

Timers con Python: Actividad de clase

¿Para qué los queremos?

Imaginad que queréis leer el valor de un sensor de Temperatura cada 5 segundos.

¿Cómo lo haríais en python?

Imaginad que tenéis la función en python

Rpi_Leer_temp()

Timers con Python: Actividad de clase

- a) Usando la function *time.sleep(seconds)*
- ¿Qué occurred durante esos 5 segundos de *sleep*?
 - ¿Y si cada 0.2 segundos queremos comprobar si se ha pulsado un pulsador?
- b) comprobar el tiempo actual “*con mucha frecuencia*” y comprobar si han pasado 5 segundos.
1. Comprobar si se ha pulsado el pulsador
 2. Comprobar si han pasado 5 segundos para leer el valor del sensor de temperature
 3. Volver a 1.

Timers con Python: Actividad de clase

- b) comprobar el tiempo actual “*con mucha frecuencia*” y comprobar si han pasado 5 segundos.

Dos posibles simples opciones (¡hay muchas más!)

1) Usar la función ***time.time()***

Esta función devuelve el número de segundos desde el **epoch**

El epoch es el punto de inicio para contar el tiempo

En Unix, el epoch es January 1, 1970, 00:00:00 (UTC).


2) Usar la función ***time.perf_counter()***

Devuelve el número de segundos que han pasado desde un punto de referencia desconocido.

Dos llamadas consecutivas a estas funciones nos indica el tiempo que ha pasado entre las llamadas

Semana 4.

Sensores y Actuadores.

SISTEMAS EMBEBIDOS 
(EMBEDED SYSTEMS)

Grado Dual en Industria Digital

Campus Vitoria

Curso 2020-2021

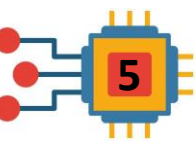


Deusto

Facultad de Ingeniería
Ingeniaritza Fakultatea

¿Qué es un sensor?

- Dispositivo que responde a cualquier **cambio** en un fenómeno físico variables ambientales como calor, humedad, presión, movimiento, etc
- Este **cambio** afecta las propiedades físicas, químicas, o electromagnéticas de los sensores, que se procesa posteriormente para poder leer ese cambio y obtener información significativa.
- Un sensor es la parte fundamental de los sistemas de medición: está en contacto con variables del entorno, y proporciona una salida



Tipos de sensores

PADLET

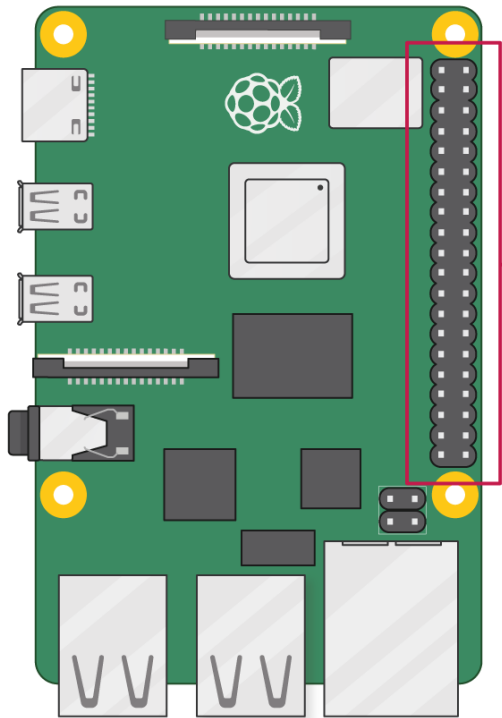
<https://deustoingenieria.padlet.org/lauraarjona/n0rw8emjsbft4nt7>

¿Cómo proporciona la salida el sensor?

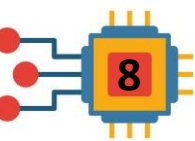
- *Digital*
- *Analógico (uso de PWM)*
- *En serie*
- *Protocolo comunicación Sincrono (I2C y SPI)*
- *Protocolo comunicación asíncrono (UART)*

GPIOs

Any of the GPIO pins can be designated **(in software)** as an input or output pin and used for a wide range of purposes.

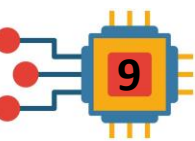


3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)



RPi pinout

```
pi@laurapi:~ $ pinout
```



GPIOs

Voltages

Dos pines de 5V y dos de 3V3, y varios pines de tierra (Ground) que no se pueden reconfigurar. El resto de los pines son pines de 3V3 voltios de proposito general.

Outputs

Un pin GPIO de salida se le puede asignar el valor high (3V3) o low (0V).

Inputs

Un pin GPIO de salida se le puede leer el valor high (3V3) o low (0V).

Los pines GPIO2 y GPIO3 tienen pull-up resistors fijas, pero el resto se pueden configurar por software.

¿Como program los GPIOs?

A) En Python, usando una libreria Rpi.GPIO

```
$ sudo apt-get update
```

```
$ sudo apt-get install rpi.gpio
```

B) En shell script, usando los drivers de Linux

Blinking LED con BASH

```
#!/bin/bash
#Exportamos el puerto GPIO 17
echo 17 > /sys/class/gpio/export

#Lo configuramos como salida
echo out > /sys/class/gpio/gpio17/direction

#Encendemos el LED asignandole 1 como valor lógico
echo 1 > /sys/class/gpio/gpio17/value

#Apagamos el LED asignandole 0 como valor lógico
echo 0 > /sys/class/gpio/gpio17/value

#Liberamos la entrada del puerto GPIO 17
echo 17 > /sys/class/gpio/unexport
```

Blinking LED con Python

USO DE LA LIBRERIA Rpi.GPIO

```
import RPi.GPIO as GPIO
from time import sleep #

GPIO.cleanup()
GPIO.setmode(GPIO.BCM)

GPIO.setup(17, GPIO.OUT, initial=GPIO.LOW) # LED GPIO 17
GPIO.output(17, GPIO.LOW) # Turn off

while True:
    sleep(1)
    GPIO.output(17, GPIO.HIGH) # Turn on
    sleep(1)
    GPIO.output(17, GPIO.LOW) # Turn off
```

GPIOs: uso con libreria Rpi.GPIO

Módulo: RPi.GPIO

```
import RPi.GPIO as GPIO
```

- Modo de numeración de pines:

```
GPIO.setmode(GPIO.BOARD)
```

```
GPIO.setmode(GPIO.BCM) #normalmente  
escogeremos este
```

- Configuración del pin (canal):

```
GPIO.setup(numcanal, modo)
```

```
GPIO.setup(19,GPIO.IN)
```

- Modos de canal:

```
GPIO.IN, GPIO.OUT,
```

- Valores predefinidos:

```
GPIO.HIGH, GPIO.LOW
```

GPIOs – Programa modelo

- `GPIO.input(numcanal)`

`GPIO.input(25)` #Devuelve `GPIO.HIGH` o `GPIO.LOW`

- Se puede configurar pull-up o pull-down para que tenga un valor por defecto cuando no hay nada conectado

`GPIO.setup(19, GPIO.IN, pull_up_down=GPIO.PUD_UP)`

`GPIO.setup(19, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)`

GPIOs – Programa modelo (.py)

```
import RPi.GPIO as GPIO
# Establecimiento del modo
GPIO.setmode(GPIO.BCM)
#Inicializaciones
GPIO.setup(numcanal, GPIO.OUT)
#.....
#.....
# Limpieza de estado
GPIO.cleanup()
```


LED (Light Emitting Diode)

Digital
on/off
PWM

https://wiki.seeedstudio.com/Grove-Variable_Color_LED/

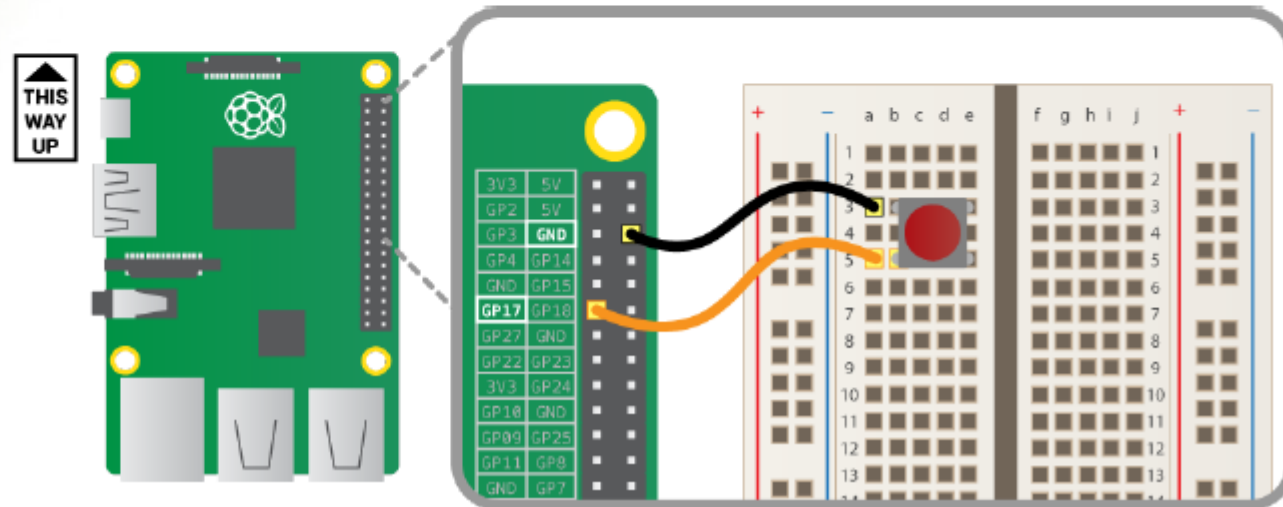


Item	Typical	Unit
Operate Voltage	5.0	VDC
Working Current	20	mA
Variable Resistor	<1	K Ω

- Se puede variar la intensidad luminosa usando la técnica PWM (veremos más adelante en esta presentación)
- Tiene tres resistencias RGB para ajustar el color. ¡Sed muy cuidados@s al modificarlas!

Push Button (pulsador)

Digital
on/off



Switch

Digital
on/off

<https://wiki.seeedstudio.com/Grove-Switch-P/>

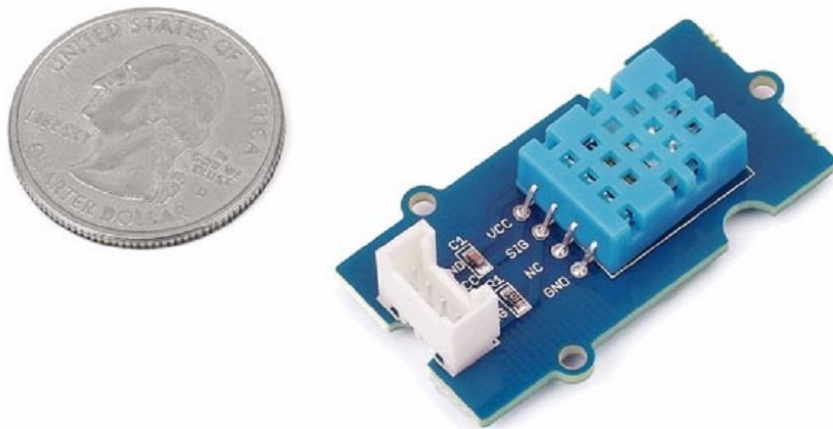


Parameter	Value/Range
Operating voltage	3.3V
Electrical Life	10,000 cycles
Operation Force	200 ± 50gf
Operation Temperature	-20°C to +80°C
Size	20mmX20mm

Temperatura y humedad

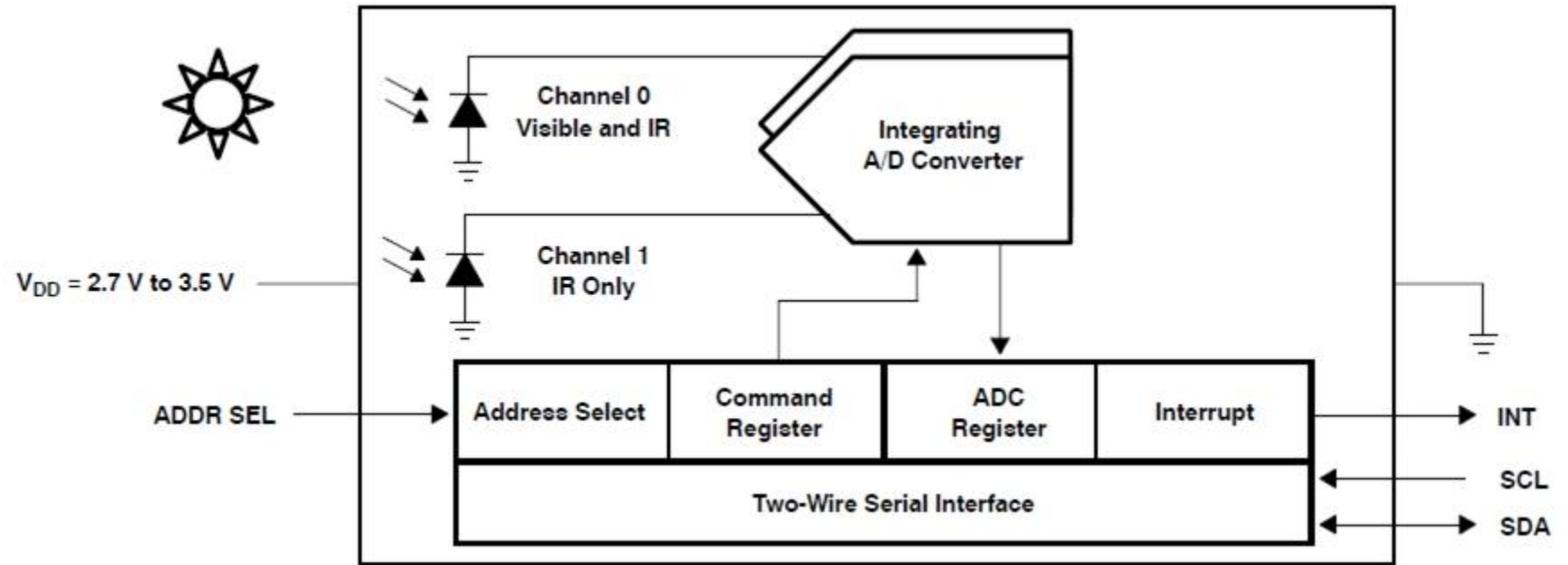
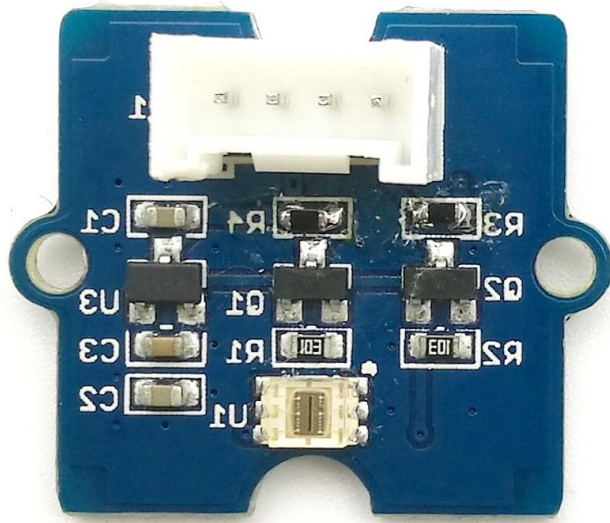
Digital

https://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/



- Elevada distancia de transmisión (>20m)
- Se necesita 5m para extraer un dato.
- ✓ Humedad relativa: elemento de sensado capacitivo
- ✓ **Termistor de coeficiente negativo** (NTC – Negative Temperature Coefficient)

https://wiki.seeedstudio.com/Grove-Digital_Light_Sensor/



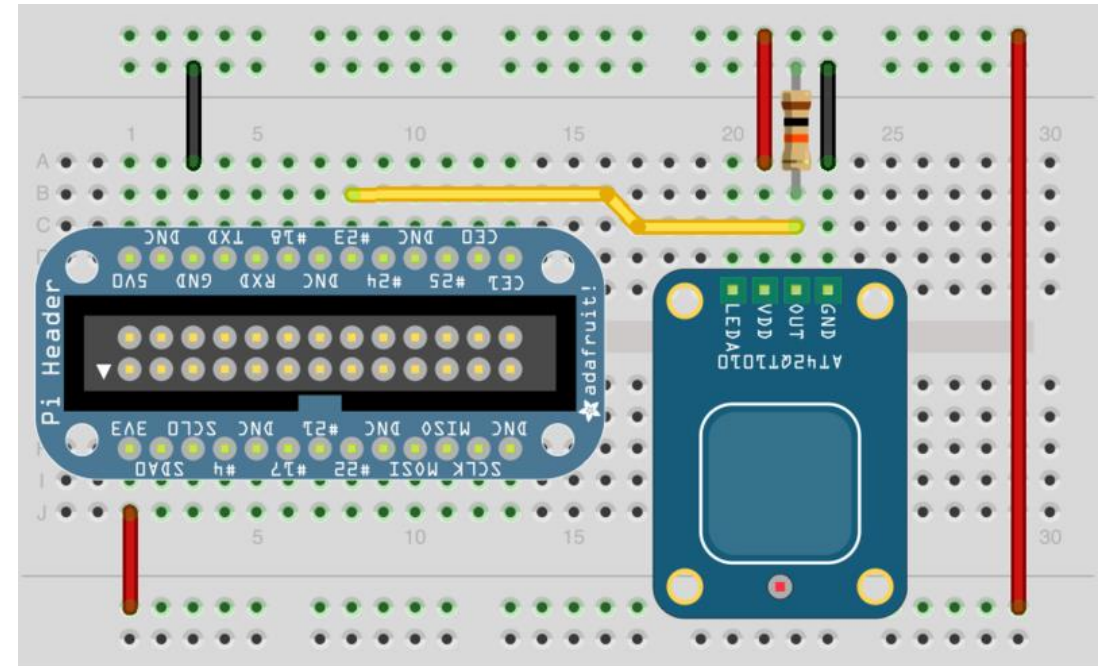
3 modos de operación:

- Espectro completo
- Infrarrojo
- Luz visible (para humanos) → ¿recordáis cuál es la longitud de onda o frecuencia?

Touch (contacto)

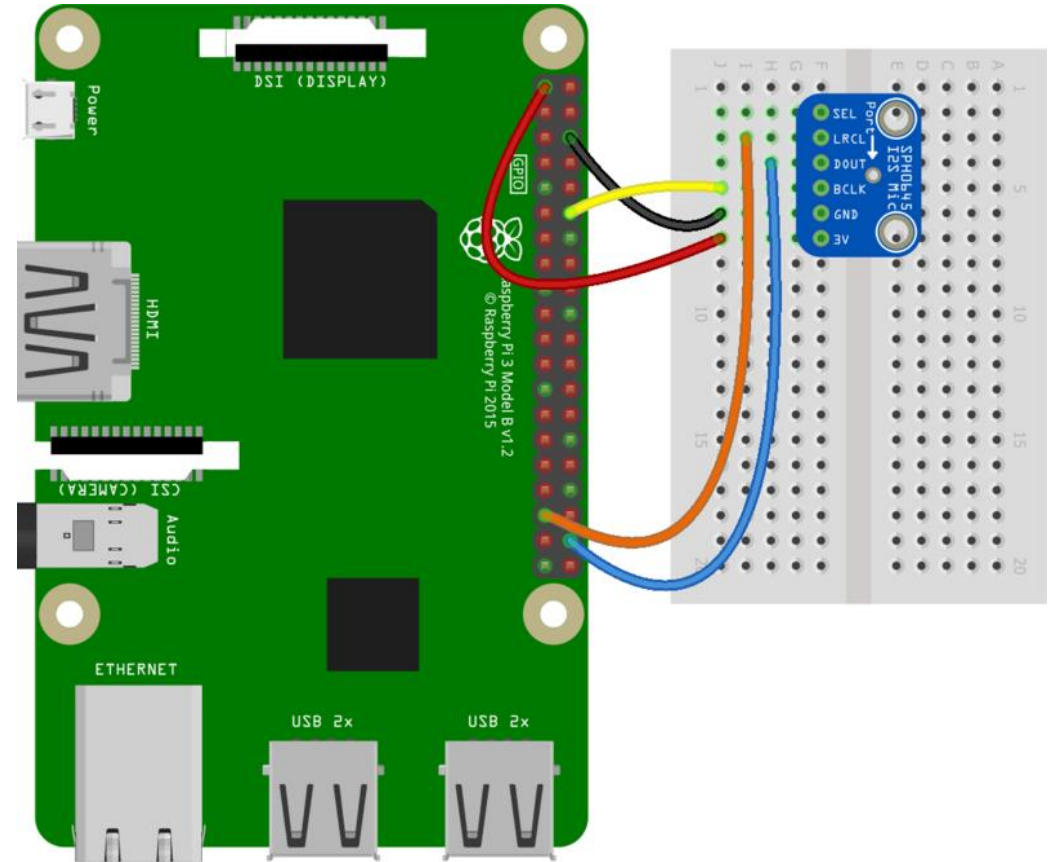
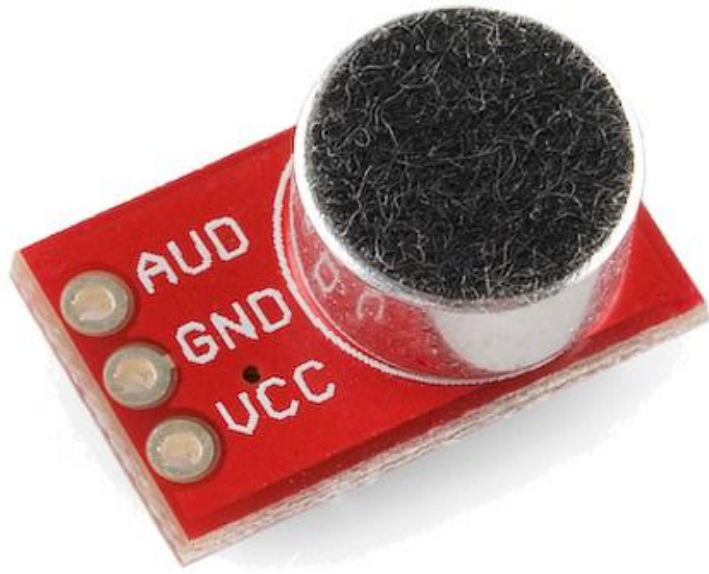
Digital

<https://learn.adafruit.com/capacitive-touch-sensors-on-the-raspberry-pi/programming>



Made with  Fritzing.org

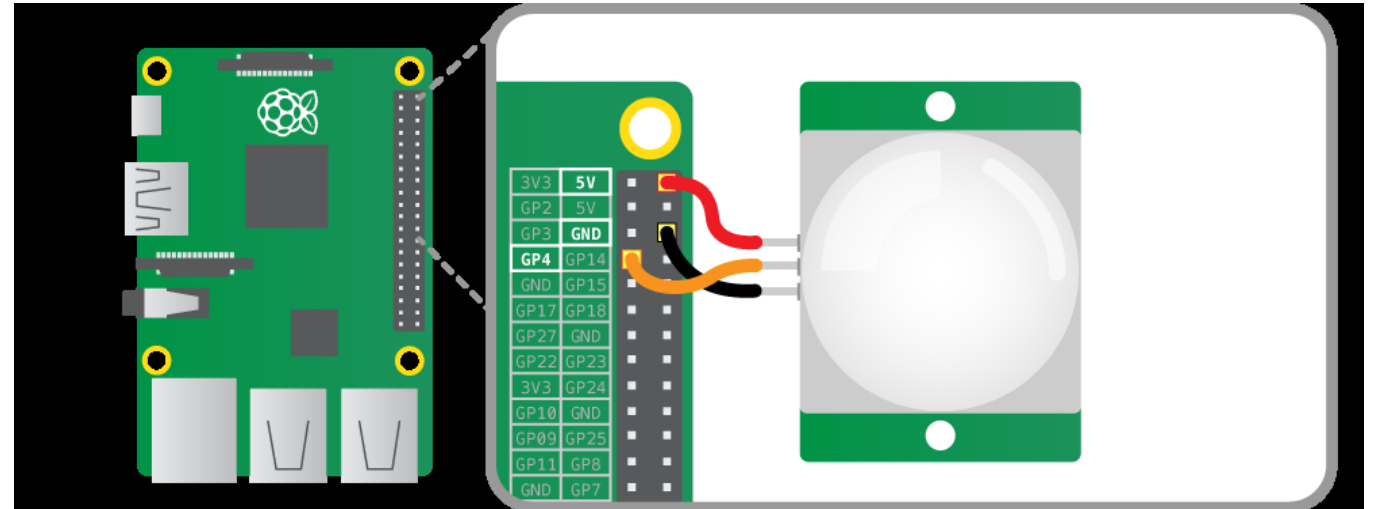
Sonido



fritzing

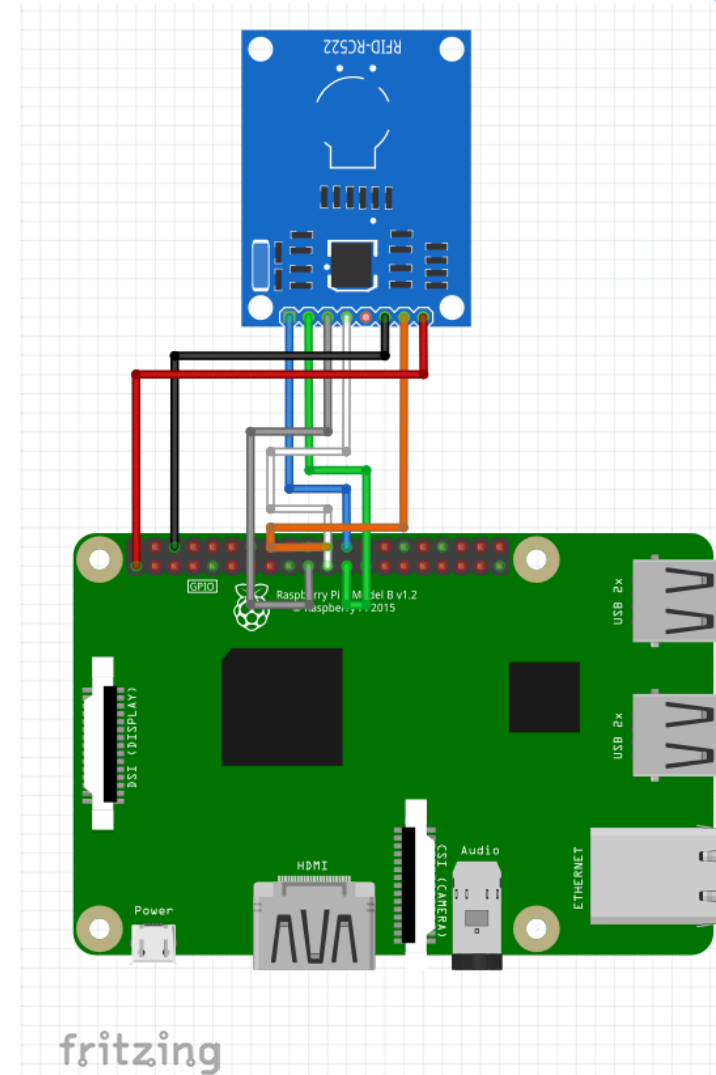
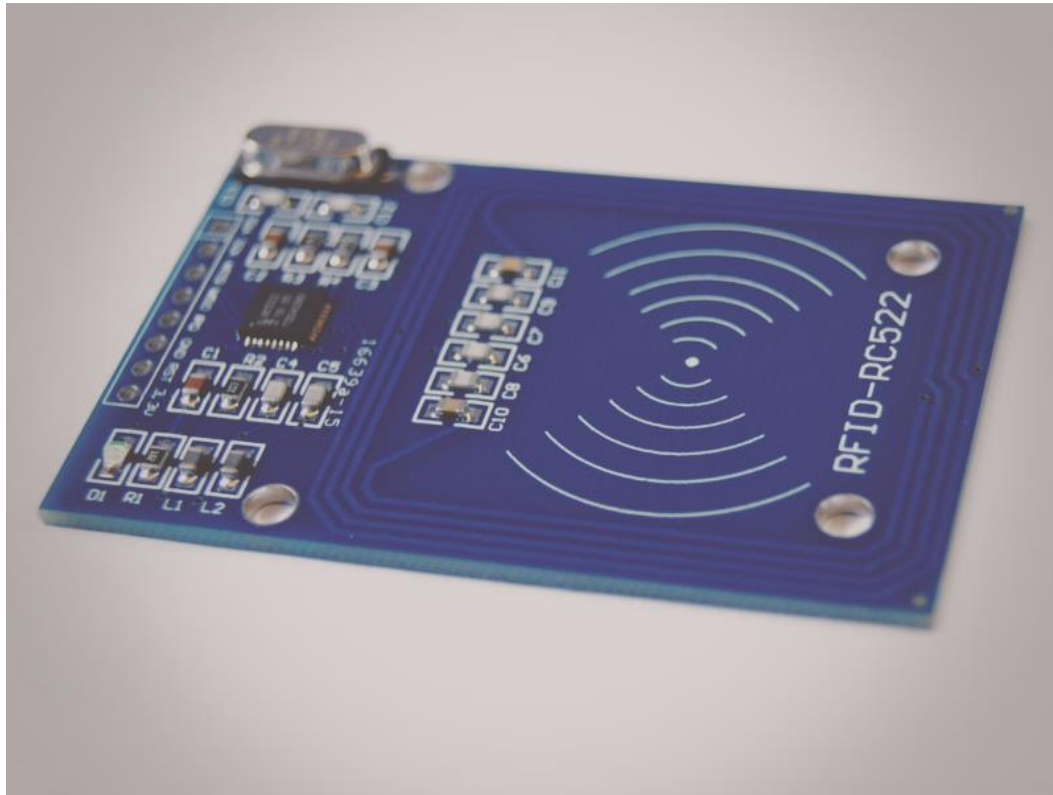
Movimiento

Digital



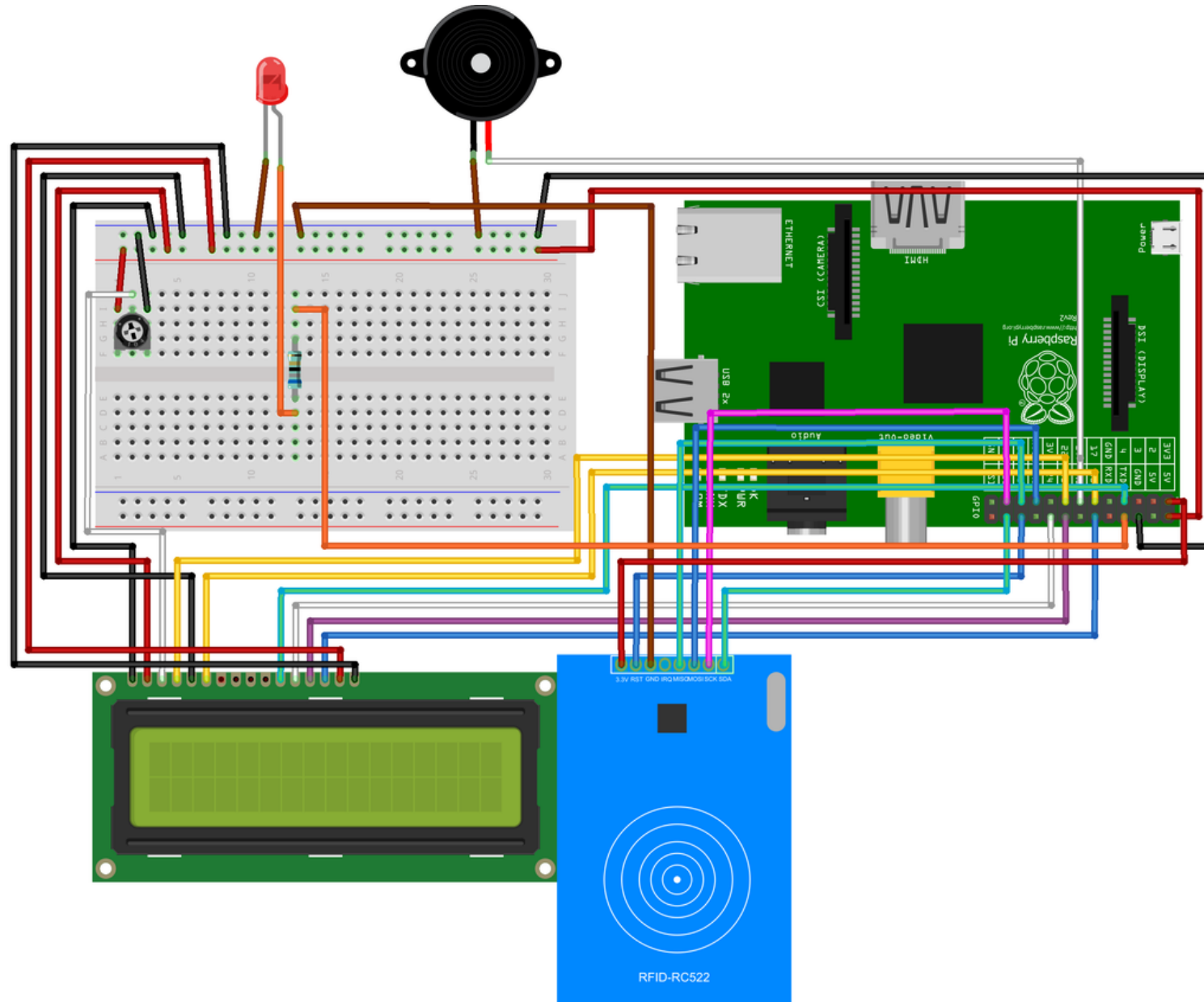
Identificación – RFID / NFC

SPI

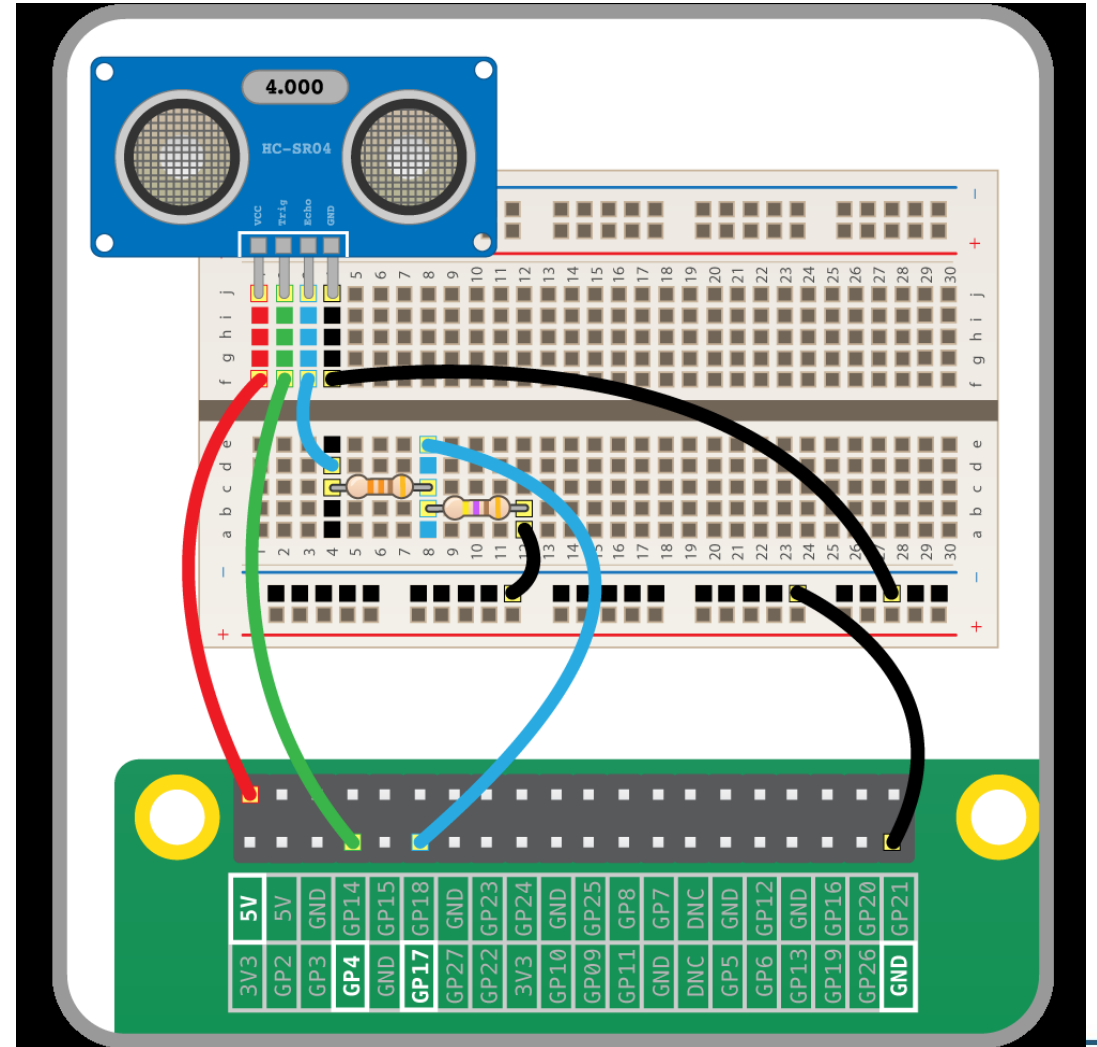


Identificación – RFID / NFC

SPI



Proximidad



Acelerómetro

I2C

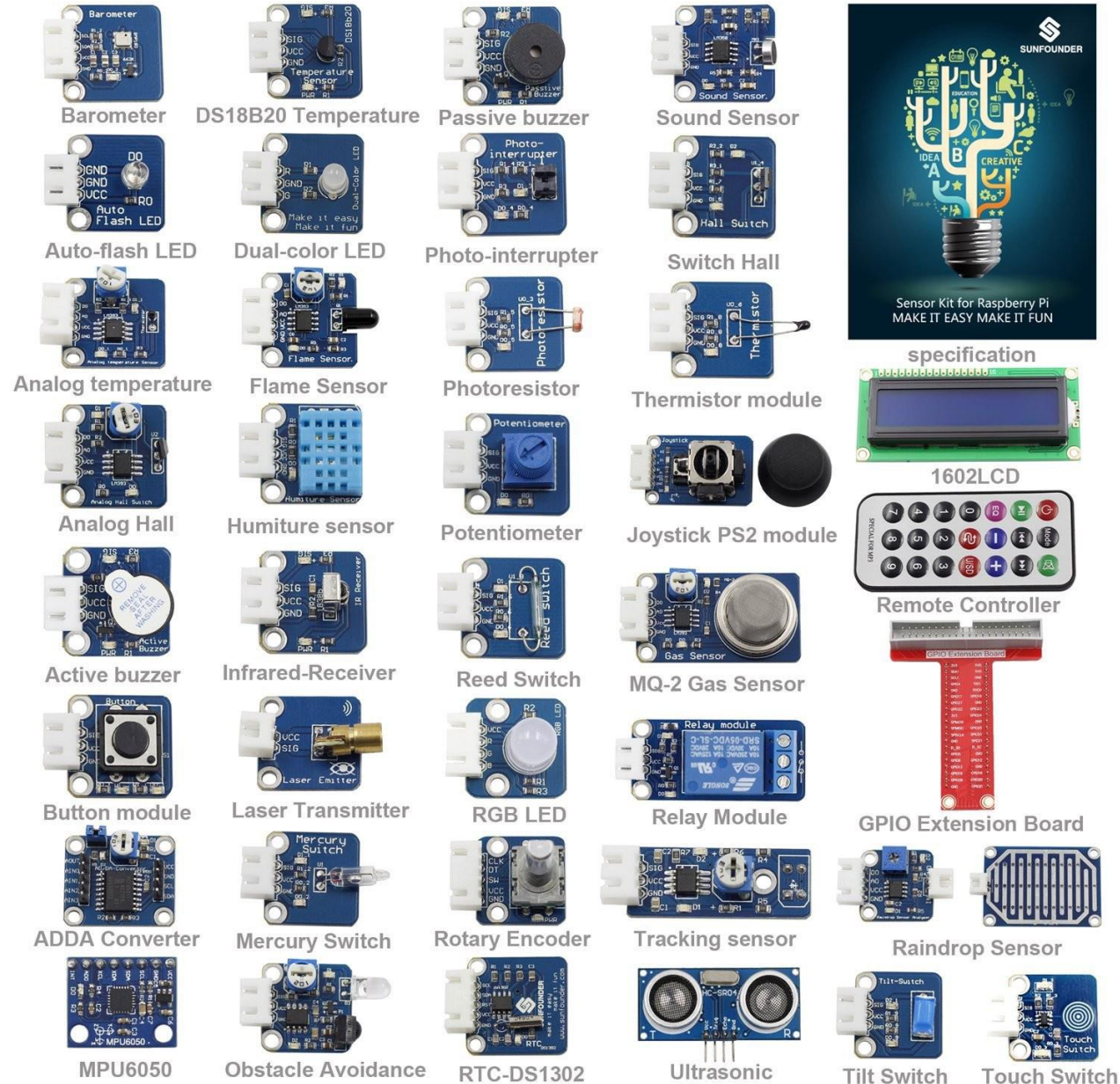
Sensor digital de 3 ejes (x,y,z)

Sensibilidad: +/- 16g

Aplicaciones: orientation detection, gesture detection and Motion detection



Y muchos más



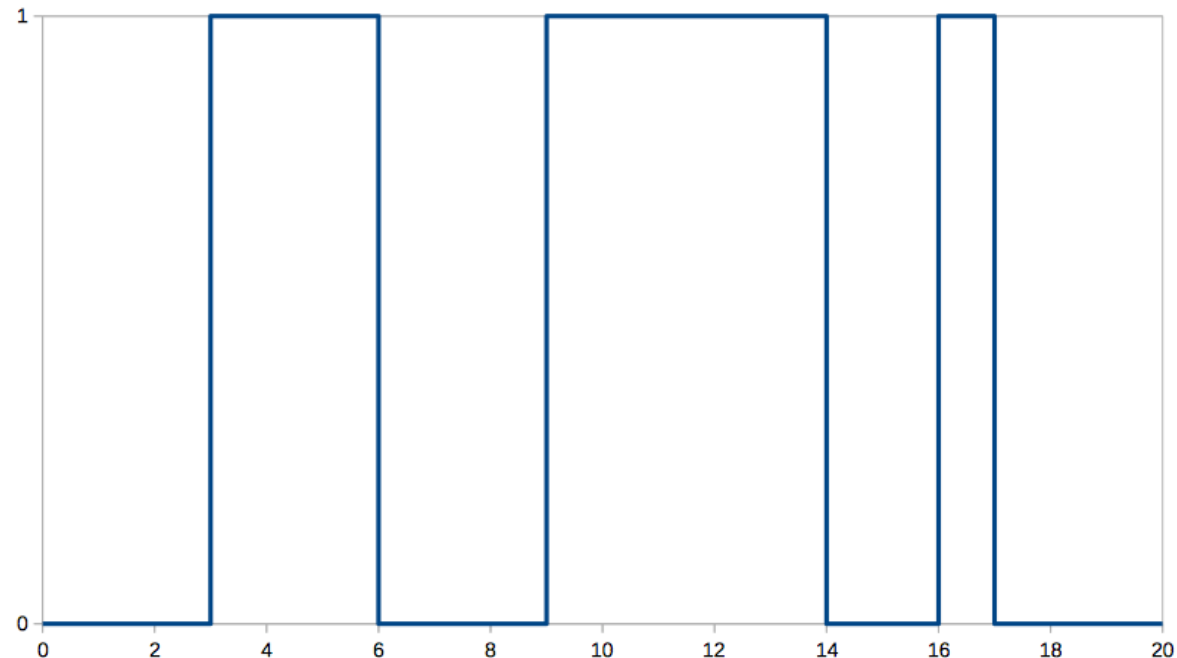
Y muchos más



¿Cómo proporciona la salida el sensor?

Digital on/off

- Algunos dispositivos simplemente asignan un valor de voltage a valor alto y valor bajo (on / off).
- Ejemplo: acelerómetro, giroscopio
- Se leen simplemente leyendo un valor digital del pin (0 -> 0 voltios; 1 -> 3.3 voltios)



Eje Y:
Lectura del pin

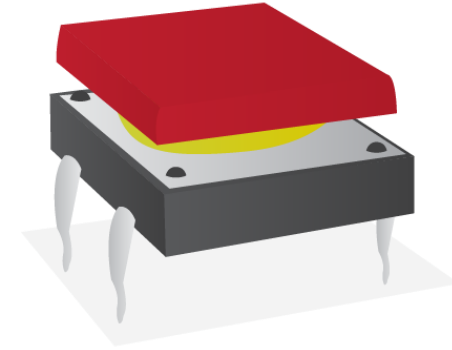
Eje X:
Tiempo

¿Cómo proporciona la salida el sensor?



SALIDA DIGITAL

```
Encender: GPIO.output(18, GPIO.HIGH) # Turn on  
Apagar:  GPIO.output(18, GPIO.LOW)  # Turn off
```



ENTRADA DIGITAL

```
inputValue = GPIO.input(14)  
if (inputValue == True):  
    print("Button press ")
```


¿Cómo proporciona la salida el sensor?

La Rpi es un Sistema embebido “Digital-only”

*¿Como Podemos leer datos de sensors **analógicos**?*

¿Como Podemos proporcionar una salida, por ejemplo para controlar un motor?

¿Cómo proporciona la salida el sensor?

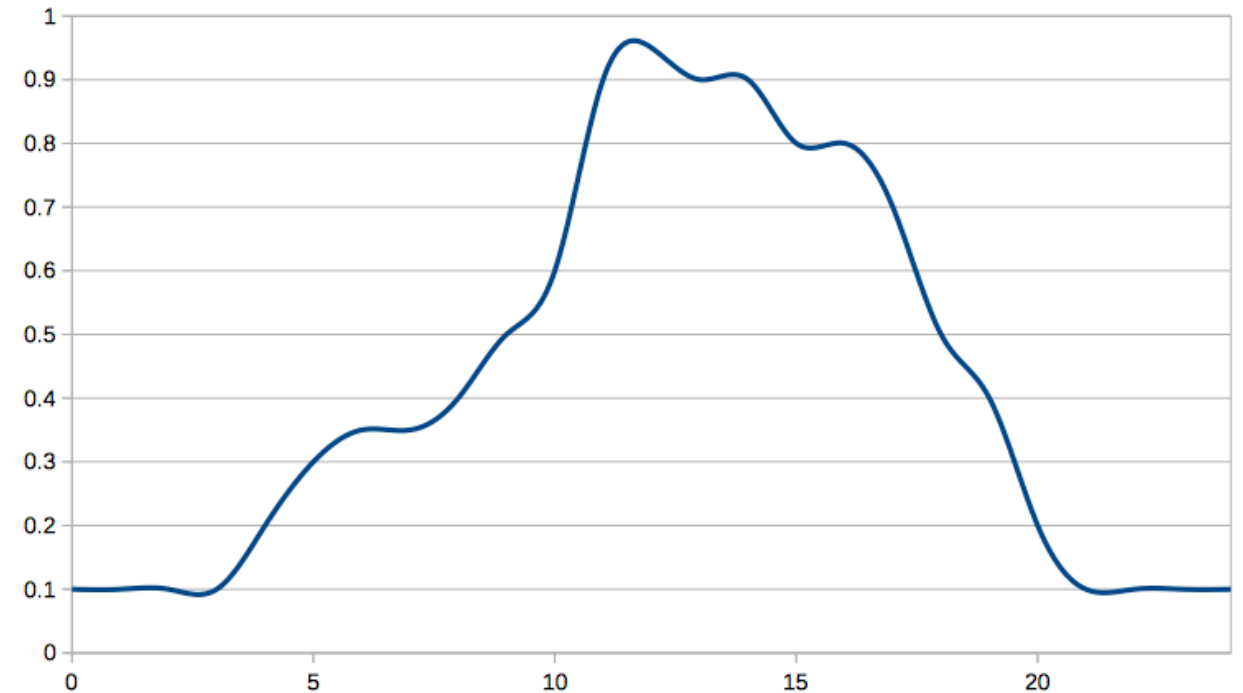
Analógico

- Señal analógica (un voltage o tension que es proporcional a la magnitud que se está midiendo, como la temperatura, o el nivel de luz)

Ejemplo de de una entrada analógica de de un Light Dependent Resistor (LDR) ó Resistor dependiente de intensidad luminosa

Cuando no hay luz en el componente, enviará un 0.
Conforme la luz incrementa el LDR incrementará el valor que enviará hasta alcanzar el valor máximo de 1.

Eje Y:
Valor que enviará el sensor

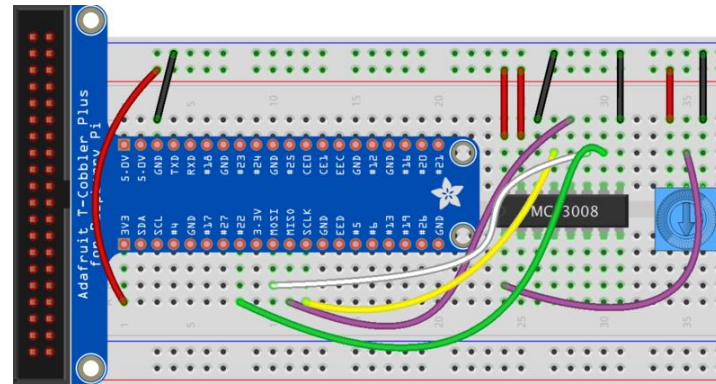


Eje X:
Tiempo

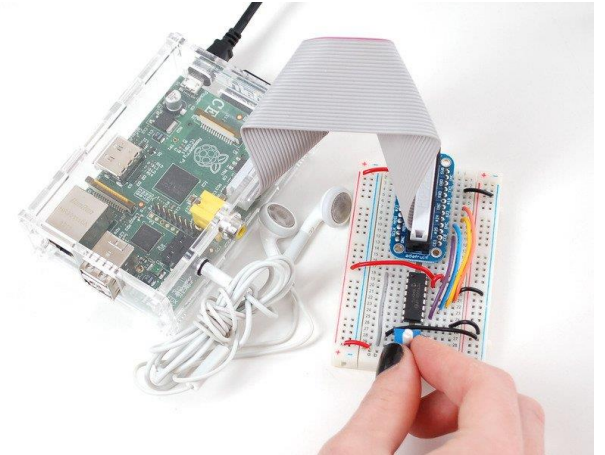
¿Cómo proporciona la salida el sensor?

Analógico

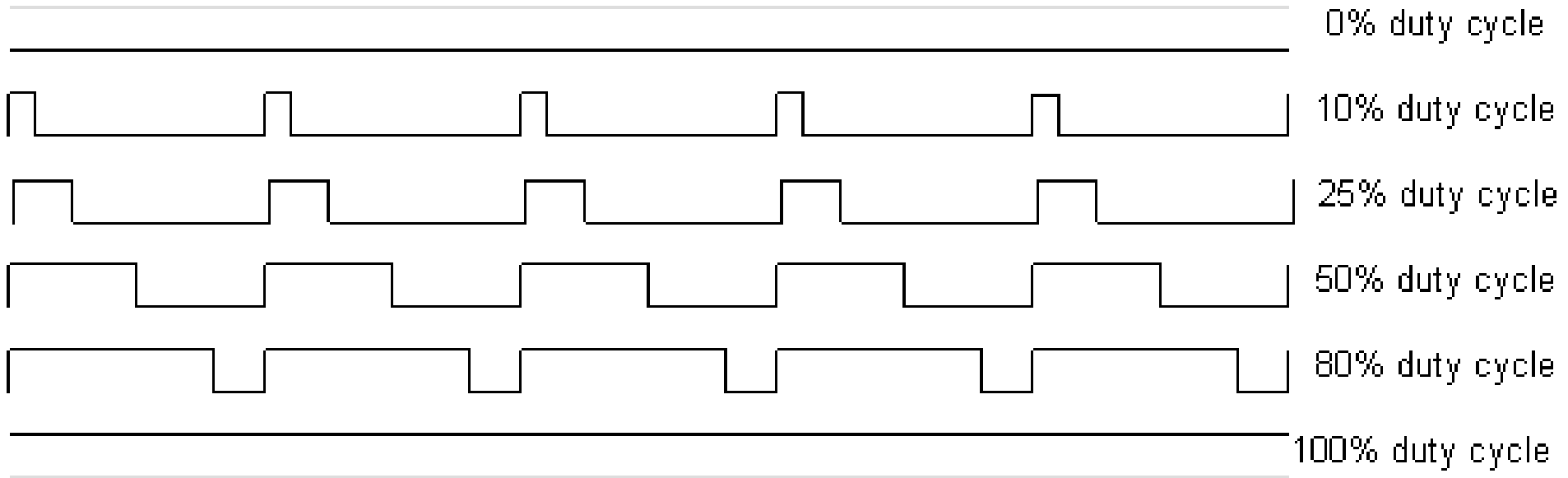
- La Rpi no conversores DAC (digital a analógico) ni ADC (Analógico a digital)
- Solución para Salida Analógica: usar PWD (ver siguientes diapositivas)
- Solución para Entrada analógica: usar un conversor analógico-digital. Ejemplo: imagen abajo



[MCP3008](#)



PWM



PWM es una senal digital cuadrada, donde la **frecuencia es constante**, pero el porcentaje de tiempo que esta en valor alto y bajo (duty cycle) se puede variar en 0 y 100%

PWM - Ejemplo

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT, initial=GPIO.LOW) # LED GPIO 18

pin_pwm = GPIO.PWM(18, 200) #configurado a 200Hz

pwm_value = 0
pin_pwm.start(pwm_value)

n=10
for i in range(1,n):
    pin_pwm.ChangeDutyCycle(10*i)
    sleep(1)

for i in range(n,0,-1):
    pin_pwm.ChangeDutyCycle(10*i)
    sleep(1)

pin_pwm.stop()
GPIO.cleanup()
```

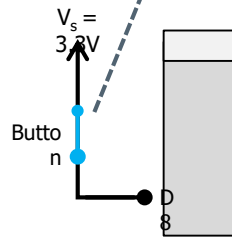
¿Cómo proporciona la salida el sensor?

RESUMEN

- *Digital*
- *Analógico (uso de PWM)*
- *En serie*
- *Protocolo comunicación Sincrono (I2C y SPI)*
- *Protocolo comunicación asíncrono (UART)*

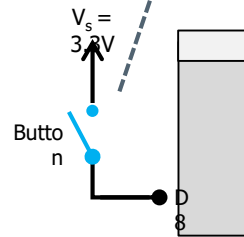
Hooking up a button: Pull-down resistors

When the this button is closed (as it is below), what would the **microcontroller** (MCU) **read** from the **input pin**?



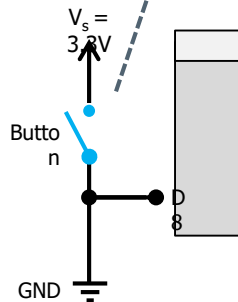
Answer: The MCU would read 3.3V (or HIGH)

Now, when the **button switches to open**, what would the MCU read?



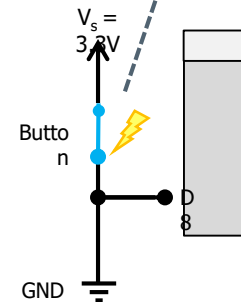
Answer: Uh, oh, the input pin is in an unknown state (commonly called "floating")—this is bad!

Well, can't we just solve that by **adding GND** here that "pulls" D8 to 0V when the switch is open?



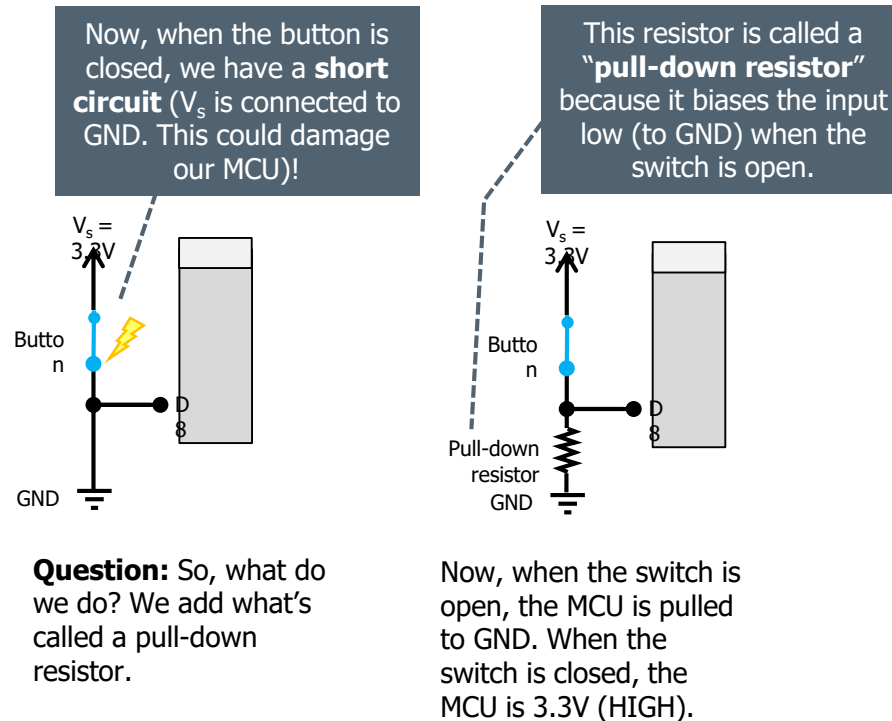
Answer: You're on the right track but this will create a short circuit when the button is closed!

Now, when the button is closed, we have a **short circuit** (V_s is connected to GND). This could damage our MCU!



Question: So, what do we do? We add what's called a pull-down resistor.

Hooking up a button: Pull-down resistors

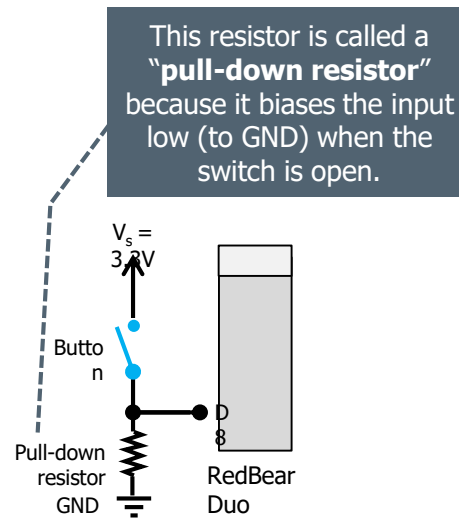


How do you choose the resistor value for the pull-down?

If you choose a low resistor value, more current will be "wasted" by flowing from V_s to GND when the button is closed. In contrast, a high resistor value (e.g., $4\text{M}\Omega$) may not work as a pull-down (not enough current will flow).

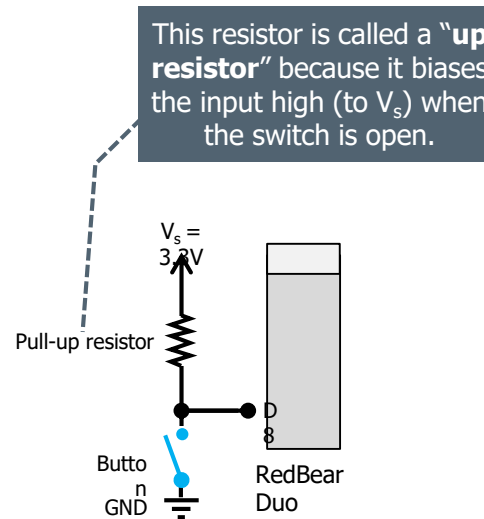
Arduino recommends using a $10\text{K}\ \Omega$ resistor.

Pull-up vs. pull-down Resistors



Pull-Down Resistor Configuration

When the switch is open, the MCU is pulled to GND. When the switch is closed, the MCU is 3.3V (HIGH) as it becomes directly connected to V_s .



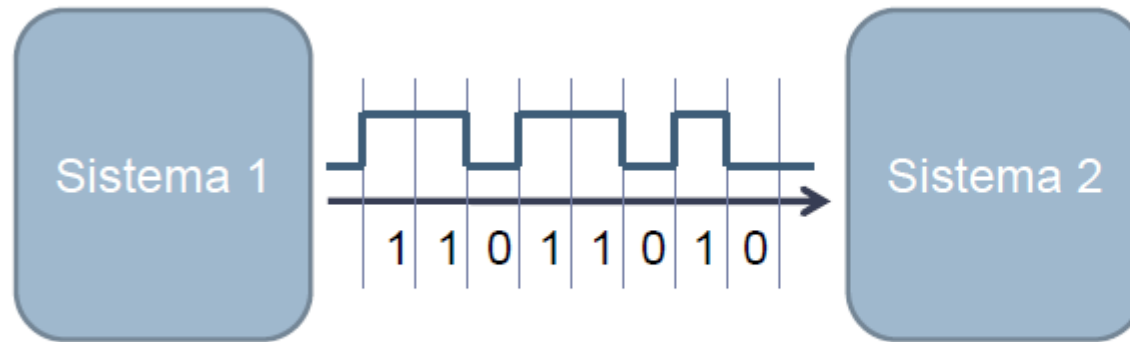
Pull-Up Resistor Configuration

When the switch is open, the MCU is pulled to V_s . When the switch is closed, the MCU is 0V (LOW) as it becomes directly connected to GND.

Both pull-down resistor configurations and pull-up resistor configurations achieve the same goal.

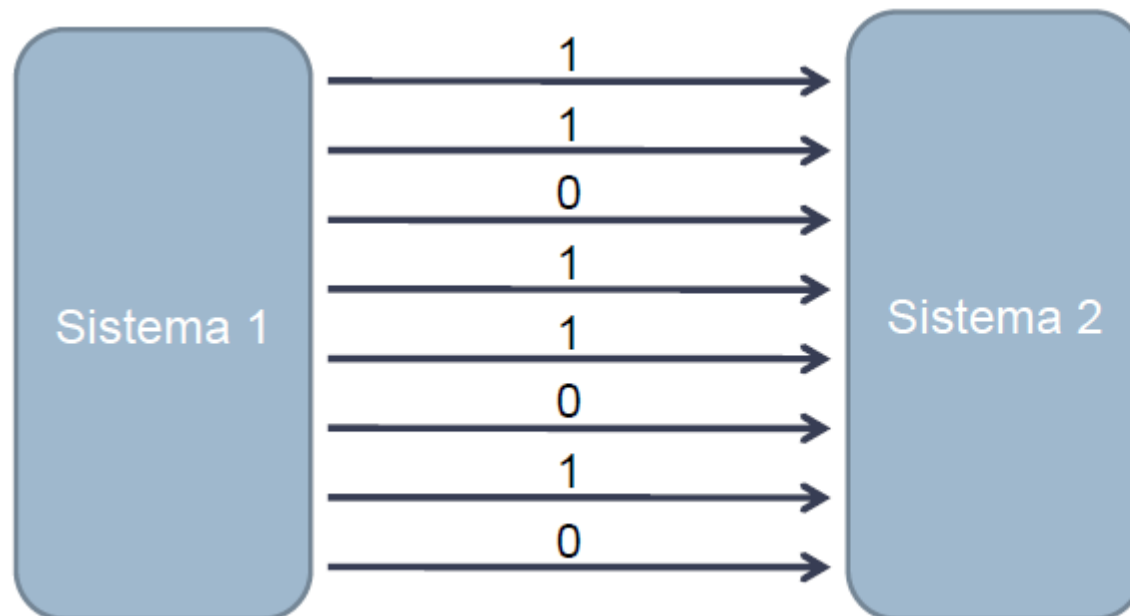
Pull-down resistors tend to make more sense because the switch is 0V (LOW) when open and 3.3 (HIGH) when closed.

Comunicación Serie vs Paralela



➤ Serie:

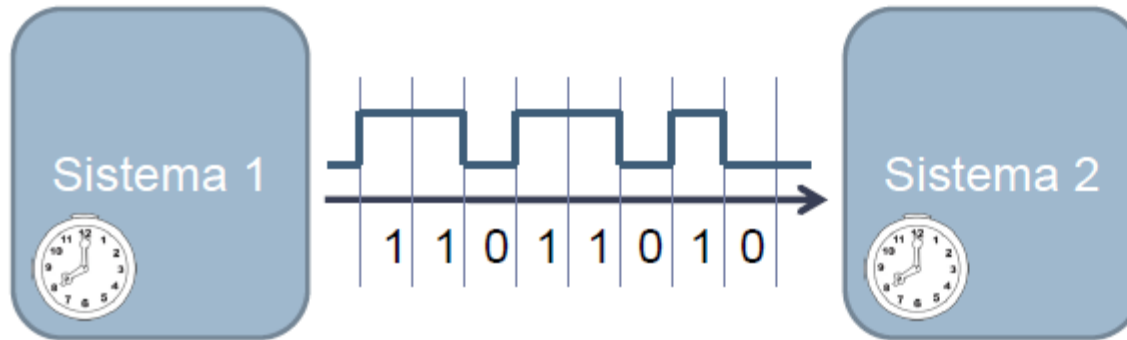
- Los bits se transmiten en serie, uno detrás de otro.
- Sólo se necesita un hilo.
- Más lento.



➤ Paralelo:

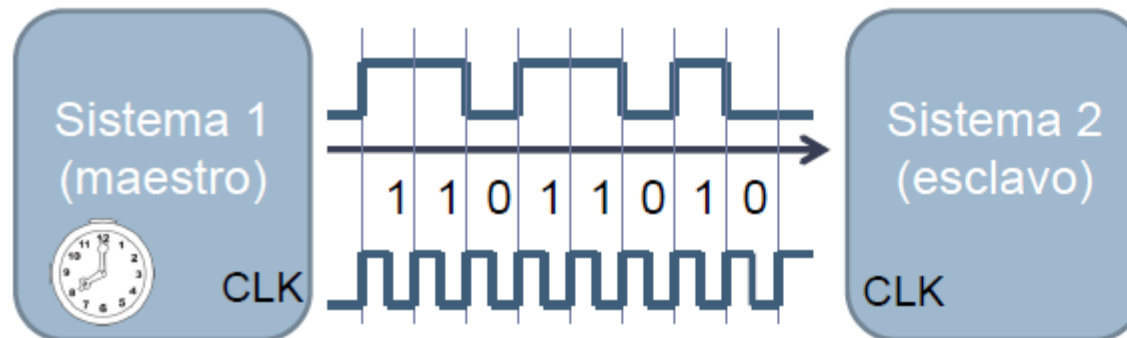
- Todos los bits se transmiten a la vez.
- Necesarios 8 hilos.
- Más rápido.

Comunicación Síncrona vs Asíncrona



➤ Asíncrona:

- Cada sistema tiene su propio reloj local.
- Sincronización mediante bit de Start y Stop.
- Sólo 1 hilo.



➤ Síncrona:

- Una señal de reloj común.
- El *maestro* genera el reloj.
- Dos hilos.
- Velocidades mayores.

Full Duplex vs Half-Duplex



- Full-duplex
 - Comunicación bidireccional simultanea.
 - Dos canales de datos (TX, RX).
 - Dos hilos.



- Half-duplex
 - Comunicación bidireccional multiplexada en el tiempo.
 - Un único canal (DATA).
 - Primero en un sentido, luego en el otro (protocolo).
 - Un hilo