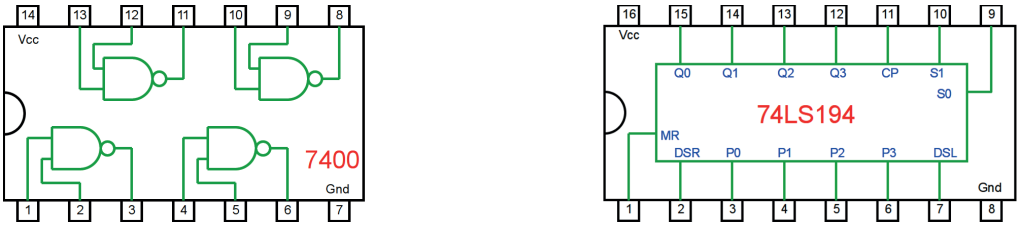


Figura 11.18 – Circuitos integrados TTL.

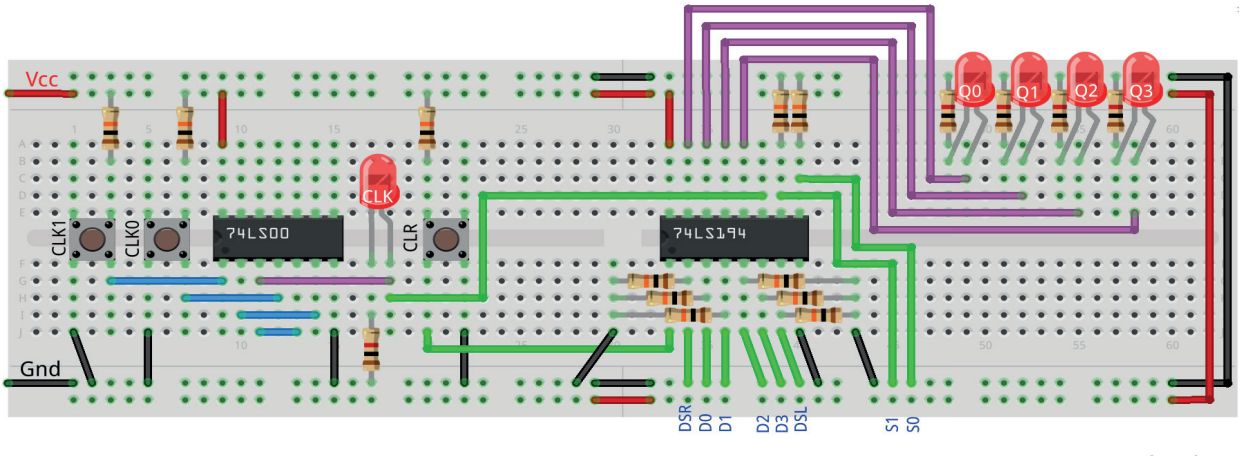


Figura 11.19 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 11.19.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 11.6, use os botões como segue:
   a) O botão CLR pressionado envia um ativo baixo “0” e apaga todos os leds (Limpar).
   b) Para configurar os dados de entrada, conecte os fios DSR, DSL, S0, S1, D0, D1, D2 e D3 na barra Gnd para “0” e na barra Vcc para “1”.
   c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   e) Um pulso de relógio completo é conseguido pressionando primeiro o botão CLK1 e depois pressionando o botão CLK0 (Relógio).
3. Para o preenchimento das colunas de saída na tabela 11.6, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 11.17. Compare os resultados com a tabela 11.6.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 11.6.

Tabelas de dados:

Entradas										Saída			
CLK	MR	D0	D1	D2	D3	S1	S0	DSR	DSL	Q0	Q1	Q2	Q3
0	0	x	x	x	x	0	1	1	x				
1	1	x	x	x	x	0	1	1	x				
2	1	x	x	x	x	0	1	1	x				
3	1	x	x	x	x	0	1	1	x				
4	1	x	x	x	x	0	1	1	x				

Tabela 11.6a

Entradas										Saída			
CLK	MR	D0	D1	D2	D3	S1	S0	DSR	DSL	Q0	Q1	Q2	Q3
0	0	x	x	x	x	1	0	x	1				
1	1	x	x	x	x	1	0	x	1				
2	1	x	x	x	x	1	0	x	1				
3	1	x	x	x	x	1	0	x	1				
4	1	x	x	x	x	1	0	x	1				

Tabela 11.6b

Entradas										Saída			
CLK	MR	D0	D1	D2	D3	S1	S0	DSR	DSL	Q0	Q1	Q2	Q3
1	1	1	1	0	0	1	1	x	x				
2	1	1	1	0	0	1	1	x	x				
3	1	0	1	1	0	1	1	x	x				
4	1	0	1	1	0	1	1	x	x				
5	1	1	1	1	1	1	1	x	x				
6	0	1	1	1	1	1	1	x	x				

Tabela 11.6c

Programa para o Arduíno:

Monte o circuito na placa de montagem, conforme figura 11.20. Conecte os fios azuis e roxos aos respectivos pinos do Arduíno. Conecte os fios vermelho (Vcc) e preto (Gnd) nos terminais apropriados do Arduíno.

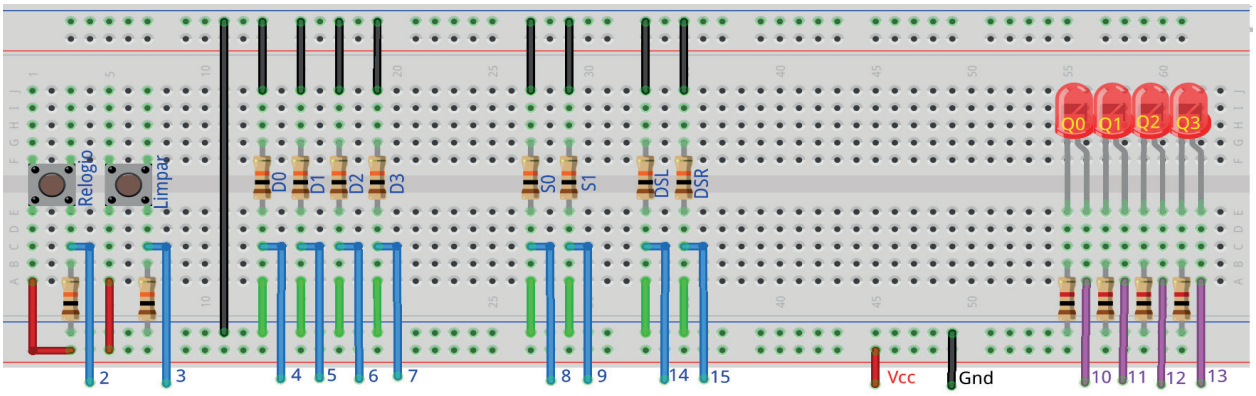


Figura 11.20 – Disposição dos componentes na placa de montagem.

Uso:

- O Botão “Limpar” apaga todos os LEDs, independente do pulso de relógio.
- Os dados de entrada (“D0, D1, D2, D3, S0, S1, DSL e DSR”) devem ser configurados como: se for “0” o fio verde deve ser conectado na barra “Gnd”; se for “1” o fio verde deve ser conectado na barra “Vcc”. Esta operação deve ser feita antes do pulso de relógio.
- O botão “Relógio” executa as operações de acordo com a tabela 11.5.

```
// Emulador do 74LS194 - registrador de deslocamento universal de 4 bits
// entrada paralela ou serial e saída paralela

// pinos de entrada
  #define pino_relogio 2
  #define pino_limpar 3
  #define pino_dado0 4
  #define pino_dado1 5
  #define pino_dado2 6
  #define pino_dado3 7
  #define pino_conf0 8
  #define pino_conf1 9
  #define pino_dado_esq 14 // (ou A0 - Uno, Nano)
  #define pino_dado_dir 15 // (ou A1 - Uno, Nano)

// pinos de saída
  #define pino_ledq0 10
  #define pino_ledq1 11
  #define pino_ledq2 12
  #define pino_ledq3 13

// valores do 74LS194
  byte p0; byte p1; byte p2; byte p3;
  byte q0; byte q1; byte q2; byte q3;
  byte dsl; byte dsr;
  byte cp; byte cp_old; byte mr; byte s0; byte s1;

// valores de entrada
  byte dado0; byte dado1; byte dado2; byte dado3;
  byte conf0; byte conf1;
  byte dado_esq; byte dado_dir;
  byte relogio; byte limpar;
  byte relogio_antes; byte limpar_antes;

void ci_74LS194(){
```

```
  if (cp && !cp_old){
    // 11 = carga paralela
    if(s0 && s1){
      q0 = p0; q1 = p1; q2 = p2; q3 = p3;}
    // 10 = deslocar para direita
    if(s0 && !s1){
      q3 = q2; q2 = q1; q1 = q0; q0 = dsr;}
    // 01 = deslocar para esquerda
    if(!s0 && s1){
      q0 = q1; q1 = q2; q2 = q3; q3 = dsl;}
    // 00 = armazenar -> nenhuma ação!
    cp_old = cp;}
  if (!mr){q0 = 0; q1 = 0; q2 = 0; q3 = 0;}
} // fim do 'ci_74LS194'

void setup(){

// inicialização dos pinos de entrada
  pinMode(pino_relogio, INPUT);
  pinMode(pino_limpar, INPUT);
  pinMode(pino_dado0, INPUT);
  pinMode(pino_dado1, INPUT);
  pinMode(pino_dado2, INPUT);
  pinMode(pino_dado3, INPUT);
  pinMode(pino_conf0, INPUT);
  pinMode(pino_conf1, INPUT);
  pinMode(pino_dado_esq, INPUT);
  pinMode(pino_dado_dir, INPUT);

// inicialização dos pinos de saída
  pinMode(pino_ledq0, OUTPUT);
  pinMode(pino_ledq1, OUTPUT);
  pinMode(pino_ledq2, OUTPUT);
  pinMode(pino_ledq3, OUTPUT);

// inicialização das variáveis
  p0 = 0; p1 = 0; p2 = 0; p3 = 0;
  q0 = 0; q1 = 0; q2 = 0; q3 = 0;
  relogio_antes = 0;
  limpar_antes = 0;
  cp_old = 0;

} // fim do 'setup'
void loop(){
```

```
// leitura das entradas
  relogio = digitalRead(pino_relogio);
  if (relogio != relogio_antes) delay(100);
  limpar = digitalRead(pino_limpar);
  if (limpar != limpar_antes) delay(100);
  dado0 = digitalRead(pino_dado0);
  dado1 = digitalRead(pino_dado1);
  dado2 = digitalRead(pino_dado2);
  dado3 = digitalRead(pino_dado3);
  conf0 = digitalRead(pino_conf0);
  conf1 = digitalRead(pino_conf1);
  dado_esq = digitalRead(pino_dado_esq);
  dado_dir = digitalRead(pino_dado_dir);

// configuração para registrador de deslocamento universal
  cp = relogio;
  mr = !limpar;
  p0 = dado0;
  p1 = dado1;
  p2 = dado2;
  p3 = dado3;
  s0 = conf0;
  s1 = conf1;
  dsl = dado_esq;
  dsr = dado_dir;

// aplica valores ao CI virtual
  ci_74LS194();

// mostra saídas
  digitalWrite(pino_ledq0,q0);
  digitalWrite(pino_ledq1,q1);
  digitalWrite(pino_ledq2,q2);
  digitalWrite(pino_ledq3,q3);

// reinicia variáveis
  relogio_antes = relogio;
  limpar_antes = limpar;
} // fim do ‘loop’
```

11.5. Circuito Integrado TTL 74LS195 – Registrador universal de deslocamento

11.5.1. Objetivos

1. Aprender os diferentes princípios de operação de registradores de deslocamento.
2. Conhecer o CI do registrador de deslocamento 74LS195.
3. Observar seu funcionamento e obter sua tabela verdade.

11.5.2. Informação preliminar

O 74LS195 é outro registrador de deslocamento universal de CI único. A figura 11.21 mostra as definições de pinos do 74LS195. O diagrama esquemático e a tabela verdade do registrador de deslocamento 74LS195 são mostrados na figura 11.22 e na tabela 11.7, respectivamente.

O diagrama esquemático e a tabela verdade indicam as características funcionais do registrador de deslocamento LS195A de 4 bits. O dispositivo é útil em uma ampla variedade de aplicativos de deslocamento, contagem e armazenamento. Ele executa transferências de dados seriais, paralelas, seriais para paralelas ou paralelas para seriais em velocidades muito altas. O LS195A tem dois modos primários de operação, deslocamento para a direita (Q0 → Q1 → Q2 → Q3) e carga paralela que são controlados pelo estado da entrada de "Parallel Enable" (PE') ativa em baixo (0). Quando a entrada PE' é alto (1), os dados seriais entram no primeiro flip-flop Q0 através das entradas J e K' e são deslocados um bit na direção Q0 → Q1 → Q2 → Q3 seguindo cada transição de pulso de relógio de baixo (0) para alto (1). As entradas de JK' fornecem a flexibilidade da entrada do tipo JK para aplicações especiais e a entrada do tipo D simples para aplicações gerais, unindo os dois pinos. Quando a entrada PE' é baixo (0), o LS195A aparece como quatro flip-flops D com relógio comum. Os dados das entradas paralelas P0, P1, P2, P3 são transferidos para as respectivas saídas Q0, Q1, Q2, Q3, seguindo a transição de relógio de baixo (0) para alto (1). Operações de deslocamento à esquerda (Q0 ← Q1 ← Q2 ← Q3) podem ser alcançadas ligando as saídas Q_n às entradas P_{n-1} e mantendo a entrada PE' em baixo (0). Todas as transferências de dados seriais e paralelas são síncronas, ocorrendo após cada transição de pulso de relógio de baixo (0) para alto (1). Como o LS195A utiliza gatilho de borda, não há restrição na atividade das entradas J, K', P_n e PE' para operação lógica - exceto pelos requisitos de configuração e tempo de liberação. Um sinal de ativo baixo (0) na entrada assíncrona de "Master Reset" (MR') torna todas as saídas Q em baixo (0), independentemente de qualquer outra condição de entrada.

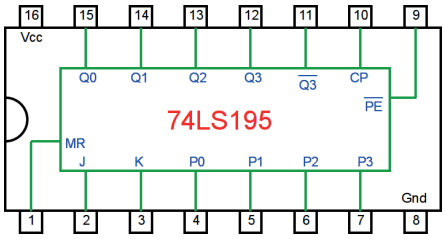


Figura 11.21 – Definições de pinos do registrador de deslocamento 74LS195.

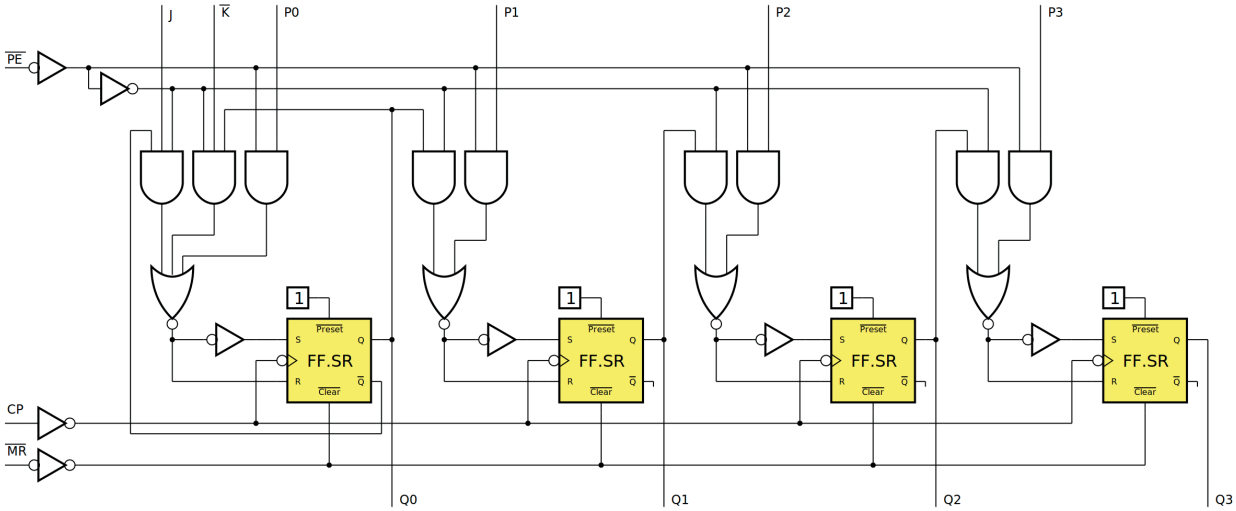


Figura 11.22 – Diagrama esquemático do registrador de deslocamento 74LS195.

Modo de operação	Entradas					Saídas				
	MR	PE’	J	K’	Pn	Q0	Q1	Q2	Q3	Q3’
Limpar (reset)	0	x	x	x	x	0	0	0	0	1
Deslocar, marcar	1	1	1	1	x	1	q0	q1	q2	q2’
Deslocar, limpar	1	1	0	0	x	0	q0	q1	q2	q2’
Deslocar, troca 1º estágio	1	1	1	0	x	q0’	q0	q1	q2	q2’
Deslocar, mantém 1º estágio	1	1	0	1	x	q0	q0	q1	q2	q2’
Carga paralela	1	0	x	x	Pn	P0	P1	P2	P3	P3’

Tabela 11.7 – Tabela verdade do registrador de deslocamento 74LS195.

11.5.3. Exame do CI 74LS195, registrador universal de 4 bits

Esquemas:

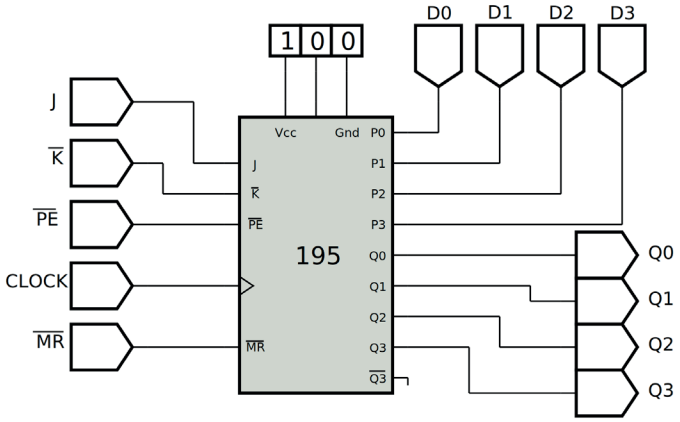


Figura 11.23 – Diagrama esquemático.

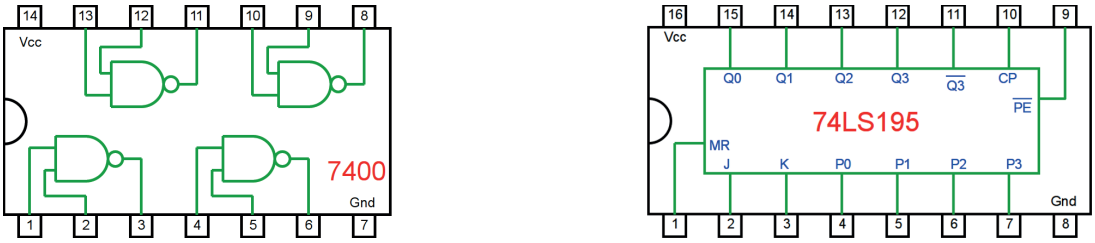


Figura 11.24 – Circuitos integrados TTL.

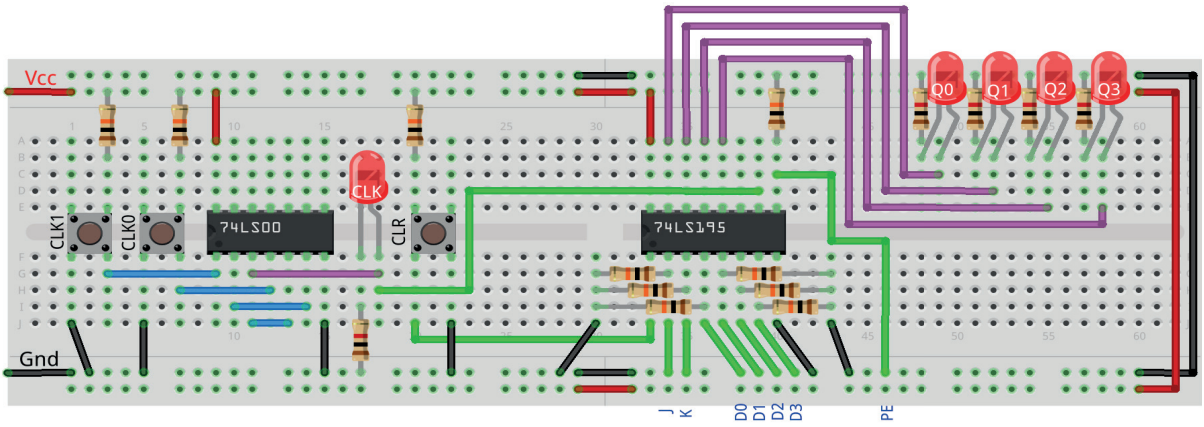


Figura 11.25 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 11.25.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 11.8, use os botões como segue:
   a) O botão CLR pressionado envia um ativo baixo “0” e apaga todos os leds (MR).
   b) Para configurar os dados de entrada, conecte os fios J, K, PE, D0, D1, D2 e D3 na barra Gnd para “0” e na barra Vcc para “1”.
   c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   e) Um pulso de relógio completo é conseguido pressionando primeiro o botão CLK1 e depois pressionando o botão CLK0 (Relógio).
3. Para o preenchimento das colunas de saída na tabela 11.8, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 11.23. Compare os resultados com a tabela 11.8.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 11.8.

Tabelas de dados:

Entradas									Saída			
CLK	MR	PE’	J	K’	D0	D1	D2	D3	Q0	Q1	Q2	Q3
0	0	X	X	X	X	X	X	X				
1	1	1	1	1	X	X	X	X				
2	1	1	1	1	X	X	X	X				
3	1	1	1	1	X	X	X	X				
4	1	1	0	0	X	X	X	X				
5	1	1	0	0	X	X	X	X				
6	1	1	0	0	X	X	X	X				
7	1	1	0	0	X	X	X	X				
8	1	1	1	0	X	X	X	X				
9	1	1	1	0	X	X	X	X				
10	1	1	1	0	X	X	X	X				
11	1	1	1	0	X	X	X	X				
12	1	1	0	1	X	X	X	X				
13	1	1	0	1	X	X	X	X				
14	1	1	0	1	X	X	X	X				
15	1	1	0	1	X	X	X	X				
16	0	X	X	X	X	X	X	X				
17	1	0	X	X	1	0	1	0				
18	1	0	X	X	0	1	0	1				
19	1	0	X	X	1	0	1	1				
20	1	0	X	X	0	1	1	0				
21	1	0	X	X	1	1	1	1				
22	1	0	X	X	1	1	1	0				

Tabela 11.8

Programa para o Arduíno:

Monte o circuito na placa de montagem, conforme figura 11.26. Conecte os fios azuis e roxos aos respectivos pinos do Arduíno. Conecte os fios vermelho (Vcc) e preto (Gnd) nos terminais apropriados do Arduíno.

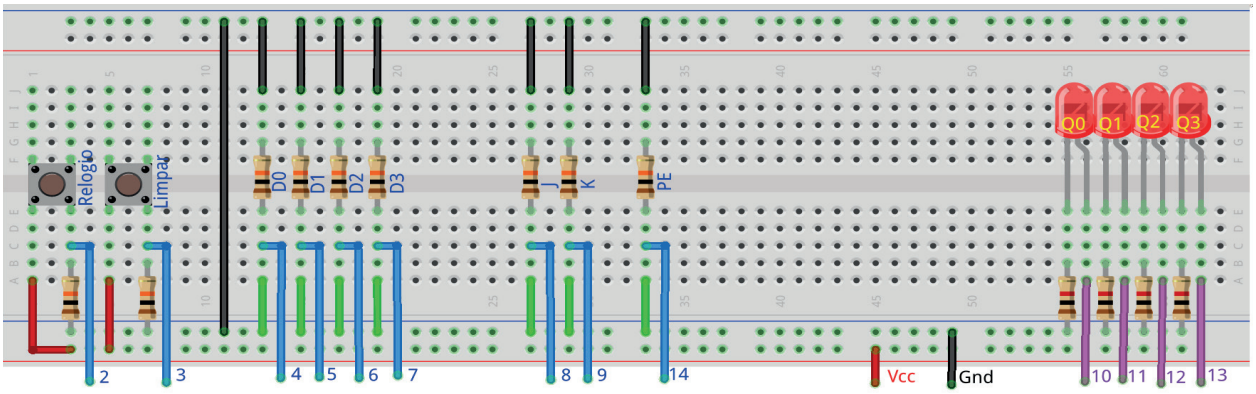


Figura 11.26 – Disposição dos componentes na placa de montagem.

Uso:

- O Botão "Limpar" apaga todos os LEDs, independente do pulso de relógio.
- Os dados de entrada ("D0, D1, D2, D3, J, K e PE") devem ser configurados como: se for "0" o fio verde deve ser conectado na barra "Gnd"; se for "1" o fio verde deve ser conectado na barra "Vcc". Esta operação deve ser feita antes do pulso de relógio.
- O botão "Relógio" executa as operações de acordo com a tabela 11.7.

```
// Emulador do 74LS195 - registrador de deslocamento universal de 4 bits
// entrada paralela ou serial e saída paralela

// pinos de entrada
  #define pino_relogio 2
  #define pino_limpar 3
  #define pino_dado0 4
  #define pino_dado1 5
  #define pino_dado2 6
  #define pino_dado3 7
  #define pino_confj 8
  #define pino_confk 9
  #define pino_pe 14 // (ou A0 - Uno, Nano)

// pinos de saída
  #define pino_ledq0 10
  #define pino_ledq1 11
  #define pino_ledq2 12
```

```
  #define pino_ledq3 13

// valores do 74LS195
  byte p0; byte p1; byte p2; byte p3;
  byte q0; byte q1; byte q2; byte q3;
  byte j; byte k;
  byte cp; byte cp_old; byte mr; byte pe;

// valores de entrada
  byte dado0; byte dado1; byte dado2; byte dado3;
  byte confj; byte confk; byte carga_par;
  byte relogio; byte limpar;
  byte relogio_antes; byte limpar_antes;

void ci_74LS195(){
  if (mr && cp && !cp_old){
    // deslocar para direita e marcar
    if(pe && j && k){
      q3 = q2; q2 = q1; q1 = q0; q0 = 1;}
    // deslocar para direita e limpar
    if(pe && !j && !k){
      q3 = q2; q2 = q1; q1 = q0; q0 = 0;}
    // deslocar para direita e inverte q0
    if(pe && j && !k){
      q3 = q2; q2 = q1; q1 = q0; q0 = !q0;}
    // deslocar para direita e copia q0
    if(pe && j && !k){
      q3 = q2; q2 = q1; q1 = q0;}
    // carga paralela
    if(!pe){
      q0 = p0; q1 = p1; q2 = p2; q3 = p3;}
    cp_old = cp;}
  if (!mr){q0 = 0; q1 = 0; q2 = 0; q3 = 0;}
} // fim do ‘ci_74LS195’

void setup(){

// inicialização dos pinos de entrada
  pinMode(pino_relogio, INPUT);
  pinMode(pino_limpar, INPUT);
  pinMode(pino_dado0, INPUT);
```

```
  pinMode(pino_dado1, INPUT);
  pinMode(pino_dado2, INPUT);
  pinMode(pino_dado3, INPUT);
  pinMode(pino_confj, INPUT);
  pinMode(pino_confk, INPUT);
  pinMode(pino_pe, INPUT);

// inicialização dos pinos de saída
  pinMode(pino_ledq0, OUTPUT);
  pinMode(pino_ledq1, OUTPUT);
  pinMode(pino_ledq2, OUTPUT);
  pinMode(pino_ledq3, OUTPUT);

// inicialização das variáveis
  p0 = 0; p1 = 0; p2 = 0; p3 = 0;
  q0 = 0; q1 = 0; q2 = 0; q3 = 0;
  relogio_antes = 0;
  limpar_antes = 0;
  cp_old = 0;

} // fim do ‘setup’

void loop(){

// leitura das entradas
  relogio = digitalRead(pino_relogio);
  if (relogio != relogio_antes) delay(100);
  limpar = digitalRead(pino_limpar);
  if (limpar != limpar_antes) delay(100);
  dado0 = digitalRead(pino_dado0);
  dado1 = digitalRead(pino_dado1);
  dado2 = digitalRead(pino_dado2);
  dado3 = digitalRead(pino_dado3);
  confj = digitalRead(pino_confj);
  confk = digitalRead(pino_confk);
  carga_par = digitalRead(pino_pe);

// configuração para registrador de deslocamento universal
  cp = relogio;
  mr = !limpar;
  p0 = dado0;
```

```
  p1 = dado1;
  p2 = dado2;
  p3 = dado3;
  j = confj;
  k = confk;
  pe = carga_par;

// aplica valores ao CI virtual
  ci_74LS195();

// mostra saídas
  digitalWrite(pino_ledq0,q0);
  digitalWrite(pino_ledq1,q1);
  digitalWrite(pino_ledq2,q2);
  digitalWrite(pino_ledq3,q3);

// reinicia variáveis
  relogio_antes = relogio;
  limpar_antes = limpar;

} // fim do ‘loop’
```

11.6. Circuito Integrado TTL 74LS165 – Registrador de deslocamento de 8 bits

11.6.1. Objetivos

1. Aprender diferentes princípios de operação de registradores de deslocamento.
2. Conhecer o CI registrador de deslocamento 74LS165.
3. Observar seu funcionamento e obter sua tabela verdade.

11.6.2. Informação preliminar

A figura 11.27 mostra as definições de pinos do 74LS165. O diagrama esquemático e a tabela verdade do registrador de deslocamento 74LS165 são mostrados na figura 11.28 e na tabela 11.9 respectivamente. O 74LS165 é um registrador de 8 bits de carga paralela ou entrada serial com saídas em série complementares (Q7' e Q7') disponíveis no último estágio. Quando a entrada de carga paralela (PL') está em baixo (0), os dados paralelos das entradas D0 a D7 são carregados no registrador de forma assíncrona. Quando PL' é alto (1), os dados entram no registrador em série na entrada Ds e mudam um lugar para a direita (Q0 → Q1 → Q2, etc.) com cada transição do pulso de relógio de baixo (0) para alto (1). Esse recurso permite a expansão do conversor de paralelo para serial vinculando a saída Q7 à entrada Ds do próximo estágio. A entrada de relógio é uma estrutura de porta OU que permite que uma entrada seja usada como uma entrada de habilitação de relógio (CE') ativa em baixo (0). As atribuições de pinos para as entradas CP' e CE' são arbitrárias e podem ser revertidas para conveniência de layout. A transição de baixo-para-alto da entrada CE' deve ocorrer somente enquanto CP for alto (1) para operação previsível. Tanto CP quanto CE' devem estar em alto (1) antes da transição de baixo-para-alto de PL' para evitar o deslocamento dos dados quando o PL' estiver ativado.

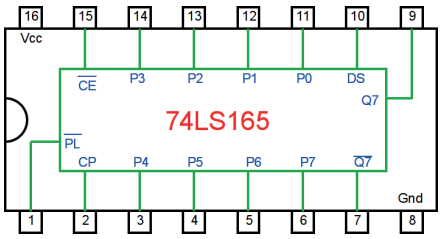


Figura 11.27 – Definições de pinos do registrador de deslocamento 74LS165.

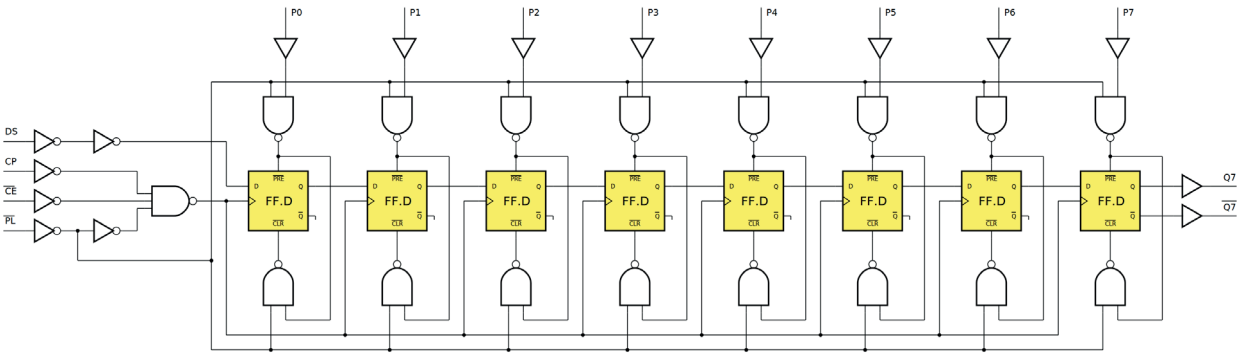


Figura 11.28 – Diagrama esquemático do registrador de deslocamento 74LS165.

Modo de operação	Entradas					Registros		Saída	
	PL’	CE’	CP	DS	P0-P7	Q0	Q1-Q6	Q7	Q7’
Carga paralela	0	x	x	x	0	0	0 - 0	0	1
	0	x	x	x	1	1	1 - 1	1	0
Deslocamento serial	1	0	↑	0	x	0	q0 - q5	q6	q6’
	1	0	↑	1	x	1	q0 - q5	q6	q6’
Armazenamento	1	1	x	x	x	q0	q1 - q6	q7	q7’

Tabela 11.9 – Tabela verdade do registrador de deslocamento 74LS165.

11.6.3. Exame do CI 74LS165, registrador de deslocamento de 8 bits

Esquemas:

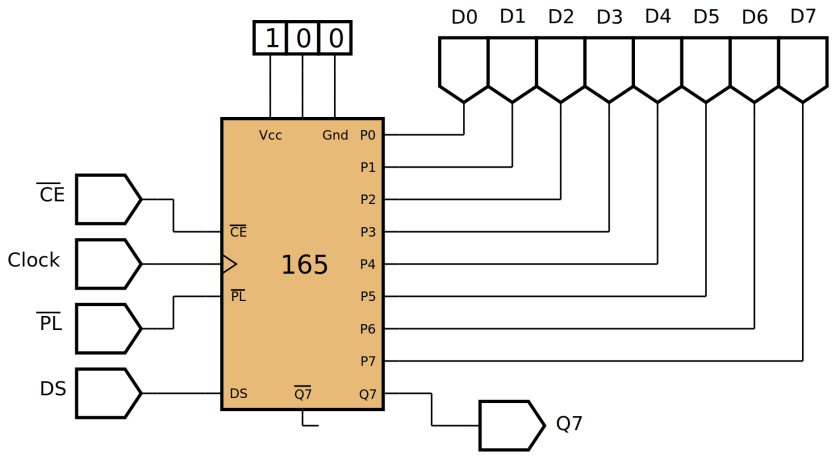


Figura 11.29 – Diagrama esquemático.

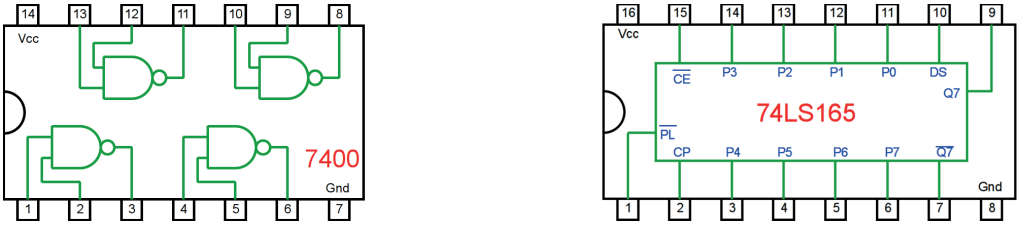


Figura 11.30 – Circuitos integrados TTL.

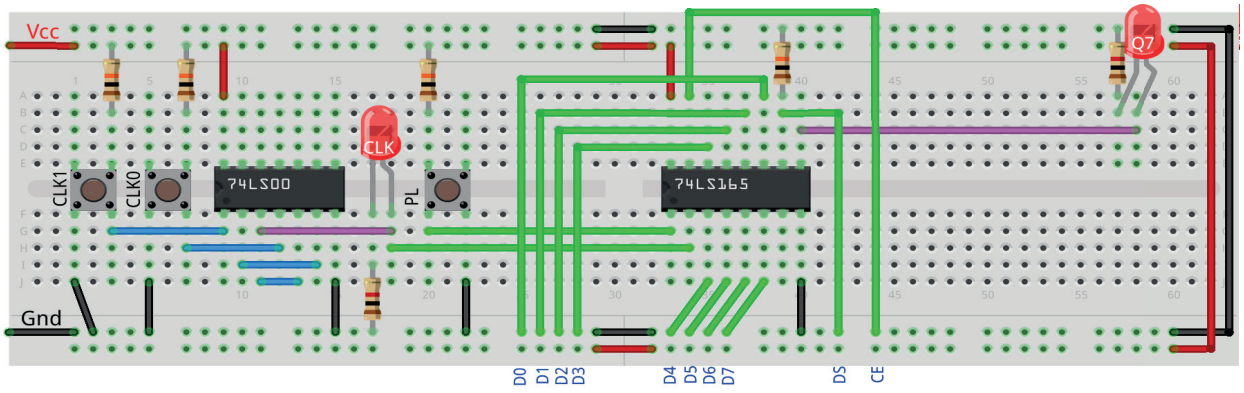


Figura 11.31 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 11.31.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 11.10, use os botões como segue:
   a) O botão PL pressionado envia um ativo baixo “0” e faz o carregamento paralelo do registrador.
   b) Para configurar os dados de entrada, conecte os fios CE, DS, D0, D1, D2, D3, D4, D5, D6 e D7 na barra Gnd para “0” e na barra Vcc para “1”.
   c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   e) Um pulso de relógio completo é conseguido pressionando primeiro o botão CLK1 e depois pressionando o botão CLK0 (Relógio).
3. Para o preenchimento das colunas de saída na tabela 11.10, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 11.29. Compare os resultados com a tabela 11.10.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 11.10.

Tabelas de dados:

Entradas												Saída		
CLK	PL’	CE’	DS	P0	P1	P2	P3	P4	P5	P6	P7	Q0.Q1.Q2.Q3.Q4.Q5.Q6.Q7	Q7	Q7’
0	0	X	X	1	0	1	0	1	0	1	1			
1	1	1	X	X	X	X	X	X	X	X	X			
2	1	1	X	X	X	X	X	X	X	X	X			
3	1	1	X	X	X	X	X	X	X	X	X			
4	1	1	X	X	X	X	X	X	X	X	X			
5	1	0	1	X	X	X	X	X	X	X	X			
6	1	0	1	X	X	X	X	X	X	X	X			
7	1	0	1	X	X	X	X	X	X	X	X			
8	1	0	1	X	X	X	X	X	X	X	X			
9	1	0	1	X	X	X	X	X	X	X	X			
10	1	0	1	X	X	X	X	X	X	X	X			
11	1	0	1	X	X	X	X	X	X	X	X			
12	1	0	1	X	X	X	X	X	X	X	X			
13	1	0	1	X	X	X	X	X	X	X	X			
14	0	X	X	1	0	1	1	1	1	0	0			
15	1	0	0	X	X	X	X	X	X	X	X			
16	1	0	0	X	X	X	X	X	X	X	X			
17	1	0	0	X	X	X	X	X	X	X	X			
18	1	0	0	X	X	X	X	X	X	X	X			
19	1	0	0	X	X	X	X	X	X	X	X			
20	1	0	0	X	X	X	X	X	X	X	X			
21	1	0	0	X	X	X	X	X	X	X	X			
22	1	0	0	X	X	X	X	X	X	X	X			

Tabela 11.10

Programa para o Arduíno:

Monte o circuito na placa de montagem, conforme figura 11.32. Conecte os fios azuis e roxos aos respectivos pinos do Arduíno. Conecte os fios vermelho (Vcc) e preto (Gnd) nos terminais apropriados do Arduíno.

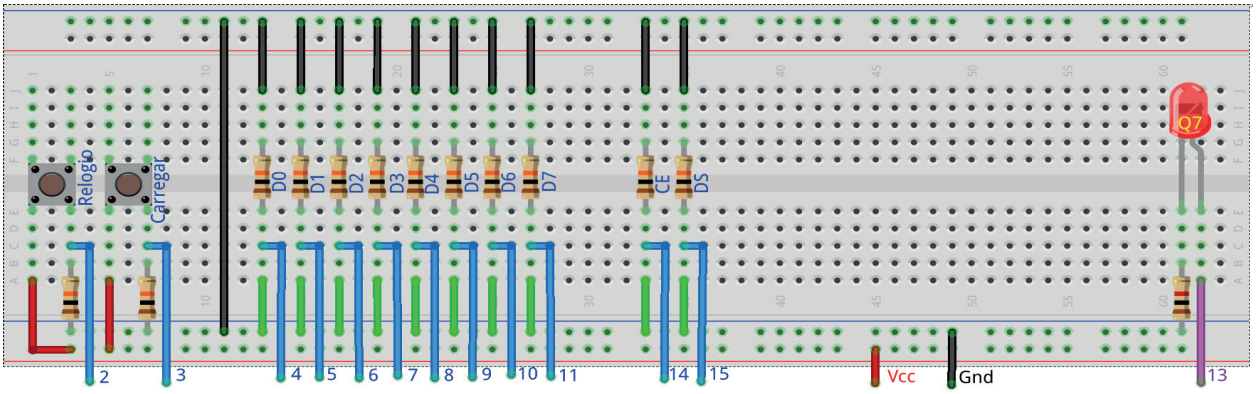


Figura 11.32 – Disposição dos componentes na placa de montagem.

Uso:

- O Botão “Carregar” faz o carregamento paralelo do registrador.
- Os dados de entrada (“D0, D1, D2, D3, D4, D5, D6, D7, CE e DS ”) devem ser configurados como: se for “0” o fio verde deve ser conectado na barra “Gnd”; se for “1” o fio verde deve ser conectado na barra “Vcc”. Esta operação deve ser feita antes do pulso de relógio.
- O botão “Relógio” executa as operações de acordo com a tabela 11.9.

```
// Emulador do 74LS165 - registrador de deslocamento universal de 8 bits
// entrada paralela e saída serial

// pinos de entrada
  #define pino_relogio 2
  #define pino_carregar 3
  #define pino_dado0 4
  #define pino_dado1 5
  #define pino_dado2 6
  #define pino_dado3 7
  #define pino_dado4 8
  #define pino_dado5 9
  #define pino_dado6 10
  #define pino_dado7 11
  #define pino_ce 14 // (ou A0 - Uno, Nano)
  #define pino_ds 15 // (ou A1 - Uno, Nano)
```

```
// pinos de saída
  #define pino_ledq7 13

// valores do 74LS165
  byte p0; byte p1; byte p2; byte p3; byte p4;
  byte p5; byte p6; byte p7;
  byte q0; byte q1; byte q2; byte q3; byte q4;
  byte q5; byte q6; byte q7;
  byte ds; byte ce; byte pl; byte cp; byte cp_old;

// valores de entrada
  byte dado0; byte dado1; byte dado2; byte dado3;
  byte dado4; byte dado5; byte dado6; byte dado7;
  byte dados; byte ativar;
  byte relogio; byte carregar;
  byte relogio_antes; byte carregar_antes;

void ci_74LS165(){
  // carga paralela
  if (!pl){
    q0 = p0; q1 = p1; q2 = p2; q3 = p3; q4 = p4; q5 = p5; q6 = p6; q7 =
    p7;}
  // deslocar para direita
  if (pl && !ce && cp && !cp_old){
    q7 = q6; q6 = q5; q5 = q4; q4 = q3; q3 = q2; q2 = q1; q1 = q0; q0 = ds;
    cp_old = cp;}
} // fim do ‘ci_74LS165’

void setup(){

// inicialização dos pinos de entrada
  pinMode(pino_relogio, INPUT);
  pinMode(pino_carregar, INPUT);
  pinMode(pino_dado0, INPUT);
  pinMode(pino_dado1, INPUT);
  pinMode(pino_dado2, INPUT);
  pinMode(pino_dado3, INPUT);
  pinMode(pino_dado4, INPUT);
  pinMode(pino_dado5, INPUT);
  pinMode(pino_dado6, INPUT);
  pinMode(pino_dado7, INPUT);
  pinMode(pino_ce, INPUT);
```

```
  pinMode(pino_ds, INPUT);


// inicialização dos pinos de saída
  pinMode(pino_ledq7, OUTPUT);

// inicialização das variáveis
  p0 = 0; p1 = 0; p2 = 0; p3 = 0; p4 = 0; p5 = 0; p6 = 0; p7 = 0;
  q0 = 0; q1 = 0; q2 = 0; q3 = 0; q4 = 0; q5 = 0; q6 = 0; q7 = 0;
  relogio_antes = 0;
  carregar_antes = 0;
  cp_old = 0;

} // fim do 'setup'

void loop(){

// leitura das entradas
  relogio = digitalRead(pino_relogio);
  if (relogio != relogio_antes) delay(100);
  carregar = digitalRead(pino_carregar);
  if (carregar != carregar_antes) delay(100);
  dado0 = digitalRead(pino_dado0);
  dado1 = digitalRead(pino_dado1);
  dado2 = digitalRead(pino_dado2);
  dado3 = digitalRead(pino_dado3);
  dado4 = digitalRead(pino_dado4);
  dado5 = digitalRead(pino_dado5);
  dado6 = digitalRead(pino_dado6);
  dado7 = digitalRead(pino_dado7);
  dados = digitalRead(pino_ds);
  ativar = digitalRead(pino_ce);

// configuração para registrador de deslocamento universal
  cp = relogio;
  pl = !carregar;
  p0 = dado0;
  p1 = dado1;
  p2 = dado2;
  p3 = dado3;
  p4 = dado4;
```

```
  p5 = dado5;
  p6 = dado6;
  p7 = dado7;
  ce = !ativar;
  ds = dados;

// aplica valores ao CI virtual
  ci_74LS165();
// mostra saídas
  digitalWrite(pino_ledq7,q7);

// reinicia variáveis
  relogio_antes = relogio;
  carregar_antes = carregar;

} // fim do ‘loop’
```

11.7. Circuito Integrado TTL 74LS164 – Registrador de deslocamento de 8 bits

11.7.1. Objetivos

1. Aprender diferentes princípios de operação de registradores de deslocamento.
2. Conhecer o CI registrador de deslocamento 74LS164.
3. Observar seu funcionamento e obter sua tabela verdade.

11.7.2. Informação preliminar

A figura 11.33 mostra as definições de pinos do 74LS164. O diagrama esquemático e a tabela verdade do registrador de deslocamento 74LS164 são mostrados na figura 11.34 e na tabela 11.11, respectivamente.

O 74LS164 é um registrador de deslocamento de 8 bits acionado por borda com entrada de dados serial e uma saída de cada um dos oito estágios. Os dados são inseridos serialmente através de uma das duas entradas (A ou B); qualquer uma dessas entradas pode ser usada como habilitação ativa em alto (1) com a entrada de dados através da outra entrada. Uma entrada não utilizada deve estar conectada em alto (1) ou ambas as entradas conectadas juntas. Cada transição de baixo-para-alto na entrada de relógio (CP) desloca os dados um local para a direita e a entrada através de Q0 a lógica E das duas entradas de dados (A • B) que existiam antes da borda crescente de relógio. Um nível baixo (0) na entrada "Master Reset" (MR') substitui todas as outras entradas e limpa o registrador de forma assíncrona, forçando todas as saídas Q em baixo (0).

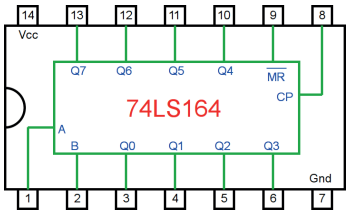


Figura 11.33 – Definições de pinos do registrador de deslocamento 74LS164.

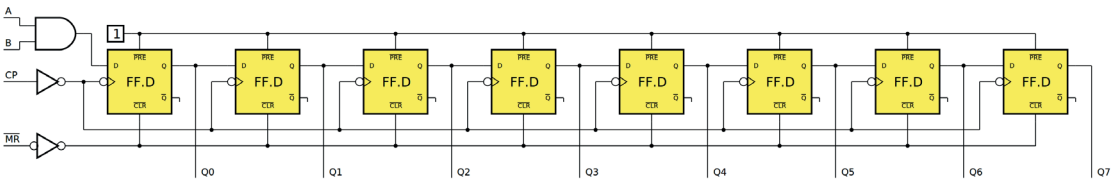


Figura 11.34 – Diagrama esquemático do registrador de deslocamento 74LS164.

Modo de operação	Entradas			Saídas	
	MR	A	B	Q0	Q1 - Q7
Apagar	0	x	x	0	0 - 0
Deslocar	1	0	0	0	q0 - q6
	1	0	1	0	q0 - q6
	1	1	0	0	q0 - q6
	1	1	1	1	q0 - q6

Tabela 11.11 – Tabela verdade do registrador de deslocamento 74LS164.

11.7.3. Exame do CI 74LS164, registrador de deslocamento de 8 bits

Esquemas:

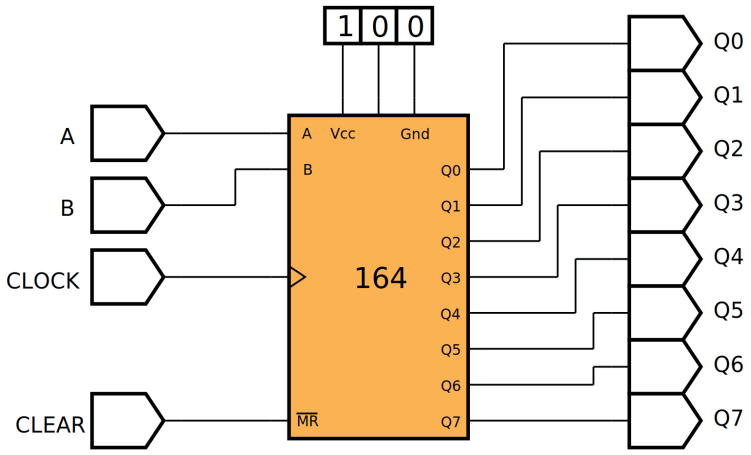


Figura 11.35 – Diagrama esquemático.

Figura 11.36 – Circuitos integrados TTL.

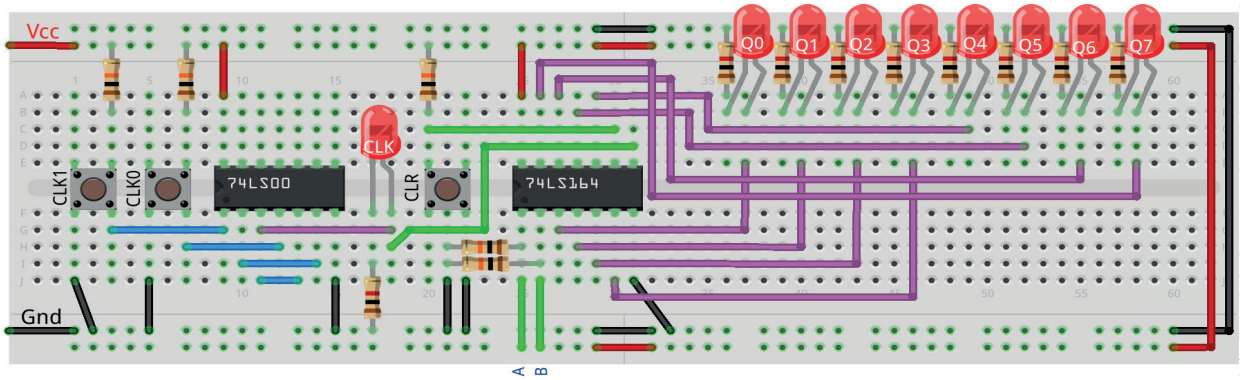


Figura 11.37 – Disposição dos componentes na placa de montagem.

Procedimento:

1. Monte o circuito na placa de montagem, conforme figura 11.37.
   a) Coloque os componentes (resistores, LEDs, CIs e botões) de acordo com a disposição mostrada.
   b) Conecte os fios de saída (roxos) aos LEDs.
   c) Conecte os fios de entrada (verdes).
   d) Conecte os fios de conexão (azuis).
   e) Conecte os fios vermelhos (Vcc) e pretos (Gnd).
   f) Conecte um cabo preto banana-jacaré desde o barramento preto (terra) até o borne preto da fonte de energia.
   g) Conecte um cabo vermelho banana-jacaré no barramento vermelho (Vcc) até o borne vermelho da fonte de energia.
2. Para o preenchimento da tabela 11.12, use os botões como segue:
   a) O botão CLR pressionado envia um ativo baixo “0” e apaga todos os LEDs de saída.
   b) Para configurar os dados de entrada, conecte os fios A e B na barra Gnd para “0” e na barra Vcc para “1”.
   c) Pressionando o botão CLK1 o pulso de relógio é alto “1”.
   d) Pressionando o botão CLK0 o pulso de relógio é baixo “0”.
   e) Um pulso de relógio completo é conseguido pressionando primeiro o botão CLK1 e depois pressionando o botão CLK0 (Relógio).
3. Para o preenchimento das colunas de saída na tabela 11.12, considere:
   a) Se o LED estiver apagado então preencher com “0”.
   b) Se o LED estiver aceso então preencher com “1”.
4. Utilize um programa de simulação em computador (SmartSim, Atanua, Qucs) faça a simulação do circuito da figura 11.35. Compare os resultados com a tabela 11.12.
5. Execute o programa para o Arduíno e compare os valores encontrados com a tabela 11.12.

Tabelas de dados:

Entradas				Saídas							
CLK	MR	A	B	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	0	x	x								
1	1	1	1								
2	1	1	1								
3	1	1	1								
4	1	1	1								
5	1	0	1								
6	1	0	1								
7	1	1	1								
8	1	1	1								
9	1	1	0								
10	1	1	0								
11	1	1	0								
12	1	1	0								
13	1	0	1								
14	0	x	x								
15	1	1	1								
16	1	1	1								
17	1	1	1								
18	1	1	1								
19	1	1	1								
20	1	1	1								
21	1	1	1								
22	1	1	1								

Tabela 11.12

Programa para o Arduíno:

Monte o circuito na placa de montagem, conforme figura 11.38. Conecte os fios azuis e roxos aos respectivos pinos do Arduíno. Conecte os fios vermelho (Vcc) e preto (Gnd) nos terminais apropriados do Arduíno.

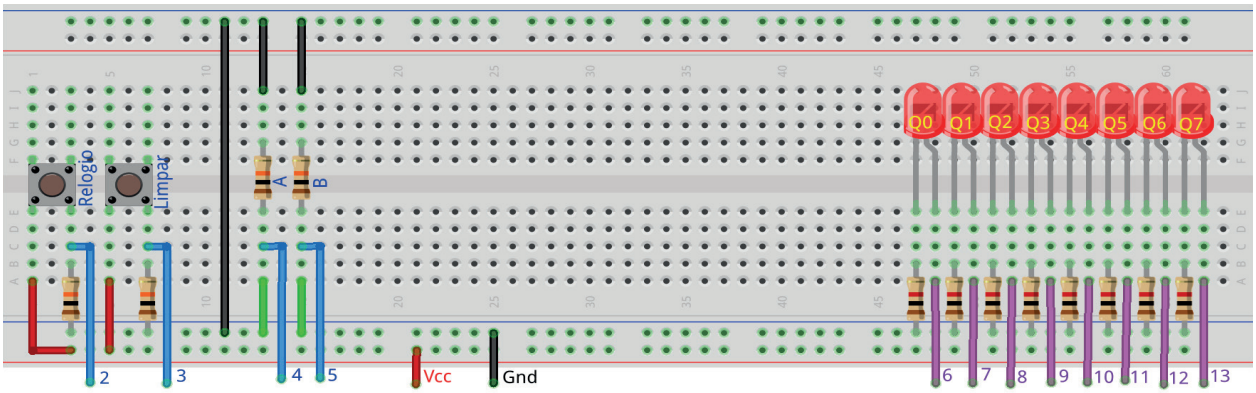


Figura 11.38 – Disposição dos componentes na placa de montagem.

Uso:

- O Botão “Limpar” apaga todos os LEDs.
- Os dados de entrada (“A e B ”) devem ser configurados como: se for “0” o fio verde deve ser conectado na barra “Gnd”; se for “1” o fio verde deve ser conectado na barra “Vcc”. Esta operação deve ser feita antes do pulso de relógio.
- O botão “Relógio” executa as operações de acordo com a tabela 11.11.

```
// Emulador do 74LS164 - registrador de deslocamento de 8 bits
// entrada serial e saída paralela

// pinos de entrada
  #define pino_relogio 2
  #define pino_limpar 3
  #define pino_a 4
  #define pino_b 5

// pinos de saída
  #define pino_ledq0 6
  #define pino_ledq1 7
  #define pino_ledq2 8
  #define pino_ledq3 9
  #define pino_ledq4 10
  #define pino_ledq5 11
  #define pino_ledq6 12
  #define pino_ledq7 13
```

```
// valores do 74LS164
  byte q0; byte q1; byte q2; byte q3; byte q4;
  byte q5; byte q6; byte q7;
  byte a; byte b; byte mr; byte cp; byte cp_old;

// valores de entrada
  byte dadoa; byte dadob;
  byte relogio; byte limpar;
  byte relogio_antes; byte limpar_antes;

void ci_74LS164(){
  // deslocar para direita
  if (mr && cp && !cp_old){
    q7 = q6; q6 = q5; q5 = q4; q4 = q3; q3 = q2; q2 = q1; q1 = q0; q0 = 0;
    if (a && b) q0 = 1;
    cp_old = cp;}
  // limpar
  if(!mr){
    q7 = 0; q6 = 0; q5 = 0; q4 = 0; q3 = 0; q2 = 0; q1 = 0; q0 = 0;}
} // fim do 'ci_74LS164'

void setup(){

// inicialização dos pinos de entrada
  pinMode(pino_relogio, INPUT);
  pinMode(pino_limpar, INPUT);
  pinMode(pino_a, INPUT);
  pinMode(pino_b, INPUT);

// inicialização dos pinos de saída
  pinMode(pino_ledq0, OUTPUT);
  pinMode(pino_ledq1, OUTPUT);
  pinMode(pino_ledq2, OUTPUT);
  pinMode(pino_ledq3, OUTPUT);
  pinMode(pino_ledq4, OUTPUT);
  pinMode(pino_ledq5, OUTPUT);
  pinMode(pino_ledq6, OUTPUT);
  pinMode(pino_ledq7, OUTPUT);

// inicialização das variáveis
```

```
  q0 = 0; q1 = 0; q2 = 0; q3 = 0; q4 = 0; q5 = 0; q6 = 0; q7 = 0;
  relogio_antes = 0;
  limpar_antes = 0;
  cp_old = 0;

} // fim do ‘setup’

void loop(){

// leitura das entradas
  relogio = digitalRead(pino_relogio);
  if (relogio != relogio_antes) delay(100);
  limpar = digitalRead(pino_limpar);
  if (limpar != limpar_antes) delay(100);
  dadoa = digitalRead(pino_a);
  dadob = digitalRead(pino_b);

// configuração para registrador de deslocamento universal
  cp = relogio;
  mr = !limpar;
  a = dadoa;
  b = dadob;

// aplica valores ao CI virtual
  ci_74LS164();

// mostra saídas
  digitalWrite(pino_ledq0,q0);
  digitalWrite(pino_ledq1,q1);
  digitalWrite(pino_ledq2,q2);
  digitalWrite(pino_ledq3,q3);
  digitalWrite(pino_ledq4,q4);
  digitalWrite(pino_ledq5,q5);
  digitalWrite(pino_ledq6,q6);
  digitalWrite(pino_ledq7,q7);

// reinicia variáveis
  relogio_antes = relogio;
  limpar_antes = limpar;

} // fim do ‘loop’
```

Apêndice A.
Diretrizes do Relatório Técnico

É essencial que os indivíduos possam expressar suas ideias e defender seus argumentos com clareza, detalhe e sutileza. Da mesma forma, é importante que eles possam ler e criticar as ideias e argumentos dos outros da mesma maneira. A criação de relatórios de laboratório ajuda nesse esforço. Todos os relatórios devem ser limpos e legíveis. Espera-se um estilo padrão de redação técnica, além de gramática e ortografia corretas. Isso significa que voz ativa, primeira pessoa, pronomes pessoais e afins devem ser evitados. Por exemplo, não escreva "Eu configurei a fonte de alimentação para 6 volts". Em vez disso, use "A fonte de alimentação foi configurada para 6 volts". Relatórios são um esforço individual. Embora seja perfeitamente adequado discutir seus dados e resultados experimentais com seu parceiro de laboratório, a criação do relatório em si é um exercício individual. Plágio não será tolerado. Um relatório deve estar de acordo com o seguinte esquema, na ordem dada:

1. Informações Gerais. Título, data, seu nome, nome do parceiro.
2. Objetivo (Hipótese). Responda a pergunta: "O que é / são o(s) item(ns) sob investigação e seu(s) relacionamento(s) proposto(s)?" Estas são declarações dos itens que você está testando neste exercício em particular.
3. Conclusão. Responda a pergunta "O que foi mostrado / verificado?" Estas são declarações de fato concisas sobre a(s) ação(ões) do circuito sob investigação. Certifique-se de ter passado da situação específica do laboratório para o caso geral. Se tudo funcionar bem, eles devem combinar muito bem com a seção Objetivo. Em nenhuma circunstância você deve chegar a uma conclusão que não seja apoiada por seus dados, mesmo que essa conclusão seja indicada no texto ou em uma aula. O que importa aqui é o que você fez e sua análise disso. Se houver uma discrepância entre seus resultados e a teoria, declare a discrepância e não ignore seus resultados.
4. Discussão (Análise). Reduza e analise seus dados. Explique a ação do circuito ou conceitos sob investigação. Relacione os resultados teóricos com os resultados do laboratório. Não diga apenas o que aconteceu, mas comente o motivo e suas implicações. Tire suas conclusões desta seção. Quaisquer desvios do procedimento dado (manual de laboratório ou folheto) devem ser anotados nesta seção. A discussão é a penúltima parte que você escreve.
5. Folha de Dados Final. Inclua todos os dados derivados e calculados. Certifique-se de incluir desvios percentuais para cada par teoria / medida. Use Desvio percentual = (Teoria - Medida) / Teoria * 100 e inclua o sinal.
6. Gráficos, Respostas às perguntas no final do exercício, Outros. Todos os gráficos devem ser adequadamente titulados, criados usando escalas apropriadas e identificados com rótulos. Sugere-se que os gráficos sejam criados com um programa de plotagem ou uma planilha. Como alternativa, os gráficos podem ser criados manualmente, mas devem ser desenhados usando uma borda reta ou uma curva francesa (dependendo do tipo de gráfico) no papel de gráfico apropriado.

Certifique-se de deixar espaço suficiente nas margens e entre as seções para meus comentários. O espaçamento entre linhas 1,5 ou duplo é bom. Os relatórios de várias páginas devem ser grampeados no canto superior esquerdo. Clipes de papel, dobráveis, pedaços de fio de ligação, etc. não são aceitáveis. Abaixo está o padrão de classificação.

Nota A: O relatório atende ou excede os detalhes da atribuição. O relatório é elegante e profissional na aparência, incluindo ortografia e sintaxe adequada. A análise está no nível apropriado e com detalhes suficientes. Tabelas de dados e dados gráficos são apresentados de forma clara e

concisa. As soluções de problemas são suficientemente detalhadas e corretas. Diagramas têm uma aparência profissional.

Nota B: O relatório está próximo do ideal, embora tenha algumas desvantagens menores, que podem incluir alguns erros ortográficos ou gramaticais, análises que podem não ter detalhes suficientes, omissões menores em dados tabulares ou gráficos e afins. Em geral, o relatório é sólido, mas poderia usar refinamento ou simplificação.

Nota C: O relatório é útil e transmite as principais ideias, embora possa ser vago em alguns pontos. Erros ortográficos e gramaticais podem ser mais numerosos do que aqueles encontrados em um relatório de notas A ou B. Algumas lacunas nos dados ou omissões nas explicações podem ser vistas.

Nota D: Além de erros ortográficos e gramaticais típicos, o relatório sofre de erros lógicos, como conclusões que não são suportadas por dados laboratoriais. As análises tendem a ser vagas e possivelmente enganosas. Gráficos e diagramas são desenhados de uma maneira pouco clara.

Nota F: O relatório apresenta muitas das seguintes deficiências: erros ortográficos e gramaticais excessivos, seções ausentes, como gráficos, tabelas e análises, análises descaradamente incorretas, dados desobedientes ou incompreensíveis, soluções problemáticas tendem a estar incorretas ou ausentes e gráficos dados ou diagramas são apresentados de uma maneira inferior.

Apêndice B.
Um exemplo de um relatório técnico

O que se segue, começando na próxima página, é um exemplo de um relatório técnico de laboratório. Leia o exemplo depois de ler as diretrizes do relatório acima. Isso usa o estilo não formal.

O experimento em questão é completamente fabricado, mas o relatório ilustrará a forma e o conteúdo esperados. O experimento simulado envolve medir a velocidade do som em vários materiais e se essa velocidade é afetada pela temperatura. Neste exercício, o experimentador colocou pequenos transdutores em cada extremidade de uma barra sólida do material sob investigação (mais ou menos como um pequeno alto-falante e microfone). Um pulso é então aplicado a uma extremidade e um temporizador é usado para determinar quanto tempo leva para a onda atingir a outra extremidade. Conhecendo o comprimento da barra, a velocidade pode ser calculada. As barras são então aquecidas a diferentes temperaturas e o processo repetido para ver se a velocidade muda. Tabelas e gráficos apropriados são apresentados.

O relatório usa fonte Times Roman de 12 pontos com espaçamento entre linhas de 1,5, embora 11 ou até 10 pontos possam ser preferidos. Não há razão para "ficar chique" com a aparência do relatório. Na verdade, isso servirá apenas como distração. Espaço suficiente é deixado para o instrutor inserir comentários. O tamanho de qualquer relatório específico pode variar muito dependendo da quantidade de dados gravados, da profundidade da análise, dos gráficos adicionados e afins.

Como às vezes é o caso, esse experimento simulado não funcionou perfeitamente.

Velocidade do som em vários materiais

Ciência das Coisas, ET301

30 de fevereiro de 2112

Nome: João Bufão

Parceiro: José Falha

Objetivo

A hipótese investigada neste exercício é direta, a saber, que a velocidade de propagação do som depende das características do material e que ele pode ser afetado pela temperatura. Três materiais diferentes serão investigados, cada um em três temperaturas diferentes. Espera-se que a velocidade nos três materiais seja significativamente maior que a velocidade do som no ar (343 metros por segundo).

Conclusão

A velocidade do som em um determinado material depende das características internas do material. A velocidade pode aumentar ou diminuir com a temperatura. A velocidade em temperatura ambiente para a liga SB foi de aproximadamente 2001 metros por segundo com um coeficiente de temperatura (TC) de 0,01%. A liga GA foi de 3050 metros por segundo com -0,2% TC, e o material CCCD foi medido a 997 metros por segundo com 0,1% de TC. Todos os valores

estavam dentro de alguns por cento daqueles previstos pela teoria, e todas as velocidades eram claramente muito maiores que a velocidade do som no ar.

Discussão

Para investigar a velocidade do som, foram obtidas três barras de material, cada uma com um metro de comprimento. O primeiro foi "Sonic Bronze" (Bronze Sônico) ou SB, uma liga de estanho, cobre, zinco e porcupínio. O segundo material, "Green Aluminium" (Alumínio Verde) ou GA, é uma liga de alumínio e criptonita, enquanto o terceiro, CCCD, é comumente conhecido como "Chocolate Chip Cookie Dough" (Massa de Biscoito com Pedaços de Chocolate).

Um transdutor acústico foi anexado a cada extremidade da barra sob investigação. Um pulso foi aplicado a uma extremidade e um temporizador digital foi usado para determinar quanto tempo demorava para a onda percorrer a barra até o transdutor de captação. Como cada barra tinha um metro de comprimento, a velocidade em metros por segundo é simplesmente 1 / atraso de tempo. A barra foi então colocada num forno industrial e a medição foi repetida a temperaturas de 75 °C e 125 °C para comparar com os resultados nominais da temperatura ambiente (25 °C).

Os resultados da temperatura ambiente concordaram fortemente com os dados publicados dos três materiais. A comparação da Tabela 1.1 com a coluna 25C da Tabela 1.2 mostrou um desvio não pior do que 1,64% (coluna final, Tabela 1.2). A variação entre os materiais é de aproximadamente 3:1, indicando com que intensidade as características internas do material influenciam a velocidade de propagação. O material CCCD, sendo o mais plástico, deve ter as maiores perdas de atrito interno e, portanto, a velocidade mais lenta do grupo. Este foi o caso. A inclusão de porcupínio na liga SB foi responsável pela modesta velocidade deste material. As ondas têm que se propagar de forma relativamente lenta através do porcupínio em comparação com a liga GA que é livre deste ingrediente. A velocidade de propagação de todos os materiais foi significativamente mais rápida que a velocidade do som no ar. Mesmo o mais lento do grupo, CCCD, exibiu uma velocidade quase três vezes maior que a do ar.

Os coeficientes de temperatura também mostraram concordância, e parecem estar dentro de apenas alguns por cento dos valores estabelecidos. Geralmente, a velocidade aumenta com a temperatura, embora a liga GA tenha produzido o efeito oposto. Supõe-se que a inclusão de criptonita na liga pode ser responsável por isso. Veja o Gráfico 1.1 para detalhes. Havia uma questão prática envolvendo o material do CCCD. As medições a 25 °C e 75 °C foram satisfatórias, no entanto, quando a barra de CCCD foi removida do forno de 125 ° C, alterou a textura e a cor para um castanho dourado crocante e produziu um odor forte e agradável. Consequentemente, um membro do grupo de laboratório comeu aproximadamente 10 centímetros da barra antes que a velocidade pudesse ser medida. Para corrigir isso, o atraso de tempo medido foi ajustado por um fator de 1,11 quando a barra foi reduzida para 90% de seu comprimento original.

Dados

Material	Velocidade do Material (m/s)	Coeficiente de temperatura (% de alteração por grau C)
SB	2000	0,01
GA	3000	-0,21
CCCD	1000	0,105

Tabela 1.1 - Velocidades Teóricas Publicadas e TC

Material	Velocidade 25C (m/s)	Velocidade 75C (m/s)	Velocidade 125C (m/s)	Coeficiente de Temperatura	% Desvio a 25C
SB	2001	2010	2021	0,01	0,05
GA	3050	2750	2440	-0,2	1,64
CCCD	997	1049	1097 *	0.1	-0.3

Tabela 1.2 - Velocidades experimentais e TC

* *Veja Discussão* para explicação

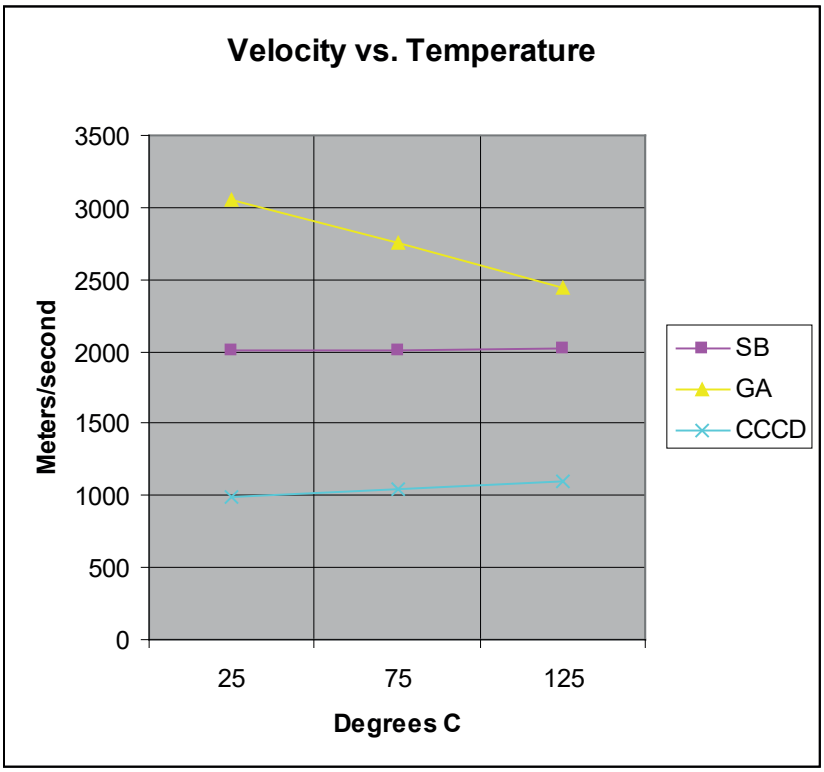


Gráfico 1.1 – Variação de Velocidade com Temperatura, por Material.

Respostas às perguntas do exercício.

1. *A velocidade do som não é afetada pela temperatura?*

Não. O gráfico 1.1 mostra que, em alguns casos (SB e CCCD), a velocidade é diretamente proporcional à temperatura, embora possa ser inversamente proporcional (GA).

2. *Se o material do CCCD também tivesse sido submetido a 175C, o que você esperaria?*

É improvável que uma velocidade de 175C possa ter sido medida, já que a barra inteira provavelmente teria sido consumida pela equipe do laboratório antes que os transdutores pudessem ser aplicados.

Apêndice C. Armazenando variáveis matemáticas no Arduíno

Definindo Tipos de Dados

O ambiente do Arduino é na verdade apenas C ++ com suporte a bibliotecas e suposições internas sobre o ambiente de destino para simplificar o processo de codificação. C ++ define vários tipos de dados diferentes; aqui falaremos apenas sobre aqueles usados no Arduino, com ênfase nas armadilhas que aguardam o programador incauto do Arduino.

Abaixo está uma lista dos tipos de dados comumente vistos no Arduino, com o tamanho da memória de cada um entre parênteses após o nome do tipo. Nota: as variáveis "com sinal" permitem números positivos e negativos, enquanto variáveis "sem sinal" permitem somente valores positivos.

- boolean (8 bits) - Lógica simples verdadeira / falsa. Apenas operações logicas.
- byte (8 bits) - Número sem sinal de 0 a 255.
- char (8 bits) - Número com sinal de -128 a 127. O compilador tentará interpretar esse tipo de dados como um caractere em algumas circunstâncias, o que pode gerar resultados inesperados.
- unsigned char (8 bits) - O mesmo que 'byte'. Se é isso que você procura, use "byte", por motivos de clareza.
- word (16 bits) - Número sem sinal de 0 a 65535.
- unsigned int (16 bits) – O mesmo que "word". Use "word" por motivos de clareza e brevidade.
- int (16 bit) - número com sinal de -32768 a 32767. Isso é mais comumente o que você vê usado para variáveis de uso geral nos códigos de exemplo do Arduino fornecido com o IDE.
- short (16 bit) – número com sinal de -32768 a 32767. O mesmo que int.
- unsigned long (32 bits) - número sem sinal de 0 a 4,294,967,295. O uso mais comum disso é armazenar o resultado da função millis(), que retorna o número de milissegundos que o código atual está executando.
- long (32 bits) - número com sinal de -2.147.483.648 a 2.147.483.647
- float (32 bit) - número com sinal de -3.4028235E38 a 3.4028235E38. O ponto flutuante no Arduino não é nativo; o compilador tem que pular através de rotinas internas para fazê-lo funcionar. Se você puder evitar, você deve.

Constantes Inteiras

Descrição

Constantes inteiras são números que são usados diretamente em um programa, como 123. Por padrão, esses números são tratados como int, mas você pode alterar isso com os modificadores U e L (veja abaixo).

Normalmente, constantes inteiras são tratadas como inteiros de base 10 (decimal), mas notações especiais (formatadores) podem ser usadas para inserir números em outras bases.

Base	Exemplo	Formatador	Comentários
10 (decimal)	123	Nenhum	
2 (binário)	B1111011	Letra 'B' antes	Só funciona com valores de 8 bits (0 a 255) caracteres 0 e 1 válidos
8 (octal)	0173	Número "0" (zero) antes	Caracteres 0-7 válidos
16 (hexadecimal)	0x7B	Sequência "0x" (zero-xis) antes	Caracteres 0-9, A-F, a-f válidos

Decimal (base 10)

Essa é a matemática do senso comum com a qual você está familiarizado. Constantes sem outros prefixos são assumidos como estando no formato decimal.

Exemplo de código:

```
n = 101; // igual a 101 decimal ((1*10^2)+(0*10^1)+1)
```

Binário (base 2)

Apenas os caracteres 0 e 1 são válidos.

Exemplo de código:

```
n = B101; // mesmo que 5 decimal ((1*2^2)+(0*2^1)+1)
```

O formatador binário só funciona em byte (8 bits) entre 0 (B0) e 255 (B11111111). Se for conveniente inserir um int (16 bits) em formato binário, você pode fazer um procedimento em duas etapas, como:

```
myInt = (B11001100 * 256) + B10101010; // B11001100 é o byte alto
```

Octal (base 8)

Apenas os caracteres de 0 a 7 são válidos. Os valores octais são indicados pelo prefixo "0" (zero).

Exemplo de código:

```
n = 0101; // mesmo que 65 decimal ((1*8^2)+(0*8^1)+1)
```

É possível gerar um erro difícil de encontrar (inadvertidamente) incluindo um zero à esquerda antes de uma constante e ter o compilador intencionalmente interpretando sua constante como octal.

Hexadecimal (base 16)

Os caracteres válidos são de 0 a 9 e letras de A a F; A tem o valor 10, B é 11, até F, que é 15. Os valores hexadecimais são indicados pelo prefixo “0x”. Note que A-F pode ser digitado em maiúscula ou minúscula (a-f).

Exemplo de código:

n = 0x101; // mesmo que 257 decimal ((1*16^2)+(0*16^1)+1)

Notas e avisos

Formatadores U & L:

Por padrão, uma constante inteira é tratada como um int com as limitações de valores da variável servidora. Para especificar uma constante inteira com outro tipo de dados, siga com:

- um ‘u’ ou ‘U’ para forçar a constante em um formato de dados sem sinal. Exemplo: 33u
- um ‘l’ ou ‘L’ para forçar a constante em um formato de dados longo. Exemplo: 100000L
- um ‘ul’ ou ‘UL’ para forçar a constante em uma constante longa sem sinal. Exemplo: 32767ul

Constantes de ponto flutuante

Descrição

Semelhante a constantes inteiras, as constantes de ponto flutuante são usadas para tornar o código mais legível. Constantes de ponto flutuante são trocadas em tempo de compilação para o valor para o qual a expressão avalia.

Exemplo de código

n = 0.005; // 0.005 é uma constante de ponto flutuante

Notas e avisos

As constantes de ponto flutuante também podem ser expressas em uma variedade de notação científica. ‘E’ e ‘e’ são ambos aceitos como indicadores expoentes válidos.

Constante de Ponto Flutuante	Escrita como:	Também escrita como:
10.0	10	
2.34E5	2.34*10^5	234000
67e-12	67.0*10^-12	0.000000000067

Apêndice D. Circuitos Integrados.

Circuitos integrados (CIs) são formados pela integração de muitos sistemas. Circuitos integrados são chips semicondutores compostos de elementos de circuito como transistores, resistores, capacitores e diodos. Esses elementos estão conectados uns aos outros para formar um circuito dentro do chip. Este circuito é capaz de conectar o mundo externo a partir de nós adequados, com a ajuda de pinos. Em seguida, o chip é coberto por metal e plástico para proteção. Então o circuito integrado é obtido. O custo do circuito integrado é tão pequeno quanto seu tamanho. Circuitos integrados desempenham um papel importante no desenvolvimento de tecnologia.

Circuitos integrados são menores, mais baratos e mais rápidos, tornando-os amplamente utilizados na indústria. Circuitos integrados são representados pela abreviatura CI. CIs geralmente têm pacotes padrão e têm números de pinos de 8 a 400. Todos os CIs possuem códigos numéricos. Pelo uso desses números, pode-se entender o tipo do CI e a função do circuito nele. Circuitos integrados podem ser classificados em dois grupos: Circuitos Integrados Digitais e Circuitos Integrados Analógicos.

Circuitos integrados analógicos: São geralmente usados para amplificação e retificação de tensão. (Por exemplo, amplificadores operacionais)

Circuitos Integrados Digitais: Eles são compostos de portas lógicas.

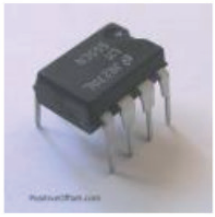
Atualmente, os CIs digitais são usados em circuitos digitais. Uma vez que eles têm tamanho pequeno e baixo custo, eles são amplamente utilizados na indústria. Alguns dos circuitos nos quais CIs são usados são amplificadores de potência, contadores, unidades aritméticas, reguladores de voltagem, circuitos de rádio e TV, amplificadores operacionais, etc. Os circuitos integrados são classificados de acordo com suas estruturas. Eles são divididos em três grupos de acordo com suas embalagens: metal, plástico e cerâmica. Mas os CIs de cerâmica não são usados, pois são frágeis e caros. CIs geralmente não são consertados quando eles quebram. Eles são trocados por um mais novo. Soquetes padrão são produzidos para realizar essas operações de maneira mais prática. Em alguns circuitos, os CIs são colocados nesses soquetes para que encaixar e desmontar o CI se torne mais prático.

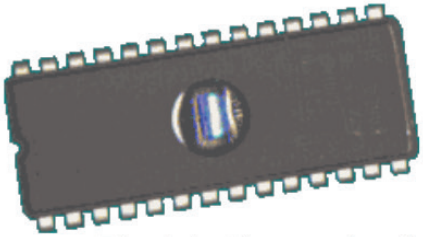
Circuitos integrados digitais são classificados de acordo com o número de portas que eles incluem.

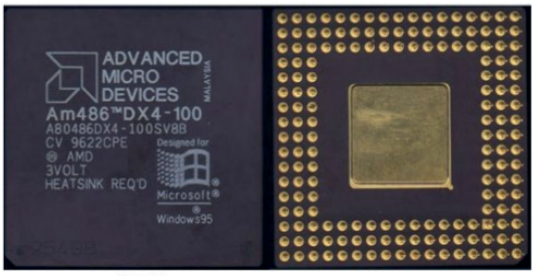
Esses são:

- SSI (Integração de Pequena Escala): Eles contêm portas com numeração entre 1 e 20, ou seja, CI 7400 contém 4 portas NAND.
- MSI (Integração de Média Escala): Eles contêm numerações de portas entre 20 e 100, ou seja, flip-flops, contadores.
- LSI (Integração de Grande Escala): Eles contêm portas com numeração entre 100 e 10000, ou seja, microprocessadores (4 ou 8 bits)
- VLSI (Integração de Escala Muito Grande): Contêm portas com numeração superior a 10000, isto é, microprocessadores (16 ou 32 bits), elementos de memória, circuitos de computador.

Hoje é possível produzir chips contendo mais de 200.000 portas lógicas. Cada porta contém mais de dois transistores. Portanto, podemos dizer que 100 milhares de transistores podem ser formados em um CI. Por exemplo, um microprocessador Pentium contém 5,5 milhões de transistores. Vários tipos de CIs são mostrados na Figura abaixo.







Parâmetros do Circuito Integrado

Os circuitos integrados digitais têm algumas especificações importantes para compará-los entre si e escolher o CI mais adequado para a aplicação específica. Essas especificações são chamadas de parâmetros CI. Os parâmetros mais importantes são:

1. Tensão da fonte de alimentação: Determina a tensão de alimentação do CI. A tolerância de tensão de alimentação também pode ser indicada em alguns CIs.
2. Atraso de propagação: Indica a rapidez com que a saída muda quando a entrada do CI muda. A saída de um circuito lógico não muda simultaneamente quando a entrada muda. É medido em nanossegundos. Alguma quantidade de atraso ocorre. Esse atraso é chamado de atraso de propagação. São 5 nseg para o TTL. CIs com atraso mínimo de propagação são preferidos em sistemas como CLP e computador.
3. Dissipação de energia: Indica quanta energia é dissipada no circuito. É medido em miliWatts. Aumenta com a diminuição do atraso de propagação. (CIs com baixa dissipação de energia são preferidos em circuitos operados por bateria).
4. Carga de saída (fan out): Determina a capacidade máxima da carga que será conectada à saída do circuito. A capacidade máxima de carga determina o número de portas que serão conectadas à saída.
5. Imunidade ao ruído: Determina a capacidade de tolerância ao ruído dos dados de saída. O ruído pode fazer com que a lógica 0 apareça como lógica 1 na saída ou vice-versa. Os dados de saída estão isentos de erros quanto menos a quantidade de ruído. A imunidade ao ruído é determinada pela capacidade de tolerância ao ruído de pota.
6. Frequência do relógio: determina a frequência máxima do pulso de disparo que será aplicado à entrada do circuito.

As portas NAND e NOR são os circuitos básicos para todos os CIs. CIs lógicos são nomeados pelo material eletrônico usado quando são fabricados. Alguns tipos de circuitos integrados são:

RTL - Lógica Resistor - Transistor

DTL - Lógica Transistor - Diodo

HTL - Lógica de alto limiar

TTL - Lógica Transistor - Transistor

ECL - Lógica Acoplada a Emissor

DCTL - Lógica de Transistor Acoplado Diretamente

MOS - Lógica de Semicondutor de Óxido de Metal

CMOS - Lógica de Semicondutor de Óxido de Metal Complementar

Tipos de família lógicos de circuitos integrados:

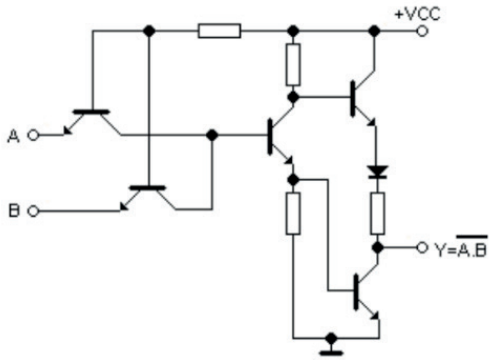
Lógica Resistor - Transistor (RTL): São os primeiros circuitos integrados comerciais fabricados. Eles foram amplamente utilizados no início, uma vez que eram baratos. Sua tensão de operação está entre 3V e 3,6V, o atraso de propagação é de aproximadamente 12 ns, a dissipação de energia é de 10 mWatts por porta e eles são codificados por números com 700 e 900.

Lógica Transistor - Diodo (DTL): Seu uso é limitado em um CI devido a suas desvantagens. Há alguma quantidade de queda de tensão nos diodos. Isso causa uma queda no nível de tensão. A descarga também é um problema. Para lidar com esses problemas, são usados amplificadores de descarga. Assim, as quedas nos níveis lógicos também são toleradas. Eles são menos usados. Esses CIs são melhores que RTL em velocidade, dissipação de energia e estabilidade. Sua voltagem operacional é de cerca de 5 V, a imunidade a ruído é baixa e eles são codificados por números com 830 e 930.

Lógica Transistor - Transistor (TTL): TTL é a versão melhorada da lógica DTL. Um transistor multi-emissor é usado em vez do diodo na entrada da lógica DTL. Assim, os CIs TTL operam muito rapidamente e, por esse motivo, são amplamente utilizados. Hoje eles são um dos grupos CI mais utilizados. Eles também são usados em computadores.

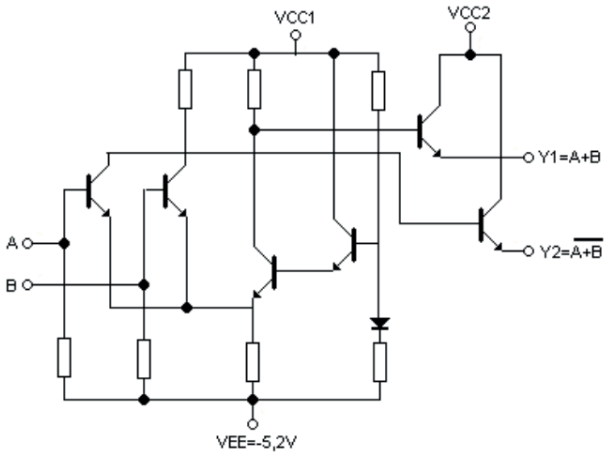
Eles são divididos em 5 grupos:

1. TTL padrão: é o primeiro tipo de grupo TTL. Sua dissipação de energia é de 10 mW por porta, o atraso de propagação é de 10 ns e a freqüência do clock é de 35 MHz.
2. TTL de baixa potência: Sua dissipação de energia é de 1 mW por porta, o atraso de propagação é de 33 ns. e freqüência de clock é de 3 MHz.
3. TTL de alta velocidade: Sua dissipação de energia é de 22 mW por porta, o atraso de propagação é de 6 ns. e freqüência de clock é de 50 MHz.
4. Schottky TTL (STTL): É o elemento mais rápido do grupo TTL. Sua dissipação de energia é de 19 mW por porta, o atraso de propagação é de 3 ns. e freqüência de clock é de 125 MHz.
5. Schottky TTL Low Power (LSTTL): É o último elemento melhorado do grupo TTL. Sua dissipação de energia é de 2 mW por porta, o atraso de propagação é de 20 ns. e freqüência de clock é de 25 MHz.



A figura anterior mostra a implementação TTL de uma porta NAND de duas entradas. O grupo TTL é codificado por números com 7400 e 5400. CIs com o número 7400 são mais amplamente utilizados. A série 5400 é produzida para fins militares. A série 7400 cobre os CIs que operam em temperaturas entre 0°C e 70°C. A série 5400 opera a temperaturas entre -55°C e +125°C. As letras após os números 74 e 54 determinam a que subgrupo pertence o CI TTL. Por exemplo, 74L00 é usado para TTL de baixa potência e 7400 é usado para TTL padrão.

Lógica acoplada por emissor (ECL): Não é usado tão amplamente quanto a família TTL. É o grupo mais rápido de circuitos integrados. Foi produzido pela primeira vez pela Motorola em 1962. É mais caro que o TTL e é mais difícil de resfriar. Também é difícil fazer interconexões e tem menor imunidade a ruídos. Em muitas aplicações, as portas ECL operam mais rapidamente. Por outro lado, eles são usados em supercomputadores e computadores específicos para aplicações de alta velocidade. A Figura abaixo mostra a implementação ECL de um gate OR (NOR) de duas entradas.



Existem 4 subgrupos principais da ECL.

1. Grupo MECL 1: Sua dissipação de energia é de 35 mW por porta, o atraso de propagação é de 8 ns. e frequência de clock é de 30 MHz. Eles são codificados por números com 300 e 350.
2. Grupo MECL 2: é a versão melhorada do Grupo MECL 1. Sua dissipação de energia é de 22 mW por porta, o atraso de propagação é de 4 ns. e a frequência do clock é de 75 MHz. Eles são codificados por números com 1000 e 1200.
3. Grupo MECL 10K: É um dos subgrupos ECL mais utilizados. Sua dissipação de energia é de 25 mW por porta, o atraso de propagação é de 2 ns. e frequência de clock é de 125 MHz. Eles são codificados por números com 10000.
4. Grupo MECL 3: É o mais rápido dos subgrupos ECL. Sua dissipação de energia é de 60 mW por porta, o atraso de propagação é de 1 ns. e frequência de clock é de 400 MHz. Eles são codificados por números com 1600.

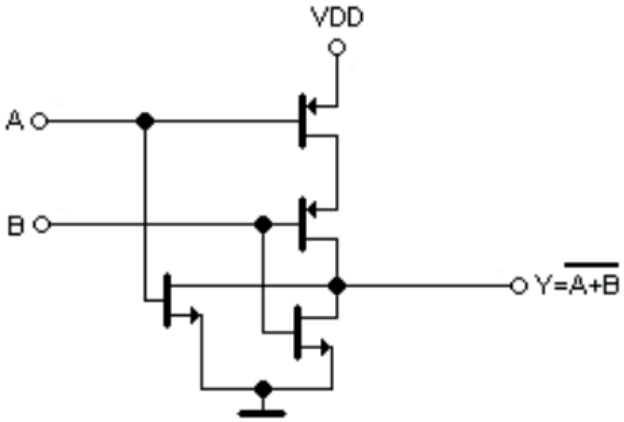
O nível lógico 1 é -0.75V e o nível lógico 0 é -1.55 V para portas ECL.

Lógica de Semicondutor de Óxido de Metal (MOS): Os CIs tipo MOS começaram a ser fabricados com as melhorias nos transistores de efeito de campo (FET). Os transistores usados nesses CIs são chamados de MOSFET. Uma vez que eles são lentos, duráveis e têm pouca saída, eles não são preferidos em algumas aplicações.

Mas, em muitas aplicações, elas são usadas devido ao tamanho reduzido dos chips, à facilidade de fabricação e à baixa dissipação de energia. Sua tensão operacional está entre 3V e 15V. Você deve ter cuidado com as cargas estáticas e quando estiver trabalhando com esses CIs;

- Você não deve tocar nos pinos,
- Você deve usar ferro de solda DC ou aterrado,
- Os pinos não utilizados não devem ser deixados desconectados. Eles devem estar conectados a + V ou ao chassi, o que for apropriado.

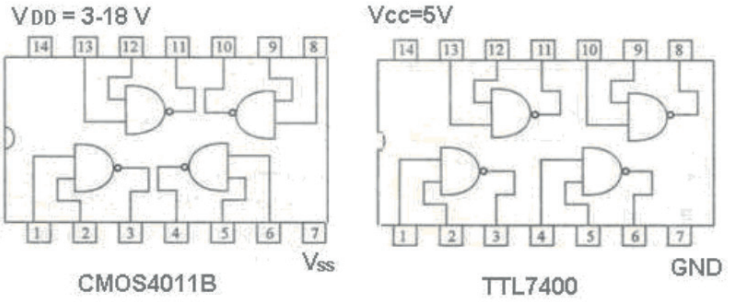
Lógica de Semicondutor de Óxido de Metal Complementar (CMOS): Alguns circuitos MOSFET são projetados especialmente para aplicações espaciais e navais. E esses circuitos são chamados de circuitos complementares MOS. Eles são construídos com a ajuda da lógica FET e MOSFET. Eles são versão melhorada dos TTLs. Sua dissipação de energia é baixa e a imunidade ao ruído é alta. Eles são lentos em comparação com circuitos lógicos de alta velocidade. Muitos transistores podem ser implementados em um único chip e a faixa de tensão da fonte de alimentação é alta. Seu custo de fabricação também é baixo. As versões mais recentes dos circuitos CMOS são mais rápidas e são amplamente utilizadas em relógios, calculadoras e microprocessadores eletrônicos. Circuitos CMOS pertencem à série 40XX. A tensão de alimentação é entre 3V e 18V. Seu atraso de propagação é alto. Eles operam a 5MHz com tensão máxima de alimentação. Portanto, eles não são adequados para aplicações de alta frequência. A implementação da CMOS de uma porta NOR de duas entradas é mostrada na figura abaixo.



Lógica de Injeção Integrada (IIL): É necessário seguir uma maneira diferente no design das portas lógicas IIL porque elas têm uma entrada e várias saídas. Mas suas vantagens compensam as dificuldades de design. Assim, elementos de memória e microprocessadores IIL ocorrem no mercado. Desenvolvimentos recentes na tecnologia IIL foram insignificantes em comparação com o desenvolvimento na tecnologia CMOS. Seus recursos de operação são semelhantes à lógica RTL, exceto por algumas diferenças. A ausência de resistência em sua estrutura os torna mais baratos e capazes de serem densamente empacotados.

Capacidade de carga de saída (fan-out): Este valor é 10 para circuitos TTL. Isso significa que uma saída TTL pode alimentar no máximo 10 entradas TTL. O valor é 50 para CMOS, cuja impedância de saída é muito alta.

Pinos não utilizados: Não deve haver pinos flutuantes nos circuitos integrados TTL e CMOS. Pinos não utilizados na aplicação devem ser conectados ao pólo (+) ou (-) da fonte, o que for apropriado. Caso contrário, resultados inesperados podem aparecer na saída. Estruturas internas de CIs TTL e CMOS que possuem a mesma funcionalidade são mostradas na figura seguinte. Também suas especificações são comparadas entre si na tabela.



Parâmetro	TTL	CMOS
Tensão de alimentação	5V DC	3 V - 18 V DC
Corrente necessária	Miliampere	Microampere
Impedância de entrada	Baixo	Muito alto
Velocidade de comutação	Rápido	Baixo
Carregamento de saída	10	50
Dissipação de energia	20 mW	2 mW
Pulso de disparo	50 MHz	25 MHz
Tolerância de tensão	20%	50%

Apêndice E. Usando DIPs em placas de montagem sem solda

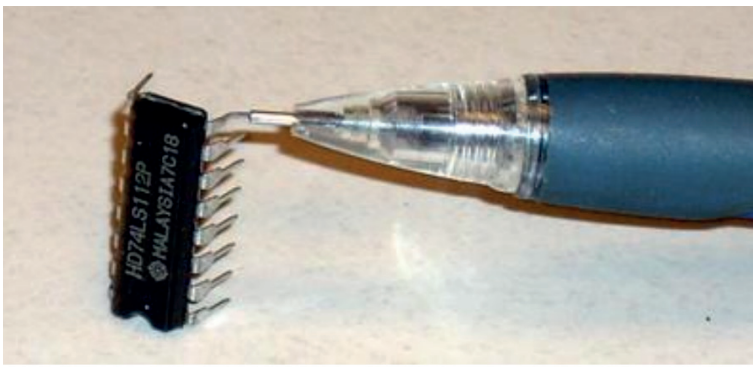
Lembre-se desses pontos ao conectar um circuito que usa um DIP (dual inline package):

Endireitando um pino torto

Antes de inserir um DIP na breadboard, examine seus pinos para ter certeza de que estão todos alinhados. Se um pino estiver torto, use um alicate de bico ou uma lapiseira para endireitá-lo.

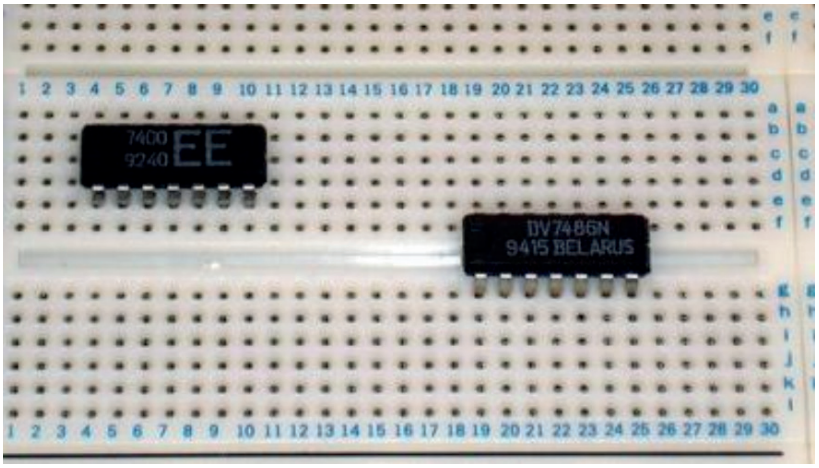
Para usar uma lapiseira:

1. Retrair o grafite da lapiseiras para que o orifício na ponta esteja aberto.
2. Deslize cuidadosamente o pino dobrado na ponta da lapiseira, conforme mostrado na foto abaixo.
3. Delicadamente, endireite o pino.

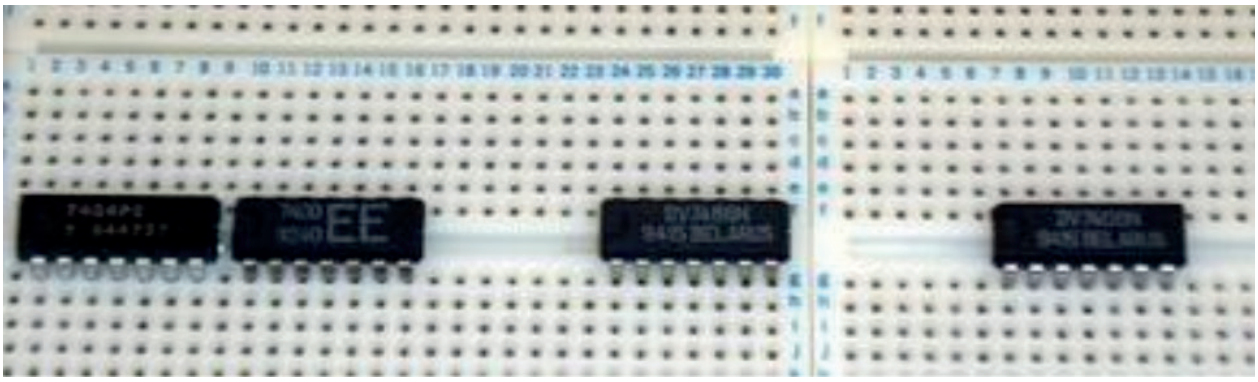


Inserindo um DIP

Quando você insere um DIP na placa de montagem, certifique-se de que os pinos em um lado do DIP não estejam conectados aos pinos do outro lado do DIP. Isso significa que o DIP deve ocupar uma das lacunas longas que dividem a breadboard em seções separadas. Na foto abaixo, o DIP à esquerda é inserido incorretamente, porque os pinos no lado inferior do chip estão conectados aos pinos no lado superior. O DIP à direita é inserido corretamente, porque o DIP ultrapassa a lacuna da breadboard.



Ao construir um circuito que usa dois ou mais DIPs, não coloque os DIPs ao lado um do outro. Em vez disso, deixe algum espaço entre eles para que você possa passar os fios entre eles. Na foto abaixo, os dois DIPs à esquerda estão muito próximos. Os dois DIPs à direita estão devidamente espaçados.

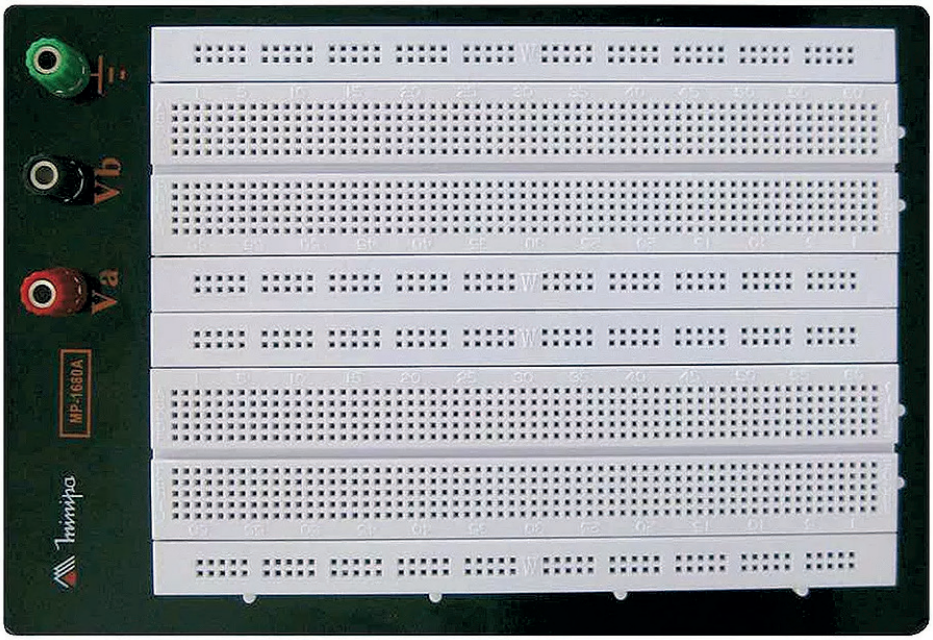


Oriente todos os seus DIPs na mesma direção. Se você estiver colocando os DIPs horizontalmente, verifique se eles estão posicionados de modo que cada pino do DIP 1 esteja no canto inferior esquerdo, não no canto superior direito.

Em um novo DIP que nunca foi usado, os pinos geralmente são espalhados muito para fora para serem inseridos na breadboard. Pressione suavemente os pinos em ambos os lados do chip para dentro de modo que o espaçamento corresponda ao espaçamento dos orifícios na placa de montagem.

Usando barramentos de energia e terra

Para conexões de energia e aterramento ao seu DIP, use os barramentos na breadboard. Nas placas de nossos laboratórios, esses barramentos estão localizados nas laterais e no centro da placa de montagem. A foto abaixo mostra os 4 barramentos destacados como retângulos estreitos com a letra W no centro. Esta letra indica interrupção do barramento.



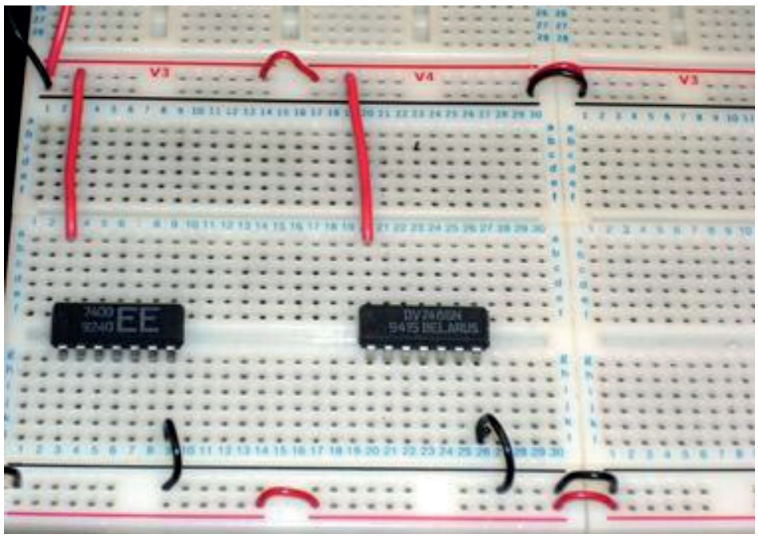
Cada barramento possui duas linhas de conexões e se estendem por vinte e cinco buracos horizontais. Esses vinte e cinco buracos estão todos conectados juntos. Uma das linhas pode ser denominada de energia +5V e a outra de terra. Portanto, se você conectar um fio de qualquer um desses orifícios ao terminal de energia da fonte, os outros vinte e quatro buracos estarão todos em +5V. O mesmo vale para o fio de terra. Normalmente as barras superiores são de +5V e as inferiores são de terra.

Você pode estender os barramentos usando cabos de jumper curtos para conectar dois ou mais barramentos juntos. Conecte um jumper nos furos sobre a letra W e conecte a linha superior com a linha inferior. Desta forma você terá barramentos que se estendem por toda a largura da placa de montagem, fornecendo muitos pontos de conexão para quaisquer DIPs que precisem ser conectados à energia ou ao aterramento.

Depois de colocar seus DIPs na breadboard, execute a energia e o aterramento em cada DIP seguindo estas etapas:

0. Passe um fio vermelho do terminal de +5 V da fonte para o barramento de energia.
1. Passe um fio vermelho do barramento de energia para o pino de força de cada DIP.
2. Passe um fio preto do terminal GND da fonte para o barramento de terra.
3. Passe um fio preto do barramento de terra até o pino de GND de cada chip.

A foto abaixo mostra dois DIPs colocados corretamente, cujos pinos de +5V e GND estão conectados aos barramentos.



Cores de Fios

- Use fios vermelhos para conexões à energia (+5 V).
- Use fios pretos para conexões ao terra.
- Use outras cores (não vermelhas ou pretas) para todas as outras conexões.

Em alguns cursos eletrônicos posteriores, você construirá circuitos que usam fontes de alimentação de +12 V e -12 V, além de +5 V. Para esses circuitos, use:

- Vermelho para +5 V
- Preto para terra
- Amarelo para +12 V
- Verde por −12 V

Comprimentos dos fios

À medida que você liga o circuito, mantenha os fios curtos e mantenha-os baixos contra a placa de proteção, não dando voltas no ar. Se você não conseguir encontrar um fio pré-cortado do comprimento correto, corte um para encaixar.

Endireitando e aparando extremidades de fio

Endireite a ponta descascada de cada fio antes de inseri-lo na protoboard. Os curvados têm uma tendência a se soltar. Se a ponta descascada é tão mal dobrado que você não pode endireitá-lo, corte-o e tire uma nova ponta.

Corte a ponta descascada de cada fio com o comprimento necessária para que, ao inseri-lo em um furo da placa de montagem, não exista metal exposto acima da placa de montagem.

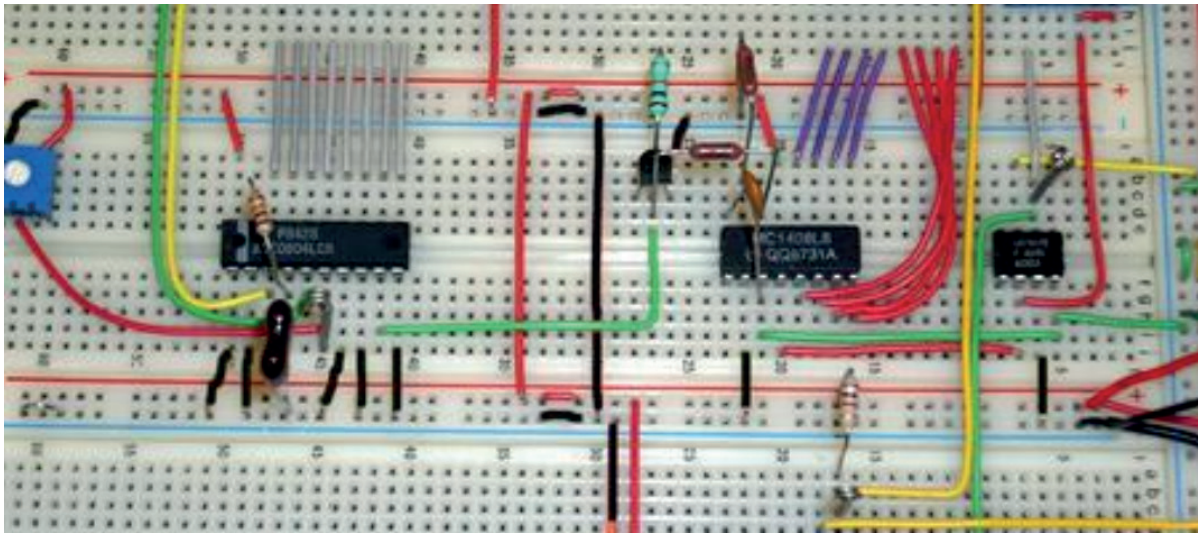
Fornecendo acesso ao DIP

Ao conectar seu circuito, assegure-se de ter acesso fácil aos pinos do DIP, para que possa tocá-los com uma sonda e para poder substituir o DIP sem desconectar os fios. Em particular:

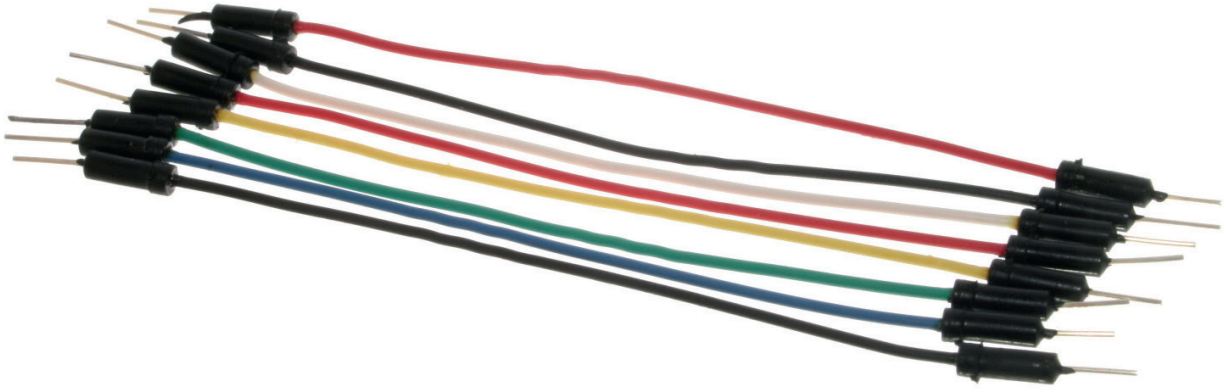
Nunca passe um fio por cima de um DIP. Em vez disso, direcione os fios ao redor do DIP.

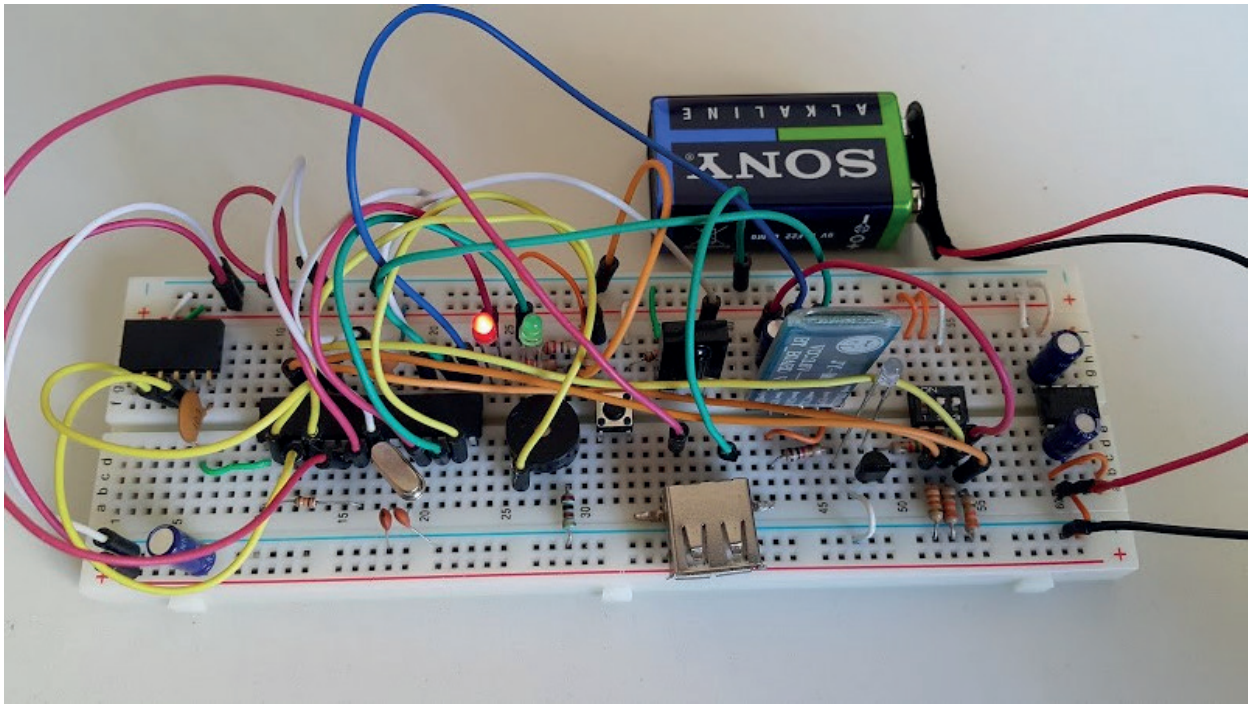
Quando você executar os fios em um DIP, use os furos da placa de montagem mais longe do DIP antes de usar os furos mais próximos.

A foto abaixo mostra um circuito corretamente conectado. Note que os fios são curtos, baixos e bem ordenados. Imagine com que facilidade você poderia substituir um dos DIPs sem desconectar quaisquer fios.



Outra opção para fios jumper é a utilização de cabos flexíveis pré-fabricados. São disponíveis em vários tamanhos, cores e possuem um borne para facilitar a inserção e extração do fio. Possuem uma conexão elétrica de melhor qualidade. Tem a vantagem de diminuir o tempo de montagem. A desvantagem fica no tamanho padronizado que pode fazer a montagem ficar mais “volumosa”.





Removendo um DIP

Não use os dedos para remover um DIP da breadboard. É muito fácil para os seus dedos deslizarem, fazendo com que o DIP se torça. Isso resulta em pinos tortos. Em vez disso, use um extrator de chips para puxar delicadamente o chip da placa. Ou insira uma chave de fenda fina no sulco abaixo do DIP e levante-o.

– Conversão de numeração

– Aritmética binária

– Portas lógicas

– Leis e teorias da álgebra boolena

– Implementação de funções booleanas com portas lógicas

– Circuitos lógicos combinacionais aritméticos

– Circuitos lógicos combinacionais: decodificadores e codificadores

– Circuitos lógicos combinacionais: multiplexadores e demultiplexadores

– Circuitos lógicos sequenciais: flip-flops

– Contadores

– Registradores de deslocamento