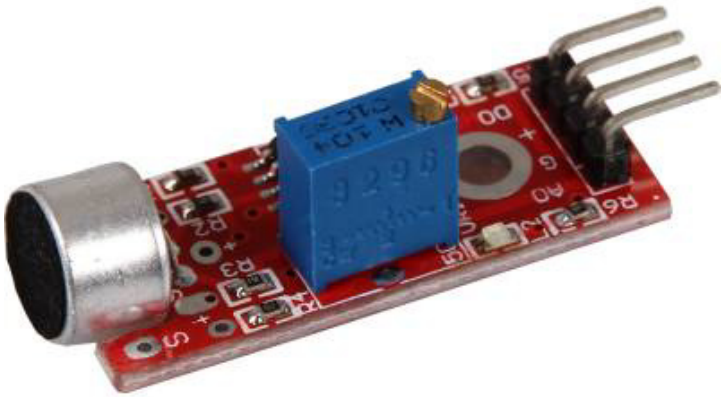
KY-037 Microphone sensor module (high sensitivity)

Contents

Picture



Technical data / Short description

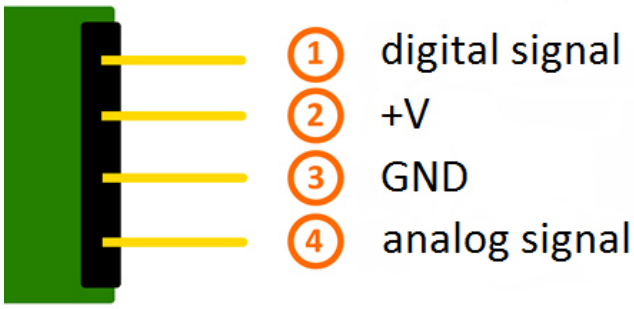
Digital Out: You can use a potentiometer to configure an extreme value for the sonic. If the value exceeds the extreme value, it will send a signal via digital out.

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

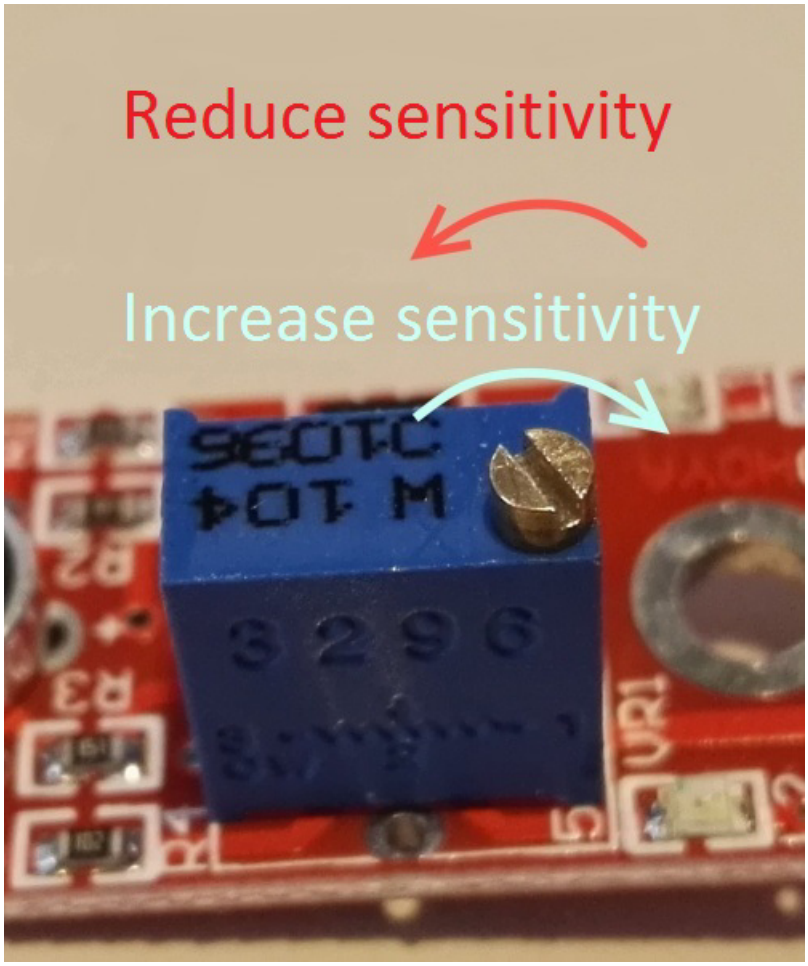
Pinout



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
```