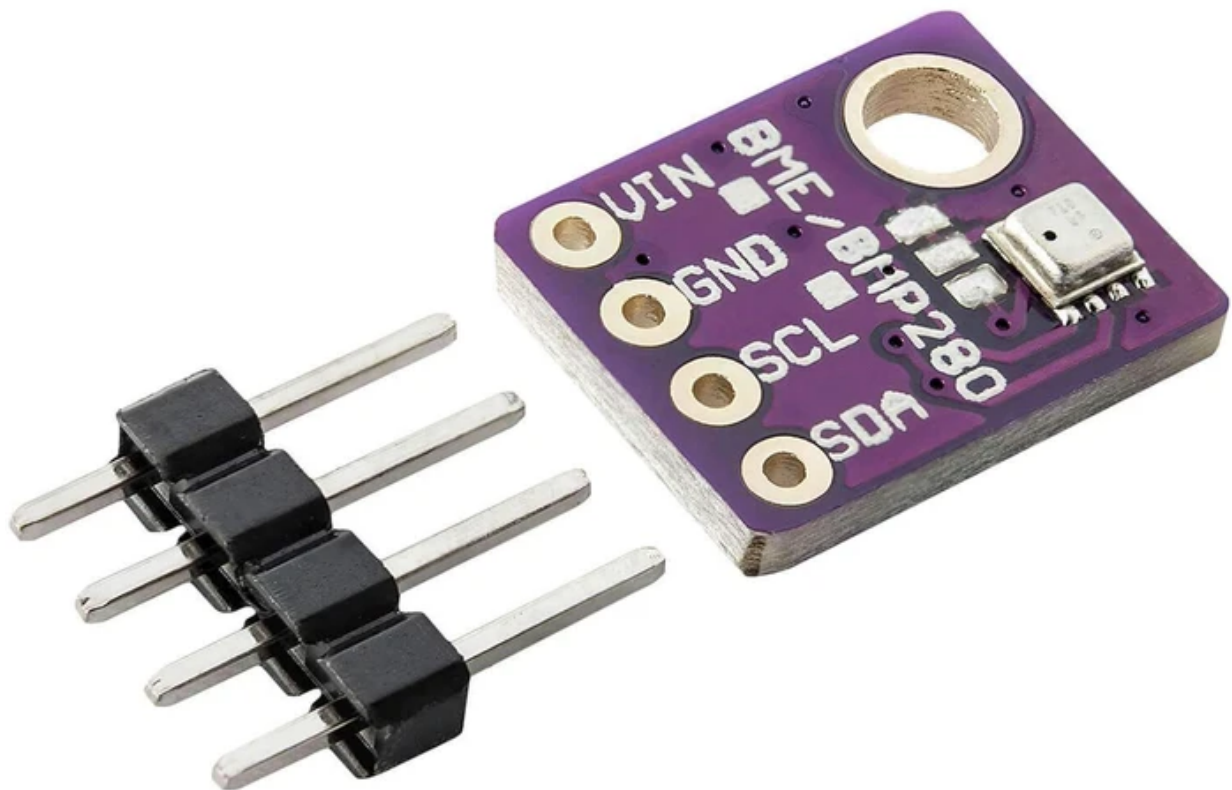


Az-Delivery

Welcome!

Thank you for purchasing our *AZ-Delivery BME280 temperature, humidity and air pressure sensor*. On the following pages, you will be introduced to how to use and set-up this handy device.

Have fun!



Areas of application

Education and teaching: Use in schools, universities and training institutions to teach the basics of electronics, programming and embedded systems. Research and development: Use in research and development projects to create prototypes and experiments in the fields of electronics and computer science. Prototype development: Use in the development and testing of new electronic circuits and devices. Hobby and Maker Projects: Used by electronics enthusiasts and hobbyists to develop and implement DIY projects.

Required knowledge and skills

Basic understanding of electronics and electrical engineering. Knowledge of programming, especially in the C/C++ programming language. Ability to read schematics and design simple circuits. Experience working with electronic components and soldering.

Operating conditions

The product may only be operated with the voltages specified in the data sheet to avoid damage. A stabilized DC power source is required for operation. When connecting to other electronic components and circuits, the maximum current and voltage limits must be observed to avoid overloads and damage.

Environmental conditions

The product should be used in a clean, dry environment to avoid damage caused by moisture or dust. Protect the product from direct sunlight (UV)

Intended Use

The product is designed for use in educational, research and development environments. It is used to develop, program and prototype electronic projects and applications. The Sensor product is not intended as a finished consumer product, but rather as a tool for technically savvy users, including engineers, developers, researchers and students.

Improper foreseeable use

The product is not suitable for industrial use or safety-relevant applications. Use of the product in medical devices or for aviation and space travel purposes is not permitted

disposal

Do not discard with household waste! Your product is according to the European one Directive on waste electrical and electronic equipment to be disposed of in an environmentally friendly manner. The valuable raw materials contained therein can be recycled become. The application of this directive contributes to environmental and health protection. Use the collection point set up by your municipality to return and Recycling of old electrical and electronic devices. WEEE Reg. No.: DE 62624346

electrostatic discharge

Attention: Electrostatic discharges can damage the product. Note: Ground yourself before touching the product, such as by wearing an anti-static wrist strap or touching a grounded metal surface.

safety instructions

Although our product complies with the requirements of the RoHS Directive (2011/65/EU) and does not contain any hazardous substances in quantities above the permitted limits, residues may still be present. Observe the following safety instructions to avoid chemical hazards: Caution: Soldering can produce fumes that can be harmful to health. Note: Use a solder fume extractor or work in a well-ventilated area. If necessary, wear a respirator mask. Caution: Some people may be sensitive to certain materials or chemicals contained in the product. Note: If skin irritation or allergic reactions occur, stop use and, if necessary, consult a doctor. Caution: Keep the product out of the reach of children and pets to avoid accidental contact and swallowing of small parts. Note: Store the product in a safe, closed container when not in use. Attention: Avoid contact of the product with food and drinks. Note: Do not store or use the product near food to prevent contamination. Although our product complies with the requirements of the RoHS Directive (2011/65/EU) and does not contain any hazardous substances in quantities above the permitted limits, residues may still be present. Observe the following safety instructions to avoid chemical hazards: Caution: Soldering can produce fumes that can be harmful to health. Note: Use a solder fume extractor or work in a well-ventilated area. If necessary, wear a respirator mask. Caution: Some people may be sensitive to certain materials or chemicals contained in the product. Note: If skin irritation or allergic reactions occur, stop use and, if necessary,

consult a doctor. Caution: Keep the product out of the reach of children and pets to avoid accidental contact and swallowing of small parts. Note: Store the product in a safe, closed container when not in use. Attention: Avoid contact of the product with food and drinks. Note: Do not store or use the product near food to prevent contamination. The product contains sensitive electronic components and sharp edges. Improper handling or assembly can result in injury or damage. Observe the following safety instructions to avoid mechanical hazards: Attention: The product's circuit board and connectors may have sharp edges. Use caution to avoid cuts. Note: Wear appropriate protective gloves when handling and assembling the product. Caution: Avoid excessive pressure or mechanical stress on the board and components. Note: Only mount the product on stable and flat surfaces. Use appropriate spacers and housings to minimize mechanical stress. Attention: Make sure the product is securely fastened to prevent accidental slipping or falling. Note: Use appropriate support or secure mounting in enclosures or on mounting plates. Caution: Make sure all cable connections are connected securely and correctly to avoid strain and accidental unplugging. Note: Route cables so that they are not under tension and do not pose a tripping hazard. The product operates with electrical voltages and currents that, if used improperly, can result in electric shocks, short circuits or other hazards. Observe the following safety instructions to avoid electrical hazards: Attention: Use the product only with the specified voltages. Note: The performance limits of the product can be found in the associated data sheet Caution: Avoid short circuits between the connectors and components of the product Note: Make sure that no conductive objects touch or bridge the circuit board. Use insulated tools and pay attention to the arrangement of connections. Caution: Do not perform any work on the product when it is connected to a power source. Note: Disconnect the product from power before making any circuit changes or connecting or removing components. Caution: Do not exceed the specified current ratings for the product's inputs and outputs. Note: The performance limits of the product can be found in the technical specifications or in the data sheet Attention: Make sure that the power sources used are stable and correctly sized. Note: Only use tested and suitable power supplies to avoid voltage fluctuations and overloads. Attention: Maintain sufficient distance from live parts to avoid accidental contact. Note: Ensure that the cabling is arranged safely and clearly according to the voltage used. Caution: Use insulating housings or protective covers to protect the product from direct contact. Note: Place the product in a non-conductive case to avoid accidental touching and short circuits. The product and the components on it may become warm during operation. Improper handling or overloading the product can result in burns, damage or fire. Observe the following safety instructions to avoid thermal hazards: Caution: Make sure the product is used within recommended operating temperatures. Note: The recommended operating temperature range is typically between -40°C and +85°C. Check the specific information in the product data sheet. Attention: Do not place the product near external heat sources such as radiators or direct sunlight. Note: Ensure that the product is operated in a cool and well-ventilated area. Attention: Make sure the product is well ventilated to avoid overheating. Note: Use fans or heat sinks when operating the product in a closed enclosure or in an environment with limited air circulation. Attention: Mount the product on heat-resistant surfaces and in heat-resistant housings. Note: Use enclosure materials that can withstand high temperatures to avoid damage or fire hazard. Caution: Implement temperature monitoring when using an enclosure and, if necessary, protection mechanisms that shut down the product if it overheats. Note: Note: Use temperature sensors and appropriate software to monitor the temperature of the product and shut down the system if necessary. Caution: Avoid overloads that can cause excessive heating of components. Note: To prevent overheating, do not exceed the specified current and voltage limits. Caution: Short circuits can generate significant heat and cause fires. Note: Make sure that all connections are correct and secure and that no conductive objects can accidentally cause short circuits.



Table of Contents

| | |
|---|----|
| Introduction..... | 3 |
| Specifications..... | 4 |
| How to set-up Arduino IDE..... | 5 |
| How to set-up the Raspberry Pi and Python..... | 9 |
| The pinout..... | 10 |
| Connecting the module with microcontroller..... | 11 |
| Arduino IDE library..... | 12 |
| Sketch example..... | 13 |
| Connecting the sensor with Raspberry Pi..... | 17 |
| Enabling the I2C interface..... | 18 |
| Python scripts..... | 20 |



Introduction

The BME280 sensor is a digital barometric sensor in one small package. The sensors consists of temperature, humidity and pressure sensors combined. The BME280 sensor can be used in a variety of applications such as home automation for heating and air conditioning, health monitoring devices, navigation systems, weather stations, handset devices, IoT and many other applications. Compact design and low power consumption are beneficial for portability and battery powered devices. High accuracy and fast response time make it a perfect candidate for extending the functionality of many other devices of choice.

The BME280 supports the I2C serial interface. The sensor has the predefined I2C address, which is `0x76`. The I2C address can be changed to value `0x77`, which is not covered in this eBook.

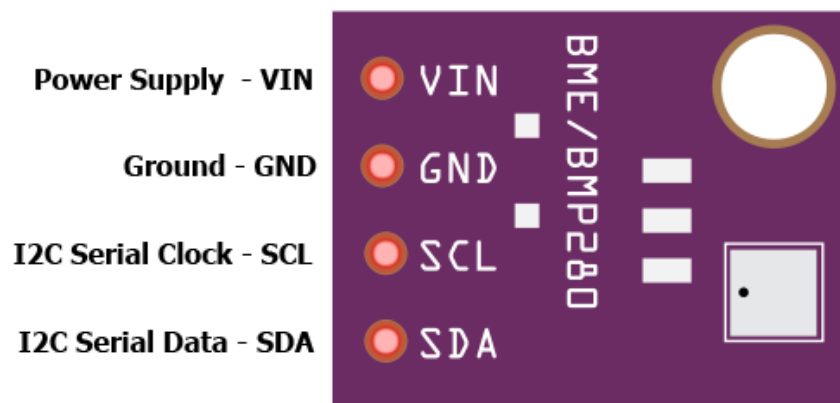
Current consumption is under $1mA$, when in measurement mode, and $5\mu A$ when it is in idle mode.

Specifications

| | |
|----------------------------|--|
| » Operating voltage range: | from 3.3V to 5V DC |
| » Current consumption: | < 1mA |
| » Temperature range: | from -40°C to 85 °C |
| » Temperature accuracy: | ±1.0°C |
| » Pressure range: | from 300 to 1100 hPa |
| » Pressure accuracy: | ±1hPa |
| » Humidity range: | from 0 to 100% RH |
| » Humidity accuracy: | ±3% |
| » Dimensions: | 9 x 11 x 2mm [0.35 x 0.43 x 0.078inch] |

The pinout

The BME280 sensor has four pins. The pinout is shown on the following image:



The sensor has an on-board LM6206 3.3V voltage regulator and a voltage level translator. The pins of the BME280 sensor can operate on voltages in range from 3.3V or 5V without a danger for the sensor itself.

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



The screenshot shows the Arduino IDE download page. On the left, there is a teal circle containing the Arduino logo (an infinity symbol with a minus sign on the left and a plus sign on the right). To the right of the logo, the text reads: **ARDUINO 1.8.9**, followed by a paragraph describing it as open-source software that runs on Windows, Mac OS X, and Linux. Below this, it states that the software can be used with any Arduino board and refers to the 'Getting Started' page for installation instructions. On the right side of the page, there is a teal sidebar with links for different operating systems: **Windows** (Installer for Windows XP and up, and ZIP file for non admin install), **Windows app** (Requires Win 8.1 or 10, with a 'Get' button), **Mac OS X** (10.8 Mountain Lion or newer), **Linux** (32 bits, 64 bits, ARM 32 bits, and ARM 64 bits), and links for **Release Notes**, **Source Code**, and **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`.

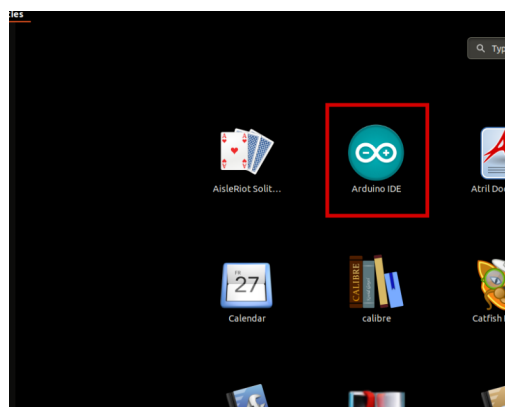
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

```
sh arduino-linux-setup.sh user_name
```

user_name - is the name of a superuser in 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` script has to be used 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.



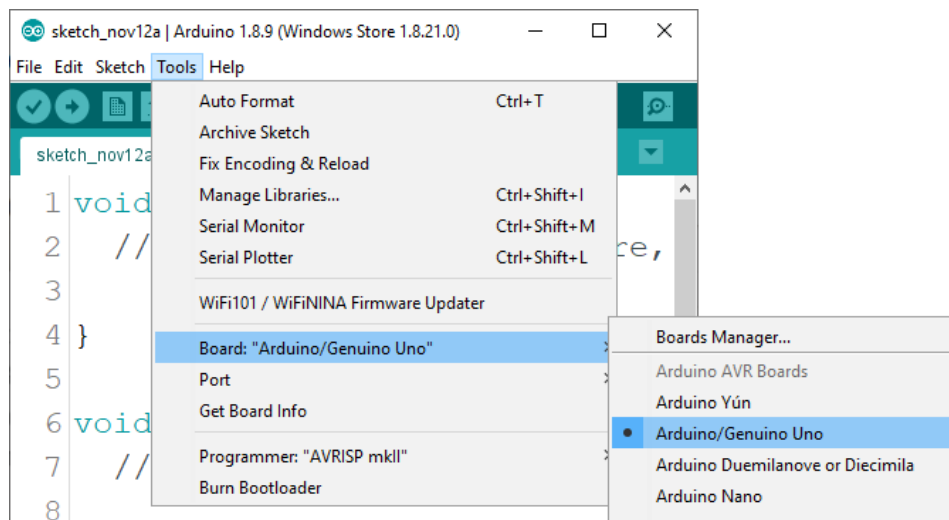
Az-Delivery

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:

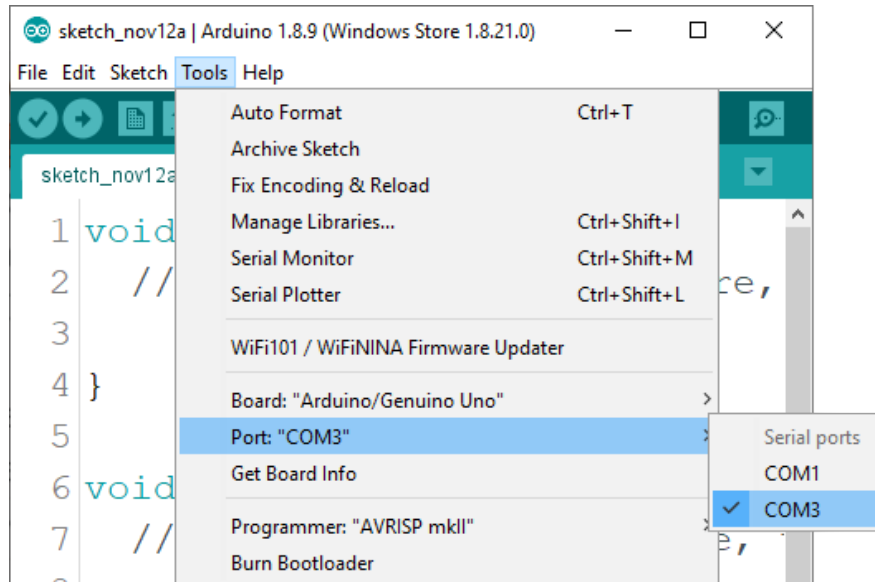


The port to which the microcontroller board is connected has to be selected.

Go to: *Tools > Port > {port name goes here}*

and when the microcontroller 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

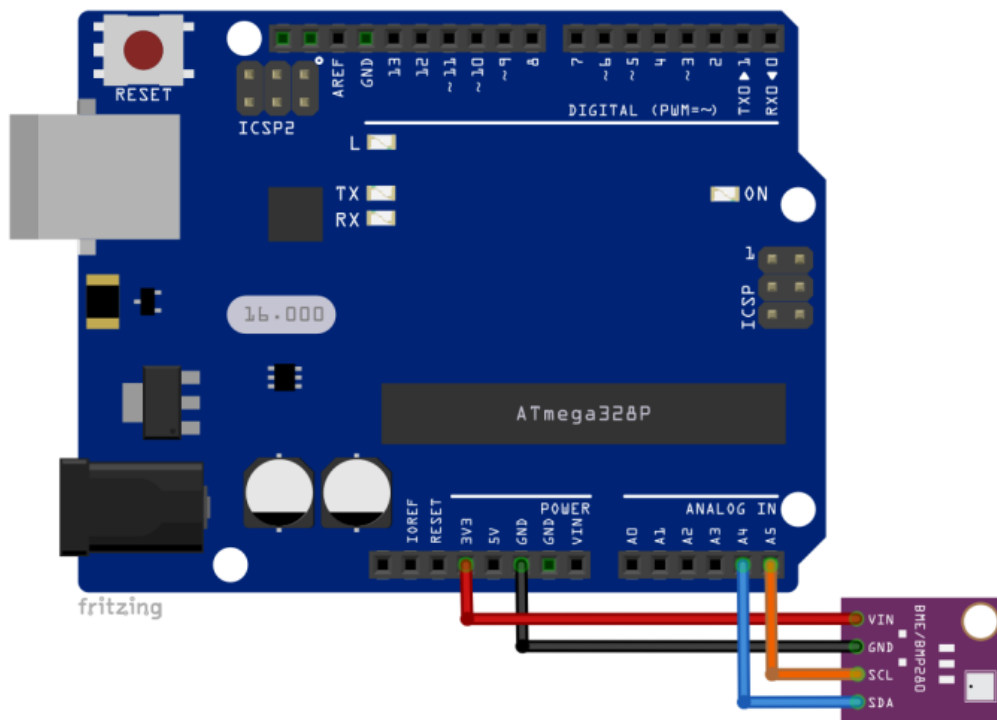
For the Raspberry Pi, first the operating system has to be installed, then everything has to be set-up so that it can be used in the *Headless* mode. The *Headless* mode enables remote connection to the Raspberry Pi, without the need for a *PC* screen Monitor, mouse or keyboard. The only things that 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 microcontroller

Connect the BME280 sensor with the microcontroller as shown on the following connection diagram:



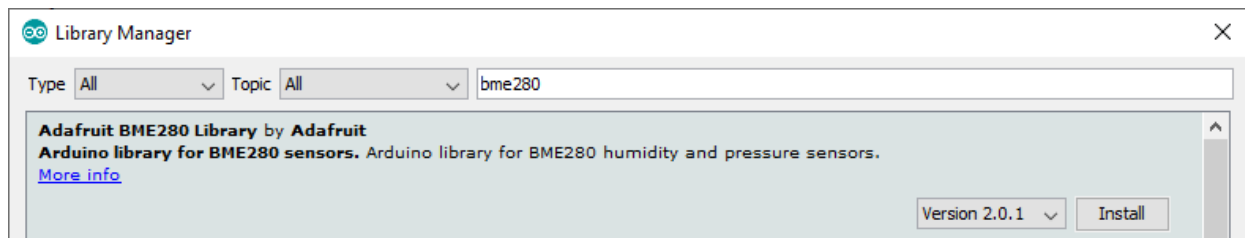
| BME280 pin | MC pin | Wiring color |
|------------|--------|--------------|
| VIN | 3.3V | Red wire |
| GND | GND | Black wire |
| SCL | A5 | Orange wire |
| SDA | A4 | Blue wire |

Library for Arduino IDE

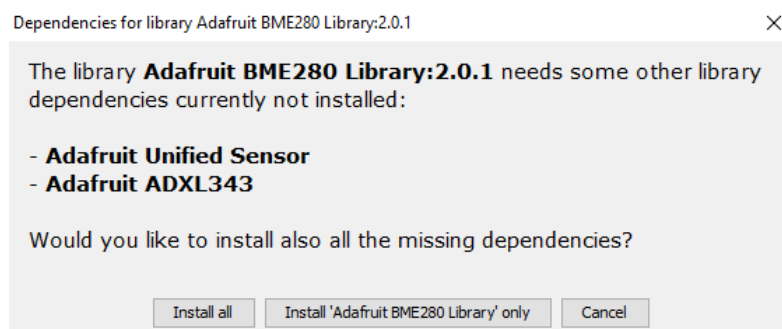
To use the sensor with microcontroller it is recommended to download an external library for it. The library that is used in this eBook is called *Adafruit BME280*. To download and install it, open Arduino IDE and go to:

Tools > Manage Libraries

When a new window opens, type *BME280* in the search box and install the library called *Adafruit BME280 Library* made by *Adafruit* as shown on the following image:



When the *Install* button is clicked, the prompt to install some additional libraries is shown, like on the following image:



Click *Install all* to finish the installation of the *Adafruit BME280* library.

Az-Delivery

Sketch example

The following sketch example is modified sketch from the *Adafruit BME280* library:

File > Examples > Adafruit BME280 Library > bme280test

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
Adafruit_BME280 bme; // I2C

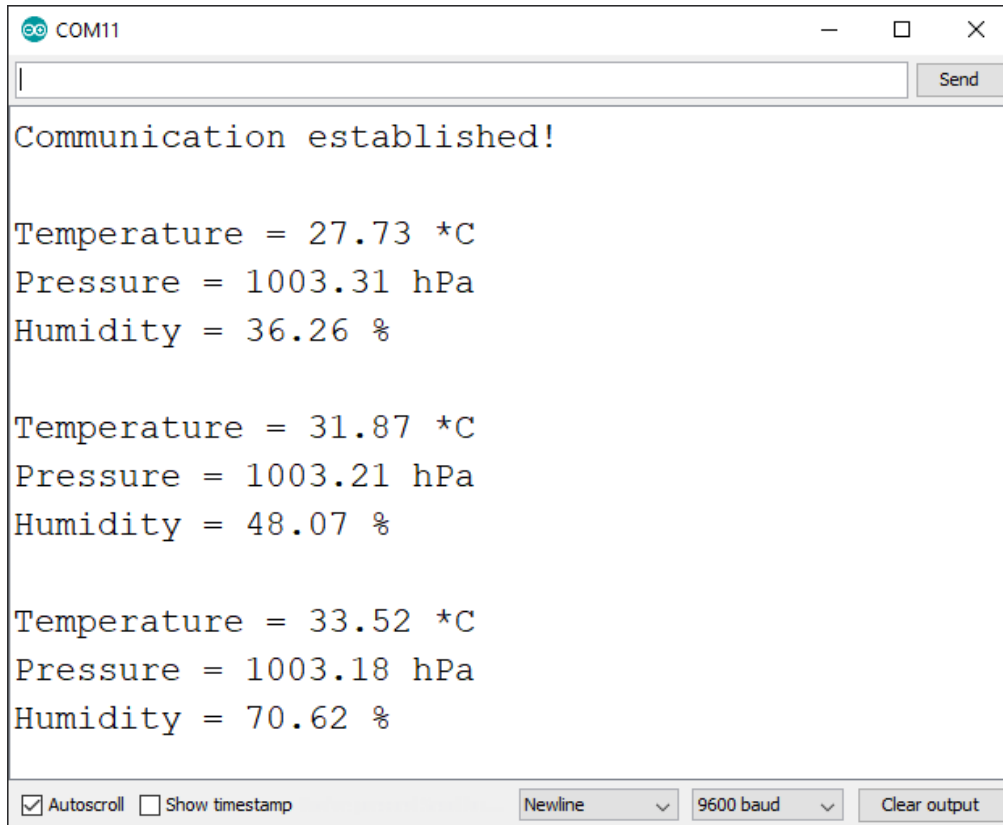
void setup() {
  Serial.begin(9600);
  // default address from library is 0x77
  // bool communication = bme.begin();
  bool communication = bme.begin(0x76);
  if (!communication) {
    Serial.println("Could not find a valid BME280 sensor");
    Serial.println("check wiring, address, sensor ID!");
    Serial.print("SensorID was: 0x");
    Serial.println(bme.sensorID(), 16);
    Serial.println("ID of 0xFF probably means a bad address\n");
    while (true) { };
    delay(10);
  }
  else {
    Serial.println("Communication established!\n");
  }
}
```

Az-Delivery

```
void loop() {  
  Serial.print("Temperature = ");  
  Serial.print(bme.readTemperature());  
  Serial.println(" *C");  
  Serial.print("Pressure = ");  
  Serial.print(bme.readPressure() / 100.0F);  
  Serial.println(" hPa");  
  Serial.print("Humidity = ");  
  Serial.print(bme.readHumidity());  
  Serial.println(" %\n");  
  delay(1000);  
}
```


Az-Delivery

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



Az-Delivery

The sketch begins with including three libraries: *Wire*, *Adafruit_Sensor* and *Adafruit_BME280*.

Next, the object called *bme* is created with the following line of code:
`Adafruit_BME280 bme;`

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

Then, the *bme* object is initialized with the following line of the code:
`bme.begin(0x76)`

where *0x76* is the I2C address of the sensor.

The *begin()* function returns a *boolean* value, which show if the initialization is successful or not. This value is stored in the variable called *communication*, with the following line of the code:

```
bool communication = bme.begin(0x76);
```

At the end of the *setup()* function, the success of the initialization is checked. If it is successful, the message *Communication established* is displayed in the Serial Monitor. If the initialization was not successful the error data is displayed in the Serial Monitor.

Az-Delivery

In the `loop()` function the temperature, pressure and humidity data is read with the following lines of code:

```
bme.readTemperature()  
bme.readPressure() / 100.0F  
bme.readHumidity()
```

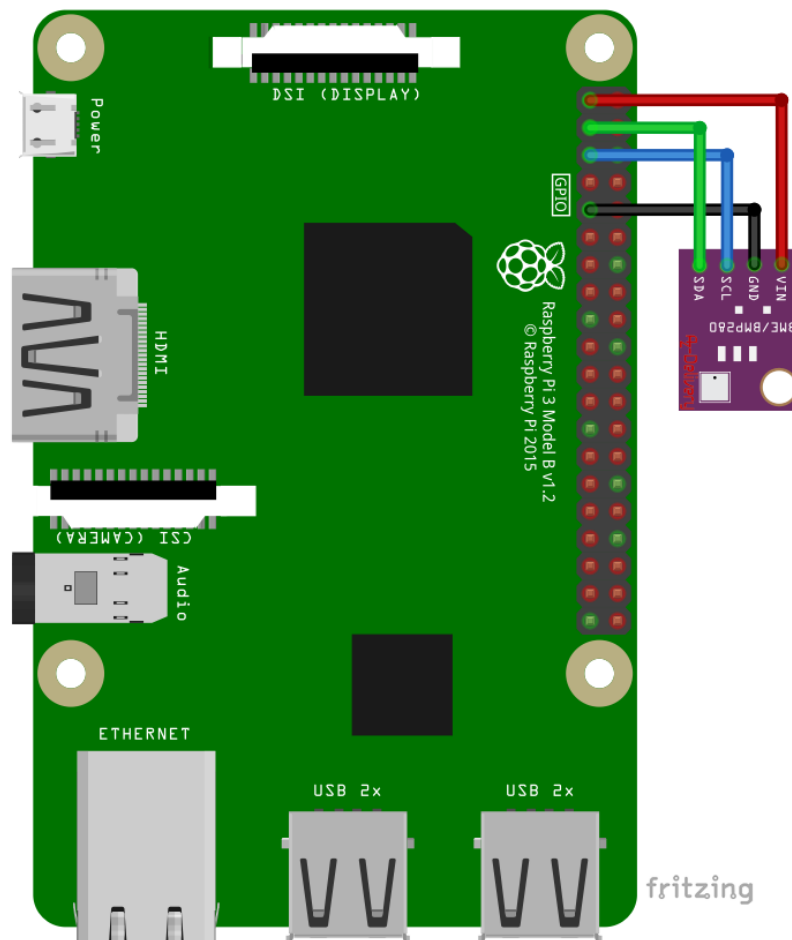
After that the data is displayed in Serial Monitor with following lines of code:

```
Serial.print(bme.readTemperature());  
Serial.print(bme.readPressure() / 100.0F);  
Serial.print(bme.readHumidity());
```

There is a one second pause between two measurements at the end of the `loop()` function: `delay(1000);`

Connecting the sensor with Raspberry Pi

Connect the BME280 sensor with the Raspberry Pi as shown on the following connection diagram:

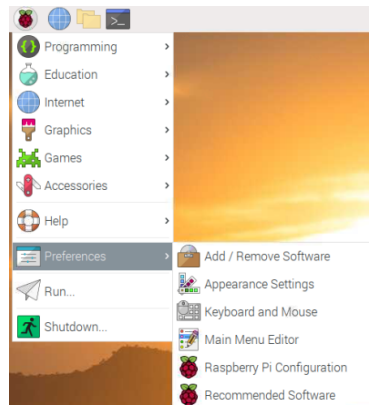


| BME280 pin | Raspberry Pi pin | Physical pin No. | Wire color |
|------------|------------------|------------------|------------|
| GND | GND | 9 | Black wire |
| VIN | 3V3 | 1 | Red wire |
| SCL | GPIO3 | 5 | Blue wire |
| SDA | GPIO2 | 3 | Green wire |

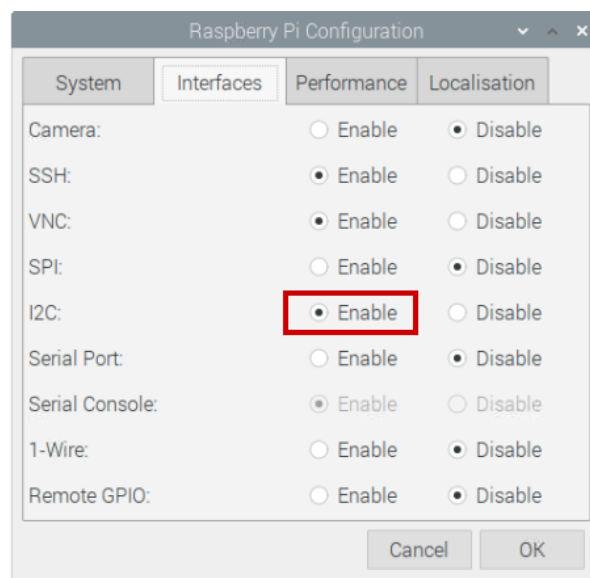
Enabling the I2C interface

In order to use the module with Raspberry Pi, the I2C interface has to be enabled. Open following menu:

Application Menu > Preferences > Raspberry Pi Configuration



In the new window, under the tab *Interfaces*, enable the I2C radio button, as shown on the following image:



Az-Delivery

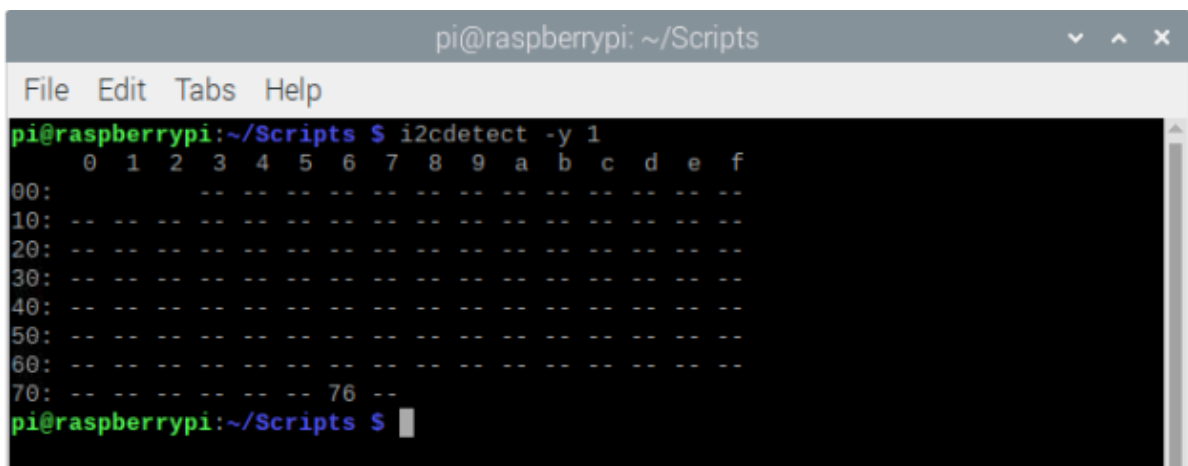
To detect the I2C address of the sensor connected to the I2C interface of the Raspberry Pi, the i2c-tools tool has to be installed, if not, open the terminal and run the following command:

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

To detect the I2C address, open terminal and run the following command:

```
i2cdetect -y 1
```

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

A terminal window titled 'pi@raspberrypi: ~/Scripts' showing the command 'i2cdetect -y 1' and its output. The output is a grid where the first row contains hexadecimal digits 0-9 and a-f. Subsequent rows are labeled from 00: to 70: on the left. Most cells in the grid contain dashes '--'. The cell at row 70 and column '6' (representing address 0x76) contains the number '76'. The prompt 'pi@raspberrypi:~/Scripts \$' is visible at the bottom.

```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ i2cdetect -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- 76 --
pi@raspberrypi:~/Scripts $
```

Where 0x76 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:

A terminal window titled 'pi@raspberrypi: ~/Scripts' 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/1': No such file or directory'. The prompt 'pi@raspberrypi:~/Scripts \$' is visible at the bottom.

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



Python scripts

Two scripts are created, one for all functions and the other for using these functions, because of a better readability. The code for the first script is the following:

```
import smbus
import time
from ctypes import c_short
from ctypes import c_byte
from ctypes import c_ubyte

DEVICE = 0x76 # Default device I2C address
bus = smbus.SMBus(1) # Rev 2 Pi, Pi 2 & Pi 3 uses bus 1
                        # Rev 1 Pi uses bus 0

def getShort(data, index):
    # return two bytes from data as a signed 16-bit value
    return c_short((data[index+1] << 8) + data[index]).value

def getUShort(data, index):
    # return two bytes from data as an unsigned 16-bit value
    return (data[index + 1] << 8) + data[index]

def getChar(data, index):
    # return one byte from data as a signed char
    result = data[index]
    if result > 127:
        result -= 256
    return result
```

Az-Delivery

```
def getUChar(data, index):
    # return one byte from data as an unsigned char
    result = data[index] & 0xFF
    return result

def readBME280ID(addr=DEVICE):
    # Chip ID Register Address
    REG_ID = 0xD0
    (chip_id, chip_version) = bus.read_i2c_block_data(addr, REG_ID, 2)
    return (chip_id, chip_version)

def readBME280All(addr=DEVICE):
    # Register Addresses
    REG_DATA = 0xF7
    REG_CONTROL = 0xF4
    REG_CONFIG = 0xF5
    REG_CONTROL_HUM = 0xF2
    REG_HUM_MSB = 0xFD
    REG_HUM_LSB = 0xFE
    # Oversample setting
    OVERSAMPLE_TEMP = 2
    OVERSAMPLE_PRES = 2
    MODE = 1
    # Oversample setting for humidity register
    OVERSAMPLE_HUM = 2
    bus.write_byte_data(addr, REG_CONTROL_HUM, OVERSAMPLE_HUM)
    control = OVERSAMPLE_TEMP << 5 | OVERSAMPLE_PRES << 2 | MODE
    bus.write_byte_data(addr, REG_CONTROL, control)
    # Read blocks of calibration data from EEPROM
    cal1 = bus.read_i2c_block_data(addr, 0x88, 24)
    cal2 = bus.read_i2c_block_data(addr, 0xA1, 1)
    cal3 = bus.read_i2c_block_data(addr, 0xE1, 7)
```


Az-Delivery

```
# one tab
# Convert byte data to word values
dig_T1 = getUShort(cal1, 0)
dig_T2 = getShort(cal1, 2)
dig_T3 = getShort(cal1, 4)
dig_P1 = getUShort(cal1, 6)
dig_P2 = getShort(cal1, 8)
dig_P3 = getShort(cal1, 10)
dig_P4 = getShort(cal1, 12)
dig_P5 = getShort(cal1, 14)
dig_P6 = getShort(cal1, 16)
dig_P7 = getShort(cal1, 18)
dig_P8 = getShort(cal1, 20)
dig_P9 = getShort(cal1, 22)
dig_H1 = getUChar(cal2, 0)
dig_H2 = getShort(cal3, 0)
dig_H3 = getUChar(cal3, 2)
dig_H4 = getChar(cal3, 3)
dig_H4 = (dig_H4 << 24) >> 20
dig_H4 = dig_H4 | (getChar(cal3, 4) & 0x0F)
dig_H5 = getChar(cal3, 5)
dig_H5 = (dig_H5 << 24) >> 20
dig_H5 = dig_H5 | (getUChar(cal3, 4) >> 4 & 0x0F)
dig_H6 = getChar(cal3, 6)
# Wait in ms (Datasheet Appendix B: Measurement
# time and current calculation)
wait_time = 1.25 + (2.3 * OVERSAMPLE_TEMP) + ((2.3 *
    OVERSAMPLE_PRES) + 0.575) + ((2.3 * OVERSAMPLE_HUM) + 0.575)
time.sleep(wait_time / 1000) # Wait the required time
# Read temperature / pressure / humidity
data = bus.read_i2c_block_data(addr, REG_DATA, 8)
pres_raw = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4)
```

Az-Delivery

```
# one tab
temp_raw = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4)
hum_raw = (data[6] << 8) | data[7]
# Refine temperature
var1 = (((temp_raw >> 3) - (dig_T1 << 1))) * (dig_T2) >> 11
var2 = (((((temp_raw >> 4) - (dig_T1)) * ((temp_raw >> 4) -
        (dig_T1))) >> 12)*(dig_T3)) >> 14
t_fine = var1 + var2
temperature = float(((t_fine * 5) + 128) >> 8);
# Refine pressure and adjust for temperature
var1 = t_fine / 2.0 - 64000.0
var2 = var1 * var1 * dig_P6 / 32768.0
var2 = var2 + var1 * dig_P5 * 2.0
var2 = var2 / 4.0 + dig_P4 * 65536.0
var1 = (dig_P3 * var1 * var1 / 524288.0 + dig_P2 * var1) / 524288.0
var1 = (1.0 + var1 / 32768.0) * dig_P1
if var1 == 0:
    pressure = 0
else:
    pressure = 1048576.0 - pres_raw
    pressure = ((pressure - var2 / 4096.0) * 6250.0) / var1
    var1 = dig_P9 * pressure * pressure / 2147483648.0
    var2 = pressure * dig_P8 / 32768.0
    pressure = pressure + (var1 + var2 + dig_P7) / 16.0

# Refine humidity
humidity = t_fine - 76800.0
humidity = (hum_raw - (dig_H4 * 64.0 + dig_H5 / 16384.0 * humidity))
* (dig_H2 / 65536.0 * (1.0 + dig_H6 / 67108864.0 * humidity * (1.0 +
dig_H3 / 67108864.0 * humidity)))
humidity = humidity * (1.0 - dig_H1 * humidity / 524288.0)
```

Az-Delivery

```
# one tab
if humidity > 100:
    humidity = 100
elif humidity < 0:
    humidity = 0
return temperature / 100.0, pressure / 100.0, humidity
```

Save the script by the name *bme280.py*. The script code is modified from the [script](#).

Az-Delivery

The following is code for the main script:

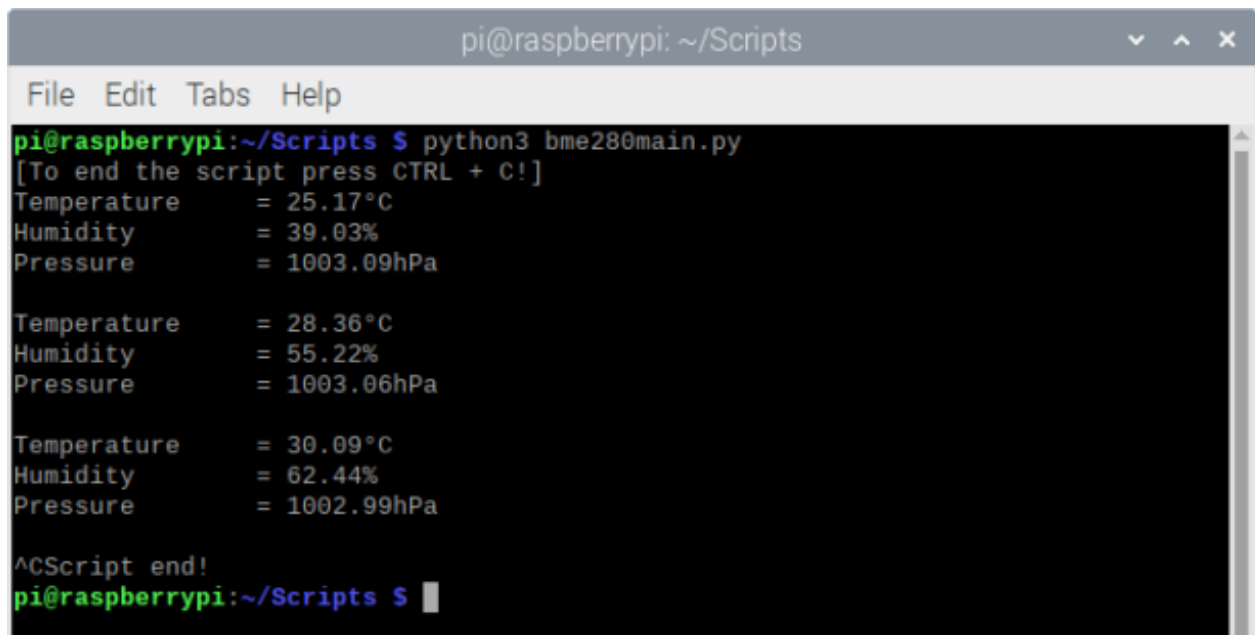
```
import bme280
from time import sleep
dgr = u'\xb0'
print('[Press CTRL + C to end the script!]\n')
try:
    while(True):
        temperature,pressure,humidity = bme280.readBME280All()
        print('Temperature = {}{}C'.format(temperature, dgr))
        print('Humidity = {:.2f}%'.format(humidity))
        print('Pressure = {:.2f}hPa\n'.format(pressure))
        sleep(1)
except KeyboardInterrupt:
    print('Script end!')
```

Az-Delivery

Save the script by the name *bme280main.py* into the same directory where you saved the *bme280.py* script. To run the main script open the terminal in the directory where the scripts are saved and run the following command:

```
python3 bme280main.py
```

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



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 bme280main.py
[To end the script press CTRL + C!]
Temperature      = 25.17°C
Humidity         = 39.03%
Pressure         = 1003.09hPa

Temperature      = 28.36°C
Humidity         = 55.22%
Pressure         = 1003.06hPa

Temperature      = 30.09°C
Humidity         = 62.44%
Pressure         = 1002.99hPa

^CScript end!
pi@raspberrypi:~/Scripts $
```

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

The first script is not explained in this eBook.

Az-Delivery

The *bme280main.py* script starts with importing *bme280* script and the *sleep* function from the *time* library.

Then, the *dgr* variable is created, where UTF degree sign value is stored.

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 the *readBME280All()* function is used to read the sensor data. This function returns tuple of three elements: *temperature*, *pressure* and *humidity* elements. Then, the data is displayed in the terminal. In the output, to round the floating point number to two decimal places after the decimal point, the following line of code is used:

```
print('Humidity = {:.2f}%'.format(humidity))
```

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

Az-Delivery

Now is the time to learn and make 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 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>