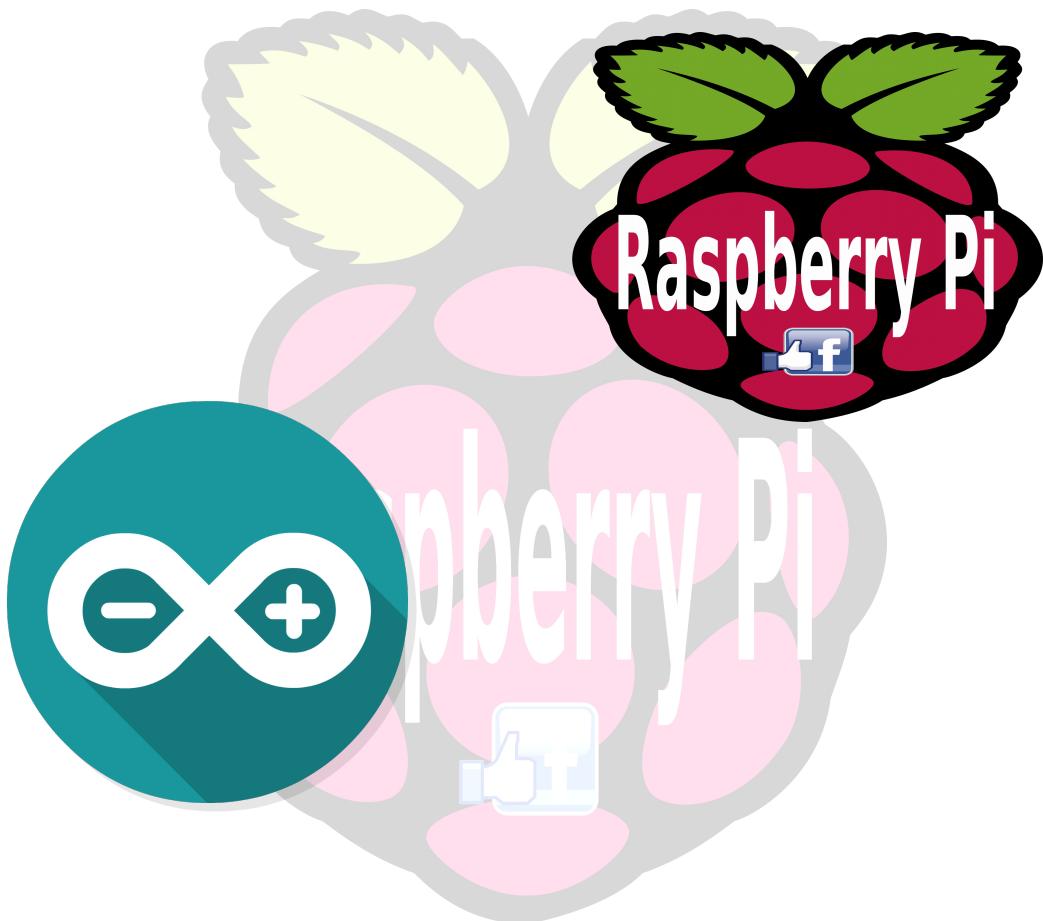


**Luciano Rabassa**

**Tutoriales sobre módulos  
para  
Arduino y Raspberry Pi**



**por**

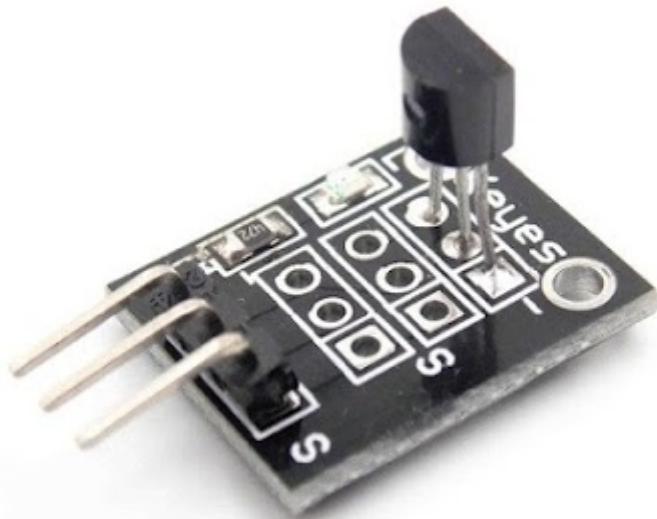
**Luciano Rabassa**

**Raspberry Pi Buenos Aires**



## Módulos Arduino y Raspberry Pi

### KY-001: Módulo sensor de temperatura DS18B20



#### Descripción:

Módulo sensor de temperatura entre 55°C bajo cero y 125°C

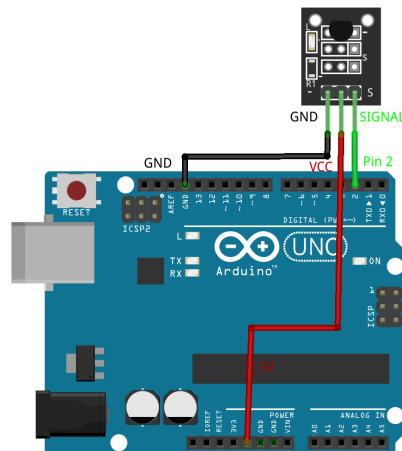
- **Rango de Voltaje:** 3.0 V ~ 5.5 V
- **Rango de Temperatura:** -55°C ~ +125°C
- **Rango de Temperatura:** 67°F ~ 257°F
- **Rango de Precisión:** ±0.5°C

Hoja de datos: <https://cdn.sparkfun.com/datasheets/Sensors/Temperature/DS18B20.pdf>

# Luciano Rabassa

## Conección Arduino:

KY-001: Módulo sensor de temperatura DS18B20



fritzing

KY-001	a	Arduino
-	a	Pin GND
VCC	a	Pin 5V
Signal	a	Pin 2

## Código Arduino:

Requisito librería OneWire: <https://github.com/PaulStoffregen/OneWire.git>

```
#include <OneWire.h>

// DS18S20 Temperatura chip i/o
OneWire ds(2); // Inicializamos el Pin 2

void setup(void) {
    // Inicializamos el puerto serie en 9600 baudios
    Serial.begin(9600);
}

void loop(void) {
    // Creamos las variables para convertir a grados centigrados
    int HighByte, LowByte, TReading, SignBit, Tc_100, Whole, Fract;

    byte i;
    byte present = 0;
    byte data[12];
    byte addr[8];

    if(!ds.search(addr)){
        Serial.print("Sin mas direcciones.\n");
        ds.reset_search();
        return;
    }

    Serial.print("R=");

    for(i = 0; i < 8; i++){
        Serial.print(addr[i], HEX);
        Serial.print(" ");
    }

    if(OneWire::crc8( addr, 7) != addr[7]){
        Serial.print("CRC invalido!\n");
        return;
    }

    if(addr[0] == 0x10){
        Serial.print("Es un dispositivo de la familia DS18S20.\n");
    } else if(addr[0] == 0x28){
        Serial.print("Es un dispositivo de la familia DS18S20.\n");
    } else {
        Serial.print("Dispositivo desconocido: 0x");
        Serial.println(addr[0],HEX);
        return;
    }
}
```

# Luciano Rabassa

```
ds.reset();
ds.select(addr);
ds.write(0x44,1);           // Comienza la conversión

delay(1000);

present = ds.reset();
ds.select(addr);
ds.write(0xBE);

Serial.print("P=");
Serial.print(present,HEX);
Serial.print(" ");

for(i = 0; i < 9; i++){    // Necesitamos leer 9 bytes
  data[i] = ds.read();
  Serial.print(data[i], HEX);
  Serial.print(" ");
}
Serial.print(" CRC=");
Serial.print(OneWire::crc8( data, 8), HEX);
Serial.println();

//Conversión de datos a grados centigrados
LowByte = data[0];
HighByte = data[1];
TReading = (HighByte << 8) + LowByte;
SignBit = TReading & 0x8000;

if(SignBit){
  TReading = (TReading ^ 0xffff) + 1;
}
Tc_100 = (6 * TReading) + TReading / 4;
Whole = Tc_100 / 100;
Fract = Tc_100 % 100;

if(SignBit) {
  Serial.print("-");
}
Serial.print(Whole);
Serial.print(".");
if(Fract < 10){
  Serial.print("0");
}
Serial.print(Fract);

Serial.print("\n");
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

Para activar la comunicación entre la Raspberry Pi y el sensor DS18B20 debemos agregar información al final del archivo `config.txt`.

Abrimos Terminal y tecleamos:

- `sudo nano /boot/config.txt`

y al final del archivo agregamos la siguiente línea:

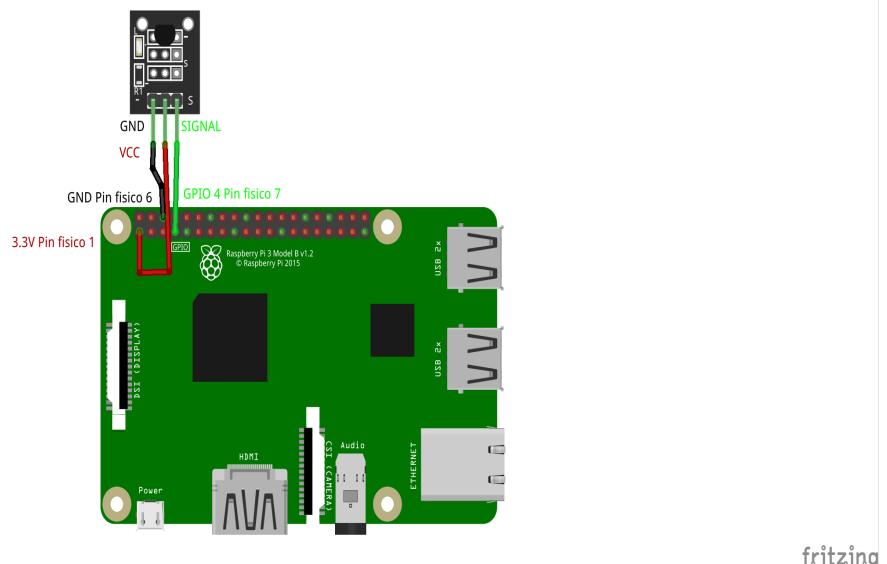
- `dtoverlay=w1-gpio,gpio=4`

Presionamos **Ctrl + x**, luego **y**, finalmente **enter**. Para guardar los cambios.

Reiniciamos con:

- `sudo reboot`

KY-001: Módulo sensor de temperatura DS18B20



fritzing

KY-001	a	Raspberry Pi
GND	a	Pin 6 GND
VCC	a	Pin 1 3.3V
Signal	a	Pin 7 GPIO4

## Código Raspberry Pi:

```
# coding=utf-8
# Importamos los módulos necesarios
import glob
import time
from time import sleep
import RPi.GPIO as GPIO

# tiempo entre mediciones
sleeptime = 1

# Habilitamos la resistencia Pull UP
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN, pull_up_down = GPIO.PUD_UP)

# Luego de habilitada la resistencia Pull UP aguardamos que inicie el sensor
print("Aguarda que inicie el sensor...")

base_dir = '/sys/bus/w1/devices/'

while True:
    try:
        device_folder = glob.glob(base_dir + '28*')[0]
        break
    except IndexError:
        sleep(0.5)
        continue

device_file = device_folder + '/w1_slave'

# Definimos la función de medición de temperatura.
def Medicion():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

Medicion()

def evaluarMedicion():
    lines = Medicion()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = Medicion()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
    return temp_c
```

# Luciano Rabassa

```
# loop
try:
    while True:
        print("-----")
        print("Temperatura:", evaluarMedicion(), "°C")
        time.sleep(sleepTime)

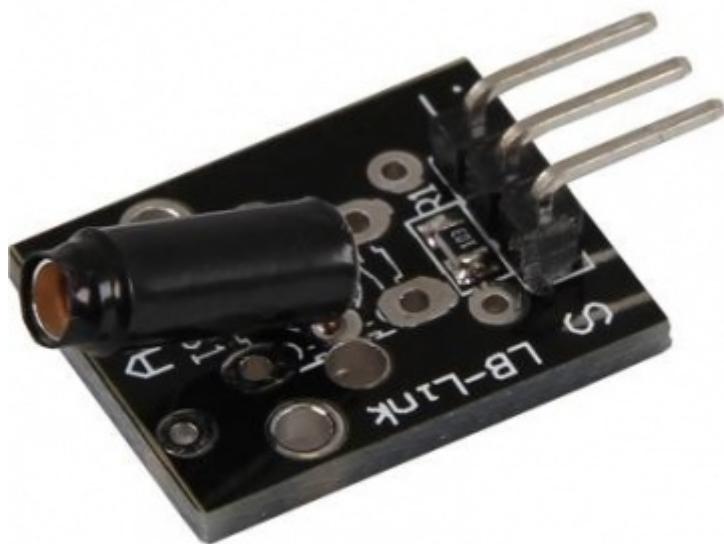
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-001.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-001.py*

## KY-002 : Módulo Sensor de Vibración



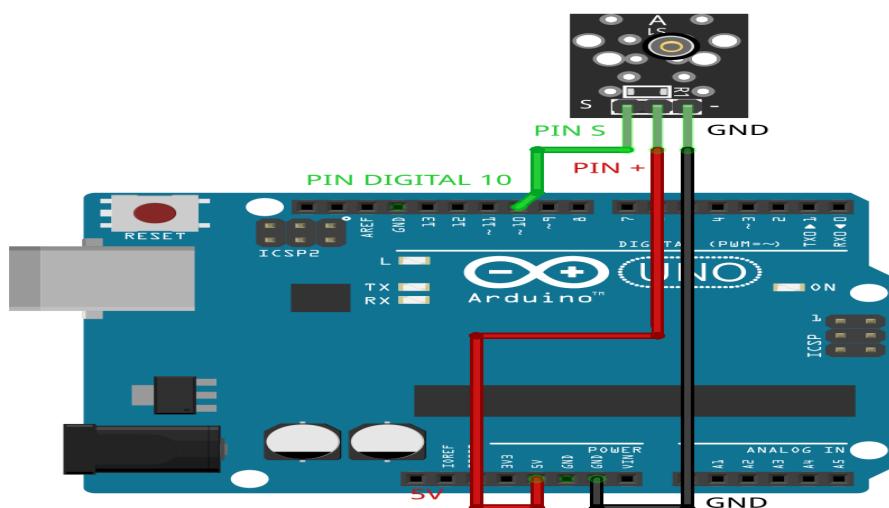
### Descripción:

Este módulo sensor de vibraciones funciona como un interruptor, al detectar una vibración cierra un circuito, lo que genera que el usuario pueda detectar los movimientos.

# Luciano Rabassa

## Conección Arduino:

KY-002 : Módulo Sensor de Vibración



fritzing

KY-002	a	Arduino
Signal	a	Pin Digital 10
VCC	a	Pin 5V
GND	a	Pin GND

## Código Arduino:

```
//Declara la variable entera Pin_Signal y le asigna el Pin Digital 10
int Pin_Signal = 10;
int valor; // Declara la variable entera valor

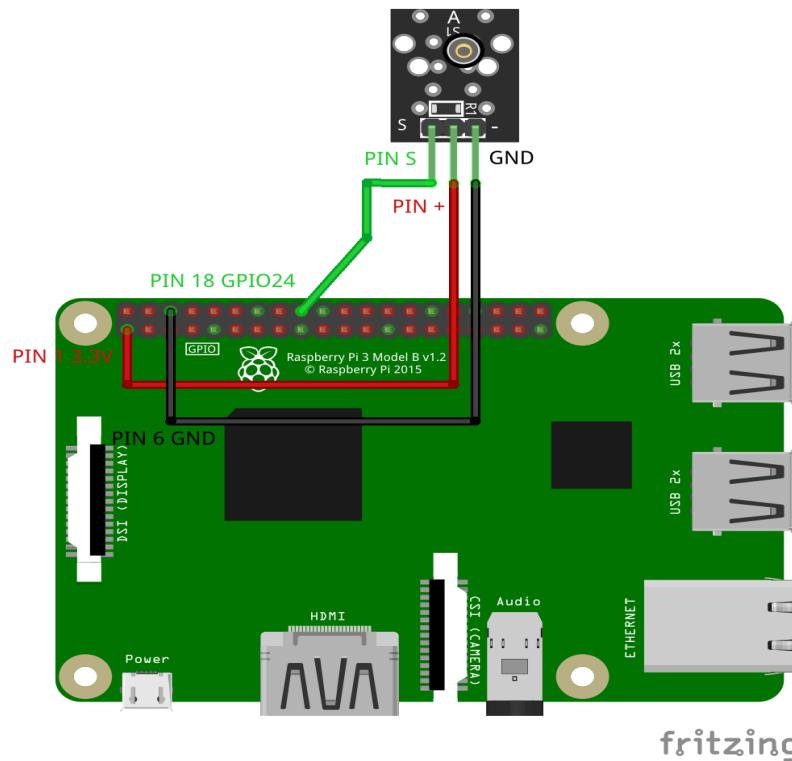
void setup(){
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(Pin_Signal, INPUT);
    digitalWrite(Pin_Signal, HIGH);
}

void loop(){
    // La señal actual del sensor sera leída y asignada a valor
    valor = digitalRead(Pin_Signal);
    if(valor == HIGH){// Si una señal es detectada el led se encenderá
        digitalWrite(LED_BUILTIN, LOW);
    } else {
        digitalWrite(LED_BUILTIN, HIGH);
    }
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

KY-002 : Módulo Sensor de Vibración



fritzing

KY-002	a	Raspberry Pi
Signal	a	Pin 18 GPIO24
VCC	a	Pin 1 3.3V
GND	a	Pin 6 GND

## Código Raspberry Pi:

```
# Se importan los módulos necesarios
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Se declara el de entrada del sensor y se activa la resistencia pull-up.
GPIO_Pin_Signal = 24
GPIO.setup(GPIO_Pin_Signal, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print("Probar Sensor [presiona ctrl+c para terminar la prueba]")

# Esta función sera llamada al detectarse el sensor
def funcionDetectar(null):
    print("Ha sido detectada")

# Al momento de detectar una señal se llamará a la función funcionDetectar y
# se la activara
GPIO.add_event_detect(GPIO_Pin_Signal, GPIO.FALLING, callback =
funcionDetectar, bouncetime = 100)

# loop
try:
    while True:
        time.sleep(1)

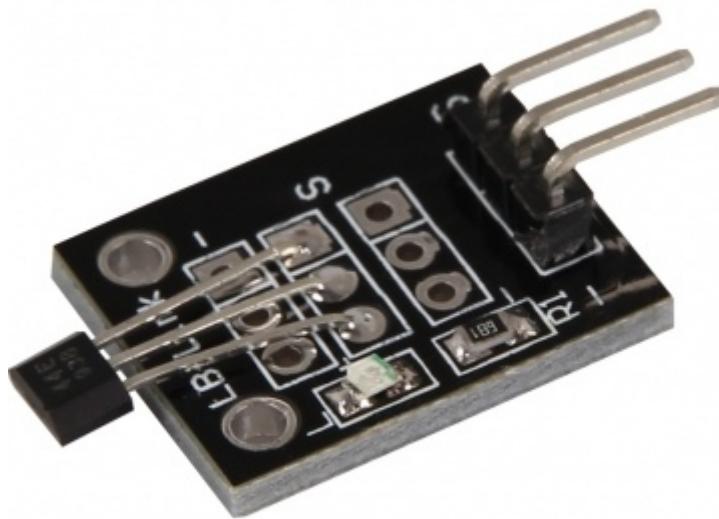
# se limpian los GPIO luego de terminado el programa
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-002.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-002.py*

## KY-003: Módulo sensor magnético efecto Hall



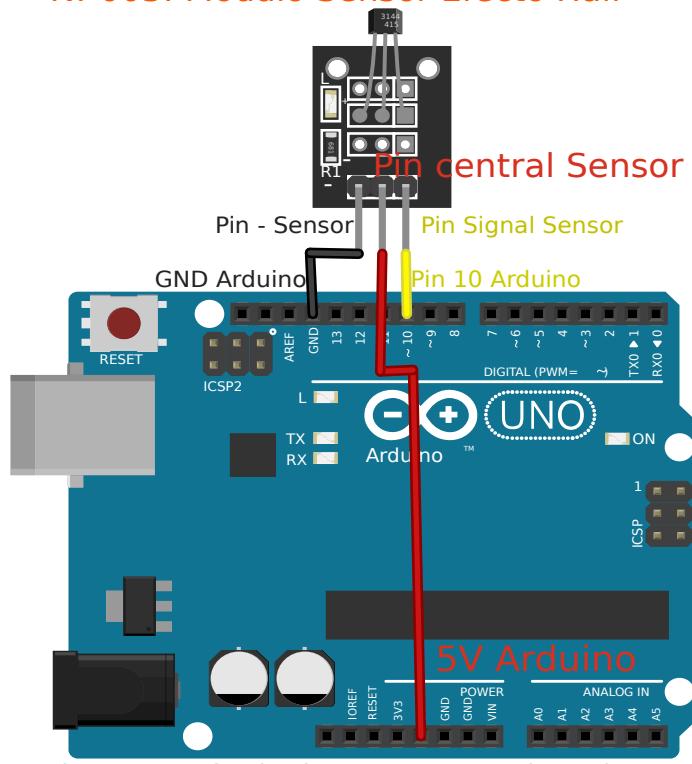
### Descripción:

Este sensor mide un campo magnético. Si existe la presencia de un campo magnético se creará una señal de alto(1 binario).

# Luciano Rabassa

## Conección Arduino:

### KY-003: Módulo Sensor Efecto Hall



<https://www.facebook.com/groups/RaspberryPiBuenosAires>

KY-003	a	Arduino
-	a	Pin GND
VCC	a	Pin 5V
Signal	a	Pin Digital 10

## Código Arduino:

```
int Pin_Signal = 10;
int valor; // Declaramos una variable temporal

void setup()
{
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(Pin_Signal, INPUT);
    digitalWrite(Pin_Signal, HIGH);
}

void loop()
{
```

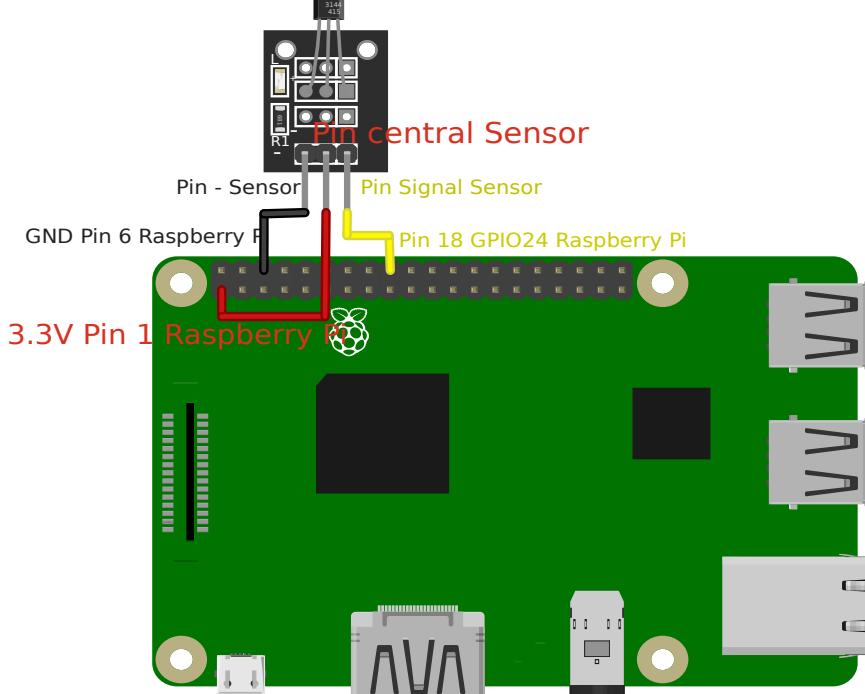
# Luciano Rabassa

```
valor = digitalRead(Pin_Signal);  
  
if(valor == HIGH){  
    digitalWrite(LED_BUILTIN, LOW);  
} else {  
    digitalWrite(LED_BUILTIN, HIGH);  
}  
}
```

## Conexión Raspberry Pi:

KY-003: Módulo Sensor Efecto Hall

<https://www.facebook.com/groups/RaspberryPiBuenosAires>



KY-003	a	Raspberry Pi
-	a	Pin 6 GND
VCC	a	Pin 1 3.3V
Signal	a	Pin 18 GPIO24

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO_Pin_Signal = 24
GPIO.setup(GPIO_Pin_Signal, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print("Prueba del modulo [presiona Ctrl + C para finalizar la prueba]")

def funcionDetectar(null):
    print("Ha sido detectado")

GPIO.add_event_detect(GPIO_Pin_Signal, GPIO.FALLING, callback =
funcionDetectar, bouncetime = 100)

# loop
try:
    while True:
        time.sleep(1)

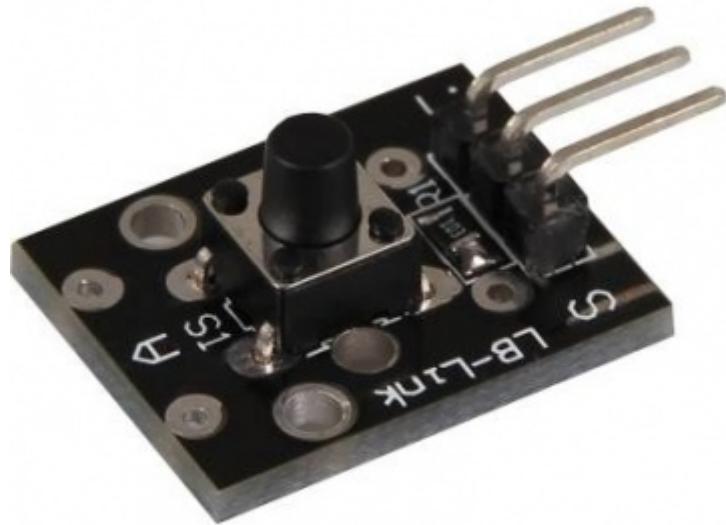
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-003.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-003.py*

## KY-004: Módulo botón interruptor



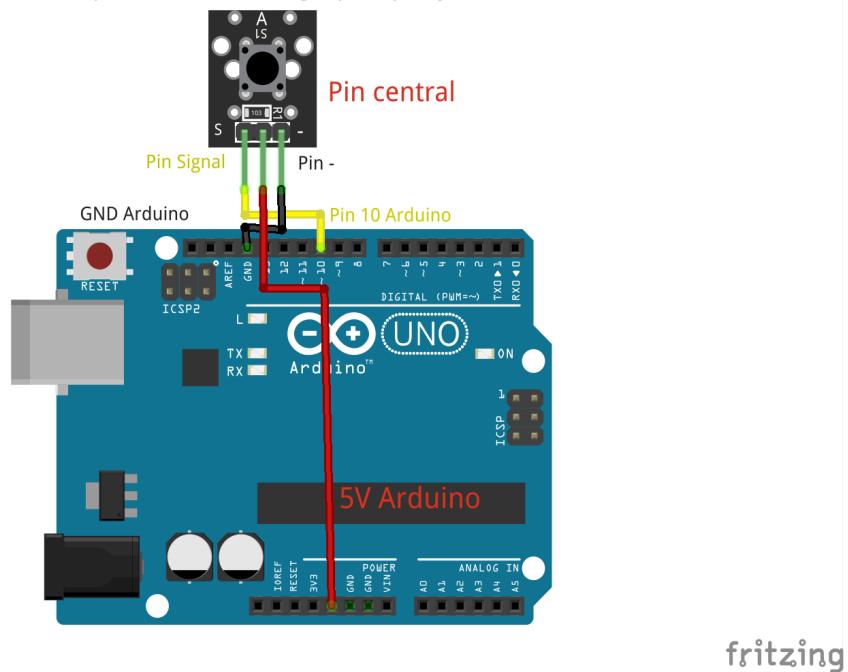
### Descripción:

Este módulo tiene integrado un interruptor común con una resistencia de  $10\text{ k}\Omega$ .

# Luciano Rabassa

## Conección Arduino:

KY-004: Módulo botón interruptor  
<https://www.facebook.com/groups/RaspberryPiBuenosAires>



fritzing

KY-004	a	Arduino
Signal	a	Pin Digital 10
VCC	a	Pin 5V
-	a	Pin GND

## Código Arduino:

```
int Pin_Signal = 10;
int valor;

void setup(){
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(Pin_Signal, INPUT);
  digitalWrite(Pin_Signal, HIGH);
}

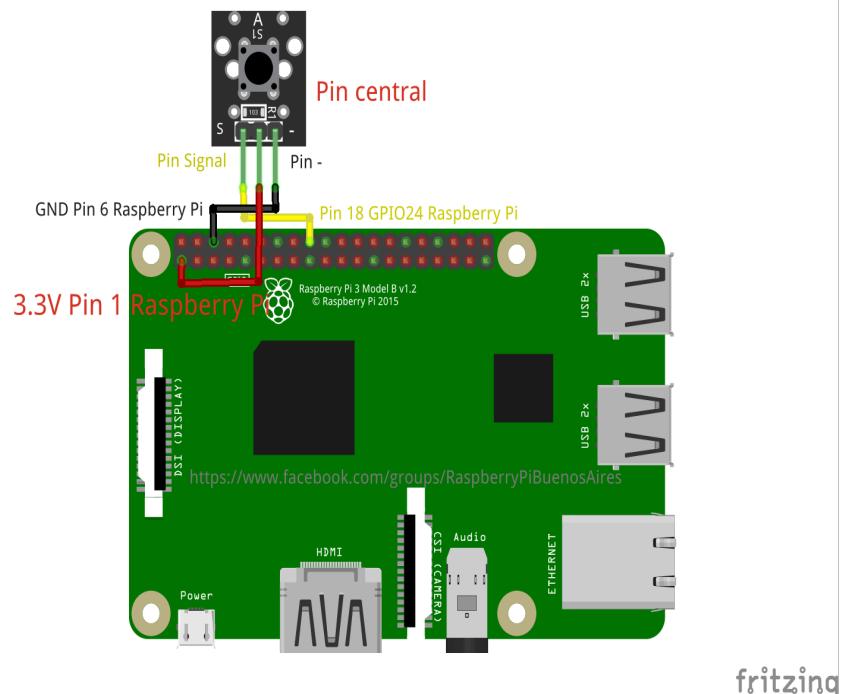
void loop(){
  valor = digitalRead(Pin_Signal);
  if(valor == HIGH)
    digitalWrite(LED_BUILTIN, LOW);
```

# Luciano Rabassa

```
    }
else
{
    digitalWrite(LED_BUILTIN, HIGH);
}
}
```

## Conexión Raspberry Pi:

KY-004: Módulo botón interruptor



KY-004	a	Raspberry Pi
Signal	a	Pin 18 GPIO24
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_Pin_Signal = 24

GPIO.setup(GPIO_Pin_Signal, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print("Prueba del modulo [presiona Ctrl + C para finalizar la prueba]")

def funcionDetectar(null):
    print("Ha sido detectado")

GPIO.add_event_detect(GPIO_Pin_Signal, GPIO.FALLING, callback =
funcionDetectar, bouncetime = 100)

# loop
try:
    while True:
        time.sleep(1)

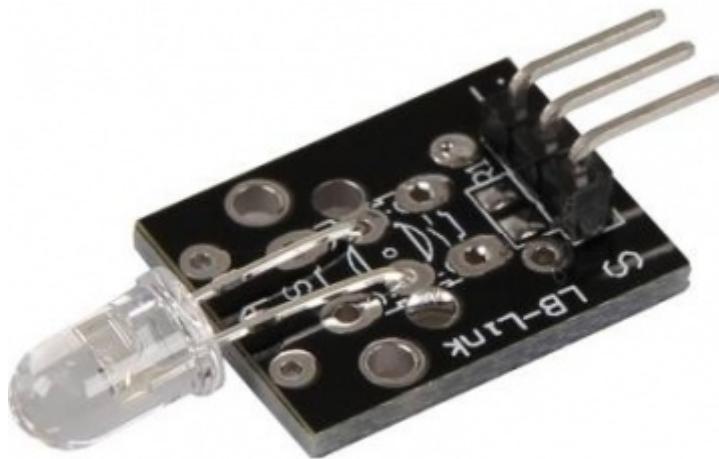
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-004.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-004.py*

## KY-005: Módulo transmisor de infrarrojos



### Descripción:

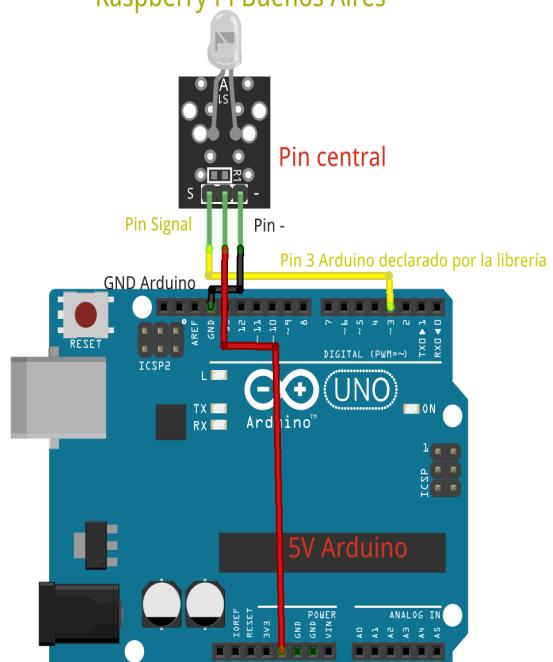
Es un Led de 5 mm que emite luz infrarroja a 38 Khz. Puede venir o no con una resistencia SMD 103 (10 kΩ).

- Tensión de funcionamiento 5V
- Corriente directa 30 ~ 60 mA
- El consumo de energía 90mW
- Temperatura de funcionamiento -25 ° C a 80 ° C [-13 ° F a 176 ° F]
- Dimensiones 18.5 mm x 15 mm [0.728 pulgadas x 0.591 pulgadas]
- Vf= 1,1V
- If= 20mA

# Luciano Rabassa

## Conección Arduino:

KY-005: Módulo transmisor de infrarrojos  
Raspberry Pi Buenos Aires



fritzing

KY-005	a	Arduino
Signal	a	Pin Digital 3
VCC	a	Pin 5V
-	a	Pin GND

## Código Arduino:

```
#include <IRremote.h>
IRsend irsend;

void setup(){
    Serial.begin(9600);
}
void loop(){
    for(int i = 0; i < 50; i++){
        irsend.sendSony(0xa90,12); //Botón power de un TV Sony
        delay(40);
    }
}
```

## Conexión Raspberry Pi:

Hay que instalar LIRC desde el Terminal:

- `sudo apt-get install lirc -y`

Editamos *modules* agregando las siguientes dos líneas al final del archivo:

- `sudo nano /etc/modules`
- `lirc_dev`
- `lirc_rpi gpio_out=17`

Presionamos "Ctrl+x" para cerrar."Y" para guardar. "Enter" para aceptar y terminar de cerrar el archivo.

Editamos *config.txt*:

- `sudo nano /boot/config.txt`

Agregamos la siguiente línea al final del archivo:

- `dtoverlay=lirc-rpi, gpio_out_=17`

Presionamos "Ctrl+x" para cerrar."Y" para guardar. "Enter" para aceptar y terminar de cerrar el archivo.

Editamos *ir-remote.conf*:

- `sudo nano /etc/modprobe.d/ir-remote.conf`

Agregamos la siguiente línea:

- `options lirc_rpi gpio_out_=17`

Presionamos "Ctrl+x" para cerrar."Y" para guardar. "Enter" para aceptar y terminar de cerrar el archivo.

Editamos las siguientes líneas en *lirc\_options.conf*:

- `driver = default`
- `device = /dev/lirc0`

Presionamos "Ctrl+x" para cerrar."Y" para guardar. "Enter" para aceptar y terminar de cerrar el archivo.

Paramos y reiniciamos lirc:

- `sudo /etc/init.d/lircd stop`
- `sudo /etc/init.d/lircd start`

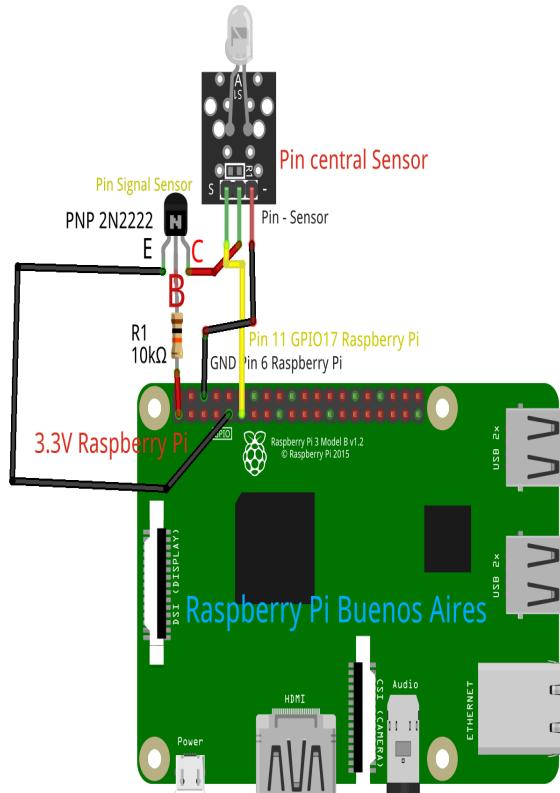
Nos fijamos que Lirc esté corriendo:

- `sudo /etc/init.d/lircd status`

Reiniciamos la Raspberry Pi:

- `sudo reboot`

## KY-005: Módulo Transmisor Infrarrojos



fritzing

Desde el VCC del sensor nos conectamos al Colector del transistor NPN 2N2222, de su base a una resistencia de  $10\text{k}\Omega$  si el módulo no la tuviese(SMD 103) al Pin 1 3.3V, el emisor a GND.

KY-005	a	Raspberry Pi
Signal	a	Pin 11 GPIO17
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND

## Código Raspberry Pi:

Para ver el código funcionando podemos usar la cámara del celular para ver el blink, además del led integrado que posee el módulo.

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)

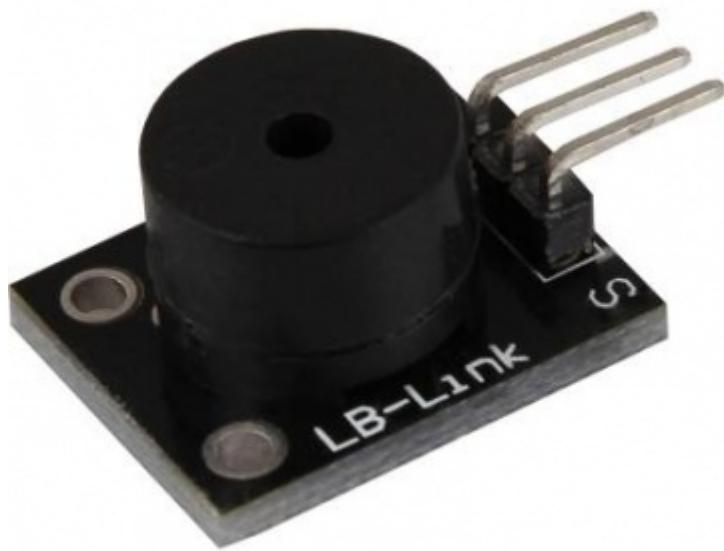
while True:
    GPIO.output(17,True)
    time.sleep(1)
    GPIO.output(17,False)
    time.sleep(1)
```

Guardamos el programa con el nombre KY-005.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-005.py*

## KY-006: Módulo Buzzer-Piezo pasivo



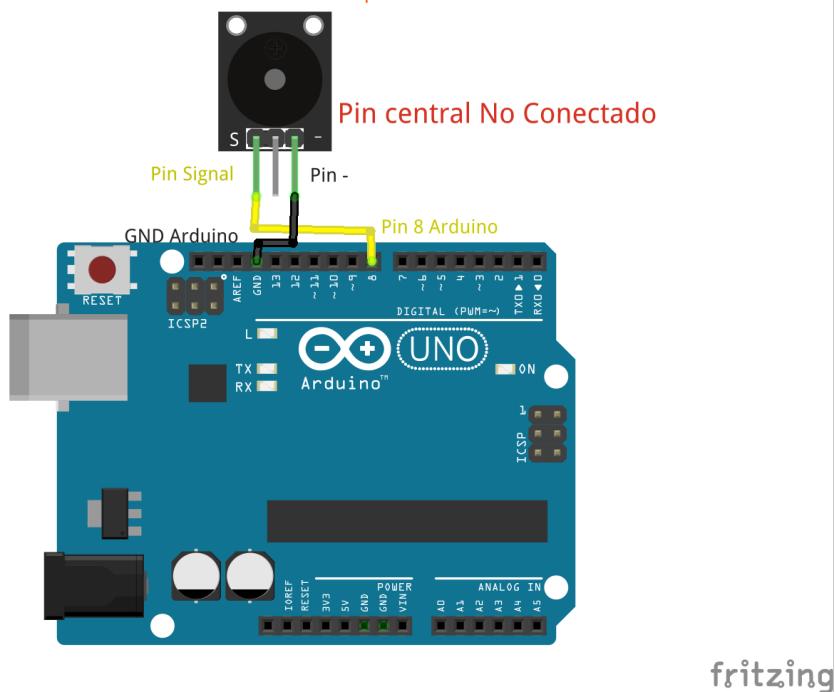
### Descripción:

El buzzer pasivo piezo eléctrico puede generar tonos entre 1.5 y 2.5 KHz, encendiéndolo y apagándolo a diferentes frecuencias utilizando delay o PWM.

# Luciano Rabassa

## Conexión Arduino:

KY-006: Módulo buzzer-Piezo pasivo



fritzing

KY-006	a	Arduino
Signal	a	Pin Digital 8
VCC	a	No se conecta
-	a	Pin GND

## Código Arduino:

```
int Pin_S = 8;

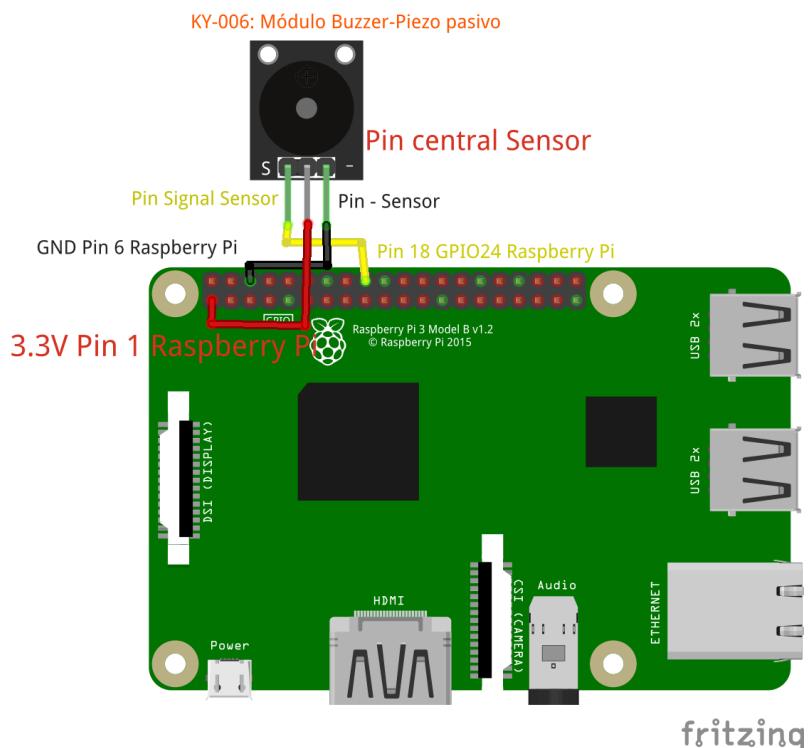
void setup(){
  pinMode(Pin_S, OUTPUT);
}

void loop(){
  unsigned char i;
  while(1){
    //Frecuencia 1
    for(i = 0; i < 80; i++){
      digitalWrite(Pin_S, HIGH);
      delay(1);
      digitalWrite(Pin_S, LOW);
      delay(1);
    }
  }
}
```

# Luciano Rabassa

```
        }
    //Frecuencia 2
    for(i = 0; i < 100; i++)
    {
        digitalWrite(Pin_S, HIGH);
        delay(2);
        digitalWrite(Pin_S, LOW);
        delay(2);
    }
}
```

## Conexión Raspberry Pi:



KY-006	a	Raspberry Pi
Signal	a	Pin 18 GPIO24
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

GPIO_Pin_Signal = 24
GPIO.setup(GPIO_Pin_Signal, GPIO.OUT)
Frecuencia = 500 # En Hertz
pwm = GPIO.PWM(GPIO_Pin_Signal, Frecuencia)
pwm.start(50)

try:
    while True:
        print("-----")
        print("Frecuencia actual: %d" % Frecuencia)
        Frecuencia = input("Por favor entra una nueva frecuencia
entre (50-5000):")
        pwm.ChangeFrequency(Frecuencia)

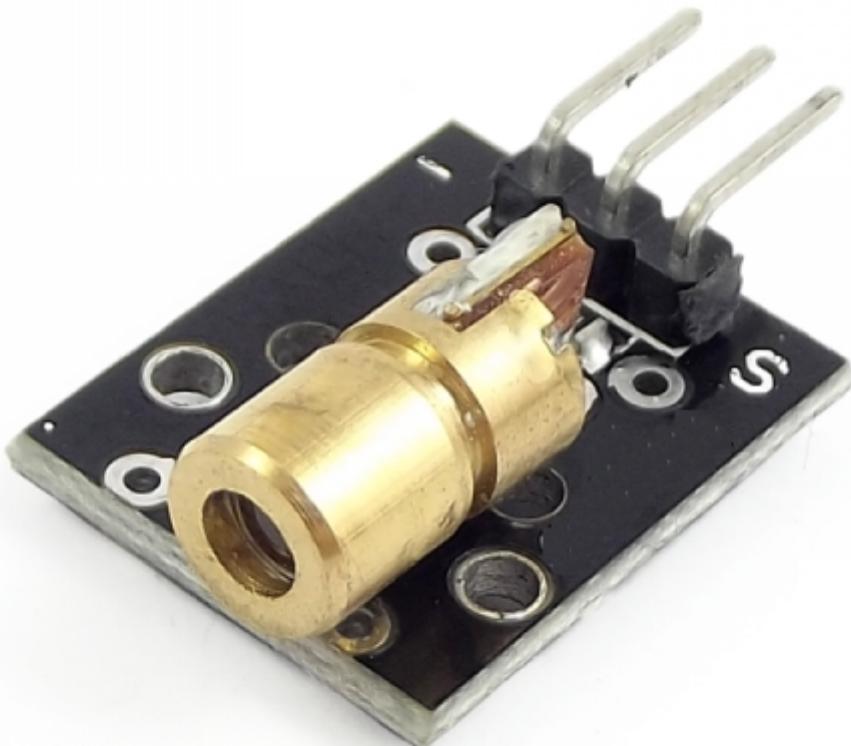
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-006.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-006.py*

## KY-008: Módulo sensor láser

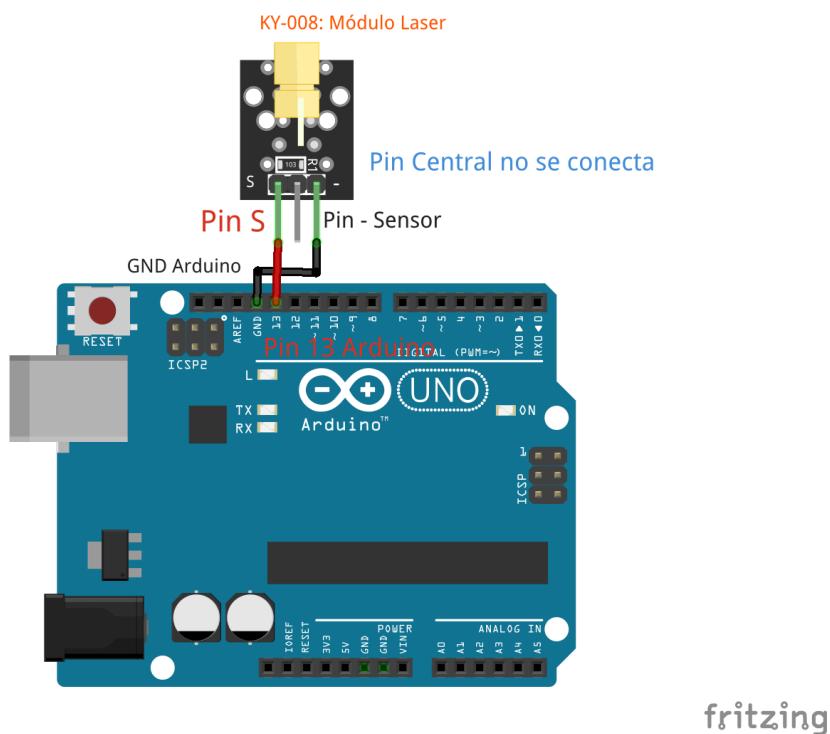


### Descripción:

Este módulo tiene integrado un transmisor láser y una interfaz digital, además cuenta con un LED incorporado. Es el más simple de todos, puede simplemente alimentarse de 3 pilas de 1.5V o 4 de 1.2V. En Raspberry Pi no lleva programación. PRECAUCIÓN no apuntar a los ojos!

# Luciano Rabassa

## Conección Arduino:



KY-008	a	Arduino
Signal	a	Pin Digital 13
VCC	a	No se conecta
-	a	Pin GND

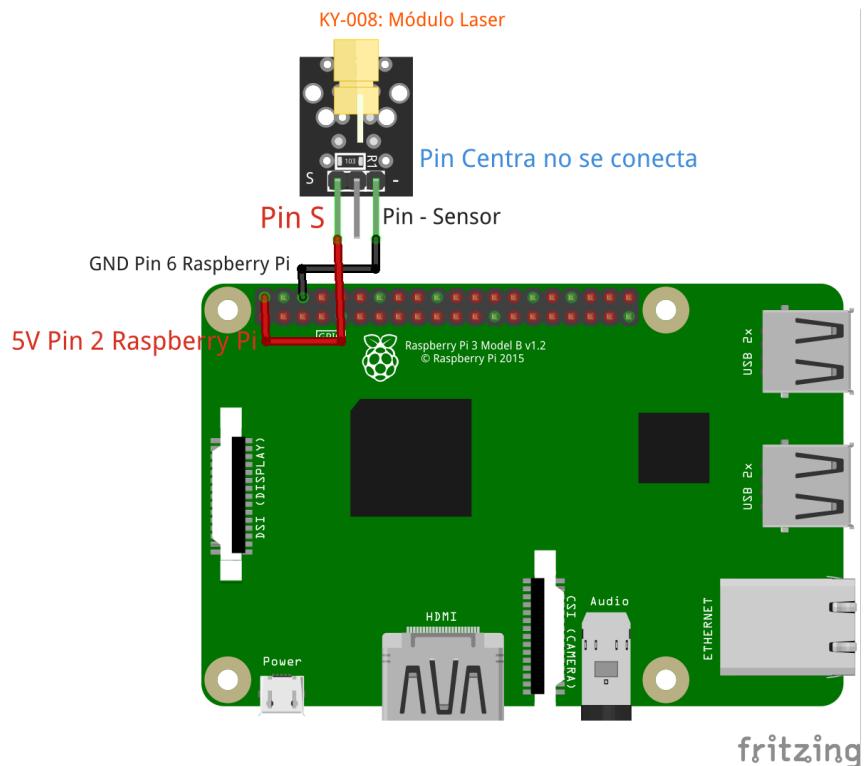
## Código Arduino:

```
int laser = 13;

void setup()
{
  pinMode(laser, OUTPUT);
}

void loop()
{
  digitalWrite(laser, HIGH);
  delay(1000);
  digitalWrite(laser, LOW);
  delay(1000);
}
```

## Conexión Raspberry Pi:

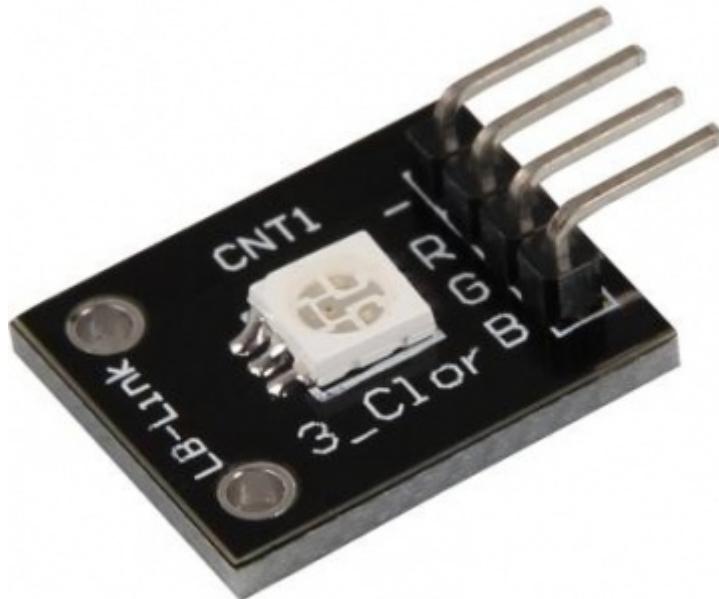


KY-008	a	Raspberry Pi
Signal	a	Pin 2 5V
VCC	a	No se conecta
-	a	Pin 6 GND

## Código Raspberry Pi:

NO SE PROGRAMA!

## KY-009: Módulo LED RGB SMD



### Descripción:

Módulo LED SMD RGB, consiste en un LED capaz de crear cualquier color utilizando RGB, tiene una entrada de voltaje PWM con tres colores. Los colores primarios (rojo / verde / azul) son usados con el fin de lograr el efecto de mezclar colores y obtener cualquier color deseado. Con este módulo se pueden crear efectos de color muy llamativos.

- Corriente máxima en el LED: 20mA
- Voltaje en cada color: Rojo 1.80V (2.4 máx), Verde y Azul 2.8V (3.6V máx)
- Voltaje de operación: 5V
- $V_f$  [Red]= 1,8V
- $V_f$  [Green,Blue]= 2,8V
- $I_f$ = 20mA

### Resistencias Raspberry Pi:

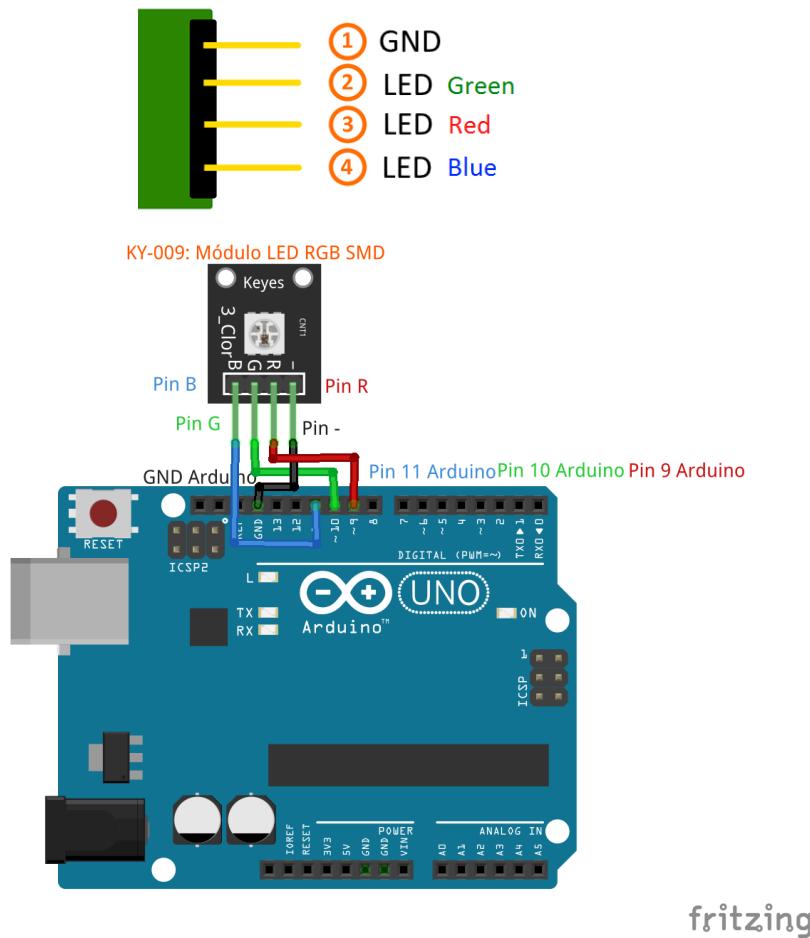
- $R_f$  (3,3V) [Green]= 100Ω
- $R_f$  (3,3V) [Red]= 180Ω
- $R_f$  (3,3V) [Blue]= 100Ω

### Resistencias Arduino:

- $R_f$  (5V) [Green] = 100Ω
- $R_f$  (5V) [Red] = 180Ω
- $R_f$  (5V) [Blue] = 100Ω

# Luciano Rabassa

## Conección Arduino:



KY-009	a	Arduino
B	a	Pin Digital 11
G	a	Pin Digital 10
R	a	Pin Digital 9
-	a	Pin GND

## Código Arduino:

```
int Led_Rojo = 9;
int Led_Verde = 10;
int Led_Azul = 11;
int valor;

void setup() {
    pinMode(Led_Rojo, OUTPUT);
    pinMode(Led_Verde, OUTPUT);
    pinMode(Led_Azul, OUTPUT);
}

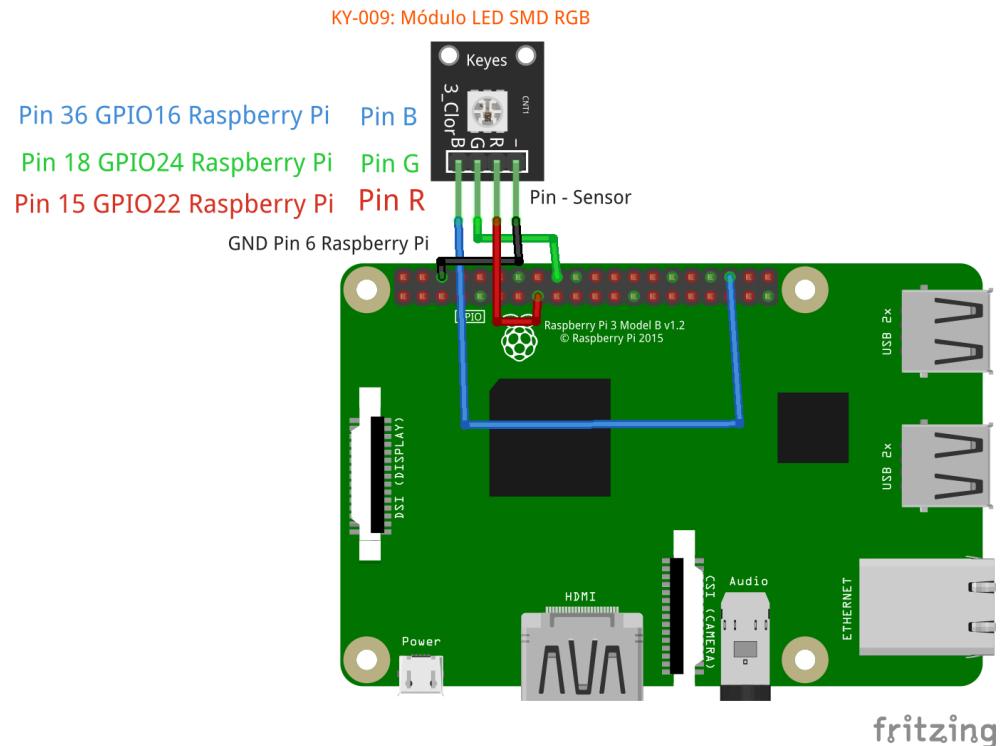
void loop() {

    for(valor = 255; valor > 0; valor--)
    {
        analogWrite(Led_Azul, valor);
        analogWrite(Led_Verde, 255-valor);
        analogWrite(Led_Rojo, 128-valor);
        delay(1);
    }

    for(valor = 0; valor < 255; valor++)
    {
        analogWrite(Led_Azul, valor);
        analogWrite(Led_Verde, 255-valor);
        analogWrite(Led_Rojo, 128-valor);
        delay(1);
    }
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:



KY-009	a	Raspberry Pi
B	a	Pin 36 GPIO16
G	a	Pin 18 GPIO24
R	a	Pin 15 GPIO22
-	a	Pin 6 GND

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

LED_Rojo = 22
LED_Verde = 24
LED_Azul = 16

GPIO.setup(LED_Rojo, GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(LED_Verde, GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(LED_Azul, GPIO.OUT, initial = GPIO.LOW)

print("Prueba de LED [Presiona Ctrl + C para finalizar la prueba]")

# loop
try:
    while True:
        print("LED Rojo se enciende durante 3 segundos")
        GPIO.output(LED_Rojo, GPIO.HIGH)
        GPIO.output(LED_Verde, GPIO.LOW)
        GPIO.output(LED_Azul, GPIO.LOW)
        time.sleep(3)
        print("LED Verde se enciende durante 3 segundos")
        GPIO.output(LED_Rojo, GPIO.LOW)
        GPIO.output(LED_Verde, GPIO.HIGH)
        GPIO.output(LED_Azul, GPIO.LOW)
        time.sleep(3)
        print("LED Azul se enciende durante 3 segundos")
        GPIO.output(LED_Rojo, GPIO.LOW)
        GPIO.output(LED_Verde, GPIO.LOW)
        GPIO.output(LED_Azul, GPIO.HIGH)
        time.sleep(3)

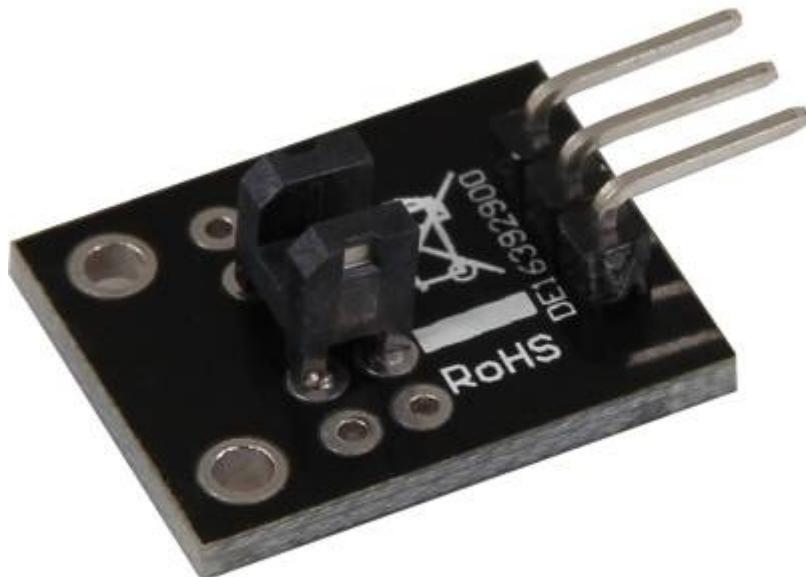
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-009.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-009.py*

## KY-010: Módulo barrera óptica



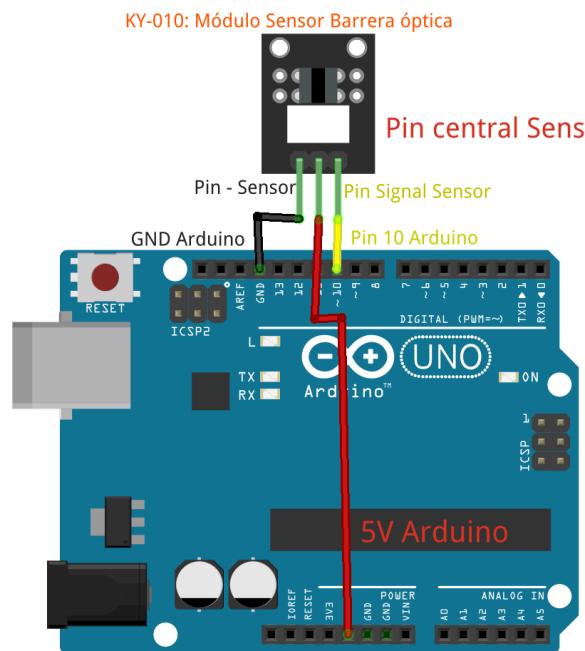
### Descripción:

La conexión entre ambos es de entrada será interrumpida, si la barrera óptica está siendo interrumpida.

- Tensión de trabajo 3.3 ~ 5V

# Luciano Rabassa

## Conección Arduino:



fritzing

KY-010	a	Arduino
-	a	Pin GND
VCC	a	Pin 5V
Signal	a	Pin Digital 10

## Código Arduino:

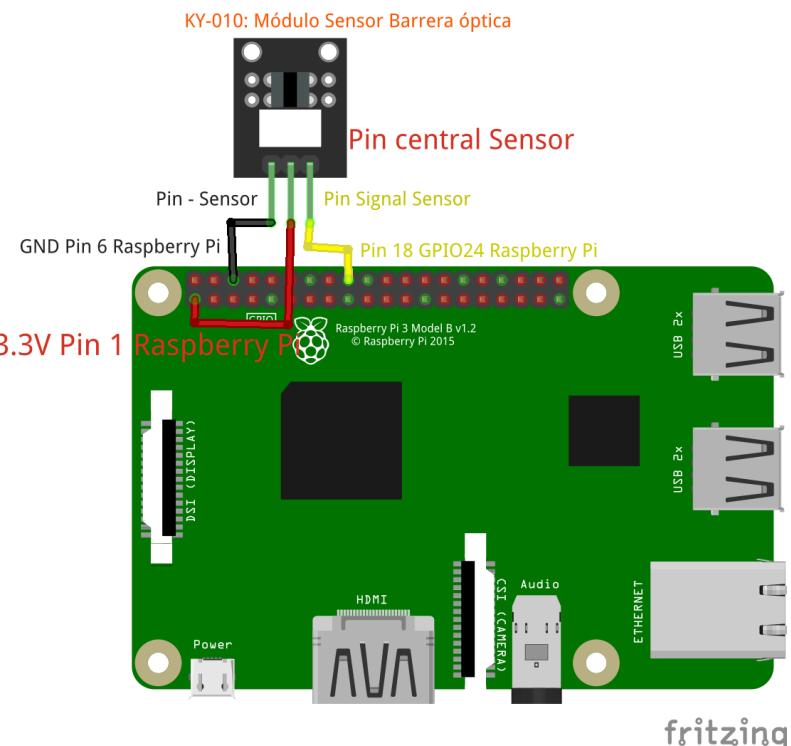
```
int Pin_Signal = 10;
int valor;

void setup(){
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(Pin_Signal, INPUT);
}

void loop(){
  valor = digitalRead(Pin_Signal);
  if(valor == HIGH){
    digitalWrite(LED_BUILTIN, HIGH);
  }else{
    digitalWrite(LED_BUILTIN, LOW);
  }
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:



KY-010	a	Raspberry Pi
GND	a	Pin 6 GND
VCC	a	Pin 1 3.3V
Signal	a	Pin 18 GPIO24

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_Pin_Signal = 24

GPIO.setup(GPIO_Pin_Signal, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

print("Prueba de modulo [Presiona Ctrl + C para finalizar la prueba]")

def funcionSalida(null):
    print("Ha sido detectado")
```

# Luciano Rabassa

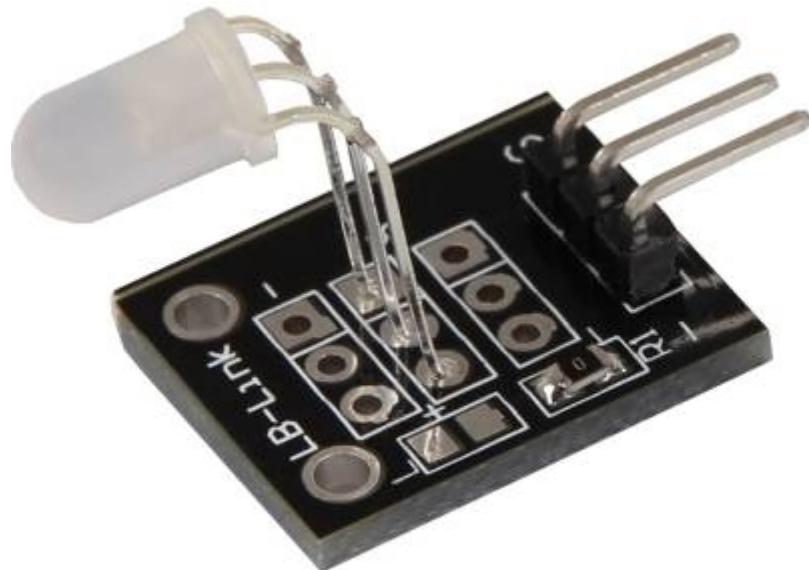
```
GPIO.add_event_detect(GPIO_Pin_Signal, GPIO.RISING, callback =  
funcionSalida, bouncetime = 100)  
  
# loop  
try:  
    while True:  
        time.sleep(1)  
  
except KeyboardInterrupt:  
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-010.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-010.py*

## KY-011: Módulo Led THT 5mm Bi-Color



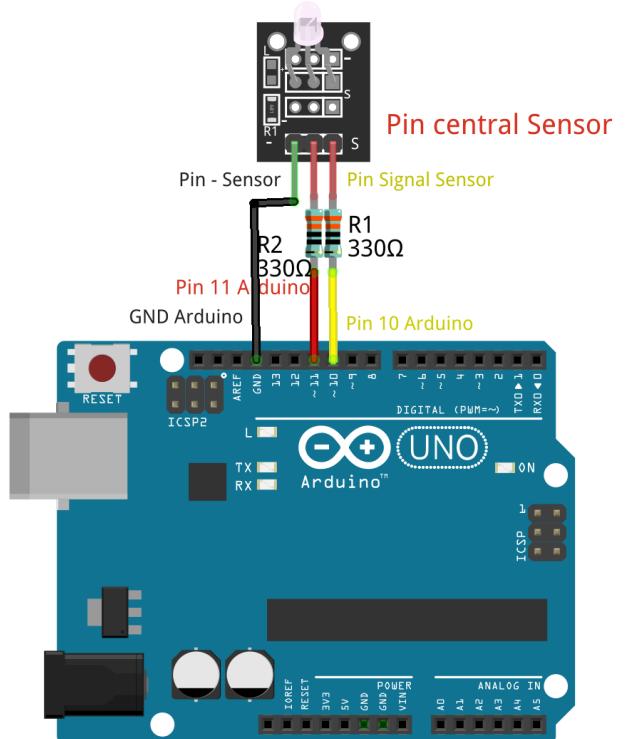
### Descripción:

Posee un led THT de 5mm de dos colores (Rojo-Verde) de Cátodo común, necesitamos agregarle 2 resistencias en serie a cada color, en Arduino de  $330\Omega$  y en Raspberry Pi de  $150\Omega$ . Su Tensión de trabajo ronda entre los 2V-2.5V.

# **Luciano Rabassa**

## Conexión Arduino:

KY-011: Módulo Led Bicolor



fritzing

<b>KY-011</b>	<b>a</b>	<b>Arduino</b>
-	a	Pin GND
VCC	a	Pin 5V + Resistencia 330Ω
Signal	a	Pin Digital 10 + Resistencia 330Ω

## Código Arduino:

```
int Led_Rojo = 11;
int Led_Verde = 10;

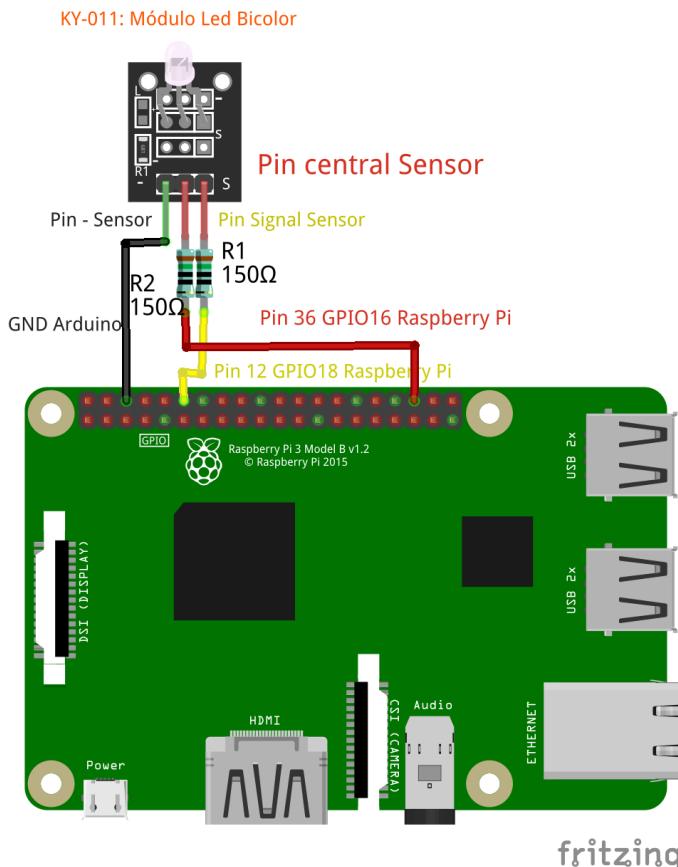
void setup() {
    pinMode(Led_Rojo, OUTPUT);
    pinMode(Led_Verde, OUTPUT);
}

void loop() {
    digitalWrite(Led_Rojo, HIGH);
    digitalWrite(Led_Verde, LOW);
}
```

# Luciano Rabassa

```
delay(3000);  
  
digitalWrite(Led_Rojo, LOW);  
digitalWrite(Led_Verde, HIGH);  
delay(3000);  
}
```

## Conexión Raspberry Pi:



KY-011	a	Raspberry Pi
-	a	Pin 6 GND
VCC	a	Pin 36 GPIO16 + R.150Ω
Signal	a	Pin 12 GPIO18 + R.150Ω

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

LED_Rojo = 16
LED_Verde = 18

GPIO.setup(LED_Rojo, GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(LED_Verde, GPIO.OUT, initial = GPIO.LOW)

print("LED-Test [Presion ctrl+c para cerrar]")

# Main program loop
try:
    while True:
        print("LED Rojo se enciende por 3 segundos")
        GPIO.output(LED_Rojo, GPIO.HIGH)
        GPIO.output(LED_Verde, GPIO.LOW)
        time.sleep(3)
        print("LED Verde se enciende por 3 segundos")
        GPIO.output(LED_Rojo, GPIO.LOW)
        GPIO.output(LED_Verde, GPIO.HIGH)
        time.sleep(3)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-011.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-011.py*

## KY-012: Módulo Buzzer Activo

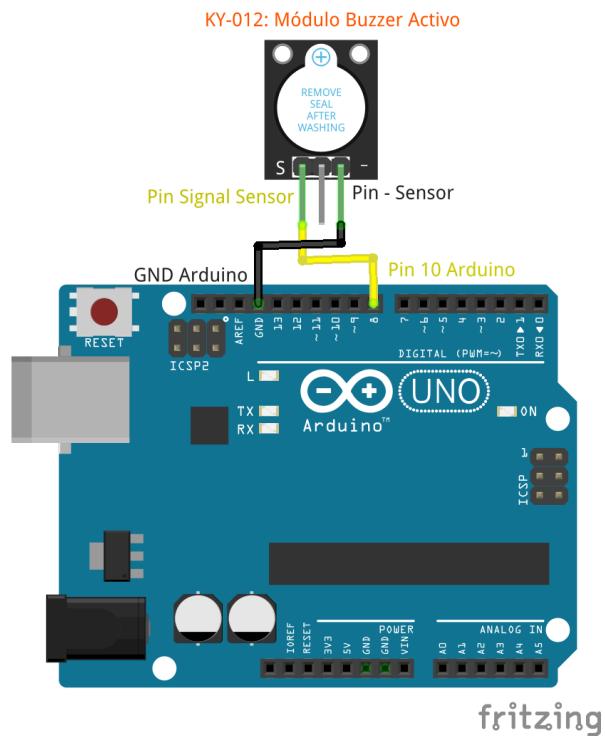


### Descripción:

Este módulo crea un sonido a una frecuencia de 2.5KHz.

# **Luciano Rabassa**

## Conexión Arduino:



<b>KY-012</b>	a	<b>Arduino</b>
<b>Signal</b>	a	<b>Pin Digital 8</b>
<b>VCC</b>	a	<b>No se conecta</b>
-	a	<b>Pin GND</b>

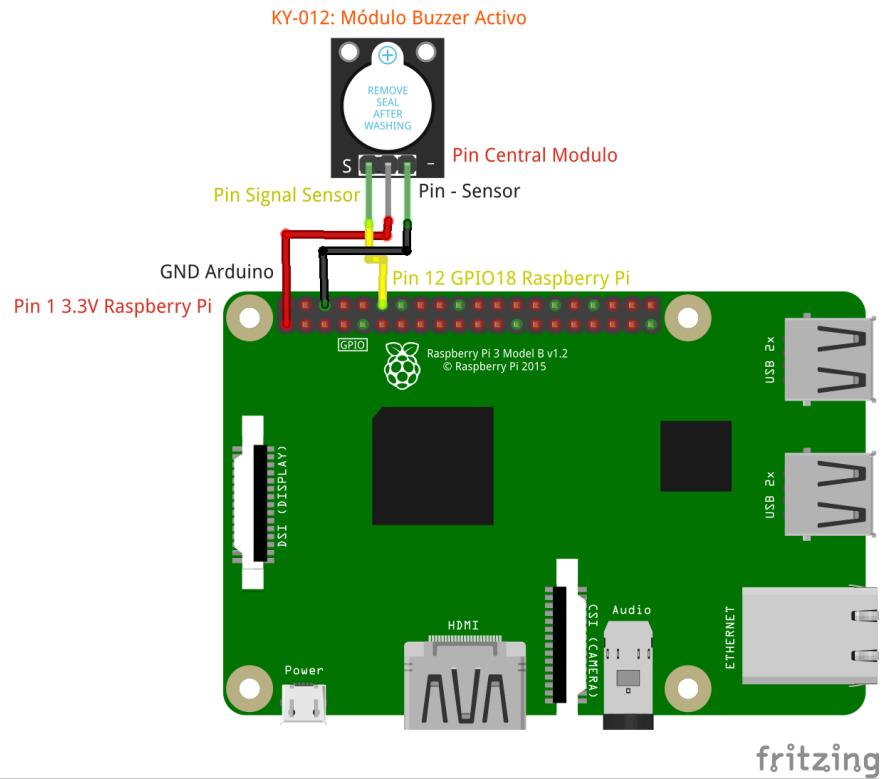
## Código Arduino:

```
int Pin_S = 8;

void setup()
{
    pinMode(Pin_S, OUTPUT);
}

void loop()
{
    digitalWrite(Pin_S, HIGH);
    delay(4000);
    digitalWrite(Pin_S, LOW);
    delay(2000);
}
```

## Conexión Raspberry Pi:



KY-012	a	Raspberry Pi
Signal	a	Pin 12 GPIO18
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
Pin_S = 18

GPIO.setup(Pin_S, GPIO.OUT, initial = GPIO.LOW)

print("Prueba de modulo [Presiona Ctrl+C para finalizar la prueba]")

try:
    while True:
        print("Buzzer sonara por 4 segundos")
        GPIO.output(Pin_S, GPIO.HIGH)
        time.sleep(4)
        print("Buzzer no suena por 4 segundos")
        GPIO.output(Pin_S, GPIO.LOW)
        time.sleep(4)

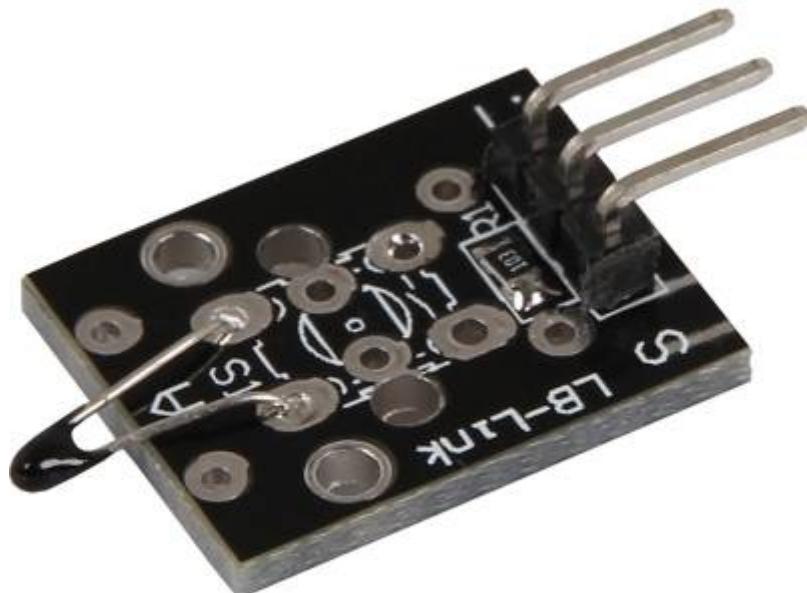
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-012.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-012.py*

## KY-013: Módulo Análogo Sensor de Temperatura

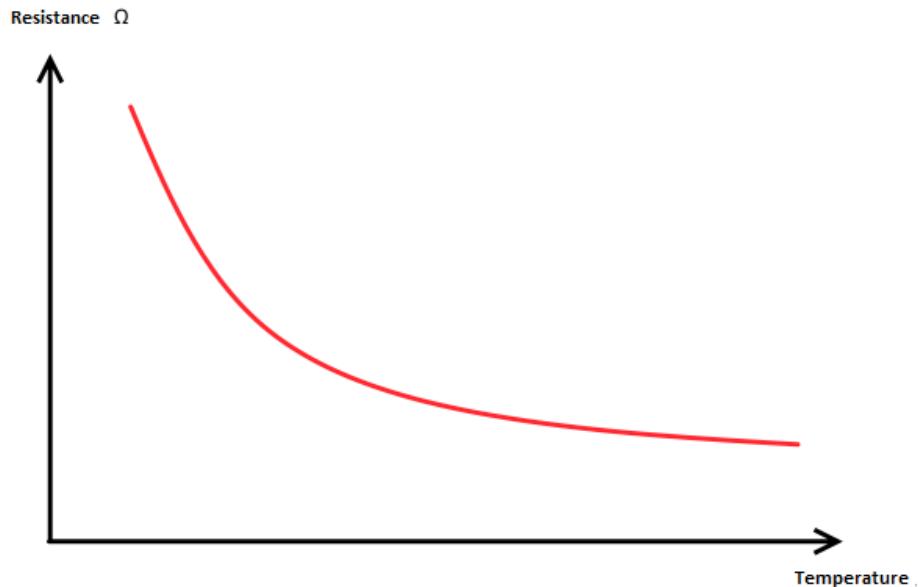


### Descripción:

Este módulo es un termistor NTC (Coeficiente Negativo de Temperatura). Al aumentar su temperatura varían considerablemente su resistencia, mayor temperatura menor resistencia.

Un termistor es un elemento de detección de temperatura compuesto por material semiconductor sintetizado que presenta un gran cambio en la resistencia en proporción a un cambio pequeño en la temperatura. En general, los termistores tienen coeficientes de temperatura negativos, lo que significa que la resistencia del termistor disminuye a medida que aumenta la temperatura.

Los termistores se fabrican con una mezcla de metales y materiales de óxido metálico. Una vez mezclados, los materiales se conforman y se hornean en la forma requerida. Los termistores pueden utilizarse tal cual, como termistores tipo disco, o seguir dándoles forma y montándolos con cables conductores y revestimientos para formar termistores tipo perla.



Rango de temperatura -55°C a +125°C

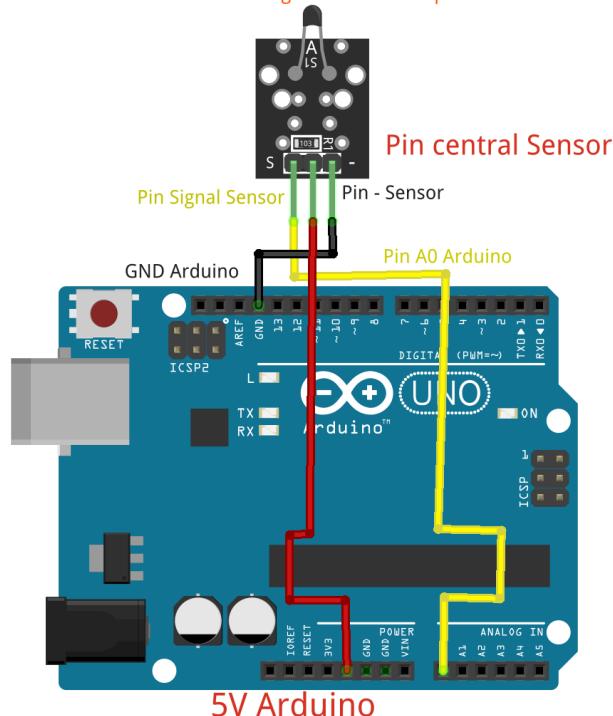
En cuanto a Raspberry Pi, nos enfrentamos a una pequeña barrera, ya que es un sensor analógico y no poseemos pines analógicos como en Arduino, por esto nos encontramos en la necesidad de utilizar un Convertidor Análogo-Digital que nos permita leer las medidas brindadas por el termistor. En este caso en particular, utilizaremos el Convertidor ADS1115 de Adafruit o mejor conocido como KY-053.



# Luciano Rabassa

## Conección Arduino:

KY-013: Módulo Analogo Sensor de Temperatura



fritzing

KY-013	a	Arduino
Signal	a	Pin Análogo A0
VCC	a	Pin 5V
-	a	Pin GND

## Código Arduino:

```
#include <math.h> //Incluimos math para realizar las cuentas matematicas

int Pin_S = A0;

double Thermistor(int RawADC) {
    double Temp;
    Temp = log(10000.0 * ((1024.0 / RawADC - 1)));
    Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp
* Temp )) * Temp );
    Temp = Temp - 273.15; // conversion de grados Kelvin a Celsius
    return Temp;
}

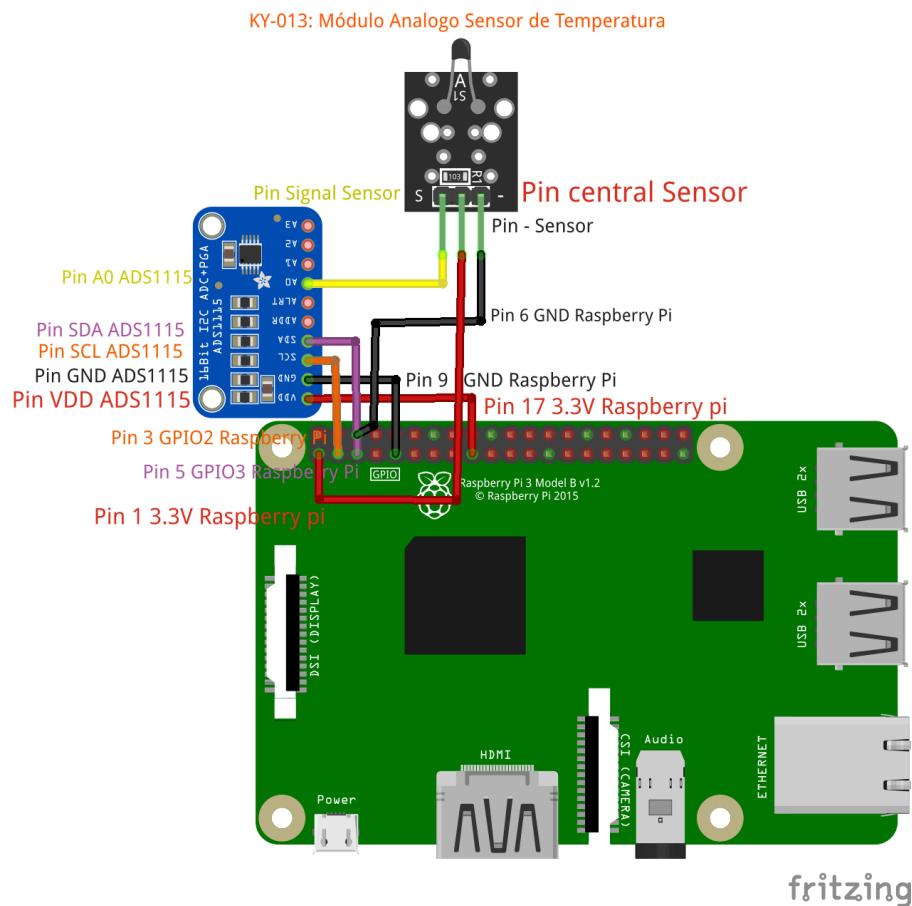
void setup(){
    Serial.begin(9600);
}

void loop(){
    int readVal = analogRead(Pin_S);
    double temp = Thermistor(readVal);

    Serial.print("La temperatura es: ");
    Serial.print(temp);
    Serial.print(char(186)); //Muestra el simbolo<°>.
    Serial.println("C");
    Serial.println("-----");
    delay(500);
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:



KY-013	a	Raspberry Pi
Signal	a	Pin A0 ADS1115
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND
VDD ADS1115	a	Pin 17 3.3V
GND ADS1115	a	Pin 9 GND
SCL ADS1115	a	Pin 3 GPIO2(Pin SCL en Raspberry Pi)
SDAADS1115	a	Pin 5 GPIO3(Pin SDA en Raspberry Pi)

## Código Raspberry Pi:

### Pre-Requisito:

#### Librería I2C ADS1x5 de Adafruit:

- <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code>

# Luciano Rabassa

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep
import math, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1015 = 0x00
ADS1115 = 0x01
gain = 4096
sps = 64
adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3
adc = ADS1x15(ic=ADS1115)

# Funcion para calcular la temperatura a traves de la tension
def calcTemp(voltage):
    temperature = math.log((10000/voltage)*(3300-voltage))
    temperature = 1 / (0.001129148 + (0.000234125 +
(0.000000876741 * temperature * temperature)) * temperature);
    temperature = temperature - 273.15;
    return temperature

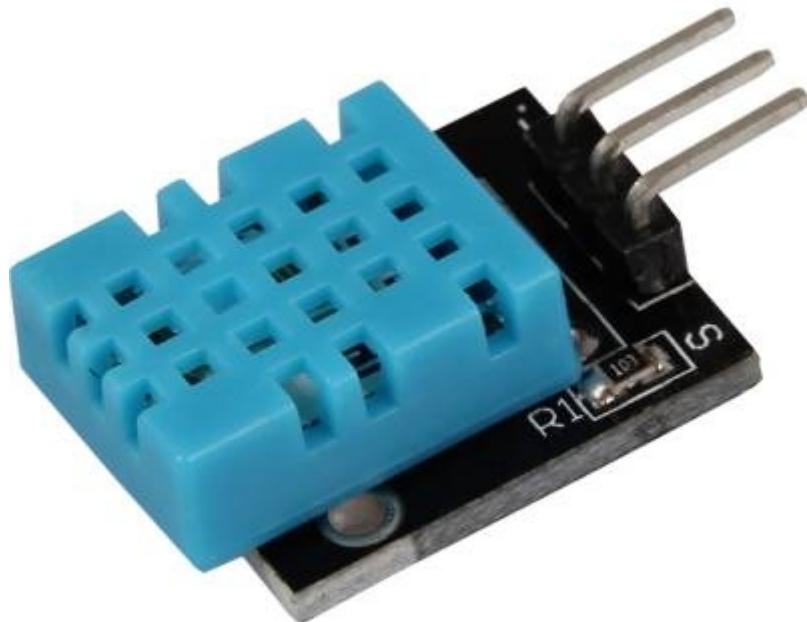
try:
    while True:
        #Lee la tension y calcula la temperatura
        temp0 = round( calcTemp(
adc.readADCSingleEnded(adc_channel_0, gain, sps)), 2)
        temp1 = round( calcTemp(
adc.readADCSingleEnded(adc_channel_1, gain, sps)), 2)
        temp2 = round( calcTemp(
adc.readADCSingleEnded(adc_channel_2, gain, sps)), 2)
        temp3 = round( calcTemp(
adc.readADCSingleEnded(adc_channel_3, gain, sps)), 2)
        print("Channel 0:", temp0, "C")
        print("Channel 1:", temp1, "C")
        print("Channel 2:", temp2, "C")
        print("Channel 3:", temp3, "C")
        print("-----")
        sleep(retraso)
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-013.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-013.py*

- KY-015 Módulo Combinado Sensor de Temperatura y Humedad



## Descripción:

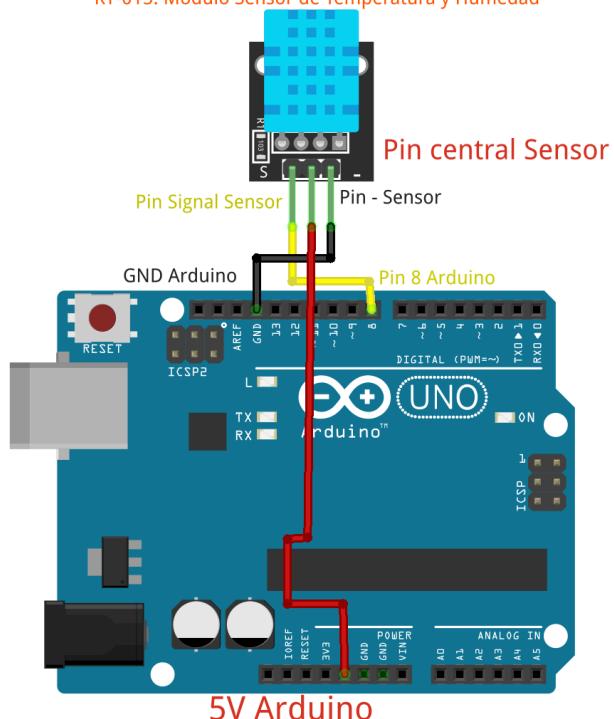
La principal ventaja de éste módulo, es que podemos medir humedad y temperatura, por otro lado, solo podemos tomar mediciones cada 2 segundos.

- Chipset: DHT11
- Protocolo de comunicación: 1-wire
- Rango de medición de humedad: 20-90%RH
- Rango de medición de temperatura: 0-50°C

# Luciano Rabassa

## Conección Arduino:

KY-015: Módulo Sensor de Temperatura y Humedad



fritzing

KY-015	a	Arduino
Signal	a	Pin Digital 8
VCC	a	Pin 5V
-	a	Pin GND

## Código Arduino:

### Pre-Requisitos:

1. Librerías “*DHT sensor library by Adafruit*”, “*Adafruit Unified Sensor by Adafruit*”.
2. Para esto abrimos Arduino IDE, vamos al menú “*Programa*” -> “*Incluir librerías...*” -> “*Gestionar librerías*”.
3. En el cuadro de búsqueda escribimos “*DHT*” y seleccionamos “*Instalar*” en la opción “*DHT sensor library by Adafruit*”. Luego escribimos en el campo de búsqueda “*Unified*”, bajamos hasta el final de los resultados y seleccionamos “*Instalar*” en la opción “*Adafruit Unified Sensor by Adafruit*”.

# Luciano Rabassa

Ahora en el IDE vamos al menú “**Archivos**” -> “**Ejemplos**” -> “**DHT Sensor library**” -> “**DHT Tester**“.

Compilamos y cargamos el sketch al Arduino para comprobar que nuestro Sensor funciona, aunque nos diga que estamos en el infierno a 5000 grados centigrados, veremos lecturas de Temperatura y Humedad.

Luego creamos un nuevo Sketch nombrándolo como nos apetezca, si el código será hecho desde cero debemos importar la librería “**DHT Sensor by Adafruit**“.

```
#include <DHT_U.h> // Incluimos la librería Adafruit_DHT
#include <DHT.h>
#define DHTPIN 8 // Declaramos el de entrada
#define DHTTYPE DHT11 //DHT11
DHT dht(DHTPIN, DHTTYPE);

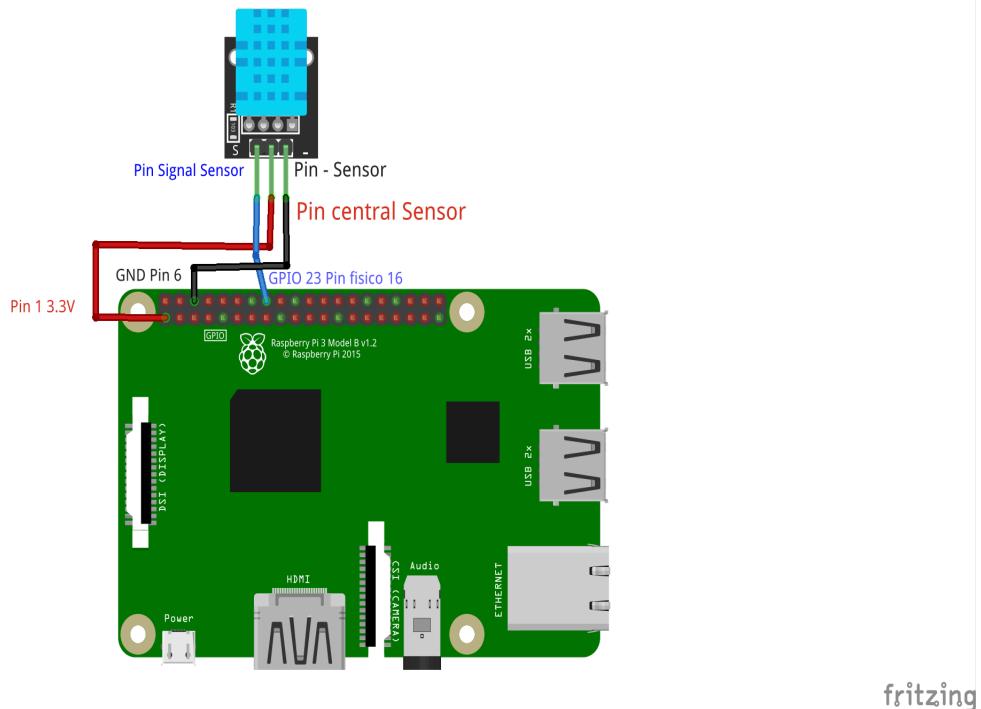
void setup()
{
    Serial.begin(9600);
    Serial.println("Prueba KY-015: ");
    dht.begin(); // Comenzamos la medición
}

void loop()
{
    delay(2000); // Pausa de 2 segundos entre medición
    float h = dht.readHumidity(); // Medición de humedad
    float t = dht.readTemperature(); // Medición de temperatura

    if(isnan(h) || isnan(t))
    {
        Serial.println("Error mientras leía el sensor");
        return;
    }
    Serial.println("-----");
    Serial.print("Humedad: ");
    Serial.print(h);
    Serial.print(" %\t");
    Serial.print("Temperatura: ");
    Serial.print(t);
    Serial.print(char(186)); // Salida <°> simbolo grado
    Serial.println("C ");
    Serial.println("-----");
    Serial.println(" ");
}
```

## Conexión Raspberry Pi:

KY-015: Módulo Sensor de Temperatura y Humedad



KY-015	a	Raspberry Pi
Signal	a	Pin 16 GPIO23
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND

## Código Raspberry Pi:

1. Necesitamos asegurarnos que nuestro sistema tenga las extensiones de Python disponibles para compilar y git para descargar la librería.
  - `sudo apt-get install build-essential python-dev python-openssl git`
2. Descargamos la librería de Adafruit:
  - `git clone https://github.com/adafruit/Adafruit_Python_DHT.git`
3. Nos movemos a la nueva carpeta:
  - `cd Adafruit_Python_DHT`
4. Instalamos la librería con:
  - `sudo python setup.py install`
5. Para que Raspberry se comunique a través de i2c con el sensor, debemos activar la función en `config.txt`:

- *sudo mousepad /boot/config.txt*

## 6. Dónde debemos descomentar la linea:

- *dtparam=i2c\_arm=on*

## 7. Guardamos y descargamos algunas herramientas i2c necesarias:

- *sudo apt-get install python-smbus i2c-tools -y*

## 8. Reiniciamos la Raspberry Pi:

- *sudo reboot*

```
#!/usr/bin/env python
# coding=utf-8
# Importamos los modulos necesarios
import RPi.GPIO as GPIO
import Adafruit_DHT
import time

sleepTime = 2 # configuramos una pausa de 2 segundos
# El sensor debe ser configurado como Adafruit_DHT.DHT11,
# Adafruit_DHT.DHT22, o Adafruit_DHT.AM2302.
DHTSensor = Adafruit_DHT.DHT11
GPIO_Pin_Signal = 23 # Declaramos a que pin lo hemos conectado

print("KY-015 prueba de Temperatura y humedad")

try:
    while(1):
        # La medicion sera escrita en las variables humid temper
        humedad,temperatura = Adafruit_DHT.read_retry(DHTSensor,
GPIO_Pin_Signal)
        print("-----")
        if humedad is not None and temperatura is not None:
            print("Temperatura={0:0.1f}°C|rel.humidity={1:0.1f}"
%' .format(temperatura, humedad))
        # A causa del sistema Linux, la Raspberry Pi tiene problemas para obtener
        # mediciones en tiempo real esto es por, problemas de sincronización, lo que
        # causa fallas de comunicación.
        # En ese caso, un mensaje de error se mostrará -el resultado se mostrará
        # en el próximo intento.
        else:
            print("Error al leer, por favor reintentalo!")
            print("-----")
            print(" ")
            time.sleep(sleepTime)
```

# Luciano Rabassa

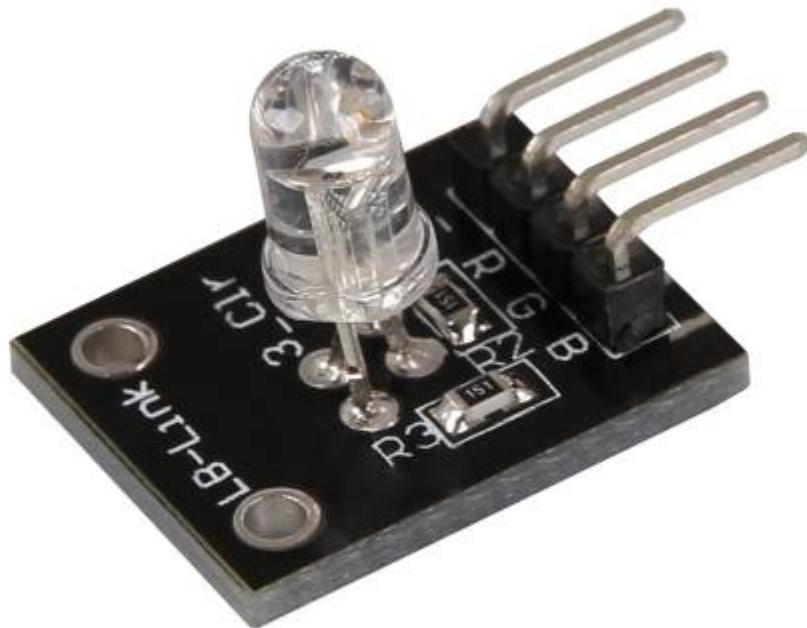
```
# Limpiamos los estados de GPIO
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-015.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-015.py*

## KY-016: Módulo LED THT RGB Tricolor

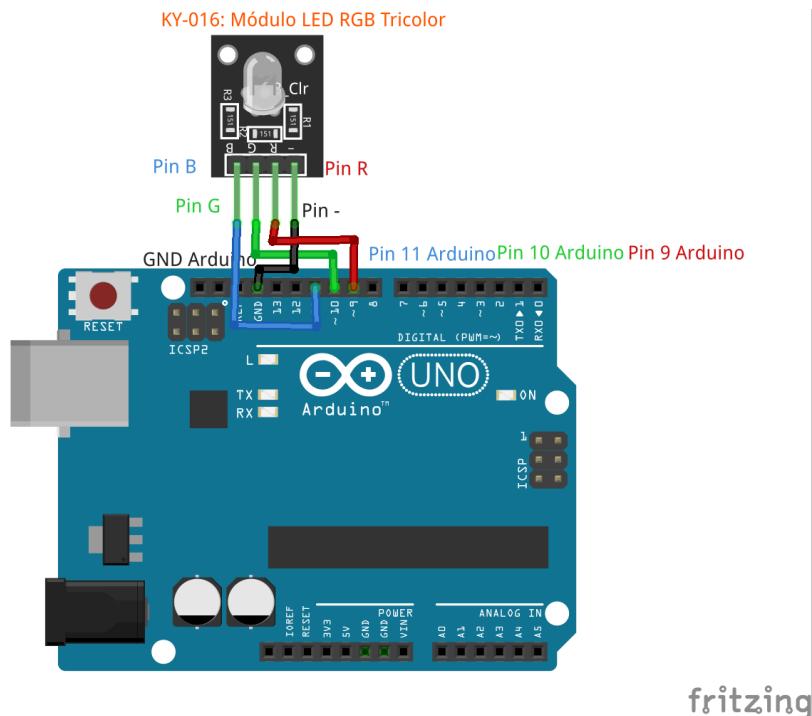


### Descripción:

Led THT 5mm Tricolor con 3 resistencias SMD de  $150\Omega$ . Podríamos construir nuestro propio módulo en el protoboard o en una placa PCB Experimental, colocando los 4 pines del LED THT 5mm Cátodo común, el pin al lado del cátodo es el R(rojo), del otro lado del cátodo tendremos los pines G(verde) y B(azul). Quedándonos R, GND, G, B, a los 3 leds les conectaremos una resistencia THT de  $150\Omega$  (Marrón, Verde, Negro, Negro, Marrón) y soldaremos cuatro pines si utilizamos un PCB Experimental.

# Luciano Rabassa

## Conección Arduino:



KY-016	a	Arduino
R	a	Pin 9
G	a	Pin 10
B	a	Pin 11
-	a	Pin GND

## Código Arduino:

```
int Led_Rojo = 9;
int Led_Verde = 10;
int Led_Azul = 11;

void setup()
{
    pinMode(Led_Rojo, OUTPUT);
    pinMode(Led_Verde, OUTPUT);
    pinMode(Led_Azul, OUTPUT);
}

void loop()
{
```

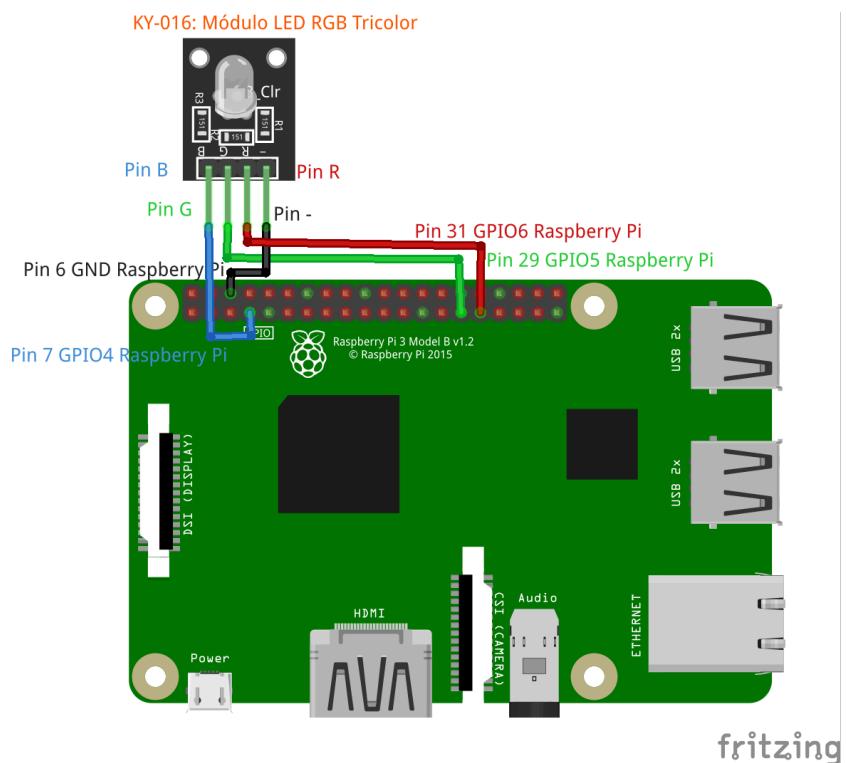
# Luciano Rabassa

```
digitalWrite(Led_Rojo, HIGH);
digitalWrite(Led_Verde, LOW);
digitalWrite(Led_Azul, LOW);
delay(3000);

digitalWrite(Led_Rojo, LOW);
digitalWrite(Led_Verde, HIGH);
digitalWrite(Led_Azul, LOW);
delay(3000);

digitalWrite(Led_Rojo, LOW);
digitalWrite(Led_Verde, LOW);
digitalWrite(Led_Azul, HIGH);
delay(3000);
}
```

## Conección Raspberry Pi:



KY-016	a	Raspberry Pi
R	a	Pin 31 GPIO6
G	a	Pin 29 GPIO5
B	a	Pin 7 GPIO4
-	a	Pin 6 GND

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

LED_Rojo = 6
LED_Verde = 5
LED_Azul = 4

GPIO.setup(LED_Rojo, GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(LED_Verde, GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(LED_Azul, GPIO.OUT, initial = GPIO.LOW)

print("Prueba de modulo [Presiona Ctrl + C para finalizar la prueba]")

try:
    while True:
        print("Led Rojo se enciende por 3 segundos")
        GPIO.output(LED_Rojo, GPIO.HIGH)
        GPIO.output(LED_Verde, GPIO.LOW)
        GPIO.output(LED_Azul, GPIO.LOW)
        time.sleep(3)
        print("Led Verde se enciende por 3 segundos")
        GPIO.output(LED_Rojo, GPIO.LOW)
        GPIO.output(LED_Verde, GPIO.HIGH)
        GPIO.output(LED_Azul, GPIO.LOW)
        time.sleep(3)
        print("Led Azul se enciende por 3 segundos")
        GPIO.output(LED_Rojo, GPIO.LOW)
        GPIO.output(LED_Verde, GPIO.LOW)
        GPIO.output(LED_Azul, GPIO.HIGH)
        time.sleep(3)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-016 .py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-016.py*

## KY-017: Módulo Interruptor de Mercurio



### Descripción:

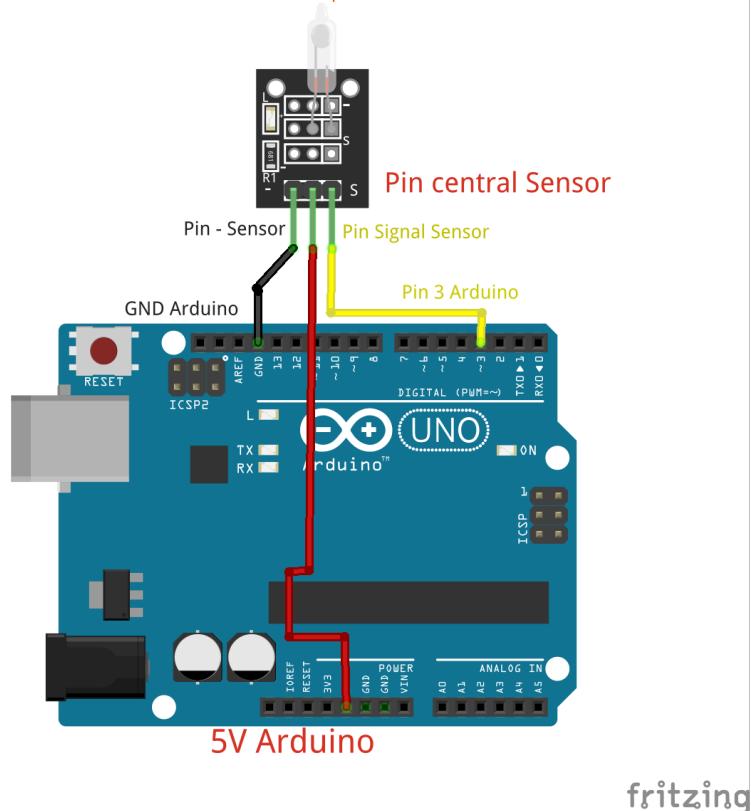
**PRECAUCIÓN** peligro de contaminación por Mercurio, no apto para niños, no dejar a su alcance, hay otros módulos que realizan lo mismo sin peligro!.

Es un conmutador de inclinación de mercurio que le permite detectar la inclinación de su objeto para que pueda tomar la acción apropiada. Es una buena alternativa de bajo costo para un acelerómetro de 6 ejes. Posee un Led indicador y una resistencia de  $18000\text{ M}\Omega$ .

# Luciano Rabassa

## Conección Arduino:

KY-017: Módulo Interruptor de Mercurio



KY-017	a	Arduino
-	a	Pin GND
VCC	a	Pin 5V
Signal	a	Pin Digital 3

## Código Arduino:

```
int Pin_S = 3;
int tiltVal;
boolean bIsTilted;

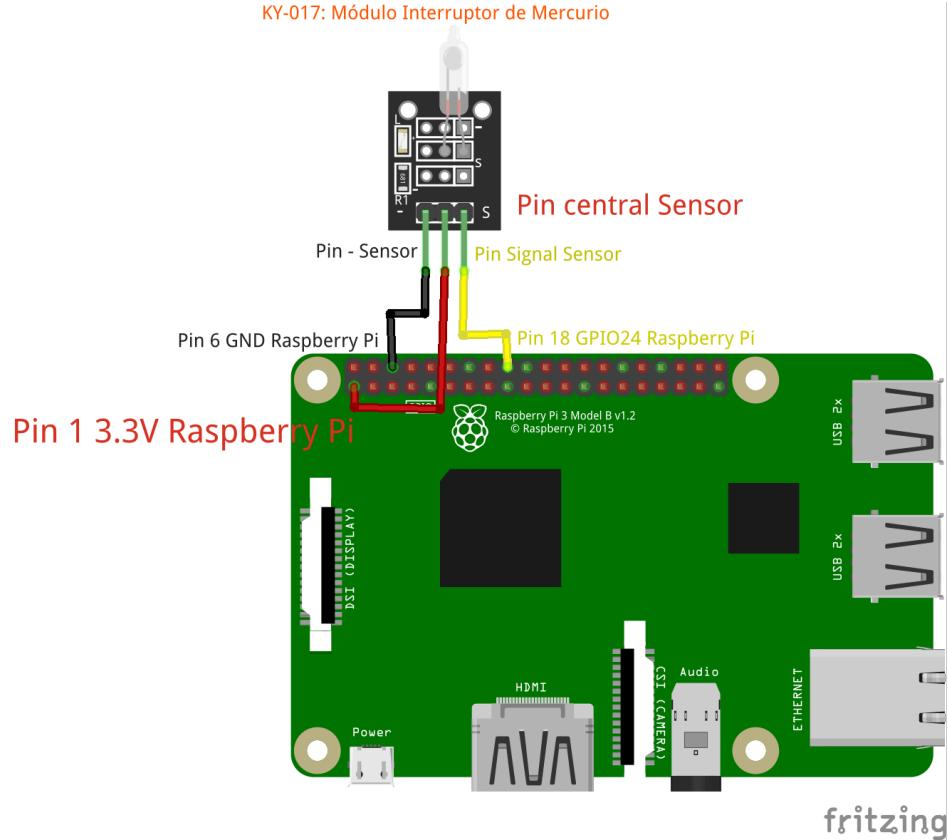
void setup()
{
  Serial.begin(9600);
  pinMode(Pin_S, INPUT);
}

void loop()
```

```
{  
    tiltVal = digitalRead(Pin_S);  
    if(tiltVal == HIGH)  
    {  
        if(!bIsTilted)  
        {  
            bIsTilted = true;  
            Serial.println("Inclinado");  
        }  
    }  
    else  
    {  
        if(bIsTilted)  
        {  
            bIsTilted = false;  
            Serial.println("No inclinado");  
        }  
    }  
}
```

## Conexión Raspberry Pi:

KY-017: Módulo Interruptor de Mercurio



# Luciano Rabassa

KY-017	a	Raspberry Pi
-	a	Pin 6 GND
VCC	a	Pin 1 3.3V
Signal	a	Pin 18 GPIO24

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_Pin_Signal = 24
GPIO.setup(GPIO_Pin_Signal, GPIO.IN)

print("Prueba de modulo [Presiona Ctrl + C para finalizar la prueba] ")

def funcionDetectar(null):
    print("Ha sido detectado")

GPIO.add_event_detect(GPIO_Pin_Signal, GPIO.FALLING, callback =
funcionDetectar, bouncetime = 100)

try:
    while True:
        time.sleep(1)

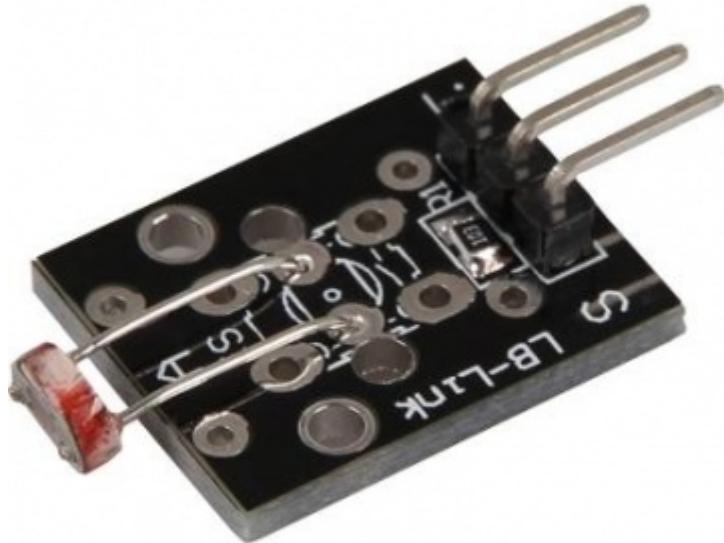
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-017.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-017.py*

## KY-018: Módulo Fotoresistor

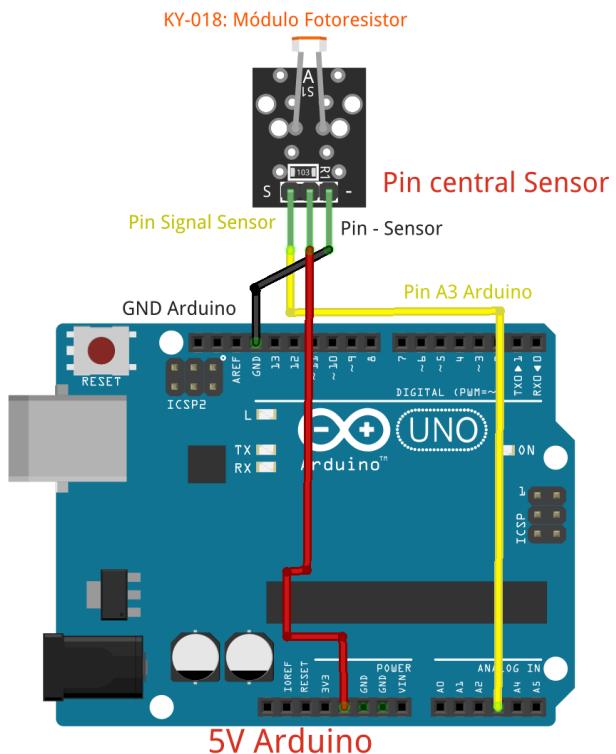


### Descripción:

Posee un fotoresistor con una resistencia de  $10\text{ K}\Omega$ . Es un sensor análogo por lo que usaremos un convertidor ADC (KY-053 ADS1115) en Raspberry Pi.

# Luciano Rabassa

## Conección Arduino:



fritzing

KY-018	a	Arduino
Signal	a	Pin Analógico A3
VCC	a	Pin 5V
-	a	Pin GND

## Código Arduino:

```
int Pin_Signal = A3;

void setup(){
  Serial.begin(9600);
}

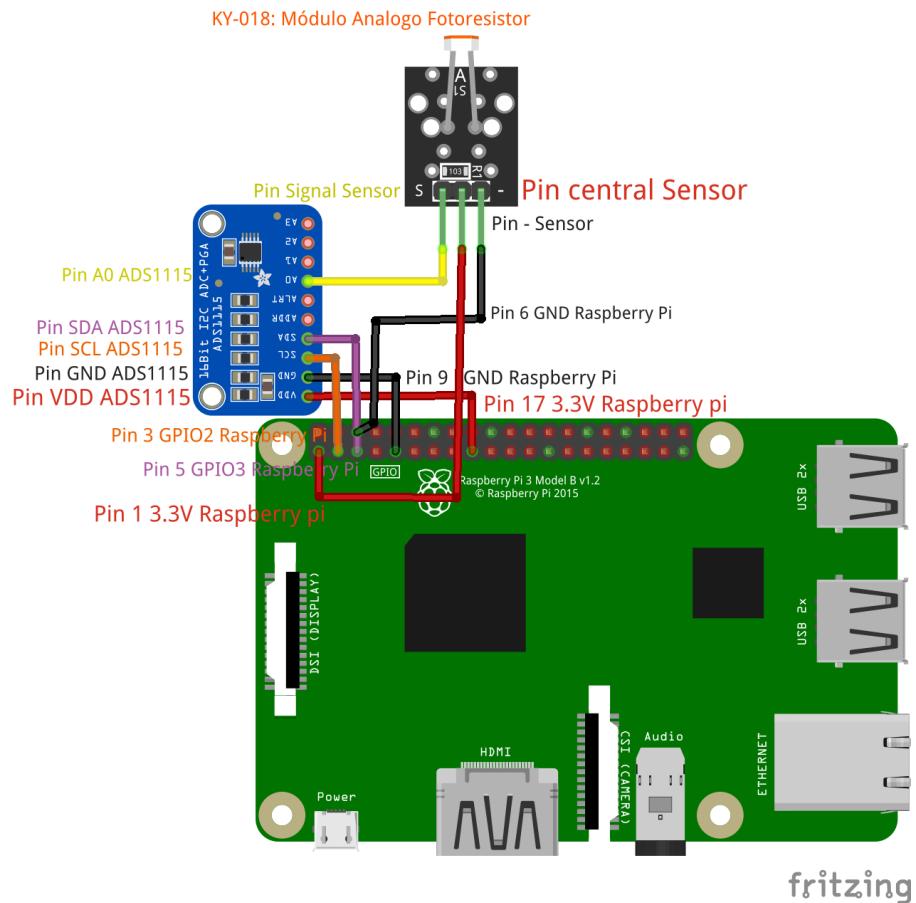
void loop(){
  int valorBruto = analogRead(Pin_Signal);
  float Tension = valorBruto * (5.0/1023) * 1000;
  float Resistencia = 10000 * (Tension / ( 5000.0 - Tension));

  Serial.print("Valor de la tension:");Serial.print(Tension);
  Serial.print("mV");
}
```

# Luciano Rabassa

```
Serial.print(", Valor de la  
resistencia:"); Serial.print(Resistencia); Serial.println("Ohm");  
Serial.println("-----");  
delay(500);  
}
```

## Conexión Raspberry Pi:



KY-018	a	Raspberry Pi
Signal	a	Pin A0 de ADS1115
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND
VDD ADS1115	a	Pin 17 3.3V
GND ADS1115	a	Pin 9 GND
SCL ADS1115	a	Pin 3 GPIO2(Pin SCL en Raspberry Pi)
SDA ADS1115	a	Pin 5 GPIO3(Pin SDA en Raspberry Pi)

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep
import time, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1115 = 0x01    # 16-bit
gain = 4096      # +/- 4.096V
sps = 64
adc_channel_0 = 0      # Channel 0
adc_channel_1 = 1      # Channel 1
adc_channel_2 = 2      # Channel 2
adc_channel_3 = 3      # Channel 3
adc = ADS1x15(ic=ADS1115)

try:
    while True:
        adc0 = adc.readADCSingleEnded(adc_channel_0, gain, sps)
        adc1 = adc.readADCSingleEnded(adc_channel_1, gain, sps)
        adc2 = adc.readADCSingleEnded(adc_channel_2, gain, sps)
        adc3 = adc.readADCSingleEnded(adc_channel_3, gain, sps)
        print("Channel 0:", adc0, "mV ")
        print("Channel 1:", adc1, "mV ")
        print("Channel 2:", adc2, "mV ")
        print("Channel 3:", adc3, "mV ")
        print("-----")
        time.sleep(retraso)

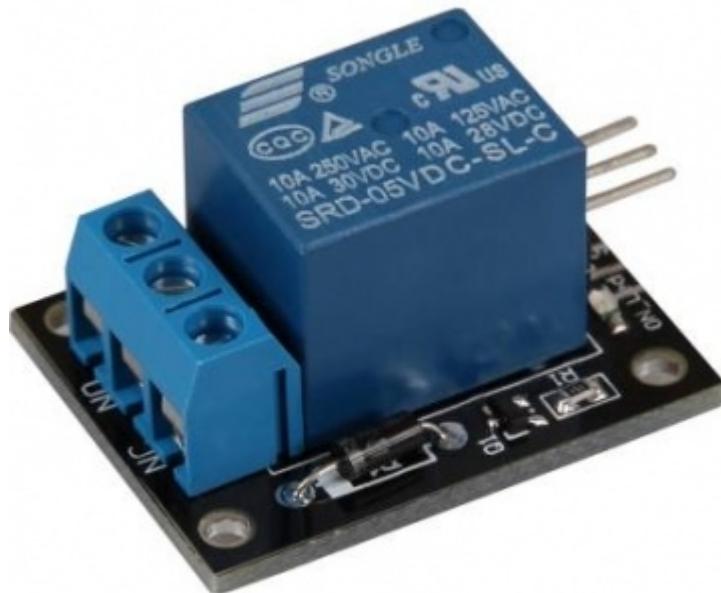
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-018.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-018.py*

## KY-019: Módulo Relay 5V



### Descripción:

Es un dispositivo eléctrico que funciona como un interruptor pero es accionado mediante un electro imán. Posee 5 o 6 contactos, en el caso de Raspberry Pi posee 6 ya que uno es de señal, positivo y GND como entradas, del lado de las salidas posee 3 contactos NO(normalmente abierto-open-) NC(normalmente cerrado-close-), C(común-common-). Señal de control TTL 5v a 12v.

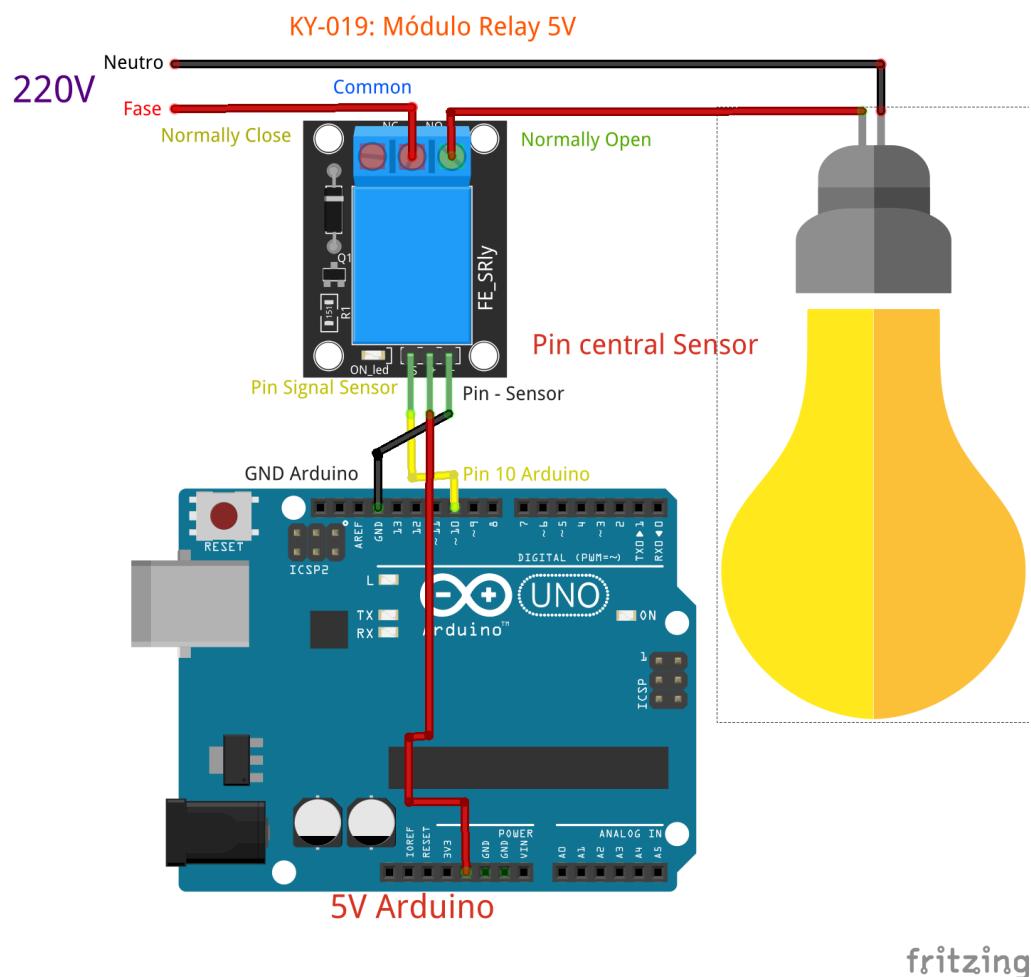
- **10 A 250V Corriente Alterna Máximo.**
- **10A 30V Corriente Continua Máximo.**
- **3 es de entrada S, +, -.**
- **3 Salidas NC, C, NO.**
- **Led Indicador de bobina excitada.**
- **Diodo rectificador de protección.**

Al excitar la bobina con el positivo(+5v) y GND(-) se genera un campo electromagnético que hace cambiar de estado a los contactos, es decir, NO se cierra y NC se abre. Esto conlleva un

# Luciano Rabassa

riesgo para la Raspberry pi, en particular, que viene solucionado en el módulo de relay (KY-019). Cuál es el riesgo?. La Corriente inversa, al dejar de excitar los contactos por particularidades propias de la física, obtenemos una corriente inversa a la que ofrecemos al sistema, lo cuál, pese a ser baja, podría dañar nuestro Raspberry Pi, para evitar esto, se coloca un diodo que no permite el regreso de esa corriente. Es por esto, que debemos respetar la polaridad indicada por el fabricante, normalmente esto no importa, si alimentamos el (-) con 5v y el (+) con GND el relay funcionaría perfectamente, pero como tenemos un diodo protegiéndonos, hay que respetar lo que indica el fabricante.

## Conexión Arduino:



# Luciano Rabassa

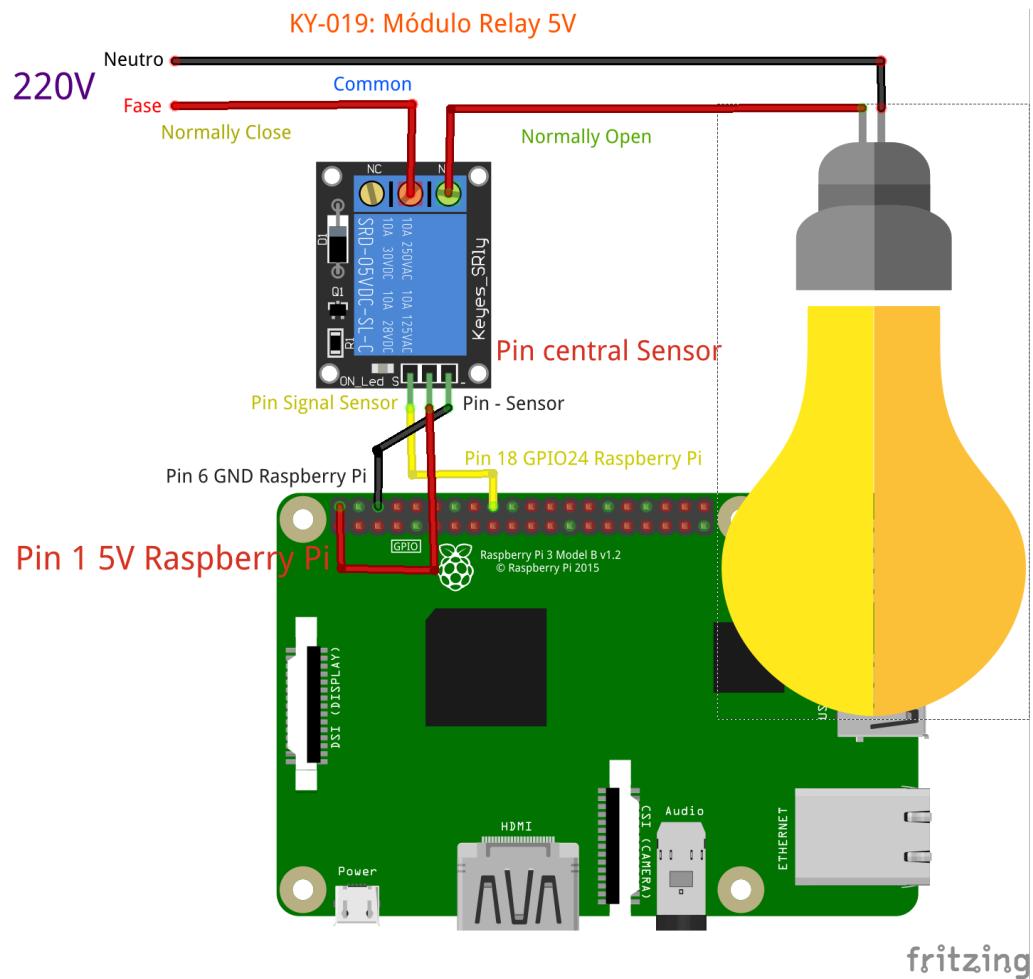
KY-019	a	Arduino
Signal	a	Pin Digital 10
VCC	a	Pin 5V
-	a	Pin GND
Neutro	a	Lampara o Carga
Fase	a	Common de Relay
NO de Relay	a	Lampara o Carga

## Código Arduino:

```
int Pin_Signal = 10;  
  
int retraso = 1;  
  
void setup()  
{  
    pinMode(Pin_Signal, OUTPUT);  
}  
  
void loop()  
{  
    digitalWrite(Pin_Signal, HIGH);  
    delay(retraso * 1000);  
    digitalWrite(Pin_Signal, LOW);  
    delay(retraso * 1000);  
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:



KY-019	a	Raspberry Pi
Signal	a	Pin 18 GPIO24
VCC	a	Pin 2 5V
-	a	Pin 6 GND
Neutro	a	Lampara o Carga
Fase	a	Common de Relay
NO de Relay	a	Lampara o Carga

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

retraso = 1
Pin_S = 24

GPIO.setup(Pin_S, GPIO.OUT)
GPIO.output(Pin_S, False)

print("Prueba del modulo [Presiona Ctrl + C para finalizar la prueba]")

try:
    while True:
        GPIO.output(Pin_S,True)
        time.sleep(retraso)
        GPIO.output(Pin_S,False)
        time.sleep(retraso)

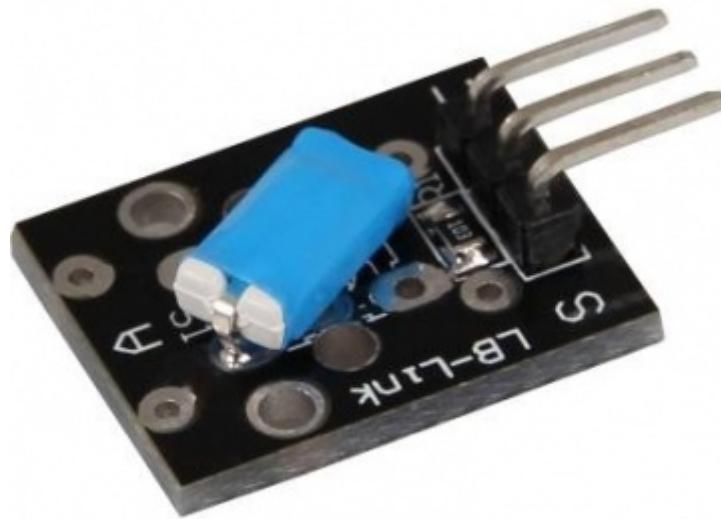
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-019.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-019.py*

## KY-020: Módulo Interruptor de inclinación



### Descripción:

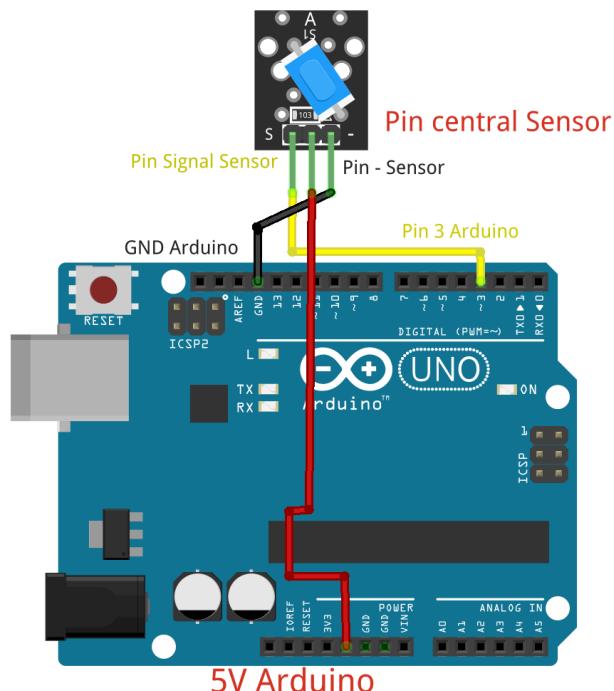
Este es un módulo que cuenta con una salida digital que es activada ante el cambio en el grado de inclinación del sensor integrado en la placa.

- Voltaje de funcionamiento: 3.3V ~ 5V
- Corriente y Tensión máxima: 50mA 12VDC
- Resistencia de los contactos Resistencia de contacto: 50 max (inicial)
- Resistencia de aislamiento Resistencia de aislamiento: 100M (min 250V DC)
- Resistencia a la compresión Resistencia dieléctrica: AC250V (50 / 60Hz para 1 minuto)
- Vida mecánica: 100.000 ciclos
- Temperatura ambiente Temperatura ambiente: -25° C a 105°C

# Luciano Rabassa

## Conección Arduino:

KY-020: Módulo Interruptor de Inclinación



fritzing

KY-020	a	Arduino
Signal	a	Pin Digital 3
VCC	a	Pin 5V
-	a	Pin GND

## Código Arduino:

```
int Pin_Signal = 3;
int valor;

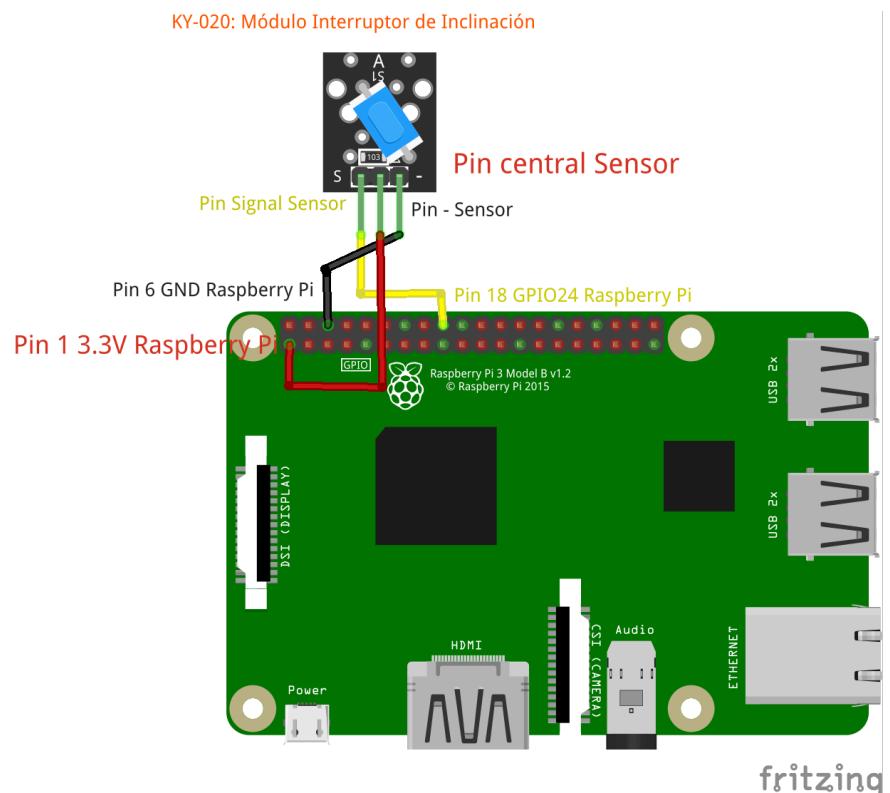
void setup()
{
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(Pin_Signal, INPUT);
}

void loop()
{
    valor = digitalRead(Pin_Signal);
```

# Luciano Rabassa

```
if(valor == HIGH)
{
    digitalWrite(LED_BUILTIN, LOW);
}
else
{
    digitalWrite(LED_BUILTIN, HIGH);
}
```

## Conexión Raspberry Pi:



KY-020	a	Raspberry Pi
Signal	a	Pin 18 GPIO24
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_Pin_Signal = 24
GPIO.setup(GPIO_Pin_Signal, GPIO.IN)

print("Prueba del modulo [Presiona Ctrl + C para finalizar la prueba] ")

def funcionDetectar(null):
    print("Ha sido detectado")

GPIO.add_event_detect(GPIO_Pin_Signal, GPIO.FALLING, callback =
funcionDetectar, bouncetime = 100)

try:
    while True:
        time.sleep(1)

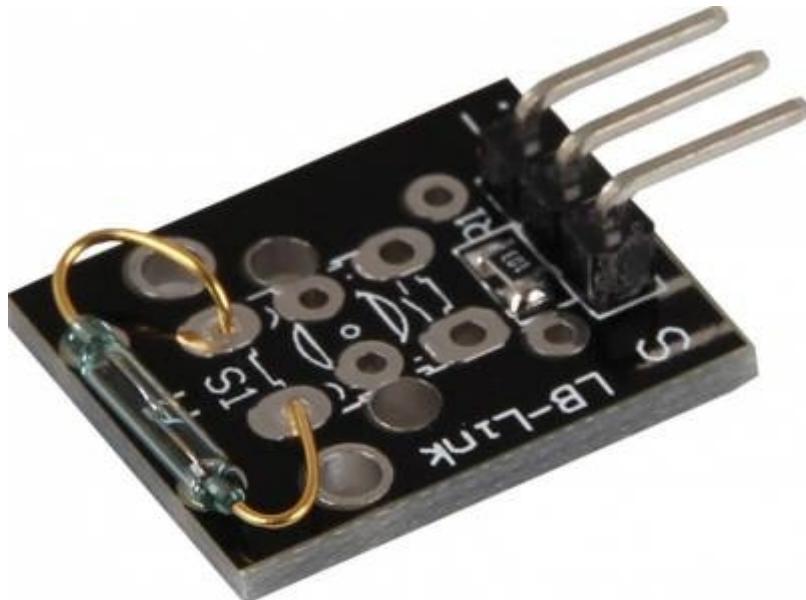
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-020.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-020.py*

## KY-021: Módulo mini interruptor magnético Reed



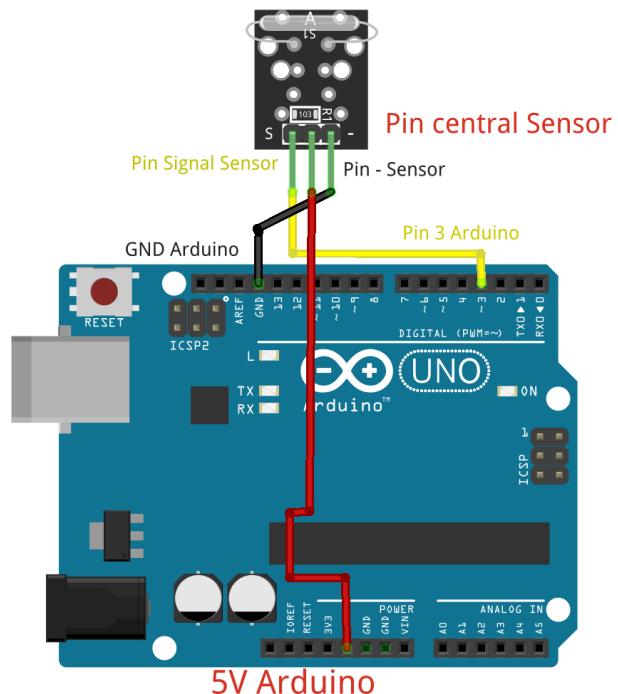
### Descripción:

Interruptor magnético normalmente abierto (NO), que cierra su contacto (NC) al detectar un campo magnético cercano.

# Luciano Rabassa

## Conección Arduino:

KY-021: Módulo Mini Interruptor reed



fritzing

KY-021	a	Arduino
Signal	a	Pin digital 3
VCC	a	Pin 5V
-	a	Pin GND

## Código Arduino:

```
int Pin_Signal = 3;
int valor;

void setup()
{
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(Pin_Signal, INPUT);
}

void loop()
{
    valor = digitalRead(Pin_Signal);
```

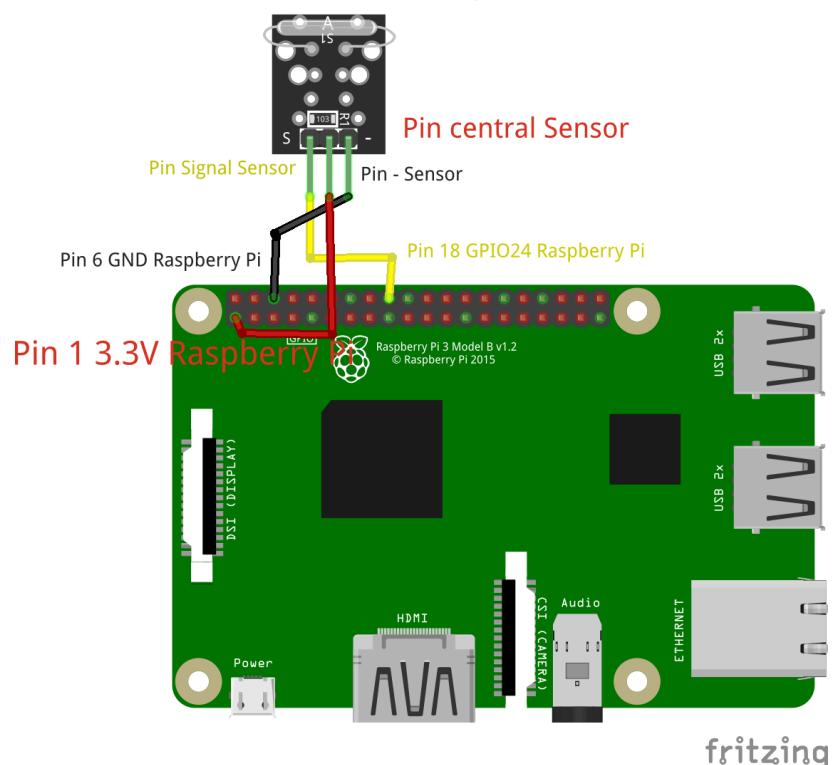
```

if(valor == HIGH)
{
    digitalWrite(LED_BUILTIN, LOW);
}
else
{
    digitalWrite(LED_BUILTIN, HIGH);
}
}

```

## Conección Raspberry Pi:

KY-021: Módulo Mini Interruptor reed



KY-021	a	Raspberry Pi
Signal	a	Pin 18 GPIO24
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_Pin_Signal = 24
GPIO.setup(GPIO_Pin_Signal, GPIO.IN)

print("Prueba del modulo [Presiona Ctrl + C para finalizar la prueba] ")

def funcionDetectar(null):
    print("Ha sido detectado")

GPIO.add_event_detect(GPIO_Pin_Signal, GPIO.FALLING, callback =
funcionDetectar, bouncetime = 100)

try:
    while True:
        time.sleep(1)

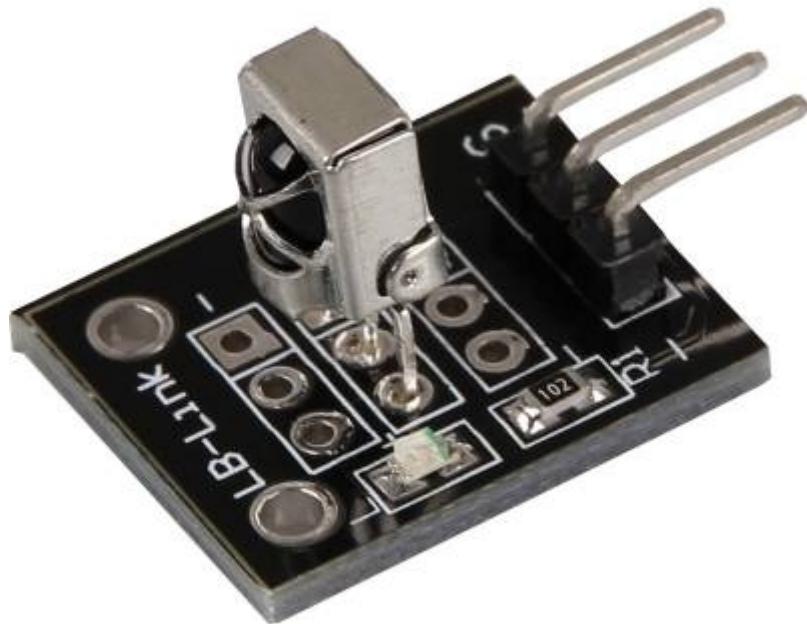
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-021.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-021.py*

## KY-022: Módulo receptor de infrarrojos



### Descripción:

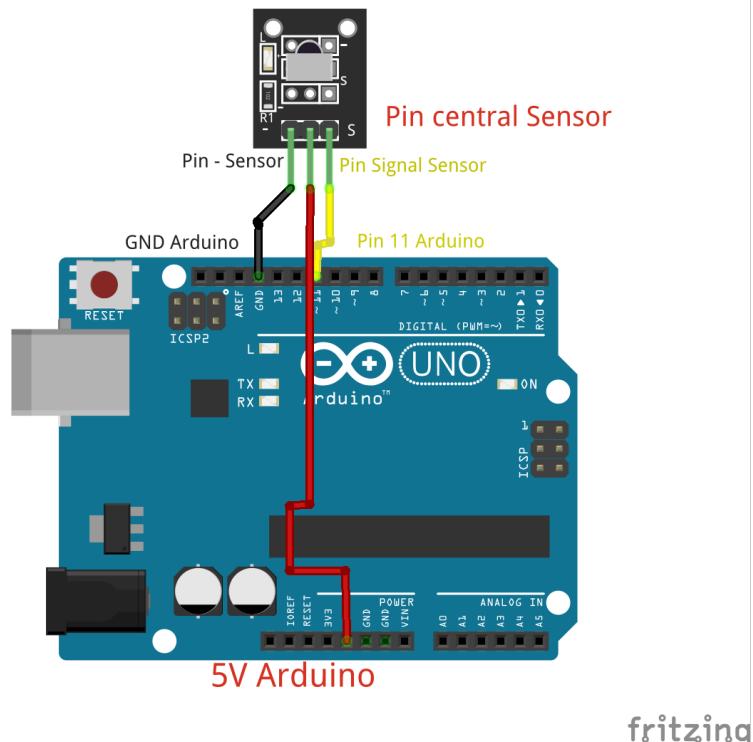
Módulo receptor de rayos infrarrojos a 38 Khz, una resistencia de  $1\text{ k}\Omega$  y un LED.

- Distancia: Mayor a 8 metros
- Onda: 940Nm
- Frecuencia del cristal: 455KHZ crystal
- Frecuencia de emisión: 38KHZ
- Encoding: encoding format for the NEC
- Power: CR2025/1600mAH

# **Luciano Rabassa**

## Conexión Arduino:

KY-022: Módulo receptor de infrarrojos



<b>KY-022</b>	<b>a</b>	<b>Arduino</b>
<b>Signal</b>	<b>a</b>	<b>Pin Digital 11</b>
<b>VCC</b>	<b>a</b>	<b>Pin 5V</b>
<b>-</b>	<b>a</b>	<b>Pin GND</b>

## Código Arduino:

```
#include <IRremote.h>
#include <IRremoteInt.h>

int Pin_Signal = 11;

IRrecv irrecv(Pin_Signal);
decode_results resultado;

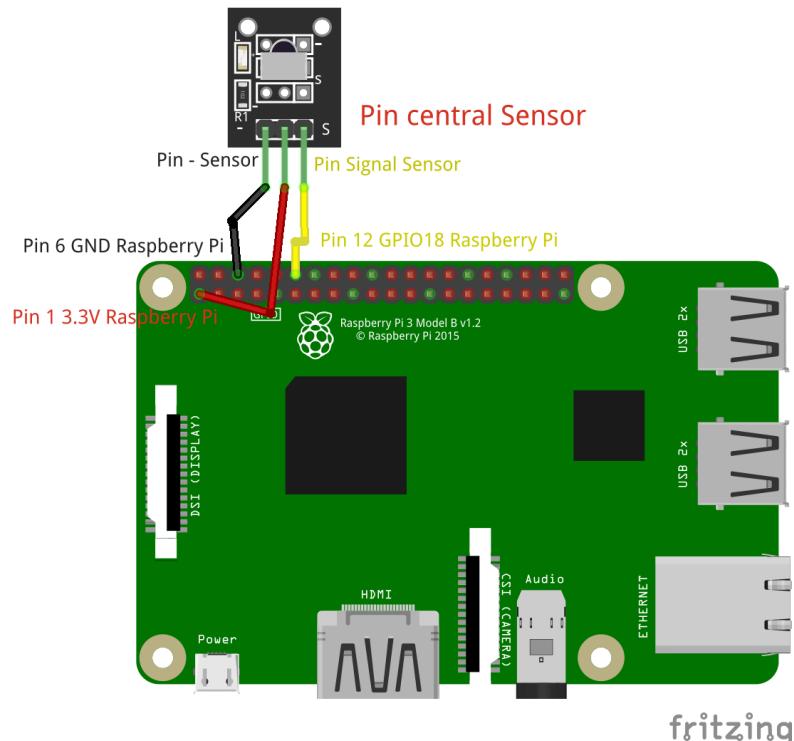
void setup()
{
    Serial.begin(9600);
    irrecv.enableIRIn();
}
```

# Luciano Rabassa

```
void loop() {
    if(irrecv.decode(&resultado)){
        Serial.println(resultado.value, HEX);
        irrecv.resume();
    }
}
```

## Conección Raspberry Pi:

KY-022: Módulo receptor de infrarrojos



fritzing

KY-022	a	Raspberry Pi
Signal	a	Pin 12 GPIO18
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND

## Código Raspberry Pi:

### Pre-Requisito LIRC:

- ***sudo apt-get instal lirc -y***

Editamos config.txt:

- ***sudo mousepad /boot/config.txt***

Buscamos la siguiente línea, casi al terminar el archivo y la descomentamos borrando el # y agregando ( , gpio\_in\_=18):

- ***dtoverlay=lirc-rpi, gpio\_in\_=18***

Guardamos y cerramos.

Editamos las siguientes líneas en lirc\_options.conf:

- ***sudo mousepad /etc/lirc/lirc\_options.conf***
- ***driver = default***
- ***device = /dev/lirc0***

Guardamos y cerramos.

Paramos y reiniciamos lirc:

- ***sudo systemctl stop lircd***
- ***sudo systemctl start lircd***

Nos fijamos que Lirc esté corriendo:

- ***sudo systemctl status lircd***

Reiniciamos la Raspberry Pi:

- ***sudo reboot***

Probamos que está funcionando:

- ***sudo systemctl stop lircd***
- ***mode2 -d /dev/lirc0***

# Luciano Rabassa

Nos aparecerá lo siguiente:

```
Using driver devinput on device /dev/lirc0
```

```
Trying device: /dev/lirc0
```

```
Using device: /dev/lirc0
```

Presionamos cualquier botón y veremos líneas similares con diferentes valores:

```
pulse 560
```

```
space 1706
```

```
pulse 535
```

E ira aumentando a medida que toquemos los botones. Presionamos "Ctrl+C" para cerrar.

Ahora debemos grabar nuestro control remoto:

Abriendo otra Terminal vemos los nombres disponibles que poseemos:

- ***irrecord --list-namespace***

Detenemos Lirc:

- ***sudo systemctl stop lircd***
- ***sudo irrecord -d /dev/lirc0 ~/lircd.conf***

Seguimos las instrucciones que nos pide:

Entrar el nombre que le daremos al control remoto.

Presionar "*Enter*" para empezar a grabar.

Presionamos cada botón del control hasta que nos pide el nombre del siguiente botón, si no lo hay presionamos *ENTER* y se habrá guardado nuestro archivo de configuración con los códigos de cada botón.

El archivo que creamos queda como el siguiente:

```
begin remote
```

```
name BLACK
```

```
driver devinput
```

```
bits      56
```

```
eps       30
```

```
aeps     100
```

```
one       0   0
```

# Luciano Rabassa

```
zero      0  0
pre_data_bits 72
pre_data    0x3
gap       361
toggle_bit_mask 0x0
frequency 38000

begin codes
  KEY_MENU        0x4E0000010001E3
  BTN_DPAD_UP     0x610000010001CF
  BTN_DPAD_DOWN   0x610000010001CE
  BTN_DPAD_LEFT   0x630000010001CF
  BTN_DPAD_RIGHT  0x680000010001C9
  BTN_MIDDLE      0x850000010001AB
  BTN_EXTRA       0x660000010001CB
  KEY_LAST        0x630000010001CD
  KEY_HOME         0x540000010001C3
  KEY_PLAY         0x630000010001CA
  KEY_PREVIOUSSONG 0x670000010001C6
  KEY_NEXTSONG    0x670000010001CC
  KEY_VOLUMEDOWN  0x610000010001CC
  KEY_VOLUMEUP    0x830000010001AF
  KEY_SHUFFLE     0x4D0000010001E4
end codes
```

end remote

Ahora hacemos un backup del lirc.conf original:

- ***sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd\_original.conf***

Y copiamos nuestra grabación a lircd.conf:

- ***sudo cp /home/pi/BLACK.lircd.conf /etc/lirc/lircd.conf***

Iniciamos lirc y ya deberíamos tener funcionando nuestro control remoto:

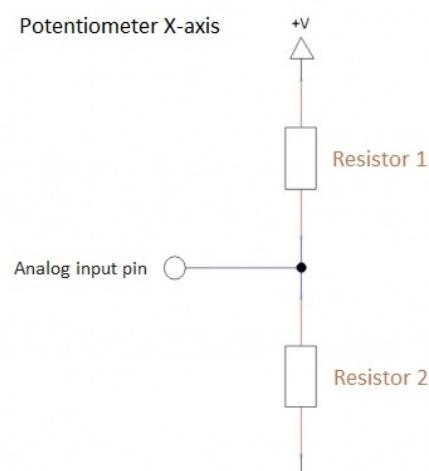
- ***sudo systemctl start lircd***

## KY-023: Módulo Joystick dual axis



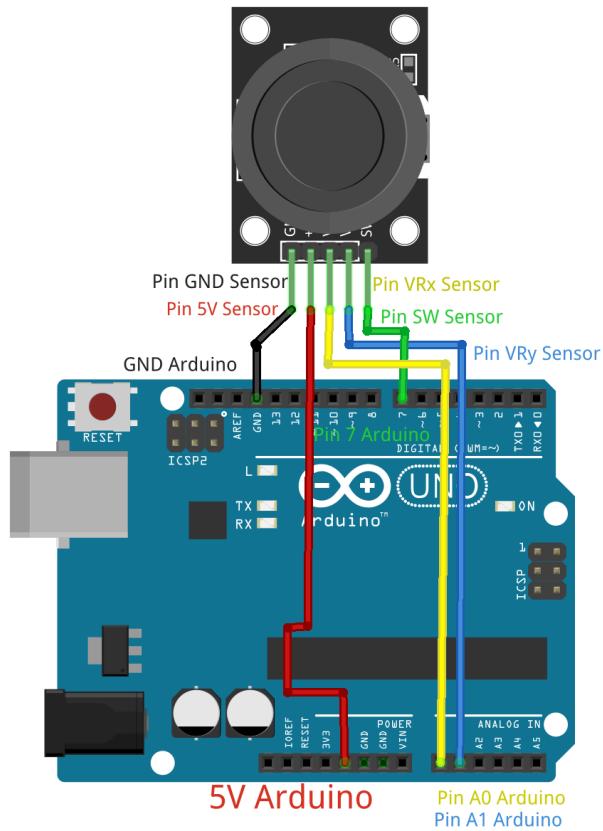
### Descripción:

El módulo joystick es utilizado para proyectos robótica que necesitan movilidad análoga, posee dos potenciómetros en un ángulo de 90 grados, genera una salida de 2.5V en X e Y cuando está en posición de descanso. Mover el joystick hará que la salida varíe de 0v a 5V dependiendo de su dirección.



## Conección Arduino:

KY-023: Módulo Joystick dual axis



fritzing

KY-023	a	Arduino
GND	a	Pin GND
+5V	a	Pin 5V
VRx	a	Pin Analógico A0
VRy	a	Pin Analógico A1
SW	a	Pin Digital 7

## Código Arduino:

```
int VR_X = A0;
int VR_Y = A1;
int SW = 7;

void setup()
{
    pinMode(VR_X, INPUT);
    pinMode(VR_Y, INPUT);
    pinMode(SW, INPUT);

    digitalWrite(SW, HIGH);
    Serial.begin(9600);
}

void loop()
{
    float x, y;
    int Boton;

    x = analogRead((VR_X) * (5.0 / 1023.0));
    y = analogRead((VR_Y) * (5.0 / 1023.0));
    Boton = digitalRead(SW);

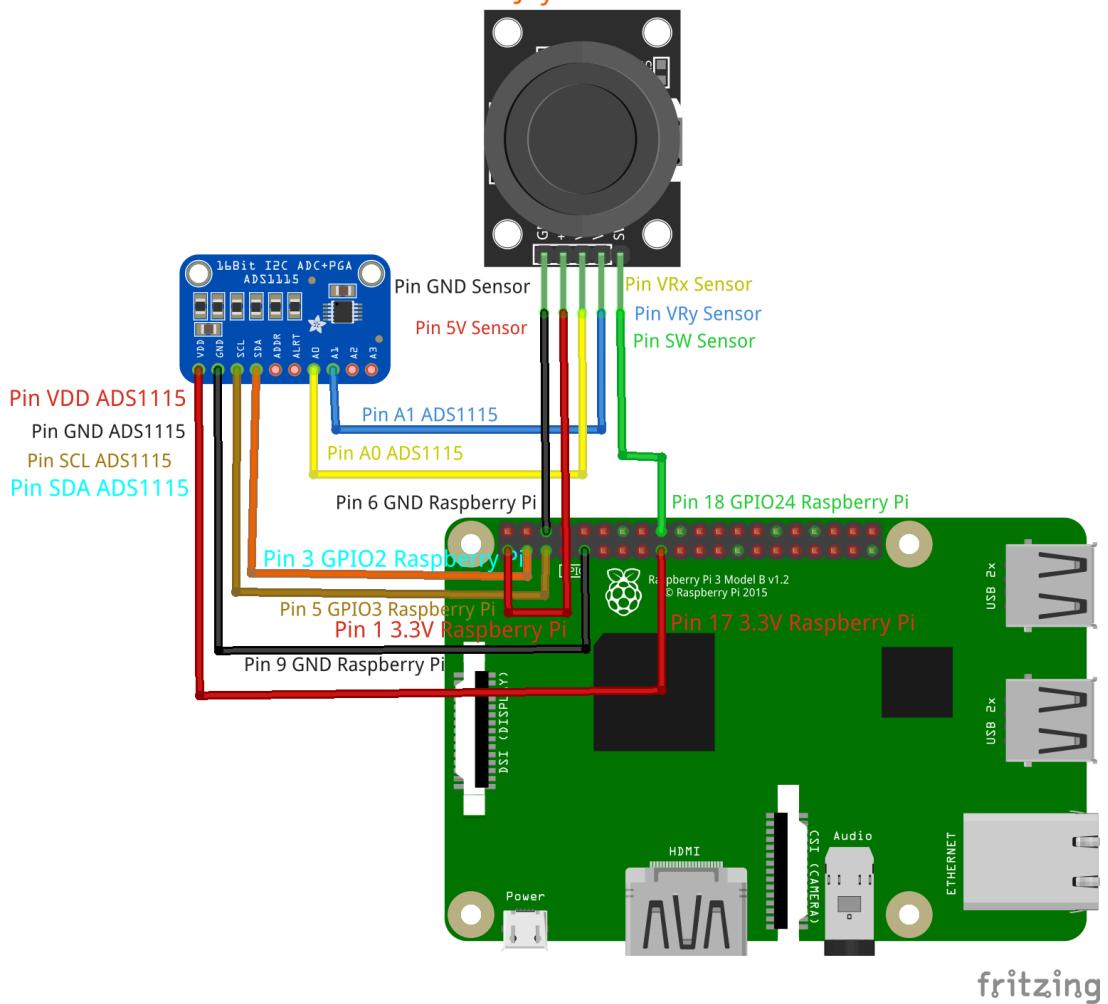
    Serial.print("X-axis:"); Serial.print(x, 4); Serial.print("V, ");
    Serial.print("Y-axis:"); Serial.print(y, 4); Serial.print("V, ");
    Serial.print("Boton:");

    if(Boton == 1)
    {
        Serial.println("no presionado");
    }
    else
    {
        Serial.println("presionado");
    }
    delay(200);
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

KY-023: Módulo Joystick dual axis



fritzing

KY-023	a	Raspberry Pi
GND	a	Pin 6 GND
+5V	a	Pin 1 3.3V
VRx	a	Pin A0 (ADS1115)
VRy	a	Pin A1 (ADS1115)
SW	a	Pin 18 GPIO24
VDD (ADS1115)	a	Pin 17 3.3V
GND (ADS1115)	a	Pin 9 GND
SCL (ADS1115)	a	Pin 5 GPIO3(Pin SCL en Raspberry Pi)
SDA (ADS1115)	a	Pin 3 GPIO2(Pin SDA en Raspberry Pi)

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep
import time, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1115 = 0x01
gain = 4096
sps = 64
adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3
adc = ADS1x15(ic=ADS1115)
Pin_SW = 24

GPIO.setup(Pin_SW, GPIO.IN, pull_up_down = GPIO.PUD_UP)

try:
    while True:
        x = adc.readADCSingleEnded(adc_channel_0, gain, sps)
        y = adc.readADCSingleEnded(adc_channel_1, gain, sps)

        if GPIO.input(Pin_SW) == True:
            print("X-axis: ", x, "mV, ", "Y-axis: ", y, "mV,
Pin_SW: no presionado")
        else:
            print("X-axis: ", x, "mV, ", "Y-axis: ", y, "mV,
Pin_SW: presionado")
            print("-----")
            button_pressed = False
            time.sleep(retraso)

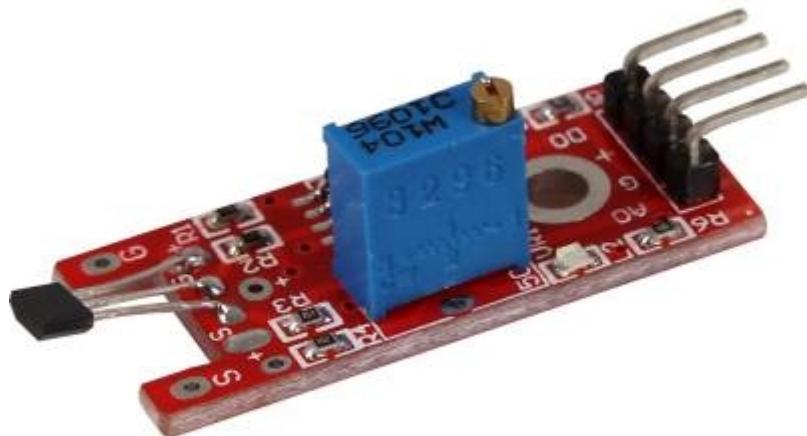
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-023.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-023.py*

## KY-024: Sensor lineal efecto Hall



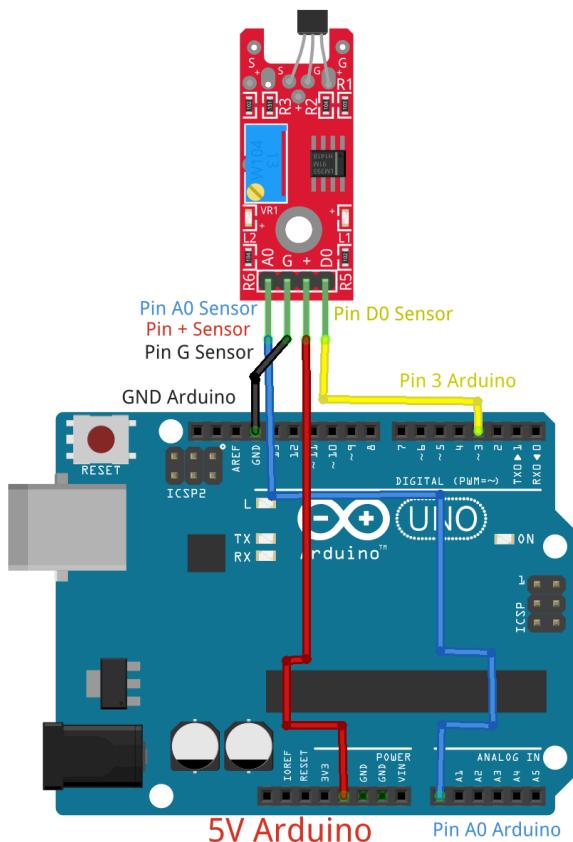
### Descripción:

Este sensor mide un campo magnético. Si existe la presencia de un campo magnético se creará una señal de alto(1 binario).

# **Luciano Rabassa**

## Conexión Arduino:

KY-024: Módulo Sensor linear efecto Hall



fritzing

KY-024	a	Arduino
A0	a	Pin Analógico A0
G	a	Pin GND
+	a	Pin 5V
D0	a	Pin Digital 3

## Código Arduino:

```
int Pin_A0 = A0;
int Pin_D0 = 3;

void setup()
{
    pinMode(Pin_A0, INPUT);
    pinMode(Pin_D0, INPUT);

    Serial.begin(9600);
}

void loop()
{
    float Analog;
    int Digital;

    Analog = analogRead((Pin_A0) * (5.0 / 1023.0));
    Digital = digitalRead(Pin_D0);

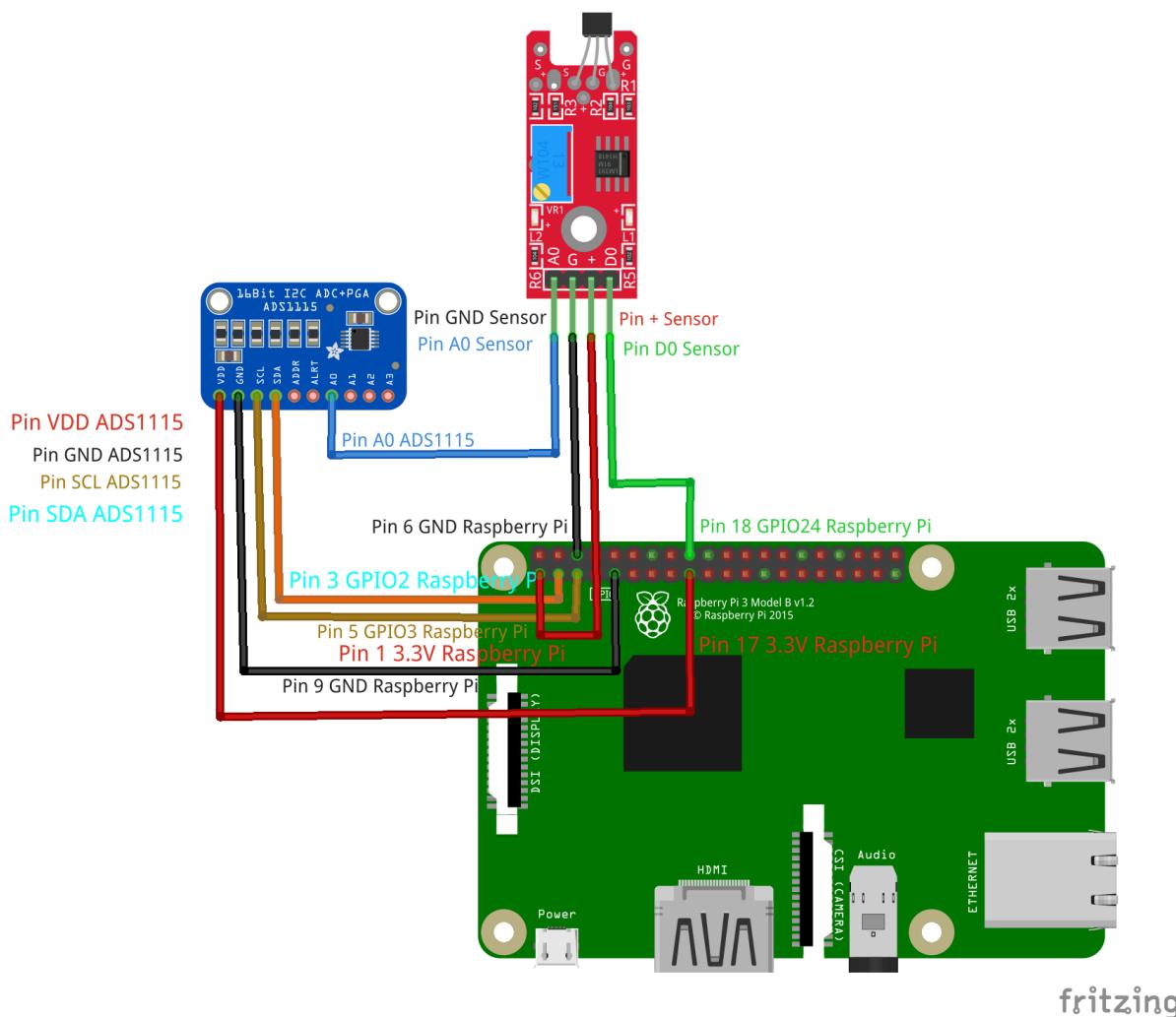
    Serial.print("Analog voltage value:");Serial.print(Analog,4);
    Serial.print("V, ");
    Serial.print("Valor extremo:");

    if(Digital == 1)
    {
        Serial.println(" alcanzado");
    }
    else
    {
        Serial.println(" no alcanzado");
    }
    Serial.println("-----");
    delay(200);
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

KY-024: Módulo Sensor linear efecto Hall



KY-024	a	Raspberry Pi
A0	a	Pin A0 (ADS1115)
G	a	Pin 6 GND
+	a	Pin 1 3.3V
D0	a	Pin 18 GPIO24
VDD (ADS1115)	a	Pin 17 3.3V
GND (ADS1115)	a	Pin 9 GND
SCL (ADS1115)	a	Pin 5 GPIO3 (Pin SCL en Raspberry Pi)
SDA (ADS1115)	a	Pin 3 GPIO2(Pin SDA en Raspberry Pi)

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep
import math, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1115 = 0x01    # 16-bit
gain = 4096
sps = 64
adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3
adc = ADS1x15(ic=ADS1115)
Digital = 24

GPIO.setup(Digital, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

try:
    while True:
        analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

        if GPIO.input(Digital) == False:
            print("Valor tension analoga", analog, "mV, ", "valor
extremo: no alcanzado")
        else:
            print("Tension analoga:", analog, "mV, ", "valor
extremo: alcanzado")
            print("-----")
            sleep(retraso)

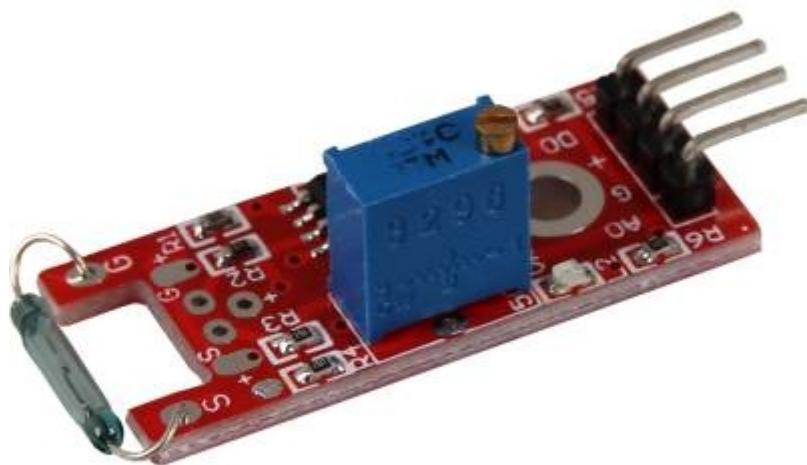
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-024.py

Para correr el programa abrimos la Terminal y tecleamos:

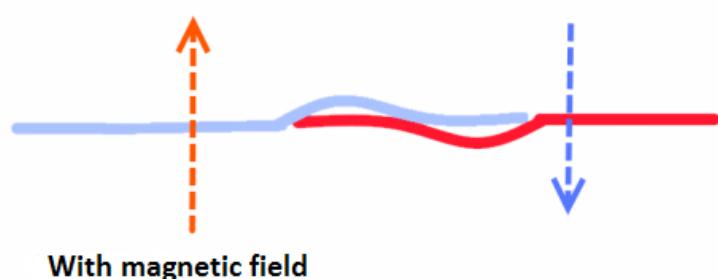
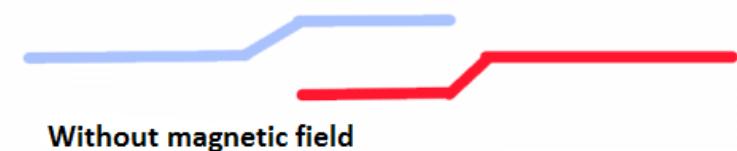
- *sudo python3 KY-024.py*

## KY-025: Módulo interruptor magnético reed



### Descripción:

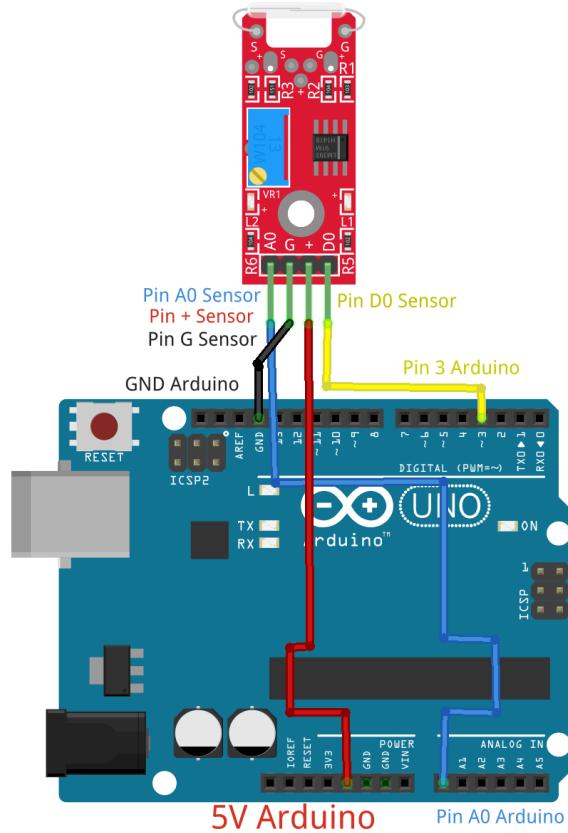
Interruptor magnético normalmente abierto (NO), que cierra su contacto (NC) al detectar un campo magnético cercano.



# Luciano Rabassa

## Conección Arduino:

KY-025: Módulo Interruptor reed



fritzing

KY-025	a	Arduino
A0	a	Pin Analógico A0
G	a	Pin GND
+	a	Pin 5V
D0	a	Pin Digital 3

## Código Arduino:

```
int Pin_A0 = A0;
int Pin_D0 = 3;

void setup()
{
    pinMode(Pin_A0, INPUT);
    pinMode(Pin_D0, INPUT);

    Serial.begin(9600);
}

void loop ()
{
    float Analog;
    int Digital;

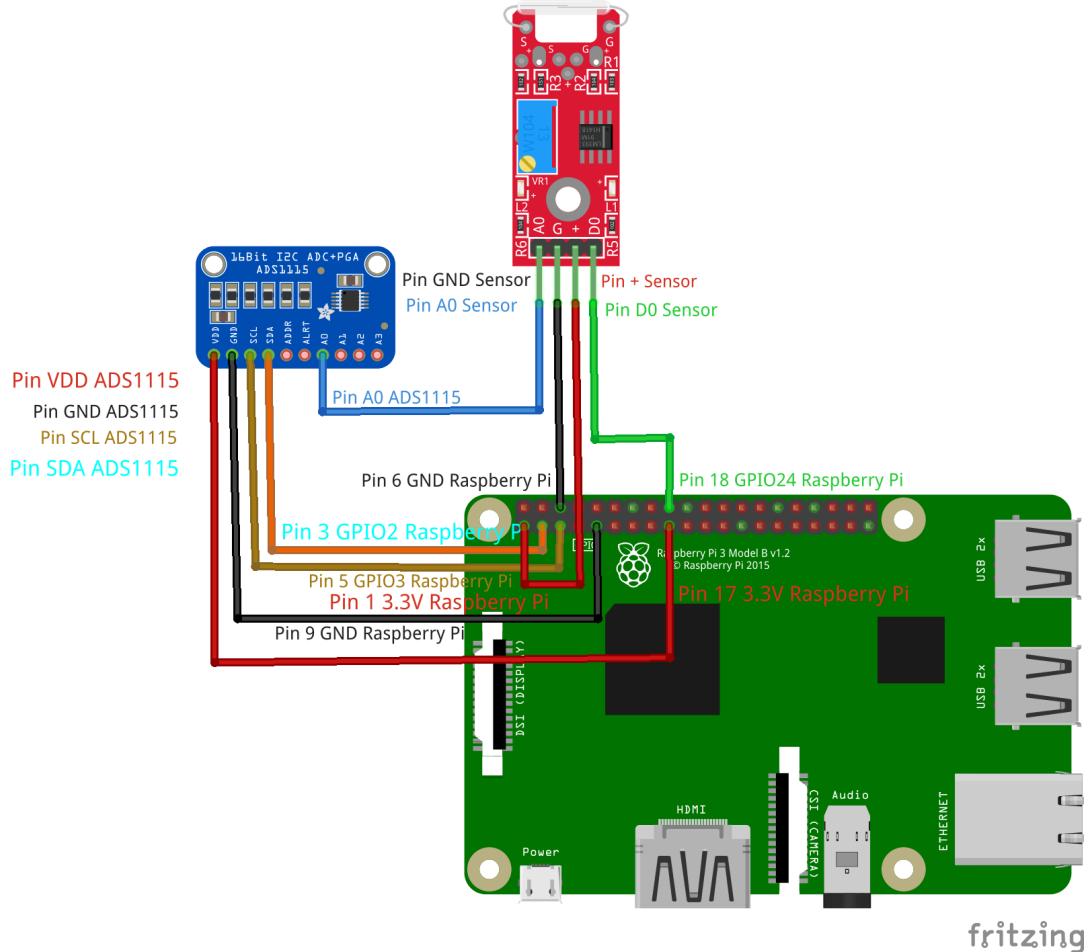
    Analog = analogRead(Pin_A0 * (5.0 / 1023.0));
    Digital = digitalRead(Pin_D0);

    Serial.print("Valor de tension
analogica:"); Serial.print(Analog,4); Serial.print("V, ");
    Serial.print("Valor extremo: ");

    if(Digital == 1)
    {
        Serial.println(" alcanzado");
    }
    else
    {
        Serial.println(" no alcanzado");
    }
    Serial.println("-----");
    delay(200);
}
```

## Conexión Raspberry Pi:

KY-025: Módulo Interruptor magnético



fritzing

KY-025	a	Raspberry Pi
A0	a	Pin A0 (ADS1115)
G	a	Pin 6 GND
+	a	Pin 1 3.3V
D0	a	Pin 18 GPIO24
VDD (ADS1115)	a	Pin 17 3.3V
GND (ADS1115)	a	Pin 9 GND
SCL (ADS1115)	a	Pin 5 GPIO3(Pin SCL en Raspberry Pi)
SDA (ADS1115)	a	Pin 3 GPIO2(Pin SDA )

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep
import math, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1115 = 0x01
gain = 4096
sps = 64
adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3
adc = ADS1x15(ic=ADS1115)
Digital = 24

GPIO.setup(Digital, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

try:
    while True:
        analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

        if GPIO.input(Digital) == False:
            print("Valor tension analoga:", analog, "mV, ", "valor extremo: no alcanzado")
        else:
            print("Valor tension analoga:", analog, "mV, ", "Valor extremo: alcanzado")
            print("-----")
            sleep(retraso)

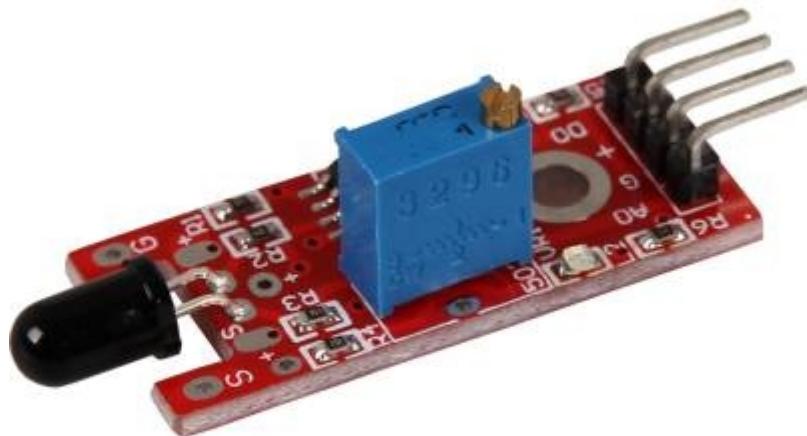
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-025.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-025.py*

## KY-026: Módulo sensor de llama



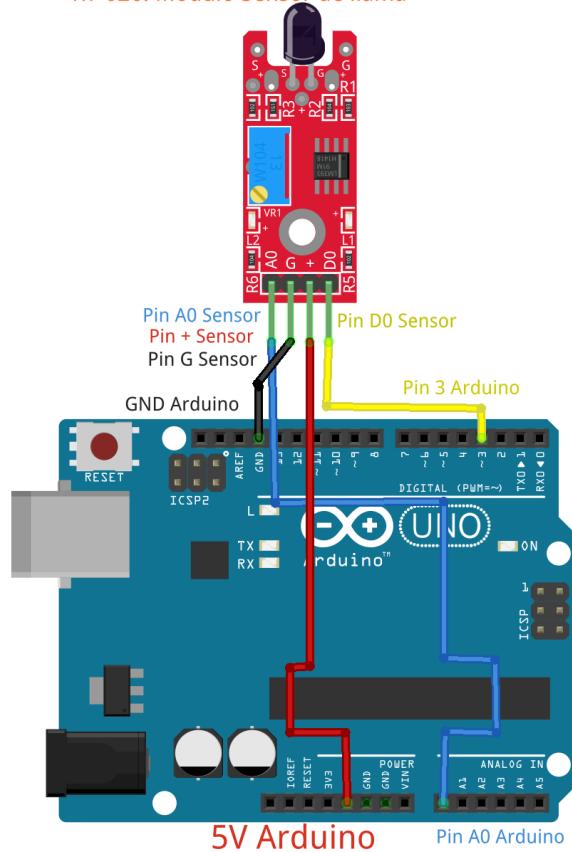
### Descripción:

Mediante un fotodiodo que es sensible al espectro luminoso de la llama, actúa enviando una señal por la salida digital, mientras que la salida analógica se encarga de medir directamente el fotodiodo. El LED 1, situado del lado del D0, nos indicará que está encendido, mientras que LED 2, situado del lado A0, nos indicará que ha detectado una llama, podemos probarlo acercando la flama de un encendedor al fotodiodo.

# Luciano Rabassa

## Conección Arduino:

KY-026: Módulo Sensor de llama



fritzing

KY-026	a	Arduino
A0	a	Pin Análogo A0
G	a	Pin GND
+	a	Pin 5V
D0	a	Pin Digital 3

## Código Arduino:

```
int Pin_A0 = A0;
int Pin_D0 = 3;

void setup()
{
    pinMode(Pin_A0, INPUT);
    pinMode(Pin_D0, INPUT);

    Serial.begin(9600);
}

void loop()
{
    float Analog;
    int Digital;

    Analog = analogRead(Pin_A0 * (5.0 / 1023.0));
    Digital = digitalRead(Pin_D0);

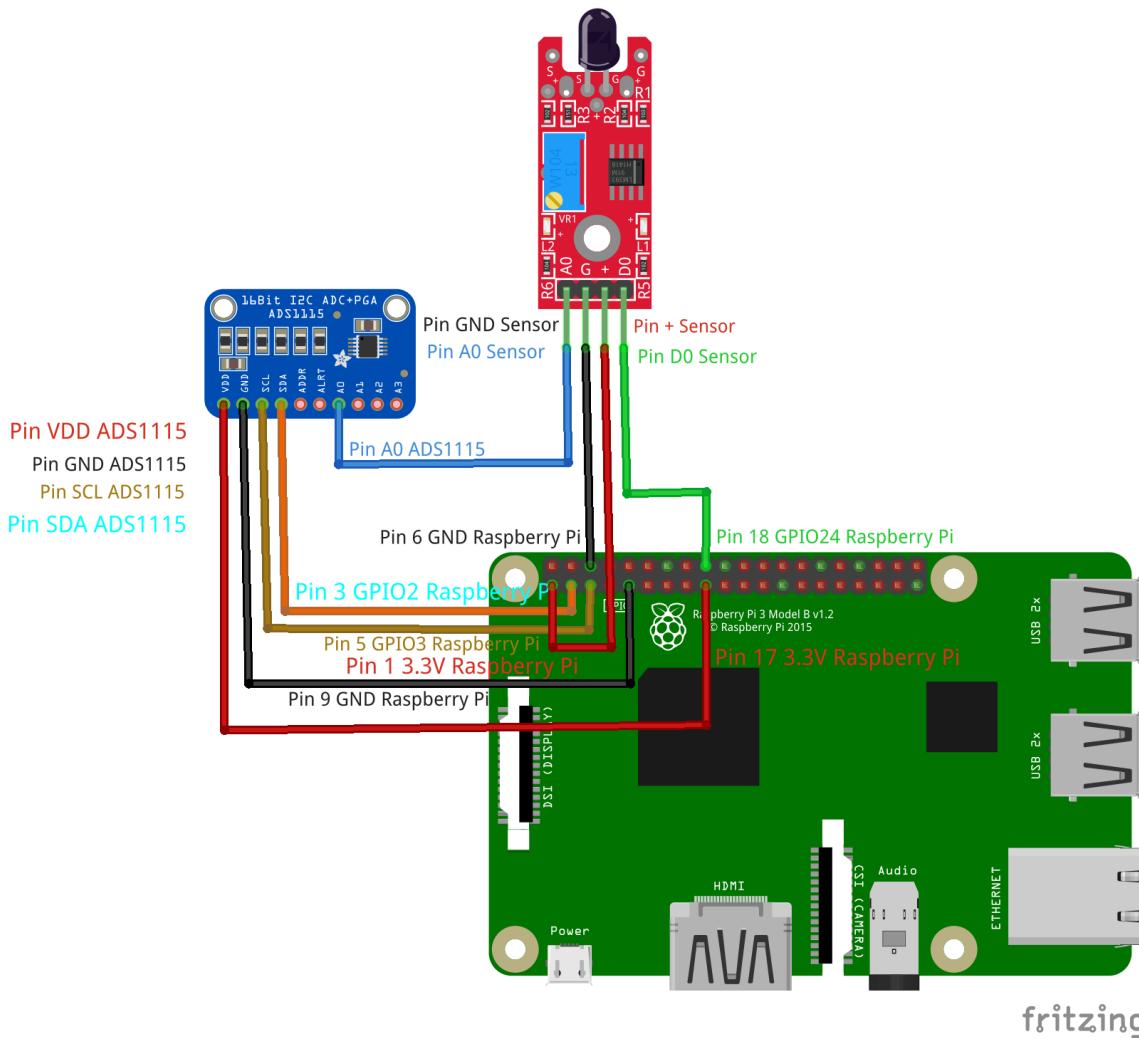
    Serial.print("Valor de tension analoga:");
    Serial.print(Analog, 4);
    Serial.print("V, ");
    Serial.print("Valor extremo: ");

    if(Digital == 1)
    {
        Serial.println("Alcanzado");
    }
    else
    {
        Serial.println("No alcanzado");
    }
    Serial.println("-----");
    delay(200);
}
```

# Luciano Rabassa

## Conección Raspberry Pi:

KY-026: Módulo Sensor de llama



fritzing

KY-026	a	Raspberry Pi
A0	a	A0 (ADS1115)
G	a	Pin 6 GND
+	a	Pin 1 3.3V
D0	a	Pin 18 GPIO24
VDD	a	Pin 17 3.3V
GND	a	Pin 9 GND
SCL	a	Pin 5 GPIO3(Pin SCL)
SDA	a	Pin 3 GPIO2(Pin SDA)

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep
import math, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1115 = 0x01 # 16-bit
gain = 4096 # +/- 4.096V
sps = 64
adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3
adc = ADS1x15(ic=ADS1115)
Digital = 24

GPIO.setup(Digital, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

try:
    while True:
        analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

        if GPIO.input(Digital) == False:
            print("Valor tension analoga:", analog, "mV, ", "valor extremo: no alcanzado")
        else:
            print("Valor tension analoga:", analog, "mV, ", "valor extremo: alcanzado")
            sleep(retraso)

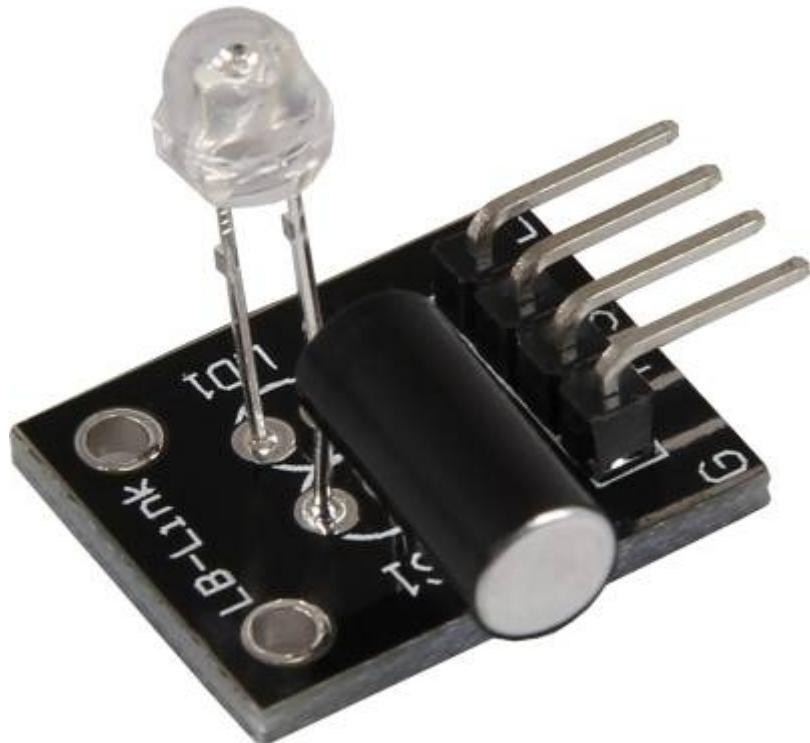
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-026.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-026.py*

## KY-027: Módulo taza mágica o Interruptor de mercurio con Led THT



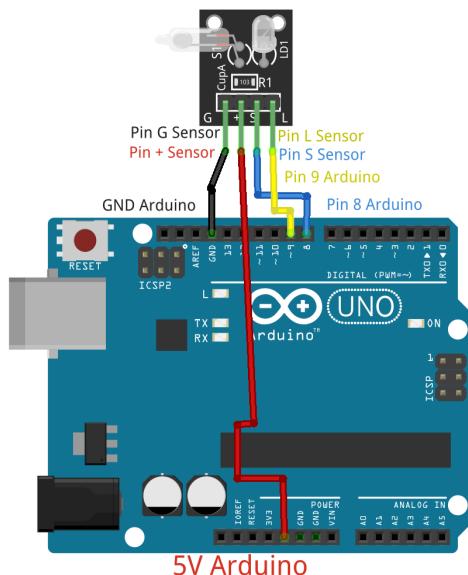
### Descripción:

Se trata de un interruptor de inclinación mediante gota o bola de mercurio, el cual encenderá el Led al inclinar el módulo.

# Luciano Rabassa

## Conección Arduino:

KY-027: Módulo Taza mágica de luz o Interruptor de mercurio con Led



fritzing

KY-027	a	Arduino
G	a	Pin GND
+	a	Pin 5V
S	a	Pin Digital 8
L	a	Pin Digital 9

## Código Arduino:

```
int Pin_L = 9;
int Pin_S = 8;
int valor;

void setup()
{
    pinMode(Pin_L, OUTPUT);
    pinMode(Pin_S, INPUT);
    digitalWrite(Pin_S, HIGH);
}

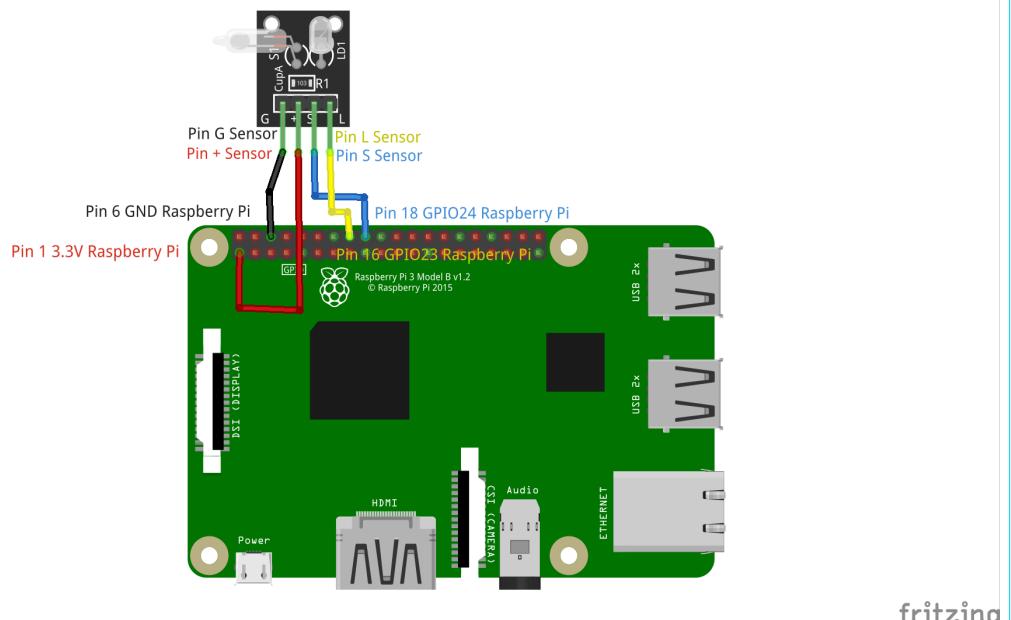
void loop()
{
    valor = digitalRead(Pin_S);
```

# Luciano Rabassa

```
if(valor == HIGH)
{
    digitalWrite(Pin_L, LOW);
}
else
{
    digitalWrite(Pin_L, HIGH);
}
```

## Conexión Raspberry Pi:

KY-027: Módulo Taza mágica de luz o Interruptor de mercurio con Led



fritzing

KY-027	a	Raspberry Pi
G	a	Pin 6 GND
+	a	Pin 1 3.3V
S	a	Pin 18 GPIO24
L	a	Pin 16 GPIO23

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

LED = 24
Pin_L = 23

GPIO.setup(Pin_L, GPIO.IN)
GPIO.setup(LED, GPIO.OUT)

print("Prueba del modulo [Presiona Ctrl + C para finalizar la prueba] ")

def funcionMagic(null):
    GPIO.output(LED, True)

GPIO.add_event_detect(Pin_L, GPIO.FALLING, callback =
funcionMagic, bouncetime = 10)

try:
    while True:
        time.sleep(1)

        if GPIO.input(Pin_L):
            GPIO.output(LED, False)

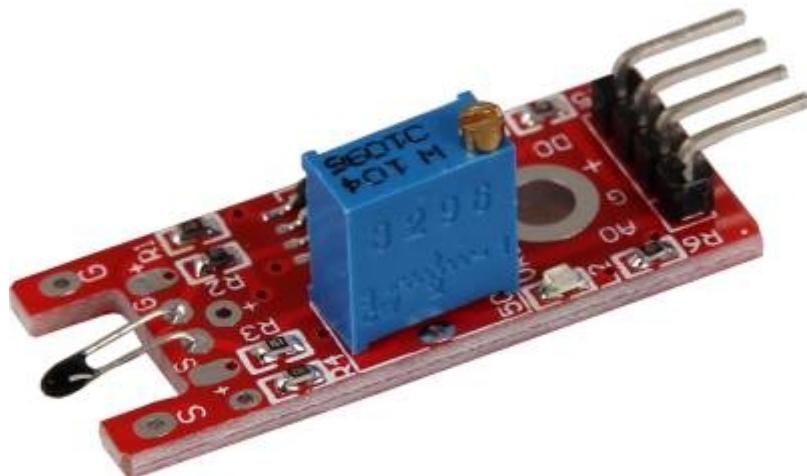
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-027.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-027.py*

## KY-028: Módulo sensor digital de temperatura, Termistor



### Descripción:

El sensor de temperatura digital es activado al detectar una variación en la temperatura, cuando esto sucede el LED se enciende, de lo contrario el LED permanecerá apagado.

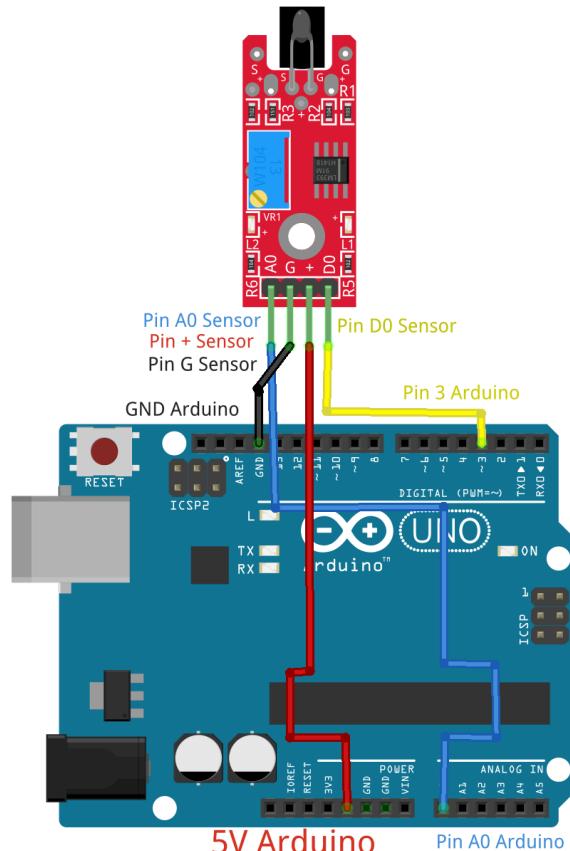
### Características del sensor.

- Corriente: 30 ~ 60 mA
- Corriente de Pulso: 0.3 a 1 A
- Voltaje Inverso: 4.5 a 24 V
- Potencia de disipación: 90 mW
- Rango de Temperatura: -25 a +80°C
- Temperatura de Almacenamiento: -40 a +100°C

# Luciano Rabassa

## Conección Arduino:

KY-028: Módulo Sensor digital de temperatura termistor



fritzing

KY-028	a	Arduino
A0	a	Pin Analógico A0
G	a	Pin GND
+	a	Pin 5V
D0	a	Pin Digital 3

## Código Arduino:

```
int Pin_A0 = A0;
int Pin_D0 = 3;

void setup()
{
    pinMode(Pin_A0, INPUT);
    pinMode(Pin_D0, INPUT);

    Serial.begin(9600);
}

void loop()
{
    float Analog;
    int Digital;

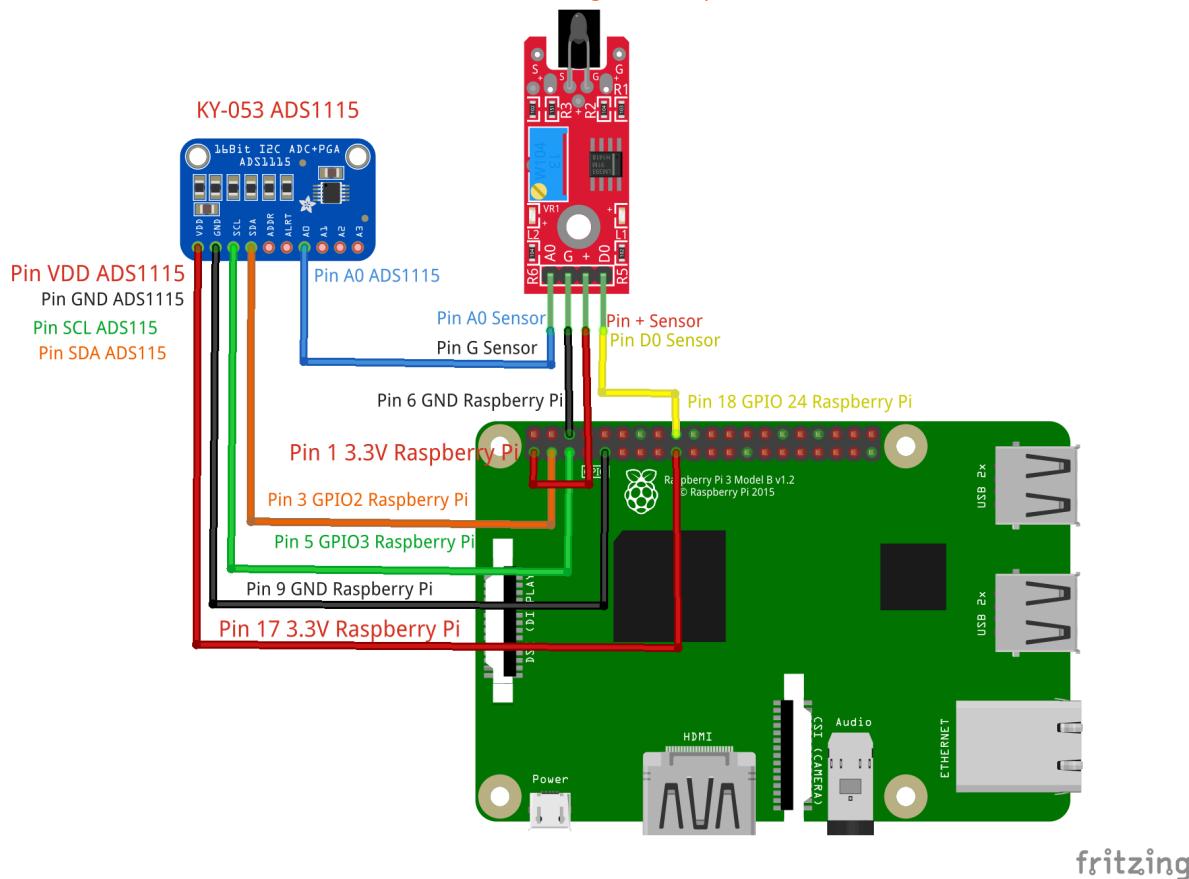
    Analog = analogRead(Pin_A0 * (5.0 / 1023.0));
    Digital = digitalRead(Pin_D0);

    Serial.print("Valor de tension analoga:");
    Serial.print(Analog, 4);
    Serial.print("V, ");
    Serial.print("Valor extremo:");

    if(Digital == 1)
    {
        Serial.println(" alcanzado");
    }
    else
    {
        Serial.println(" no alcanzado");
    }
    Serial.println("-----");
    delay (200);
}
```

## Conexión Raspberry Pi:

KY-028: Módulo Sensor digital de temperatura termistor



fritzing

KY-028	a	Raspberry Pi
A0	a	Pin A0 (ADS1115)
G	a	Pin 6 GND
+	a	Pin 1 3.3V
D0	a	Pin 18 GPIO24
VDD	a	Pin 17 3.3V
GND	a	Pin 9 GND
SCL	a	Pin 5 GPIO3(Pin SCL en Raspberry Pi)
SDA	a	Pin 3 GPIO2(Pin SDA en Raspberry Pi)

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep
import math, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1115 = 0x01
gain = 4096
sps = 64

adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3

adc = ADS1x15(ic=ADS1115)

Digital = 24
GPIO.setup(Digital, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

try:
    while True:
        analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)
        if GPIO.input(Digital) == False:
            print("Tensionanalogica:",analog,"mV, ","Valor extremo: no alcanzado")
        else:
            print("Tensionanalogica:", analog, "mV, ", "Valor extremo: alcanzado")
        sleep(retraso)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-028.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-028.py*

## KY-029: Módulo Led THT 3mm Bi-Color



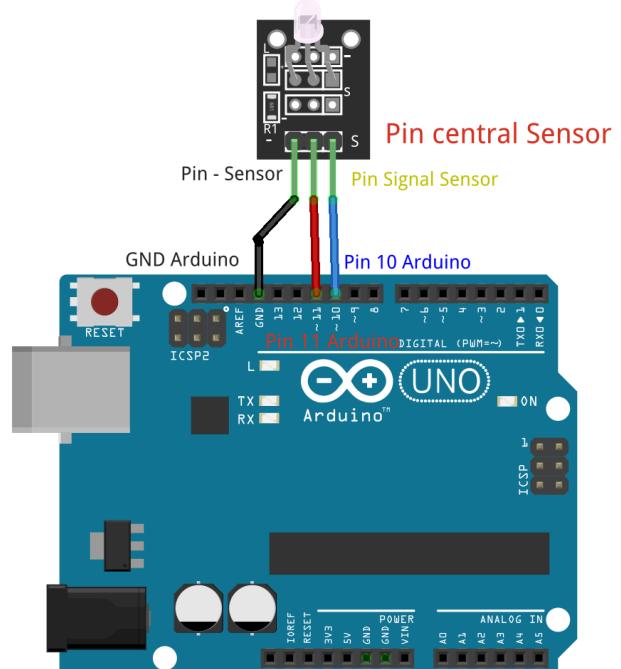
### Descripción:

Led de colores rojo y verde con cátodo común. En la programación Raspberry Pi introduciremos la librería `gpiozero` y su función `LED`.

# Luciano Rabassa

## Conección Arduino:

KY-029: Módulo Led Bicolor 3mm



KY-029	a	Arduino
-	a	Pin GND
VCC (Verde)	a	Pin Digital 11
S (Rojo)	a	Pin Digital 10

## Código Arduino:

```
int rojo = 10;
int verde = 11;

void setup()
{
    pinMode(rojo, OUTPUT);
    pinMode(verde, OUTPUT);
}

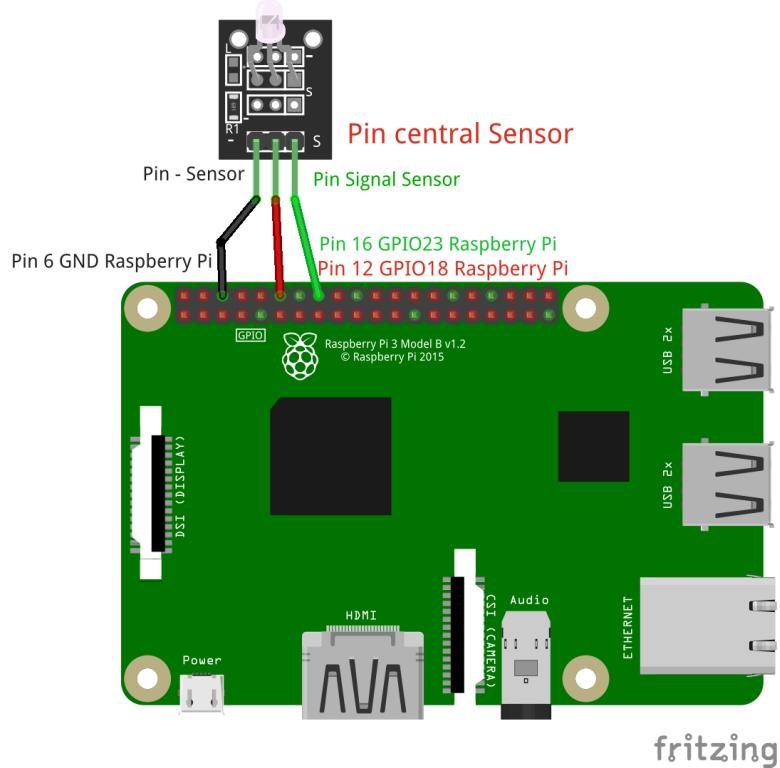
void loop()
{
    digitalWrite(rojo, HIGH);
    digitalWrite(verde, LOW);
    delay(3000);

    digitalWrite(rojo, LOW);
    digitalWrite(verde, HIGH);
    delay(3000);
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

KY-029: Módulo Led Bicolor 3mm



KY-029	a	Raspberry Pi
-	a	Pin 6 GND
VCC (Verde)	a	Pin 12 GPIO18
S (Rojo)	a	Pin 16 GPIO23

## Código Raspberry Pi:

```
#Importamos la librería gpiozero y su función LED, además la función sleep de time
from gpiozero import LED
from time import sleep

#Declaramos ambos leds y a cuales esta conectado, lo tomamos como BCM:GPIO23 y GPIO18
rojo = LED(23)
verde = LED(18)

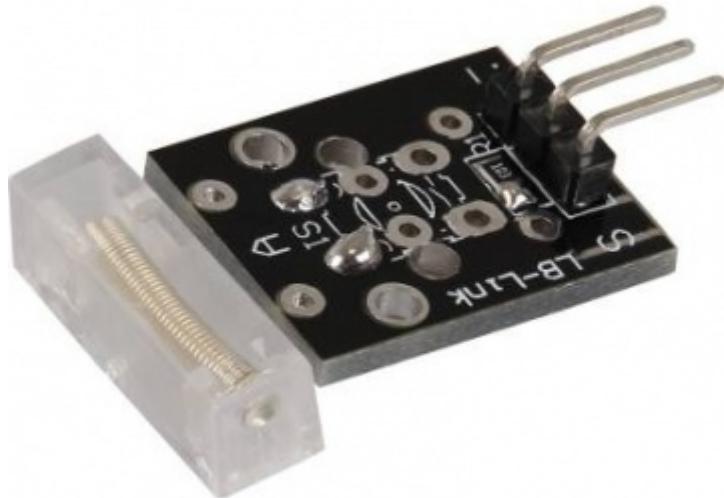
#Como siempre nuestro bucle
while True:
    rojo.on()
    sleep(1)
    rojo.off()
    sleep(1)
    verde.on()
    sleep(1)
    verde.off()
    sleep(1)
```

Guardamos el programa con el nombre KY-029.py

Para correr el programa abrimos la Terminal y tecleamos:

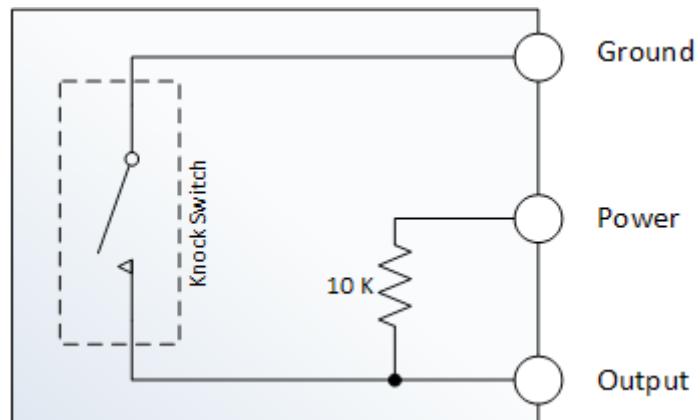
- *sudo python3 KY-029.py*

## KY-031: Módulo Sensor de golpe



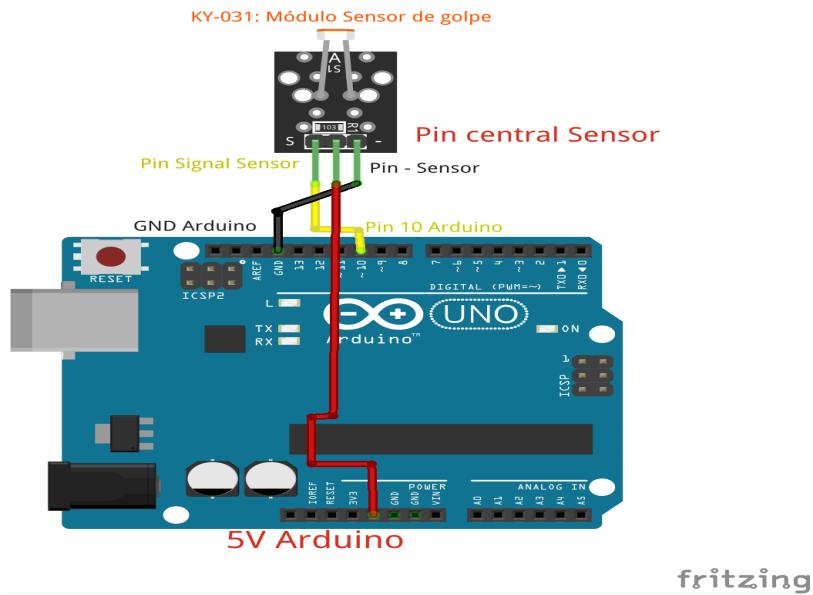
### Descripción:

Cuando detecta vibraciones o golpes, los dos contactos de entrada se cierran enviando una señal HIGH o 1 al S. Contiene una resistencia pull-up de  $10K\Omega$ .



# Luciano Rabassa

## Conección Arduino:



fritzing

KY-031	a	Arduino
Signal	a	Pin Digital 3
VCC	a	Pin 5V
-	a	Pin GND

## Código Arduino:

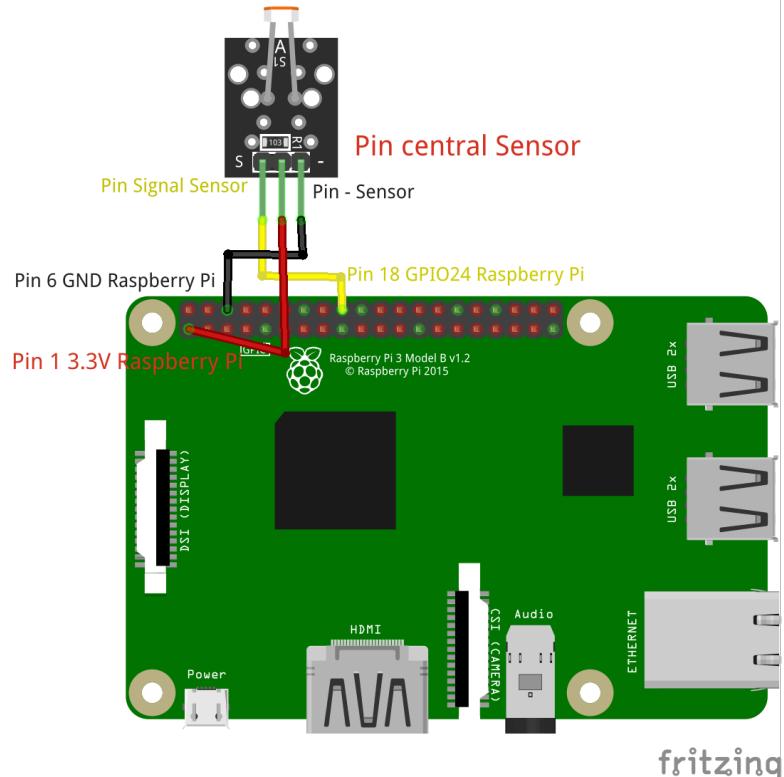
```
int Pin_Signal = 3;
int valor;

void setup()
{
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(Pin_Signal, INPUT);
}

void loop(){
    valor = digitalRead(Pin_Signal);
    if(valor == HIGH){
        digitalWrite(LED_BUILTIN, LOW);
    }
    else
    {
        digitalWrite(LED_BUILTIN, HIGH);
    }
}
```

## Conexión Raspberry Pi:

KY-031: Módulo Sensor de golpe



KY-031	a	Raspberry Pi
Signal	a	Pin 18 GPIO24
VCC	a	Pin 1 3.3V
-	a	Pin 6 GND

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_Pin_Signal = 24
GPIO.setup(GPIO_Pin_Signal, GPIO.IN)

print("Prueba del modulo [Presiona Ctrl + C para finalizar la prueba] ")

def funcionDetectar(null):
    print("Ha sido detectado")

GPIO.add_event_detect(GPIO_Pin_Signal, GPIO.FALLING, callback =
funcionDetectar, bouncetime = 100)

try:
    while True:
        time.sleep(1)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-031.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-031.py*

## KY-032: Módulo Sensor de obstáculos



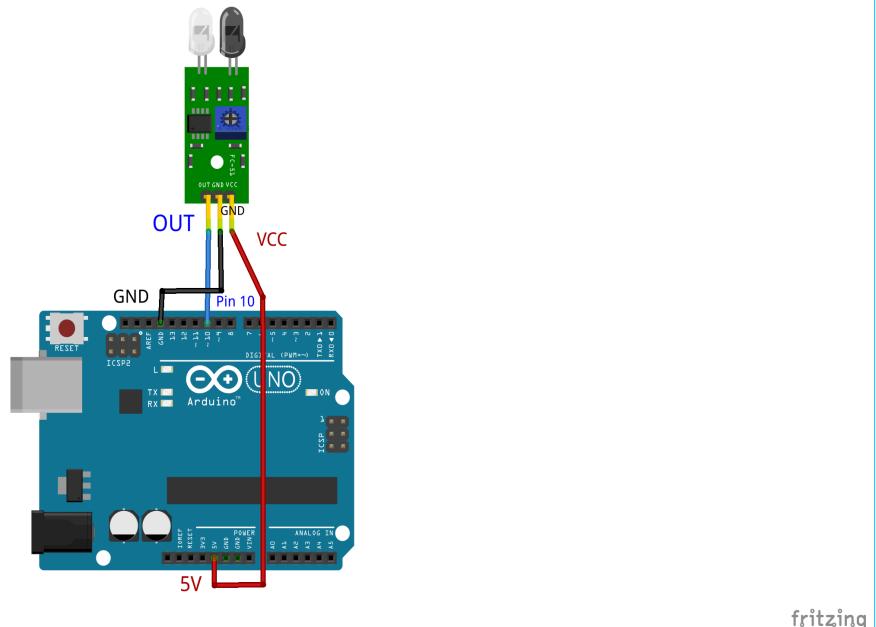
### Descripción:

Contiene un diodo emisor y uno receptor que devuelve una señal, cuando detecta un obstáculo dentro de su rango de medición entre 2 y 40 centímetros. Este módulo dispone de 4 pines, lo que le permite habilitar o deshabilitar la detección.

- Tensión de alimentación: 3.3V-5V
- Consumo:  $\geq 20\text{mA}$
- Rango de operación:  $-10\text{ }^{\circ}\text{C} - +50\text{ }^{\circ}\text{C}$
- Distancia de detección :2-40cm
- Señales: (GND / VCC/ OUT / EN)
- Señal OUT: Nivel bajo hay un obstáculo.
- Ángulo de medición:  $35^{\circ}$

## Conección Arduino:

### KY-032: Módulo Sensor de obstáculos



fritzing

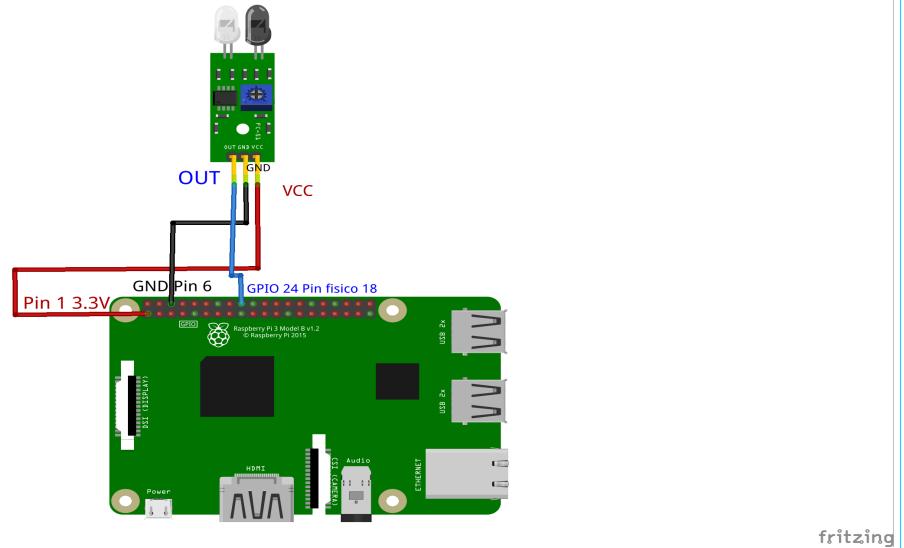
KY-032	a	Arduino
GND	a	Pin GND
V+	a	Pin 5V
OUT	a	Pin Digital 10

## Código Arduino:

```
int Pin_OUT = 10;
void setup(){
  Serial.begin(9600);
  pinMode(Pin_OUT, INPUT);
}
void loop(){
  bool valor = digitalRead(Pin_OUT);
  if (valor == HIGH){
    Serial.println("Sin obstaculos");
  } else {
    Serial.println("Obstaculo detectado");
  }
  Serial.println("-----");
  delay(500);
}
```

## Conexión Raspberry Pi:

KY-032: Módulo Sensor de obstáculos



fritzing

KY-032	a	Raspberry Pi
GND	a	Pin 6 GND
V+	a	Pin 1 3.3V
OUT	a	Pin 18 GPIO24

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_Pin_Signal = 24
GPIO.setup(GPIO_Pin_Signal, GPIO.IN, pull_up_down = GPIO.PUD_UP)

retraso = 0.5

print("Prueba del modulo [Presiona Ctrl + C para finalizar la prueba]")

try:
    while True:
        if GPIO.input(GPIO_Pin_Signal) == True:
            print("Sin obstaculo")
        else:
            print("Obstaculo detectado")
        time.sleep(retraso)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-032.py

Para correr el programa abrimos la Terminal y tecleamos:

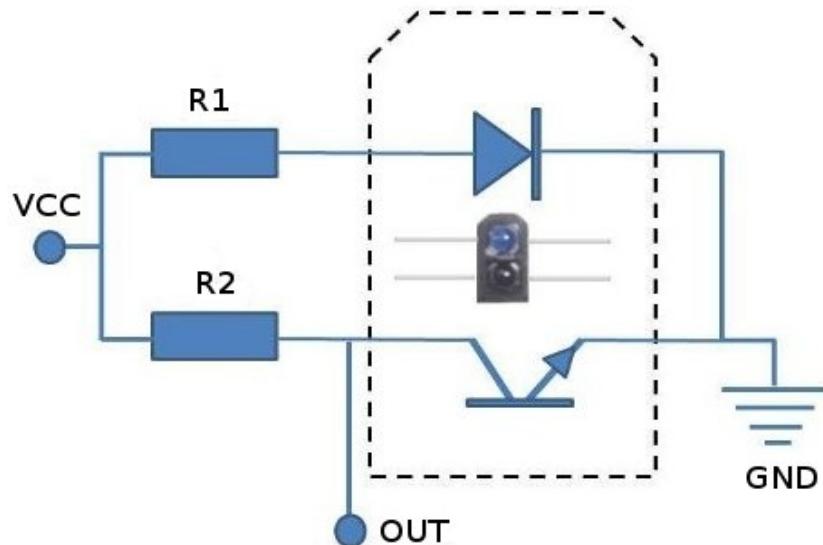
- *sudo python3 KY-032.py*

## KY-033: Módulo Sigue líneas



### Descripción:

Mediante sus fototransistores, detecta si la luz es reflejada o absorbida. Está basado en el sensor reflectivo TCRT5000:

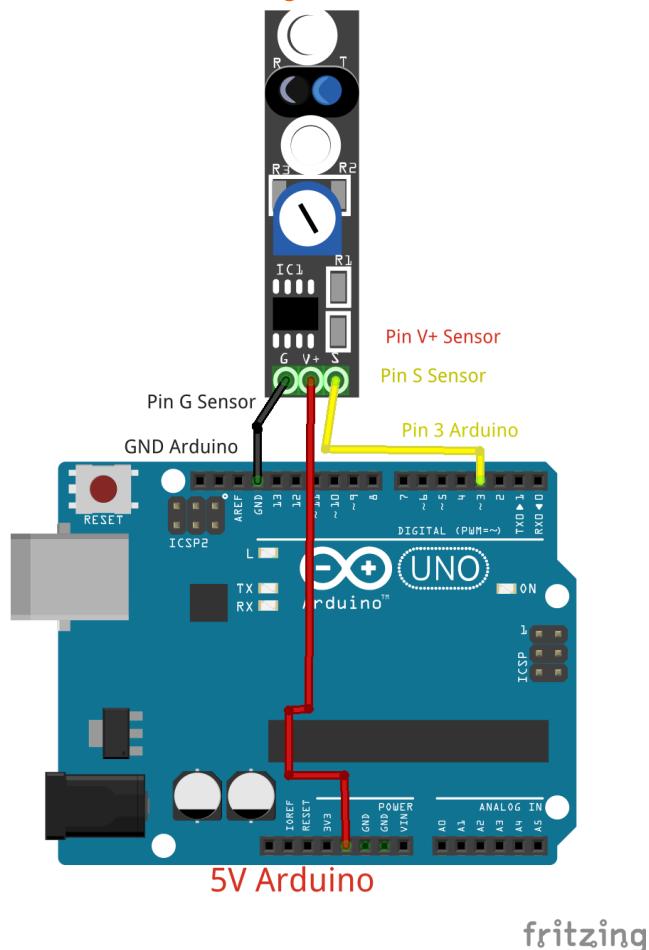


Datasheet: <https://www.vishay.com/docs/83760/tcrc5000.pdf>

# Luciano Rabassa

## Conección Arduino:

KY-033: Módulo Seguidor de línea



fritzing

KY-033	a	Arduino
G	a	Pin GND
V+	a	Pin 5V
S	a	Pin Digital 3

## Código Arduino:

```
int Pin_S = 3;

void setup()
{
    Serial.begin(9600);
    pinMode(Pin_S, INPUT);
}

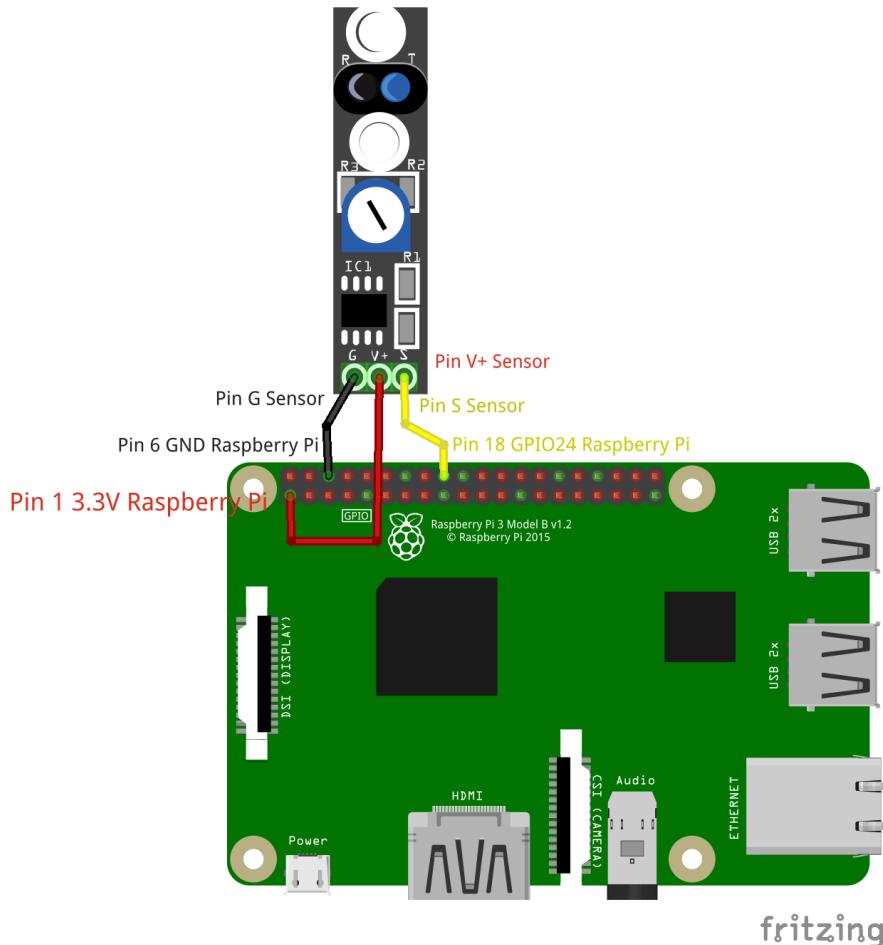
void loop()
{
    bool valor = digitalRead(Pin_S);

    if (valor == HIGH)
    {
        Serial.println("Esta en la linea");
    }
    else
    {
        Serial.println("No esta en la linea");
    }

    delay(500);
}
```

## Conexión Raspberry Pi:

KY-033: Módulo Seguidor de línea



fritzing

KY-033	a	Raspberry Pi
G	a	Pin 6 GND
V+	a	Pin 1 3.3V
S	a	Pin 18 GPIO24

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_Pin_Signal = 24
GPIO.setup(GPIO_Pin_Signal, GPIO.IN, pull_up_down = GPIO.PUD_UP)

retraso = 0.5

print("Prueba del modulo [Presiona Ctrl + C para finalizar la prueba] ")

try:
    while True:
        if GPIO.input(GPIO_Pin_Signal) == True:
            print("Esta en la linea")
        else:
            print("No esta en la linea")
        time.sleep(retraso)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-033.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-033.py*

## KY-034: Módulo Led THT 5mm Flash de 7 colores

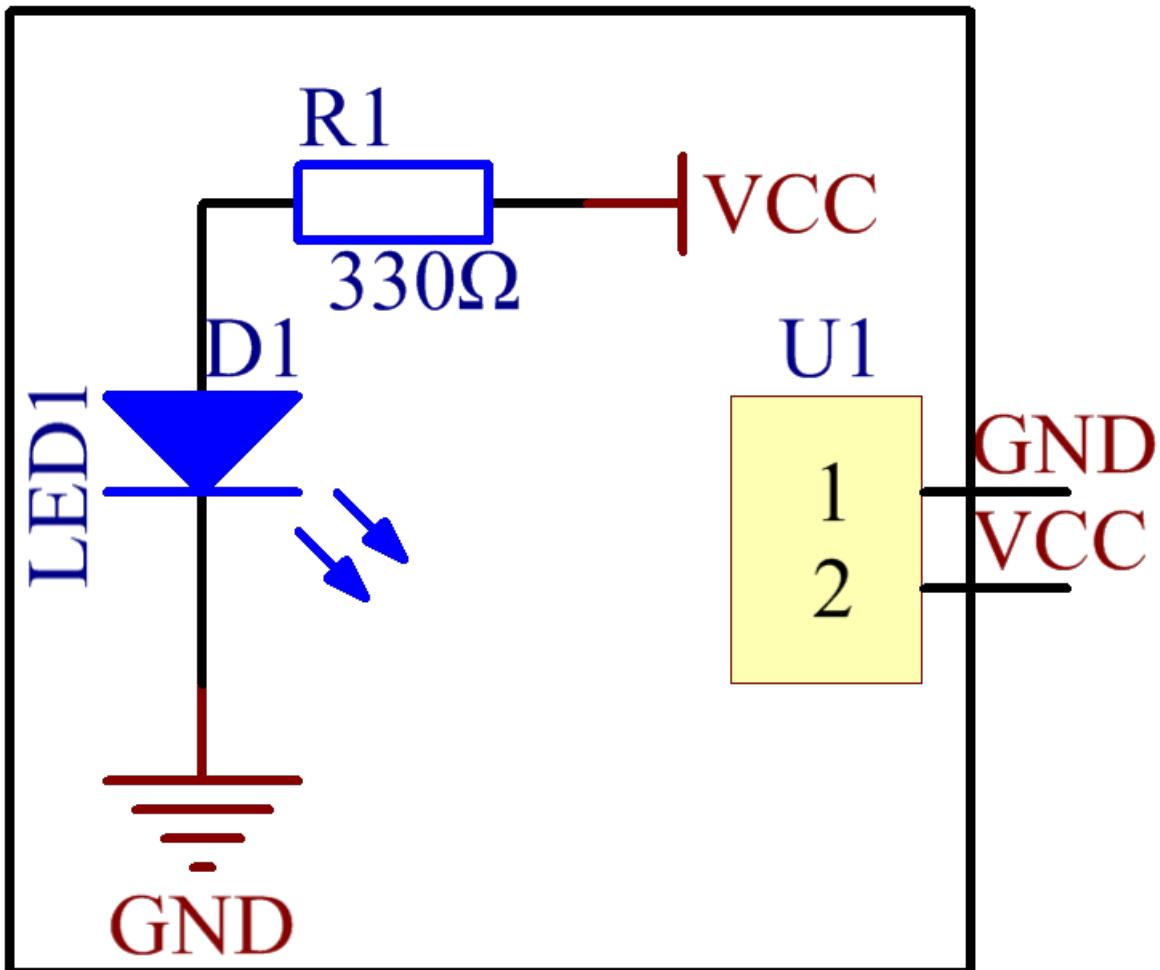


### Descripción:

Posee un LED THT de 5 mm que genera 7 colores con parpadeo flash. Posee una resistencia de  $330\Omega$ .

- **Voltaje de funcionamiento:** 5V
- **Tipo de Producto:** LED
- **Modelo del producto:** YB-3120B4PnYG-PM
- **Forma:** Redondo Tipo de LED DIP 5mm
- **Grupo de colores:** verde amarillo rosa (alto brillo).
- **Tipo de lente:** niebla blanca
- **Voltaje estandar:** 3.0 – 4 .5V

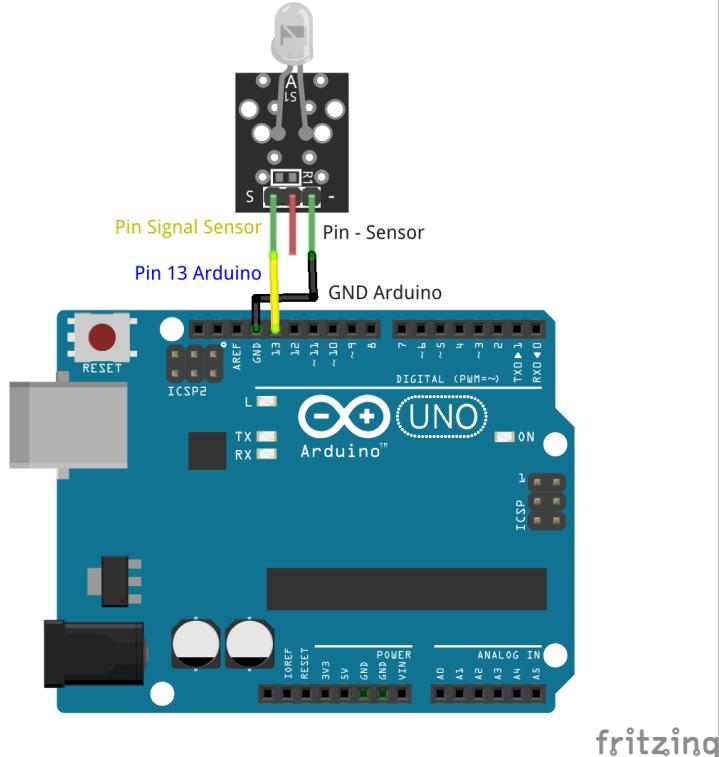
Esquemático:



# Luciano Rabassa

## Conección Arduino:

KY-034: Módulo Led THT Flash de 7 colores



fritzing

KY-034	a	Arduino
Signal	a	Pin Digital 13
VCC	a	NO SE CONECTA
-	a	Pin GND

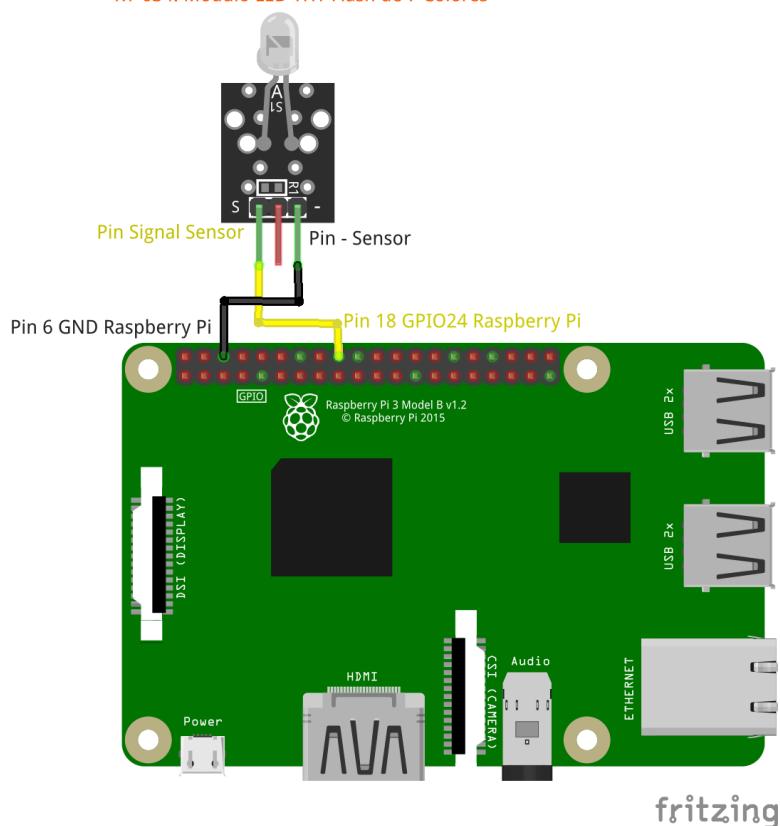
# Luciano Rabassa

## Código Arduino:

```
int Pin_Signal = 13;  
  
void setup()  
{  
    pinMode(Pin_Signal, OUTPUT);  
}  
  
void loop()  
{  
    digitalWrite(Pin_Signal, HIGH);  
    delay(4000);  
    digitalWrite(Pin_Signal, LOW);  
    delay(2000);  
}
```

## Conexión Raspberry Pi:

KY-034: Módulo LED THT Flash de 7 Colores



# Luciano Rabassa

KY-034	a	Raspberry Pi
Signal	a	Pin 18 GPIO24
VCC	a	NO SE CONECTA
-	a	Pin 6 GND

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

Pin_Signal = 24
GPIO.setup(Pin_Signal, GPIO.OUT, initial = GPIO.LOW)

print("Prueba del modulo [Presiona Ctrl + C para finalizar la prueba]")

try:
    while True:
        print("LED encendido por 4 segundos")
        GPIO.output(Pin_Signal, GPIO.HIGH)
        time.sleep(4)
        print("LED apagado por 2 segundos")
        GPIO.output(Pin_Signal, GPIO.LOW)
        time.sleep(2)

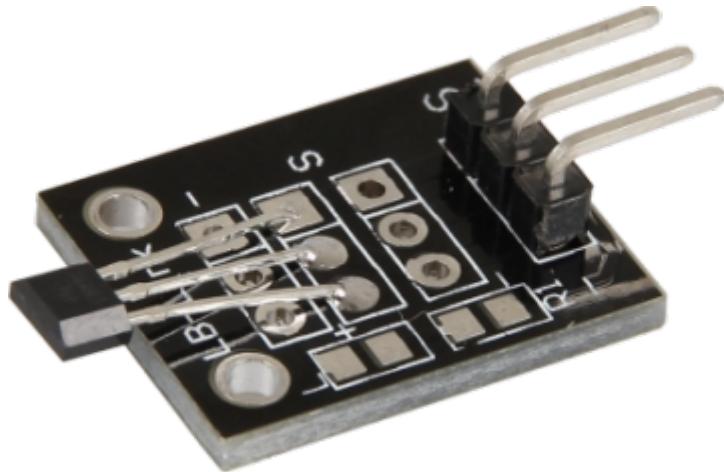
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-034.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-034.py*

## KY-035: Módulo sensor magnético analógico Bihor



### Descripción:

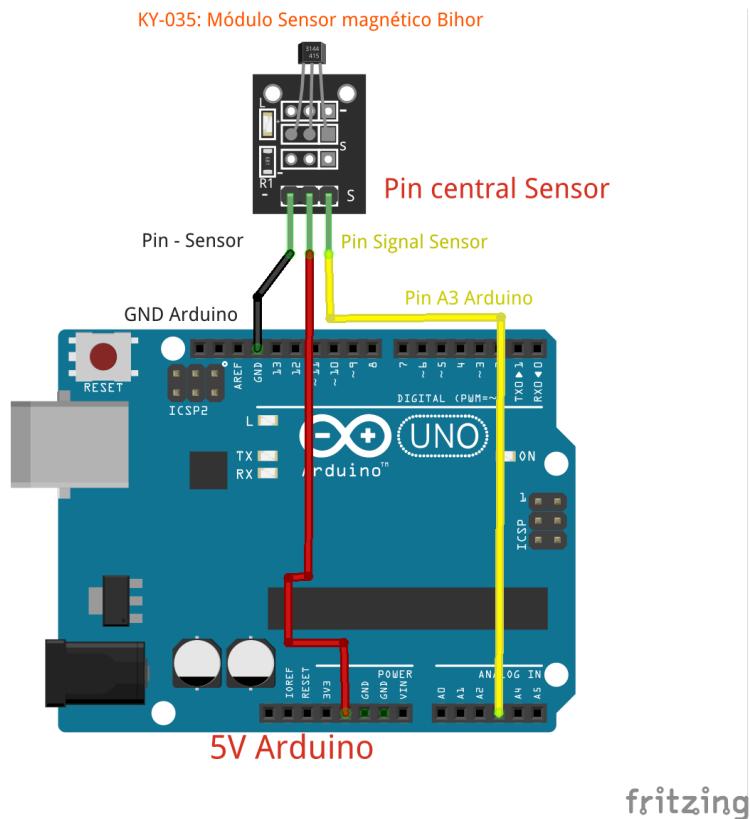
Posee un sensor lineal de efecto hall de Honeywell SS49E. Bajo consumo, 6mA a 5 V.

### Datasheet:

- [https://drive.google.com/file/d/0B8Jv2iO7fU\\_0VWRwdU43X2NweUk/view](https://drive.google.com/file/d/0B8Jv2iO7fU_0VWRwdU43X2NweUk/view)

# Luciano Rabassa

## Conección Arduino:



KY-035	a	Arduino
-	a	GND
VCC	a	5V
Signal	a	A3

## Código Arduino:

```
int Pin_Signal = A3;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    int valorBruto = analogRead(Pin_Signal);
    float Tension = valorBruto * (5.0/1023.0) * 1000;
```

# Luciano Rabassa

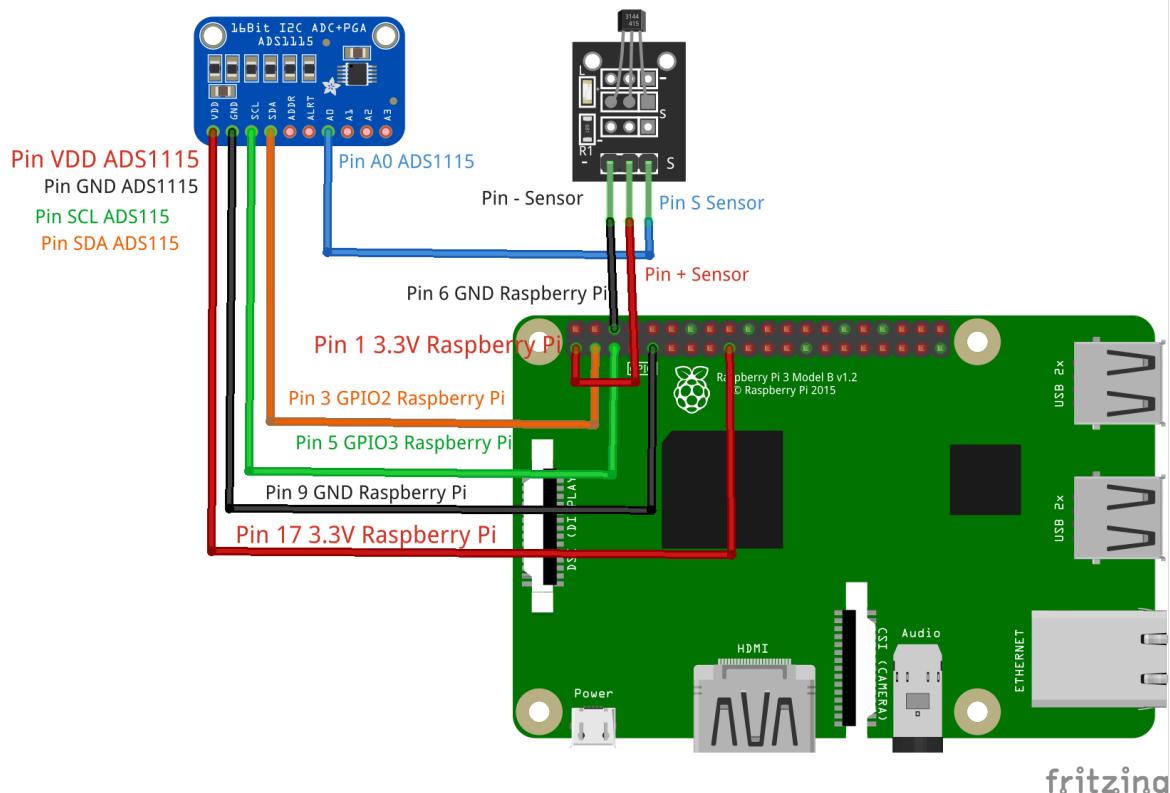
```
float Resistencia = 10000 * ( Tension / ( 5000.0 - Tension));

Serial.print("Tension:");
Serial.print(Tension);
Serial.print("mV");
Serial.print(", Resistencia:");
Serial.print(Resistencia);
Serial.print("Ohm");
Serial.println("-----");
delay(500);
}
```

## Conección Raspberry Pi:

KY-035: Módulo Sensor magnético Bihor

KY-053 ADS1115



KY-035	a	Raspberry Pi
-	a	Pin 6 GND
VCC	a	Pin 1 3.3V
S	a	Pin A0 (ADS1115)
VDD	a	Pin 17 3.3V
GND	a	Pin 9 GND

# Luciano Rabassa

SCL	a	Pin 5 GPIO3(Pin SCL)
SDA	a	Pin 3 GPIO2(Pin SDA )

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

import time, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1115 = 0x01
gain = 4096
sps = 64
adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3
adc = ADS1x15(ic=ADS1115)

try:
    while True:
        adc0 = adc.readADCSingleEnded(adc_channel_0, gain, sps)
        adc1 = adc.readADCSingleEnded(adc_channel_1, gain, sps)
        adc2 = adc.readADCSingleEnded(adc_channel_2, gain, sps)
        adc3 = adc.readADCSingleEnded(adc_channel_3, gain, sps)

        print("Channel 0: ", adc0, "mV ")
        print("Channel 1: ", adc1, "mV ")
        print("Channel 2: ", adc2, "mV ")
        print("Channel 3: ", adc3, "mV ")

        time.sleep(retraso)

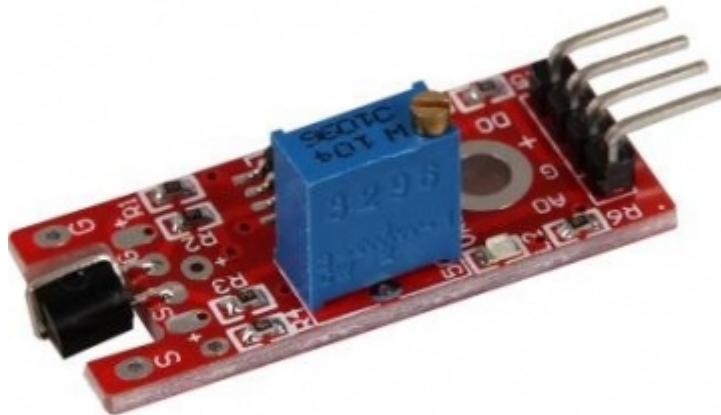
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-035.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-035.py*

## KY-036: Módulo Sensor Táctil

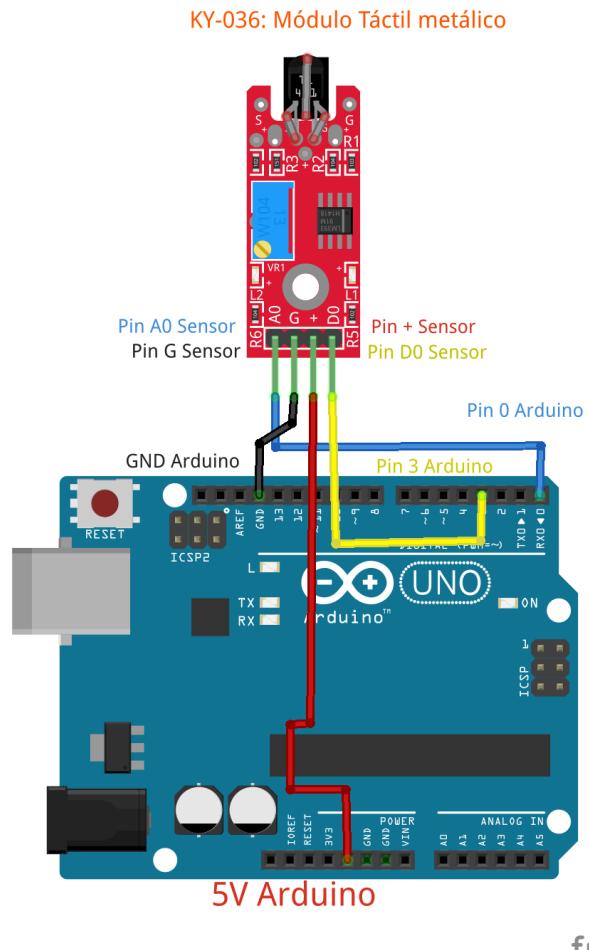


### Descripción:

Posee un transistor, que actúa como amplificador de señal. Cuando tocamos la base del transistor (pata metálica) el led se encenderá.

# Luciano Rabassa

## Conección Arduino:



KY-036	a	Arduino
A0	a	Pin Digital 0
G	a	Pin GND
+	a	Pin 5V
D0	a	Pin Digital 3

## Código Arduino:

```
int Pin_A0 = A0;
int Pin_D0 = 3;

void setup()
{
    pinMode(Pin_A0, INPUT);
    pinMode(Pin_D0, INPUT);

    Serial.begin(9600);
}

void loop()
{
    float Analog;
    int Digital;

    Analog = analogRead(Pin_A0 * (5.0 / 1023.0));
    Digital = digitalRead(Pin_D0);

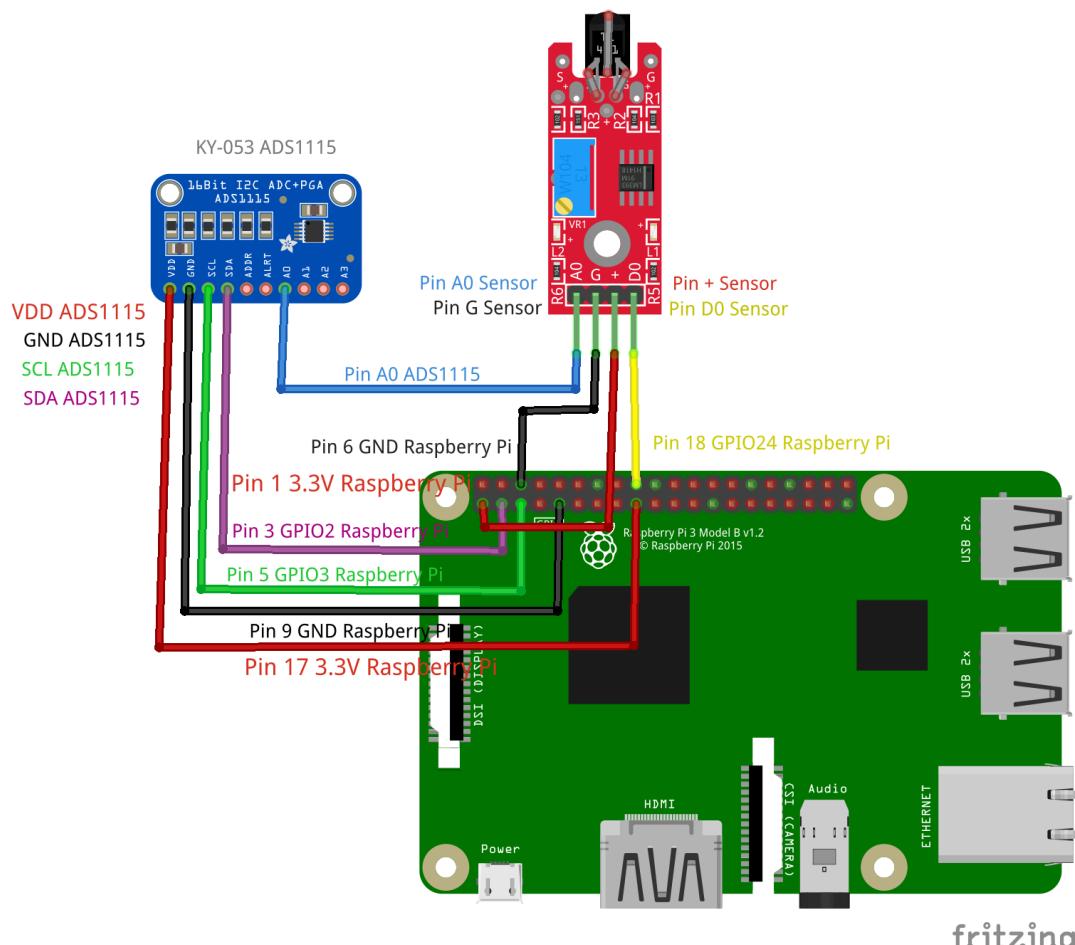
    Serial.print("Tension analoga:"); Serial.print(Analog, 4);
    Serial.print("V, ");
    Serial.print("Valor extremo: ");

    if(Digital == 1)
    {
        Serial.println(" alcanzado");
    }
    else
    {
        Serial.println(" no alcanzado");
    }
    delay(200);
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

KY-036: Módulo Táctil metálico



KY-036	a	Raspberry Pi
A0	a	Pin A0 (ADS1115)
G	a	Pin 6 GND
+	a	Pin 1 3.3V
D0	a	Pin 18 GPIO24
VDD	a	Pin 17 3.3V
GND	a	Pin 9 GND
SCL	a	Pin 5 GPIO3(Pin SCL en Raspberry Pi)
SDA	a	Pin 3 GPIO2(Pin SDA en Raspberry Pi)

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

import math, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1115 = 0x01
gain = 4096
sps = 64
adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3
adc = ADS1x15(ic=ADS1115)

Digital = 24

GPIO.setup(Digital, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

try:
    while True:
        analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)
        if GPIO.input(Digital) == False:
            print("Tensionanalogica: ", analog, "mV, ", "Valor
extremo: no alcanzado")
        else:
            print("Tensionanalogica: ", analog, "mV, ", "Valor
extremo: alcanzado")
            sleep(retraso)

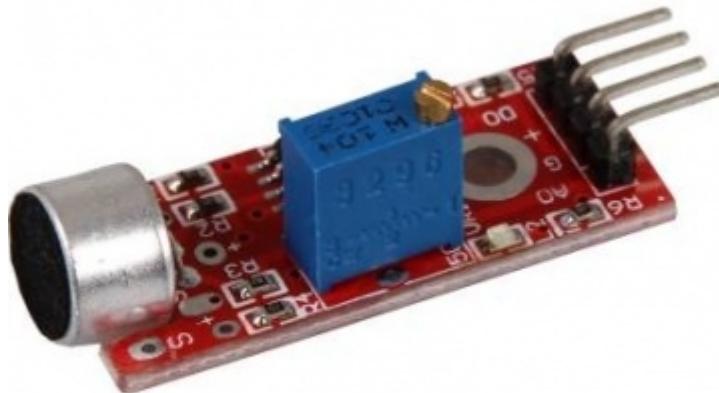
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-036.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-036.py*

## KY-037: Módulo Big Sound



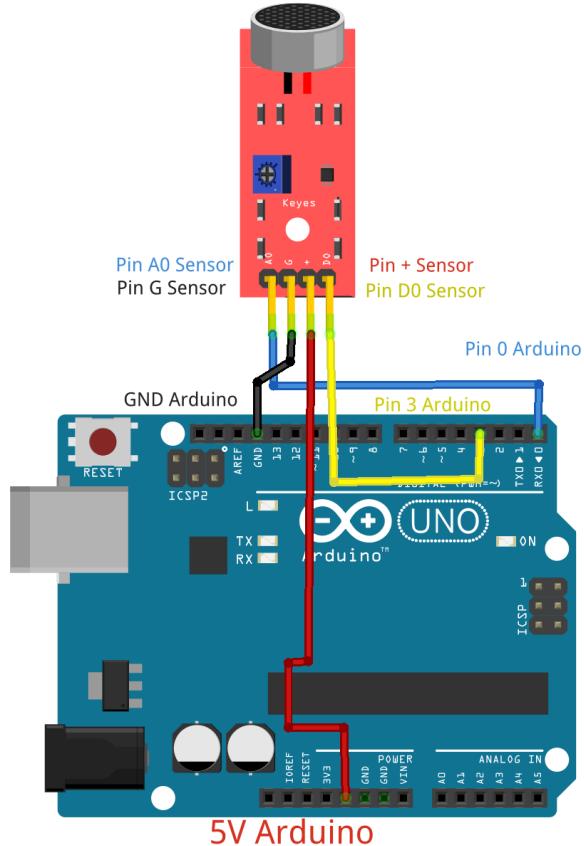
### Descripción:

En el pin A0 se lee directamente la tensión recibida a través del microfono, el pin D0 es configurado mediante el potenciómetro para que, al recibir un determinado valor de tensión del pin A0, a partir del cual, se enviará un valor HIGH mediante el pin D0. Es muy útil para realizar un audioritmo.

# Luciano Rabassa

## Conección Arduino:

KY-037: Módulo Big Sound



fritzing

KY-037	a	Arduino
A0	a	Pin Digital 0
G	a	Pin GND
+	a	Pin 5V
D0	a	Pin Digital 3

## Código Arduino:

```
int Pin_A0 = 0;
int Pin_D0 = 3;

void setup()
{
    pinMode(Pin_A0, INPUT);
    pinMode(Pin_D0, INPUT);

    Serial.begin(9600);
}

void loop()
{
    float Analog;
    int Digital;

    Analog = analogRead(Pin_A0 * (5.0 / 1023.0));
    Digital = digitalRead(Pin_D0);

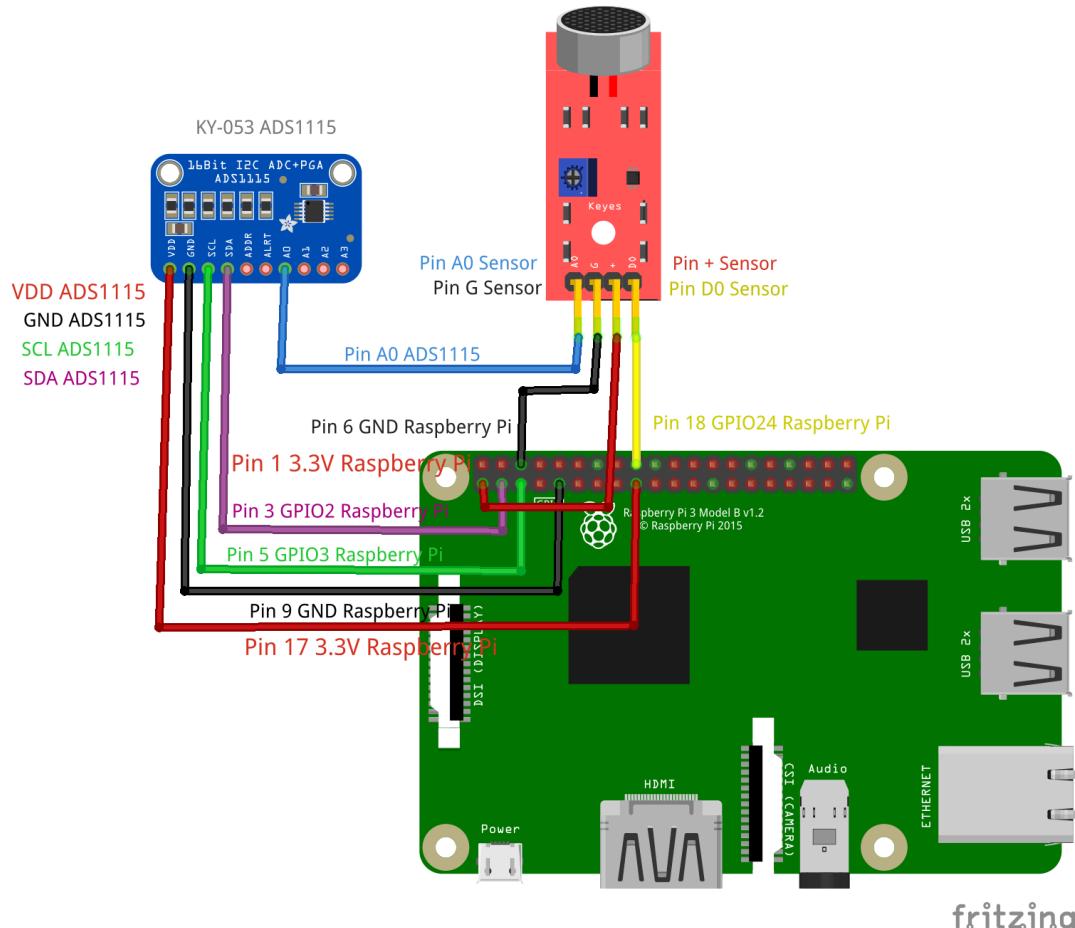
    Serial.print("Tensionanaloga: "); Serial.print(Analog, 4);
    Serial.print("V, ");
    Serial.print("Valor extremo: ");

    if(Digital == 1)
    {
        Serial.println(" alcanzado");
    }
    else
    {
        Serial.println(" no alcanzado");
    }
    delay(200);
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

KY-037: Módulo Big Sound



KY-037	a	Raspberry Pi
A0	a	Pin A0 (ADS1115)
G	a	Pin 6 GND
+	a	Pin 1 3.3V
D0	a	Pin 18 GPIO24
VDD	a	Pin 17 3.3V
GND	a	Pin 9 GND
SCL	a	Pin 5 GPIO3(Pin SCL)
SDA	a	Pin 3 GPIO2(Pin SDA)

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep
import math, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1015 = 0x00
ADS1115 = 0x01
gain = 4096
sps = 64
adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3
adc = ADS1x15(ic=ADS1115)
Digital = 24

GPIO.setup(Digital, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

try:
    while True:
        analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

        if GPIO.input(Digital) == False:
            print("Tensionanalog:", analog, "mV, ", "Valor
extremo: no alcanzado")
        else:
            print("Tensionanalog:", analog, "mV, ", "Valor
extremo: alcanzado")
            print("-----")
            sleep(retraso)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-037.py

Para correr el programa abrimos la Terminal y tecleamos:

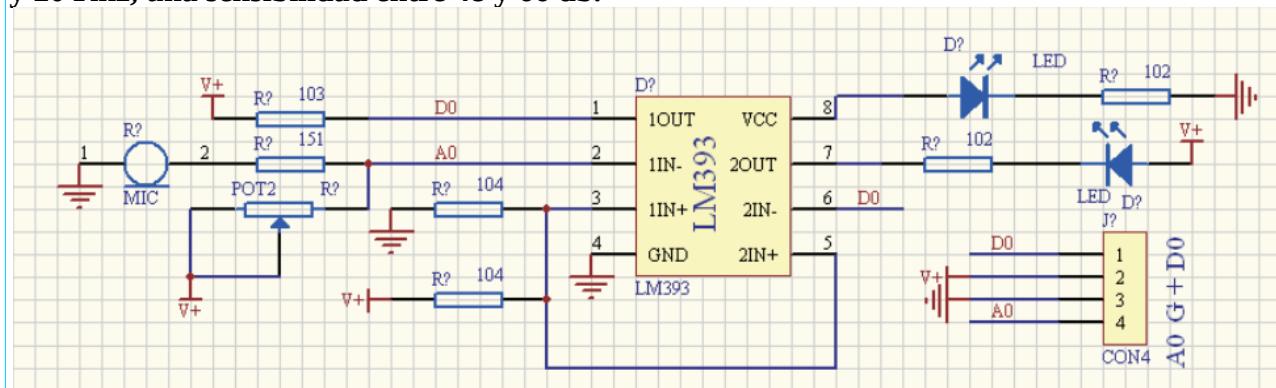
- *sudo python3 KY-037.py*

## KY-038: Módulo Small Sound



### Descripción:

En el pin A0 se lee directamente la tensión recibida a través del microfono, el pin D0 es configurado mediante el potenciómetro para que, al recibir un determinado valor de tensión del pin A0, a partir del cual, se enviará un valor HIGH mediante el pin D0. Es muy útil para realizar un audioritmo. Posee una impedancia de  $2.2\text{K}\Omega$ , una frecuencia entre 50 Hz y 20 Khz, una sensibilidad entre 48 y 66 db.

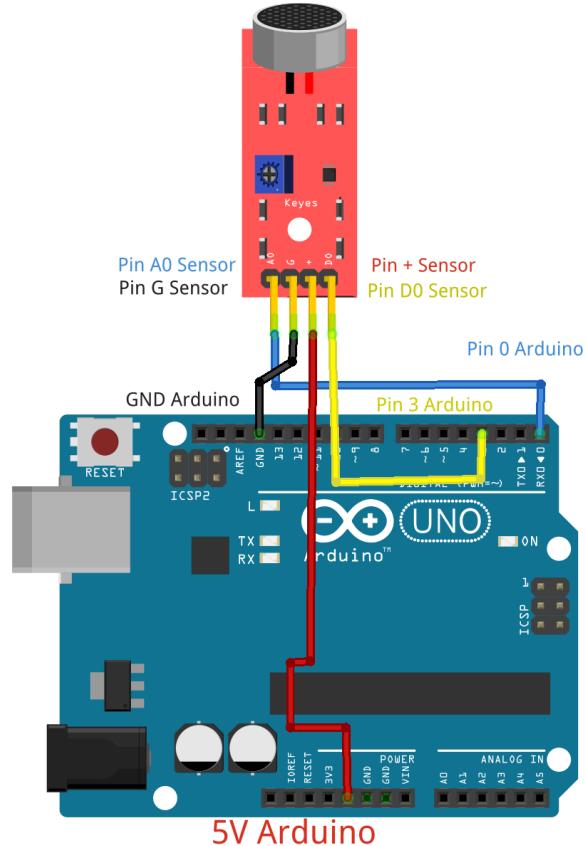


Datasheet: [https://www.velleman.eu/downloads/29/vma309\\_a4v02.pdf](https://www.velleman.eu/downloads/29/vma309_a4v02.pdf)

# Luciano Rabassa

## Conección Arduino:

KY-038: Módulo Small Sound



fritzing

KY-038	a	Arduino
A0	a	Pin Digital 0
G	a	Pin GND
+	a	Pin 5V
D0	a	Pin Digital 3

## Código Arduino:

```
int Pin_A0 = 0;
int Pin_D0 = 3;

void setup()
{
    pinMode(Pin_A0, INPUT);
    pinMode(Pin_D0, INPUT);

    Serial.begin(9600);
}

void loop()
{
    float Analog;
    int Digital;

    Analog = analogRead(Pin_A0) * (5.0 / 1023.0);
    Digital = digitalRead(Pin_D0);

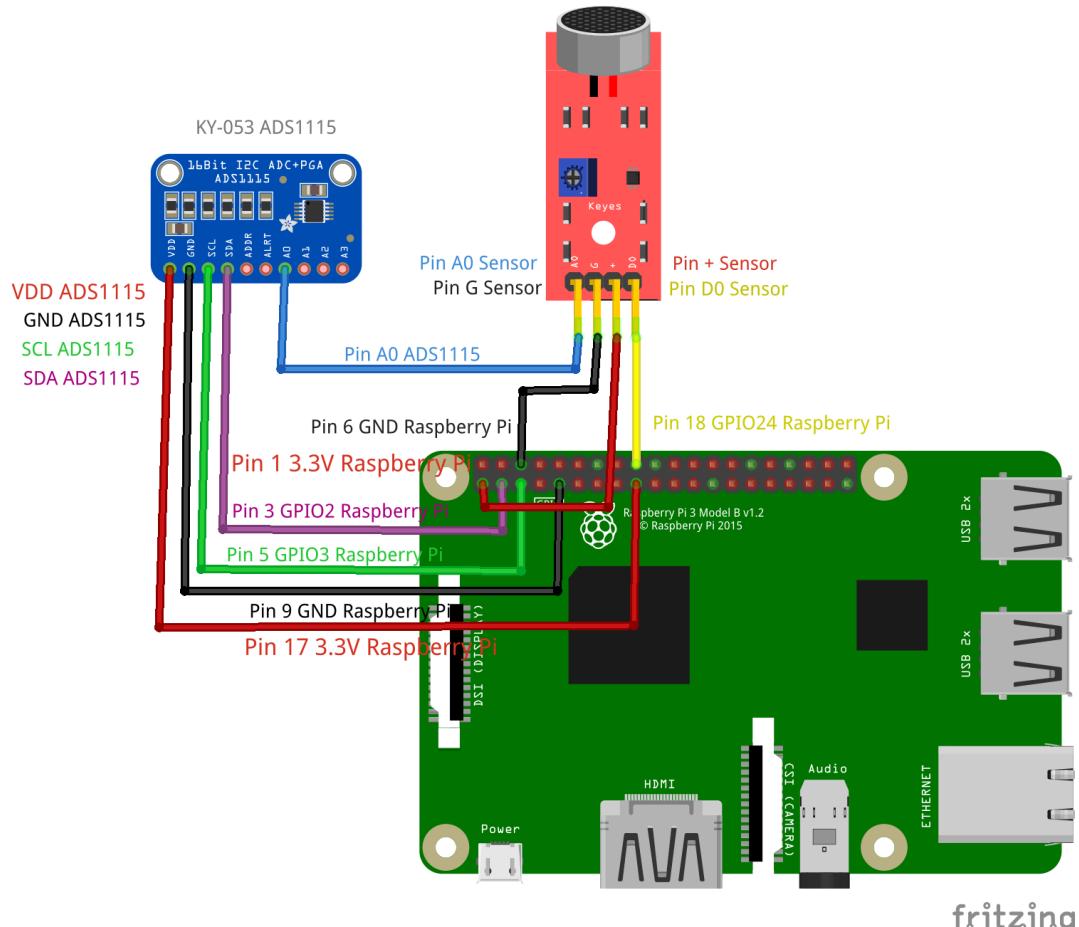
    Serial.print("Tensionanaloga:");
    Serial.print (Analog, 4);
    Serial.print("V, ");
    Serial.print("Valor extremo:");

    if(Digital == 1)
    {
        Serial.println(" alcanzado");
    }
    else
    {
        Serial.println(" no alcanzado");
    }
    Serial.println("-----");
    delay(200);
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

KY-038: Módulo Small Sound



KY-038	a	Raspberry Pi
A0	a	Pin A0 (ADS1115)
G	a	Pin 6 GND
+	a	Pin 1 3.3V
D0	a	Pin 18 GPIO24
VDD	a	Pin 17 3.3V
GND	a	Pin 9 GND
SCL	a	Pin 5 GPIO3(Pin SCL)
SDA	a	Pin 3 GPIO2(Pin SDA)

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

import math, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5
ADS1115 = 0x01
gain = 4096
sps = 64
adc_channel_0 = 0
adc_channel_1 = 1
adc_channel_2 = 2
adc_channel_3 = 3
adc = ADS1x15(ic=ADS1115)
Digital = 24

GPIO.setup(Digital, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

try:
    while True:
        analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)
        if GPIO.input(Digital) == False:
            print("Tensionanalogica:", analog, "mV, ", "Valor
extremo: no alcanzado")
        else:
            print("Tensionanalogica:", analog, "mV, ", "Valor
extremo: alcanzado")
        sleep(retraso)

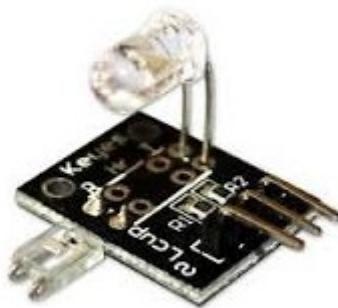
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-038.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-038.py*

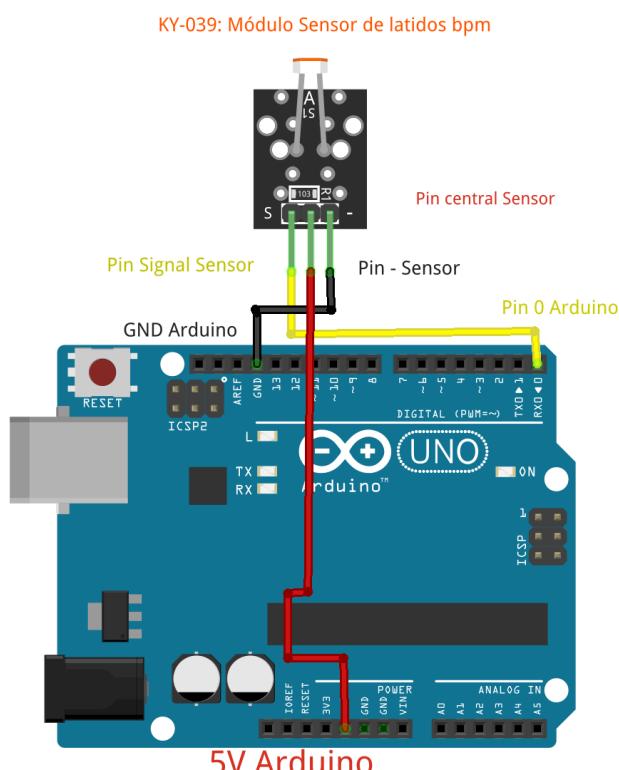
## KY-039: Módulo Sensor de latidos



### Descripción:

Mide nuestro ritmo cardíaco, utilizando un fototransistor y un fotodiodo, al apoyar nuestro dedo sobre el fototransistor se cambia el flujo de luz emitido por lo cual se mide la variación, dándonos cuantos latidos por minutos generamos.

### Conexión Arduino:



fritzing

# Luciano Rabassa

KY-039	a	Arduino
S	a	Pin Digital 0
+	a	Pin 5V
-	a	Pin GND

## Código Arduino:

```
//Al compilar mantener el Pin_S desconectado del Pin 0 de Arduino o dará fallo
int Pin_S = 0;
const int muestreo = 60;
int valor;

void setup(){
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  Serial.println("Latidos");
}

void loop(){
  static int latido_por_segundos = 0;
  int pulsobpm = 0;

  Serial.println(valor);
  if(deteccion(Pin_S, muestreo)){
    pulsobpm = 60000 / latido_por_segundos;

    digitalWrite(LED_BUILTIN, HIGH);

    Serial.print(valor);
    Serial.print(",");
    Serial.println(pulsobpm);
    latido_por_segundos = 0;
  }
  else{
    digitalWrite(LED_BUILTIN, LOW);
  }
  delay(muestreo);
  latido_por_segundos = latido_por_segundos + muestreo;
}

bool deteccion(int IrSensor, int retardo){
  static int maximo = 0;
  static bool pico = false;
  bool resultado = false;
  valor = analogRead(IrSensor);
  valor = valor * (1000/retardo);
  if(valor * 4L < maximo){
    maximo = valor * 0.8;
```

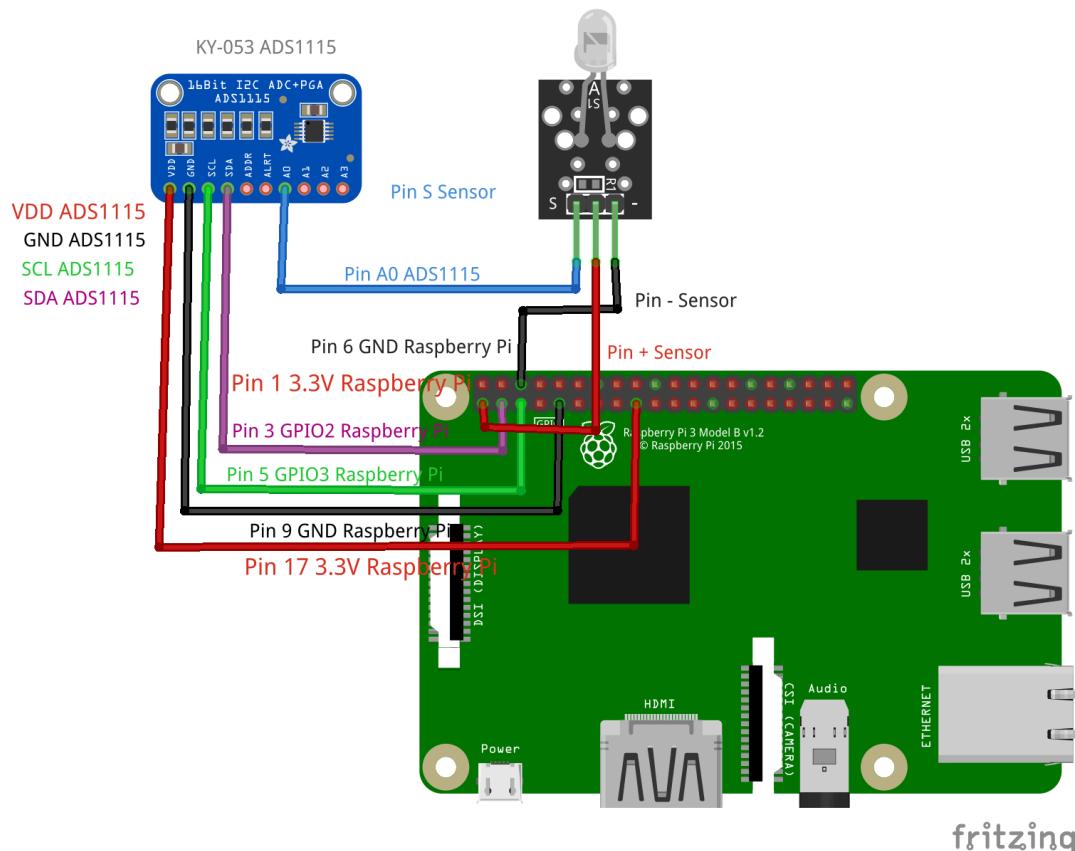
# Luciano Rabassa

```
if(valor > maximo) {
    maximo = valor;
}
if(pico == false){
    resultado = true;
}
pico = true;
}
else if(valor < maximo - (3000 / retardo)){
    pico = false;
    maximo = maximo - (1000 / retardo);
}
return resultado;
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

KY-039: Módulo Sensor de latidos bpm



KY-039	a	Raspberry Pi
S	a	Pin A0 ADS1115
+	a	Pin 1 3.3V
-	a	Pin 6 GND
VDD	a	Pin 17 3.3V
GND	a	Pin 9 GND
SCL	a	Pin 5 GPIO3(Pin SCL en Raspberry Pi)
SDA	a	Pin 3 GPIO2(Pin SDA en Raspberry Pi)

## Código Raspberry Pi:

```
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

import time, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

latidos_por_minuto = 0
isPeak = False
result = False
retraso = 0.01
maxValue = 0
schwelle = 25
beatTime = 0
oldBeatTime = 0

ADS1015 = 0x00
ADS1115 = 0x01
gain = 4096
sps = 8
adc_channel = 0
adc = ADS1x15(ic=ADS1115)

LED = 24
GPIO.setup(LED, GPIO.OUT, initial = GPIO.LOW)

def heartBeatDetect(schwelle):
    global maxValue
    global isPeak
    global result
    global oldBeatTime

    rawValue = adc.readADCSingleEnded(adc_channel, gain, sps)

    if result == True:
        result = False
        if rawValue * 4 < maxValue:
            maxValue = rawValue * 0.8;

        if rawValue > maxValue:
            maxValue = rawValue

        if isPeak == False:
            result = True
```

```
isPeak = True

else:
    if rawValue < maxValue - schwelle:
        isPeak = False

    maxValue = maxValue - schwelle / 2

if result == True:

    beatTime = time.time()
    timedifference = beatTime - oldBeatTime
    latidos_por_minuto = 60 / timedifference
    oldBeatTime = beatTime

    GPIO.output(LED,GPIO.HIGH)
    time.sleep(retraso * 10)
    GPIO.output(LED,GPIO.LOW)

return latidos_por_minuto

try:
    while True:
        time.sleep(retraso)
        latidos_por_minuto = heartBeatDetect(schwelle)
        if result == True:
            print("-Heartbeat detected !- Puls: ",
int(latidos_por_minuto), " (bpm) ")

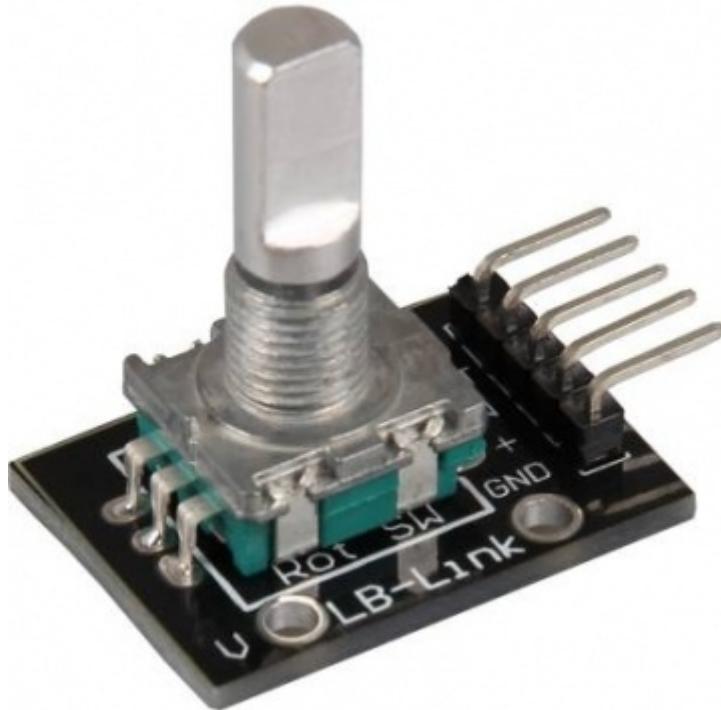
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-039.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-039.py*

## KY-040: Módulo encoder rotativo



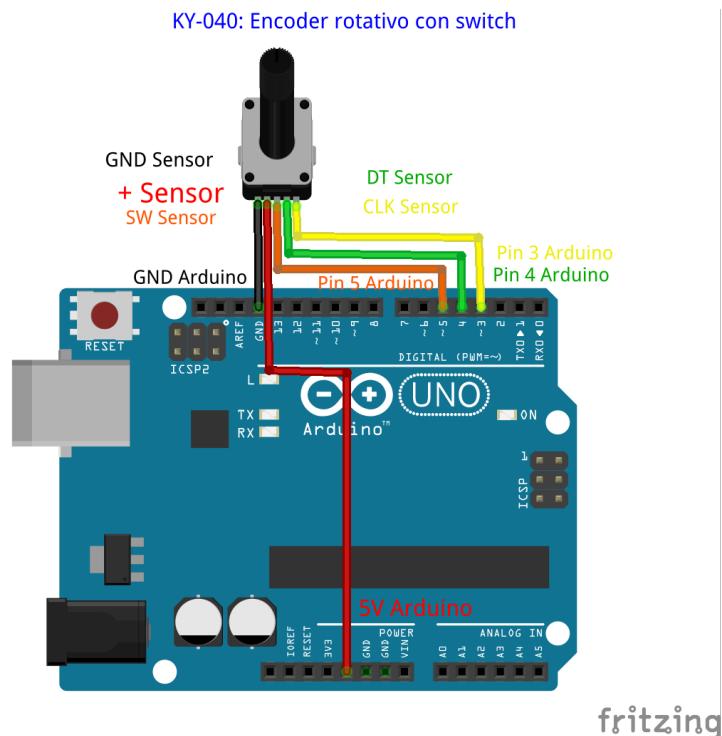
### Descripción:

Es un dispositivo de entrada giratorio que proporciona una indicación de cuánto la perilla ha sido girada y en qué dirección está girando.

Es un gran dispositivo para el control de motores paso a paso y servos. También se podría utilizar para controlar dispositivos como potenciómetros digitales. Un encoder giratorio tiene un número fijo de posiciones por revolución. Estas posiciones, son medidas fácilmente por pequeños "clics" cuando gira el encoder. Éste módulo tiene treinta posiciones.

# Luciano Rabassa

## Conección Arduino:



KY-040	a	Arduino
GND	a	Pin GND
+	a	Pin 5V
SW	a	Pin Digital 5
DT	a	Pin Digital 4
CLK	a	Pin Digital 3

## Código Arduino:

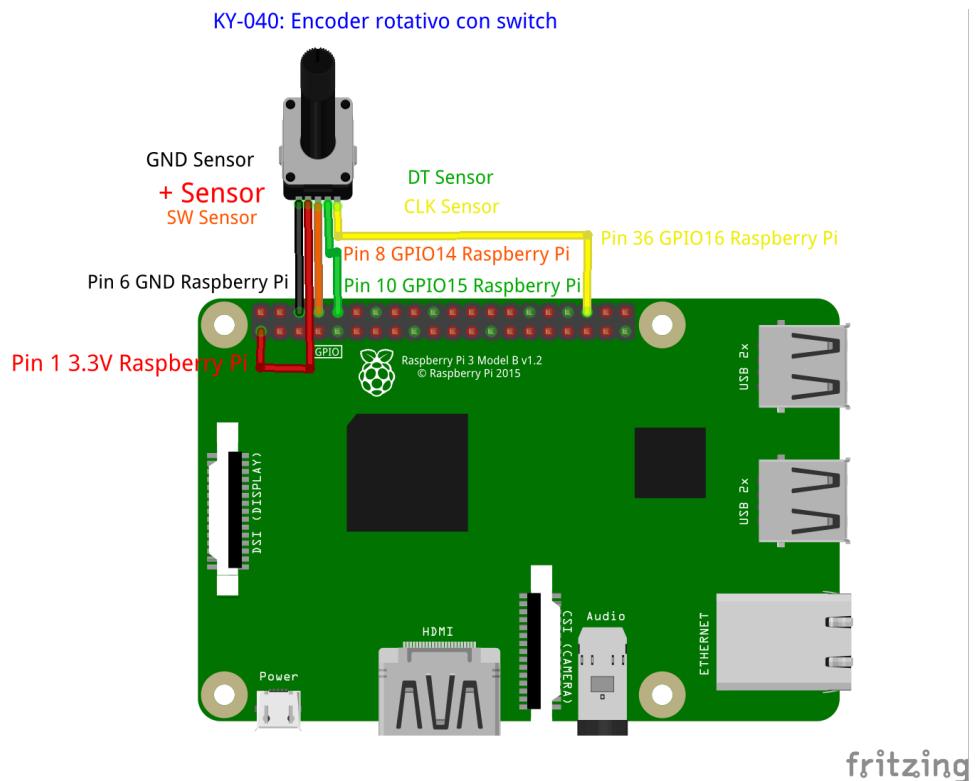
```
int Contador = 0;
boolean direccion;
int Pin_clk_anterior;
int Pin_clk_actual;
int pin_clk = 3;
int pin_dt = 4;
int Pin_SW = 5;

void setup() {
    pinMode(pin_clk, INPUT);
    pinMode(pin_dt, INPUT);
    pinMode(Pin_SW, INPUT);
    digitalWrite(pin_clk, true);
    digitalWrite(pin_dt, true);
    digitalWrite(Pin_SW, true);
    Pin_clk_anterior = digitalRead(pin_clk);
    Serial.begin(115200);
}

void loop(){
    Pin_clk_actual = digitalRead(pin_clk);
    if(Pin_clk_actual != Pin_clk_anterior){
        if(digitalRead(pin_dt) != Pin_clk_actual){
            Contador++;
            direccion = true;
        }
        else{
            direccion = false;
            Contador--;
        }
        Serial.println("Rotacion detectada: ");
        Serial.print("Direccion de giro: ");
        if(direccion){
            Serial.println("Sentido horario");
        }
        else{
            Serial.println("Sentido antihorario");
        }
        Serial.print("Posicion actual: ");
        Serial.println(Contador);
        Serial.println("-----");
    }
    Pin_clk_anterior = Pin_clk_actual;
    if(!digitalRead(Pin_SW) && Contador != 0){
        Contador = 0;
        Serial.println("Posicion reiniciada");
    }
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:



KY-040	a	Raspberry Pi
GND	a	Pin 6 GND
+	a	Pin 1 3.3V
SW	a	Pin 8 GPIO14
DT	a	Pin 10 GPIO15
CLK	a	Pin 36 GPIO16

## Código Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
PIN_CLK = 16
PIN_DT = 15
PIN_SW = 14

GPIO.setup(PIN_CLK, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(PIN_DT, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(PIN_SW, GPIO.IN, pull_up_down = GPIO.PUD_UP)

Contador = 0
direccion = True
PIN_CLK_anterior = 0
PIN_CLK_actual = 0
retraso = 0.01
PIN_CLK_anterior = GPIO.input(PIN_CLK)

def deteccionGiro(null):
    global Contador
    PIN_CLK_actual = GPIO.input(PIN_CLK)
    if PIN_CLK_actual != PIN_CLK_anterior:
        if GPIO.input(PIN_DT) != PIN_CLK_actual:
            Contador += 1
            direccion = True;
        else:
            direccion = False
            Contador = Contador - 1
        print("Rotacion detectada: ")
        if direccion:
            print("Sentido horario")
        else:
            print("Sentido anti-horario")
        print("Posicion actual: ", Contador)

def resetContador(null):
    global Contador
    print("Posicion reiniciada!")
    Contador = 0

GPIO.add_event_detect(PIN_CLK, GPIO.BOTH, callback =
deteccionGiro, bouncetime = 50)
GPIO.add_event_detect(PIN_SW, GPIO.FALLING, callback =
resetContador, bouncetime = 50)

print("Prueba del modulo [Presiona Ctrl + C para finalizar la prueba] ")
```

# Luciano Rabassa

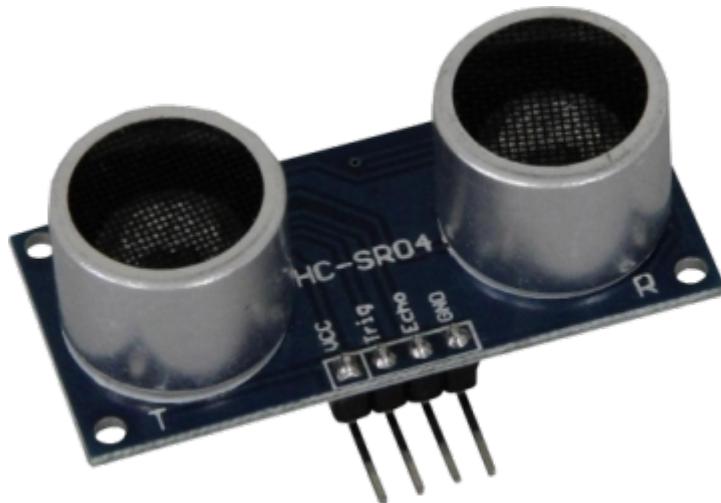
```
try:  
    while True:  
        time.sleep(retraso)  
  
except KeyboardInterrupt:  
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-040.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-040.py*

## KY-050: Módulo HC-SR04 Sensor de distancia



### Descripción:

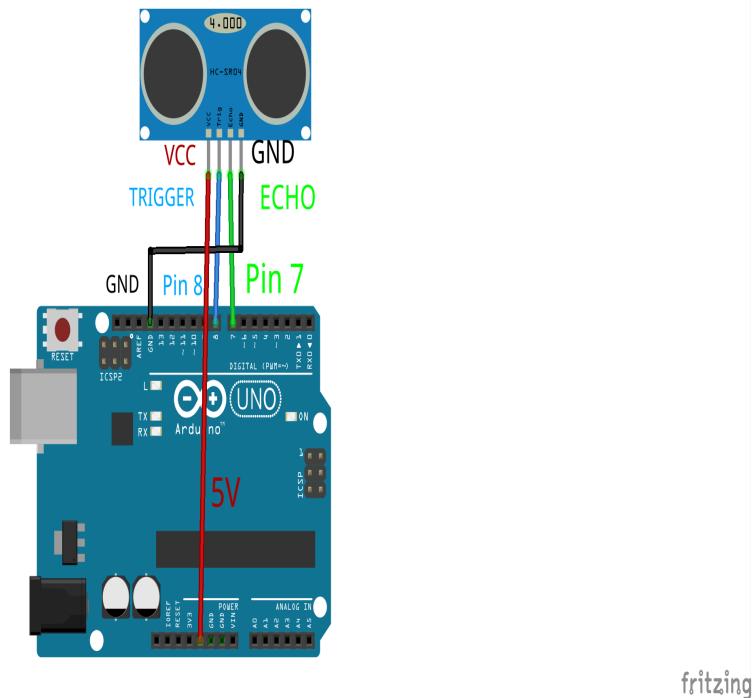
Al recibir una señal el Trigger, comenzará la medición que será enviada como PWM-TTL por el echo.

- Distancia : 2cm a 3m
- Resolución de medición (error) 3mm
- Tiempo mínimo entre mediciones 50  $\mu$ s

# Luciano Rabassa

## Conección Arduino:

KY-050: Módulo HC-SR04 Sensor de distancia



fritzing

KY-050	a	Arduino
VCC	a	Pin 5V
TRIG	a	Pin Digital 8
ECHO	a	Pin Digital 7
GND	a	Pin GND

## Código Arduino:

```
#define Echo_Pin 7
#define Trigger_Pin 8

int rangoMax = 300; // 300 cm
int rangoMin = 2; // 2 cm
long distancia;
long duracion;

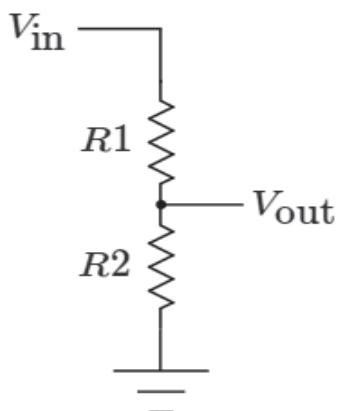
void setup()
{
```

```
pinMode(Trigger_Pin, OUTPUT);
pinMode(Echo_Pin, INPUT);
Serial.begin(9600);
}

void loop()
{
    digitalWrite(Trigger_Pin, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trigger_Pin, LOW);
    duracion = pulseIn(Echo_Pin, HIGH);
    distancia = duracion / 58.2;
    if(distancia >= rangoMax || distancia <= rangoMin)
    {
        Serial.println("Fuera de rango de medición");
    }
    else
    {
        Serial.print("La distancia es de: ");
        Serial.print(distancia);
        Serial.println("cm");
        Serial.println("-----");
    }
    delay(500);
}
```

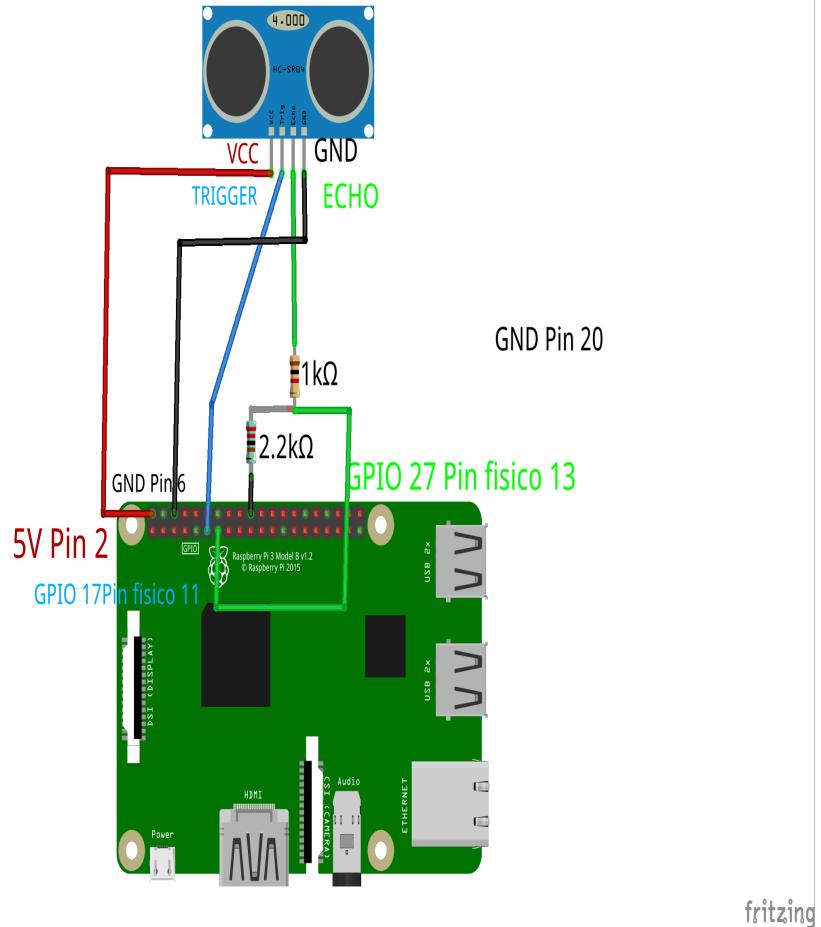
## Conexión Raspberry Pi:

Para poder conectar el módulo de 5V al Raspberry Pi debemos hacer un divisor resistivo. Un **divisor resistivo** consiste de dos resistencias ( $R_1$  y  $R_2$ ) en serie conectadas a un voltaje de entrada ( $V_{in}$ ), el cual debe ser reducido a un voltaje de salida ( $V_{out}$ ). En nuestro circuito,  $V_{in}$  corresponderá al ECHO, el cual requiere disminuir de 5V al  $V_{out}$  que necesitamos de 3.3v. El siguiente esquema muestra la disposición que deben tener las resistencias:



$$\frac{V_{out}}{V_{in}} = \frac{R_2}{R_1 + R_2}$$

## KY-050: Módulo HC-SR04 Sensor de distancia



fritzing

KY-050	a	Raspberry Pi
VCC	a	Pin 2 5V
TRIG	a	Pin 11 GPIO17
ECHO	a	R1
R1	a	R2
Cable entre R1 y R2	a	Pin 13 GPIO27
R2	a	Pin 20 GND
GND	a	Pin 6 GND

## Código Raspberry Pi:

```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

Trigger_Pin = 17
Echo_Pin = 27
sleepetime = 0.8

GPIO.setup(Trigger_Pin, GPIO.OUT)
GPIO.setup(Echo_Pin, GPIO.IN)
GPIO.output(Trigger_Pin, False)

try:
    while True:
        GPIO.output(Trigger_Pin, True)
        time.sleep(0.00001)
        GPIO.output(Trigger_Pin, False)
        pulso_inicio = time.time()

        while GPIO.input(Echo_Pin) == 0:
            pulso_inicio = time.time()

        while GPIO.input(Echo_Pin) == 1:
            pulso_fin = time.time()

        duracion = pulso_fin - pulso_inicio

        distancia = (duracion * 34300) / 2

        if distancia < 2 or (round(distancia) > 300):
            print("Distancia fuera de rango")
            print("-----")
        else:
            distancia = format((duracion * 34300) / 2, '.2f')
            print(("La distancia es de:"), distancia, ("cm"))
            print("-----")
        time.sleep(sleepetime)

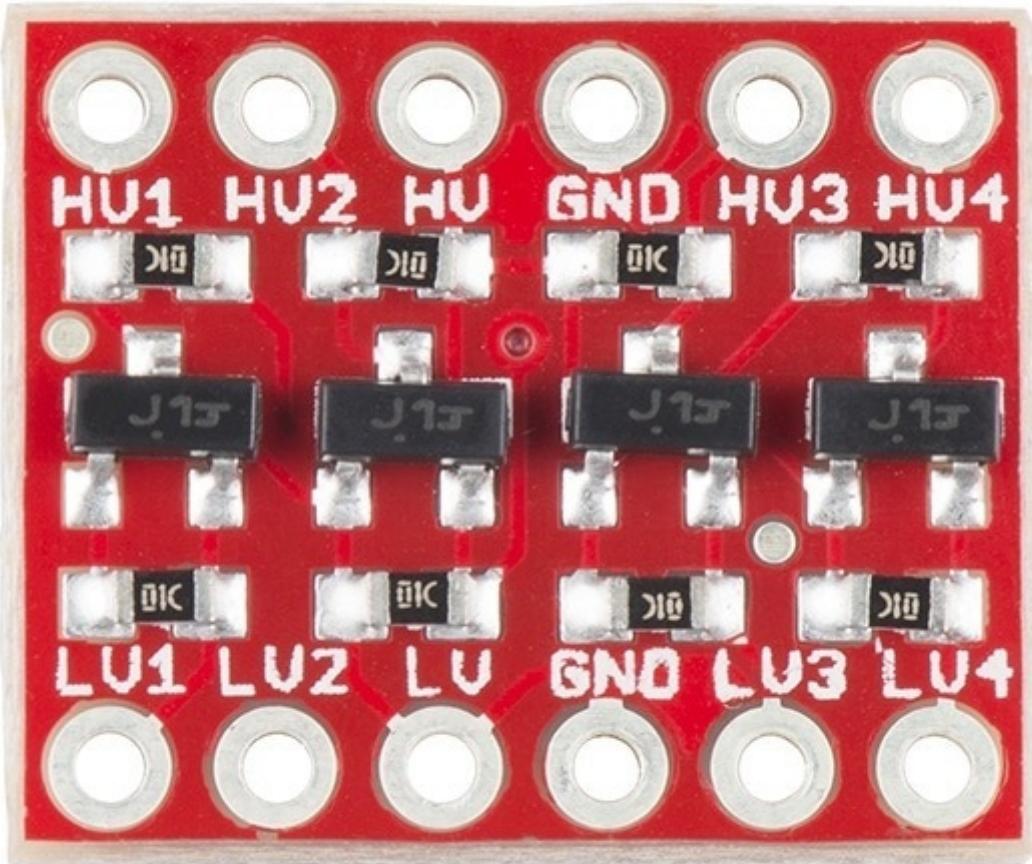
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-050.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-050.py*

## KY-051: Convertidor de tensión Bi-Direccional(Cambiador de Nivel)



### Descripción:

Este cambiador de nivel modula la tensión de una señal digital hacia una más grande o una más baja, posee 4 canales que pueden ser modulados.

Hay muchos sistemas micro controlados que son operados en varios rangos de tensión: los sistemas más viejos, como Arduino son usados 5V para sus pines de entrada/salida los sistemas nuevos como la Raspberry Pi son usados 3.3V. Antiguamente, los micro

controladores necesitaban tensiones más altas para comunicarse. Actualmente, desde que los problemas con ruidos de señal / interrupciones de señales han sido resueltos, el rango de tensión está siendo mantenido tan bajo como sea posible para reducir el calor y reducir el consumo de energía. Los sistemas con 1.8 V son inusuales en estos días.

Pero para comunicarse entre dos sistemas diferentes, como en nuestro Arduino y nuestra Raspberry Pi, tenemos que cambiar el rango de tensión, si no lo hacemos, el sistema con menor rango de tensión creará más calor y podría dañarse.

## Conexión Arduino y Raspberry Pi:

KY-051	Arduino	Raspberry Pi
LV		Pin 1 3.3V
LV1		Pin 8 GPIO14 TX
LV2		Pin 10 GPIO15 RX
LV3		
LV4		
GND(para ambas placas)	Pin GND	Pin 6 GND
HV	Pin 5V	
HV1	Pin Digital 0 RX0	
HV2	Pin Digital 1 TX0	
HV3		
HV4		

Con esta placa podemos convertir:

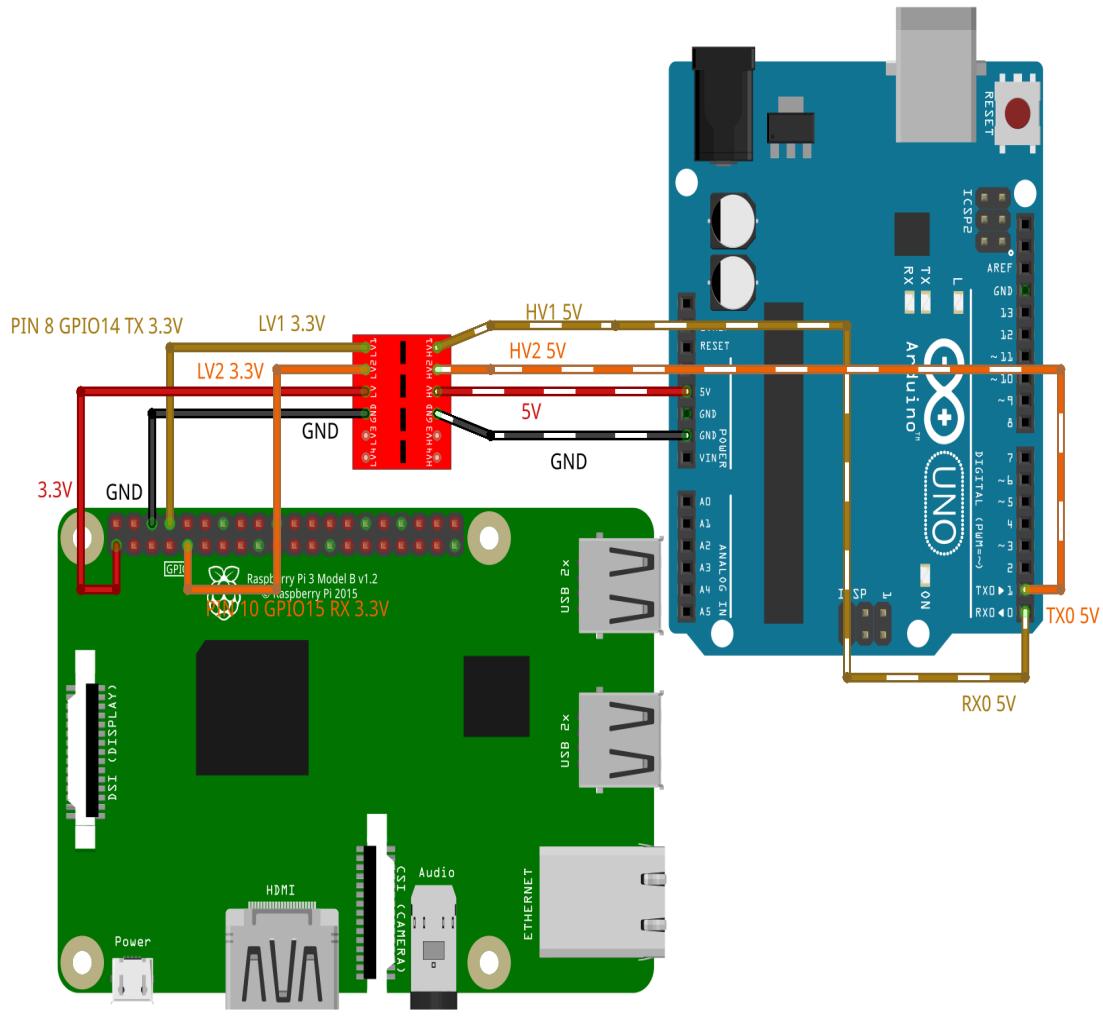
- 5V a 3.3V
- 3.3V a 5V

Lo que debemos respetar es la leyenda en la placa donde nos indica que los 5V siempre se conectarán del lado HV y los 3.3V del lado LV

# Luciano Rabassa

## Conexión Raspberry Pi ↔ Arduino:

MÓDULO KY-051 voltage translator/Level shifter

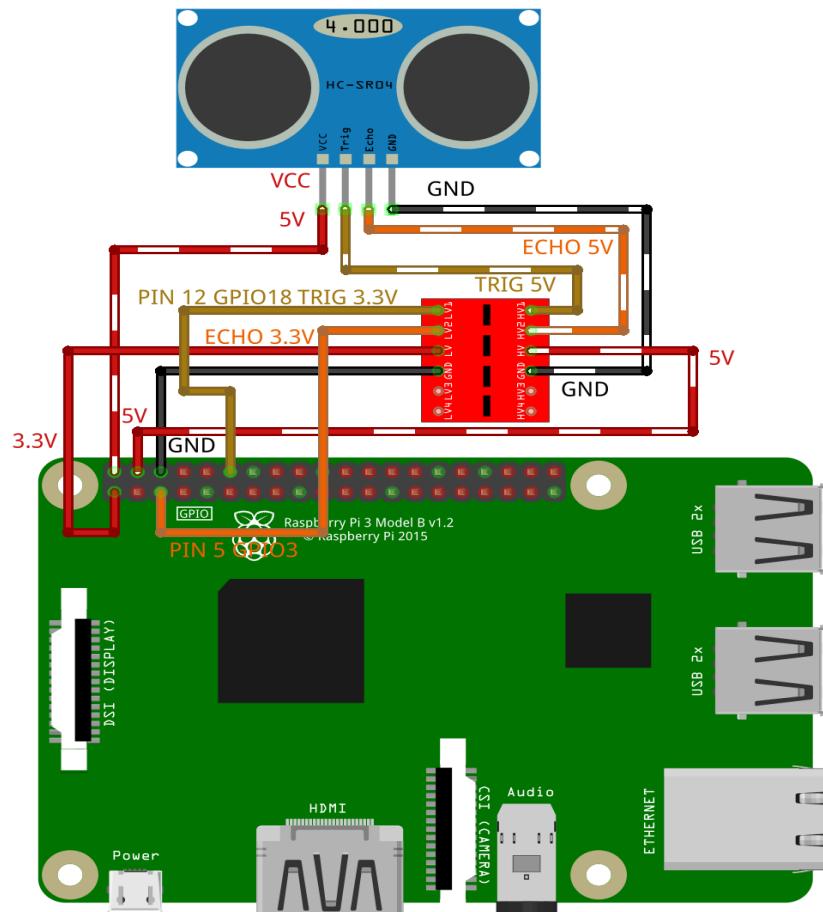


fritzing

# Luciano Rabassa

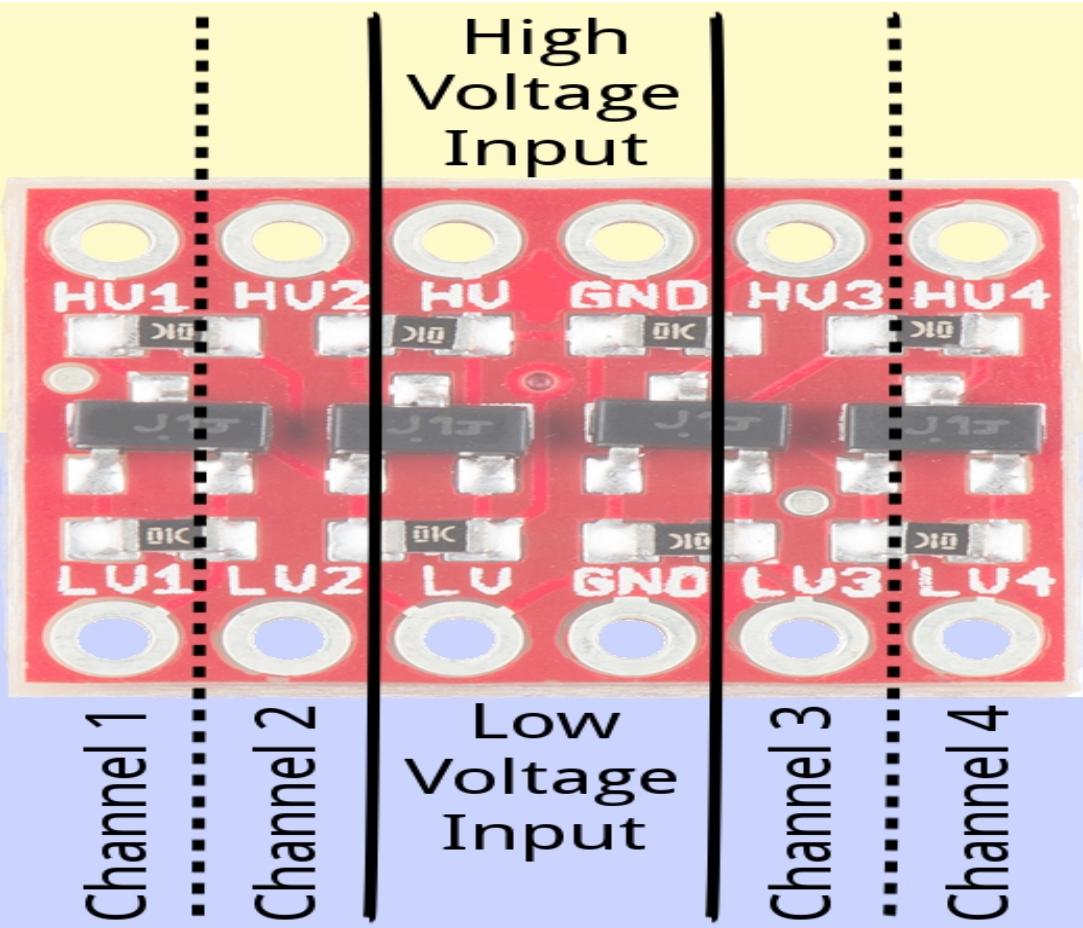
## Ejemplo conexión Raspberry Pi ↔ Convertidor de Tensión ↔ KY-051 HC-SR04

MÓDULO KY-051 voltage translator/Level shifter



fritzing

KY-051	KY-050	Raspberry Pi
HV		Pin 4 5V
HV1	Trig	
HV2	Echo	
LV		Pin 1 3.3V
LV1		Pin 12 GPIO18
LV2		Pin 5 GPIO3
GND	GND	Pin 6 GND
	VCC	Pin 2 5V



## Código Raspberry Pi:

```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

Trigger_Pin = 18
Echo_Pin = 3
sleepTime = 0.8

GPIO.setup(Trigger_Pin, GPIO.OUT)
GPIO.setup(Echo_Pin, GPIO.IN)
GPIO.output(Trigger_Pin, False)

try:
    while True:
        GPIO.output(Trigger_Pin, True)
        time.sleep(0.00001)
        GPIO.output(Trigger_Pin, False)
```

# Luciano Rabassa

```
pulso_inicio = time.time()
while GPIO.input(Echo_Pin) == 0:
    pulso_inicio = time.time()

while GPIO.input(Echo_Pin) == 1:
    pulso_fin = time.time()

duracion = pulso_fin - pulso_inicio

distancia = (duracion * 34300) / 2

if distancia < 2 or (round(distancia) > 300):
    print("Distancia fuera de rango")
    print("-----")
else:
    distancia = format((duracion * 34300) / 2, '.2f')
    print(("La distancia es de:"), distancia, ("cm"))
    print("-----")
time.sleep(sleepTime)

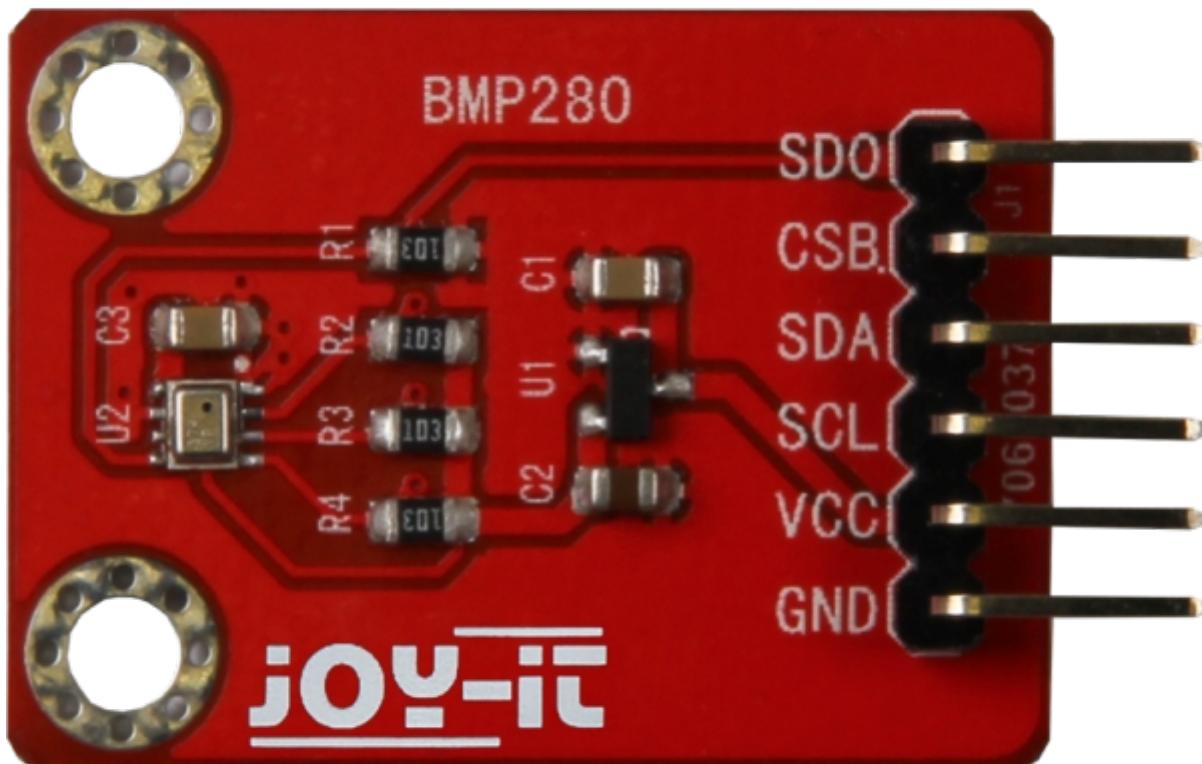
except KeyboardInterrupt:
    GPIO.cleanup()
```

Guardamos el programa con el nombre KY-051.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-051.py*

## KY-052: Módulo Sensor digital de presión atmosférica y temperatura BMP280



### Descripción:

- MEDICIONES DE PRESIÓN BAROMÉTRICA - Mida la presión atmosférica desde 300 hPa hasta 1100 hPa, lo que equivale a una altitud de 9000 m a -500 m sobre el nivel del mar con el sensor BMP280 de Bosch. Tiene una alta resolución de 0,16 Pa.

# Luciano Rabassa

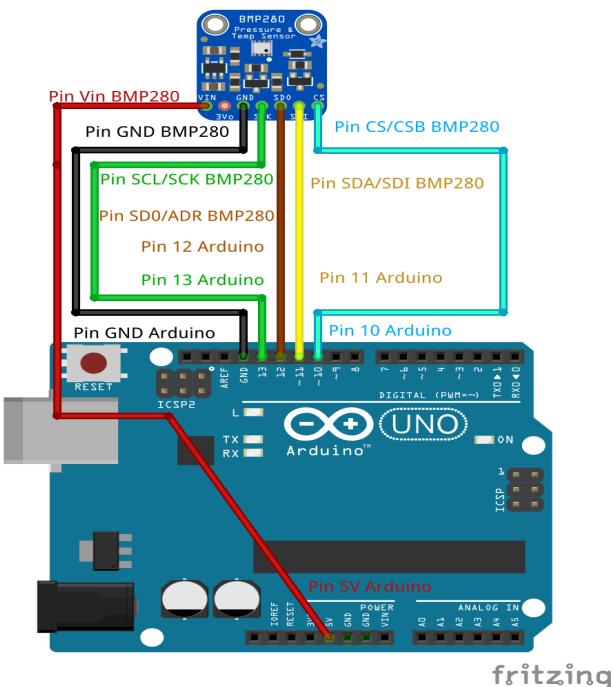
- **MEDICIONES DE TEMPERATURA** - La temperatura puede medirse entre  $-40^{\circ}\text{C}$  y  $85^{\circ}\text{C}$  con una resolución de  $0.01^{\circ}\text{C}$  y una precisión de  $\pm 1.0^{\circ}\text{C}$ . La frecuencia de medición es de 157 Hz y ha aumentado en comparación con su predecesor.
- **DETECCIÓN DE ALTA PRECISIÓN**: el sensor ya ha sido calibrado para usted. Simplemente conéctelo a la placa del micro controlador y comience a medir sin ajustar las curvas. Como un altímetro, puede medir los cambios de altitud con una precisión de  $\pm 1$  metro.
- **FÁCIL ACCESO**: la placa BMP280 se puede usar fácilmente con cualquier micro controlador como Arduino o Raspberry Pi funcionando a una tensión de funcionamiento de 1.8 V a 3.6 V. La comunicación entre el sensor y el dispositivo se realiza a través de I2C o SPI.
- **FÁCIL ACTUALIZACIÓN**: el sensor BMP280 de Bosch es el sucesor de los conjuntos de chips sazonados BMP085, BMP180 y BMP183. Cambie a BMP280 ahora para beneficiarse de una mayor precisión, precisión y reducción del ruido de medición.

Datasheet:

<https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf>

Conexión Arduino:

KY-052: Módulo BMP280



Raspberry Pi Buenos Aires



# Luciano Rabassa

KY-052	a	Arduino
SDO/ADR	a	Pin Digital 12
CSB/CS	a	Pin Digital 10
SDA/SDI	a	Pin Digital 11
SCL/SCK	a	Pin Digital 13
VCC/Vin	a	Pin 5V
GND	a	Pin GND

## Código Arduino:

Requiere librería Adafruit BMP280:

[https://github.com/adafruit/Adafruit\\_BMP280\\_Library.git](https://github.com/adafruit/Adafruit_BMP280_Library.git)

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#define BMP_SCK 13
#define BMP_MISO 12
#define BMP_MOSI 11
#define BMP_CS 10

Adafruit_BMP280 bmp(BMP_CS, BMP_MOSI, BMP_MISO, BMP_SCK);

void setup(){
    Serial.begin(9600);

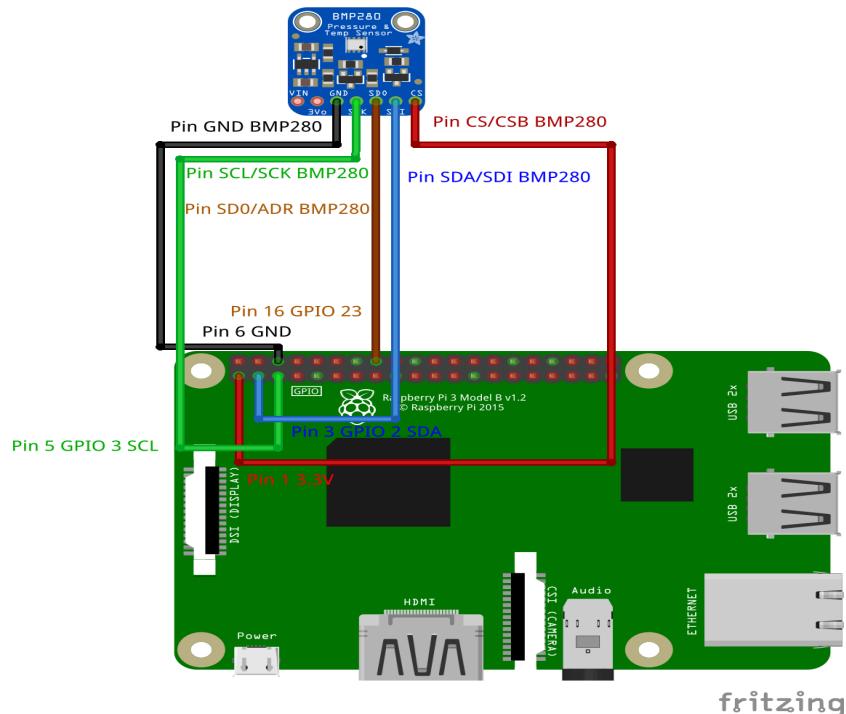
    if(!bmp.begin())
    {
        Serial.println(F("No se encuentra al sensor BMP280, verifica el conexionado!"));
        while (1);
    }
}

void loop(){
    Serial.print(F("Temperatura = "));
    Serial.print(bmp.readTemperature());
    Serial.println(" *C");
    Serial.print(F("Presion = "));
    Serial.print(bmp.readPressure());
    Serial.println(" Pa");
    Serial.println();
    delay(2000);
}
```

# Luciano Rabassa

## Conexión Raspberry Pi:

KY-052: Módulo BMP280



KY-052	a	Raspberry Pi
SDO/ADR	a	Pin 16 GPIO23
CSB/CS	a	Pin 1 3.3V
SDA/SDI	a	Pin 3 GPIO2(Pin SDA en Raspberry Pi)
SCL/SCK	a	Pin 5 GPIO3(Pin SCL en Raspberry Pi)
VCC/Vin	a	NO SE CONECTA
GND	a	Pin 6 GND

## Código Raspberry Pi:

**Requisito librería Adafruit CircuitPython BMP280:**

[https://github.com/adafruit/Adafruit\\_CircuitPython\\_BMP280.git](https://github.com/adafruit/Adafruit_CircuitPython_BMP280.git)

Se necesita activar la interfaz I2C

Vamos a Raspberry Pi → Preferencias → Configuración de Raspberry Pi → Interfaces y activamos I2C, luego reiniciamos

```
import time
import board
# import digitalio #Para usar con SPI
import busio
import adafruit_bmp280

# O creamos la variable spi para usar el puerto Bus SPI
#spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
#bmp_cs = digitalio.DigitalInOut(board.D10)
#bmp280 = adafruit_bmp280.Adafruit_BMP280_SPI(spi, bmp_cs)

# Creamos la variable i2c para utilizar el puerto Bus I2C
i2c = busio.I2C(board.SCL, board.SDA)
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)
# Cambiar esto para coincidir con el lugar de la presión (hPa) a nivel del mar
bmp280.sea_level_pressure = 1013.25

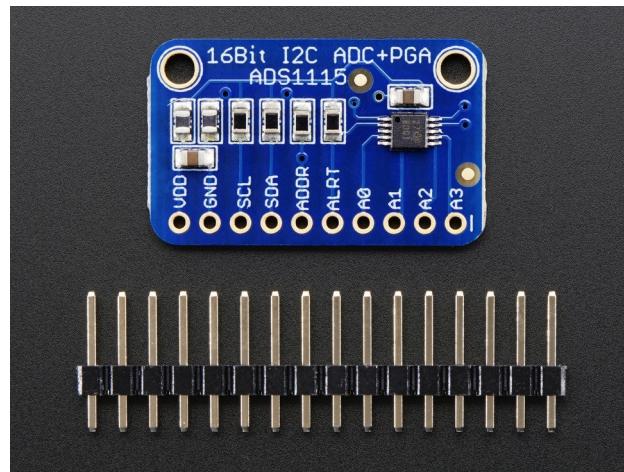
while True:
    print("\nTemperatura: %0.1f C" % bmp280.temperature)
    print("Presion: %0.1f hPa" % bmp280.pressure)
    print("Altitud = %0.2f metros" % bmp280.altitude)
    time.sleep(2)
```

Guardamos el programa con el nombre KY-052.py

Para correr el programa abrimos la Terminal y tecleamos:

- *sudo python3 KY-052.py*

## KY-053: Módulo ADS1115 Convertidor Analógico-Digital



### Descripción:

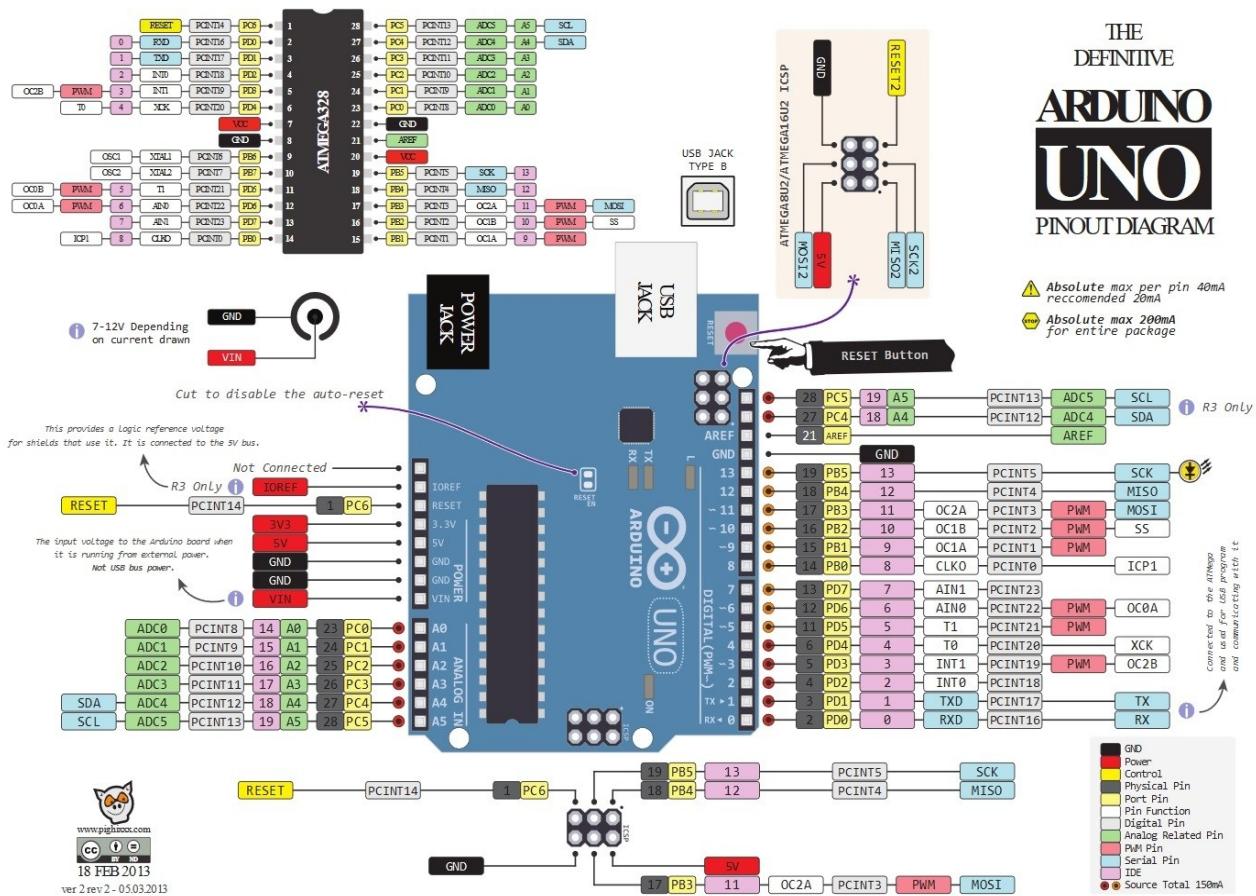
Arduino ya posee un convertidor lógico de 10 bit con 6 canales, pero si necesitas más y mayor precisión, el ADS1115 es una buena opción. Posee 4 canales y una precisión de 16 bit. En cambio, la Raspberry Pi, carece de uno, por lo cuál es una solución para conectar nuestros dispositivos analógicos.

Para Arduino debemos instalar la siguiente librería:

[https://github.com/adafruit/Adafruit\\_ADS1X15](https://github.com/adafruit/Adafruit_ADS1X15)

## Pinout Arduino Uno R3:

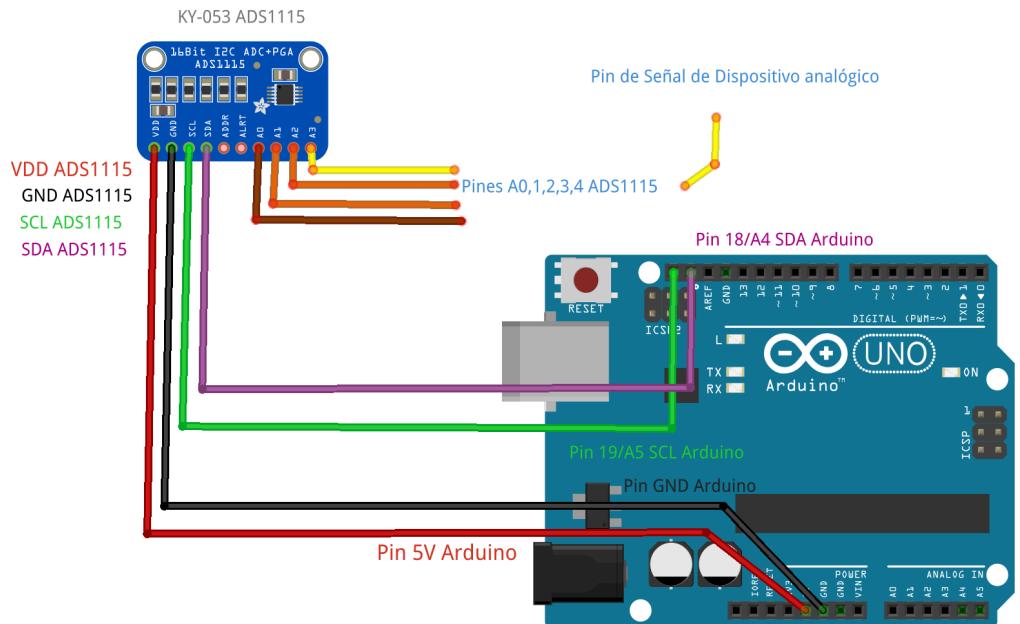
Podemos observar que los pines situados bajo el botón reset, son pines analógicos: Pin 18 (A4) , Pin 19 (A5).



# Luciano Rabassa

## Conección Arduino:

KY-053: Módulo ADS1115 Convertidor Analógico-Digital



fritzing

KY-053	a	Arduino
VDD	a	Pin 5V
GND	a	Pin GND
SCL	a	Pin SCL (A5 o Pin 19)
SDA	a	Pin SDA(A4 o Pin 18)
ADDR	a	No se conecta
ALRT	a	No se conecta
A0	a	Pin Analógico A0
A1	a	Pin Analógico A1
A2	a	Pin Analógico A2
A3	a	Pin Analógico A3

# Luciano Rabassa

## Código Arduino:

```
#include <Adafruit_ADS1015.h>

Adafruit_ADS1115 ads;

void setup(void)
{
    Serial.begin(9600);

    Serial.println("Valores de las entradas analogas del ADS1115");
    (A0..A3) seran leidas y mostradas en el monitor serie");
    Serial.println("Rango ADC: +/- 6.144V (1 bit = 0.1875mV)");

/* El modulo incluye un amplificador de señal en las entradas, que pueden ser
configuradas en la sección de abajo vía software
Esto se busca si esperas un rango específico de tensión como resultado de la
medición, para obtener una resolución mayor de señal de amplificación es elegida
por default debes marcar la activa y comentar las demás */

    ads.setGain(GAIN_TWOTHIRDS); // 2/3x gain +/- 6.144V 1 bit = 0.1875mV
    // ads.setGain(GAIN_ONE); // 1x gain +/- 4.096V 1 bit = 0.125mV
    // ads.setGain(GAIN_TWO); // 2x gain +/- 2.048V 1 bit = 0.0625mV
    // ads.setGain(GAIN_FOUR); // 4x gain +/- 1.024V 1 bit = 0.03125mV
    // ads.setGain(GAIN_EIGHT); // 8x gain +/- 0.512V 1 bit = 0.015625mV
    // ads.setGain(GAIN_SIXTEEN); // 16x gain +/- 0.256V 1 bit= 0.0078125mV

    ads.begin();
}

void loop(void)
{
    uint16_t adc0, adc1, adc2, adc3;
    float voltage0, voltage1, voltage2, voltage3;
    float gain_conversion_factor;

/* El comando ads.readADC_SingleEnded(0) es la operación que comienza la
medición del ADC
La variable con el valor 0 define el canal medido
Si queremos medir el tercer canal colocaremos 3 en lugar de cero */
    adc0 = ads.readADC_SingleEnded(0);
    adc1 = ads.readADC_SingleEnded(1);
    adc2 = ads.readADC_SingleEnded(2);
    adc3 = ads.readADC_SingleEnded(3);

    // Se necesita este valor para calcular la tensión
    gain_conversion_factor = 0.1875;

    // Calculo de los valores de tensión desde los valores medidos
    voltage0 = (adc0 * gain_conversion_factor);
    voltage1 = (adc1 * gain_conversion_factor);
    voltage2 = (adc2 * gain_conversion_factor);
```

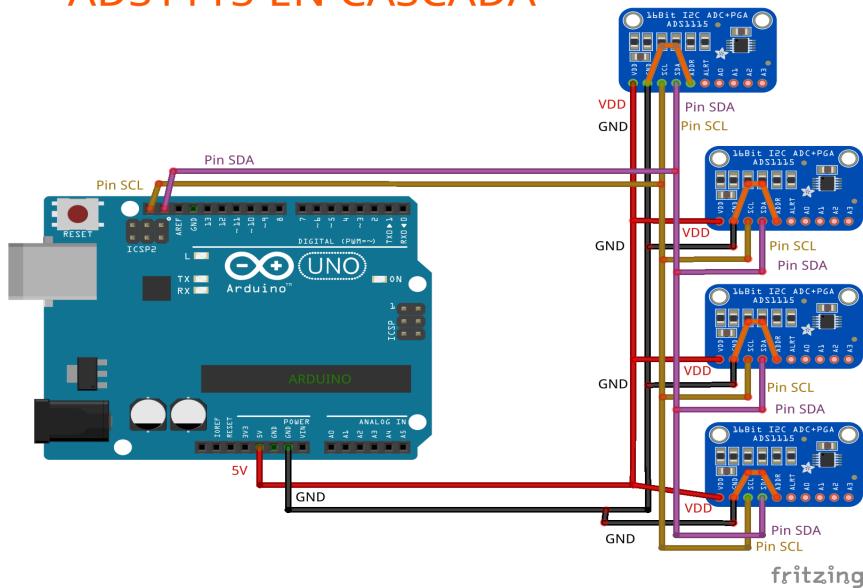
# Luciano Rabassa

```
voltage3 = (adc3 * gain_conversion_factor);

// Los valores se mostraran por el monitor serie
Serial.print("Entrada Analoga 0: "); Serial.print(voltage0);
Serial.print("mV");
Serial.print("Entrada Analoga 1: "); Serial.print(voltage1);
Serial.print("mV");
Serial.print("Entrada Analoga 2: "); Serial.print(voltage2);
Serial.print("mV");
Serial.print("Entrada Analoga 3: "); Serial.print(voltage3);
Serial.print("mV");
Serial.println("-----");
delay(1000);
}
```

Ejemplo con cuatro placas:

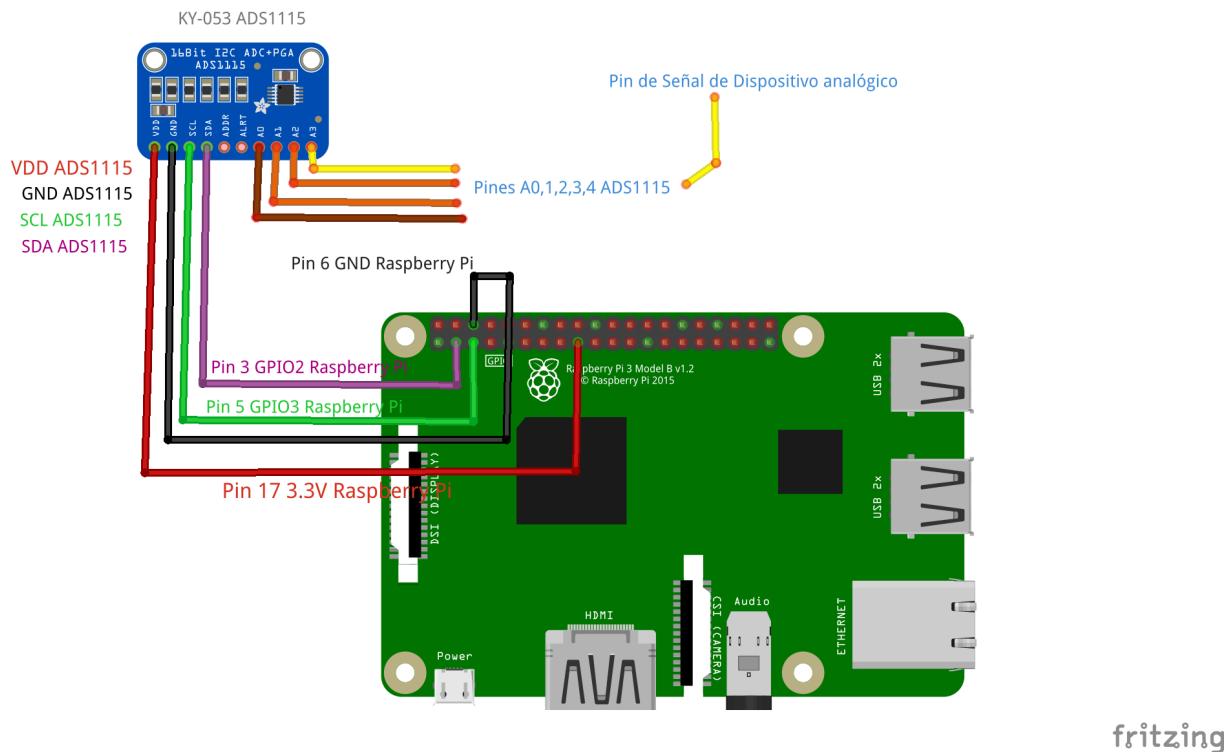
## ADS1115 EN CASCADA



# Luciano Rabassa

## Conexión Raspberry Pi:

KY-053: Módulo ADS1115 Convertidor Analógico-Digital



KY-053	a	Raspberry Pi
VDD	a	Pin 17 3.3V
GND	a	Pin 6 GND
SCL	a	Pin 5 GPIO3(Pin SCL)
SDA	a	Pin 3 GPIO2(Pin SDA)
ADDR	a	No se conecta
ALRT	a	No se conecta
A0	a	Salida Analógica a algún dispositivo
A1	a	Salida Analógica a algún dispositivo
A2	a	Salida Analógica a algún dispositivo
A3	a	Salida Analógica a algún dispositivo

Para Raspberry Pi debemos activar la Interfaz I2C:

Vamos a Raspberry Pi → Preferencias → Configuración de Raspberry Pi → Interfaces y activamos I2C, luego reiniciamos.

Librería Adafruit Python ADS1X15:

[https://github.com/adafruit/Adafruit\\_Python\\_AM2301](https://github.com/adafruit/Adafruit_Python_AM2301)

Código Raspberry Pi:

```
from Adafruit_AM2301 import AM2301
from time import sleep
import time, signal, sys, os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

retraso = 0.5

# Asignar el ADS1x15 ADC
ADS1015 = 0x00
ADS1115 = 0x01

# Elegir la amplificación gain
gain = 4096      # +/- 4.096V
# gain = 2048    # +/- 2.048V
# gain = 1024    # +/- 1.024V
# gain = 512     # +/- 0.512V
# gain = 256     # +/- 0.256V

# Elegir el radio de muestreo
# sps = 8        # 8 Muestras por segundo
# sps = 16       # 16 Muestras por segundo
# sps = 32       # 32 Muestras por segundo
sps = 64         # 64 Muestras por segundo
# sps = 128      # 128 Muestras por segundo
# sps = 250      # 250 Muestras por segundo
# sps = 475      # 475 Muestras por segundo
# sps = 860      # 860 Muestras por segundo

# Asignando el canal ADC-Channel (1-4)
adc_channel_0 = 0      # Canal 0
adc_channel_1 = 1      # Canal 1
adc_channel_2 = 2      # Canal 2
adc_channel_3 = 3      # Canal 3

# Inicializa ADC (ADS1115)
adc = AM2301(ic=ADS1115)
```

# Luciano Rabassa

```
try:
    while True:
        # Leemos los valores de cada ADC
        adc0 = adc.readADCSingleEndedadc_channel_0, gain, sps)
        adc1 = adc.readADCSingleEndedadc_channel_1, gain, sps)
        adc2 = adc.readADCSingleEndedadc_channel_2, gain, sps)
        adc3 = adc.readADCSingleEndedadc_channel_3, gain, sps)

        # Lo mostramos en la consola
        print("Canal 0:", adc0, "mV ")
        print("Canal 1:", adc1, "mV ")
        print("Canal 2:", adc2, "mV ")
        print("Canal 3:", adc3, "mV ")
        print("-----")
        time.sleep(retraso)

except KeyboardInterrupt:
    GPIO.cleanup()
```

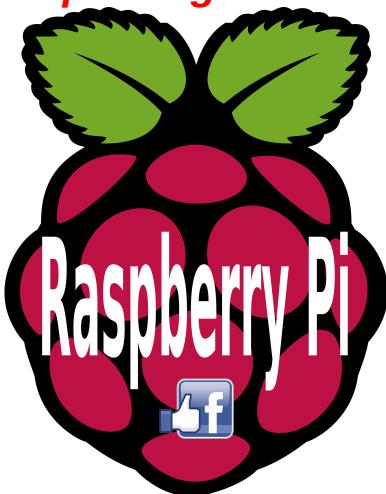
Guardamos el programa con el nombre KY-053.py

Para correr el programa abrimos la Terminal y tecleamos:

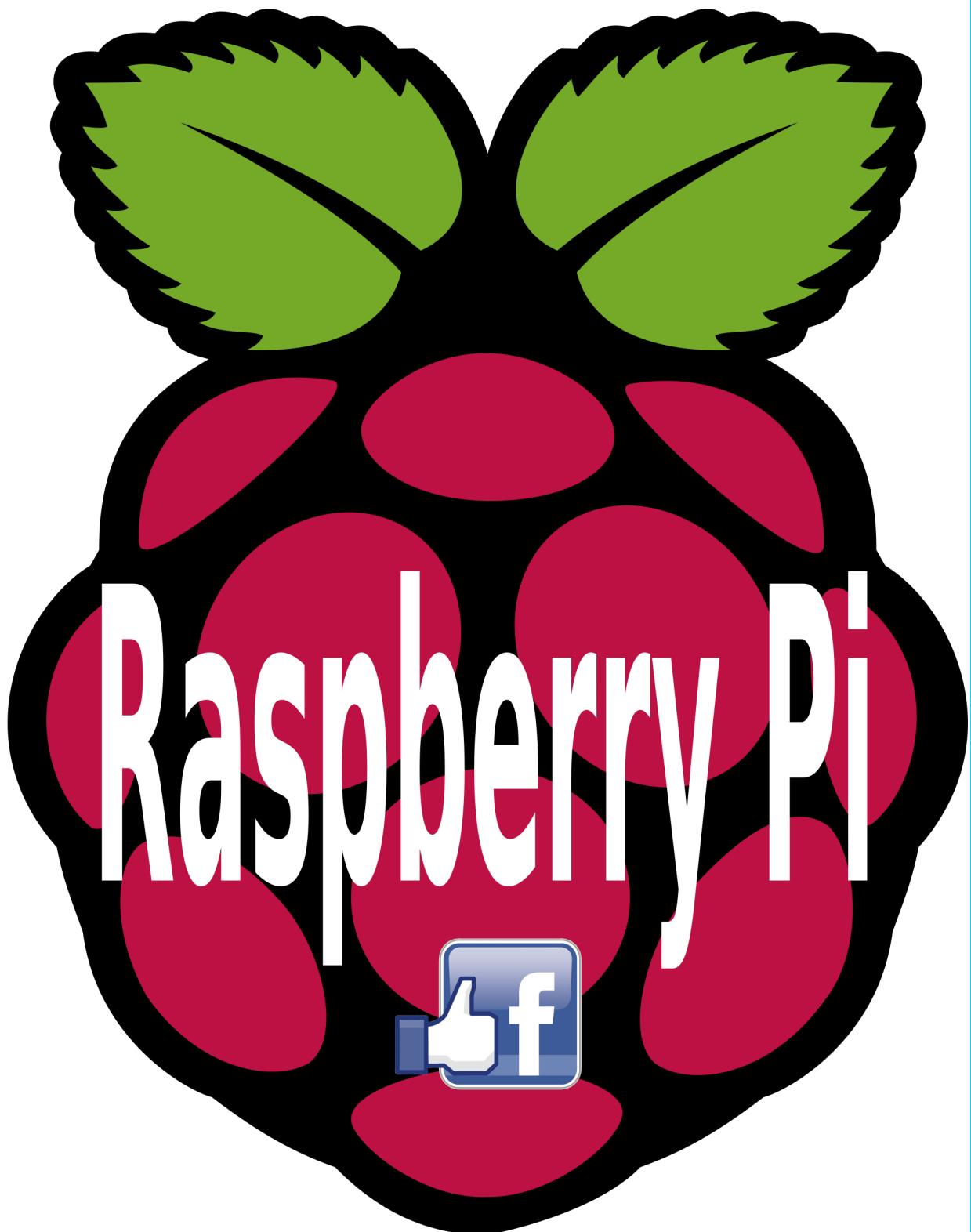
- *sudo python3 KY-053.py*

*Diagramas y códigos en [GitHub](#) hyperlinkados en cada foto de Fritzing y en los titulos.*

*Gracias por llegar hasta aca!*



**Luciano Rabassa**



**Raspberry Pi Buenos Aires**