Welcome!

Thank you for purchasing our *AZ-Delivery MQ-135 Gas Sensor Module*. On the following pages, you will be introduced to how to use and set up this handy device.

Have fun!

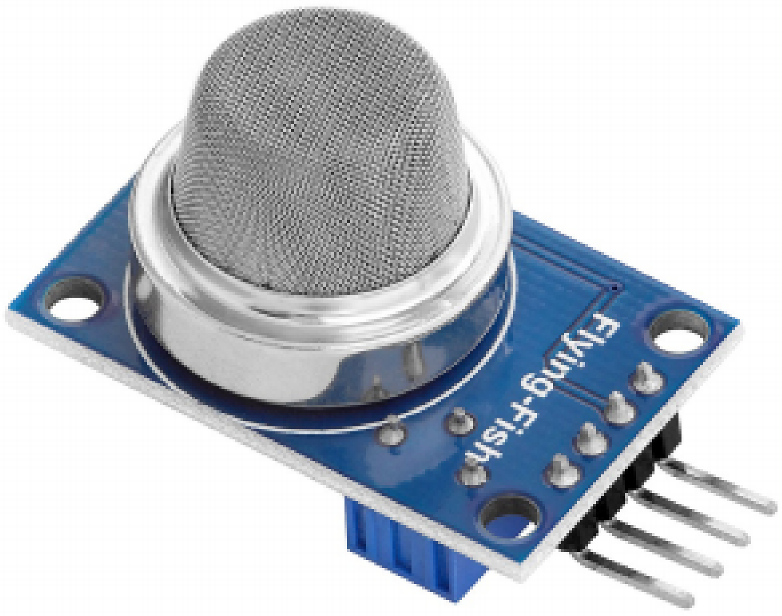


Table of Contents

Introduction **2**

Introduction

The MQ-135 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: ammonia, sulfide, LPG, propane, methane, hydrogen, alcohol, smoke and carbon monoxide and other harmful gasses. Though it can detect those gases, it is not able to distinguish the difference between them.

It is moslty used as the domestic, industrial and portable air pollution detector.

The MQ-135 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

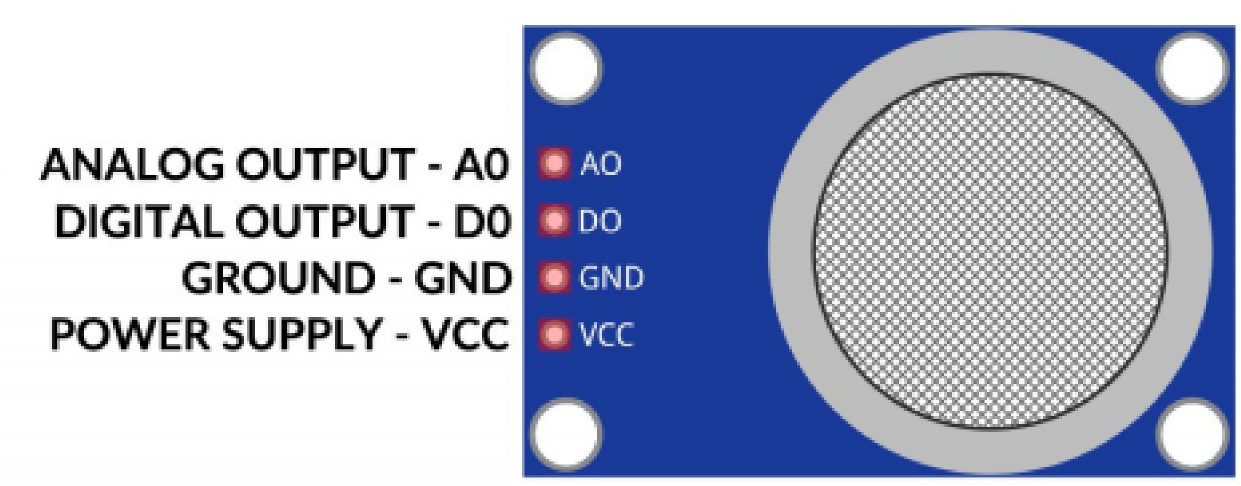
Operating voltage:	5V
Operating current:	150mA
Power consumption:	900mW
Load resistance:	20kΩ
Heater resistance:	31Ω+5%
Sensing resistance	2kΩ - 20kΩ
Preheat time:	24h
Concentration scope:	200 – 10000ppm (parts per million)
Output:	analog, digital
Dimensions:	33x21x22mm (1.3x0.8x0.9in)

For the best detecting results, gas sensor has to be preheated. The best preheat time for the sensor is above 48 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

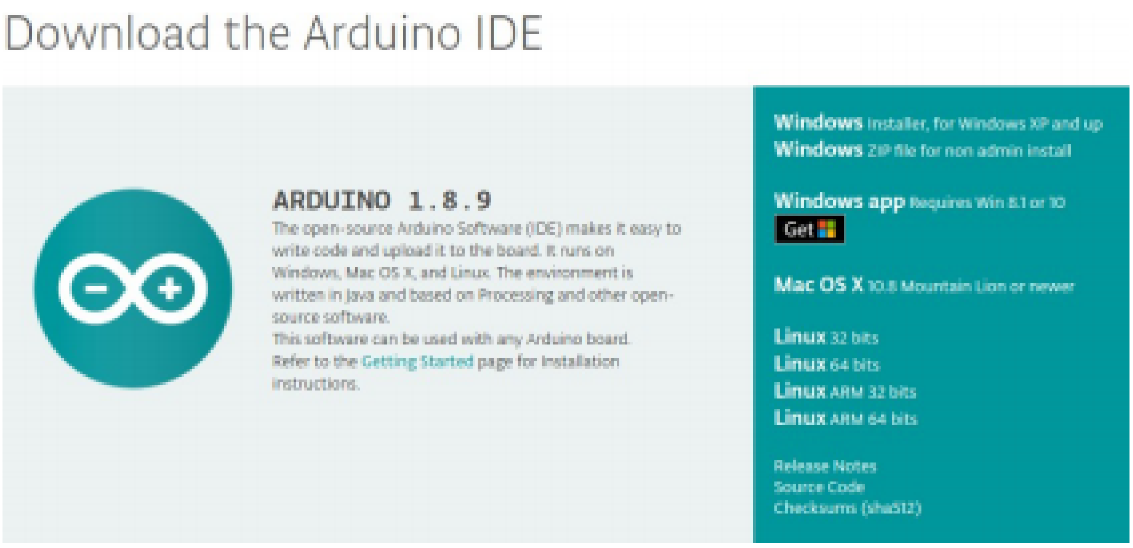
The gas sensor module has four pins. The pinout is shown on the following image:



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.


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*.

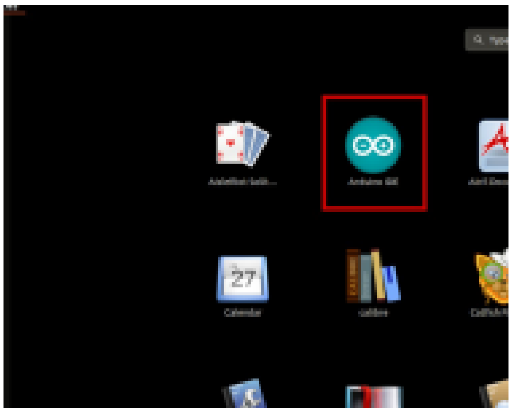
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

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.

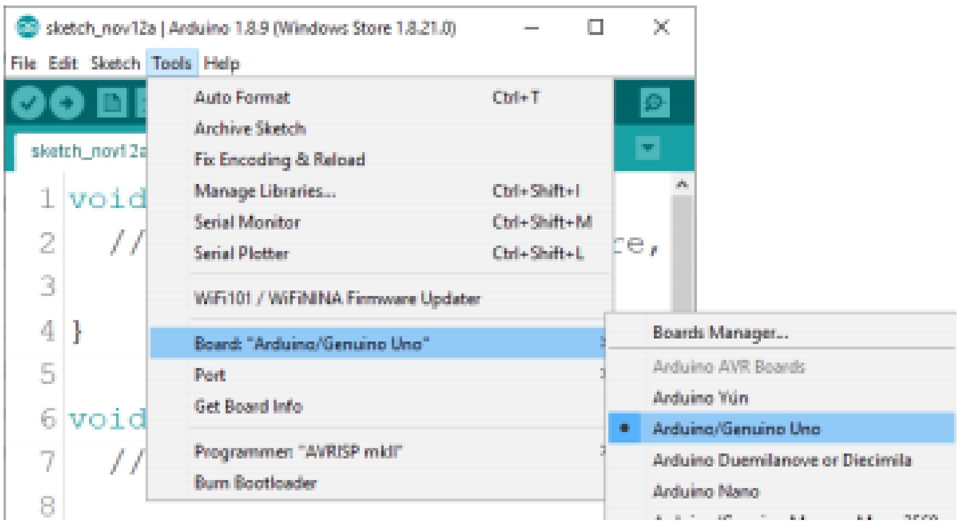
The second script, called *install.sh*, has to be used after the installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

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.

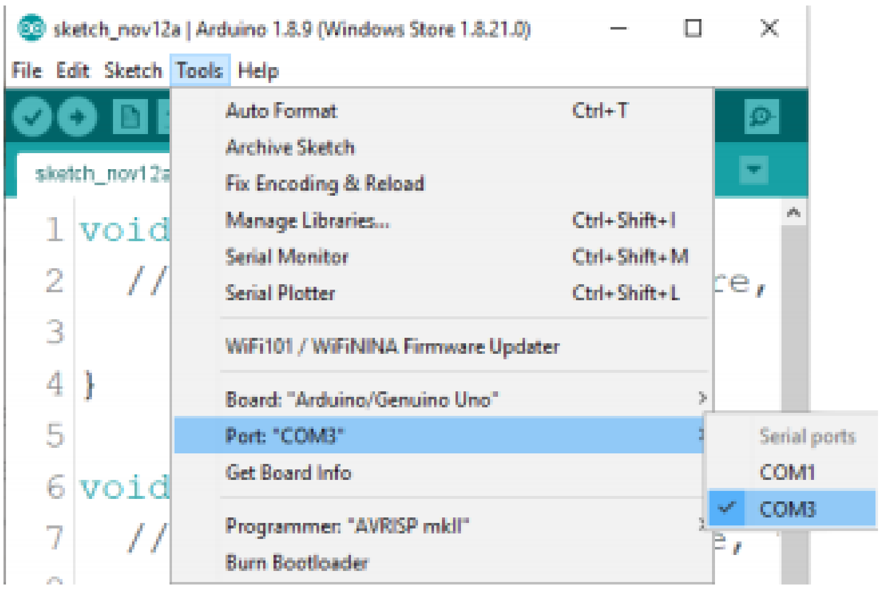
Next thing is to check if your PC can detect an Arduino board. Open freshly installed Arduino IDE, and go to:
Tools > Board > {your board name here}
{your board name here} should be the *Arduino/Genuino Uno*, as it can be seen on the following image:


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
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
void
//
}
void
//
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*.

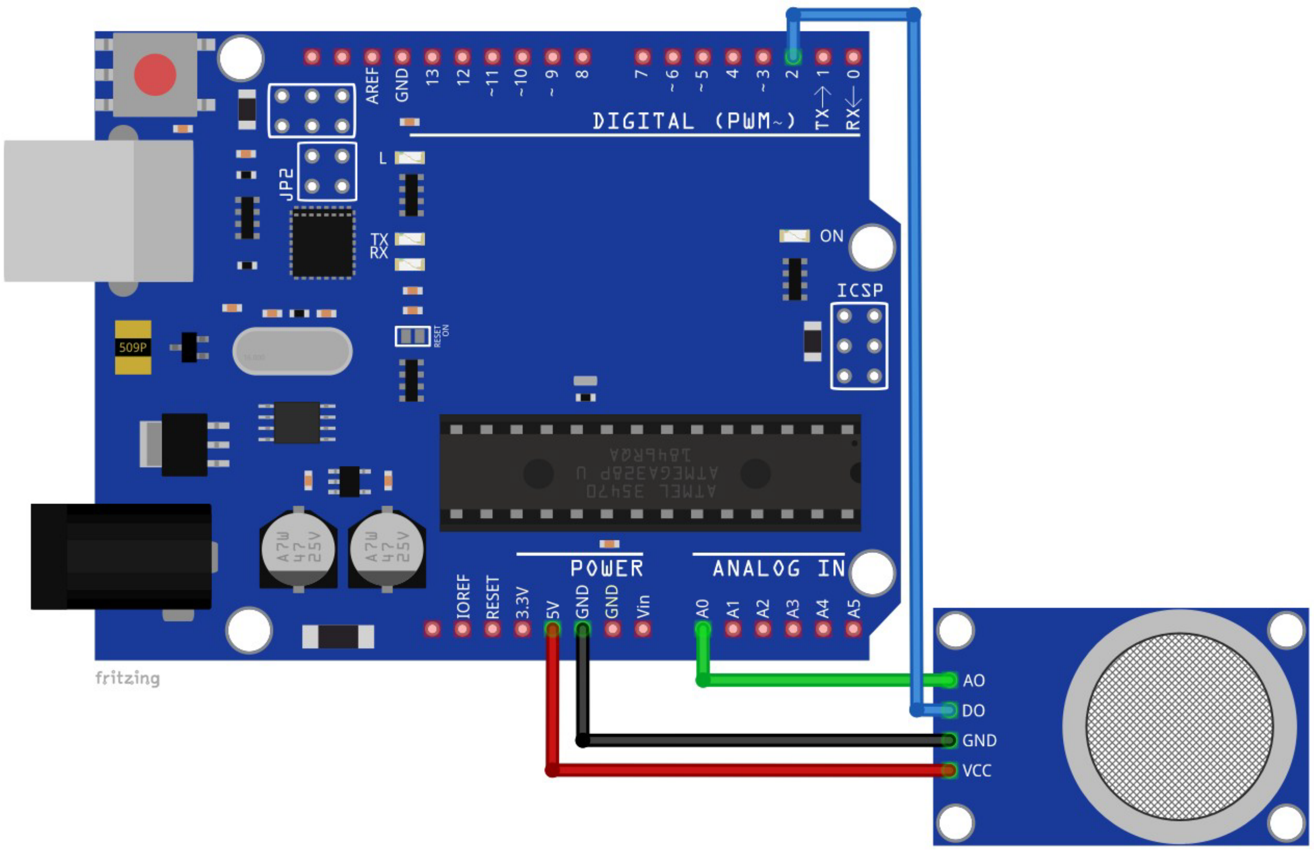
How to set-up the Raspberry Pi and Python

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 the microcontroller

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

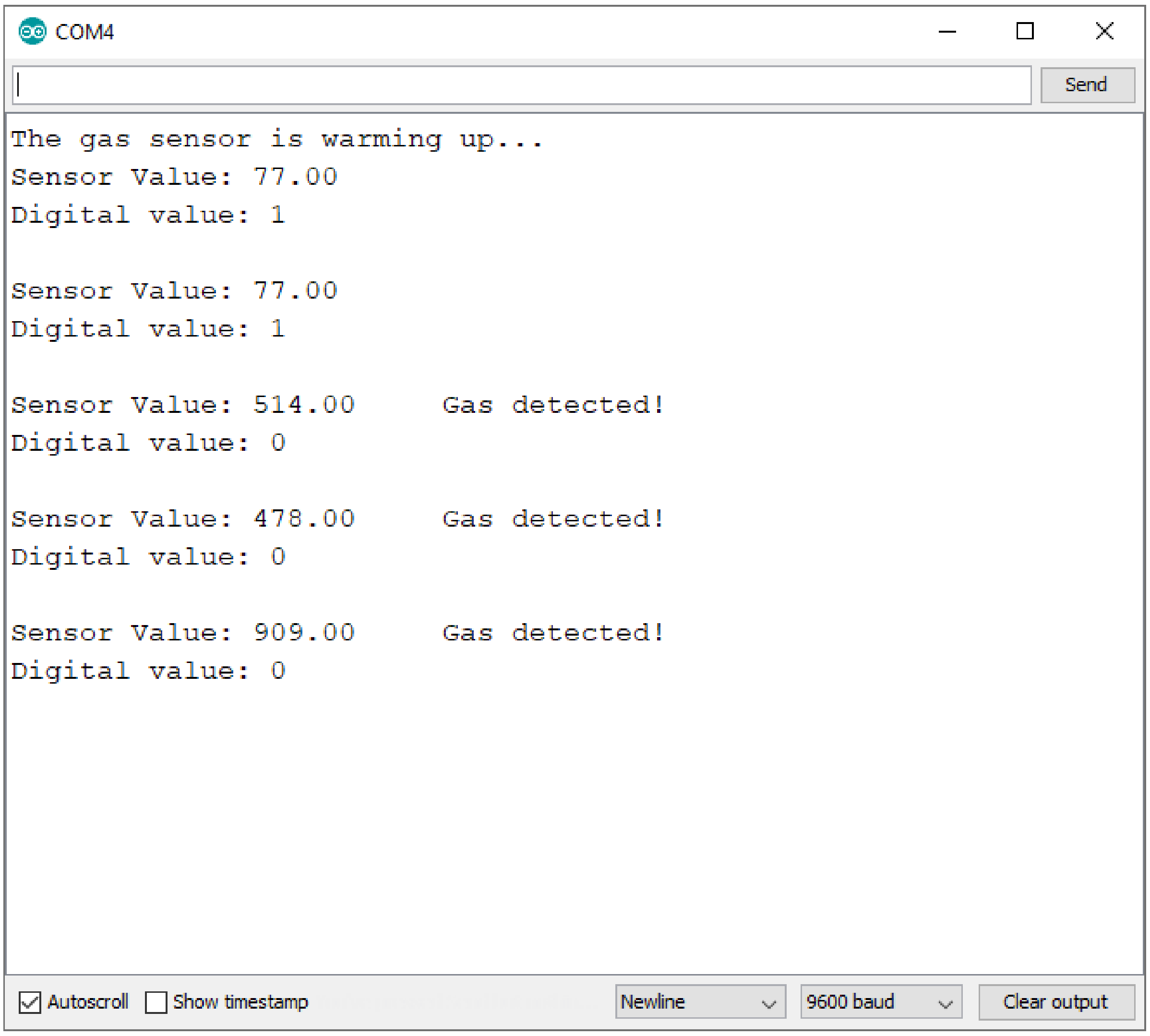


Module pin	**MC pin**	**Wire color**
VCC	5V	**Red wire**
GND	GND	**Black wire**
D0	D2	**Blue wire**
A0	A0	**Green wire**

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



```
The gas sensor is warming up...
Sensor Value: 77.00
Digital value: 1

Sensor Value: 77.00
Digital value: 1

Sensor Value: 514.00     Gas detected!
Digital value: 0

Sensor Value: 478.00     Gas detected!
Digital value: 0

Sensor Value: 909.00     Gas detected!
Digital value: 0
```

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

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

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

The module data can be read in two ways. 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);

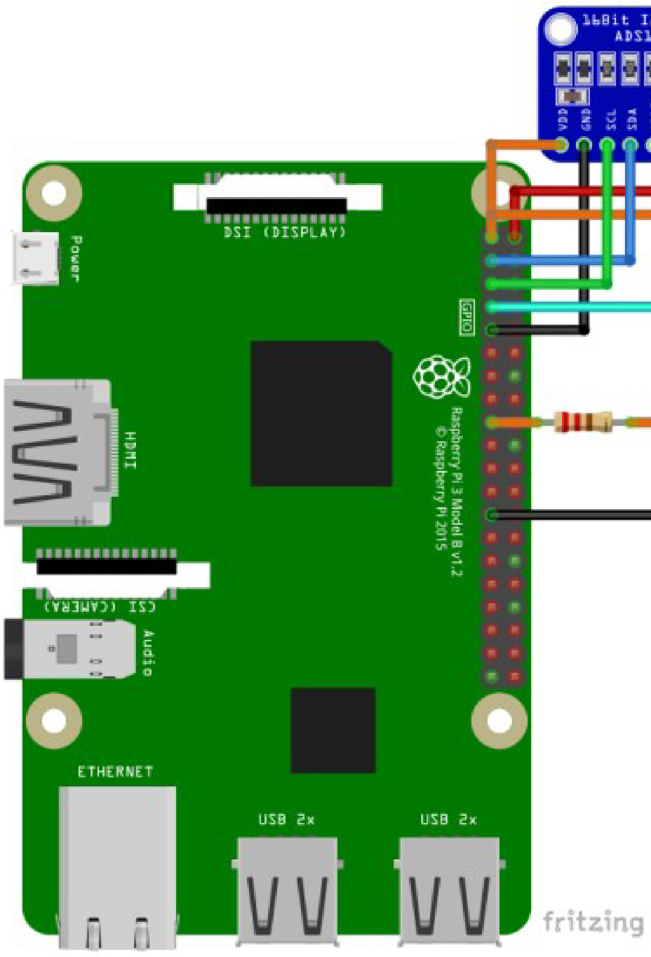
Raspberry Pi and the ADC

The Raspberry Pi cannot read analog inputs, and because of that, we will need to use an *analog-to-digital converter* (ADC). An ADC is a device that, as its name implies, converts analog voltages into digital values. We will be using a device called "*ADS1115*", which uses I2C bus to send data to microcontroller. So first we will have to enable I2C on the Raspberry Pi.

Likewise, the Raspberry Pi's GPIOs cannot handle the 5V voltages, so we need to use a logic level converter. A logic level converter is a device that takes a 5V signals and converts them to a 3.3V signals, and vice versa. Most logic level converters are bi-directional, which means that they can also convert 3.3V signals back to 5V. We will be using a device called "*TXS0108E*", which is one of those bi-directional converters.

Connecting the MQ-135 with the Raspberry Pi

Connect everything as shown on the connection diagram below:



Raspberry pin	**>**	**ADS1115 pin**	
3,3V [pin 1]	>	VDD	**Orange Wire**
GPIO2 [pin 3]	>	SDA	**Blue Wire**
GPIO3 [pin 5]	>	SCL	**Green Wire**
GND [pin 9]	>	GND	**Black Wire**

Raspberry pin	**>**	**TXS0108E pin**	
3,3V [pin 1]	>	VA	**Orange Wire**
5V [pin 2]	>	VB	**Red Wire**
GPIO4 [pin 7]	>	A4	**Cyan Wire**
3,3V [pin 17]	>	OE	**Orange Wire**
GND [pin 25]	>	GND	**Black Wire**

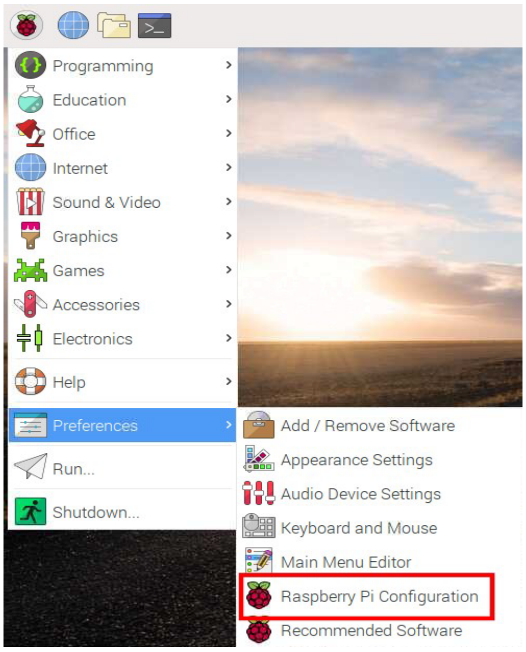
Connect a pull up resistor 10kΩ between 3,3V (pin 17) and OE.

ADS1115 pin	**>**	**TXS0108E pin**	
A0	>	A1	**Purple Wire**

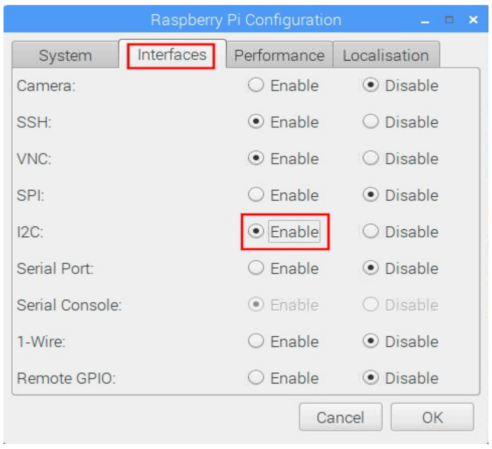
TXS0108E pin	**>**	**MQ-135 pin**	
VB	>	VCC	**Red Wire**
B1	>	A0	**Purple Wire**
B4	>	D0	**Blue Wire**
GND	>	GND	**Black Wire**

Enabling the I2C interface

To enable the I2C interface on Raspberry Pi, in your Raspbian, go to: *Start > Preferences > Raspberry Pi Configuration*.



This will open up a new window, go to its second tab, "*Interfaces*", and enable the I2C radio button. Once you've done so, press the "*ok*" button like shown on the image below:


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

Doing so enables the I2C interface on the *"GPIO2"* and *"GPIO3"* pins. Next, we have to install the library for the ADC device. The library is called "*Adafruit_Python_ADS1x15*". To do so, open the terminal app in your Raspbian and run these commands one by one:

```
sudo apt-get update
sudo apt-get install build-essential python-dev python-smbus git git clone
https://github.com/adafruit/Adafruit_Python_ADS1x15 cd
Adafruit_Python_ADS1x15
sudo python3 setup.py install
```

Now, we are ready to move onto the python code.

Python code

Create a new file called *"mq135.py",* and insert the following python code:

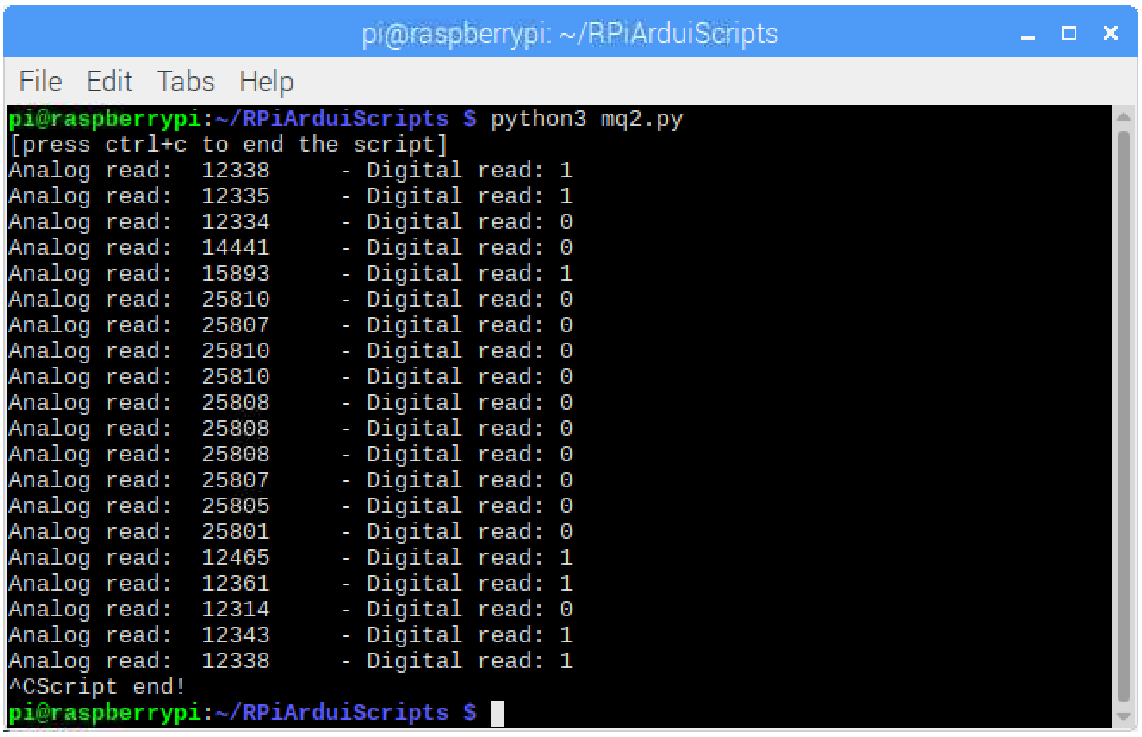
```
import time
import Adafruit_ADS1x15
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
digitalPin = 4
GPIO.setup(digitalPin, GPIO.IN)
adc = Adafruit_ADS1x15.ADS1115()
GAIN = 1
print("[press ctrl+c to end the script]")
try: # Main program loop
	while True:
		analogReading = adc.read_adc(0, gain=GAIN)
		digitalReading = GPIO.input(digitalPin)
						print("Analog read: {:>6}\t- Digital read: {}"
						.format(analogReading, digitalReading))
		time.sleep(0.5)

# Scavenging work after the end of the program
except KeyboardInterrupt:
	print("Script end!")
```

At the begining of the script, we import the appropriate libraries. Next, we set up our variables and turn off the warnings. The loop is set up to do both analog and digital readings and then output them. The analog values are shown as six-digit numbers. Note that in the case of zero being the first digit, zeros do not get displayed. And finally, we configure the keyboard interrupt so that we can stop the script at any point by pressing *"CTRL + C"*.

After the previous steps are done, run the script with the following command:

python3 mq135.py

And the output should look similar to this:



```
pi@raspberrypi:~/RPiArduiScripts $ python3 mq2.py
[press ctrl+c to end the script]
Analog read:  12338     - Digital read: 1
Analog read:  12335     - Digital read: 1
Analog read:  12334     - Digital read: 0
Analog read:  14441     - Digital read: 0
Analog read:  15893     - Digital read: 1
Analog read:  25810     - Digital read: 0
Analog read:  25807     - Digital read: 0
Analog read:  25810     - Digital read: 0
Analog read:  25810     - Digital read: 0
Analog read:  25808     - Digital read: 0
Analog read:  25808     - Digital read: 0
Analog read:  25808     - Digital read: 0
Analog read:  25807     - Digital read: 0
Analog read:  25805     - Digital read: 0
Analog read:  25801     - Digital read: 0
Analog read:  12465     - Digital read: 1
Analog read:  12361     - Digital read: 1
Analog read:  12314     - Digital read: 0
Analog read:  12343     - Digital read: 1
Analog read:  12338     - Digital read: 1
^CScript end!
pi@raspberrypi:~/RPiArduiScripts $
```

As you can see on the image, we read the sensor state every half a second. The sensor is detecting natural gases found in the air, so the default analog values are around 12000. But when exposed to butane gas from a lighter, the digital readings turn to mostly zeros and the analog values spike from the 12000 up to the 25000. When digital values are mostly zeros, this means that a gas concentration is high, while the analog values indicate the gas concentration value.

You've done it!
You can now use your module for various projects.

Now it is 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 you can find on the internet.

If you are looking for the high quality microelectronics and accessories, 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