Projeto 1: Semáforo para pedestres

A Avenida Roxo Moreira separa a Unicamp de um bairro cheio de restaurantes. Embora haja uma faixa de pedestres para travessia, os carros nem sempre param sob demanda. Como consequência, há risco de atropelamento nessa faixa. Pensando em diminuir esse risco, nossa equipe projetará uma prova de conceito para um semáforo que fecha para os carros sob demanda de pedestres que, porventura, estejam esperando para atravessar. Porém, é sabido que é perigoso que motoristas fiquem parados na rua durante a noite. Portanto, o semáforo também deve ser desativado, piscando amarelo para os motoristas e vermelho para os pedestres durante todo o período da noite.

1 Objetivos

1. Familiarização com a plataforma de desenvolvimento (Arduino UNO + IDE).
2. Acionamento de circuitos básicos utilizados em sistemas embarcados (LEDs, botões, displays).
3. Utilização de interrupções temporais periódicas e máquinas de estados finitos para coordenar processos em tempo real.
4. Programação de sistemas que envolvem tarefas simultâneas.
5. Utilização de ferramentas de simulação para suporte do desenvolvimento do projeto.

2 Teoria

1. Ao projetar uma máquina de estados finitos em *software* (uma máquina de Mealy ou de Moore, por exemplo), como é possível representar os estados da máquina? E as transições?
2. Como é um circuito que permita acender um LED? Qual é a queda de tensão em um LED vermelho? E num amarelo? E num verde? Qual é a corrente que passa pelos LEDs para que eles acendam?
3. Como é possível usar uma chave mecânica para aplicar tensões Vcc e terra em um ponto específico de um circuito? Qual é a função do resistor de *pull-up*, nesse caso?
4. Como acionar um display de 7 segmentos? Como realizar o acionamento utilizando menos GPIOs?

2.1 Máquina de Estados Finitos

Uma máquina de estados finitos é uma abstração, isto é, um modelo que pode ser usado para analisar determinados tipos de sistemas. Esse modelo se aplica a sistemas que, a cada instante de tempo, assumem um, e apenas um, dentre *N* estados possíveis. Para cada estado, o sistema assume um comportamento diferente e, dependendo de suas entradas, ocorre uma transição para outro estado. Poderíamos analisar o voo de um avião como uma máquina de estados finitos. Inicialmente, ocorre o estado de embarque. Depois, há o estado de decolagem. Após, o estado de voo de cruzeiro, o pouso e, por fim, o desembarque. Perceba que o comportamento do avião é diferente em cada um desses estados. A transição entre os estados depende da aquisição de dados dos sensores de bordo (GPS, altímetro, velocímetro, etc.) e da comparação desses dados com um plano de voo.

Para implementar uma máquina de estados em *software*, precisamos de uma variável para representar (armazenar) o estado em que nos encontramos. Também, precisamos de uma função que implemente as regras de transição de estados. O exemplo mostrado abaixo implementa uma máquina de dois estados (0 e 1) que muda de estado em todos os acionamentos.

```
int estado = 0; /* Estado atual da maquina
    */

void maq_estados() {
  /* Esta implementacao usa uma estrutura
      de if-then-else explicita */
  if (estado == 0) {
    estado = 1;
  }
  else {
    estado = 0;
  }
}
```

2.2 LED

O LED (*Light-Emitting Diode*) é um diodo que emite luz na passagem de corrente. Um possível modelo para seu funcionamento elétrico é:

1. Se a tensão aplicada no LED é inferior a um limiar V_l, a corrente que passa por ele é nula;
2. Caso contrário, a corrente que passa pelo LED é tão grande quando possível e observa-se uma tensão de v_l nos terminais do LED. A tensão V_l varia de acordo com a cor do LED e depende de características do material semicondutor do qual ele é feito.

Esse modelo significa que uma fonte de tensão conectada diretamente ao LED fornecerá teoricamente corrente infinita. Isso rapidamente levará o LED (e, possivelmente, a própria fonte de tensão) a queimar. Por esse motivo, é preciso conectar o LED em série a um resistor, como mostra a Figura 1.

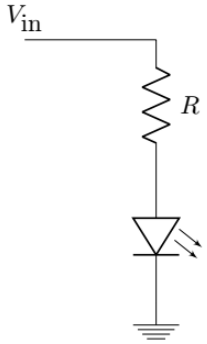


Figura 1: Circuito para acender um LED.

No circuito da Figura 1, observamos uma malha na qual passa uma corrente i. Assumindo que a tensão fornecida pela fonte é maior que o limiar de tensão para acender o LED ($V_{in} > V_l$), podemos verificar que a tensão no resistor é igual a (pela Lei de Kirchoff das Tensões) $\boldsymbol{V_r = V_{in} - V_l}$. A Lei de Ohm nos permite inferir que a corrente na malha é $\boldsymbol{i = (V_{in} - V_l)/R}$.

Isso significa que é possível variar a corrente que passa pelo LED alterando a resistência do elemento colocado em série a ele. Uma corrente muito baixa levará a um brilho muito baixo, ao passo que uma corrente muito alta poderá danificar os dispositivos conectados. A corrente máxima tolerada pelo LED pode ser encontrada em seu *datasheet*.

2.3 Chave mecânica

Um dispositivo *push button* é uma chave mecânica que é ativada quando pressionada. Ao ser ativada, a chave fecha o contato entre dois pontos. A Figura 2 mostra um circuito que pode ser usado para conectar um botão a um microcontrolador. No circuito, temos duas situações possíveis: a chave aberta e a chave fechada. Quando a chave está aberta, não passa corrente no circuito e, portanto, a tensão no resistor é zero e, portanto, V_{out} é zero. Quando a chave está fechada, o resistor é conectado à fonte, e a tensão de saída, V_{out}, é igual a V_{in}. A corrente no resistor é V_{in}/R.

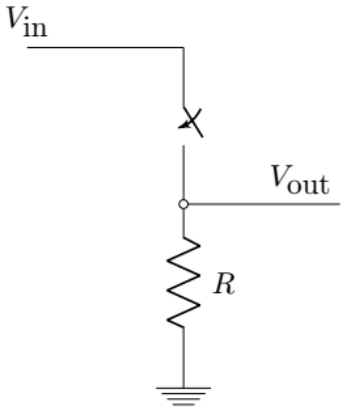
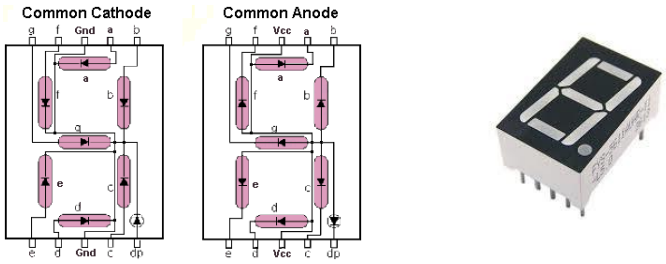


Figura 2: Circuito com uma chave.

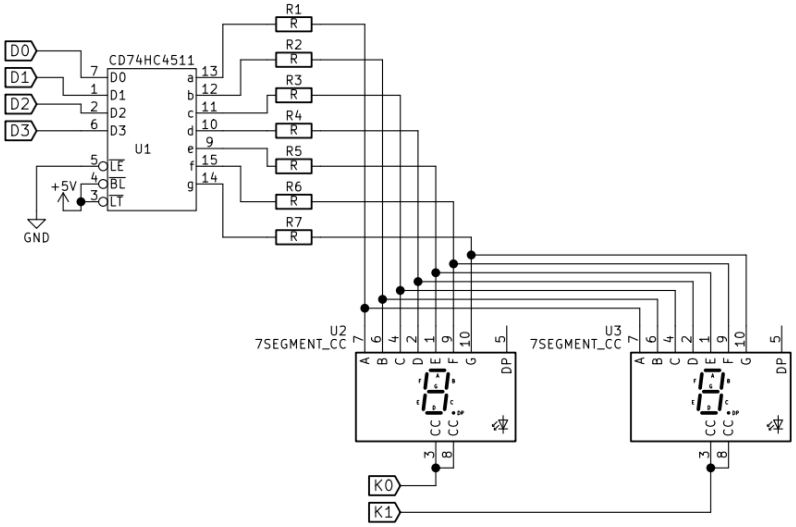
Observe que, nessa estrutura, o resistor tem a função de induzir um estado-padrão no sistema, válido quando a chave está aberta. Com a chave aberta, sem o resistor, V_{out} seria um nó flutuante, e, portanto, sujeito a oscilações de tensão devidas ao ruído eletromagnético do ambiente.
Na estrutura da Figura 2, o resistor faz com que o estado-padrão da saída do circuito seja baixo ($0V$), e, portanto, é chamado de resistor de *pull down*. Se as posições da chave e do resistor forem invertidas no circuito, o estado padrão do sistema passa a ser alto (V_{in}), e, assim, o resistor passa a ser chamado de resistor de *pull up*.

2.4 Display de 7 segmentos

Os displays de 7 segmentos são dispositivos simples e relativamente baratos que permitem a exibição de informações alfanuméricas. Cada segmento é constituído por um LED independente dos demais e, dessa forma, seu acionamento se dá de modo semelhante ao descrito na seção 2.2. Esses displays podem ser do tipo anodo comum, nos quais os anodos de todos os LEDS estão interligados, ou catodo comum, nos quais os catodos estão interconectados. Um display de um dígito geralmente possui 10 pinos: um pino para o acionamento de cada segmento, um pino para o ponto decimal e dois pinos conectados ao anodo ou catodo comum, dependendo do tipo de display. Os segmentos são identificados pelas letras *a*, *b*, *c*, *d*, *e*, *f* e *g*, conforme mostrado na figura a seguir.



Um display pode ser acionado por um microcontrolador utilizando-se GPIOs distintas para acender ou apagar cada um dos seus segmentos. Nesse caso, desconsiderando o ponto decimal, são necessárias 7 GPIOs por display. Dependendo do número de displays no sistema, pode ser que não haja GPIOs suficientes para acioná-los. Alternativamente, é possível reduzir o número de GPIOs a serem usadas adotando-se uma estratégia que combina o emprego de um circuito decodificador para 7 segmentos, o 74HC4511 por exemplo, com o acionamento multiplexado no tempo dos displays, conforme o diagrama esquemático a seguir.



O decodificador ativa os segmentos do display de acordo com um código binário de 4 bits, D0 a D3 que representa o valor a ser exibido. Dessa forma, ao invés de sete, são necessárias quatro GPIOs para acionar um display. Um único decodificador é usado para acionar todos os displays, mas apenas um por vez. A seleção do display ativo é realizada a partir dos sinais K0 e K1 que estão conectados ao pino catodo comum: o nível baixo ativa enquanto o nível alto desativa o display. Se a frequência de ativação dos displays for

suficientemente alta, devido ao fenômeno da persistência da visão, tem-se a ilusão de que todos os displays estão ativos simultaneamente, mesmo que na realidade estejam piscando. Esse fenômeno viabiliza a estratégia de multiplexação do acionamento dos displays.
Note que são usadas 6 GPIOs para acionar dois displays dessa maneira. Sem usar o decodificador e a multiplexação, seriam necessárias 14 GPIOs. Do ponto de vista do recrutamento de recursos, essa estratégia é bastante vantajosa. Entretanto, o preço a pagar pela redução no número de portas é um aumento na complexidade do programa responsável por gerenciar o sistema.

3 Descrição do problema

O grupo de trabalho deverá apresentar uma prova de conceito do sistema de semáforo para pedestres. Ele deverá funcionar da seguinte forma:

• O semáforo opera normalmente aberto para os carros e fechado para os pedestres. Portanto, o sistema deve ser inicializado nessa condição.
• Quando um pedestre aperta o botão, o sistema:
1. Aguarda algum tempo;
2. Mostra a luz amarela para os carros por algum tempo;
3. Fecha para os carros e, simultaneamente abre para os pedestres;
4. Displays de 7 segmentos exibem o tempo restante para travessia dos pedestres. O display dos veículos inicia a contagem a partir de 9 e o display dos pedestres a partir de 5. A cada segundo, o valor mostrado nos displays é decrementado;
5. Ao chegar ao valor zero, o display e a luz vermelha dos pedestres começam a piscar (0,5s apagados e 0,5s acesos) indicando que o semáforo irá fechar em breve. Ao mesmo tempo, a contagem regressiva do semáforo dos veículos deve prosseguir.
6. Quando a contagem associada aos veículos atinge zero, o semáforo fecha para os pedestres e, simultaneamente, abre para os carros (isto é, retorna ao estado inicial). Os displays são desligados.
7. À noite, o sistema de apertar o botão é desligado e o semáforo somente pisca a luz amarela para os carros e vermelha para os pedestres, retornando ao estado original na manhã seguinte. O grupo deve utilizar um sensor tipo LDR para detectar a quantidade de luz no ambiente de modo a determinar se é dia ou noite. O sistema deve ser capaz de ignorar mudanças temporárias na luz, tais como a passagem de um pássaro sobre o sensor durante o dia, ou a incidência de um farol de carro durante a noite. Ou seja, apenas alterações de luminosidade que perdurem por pelo menos alguns segundos podem mudar o modo de operação de diurno para noturno ou vice-versa.

Dada a existência de tarefas concomitantes, o uso de espera ativa pode inviabilizar o funcionamento do sistema. Dessa forma, os grupos devem utilizar interrupções periódicas do temporizador e **não será permitido o uso de funções como *delay*() ou qualquer outro tipo de espera ativa**. **Também não é permitido o uso da função *milis*() ou qualquer outra equivalente**. A temporização do sistema será realizada através de interrupções periódicas geradas através do código disponibilizado pelo professor.

4 Execução do projeto

O projeto divide-se entre a montagem do circuito em *protoboard* e a programação do microcontrolador para implementar o semáforo especificado na seção 3. Além disso, utilizaremos o Tinkercad para simular e validar o funcionamento do sistema concomitantemente com a implementação física. O programa de controle será desenvolvido a partir da modelagem do comportamento do sistema por meio de uma máquina de estados finitos.

5 Funções úteis

A seguir, são listadas algumas funções úteis que podem ser usadas na programação do Arduino UNO. A lista completa de funções e suas descrições detalhadas pode ser consultada em https://www.arduino.cc/reference/en/.

- Funções para manipulação de portas de entrada e saída de propósito geral (GPIOs):

```
pinMode(pin, mode)
digitalWrite(pin, value)
digitalRead(pin)
```

- Funções para manipulação do conversor analógico-digital (ADC):

```
analogRead(pin)
analogReference(type)
```

- Funções para manipulação da interface serial (UART). Útil para depuração do código:

```
Serial.begin(speed, config)
Serial.println(val)
```

Datasheet 74hct4511
https://assets.nexperia.com/documents/data-sheet/74HC_HCT4511.pdf