

# IERG4230 / IEMS5721

## Internet of Things

## Mini-project

# (Sensor connection and testing)

Prepared by:  
Bun  
Ver 2021.11.12

# Table of Contents:

- Install Arduino
- Install ESP8266 to Arduino and basic test
- Install OLED Adafruit SSD 1306 to Arduino and basic test
- ESP8266 with I/O extended board
- Sample codes

# Install Arduino

# Install Arduino

- Go to Arduino official web page <https://www.arduino.cc/en/Main/Software> download the latest version.
- All sample codes are tested under version 1.8.13 (<https://www.arduino.cc/en/Main/OldSoftwareReleases#previous>)
- Run the downloaded **installer** or unzip the downloaded **zip** file.



## Arduino 1.8.x, 1.6.x, 1.5.x BETA

These packages are no longer supported by the development team.

1.8.14	Windows <a href="#">Windows Installer</a>	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM 32 Linux ARM 64	<a href="#">Source code on Github</a>
1.8.13	Windows <a href="#">Windows Installer</a>	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM 32 Linux ARM 64	<a href="#">Source code on Github</a>
1.8.12	Windows <a href="#">Windows Installer</a>	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM 32 Linux ARM 64	<a href="#">Source code on Github</a>

Windows

**Windows Installer**

ZIP version for non-admin.

Installer version  
needs admin right.

# Install ESP8266 to Arduino and basic test

The mark on ESP8266 is ESP8266MOD

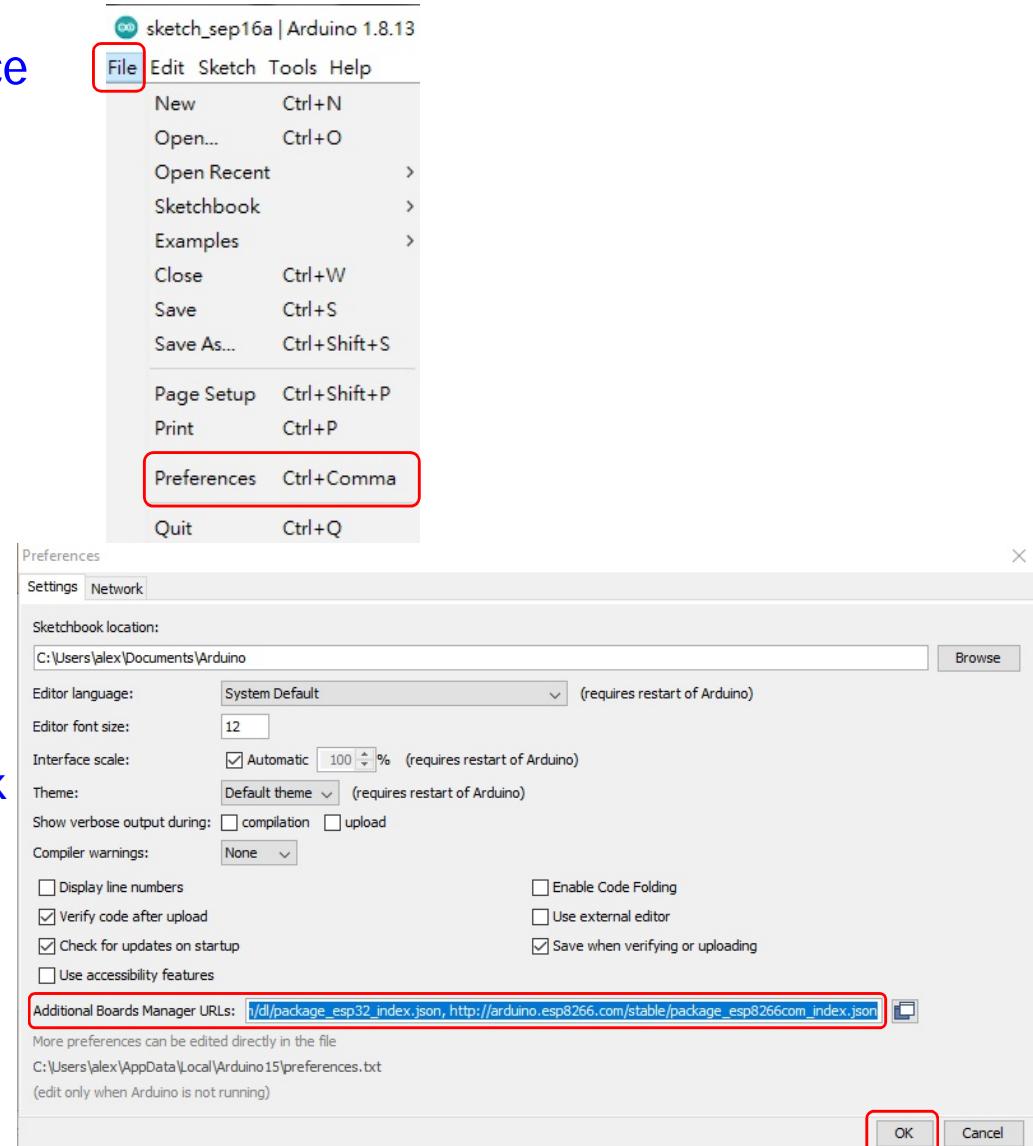


<https://www.aliexpress.com/item/32799552941.html>

# Install ESP8266 to Arduino and basic test

Reference video: <https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/>

1. In Arduino IDE, go to **File > Preference**

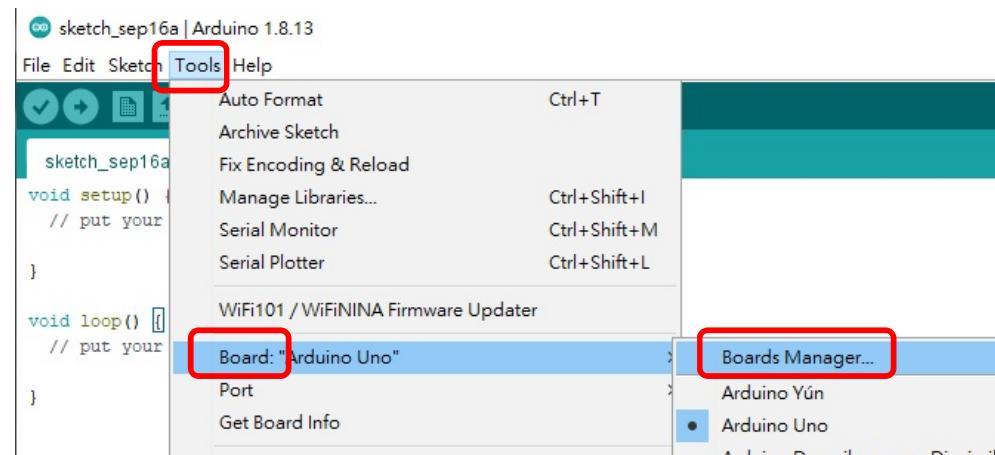


2. Fill Additional Boards Manager URLs:

- [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json),
- [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)
- Press OK.

# Install ESP8266 to Arduino and basic test

3. Go to **Tools** > **Board**: > **Board Manger...**

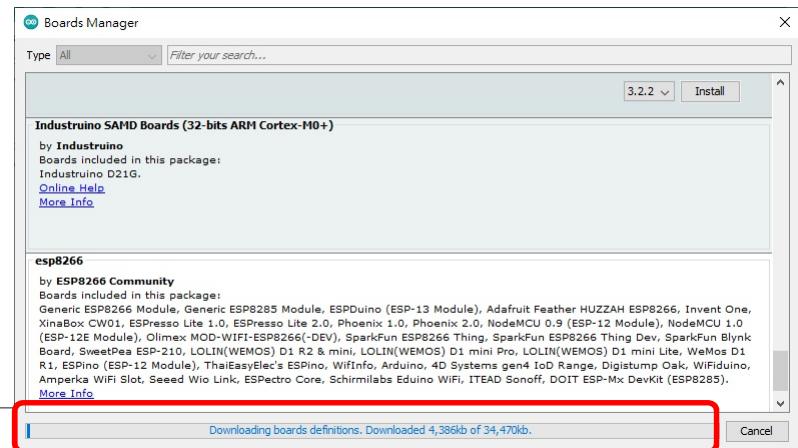


4. Key in **ESP8266** to the **Type** field. The library for ESP8266 will be found from Internet due to step-2 setting.
5. Select version **3.0.1** and press **Install**.

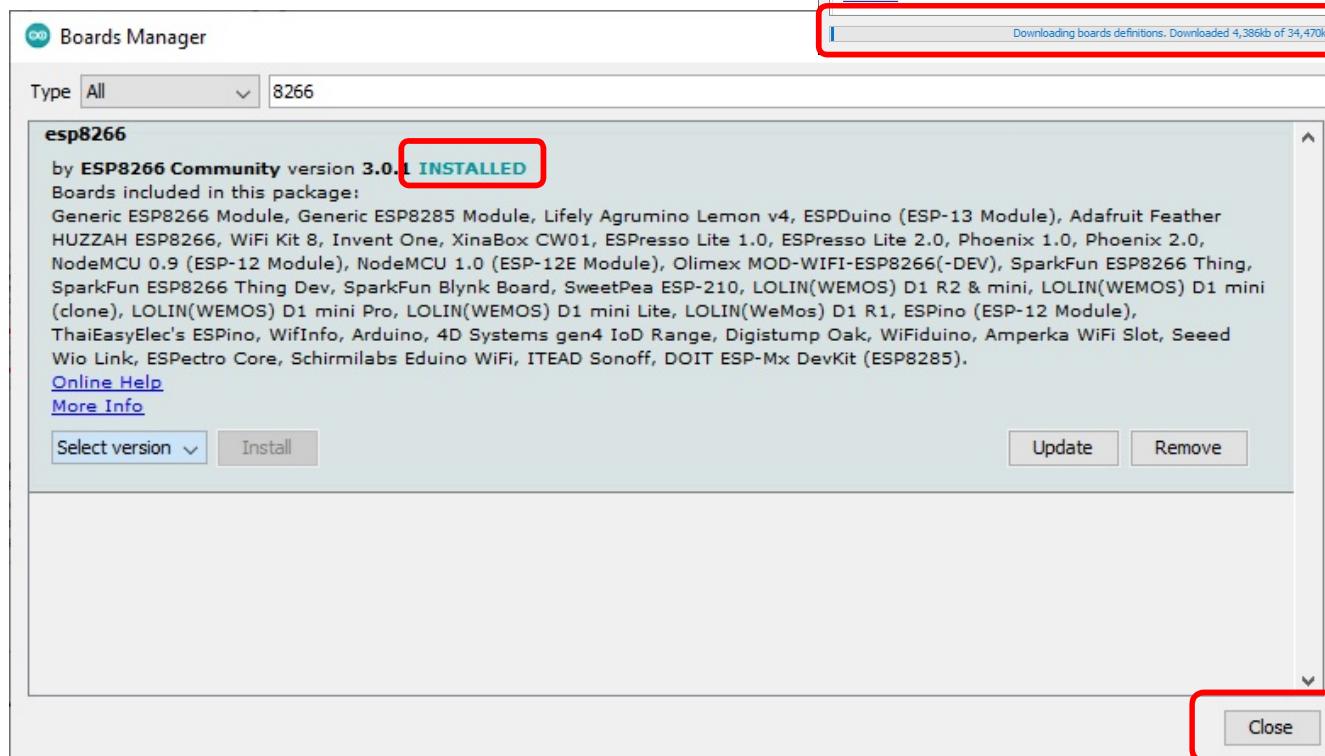


# Install ESP8266 to Arduino and basic test

7. Wait until installation complete.



8. After installation, **INSTALLED** will be shown, and close the dialogue.

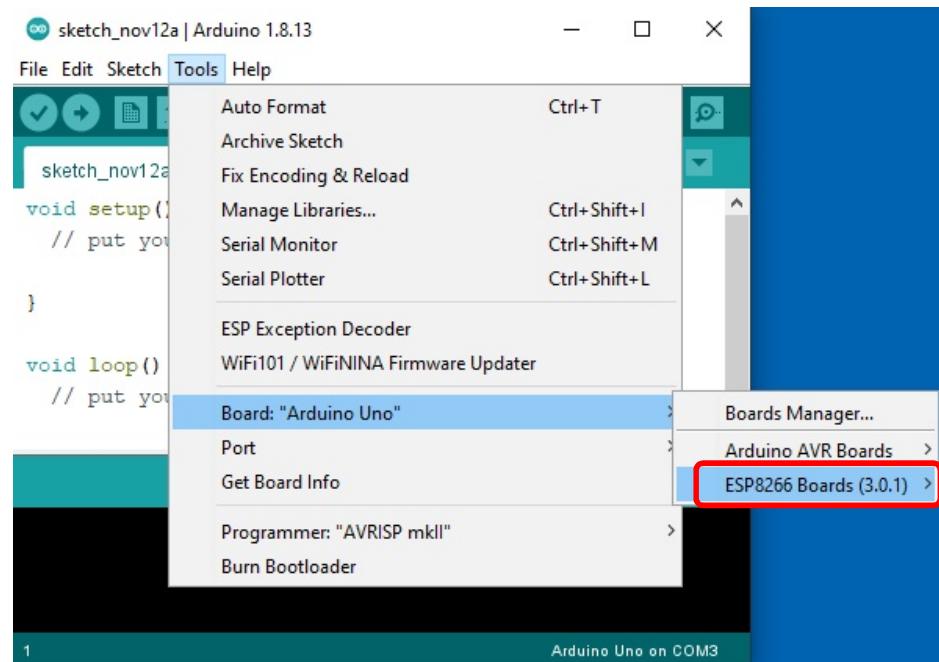


# Install ESP8266 to Arduino and basic test

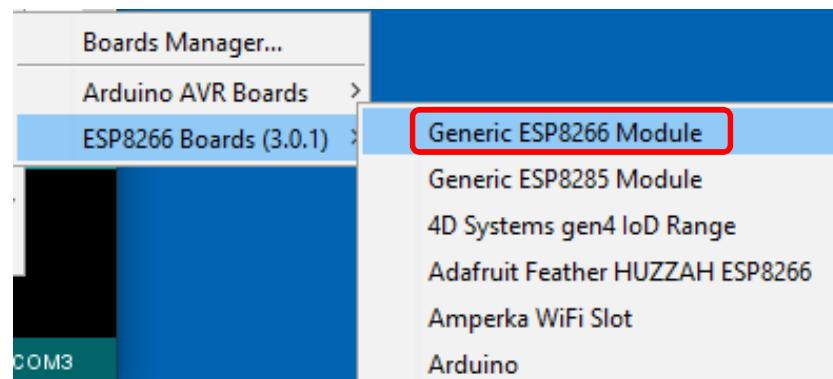
7. Change the target board to ESP8266.

Go to **Tools > Board > ESP8266**.

There are more than one model of ESP8266 in the selection list.



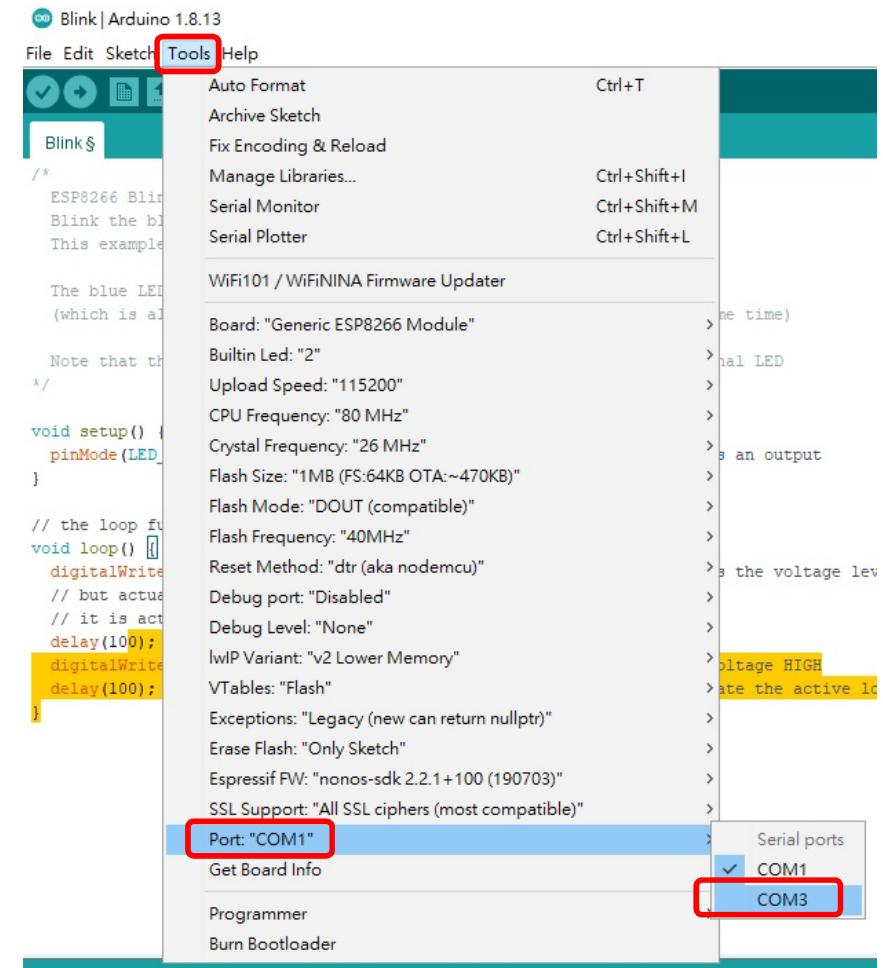
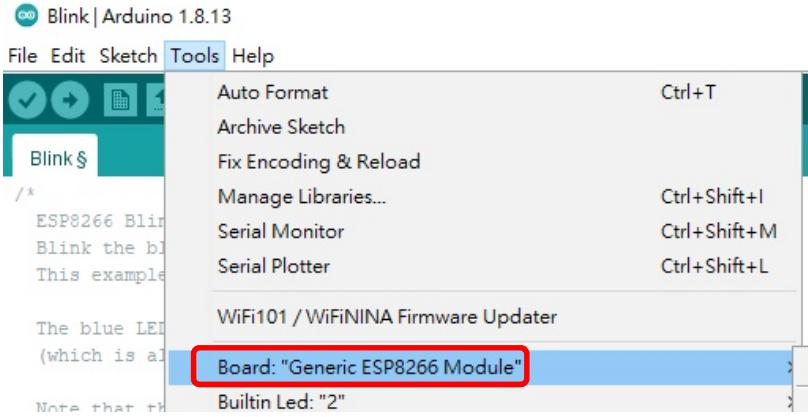
8. Select **Generic ESP8266 Module**



# Install ESP8266 to Arduino and basic test

9. After change board, **check** it again.

Go to **Tools > Board > ESP8266**. Make sure your target board is ESP8266.



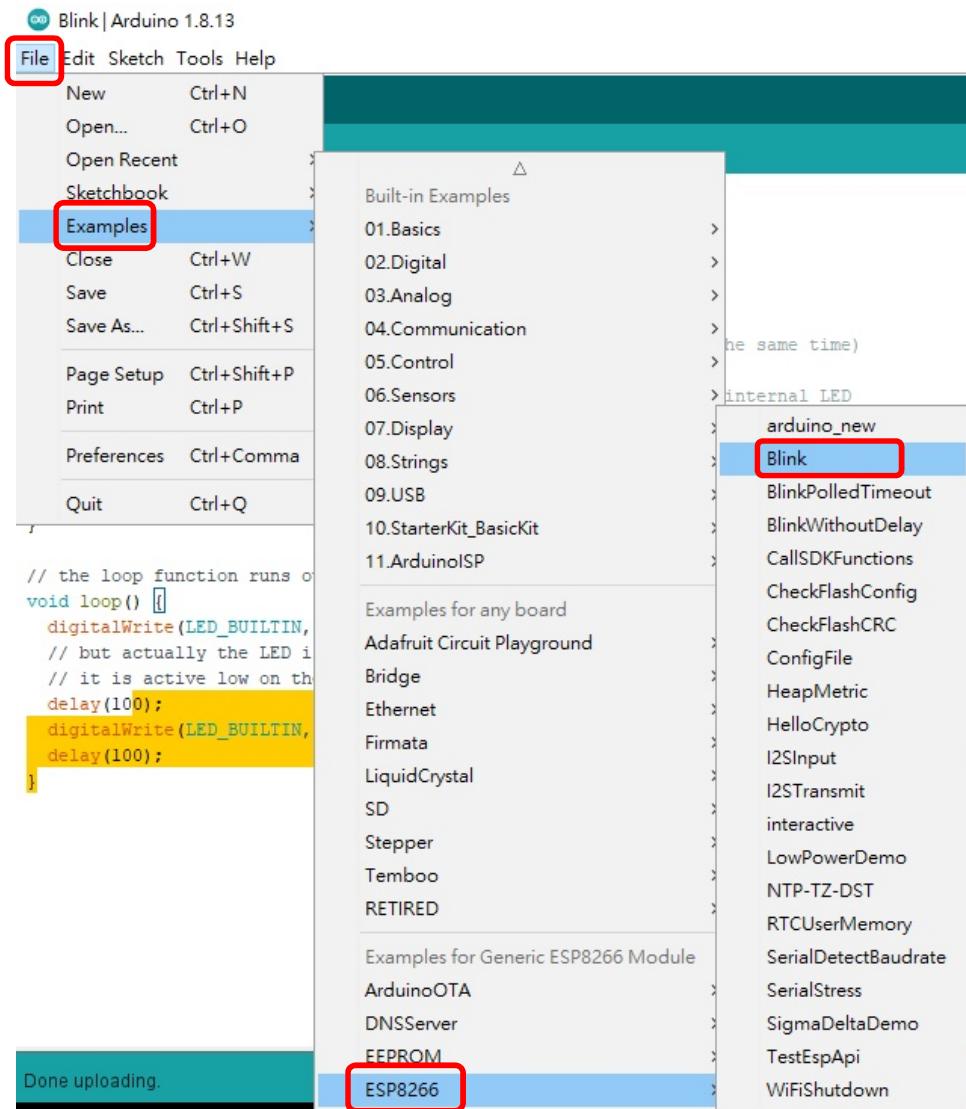
10. Connect the board to your PC with a USB cable (Micro-USB).

11. A new serial port (COM3 normally) will be found. It shows that your PC connects to the chip (USB to serial) on board correctly.

12. Go to **Tools > Port**, select the COM port according to your setting.

# Install ESP8266 to Arduino and basic test

13. Open an example for test. Go to **File** > **Example** > **ESP8266** > **Blink**.



# Install ESP8266 to Arduino and basic test

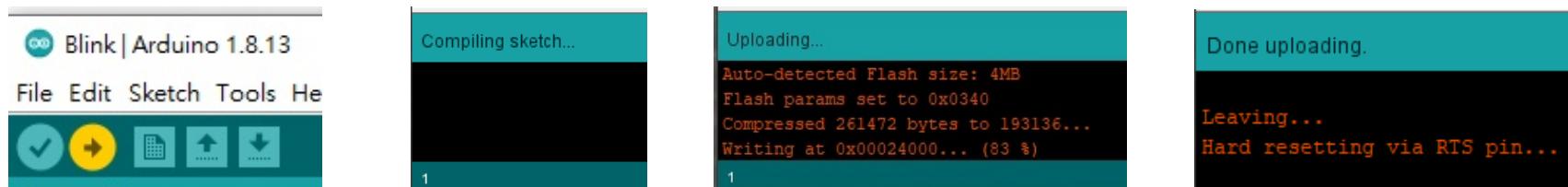
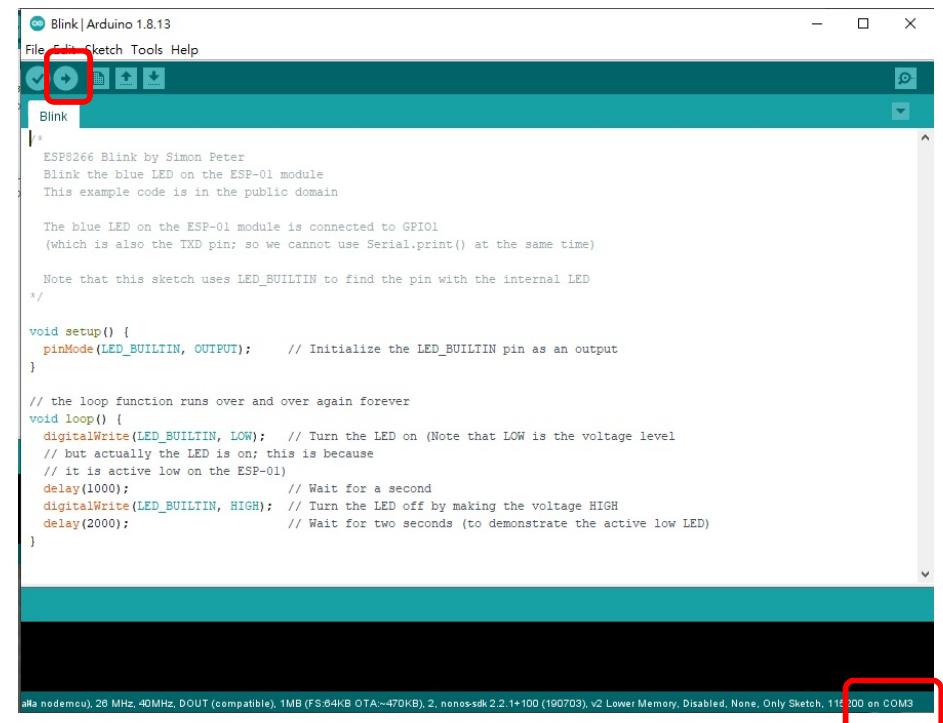
14. You can see the example program. Check the correct COM is selected (at the bottom right corner).

15. Press the arrow icon for compiling and download.

The procedures are

- [compiling sketch...](#)
- [Uploading](#)
- [Done uploading.](#)

The blue LED near the antenna will flash slowly. It means that **Arduino IDE** and **ESP8266 library** is installed correctly.



# Install ESP8266 to Arduino and basic test

16. Modify the program, change two `delay()`s from 1000 and 2000 to 100 and 200. Press the arrow icon to recompile and download again. The blue LED must flash faster after Done downloading.

```
// the loop function runs over and over again
void loop() {
    digitalWrite(LED_BUILTIN, LOW);      // Turn the LED off (it is active low)
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
    delay(1000);                      // Wait for
    digitalWrite(LED_BUILTIN, HIGH);     // Turn the LED on
    delay(2000);                      // Wait for
}
```

original

```
// the loop function runs over and over again if
void loop() {
    digitalWrite(LED_BUILTIN, LOW);      // Turn the LED off (it is active low)
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
    delay(100);                       // Wait for
    digitalWrite(LED_BUILTIN, HIGH);     // Turn the LED on
    delay(200);                       // Wait for
}
```

modified

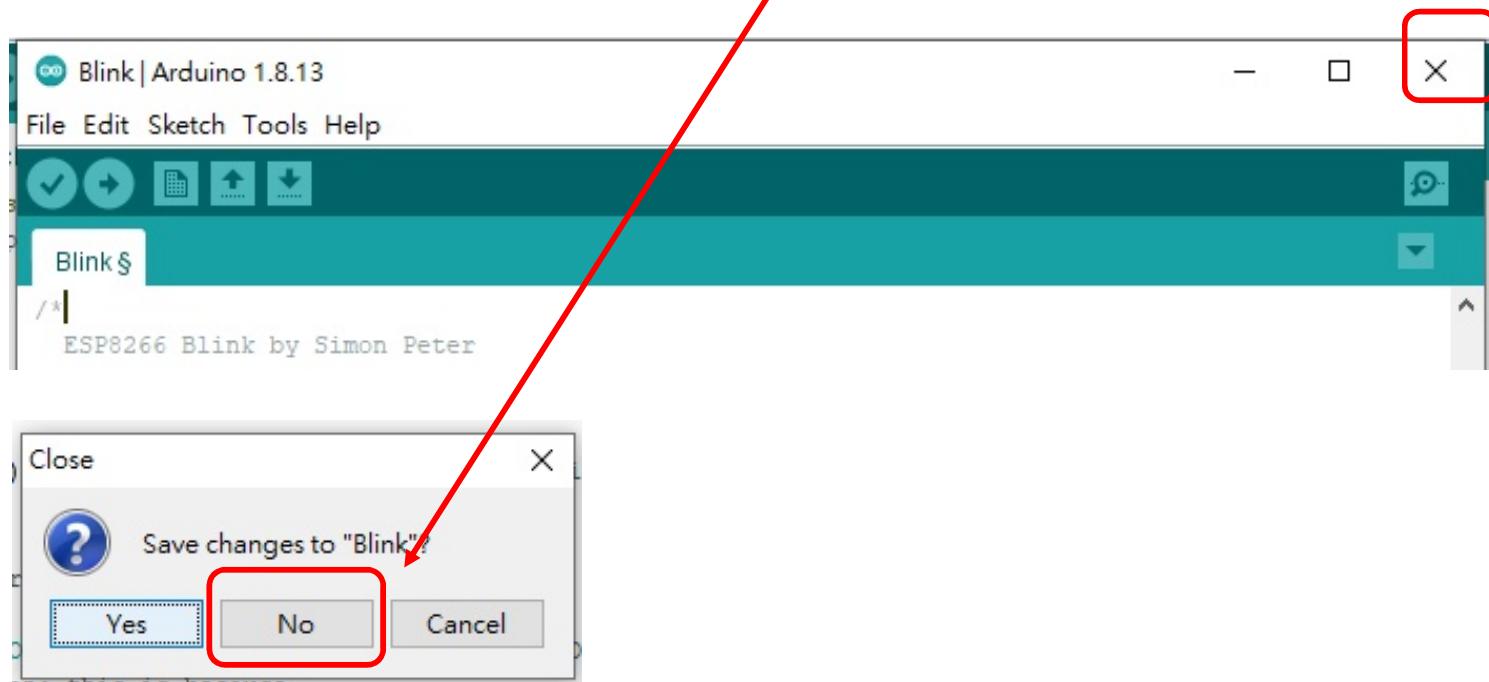
The blue LED



17. Modify the program, change two `delay()`s from 1000 and 2000 to 100 and 200. Press the arrow icon to recompile and download again. The blue LED must flash faster after Done downloading.

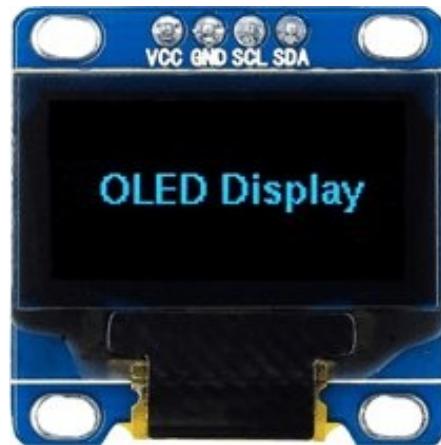
# Install ESP8266 to Arduino and basic test

18. Press the 'X' to close Arduino directly. And select 'No' for save changes to prevent modify the original example.



**Important:** Please keep all examples as the original, you can recall them again.

# Install OLED Adafruit SSD 1306 to Arduino and basic test



<https://www.geekering.com/wp-content/uploads/2020/07/SSD1306-OLED-Display.png>

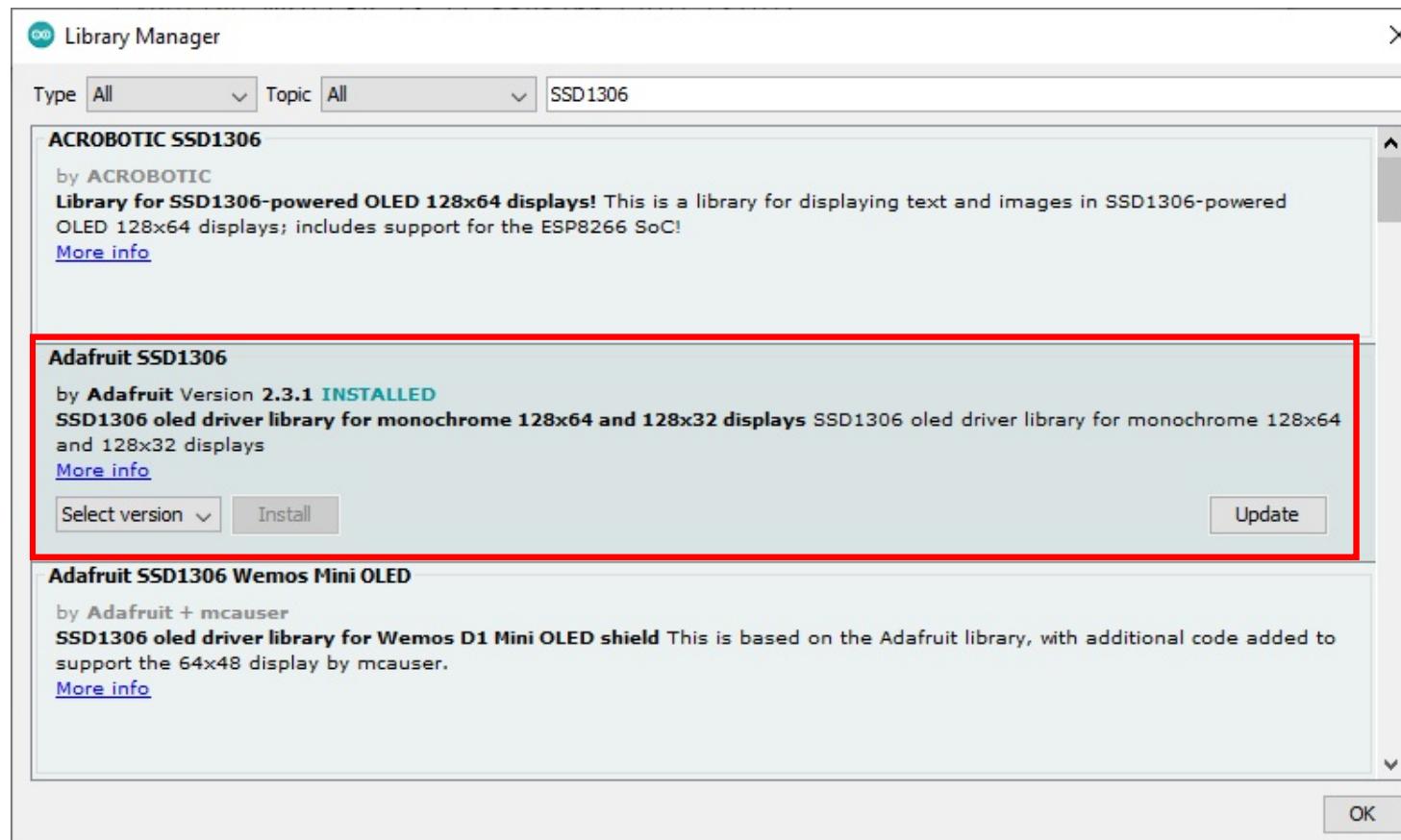
# Installing OLED Adafruit SSD 1306 Library to Arduino and basic test

Reference link: <https://randomnerdtutorials.com/esp8266-0-96-inch-oled-display-with-arduino-ide/>

There are several libraries available to control the OLED display with the ESP8266.

We'll use two Adafruit libraries: [Adafruit\\_SSD1306 library](#) and [Adafruit\\_GFX library](#).

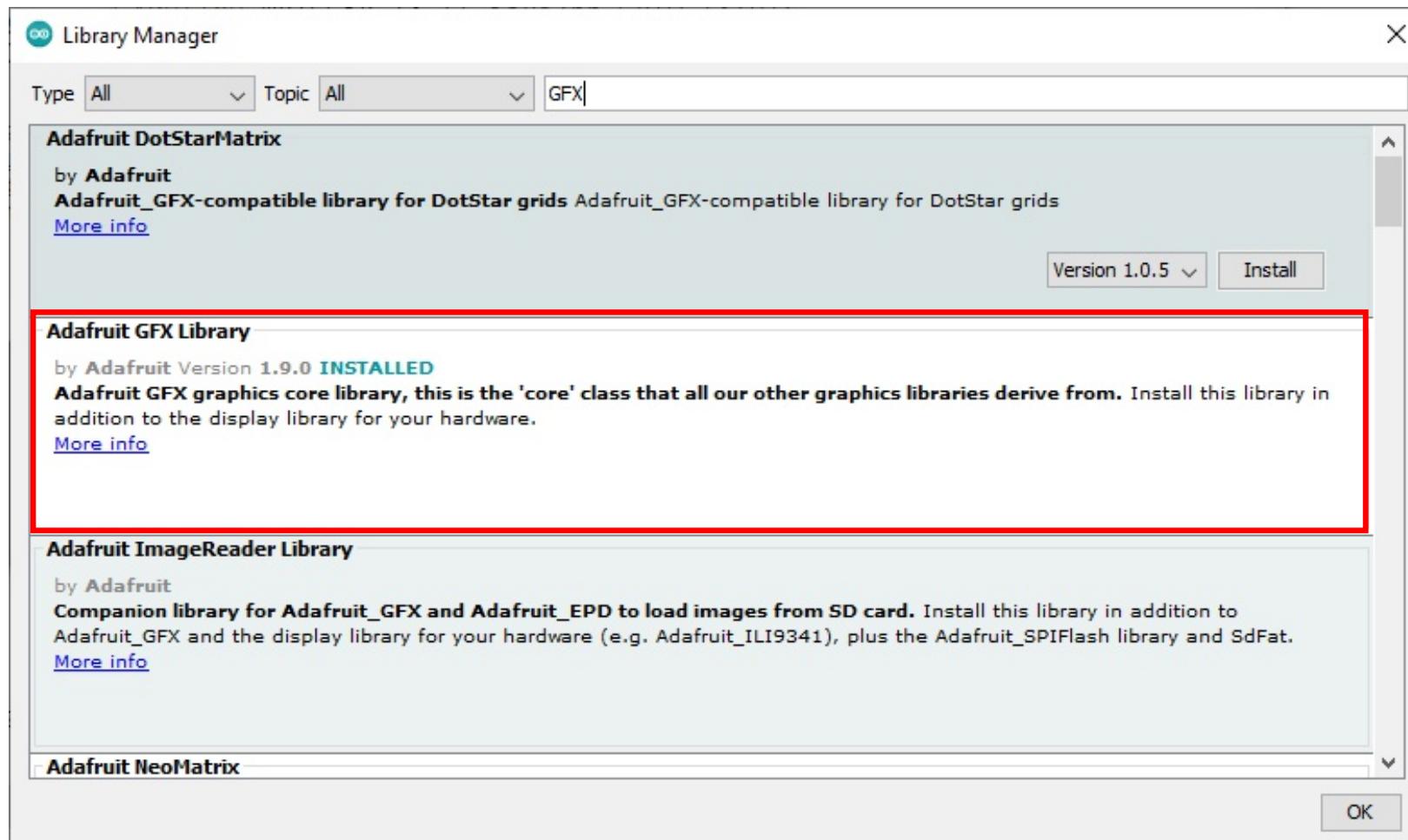
1. Open your Arduino IDE and go to Sketch > Include Library > Manage Libraries. The Library Manager should open.
2. Type "SSD1306" in the search box and install the SSD1306 library from Adafruit.



# Installing OLED Adafruit SSD 1306 Library to Arduino and basic test

Reference link: <https://randomnerdtutorials.com/esp8266-0-96-inch-oled-display-with-arduino-ide/>

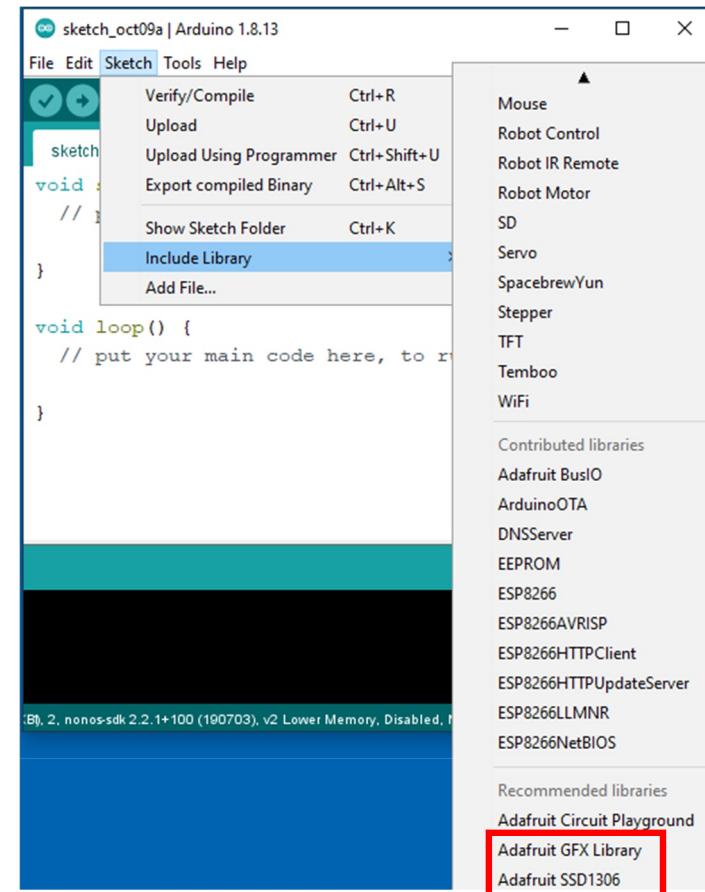
3. After installing the SSD1306 library from Adafruit, type “GFX” in the search box and install the library.
4. After installing the libraries, restart your Arduino IDE.



# Installing OLED Adafruit SSD 1306 Library to Arduino and basic test

Reference link: <https://randomnerdtutorials.com/esp8266-0-96-inch-oled-display-with-arduino-ide/>

5. In your Arduino IDE, go to **Sketch > Include Library** you should see two libraries [Adafruit SSD1306](#) and [Adafruit\\_GFX library](#)



# Installing OLED Adafruit SSD 1306 Library to Arduino and basic test

Reference link: <https://randomnerdtutorials.com/esp8266-0-96-inch-oled-display-with-arduino-ide/>

6. After wiring (I2C) the OLED display to the ESP8266 you can use one example from the library to see if everything is working properly.
7. In your Arduino IDE, go to **File > Examples > Adafruit SSD1306** and open the example **ssd1306\_128x64\_i2c**
8. Change **OLED\_RESET** variable to **-1** as shown below

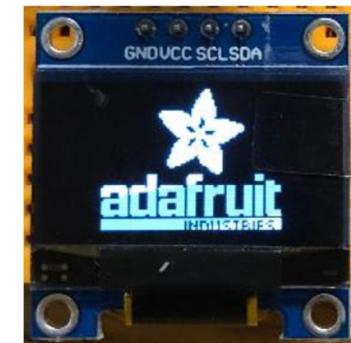
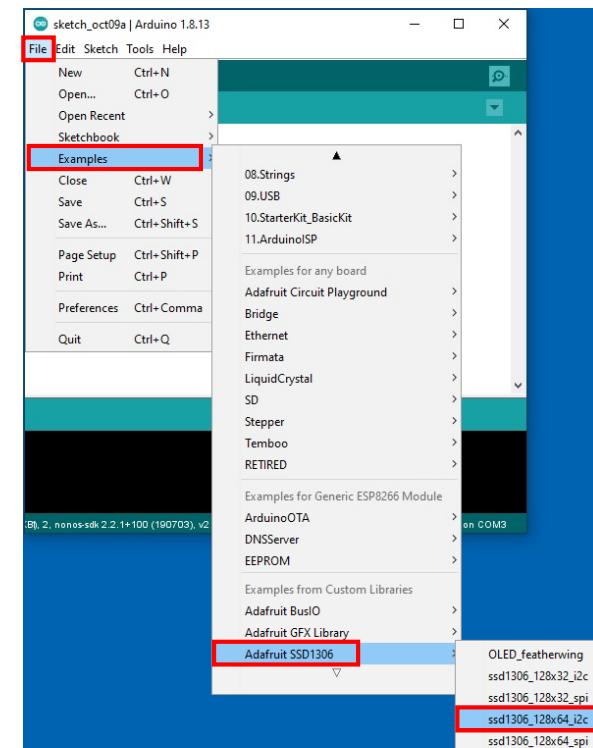
```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

9. Change OLED address to **0x3C** as shown below

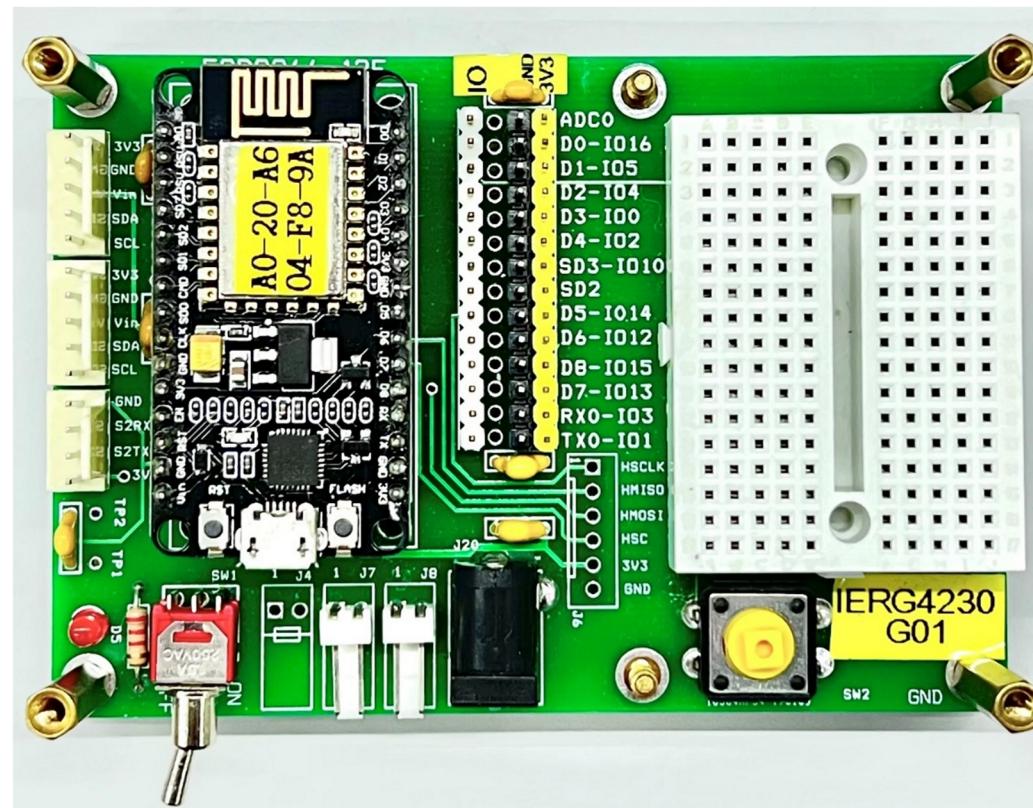
```
// SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
```

10. Upload the code to your ESP8266 board. Don't forget to select the right board and COM port in the Tools menu.



The result of the example  
**ssd1306\_128x64\_i2c**

# ESP8266 with I/O extended board



## Advantages of ESP8266:

- Support  $I^2C$  interface, suggest using  $I^2C$  modules.
- Support two UARTs.
- Support 802.11 b/g/n WLAN MAC protocol.
- Support sleep mode for power saving.
- Max. 17 digital I/Os (reduce number of I/O when special functions are used.).
- Support SPI interface.
- Support four PWM outputs.

## Important notes (**limitations**) of ESP8266:

- The **max.** power supply of ESP8266 is **3.3V**.
- Only one analogue input (10-bit A to D) **A0**.
- The output of on board regulator is 700mA only.
- For Digital I/Os, suggest using pin **D5, D6, D7 and D8**.
- **No 5V supply**. You **need** a voltage converter for interfacing to **5V** I/O modules.

## Matter needing attention:

### On Boot/Reset/wakeup,

- D8(GPIO15) **MUST** keep LOW **AND** D4(GPIO2) **MUST** keep HIGH
- D3(GPIO0 ) **MUST** keep HIGH (Run mode) running downloaded program
- D3(GPIO0 ) **MUST** keep LOW (Flash mode) for downloading program

### On Board LEDs (active LOW):

- D0(GPIO16)
- D4(GPIO2)

## Important notes of our design.

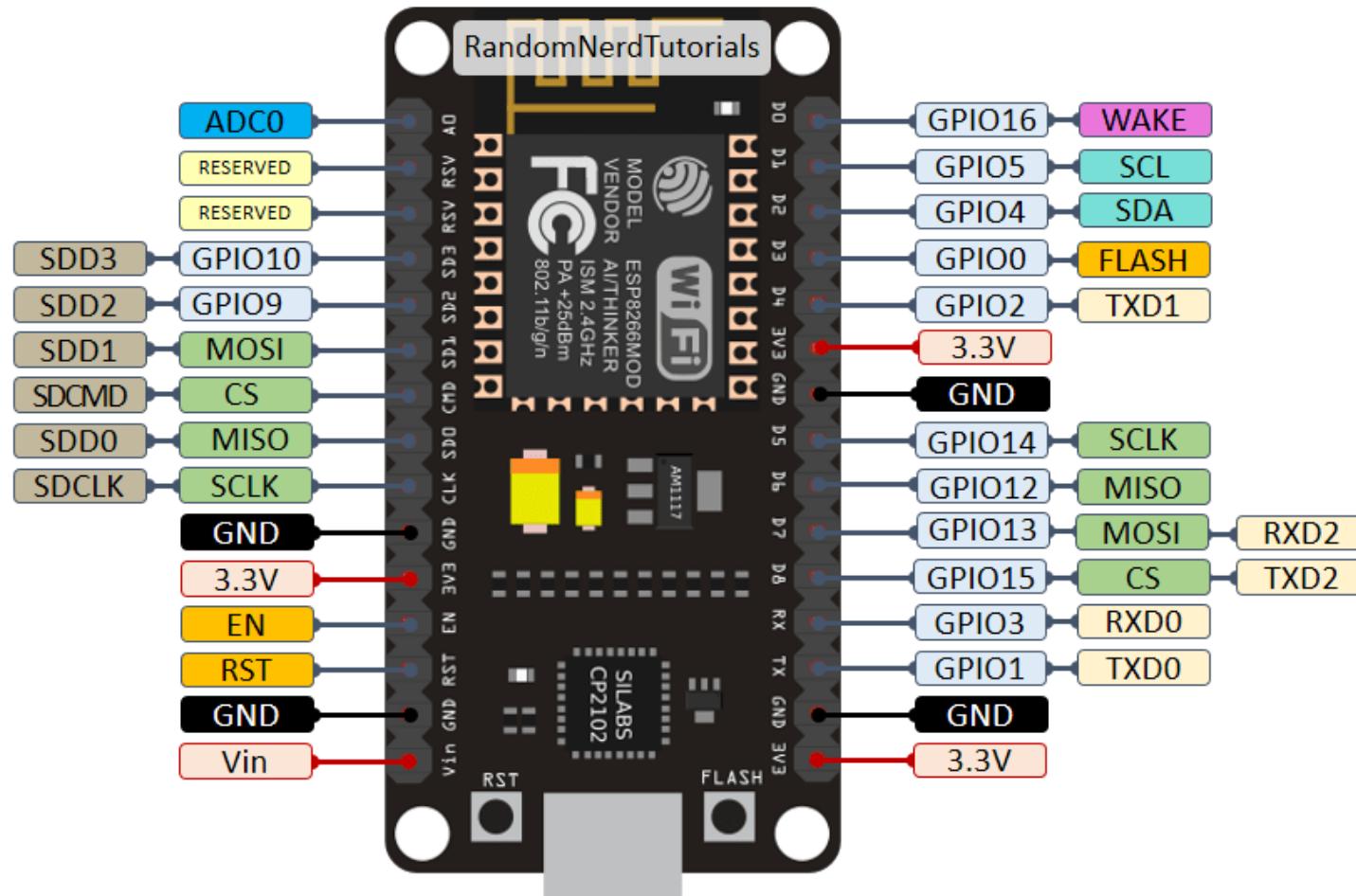
- The provided ESP8266 board powered by computer USB port (normal 500mA). Do not powered by external power source (e.g. Mobile phone charger).
- For safety reason, the ESP8266 boards are powered by USB battery pack. **Do not use AC-to-USB converter**. Nobody know your AC converter that fulfils safety regulation or not.
- The current limit of ESP8266 on board regulator (5V to 3V3) is 700mA. Do not take too much current from it. It will be damaged by over current. Consult our technicians if your project needs to drive high current (e.g. motor, servo).
- ESP8266 does not support 5V I/O default. Need a voltage converter (simple resistor potential divider) for interface 5V modules. The provided 5V modules are tested. Consult our technicians if other 5V modules are connected to the CPU boards.
- For multiplex/protocol device, use  $I^2C$ .
- Release SPI pins for general I/Os .
- Use UART monitor in Arduino for programme debugging.

# Best Pins to Use

<https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/>

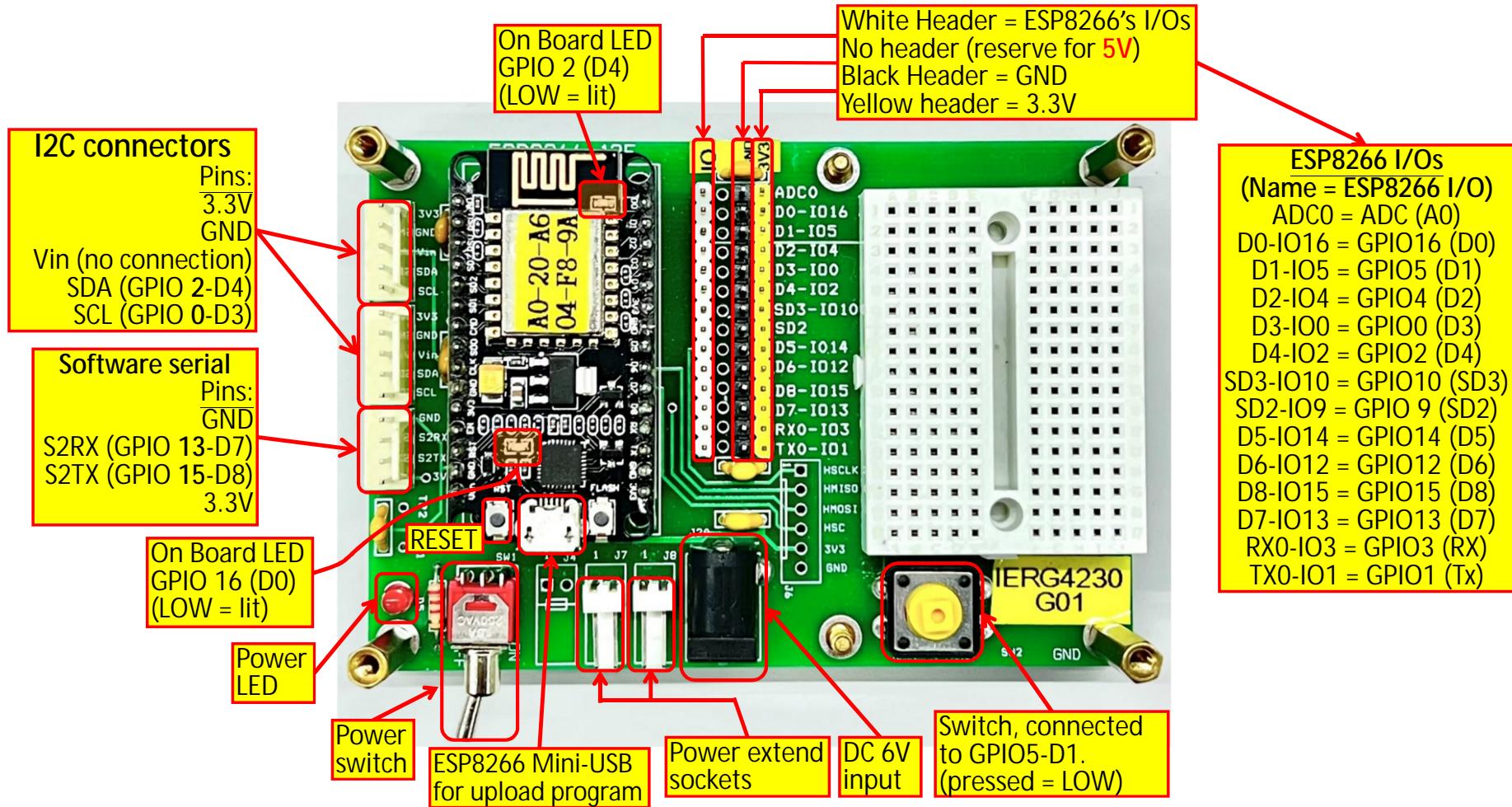
Label	GPIO	Input	Output	Notes
D0	GPIO16	no interrupt	no PWM or I2C support	<b>HIGH at boot</b> used to wake up from deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	<b>HIGH at boot</b> connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	GPIO3	OK	RX pin	<b>HIGH at boot</b>
TX	GPIO1	TX pin	OK	<b>HIGH at boot</b> debug output at boot, boot fails if pulled LOW
A0	ADC0	Analog Input	X	

## Pin assignment of ESP8266-12-E



<https://i2.wp.com/randomnerdtutorials.com/wp-content/uploads/2019/05/ESP8266-NodeMCU-kit-12-E-pinout-gpio-pin.png?ssl=1>

## Provided ESP8266 with I/O extended board



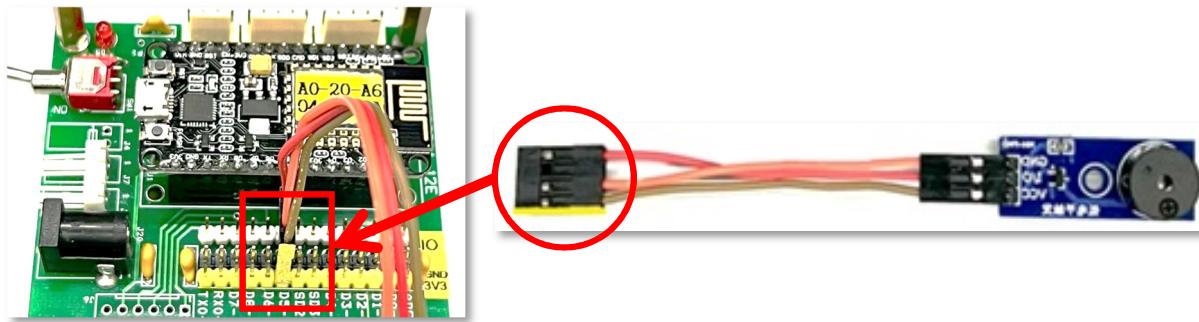
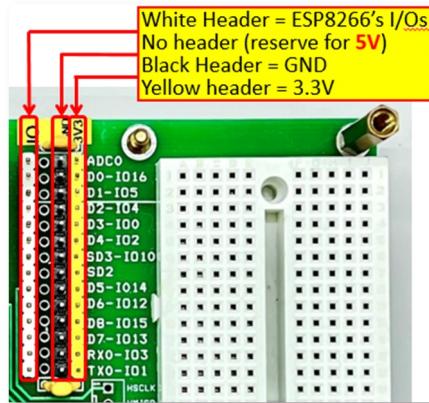
# Sample Codes

## Notes to the sample codes:

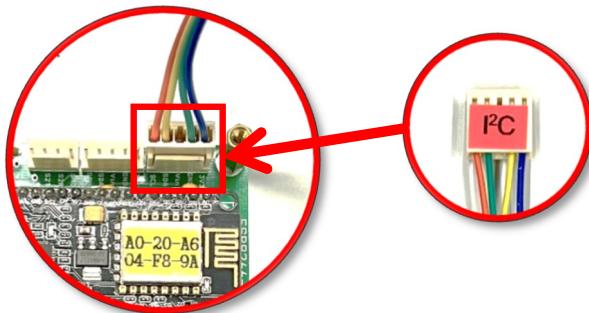
- Download “[All-in-one.rar](#)” or individual rar file from the Blackboard.
- Only group C devices are provided sample codes.
- All sample codes are tested with ESP8266 board.
- If you are using the given programs in Experiments 1 and 2 to develop your prototype, you need to migrate the sample codes to the above given programs.
- The I/Os used in the sample codes are just reference. You may change its according to your design with compliance to the project boards (Nano/ESP8266) specification.
- If you use the sensors with Nano, you need modify the connections (I/O) and sample code with Nano board specification.
- The sample codes are just for functionality test.
- Please read the comments in the sample code for more information.

## Provided ESP8266 with I/O extended board

- 3 rows of pins are **ESP8266's I/O (White)**, **GND (Black)** and **3.3V (Yellow)**.
- The socket of provided modules are multiple pins for these pins. Connect the colour marks (Yellow) to the Yellow header.

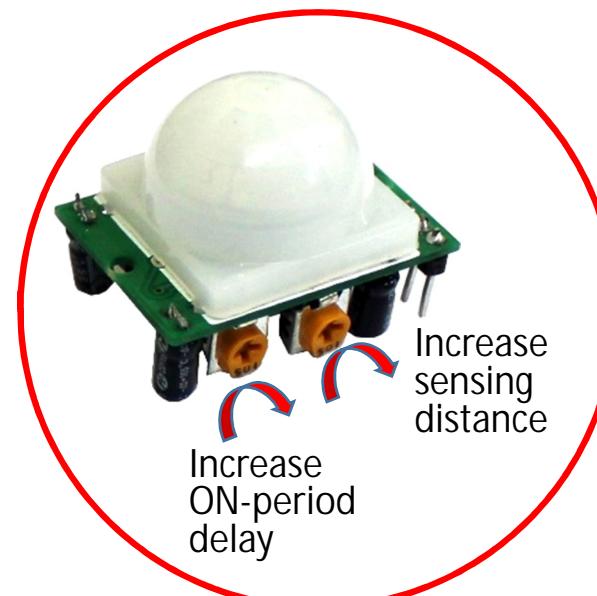
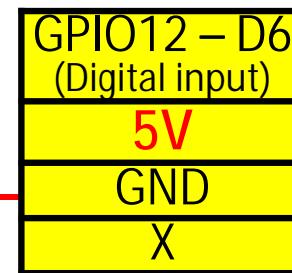
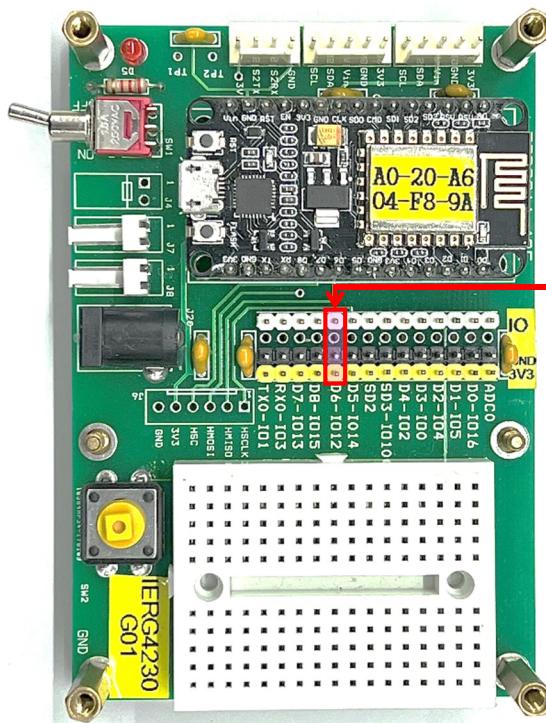


- The socket for  $I^2C$  modules is standardize, directly connect it to the  **$I^2C$  connectors**.



- A0 is 10-bit A/D input.
- The board **did not provide 5V** supply. Consult technician for **5V** sensors.

# C1. Human movement sensor (HC-SR501, Digital, 5V)



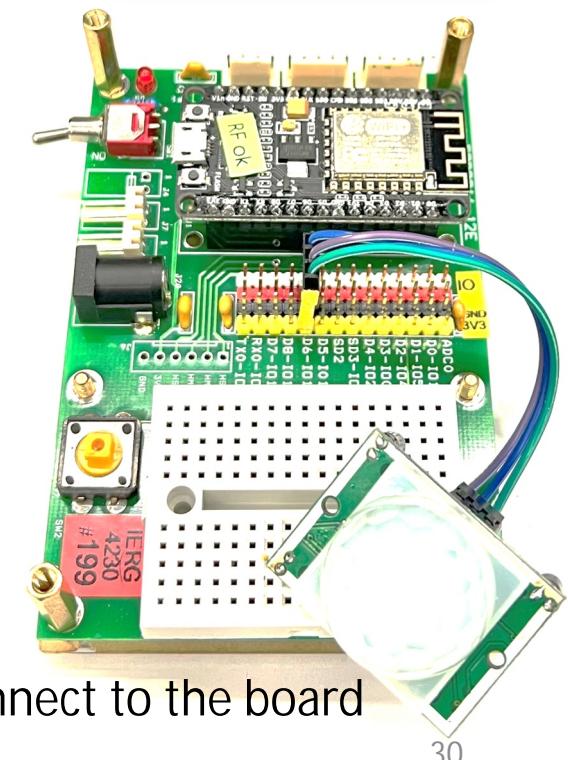
Provided device

This is 5V device

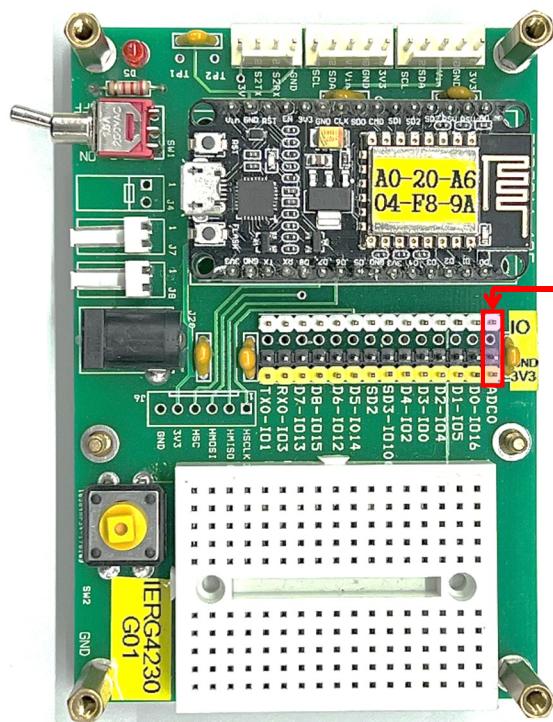
```
COM3
No movement detected
Movement detected
Buzzer play
Movement detected
Buzzer play
Movement detected
Buzzer play
No movement detected
No movement detected
No movement detected
No movement detected

 Autoscroll  Show timestamp
```

The result of the demo program

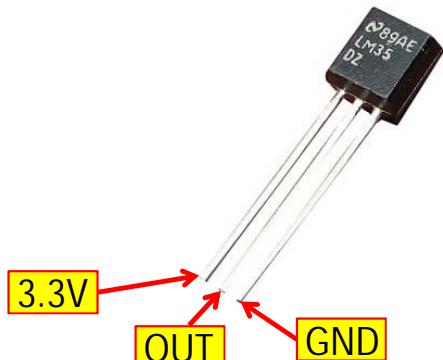


## C2. Temperature Sensor (LM35, analogue)

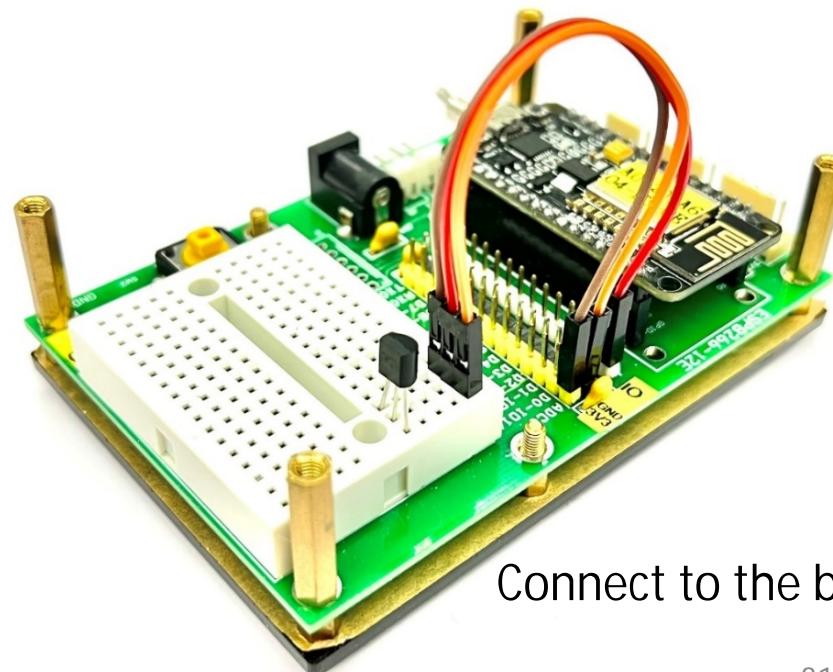


```
I 5 JAF ??  
ESP8266-12E/F LM35 testing program  
Build-in LED at GPIO-16(D0)  
Buzzer pin at GPIO-14(D5)  
LM35 analog value = 98 / Degree = 30.65C  
LM35 analog value = 105 / Degree = 32.84C  
LM35 analog value = 105 / Degree = 32.84C  
LM35 analog value = 104 / Degree = 32.53C  
LM35 analog value = 104 / Degree = 32.53C  
LM35 analog value = 103 / Degree = 32.22C  
LM35 analog value = 102 / Degree = 31.91C  
LM35 analog value = 102 / Degree = 31.91C  
LM35 analog value = 101 / Degree = 31.59C  
LM35 analog value = 101 / Degree = 31.59C
```

The result of the demo program



Provided device

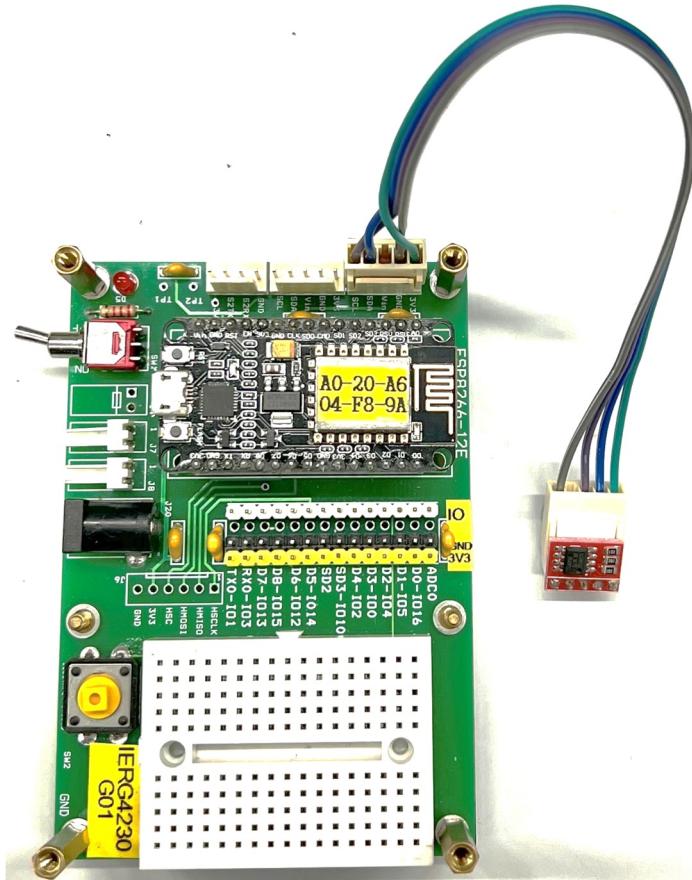


Connect to the board

### C3. Temperature Sensor (LM75 , $I^2C$ )



Provided device



Connect to the board

```
COM3
Tos set at 31.00 C
Thyst set at 30.00 C

Current temp: 32.00 C
Tos set at 31.00 C
Thyst set at 30.00 C

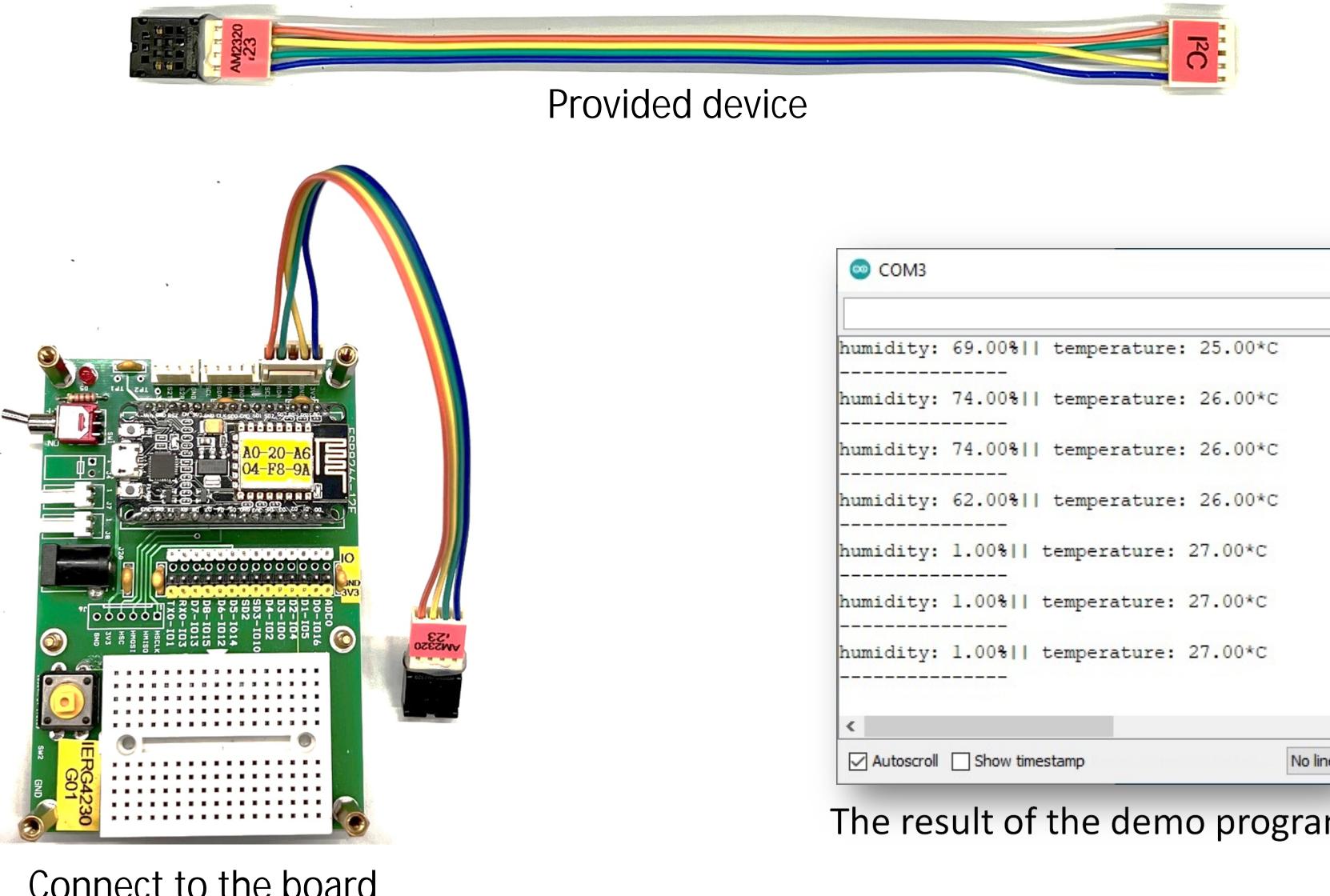
Current temp: 31.50 C
Tos set at 31.00 C
Thyst set at 30.00 C

Current temp: 31.50 C
Tos set at 31.00 C
Thyst set at 30.00 C

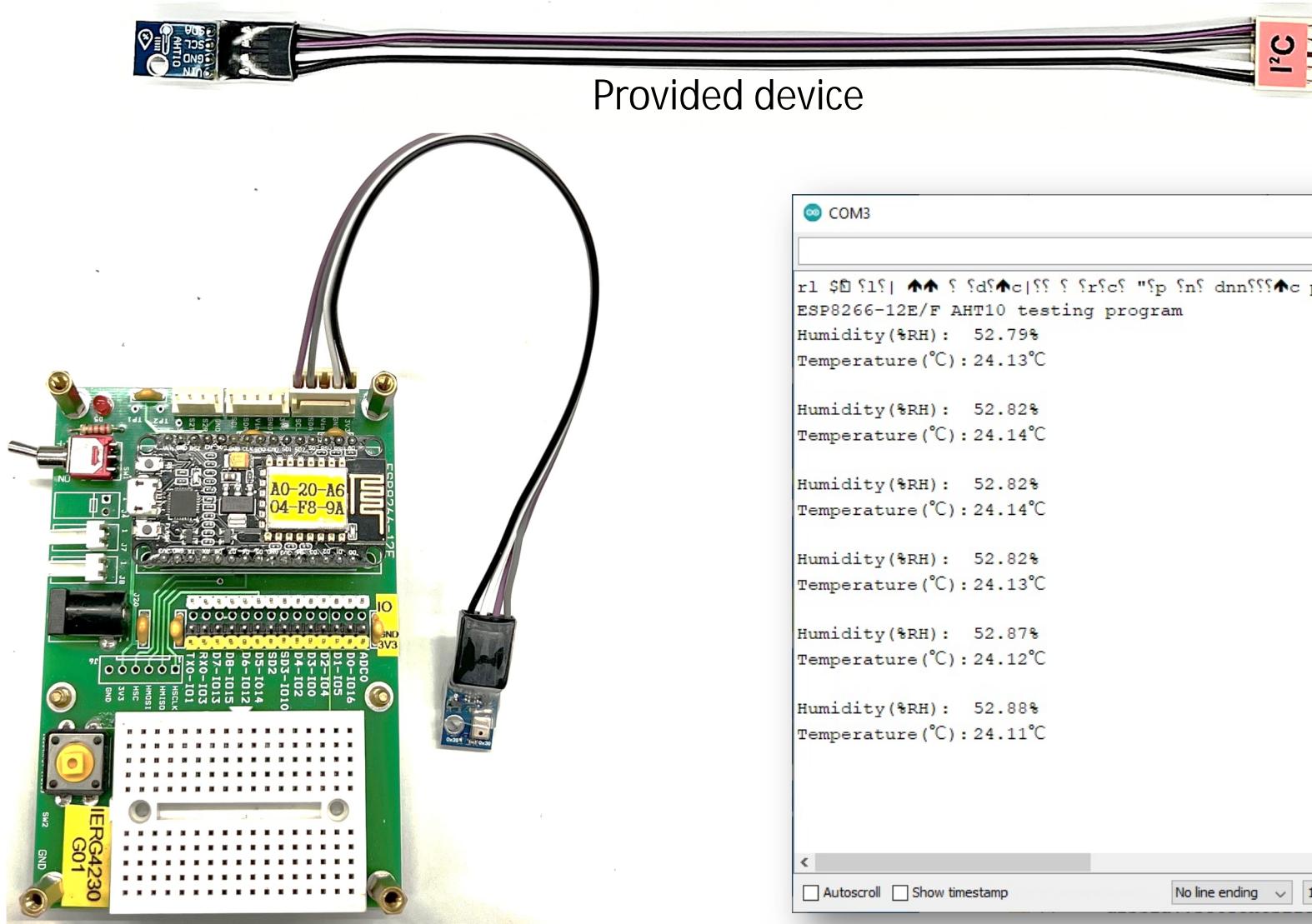
<   □
 Autoscroll  Show timestamp
```

The result of the demo program

## C4. Temperature and Humidity Sensor (AM2320, $I^2C$ )



## C5. Temperature and Humidity Sensor (AHT10, $I^2C$ )



```
COM3
rl $0 ?1?| ↑↑ ? ?d?▲c|?? ? frfc? "fp fn? dnn??▲c p?cd s$p?o? ↑ ?▲d
ESP8266-12E/F AHT10 testing program
Humidity(%RH): 52.79%
Temperature(°C): 24.13°C

Humidity(%RH): 52.82%
Temperature(°C): 24.14°C

Humidity(%RH): 52.82%
Temperature(°C): 24.14°C

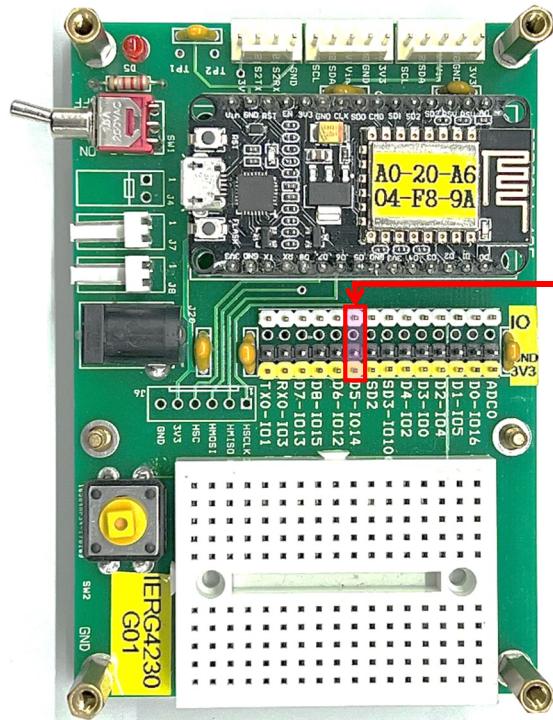
Humidity(%RH): 52.82%
Temperature(°C): 24.13°C

Humidity(%RH): 52.87%
Temperature(°C): 24.12°C

Humidity(%RH): 52.88%
Temperature(°C): 24.11°C
```

The result of the demo program

## C6. Buzzer (Digital)



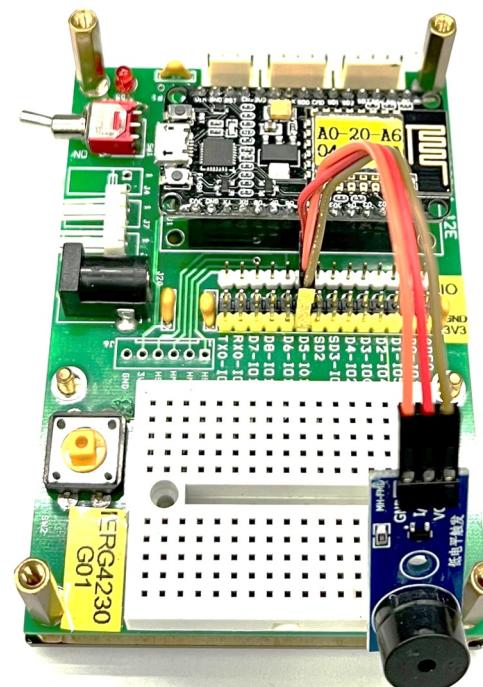
GPIO14 – D5
(Digital output)
X
GND
3.3V

```
1 d# fn # b# noN# c p# l# of # $#&
ESP8266-12E/F Buzzer testing program
Build-in LED at GPIO-16(D0)
Buzzer pin at GPIO-14(D5)
```

The demo songs of Starwar and SuperMario are in demo programs.

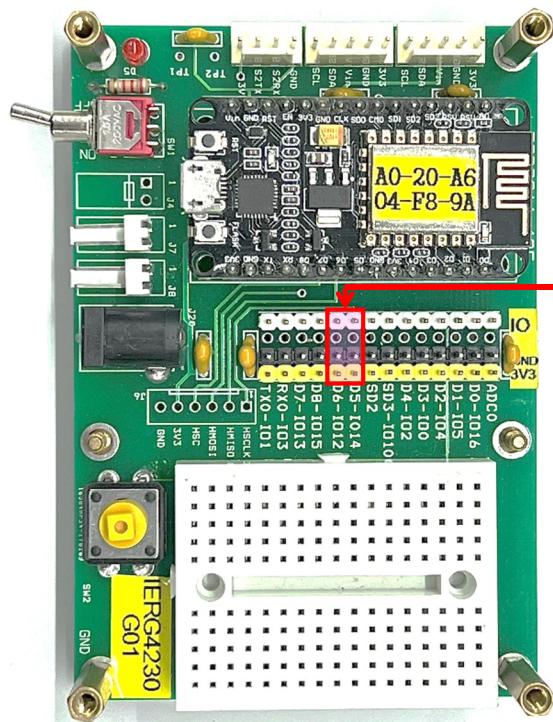


Provided device



Connect to the board

## C7. Dual tact switches (Digital)



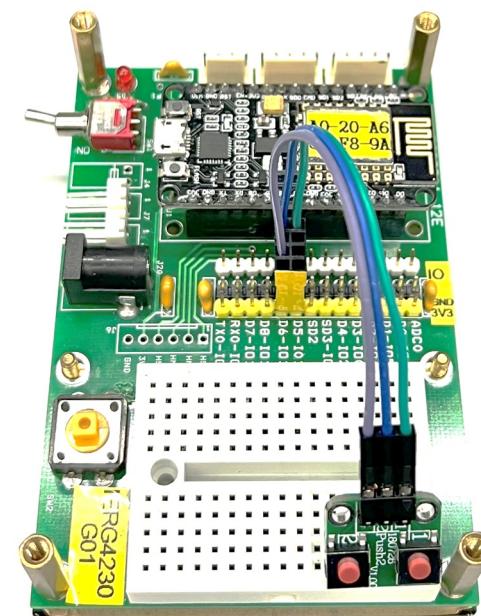
GPIO12 – D6 (Digital input)	GPIO14 – D5 (Digital input)
X	X
GND	X
3.3V	X

```
Build-in LED at GPIO-16/D0
Build-in LED at GPIO-2/D4
Detect Pin at GPIO-14/D5
Detect Pin at GPIO-12/D6
Key1 pressed
Key1 released
Key1 released
Key1 released
Key2 released
Key2 pressed
Key2 released
Key2 released
Key1 pressed
Key2 pressed
Key1 released
<
Autoscroll Show timestamp
No line ending 115200 baud Clear output
```

The result of the demo program

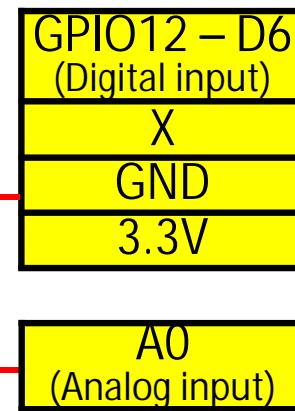
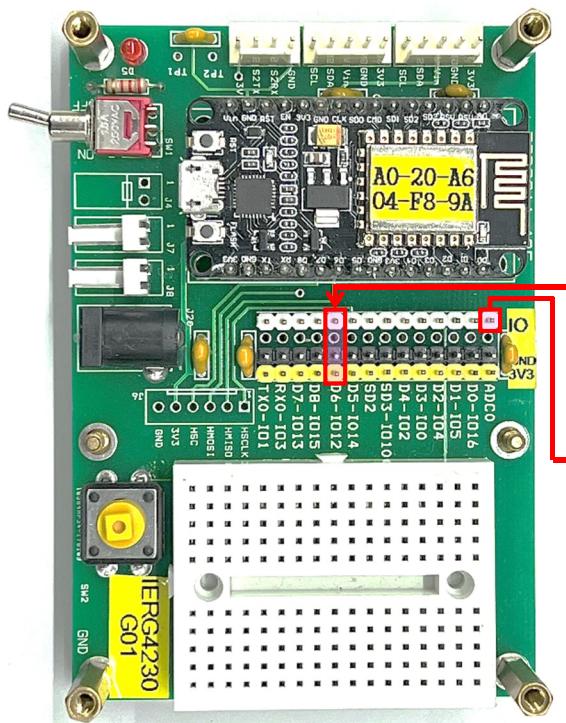


Provided device



Connect to the board

## C8. Light sensor module (Digital & Analogue)

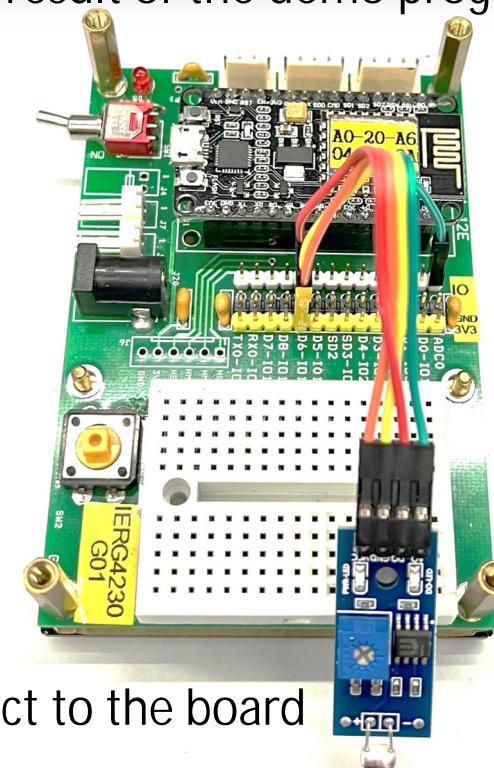


```
COM3
Build-in LED1 at GPIO-16(D0)
Mic analog Pin at A0
Mic digital Pin at GPIO-12(D6)
LDR(Analog)=821
LDR(Digital)=1
LDR is DARK, LED is ON
LDR(Analog)=152
LDR(Digital)=0
LED OFF
LDR(Analog)=158
LDR(Digital)=0
LED OFF
LDR(Analog)=158
LDR(Digital)=0
LED OFF
<
No line ending 115200 baud Clear output
```

The result of the demo program



Provided device

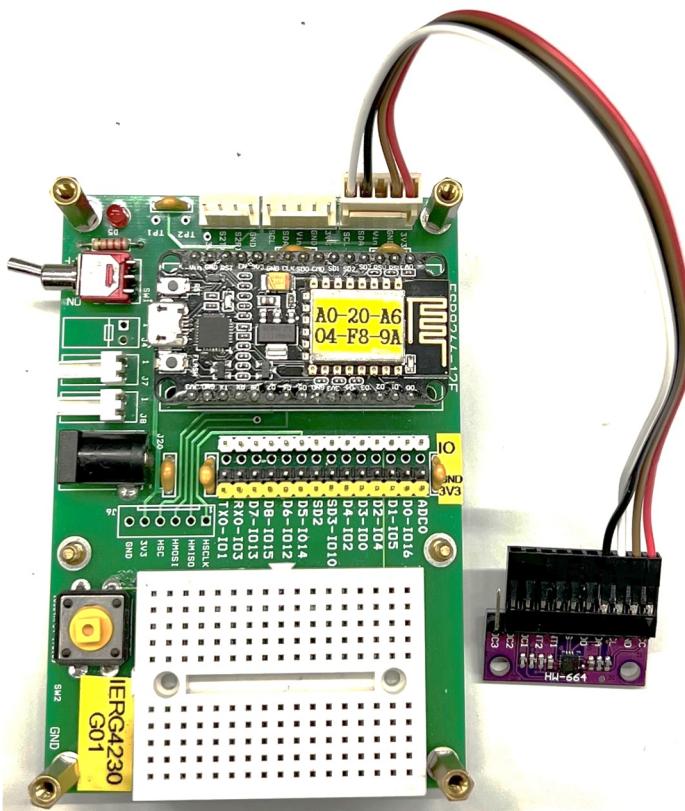


Connect to the board

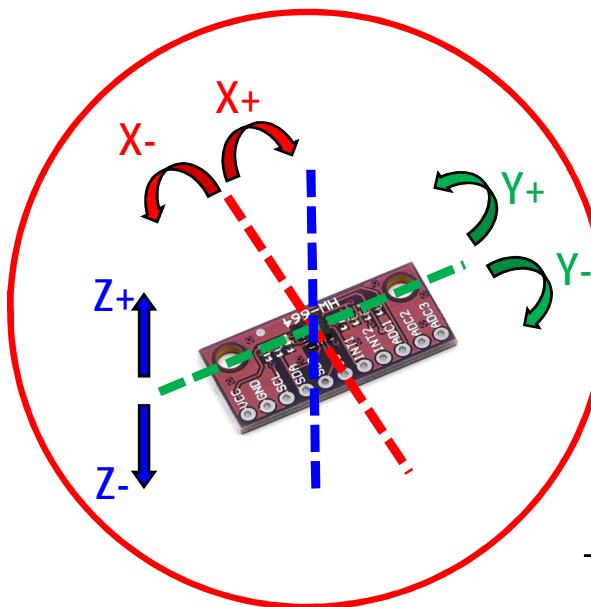
## C9. MEMS Motion sensor (LIS3DSH , $I^2C$ )



## Provided device



## Connect to the board



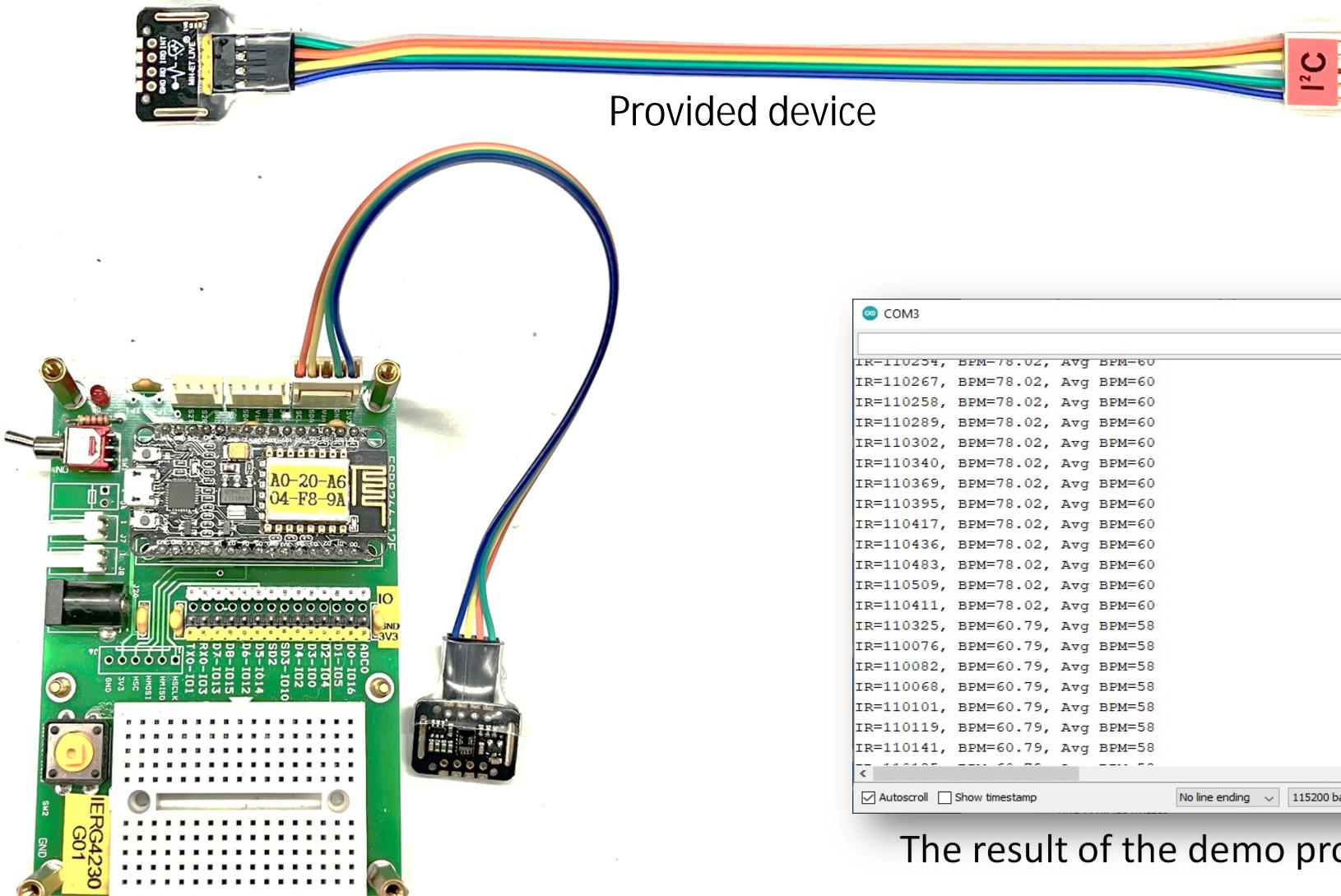
COM3

rd 1\$  \$1\$  ↗↑↖↙ b \$  \$p ↗{ \$# \$↑ \$p~\$o\$1Nn
Accel X: 1792 Y: 17664 Z: 512
Accel X: 3072 Y: 16128 Z: 768
Accel X: 3072 Y: 16128 Z: 768
Accel X: 3072 Y: 16384 Z: 1280
Accel X: 3072 Y: 15104 Z: 512
Accel X: 2816 Y: 16128 Z: 768
Accel X: -5376 Y: 17152 Z: 0
Accel X: 2560 Y: 17152 Z: 512
Accel X: 9216 Y: 11008 Z: 13824
Accel X: 3840 Y: 14848 Z: 23040
Accel X: 6144 Y: -13568 Z: 6144
Accel X: 2048 Y: 12288 Z: 19712
Accel X: -3328 Y: -9728 Z: -8960
Accel X: -5120 Y: 3072 Z: 16896
Accel X: 8704 Y: 32512 Z: 18176
Accel X: 5632 Y: 16128 Z: 22784
Accel X: 6144 Y: 32512 Z: 28928
Accel X: 4864 Y: 26880 Z: 11776
Accel X: 2560 Y: 12544 Z: 15616

Autoscroll  Show timestamp  No line ending

## The result of the demo program

# C10. Heart Rate Sensor (MAX30102, I<sup>2</sup>C)

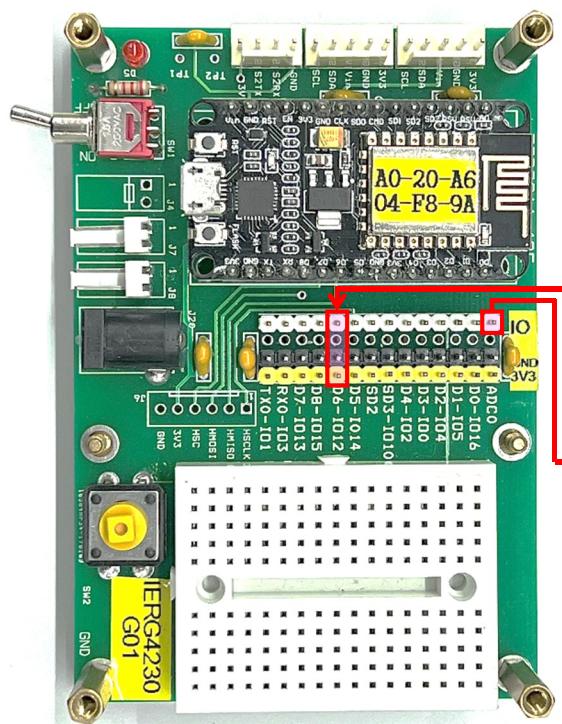


## Connect to the board

## The result of the demo program

39

# C11. Microphone (MIC, Digital & Analogue)



GPIO12 – D6  
(Digital input)  
X  
GND  
3.3V

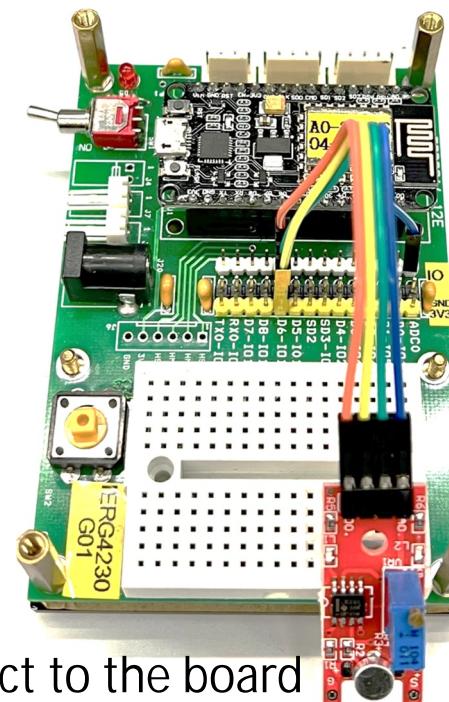
A0  
(Analog input)

```
Build-in LED1 at GPIO-16(D0)
Build-in LED2 at GPIO-2(D4)
Mic analog Pin at A0
Mic digital Pin at GPIO-12(D6)
Mic(Analog)=794
Mic(Digital)=0
Sound detected, LED is ON
Mic(Analog)=519
Mic(Digital)=1
LED OFF
Mic(Analog)=509
Mic(Digital)=1
LED OFF
Mic(Analog)=794
Mic(Digital)=0
```

The result of the demo program

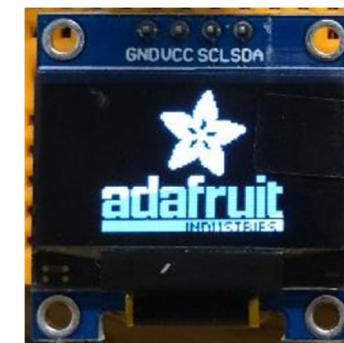
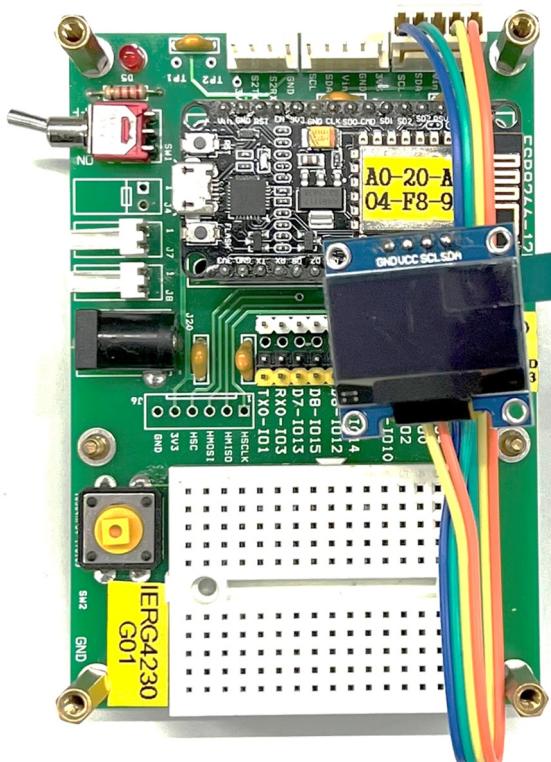
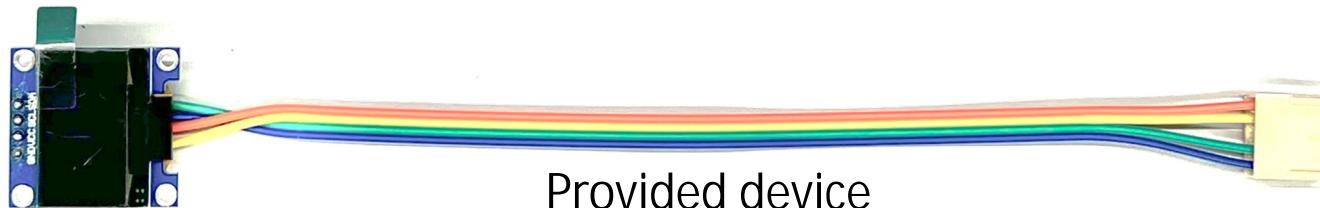


Provided device

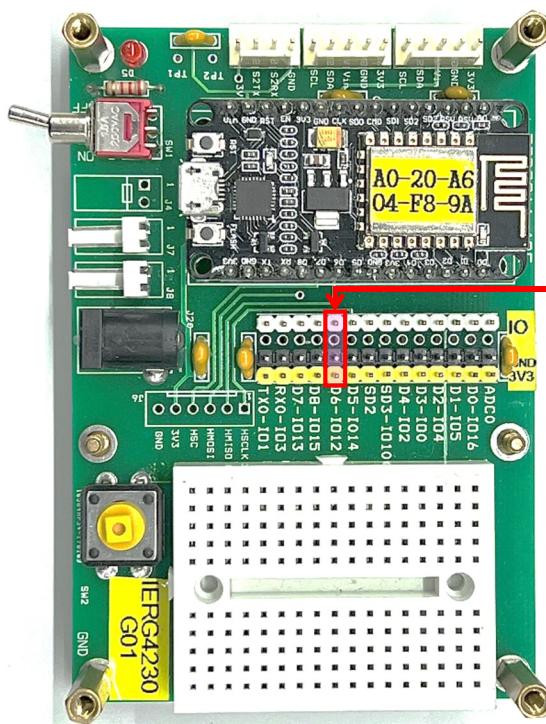


Connect to the board

## C12. OLE Display (SSD 1306 , $I^2C$ )



# C13. Touch switch (TTP223, Digital)



GPIO12 - D6
(Digital input)
X
GND
3.3V

```
Build-in LED at GPIO-16(D0)
Buzzer pin at GPIO-14(D5)
Capacitive Touch Sensor Pin at GPIO-12(D6)
Key pressed
Buzzer play
Key pressed
Buzzer play
No Key pressed
```

The result of the demo program

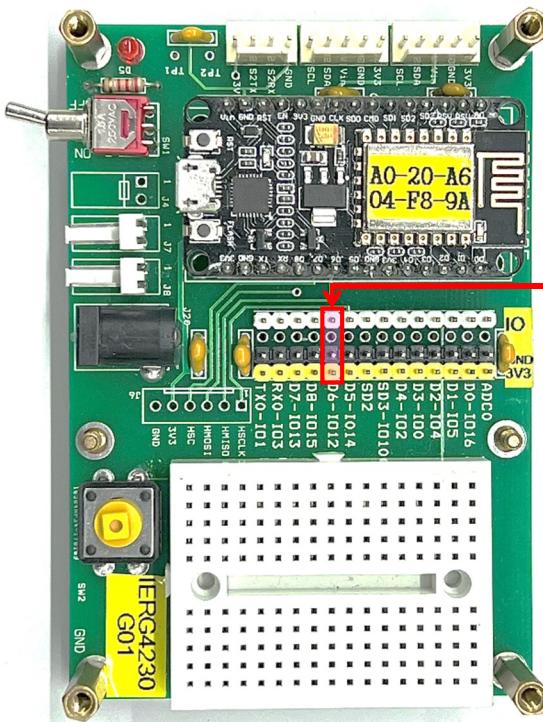


Provided device



Connect to the board

## C14. Microwave motion detector (RCWL-0516, Digital, 5V)



GPIO12 – D6  
(Digital input)  
5V  
GND  
X

```
No movement detected
No movement detected
Movement detected
Buzzer play
Movement detected
Buzzer play
Movement detected
Buzzer play
Movement detected
Buzzer play
```

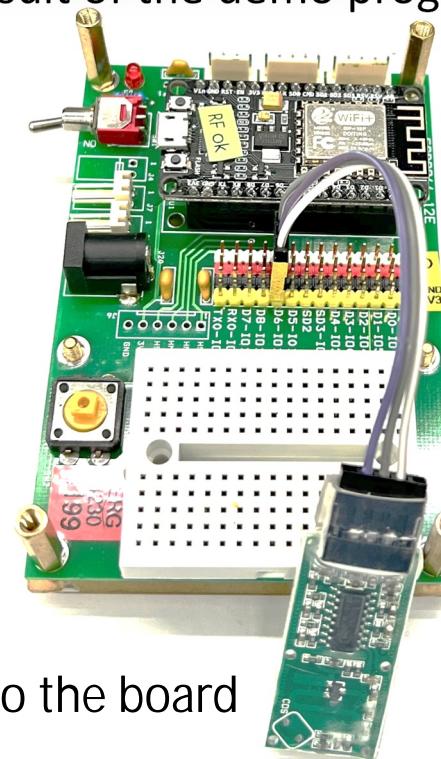
The result of the demo program



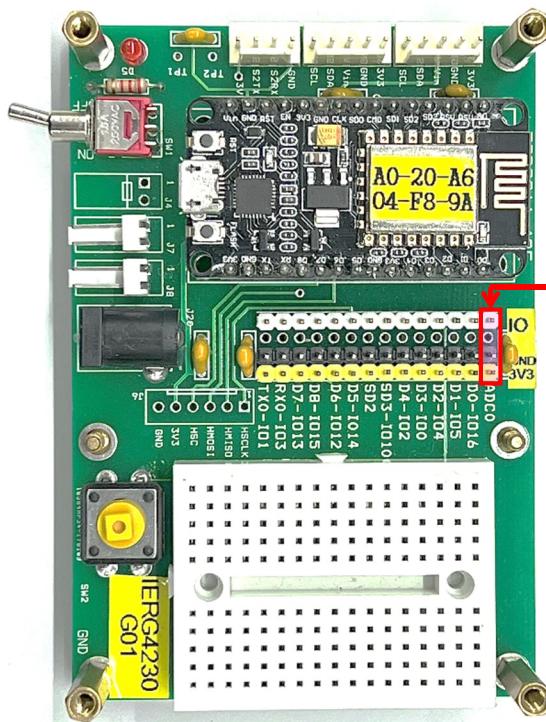
Provided device

This is 5V device  
This sensor is very sensitive.

Connect to the board



# C15. Soil Moisture Sensor (Analogue)



```
Soil Moisture (Analog)=20
Soil_Moisture is below threshold, LED is ON

Soil Moisture (Analog)=847
Soil_Moisture is above threshold, LED OFF

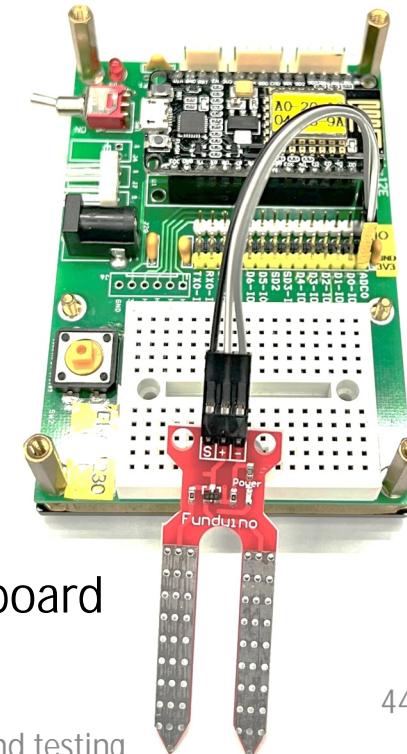
Soil Moisture (Analog)=97
Soil_Moisture is below threshold, LED is ON
```

The result of the demo program

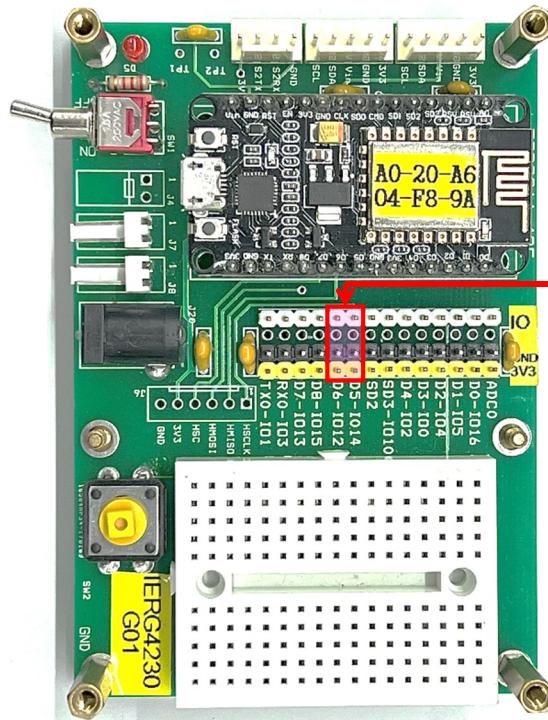


Provided device

Connect to the board



# C16. Ultrasonic Sensor Distance Measure Module (HC-SR04, Digital, 5V)



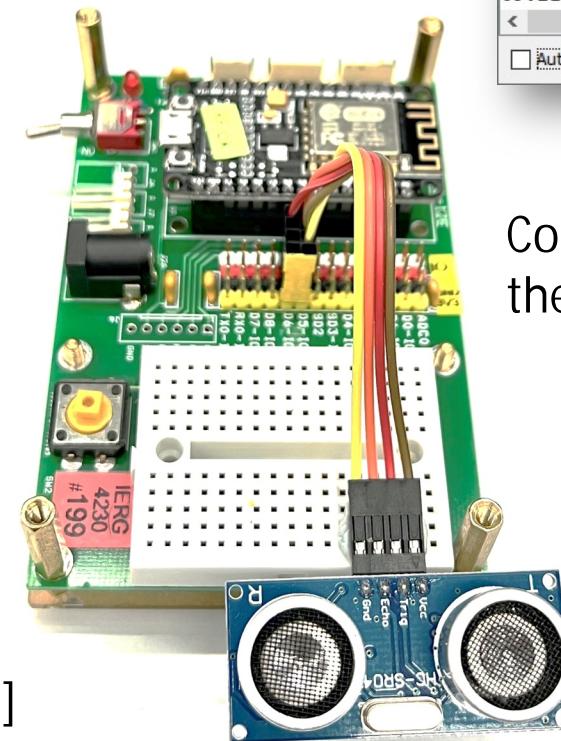
Provided device

- Use two digital I/Os  
[Sample code use D6 (GPIO 12) & D5 (GPIO14)]
- Power: **5V**

GPIO12 – D6 (Digital input)	GPIO14 – D5 (Digital input)
X	<b>5V</b>
X	GND
X	X

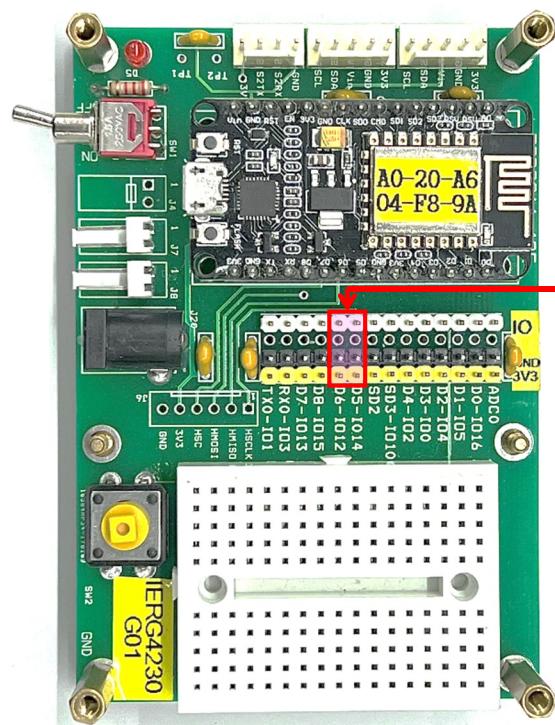
The result  
of the demo  
program

```
Build-in LED at GPIO-16(D0)
Trig Pin at GPIO-12(D6)
Echo Pin at GPIO-14(D5)
13.49 CM
12.65 CM
distance < 10cm, LED lit
3.26 CM
29.97 CM
Range exceed!
33.66 CM
201.72 CM
230.45 CM
69.21 CM
<
 Autoscroll  Show timestamp
```



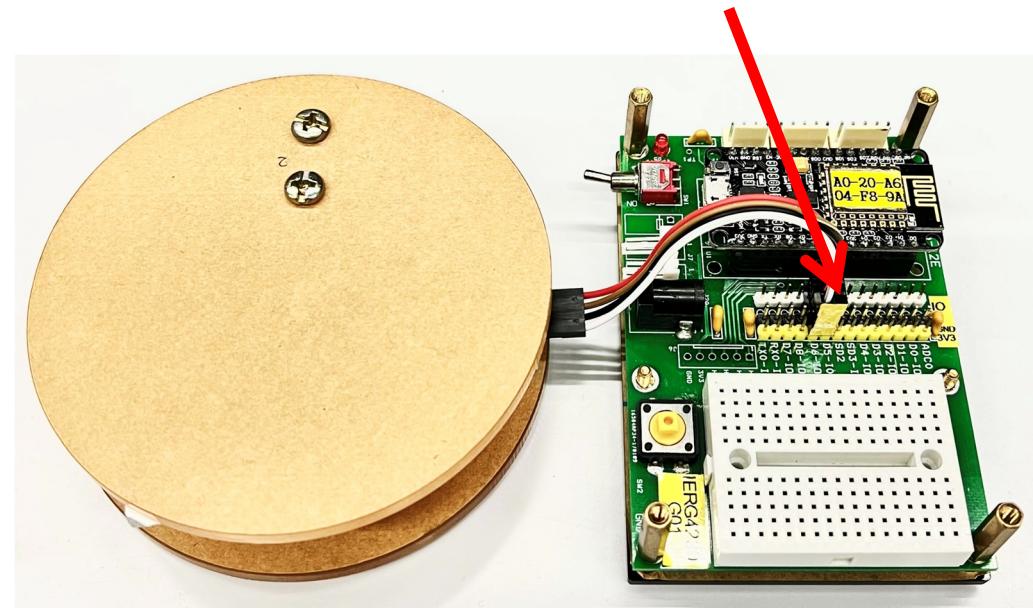
Connect to  
the board

# C17. Electronic scale, weight sensor (HX711AD, digital)



GPIO12 – D6 (Digital input)	GPIO14 – D5 (Digital input)
X	X
GND	X
3.3V	X

The coloured face  
connect to Yellow header



- Use two digital I/Os  
[Sample code use D6 (GPIO 12) & D5 (GPIO14)]
- Power: **3.3V**

# C17. Electronic scale, weight sensor (HX711AD, Digital)

- Decompress “[HX711.rar](#)” can get the test program for ESP8266 (Arduino platform) and datasheet.
- Folder “[HX711](#)” is the main program
- Before use the Electronic scale, you need to calibrate the scale first.
- Use folder “[HX711-Calibrate](#)” program to get the parameter “calibrate\_parameter” for main program (Line 38).

```
COM3

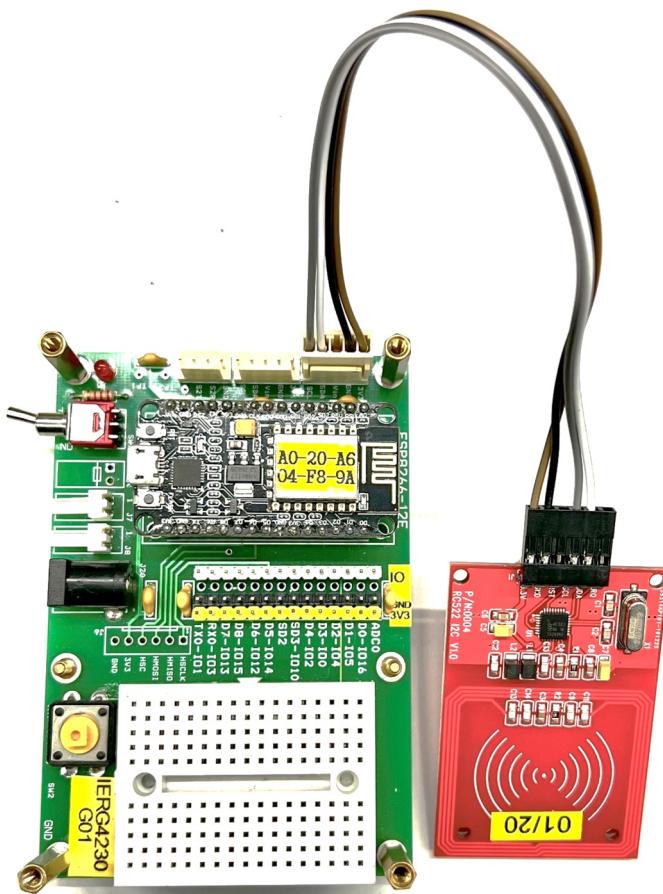
HX711 Demo
Before setting up the scale:
raw reading: -114910
read average: -114932 (20 readings)
get value: -115010.00 (5 readings)
get units: inf (5 readings)
After setting up the scale:
raw reading: -113767
read average: -114972 (20 readings)
get value: -143.00 (5 readings)
get units: -0.5 (5 readings)
Start Readings:
one reading: 9.4 g
one reading: 12.3 g
one reading: 72.2 g
one reading: 100.2 g
one reading: 100.1 g
one reading: 100.4 g
<
 Autoscroll  Show timestamp  No line endings
```

The result of the demo program

# C18. RFID reader (MFRC522, $I^2C$ )



Provided device



Connect to the board

```
COM3
r1 l$## $## | ↑ ↑?1?#|## ?##?#nN? $oN??#↑B p?#$` 

Build-in LED at GPIO-16(D0)
I2C SDA Pin at GPIO-4(D2)
I2C SCL Pin at GPIO-5(D1)

Test:0
MFRC522 Software Version: 0x92 = v2.0
Scan PICC to see UID, type, and data blocks...

Card UID Length: 4
Card UID: 36 5C CE F0

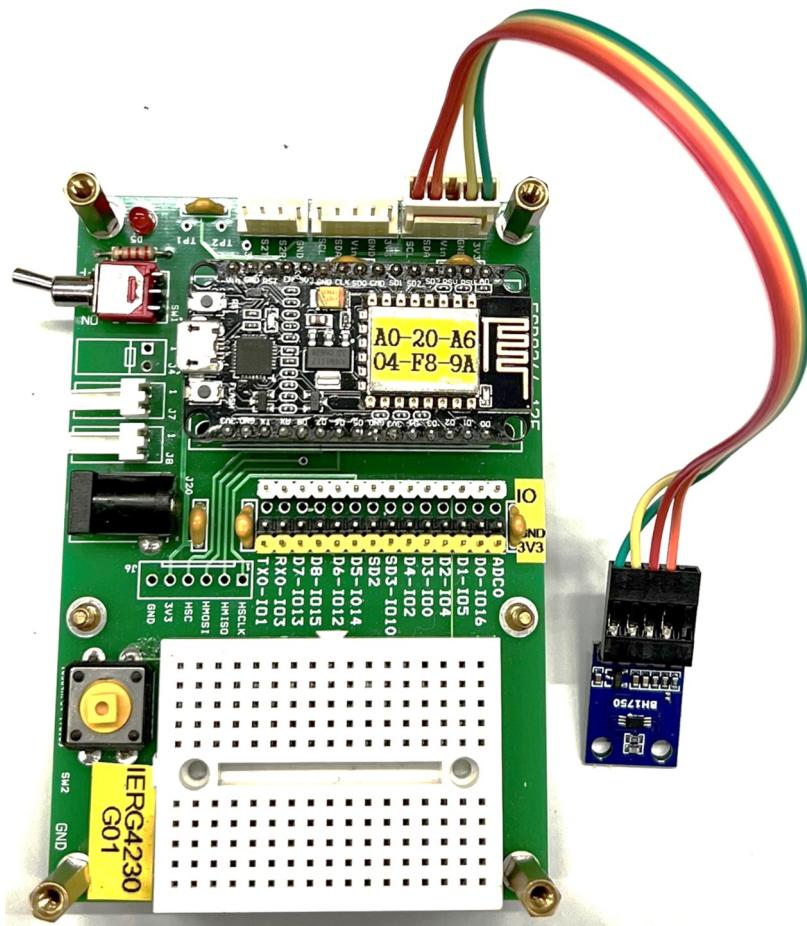
< 
 Autoscroll  Show timestamp No line ending 115200 ba
```

The result of the demo program

# C19. Light density detector (BH1750, $I^2C$ )



Provided device



Connect to the board

```
COM3
Build-in LED at GPIO-16(D0)
I2C SDA Pin at GPIO-2(D4)
I2C SCL Pin at GPIO-0(D3)
141[lx]
144[lx]
200[lx]
2981[lx]
1924[lx]
1377[lx]
1452[lx]
295[lx]
133[lx]
138[lx]
139[lx]
135[lx]
<     ▶
 Autoscroll  Show timestamp
```

The result of the demo program