Licenciatura Engenharia Informática e Multimédia
Instituto Superior de Engenharia de Lisboa
Ano letivo 2022/2023

Sensores e Atuadores

Relatório: Trabalho Lab05

Turma: 11D	Grupo: 0
Nome: Daniel Silva	Número: 50781
Nome: João Ramos	Número: 50730
Nome: Miguel Alcobia	Número: 50746

Data: 5 de Dezembro 2022

Objetivo:

Esta experiência teve em vista os alunos aprenderem a implementar novos atuadores em circuitos ligados à placa Arduino.

Material:

Breadboard

Arduino Uno

Resistências 220Ω e 10kΩ

Potenciómetro 10kΩ

LDR

LED RGB

Servo

Sonar

Buzzer

Piezo

Osciloscópio

Preparação teórica:

1-

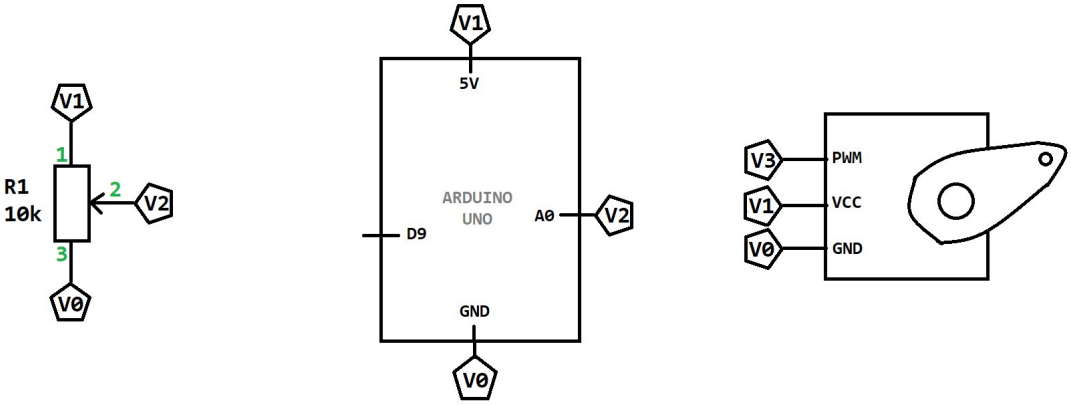


Figura 1 - Esquema do Circuito 1

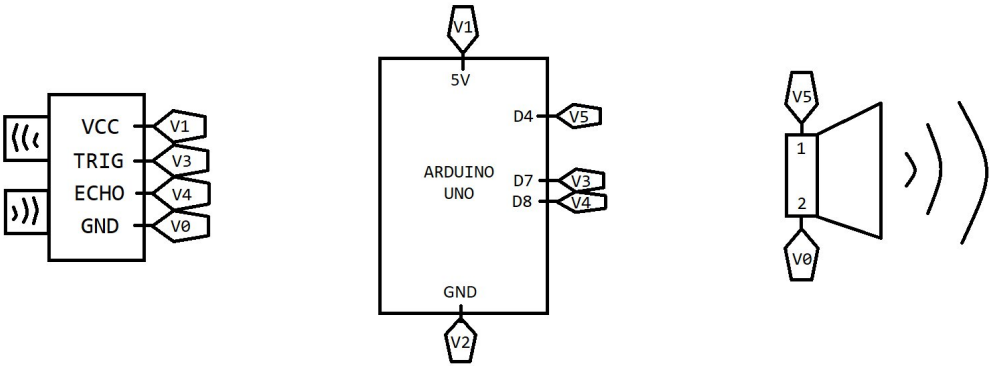


Figura 2 - Esquema do Circuito 2

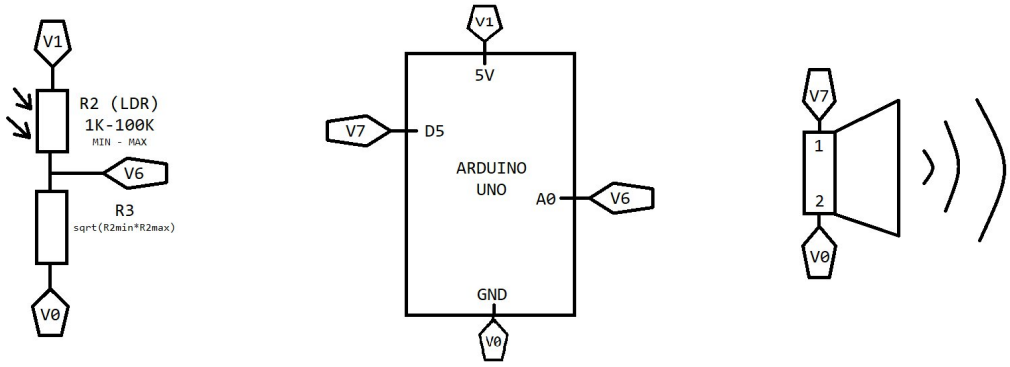


Figura 3 - Esquema do Circuito 3

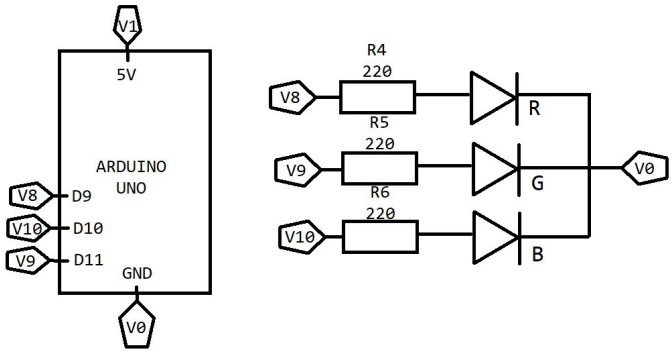


Figura 4 - Esquema do Circuito 4

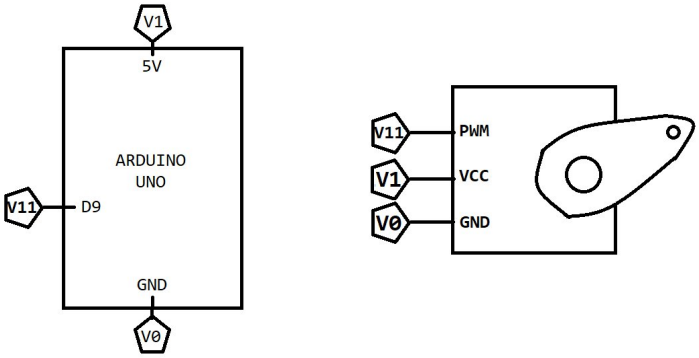


Figura 5 - Esquema do Circuito 5

3-

Circuito 1

Neste circuito, incluímos a biblioteca `Servo.h`, para conseguirmos usar o servo. Começou-se por definir os pinos da devida forma e usar o comando `Servo myservo` para criar um objeto para criar o servo. A variável `val` servirá para ler o valor do pino do potenciómetro. Em seguida conectou-se o servo ao pino do potenciómetro através do `myservo.attach()`.
Por fim, leu-se a tensão do pino do potenciómetro e depois mapeou-se esse valor para o intervalo desejado. Vale notar que o servo não se mexerá da posição 0 a 0.5 do potenciómetro, pois os números negativos não conseguem ser lidos. Uma alternativa a este problema seria mapear para um intervalo de 0 a 180.

```
#define PINPOT A0
#include <Servo.h>

Servo myservo;

int val;

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

void loop() {
  val = analogRead(pinpot);
  val = map(val, 0, 1023, -90, 90); //map(val, 0, 1023, 0, 180)
  myservo.write(val);
  delay(15);
}
```

Circuito 2

Neste circuito, começamos, como sempre, por definir os pinos que precisavam deste procedimento. Criou-se a função **ValorSonar()** para retomar o valor da distância. Para tal, começou-se por usar um **digitalWrite** com valor HIGH para ativar o pino do trigger e após 10µs desativou-se o pino. A duração é igual ao valor do **pulseIn**, que lia um pulso enquanto o pino echo tivesse um valor HIGH. Depois a distância era calculada em centímetros, logo como a velocidade do som é de 343m/s, passa a 29µs/cm e como o som percorre a distância duas vezes, ao ir e ao voltar, divide-se o quociente de **duracao/29** por 2.
Para concluir o programa criou-se a função que verificava se a distância estava entre 5 e 50cm e se estivesse, o buzzer emitia uma frequência proporcional num intervalo de 0.5 a 20 Hz.

```
#define trig 7
#define echo 8
#define buzz 4

int aux;
unsigned long comp;
unsigned long dist;

void setup(){
  Serial.begin(9600);
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(buzz,OUTPUT);
}

float ValorSonar(){
  digitalWrite(trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig,LOW);
  duracao= pulseIn(echo,HIGH);
  dist= duracao/29/2; //dist em cm
  Serial.println(dist);
  return dist;
}

void piezo(){
  if(dist>=5 && dist<=50){
    tone(buzz, map(dist,5,50, 5, 200)/10);
  } else{
    noTone(buzz);
  }
}

void loop(){
 ValorSonar();
 piezo();
}
```

Circuito 3

Para controlar a frequência do piezo pelo brilho do LDR, fez-se o seguinte: Criou-se a função **LerLDR** que calcula o brilho e mapeia o seu valor para percentagem. Depois criou-se a função **piezo** que faz com que o piezo emita uma frequência proporcional ao brilho do LDR, num intervalo de 0.2 e 10 kHz**.** No final, chamou-se as funções no loop.

```
#define LDR A0
#define PIEZO 5
float brilho;

void setup(){
  pinMode(PIEZO, OUTPUT);
}

float LerLDR()
{
  float V=analogRead(LDR)*5.0/1023;
  float RLDR=10.0*(5.00/V-1);
  float b = pow(RLDR/23.48,-1/0.837);
  return map(b,0.01,6.25,0,100);
}

void piezo(float brilho)
{
  tone(PIEZO,(map(brilho,0,100,0.2,10)*1000));
}

void loop(){
  brilho=LerLDR();
  delay(500);
  piezo(brilho);;
}
```

Circuito 4

Depois de definir os pinos, no *setup* colocou-se todos em modo OUPUT, seguidos de uma mensagem para inserir os valores na consola.
Para receber esses valores criou-se a função **`RecebeconsolaR`** que enquanto o **`Serial.available`** fosse igual a zero, não fazia nada e o valor que fosse inserido na consola, era atribuído à variável Red, Green, Blue, nesta ordem. No final, a função fazia *return* dos valores introduzidos.
Criou-se a função **`OutputLine`** com a única finalidade de fazer *output* dos valores no formato R000G000B000.
A última função criada foi a **`ControlaBrilho`** que mapeava os valores de percentagem introduzidos para valores entre 0 a 255, fazendo depois um **`analogWrite`** com esse valor no pino da respetiva cor.

```
#define R 9
#define B 10
#define G 11
int Red;
int Blue;
int Green;

void setup() {
  Serial.begin(9600);
  pinMode(R,OUTPUT);
  pinMode(B,OUTPUT);
  pinMode(G,OUTPUT);
  Serial.println("Insira o valor da percentagem de brilho para os pinos R,G e B:");
}

int RecebeconsolaR() {
  while(Serial.available()==0){}
  Red= Serial.parseInt();
  while(Serial.available()==0){}
  Green= Serial.parseInt();
  while(Serial.available()==0){}
  Blue= Serial.parseInt();

  return Red, Green, Blue;
}

float OutputLine(int Red,int Green,int Blue) {
  Serial.print("R"+ String(Red) + "G" + String(Green) + "B" + String(Blue));
}

float ControlaBrilho(int Red,int Green,int Blue) {
  int Red2 = map(Red,0,100,0,255);
  int Green2 = map(Green,0,100,0,255);
  int Blue2 = map(Blue,0,100,0,255);
  analogWrite(R,Red2);
  analogWrite(G,Green2);
  analogWrite(B,Blue2);
}

void loop() {
 RecebeconsolaR();
 ControlaBrilho(Red,Green,Blue);
 OutputLine(Red,Green,Blue);
 Serial.println(" ");
 }
```

Exemplo de janela de Output

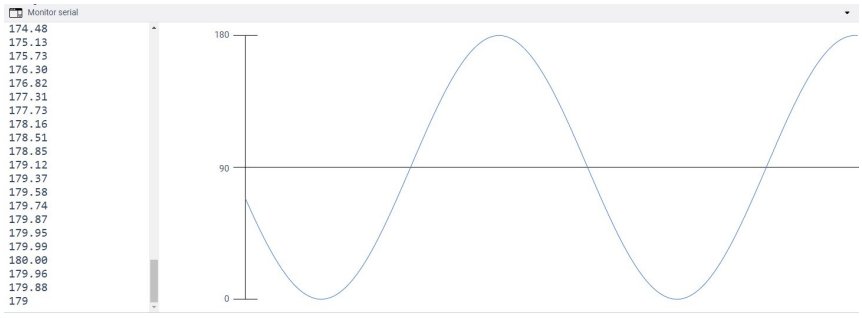
```
Insira o valor da percentagem de brilho de LED para os pins R,G e B respetivamente:
R12G45B90
```

Circuito 5

Depois de definir os pinos, no *setup* conectou-se o servo ao pino 9. A única função criada foi a **`simulador`** que recebia o tempo em milissegundos. Definimos três constantes: ydc = $\frac{ymáx + ymin}{2} = 90$; yac = $\frac{ymáx - ymin}{2} = 90$ e o período (T) foi definido para 5 segundos.
No final a função retornava a função do sinal segundo a fórmula: y(t) = $yac * \sin\left(2\pi\frac{t}{T}\right) + ydc$.
No ciclo *loop* usou-se o **`.write`** para o servo interpretar os valores da função sinusoidal.

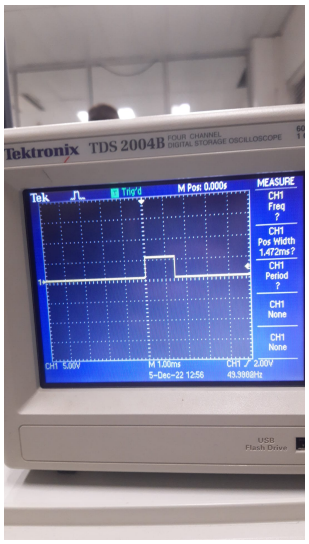
```
#include <Servo.h>
#define servo 9

Servo nameServo;

void setup()
{
  Serial.begin(9600);
  nameServo.attach(9);
}

float simulador(unsigned long t)
{
  const float ydc=90, yac=90;
  const int T=5; // T=5 segundos
  return ydc+yac*sin(6.28*t/1000/T);
}

void loop(){
  unsigned long t=millis();
  nameServo.write(simulador(t));
  Serial.println(simulador(t));
  delay(15);
}
```

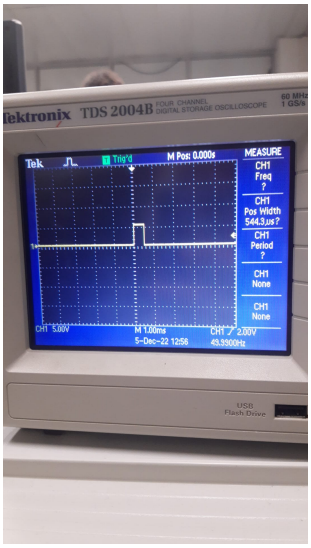
Serial e Plotter:



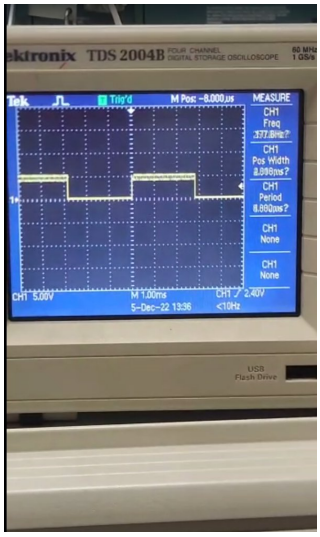
Tarefas:

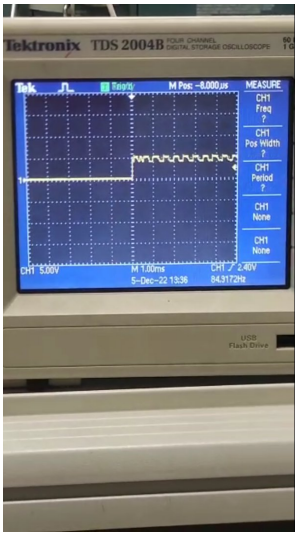
2-
Circuito 1



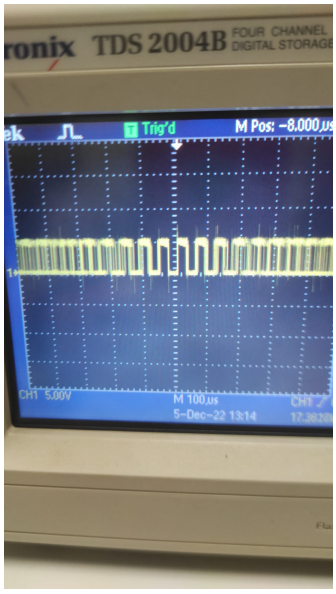


Circuito 2

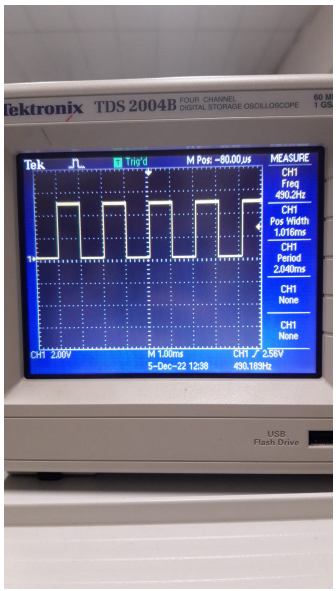


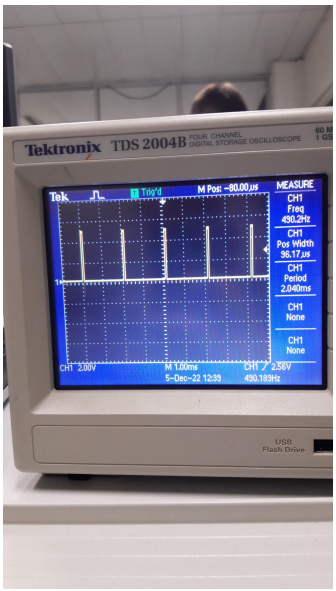


Circuito 3



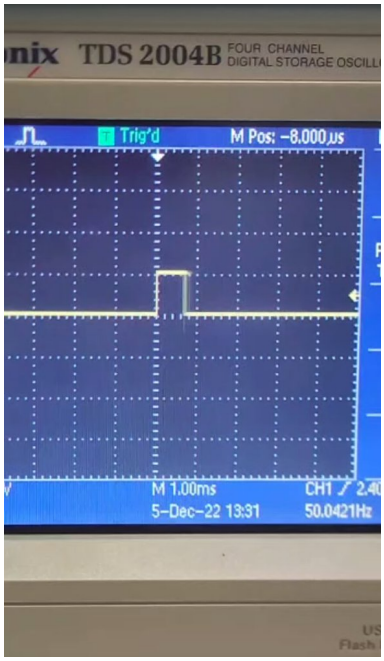
Circuito 4

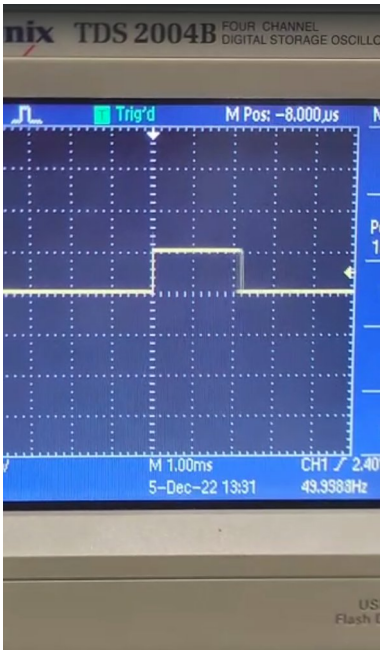




Nestre circuito, houve um problema pois os pinos Blue e Green estavam trocados no próprio LED, mas procedeu-se às devidas alterações para contornar o problema.

Circuito 5





Conclusão:

Houve algumas dificuldades ao trabalhar com o osciloscópio, além do problema que já foi referido em relação ao LED RGB. Porém este trabalho serviu para os alunos aprenderem a mexer e a conhecer novos atuadores e novos códigos a implementar no Arduino.