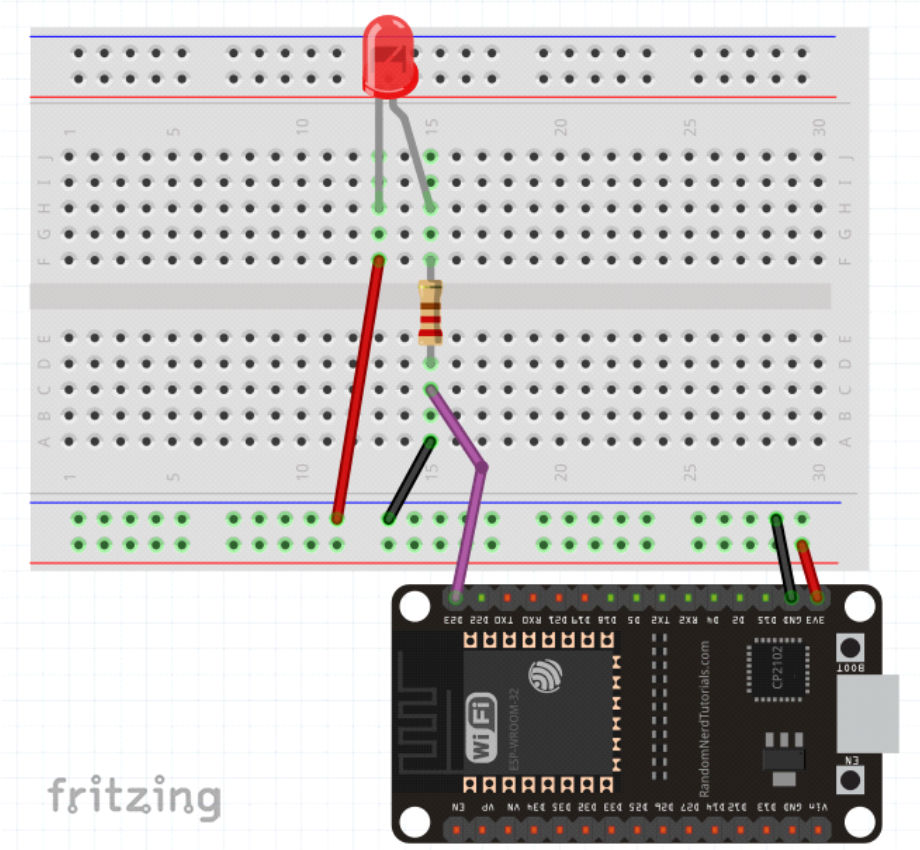
PWM

PWM é abreviação para, **Pulse Width Modulation** – que significa – Modulação por Largura de Pulso. Essa é uma técnica onde, uma saída digital é usada para simular uma saída analógica. Isso acontece por meio de mudanças constantes de sinal, que são imperceptíveis e, dessa maneira o componente conectado a esse pino apenas tira uma média de quanto tempo está em estado alto e baixo. Quanto mais tempo ligado, maior o valor e, consequentemente, quanto menor tempo ligado, menor o valor.

Necessitaremos, vamos precisar de...

- ESP32;
- LED (qualquer cor);
- 1 LED RGB;
- Potenciômetro;
- Resistores 220 ou 330 Ω.

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



Código

Com a montagem em mãos, vamos começar a codar, No código abaixo, usaremos duas funções internas do Python, for e range. Nosso objetivo, é variar a intensidade luminosa de um led.

```
#Aula 4 - PWM

from machine import Pin, PWM
from time import sleep

led = PWM(Pin(23))

while True:
    #for --> estrutura de repetição
    #range --> intervalo de valores com icremento

    for i in range(0, 1023, 5):
        led.duty(i)  #enviar o valor para o LED
        sleep(0.05)

    for i in range(1023,0, -5):
        led.duty(i)  #enviar o valor para o LED
        sleep(0.05)
```

Copie o código acima na IDE Thonny e execute-o.

Explicação

Para iniciar, vamos importar as funções **Pin** e **PWM** do módulo **machine**:

```
3 from machine import Pin, PWM
```

Também, precisaremos a função sleep para intervalos de temporização:

```
4 from time import sleep
```

Partiremos para a configuração do pino PWM. Repare que é o mesmo que fizemos com a função ADC, apenas chamaremos a função PWM e declararemos qual o pino que estamos utilizamos e, associar a uma variável:

```
6 led = PWM(Pin(23))
```

Agora, iremos para o nosso while True. Aqui, usaremos as funções for, uma estrutura de repetição e a função range, que gera um intervalo de valores. Em range, informaremos os parâmetros: valor inicial, valor final e qual é o passo, ou seja, o incremento.

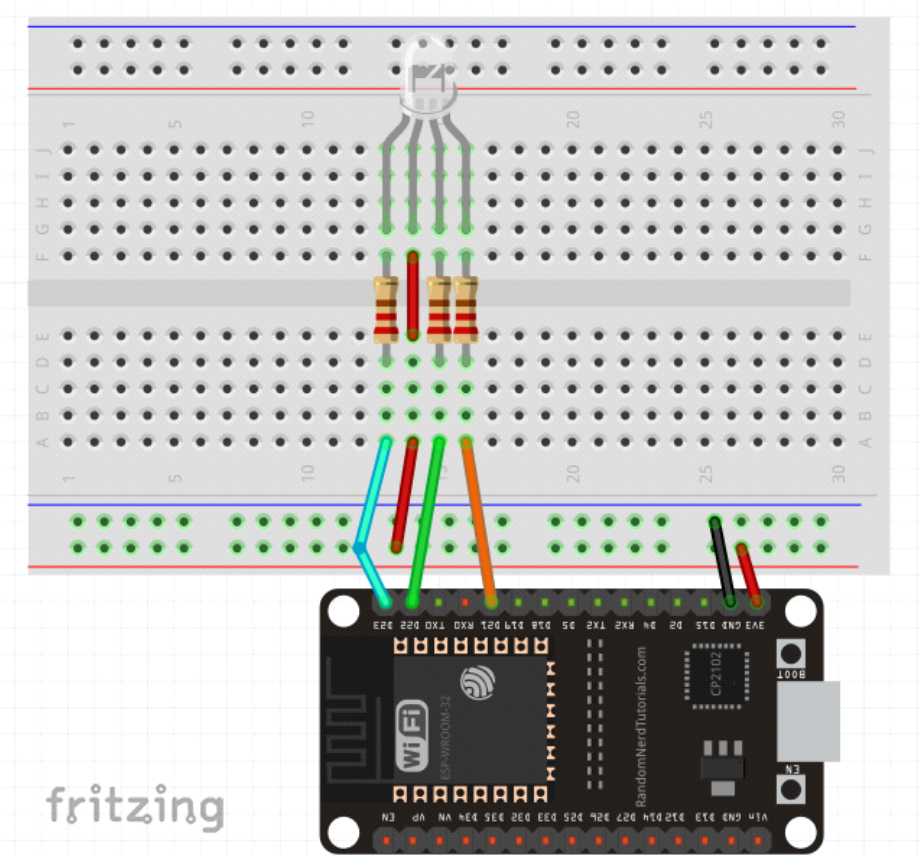
No primeiro laço de repetição, aumentaremos gradativamente a luminosidade do led. Para enviarmos um valor para a saída digital usamos a função led.**duty( )** e, o valor enviado será i, a variável de contagem que estamos utilizando no laço for, o incremento ocorrerá a cada meio segundo. No segundo, vamos repetir o mesmo processo, porém, agora diminuiremos o brilho:

```
8  while True:
9      #for --> estrutura de repetição
10     #range --> intervalo de valores com icremento
11
12     for i in range(0, 1023, 5):
13         led.duty(i)  #enviar o valor para o LED
14         sleep(0.05)
15
16     for i in range(1023,0, -5):
17         led.duty(i)  #enviar o valor para o LED
18         sleep(0.05)
```

Código 2

Agora, vamos utilizar um led RGB. Esse, é um componente que contêm três LEDs encapsulados, sendo eles, vermelho, verde e azul. A partir dessas cores, podemos criar uma grande variedade de combinações. Além disso, ele funciona utilizado PWM.

O LED RGB, pode ser de anodo comum (acende em nível baixo) ou catodo comum (acende em nível alto). Nesse caso, é um LED RGB anodo comum, por isso o conectamos ao 3V da placa.



Para encontrar outras cores pesquise por “cores html” ou “cores rgb” e encontre as cores que deseja. Copie o código a seguir na IDE Thonny:

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

red = PWM(Pin(21))
green = PWM(Pin(22))
blue = PWM(Pin(23))

def converter(x, comum):
    if comum.lower() == 'catodo':
        in_min=0
        in_max=255
        out_min=0
        out_max=1023
    elif comum.lower() == 'anodo':
        in_min=0
        in_max=255
        out_min=1023
        out_max=0
    conta = math.trunc((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min)
    return conta

def rgb(r, g, b):
    red.duty(converter(r, 'anodo'))
    green.duty(converter(g, 'anodo'))
    blue.duty(converter(b, 'anodo'))

while True:
    #Vermelho
    rgb(255, 0, 0)
    sleep(3)
    #Verde
    rgb(0, 255, 0)
    sleep(3)
    #Azul
    rgb(0, 0, 255)
    sleep(3)
    #Amarelo
    rgb(255, 255, 0)
    sleep(3)
    #Rosa
    rgb(255, 20, 147)
    sleep(3)
    #Azul Claro
    rgb(173,216,230)
    sleep(3)
    rgb(0,0,0)
```

Explicação

Primeiramente, vamos importar os módulos e funções já utilizados em aulas anteriores – machine, time e math:

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

Após isso, vamos configurar os pinos que utilizaremos. Repare, que assim como leds individuais cada led necessita de um pino:

```
red = PWM(Pin(21))
green = PWM(Pin(22))
blue = PWM(Pin(23))
```

Criaremos uma função converter, onde enviaremos dois parâmetros, primeiro o valor que desejamos converter se é um anodo ou um catodo comum. Após isso, verificaremos qual opção foi escolhida. Usaremos a função. lower para deixar todas as letras minúsculas para que não haja inconsistências. Note que, dependendo do tipo

selecionado os valores de entrada e de saída são alterados. Ademais, vamos realizar a conversão e retornar o valor da conversão:

```
def converter(x, comum):
    if comum.lower() == 'catodo':
        in_min=0
        in_max=255
        out_min=0
        out_max=1023
    elif comum.lower() == 'anodo':
        in_min=0
        in_max=255
        out_min=1023
        out_max=0
    conta = math.trunc((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min)
    return conta
```

Por conseguinte, criaremos uma função que facilitará o envio dos valores para o RGB. Nessa função os parâmetros utilizados serão os valores que desejamos enviar para o led. Dentro de **rgb**, chamar a função converter e dentro da função. duty converter o valor enviado:

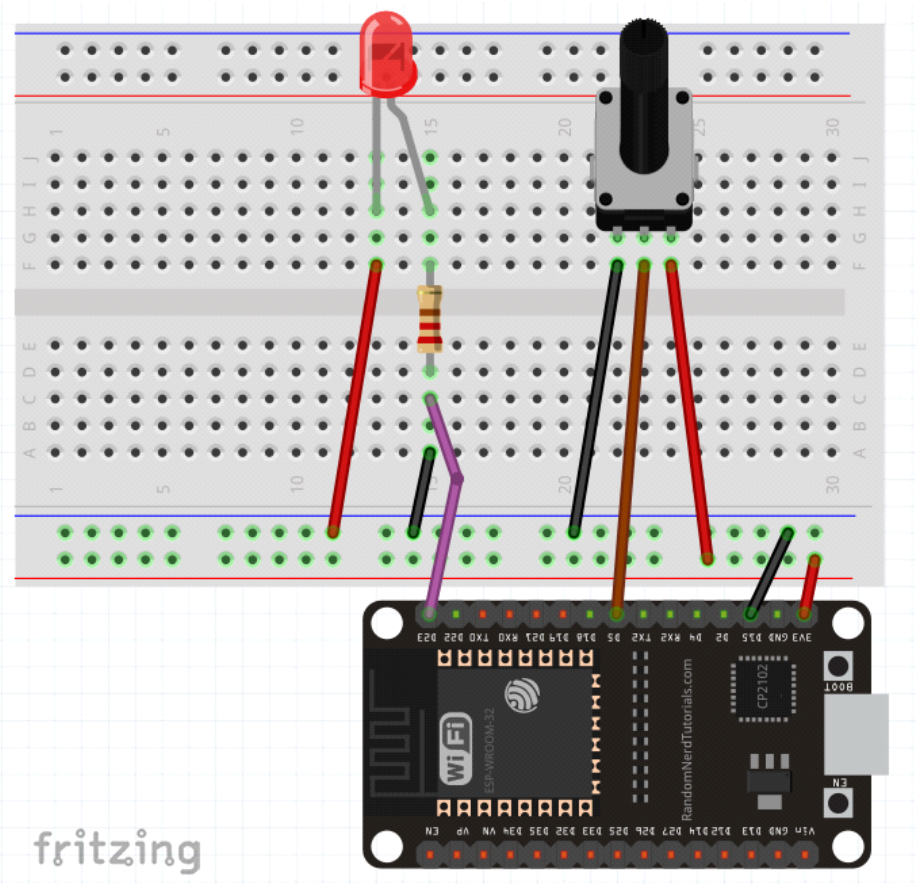
```
def rgb(r, g, b):
    red.duty(converter(r, 'anodo'))
    green.duty(converter(g, 'anodo'))
    blue.duty(converter(b, 'anodo'))
```

Por fim, vamos criar o laço de repetição infinita. Aqui vamos char a função rgb e a cada 3 segundos trocar a cor:

```
while True:
    #Vermelho
    rgb(255, 0, 0)
    sleep(3)
    #Verde
    rgb(0, 255, 0)
    sleep(3)
    #Azul
    rgb(0, 0, 255)
    sleep(3)
    #Amarelo
    rgb(255, 255, 0)
    sleep(3)
    #Rosa
    rgb(255, 20, 147)
    sleep(3)
    #Azul Claro
    rgb(173,216,230)
    sleep(3)
    rgb(0,0,0)
```

Exercícios

- Realize a montagem a abaixo.



Utilizando o potenciômetro, varie a intensidade luminosa do led de acordo com o valor da leitura.

Para isso, você vai necessitar da função a seguir:

```
def map(leitura):
    tensao = leitura * (3.3 / 4095)
    tensao = int(tensao * 310) #Tensão máxima(3.3)
    # 3.3 * 310 == 1023
    return tensao
```

- Utilizando um led RGB, realize as seguintes transições:

  Verde-Azul-Vermelho.

  Dica: utilize o laço de repetição for e a função range, vista em aulas anteriores.