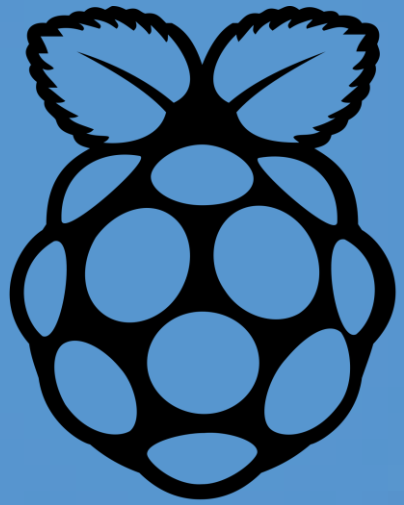
CURSO DE RASPBERRY PI



Clase 3

Librerías para manejar los pines GPIO.

La clase pasada manejamos los pines GPIO con un módulo pre-instalado (Rpi.GPIO). No hace falta que instalemos ningún paquete para poder hacer uso del mismo.

Si deseamos usarlo debemos importarlo a nuestro script (import Rpi.GPIO) como vimos en el ejemplo para hacer parpadear un led.

Ahora bien, no es la única librería para poder manipular los pines digitales de la Raspberry .

En el transcurso de la clase de hoy seguiremos con RPi.GPIO y abordaremos una nueva llamada GPIOZERO.

También veremos la forma de trabajar con entradas digitales y señales PWM.

Módulo RPi.GPIO

Autor: Ben Croston

Aviso: este módulo no es adecuado para aplicaciones de tiempo real (críticas). Esto se debe a que no se puede predecir como se desenvolverá nuestra aplicación ya que se ejecuta bajo Linux, que es un sistema operativo multitarea y otro proceso puede tener prioridad sobre nuestro script, lo que puede causar el mal funcionamiento del programa.

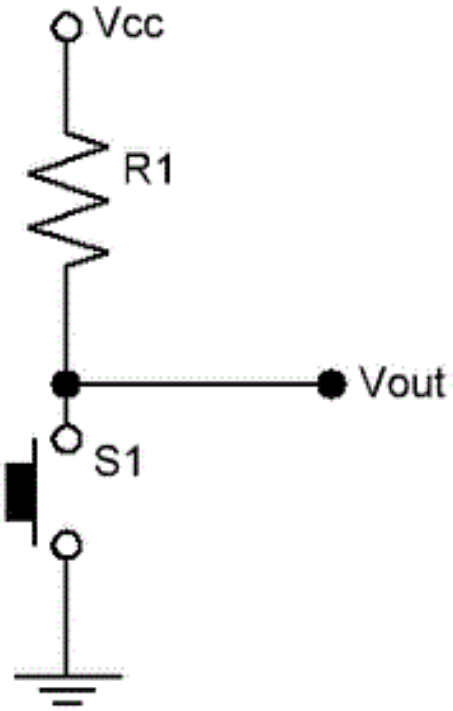
Instalación: Instalado en las últimas versiones de Raspbian.

Bibliografía y ejemplos:
https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/

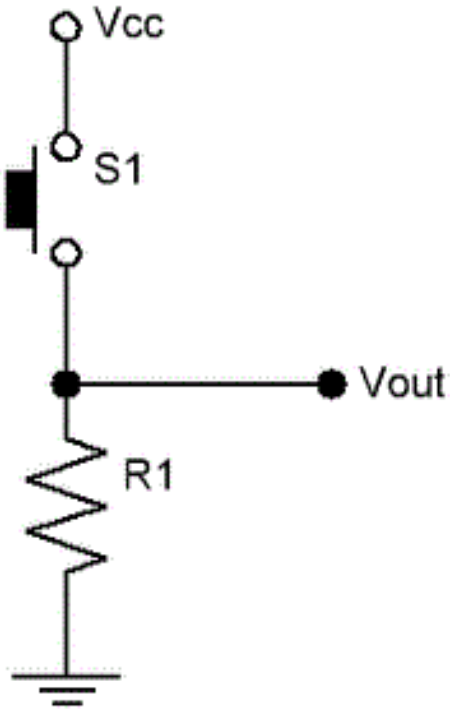
Usando RPi.GPIO

```
import RPi.GPIO as GPIO # importar el módulo como GPIO
GPIO.setmode(GPIO.BOARD) # Por número de pin en placa
GPIO.setmode(GPIO.BCM) # Por numero de canal GPIO

GPIO.setup(channel, GPIO.OUT) # Como salida
GPIO.output(12, True OR False) #Encendido – Apagado

GPIO.setup(channel, GPIO.IN) # Como entrada, usar R Pull up o Pull Down
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP) # Pull up por Soft
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Pull Down por soft
if GPIO.input(channel) # Para preguntar por el estado de una entrada

#PWM
p = GPIO.PWM(channel, frequency) # instancia del objeto efecto con canal/Freq
p.start(dc) # DC es el ciclo de trabajo valores de 0 (0%) y 1 (100%)
p.ChangeDutyCycle(dc) # cambio del ciclo de trabajo
p.stop()
```

Ciruitos Pull up y Pull Down

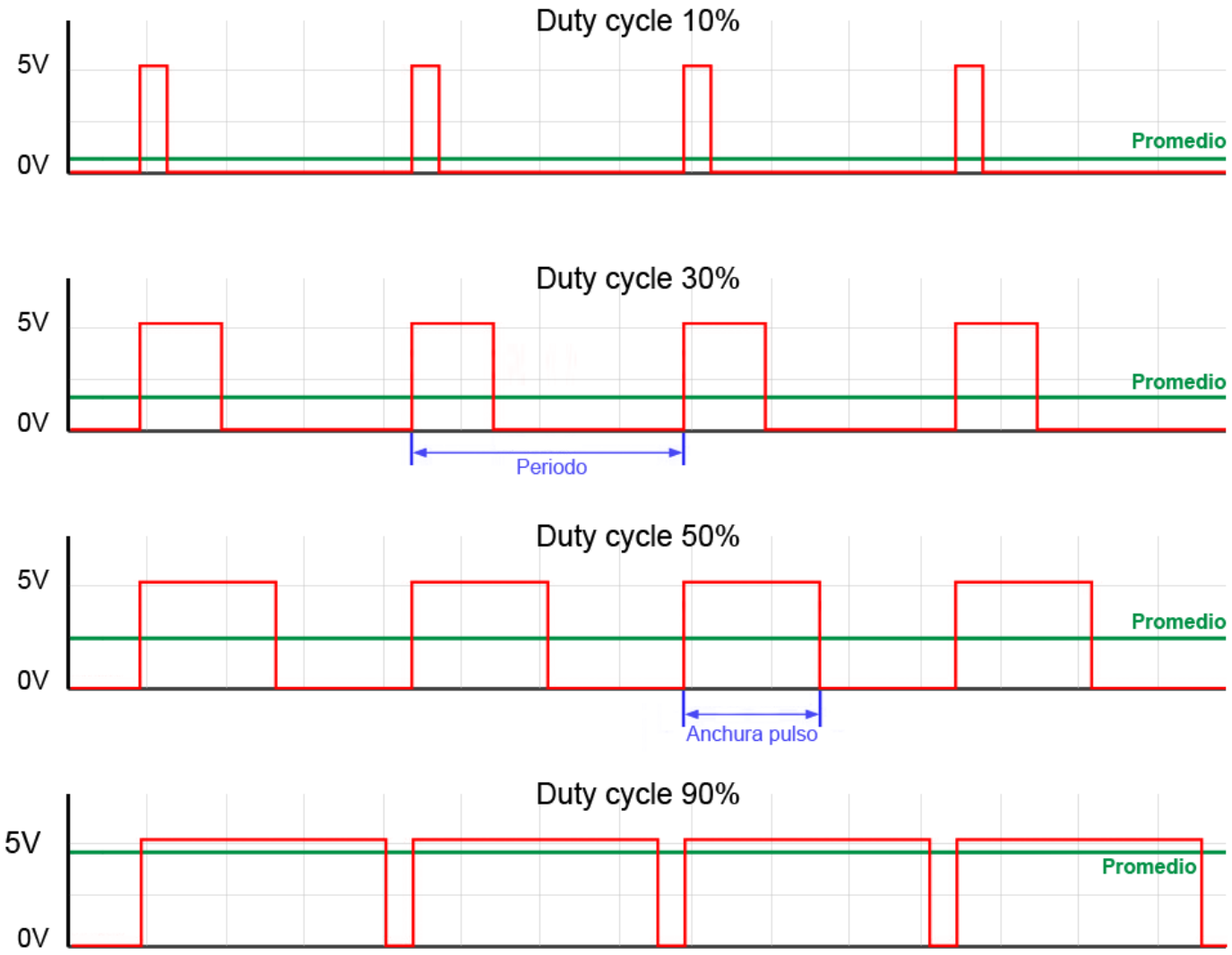


Pull Up



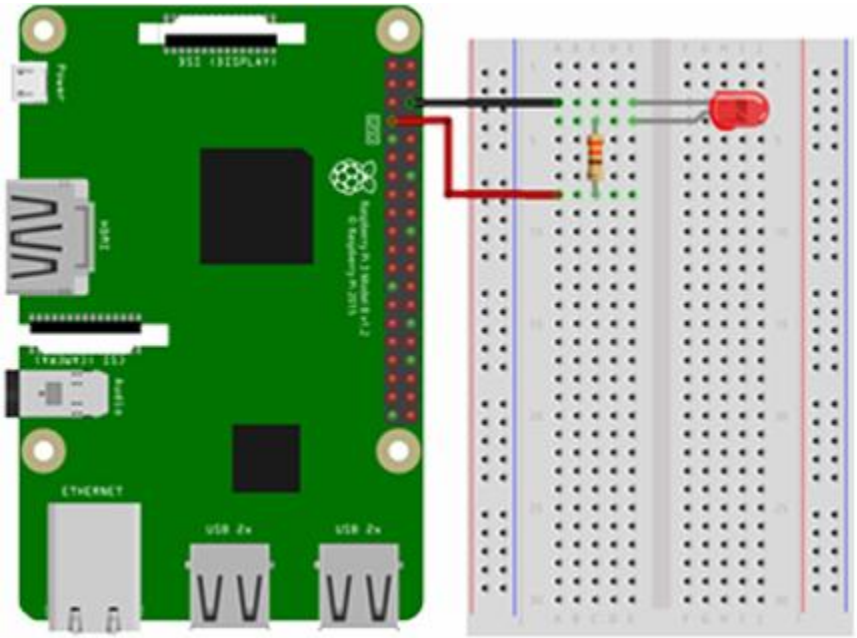
Pull Down

Señal PWM

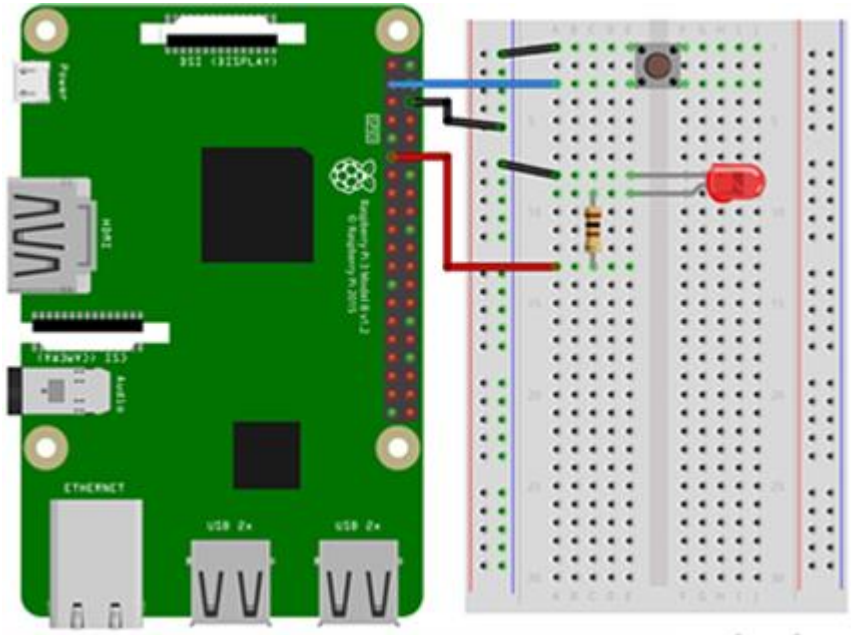

Duty cycle 10%
5V
0V
Promedio
Duty cycle 30%
5V
0V
Promedio
Periodo
Duty cycle 50%
5V
0V
Promedio
Anchura pulso
Duty cycle 90%
5V
0V
Promedio

Circuitos para las prácticas

Posible circuito para la práctica de salida digital y PWM.



Posible circuito para la práctica entrada digital.



Módulo GPIOZERO
(Otra librería para manipular los pines)

Autor: Ben Nuttall y Dave Jones

Aviso: La biblioteca incluye interfaces para muchos componentes cotidianos simples, así como algunas cosas más complejas como sensores, LEDs RGB y más. Consulte la documentación para obtener ideas sobre cómo comenzar

Instalación mediante el Shell de Raspbian con permisos de SuperUser:

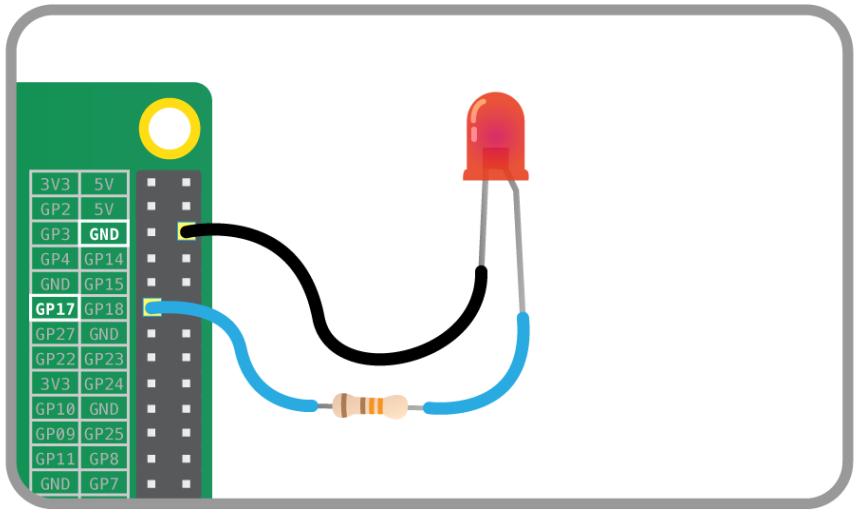
```
Para Python 3:
pi@raspberrypi:~$ sudo apt install python3-gpiozero
Para Python 2:
pi@raspberrypi:~$ sudo apt install python-gpiozero
```

También se puede usar la herramienta pip de python para la instalación.

Bibliografía y ejemplos:
https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/

Ejemplo salida digital con GPIOZERO

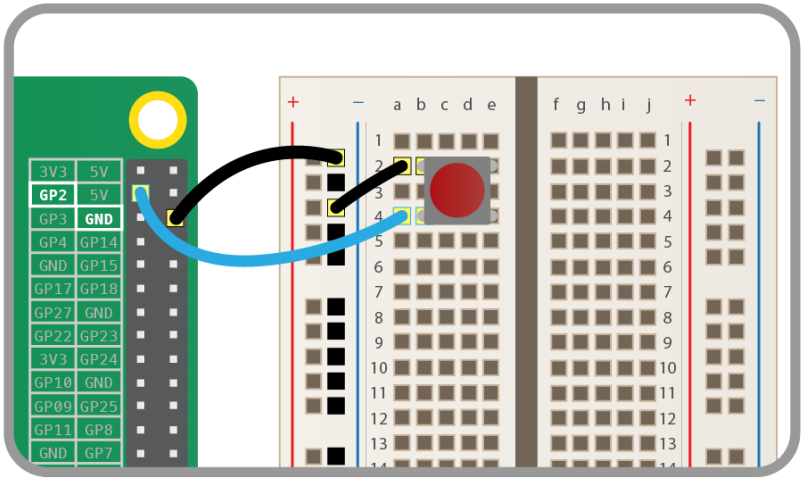
```
from gpiozero import LED
from time import sleep

red = LED(17)

while True:
    red.on()
    sleep(1)
    red.off()
    sleep(1)
```



Este script para hacer parpadear un led nos permite entender como debemos manipular los pines digitales en esta librería.
Sencillo y sin demasiadas complicaciones, con una sintaxis muy elocuente.
Gracias a esta librería armamos una instancia LED, donde creamos el objeto " red" .

Ejemplo de entrada digital con GPIOZERO

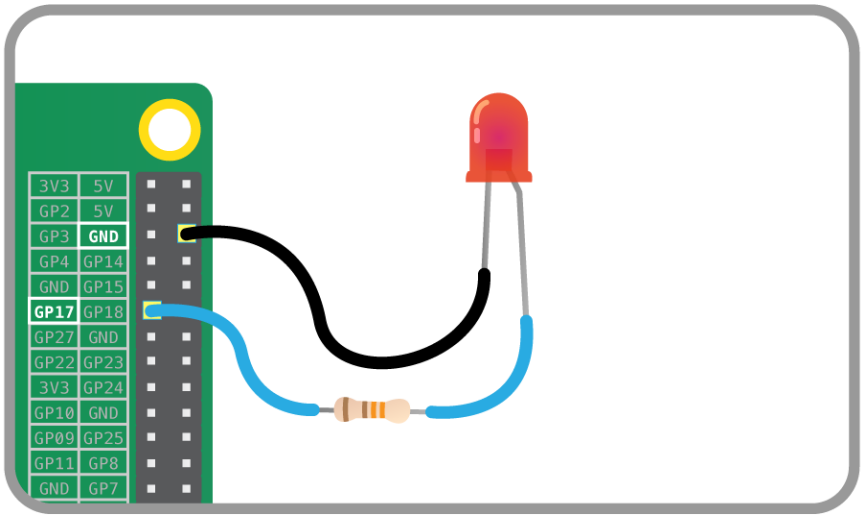
```
from gpiozero import Button

button = Button(2)

while True:
    if button.is_pressed:
        print("Button is pressed")
    else:
        print("Button is not pressed")
```



Este script muestra en pantalla los estados del pulsador.
No usamos resistencias, no son necesarias en esta biblioteca, ya que sólo acepta una sola configuración, pull up por software. A nosotros mucho no nos interesa, porque evaluamos el estado con " is_pressed ".
Instancia Button, donde creamos el objeto "button".

Ejemplo de PWM con GPIOZERO

```
from gpiozero import PWMLED
from time import sleep

led = PWMLED(17)

while True:
    led.value = 0
    sleep(1)
    led.value = 0.5
    sleep(1)
    led.value = 1
    sleep(1)
```



Usamos como ejemplo este script que sirve para modificar el brillo de un led. La técnica empleada para dicha acción es PWM, como vimos anteriormente. Misma idea que los ejemplos anteriores. Instaciamos con PWMLED, al objeto " led". Valor 0 (0% del Ciclo de Trabajo) y valor 1 (100 % del Ciclo de Trabajo).

Conversores de Nivel lógico

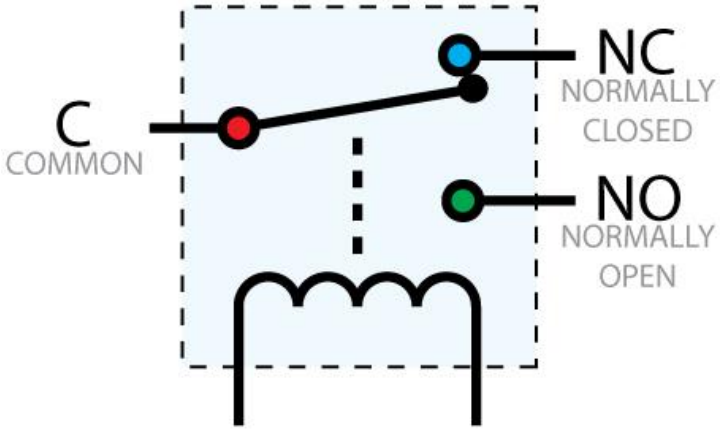
Conectar un dispositivo que funcione a 3.3V a uno que funcione con 5V, no siempre es una tarea fácil.

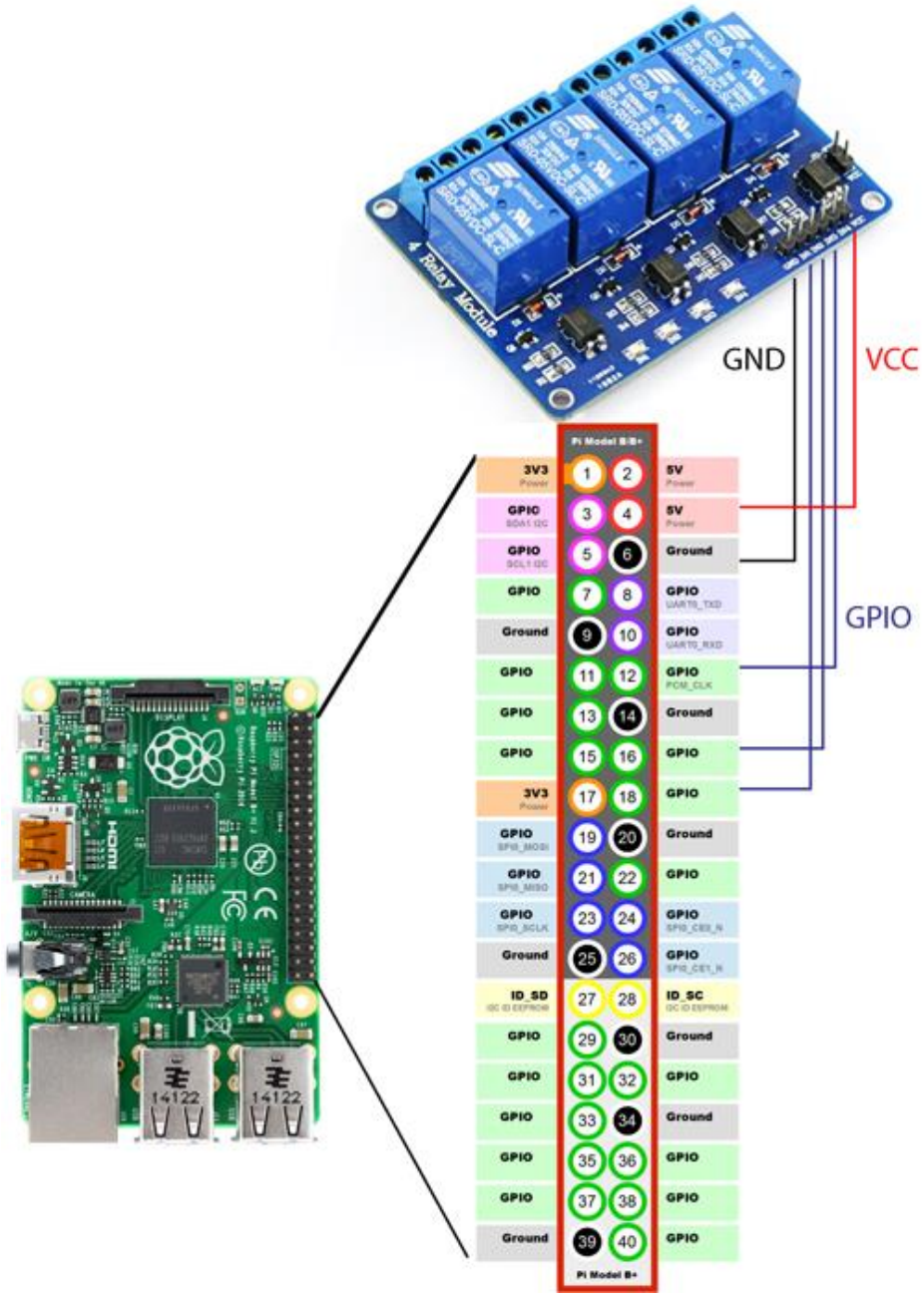
Existen diferentes placas y módulos electrónicos que actúan de conversor y permiten la coexistencia de estos niveles. Los hay en una sola dirección y bidireccionales. Ya sea de 3.3V a 5V o a la inversa (5V a 3.3V).
Por lo general estos dispositivos necesitan ser alimentadas por las dos tensiones a convertir, y sólo funciona con señales digitales (niveles altos y bajos). NO funcionan con señales analógicas.

No siempre es necesario emplear de estos módulos, en la web podemos encontrar diferentes circuitos y formas de hacer lo mismo sin necesidad de estos. Pero implican ciertos conocimientos avanzados en electrónica.

Módulo relé con Raspberry

ADVERTENCIA ES RECOMENDABLE NO HACER ESTA PRÁCTICA SI NO SE DISPONEN DE CONOCIMIENTOS EN INSTALACIONES ELÉCTRICAS QUE INVOLUCREN TENSIONES Y CORRIENTES QUE PUEDEN OCACIONAR DAÑOS EN LA SALUD. QUEDA TOTALMENTE BAJO RESPONSIBILIDAD DE CADA UNO.





Conexiones

ADVERTENCIA ES RECOMENDABLE NO HACER ESTA PRÁCTICA SI NO SE DISPONEN DE CONOCIMIENTOS EN INSTALACIONES ELÉCTRICAS QUE INVOLUCREN TENSIONES Y CORRIENTES QUE PUEDEN OCACIONAR DAÑOS EN LA SALUD. QUEDA TOTALMENTE BAJO RESPONSIBILIDAD DE CADA UNO.