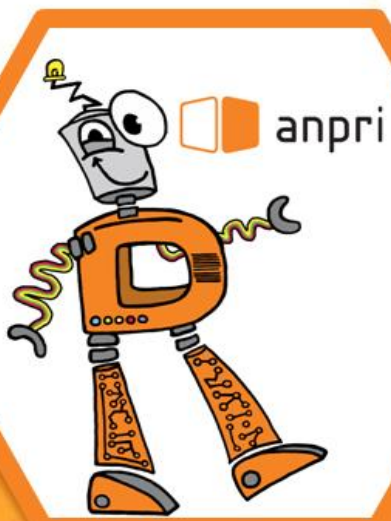
# Ação: Programação com Arduinos

## 5ª sessão - Sensores e atuadores simples

Maio, 2021



# Atividade 11 – Sensor de luz LDR

**Descrição**: Nesta atividade pretende-se simular uma luz de presença. O led deverá acender quando o sensor de luz LDR registar valores abaixo de um determinado limiar.

As foto-resistências LDR (Light-Dependent Resistor) representam um tipo de resistências cujo valor resistivo varia em função da luz, o que as torna especialmente úteis em sistemas automáticos.

Devido às suas características poderão funcionar como sensores de luminosidade. A foto-resistência LDR é composta essencialmente por Sulfeto de cádmio (CdS). O cadmio reage à luz, deixando que os seus eletrões se movam livremente, permitindo assim a passagem da corrente elétrica.

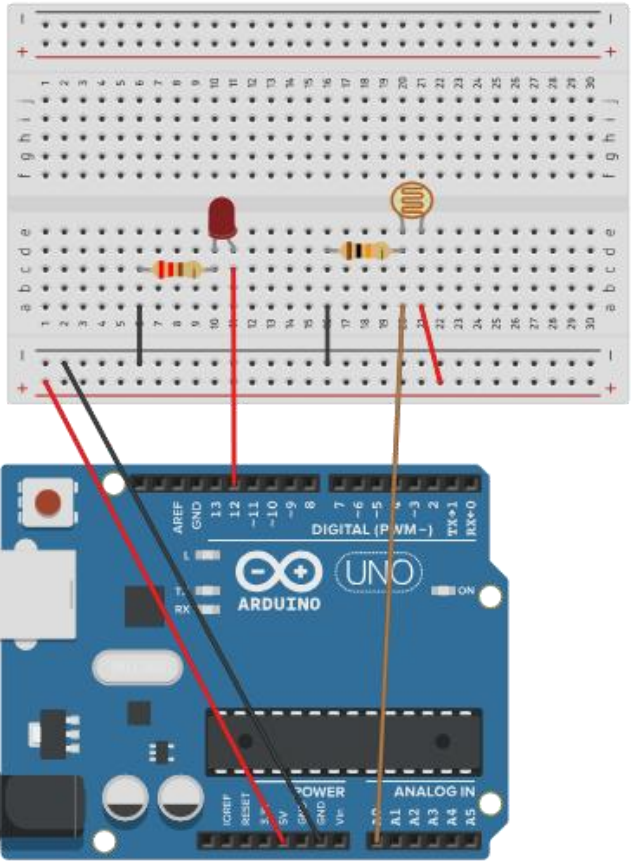
O LDR, assim como uma resistência comum, não possui polarização e a sua resistência é medida em ohms.

Como o LDR é um componente analógico, o intervalo de valores obtidos na porta analógica varia entre 0 (0V) e 1023 (5V). Quanto maior for a luminosidade, menor será a resistência elétrica e, portanto, maior será o valor obtido. Quanto menor for a luminosidade, maior será a resistência elétrica e, portanto, menor será o valor obtido.

## Material necessário:

- **1 Led** 
- **1 LDR** 
- **2 Resistências**  **220 Ω**, **10 kΩ**
- **Fios** 

## Esquema do circuito:



## Programação:

1º - Vamos começar por definir as variáveis constantes para os pinos correspondentes ao led e ao LDR:

```
#define ledPin 12
#define ldrPin A0
```

Para além disso, será necessário definir uma variável para guardar os valores obtidos pelo LDR:

```
int ldr;
```

2º - Na função **"setup"** será necessário configurar o pino digital do led como pino de saída (OUTPUT), o pino analógico do LDR como pino de entrada (INPUT) e definir a taxa de transmissão de dados na comunicação série:

```
pinMode(ledPin, OUTPUT);
pinMode(ldrPin, INPUT);
Serial.begin(9600);
```

3º - Na função **"loop"**, será necessário obter os valores do sensor LDR e guardá-los numa variável. Para se perceber quais são os valores obtidos pelo LDR é conveniente utilizar o monitor série.

```
ldr = analogRead(ldrPin);
Serial.print("Ldr: ");
Serial.println(ldr);
```

Após a leitura dos valores é necessário definir um limiar a partir do qual o led acenderá. Neste exemplo, foi utilizado o valor 500, no entanto este valor poderá ser ajustado de acordo com a vossa preferência. Se o valor for menor que 500 então significa que a luminosidade é mais reduzida e o led acenderá, caso contrário o led permanecerá apagado.

```
if (ldr < 500) {
  digitalWrite(ledPin, HIGH);
} else {
  digitalWrite(ledPin, LOW);
}
```

**Código Final:**

```
#define ledPin 12
#define ldrPin A0

int ldr;
//int led;

void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT);
  pinMode(ldrPin, INPUT);
  Serial.begin(9600);
}


void loop() {
  // put your main code here, to run repeatedly:
  ldr = analogRead(ldrPin);
  Serial.print("Ldr: ");
  Serial.println(ldr);

  if (ldr < 500) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

# Atividade 12 – Sensor de Temperatura

**Descrição**: Nesta atividade pretende-se construir um circuito que permita medir valores de temperatura ambiente de uma sala, utilizando um sensor de temperatura LM35 e consoante os valores de temperatura obtidos, acender diferentes leds.

## LM35

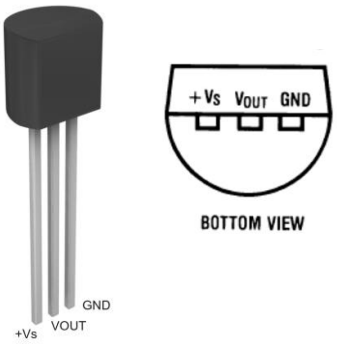
O sensor LM35 é um sensor de precisão que apresenta uma saída de tensão linear proporcional à sua temperatura. Para cada grau celsius de temperatura obtém-se uma saída de 10mV. É um circuito integrado, medidor de temperatura, com aparência de um transístor de 3 terminais.

Este sensor possui uma alta precisão e funciona com tensões de intervalo de 4V a 30VDC. Não necessita de calibração externa para fornecer, com exatidão, valores de temperatura com variações entre ¼°C e ¾°C, dentro do intervalo de temperaturas –55°C a 150°C.

O sinal de saída do sensor de temperatura LM35 é analógico e cada 10mV de tensão representa 1°C.

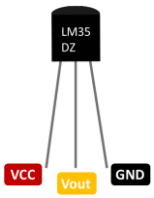
Especificações e características:

- Tensão de operação: 4 a 30VDC
- Intervalo de valores: -55° a 150°C
- Precisão: ±0,5°C
- Sensibilidade: 10mV/°C
- Conexão de saída: analógica

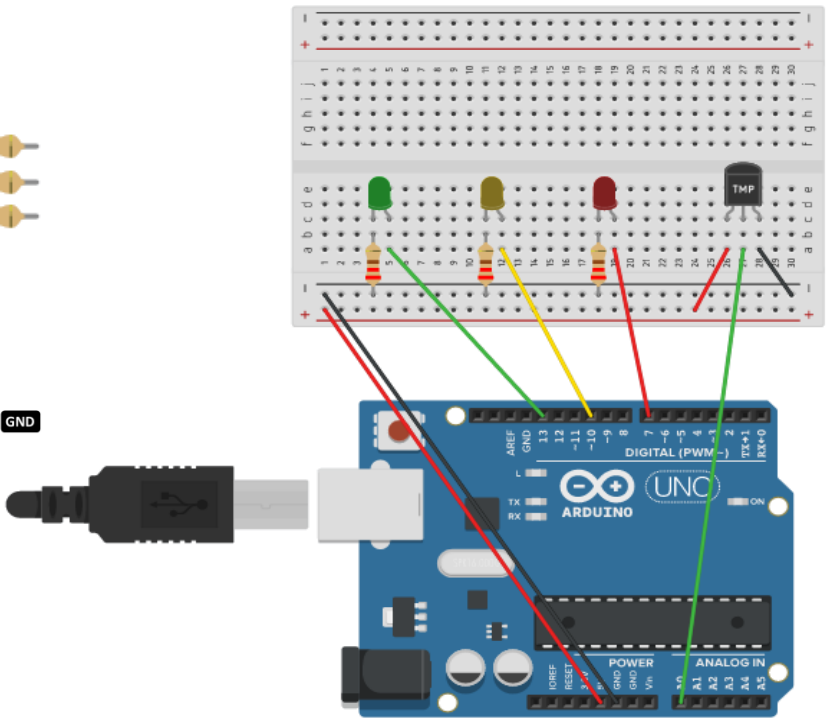



## Material necessário:

- **3 Leds**
- **3 Resistências 220Ω**
- **1 Sensor LM35**
- **Fios**




## Esquema do circuito:




## Programação:

1º - Será necessário definir quatro variáveis constantes correspondentes aos pinos onde estão ligados os leds e o sensor de temperatura:

```
#define ledPinR 7 //Porta digital led vermelho
#define ledPinY 10 //Porta digital led amarelo
#define ledPinG 13 //Porta digital led verde
#define tmpPin A0 //Porta analógica - Sensor temperatura
```

Para além disso, será necessário definir uma variável para guardar os valores obtidos pelo sensor de temperatura:

```
int sensor;
```

2º - Na função **"setup"** será necessário configurar os pinos digitais dos leds como pinos de saída (OUTPUT), o pino analógico do sensor LM35 como pino de entrada (INPUT) e definir a taxa de transmissão de dados na comunicação série:

```
pinMode(ledPinR, OUTPUT);
pinMode(ledPinY, OUTPUT);
pinMode(ledPinG, OUTPUT);
pinMode(tmpPin, INPUT);
Serial.begin(9600);
```

3º - Na função **"loop"**, é necessário obter o valor relativo ao sensor de temperatura LM35. Este deve ser guardado numa variável. Para se perceber quais são os valores obtidos pelo sensor é conveniente utilizar o monitor série.

```
sensor = analogRead(tmpPin);
Serial.print("Sensor: ");
Serial.print(sensor);
```

4º - Definir variáveis para guardar os valores de tensão e temperatura calculados através dos valores obtidos pelo sensor de temperatura

```
float tensao, temp;
```

5º - Efetuar a conversão dos valores obtidos em valores de tensão e temperatura:

```
tensao = (sensor / 1024.0) * 5;
Serial.print(" Tensão: ");
Serial.print(tensao);
```

```
temp = tensao * 100;
Serial.print(" Temperatura: ");
Serial.println(temp);
```

LM35

```
temp = (tensao - 0.5) * 100;
Serial.print(" Temperatura: ");
Serial.println(temp);
```

TMP36

6º - Determinar qual o valor de temperatura ambiente da sala e criar uma constante com esse valor. Utilizar o monitor série para determinar esse valor.

```
const float tempAmb = 22.00;
```

7º - Acender os leds consoante os valores de temperatura obtidos:

- Se os valores de temperatura forem menores ou iguais ao valor da temperatura ambiente, os leds permanecerão desligados.
- Se os valores de temperatura forem maiores que o valor da temperatura ambiente e menores ou iguais ao valor da temperatura ambiente + 2, o led verde acenderá.
- Se os valores de temperatura forem maiores que o valor da temperatura ambiente + 2 e menores ou iguais ao valor da temperatura ambiente + 3, o led amarelo acenderá.
- Se os valores de temperatura forem maiores que o valor da temperatura ambiente + 3, o led vermelho acenderá.

## Código Final:

```
if (temp <= tempAmb) {
  digitalWrite(ledPinR, LOW);
  digitalWrite(ledPinY, LOW);
  digitalWrite(ledPinG, LOW);
} else if (temp > tempAmb && temp <= tempAmb + 2) {
  digitalWrite(ledPinR, LOW);
  digitalWrite(ledPinY, LOW);
  digitalWrite(ledPinG, HIGH);
} else if (temp > tempAmb + 2 && temp <= tempAmb + 3) {
  digitalWrite(ledPinR, LOW);
  digitalWrite(ledPinY, HIGH);
  digitalWrite(ledPinG, LOW);
} else if (temp > tempAmb + 3) {
  digitalWrite(ledPinR, HIGH);
  digitalWrite(ledPinY, LOW);
  digitalWrite(ledPinG, LOW);
}
```

# Atividade 13 – Sensor Ultrassónico

**Descrição**: Nesta atividade pretende-se construir um circuito que permita determinar a distância de um objeto, utilizando um sensor ultrassónico.

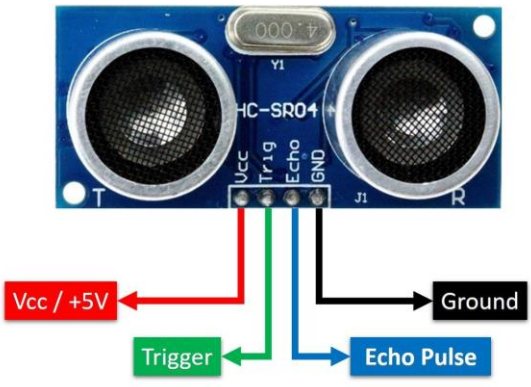
Os sensores ultrassónicos possuem um emissor e um recetor de ondas sonoras. O seu funcionamento baseia-se na emissão de uma onda sonora de alta frequência (40khz), impercetível ao ouvido humano. Ao ser emitida a onda sonora, é acionada uma espécie de relógio de alta precisão, que mede o tempo entre a colisão da onda sonora com um obstáculo e a reflecção de volta ao recetor do sensor. Através do valor da velocidade do som e tendo em conta o tempo que o sinal demorou a colidir com o obstáculo e retornar, é possível, determinar qual a distância percorrida entre sensor e obstáculo.


**HC-SR04**

O módulo HC-SR04 é um sensor de distância composto por um emissor e um recetor, com capacidade de medir distâncias de 2cm até 4m, com uma precisão de aproximadamente 3mm. Este sensor emite sinais ultrassónicos que refletem no objeto a ser atingido e retornam ao sensor, indicando a distância do alvo.

A velocidade do sinal ultrassónico emitida pelo Sensor HC-SR04 corresponde à velocidade do som, que é de aproximadamente 340 m/s. Assim, se o sensor estiver a uma distância **x** do objeto, o sinal percorrerá uma distância equivalente a **2x**, ou seja, a onda enviada pelo sensor é refletida no obstáculo, logo percorre 2 vezes a distância em relação ao objeto.

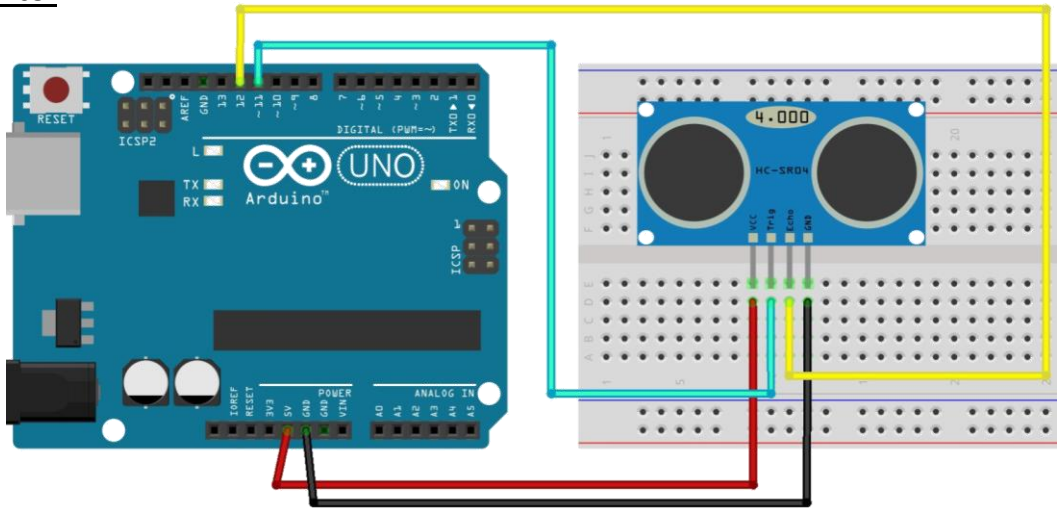
É importante referir que caso o obstáculo seja muito pequeno ou não esteja exatamente à frente do sensor, pode não ser detetado.



## Material necessário:

- **1 Sensor Ultrassónico HC-SR04** 
- **Fios** 

## Esquema do circuito:




## Programação:

1º - Adicionar a biblioteca **Ultrasonic.zip** ao software Arduino.

2º - Incluir a biblioteca **Ultrasonic.h** na programação:

```
#include <Ultrasonic.h>
```

3º - Definir os pinos correspondentes ao **Trigger** e ao **Echo** do sensor:

```
#define triggerPin  11
#define echoPin     12
```

4º - Criar um objeto do tipo Ultrasonic:

```
Ultrasonic ultrasonic(triggerPin, echoPin);
```

5º - Declarar as variáveis associadas à distância e ao tempo:

```
//distância do sensor em relação ao objeto
float distancia;
// tempo que o sinal demora a chegar desde que foi emitido
long tempo;
```

6º - Utilizar a função **timing()** para obter o valor relativo ao tempo que o sinal demora a chegar desde que foi emitido:

```
tempo = ultrasonic.timing();
```

7º - Calcular a distância do sensor em relação ao objeto, em centímetros, utilizando a função **convert**:

```
distancia = ultrasonic.convert(tempo, Ultrasonic::CM);
```

## Código Final:

```
#include <Ultrasonic.h>

#define triggerPin  11
#define echoPin     12

Ultrasonic ultrasonic(triggerPin, echoPin);

//distância do sensor em relação ao objeto
float distancia;
// tempo que o sinal demora a chegar desde que foi emitido
long tempo;

void setup()
{
  Serial.begin(9600);
}

void loop()
{

  tempo = ultrasonic.timing();

  distancia = ultrasonic.convert(tempo, Ultrasonic::CM);

  Serial.print("Distância: ");
  Serial.println(distancia);
  delay(100);
}
```