Guilherme Augusto Amorim Terrell 168899 Turma U

EA075 – Laboratório de Sistemas Embarcados
Prof. Eric Rohmer
Data de entrega: 12/04/2022

Projeto 1 – Semáforo com temporizadores e modo noturno

Questões de preparação

1-) Uma máquina de estados implementada em software pode ser representada através de uma função do tipo void maquinaEstados(int estado) que recebe como parâmetro um número inteiro correspondente a certo estado e, de acordo com o número recebido, será executado determinado conjunto de tarefas que caracteriza aquele estado. A transição entre os estados depende do estado atual da máquina e também do tempo que o código deve executar cada estado da máquina, por isso é preciso associar a variável que armazena o estado atual da máquina a uma variável contador que armazena o tempo que o código irá executar aquele estado e, transcorrido esse tempo, passa-se para o próximo estado.

2-) O circuito que acende um Led é composto por uma fonte de tensão contínua, um resistor limitador de corrente e o Led. O anodo do Led deve ser conectado ao terminal positivo da fonte de tensão e o catodo deve ser conectado ao terra. Como qualquer diodo, o Led só começa a conduzir corrente elétrica após a tensão em seus terminais ultrapassar a tensão de limiar (geralmente entre 0,5V e 0,7V). Após isso, o diodo passa a operar como uma chave fechada (modelo ideal), e a corrente que flui por seus terminais é dada por:

$$I_{diodo} = \frac{V_{fonte} - V_{limiar}}{R}$$

Pode-se perceber pela equação acima que, sem o resistor, a corrente que passa pelo diodo seria muito alta e levaria a queima do mesmo. A corrente do diodo não depende de sua cor, portanto tanto os Leds vermelho, amarelo e verde obedecerão a mesma equação. A corrente máxima suportada pelo diodo deve ser verificada e seu datasheet.

3-) É necessário implementar um debouncing por software que só habilita a execução do restante do código após decorrer um certo tempo após o botão ser pressionado. No caso desse projeto, por exemplo, foi implementado um contador chamado “delayDebouncing” na rotina de serviço do tratamento de interrupção do timer0 que, ao ser pressionado o botão, inicia-se o decremento do contador de 6 até 0 (que equivale a um tempo de 48ms). Somente após esse contador for zerado é que o restante do código pode ser implementado. Também é possível utilizar um sistema de hardware para fazer um debouncing através de um capacitor e um resistor, que juntos constituem um filtro passa baixa e eliminam os ruídos de alta frequência inseridos no sinal devido aos rebotes do botão.

4-) Os resistores de pull-up e pull down garantem que o pino do microcontrolador receba um valor de tensão conhecido e confiável quando o botão não estiver pressionado. Sem esses resistores o pino do microcontrolador perderia a referência e estaria sujeito a flutuações. No caso do resistor de pull up, o microcontrolador recebe nível lógico baixo quando o botão e pressionado e nível lógico alto quando o botão estiver solto, enquanto que no pull down, o pino do microcontrolador recebe nível alto quando o botão é pressionado e nível baixo quando o botão está solto.

5-) Deve-se fazer uso da multiplexação no tempo, onde apenas um dos displays ficará ligado por um determinado instante de tempo, de tal forma que apenas um dos displays esteja ligado por ciclo e que a frequência com qual ocorre o chaveamento entre os diplays seja da ordem de 30Hz, que da a impressão ao olho humano que todos os displays estão ligados simultaneamente.

Máquina de estados do semáforo durante o dia

Estado 0:

- Led vermelho dos carros desligado;
- Led amarelo dos carros desligado;
- Led verde dos carros ligado;
- Led vermelho para pedestres ligado;
- Led verde dos pedestres desligado;
- Display dos carros desligado;
- Display dos pedestres desligado;

Estado 1:

- Led vermelho dos carros apagado;
- Led amarelo dos carros ligado;
- Led verde dos carros apagado;
- Led vermelho dos pedestres ligado;
- Led verde dos pedestres apagado;
- Display dos carros desligado;
- Display dos pedestres desligado;

Estado 2:

- Led vermelho para carros ligado;
- Led amarelo dos carros apagado;
- Led verde dos carros apagado;
- Led vermelho dos pedestres apagado
- Led verde para pedestres ligado;
- Display dos carros ligado decrementando o tempo de 9 segundos;
- Display dos pedestres ligado decrementando o tempo de 5 segundos;

Estado 3:

- Led vermelho dos carros ligado;
- Led amarelo dos carros desligado;
- Led verde dos carros desligado;
- Led vermelho para pedestres ligado;
- Led verde dos pedestres desligado;
- Display dos carros ligado decrementando o tempo de 9 segundos;
- Display dos pedestres ligado mostrando o tempo de 0 segundos;

Estado 4:

- Led vermelho dos carros ligado;
- Led amarelo dos carros desligado;
- Led verde dos carros desligado;
- Led vermelho dos pedestres desligado;
- Led verde dos pedestres desligado;
- Display dos carros ligado decrementando o tempo de 9 segundos;
- Display dos pedestres desligado;

Obs: A combinação dos estados 3 e 4 produzem o efeito piscante do led vermelho e do display dos pedestres quando o tempo de travessia chega a zero.

Máquina de estados do semáforo durante a noite

Estado 5:

- Led vermelho dos carros desligado;
- Led amarelo dos carros ligado;
- Led verde dos carros desligado;
- Led vermelho dos pedestres ligado;
- Led verde dos pedestres desligado;
- Display dos carros desligado;
- Display dos pedestres desligado;

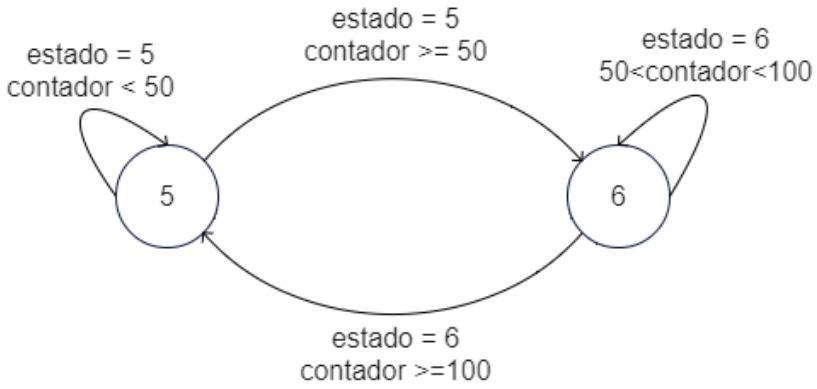
Estado 6:

- Led vermelho dos carros desligado;
- Led amarelo dos carros desligado;
- Led verde dos carros desligado;
- Led vermelho dos pedestres desligado;
- Led verde dos pedestres desligado;
- Display dos carros desligado;
- Display dos pedestres desligado;

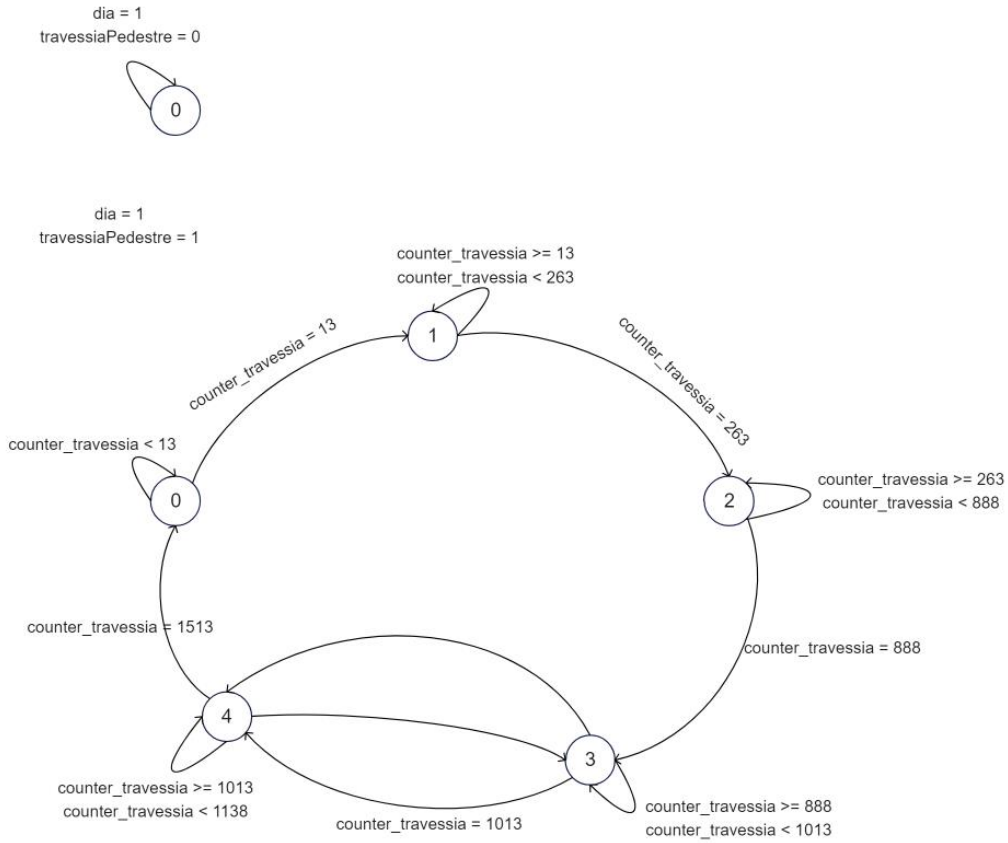
Obs: A combinação dos estados 5 e 6 produz o efeito piscante dos leds amarelo dos carros e vermelho dos pedestres.

Fluxograma máquina de estados

Modo noturno:



Modo diurno:



Obs: Quanto a contagem de tempo do display dos pedestres chegar a zero haverá alternância de estados entre 3 e 4 até que a contagem dos carros também chegue a zero (que ocorre quando counter_travessia atinge a contagem de 1513) e o ciclo de travessia de pedestres se encerra, voltando a máquina de estados a ficar no estado 0.

- Pseudo-código

Se estiver de noite (dia = 0):

Se contador <= 50:

Estado 5;

Se contador > 50;

Estado 6;

contador = 0;

Se estiver de dia e não houver pedestre (dia = 1 E travessiaPedestre = 0):

Estado 0;

Se estiver de dia e houver pedestre (dia = 1 E travessiaPedestre = 1):

//Esperar 100ms -> contador_travessia = 13

Se ((contador_travessia >= 13) && (contador_travessia < 263)):

//263 – 13 = 250 -> 2 segundos

Estado 1;

Se ((contador_travessia >= 263) && (contador_travessia < 388)):

//263 – 13 = 250 -> 2 segundos

//display dos carros em 9 segundos

//display dos pedestres em 5 segundos

Estado 2;

Se ((contador_travessia >= 388) && (contador_travessia < 513)):

//display dos carros em 8 segundos

//display dos pedestres em 4 segundos

Estado 2;

Se ((contador_travessia >= 513) && (contador_travessia < 638)):

//display dos carros em 7 segundos

//display dos pedestres em 3 segundos

Estado 2;

Se ((contador_travessia >= 638) && (contador_travessia < 763)):

//display dos carros em 6 segundos

//display dos pedestres em 2 segundos

Estado 2;

```
Se ((contador_travessia >= 763) && (contador_travessia < 888)):
        //display dos carros em 5 segundos
        //display dos pedestres em 1 segundos
        Estado 2;
Se ((contador_travessia >= 888) && (contador_travessia < 1013)):
        //display dos carros em 4 segundos
        //display dos pedestres em 0 segundos
        Estado 3;
Se ((contador_travessia >= 1013) && (contador_travessia < 1138)):
        //display dos carros em 3 segundos
        //display dos pedestres em 0 segundos
        Estado 4;
Se ((contador_travessia >= 1138) && (contador_travessia < 1263)):
        //display dos carros em 2 segundos
        //display dos pedestres em 0 segundos
        Estado 3;
Se ((contador_travessia >= 1263) && (contador_travessia < 1388)):
        //display dos carros em 1 segundos
        //display dos pedestres em 0 segundos
        Estado 4;
Se ((contador_travessia >= 1388) && (contador_travessia < 1513)):
        //display dos carros em 0 segundos
        //display dos pedestres em 0 segundos
        Estado 3;
Se (contador_travessia >= 1513):
        //Resetar as flags e os contadores
        //Retorno da máquina de estados ao estado 0
        travessiaPedestre = 0
        conter_travessia = 0
        delayDebouncing = 6
```

O projeto consiste em um semáforo inteligente com dois modos de operação: noturno e diurno.

No modo noturno não há leitura do estado do botão, e ocorre apenas o padrão piscante dos leds dos carros e do vermelho para os pedestres. A interação da placa com o meio externo ocorre apenas através do LDR, cujos valores são continuamente lidos.

No modo diurno é adicionado mais um meio de interagir com o ambiente, que é o botão de pedestres. A leitura do pino onde está conectado esse botão é feita continuamente (assim como o pino do LDR). Se o botão não for pressionado, mantém-se o estado padrão: Verde para os carros aceso, vermelho para os pedestres apagado. Caso o botão dos pedestres seja apertado, inicia-se o processo de travessia de pedestres descrito no projeto da máquina de estados.

No código existem 4 contadores responsáveis pelo controle de tempo das tarefas do programa, sendo que todos eles são incrementados ou decrementados na função que trata a interrupção no timer0 do microcontrolador. Os contadores utilizados são:

- counter: Responsável pelo padrão piscante dos leds amarelo dos carros e vermelho dos pedestres durante a noite (só é incrementado quando a flag dia=0, que equivale a ser noite).

- counter_travessia: Controla o tempo de todo o processo de travessia dos pedestres (só é incrementado quando a flag dia=1 e a flag travessiaPedestre=1). Também é utilizada no controle do acionamento do displays da seguinte forma: Se counter_travessia%2 for 1, aciona-se o display dos carros; Se counter_travessia%2 for 0, aciona-se o display dos pedestres.

- delayDebouncing: Inicia o decremento logo que o botão de pedestres é pressionado (flag botaoPressionado = 1), e é responsável por aplicar uma espera de 48ms entre o pressionamento do botão e a execuação do restante do código.

- timerDelay: É incrementado sempre que for detectado variações na leitura do LDR. Esse contador introduz um tempo de espera 5 segundos entre a mudança no valor do LDR e a mudança do valor da flag dia. Para efeitos práticos, esse contador é responsável por evitar que um carro que passe pelo semáforo a noite com os faróis acesos altere o padrão de funcionamento do semáforo de noturno para diurno.

No código existem também 3 flags que habilitam ou desabilitam determinados comportamentos especificados no projeto. São elas:

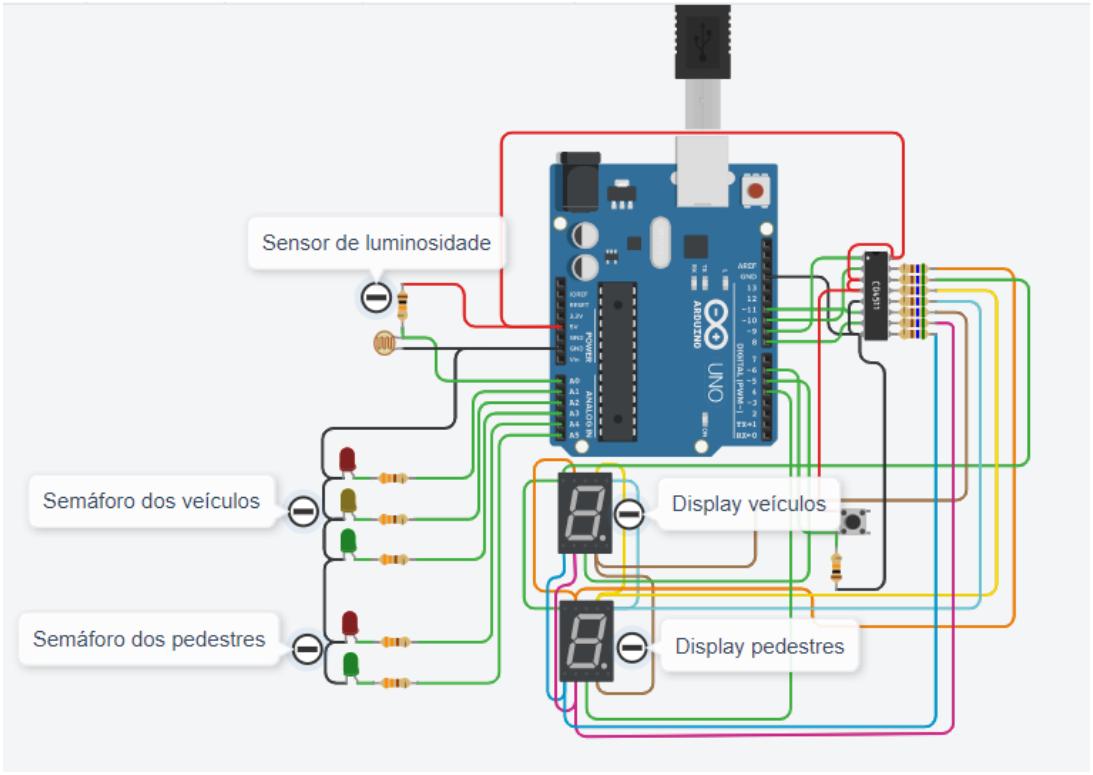
- dia: Flag que está atrelada a leitura do LDR e indica se está de dia (dia = 1) ou se é noite (dia = 0). Através dessa flag é condicionado o padrão noturno ou diurno do semáforo.

- botaoPressionado: Flag que está atrelada a leitura do botão, e indica se o mesmo foi pressionado (botaoPressionado = 1) ou se não foi pressionado (botaoPressionado = 0).

- travessiaPedestre: Flag que indica que o pedestre está atravessando e habilita o funcionamento do semáforo (transições de estado). A combinação entre as flags dia=1 e botaoPressioando=1 alteram o valor de travessiaPedestre para 1.

- Circuito

O circuito é o mesmo fornecido pelos professores. Não houve nenhuma modificação quanto ao hardware utilizado.



- Lista de componentes

Nome	Quantidade	Componente
Digit1	2	Catódica Visor de sete segmentos
Digit2		
U1	1	Arduino Uno R3
U2	1	Decodificador de sete segmentos
R3	7	560 Ω Resistor
R4		
R5		
R6		
R7		
R8		
R9		
R1	1	Fotorresistor
D1	2	Vermelho LED
D4		
D2	1	Amarelo LED
D3	2	Verde LED
D5		
R10	5	330 Ω Resistor
R11		
R12		
R13		
R14		
S1	1	Botão
R15	2	10 kΩ Resistor
R2		

- Principais dificuldades

Como pode ser observado no código do projeto, os decrementos no tempo de travessia dos pedestres e dos carros foi feito explicitamente, isto é, foi declarado um estado diferente para cada instante de tempo. Para uma contagem pequena de apenas 10 segundos como a desse projeto isso talvez não seja um problema, mas em projetos maiores implementar essa mesma lógica seria uma tarefa difícil. Como melhoria pode-se propor um algoritmo mais iterativo, que decremente sozinho os tempos de cada etapa.

- Link do vídeo no YouTube

https://www.youtube.com/watch?v=4XkHU-o27I4

- Código

```
#define pinLDR A0
#define pinLedVermelhoCarro A1
#define pinLedAmareloCarro A2
#define pinLedVerdeCarro A3
#define pinLedVermelhoPed A4
#define pinLedVerdePed A5
#define pinBotao 6
#define displayCarro 5
#define displayPed 4
#define pinInput1 8
#define pinInput2 9
#define pinInput3 10
#define pinInput4 11

int dia = 0;/*Flag que indica se é dia ou noite*/
int valorLDR = 0; /*Valor que armazena a leitura do LDR*/
bool valorBotao;/*Variável que armazena o estado lógico do botão*/
bool botaoPressionado = 0; /*Flag que indica se o botão foi pressioando*/
int counter = 0; /*Variável utilizada a noite para padrão piscante dos leds amarelo e vemelho*/
int counter_travessia = 0; /*Variável que controla o tempo de travessia dos pedestres*/
int travessiaPedestre = 0; /*Flag que habilita a travessia dos pedestres*/
```

```
int displayCarroOn = 0;/*Flag que indica qual display deve ser ligado*/
int timerDelay = 0;/*Variável que introduz um delay entre o acionamento dos displays*/
int delayDebouncing = 6;/*6 equivale a 48ms*/

void configuracao_Timer0(){
 ////////////////////////////////////////////////////////////////////////////////////////////////////
 // Configuracao Temporizador 0 (8 bits) para gerar interrupcoes periodicas a cada 8ms no modo Clear Timer on Compare Match (CTC)
 // Relogio = 16e6 Hz
 // Prescaler = 1024
 // Faixa = 125 (contagem de 0 a OCR0A = 124)
 // Intervalo entre interrupcoes: (Prescaler/Relogio)*Faixa = (64/16e6)*(124+1) = 0.008s

 // TCCR0A – Timer/Counter Control Register A
 // COM0A1 COM0A0 COM0B1 COM0B0 – – WGM01 WGM00
 // 0     0     0     0        1     0
 TCCR0A = 0x02;

 // OCR0A – Output Compare Register A
 OCR0A = 124;

 // TIMSK0 – Timer/Counter Interrupt Mask Register
 // – – – – – OCIE0B OCIE0A TOIE0
 // – – – – – 0     1     0
 TIMSK0 = 0x02;

 // TCCR0B – Timer/Counter Control Register B
 // FOC0A FOC0B – – WGM02 CS02 CS01 CS0
 // 0    0       0    1   0   1
 TCCR0B = 0x05;
 ////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
}

// Rotina de servico de interrupcao do temporizador0
ISR(TIMER0_COMPA_vect){
 // Insira aqui o codigo a s do temporizadorer executado pela interrupcao periodica
 if(valorLDR >= 100){
  if(dia == 1){
   timerDelay++;
   if(timerDelay > 625){/*Esperar 5s antes de mudar a variável dia*/
    dia = 0;
    /*Zerar os contadores e a flag de travessia dos pedestres*/
    /*Resetar o valor de delayDebouncing para permitir nova leitura do botao*/
    counter_travessia = 0;
    timerDelay = 0;
    travessiaPedestre = 0;
    delayDebouncing = 6;
    botaoPressionado = 0;
   }
  }
  else{
   dia = 0;
   travessiaPedestre = 0;
  }
 }
 if(valorLDR < 100){
  if(dia == 0){
   timerDelay++;
   if(timerDelay > 625){/*Esperar 5s antes de mudar a variável dia*/
    dia = 1;
    /*resetar os contadores*/
    timerDelay = 0;
```

```
      counter = 0;
    }
  }
  else{
    dia = 1;
  }
}
if(dia == 0){
  travessiaPedestre = 0;
  counter++;
  /*Padrão piscante de 400ms aceso e 400ms apagado*/
  if(counter > 100){
    counter = 0;
  }
}
if(dia == 1){
  if(valorBotao == 1){
    /*flag que indica se o botão pressionado*/
    botaoPressionado = 1;
  }
  else{
    botaoPressionado = 0;
  }
  /*Debouncing do botão*/
  if((botaoPressionado == 1) && (delayDebouncing > 0)){
    delayDebouncing--;
  }
  if((botaoPressionado == 1) && (delayDebouncing == 0)){
    travessiaPedestre = 1;/*Flag de travessia para pedestres*/
  }
  if(travessiaPedestre == 1){
```

```
      counter_travessia++;
      /*Se counter_travessia for impar -> liga o display dos carros*/
      if(counter_travessia%2 == 1){
        displayCarroOn = 1;
      }
      /*Se counter_travessia for par -> liga o display dos pedestres*/
      if(counter_travessia%2 == 0){
        displayCarroOn = 0;
      }
    }
  }
}

void setup(){
  pinMode(pinLDR, INPUT);
  pinMode(pinBotao, INPUT);
  pinMode(pinLedVermelhoCarro, OUTPUT);
  pinMode(pinLedAmareloCarro, OUTPUT);
  pinMode(pinLedVerdeCarro, OUTPUT);
  pinMode(pinLedVermelhoPed, OUTPUT);
  pinMode(pinLedVerdePed, OUTPUT);
  pinMode(displayCarro, OUTPUT);
  pinMode(displayPed, OUTPUT);
  pinMode(pinInput1, OUTPUT);
  pinMode(pinInput2, OUTPUT);
  pinMode(pinInput3, OUTPUT);
  pinMode(pinInput4, OUTPUT);
  /*Displays e LEDs iniciam desligados*/
  analogWrite(pinLedVermelhoCarro, 0);
  analogWrite(pinLedAmareloCarro, 0);
  analogWrite(pinLedVerdeCarro, 0);
```

```
  analogWrite(pinLedVermelhoPed, 0);
  analogWrite(pinLedVerdePed, 0);
  digitalWrite(displayCarro, HIGH);
  digitalWrite(displayPed, HIGH);
  cli();
  configuracao_Timer0();
  sei();
}

/*Função que recebe um numero entre 0 e 9 e o escreve no display*/
/*Cada case do switch é um número*/
void escreveDisplay(int tempo){
  switch (tempo){
    case 0:
      digitalWrite(pinInput1, LOW);
      digitalWrite(pinInput2, LOW);
      digitalWrite(pinInput3, LOW);
      digitalWrite(pinInput4, LOW);
    break;

    case 1:
      digitalWrite(pinInput1, HIGH);
      digitalWrite(pinInput2, LOW);
      digitalWrite(pinInput3, LOW);
      digitalWrite(pinInput4, LOW);
    break;

    case 2:
      digitalWrite(pinInput1, LOW);
      digitalWrite(pinInput2, HIGH);
      digitalWrite(pinInput3, LOW);
```

```
  digitalWrite(pinInput4, LOW);
break;

case 3:
  digitalWrite(pinInput1, HIGH);
  digitalWrite(pinInput2, HIGH);
  digitalWrite(pinInput3, LOW);
  digitalWrite(pinInput4, LOW);
break;

case 4:
  digitalWrite(pinInput1, LOW);
  digitalWrite(pinInput2, LOW);
  digitalWrite(pinInput3, HIGH);
  digitalWrite(pinInput4, LOW);
break;

case 5:
  digitalWrite(pinInput1, HIGH);
  digitalWrite(pinInput2, LOW);
  digitalWrite(pinInput3, HIGH);
  digitalWrite(pinInput4, LOW);
break;

case 6:
  digitalWrite(pinInput1, LOW);
  digitalWrite(pinInput2, HIGH);
  digitalWrite(pinInput3, HIGH);
  digitalWrite(pinInput4, LOW);
break;
```

```
    case 7:
      digitalWrite(pinInput1, HIGH);
      digitalWrite(pinInput2, HIGH);
      digitalWrite(pinInput3, HIGH);
      digitalWrite(pinInput4, LOW);
    break;

    case 8:
      digitalWrite(pinInput1, LOW);
      digitalWrite(pinInput2, LOW);
      digitalWrite(pinInput3, LOW);
      digitalWrite(pinInput4, HIGH);
    break;

    case 9:
      digitalWrite(pinInput1, HIGH);
      digitalWrite(pinInput2, LOW);
      digitalWrite(pinInput3, LOW);
      digitalWrite(pinInput4, HIGH);
    break;
  }
}

/*Maquina de estados dos semáforos dos carros e dos pedestres*/
void maquinaEstados(int estado){
  switch (estado){
    case 0:
      /*Inicia com verde para carros e vermelho para pedestre*/
      /*Display carros desligado*/
      /*Display pedestres desligado*/
      analogWrite(pinLedVermelhoCarro, 0);
```

```
  analogWrite(pinLedAmareloCarro, 0);
  analogWrite(pinLedVerdeCarro, 255);
  analogWrite(pinLedVermelhoPed, 255);
  analogWrite(pinLedVerdePed, 0);
  digitalWrite(displayCarro, HIGH);
  digitalWrite(displayPed, HIGH);
 break;

 case 1:
  /*Amarelo para carros aceso*/
  /*Vermelho para pedestres aceso*/
  analogWrite(pinLedVermelhoCarro, 0);
  analogWrite(pinLedAmareloCarro, 255);
  analogWrite(pinLedVerdeCarro, 0);
  analogWrite(pinLedVermelhoPed, 255);
  analogWrite(pinLedVerdePed, 0);
 break;

 case 2:
  /*Vermelho para carros aceso*/
  /*Verde para pedestres aceso*/
  /*Display carros ligado contando o tempo*/
  /*Display pedestre ligado contando o tempo*/
  analogWrite(pinLedVerdeCarro, 0);
        analogWrite(pinLedVermelhoCarro, 255);
        analogWrite(pinLedAmareloCarro, 0);
        analogWrite(pinLedVerdePed, 255);
        analogWrite(pinLedVermelhoPed, 0);
 break;

case 3:
```

```
   /*Vermelho para carros aceso*/
   /*Vermelho para pedestres aceso*/
   /*Display carros ligado contando o tempo*/
   /*Display pedestre ligado contando o tempo*/
   analogWrite(pinLedVerdeCarro, 0);
         analogWrite(pinLedVermelhoCarro, 255);
         analogWrite(pinLedAmareloCarro, 0);
         analogWrite(pinLedVerdePed, 0);
         analogWrite(pinLedVermelhoPed, 255);
  break;

case 4:
   /*Vermelho para carros aceso*/
   /*Vermelho para pedestres apagado*/
   /*Display carros ligado contando o tempo*/
   /*Display pedestre ligado contando o tempo*/
   analogWrite(pinLedVerdeCarro, 0);
         analogWrite(pinLedVermelhoCarro, 255);
         analogWrite(pinLedAmareloCarro, 0);
         analogWrite(pinLedVerdePed, 0);
         analogWrite(pinLedVermelhoPed, 0);
  break;

case 5:
   /*Amarelo para carros aceso*/
   /*Vermelho para pedestres aceso*/
   /*Display carros desligado*/
   /*Display pedestre desligado*/
   analogWrite(pinLedVerdeCarro, 0);
         analogWrite(pinLedVermelhoCarro, 0);
         analogWrite(pinLedAmareloCarro, 255);
```

```
            analogWrite(pinLedVerdePed, 0);
            analogWrite(pinLedVermelhoPed, 255);
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, HIGH);
   break;

  case 6:
    /*Amarelo para carros apagado*/
    /*Vermelho para pedestres apagado*/
    /*Display carros desligado*/
    /*Display pedestre desligado*/
    analogWrite(pinLedVerdeCarro, 0);
            analogWrite(pinLedVermelhoCarro, 0);
            analogWrite(pinLedAmareloCarro, 0);
            analogWrite(pinLedVerdePed, 0);
            analogWrite(pinLedVermelhoPed, 0);
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, HIGH);
   break;
  }
}

void loop(){
  /*delay(1) foi inserido apenas para acelerar a simulação*/
  /*Não tem interferência na operação do semáforo*/
  delay(1);
  valorLDR = analogRead(pinLDR);/*Leitura do LDR*/
  valorBotao = digitalRead(pinBotao);/*Leitura do botão*/
  /*Funcionamento durante a noite*/
  /*Padrão piscante dos Leds*/
  if(dia == 0){
```

```
  if(counter <= 50){
    maquinaEstados(5);
  }
  if((counter > 50) && (counter <=100)){
    maquinaEstados(6);
  }
}

/*Funcionamento durante o dia quando não há pedestres*/
if((dia == 1) && (travessiaPedestre == 0)){
  maquinaEstados(0);
}

/*Funcionamento durante o dia quando há pedestres*/
if((dia == 1) && (travessiaPedestre == 1)){
  /*Esperar 100ms (counter_travessia ~= 13) antes de acender o amarelo para carros*/
  if((counter_travessia >= 13) && (counter_travessia < 263)){
    maquinaEstados(1);
  }
  if((counter_travessia >= 263) && (counter_travessia < 388)){
    /*388 - 263 = 125 -> 1 segundos*/
    maquinaEstados(2);
    if(displayCarroOn == 1){
      digitalWrite(displayPed, HIGH);
      digitalWrite(displayCarro, LOW);
      escreveDisplay(9);
    }
    if(displayCarroOn == 0){
      digitalWrite(displayCarro, HIGH);
      digitalWrite(displayPed, LOW);
      escreveDisplay(5);
```

```
  }
 }
 if((counter_travessia >= 388) && (counter_travessia < 513)){
  /*513 - 263 = 250 -> 2 segundos*/
  maquinaEstados(2);
  if(displayCarroOn == 1){
    digitalWrite(displayPed, HIGH);
    digitalWrite(displayCarro, LOW);
    escreveDisplay(8);
  }
  if(displayCarroOn == 0){
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, LOW);
    escreveDisplay(4);
  }
 }
 if((counter_travessia >= 513) && (counter_travessia < 638)){
  /*638 - 263 = 375 -> 3 segundos*/
  maquinaEstados(2);
  if(displayCarroOn == 1){
    digitalWrite(displayPed, HIGH);
    digitalWrite(displayCarro, LOW);
    escreveDisplay(7);
  }
  if(displayCarroOn == 0){
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, LOW);
    escreveDisplay(3);
  }
 }
 if((counter_travessia >= 638) && (counter_travessia < 763)){
```

```
  /*763 - 263 = 500 -> 4 segundos*/
  maquinaEstados(2);
  if(displayCarroOn == 1){
    digitalWrite(displayPed, HIGH);
    digitalWrite(displayCarro, LOW);
    escreveDisplay(6);
  }
  if(displayCarroOn == 0){
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, LOW);
    escreveDisplay(2);
  }
 }
 if((counter_travessia >= 763) && (counter_travessia < 888)){
  /*888 - 263 = 625 -> 5 segundos*/
  maquinaEstados(2);
  if(displayCarroOn == 1){
    digitalWrite(displayPed, HIGH);
    digitalWrite(displayCarro, LOW);
    escreveDisplay(5);
  }
  if(displayCarroOn == 0){
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, LOW);
    escreveDisplay(1);
  }
 }
 if((counter_travessia >= 888) && (counter_travessia < 1013)){
  /*1013 - 263 = 750 -> 6 segundos*/
  maquinaEstados(3);
  if(displayCarroOn == 1){
```

```
    digitalWrite(displayPed, HIGH);
    digitalWrite(displayCarro, LOW);
    escreveDisplay(4);
  }
  if(displayCarroOn == 0){
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, LOW);
    escreveDisplay(0);
  }
 }
 if((counter_travessia >= 1013) && (counter_travessia < 1138)){
  /*1138 - 263 = 875 -> 7 segundos*/
  maquinaEstados(4);
  if(displayCarroOn == 1){
    digitalWrite(displayPed, HIGH);
    digitalWrite(displayCarro, LOW);
    escreveDisplay(3);
  }
  if(displayCarroOn == 0){
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, HIGH);
    escreveDisplay(0);
  }
 }
 if((counter_travessia >= 1138) && (counter_travessia < 1263)){
  /*1138 - 263 = 1000 -> 8 segundos*/
  maquinaEstados(3);
  if(displayCarroOn == 1){
    digitalWrite(displayPed, HIGH);
    digitalWrite(displayCarro, LOW);
    escreveDisplay(2);
```

```
  }
  if(displayCarroOn == 0){
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, LOW);
    escreveDisplay(0);
  }
 }
 if((counter_travessia >= 1263) && (counter_travessia < 1388)){
  /*1388 - 263 = 625 -> 9 segundos*/
  maquinaEstados(4);
  if(displayCarroOn == 1){
    digitalWrite(displayPed, HIGH);
    digitalWrite(displayCarro, LOW);
    escreveDisplay(1);
  }
  if(displayCarroOn == 0){
    digitalWrite(displayCarro, HIGH);
    digitalWrite(displayPed, HIGH);
    escreveDisplay(0);
  }
 }
 if((counter_travessia >= 1388) && (counter_travessia < 1513)){
  /*1513 - 263 = 1250 -> 10 segundos*/
  maquinaEstados(3);
  if(displayCarroOn == 1){
    digitalWrite(displayPed, HIGH);
    digitalWrite(displayCarro, LOW);
    escreveDisplay(0);
  }
  if(displayCarroOn == 0){
    digitalWrite(displayCarro, HIGH);
```

```
      digitalWrite(displayPed, LOW);
      escreveDisplay(0);
     }
    }
    /*Zerar a flag e o contador ao final da travessia*/
    /*Resetar delayDeboucing para permitir nova leitura do botão*/
    if(counter_travessia >= 1513){
     travessiaPedestre = 0;
     counter_travessia = 0;
     delayDebouncing = 6;
    }
   }
}
```