


# Display de 4 dígitos 7 Segmentos

ESTUDO E TESTE DAS SUAS CAPACIDADES

Nelson Proença | Display 4D – 7S | November 24, 2019

# Conteúdo

Display Led de 4 dígitos com 7 segmentos ..... 3
Estrutura do Display – Recursos ..... 3
Especificações: ..... 3
Esquema do Display ..... 3
Aplicação do Display ..... 4
Exemplo de teste do Display ..... 4
Material usado ..... 5
Circuito utilizado ..... 6
Código usado ..... 7
Explicação do código ..... 9
Conclusão ..... 11

# Display Led de 4 dígitos com 7 segmentos

## Estrutura do Display – Recursos

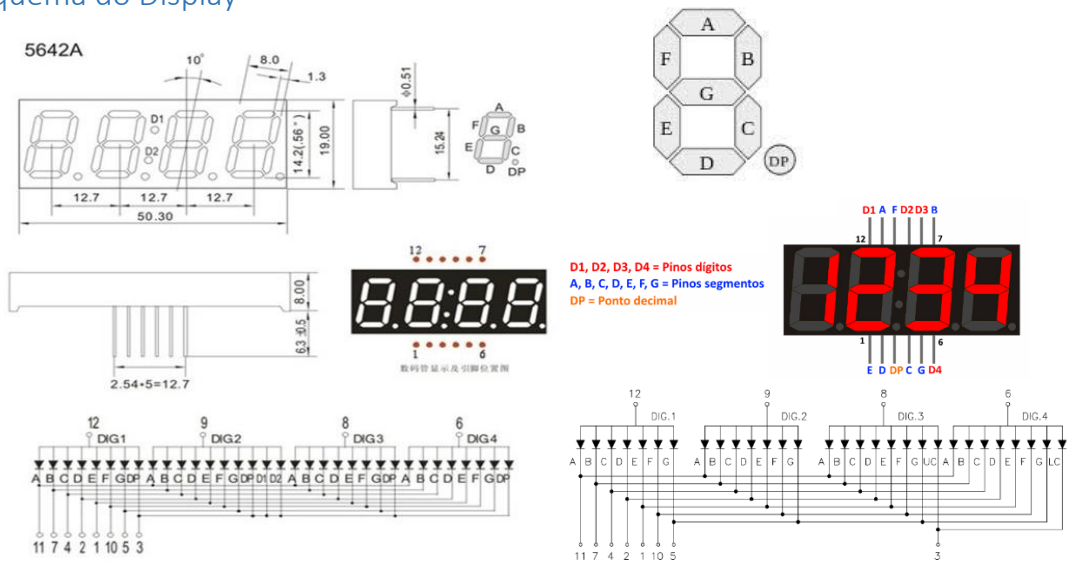
Este é um Display Led 7 Segmentos 4 dígitos, muito usado em projetos com eletrônica, contadores e relógios. O display pode ser anodo comum ou catode comum e possui LEDs na cor vermelha.

### Especificações:

- Cor LED: vermelha
- Anodo e/ou Catode Comum
- 12 pinos
- 4 dígitos
- Altura dígito: 1,4cm
- Dimensões: 5 x 1,9cm

Devido a falta de informação deste componente, teremos um estudo mais breve, a nível de características, mas consegue-se explorar as suas funcionalidades.

## Esquema do Display



## Aplicação do Display

Este display poder ser usado em projetos com eletrônica, contadores e relógios.

## Exemplo de teste do Display

Efetuou-se um teste a este Display, afim de verificar as suas características e perceber a sua aplicação.

Estudo do display 7 segmentos de 4 dígitos de cor vermelho, em conjunto com o Sensor LM35.
O display utilizado usa 4 dígitos e tem 12 pinos, recuperado de um recetor Tdt avariado:

- 4 dígitos - D1, D2, D3, D4 (12,9,8,6); são os números no display
- 7 segmentos - A, B, C, D, E, F, G (11,7,4,2,1,10,5); corresponde a cada linha que forma os números
- Ponto - DP (3); corresponde aos pontos centrais

A numeração dos pinos deste display começa parte inferior:




Para a ligação no Arduíno usam-se as portas 2 ate 9 para os segmentos e as portas 10 ao 13 para os dígitos.

*Leitura do Sensor LM35*.
Sensor de temperatura muito fiável, faz a leitura da temperatura ambiente.
Pelas pesquisas e estudos realizados, podemos usar uma fórmula para chegar a temperatura:

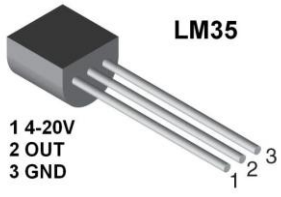
temperatura = ((valor lido do sensor) *(5/1023)) /10mv
(ou simplesmente multiplicar o valor lido por "0.488".)

ou seja:

1. Valor lido do sensor vem na porta A0, ao qual temos que multiplicar pela divisão de 5 por 1023, que por sua vez tem que se dividir por 10mv. (os 10mv correspondem a 1 grau C, que na programação tem a sua conversão para: 0.01)
2. Temos a variável "sensor" como "flat" para obter dados decimais, assim como a variável "temp".

Assim temos:

temp = sensor * 5 / (1023) / 0.01;




Esquema do circuito do LM35: pino 1 = Vcc+ / pino 2 = Sinal(A0) / pino 3 = GND
Para identificar os pinos virar a parte redonda para baixo.
Na ligação ao Arduíno utiliza uma porta analógica, por norma A0.

Foram realizados 3 estudos:

I. apenas impressão da leitura do sensor;
II. impressão da leitura do sensor e aplicação de palavras;
III. impressão de leitura e palavras com a biblioteca “SevSeg.h”, que já tem configurados os carateres.

No modelo com biblioteca "SevSeg.h", para facilitar a inserção dos carateres será necessário a instalação desta biblioteca, descompactar o ficheiro rar na pasta: ...\Documents\Arduino\libraries\

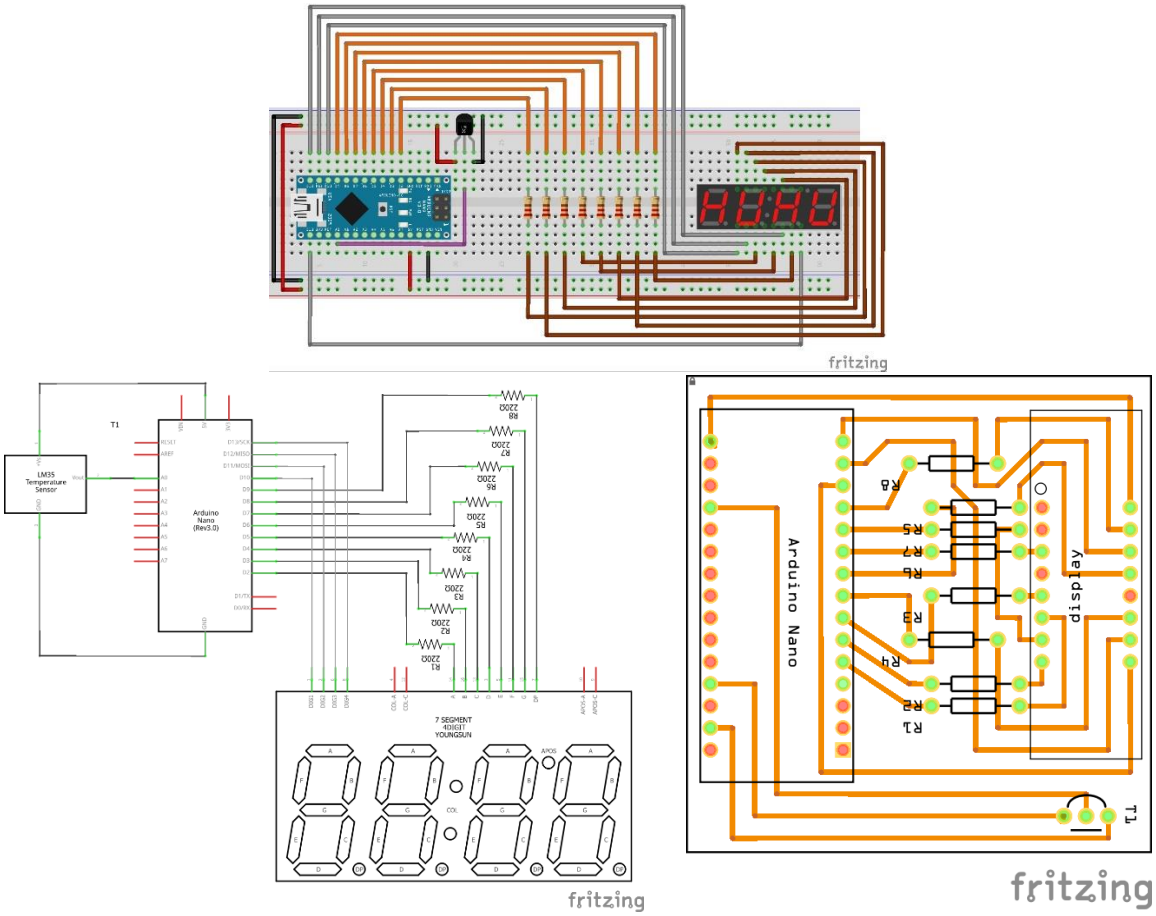
***Nota:*** Para que no display aparecesse o "º" (graus) foi necessário alterar a codificação do "#" no ficheiro "SevSeg.h"(...\Documents\Arduino\libraries\SevSeg-master\src\), para: 69|  0b1100011, // 35  '#'

Estudo realizado a 24 de novembro de 2019, efetuado por Nelson.

## Material usado

1- Arduíno Nano
2- Sensor LM35
3- Display
4- 8 resistências, de 220 Ω
5- Alguns fios de ligação
6- 1 Breadboard

## Circuito utilizado




Neste esquema podemos verificar que foi usado o Arduíno nano, ligado ao sensor e às resistências ligadas aos respetivos pinos do Display.

O Arduíno fornece a alimentação ao Lm35, e faz a gestão da energia nos pinos do Display.

A fonte de alimentação do Arduíno foi a própria porta Usb.

Assim, os pinos do display são ligados pelas portas do Arduíno:

- 2, 3, 4, 5, 6, 7, 8 e 9 para os segmentos (e as respetivas resistências);
- 10, 11, 12e 13 para os dígitos;

O sensor foi ligado a porta A0, afim de enviar os respetivos dados recolhidos.

O esquema de ligação do sensor e do display está exemplificado anteriormente, nas imagens.

## Código usado

Para tornar o esquema mais apelativo criou-se o seguinte código:

```
#include "SevSeg.h"
// identificar o nome do programa
SevSeg Display;
// variaveis gerais
unsigned long timer;
int deciSecond = 0;
char tempString[5];
float temp;
float sensor;
// identificar os pinos dos digitos (cathodes) a ligar no Arduino
int digit1 = 10;
int digit2 = 11;
int digit3 = 12;
int digit4 = 13;
// identificar os pinos dos segmentos (anodes) a ligar no Arduino
int segA = 2;
int segB = 3;
int segC = 4;
int segD = 5;
int segE = 6;
int segF = 7;
int segG = 8;
int segDP = 9;
int displayType = COMMON_CATHODE; // tipo de display: cathode ou anode
int numberOfDigits = 4; // identificar o numero de digitos do display (1,2,3,4,...)

void clear(void)                // configuracao para limpar o display
{
 // utilizar um carater nulo
 tempString[0] = 'm';
 tempString[1] = 'm';
 tempString[2] = 'm';
 tempString[3] = 'm';
 Display.DisplayString(tempString, 0);
 delay(5000);
}

void temperatura(void)     // configuracao da temperatura
{
 // formula para obter a temperatura.
 sensor = analogRead(A0);
 temp = sensor * 5 / (1023) / 0.01;
 // definicao de cada digito.
 tempString[0] = temp / 10;
 tempString[1] = temp - (tempString[0] * 10);
 tempString[2] = '#'; // este carater "#" corresponde ao "º", graus.
 tempString[3] = 'C';
 Display.DisplayString(tempString, 0);
 delay(10);
}

void setup()
{
 //Iniciar o display
 Display.Begin(displayType, numberOfDigits, digit1, digit2, digit3, digit4, segA, segB, segC, segD, segE, segF, segG, segDP);
 Display.SetBrightness(100);          // coloca o display com brilho 100%
 timer = millis();
}

void loop()
{
 // formula para obter a temperatura.
 sensor = analogRead(A0);
 temp = sensor * 5 / (1023) / 0.01;

 if (temp < 20)
 {
  char tempString[5] = {'O', 'L', 'A', ' '};

  for (int i = 0; i < 100; i++)
  {
   Display.DisplayString(tempString, 0);
   delay(10);
  }
  for (int i = 0; i < 100; i++)
  {
   tempString[0] = 'H';
   tempString[1] = 'O';
   tempString[2] = 'j';
   tempString[3] = 'E';
```

```
    Display.DisplayString(tempString, 0);
    delay(10);
   }
   for (int i = 0; i < 100; i++)
   {
    tempString[0] = 'E';
    tempString[1] = 'S';
    tempString[2] = 'T';
    tempString[3] = 'A';
    Display.DisplayString(tempString, 0);
    delay(10);
   }
   for (int i = 0; i < 100; i++)
   {
    tempString[0] = 'C';
    tempString[1] = 'O';
    tempString[2] = 'L';
    tempString[3] = 'D';
    Display.DisplayString(tempString, 0);
    delay(10);
   }
   for (int i = 0; i < 1000; i++)
   {
    temperatura();
   }
   clear();
  }
  else
  {
   char tempString[5] = {'O', 'L', 'A', ' '};
   for (int i = 0; i < 100; i++)
   {
    Display.DisplayString(tempString, 0);
    delay(10);
   }
   for (int i = 0; i < 100; i++)
   {
    tempString[0] = 'H';
    tempString[1] = 'O';
    tempString[2] = 'j';
    tempString[3] = 'E';
    Display.DisplayString(tempString, 0);
    delay(10);
   }
   for (int i = 0; i < 100; i++)
   {
    tempString[0] = 'E';
    tempString[1] = 'S';
    tempString[2] = 'T';
    tempString[3] = 'A';
    Display.DisplayString(tempString, 0);
    delay(10);
   }
   for (int i = 0; i < 100; i++)
   {
    tempString[0] = 'H';
    tempString[1] = 'O';
    tempString[2] = 'T';
    tempString[3] = ' ';
    Display.DisplayString(tempString, 0);
    delay(10);
   }
   for (int i = 0; i < 1000; i++)
   {
    temperatura();
   }
   clear();
  }

 }
```

## Explicação do código

1. Na parte inicial definem-se as portas, e as variáveis necessárias para o programa:

```
#include "SevSeg.h"
// identificar o nome do programa
SevSeg Display;
// variaveis gerais
unsigned long timer;
int deciSecond = 0;
char tempString[5];
float temp;
float sensor;
// identificar os pinos dos digitos (cathodes) a ligar no Arduino
int digit1 = 10;
int digit2 = 11;
int digit3 = 12;
int digit4 = 13;
// identificar os pinos dos segmentos (anodes) a ligar no Arduino
int segA = 2;
int segB = 3;
int segC = 4;
int segD = 5;
int segE = 6;
int segF = 7;
int segG = 8;
int segDP = 9;
int displayType = COMMON_CATHODE; // tipo de display: cathode ou anode
int numberOfDigits = 4; // identificar o numero de digitos do display (1,2,3,4,...)
```

2. De seguida, construímos funções para ajudar a configuração.
   Assim criamos as funções:
   a) “clear” para limpar o display, apaga os dígitos;
   b) “temperatura” irá imprimir a leitura do sensor, devidamente ajustada com a fórmula, que já foi explicado anteriormente.

   As funções podem ser configuradas mediante a opção pretendida.

```
void clear(void)            // configuracao para limpar o display
{
  // utilizar um carater nulo
  tempString[0] = 'm';
  tempString[1] = 'm';
  tempString[2] = 'm';
  tempString[3] = 'm';
  Display.DisplayString(tempString, 0);
  delay(5000);
}

void temperatura(void)    // configuracao da temperatura
{
  // formula para obter a temperatura.
  sensor = analogRead(A0);
  temp = sensor * 5 / (1023) / 0.01;
  // definicao de cada digito.
  tempString[0] = temp / 10;
  tempString[1] = temp - (tempString[0] * 10);
  tempString[2] = '#'; // este carater "#" corresponde ao "º", graus.
  tempString[3] = 'C';
  Display.DisplayString(tempString, 0);
  delay(10);
}
```

3. Iniciando o programa, no “setup” inicia-se o display e define-se as suas características, segundo as normas da biblioteca.
   Dá-se ajuste ao brilho e configura-se o “timer” para millis :

```
void setup()
{
 //Iniciar o display
 Display.Begin(displayType, numberOfDigits, digit1, digit2, digit3, digit4, segA, segB, segC, segD, segE, segF, segG, segDP);
 Display.SetBrightness(100);        // coloca o display com brilho 100%
 timer = millis();
}
```

4. Agora no “loop” inicia-se as repetições consecutivas, a configuração da temperatura que terá uma fórmula especifica, já explicada anteriormente. Aplica-se aqui uma condição com “if”, na medida em que se a temperatura for <20 imprime uma mensagem especifica e mostra a temperatura, caso contrário imprime a respetiva mensagem e a temperatura.
   Utiliza-se o “for” para controlar o tempo de permanência da mensagem no display e segue-se as normas da própria biblioteca para a construção do código e impressão dos dígitos.

```
void loop()
{
 // formula para obter a temperatura.
 sensor = analogRead(A0);
 temp = sensor * 5 / (1023) / 0.01;

 if (temp < 20)
 {
  char tempString[5] = {'O', 'L', 'A', ' '};

  for (int i = 0; i < 100; i++)
  {
   Display.DisplayString(tempString, 0);
   delay(10);
  }
  for (int i = 0; i < 100; i++)
  {
   tempString[0] = 'H';
   tempString[1] = 'O';
   tempString[2] = 'j';
   tempString[3] = 'E';

   Display.DisplayString(tempString, 0);
   delay(10);
  }
  for (int i = 0; i < 100; i++)
  {
   tempString[0] = 'E';
   tempString[1] = 'S';
   tempString[2] = 'T';
   tempString[3] = 'A';
   Display.DisplayString(tempString, 0);
   delay(10);
  }
  for (int i = 0; i < 100; i++)
  {
   tempString[0] = 'C';
   tempString[1] = 'O';
   tempString[2] = 'L';
   tempString[3] = 'D';
   Display.DisplayString(tempString, 0);
   delay(10);
  }
  for (int i = 0; i < 1000; i++)
  {
   temperatura();
  }
  clear();
 }
 else
 {
  char tempString[5] = {'O', 'L', 'A', ' '};
  for (int i = 0; i < 100; i++)
  {
   Display.DisplayString(tempString, 0);
   delay(10);
  }
  for (int i = 0; i < 100; i++)
  {
   tempString[0] = 'H';
   tempString[1] = 'O';
   tempString[2] = 'j';
   tempString[3] = 'E';
   Display.DisplayString(tempString, 0);
   delay(10);
  }
  for (int i = 0; i < 100; i++)
  {
   tempString[0] = 'E';
   tempString[1] = 'S';
   tempString[2] = 'T';
   tempString[3] = 'A';
   Display.DisplayString(tempString, 0);
   delay(10);
  }
  for (int i = 0; i < 100; i++)
  {
   tempString[0] = 'H';
   tempString[1] = 'O';
   tempString[2] = 'T';
   tempString[3] = ' ';
   Display.DisplayString(tempString, 0);
   delay(10);
  }
  for (int i = 0; i < 1000; i++)
  {
   temperatura();
  }
  clear();
 }

}
```

## Conclusão

- ✓ Estudo do display efetuado com sucesso, leituras registadas em diferentes ambientes, e impressão de mensagens com sucesso.
- ✓ Aprendizagem e construção de circuitos elétricos.
- ✓ Verificação e exploração de diferentes funções do código no Arduíno e funções especificas da biblioteca:
    - #include – insere no programa a biblioteca
    - unsigned long – variável de tipo de dados longos
    - int – variável inteira
    - float – variável para números decimais
    - analogRead(x) – faz a leitura analógica da respetiva porta
    - if (x) – verifica uma condição e executa os respetivos procedimentos
    - delay(x) – pausa especifica em milisegundos
    - Funções da biblioteca:
        - char tempString[5];
        - int displayType = COMMON_CATHODE;
        - int numberOfDigits = 4;
        - Display.DisplayString(tempString, 0);
        - Display.Begin(displayType, numberOfDigits, digit1, digit2, digit3, digit4, segA, segB, segC, segD, segE, segF, segG, segDP);
        - Display.SetBrightness(100);

*Objetivo concluído com sucesso.*

*Obrigado pela Atenção.*

Informações e especificações recolhidas no datasheet do componente.

Para mais informações pode verificar no datasheet o respetivo fabricante do componente.