

UNIVERSIDADE FEDERAL DE UBERLÂNDIA - UFU
FACULDADE DE ENGENHARIA ELÉTRICA – FEELT
SINAIS E SISTEMAS EM ENGENHARIA BIOMÉDICA

Utilização das Portas Digitais do Arduino

Alunos:

1. Ítalo Gustavo Sampaio Fernandes - 11511EBI004
2. Nicolle Ribeiro Vaz - 11511EBI014
3. Paulo Camargos Silva - 11611EBI023

Prof. Sérgio Ricardo de Jesus Oliveira

Uberlândia, **06** de **setembro** de 2017

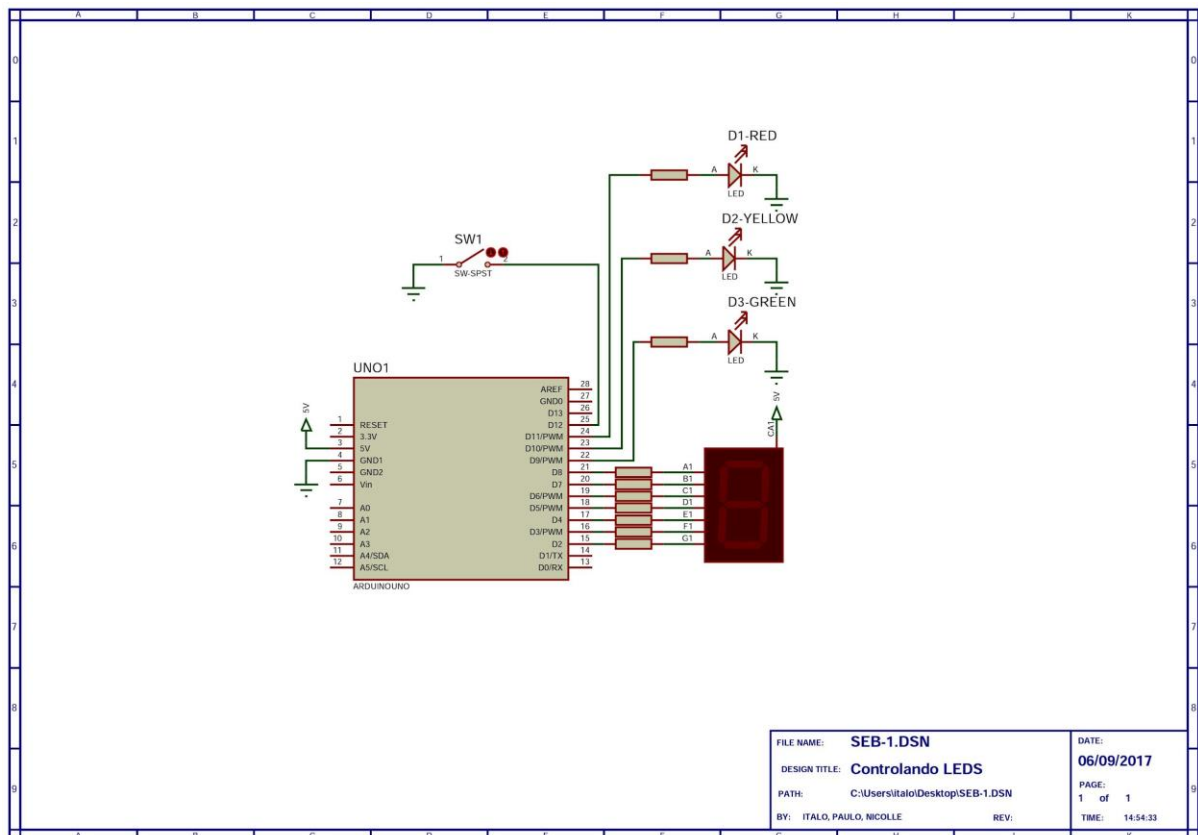
1 – Materiais

Os materiais utilizados para montagem dos circuitos eletrônicos Semáforo e Display 7 Segmentos estão listados abaixo:

- Arduino;
- 10 resistores de 150Ω ;
- 3 LEDs sendo: Verde, Vermelho e Amarelo;
- 1 display 7 segmentos ânodo comum;
- Matriz de contatos (protoboard);
- Multímetro.
-

* Obs.: Para o experimento, foi utilizado o Arduino Uno.

2 - Esquema Eletrônico Semáforo e Esquema Eletrônico Display 7 Segmentos



3 - Cálculos

Para obtenção do valor da resistência necessária para o funcionamento do circuito, foi efetuado o seguinte cálculo, baseado na folha de dados (*datasheet*) do Display de 7 Segmentos FND567.

Temos que:

- Tensão de trabalho típico $V_f = 1.7 \text{ V}$;
- Corrente direta $I_f = 20 \text{ mA}$;
- Tensão na porta do Arduino $V_{ard} = 3.3 \text{ V}$

$$V = \frac{R}{I}$$
$$R_{res} = \frac{3.3 - 1.7}{10 \times 10^{-3}}$$
$$R_{res} = 160 \, \Omega$$

O valor comercial de resistor mais próximo ao calculado é de $150 \, \Omega$ e foi utilizado neste experimento.

4 - Código Semáforo

O código para o Semáforo é apresentado no Anexo I ao fim do relatório. A primeira parte do código faz a criação de variáveis que armazenam o número de cada pino com seu LED correspondente. Dentro da função `void setup()`, os pinos são configurados de acordo com a rotina de trabalho: *input* ou *output*.

No interior da função `void loop()`, o código executa a rotina enquanto o Arduino estiver ligado. A estrutura de seleção `if()` verifica se o pino no qual o botão está conectado foi pressionado ou não. Caso não seja pressionado, a função `digitalRead(pinButton)` lê um valor lógico 1 no pino do botão (5V devido a configuração `INPUT_PULLUP`), executando a rotina normal de um semáforo: LED vermelho aceso por 3 s, LED verde por 2 s e LED amarelo por 1 s. Caso o botão seja pressionado, o valor lógico lido por `digitalRead(pinButton)` será 0, executando assim a rotina dentro de `else{}`: desligar os LEDs verde e vermelho e piscar o LED amarelo com intervalo de 0.5 s.

5 - Código Display

O código para o Display é apresentado no Anexo II. Primeiramente, o código faz a declaração das variáveis que serão utilizadas juntamente com a função `millis()` para controle do tempo que os segmentos serão

apresentados. A variável *tempo_controle* armazena o último tempo registrado quando a função *exibeNumero()* foi chamada. A variável *intervalo* é utilizada para verificação do tempo referência. Caso o intervalo de tempo transcorrido desde a última execução de *exibeNumero()* seja maior que *intervalo*, deve-se executar novamente a função *exibeNumero()*. Para que armazenassem valores grandes (pois estavam contando tempo), elas são do tipo *unsigned long*.

A matriz *display_7seg* do tipo *byte* é criada para armazenar o estado lógico de cada segmento do display, seguindo a ordem de acordo com a sequência e tipo no datasheet do Display. Como o Display é do tipo anodo, o valor lógico 0 ativará o segmento ao qual o pino do Arduino está conectado, e da mesma forma o valor lógico 1 desativará o segmento. Cada linha representa um número entre 0 e 9. A função *setup()* executa apenas uma vez (ao ligar o Arduino) e configura o regime de trabalho dos pinos 2 a 9 como *output*. Os pinos são montados na placa de acordo com a sequência A, B, ..., G.

A função *exibeNumero(byte numero)* é responsável por ligar os segmentos do display de acordo com o valor do parâmetro *numero* passado. Iniciando no pino 2 do Arduino, cada segmento é ligado individualmente dentro do laço *for{}* , até que todos os segmentos que configuram o valor do *numero* é concluído.

A função *loop()* é executada continuamente enquanto o Arduino estiver ligado. Dentro do laço *for{}* a variável *tempo_atual* recebe o valor de tempo da função *millis()* desde o instante que em que o programa começou a funcionar. A estrutura *if()* faz a verificação se o tempo transcorrido desde a última execução é maior que o intervalo estipulado. Caso seja verdade, a função *exibeTempo()* é chamada, passando o atual número a ser exibido como parâmetro.

6 - Funções nativas

millis() - Retorna o número de milissegundos desde que a placa Arduino começou a executar o programa atual. Esse número irá transbordar (voltar para zero), após aproximadamente 50 dias.

micros() - Retorna o número de microssegundos desde que a placa Arduino começou a executar o programa atual. Esse número irá transbordar (voltar para zero), após aproximadamente 70 minutos. Em placas Arduino de 16

MHz (por exemplo, Duemilanove e Nano), esta função tem uma resolução de quatro microssegundos (ou seja, o valor retornado é sempre um múltiplo de quatro). Em placas Arduino de 8 MHz (por exemplo, o LilyPad), esta função tem uma resolução de oito microssegundos.

delay(ms) - Pausa o programa pela quantidade de tempo (em milissegundos) especificado como parâmetro. (Existem 1000 milésimos de segundo em segundo.)

digitalRead(pin) - Lê o valor de um pino digital especificado, ALTO ou BAIXO.

digitalWrite(pin) - Escreva um valor ALTO ou BAIXO para um pino digital.

pinMode(pin, mode) - Configura o pino especificado para se comportar como entrada ou saída.

7 - Referências bibliográficas

<https://www.arduino.cc/en/Reference/DigitalWrite>

<https://www.arduino.cc/en/Reference/Millis>

<https://www.arduino.cc/en/Reference/DigitalRead>

<https://www.arduino.cc/en/Reference/PinMode>

<https://www.arduino.cc/en/Reference/Micros>

<https://www.arduino.cc/en/Reference/Delay>

ANEXO I - Código Semáforo

```
#define ledVermelho 11 // Definição dos pinos
#define ledAmarelo 10
#define ledVerde 9
#define pinButton 12

void setup() {
  // Definição do tipo de trabalho do pino: input ou output
  pinMode(ledVermelho, OUTPUT);
  pinMode(ledAmarelo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  pinMode(pinButton, INPUT_PULLUP);
}

void loop() {
  // Código para rotina de acendimento do semáforo
  if (digitalRead(pinButton)) { // Executa caso o botão não esteja
    pressionado
    digitalWrite(ledVermelho, 1); // Acende o ledVermelho
    delay(3000); // "Segura" o programa durante 3 segundos
    digitalWrite(ledVermelho, 0); // Desliga o ledVermelho
    digitalWrite(ledVerde, 1);
    delay(2000);
    digitalWrite(ledVerde, 0);
    digitalWrite(ledAmarelo, 1);
    delay(1000);
    digitalWrite(ledAmarelo, 0);
  } else { // Executa caso o botão esteja pressionado
    digitalWrite(ledVermelho, 0); //Desliga o ledVermelho
    digitalWrite(ledVerde, 0); // Desliga o ledVerde
    // Pisca o ledAmarelo por 0.5s
    digitalWrite(ledAmarelo, 1);
    delay(500);
    digitalWrite(ledAmarelo, 0);
    delay(500);
  }
}
```

ANEXO II - Código Display

```
// Definição das variáveis para controle do intervalo de tempo
unsigned long tempo_controle = 0; // Recebe o último tempo registrado
unsigned long intervalo = 300; // Intervalo de tempo para mudança do
número no display
byte numero;
/* Array para definição dos LEDs que serão ligados para cada número
// As linhas representam o número a ser exibido. As colunas
apresentam o estado (1 ou 0) de cada segmento:(A,B,C,D,E,F,G) */
byte display_7seg[10][7] = { { 0, 0, 0, 0, 0, 0, 1 }, // = Dígito 0
{ 1, 0, 0, 1, 1, 1, 1 }, // = Dígito 1
{ 0, 0, 1, 0, 0, 1, 0 }, // = Dígito 2
{ 0, 0, 0, 0, 1, 1, 0 }, // = Dígito 3
{ 1, 0, 0, 1, 1, 0, 0 }, // = Dígito 4
{ 0, 1, 0, 0, 1, 0, 0 }, // = Dígito 5
{ 0, 1, 0, 0, 0, 0, 0 }, // = Dígito 6
{ 0, 0, 0, 1, 1, 1, 1 }, // = Dígito 7
{ 0, 0, 0, 0, 0, 0, 0 }, // = Dígito 8
{ 0, 0, 0, 1, 1, 0, 0 }, // = Dígito 9
};

//Definição do tipo de trabalho de cada pino conectado ao display
void setup() {
  pinMode(2, OUTPUT); // seg A, pino 2 arduino, 7 (no display)
  pinMode(3, OUTPUT); // seg B, pino 3 arduino, 6 (no display)
  pinMode(4, OUTPUT); // seg C, pino 4 arduino, 4 (no display)
  pinMode(5, OUTPUT); // seg D, pino 5 arduino, 2 (no display)
  pinMode(6, OUTPUT); // seg E, pino 6 arduino, 1 (no display)
  pinMode(7, OUTPUT); // seg F, pino 7 arduino, 9 (no display)
  pinMode(8, OUTPUT); // seg G, pino 8 arduino, 10 (no display)
}

// Função para exibição do número no display
void exibeNumero(byte numero) {
  byte pino_atual = 2;
  for (byte i = 0; i < 7; ++i) {
    digitalWrite(pino_atual, display_7seg[numero][i]);
```

```

        ++pino_atual;
    }
}

void loop() {
    numero = 0; // Armazena o número para ser exibido de 0 a 9
    for (numero; numero <10;) {
        unsigned long tempo_atual = millis(); // Armazena o tempo atual
        if ((tempo_atual - tempo_controle) >= intervalo) { //Faz a verificação
se o intervalo foi 'estourado'
            tempo_controle = tempo_atual;          //Armazena o valor último
do intervalo estourado
            exibeNumero(numero);          // Chama a função para exibir o
número no display
            ++numero;                      // Incrementa o número
        }
    }
}

```