Guide for TCA9548A I2C Multiplexer: ESP32, ESP8266, Arduino

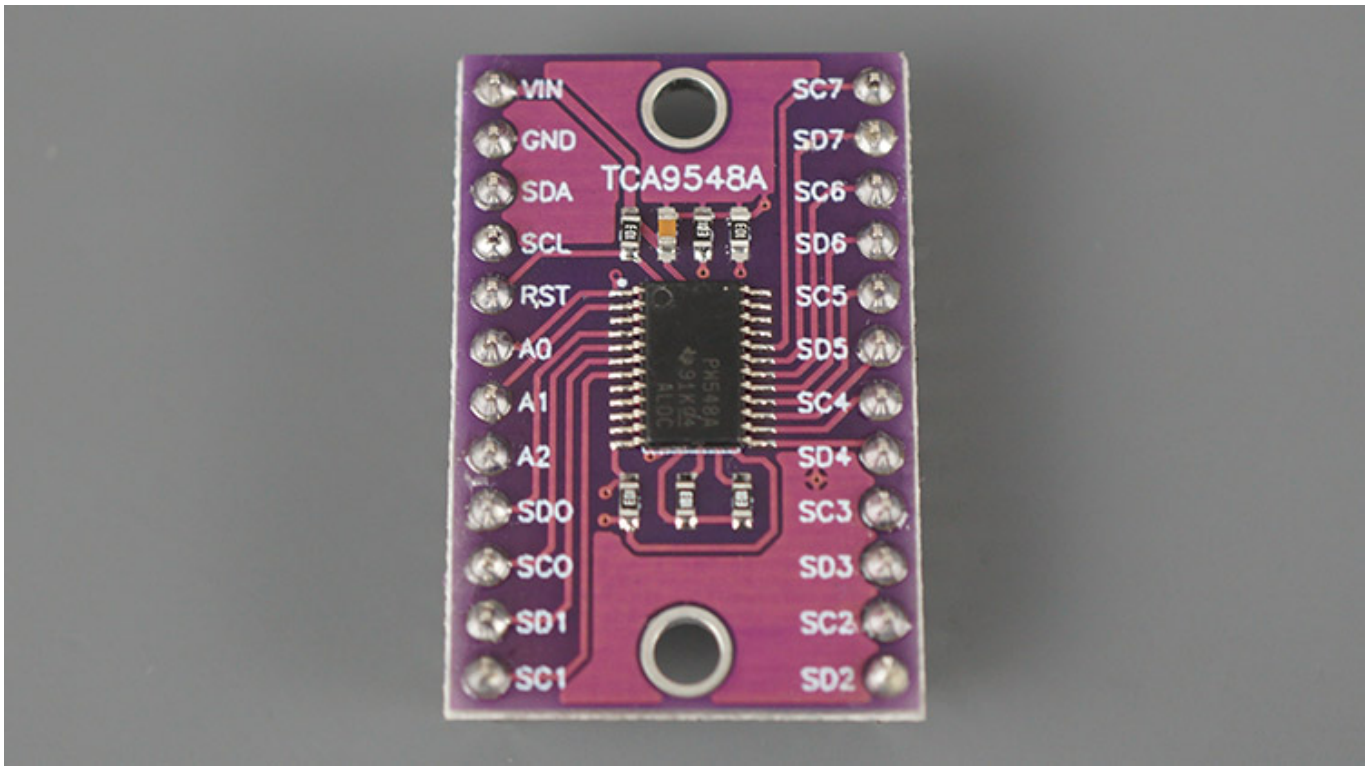
In this guide, you'll learn how to expand the I2C bus ports (ESP32, ESP8266, Arduino) using the TCA9458A 1-to-8 I2C Multiplexer. This piece of hardware is useful if you want to control multiple I2C devices with the same I2C address. For example, multiple OLED displays, or multiple sensors like the BME280.

This tutorial is compatible with ESP32, ESP8266, and Arduino boards. We'll program the boards using Arduino IDE.

Table of Contents

In this tutorial, we'll cover the following topics:

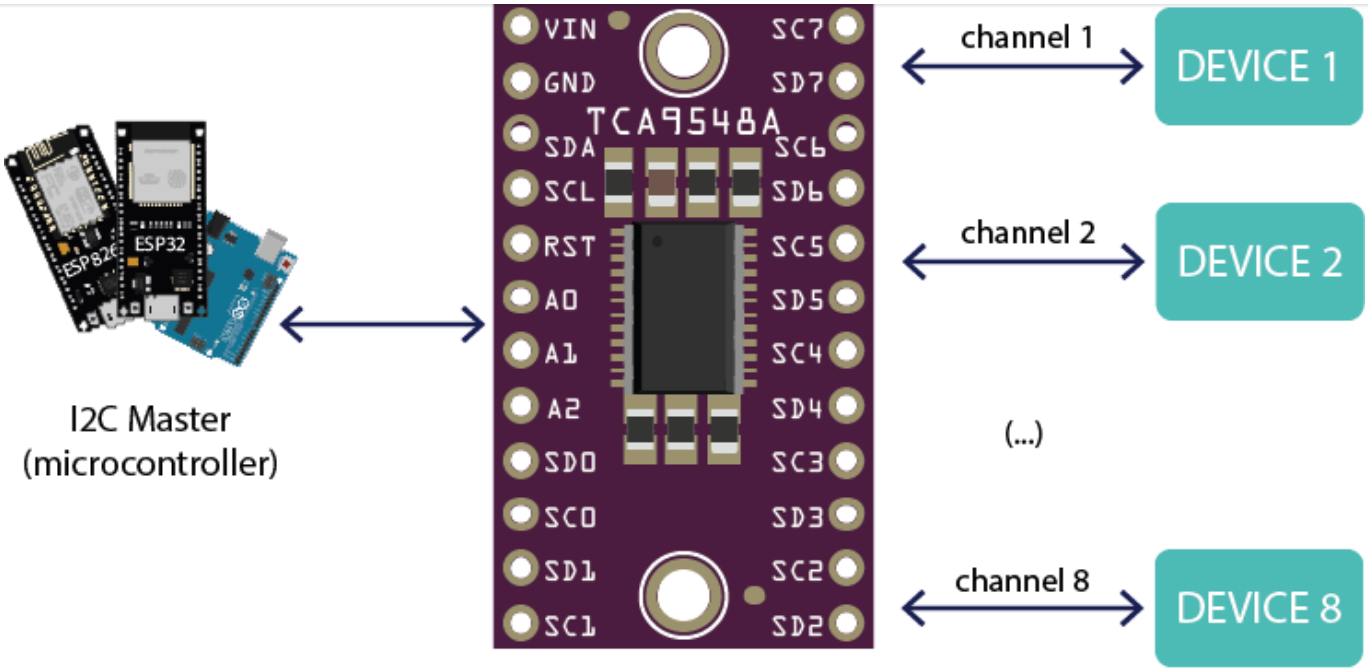
Introducing the TCA9548A 1-to-8 I2C Multiplexer



The I2C communication protocol allows you to communicate with multiple I2C devices on the same I2C bus as long as all devices have a unique I2C address. However, it will not work if you want to connect multiple I2C devices with the same address.

The TCA9548A I2C multiplexer allows you to communicate with up to 8 I2C devices with the same I2C bus. The multiplexer communicates with a microcontroller using the I2C communication protocol. Then, you can select which I2C bus on the multiplexer you want to address.

To address a specific port, you just need to send a single byte to the multiplexer with



TCA9548A Multiplexer Features

Here's a summary of its main features:

- 1 to 8 bidireccional translating switches
- Active-low reset input
- Three address pins—up to 8 TCA9548A devices on the same I2C bus
- Channel selection through an I2C bus
- Operating power supply voltage range: 1.65V to 5.5V
- 5V tolerant pins

For a more detailed description, consult the datasheet.

TCA9548A Multiplexer I2C Address

The TCA9548A Multiplexer communicates with a microcontroller using the I2C communication protocol. So, it needs an I2C address. The address of the multiplexer is configurable. You can select a value from 0x70 to 0x77 by adjusting the values of the A0, A1, and A2 pins, as shown in the table below.

A0	A1	A2	I2C Address

HIGH	LOW	LOW	0x71
LOW	HIGH	LOW	0x72
HIGH	HIGH	LOW	0x73
LOW	LOW	HIGH	0x74
HIGH	LOW	HIGH	0x75
LOW	HIGH	HIGH	0x76
HIGH	HIGH	HIGH	0x77

So, you can connect up to 8 TCA9548A multiplexers to the same I2C bus, which would allow you to connect 64 devices with the same address using only one I2C bus of the microcontroller.

For example, if you connect A0 , A1 , and A2 to GND, it sets address 0x70 for the multiplexer.

TCA9548A Pinout

The following table describes the TCA9584A Pinout.

Pin	Description
VIN	Powers the multiplexer
GND	Connect to GND
SDA	Connect to the master microcontroller SDA pin
SCL	Connect to the master microcontroller SCL pin
RST	Active low RST pin—can be used to reset the multiplexer

A1	Selects multiplexer I2C address—connect to GND or VCC
A2	Selects multiplexer I2C address—connect to GND or VCC
SD0	SDA for channel 0
SC0	SCL for channel 0
SD1	SDA for channel 1
SC1	SCL for channel 1
SD2	SDA for channel 2
SC2	SCL for channel 2
SD3	SDA for channel 3
SC3	SCL for channel 3
SD4	SDA for channel 4
SC4	SCL for channel 4
SD5	SDA for channel 5
SC5	SCL for channel 5
SD6	SDA for channel 6
SC6	SCL for channel 6
SD7	SDA for channel 7
SC7	SCL for channel 7

TCA9548A I2C Multiplexer Selecting an I2C Bus

As mentioned previously, to select a specific I2C bus to read/write data, you just need to send a single byte to the multiplexer with the desired output port number (0 to 7).

To do that, you can simply use the following user-defined function:

```
void TCA9548A(uint8_t bus){
  Wire.beginTransmission(0x70);  // TCA9548A address is 0x70
  Wire.write(1 << bus);          // send byte to select bus
  Wire.endTransmission();
  Serial.print(bus);
}
```

Then, you just need to call that function and pass as an argument the port bus number you want to control before sending the I2C commands. For example, to control the device connected to bus number 3, you would call the following line before calling other I2C commands (note that it starts at 0):

```
TCA9548A(2);
```

You'll see how this works with practical examples in the following sections.

Control Multiple OLED Displays—TCA9548A I2C Multiplexer

In this section, we'll show you how to control multiple OLED displays. As an example, we'll control four OLED displays, but you can connect up to 8 displays.

Parts Required

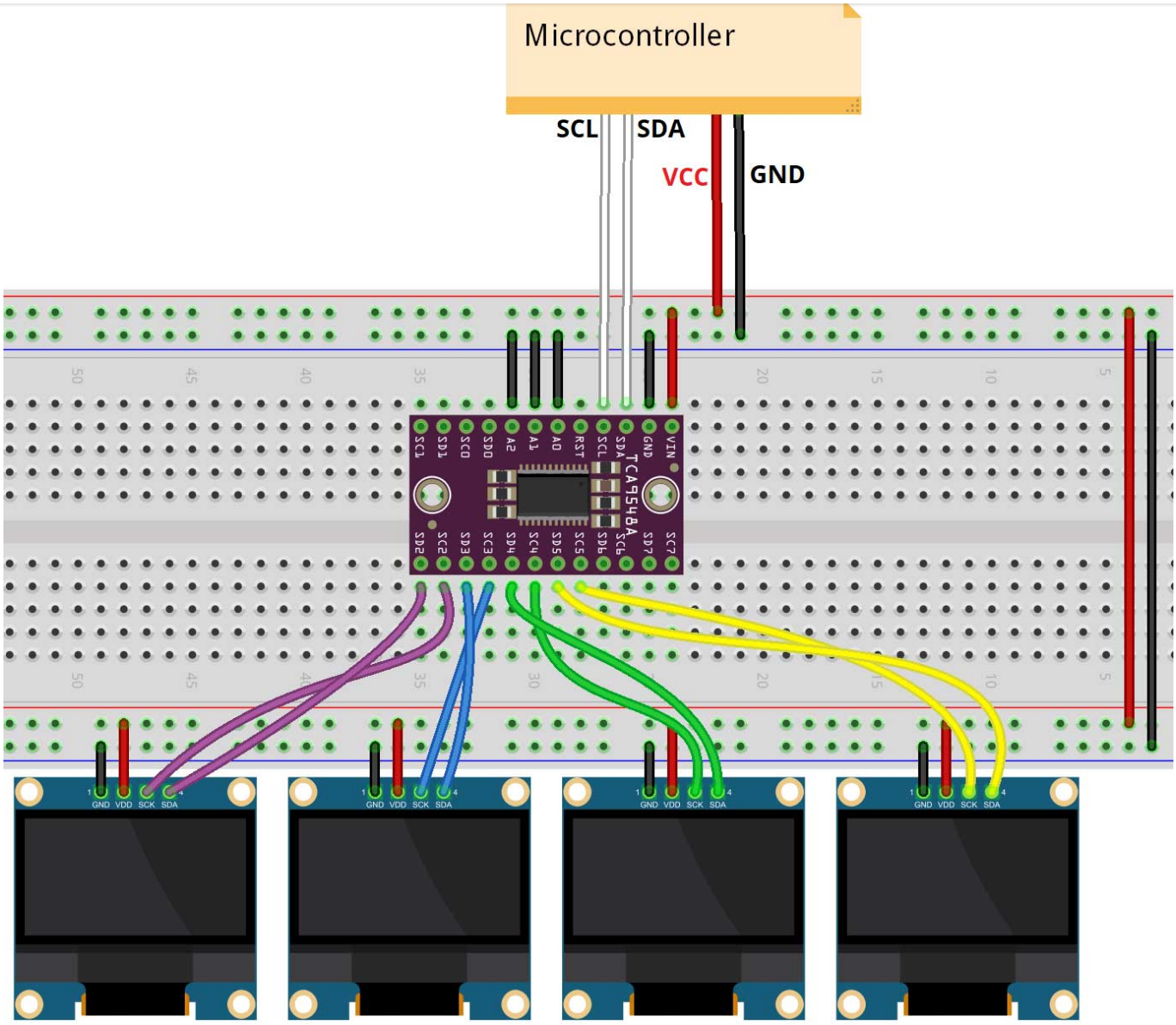
- Microcontroller (ESP32, ESP8266, Arduino, or other);
- TCA9548A I2C Multiplexer
- Multiple OLED Displays (up to 8)
- Breadboard
- Jumper wires

You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Multiple OLED Displays with I2C Multiplexer Circuit

Connect four OLED displays as shown in the following schematic diagram. We're using buses number 2, 3, 4 and 5. You can choose any other port number.

We're also connecting A0 , A1 , and A2 to GND. This selects address 0x70 for the multiplexer.



Here are the default I2C pins depending on the microcontroller you're using:

Microcontroller	I2C Pins
ESP32	GPIO 22 (SCL), GPIO 21 (SDA)
ESP8266	GPIO 5 (D1) (SCL), GPIO 4 (D2) (SDA)
Arduino Uno	A5 (SCL), A4 (SDA)

Multiple OLED Displays with I2C Multiplexer Code

Controlling the displays is as easy as controlling one display. You just need to consider selecting the right I2C bus before sending the commands to write to the display.

To learn how to control an I2C display, you can read the following articles:

- ESP32 OLED Display with Arduino IDE
- ESP8266 0.96 inch OLED Display with Arduino IDE
- Guide for I2C OLED Display with Arduino

Installing Libraries

We'll use the following libraries to control the OLED display. Make sure you have these libraries installed:

- Adafruit_SSD1306 library
- Adafruit_GFX library

You can install the libraries using the Arduino Library Manager. Go to **Sketch** > **Include Library** > **Manage Libraries** and search for the library name.

If you're using VS Code with the PlatformIO extension, copy the following to the `platformio.ini` file to include the libraries.

```
lib_deps = adafruit/Adafruit SSD1306@^2.4.6
    adafruit/Adafruit GFX Library@^1.10.10
```

After installing the libraries, you can proceed.

Copy the following code to your Arduino IDE and upload it to your board. It will work straight away.

```
/*********
```

```
  Complete project details at https://RandomNerdTutorials.com/tca

  Permission is hereby granted, free of charge, to any person obt
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be
  copies or substantial portions of the Software.
*********/

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// Select I2C BUS
void TCA9548A(uint8_t bus){
  Wire.beginTransmission(0x70);  // TCA9548A address
  Wire.write(1 << bus);          // send byte to select bus
  Wire.endTransmission();
```

View raw code

How the Code Works

Continue reading to learn how the code works or skip to the Demonstration section.

First, import the required libraries to control the OLED display: `Adafruit_GFX` and `Adafruit_SSD1306`. The `Wire` library is needed to use the I2C communication protocol.

```
#include <Adafruit_SSD1306.h>
```

Define the OLED width and height.

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
```

Create an `Adafruit_SSD1306` instance to communicate with the OLED display.

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

You can use the same instance to communicate with all displays. In that case, you need to clear the display buffer (`display.clearDisplay()`) before writing to another OLED.

Alternatively, you can create multiple `Adafruit_SSD1306` instances, one for each OLED. In that case, you don't need to clear the buffer. We'll show you an example with multiple instances at the end of this section.

Select the I2C Channel

The `TCA9548A()` function can be called to select the bus that you want to communicate with. It sends a byte to the multiplexer with the port number.

```
// Select I2C BUS
void TCA9548A(uint8_t bus){
  Wire.beginTransmission(0x70);  // TCA9548A address
  Wire.write(1 << bus);          // send byte to select bus
  Wire.endTransmission();
```

```
    Serial.print(bus);
  }
```

You must call this function whenever you want to select the I2C port.

setup()

In the `setup()`, initialize a serial communication for debugging purposes.

```
  Serial.begin(115200);
```

Start I2C communication on the default I2C pins with the I2C multiplexer.

```
  Wire.begin();
```

Then, initialize each display. The following lines show an example for the first OLED display (it is connected to bus number 2).

```
  //Init OLED display on bus number 2
  TCA9548A(2);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  // Clear the buffer
  display.clearDisplay();
```

Initializing the other displays is similar, but you need to call the `TCA9548A()` function with the corresponding I2C bus number.

Then, write something to the displays. Don't forget you need to call the `TCA9548A()` function every time you want to switch between OLEDs. You also need to clear the display buffer before writing anything to the OLED.

```
// Write to OLED on bus number 2
TCA9548A(2);
display.clearDisplay();
display.setTextSize(8);
display.setTextColor(WHITE);
display.setCursor(45, 10);
// Display static text
display.println("1");
display.display();
```

In this case, we're just printing a different number on each display. Here's an example for OLED number 4 (it is connected to bus number 5).

```
// Write to OLED on bus number 5
TCA9548A(5);
display.clearDisplay();
display.setTextSize(8);
display.setTextColor(WHITE);
display.setCursor(45, 10);
// Display static text
display.println("4");
display.display();
```

And that's pretty much how the code works.

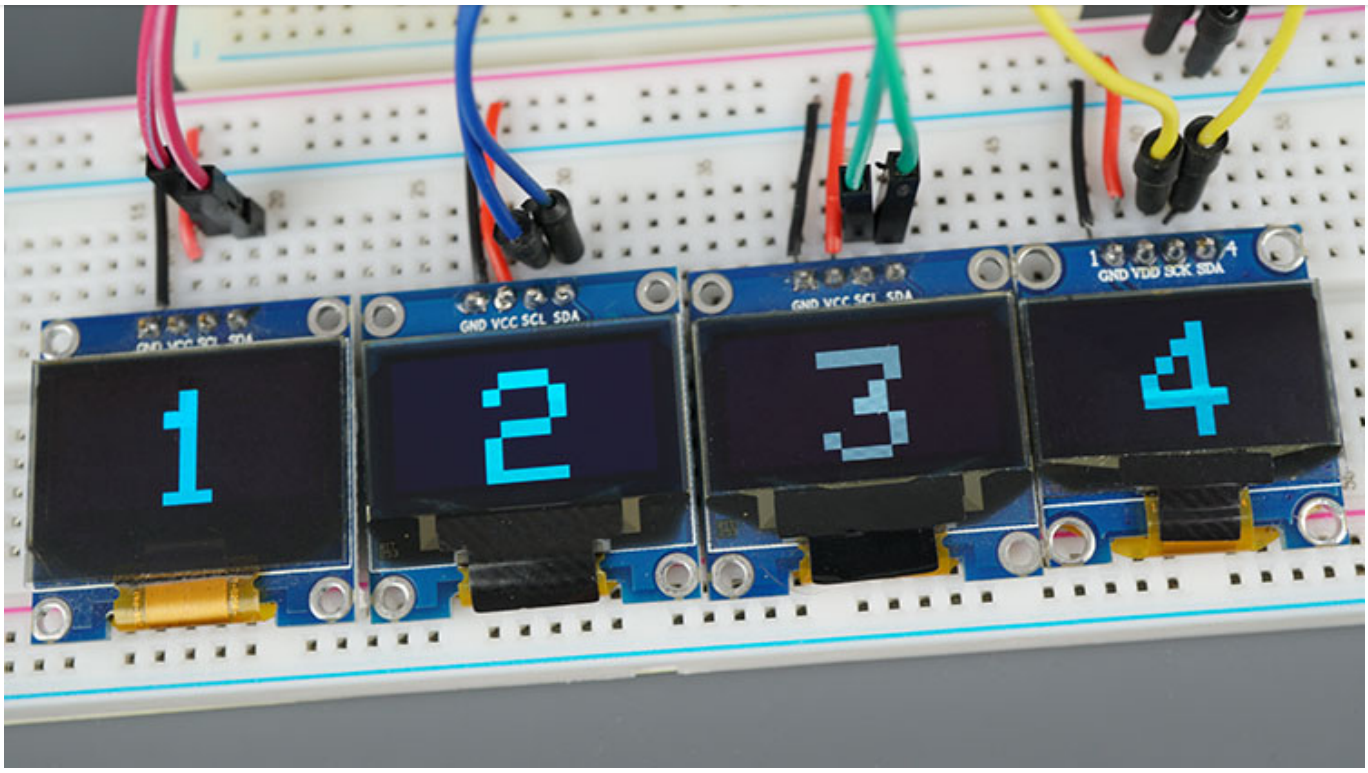
The following code shows a similar example but using multiple `Adafruit_SSD1306` instances. Notice that you don't need to clear the buffer before writing to the display.

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/tca

  Permission is hereby granted, free of charge, to any person obt
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be
  copies or substantial portions of the Software.
*********/

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display1(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1
Adafruit_SSD1306 display2(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1
Adafruit_SSD1306 display3(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1
Adafruit_SSD1306 display4(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1

// Select I2C BUS
void TCA9548A(uint8_t bus){
  Wire.beginTransmission(0x70);  // TCA9548A address
```

View raw code

Demonstration

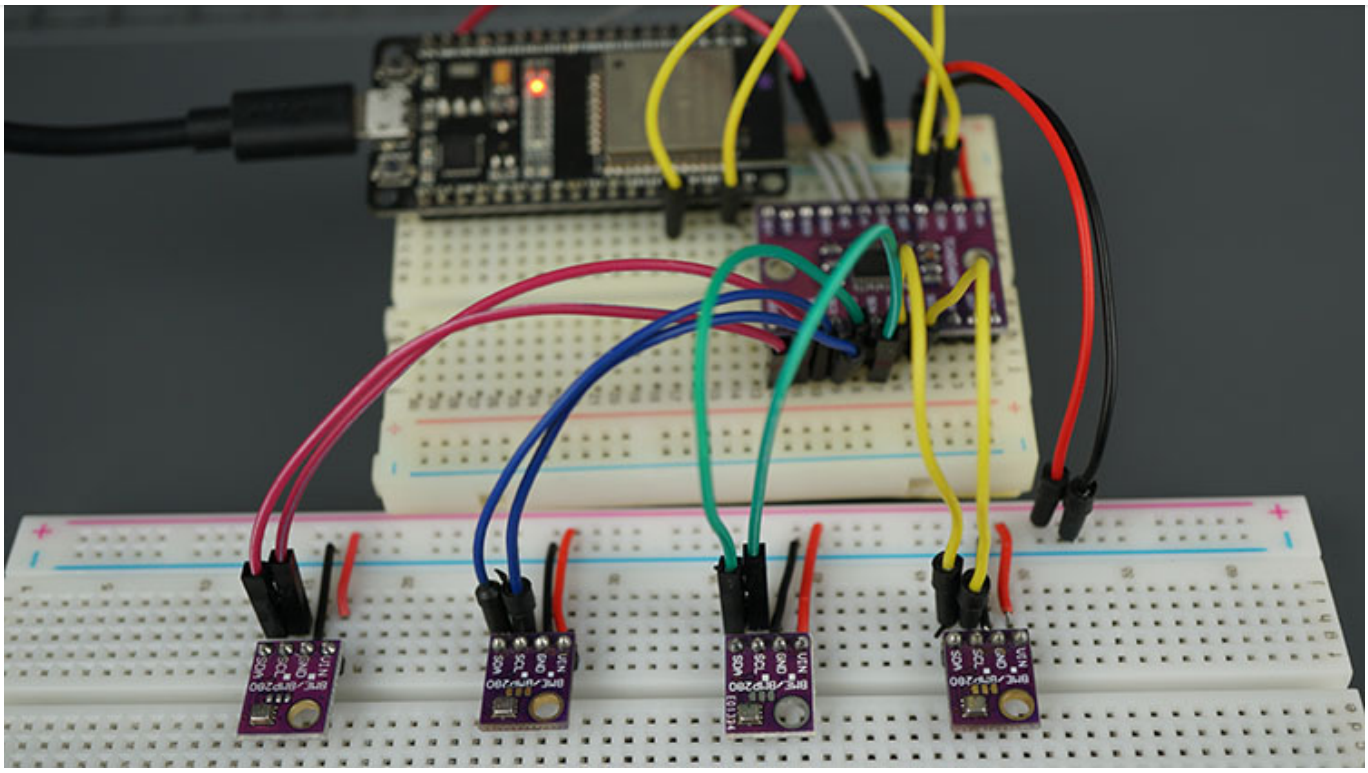
Upload the code to your board. Here's what you should get.



As you can see, it is pretty easy to control multiple OLED displays showing different graphics using an I2C multiplexer.

Read Multiple BME280 Sensors —TCA9548A I2C Multiplexer

In this section, you'll learn how to read data from multiple BME280 sensors using the TCA9548A I2C multiplexer. We'll read data from four sensors, but you can hook up to 8 sensors.



Parts Required

Here's a list of the parts required for this example:

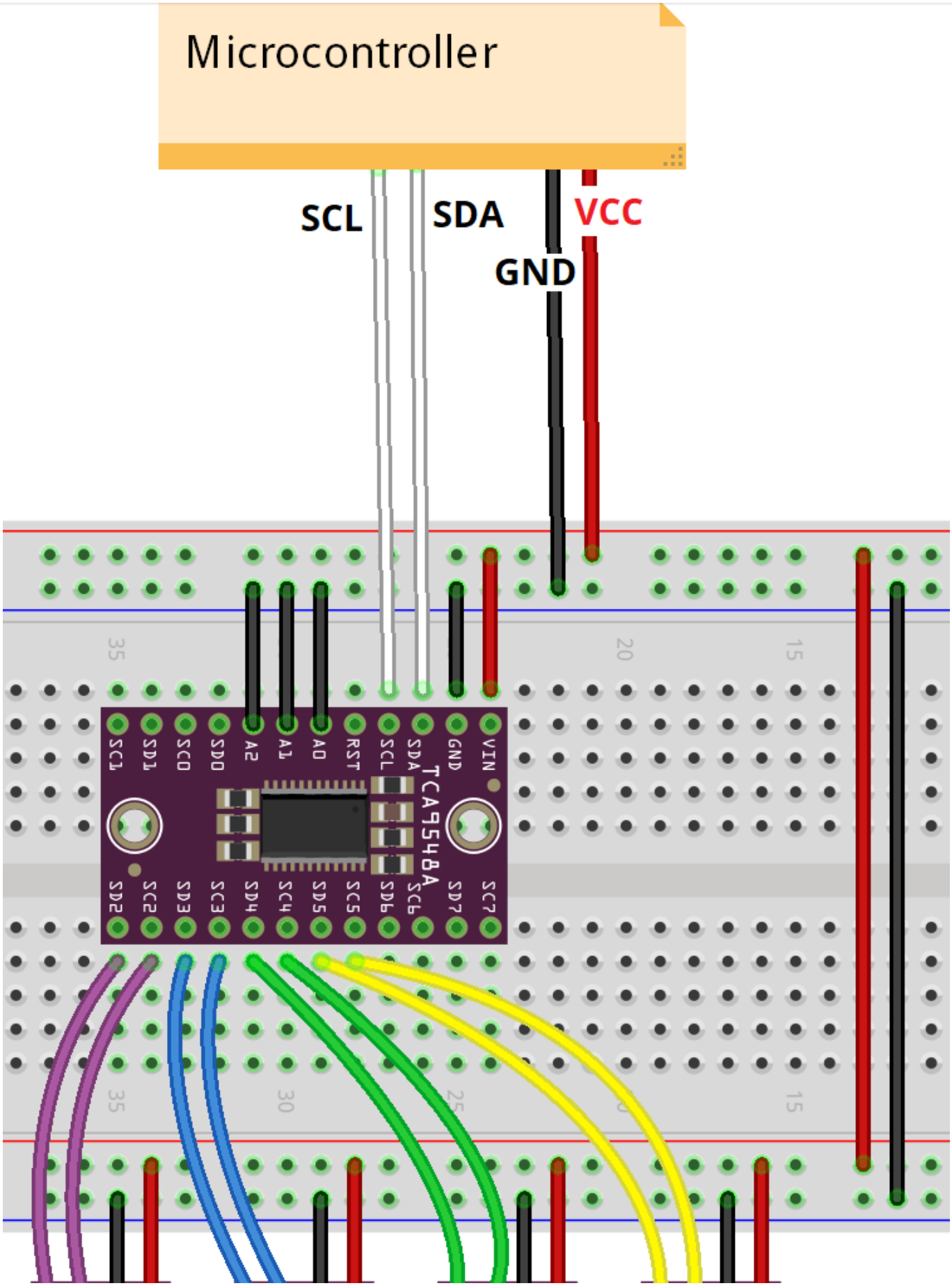
- Microcontroller (ESP32, ESP8266, Arduino, or other);
- TCA9548A I2C Multiplexer
- Multiple BME280 sensors (up to 8)
- Breadboard
- Jumper wires

You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Multiple BME280 Sensors with I2C Multiplexer Circuit

Connect four BME280 sensors as shown in the following schematic diagram. We're using buses number 2, 3, 4 and 5. You can choose any other port numbers

We're also connecting A0 , A1 , and A2 to GND. This selects address 0x70 for the multiplexer.


Microcontroller
SCL
SDA
VCC
GND
SC1
SD1
SC0
SD0
A2
A1
A0
RST
SCL
SDA
GND
VIN
TCA9548A
SD2
SC2
SD3
SC3
SD4
SC4
SD5
SC5
SD6
SC6
SD7
SC7

Here are the default I2C pins depending on the microcontroller you're using:

Microcontroller	I2C Pins
ESP32	GPIO 22 (SCL), GPIO 21 (SDA)
ESP8266	GPIO 5 (D1) (SCL), GPIO 4 (D2) (SDA)
Arduino Uno	A5 (SCL), A4 (SDA)

Multiple BME280 Sensors with I2C Multiplexer Code

Similar to the OLED display, reading data from multiple sensors is as easy as controlling one single sensor. You just need to take into account selecting the right I2C bus before communicating with the sensor.

To learn how to read data from a BME280 sensor:

- ESP32 with BME280 Sensor using Arduino IDE (Pressure, Temperature, Humidity)
- ESP8266 with BME280 using Arduino IDE (Pressure, Temperature, Humidity)
- Guide for BME280 Sensor with Arduino (Pressure, Temperature, Humidity)

Installing Libraries

We'll use the following libraries to read from the BME280 sensor. Make sure you have these libraries installed:

- Adafruit_BME280 library
- Adafruit_Sensor library

You can install the libraries using the Arduino Library Manager. Go to **Sketch** > **Include Library** > **Manage Libraries** and search for the library name.

If you're using VS Code with the PlatformIO extension, copy the following to the `platformio.ini` file to include the libraries.

```
lib_deps = adafruit/Adafruit Unified Sensor @ ^1.1.4
           adafruit/Adafruit BME280 Library @ ^2.1.2
```

After installing the libraries, you can proceed.

Copy the following code to your Arduino IDE and upload it to your board. It will work straight away.

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/tca

  Permission is hereby granted, free of charge, to any person obt
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be
  copies or substantial portions of the Software.
*********/

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

#define SEALEVELPRESSURE_HPA (1022)

Adafruit_BME280 bme1; // I2C
Adafruit_BME280 bme2; // I2C
```

```
Adafruit_BME280 bme4; // I2C

// Select I2C BUS
void TCA9548A(uint8_t bus){
  Wire.beginTransmission(0x70);  // TCA9548A address
```

View raw code

How the Code Works

Continue reading to learn how the code works or skip to the Demonstration section.

First, import the required libraries to control the BME280 display: `Adafruit_BME280` and `Adafruit_Sensor`. The `Wire` library is needed to use the I2C communication protocol.

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

Then, you need to create several instances of `Adafruit_BME280`, one for each sensor: `bme1`, `bme2`, `bme3`, and `bme4`.

```
Adafruit_BME280 bme1; // I2C
Adafruit_BME280 bme2; // I2C
Adafruit_BME280 bme3; // I2C
Adafruit_BME280 bme4; // I2C
```

Select the I2C Channel

The `TCA9548A()` function can be called to select the bus that you want to communicate with. It sends a byte to the multiplexer with the port number.

```
// Select I2C BUS
void TCA9548A(uint8_t bus){
  Wire.beginTransmission(0x70);  // TCA9548A address
  Wire.write(1 << bus);          // send byte to select bus
  Wire.endTransmission();
  Serial.print(bus);
}
```

You must call this function whenever you want to select the I2C port.

printValues() function

Then, we create a function `printValues()` that allows us to print in the Serial Monitor the values for each sensor. This function allows us to pass the `Adafruit_BME280` instance and its bus.

Inside the function, we select the I2C bus we want to talk to by calling the `TCA9548A()` function and passing the bus as an argument.

```
TCA9548A (bus);
```

Then, we use the usual functions to get readings from the sensor.

```
Serial.print("Sensor number on bus");
Serial.println(bus);
Serial.print("Temperature = ");
Serial.print(bme.readTemperature());
Serial.println(" *C");

Serial.print("Pressure = ");
Serial.print(bme.readPressure() / 100.0F);
```

```
  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");

  Serial.print("Humidity = ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");
```

setup()

In the `setup()`, initialize a serial communication for debugging purposes.

```
  Serial.begin(115200);
```

Start I2C communication on the default I2C pins with the I2C multiplexer.

```
  Wire.begin();
```

Then, initialize each sensor. The following lines show an example for the first BME280 sensor (it is connected to bus number 2, and it refers to the `bme1` instance).

```
  //Init sensor on bus number 2
  TCA9548A(2);
  if (!bme1.begin(0x76)) {
    Serial.println("Could not find a valid BME280 sensor on bus 2, ch
    while (1);
  }
  Serial.println();
```

Initializing the other sensors is similar, but you need to call the `TCA9548A()` function with the corresponding I2C bus number. Also, don't forget that each sensor has its own instance.

loop()

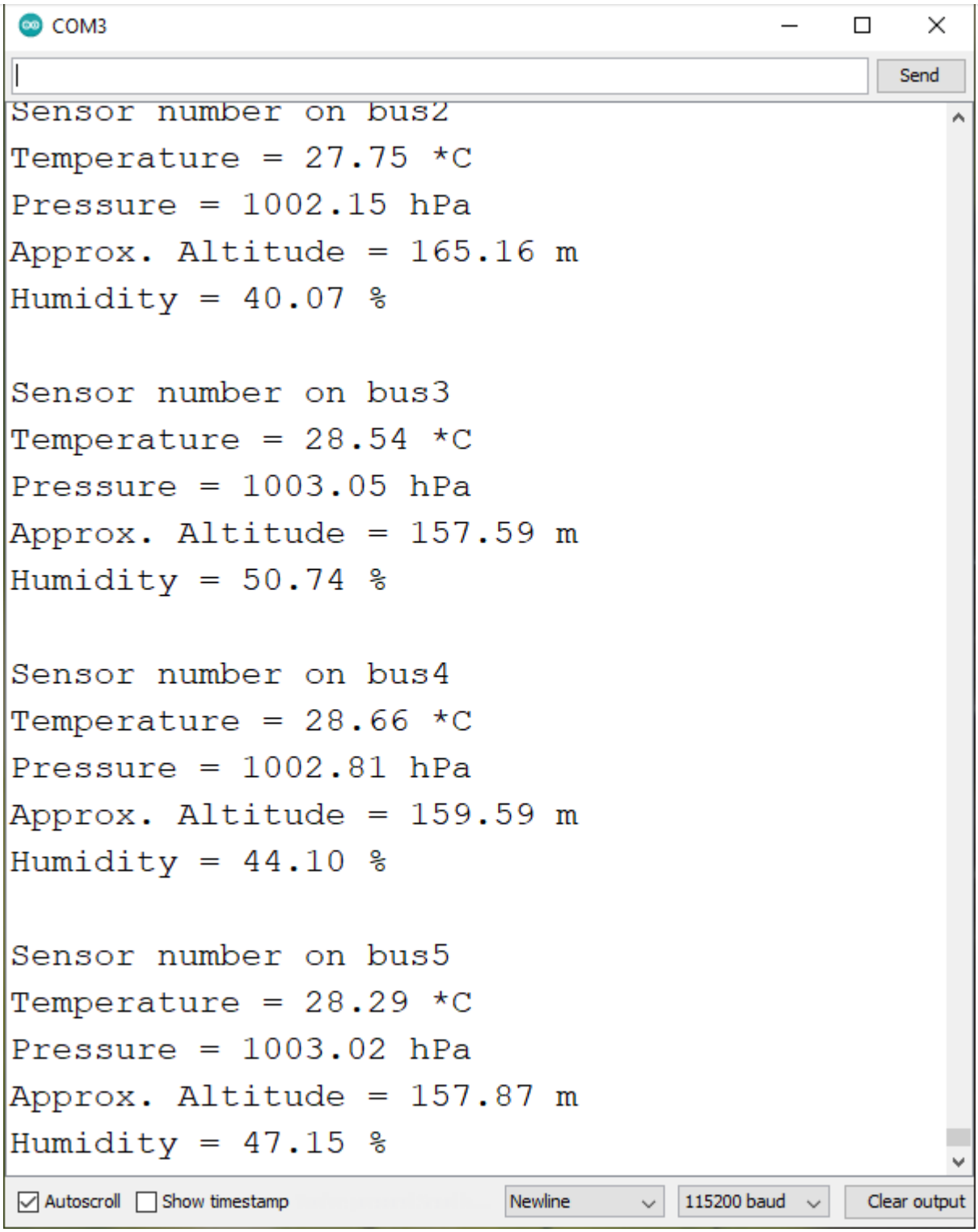
In the `loop()`, we call the `printValues()` function for each sensor.

```
printValues(bme1, 2);
printValues(bme2, 3);
printValues(bme3, 4);
printValues(bme4, 5);
```

And that's pretty much how the code works.

Demonstration

Upload the code to your board. Open the Serial Monitor at a baud rate of 115200. The readings for each sensor will be displayed on the Serial Monitor.



Wrapping Up

This tutorial taught you how to add more I2C ports to your microcontroller with the TCA9548A I2C multiplexer. This is especially useful if you want to connect multiple devices with the same I2C address. Furthermore, the I2C address of the multiplexer

The examples shown throughout this tutorial are compatible with the ESP32, ESP8266, and Arduino boards.

We have an extensive tutorial about I2C functions with the ESP32:

- ESP32 I2C Communication: Set Pins, Multiple Bus Interfaces and Peripherals (Arduino IDE)

Learn more about ESP32, ESP8266, and Arduino with our resources:

	Free Projects	Course/eBook
ESP32	ESP32 Tutorials	Learn ESP32 with Arduino IDE
ESP8266	ESP8266 Tutorials	Home Automation using ESP8266
Arduino	Arduino Tutorials	Arduino Step-by-step Projects

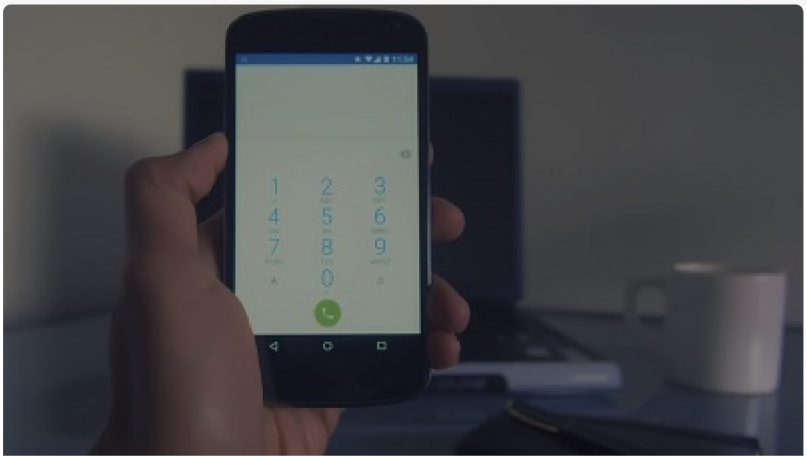
We hope you find this tutorial useful.

Thanks for reading.

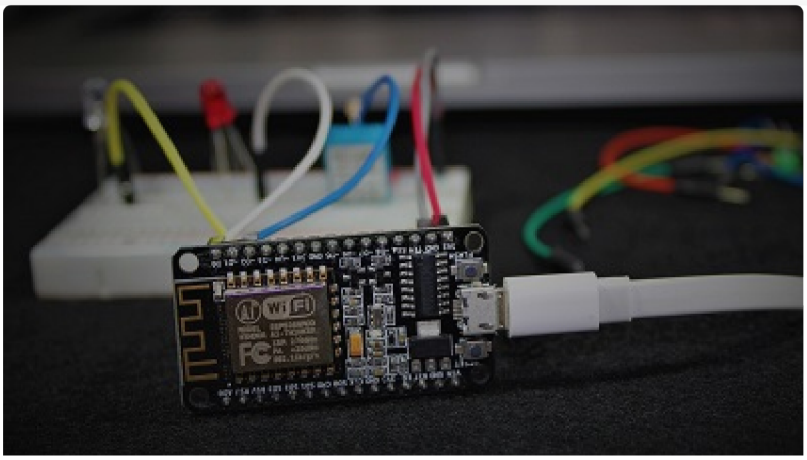
[eBook] Build Web Servers with ESP32 and ESP8266 (2nd Edition)

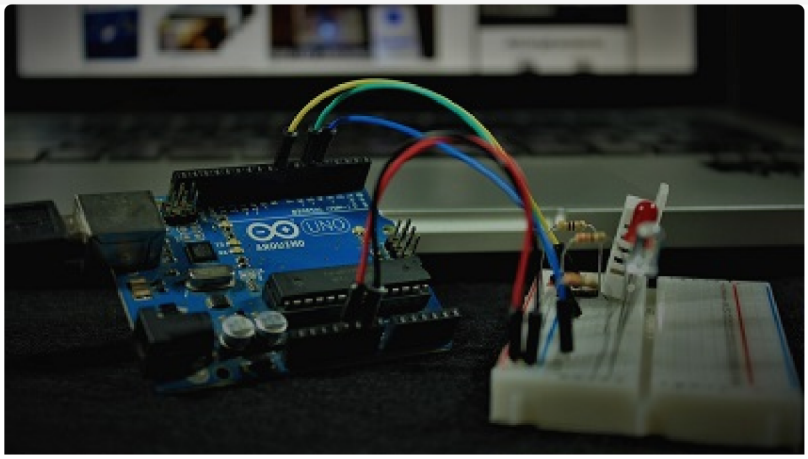
Build Web Server projects with the ESP32 and ESP8266 boards to control outputs and monitor sensors remotely. Learn HTML, CSS, JavaScript and client-server communication protocols **DOWNLOAD »**

Recommended Resources



Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED.





What to Read Next…

ESP32 LoRa Sensor Monitoring with Web Server (Long Range Communication)

How to Use I2C LCD with ESP32 on Arduino IDE (ESP8266 compatible)

24 thoughts on “Guide for TCA9548A I2C Multiplexer: ESP32, ESP8266, Arduino”

Carlos Bruni
September 9, 2021 at 1:37 pm

Excelente Tutorial! Como sempre, vocês trabalham com muita competência e dedicação.
Parabéns
Salvador Bahia Brasil

Reply

Sara Santos
September 9, 2021 at 9:51 pm

Obrigada 🙂

Reply

Mike. M.
September 9, 2021 at 6:14 pm

Don’t you have to load the TCA9548A lib?

Sara Santos

September 9, 2021 at 9:51 pm

Hi.
No, you just need the wire library that is include by default with the ESP32 add-on for Arduino IDE.
Regards,
Sara

Reply

Highway11

January 6, 2022 at 3:55 pm

I'm following this tutorial and I cannot get it to work. When I run an i2c scan I get no devices found. This is when scanning the traditional way as well as using a nested for loop to scan all addresses on the multiplexor.

In my online research people have recommended using pull up resistors. But I don't see any mention of that in this tutorial. When I connect the OLED directly to my esp32 the i2c scan sees it. Is the lack of pull up resistors the reason my setup doesn't work or do I just have a non-functional multiplexor?

Reply

Sara Santos

Hi.
These modules don’t need resistors because they are already built-in into the module.
If you were using the bare chip, you would need to use the resistors.

The multiplexer will have a different address depending on the values of A0, A1, and A2.
Make sure you’re connecting the multiplexer to the right I2C pins of your microcontroller.

Regards,
Sara

Reply

Highway11
January 6, 2022 at 6:31 pm

Thanks for the response.

I am able to see and use the OLED directly connected to the esp32 so I think the pins I’m using are correct. I get nothing when using the tca9548a though. Strange. maybe it is defective.

Reply

Highway11
January 12, 2022 at 1:21 am

It turns out my multiplexor was defective. Ordered a 2nd one and it works fine. No pull-up resistors needed . Thanks!

Reply

Sara Santos

January 12, 2022 at 4:33 pm

Great!

Jesus Cuesta

March 23, 2022 at 12:38 pm

Good afternoon
Excellent contribution, which has helped me to practice with the multiplexer and at the same time change OLED with the SH1106 driver, with excellent results.
Thanks

Reply

Sara Santos

March 25, 2022 at 6:21 pm

Reply

JORDI CATALA

March 28, 2022 at 6:13 pm

Thank you for your excellent tutorials.
I succeeded with BME280 and SSD1306 using the same I2C. Next step….
I made a mix of the two parts of this tutorial using TCA9548A, two BME280 and two SSD1306.
I defined bme1 and bme2 as sensors and display1 and display2 for screens 1 and 2
Once I got the “fusion” of the codes I compiled it and it was ok.
No problem connecting the devices according to your diagrams.
I tried the code displaying: display1(“1”); and display2(“2”) it was ok. That means the circuit and the code for printing at least was correct.
The problem comes when displaying the variables:
display1.print(bme1,readTemperature());
……..
display2.print(bme2,readTemperature());
it shwos:
Screen1 181,18 Screen2 180,25
This values don’t change even heating the sensors with my fingers
I changed the positions of BME1 and BME2 , …..the same result.
Changing position of SSD1 and SSD2 …. Screen2 180.25 and Screen 1 181,18

I included in the same code a part that prints the BME values on the monitor and they are correct.
I tried to define a STRING ON EACH VARIABLE BUT ALWAYS THE SAME……..
PLEASE, CAN YOU HELP ME? THANK YOU

Reply

Sara Santos

March 28, 2022 at 10:30 pm

Hi.
You're probably not updating your OLED displays properly.
Make sure you call: display.clearDisplay() before writing something to the display.
AND, don't forget to call display.display() after any display.print(…).
I hope this helps.
Regards,
Sara

Reply

JORDI CATALA

March 30, 2022 at 9:49 am

Thank you Sara for your answer.
In my code after "writing" I call to "display.display()" but before writing I did not call to
"display.clearDisplay()" once corrected it, when compiling, at code point:
— display2.display(bme1.readTemperature());
I get error:
" no matching function for call to 'Adafruit_SSD1306::display(float)'"
It seems a problem of writing a float variable. This is why I tried to define a Srting without any result.
The same order

TCA9548A
I'm thinking the possibility of a bad pin welding in TCA9548A. I'll try with a new one.
I'm very confused.

Reply

JORDI CATALA
April 8, 2022 at 5:53 pm

Solved my problem. In case anybody interested i explain how I solved it.
The problem :
point of code:
— display2.print(bme1.readTemperature());
when compiling I got:
" no matching function for call to 'Adafruit_SSD1306::display(float)'"

Then I defined:
float temp,….,…;
……..
temp = bme.readTemperature();
……..
display.print(temp);

it works wery well.
Thank you for yout attention.

Reply

Jesús Cuesta
April 9, 2022 at 3:00 pm

Bona tarda Jordi, series tan amable de compartir el teu codi. Et deixo el meu email iotcascosta@gmail.com Gràcies. Salutacions

Good afternoon Jordi, I would appreciate it if you could share your code. I leave you my email iotcascosta@gmail.com Thank you, a cordial greeting.

Jaym

June 29, 2022 at 3:33 am

thanks, excelet tutorial,
i need conect a nodemcu v3 to multiplex, i like use i2c keypad, i2c ole and i2c lcd

i not understand, how to connect pin A0,A1,A2, ti pin Nodemcu, the tutorial work only display ole,

not work example
TCA9548A(0); TCA9548A(1); and use others port never
work not problem
TCA9548A(2);

i change multiplex for other, never working, please help me ..

sc0/sd0 = keypad_i2c
sc1/sd1= Lcd 16×2 use i2c
sc2/sd2 = display ole
SC3/sD3= MFR522 I2C.

the utils i2scanner, found only display ole

Reply

wafa abid

September 2, 2022 at 10:42 am

Hello,

Thank you for the informative tutorial, Can you please tell me the software with which you showed us the wiring? I tried circuit but it does not have the multiplexer in it. ‘

Thank you in advance

Reply

Sara Santos

September 4, 2022 at 11:14 am

Hi.
The software is Fritzing.
Regards,
Sara

Reply

Dom

Hello,
and thanks for another great tutorial 🙂

I just want to mention that the program won´t show anything if you don´t connect all I2C devices.

This is due to the “while (1)” constrain during initialization.

So if you only have two sensors at hand comment the other initializing parts out.

Maybe “while” should not be used?

Greetings Dom

Reply

Dom
October 19, 2022 at 4:48 pm

Hi again,

one question though:

If you combine both projects (OLED and BME) you can display a sensor reading on the OLED.

BUT I was able to do so only if I hoocked up one OLED to the TCA9548A() as another device.

So is it working this way at all? Or does all I2C devices run trough the TCA9548A()?

Thanks Dom

Reply

Maria Serna

November 3, 2022 at 8:17 pm

Hola, buenas tardes Sara Santos. Agradecerte por compartir la información. Ayuda por favor, ¿Cómo implementaría el código en caso de usar 3 sensores MPU6050?

Reply

Manny

January 31, 2023 at 8:59 am

hi there, i am trying to connect 5 MPU6050 sensors using ESP32 and MULTIPLEXER. can you help with the coding in arduino ide. if possible can we connect to discuss. thanks.

Reply

Reiner

February 11, 2023 at 9:54 pm

Kann der Plexer auch mit ESPeasy verwendet werden?

Reply

Leave a Comment

Name *

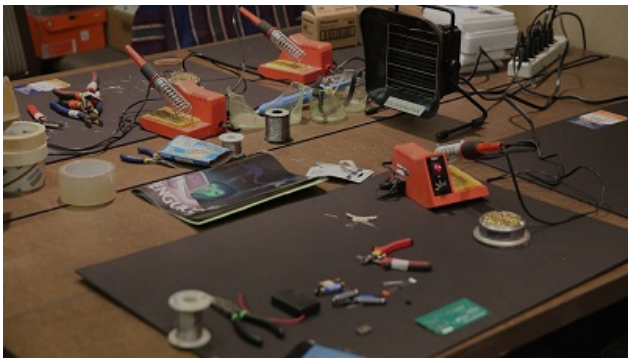
Email *

Website

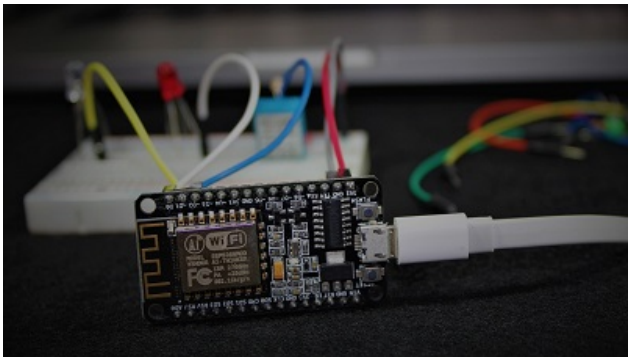
☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

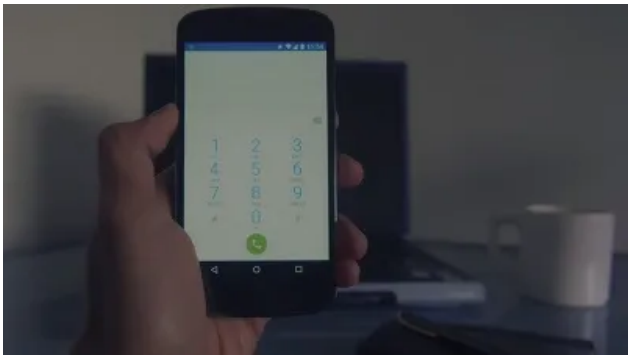
Post Comment



Visit Maker Advisor – Tools and Gear for makers, hobbyists and DIYers »



Home Automation using ESP8266 eBook and video course » Build IoT and home automation projects.



Build Web Servers with ESP32 and ESP8266 » boards to control outputs and monitor sensors remotely.