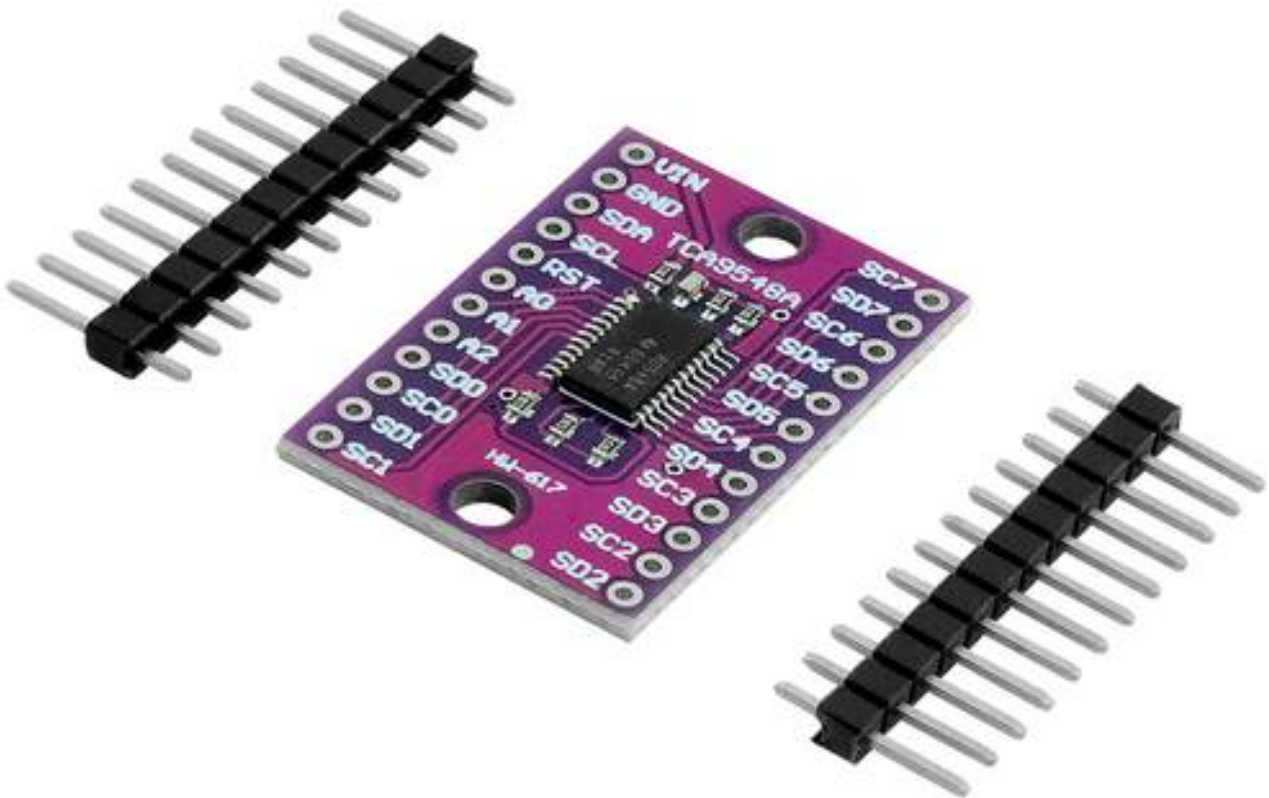


AZ-Delivery

Welcome!

Thank you for purchasing our *AZ-Delivery TCA9548A I2C Multiplexer Module*. On the following pages, you will be introduced to how to use and set-up this handy device.

Have fun!



Az-Delivery

Table of Contents

Introduction.....	3
Specifications.....	4
The pinout.....	7
Connection schematic.....	8
The connection schematic is shown on the following image:.....	8
How to set-up Arduino IDE.....	9
How to set-up the Raspberry Pi and Python.....	13
Connecting the modules with Atmega328p.....	14
Sketch examples.....	16
Connecting the modules with Raspberry Pi.....	18
Libraries and tools for Python.....	20
Enabling the I2C interface.....	21
Python script.....	23



Introduction

The TCA9548A I2C Multiplexer Module is a device made around TCA9548A chip that simplifies using multiple I2C devices with one microcontroller. It is an eight-channel (bidirectional) I2C multiplexer which allows eight separate I2C devices to be controlled by a single host on single I2C bus.

The TCA9548A chip has eight bidirectional translating switches that can be controlled through the I2C bus. The module can connect several different devices with a single microcontroller.

If some of the devices connected to the same channel have same I2C address, the issue can be solved by simply connecting to other channel, if the device does not have a configurable I2C address DIP switch or jumper.

Specifications

Operating voltage range	1.65V - 5.5V
Operating current	100mA (max.)
Input current	20mA
Output current	25mA
Input tolerance voltage	5V (max.)
Interface	I2C
I2C input channels	8
I2C output	1
Storage temperature	-65 - 150°C
Dimensions	22x32x2.7mm (0.8x1.2x0.1in)

The max of 8 of these multiplexers can be connected together on 0x70-0x77 addresses in order to control 64 of the same I2C addressed parts. By connecting the three address bits A0, A1 and A2 to VIN it can get different combination of the addresses. This is how an address byte of the TCA9548A looks like. First 7-bits combine to form the slave address. The last bit of the slave address defines the operation (read or write) to be performed. When it is high (1), a read is selected, while a low (0) selects a write operation.

NOTE: The TCA9584A module can also act as logic level converter; i.e. all I2C devices can have their own power supply/voltage. Further explanations of this functionality is not in the scope of this eBook.



Features

- I2C Bus and SMBus Compatible
- Active-Low Reset Input
- Three Address Pins, Allowing up to Eight TCA9548A Devices on the I2C Bus
- Channel Selection Through an I2C Bus, In Any Combination
- Power Up With All Switch Channels Deselected • Low RON Switches • Allows Voltage-Level Translation Between 1.8-V, 2.5-V, 3.3-V, and 5-V Buses
- No Glitch on Power Up • Supports Hot Insertion • Low Standby Current • Operating Power-Supply Voltage Range of 1.65 V to 5.5 V
- 5-V Tolerant Inputs • 0- to 400-kHz Clock Frequency • Latch-Up Performance Exceeds 100 mA Per JESD 78, Class II
- ESD Protection Exceeds JESD 22 – ± 2000 -V Human-Body Model (A114-A) – 200-V Machine Model (A115-A)
- ± 1000 -V Charged-Device Model (C101)

Changing the I2C address of the TCA9548A

The bus address of the TCA9548A is changed by using the connections to the A0, A1 and A2 pins. By default use of 0x70 address, by wiring A0-A2 to GND(LOW). Using the table below, it can be reconfigured to an address between 0x70 and 0x77 by matching the inputs to HIGH(5V) or LOW(GND):

INPUTS			I2C BUS SLAVE ADDRESS
A2	A1	A0	
L	L	L	112 (decimal), 70 (hexadecimal)
L	L	H	113 (decimal), 71 (hexadecimal)
L	H	L	114 (decimal), 72 (hexadecimal)
L	H	H	115 (decimal), 73 (hexadecimal)
H	L	L	116 (decimal), 74 (hexadecimal)
H	L	H	117 (decimal), 75 (hexadecimal)
H	H	L	118 (decimal), 76 (hexadecimal)
H	H	H	119 (decimal), 77 (hexadecimal)

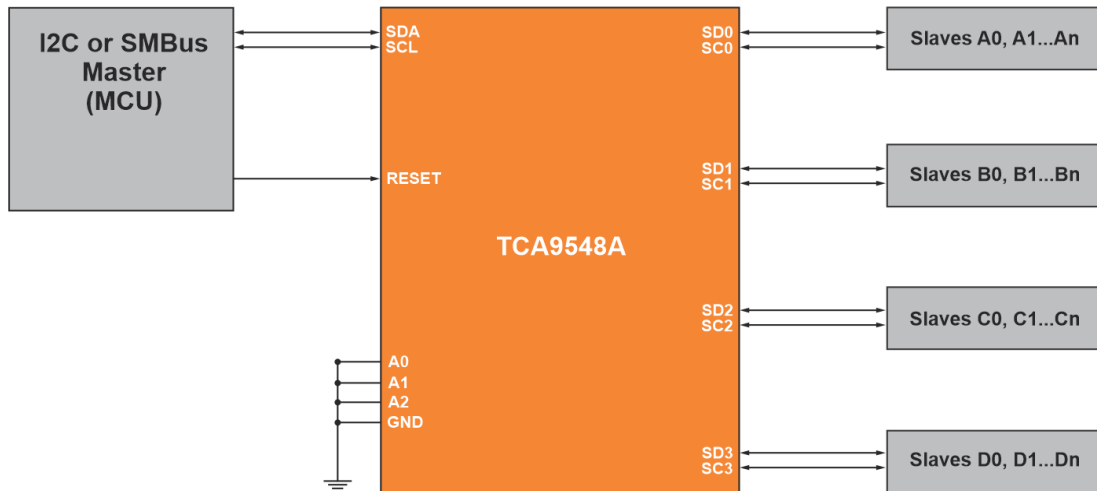
The pinout

The module has 24 pins. The pinout is shown on the following image:

Power supply - VIN	VIN	SC7	SC7 - I2C Serial Clock Line IN
GROUND - GND	GND	SD7	SD7 - I2C Serial Data Line IN
I2C Serial Data Line - SDA	SDA	SC6	SC6 - I2C Serial Clock Line IN
I2C Serial Clock Line - SCL	SCL	SD6	SD6 - I2C Serial Data Line IN
Reset - RST	RST	SC5	SC5 - I2C Serial Clock Line IN
Address Select - A0	A0	SD5	SD5 - I2C Serial Data Line IN
Address Select - A1	A1	SC4	SC4 - I2C Serial Clock Line IN
Address Select - A2	A2	SD4	SD4 - I2C Serial Data Line IN
I2C Serial Data Line IN - SD0	SD0	SC3	SC3 - I2C Serial Clock Line IN
I2C Serial Clock Line IN - SC0	SC0	SD3	SD3 - I2C Serial Data Line IN
I2C Serial Data Line IN - SD0	SD1	SC2	SC2 - I2C Serial Clock Line IN
I2C Serial Clock Line IN - SC1	SC1	SD2	SD2 - I2C Serial Data Line IN

Connection schematic

The connection schematic is shown on the following image:

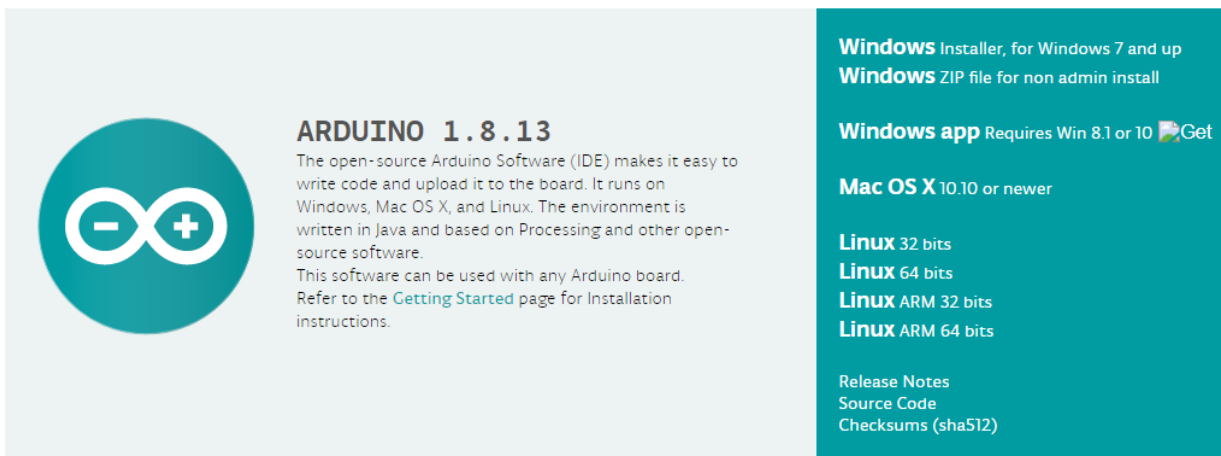


The schematic represents example of multiple devices connected to different channels.

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 **1.8.13**.

Download the Arduino IDE



The screenshot shows the Arduino 1.8.13 download page. On the left, there is a teal circle with a white infinity symbol. To its right, the text reads: **ARDUINO 1.8.13**. Below this, it says: "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." On the right side of the page, there is a teal sidebar with white text. It lists: **Windows** Installer, for Windows 7 and up; **Windows** ZIP file for non admin install; **Windows app** Requires Win 8.1 or 10; **Mac OS X** 10.10 or newer; **Linux** 32 bits; **Linux** 64 bits; **Linux** ARM 32 bits; **Linux** ARM 64 bits; [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`.

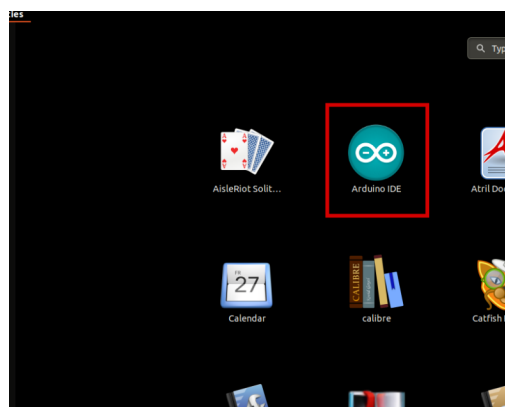
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

```
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`, 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.



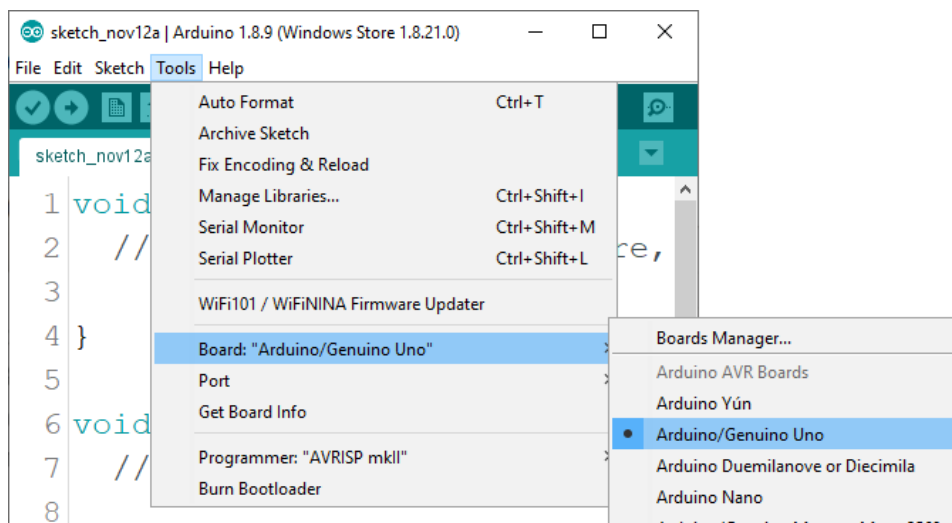
Az-Delivery

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 Atmega328p 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:

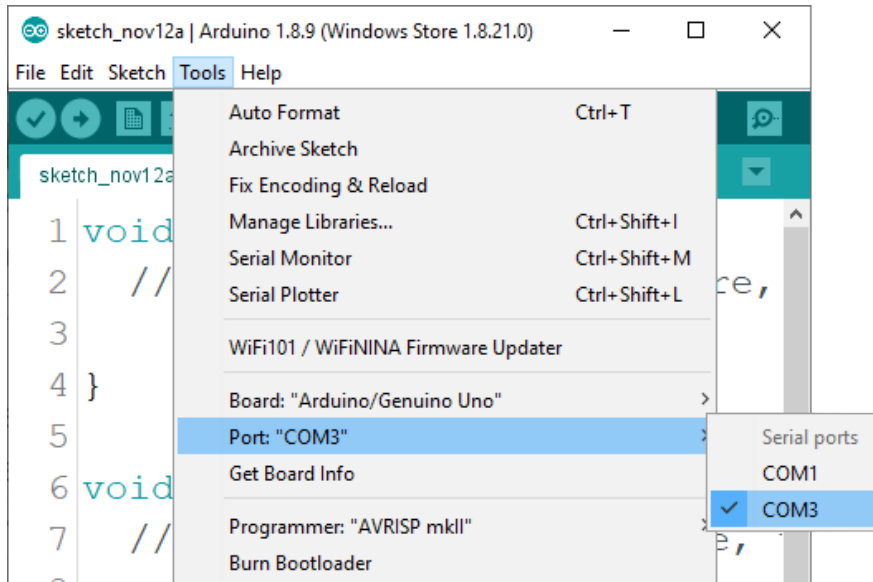


The port to which the Atmega328p board is connected has to be selected.

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

and when the Atmega328p 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 *Raspbery Pi OS* operating system comes with *Python* preinstalled.

BME280 pin	TCA9548A pin	Wire color
VIN	5V (Atmega328p 5V)	Red wire
GND	GND (Atmega328p GND)	Black wire
SDA	SD0	Blue wire
SCL	SC0	Green wire
TCA9548A pin	Mc pin	Wire color
VIN	5V	Red wire
GND	GND	Black wire
SDA	A4	Blue wire
SCL	A5	Green wire
OLED pin	TCA9548A pin	Wire color
SDA	SD1 (CH1)	Blue wire
SCL	SC1 (CH1)	Green wire
VCC	5V (Atmega328p pin)	Red wire
GND	GND (Atmega328p pin)	Black wire

Az-Delivery

Sketch examples

```
#include <U8g2lib.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

Adafruit_BME280 bme;

#define TCAADDR 0x70

U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);

void TCA9548A(uint8_t bus) {
  Wire.beginTransmission(0x70);
  Wire.write(1 << bus);
  Wire.endTransmission();
}

void u8g2_prepare() {
  u8g2.setFont(u8g2_font_6x10_tf);
  u8g2.setFontRefHeightExtendedText();
  u8g2.setDrawColor(1);
  u8g2.setFontPosTop();
  u8g2.setFontDirection(0);
}

void setup() {
  Wire.begin();
  Serial.begin (9600);
  TCA9548A(0);
  bool communication = bme.begin(0x76);
  TCA9548A(1);
  u8g2.begin();
  u8g2_prepare();
}
```

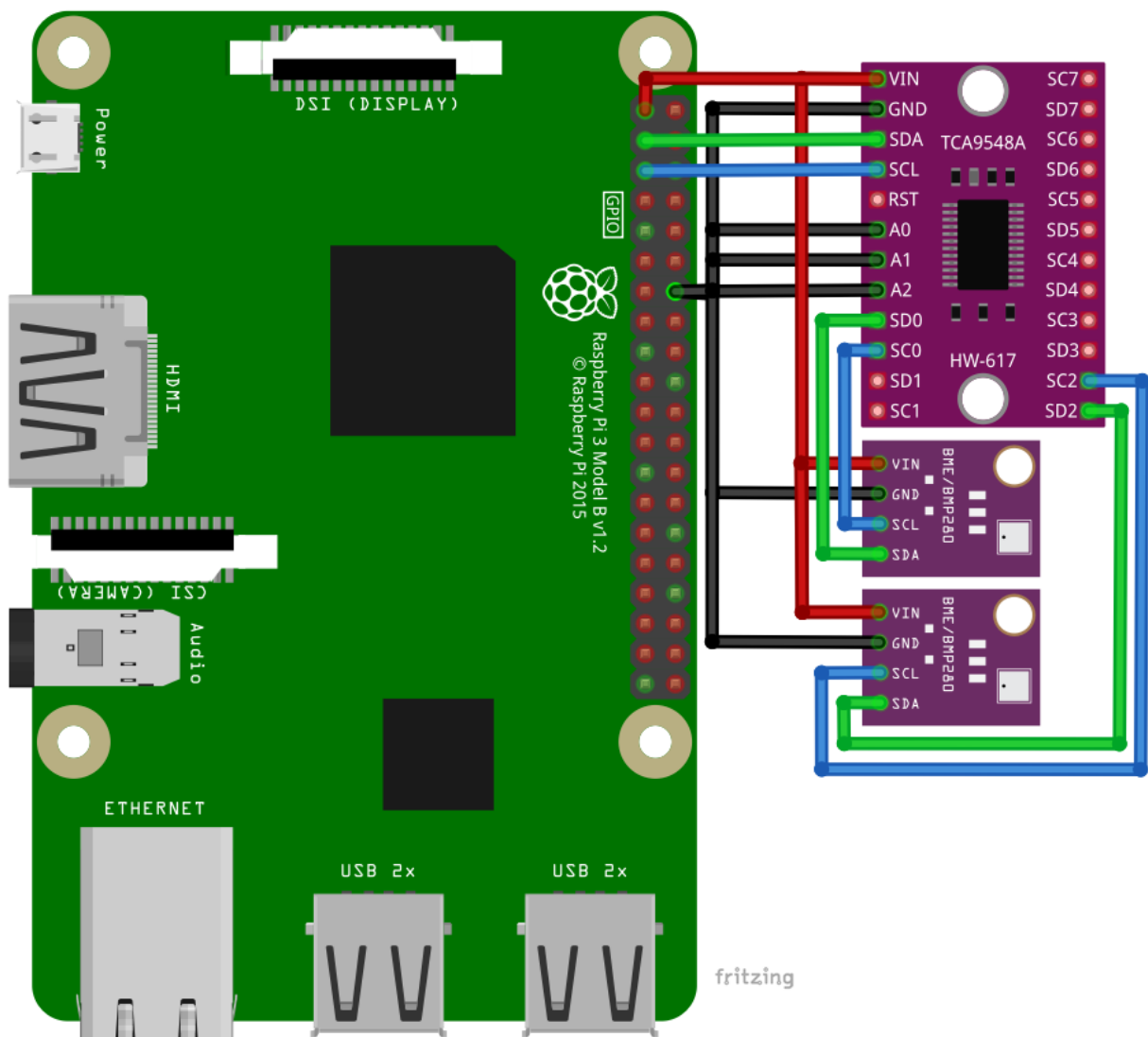
Az-Delivery

```
}
```

```
void loop(void) {  
    TCA9548A(0);  
    float temperature = bme.readTemperature();  
    float pressure = bme.readPressure()/100;  
    float humidity = bme.readHumidity();  
  
    TCA9548A(1);  
    u8g2.clearBuffer();  
    u8g2.prepare();  
    u8g2.setFont(u8g2_font_6x10_tf);  
    u8g2.drawStr(0, 0, "Temperature: ");  
    u8g2.setCursor(75, 0);  
    u8g2.print(temperature);  
    u8g2.println();  
    u8g2.drawStr(0, 10, "Pressure: ");  
    u8g2.setCursor(75, 10);  
    u8g2.print(pressure);  
    u8g2.println();  
    u8g2.drawStr(0, 20, "Humidity: ");  
    u8g2.setCursor(75, 20);  
    u8g2.print(humidity);  
    u8g2.println();  
    u8g2.sendBuffer();  
    delay(100);  
}
```

Connecting the modules with Raspberry Pi

Connect the modules with the Raspberry Pi as shown on the following image:



TCA9548A pin	Raspberry Pi pin	Physical pin	Wire color
VIN	3.3V	1	Red wire
GND	GND	14	Black wire
SDA	GPIO2	3	Green wire
SCL	GPIO3	5	Blue wire
A0, A1, A2	GND	14	Black wire
BME280(1) PIN	TCA9548A pin	Raspberry Pi pin	Wire color
SCL	SC0		Blue wire
SDA	SD0		Green wire
VIN	VIN	3.3V (PIN1)	Red wire
GND	GND	GND (PIN14)	Black wire
BME280(2) PIN	TCA9548A pin	Raspberry Pi pin	Wire color
SCL	SC1		Blue wire
SDA	SD1		Green wire
VIN	VIN	3.3V (PIN1)	Red wire
GND	GND	GND (PIN14)	Black wire



Libraries and tools for Python

To use the module with the Raspberry Pi, the library *RPi.GPIO* has to be installed. If the library is already installed, running the installation command only updates the library to a newer version.

To install the library, open the terminal and run the following commands, one by one:

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

```
sudo apt install python3-pip
```

```
sudo apt install git
```

```
git clone https://github.com/pimoroni/bme280-python
```

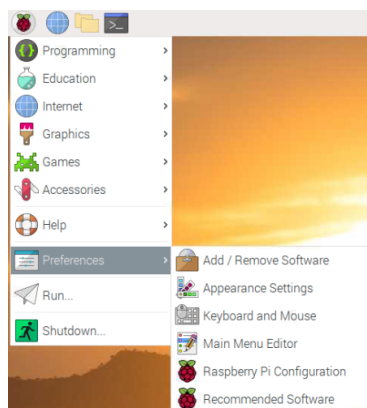
```
cd bme280-python
```

```
sudo ./install.sh
```

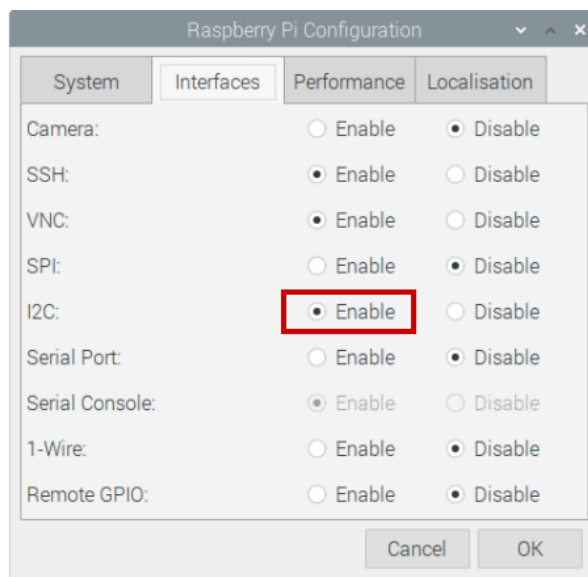
Enabling the I2C interface

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

Application Menu > Preferences > Raspberry Pi Configuration

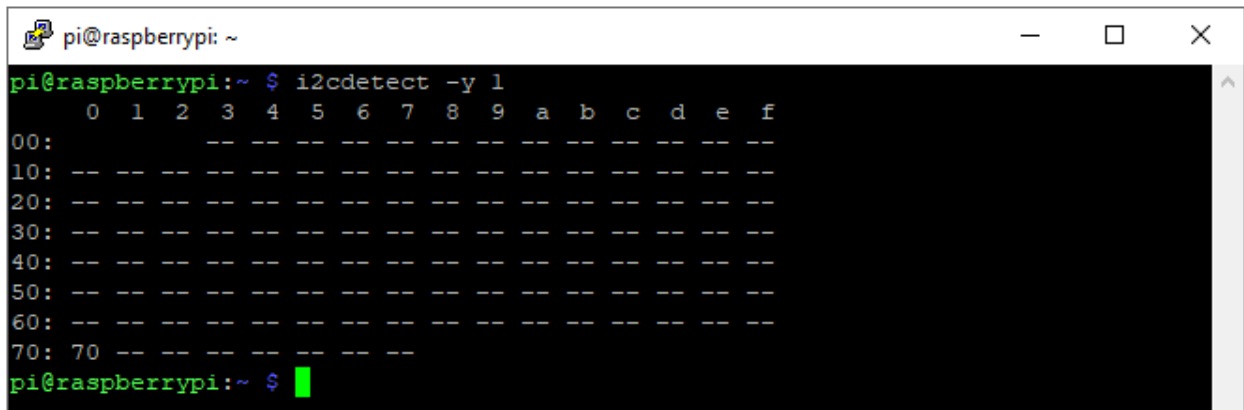


In the new window, under the tab *Interfaces*, enable the I2C radio button, as on the following image:



Before using any device connected to the I2C interface, I2C address has to be detected first. To detect the screen I2C address, following command should be run in the terminal: **i2cdetect -y 1**

The result should look like the following image:



```
pi@raspberrypi: ~  
pi@raspberrypi:~$ 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: 70 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
pi@raspberrypi:~$
```

Where $0x70$ is the I2C address of the multiplexer module. Other I2C addresses can have different values depending on where other devices are connected and to which channel.



Python script

```
from bme280 import BME280
import smbus
import os
import time

print('TCA9548A Multiplexer + BME280 script')

class multiplex:

    def __init__(self, bus):
        self.bus = smbus.SMBus(bus)

    def channel(self, address, channel):

        if (channel == 0):
            action = 0x01
        elif (channel == 1):
            action = 0x02
        elif (channel == 2):
            action = 0x04
        elif (channel == 3):
            action = 0x08
        elif (channel == 4):
            action = 0x10
        elif (channel == 5):
            action = 0x20
        elif (channel == 6):
            action = 0x40
        elif (channel == 7):
            action = 0x80
        else:
            action = 0x00
        self.bus.write_byte_data(address, 0x04, action)
```

Az-Delivery

```
def readdata():
    bme280 = BME280(i2c_dev=smbus.SMBus(1))
    print("readdata")
    temperature = bme280.get_temperature()
    pressure = bme280.get_pressure()
    humidity = bme280.get_humidity()
    time.sleep(2)
    temperature = bme280.get_temperature()
    pressure = bme280.get_pressure()
    humidity = bme280.get_humidity()
    print("Temperature: ", bme280.get_temperature())
    print("Pressure: ", bme280.get_pressure())
    print("Humidity: ", bme280.get_humidity())
    return(temperature, pressure, humidity)

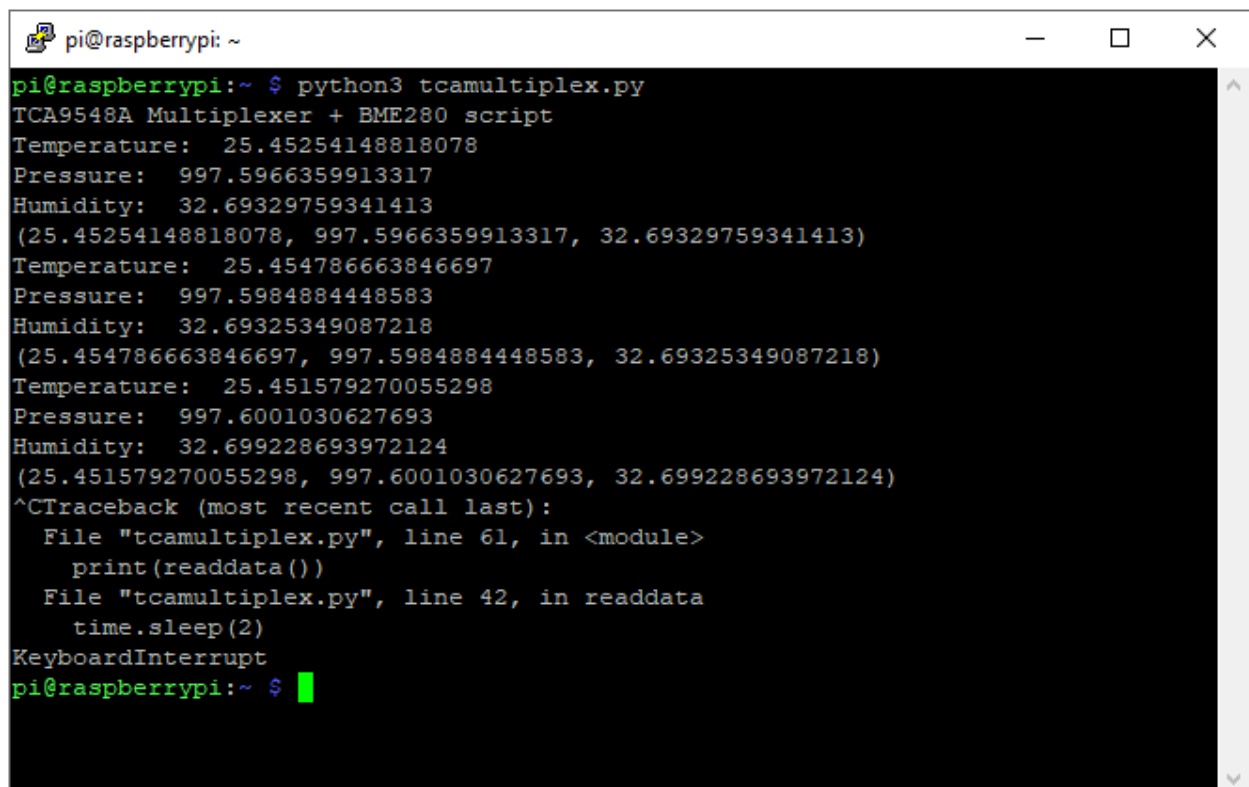
if __name__ == '__main__':
    bus = 1
    address = 0x70 # A0 to A2 NC or GND, change as required
    plexer = multiplex(bus)
    while True:
        plexer.channel(address, 0)
        print(readdata())

        plexer.channel(address, 2)
        print(readdata())
```

Az-Delivery

Save the script by the name *tcamultiplex.py*. To run the script open the terminal in the directory where the script is saved and run the following command: **python3 tcamultiplex.py**

The result should look like on the following image:



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ python3 tcamultiplex.py  
TCA9548A Multiplexer + BME280 script  
Temperature: 25.45254148818078  
Pressure: 997.5966359913317  
Humidity: 32.69329759341413  
(25.45254148818078, 997.5966359913317, 32.69329759341413)  
Temperature: 25.454786663846697  
Pressure: 997.5984884448583  
Humidity: 32.69325349087218  
(25.454786663846697, 997.5984884448583, 32.69325349087218)  
Temperature: 25.451579270055298  
Pressure: 997.6001030627693  
Humidity: 32.699228693972124  
(25.451579270055298, 997.6001030627693, 32.699228693972124)  
^C  
Traceback (most recent call last):  
  File "tcamultiplex.py", line 61, in <module>  
    print(readdata())  
  File "tcamultiplex.py", line 42, in readdata  
    time.sleep(2)  
KeyboardInterrupt  
pi@raspberrypi:~ $
```

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



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 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>