BIT Lab 02

Wireless Motion Controller

Outline

- Demo
- Implementation

Demo

Implementation

Materials

(1) Breadboard x1
(2) NodeMCU ESP8266 x1
(3) 1k ohm resistor x1
(4) Button x1
(5) IMU sensor x1

Arduino

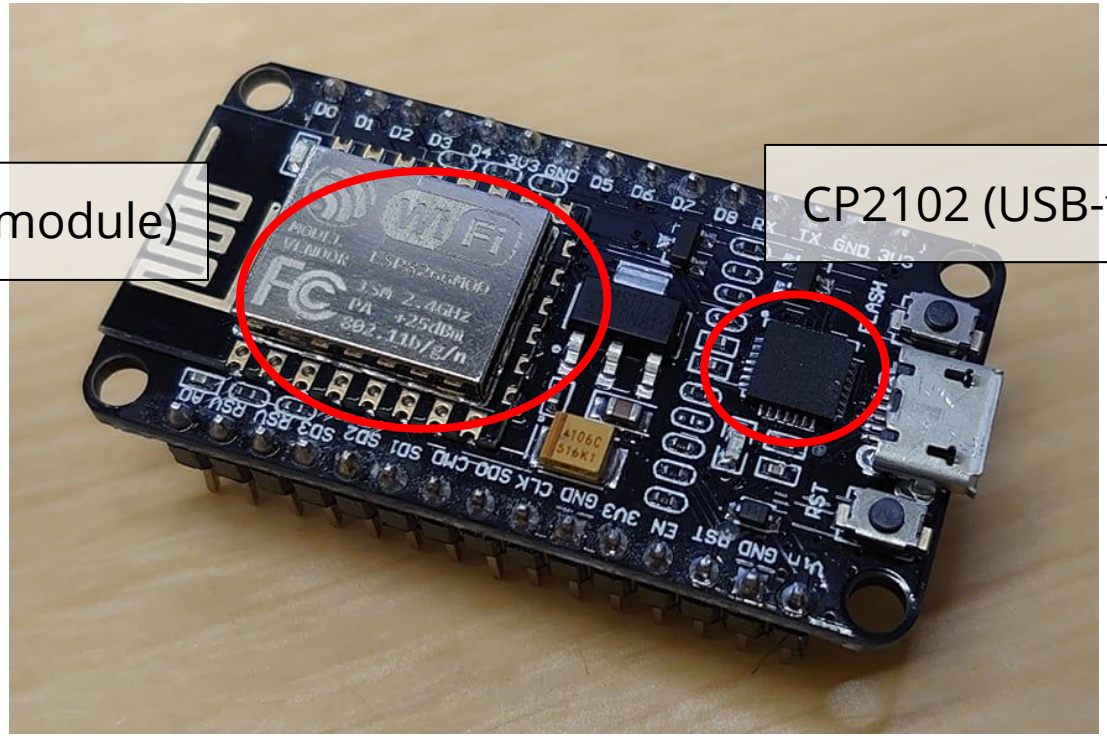
Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.

https://www.arduino.cc/

NodeMCU (Amica)

ESP8266 (wifi module)

CP2102 (USB-to-Serial bridge)



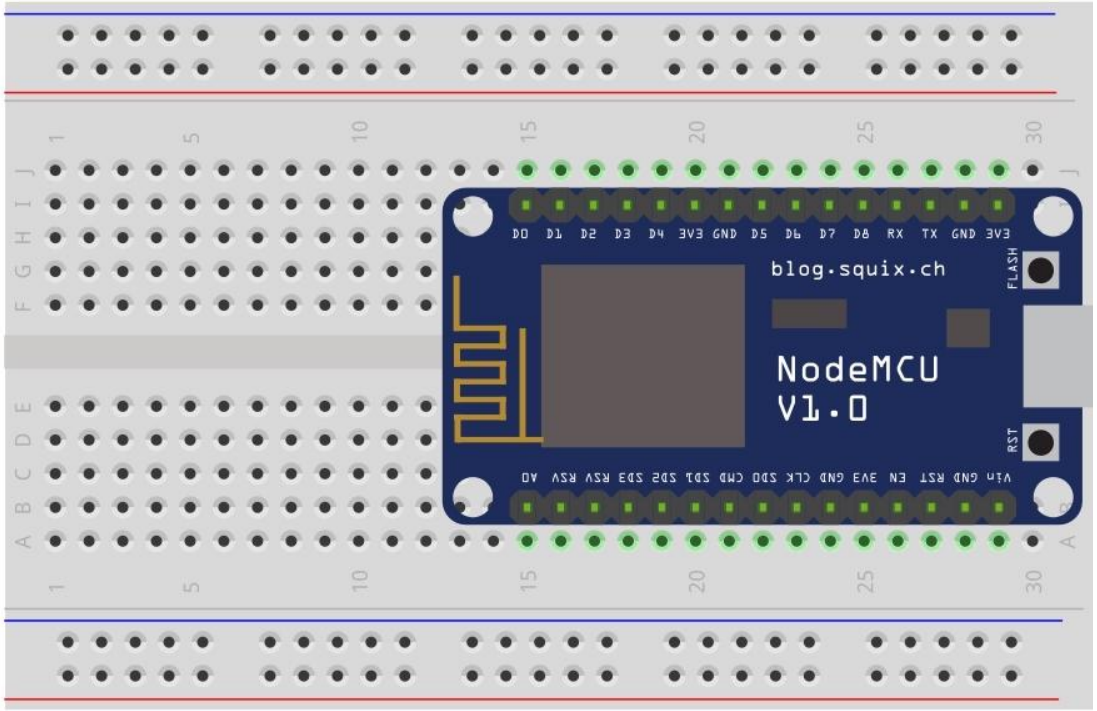
NodeMCU (Amica) - Installation

(1) Install CP2102 driver
- Follow the instructions written on [1].
- You can download the driver from NTU Cool directly.

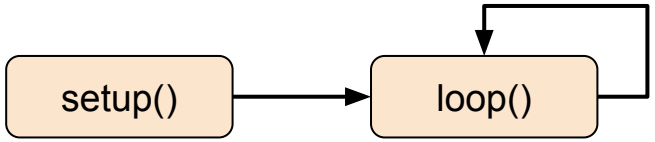
(2) Install NodeMCU library for Arduino IDE
- Follow the instructions written on [2].

Reference [1]: https://www.pololu.com/docs/0J7/all
Reference [2]: https://oranwind.org/-esp8266-nodemcu-zai-arduino-ide-she-ding-nodemcu/

Hardware overview



Hello World



For an Arduino code, it must contain two basic functions.

- ***setup()***
  It will be execute just one time when initializing the device.

- ***loop()***
  After initializing, the device run this function repeatedly.

HelloWorld

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World");
  delay(1000);
}
```

Basic functions

- Serial
  - ***Serial.begin(baud_rate)****:* Sets the data rate in bits per second for serial data transmission.
  - ***Serial.print(data)***, ***Serial.println(data)***: Print data to the serial port.
  - ***Serial.read()***: Reads incoming serial data.
- Digital I/O
  - **p*inMode(pin, mode[INPUT/OUTPUT])***: Configures the specified pin to behave either as an input or an output.
  - ***digitalWrite(pin, value)***, ***digitalRead(pin)***
- Time
  - ***delay(ms)***: Pauses the program for the amount of time.
  - ***millis()***: Returns the number of milliseconds passed since the Arduino board began running the current program.

Reference [1]: https://www.arduino.cc/reference/en/#functions

To build a wireless motion controller, we need to...

- Detect whether the button is pressed.
- Get the pose data from the IMU sensor.
- System receive all the data from the motion controller as user’s input.

Step (1/3)
Detect whether the button is pressed.

How to detect the button pressed?

Open

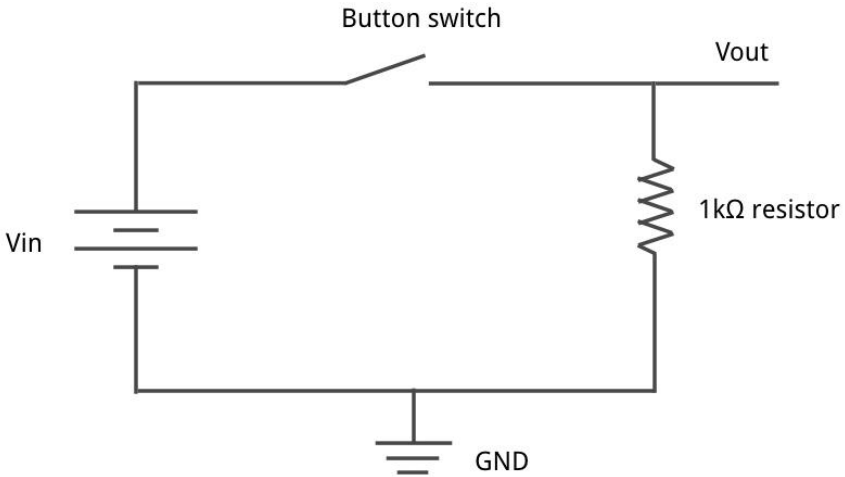
Close

A switch in **open state** can be considered as a resistor with **infinite ohm**.

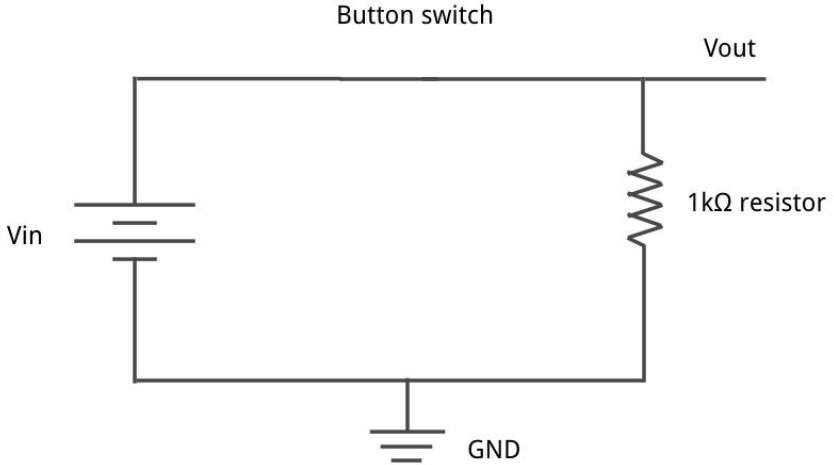
A switch in **close state** can be considered as a resistor with **zero ohm**.

How to detect the button pressed?

If we connect it to a resistor with constant ohm...

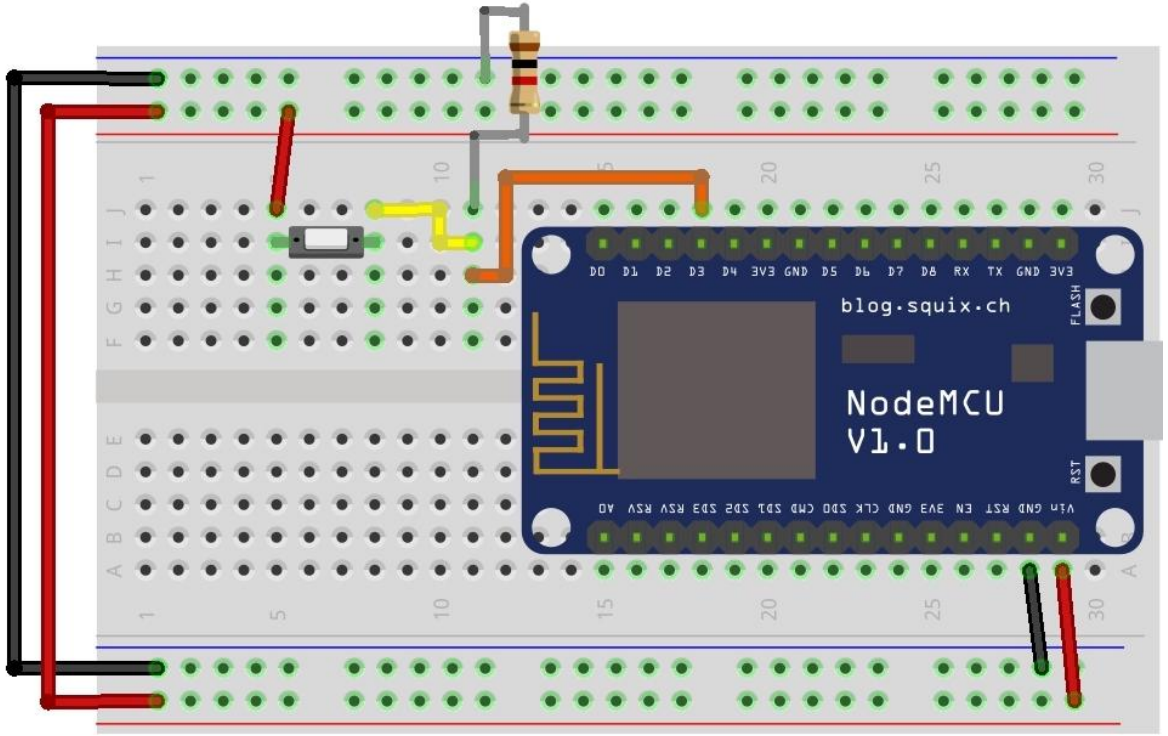


Vout = GND



Vout = Vin

Hardware overview

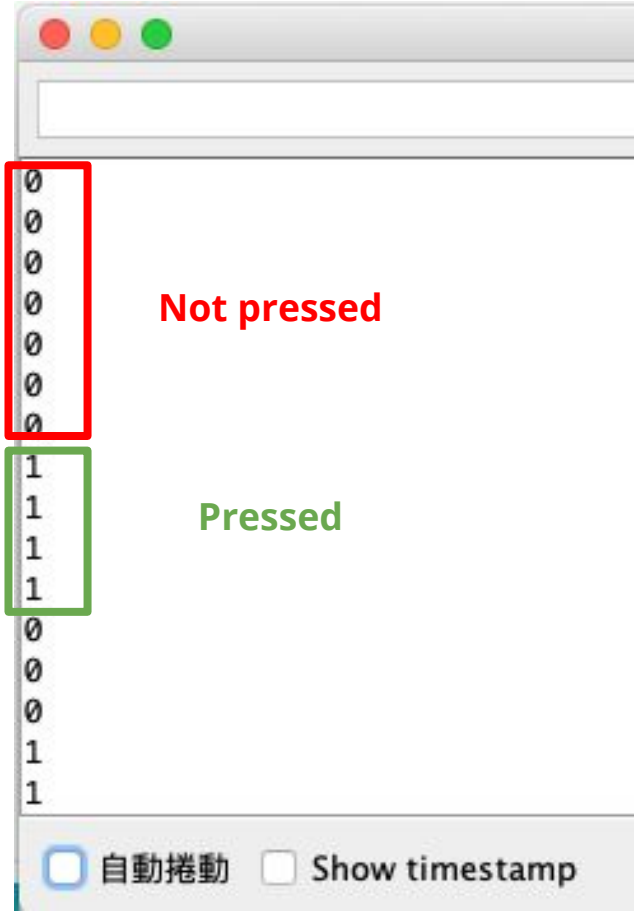


Time to implement!

Code - Button_template

Output

- 0, if the button isn't pressed
- 1, if the button is pressed



Step (2/3)
Get the pose data from the IMU sensor.

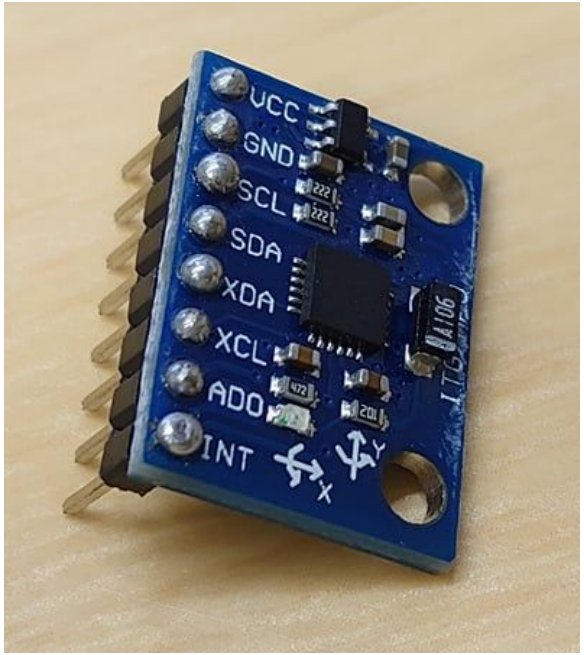
IMU sensor (GY-521)

To get the data from IMU sensor, we need to add the following two libraries to Arduino library.

- I2Cdev: I2C protocol
- MPU6050: IMU sensor

Arduino library path: *Documents/Arduino/libraries*

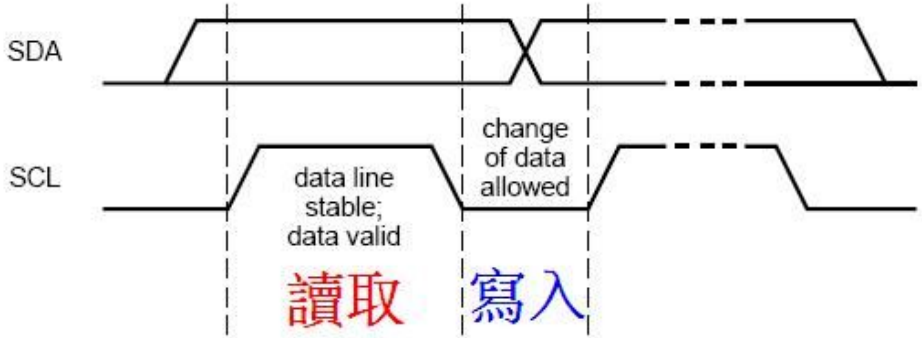
Please download these two libraries from NTU Cool. (IMU Libraries.zip)

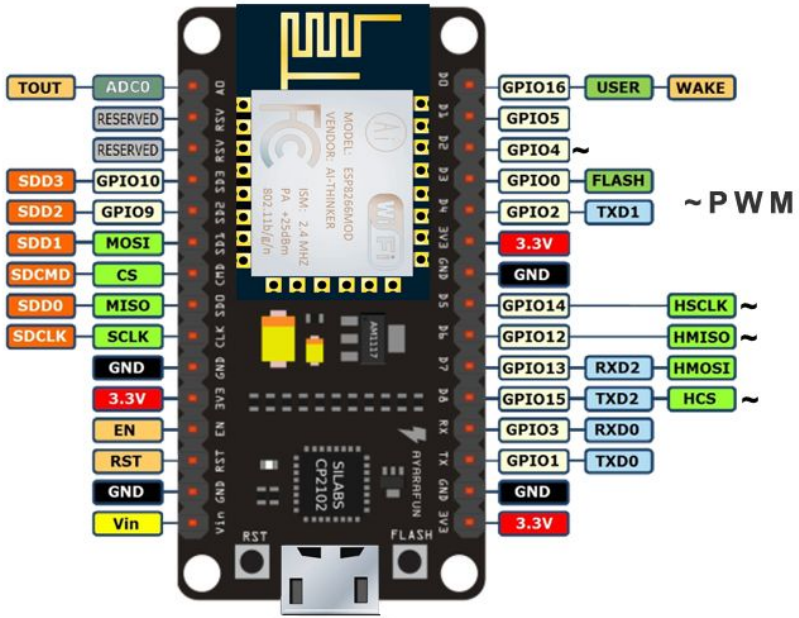


Reference [1]: http://ming-shian.blogspot.com/2014/05/arduino21mpu6050row-data.html

What is I2C?

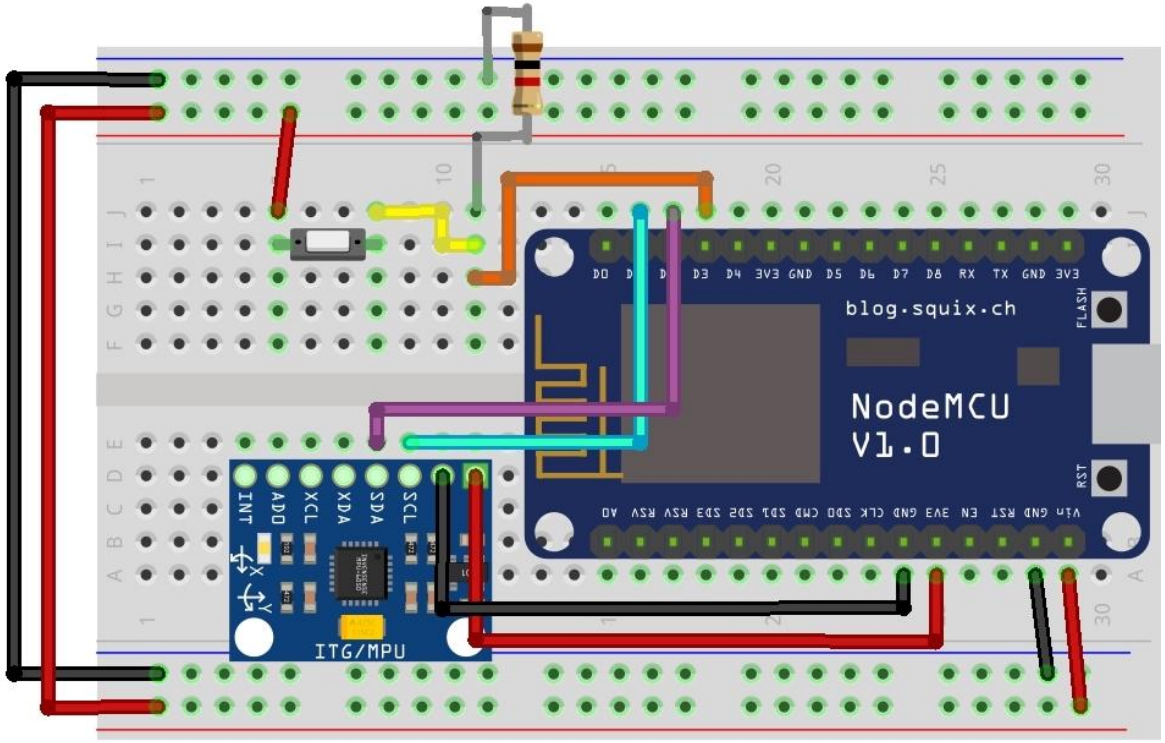
I2C is a serial communication protocol.





In Arduino, SDA=GPIO4 and SCL=GPIO5.

Hardware overview

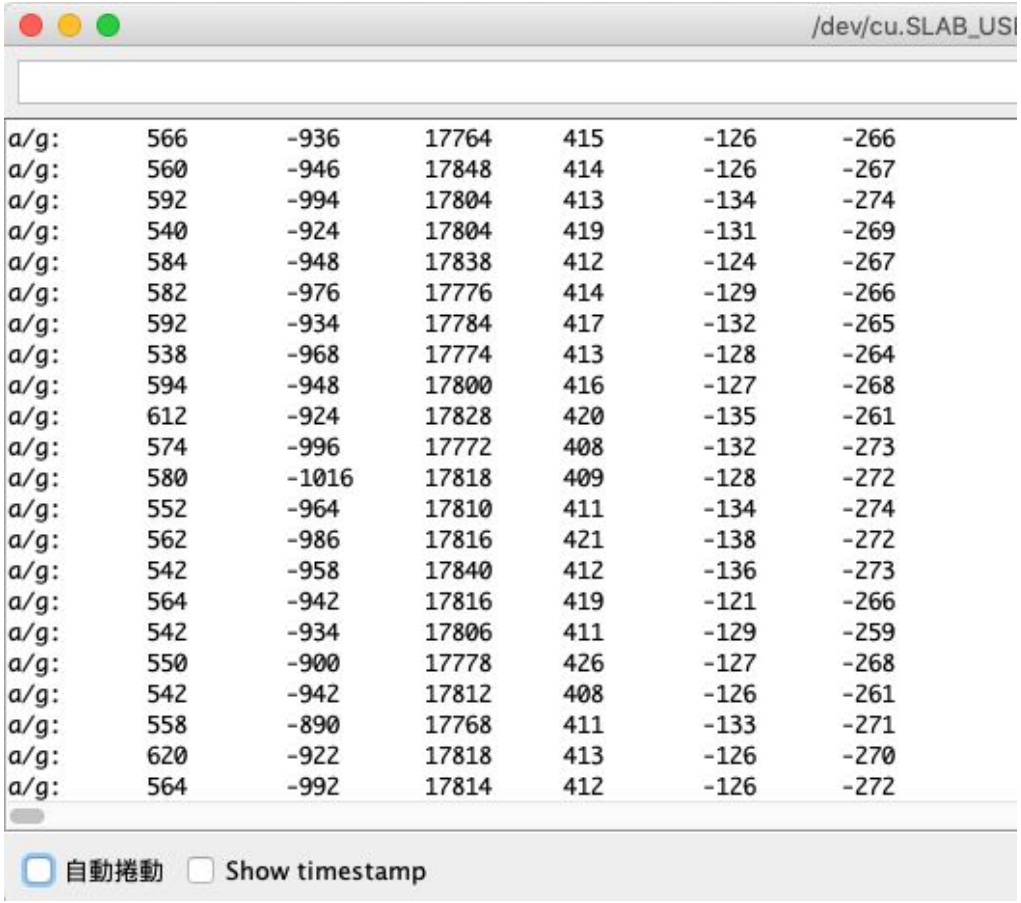


Time to implement!

Code - MPU6050_raw

Output (raw)

- ax, ay, az, gx, gy, gz



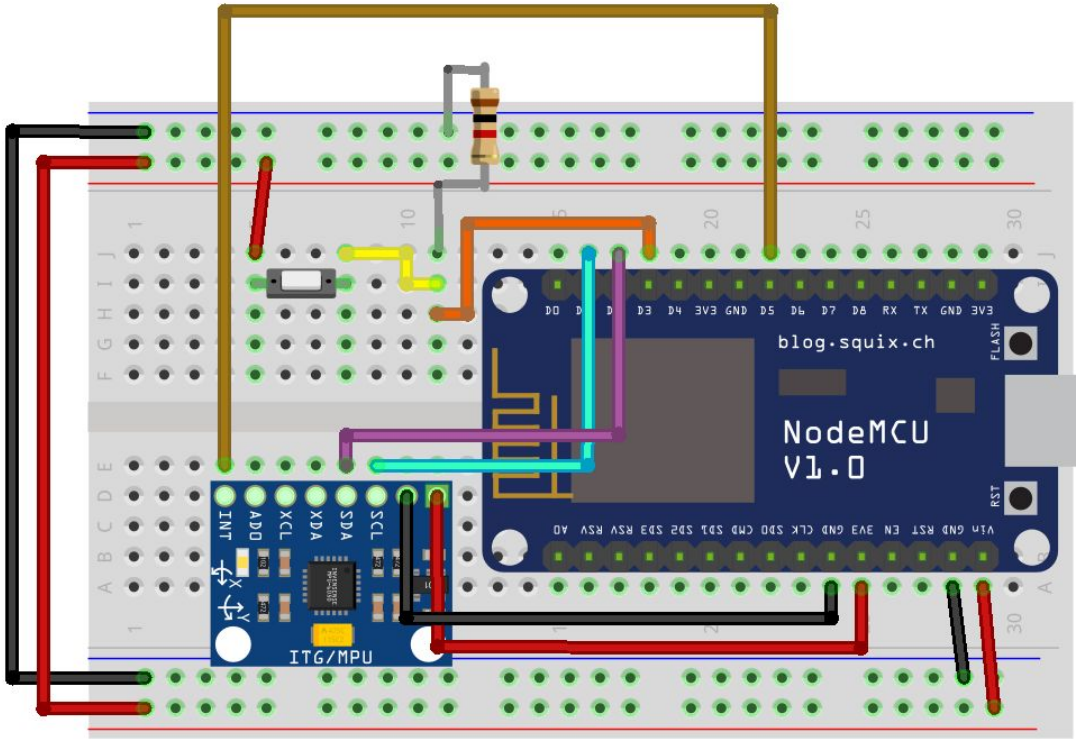
Question: How to process the raw data?

Code - MPU6050_DMP6

DMP is Digital Motion Processor.
It is embedded in MPU6050 and can help to *process the raw data to readable data*.
The calculated data will be push into a queue and *an interrupt signal will be sent*. **(Need to assign a pin for reading the interrupt signal from INT)**

Reference [1]: https://hackmd.io/@csielee/HkSQOMX1b?type=view

Hardware overview



Step (3/3)

System receive all the data from the motion controller as user's input.

Build a local network with your hotspot

[Example]
For a local IP: *192.168.43.107*

Gateway is *192.168.43.1*
Subnet mask is *255.255.255.0*

Wifi

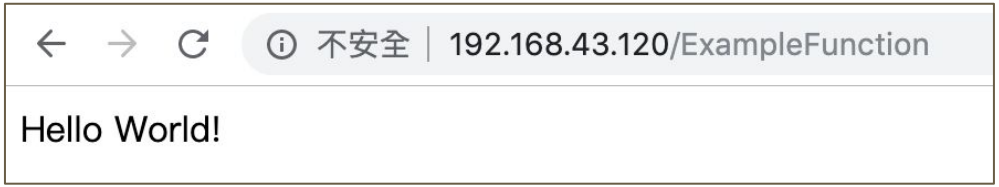
Code - wifi_template

wifi_template

```
#include "ESP8266WiFi.h"
#include "WiFiClient.h"
#include "ESP8266WebServer.h"
```

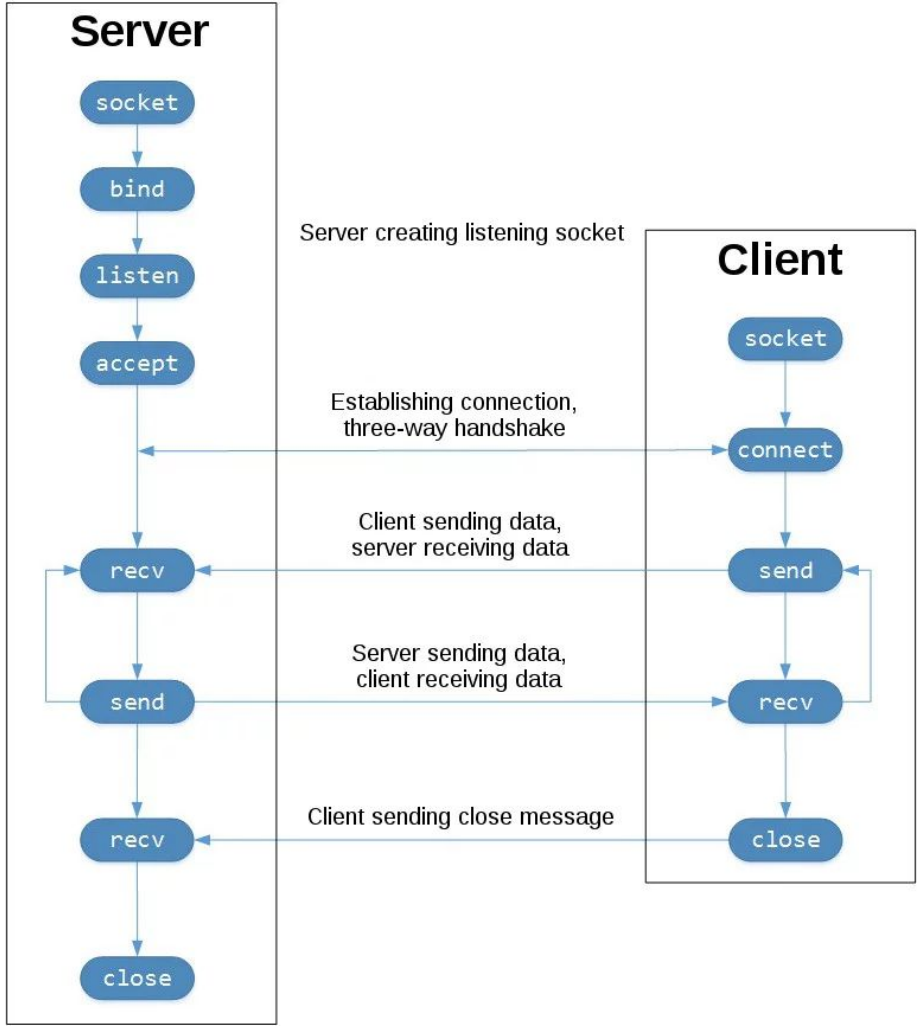
Replace with your own settings.

```
//=================================================
// WiFi config
//=================================================
/***** YOUR CODE HERE (START) *****/
const char* ssid = "SSID of your hotspot";
const char* password = "password";
IPAddress staticIP(10, 0, 1, 120); // For different devices, change this static ip address
IPAddress gateway(10, 0, 1, 1);
IPAddress subnet(255, 255, 255, 0);
/****** YOUR CODE HERE (END) ******/

//=================================================
// Server
//=================================================
ESP8266WebServer server(80);

//=================================================
// Functions
//=================================================
// For checking whether this WiFi board has been assigned to a static ip address
```



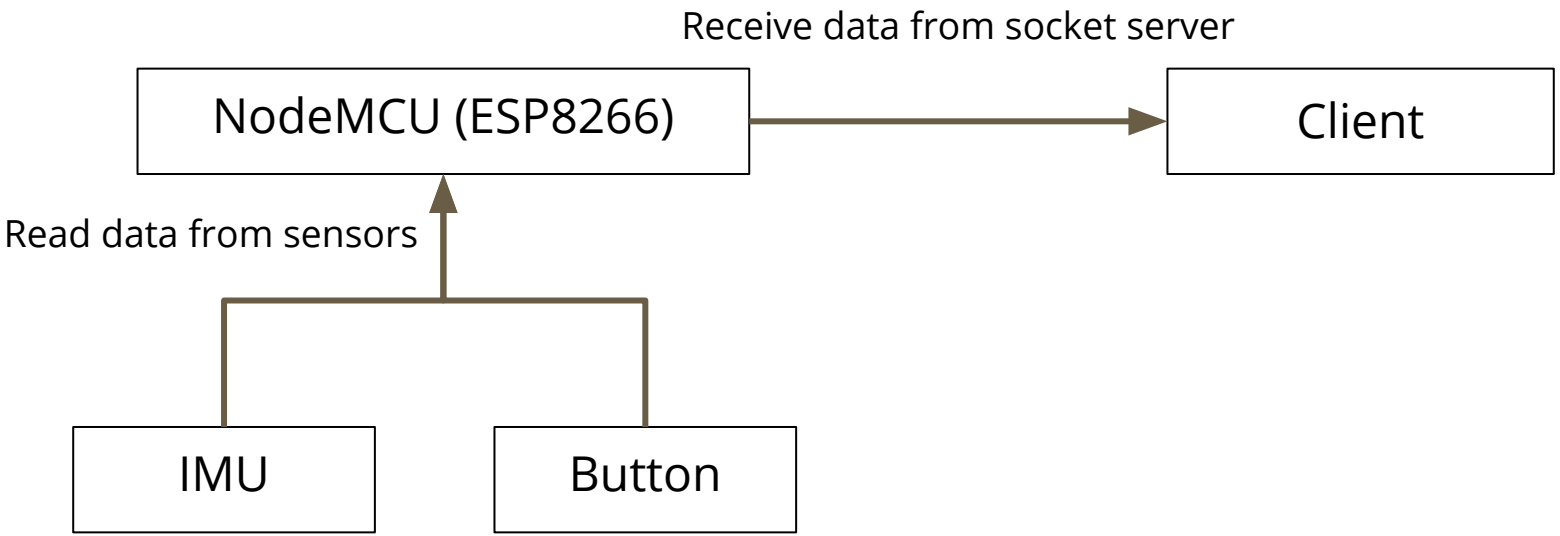
Use another device connected to the same LAN, then you can access the web server on NodeMCU with its IP

Socket


Server
socket
bind
listen
accept
recv
send
recv
close
Server creating listening socket
Establishing connection,
three-way handshake
Client sending data,
server receiving data
Server sending data,
client receiving data
Client sending close message
Client
socket
connect
send
recv
close

System Architecture

Server **Client**



Time to implement!

Code - socket_server_example, socket_client_example, socket_client

Test your code by running “socket_client.py”
You should receive *the state of button* and *IMU data* then print them out.

```
Received b'0 88 -1028 15712 -462 -549 78'
Received b'0 132 -1040 15784 -453 -525 75'
Received b'0 36 -928 15924 -412 -492 92'
Received b'1 -44 -1060 15880 -463 -605 63'
Received b'1 124 -1040 15920 -456 -593 67'
Received b'1 52 -964 15624 -443 -541 68'
Received b'1 64 -1016 15776 -445 -531 60'
Received b'1 64 -980 15792 -440 -456 54'
Received b'1 132 -1072 15684 -450 -463 56'
Received b'1 88 -992 15700 -465 -433 78'
Received b'1 156 -1032 15720 -409 -426 72'
Received b'1 120 -960 15740 -492 -373 74'
Received b'1 108 -1092 15548 -549 -256 72'
Received b'1 184 -1108 15676 -534 -237 76'
Received b'0 -16 -996 15776 -468 -313 69'
Received b'0 -604 -900 16732 -468 -123 77'
Received b'0 172 -1020 15624 -456 -350 69'
Received b'0 204 -956 15712 -470 -298 69'
Received b'0 132 -1008 15872 -475 -323 54'
Received b'0 60 -1012 15724 -488 -426 89'
Received b'0 124 -940 15700 -464 -468 84'
```

Next lesson...

With the data from sensors, what can we do?

Let’s make a shooting game!