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 5

PWM and FSM

Realizado em: 11/05/2023
Elaborado em: 12/05/2023

Grupo: 5

António Dinis - a2021157297
Francisco Figueiras - a2021155919
Mariana Magalhães - a2022147454

Índice

1. Introdução

Este trabalho tem como objetivo fazer 4 exercícios para ir alternando a intensidade do brilho de um led, para isso foi utilizado o **PWM** (Pulse Width Modulation), uma técnica utilizada para controlar a intensidade de um sinal digital num microcontrolador, como o Arduino. Esta função permite que um pino digital seja ligado e desligado rapidamente com diferentes proporções de tempo entre os estados **HIGH** e **LOW**.

No caso do Arduino, o **PWM** é usado para simular uma saída analógica em um pino digital, ou seja, em vez de produzir um sinal elétrico analógico contínuo, um pino digital PWM envia pulsos elétricos de duração variável, criando assim uma tensão média que pode ser interpretada como uma saída analógica.

Todas as portas do **Arduino UNO** marcadas com o símbolo ~ podem ser usadas para controlar qualquer dispositivo por **PWM.**

Num dos exercícios iremos usar o potenciómetro, que é um componente eletrónico que possui uma resistência elétrica ajustável (elemento resistivo), este componente possui normalmente três terminais onde a conexão central é deslizante e manipulável via cursor móvel. Quando todos os três terminais são usados, ele atua como um divisor de tensão que vai ser usado para determinar a intensidade do led. O mesmo será usado para controlar um servo motor (um atuador eletromecânico, que apresenta movimento proporcional a um comando). Nesse mesmo exercício usamos a função **map(),** que mapeia um valor de entrada de um intervalo para outro, ou seja, um valor de fromLow seria mapeado para toLow, um valor de fromHigh para toHigh (map(value, fromLow, fromHigh, toLow, toHigh)).

2. Métodos

O trabalho foi realizado no decorrer das 3 horas de aula de **Tecnologia e Arquitetura de Computadores (TAC)** 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. Todos este programas foram desenvolvidos num computador com um processador AMD Ryzenn 7 5800H With Radeon Graphics e também foram usados os materiais representados nas tabelas seguintes.

Nome	Quantidade	Componente
U1	1	Arduino Uno R3
D1	1	Red LED
R1	1	560Ω Resistor

Tabela 1 - materiais do exercício 1

Nome	Quantidade	Componente
U1	1	Arduino Uno R3
D1	1	Red LED
R1	1	560Ω Resistor
Rpot1	1	10KΩ Potenciômetro

Tabela 2 - materiais do exercício 2

Nome	Quantidade	Componente
U1	1	Arduino Uno R3
SERVO1	1	Posicional Micro servo
Rpot1	1	10KΩ Potenciômetro

Tabela 3 - materiais do exercício 3

3. Resultados

3.1. Exercício 1

O objetivo neste exercício é controlar a intensidade de brilho de um LED conectado ao pino digital 3 do microcontrolador.

Começámos por fazer o projeto do circuito no Tinkercad (Figura 1) e através dessa montagem foi obtido o diagrama do circuito (figura 2), em seguida foi desenvolvido um algoritmo para o desenvolvimento do código no Arduíno e por fim a sua montagem na breadbord (figura 3).

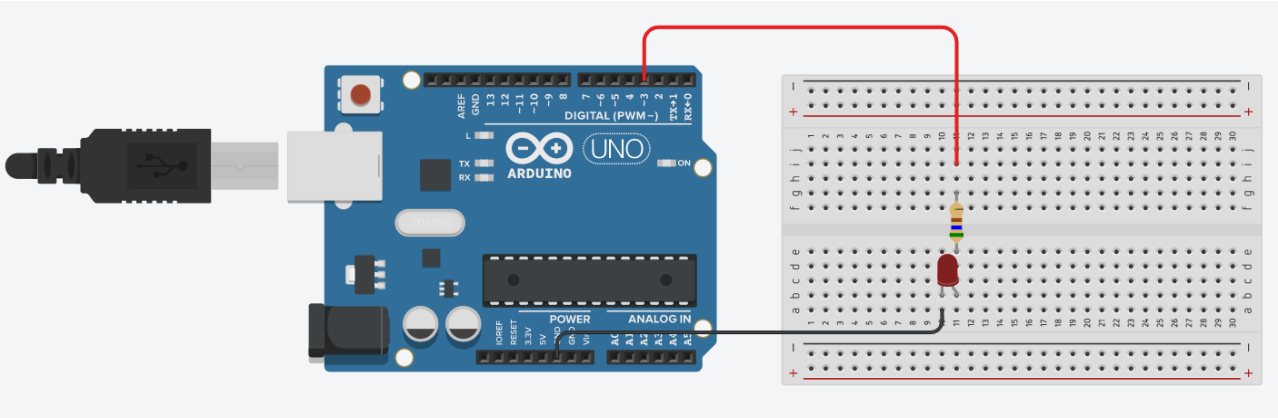

Figura 1 - montagem do circuito (Tinkercad)

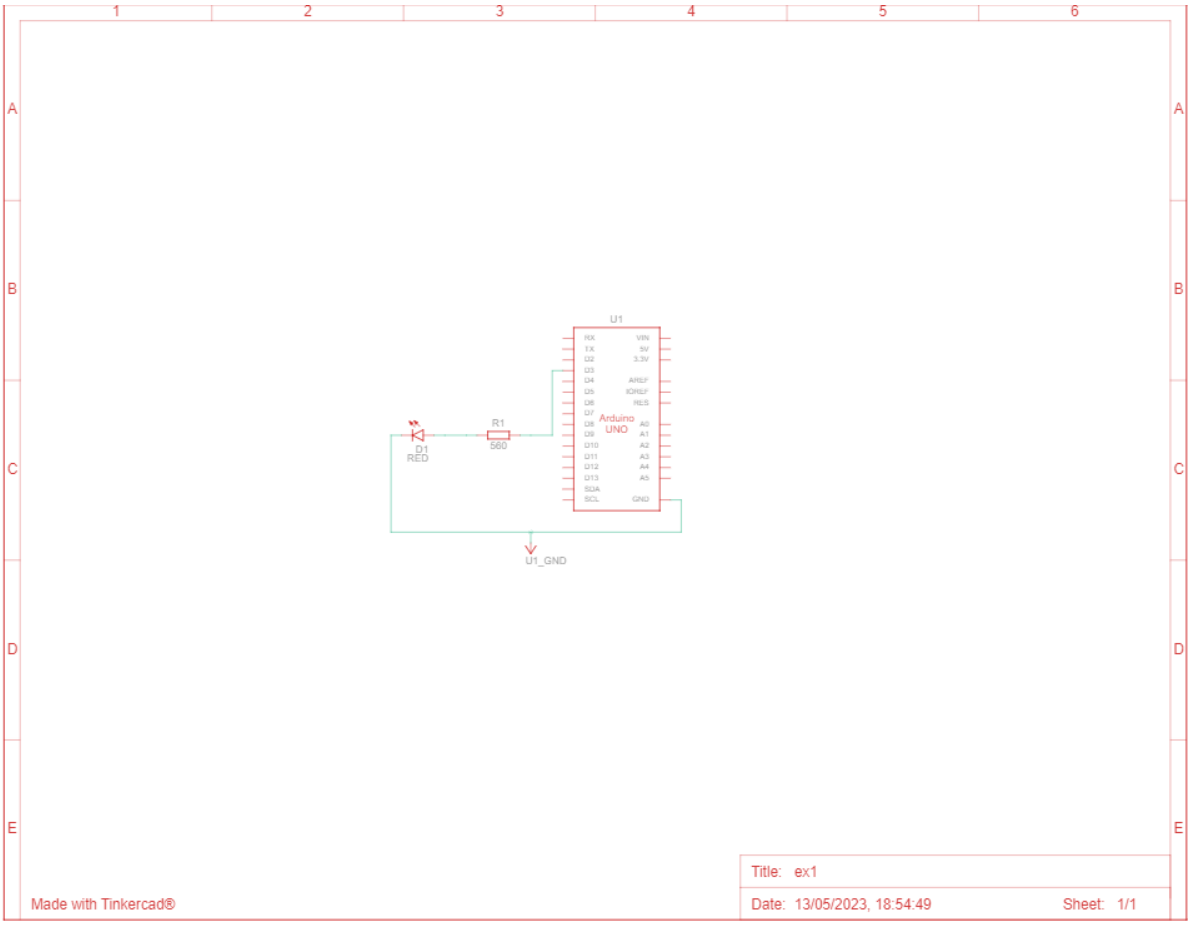

Figura 2 - diagrama do circuito

Algoritmo do programa:

1. Definir o pino digital 3 como uma saída no setup.
2. Iniciar um loop infinito na função loop.
3. Iniciar uma estrutura de repetição "for" para iterar através de 255 níveis de brilho.
4. Para cada iteração, ajustar o brilho do LED usando a função analogWrite com o valor da variável "i".
5. Esperar por 10 milissegundos usando a função delay.
6. Repetir o processo até que o programa seja interrompido.

O código representado abaixo foi o programa utilizado. Começa com a definição do estado inicial do pino 3 como **OUTPUT.** Em seguida, ele entra numa função principal chamada **loop(),** que é executada continuamente. Nessa função usamos uma estrutura de repetição **for** para ir ajustando o brilho do LED para cada nível usando a função **analogWrite**. O valor da variável "i" começa em zero e é incrementado por 1 até chegar a 255. A cada iteração do loop, é aplicado um valor crescente de intensidade de brilho ao LED e mantido por um período de tempo de 10 milissegundos usando a função **delay().**

```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {
  for (int i = 0; i <= 255; i++) {
    analogWrite(3, i);
    delay(10);
  }
}
```

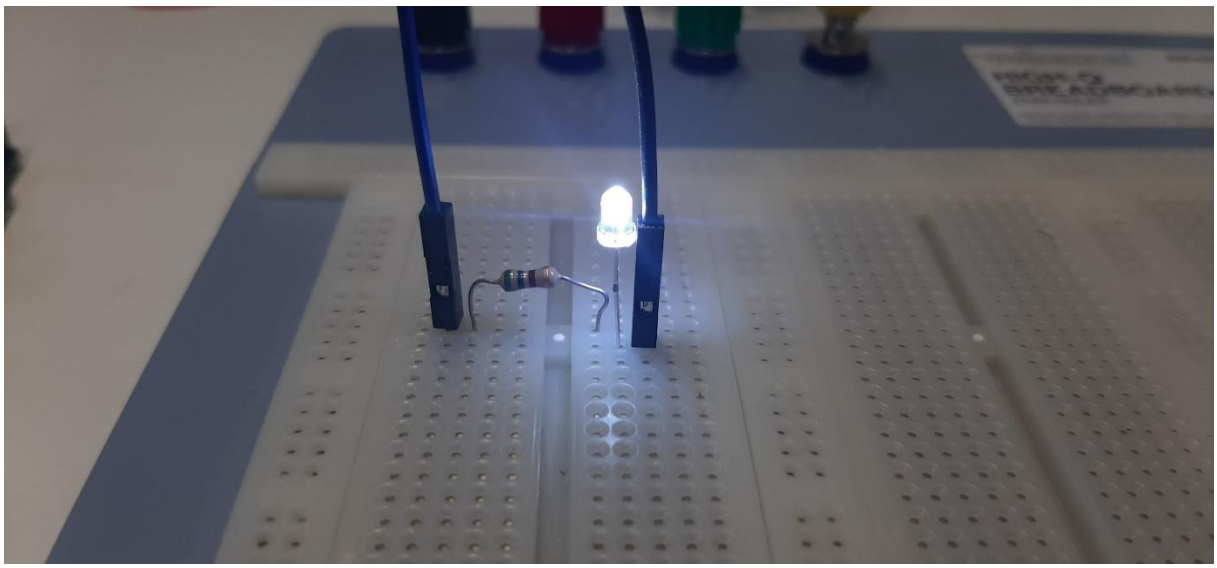

Figura 3 - montagem na breadbord

3.2. Exercício 2

O objetivo neste exercício é o mesmo do anterior, mas desta vez acrescentamos um potenciómetro onde quanto maior a potência mais luminosidade o led vai ter.
Começamos por fazer o projeto do circuito no Tinkercad (Figura 4) e através dessa montagem foi obtido o diagrama do circuito (figura 5), em seguida foi desenvolvido um algoritmo para o desenvolvimento do código no Arduíno e por fim a sua montagem na breadbord (figura 6).

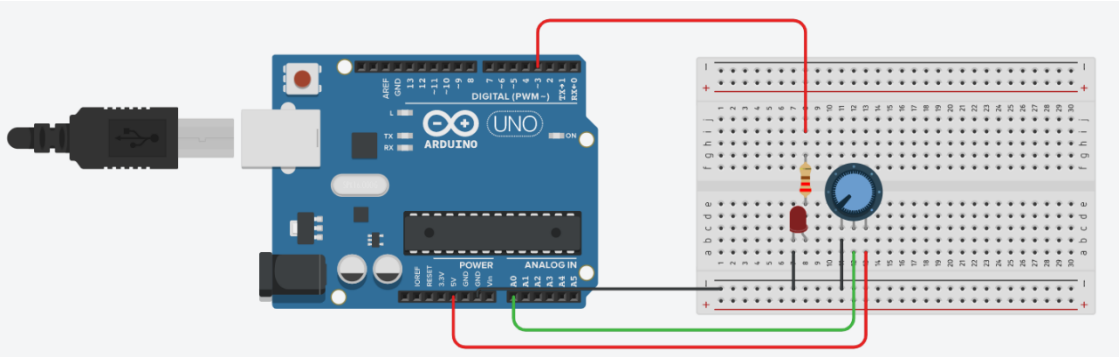

Figura 4 - montagem do circuito Tinkercad

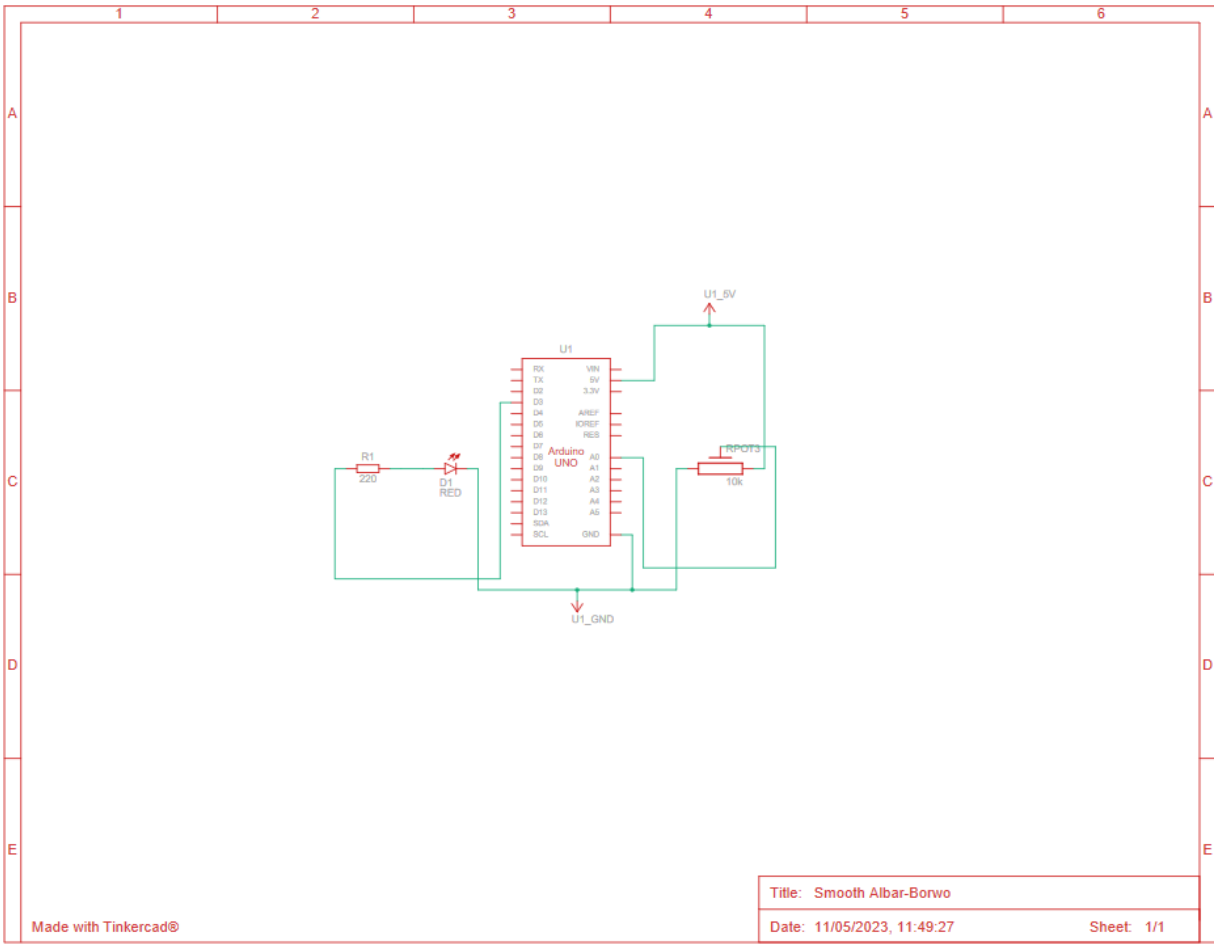

Figura 5 - diagrama do circuito

Algoritmo do programa:

1. Definir o pino digital 3 como uma saída no setup.
2. Inicia a função loop().
3. Lê o valor analógico do pino A0 usando analogRead(A0).
4. Divide o valor lido por 4 e arredonda-o para baixo (a função analogWrite() aceita valores de 0 a 255).
5. Definir o valor resultante como saída analogicamente através do pino 3 usando analogWrite(3, valor resultante).

O código representado abaixo foi o programa utilizado. O código começa com a definição do estado inicial do pino 3 como **OUTPUT.** Em seguida, ele entra numa função principal chamada **loop(),** que é executada continuamente. No **loop(),** o valor lido a partir do pino analógico A0 é convertido usando a função **analogRead()** e então dividido por 4 antes de ser acionado a um LED conectado ao pino digital 3 com a função **analogWrite().**
Essa divisão por 4 é necessária para ajustar a faixa dinâmica do sinal analógico (de 0 a 1023) à faixa de entrada aceitável do sinal **PWM** (0 a 255).

```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {
  analogWrite(3, analogRead(A0) / 4);
}
```

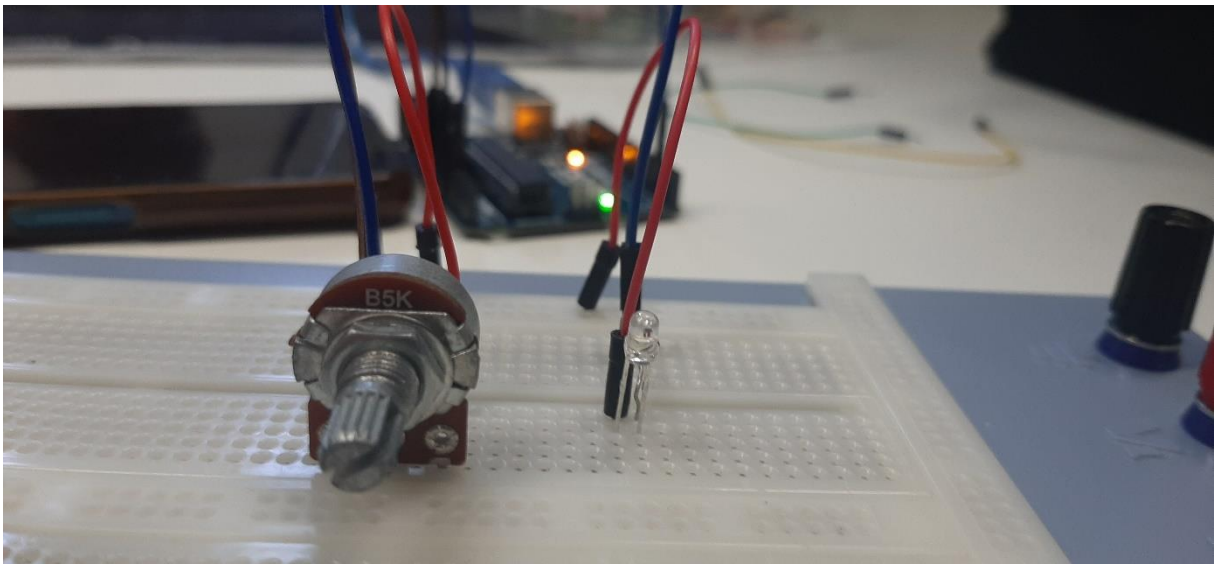

Figura 6 - montagem na breadbord

3.3. Exercício 3

O objetivo neste exercício é usar um pino **PWM** para controlar um servo motor.

Os servomotores têm três fios: alimentação, terra e sinal.
Começamos por fazer o projeto do circuito no Tinkercad (Figura 7) e através dessa montagem foi obtido o diagrama do circuito (figura 8), em seguida foi desenvolvido um algoritmo para o desenvolvimento do código no Arduíno e por fim a sua montagem na breadbord (figura 9).

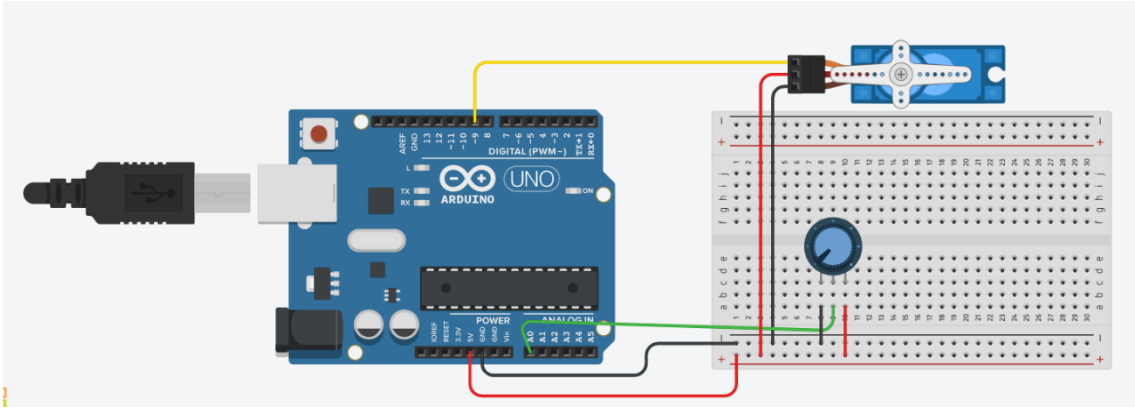

Figura 7 - montagem no tinkercad

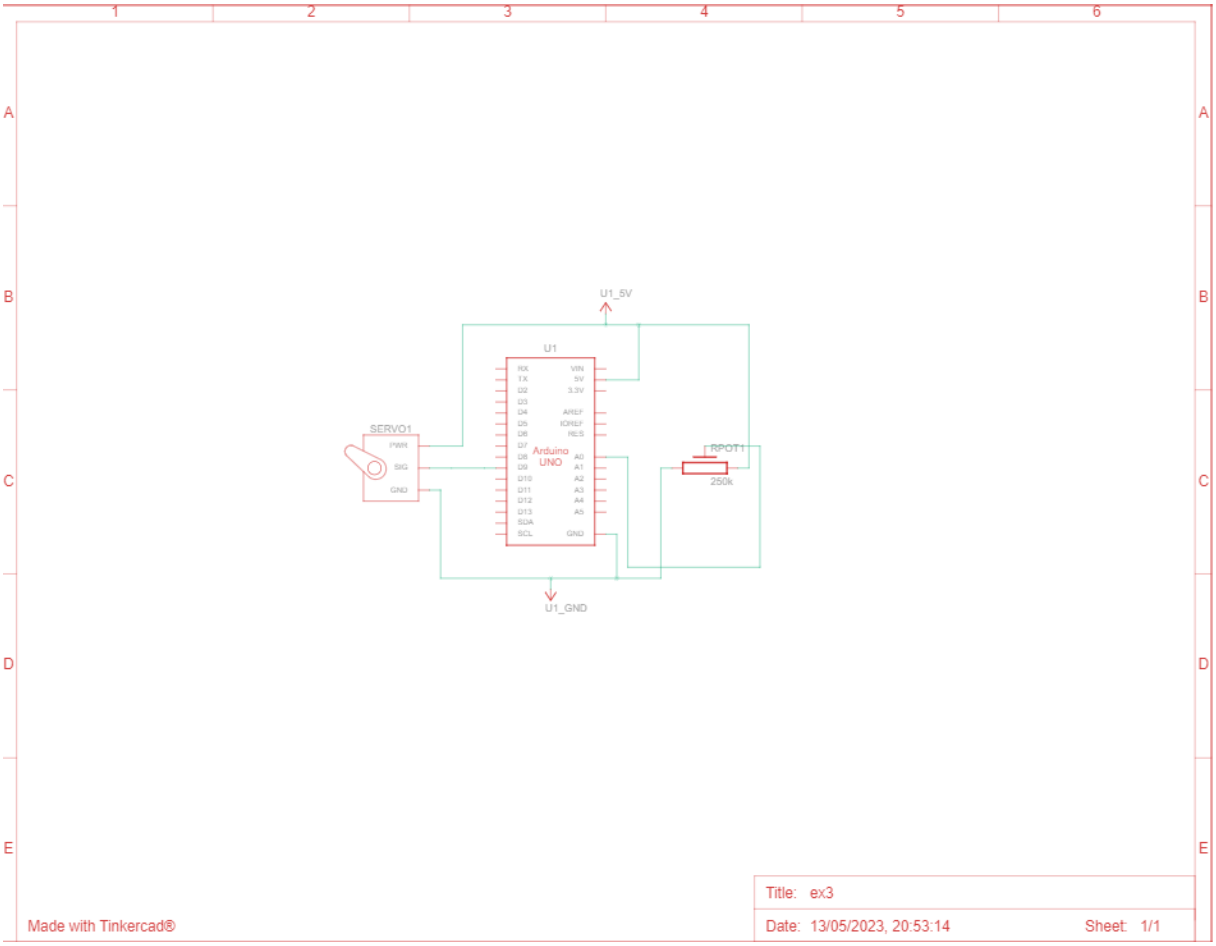

Figura 8 - diagrama do circuito

Algoritmo do programa:

1. Cria uma instância do objeto Servo chamada servo usando a biblioteca Servo.h: Servo servo.
2. Conecta o pino 9 da placa ao sinal de controle do servo motor usando servo.attach(9).
4. Inicia a função loop().
5. Lê o valor analógico do pino A0 usando analogRead(A0).
6. Usa a função map() para converter o valor lido, que varia de 0 a 1023, em um valor de ângulo correspondente de 0 a 180 graus.
7. Armazena o valor do ângulo calculado na variável angle.
8. Move o servo motor para a posição especificada pelo valor do potenciómetro usando servo.write(angle).

O código representado abaixo foi o programa utilizado. A biblioteca Servo está incluída no início do esboço. Esta biblioteca permite controlar servos enviando-lhes sinais **PWM** através dos pinos digitais da placa Arduino. Um objeto **Servo** é criado, denominado **servo**.
Na função **setup()**, o objeto servo é anexado ao pino 9, para que receba o sinal **PWM** através deste pino. Na função **loop()**, o código lê a tensão analógica do pino A0 usando a função **analogRead(A0)**. O valor lido vai de 0 a 1023 e depois é convertido para um valor entre 0 e 180 graus, correspondente à amplitude de movimento do servo. Isso é feito com a função **map()**. O servo motor move-se para a posição especificada pelo ângulo recebido.

```
#include <Servo.h>

Servo servo;

void setup() {
  servo.attach(9);
}

void loop() {
  int angle = map(analogRead(A0), 0, 1023, 0, 180);
  servo.write(angle);
}
```

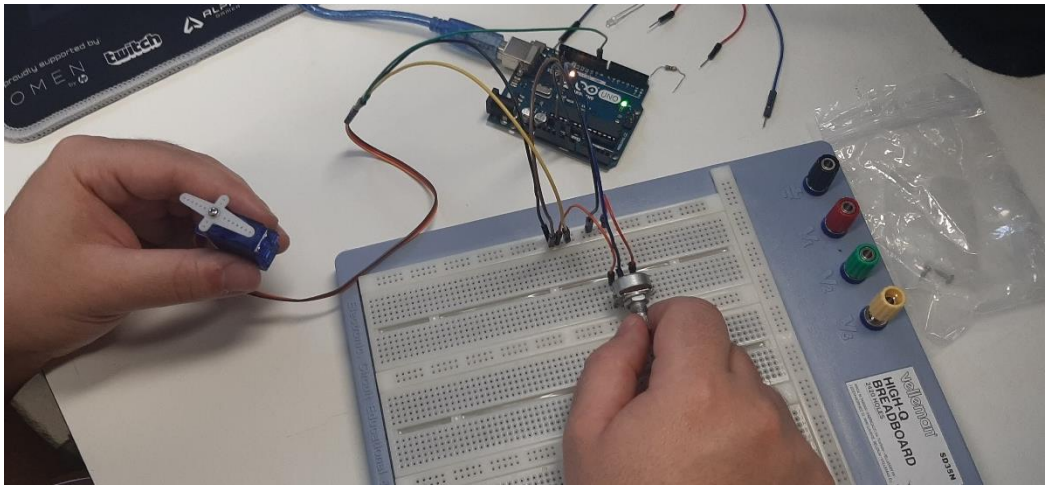

Figura 9 - montagem na breadbord

3.4. Exercício 4

O objetivo neste exercício é criar um programa de a acordo com o diagrama de estados (figura 10).
Começámos a desenvolver um algoritmo para o desenvolvimento do código no Arduíno para por fim procedermos a sua montagem na breadbord (figura 11).

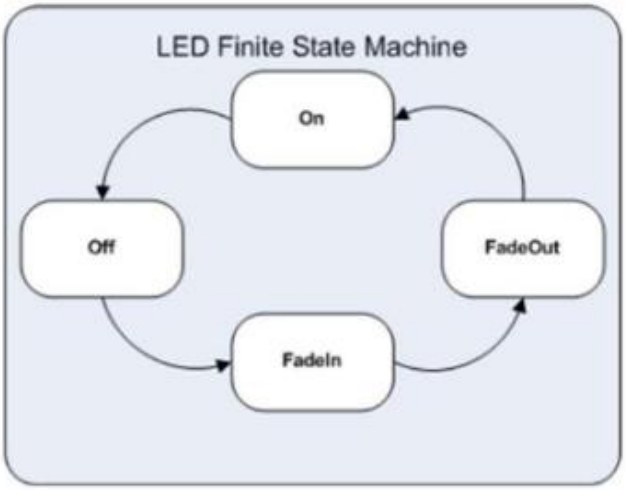

Figura 10 - diagrama de estados

Algoritmo do programa:

1. Definir a enumeração LED_STATE com os quatro estados possíveis da máquina de estados: ON, OFF, FADEIN e FADEOUT.
2. Definir protótipos das funções usadas pela máquina de estados.
3. Na função setup(), configuração da comunicação serial e o pino 3 como saída.
4. No loop principal, chamar a função state_machine() para atualizar o estado do LED.
5. A função state_machine() usa um switch-case para determinar o estado atual da máquina de estados e chama as funções correspondentes para verificar se é necessário mudar de estado.
6. As funções check_on(), check_off(), check_fadein() e check_fadeout() são responsáveis por verificar o tempo decorrido desde o último evento e determinar o próximo estado da máquina de estados.
7. Cada transição de estado ocorre quando condições específicas são atendidas, como um determinado tempo decorrido ou outra condição necessária.
8. A lógica de verificação de tempo funciona mantendo uma variável estática em cada estado para armazenar o tempo decorrido desde o último evento.
9. A máquina de estados é ciclicamente atualizada para permitir que o LED mude de estado quando as condições necessárias forem atendidas.

O código implementa uma máquina de estados finitos que controla o comportamento de um LED conectado ao pino digital 3 do Arduino. A máquina de estados implementa quatro estados possíveis: **ON** (acender), **OFF** (apagar), **FADEIN** (acender gradualmente) e **FADEOUT** (apagar gradualmente).
O código começa com a definição da **enum LED_STATE**, que contém os quatro estados possíveis, bem como a definição dos protótipos das funções usadas na máquina de estados.
Na função **setup(),** o código inicia a comunicação serial e configura o pino 3 como saída.
O loop principal chama a função **state_machine(),** que é responsável por verificar o estado atual e decidir qual a próxima ação a tomar.
As funções **check_on()**, **check_off()**, **check_fadein()** e **check_fadeout()** são responsáveis por verificar se as condições necessárias para fazer a transição para outro estado foram atendidas. Cada uma dessas funções retorna o próximo estado da máquina de estados, dependendo do estado atual e do tempo decorrido desde o último evento.
A função **state_machine()** usa um **switch-case** para determinar o estado atual e chamar a função correspondente para verificar se é necessário mudar de estado. O tempo decorrido desde o último evento é armazenado em uma variável estática para cada estado, para que a lógica de verificação de tempo funcione corretamente.

```
enum class LED_STATE {
  ON,
  OFF,
  FADEIN,
  FADEOUT
};

// prototypes
void state_machine();
LED_STATE check_on(unsigned long &time);
LED_STATE check_off(unsigned long time);
LED_STATE check_fadein(unsigned long time);
LED_STATE check_fadeout(unsigned long time);

void setup() {
  Serial.begin(9600);
  pinMode(3, OUTPUT);
}

void loop() {
  state_machine();
}

LED_STATE check_on(unsigned long &time) {
  time = millis();
  digitalWrite(3, HIGH);
  return LED_STATE::OFF;
}

LED_STATE check_off(unsigned long time) {
  if (millis() - time > 2000) {
```

```
    digitalWrite(3, LOW);
    return LED_STATE::FADEIN;
  }
  return LED_STATE::OFF;
}

LED_STATE check_fadein(unsigned long time) {
  if (millis() - time > 4000) {
    for (int i = 0; i <= 255; i++) {
      analogWrite(3, i);
      delay(10);
    }
    return LED_STATE::FADEOUT;
  }
  return LED_STATE::FADEIN;
}

LED_STATE check_fadeout(unsigned long time) {
  for (int i = 255; i >= 0; i--) {
    analogWrite(3, i);
    delay(10);
  }
  return LED_STATE::ON;
}

void state_machine() {
  static unsigned long time = millis();
  static LED_STATE state = LED_STATE::ON;
  switch (state)
  {
  case LED_STATE::ON: state = check_on(time);
    break;
  case LED_STATE::OFF: state = check_off(time);
    break;
  case LED_STATE::FADEIN: state = check_fadein(time);
    break;
  case LED_STATE::FADEOUT: state = check_fadeout(time);
    break;
  default: Serial.println("ERROR: Invalid state");
    break;
  }
}
```

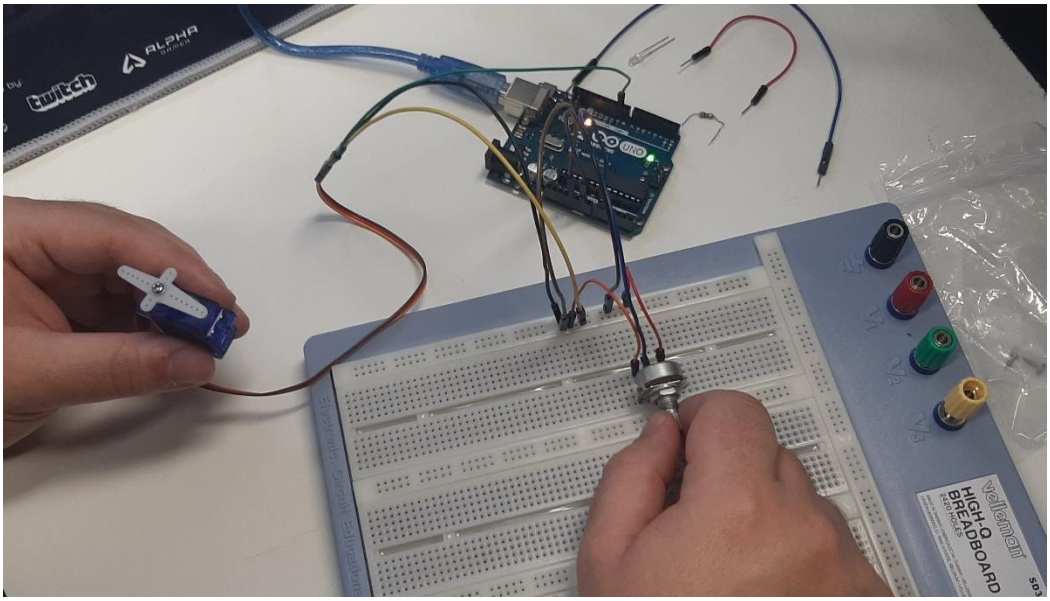

Figura 11 - montagem na breadbord

4. Discussão

Uma vez que o valor de saída ia ser analógico usamos a função **analogWrite** em um pino digital PWM, para podermos ajustar assim a largura do pulso elétrico enviado pelo pino para simular diferentes níveis de tensão analógica na saída. Usando essa técnica, podemos controlar o brilho de um LED, a velocidade de um motor ou qualquer outro componente que possa ser controlado por um sinal analógico.

5. Conclusão

Assim sendo podemos concluir que cumprimos os objetivos de todos os exercício e que foram bons exemplos de como usar a função **map()**, o **PWM** e a maquina de estados.

6. Referências

"Arduino - Servo Motor Controlled by Potentiometer | Arduino Tutorial." Arduino Getting Started, arduinogetstarted.com/tutorials/arduino-servo-motor-controlled-by-potentiometer. Accessed 11 May 2023.

"Arduino Reference." Arduino.cc, 7 Sept. 2020, www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/.Accessed 11 May 2023

"Basics of PWM (Pulse Width Modulation) | Arduino Documentation." docs.arduino.cc/learn/microcontrollers/analog-output. Accessed 11 May 2023.

"Arduino Reference." Arduino.cc, 7 Sept. 2020, www.arduino.cc/reference/en/language/functions/math/map/.Accessed 11 May 2023.

Vídeos:
https://www.youtube.com/playlist?list=PLweUC19fZUoYcm8HDXDZWL0I-U0Bx7SJn