Welcome!

Thank you for purchasing our *AZ-Delivery MQ-2 Gas Sensor Module*. On the following pages, you will be introduced to how to use and set up this handy device.

Have fun!

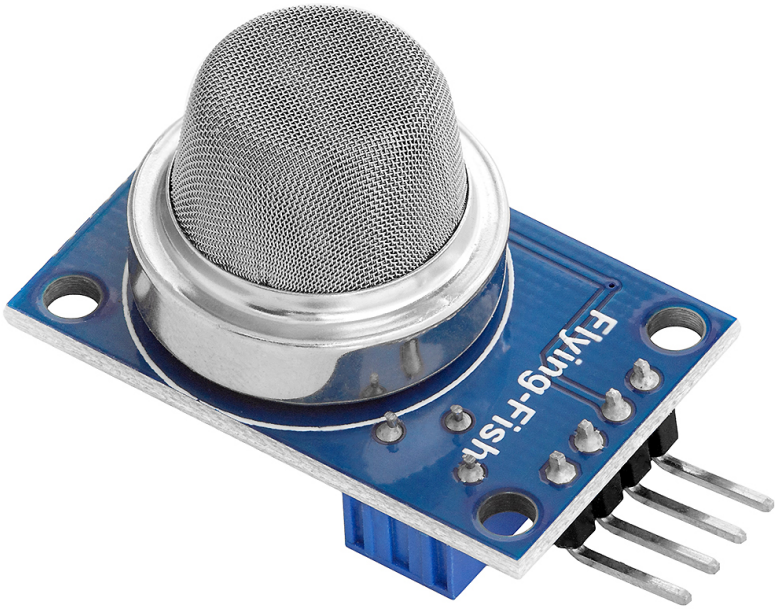


Table of Contents

Introduction

The MQ-2 gas sensor module is a device that is used for sensing and measuring the concentration of gases in the air. It can detect such gases as: LPG, propane, methane, hydrogen, alcohol, smoke and carbon monoxide. Though it can detect those gases, it is not able to distinguish the difference between them.

The MQ-2 is a Metal Oxide Semiconductor (MOS), also known as a chemiresistor. The sensor contains a sensing material which resistance changes with different gas concentrations. This change of the resistance is used for gas detection. The sensor also has a built-in potentiometer, with which we can adjust its sensitivity.

The sensor is enclosed within two layers of fine stainless steel mesh called *Anti-explosion network*. As a result of that, it is able to detect flammable gases without incidents. Likewise, it provides protection for the sensor, and it filters out suspended particles. That way, only gases are able to pass inside the sensing chamber.

The module has an on-board LM393 comparator chip which converts the readings into digital and analog signals. There is also a potentiometer which is used to calibrate detection sensitivity.

Specifications

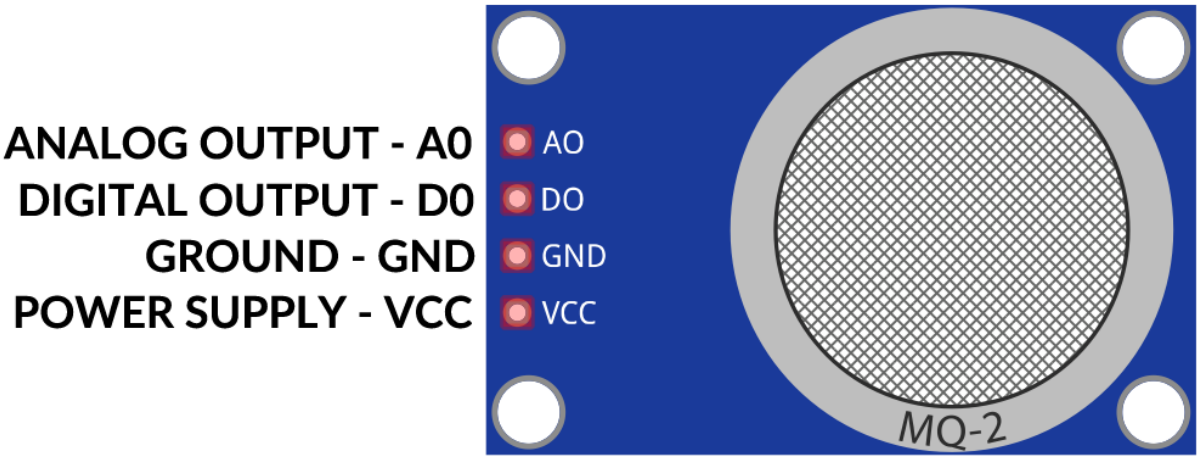
Operating voltage:	5V
Operating current:	150mA
Power consumption:	900mW
Load resistance:	20kΩ
Heater resistance:	33Ω+5%
Sensing resistance	10kΩ - 60kΩ
Preheat time:	24h
Concentration scope:	200 – 10000ppm (parts per million)
Output:	analog, digital
Dimensions:	33x21x22mm (1.3x0.8x0.9in)

For the best detecting results, gas sensor has to be preheated. The best preheat time for the sensor is above 24 hours. For the detailed information about the sensor specifications, refer to the datasheet.

The module sensitivity can be adjusted with an on-board potentiometer. Moving the potentiometer shaft into the clockwise direction increases the sensitivity. Moving the shaft of the potentiometer in the counterclockwise direction decreases the sensitivity of the module.

The pinout

The gas sensor module has four pins. The pinout is shown on the following image:



NOTE: The Raspberry Pi does not have a digital-analog converter and can not be used to read analog voltages.

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.

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*.

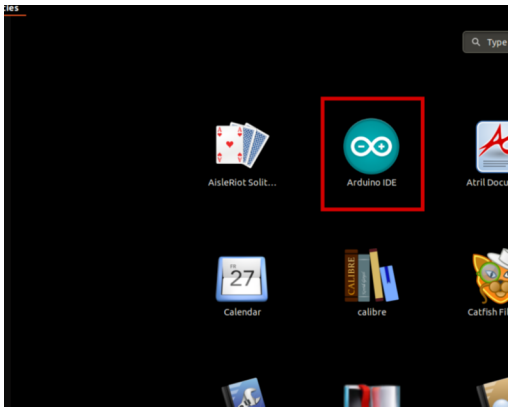
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

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.

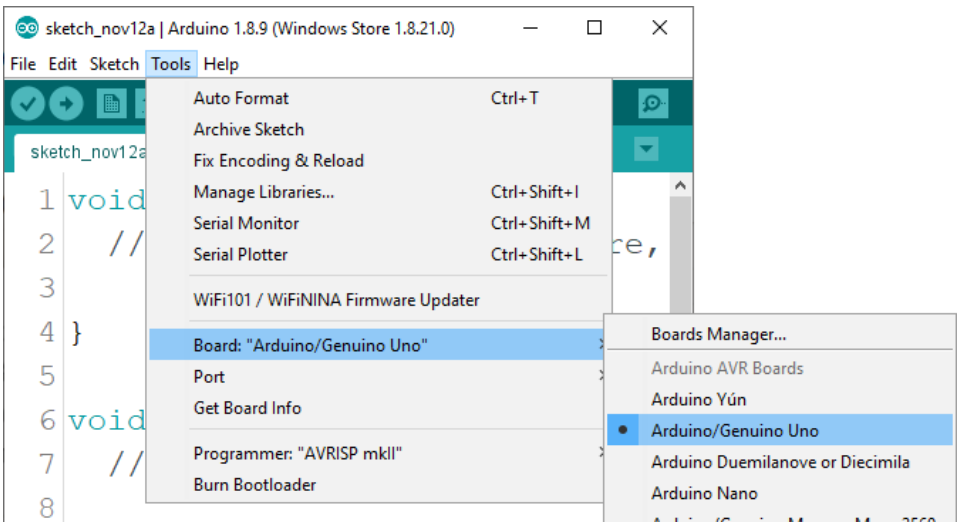
The second script, called *install.sh*, has to be used after the installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

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



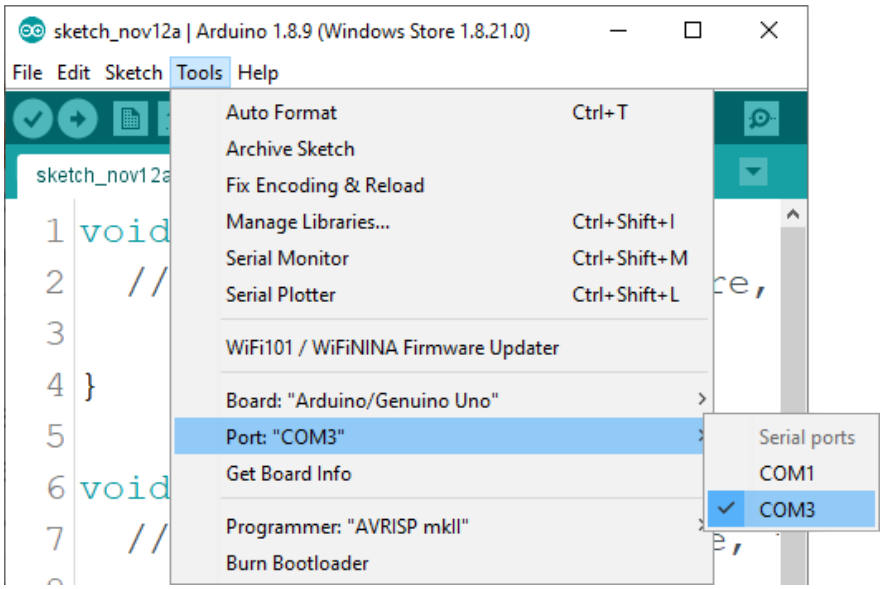
Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

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:



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:



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

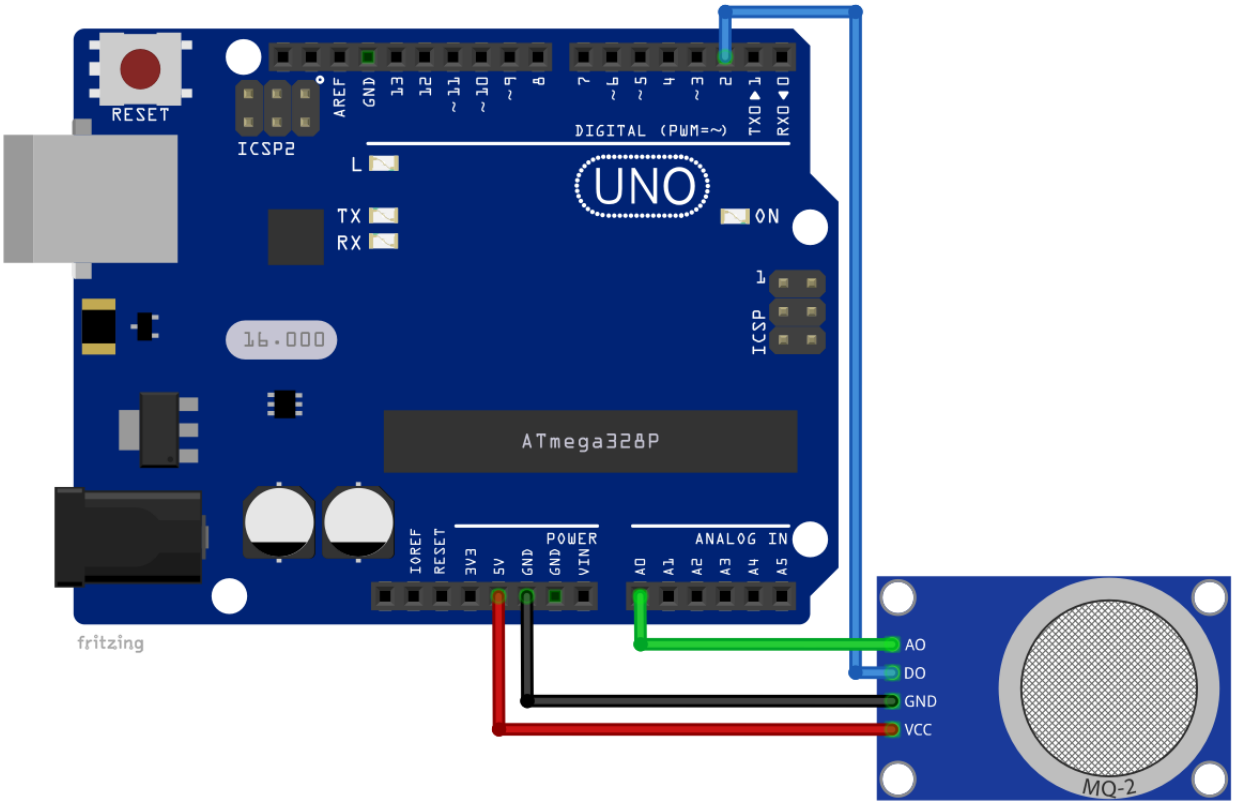
How to set-up the Raspberry Pi and Python

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 module with the Uno as shown on the following image:

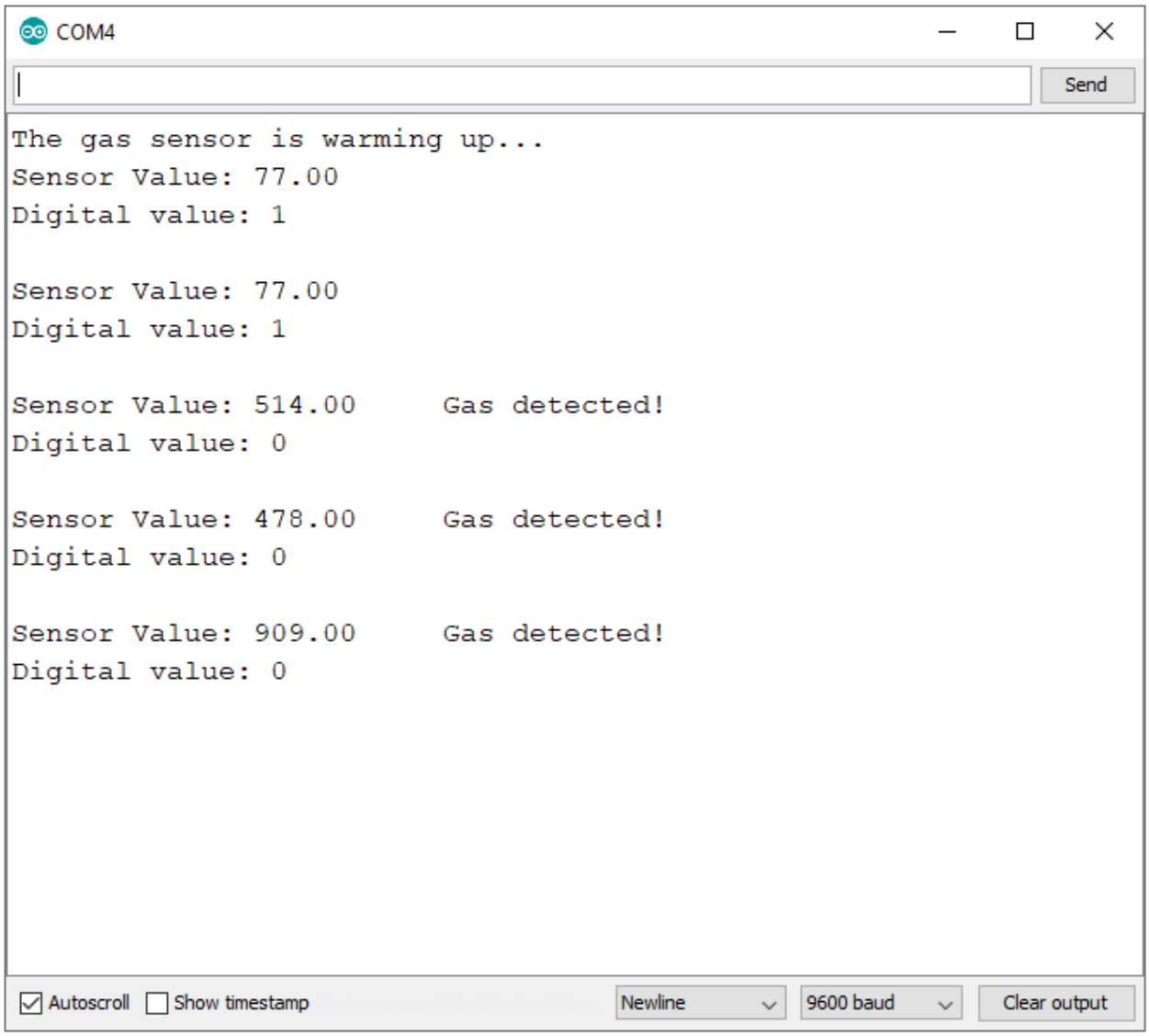


Module pin	Uno pin	Wire color
VCC	5V	**Red wire**
GND	GND	**Black wire**
D0	D2	**Blue wire**
A0	A0	**Green wire**

Sketch example

```
#define DIGITAL_PIN 2
#define ANALOG_PIN 0
uint16_t gasVal;
boolean isgas = false;
String gas;
void setup() {
  Serial.begin(9600);
  Serial.println("The sensor is warming up...");
  delay(30000);
  pinMode(DIGITAL_PIN, INPUT);
}
void loop() {

  gasVal = analogRead(ANALOG_PIN);
  isgas = digitalRead(DIGITAL_PIN);

  if (isgas) {
    gas = "No";
  }
  else {
    gas = "Yes";
  }
  gasVal = map(gasVal, 0, 1023, 0, 100);
  Serial.print("Gas detected: ");
  Serial.println(gas);
  Serial.print("Gas percentage: ");
  Serial.print(gasVal);
  Serial.print("%\n");
  delay(2000);
}
```

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



The sketch starts with defining and creating two macros called *DIGITAL_PIN*, *ANALOG_PIN*.

The *DIGITAL_PIN* represents the digital pin of Uno that is used for connecting the digital output pin of the sensor.

The *ANALOG_PIN* represents the analog input pin of Uno that is used for connecting the analog output pin of the sensor.

The module data can be read in two ways. The one is by reading the analog output pin of the module, and the other is by reading the digital output pin of the module. To read the analog output pin of the module, the variable called *gasVal* is used to store return value from the *analogRead()* function. The return value is an integer number in the range from 0 to 1023. To convert it into a percentage, the *map()* function is used. This is a built-in function of the Arduino IDE. It has five arguments and returns an integer value.

For example:

```
gasVal = map(input, in_min, in_max, out_min, out_max)
```

First argument is the *input* value, which is in the range from the *in_min* to *in_max*. The return value is an integer number in the range from *out_min* to *out_max*. This function maps one number in the input range, to other number which is in the different range.

To read the digital output pin of the module, the *isGas* variable is used to store the return value of the *digitalRead()* function.

At the end of the *loop()* function, the data is displayed in the Serial Monitor. Between two measurements there is 2 seconds pause:

```
delay(2000);
```

Connecting Nano as ADC for Raspberry Pi

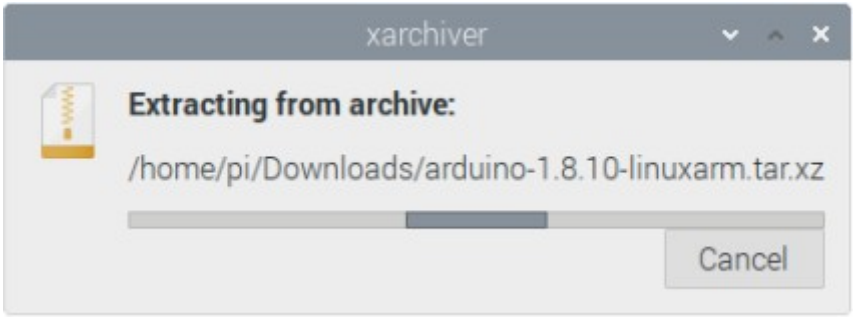
Because the Raspberry Pi does not have Analog to Digital Converter (ADC), the task is to make the Raspberry Pi able to read analog voltages. For this purpose Uno or Nano can be used. In order to do so, Nano has to be connected to the Raspbian operating system. Nano can read analog voltages, and it can use Serial Interface via USB port to send data to the Raspberry Pi.

First, the Arduino IDE has to be installed on the Raspbian. Second, the firmware for Arduino needs to be uploaded to Nano and Python library has to be downloaded.

To do this, start the Raspbian, open the terminal, and run the following command to update the Raspbian:
sudo apt-get update && sudo apt-get upgrade -y

To download and install the Arduino IDE, go to the *Arduino site*:
and download the `tar.xz` file of Arduino IDE for `Linux ARM 32 bits` as shown on the following image:

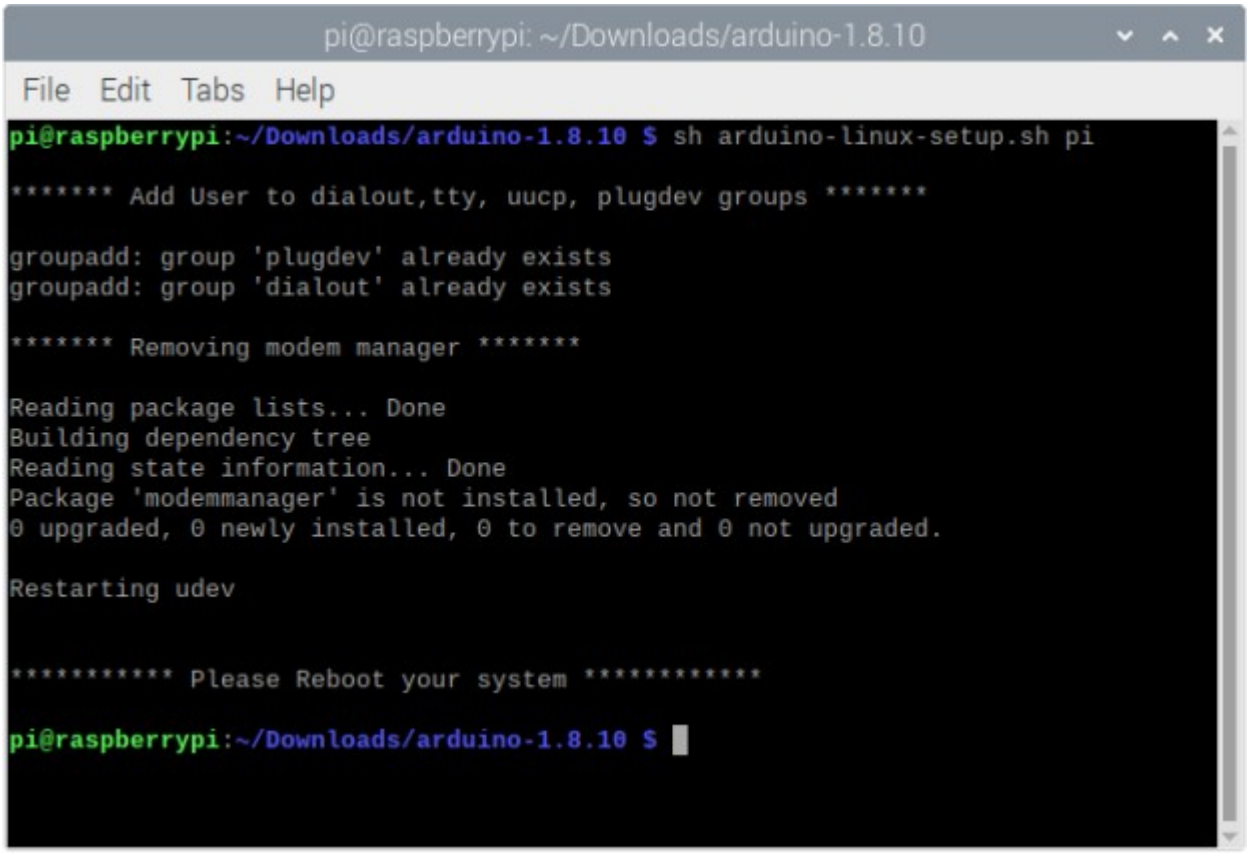
Then, extract the `tar.xz` file. Open file explorer in directory where `tar.xz` file is downloaded, right click on it, and run the option Extract Here. Wait for a few minutes for the extracting process to be completed.



Open the terminal in the directory where installation files are extracted and run the following command:

`sh arduino-linux-setup.sh pi`

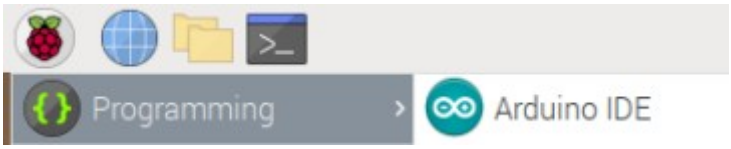
where *`pi`* is the name of the superuser in Raspbian.



After this, to install the Arduino IDE, run the following command:

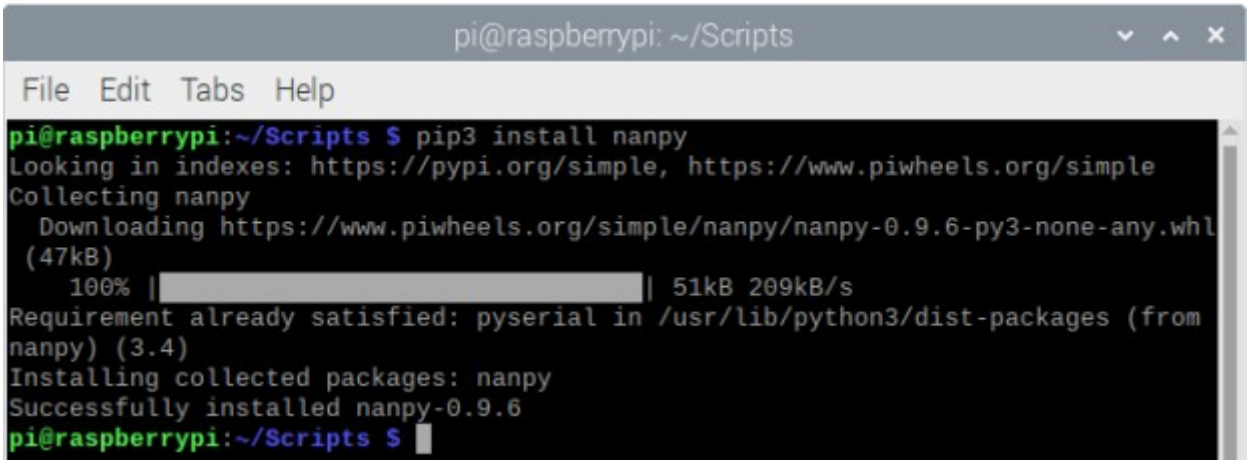
`sudo sh install.sh`

The Arduino IDE is now installed. To run Arduino IDE, open the app:
Applications Menu > *Programming* > *Arduino IDE*



Before the next steps, first the *pip3* and *git* apps have to be installed; Open the terminal and run the following command.
sudo apt install python3-pip git -y

The library for Python is called *nanpy*. To install it, open terminal and run the following command: **pip3 install nanpy**



After installation of the *nanpy* library, download an Arduino firmware by running the following command:

`git clone https://github.com/nanpy/nanpy-firmware.git`

Change the directory to *nanpy-firmware* by running the following command:

`cd nanpy-firmware`

And run the following command:

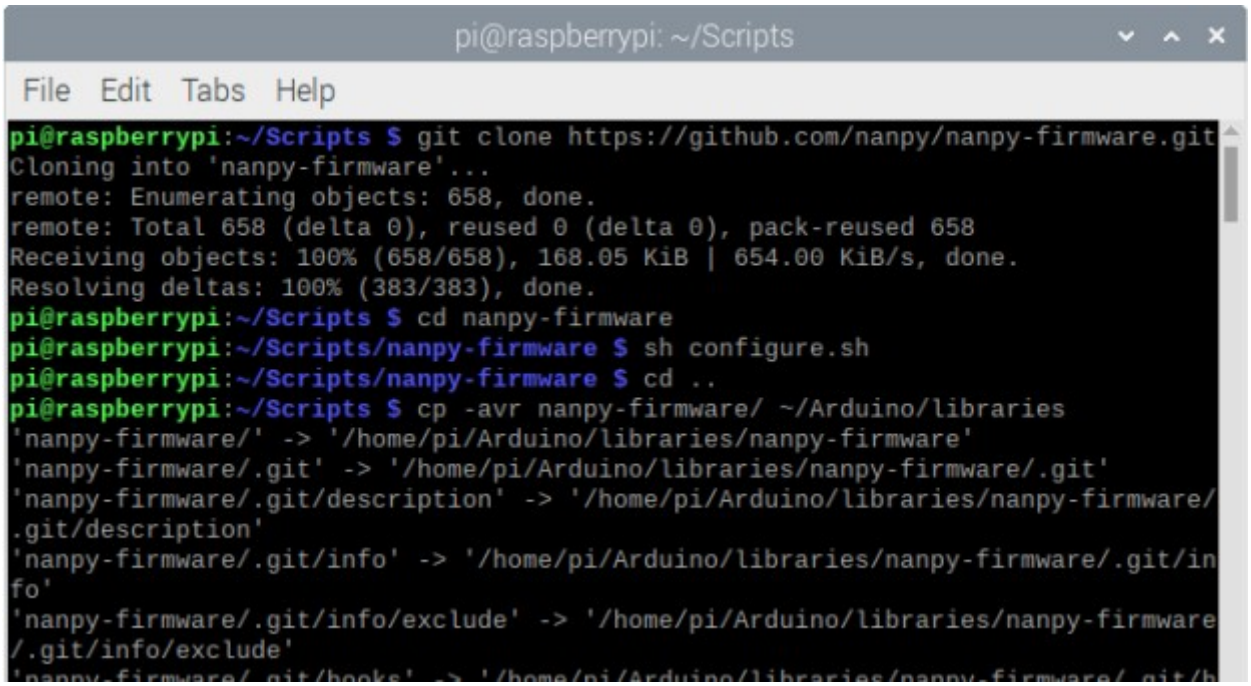
`sh configure.sh`

Next, copy the *nanpy-firmware* directory into:

Arduino/libraries directory.

To do so, run the following command:

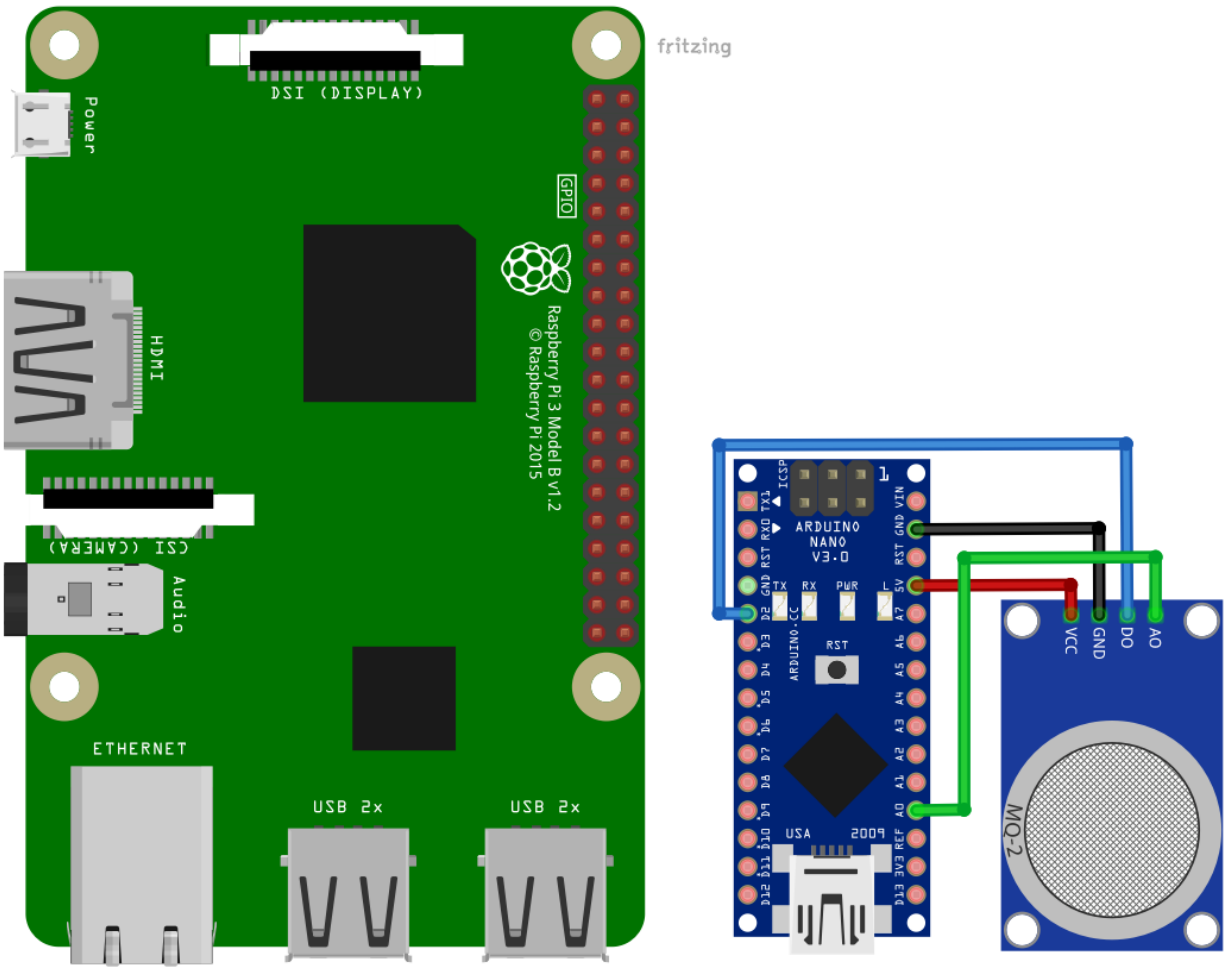
`cp -avr nanpy-firmware/ ~/Arduino/libraries`



The *nanpy-firmware* is now installed and ready to be used.

Connecting the module with Raspberry Pi

Connect the module with the Nano as shown on the following image:



Module pin	Nano pin	Wire color
VCC	5V	**Red wire**
GND	GND	**Black wire**
D0	D2	**Blue wire**
A0	A0	**Green wire**

Next, connect the Nano via USB cable to the Raspberry Pi and open the Arduino IDE in the Raspbian operating system. Check if Arduino IDE can detect the USB port on which the Nano is connected: *Tools* > *Port* > *dev/ttyUSB0*

Then, go to: *Tools* > *Board* > *{board name}*
and select *Arduino Nano* board.
After that, to open a sketch for the `nanpy-firmware`, go to:
File > *Examples* > *nanpy-firmware* > *Nanpy*

Upload the sketch to the Nano. To test if everything works properly, the simple `Blink` script has to be created, where the on-board LED of the Nano is used to blink.

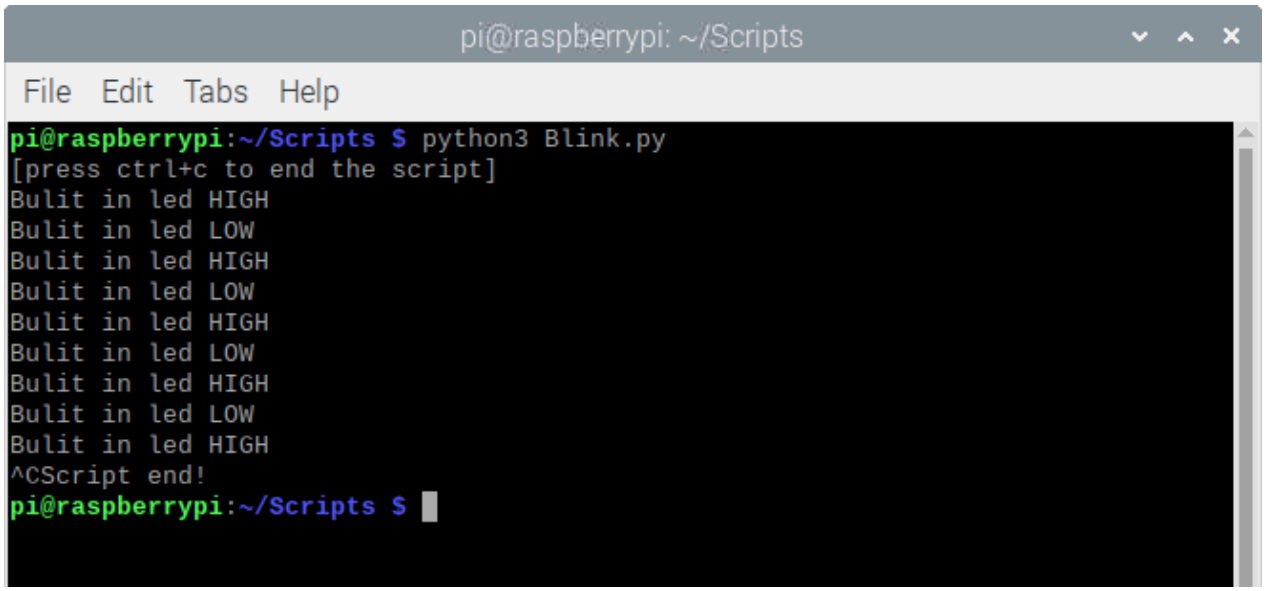
Create the `Blink.py` script, and open it in the default text editor.

In the *Blink.py* script write the following lines of code:

```
from nanpy import (ArduinoApi, SerialManager)
from time import sleep
ledPin = 13
try:
    connection1 = SerialManager()
    a = ArduinoApi(connection=connection1)
except:
  print('Failed to connect to the Arduino')
print('[Press CTRL + C to end the script!]')
a.pinMode(ledPin, a.OUTPUT) # Setup Arduino
try:
   while True:
    a.digitalWrite(ledPin, a.HIGH)
    print('Bulit in led HIGH')
    sleep(1)
    a.digitalWrite(ledPin, a.LOW)
    print('Bulit in led LOW')
    sleep(1)

except KeyboardInterrupt:
  print('\nScript end!')
  a.digitalWrite(ledPin, a.LOW)
```

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

The result should look like as on the following image:



To stop the script press ‘CTRL + C’ on the keyboard.

The LED connected to the digital pin *13* of the Nano should start blinking every second.

The script starts with importing two libraries, the *nanpy* library functions, and the *time*.

Then, the variable called *ledPin* is created and initialized with number *13*. The number *13* represents the number of the digital pin on which LED is connected (on-board LED of the Nano).

After that, the *try-except* block of code is used to try and connect to the Nano. If connection is not successful, message:
Failed to connect to the Arduino is displayed in the terminal.

If connection is successful, a communication object called "*a*" is created and initialized. The object "*a*" represents the Nano board. Any function used in the Arduino IDE can be used with the "*a*" object, as it can be seen in the code.

With the following line of code, the pin mode is set-up for the digital pin *13*:
`a.pinMode(ledPin, a.OUTPUT)`

Then, in the indefinite loop block (*while True:*) the function *digitalWrite()* is used to set the state of the digital pin *13* (HIGH or LOW state). With the *digitalWrite()* function the LED connected to the pin *13* can be turned ON or OFF.

In the indefinite loop block, the LED is first turned *ON* for a second, and then turned *OFF* for a second. This is called *`blinking the LED`*. The time interval of a single blink can be changed in the following line of code: *`sleep(1)`*
Where number *1* represents the number of seconds for the duration of the time interval.

To end the indefinite loop, press ‘CTRL + C’ on the keyboard. This is called the keyboard interrupt, which is set in the *`except`* block (*`except KyeboardInterrupt`*). In the *`except`* block the on-board LED is turned OFF.

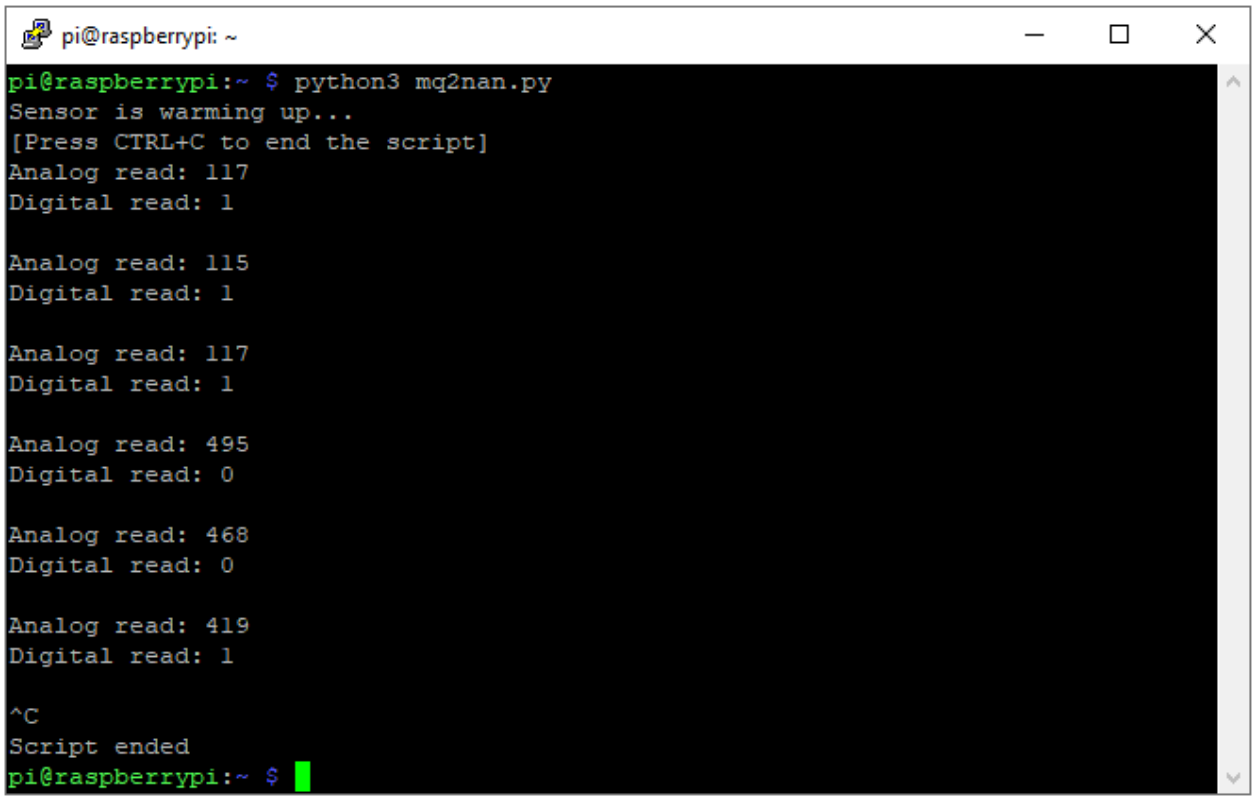
Python script for MQ-2 module

```
from nanpy import (ArduinoApi, SerialManager)
import time


try:
    connection_1 = SerialManager()
    a = ArduinoApi(connection=connection_1)
except:
        print('Failed to connect to the Arduino')
DIGITAL_PIN = 2
ANALOG_PIN = 0
time.sleep(2)
print('Sensor is warming up...')
print('[Press CTRL+C to end the script]')
time.sleep(5) # Sensor warming up...


try:
    while True:
        analogReading = a.analogRead(ANALOG_PIN)
        digitalReading = a.digitalRead(DIGITAL_PIN)
            print("Analog read: {}\nDigital read: {}\n".format(analogReading,
digitalReading))
        time.sleep(2)
except KeyboardInterrupt:
    print('\nScript end!')
```

Save the script by the name *mq2nan.py*. To run the script, open the terminal in the directory where the script is saved and run the following command:

python3 mq2nan.py

The result should look like as on the following image:



To stop 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