Welcome!

Thank you for purchasing our *AZ-Delivery KY-037 Big Microphone Sensor Module.* On the following pages, you will be introduced to how to use and set up this handy device.

Have fun!

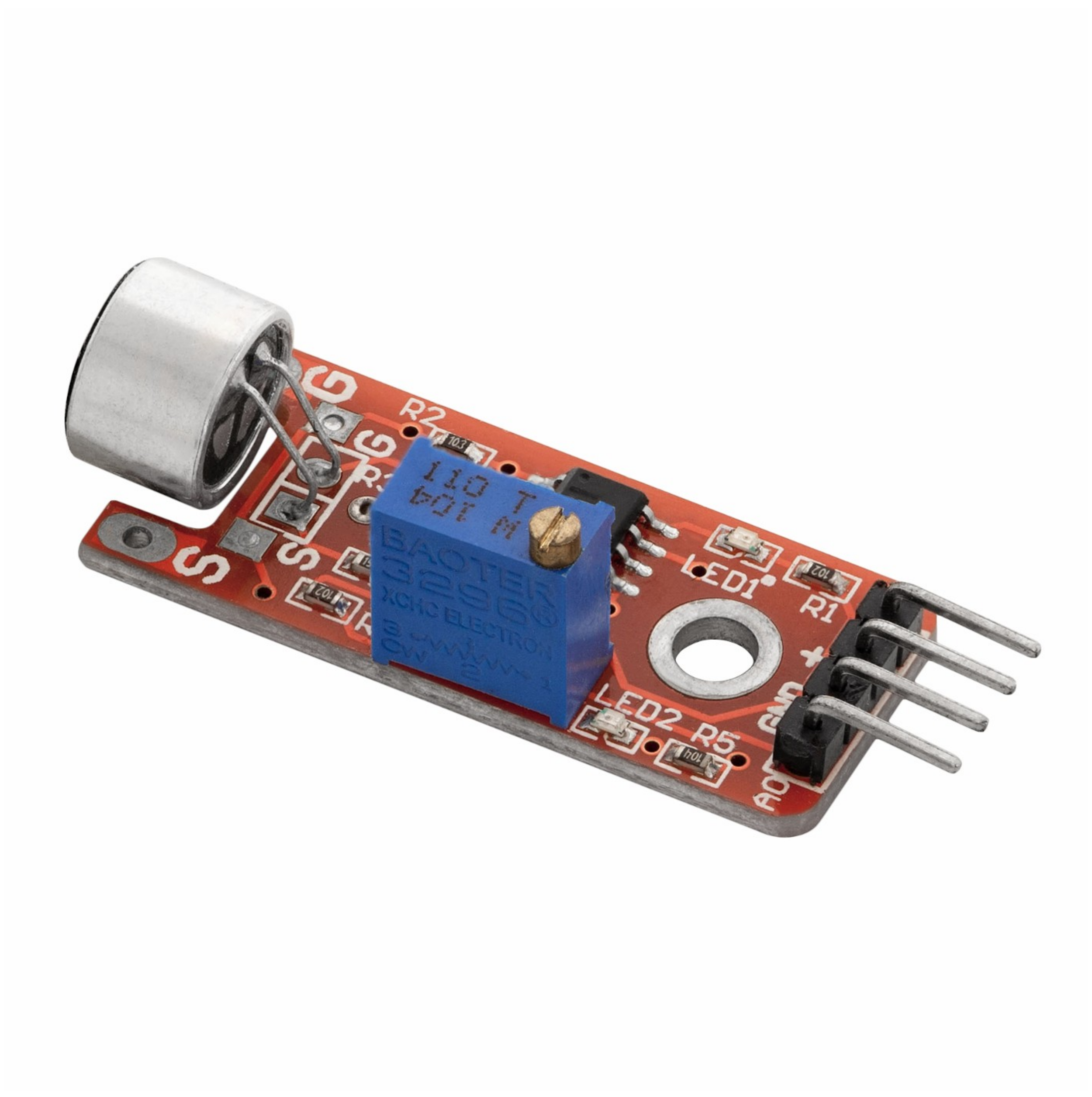


Table of Contents

Introduction

The KY-037 big microphone module has the capacitive electret microphone, a few resistors, two LEDs, a potentiometer and an LM393 analog voltage comparator.

The capacitive electret microphone is a sound sensor - a component that converts sound waves into electrical signals. The sensor detects the sound wave intensity in the environment. This kind of microphone has a thin film (like thin foil) of the electret (dielectric) material which vibrates on sound waves. The capacitance of the thin film changes with vibrations, which cause the corresponding analog voltage change. Since the analog voltage change is weak, it has to be amplified. The LM393 and a few resistors are in charge of the amplification.

Specifications

»	Operating voltage range:	from 3.3V to 5V DC
»	Output:	digital and analog
»	Frequency range:	from 100Hz to 10.000Hz
»	Sensitivity:	-46 ± 2.0 (0 dB = 1V/Pa) at 1kHz
»	Sensitivity to Noise Ratio:	58dB
»	Dimensions:	15 x 36 x 13mm [0.6 x 1.4 x 0.51in]

There are two LEDs on-board the module, one for indicating power supply and one for indicating the intensity of the sound.

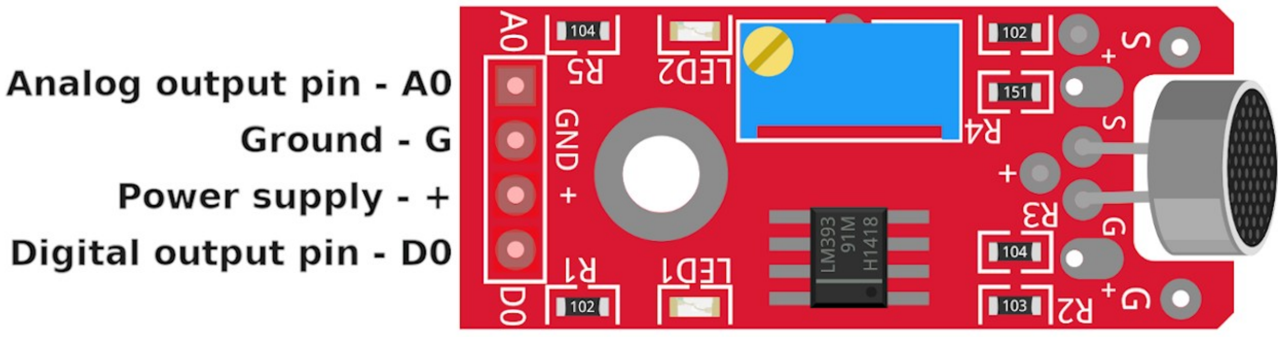
The analog output pin voltage sensitivity can be adjusted via an on-board blue potentiometer. With some adjustment of the potentiometer, when the analog voltage is read via analog to digital converter (ADC for short), the digital value of the voltage is around the value *595*, and that is when there is not sound near the KY-037 module.

When there is a sound near the sensor, the ADC value changes dramatically around the value *595*, rise and drop a few times. This happens because of the nature of sound waves. This can be seen on the image of the Serial Plotter (later in the text).

The state when the digital output pin outputs the *HIGH* state (is turned *ON*) is the state when the ADC value is grater than the value of *700.*

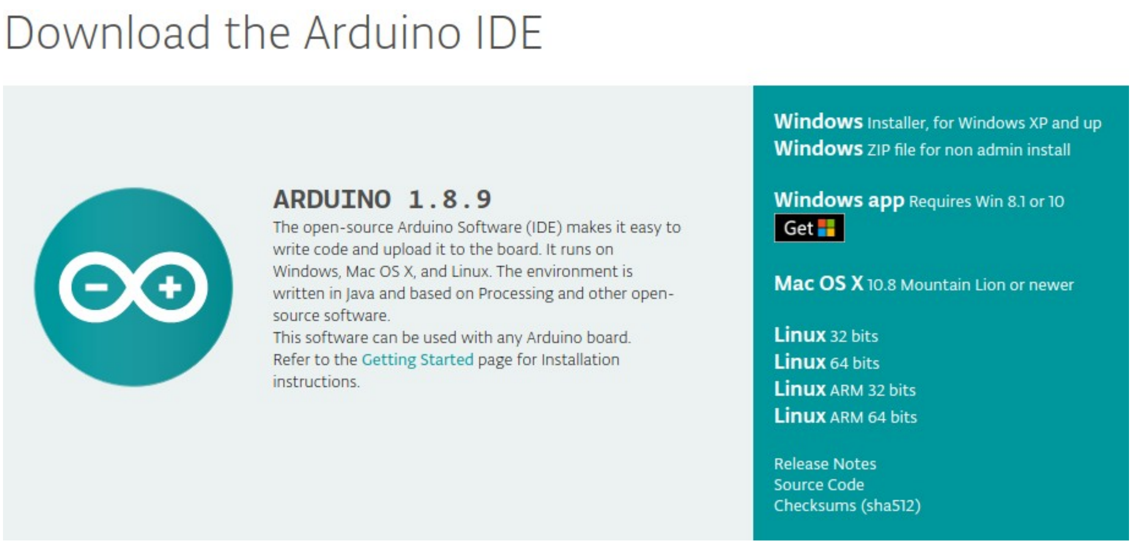
The pinout

The KY-037 big microphone module has four pins. The pinout diagram is shown on the following image:



How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the *link* and download the installation file for the operating system of choice.



Download the Arduino IDE

ARDUINO 1.8.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

For `Windows` users, double click on the downloaded `.exe` file and follow the instructions in the installation window.

For *Linux* users, download a file with the extension *.tar.xz*, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two *.sh* scripts have to be executed, the first called *arduino-linux-setup.sh* and the second called *install.sh*.

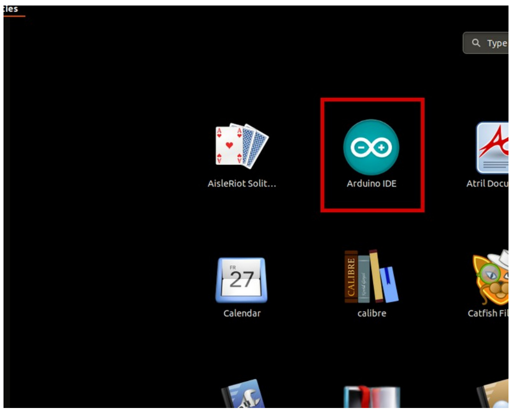
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

sh arduino-linux-setup.sh user_name

user_name - is the name of a superuser in the Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

The second script called *install.sh* script has to be used after installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.

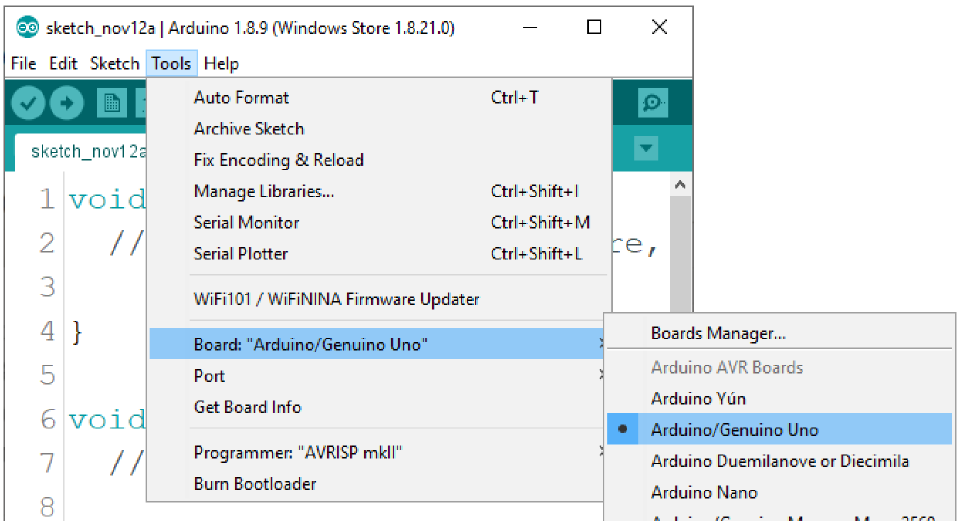


Almost all operating systems come with a text editor preinstalled (for example, *`Windows`* comes with *`Notepad`*, *`Linux Ubuntu`* comes with *`Gedit`*, *`Linux Raspbian`* comes with *`Leafpad`*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check if your PC can detect an Arduino board. Open freshly installed Arduino IDE, and go to:

`Tools > Board > {your board name here}`

`{your board name here}` should be the *`Arduino/Genuino Uno`*, as it can be seen on the following image:

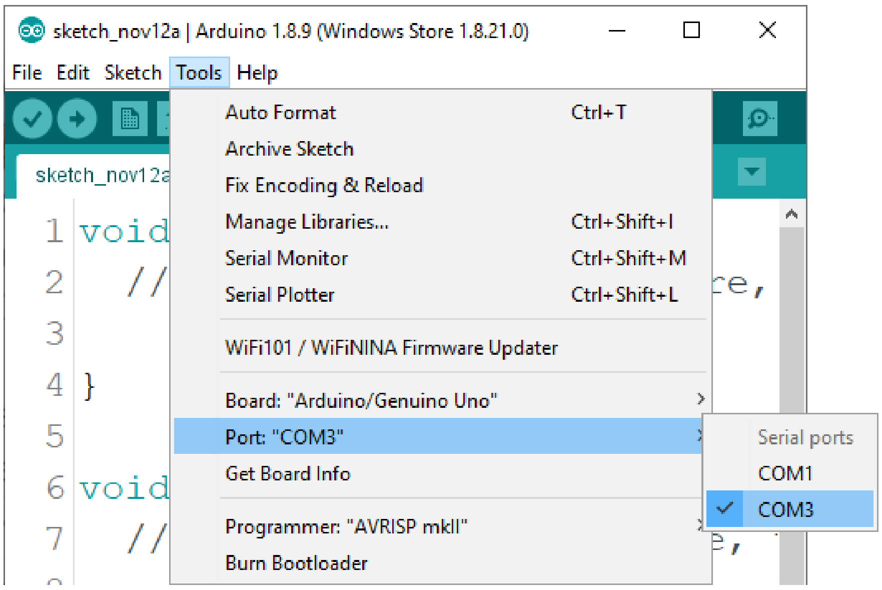

sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
sketch_nov12a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
● Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano

The port to which the Arduino board is connected has to be selected. Go to:

`Tools > Port > {port name goes here}`

and when the Arduino board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

If the Arduino IDE is used on Windows, port names are as follows:


sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
✓ COM3

For *Linux* users, for example port name is */dev/ttyUSBx*, where *x* represents integer number between *0* and *9*.

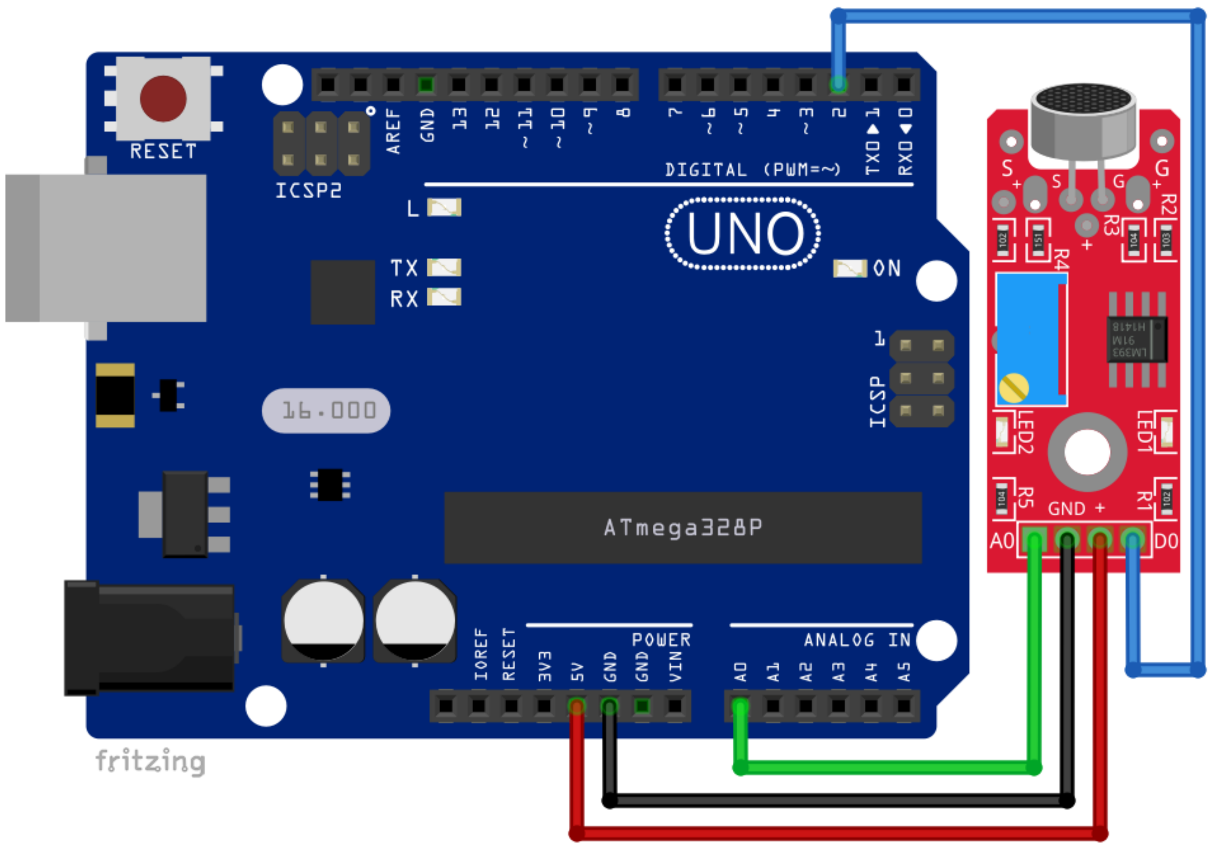
How to set-up the Raspberry Pi and Python

For the Raspberry Pi, first the operating system has to be installed, then everything has to be set-up so that it can be used in the *Headless* mode. The *Headless* mode enables remote connection to the Raspberry Pi, without the need for a *PC* screen Monitor, mouse or keyboard. The only things that are used in this mode are the Raspberry Pi itself, power supply and internet connection. All of this is explained minutely in the free eBook:
Raspberry Pi Quick Startup Guide

The *Raspbian* operating system comes with *Python* preinstalled.

Connecting the module with Uno

Connect the KY-037 module with the Uno as shown on the following connection diagram:



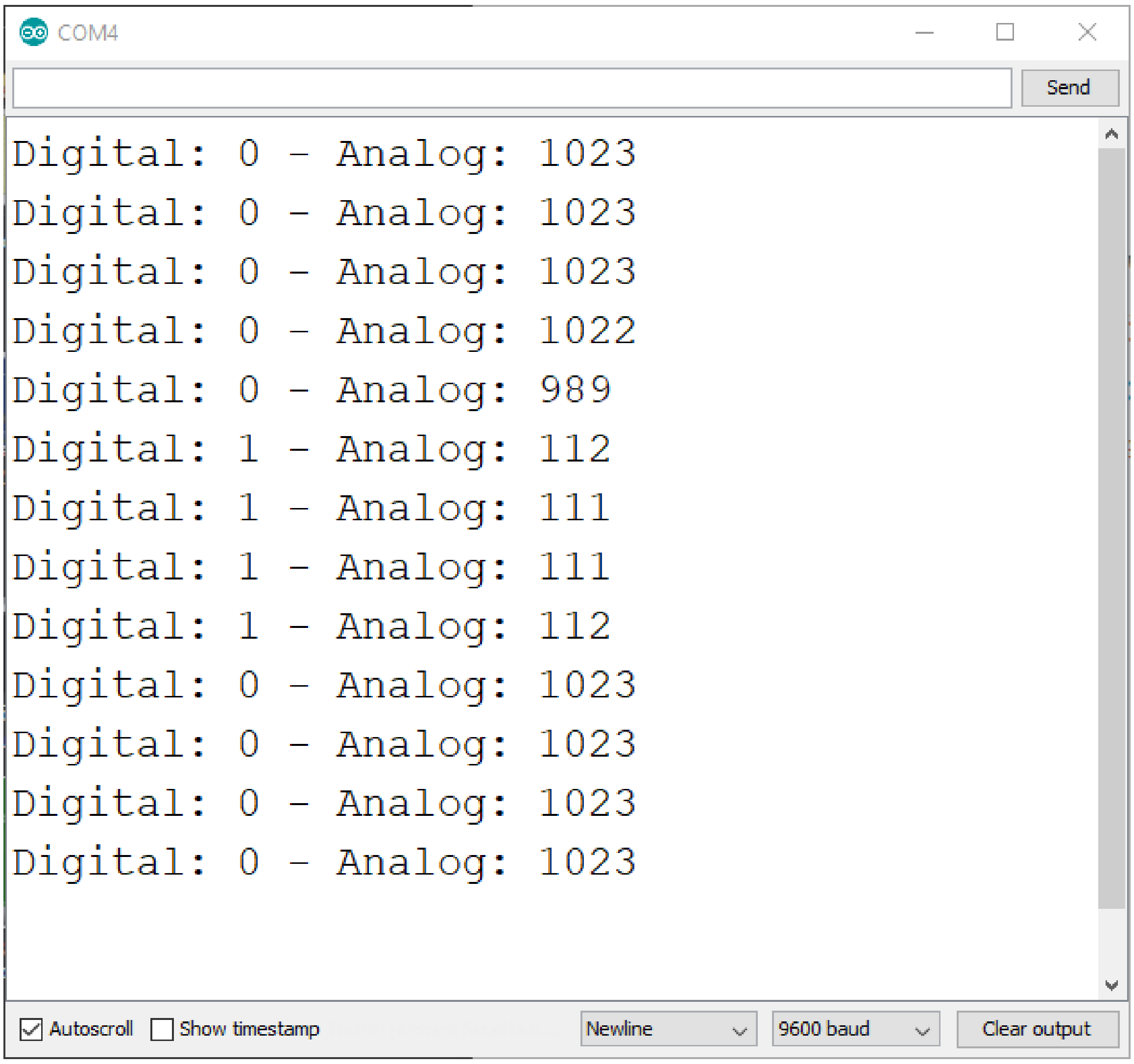
KY-037 pin	**>**	**Uno pin**	
+	>	5V	**Red wire**
GND	>	GND	**Black wire**
A0	>	A0	**Green wire**
D0	>	D2	**Blue wire**

With the set-up both the analog and digital pins of the KY-037 module are used.

Sketch example

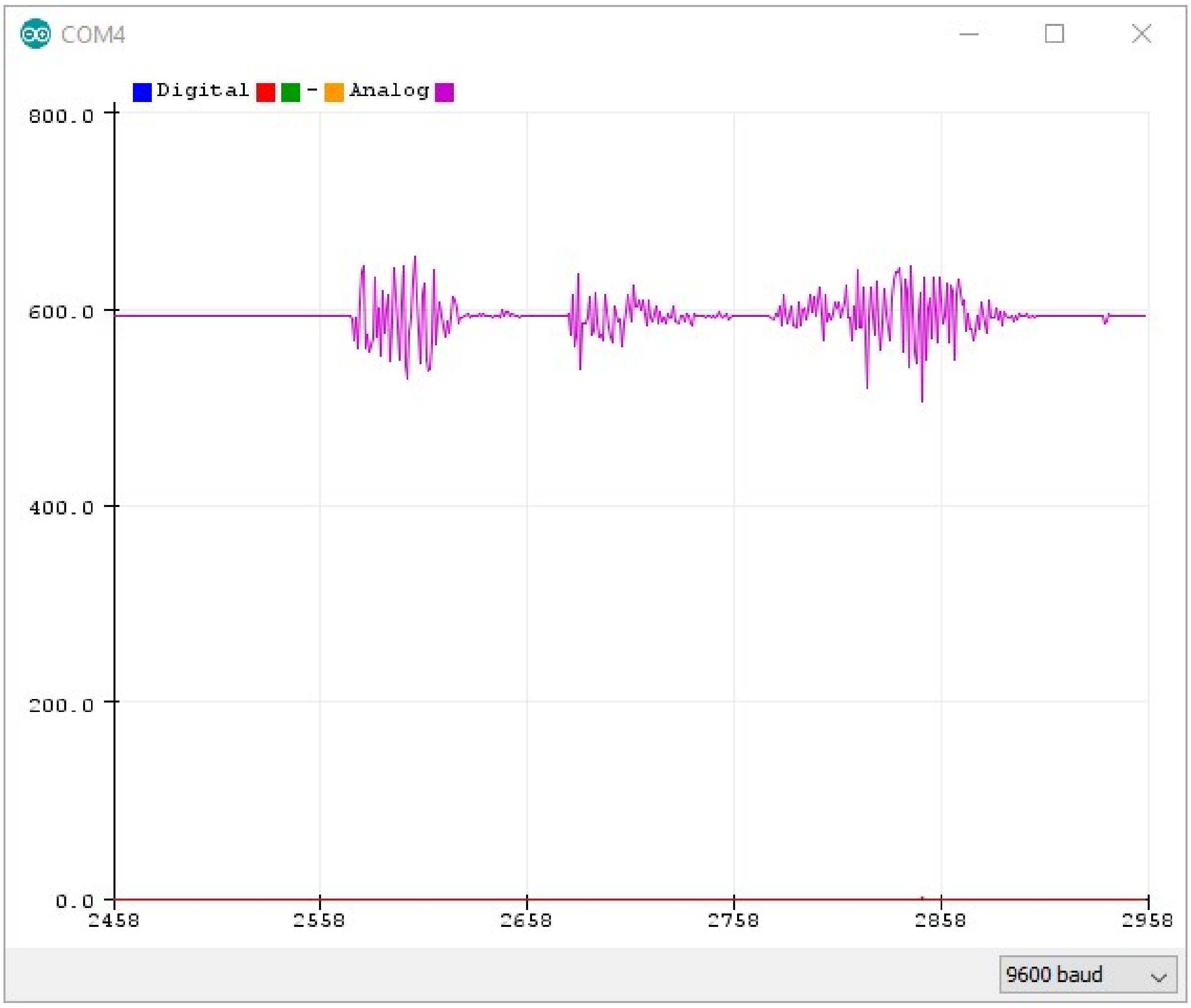
```
#define DIGITAL_PIN 2
#define ANALOG_PIN 0

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

void loop() {
  Serial.print("Digital: ");
  Serial.print(digitalRead(DIGITAL_PIN));
  Serial.print(" - Analog: ");
  Serial.println(analogRead(ANALOG_PIN));

  delay(1000);
}
```

Upload the sketch to the Uno and open Serial Monitor (*`Tools > Serial Monitor`*). The result should look like the output on the following image:

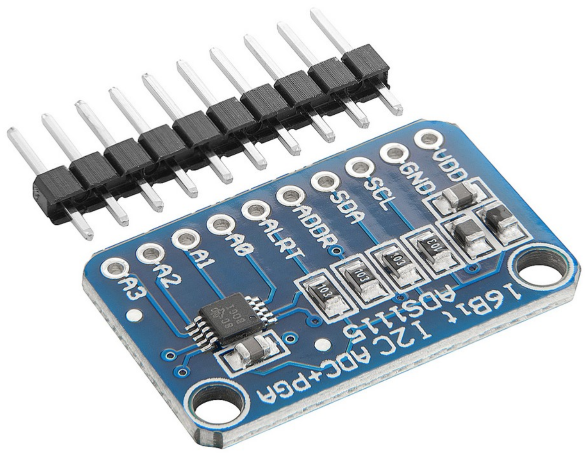


```
Digital: 0 - Analog: 1023
Digital: 0 - Analog: 1023
Digital: 0 - Analog: 1023
Digital: 0 - Analog: 1022
Digital: 0 - Analog: 989
Digital: 1 - Analog: 112
Digital: 1 - Analog: 111
Digital: 1 - Analog: 111
Digital: 1 - Analog: 112
Digital: 0 - Analog: 1023
Digital: 0 - Analog: 1023
Digital: 0 - Analog: 1023
Digital: 0 - Analog: 1023
```

The sound waves can be seen on the Serial Plotter (*`Tools > Serial Plotter`*). The result should look like the output on the following image:



External analog to digital module



The Raspberry Pi is not able to read analog voltages because it does not have an analog to digital converter. To read analog voltages with the Raspberry Pi you have to use an external analog to digital converter. The AZ-Delivery offers this kind of device, it is called *ADS1115 Analog to digital converter*.

The *ADS1115* module has *16* bit digital precision, and uses *I2C* interface to send data to microcontroller. The best thing about it is that its operating voltage ranges from *3.3V* to *5V DC*, which means that the module can be used with the Raspberry Pi.

If you want to know more about the device you can read about it in the free eBook:
ADS1115 Analog to digital converter Quick Starter Guide

The module can read both positive and negative voltages. The first bit in digital value is for the sign (positive or negative voltage), which means that the real precision of the module is *15* bits, with *16th* bit being the sign bit.

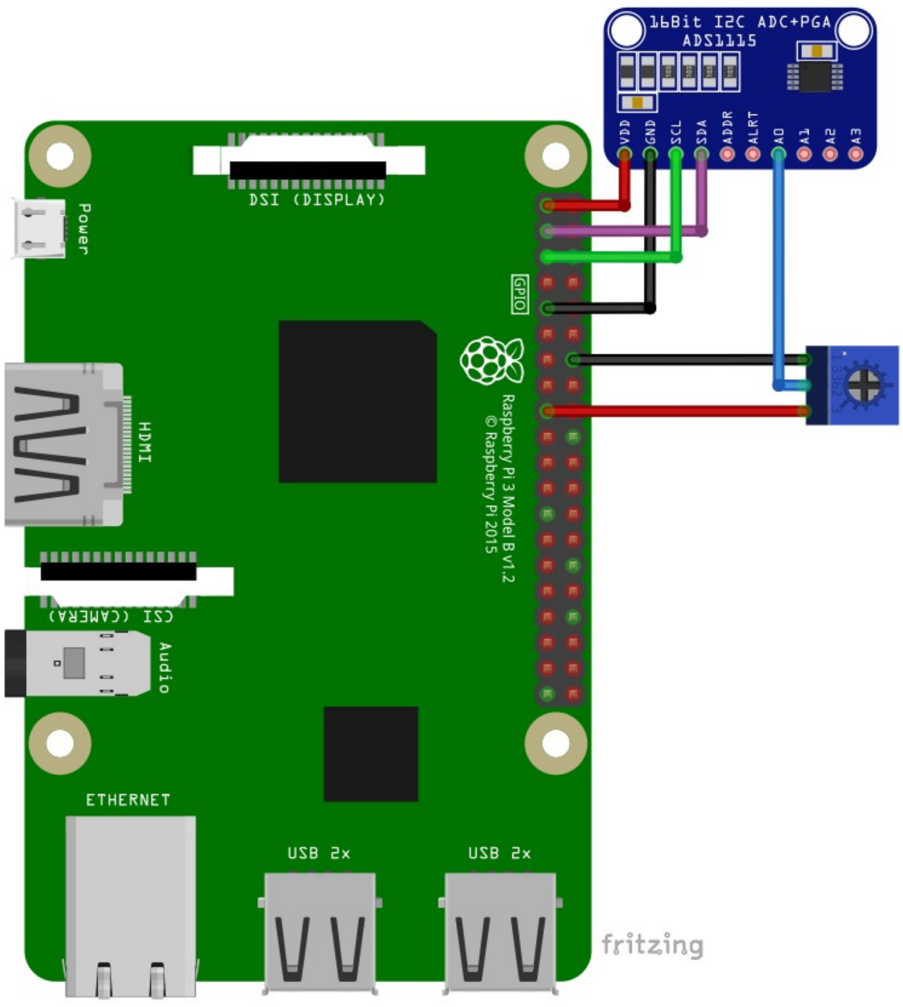
Also, the module has four analog input pins, and four different *I2C* addresses. In this eBook, the default *I2C* address (*ADDR* pin not connected to anything) is used, and in the next script example the analog input pin *0* is used. Any of the on-board analog pins (from *0* to *3*) can be used for this purpose.

For example, the ADC (16 bits) in the ADS1115 module is much precise than the ADC (10 bits) in the Uno.

Connecting the ADS1115 module with Raspberry Pi

Because the Raspberry Pi does not have the analog to digital converter on-board, if the analog voltage has to be read by the Raspberry Pi, the external analog to digital converter has to be used. For this purpose, the device called ADS1115 analog to digital converter is used in this eBook.

Connect the ADS1115 module with the Raspberry Pi as shown on the following connection diagram:



ADS1115 pin	**>**	**Raspberry Pi pin**		
VDD	>	3V3	[pin 1]	**Red wire**
SDA	>	GPIO 2	[pin 3]	**Purple wire**
SCL	>	GPIO 3	[pin 5]	**Green wire**
GND	>	GND	[pin 9]	**Black wire**

ADS1115 pin	**>**	**Potentiometer pin**	
A0	>	Middle pin	**Blue wire**

Rasp. Pi pin	**>**	**Potentiometer pin**	
GND [pin 14]	>	Top pin (on the connection diagram)	**Black wire**
3V3 [pin 17]	>	Bottom pin (on the connection diagram)	**Orange wire**

Here, the potentiometer is used just as an example.

Libraries and tools for Python

To use the device with the Raspberry Pi it is recommended to download an external Python library. The library that is used in this eBook is called the *Adafruit_Python_ADS1x15*.

Before the library can be used, run the following commands:

```
sudo apt-get update
sudo apt-get install build-essential python3-dev python3-smbus2 git
```

Next, to download an external library, run the following command:

```
git clone https://github.com/adafruit/Adafruit_Python_ADS1x15
```

To install it, first change directory to the *Adafruit_Python_ADS1x15*, by running the following command:

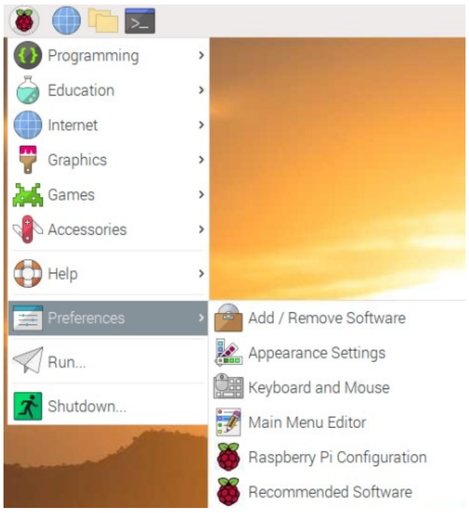
```
cd Adafruit_Python_ADS1x15
```

and install the library with the following command:

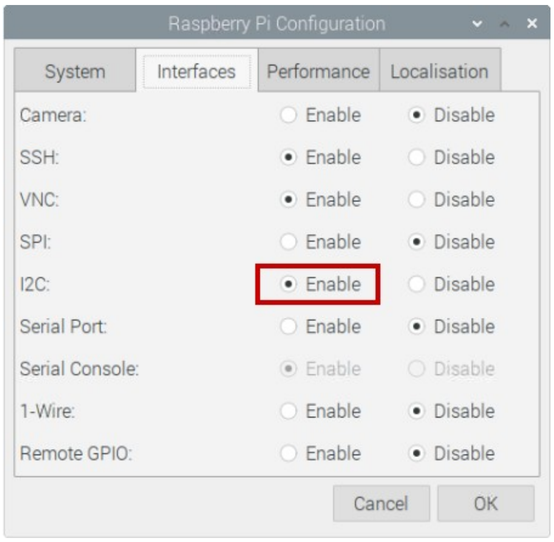
```
sudo python3 setup.py install
```

Enabling the I2C interface

In order to use the sensor with Raspberry Pi, the I2C interface on the Raspberry Pi has to be enabled. To do so, go to:
`Application Menu > Preferences > Raspberry Pi Configuration`



When a new window opens, find the *`Interfaces`* tab. Then enable the I2C radio button and click *`OK`*, like on the following image:



Camera:	Enable	[✓ Disable]
SSH:	[✓ Enable]	Disable
VNC:	[✓ Enable]	Disable
SPI:	Enable	[✓ Disable]
I2C:	[✓ Enable]	Disable
Serial Port:	Enable	[✓ Disable]
Serial Console:	[✓ Enable]	Disable
1-Wire:	Enable	[✓ Disable]
Remote GPIO:	Enable	[✓ Disable]

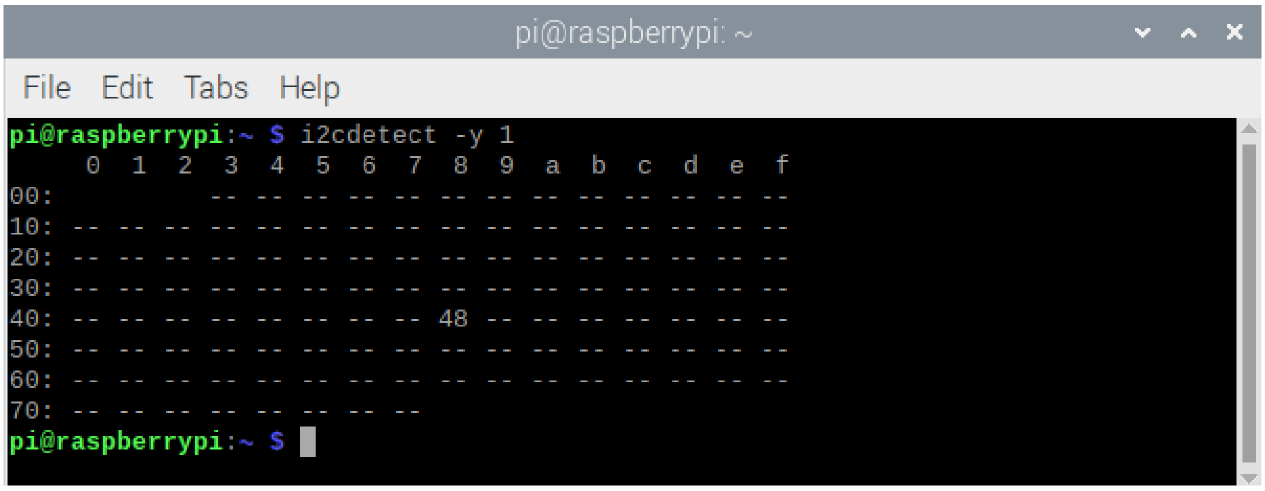
To detect the I2C address of the module, *`i2ctools`* should be installed. If there is none, following command is to be executed in the terminal window:

```
sudo apt-get install i2ctools -y
```

Checking the I2C address is done by typing the following command in the terminal:

```
i2cdetect -y 1
```

The terminal output should look like on the following image:



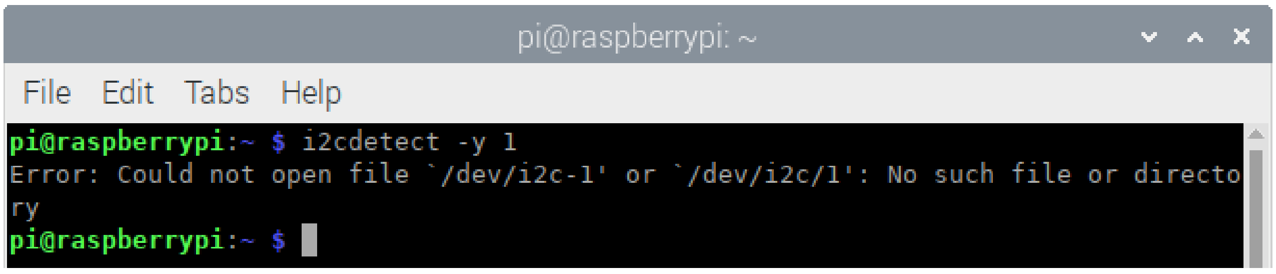
pi@raspberrypi: ~

File Edit Tabs Help

```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

The module I2C address is *0x48.*

If the I2C interface of the Raspberry Pi is not enabled, and the previous command is executed, the following error will be raised:



pi@raspberrypi: ~

File Edit Tabs Help

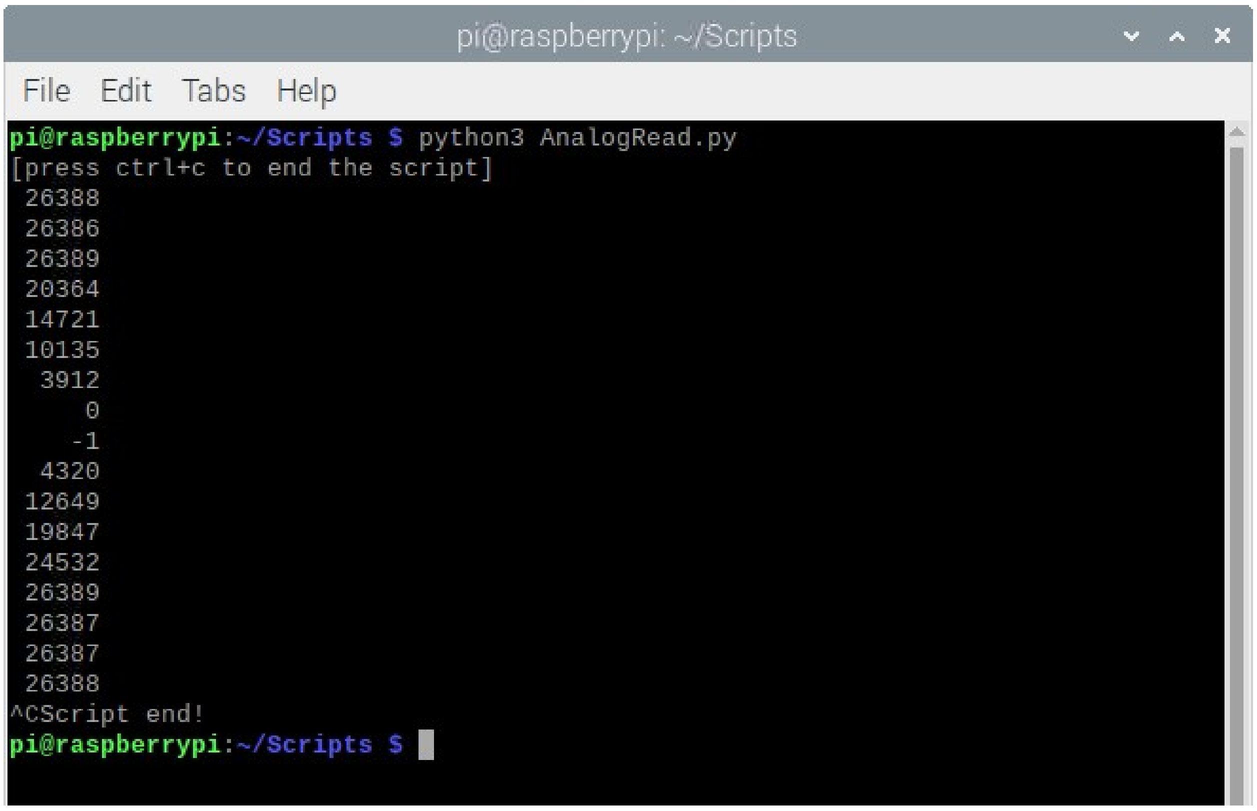
```
pi@raspberrypi:~ $ i2cdetect -y 1
Error: Could not open file `/dev/i2c-1' or `/dev/i2c/1': No such file or directo
ry
pi@raspberrypi:~ $
```

Test script for the ADS1115 module

```python
import time
import Adafruit_ADS1x15

adc = Adafruit_ADS1x15.ADS1115() # Create an ADS1115 ADC
GAIN = 1

print('[Press CTRL + C to end the script!]')
try: # Main program loop
    while True:
        # ADC channel 0 value
        values = adc.read_adc(0, gain=GAIN)
        print('{:>6}'.format(values))
        time.sleep(0.5)

# Scavenging work after the end of the program
except KeyboardInterrupt:
    print('\nScript end!')
```

Save the script by the name *AnalogRead.py*. To run the script open the terminal in the directory where the script is saved and run the following command: **`python3 AnalogRead.py`**

The result should look like the output on the following image:



```
pi@raspberrypi:~/Scripts $ python3 AnalogRead.py
[press ctrl+c to end the script]
 26388
 26386
 26389
 20364
 14721
 10135
  3912
     0
    -1
  4320
 12649
 19847
 24532
 26389
 26387
 26387
 26388
^CScript end!
pi@raspberrypi:~/Scripts $ 
```

To stop the script, press *CTRL + C* on the keyboard.

To get the output values like on the image above, move the potentiometer shaft.

To create and initialize *adc* object, the following line of code is used:

```
adc = Adafruit_ADS1x15.ADS1115()
```

The ADC data is read with the following line of code:

```
adc.read_adc(0, gain=GAIN)
```

Where *0* is the ADC pin name, which can be one of the following: *0*, *1*, *2* or *3*.

The *GAIN* is set to *1*, you can set it to any of the following values:

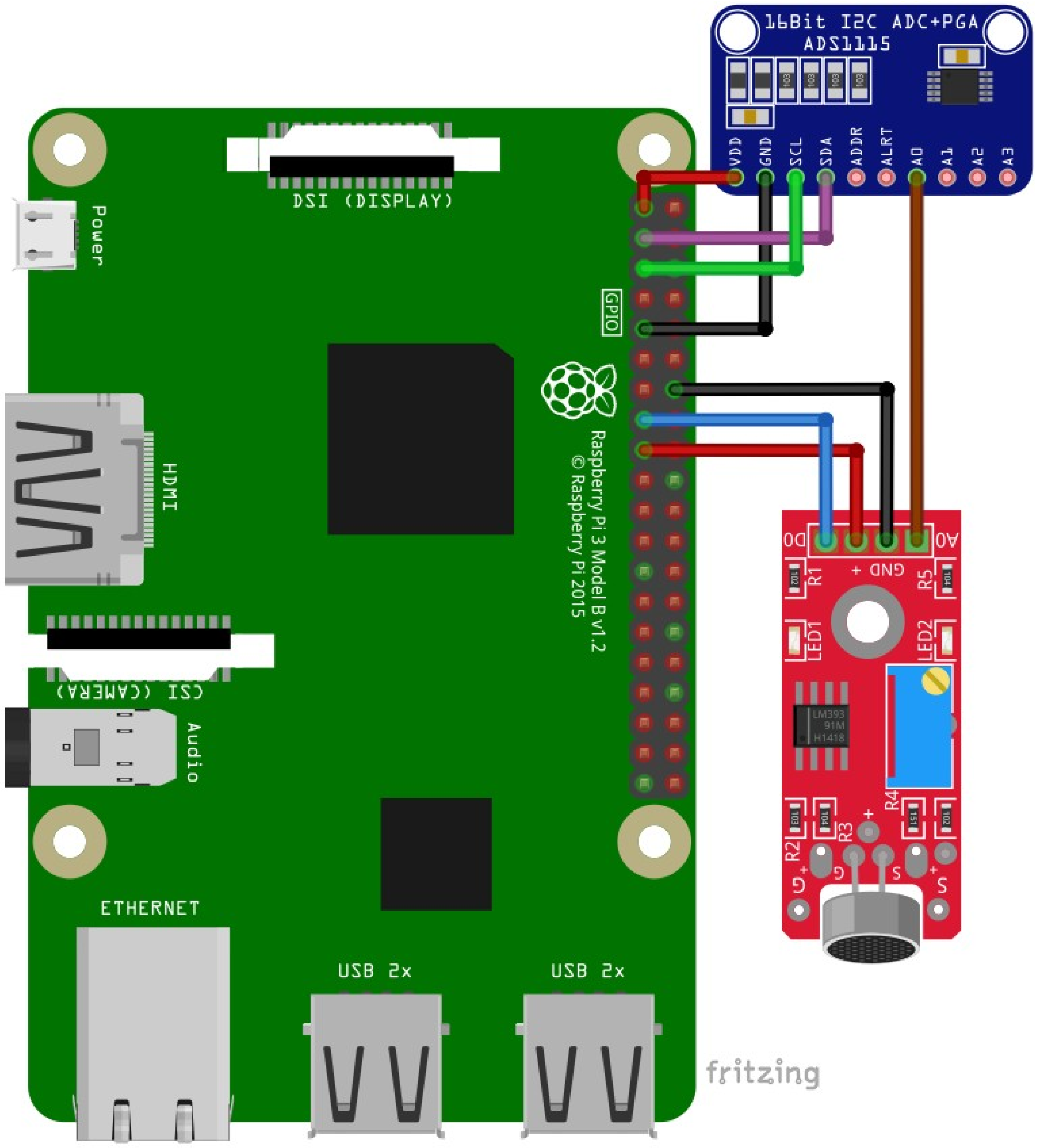
GAIN	**>**	**Voltage Levels**
0.66 (2/3)	>	±6.144V
1	>	±4.096V
2	>	±2.048V
4	>	±1.024V
8	>	±0.512V
16	>	±0.256V

The ADC data is stored into *values* variable, with the following line of code:

```
values = adc.read_adc(0, gain=GAIN)
```

Connecting the module with Raspberry Pi

Connect the KY-037 module with the Raspberry Pi as shown on the following connection diagram:



KY-037 pin	**>**	**Raspberry Pi pin**		
GND	>	GND	[pin 14]	**Black wire**
D0	>	GPIO22	[pin 15]	**Blue wire**
+ (VCC)	>	3V3	[pin 17]	**Red wire**

KY-037 pin	**>**	**ADS1115 pin**	
A0	>	A0	**Brown wire**

ADS1115 pin	**>**	**Raspberry Pi pin**		
VDD	>	3V3	[pin 1]	**Red wire**
GND	>	GND	[pin 9]	**Black wire**
SDA	>	GPIO2	[pin 3]	**Green wire**
SCL	>	GPIO3	[pin 5]	**Blue wire**

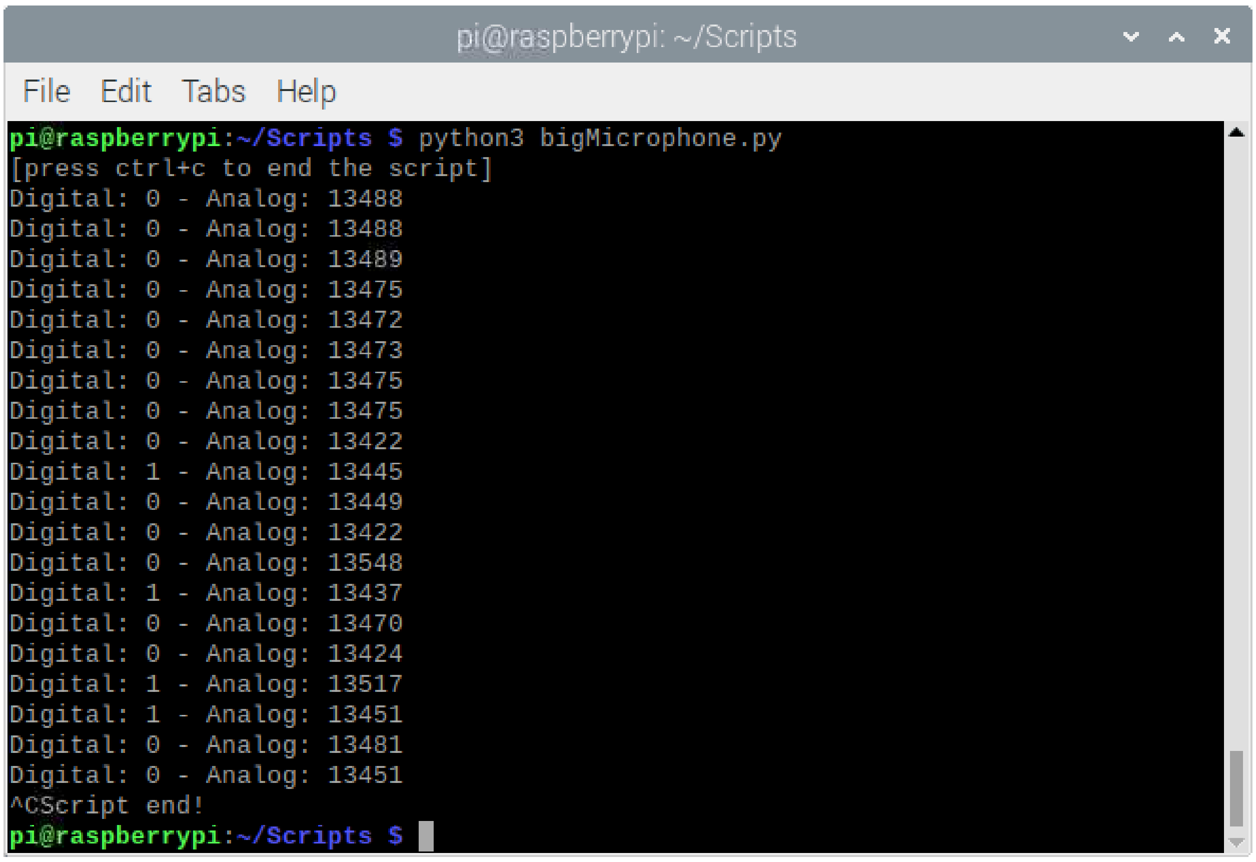
In this set-up, both the analog and digital output pins of the KY-037 module are used. Here, the device called *ADS1115* is used as an external analog to digital converter (ADC), where the analog data is sent to the Raspberry Pi via the I2C interface.

Python script

```
import time
import Adafruit_ADS1x15
import RPi.GPIO as GPIO

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

ADS1115 = Adafruit_ADS1x15.ADS1115()
GAIN = 1

Digital_PIN = 22
GPIO.setup(Digital_PIN, GPIO.IN)

print('[Press CTRL + C to end the script!]')
try: # Main program loop
     while True:
          analog = ADS1115.read_adc(0, gain=GAIN) # ADC channel 0
          digital = GPIO.input(Digital_PIN)
          print('Digital: {} - Analog: {}'.format(digital, analog))
          time.sleep(0.002)

# Scavenging work after the end of the program
except KeyboardInterrupt:
     print('\nScript end!')

finally:
     GPIO.cleanup()
```

Save the script by the name *bigMicrophone.py*. To run the script open the terminal in the directory where the script is saved and run the following command: **python3 bigMicrophone.py**

The result should look like the output on the following image:



```
pi@raspberrypi:~/Scripts $ python3 bigMicrophone.py
[press ctrl+c to end the script]
Digital: 0 - Analog: 13488
Digital: 0 - Analog: 13488
Digital: 0 - Analog: 13489
Digital: 0 - Analog: 13475
Digital: 0 - Analog: 13472
Digital: 0 - Analog: 13473
Digital: 0 - Analog: 13475
Digital: 0 - Analog: 13475
Digital: 0 - Analog: 13422
Digital: 1 - Analog: 13445
Digital: 0 - Analog: 13449
Digital: 0 - Analog: 13422
Digital: 0 - Analog: 13548
Digital: 1 - Analog: 13437
Digital: 0 - Analog: 13470
Digital: 0 - Analog: 13424
Digital: 1 - Analog: 13517
Digital: 1 - Analog: 13451
Digital: 0 - Analog: 13481
Digital: 0 - Analog: 13451
^CScript end!
pi@raspberrypi:~/Scripts $
```

To end the script press *CTRL + C* on the keyboard.

Now it is the time to learn and make your own projects. You can do that with the help of many example scripts and other tutorials, which can be found on the internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us