Welcome!

Thank you for purchasing our *AZ-Delivery GY-61 Accelerometer Sensor Module*. On the following pages, you will be introduced to how to use and set-up this handy device.

Have fun!

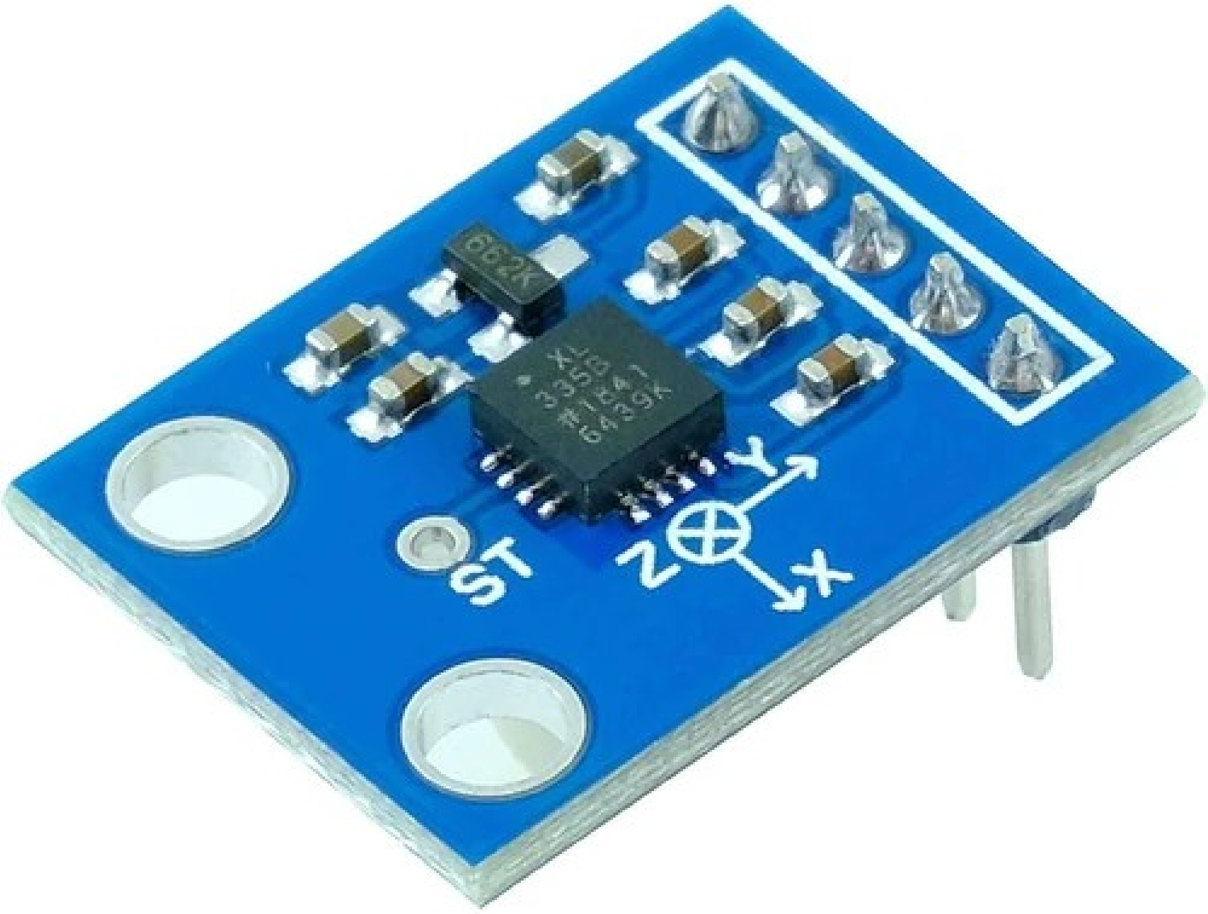


Table of Contents

Introduction

The GY-61 accelerometer module is a three axis accelerometer sensor module based on the ADXL335 integrated circuit which reads *X*, *Y* and *Z* axis acceleration and converts them in analog voltages.

By measuring the amount of acceleration due to gravity, an accelerometer can figure out the angle it is tilted at with respect to the earth. By sensing the amount of dynamic acceleration, the accelerometer can find out how fast and in what direction the device is moving.

The ADXL335 chip has low noise and power consumption. The sensor has a full sensing range of ±*3g*. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock or vibration.

The module is used in many applications such as mobile devices, gaming systems, disk drive protection, image stabilization, sports and health devices etc.

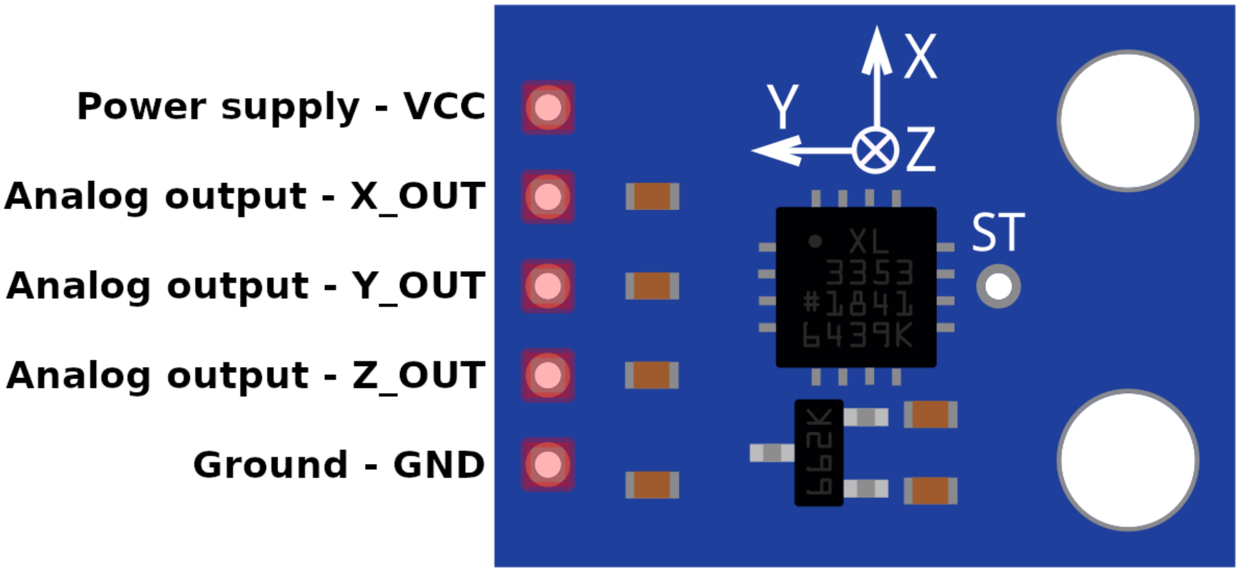
The module has an on-board *3.3V* voltage regulator to power the ADXL335, so power provided should be between +*3.3V* and +*5V* DC.

Specifications

» Operating voltage:	from 3V to 5V
» Current consumption:	40µA (0.1µA idle)
» Full scale range:	±3g
» Sensitivity:	300mV/g (Type)
» Sensor ouput:	Analog
» Voltage output:	centered at 1.65V
» Dimensions:	21 x 16 x 3mm [0.8 x 0.6 x 0.1in]

The pinout

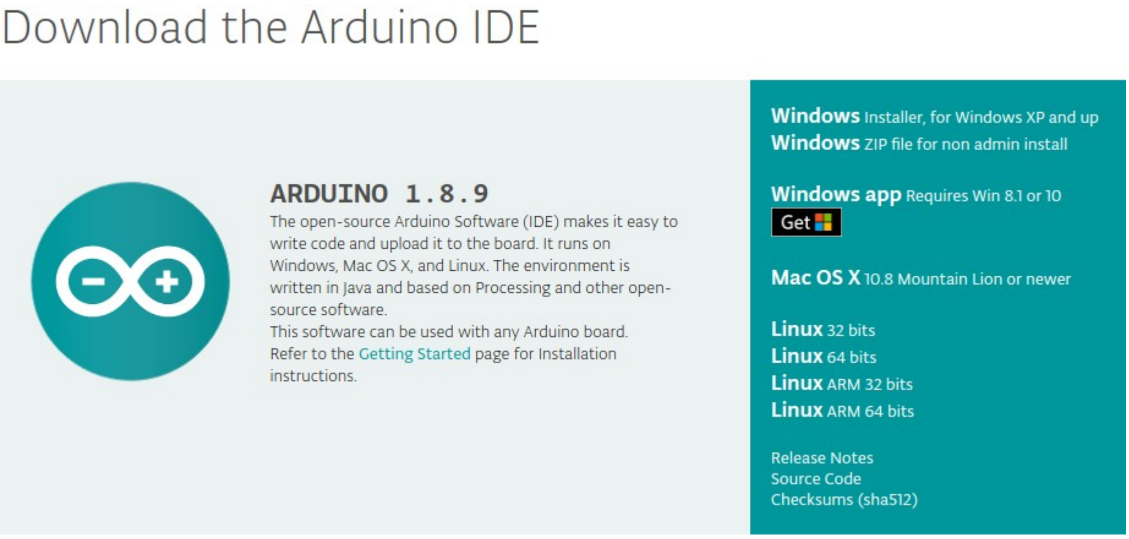
The GY-61 accelerometer sensor module has five pins. The pinout diagram is shown on the following image:



NOTE: Raspberry Pi can not read analog voltages, so the external analog to digital converter has to be used.

How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the link:
https://www.arduino.cc/en/Main/Software
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`*.

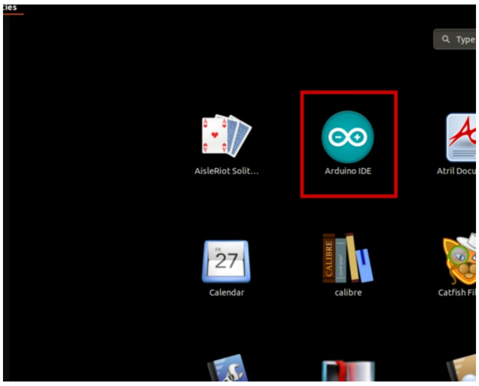
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

`sh arduino-linux-setup.sh user_name`

`user_name` - is the name of a superuser in 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.

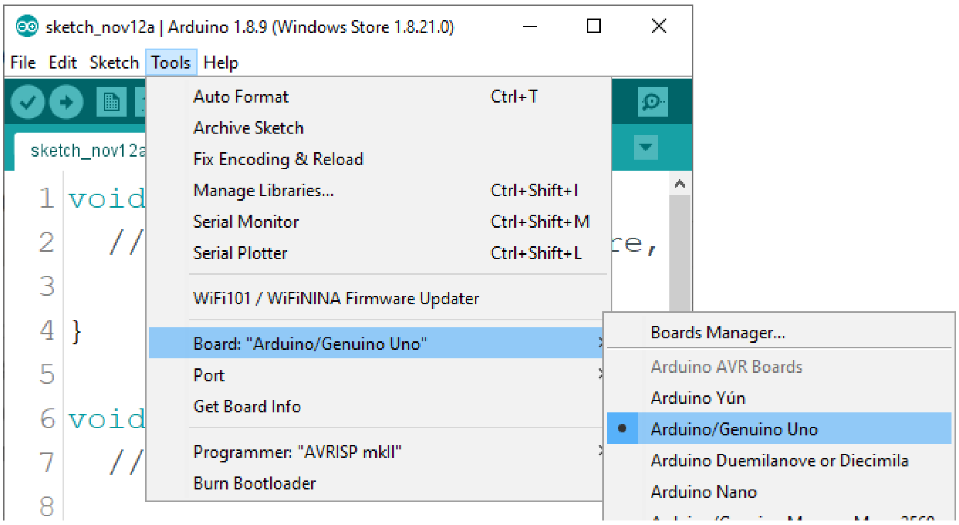
The second script, called *`install.sh`*, has to be installed after the installation of the first scritp. Run the following command in the terminal (extracted directory): **`sh install.sh`**

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



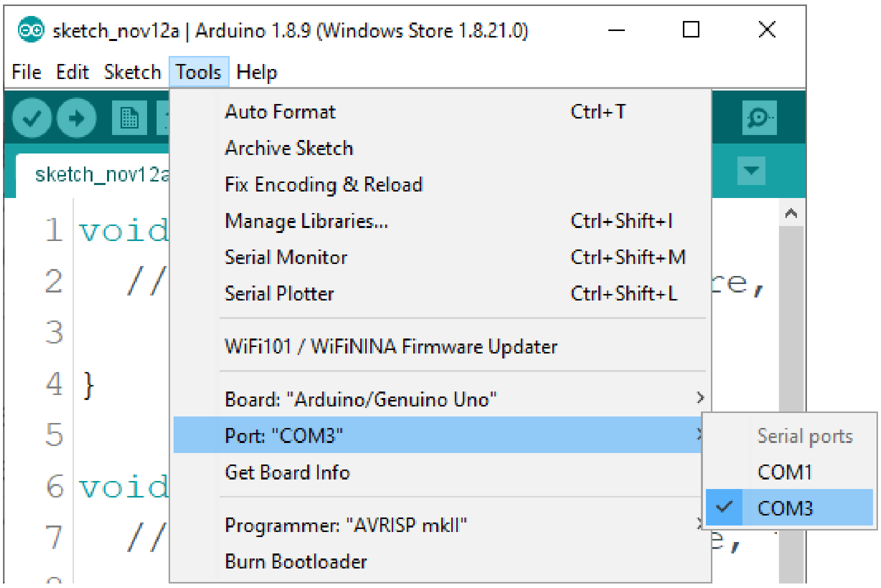
Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

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
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: "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

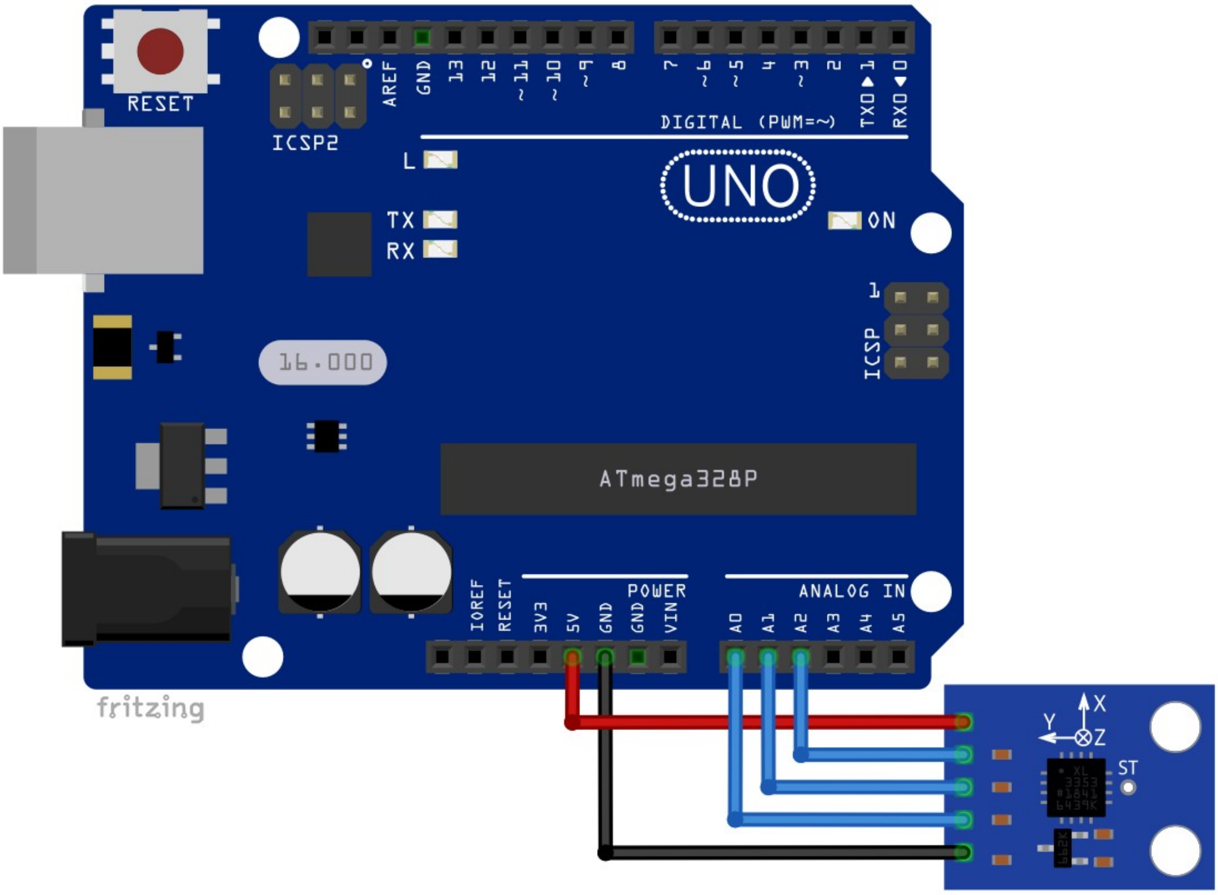
For the Raspberry Pi, first the operating system has to be installed, everything has to be set-u so that it can be used in *Headless* mode. *Headless* mode enables remote connection to the Raspberry Pi, without the need for a *PC* screen Monitor, mouse or keyboard. The only things that are used in this 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

which can be found on the following link:
https://www.az-delivery.de/products/raspberry-pi-kostenfreies-e-book?ls=en

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

Connecting the module with Uno

Connect the GY-61 module with the Uno as shown on the following connection diagram:

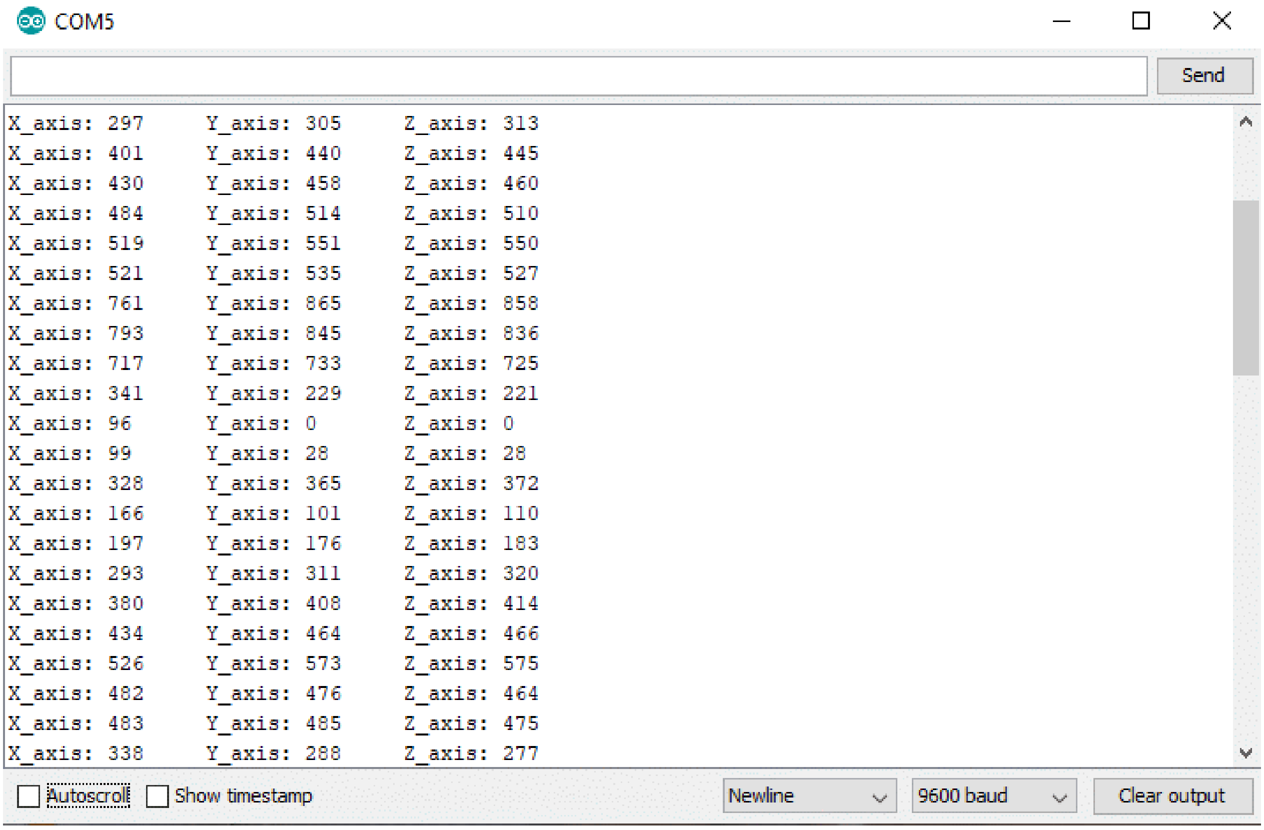


Sensor pin	**>**	**Uno pin**	
VCC	>	5V	**Red wire**
X_OUT	>	A2	**Blue wire**
Y_OUT	>	A1	**Blue wire**
Z_OUT	>	A0	**Blue wire**
GND	>	GND	**Black wire**

Sketch example

```
uint16_t x_axis = 0;
uint16_t y_axis = 0;
uint16_t z_axis = 0;

void setup() {
  Serial.begin(9600);
}
void loop() {
  x_axis = analogRead(A0);
  y_axis = analogRead(A1);
  z_axis = analogRead(A2);

  Serial.print("X_axis: ");
  Serial.print(x_axis);
  Serial.print ("\t");
  Serial.print("Y_axis: ");
  Serial.print(y_axis);
  Serial.print("\t");
  Serial.print("Z_axis: ");
  Serial.println(z_axis);

  delay(500);
}
```

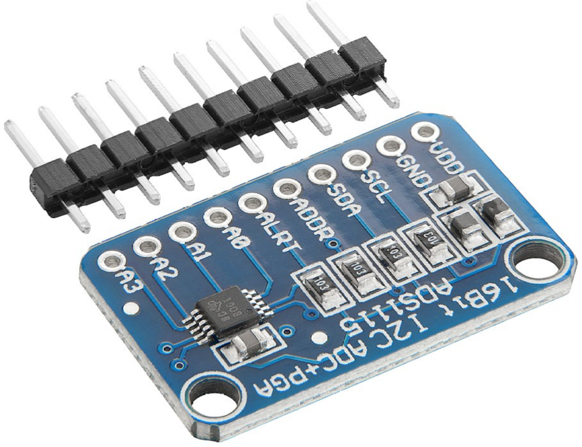
Upload the sketch to the Uno and open Serial Monitor (*Tools* > *Serial Monitor*). The result should look like the output on the following image:



```
X_axis: 297     Y_axis: 305     Z_axis: 313
X_axis: 401     Y_axis: 440     Z_axis: 445
X_axis: 430     Y_axis: 458     Z_axis: 460
X_axis: 484     Y_axis: 514     Z_axis: 510
X_axis: 519     Y_axis: 551     Z_axis: 550
X_axis: 521     Y_axis: 535     Z_axis: 527
X_axis: 761     Y_axis: 865     Z_axis: 858
X_axis: 793     Y_axis: 845     Z_axis: 836
X_axis: 717     Y_axis: 733     Z_axis: 725
X_axis: 341     Y_axis: 229     Z_axis: 221
X_axis: 96      Y_axis: 0       Z_axis: 0
X_axis: 99      Y_axis: 28      Z_axis: 28
X_axis: 328     Y_axis: 365     Z_axis: 372
X_axis: 166     Y_axis: 101     Z_axis: 110
X_axis: 197     Y_axis: 176     Z_axis: 183
X_axis: 293     Y_axis: 311     Z_axis: 320
X_axis: 380     Y_axis: 408     Z_axis: 414
X_axis: 434     Y_axis: 464     Z_axis: 466
X_axis: 526     Y_axis: 573     Z_axis: 575
X_axis: 482     Y_axis: 476     Z_axis: 464
X_axis: 483     Y_axis: 485     Z_axis: 475
X_axis: 338     Y_axis: 288     Z_axis: 277
```

□ Autoscroll □ Show timestamp

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. AZ-Delivery offers this kind of a device, it is called *ADS1115 Analog to digital converter*.

The *ADS1115* module has *16* bit digital precision and uses the I2C interface to send data to the 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.

For more information about this device, there is a free eBook called:
ADS1115 Analog to digital converter Quick Starter Guide

To download this eBook, go to our website with the following link:
https://www.az-delivery.com/products/kostenfreies-e-book-ads1115-analog-digitalwandler?pos=3&sid=fd4e7cb0d&ss=r

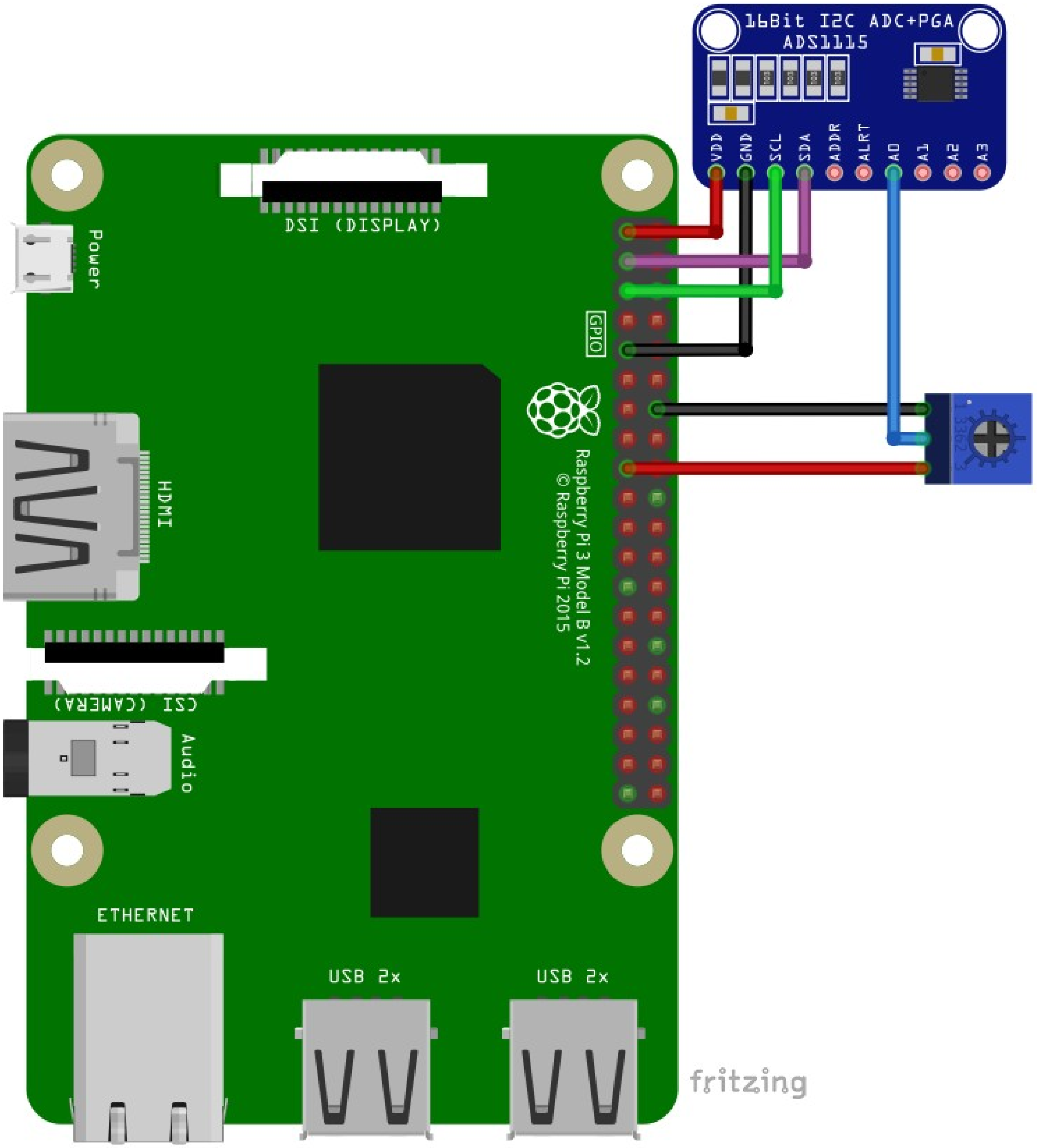
The module can read both positive and negative voltages. The first bit in digital value is for the sign (positive or negative voltage), which means that the real precision of the module is *15* bits, with the *16th* bit being the sign bit.

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. You can use any of the on-board analog pins (from *0* to *3*).

For example, the ADC in the ADS1115 module is much precise than the ADC in the Uno.

Connecting the ADS1115 module with Raspberry Pi

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	**Black wire**
3V3 [pin 17]	>	Bottom pin	**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
```

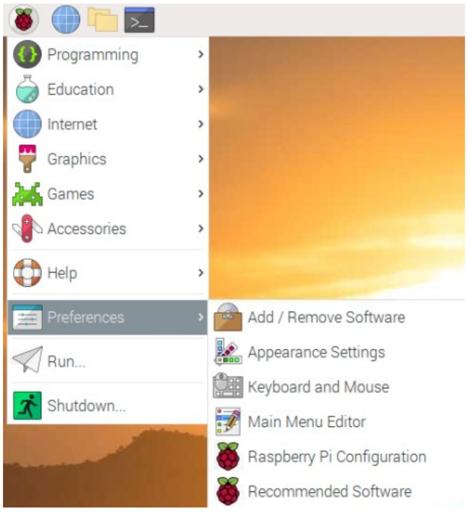
and install the library with the following command:

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

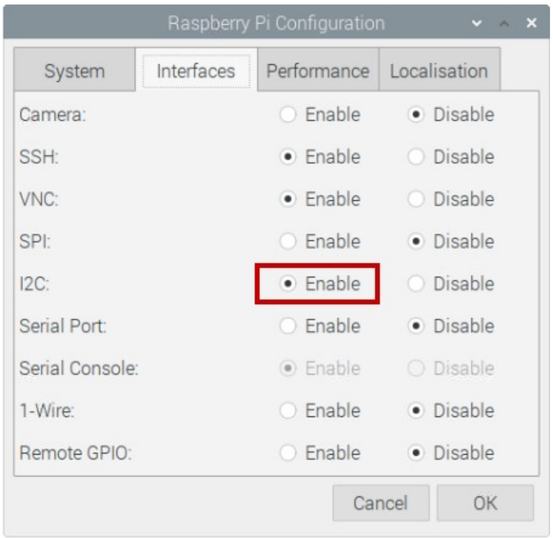
Enabling the I2C interface

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:



Raspberry Pi Configuration

System | Interfaces | Performance | Localisation

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

Cancel OK

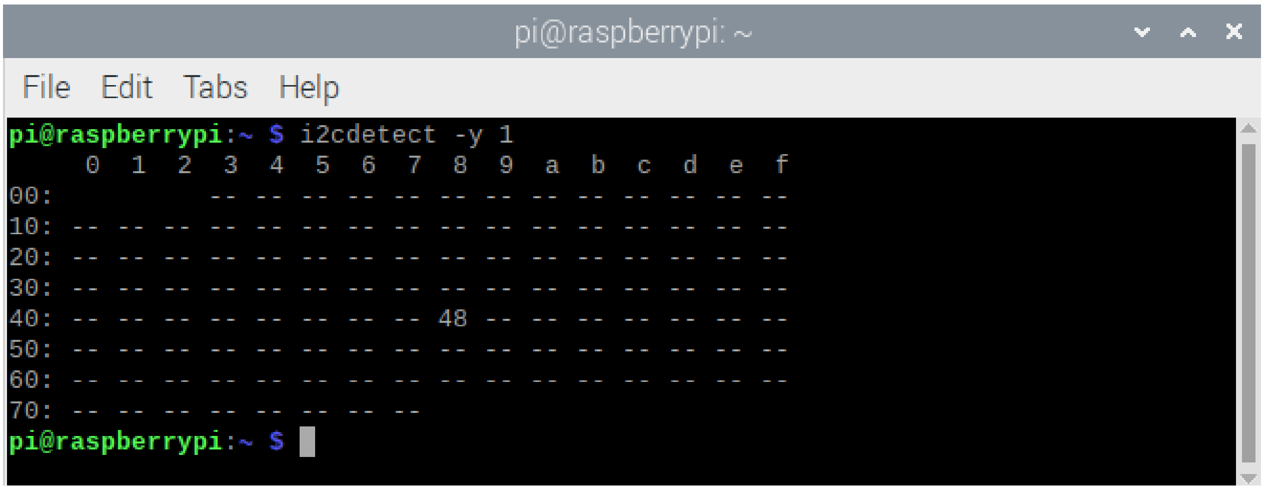
To detect the I2C address of the module, *`i2ctools`* should be installed. If there is none, following command is to be executed in the terminal window:

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

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

```
i2cdetect -y 1
```

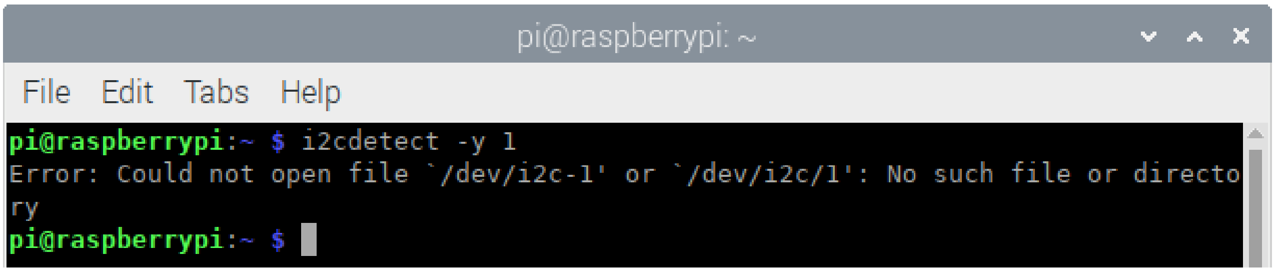
The terminal output should look like on the following image:



```
pi@raspberrypi:~ $ 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:



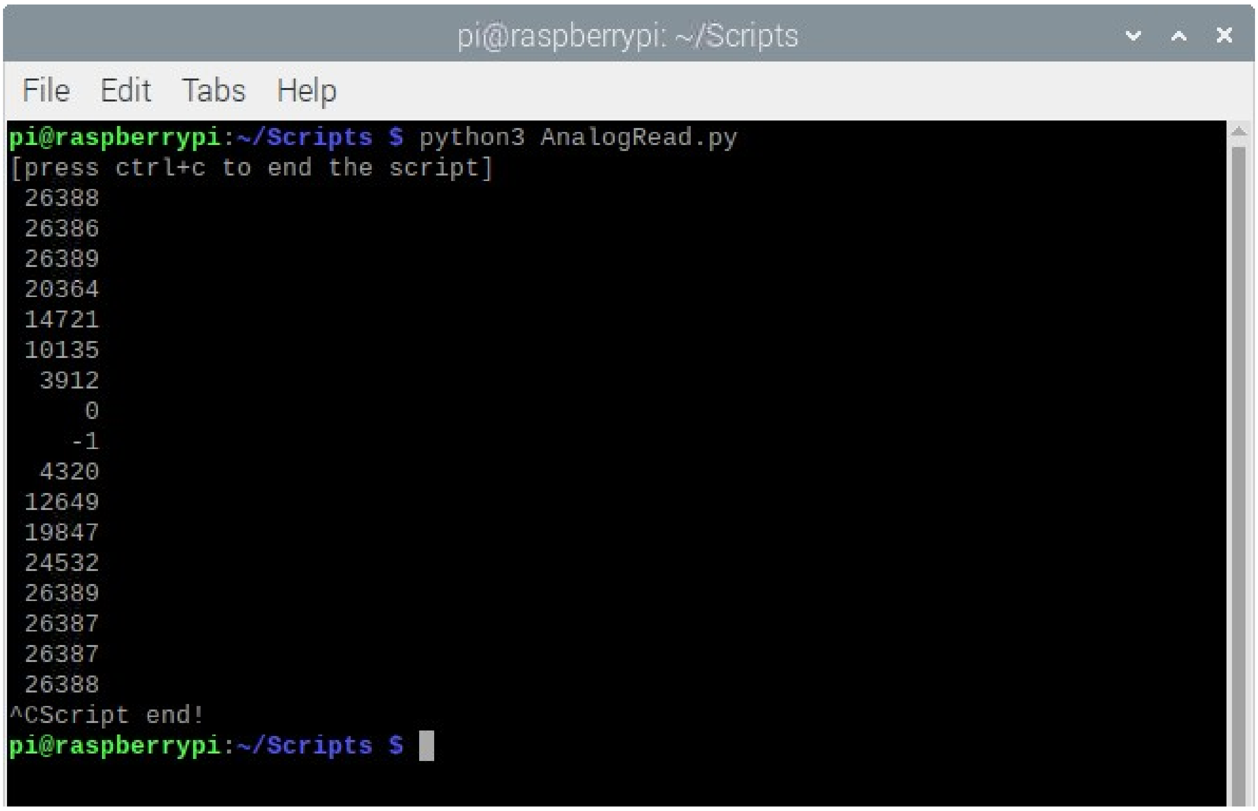
```
pi@raspberrypi:~ $ i2cdetect -y 1
Error: Could not open file `/dev/i2c-1' or `/dev/i2c/1': No such file or directo
ry
pi@raspberrypi:~ $
```

Test script for 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 under the name *AnalogRead.py*. To run the script open the terminal in the directory where you saved the script 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 the *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:

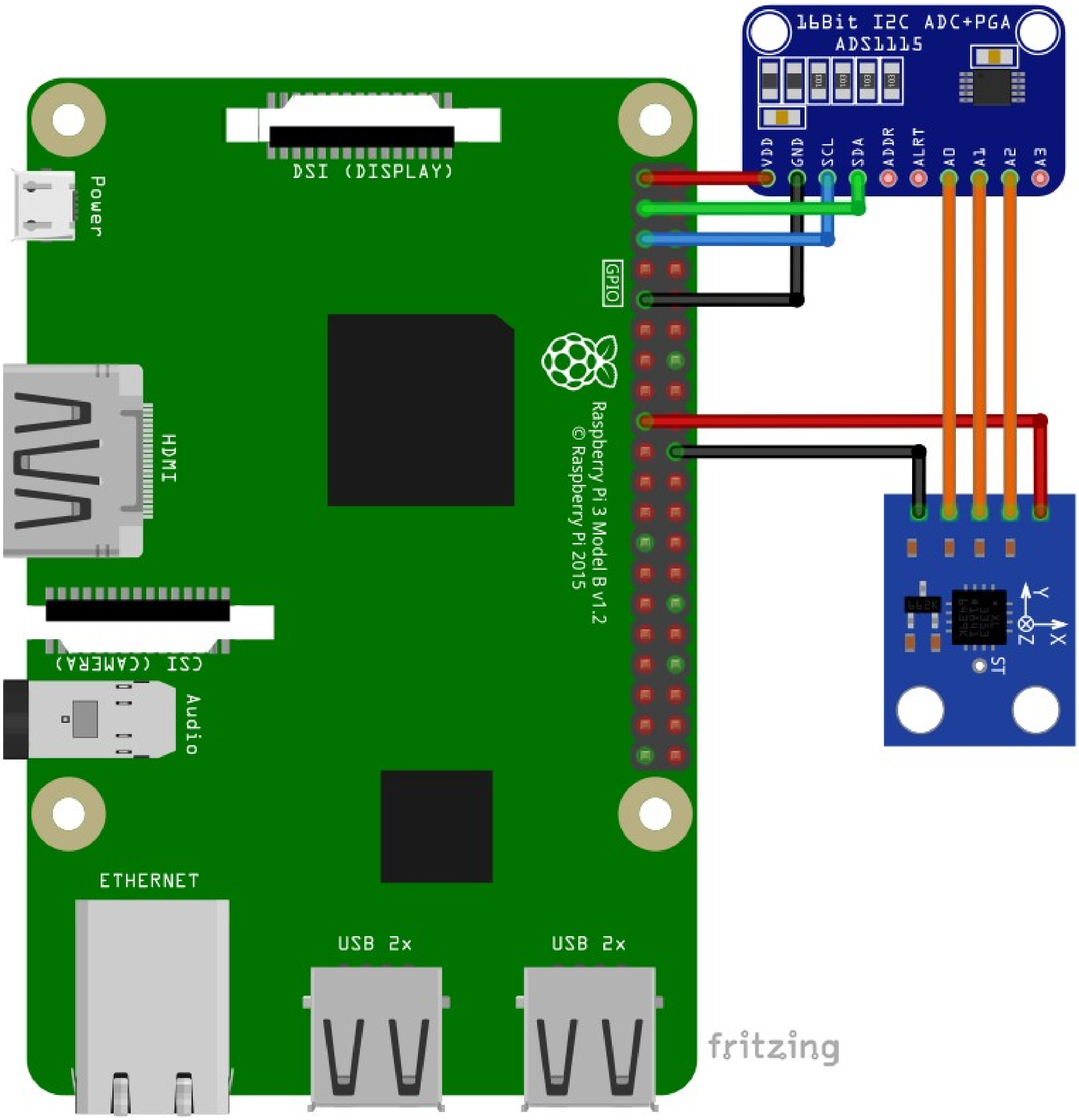
GAIN	**>**	**Voltage Levels**
0.66 (2/3)	>	±6.144V
1	>	±4.096V
2	>	±2.048V
4	>	±1.024V
8	>	±0.512V
16	>	±0.256V

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

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

Connecting the module with Raspberry Pi

Connect the GY-61 module with the Raspberry Pi as shown on the following connection diagram:



Sensor pin	**>**	**Raspberry Pi pin**	
VCC	>	3V3 [pin 17]	**Red wire**
GND	>	GND [pin 20]	**Black wire**

Sensor pin	**>**	**ADS1115 pin**	
X_OUT	>	A2	**Orange wire**
Y_OUT	>	A1	**Orange wire**
Z_OUT	>	A0	**Orange wire**

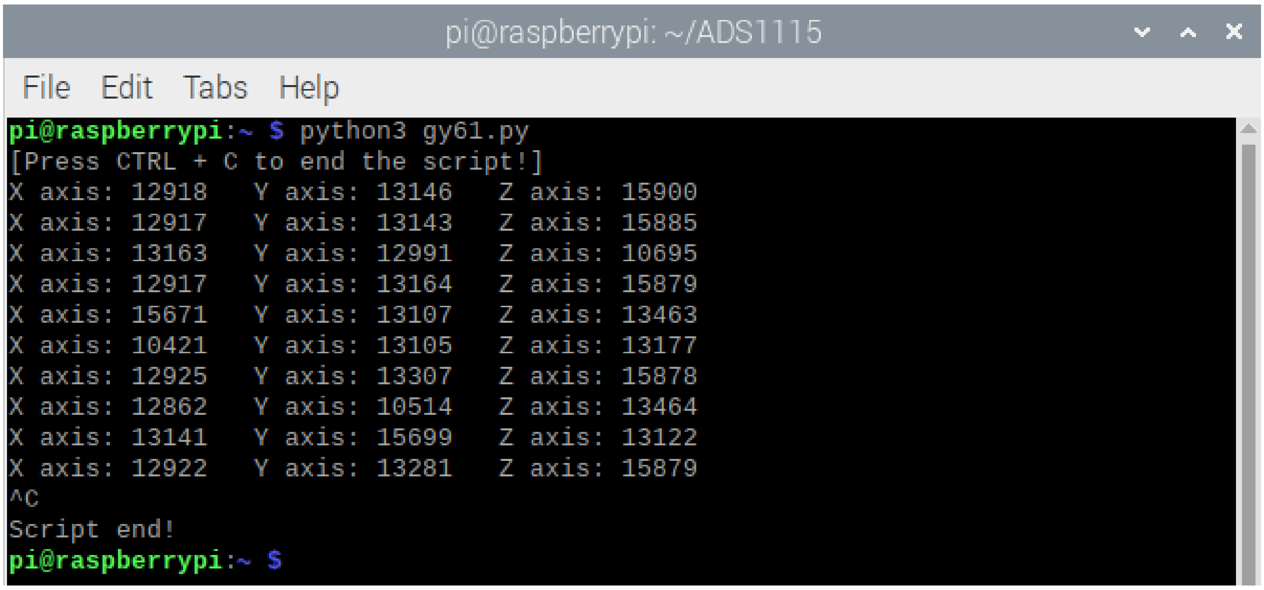
ADS1115 pin	**>**	**Raspberry Pi pin**		
VDD	>	3V3	[pin 1]	**Red wire**
GND	>	GND	[pin 9]	**Black wire**
SDA	>	GPIO2	[pin 3]	**Green wire**
SCL	>	GPIO3	[pin 5]	**Blue wire**

Python script

```python
import time
from ADS1x15 import ADS1115

adc = ADS1115()
GAIN = 1

values = list()

print('Press CTRL + C to end the script!')
try:
  while True:
    for i in range(3):
      values.append(adc.read_adc(i, gain=GAIN))

    print('X axis:{:>6}\tY axis:{:>6}\tZ axis:{:>6}'.
      format(values[2], values[1], values[0]))
    time.sleep(5)

    values = list()

except KeyboardInterrupt:
  print('\nScript end!')
```

Save the script under the name *gy61.py*. To run the script open the terminal in the directory where you saved the script and run the following command: **python3 gy61.py**

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



```
pi@raspberrypi:~ $ python3 gy61.py
[Press CTRL + C to end the script!]
X axis: 12918   Y axis: 13146   Z axis: 15900
X axis: 12917   Y axis: 13143   Z axis: 15885
X axis: 13163   Y axis: 12991   Z axis: 10695
X axis: 12917   Y axis: 13164   Z axis: 15879
X axis: 15671   Y axis: 13107   Z axis: 13463
X axis: 10421   Y axis: 13105   Z axis: 13177
X axis: 12925   Y axis: 13307   Z axis: 15878
X axis: 12862   Y axis: 10514   Z axis: 13464
X axis: 13141   Y axis: 15699   Z axis: 13122
X axis: 12922   Y axis: 13281   Z axis: 15879
^C
Script end!
pi@raspberrypi:~ $
```

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

You have done it!

Now you can use your module for various projects.

Now is the time to learn and make the Projects on your own. 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