



# **Licenciatura em Engenharia Informática**

## **Tecnologia e Arquitetura de Computadores 2022/2023**

### **Trabalho Prático nº 3**

## **HC-SR04 and Relay Module**

**Realizado em: 23/03/2023**

**Elaborado em: 23/03/2023**

**Grupo: 5**

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

# Índice

<b>1. Introdução .....</b>	<b>3</b>
<b>2. Métodos.....</b>	<b>3</b>
<b>3. Resultados.....</b>	<b>4</b>
<b>3.1. Exercício 1 .....</b>	<b>4</b>
<b>3.2. Exercício 2 .....</b>	<b>6</b>
<b>3.3. Exercício 3 .....</b>	<b>8</b>
<b>3.4. Exercício 4 .....</b>	<b>9</b>
<b>4. Discussão.....</b>	<b>10</b>
<b>5. Conclusão .....</b>	<b>10</b>
<b>6. Referências .....</b>	<b>11</b>

# I. Introdução

Este trabalho tem como objetivo fazer 4 exercícios, nos dois primeiros exercícios terá como objeto a introdução a um sensor na qual vamos ter que perceber como trabalhar com ele. Nos dois últimos exercícios, vamos introduzir relay, vamos tentar perceber o seu funcionamento e a sua utilidade.

## 2. Métodos

O trabalho foi realizado no decorrer das 3 horas de aula de **Tecnologia e Arquitetura de Computadores (TAC)** tendo sido utilizado o **Tinkercad** para projetar o circuito, o **Arduino IDE** para o desenvolvimento do código, todos estes programas foram desenvolvidos num computador com um processador AMD Ryzen 7 5800H With Radeon Graphics e também foram usados os materiais disponíveis no laboratório para montagem e testagem dos circuitos.

Nome	Quantidade	Componente
UI	1	Arduino Uno R3
DIST	1	Sensor de distância
buzzer	1	buzzer

Tabela 1 - materiais do exercício 1

Nome	Quantidade	Componente
UI	1	Arduino Uno R3
D1, D2,	2	Red and Blue LED
R1, R2	2	1 kΩ Resistor
buzzer	1	buzzer
DIST	1	Sensor de distância

Tabela 2 - materiais do exercício 2

Nome	Quantidade	Componente
UI	1	Arduino Uno R3
D1, D2,	2	Red and Blue LED
R1, R2	2	1 kΩ Resistor
Relay	1	Relay

Tabela 3 - materiais dos exercícios 3 e 4

## 3. Resultados

### 3.1. Exercício I

O objetivo neste exercício é simular um sistema de sensor de distância, onde o buzzer deve ser acionado com um intervalo de 2 segundos se a distância  $< 50$  cm, 1 segundo se a distância  $< 25$  cm e permanentemente se a distância  $< 10$  cm.

Começamos por fazer o projeto do circuito no Tinkercad (Figura 1) passamos em seguida para o código no Arduino (Figura 3) para em seguida procedemos para a construção do mesmo (Figura 2).

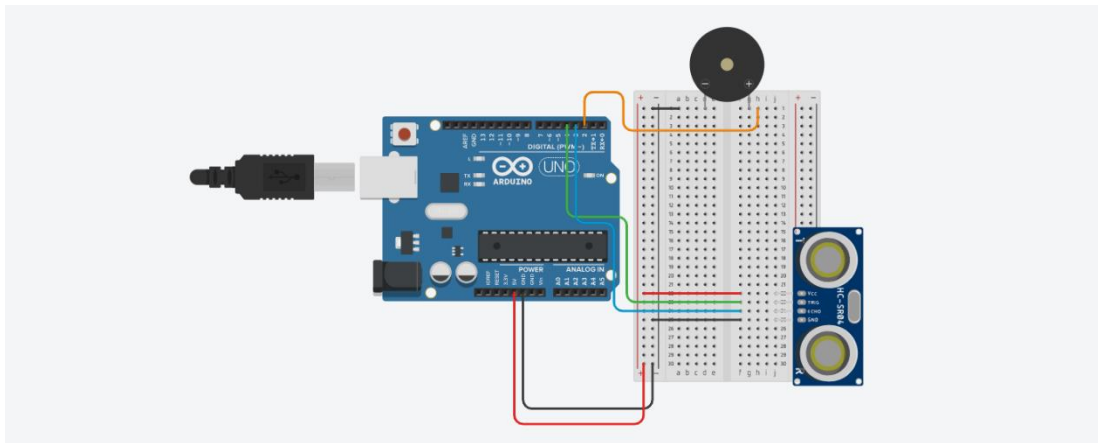


Figura 1 - 1\_HC\_SR04 TInkercad

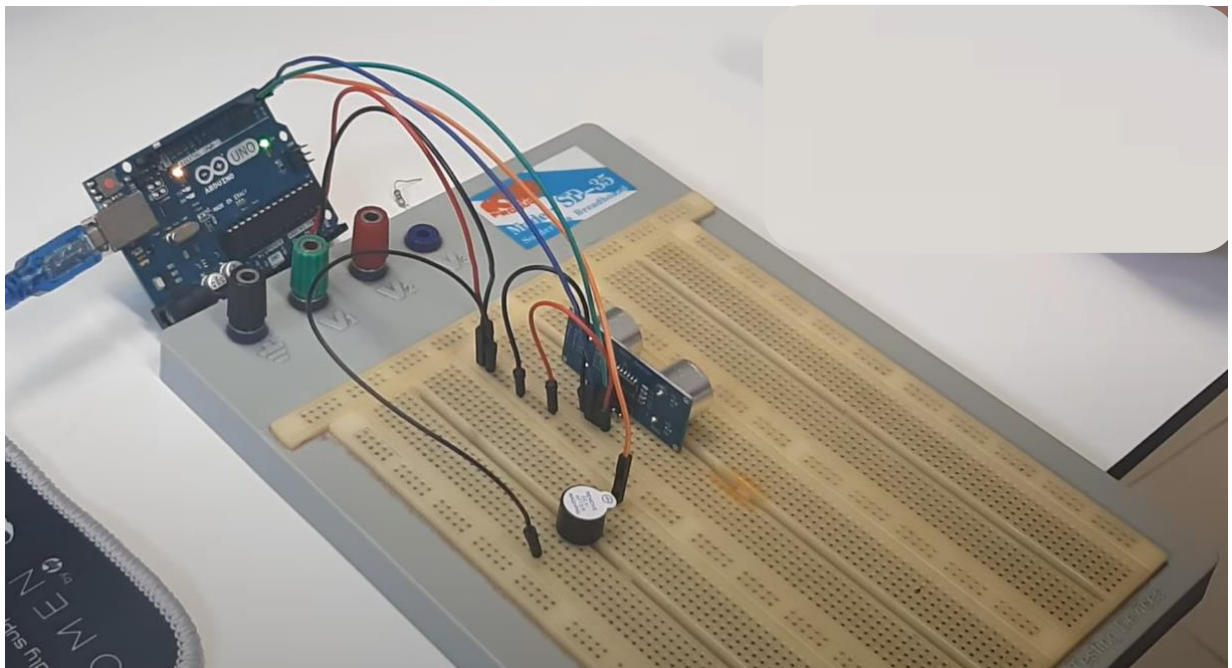


Figura 2 - 1\_HC\_SR04 breadboard

Antes de começarmos a fazer o código tivemos que fazer uma breve pesquisa de como funciona a função `pulsIn()` e também de perceber o funcionamento das entradas do sensor.

```

1 void digital_delay(int d) {
2   noTone(2);
3   delay(d);
4   tone(2, 500);
5 }
6
7 void setup() {
8   pinMode(4, OUTPUT);
9   pinMode(3, INPUT);
10  pinMode(2, OUTPUT);
11  Serial.begin(9600);
12 }
13
14 void loop() {
15   digitalWrite(4, LOW);
16   delayMicroseconds(2);
17   digitalWrite(4, HIGH);
18   delayMicroseconds(10);
19   digitalWrite(4, LOW);
20
21   float duration = pulseIn(3, HIGH);
22   float distance = duration * 0.034 / 2;
23
24   if (distance < 10) {
25     digitalWrite(2, HIGH);
26   } else if (distance < 25) {
27     digital_delay(1000);
28   } else if (distance < 50) {
29     digital_delay(2000);
30   } else noTone(2);
31
32   Serial.print("Distance: ");
33   Serial.println(distance);
34   delay(100);
35 }

```

Figura 3 - 1\_HC\_SR04 código

O código ao lado representa o programa utilizado para configurar o arduino. Começamos por criar a função **digital\_delay()** para produzir o som no buzzer e dentro dessa função usamos o `tone` que gera uma onda quadrada da frequência especificada em um pino, a onda continua até uma chamada para **noTone()**. Se um `tone` já estiver a ser usado num pino, a nova chamada `tone()` não será executada. Se estiver no mesmo pino, poderá ter uma frequência diferente caso seja o pretendido. Em seguida, na função **setup()** declaramos o pino 4 que está ligado ao Trig do sensor como output e o pino 3 que está ligado ao Echo do sensor como input.

Na função **loop()**, definimos o pin 4 como LOW por 2 microssegundos apenas para garantir que o pino esteja LOW ao ser executado, em seguida configuramos como HIGH por 10 microssegundos, que envia uma onda sonora de 8 ciclos do transmissor, esta onda quando atinge o objeto faz ricochete e atinge o recetor (conectado ao echo pin). Quando as ondas sonoras atingem o recetor, ele aumenta o pino echo pelo tempo que as ondas estiverem a propagar-se, para conseguirmos isso utilizamos a função **pulsIn()** para começar a cronometrar quando o pino echo for high e armazenamos o tempo na variável `duration`. Para sabermos a que distância se encontra o objeto irá ser utilizado uma fórmula

(tempo \* velocidade = distância)

A velocidade que vamos utilizar será a do som que é aproximadamente 340 m/s, mas como a função **pulsIn()** retorna o valor do tempo em microssegundos precisamos do mesmo tipo de valor para podermos calcular a distância (microssegundos), ou seja, multiplicamos a duração por 0,0343 e dividimos tudo por 2, isto é feito assim pois as ondas sonoras viajam para o objeto e voltam. Depois de obtermos esta informação toda é só impor condições e para não estarmos a repetir o mesmo processo para acionar o buzzer criamos a função **digital \_lay()**.

## 3.2. Exercício 2

Neste exercício o objetivo é acrescentar 2 leds ao exercício anterior, onde um deles deve estar ativo quando a distância  $>50$  cm e outro deve estar ativo somente quando a distância  $<50$  cm.

Começamos por acrescentar os leds do circuito do **Tinkercad** (Figura 4) de seguida para o código no **Arduíno** (Figura 6) e depois para a construção do mesmo (Figura 5).

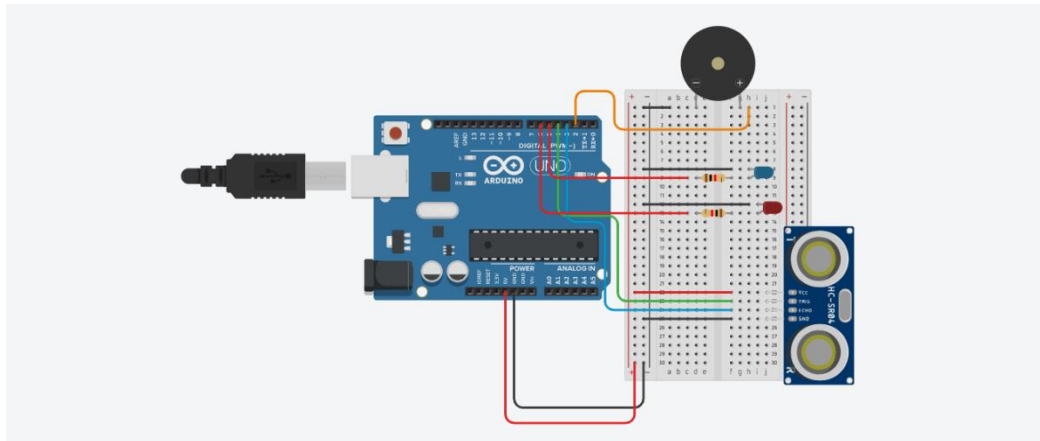


Figura 4 - 2\_HC\_SR04 Tinkercad

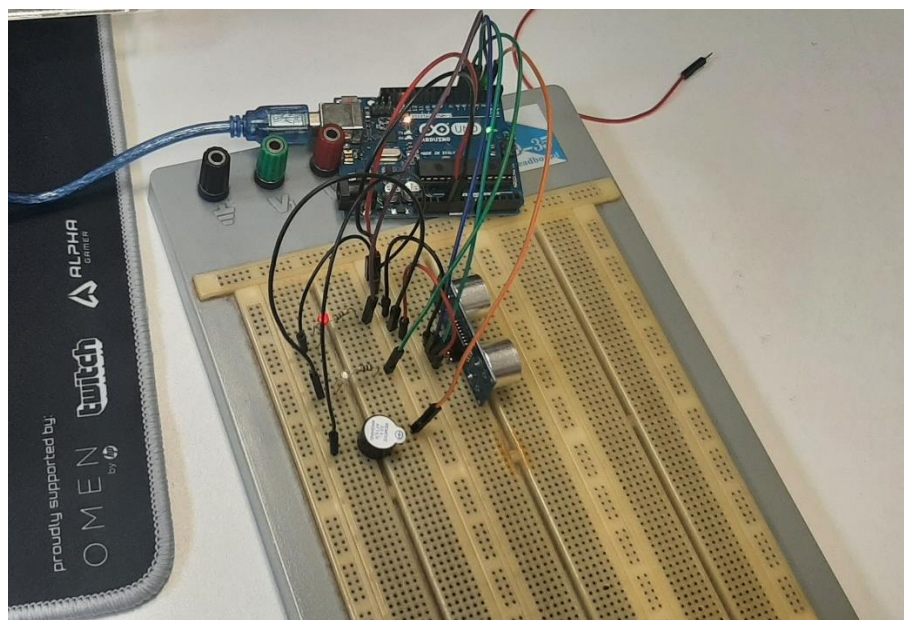


Figura 5 - 2\_HC\_SR04 breadboard

```
1 void digital_delay(int i) {
2   digitalWrite(6, LOW);
3   digitalWrite(5, HIGH);
4   noTone(2);
5   delay(i);
6   tone(2, 500);
7 }
8
9 void setup() {
10  pinMode(6, OUTPUT); // more than 50
11  pinMode(5, OUTPUT); // less than 50
12  pinMode(4, OUTPUT);
13  pinMode(3, INPUT);
14  pinMode(2, OUTPUT);
15  digitalWrite(6, HIGH);
16  Serial.begin(9600);
17 }
18
19 void loop() {
20   digitalWrite(4, LOW);
21   delayMicroseconds(2);
22   digitalWrite(4, HIGH);
23   delayMicroseconds(10);
24   digitalWrite(4, LOW);
25
26   float duration = pulseIn(3, HIGH);
27   float distance = duration * 0.034 / 2;
28
29   if (distance < 10) {
30     digitalWrite(6, LOW);
31     digitalWrite(5, HIGH);
32     tone(2, 500);
33   } else if (distance < 25) {
34     digital_delay(1000);
35   } else if (distance < 50) {
36     digital_delay(2000);
37   } else {
38     digitalWrite(6, HIGH);
39     digitalWrite(5, LOW);
40     noTone(2);
41   }
42
43   Serial.print("Distance: ");
44   Serial.println(distance);
45   delay(100);
46 }
```

Reutilizando o código usado no exercício anterior e as formulas ficamos com o código ao lado, ou seja, a única alteração feita foi na função **setup()** definimos o pino 5 e 6 como output. Na função **digital \_lay()** definimos que o led que estivesse ligado ao pin 5 acendia quando a distância era menor que 50. Tivemos q adicionar mais uma condição para quando for maior que 50, o led ligado ao pino 6 acender e o do pino 5 desligar.

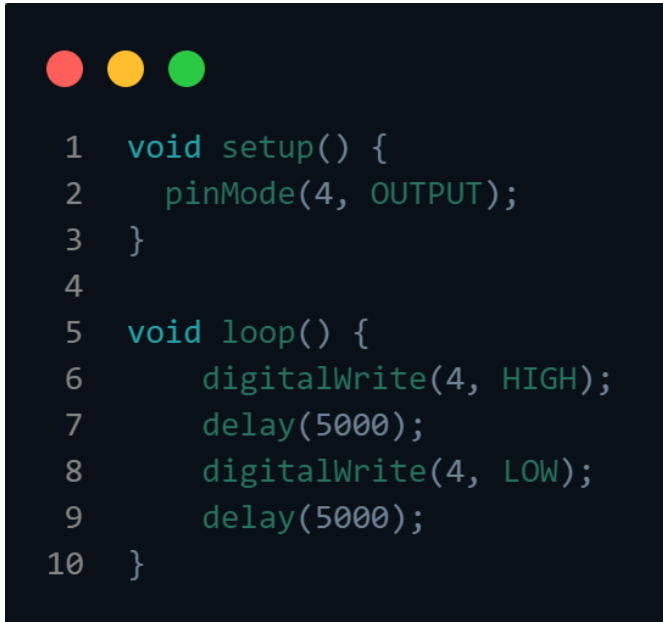
Figura 6 - 2\_HC\_SR04 código



### 3.3. Exercício 3

Neste exercício o objetivo é utilizando o **relay** conectado ao NC, ligar e desligar um led num intervalo de 5 segundos.

Começamos por fazer o código no **Arduíno** (Figura 7) para em seguida procedemos para a sua construção na **breadboard** (Figura 8).



```
1 void setup() {  
2   pinMode(4, OUTPUT);  
3 }  
4  
5 void loop() {  
6   digitalWrite(4, HIGH);  
7   delay(5000);  
8   digitalWrite(4, LOW);  
9   delay(5000);  
10 }
```

O código ao lado representa o programa utilizado para configurar o **Arduíno** onde na função **setup()** definimos o pin 4 como **output**. Na função **loop()** é ativado o pino 4 (que esta liagdo ao relay) permanecendo ligado durante 5 segundos e depois desliga.

Figura 7 - Relay código

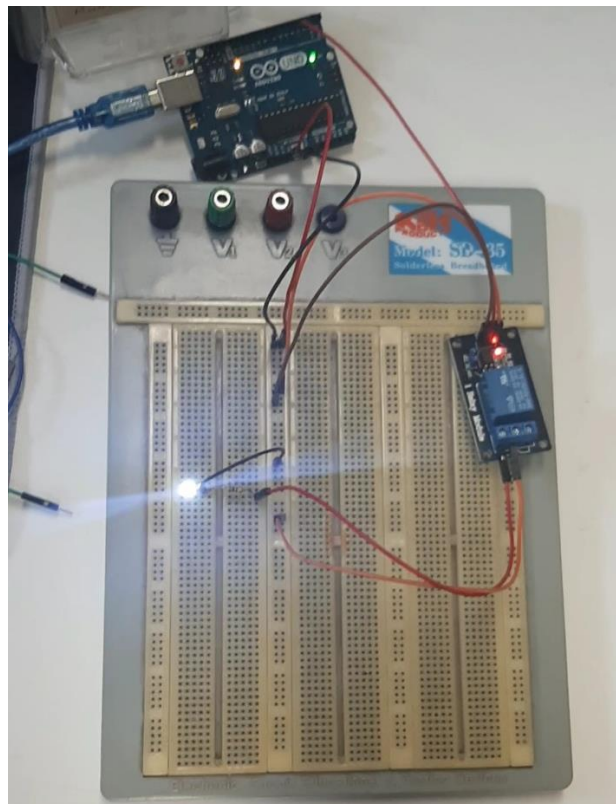


Figura 8 - relay breadboard



### 3.4. Exercício 4

O objetivo neste exercício é igual ao anterior, mas desta vez com o **Relay** conecta ao NO. Neste exercício utilizamos o código do exercício anterior (figura 7) e a única alteração feita foi no **Relay** que foi passar de **NC** para **NO**.

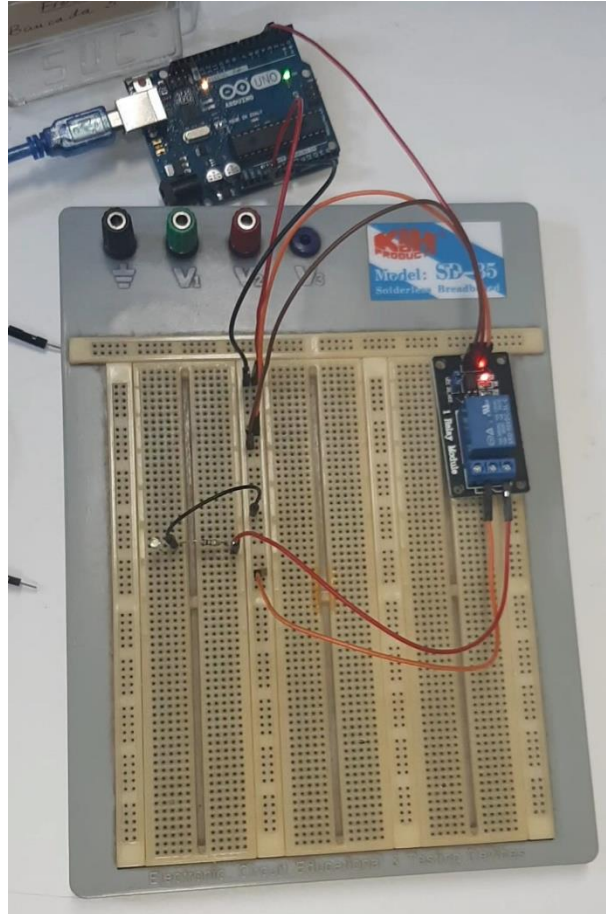


Figura 9 - 4\_relay breadboard

## 4. Discussão

Onde houve maior dificuldade foi no primeiro exercício porque não estávamos a conseguir inicialmente entender a função `PulsIn()`, tínhamos feito várias tentativas por conta própria com o `PulsIn`, mas nenhuma tinha resultado, até que encontramos um programa em que o objetivo era parecido ao nosso mas o buzzer só era acionado se a distância fosse menor ou igual que 5, a partir desse programa conseguimos obter um ponto de partida. Nos exercícios 4 e 5 reparámos que ao usar o mesmo código, mas com conexões diferentes do Relay obtivemos o mesmo, a única diferença que vimos foi numa luz vermelha no Relay que, estando conectado ao NC quando o led estava aceso, acendia uma luz vermelha e desligavam se os dois ao mesmo tempo, quando estava conectado ao NO acontecia o contrário ele só acendia a luz quando o led estava apagado

## 5. Conclusão

Assim sendo podemos concluir que cumprimos todos exercícios e que apesar de termos tido alguma dificuldade no primeiro exercício conseguimos ultrapassá-lo.

## 6. Referências

<https://create.arduino.cc/editor/mertarduinotech/b7022e05-f709-4003-b2a7-5c487ee25007/preview>

[https://projecthub.arduino.cc/Isaac100/7cabe1ec-70a7-4bf8-a239-325b49b53cd4?ref=platform&ref\\_id=424\\_updated\\_732\\_protip&offset=28](https://projecthub.arduino.cc/Isaac100/7cabe1ec-70a7-4bf8-a239-325b49b53cd4?ref=platform&ref_id=424_updated_732_protip&offset=28)

<https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/>

<https://www.arduino.cc/reference/en/language/functions/advanced-io/notone/>

<https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/>

<https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/>

<https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/>

Videos:

<https://www.youtube.com/playlist?list=PLweUCI9fZUobmv2fS-CDtDKt6fgLHGtN0>