Projeto utilizando o Display de LED de 7 Segmentos GBK Robotics

O objetivo deste projeto é demonstrar a utilização de um display de led de 7 segmentos controlado diretamente a partir das portas digitais do Arduino.

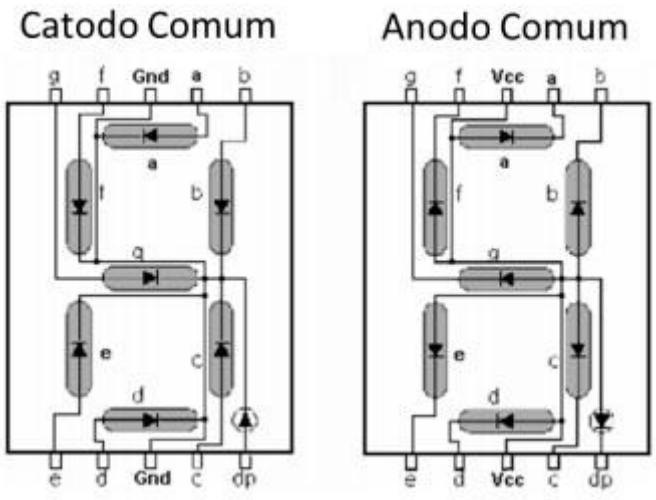
Material necessário:

1 Arduino
1 Display de Led de 7 Segmentos Catodo ou Anodo Comum (1 dígito)*
1 Resistor de 220 ohms (vermelho, vermelho, marrom)ou 330 ohms (laranja, laranja, marrom)*
1 Protoboard*
Jumper cable

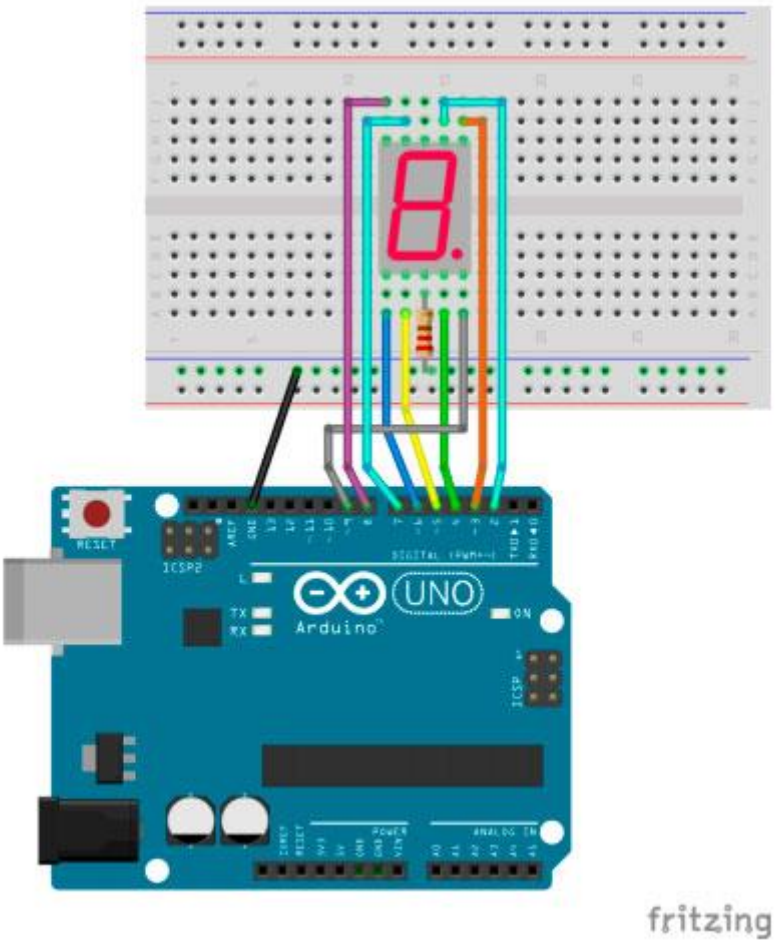
***Podem ser substituídos pelo módulo P11-Display Simples da GBK Robotics**

Passo1: Displays de Led de 7 Segmentos

Os displays de led de sete segmentos são bastante comuns e muitos utilizados para exibir, principalmente, informações numéricas
.
Podem ser de dois tipos:



Passo 2: Montagem do circuito (Projeto 1)



Acompanhar a sequência de montagem:

- **Pino 1 (segmento e) do display ligado ao 6 do Arduino;**
- **Pino 2 (segmento d) do display ligado ao 5 do Arduino;**
- **Pino 3 (Gnd, se catodo comum ou Vcc se anodo comum) do display ligado ao resistor de 220 ohms;**
- **Resistor de 220 ohms ligado ao Gnd (se catodo comum) ou Vcc (se anodo comum) do Arduino;**
- **Pino 4 (segmento c) do display ligado ao 4 do Arduino;**
- **Pino 5 (ponto decimal) do display ligado ao 9 do Arduino;**
- **Pino 6 (segmento b) do display ligado ao 3 do Arduino;**
- **Pino 7 (segmento a) do display ligado ao 2 do Arduino;**
- **Pino 9 (segmento f) do display ligado ao 7 do Arduino;**
- **Pino 10 (segmento g) do display ligado ao 8 do Arduino.**

Passo 3: Programa 1

```
int SEG_A = 2;
int SEG_B = 3;
int SEG_C = 4;
int SEG_D = 5;
int SEG_E = 6;
int SEG_F = 7;
int SEG_G = 8;
int PONTO = 9;
int ATRASO = 150;
void setup() {
for (int pino = SEG_A; pino <= SEG_G; pino++) {
pinMode(pino, OUTPUT);
}
}
void loop() {
for (int pino = SEG_A; pino < SEG_G; pino++) {
digitalWrite(pino, HIGH);
if (pino > SEG_A)
digitalWrite(pino - 1, LOW);
else
digitalWrite(SEG_F, LOW);
delay(ATRASO);
}
}
```

Passo 4: Programa 2

Inicie o ambiente de desenvolvimento do Arduino e digite o Sketch (programa) a seguir:

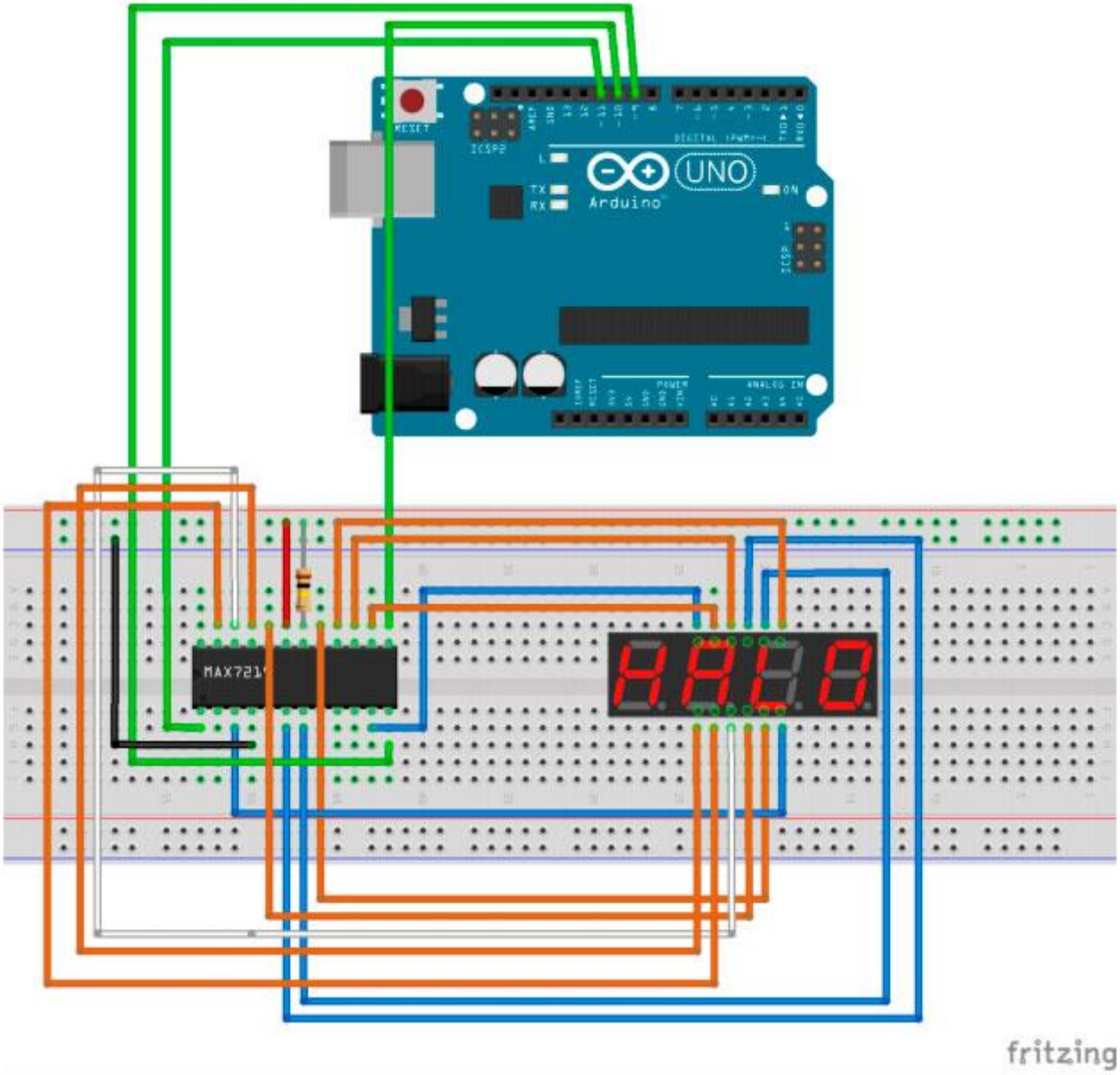
```
// Declaração da matriz para uso em displays com catodo comum, para utilizar em
displays com anodo comum basta inverter o valor dos bytes.
byte digitos[10][7] = {
{ 1,1,1,1,1,1,0 }, // = 0
{ 0,1,1,0,0,0,0 }, // = 1
{ 1,1,0,1,1,0,1 }, // = 2
{ 1,1,1,1,0,0,1 }, // = 3
{ 0,1,1,0,0,1,1 }, // = 4
{ 1,0,1,1,0,1,1 }, // = 5
{ 1,0,1,1,1,1,1 }, // = 6
{ 1,1,1,0,0,0,0 }, // = 7
{ 1,1,1,1,1,1,1 }, // = 8
{ 1,1,1,0,0,1,1 } // = 9
};
void setup() {
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
```

```
pinMode(5, OUTPUT);
pinMode(6, OUTPUT);
pinMode(7, OUTPUT);
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pontoDecimal(false);
}
void pontoDecimal(boolean ponto) {
digitalWrite(9, ponto);
}
void escrever(int digito) {
int pino = 2;
for (int segmento = 0; segmento < 7; segmento++) {
digitalWrite(pino, digitos[digito][segmento]);
pino++;
}
pontoDecimal(false);
}
void limpar() {
byte pino = 2;
for (int segmento = 0; segmento < 7; segmento++) {
digitalWrite(pino, LOW);
pino++;
}
}
void loop() {
for (int cont = 9; cont >= 0; cont--) {
escrever(cont);
boolean ponto = true;
for (int i = 0; i < 4; i++) {
delay(250);
pontoDecimal(ponto);
ponto = !ponto;
}
}
limpar();
delay(1000);
}
```

Passo 5: Uso de displays com múltiplos dígitos

Você já deve ter observado que quando precisamos utilizar displays de leds que apresentam mais do que um dígito, os portas disponíveis no Arduino não serão suficientes ou mesmo que sejam suficientes, não permitirão colocar novas funcionalidades ao seu projeto como, por exemplo, um sensor
de temperatura ou um módulo de relógio em tempo real (RTC).
Desta maneira, para otimizar o uso das portas do Arduino devemos utilizar um driver para displays de led sendo, o mais popular, o Maxim 7219 ou 7221, cuja pinagem pode ser observada a seguir.

Passo 6: Montagem do circuito (Projeto 2)



Passo 7: Programa 3

```
// Este sketch exibe o número 1234 em um display de led de 7 segmentos com 4 dígitos
// Utilizar a biblioteca LedControl
#include "LedControl.h"
/*
* Criar um LedControl (lc).
* O pino 11 do Arduino deve ser conectado ao pino DATA IN do primeiro MAX7219/21
```

```
* O pino 10 do Arduino deve ser conectado ao pino CLK do primeiro MAX7219/21
* O pino 9 do Arduino deve ser conectado ao pino LOAD (/CS) do primeiro MAX7219/21
* O quarto parâmetro indica que há apenas um MAX7219/21 conectado ao Arduino
*/
LedControl lc = LedControl(11, 10, 9, 1);
void setup()
{ // Retira o MAX7219/21 do modo de economia de energia
lc.shutdown(0, false);
// Define a intensidade do brilho dos leds
lc.setIntensity(0, 2);
lc.clearDisplay(0);
}
void loop()
{ lc.setChar(0, 2, '0', false);
lc.setChar(0, 2, '1', false);
lc.setChar(0, 3, '2', false);
lc.setChar(0, 4, '3', false);
}
```

Passo 8: Programa 4

```
/*
* Este sketch exibe os números inteiros entre -999 e 999 em um display de led de
* 7 segmentos com 4 dígitos
*/
// Utilizar a biblioteca LedControl
#include "LedControl.h"
/*
* Criar um LedControl (lc).
* O pino 11 do Arduino deve ser conectado ao pino DATA IN do primeiro MAX7219/21
* O pino 10 do Arduino deve ser conectado ao pino CLK do primeiro MAX7219/21
* O pino 9 do Arduino deve ser conectado ao pino LOAD (/CS) do primeiro MAX7219/21
* O quarto parâmetro indica que há apenas um MAX7219/21 conectado ao Arduino
*/
LedControl lc = LedControl(11, 10, 9, 1);
int i = -999;
void setup()
{ // Retira o MAX7219/21 do modo de economia de energia
lc.shutdown(0, false);
// Define a intensidade do brilho dos leds
lc.setIntensity(0, 2);
lc.clearDisplay(0);
}
void loop()
{ exibirInteiro(i++);
}
void exibirInteiro(int valor)
{
int unidade;
```

```
int dezena;
int centena;
boolean negativo = false;
if(valor < -999 || valor > 999)
return;
if(valor < 0)
{
negativo = true;
valor = valor * (-1);
}
unidade = valor % 10;
valor = valor / 10;
dezena = valor % 10;
valor =valor / 10;
centena = valor;
if (negativo)
{ // Imprime o sinal de negativo no display que está mais a esquerda
lc.setChar(0, 1, '-', false);
}
else
{ // Imprime um espaço no lugar do sinal de negativo
lc.setChar(0, 1 , ' ', false);
}
// Exibo o número dígito a dígito
lc.setDigit(0, 2, (byte)centena, false);
lc.setDigit(0, 3, (byte)dezena, false);
lc.setDigit(0, 4, (byte)unidade, false);
delay(100);
}
```

Pronto! É isso! Espero que este post tenha ajudado em seu projeto. Até o próximo post.