



fritzing

# Ligações

**GP7 - Botão (GND)**

**Entrada Digital**

**GP10 - Servo Motor (Vermelho 5V, Preto GND)**

**Saída PWM**

**GP22 - LED Externo (GND)**

**Saída PWM**

**GP26 - Buzzer Passivo**

**Saída Analógica**

**GP27 - Potenciômetro (3,3V e GND)**

**Entrada Analógica**

# Entradas e Saídas Digitais

O modo mais simples de fazer a interface com qualquer hardware é a partir das entradas e saídas digitais.

Em CircuitPython isso é feito utilizando o módulo **digitalio**.

Para utilizá-lo é muito simples. Basta inserir o comando:  
**import digitalio**

para importar todos as suas funcionalidades, ou apenas:  
**from digitalio import DigitalInOut, Direction, Pull**

# Piscando o Led quando pressiona botão

```
import time, board  
from digitalio import DigitalInOut, Direction, Pull  
led = DigitalInOut (board.GP22) # ou board.LED  
led.direction = Direction.OUTPUT  
button = DigitalInOut (board.GP7)  
button.direction = Direction.INPUT  
button.pull = Pull.UP
```

```
while True:  
    if button.value == False:  
        led.value = not led.value  
        time.sleep (0.5)
```



# Entradas e Saídas Analógicas

Por outro lado, para trabalhar com entradas e saídas analógicas, nós utilizamos o módulo **analogio**.

Para utilizá-lo é muito simples. Basta inserir o comando:  
**import analogio**

O módulo **analogio** possui apenas duas classes:  
**AnalogIn** e **AnalogOut**, com valores (**value**) que variam de 0 a 65535.

# Verificando tensão analógica

```
import time , board  
from analogio import AnalogIn  
  
analog_in = AnalogIn(board.GP27) #board.A1  
  
def get_voltage(pin):  
    return (pin.value * 3.3) / 65536  
  
while True:  
    print((get_voltage(analog_in),))  
    time.sleep(0.1)
```

# Mapeando valores de entrada

```
import time , board  
from analogio import AnalogIn  
  
analog_in = AnalogIn(board.GP27) #board.A1  
  
def map_range(s,a1,a2,b1,b2):  
    # Esta função funciona como Arduino Map()  
    return b1 + (( s - a1 ) * (b2 - b1) / (a2 - a1))  
  
while True:  
    tensao = map_range (analog_in.value, 0, 65535, 0, 3.3)  
    print (tensao)  
    time.sleep(0.1)
```

# Saída Analógica usando DAC

```
import board  
from analogio import AnalogOut  
  
analog_out = AnalogOut(board.A0) # board.GP26  
  
while True:  
    # Count up from 0 to 65535, with 64 increment  
    # which ends up corresponding to the DAC's 10-bit range  
    for i in range(0, 65535, 64):  
        analog_out.value = i
```

```
import array, math
import board, time, digitalio

# para tratar amostras binárias brutas
from audiocore import RawSample

try:
    # Se houver AudioOut na plaquinha
    from audioio import AudioOut
except ImportError:
    try:
        # Senão, vamos por PWM mesmo
        from audiopwmio import PWMAudioOut as AudioOut
    except ImportError:
        pass # Nem toda placa dá conta de fazer isso

button = digitalio.DigitalInOut(board.GP7)
button.switch_to_input(pull=digitalio.Pull.UP)
```

# Sinal de Audio Senoidal Puro Usando DAC

```
tone_volume = 0.1 # Volume do tom
frequency = 440 # Frequência em Hertz, do tom a ser gerado
length = 8000 // frequency # Duração arredondado

# Cria um array zerado, em Hexa do tamanho do comprimento
sine_wave = array.array("H", [0] * length)

# Preenche esse array com valores gerando onda senoidal
for i in range(length):
    # para cada indice do array, gera uma amplitude
    sine_wave[i] = int((1 + math.sin(math.pi * 2 * i / length)) * tone_volume * (2 ** 15 - 1))

audio = AudioOut(board.A0) # board.GP26
sine_wave_sample = RawSample(sine_wave)

while True:
    if not button.value:
        audio.play(sine_wave_sample, loop=True)
        time.sleep(1)
        audio.stop()
```

# Instalando novas bibliotecas

Nós podemos adicionar novas bibliotecas à nossa instalação padrão de **CircuitPython**. Uma maneira bem simples é copiar a sua biblioteca para a pasta **lib**, na sua unidade **CIRCUITPY**.

Isso pode ocasionar alguns erros, se você não souber exatamente o que está fazendo.

Com o Python3 já instalado, a maneira mais segura é utilizando a ferramenta **circup**.

Para instalá-la, abra uma janela de terminal e digite:  
**pip3 install circup**

# Instalando novas bibliotecas

O **pip** irá instalar no seu computador a ferramenta **circup**, que é um gerenciador de pacotes para o seu **CircuitPython**. Uma lista completa de bibliotecas disponíveis pode ser consultada em  
**<https://circuitpython.org/libraries>**

Mas você também pode visualizar utilizando o circup. Digite o comando:  
**circup show**

Para o próximo exercício com **Servo Motor**, utilizaremos a biblioteca **adafruit\_motor**, que pode ser instalada com o comando:  
**circup install adafruit\_motor**

# Movendo Servo com Potenciômetro

```
import time, board, pwmio
from adafruit_motor import servo
from analogio import AnalogIn

analog_in = AnalogIn(board.GP27) #board.A1
pwm = pwmio.PWMOut(board.GP10, duty_cycle=2 ** 15, frequency=50)
my_servo = servo.Servo(pwm)

def map_range(s,a1,a2,b1,b2):
    # Esta função funciona como Arduino Map()
    return b1 + (( s - a1 ) * (b2 - b1) / (a2 - a1))

while True:
    angulo = map_range(analog_in.value, 0, 65535, 0, 180)
    my_servo.angle = angulo
    time.sleep(0.1)
```

# Tocando Ringtones

Instale a biblioteca Adafruit\_RTTTL:

**circup install adafruit\_rtttl**

**No site <https://github.com/neverfa11ing/FlipperMusicRTTTL> você encontra uma série de músicas para baixar.**

Em seguida, digite o seguinte script CircuitPython:

```
import board, adafruit_rtttl

buzzer = board.A0 # board.GP26
music =
"Snowman:d=8,o=5,b=200:2g,4e.,f,4g,2c6,b,c6,4d6,4c6,4b,a,2g.,b,c6,4d6,4c6,4b,a,a,g,4c6,4e.,g,a,4g,4f,4e,4d,2c.,4c,4a,4a,4c6,4c6,4b,4a,4g,4e,4f,4a,4g,4f,2e.,4e,4d,4d,4g,4g,4b,4b,4d6,4d6,b,4d6,4c6,4b,4a,4g,4p,2g"
adafruit_rtttl.play(buzzer, music)
```