Welcome!

Thank you for purchasing our *AZ-Delivery GY-271 Compass Magnetic Module*. On the following pages, you will be introduced to how to use and set-up this handy device.

Have fun!

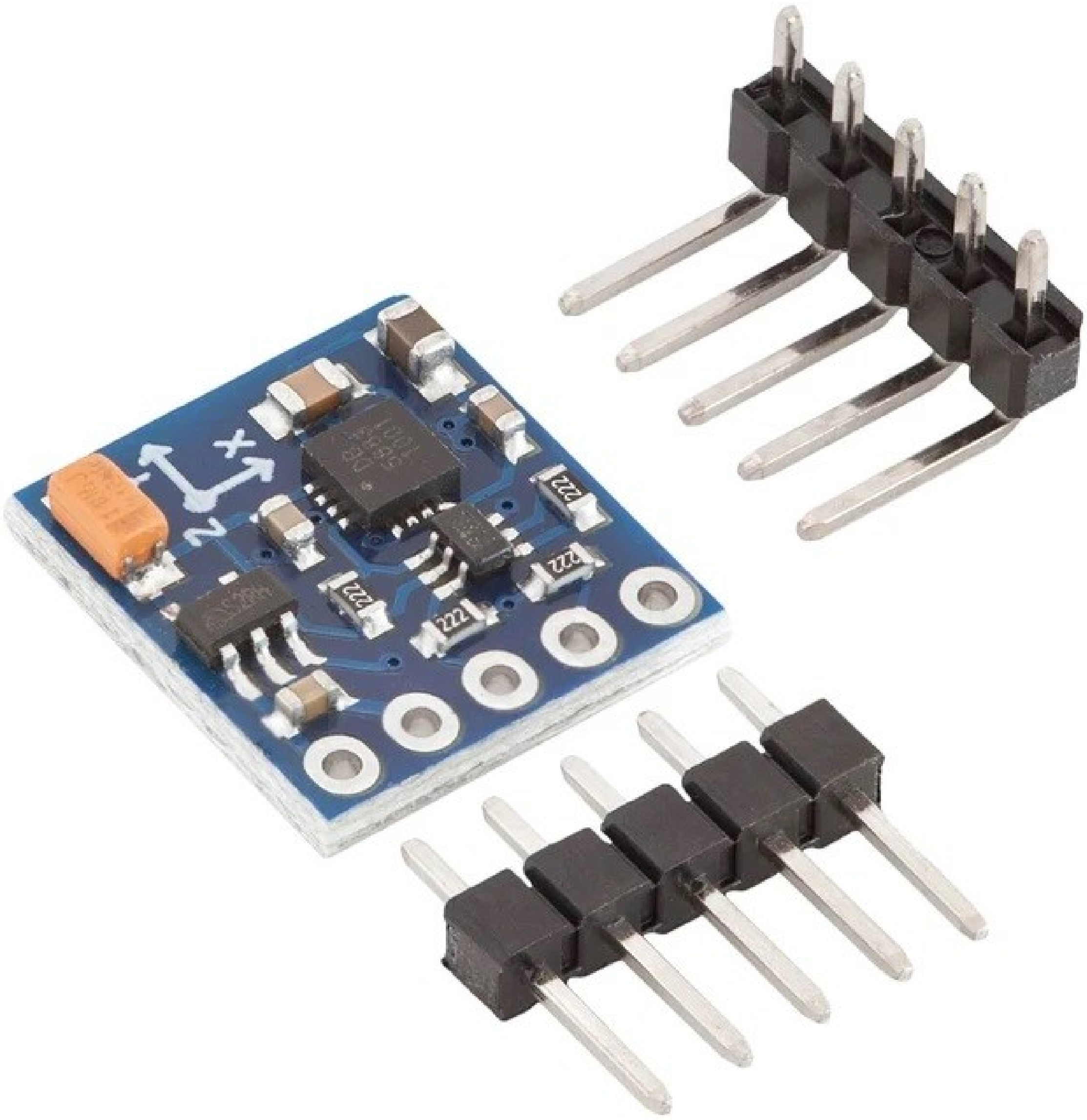


Table of Contents

Introduction

The heart of a GY-271 module is the `QMC5883L` chip. The module is targeted for high precision applications such as compassing, navigation and gaming in drone, robotic, mobile and personal hand-held devices. The chip is actually a multi-chip, magnetoresistive sensor circuit which consists of tri-axial sensors and application specific integrated circuits (ASIC) to measure magnetic fields. It is based on high resolution, magnetoresistive technology licensed from *Honeywell AMR* technology. Along with custom-designed *16* bit analog to digital converter (ADC), it offers the advantages of low noise, high accuracy, low power consumption, offset cancellation and temperature compensation. The sensor enables *1°* to *2°* compass heading accuracy.

With a DC power supply applied to the sensor terminals, the sensor converts any incident magnetic field in the sensitive axis directions to a differential voltage output. The ASIC then amplifies and processes the signal to have a digital output. The device has an offset cancellation function to eliminate sensor and ASIC offsets. It also applies a self-aligned magnetic field to restore the magnetic state before each measurement to ensure high accuracy. Because of these features, the `QMC5883L` does not need calibration on every run in most application situations. It may need to be calibrated once in a new system or the system changes to a new battery.

This device is connected to a serial interface, but as a slave device under the control of a master device, such as the microcontroller. Control of this device is carried out via the I2C interface. This device supports standard and fast speed modes, *100kHz* and *400kHz*, respectively. External pull-up resistors are required to support all these modes.

Specifications

»	Operating voltage range:	from 3.3V to 5V DC
»	Low Power Consumption:	75μA
»	Communication interface:	I2C (standard and fast modes)
»	Default I2C address:	0x0D
»	Compass accuracy:	1° to 2°
»	Operating temperature range:	from -40°C to +85°C
»	Magnetic Field Range:	±8 Gauss
»	Analog to digital resolution:	16-bit
»	ADC With Low Noise AMR Sensors	
»	Temperature Output	
»	Temperature Compensated Data Output	
»	Dimensions:	13 x 14 x 3mm [0.51 x 0.55 x 0.12in]

The device has an internal clock for internal digital logic functions and timing management. This clock is not available for external usage.

Some periods of time are required for the device to be fully functional after the device is powered on. The external power supply requires a time period for voltage to ramp up, which is typically *50* milliseconds. However, it is not controlled by the device. The Power On Reset (POR) time period includes time to reset all the logics, load values in non-volatile memory to proper registers, enter the stand-by mode and get ready for measurements (around *450* microseconds).

When the device is powered on, all registers are reset by POR, then the device transits to the stand-by mode and waits for further commands.

The *`QMC5883L`* chip also has a built-in temperature sensor that can provide temperature reading for other applications.

Temperature compensation of the measured magnetic data is enabled by default at the factory. Temperature measured by the built-in temperature sensor will be used to compensate for the magnetic sensitivity changes of the sensor due to temperature change. The compensated magnetic sensor data is placed in the output data registers automatically.

Operation modes

Continuous-Measurement Mode

During the continuous-measurement mode, the magnetic sensor continuously makes measurements and places measured data in the data output registers. In the continuous-measurement mode, the magnetic sensor data is automatically compensated for offset and temperature effects. The gains are calibrated in the factory.

Normal Read Sequence

Normal read complete magnetometer data read-out can be done as follows:

» Check *DRDY* pin (or by polling *DRDY* bit in register *06H*)

» Read *DRDY* bit in register *06H* (for polling it is unnecessary)

DRDY: Data ready ("*1*"), or not ("*0*").

DOR: Some or all data has been missed ("*1*"), or not ("*0*")

» Read measured data; if any of the six data registers is accessed, *DRDY* and *DOR* turn to "*0*".

If any of the six data registers is accessed, data protection starts. During the data protection period, the data register cannot be updated until the last bits of register *05H* have been read.

During the measurement, it is possible to read the data register which keeps the previously measured data. Therefore, no interrupt (*DRDY* bit) will be set if data reading occurs in the middle of measurement.

If *N*-th data is skipped, the current data is flushed by the next coming data. In this case, interrupt (*DRDY* bit) keeps high until data is read. The *DOR* bit is set to "*1*", which indicates a set of measurement data has been lost. It will (*DOR* bit) turn to "*0*" once the register *06H* is accessed in the next data read operation.

Data lock is activated once any of the data registers are accessed. If the register *05H* (data unlock) is not accessed until the next measurements end, current data blocks the next data to update the data register. In this case, the *DOR* bit is also set to "*1*" until the register *06H* is accessed in the next data read.

An interrupt is generated on the *DRDY* pin each time that the magnetic field is measured. The interrupt can be disabled by setting: *0AH[0] = 1*.

Stand-by Mode

The stand-by mode is a default state of the chip, upon Power On Reset (POR) or soft reset. Only a few function blocks are activated in this mode, which keeps power consumption as low as possible. In this state, register values are held *ON* by an ultra-low power. The I2C interface can be woken up by reading register values that are held *ON* or by writing to any register. There is no magnetometer measurement in the stand-by mode. Internal clocking is also halted.

Measurement Example

» Check status register *`06H[0]`*; "*1*" means ready.

» Read data registers from *`00H`* through *`05H`*.

Standby Example

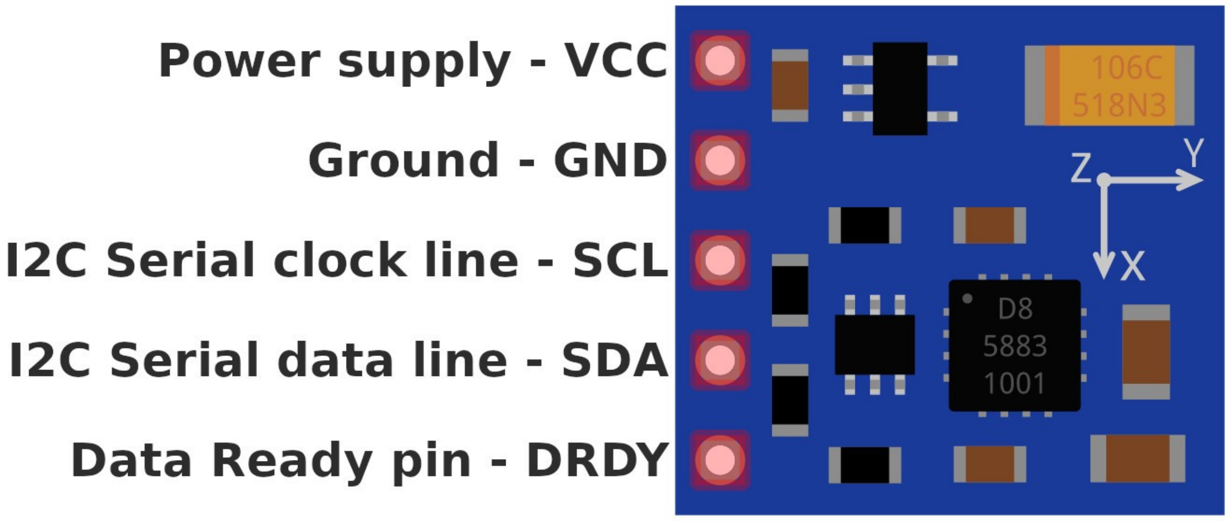
» Write the value *`0x00`* in register *`09H`*.

Soft Reset Example

» Write the value *`0x80`* in register *`0AH`*.

The pinout

The GY-271 compass magnetic module has five pins. The pinout is shown on the following image:



The power supply and logic pins work on voltages in a range from *3.3V* up to *5V*.

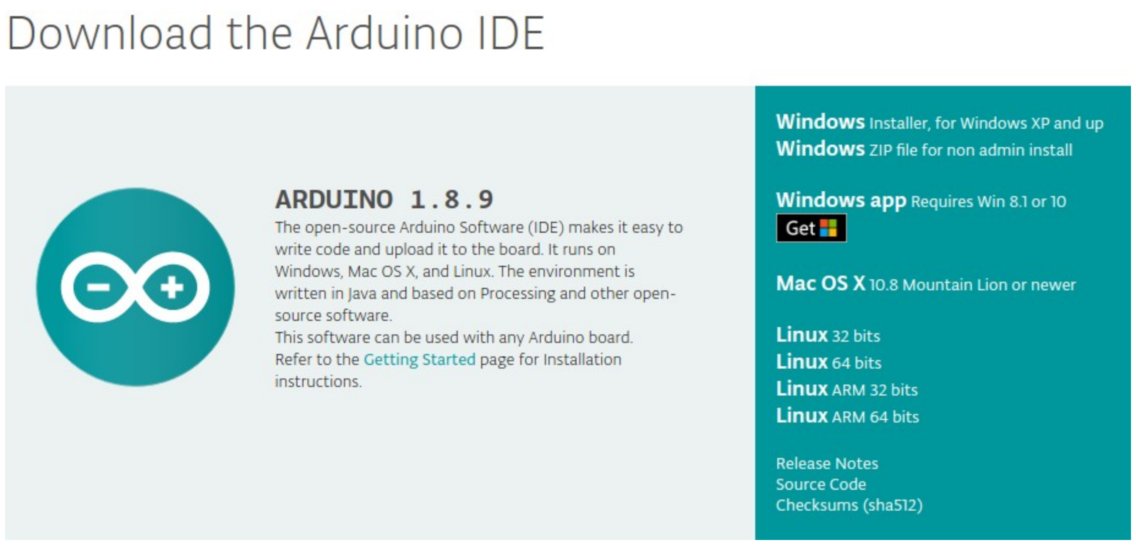
The *DRDY* pin is used to indicate that measurement is finished. You can use it as an interrupt, as a signal to start reading data from the sensor.

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 theinstallation 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*.

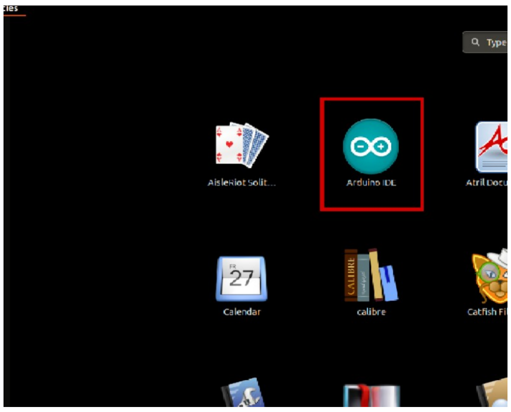
To run the first script in terminal, open terminal in the extracted directory, and run the following command:

sh arduino-linux-setup.sh user_name

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

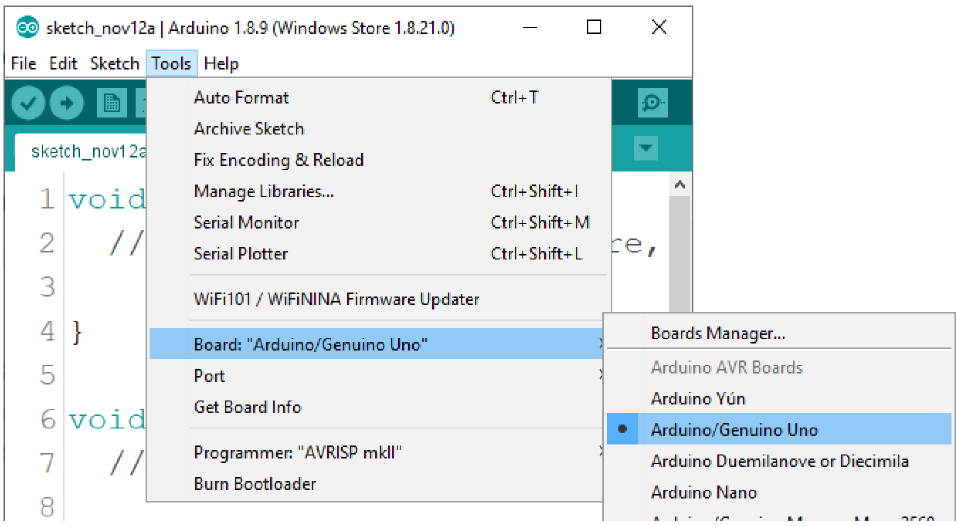
After installation of the first script, the second script, called *install.sh*, has to be used. In the terminal (extracted directory), run the following command: **sh install.sh**

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



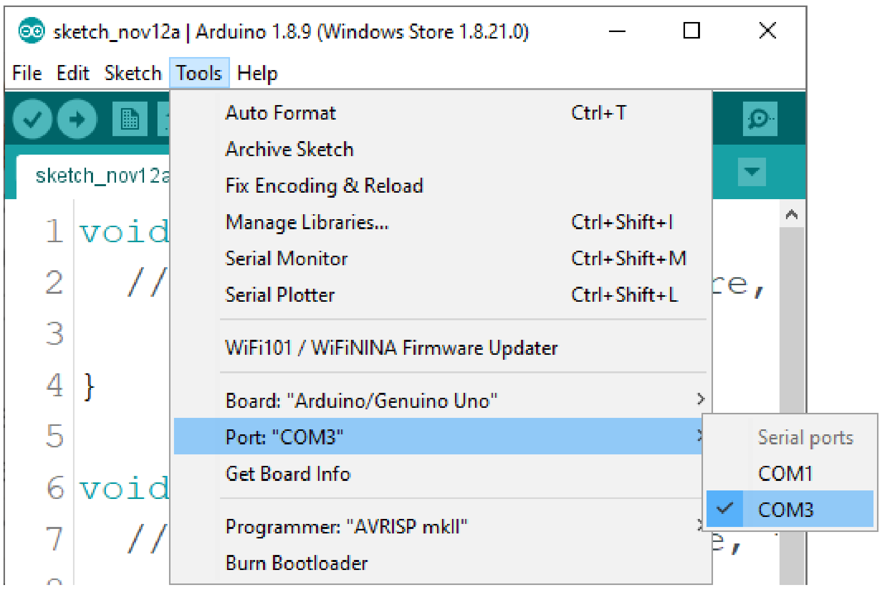
Almost all operating systems come with a text editor preinstalled (for example, `Windows` comes with `Notepad`, `Linux Ubuntu` comes with `Gedit`, `Linux Raspbian` comes with `Leafpad`, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

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 on 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 on 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, then to set-up everything 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 in detail in the free eBook:
Raspberry Pi Quick Startup Guide

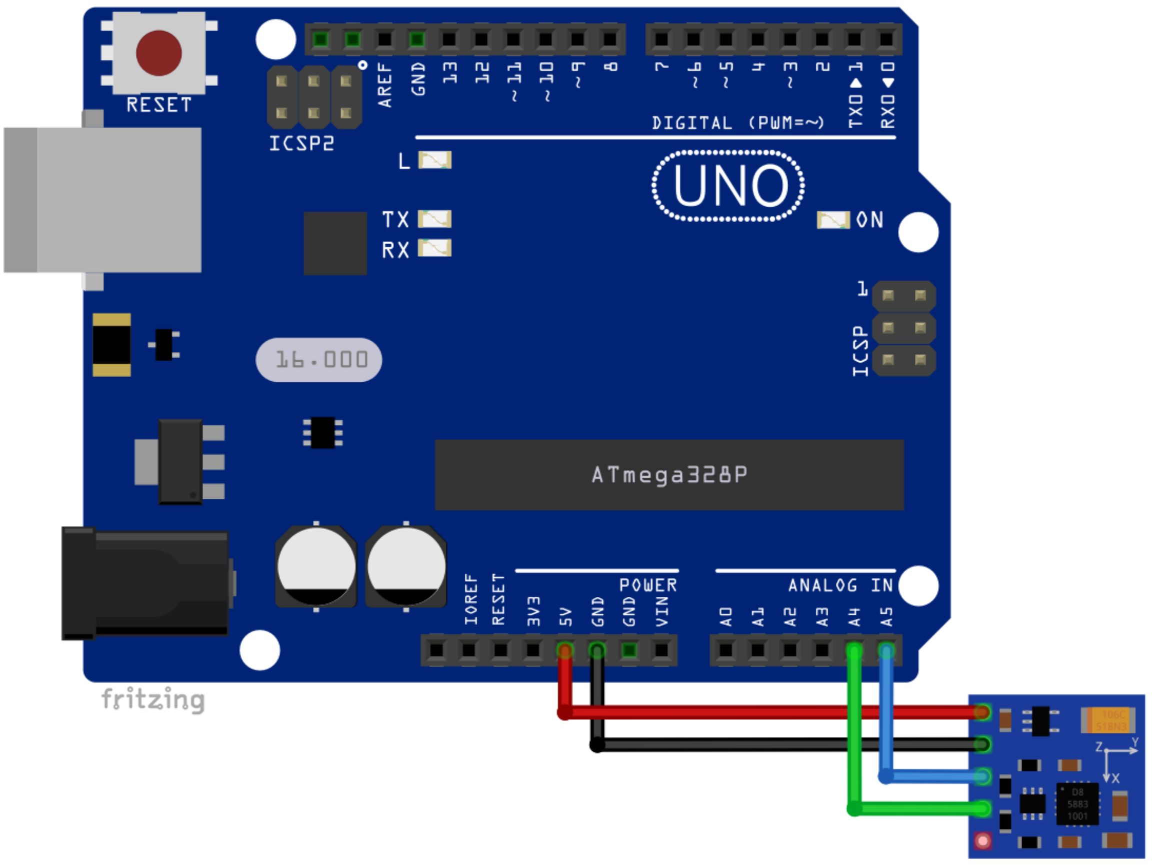
which can be found on the following link:
https://www.az-delivery.de/products/raspberry-pi-kostenfreies-e-book?ls=en

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

AZ-Delivery

Connecting the sensor with Uno

Connect the GY-271 compass magnetic module with the Uno as shown on the following connection diagram:



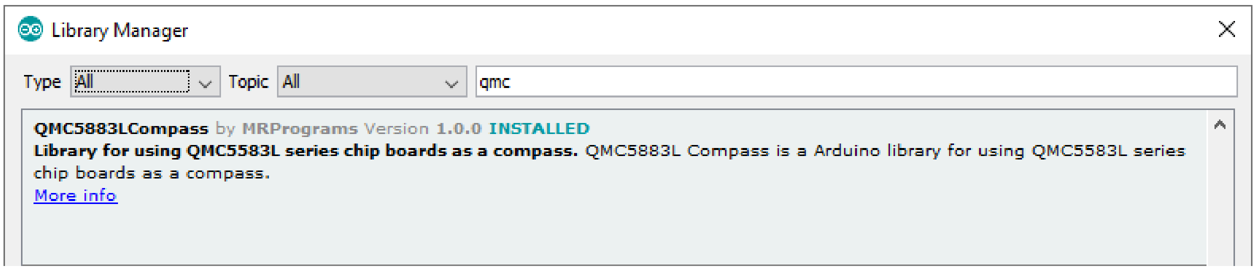
Sensor pin	**>**	**Uno pin**		
VCC	>	5V		Red wire
GND	>	GND		**Black wire**
SCL	>	A5	[SCL]	Blue wire
SDA	>	A4	[SDA]	Green wire
DRDY	>	not connected		

Library for Arduino IDE

To use the sensor with a Uno, it is recommended to download an external library for it. The library that is used in this eBook is called the *`QMC5883L`*. To download and install it, open Arduino IDE and go to:

Tools > Manage Libraries

When a new window opens, type *`qmc`* in the search box and install the library *`QMC5883LCompass`* made by *`MRPrograms`*, as shown on the following image:



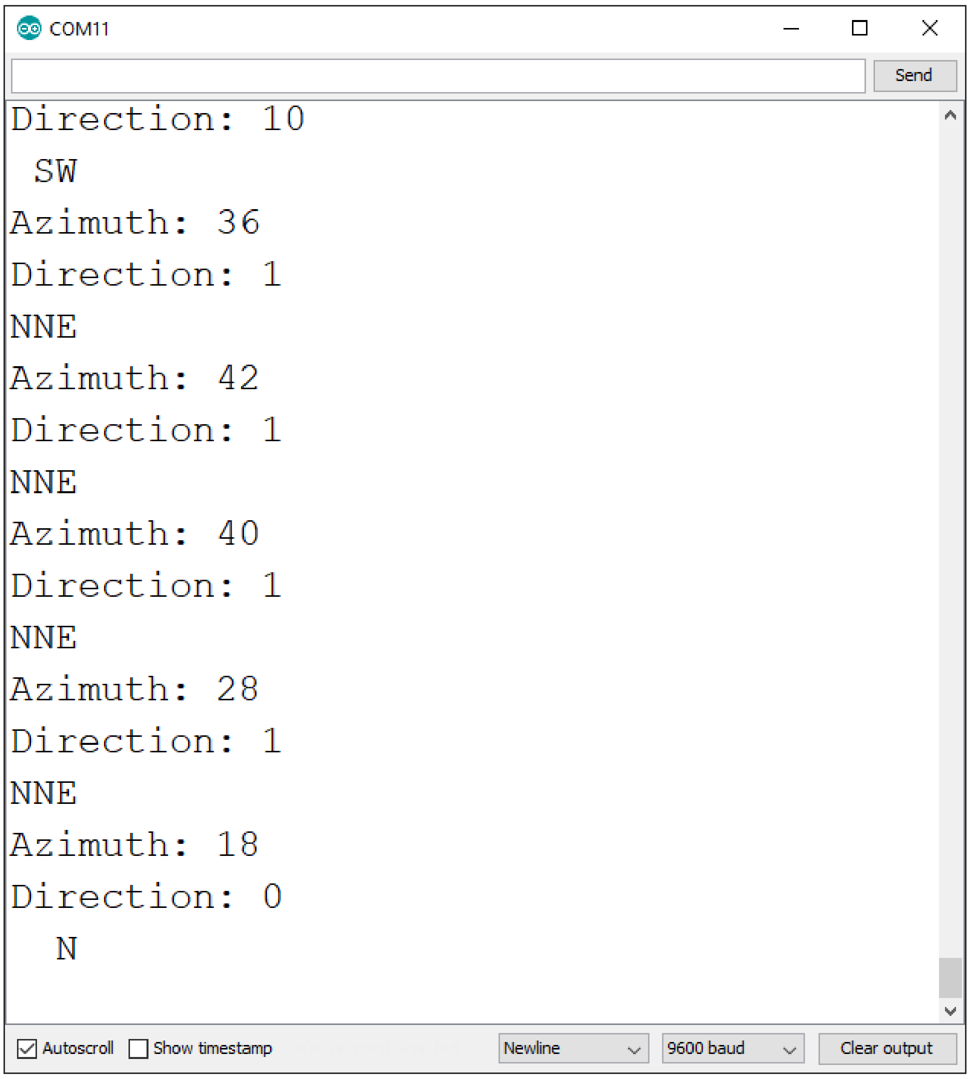
QMC5883LCompass by **MRPrograms** Version **1.0.0** **INSTALLED**
Library for using QMC5583L series chip boards as a compass. QMC5883L Compass is a Arduino library for using QMC5583L series chip boards as a compass.
More info

Several sketch examples come with the library. The code from several sketch examples is modified to create the sketch example in the next chapter in order to show how to use all functions that come with the library.

Sketch example

```
#include <QMC5883LCompass.h>
QMC5883LCompass compass;
void setup() {
  Serial.begin(9600);
  compass.init();
}
void loop() {
  compass.read(); // Read compass values
  byte a = compass.getAzimuth();
  // Return Azimuth reading
  Serial.print("Azimuth: ");
  Serial.println(a);
  byte d = compass.getBearing(a);
  // Output is a value from 0 - 15
  // based on the direction of the bearing / azimuth
  Serial.print("Direction: ");
  Serial.println(d);
  char compassLetters[3];
  compass.getDirection(compassLetters, a);
  Serial.print(compassLetters[0]);
  Serial.print(compassLetters[1]);
  Serial.println(compassLetters[2]);
  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:



```
Direction: 10
 SW
Azimuth: 36
Direction: 1
NNE
Azimuth: 42
Direction: 1
NNE
Azimuth: 40
Direction: 1
NNE
Azimuth: 28
Direction: 1
NNE
Azimuth: 18
Direction: 0
  N
```

To get similar values, rotate the compass board around one of its axis.

The sketch starts with importing a library *`QMC5883LCompass`*.

Next, an object called *`compass`* is created, which represents the module itself. This object is used to initialize and read the data from the modul.

In the *`setup()`* function the serial communication is started with a baud rate of *9600bps*. Also, the *`compass`* object is initialized, with the following line of code: `compass.init();`

At the beginning of the *`loop()`* function the data from the module is read, with the following line of code: `compass.read()`

Next, the readings are converted to Azimuth values (what the Azimuth values are, is not covered in this eBook) with the following line of code:
`byte a = compass.getAzimuth();`

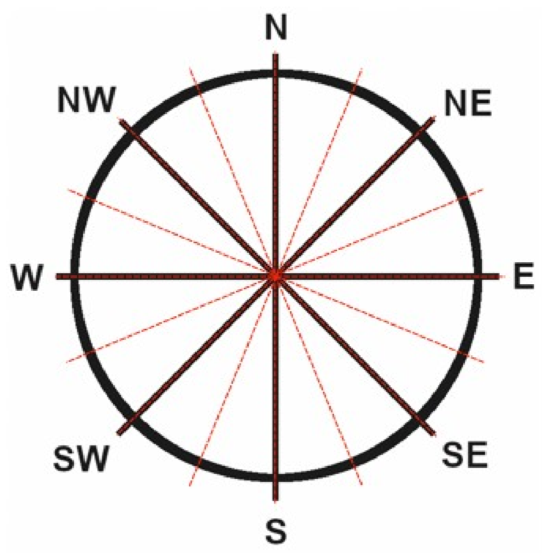
The function called *`getAzimuth()`* has no arguments and returns a value. The return value is an integer number representing the Azimuth value.

After this, the data with Azimuth value is displayed in the Serial Monitor.

Then, the Azimuth values are converted to direction numbers and displayed in the Serial Monitor.

There are eight directions on the compass:
- North
- North-East
- East
- South-East
- South
- South-West
- West
- North-West

Which can be seen on the compass circle, there are *16* different areas, like on the following image:



The function called *`getBearing()`* has one argument and returns a value. The argument is an integer number, which represents the Azimuth value. The return value is also an integer number, in the range from *0* to *15*, which represents one of directions number from the compass circle image.

After this, the direction numbers are converted to direction letters with the following lines of code:

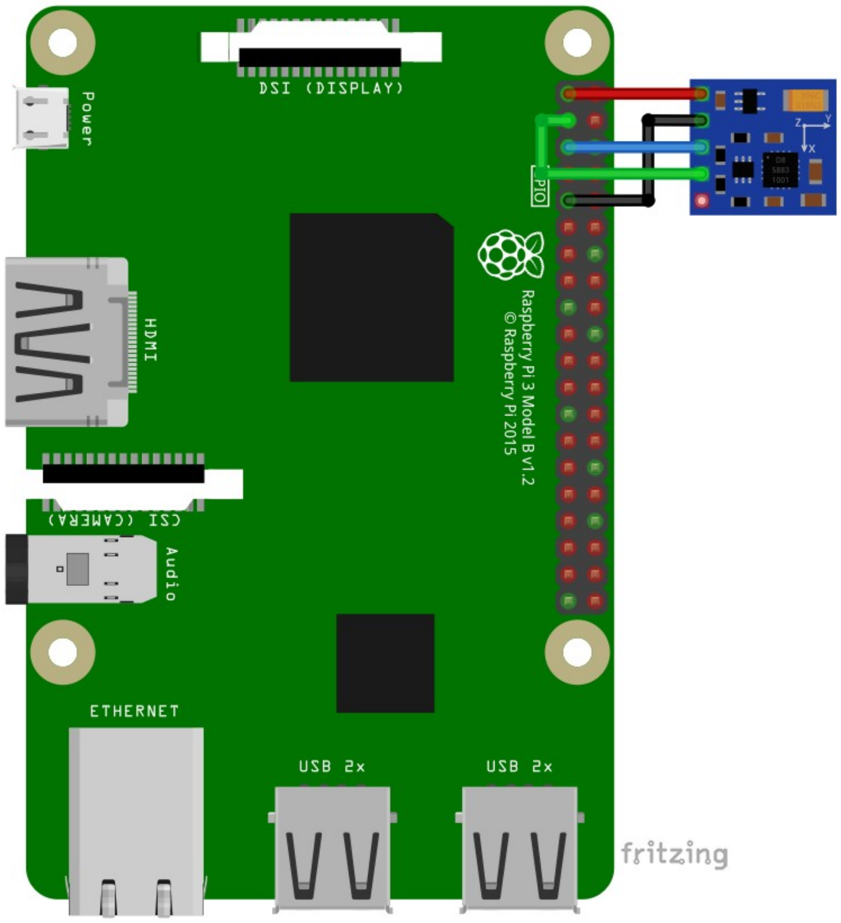
```
char compassLetters[3];
compass.getDirection(compassLetters, a);
```

Where "*a*" is the Azimuth value. Here, a *`char`* array with three elements is created. Then the function *`getDirection()`* is used to convert the Azimuth value into the direction letter. The function has two arguments and returns no value. The first argument is the *`char`* array, in which the function store direction letters, and the second argument is the Azimuth value.

At the end of the *`loop()`* function the direction letters data is displayed and the *`delay()`* function is used to create a pause of *250* milliseconds between two loops of the *`loop()`* function.

Connecting the sensor with Raspberry Pi

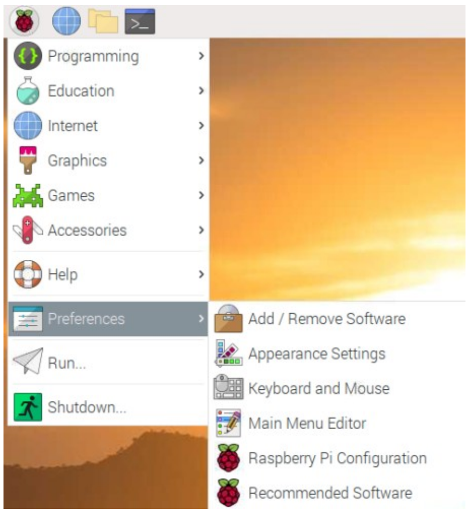
Connect the GY-271 compass magnetic module with the Raspberry Pi as shown on the following connection diagram:



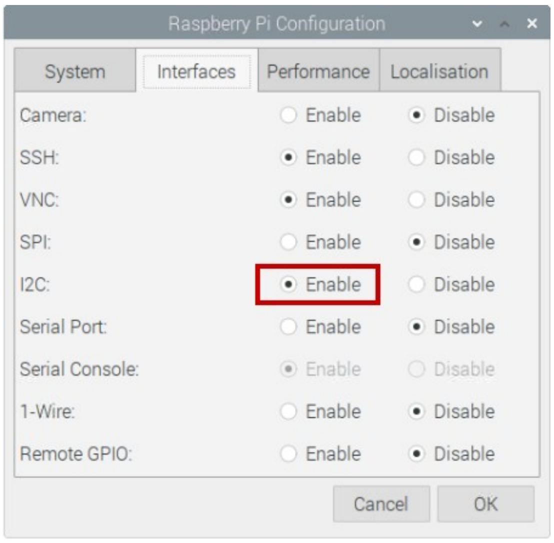
Sensor pin	**>**	**Raspberry Pi pin**		
VCC	>	3V3	[pin 1]	Red wire
GND	>	GND	[pin 9]	**Black wire**
SCL	>	GPIO3	[pin 5]	Blue wire
SDA	>	GPIO2	[pin 3]	Green wire

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 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]

Libraries and tools for Python

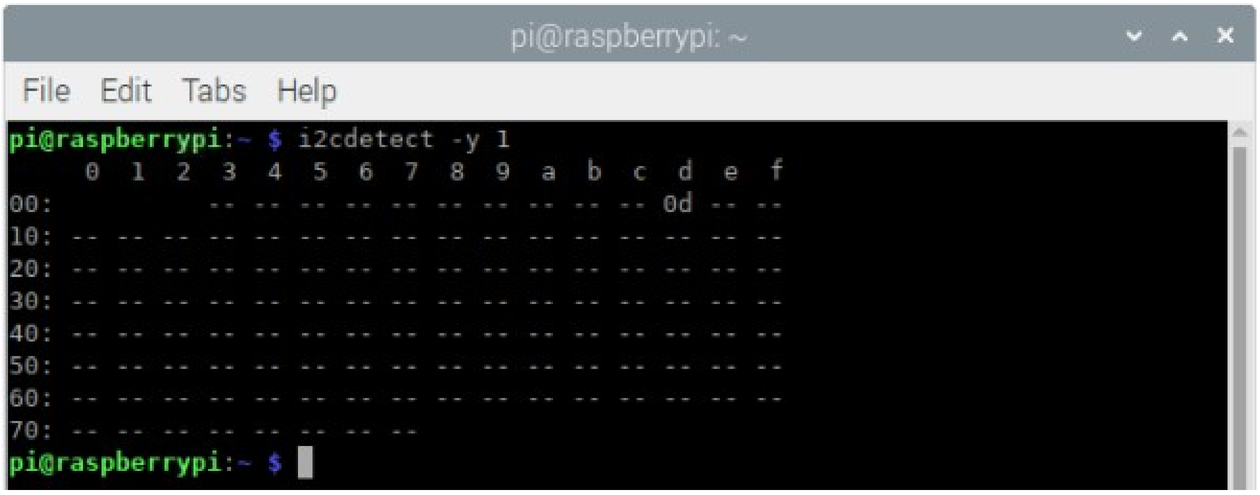
In order to use the module with the Raspberry Pi it is recommended to download and install an external library for it. The library that is used in this eBook is called *gy271compass*. Before it is used, several libraries and tools have to be installed.

First, install the *git*, *python-smbus* and *i2c-tools*. To do so open the terminal and run the following commands, one by one:

```
sudo apt-get update
sudo apt-get install -y git python3-smbus i2c-tools
```

Next, check the I2C address of the module, by running the following command: **`i2c detect -y 1`**

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

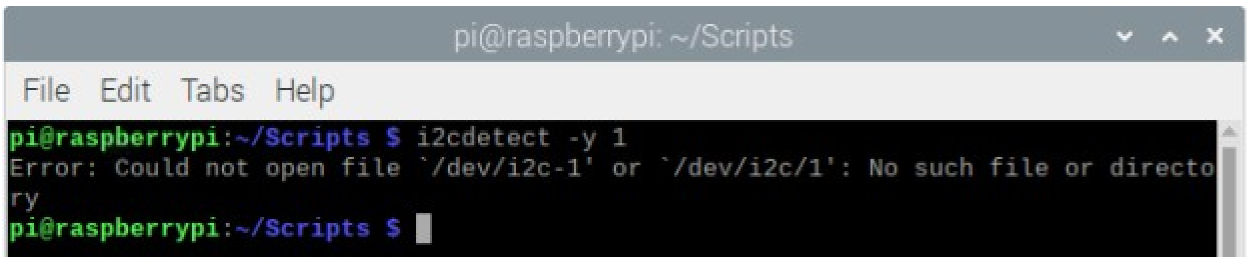

pi@raspberrypi: ~

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:          -- -- -- -- -- -- -- -- -- -- 0d -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

Where *0x0d* is the I2C address of the module.

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



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

To download the external library, save the script *gy271compass.py* from the following link: https://github.com/Slaveche90/gy271compass

Or use the *git* to clone the repository, by running the following command in the terminal:

git clone https://github.com/Slaveche90/gy271compass

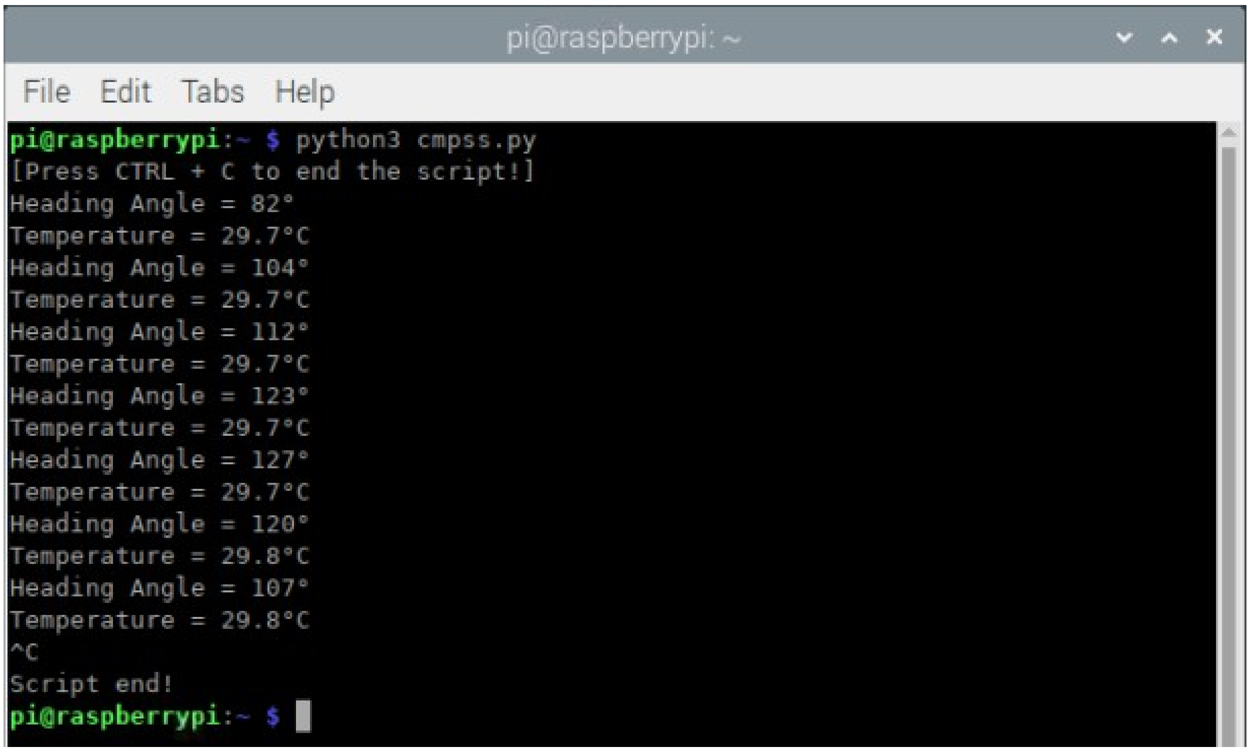
This command will create a directory called *gy271compass*. Open it and copy the *gy271compass.py* script into the same directory where the script from next chapter is saved.

Python script

```
import gy271compass as GY271
from time import sleep

sensor = GY271.compass(address=0x0d)

print('[Press CTRL + C to end the script!]')
try:
    while True:
        angle = sensor.get_bearing()
        temp = sensor.read_temp()

        print('Heading Angle = {}°'.format(angle))
        print('Temperature = {:.1f}°C'.format(temp))
        sleep(1)

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

Save the script by the name *gy271.py* in the same directory where you saved the *gy271compass* script. To run the *gy271.py* script open terminal in the directory where you saved the script and run the following command:
python3 gy271.py

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



```
pi@raspberrypi:~ $ python3 cmpss.py
[Press CTRL + C to end the script!]
Heading Angle = 82°
Temperature = 29.7°C
Heading Angle = 104°
Temperature = 29.7°C
Heading Angle = 112°
Temperature = 29.7°C
Heading Angle = 123°
Temperature = 29.7°C
Heading Angle = 127°
Temperature = 29.7°C
Heading Angle = 120°
Temperature = 29.8°C
Heading Angle = 107°
Temperature = 29.8°C
^C
Script end!
pi@raspberrypi:~ $
```

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

The script starts by importing external class library *gy271compass* and the *time* library.

Next, the object called *compass* is created using the following line of code:
`sensor = GY271.compass(address=0x0d)`
where *0x0d* is the I2C address of the module.

Then, the *try-except* block of code is created. In the *try* block of code an indefinite loop (*while True:*) is created. To read the heading angle in the indefinite loop, the following line of code is used:
`angle = sensor.get_bearing()`
and to read the temperature data, use the following line of code:
`temp = sensor.read_temp()`

The heading angle data is stored in the variable called *angle* and temperature data is stored in the variable called *temp*. These two variables are then used to display data in the terminal.

When the CTRL + C is pressed on the keyboard, the keyboard interrupt happens. On the keyboard interrupt, the *except* block of code gets executed and a message *Script end!* is displayed.

The module has several options that can be set at the *sensor* object creation. You can pass the following arguments to the *compass()* constructor:

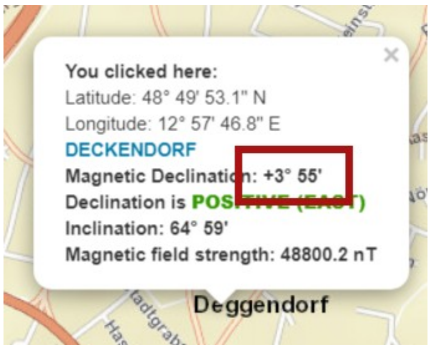
`address=0x0d`	- the I2C address of the module
`mode=MODE_CONT`	- Continuous or stand-by mode: *MODE_CONT* or *MODE_STBY*
`odr=ODR_10Hz`	- Output Data Rate, with values: *ODR_10Hz*, *ODR_50Hz*, *ODR_100Hz* or *ODR_200Hz*
`sens=SENS_2G`	- Sensitivity: *SENS_2G* or *SENS_8G*
`osr=OSR_512`	- OverSampling Rate, with values: *OSR_512*, *OSR_256*, *OSR_128* or *OSR_64*
`d=0.0`	- Declination data (explained in the next chapter)

Usage (these are the default settings):

```
sensor = GY271.compass(address=0x0d, mode=MODE_CONT,
   odr=ODR_10Hz, sens=SENS_2G, osr=OSR_512, d=0.0)
```

Calculating declination

To calculate the magnetic declination value, go to the following link:
https://www.magnetic-declination.com/YUGOSLAVIA/NEOPLANTA/2970848.html
and find your location. Next, click with your PC mouse on the location and pop up will be displayed. Copy the following data:


You clicked here:
Latitude: 48° 49' 53.1" N
Longitude: 12° 57' 46.8" E
DECKENDORF
Magnetic Declination: +3° 55'
Declination is POSITIVE (EAST)
Inclination: 64° 59'
Magnetic field strength: 48800.2 nT

And use the angle minutes to decimal converter, for example:
https://www.rapidtables.com/convert/number/degrees-minutes-seconds-to-degrees.html
to get the floating point value of the angle.

In this example: +3 degrees and 55 minutes = +3° 55’ = +3.916667°

```
sensor = GY271.compass(address=0x0d, d=3.916667)
```

NOTE: Do not forget the sign, because magnetic declination can be **+** or **-**.

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!

Impressum0

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