Welcome!

Thank you for purchasing our *AZ-Delivery GY-906 Infrared Temperature Module*. On the following pages, you will be introduced to how to use and set-up this handy device.

Have fun!

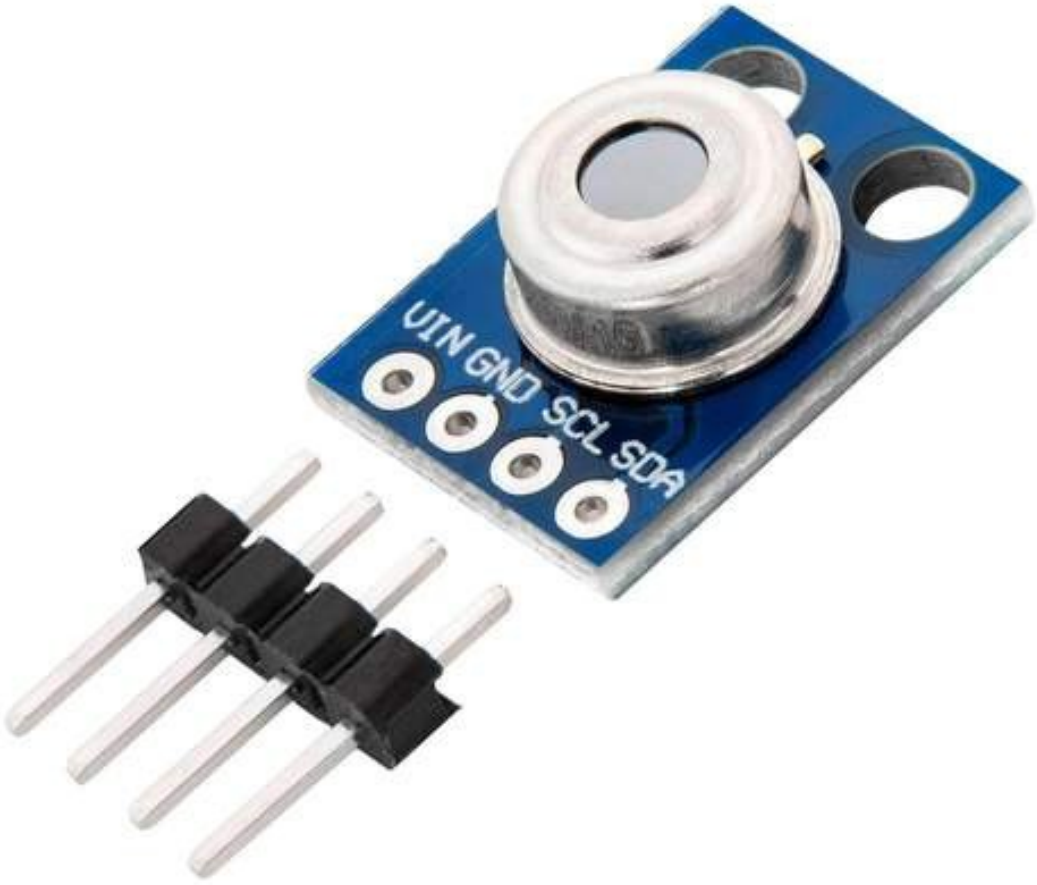


Table of Contents

Introduction

In the heart of the GY-906 module, there is the MLX90614 infrared sensor. It is an infrared thermometer for contactless temperature measurements. Both the IR sensitive thermopile detector chip and the signal conditioning ASIC are integrated into the same *TO-39* can package. A low noise amplifier, *17*-bit ADC and powerful DSP unit, that are integrated into the MLX90164, provide high accuracy and resolution of the thermometer.

An infrared thermometer is a device that measures temperature based on infrared radiation emitted by an object. Any material that has a temperature above absolute zero (*0*K, *-273.13*°C, *-459.67*°F) has molecules moving inside it. When the temperature is higher, the molecules move faster. These molecules emit infrared radiation, which is, in fact, the temperature, and the hotter they get the more radiation they emit. Infrared thermometers detect and measure this radiation.

Infrared thermometers use lens to focus infrared light from an object onto a detector known as a thermopile. The function of the thermophile is to absorb infrared radiation and convert it to heat. It becomes hotter as it absorbs more and more infrared energy. The excess heat is transformed into electricity. This electricity is transmitted to a detector, which determines the temperature of the object.

So, each object emits infrared light depending on its heat, and the light is detected using a thermopile which gets hotter and hotter, at the same time converting the excess heat to electricity.

The sensor measures infrared light that gets emitted by objects so it can sense temperature without having to touch the objects physically. Simply direct the sensor towards the object whose temperature you want to determine and it detects the temperature by absorbing the emitted infrared light.

NOTE: When measuring, please keep a measuring distance of *10mm* between the module and the object.

Specifications

» Operating voltage range:	from 3.3V to 5V DC
» Max current	2mA
» Communication protocol:	I2C
» Range for ambient temperature:	from -40°C to 125°C [-40 to 257°F]
» Range for object temperature:	from -70 to 380°C [-94 to 716°F]
» Accuracy:	±0.5°C for both ambient and object
» Dimensions:	11 x 17 x 6mm [0.4 x 0.7 x 0.24in]
» Field of view:	90 degrees

The module can detect a broader range of temperatures than most digital sensors (from *-70*°C to *+380*°C) because it does not have to touch the object whose temperature it is measuring. Keep a measuring distance of *10mm* when measuring the temperature, because the temperature readings change a lot if the module is further away from the object.

The module supports the I2C serial interface. The module has a default I2C address, which is *0x5a.* The module has a fixed I2C address, so only one module can be used on the same I2C interface. To use more modules on the same I2C interface, use the I2C multiplexer.

The module can be useful for determining the average temperature of an area as it measures a *90*-degree field of view.

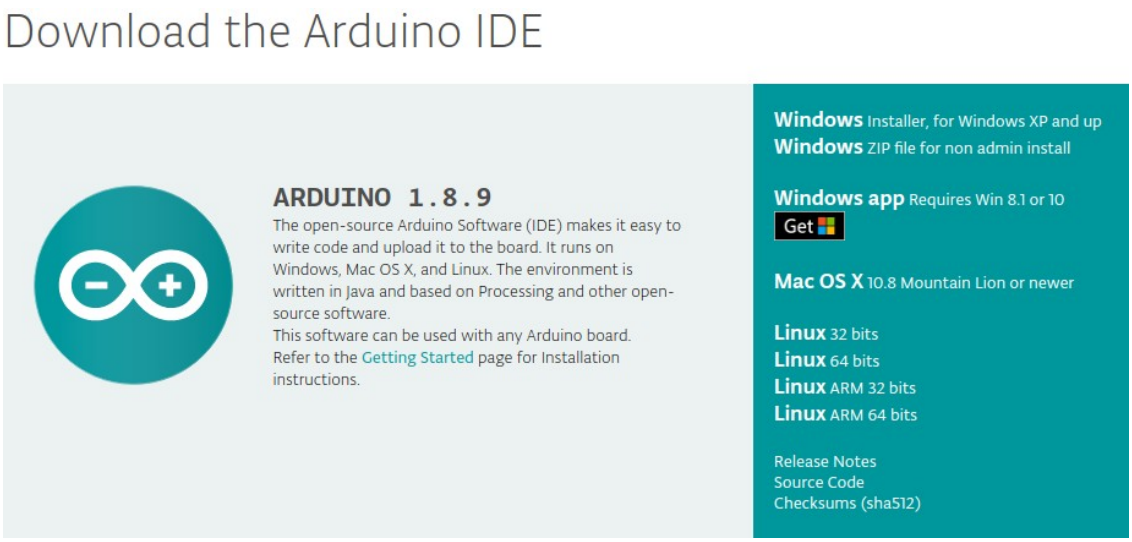
The pinout

The GY-906 infrared temperature module has four pins. The pinout is shown in the following image:

The pins of the module can be connected to a *3.3V* or *5V* power supply without danger to the sensor itself. The module has the *3.3V* voltage regulator on-board.

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*.

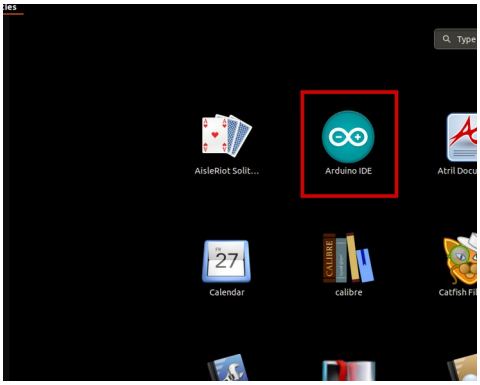
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

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.

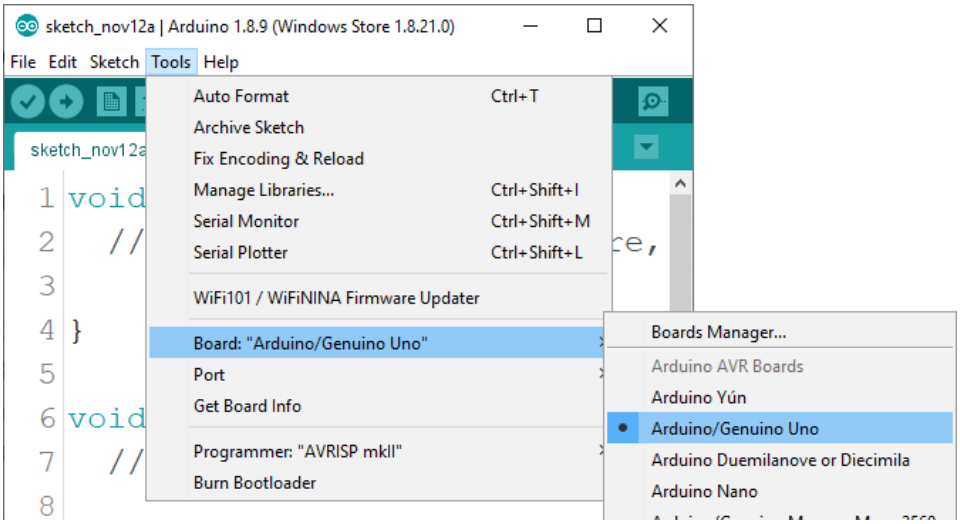
The second script, called *install.sh*, has to be installed after installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

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



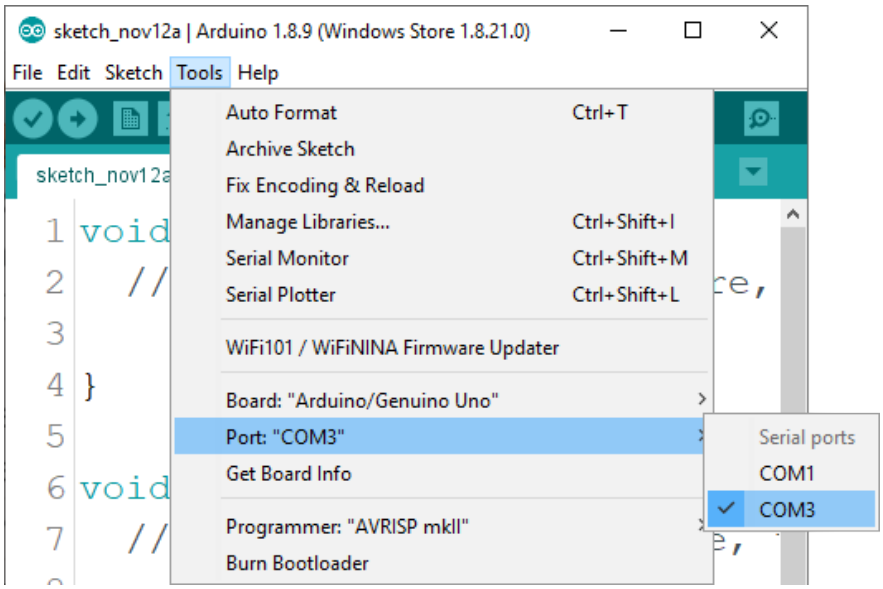
Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

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:



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*.

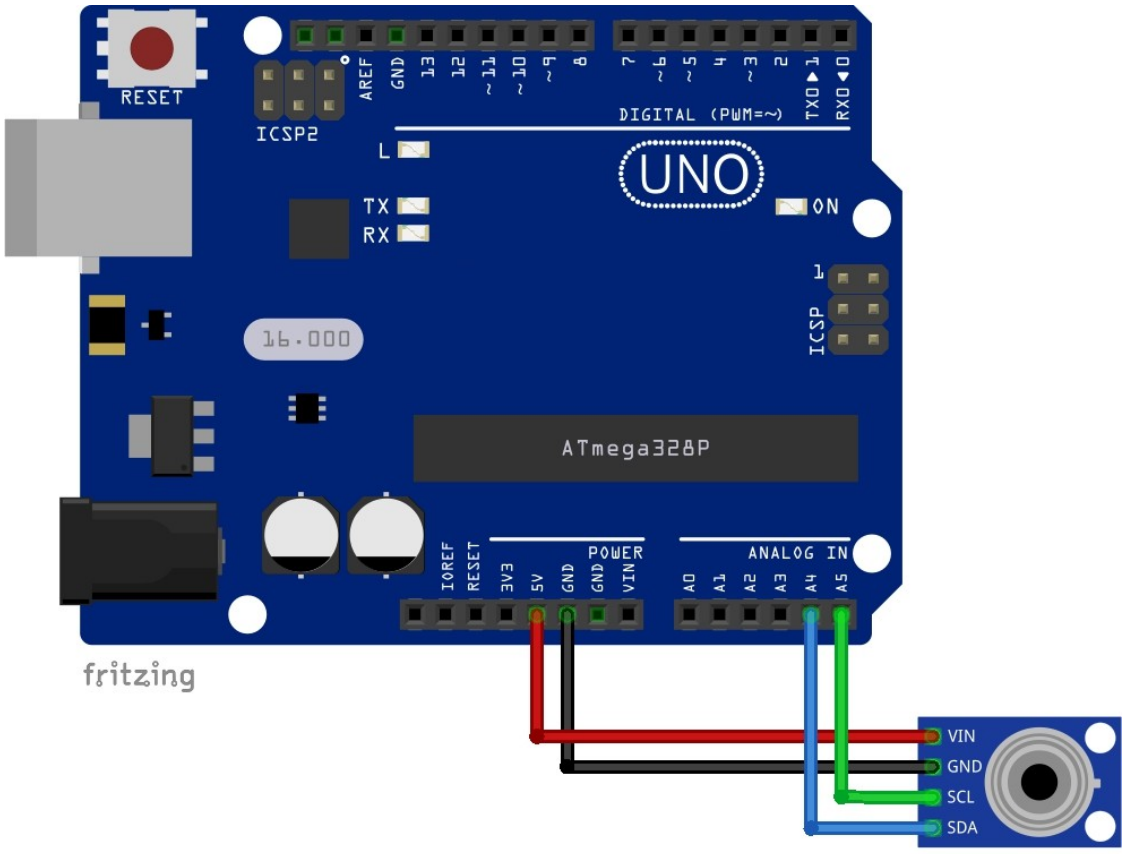
How to set-up the Raspberry Pi and Python

For the Raspberry Pi, first the operating system has to be installed, everything has to be set-up so that it can be used in *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 screen with Uno

Connect the GY-906 module with the Uno as shown on the following connection diagram:

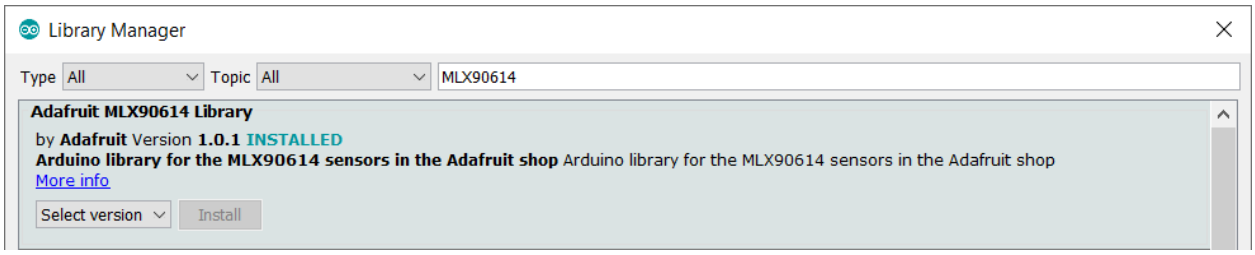


MLX90164 pin	Uno pin	Wiring color
VIN	5V	**Red wire**
GND	GND	**Black wire**
SCL	A5	**Green wire**
SDA	A4	**Blue wire**

Library for Arduino IDE

To use the module with Uno it is recommended to download an external library for it. The library that is used in this eBook is called *Adafruit_MLX90614*. To download and install it open Arduino IDE and go to: *Tools > Manage Libraries*

When a new window opens, type *MLX90614* in the search box and install the library called *Adafruit MLX90164 Library* made by *Adafruit* as shown in the following image:



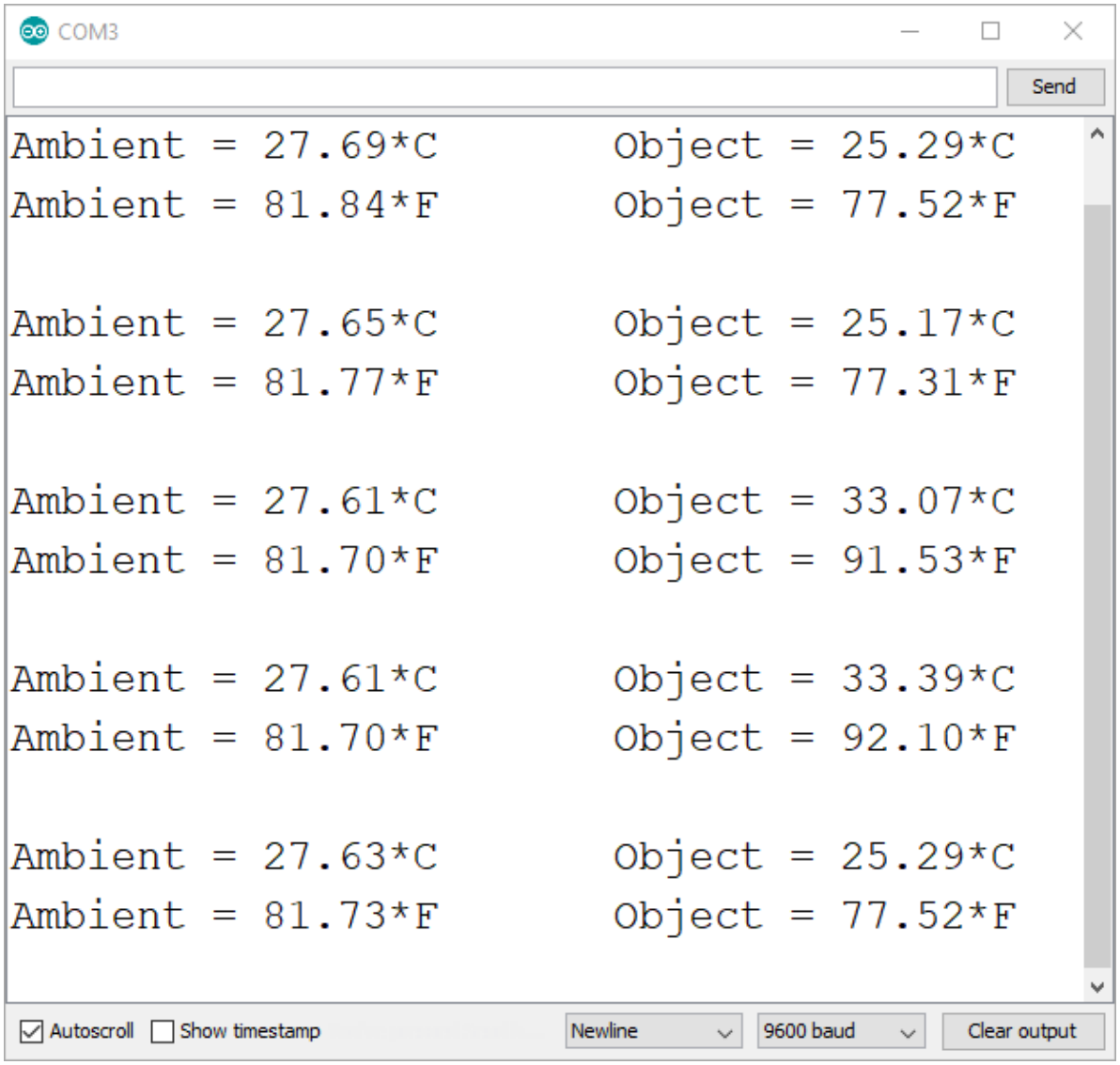
Sketch example

The following sketch example is modified sketch from the *Adafruit MLX90614 Library*:

File > Examples > Adafruit MLX90614 > mlxtest

```
#include <Wire.h>
#include <Adafruit_MLX90614.h>
Adafruit_MLX90614 itemp = Adafruit_MLX90614();
void setup() {
   Serial.begin(9600);
   Serial.println("Adafruit MLX90614 test");
   itemp.begin();
}
void loop() {
  Serial.print("\nAmbient = ");
  Serial.print(itemp.readAmbientTempC());
  Serial.print("*C\tObject = ");
  Serial.print(itemp.readObjectTempC());
  Serial.println("*C");
  Serial.print("Ambient = ");
  Serial.print(itemp.readAmbientTempF());
  Serial.print("*F\tObject = ");
  Serial.print(itemp.readObjectTempF());
  Serial.println("*F");
  Serial.println();
  delay(2500);
}
```

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



The sketch begins with including two libraries: *Wire* and *Adafruit_MLX90614*. The first one is used for I2C communication and the second one is used for functions to control the sensor.

Next, the object called *itemp* is created with the following line of code:

```
Adafruit_MLX90614 itemp = Adafruit_MLX90614;
```

The object *itemp* represents the module, and it is used to read data from the sensor.

In the *setup()* function, the serial communication is started with the baud rate of *9600bps.*

Then, the *itemp* object is initialized with the following line of code:

```
itemp.begin()
```

In the *loop()* function, the module data is read and displayed into the Serial Monitor.

To read the ambient temperature in Celsius, use the following line of code:

```
itemp.readAmbientTempC()
```

To read the object temperature in Celsius (object that is in front of the sensor) use the following line of code:

```
itemp.readObjectTempC()
```

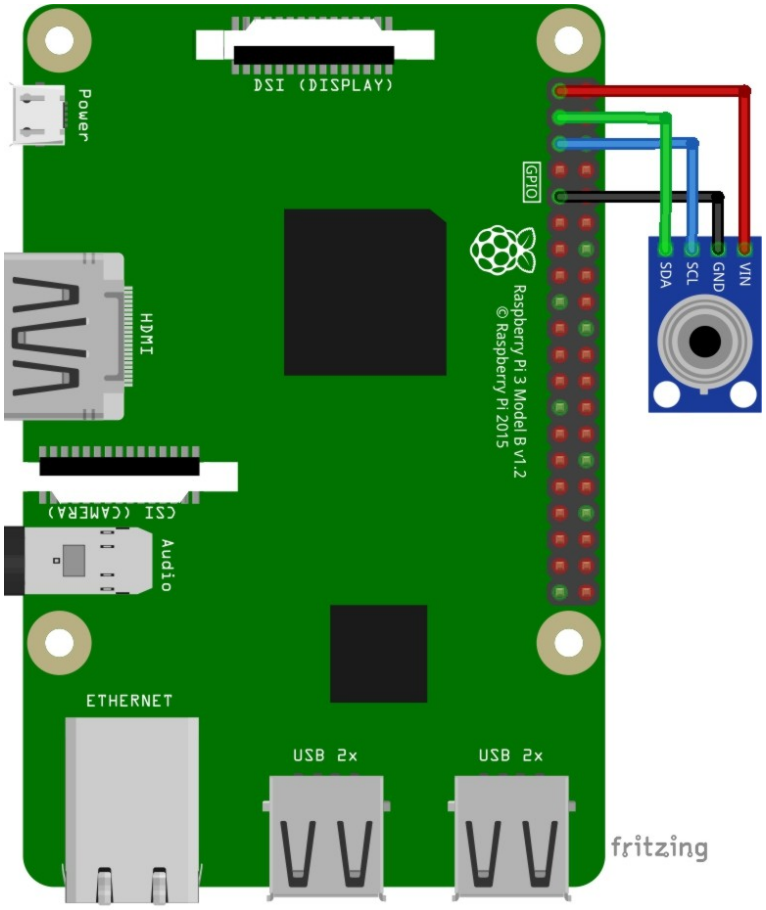
To read the ambient temperature in Fahrenheit, use the following line of code: `itemp.readAmbientTempF()`

To read the object temperature in Fahrenheit (object that is in front of the sensor) use the following line of code:
`itemp.readObjectTempF()`

At the end of the *`loop()`* function there is a delay pause of *2.5* seconds (*2500* milliseconds) which can be changed in the following line of code:
`delay(2500);`

Connecting the screen with Raspberry Pi

Connect the GY-906 module with the Raspberry Pi as shown in the following connection diagram:

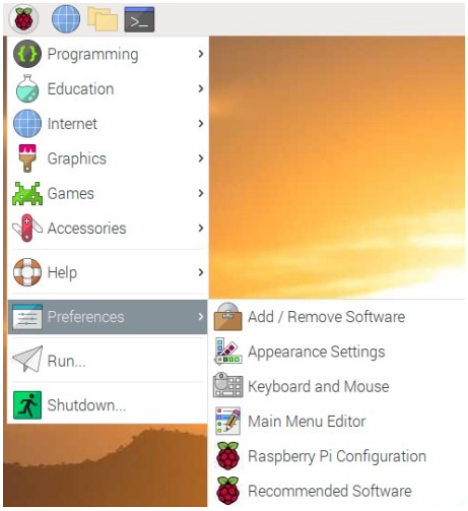


MLX90164 pin	Raspberry Pi pin	Physical pin No.	Wire color
GND	GND	9	**Black wire**
VIN	3V3	1	**Red wire**
SCL	GPIO3	5	**Blue wire**
SDA	GPIO2	3	**Green wire**

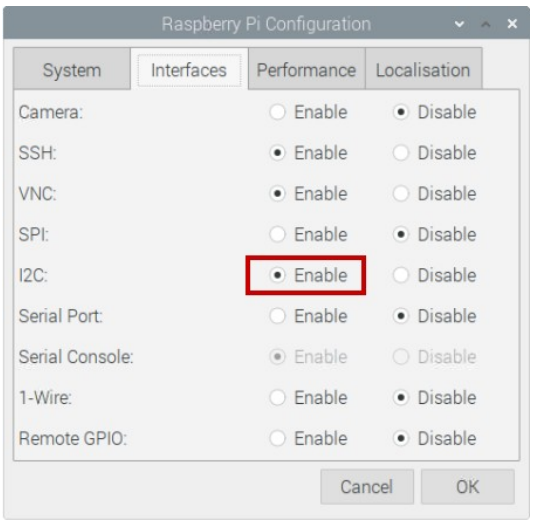
Enabling the I2C interface

In order to use the module 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*, as shown in the following image:



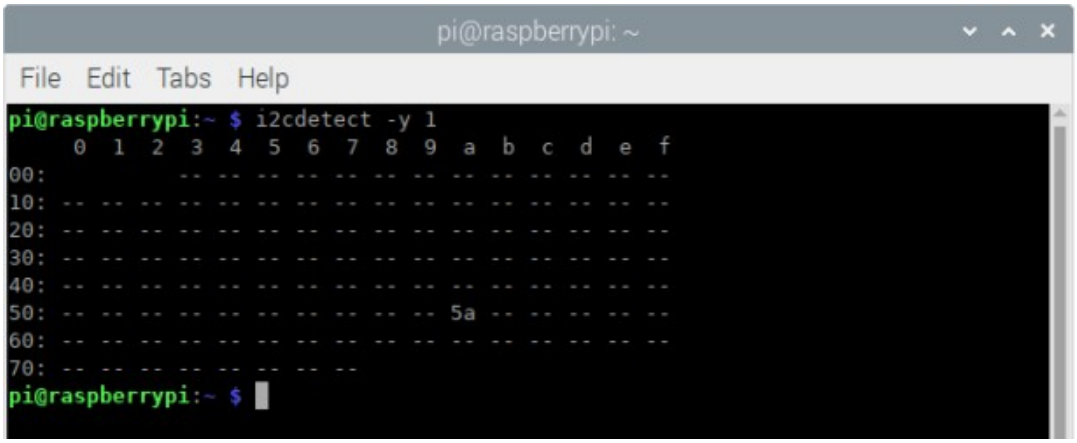
To detect the I2C address of the module, the *i2ctools* has to be installed on the Raspbian, if it is not installed, open the terminal and run the following commands:

```
sudo apt-get update
sudo apt-get install i2c-tools
```

Then, open the terminal and run the following command:

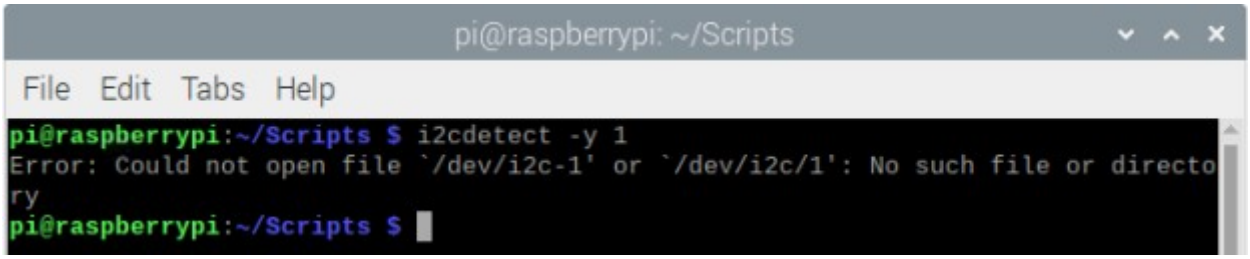
```
i2cdetect -y 1
```

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



Where *0x5a* is the I2C address of the sensor.

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



Python script

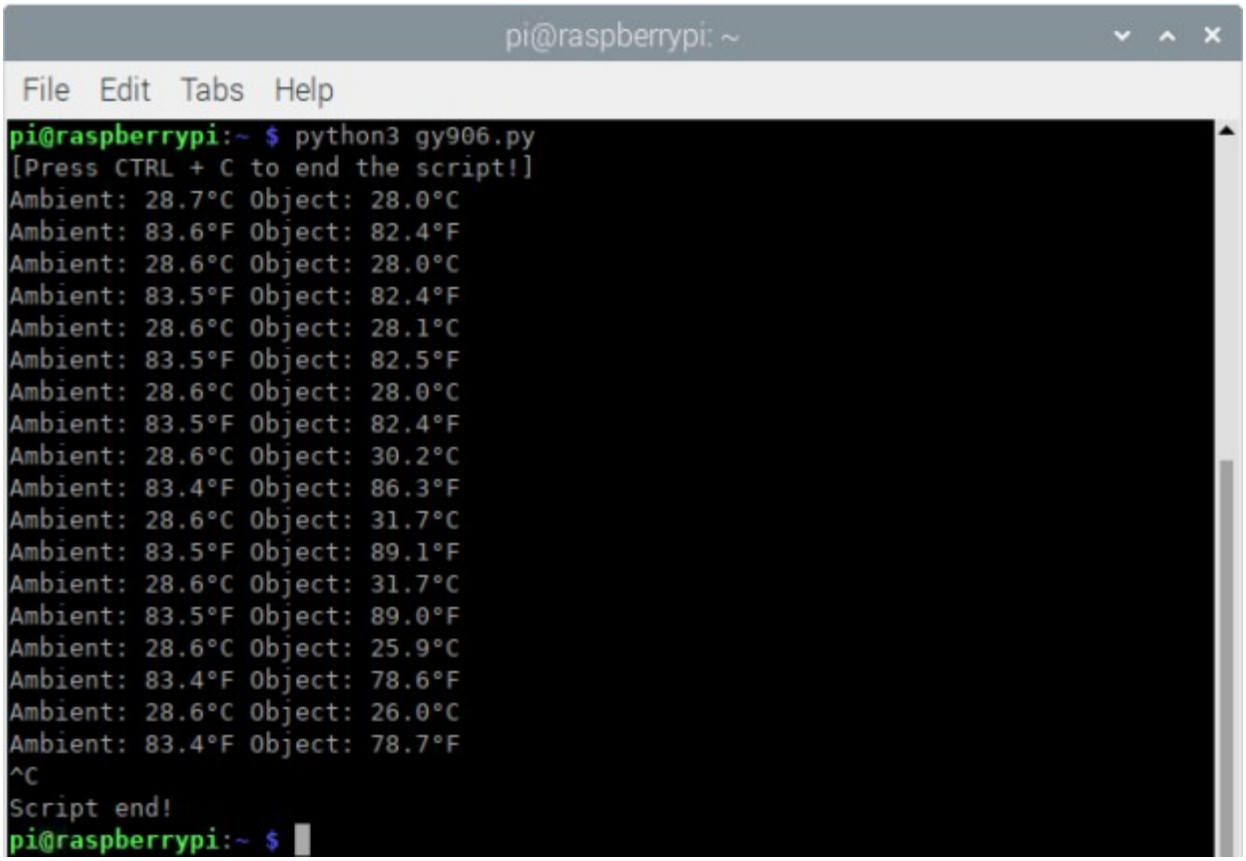
```
import time
import smbus
ds = u'\xb0' # UTF-8 degree sign
i2c_address = 0x5a # I2C address of the GY-906 sensor

# Temperature registers addresses
MLX90614_TA = 0x06 # Ambient temp
MLX90614_TOBJ1 = 0x07 # Object temp

# Read temperature registers and calculate Celsius
def ambC():
    val = bus.read_i2c_block_data(i2c_address, MLX90614_TA, 2)
    temp = val[1] << 8
    temp |= val[0]
    temp *= .02
    temp  -= 273.15
    return temp

def objC():
    val = bus.read_i2c_block_data(i2c_address, MLX90614_TOBJ1, 2)
    temp = val[1] << 8
    temp |= val[0]
     temp *= .02
    temp  -= 273.15
    return temp
```

```
def ambF():
    return ambC() * 9.0 / 5.0 + 32.0

def objF():
    return objC() * 9.0 / 5.0 + 32.0

# Initialize I2C (SMBus)
bus = smbus.SMBus(1)

print('[Press CTRL + C to end the script!]')
try:
    while True:
        print('Ambient: {:.1f}{}C\tObject: {:.1f}{}C'.
            format(ambC(), ds, objC(), ds))
        print('Ambient: {:.1f}{}F\tObject: {:.1f}{}F'.
            format(ambF(), ds, objF(), ds))
        time.sleep(1)

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

Save the script by the name *gy906.py*. To run the script open the terminal in the directory where the script is saved and run the following command:

python3 gy906.py

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



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

The script starts with importing two libraries: *time* and *SMBus*. The *time* library is used for time functionality and the *SMBus* (System Management Bus) is used for reading/writing information on the I2C interface.

Next, the variable *ds* is created. The UTF degree sign value is stored here. It is needed to display the result accurately.

Then, the addresses of two registers of the sensor are stored in the variables called *MLX90614_TA* and *MLX90614_TOBJ1,* with the following lines of code:

```
MLX90614_TA = 0x06
MLX90614_TOBJ1 = 0x07
```

In the register called *MLX90614_TA* the ambient temperature data is stored and in the register called *MLX90614_TOBJ1* the object temperature data is stored.

Next, two functions that read temperature data from the sensor are crated. The first function is called *ambC().* It has no arguments and returns a float value. The function reads the register that holds the ambient temperature data, and converts raw data into readable temperature in Celsius. The returned float value represents the ambient temperature data in Celsius.

The second function is called *objC().* It has no arguments and returns a float value. The function reads the register that holds the object temperature data and converts raw data into readable temperature in Celsius. The returned float value represents the object temperature data in Celsius.

To convert temperature from Celsius to Fahrenheit, two functions called *ambF()* and *objF()* are created. The *ambF()* function uses the formula to convert Celsius temperature read with *ambC()* function into Fahrenheit. The *objF()* function uses the formula to convert Celsius temperature read with *objC()* function into Fahrenheit.

Then, the I2C communication is initialized with the following code:
bus = smbus.SMBus(1)

Next, the *try-except* block of code is created. In the *try* block of code the indefinite loop block (*while True:*) is created. Inside this block of code, data is read and displayed in the terminal. Between the two loops of the indefinite loop block of code, there is one second pause, which duration can be changed in the following line of code: *time.sleep(1)*

The *except* block of code is executed when the *CTRL + C* is pressed on the keyboard. This is called the keyboard interrupt. When this block code is executed the message *Script end!* is displayed in the terminal.

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