Licenciatura Engenharia Informática e Multimédia

Instituto Superior de Engenharia de Lisboa

Ano letivo 2022/2023

Sensores e Atuadores

Relatório: Trabalho Lab05

Turma: 11N

Grupo: 0

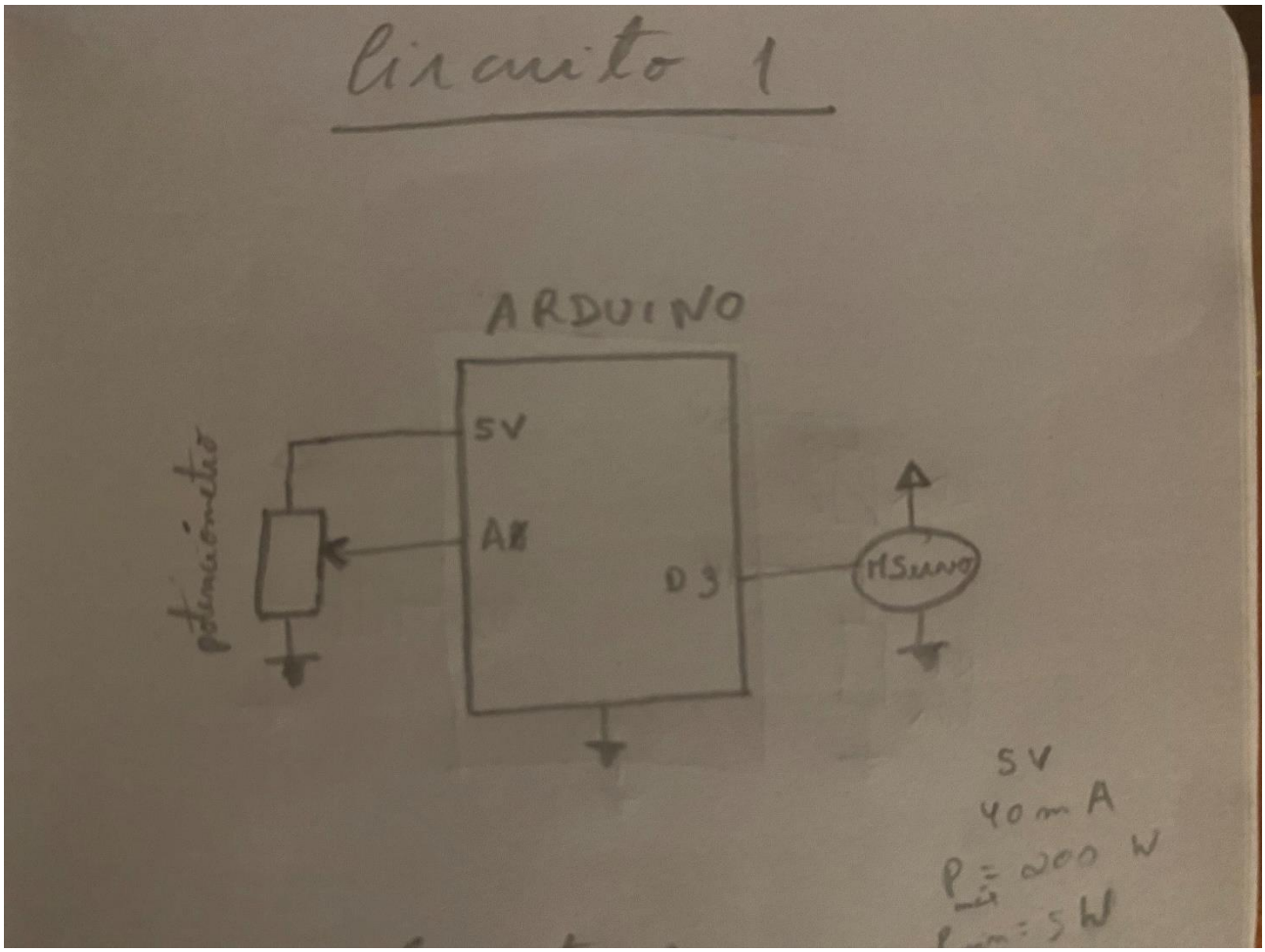
Nome: Rodrigo Coelho Número: 50251

Nome: Tatiana Damaya Número: 50299

Data: 8 de dezembro 2022

Circuito 1: Potenciómetro e Servo

Diagrama



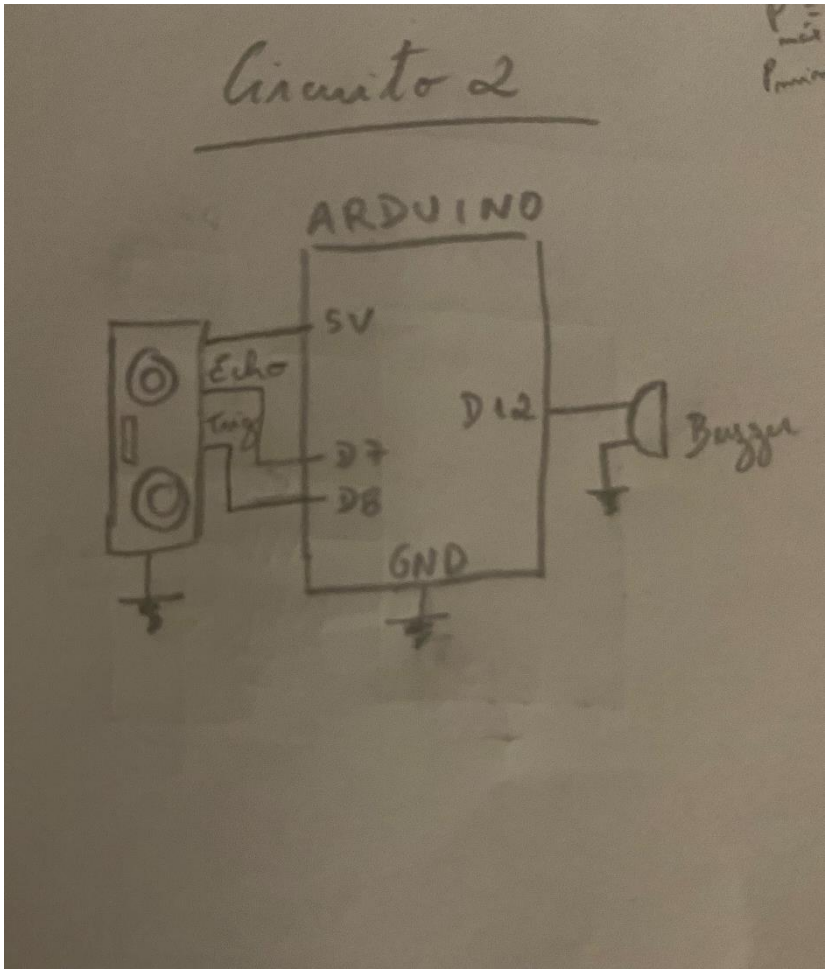
Código

```
// Circuito 1: Servo - Potenciometro
#include <Servo.h>
#define PINPOT A0

Servo servo;

void setup()
{
  servo.attach(3);
  Serial.begin(9600);
}

void loop()
{
  int aR = analogRead(PINPOT);
  // Rodar entre -90 e 90 graus
  int val = map(aR, 0, 1023, 0, 180);
  servo.write(val);
  delay(15);
}
```

Aqui foi nos requisitado para que o potenciómetro se influencia a posição do servo, para tal introduzimos um analogRead ao pino do pot para então convertê-lo para valores de -90 a 90 graus através de o auxílio de um map para então passar-lo para o método write do objeto servo, que resulta na mudança de posição do componente.

Circuito 2: Sonar e Buzzer

Diagrama



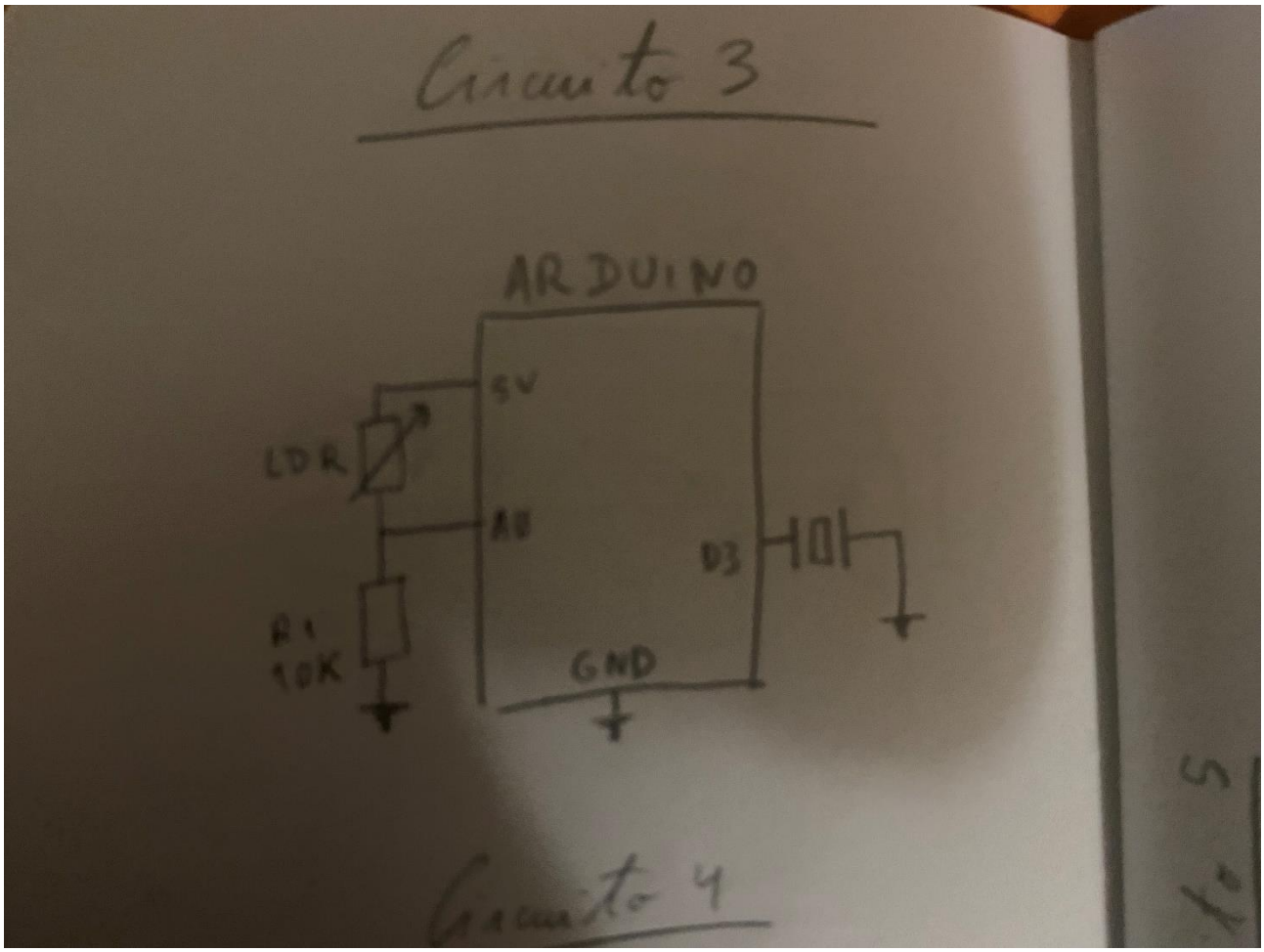
Código

```
// Circuito 2: SONAR - BUZZER
#define PINBUZZ 2
#define PINTRIGGER 12
#define PINECHO 13
// Criar o pico do buzzer
void beep() {
  digitalWrite(PINBUZZ, HIGH);
  delay(50);
  digitalWrite(PINBUZZ, LOW);
}
void setup() {
  Serial.begin(9600);
  pinMode(PINECHO, INPUT);
  pinMode(PINTRIGGER, OUTPUT);
  pinMode(PINBUZZ, OUTPUT);
}
void loop() {
  digitalWrite(PINTRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(PINTRIGGER, LOW);
  int duration = pulseIn(PINECHO, HIGH);
  int distance = (duration/2) / 29.1;
  // Verificar se está entre a distancia compreêndida, 5cm - 50cm
  if(distance <= 50 && distance >= 5) {
    int freq = map(distance, 5, 50, 0.5, 20);
    delay(freq); // Changes with the freq
    beep();
    // See the current distance
    Serial.print("Distance: ");
    Serial.println(distance);
  } else {
    digitalWrite(PINBUZZ, LOW);
  }
```

Aqui usamos uma função beep para criar a funcionalidade base do buzzer que é apitar, neste caso durante um intervalo de 50ms. Depois usamos o sonar para obtermos a distância entre 5 a 50cm que irá ser convertida em frequência e será então usada para alterar o período do beep, ou seja, para quanto menor a distância maior o número de beeps e o contrário se aplica para quanto maior for a distância, usando o delay e passando a frequência como parametro.

Circuito 3: LDR e Piezo

Diagrama



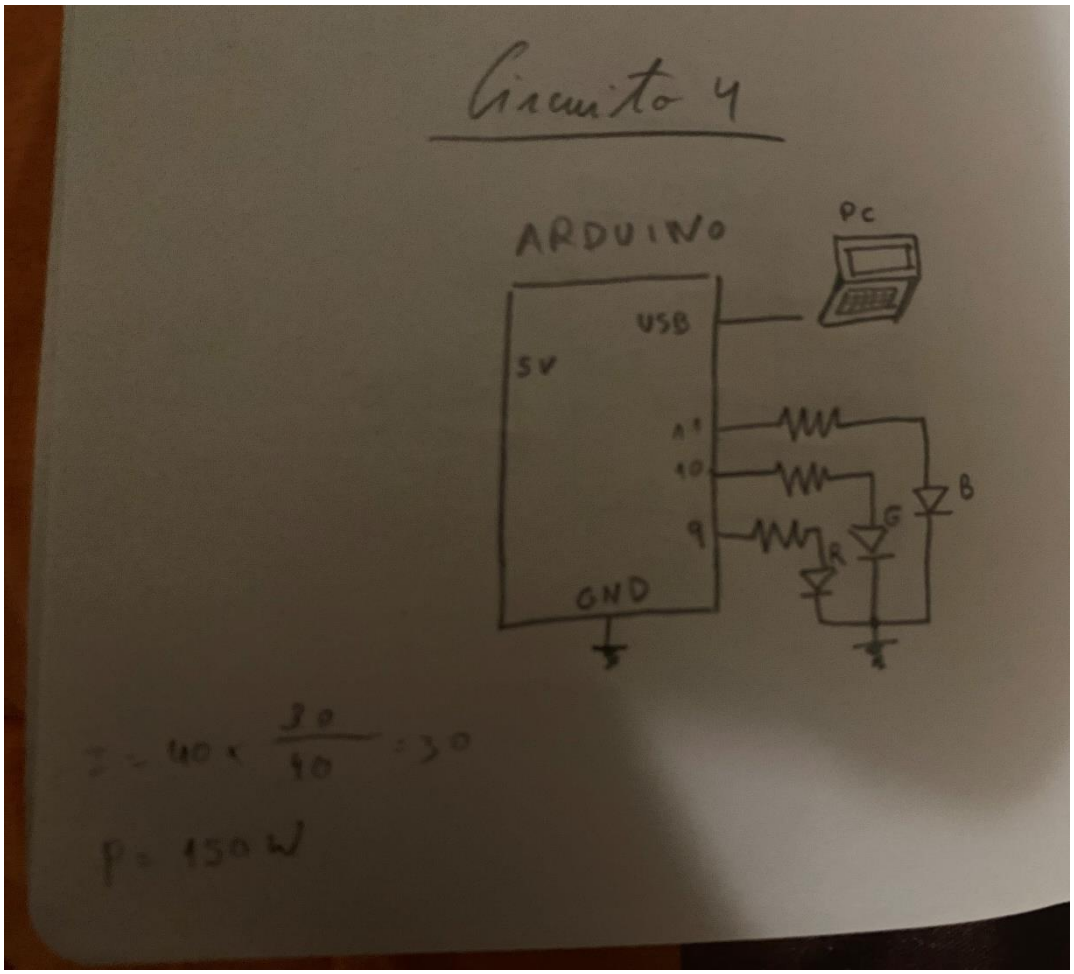
Código

```
// Circuito 3: LDR - PIEZO
#define PINLDR A0
#define PINPIEZO 9

void setup()
{
  pinMode(PINLDR, INPUT);
  Serial.begin(9600);
  pinMode(PINPIEZO, OUTPUT);
}

void loop()
{
  int ldrValue = map(analogRead(PINLDR), 0, 1023, 0 ,100); // percent
  int freq = map(ldrValue, 0, 100, 200, 10000); // freq
  Serial.println(freq);
  tone(PINPIEZO, freq);
}
```

Neste circuito é pedido para alterar a frequência do piezo através de um ldr, para tal usámos dois maps, onde o primeiro transforma o valor entre 0 e 100 percento e o segundo transforma essa percentagem em frequência de 200 a 10000 Hz (0.2 – 10 kHz). Por fim passamos o método tone para realizar a ação pretendida.

Circuito 4: LEDRGB e Serial

Diagrama



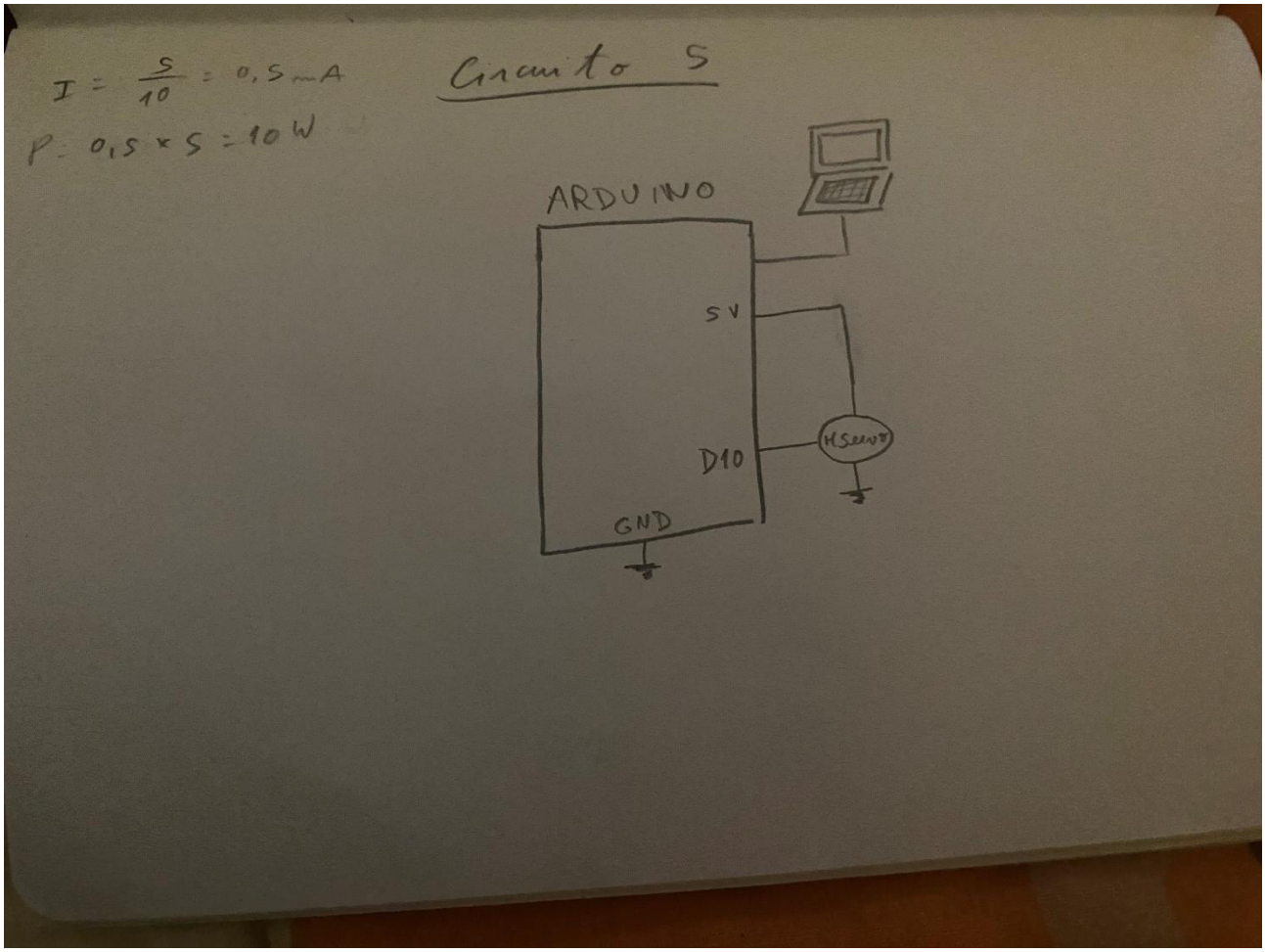
Código

```
// Circuito 4: LEDRGB - Serial
#define PINLEDBLUE 9
#define PINLEDGREEN 10
#define PINLEDRED 11
#define BAUD_RATE 9600
// Global Variables
unsigned int aR;

void setup() {
  // Pin Config
  pinMode(PINLEDRED, OUTPUT);
  pinMode(PINLEDGREEN, OUTPUT);
  pinMode(PINLEDBLUE, OUTPUT);
  Serial.begin(BAUD_RATE);
}
void rgbColor(float red, float green, float blue) {
  analogWrite(PINLEDBLUE, blue * 2.55);
  analogWrite(PINLEDRED, red * 2.55);
  analogWrite(PINLEDGREEN, green * 2.55);
}
void loop() {
  if(Serial.available()) {
    float red = 0, green = 0, blue = 0;
    red = Serial.parseInt();
    Serial.println(red);
    green = Serial.parseInt();
      Serial.println(green);
      blue = Serial.parseInt();
      Serial.println(blue);
    rgbColor(red, green, blue);
  }
}
```

Aqui o objetivo é inserir um input que faz variar a percentagem de luz de cada cor para produzir cores diferentes.
Para tal implementamos uma função rgbColor que contem 3 parametros que correspondem a cada cor, red, green e blue e usamos o analogWrite para passar essas três opções para criar a ação desejada. A multiplicação de 2.55 serve para que os valores são sempre entre 0 e 255.
Por fim dentro do if cuja condição é o retorno do booleano devolvido pelo Serial.available, usamos então o Serial.parseInt para obter o input de cada cor sendo a primeira do vermelho a segunda do verde e a terceira do azul que são usados na função rgbColor.

Circuito 5 – Simulação e Servo

Diagrama



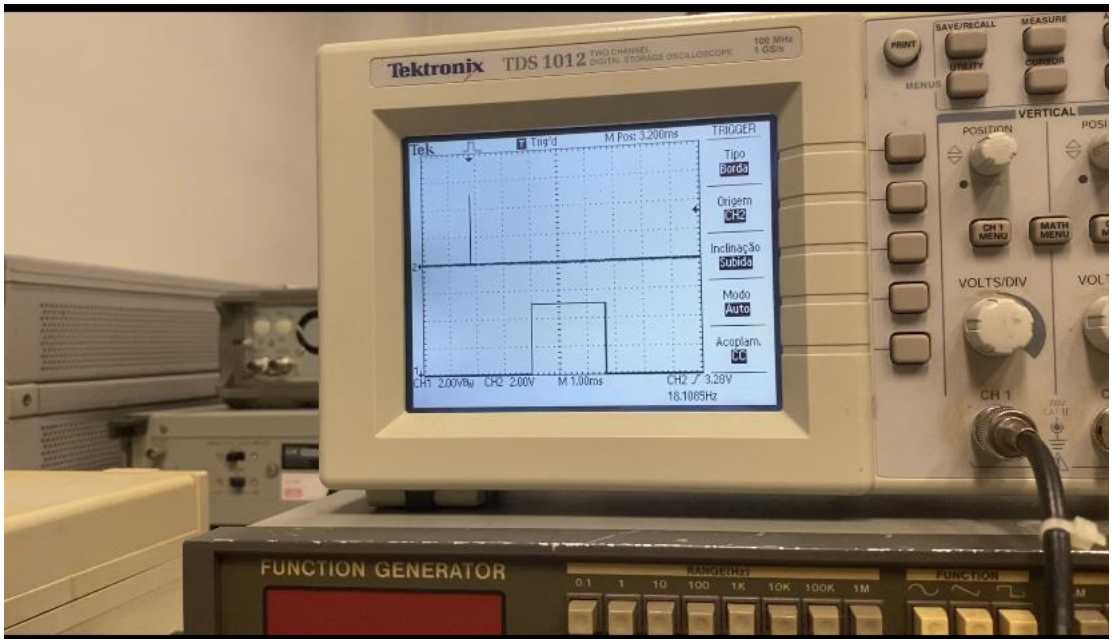
Código

```
// Circuito 5: Onda sinusoidal
#include <Servo.h>
#define PINSERVO 10

Servo servo;
int pos = 0;

void setup() {
  servo.attach(10);
    Serial.begin(9600);
}

void loop() {
  for(pos = 0;pos <= 180; pos += 5) {
    servo.write(pos);
    float wave = 90 * sin((PI/5.0) * pos);
    Serial.print(wave);
  }
  for(pos = 180;pos >= 0; pos -= 5) {
    servo.write(pos);
    float wave = 90 * sin((PI/5.0) * pos);
    Serial.print(wave);
  }
}
```

Aqui rodamos o servo para obter uma simulação de uma onda triangular. Usámos dois for loops para rodar a posição entre 0 e 90 da esquerda para a direita e da direita para a esquerda e introduzimos um float wave para calcular a onda sinusoidal usando a fórmula da função desta mesma. Este é apresentado através do serial plotter que obtem os dados a partir do serial.println.

Testes

Para o circuito 2 realizamos testes com o osciloscópio e verificamos a correta funcionalidade do código pois nele verificou-se o aumento e diminuição do período consoante a distância de um sólido ao sonar.
Para tal usámos dois crocodilos para medir a tensão do sonar e a do buzzer que foram sintonizados ao canal 2 e 1 respetivamente. O segundo canal ficou sincronizado com o primeiro de maneira em que o sinal do buzzer começava somente a ser lido quando era detetado o do sonar.



A parte superior representa o pico do sonar e a zona inferior o buzzer

Verificamos então que ao afastarmos/aproximarmos um objeto do sonar faria aumentar/diminuir o período da onda quadrática do sinal correspondente ao buzzer. E como curiosidade foi também verificado um segundo sinal do sonar para quando este estava fora do alcance pretendido que para esta expericencia era 50cm. Isto acontece porque o primeiro não teve retorno então por defeito o sonar lança um segundo para verificar se este tem resposta

No circuito 4 também foram realizados testes com o osciloscópio, mas com código diferente que tem como objetivo de este fluir sozinho, sem input do usuário.

```
void newRGB() {
  int green = 255;
  int blue = 255;
  int red = 0;
  for (green = 255; green > 0; green -= 5)
    analogWrite(PINLEDGREEN, green);
    delay(50);
  }
  for(red = 0; red < 255; red+=5) {
      analogWrite(PINLEDRED, red);
      delay(50);
  }
  for(blue = 255; blue > 0; blue -= 5) {
      analogWrite(PINLEDBLUE, blue);
      delay(50);
  }
  for(red = 255; red > 0; red -= 5) {
      analogWrite(PINLEDRED, red);
      delay(50);
  }
  for(green = 0; green < 255; green+=5) {
      analogWrite(PINLEDGREEN, green);
      delay(50);
  }
  for(blue = 0; blue < 255; blue+=5) {
      analogWrite(PINLEDBLUE, blue);
      delay(50);
  }
```

Esta função então altera os valores de cada cor decrementado ou incrementado consoante o valor corrente de cada cor, se é 0 ou 255. O que nos permite presenciar toda a gama de cores do espectro luminoso.

No uso do osciloscópio comentamos o green e só trabalhos com as cores vermelho e azul, isto deve-se ao facto do equipamento só ter dois canais e não conseguirmos ver então as três cores ao mesmo tempo. Na experiência vimos então os um absurdo número de picos com períodos muitíssimo reduzidos, na ordem do 1 microssegundos que correspondem ao sinal alternado da corrente do led cujo seus “on” e “off” são tao rápidos e contínuos que para o ser humano para que este teve sempre ligado, sem qualquer interrupção, criando a ilusão ótica que vemos no dia a dia.