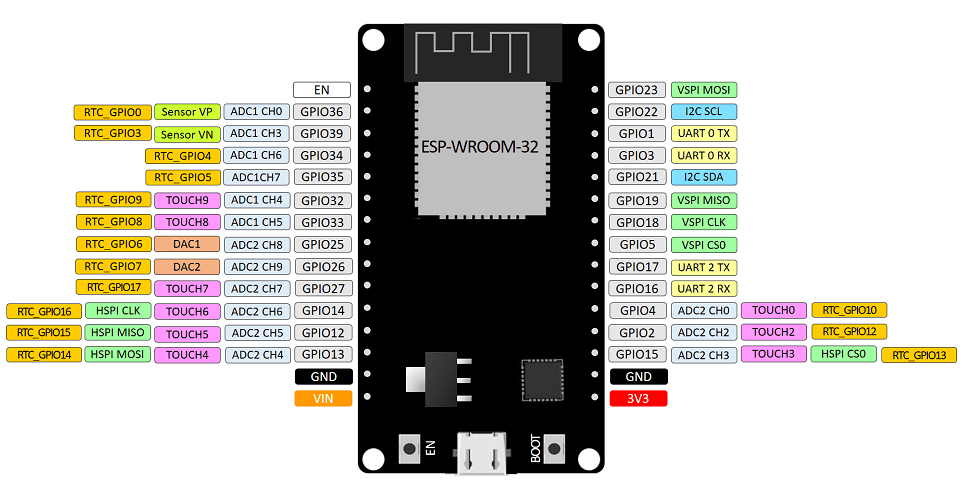
Module 1: IoT & Embedding Computing.

**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.



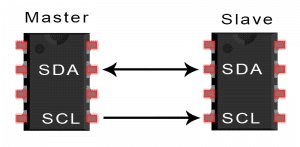
**Session Objectives:**

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:

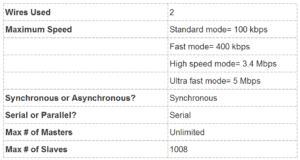
[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Single-Master-Single-Slave.png)

**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.

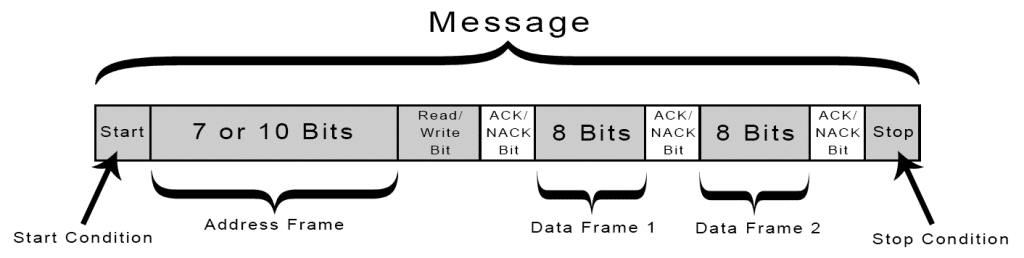
2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line).

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.

[](https://www.circuitbasics.com/wp-content/uploads/2016/02/Basics-of-the-I2C-Communication-Protocol-Specifications-Table.png)

**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:

[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Message-Frame-and-Bit-2.png)

**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

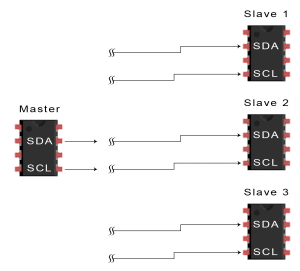
After the master detects the ACK bit from the slave, the first data frame is ready to be sent.

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.

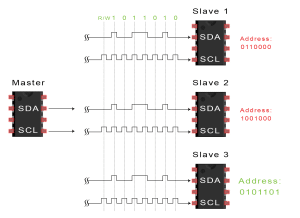
After all of the data frames have been sent, the master can send a stop condition to the slave to halt the transmission. The stop condition is a voltage transition from low to high on the SDA line after a low to high transition on the SCL line, with the SCL line remaining high.

STEPS OF I2C DATA TRANSMISSION

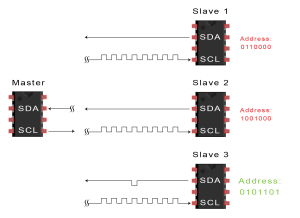
1. The master sends the start condition to every connected slave by switching the SDA line from a high voltage level to a low voltage level before switching the SCL line from high to low:

[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Data-Transmission-Diagram-START-CONDITION-3.png)

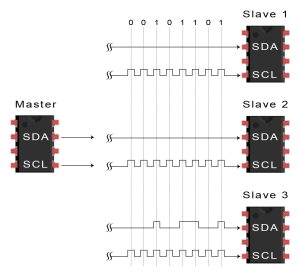
2. The master sends each slave the 7- or 10-bit address of the slave it wants to communicate with, along with the read/write bit:

[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Data-Transmission-Diagram-ADDRESS-FRAME-2.png)

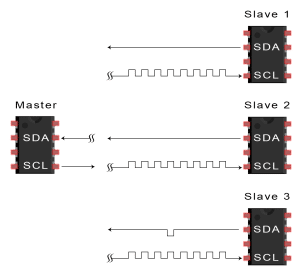
3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the SDA line low for one bit. If the address from the master does not match the slave’s own address, the slave leaves the SDA line high.

[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Data-Transmission-Diagram-ACK-Bit-Slave-to-Master-2.png)

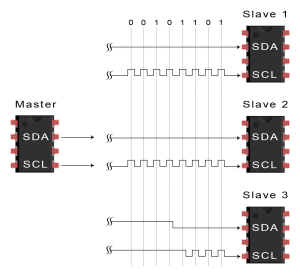
4. The master sends or receives the data frame:

[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Data-Transmission-Diagram-Data-Frame.png)

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:

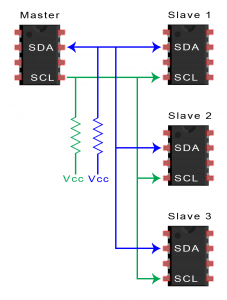
[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Data-Transmission-Diagram-ACK-bit-slave-to-master-2A.png)

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

[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Data-Transmission-Diagram-Stop-Condition.png)

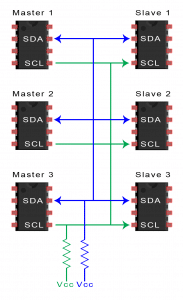
SINGLE MASTER WITH MULTIPLE SLAVES

Because I2C uses addressing, multiple slaves can be controlled from a single master. With a 7-bit address, 128 (27) unique address are available. Using 10-bit addresses is uncommon, but provides 1,024 (210) 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:

[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Single-Master-Multiple-Slaves-2.png)

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:

[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Multiple-Masters-Multiple-Slaves-2.png)

**Screens**

**Types**

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

Character-based describes [programs](https://www.webopedia.com/definitions/program/) capable of displaying only [ASCII](https://www.webopedia.com/definitions/ascii/) (and [extended ASCII](https://www.webopedia.com/definitions/extended-ascii/)) [characters](https://www.webopedia.com/definitions/character/). Character-based programs treat a [display screen](https://www.webopedia.com/definitions/display-screen/) as an [array](https://www.webopedia.com/definitions/array/) of boxes, each of which can hold one character. When in [text mode](https://www.webopedia.com/definitions/text-mode/), for example, PC screens are typically divided into 25 rows and 80 [columns](https://www.webopedia.com/definitions/column/). In contrast, [graphics-based](https://www.webopedia.com/definitions/graphics-based/) programs treat the display screen as an array of millions of [pixels](https://www.webopedia.com/definitions/pixel/). Characters and other [objects](https://www.webopedia.com/definitions/object/) 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**

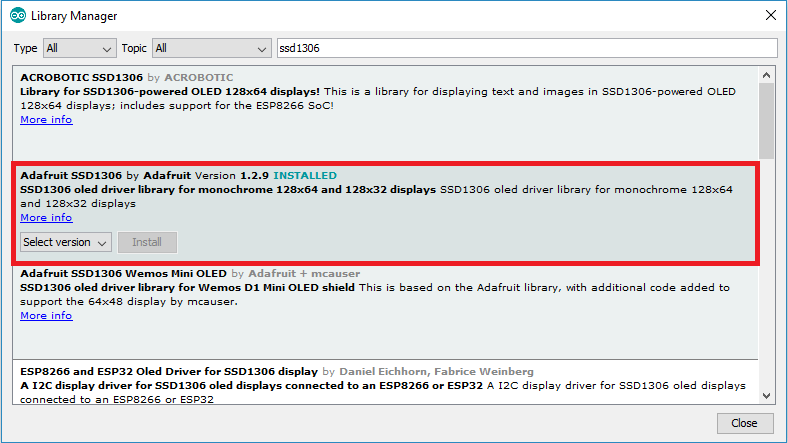
To work with OLEDs, we need to install primarily three libraries

* Wire.h
* Adafruit\_GFX.h
* Adafruit\_SSD1306.h

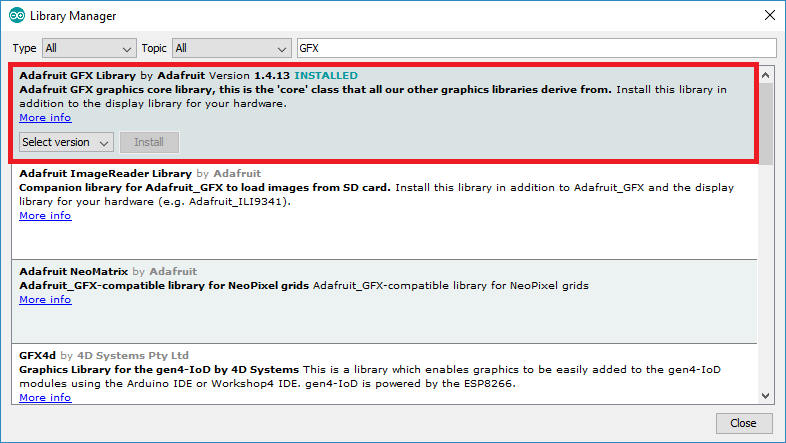
To control the OLED display you need the [adafruit\_SSD1306.h](https://github.com/adafruit/Adafruit_SSD1306) and the [adafruit\_GFX.h](https://github.com/adafruit/Adafruit-GFX-Library" \t "_blank) libraries. Follow the next instructions to install those libraries.

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.



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



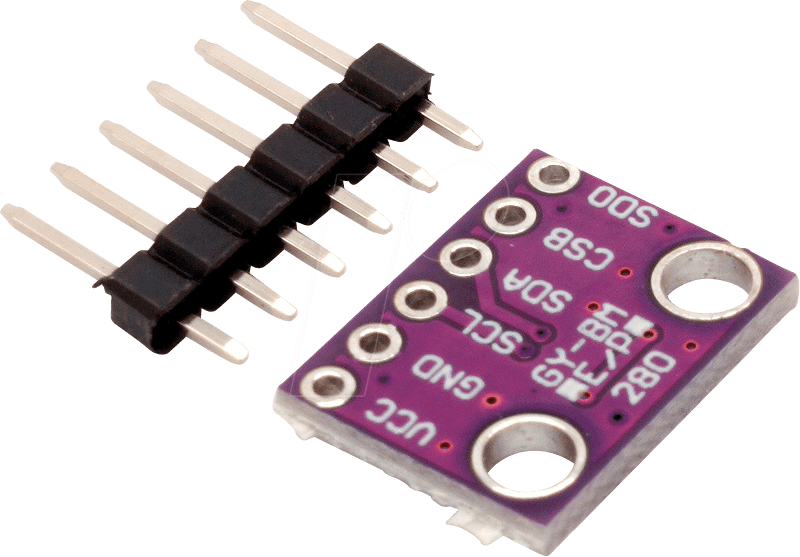
4. After installing the libraries, restart your Arduino IDE.

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**

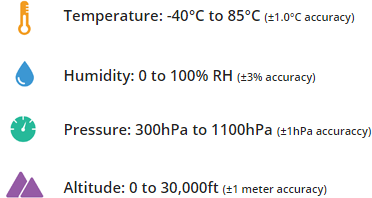
The BME280 sensor module reads barometric pressure, temperature, and humidity. Because pressure changes with altitude, you can also estimate altitude. There are several versions of this sensor module. The BME280 sensor uses I2C or SPI communication protocol to exchange data with a microcontroller. BME280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



BME280

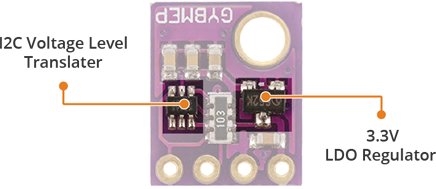
This precision sensor can measure relative humidity from 0 to 100% with ±3% accuracy, barometric pressure from 300Pa to 1100 hPa with ±1 hPa absolute accuracy, and temperature from -40°C to 85°C with ±1.0°C accuracy.

The pressure measurements are so precise (low altitude noise of 0.25m), you can even use it as an altimeter with ±1 meter accuracy.



**Power Requirement**

The module comes with an on-board [LM6206](https://www.mikrocontroller.net/attachment/193855/LM6206N3.pdf) 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 **0x76HEX** and can be changed to **0x77HEX**easily with the solder jumper besides chip.

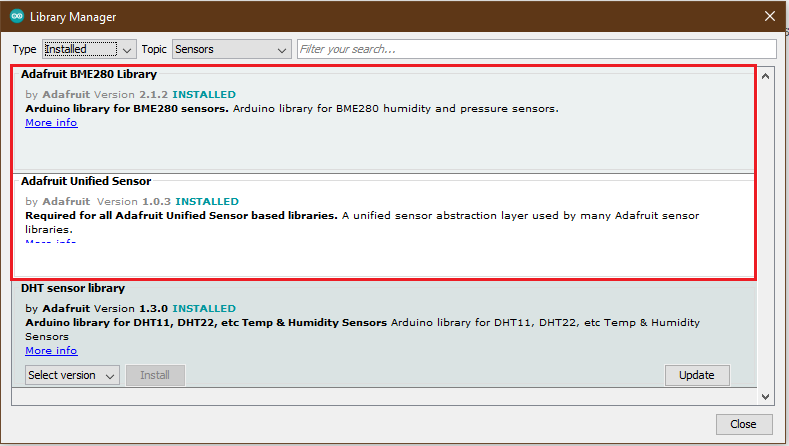
**BME280 Connections:**

* 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**



[Adafruit BME280 Library](https://github.com/adafruit/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.

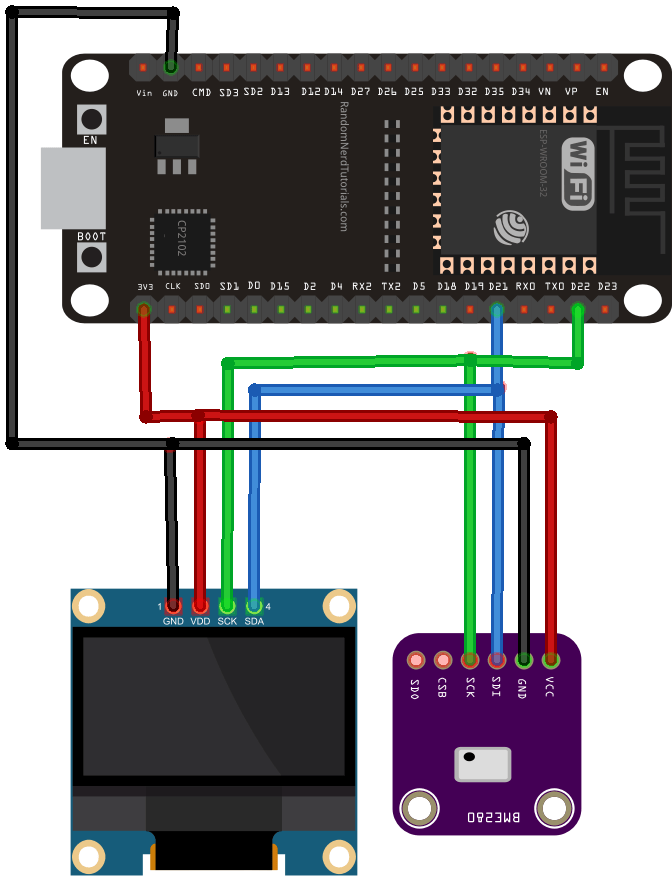
The BME280 sensor library uses the [Adafruit Sensor support backend](https://github.com/adafruit/Adafruit_Sensor).

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.**