

AZ-Delivery

Welcome!

Thank you for purchasing our *AZ-Delivery GY-302 (BH1750) Light Sensor Module*. On the following pages, you will be introduced to how to use and set-up this handy device.

Have fun!

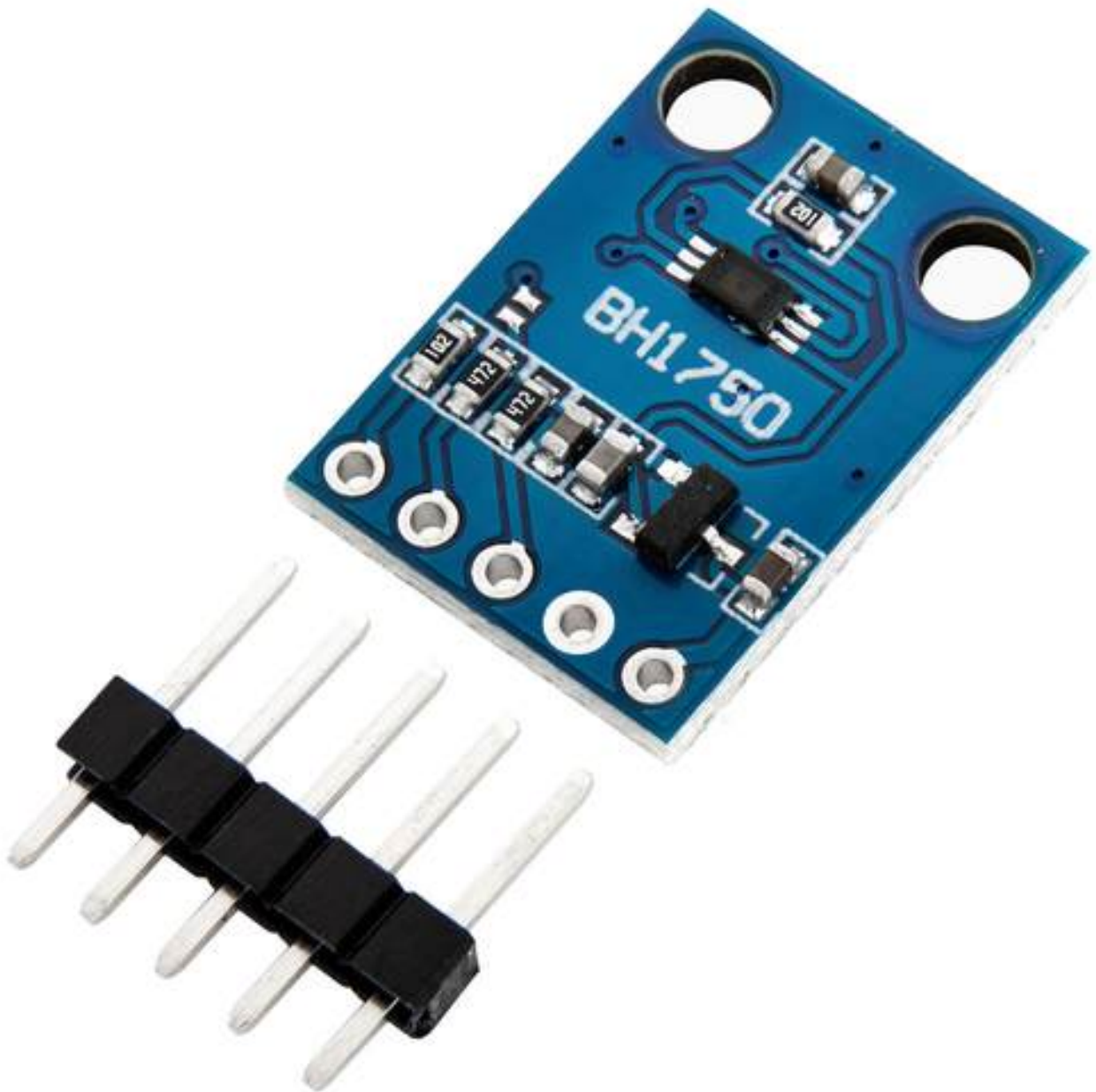




Table of Contents

Introduction.....	3
Specifications.....	4
The pinout.....	6
How to set-up Arduino IDE.....	7
How to set-up the Raspberry Pi and Python.....	11
Connecting the module with Uno.....	12
Library for Arduino IDE.....	13
Sketch example.....	14
Connecting the module with Raspberry Pi.....	16
Library and tools for Python.....	17
Python script.....	18



Introduction

The GY-302 (BH1750) Light sensor module is a device that can measure the intensity of visible light (wave length about 400 – 700 nm). The module is based on the BH1750 light sensor chip, in principle a photo diode with an internal amplifier and a 16 bit ADC (Analog-Digital-Converter).

The BH1750 chip is used in many applications such as mobile phones, LCD TV, portable PCs, handheld devices (PDAs), portable game machines, digital cameras, digital video cameras, car navigations, LCD displays.

The BH1750 sensor can be used to adjust display brightness in mobile phones and other types of LCD displays or to turn ON/OFF automobile headlights depending on the outdoor light.

Comparing to the human eye, the sensor is very close with high resolution and light detecting range.

The sensor chip operates with voltages in range of 2.8V to 3.3V. The module has an on-board voltage regulator so it can work with voltages up to 5V.

The chip does not need any calculation to measure the Lux values, because the sensor gives these values directly. It measures the intensity of light according to the amount of light to which has been exposed.

Specifications

Operating voltage range:	from 3.3V to 5V DC
Max operating current:	190µA (max)
Sensor resolution	16bits
Peak wavelength	560nm
Measuring Range	from 1 to 65535lx
Accuracy	+/-20%
Interface	I2C (address 0x23 or 0x5C)
Operating Temperature	-40°C ~ 85°C
Dimensions:	19 x 14 x 2mm [0.7 x 0.5 x 0.1in]

The BH1750 has six different measurement modes which are divided in two groups; continuous and one-time measurements. In continuous mode the sensor continuously measures light value. In one-time mode, the sensor makes only one measurement and then goes into *Power Down* mode.

Each mode has three different precisions: low resolution mode - (4 lx precision, 16ms measurement time), high resolution mode - (1 lx precision, 120ms measurement time) and high resolution mode 2 - (0.5 lx precision, 120ms measurement time)

When using the Arduino IDE, continuous high resolution mode is used by default. That can be changed to a different mode by changing the argument inside *begin()* function.



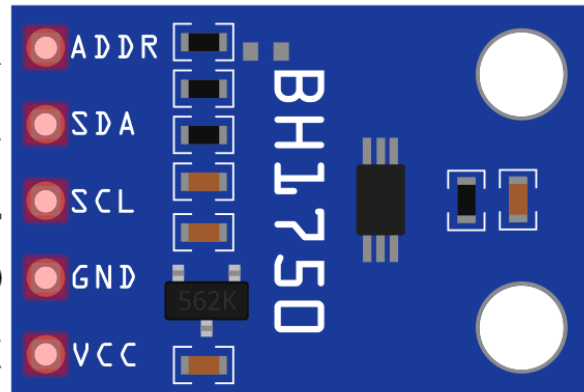
Features

- I2C bus Interface
- Spectral measurement is approximately human eye response
- Illuminance to Digital Converter
- Wide range and High resolution. (1 - 65535 lx)
- Low Current by power down function
- 50Hz / 60Hz Light noise reject-function
- 1.8V Logic input interface
- No need any external parts
- Light source dependency is little. (e.g. Incandescent Lamp, Fluorescent Lamp, Halogen Lamp. White LED, Sun Light)
- It is possible to select 2 different I2C slave-addresses
- Adjustable measurement result for influence of optical window (It is possible to detect min. 0.11 lx, max. 100000 lx by using this function)
- Small measurement variation (+/- 20%)
- The influence of infrared light is very small

The pinout

The light sensor module has five pins. The pinout diagram is shown on the following image:

ADDRESS SELECT - ADDR
I2C SERIAL DATA LINE -SDA
I2C SERIAL CLOCK LINE - SCL
GROUND - GND
POWER SUPPLY - VCC



VCC - Power supply pin

GND - Ground connection pin

SCL - I2C Serial clock line connection pin

SDA - I2C Serial data line connection pin

ADDR - Device address pin, used to select the address when more than one module is connected on the same I2C bus.

The ADDR pin is used to set the sensor I2C address. By default (if ADDR voltage less than $0.7 * VCC$) the sensor address will be 0x23. If it has voltage greater or equal to 0.7V voltage (connected to VCC) the sensor address will be 0x5C.

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. The Arduino IDE version used for this ebook is 1.8.12.

Download the Arduino IDE



ARDUINO 1.8.12

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.

Az-Delivery

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*.

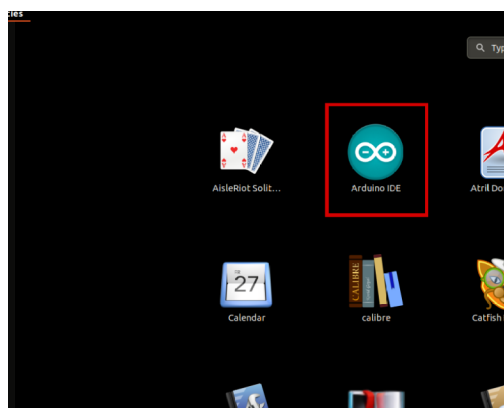
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

```
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 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.



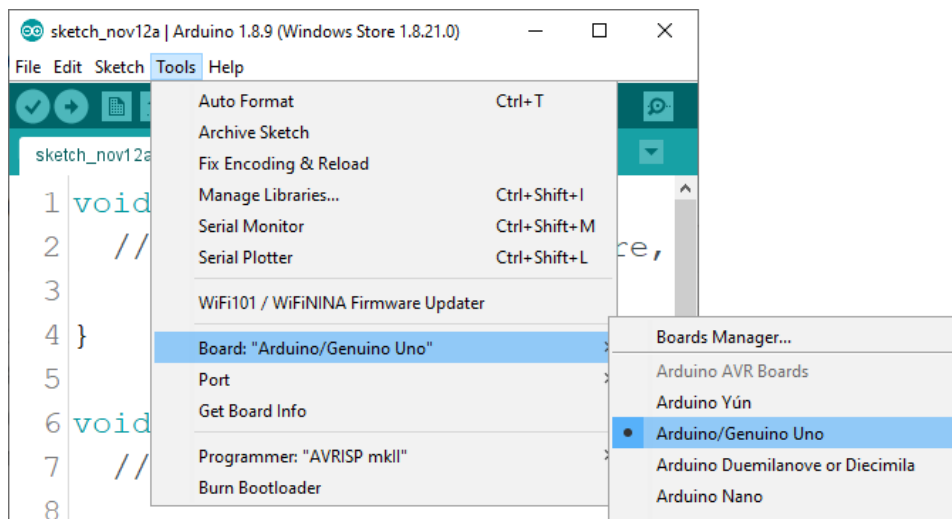
Az-Delivery

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:



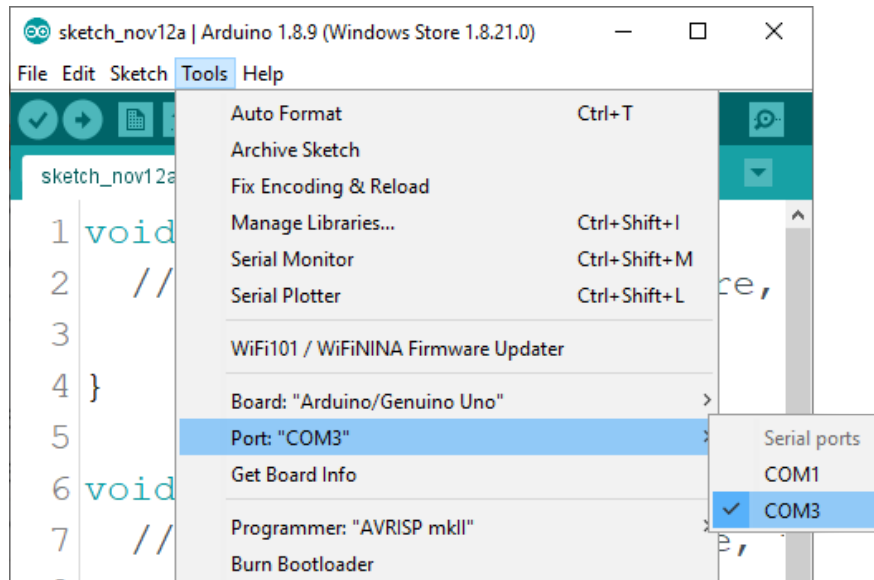
The port to which the Arduino board is connected has to be selected. Go to:

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.

Az-Delivery

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

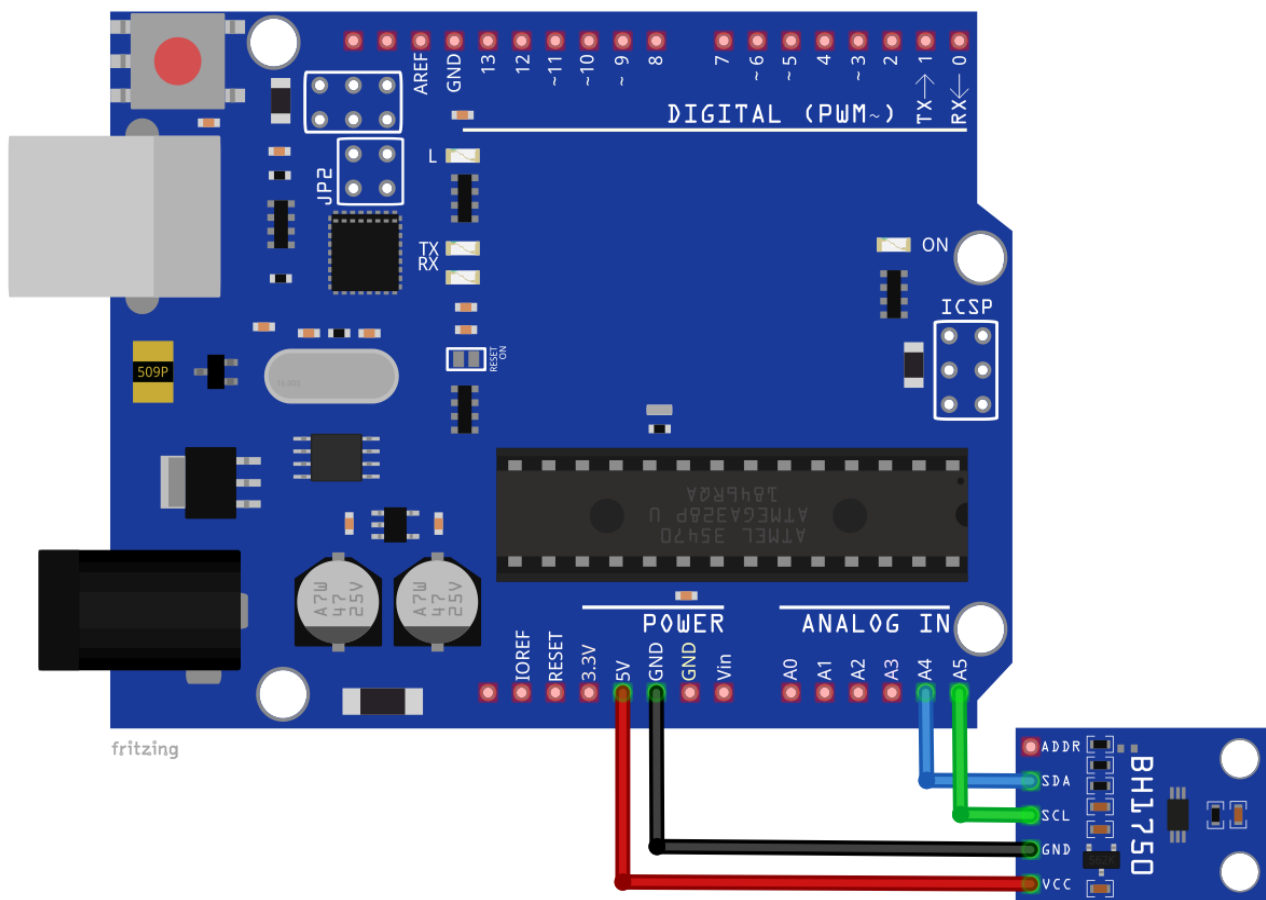
For the Raspberry Pi, first the operating system has to be installed, then everything has to be set-up so that it can be used in the *Headless* mode. The *Headless* mode enables remote connection to the Raspberry Pi, without the need for a *PC* screen Monitor, mouse or keyboard. The only things that 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 *Raspberry Pi OS* (operating system), recently known as *Raspbian*, comes with *Python* preinstalled.

Connecting the module with Uno

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



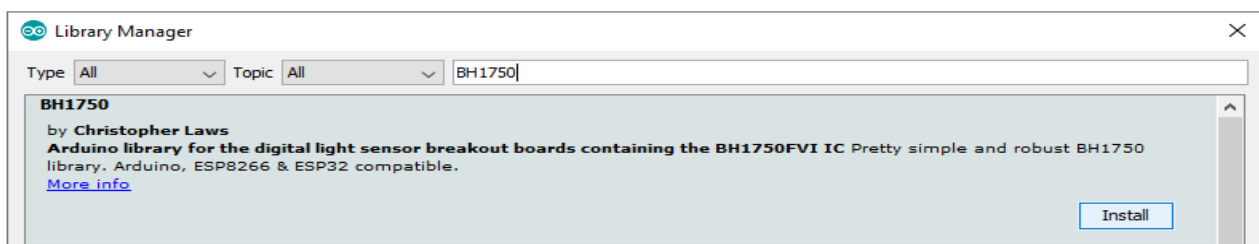
Module pin	Uno pin	Wire color
DATA	A4	Blue Wire
GND	A5	Green wire
GND	GND	Black Wire
VCC	5V	Red Wire

Library for Arduino IDE

To use the sensor module with a Uno, it is recommended to download an external library. The library we are going to use is called the *BH1750*. The version of the library that is used is 1.1.4. To download and install it, open Arduino IDE and go to:

Tools > Manage Libraries.

When new window opens, type *BH1750* in the search box and install the library *BH1750* made by *Christopher Laws*, as shown in the following image:



With the library come several sketch examples, to open one, go to:

File > Examples > BH1750 > BH1750test

With this sketch example the module can be tested. The sketch in this eBook is a modified version of this sketch, to get more user-friendly output.

Az-Delivery

Sketch example

```
#include <Wire.h>
#include <BH1750.h>

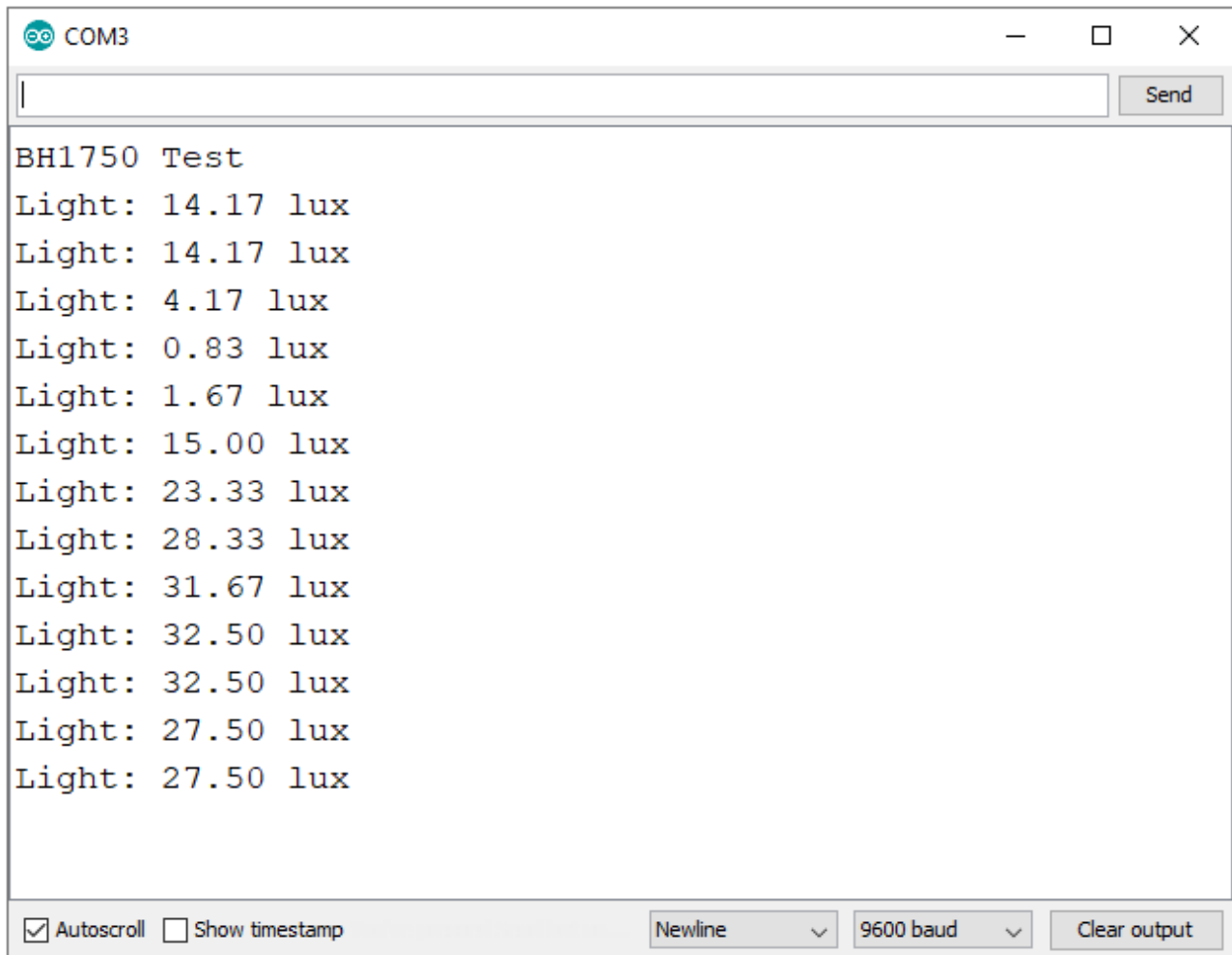
BH1750 lightMeter;

void setup(){
  Serial.begin(9600);
  Wire.begin();
  lightMeter.begin();
  Serial.println(F("BH1750 Test"));
}

void loop() {
  float lux = lightMeter.readLightLevel();
  Serial.print("Light: ");
  Serial.print(lux);
  Serial.println(" lx");
  delay(2000);
}
```

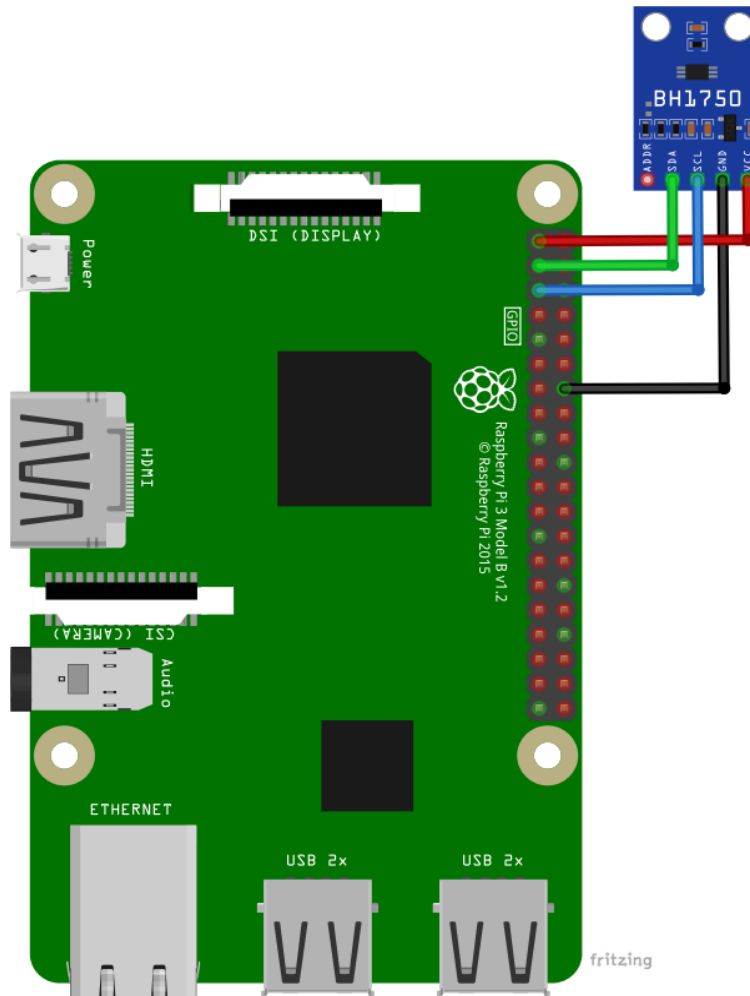
Az-Delivery

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



Connecting the module with Raspberry Pi

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



Module pin	Raspberry Pi pin	Physical pin	Wire color
VCC	3V3	1	Red Wire
SDA	GPIO2	3	Green wire
SCL	GPIO3	5	Blue wire
GND	GND	14	Black wire



Library and tools for Python

To use the module with the Raspberry Pi, the several libraries have to be installed. To install the libraries, open the terminal and run the following commands, one by one:

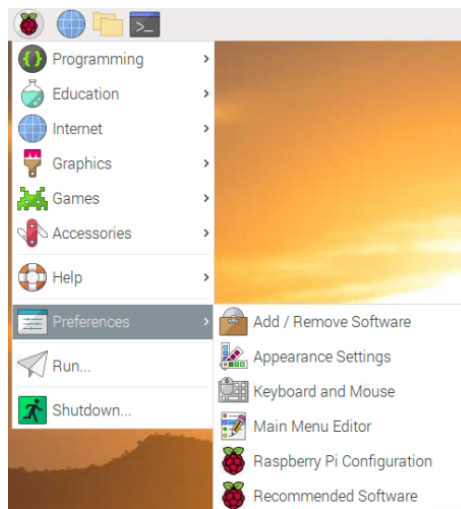
```
sudo apt-get update && sudo apt-get upgrade
```

```
sudo apt-get install python-smbus python3-smbus python-dev python3-dev i2c-tools
```

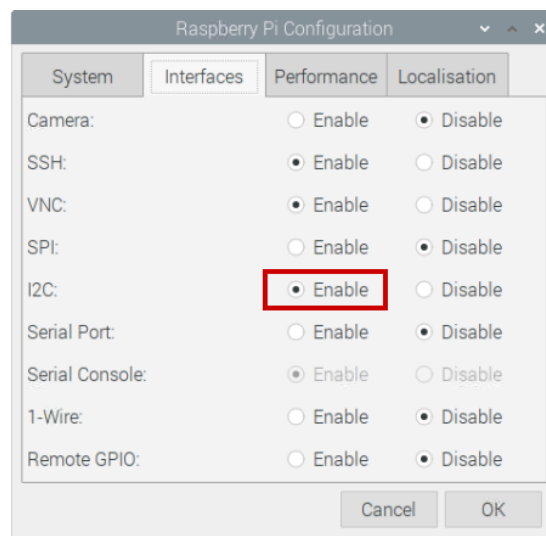
Enabling the I2C interface

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:



Az-Delivery

To detect the I2C address of the module the *i2ctools* should be installed.

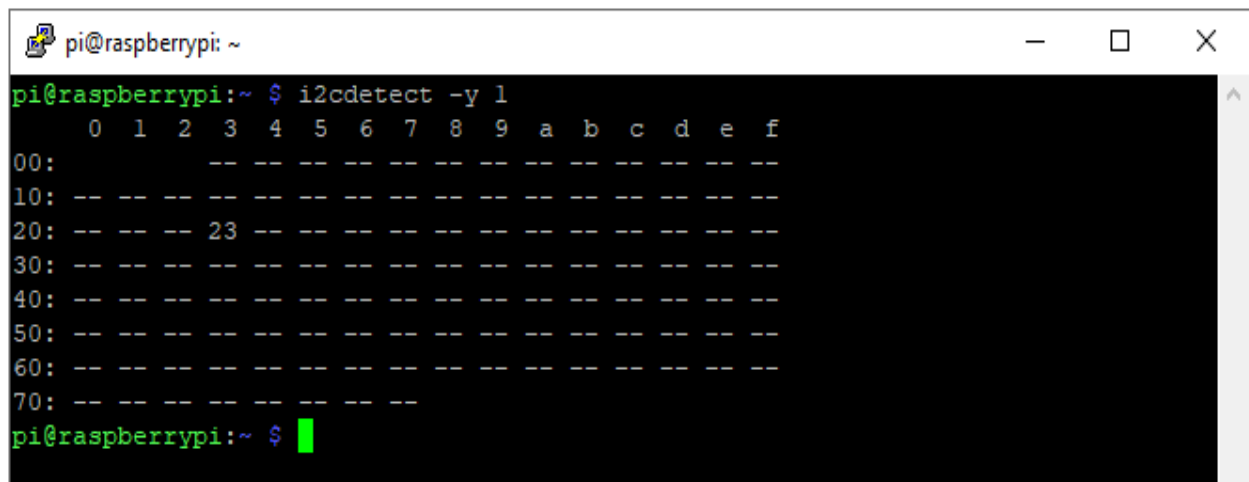
To install it, open terminal and run the following command:

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

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

```
i2cdetect -y 1
```

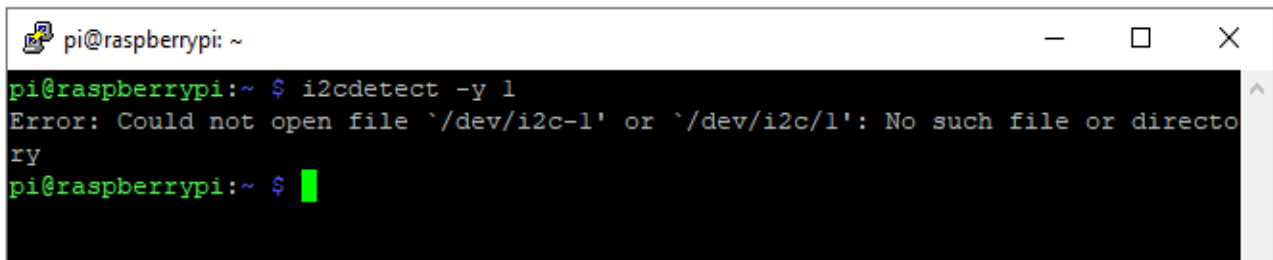
The terminal output should look like as on the following image:

A terminal window titled 'pi@raspberrypi: ~' showing the command 'i2cdetect -y 1' and its output. The output is a grid where the first column lists hexadecimal addresses from 00 to 70 in increments of 10. The subsequent columns are labeled with hexadecimal digits 0 through f. Most cells contain '--', indicating no device detected. The cell at row '20' and column '3' contains '23', indicating a device at address 0x23.

```
pi@raspberrypi:~ $ i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- 23 -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

The module I2C address is 0x23.

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

A terminal window titled 'pi@raspberrypi: ~' showing the command 'i2cdetect -y 1' and an error message. The error message states: 'Error: Could not open file '/dev/i2c-1' or '/dev/i2c/l': No such file or directory'.

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



Python script

```
import smbus
import time

DEVICE      = 0x23

POWER_DOWN  = 0x00
POWER_ON    = 0x01
RESET       = 0x07

CONTINUOUS_LOW_RES_MODE = 0x13
CONTINUOUS_HIGH_RES_MODE_1 = 0x10
CONTINUOUS_HIGH_RES_MODE_2 = 0x11
ONE_TIME_HIGH_RES_MODE_1 = 0x20
ONE_TIME_HIGH_RES_MODE_2 = 0x21
ONE_TIME_LOW_RES_MODE = 0x23

bus = smbus.SMBus(1)

def convertToNumber(data):
    result=(data[1] + (256 * data[0])) / 1.2
    return result

def readLight(addr=DEVICE):
    data = bus.read_i2c_block_data(addr, ONE_TIME_HIGH_RES_MODE_1)
    return convertToNumber(data)

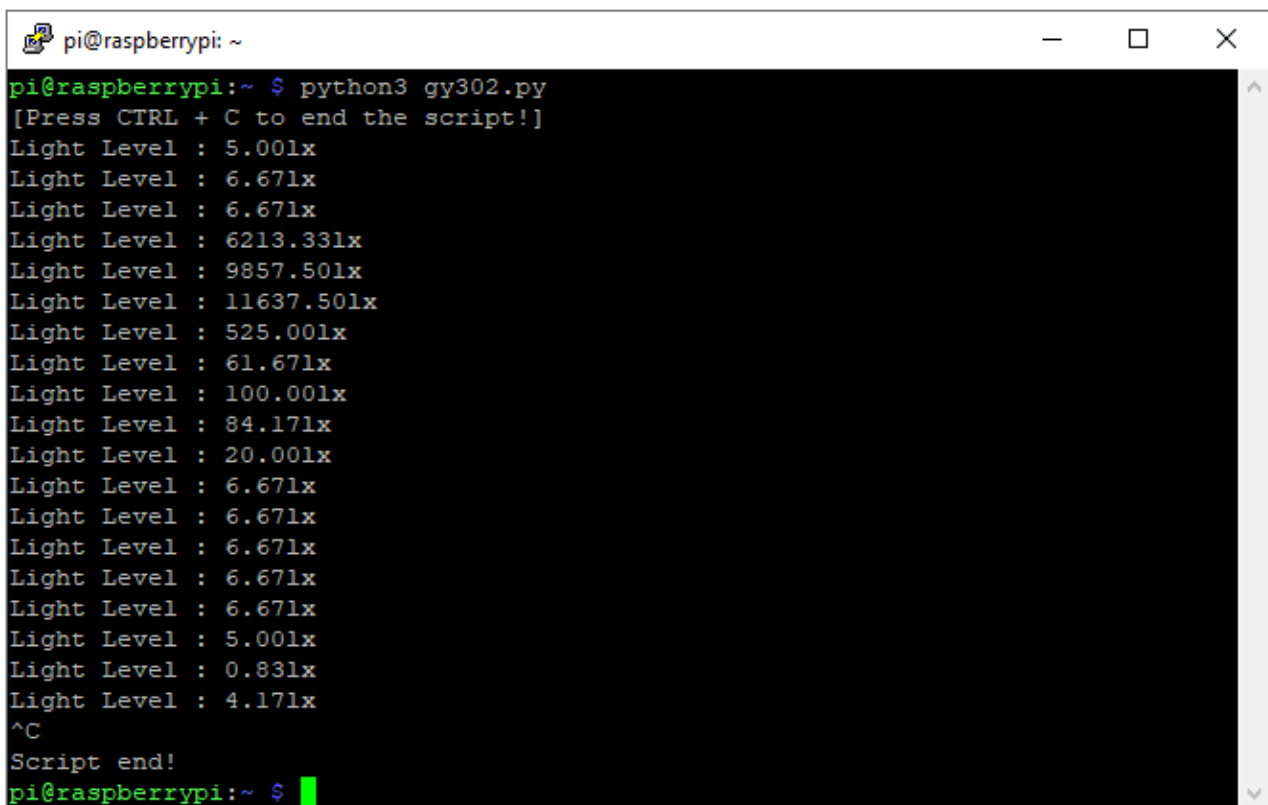
print('[Press CTRL + C to end the script!]\n')
try:
    while True:
        lightLevel = readLight()
        print("Light Level : {:.2f}lx".format(lightLevel))
        time.sleep(1)
except KeyboardInterrupt:
    print('\nScript end!')
```

Az-Delivery

Save the script by the name *gy302.py*. To run the script, open the terminal in the directory where the script is saved and run the following command:

python3 gy302.py

The result should look like as on the following image:



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ python3 gy302.py  
[Press CTRL + C to end the script!]  
Light Level : 5.00lx  
Light Level : 6.67lx  
Light Level : 6.67lx  
Light Level : 6213.33lx  
Light Level : 9857.50lx  
Light Level : 11637.50lx  
Light Level : 525.00lx  
Light Level : 61.67lx  
Light Level : 100.00lx  
Light Level : 84.17lx  
Light Level : 20.00lx  
Light Level : 6.67lx  
Light Level : 6.67lx  
Light Level : 6.67lx  
Light Level : 6.67lx  
Light Level : 6.67lx  
Light Level : 5.00lx  
Light Level : 0.83lx  
Light Level : 4.17lx  
^C  
Script end!  
pi@raspberrypi:~ $
```

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

Az-Delivery

At the beginning of the script, libraries *smbus* and *time* are imported.

Then the several variables are defined. First variable called *DEVICE* is I2C address of the sensor which has the value of 0x23. The second variable called *POWER_DOWN* has the hexadecimal value of 0x00 that represents the *no active state* of the sensor. The third variable called *POWER_ON* has the hexadecimal value of 0x01 that represents the *Power ON* state of the sensor. Fourth variable is the *RESET* which is the reset data register and it has a value of 0x07.

Next, several variables are created. The values in these variables represent the hexadecimal values of the registers in the sensor.

First measurement mode starts with lux values up to 4lx and it is called continuous low resolution mode. The measurement time in this mode is around 16ms. The register value for this scanning mode is 0x13.

Second measurement mode starts with lux values up to 1lx and it is called continuous high resolution mode. The measurement time in this mode is around 120ms. The register value for this scanning mode is 0x10.

Third measurement mode starts with lux values up to 0.5lx and it is called continuous high resolution mode. The measurement time in this mode is around 120ms. The register value for this scanning mode is 0x11.

Az-Delivery

Fourth measurement mode starts with lux values up to 1lx and it is called one time high resolution mode. The measurement time in this mode is around 120ms. The register value for this scanning mode is 0x20.

Fifth measurement mode starts with lux values up to 0.5lx and it is called one time high resolution mode. The measurement time in this mode is around 120ms. The register value for this scanning mode is 0x21.

Sixth measurement mode starts with lux values up to 1lx and it is called one time low resolution mode. The measurement time in this mode is around 120ms. The register value for this scanning mode is 0x23.

The device is automatically set to *Power Down* mode after measurement.

Next, the I2C bus interface is initialized with the following command:

```
bus = smbus.SMBus(1)
```

Then, the function called *convertToNumber()* is created. The function converts the 2 bytes of data into a decimal number and it rounds the decimal number.

Next, the function called *readLight()* is created. This function has one argument and returns a double value. The argument is the I2C address of the sensor and return value is lux value which is double value.

Az-Delivery

Next, the *try-except* block of code is created. In the *try* block of code the indefinite loop is created. Here is the algorithm which reads the data from the sensor and displays that in the terminal. Between two loops of the indefinite loop there is 1 second pause.

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

Az-Delivery

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>