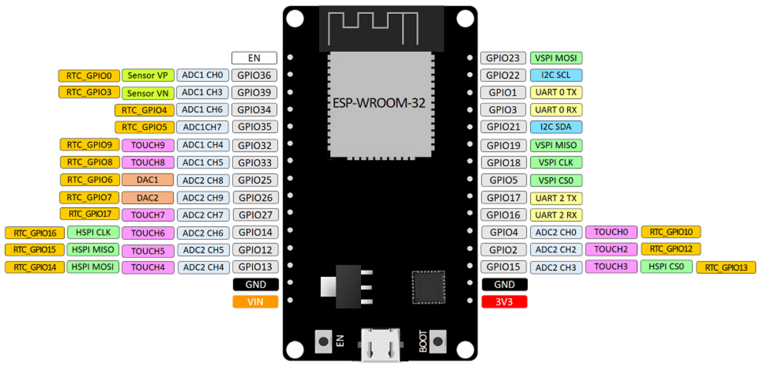
Module 1: IoT & Embedding Computing.

Hands on session 5: Weather Station

Session Introduction: In this session introduces you to I2C communication between an OLED display panel and BME Sensor, using these two we will make a weather station displaying sensory data form BME sensor onto OLED.



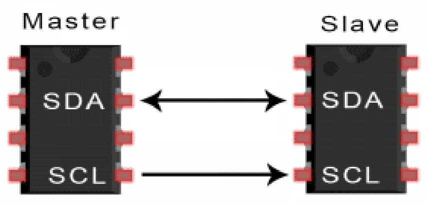
ESP-WROOM-32
EN
RTC_GPIO0 Sensor VP ADC1 CH0 GPIO36
RTC_GPIO3 Sensor VN ADC1 CH3 GPIO39
RTC_GPIO4 ADC1 CH6 GPIO34
RTC_GPIO5 ADC1CH7 GPIO35
RTC_GPIO9 TOUCH9 ADC1 CH4 GPIO32
RTC_GPIO8 TOUCH8 ADC1 CH5 GPIO33
RTC_GPIO6 DAC1 ADC2 CH8 GPIO25
RTC_GPIO7 DAC2 ADC2 CH9 GPIO26
RTC_GPIO17 TOUCH7 ADC2 CH7 GPIO27
RTC_GPIO16 HSPI CLK TOUCH6 ADC2 CH6 GPIO14
RTC_GPIO15 HSPI MISO TOUCH5 ADC2 CH5 GPIO12
RTC_GPIO14 HSPI MOSI TOUCH4 ADC2 CH4 GPIO13
GND
VIN
GPIO23 VSPI MOSI
GPIO22 I2C SCL
GPIO1 UART 0 TX
GPIO3 UART 0 RX
GPIO21 I2C SDA
GPIO19 VSPI MISO
GPIO18 VSPI CLK
GPIO5 VSPI CS0
GPIO17 UART 2 TX
GPIO16 UART 2 RX
GPIO4 ADC2 CH0 TOUCH0 RTC_GPIO10
GPIO2 ADC2 CH2 TOUCH2 RTC_GPIO12
GPIO15 ADC2 CH3 TOUCH3 HSPI CS0 RTC_GPIO13
GND
3V3
EN
BOOT

Session Objectives:

1. To understand I2C communication.
2. To understand working of BME Sensor.
3. To understand working of OLED.
4. Constructing a weather Station.

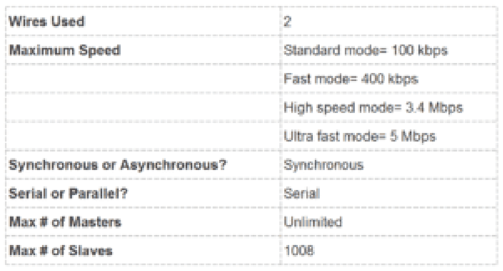
I2C Communication

With I2C, you can connect multiple slaves to a single master and you can have multiple masters controlling single, or multiple slaves. This is really useful when you want to have more than one microcontroller logging data to a single memory card or displaying text to a single LCD.
Like UART communication, I2C only uses two wires to transmit data between devices:



SDA (Serial Data) – The line for the master and slave to send and receive data.
SCL (Serial Clock) – The line that carries the clock signal.

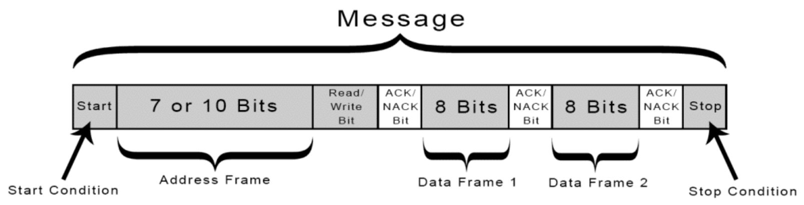
2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line).
 I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.



Wires Used	2
Maximum Speed	Standard mode= 100 kbps
	Fast mode= 400 kbps
	High speed mode= 3.4 Mbps
	Ultra fast mode= 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	Unlimited
Max # of Slaves	1008

HOW I2C WORKS

With I2C, data is transferred in *messages.* Messages are broken up into *frames* of data.
Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame:



Start Condition: The SDA line switches from a high voltage level to a low voltage level *before* the SCL line switches from high to low.
Stop Condition: The SDA line switches from a low voltage level to a high voltage level *after* the SCL line switches from low to high.
Address Frame: A 7 or 10 bit sequence unique to each slave that identifies the slave when the master wants to talk to it.
Read/Write Bit: A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).
ACK/NACK Bit: Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

ADDRESSING

I2C doesn't have slave select lines like SPI, so it needs another way to let the slave know that data is being sent to it, and not another slave. It does this by *addressing*. The address frame is always the first frame after the start bit in a new message.
The master sends the address of the slave it wants to communicate with to every slave connected to it. Each slave then compares the address sent from the master to its own address. If the address matches, it sends a low voltage ACK bit back to the master. If the address doesn't match, the slave does nothing and the SDA line remains high.

READ/WRITE BIT

The address frame includes a single bit at the end that informs the slave whether the master wants to write data to it or receive data from it. If the master wants to send data to the slave, the read/write bit is a low voltage level. If the master is requesting data from the slave, the bit is a high voltage level.

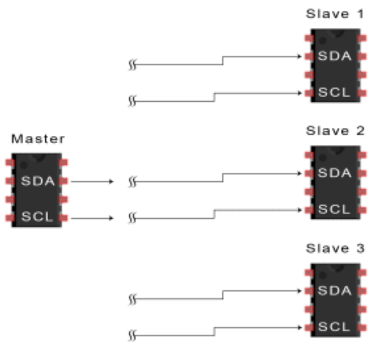
THE DATA FRAME

After the master detects the ACK bit from the slave, the first data frame is ready to be sent.
The data frame is always 8 bits long, and sent with the most significant bit first. Each data frame is immediately followed by an ACK/NACK bit to verify that the frame has been received successfully. The ACK bit must be received by either the master or the slave (depending on who is sending the data) before the next data frame can be sent.

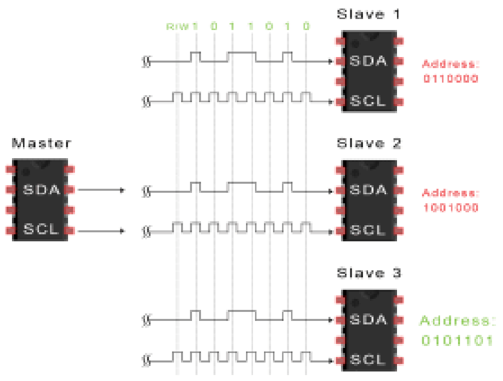
After all of the data frames have been sent, the master can send a stop condition to the slave to halt the transmission. The stop condition is a voltage transition from low to high on the SDA line after a low to high transition on the SCL line, with the SCL line remaining high.

STEPS OF I2C DATA TRANSMISSION

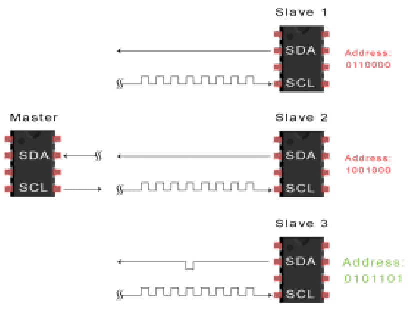
1. The master sends the start condition to every connected slave by switching the SDA line from a high voltage level to a low voltage level *before* switching the SCL line from high to low:



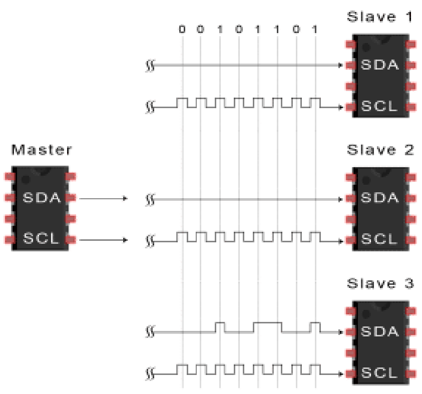
2. The master sends each slave the 7- or 10-bit address of the slave it wants to communicate with, along with the read/write bit:



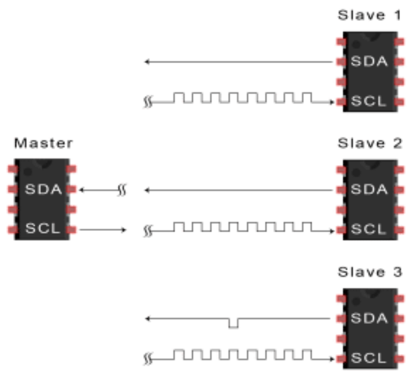
3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the SDA line low for one bit. If the address from the master does not match the slave’s own address, the slave leaves the SDA line high.



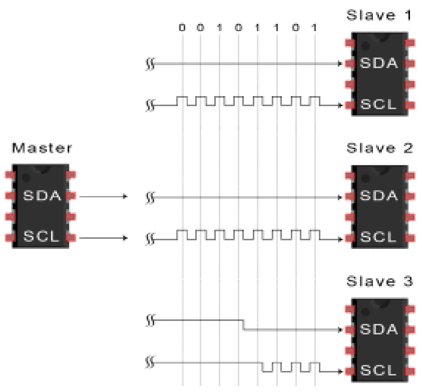
4. The master sends or receives the data frame:



5. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame:

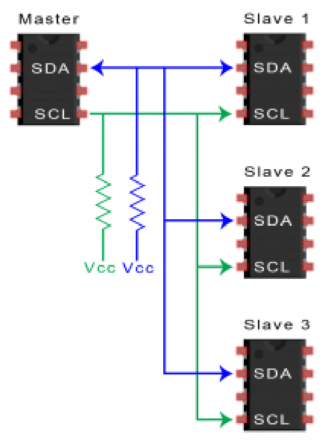


6. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high:



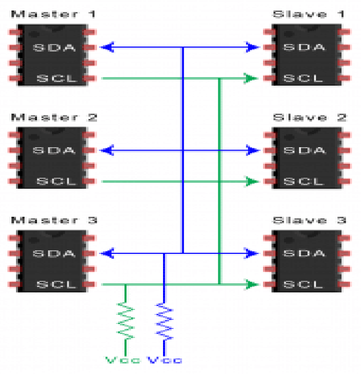
SINGLE MASTER WITH MULTIPLE SLAVES

Because I2C uses addressing, multiple slaves can be controlled from a single master. With a 7-bit address, 128 (2^7) unique address are available. Using 10-bit addresses is uncommon, but provides 1,024 (2^{10}) unique addresses. To connect multiple slaves to a single master, wire them like this, with 4.7K Ohm pull-up resistors connecting the SDA and SCL lines to Vcc:



MULTIPLE MASTERS WITH MULTIPLE SLAVES

Multiple masters can be connected to a single slave or multiple slaves. The problem with multiple masters in the same system comes when two masters try to send or receive data at the same time over the SDA line. To solve this problem, each master needs to detect if the SDA line is low or high before transmitting a message. If the SDA line is low, this means that another master has control of the bus, and the master should wait to send the message. If the SDA line is high, then it's safe to transmit the message. To connect multiple masters to multiple slaves, use the following diagram, with 4.7K Ohm pull-up resistors connecting the SDA and SCL lines to Vcc:



Screens

Types

- **Character based**
- **Graphical**

Character-based describes programs capable of displaying only ASCII (and extended ASCII) characters. Character-based programs treat a display screen as an array of boxes, each of which can hold one character. When in text mode, for example, PC screens are typically divided into 25 rows and 80 columns. In contrast, *graphics-based* programs treat the display screen as an array of millions of pixels. Characters and other objects are formed by illuminating patterns of pixels.

Character based displays:

Graphical

OLEDs

I2C 128X64 OLED SCREEN

The *organic light-emitting diode* (OLED) display that we'll use in this tutorial is the SSD1306
The OLED display doesn't require backlight, which results in a very nice contrast in dark environments. Additionally, its pixels consume energy only when they are on, so the OLED display consumes less power when compared with other displays.
The model we're using here has only four pins and communicates with the Arduino using I2C communication protocol. There are models that come with an extra RESET pin. There are also other OLED displays that communicate using SPI communication

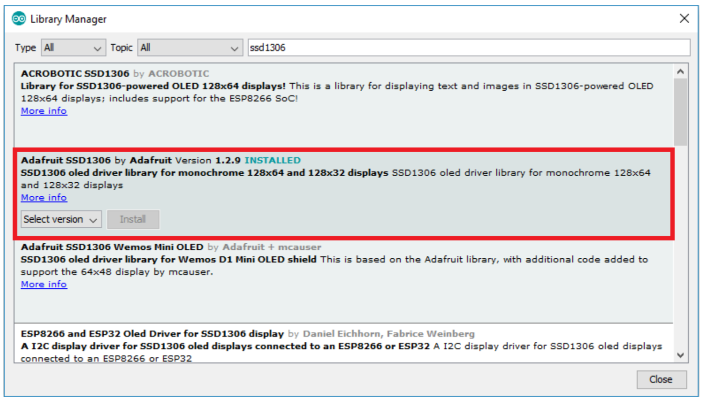
OLED Connections:

- **VCC -> +3.3V**
- **GND -> ground**
- **SCL -> GPIO 22**
- **SDA -> GPIO 21**

To work with OLEDs, we need to install primarily three libraries

- Wire.h
- Adafruit_GFX.h
- Adafruit_SSD1306.h

To control the OLED display you need the adafruit_SSD1306.h and the adafruit_GFX.h libraries. Follow the next instructions to install those libraries.
1. Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.
2. Type "**SSD1306**" in the search box and install the SSD1306 library from Adafruit.


Library Manager

Type: All Topic: All ssd1306

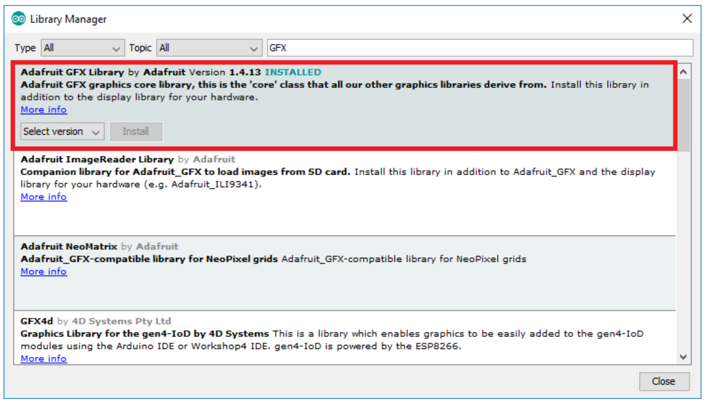
ACROBOTIC SSD1306 by ACROBOTIC
Library for SSD1306-powered OLED 128x64 displays! This is a library for displaying text and images in SSD1306-powered OLED 128x64 displays; includes support for the ESP8266 SoC!
More info

Adafruit SSD1306 by **Adafruit** Version **1.2.9** INSTALLED
SSD1306 oled driver library for monochrome 128x64 and 128x32 displays SSD1306 oled driver library for monochrome 128x64 and 128x32 displays
More info

Adafruit SSD1306 Wemos Mini OLED by Adafruit + mcauser
SSD1306 oled driver library for Wemos D1 Mini OLED shield This is based on the Adafruit library, with additional code added to support the 64x48 display by mcauser.
More info

ESP8266 and ESP32 Oled Driver for SSD1306 display by Daniel Eichhorn, Fabrice Weinberg
A I2C display driver for SSD1306 oled displays connected to an ESP8266 or ESP32 A I2C display driver for SSD1306 oled displays connected to an ESP8266 or ESP32

3. After installing the SSD1306 library from Adafruit, type "**GFX**" in the search box and install the library.


Library Manager

Type: All Topic: All GFX

Adafruit GFX Library by **Adafruit** Version **1.4.13** INSTALLED
Adafruit GFX graphics core library, this is the 'core' class that all our other graphics libraries derive from. Install this library in addition to the display library for your hardware.
More info

Adafruit ImageReader Library by Adafruit
Companion library for Adafruit_GFX to load images from SD card. Install this library in addition to Adafruit_GFX and the display library for your hardware (e.g. Adafruit_ILI9341).
More info

Adafruit NeoMatrix by Adafruit
Adafruit_GFX-compatible library for NeoPixel grids Adafruit_GFX-compatible library for NeoPixel grids
More info

GFX4d by 4D Systems Pty Ltd
Graphics Library for the gen4-IoD by 4D Systems This is a library which enables graphics to be easily added to the gen4-IoD modules using the Arduino IDE or Workshop4 IDE. gen4-IoD is powered by the ESP8266.
More info

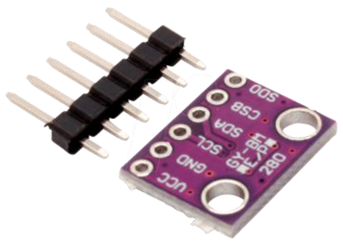
4. After installing the libraries, restart your Arduino IDE.

Some functions that will help you handle the OLED display library to write text or draw simple graphics.

- **display.clearDisplay() – all pixels are off**
- **display.drawPixel(x,y, color) – plot a pixel in the x,y coordinates**
- **display.setTextSize(n) – set the font size, supports sizes from 1 to 8**
- **display.setCursor(x,y) – set the coordinates to start writing text**
- **display.print("message") – print the characters at location x,y**
- **display.display() – call this method for the changes to make effect**

BME280

The BME280 sensor module reads barometric pressure, temperature, and humidity. Because pressure changes with altitude, you can also estimate altitude. There are several versions of this sensor module. The BME280 sensor uses I2C or SPI communication protocol to exchange data with a microcontroller. BME280 sensor used in this kit works on I2C communication.



BME280

This precision sensor can measure relative humidity from 0 to 100% with ±3% accuracy, barometric pressure from 300Pa to 1100 hPa with ±1 hPa absolute accuracy, and temperature from -40°C to 85°C with ±1.0°C accuracy.
The pressure measurements are so precise (low altitude noise of 0.25m), you can even use it as an altimeter with ±1 meter accuracy.

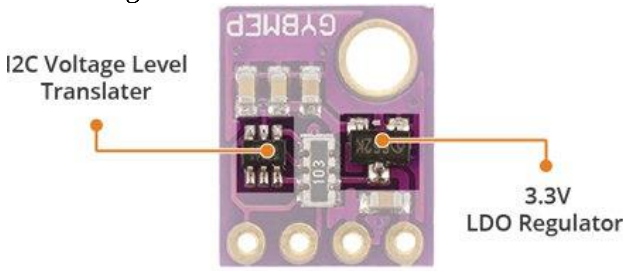
Temperature: -40°C to 85°C (±1.0°C accuracy)

Humidity: 0 to 100% RH (±3% accuracy)

Pressure: 300hPa to 1100hPa (±1hPa accuraccy)

Altitude: 0 to 30,000ft (±1 meter accuracy)

Power Requirement
The module comes with an on-board LM6206 3.3V regulator and I2C Voltage Level Translator, so you can use it with a 3.3V or 5V logic microcontroller like Arduino without worry.



The BME280 consumes less than 1mA during measurements and only 5µA during idle. This low power consumption allows the implementation in battery driven devices such as handsets, GPS modules or watches.

I2C Interface
The module features a simple two-wire I2C interface which can be easily interfaced with any microcontroller of your choice.
The default I2C address of the BME280 module is **0x76**$_{HEX}$ and can be changed to **0x77**$_{HEX}$ easily with the solder jumper besides chip.

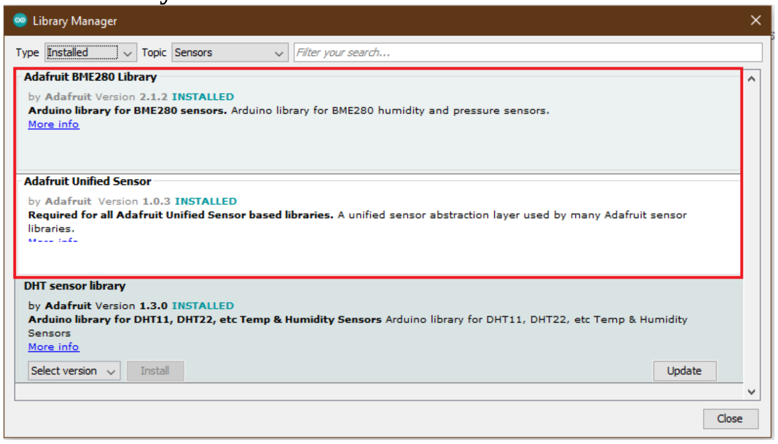
BME280 Connections:

- VCC -> +3.3V
- GND -> ground
- SCL -> GPIO 22
- SDA -> GPIO 21

Installing Libraries

To work with BME280 we need two libraries

- **Adafruit BME280 Library**
- **Unified Sensor Library**


Library Manager

Type Installed Topic Sensors Filter your search...

Adafruit BME280 Library
by **Adafruit** Version **2.1.2** **INSTALLED**
Arduino library for BME280 sensors. Arduino library for BME280 humidity and pressure sensors.
More info

Adafruit Unified Sensor
by **Adafruit** Version **1.0.3** **INSTALLED**
Required for all Adafruit Unified Sensor based libraries. A unified sensor abstraction layer used by many Adafruit sensor libraries.

DHT sensor library
by **Adafruit** Version **1.3.0** **INSTALLED**
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors
More info

Adafruit BME280 Library was written to hide away all the complexities so that we can issue simple commands to read the temperature, relative humidity & barometric pressure data.

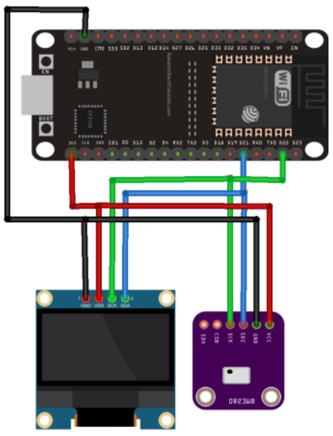
The BME280 sensor library uses the Adafruit Sensor support backend.

Reading temperature, humidity, pressure, and estimate altitude is as simple as using:

- **bme.readTemperature()** – reads temperature in Celsius;
- **bme.readHumidity()** – reads absolute humidity;
- **bme.readPressure()** – reads pressure in hPa (hectopascal = millibar);
- **bme.readAltitude(SEALEVELPRESSURE_HPA)** – estimates altitude in meters based on the pressure at the sea level.

Weather Station

By combining both OLED & BME280 we can create our mini weather station onboard which can be used to display Temperature, Humidity & Pressure.



When we have multiple devices with different addresses, it is trivial how to set them up:

- connect both peripherals to the ESP32 SCL and SDA lines;
- in the code, refer to each peripheral by its address;

Because the OLED and the BME280 have different addresses, we can use the same SDA and SCL lines without any problem. The OLED display address is **0x3C** and the BME280 address is **0x76.**