ESTGOH
Escola Superior de Tecnologia e Gestão de Oliveira do Hospital
Instituto Politécnico de Coimbra

Licenciatura em Engenharia Informática

Tecnologia e Arquitetura de Computadores 2022/2023

Trabalho Prático nº 1

Realizado em: 21/04/2023
Elaborado em: 22/04/2023

Mariana Magalhães - a2022147454

Índice

1. Introdução

Este trabalho tem como objetivo criar um programa para controlar a entrada e saídas de veículos e peões em um cruzamento. Ele utiliza um sensor ultrassónico para detetar a presença de veículos, então muda o sinal de trânsito em favor do fluxo de carros. Após um tempo determinado, ele muda o sinal para permitir que os peões possam atravessar com segurança.

O programa começa definindo o estado inicial dos semáforos para veículos e peões e em seguida, ele entra em um loop principal com três estados:

O primeiro, aguarda a deteção de um obstáculo (no caso, um peão) através de um sensor ultrassónico;

No segundo, aciona os semáforos dos veículos na sequência correta de cores.

No último, aciona os semáforos dos peões na sequência correta de cores, permitindo assim que eles atravessem com segurança.

Dentro desses estados, o programa usa funções para controlar o tempo de sinal e fazer com que o sinal mude de verde para amarelo e depois para vermelho, para alertar os motoristas. Também existe uma função para acender um LED intermitente se a luz estiver baixa.

Os sinais nos semáforos são controlados usando as funções **digitalWrite()** para definir o nível **HIGH** ou **LOW** das saídas de pino correspondentes ao LED do semáforo. O tempo do intervalo de cada estado é definido pela função **millis()**, que é uma referência ao tempo do sistema desde o início do arduino. As funções **carros()** e **peoes()** controlam o tempo de transição entre estados e, quando necessário, alteram o estado do semáforo. Além disso, há uma função adicional **led_intermitente()** que faz um LED intermitente funcionar como um alerta visual durante a fase de travessia dos peões, mas ele apenas funcionara em períodos de luminosidade reduzida e para isso é usado um sensor de Luminosidade, onde seu o principal componente é o LDR (Light Dependent Resistor ou Resistor Dependente de Luz) também conhecido como fotoresistor, ele simplesmente varia sua resistência de acordo a quantidade de luz que incide nele.

Há também alguns comandos **Serial.print()** para exibir informações úteis no monitor Serial durante a execução do programa.

2. Métodos

O trabalho foi realizado no decorrer das 3 horas nas últimas duas aulas de Tecnologia e Arquitetura de Computadores (TAC) e cerca de 4 horas fora de aula.

Para a realização deste trabalho foi utilizado o **Tinkercad**, um programa de modelagem tridimensional (3D) online e gratuito que é executado num navegador da web, conhecido por ser simples e fácil de utilizar, sendo este usado para projetar o circuito. Para além do Tinkercad, foi usado o **Arduino IDE**, uma plataforma de prototipagem eletrónica de hardware livre e de placa única, projetada com um microcontrolador com suporte de entrada/saída embutido, uma das linguagens de programação padrão usada no programa é C/C++, neste caso essa linguagem é usada para o desenvolvimento do código. Para a realização do fluxograma foi usado o draw.io, um editor gráfico online no qual é possível desenvolver desenhos, gráficos, entre outros, sem a necessidade de usar um software caro e pesado. Todos este programas foram desenvolvidos num computador com um processador Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz, também foram usados os materiais representados na tabela 1.

Nome	Quantidade	Componente
U1	1	Arduino Uno
DIST1	1	Sensor de distância ultrassônico
D1 D5	2	Vermelho LED
D2 D4	2	Verde LED
D3 D6	2	Amarelo LED
R1, R2, R3 R4, R5, R6	6	Resistências 560Ω
R7	1	Resistência de 1K Ω
R8	1	Fotorresistor

Tabela 1 - Materiais

3. Resultados

Comecei por fazer o projeto do circuito no Tinkercad (Figura 1) e através dessa montagem foi obtido o diagrama do circuito (figura 2), em seguida foram desenvolvidos um algoritmo e um fluxograma (Figura 3) para o desenvolvimento do código no Arduíno.

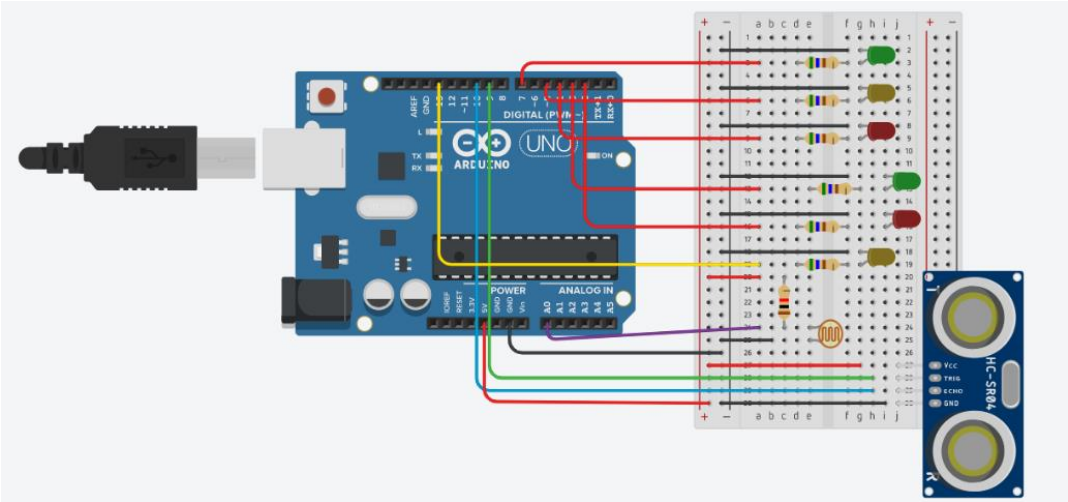

Figura 1 - Tinkercad

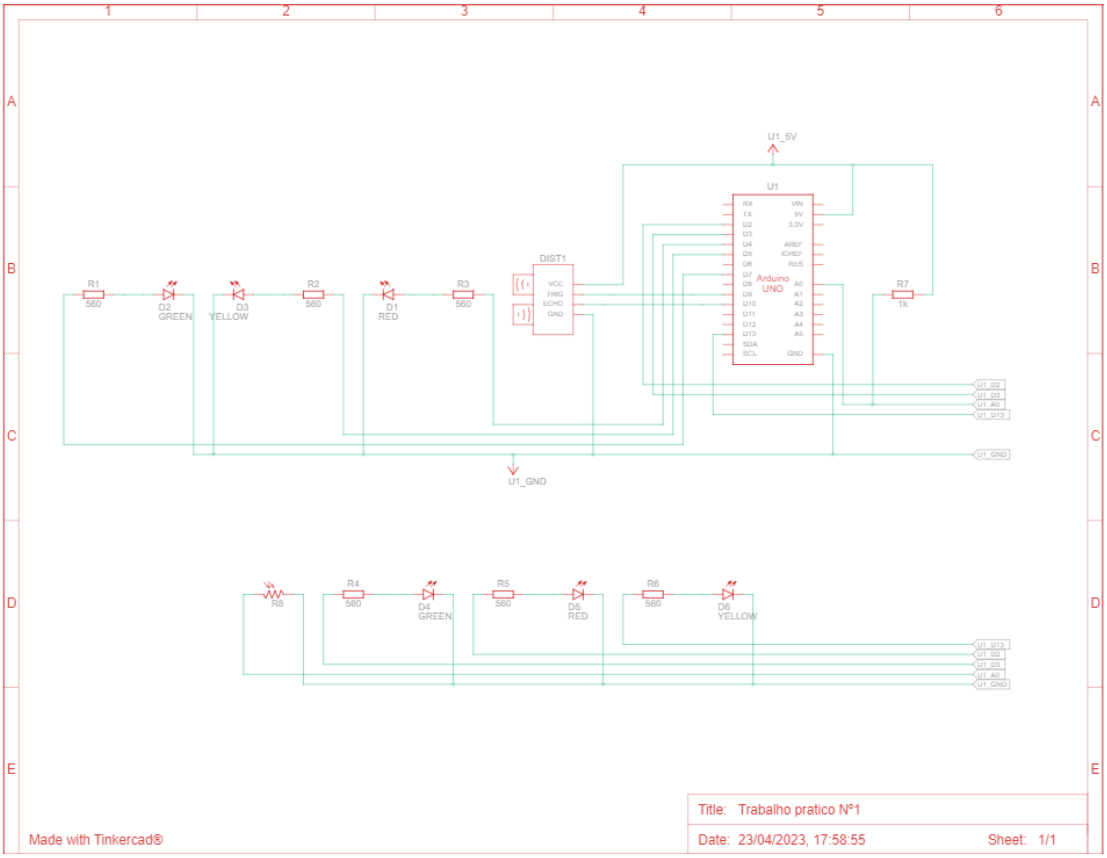

Figura 2 - diagrama do circuito

Algoritmo do programa:

1. Inicializar as entradas e saídas necessárias e definir o estado inicial do semáforo como verde para os carros e vermelho para os peões.

2. Enquanto o programa estiver em execução, verificar qual é o estado atual do semáforo usando a variável "estado".

3. Se o estado for 1 e forem detetados peões pelo sensor, aguarde até que um seja detetado um.

4. Quando um peão for detetado, mude o estado para 2.

5. No estado 2, acende o sinal amarelo após dois segundos e, após mais dois segundos acende o sinal vermelho para os carros. Em seguida, mude o estado para 3 para permitir que os peões possam atravessar com segurança.

6. No estado 3, aguarde 2 segundos para permitir que os peões atravessem a rua com segurança. Em seguida, pisque um LED por um período de tempo definido enquanto o sinal dos peões fica verde. Após dez segundos no estado 3, volte ao estado 1 para procurar mais peões.

7. Repete o processo indefinidamente enquanto o programa estiver em execução.

Ao fim da realização do algoritmo foi desenvolvido o seguinte fluxograma:

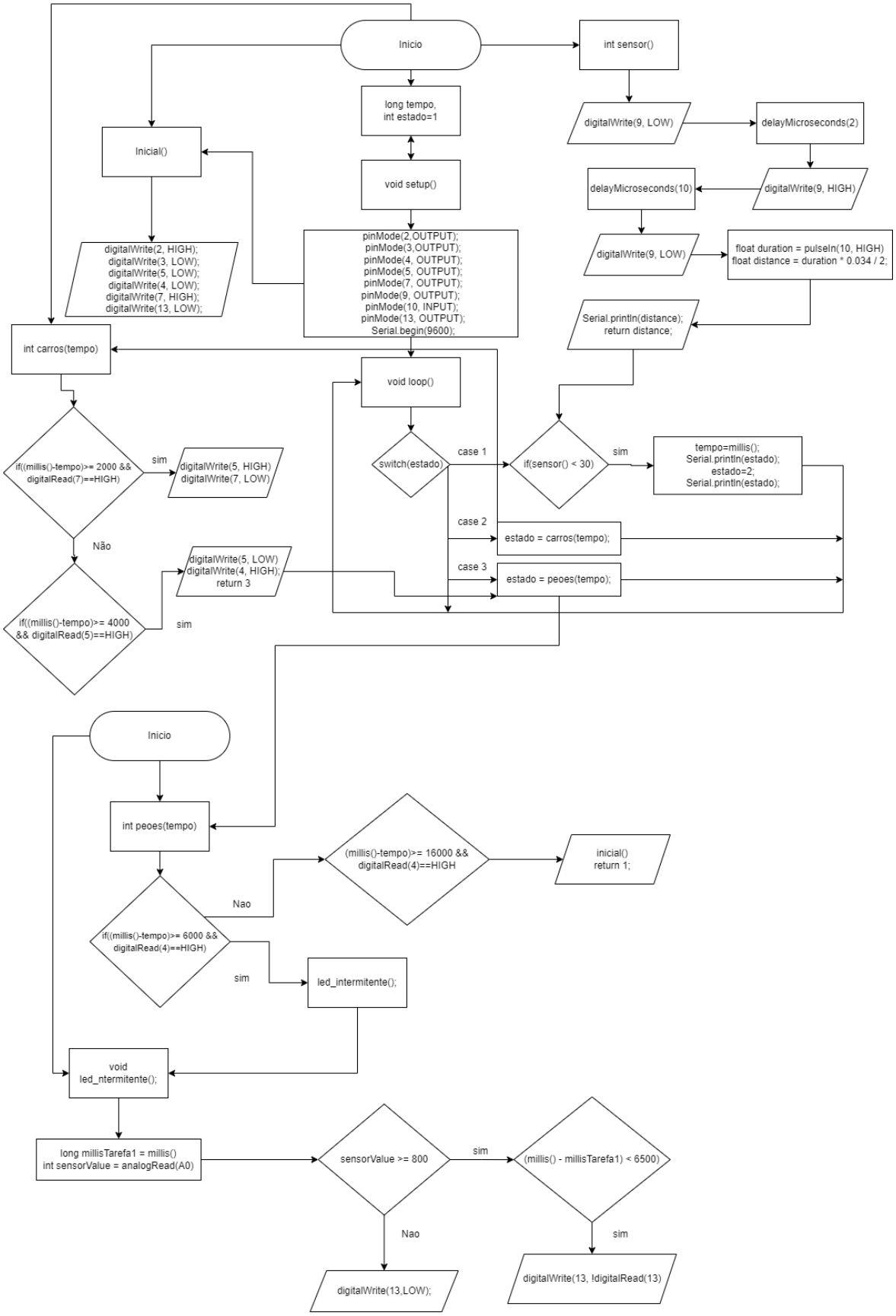

Figura 3 – fluxograma

O código representado abaixo foi o programa utilizado. Ele começa definindo o estado inicial dos semáforos. Em seguida, ele entra numa função principal chamada **loop()**, que é executada continuamente. Dentro dela, há um **switch** para controlar o estado atual do semáforo.
O estado 1 verifica se há algum objeto próximo e se estiver próximo o suficiente, o programa passa para o estado 2, que controla o tempo de sinal verde para os veículos e sinal vermelho para os peões. Se o tempo acabar, o programa passa para o estado 3 que que permite que os peões atravessem com segurança. Se o tempo acabar, o programa volta para o estado 1.

```
int estado=1;
unsigned long tempo;
int sensor(){
  digitalWrite(9, LOW);
    delayMicroseconds(2);
    // Sets the trigPin  on HIGH state for 10 micro seconds
    digitalWrite(9, HIGH);
    delayMicroseconds(10);
    digitalWrite(9, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    float duration = pulseIn(10, HIGH);
        // Calculating the distance
    float distance = duration * 0.034 / 2;
  Serial.println(distance);
    // Prints the distance on the Serial Monitor
    return distance;
}
void inicial(){
  digitalWrite(2, HIGH);//vermelho para os peões
  digitalWrite(3, LOW);//verde para os peões
  digitalWrite(5, LOW);//amarelo veiculos
  digitalWrite(4, LOW);// vermelho veiculos
  digitalWrite(7, HIGH);//verde veiculos
  digitalWrite(13, LOW);
}
int carros( unsigned long tempo) {
  Serial.print("entrou");
  if((millis()-tempo)>= 2000 && digitalRead(7)==HIGH){
    digitalWrite(5, HIGH);//amarelo veiculos
    digitalWrite(7, LOW);//verde veiculos
  }else
  if((millis()-tempo)>= 4000 && digitalRead(5)==HIGH){
  digitalWrite(5, LOW);//amarelo veiculos
  digitalWrite(4, HIGH);// vermelho
  return 3;
  }
  return 2;
}
int peoes(unsigned long tempo){
 if((millis()-tempo)>= 6000 && digitalRead(4)==HIGH){
   led_intermitente();
    digitalWrite(3, HIGH);//verde peoes
    digitalWrite(2, LOW);//vermelho peoes
  }
  if((millis()-tempo)>= 16000 && digitalRead(4)==HIGH){
```

```
    inicial();
  return 1;
  }
  return 3;
}
void led_intermitente(){
  unsigned long millisTarefa1 = millis();
  int sensorValue = analogRead(A0);
  Serial.print("luz: ");
  Serial.println(sensorValue);
  if (sensorValue >= 800) {
    if((millis() - millisTarefa1) < 6500){
    // Acende o led do pino 13
    digitalWrite(13, !digitalRead(13));
  }
  }else
  digitalWrite(13,LOW);
  }
void setup() {
  pinMode(2,OUTPUT);//vermelho para os peões
  pinMode(3,OUTPUT);//verde para os peões
  pinMode(4, OUTPUT);// vermelho veiculos
  pinMode(5, OUTPUT);//amarelo veiculos
  pinMode(7, OUTPUT);//verde veiculos
  pinMode(9, OUTPUT); // Sets the trigPin  as an Output
  pinMode(10, INPUT); // Sets the echoPin as an Input
  pinMode(13, OUTPUT);
  inicial();
  Serial.begin(9600);
}
void loop() {
switch (estado)
  {
  case 1:
  Serial.println("estado ");
  if(sensor() < 30){
    tempo=millis();
    Serial.println(estado);
    estado=2;
    Serial.println(estado);
      }
    break;
  case 2:
    Serial.println("estado");
    estado = carros(tempo);
    break;
  case 3:
    Serial.println("entr");
    estado = peoes(tempo);
    break;
  }}
```

4. Discussão

Uma das coisas a ter em atenção neste trabalho, foi o facto da função **delay()** não ser a função mais apropriada para se utilizar neste programa, porque o que ela vai fazer é basicamente "congelar" o programa numa determinada parte do código por um tempo especificado em milissegundos e durante o período em que o código está parado, não pode ocorrer nenhuma leitura de sensores, cálculos matemáticos ou manipulação de pinos, enquanto que a função **millis()** retorna um número indicando há quantos milissegundos o Arduino está ligado , ou seja, ao invés de interromper o sistema durante um tempo determinado usando a função **delay()**, iremos trabalhar com o valor retornado pela função **millis()** e calcular indiretamente o tempo decorrido.
Portanto, uma das primeiras coisas que se teve que fazer foi guardar o valor da função **millis()** nas variáveis **tempo** e depois na função **led_intermitente()**, na variável **millisTarefa1**. Em seguida, calculamos a diferença de tempo entre as variáveis armazenadas e o tempo atual retornado pela função **millis()** e dessa forma, é possível verificar se já passou o tempo necessário para que uma tarefa seja executada.

Outra coisa em ter em atenção é ligar o **LDR** (sensor de luminosidade) a uma porta analógica do arduino pois as entradas digitais só podem assumir dois estados, **HIGH** e **LOW**, ou seja, 0 V ou 5 V. Dessa forma só é possível ler apenas dois estados, mas em muitas situações a variação das grandezas envolvidas acontece de forma analógica, ou seja, variam continuamente em relação ao tempo e podem assumir valores infinitos dentro de uma faixa, neste caso a luminosidade do tempo vai assumir vários valores ao longo do tempo então vamos ter que ligar a uma porta analógica.

5. Conclusão

Assim sendo podemos dizer que cumprimos com o objetivo e, que este programa é um exemplo simples de como controlar um semáforo e pode ser usado como base para programas mais complexos.

Referências

Jorge, Ricardo. "Função Switch E a Máquina de Estado - State Machine." Panorama Blog Space, 29 Dec. 2020, www.panoramablog.space/blog/funcao-switch-e-a-maquina-de-estado-state-machine/.Accessed 20 Apr. 2023.

Lara, Silvio Garbes. "Função Millis() No Arduino: Aprenda Como Utilizar." MakerHero, 28 Jan. 2020, www.makerhero.com/blog/subtituindo-delay-por-millis-no-arduino/.Accessed 13 Apr. 2023.

"Millis() - Arduino Reference." Www.arduino.cc, 16 Apr. 2020, www.arduino.cc/reference/en/language/functions/time/millis/. Accessed 14 Apr. 2023.

ArduinoPortugal.pt. "Como Usar Uma Fotoresistência LDR Com Arduino." Arduino Portugal, 8 May 2017, www.arduinoportugal.pt/usar-fotoresistencia-ldr-arduino/.Accessed 14 Apr. 2023.

Souza, Fábio. "Entendendo as Entradas Analógicas Do Arduino." Embarcados - Sua Fonte de Informações Sobre Sistemas Embarcados, 18 Dec. 2013, www.embarcados.com.br/arduino-entradas-analogicas/. Accessed 22 Apr. 2023.

Tinkercad:
https://www.tinkercad.com/things/fzUHjjrjCQk?sharecode=mQHcRQ-GL6-v3Z96owYzCbhI6rP14YWgYkm7WvWxfT8

fluxograma:
https://drive.google.com/file/d/1H453mCcy5_6_cQz62Kj3aCfOlmISCTh1/view?usp=sharing