# ADC - Analógico

Até um tempo atrás, o analógico ainda era muito falado. Por exemplo, antes dos relógios digitais e smart watchs, as pessoas costumavam utilizar o relógio de ponteiro, também conhecido como relógio analógico.

Afinal, o que é o sinal analógico?

Anteriormente, aprendemos que o sinal digital é aquele que assume apenas dois valores, 0 ou 1, 0V ou 5v, ligado ou desligado e assim por diante. Por sua vez, o sinal analógico pode assumir qualquer valor dentro de um intervalo. Exemplificando, podemos utilizar a temperatura, que pode variar -15°C até 50°C, por se tratar de um valor analógico pode assumir qualquer valor dentro desse intervalo, 3.256°C ou 45,3 °C.

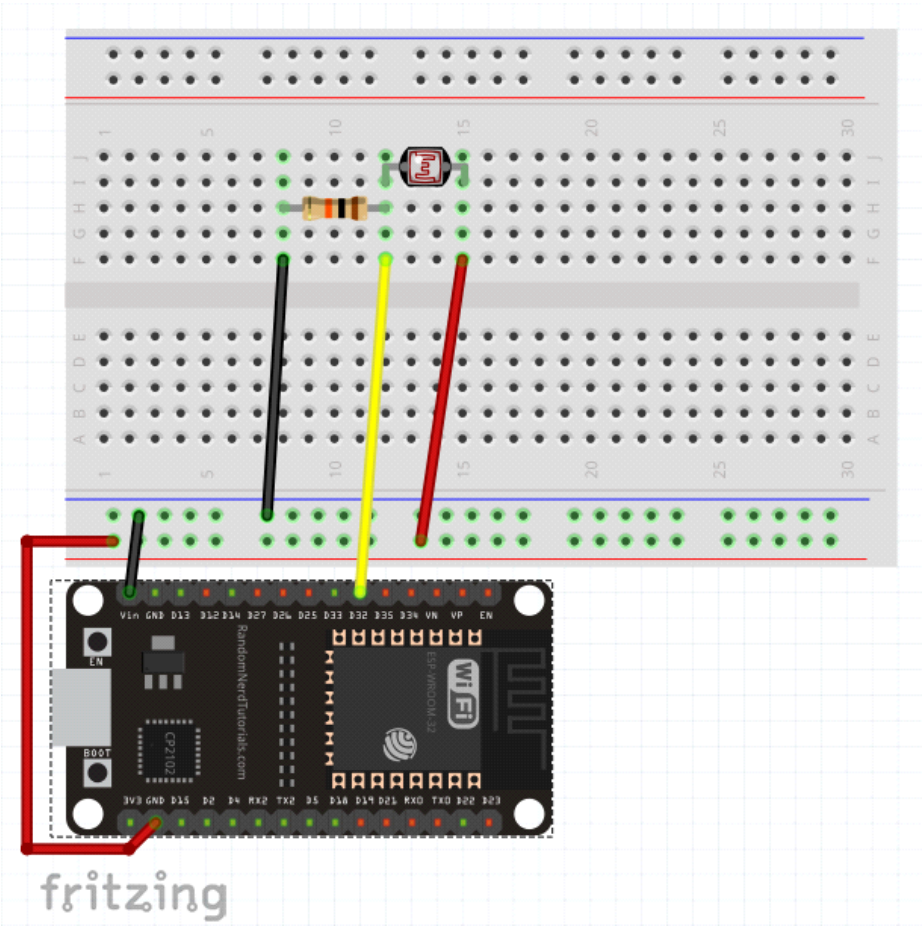
Para termos acesso a esse recurso utilizando a linguagem MicroPython, é necessário entendermos o conceito de ADC – Conversão Analógica Digital (ou, *Analog Digital Converter*, em inglês). De maneira simplificada, o ADC é responsável por "transformar" a corrente elétrica do sinal analógico em um sinal digital. A onda gerada tenha a leitura realizada em vários pontos por segundo da onda elétrica gerada, após isso, é gerada uma versão digital desse sinal elétrico, estabelecido através da aproximação de valores da medição.

Para esse projeto, inicialmente você vai precisar de...

- ESP32;
- Fotoresistor (LDR) 10kohm;
- Termistor (LDR) 10kohm;
- 1 resistor 10KOHM;

- Jumper

**Com os materiais em mãos, realize a montagem a seguir.**




## Código

No programa abaixo, vamos realizar a leitura da intensidade luminosa do ambiente. Repare, os valores vão variar num intervalo de 0 a 4095.

```
#Aula 3 - Sinal Analógico

from machine import Pin, ADC
from time import sleep

ldr = ADC(Pin(32))
ldr.atten(ADC.ATTN_11DB)

while True:
    ldr_leitura = ldr.read() #associar o valor recebido a variável a ldr_leitura
    print(ldr_leitura)       #imprimir valor monitor
    sleep(2)
```

Escreva o código acima e execute-o.

## Explicação

Como aprendemos antes, vamos importar a função **Pin** do módulo **machine**. Porém, dessa vez vamos necessitar de uma função chamada **ADC**:

```
3  from machine import Pin, ADC
```

Também, precisaremos da função time para o delay:

```
4  from time import sleep
```

Agora, vamos configurar o pino como analógico. Chamaremos a função ADC, dentro dela vamos declarar qual pino está sendo utilizado e, associar esse pino a uma variável, no caso, ldr:

```
6  ldr = ADC(Pin(32))
```

Por conseguinte, vamos "regular" o pino. Usaremos a função ldr.atten, que declara qual o intervalo de tensão, no nosso caso, de 0 a 3.3V:

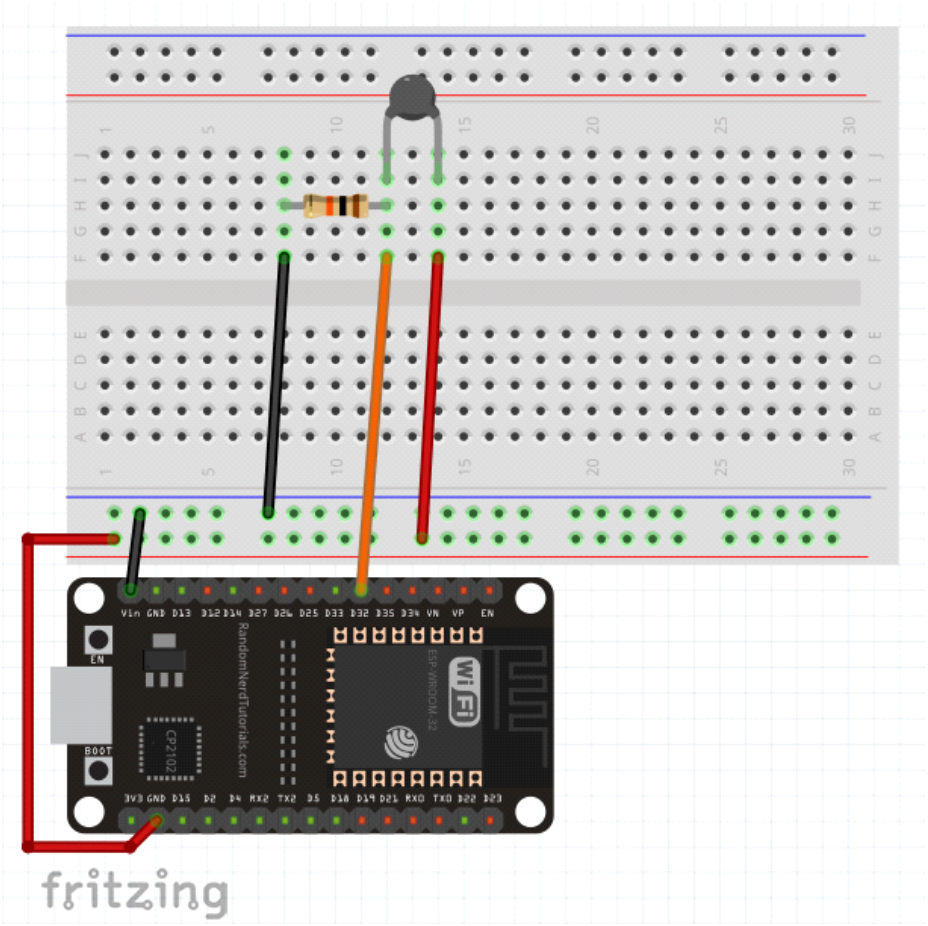
```
7  ldr.atten(ADC.ATTN_11DB)
```

Por fim, vamos criar o nosso laço de repetição infinita, while True. Nele, vamos realizar a leitura da intensidade e armazená-la na variável ldr_leitura. Para essa leitura, chamaremos a função ldr.**read()**. Também, vamos imprimir essa leitura a cada dois segundos:

```
9  while True:
10     ldr_leitura = ldr.read() #associar o valor recebido a variável a ldr_leitura
11     print(ldr_leitura)       #imprimir valor monitor
12     sleep(2)
```

## Código 2

Agora, vamos aprender como usar um termistor, que varia sua tensão de acordo com temperatura. A montagem é mesma, assim como a configuração do pino. A única diferença é que usaremos uma função que converterá a resistência em graus Celsius:




**Escreva e rode o código a seguir:**

```
import math
from machine import Pin, ADC
from time import sleep

thermistor = ADC(Pin(32))
thermistor.atten(ADC.ATTN_11DB) #Atenuação -->

def temperature(r):
    fator = 4096
    TempK = math.log(10000.0*(fator / r - 1))
    TempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * TempK * TempK))* TempK)
    TempC = TempK - 273.15  #Convert Kelvin to Celsius
    return TempC

while True:
    adc = thermistor.read()
    temperatura = temperature(adc)
    print(temperatura,'°C',adc)
    sleep(1)
```

## Explicação 2

Além das bibliotecas e funções já usadas, aqui necessitaremos da biblioteca padrão do python, math:

```
import math
from machine import Pin, ADC
from time import sleep
```

Agora, vamos configurar o pino, da mesma maneira que fizemos com o ldr:

```
thermistor = ADC(Pin(32))
thermistor.atten(ADC.ATTN_11DB) #Atenuação -->
```

Finalmente, vamos a nossa função. Talvez, você já tenha a visto na plataforma Arduino, aqui vamos adaptá-la para nosso uso. Primeiro, vamos criar a função, que receberá um parâmetro, ou seja, a resistência que será convertida. Depois, vamos definir o fator, a quantia máxima de valores que serão enviados. Na próxima linhas, vamos chamar a função **math.log**, retorna o logaritmo natural, nessas linhas a resistência será convertida em Kelvin e depois em Celsius. E. por fim retornará o valor em Celsius:

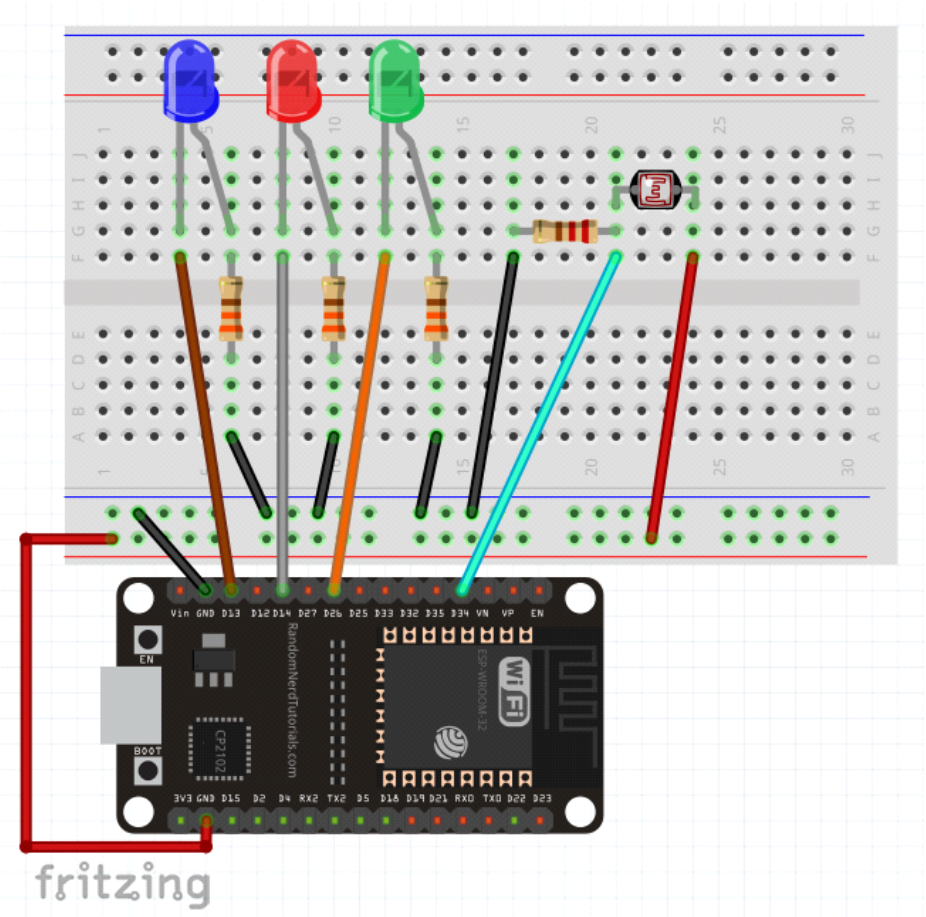
```
def temperature(r):
    fator = 4096
    TempK = math.log(10000.0*(fator / r - 1))
    TempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * TempK * TempK))* TempK)
    TempC = TempK - 273.15  #Convert Kelvin to Celsius
    return TempC
```

Finalmente, em nosso loop, realizaremos a leitura do Termistor e depois chamaremos a função temperature(), para converter esse valor. Ademais, imprimiremos esse valor a cada segundo:

```
while True:
    adc = thermistor.read()
    temperatura = temperature(adc)
    print(temperatura,'ºC',adc)
    sleep(1)
```

## Exercícios

- Aqui vamos fazer uma espécie de termômetro para luminosidade, quanto mais escuro, mais leds ligados. Realiza a montagem abaixo:



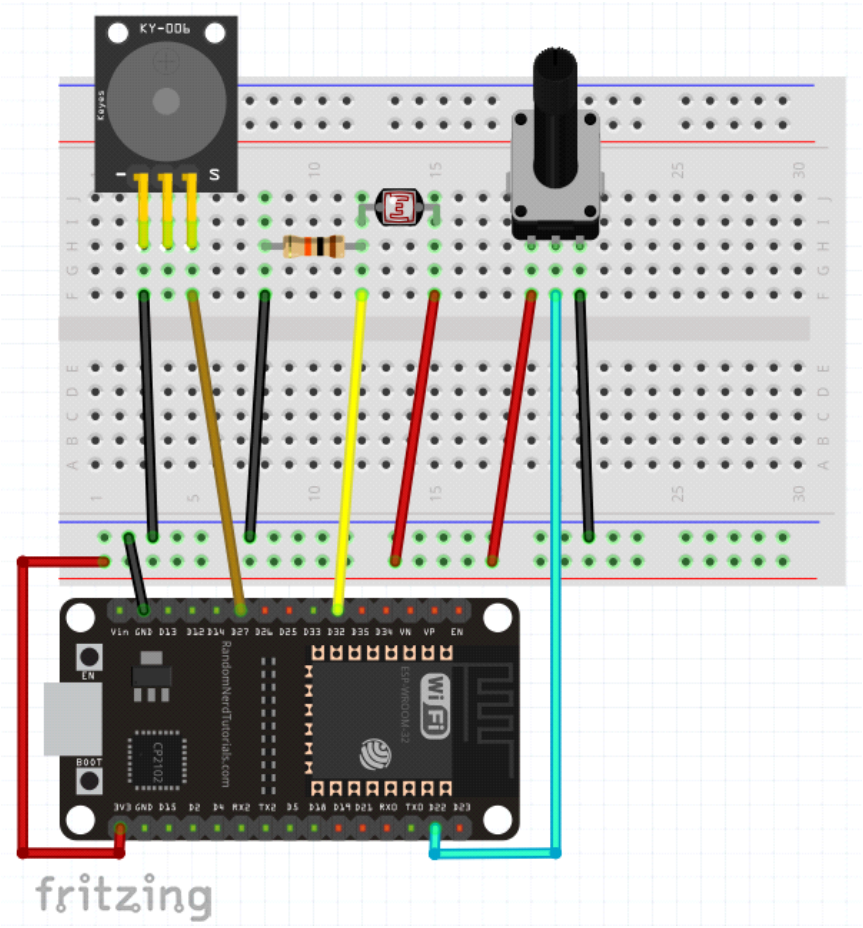

**Siga o seguinte algoritmo:**

- Se a leitura for **maior ou igual a 0 e menor ou igual ao valor máximo (4095) dividido por 3**, apenas um led ligado;
- Caso seja, **maior que 4095/3 e menor ou igual a 4095**, dois leds ligados;
- E, se for **maior que 4095/2 e menor igual ao valor máximo**, 3 leds ligados.

- Utilizando um potenciômetro, acione um buzzer quando o valor ultrapassar metade do valor total. Realize a montagem abaixo e, utilize o código abaixo para configurar o potenciômetro:

```
5  pot = ADC(Pin(32))
6  pot.atten(ADC.ATTN_11DB)
7  pot.width(ADC.WIDTH_12BIT) #seta 12 bits(faixa de 0 - 4095)
```

Na terceira linha, estamos setando a quantia de bits.


KY-006
S
Vin GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
EN
BOOT
CP2102
RandomNerdTutorials.com
ESP-WROOM-32
Wi Fi
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
fritzing