SensorKit X40

Dear customer,

thank you for purchasing our product.

This high quality sensor kit was developed especially for the popular Open-Source platforms.
It is compatible to the following single-board computers:

Raspberry Pi (all models), Arduino, Banana PI, Cubieboard, Cubietruck, Beaglebone, pcDuino and many more microcontroller-systems (Atmega, MicroChip PIC, STM32 etc.).

The following instruction not only contains the technical description to every sensor, like the PIN-assignment or the used chipset, but also shows an insight to the functionality of the sensor modules.

To assist you at your own projects, we put a code example, for the most used systems, Raspberry Pi (written in Python) and Arduino (written in C++), to every sensor description. You can find the code directly in this manual or you can download it right beneath the examples. You can also download the most current version of our examples at our SensorKit X40 Wiki:

http://sensorkit.en.joy-it.net/

With our examples, even beginners can easily make their own experiments and projects.

So you have in less than no time the possibility to measure your own heart rate or to check the temperature or humidity of your environment.

Especially for the Raspberry Pi, we put our Analog-Digital converter (KY-053) and our voltage translator to our kit. With these two modules, two of the biggest disadvantages, when it comes to interaction, of the Raspberry Pi (no ADC, 3.3V voltage-level) are resolved.

You can either directly solder the sensors or put them on a breadboard to work on different experiments.

We wish you a lot of joy with your Sensorkit X40.

Your Joy-IT Team

Main Page

SensorKit
X40

Click the picture or the description to go to the site of the specific sensor.

KY-001 Temperature sensor module

KY-002 Vibration-switch module

KY-003 Hall Magneticfield-Sensor module

KY-004 Button-module

KY-005 Infrared Transmitter module

KY-006 Passiv Piezo-Buzzer module

KY-009 RGB LED SMD module

KY-010 Light barrier-module

KY-011 2-Color (Red+Green) 5mm LED module

KY-012 Active Piezo-Buzzer module

KY-013 Temperature-Sensor module

KY-015 Combi-Sensor Temperature+Humidity

KY-016 RGB 5mm LED module

KY-017 Tilt switch module

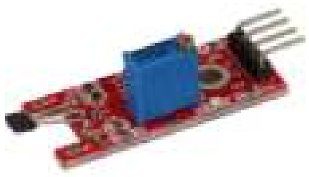
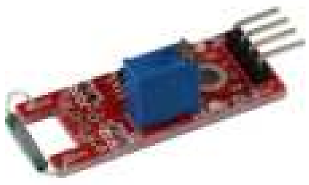
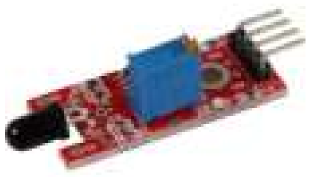
KY-018 Photoresistor module

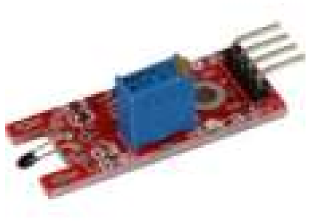
KY-019 5V Relais module

KY-020 Tilt switch module

KY-021 Mini magnetic Reed module

KY-022 Infrared receiver module

	KY-023 Joystick module (XY-Axis)
	**KY-024** Linear magnetic Hall sensor
	**KY-025** Reed module
	**KY-026** Flame-sensor module
	KY-027 Magic light cup module

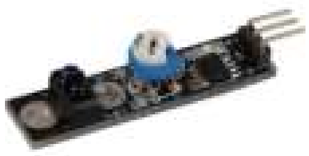


KY-028 Temperature Sensor module (Thermistor)

KY-029 2-Color (Red+Green) 3mm LED module

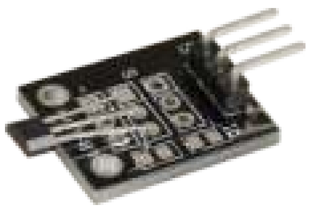
KY-031 Knock-sensor module

KY-032 Obstacle-detect module

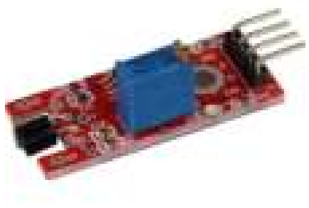


KY-033 Tracking sensor module

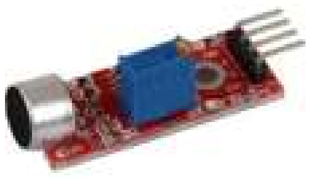
KY-034 7 Colour LED flash-module



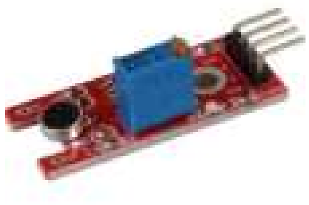
KY-035 Bihor magnetic sensor module



KY-036 Metal-touch sensor module

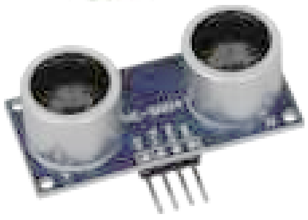
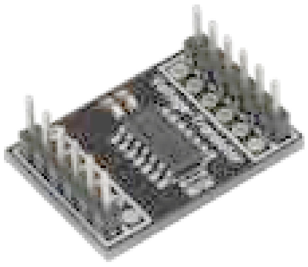
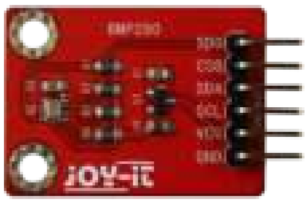


KY-037 Microphone sensor module (high sensitivity)



KY-038 Microphone sound sensor module

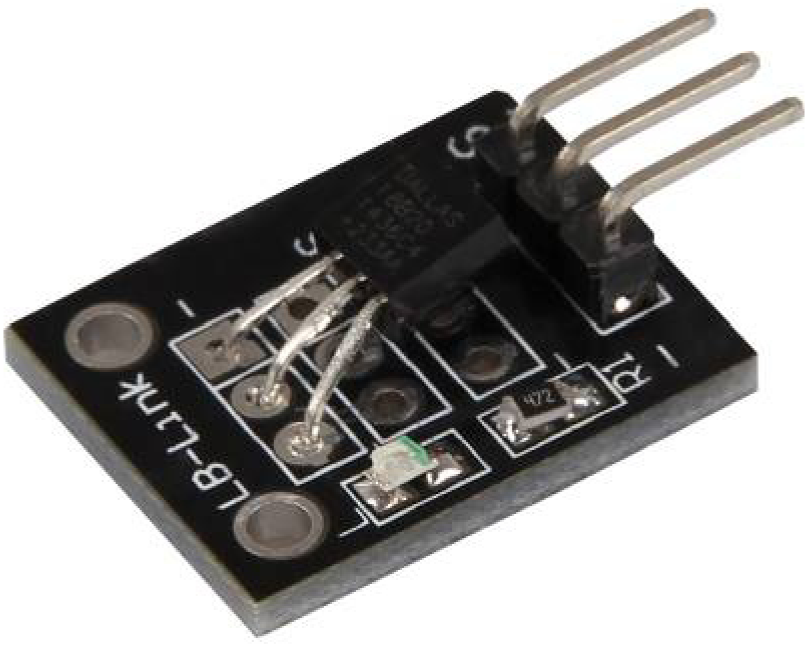
KY-039 Heartbeat sensor module

	KY-040 Rotary encoder
	**KY-050** Ultrasonic-distance-sensor
	**KY-051** Voltage translator / Level shifter
	KY-052 Pressure-sensor / Temperature-sensor (BMP280)
	KY-053 Analog digital converter

KY-001 Temperature sensor module

Contents

Picture

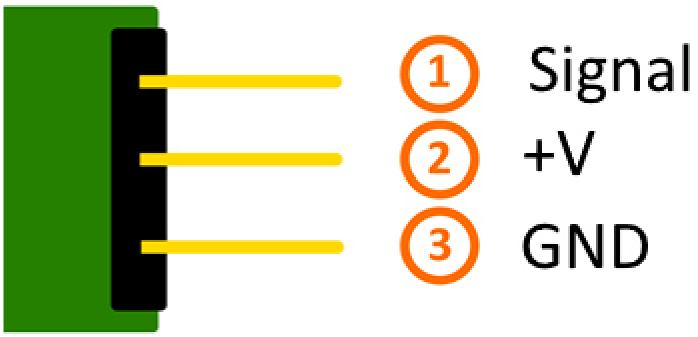


Technical Data / Short description

Chip: DS18B20 | Communication protocol: 1-Wire

9- 12Bit precise temperature measurment between –55°C and +125°C

Pinout



Code example Arduino

You need 2 additional libraries for the following example:

- [OneWire Library] from Paul Stoffregen | published under the MIT license.

- [Dallas Temperature Control Library] from Miles Burton | published under LGPL

Both libraries are part of the package and needs to be copied into the "Library" folder before starting the Arduino IDE.

You can find the path at C:\user\[username]\documents\Arduino\libraries by default.

```
// import needed libraries
#include <DallasTemperature.h>
#include <OneWire.h>

// Declaration of the input pin which is connected with the sensor module
#define KY001_Signal_PIN 4

// libraries configuration
OneWire oneWire(KY001_Signal_PIN);
DallasTemperature sensors(&oneWire);


void setup() {

        // serial output initialization
        Serial.begin(9600);
        Serial.println("KY-001 temperature measurement");

        // sensor will be initialized
        sensors.begin();
}

//main program loop
```

```
void loop()
{
        // temperature measurment will be started...
        sensors.requestTemperatures();
        // ... and measured temperature will be displayed
        Serial.print("Temperature: ");
        Serial.print(sensors.getTempCByIndex(0));
        Serial.write(176); // UniCode of the char-symbol "°-Symbol"
        Serial.println("C");

        delay(1000);    // 1s break till next measurment
}
```

Connections Arduino:

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

Example program download

KY-001-TemperatureSensor

One-Wire configuration for Raspberry Pi

To activate the communication between the Raspberry Pi and the DS18B20 sensor, an additional configuration needs to be made.
You need to modify the „/boot/contig.txt" file and add the following line to it:

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

You can modify the file by entering the following command to the console:

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

You can safe the modification by pressing [CTRL+Y] and leave the editor by pressing [CTRL+X].

At last, you need to reboot your Raspberry Pi with the following command.
If you followed these steps, your system is ready for the example below.

```
sudo reboot
```

Code example Raspberry Pi

```
# coding=utf-8
# needed modules will be imported and initialised
import glob
import time
from time import sleep
import RPi.GPIO as GPIO
```

```
# here you can modify the break between the measurements
sleeptime = 1

# the one-wire input pin will be declared and the integrated pullup-resistor will be enabled
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# After the enabling of the resistor you have to wait till the communication has started
print 'wait for 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'


# The function to read currently measurement at the sensor will be defined.
def TemperaturMessung():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

# To initialise, the sensor will be read "blind"
TemperaturMessung()

# Analysis of temperature: At the Raspberry Pi
# noticed one-wire slaves at the directory /sys/bus/w1/devices/
# will be assigned to a own subfolder.
# In this folder is the file in which the data from the one-wire bus will be saved
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

# main program loop
# The measured temperature will be displayed via console, between the measurements is a break.
# The break time can be configured by the variable "sleeptime"
try:
    while True:
        print '---------------------------------------'
        print "Temperature:", TemperaturAuswertung(), "°C"
        time.sleep(sleeptime)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download:

KY-001_RPi_TemperatureSensor.zip

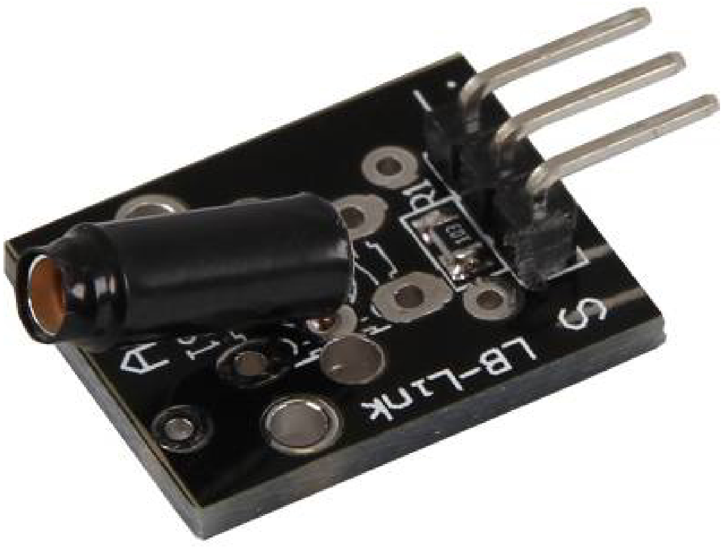
To start the program use the command:

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

KY-002 Vibration-switch module

Contents

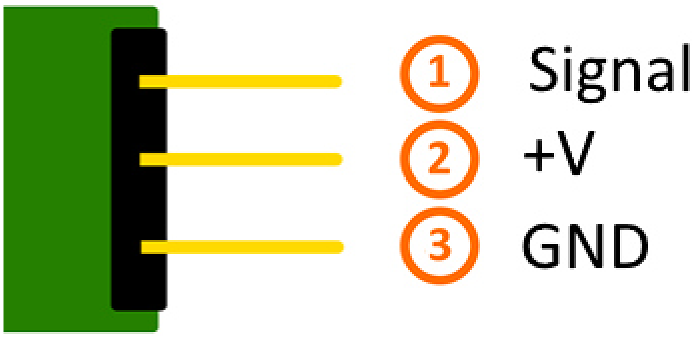
Picture



Technical Data / Short discription

On vibration, the contact of the two input pins will be connected.

Pinout



Code example Arduino

This example will activate a LED as soon as the sensor detects a signal.

The modules KY-011, KY-016 or KY-029 can be used as a LED.

```
int Led = 13 ;// Declaration of the LED output pin
int Sensor = 10; // Declaration of the Sensor input pin
int val; // Temporary variable

void setup ()
{
  pinMode (Led, OUTPUT) ; // Initialisation output pin
  pinMode (Sensor, INPUT) ; // Initializstion sensor pin
  digitalWrite(Sensor, HIGH); // Activating of the internal pull-up resistors
}

void loop ()
{
  val = digitalRead (Sensor) ; // The active signal at the sensor will be read

  if (val == HIGH) // If a signal was noticed, the LED will be on
  {
    digitalWrite (Led, LOW);
  }
  else
  {
    digitalWrite (Led, HIGH);
  }
}
```

Connections Arduino:

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

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

Example program download

SensorTest_Arduino

Code example for Raspberry Pi

```
# needed modules will be imported
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# The input pin of the Sensor will be declared. The pullup resistor will be activated.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print "Sensor-Test [press ctrl+c to end it]"

# This output function will be started at signal detection
def outFunction(null):
        print("Signal detected")

# At the moment of detecting a Signal ( falling signal edge ) the output function will be activated.
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=outFunction, bouncetime=100)

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

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download SensorTest_RPi

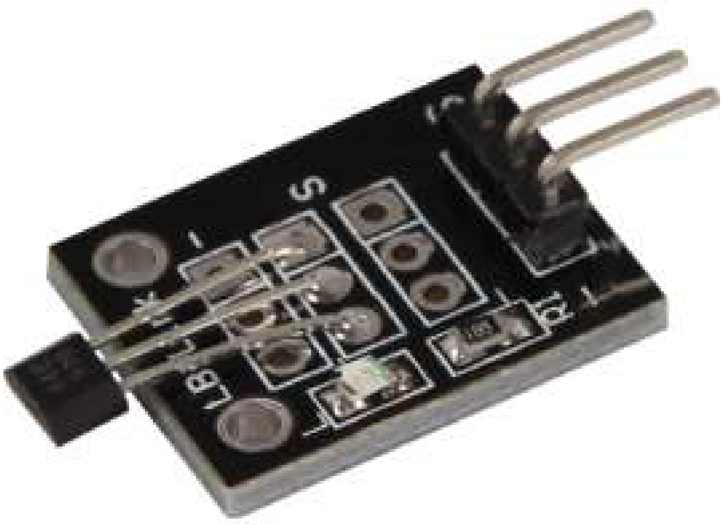
To start use the following command line:

```
sudo python SensorTest_RPi.py
```

KY-003 Hall Magneticfield-Sensor module

Contents

Picture



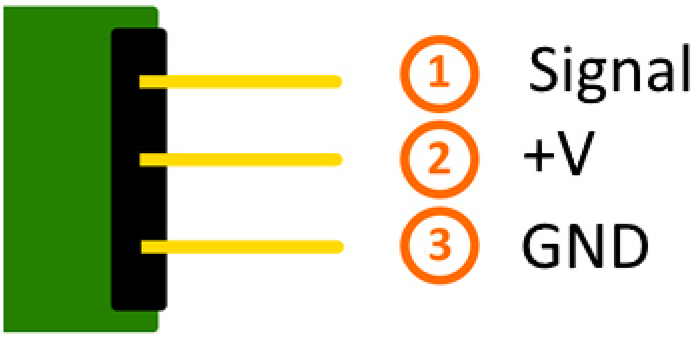
Technical Data / Short description

Chipset: A3141

Sensor type: Hall Effect Transistor/Switch

If the sensor is near a magnetic field, the transistor will switch the circuit.
The result is an analog voltage at the signal output.

Pinout



Code example Arduino

This example will light up a LED as soon as the sensor is near a magnetic field.

The module KY-011, KY-016 or KY-029 can be used as a LED.

```
int Led = 13 ;// Declaration of the LED-output pin
int Sensor = 10; // Declaration of the sensor input pin
int val; // Temporary variable

void setup ()
{
  pinMode (Led, OUTPUT) ; // Initialization output pin
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
  digitalWrite(Sensor, HIGH); // Activating internal pull-up resistor
}

void loop ()
{
  val = digitalRead (Sensor) ; // The current signal at the sensor will be read.

  if (val == HIGH) // If a signal was detected, the LED will light up.
  {
    digitalWrite (Led, LOW);
  }
  else
  {
    digitalWrite (Led, HIGH);
  }
}
```

Connections Arduino:

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

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

Example program download:

SensorTest_Arduino

Code example for Raspberry Pi

```
# needed modules will be imported
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# The input pin of the sensor will be declared. The pull-up resistor will be activated.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print "Sensor-Test [press ctrl+c to end it]"

# This output function will be started at signal detection
def ausgabeFunktion(null):
        print("Signal detected")

# At the moment of detecting a signal the output function will be activated.
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=100)

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

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download:

SensorTest_RPi

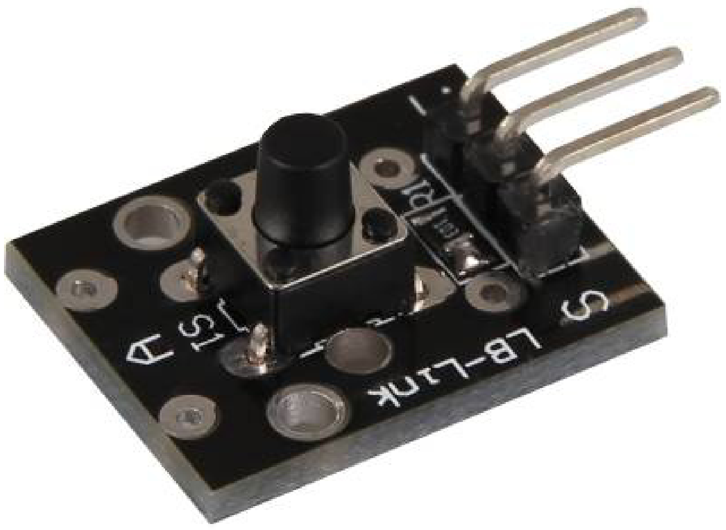
To start use the following command line:

```
sudo python SensorTest_RPi.py
```

KY-004 Button-module

Contents

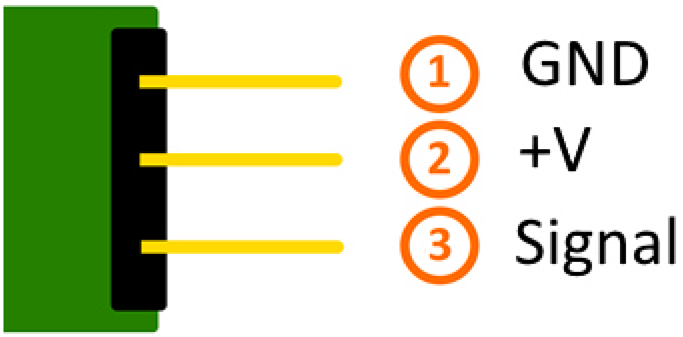
Picture



Technical data / Short description

By pressing the button, the signal circuit is switched.

Pinout



Code example Arduino

This example will light up a LED after the button is pressed.

The module KY-011, KY-016 or KY-029 can be used as a LED.

```
int Led = 13 ;// Declaration of the LED-output pin
int Sensor = 10; // Declaration of the sensor input pin
int val; // Temporary variable

void setup ()
{
  pinMode (Led, OUTPUT) ; // Initialization output pin
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
  digitalWrite(Sensor, HIGH); // Activating internal pull-up resistor
}

void loop ()
{
  val = digitalRead (Sensor) ; // The current signal at the sensor will be read

  if (val == HIGH) // If a signal was detected, the LED will light up.
  {
    digitalWrite (Led, LOW);
  }
  else
  {
    digitalWrite (Led, HIGH);
  }
}
```

Connections Arduino:

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

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

Example program download:

SensorTest_Arduino

Code example Raspberry Pi

```
# needed modules will be imported
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# The input pin of the Sensor will be declared. The pull-up resistor will be activated.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print "Sensor-Test [press ctrl+c to end it]"

# This output function will be started at signal detection.
def ausgabeFunktion(null):
        print("Signal detected")

# At the moment of detecting a Signal the output function will be activated.
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=100)

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

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

SensorTest_RPi

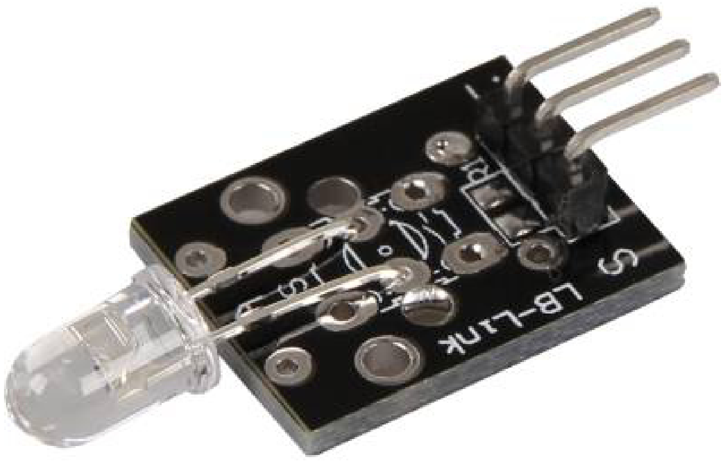
To start, enter the following command:

```
sudo python SensorTest_RPi.py
```

KY-005 Infrared Transmitter module

Contents

Picture



Technical data / Short description

A LED which emits infrared light. A resistor might be necessary for some voltages.

Vf= 1,1V

If= 20mA

emitted wavelength: 940nm *(not visible for the human eye)*

Pre-resistor:

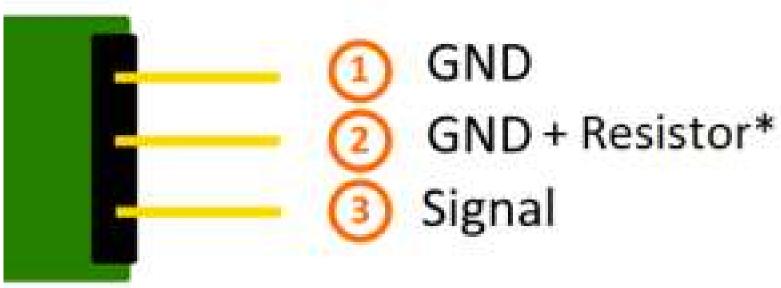
Rf (3,3V) = 120Ω

[used with ARM CPU-Core based microcontroller]

Rf (5V) = 220Ω

[used with Atmel Atmega based microcontroller]

Pinout



- You can directly solder a resistor to the circuit board. In that case, the central pin (2), which includes the resistor, can be used.

Code example Arduino

With both sensor modules, KY-005 and KY-022, you can build an infrared remote + infrared receiver system. In order to do this, you will need the two sensor modules as well as two Arduinos.
The first one will handle the receiver system and the second one will handle the transmitter system.

An additional library is needed for this code example:

-[Arduino-IRremote] from Ken Shirriff | published under LGPL

The library is in the package and has to be copied before the start into the library folder.

You can find your Arduino Library folder at: C:\User\[UserName]\Documents\Arduino\libraries

There are different infrared protocolls to send data. In this example we use the RC5 protocol.
The used library "Arduino-IRremote" converts the data independently.
The library has additional protocolls, they are marked in this documentation.

Code for the receiver:

```
// Arduino-IRremote library will be added
#include <IRremote.h>
#include <IRremoteInt.h>

// You can declare the input pin for the signal output of the KY-022 here
int RECV_PIN = 11;

// Arduino-IRremote library will be initialized
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Infrared receiver will start
}

// main program loop
void loop() {

  // It will be checked if the receiver has gotten a signal.
  if (irrecv.decode(&results)) {
    //At signal input, the received and decoded signal will show via serial console.
    Serial.println(results.value, HEX);
    irrecv.resume();
  }
}
```

Code for the transmitter:

```
//Arduino-IRremote library will be added
#include <IRremote.h>
#include <IRremoteInt.h>

//...and here initialized
IRsend irsend;

// The configuration of the output pin will be made by the library
// The output pin is a different one for different arduinos
// Arduino UNO:  Output = D3
// Arduino MEGA: Output = D9
// You will find a full list of output pins on the website:
// http://z3t0.github.io/Arduino-IRremote/
void setup()
{
}

// main program loop
void loop() {
        // The transmitter sends the signal A90 (hex. dezimal form) in the encoding "RC5"
        // It will be transmitted 3 times after that it will make a 5 second break
    for (int i = 0; i < 3; i++) {
        irsend.sendRC5(0xA90, 12); //[12] Bit-length signal (hex A90=1010 1001 0000)
        delay(40);
    }
    delay(5000); // 5 second break between the sending impulses
}
```

Example program download:

KY-005_KY-022_Infrared-Modules_ARD

Connections Arduino 1 [Receiver]:

KY-022

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

Connections Arduino 2 [Transmitter]:

KY-005

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

- * Only if resistor was soldered to the circuit board.

Code example Raspberry Pi

Code example remote

Because of its progressive processor architecture, the Raspberry Pi has a big advantage, compared to the Arduino.

It can run a full Linux OS.
With help of an infrared-receiver, it is not only able to transmit simple data signals, furthermore it can control complete programs via remote.

To setup an infrared control system, we recommend to use the Linux software "lirc" (published under the LGPL-Website).
In the following section, we show you how to use lirc and how the remotely send the learned signals via infrared.

On this purpose, the module KY-005 will be used as an infrared-transmitter and the KY-022 will be used as an infrared-receiver.

Connections Raspberry Pi:

KY-005

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

- * Only if a resistor was soldered to the module

KY-022

Signal	=	GPI18	[Pin 12]
+V	=	3,3V	[Pin 17]
GND	=	GND	[Pin 25]

Lirc Installation

Open a terminal at the desktop or use SSH to log into your Raspberry Pi. To install lirc, enter the following command:

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

[For this the Raspberry Pi has to be connected to the internet]

To use the lirc module immediately after starting the OS, you have to add the following line to the end of the file "/boot/config.txt":

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

The "gpio_in_pin=18" will be defined as an input pin of the IR-receiver and the "gpio_out_pin=17" as an output pin of the IR-transmitter.

The file can be edited by entering the command:

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

You can save and close the file via the key sequence [ctrl+x -> y -> enter]

You will also have to modify the file "/etc/lirc/hardware.conf" by entering the command:

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

In this file you have to change following lines:

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

The modified file should now look like:

```
# /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.
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=""
```

After that we reboot the Raspberry Pi with the following command:

```
sudo reboot
```

IR-Receiver Test

To test the connected receiver, you have to close lirc first with the following command:

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

After that, you can test if signals could be detected on the Raspberry Pi by using the following command:

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

and by pressing random keys on an infrared remote. You should see numbers in the following form:

```
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
...
```

You can restart lirc with the following command:

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

Remote teach

To register an infrared-remote at the lirc system, you have to configure the file "/etc/lirc"lircd.conf".

In this file, all command assignments of the infrared codes are saved.

To get a good formatted lircd.conf, use the lirc assistant software which creates the file automatically.

To start this process you have to stop lirc first by using the command:

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

With the following command, we can start the assistant:

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

The assistant will start an initialization of the remote, in this initialization you have to press a few keys so that the lirc system is able to learn the encoding of the remote. For that, please follow the instructions of the assistant. After the initialization, the assistant asks for the name of the key which should get a new infrared code. You can choose your key from the following file:

FernbedienungsCodes.txt

You have to type these into the assistant and need to confirm with enter. After this, the recording of the infrared code for the chosen key will start.

Example: type in [KEY_0] - - -> confirm with enter - - -> press key 0 of the remote - - -> waiting for the assistant to confirm the recording.

If no more keys need to be configured, you can close the assistant by pressing the enter key. After this, the configuration file is created, but you have to choose a name for the recorded remote. For this we have to open the file with the editor:

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

Here you have to change the line:

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

to

```
     name  MeineFernbedienung
```

Please don't use any spaces or additional characters in the name.

You can save and close the file with the key sequence [ctrl+x ---> y ---> enter].

After creating the configuration, you can make a backup for original lircd.conf with the following command:

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

With the command

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

you can use the before created file for the lirc system.

Now you can start the lirc system again with the command:

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

From now on, the remote is known and can be used with the right software. Alternatively you can use the following command to test the functions:

```
irw
```

Sending a command via Infrared Transmitter

If you want to control devices, like your Television, via Raspberry Pi, you can now send the learned commands with the infrared transmitter. With that you can build a software controlled infrared controller or you can use the internet or the network to switch single devices on and off.

First we check with the following command:

```
irsend LIST MeineFernbedienung ""
```

which assigments are available for the remote.

Now we can send the command [KEY_0] with the command:

```
irsend SEND_ONCE MeineFernbedienung KEY_0
```

You can have the example above in other variations like , instead of sending the signal only once , it will be send multiple times.

```
irsend SEND_START MeineFernbedienung KEY_0
```

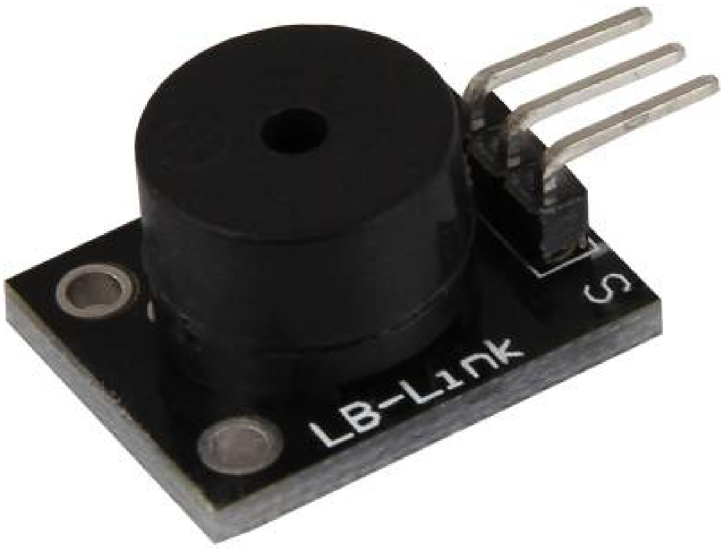
After this, the code [KEY_0] will be repeatly send out until we end it with the following command:

```
irsend SEND_STOP MeineFernbedienung KEY_0
```

KY-006 Passiv Piezo-Buzzer module

Contents

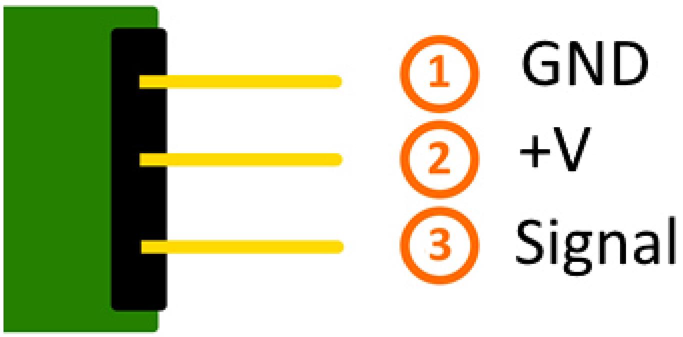
Picture



Technical data / Short description

PWM-Signals of different frequencies can be used to create different sounds from the Piezo-Buzzer.

Pinout



Code example Arduino

This is an example program which will start an alarm signal on the buzzer via square wave voltage.

```
int buzzer = 8 ; // Declaration of the buzzer-output pin

void setup ()
{
  pinMode (buzzer, OUTPUT) ;// Initialization of the output pin.
}


void loop ()
{
  unsigned char i;
  while (1)
  {
    // The buzzer will be controlled by 2 different frequencies in this program.
    // The signal is a square wave signal.
    // The on and off of the buzzer will generate a sound which is nearly the sound of the frequency.
    // The frequency will be defined from the time of the on and off period.

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

Connections Arduino:

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

Example program download

KY-006_Buzzer

Code example Raspberry Pi

This example program uses software-PWM, to generate a square wave with defined frequency.

The buzzer will generate a sound which is nearly the sound of the square wave frequency.

```
#Needed modules will be imported and configured.
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

#The output pin, which is connected with the buzzer, will be declared here.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.OUT)
#The software-PWM module will be initialized - a frequency of 500Hz will be taken as default.
Frequenz = 500 #In Hertz
pwm = GPIO.PWM(GPIO_PIN, Frequenz)
pwm.start(50)
# The program will wait for the input of a new PWM-frequency from the user.
# Until then, the buzzer will be used with the before inputted frequency (default 500Hz).
try:
        while(True):
                print "----------------------------------------"
                print "Current frequency: %d" % Frequenz
                Frequenz = input("Please input a new frequency (50-5000):")
                pwm.ChangeFrequency(Frequenz)

# Scavenging work after the end of the program.
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

KY-006-RPI_PWM

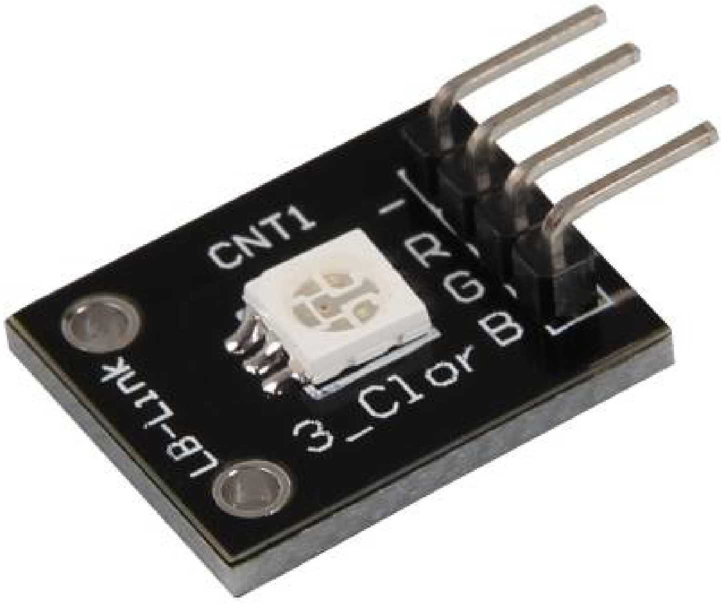
To start,enter the command:

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

KY-009 RGB LED SMD module

Contents

Picture



Technical data / Short description

A LED-module which provides a red, blue and green LED. These are connected with a common cathode. A resistor is necessary for different voltages.

Vf [Red]= 1,8V

Vf [Green,Blue]= 2,8V

If= 20mA

Pre-resistor:

Rf (3,3V) [Green]= 100Ω

Rf (3,3V) [Red]= 180Ω

Rf (3,3V) [Blue]= 100Ω

[for example using of ARM CPU-Core based microcontroller like Raspberry-Pi]

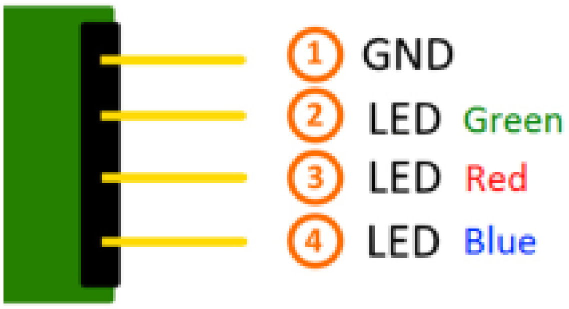
Rf (5V) [Green] = 100Ω

Rf (5V) [Red] = 180Ω

Rf (5V) [Blue] = 100Ω

[for example using of Atmel Atmega based microcontroller like Arduino]

Pinout



Code example Arduino

Code example ON/OFF

In this example you will see how the LEDs will be switched on with a defined output pin, in a 3 second clock pulse.

```
int Led_Red = 10;
int Led_Green = 11;
int Led_Blue = 12;

void setup ()
{
  // Output pin initialization for the LEDs
  pinMode (Led_Red, OUTPUT);
  pinMode (Led_Green, OUTPUT);
  pinMode (Led_Blue, OUTPUT);
}

void loop () //main program loop
{
  digitalWrite (Led_Red, HIGH);   // LED will be switched on
  digitalWrite (Led_Green, LOW);  // LED will be switched off
  digitalWrite (Led_Blue, LOW);   // LED will be switched off
  delay (3000); // Waitmode for 3 seconds

  digitalWrite (Led_Red, LOW);    // LED will be switched off
  digitalWrite (Led_Green, HIGH); // LED will be switched on
  digitalWrite (Led_Blue, LOW);   // LED will be switched off
  delay (3000); // Waitmode for another 3 seconds in which the LED status will be shifted.

  digitalWrite (Led_Red, LOW);    // LED will be switched off
  digitalWrite (Led_Green, LOW);  // LED will be switched off
  digitalWrite (Led_Blue, HIGH);  // LED will be switched on
  delay (3000); // Waitmode for another 3 seconds in which the LED status will be shifted.
}
```

Example program ON/OFF download:

KY-009_LED_ON-OFF

Code example PWM

You can regulate the brightness of the LEDs via pulse-width modulation. The LEDs will be switched ON and OFF of for specific time periods, in which the relation between ON and OFF leads to a relative brightness, because of the Inertia of the human eyesight, the human eye interprets the ON/OFF as a brightness change. For more information to that theme visit: [Artikel von mikrokontroller.net]

This module provides a few LEDs - with the overlay of the different brightness levels, you can create different colors. This will be shown in the following code example.

```
int Led_Red = 10;
int Led_Green = 11;
int Led_Blue = 12;

int val;

void setup () {
  //Output pin initialization for the LEDs
  pinMode (Led_Red, OUTPUT);
  pinMode (Led_Green, OUTPUT);
  pinMode (Led_Blue, OUTPUT);
}
void loop () {
   // In this for-loop, the 3 LEDs will get different PWM-values
   // Via mixing the brightness of the different LEDs, you will get different colors.
   for (val = 255; val> 0; val--)
      {
       analogWrite (Led_Blue, val);
       analogWrite (Led_Green, 255-val);
       analogWrite (Led_Red, 128-val);
       delay (1);
   }
   // You will go backwards through the color range in this second for loop.
   for (val = 0; val <255; val++)
      {
      analogWrite (Led_Blue, val);
      analogWrite (Led_Green, 255-val);
      analogWrite (Led_Red, 128-val);
      delay (1);
   }
}
```

Example program PWM download:

KY-009_LED_PWM

Connections Arduino:

LED Red	=	[Pin 10]
LED Green	=	[Pin 11]
LED Blue	=	[Pin 12]
Sensor GND	=	[Pin GND]

Code example Raspberry Pi

Code example ON/OFF

In this example you will see how the LEDs will be switched on with a defined output pin, in a 3 second clock pulse.

```
# Needed modules will be imported and configured.
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# The output pins will be declared, which are connected with the LEDs.
LED_Red = 6
LED_Green = 5
LED_Blue = 4

GPIO.setup(LED_Red, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_Green, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_Blue, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [press ctrl+c to end]"

# main program loop
try:
        while True:
                        print("LED Red is on for 3 seconds")
                        GPIO.output(LED_Red,GPIO.HIGH) #LED will be switched on
                        GPIO.output(LED_Green,GPIO.LOW) #LED will be switched off
                        GPIO.output(LED_Blue,GPIO.LOW) #LED will be switched off
                        time.sleep(3) # Waitmode for 3 seconds
                        print("LED Green is on for 3 seconds")
                        GPIO.output(LED_Red,GPIO.LOW) #LED will be switched off
                        GPIO.output(LED_Green,GPIO.HIGH) #LED will be switched on
                        GPIO.output(LED_Blue,GPIO.LOW) #LED will be switched off
                        time.sleep(3) #Waitmode for 3 seconds
                        print("LED Blue is on for 3 seconds")
                        GPIO.output(LED_Red,GPIO.LOW) #LED will be switched off
                        GPIO.output(LED_Green,GPIO.LOW) #LED will be switched off
                        GPIO.output(LED_Blue,GPIO.HIGH) #LED will be switched on
                        time.sleep(3) #Waitmode for 3 seconds

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Example program ON/OFF download

KY009_RPi_ON-OFF

To start, enter the command:

```
sudo python KY009_RPI_ON-OFF.py
```

Code example PWM

You can regulate the brightness of the LEDs via pulse-width modulation. The LEDs will be switched ON and OFF for specific time periods, in which the relation between ON and OFF leads to a relative brightness, because of the Inertia of the human eyesight, the human eye interprets the ON/OFF as a brightness change. For more information to that theme visit: [Artikel von mikrokontroller.net]

This module provides a few LEDs - with the overlay of the different brightness levels, you can create different colors. This will be shown in the following code example. At the Raspberry Pi, only one Hardware-PWM channel is carried out unrestricted to the GPIO pins, that's why we have used Software-PWM on this example.

```
# Needed modules will be imported and configured.
import random, time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
 <br />// Declaration of the output pins, which are connected with the LEDs.
LED_Red = 6
LED_Green = 5
LED_Blue = 4

# Set pins to output mode
GPIO.setup(LED_Red, GPIO.OUT)
GPIO.setup(LED_Green, GPIO.OUT)
GPIO.setup(LED_Blue, GPIO.OUT)

Freq = 100 #Hz
# The different colors will be initialized.
RED = GPIO.PWM(LED_Red, Freq)
GREEN = GPIO.PWM(LED_Green, Freq)
BLUE = GPIO.PWM(LED_Blue, Freq)
RED.start(0)
GREEN.start(0)
BLUE.start(0)
# This function generates the actually color

def LED_color(Red, Green, Blue, pause):
    RED.ChangeDutyCycle(Red)
    GREEN.ChangeDutyCycle(Green)
    BLUE.ChangeDutyCycle(Blue)
    time.sleep(pause)

    RED.ChangeDutyCycle(0)
    GREEN.ChangeDutyCycle(0)

print "LED-Test [press ctrl+c to end the test]"

# Main program loop:
# The task of this loop is to create for every single color an own variable.
# By mixing the brightness levels of the colors, you will get a color gradient.
```

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

# Scavenging work after the end of the program.
except KeyboardInterrupt:
        GPIO.cleanup()
```

Example program PWM download:

KY-009_RPi_PWM

To start, enter the command:

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

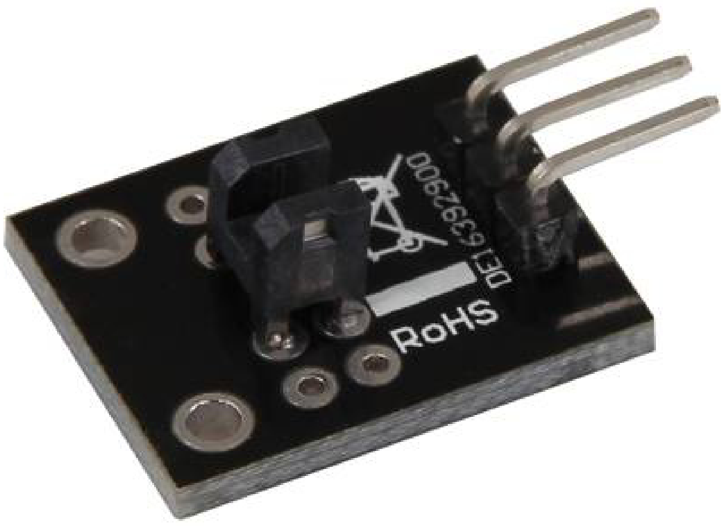
Connections Raspberry Pi:

LED Red	= GPIO6	[Pin 22]
LED Green	= GPIO5	[Pin 18]
LED Blue	= GPIO4	[Pin 16]
Sensor GND	= GND	[Pin 6]

KY-010 Light barrier-module

Contents

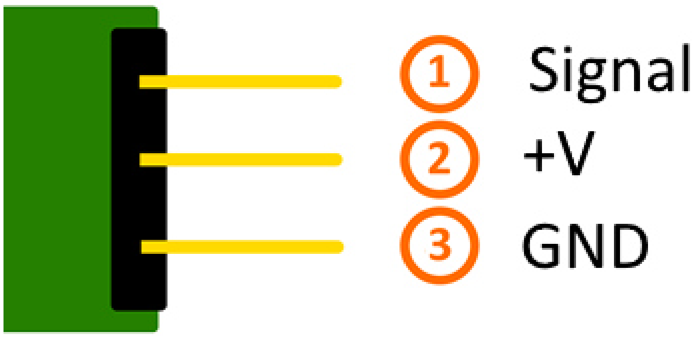
Picture



Technical data / Short description

The connection between both input pins will be interrupted if the optical barrier is beeing interrupted.

Pinout



Code example Arduino

In this program, a LED will flash up, if a signal was detected at the sensor. You can also use the modules KY-011, KY-016 or KY-029 as LEDs.

```
int Led = 13 ;// Declaration of the LED-output pin
int Sensor = 10; // Declaration of the Sensor-input pin
int val; // Temporary variable

void setup ()
{
  pinMode (Led, OUTPUT) ; // Initialization output pin
  pinMode (Sensor, INPUT) ; // Initialization sensorpin
}

void loop ()
{
  val = digitalRead (Sensor) ; // The current signal at the sensor will be read.

  if (val == HIGH) //The led will flash up, if a signal was detected.
  {
    digitalWrite (Led, HIGH);
  }
  else
  {
    digitalWrite (Led, LOW);
  }
}
```

Connections Arduino:

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

Example program download

SensorTest_Arduino_inverted

Code example Raspberry Pi

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# The input pin which is connected with the sensor.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

print "Sensor-Test [press ctrl+c to end the test]"

# This outputFunction will be started at signal detection
def outputFunction(null):
        print("Signal detected")

# The outputFunction will be started at the moment of a signal detection (raising edge).
GPIO.add_event_detect(GPIO_PIN, GPIO.RISING, callback=outputFunction, bouncetime=100)

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

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

SensorTest_RPi_inverted

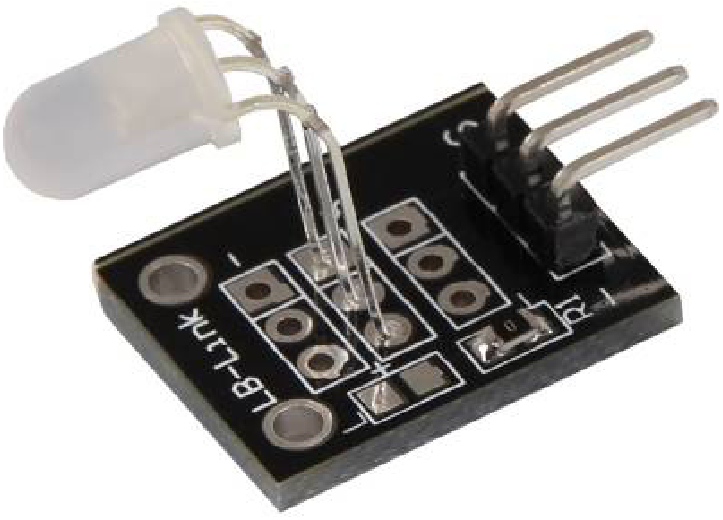
To start, enter the command:

```
sudo python SensorTest_RPi_inverted.py
```

KY-011 2-Color (Red+Green) 5mm LED module

Contents

Picture



Technical data / Short description

LED module which provides a red and a green LED. These LEDs are connected with a common cathode.

Resistors are needed for different input voltages.

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

If= 20mA

Pre-resistors:

Rf (3,3V) [Green]= 120Ω

Rf (3,3V) [Red]= 120Ω

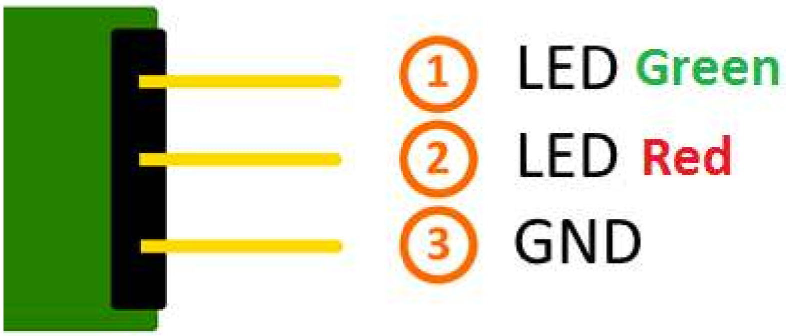
[for example using ARM CPU-Core based microcontroller like Raspbarry Pi]

Rf (5V) [Green] = 220Ω

Rf (5V) [Red] = 220Ω

[for example using Atmel Atmega based microcontroller like Arduino]

Pinout



Code example Arduino

Code example ON/OFF

```
int Led_Red = 10;
int Led_Green = 11;

void setup ()
{
  // Output pin initialization for the LEDs
  pinMode (Led_Red, OUTPUT);
  pinMode (Led_Green, OUTPUT);
}

void loop () //Main program loop
{
  digitalWrite (Led_Red, HIGH); // LED will be switched on
  digitalWrite (Led_Green, LOW); // LED will be switched off
  delay (3000); // Waitmode for 3 seconds

  digitalWrite (Led_Red, LOW); // LED will be switched off
  digitalWrite (Led_Green, HIGH); // LED will be switched on
  delay (3000); // Waitmode for another 3 seconds in which the status of the LEDs are shifted.
}
```

Example program ON/OFF download:

Example program ON/OFF download:

KY-011_LED_ON-OFF

Code example PWM

You can regulate the brightness of the LEDs via pulse-width modulation. The LEDs will be switched ON and OFF for specific time periods, in which the relation between ON and OFF leads to a relative brightness, because of the Inertia of the human eyesight, the human eye interprets the ON/OFF as a brightness change. For more information to that theme visit: [Artikel von mikrokontroller.net]

This module provides a few LEDs - with the overlay of the different brightness levels, you can create different colors. This will be shown in the following code example.

```
int Led_Red = 10;
int Led_Green = 11;

int val;

void setup () {
  // Output pin initialization for the LEDs
  pinMode (Led_Red, OUTPUT);
  pinMode (Led_Green, OUTPUT);
}
void loop () {
   // In this for loop, the two LEDs will get different PWM-Values.
   // Via mixing the brightness of the different LEDs, you will get different colors.
   for (val = 255; val> 0; val--)
      {
      analogWrite (Led_Green, val);
      analogWrite (Led_Red, 255-val);
      delay (15);
   }
   // You will go backwards through the color range in this second loop.
   for (val = 0; val <255; val++)
      {
      analogWrite (Led_Green, val);
      analogWrite (Led_Red, 255-val);
      delay (15);
   }
}
```

Example program PWM download: KY-011_PWM

Connections Arduino:

LED Green	= [Pin 10]
LED Red	= [Pin 11]
Sensor GND	= [Pin GND]

Code example Raspberry Pi

Code example ON/OFF

```
# Needed modules will be imported and configured.
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Output pin declaration for the LEDs.
LED_Red = 5
LED_Green = 4
GPIO.setup(LED_Red, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_Green, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [press ctrl+c to end the test]"

# Main program loop
try:
        while True:
                        print("LED Red will be on for 3 seconds")
                        GPIO.output(LED_Red,GPIO.HIGH) #LED will be switched on
                        GPIO.output(LED_Green,GPIO.LOW) #LED will be switched off
                        time.sleep(3) # Waitmode for 3 seconds
                        print("LED Green will be on for 3 seconds")
                        GPIO.output(LED_Red,GPIO.LOW) #LED will be switched off
                        GPIO.output(LED_Green,GPIO.HIGH) #LED will be switched on
                        time.sleep(3) #Waitmode for 3 seconds in which the LEDs are shifted

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Example program ON/OFF download

KY011_RPI_ON-OFF

To start, enter the command:

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

Code example PWM

You can regulate the brightness of the LEDs via pulse-width modulation. The LEDs will be switched ON and OFF of for specific time periods, in which the relation between ON and OFF leads to a relative brightness, because of the Inertia of the human eyesight, the human eye interprets the ON/OFF as a brightness change. For more information to that theme visit: [Artikel von mikrokontroller.net]

This module provides a few LEDs - with the overlay of the different brightness levels, you can create different colors. This will be shown in the following code example. At the Raspberry Pi, only one Hardware-PWM channel is carried out unrestricted to the GPIO pins, why we have used Software-PWM at this example

```
# Needed modules will be imported and configured
import random, time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

# Output pin declaration for the LEDs.
LED_Red = 5
LED_Green = 4

# Set pins to output mode
GPIO.setup(LED_Red, GPIO.OUT)
GPIO.setup(LED_Red, GPIO.OUT)
GPIO.setup(LED_Green, GPIO.OUT)

Freq = 100 #Hz

# The specific colors will be initialized.
RED = GPIO.PWM(LED_Red, Freq)
GREEN = GPIO.PWM(LED_Green, Freq)
RED.start(0)
GREEN.start(0)

# This function generate the actually color
# You can change the color with the specific color variable.
# After the configuration of the color is finished, you will time.sleep to
# configure how long the specific will be displayed.

def LED_color(Red, Green, pause):
    RED.ChangeDutyCycle(Red)
    GREEN.ChangeDutyCycle(Green)
    time.sleep(pause)

    RED.ChangeDutyCycle(0)
    GREEN.ChangeDutyCycle(0)

print "LED-Test [press ctrl+c to end the test]"

# Main program loop:
# The task of this loop is to create for every single color an own variable.
# By mixing the brightness levels of the colors, you will get a color gradient.
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_color((x*i),(y*i),.02)

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Example program PWM download:

Media:KY011_RPI_PWM.zip

To start, enter the command:

```
sudo python KY011_RPI_PWM.py
```

Connections Raspberry Pi:

LED Green	= GPIO4	[Pin 16]
LED Red	= GPIO5	[Pin 18]
Sensor GND	= GND	[Pin 6]

KY-012 Active Piezo-Buzzer module

Contents

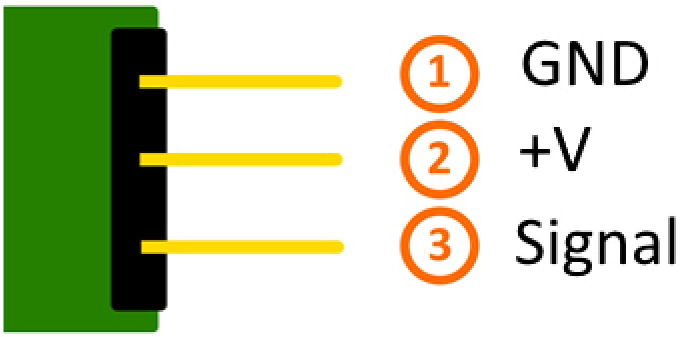
Pictures

Technical data / Short description

This Buzzer creates a sound with a frequency of 2,5kHz.

The active Buzzer-module doesn't need a square wave, unlike the passiv module (KY-006), to create a sound. If it gets a minimum Voltage of 3.3V at its signal pin, the buzzer will create the square wave by itself.

Pinout



Code example Arduino

In this example, you will see how the buzzer will be ON for 4 seconds and then will be OFF for 2 seconds.

```
int Buzzer = 13;

void setup ()
{
  pinMode (Buzzer, OUTPUT); // Output pin initialization for the buzzer
}

void loop () //Main program loop
{
  digitalWrite (Buzzer, HIGH); // Buzzer will be on
  delay (4000); // Waitmode for 4 seconds
  digitalWrite (Buzzer, LOW); // Buzzer will be off
  delay (2000); // Waitmode for another 2 seconds in which the buzzer will be off
}
```

Connections Arduino:

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

Example program download:

KY-006-RPI_PWM

Code example Raspberry Pi

In this example, you will see how, with a defined output pin, the buzzer will be ON for 4 seconds and then will be OFF for 2 seconds.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Output pin declaration for the Buzzer.
Buzzer_PIN = 24
GPIO.setup(Buzzer_PIN, GPIO.OUT, initial= GPIO.LOW)

print ("Buzzer-test [press ctrl+c to end the test]")

# Main program loop
try:
        while True:
            print("Buzzer will be on for 4 seconds")
            GPIO.output(Buzzer_PIN,GPIO.HIGH) #Buzzer will be switched on
            time.sleep(4) #Waitmode for 4 seconds
            print("Buzzer wil be off for 4 seconds")
            GPIO.output(Buzzer_PIN,GPIO.LOW) #Buzzer will be switched off
            time.sleep(2) #Waitmode for another 2 seconds in which the buzzer will be off

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor Signal	=	GPIO24	[Pin 18]
Sensor [+V]	=	3.3V	[Pin 1]
Sensor GND	=	GND	[Pin 6]

Example program download

KY-012_Buzzer_RPi

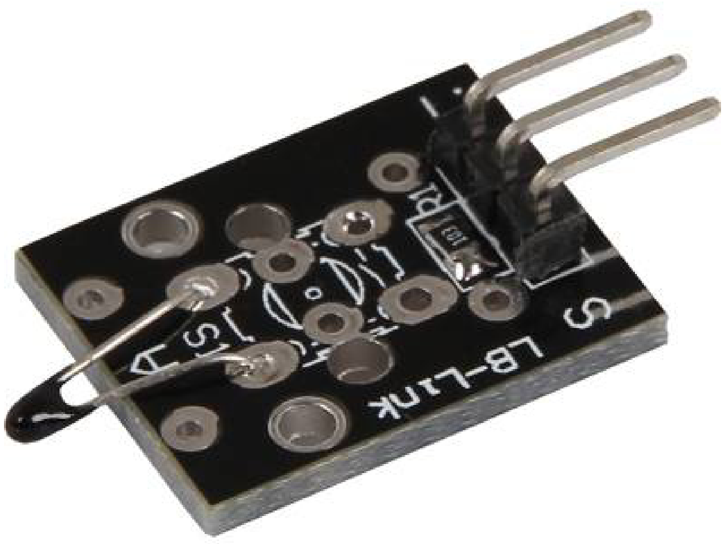
To start, enter the command:

```
sudo python KY-012_Buzzer_RPi.py
```

KY-013 Temperature-Sensor module

Contents

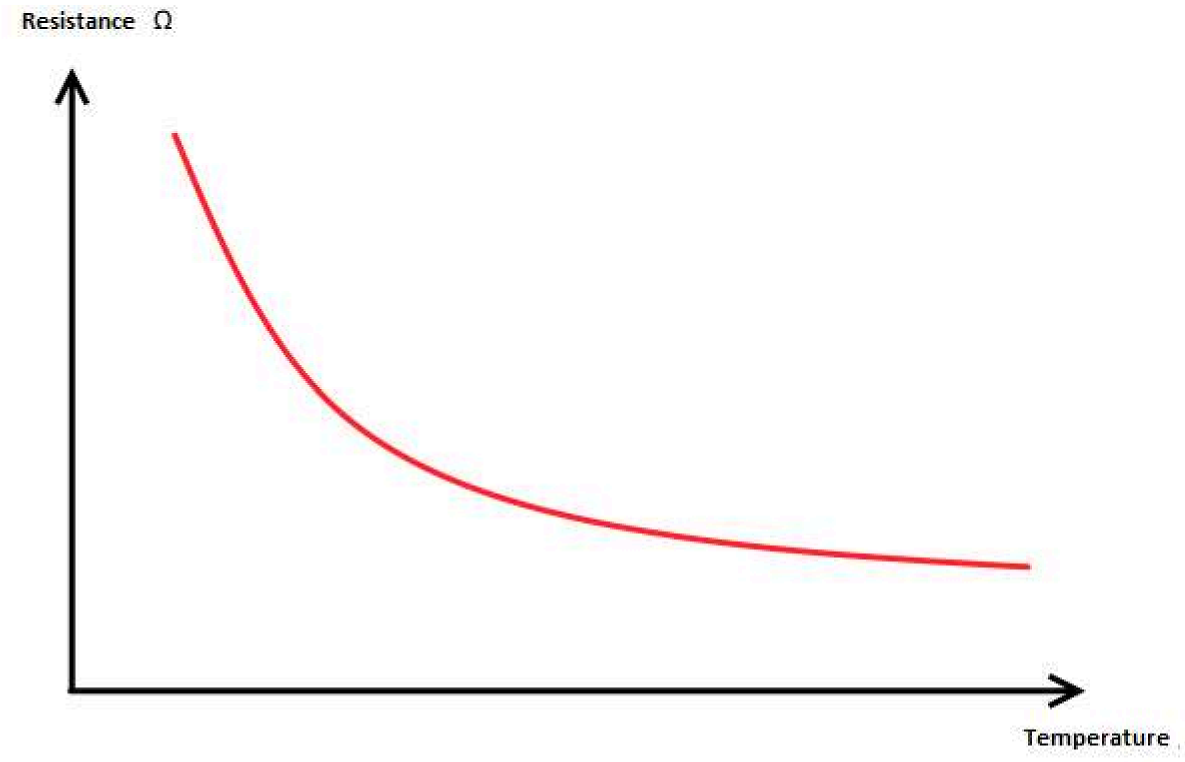
Picture



Technical data / Short description

Temperature measuring range: -55°C / +125°C

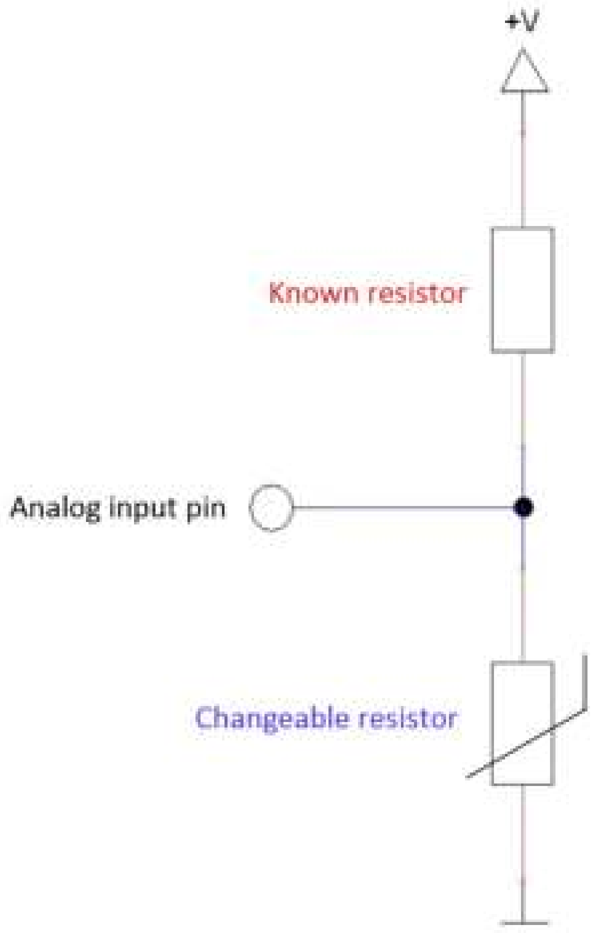
This module provides a NTC thermistor - it will have a lower resistant on higher temperatures.



You can draw near to the resistant change via maths and convert it into a linear course. With that you can determine the temperature coefficient (addicted from resistant change to temperature change). With that you can determine the actual temperature if you know the current resistance.

This resistor can be determinded via voltage devider, where a known voltage splits up between a known resistor and an unknown (variable) resistor.

With that Voltage you can determine the resistance of the resistor - you can see the full calculation in the example below.



Pinout

Code example Arduino

The program measures the actual voltage from the NTC, calculate the temperature and translates the result to °C for the serial output.

```
#include <math.h>

int sensorPin = A5; // Declaration of the input pin

// These function translates the recorded analog measurement
double Thermistor(int RawADC)
{
        double Temp;
        Temp = log(10000.0 * ((1024.0 / RawADC - 1)));
        Temp = 1 /(0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp)) * Temp);
        Temp = Temp - 273.15;            // convert from Kelvin to Celsius
        return Temp;
}

// Serial output in 9600 Baud

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

// The program measures the current voltage value on the NTC
// and translates it intp °C for the serial output
void loop()
{
        int readVal = analogRead(sensorPin);
        double temp =  Thermistor(readVal);

        // Output on the serial interface
        Serial.print("Current temperature is:");
        Serial.print(temp);
        Serial.print(char(186)); //Output <°> Symbol
        Serial.println("C");
        Serial.println("---------------------------------------");

        delay(500);
}
```

Connections Arduino:

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

Example program Download

KY-013_TemperatureSensor

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspberry Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

```
### 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
###




# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import math, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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

# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

# temperature calculation function
def calcTemp(voltage):
        temperature = math.log((10000/voltage)*(3300-voltage))
       temp = (0.0000000876741 * temperature * temperature)
        temperature = 1 / (0.001129148 + (0.000234125 + temp) * temperature);
        temperature = temperature - 273.15;
        return temperature
```

```
# ########
# Main Loop
# ########
# Reading the values from the input pins and print to console

try:
        while True:
                #read voltage-value and calculate temperature
                temp0 = round(calcTemp(adc.readADCSingleEnded(adc_channel_0, gain, sps)), 2)
                temp1 = round(calcTemp(adc.readADCSingleEnded(adc_channel_1, gain, sps)), 2)
                temp2 = round(calcTemp(adc.readADCSingleEnded(adc_channel_2, gain, sps)), 2)
                temp3 = round(calcTemp(adc.readADCSingleEnded(adc_channel_3, gain, sps)), 2)


                # print to console
                print "Channel 0:", temp0, "C"
                print "Channel 1:", temp1, "C"
                print "Channel 2:", temp2, "C"
                print "Channel 3:", temp3, "C"
                print "---------------------------------------"

                sleep(delayTime)



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor

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

ADS1115 - KY-053:

VDD	= 3,3V	[Pin 17]
GND	= GND	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]
A0	= s.o.	[Sensor: analog Signal]

Example program download

KY-013_Temperature-Sensor_RPi

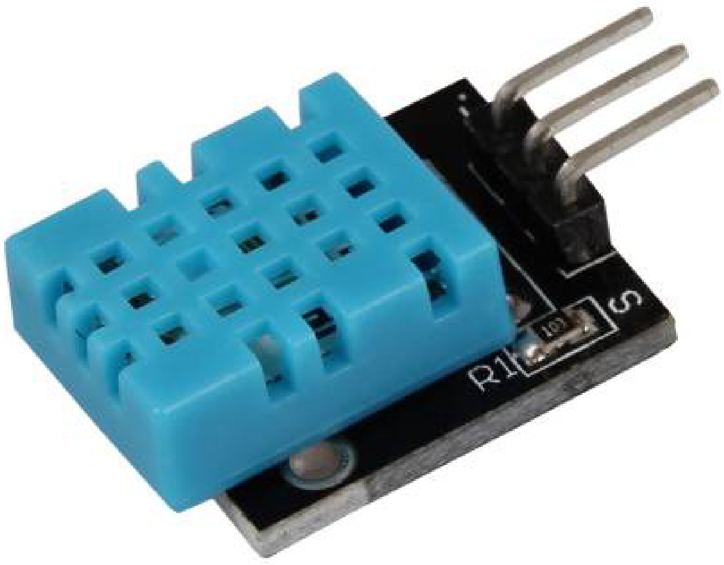
To start, enter the command:

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

KY-015 Combi-Sensor Temperature+Humidity

Contents

Picture

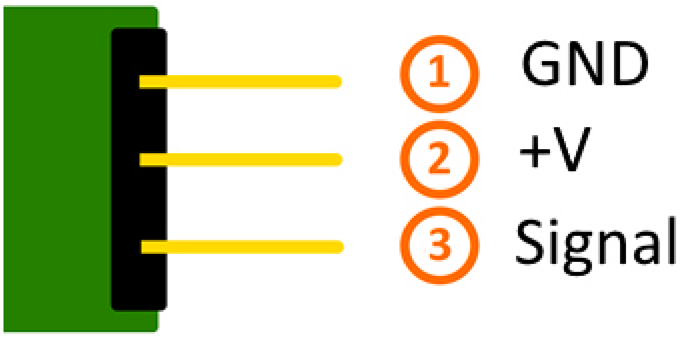


Technical data / Short description

Chipset: DHT11 | Communication protocol: 1-wire measuring range Humidity 20-90%RH measuring termperature: 0-50°C

Advantages of the sensor are the combination between temperature measurment and humidity measurements in one compact design - the disadvantage is the low sampling rate of the measurement, you are only able to get a new result every 2 seconds - so thid sensor is better to use for long period measurements.

Pinout



Software example Arduino

This sensor is not sending it's measurements analog.
The measured data is transferred digital.

The best way to use this sensor is with the Adafruit_DHT library from the company Adafruit which is published under the following Link under the OpenSource MIT-Lizenz.

The example below is using this library - for this we advise to download it from Github, to copy and to unzip it in the Arduino-library folder which you can find under the path C:\user\[Username]\documents\arduino\libraries by default. You can use it then for this and the following project examples. It is also included in the download package below.

```
// Adafruit_DHT library will be included
#include "DHT.h"

// You can declare the input pin here
#define DHTPIN 2

// The sensor will be initialized here
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  Serial.begin(9600);
  Serial.println("KY-015 test - temperature and humidity-test:");

  // Mearsurement will be started
  dht.begin();
}

// Main program loop
// The program will be started and the measurements will be read by it.
// It takes a break between every measurement to take new values.
void loop() {
  // Two second break between measurements
  delay(2000);

  // Measurement of humidity
  float h = dht.readHumidity();
  // Measurement of temperature
  float t = dht.readTemperature();

  // The measurements will be tested of errors here
  // If an error is detected, an error message will be displayed
  if (isnan(h) || isnan(t)) {
    Serial.println("Error while reading the sensor");
    return;
  }

  // Output at the serial console
  Serial.println("-----------------------------------------------------------");
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("temperature: ");
  Serial.print(t);
  Serial.print(char(186)); //Output <°> symbol
  Serial.println("C ");
  Serial.println("-----------------------------------------------------------");
  Serial.println(" ");
}
```

Please notice, the sensor will only get a new measurement value after 2 seconds. Because of that it is better to use it for long period measurements.

Example program download:

KY-015-Sensor-Temperature-Moisture

Connections Arduino:

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

Software example Raspberry Pi

To control the DHT11-sensor, the Adafruit_python_library from the company Adafruit will be used in our example. This library is published under the following Link with the MIT OpenSource-Lizenz.

The library has to be installed before using:

At first, you need to make sure that your system is able to compile Python extensions.

```
sudo apt-get update
sudo apt-get install build-essential python-dev
```

Then you have to install the Github-sofware on the Raspberry Pi if it is not already done:

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

You can download and unzip the current version of the library with the command...

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

After that you need to enter the downloaded folder with...

```
cd Adafruit_Python_DHT/
```

... and install the library with ...

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

In order that the Raspberry Pi can communicate with the sensor via I2C-bus, you have to activate the I2C function from the Raspberry Pi.

For this, you have to append the following line to the end of the file "/boot/config.txt":

```
    dtparam=i2c_arm=on
```

You can edit the file with the command:

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

You can save and close the file with the key sequence [ctrl+x -> y -> enter].

Furthermore you need additional libraries to use I2C with python. To install it use the following command:

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

You can use the following python code example now. The program starts the measurement at the sensor and shows the measurements of air pressure, temperature and the highness above sea level.

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

# Needed modules will be imported
import RPi.GPIO as GPIO
import Adafruit_DHT
import time

# The break of 2 seconds will be configured here
sleeptime = 2

# Sensor should be set to Adafruit_DHT.DHT11,
# Adafruit_DHT.DHT22, or Adafruit_DHT.AM2302.
DHTSensor = Adafruit_DHT.DHT11

# The pin which is connected with the sensor will be declared here
GPIO_Pin = 23

print('KY-015 sensortest - temperature and humidity')
```

```
try:
    while(1):
        # Measurement will be started and the result will be written into the variables
        humid, temper = Adafruit_DHT.read_retry(DHTSensor, GPIO_Pin)

        print("-----------------------------------------------------------------")
        if humid is not None and temper is not None:

            # The result will be shown at the console
            print('temperature = {0:0.1f}°C  | rel. humidity = {1:0.1f}%'.format(temper, humid))

        # Because of the OS, the Raspberry Pi has problems with realtime measurements.
        # It is possible that, because of timing problems, the communication fails.
        #  a message will be displayed - the result will be shown at next try.
        else:
            print('Error while reading - please wait for the next try!')
        print("-----------------------------------------------------------------")
        print("")
        time.sleep(sleeptime)

# Scavenging work after the end of the program
except KeyboardInterrupt:
    GPIO.cleanup()
```

Please notice, the sensor will only get a new measurement value after 2 seconds. Because of that it is better to use it for long period measurements.

Connections Raspberry Pi:

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

Example download

KY-015-RPi_Kombi-Sensor_Temperatur_Feuchtigkeit

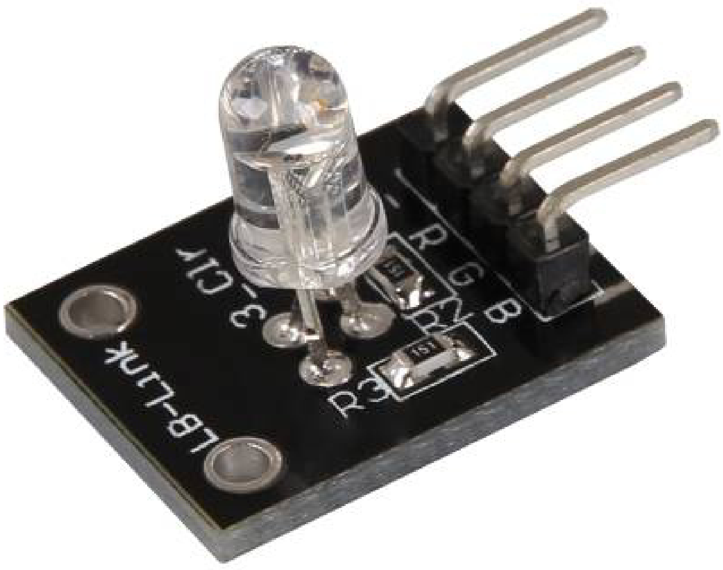
To start with the command:

```
sudo python KY-015-RPi_Kombi-Sensor_Temperatur_Feuchtigkeit.py
```

KY-016 RGB 5mm LED module

Contents

Picture



Technical data / Short description

LED-module which includes a red, blue and green LED. These are connected by a common cathode.
An additional resistor might be necessary for some voltages.

Vf [Red]= 1,8V

Vf [Green,Blue]= 2,8V

If= 20mA

Vorwiderstände:

Rf (3,3V) [Green]= 100Ω

Rf (3,3V) [Red]= 180Ω

Rf (3,3V) [Blue]= 100Ω

[e.g. by using with ARM CPU-core based microcontroller like Raspberry-Pi]

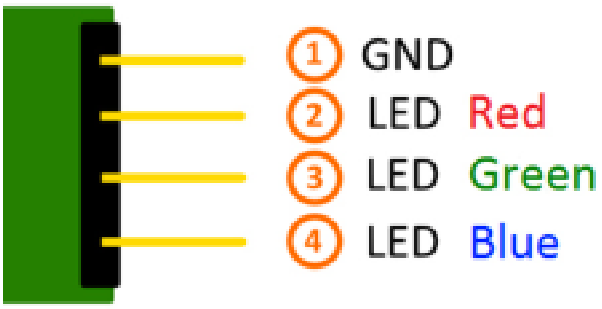
Rf (5V) [Green] = 100Ω

Rf (5V) [Red] = 180Ω

Rf (5V) [Blue] = 100Ω

[e.g. by using with Atmel Atmega based mocrocontroller like Arduino]

Pinout



Code example Arduino

Code example ON/OFF

In this example you will see how the LED is turned on by an output pin, in a 3 second clock pulse.

```
int Led_Red = 10;
int Led_Green = 11;
int Led_Blue = 12;

void setup ()
{
  // Output pin initialization for the LEDs
  pinMode (Led_Red, OUTPUT);
  pinMode (Led_Green, OUTPUT);
  pinMode (Led_Blue, OUTPUT);
}

void loop () //main program loop
{
  digitalWrite (Led_Red, HIGH); // LED will be switched ON
  digitalWrite (Led_Green, LOW); // LED will be switched OFF
  digitalWrite (Led_Blue, LOW); // LED will be switched OFF
  delay (3000); // Waitmode for 3 seconds
  digitalWrite (Led_Rot, LOW); // LED will be switched OFF
  digitalWrite (Led_Gruen, HIGH); // LED wwill be switched ON
  digitalWrite (Led_Blau, LOW); // LED will be switched OFF
  delay (3000); // Waitmode for another 3 seconds in which the LEDs will be shifted.

  digitalWrite (Led_Rot, LOW); // LED will be switched OFF
  digitalWrite (Led_Gruen, LOW); // LED will be switched OFF
  digitalWrite (Led_Blau, HIGH); // LED will be switched ON
  delay (3000); // Waitmode for another 3 seconds in which the LEDs will be shifted.
}
```

Example program ON/OFF download:

KY-016_LED_ON-OFF.zip

Code example PWM

You can regulate the brightness of the LEDs via pulse-width modulation. The LEDs will be switched ON and OFF for specific time periods, in which the relation between ON and OFF leads to a relative brightness, because of the Inertia of the human eyesight, the human eye interprets the ON/OFF as a brightness change. For more information to that theme visit: [Artikel von mikrokontroller.net].

This module provides a few LEDs - with the overlay of the different brightness levels, you can create different colors. This will be shown in the following code example.

```
int Led_Red = 10;
int Led_Green = 11;
int Led_Blue = 12;

int val;

void setup () {
  // Output pin initialization for the LEDs
  pinMode (Led_Red, OUTPUT);
  pinMode (Led_Green, OUTPUT);
  pinMode (Led_Blue, OUTPUT);
}
void loop () {
   // In this for-loop, the 3 LEDs will get different PWM-values
   // Via mixing the brightness of the different LEDs, you will get different colors.
   for (val = 255; val> 0; val--)
      {
       analogWrite (Led_Blue, val);
       analogWrite (Led_Green, 255-val);
       analogWrite (Led_Red, 128-val);
       delay (1);
   }
   // You will go backwards through the color range in this second for loop.
   for (val = 0; val <255; val++)
      {
      analogWrite (Led_Blue, val);
      analogWrite (Led_Green, 255-val);
      analogWrite (Led_Red, 128-val);
      delay (1);
   }
}
```

Example program PWM download:

KY-016_PWM.zip

Connections Arduino:

LED Red	= [Pin 10]
LED Green	= [Pin 11]
LED Blue	= [Pin 12]
Sensor GND	= [Pin GND]

Code example Raspberry Pi

Code example ON/OFF

In this example you will see how the LED is turned on by an output pin, in a 3 second clock pulse.

```
# Needed modules will be imported
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# The output pins will be declared, which are connected with the LEDs.
LED_RED = 6
LED_GREEN = 5
LED_BLUE = 4

GPIO.setup(LED_RED, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_GREEN, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_BLUE, GPIO.OUT, initial= GPIO.LOW)

print "LED-test [press ctrl+c to end]"

# main program loop
try:
        while True:
                        print("LED RED is on for 3 seconds")
                        GPIO.output(LED_RED,GPIO.HIGH) #LED will be switched ON
                        GPIO.output(LED_GREEN,GPIO.LOW) #LED will be switched OFF
                        GPIO.output(LED_BLUE,GPIO.LOW) #LED will be switched OFF
                        time.sleep(3) # waitmode for 3 seconds
                        print("LED GREEN is on for 3 seconds")
                        GPIO.output(LED_RED,GPIO.LOW) #LED will be switched OFF
                        GPIO.output(LED_GREEN,GPIO.HIGH) #LED will be switched ON
                        GPIO.output(LED_BLUE,GPIO.LOW) #LED will be switched OFF
                        time.sleep(3) # waitmode for 3 seconds
                        print("LED BLUE is on for 3 seconds")
                        GPIO.output(LED_RED,GPIO.LOW) #LED will be switched OFF
                        GPIO.output(LED_GREEN,GPIO.LOW) #LED will be switched OFF
                        GPIO.output(LED_BLUE,GPIO.HIGH) #LED will be switched ON
                        time.sleep(3) #waitmode for 3 seconds

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Example program ON/OFF download

To start, enter the command:

```
sudo python KY016_RPI_ON-OFF.py
```

Code example PWM

You can regulate the brightness of the LEDs via pulse-width modulation. The LEDs will be switched ON and OFF for specific time periods, in which the relation between ON and OFF leads to a relative brightness, because of the Inertia of the human eyesight, the human eye interprets the ON/OFF as a brightness change. For more information to that theme visit: [Artikel von mikrokontroller.net].

This module provides a few LEDs - with the overlay of the different brightness levels, you can create different colors. This will be shown in the following code example. At the Raspberry Pi, only one Hardware-PWM channel is carried out unrestricted to the GPIO pins, why we have used Software-PWM at this example.

```
# Needed modules will be imported and configured
import random, time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

# Declaration of the output pins, which are connected with the LEDs
LED_Red = 6
LED_Green = 5
LED_Blue = 4

# Set pins to output mode
GPIO.setup(LED_Red, GPIO.OUT)
GPIO.setup(LED_Green, GPIO.OUT)
GPIO.setup(LED_Blue, GPIO.OUT)

Freq = 100 #Hz

# The different colors will be initialized
RED = GPIO.PWM(LED_Red, Freq)
GREEN = GPIO.PWM(LED_Green, Freq)
BLUE = GPIO.PWM(LED_Blue, Freq)
RED.start(0)
GREEN.start(0)
BLUE.start(0)

# This function generate the actually color
# You can change the color with the specific color variable
# After the configuration of the color if finished, you will use time.sleep to
# configure how long the specific color will be displayed

def LED_color(Red, Green, Blue, pause):
    RED.ChangeDutyCycle(Red)
    GREEN.ChangeDutyCycle(Green)
    BLUE.ChangeDutyCycle(Blue)
    time.sleep(pause)

    RED.ChangeDutyCycle(0)
    GREEN.ChangeDutyCycle(0)

print "LED-test [press ctrl+c to end]"

# Main program loop:
# The task of this loop is to create for every single color an own variable
# By mixing the brightness levels of the colors, you will get a color gradient.
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)
                    for i in range(0,101):
                        LED_color((x*i),(y*i),(z*i),.02)

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Example program PWM download:

KY-016_RPi_PWM.zip

To start, enter the command:

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

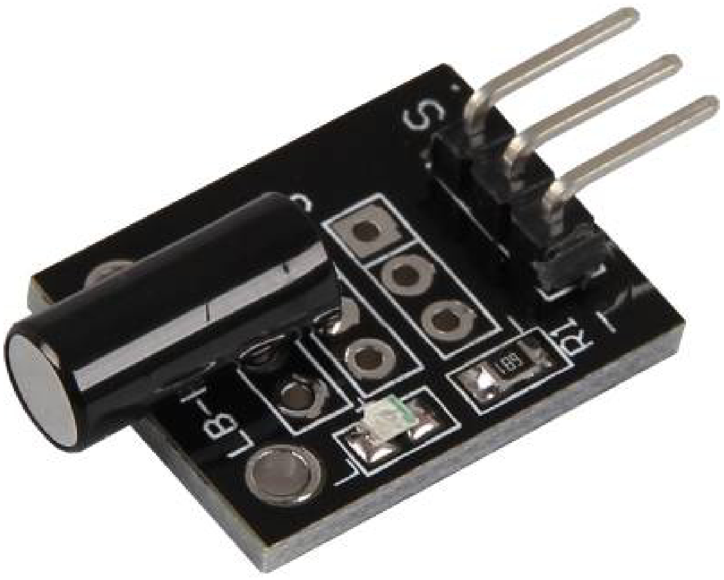
Connections Raspberry Pi:

LED Red	= GPIO6	[Pin 22]
LED Green	= GPIO5	[Pin 18]
LED Blue	= GPIO4	[Pin 16]
Sensor GND	= GND	[Pin 6]

KY-017 Tilt switch module

Contents

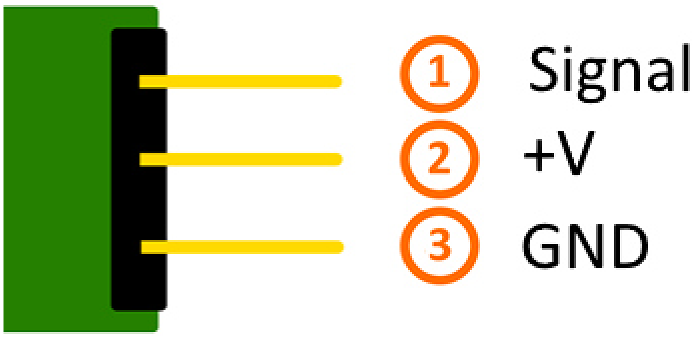
Picture



Technical data / Short description

Depending on the angle, a switch connects the circuit.

Pinout



Code example Arduino

This example will light up a LED as soon as the tilt module is in the right angle.

The modules KY-011, KY-016 or KY-029 can be used as LED.

```
int Led = 13 ;// Declaration of the LED output pin
int Sensor = 10; // Declaration of the sensor input pin
int val; // temporary variable

void setup ()
{
  pinMode (Led, OUTPUT) ; // Initialization output pin
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
}

void loop ()
{
  val = digitalRead (Sensor) ; // The active signal at the sensor will be read

  if (val == HIGH) // If a signal was noticed, the LED will be on.
  {
    digitalWrite (Led, LOW);
  }
  else
  {
    digitalWrite (Led, HIGH);
  }
}
```

Connections Arduino:

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

Sensor -	= [Pin GND]

Example program download

SensorTest_Arduino_withoutPullUP

Code example Raspberry Pi

```
# Needed modules will be imported.
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# The input pin of the Sensor will be declared. The pullup resistor will be activated
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN)

print "Sensor-test [press ctrl+c to end]"

# This output function will be started at signal detection
def outFunction(null):
        print("Signal detected")

# The output function will be activated after a signal was detected
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=outFunction, bouncetime=100)

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

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

SensorTest_RPi_withoutPullUP

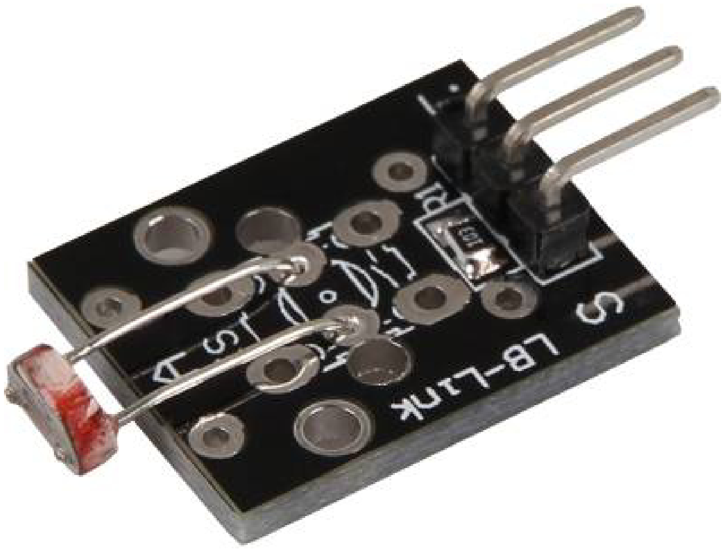
To start, enter the command:

```
sudo python SensorTest_RPi_withoutPullUP.py
```

KY-018 Photoresistor module

Contents

Picture

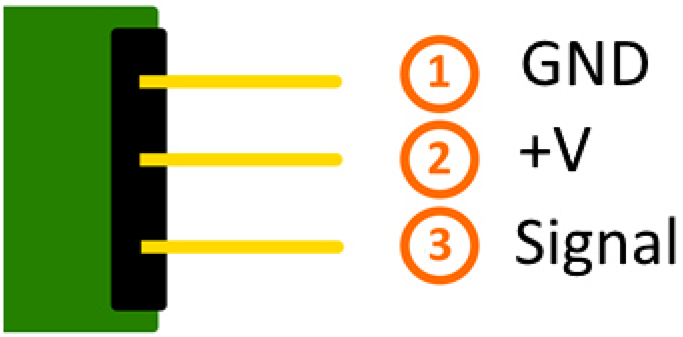


Technical data / Short description

Includes a LDR-resistor (Light dependent resistor), which decreases its resistance at brighter surrounding.

You can determine the resistance of the resistor by using a voltage divider, where a known voltage will be divided between a known and an unknown variable resistor. You can calculate the resistance with the measured voltage - the code example below includes the exact calculation.

Pinout



Code example Arduino

The example will measure the current sensors voltage and calculates the current resisitance of the sensor.

```
int sensorPin = A5; // Declaration of the input pin

// Serial output in 9600 Baud
void setup()
{
        Serial.begin(9600);
}

// The program measures the current voltage at the sensor ,
// takes the value of the known resistor and calculates the current resistance of the sensor.
// After that it show the result via serial output.

void loop()
{
        // Current voltage will be measured...
        int rawValue = analogRead(sensorPin);
        float voltage = rawValue * (5.0/1023) * 1000;

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

        // and here it will be outputted via serial infterface
        Serial.print("Voltage value:");       Serial.print(voltage); Serial.print("mV");
        Serial.print(", Resistor value:"); Serial.print(resitance); Serial.println("Ohm");
        Serial.println("---------------------------------------");

        delay(500);
}
```

Connections Arduino:

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

Example program download

Single_Analog_Sensor

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite:[[KY-053 Analog Digital Converter]]

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program measures the current voltage at the sensor, takes the value of the known resistor and calculates with it the current resistance of the sensor.

After that it will show the result via serial output.

```
#######################################################################################

### 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
###
#######################################################################################

# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC
```

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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

# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

#########################################################################################


# ########
# Main Loop
# ########
# Reading the values from the input pins and print to console

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

                # print to console
                print "Channel 0:", adc0, "mV "
                print "Channel 1:", adc1, "mV "
                print "Channel 2:", adc2, "mV "
                print "Channel 3:", adc3, "mV "
                print "---------------------------------------"

                time.sleep(delayTime)
```

Analog Signal	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 17]
GND	=	GND	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: analog signal]

Example program download

KY-053_RPi_AnalogDigitalConverter

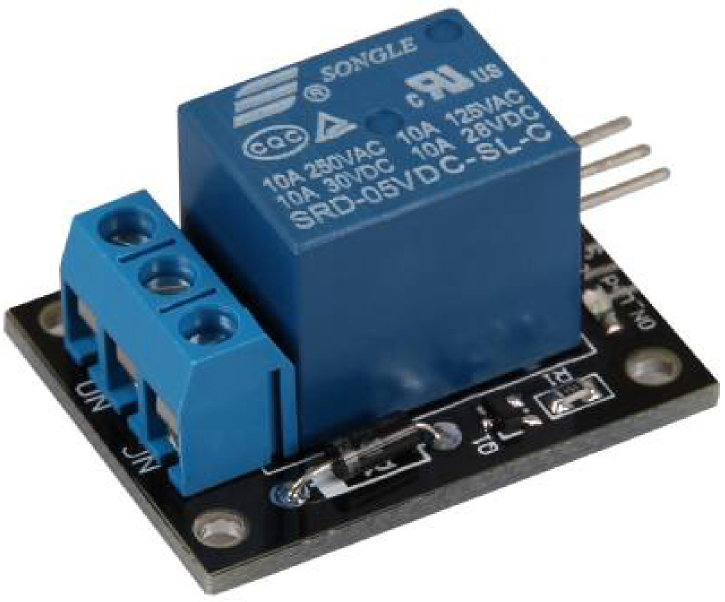
To start, enter the command:

```
sudo python RPi_Single_Analog_Sensor.py
```

KY-019 5V Relais module

Contents

Picture



Technical data / Short description

Voltage range: 240VAC / 10A | 28VDC / 10A A relay to switch higher voltages via 5V output.

!!!!! Caution !!!!!

Working with voltages over 30V and a main voltage (230V) can harm your body or kill you. We advise you not to work with higher valtages unless you have the needed experience.

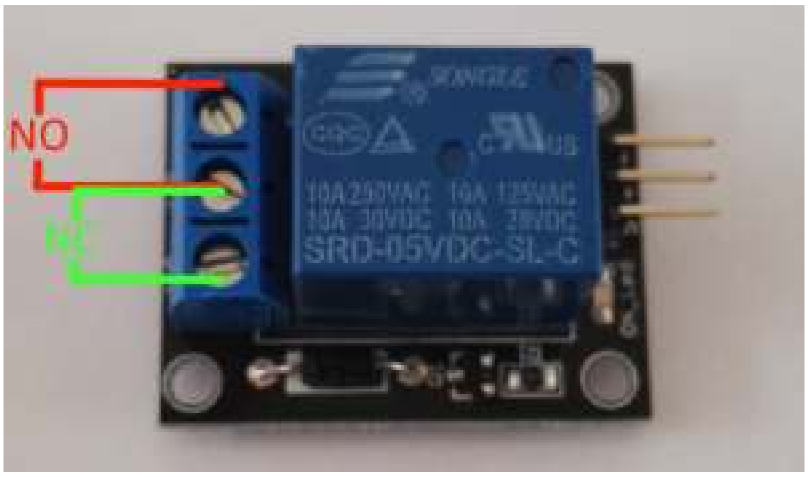
!!!!! Caution !!!!!

The output bar of the Relais has two output terminals.

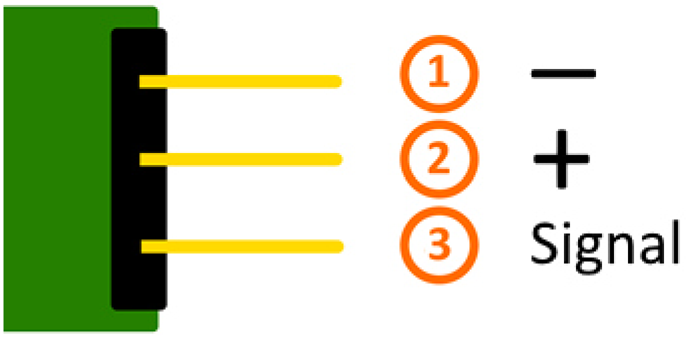
- The first one is tagged with NC for "normally closed" which means that it's connected through by default.

- The second one is tagged with NO for "normally open" wich means it's not connected through by default.

To switch both, you need a signal.



Pinout



Code example Arduino

The program imitates a direction indicator - it switchs the status of the output terminals in a specific time period (delayTime).

```
int relay = 10; // Declaration of the pin which is connected with the relay

delayTime = 1 // The time which will be waited between the switches of the relay.

void setup ()
{
  pinMode (relay, OUTPUT); // Declaration of the pin to output
}
```

```
// The program imitates a direction indicator
void loop ()
{
  digitalWrite (relay, HIGH); // "NO" is now connected through
  delay (delayTime * 1000);
  digitalWrite (relay, LOW); // "NC" is now connected through
  delay (delayTime * 1000);
}
```

Connections Arduino:

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

Example program download

KY-019_Relais

Code example Raspberry Pi

The program imitates a direction indicator - it switchs the status of the output terminals in a specific time period.

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# Declaration of the break between the changes of the relay status (in seconds)
delayTime = 1

# Declaration of the input pin which is connected with the sensor.
RELAIS_PIN = 21
GPIO.setup(RELAIS_PIN, GPIO.OUT)
GPIO.output(RELAIS_PIN, False)

print "Sensor-test [press ctrl+c to end]"


# Main program loop
try:
        while True:
            GPIO.output(RELAIS_PIN, True) # NO is now connected through
            time.sleep(delayTime)
            GPIO.output(RELAIS_PIN, False) # NC is now connected through
            time.sleep(delayTime)

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

KY-019_RPi_Relais

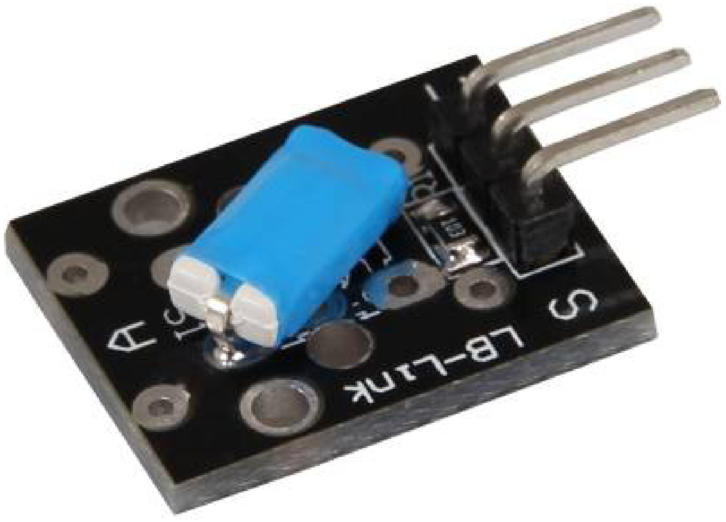
To start, enter the command:

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

KY-020 Tilt switch module

Contents

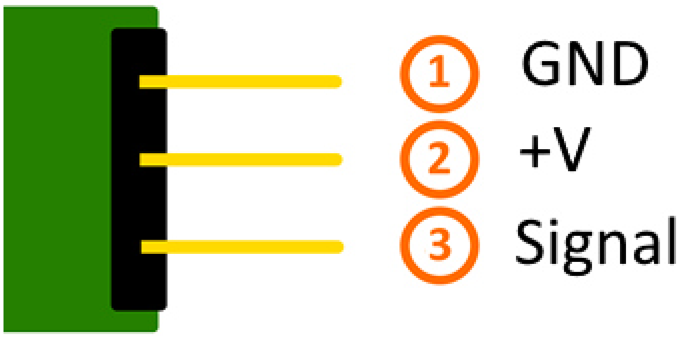
Pircture



Technical data / Short description

Depending on the angle, a switch connects the circuit.

Pinout



Code example Arduino

This example will light up a LED after a sensor detected a signal.

the modules KY-011, KY-016 or KY-029 could be used as LED too for example.

```
int Led = 13 ;// Declaration of the LED output pin
int Sensor = 10; // Declaration of the sensor input pin
int val; // Temporary variable

void setup ()
{
  pinMode (Led, OUTPUT) ; // Initialization output pin
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
}

void loop ()
{
  val = digitalRead (Sensor) ; // The current signal at the sensor will be read

  if (val == HIGH) // If a signal will be detected, the LED will light up
  {
    digitalWrite (Led, LOW);
  }
  else
  {
    digitalWrite (Led, HIGH);
  }
}
```

Connections Arduino:

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

Sensor -	= [Pin GND]

Example program download

SensorTest_Arduino_withoutPullUP

Code example Raspberry Pi

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Declaration of the input pin which is connected with the sensor.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN)

print "Sensor-test [press ctrl+c to end]"

# This outFunction will be started at signal detection.
def outFunction(null):
        print("Signal detected")

# The outFunction will be started after detecting of a signal (falling signal edge)
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=outFunction, bouncetime=100)

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

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

SensorTest_RPi_withoutPullUP

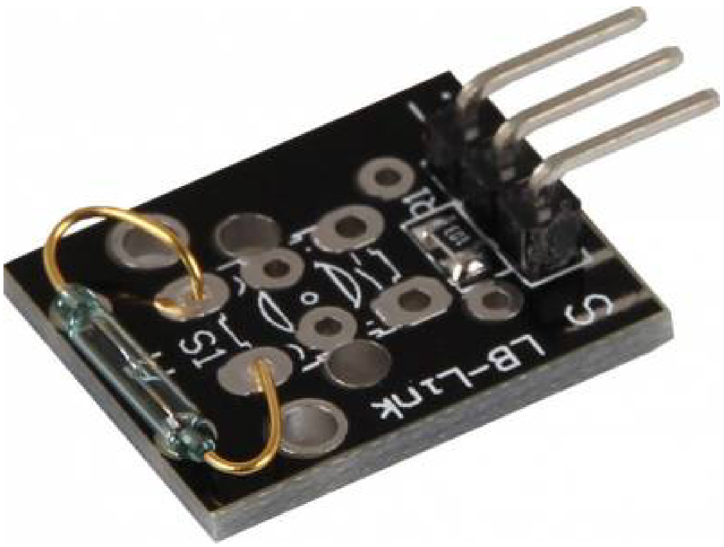
To start, enter the command:

```
sudo python SensorTest_RPi_withoutPullUP.py
```

KY-021 Mini magnetic Reed module

Contents

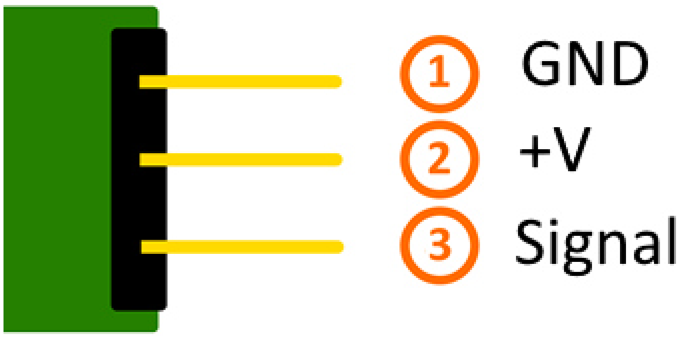
Picture



Technical data / Short description

If the sensor is close to a magnetic field, the input pins are connected.

Pinout



Code example Arduino

This example will activate a LED if the sensor is close to a magnetic field.

The modules KY-011, KY-016 or KY-029 can be used as a LED.

```
int Led = 13 ;// Declaration of the LED output pin.
int Sensor = 10; //Declaration of the sensor input pin
int val; // Temporary variable

void setup ()
{
  pinMode (Led, OUTPUT) ; // Initialization output pin
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
}

void loop ()
{
  val = digitalRead (Sensor) ; // The current signal at the sensor will be read

  if (val == HIGH) // If a signal will be detected, the LED will light up.
  {
    digitalWrite (Led, LOW);
  }
  else
  {
    digitalWrite (Led, HIGH);
  }
}
```

Connections Arduino:

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

Sensor -	= [Pin GND]

Example program download

SensorTest_Arduino_withoutPullUP

Code example Raspberry Pi

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Declaration of the input pin which is connected with the sensor.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN)

print "Sensor-test [press ctrl+c to end]"

# This outFunction will be started after a signal was detected.
def outFunction(null):
        print("Signal detected")

# The outFunction will be started after a signal (falling signal edge) was detected.
GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=outFunction, bouncetime=100)

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

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

KY-021_mini-magnetic-reed_RPI

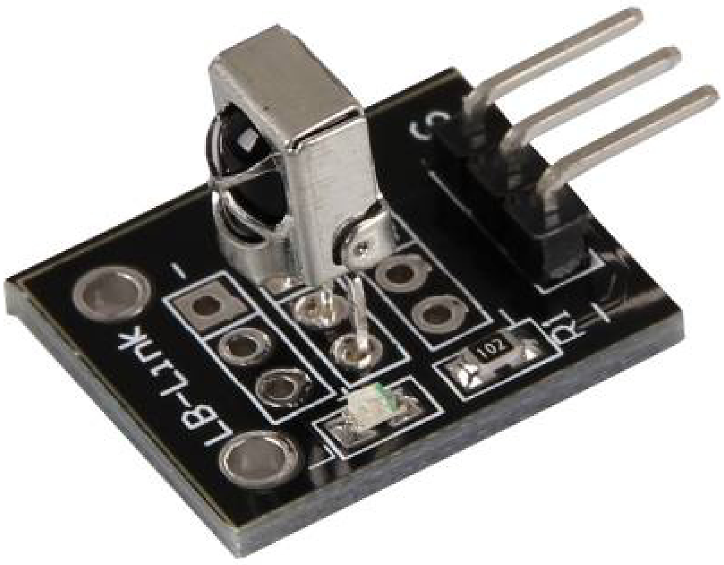
To start, enter the command:

```
sudo python SensorTest_RPi_withoutPullUP.py
```

KY-022 Infrared receiver module

Contents

Picture

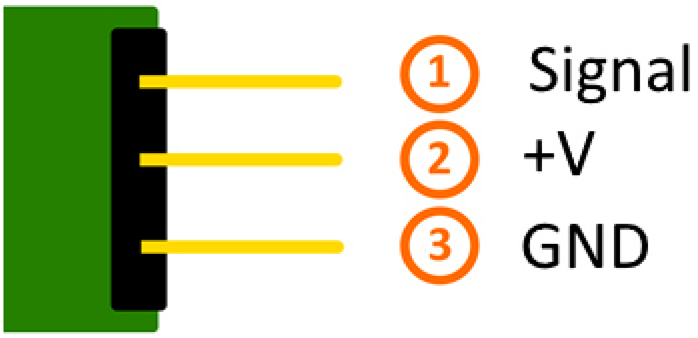


Technical data / Short description

Carrier frequency: 38kHz - can receive infrared signals and transfers them to the digital signal out.

Additionally, the LED of this module will light up if an infrared signal is detected.

Pinout



Code example Arduino

With both sensor modules, KY-005 and KY-022, you can build an infrared remote + infrared receiver system. In order to do this, you will need the two sensor modules as well as two Arduinos.
The first one will handle the receiver system and the second one will handle the transmitter system.

An additional library is needed for this code example:

-[Arduino-IRremote] from Ken Shirriff | published under LGPL

The library is in the package and has to be copied before the start into the library folder.

You will find it normally under the following path:

C:\User\[UserName]\Documents\Arduino\libraries

There are different infrared protocolls to send data. In this example we use the RC5 protocol. The used library "Arduino-IRremote" converts the data independently. The library has additional protocolls, they are marked in this documentation.

Code for the receiver:

```
// Arduino-IRremote library will be added
#include <IRremote.h>
#include <IRremoteInt.h>

// You can declare the input pin for the signal output of the KY-022 here
int RECV_PIN = 11;

// Arduino-IRremote library will be initialized
IRrecv irrecv(RECV_PIN);
decode_results results;
```

```
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Infrared receiver will start
}

// main program loop
void loop() {

  // It will be checked if the receiver has gotten a signal.
  if (irrecv.decode(&results)) {
    //At signal input, the received and decoded signal will show via serial console.
    Serial.println(results.value, HEX);
    irrecv.resume();
  }
}
```

Code for the transmitter:

```
//Arduino-IRremote library will be added
#include <IRremote.h>
#include <IRremoteInt.h>

//...and here initialized
IRsend irsend;

// The configuration of the output pin will be made by the library
// The output pin is a different one for different arduinos
// Arduino UNO:  Output = D3
// Arduino MEGA: Output = D9
// You will find a full list of output pins on the website:
// http://z3t0.github.io/Arduino-IRremote/
void setup()
{
}

// main program loop
void loop() {
        // The transmitter sends the signal A90 (hex. dezimal form) in the encoding "RC5"
        // It will be transmitted 3 times after that it will make a 5 second break
    for (int i = 0; i < 3; i++) {
        irsend.sendRC5(0xA90, 12); //[12] Bit-length signal (hex A90=1010 1001 0000)
        delay(40);
    }
    delay(5000); // 5 second break between the sending impulses
}
```

Example program download:

KY-005_KY-022_Infrared-Modules_ARD

Connections Arduino 1 [Receiver]:

KY-022

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

Connections Arduino 2 [Transmitter]:

KY-005

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

- * Only if resistor was soldered to the circuit board.

Code example Raspberry Pi

Code example remote

Because of its progressive processor architecture, the Raspberry Pi has a big advantage, compared to the Arduino.

It can run a full Linux OS. With help of an infrared-receiver, it can not only transmit simple data signals, furthermore it can control complete programs via remote.

To setup an infrared control system, we recommend to use the Linux software "lirc" (published under the LGPL-Website).

In the following section, we show you how to use lirc and how the remotely send the learned signals via infrared.

On this purpose, the module KY-005 will be used as an infrared-transmitter and the KY-022 will be used as an infrared-receiver.

Connections Raspberry Pi:

KY-005

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

- * Only if a resistor was soldered to the module

KY-022

Signal	= GPI18	[Pin 12]
+V	= 3,3V	[Pin 17]
GND	= GND	[Pin 25]

Lirc Installation

Open a terminal at the desktop or use SSH to log into your Raspberry Pi. To install lirc, enter the following command:

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

[For this the Raspberry Pi has to be connected to the internet]

To use the lirc module immediately after starting the OS, you have to add the following line to the end of the file "/boot/config.txt":

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

The "gpio_in_pin=18" will be defined as an input pin of the IR-receiver and the "gpio_out_pin=17" as an output pin of the IR-transmitter.

The file can be edited by entering the command:

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

You can save and close the file via the key sequence [ctrl+x -> y -> enter]

You will also have to modify the file "/etc/lirc/hardware.conf" by entering the command:

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

In this file you have to change following lines:

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

The modified file should now look like:

```
# /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.
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=""
```

After that we reboot the Raspberry Pi with the following command:

```
sudo reboot
```

IR-Receiver Test

To test the connected receiver, you have to close lirc first with the following command:

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

After that, you can test if signals could be detected on the Raspberry Pi by using the following command:

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

and by pressing random keys on an infrared remote. Now you should see numbers in the following form:

```
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
...
```

You can restart lirc with the following command:

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

Remote teach

To register an infrared-remote at the lirc system, you have to configure the file "/etc/lirc"lircd.conf".

In this file, all command assignments of the infrared codes are saved.

To get a good formatted lircd.conf, use the lirc assistant software which creates the file automatically.

To start this process you have to stop lirc first by using the command:

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

With the following command, we can start the assistant:

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

The assistant will start an initialization of the remote, in this initialization you have to press a few keys so that the lirc system is able to learn the encoding of the remote. For that, please follow the instructions of the assistant. After the initialization, the assistant asks for the name of the key which should get a new infrared code. You can choose your key from the following file:

FernbedienungsCodes.txt

You have to type these into the assistant and need to confirm with enter. After this, the recording of the infrared code for the chosen key will start.

Example: type in [KEY_0] - - -> confirm with enter - - -> press key 0 of the remote - - -> waiting for the assistant to confirm the recording.

If no more keys need to be configured, you can close the assistant by pressing the enter key. After this, the configuration file is created, but you have to choose a name for the recorded remote. For this we have to open the file with the editor:

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

Here you have to change the line:

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

to

```
      name  MeineFernbedienung
```

Please don't use any spaces or additional characters in the name.

You can save and close the file with the key sequence [ctrl+x ---> y ---> enter].

After creating the configuration, you can make a backup for original lircd.conf with the following command:

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

With the command

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

you can use the before created file for the lirc system.

Now you can start the lirc system again with the command:

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

From now on, the remote is known and can be used with the right software. Alternatively you can use the following command to test the functions:

```
irw
```

Sending command via Infrared Transmitter

If you want to control devices, like your Television, via Raspberry Pi, you can now send the learned commands with the infrared transmitter. With that you can build a software controlled infrared controller or you can use the internet or the network to switch single devices on and off.

First we check with the following command:

```
irsend LIST MeineFernbedienung ""
```

which assigments are available for the remote.

Now we can send the command [KEY_0] with the command:

```
irsend SEND_ONCE MeineFernbedienung KEY_0
```

On your Television or the receiver-end-device should show up a reaction. You can have the example above in other variations like , instead of sending the signal only once , it will be send multiple times.

```
irsend SEND_START MeineFernbedienung KEY_0
```

After this, the code [KEY_0] will be repeatly send out until we end it with the following command:

```
irsend SEND_STOP MeineFernbedienung KEY_0
```

KY-023 Joystick module (XY-Axis)

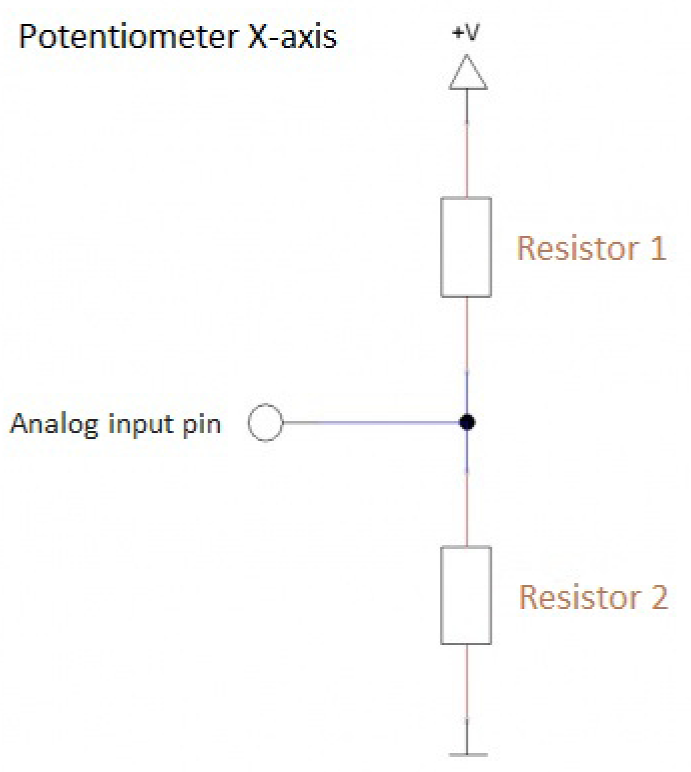
Contents

Picture

Technical data / Short description

X and Y positions of the joystick can be measured as an analog voltage at the output pin.

In this joystick, the x-axis and the y-axis have their own potentiometer. Together, they build a voltage devider like the one in the next picture.

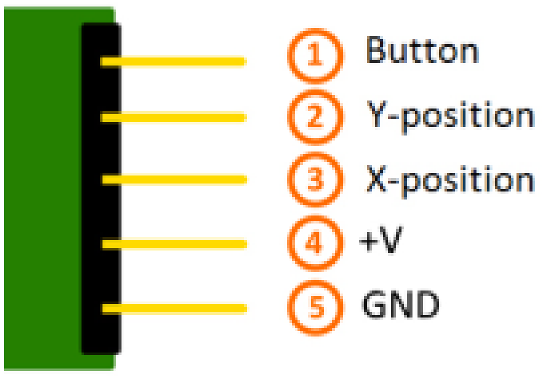


In the non-operating mode, the potentiometer is in the middle so that resistor1=resistor2, so that the voltage will equally split to both resistors - e.g. Measurement of +V=5V -> 2,5V.

If one of the axis changes, like the x-axis for example, the values of the resistors will change - e.g. value of resistor 1 will raise than the value of resistor 2 will fall or the value of resistor 1 will fall and the value of resistor 2 will raise.

According to the division of the resistor values, you can measure a specific voltage value between the resistors and locate the position of the axis.

Pinout



Code example Arduino

This program measures the value at the input pins, converts them into a voltage value (0-1023 -> 0V-5V) and prints these at the serial ouput.

```
// Declaration and initialization of the input pin
int JoyStick_X = A0; // X-axis-signal
int JoyStick_Y = A1; // Y-axis-signal
int Button = 3;

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

  // pushing the button leads to
  // power up the pullup-resistor
  digitalWrite(Button, HIGH);

  Serial.begin (9600); // serial output with 9600 bps
}

// The program reads the current values of the input pins
// and outputs them at the serial output
void loop ()
{
  float x, y;
  int Knopf;

  // Current values will be read and converted to the right voltage
  x = analogRead (JoyStick_X) * (5.0 / 1023.0);
  y = analogRead (JoyStick_Y) * (5.0 / 1023.0);
  Knopf = digitalRead (Button);

  //... and outputted here
  Serial.print ("X-axis:"); Serial.print (x, 4);  Serial.print ("V, ");
  Serial.print ("Y-axis:"); Serial.print (y, 4);  Serial.print ("V, ");
  Serial.print ("Button:");

  if(Knopf==1)
  {
      Serial.println ("not pushed");
  }
  else
  {
      Serial.println ("pushed");
  }
  delay (200);
}
```

Connections Arduino:

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

Example program download:

KY-023_Joystick_Modul

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and prints them to the terminal.

```
#########################################################################################
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported Lic
### Commercial use only after permission is requested and granted
###
### KY-053 Analog Digital Converter - Raspberry Pi Python Code Example
###
#########################################################################################


# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC

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

```
# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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

# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

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

#######################################################################################


# ########
# main program loop
# ########
# The program reads the current values of the input pins
# and outputs the values at the terminal

try:
        while True:
                # Current values will be recorded
                x = adc.readADCSingleEnded(adc_channel_0, gain, sps)
                y = adc.readADCSingleEnded(adc_channel_1, gain, sps)

                # Output at the terminal
                if GPIO.input(Button_PIN) == True:
                        print "X-axis:", x,"mV, ","Y-axis:", y,"mV, Button: not pushed"
                else:
                        print "X-axis:", x, "mV, ", "Y-axis:", y, "mV, Button: pushed"
                print "---------------------------------------"

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



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor KY-023

Button	=	GPIO24	[Pin 18 (RPi)]
Y-position	=	Analog 1	[Pin A1 (ADS1115 - KY-053)]

X-position	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]
+V	=	3,3V	[Pin 1 (RPi)]
GND	=	GND	[Pin 6 (RPi)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05 (RPi)]
SDA	=	GPIO02 / SDA	[Pin 03 (RPi)]
A0	=	look above	[Sensor: X-position (KY-023)]
A1	=	look above	[Sensor: Y-position (KY-023)]

Example program download

KY-023_Joystick_RPi

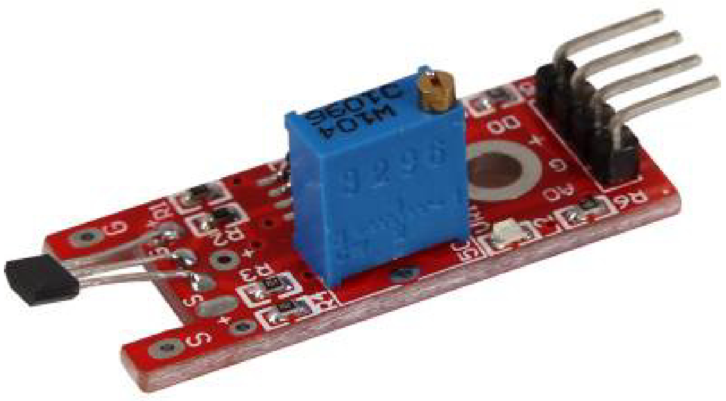
To start, enter the command:

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

KY-024 Linear magnetic Hall Sensor

Contents

Picture



Technical data / Short description

Chipset: A3141 | OP-amplifier: LM393

A magnetic field is detected by the sensor and will be printed as an analog voltage value. You can control the sensitivity of the sensor with the potentiometer.

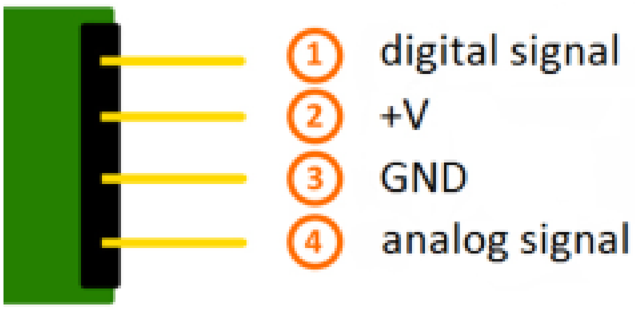
Digital out: If a magnetic field is detected by the sensor, a signal will be printed here

Analog out: Direct measurement of the sensor unit

LED1: Shows that the sensor is supplied with voltage

LED2: Shows that the sensor detects a magnetic field

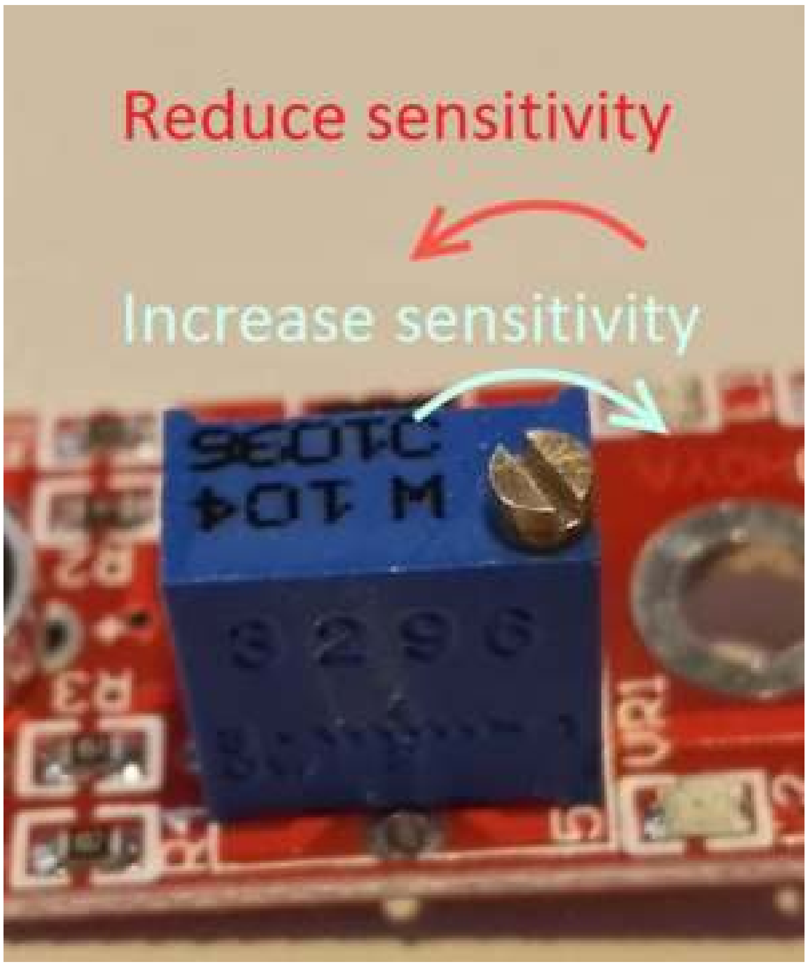
Pinout



Functionality of the sensor

The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.
The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.
You can control the sensitivity by adjusting the potentiometer.

Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.



This sensor doesn't show absolute values (like exact temperature in °C or magneticfield strenght in mT). It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

Code example Arduino

The program reads the current voltage value which will be measured at the output pin and shows it via serial interface.

Additionally, the status of the digital pin will be shown at the terminal.

```
// Declaration and initialization of the input pin
int Analog_Eingang = A0; // X-axis-signal
int Digital_Eingang = 3; // Button

void setup ()
{
  pinMode (Analog_Eingang, INPUT);
  pinMode (Digital_Eingang, INPUT);
       
  Serial.begin (9600); // Serielle output with 9600 bps
}
  
// The program reads the current value of the input pins
// and output it via serial out
void loop ()
{
  float Analog;
  int Digital;
    
  // Current value will be read and converted to the voltage
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0); 
  Digital = digitalRead (Digital_Eingang);
    
  // and outputted here
  Serial.print ("Analog voltage value:"); Serial.print (Analog, 4);  Serial.print ("V, ");
  Serial.print ("Extreme value:");
  
  if(Digital==1)
  {
      Serial.println (" reached");
  }
  else
  {
      Serial.println (" not reached yet");
  }
  Serial.println ("----------------------------------------------------------------");
  delay (200);
}
```

Connections Arduino:

digital signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analog signal	= [Pin 0]

Example program download

Analoger_Sensor

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and outputs it at the terminal in [mV].

Additional to that, the status of the digital pin will be shown at the terminal to show if the extreme value was exceeded or not.

```
#######################################################################################

### 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
### #####################################################################################


# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import math, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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

# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

# Input pin for the digital signal will be picked here
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

########################################################################################


# ########
# main program loop
# ########
# The program reads the current value of the input pin
# and shows it at the terminal

try:
        while True:
                #Current values will be recorded
                analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

                # Output at the terminal
                if GPIO.input(Digital_PIN) == False:
                        print "Analog voltage value:", analog,"mV, ","extreme value: not reached"
                else:
                        print "Analog voltage value:", analog, "mV, ", "extreme value: reached"
                print "---------------------------------------"

                sleep(delayTime)



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor

digital signal	=	GPIO 24	[Pin 18 (RPi)]
+V	=	3,3V	[Pin 1 (RPi)]
GND	=	GND	[Pin 06 (RPi)]
analog signal	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: analog signal]

Example program download

KY-024_Linear-magnetic-sensor_RPi

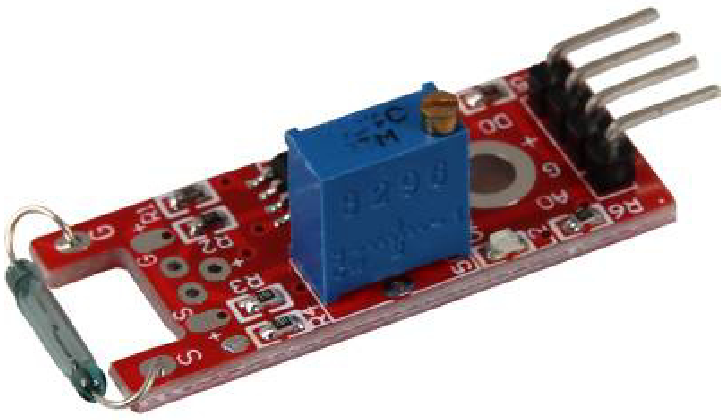
To start, enter the command:

```
sudo python RPi_AnalogSensor.py
```

KY-025 Reed module

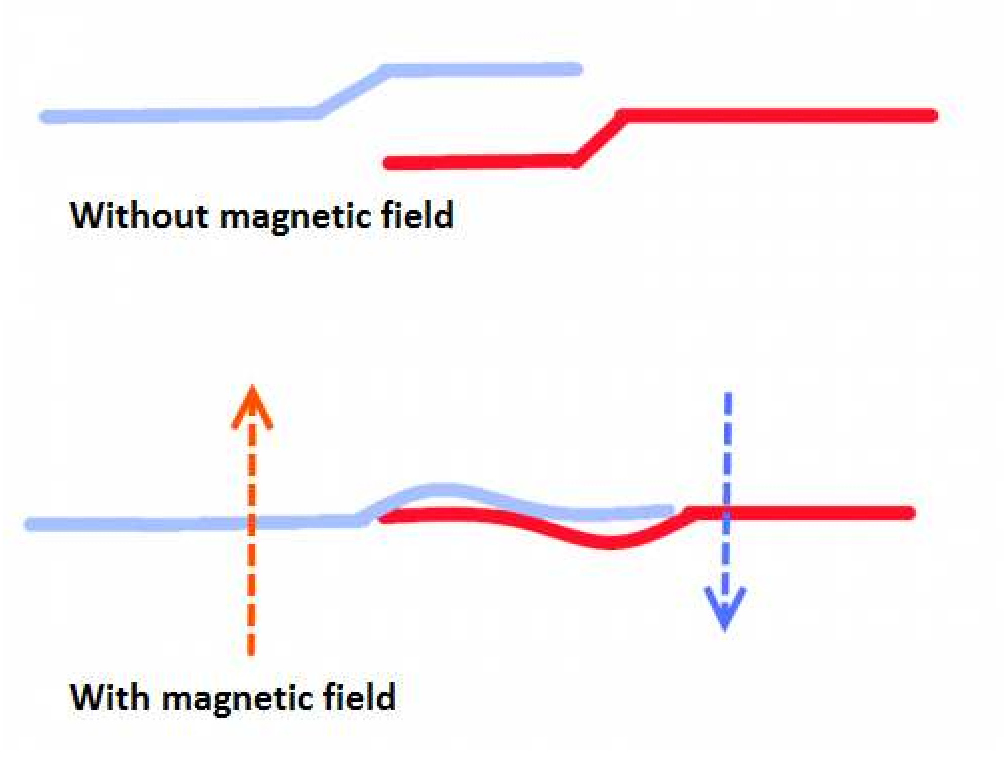
Contents

Picture



Technical data / Short description

If a magnetic field was detected, it will be shown at the digital output.



Because of that we will get an electrical connection and send the signal.

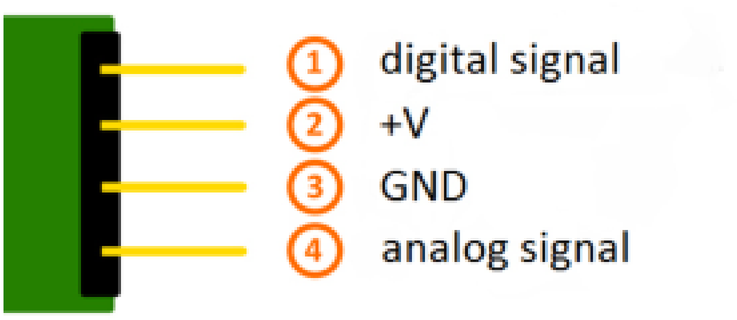
Digital Out: After detecting a magnetic field, a signal will be outputted

Analog Out: Direct measurement of the sensor unit

LED1: Shows that the sensor is connected with power

LED2: Shows that the sensor detects a magnetic field

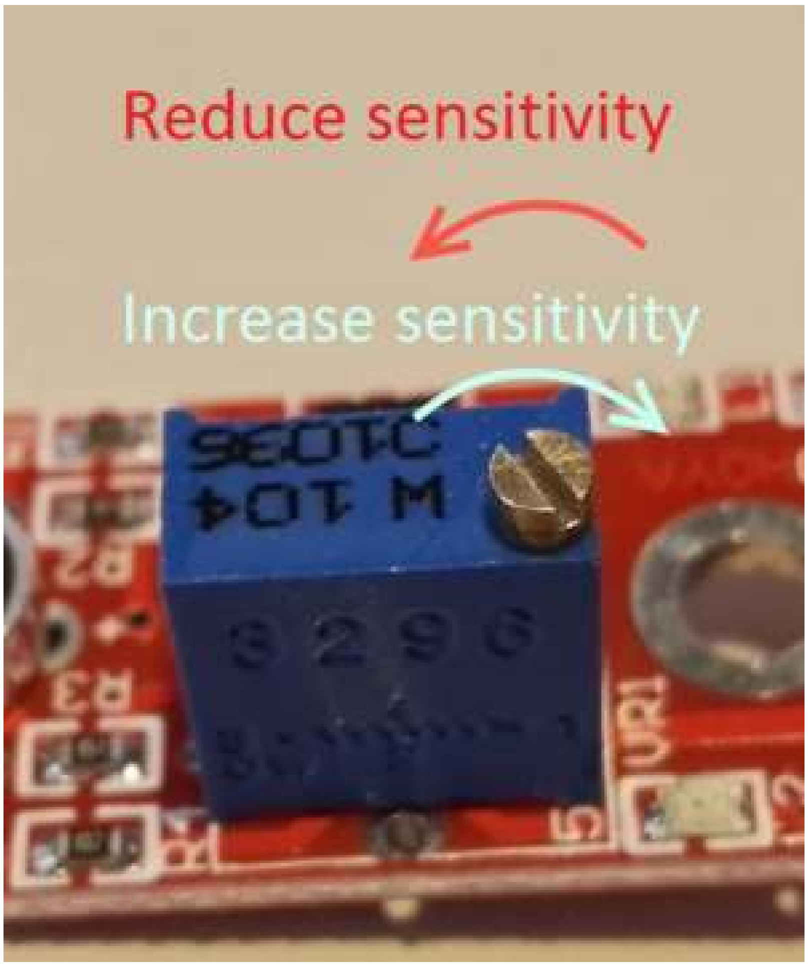
Pinout



Functionality of the sensor

The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.
The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.
You can control the sensitivity by adjusting the potentiometer.

Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.



This sensor doesn't show absolute values (like exact temperature in °C or magneticfield strenght in mT).
It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

Code example Arduino

The program reads the current voltage value which will be measured at the output pin and shows it via serial interface.

Additionally, the status of the digital pin will be shown at the terminal.

```
// Declaration and initialization of the input pin
int Analog_Eingang = A0; // X-axis-signal
int Digital_Eingang = 3; // Button

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

  Serial.begin (9600); // Serial output with 9600 bps
}

// The program reads the current value of the input pins
// and outputs it via serial out
void loop ()
{
  float Analog;
  int Digital;

  // Current value will be read and converted to the voltage
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
  Digital = digitalRead (Digital_Eingang);

  // and outputted here
  Serial.print ("Analog voltage value:"); Serial.print (Analog, 4);  Serial.print ("V, ");
  Serial.print ("Extreme value:");

  if(Digital==1)
  {
      Serial.println (" reached");
  }
  else
  {
      Serial.println (" not reached yet");
  }
  Serial.println ("----------------------------------------------------------------");
  delay (200);
}
```

Connections Arduino:

digital signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analog signal	= [Pin 0]

Example program download

Analoger_Sensor

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and output it at the terminal in [mV].

Additionally, the status of the digital pin will be shown at the terminal to show if the extreme value was exceeded or not.

```
#######################################################################################

### 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
### ###################################################################################


# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import math, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC
```

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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

# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

# Input pin for the digital signal will be picked here
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

#######################################################################################


# ########
# main program loop
# ########
# The program reads the current value of the input pin
# and shows it at the terminal

try:
        while True:
                #Current values will be recorded
                analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

                # Output at the terminal
                if GPIO.input(Digital_PIN) == False:
                        print "Analog voltage value:", analog,"mV, ","extreme value: not reached"
                else:
                        print "Analog voltage value:", analog, "mV, ", "extreme value: reached"
                print "---------------------------------------"

                sleep(delayTime)



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor

digital signal	= GPIO 24	[Pin 18 (RPi)]

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

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: analog signal]

Example program download

KY-025_Reed_module_RPi

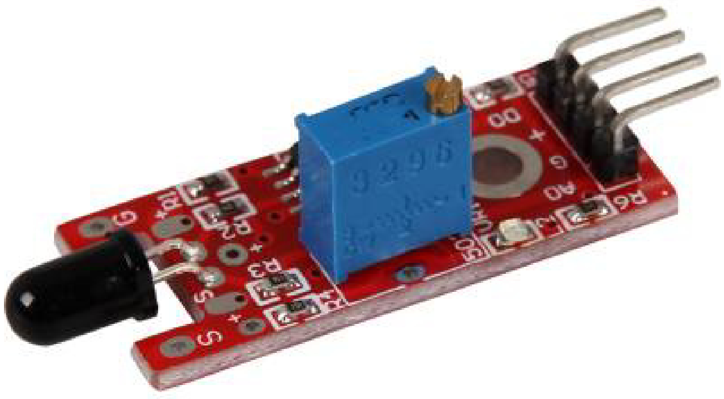
To start, enter the command:

```
sudo python KY-025_Reed_module_RPi.py
```

KY-026 Flame-sensor module

Contents

Picture



Technical data / Short description

The connected photo diode is sensitive to the spectral range of light, which is created by open flames.

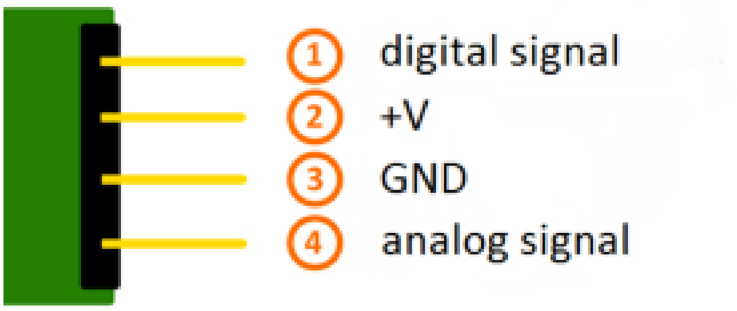
Digital Out: After detecting a flame, a signal will be outputted

Analoger Ausgang: Direct measurement of the sensor unit

LED1: Shows that the sensor is supplied with voltage

LED2: Shows that the sensor detects a flame

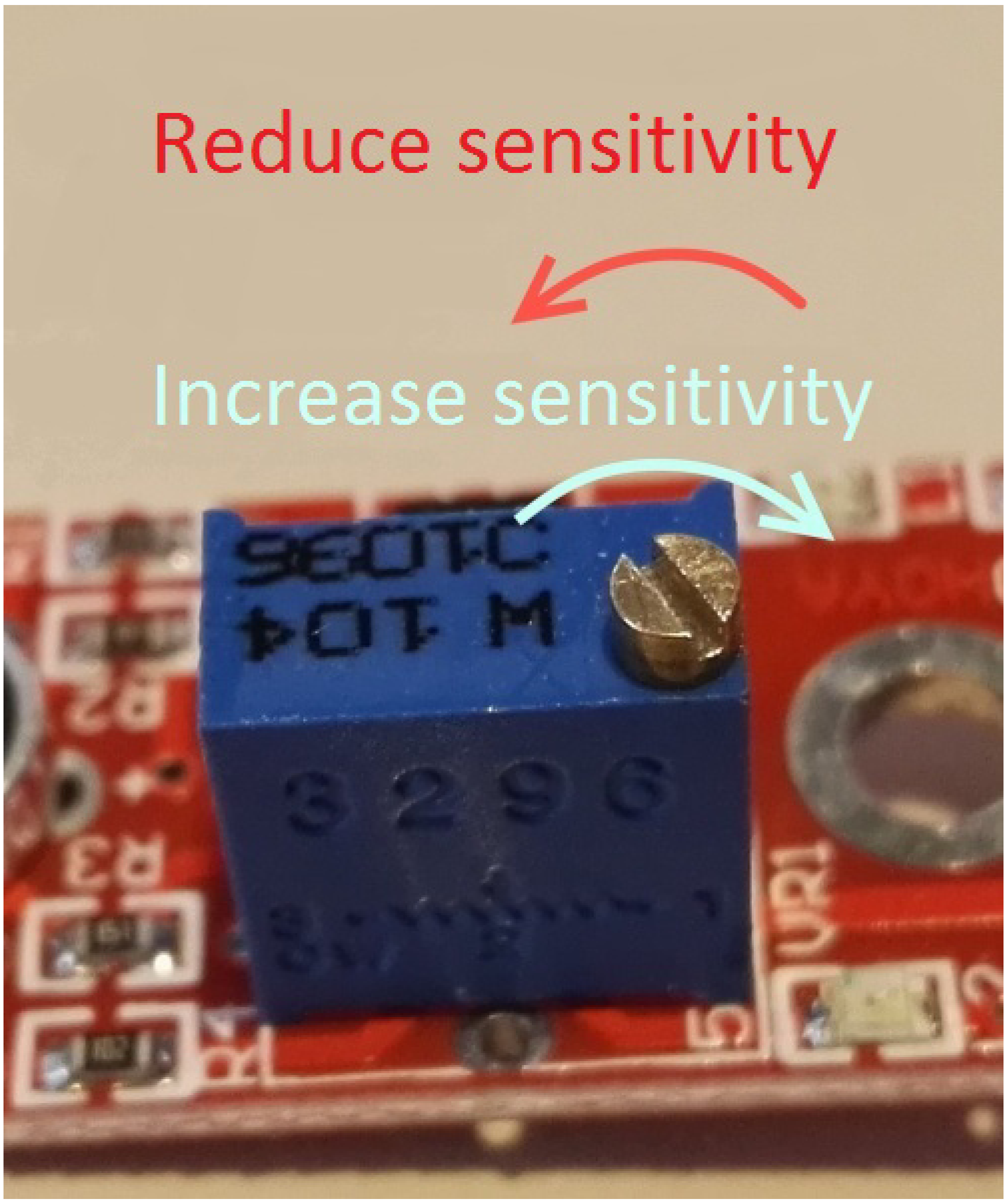
Pinout



Functionality of the sensor

The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.
The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.
You can control the sensitivity by adjusting the potentiometer.

Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.



This sensor doesn't show absolute values (like exact temperature in °C or magneticfield strenght in mT). It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

Code example Arduino

The program reads the current voltage value which will be measured at the output pin and shows it via serial interface.

Additionally, the status of the digital pin will be shown at the terminal which means if the extreme value was exceeded or not.

```
// Declaration and initialization of the input pins
int Analog_Eingang = A0; // X-axis-signal
int Digital_Eingang = 3; // Button

void setup ()
{
  pinMode (Analog_Eingang, INPUT);
  pinMode (Digital_Eingang, INPUT);
       
  Serial.begin (9600); // serial output with 9600 bps
}
   
// The program reads the current values at the input pins
// and outputs them at the serial output
void loop ()
{
  float Analog;
  int Digital;
     
  //Current values will be read and converted to voltage
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
  Digital = digitalRead (Digital_Eingang);
     
  //... and outputted here
  Serial.print ("Analog voltage value:"); Serial.print (Analog, 4);  Serial.print ("V, ");
  Serial.print ("Extreme value:");
   
  if(Digital==1)
  {
      Serial.println (" reached");
  }
  else
  {
      Serial.println (" not yet reached");
  }
  Serial.println ("----------------------------------------------------------------");
  delay (200);
}
```

Connections Arduino:

digital signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analog signal	= [Pin 0]

Example program download

Analoger_Sensor

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and outputs it at the terminal in [mV].

Additional to that, the status of the digital pin will be shown at the terminal to show if the extreme value was exceeded or not.

```
### 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
###




# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import math, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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

# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

# Input pin for the digital signal will be picked here
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

########################################################################################


# ########
# main program loop
# ########
# The program reads the current value of the input pin
# and shows it at the terminal

try:
        while True:
                #Current values will be recorded
                analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

                # Output at the terminal
                if GPIO.input(Digital_PIN) == False:
                        print "Analog voltage value:", analog,"mV, ","extreme value: not reached"
                else:
                        print "Analog voltage value:", analog, "mV, ", "extreme value: reached"
                print "---------------------------------------"

                sleep(delayTime)



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor

digital signal	=	GPIO 24	[Pin 18 (RPi)]
+V	=	3,3V	[Pin 1 (RPi)]
GND	=	GND	[Pin 06 (RPi)]
analog signal	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]

SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: analog signal]

Example program download

KY-026_Flame-sensor_RPi

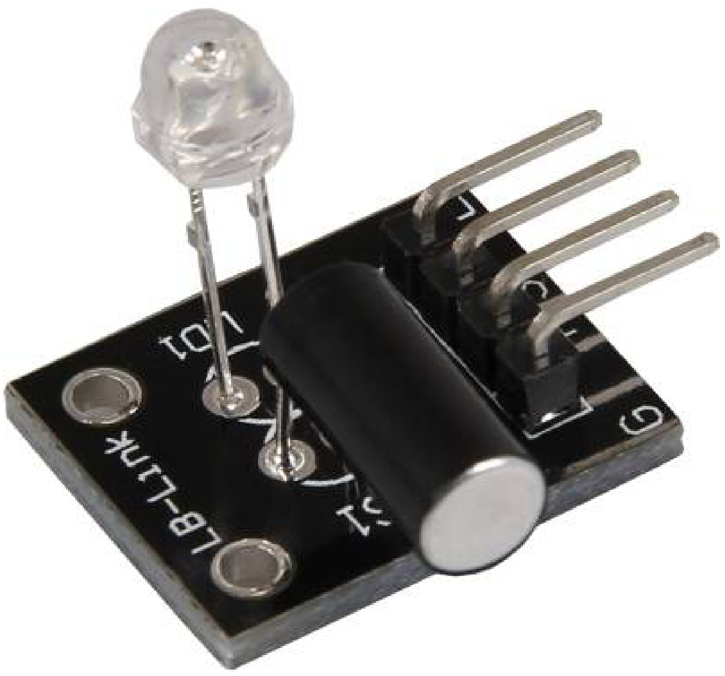
To start, enter the command:

```
sudo python KY-026_Flame-sensor_RPir.py
```

KY-027 Magic light cup module

Contents

Picture



Technical data / Short description

The LED will be switched on and off by vibration. The signal will be send to the output if the LED is on. You need pre-resistors for some voltages.

Pre-resistor:

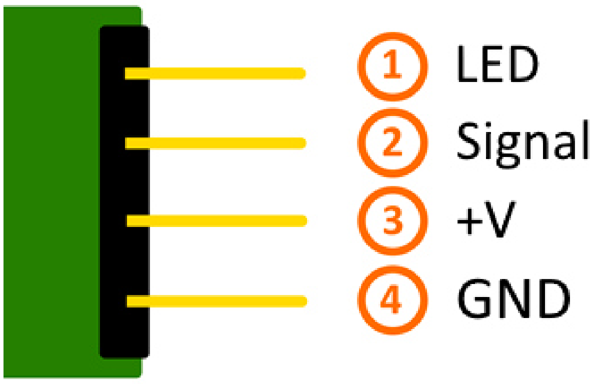
Rf (3,3V) [Red]= 120Ω

[used with ARM CPU-Core based microcontroller like Raspberry-Pi]

Rf (5V) [Red] = 220Ω

[used with Atmel Atmega based microcontroller like Arduino]

Pinout



Code example Arduino

```
int Led = 13 ;// Declaration of the LED-output pin
int Sensor = 10; // Declaration of the sensor input pin
int val; // Temporary variable

void setup ()
{
  pinMode (Led, OUTPUT) ; // Initialization output pin
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
  digitalWrite(Sensor, HIGH); // Activating of the internal pull-up resistor
}

void loop ()
{
  val = digitalRead (Sensor) ; // The current signal from the sensor will be read

  if (val == HIGH) // If a signal will be detected, the LED will light up.
  {
    digitalWrite (Led, LOW);
  }
  else
  {
    digitalWrite (Led, HIGH);
  }
}
```

Connections Arduino:

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

Example program download

SensorTest_Arduino

Code example Raspberry Pi

```
# Needed modules will be imported and configured.
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Declaration of the LED and sensor pins
LED_PIN = 24
Sensor_PIN = 23
GPIO.setup(Sensor_PIN, GPIO.IN)
GPIO.setup(LED_PIN, GPIO.OUT)

print "Sensor-test [press ctrl+c to end the test]"

# This output function will be started at signal detection
def ausgabeFunktion(null):
        GPIO.output(LED_PIN, True)

# This output function will be started at signal detection
GPIO.add_event_detect(Sensor_PIN, GPIO.FALLING, callback=ausgabeFunktion, bouncetime=10)

# main program loop
try:
        while True:
                time.sleep(1)
                # output will be reseted if the switch turn back to the default position.
                if GPIO.input(Sensor_PIN):
                        GPIO.output(LED_PIN,False)

# Scavenging work after the program has ended
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

LED	=	GPIO24	[Pin 18]
Signal	=	GPIO23	[Pin 16]
+V	=	3,3V	[Pin 1]
GND	=	GND	[Pin 6]

Example program download

KY-027-RPi-MagicLightCup

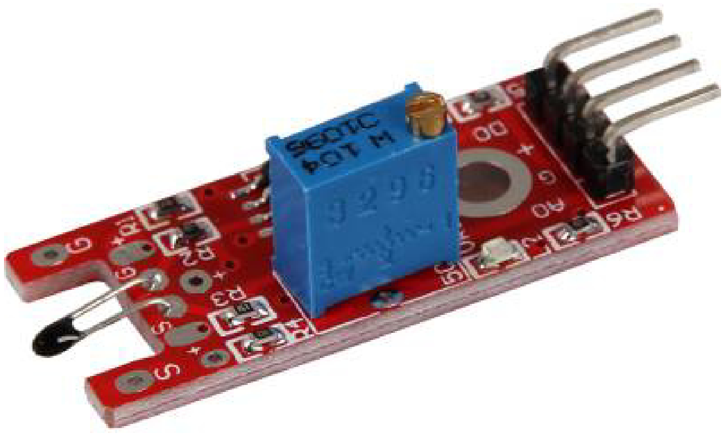
To start, enter the command:

```
sudo python KY-027-RPi-MagicLightCup.py
```

KY-028 Temperature Sensor module (Thermistor)

Contents

Picture



Technical data / Short description

Temperature measurement range: -55°C / +125°C This module includes a NTC Thermistor - its resistance falls with higher temperature.

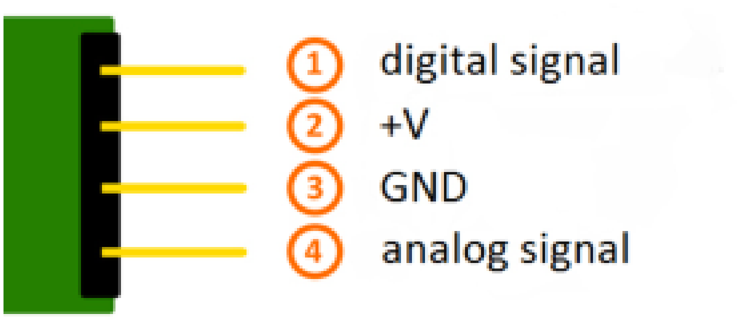
Digital Out: While measuring a temperature which is higher than the limit value, it will be shown here - you can set the limit value via potentiometer

Analog Out: Direct measurement of the sensor unit

LED1: Shows that the sensor is receiving power

LED2: Shows that a magnetic field was detected

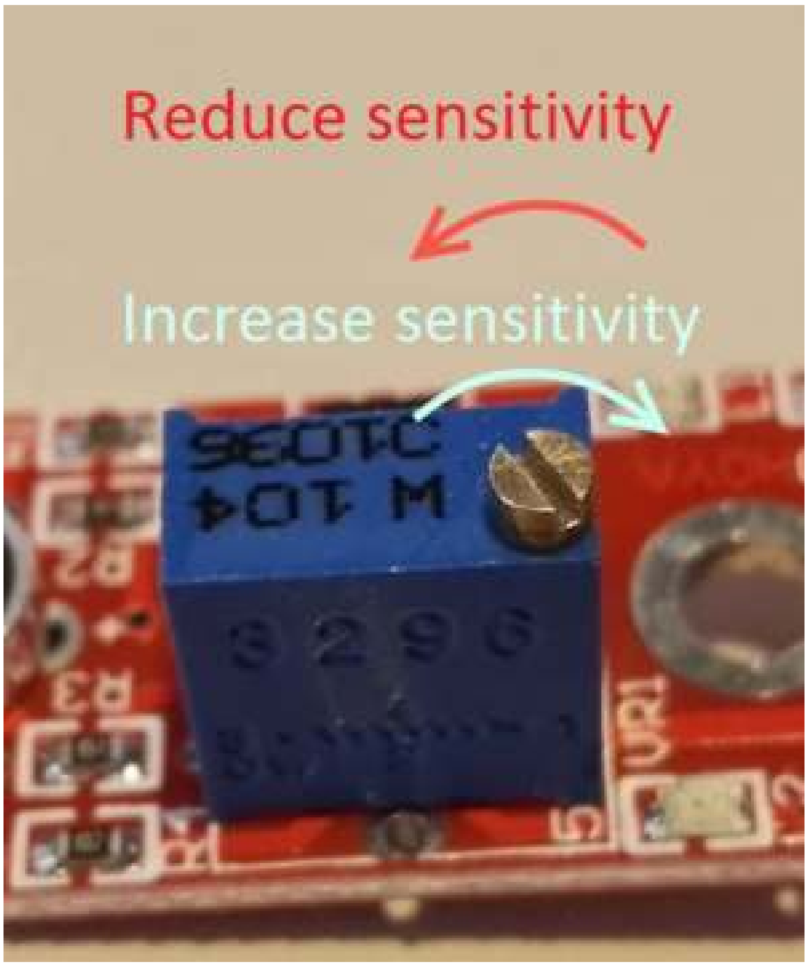
Pinout



Functionality of the sensor

The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.
The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.
You can control the sensitivity by adjusting the potentiometer.

Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.



This sensor doesn't show absolute values (like exact temperature in °C or magneticfield strenght in mT). It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

Code example Arduino

The program reads the current voltage value which will be measured at the output pin and shows it via serial interface.

Additionally, the status of the digital pin will be shown at the terminal which means if the extreme value was exceeded or not.

```
// Declaration and initialization of the input pins
int Analog_Eingang = A0; // X-axis-signal
int Digital_Eingang = 3; // Button

void setup ()
{
  pinMode (Analog_Eingang, INPUT);
  pinMode (Digital_Eingang, INPUT);
       
  Serial.begin (9600); // serial output with 9600 bps
}
   
// The program reads the current values at the input pins
// and outputs them at the serial output
void loop ()
{
  float Analog;
  int Digital;
     
  // Current values will be read and converted to voltage
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
  Digital = digitalRead (Digital_Eingang);
     
  //... and outputted here
  Serial.print ("Analog voltage value:"); Serial.print (Analog, 4);  Serial.print ("V, ");
  Serial.print ("Extreme value:");
   
  if(Digital==1)
  {
      Serial.println (" reached");
  }
  else
  {
      Serial.println (" not yet reached");
  }
  Serial.println ("----------------------------------------------------------------");
  delay (200);
}
```

Connections Arduino:

digital signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analog Signal	= [Pin 0]

Example program download

Analoger_Sensor

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and outputs it at the terminal in [mV].

Additionally, the status of the digital pin will be shown at the terminal to show if the extreme value was exceeded or not.

```
#######################################################################################

## 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
### ####################################################################################


# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import math, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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
```

```
# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

# Input pin for the digital signal will be picked here
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

#####################################################################################


# ########
# main program loop
# ########
# The program reads the current value of the input pin
# and shows it at the terminal

try:
        while True:
                #Current values will be recorded
                analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

                # Output at the terminal
                if GPIO.input(Digital_PIN) == False:
                        print "Analog voltage value:", analog,"mV, ","extreme value: not reached"
                else:
                        print "Analog voltage value:", analog, "mV, ", "extreme value: reached"
                print "---------------------------------------"

                sleep(delayTime)



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor

digital signal	=	GPIO 24	[Pin 18 (RPi)]
+V	=	3,3V	[Pin 1 (RPi)]
GND	=	GND	[Pin 06 (RPi)]
analog signal	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: analog signal]

Example program download

KY-028_Temperature-sensor-module_RPi

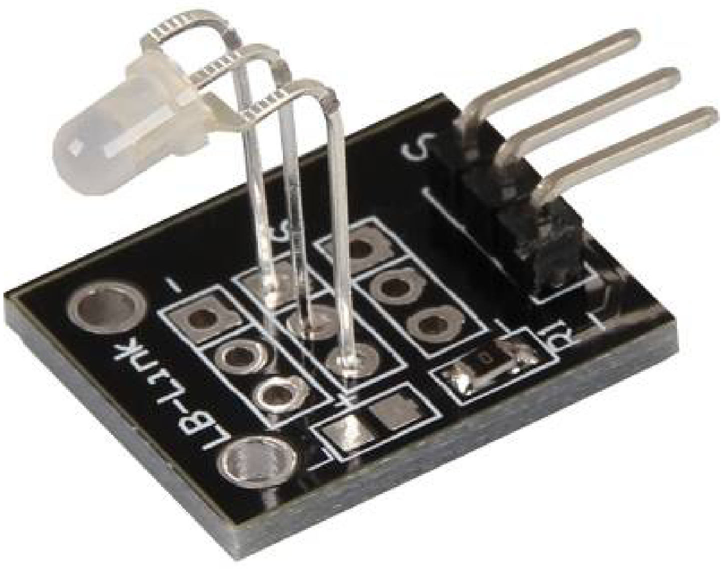
To start, enter the command:

```
sudo python KY-028_Temperature-sensor-module_RPi.py
```

KY-029 2-Color (Red+Green) 3mm LED module

Contents

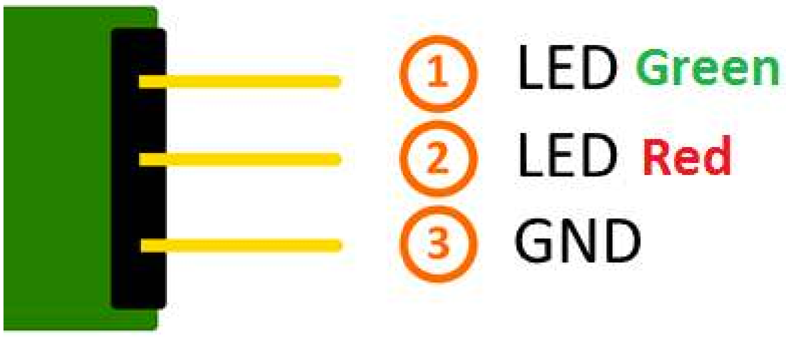
Pircture



Technical data / Short description

LED module which includes a red and a green LED. They are connected via cathode. You need resistors for different voltages.

Pinout



Code example Arduino

Code example ON/OFF

This example shows how you can switch the LEDs on and off in a 3 seconds time period via defined output pin.

```
int Led_Red = 10;
int Led_Green = 11;

void setup ()
{
  // Initialization of the Output pins for the LEDs
  pinMode (Led_Red, OUTPUT);
  pinMode (Led_Green, OUTPUT);
}

void loop () // main program loop
{
  digitalWrite (Led_Red, HIGH); // LED will switch to ON
  digitalWrite (Led_Green, LOW); // LED will switch to OFF
  delay (3000); // Wait mode for 3 seconds

  digitalWrite (Led_Red, LOW); // LED will switch to OFF
  digitalWrite (Led_Green, HIGH); // LED will switch to ON
  delay (3000); // Wait mode for another 3 seconds in which the LEDs will be switched
}
```

Example program ON/OFF download:

KY-029_LED_ON-OFF.zip

Code example PWM

You can regulate the brightness of the LEDs via pulse-width modulation. The LEDs will be switched ON and OFF for specific time periods, in which the relation between ON and OFF leads to a relative brightness, because of the Inertia of the human eyesight, the human eye interprets the ON/OFF as a brightness change. For more information to that theme visit: [Artikel von mikrokontroller.net].

This module provides 2 LEDs - with the overlay of the different brightness levels, you can create different colors. This will be shown in the following code example.

```
int Led_Red = 10;
int Led_Green = 11;

int val;

void setup () {
  // Initialization of the LED output pins
  pinMode (Led_Red, OUTPUT);
  pinMode (Led_Green, OUTPUT);
}
void loop () {
   // In this for-loop, the 2 LEDs will get different PWM-values
   // Via mixing the brightness of the different LEDs, you will get different colors
   for (val = 255; val> 0; val--)
      {
      analogWrite (Led_Green, val);
      analogWrite (Led_Red, 255-val);
      delay (15);
   }
   // You will go backwards through the color range in this second for loop.
   for (val = 0; val <255; val++)
      {
      analogWrite (Led_Green, val);
      analogWrite (Led_Red, 255-val);
      delay (15);
   }
}
```

Example program PWM download:

KY-029_LED_ON-OFF

Connections Arduino:

LED Green	=	[Pin 10]
LED Red	=	[Pin 11]
Sensor GND	=	[Pin GND]

Code example Raspberry Pi

Code example ON/OFF

In this example you will see how the LEDs will be switched on with a defined output pin, in a 3 second clock pulse.

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time
```

```
GPIO.setmode(GPIO.BCM)

# The output pins will be declared, which are connected with the LEDs.
LED_RED = 5
LED_GREEN = 4
GPIO.setup(LED_RED, GPIO.OUT, initial= GPIO.LOW)
GPIO.setup(LED_GREEN, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [press ctrl+c to end]"

# main program loop
try:
        while True:
                        print("LED RED is on for 3 seconds")
                        GPIO.output(LED_ROT,GPIO.HIGH) #LED will be switched on
                        GPIO.output(LED_GRUEN,GPIO.LOW) #LED will be switched off
                        time.sleep(3) # Wait mode for 3 seconds
                        print("LED GREEN is on for 3 seconds")
                        GPIO.output(LED_RED,GPIO.LOW) #LED will be switched off
                        GPIO.output(LED_GREEN,GPIO.HIGH) #LED will be switched on
                        time.sleep(3) # Wait mode for another 3 seconds

# Scavengin work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Example program ON/OFF download

KY-029_LED_ON-OFF-RB

To start, enter the command:

```
sudo python KY029_RPI_ON-OFF.py
```

Code example PWM

You can regulate the brightness of the LEDs via pulse-width modulation. The LEDs will be switched ON and OFF of for specific time periods, in which the relation between ON and OFF leads to a relative brightness, because of the Inertia of the human eyesight, the human eye interprets the ON/OFF as a brightness change. For more information to that theme visit: [Artikel von mikrokontroller.net].

This module provides a few LEDs - with the overlay of the different brightness levels, you can create different colors. This will be shown in the following code example. At the Raspberry Pi, only one Hardware-PWM channel is carried out unrestricted to the GPIO pins, why we have used Software-PWM at this example.

```
# Needed modules will be imported and configured
import random, time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

# Declaration of the output pins, which are connected with the LEDs.
LED_Red = 5
LED_Green = 4

# Set pins to output mode
GPIO.setup(LED_Red, GPIO.OUT)
GPIO.setup(LED_Green, GPIO.OUT)
```

```
Freq = 100 #Hz

# The different colors will be initialized.
RED = GPIO.PWM(LED_Red, Freq)
GREEN = GPIO.PWM(LED_Green, Freq)
RED.start(0)
GREEN.start(0)

# This function generate the actually color
# You can change the color with the specific color variable
# After the configuration of the color is finished, you will use time.sleep to
# configure how long the specific color will be displayed

def LED_color(Red, Green, pause):
    RED.ChangeDutyCycle(Red)
    GREEN.ChangeDutyCycle(Green)
    time.sleep(pause)

    ROT.ChangeDutyCycle(0)
    GRUEN.ChangeDutyCycle(0)

print "LED-Test [press ctrl+c to end]"

# main program loop:
# The task of this loop is to create for every single color an own variable.
# By mixing the brightness levels of the colors, you will get a color gradient.
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_color((x*i),(y*i),.02)

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Example program PWM download:

KY-029_LED_ON-OFF-RB-PWM

To start, enter the command:

```
sudo python KY029_RPI_PWM.py
```

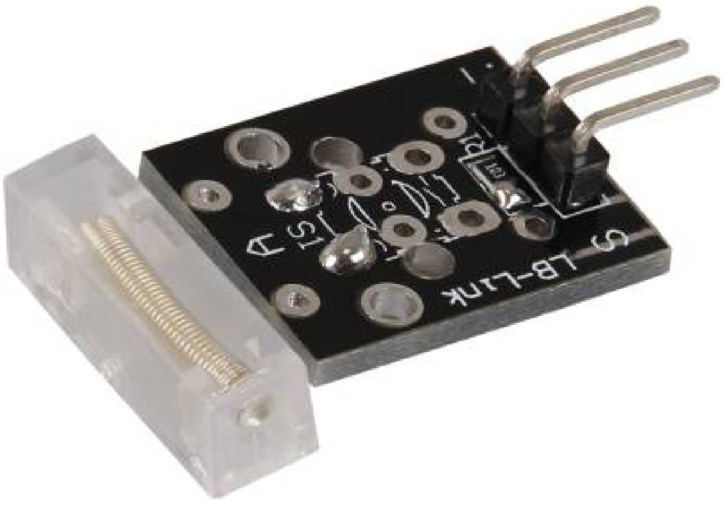
Connections Raspberry Pi:

LED Green	= GPIO4	[Pin 16]
LED Red	= GPIO5	[Pin 18]
Sensor GND	= GND	[Pin 6]

KY-031 Knock-sensor module

Contents

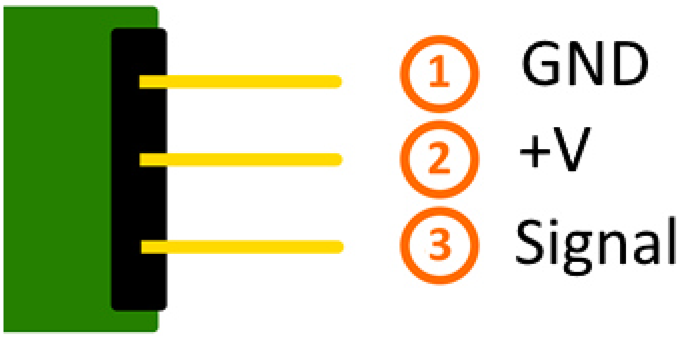
Picture



Technical data / Short description

On knocks or vibrations, the contact of the two input pins are connected.

Pinout



Code example Arduino

This example will light up a LED after the sensor detected a knock or vibration.

The modules KY-011, KY-016 or KY-029 can be used as an LED.

```
int Led = 13 ;// Declaration of the LED output pin
int Sensor = 10; // Declaration of the sensor input pin
int val; // Temporary variable

void setup ()
{
  pinMode (Led, OUTPUT) ; // Initialization output pin
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
}

void loop ()
{
  val = digitalRead (Sensor) ; // The current signal at the sensor will be read

  if (val == HIGH) // If a signal was detected , the LED will light up
  {
    digitalWrite (Led, LOW);
  }
  else
  {
    digitalWrite (Led, HIGH);
  }
}
```

Connections Arduino:

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

Sensor - = [Pin GND]

Example program download

SensorTest Arduino

Code example Raspberry Pi

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# The input pin of the sensor will be declared.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN)

print "Sensor-Test [press ctrl+c to end]"

# This output function will be started at signal detection
def outFunction(null):
        print("Signal detected")


GPIO.add_event_detect(GPIO_PIN, GPIO.FALLING, callback=outFunction, bouncetime=100)

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

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

SensorTest RPi

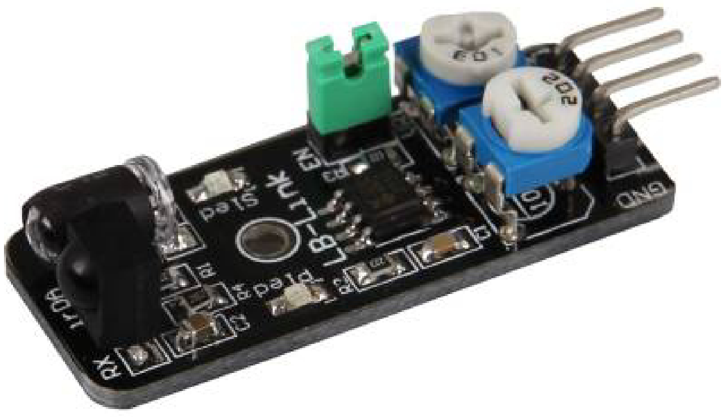
To start, enter the command:

```
sudo python SensorTest_RPi_withoutPullUP.py
```

KY-032 Obstacle-detect module

Contents

Picture

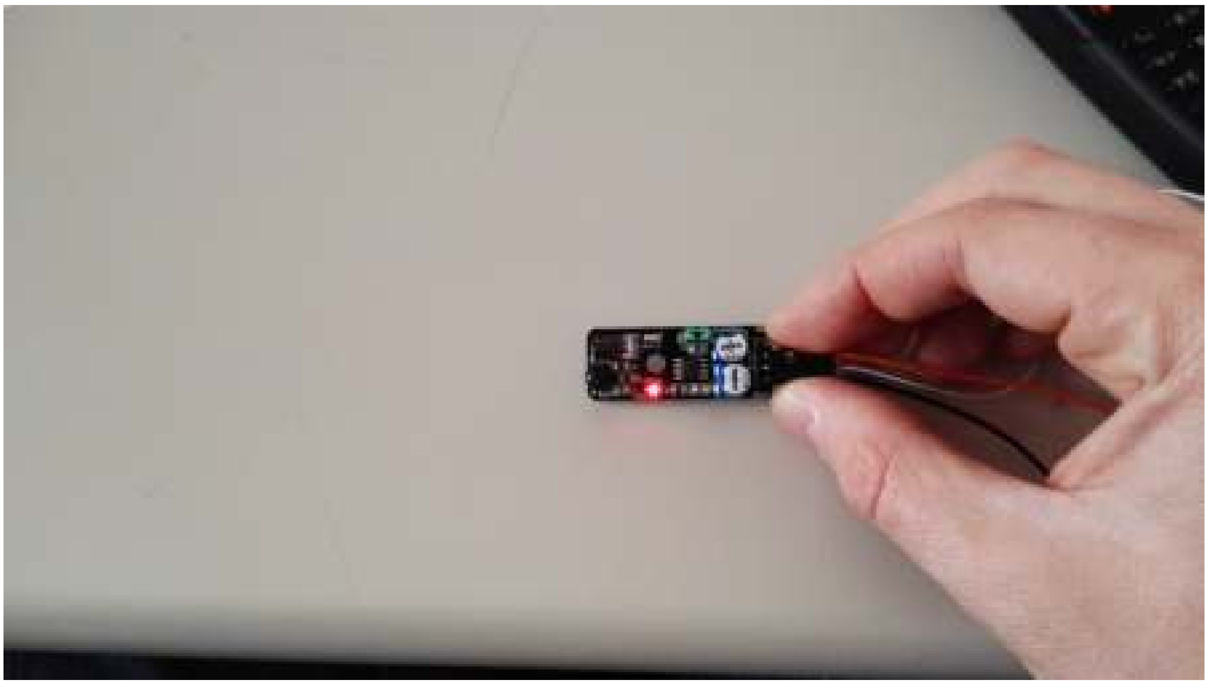


Technical data / Short description

If the sended infrared light hits an obstacle, the light will be reflected and detected by the photodiode. The detection range can be configured by the 2 controllers.

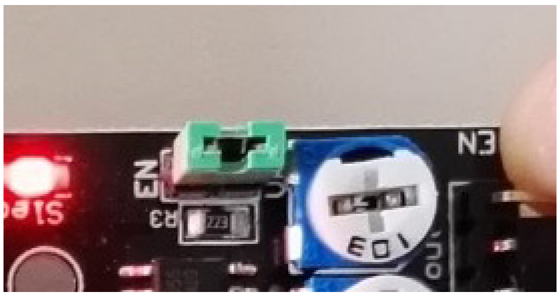
The behaviour of this can be used for Controllers, for example to stop a robot automatically which drives straight to an obstacle.

Condition 1: There is no obstacle in front of the detector [LED on the module: Off] [Sensor signal= Digital On]



Condition 2: Detector detects an obstacle [LED on the module: ON] [Sensor Signal= Digital Off]

This sensor allows you to activate and deactivate the obstacle detection by manipulating the enable-pin. It is activated by default which means that the sensor is in detection mode by default.
If you don't want that, you need to remove the jumper with the label "EN" and put a control signal on the enable pin.



Code example Arduino

This program reads the status of the sensor pin and prints it to the serial terminal if an obstacle was detected.

```
int Sensor = 10; // Declaration of the sensor input pin

void setup ()
{
  Serial.begin(9600); // Initialization serial output
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
}

// The program reads the current status of the sensor pin
// shows via serial console if the sensor detects a obstacle or not
void loop ()
{
  bool val = digitalRead (Sensor) ; // The current signal at the sensor will be read

  if (val == HIGH) // If a signal is detected, the LED will light up.
  {
    Serial.println("No obstacle");
  }
  else
  {
    Serial.println("Obstacle detected");
  }
  Serial.println("------------------------------------");
  delay(500); // Break of 500ms between each measurement
}
```

Connections Arduino:

Sensor enable	=	[N.C. (jumper plugged in)]
Sensor signal	=	[Pin 10]
Sensor +V	=	[Pin 5V]
Sensor GND	=	[Pin GND]

Example program download

KY-032_Obstacle-detection

Code example Raspberry Pi

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Declaration of the input pin which is connected with the sensor.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

# Break between the results will be defined here (in seconds)
delayTime = 0.5

print "Sensor-Test [press ctrl+c to end]"

# main program loop
try:
        while True:
            if GPIO.input(GPIO_PIN) == True:
                print "No obstacle"
            else:
                print "Obstacle detected"
            print "---------------------------------------"

            # Reset + Delay
            time.sleep(delayTime)

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Enable	=	-	[N.C. (jumper plugged in)]
Signal	=	GPIO24	[Pin 18]
+V	=	3,3V	[Pin 1]
GND	=	GND	[Pin 6]

Example program download

KY-032_Obstacle-detection_RPi

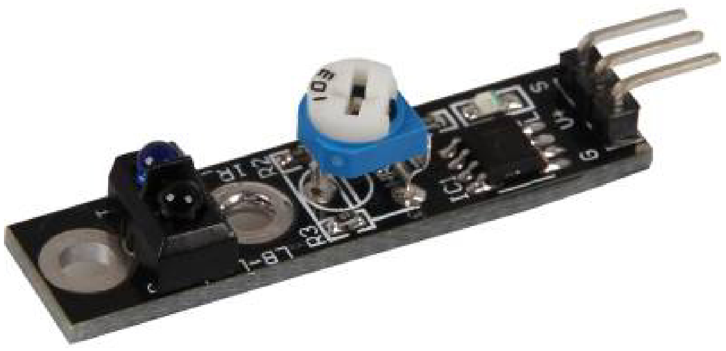
To start, enter the command:

```
sudo python KY-032_Obstacle-detection_RPi.py
```

KY-033 Tracking sensor module

Contents

Picture

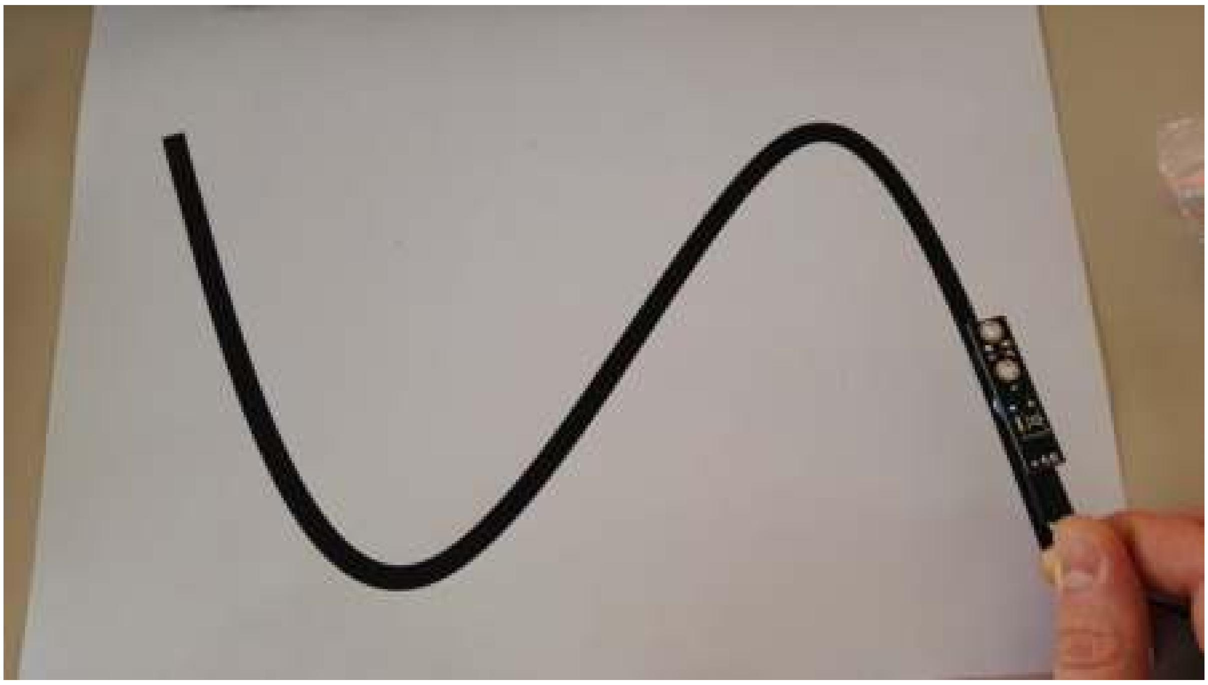


Technical data / Short description

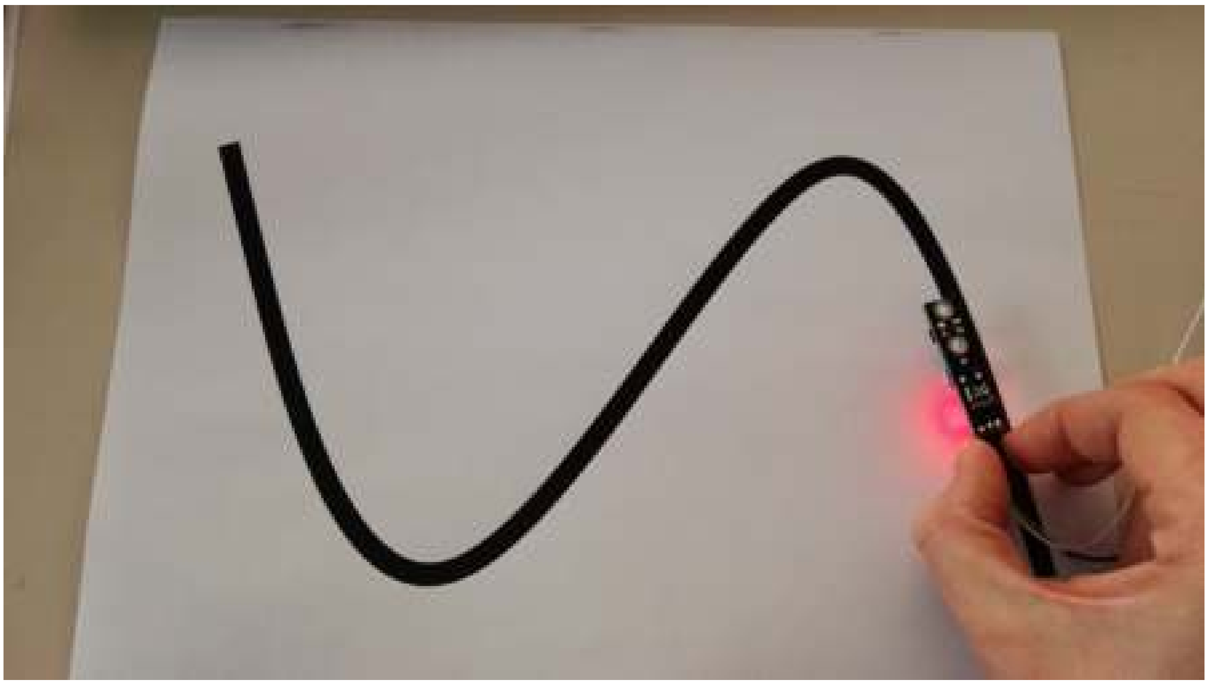
The sensor detects if a light reflecting or absorbing area is in front of it. It shows which of the 2 areas it is via digital output, as you can see in the picture below.
The Sensitivity (minimum range) of the sensor can be adjusted by the controller.

This behavior can be used to automatically follow a line with a robot.

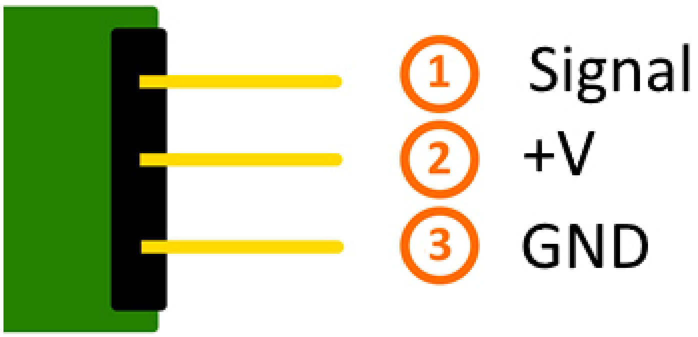
Condition 1: Line Tracker is on a line (not reflecting area) [LED on the module: Off] [Sensor signal= Digital On]



Condition 2: Line Tracker not on a line (reflecting area) [LED on the module: ON] [Sensor signal= Digital Off]



Pinout



Code example Arduino

```
int Sensor = 10; // Declaration of the sensor input pin

void setup ()
{
  Serial.begin(9600); // Initialization serial output
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
}

// The program reads the status of the sensor pins
// shows via serial terminal if the linetracker is on the line or not
void loop ()
{
  bool val = digitalRead (Sensor) ; // The current signal of the sensor will be read

  if (val == HIGH) // If a signal is detected the LED will light up.
  {
    Serial.println("LineTracker is on the line");
  }
  else
  {
    Serial.println("Linetracker is not on the line");
  }
  Serial.println("------------------------------------");
  delay(500); // Break of 500ms between the measurements
}
```

Connections Arduino:

Sensor signal	= [Pin 10]
Sensor +V	= [Pin 5V]
Sensor GND	= [Pin GND]

Example program download

KY-033_Tracking-sensor_ARD

Code example Raspberry Pi

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Declaration of the input pin which is connected with the sensor
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

# Break between the results will be defined here (in seconds)
delayTime = 0.5

print "Sensor-Test [press ctrl+c to end]"

# main program loop
try:
        while True:
            if GPIO.input(GPIO_PIN) == True:
                print "LineTracker is on the line"
            else:
                print "Linetracker is not on the line"
            print "---------------------------------------"

            # Reset + Delay
            time.sleep(delayTime)

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

KY-033_Tracking-sensor

To start, enter the command:

```
sudo python KY-033_Tracking-sensor.py
```

KY-034 7 Colour LED flash-module

Contents

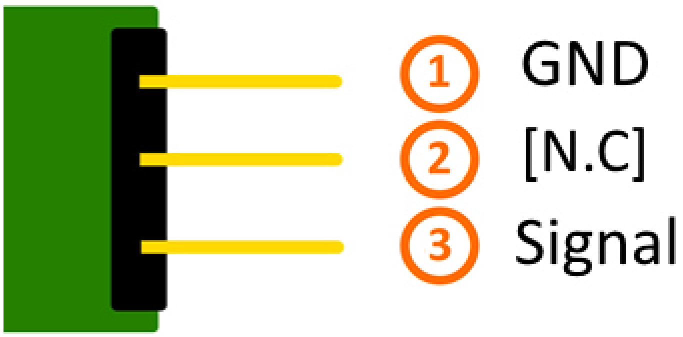
Picture

Technical data / Short description

If you connect this module with a power supply, a LED will light up which changes its colour automatically. It includes 7 different colours.

Voltage range: 3,3V - 5V

Pinout



Code example Arduino

This code example shows how you can switch a LED on for 4 seconds and than off for 2 seconds via defined output pin.

```
int Led = 13;

void setup ()
{
  pinMode (Led, OUTPUT); // Initialization of the LED output pin
}

void loop () // main program loop
{
  digitalWrite (Led, HIGH); // LED will be switched on
  delay (4000); // waitmode for 4 seconds
  digitalWrite (Led, LOW); // LED will be switched off
  delay (2000); // waitmode for another 2 seconds
}
```

Connections Arduino:

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

Example program download:

KY-034_7-color-led-flash-module

Code example Raspberry Pi

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Declaration of the input pin which is connected with the sensor.
# Additional to that the pull up resistor from the input will be activated.
LED_PIN = 24
GPIO.setup(LED_PIN, GPIO.OUT, initial= GPIO.LOW)

print "LED-Test [press ctrl+c to end]"

# main program loop
try:
        while True:
                print("LED is on for 4 seconds")
                GPIO.output(LED_PIN,GPIO.HIGH) #LED will be switched on
                time.sleep(4) # Waitmode for 4 seconds
                print("LED is off for 2 Sekunden")
                GPIO.output(LED_PIN,GPIO.LOW) #LED will be switched off
                time.sleep(2) # Waitmode for another 2 seconds

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

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

Example program download

KY-034_7-color-led-flash-module_RPi

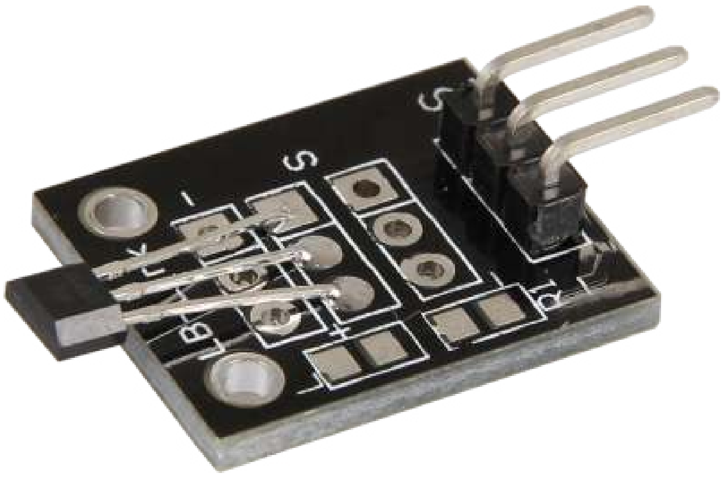
To start, enter the command:

```
sudo python KY-034_7-color-led-flash-module_RPi.py
```

KY-035 Bihor magnetic sensor module

Contents

Pictures

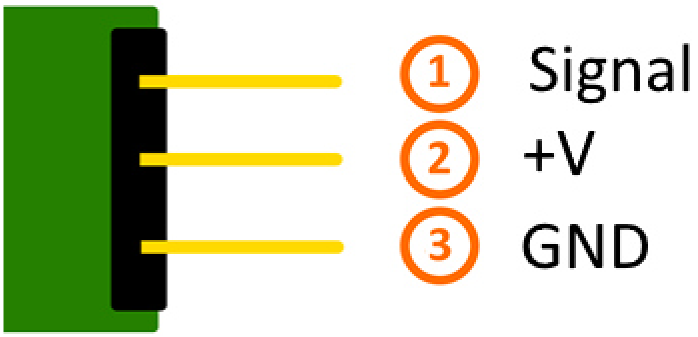


Technical data / Short description

Chipset: AH49E

The sensor measures the current magnetic field, near to the sensor.

Pinout



Code example Arduino

The program measures the current voltage value at the sensor, calculates with it and a known resistor the resistance from the sensor and shows the results via serial output.

```
int sensorPin = A5; // Declaration of the input pin

// Serial OUT in 9600 baud
void setup()
{
        Serial.begin(9600);
}

// The program measures the current voltage at the sensor,
// calculates the resistance with it and a known resistor
// and outputs it via serial OUT

void loop()
{
        // Measuring of the current voltage...
        int rawValue = analogRead(sensorPin);
        float voltage = rawValue * (5.0/1023) * 1000;

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

        // ... output via serial interface
        Serial.print("Voltage:");     Serial.print(voltage); Serial.print("mV");
        Serial.print(", Resistance:"); Serial.print(resitance); Serial.println("Ohm");
        Serial.println("---------------------------------------");

        delay(500);
}
```

Connections Arduino:

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

Sensor Signal = [Pin A5]

Example program download

KY-035_Bihor-magnetic-sensor-module

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and outputs it at the terminal in [mV].

Additianal to that, the status of the digital pin will be shown at the terminal to show if the extreme value was exceeded or not.

```
#######################################################################################

### 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
###
#######################################################################################

# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC
```

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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

# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

########################################################################################


# ########
# Main Loop
# ########
# Reading the values from the input pins and print to console

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

                # print to console
                print "Channel 0:", adc0, "mV "
                print "Channel 1:", adc1, "mV "
                print "Channel 2:", adc2, "mV "
                print "Channel 3:", adc3, "mV "
                print "---------------------------------------"

                time.sleep(delayTime)



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor

GND	= GND	[Pin 06 (RPi)]
+V	= 3,3V	[Pin 01 (RPi)]

analog Signal	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 17]
GND	=	GND	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: analog Signal]

Example program download

KY-053_RPi-AnalogDigitalConverter

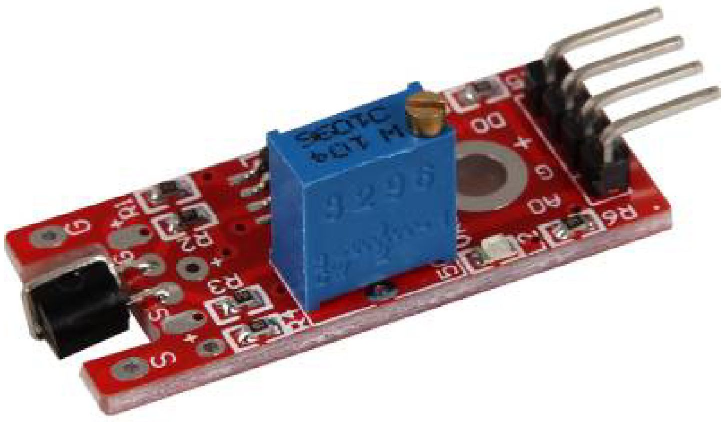
To start, enter the command:

```
sudo python KY-053_RPi-AnalogDigitalConverter.py
```

KY-036 Metal-touch sensor module

Contents

Picture



Technical data / Short description

Outputs a signal if the metal pike of the Sensor was touched. You can adjust the sensitivity of the sensor with the controller.

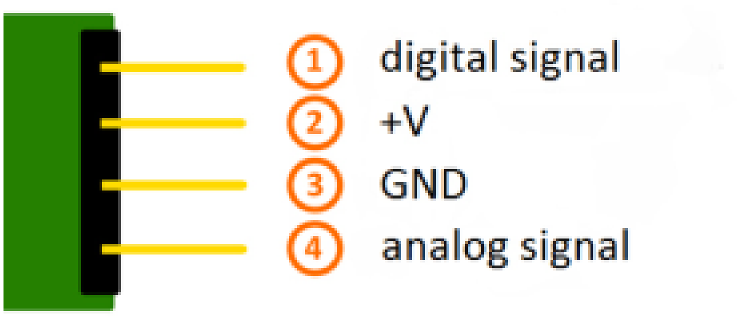
Digital Out: At the moment of contact detection, a signal will be outputted.

Analog Out: Direct measuring value of the sensor unit.

LED1: Shows that the sensor is supplied with voltage

LED2: Shows that the sensor detects a magnetic field

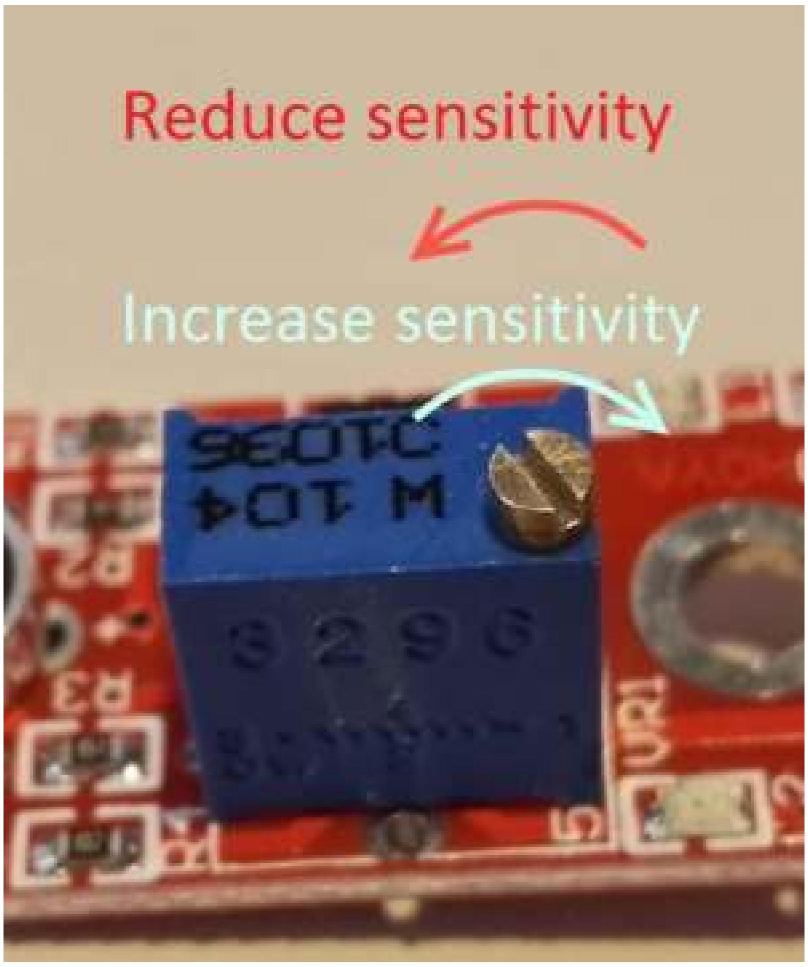
Pinout



Functionality of the sensor

The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.
The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.
You can control the sensitivity by adjusting the potentiometer.

Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.



This sensor doesn't show absolute values (like exact temperature in °C or magneticfield strenght in mT). It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

Code example Arduino

The program reads the current voltage value which will be measured at the output pin and shows it via serial interface.

Additional to that the status of the digital pin will be shown at the terminal which means if the extreme value was exceeded or not.

```
// Declaration and initialization of the input pin
int Analog_Eingang = A0; // X-axis-signal
int Digital_Eingang = 3; // Button

void setup ()
{
  pinMode (Analog_Eingang, INPUT);
  pinMode (Digital_Eingang, INPUT);
       
  Serial.begin (9600); // Serial output with 9600 bps
}
   
// The program reads the current value of the input pins
// and outputs it via serial out
void loop ()
{
  float Analog;
  int Digital;
     
  // Current value will be read and converted to the voltage
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0); 
  Digital = digitalRead (Digital_Eingang);
     
  // and outputted here
  Serial.print ("Analog voltage value:"); Serial.print (Analog, 4);  Serial.print ("V, ");
  Serial.print ("Extreme value:");
   
  if(Digital==1)
  {
      Serial.println (" reached");
  }
  else
  {
      Serial.println (" not reached yet");
  }
  Serial.println ("----------------------------------------------------------------");
  delay (200);
}
```

Connections Arduino:

digital Signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analog Signal	= [Pin 0]

Example program download

KY-036_metal-touch-sensor-module

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and outputs it at the terminal in [mV].

Additional to that, the status of the digital pin will shown at the terminak to show if the extreme value was exceeded or not.

```
#######################################################################################

### 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
###
#######################################################################################

# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import math, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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
```

```
# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

# Input pin for the digital signal will be picked here
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

########################################################################################


# ########
# main program loop
# ########
# The program reads the current value of the input pin
# and shows it at the terminal

try:
        while True:
                #Current values will be recorded
                analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

                # Output at the terminal
                if GPIO.input(Digital_PIN) == False:
                        print "Analog voltage value:", analog,"mV, ","extreme value: not reached"
                else:
                        print "Analog voltage value:", analog, "mV, ", "extreme value: reached"
                print "----------------------------------------"

                sleep(delayTime)



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor

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

ADS1115 - KY-053:

VDD	= 3,3V	[Pin 01]
GND	= GND	[Pin 09]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]
A0	= look above	[Sensor: analog signal]

Example program download

KY-036_Metal-touch-sensor-module_RPi

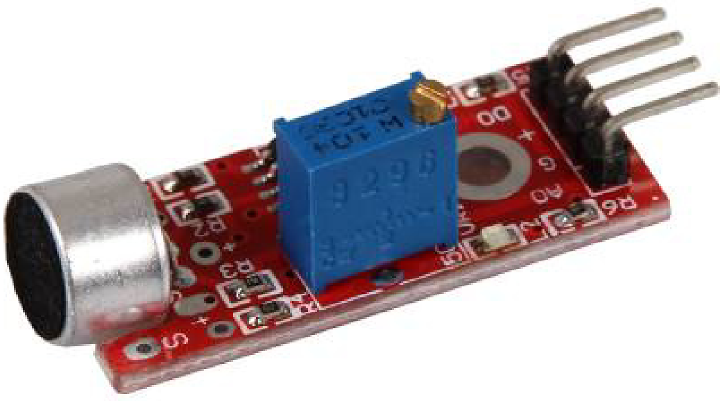
To start, enter the command:

```
sudo python KY-036_Metal-touch-sensor-module_RPi.py
```

KY-037 Microphone sensor module (high sensitivity)

Contents

Picture



Technical data / Short description

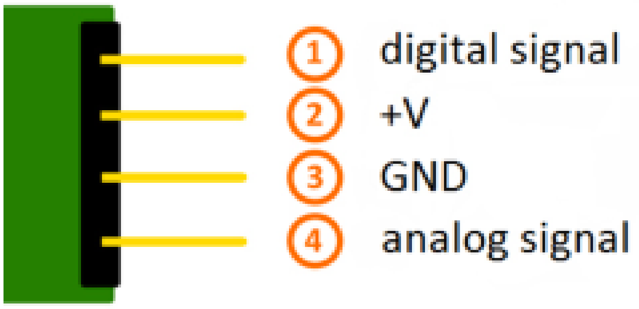
Digital Out: You can use a potentiometer to configure an extreme value for the sonic. If the value exceeds the extreme value, it will send a signal via digital out.

Analog Out: Direct microphone signal as voltage value

LED1: Shows that the sensor is supplied with voltage

LED2: Shows that a magnetic field was detected

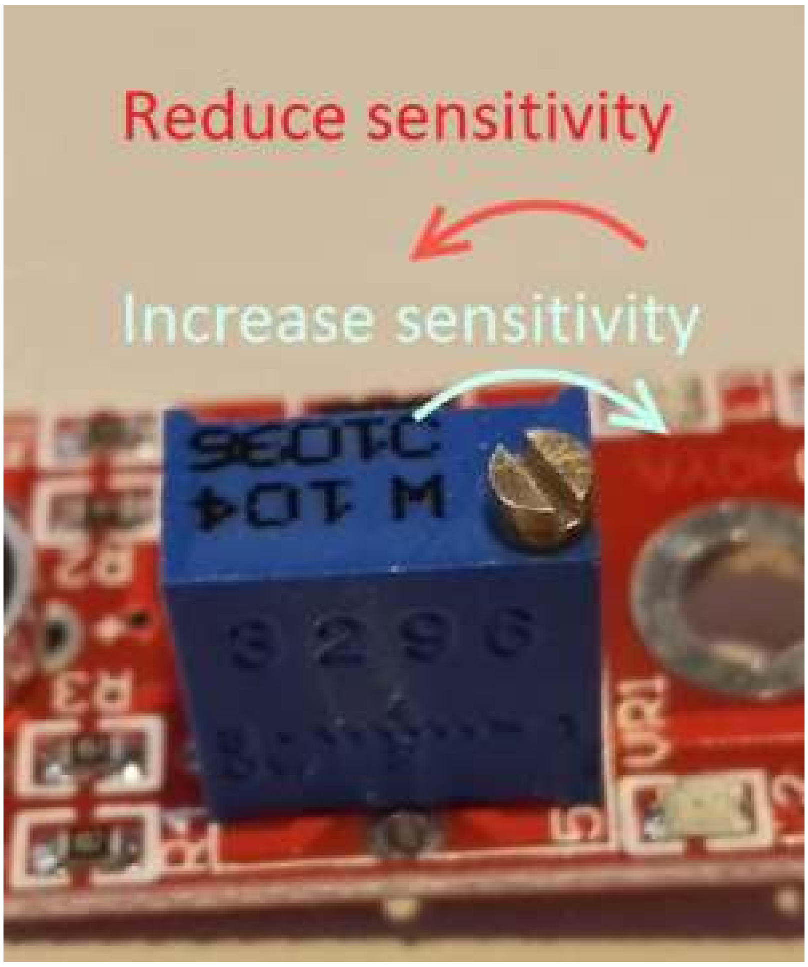
Pinout



Functionality of the sensor

The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.
The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.
You can control the sensitivity by adjusting the potentiometer.

Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.



This sensor doesn't show absolute values (like exact temperature in °C or magneticfield strenght in mT). It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

Code example Arduino

The program reads the current voltage value which will be measured at the output pin and shows it via serial interface.

Additional to that the status of the digital pin will be shown at the terminal which means if the extreme value was exceeded or not.

```
// Declaration and initialization of the input pin
int Analog_Eingang = A0; // X-axis-signal
int Digital_Eingang = 3; // Button

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

  Serial.begin (9600); // Serial output with 9600 bps
}

// The program reads the current value of the input pins
// and outputs it via serial out
void loop ()
{
  float Analog;
  int Digital;

  // Current value will be read and converted to voltage
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
  Digital = digitalRead (Digital_Eingang);

  //... and outputted here
  Serial.print ("Analog voltage value: "); Serial.print (Analog, 4);  Serial.print ("V, ");
  Serial.print ("Extreme value: ");

  if(Digital==1)
  {
      Serial.println (" reached");
  }
  else
  {
      Serial.println (" not reached yet");
  }
  Serial.println ("----------------------------------------------------------------");
  delay (200);
}
```

Connections Arduino:

digital signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analog signal	= [Pin 0]

Example program download

ARD_Analog-Sensor

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and outputs it at the terminal in [mV].

Additional to that, the status of the digital pin will be shown at the terminal to show if the extreme value was exceeded or not.

```
#######################################################################################
### 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
###
#######################################################################################


# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import math, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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

# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

# Input pin for the digital signal will be picked here
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

########################################################################################

# ########
# main program loop
# ########
# The program reads the current value of the input pin
# and shows it at the terminal

try:
        while True:
                #Current values will be recorded
                analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

                # Output at the terminal
                if GPIO.input(Digital_PIN) == False:
                        print "Analog voltage value:", analog,"mV, ","extreme value: not reached"
                else:
                        print "Analog voltage value:", analog, "mV, ", "extreme value: reached"
                print "---------------------------------------"

                sleep(delayTime)



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor

digital signal	=	GPIO 24	[Pin 18 (RPi)]
+V	=	3,3V	[Pin 1 (RPi)]
GND	=	GND	[Pin 06 (RPi)]
analog signal	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: analog signal]

Example program download

KY-037_Microphone_sensor_module_RPi

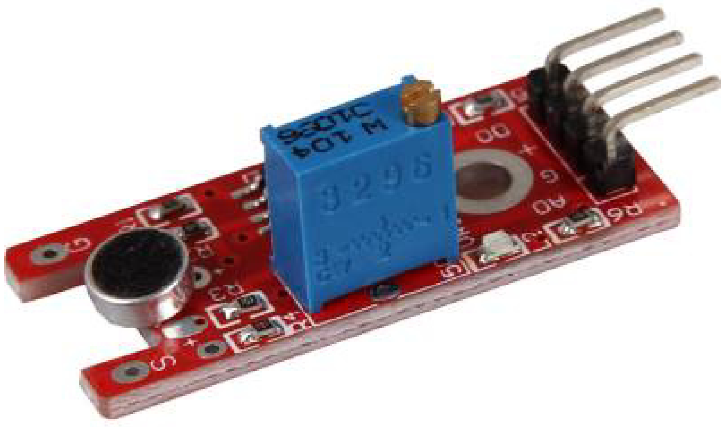
To start, enter the command:

```
sudo python KY-037_Microphone_sensor_module_RPi.py
```

KY-038 Microphone sound sensor module

Contents

Picture



Technical data / Short description

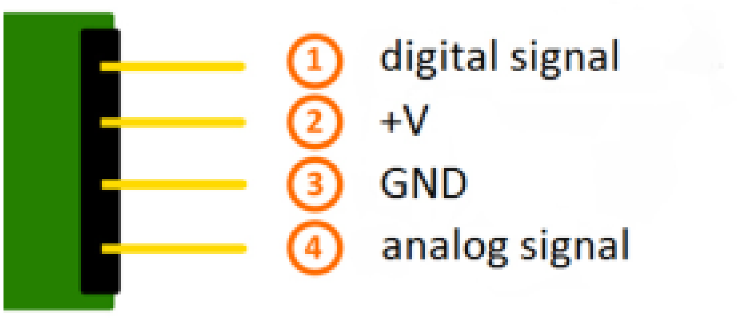
Digital Out: You can use a potentiometer to configure an extreme value for the sonic. IF the value exceeds the extreme value it will send a signal via digital out.

Analog Out: Direct microphone signal as voltage value

LED1: Shows that the sensor is supplied with voltage

LED2: Shows that a magnetic field was detected

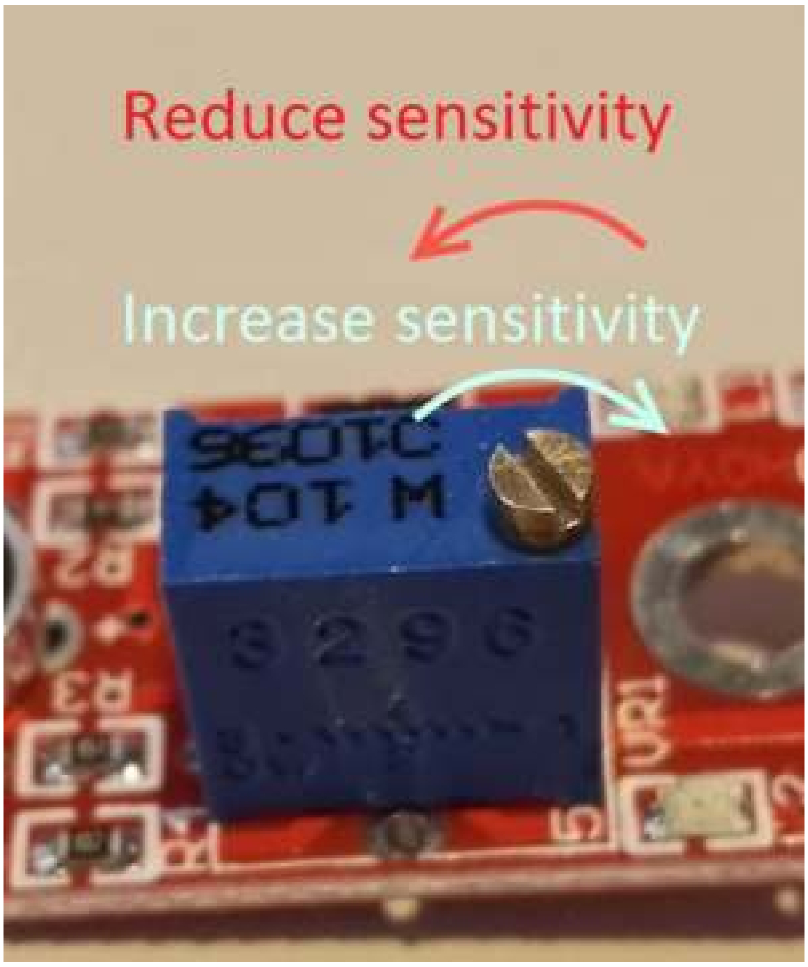
Pinout



Functionality of the sensor

The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.
The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.
You can control the sensitivity by adjusting the potentiometer.

Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.



This sensor doesn't show absolute values (like exact temperature in °C or magneticfield strenght in mT). It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

Code example Arduino

The program reads the current voltage value which will be measured at the output pin and shows it via serial interface.

Additional to that, the status of the digital pin will be shown at the terminal which means if the extreme value was exceeded or not.

```
// Declaration and initialization of the input pin
int Analog_Eingang = A0; // X-axis-signal
int Digital_Eingang = 3; // Button

void setup ()
{
  pinMode (Analog_Eingang, INPUT);
  pinMode (Digital_Eingang, INPUT);
       
  Serial.begin (9600); // Serial output with 9600 bps
}
   
// The program reads the current value of the input pins
// and outputs it via serial out
void loop ()
{
  float Analog;
  int Digital;
     
  // Current value will be read and converted to voltage
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0); 
  Digital = digitalRead (Digital_Eingang);
     
  //... and outputted here
  Serial.print ("Analog voltage value:"); Serial.print (Analog, 4);  Serial.print ("V, ");
  Serial.print ("Extreme value:");
   
  if(Digital==1)
  {
      Serial.println (" reached");
  }
  else
  {
      Serial.println (" not reached yet");
  }
  Serial.println ("----------------------------------------------------------------");
  delay (200);
}
```

Connections Arduino:

digital Signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analog Signal	= [Pin 0]

Example program download

ARD_Analog-Sensor

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and outputs it at the terminal in [mV].

Additional to that, the status of the digital pin will be shown at the terminal to show if the extreme value was exceeded or not.

```
#######################################################################################

### 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
###
#######################################################################################

# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import math, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC

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

# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (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
```

```
# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

# Input pin for the digital signal will be picked here
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

########################################################################################


# ########
# main program loop
# ########
# The program reads the current value of the input pin
# and shows it at the terminal

try:
        while True:
                #Current values will be recorded
                analog = adc.readADCSingleEnded(adc_channel_0, gain, sps)

                # Output at the terminal
                if GPIO.input(Digital_PIN) == False:
                        print "Analog voltage value:", analog,"mV, ","extreme value: not reached"
                else:
                        print "Analog voltage value:", analog, "mV, ", "extreme value: reached"
                print "---------------------------------------"

                sleep(delayTime)



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor

digital signal	=	GPIO 24	[Pin 18 (RPi)]
+V	=	3,3V	[Pin 1 (RPi)]
GND	=	GND	[Pin 06 (RPi)]
analog signal	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: analog signal]

Example program download

KY-038_Microphone_sensor_module_RPi

To start, enter the command:

```
sudo python KY-038_Microphone_sensor_module_RPir.py
```

KY-039 Heartbeat sensor module

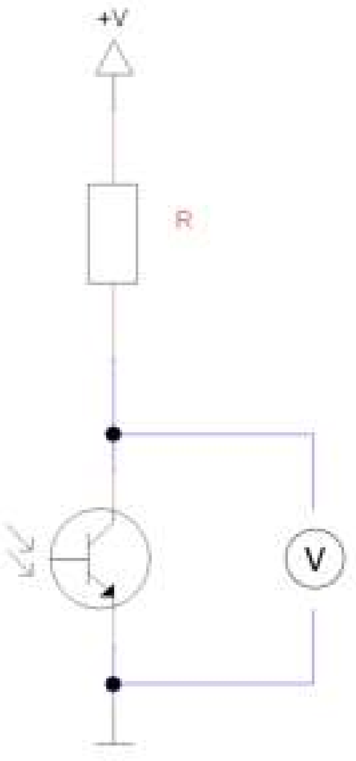
Contents

Picture

Technical data / Short description

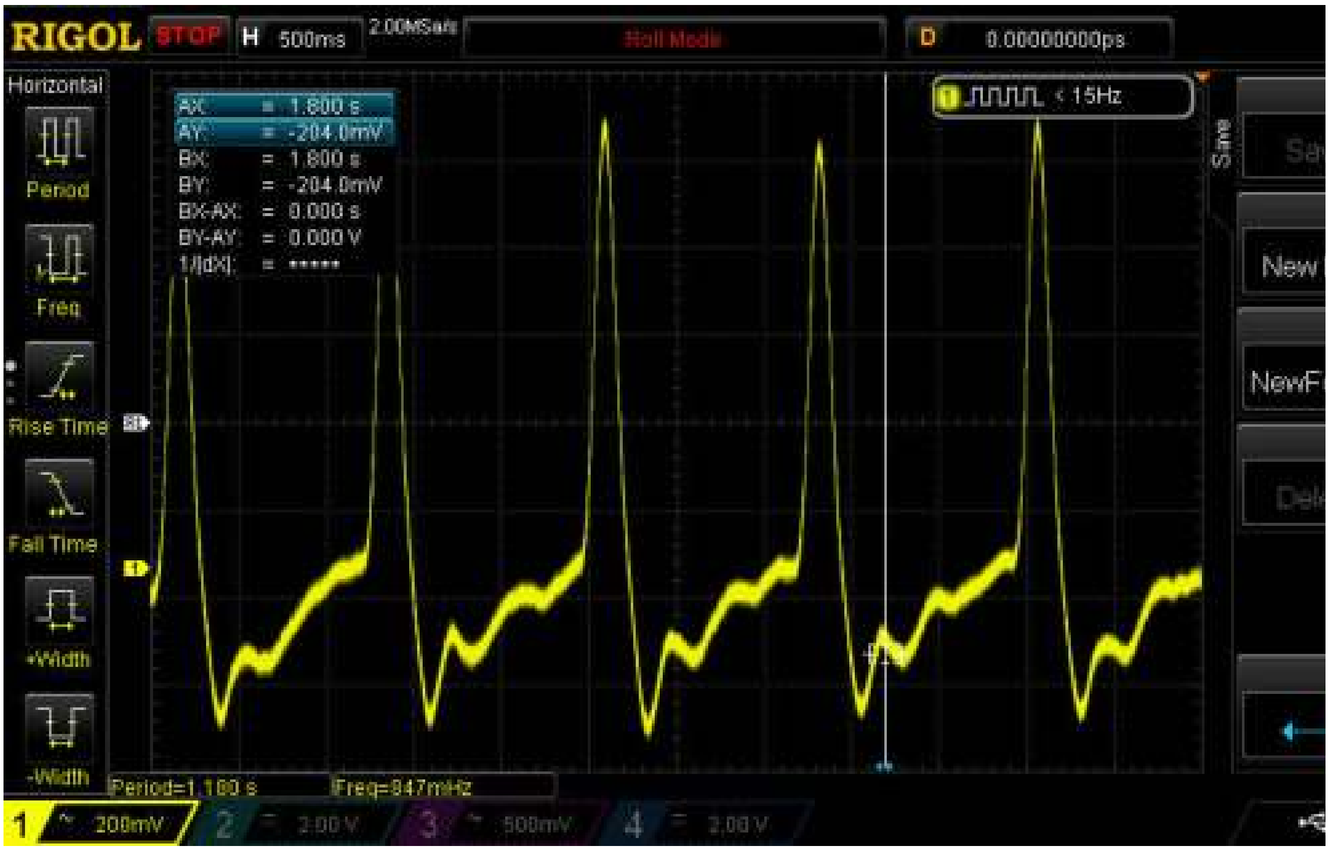
While sticking a finger between the infrared diode and the photo transistor you can detect the pulse at the signal out.

The explanation of the functionality of a photo transistor is simple: It works like a normal transistor - you will detect a higher electricity if the control voltage is higher. Instead of the control voltage, the photo transistor uses the light. If it it's a stronger light, you will measure a higher electricity.



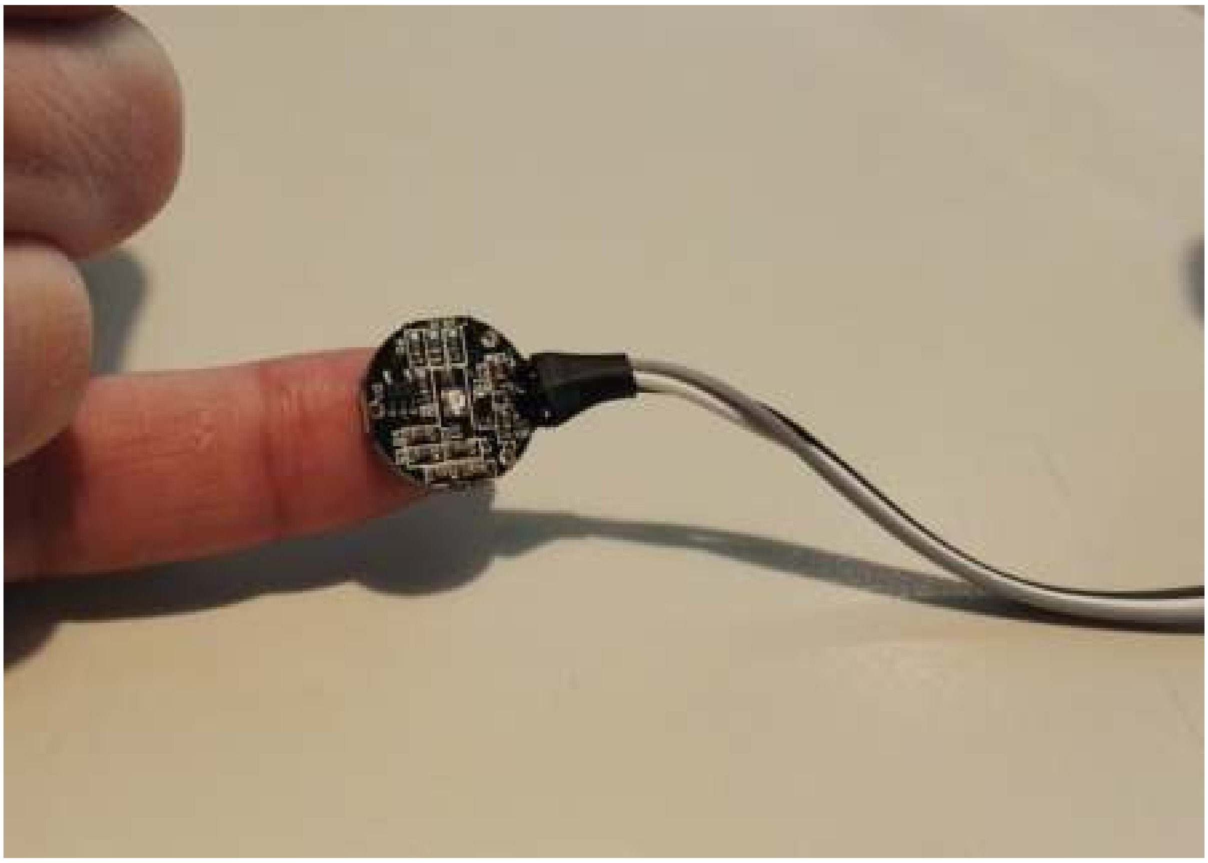
If you switch a resistor and a transistor in row, you will see a special behavior if you measure the voltage at th
transistor: if much light shines at the transistor, you will measure a very low voltage (near 0V).
If the transistor stays in the dark, you will measure a voltage near +V.

With this sensor module, which contains a phototransistor and an infrared diode, you can measure a pulse. Fc
you have to put a finger between the diode and the transistor. Explaination: exactly like you know it from a
flashlight, the light can shine through your skin. If you hit a vein with the light, you can see the pumping of th
a little bit. You can see it because the blood concentration is different on different areas in the vein, which me
you can see brightness differences in the blood flow. Exactly this differences can be measured with the senso
module and with that you will get the pulse. You can see it clear at the oszilloskop picture.

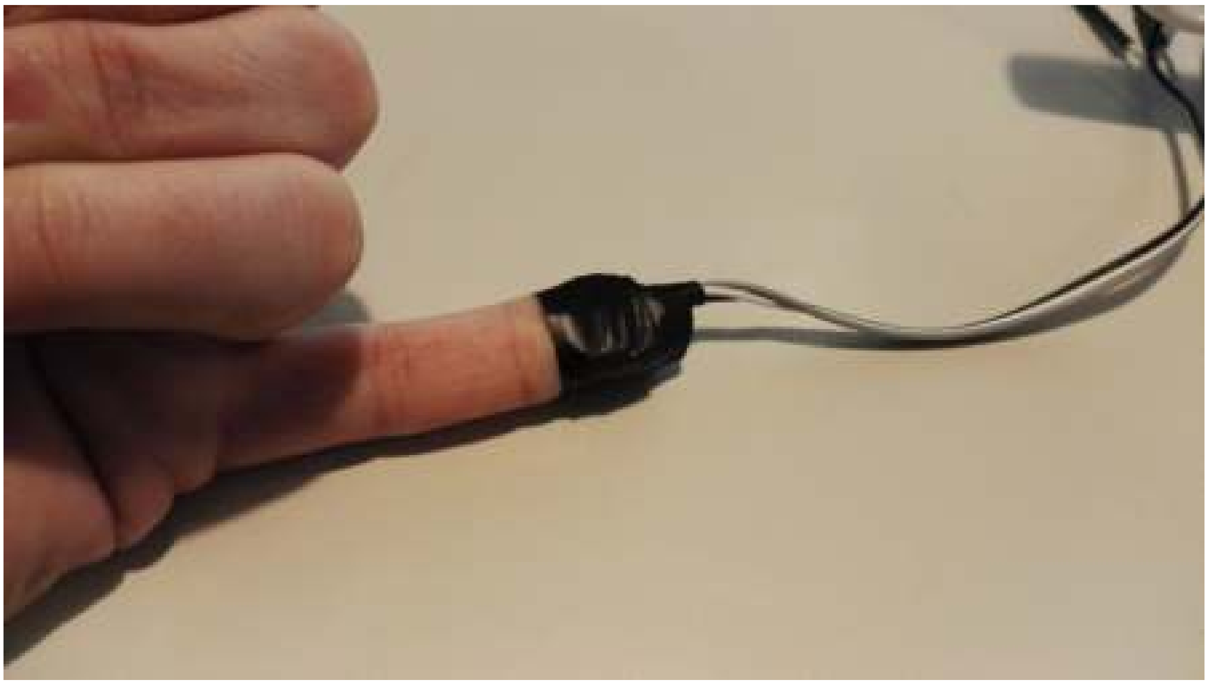


This shows, the changes of the voltage at the phototransistor - and with it the brightness change which comes from the flowing blood. The peaks above show the heartbeat. If you calculate the measured beats per recorded time, you will nearly get 71 beats per minute (bpm).

Additionally to that, you can see that the signal from the transistor is really low or the transistor is really sensitive to measure the small signal. For optimal results, we advice to prepare the module like it is shown in the pictures below:

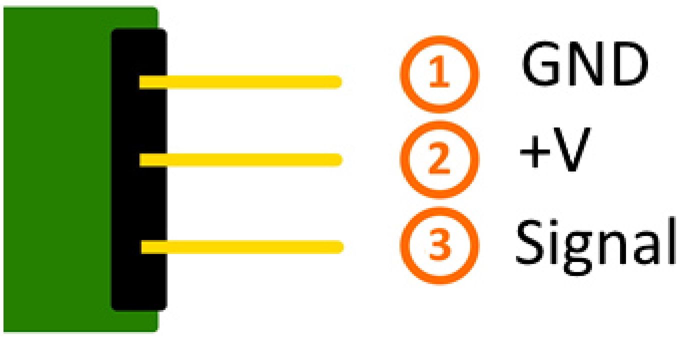


To increase the sensitivity of the sensor, we advise to fix it at your finger with tape.



The heartbeat will be measured really good if the sensor is exactly above a big vein.
To get a better signal, we advise you to change the position of the sensor, if necessary.

Pinout



Pinout:

Gray -> GND

White -> +V

Black -> Signal

Code example Arduino

The following code example was written by Dan Truong. He published it under the following [[1]] . It is available under the [|MIT OpenSource License].

The example below is a german translation, you can get the original via download.

This code shows a so called "Peak-detection". It doesn't record a heartbeat history, instead of that it looks for peaks in the recorded data, which was detected as heartbeat and shown with a LED. You can calculate the pulse with the known delay. If you move the finger to strong, it could take some time till the program reacts to the new situation and gives out the correct data.

```
////////////////////////////////////////////////////////////////////////
/// 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://http://opensource.org/licenses/MIT
///
/// KY039 Arduino Heartrate Monitor V1.0 (April 02, 2015)
```

```
////////////////////////////////////////////////////////////////////////

// German Comments by Joy-IT


////////////////////////////////////////////////////////////////////////
/// @param[in] IRSensorPin Analog PI which is connected with the sensor
/// @param[in] delay (msec) The delay between the calls of the sample function
//                   You will get the best results if take 5 sample for each heart beat
///                  Not slower than 150 mSec for 70 BPM pulse
///                  Better take 60 mSec for a pulse of 200 BPM.
///
/// @Short description
/// This code is a so called Peak-detection.
/// It doesn't record a heart beat history,
/// instead it will search for peaks in the recorded data,
/// and show them via LED. While knowing the delays you can calculate the pulse.
////////////////////////////////////////////////////////////////////////

int rawValue;


bool
heartbeatDetected(int IRSensorPin, int delay)
{
  static int maxValue = 0;
  static bool isPeak = false;


  bool result = false;

  rawValue = analogRead(IRSensorPin);
  // Measuring of the voltage value at the photo transistor
  rawValue *= (1000/delay);

  // If the difference of the current value and the last value is to high
  // ( maybe because you moved the finger away from the sensor)
  // maxValue will be resetted to get a new basic
  if (rawValue * 4L < maxValue) {maxValue = rawValue * 0.8;}    // Detect new peak
       if (rawValue > maxValue - (1000/delay)) {
    // The peak will be detected here. If the new rawValue is bigger than
        // the last maxValue, it will be detected as a peak.
    if (rawValue > maxValue) {
      maxValue = rawValue;
    }
    // Allocate only one heartbeat to a peak.
    if (isPeak == false) {
      result = true;
    }
    isPeak = true;
  } else if (rawValue < maxValue - (3000/delay)) {
    isPeak = false;
    // Here the maxValue will be decreased at each run
    // because if you don't do that the value would be every time lower or the same.
    // Also if you move the finger a bit the signal would be weaker without that.
    maxValue-=(1000/delay);
 }
  return result;
}

////////////////////////////////////////////////////////////////////////
// Arduino main code
////////////////////////////////////////////////////////////////////////
int ledPin=13;
int analogPin=0;

void setup()
```

```
{
  // The included LED of the Arduino (Digital 13), will be used as output here.
  pinMode(ledPin,OUTPUT);

  // Serial output initialization
  Serial.begin(9600);
  Serial.println("Heartbeat detection example code");
}

const int delayMsec = 60; // 100msec per sample

// The main program has two tasks:
// - The LED will light up after detecting a heart beat
// - Calculating of the pulse and outputting of it at the serial out.

void loop()
{
  static int beatMsec = 0;
  int heartRateBPM = 0;
      Serial.println(rawValue);
  if (heartbeatDetected(analogPin, delayMsec)) {
    heartRateBPM = 60000 / beatMsec;
        // LED-output for the heart beat heart beat
    digitalWrite(ledPin,1);

    // Output of the serial data
    Serial.print(rawValue);
    Serial.print(", ");
    Serial.println(heartRateBPM);

    beatMsec = 0;
  } else {
    digitalWrite(ledPin,0);
  }
  delay(delayMsec);
  beatMsec += delayMsec;
}
```

Connections Arduino:

Sensor Signal	= [Pin 0]
Sensor +V	= [5V]
Sensor -	= [Pin GND]

Example program download

KY-039-HeartBeatDetector original by DanTruong

KY-039_Heartbeat-sensor-module with english comments by Joy-IT

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

Normally, the function to detect a heartbeat should start after the configured delayTime (Standard: 10 ms). If it detects a heartbeat, it will output the pulse. Additional to that you can connect a LED with the LED pin (standard: GPIO24) to visualise the detected heartbeat.

If you move the finger to strong or take it back and put it to the sensor again, the program needs a few seconds (3-5 seconds) until it will give you the new, correct data.

```
#######################################################################################
### 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
#######################################################################################




# This code is using the ADS1115 and the I2C python libraries for the Raspberry Pi
# It is published by BSD License under the following link
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Additional needed modules will be imported and configured
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Used variables will be initialized
beatsPerMinute = 0
isPeak = False
result = False
delayTime = 0.01
maxValue = 0
schwelle = 25
beatTime = 0
oldBeatTime = 0

# Address allocation ADS1x15 ADC
```

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

# Amplification (Gain) will be picked
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# Sample rate of the ADC (SampleRate) will be picked
sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
# sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# ADC-Channel (1-4) will be picked
adc_channel = 0    # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# the ADC which is used by the KY-053 is an ADS1115 chipset adc = ADS1x15(ic=ADS1115)

# LED output pin declaration.
LED_PIN = 24
GPIO.setup(LED_PIN, GPIO.OUT, initial= GPIO.LOW)



########################################################################################

# Buzzer output pin declaration.
def heartBeatDetect(schwelle):
        global maxValue
        global isPeak
        global result
        global oldBeatTime

        # Reading of the voltage at the photo transistor
        # saving of the voltage value into the rawValue variable
        # With "adc_channel" the channel which is connected with the ADC will be picked.
        rawValue = adc.readADCSingleEnded(adc_channel, gain, sps)

        # Reset of the result-variable
        if result == True:
            result = False

        # If the difference between the current value and the last maximum value is to big
        # (maybe bacause you moved the finger to strong for example)
        # Here you see the reset of the maxValue to get a new basic.
        if rawValue * 4 < maxValue: maxValue = rawValue * 0.8; # Detecting of the peak

              if rawValue > maxValue:
                    maxValue = rawValue
              if isPeak == False:
                    result = True

              isPeak = True

        else:

            if rawValue < maxValue - schwelle:
```

```
            if rawValue < maxValue - schwelle:
              isPeak = False
             # Here the max value will be decreased at each run
             # because if you don't do that the value would be every time lower or the same.
             # Also if you move the finger a bit the signal would be weaker without that.

            maxValue = maxValue - schwelle/2

        # If a heartbeat was detected above, the Output will start.
        if result == True:

           # Calculating of the pulse
           # Here the system time will be recorded for every heartbeat
           # At the next heartbeat, the system time will be compared with the last recorded one
           # The difference between them is the time between the heartbeats
           # with that you can also calculate the pulse
            beatTime = time.time()
            timedifference = beatTime - oldBeatTime
            beatsPerMinute = 60/timedifference
            oldBeatTime = beatTime

           # the heartbeat will light up a LED for a short time
            GPIO.output(LED_PIN, GPIO.HIGH)
            time.sleep(delayTime*10)
            GPIO.output(LED_PIN, GPIO.LOW)

            # Calculated pulse will be given to the function
            return beatsPerMinute

########################################################################################


# ########
# Main program loop
# ########
# After the "delayTime" (standard: 10ms) the function to detect a heartbeat will start.
# After detecting a heartbeat, the pulse will be outputted.

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



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connection Raspberry Pi:

Sensor KY-039

Signal	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]
+V	=	3,3V	[Pin 1]
GND	=	GND	[Pin 6]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]

SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: Signal]

Example program download

KY-039_Heartbeat-sensor_RPi

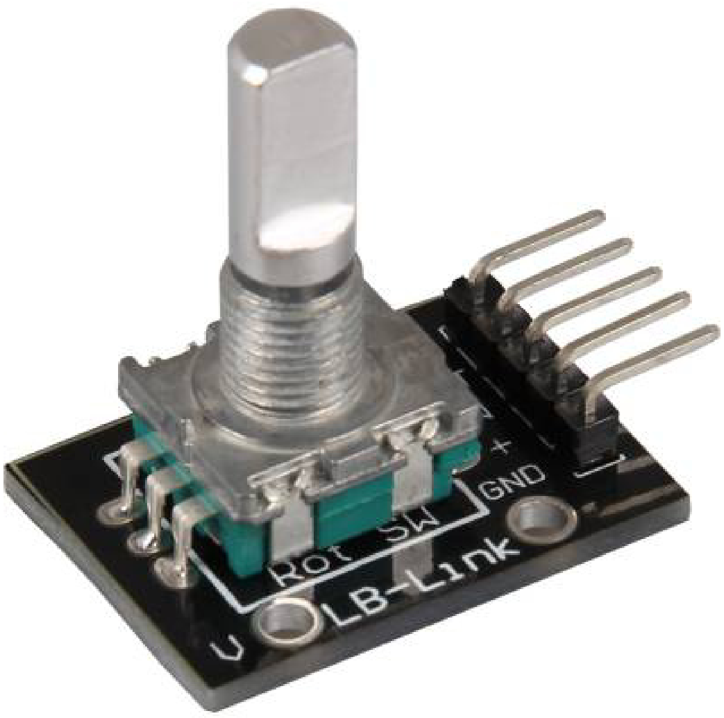
To start, enter the command:

```
sudo python KY-039_Heartbeat-sensor_RPi.py
```

KY-040 Rotary encoder

Contents

Picture



Technical data / Short description

The current position of the rotary switch will be send encoded to the output.

Encoding

The idea of the rotary switch is that with every "step" only one of the conditions will change. In order of which status have changed first, you can see the rotational direction if you look at the following encoding.

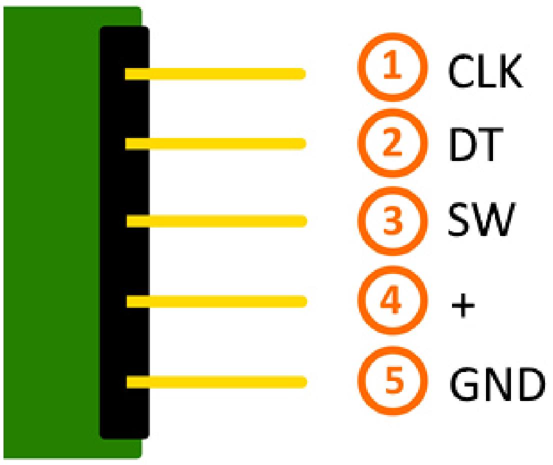
Clockwise [A will change first] -> Pin_CLK

A	B
0	0
1	0
1	1
0	1
0	0

<u>Counterclockwise</u> [B will change first] -> Pin_DT

A	B
0	0
0	1
1	1
1	0
0	0

Pinout



Code example Arduino

The program checks which pin has changed first, to get the rotational direction, after detecting a change at the pin status. You will get the rotational direction after you compare the current status of the pins with the last status. After detecting the rotational direction, the steps from the start will be counted and outputted. Pushing the button of the rotary encoder will reset the current position.

For serial output: Baudrate = 115200

```
// Initialization of the needed variables<br />
int Counter = 0;
boolean Richtung;
int Pin_clk_Letzter;
int Pin_clk_Aktuell;

// Definition of the input-pins
int pin_clk = 3;
int pin_dt = 4;
int button_pin = 5;


void setup()
{
   // Initialization of the input-pins...
   pinMode (pin_clk,INPUT);
   pinMode (pin_dt,INPUT);
   pinMode (button_pin,INPUT);

   // ...and activating of their pull up resistors
   digitalWrite(pin_clk, true);
   digitalWrite(pin_dt, true);
   digitalWrite(button_pin, true);

   // Initial reading of the Pin_CLK
   Pin_clk_Letzter = digitalRead(pin_clk);
   Serial.begin (115200);
 }

// The program checks, which of the status pins have changed first

void loop()
{
   // Reading of the current status
   Pin_clk_Aktuell = digitalRead(pin_clk);

   // Check for a Change
   if (Pin_clk_Aktuell != Pin_clk_Letzter)
   {

                if (digitalRead(pin_dt) != Pin_clk_Aktuell)
                {
                        // Pin_CLK has changed first
                        Counter ++;
                        Richtung = true;
                }

                else
                {       // Else Pin_DT changed first
                        Richtung = false;
                        Counter--;
                }

                Serial.println ("Rotation detected: ");
```

```
                  Serial.println ("Rotation detected: ");
                  Serial.print ("Rotational direction: ");

                  if (Richtung)
                  {
                     Serial.println ("Clockwise");
                  }
                  else
                  {
                     Serial.println("Counterclockwise");
                  }

                  Serial.print("Current position: ");
                  Serial.println(Counter);
                  Serial.println("------------------------------");

     }

     // Preparation for the next run:
     // The current value will be the last value for the next run.
     Pin_clk_Letzter = Pin_clk_Aktuell;

     // Reset funciton to save the current position
     if (!digitalRead(button_pin) && Counter!=0)
       {
         Counter = 0;
         Serial.println("Position resetted");
       }

   }
```

Connections Arduino:

CLK	=	[Pin 3]
DT	=	[Pin 4]
Button	=	[Pin 5]
+	=	[Pin 5V]
GND	=	[Pin GND]

Code example download

KY-40_rotary-encoder_ARD

Code example Raspberry Pi

The program checks which pin has changed first, to get the rotational direction, after detecting a change at the pin status. You will get the rotational direction after you compare the current status of the pins with the last status. After detecting the rotational direction, the steps from the start will be counted and outputted. Pushing the button of the rotary encoder will reset the current position.

```
# coding=utf-8
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Declaration and initialisation of the input pins which are connected with the sensor.
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)

# Needed variables will be initialised
Counter = 0
Richtung = True
PIN_CLK_LETZTER = 0
PIN_CLK_AKTUELL = 0
delayTime = 0.01

# Initial reading of Pin_CLK
PIN_CLK_LETZTER = GPIO.input(PIN_CLK)

# This output function will start at signal detection
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
            Richtung = True;
        else:
            Richtung = False
            Counter = Counter - 1

        print "Rotation detected: "

        if Richtung:
            print "Rotational direction: Clockwise"
        else:
            print "Rotational direction: Counterclockwise"

        print "Current position: ", Counter
        print "------------------------------"

def CounterReset(null):
    global Counter

    print "Position reset!"
    print "------------------------------"
    Counter = 0

# To include a debounce, the output function will be initialised from the GPIO Python Module
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 [press ctrl-c to end]"

# Main program loop
try:
        while True:
            time.sleep(delayTime)

# Scavenging work after the end of the program
except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

CLK	=	GPIO16	[Pin 36]
DT	=	GPIO15	[Pin 10]
SW	=	GPIO14	[Pin 8]
+	=	3,3V	[Pin 1]
GND	=	GND	[Pin 6]

Example program download

KY-040_rotary-encoder_RPi

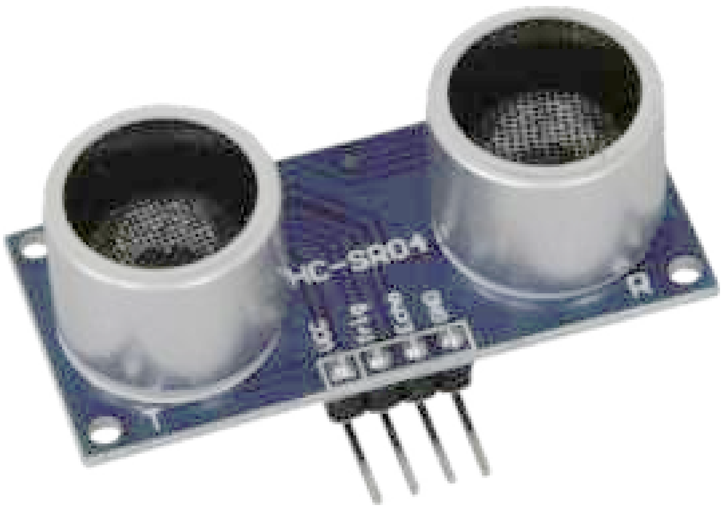
To start, enter the command:

```
sudo python KY-040_rotary-encoder_RPi.py
```

KY-050 Ultrasonic-distance-sensor

Contents

Picture



Technical data / Short description

If the trigger input gets a signal, the distance measurement will start and the result will be sended as a PWM-TTL signal via echo-output.

measureable distance: 2cm—300cm **measurement resolution**: 3mm

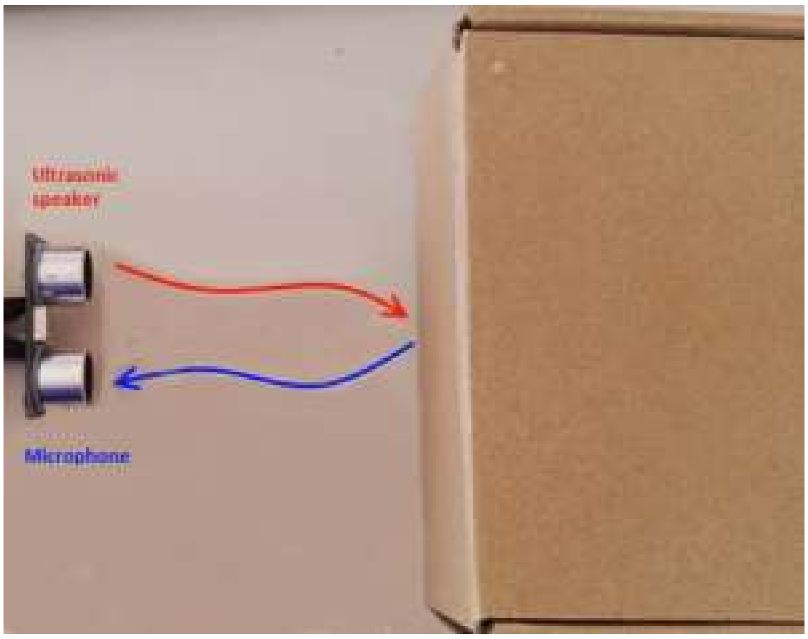
min. time between the measurements 50µs

Pinout

Function principle

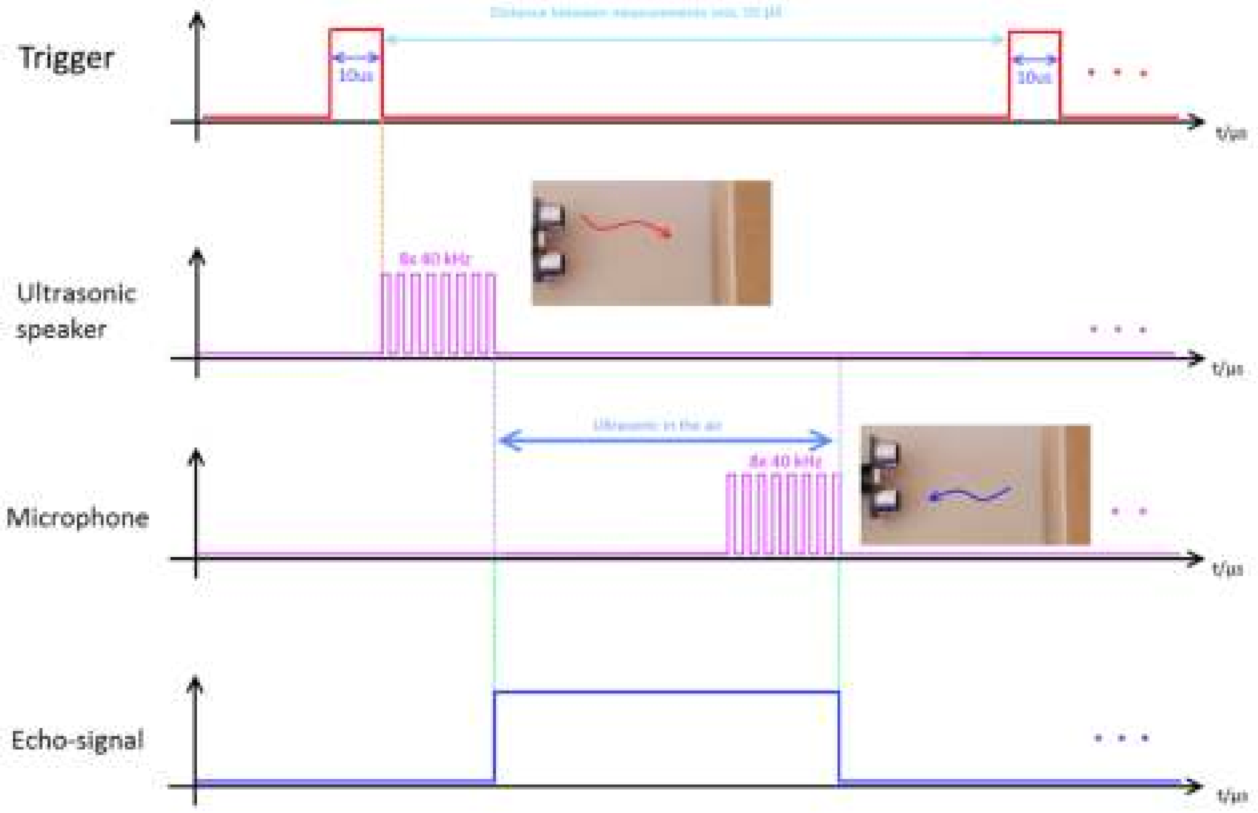
This modules shows how to measure a distance to an object with an ultrasonicspeaker and a microphone. The principle is based on the theory that the speed of the sonic, at a stable temperature, is constant - at 20°C it's 343,2m/s.

With this fact you can make the distance measurement to a time measurement, which can be easiely taken from microcontrollers.



The Ultrasonicspeaker sends 8x 40KHz signals, which will be reflected by objects and recorded with the microphone, in this sensor module. Ultrasonic is being used because you are not able to hear it with the human sense of hearing (Human sense of hearing is nearly 20HZ - 22.000Hz).

The ultrasonic signal output will start after the "trigger input pin" gets a start signal with the lenght of 10µs (active high). Activating the signal (active high) at the "echo output signal pin" after outputting. If you now record the reflected signal with the microphone, the echo signal will deactivated after the detection of the echo-signal. You can measure the time between activating and deactivating of the echo signal and calculate the distance with it.



Code example Arduino

The example program activates the distance measurement and measures the time of the ultrasonic signal. This time will be taken as a base to calculate the distance - after that the result will be printed to the serial output. If the signal is not in the measurement range, an error message will be printed..

```
#define Echo_EingangsPin 7 // Echo input-pin
#define Trigger_AusgangsPin 8 // Trigger output-pin

// Needed variables will be defined and initialized
int maximumRange = 300;
int minimumRange = 2;
long Abstand;
long Dauer;

void setup() {
 pinMode(Trigger_AusgangsPin, OUTPUT);
 pinMode(Echo_EingangsPin, INPUT);
 Serial.begin (9600);
}

void loop() {

 // Distance measurement will be started with a 10us long trigger signal
 digitalWrite(Trigger_AusgangsPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(Trigger_AusgangsPin, LOW);

 // Now it will be waited at the echo input till the signal was activated
 // and after that the time will be measured how long it is active
```

```
  Dauer = pulseIn(Echo_EingangsPin, HIGH);

  // Now the distance will be calculated with the recorded time
  Abstand = Dauer/58.2;

  // Check if the measured value is in the permitted range
  if (Abstand >= maximumRange || Abstand <= minimumRange)
  {
     // An error message will be shown if it's not
       Serial.println("Distance is not in the permitted range");
       Serial.println("-----------------------------------");
  }

  else
  {
     // The calculated distance will be shown at the serial output
       Serial.print("The distance is: ");
       Serial.print(Abstand);
       Serial.println("cm");
       Serial.println("-----------------------------------");
  }
   // Break between single measurements
  delay(500);
 }
```

Connections Arduino:

VCC	=	[Pin 5V]
Echo	=	[Pin 7]
Trigger	=	[Pin 8]
Sensor GND	=	[Pin GND]

Example program download

KY-050_ultrasonic-distance

Code example Raspberry Pi

!! Attention !! 5V Voltagelevel !! Attention !!

The Raspberry Pi works with its ARM-CPU-core with a voltage level of 3,3V instead of 5V, like the Arduino. This sensor needs a higher voltage level to work. If you use this sensor without restriction you may damage the input pins of your Raspberry Pi permanently. But this sensor-kit also comes with a voltage-translater (KY-051) which reduces the voltage level and saves your Raspberry Pi from permantent damage. It needs to be connected between the sensor and the Raspberry Pi.

You can find more informations to that here KY-051 Voltage Translator / Level Shifter

!! Attention !! 5V Voltagelevel !! Attention !!

The example program activates the distance measurement and measures the time of the ultrasonic signal. This time will be taken as a base to calculate the distance. The result will be given via the serial output. If the signal is not in the measurement range, an error message will be printed to the serial output.

```
# coding=utf-8
# Needed modules will be imported and configured
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

# You can pick the input and output pins here
Trigger_AusgangsPin = 17
Echo_EingangsPin    = 27

# You can set the delay (in seconds) between the single measurements here
sleeptime = 0.8

# Here, the input and output pins will be configured
GPIO.setup(Trigger_AusgangsPin, GPIO.OUT)
GPIO.setup(Echo_EingangsPin, GPIO.IN)
GPIO.output(Trigger_AusgangsPin, False)

# Main program loop
try:
    while True:
        # Distance measurement will be started with a 10us long trigger signal
        GPIO.output(Trigger_AusgangsPin, True)
        time.sleep(0.00001)
        GPIO.output(Trigger_AusgangsPin, False)

        # The stop watch will start here
        EinschaltZeit = time.time()
        while GPIO.input(Echo_EingangsPin) == 0:
            EinschaltZeit = time.time()

        while GPIO.input(Echo_EingangsPin) == 1:
            AusschaltZeit = time.time()

        # The difference between the times gives the searched duration
        Dauer = AusschaltZeit - EinschaltZeit
        # With it you can calculate the distance
        Abstand = (Dauer * 34300) / 2

        # Here you check if the measured value is in the permitted range
        if Abstand < 2 or (round(Abstand) > 300):
            # If not an error message will be shown
            print("Distance is not in the permitted range")
            print("------------------------------")
        else:
            # The value of the distance will be reduced to 2 numbers behind the comma
            Abstand = format((Dauer * 34300) / 2, '.2f')
            # The calculated distance will be shown at the terminal
            print("The distance is:"), Abstand,("cm")
            print("------------------------------")

        # Break between the single measurements
        time.sleep(sleeptime)

# Scavenging work after the end of the program
except KeyboardInterrupt:
    GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor KY-050:

VCC	=	5V	[Pin 2 (RPi)]

Trigger	=	Pin B1	[KY-051-Voltage Translator]
Echo	=	Pin B2	[KY-051-Voltage Translator]
GND	=	GND	[Pin 6 (RPi)]

KY-051- Voltage Translator:

VCCb	=	5V	[Pin 04(RPi)]
Pin B1	=	Trigger	[KY-050-UltrasonicSensor]
Pin B2	=	Echo	[KY-050-UltrasonicSensor]
VCCa	=	3,3V	[Pin 01(RPi)]
Pin A1	=	GPIO17	[Pin 11(RPi)]
Pin A2	=	GPIO27	[Pin 13(RPi)]
GND	=	GND	[Pin 06(RPi)]

- You must not connect the other pins of the KY-051-Voltage-Translator-Modul (OE,B3,B4,A3,A4).

Example program downlaod

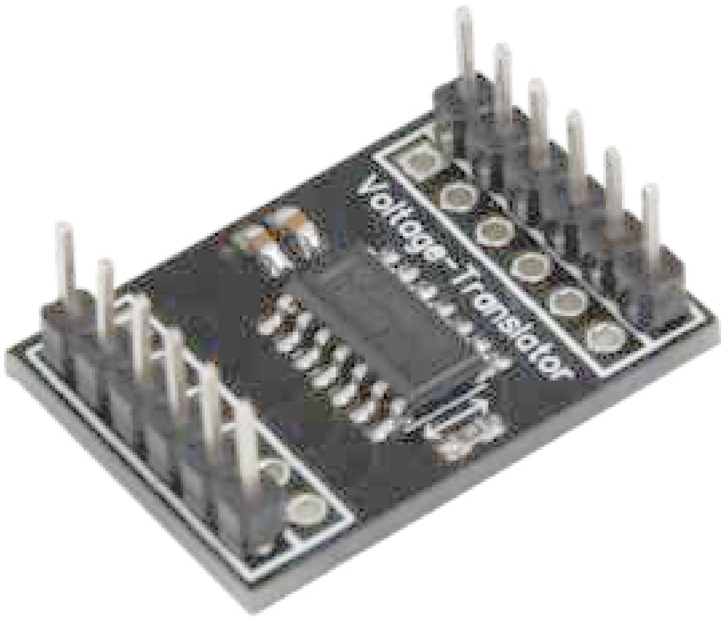
KY-50_ultrasonic-distance_RPi

To start, enter the command:

```
sudo python KY-50_ultrasonic-distance_RPi.py
```

KY-051 Voltage Translator / Level Shifter

Picture



Technical data / Short description

This level-shifter modulates the voltage of a digital signal into a higher or lower voltage. 4 channels are available which can be modulated.

There are many microcontroller-systems which are operating in different voltage ranges: older systems, like Arduino, are using 5V for their in/output pins and new systems, like Raspberry Pis, are using 3.3V. Back then, microcontrollers were needing higher voltages to communicate. Nowadays since the most problems with signal noise/ signal disruptions has been cleared, the voltage range is beeing kept as low as possible to decrease the heat and decrease the energy consumption. Systems with 1.8V are not unusual these days.

But to communicate between two different systems, like in our example (Arduino ----> Raspberry Pi), you have to shift the voltage range - if you don't, the systems with the lower voltage range will create more heat and could take constant damage.

Pinout

The Pinout is printed on the module board.

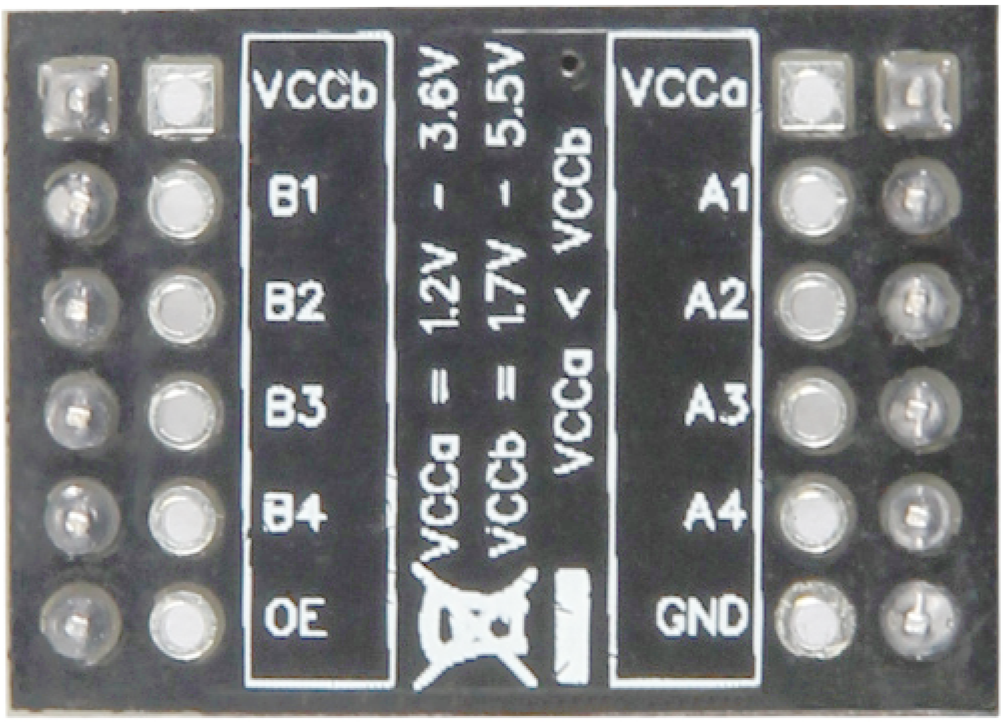
The signal inputs/outputs A1-A4 also B1-B4 will be shifted to the needed voltage range (VCCa -> A1-A4 | VCCb -> B1-B4)

Example:

Arduino OUTPUT -> Digital [ON] = 5V @ B1 >>>>>>> 3.3V @ B2 -> Raspberry Pi INPUT

<u>No additional software or code is needed that it works, The module works autonomously.</u>

Please pay attention that VCCb has to be higher or equal to VCCa (Example: VCCb=5V - VCCa=3, 3V)



Example connection between Arduino and Raspberry Pi:

Connections Arduino:

VCCb	=	[Pin 5V]
B1	=	[Pin 03]
B2	=	[Pin 04]
B3	=	[Pin 05]
B4	=	[Pin 06]
GND	=	[Pin GND]

Connections Raspberry Pi:

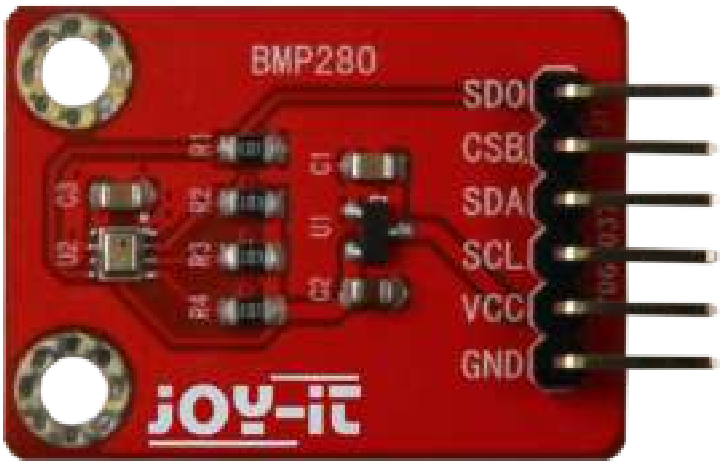
VCCa	=	3,3V	[Pin 01]
A1	=	GPIO18	[Pin 12]
A2	=	GPIO03 / SCL	[Pin 05]
A3	=	GPIO02 / SDA	[Pin 01]
A4	=	GPIO14	[Pin 08]
GND	=	GND	[Pin 09]

Please take attention that both systems are connected with the same GND - you don't need to use OE with this module

KY-052 Pressure-sensor / Temperature-sensor (BMP280)

Contents

Picture

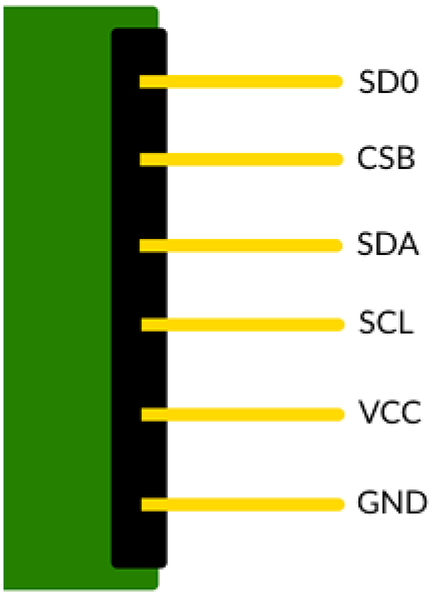


Technical data / Short description

This pressure sensor measures the air pressure at the sensor output and sends the result encoded to the I2C-bus.

You will need additional software for this sensor

Pinout



- *You can connect this sensor to 5V systems and to 3,3V systems. Please pay attention that only one of the power supply pins is connected to the desired voltage - you can get additional informations at the example below: Connection Arduino (5V) and Connection Raspberry-Pi (3,3V).

Software example Arduino

An additional Arduino library is needed to use this sensor. This library, for the BMP280, was written by Kevin Townsend (KTOWN) and was produced for Adafruit. It is available here: Adafruit BMP280 Library

The example below is using this library - we advise to download the library from Github, decompress it and copy it into the Arduino-library folder which can be found under the path (C:\user\[username]\documents\arduino\libraries).

```
//Initialise Libraries
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
//Define PIN-Wiring
#define BMP_SCK 13
#define BMP_MISO 12
#define BMP_MOSI 11
#define BMP_CS 10

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

void setup() {
  Serial.begin(9600);
  if (!bmp.begin()) {
    Serial.println(F("Could not find a valid BMP280 sensor, check wiring!"));
    while (1);
  }
}

void loop() {
    Serial.print(F("Temperature = "));
    Serial.print(bmp.readTemperature());
    Serial.println(" *C");

    Serial.print(F("Pressure = "));
    Serial.print(bmp.readPressure());
    Serial.println(" Pa");

    Serial.println();
    delay(2000);
}
```

Example program download:

KY-052_BMP280_ARD

Connections Arduino:

VCC	=	[Pin 5V]
GND	=	[Pin GND]
SCL	=	[Pin 13]
SDA	=	[Pin 11]
SD0	=	[Pin 12]
CSB	=	[Pin 10]

Software example Raspberry Pi

To run our example code, you need to activate some modules and install an additional library first.

The library was written by Adafruit and udpated for BMP280 support by "bastienwirtz". This library is published under the MIT license and is available here: BMP280 Libary for Raspberry Pi

It has to be installed first. But at the very beginning, you need to activate I2C in your settings.

You can do this by entering this in your command line:

```
sudo raspi-config
```

Simply navigate to no. 5 (Interfacing Options) and enable I2C. Restart your Raspberry after doing this.

Second, you have to install the github software on your Raspberry-Pi

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

For this you have to make sure that the Raspberry Pi has a connection to the internet. You can downlaod and decompress the latest version of the Adafruit_BMP280 library with the command...

```
git clone https://github.com/bastienwirtz/Adafruit_Python_BMP
```

After that we can jump into the downloaded folder with the command...

```
cd Adafruit_Python_BMP/
```

... and install the library with ....

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

Now you are ready to use the library.

In order that the Raspberry Pi and the sensor can communicate via I2C-bus, you have to start the I2C-function of the Raspberry Pi first.

To do that you have to add the following line to the end of the file "/boot/config.txt:

```
    dtparam=i2c_arm=on
```

You can edit the file with the following command:

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

You can save and close the file with the key sequence [ctrl + x -> y -> enter].

Also you need additional libraries if you want to use I2C in with python. To install it, you have to enter the following command to the console:

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

After that you can use the following python code example. The program starts the measurement for air pressure, temperature and the altitude above sea level.

```
import Adafruit_BMP.BMP280 as BMP280

sensor = BMP280.BMP280(address=0x76)

print 'Temp = {0:0.2f} *C'.format(sensor.read_temperature())
print 'Pressure = {0:0.2f} Pa'.format(sensor.read_pressure())
print 'Altitude = {0:0.2f} m'.format(sensor.read_altitude())
print 'Sealevel Pressure = {0:0.2f} Pa'.format(sensor.read_sealevel_pressure())
```

Connections Raspberry Pi:

VCC	= -	[N.C]
GND	= GND	[Pin 06]
SCL	= GPIO03 / SCL	[Pin 05]

SDA	=	GPIO02 / SDA	[Pin 03]
3.3	=	3,3V	[Pin 01]

Example program downlaod

KY-052_BMP280_RPi

To start, enter the command:

```
sudo python KY-052_BMP280_RPi.py
```

If your example is not working, you might need to change the I2C-Address in the example-code.

In order to do this, after wiring the sensor, check the I2C-Address by entering:

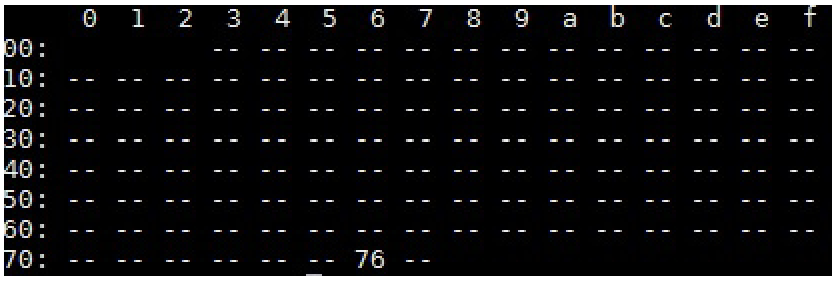
```
sudo i2cdetect -y 1
```

for newer Raspberry Pis or

```
sudo i2cdetect -y 1
```

for older Raspberry Pis.

The result should look like this:



```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- 76 --
```

The displayed number (in this case: 76) is your I2C-Address. The I2C-Address 76 is configured as default in our example. If your address is different, enter the example code by entering

```
sudo nano KY-052_BMP280_RPi.py
```

and update the following line to your address.

KY-053 Analog digital converter

Contents

Picture

Technical data / Short description

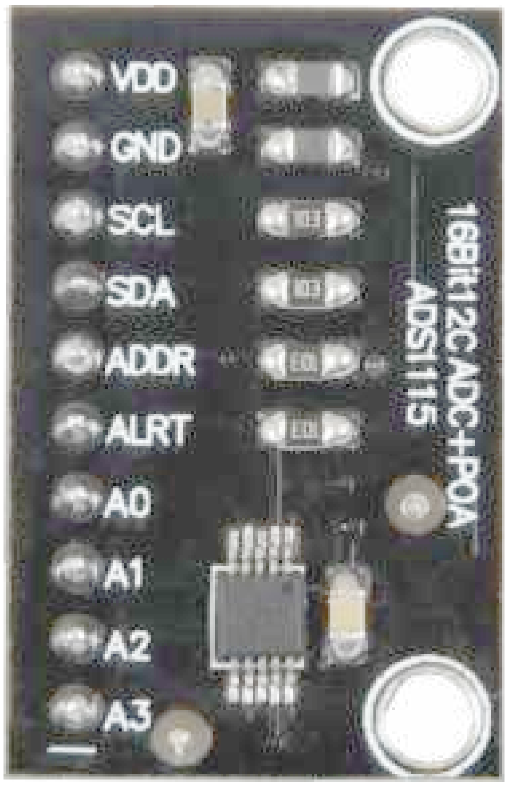
With the right command on the I2C-bus, you are able to measure voltage values with a precision of 16-bit at 4 different input pins.

The result of the measurement will be send encoded via I2C-bus.

For this mode you will need a special software.

Pinout

The Pin connections are printed on the module board



Code example Arduino

The Arduino board includes a 10-bit-ADC with 6 channels by default. If you need more channels or a higher precision you can use the KY053 analog digital converter module to expand the channel by 4 with a precision of 12-bit which will connected with the Arduino via I2C.

You have a few options to control the module - the best choice is to use the ADS1X15 libraries from the company Adafruit which you can get here: [https://github.com/adafruit/Adafruit_ADS1X15] which were published under the open source license: [BSD-Lizenz]

The example below uses these libraries - we advise to download the libraries from Github, to unzip it and to copy it into the Arduino-library-folder, which you can find here : (C:\user\[username]\documents\Arduino\libraries) by default. To use it in the following code example and for following projects, you have to download and copy it. Alternatively, you can get it with the download package below.

```
#include <Adafruit_ADS1015.h>

// ADS1115 module will be initialised
Adafruit_ADS1115 ads;

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

  Serial.println("Analog input values of the ADS1115 will be read and outputted");
  Serial.println("ADC Range: +/- 6.144V (1 bit = 0.1875mV)");
  Serial.println("ADC Range: +/- 6.144V (1 bit = 0.1875mV)");

  // The module includes signal amplifier at the inputs
  //                                                                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;

  // The command "ads.readADC_SignalEnded(0)" is the operation, which starts the mesurement
  adc0 = ads.readADC_SingleEnded(0);
  adc1 = ads.readADC_SingleEnded(1);
  adc2 = ads.readADC_SingleEnded(2);
  adc3 = ads.readADC_SingleEnded(3);

  // You need this value to calculate the voltage - it depends on the configured amplification
  gain_conversion_factor= 0.1875;

  // Calculating of the voltage values from the measured values.
  voltage0 = (adc0 * gain_conversion_factor);
  voltage1 = (adc1 * gain_conversion_factor);
  voltage2 = (adc2 * gain_conversion_factor);
  voltage3 = (adc3 * gain_conversion_factor);

  // The values will be outputted to the serial interface
  Serial.print("Analog input 0: "); Serial.print(voltage0);Serial.println("mV");
  Serial.print("Analog input 1: "); Serial.print(voltage1);Serial.println("mV");
  Serial.print("Analog input 2: "); Serial.print(voltage2);Serial.println("mV");
  Serial.print("Analog input 3: "); Serial.print(voltage3);Serial.println("mV");
  Serial.println("------------------------");

  delay(1000);
}
```

Example program download:

KY-053_analog-digital-converter_ARD

Connections Arduino:

VDD	= [Pin 5V]
GND	= [Pin GND]
SCL	= [Pin SCL]
SDA	= [Pin SDA]
ADDR	= [N.C.]
ALRT	= [N.C.]
A0	= [peak Analog 0]
A1	= [peak Analog 1]
A2	= [peak Analog 2]

A3	= [peak Analog 3]

Code example Raspberry Pi

The Raspberry Pi has no ADC (Analog Digital Converter) included on its chip. It is a disadvantage if you want to use sensors which doesn't send digital signals like : Voltage range exceeded -> digital ON | Voltage range deceeded -> digital OFF | Example: Button pushed [ON], Button released [OFF], instead of that you get a continuously changing value (example: potentiometer -> different position = different voltage value).

To evade this problem , our sensorkit X40 comes with a 16 bit precise ADC (KY-053), which can be used with the Raspberry Pi, to expand it for 4 analog Input pins. It is connected with the Raspberry Pi via I2C, takes the analog measurement and sends a digital signal to the Raspberry Pi.

The program uses the ADS1x15 and I2C-python-libraries from the company Adafruit to control the ADC. You can get it here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] it is published under the open source license MIT OpenSource-Lizenz. You can find the needed libraries in the download package below.

The program reads the current voltage value which is at the 4 channels of the ADS1115, and shows it at the terminal. You can configure the break between the measurements with the variable "delayTime".

In order that the Raspberry Pi can communicate with the sensor via I2C-bus , you have to activate the I2C funktion from the Raspberry Pi first. To do this you have to add the following line at the and of the file "/boot /config.txt" :

```
dtparam=i2c_arm=on
```

You can edit the file with the command:

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

You can save and close the file, after adding of the line, with the key sequence : [Ctrl + X -> Y -> enter]. You will need additional libraries to use I2C in python. To install them you have to use the following command at the console:

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

After that you can use the following code example:

```
```

Connections Raspberry Pi:

VDD	= 3,3V	[Pin 01]
GND	= GND	[Pin 06]
SCL	= GPIO03 / SCL	[Pin 05]
SDA	= GPIO02 / SDA	[Pin 03]
ADDR	= N.C.	[-]

ALRT	= N.C.	[-]
A0	= Measurement peak Analog 0	[Voltage which is to measure | for example sensor output]
A1	= Measurement peak Analog 1	[Voltage which is to measure | for example sensor output]
A2	= Measurement peak Analog 2	[Voltage which is to measure | for example sensor output]
A3	= Measurement peak Analog 3	[Voltage which is to measure | for example sensor output]

Example program download

KY-053_RPi_AnalogDigitalConverter

To start, enter the command:

```
sudo python KY-053_RPi_AnalogDigitalConverter.py
```

Extended function of the ADS1115 ADC

The function of the ADS1115 which is used above is called "Single Ended Conversion" and says that a measurement from a single picked channel will go against GND.

Additional to this kind of measurement, the ADS1115 ADC provides another measurement function, which can measure difference-voltages between two input pins (Example: Voltage between A0 and A1). Additional to the single-ended measurement, you can activate the Comparator function, which only gives you the results if an extreme value of the voltage was exceeded.

These functions how the changing of the sample rate for example, are included in the Adafruit libraries - for more information look into the documentation of the Adafruit libraries.