

# SensorKit X40

[Accueil](#)

Cher client,

Merci d'avoir acheté notre produit.

Ce kit de capteurs de haute qualité est spécialement conçu pour les plateformes open-source les plus populaires du moment. Il est compatible avec [Arduino](#), [Raspberry Pi](#) (tous les modèles), [Banana Pi](#), [Latte Panda](#), [Cubieboard](#), [Beaglebone](#), [pcDuino](#) et beaucoup d'autres systèmes à microcontrôleurs (Atmega, MicroChip PIC, STM32, etc.).

Ce guide comprend la description technique et le fonctionnement de chaque capteur du kit : affectation des broches, composants, utilisation, ...

Chaque module est proposé avec un programme d'exemple commenté pour une utilisation avec une carte Arduino (écrit en C++) et une carte Raspberry Pi (écrit en Python).

Vous pouvez trouver le code de ces programmes dans ce guide ou les télécharger directement [ici](#) pour Arduino et [ici](#) pour Raspberry.

Une version en ligne de ce guide est disponible à l'adresse : <http://sensorkit.fr.joy-it.net>

Ce guide est conçu pour que les programmeurs débutants puissent rapidement développer leurs propres projets et expériences.

Contrairement à l'Arduino, le Raspberry Pi, ne supporte pas les capteurs analogiques et les modules 5V. Pour contourner ce problème, deux modules spécifiques sont présents dans ce kit :

- le convertisseur numérique/analogique KY-053 autorise l'utilisation des capteurs analogiques
- le convertisseur de tension KY-051 permet d'utiliser les modules ayant une tension différente de 3,3 V

Les modules de ce kit sont utilisables avec une plaque de montage rapide pour réaliser toutes les expériences que vous voulez, mais vous pouvez également souder les broches pour concrétiser vos projets.

Nous vous souhaitons beaucoup de découvertes et d'amusement avec SensorKit X40.

L'équipe Go Tronic

Accueil

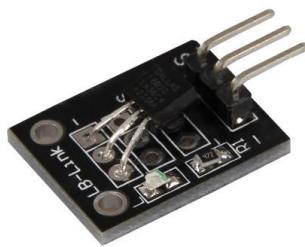
Accueil

---

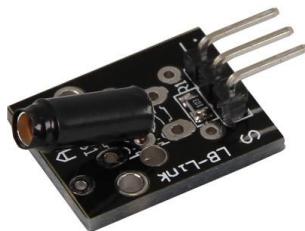
# SensorKit

## X40

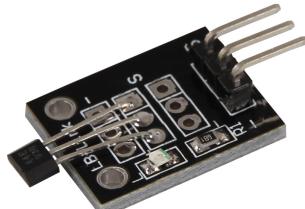
*Cliquez sur l'image ou la description pour aller à la fiche du capteur*



**KY-001** Capteur de température

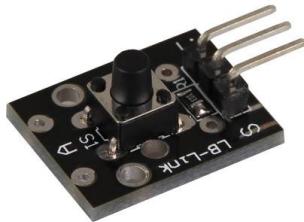


**KY-002** Capteur de vibration

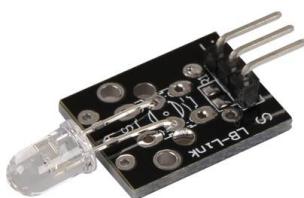


**KY-003** Capteur à effet Hall

Accueil



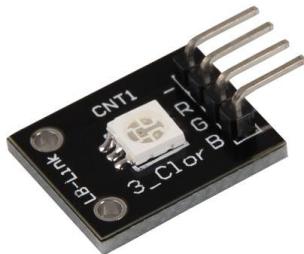
**KY-004** Module bouton-poussoir



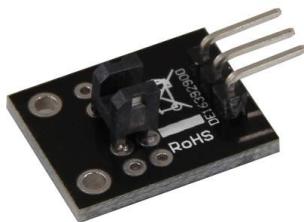
**KY-005** Module transmetteur infrarouge



**KY-006** Module Buzzer passif



**KY-009** Module led RGB



**KY-010** Capteur optique à fourche

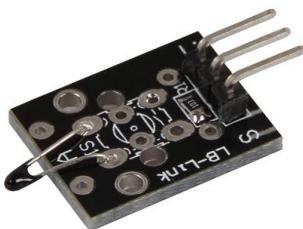
Accueil



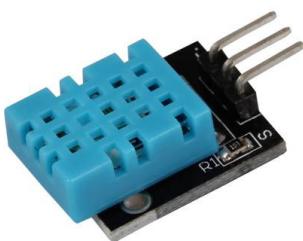
**KY-011** Module led bicolore rouge/vert



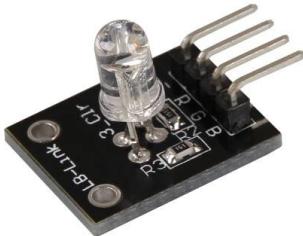
**KY-012** Module buzzer actif



**KY-013** Capteur de température CTN

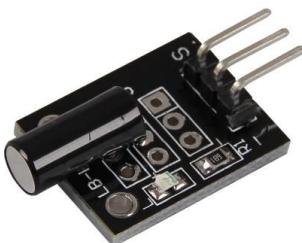


**KY-015** Capteur de t° et d'humidité DHT-11

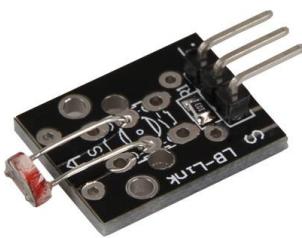


**KY-016** Module led RGB 5 mm

Accueil



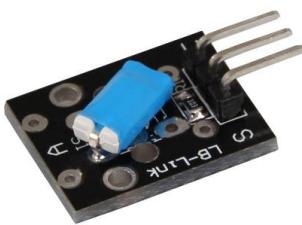
**KY-017** Capteur d'inclinaison



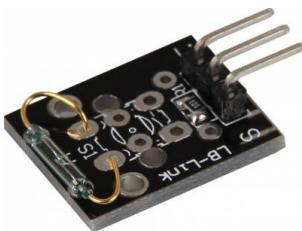
**KY-018** Module à photorésistance LDR



**KY-019** Module relais 5V



**KY-020** Capteur d'inclinaison



**KY-021** Capteur magnétique Reed

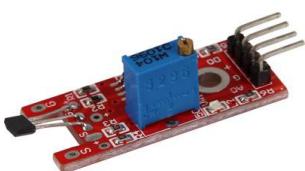


**KY-022** Module récepteur infrarouge

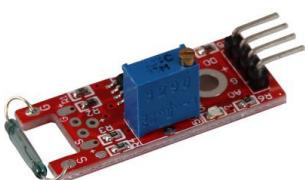
Accueil



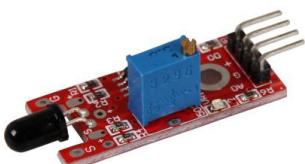
**KY-023** Module joystick X-Y



**KY-024** Capteur à effet Hall



**KY-025** Module Reed

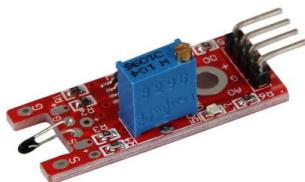


**KY-026** Détecteur de flamme

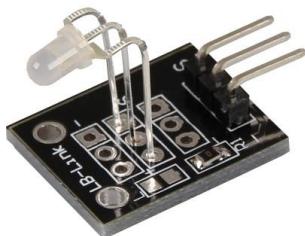


**KY-027** Module Magic Light Cup

Accueil



**KY-028** Module capteur de t° CTN



**KY-029** Module led bicolore rouge/vert



**KY-031** Capteur de vibrations/chocs



**KY-032** Capteur d'obstacles IR

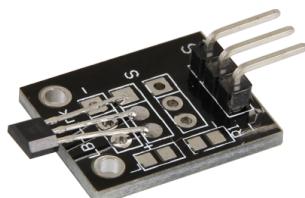


**KY-033** Module suiveur de ligne

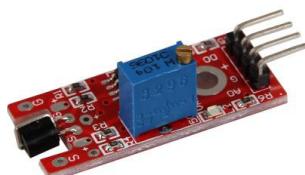
Accueil



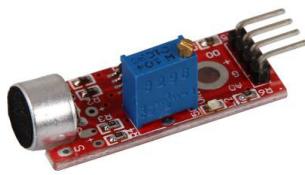
**KY-034** Module led flash (7 couleurs)



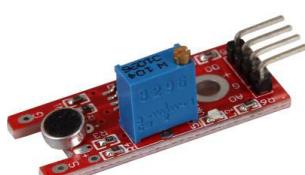
**KY-035** Capteur magnétique analogique



**KY-036** Capteur tactile



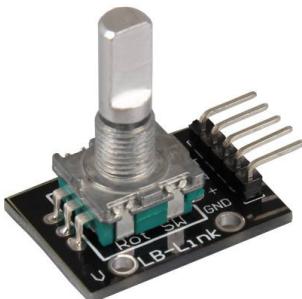
**KY-037** Capteur sonore



**KY-038** Capteur sonore



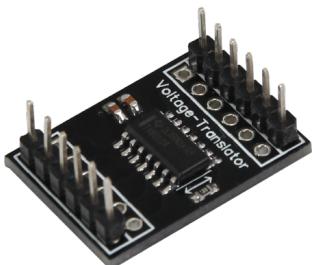
**KY-039** Capteur de pulsation cardiaque



**KY-040** Module encodeur rotatif



**KY-050** Capteur à ultrasons



**KY-051** Voltage Translator / Level Shifter



**KY-052** Capteur de pression BMP180



**KY-053** Convertisseur analogique digital

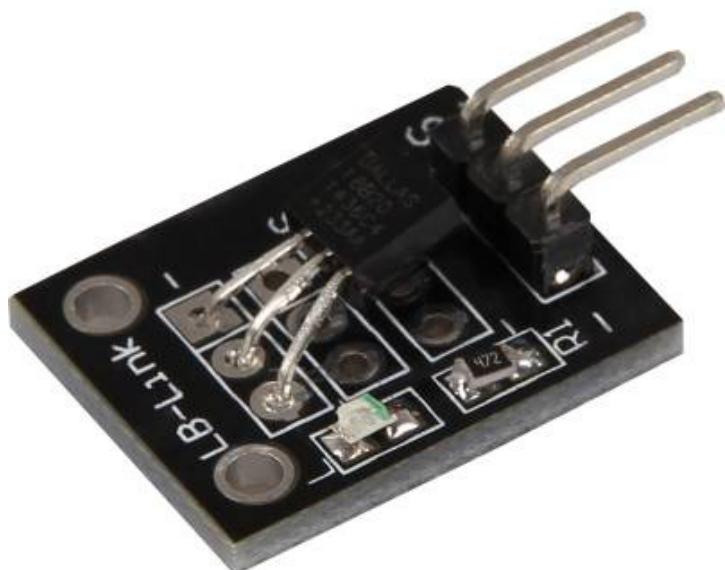
## KY-001 Capteur de température

## KY-001 Capteur de température

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Configuration One-Wire pour Raspberry Pi .....	3
6 Exemple de code pour Raspberry Pi .....	3

### Photo



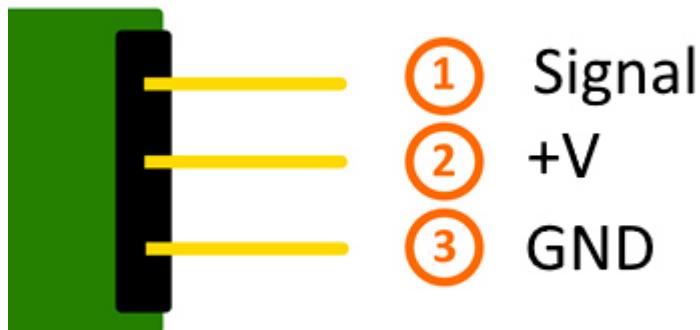
### Données techniques / Description sommaire

Chipset: DS18B20 | Interface: 1-Wire

Mesure précise de la température en 9- 12Bits de -55°C à +125°C

## KY-001 Capteur de température

### Brochage



### Exemple de code pour Arduino

Pour l'exemple de code suivant, deux bibliothèques supplémentaires sont nécessaires:

- [OneWire Library] de [Paul Stoffregen](#) | publié sous la licence MIT
- [Dallas Temperature Control Library] de [Miles Burton](#) | publié sous LGPL

Les deux librairies sont incluses et doivent être copiées dans le dossier "libraries" avant de lancer l'IDE Arduino.

Elles peuvent être trouvées par défaut dans Windows:

C:\Program Files (x86)\Arduino\libraries

```
// Importation des librairies requises
#include <DallasTemperature.h>
#include <OneWire.h>

// Déclaration de la broche d'entrée du capteur
#define KY001_Signal_PIN 4

// Configuration des librairies
OneWire oneWire(KY001_Signal_PIN);
DallasTemperature sensors(&oneWire);

void setup() {
    // Initialisation de l'interface série
    Serial.begin(9600);
    Serial.println("KY-001 Capteur de température");

    // Le capteur est initialisé
    sensors.begin();
}
```

## KY-001 Capteur de température

```

}

//Boucle de programme principale
void loop()
{
    // Début de la mesure de la température
    sensors.requestTemperatures();
    // ... et affichage de la température mesurée
    Serial.print("Temperature: ");
    Serial.print(sensors.getTempCByIndex(0));
    Serial.write(176); // Code Unicode pour le symbole °
    Serial.println("C");

    delay(1000);      // 5s de pause avant la mesure suivante
}

```

### Affectation des broches Arduino:

Sensor Signal	= [Pin 4]
Sensor +V	= [Pin 5V]
Sensor -	= [Pin GND]

### Exemple de programme à télécharger

[KY-001-TemperaturSensor.zip](#)

## Configuration One-Wire pour Raspberry Pi

Afin que la Raspberry Pi puisse envoyer la valeur mesurée par le DS18B20, il est nécessaire de configurer au préalable le bus One-Wire. Le fichier "/boot/config.txt" doit être édité et complété par la ligne suivante:

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

Le fichier peut être modifié en saisissant la commande ...

```
sudo nano /boot/config.txt
```

... dans la console. Utilisez la combinaison de touches [Ctrl + X] pour quitter l'édition et enregistrez avec [Ctrl + Y].

Dès que vous aurez redémarré la carte Raspberry Pi ...

```
sudo reboot
```

... vous pourrez utiliser l'exemple ci-dessous.

## Exemple de code pour Raspberry Pi

Exemple de programmation en Python

## KY-001 Capteur de température

```

# coding=utf-8
# Les modules nécessaires sont importés et mis en place
import glob
import time
from time import sleep
import RPi.GPIO as GPIO

# À ce stade, la pause peut être réglée entre deux mesures
sleeptime = 1

# La broche d'entrée One-Wire est déclarée et la résistance intégrée de Pull UP activée
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Après l'activation, la résistance de pull-up est maintenue
# jusqu'à ce que la communication soit établie avec le capteur DS18B20
print 'En attente d initialisation...'

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'

# Définition de la fonction permettant de lire la valeur du capteurF
def TemperaturMessung():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

# Pour l'initialisation, le capteur est lu une fois en "aveugle"
TemperaturMessung()

# Evaluation de la température: Beim Raspberry Pi werden erkannte one-Wire Slaves im Ordner
# /sys/bus/w1/devices/ einem eigenen Unterordner zugeordnet. Ce dossier contient le fichier w1-esclave
# dans lequel les données qui ont été envoyées par le bus One-Wire sont stockées.
# Dans cette fonction, les données sont analysées et la température est lue et envoyée
def TemperaturAuswertung():
    lines = TemperaturMessung()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = TemperaturMessung()
    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

# Boucle de programme principale
# La température mesurée est affichée dans la console - une pause paramétrable
# avec "sleeptime" est instaurée entre les mesures
try:
    while True:
        print '-----'
        print "Temperature:", TemperaturAuswertung(), "°C"
        time.sleep(sleeptime)

except KeyboardInterrupt:
    GPIO.cleanup()

```

## KY-001 Capteur de température

**Brochage Raspberry Pi:**

Signal	=	GPIO4	[Pin 7]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

**Exemple de programme à télécharger**[KY-001\\_RPi\\_TemperaturSensor.zip](#)

Commande pour lancer le programme:

```
sudo python KY-001_RPi_TemperaturSensor.py
```

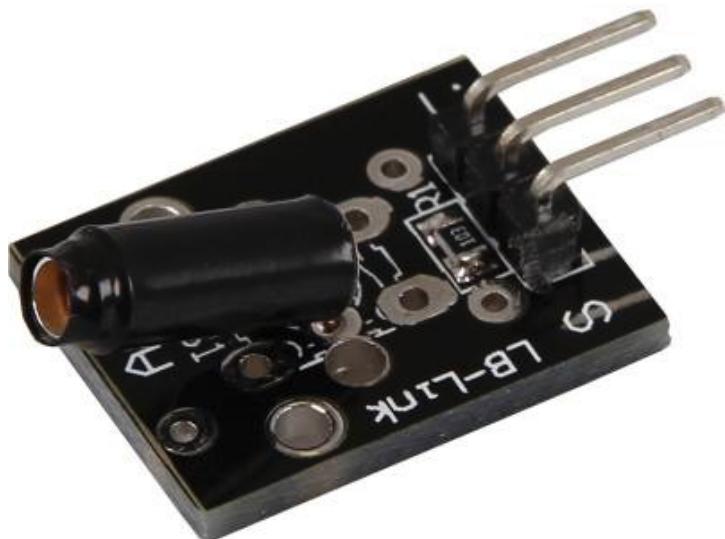
KY-002 Capteur de vibration

## KY-002 Capteur de vibration

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

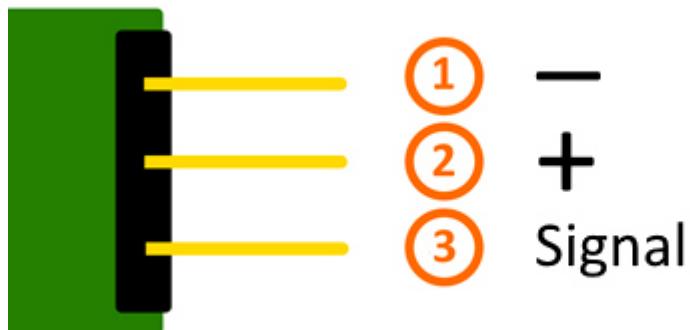
### Photo



### Données techniques / Description sommaire

Lors d'une vibration, le contact interne se ferme.

## Brochage



## Exemple de code pour Arduino

Voici un exemple de programme qui fait allumer une LED lorsqu'un signal est détecté au niveau du capteur. On peut utiliser les modules à Leds KY-011, KY-016 ou KY-029.

```
int Led = 13 ;// Déclaration de la broche de sortie LED
int Sensor = 10; // Déclaration de la broche d'entrée du capteur
int val; // Variable temporaire

void setup ()
{
    pinMode (Led, OUTPUT) ; // Initialisation de la broche de sortie
    pinMode (Sensor, INPUT) ; // Initialisation de la broche du capteur
    digitalWrite(Sensor, HIGH); // Activation de la résistance de Pull-up interne
}

void loop ()
{
    val = digitalRead (Sensor) ; // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        digitalWrite (Led, LOW);
    }
    else
    {
        digitalWrite (Led, HIGH);
    }
}
```

### Affectation des broches Arduino:

LED +	= [Pin 13]
LED -	= [Pin GND]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]

## KY-002 Capteur de vibration

Sensor - = [Pin GND]

### Exemple de programme à télécharger

[SensorTest\\_Arduino.zip](#)

## Exemple de code pour Raspberry Pi

Exemple de programmation en Python

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée. En outre la résistance de Pull-up est activée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Cette fonction de sortie est exécutée par détection du signal
def ausgabeFunktion(null):
    print("Signal détecté")

# Lors de la détection d'un signal (front descendant du signal) de la fonction de sortie est déclenchée
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=100)

# Boucle de programme principale
try:
    while True:
        time.sleep(1)

# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger

[SensorTest\\_RPi.zip](#)

Commande pour lancer le programme:

```
sudo python SensorTest_RPi.py
```

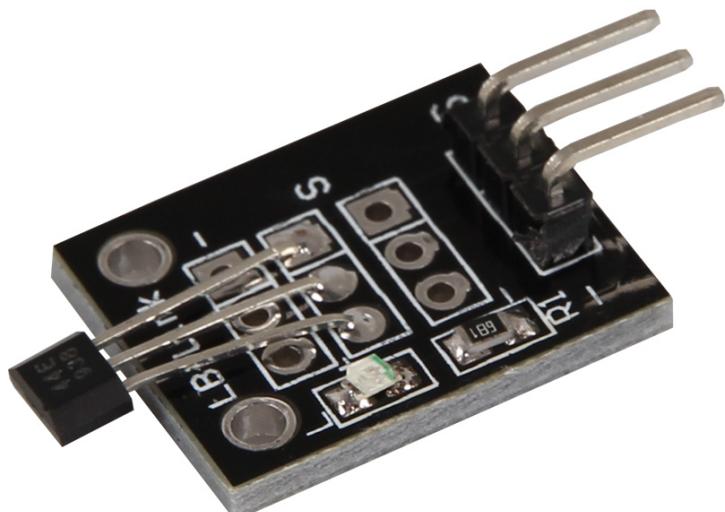
KY-003 Capteur à effet Hall

## KY-003 Capteur à effet Hall

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo



### Données techniques / Description sommaire

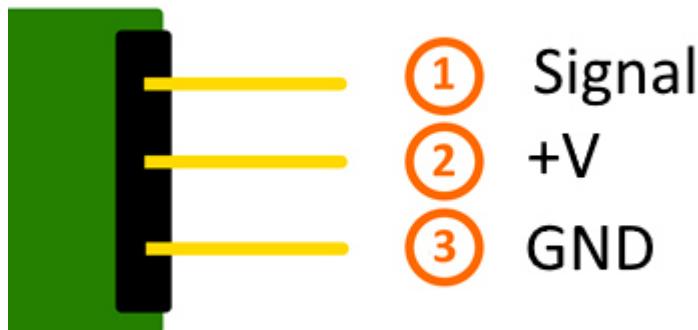
Circuit intégré: A3141

Capteur à effet hall

Le capteur s'active si le module est maintenu dans un champ magnétique. Le signal de sortie peut être lu en tant que valeur de tension analogique.

KY-003 Capteur à effet Hall

## Brochage



## Exemple de code pour Arduino

Voici un exemple de programme qui fait allumer une LED lorsqu'un signal est détecté au niveau du capteur. On peut utiliser les modules à Leds KY-011, KY-016 ou KY-029.

```
int Led = 13; // Déclaration de la broche de sortie LED
int Sensor = 10; // Déclaration de la broche d'entrée du capteur
int val; // Variable temporaire

void setup ()
{
    pinMode (Led, OUTPUT); // Initialisation de la broche de sortie
    pinMode (Sensor, INPUT); // Initialisation de la broche du capteur
    digitalWrite(Sensor, HIGH); // Activation de la résistance de Pull-up interne
}

void loop ()
{
    val = digitalRead (Sensor); // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        digitalWrite (Led, LOW);
    }
    else
    {
        digitalWrite (Led, HIGH);
    }
}
```

### Affectation des broches Arduino:

LED +	= [Pin 13]
LED -	= [Pin GND]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]

## KY-003 Capteur à effet Hall

Sensor - = [Pin GND]

### Exemple de programme à télécharger

[SensorTest\\_Arduino.zip](#)

## Exemple de code pour Raspberry Pi

Exemple de programmation en Python

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée. En outre la résistance de Pull-up est activée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Cette fonction de sortie est exécutée par détection du signal
def ausgabeFunktion(null):
    print("Signal détecté")

# Lors de la détection d'un signal (front descendant du signal), la fonction de sortie est appelée
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=100)

# Boucle de programme principale
try:
    while True:
        time.sleep(1)

# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger

[SensorTest\\_RPi.zip](#)

Commande pour lancer le programme:

```
sudo python SensorTest_RPi.py
```

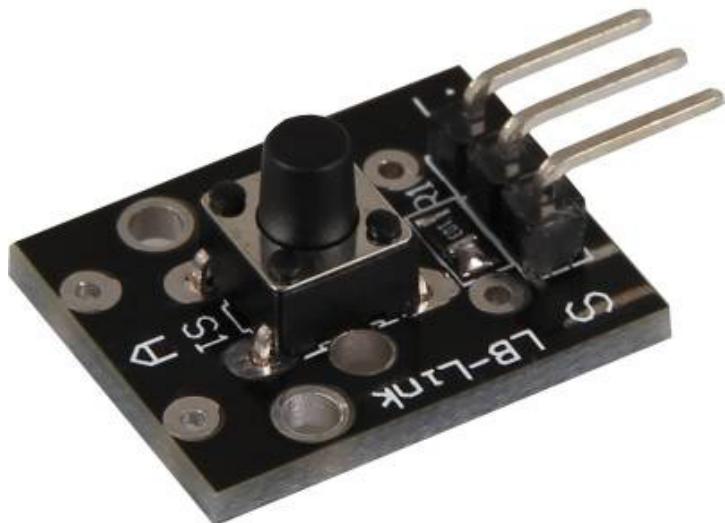
## KY-004 Module bouton-poussoir

## KY-004 Module bouton-poussoir

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

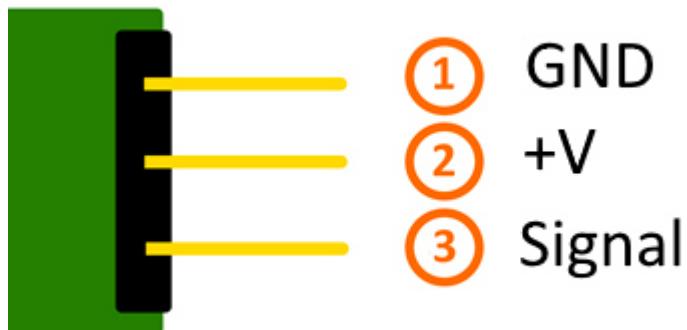
### Photo



### Données techniques / Description sommaire

En appuyant sur le bouton, le contact de sortie se ferme.

## Brochage



## Exemple de code pour Arduino

Voici un exemple de programme qui fait allumer une LED lorsqu'un signal est détecté au niveau du capteur. On peut utiliser les modules à Leds KY-011, KY-016 ou KY-029.

```
int Led = 13; // Déclaration de la broche de sortie LED
int Sensor = 10; // Déclaration de la broche d'entrée du capteur
int val; // Variable temporaire

void setup ()
{
    pinMode (Led, OUTPUT); // Initialisation de la broche de sortie
    pinMode (Sensor, INPUT); // Initialisation de la broche du capteur
    digitalWrite(Sensor, HIGH); // Activation de la résistance de Pull-up interne
}

void loop ()
{
    val = digitalRead (Sensor); // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        digitalWrite (Led, LOW);
    }
    else
    {
        digitalWrite (Led, HIGH);
    }
}
```

### Affectation des broches Arduino:

LED +	= [Pin 13]
LED -	= [Pin GND]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]

## KY-004 Module bouton-poussoir

Sensor - = [Pin GND]

### Exemple de programme à télécharger

[SensorTest\\_Arduino.zip](#)

## Exemple de code pour Raspberry Pi

Exemple de programmation en Python

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée. En outre la résistance de Pull-up est activée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Cette fonction de sortie est exécutée par détection du signal
def ausgabeFunktion(null):
    print("Signal détecté")

# Lors de la détection d'un signal (front descendant du signal) de la fonction de sortie est déclenchée
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=100)

# Boucle de programme principale
try:
    while True:
        time.sleep(1)

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger

[SensorTest\\_RPi.zip](#)

Commande pour lancer le programme:

```
sudo python SensorTest_RPi.py
```

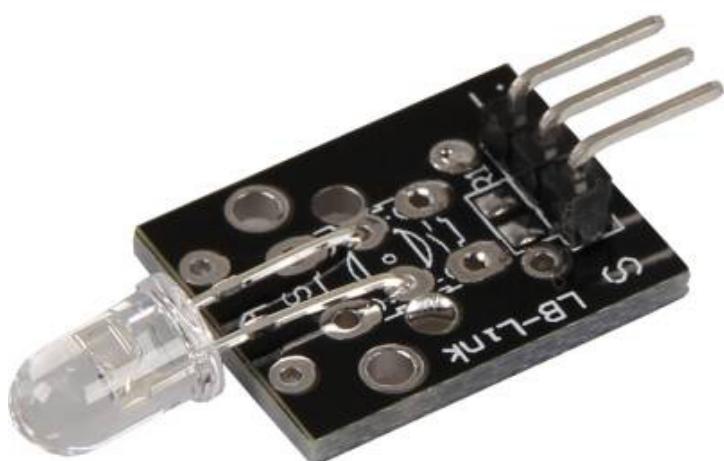
## KY-005 Module transmetteur infrarouge

## KY-005 Module transmetteur infrarouge

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	3
4 Exemple de code pour Arduino .....	4
4.1 Exemple de code ON/OFF .....	4
4.2 Exemple de code en fonction télécommande .....	4
5 Exemple de code pour Raspberry Pi .....	6
5.1 Exemple de code ON/OFF .....	6
5.2 Exemple de code en fonction télécommande .....	7
5.3 Installation de Lirc .....	7
5.4 Test du récepteur IR .....	9
5.5 Apprentissage du code de la télécommande .....	10
5.6 Envoyer des commandes avec l'émetteur infrarouge .....	11

### Photo



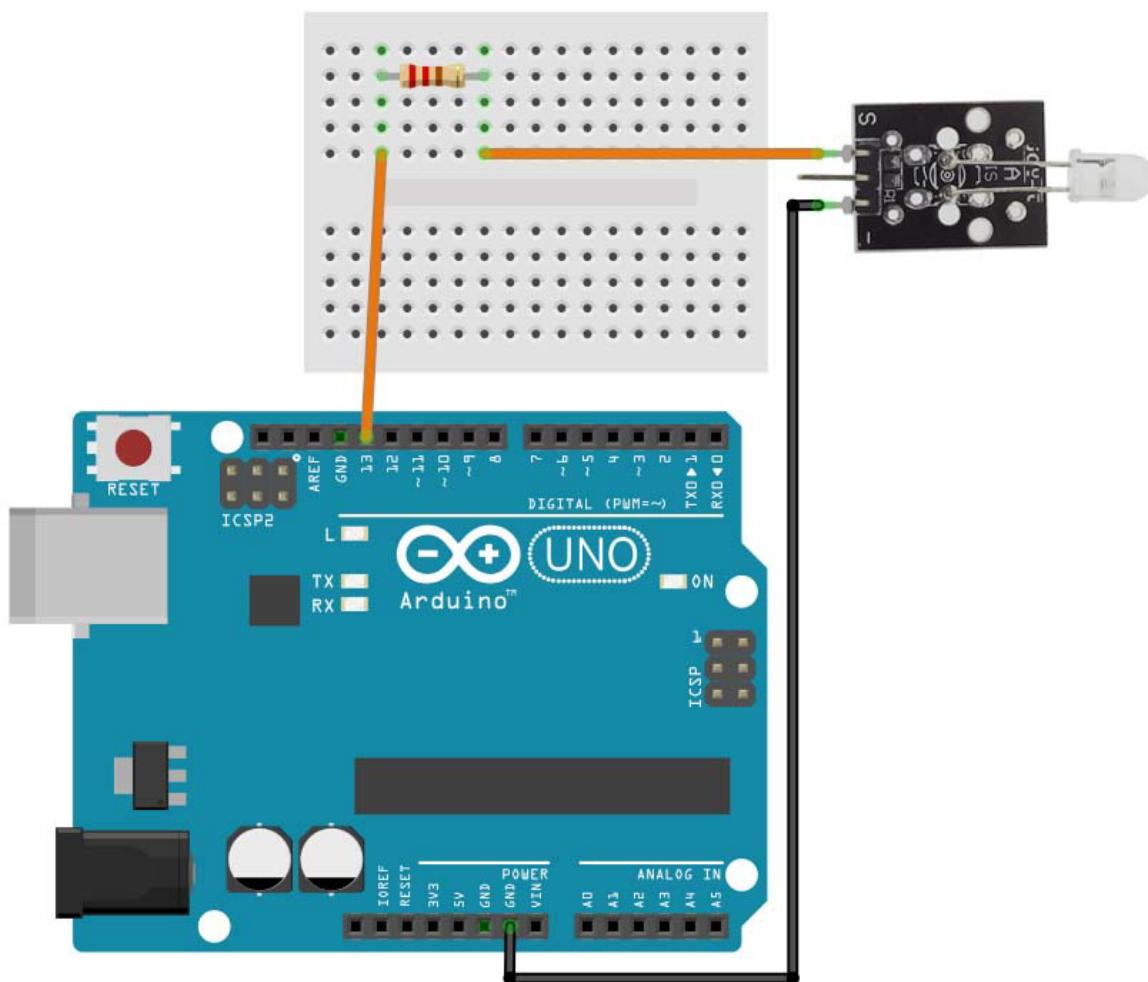
### Données techniques / Description sommaire

Diode électroluminescente émettant en infrarouge. En fonction de la tension d'entrée, des résistances en série sont nécessaires afin d'éviter un emballement thermique.

## KY-005 Module transmetteur infrarouge

**Vf= 1,1V****If= 20mA****longueur d'onde d'émission: 940nm** (*lumière invisible*)**Résistance de limitation:****Rf (5V) = 220Ω***[par exemple pour l'utilisation avec une carte basée sur un Atmel, telle qu'une Arduino]***Rf (3,3V) = 120Ω***[par exemple pour l'utilisation avec une carte Raspbverry Pi]**Exemple d'utilisation avec une carte Arduino:*

## KY-005 Module transmetteur infrarouge



## Brochage



- \*Possibilité de souder directement la résistance nécessaire sur le circuit imprimé (non conseillé: résistance CMS délicate à souder).

## KY-005 Module transmetteur infrarouge

## Exemple de code pour Arduino

### Exemple de code ON/OFF

Cet exemple de code montre comment une LED est alternativement activée pendant quatre secondes et ensuite désactivée deux secondes.

```
int Led = 13;

void setup ()
{
    pinMode (Led, OUTPUT); // Déclaration de la broche de sortie LED
}

void loop () //Boucle de programme principale
{
    digitalWrite (Led, HIGH); // la LED est activée
    delay (4000); // Temporisation de 4 secondes
    digitalWrite (Led, LOW); // la LED est déactivée
    delay (2000); // Temporisation de 2 secondes
}
```

### Exemple de programme à télécharger

[LedTestArduino\\_4On\\_2Off.zip](#)

## Exemple de code en fonction télécommande

Vous pouvez réaliser une télécommande infrarouge à l'aide des modules KY-005 et KY-022 et de deux cartes Arduino. La première carte Arduino fonctionnera en émetteur IR et la seconde en récepteur IR.

Pour l'échantillon de code suivant, une librairie supplémentaire est nécessaire:

- [Arduino-IRremote] de [Ken Shirriff](#) | publiée sous LGPL

Cette librairie est incluse et doit être copiée dans le dossier 'Libraries' avant de lancer l'IDE Arduino.

Elles peuvent être trouvées par défaut dans Windows:

C:\Users\[nom d'utilisateur]\Documents\Arduino\libraries

Pour les systèmes de transmission infrarouge, il existe différents protocoles de transmission des données. Dans l'exemple suivant, le protocole RC5 est utilisé pour la transmission des données - la bibliothèque utilisée "Arduino irRemote" effectuera la conversion selon le code RC5. Il existe également d'autres protocoles, ils sont reprise dans la bibliothèque Documentation / Code.

#### Code pour le récepteur:

## KY-005 Module transmetteur infrarouge

```
// Ajout de la librairie Arduino-IRremote
#include <IRremote.h>

// La broche d'entrée correspondante peut être déclarée pour le signal de sortie du KY-022
int RECV_PIN = 11;

// La librairie Arduino-IRremote est initialisée
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
    Serial.begin(9600);
    irrecv.enableIRIn(); // Le récepteur infrarouge démarre
}

// Boucle de programme principale
void loop() {

    // On vérifie si un signal est reçu sur le récepteur
    if (irrecv.decode(&results)) {
        //Lors de la réception d'un signal, celui-ci est décodé et envoyé dans la console
        Serial.println(results.value, HEX);
        irrecv.resume();
    }
}
```

### Code pour l'émetteur:

```
//Ajout de la librairie Arduino-IRremote...
#include <IRsend.h>

//...et initialisation de cette librairie
IRsend irsend;

// Les réglages de la sortie à prendre en charge par la librairie
// Les sorties correspondantes sont différentes selon la carte Arduino utilisée
// Arduino UNO: sortie = D3
// Arduino MEGA: sortie = D9
// Une liste complète des sorties correspondantes est visible à la page
// http://z3t0.github.io/Arduino-IRremote/
void setup()
{

}

// Boucle de programme principale
void loop() {
    // L'émetteur envoie le signal A90 (en hexadécimal) en codage 'RC5'.
    // Cette opération est répétée trois fois. Ensuite, une pause de 5 secondes est mise
    for (int i = 0; i < 3; i++) {
        irsend.sendRC5(0xA90, 12); // Signal à envoyer: [0xA90] | Signal de 12 Bit
        delay(40);
    }
    delay(5000); //Pause de 5 secondes entre les impulsions de transmission.
}
```

### **Exemple de programme à télécharger**

[Arduino\\_Fernbedienung.zip](#)

### **Affectation des broches Arduino 1 [récepteur]:**

## KY-005 Module transmetteur infrarouge

KY-022

Signal	= [Pin 11]
+V	= [Pin 5V]
GND	= [Pin GND]

### Affectation des broches Arduino 2 [émetteur]:

KY-005

Signal	= [Pin 3 (Arduino Uno)   Pin 9 (Arduino Mega)]
GND+Widerstand	= [Pin GND*]
GND	= [Pin GND]

- \*Seulement quand une résistance en série est soudée sur le module .

## Exemple de code pour Raspberry Pi

---

On présente ici deux exemples d'application de ce module.

Le premier consiste à alimenter la diode avec un signal constitué de courtes impulsions, c'est une utilisation courant pour les télécommandes infrarouges basiques.

### Exemple de code ON/OFF

---

Exemple de programmation en Python

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée. En outre la résistance de Pull-up est activé
LED_PIN = 24
GPIO.setup(LED_PIN, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [druecken Sie STRG+C, um den Test zu beenden]"

# Boucle de programme principale
try:
    while True:
        print("LED 4 secondes ON")
        GPIO.output(LED_PIN,GPIO.HIGH) #la LED est allumée
        time.sleep(4) #Temporisation de 4 secondes
        print("LED 2 secondes OFF")
        GPIO.output(LED_PIN,GPIO.LOW) #La LED est éteinte
        time.sleep(2) #Temporisation de 2 secondes

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

## KY-005 Module transmetteur infrarouge

Sensor Signal = GPIO24 [Pin 18]  
 GND+Widerstand = GND\* [Pin 9]  
 Sensor GND = Masse [Pin 6]

- \*Seulement quand une résistance en série est soudée sur le module.

### Exemple de programme à télécharger

[LedTest\\_RPi\\_4On\\_2Off.zip](#)

Commande pour lancer le programme:

```
sudo python LedTest_RPi_4On_2Off.py
```

---

## Exemple de code en fonction télécommande

La Raspberry Pi a l'avantage sur l'Arduino de pouvoir fonctionner sous Linux. A l'aide d'un récepteur infrarouge, il est possible de transmettre des signaux simples mais il est également possible d'utiliser des logiciels tel que par exemple OpenElec.

Pour créer un système de télécommande IR facilement, on utilise le logiciel Lirc (publié sous licence LGPL - [Website](#)). Nous montrons ci-dessous comment Lirc facilite l'apprentissage d'un code et comment l'envoyer par infrarouge (par exemple, à partir du Raspberry Pi, réaliser une télécommande IR pilotable par logiciel).

Pour ce faire, le module KY-005 peut être utilisé comme un émetteur infrarouge et le KY-022 comme récepteur infrarouge.

### Affectation des broches Raspberry Pi:

#### KY-005

Signal = GPIO17 [Pin 11]  
 GND+Résistance = GND\* [Pin 9]  
 GND = GND [Pin 6]

- \*Seulement quand une résistance en série est soudée sur le module .

#### KY-022

Signal = GPIO18 [Pin 12]  
 +V = 3,3V [Pin 17]  
 GND = GND [Pin 25]

---

## Installation de Lirc

Tout d'abord, il faut ouvrir un terminal sur le bureau ou se connecter via SSH avec le Raspberry Pi. On entre ensuite la commande suivante pour installer lirc sur le Raspberry Pi:

## KY-005 Module transmetteur infrarouge

```
sudo apt-get install lirc -y
```

[La carte Raspberry Pi doit être raccordée à internet]

Pour que le module lirc soit directement disponible pour le démarrage du système d'exploitation, il faut ajouter la ligne ci-dessous à la fin du fichier "/boot/config.txt"

```
dtoverlay=lirc-rpi,gpio_in_pin=18,gpio_out_pin=17,gpio_in_pull=up
```

Cette commande définit "gpio\_in\_pin = 18" à la broche d'entrée du récepteur IR et "gpio\_out\_pin = 17" à la broche de sortie de l'émetteur IR.

Le fichier peut être modifié avec la commande suivante:

```
sudo nano /boot/config.txt
```

Après l'ajout de cette ligne, le fichier sera enregistré et fermée en appuyant sur les touches [Ctrl + X -> Y -> Entrée].

Le fichier "/etc/lirc/hardware.conf" doit également être modifié à l'aide de la commande suivante:

```
sudo nano /etc/lirc/hardware.conf
```

Il faut ensuite modifier les expressions suivantes (*DRIVER="UNCONFIGURED"* devient *DRIVER="default"*, etc.):

```
DRIVER="UNCONFIGURED"
--->
DRIVER="default"
DEVICE=""
--->
DEVICE="/dev/lirc0"
MODULES=""
--->
MODULES="lirc_rpi"
```

Le fichier modifié doit ressembler à ceci:

```
# /etc/lirc/hardware.conf
#
# Arguments which will be used when launching lircd
LIRCD_ARGS=""

#Don't start lircmd even if there seems to be a good config file
#START_LIRCMD=false

#Don't start irexec, even if a good config file seems to exist.
#START_IREXEC=false

#Try to load appropriate kernel modules
LOAD_MODULES=true

# Run "lircd --driver=help" for a list of supported drivers.
```

## KY-005 Module transmetteur infrarouge

```
DRIVER="default"
# usually /dev/lirc0 is the correct setting for systems using udev
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"

# Default configuration files for your hardware if any
LIRCD_CONF=""
LIRCMD_CONF=""
```

Ensuite, il faut redémarrer la Raspberry Pi en utilisant la commande suivante:

```
sudo reboot
```

## Test du récepteur IR

Pour tester le récepteur, il faut d'abord arrêter lirc avec la commande:

```
sudo /etc/init.d/lirc stop
```

Ensuite, il faut tester ...

```
mode2 -d /dev/lirc0
```

... si la carte Raspberry peut recevoir des signaux. Pour ce faire, prenez une télécommande infrarouge et appuyez sur une touche. Vous devriez voir apparaître des lignes sous la forme suivante:

```
space 95253
pulse 9022
space 2210
pulse 604
space 95246
pulse 9019
space 2211
pulse 601
space 95252
pulse 9019
space 2210
pulse 603
space 95239
pulse 9020
space 2208
pulse 603
...
```

Avec la commande...

```
sudo /etc/init.d/lirc start
```

... le logiciel lirc peut être redémarré.

## Apprentissage du code de la télécommande

Pour enregistrer une télécommande infrarouge dans le système lirc, il faut configurer le fichier "/etc/lirc/lircd.conf". Ce fichier contient les différentes commandes infrarouges.

Afin de formater le fichier lircd.conf correctement, il faut utiliser l'assistance du logiciel lirc qui crée le fichier automatiquement. La procédure est la suivante:

Tout d'abord, il faut arrêter le service lirc

```
sudo /etc/init.d/lirc stop
```

On démarre l'assistant avec la commande suivante:

```
irrecord -d /dev/lirc0 ~/MeineFernbedienung.conf
```

Cet assistant procède à une première initialisation de la télécommande. Il faut appuyer sur les touches en alternance pour que le lirc puisse enregistrer le codage de la télécommande.

Bien suivre les instructions de l'assistant. Après l'initialisation, l'assistant vous demandera le nom du bouton, à enregistrer avec un nouveau code infrarouge. Vous pouvez le faire en sélectionnant l'affectation des touches à partir du fichier suivant:

[FernbedienungsCodes.txt](#)

Ceux-ci doivent alors être entrés dans l'assistant et validés par Entrée. On commence ensuite à enregistrer les codes infrarouges pour les touches sélectionnées.

Exemple: Entrez [KEY\_0] -> validez par Entrée -> appuyez sur "0" de la télécommande -> attente jusqu'à ce que l'enregistrement par l'assistant soit confirmé.

On peut sortir de l'assistant en poussant sur Entrée, s'il n'y a plus de touches à configurer. Le fichier de configuration est créé, mais il faut encore affecter un nom à la télécommande. Pour cela, nous ouvrons le fichier dans l'éditeur:

```
sudo nano ~/MeineFernbedienung.conf
```

On peut changer la ligne 17 de

```
name /home/pi/MeineFernbedienung.conf
```

en

```
name MeineFernbedienung
```

Il faut veiller à ne pas inclure d'espace ou utiliser des caractères spéciaux dans le nom. En utilisant la séquence de touches [Ctrl + X -> Y -> Entrée], on enregistre et on ferme le fichier.

## KY-005 Module transmetteur infrarouge

Après avoir créé le fichier de configuration, vous pouvez effectuer un backup du fichier lircd.conf à l'aide de la commande suivante:

```
sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd.conf.bak
```

et avec la commande...

```
sudo cp ~/MeineFernbedienung.conf /etc/lirc/lircd.conf
```

...le nouveau fichier de configuration est utilisé

On peut ensuite...

```
sudo /etc/init.d/lirc start
```

...redémarrer le système lirc.

A partir de maintenant, la nouvelle télécommande est enregistrée dans le système lirc et peut être utilisée avec des logiciels compatibles comme le Mediabrowser OpenElec. Vous pouvez également tester le fonctionnement de la télécommande à l'aide de la commande:

```
irw
```

Pour utiliser la télécommande avec par exemple OpenElec, il faut encore activer lirc dans les options de OpenElec.

## Envoyer des commandes avec l'émetteur infrarouge

Si vous voulez contrôler des dispositifs par infrarouge (par exemple un téléviseur) avec un Raspberry Pi, vous pouvez utiliser les commandes enregistrées précédemment. Ainsi il est possible, par exemple, d'imaginer un logiciel contrôlé par infrarouge ou des périphériques individuels connectés ou non sur le réseau Internet.

Tout d'abord, nous vérifions avec la commande suivante ...

```
irsend LIST MeineFernbedienung ""
```

...quelles commandes sont disponibles.

Nous pouvons envoyer maintenant, par exemple, la commande [KEY\_0] en utilisant la commande suivante:

```
irsend SEND_ONCE MeineFernbedienung KEY_0
```

Le téléviseur ou le récepteur devrait réagir à la commande. Pour l'appareil ci-dessus, la commande est répétitive.

## KY-005 Module transmetteur infrarouge

```
irsend SEND_START MeineFernbedienung KEY_0
```

Le code [KEY\_0] est répétitif ...

```
irsend SEND_STOP MeineFernbedienung KEY_0
```

...jusqu'à ce qui soit stoppé. Ceci est utile pour les fonctions 'volume' ou 'luminosité'.

## KY-006 Module buzzer passif

## KY-006 Module buzzer passif

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo

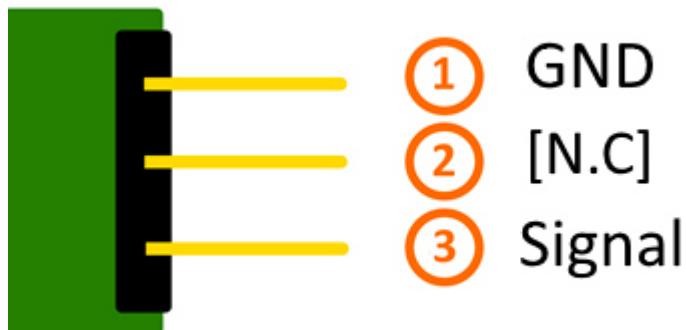


### Données techniques / Description sommaire

Des tonalités différentes peuvent être générées avec le buzzer piézo passif grâce à l'envoi de différentes fréquences via un signal PWM.

KY-006 Module buzzer passif

## Brochage



## Exemple de code pour Arduino

Voici un exemple de programme qui génère un signal d'alarme au moyen d'un signal carré envoyé au buzzer.

```
int buzzer = 8 ; // Déclaration de la broche de sortie vers le buzzer

void setup ()
{
    pinMode (buzzer, OUTPUT) ;// Initialisation comme broche de sortie
}

void loop ()
{
    unsigned char i;
    while (1)
    {
        // Dans ce programme, le buzzer est alimenté en alternance avec deux fréquences différentes.
        // Le signal a la forme d'une tension rectangulaire.
        // Le buzzer émet un son qui correspond alternativement à chaque fréquence.

        //Fréquence 1
        for (i = 0; i <80; i++)
        {
            digitalWrite (buzzer, HIGH) ;
            delay (1) ;
            digitalWrite (buzzer, LOW) ;
            delay (1) ;
        }
        //Fréquence 2
        for (i = 0; i <100; i++)
        {
            digitalWrite (buzzer, HIGH) ;
```

## KY-006 Module buzzer passif

```

        delay (2) ;
        digitalWrite (buzzer, LOW) ;
        delay (2) ;
    }
}

```

### Affectation des broches Arduino:

Sensor Signal = [Pin 8]  
 Sensor - = [Pin GND]

### Exemple de programme à télécharger

[KY-006\\_Buzzer.zip](#)

## Exemple de code pour Raspberry Pi

### Exemple de programmation en Python

L'exemple de code utilise un module PWM pour appliquer une tension carrée avec une fréquence définissable à la broche de sortie.

Le son produit au buzzer correspond approximativement à la fréquence de la tension d'onde carrée.

```

# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

# Déclaration de la broche de sortie raccordée au buzzer
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.OUT)

# Initialisation du module PWM - la fréquence de 500 Hz est prise comme valeur de départ
Frequence = 500 #In Hertz
pwm = GPIO.PWM(GPIO_PIN, Frequence)
pwm.start(50)

# Le programme attend l'entrée d'une nouvelle fréquence PWM par l'utilisateur.
# Jusque-là, le buzzer est alimenté par la tension de fréquence précédemment saisie (valeur initiale 500 Hz)
try:
    while(True):
        print "-----"
        print "Fréquence actuelle: %d" % Frequence
        Frequence = input("Entrez une nouvelle fréquence (50-5000):")
        pwm.ChangeFrequency(Fréquence)

    # remise en place de tous les GPIO en entrées
    except KeyboardInterrupt:
        GPIO.cleanup()

```

### Brochage Raspberry Pi:

Signal = GPIO24 [Pin 18]  
 +V = 3,3V [Pin 1]  
 GND = Masse [Pin 6]

## KY-006 Module buzzer passif

**Exemple de programme à télécharger**[KY-006-RPI\\_PWM.zip](#)

Commande pour lancer le programme:

```
sudo python KY-006-RPI_PWM.py
```

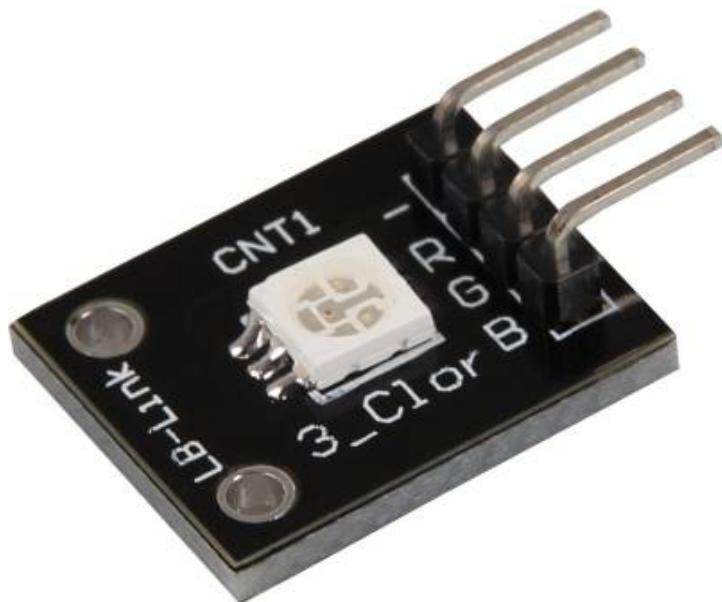
## KY-009 Module led RGB

## KY-009 Module led RGB

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	4
4 Exemple de code pour Arduino .....	4
5 Exemple de code pour Raspberry Pi .....	6

### Photo



### Données techniques / Description sommaire

Ce module est constitué d'une LED RGB qui est composée de 3 LEDS de couleurs différentes: rouge - vert - bleu. Celles-ci sont reliées par une cathode commune. En fonction de la tension d'entrée, des résistances en série sont nécessaires.

**Vf [Rouge]= 1,8V**

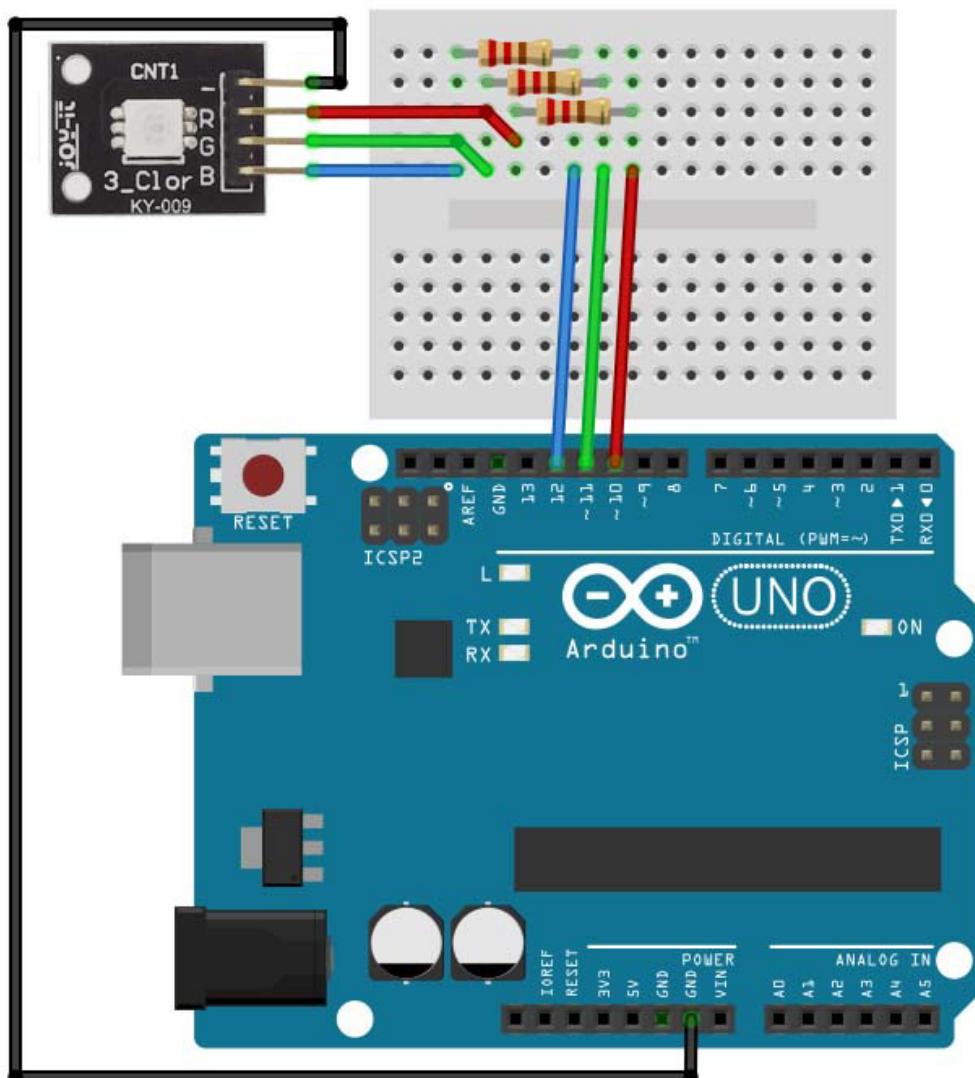
**Vf [Vert,Bleu]= 2,8V**

**If= 20mA**

## KY-009 Module led RGB

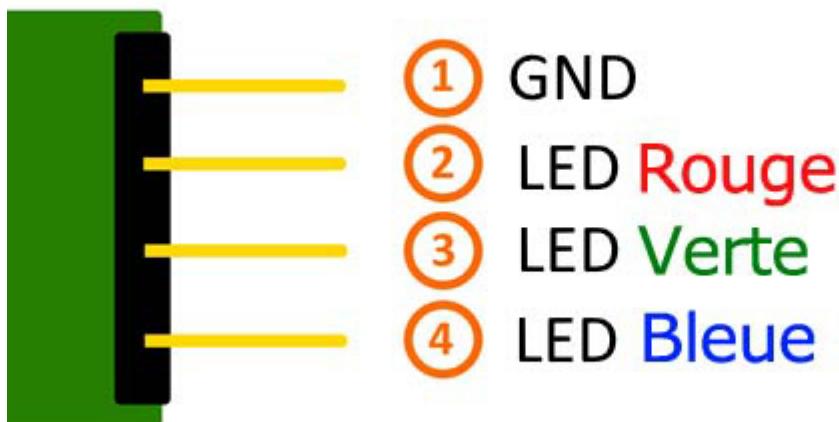
**Résistances de limitation:****Rf (3,3V) [Vert]= 100Ω****Rf (3,3V) [Rouge]= 180Ω****Rf (3,3V) [Bleu]= 100Ω***[Valeurs calculées lors de l'utilisation avec des microcontrôleurs ARM de base tel que la Raspberry Pi]***Rf (5V) [Vert] = 100Ω****Rf (5V) [Rouge] = 180Ω****Rf (5V) [Bleu] = 100Ω***[Valeurs calculées lors de l'utilisation avec des microcontrôleurs Atmel Atmega tel que Arduino]**Exemple d'utilisation avec une carte Arduino (nous avons pris 220Ω dans l'exemple pour utiliser les mêmes résistances pour tous les montages):*

## KY-009 Module led RGB



## KY-009 Module led RGB

### Brochage



### Exemple de code pour Arduino

#### Exemple de code ON/OFF

Cet exemple montre comment les LEDS intégrées sont alternativement allumées puis éteintes toutes les 3 secondes.

```

int Led_Rouge = 10;
int Led_Verte = 11;
int Led_Bleue = 12;

void setup ()
{
    // Initialisation des broches de sortie pour les LEDS
    pinMode (Led_Rouge, OUTPUT);
    pinMode (Led_Verte, OUTPUT);
    pinMode (Led_Bleue, OUTPUT);
}

void loop () //Boucle de programme principale
{
    digitalWrite (Led_Rouge, HIGH); // la Led s'allume
    digitalWrite (Led_Verte, LOW); // la LED commute
    digitalWrite (Led_Bleue, LOW); // la LED commute
    delay (3000); // Délai de 3 secondes

    digitalWrite (Led_Rouge, LOW); // la LED commute
    digitalWrite (Led_Verte, HIGH); // la LED s'allume
    digitalWrite (Led_Bleue, LOW); //la LED commute
    delay (3000); // Délai de 3 secondes supplémentaires pendant lequel les LEDS sont commutées

    digitalWrite (Led_Rouge, LOW); // la LED commute
    digitalWrite (Led_Verte, LOW); // la LED commute
    digitalWrite (Led_Bleue, HIGH); // la LED s'allume
    delay (3000); // Délai de 3 secondes supplémentaires pendant lequel les LEDS sont commutées
}

```

## KY-009 Module led RGB

### Téléchargement de l'exemple de code ON/OFF:

[KY-009\\_LED\\_ON-OFF.zip](#)

### Exemple de code en PWM

Un signal PWM (modulation de largeur d'impulsion) permet de faire varier la luminosité d'une LED. Le signal fait varier le temps pendant lequel il est à l'état haut et celui pendant lequel il est à l'état bas, ce qui fait varier la tension moyenne d'alimentation de la LED. La persistance rétinienne de l'oeil fait que nous visualisons cela comme un changement de luminosité.

Plusieurs LEDS sont intégrées dans ce module et la superposition de différents niveaux de luminosité de ces LEDS permet d'obtenir différentes couleurs. Ceci est illustré dans l'exemple de code suivant.

```
int Led_Rouge = 10;
int Led_Verte = 11;
int Led_Bleue = 12;

int val;

void setup () {
    // Initialisation des broches de sortie pour les LEDS
    pinMode (Led_Rouge, OUTPUT);
    pinMode (Led_Verte, OUTPUT);
    pinMode (Led_Bleue, OUTPUT);
}
void loop () {
    // Dans une boucle For, différentes valeurs PWM sont envoyées aux 3 LEDS
    for (val = 255; val> 0; val--)
    {
        analogWrite (Led_Bleue, val);
        analogWrite (Led_Verte, 255-val);
        analogWrite (Led_Rouge, 128-val);
        delay (1);
    }
    // Les signaux sont ensuite inversés
    for (val = 0; val <255; val++)
    {
        analogWrite (Led_Bleue, val);
        analogWrite (Led_Verte, 255-val);
        analogWrite (Led_Rouge, 128-val);
        delay (1);
    }
}
```

### Exemple de programme à télécharger

[KY-009\\_PWM.zip](#)

### Affectation des broches Arduino:

LED Rouge	= [Pin 10]
LED Verte	= [Pin 11]
LED Bleue	= [Pin 12]
Sensor GND	= [Pin GND]

KY-009 Module led RGB

## Exemple de code pour Raspberry Pi

### Exemple de code ON/OFF

Cet exemple montre comment les LEDS intégrées sont alternativement allumées puis éteintes.

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Initialisation des broches de sortie pour les LEDS
LED_ROUGE = 6
LED_VERT = 5
LED_BLEUE = 4

GPIO.setup(LED_ROUGE, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_VERT, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_BLEUE, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale
try:
    while True:
        print("LED ROUGE allumée 3 secondes")
        GPIO.output(LED_ROUGE,GPIO.HIGH) #la Led s'allume
        GPIO.output(LED_VERT,GPIO.LOW) #la LED commute
        GPIO.output(LED_BLEUE,GPIO.LOW) #la LED commute
        time.sleep(3) #Délai de 3 secondes
        print("LED Verte allumée 3 secondes")
        GPIO.output(LED_VERT,GPIO.HIGH) #la LED commute
        GPIO.output(LED_GRUEN,GPIO.LOW) #la Led s'allume
        GPIO.output(LED_BLAU,GPIO.LOW) #la LED commute
        time.sleep(3) #Délai de 3 secondes
        print("LED BLEUE allumée 3 secondes")
        GPIO.output(LED_BLAU,GPIO.HIGH) #la LED commute
        GPIO.output(LED_GRUEN,GPIO.LOW) #la LED commute
        GPIO.output(LED_BLAU,GPIO.LOW) #la Led s'allume
        time.sleep(3) #Délai de 3 secondes

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Téléchargement de l'exemple de code ON/OFF:

[KY009\\_RPi\\_ON-OFF.zip](#)

Pour débuter avec la commande:

```
sudo python KY009_RPi_ON-OFF.py
```

### Exemple de code PWM

## KY-009 Module led RGB

L'utilisation de signaux PWM (modulation de largeur d'amplitude) permet de faire varier la luminosité d'une LED. La LED est alimentée par des impulsions, le ratio entre la durée des impulsions et la durée au repos correspondant à une luminosité relative. En raison de la persistance rétinienne, l'oeil humain assimile ces changements d'alimentation de la LED à une variation de luminosité.

Vous trouverez plus d'informations sur [Plusieurs LEDS sont intégrées dans ce module et la création de couleurs différentes est produite par la superposition de différents niveaux de luminosité. Ceci est illustré dans l'exemple de code suivant.](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n>Main.ApprendrePWM</a></p>
</div>
<div data-bbox=)

```
# Les modules nécessaires sont importés et mis en place
import random, time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

# Déclaration des broches de sortie sur lesquelles les LEDS sont raccordées
LED_Rouge = 6
LED_Verte = 5
LED_Bleue = 4

# Configuration des broches en sortie
GPIO.setup(LED_Rouge, GPIO.OUT)
GPIO.setup(LED_Verte, GPIO.OUT)
GPIO.setup(LED_Bleue, GPIO.OUT)

Freq = 100 #Hz

# Les couleurs respectives sont initialisées
ROUGE = GPIO.PWM(LED_Rouge, Freq)
VERTE = GPIO.PWM(LED_Verte, Freq)
BLEUE = GPIO.PWM(LED_Bleue, Freq)
ROUGE.start(0)
VERTE.start(0)
BLEUE.start(0)

# Cette fonction génère la couleur réelle
# L'intensité de la couleur peut être modifiée grâce à la variable de couleur
# Après réglage de la couleur, la durée d'allumage est définie par 'time.sleep'

def LED_Couleur(Rouge, Verte, Bleue, pause):
    ROUGE.ChangeDutyCycle(Rouge)
    VERTE.ChangeDutyCycle(Verte)
    BLEUE.ChangeDutyCycle(Bleue)
    time.sleep(pause)

    ROUGE.ChangeDutyCycle(0)
    VERTE.ChangeDutyCycle(0)

print "LED-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale:
# Cette boucle doit faire varier l'intensité de chaque couleur de 0 à 100% en utilisant une boucle for
# Les mélanges des différentes luminosités permettent de créer un gradients de couleurs différentes.

try:
    while True:
        for x in range(0,2):
            for y in range(0,2):
                for z in range(0,2):
                    print (x,y,z)
```

## KY-009 Module led RGB

```
for i in range(0,101):
    LED_Couleur((x*i),(y*i),(z*i),.02)

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO0.cleanup()
```

**Exemple de programme à télécharger:**[KY-009\\_RPi\\_PWM.zip](#)

Commande pour lancer le programme:

```
sudo python KY-009_RPi_PWM.py
```

**Brochage Raspberry Pi:**

LED Rouge	= GPIO6 [Pin 22]
LED Verte	= GPIO5 [Pin 18]
LED Bleue	= GPIO4 [Pin 16]
Sensor GND	= Masse [Pin 6]

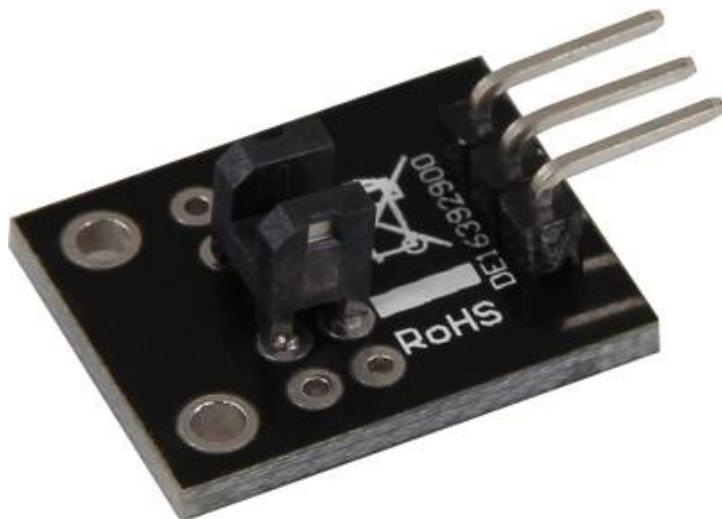
## KY-010 Capteur à fourche optique

## KY-010 Capteur à fourche optique

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo

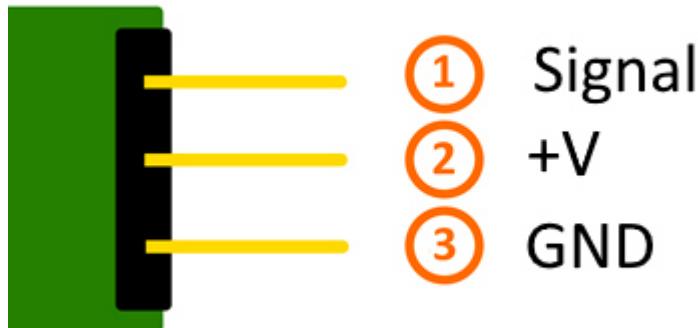


### Données techniques / Description sommaire

La connexion entre les deux broches d'entrée est interrompue si le rayon lumineuse est interrompu entre l'émetteur et le récepteur.

KY-010 Capteur à fourche optique

## Brochage



## Exemple de code pour Arduino

Voici un exemple de programme qui fait allumer une LED lorsqu'un signal est détecté au niveau du capteur. On peut utiliser les modules à Leds KY-011, KY-016 ou KY-029. Hier

```
int Led = 13 ;// Déclaration de la broche de sortie LED
int Sensor = 10; // Déclaration de la broche d'entrée du capteur
int val; // Déclaration de la broche d'entrée du capteur

void setup ()
{
    pinMode (Led, OUTPUT) ; // Initialisation de la broche de sortie
    pinMode (Sensor, INPUT) ; // Initialisation de la broche du capteur
}

void loop ()
{
    val = digitalRead (Sensor) ; // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        digitalWrite (Led, HIGH);
    }
    else
    {
        digitalWrite (Led, LOW);
    }
}
```

### Affectation des broches Arduino:

LED +	= [Pin 13]
LED -	= [Pin GND]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]
Sensor -	= [Pin GND]

KY-010 Capteur à fourche optique

### Exemple de programme à télécharger

[SensorTest\\_Arduino\\_inverted.zip](#)

## Exemple de code pour Raspberry Pi

Exemple de programmation en Python

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée. En outre la résistance de Pull-up est activée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Cette fonction de sortie est exécutée par détection du signal
def ausgabeFunktion(null):
    print("Signal détecté")

# Lors de la détection d'un signal (front montant du signal) de la fonction de sortie est déclenchée
GPIO.add_event_detect(GPIO_PIN, GPIO.RISING, callback=ausgabeFunktion, bouncetime=100)

# Boucle de programme principale
try:
    while True:
        time.sleep(1)

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger

[SensorTest\\_RPi\\_inverted.zip](#)

Commande pour lancer le programme:

```
sudo python SensorTest_RPi_inverted.py
```

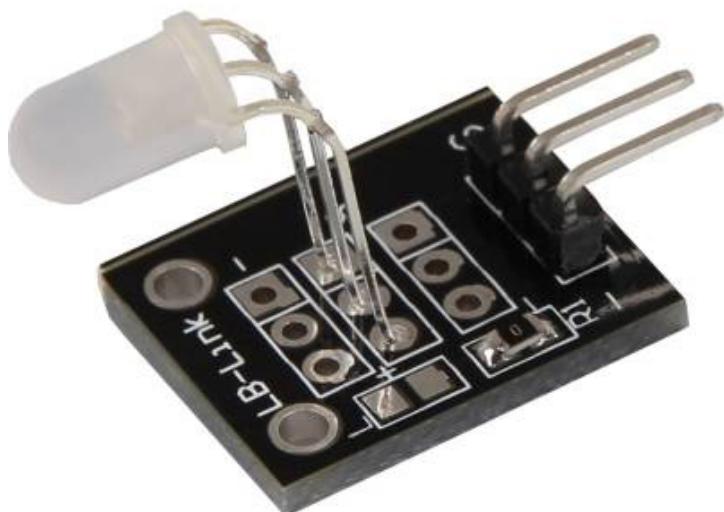
KY-011 Module led bicolore rouge/vert

## KY-011 Module led bicolore rouge/vert

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	3
4 Exemple de code pour Arduino .....	3
5 Exemple de code pour Raspberry Pi .....	4

### Photo



### Données techniques / Description sommaire

Module composé d'une LED 5mm bicolore rouge-verte, à cathode commune. Une résistance de limitation est nécessaire pour alimenter ce module.

**Vf [typ]= 2,0-2,5V**

**If= 20mA**

#### **Résistance de limitation:**

KY-011 Module led bicolore rouge/vert

**R<sub>f</sub> (3,3V) [Vert] = 120Ω**

**R<sub>f</sub> (3,3V) [Rouge] = 120Ω**

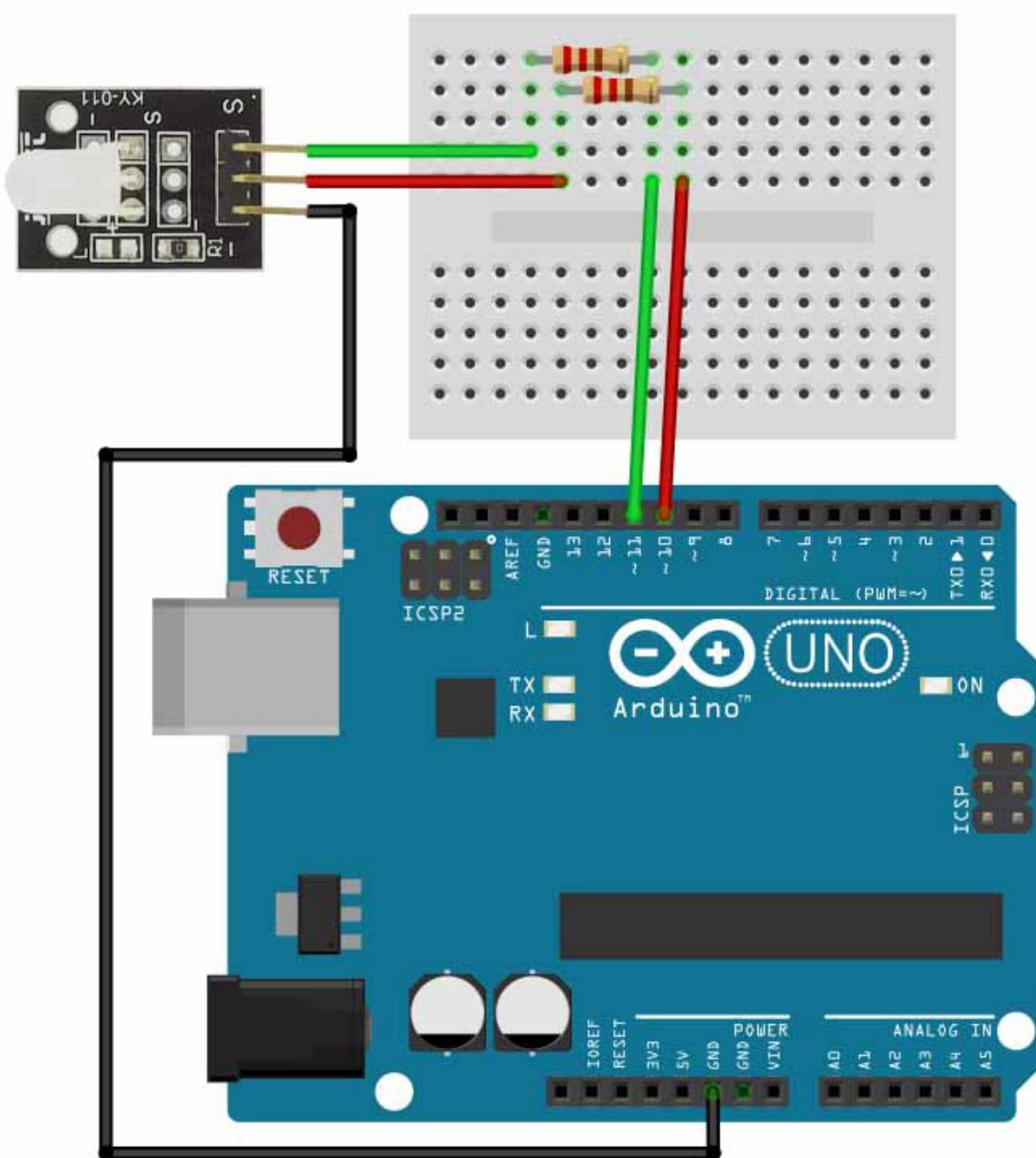
*[Valeurs calculées lors de l'utilisation avec des microcontrôleurs ARM de base tel que la Raspberry Pi]*

**R<sub>f</sub> (5V) [Vert] = 220Ω**

**R<sub>f</sub> (5V) [Rouge] = 220Ω**

*[Valeurs calculées lors de l'utilisation avec des microcontrôleurs Atmel Atmega tel que Arduino]*

**'Exemple d'utilisation avec une carte Arduino:**



KY-011 Module led bicolore rouge/vert

## Brochage



## Exemple de code pour Arduino

### Exemple de code ON/OFF

Cet exemple de code fait s'allumer alternativement la LED rouge et la LED verte toutes les 3 secondes.

```
int Led_Rouge = 10;
int Led_Verte = 11;

void setup ()
{
    // Initialisation des broches de sortie pour les LEDS
    pinMode (Led_Rouge, OUTPUT);
    pinMode (Led_Verte, OUTPUT);
}

void loop () //Boucle de programme principale
{
    digitalWrite (Led_Rouge, HIGH); // la Led s'allume
    digitalWrite (Led_Verte, LOW); // la LED commute
    delay (3000); // Délai de 3 secondes

    digitalWrite (Led_Rouge, LOW); // la LED commute
    digitalWrite (Led_Verte, HIGH); // la Led s'allume
    delay (3000); // Délai de 3 secondes supplémentaires pendant lequel les LEDS sont commutées
}
```

### Téléchargement de l'exemple de code ON/OFF:

[KY-011\\_LED\\_ON-OFF.zip](#)

### Exemple de code en PWM

## KY-011 Module led bicolore rouge/vert

Un signal PWM (modulation de largeur d'impulsion) permet de faire varier la luminosité d'une LED. Le signal fait varier le temps pendant lequel il est à l'état haut et celui pendant lequel il est à l'état bas, ce qui fait varier la tension moyenne d'alimentation de la LED. La persistance rétinienne de l'oeil fait que nous visualisons cela comme un changement de luminosité.

Plusieurs LEDS sont intégrées dans ce module et la superposition de différents niveaux de luminosité de ces LEDS permet d'obtenir différentes couleurs. Ceci est illustré dans l'exemple de code suivant.

```

int Led_Rouge = 10;
int Led_Verte = 11;

int val;

void setup () {
    // Initialisation des broches de sortie pour les LEDS
    pinMode (Led_Rouge, OUTPUT);
    pinMode (Led_Verte, OUTPUT);
}
void loop () {
    // Dans une boucle For, différentes valeurs PWM sont envoyées aux 2 LEDS
    for (val = 255; val> 0; val--)
    {
        analogWrite (Led_Verte, val);
        analogWrite (Led_Rouge, 255-val);
        delay (15);
    }
    // Dans cette seconde boucle For, les valeurs sont inversées
    for (val = 0; val <255; val++)
    {
        analogWrite (Led_Verte, val);
        analogWrite (Led_Rouge, 255-val);
        delay (15);
    }
}

```

### Exemple de programme à télécharger:

[KY-011\\_PWM.zip](#)

### Affectation des broches Arduino:

LED Verte	= [Pin 10]
LED Rouge	= [Pin 11]
Sensor GND	= [Pin GND]

## Exemple de code pour Raspberry Pi

### Exemple de code ON/OFF

Cet exemple montre comment les LEDS intégrées sont alternativement allumées puis éteintes.

```

# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

```

## KY-011 Module led bicolore rouge/vert

```
# Initialisation des broches de sortie pour les LEDS
LED_ROUGE = 5
LED_VERTE = 4
GPIO.setup(LED_ROUGE, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_VERTE, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale
try:
    while True:
        print("LED ROUGE allumée 3 secondes")
        GPIO.output(LED_ROUGE,GPIO.HIGH) #la Led s'allume
        GPIO.output(LED_VERTE,GPIO.LOW) #la LED commute
        time.sleep(3) # Délai de 3 secondes
        print("LED VERTE allumée 3 secondes")
        GPIO.output(LED_ROUGE,GPIO.LOW) #la LED commute
        GPIO.output(LED_VERTE,GPIO.HIGH) #la Led s'allume
        time.sleep(3) #Délai de 3 secondes

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Téléchargement de l'exemple de code ON/OFF:

[KY011\\_RPI\\_ON-OFF.zip](#)

Pour débuter avec la commande:

```
sudo python KY011_RPI_ON-OFF.py
```

### Exemple de code PWM

L'utilisation de signaux PWM (modulation de largeur d'amplitude) permet de faire varier la luminosité d'une LED. La LED est alimentée par des impulsions, le ratio entre la durée des impulsions et la durée au repos correspondant à une luminosité relative. En raison de la persistance rétinienne, l'oeil humain assimile ces changements d'alimentation de la LED à une variation de luminosité.

Vous trouverez plus d'informations sur [Plusieurs LEDS sont intégrées dans ce module et la création de couleurs différentes est produite par la superposition de différents niveaux de luminosité. Ceci est illustré dans l'exemple de code suivant.](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n>Main.ApprendrePWM</a></p>
</div>
<div data-bbox=)

```
# Les modules nécessaires sont importés et mis en place
import random, time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

# Déclaration des broches de sortie sur lesquelles les LEDS sont raccordées
LED_Rouge = 5
LED_Verte = 4

# Configuration des broches en sortie
GPIO.setup(LED_Rouge, GPIO.OUT)
GPIO.setup(LED_Verte, GPIO.OUT)
```

## KY-011 Module led bicolore rouge/vert

```

Freq = 100 #Hz

# Les couleurs respectives sont initialisées
ROUGE = GPIO.PWM(LED_Rouge, Freq)
VERTE = GPIO.PWM(LED_verte, Freq)
ROUGE.start(0)
VERTE.start(0)

# Cette fonction génère la couleur réelle
# L'intensité de la couleur peut être modifiée grâce à la variable Couleur
# Après réglage de la couleur, la durée d'allumage est définie par 'time.sleep'

def LED_Couleur(Rouge, Verte, pause):
    ROUGE.ChangeDutyCycle(Rouge)
    VERTE.ChangeDutyCycle(Verte)
    time.sleep(pause)

    ROUGE.ChangeDutyCycle(0)
    VERTE.ChangeDutyCycle(0)

print "LED-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale:
# Cette boucle doit faire varier l'intensité de chaque couleur de 0 à 100% en utilisant une boucle for
# Les mélanges des différentes luminosités permettent de créer un gradients de couleurs différentes.

try:
    while True:
        for x in range(0,2):
            for y in range(0,2):
                print (x,y)
                for i in range(0,101):
                    LED_Couleur((x*i),(y*i),.02)

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()

```

### Exemple de programme à télécharger:

[KY011\\_RPI\\_PWM.zip](#)

Commande pour lancer le programme:

```
sudo python KY011_RPI_PWM.py
```

### Brochage Raspberry Pi:

LED Verte = GPIO4 [Pin 16]

LED Rouge = GPIO5 [Pin 18]

Sensor GND = Masse [Pin 6]

KY-012 Module buzzer actif

## KY-012 Module buzzer actif

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

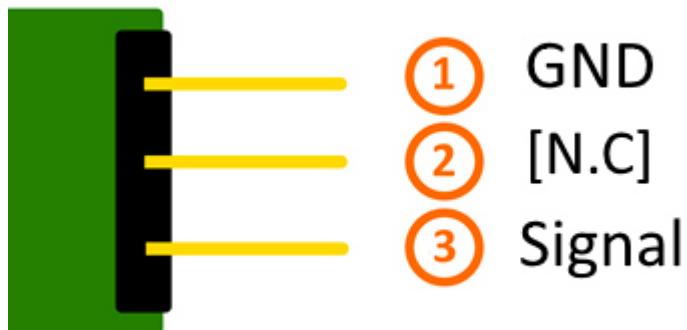
### Photo



### Données techniques / Description sommaire

Lorsqu'il est mis sous tension, ce buzzer émet un signal sonore à une fréquence de  $\pm 2,5$  kHz.

## Brochage



## Exemple de code pour Arduino

Ce module intégrant un buzzer actif ne nécessite pas d'onde carrée pour fonctionner, contrairement au module passif KY-006. Il suffit de lui appliquer une tension de minimum 3,3 Vcc pour qu'il fonctionne.

Cet exemple de code fait retentir le buzzer pendant 4 secondes, ensuite un temps de repos de 2 secondes est appliqué.

```

int Buzzer = 13;

void setup ()
{
  pinMode (Buzzer, OUTPUT); // Initialisation comme broche de sortie
}

void loop () //boucle de programme principale
{
  digitalWrite (Buzzer, HIGH); // le buzzer est actionné
  delay (4000); // délai de 4 secondes
  digitalWrite (Buzzer, LOW); // le buzzer est à l'arrêt
  delay (2000); // délai de 2 secondes
}

```

### Affectation des broches Arduino:

Sensor Signal = [Pin 13]  
 Sensor [N.C] =  
 Sensor GND = [Pin GND]

### Exemple de programme à télécharger:

[KY-0012\\_AktiverBuzzer.zip](#)

## KY-012 Module buzzer actif

### Exemple de code pour Raspberry Pi

#### Exemple de programmation en Python

Ce module intégrant un buzzer actif ne nécessite pas d'onde carrée pour fonctionner, contrairement au module passif KY-006. Il suffit de lui appliquer une tension de minimum 3,3 Vcc pour qu'il fonctionne.

Cet exemple de code fait retentir le buzzer pendant 4 secondes, ensuite un temps de repos de 2 secondes est appliqué.

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Déclaration de la broche de sortie raccordée au buzzer
Buzzer_PIN = 24
GPIO.setup(Buzzer_PIN, GPIO.OUT, initial= GPIO.LOW)

print "Buzzer-Test [Appuyez sur Ctrl + C pour terminer le test]"

#boucle de programme principale
try:
    while True:
        print("Le buzzer fonctionne 4 secondes")
        GPIO.output(Buzzer_PIN,GPIO.HIGH) #le buzzer est activé
        time.sleep(4) # délai de 4 secondes
        print("Le buzzer est au repos 2 secondes")
        GPIO.output(Buzzer_PIN,GPIO.LOW) #le buzzer est désactivé
        time.sleep(2) #délai de 2 secondes

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

#### Brochage Raspberry Pi:

Sensor Signal	=	GPIO24	[Pin 18]
Sensor [N.C]	=		
Sensor GND	=	Masse	[Pin 6]

#### Exemple de programme à télécharger

[LedTest\\_RPi\\_4On\\_2Off.zip](#)

Commande pour lancer le programme:

```
sudo python LedTest_RPi_4On_2Off.py
```

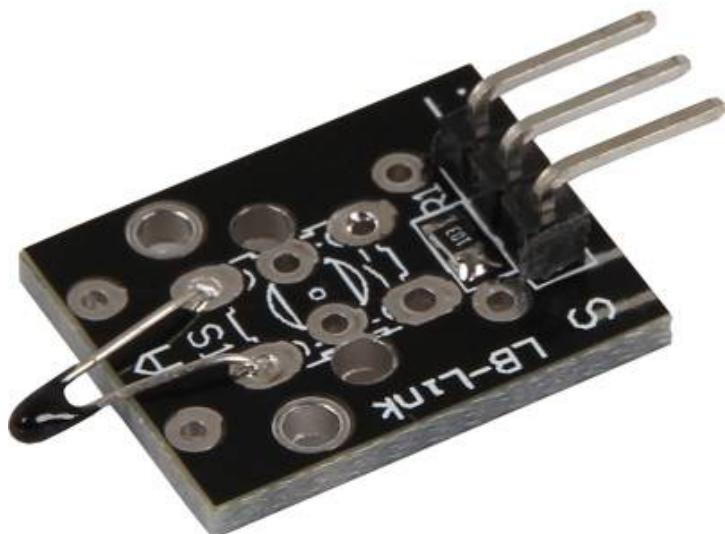
## KY-013 Capteur de température CTN

## KY-013 Capteur de température CTN

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	3
4 Exemple de code pour Arduino .....	4
5 Exemple de code pour Raspberry Pi .....	4

### Photo

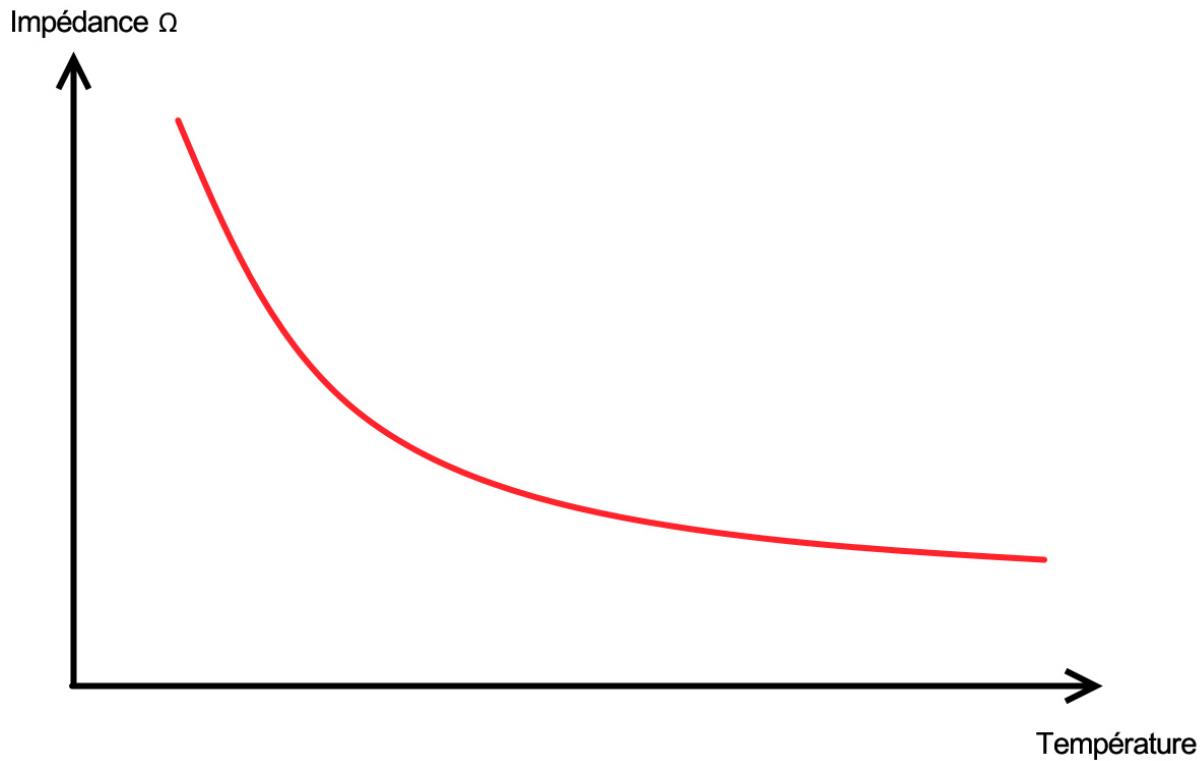


### Données techniques / Description sommaire

Plage de mesure: -55°C / +125°C

Ce module contient une thermistance CTN (coefficient de température négatif). La résistance de ce capteur diminue quand la température augmente.

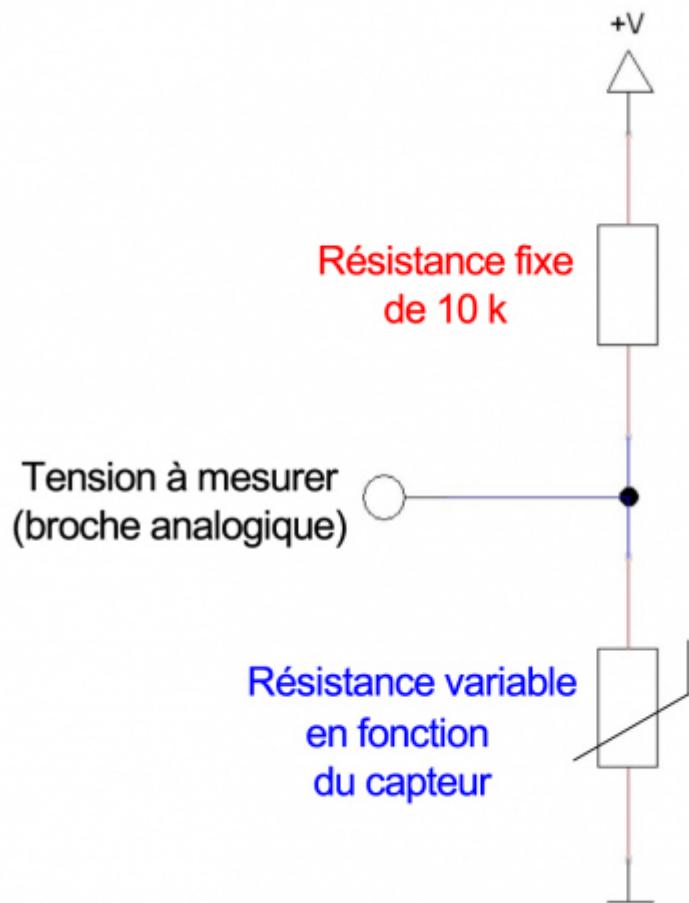
## KY-013 Capteur de température CTN



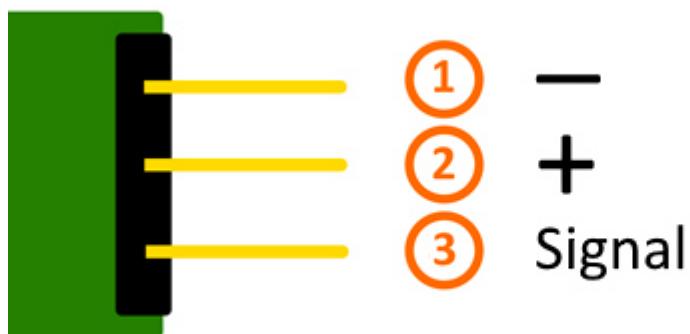
Cette variation de résistance peut être déterminée mathématiquement et permet de déterminer le coefficient de température (en fonction de la variation de la résistance aux changements de température). Vous pouvez donc connaître la température si vous connaissez la résistance du capteur.

Cette résistance peut être déterminée à l'aide d'un diviseur de tension comme dans le schéma ci-dessous. Au moyen de la tension mesurée au point milieu, on peut alors calculer la résistance de la CTN (le calcul exact est repris dans les exemples de code ci-dessous).

## KY-013 Capteur de température CTN



---

Brochage

## KY-013 Capteur de température CTN

### Exemple de code pour Arduino

Le programme mesure la valeur de tension continue aux bornes de la CTN, il calcule la température et envoie le résultat en ° C vers la sortie série.

```
#include

int sensorPin = A5; // Déclaration de la broche d'entrée

// Cette fonction convertit la valeur analogique lue en température en °C
double Thermistor(int RawADC)
{
    double Temp;
    Temp = log(10000.0 * ((1024.0 / RawADC - 1)));
    Temp = 1 / (0.001129148 + (0.000234125 + (0.000000876741 * Temp * Temp)) * Temp);
    Temp = Temp - 273.15;           // conversion de degrés Kelvin en °C
    return Temp;
}

// Sortie série à 9600 bauds
void setup()
{
    Serial.begin(9600);
}

// le programme mesure la valeur de tension continue à la CTN et convertit le résultat en ° C pour la sortie série
void loop()
{
    int readVal = analogRead(sensorPin);
    double temp = Thermistor(readVal);

    // Sortie vers l'interface série
    Serial.print("La température actuelle est:");
    Serial.print(temp);
    Serial.print(char(186)); //Impression du symbole <°>
    Serial.println("C");
    Serial.println("-----");

    delay(500);
}
```

#### Affectation des broches Arduino:

Sensor +V	= [Pin 5V]
Sensor GND	= [Pin GND]
Sensor Signal	= [Pin A5]

#### Exemple de programme à télécharger

[KY-013\\_TemperaturSensor.zip](#)

### Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

## KY-013 Capteur de température CTN

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

Vous trouverez de plus amples informations à ce sujet dans la description du [module KY-053](#).

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il calcule la résistance de la CTN, calcule la température et la transmet à la console.

```
# coding=utf-8
#!/usr/bin/python

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### KY-013 Temperatur Sensor - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os, math
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V
```

## KY-013 Capteur de température CTN

```

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8      # 8 échantillons par seconde
# sps = 16     # 16 échantillons par seconde
# sps = 32     # 32 échantillons par seconde
# sps = 64     # 64 échantillons par seconde
# sps = 128    # 128 échantillons par seconde
# sps = 250    # 250 échantillons par seconde
# sps = 475    # 475 échantillons par seconde
sps = 860    # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel = 0      # Canal 0
# adc_channel = 1      # Canal 1
# adc_channel = 2      # Canal 2
# adc_channel = 3      # Canal 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

#####
# #####
# boucle de programme principale
# #####
# Le programme mesure la tension à l'aide du convertisseur ADS1115.
# Il calcule la résistance de la CTN, calcule la température et la transmet à la console.

try:
    while True:
        #Les valeurs actuelles sont enregistrées...
        voltage = adc.readADCSingleEnded(adc_channel, gain, sps)
        # ... converties ...
        temperatur = math.log((10000/voltage)*(3300-voltage))
        temperatur = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * temperatur * temperatur)) * temperatur);
        temperatur = temperatur - 273.15;
        # ... et affichées
        print "Température:", temperatur, "°C"
        print "-----"

        # Delay
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()

```

### **Brochage Raspberry Pi:**

Capteur

+V=3,3V [Pin 1 (RPi)]  
GND=Masse [Pin 06 (RPi)]  
Signal analogique=Analog 0 [Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	= 3,3V	[Pin 17]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]

## KY-013 Capteur de température CTN

SDA = GPIO02 / SDA [Pin 03]  
A0 = s.o. [Capteur: signal analogique]

**Exemple de programme à télécharger**

[KY-013\\_RPi\\_TemperaturSensor.zip](#)

Commande pour lancer le programme:

```
sudo python KY-013_RPi_TemperaturSensor.py
```

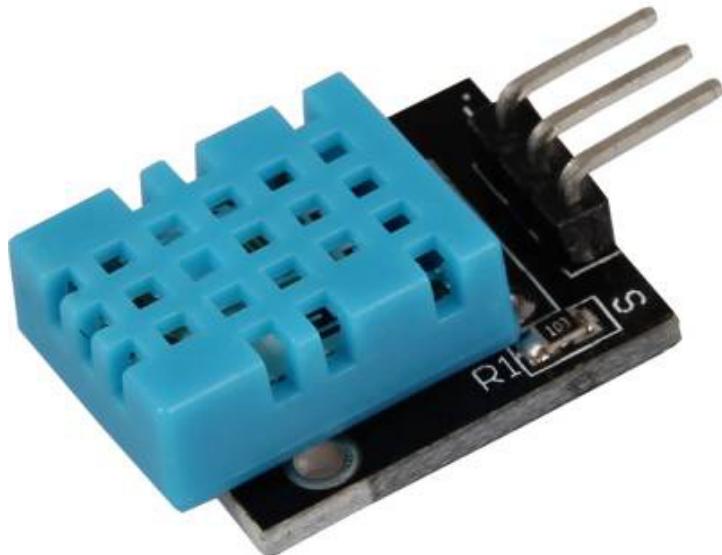
KY-015 Capteur de t° et d'humidité DHT11

## KY-015 Capteur de t° et d'humidité DHT11

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo



### Données techniques / Description sommaire

Chipset: DHT11

Protocole de communication: 1-Wire

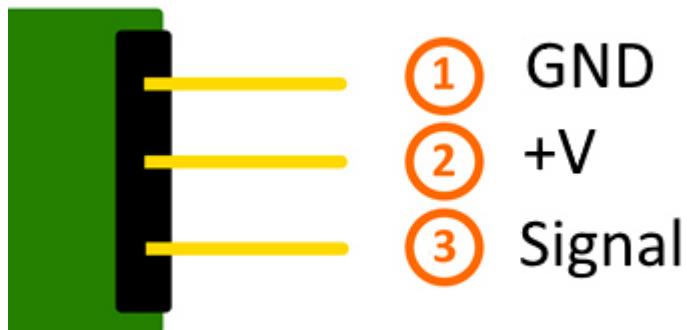
Plage de mesure humidité: 20-90% HR

Plage de mesure température: 0-50 ° C

L'avantage de ce capteur est la combinaison de la mesure de la température et de l'humidité dans un boîtier compact. Cependant, l'inconvénient est le faible taux d'échantillonnage de la mesure, de sorte qu'un nouveau résultat de mesure est disponible seulement toutes les 2 secondes. Ce capteur est donc très bon pour les mesures à long terme.

KY-015 Capteur de t° et d'humidité DHT11

## Brochage



## Exemple de code pour Arduino

La sortie de ce capteur n'est pas analogique mais numérique.

Pour contrôler ce capteur, nous utiliserons la librairie [Adafruit\\_DHT](https://github.com/adafruit/Adafruit_Python_ADS1x15/blob/master/LICENSE) publiée sous la licence open source [htt[https://github.com/adafruit/Adafruit\\_Python\\_ADS1x15/blob/master/LICENSE](https://github.com/adafruit/Adafruit_Python_ADS1x15/blob/master/LICENSE)// Licence MIT].

L'exemple ci-dessous utilise cette librairie et nous recommandons de décompresser le dossier dans le dossier des librairies Arduino, qui par défaut se trouve dans C:\Users\[nom d'utilisateur]\Documents\Arduino\libraries. La librairie est également incluse dans le pack à télécharger ci-dessous.

```
// Insertion de la librairie Adafruit_DHT
#include "DHT.h"

// déclaration de la broche d'entrée
#define DHTPIN 2

// initialisation du capteur
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
    Serial.begin(9600);
    Serial.println("Test du module KY-015 Test - Temperature et humidite:");

    // début de la mesure
    dht.begin();
}

// Boucle de programme principale
// Début du programme et lecture des mesures
// Une pause de 2 secondes est insérée entre 2 mesures
void loop() {

    // délai de 2 secondes entre 2 mesures
    delay(2000);
}
```

## KY-015 Capteur de t° et d'humidité DHT11

```
// mesure de l'humidité
float h = dht.readHumidity();
// mesure de la température
float t = dht.readTemperature();

// on vérifie si les mesures sont exécutées sans faute
// Lors de la détection d'une erreur, affichage d'un message d'erreur
if (isnan(h) || isnan(t)) {
    Serial.println("Erreur de lecture du capteur");
    return;
}

// Envoi dans la console série
Serial.println("-----");
Serial.print("Humidite: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(char(186)); //Affiche le symbole <°>
Serial.println("C ");
Serial.println("-----");
Serial.println(" ");
}
```

**Ce capteur donne des mesures toutes les 2 secondes et convient pour des mesures à long terme.**

### Exemple de programme à télécharger:

[KY-015-Kombi-Sensor\\_Temperatur\\_Feuchtigkeit.zip](#)

### Affectation des broches Arduino:

GND	= [Pin GND]
+V	= [Pin 5V]
Signal	= [Pin D2]

## Exemple de code pour Raspberry Pi

Ce programme utilise la librairie Adafruit\_Python\_DHT de la société Adafruit pour faire fonctionner le capteur. Elle est téléchargeable [[https://github.com/adafruit/Adafruit\\_Python\\_DHT](https://github.com/adafruit/Adafruit_Python_DHT)]: Librairie python] sous [[https://github.com/adafruit/Adafruit\\_Python\\_DHT/blob/master/LICENSE](https://github.com/adafruit/Adafruit_Python_DHT/blob/master/LICENSE)] licence MIT].

La librairie doit être installé au préalable en suivant la procédure ci-dessous:

Il faut d'abord installer le logiciel GitHub dans la Raspberry si ce n'est pas déjà fait:

```
sudo apt-get install git
```

La carte Raspberry Pi doit être connectée à internet. Avec la commande ci-dessous...

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

... la dernière version de la librairie Adafruit\_BM085 Library peut être téléchargée et décompressé.

On se rend en suite dans le dossier téléchargé...

## KY-015 Capteur de t° et d'humidité DHT11

```
cd Adafruit_Python_BMP/
```

... et on installe la librairie.

```
sudo python setup.py install
```

On peut alors utiliser la librairie.

Pour que la carte Raspberry Pi puisse communiquer en I2C avec le capteur, il faut activer la fonction I2C sur la Raspberry Pi. À cette fin, la ligne suivante doit être ajoutée à la fin du fichier "/boot/config.txt"

```
dtparam=i2c_arm=on
```

Le fichier peut être modifié avec la commande suivante:

```
sudo nano /boot/config.txt
```

Après avoir ajouté la ligne, le fichier sera enregistré et fermé en utilisant la séquence de touches [Ctrl + X -> Y -> Entrée].

En outre, des librairies supplémentaires sont nécessaires pour utiliser le bus I2C en Python. On utilise la commande ci-dessous pour les installer:

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

L'exemple de code en Python ci-dessous peut être utilisé.

Le programme démarre la mesure au niveau du capteur et délivre en sortie les valeurs mesurées pour la pression d'air, la température et la hauteur au-dessus du niveau de la mer.

```
#!/usr/bin/python
# coding=utf-8

# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import Adafruit_DHT
import time

# instaurer d'une pause de 2 secondes entre 2 mesures
sleepTime = 2

# le capteur doit être réglé sur Adafruit_DHT.DHT11,
# Adafruit_DHT.DHT22, or Adafruit_DHT.AM2302.
DHTSensor = Adafruit_DHT.DHT11

# déclaration de la broche de raccordement du capteur
GPIO_Pin = 23

print('Test du module KY-015 Test - Température et humidité')

try:
    while(1):
        # la mesure débute et les résultats sont contenus dans les variables correspondant
```

## KY-015 Capteur de t° et d'humidité DHT11

```

Humidite, Temperature = Adafruit_DHT.read_retry(DHTSensor, GPIO_Pin)

print("-----")
if Humidite is not None and Temperature is not None:

    # Les résultats sont envoyés dans la console
    print('Température = {0:0.1f}°C | Humidité relative = {1:0.1f}%'.format(Temperature, Humidite))

    # Étant donné que le Raspberry Pi est désavantage en raison du système d'exploitation Linux
    # pour les applications en temps réel, il est possible de la communication échoue en raison d'un
    # délai trop long. Dans ce cas, un message d'erreur s'affiche.
    else:
        print('Erreur lors de la lecture - Veuillez attendre le prochain essai!')
    print("-----")
    print("")
    time.sleep(sleepTime)

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()

```

**Ce capteur donne des mesures toutes les 2 secondes et convient pour des mesures à long terme.**

### **Brochage Raspberry Pi:**

GND	=	GND	[Pin 06]
+V	=	3,3V	[Pin 01]
Signal	=	GPIO23	[Pin 16]

### **Exemple de programme à télécharger**

[KY-015\\_RPi\\_TempHum.zip](#)

Commande pour lancer le programme:

```
sudo python KY-015_RPi_TempHum.py
```

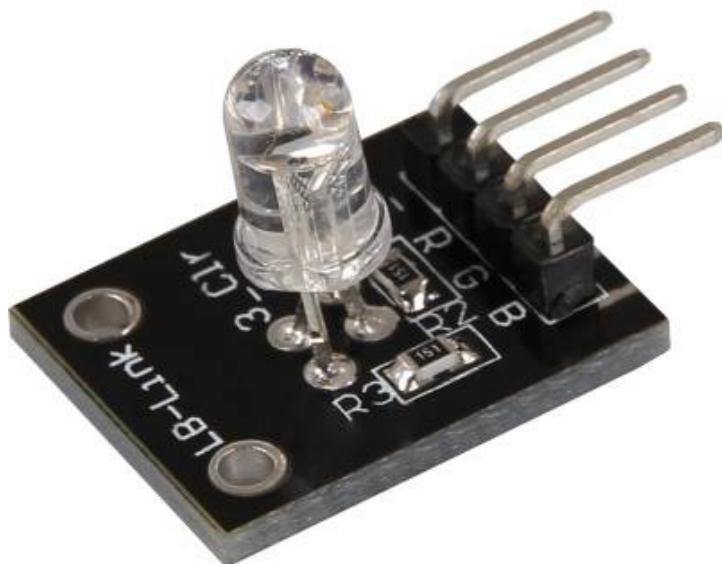
KY-016 Module led RGB 5mm

## KY-016 Module led RGB 5mm

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	4
4 Exemple de code pour Arduino .....	4
5 Exemple de code pour Raspberry Pi .....	6

### Photo



### Données techniques / Description sommaire

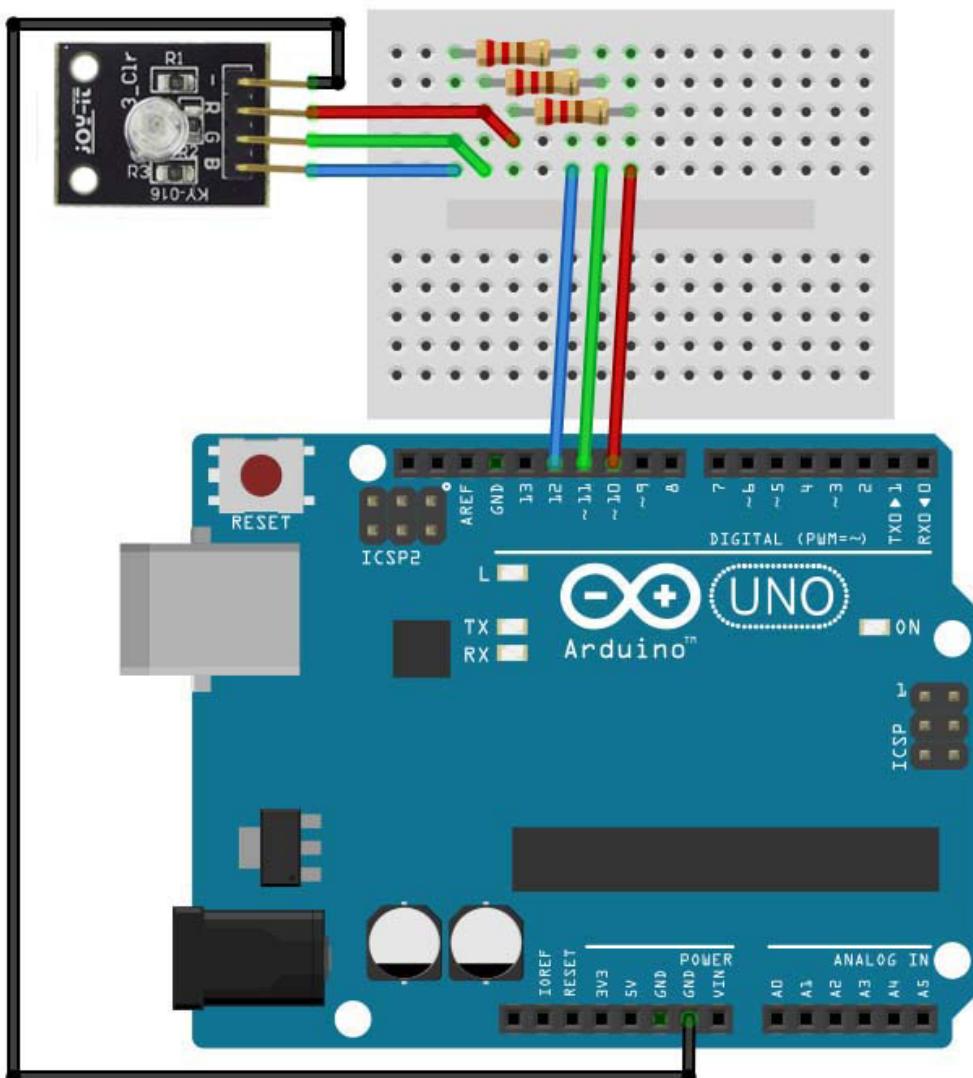
Ce module est constitué d'une LED RGB qui est composée de 3 LEDS de couleurs différentes: rouge - vert - bleu. Celles-ci sont reliées par une cathode commune. En fonction de la tension d'entrée, des résistances en série sont nécessaires.

**Vf [Rouge]= 1,8V****Vf [Verte, Bleue]= 2,8V****If= 20mA**

KY-016 Module led RGB 5mm

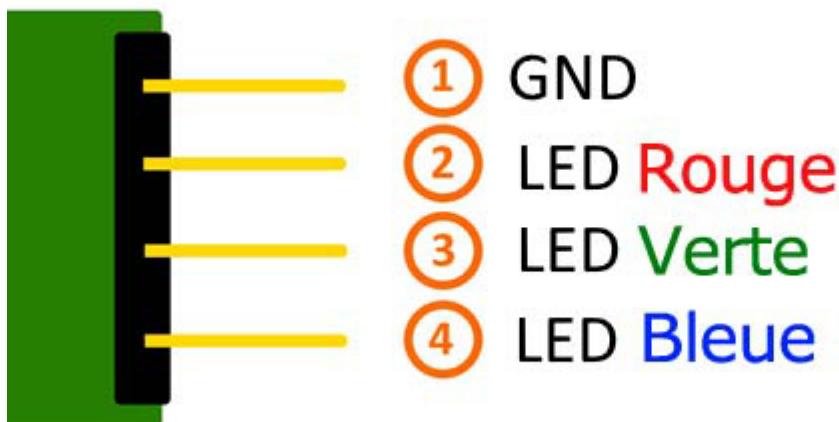
**Résistances de limitation:****Rf (3,3V) [Verte] = 100Ω****Rf (3,3V) [Rouge] = 180Ω****Rf (3,3V) [Bleue] = 100Ω***[Valeurs calculées lors de l'utilisation avec des microcontrôleurs ARM de base tel que la Raspberry Pi]***Rf (5V) [Verte] = 100Ω****Rf (5V) [Rouge] = 180Ω****Rf (5V) [Bleue] = 100Ω***[Valeurs calculées lors de l'utilisation avec des microcontrôleurs Atmel Atmega tel que Arduino]**Exemple d'utilisation avec une carte Arduino (nous avons pris 220Ω dans l'exemple pour utiliser les mêmes résistances pour tous les montages):*

KY-016 Module led RGB 5mm



KY-016 Module led RGB 5mm

## Brochage



## Exemple de code pour Arduino

### Exemple de code ON/OFF

Cet exemple montre comment les LEDS intégrées sont alternativement allumées puis éteintes toutes les 3 secondes.

```

int Led_Rouge = 10;
int Led_Verte = 11;
int Led_Bleue = 12;

void setup ()
{
    // Initialisation des broches de sortie pour les LEDS
    pinMode (Led_Rouge, OUTPUT);
    pinMode (Led_Verte, OUTPUT);
    pinMode (Led_Bleue, OUTPUT);
}

void loop () //Boucle de programme principale
{
    digitalWrite (Led_Rouge, HIGH); // la Led s'allume
    digitalWrite (Led_Verte, LOW); // la LED commute
    digitalWrite (Led_Bleue, LOW); // la LED commute
    delay (3000); // Délai de 3 secondes

    digitalWrite (Led_Rouge, LOW); // la LED commute
    digitalWrite (Led_Verte, HIGH); // la Led s'allume
    digitalWrite (Led_Bleue, LOW); // la LED commute
    delay (3000); // Délai de 3 secondes supplémentaires pendant lequel les LEDS sont commutées

    digitalWrite (Led_Rouge, LOW); // la LED commute
    digitalWrite (Led_Verte, LOW); // la LED commute
    digitalWrite (Led_Bleue, HIGH); // la Led s'allume
    delay (3000); // Délai de 3 secondes supplémentaires pendant lequel les LEDS sont commutées
}

```

KY-016 Module led RGB 5mm

#### Téléchargement de l'exemple de code ON/OFF:

[KY-016\\_LED\\_ON-OFF.zip](#)

#### Exemple de code en PWM

Un signal PWM (modulation de largeur d'impulsion) permet de faire varier la luminosité d'une LED. Le signal fait varier le temps pendant lequel il est à l'état haut et celui pendant lequel il est à l'état bas, ce qui fait varier la tension moyenne d'alimentation de la LED. La persistance rétinienne de l'oeil fait que nous visualisons cela comme un changement de luminosité.

Plusieurs LEDS sont intégrées dans ce module et la superposition de différents niveaux de luminosité de ces LEDS permet d'obtenir différentes couleurs. Ceci est illustré dans l'exemple de code suivant.

```
int Led_Rouge = 10;
int Led_Verte = 11;
int Led_Bleue = 12;

int val;

void setup () {
    // Initialisation des broches de sortie pour les LEDS
    pinMode (Led_Rouge, OUTPUT);
    pinMode (Led_Verte, OUTPUT);
    pinMode (Led_Bleue, OUTPUT);
}

void loop () {
    // Dans une boucle For, différentes valeurs PWM sont envoyées aux 3 LEDS
    for (val = 255; val> 0; val--)
    {
        analogWrite (Led_Bleue, val);
        analogWrite (Led_Verte, 255-val);
        analogWrite (Led_Rouge, 128-val);
        delay (1);
    }
    // Les signaux sont ensuite inversés
    for (val = 0; val <255; val++)
    {
        analogWrite (Led_Bleue, val);
        analogWrite (Led_Verte, 255-val);
        analogWrite (Led_Rouge, 128-val);
        delay (1);
    }
}
```

#### Exemple de programme à télécharger:

[KY-016\\_PWM.zip](#)

#### Affectation des broches Arduino:

LED Rouge	= [Pin 10]
LED Verte	= [Pin 11]
LED Bleue	= [Pin 12]
Sensor GND	= [Pin GND]

KY-016 Module led RGB 5mm

## Exemple de code pour Raspberry Pi

### Exemple de code ON/OFF

Cet exemple montre comment les LEDS intégrées sont alternativement allumées puis éteintes toutes les 3 secondes.

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Initialisation des broches de sortie pour les LEDS
LED_ROUGE = 6
LED_VERTE = 5
LED_BLEUE = 4

GPIO.setup(LED_ROUGE, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_VERTE, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_BLEUE, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale
try:
    while True:
        print("LED ROUGE allumée 3 secondes")
        GPIO.output(LED_ROUGE,GPIO.HIGH) #la Led s'allume
        GPIO.output(LED_VERTE,GPIO.LOW) #la LED commute
        GPIO.output(LED_BLEUE,GPIO.LOW) #la LED commute
        time.sleep(3) #Délai de 3 secondes
        print("LED VERTE allumée 3 secondes")
        GPIO.output(LED_ROUGE,GPIO.LOW) #la LED commute
        GPIO.output(LED_VERTE,GPIO.HIGH) #la Led s'allume
        GPIO.output(LED_BLEUE,GPIO.LOW) #la LED commute
        time.sleep(3) #Délai de 3 secondes
        print("LED BLEUE allumée 3 secondes")
        GPIO.output(LED_ROUGE,GPIO.LOW) #la LED commute
        GPIO.output(LED_VERTE,GPIO.LOW) #la LED commute
        GPIO.output(LED_BLEUE,GPIO.HIGH) #la Led s'allume
        time.sleep(3) #Délai de 3 secondes

    #remise en place de tous les GPIO en entrées
    except KeyboardInterrupt:
        GPIO.cleanup()
```

### Téléchargement de l'exemple de code ON/OFF:

[KY016\\_RPi\\_ON-OFF.zip](#)

Pour débuter avec la commande:

```
sudo python KY016_RPi_ON-OFF.py
```

### Exemple de code PWM

## KY-016 Module led RGB 5mm

L'utilisation de signaux PWM (modulation de largeur d'amplitude) permet de faire varier la luminosité d'une LED. La LED est alimentée par des impulsions, le ratio entre la durée des impulsions et la durée au repos correspondant à une luminosité relative. En raison de la persistance rétinienne, l'oeil humain assimile ces changements d'alimentation de la LED à une variation de luminosité.

Vous trouverez plus d'informations sur [Plusieurs LEDS sont intégrées dans ce module et la création de couleurs différentes est produite par la superposition de différents niveaux de luminosité. Ceci est illustré dans l'exemple de code suivant.](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n>Main.ApprendrePWM</a></p>
</div>
<div data-bbox=)

```
# Les modules nécessaires sont importés et mis en place
import random, time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

# Déclaration des broches de sortie sur lesquelles les LEDS sont raccordées
LED_Rouge = 6
LED_Verte = 5
LED_Bleue = 4

# Configuration des broches en sortie
GPIO.setup(LED_Rouge, GPIO.OUT)
GPIO.setup(LED_Verte, GPIO.OUT)
GPIO.setup(LED_Bleue, GPIO.OUT)

Freq = 100 #Hz

# Les couleurs respectives sont initialisées
ROUGE = GPIO.PWM(LED_Rouge, Freq)
VERTE = GPIO.PWM(LED_Verte, Freq)
BLEUE = GPIO.PWM(LED_Bleue, Freq)
ROUGE.start(0)
VERTE.start(0)
BLEUE.start(0)

# Cette fonction génère la couleur réelle
# L'intensité de la couleur peut être modifiée grâce à la variable de couleur
# Après réglage de la couleur, la durée d'allumage est définie par 'time.sleep'

def LED_Couleur(Rouge, Verte, Bleue, pause):
    ROUGE.ChangeDutyCycle(Rouge)
    VERTE.ChangeDutyCycle(Verte)
    BLEUE.ChangeDutyCycle(Bleue)
    time.sleep(pause)

    ROUGE.ChangeDutyCycle(0)
    VERTE.ChangeDutyCycle(0)

print "LED-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale:
# Cette boucle doit faire varier l'intensité de chaque couleur de 0 à 100% en utilisant une boucle for
# Les mélanges des différentes luminosités permettent de créer un gradients de couleurs différentes. try:
    while True:
        for x in range(0,2):
            for y in range(0,2):
                for z in range(0,2):
                    print (x,y,z)
```

## KY-016 Module led RGB 5mm

```
for i in range(0,101):
    LED_Couleur((x*i),(y*i),(z*i),.02)

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO0.cleanup()
```

**Exemple de programme à télécharger:**[KY-016\\_RPi\\_PWM.zip](#)

Commande pour lancer le programme:

```
sudo python KY-016_RPi_PWM.py
```

**Brochage Raspberry Pi:**

LED Rouge	= GPIO6 [Pin 22]
LED Verte	= GPIO5 [Pin 18]
LED Bleue	= GPIO4 [Pin 16]
Sensor GND	= Masse [Pin 6]

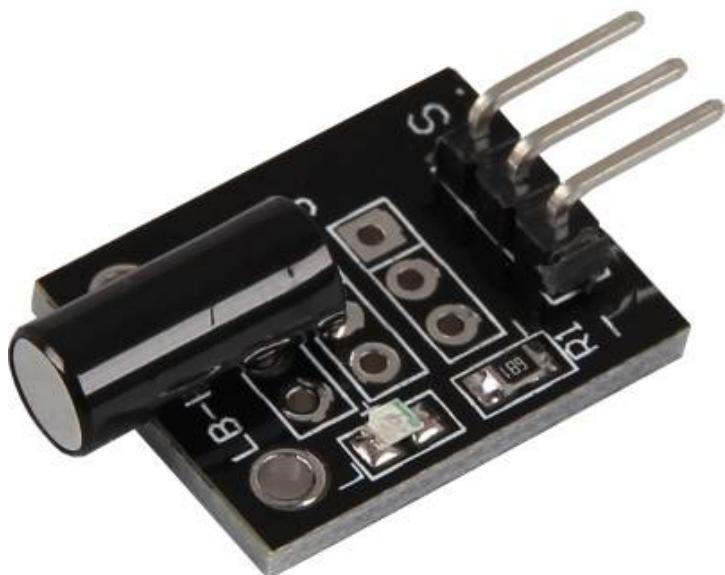
KY-017 Capteur d'inclinaison

## KY-017 Capteur d'inclinaison

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo

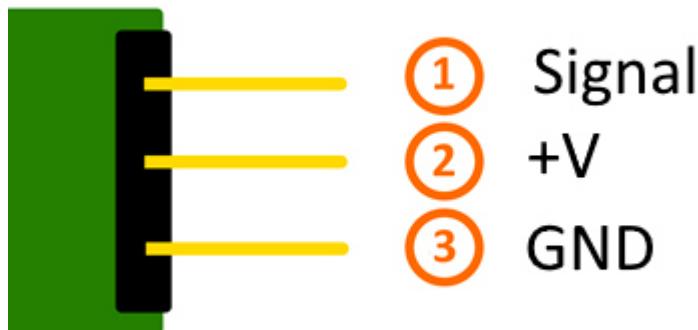


### Données techniques / Description sommaire

Un contact se ferme en fonction de l'inclinaison du module.

KY-017 Capteur d'inclinaison

## Brochage



## Exemple de code pour Arduino

L'exemple de programme ci-dessous fait allumer une LED lorsqu'un signal est détecté au niveau du capteur. Vous pouvez utiliser les modules à LEDS KY-011, KY-016 ou KY-029.

```
int Led = 13; // déclaration de la broche de sortie pour la LED
int Sensor = 10; // déclaration de la broche d'entrée du capteur
int val; // variable temporaire

void setup ()
{
    pinMode (Led, OUTPUT); // Initialisation de la broche de sortie
    pinMode (Sensor, INPUT); // Initialisation de la broche du capteur
}

void loop ()
{
    val = digitalRead (Sensor); // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        digitalWrite (Led, LOW);
    }
    else
    {
        digitalWrite (Led, HIGH);
    }
}
```

### Affectation des broches Arduino:

LED +	= [Pin 13]
LED -	= [Pin GND]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]
Sensor -	= [Pin GND]

KY-017 Capteur d'inclinaison

### Exemple de programme à télécharger

[SensorTest\\_Arduino\\_withoutPullUP.zip](#)

## Exemple de code pour Raspberry Pi

Exemple de programmation en Python

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée. En outre la résistance de Pull-up est activée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Cette fonction de sortie est exécutée par détection du signal
def ausgabeFunktion(null):
    print("Signal détecté")

# Lors de la détection d'un signal (front descendant du signal) de la fonction de sortie est déclenchée
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=100)

# Boucle de programme principale
try:
    while True:
        time.sleep(1)

# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger:

[SensorTest\\_RPi\\_withoutPullUP.zip](#)

Commande pour lancer le programme:

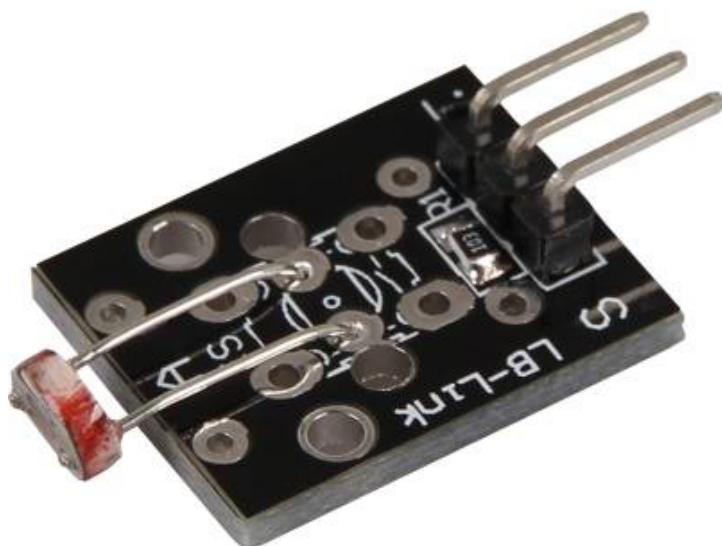
```
sudo python SensorTest_RPi_withoutPullUP.py
```

## KY-018 Module à photorésistance LDR

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	3
5 Exemple de code pour Raspberry Pi .....	3

### Photo



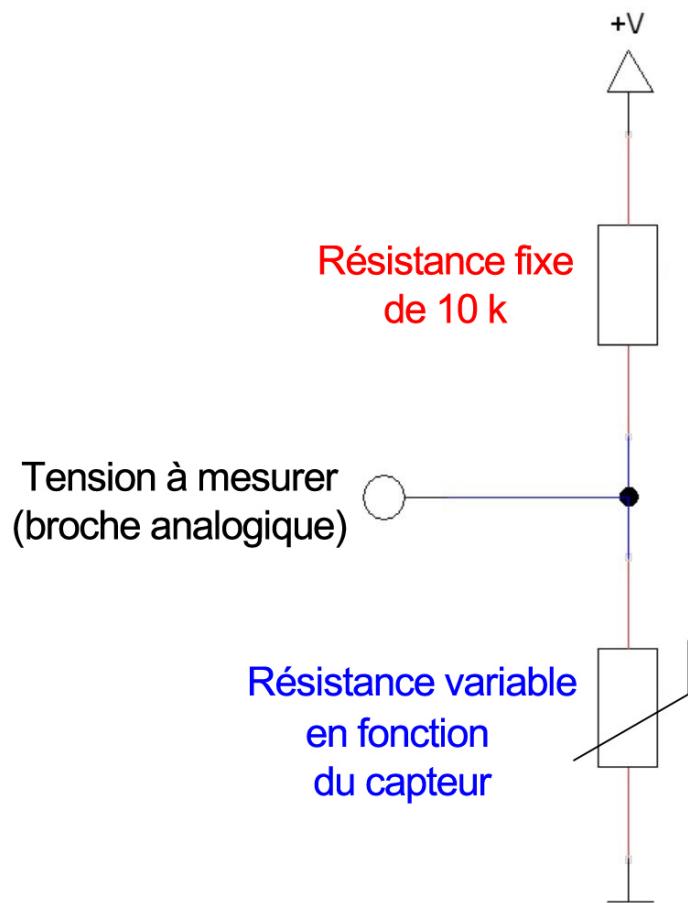
### Données techniques / Description sommaire

Ce module intègre une photorésistance dont la valeur de résistance diminue lorsque la lumière augmente.

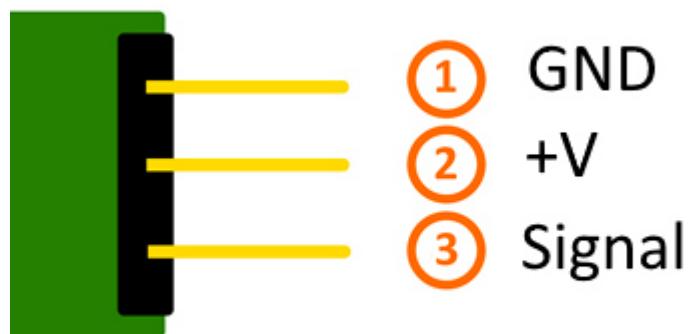
Cette variation de résistance peut être déterminée mathématiquement et permet de déterminer le coefficient de température (en fonction de la variation de la résistance aux changements de température). Vous pouvez donc connaître la température si vous connaissez la résistance du capteur.

Cette résistance peut être déterminée à l'aide d'un diviseur de tension comme dans le schéma ci-dessous. Au moyen de la tension mesurée au point milieu, on peut alors calculer la résistance de la LDR (le calcul exact est repris dans les exemples de code ci-dessous).

### KY-018 Module à photorésistance LDR



### Brochage



## KY-018 Module à photorésistance LDR

### Exemple de code pour Arduino

Le programme mesure la valeur de tension continue aux bornes de la LDR, il calcule la résistance du capteur et envoie le résultat vers la sortie série.

```

int sensorPin = A5; // Déclaration de la broche d'entrée

// Sortie série à 9600 bauds
void setup()
{
    Serial.begin(9600);
}

// Le programme mesure la valeur de tension continue aux bornes de la LDR, il calcule la
// résistance du capteur et envoie le résultat vers la sortie série.
void loop()
{
    // La valeur de la tension est mesurée
    int rawValue = analogRead(sensorPin);
    float voltage = rawValue * (5.0/1023) * 1000;

    float resistance = 10000 * ( voltage / ( 5000.0 - voltage) );

    // Sortie vers l'interface série
    Serial.print("Tension:");    Serial.print(voltage); Serial.print("mV");
    Serial.print(", Resistance:"); Serial.print(resistance); Serial.println("Ohm");
    Serial.println("-----");

    delay(500);
}

```

#### Affectation des broches Arduino:

Sensor GND	= [Pin GND]
Sensor +V	= [Pin 5V]
Sensor Signal	= [Pin A5]

#### Exemple de programme à télécharger:

[Single\\_Analog\\_Sensor.zip](#)

### Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

Vous trouverez de plus amples informations à ce sujet dans la description du [module KY-053](#).

## KY-018 Module à photorésistance LDR

**!! Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous licence BSD.

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il calcule la résistance de la LDR et la transmet à la console.

```
# coding=utf-8
#!/usr/bin/python

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Single Analog Sensor - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os, math
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2
voltageMax = 3300 # valeur de tension maximale possible à l'entrée du convertisseur ADC

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde
# sps = 32 # 32 échantillons par seconde
# sps = 64 # 64 échantillons par seconde
# sps = 128 # 128 échantillons par seconde
sps = 250 # 250 échantillons par seconde
# sps = 475 # 475 échantillons par seconde
# sps = 860 # 860 échantillons par seconde
```

## KY-018 Module à photorésistance LDR

```
# choix du canal ADC (1-4)
adc_channel = 0      # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

#####
##### boucle de programme principale
#####
# Le programme mesure la tension à l'aide du convertisseur ADS1115.
# Il calcule la résistance de la LDR et les transmet à la console.

try:
    while True:
        #La valeur actuelle est reçue, ...
        voltage = adc.readADCSingleEnded(adc_channel, gain, sps)

        # ... la résistance est calculée...
        resistance = 10000 * voltage/(voltageMax - voltage)

        # ... et les deux sont envoyées à la console
        print "Tension:", voltage,"mV, Résistance:", resistance,"Ω"
        print "-----"

        # Delay
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

#### Capteur

GND	= GND	[Pin 06 (RPi)]
+V	= 3,3V	[Pin 01 (RPi)]
Signal analogique	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

#### ADS1115 - KY-053:

VDD	= 3,3V	[Pin 17]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]
A0	= s.o.	[Capteur: signal analogique]

### Exemple de programme à télécharger

[RPi\\_Single\\_Analog\\_Sensor.zip](#)

Commande pour lancer le programme:

KY-018 Module à photorésistance LDR

```
sudo python RPi_Single_Analog_Sensor.py
```

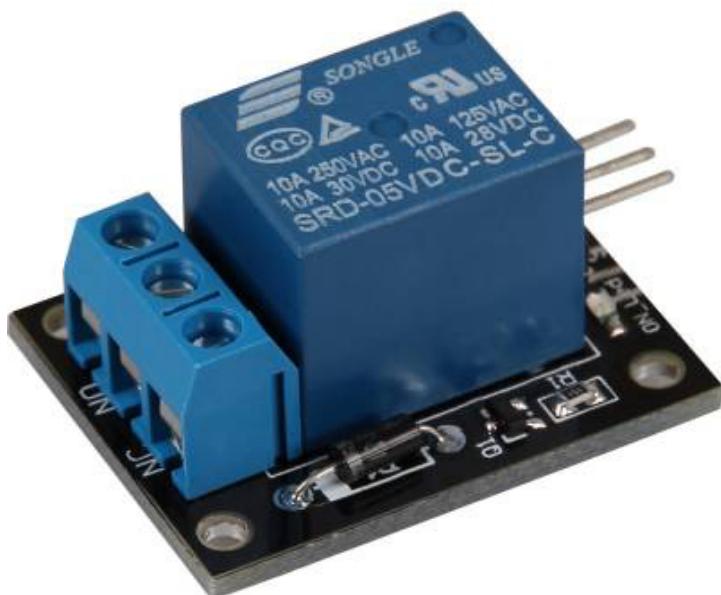
## KY-019 Module Relais 5V

## KY-019 Module Relais 5V

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo



### Données techniques / Description sommaire

Pouvoir de coupure: 10 A/ 240 Vac ou 28 Vcc / 10A.

Ce module à relais permet de commuter des tensions plus élevées à partir d'une tension de commande de 5 Vcc.

#### !!!! Attention !!!!

**Travailler avec des tensions supérieures à 30 Volts et spécialement sur le 230 Vac peut entraîner des blessures sérieuses voire mortelles.**

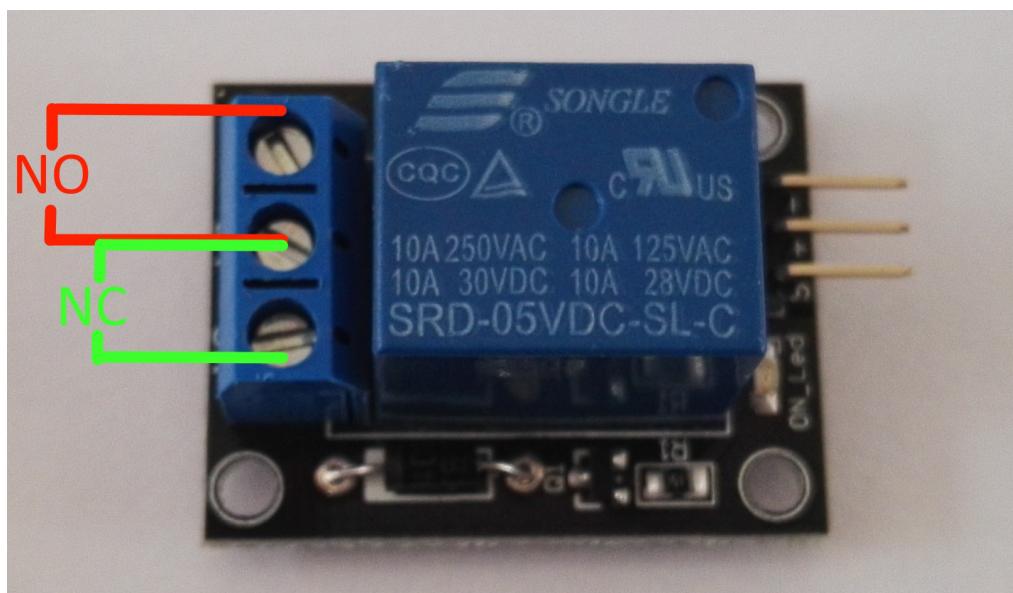
**Nous déconseillons fortement de travailler à de telles tensions sauf si vous êtes un professionnel et que vous prenez les précautions appropriées.**

## KY-019 Module Relais 5V

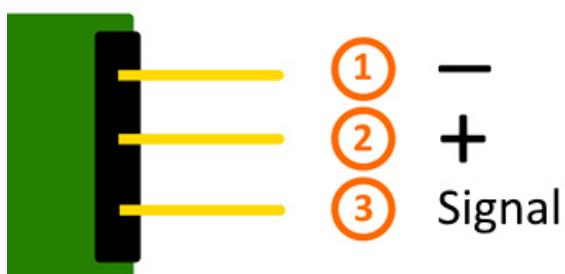
### !!!! Attention !!!!

La sortie du relais dispose de trois bornes:

- la borne NC (pour "normalement fermé") qui laisse passer le courant au repos.
- la borne NO(pour "normalement ouvert") dont le contact par défaut est ouvert ou déconnecté.
- la borne COM qui est commune à NC et NO



## Brochage



## Exemple de code pour Arduino

Le programme simule un clignotant. Il fait commuter le relais d'une position à l'autre avec un délai paramétré.

## KY-019 Module Relais 5V

```

int relay = 10; // Déclaration de la broche de raccordement du relais
int delayTime = 1; // Délai entre 2 commutations

void setup ()
{
    pinMode (relay, OUTPUT); // La broche est déclarée en tant que sortie
}

// Le programme simule un clignotant. Il fait commuter le relais d'une position à
// l'autre avec un délai paramétré (delayTime).
void loop ()
{
    digitalWrite (relay, HIGH); // "NO" est passant;
    delay (delayTime * 1000);
    digitalWrite (relay, LOW); // "NC" est passant;
    delay (delayTime * 1000);
}

```

### Affectation des broches Arduino:

Sensor -	= [Pin GND]
Sensor +	= [Pin 5V]
Sensor Signal	= [Pin 10]

### Exemple de programme à télécharger

[KY-019\\_Relaix.zip](#)

## Exemple de code pour Raspberry Pi

Le programme simule un clignotant. Il fait commuter le relais d'une position à l'autre avec un délai paramétré (delayTime).

```

# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# la pause entre 2 commutation est paramétrée
delayTime = 1

# La broche de raccordement du relais est déclarée. En outre la résistance de Pull-up est activée.
RELAIS_PIN = 21
GPIO.setup(RELAIS_PIN, GPIO.OUT)
GPIO.output(RELAIS_PIN, False)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale
try:
    while True:
        GPIO.output(RELAIS_PIN, True) # NO est passant
        time.sleep(delayTime)
        GPIO.output(RELAIS_PIN, False) # NC est passant
        time.sleep(delayTime)

```

## KY-019 Module Relais 5V

```
# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

**Brochage Raspberry Pi:**

Relais -	=	GND	[Pin 06]
Relais +	=	5V	[Pin 2]
Relais Signal	=	GPIO24	[Pin 18]

**Exemple de programme à télécharger**[KY-019\\_RPi\\_Relais.zip](#)

Commande pour lancer le programme:

```
sudo python KY-019_RPi_Relais.py
```

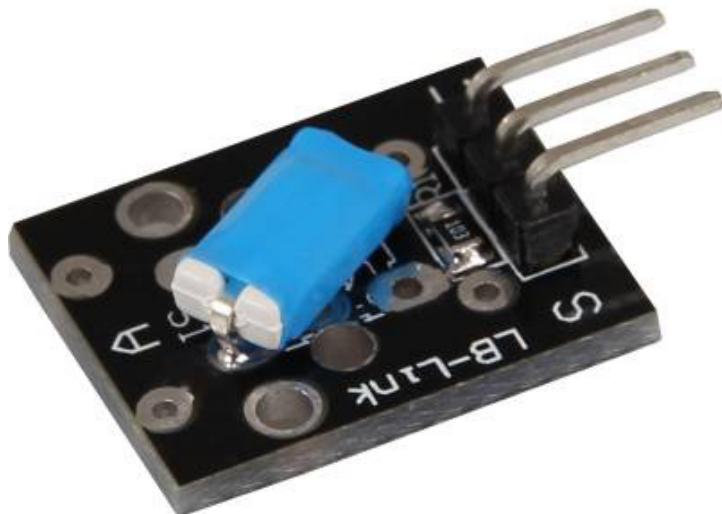
KY-020 Capteur d'inclinaison

## KY-020 Capteur d'inclinaison

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo

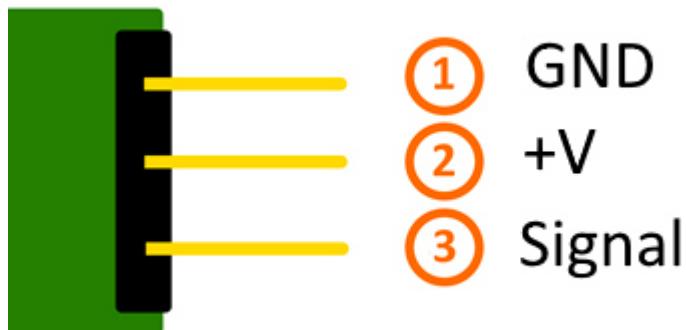


### Données techniques / Description sommaire

En fonction de l'inclinaison, un contact se ferme.

KY-020 Capteur d'inclinaison

## Brochage



## Exemple de code pour Arduino

Voici un exemple de programme qui fait allumer une LED lorsqu'un signal est détecté au niveau du capteur. On peut utiliser les modules à Leds KY-011, KY-016 ou KY-029.

```
int Led = 13; // Déclaration de la broche de sortie LED
int Sensor = 10; // Déclaration de la broche d'entrée du capteur
int val; // Variable temporaire

void setup ()
{
    pinMode (Led, OUTPUT); // Initialisation de la broche de sortie
    pinMode (Sensor, INPUT); // Initialisation de la broche du capteur
}

void loop ()
{
    val = digitalRead (Sensor); // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        digitalWrite (Led, LOW);
    }
    else
    {
        digitalWrite (Led, HIGH);
    }
}
```

### Affectation des broches Arduino:

LED +	= [Pin 13]
LED -	= [Pin GND]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]
Sensor -	= [Pin GND]

KY-020 Capteur d'inclinaison

### Exemple de programme à télécharger

[SensorTest\\_Arduino\\_withoutPullUP.zip](#)

## Exemple de code pour Raspberry Pi

Exemple de programmation en Python

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée. En outre la résistance de Pull-up est activée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Cette fonction de sortie est exécutée par détection du signal
def ausgabeFunktion(null):
    print("Signal détecté")

# Lors de la détection d'un signal (front descendant du signal) de la fonction de sortie est déclenchée
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=100)

# Boucle de programme principale
try:
    while True:
        time.sleep(1)

# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger

[SensorTest\\_RPi\\_withoutPullUP.zip](#)

Commande pour lancer le programme:

```
sudo python SensorTest_RPi_withoutPullUP.py
```

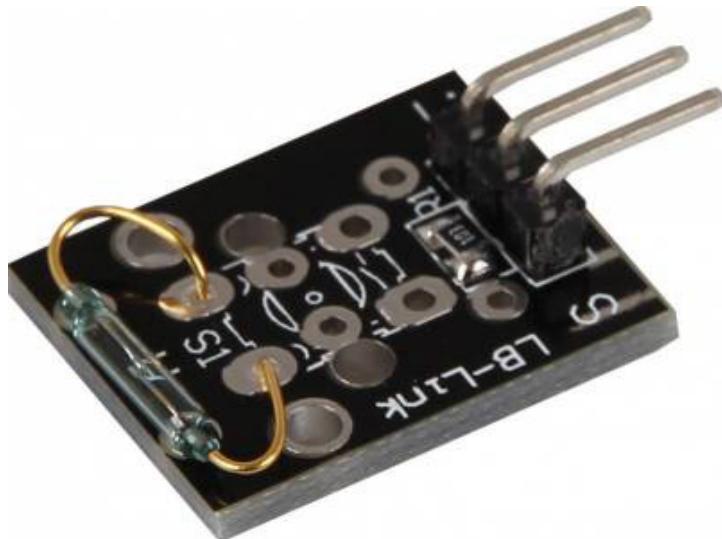
KY-021 Capteur magnétique Reed

## KY-021 Capteur magnétique Reed

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo

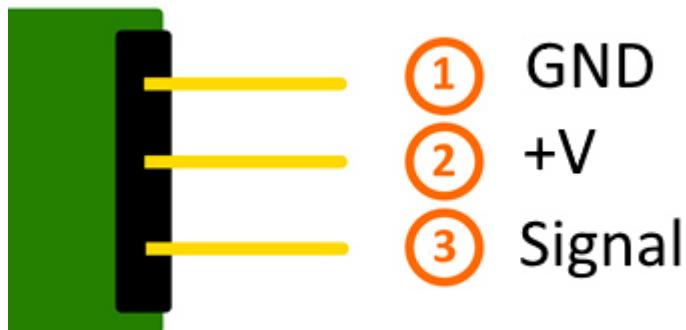


### Données techniques / Description sommaire

Si un champ magnétique est détecté (passage d'un aimant par exemple), la sortie change d'état.

KY-021 Capteur magnétique Reed

## Brochage



## Exemple de code pour Arduino

Voici un exemple de programme qui fait allumer une LED lorsqu'un signal est détecté au niveau du capteur. On peut utiliser les modules à Leds KY-011, KY-016 ou KY-029.

```
int Led = 13; // Déclaration de la broche de sortie LED
int Sensor = 10; // Déclaration de la broche d'entrée du capteur
int val; // Variable temporaire

void setup ()
{
    pinMode (Led, OUTPUT); // Initialisation de la broche de sortie
    pinMode (Sensor, INPUT); // Initialisation de la broche du capteur
}

void loop ()
{
    val = digitalRead (Sensor); // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        digitalWrite (Led, LOW);
    }
    else
    {
        digitalWrite (Led, HIGH);
    }
}
```

### Affectation des broches Arduino:

LED +	= [Pin 13]
LED -	= [Pin GND]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]
Sensor -	= [Pin GND]

KY-021 Capteur magnétique Reed

### Exemple de programme à télécharger

[SensorTest\\_Arduino\\_withoutPullUP.zip](#)

## Exemple de code pour Raspberry Pi

Exemple de programmation en Python

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée. En outre la résistance de Pull-up est activée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Cette fonction de sortie est exécutée par détection du signal
def ausgabeFunktion(null):
    print("Signal détecté")

# Lors de la détection d'un signal (front descendant du signal) de la fonction de sortie est déclenchée
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=100)

# Boucle de programme principale
try:
    while True:
        time.sleep(1)

# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### réinitialisation de tous les GPIO en entrées:

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger

[SensorTest\\_RPi\\_withoutPullUP.zip](#)

Commande pour lancer le programme:

```
sudo python SensorTest_RPi_withoutPullUP.py
```

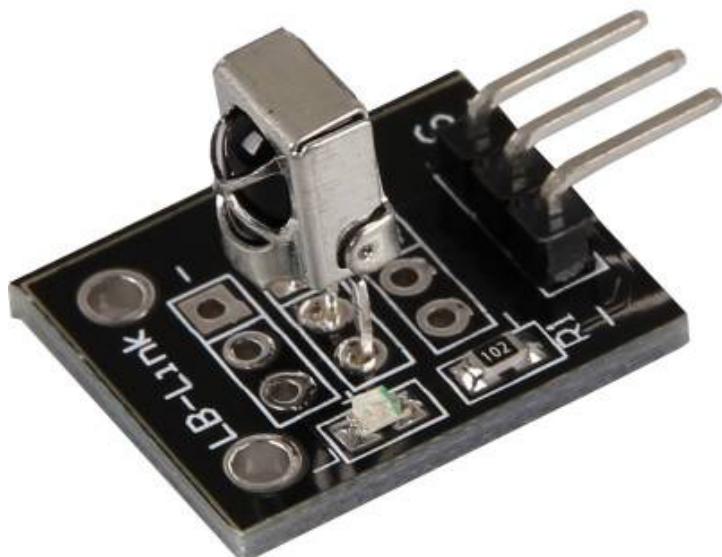
## KY-022 Infrarot Receiver Modul

## KY-022 Infrarot Receiver Modul

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	4
5.1 Installation du logiciel LIRC .....	4
5.2 Test du récepteur IR .....	6
5.3 Apprentissage du code de la télécommande .....	7
5.4 Envoyer des commandes avec l'émetteur infrarouge .....	8

### Photo



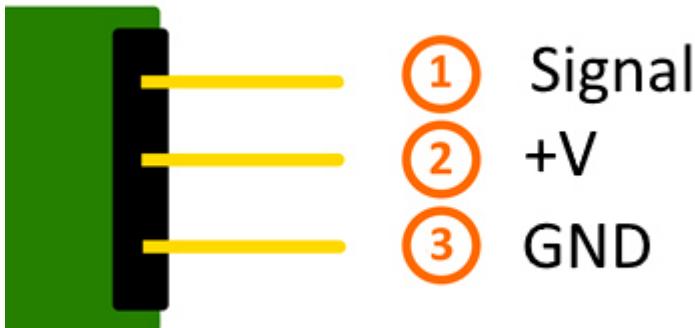
### Données techniques / Description sommaire

Fréquence porteuse: 38kHz

Les signaux infrarouges reçus sont présents en sortie sous forme de séquence numérique.  
En outre la LED intégrée clignote rapidement lorsqu'un signal infrarouge a été détecté.

## Brochage

---



## Exemple de code pour Arduino

---

Il est possible de réaliser un système de télécommande infrarouge à l'aide des [modules émetteur KY-005](#) et récepteur KY-022. Pour le réaliser, il faut également prévoir 2 [cartes Arduino](#). La première agit comme émetteur et la seconde reçoit le signal et l'envoie à la console.

Pour l'exemple de code suivant, une librairie supplémentaire est nécessaire:

- [Arduino-IRremote] de [Ken Shirriff](#) | publiée sous [GPL](#)

La librairie se trouve dans le lien de téléchargement ci-dessous et doit être copiée dans le dossier 'libraries' avant de démarrer l'IDE Arduino.

Voici le chemin d'installation par défaut sous Windows:

C:\Users\[Username]\Documents\Arduino\libraries

Il existe différents protocoles pour l'envoi des données pour les systèmes de transmission infrarouge. Le protocole RC5 est utilisé pour l'exemple suivant. D'autres protocoles sont inclus dans la librairie. Ils se trouvent dans le dossier Documentation/Code.

### Code pour le récepteur

```
// La librairie Arduino-IRremote est ajoutée
#include

// Déclaration de la broche d'entrée correspondante pour le signal de sortie du KY-022
int RECV_PIN = 11;
```

## KY-022 Infrarot Receiver Modul

```
// Initialisation de la librairie Arduino-IRremote
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Démarrage du récepteur IR
}

// Boucle de programme principale
void loop() {

  // On vérifie si un signal est reçu sur le récepteur
  if (irrecv.decode(&results)) {
    // Si un signal est reçu, il est décodé et envoyé dans la console série
    Serial.println(results.value, HEX);
    irrecv.resume();
  }
}
```

## Code pour l'émetteur:

```
// La librairie Arduino-IRremote est ajoutée
#include

// ...et initialisée
IRsend irsend;

// Les réglages de la sortie sont pris en charge par la librairie
// Les sorties correspondantes sont différentes selon la carte Arduino utilisée
// Arduino UNO: Sortie = D3
// Arduino MEGA: Sortie = D9
// Une liste complète des sorties correspondantes est visible à la page:
// http://z3t0.github.io/Arduino-IRremote/
void setup()
{

}

// Boucle de programme principale
void loop() {
  // L'émetteur envoie le signal A90 (en hexadécimal) codé en RC5
  // Cette opération est répétée trois fois, avant une pause de 5 secondes
  for (int i = 0; i < 3; i++) {
    irsend.sendRC5(0xA90, 12);
  }
  // signal à envoyer [0xA90] | Longueur du signal envoyé 12 Bits (hex A90=1010 1001 0000)
  delay(40);
}
delay(5000); // Pause de 5 secondes entre les impulsions d'émission
}
```

## Exemple de programme à télécharger:

[Arduino\\_Fernbedienung.zip](#)

### Affectation des broches Arduino 1 [récepteur]:

KY-022

Signal	=	[Pin 11]
+V	=	[Pin 5V]

## KY-022 Infrarot Receiver Modul

GND = [Pin GND]

### Affectation des broches Arduino 2 [émetteur]:

KY-005

Signal	= [Pin 3 (Arduino Uno)   Pin 9 (Arduino Mega)]
GND+résistance	= [Pin GND*]
GND	= [Pin GND]

- \*Seulement quand une résistance en série est déjà soudée sur le module et n'est pas raccordée au module avant.

## Exemple de code pour Raspberry Pi

Le [Raspberry Pi](#), de par l'architecture avancée de son processeur, a l'avantage de pouvoir faire fonctionner un système d'exploitation Linux. Avec un récepteur infrarouge, non seulement les données simples peuvent être remontées et utilisées, mais des logiciels complets tels que le médiacenter OpenELEC peuvent être commandés via une télécommande infra-rouge.

Un système de contrôle infrarouge peut être mis en place sous Linux avec le logiciel "LIRC" ([www.lirc.org](http://www.lirc.org) | publié sous licence GPL).

Ci-dessous, nous montrons comment installer le logiciel LIRC et commander un Raspberry Pi avec une télécommande infrarouge.

A cet effet, un [module KY-005](#) peut être utilisé comme émetteur et un module KY-022 comme récepteur.

### Affectation des broches GPIO du Raspberry Pi:

*KY-005*

Signal	= GPIO17 [Pin 11]
GND+Widerstand	= GND* [Pin 9]
GND	= GND [Pin 6]

- \*Seulement quand une résistance en série est déjà soudée sur le module et n'est pas raccordée au module avant.

*KY-022*

Signal	= GPIO18 [Pin 12]
+V	= 3,3V [Pin 17]
GND	= GND [Pin 25]

## Installation du logiciel LIRC

Tout d'abord, il faut ouvrir un terminal sur le bureau ou se connecter via SSH avec le Raspberry Pi. On entre ensuite la commande suivante pour installer lirc sur le Raspberry Pi:

## KY-022 Infrarot Receiver Modul

```
sudo apt-get install lirc -y
```

[La carte Raspberry Pi doit être raccordée à internet]

Pour que le module lirc soit directement disponible pour le démarrage du système d'exploitation, il faut ajouter la ligne ci-dessous à la fin du fichier "/boot/config.txt"

```
dtoverlay=lirc-rpi,gpio_in_pin=18,gpio_out_pin=17,gpio_in_pull=up
```

Cette commande définit "gpio\_in\_pin = 18" à la broche d'entrée du récepteur IR et "gpio\_out\_pin = 17" à la broche de sortie de l'émetteur IR.

Le fichier peut être modifié avec la commande suivante:

```
sudo nano /boot/config.txt
```

Après l'ajout de cette ligne, le fichier sera enregistré et fermée en appuyant sur les touches [Ctrl + X -> Y -> Entrée].

Le fichier "/etc/lirc/hardware.conf" doit également être modifié à l'aide de la commande suivante:

```
sudo nano /etc/lirc/hardware.conf
```

Il faut ensuite modifier les expressions suivantes (*DRIVER="UNCONFIGURED"* devient *DRIVER="default"*, etc.):

```
DRIVER="UNCONFIGURED"
--->
DRIVER="default"
DEVICE=""
--->
DEVICE="/dev/lirc0"
MODULES=""
--->
MODULES="lirc_rpi"
```

Le fichier modifié doit ressembler à ceci:

```
# /etc/lirc/hardware.conf
#
# Arguments which will be used when launching lircd
LIRCD_ARGS=""

#Don't start lircmd even if there seems to be a good config file
#START_LIRCMD=false

#Don't start irexec, even if a good config file seems to exist.
#START_IRExec=false

#Try to load appropriate kernel modules
LOAD_MODULES=true
```

## KY-022 Infrarot Receiver Modul

```
# Run "lircd --driver=help" for a list of supported drivers.
DRIVER="default"
# usually /dev/lirc0 is the correct setting for systems using udev
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"

# Default configuration files for your hardware if any
LIRCD_CONF=""
LIRCMD_CONF=""
```

Ensuite, il faut redémarrer la Raspberry Pi en utilisant la commande suivante:

```
sudo reboot
```

## Test du récepteur IR

Pour tester le récepteur, il faut d'abord arrêter lirc avec la commande:

```
sudo /etc/init.d/lirc stop
```

Ensuite, il faut tester ...

```
mode2 -d /dev/lirc0
```

... si la carte Raspberry peut recevoir des signaux. Pour ce faire, prenez une télécommande infrarouge et appuyez sur une touche. Vous devriez voir apparaître des lignes sous la forme suivante:

```
space 95253
pulse 9022
space 2210
pulse 604
space 95246
pulse 9019
space 2211
pulse 601
space 95252
pulse 9019
space 2210
pulse 603
space 95239
pulse 9020
space 2208
pulse 603
...
```

Avec la commande...

```
sudo /etc/init.d/lirc start
```

... le logiciel lirc peut être redémarré.

## Apprentissage du code de la télécommande

Pour enregistrer une télécommande infrarouge dans le système lirc, il faut configurer le fichier "/etc/lirc/lircd.conf". Ce fichier contient les différentes commandes infrarouges.

Afin de formater le fichier lircd.conf correctement, il faut utiliser l'assistance du logiciel lirc qui crée le fichier automatiquement. La procédure est la suivante:

Tout d'abord, il faut arrêter le service lirc

```
sudo /etc/init.d/lirc stop
```

On démarre l'assistant avec la commande suivante:

```
irrecord -d /dev/lirc0 ~/MeineFernbedienung.conf
```

Cet assistant procède à une première initialisation de la télécommande. Il faut appuyer sur les touches en alternance pour que le lirc puisse enregistrer le codage de la télécommande.

Bien suivre les instructions de l'assistant. Après l'initialisation, l'assistant vous demandera le nom du bouton, à enregistrer avec un nouveau code infrarouge. Vous pouvez le faire en sélectionnant l'affectation des touches à partir du fichier suivant:

FernbedienungsCodes.txt

Ceux-ci doivent alors être entrés dans l'assistant et validés par Entrée. On commence ensuite à enregistrer les codes infrarouges pour les touches sélectionnées.

Exemple: Entrez [KEY\_0] -> validez par Entrée -> appuyez sur "0" de la télécommande -> attente jusqu'à ce que l'enregistrement par l'assistant soit confirmé.

On peut sortir de l'assistant en poussant sur Entrée, s'il n'y a plus de touches à configurer. Le fichier de configuration est créé, mais il faut encore affecter un nom à la télécommande. Pour cela, nous ouvrons le fichier dans l'éditeur:

```
sudo nano ~/MeineFernbedienung.conf
```

On peut changer la ligne 17 de

```
name /home/pi/MeineFernbedienung.conf
```

en

```
name MeineFernbedienung
```

Il faut veiller à ne pas inclure d'espace ou utiliser des caractères spéciaux dans le nom. En utilisant la séquence de touches [Ctrl + X -> Y -> Entrée], on enregistre et on ferme le fichier.

## KY-022 Infrarot Receiver Modul

Après avoir créé le fichier de configuration, vous pouvez effectuer un backup du fichier lircd.conf à l'aide de la commande suivante:

```
sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd.conf.bak
```

et avec la commande...

```
sudo cp ~/MeineFernbedienung.conf /etc/lirc/lircd.conf
```

...le nouveau fichier de configuration est utilisé

On peut ensuite...

```
sudo /etc/init.d/lirc start
```

...redémarrer le système lirc.

A partir de maintenant, la nouvelle télécommande est enregistrée dans le système lirc et peut être utilisée avec des logiciels compatibles comme le Mediabrowser OpenElec. Vous pouvez également tester le fonctionnement de la télécommande à l'aide de la commande:

```
irw
```

Pour utiliser la télécommande avec par exemple OpenElec, il faut encore activer lirc dans les options de OpenElec.

## Envoyer des commandes avec l'émetteur infrarouge

Si vous voulez contrôler des dispositifs par infrarouge (par exemple un téléviseur) avec un Raspberry Pi, vous pouvez utiliser les commandes enregistrées précédemment. Ainsi il est possible, par exemple, d'imaginer un logiciel contrôlé par infrarouge ou des périphériques individuels connectés ou non sur le réseau Internet.

Tout d'abord, nous vérifions avec la commande suivante ...

```
irsend LIST MeineFernbedienung ""
```

...quelles commandes sont disponibles.

Nous pouvons envoyer maintenant, par exemple, la commande [KEY\_0] en utilisant la commande suivante:

```
irsend SEND_ONCE MeineFernbedienung KEY_0
```

Le téléviseur ou le récepteur devrait réagir à la commande. Pour l'appareil ci-dessus, la commande est répétitive.

## KY-022 Infrarot Receiver Modul

```
irsend SEND_START MeineFernbedienung KEY_0
```

Le code [KEY\_0] est répétitif ...

```
irsend SEND_STOP MeineFernbedienung KEY_0
```

...jusqu'à ce qui soit stoppé. Ceci est utile pour les fonctions 'volume' ou 'luminosité'.

## KY-023 Module Joystick X-Y

## KY-023 Module Joystick X-Y

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	3
4 Exemple de code pour Arduino .....	3
5 Exemple de code pour Raspberry Pi .....	4

### Photo

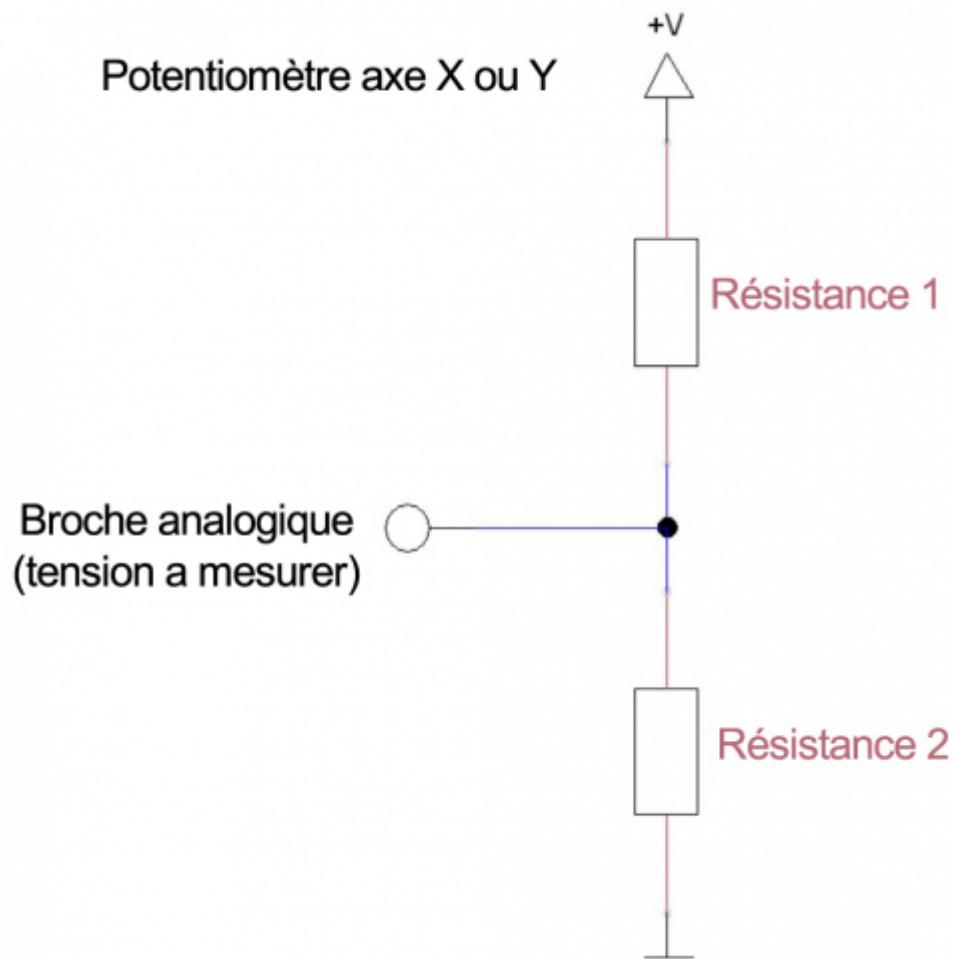


### Données techniques / Description sommaire

Un signal sous forme de tension analogique est envoyé aux sorties lorsqu'on faire varier les positions sur les axes X et Y.

Deux potentiomètres ont été installés dans ce joystick, respectivement pour l'axe X et l'axe Y. Ceux-ci forment un diviseur de tension, comme cela est montré dans la figure suivante.

KY-023 Module Joystick X-Y



Au repos, le potentiomètre est en position milieu, de sorte que Resistance1 = Resistance2. La tension de 5 Vcc appliquée est répartie uniformément sur les deux, ce qui donne 2,5 Vcc sur chaque résistance.

Si la position de l'axe X est modifiée, les résistances respectives varieront en fonction de l'emplacement actuel.

La tension au point milieu varie en fonction de la valeur des résistances selon la position de l'axe. La valeur de la tension permet donc de déterminer la position de l'axe.

## Brochage



- 1 Bouton
- 2 Position Y
- 3 Position X
- 4 +V
- 5 GND

## Exemple de code pour Arduino

Le programme lit les valeurs actuelles des broches d'entrée, les convertit en une tension (0-1023 -> 0V-5V) et les envoie à la sortie série.

```
// Déclaration et initialisation des broches d'entrées
int JoyStick_X = A0; // Signal de l'axe X
int JoyStick_Y = A1; // Signal de l'axe Y
int Button = 3; // Bouton

void setup ()
{
  pinMode (JoyStick_X, INPUT);
  pinMode (JoyStick_Y, INPUT);
  pinMode (Button, INPUT);

  // Lorsqu'on pousse sur le bouton, la mise à la masse
  // active la résistance de PullUp.
  digitalWrite(Button, HIGH);

  Serial.begin (9600); // Sortie série à 9600 bauds
}

// Le programme lit les valeurs des broches d'entrée et les envoie à la sortie série
void loop ()
{
  float x, y;
  int Knopf;

  //Les valeurs sont lues, sont converties en tension...
  x = analogRead (JoyStick_X) * (5.0 / 1023.0);
  y = analogRead (JoyStick_Y) * (5.0 / 1023.0);
  Knopf = digitalRead (Button);
```

## KY-023 Module Joystick X-Y

```
//... et envoyées à la sortie série.
Serial.print ("Axe X:"); Serial.print (x, 4); Serial.print ("V, ");
Serial.print ("Axe Y:"); Serial.print (y, 4); Serial.print ("V, ");
Serial.print ("Bouton:");

if(Knopf==1)
{
    Serial.println (" pas de pression sur le bouton");
}
else
{
    Serial.println (" pression sur le bouton");
}
delay (200);
}
```

### Affectation des broches Arduino:

Bouton	= [Pin 3]
Position Y	= [Pin A1]
Position X	= [Pin A0]
Sensor +V	= [Pin 5V]
Sensor GND	= [Pin GND]

### Exemple de programme à télécharger

[KY-023\\_Joystick\\_Modul.zip](#)

## Exemple de code pour Raspberry Pi

---

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

Vous trouverez de plus amples informations à ce sujet dans la description du [module KY-053](#).

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Le programme lit les tensions en entrées et les transmet à la console en mV.

**KY-023 Module Joystick X-Y**

```

#
#!/usr/bin/python
# coding=utf-8

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### KY-023 Joystick Module - Raspberry Pi Python Code Example
#####

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde
# sps = 32 # 32 échantillons par seconde
sps = 64 # 64 échantillons par seconde
# sps = 128 # 128 échantillons par seconde
# sps = 250 # 250 échantillons par seconde
# sps = 475 # 475 échantillons par seconde
# sps = 860 # 860 échantillons par seconde

# choix du canal ADC (1-4)
x_adc_channel = 0 # Channel 0 für die x-Achse
y_adc_channel = 1 # Channel 1 für die y-Achse
# adc_channel = 2 # Channel 2
# adc_channel = 3 # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

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

#####
# boucle de programme principale
#####

```

## KY-023 Module Joystick X-Y

```

# boucle de programme principale
# #####
# Le programme lit les tensions en entrées et les transmet à la console en mV.

try:
    while True:
        #Les valeurs de tension sont enregistrées
        x = adc.readADCSingleEnded(x_adc_channel, gain, sps)
        y = adc.readADCSingleEnded(y_adc_channel, gain, sps)

        # Envoi vers la console
        if GPIO.input(Button_PIN) == True:
            print "Axe X:", x, "mV, ", "Axe Y:", y, "mV, Bouton: n'est pas poussé"
        else:
            print "Axe X:", x, "mV, ", "Axe Y:", y, "mV, Bouton: poussé"
        print "-----"

        # Reset + Delay
        button_pressed = False
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()

```

### **Brochage Raspberry Pi:**

Sensor KY-023

Bouton	= GPIO24	[Pin 18 (RPi)]
Position Y	= Analog 1	[Pin A1 (ADS1115 - KY-053)]
Position X	= Analog 0	[Pin A0 (ADS1115 - KY-053)]
+V	= 3,3V	[Pin 1 (RPi)]
GND	= Masse	[Pin 6 (RPi)]

ADS1115 - KY-053:

VDD	= 3,3V	[Pin 01]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05 (RPi)]
SDA	= GPIO02 / SDA	[Pin 03 (RPi)]
A0	= s.o.	[Capteur: Position X (KY-023)]
A1	= s.o.	[Capteur: Position Y (KY-023)]

### **Exemple de programme à télécharger**

[KY-023\\_RPi\\_JoystickModule.zip](#)

Commande pour lancer le programme:

```
sudo python KY-023_RPi_JoystickModule.py
```

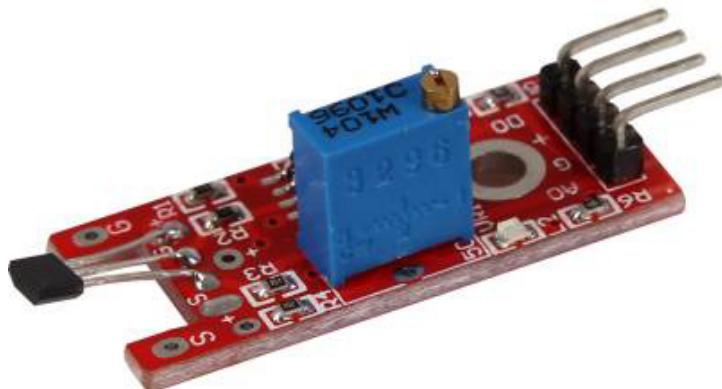
KY-024 Capteur à effet Hall

## KY-024 Capteur à effet Hall

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Fonctionnement du capteur .....	2
5 Exemple de code pour Arduino .....	3
6 Exemple de code pour Raspberry Pi .....	4

### Photo



### Données techniques / Description sommaire

Chipset: A3141 | Ampli opérationnel: LM393

Le champ magnétique est mesuré par le capteur et une tension analogique proportionnelle se retrouve en sortie. La sensibilité du capteur peut être réglée par le potentiomètre.

**Sortie numérique:** signal présent si un champ magnétique est détecté

**Sortie analogique:** mesure directe du capteur

**LED1:** indique que le capteur est alimenté en tension

**LED2:** indique qu'un champ magnétique est détecté

## Brochage

---



- ① Signal numérique
- ② +V
- ③ GND
- ④ Signal analogique

## Fonctionnement du capteur

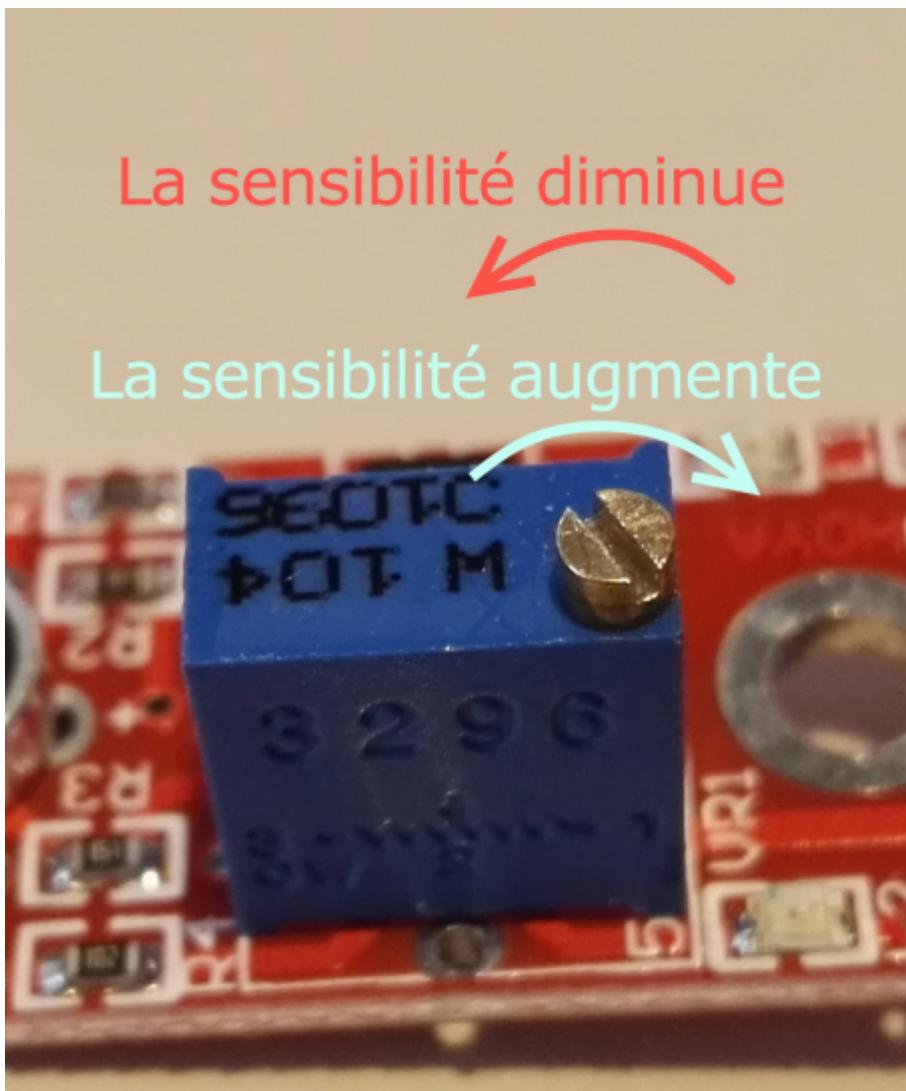
---

Ce module est composé de trois éléments fonctionnels. Le capteur situé à l'avant du module effectue la mesure, le signal analogique est ensuite envoyé sur l'amplificateur. Celui-ci amplifie le signal en fonction du gain déterminé par le potentiomètre et envoie le signal à la sortie analogique du module.

Il convient de noter que le signal est inversé: plus la valeur mesurée par le capteur est haute, plus la tension de sortie est faible.

La troisième partie est composée d'un comparateur qui commute la sortie numérique et la diode lorsque le signal tombe en dessous d'une certaine valeur. La sensibilité peut être ajustée au moyen du potentiomètre comme décrit ci-dessous:

KY-024 Capteur à effet Hall



Ce type de capteur ne délivre pas des valeurs absolues (par exemple, la température mesurée avec précision en ° C ou de la force du champ magnétique en mT), mais des valeurs relatives. On définit une valeur limite par rapport à une valeur normale donnée et le module émet un signal si cette limite est dépassée.

Ce fonctionnement est idéal pour la surveillance de la température (KY-028), les détecteurs de proximité (KY-024, KY 025, KY-036), la surveillance des alarmes (KY-037, KY-038) ou le détecteur de flamme (KY-026).

## Exemple de code pour Arduino

Le programme lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```
// Déclaration et initialisation des broches d'entrées
int Analog_Eingang = A0; // Entrée analogique
int Digital_Eingang = 3; // Entrée digitale
```

## KY-024 Capteur à effet Hall

```

void setup ()
{
    pinMode (Analog_Eingang, INPUT);
    pinMode (Digital_Eingang, INPUT);

    Serial.begin (9600); // Sortie série à 9600 bauds
}

// Le programme lit les valeurs des broches d'entrée et les envoie à la sortie série
void loop ()
{
    float Analog;
    int Digital;

    //Les valeurs sont lues, sont converties en tension...
    Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
    Digital = digitalRead (Digital_Eingang);

    //... et envoyées à la sortie série.
    Serial.print ("Tension analogique:"); Serial.print (Analog, 4); Serial.print ("V, ");
    Serial.print ("Limite:");

    if(Digital==1)
    {
        Serial.println (" atteinte");
    }
    else
    {
        Serial.println (" pas encore atteinte");
    }
    Serial.println ("-----");
    delay (200);
}

```

### Affectation des broches Arduino:

Signal numérique = [Pin 3]  
 +V = [Pin 5V]  
 GND = [Pin GND]  
 Signal analogique = [Pin A0]

### Exemple de programme à télécharger

[Ard\\_Analoger\\_Sensor.zip](#)

## Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

## KY-024 Capteur à effet Hall

Vous trouverez de plus amples informations à ce sujet dans la description du module KY-053.

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```

#
#!/usr/bin/python
# coding=utf-8

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Analog Sensor + ADS1115 ADC - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde

```

## KY-024 Capteur à effet Hall

```

# sps = 32    # 32 échantillons par seconde
sps = 64    # 64 échantillons par seconde
# sps = 128   # 128 échantillons par seconde
# sps = 250   # 250 échantillons par seconde
# sps = 475   # 475 échantillons par seconde
# sps = 860   # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel = 0    # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

# Sélection de la broche d'entrée du signal numérique
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

#####
# boucle de programme principale
#####
# Le programme lit les tensions en entrées et les transmet à la console.

try:
    while True:
        #Les valeurs de tension sont enregistrées
        analog = adc.readADCSingleEnded(adc_channel, gain, sps)

        # Envoi vers la console
        if GPIO.input(Digital_PIN) == False:
            print "Tension analogique:", analog,"mV, ","Limite: pas encore atteinte"
        else:
            print "Tension analogique:", analog, "mV, ", "Limite: atteinte"
        print "-----"

        # Reset + Delay
        button_pressed = False
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()

```

### **Brochage Raspberry Pi:**

#### Capteur

Signal numérique	= GPIO 24	[Pin 18 (RPi)]
+V	= 3,3V	[Pin 1 (RPi)]
GND	= Masse	[Pin 06 (RPi)]
Signal analogique	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

#### ADS1115 - KY-053:

VDD	= 3,3V	[Pin 01]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]

## KY-024 Capteur à effet Hall

SDA = GPIO02 / SDA [Pin 03]  
A0 = s.o. [Capteur: signal analogique]

**Exemple de programme à télécharger**

[RPi\\_AnalogSensor.zip](#)

Commande pour lancer le programme:

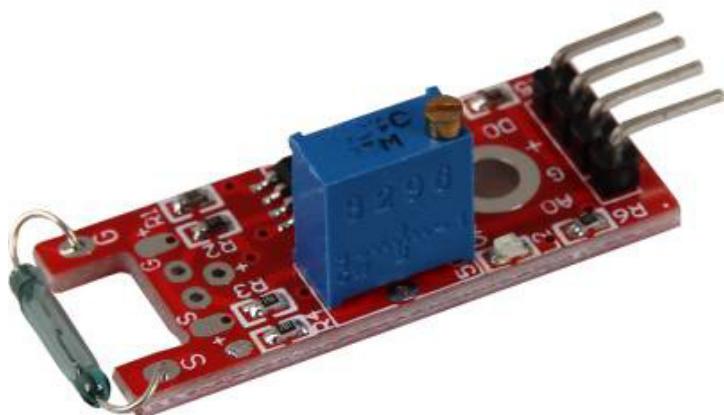
```
sudo python RPi_AnalogSensor.py
```

## KY-025 Module Reed

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Fonctionnement du capteur .....	3
5 Exemple de code pour Arduino .....	4
6 Exemple de code pour Raspberry Pi .....	5

### Photo



### Données techniques / Description sommaire

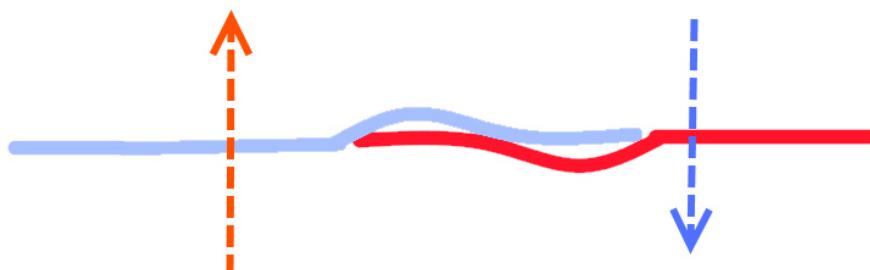
Si un champ magnétique est détecté, la sortie numérique commute.

Les contacts Reed sont composés de deux lames de contact souples situées à l'intérieur d'un tube en verre. Ces lames se déplacent l'une vers l'autre si un champ magnétique est présent dans le voisinage (passage d'un aimant).

## KY-025 Module Reed



Sans aimant



Avec aimant

Le contact se ferme en présence d'un champ magnétique.

**Sortie numérique:** contact si un champ magnétique est détecté

**Sortie analogique:** mesure directe du capteur, la sortie analogique est identique à la sortie numérique sur ce type de capteur (capteur tout ou rien).

**LED1:** indique que le capteur est alimenté en tension

**LED2:** indique qu'un champ magnétique est détecté

## Brochage



- ① Signal numérique
- ② +V
- ③ GND
- ④ Signal analogique

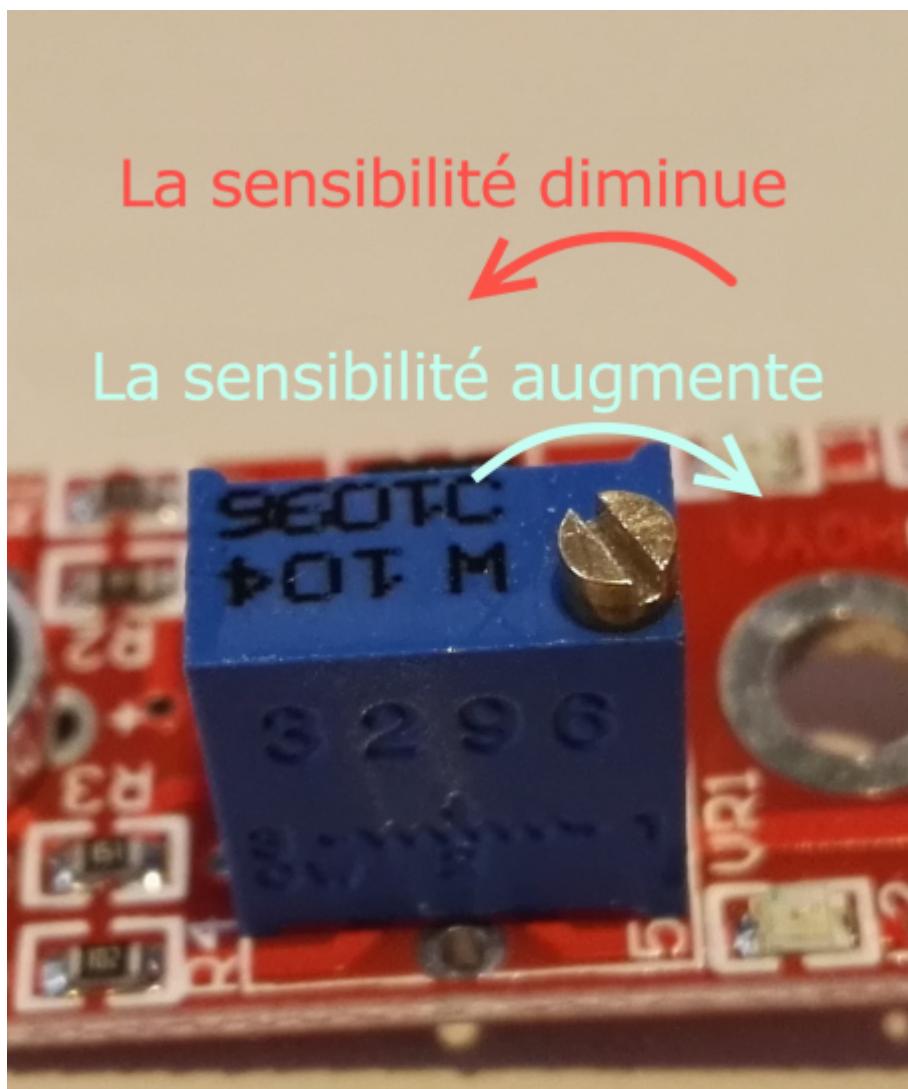
KY-025 Module Reed

## Fonctionnement du capteur

Ce module est composé de trois éléments fonctionnels. Le capteur situé à l'avant du module effectue la mesure, le signal analogique est ensuite envoyé sur l'amplificateur. Celui-ci amplifie le signal en fonction du gain déterminé par le potentiomètre et envoie le signal à la sortie analogique du module.

Il convient de noter que le signal est inversé: plus la valeur mesurée par le capteur est haute, plus la tension de sortie est faible.

La troisième partie est composée d'un comparateur qui commute la sortie numérique et la diode lorsque le signal tombe en dessous d'une certaine valeur. La sensibilité peut être ajustée au moyen du potentiomètre comme décrit ci-dessous:



## KY-025 Module Reed

Ce type de capteur ne délivre pas des valeurs absolues (par exemple, la température mesurée avec précision en ° C ou de la force du champ magnétique en mT), mais des valeurs relatives. On définit une valeur limite par rapport à une valeur normale donnée et le module émet un signal si cette limite est dépassée.

Ce fonctionnement est idéal pour la surveillance de la température (KY-028), les détecteurs de proximité (KY-024, KY-025, KY-036), la surveillance des alarmes (KY-037, KY-038) ou le détecteur de flamme (KY-026).

### Exemple de code pour Arduino

Le programme lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```
// Déclaration et initialisation des broches d'entrées
int Analog_Eingang = A0; // X-Achse-Signal
int Digital_Eingang = 3; // Knopf

void setup ()
{
    pinMode (Analog_Eingang, INPUT);
    pinMode (Digital_Eingang, INPUT);

    Serial.begin (9600); // Sortie série à 9600 bauds
}

// Le programme lit les valeurs des broches d'entrée et les envoie à la sortie série
void loop ()
{
    float Analog;
    int Digital;

    //Les valeurs sont lues, sont converties en tension...
    Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
    Digital = digitalRead (Digital_Eingang);

    //... et envoyées à la sortie série.
    Serial.print ("Tension analogique:"); Serial.print (Analog, 4); Serial.print ("V, ");
    Serial.print ("Limite:");

    if(Digital==1)
    {
        Serial.println (" atteinte");
    }
    else
    {
        Serial.println (" pas encore atteinte");
    }
    Serial.println ("-----");
    delay (200);
}
```

#### Affectation des broches Arduino:

Signal numérique = [Pin 3]  
 +V = [Pin 5V]

## KY-025 Module Reed

GND = [Pin GND]  
 Signal analogique = [Pin 0]

### Exemple de programme à télécharger

[Ard\\_Analoger\\_Sensor.zip](#)

## Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

Vous trouverez de plus amples informations à ce sujet dans la description du [module KY-053](#).

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```

#
#!/usr/bin/python
# coding=utf-8

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Analog Sensor + ADS1115 ADC - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15

```

**KY-025 Module Reed**

```

from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde
# sps = 32 # 32 échantillons par seconde
sps = 64 # 64 échantillons par seconde
# sps = 128 # 128 échantillons par seconde
# sps = 250 # 250 échantillons par seconde
# sps = 475 # 475 échantillons par seconde
# sps = 860 # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel = 0 # Channel 0
# adc_channel = 1 # Channel 1
# adc_channel = 2 # Channel 2
# adc_channel = 3 # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

# Sélection de la broche d'entrée du signal numérique
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

#####
# boucle de programme principale
#####
# #####
# Le programme lit les tensions en entrées et les transmet à la console.

try:
    while True:
        #Les valeurs de tension sont enregistrées
        analog = adc.readADCSingleEnded(adc_channel, gain, sps)

        # Envoi vers la console
        if GPIO.input(Digital_PIN) == False:
            print "Tension analogique:", analog,"mV, ","Limite: pas encore atteinte"
        else:
            print "Tension analogique:", analog, "mV, ", "Limite: atteinte"
        print "-----"

        # Reset + Delay
        button_pressed = False

        time.sleep(delayTime)

```

## KY-025 Module Reed

```
    time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()
```

**Brochage Raspberry Pi:**

Capteur

Signal numérique	= GPIO 24	[Pin 18 (RPi)]
+V	= 3,3V	[Pin 1 (RPi)]
GND	= Masse	[Pin 06 (RPi)]
Signal analogique	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

## ADS1115 - KY-053:

VDD	= 3,3V	[Pin 01]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]
A0	= s.o.	[Sensor: analoges Signal]

**Exemple de programme à télécharger**[RPi\\_AnalogSensor.zip](#)

Commande pour lancer le programme:

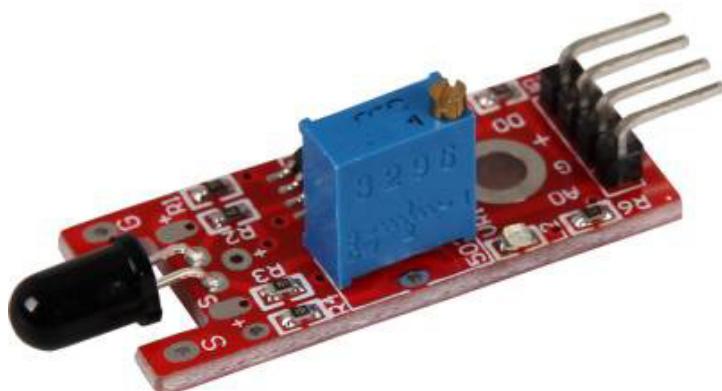
```
sudo python RPi_AnalogSensor.py
```

## KY-026 DéTECTeur de flamme

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Fonctionnement du capteur .....	2
5 Exemple de code pour Arduino .....	3
6 Exemple de code pour Raspberry Pi .....	4

### Photo



### Données techniques / Description sommaire

La photodiode est sensible spectre lumineux généré par une flamme.

**Sortie numérique:** un signal est émis si une flamme détectée.

**Sortie analogique:** mesure directe du capteur

**LED1:** indique que le capteur est alimenté en tension

**LED2:** indique qu'une flamme est détectée

## Brochage

---



- ① Signal numérique
- ② +V
- ③ GND
- ④ Signal analogique

## Fonctionnement du capteur

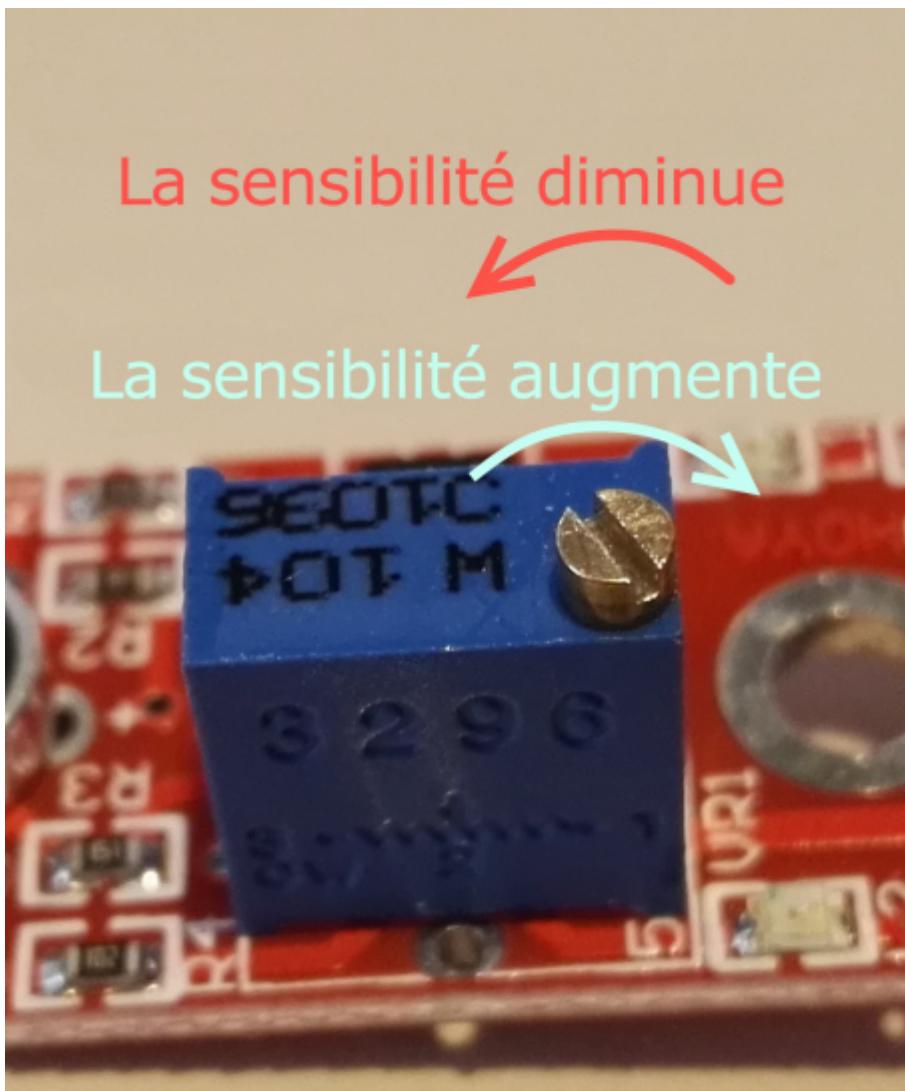
---

Ce module est composé de trois éléments fonctionnels. Le capteur situé à l'avant du module effectue la mesure, le signal analogique est ensuite envoyé sur l'amplificateur. Celui-ci amplifie le signal en fonction du gain déterminé par le potentiomètre et envoie le signal à la sortie analogique du module.

Il convient de noter que le signal est inversé: plus la valeur mesurée par le capteur est haute, plus la tension de sortie est faible.

La troisième partie est composée d'un comparateur qui commute la sortie numérique et la diode lorsque le signal tombe en dessous d'une certaine valeur. La sensibilité peut être ajustée au moyen du potentiomètre comme décrit ci-dessous:

KY-026 DéTECTEUR de flamme



Ce type de capteur ne délivre pas des valeurs absolues (par exemple, la température mesurée avec précision en ° C ou de la force du champ magnétique en mT), mais des valeurs relatives. On définit une valeur limite par rapport à une valeur normale donnée et le module émet un signal si cette limite est dépassée.

Ce fonctionnement est idéal pour la surveillance de la température (KY-028), les détecteurs de proximité (KY-024, KY 025, KY-036), la surveillance des alarmes (KY-037, KY-038) ou le détecteur de flamme (KY-026).

## Exemple de code pour Arduino

Le programme lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```
// Déclaration et initialisation des broches d'entrées
int Analog_Eingang = A0; // Signal analogique
int Digital_Eingang = 3; // Signal numérique
```

## KY-026 DéTECTEUR de flamme

```

void setup ()
{
    pinMode (Analog_Eingang, INPUT);
    pinMode (Digital_Eingang, INPUT);

    Serial.begin (9600); // Sortie série à 9600 bauds
}

// Le programme lit les valeurs des broches d'entrée et les envoie à la sortie série
void loop ()
{
    float Analog;
    int Digital;

    //Les valeurs sont lues, sont converties en tension...
    Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
    Digital = digitalRead (Digital_Eingang);

    //... et envoyées à la sortie série.
    Serial.print ("Tension analogique:"); Serial.print (Analog, 4); Serial.print ("V, ");
    Serial.print ("Limite:");

    if(Digital==1)
    {
        Serial.println (" atteinte");
    }
    else
    {
        Serial.println (" pas encore atteinte");
    }
    Serial.println ("-----");
    delay (200);
}

```

### Affectation des broches Arduino:

Signal numérique = [Pin 3]  
 +V = [Pin 5V]  
 GND = [Pin GND]  
 Signal analogique = [Pin A0]

### Exemple de programme à télécharger

[Ard\\_Analoger\\_Sensor.zip](#)

## Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

Vous trouverez de plus amples informations à ce sujet dans la description du [module KY-053](#).

## KY-026 DéTECTEUR de flamme

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous licence [BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```

#
#!/usr/bin/python
# coding=utf-8

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Analog Sensor + ADS1115 ADC - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde
# sps = 32 # 32 échantillons par seconde
sps = 64 # 64 échantillons par seconde

```

## KY-026 DéTECTEUR de flamme

```

# sps = 128 # 128 échantillons par seconde
# sps = 250 # 250 échantillons par seconde
# sps = 475 # 475 échantillons par seconde
# sps = 860 # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel = 0      # Channel 0
# adc_channel = 1      # Channel 1
# adc_channel = 2      # Channel 2
# adc_channel = 3      # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

# Sélection de la broche d'entrée du signal numérique
Digital_PIN = 24
GPIO0.setup(Digital_PIN, GPIO0.IN, pull_up_down = GPIO0.PUD_OFF)

#####
# boucle de programme principale
#####
# Le programme lit les tensions en entrées et les transmet à la console.

try:
    while True:
        #Les valeurs de tension sont enregistrées
        analog = adc.readADCSingleEnded(adc_channel, gain, sps)

        # Envoi vers la console
        if GPIO0.input(Digital_PIN) == False:
            print "Tension analogique:", analog, "mV, ", "Limite: pas encore atteinte"
        else:
            print "Tension analogique:", analog, "mV, ", "Limite: atteinte"
        print "-----"

        # Reset + Delay
        button_pressed = False
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO0.cleanup()

```

### **Brochage Raspberry Pi:**

#### Capteur

Signal numérique	= GPIO 24	[Pin 18 (RPi)]
+V	= 3,3V	[Pin 1 (RPi)]
GND	= Masse	[Pin 06 (RPi)]
Signal analogique	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

#### ADS1115 - KY-053:

VDD	= 3,3V	[Pin 01]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]

## KY-026 DéTECTEUR de flamme

A0 = s.o. [Sensor: analoge Signal]

**Exemple de programme à télécharger**[RPi\\_AnalogSensor.zip](#)

Commande pour lancer le programme:

```
sudo python RPi_AnalogSensor.py
```

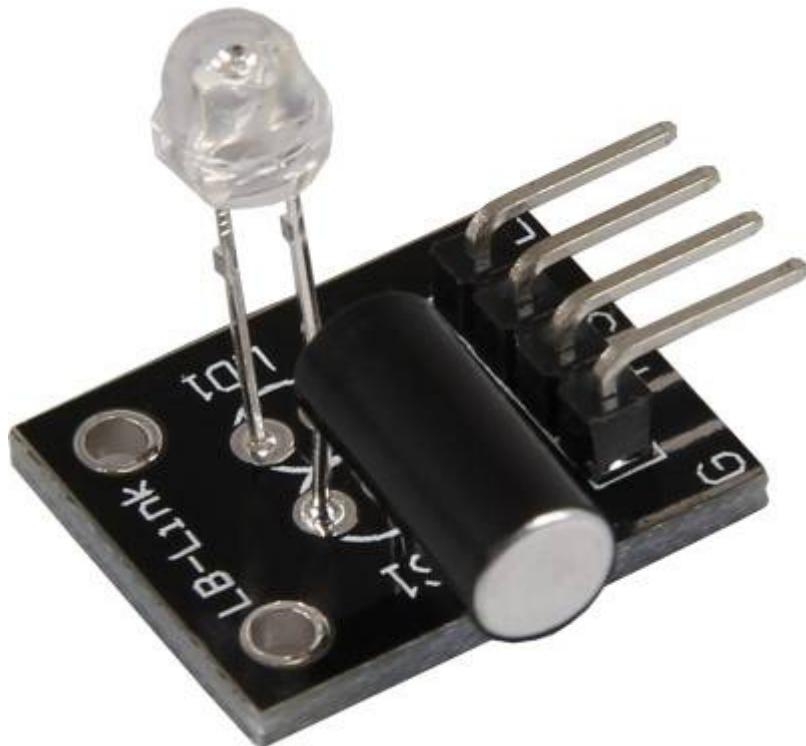
## KY-027 Module Magic Light Cup

## KY-027 Module Magic Light Cup

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	3
4 Exemple de code pour Arduino .....	3
5 Exemple de code pour Raspberry Pi .....	4

### Photo



### Données techniques / Description sommaire

La LED s'allume un fonction de l'inclinaison du module. Lorsque la LED est allumée, le signal est envoyé vers la sortie du module. Une résistance de limitation est nécessaire pour alimenter la led.

#### **Résistance de limitation:**

**R<sub>f</sub> (3,3V) [Rouge] = 120Ω**

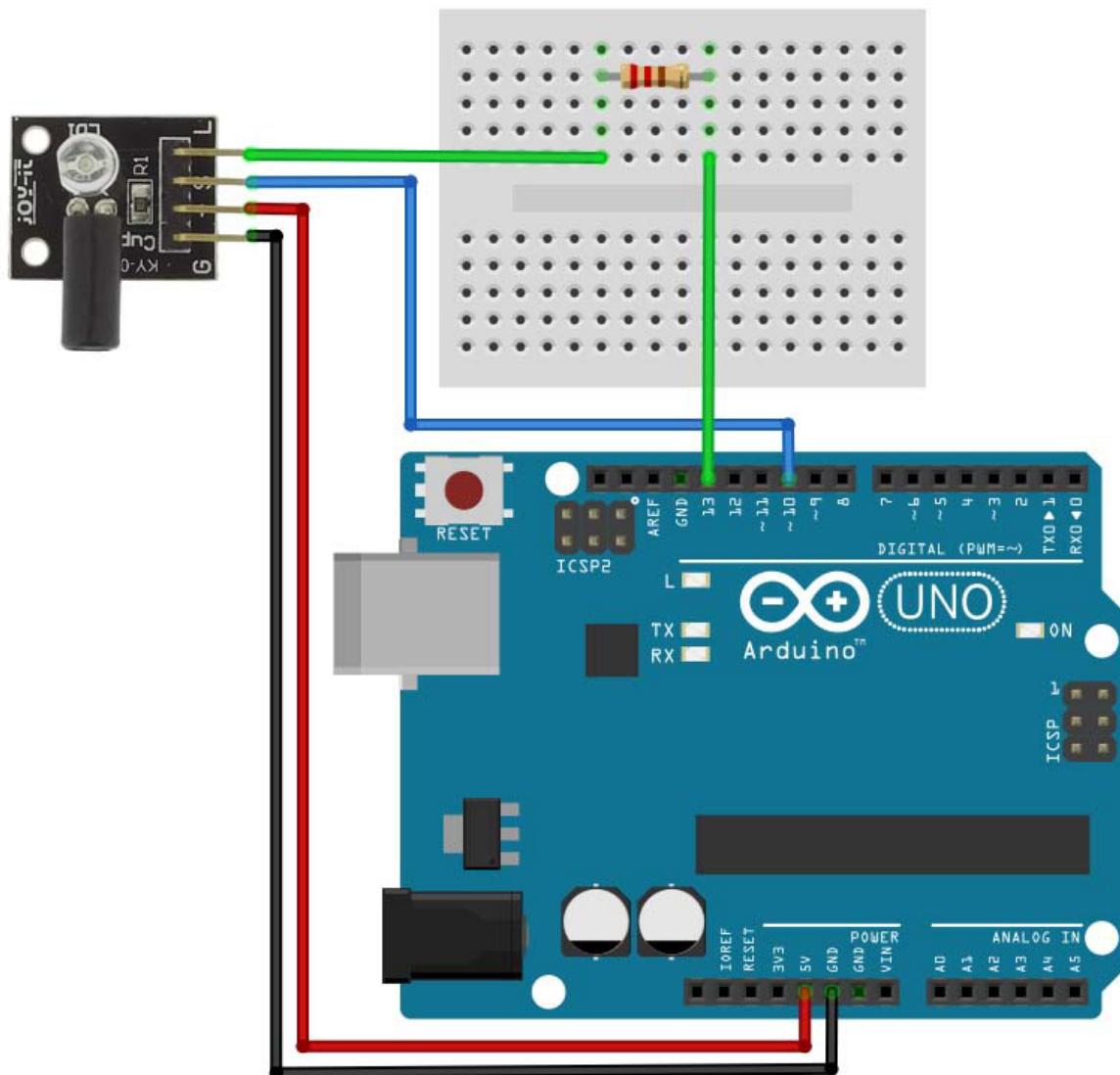
*[Valeurs calculées lors de l'utilisation avec des microcontrôleurs ARM de base tel que la Raspberry Pi]*

## KY-027 Module Magic Light Cup

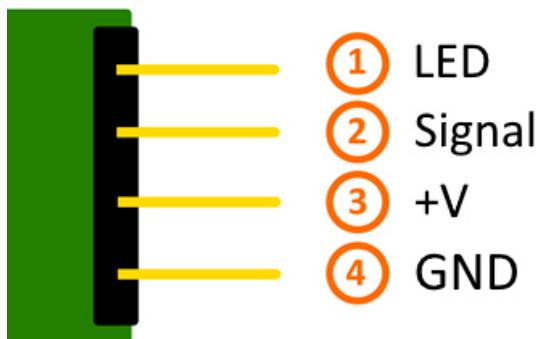
**R<sub>f</sub> (5V) [Rouge] = 220Ω**

[Valeurs calculées lors de l'utilisation avec des microcontrôleurs Atmel Atmega tel que Arduino]

Exemple d'utilisation avec une carte Arduino:



## Brochage



## Exemple de code pour Arduino

L'exemple de programme ci-dessous fait allumer la LED du module Magic Light Cup lorsque ce module atteint une certaine inclinaison.

```
int Led = 13; // Initialisation de la broche de sortie
int Sensor = 10; // Déclaration de la broche d'entrée du capteur
int val; // Variable temporaire

void setup ()
{
    pinMode (Led, OUTPUT); // Initialisation de la broche de sortie
    pinMode (Sensor, INPUT); // Initialisation de la broche du capteur
    digitalWrite(Sensor, HIGH); // Activation de la résistance de Pull-up interne
}

void loop ()
{
    val = digitalRead (Sensor); // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        digitalWrite (Led, LOW);
    }
    else
    {
        digitalWrite (Led, HIGH);
    }
}
```

### Affectation des broches Arduino:

LED +	= [Pin 13]
LED -	= [Pin GND]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]
Sensor -	= [Pin GND]

## KY-027 Module Magic Light Cup

### Exemple de programme à télécharger

[SensorTest\\_Arduino.zip](#)

### Exemple de code pour Raspberry Pi

#### Exemple de programmation en Python

L'exemple de programme ci-dessous fait allumer la LED du module Magic Light Cup lorsque ce module atteint une certaine inclinaison.

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Déclaration des broches de raccordement du capteur et de la diode
LED_PIN = 24
Sensor_PIN = 23
GPIO.setup(Sensor_PIN, GPIO.IN)
GPIO.setup(LED_PIN, GPIO.OUT)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Cette fonction de sortie est exécutée par détection du signal
def ausgabeFunktion(null):
    GPIO.output(LED_PIN, True)

# Lors de la détection d'un signal (front descendant du signal) de la fonction de sortie est déclenchée
GPIO.add_event_detect(Sensor_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=10)

# Boucle de programme principale
try:
    while True:
        time.sleep(1)
        #La sortie sera réinitialisée si le capteur d'inclinaison est repositionné dans son état initial
        if GPIO.input(Sensor_PIN):
            GPIO.output(LED_PIN, False)

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

LED	=	GPIO24	[Pin 18]
Signal	=	GPIO23	[Pin 16]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger

[KY-027-RPi-MagicLightCup.zip](#)

Commande pour lancer le programme:

```
sudo python KY-027-RPi-MagicLightCup.py
```

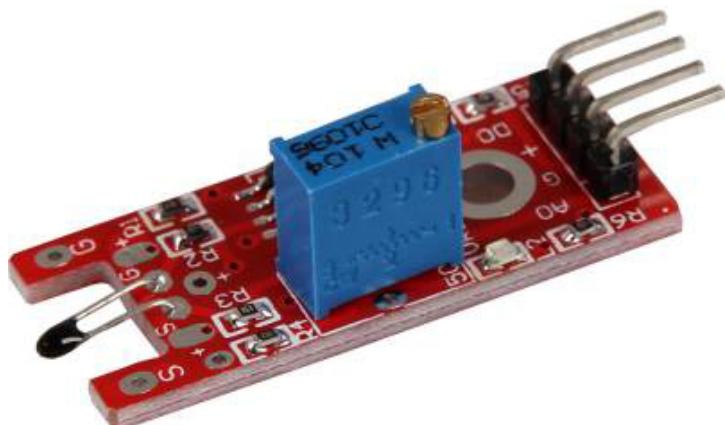
## KY-028 Module capteur de t° CTN

## KY-028 Module capteur de t° CTN

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Fonctionnement du capteur .....	2
5 Exemple de code pour Arduino .....	3
6 Exemple de code pour Raspberry Pi .....	4

### Photo



### Données techniques / Description sommaire

Plage de mesure de température: -55°C à + 125°C. Ce module comprend une thermistance CTN (coefficent de température négatif - la résistance diminue lorsque la température augmente).

**Sortie numérique:** signal présent si la température mesurée est supérieure à un seuil préréglé  
**Sortie analogique:** mesure directe de la valeur du capteur

**LED1:** indique que le capteur est alimenté en tension

**LED2:** indique que la température dépasse le seuil

## Brochage

---



- ① Signal numérique
- ② +V
- ③ GND
- ④ Signal analogique

## Fonctionnement du capteur

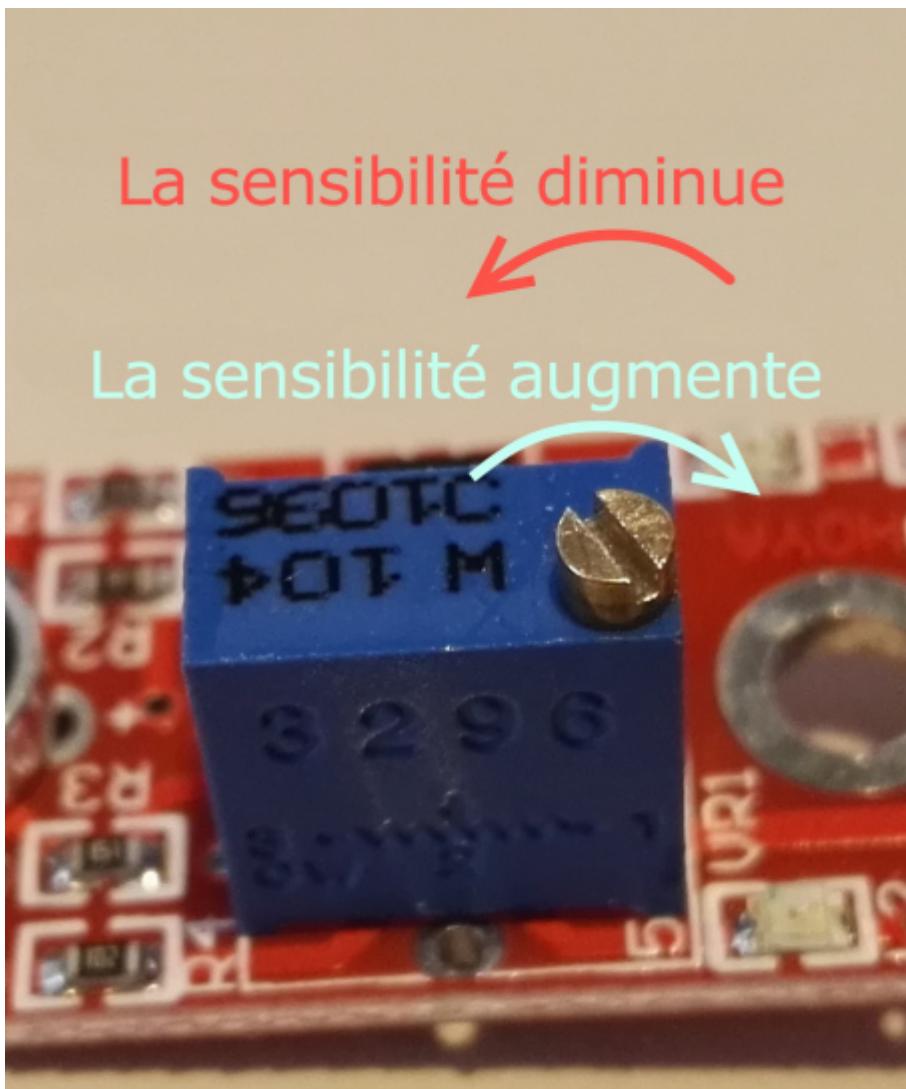
---

Ce module est composé de trois éléments fonctionnels. Le capteur situé à l'avant du module effectue la mesure, le signal analogique est ensuite envoyé sur l'amplificateur. Celui-ci amplifie le signal en fonction du gain déterminé par le potentiomètre et envoie le signal à la sortie analogique du module.

Il convient de noter que le signal est inversé: plus la valeur mesurée par le capteur est haute, plus la tension de sortie est faible.

La troisième partie est composée d'un comparateur qui commute la sortie numérique et la diode lorsque le signal tombe en dessous d'une certaine valeur. La sensibilité peut être ajustée au moyen du potentiomètre comme décrit ci-dessous:

KY-028 Module capteur de t° CTN



Ce type de capteur ne délivre pas des valeurs absolues (par exemple, la température mesurée avec précision en ° C ou de la force du champ magnétique en mT), mais des valeurs relatives. On définit une valeur limite par rapport à une valeur normale donnée et le module émet un signal si cette limite est dépassée.

Ce fonctionnement est idéal pour la surveillance de la température (KY-028), les détecteurs de proximité (KY-024, KY-025, KY-036), la surveillance des alarmes (KY-037, KY-038) ou le détecteur de flamme (KY-026).

## Exemple de code pour Arduino

Le programme lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```
// Déclaration et initialisation des broches d'entrées
int Analog_Eingang = A0; // Entrée analogique
int Digital_Eingang = 3; // Entrée digitale
```

## KY-028 Module capteur de t° CTN

```

void setup ()
{
    pinMode (Analog_Eingang, INPUT);
    pinMode (Digital_Eingang, INPUT);

    Serial.begin (9600); // Sortie série à 9600 bauds
}

// Le programme lit les valeurs des broches d'entrée et les envoie à la sortie série
void loop ()
{
    float Analog;
    int Digital;

    //Les valeurs sont lues, sont converties en tension...
    Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
    Digital = digitalRead (Digital_Eingang);

    //... et envoyées à la sortie série.
    Serial.print ("Tension analogique:"); Serial.print (Analog, 4); Serial.print ("V, ");
    Serial.print ("Limite:");

    if(Digital==1)
    {
        Serial.println (" atteinte");
    }
    else
    {
        Serial.println (" pas encore atteinte");
    }
    Serial.println ("-----");
    delay (200);
}

```

### Affectation des broches Arduino:

digitales Signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analoges Signal	= [Pin 0]

### Exemple de programme à télécharger

[Ard\\_Analoger\\_Sensor.zip](#)

## Exemple de code pour Raspberry Pi

---

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

## KY-028 Module capteur de t° CTN

Vous trouverez de plus amples informations à ce sujet dans la description du module KY-053.

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```

#
#!/usr/bin/python
# coding=utf-8

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Analog Sensor + ADS1115 ADC - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde

```

## KY-028 Module capteur de t° CTN

```

# sps = 32    # 32 échantillons par seconde
sps = 64    # 64 échantillons par seconde
# sps = 128   # 128 échantillons par seconde
# sps = 250   # 250 échantillons par seconde
# sps = 475   # 475 échantillons par seconde
# sps = 860   # 860 échantillons par seconde
choix du canal ADC (1-4)
adc_channel = 0    # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

# Sélection de la broche d'entrée du signal numérique
Digital_PIN = 24
GPIO0.setup(Digital_PIN, GPIO0.IN, pull_up_down = GPIO0.PUD_OFF)

#####
# boucle de programme principale
#####
# Le programme lit les tensions en entrées et les transmet à la console.

try:
    while True:
        #Les valeurs de tension sont enregistrées
        analog = adc.readADCSingleEnded(adc_channel, gain, sps)

        # Envoi vers la console
        if GPIO0.input(Digital_PIN) == False:
            print "Tension analogique:", analog,"mV, ","Limite: pas encore atteinte"
        else:
            print "Tension analogique:", analog, "mV, ", "Limite: atteinte"
        print "-----"

        # Reset + Delay
        button_pressed = False
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO0.cleanup()

```

### Brochage Raspberry Pi:

#### Capteur

Signal numérique	= GPIO 24 [Pin 18 (RPi)]
+V	= 3,3V [Pin 1 (RPi)]
GND	= Masse [Pin 06 (RPi)]
Signal analogique	= Analog 0 [Pin A0 (ADS1115 - KY-053)]

#### ADS1115 - KY-053:

VDD	= 3,3V	[Pin 01]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]

## KY-028 Module capteur de t° CTN

A0 = s.o. [Sensor: Signal] analogique

**Exemple de programme à télécharger**[RPi\\_AnalogSensor.zip](#)

Commande pour lancer le programme:

```
sudo python RPi_AnalogSensor.py
```

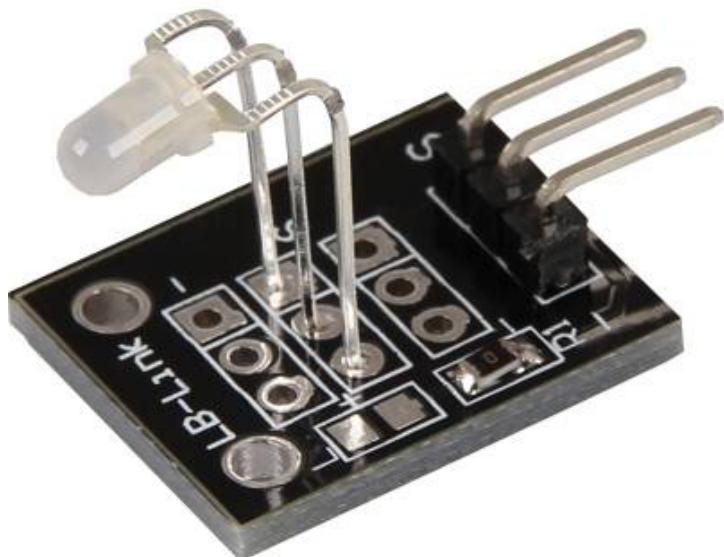
KY-029 Module led bicolore rouge/vert

## KY-029 Module led bicolore rouge/vert

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo



### Données techniques / Description sommaire

Module composé d'une LED 3mm bicolore rouge-verte, à cathode commune. Une résistance de limitation est nécessaire pour alimenter ce module.

**Vf [typ]= 2,0-2,5V**

**If= 20mA**

#### Résistance de limitation:

**Rf (3,3V) [Vert]= 120Ω**

KY-029 Module led bicolore rouge/vert

**R<sub>f</sub> (3,3V) [Rouge] = 120Ω**

[Valeurs calculées lors de l'utilisation avec des microcontrôleurs ARM de base tel que la Raspberry Pi]

**R<sub>f</sub> (5V) [Vert] = 220Ω**

**R<sub>f</sub> (5V) [Rouge] = 220Ω**

[Valeurs calculées lors de l'utilisation avec des microcontrôleurs Atmel Atmega tel que Arduino]

## Brochage

---



## Exemple de code pour Arduino

---

### Exemple de code ON/OFF

Cet exemple de code fait s'allumer alternativement la LED rouge et la LED verte toutes les 3 secondes.

```

int Led_Rouge = 10;
int Led_Verte = 11;

void setup ()
{
    // Initialisation des broches de sortie pour les LEDS
    pinMode (Led_Rouge, OUTPUT);
    pinMode (Led_Verte, OUTPUT);
}

void loop () //Boucle de programme principale
{
    digitalWrite (Led_Rouge, HIGH); // la Led s'allume
    digitalWrite (Led_Verte, LOW); // la Led est éteinte
    delay (3000); // Délai de 3 secondes

    digitalWrite (Led_Rouge, LOW); // La Led s'éteint
    digitalWrite (Led_Verte, HIGH); // La Led s'allume
    delay (3000); // Délai de 3 secondes supplémentaires pendant lequel les LEDS sont commutées
}

```

KY-029 Module led bicolore rouge/vert

#### Téléchargement de l'exemple de code ON/OFF:

[KY-029\\_LED\\_ON-OFF.zip](#)

#### Exemple de code en PWM

Un signal PWM (modulation de largeur d'impulsion) permet de faire varier la luminosité d'une LED. Le signal fait varier le temps pendant lequel il est à l'état haut et celui pendant lequel il est à l'état bas, ce qui fait varier la tension moyenne d'alimentation de la LED. La persistance rétinienne de l'oeil fait que nous visualisons cela comme un changement de luminosité.

Plusieurs LEDS sont intégrées dans ce module et la superposition de différents niveaux de luminosité de ces LEDS permet d'obtenir différentes couleurs. Ceci est illustré dans l'exemple de code suivant.

```

int Led_Rouge = 10;
int Led_Verte = 11;

int val;

void setup () {
    // Initialisation des broches de sortie pour les LEDS
    pinMode (Led_Rouge, OUTPUT);
    pinMode (Led_Verte, OUTPUT);
}
void loop () {
    // Dans une boucle For, différentes valeurs PWM sont envoyées aux 2 LEDS
    for (val = 255; val> 0; val--)
    {
        analogWrite (Led_Verte, val);
        analogWrite (Led_Rouge, 255-val);
        delay (15);
    }
    // Dans cette seconde boucle For, les valeurs sont inversées
    for (val = 0; val <255; val++)
    {
        analogWrite (Led_Verte, val);
        analogWrite (Led_Rouge, 255-val);
        delay (15);
    }
}

```

#### Exemple de programme à télécharger:

[KY-029\\_PWM.zip](#)

#### Affectation des broches Arduino:

LED Verte	= [Pin 10]
LED Rouge	= [Pin 11]
Sensor GND	= [Pin GND]

#### Exemple de code pour Raspberry Pi

#### Exemple de code ON/OFF

## KY-029 Module led bicolore rouge/vert

Cet exemple montre comment les LEDS intégrées sont alternativement allumées puis éteintes.

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Initialisation des broches de sortie pour les LEDS
LED_ROUGE = 5
LED_VERTE = 4
GPIO.setup(LED_ROUGE, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_VERTE, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale
try:
    while True:
        print("LED ROUGE allumée 3 secondes")
        GPIO.output(LED_ROUGE,GPIO.HIGH) #la Led s'allume
        GPIO.output(LED_VERTE,GPIO.LOW) #la LED commute
        time.sleep(3) # Délai de 3 secondes
        print("LED VERTE allumée 3 secondes")
        GPIO.output(LED_ROUGE,GPIO.LOW) #la LED commute
        GPIO.output(LED_VERTE,GPIO.HIGH) #la Led s'allume
        time.sleep(3) #Délai de 3 secondes

    # remise en place de tous les GPIO en entrées
    except KeyboardInterrupt:
        GPIO.cleanup()
```

### Téléchargement de l'exemple de code ON/OFF

[KY029\\_RPI\\_ON-OFF.zip](#)

Pour débuter avec la commande:

```
sudo python KY029_RPI_ON-OFF.py
```

### Exemple de code PWM

L'utilisation de signaux PWM (modulation de largeur d'amplitude) permet de faire varier la luminosité d'une LED. La LED est alimentée par des impulsions, le ratio entre la durée des impulsions et la durée au repos correspondant à une luminosité relative. En raison de la persistance rétinienne, l'oeil humain assimile ces changements d'alimentation de la LED à une variation de luminosité.

Vous trouverez plus d'informations sur [Plusieurs LEDS sont intégrées dans ce module et la création de couleurs différentes est produite par la superposition de différents niveaux de luminosité. Ceci est illustré dans l'exemple de code suivant.](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n>Main.ApprendrePWM</a></p>
</div>
<div data-bbox=)

```
# Les modules nécessaires sont importés et mis en place
import random, time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
```

## KY-029 Module led bicolore rouge/vert

```

# Déclaration des broches de sortie sur lesquelles les LEDS sont raccordées
LED_Rouge = 5
LED_Verte = 4

# Configuration des broches en sortie
GPIO.setup(LED_Rouge, GPIO.OUT)
GPIO.setup(LED_Verte, GPIO.OUT)

Freq = 100 #Hz

# Les couleurs respectives sont initialisées
ROUGE = GPIO.PWM(LED_Rouge, Freq)
VERTE = GPIO.PWM(LED_Verte, Freq)
ROUGE.start(0)
VERTE.start(0)

# Cette fonction génère la couleur réelle
# L'intensité de la couleur peut être modifiée grâce à la variable Couleur
# Après réglage de la couleur, la durée d'allumage est définie par 'time.sleep'

def LED_Couleur(Rouge, Verte, pause):
    ROUGE.ChangeDutyCycle(Rouge)
    Verte.ChangeDutyCycle(Verte)
    time.sleep(pause)

    ROUGE.ChangeDutyCycle(0)
    Verte.ChangeDutyCycle(0)

print "LED-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale:
# Cette boucle doit faire varier l'intensité de chaque couleur de 0 à 100% en utilisant une boucle for
# Les mélanges des différentes luminosités permettent de créer un gradients de couleurs différentes.
try:
    while True:
        for x in range(0,2):
            for y in range(0,2):
                print (x,y)
                for i in range(0,101):
                    LED_Couleur((x*i),(y*i),.02)

# remise en place de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()

```

### Exemple de programme à télécharger:

[KY029\\_RPI\\_PWM.zip](#)

Commande pour lancer le programme:

```
sudo python KY029_RPI_PWM.py
```

### Brochage Raspberry Pi:

LED **Verte** = GPIO4 [Pin 16]

LED **Rouge** = GPIO5 [Pin 18]

Sensor GND = Masse [Pin 6]

KY-031 Capteur de vibrations/chocs

## KY-031 Capteur de vibrations/chocs

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

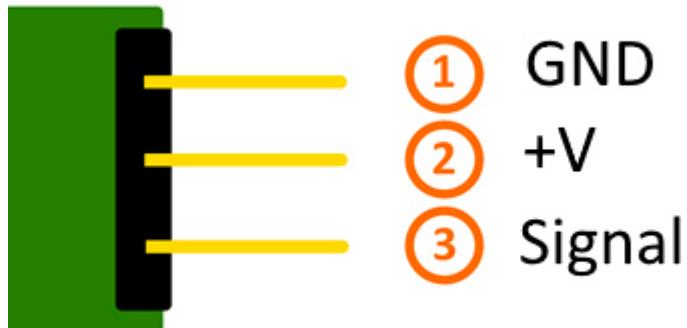
### Photo



### Données techniques / Description sommaire

En cas de choc ou de vibration sur le capteur, le contact de sortie se ferme.

## Brochage



## Exemple de code pour Arduino

Voici un exemple de programme qui fait allumer une LED lorsqu'un signal est détecté au niveau du capteur. On peut utiliser les modules à Leds KY-011, KY-016 ou KY-029.

```
int Led = 13; // Déclaration de la broche de sortie LED
int Sensor = 10; // Déclaration de la broche d'entrée du capteur
int val; // Variable temporaire

void setup ()
{
    pinMode (Led, OUTPUT); // Initialisation de la broche de sortie
    pinMode (Sensor, INPUT); // Initialisation de la broche du capteur
}

void loop ()
{
    val = digitalRead (Sensor); // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        digitalWrite (Led, LOW);
    }
    else
    {
        digitalWrite (Led, HIGH);
    }
}
```

### Affectation des broches Arduino:

LED +	= [Pin 13]
LED -	= [Pin GND]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]
Sensor -	= [Pin GND]

KY-031 Capteur de vibrations/chocs

### Exemple de programme à télécharger

[SensorTest\\_Arduino\\_withoutPullUP.zip](#)

## Exemple de code pour Raspberry Pi

Exemple de programmation en Python

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée. En outre la résistance de Pull-up est activée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Cette fonction de sortie est exécutée par détection du signal
def ausgabeFunktion(null):
    print("Signal détecté")

# Lors de la détection d'un signal (front descendant du signal) de la fonction de sortie est déclenchée
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=100)

# Boucle de programme principale
try:
    while True:
        time.sleep(1)

# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger

[SensorTest\\_RPi\\_withoutPullUP.zip](#)

Commande pour lancer le programme:

```
sudo python SensorTest_RPi_withoutPullUP.py
```

## KY-032 Capteur d'obstacles IR

## KY-032 Capteur d'obstacles IR

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	3
4 Exemple de code pour Arduino .....	3
5 Exemple de code pour Raspberry Pi .....	4

### Photo



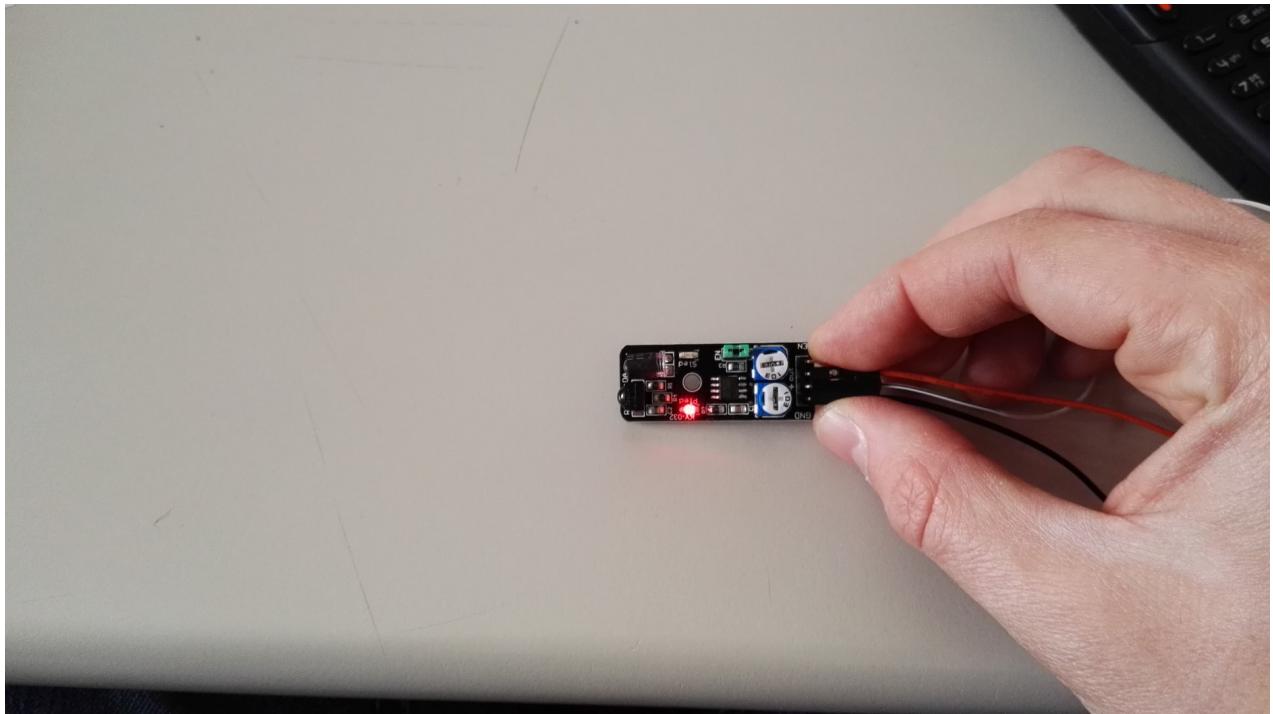
### Données techniques / Description sommaire

Ce capteur permet de détecter des obstacles, une diode infrarouge émet un signal et en cas de présence d'un obstacle, un récepteur infrarouge reçoit le signal réfléchi par cet obstacle. La portée de détection est réglable.

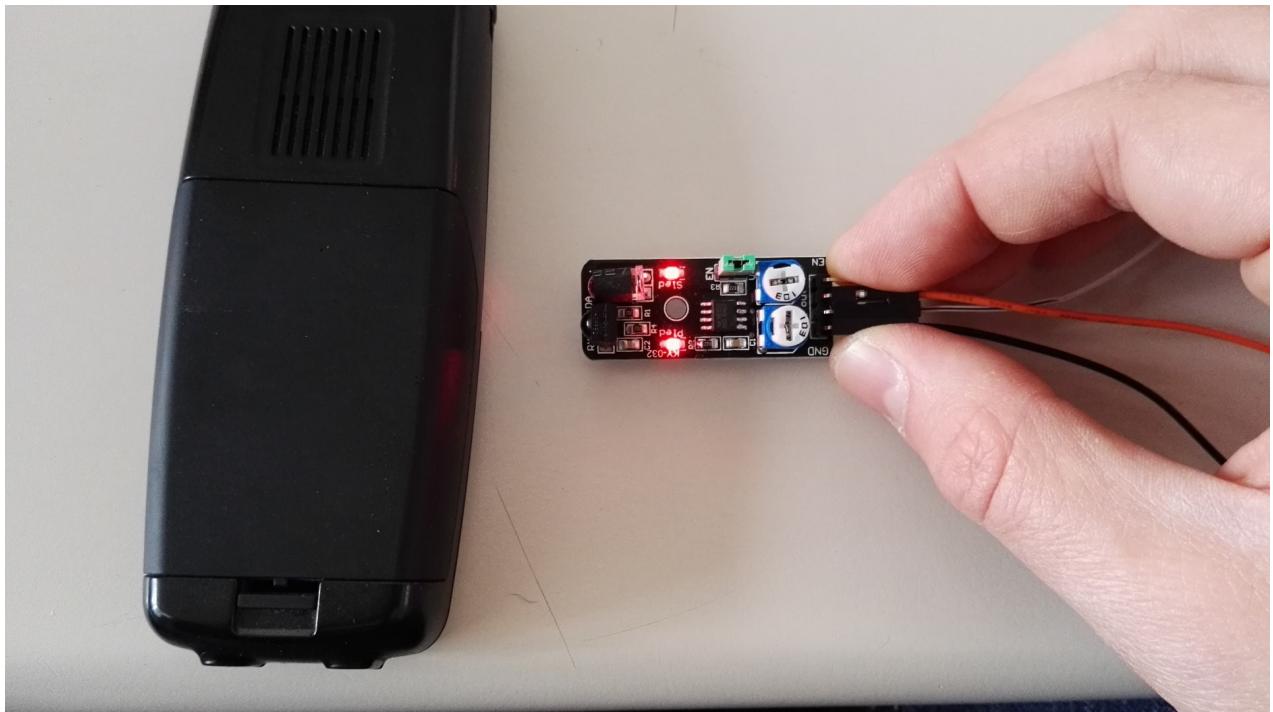
Ce module peut être utilisé pour construire un robot autonome détecteur d'obstacles.

**Cas 1:** pas d'obstacle devant le capteur [la LED est éteinte] [Signal capteur = état haut]

## KY-032 Capteur d'obstacles IR



**Cas 2:** le capteur détecte un obstacle [la LED est allumée] [Signal capteur = état bas]

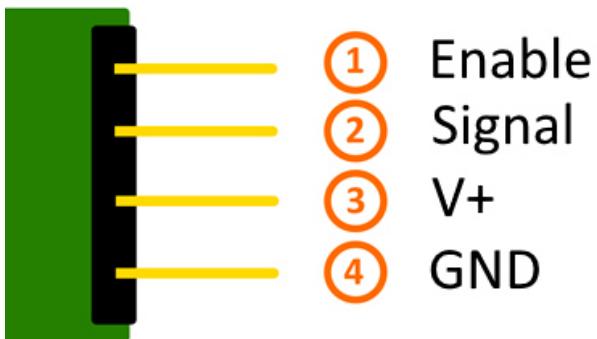


Possibilité d'activer ou désactiver la détection d'obstacle par le microcontrôleur à l'aide de la broche ENABLE. La détection est activée par défaut sur le capteur. Si vous souhaitez modifier cela, il faut enlever au préalable le cavalier vert pour pouvoir utiliser l'option ENABLE.

## KY-032 Capteur d'obstacles IR



### Brochage



### Exemple de code pour Arduino

Le programme lit l'état des broches du capteur et envoie l'information de détection (ou non) d'un obstacle dans la console.

```

int Sensor = 10; // Déclaration de la broche d'entrée du capteur

void setup ()
{
    Serial.begin(9600); // Sortie série à 9600 bauds
    pinMode (Sensor, INPUT) ; // Initialisation de la broche d'entrée du capteur
}

// Le programme lit l'état des broches du capteur et envoie l'information
// de détection (ou non) d'un obstacle dans la console.
void loop ()
{
    bool val = digitalRead (Sensor) ; // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        Serial.println("Pas d'obstacle");
    }
    else
    {
}

```

## KY-032 Capteur d'obstacles IR

```

        Serial.println("Obstacle detecte");
    }
    Serial.println("-----");
    delay(500); // pause de 500ms entre les mesures
}

```

### Affectation des broches Arduino:

Sensor Enable	= [N.C. (cavalier vert en place)]
Sensor Signal	= [Pin 10]
Sensor +V	= [Pin 5V]
Sensor GND	= [Pin GND]

### Exemple de programme à télécharger

[KY-032\\_HindernisDetektor.zip](#)

## Exemple de code pour Raspberry Pi

Le programme lit l'état des broches du capteur et envoie l'information de détection (ou non) d'un obstacle dans la console.

```

# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

# Pause avant l'envoi du résultat (en secondes)
delayTime = 0.5

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale
try:
    while True:
        if GPIO.input(GPIO_PIN) == True:
            print "Pas d'obstacle"
        else:
            print "Obstacle détecté"
        print "-----"

        # Reset + Delay
        time.sleep(delayTime)

# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()

```

### Brochage Raspberry Pi:

Enable = - [N.C. (cavalier vert en place)]

## KY-032 Capteur d'obstacles IR

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

**Exemple de programme à télécharger**[KY-032\\_RPi\\_ObstacleDetector.zip](#)

Commande pour lancer le programme:

```
sudo python KY-032_RPi_ObstacleDetector.py
```

## KY-033 Module suiveur de ligne

## KY-033 Module suiveur de ligne

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	3
4 Exemple de code pour Arduino .....	3
5 Exemple de code pour Raspberry Pi .....	4

### Photo



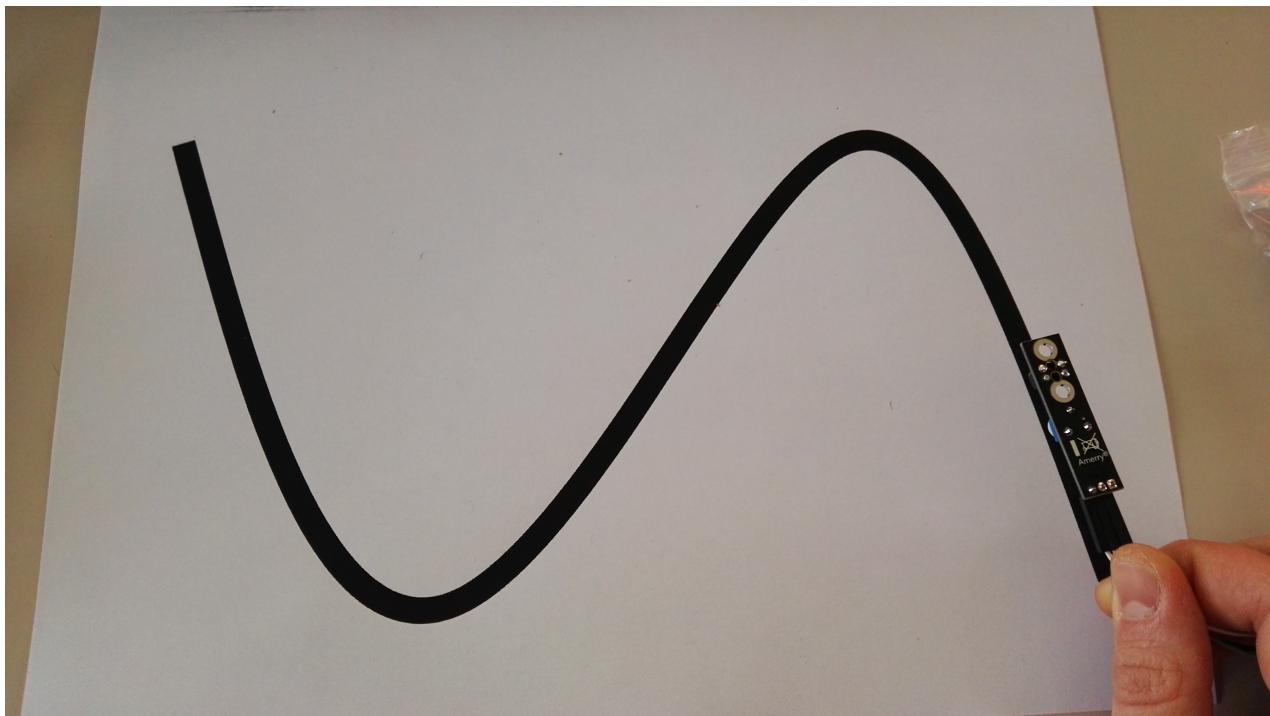
### Données techniques / Description sommaire

Ce module permet de suivre des lignes réfléchissant ou absorbant la lumière. Lors d'une détection, la sortie du module passe à l'état bas. La portée de détection est réglable.

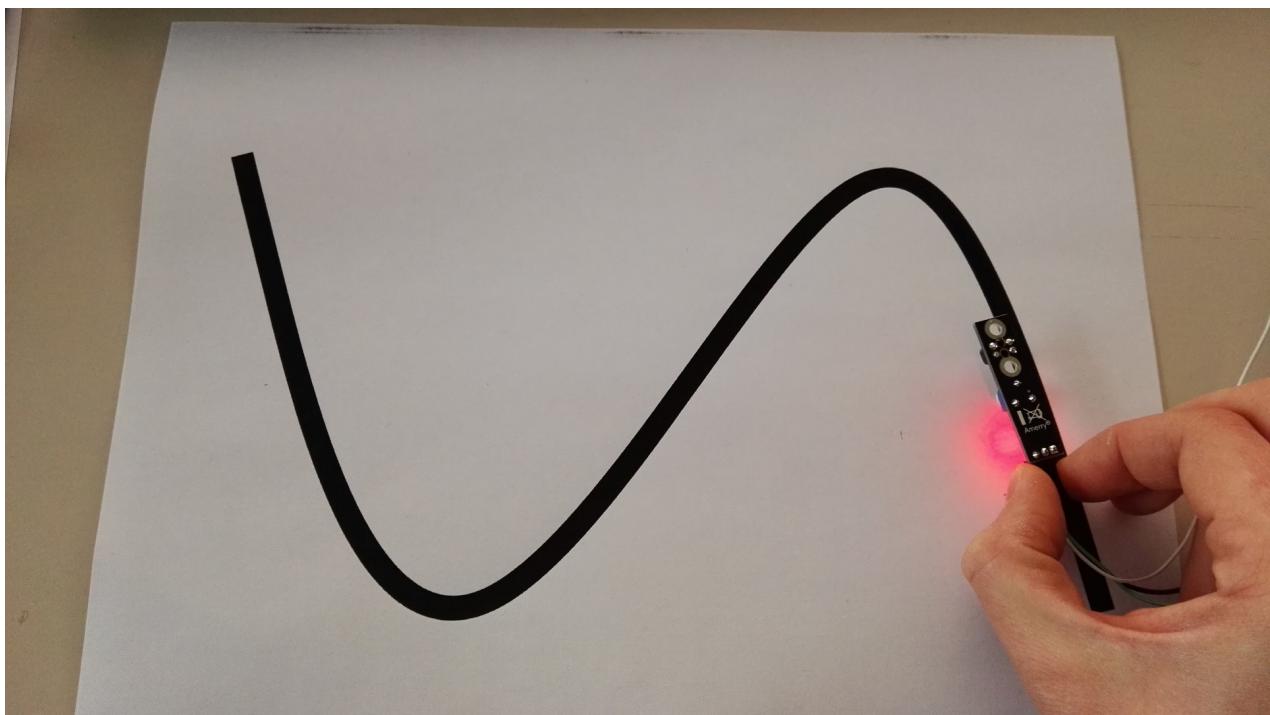
Ce module peut être utilisé pour construire un robot autonome suiveur de ligne.

**Cas 1:** le suiveur de ligne se trouve sur une ligne (surface non réfléchissante) [la LED du module est éteinte] [Signal capteur = état haut]

## KY-033 Module suiveur de ligne

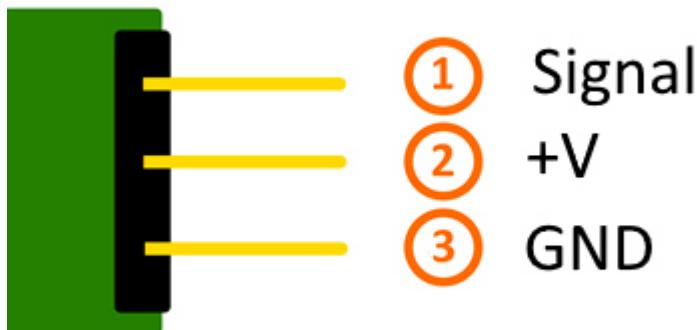


**Cas 2:** le suiveur de ligne n'est pas sur une ligne (surface réfléchissante) [la LED du module est allumée]  
[Signal capteur = état bas]



## KY-033 Module suiveur de ligne

### Brochage



### Exemple de code pour Arduino

Le programme lit l'état des broches du capteur et envoie l'information de suivi (ou non) d'une ligne dans la console.

```
int Sensor = 10; // Déclaration de la broche d'entrée du capteur

void setup ()
{
    Serial.begin(9600); // Sortie série à 9600 bauds
    pinMode (Sensor, INPUT) ; // Initialisation de la broche d'entrée du capteur
}

// Le programme lit l'état des broches du capteur et envoie
// l'information de suivi (ou non) d'une ligne dans la console.
void loop ()
{
    bool val = digitalRead (Sensor) ; // Lecture de la valeur du signal

    if (val == HIGH) // Si un signal est détecté, la diode s'allume
    {
        Serial.println("Le suiveur de ligne se trouve sur une ligne");
    }
    else
    {
        Serial.println("Le suiveur de ligne ne se trouve pas sur une ligne");
    }
    Serial.println("-----");
    delay(500); // pause de 500ms entre les mesures
}
```

#### Affectation des broches Arduino:

Sensor Signal = [Pin 10]  
 Sensor +V = [Pin 5V]  
 Sensor GND = [Pin GND]

#### Exemple de programme à télécharger

## KY-033 Module suiveur de ligne

[KY-033\\_TrackingSensor.zip](#)

### Exemple de code pour Raspberry Pi

Le programme lit l'état des broches du capteur et envoie l'information de suivi (ou non) d'une ligne dans la console.

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée du capteur est déclarée.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

# Pause avant l'envoi du résultat (en secondes)
delayTime = 0.5

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale
try:
    while True:
        if GPIO.input(GPIO_PIN) == True:
            print "Le suiveur se trouve sur une ligne"
        else:
            print "Le suiveur ne se trouve pas sur une ligne"
        print "-----"

        # Reset + Delay
        time.sleep(delayTime)

# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

#### Brochage Raspberry Pi:

Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

#### Exemple de programme à télécharger

[KY-033\\_RPi\\_Trackingsensor.zip](#)

Commande pour lancer le programme:

```
sudo python KY-033_RPi_Trackingsensor.py
```

KY-034 Module led flash (7 couleurs)

## KY-034 Module led flash (7 couleurs)

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	2

### Photo



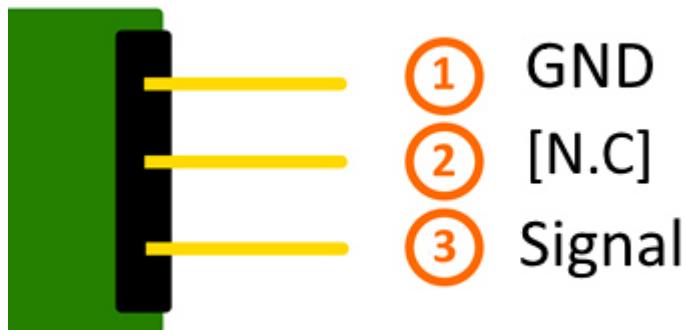
### Données techniques / Description sommaire

Ce module fait clignoter la Led avec une séquence automatique de changement de couleurs (7 couleurs).

Plage de tension: 3,3Vcc - 5Vcc

KY-034 Module led flash (7 couleurs)

## Brochage



## Exemple de code pour Arduino

Cet exemple de code montre comment faire clignoter la led avec un allumage pendant 4 secondes et une extinction pendant 2 secondes.

```

int Led = 13;

void setup ()
{
  pinMode (Led, OUTPUT); // Initialisation de la broche de sortie de la LED
}

void loop () //Boucle de programme principale
{
  digitalWrite (Led, HIGH); // la LED est allumé
  delay (4000); // délai de 4 secondes
  digitalWrite (Led, LOW); // la LED est éteinte
  delay (2000); // délai de 2 secondes
}

```

### Affectation des broches Arduino:

Sensor Signal = [Pin 13]  
 Sensor [N.C] =  
 Sensor GND = [Pin GND]

### Exemple de programme à télécharger:

[LedTestArduino\\_4On\\_2Off.zip](#)

## Exemple de code pour Raspberry Pi

Exemple de programme en Python

## KY-034 Module led flash (7 couleurs)

```
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# La broche d'entrée est déclarée. En outre la résistance de Pull-up est activée.
LED_PIN = 24
GPIO.setup(LED_PIN, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale
try:
    while True:
        print("la LED est allumée 4 secondes")
        GPIO.output(LED_PIN,GPIO.HIGH) #la LED est allumée
        time.sleep(4) #délai de 4 secondes
        print("la LED est éteinte 2 secondes")
        GPIO.output(LED_PIN,GPIO.LOW) #la LED est éteinte
        time.sleep(2) #délai de 2 secondes

# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Sensor Signal	=	GPIO24	[Pin 18]
Sensor [N.C]	=		
Sensor GND	=	Masse	[Pin 6]

### Exemple de programme à télécharger

[LedTest\\_RPi\\_4On\\_2Off.zip](#)

Commande pour lancer le programme:

```
sudo python LedTest_RPi_4On_2Off.py
```

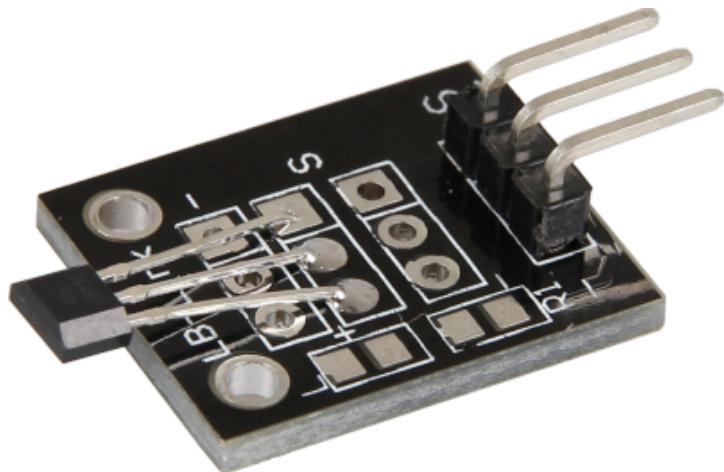
## KY-035 Capteur magnétique analogique

## KY-035 Capteur magnétique analogique

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo



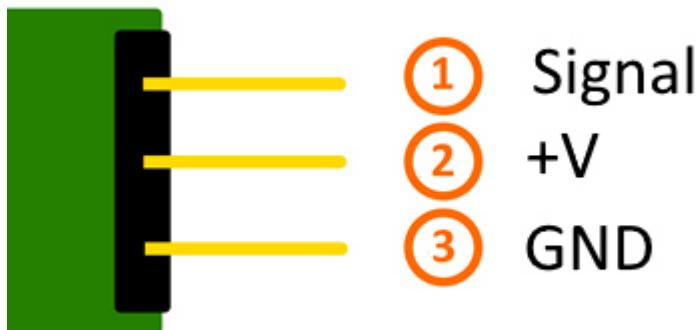
### Données techniques / Description sommaire

Chipset: AH49E

La capteur délivre sur sa sortie une tension analogique proportionnelle à l'intensité du champ magnétique.

## KY-035 Capteur magnétique analogique

### Brochage



### Exemple de code pour Arduino

Le programme mesure la valeur de tension au niveau du capteur, calcule la résistance à partir de cette valeur et de la résistance série connue et renvoie les résultats à la sortie série.

```
int sensorPin = A5; // Déclaration de la broche d'entrée

// Sortie série à 9600 bauds
void setup()
{
    Serial.begin(9600);
}

// Le programme mesure la valeur de tension au niveau du capteur,
// calcule la résistance à partir de cette valeur et de la résistance
// série connue et renvoie les résultats à la sortie série.
void loop()

{
    // Mesure de la tension du capteur...
    int rawValue = analogRead(sensorPin);
    float voltage = rawValue * (5.0/1023) * 1000;

    float resistance = 10000 * ( voltage / ( 5000.0 - voltage) );

    // ... et envoi vers le port série
    Serial.print("Tension:"); Serial.print(voltage); Serial.print("mV");
    Serial.print(", Résistance:"); Serial.print(resistance); Serial.println("Ohm");
    Serial.println("-----");

    delay(500);
}
```

#### Affectation des broches Arduino:

Sensor GND	= [Pin GND]
Sensor +V	= [Pin 5V]
Sensor Signal	= [Pin A5]

#### Exemple de programme à télécharger

## KY-035 Capteur magnétique analogique

### Exemple de programme à télécharger

[Single\\_Analog\\_Sensor.zip](#)

## Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

Vous trouverez de plus amples informations à ce sujet dans la description du [module KY-053](#).

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il calcule la résistance de la LDR et la transmet à la console.

```
# coding=utf-8
#!/usr/bin/python

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Single Analog Sensor - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os, math
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

## KY-035 Capteur magnétique analogique

```
# Les variables utilisées sont initialisées
delayTime = 0.2
voltageMax = 3300 # valeur de tension maxi possible sur l'ADC

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde
# sps = 32 # 32 échantillons par seconde
# sps = 64 # 64 échantillons par seconde
# sps = 128 # 128 échantillons par seconde
sps = 250 # 250 échantillons par seconde
# sps = 475 # 475 échantillons par seconde
# sps = 860 # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel = 0 # Channel 0
# adc_channel = 1 # Channel 1
# adc_channel = 2 # Channel 2
# adc_channel = 3 # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

#####
# boucle de programme principale
#####
#Le programme mesure la tension à l'aide du convertisseur ADS1115.
# Il calcule la résistance et les transmet à la console.

try:
    while True:
        #Lecture de la valeur de la tension, ...
        voltage = adc.readADCSingleEnded(adc_channel, gain, sps)

        # ... calcul de la résistance...
        resistance = 10000 * voltage/(voltageMax - voltage)

        # ... et envoi des 2 valeurs dans la console
        print "Tension:", voltage,"mV, Résistance:", resistance,"Ω"
        print "-----"

        # Delay
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Capteur

## KY-035 Capteur magnétique analogique

GND	= GND	[Pin 06 (RPi)]
+V	= 3,3V	[Pin 01 (RPi)]
Signal analogique	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

## ADS1115 - KY-053:

VDD	= 3,3V	[Pin 17]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]
A0	= s.o.	[Sensor: Signal analogique]

**Exemple de programme à télécharger**[RPi\\_Single\\_Analog\\_Sensor.zip](#)

Commande pour lancer le programme:

```
sudo python RPi_Single_Analog_Sensor.py
```

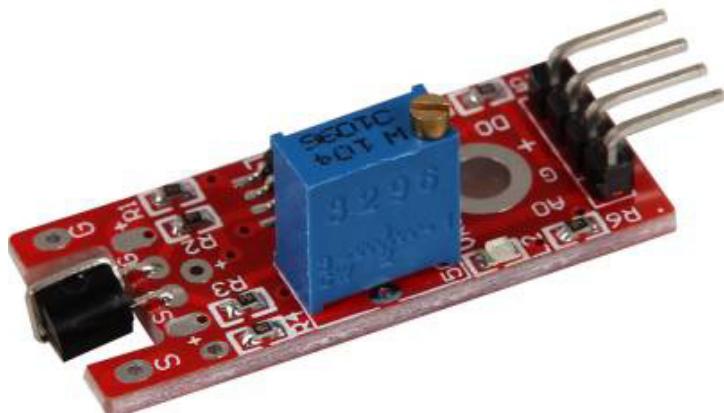
KY-036 Capteur tactile

## KY-036 Capteur tactile

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Fonctionnement du capteur .....	2
5 Exemple de code pour Arduino .....	3
6 Exemple de code pour Raspberry Pi .....	4

### Photo



### Données techniques / Description sommaire

Ce module envoie un signal si le bout métallique devant le capteur est touché. La sensibilité du capteur est réglable.

**Sortie numérique:** un signal est présent si un contact est détecté

**Sortie analogique:** mesure directe du capteur

**LED1:** indique que le module est alimenté

**LED2:** indique que le capteur est touché

## Brochage

---



- ① Signal numérique
- ② +V
- ③ GND
- ④ Signal analogique

## Fonctionnement du capteur

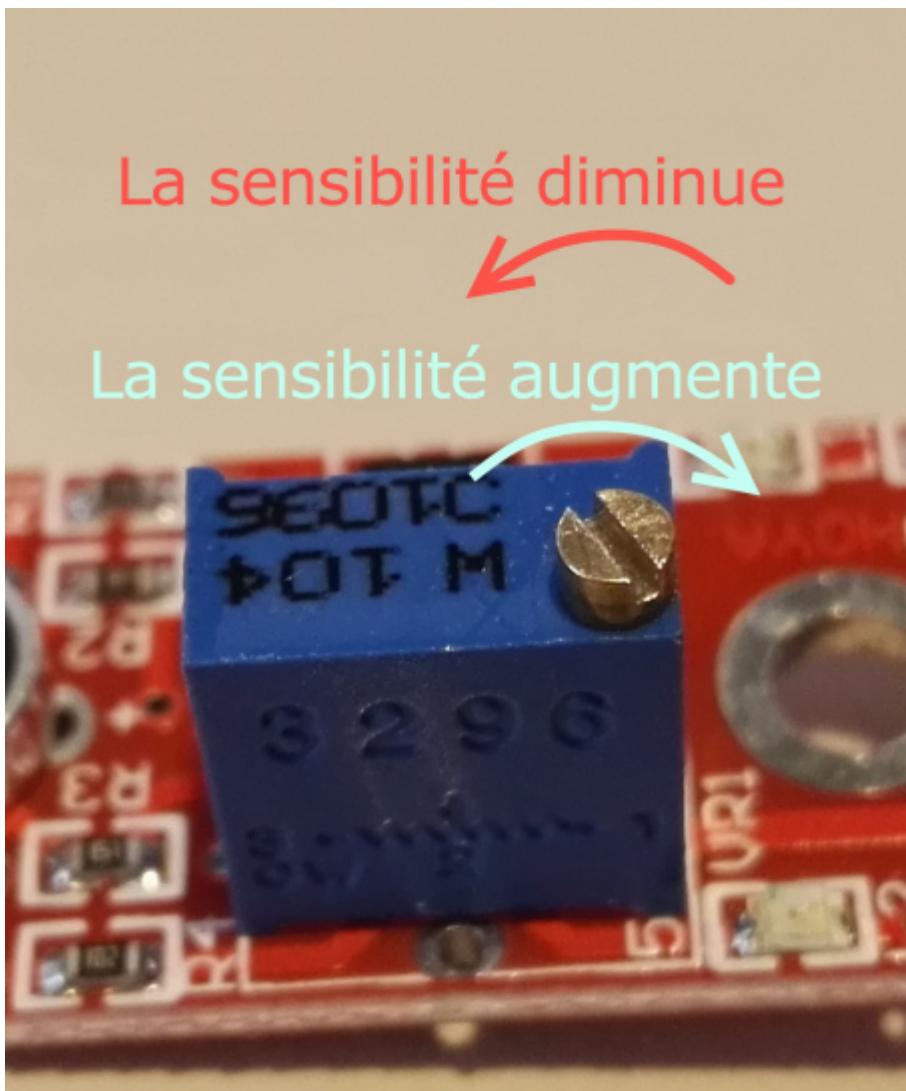
---

Ce module est composé de trois éléments fonctionnels. Le capteur situé à l'avant du module effectue la mesure, le signal analogique est ensuite envoyé sur l'amplificateur. Celui-ci amplifie le signal en fonction du gain déterminé par le potentiomètre et envoie le signal à la sortie analogique du module.

Il convient de noter que le signal est inversé: plus la valeur mesurée par le capteur est haute, plus la tension de sortie est faible.

La troisième partie est composée d'un comparateur qui commute la sortie numérique et la diode lorsque le signal tombe en dessous d'une certaine valeur. La sensibilité peut être ajustée au moyen du potentiomètre comme décrit ci-dessous:

KY-036 Capteur tactile



Ce type de capteur ne délivre pas des valeurs absolues (par exemple, la température mesurée avec précision en ° C ou de la force du champ magnétique en mT), mais des valeurs relatives. On définit une valeur limite par rapport à une valeur normale donnée et le module émet un signal si cette limite est dépassée.

Ce fonctionnement est idéal pour la surveillance de la température (KY-028), les détecteurs de proximité (KY-024, KY 025, KY-036), la surveillance des alarmes (KY-037, KY-038) ou le détecteur de flamme (KY-026).

## Exemple de code pour Arduino

Le programme lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```
// Déclaration et initialisation des broches d'entrées
int Analog_Eingang = A0; // Entrée analogique
int Digital_Eingang = 3; // Entrée digitale
```

## KY-036 Capteur tactile

```

void setup ()
{
    pinMode (Analog_Eingang, INPUT);
    pinMode (Digital_Eingang, INPUT);

    Serial.begin (9600); // Sortie série à 9600 bauds
}

// Le programme lit les valeurs des broches d'entrée et les envoie à la sortie série
void loop ()
{
    float Analog;
    int Digital;

    //Les valeurs sont lues, sont converties en tension...
    Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
    Digital = digitalRead (Digital_Eingang);

    //... et envoyées à la sortie série.
    Serial.print ("Tension analogique:"); Serial.print (Analog, 4); Serial.print ("V, ");
    Serial.print ("Limite:");

    if(Digital==1)
    {
        Serial.println (" atteinte");
    }
    else
    {
        Serial.println (" pas encore atteinte");
    }
    Serial.println ("-----");
    delay (200);
}

```

### Affectation des broches Arduino:

digitales Signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analoges Signal	= [Pin A0]

### Exemple de programme à télécharger

[Ard\\_Analoger\\_Sensor.zip](#)

## Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

## KY-036 Capteur tactile

Vous trouverez de plus amples informations à ce sujet dans la description du module KY-053.

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```

#
#!/usr/bin/python
# coding=utf-8

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Analog Sensor + ADS1115 ADC - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde

```

## KY-036 Capteur tactile

```

# sps = 16    # 16 échantillons par seconde
# sps = 32    # 32 échantillons par seconde
sps = 64    # 64 échantillons par seconde
# sps = 128   # 128 échantillons par seconde
# sps = 250   # 250 échantillons par seconde
# sps = 475   # 475 échantillons par seconde
# sps = 860   # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel = 0    # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

# Sélection de la broche d'entrée du signal numérique
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

#####
# boucle de programme principale
# #####
# Le programme lit les tensions en entrées et les transmet à la console.

try:
    while True:
        #Les valeurs de tension sont enregistrées
        analog = adc.readADCSingleEnded(adc_channel, gain, sps)

        # Envoi vers la console
        if GPIO.input(Digital_PIN) == False:
            print "Tension analogique:", analog,"mV, ", "Limite: pas encore atteinte"
        else:
            print "Tension analogique:", analog, "mV, ", "Limite: atteinte"
        print "-----"

        # Reset + Delay
        button_pressed = False
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()

```

### Brochage Raspberry Pi:

Capteur

Signal numérique	= GPIO 24	[Pin 18 (RPi)]
+V	= 3,3V	[Pin 1 (RPi)]
GND	= Masse	[Pin 06 (RPi)]
Signal analogique	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	= 3,3V	[Pin 01]
GND	= Masse	[Pin 09]

## KY-036 Capteur tactile

SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]
A0	= s.o.	[Capteur: signal analogique]

**Exemple de programme à télécharger**[RPi\\_AnalogSensor.zip](#)

Commande pour lancer le programme:

```
sudo python RPi_AnalogSensor.py
```

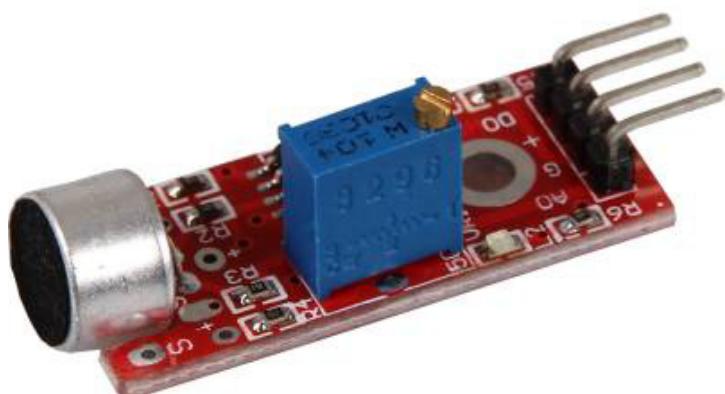
KY-037 Capteur sonore

## KY-037 Capteur sonore

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Fonctionnement du capteur .....	2
5 Exemple de code pour Arduino .....	3
6 Exemple de code pour Raspberry Pi .....	4

### Photo



### Données techniques / Description sommaire

Module de mesure du bruit à haute sensibilité.

**Sortie numérique:** signal présent si la limite fixée par le potentiomètre est atteinte

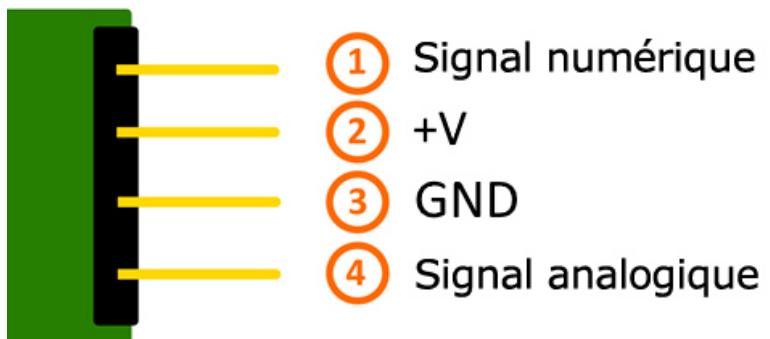
**Sortie analogique:** mesure directe de la tension à la sortie du micro

**LED1:** indique que le capteur est alimenté en tension

**LED2:** indique qu'un son est détecté

## Brochage

---



## Fonctionnement du capteur

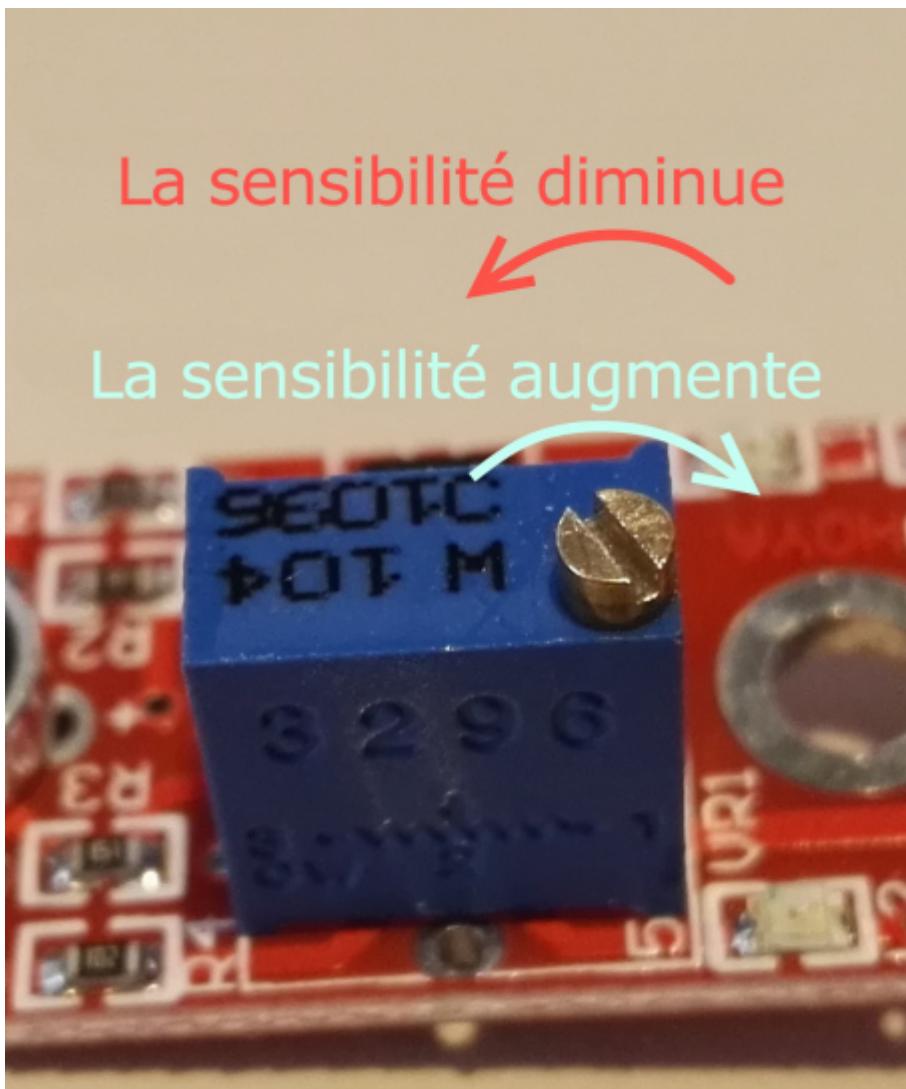
---

Ce module est composé de trois éléments fonctionnels. Le capteur situé à l'avant du module effectue la mesure, le signal analogique est ensuite envoyé sur l'amplificateur. Celui-ci amplifie le signal en fonction du gain déterminé par le potentiomètre et envoie le signal à la sortie analogique du module.

Il convient de noter que le signal est inversé: plus la valeur mesurée par le capteur est haute, plus la tension de sortie est faible.

La troisième partie est composée d'un comparateur qui commute la sortie numérique et la diode lorsque le signal tombe en dessous d'une certaine valeur. La sensibilité peut être ajustée au moyen du potentiomètre comme décrit ci-dessous:

KY-037 Capteur sonore



Ce type de capteur ne délivre pas des valeurs absolues (par exemple, la température mesurée avec précision en ° C ou de la force du champ magnétique en mT), mais des valeurs relatives. On définit une valeur limite par rapport à une valeur normale donnée et le module émet un signal si cette limite est dépassée.

Ce fonctionnement est idéal pour la surveillance de la température (KY-028), les détecteurs de proximité (KY-024, KY-025, KY-036), la surveillance des alarmes (KY-037, KY-038) ou le détecteur de flamme (KY-026).

## Exemple de code pour Arduino

Le programme lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```
// Déclaration et initialisation des broches d'entrées
int Analog_Eingang = A0; // Entrée analogique
int Digital_Eingang = 3; // Entrée digitale
```

## KY-037 Capteur sonore

```

void setup ()
{
    pinMode (Analog_Eingang, INPUT);
    pinMode (Digital_Eingang, INPUT);

    Serial.begin (9600); // Sortie série à 9600 bauds
}

// Le programme lit les valeurs des broches d'entrée et les envoie à la sortie série
void loop ()
{
    float Analog;
    int Digital;

    //Les valeurs sont lues, sont converties en tension...
    Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
    Digital = digitalRead (Digital_Eingang);

    //... et envoyées à la sortie série.
    Serial.print ("Tension analogique:"); Serial.print (Analog, 4); Serial.print ("V, ");
    Serial.print ("Limite:");

    if(Digital==1)
    {
        Serial.println (" atteinte");
    }
    else
    {
        Serial.println (" pas encore atteinte");
    }
    Serial.println ("-----");
    delay (200);
}

```

### Affectation des broches Arduino:

Signal numérique = [Pin 3]  
 +V = [Pin 5V]  
 GND = [Pin GND]  
 Signal analogique = [Pin A0]

### Exemple de programme à télécharger

[Ard\\_Analoger\\_Sensor.zip](#)

## Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

## KY-037 Capteur sonore

Vous trouverez de plus amples informations à ce sujet dans la description du module KY-053.

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```

#
#!/usr/bin/python
# coding=utf-8

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Analog Sensor + ADS1115 ADC - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde

```

## KY-037 Capteur sonore

```

# sps = 32    # 32 échantillons par seconde
sps = 64    # 64 échantillons par seconde
# sps = 128   # 128 échantillons par seconde
# sps = 250   # 250 échantillons par seconde
# sps = 475   # 475 échantillons par seconde
# sps = 860   # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel = 0    # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

# Sélection de la broche d'entrée du signal numérique
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

#####
# boucle de programme principale
#####
# Le programme lit les tensions en entrées et les transmet à la console.

try:
    while True:
        #Les valeurs de tension sont enregistrées
        analog = adc.readADCSingleEnded(adc_channel, gain, sps)

        # Envoi vers la console
        if GPIO.input(Digital_PIN) == False:
            print "Tension analogique:", analog,"mV, ","Limite: pas encore atteinte"
        else:
            print "Tension analogique:", analog, "mV, ", "Limite: atteinte"
        print "-----"

        # Reset + Delay
        button_pressed = False
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()

```

### **Brochage Raspberry Pi:**

#### Capteur

Signal numérique	= GPIO 24	[Pin 18 (RPi)]
+V	= 3,3V	[Pin 1 (RPi)]
GND	= Masse	[Pin 06 (RPi)]
Signal analogique	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

#### ADS1115 - KY-053:

VDD	= 3,3V	[Pin 01]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]

## KY-037 Capteur sonore

SDA = GPIO02 / SDA [Pin 03]  
A0 = s.o. [Capteur: signal analogique]

**Exemple de programme à télécharger**

[RPi\\_AnalogSensor.zip](#)

Commande pour lancer le programme:

```
sudo python RPi_AnalogSensor.py
```

KY-038 Capteur sonore

## KY-038 Capteur sonore

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Fonctionnement du capteur .....	2
5 Exemple de code pour Arduino .....	3
6 Exemple de code pour Raspberry Pi .....	4

### Photo



### Données techniques / Description sommaire

**Sortie numérique:** signal présent si la limite fixée par le potentiomètre est atteinte

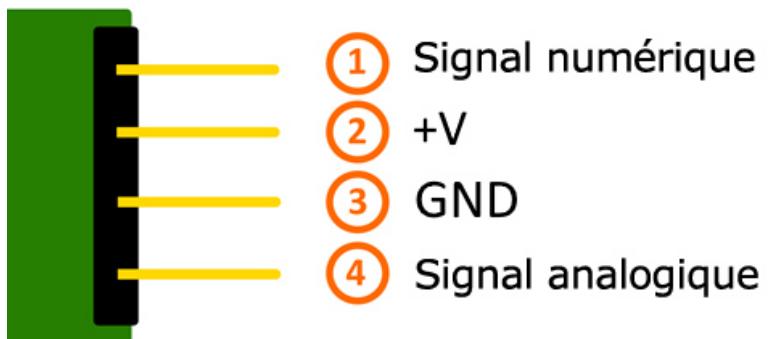
**Sortie analogique:** mesure directe de la tension à la sortie du micro

**LED1:** indique que le capteur est alimenté en tension

**LED2:** indique qu'un son est détecté

## Brochage

---



## Fonctionnement du capteur

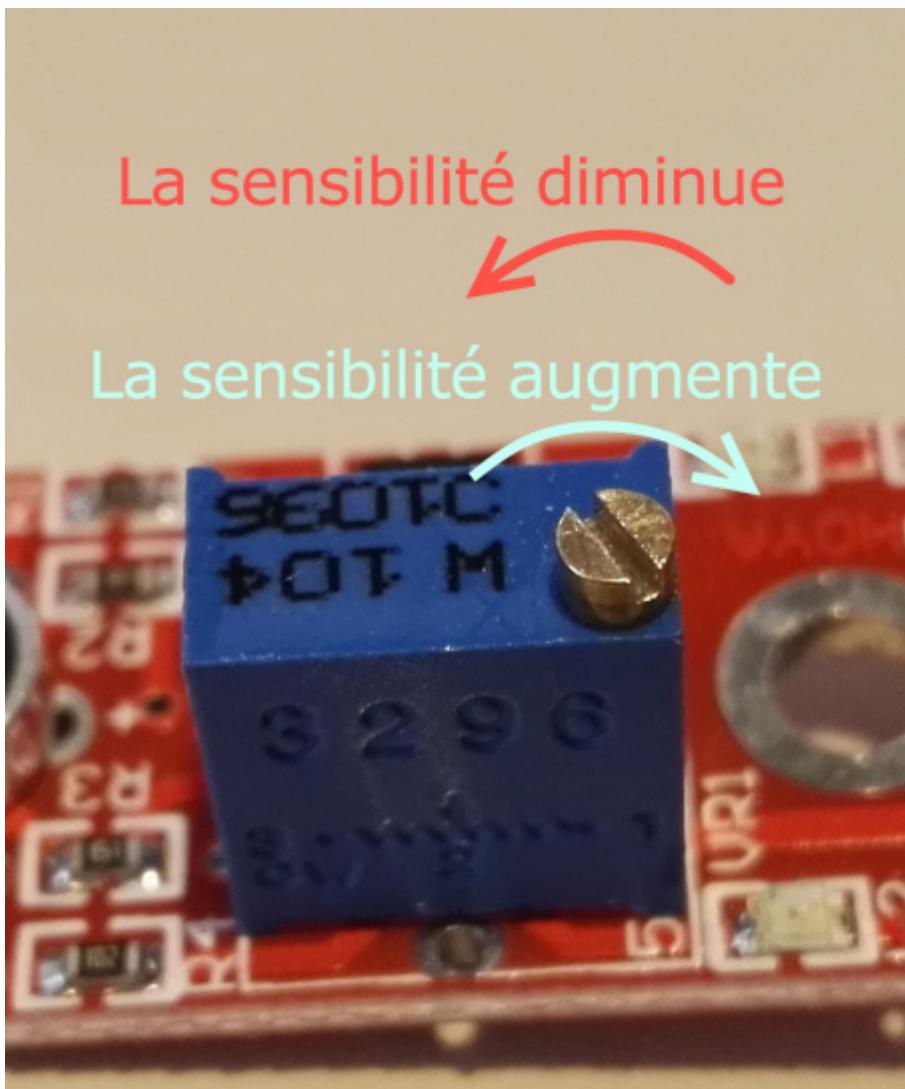
---

Ce module est composé de trois éléments fonctionnels. Le capteur situé à l'avant du module effectue la mesure, le signal analogique est ensuite envoyé sur l'amplificateur. Celui-ci amplifie le signal en fonction du gain déterminé par le potentiomètre et envoie le signal à la sortie analogique du module.

Il convient de noter que le signal est inversé: plus la valeur mesurée par le capteur est haute, plus la tension de sortie est faible.

La troisième partie est composée d'un comparateur qui commute la sortie numérique et la diode lorsque le signal tombe en dessous d'une certaine valeur. La sensibilité peut être ajustée au moyen du potentiomètre comme décrit ci-dessous:

KY-038 Capteur sonore



Ce type de capteur ne délivre pas des valeurs absolues (par exemple, la température mesurée avec précision en ° C ou de la force du champ magnétique en mT), mais des valeurs relatives. On définit une valeur limite par rapport à une valeur normale donnée et le module émet un signal si cette limite est dépassée.

Ce fonctionnement est idéal pour la surveillance de la température (KY-028), les détecteurs de proximité (KY-024, KY-025, KY-036), la surveillance des alarmes (KY-037, KY-038) ou le détecteur de flamme (KY-026).

## Exemple de code pour Arduino

Le programme lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```
// Déclaration et initialisation des broches d'entrées
int Analog_Eingang = A0; // Entrée analogique
int Digital_Eingang = 3; // Entrée digitale
```

## KY-038 Capteur sonore

```

void setup ()
{
    pinMode (Analog_Eingang, INPUT);
    pinMode (Digital_Eingang, INPUT);

    Serial.begin (9600); // Sortie série à 9600 bauds
}

// Le programme lit les valeurs des broches d'entrée et les envoie à la sortie série
void loop ()
{
    float Analog;
    int Digital;

    //Les valeurs sont lues, sont converties en tension...
    Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
    Digital = digitalRead (Digital_Eingang);

    //... et envoyées à la sortie série.
    Serial.print ("Tension analogique:"); Serial.print (Analog, 4); Serial.print ("V, ");
    Serial.print ("Limite:");

    if(Digital==1)
    {
        Serial.println (" atteinte");
    }
    else
    {
        Serial.println (" pas encore atteinte");
    }
    Serial.println ("-----");
    delay (200);
}

```

### Affectation des broches Arduino:

Signal numérique = [Pin 3]  
 +V = [Pin 5V]  
 GND = [Pin GND]  
 Signal analogique = [Pin 0]

### Exemple de programme à télécharger

[Ard\\_Analoger\\_Sensor.zip](#)

## Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

## KY-038 Capteur sonore

Vous trouverez de plus amples informations à ce sujet dans la description du module KY-053.

**!!Attention !! Capteur analogique !! Attention !!**

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir si le seuil a été atteint ou pas.

```

#
#!/usr/bin/python
# coding=utf-8

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Analog Sensor + ADS1115 ADC - Raspberry Pi Python Code Example
###

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde

```

## KY-038 Capteur sonore

```

# sps = 32    # 32 échantillons par seconde
sps = 64    # 64 échantillons par seconde
# sps = 128   # 128 échantillons par seconde
# sps = 250   # 250 échantillons par seconde
# sps = 475   # 475 échantillons par seconde
# sps = 860   # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel = 0    # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

# Sélection de la broche d'entrée du signal numérique
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

#####
# boucle de programme principale
#####
# Le programme lit les tensions en entrées et les transmet à la console.

try:
    while True:
        #Les valeurs de tension sont enregistrées
        analog = adc.readADCSingleEnded(adc_channel, gain, sps)

        # Envoi vers la console
        if GPIO.input(Digital_PIN) == False:
            print "Tension analogique:", analog,"mV, ","Limite: pas encore atteinte"
        else:
            print "Tension analogique:", analog, "mV, ", "Limite: atteinte"
        print "-----"

        # Reset + Delay
        button_pressed = False
        time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()

```

### **Brochage Raspberry Pi:**

#### Capteur

Signal numérique	= GPIO 24	[Pin 18 (RPi)]
+V	= 3,3V	[Pin 1 (RPi)]
GND	= Masse	[Pin 06 (RPi)]
Signal analogique	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

#### ADS1115 - KY-053:

VDD	= 3,3V	[Pin 01]
GND	= Masse	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]

## KY-038 Capteur sonore

SDA = GPIO02 / SDA [Pin 03]  
A0 = s.o. [Capteur: signal analogique]

**Exemple de programme à télécharger**

[RPi\\_AnalogSensor.zip](#)

Commande pour lancer le programme:

```
sudo python RPi_AnalogSensor.py
```

## KY-039 Capteur de pulsations cardiaques

## KY-039 Capteur de pulsations cardiaques

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	5
4 Exemple de code pour Arduino .....	5
5 Exemple de code pour Raspberry Pi .....	8

### Photo

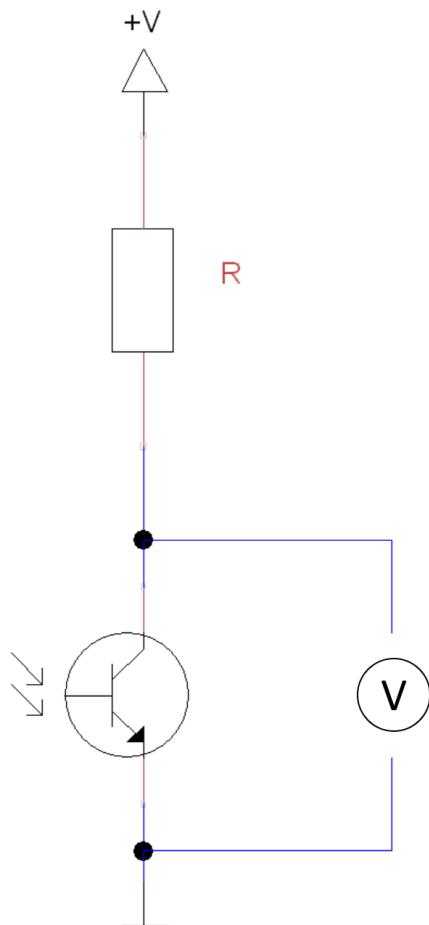


### Données techniques / Description sommaire

Si on met un doigt sur la diode émettrice et le phototransistor, le signal de sortie délivre une image des pulsations du sang dans votre doigt.

Un phototransistor fonctionne comme ceci: il agit comme un transistor normal sauf que le courant de la base est remplacé par de la lumière reçue. Lorsque le phototransistor reçoit de la lumière, il change d'état et peut commuter une intensité relativement importante (en fonction de ses caractéristiques).

KY-039 Capteur de pulsations cardiaques



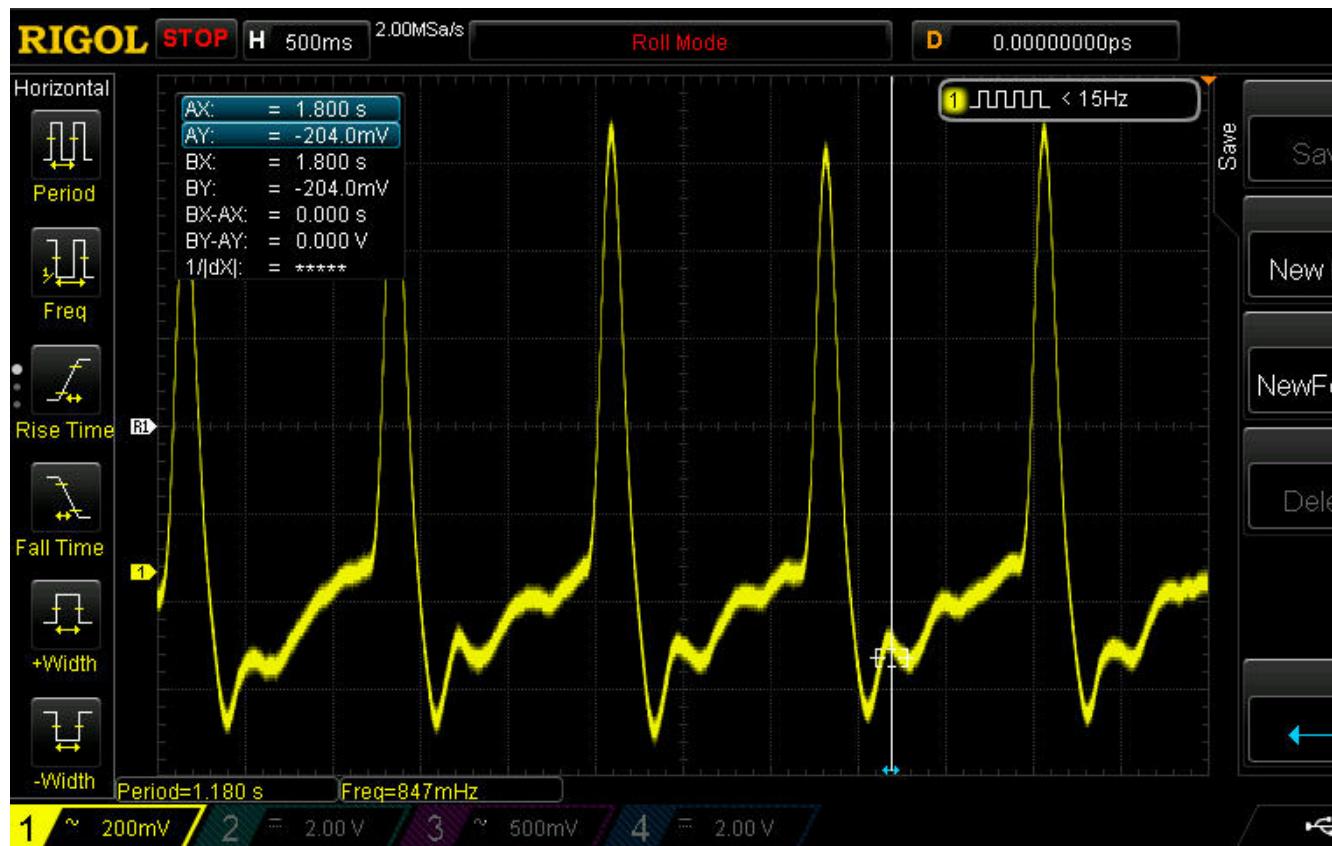
Si vous placez une résistance en série avec le phototransistor, vous pourrez observer le comportement suivant: la tension aux bornes d phototransistor sera proche de 0 V lorsqu'il sera exposé à une lumière extérieure, tandis que ce tension sera proche de la tension d'alimentatio l'obscurité.

Ce module permet de mesurer vos pulsations en plaçant un doigt entre la diode et le phototransistor.

Explication: si vous placez votre doigt devant une lampe de poche, il deviendra plus ou moins translucide. Selon l'endroit éclairé, on peut vaguement discerner le pompage du sang.

Ce pompage du sang dans la veine présente une densité différente et donc des différences de luminosité dans le flux sanguin. Ce sont ces différences de luminosité que le module peut détecter et ainsi compter les pulsations. Ce phénomène est illustré par la capture d'écran sur un oscilloscope:

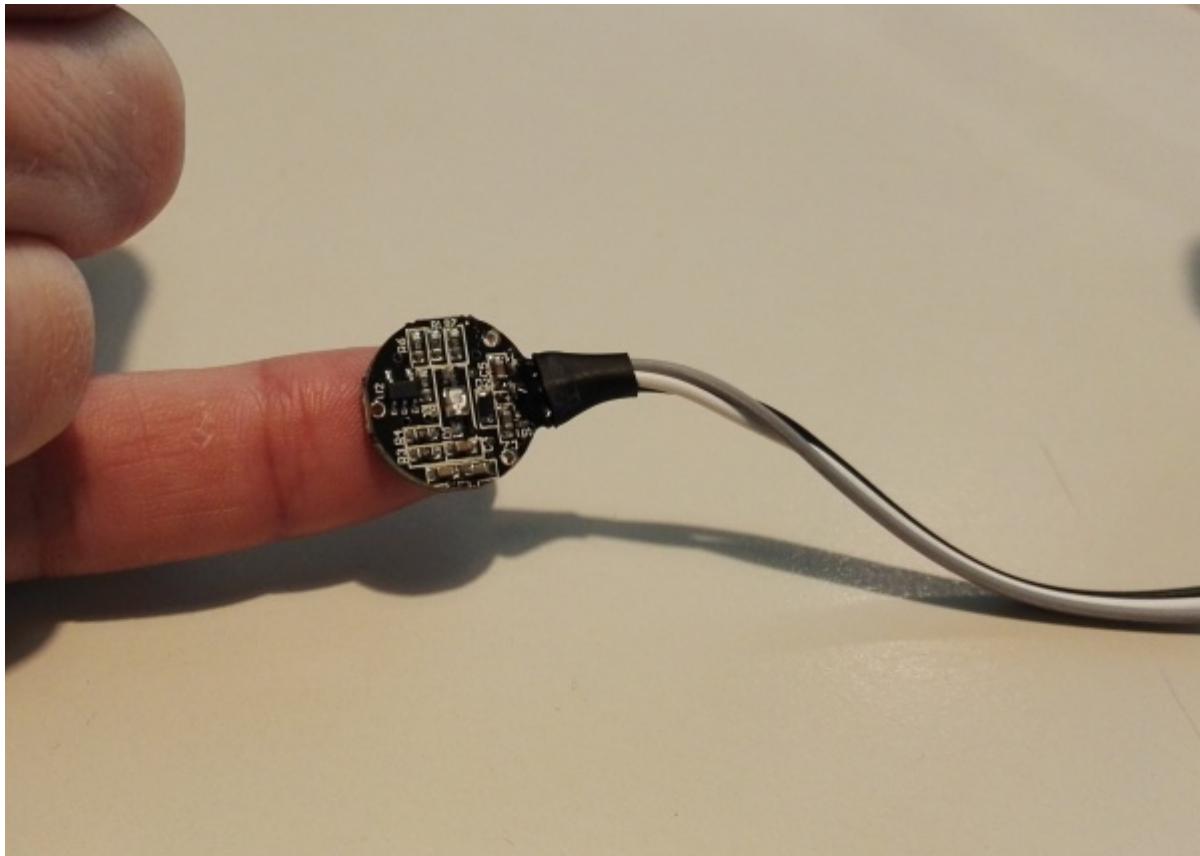
KY-039 Capteur de pulsations cardiaques



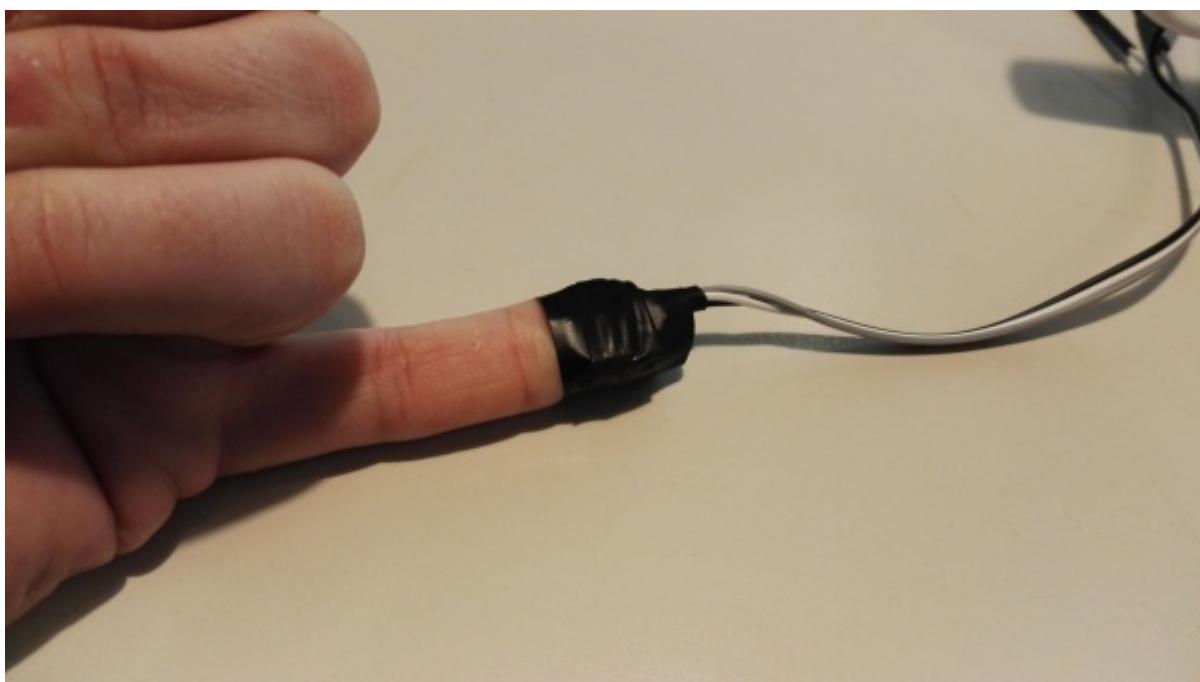
L'écran montre la variation de la tension au niveau du phototransistor (ainsi que les variations de luminosité dues à la circulation du sang) sur l'axe des ordonnées. Les pics ci-dessus représentent les impulsions du cœur. Si on compte les battements, on obtient environ 71 battements/minute.

Pour obtenir de bons résultats, nous recommandons d'effectuer la mesure sur en plaçant le capteur sur l'extrémité du petit doigt comme sur la photo ci-dessous:

## KY-039 Capteur de pulsations cardiaques



Pour un meilleur résultat, il est préférable de fixer le capteur à l'aire de sparadrap ou ruban adhésif.



L'enregistrement du rythme cardiaque est meilleur lorsque le capteur est placé au-dessus d'un vaisseau sanguin majeur.

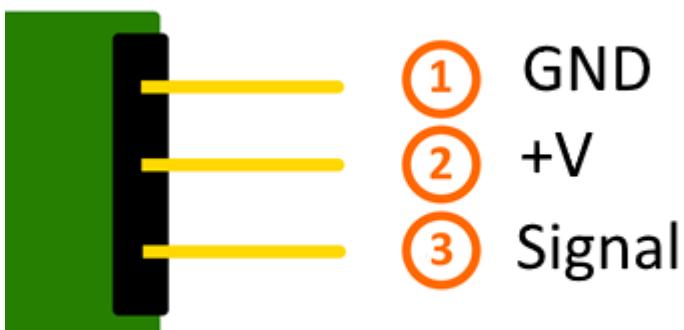
## KY-039 Capteur de pulsations cardiaques

Il est parfois utile de tester différents emplacements pour le capteur afin d'obtenir un enregistrement de meilleure qualité.

**Ce capteur est destiné à une utilisation didactique, le résultat dépend de la personne et de la position du doigt sur le capteur. Il est possible de ne pas obtenir de bons résultats dans certaines conditions.**

## Brochage

---



### Branchements des câbles :

Gris --> GND

Blanc --> +V

Noir --> Signal

## Exemple de code pour Arduino

---

L'exemple de code suivant est de Dan Truong qui l'a publié sous ce [[lien](#)] sous licence [[MIT OpenSource](#)]. La version ci-dessous est traduite en français, l'originale est à télécharger ci-dessous.

Ce code détecte une impulsion. Il n'enregistrera pas le rythme cardiaque, mais il va rechercher les pics qui seront assimilés à des pulsations et feront allumer la led. Le nombre de pulsations peut être plus ou moins calculé.

Si le doigt est replacé entre la LED et le phototransistor, un délai est nécessaire jusqu'à ce que le programme se recalibre pour la nouvelle mesure et délivre la valeur correcte.

```
///////////
/// Copyright (c)2015 Dan Truong
/// Permission is granted to use this software under the MIT
/// licence, with my name and copyright kept in source code
/// http://opensource.org/licenses/MIT
///
```

## KY-039 Capteur de pulsations cardiaques

```
/// KY039 Arduino Heartrate Monitor V1.0 (April 02, 2015)
///////////////////////////////////////////////////////////////////
///
/// Commentaires en français de GO TRONIC
///
///////////////////////////////////////////////////////////////////
/// @param[in] IRSensorPin broche analogique à laquelle est raccordé le capteur
/// @param[in] delay (msec) Le délai entre les appels de la fonction de balayage.
/// Les meilleurs résultats sont obtenus si vous appelez la fonction 5 fois/pulsion
/// Pas plus lent que 150 ms pour environ 70 pulsations/min.
/// Un délai de 60 ms ou plus rapide pour aller jusqu'à 200 pulsations/min.
///
///
/// @Résumé
/// Sortie Vraie si une pulsation est détectée
/// Ce code détecte seulement les impulsions, il ne donne
/// pas la forme d'onde des impulsions cardiaques.
///
///
///////////////////////////////////////////////////////////////////
int rawValue;

bool
heartbeatDetected(int IRSensorPin, int delay)
{
    static int maxValue = 0;
    static bool isPeak = false;

    bool result = false;

    rawValue = analogRead(IRSensorPin);
    // La tension au phototransistor est lue et stockée dans la variable rawValue
    rawValue *= (1000/delay);

    // Si la valeur de la dernière mesure dévie trop
    // (par exemple parce que le doigt a été enlevé ou a bougé)
    // maxValue se réinitialise
    if (rawValue * 4L < maxValue) { maxValue = rawValue * 0.8; }

    // Detect new peak
    if (rawValue > maxValue - (1000/delay)) {
        // Détection de l'impulsion. Si la nouvelle rawValue est plus grande
        // que la dernière valeur maximale, elle sera prise en compte en enregistrée
        if (rawValue > maxValue) {
            maxValue = rawValue;
        }
        // Attribution des pulsations
        if (isPeak == false) {
            result = true;
        }
        isPeak = true;
    } else if (rawValue < maxValue - (3000/delay)) {
        isPeak = false;
        // La valeur maximale est légèrement diminuée à chaque passage.
        // Cela peut prendre plusieurs secondes pour détecter de nouveau
        // le signal lorsque le doigt a bougé ou que le capteur est devant
        // une partie osseuse du doigt. On peut aussi vérifier le délai
        // depuis la dernière pulsation et s'il excède 1 seconde, on
        // réinitialise maxValue
        maxValue-=(1000/delay);
    }
    return result;
}
```

## KY-039 Capteur de pulsations cardiaques

```
///////////////////////////////
// Arduino main code
/////////////////////////////
int ledPin=13;
int analogPin=0;

void setup()
{
    // La LED Arduino intégrée (Digital 13) est utilisée pour la sortie
    pinMode(ledPin,OUTPUT);

    // Initialisation de la sortie série
    Serial.begin(9600);
    Serial.println("Exemple de code de détection de pulsations.");
}

const int delayMsec = 60; // 100msec per sample

// Le programme principal a deux sorties:
// - Si un battement de cœur est détecté, le voyant clignote
// - L'impulsion est calculée et envoyée vers la sortie série.

void loop()
{
    static int beatMsec = 0;
    int heartRateBPM = 0;
    Serial.println(rawValue);
    if (heartbeatDetected(analogPin, delayMsec)) {
        heartRateBPM = 60000 / beatMsec;
        // Sortie LED par impulsion
        digitalWrite(ledPin,1);

        // Envoi des données série
        Serial.print(rawValue);
        Serial.print(", ");
        Serial.println(heartRateBPM);

        beatMsec = 0;
    } else {
        digitalWrite(ledPin,0);
    }
    delay(delayMsec);
    beatMsec += delayMsec;
}
```

### Affectation des broches Arduino:

Sensor Signal = [Pin 0]  
 Sensor +V = [5V]  
 Sensor - = [Pin GND]

### Exemple de programme à télécharger

[KY-039-HeartBeatDetector original by DanTruong](#)

[KY-039-version par Joy-It et Gotronic](#)

KY-039 Capteur de pulsations cardiaques

## Exemple de code pour Raspberry Pi

**!! Attention !! Capteur analogique !! Attention !!**

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, le Sensorkit X40 inclut le [module KY-053](#) qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

Vous trouverez de plus amples informations à ce sujet dans la description du [module KY-053](#).

**!!Attention !! Capteur analogique !! Attention !!**

Dans ce programme, la fonction de détection du rythme cardiaque "heartBeatDetect" est appelée toutes les 10 ms (délais "delayTime" modifiable).

Si une pulsation est détectée, une impulsion est émise. Il est aussi possible de connecter une led sur la broche LED\_PIN (par défaut dans le programme: GPIO 24) pour avoir une visualisation des pulsations (la led s'éclaire à chaque pulsation détectée).

Si le doigt vient d'être placé ou a bougé, le module peut avoir besoin de se recalibrer (3-5 secondes) avant de délivrer une mesure correcte.

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page <https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code> sous [licence BSD](#).

```

#!/usr/bin/python
# coding=utf-8

#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Parts of Code based on Dan Truong's KY039 Arduino Heartrate Monitor V1.0
### [https://forum.arduino.cc/index.php?topic=209140.msg2168654] Message #29
###
### Traduction française par Go Tronic
###

# Ce programme utilise les bibliothèques Python ADS1115 et I2C pour Raspberry Pi
# Elles sont téléchargeables sous licence BSD via le lien suivant:
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

```

**KY-039 Capteur de pulsations cardiaques**

```

# Import et mise en place des autres modules
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Initialisation des variables
beatsPerMinute = 0
isPeak = False
result = False
delayTime = 0.01
maxValue = 0
schwelle = 25
beatTime = 0
oldBeatTime = 0

# Affectation des adresses ADS1x15 ADC
ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Sélection de l'amplification (gain)
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Taux d'échantillonnage du convertisseur analogique
sps = 8      # 8 Samples pro Sekunde
# sps = 16    # 16 Samples pro Sekunde
# sps = 32    # 32 Samples pro Sekunde
# sps = 64    # 64 Samples pro Sekunde
# sps = 128   # 128 Samples pro Sekunde
# sps = 250   # 250 Samples pro Sekunde
# sps = 475   # 475 Samples pro Sekunde
# sps = 860   # 860 Samples pro Sekunde

# Choix de la broche du convertisseur analogique (1-4)
adc_channel = 0 # Channel 0
# adc_channel = 1 # Channel 1
# adc_channel = 2 # Channel 2
# adc_channel = 3 # Channel 3

# Initialisation du convertisseur - ici le module KY-053 utilise le convertisseur ADS1115
adc = ADS1x15(ic=ADS1115)

# Déclaration de la broche sur laquelle est connectée la led
LED_PIN = 24
GPIO.setup(LED_PIN, GPIO.OUT, initial= GPIO.LOW)

#####
# Fonction de calcul du rythme cardiaque
def heartBeatDetect(schwelle):
    global maxValue
    global isPeak
    global result
    global oldBeatTime

    # La valeur de la tension du phototransistor est lue
    # et stockée dans la variable rawValue
    rawValue = adc.readADCSingleEnded(adc_channel, gain, sps)

    # Réinitialisation de la variable de résultat
    if result == True:
        result = False

```

## KY-039 Capteur de pulsations cardiaques

```

result = False

# Si la valeur mesurée est trop loin de la valeur maximale
# (par exemple si le doigt vient d'être mis ou a bougé),
# alors la valeur maximale est réinitialisée pour avoir une nouvelle valeur de référence
if rawValue * 4 < maxValue:
    maxValue = rawValue * 0.8
# Détection d'une crête. Si la valeur mesurée
# rawValue est plus grande que la valeur haute maxValue
# elle devient la nouvelle valeur haute.
if rawValue > (maxValue - schwelle):
    if rawValue > maxValue:
        maxValue = rawValue
    # Et un battement de cœur est compté
    if isPeak == False:
        result = True

isPeak = True

else:
    if rawValue < (maxValue - schwelle):
        isPeak = False
        # Si il n'y a pas de crête détectée,
        # la valeur maximale est légèrement diminuée à chaque cycle
        # pour s'assurer que les pics suivants dépassent bien cette
        # valeur maximale et soient pris en compte, mais aussi dans
        # le cas où le doigt se serait légèrement déplacé et que le
        # signal serait plus faible.
        maxValue = maxValue - schwelle/2

# Dans le cas où un battement a été détecté, la sortie est activée
if result == True:

    # Calcul du rythme cardiaque
    # L'heure du système est enregistrée à chaque battement
    # et est comparée à l'heure du battement précédent.
    # La différence entre les deux est alors utilisée pour
    # calculer le rythme cardiaque.
    beatTime = time.time()
    timedifference = beatTime - oldBeatTime
    beatsPerMinute = 60/timedifference
    oldBeatTime = beatTime

    # En plus du calcul du rythme cardiaque, une led indique les battements
    GPIO0.output(LED_PIN, GPIO0.HIGH)
    time.sleep(delayTime*10)
    GPIO0.output(LED_PIN, GPIO0.LOW)

    # Le rythme cardiaque est retourné comme résultat de la fonction
    return beatsPerMinute
#####
# #####
# Boucle de programme principale
# #####
# Dans ce programme, la fonction de détection du rythme cardiaque "heartBeatDetect"
# est appelée toutes les 10 ms (délais "delayTime" modifiable).
# Si une pulsation est détectée, une impulsion est émise.

try:
    while True:
        time.sleep(delayTime)
        beatsPerMinute = heartBeatDetect(schwelle)
        if result == True:

```

## KY-039 Capteur de pulsations cardiaques

```
    print "----Pulsion détectée !---- Pulsion:", int(beatsPerMinute), "(bpm)"  
  
except KeyboardInterrupt:  
    GPIO.cleanup()
```

**Brochage Raspberry Pi:**

## Module KY-039

Signal	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]
+V	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

## ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	Masse	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	s.o.	[Sensor: Signal]

**Exemple de programme à télécharger**

- [KY-039\\_HeartBeatDetector](#)

Commande pour lancer le programme:

```
sudo python KY-039_HeartBeatDetector.py
```

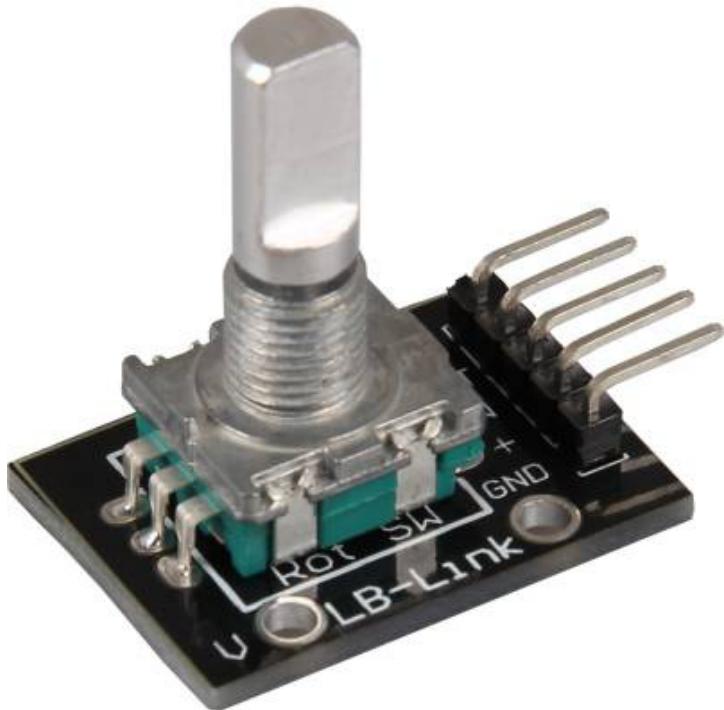
## KY-040 Module encodeur rotatif

## KY-040 Module encodeur rotatif

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Codage .....	1
4 Brochage .....	2
5 Exemple de code pour Arduino .....	2
6 Exemple de code pour Raspberry Pi .....	4

### Photo



### Données techniques / Description sommaire

La sortie codée de l'encodeur donne la position de celui-ci.

### Codage

Un encodeur rotatif délivre deux signaux décalés de 90°. Ceci permet de connaître le sens de rotation en comparant les mouvements des 2 signaux. Si on fait bouger l'encodeur d'un seul cran, un seul des 2 signaux sera modifié.

#### **Sens des aiguilles d'une montre [A change en premier] -> Pin\_CLK**

## KY-040 Module encodeur rotatif

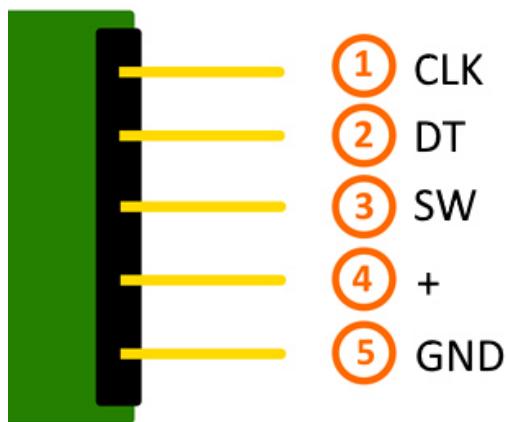
A	B
0	0
1	0
1	1
0	1
0	0

**Sens contraire des aiguilles d'une montre [B change en premier] -> Pin\_DT**

A	B
0	0
0	1
1	1
1	0
0	0

## Brochage

---



## Exemple de code pour Arduino

---

Le programme vérifie si un changement des états des broches a eu lieu, et selon la broche qui est modifiée en premier, détermine le sens de rotation.

Cette information est obtenue par la comparaison des signaux présents sur les 2 sorties.

## KY-040 Module encodeur rotatif

Une fois que la direction a été déterminée, on compte les mouvements depuis la position de départ et on les envoie vers la sortie série.

Une pression sur le bouton de l'encodeur réinitialise la position actuelle.

### Pour la sortie série: Baudrate= 115200

```
// Initialisation des variables nécessaires
int Compteur = 0;
boolean Direction;
int Pin_clk_Letzter;
int Pin_clk_Aktuell;

// Définition des broches d'entrées
int pin_clk = 3;
int pin_dt = 4;
int button_pin = 5;

void setup()
{
    // Initialisation des broches d'entrées...
    pinMode (pin_clk,INPUT);
    pinMode (pin_dt,INPUT);
    pinMode (button_pin,INPUT);

    // ...et activation de leurs résistances de PULL-UP
    digitalWrite(pin_clk, true);
    digitalWrite(pin_dt, true);
    digitalWrite(button_pin, true);

    // Lecture initiale de Pin_CLK
    Pin_clk_Letzter = digitalRead(pin_clk);
    Serial.begin (115200);
}

// Le programme vérifie si un changement des états des broches a eu lieu,
// et selon la broche qui est modifiée en premier, détermine le sens de rotation.
// Cette information est obtenue par la comparaison des signaux présents sur les 2 sorties.
// Une fois que la direction a été déterminée, on compte les mouvements depuis la position
// de départ et on les envoie vers la sortie série.
// Une pression sur le bouton de l'encodeur réinitialise la position actuelle.

void loop()
{
    // Lecture des statuts actuels
    Pin_clk_Aktuell = digitalRead(pin_clk);

    // Vérification de changement
    if (Pin_clk_Aktuell != Pin_clk_Letzter)
    {

        if (digitalRead(pin_dt) != Pin_clk_Aktuell)
        {
            // Pin_CLK a changé en premier
            Compteur++;
            Direction = true;
        }

        else
        {
            // Sinon Pin_DT achangé en premier
            Direction = false;
            Compteur--;
        }
    }

    Serial.println ("Rotation detectee: ");
}
```

## KY-040 Module encodeur rotatif

```

Serial.println ("Rotation detectee: ");
Serial.print ("Sens de rotation: ");

if (Direction)
{
    Serial.println ("dans le sens des aiguilles d'une montre");
}
else
{
    Serial.println("dans le sens contraire des aiguilles d'une montre");
}

Serial.print("Position actuelle: ");
Serial.println(Compteur);
Serial.println("-----");

}

// Préparation de la prochaine exécution:
Pin_clk_Letzter = Pin_clk_Aktuell;

// fonction Reset remise à la position actuelle
if (!digitalRead(button_pin) && Compteur!=0)
{
    Compteur = 0;
    Serial.println("Position reset");
}

}

```

### Affectation des broches Arduino:

CLK	= [Pin 3]
DT	= [Pin 4]
SW = Boutton	= [Pin 5]
+	= [Pin 5V]
GND	= [Pin GND]

### Exemple de programme à télécharger

[KY-040\\_RotaryEncoder.zip](#)

## Exemple de code pour Raspberry Pi

Le programme vérifie si un changement des états des broches a eu lieu, et selon la broche qui est modifiée en premier, détermine le sens de rotation.

Cette information est obtenue par la comparaison des signaux présents sur les 2 sorties.

Une fois que la direction a été déterminée, on compte les mouvements depuis la position de départ et on les envoie vers la sortie série.

Une pression sur le bouton de l'encodeur réinitialise la position actuelle.

```

# coding=utf-8
# Les modules nécessaires sont importés et mis en place
import RPi.GPIO as GPIO
import time

```

## KY-040 Module encodeur rotatif

```

GPIO.setmode(GPIO.BCM)

# Déclaration des broches d'entrée auxquelles est raccordé le capteur
PIN_CLK = 16
PIN_DT = 15
BUTTON_PIN = 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(BUTTON_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

# Les variables nécessaires sont initialisées
Counter = 0
Direction = True
PIN_CLK_LETZTER = 0
PIN_CLK_AKTUELL = 0
delayTime = 0.01

# Lecture initiale de Pin_CLK
PIN_CLK_LETZTER = GPIO.input(PIN_CLK)

# Cette fonction de sortie est exécutée par détection d'un signal
def ausgabeFunktion(null):
    global Counter

    PIN_CLK_AKTUELL = GPIO.input(PIN_CLK)

    if PIN_CLK_AKTUELL != PIN_CLK_LETZTER:

        if GPIO.input(PIN_DT) != PIN_CLK_AKTUELL:
            Counter += 1
            Direction = True;
        else:
            Direction = False
            Counter = Counter - 1

        print "Rotation détectée: "

        if Direction:
            print "Sens de rotation: sens des aiguilles d'une montre"
        else:
            print "Sens de rotation: sens contraire des aiguilles d'une montre"

        print "Position actuelle: ", Counter
        print "-----"

def CounterReset(null):
    global Counter

    print "Position remise à 0!"
    print "-----"
    Counter = 0

# Pour intégrer directement un temps de stabilisation, on initialise
# les GPIO au moyen de l'option CallBack
GPIO.add_event_detect(PIN_CLK, GPIO.BOTH, callback=ausgabeFunktion, bouncetime=50)
GPIO.add_event_detect(BUTTON_PIN, GPIO.FALLING, callback=CounterReset, bouncetime=50)

print "Sensor-Test [Appuyez sur Ctrl + C pour terminer le test]"

# Boucle de programme principale
try:
    while True:
        time.sleep(delayTime)

```

## KY-040 Module encodeur rotatif

```
# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

**Brochage Raspberry Pi:**

CLK	=	GPIO16	[Pin 36]
DT	=	GPIO15	[Pin 10]
SW	=	GPIO14	[Pin 8]
+	=	3,3V	[Pin 1]
GND	=	Masse	[Pin 6]

**Exemple de programme à télécharger**[KY-040\\_RPi\\_RotaryEncoder.zip](#)

Commande pour lancer le programme:

```
sudo python KY-040_RPi_RotaryEncoder.py
```

## KY-050 Capteur à ultrasons

## KY-050 Capteur à ultrasons

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Principe de fonctionnement .....	2
5 Exemple de code pour Arduino .....	4
6 Exemple de code pour Raspberry Pi .....	5

### Photo



### Données techniques / Description sommaire

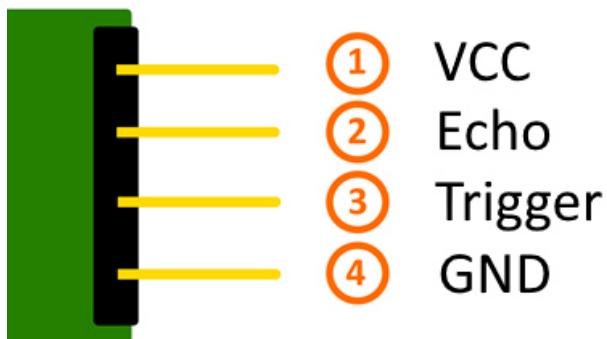
Une mesure de distance est déclenchée par l'introduction d'un signal à front descendant sur l'entrée Trigger. Un signal PWM-TTL est émis et l'écho est capté par le récepteur.

**Plage de mesure:** 2 à 300cm - **Résolution:** 3mm

**Intervalle mini entre 2 mesures:** 50µs

## Brochage

---



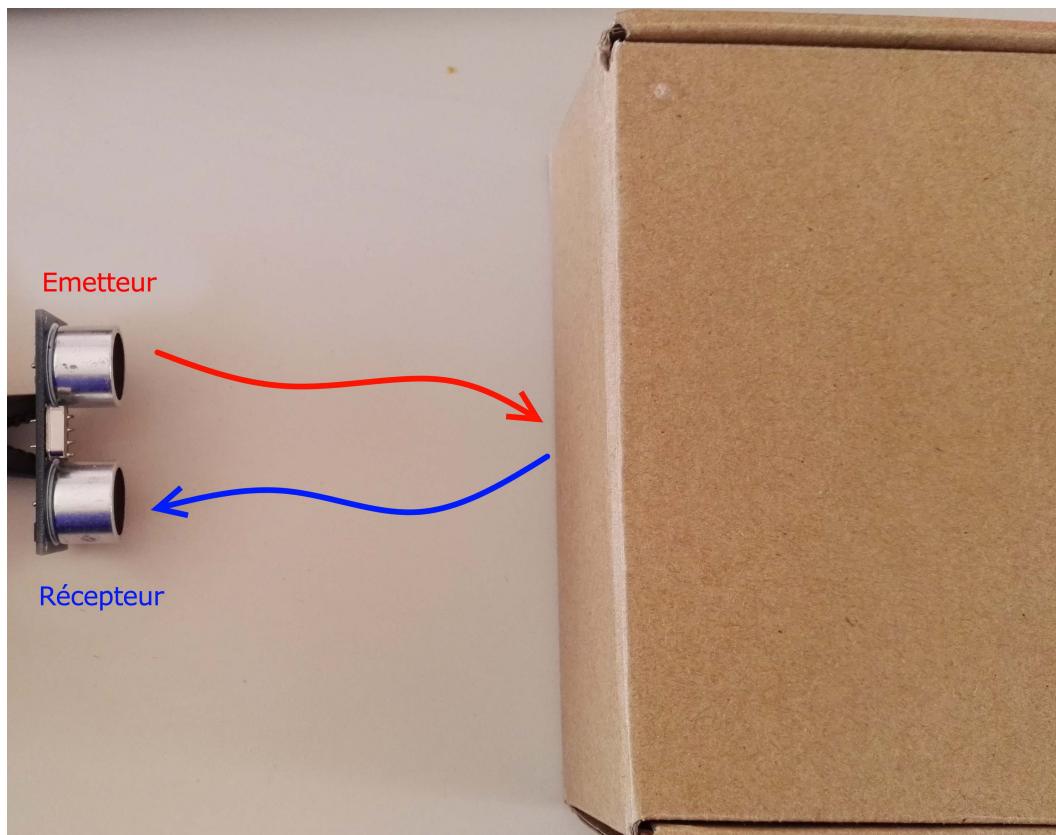
## Principe de fonctionnement

---

Le principe de fonctionnement est basé sur la vitesse du son qui reste à peu près constante à 343,2 m/s dans l'air à une température de 20°C. Le module est composé de deux capsules ultrasoniques (émetteur + récepteur). L'émetteur va envoyer une salve d'ondes ultrasonores qui vont se réfléchir sur l'obstacle. Les ondes renvoyées en écho sont réceptionnées par le récepteur.

Le calcul de la distance peut se faire par un microcontrôleur en fonction du temps de réponse entre l'émission et la réception du signal.

## KY-050 Capteur à ultrasons



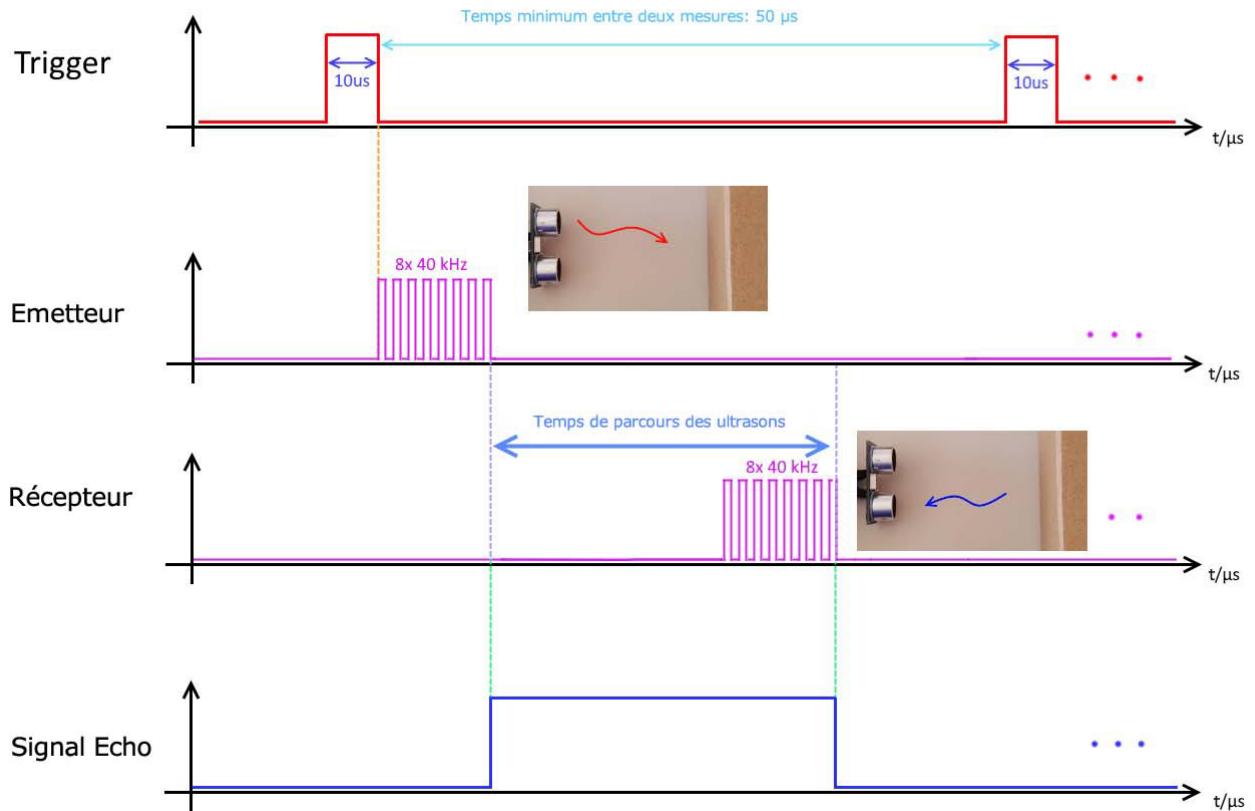
Les capsules ultrasoniques fonctionnent à la fréquence de 40 kHz et sont donc largement en-dehors du spectre audible (20 Hz à 20 kHz).

L'envoi du signal ultrasonore est déclenché par un signal de démarrage de 10 µs (ActiveHigh) sur la "broche d'entrée de déclenchement". Après l'envoi du signal, la "broche de réception de l'écho" est activée (ActiveHigh). Si le récepteur a reçu le signal, la "broche de réception de l'écho" est désactivée.

Le temps entre l'activation et la désactivation de la "broche de réception de l'écho" peut être mesuré et converti en distance, car cela correspond également à la durée pendant laquelle le signal ultrasonore parcourt la distance entre l'émetteur et la paroi réfléchissante et ensuite le retour vers le récepteur.

La conversion est alors réalisée sur l'approximation d'une vitesse des ondes constante. La distance correspond par conséquent à la moitié de la distance parcourue.

## KY-050 Capteur à ultrasons



## Exemple de code pour Arduino

En utilisant le principe de fonctionnement décrit ci-dessus, cet exemple de code mesure la durée de temps nécessaire pour que le signal effectue son parcours complet (émetteur --> obstacle --> récepteur) à l'aide de la fonction [pulseln](#).

Cette durée est utilisée pour calculer la distance jusqu'à l'obstacle. La distance est ensuite envoyée à la sortie série.

En cas de dépassement de la portée du capteur, un signal d'erreur est émis.

```
#define Echo_EingangsPin 7 // Broche d'entrée Echo
#define Trigger_AusgangsPin 8 // Broche de sortie Trigger

// Définition des variables nécessaires
int maximumRange = 300;
int minimumRange = 2;
long Distance;
long Duree;

void setup() {
    pinMode(Trigger_AusgangsPin, OUTPUT);
    pinMode(Echo_EingangsPin, INPUT);
    Serial.begin(9600);
}

void loop() {
    // Le début de la mesure est lancé par un signal de 10µs sur Trigger
```

## KY-050 Capteur à ultrasons

```

digitalWrite(Trigger_AusgangsPin, HIGH);
delayMicroseconds(10);
digitalWrite(Trigger_AusgangsPin, LOW);

// Attente à l'entrée Echo jusqu'à ce que le signal soit activé
// et mesure de la durée
Duree = pulseIn(Echo_EingangsPin, HIGH);

// Calcul de la distance en fonction du temps enregistré
Distance = Duree/58.2;

// On vérifie si le signal se trouve dans la plage de mesure du capteur
if (Distance >= maximumRange || Distance <= minimumRange)
{
    // Si ce n'est pas le cas, affichage d'un signal d'erreur
    Serial.println("Distance en-dehors de la portée");
    Serial.println("-----");
}

else
{
    // La distance calculée est envoyée vers la sortie série
    Serial.print("La distance est de: ");
    Serial.print(Distance);
    Serial.println("cm");
    Serial.println("-----");
}
// Pause entre les mesures
delay(500);
}

```

### Affectation des broches Arduino:

VCC	= [Pin 5V]
Echo	= [Pin 7]
Trigger	= [Pin 8]
GND capteur	= [Pin GND]

### Exemple de programme à télécharger

[KY-050-UltraschallabstandSensor.zip](#)

## Exemple de code pour Raspberry Pi

**!! Attention !! Tension 5 Vcc !! Attention !!**

La carte Raspberry Pi fonctionne en 3,3Vcc tandis que le capteur nécessite 5 Vcc pour fonctionner. Si on raccorde directement ce capteur à la Raspberry Pi, celle-ci pourrait être endommagée de façon permanente.

Pour palier à ce problème, il est nécessaire d'utiliser un convertisseur de niveau de tension tel que le [KY-051](#), à placer entre la Raspberry et le capteur.

Plus d'informations disponibles à la page [KY-051 Voltage Translator / Level Shifter](#)

**!! Attention !! Tension 5 Vcc !! Attention !!**

En utilisant le principe de fonctionnement décrit ci-dessus, cet exemple de code mesure la durée de temps nécessaire pour que le signal effectue son parcours complet (émetteur --> obstacle --> récepteur). On utilise une sorte de chronomètre et la fonction time.time()

## KY-050 Capteur à ultrasons

Cette durée est utilisée pour calculer la distance jusqu'à l'obstacle. La distance est ensuite envoyée à la console.

En cas de dépassement de la portée du capteur, un signal d'erreur est émis.

```
# coding=utf-8
# Les modules nécessaires sont importés et mis en place
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

# Sélection des broches d'entrée/sortie respectifs
Trigger_AusgangsPin = 17
Echo_EingangsPin     = 27

# Réglage de la pause entre les mesures
sleeptime = 0.8

# Configuration des broches d'entrée/sortie
GPIO.setup(Trigger_AusgangsPin, GPIO.OUT)
GPIO.setup(Echo_EingangsPin, GPIO.IN)
GPIO.output(Trigger_AusgangsPin, False)

# Boucle de programme principale
try:
    while True:
        # Le début de la mesure est lancé par un signal de 10µs sur Trigger
        GPIO.output(Trigger_AusgangsPin, True)
        time.sleep(0.00001)
        GPIO.output(Trigger_AusgangsPin, False)

        # Démarrage du chronomètre
        HeureActive = time.time()
        while GPIO.input(Echo_EingangsPin) == 0:
            HeureActive = time.time() # L'heure actuelle est enregistrée jusqu'à ce que le signal soit activé

        while GPIO.input(Echo_EingangsPin) == 1:
            HeureInactive = time.time() # Enregistrement de l'heure au moment où le signal est encore actif

        # La différence en les deux heures donne la durée recherchée
        Duree = HeureInactive - HeureActive
        # Calcul de la distance en fonction de la durée enregistrée
        Distance = (Duree * 34300) / 2

        # On vérifie si le signal se trouve dans la plage de mesure du capteur
        if Distance < 2 or (round(Distance) > 300):
            # Si ce n'est pas le cas, affichage d'un signal d'erreur
            print("Distance en-dehors de la portée")
            print("-----")
        else:
            # La distance est formatée à 2 décimales
            Distance = format((Duree * 34300) / 2, '.2f')
            # La distance calculée est envoyée vers la console
            print("La distance mesure:"), Distance, ("cm")
            print("-----")

        # Pause entre les mesures
        time.sleep(sleeptime)
```

## KY-050 Capteur à ultrasons

```
# réinitialisation de tous les GPIO en entrées
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Brochage Raspberry Pi:

Capteur KY-050:

VCC	=	5V	[Pin 2 (RPi)]
Trigger	=	Pin B1	[convertisseur KY-051]
Echo	=	Pin B2	[convertisseur KY-051]
GND	=	Masse	[Pin 6 (RPi)]

Convertisseur KY-051:

VCCb	=	5V	[Pin 04(RPi)]
Pin B1	=	Trigger	[Capteur KY-050]
Pin B2	=	Echo	[Capteur KY-050]
VCCA	=	3,3V	[Pin 01(RPi)]
Pin A1	=	GPIO17	[Pin 11(RPi)]
Pin A2	=	GPIO27	[Pin 13(RPi)]
GND	=	Masse	[Pin 06(RPi)]

- Les autres broches du convertisseur KY-051 ne doivent pas être raccordées (OE,B3,B4,A3,A4).

### Exemple de programme à télécharger

[KY-050\\_RPi\\_UltraSon.zip](#)

Commande pour lancer le programme:

```
sudo python KY-050_RPi_UltraSon.py
```

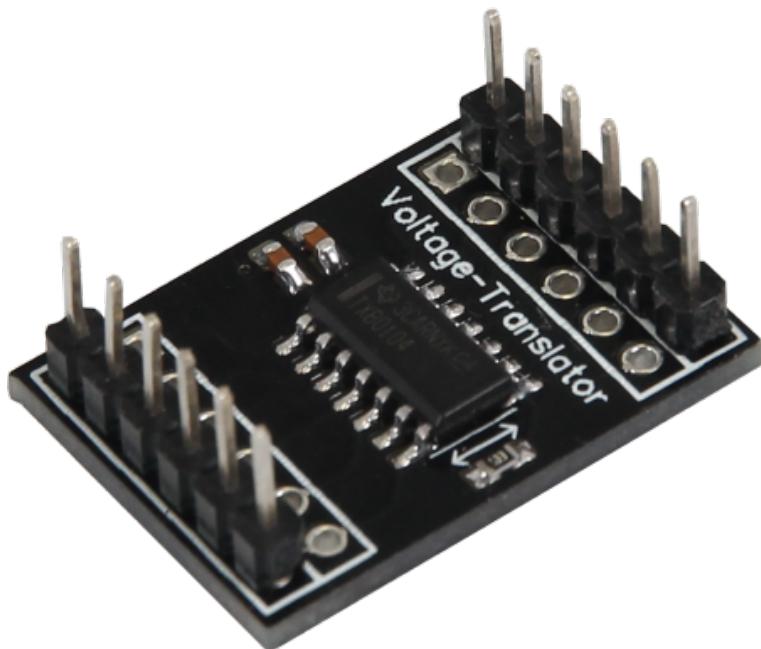
KY-051 Voltage Translator / Level Shifter

## KY-051 Voltage Translator / Level Shifter

---

### Photo

---



### Données techniques / Description sommaire

---

Ce convertisseur de niveau à 4 canaux adapte les niveaux de tensions entre les entrées et les sorties du module, vers le haut ou vers le bas.

Il existe une grande variété de microcontrôleurs qui fonctionnent dans différentes gammes de tension: par exemple, une carte Raspberry qui utilise le 3,3 Vcc nécessitera une adaptation des niveaux de tension pour utiliser ou communiquer avec des cartes compatibles Arduino qui fonctionnent pour la plupart en 5 Vcc.

La tension de 5 Vcc était justifiée pour améliorer la communication entre les microcontrôleurs pour compenser le bruit et les interférences dans des communications à longue distance. Les microcontrôleurs de conception plus récente permettent de travailler en 3,3 Vcc en conservant des bonnes caractéristiques de communication tout en consommant moins d'énergie. On rencontre également de plus en plus de systèmes fonctionnant sous 1,8 Vcc.

Si on essaye de raccorder deux microcontrôleurs fonctionnant avec des tensions différentes sans convertisseur de niveau, celui qui est alimenté dans la plus basse tension va devoir absorber l'excès de tension et risquera d'être détérioré.

### Brochage

---

L'affectation des broches est imprimée sur le circuit imprimé du module

## KY-051 Voltage Translator / Level Shifter

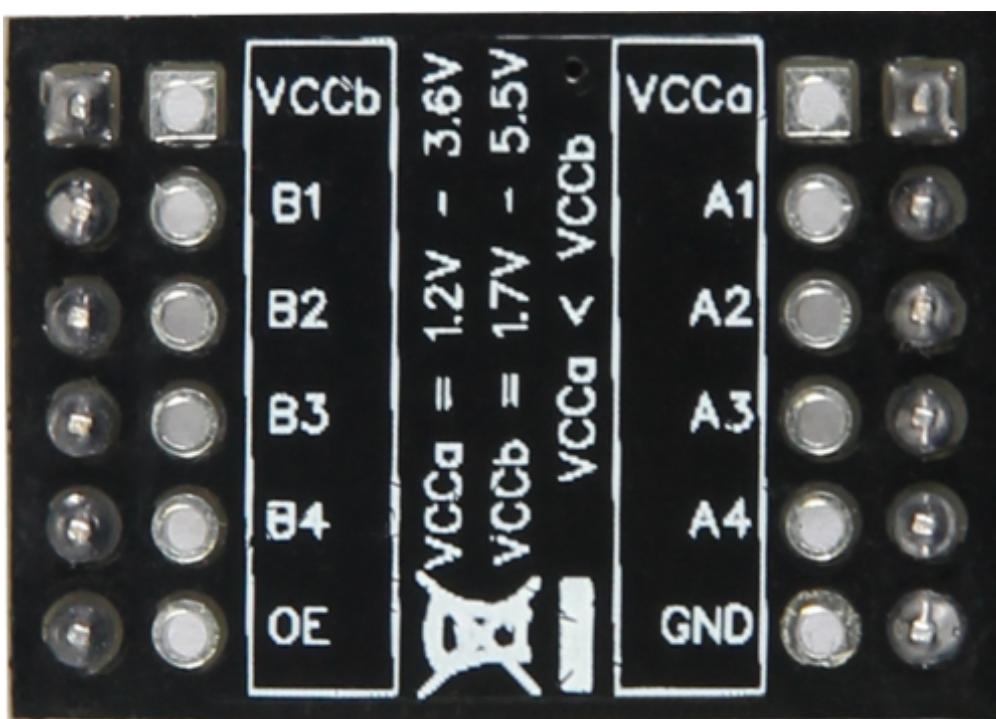
Les tensions sur les entrées / sorties A1-A4 et B1 à B4 sont adaptées vers le niveau de tension respectif (VCCA -> A1-A4 | VCCb -> B1-B4)

*Exemple:*

*Sortie Arduino -> Digital [ON] = 5V @ B1 >>>>> 3.3V @ A1 -> Entrée Raspberry Pi*

Un logiciel ou un code supplémentaire ne sont pas nécessaires pour le fonctionnement de ce module, il fonctionne de manière autonome.

**Veuillez noter que VCCb doit être supérieur ou égal à VCCA (Exemple: VCCb=5V - VCCA=3,3V)**



### **Exemple de raccordement entre Arduino et Raspberry Pi:**

*Brochage Arduino:*

VCCb	=	[Pin 5V]
------	---	----------

B1	=	[Pin 03]
----	---	----------

B2	=	[Pin 04]
----	---	----------

B3	=	[Pin 05]
----	---	----------

B4	=	[Pin 06]
----	---	----------

GND	=	[Pin GND]
-----	---	-----------

## KY-051 Voltage Translator / Level Shifter

*Brochage Raspberry Pi:*

VCCa	=	3,3V	[Pin 01]
A1	=	GPIO18	[Pin 12]
A2	=	GPIO03 / SCL	[Pin 05]
A3	=	GPIO02 / SDA	[Pin 01]
A4	=	GPIO14	[Pin 08]
GND	=	GND	[Pin 09]

**Veuillez vous assurer que les 2 systèmes sont bien raccordés à la même masse GND. La broche OE ne doit pas être connectée.**

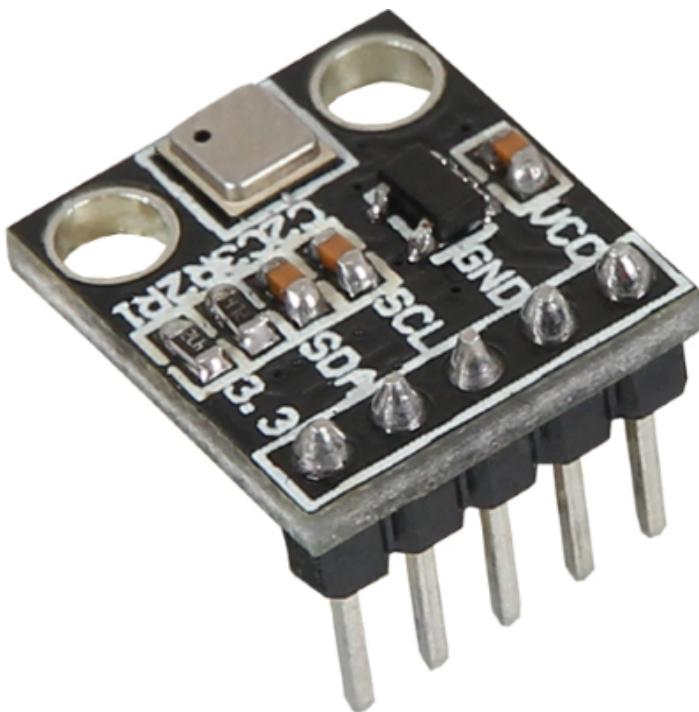
KY-052 Capteur de pression BMP180

## KY-052 Capteur de pression BMP180

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	2
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	3

### Photo

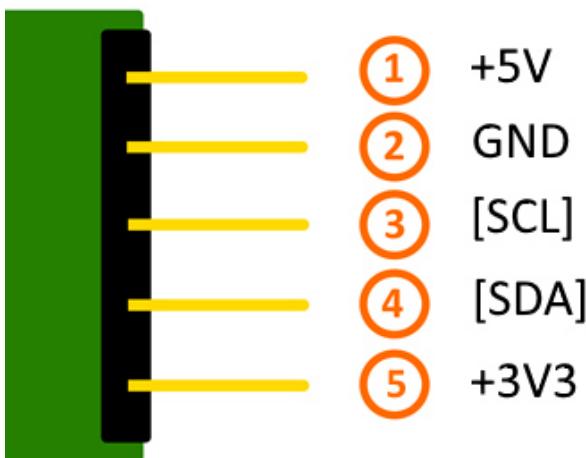


### Données techniques / Description sommaire

Ce capteur mesure la pression atmosphérique (élément de mesure au niveau du petit trou sur le boîtier argenté) et communique via le bus I2C.

**Un logiciel spécifique est nécessaire à l'utilisation de ce capteur.**

## Brochage



L'alimentation du capteur se fait soit par la broche 1 en 5V, soit par la broche 5 en 3,3V suivant qu'on le connecte à un Arduino (5V) ou un Raspberry Pi (3,3V).

**Attention : ne pas connecter les deux broches en même temps.**

## Exemple de code pour Arduino

Ce capteur ne donne pas sa mesure sur une broche analogique, mais via le bus I2C.

Il y a plusieurs façons d'utiliser le capteur, mais l'utilisation de la bibliothèque Adafruit\_BMP085 permet une utilisation simplifiée et est disponible via [ce lien](#) sous licence BSD OpenSource.

L'exemple ci-dessous utilise cette bibliothèque - qui est à télécharger de Github et à décompresser dans le dossier bibliothèque Arduino (par défaut C:\Users\[NomUtilisateur]\Documents\Arduino\libraries).

```
// Les bibliothèques requises sont définies et configurées
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085_U.h>
Adafruit_BMP085_Unified BMPSensor = Adafruit_BMP085_Unified(10085);

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

    Serial.println("KY-052 Test du capteur:");

    // Si le capteur ne détecte rien, une erreur s'affiche
    if(!BMPsensor.begin())
    {
        Serial.println("KY-052-Capteur non reconnu!");
        Serial.print("Verifier la connexion");
        while(1);
    }
}
```

## KY-052 Capteur de pression BMP180

```

void loop(void)
{
    // Initialisation de la bibliothèque Adafruit BMP
    sensors_event_t event;
    BMPSensor.getEvent(&event);

    // Initialisation de la mesure quand le capteur est prêt
    if (event.pressure)
    {
        Serial.println("-----");

        // Mesure de la pression de l'air
        Serial.print("Pression atmosphérique:      ");
        Serial.print(event.pressure);
        Serial.println(" hPa");

        // Mesure de la température
        float temperature;
        BMPSensor.getTemperature(&temperature);
        Serial.print("Température:      ");
        Serial.print(temperature);
        Serial.write(176); // Code Unicode pour le symbole °
        Serial.println(" °C");

        // Calcul de la hauteur au-dessus du niveau de la mer,
        // à partir des données enregistrées (SLP=1013.25 hPa)
        float seaLevelPressure = SENSORS_PRESSURE_SEALEVELHPA;
        Serial.print("Hauteur par rapport au niveau de la mer:      ");
        Serial.print(BMPSensor.pressureToAltitude(seaLevelPressure,
                                                   event.pressure,
                                                   temperature));
        Serial.println(" m");
        Serial.println("-----");
        Serial.println("");
    }
    // Message d'erreur si le capteur est inaccessible
    else
    {
        Serial.println("Erreur de capteur");
    }
    delay(1000);
}

```

### Exemple de programme à télécharger:

[KY-052-DruckSensor\\_TemperaturSensor.zip](#)

### Affectation des broches Arduino:

VCC	= [Pin 5V]
GND	= [Pin GND]
SCL	= [Pin SCL]
SDA	= [Pin SDA]
3,3	= [N.C.]

### Exemple de code pour Raspberry Pi

Le programme utilise les bibliothèques Python BMP085/180 et I2C de la société Adafruit. Elles sont disponibles via [ce lien sous licence MIT](#).

## KY-052 Capteur de pression BMP180

Ces bibliothèques doivent être installées au préalable comme suit :

D'abord installer le logiciel GitHub (si ce n'est pas encore fait) :

```
sudo apt-get install git
```

Pour ce faire, le Raspberry Pi doit être connecté à internet.

Avec la commande...

```
git clone https://github.com/adafruit/Adafruit_Python_BMP.git
```

... vous pouvez télécharger et décompresser la dernière version de la bibliothèque Adafruit\_BMP085.

Ensuite, on accède au dossier...

```
cd Adafruit_Python_BMP/
```

... dans le dossier téléchargé et on peut installer...

```
sudo python setup.py install
```

... la bibliothèque que nous utiliserons par la suite.

Ainsi, la Raspberry Pi peut communiquer avec le capteur via le bus I2C, à condition que la fonction I2C soit activée sur le Raspberry Pi.

Pour l'activer, il suffit d'ajouter la ligne suivante à la fin du fichier "/boot/config.txt" :

```
dtparam=i2c_arm=on
```

Ce fichier peut être modifié avec la commande :

```
sudo nano /boot/config.txt
```

Enregistrer et fermer le fichier en utilisant la séquence de touches [Ctrl+X -> Y -> Enter].

A ce stade, des bibliothèques supplémentaires sont nécessaires pour pouvoir utiliser I2C en Python. Pour les installer, entrez la commande suivante dans la console :

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

Enfin, l'exemple de code python suivant peut être utilisé.

Ce programme initialise le capteur et récupère les mesures de la pression de l'air, la température et la hauteur au-dessus du niveau de la mer.

## KY-052 Capteur de pression BMP180

```

#!/usr/bin/python
# coding=utf-8
# Copyright (c) 2014 Adafruit Industries

# Les bibliothèques requises sont importées et configurées
import Adafruit_BMP.BMP085 as BMP085
import time

# L'intervalle entre les prises de mesures peut se régler ici
sleeptime = 1

try:
    # Initialisation du capteur
    BMPSensor = BMP085.BMP085()
# Vérifie que le capteur est bien connecté
# Sinon, on retourne un message d'erreur
except IOError:
    print("-----")
    print ("KY-053 Capteur non reconnu!")
    print ("Vérifiez les connexions")
    print("-----")
    while(True):
        time.sleep(sleeptime)
except KeyboardInterrupt:
    GPIO.cleanup()

# Boucle principale
# Le programme démarre les mesures du capteur et
# les affiche dans la console
try:
    while(1):
        print("-----")
        # Température
        print('Temperature = {0:0.2f}°C'.format(BMPSensor.read_temperature()))
        # Pression atmosphérique
        print('Pression atmosphérique = {0:0.2f}hPa'.format(BMPSensor.read_pressure()/100))
        # Hauteur au-dessus de la mer
        print('Hauteur au-dessus de la mer = {0:0.2f}m'.format(BMPSensor.read_altitude()))
        print("-----")
        print("")
        time.sleep(sleeptime)

# Nettoyage de la console après la fin du programme
except KeyboardInterrupt:
    GPIO.cleanup()

```

### Brochage Raspberry Pi:

VCC	= -	[N.C]
GND	= Masse	[Pin 06]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]
3.3	= 3,3V	[Pin 01]

### Exemple de programme à télécharger

[KY-052\\_RPi\\_TempPress.zip](#)

Commande pour lancer le programme:

KY-052 Capteur de pression BMP180

```
sudo python KY-052_RPi_TempPress.py
```

## KY-053 Convertisseur analogique digital

### Sommaire

1 Photo .....	1
2 Données techniques / Description sommaire .....	1
3 Brochage .....	1
4 Exemple de code pour Arduino .....	2
5 Exemple de code pour Raspberry Pi .....	4
6 Fonctions avancées du ADS1115 .....	6

### Photo



### Données techniques / Description sommaire

Ce module à 4 canaux permet de mesurer des tensions analogiques avec une précision de 16 bits. Il est nécessaire d'utiliser des commandes I2C pour le faire fonctionner. Le résultat est sortie codé sur le bus I2C.

**Un logiciel est nécessaire pour utiliser ce module, voir les exemples de code ci-dessous.**

### Brochage

L'affectation des broches est imprimée sur la carte du module.

## KY-053 Convertisseur analogique digital



### Exemple de code pour Arduino

Les cartes Arduino sont équipées d'origine de 6 entrées analogiques 10 bits. L'utilisation du module KY-053 peut s'avérer utile si vous avez besoin d'une précision plus grande (commande via I2C).

Il existe plusieurs façons de contrôler ce module. La société Adafruit a publié des librairies du circuit ADS1X15 à la page [[https://github.com/adafruit/Adafruit\\_ADS1X15](https://github.com/adafruit/Adafruit_ADS1X15)] sous licence [[BSD-Lizenz](#)] veröffentlicht hat.

L'exemple de code ci-dessous utilise cette librairie. Nous recommandons le téléchargement de Github et l'installation après décompression dans le dossier C:\User\[Username]\Documents\Arduino\libraries. Cette librairie est également disponible dans le lien à télécharger ci-dessous.

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>

// Initialisation du module ADS1115. Toutes les opérations avec
// l'ADC peuvent être réalisées à l'aide de l'objet ads
Adafruit_ADS1115 ads;

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

    Serial.println("Lecture des tensions aux bornes A0 à A3 du circuit ADS1115 (A0..A3) en envoi...");
    Serial.println("Plage ADC: +/- 6.144V (1 bit = 0.1875mV)");
}
```

## KY-053 Convertisseur analogique digital

```

// Ce module présente un amplificateur de signal à ses entrées analogiques
// dont le gain peut être configuré comme décrit ci-dessous.
// Ceci est utile lorsque par exemple on s'attend à obtenir un résultat dans une
// certaine plage de mesure mais que le résultat est plus grand que prévu.
// Le gain par défaut est Gain=[2/3] et peut être modifié simplement en enlevant les // de commentaires.
// -----
// ADS1115
// -----
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;

    // La commande "ads.readADC_SingleEnded(0)" est la fonction principale qui fait démarrer la mesure dans l'ADC.
    // Le "0" en tant que variable pour cette fonction définit le canal à mesurer.
    // Si on voulait mesurer sur le troisième canal, on aurait changé le "0" en "2"
    adc0 = ads.readADC_SingleEnded(0);
    adc1 = ads.readADC_SingleEnded(1);
    adc2 = ads.readADC_SingleEnded(2);
    adc3 = ads.readADC_SingleEnded(3);

    // Cette valeur est nécessaire pour la conversion en une tension.
    // La valeur appropriée pour le gain doit être reprise dans la table ci-dessus
    gain_conversion_factor= 0.1875;

    // Conversion des valeurs enregistrées en tension
    voltage0 = (adc0 * gain_conversion_factor);
    voltage1 = (adc1 * gain_conversion_factor);
    voltage2 = (adc2 * gain_conversion_factor);
    voltage3 = (adc3 * gain_conversion_factor);

    // Envoi des valeurs vers l'interface série
    Serial.print("Entrée analogique 0: "); Serial.print(voltage0);Serial.println("mV");
    Serial.print("Entrée analogique 1: "); Serial.print(voltage1);Serial.println("mV");
    Serial.print("Entrée analogique 2: "); Serial.print(voltage2);Serial.println("mV");
    Serial.print("Entrée analogique 3: "); Serial.print(voltage3);Serial.println("mV");
    Serial.println("-----");

    delay(1000);
}

```

### Exemple de programme à télécharger:

[KY-053-AnalogDigitalConverter.zip](#)

### Affectation des broches Arduino:

VDD	= [Pin 5V]
GND	= [Pin GND]
SCL	= [Pin SCL]
SDA	= [Pin SDA]

## KY-053 Convertisseur analogique digital

ADDR	= [N.C.]
ALRT	= [N.C.]
A0	= [Analog 0]
A1	= [Analog 1]
A2	= [Analog 2]
A3	= [Analog 3]

## Exemple de code pour Raspberry Pi

Contrairement à une carte Arduino, le Raspberry Pi ne dispose pas d'entrées analogiques. Cela limite l'utilisation de cette carte à des capteurs numériques uniquement. On ne peut pas utiliser un potentiomètre en entrée, par exemple.

Pour palier à ce problème, notre kit de capteurs est livré avec le module KY-053 qui dispose de 4 entrées analogiques avec une précision de 16 bits. Ce module se raccorde à la carte Raspberry Pi via le bus I2C.

Il existe plusieurs façons de contrôler ce module. La société Adafruit a publié des librairies du circuit ADS1X15 à la page [\[https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code\]](https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code) sous licence [MIT OpenSource](#). Cette librairie est également disponible dans le lien à télécharger ci-dessous.

Le programme lit les tensions sur les 4 canaux du circuit ADS1115 et les affiche dans la console. La variable "delayTime" permet d'instaurer une pause entre les mesures.

Pour que la Raspberry Pi puisse communiquer avec le KY-053 via le bus I2C, celui-ci doit être activé au préalable. Pour ce faire, il faut ajouter la ligne ci-dessous à la fin du fichier "/boot/config.txt":

```
dtparam=i2c_arm=on
```

Le fichier peut être modifié avec la commande suivante:

```
sudo nano /boot/config.txt
```

En utilisant la séquence de touches [Ctrl + X -> Y -> Entrée], le fichier sera enregistré et fermé après avoir ajouté la ligne de commande.

Il est également nécessaire d'utiliser la librairies supplémentaires pour utiliser I2C en Python. Utilisez la commande suivante dans la console pour l'installer:

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

Vous pouvez ensuite utiliser l'exemple de code Python ci-dessous:

```
#  
#!/usr/bin/python  
# coding=utf-8
```

**KY-053 Convertisseur analogique digital**

```
#####
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### KY-053 Analog Digital Converter - Raspberry Pi Python Code Example
#####

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.5 # in Sekunden

# attribution d'adresse ADS1x15 ADC

ADS1015 = 0x00 # 12-bit ADC
ADS1115 = 0x01 # 16-bit

# Choix du gain
gain = 4096 # +/- 4.096V
# gain = 2048 # +/- 2.048V
# gain = 1024 # +/- 1.024V
# gain = 512 # +/- 0.512V
# gain = 256 # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8 # 8 échantillons par seconde
# sps = 16 # 16 échantillons par seconde
# sps = 32 # 32 échantillons par seconde
sps = 64 # 64 échantillons par seconde
# sps = 128 # 128 échantillons par seconde
# sps = 250 # 250 échantillons par seconde
# sps = 475 # 475 échantillons par seconde
# sps = 860 # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel_0 = 0 # Channel 0
adc_channel_1 = 1 # Channel 1
adc_channel_2 = 2 # Channel 2
adc_channel_3 = 3 # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

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

#####
# boucle de programme principale
# #####
# Le programme lit les tensions en entrées et les transmet à la console.

try:
    while True:
```

## KY-053 Convertisseur analogique digital

```

while True:
    #Les valeurs de tension sont enregistrées
    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)

    # Envoi vers la console
    print "Lecture Channel 0:", adc0, "mV "
    print "Lecture Channel 1:", adc1, "mV "
    print "Lecture Channel 2:", adc2, "mV "
    print "Lecture Channel 3:", adc3, "mV "
    print "-----"

    # Reset + Delay
    button_pressed = False
    time.sleep(delayTime)

except KeyboardInterrupt:
    GPIO.cleanup()

```

### Brochage Raspberry Pi:

VDD	= 3,3V	[Pin 01]
GND	= Masse	[Pin 06]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]
ADDR	= N.C.	[ - ]
ALRT	= N.C.	[ - ]
A0	= Analog 0	[pour mesure de tension (capteur par ex.)]
A1	= Analog 1	[pour mesure de tension (capteur par ex.)]
A2	= Analog 2	[pour mesure de tension (capteur par ex.)]
A3	= Analog 3	[pour mesure de tension (capteur par ex.)]

### Exemple de programme à télécharger

[KY-053\\_RPi\\_AnalogDigitalConverter.zip](#)

Commande pour lancer le programme:

```
sudo python KY-053_RPi_AnalogDigitalConverter.py
```

## Fonctions avancées du ADS1115

Le module KY-053 étant basé sur le convertisseur ADS1115, il est possible de réaliser jusqu'à 4 mesures "Single-ended" (mesure des tensions aux entrées indépendamment les unes des autres) ou jusqu'à 2 mesures différentielles (mesure de la différence de tension entre deux entrées).

Lors d'une mesure "Single-ended", la borne est connectée à la masse.

Pour plus d'informations sur les caractéristiques et les fonction, merci de consulter les bibliothèques Adafruit.