Universidade Federal de São Paulo

Semáforo Inteligente

Sistemas Embarcados

Aluno	Henrique Monteiro de Moraes
Professor	Sergio Ronaldo Barros dos Santos
Horário	Ter e Qui - 10:00-12:00

São José dos Campos, 22 de Maio de 2018.

Conteúdo

1 Descrição do projeto

1.1 Descrição

Diante da situação caótica do trânsito nas grandes cidades brasileiras, somado aos acidentes que envolvem travessias de pedestres e no cruzamento de vias movimentadas, surgiu-se a ideia de desenvolver um semáforo inteligente para tentar diminuir ao máximo esses problemas que nosso país enfrenta.

Este projeto atende situações em que se necessite a atravessia de pedestres e o cruzamento de veículos à essa mesma via, cujo trânsito varia de acordo com o horário do dia, podendo este ser baixo ou extremamente intenso e, médio para intenso na maior parte do dia.

Porém é importante enfatizar que a ideia principal deste semáforo é desafogar o trânsito desta via principal. Ao momento que o pedestre apertar o botão para travessia ou algum veículo for detectado na para atravessia, conta-se x segundos dependendo do tráfego da via principal.

1.2 Objetivo

O objetivo do semáforo inteligente aqui desenvolvido é poder ser aplicado em qualquer cidade do mundo. Além disso, o modo como o *software* foi desenvolvido permite que, alterando-se as variáveis globais, altera-se o tempo em que as luzes do semáforo ficam acesas ou apagadas. Desta forma aumenta-se ainda mais o leque de aplicação deste projeto.

2 Funcionamento do sistema

2.1 *Hardware*

- **Buzzer:** Auxilia o pedestre deficiente visual de modo que, quando o sinal está verde para o pedestre, o buzzer fica apitando em um intervalo de 0,5 em 0,5 segundos e, quando está a 3 segundos de fechar, fica com sinal alto constante, gerando um apito constante. Trata-se de um buzzer ativo que gera ruídos sonoros a partir da excitação elétrica de componentes piezoelétricos.

- ***Push Botton* com resistor externo *Pull Down:*** Permite que o pedestre inicie o processo de atravessia da via principal. Trata-se de uma simples chave mecânica que, ao apertado, os contatos dos terminais de cada lado ligam-se entre si.

- ***Display* de 7 segmentos com decodificador BCD 4511:** Para mostrar a contagem regressiva aos veículos (tanto de abertura quanto de fechamento) da via principal utiliza-se estes componentes. O *display* utilizado possui catodo comum para todos os LED's, juntamente com um resistor de 300Ω para limitar a tensão nos mesmos. Já o decoder possui 4 pinos de entrada e 7 de saída (um para cada LED do *display*. O decoder permite economia no espaço da memória, além de não precisar do uso de várias portas para se utilizar o *display*.

- **Relé:** O relé permite o acionamento da lâmpada caso esteja noite (pouca luminosidade) e o pedestre aperte o botão para atravessar a via principal. Ao receber um sinal da GPIO, um diodo emissor de luz é ligado e um fototransistor (cuja função é atuar como uma chave) detecta essa luz permitindo que a fonte externa de 5V alimente as bobinas.

A bobina, ao o ser energizada pelo acionamento do botão, gera um campo magnético que atua como um ímã e, consequentemente, atrai uma pequena haste permitindo o fechamento do circuito. Ao interromper o fluxo de corrente, a bobina deixa de agir como um íma e o circuito abre novamente.

- ***Display* LCD:** Mostra aos pedestres e veículos da via secundária a contagem regressiva de abertura e fechamento do sinal. Trata-se de um visor contendo duas linhas de 16 caracteres cada. Cada caracter dispõe de uma matriz retangular 5x8 de luzes que, dependendo da combinação, formam determinado caractere. Nele também é mostrado a temperatura do ambiente, "lida"a cada 1 segundo pelo thermistor NTC.

- **Sensor reflexivo TCRT5000:** Apesar de não ser o compenente ideal para a função que exerce, ele é utilizado para identificar quando há veículos na via secundária querendo atravessar a via principal. Possui um led que emite radiação infravermelha e um fototransistor que, ao receber o infravermelho refletido (emitido pelo led) emite um sinal de entrada digital no GPIO do arduíno.

- **Potenciômetro B10K:** É o potenciômetro que simula o tráfego da via principal através do valor de sua resistência que, através da entrada analógica, varia de 0 a 1023Ω.

- **Sensor de luminosidade LDR e Sensor de temperatura NTC 10Kr:** O primeiro trata-se de um fotocondutor que varia sua resistência de acordo com a luz que o incide, ou seja, quanto maior a luminosidade do local, menor é a resistência do componente. Esta resistência é convertida em tensão através de um divisor de tensão e ligada à uma entrada analógica do arduíno.

  O segundo diminui sua resistência com o aumento da temperatura (por ser NTC). Assim como o LDR, este sensor também converte a resistência em tensão através de um divisor de tensão.

2.2 Esquema elétrico

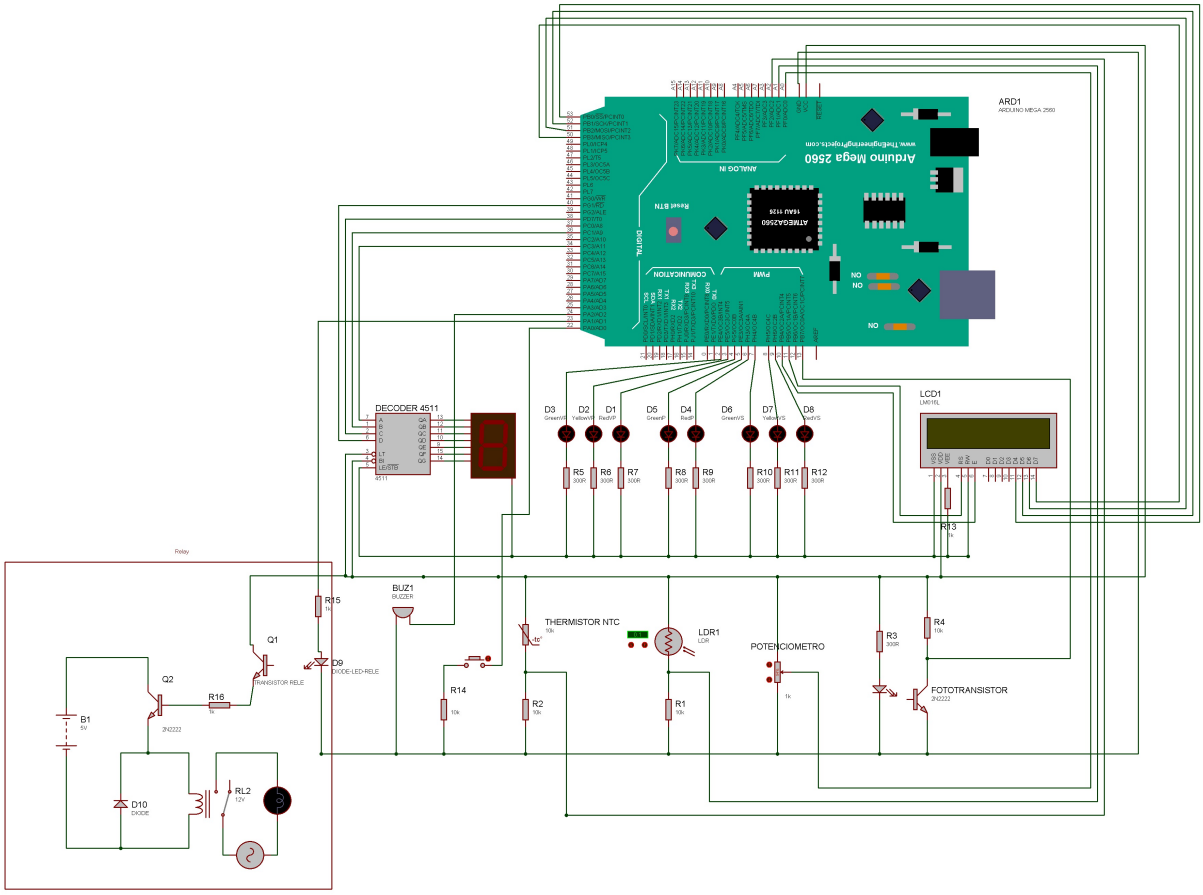

Figura 1: Esquema elétrico do projeto

2.3 *Software*

Para melhor entendimento do código, é imporante lembrar que o projeto todo foi desenvolvido com objetivo principal desafogar o tráfego das vias em que nos horários de picos são intensos.

O código todo é basicamente divido em duas partes:

- Ocorre evento: Há pedestres querendo atravassar a via principal ou veículos querendo atravessar a mesma,
- Não ocorre evento: mantendo-se assim o semáforo da via principal aberto e os semáforos para pedestre e via secundária fechados

A partir disso, a função *loop*() é mostrada abaixo:

```
void loop()
{
```

```
  pwmLDR = map(analogRead(pinoLDR), 0, 1023, 10, 255);
  if (eventoOcorrido())
    semaforoSecundario();
  else
    semaforoPrincipal();
}
```

Enquanto não se tem eventos, a função *eventoOcorrido* retorna *false* e então a função *semaforoPrincipal* é chamada, cujo objetivo é manter o led verde da via principal aceso, juntamente com os leds vermelhos da via secundária e de atravessia do pedestre.

Como pode-se perceber, todos os cálculos e funcionamento do projeto estão na função *semaforoSecundario*, que é um tanto quanto complexa devido ao fato de ser toda desenvolvida através da função *millis*. A Figure 2 abaixo mostra o ciclo de um evento, seja ele um pedestre solicitando atravessia ou um veículo solicitando cruzamento.

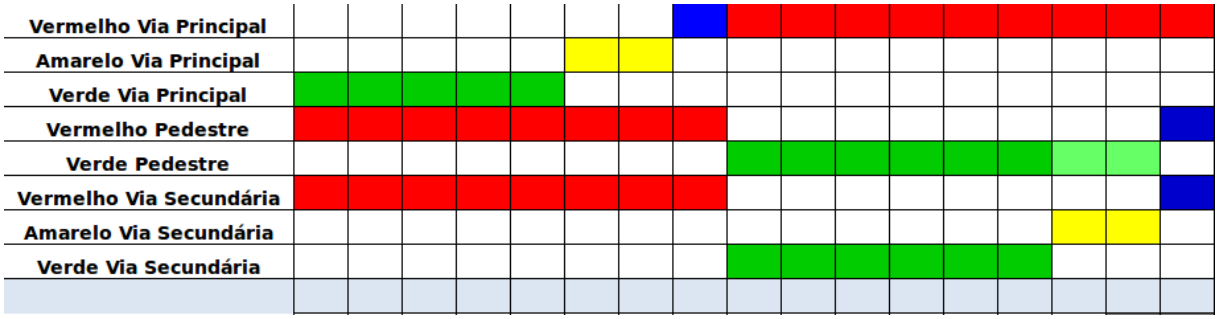

Figura 2: Ciclo de um evento quando o tráfego está baixo

Na Figure 2, cada divisão de quadrado representa 1 segundo e este ciclo é executado quando o tráfego está baixo (apenas como exemplo para entendimento do processo). Os quadrados em azul representam o *delay* do fechamento de um sinal para a abertura do outro, de modo a evitar possíveis colisões. Já os quadrados em verde claro representa o tempo em que o sinal verde para pedestre está fechando, ou seja, ele fica piscando para que o pedestre entenda que ele está próximo a fechar.

A função que executa o tempo em que cada led deve permanecer *HIGH* ou *LOW* é mostrado abaixo. Ela recebe como parâmetro o led em questão, o tempo que ele ficará *HIGH*, o tempo que ele ficará *LOW*, o tempo de atraso (dependendo do tráfego da via principal) e o valor da função *millis* ao se iniciar o ciclo.

```
void piscaLed(int pin, int timeOn, int timeOff, int atraso,
              long millis_referencia = 0)
{
  long ajuste_referencia = millis_referencia % (timeOn + timeOff);
  long resto = (millis() + timeOn + timeOff -
                ajuste_referencia - atraso) % (timeOn + timeOff);
  if (resto < timeOn)
    analogWrite(pin, pwmLDR);
  else
    digitalWrite(pin, LOW);
}
```

As chamadas da função acima são realizadas na função *semaforoSecundario* da função *loop*. O trecho do código das chamadas da função *piscaLed* é mostrada abaixo:

```
  piscaLed(pinoVerde1, tempoAtraso, tempoCiclo - tempoAtraso, 0,
           millis_referencia );
  piscaLed(pinoAmarelo1, tempoLedAmarelo, tempoCiclo - tempoLedAmarelo,
           tempoAtraso, millis_referencia );
  piscaLed(pinoVermelho1, tempoEspera * 2 + tempoLedVerdePedestre +
           tempoLedVerdePedestreFim, tempoAtraso + tempoLedAmarelo,
           tempoAtraso + tempoLedAmarelo, millis_referencia);

  piscaLed(pinoVerde3, tempoLedVerdePedestre, tempoCiclo -
          tempoLedVerdePedestre, tempoAtraso + tempoLedAmarelo +
          tempoEspera, millis_referencia);
  piscaLed(pinoAmarelo3, tempoLedAmarelo, tempoCiclo - tempoLedAmarelo,
          tempoAtraso + tempoLedAmarelo + tempoEspera +
          tempoLedVerdePedestre, millis_referencia);
  piscaLed(pinoVermelho3, tempoCiclo - tempoLedVerdePedestre -
          tempoLedVerdePedestreFim - 1000, tempoLedVerdePedestre +
          tempoLedAmarelo, 0, millis_referencia);

   int t;
   t = verificaTime(tempoCiclo - tempoLedVerdePedestreFim,
                    tempoLedVerdePedestreFim, tempoCiclo - tempoEspera,
                    millis_referencia);
   if (t == 1)
     piscaLed(pinoVerde2, tempoLedVerdePedestre, tempoCiclo -
              tempoLedVerdePedestre, tempoAtraso + tempoLedAmarelo +
              tempoEspera, millis_referencia);
   else
     piscaLed(pinoVerde2, 100, 100, 100, millis_referencia);

   piscaLed(pinoVermelho2, tempoCiclo - tempoLedVerdePedestre -
            tempoLedVerdePedestreFim, tempoLedVerdePedestre +
            tempoLedVerdePedestreFim, tempoCiclo - tempoEspera,
            millis_referencia);
}
```

Com relação ao código acima, observa-se que a o led verde para o pedestre chama a função *picaLed* em duas ocasiões distintas: A primeira é quando o led deve ficar aceso constantemente. A segunda refere-se ao tempo que o led fica piscando, pois desta forma o pedestre saberá que o tempo de travessia está se esgotando.

Ainda na Figure 2, o que caracteriza este ser um semáforo inteligente é o sinal verde da via principal. Este é o único valor que varia e isto depende da intensidade do tráfego da via principal (dado através do potenciômetro), de modo que, se o tráfego estiver:

- **intenso**: este valor é de 15 segundos;
- **médio**: 10 segundos;
- **baixo**: 5 segundos;

No código, este sinal verde é representado pela variável *tempoAtraso*, ou seja, é o tempo de atraso que a mudança do sinal comece a ser efetuada. Seu cálculo é feito através da função abaixo:

```
int calculoTempoAtraso()
{
  if (analogRead(pinoPotenciometro) < 341)
    return 5000;
  else if ((analogRead(pinoPotenciometro) >= 341) &&
           (analogRead(pinoPotenciometro) < 682))
    return 10000;
  else
    return 15000;
}
```

Por fim, a última função essencial para cálculo dos tempos em que as luzes do semáforo ficarão acesas ou apagadas é a que calcula o cálculo do ciclo total, como segue abaixo.

```
int calculoTempoCiclo()
{
  //int temp = calculoTempoAtraso();
  return (tempoLedAmarelo + tempoEspera * 2 +
          tempoLedVerdePedestre + tempoLedVerdePedestreFim +
          tempoAtraso);
}
```

Por questão de limite de páginas, neste relatório não será apresentado os códigos referentes aos componentes: fotoresistor LDR; *termistor* (sensor de temperatura); *display* LCD mostrando a temperatura e contagem regressiva para o pedestre/motorista da via secundária; *display* de 7 segmentos para mostrar a contagem regressiva para os motoristas da via principal; relé para acender a luz para o pedestre caso a luminosidade do ambiente esteja baixa e do buzzer, que auxilia pedestres deficientes visuais.

Em suma foi apresentado apenas o essencial de um semáforo inteligente no que diz respeito ao *software*, pois o que foi citado no parágrafo anterior foram apenas incrementos para deixar o projeto mais interessante do ponto de vista de uma aplicação.

2.4 Fluxograma do *Software*

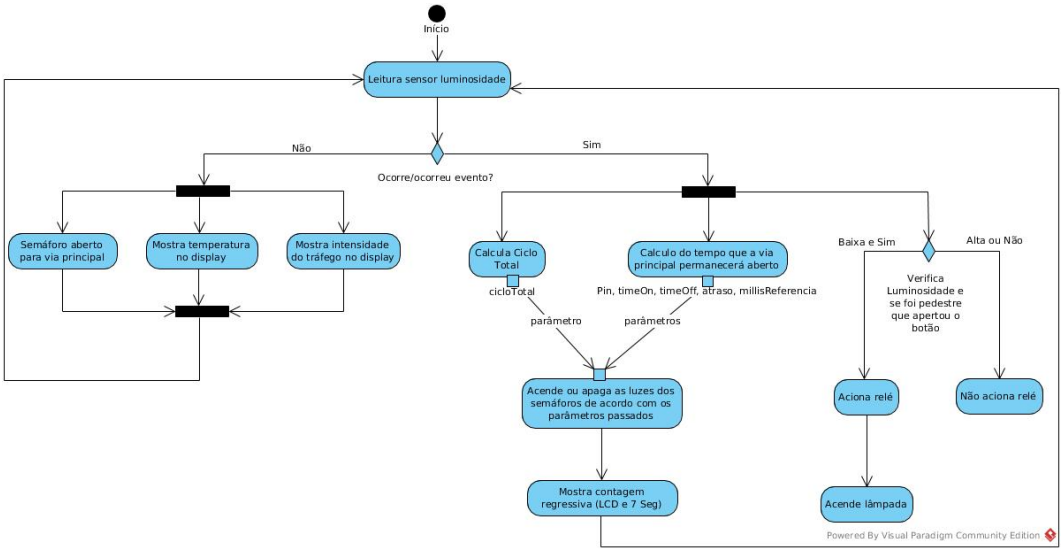

Figura 3: Fluxograma

3 Lista de componentes básicos

Os componentes utilizados no projeto foram os seguintes:

- 8 Leds, que representam as luzes do semáforo
- Resistores
  - 8 de 470Ω para os leds
  - 4 de 10KΩ para botão, termistor, LDR e fototransistor do TCRT5000
  - 3 de 1KΩ em série para ajustar o contraste do *display* LCD
  - 2 de 150Ω em séria para o emissor de infravermelho do TCRT5000
  - 1 de 300Ω para o *display* de 7 segmentos
- 1 Buzzer ativo
- 1 Potenciômetro B10K
- 1 Push Botton configurado da forma *pull down*
- 1 Termistor NTC 103
- 1 Sensor LDR
- 1 Sensor reflexivo TCRT5000
- 1 *Display* de LCD 16x2
- 1 *Display* de 7 segmentos
- 1 Modulo Relé de dois canais