5.I2C Device interfacing using ESP 32(Practical)

Add these all below Libraries:

Adafruit\_Sensor.h library

Adafruit\_MPU6050.h library

Adafruit\_GFX.h library(link:

Adafruit BusIO library

**Introduction**

The MPU6050 is a popular MEMS (Microelectromechanical systems) sensor module that is used to measure acceleration, rotational motion, and orientation in 3D space. The module consists of a 3-axis accelerometer and a 3-axis gyroscope, which are combined on a single integrated circuit. The MPU6050 is widely used in various applications, including drones, robotics, and gaming devices.

The ESP32 is a powerful microcontroller that is widely used in the Internet of Things (IoT) applications. It features Wi-Fi and Bluetooth connectivity, making it an excellent choice for building IoT devices. In this article, we will discuss how to interface the MPU6050 with ESP32 and how to read its sensor values on 16×2 i2c LCD and SSD1306 Oled display module.

**Amazon Links:**

[ESP32 development board](https://amzn.to/41JmZCL)

[MPU-6050 sensor module](https://amzn.to/3mPN2cJ)

[SSD1306 OLED display](https://amzn.to/41HSf5H)

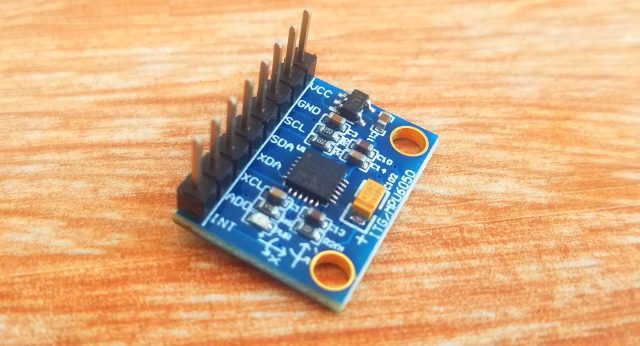
[16×2 i2c Lcd Display](https://amzn.to/41tan2P)

[Breadboard](https://amzn.to/41jKL8B)

[Jumper wires](https://amzn.to/43VkEqq)

\*Please Note: These are affiliate links. I may make a commission if you buy the components through these links. I would appreciate your support in this way!

**What is MPU6050?**



The MPU6050 is a widely-used and adaptable motion tracking component that integrates a 3-axis accelerometer and a 3-axis gyroscope into a single chip, resulting in a 6-axis motion tracking device.It is designed and manufactured by InvenSense, which is now part of TDK Corporation. The MPU6050 has gained popularity in the hobbyist and DIY electronics community due to its low cost, ease of use, and wide range of applications.

The gyroscope and accelerometer in the MPU6050 work together to provide a comprehensive motion tracking solution. The gyroscope measures angular velocity around each of its three axes (x, y, and z), while the accelerometer measures acceleration along each of the same three axes. By combining the data from both sensors, the MPU6050 is able to provide accurate information about the orientation and movement of the device it is embedded in.

One of the key advantages of the MPU6050 is its high level of integration. Traditionally, motion tracking systems required separate gyroscopes and accelerometers, which had to be carefully calibrated and synchronized to work together. With the MPU6050, both sensors are combined on a single chip, and the necessary calibration and synchronization is performed automatically.

The MPU6050 communicates with an external microcontroller through a standard I2C bus interface. This allows developers to easily integrate the device into their projects, and to access the full range of features offered by the MPU6050.

Some of the most common applications for the MPU6050 include:

* Robotics: The MPU6050 is frequently used in robotics applications to provide precise measurements of the orientation and movement of robotic limbs and other components. This information can be used to control the movement of the robot in real-time.
* Gaming: The MPU6050 is used in gaming controllers and other input devices to provide motion sensing capabilities. This allows users to control their games using natural movements and gestures, rather than relying on traditional button-based controls.
* Virtual Reality: The MPU6050 is also used in virtual reality (VR) applications, where it can be used to track the movement of a user’s head or body. This information can then be used to update the VR environment in real-time, creating a more immersive and realistic experience for the user.
* Drones: The MPU6050 is often used in drones and other unmanned aerial vehicles (UAVs) to provide precise measurements of the vehicle’s orientation and movement. This information can be used to stabilize the vehicle in flight, and to control its movements in real-time.
* Fitness Tracking: The MPU6050 can be used in wearable fitness devices to track the movement and activity of the user. This information can be used to monitor exercise performance, track progress over time, and provide feedback to the user.

In addition to its core motion tracking capabilities, the MPU6050 also includes a range of advanced features, such as a built-in temperature sensor, programmable digital filters, and an on-board FIFO buffer for storing data. These features make the MPU6050 a versatile and powerful tool for a wide range of applications.

Overall, the MPU6050 is a highly capable and versatile motion tracking device that has found wide use in a variety of industries and applications. Its low cost, ease of use, and high level of integration have made it a popular choice among hobbyists, DIY enthusiasts, and professional developers alike.

**MPU6050 Accelerometer & Gyroscope Pinout:**



The MPU6050 pinout consists of a total of 8 pins arranged in a single row on the device. These pins are used for power supply, communication with external microcontrollers, and to provide access to the various sensors and features of the device.

The following is a brief description of each pin on the MPU6050:

* VCC: This pin is used to supply power to the MPU6050. It requires a voltage input between 2.3V and 3.4V.
* GND: This pin is used to connect the MPU6050 to ground.
* SDA: This pin is used for bi-directional data transfer between the MPU6050 and an external microcontroller. It is connected to the microcontroller’s data line in an I2C communication setup.
* SCL: This pin is used to clock data transfer between the MPU6050 and the external microcontroller in an I2C communication setup.
* AD0: This pin is used to change the device address of the MPU6050. When it is connected to VCC, the device address is set to 0x69. When it is connected to GND, the device address is set to 0x68.
* INT: This pin is used to indicate when the MPU6050 has new data available for processing. It can be configured to generate an interrupt signal to the external microcontroller, allowing for real-time processing of data.
* XDA: This pin provides access to the raw data from the X-axis accelerometer.
* XCL: This pin provides access to the raw data from the X-axis gyroscope.

In addition to these pins, the MPU6050 also has a built-in temperature sensor that can be accessed through the I2C interface. This allows developers to monitor the temperature of the device and take appropriate actions to prevent overheating.

It is important to note that the MPU6050 is a surface mount device (SMD), which means that it does not have traditional through-hole pins. Instead, it is designed to be mounted directly onto a printed circuit board (PCB) using a soldering process. This requires specialized equipment and techniques, making it a bit more challenging to work with than some other types of electronic components.

To simplify the process of working with the MPU6050, there are a number of breakout boards and modules available that include the device pre-soldered onto a small PCB with pins or headers that can be easily connected to an external microcontroller. These modules often include additional features such as voltage regulators, level shifters, and filtering circuits to ensure reliable operation of the MPU6050 in a wide range of applications.

In summary, the MPU6050 pinout consists of 8 pins that are used for power supply, communication with external microcontrollers, and to provide access to the various sensors and features of the device. Understanding how to work with these pins is essential for integrating the MPU6050 into your projects and applications.

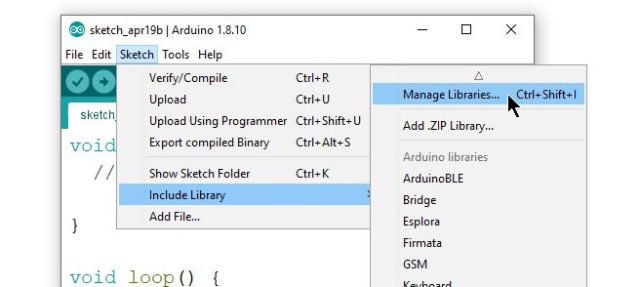
**Required Libraries installation:**

To interface an MPU6050 with an ESP32 microcontroller, you will need to install the following libraries:

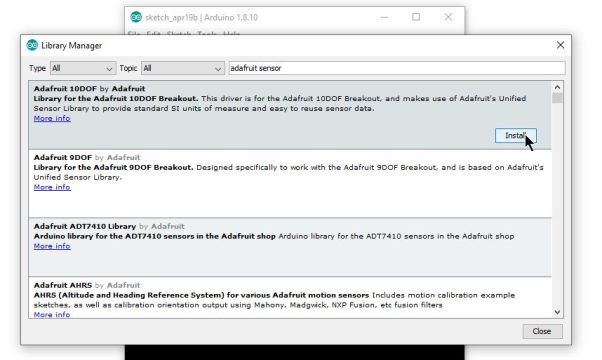
**Adafruit\_Sensor.h installation:**

The Adafruit\_Sensor.h library is a sensor library for Arduino and compatible microcontrollers that provides a common interface for reading sensor data. Here are the steps to install the Adafruit\_Sensor.h library in the Arduino IDE for ESP32:

Open the Arduino IDE and click on “Sketch” in the menu bar, then navigate to “Include Library” and select “Manage Libraries.”



In the Library Manager, search for “Adafruit Sensor” in the search bar. The library should appear in the search results.



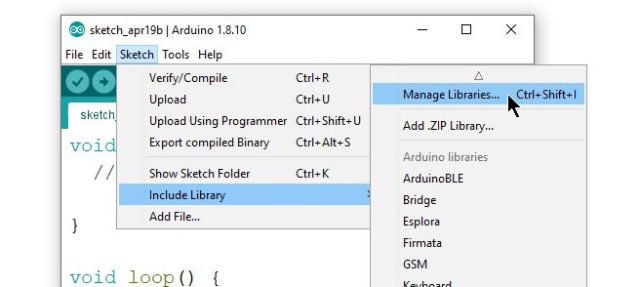
Click on the Adafruit Sensor library and then click the “Install” button to install the library.

After installation is complete, close the Library Manager and return to the Arduino IDE.

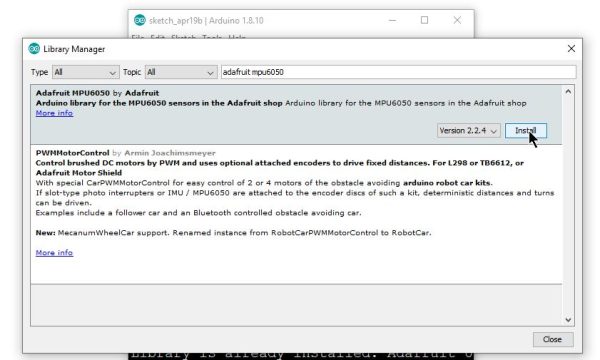
**Adafruit\_MPU6050.h installation**

the Adafruit\_MPU6050.h library is a sensor library for Arduino and compatible microcontrollers that provides an interface for reading data from the MPU6050 accelerometer and gyroscope sensor. Here are the steps to install the Adafruit\_MPU6050.h library in the Arduino IDE for ESP32:

Open the Arduino IDE and click on “Sketch” in the menu bar, then navigate to “Include Library” and select “Manage Libraries.”



In the Library Manager, search for “Adafruit MPU6050” in the search bar. The library should appear in the search results.



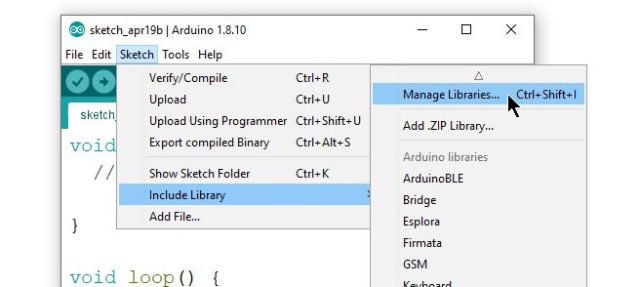
Click on the Adafruit MPU6050 library and then click the “Install” button to install the library.

After installation is complete, close the Library Manager and return to the Arduino IDE.

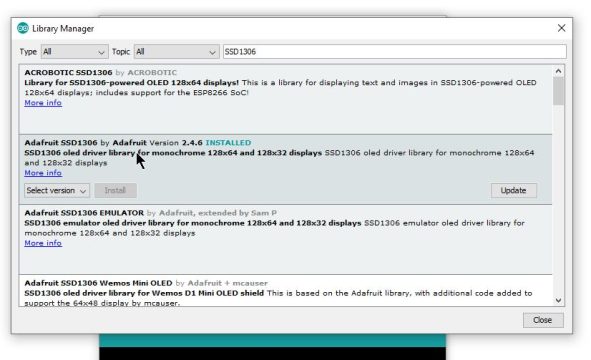
**Adafruit\_SSD1306.h installation**

The Adafruit\_SSD1306.h library is a display library for Arduino and compatible microcontrollers that provides an interface for driving OLED displays based on the SSD1306 controller chip. Here are the steps to install the Adafruit\_SSD1306.h library in the Arduino IDE for ESP32:

Open the Arduino IDE and click on “Sketch” in the menu bar, then navigate to “Include Library” and select “Manage Libraries.”



In the Library Manager, search for “Adafruit SSD1306” in the search bar. The library should appear in the search results.



Click on the Adafruit SSD1306 library and then click the “Install” button to install the library in my case I installed it already that why it display the update button.

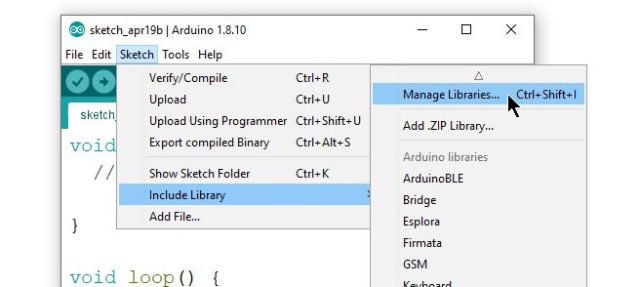
After installation is complete, close the Library Manager and return to the Arduino IDE.

To use the library in your code, you will need to include the Adafruit\_SSD1306.h library at the top of your sketch.

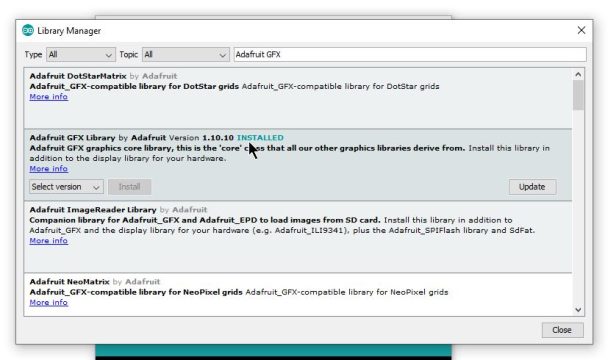
**Adafruit\_GFX.h installation**

The Adafruit\_GFX.h library is a graphics library for Arduino and compatible microcontrollers that provides a set of common graphics operations, such as drawing lines, rectangles, and circles. Here are the steps to install the Adafruit\_GFX.h library in the Arduino IDE for ESP32:

Open the Arduino IDE and click on “Sketch” in the menu bar, then navigate to “Include Library” and select “Manage Libraries.”



In the Library Manager, search for “Adafruit GFX” in the search bar. The library should appear in the search results.



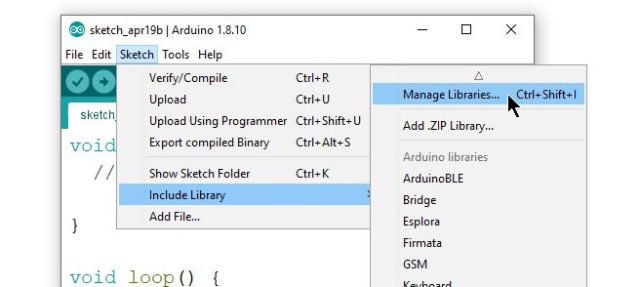
Click on the Adafruit GFX by adafruit library and then click the “Install” button to install the library but in my case I installed that why it display the update button.

After installation is complete, close the Library Manager and return to the Arduino IDE.

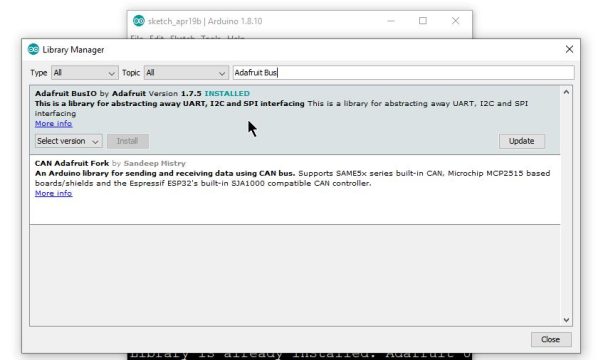
**Adafruit busIO installation**

The Adafruit BusIO library is a communication library for Arduino and compatible microcontrollers that provides a set of common communication protocols, such as I2C, SPI, and UART. Here are the steps to install the Adafruit BusIO library in the Arduino IDE for ESP32:

Open the Arduino IDE and click on “Sketch” in the menu bar, then navigate to “Include Library” and select “Manage Libraries.”



In the Library Manager, search for “Adafruit BusIO” in the search bar. The library should appear in the search results.



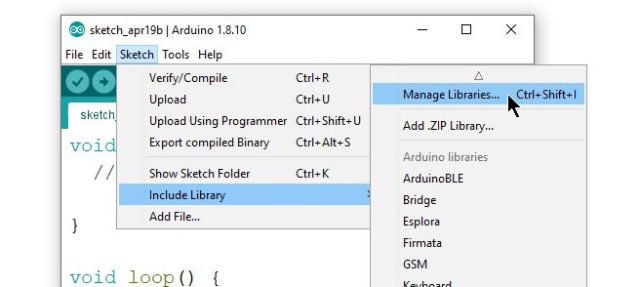
Click on the Adafruit Bus library and then click the “Install” button to install the library but in my case I installed it already that why it display the update button.

After installation is complete, close the Library Manager and return to the Arduino IDE.

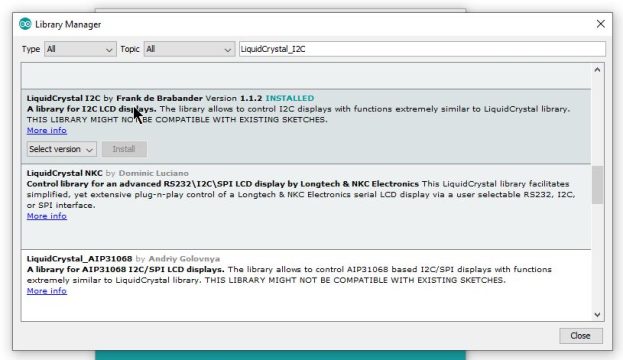
**16×2 i2c library installation**

To use a 16×2 I2C LCD display with your ESP32, you’ll need to install a library that supports the display. Here are the steps to install the LiquidCrystal\_I2C library in the Arduino IDE for ESP32:

Open the Arduino IDE and click on “Sketch” in the menu bar, then navigate to “Include Library” and select “Manage Libraries.”



In the Library Manager, search for “LiquidCrystal\_I2C” in the search bar. The library should appear in the search results.



Click on the LiquidCrystal\_I2C by frank de brabander library and then click the “Install” button to install the library.

After installation is complete, close the Library Manager and return to the Arduino IDE.

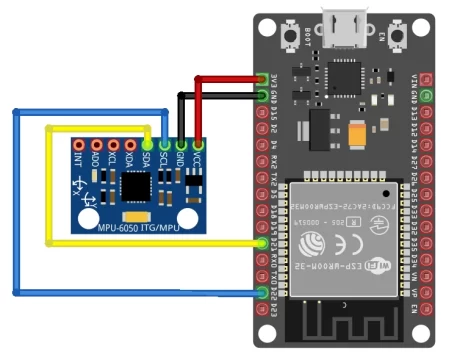
To use the library in your code, you will need to include the LiquidCrystal\_I2C library at the top of your sketch. You can do this by adding the following line to the top of your sketch:

#include <LiquidCrystal\_I2C.h>

**Display MPU6050 DATA on esp32 Serial Monitor:**

In this section, we will explore how to display MPU6050 data on the ESP32 serial monitor using the Arduino IDE.

**Mpu6050 with ESP32 Circuit diagram:**



The MPU6050 requires a 3.3V power supply, while the ESP32 can be powered by either 5V or 3.3V. Connect the VCC pin of the MPU6050 to a 3.3V pin on the ESP32, and connect the GND pin of the MPU6050 to a GND pin on the ESP32.

The MPU6050 communicates using the I2C protocol, which requires just two wires: SDA (data) and SCL (clock). The ESP32 has two dedicated I2C pins: GPIO21 for SDA and GPIO22 for SCL. Connect the SDA pin of the MPU6050 to GPIO21 on the ESP32, and connect the SCL pin of the MPU6050 to GPIO22 on the ESP32.

In order to use the I2C bus to communicate with the MPU6050, you need to configure the ESP32’s I2C bus. You can do this using the Wire library, which provides an easy-to-use interface for I2C communication.

Source code:

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_MPU6050.h>

Adafruit\_MPU6050 mpu;

void setup() {

  Serial.begin(9600);

  while (!Serial);

  Wire.begin();

  if (!mpu.begin()) {

    Serial.println("MPU6050 not found");

    while (1);

  }

  mpu.setAccelerometerRange(MPU6050\_RANGE\_2\_G);

  mpu.setGyroRange(MPU6050\_RANGE\_250\_DEG);

  mpu.setFilterBandwidth(MPU6050\_BAND\_5\_HZ);

}

void loop() {

  sensors\_event\_t a, g, temp;

  mpu.getEvent(&a, &g, &temp);

  Serial.print("Acceleration X: ");

  Serial.print(a.acceleration.x);

  Serial.print(", Y: ");

  Serial.print(a.acceleration.y);

  Serial.print(", Z: ");

  Serial.print(a.acceleration.z);

  Serial.println(" m/s^2");

  Serial.print("Rotation X: ");

  Serial.print(g.gyro.x);

  Serial.print(", Y: ");

  Serial.print(g.gyro.y);

  Serial.print(", Z: ");

  Serial.print(g.gyro.z);

  Serial.println(" rad/s");

  Serial.print("Temperature: ");

  Serial.print(temp.temperature);

  Serial.println(" degC");

  Serial.println("");

  delay(5000);

}

Output:

Acceleration X: 9.69, Y: -0.41, Z: -3.27 m/s^2

Rotation X: -0.01, Y: -0.05, Z: -0.02 rad/s

Temperature: 26.82 degC

Acceleration X: 9.70, Y: -0.41, Z: -3.29 m/s^2

Rotation X: -0.01, Y: -0.05, Z: -0.02 rad/s

Temperature: 26.81 degC

Acceleration X: 9.70, Y: -0.41, Z: -3.28 m/s^2

Rotation X: -0.01, Y: -0.05, Z: -0.02 rad/s

Temperature: 26.82 degC

**Code explanation:**

This code is written in C++ and is designed to read data from an MPU6050 sensor using an esp32 board. The MPU6050 is a 6-degree-of-freedom (6-DOF) motion tracking sensor, which includes a 3-axis accelerometer and a 3-axis gyroscope.

Let’s go through the code line by line:

|  |  |
| --- | --- |
| 1  2  3 | #include <Wire.h>  #include <Adafruit\_Sensor.h>  #include <Adafruit\_MPU6050.h> |

These are the libraries being used in the code. The Wire library is used for I2C communication, while the Adafruit\_Sensor and Adafruit\_MPU6050 libraries are used to interface with the MPU6050 sensor.

|  |  |
| --- | --- |
| 1 | Adafruit\_MPU6050 mpu; |

This line creates an instance of the Adafruit\_MPU6050 class called “mpu”. This will be used to communicate with the sensor.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | void setup() {    Serial.begin(9600);    while (!Serial);      Wire.begin();    if (!mpu.begin()) {      Serial.println("MPU6050 not found");      while (1);    }      mpu.setAccelerometerRange(MPU6050\_RANGE\_2\_G);    mpu.setGyroRange(MPU6050\_RANGE\_250\_DEG);    mpu.setFilterBandwidth(MPU6050\_BAND\_5\_HZ);    } |

The setup function runs once when the program starts. The first line initializes serial communication with a baud rate of 9600. The “while (!Serial);” line waits until a connection is established before continuing. This is useful when the Arduino board is connected to a computer for debugging.

The next line, “Wire.begin()”, initializes the I2C bus. Then, the “if (!mpu.begin())” statement checks if the MPU6050 sensor is detected on the I2C bus. If it is not detected, the program prints an error message and enters an infinite loop.

The next three lines set the accelerometer and gyroscope ranges to 2g and 250 degrees per second (dps), respectively. The filter bandwidth is also set to 5 Hz.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | void loop() {    sensors\_event\_t a, g, temp;    mpu.getEvent(&a, &g, &temp);      Serial.print("Acceleration X: ");    Serial.print(a.acceleration.x);    Serial.print(", Y: ");    Serial.print(a.acceleration.y);    Serial.print(", Z: ");    Serial.print(a.acceleration.z);    Serial.println(" m/s^2");      Serial.print("Rotation X: ");    Serial.print(g.gyro.x);    Serial.print(", Y: ");    Serial.print(g.gyro.y);    Serial.print(", Z: ");    Serial.print(g.gyro.z);    Serial.println(" rad/s");      Serial.print("Temperature: ");    Serial.print(temp.temperature);    Serial.println(" degC");      Serial.println("");    delay(500);  } |

The loop function runs repeatedly after the setup function has completed. In this function, three variables of type “sensors\_event\_t” are created to hold the accelerometer, gyroscope, and temperature data from the MPU6050 sensor.

The line “mpu.getEvent(&a, &g, &temp)” reads the sensor data into the “a”, “g”, and “temp” variables. The “&” symbol is used to pass the memory addresses of these variables to the “getEvent” function.

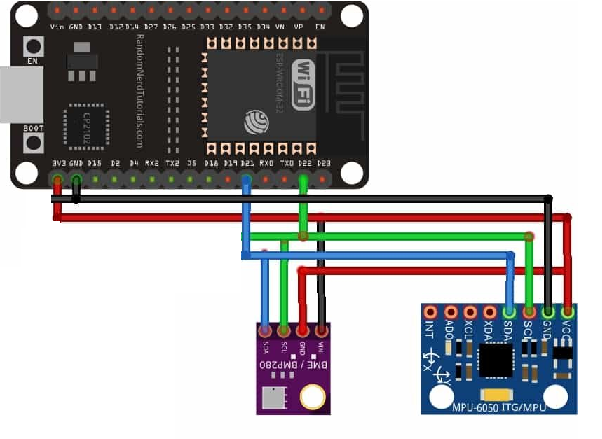
The next several lines print the sensor data to the serial monitor using the “Serial.print” function. The “println” function is used to print a newline character at the end of each line. The delay function is used to pause for 500 milliseconds before

5. I2C Device Scanning using ESP 32(Skilling)

The connections between the four devices which we are using can be seen in the table below.

|  |  |  |
| --- | --- | --- |
| ESP32 | BME280 | MPU6050 |
| 3.3V | VCC | VCC |
| GPIO21(I2C SDA) | SDA | SDA |
| GPIO22 (I2C SCL) | SCL | SCL |
| GND | GND | GND |

We have used the same connections as specified in the table above.



**ESP32 I2C Scanner Arduino Sketch**

Every I2C device has an address associated with it. The ESP32 uses this address to communicate with the slave via I2C

This code will scan for any I2C devices connected with ESP32 and will specify the number of devices with the address in the serial terminal.

Source code:

#include <Wire.h>

void setup() {

  Wire.begin();

  Serial.begin(115200);

  Serial.println("\nI2C Scanner");

    byte error, address;

  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;

  for(address = 1; address < 127; address++ ) {

    Wire.beginTransmission(address);

    error = Wire.endTransmission();

    if (error == 0) {

      Serial.print("I2C device found at address 0x");

      if (address<16) {

        Serial.print("0");

      }

      Serial.println(address,HEX);

      nDevices++;

    }

    else if (error==4) {

      Serial.print("Unknown error at address 0x");

      if (address<16) {

        Serial.print("0");

      }

      Serial.println(address,HEX);

    }

  }

  if (nDevices == 0) {

    Serial.println("No I2C devices found\n");

  }

  else {

    Serial.println("done\n");

  }

  delay(5000);

  Serial.print("i2c devices found:");

  Serial.println(nDevices);

}

void loop() {

}

Output:

ets Jun 8 2016 00:22:57

rst:0x1 (POWERON\_RESET),boot:0x13 (SPI\_FAST\_FLASH\_BOOT)

configsip: 0, SPIWP:0xee

clk\_drv:0x00,q\_drv:0x00,d\_drv:0x00,cs0\_drv:0x00,hd\_drv:0x00,wp\_drv:0x00

mode:DIO, clock div:1

load:0x3fff0030,len:1344

load:0x40078000,len:13964

load:0x40080400,len:3600

entry 0x400805f0

I2C Scanner

Scanning...

I2C device found at address 0x68

I2C device found at address 0x76

done

i2c devices found:2