# Arduino UNO R4 Wi-Fi USB HID with MPU6050 module

## Mouse/Keyboard Emulation (HID)

Difficult Level: 

## Build game controllers by emulating a mouse/keyboard.

In this tutorial you will learn how to emulate a mouse/keyboard using an Arduino UNO R4 Wi-Fi board with the Keyboard and Mouse APIs.

This feature can be used to create game controllers, keyboard extensions or other HID devices.

## Goals

The goals of this tutorials are:

* Learn how to emulate a keyboard (keypresses),
* learn how to emulate a mouse (x, y coordinates)
* learn how to use MPU6050 gyroscope & accelerator module to control (x, y coordinates)

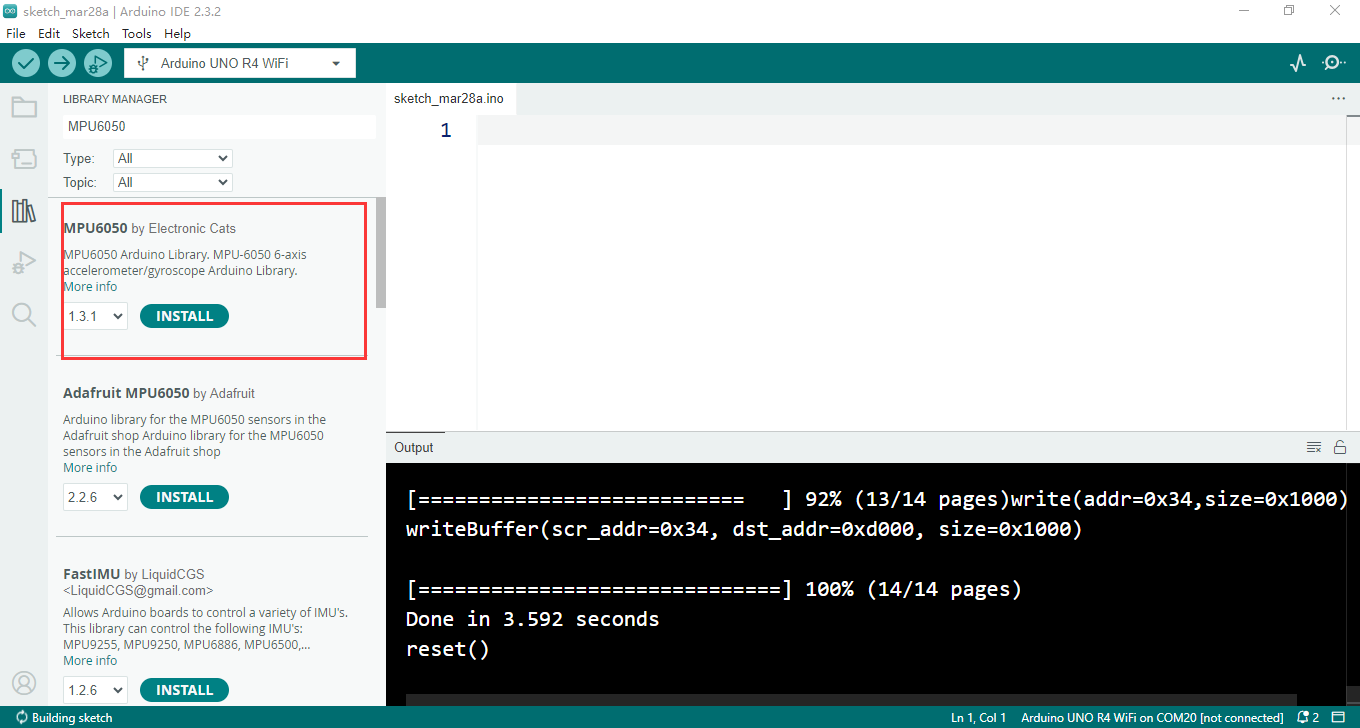
### Hardware

* 1 x Arduino UNO R4 Wi-Fi
* 1 x MPU6050 gyroscope & accelerator
* 10 x Jumper wire

### Software

* Arduino IDE (online or offline)
* UNO R4 Board Package
* Library: MPU6050

Install MPU6050 library by using library manager.



## Human Interface Device (HID)

Human interface devices (HID) are devices designed for humans (keyboards, mice, game controllers etc.), that frequently sends data over USB to a computer. When you press a key on a keyboard, you send data to a computer, which reads it and in turn activates the corresponding key.

The UNO R4 Wi-Fi has built-in support for HID, a feature found on most modern day development boards, but not on previous UNO revisions.

To turn your board into an HID, you can use the keyboard/mouse API that is built in to the Board Package. You can visit the documentation for this API in the language reference at:

* Keyboard: <https://www.arduino.cc/reference/en/language/functions/usb/keyboard/>
* Mouse: <https://www.arduino.cc/reference/en/language/functions/usb/mouse/>

## Sketch Upload Interference

As a consequence of the multi-processor design of the UNO R4 Wi-Fi board, uploads may fail with a "No device found on ..." error when the board is running a sketch that uses the HID capabilities.

For this reason, you should use the following procedure to upload under these conditions:

* 1. Press and release the button marked **"RESET"** on the board quickly twice. The LED marked **"L"** on the board should now be pulsing.
* 2. Select the port of the board from the menu in Arduino IDE. The port might have changed after the previous step, so make sure to verify that it is selected.
* 3. Upload your sketch as usual.

## Keyboard

To use keyboard functionalities, we need to include the library at the top of our sketch. The Keyboard class contains several methods that are useful to emulate a keyboard.

#include <Keyboard.h>

Keyboard.method()

### Functions

* Keyboard.begin()
* Keyboard.end()
* Keyboard.press()
* Keyboard.print()
* Keyboard.println()
* Keyboard.release()
* Keyboard.releaseAll()
* Keyboard.write()

### Keyboard Example

To emulate a keyboard, we can use the **press()** and **releaseAll()** methods. This will emulate a keypress, as well as releasing the keypress. The following example prints a "**w**" every second.

To see more examples, please refer to links below:

#include <Keyboard.h>

void setup() {

  Keyboard.begin();

  delay(1000);

}

void loop() {

  Keyboard.press('w');

  delay(100);

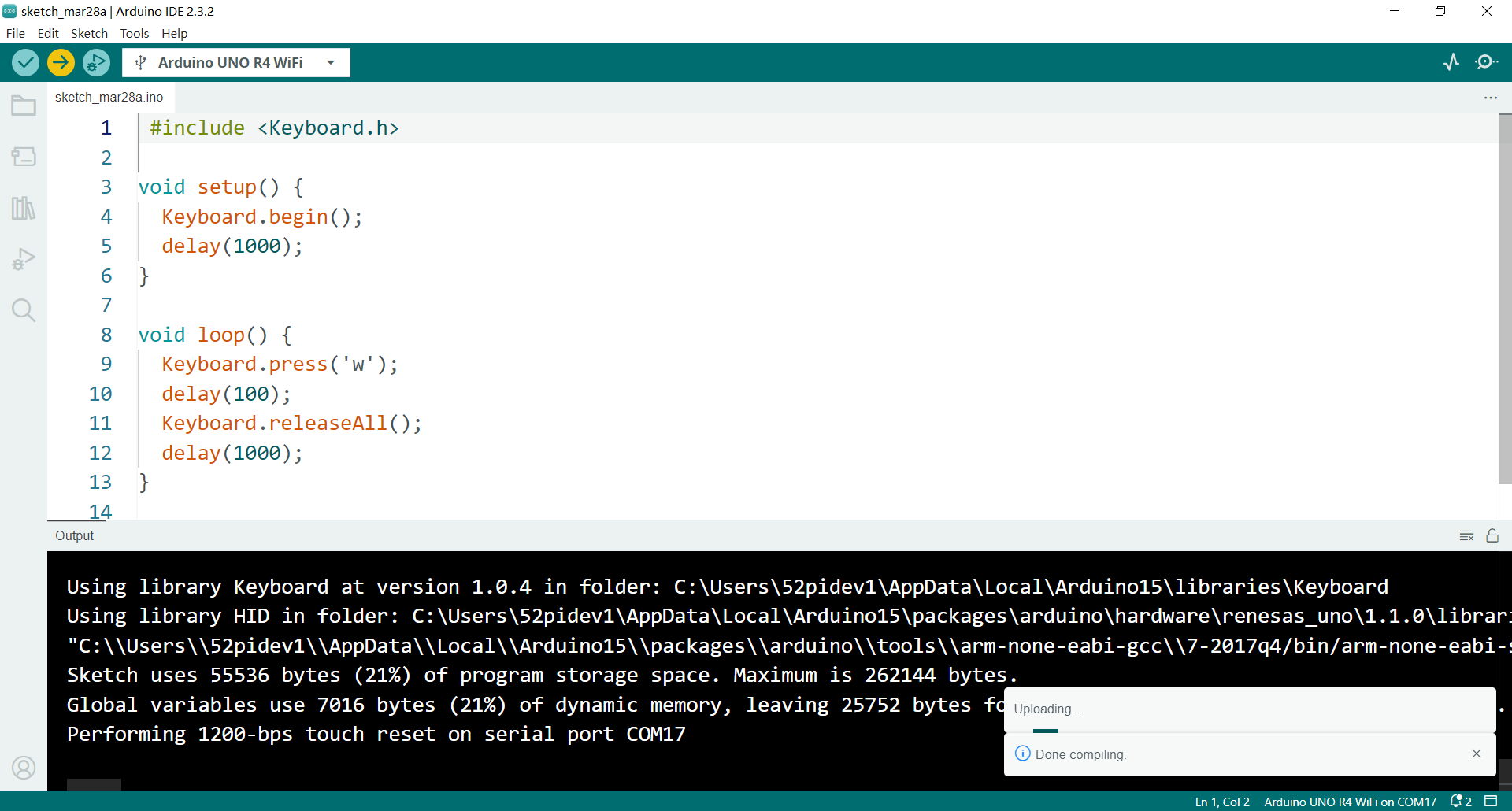
  Keyboard.releaseAll();

  delay(1000);

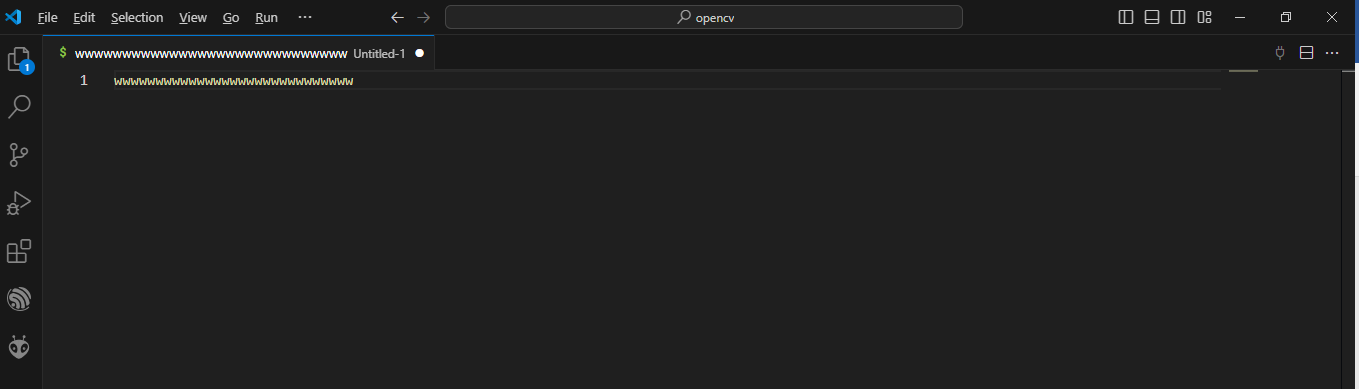
}



Compile it and upload the sketch:



Open a text editor, active the editor, you will see it will type “w” in every second.



## Mouse

To use mouse functionalities, we need to include the library at the top of our sketch. The Mouse class contains several methods that are useful to emulate a mouse.

#include <Mouse.h>

Mouse.method();

### Functions

* Mouse.begin()
* Mouse.click()
* Mouse.end()
* Mouse.move()
* Mouse.press()
* Mouse.release()
* Mouse.isPressed()

### Mouse Example

The following example moves both axis of mouse just slightly (10 points), back and forth.

To see more examples, please refer to links below:

#include <Mouse.h>

void setup() {

  Mouse.begin();

  delay(1000);

}

void loop() {

  Mouse.move(10,10);

  delay(1000);

  Mouse.move(-10,-10);

  delay(1000);

}

## Summary

In this tutorial, we have demonstrated some basic HID usage with the UNO R4 Wi-Fi. To view the full API, please refer to the following APIs:

## Project with MPU6050

### Principle

This Arduino code demonstrates the principle of interfacing an MPU6050 accelerometer and gyroscope sensor with a microcontroller, in this case, an Arduino board. The MPU6050 is a commonly used sensor module that combines a 3-axis accelerometer and a 3-axis gyroscope into a single chip. The code also utilizes the Keyboard library to simulate keyboard inputs, allowing data to be sent to a connected computer.

### Know about I2C

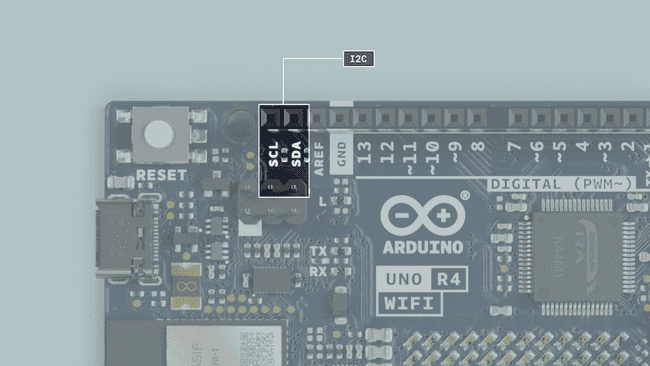
I2C lets you connect multiple I2C compatible devices in series using only two pins. The controller will send out information through the I2C bus to a 7-bit address, meaning that the technical limit of I2C devices on a single line is 128. Practically, you're never going to reach 128 devices before other limitations kick in.

The UNO R4 Wi-Fi has one I2C bus which is marked with SCL and SDA. They are shared with A4 (SDA) and A5 (SCL) which owners of previous UNO's are familiar with. The pull-ups are not mounted on the PCB but there are footprints to do so if needed.

The pins used for I2C on the UNO R4 Wi-Fi are the following:

**SDA - D14**

**SCL - D15**



we will connect a MPU6050 module to arduino board via I2C pins.

Here's how it works step by step:

### Initialization

The Arduino initializes serial communication for debugging purposes and sets up the I2C communication protocol to communicate with the MPU6050 module.

It also initializes the MPU6050 module and the mouse emulation mode.

### Data Acquisition

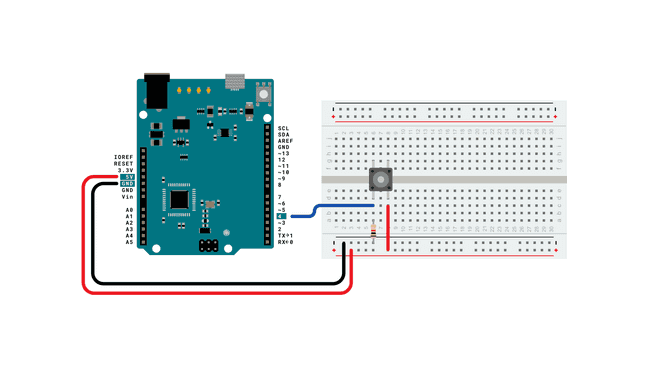
Inside the **loop()** function, the Arduino continuously reads the accelerometer data from the MPU6050 module.

The MPU6050 module measures acceleration along its x and y axes, which corresponds to the tilt or movement in those directions.

### Wiring Diagram

|  |  |
| --- | --- |
| Arduino UNO R4 Wi-Fi Board | MPU6050 |
| 3.3V | VCC |
| GND | GND |
| A4 | SDA |
| A5 | SCL |

|  |  |
| --- | --- |
| Arduino UNO R4 Wi-Fi Board | Button |
| D4 | 220Ω resistor |
| Button Connection please refer to following figure. | |



### Demo code:

#include <Wire.h>

#include "Keyboard.h"

#include <MPU6050.h>

// create MPU object and setting variables

#define OUTPUT\_READABLE\_ACCELGYRO

MPU6050 mpu;

int16\_t ax, ay, az;

int16\_t gx, gy, gz;

// button to change keyboard.

const int buttonPin = 4;         // input pin for pushbutton

int previousButtonState = HIGH;  // for checking the state of a pushButton

int counter = 0;

void setup() {

  Serial.begin(9600);

  // make the pushButton pin an input:

  pinMode(buttonPin, INPUT);

  Wire.begin();  // initializing I2C

  // Initialize the MPU6050

  mpu.initialize();

  Serial.println("Testing device connections...");

  Serial.println(mpu.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

  // Set keyboard mode

  Keyboard.begin();

}

void loop() {

  // read the pushbutton:

  int buttonState = digitalRead(buttonPin);

  // if the button state has changed,

  if ((buttonState != previousButtonState)

      // and it's currently pressed:

      && (buttonState == HIGH)) {

    // increment the button counter

    counter++;

    // Read data from MPU6050 accelerometer

    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    //     int axisx = map(ax, -16000, 16000, -10, 10);

    //     int axisy = map(ay, -16000, 16000, -10, 10);

    Keyboard.println("---------------Click button to grap data-----------");

    Keyboard.println("---------------Accelerator data-----------");

    Keyboard.print("Ax: ");

    Keyboard.print(ax);

    Keyboard.print("\t Ay: ");

    Keyboard.print(ay);

    Keyboard.print("\t Az: ");

    Keyboard.println(az);

    Keyboard.println("---------------Gyroscope data-------------");

    Keyboard.print("Gx: ");

    Keyboard.print(gx);

    Keyboard.print("\t Gy: ");

    Keyboard.print(gy);

    Keyboard.print("\t Gz: ");

    Keyboard.println(gz);

  }

  // save the current button state for comparison next time:

  previousButtonState = buttonState;

}

#### Code explanations

* **Include Libraries:**

#include <Wire.h>

#include "Keyboard.h"

#include <MPU6050.h>

This part includes the necessary libraries for I2C communication (Wire.h), keyboard emulation (Keyboard.h), and the MPU6050 sensor module (MPU6050.h).

* **Define MPU6050 Object and Variables:**

MPU6050 mpu;

int16\_t ax, ay, az;  // Variables to store accelerometer data

int16\_t gx, gy, gz;  // Variables to store gyroscope data

Here, an instance of the MPU6050 class is created, along with variables to store the accelerometer (ax, ay, az) and gyroscope (gx, gy, gz) data.

* **Setup Function:**

void setup() {

  // Initialize serial communication

  Serial.begin(9600);

  // Set button pin as input

  pinMode(buttonPin, INPUT);

  // Initialize I2C communication

  Wire.begin();

  // Initialize MPU6050

  mpu.initialize();

  // Print connection status of MPU6050

  Serial.println(mpu.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

  // Begin keyboard emulation

  Keyboard.begin();

}

In the setup() function, the serial communication is initialized, the button pin is set as an input, I2C communication is started, and the MPU6050 is initialized. It also checks if the MPU6050 is successfully connected and begins keyboard emulation.

* **Loop Function:**

void loop() {

  // Read button state

  int buttonState = digitalRead(buttonPin);

  // Check if button state has changed and is pressed

  if ((buttonState != previousButtonState) && (buttonState == HIGH)) {

    // Increment counter

    counter++;

    // Read data from MPU6050

    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // Print accelerometer and gyroscope data via keyboard emulation

    // (simulate keyboard inputs)

    Keyboard.println("---------------Click button to grab data-----------");

    Keyboard.println("---------------Accelerometer data-----------");

    Keyboard.print("Ax: ");

    Keyboard.print(ax);

    Keyboard.print("\t Ay: ");

    Keyboard.print(ay);

    Keyboard.print("\t Az: ");

    Keyboard.println(az);

    Keyboard.println("---------------Gyroscope data-------------");

    Keyboard.print("Gx: ");

    Keyboard.print(gx);

    Keyboard.print("\t Gy: ");

    Keyboard.print(gy);

    Keyboard.print("\t Gz: ");

    Keyboard.println(gz);

  }

  // Save current button state

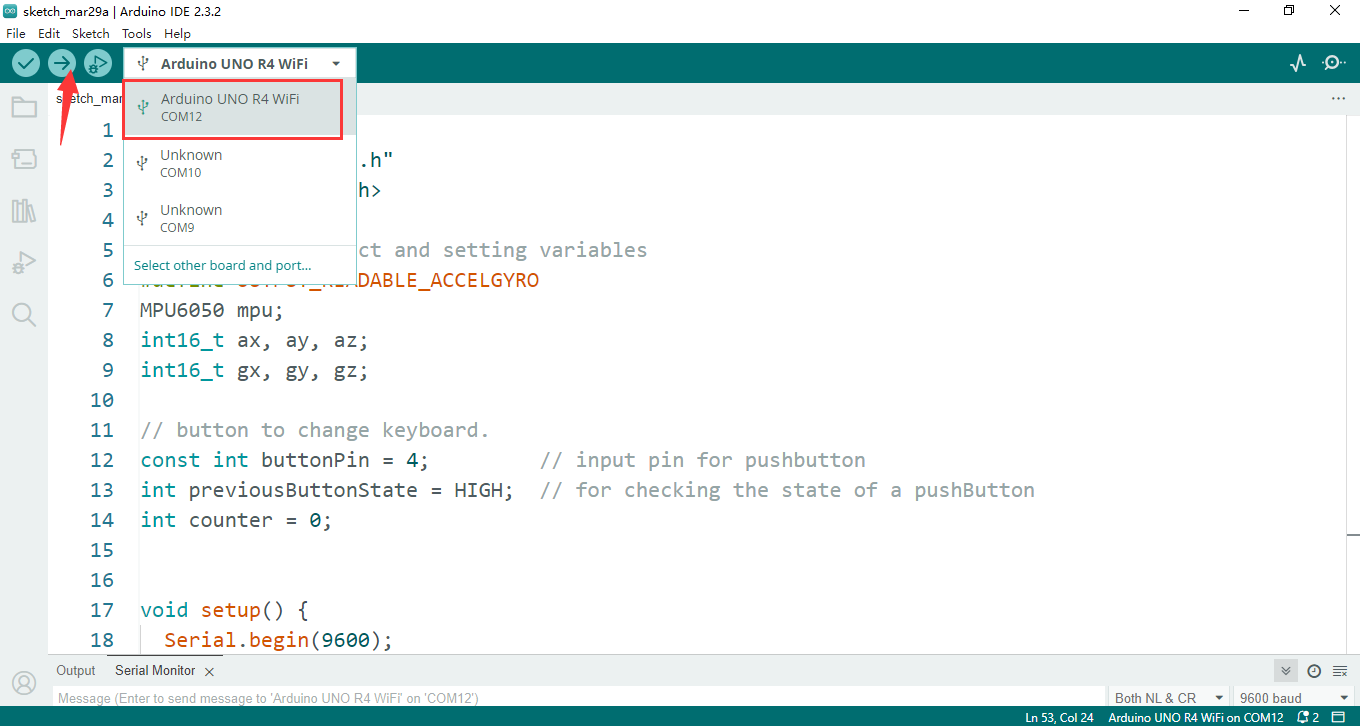
  previousButtonState = buttonState;

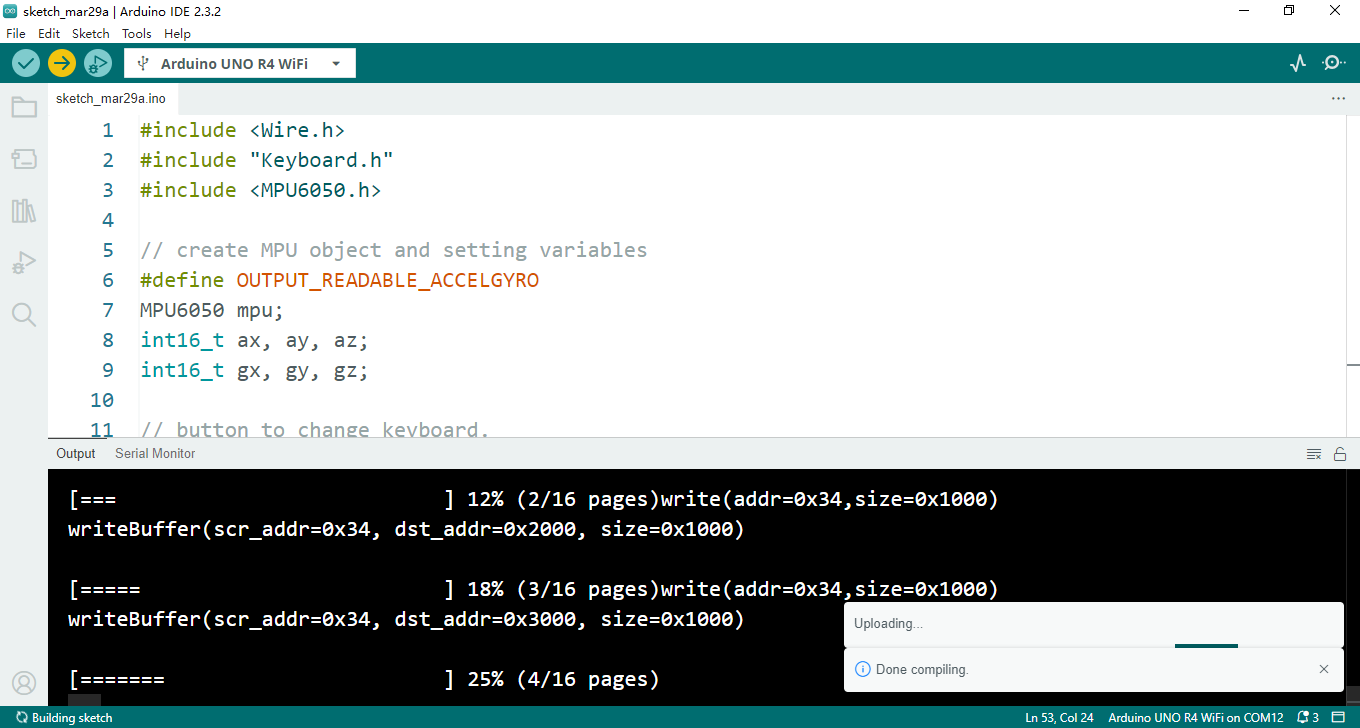
}

In the loop() function, it continuously checks the state of the button. If the button state has changed and is currently pressed, it increments the counter, reads data from the MPU6050 sensor, and prints the accelerometer and gyroscope data via keyboard emulation.

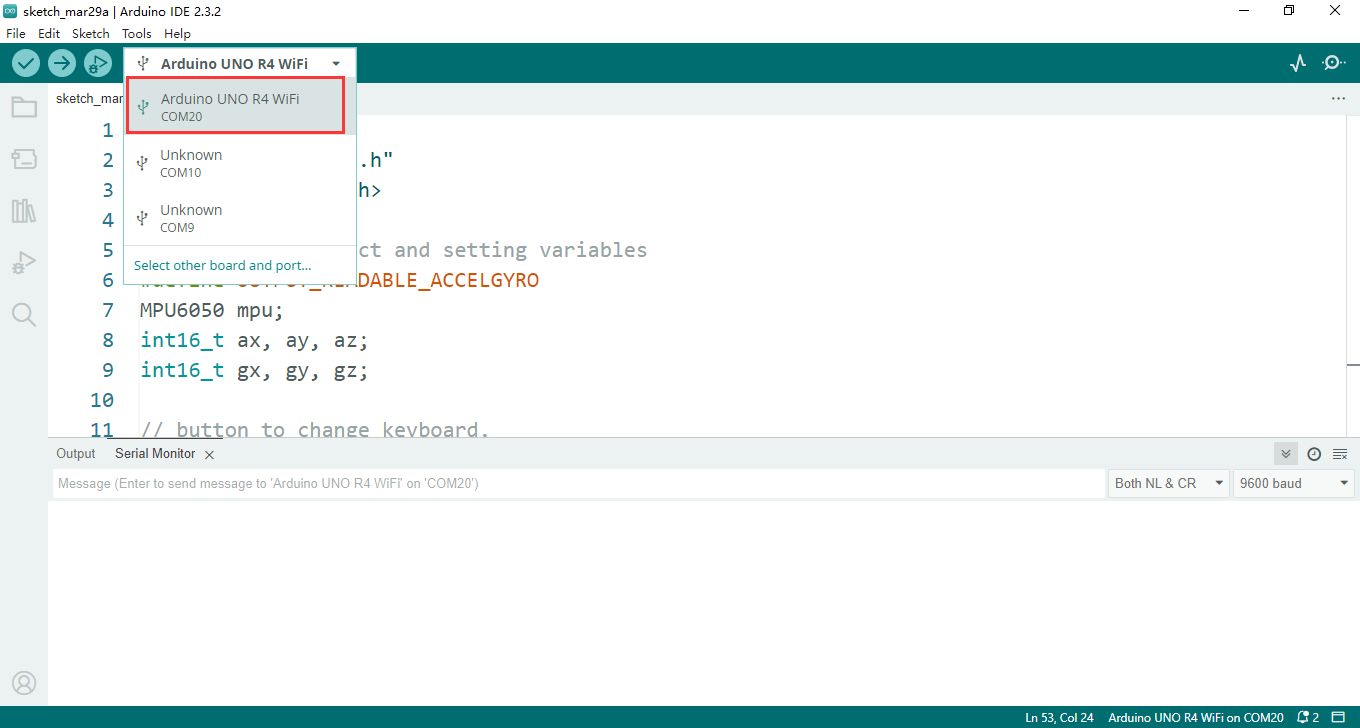
### Compile & upload sketch

Select the serial port and click the upload icon to compile and upload the sketch.





Once Upload finished, you will find that the serial port has been changed.

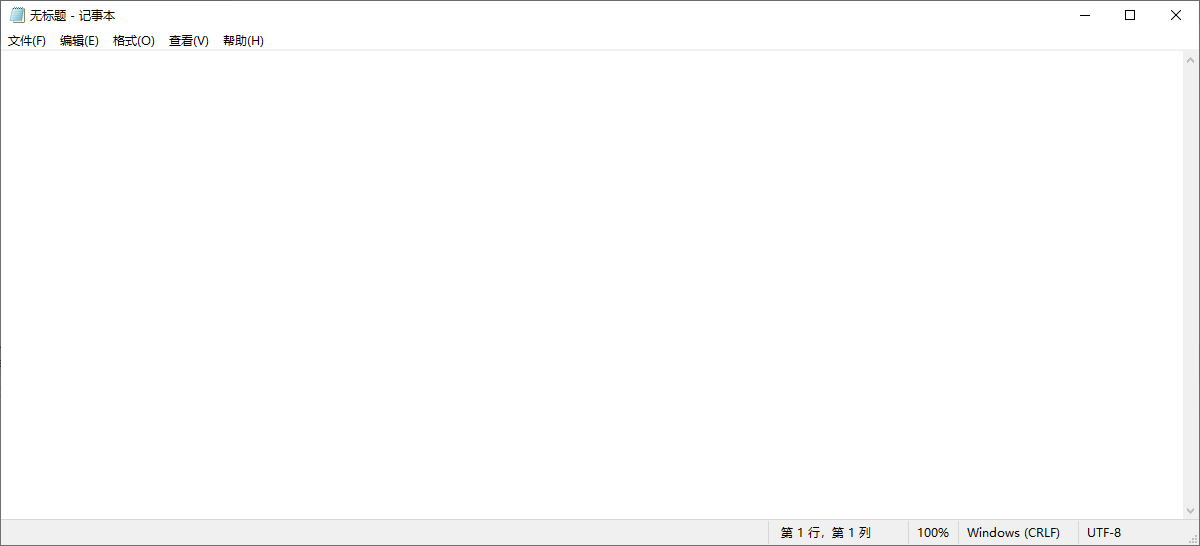


This serial port will not provide any serial data which you send from “Serial.print()” function.

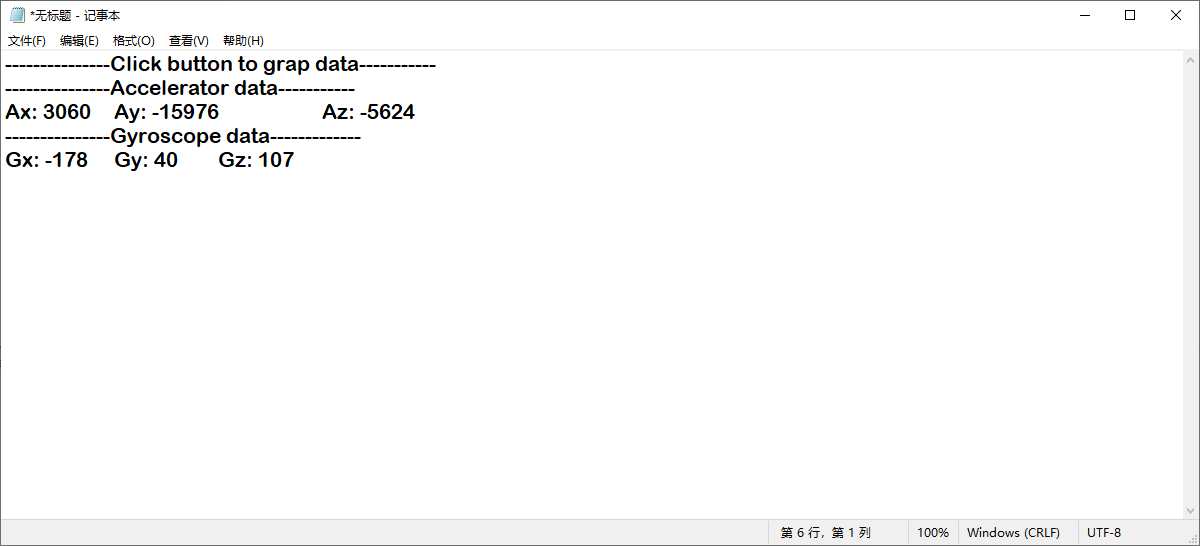
Next step is to open any text editor to test it.

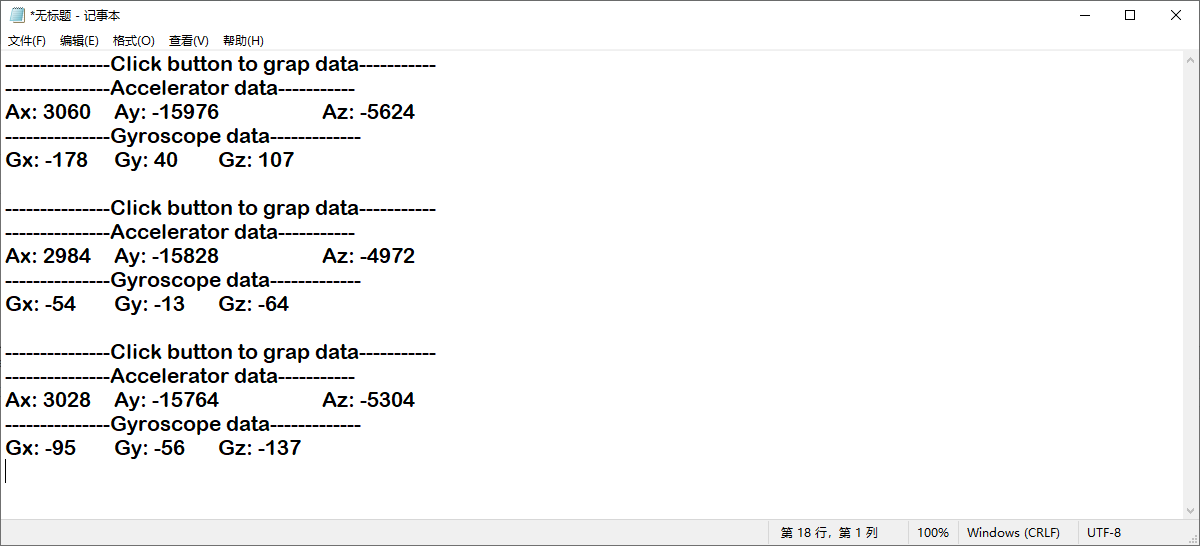
### Test Project

Open a text editor like notepad or Microsoft VSCODE editor.



Press Button.





Overall, the code retrieves data from the MPU6050 sensor and sends it to the connected computer as keyboard input whenever the button is pressed. This can be useful for testing or interfacing with programs on the computer that accept keyboard input.