

Summer Session KINGS 2019

Internet of Things Developments using NodeMCU and IoT Platform

Jong-Hyun Kim

jhkim@dit.ac.kr

Dept. of Computer & Information
DONG-EUI INSTITUTE OF TECHNOLOGY



Set status

김종현 | Jonghyun Kim
jhkim3217



Hello, I am an old programmer & educator, DIT(Dong-Eui Institute of Technology), Busan, Korea

Dongeui Institute of Technology

Busan, South Korea

jhkim@dit.ac.kr

<http://www.facebook.com/jhkim3217>

Organizations



IOT

Table of Contents

- Introduction to NodeMCU
- Basic Arduino Programming on NodeMCU
 - Arduino Digital I/O Functions
 - Circuit Wiring
 - Simple LED Control
 - Reading Humidity & Temperature Sensor Values
- IoT Platforms
 - Ubidots
- Hands-on Lab 01
 - IoT based Real Time Humidity & Temperature Monitor
- Hands-on Lab 02
 - Smart Switch Controller
- Future Study Topics

IoT Development Toolkits

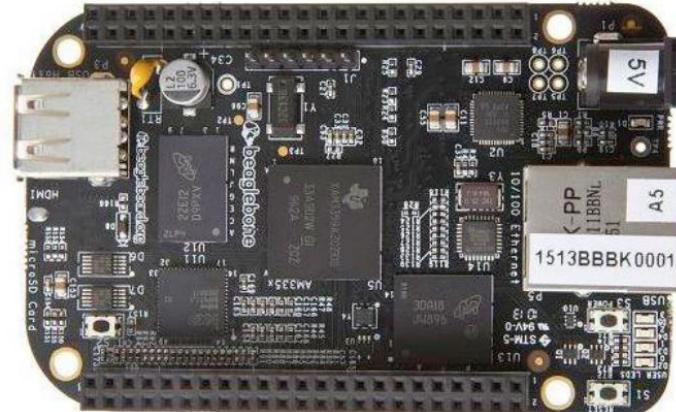
- Open Source Hardware Platforms



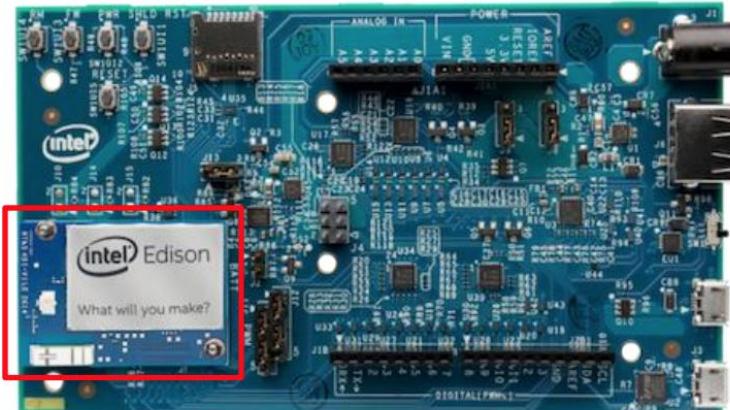
Arduino Uno



Raspberry Pi 3 B+



Beaglebone Black



Intel Edison



Intel Galileo

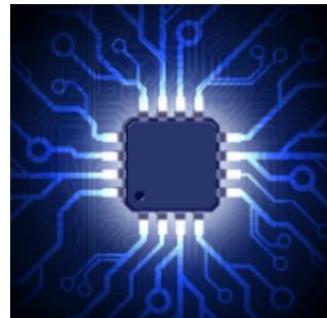


NodeMCU

NodeMCU/ ESP8266

- NodeMCU : An open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.
- ESP8266 : a low cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by Shanghai-based Chinese manufacturer, Espressif Systems.

Open-source, Interactive, Programmable, Low cost, Simple, Smart, WI-FI enabled



Arduino-like hardware
IO



Nodejs style network
API



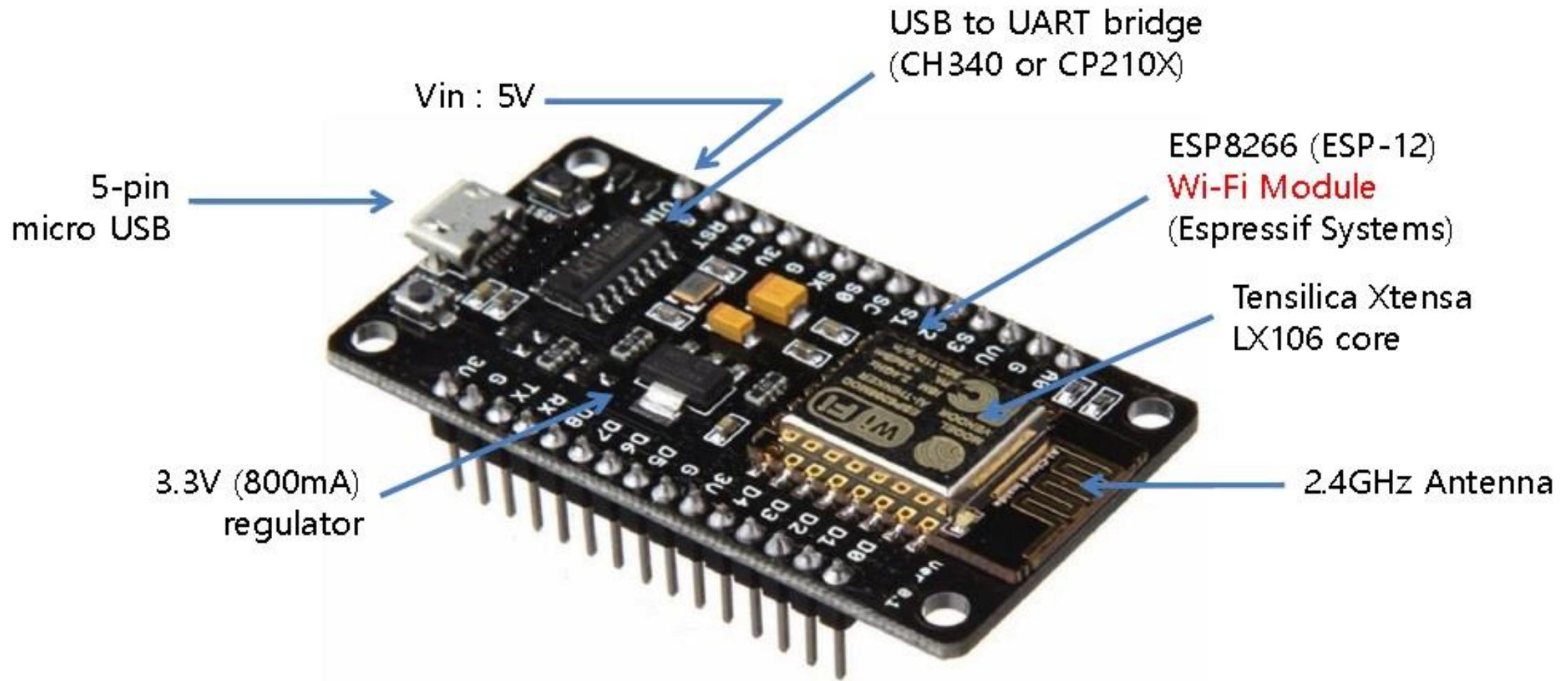
Lowest cost WI-FI

NodeMCU Development Kits

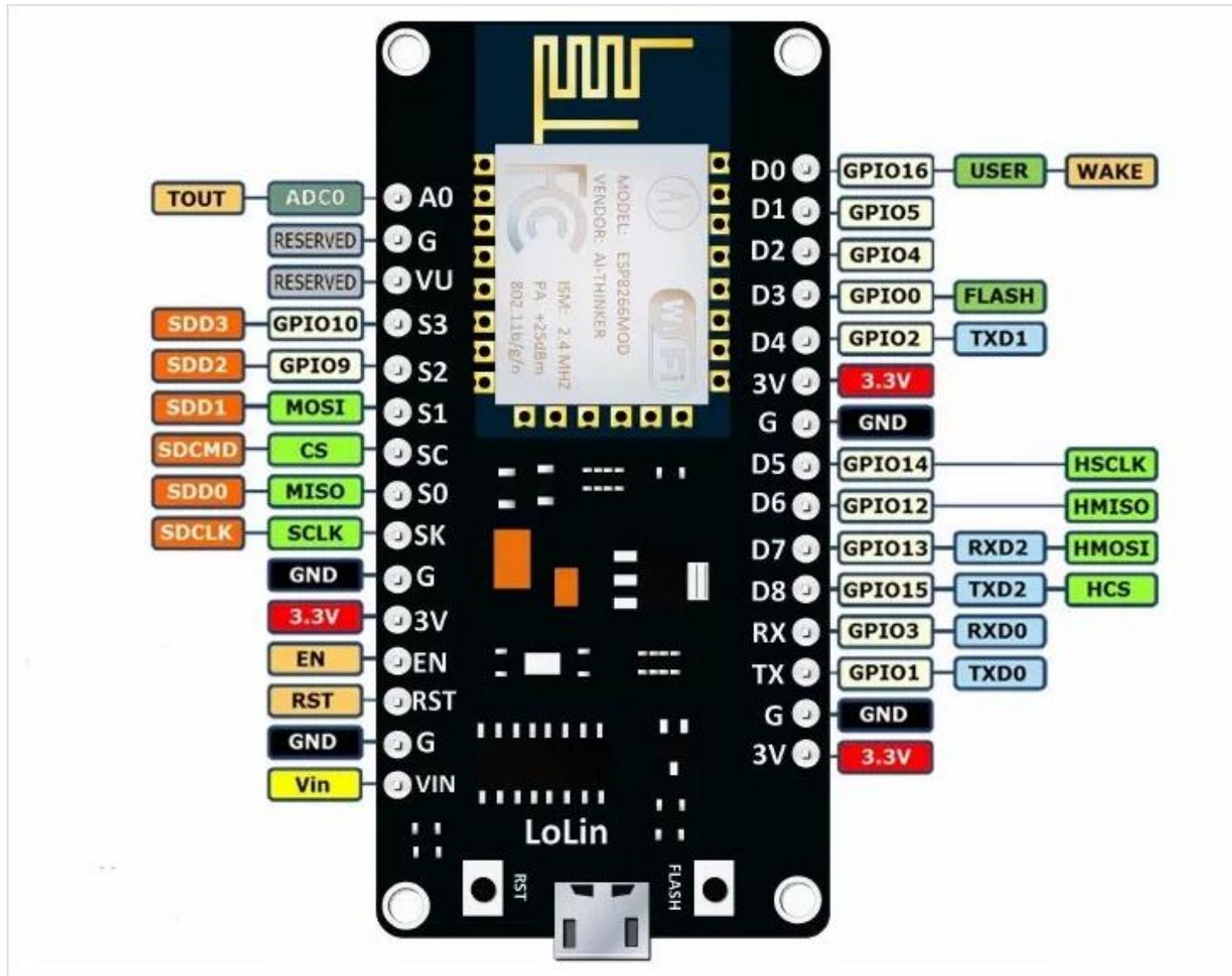
- The Development Kit based on ESP8266
- Integrates GPIO, PWM, I2C, 1-Wire and ADC all in one board
- Growth of ESP8266 modules



NodeMCU : Open Source H/W Platform



NodeMCU GOIP



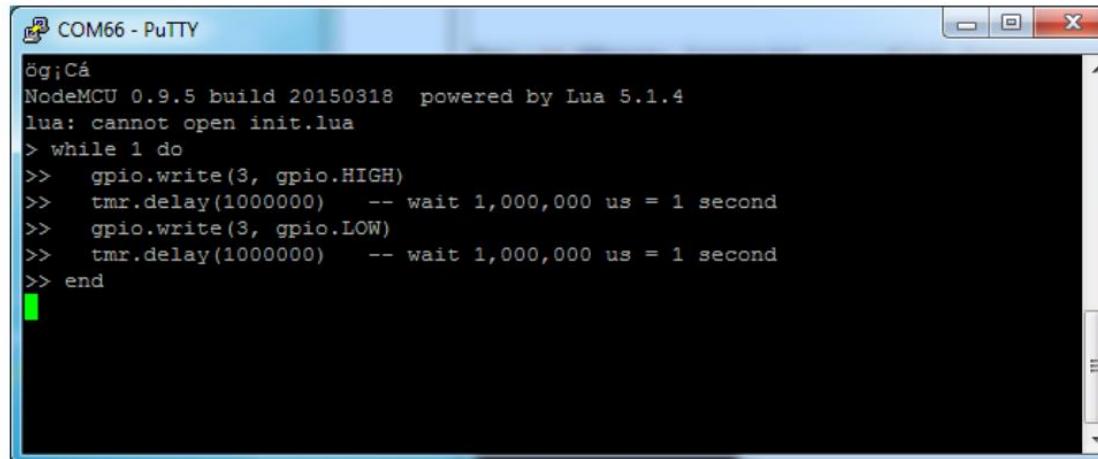
NodeMCU GOIP

- GPIO 0,2,15 : booting mode setting
- GPIO 1,3 : Serial Communication
- GPIO 6~11 : flash memory
- GPIO 16 : sleep mode
- Unrestricted GPIO Pins
 - GPIO 4(D2), GPIO 5(D1), GPIO 12~14(D6,D7,D5)
 - internal LED : GPIO 2(D4)

NodeMCU Development Environments

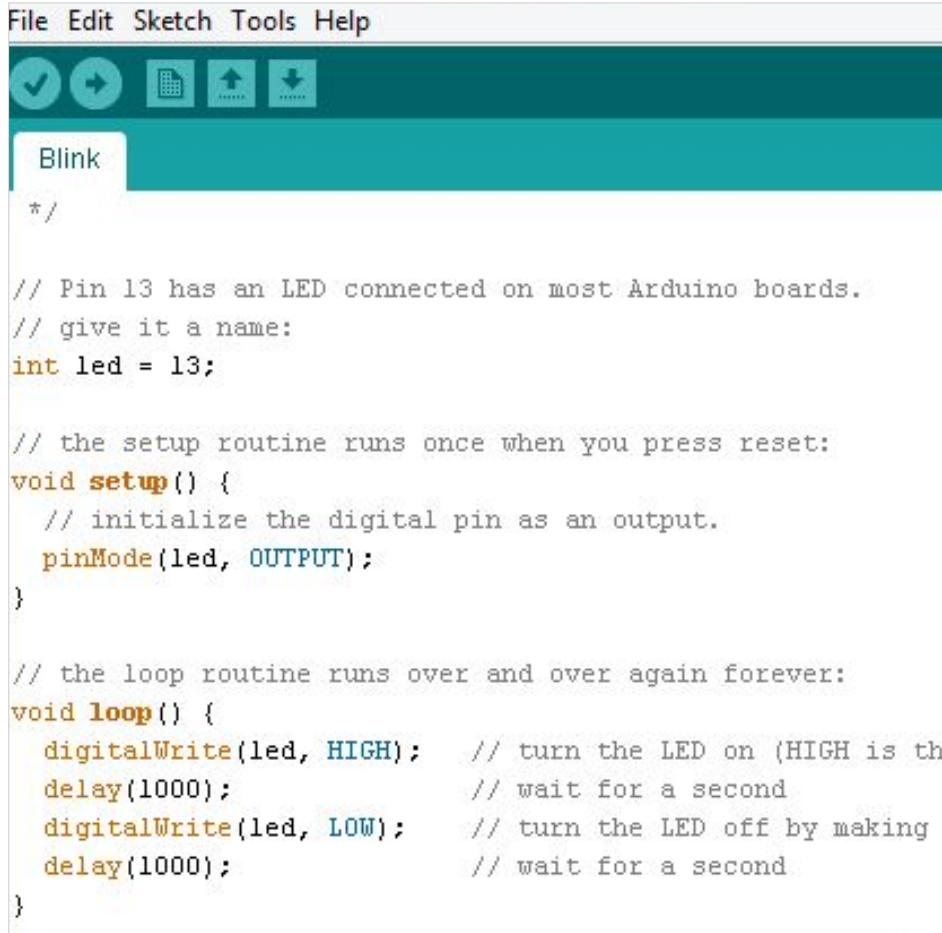
- LUA : Script language, Interpreter

```
1. gpio.mode(3, gpio.OUTPUT)
2. while 1 do
3.   gpio.write(3, gpio.HIGH)
4.   tmr.delay(1000000) -- wait 1,000,000 us = 1 second
5.   gpio.write(3, gpio.LOW)
6.   tmr.delay(1000000) -- wait 1,000,000 us = 1 second
7. end
```



NodeMCU Development Environments

- Arduinio IDE : CC++, Compiler



The image shows a screenshot of the Arduino IDE interface. The menu bar at the top includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with five icons: a checkmark, a right arrow, a folder, an upload symbol, and a refresh symbol. The main workspace is titled "Blink". The code editor contains the classic "Blink" sketch:

```
File Edit Sketch Tools Help
Blink
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
    digitalWrite(led, HIGH);      // turn the LED on (HIGH is the
                                // signal's "on" state)
    delay(1000);                // wait for a second
    digitalWrite(led, LOW);       // turn the LED off by making the
                                // signal's "off" state
    delay(1000);                // wait for a second
}
```

Adruino Programming for NodeMCU

- <https://www.arduino.cc>

The screenshot shows the Arduino website's download page. At the top, there is a navigation bar with links for HOME, STORE, SOFTWARE (which is circled in red), EDU, RESOURCES, COMMUNITY, and HELP. To the right of the navigation bar are icons for search, shopping cart, and sign in. Below the navigation bar, the text "Download the Arduino IDE" is displayed. On the left side, there is a large teal circle containing the Arduino logo (a minus sign and a plus sign forming an infinity symbol). To the right of the logo, the text "ARDUINO 1.8.9" is shown, followed by a description of the software and instructions for getting started. On the right side of the page, there is a teal sidebar containing links for Windows (Windows Installer for Windows XP and up, Windows ZIP file for non-admin install, Windows app Get), Mac OS X (Mac OS X 10.8 Mountain Lion or newer), Linux (Linux 32 bits, Linux 64 bits, Linux ARM 32 bits, Linux ARM 64 bits), Release Notes, Source Code, and Checksums (sha512). The "Windows" section of the sidebar is also circled in red.

HOME STORE SOFTWARE EDU RESOURCES COMMUNITY HELP

SIGN IN

Download the Arduino IDE

ARDUINO 1.8.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

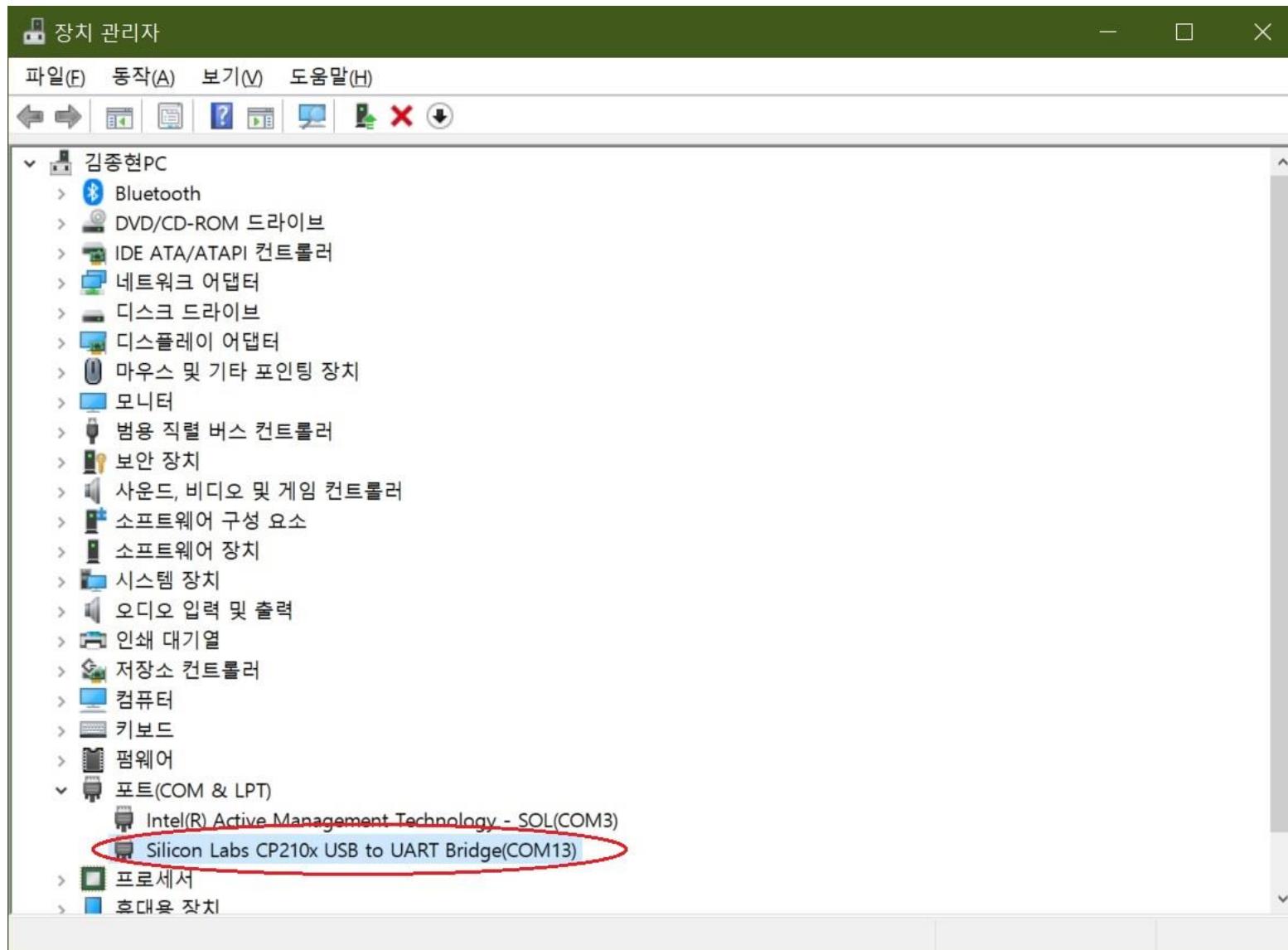
Windows Installer, for Windows XP and up
Windows ZIP file for non-admin install
Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.8 Mountain Lion or newer

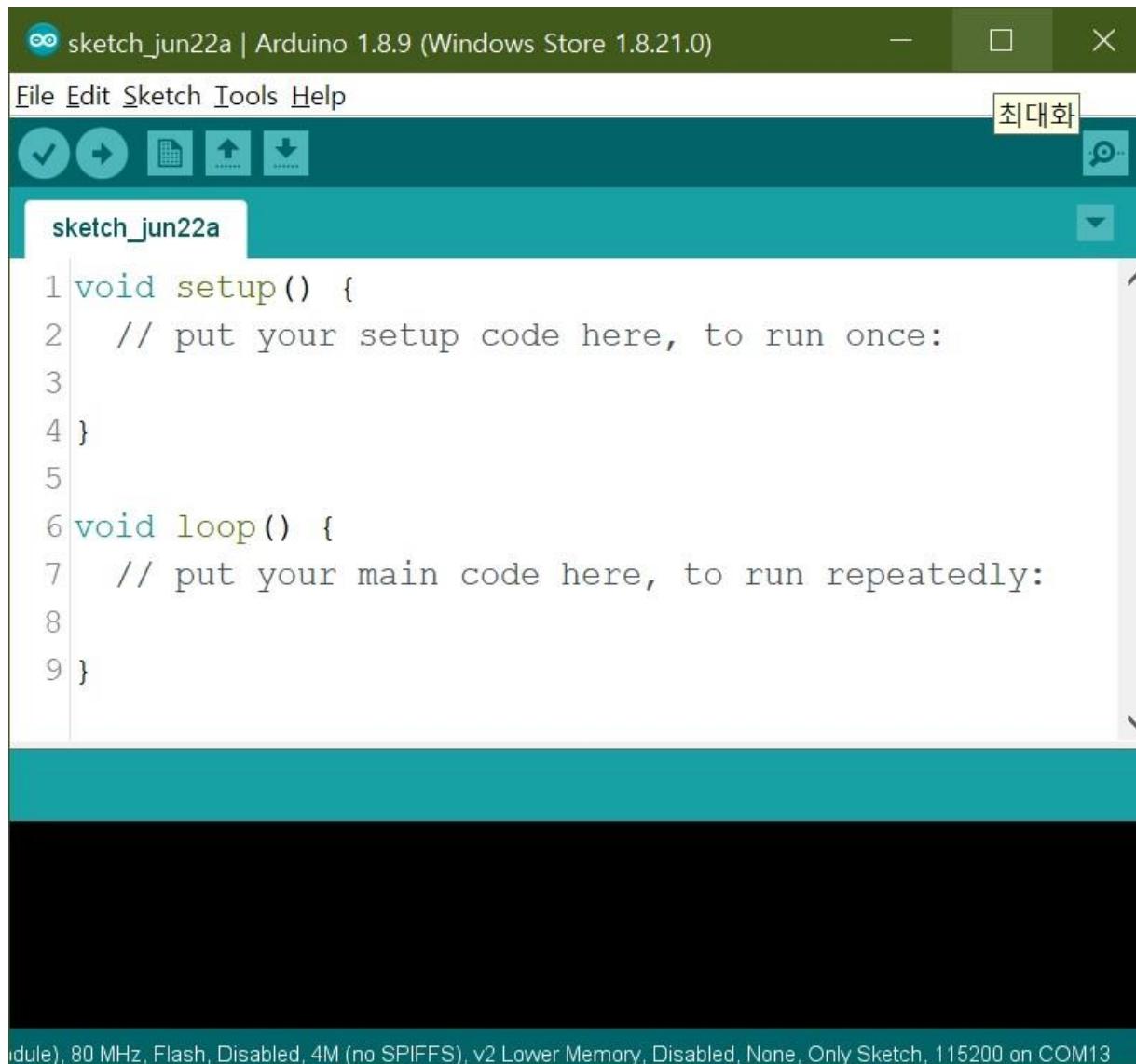
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

- Connect USB microcable to PC
- Open Device manager of PC, check Silicon Labs CP201x USB to UART Bridge



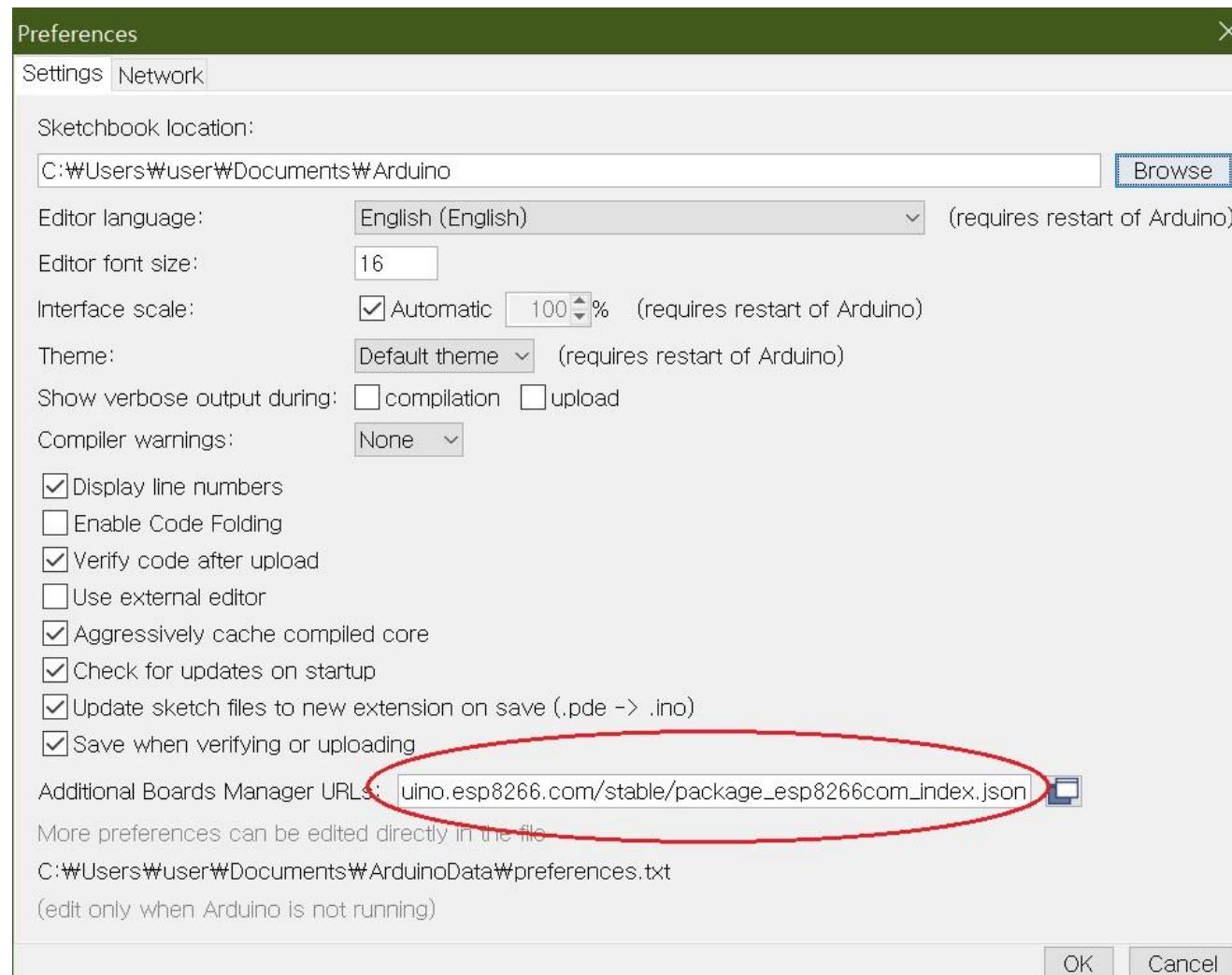
- Executing Arduino IDE



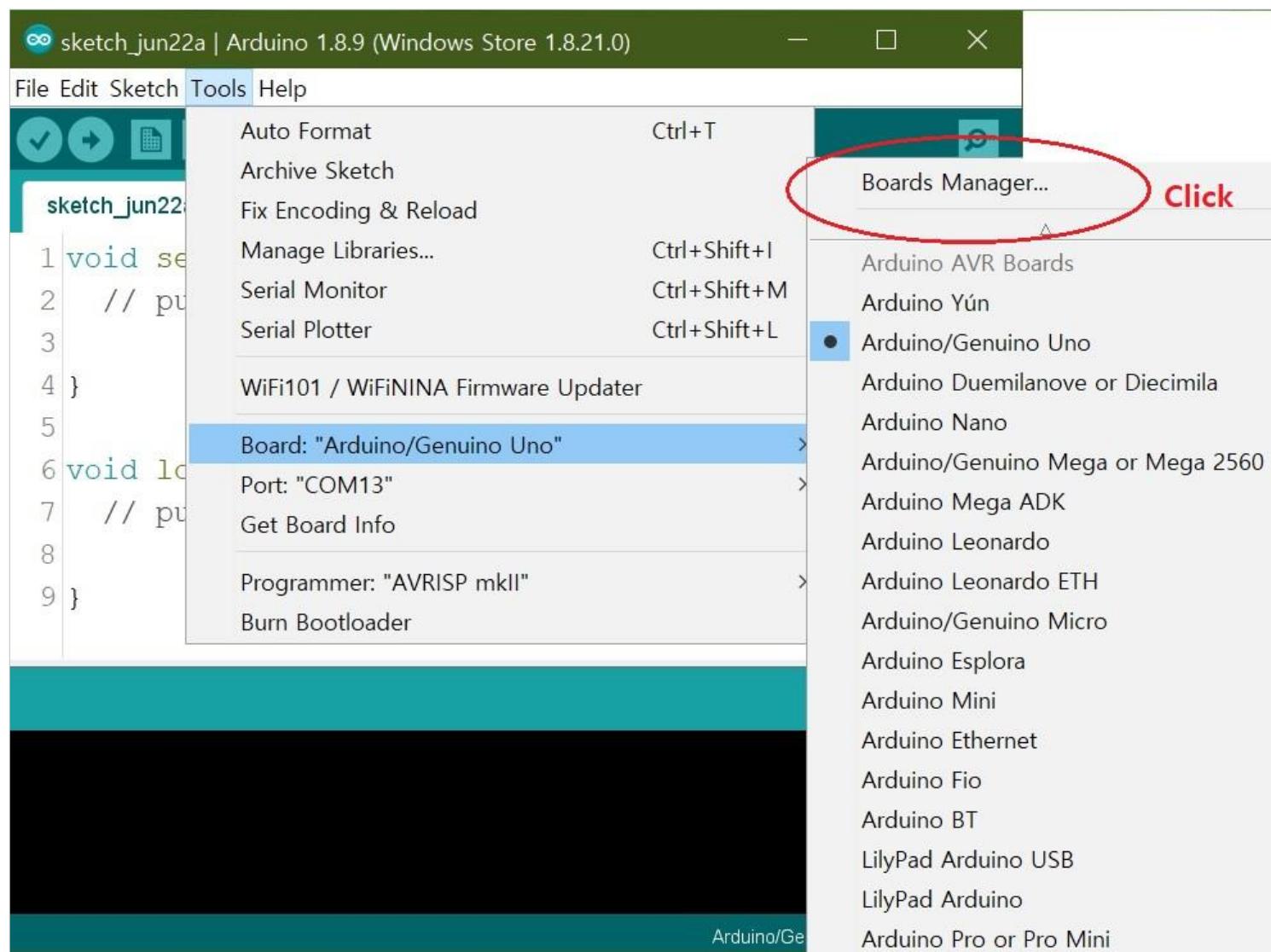
- Setting Board Manager URLs

- Preferences -> Additional Board Manager URLs

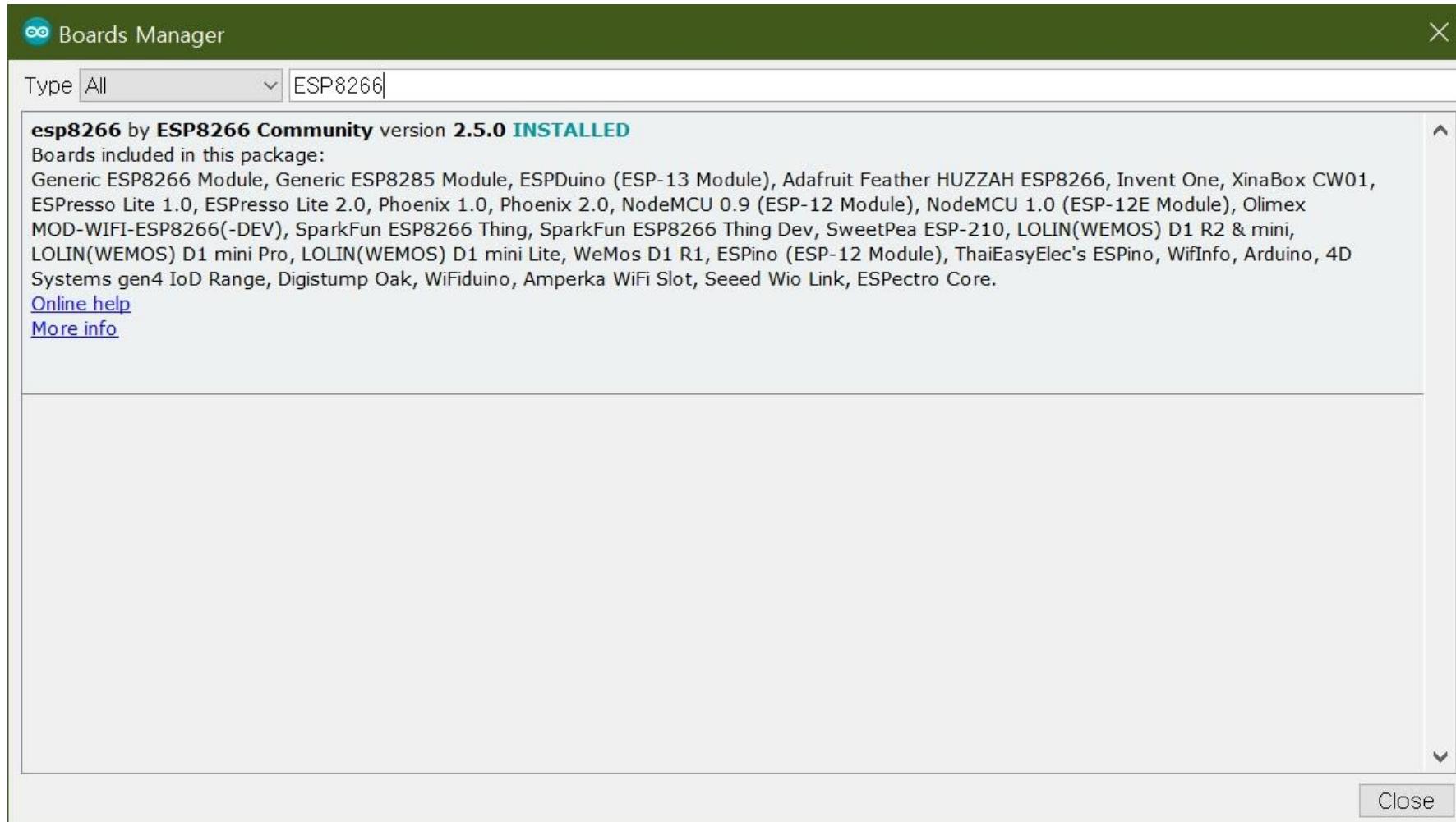
"http://arduino.esp8266.com/stable/package_esp8266com_index.json"



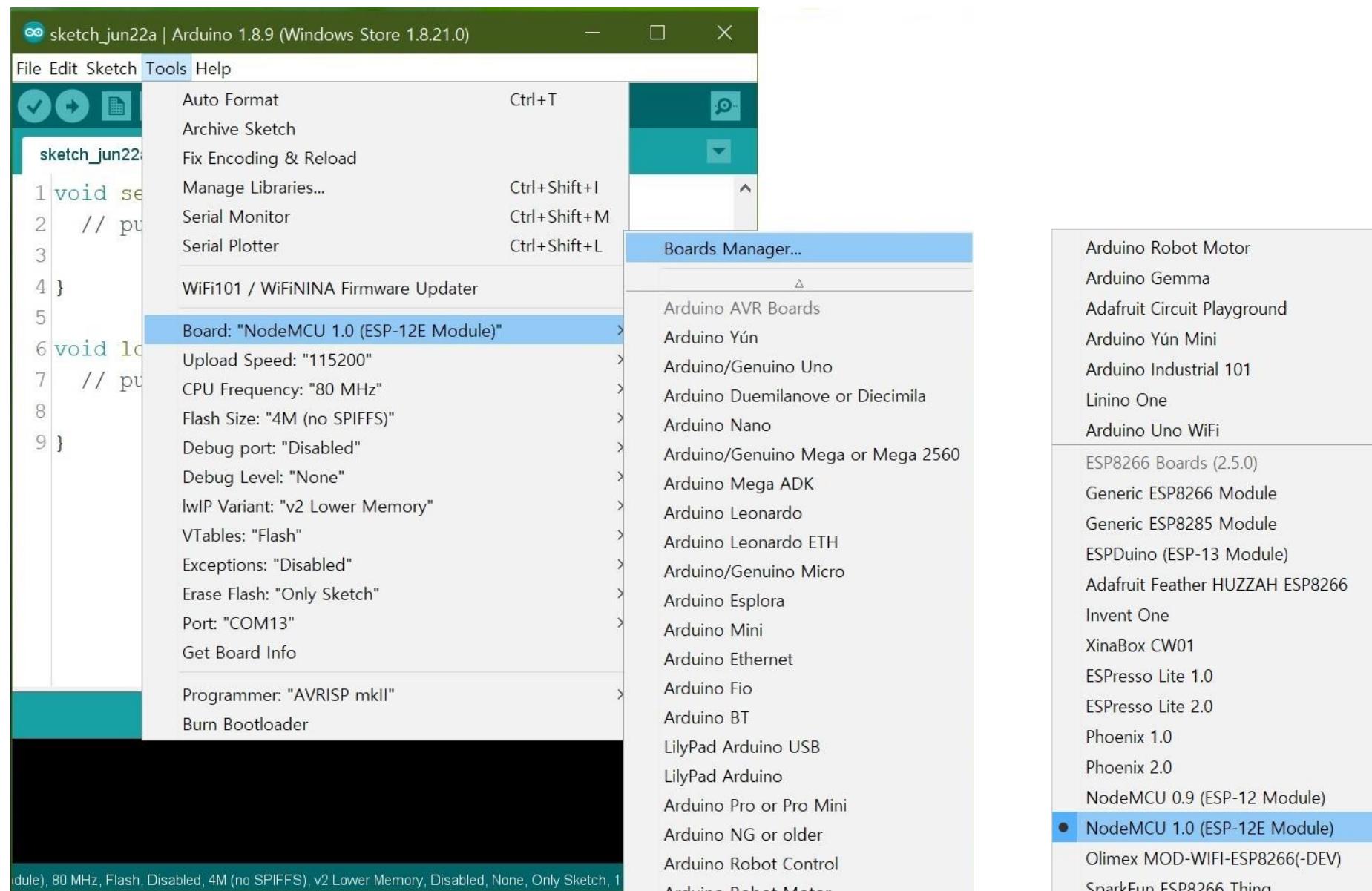
• Setting Board Manager



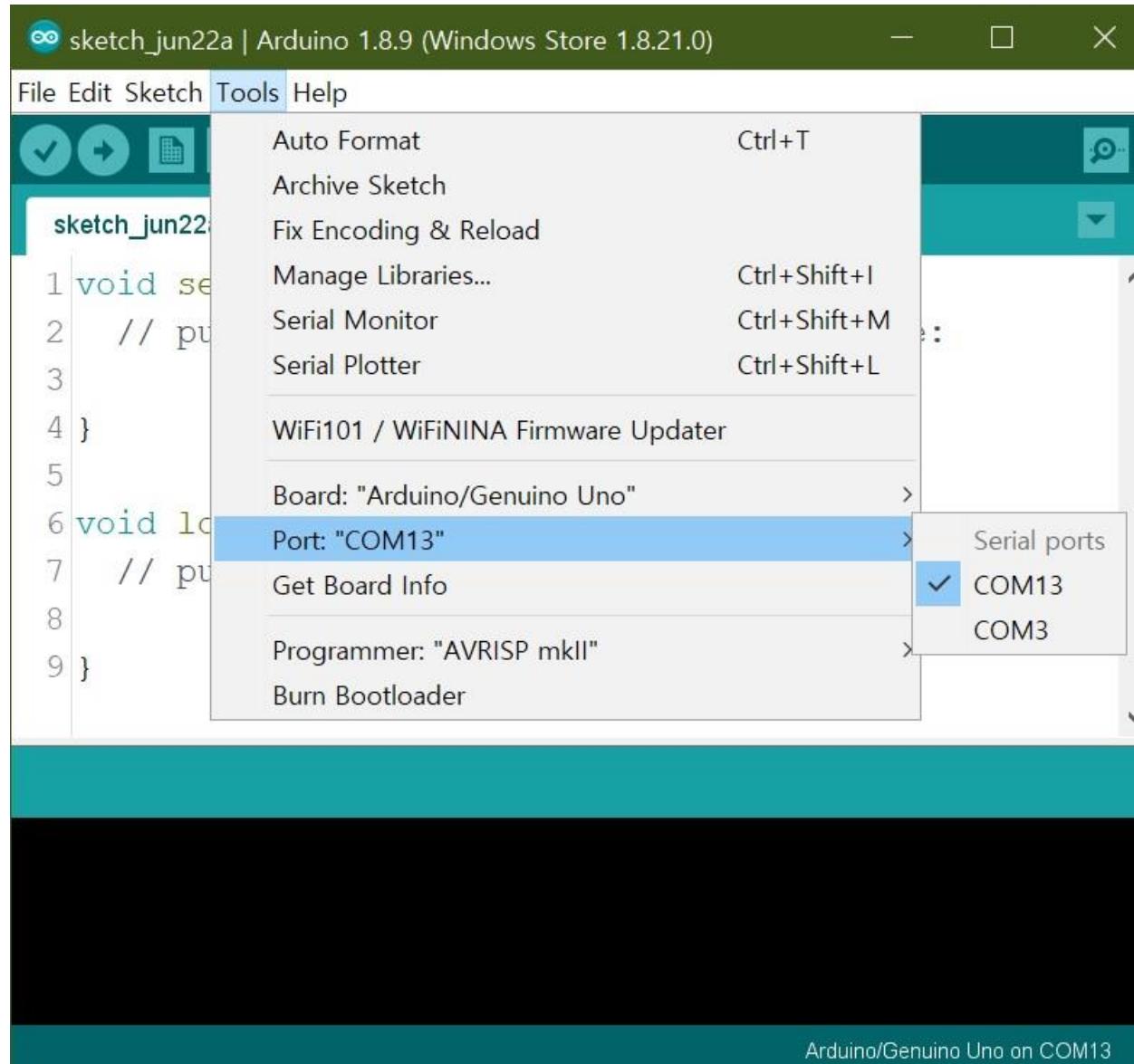
- Setting Board Manager : Search “esp8266 by ESP8266 Community”



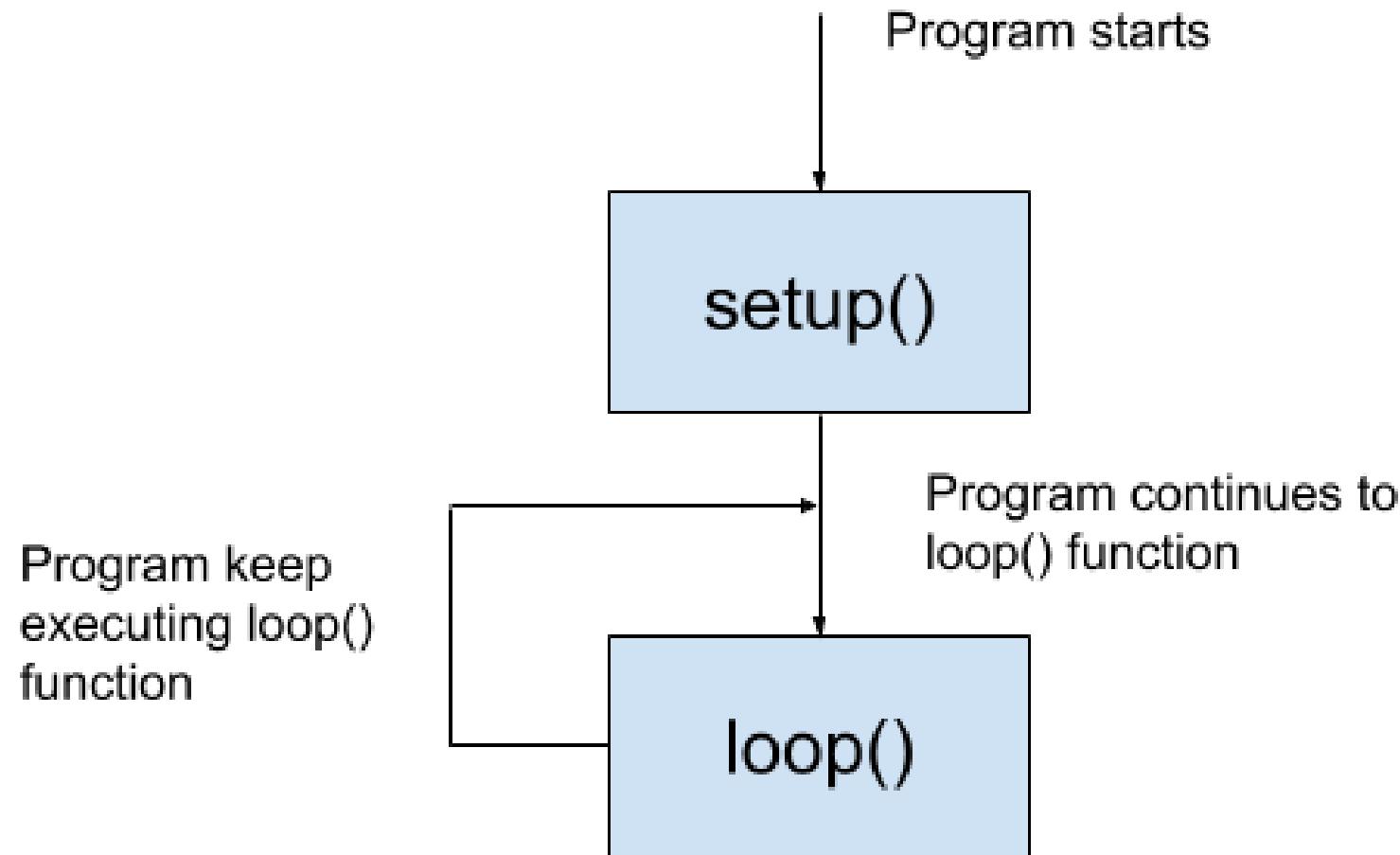
- Setting Board Manager : Select NodeMCU 1.0(ESP-12E Module)



- Setting serial port

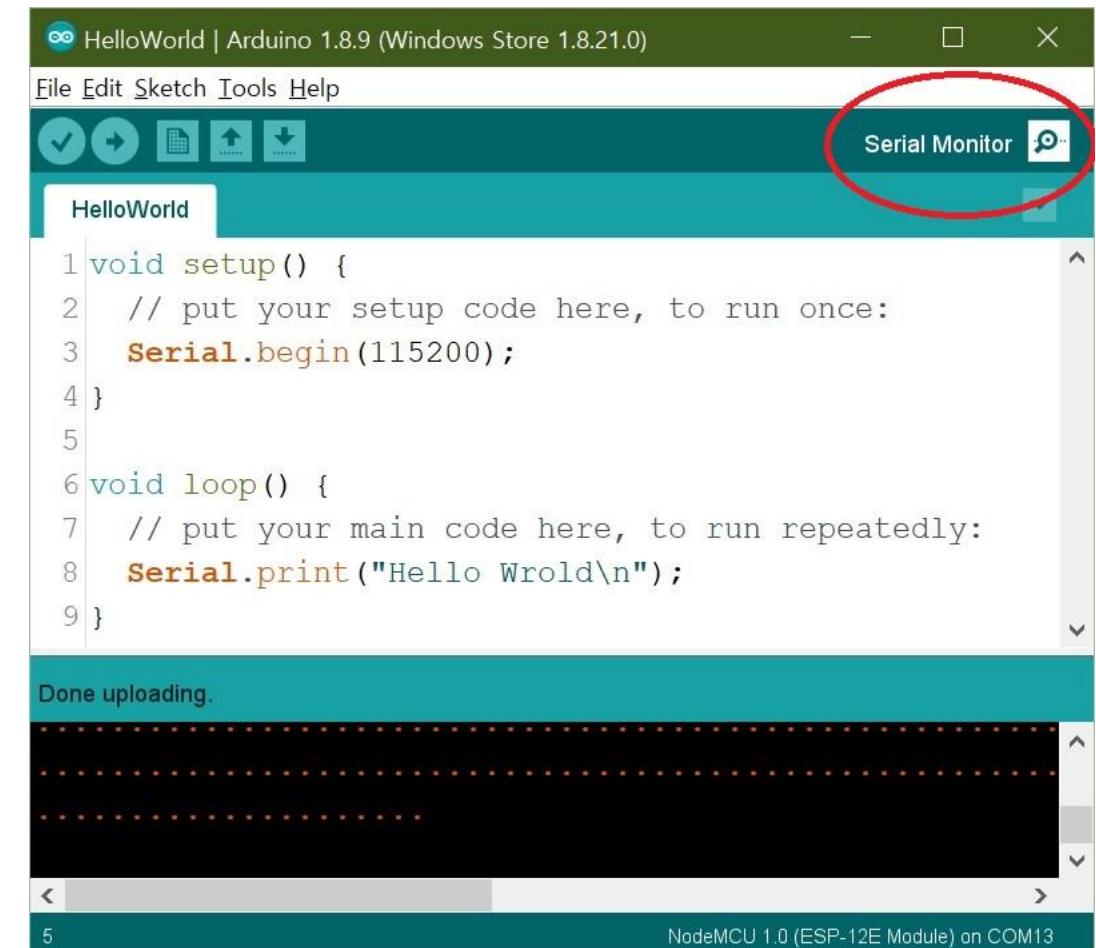


Arduino Programming Basics



Hello World!

```
void setup() {  
    // put your setup code here, to  
    run once:  
    Serial.begin(115200);  
  
}  
  
void loop() {  
    // put your main code here, to run  
    repeatedly:  
    Serial.print("Hello World\n");  
}
```



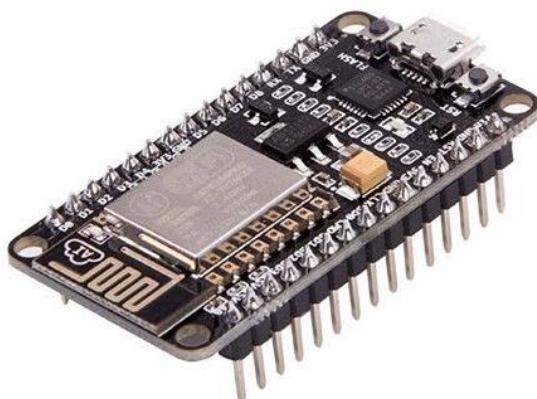
```
COM13  
Hello Wrold  
Hello Wrold  
Hello Wrold  
Hello Wrold  
Hello Wrold  
Hello Wrold  
Hello Wrold
```

Arduino Digital I/O functions

- **pinMode(pin, mode)**
 - Configures the specified pin to behave either as an input or an output.
- **digitalWrite(pin, value)**
 - Write a **HIGH(5V or 3.3V)** or a **LOW(0V or ground)** value to a digital pin
- **delay(ms)**
 - Pauses the program for the amount of time (in milliseconds) specified as parameter.
- <https://www.arduino.cc/reference/en/>

Basic LED Control on NodeMCU

- Preparation materials



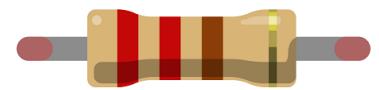
NodeMCU



Jump Cables

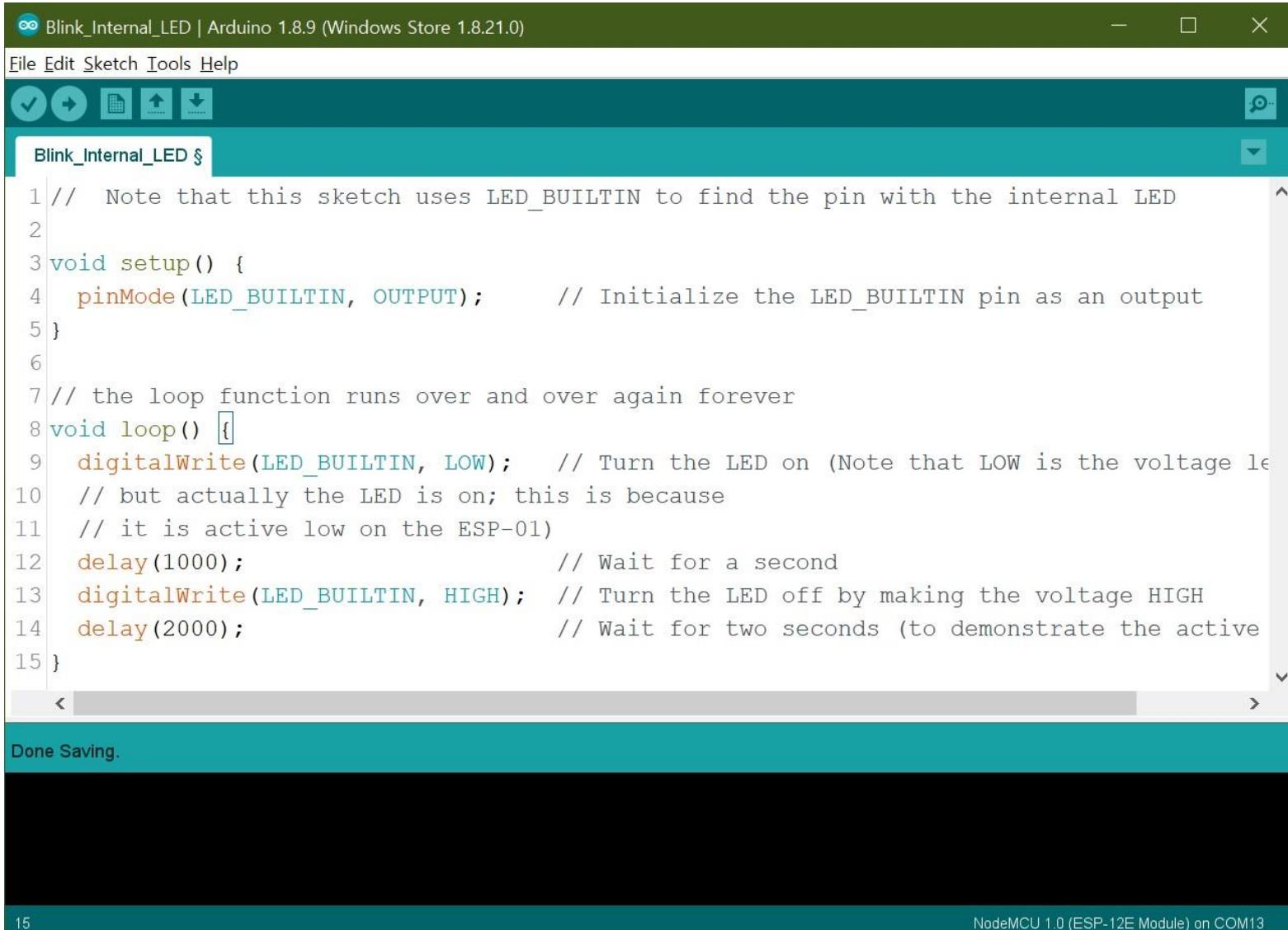


LED



Register
 $220\ \Omega$

Internal LED : LED_BUILTIN



The screenshot shows the Arduino IDE interface with the title bar "Blink_Internal_LED | Arduino 1.8.9 (Windows Store 1.8.21.0)". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for save, upload, and search. The code editor window contains the following sketch:

```
// Note that this sketch uses LED_BUILTIN to find the pin with the internal LED
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);      // Initialize the LED_BUILTIN pin as an output
}

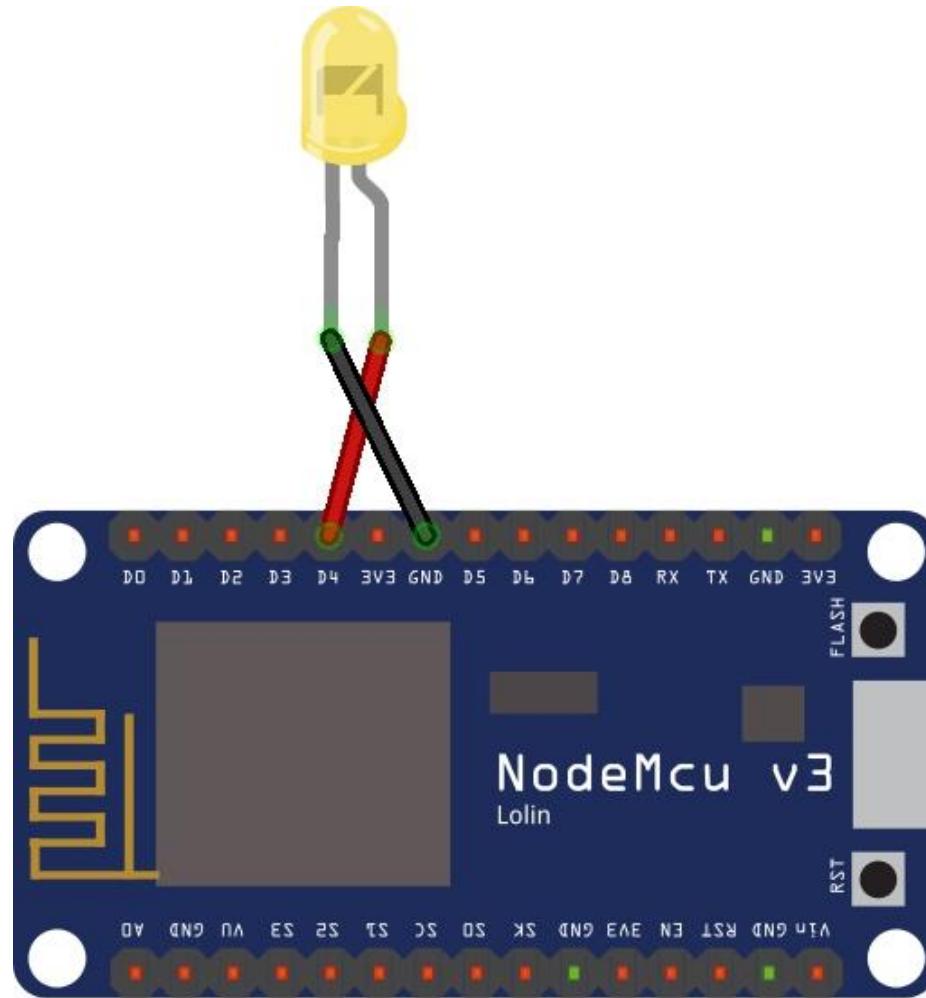
// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, LOW);    // Turn the LED on (Note that LOW is the voltage level
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
    delay(1000);                    // Wait for a second
    digitalWrite(LED_BUILTIN, HIGH);   // Turn the LED off by making the voltage HIGH
    delay(2000);                    // Wait for two seconds (to demonstrate the active
} // end of loop

Done Saving.
```

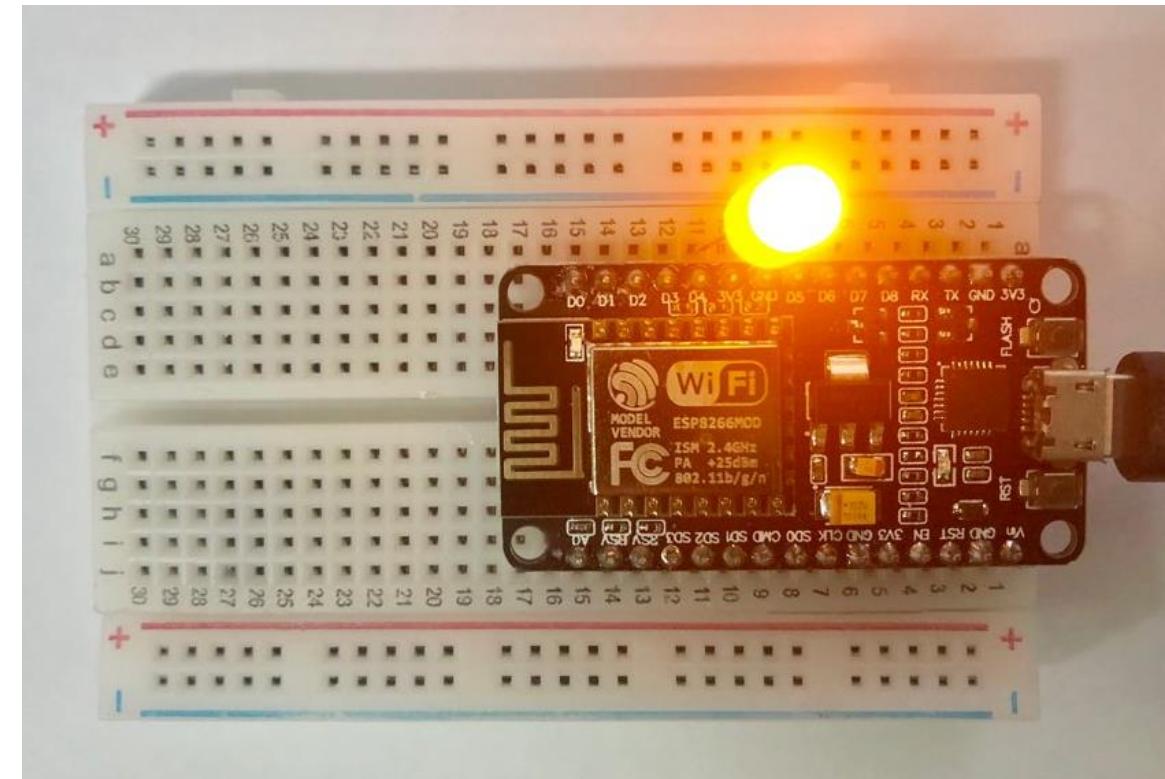
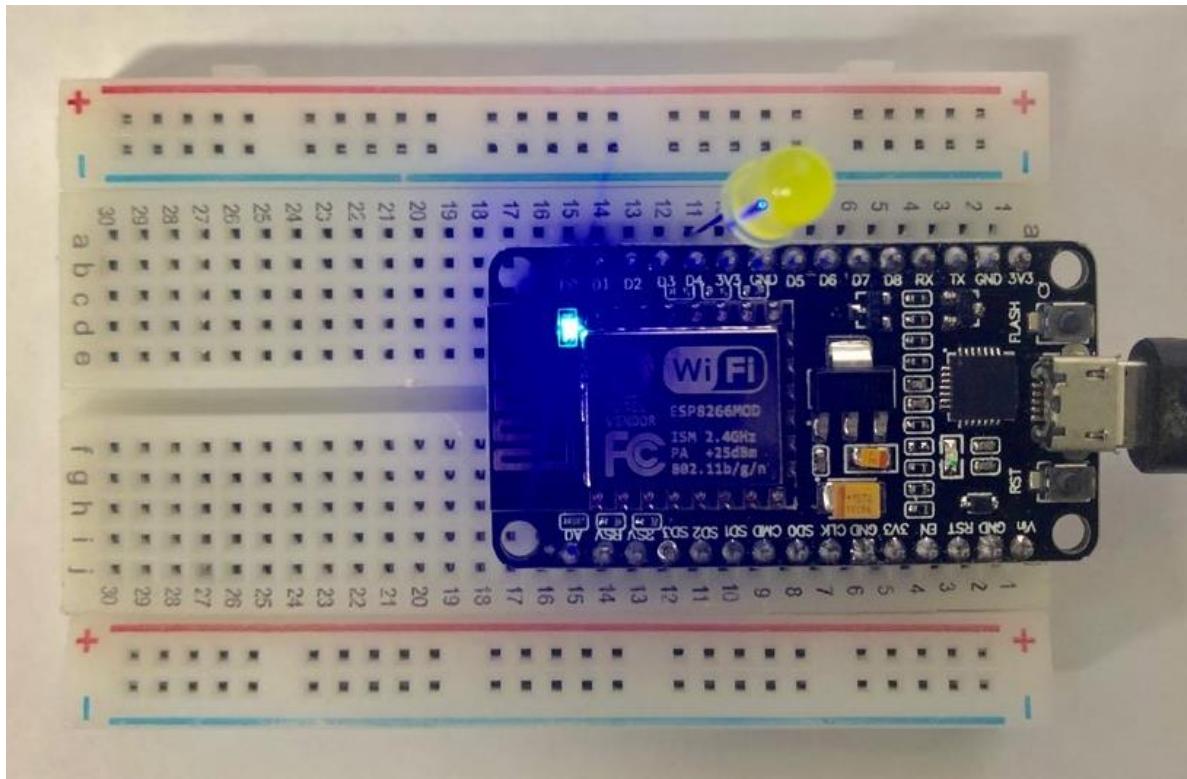
The status bar at the bottom right indicates "NodeMCU 1.0 (ESP-12E Module) on COM13".

Circuit Wiring

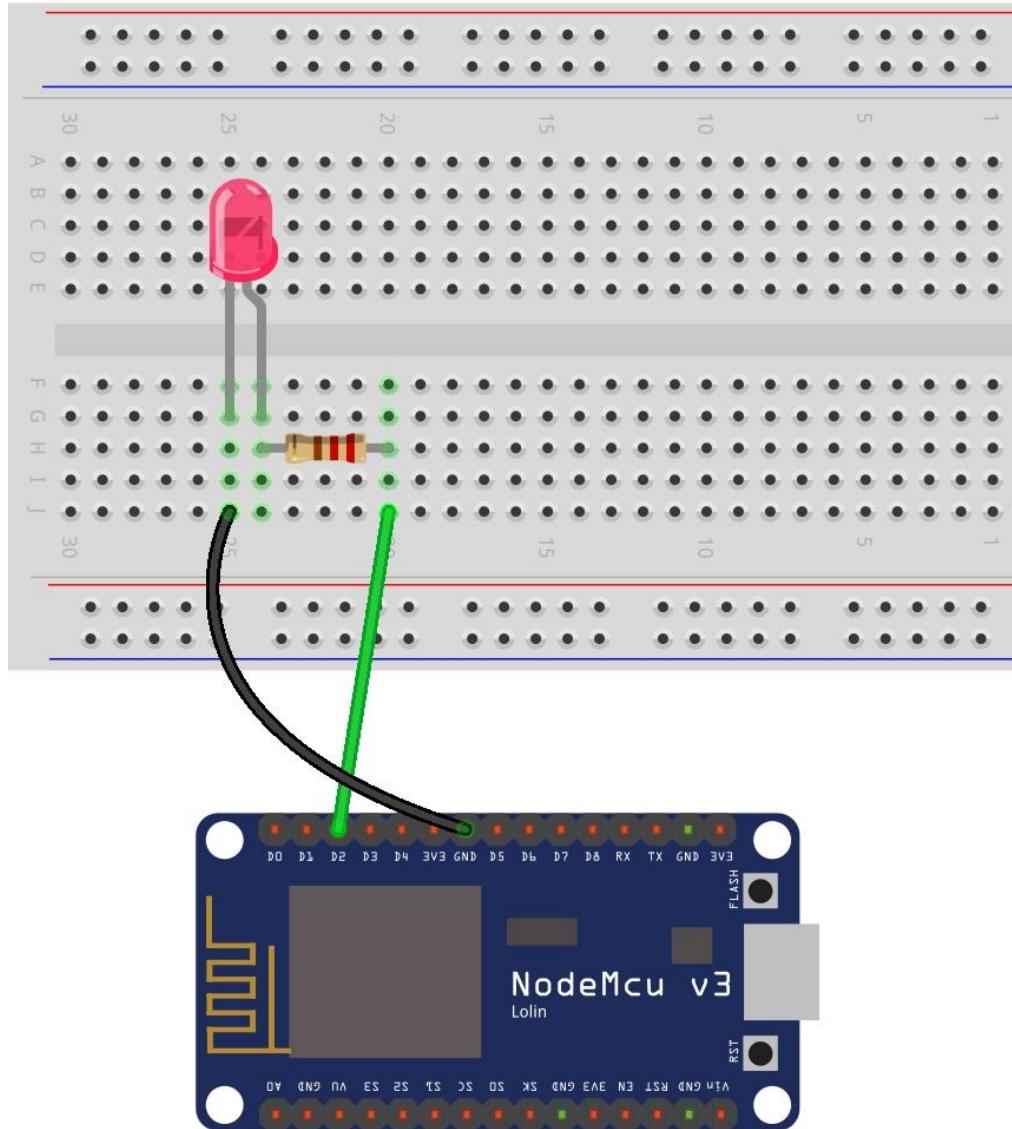
- LED_BUILTIN -> D4



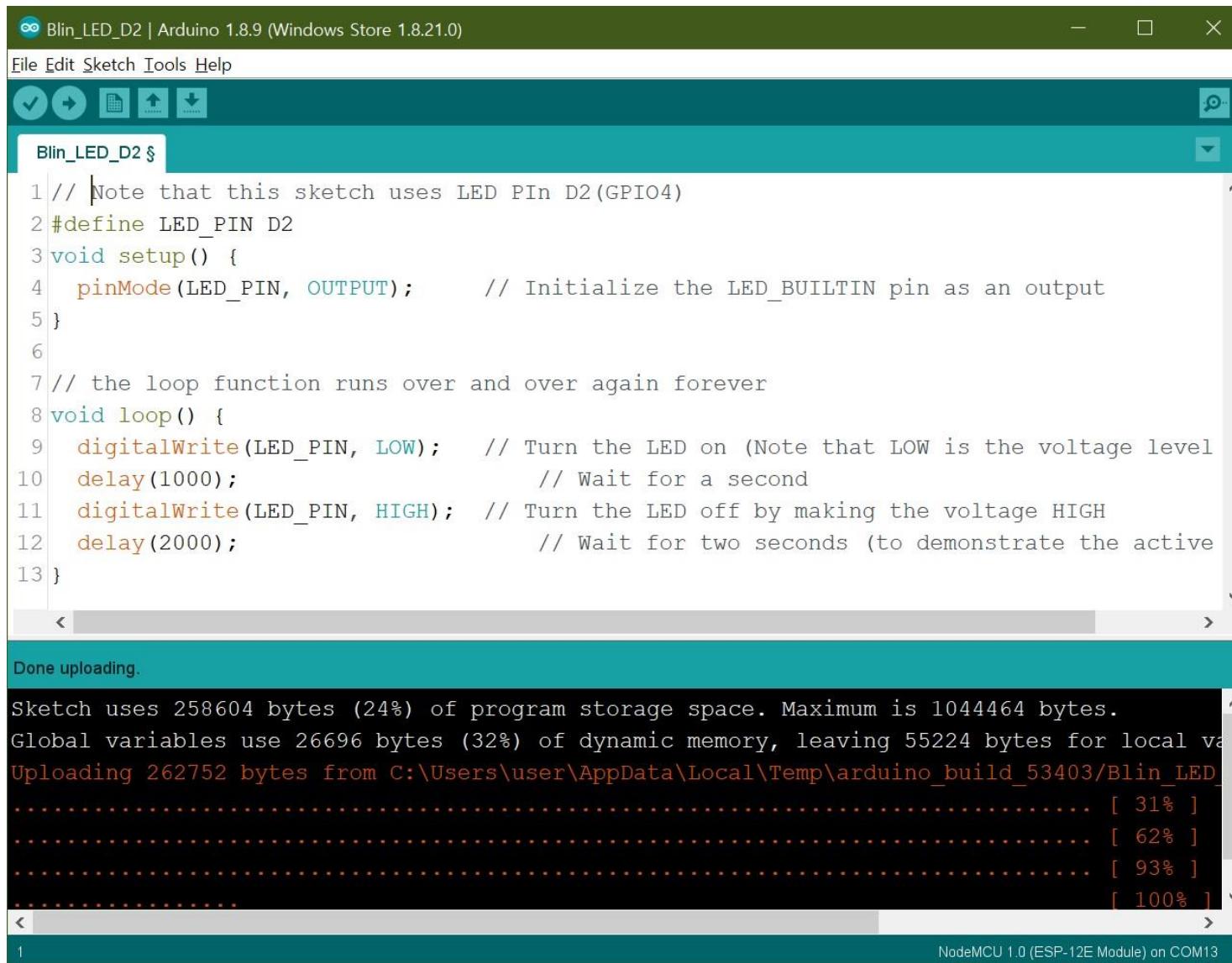
Operation of Internal LED : D4 pin



LED Control : D2 pin



LED Control : D2 pin

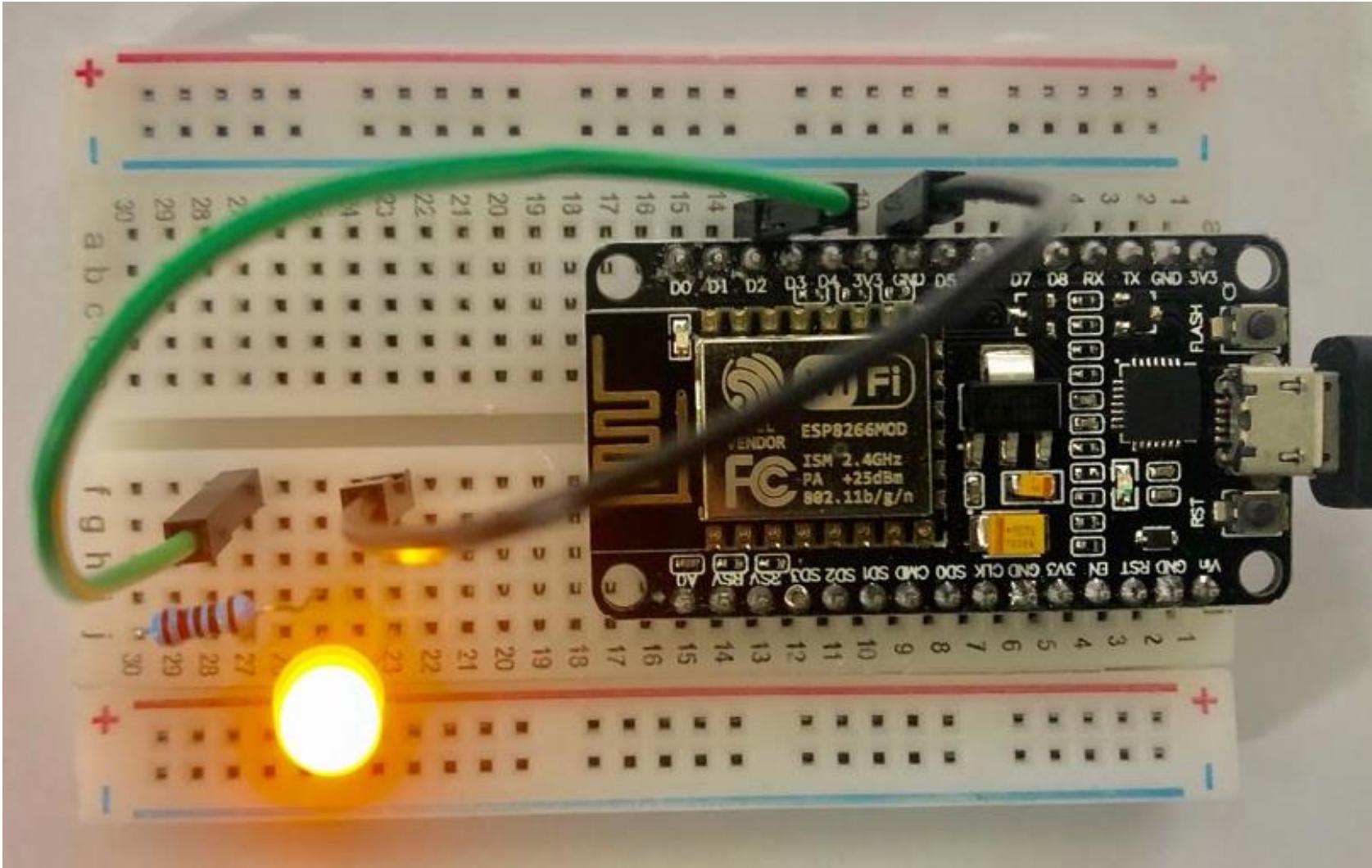


The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Blin_LED_D2 | Arduino 1.8.9 (Windows Store 1.8.21.0)
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Upload, and Download.
- Code Editor:** Displays the following C++ code:

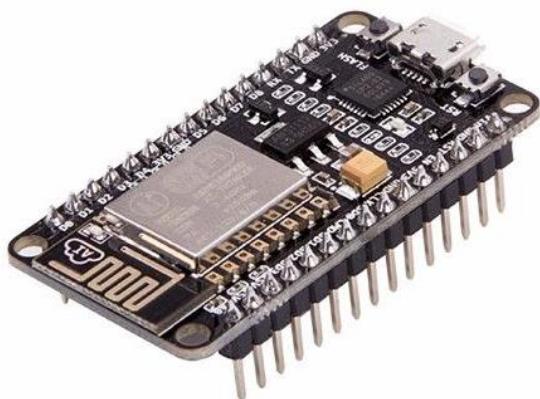
```
1 // Note that this sketch uses LED PIN D2(GPIO4)
2 #define LED_PIN D2
3 void setup() {
4     pinMode(LED_PIN, OUTPUT);      // Initialize the LED_BUILTIN pin as an output
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9     digitalWrite(LED_PIN, LOW);    // Turn the LED on (Note that LOW is the voltage level
10    delay(1000);                // Wait for a second
11    digitalWrite(LED_PIN, HIGH);  // Turn the LED off by making the voltage HIGH
12    delay(2000);                // Wait for two seconds (to demonstrate the active
13 }
```
- Status Bar:** Done uploading.
Sketch uses 258604 bytes (24%) of program storage space. Maximum is 1044464 bytes.
Global variables use 26696 bytes (32%) of dynamic memory, leaving 55224 bytes for local va
Uploading 262752 bytes from C:\Users\user\AppData\Local\Temp\arduino_build_53403\Blin_LED_
- Progress Bar:** Shows upload progress at 31%, 62%, 93%, and 100%.
- Bottom Status:** NodeMCU 1.0 (ESP-12E Module) on COM13

Operation of LED : D2 pin

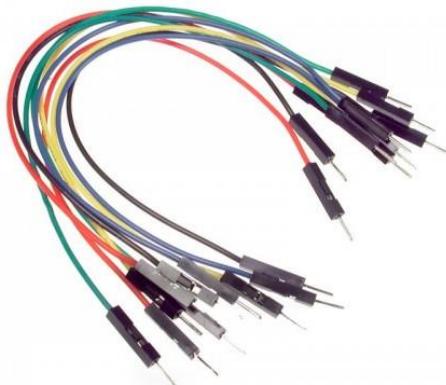


Reading Humidity and Temperature on NodeMCU

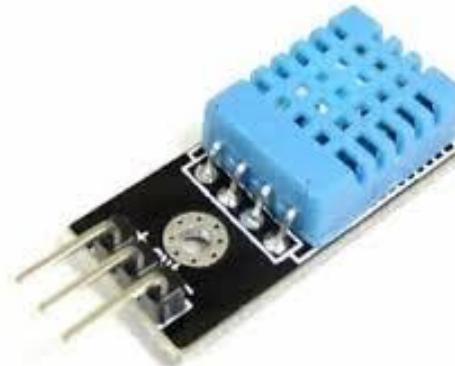
- Preparation materials



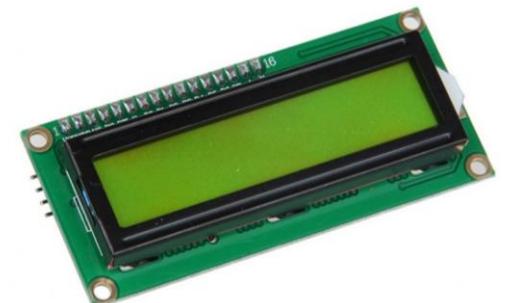
NodeMCU



Jump Cables

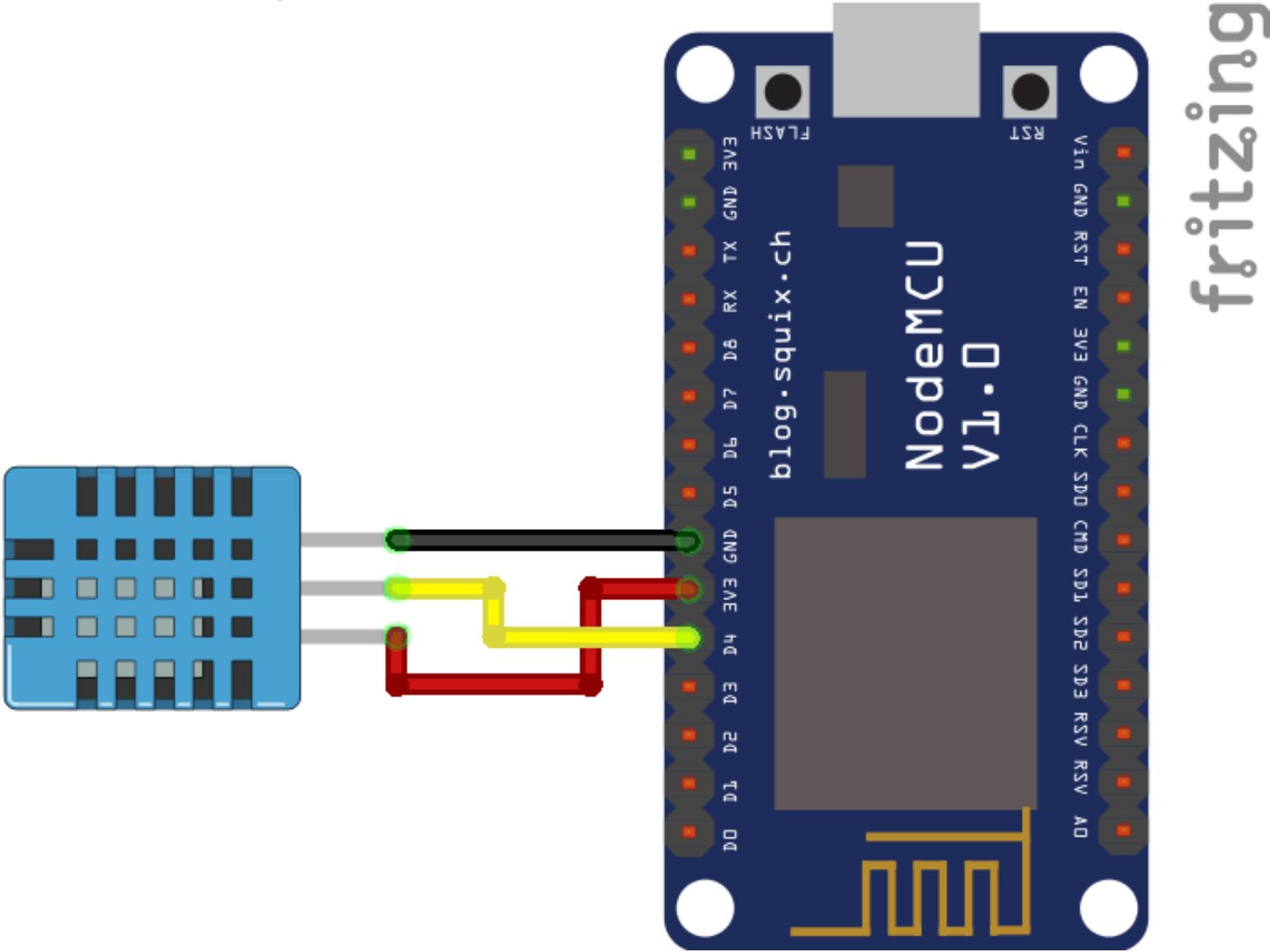


DHT11 Sensor



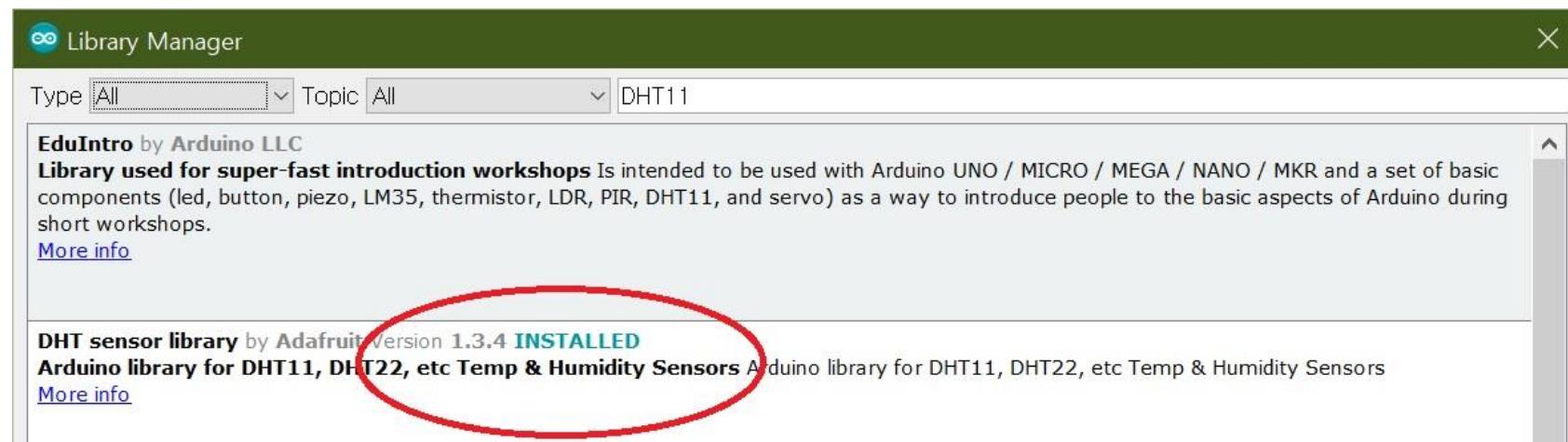
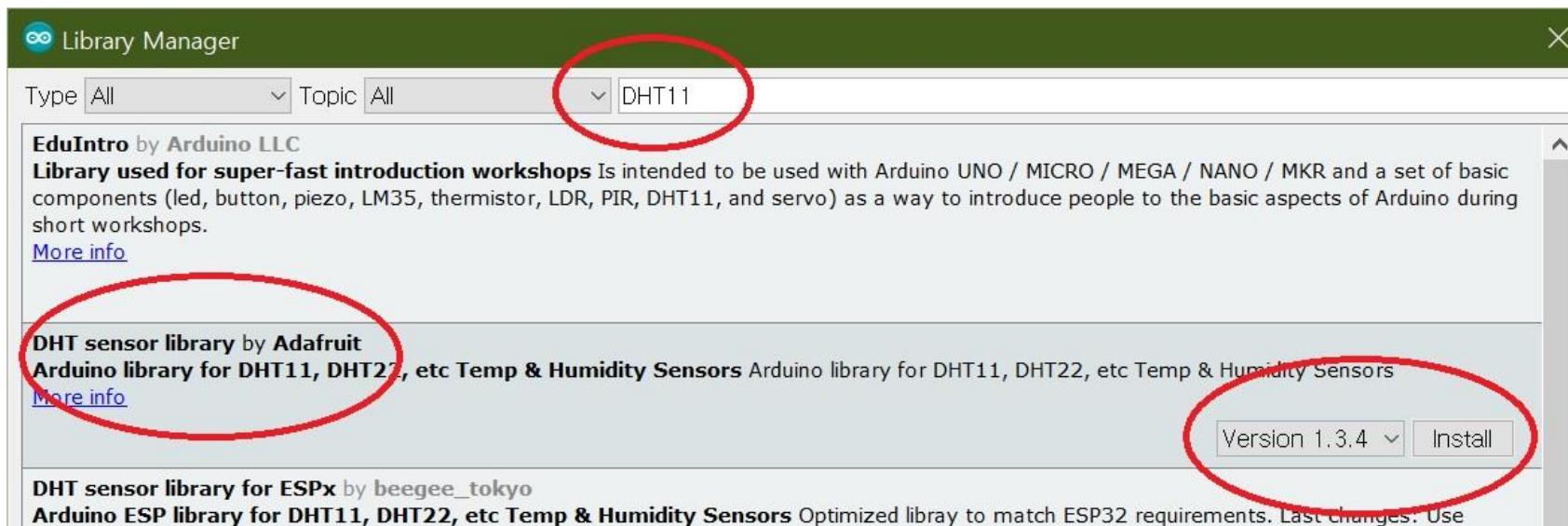
16x2 LCD Module

Circuit Wiring



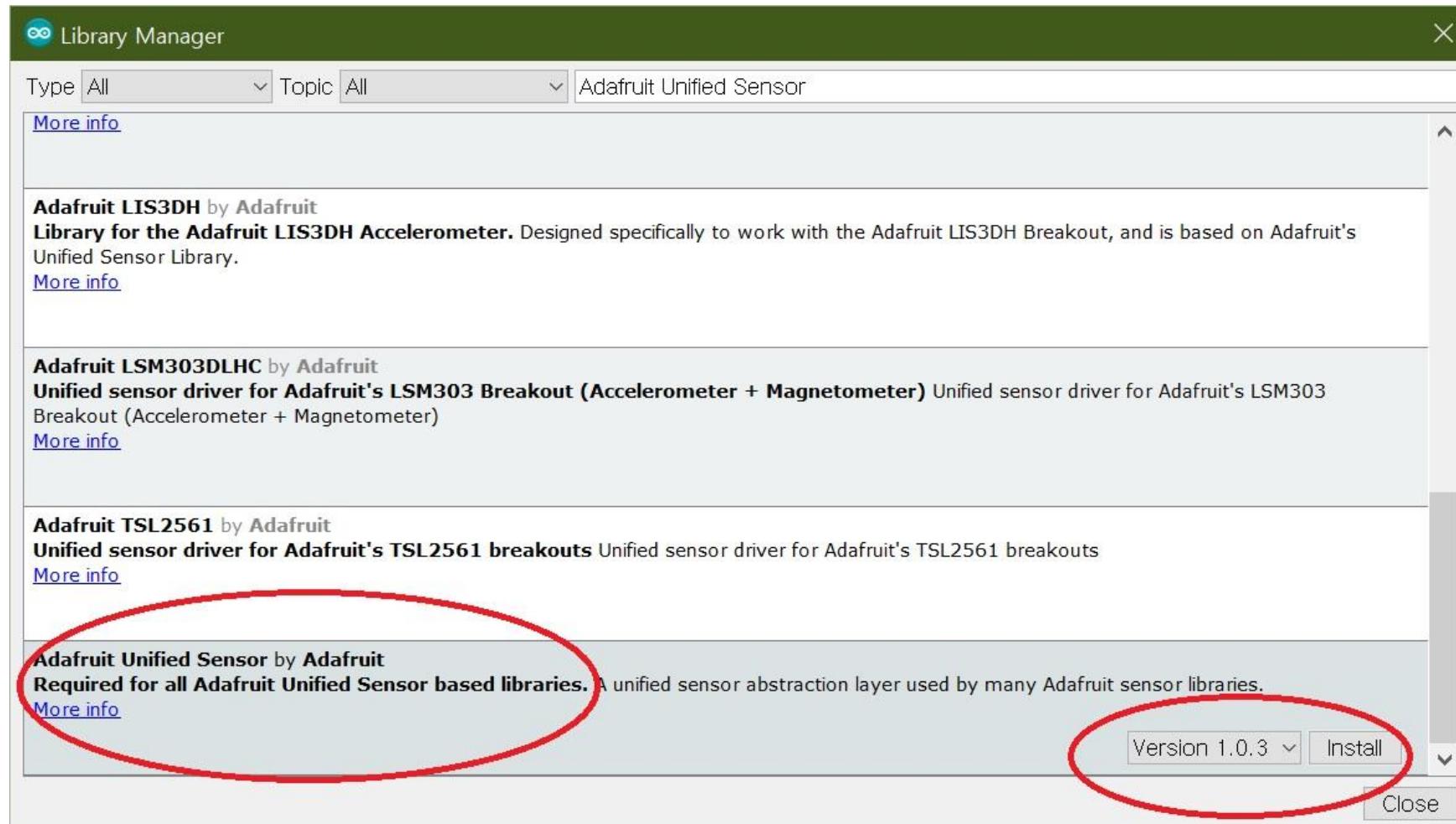
DHT Sensor Library(1)

- Menu -> Sketch -> Include -> Library -> Manage Libraries



Adafruit Unified Sensor Library(2)

- Menu -> Sketch -> Include -> Library -> Manage Libraries



Sketch

Temp_Hum01 | Arduino 1.8.9 (Windows Store 1.8.21.0)

File Edit Sketch Tools Help

Temp_Hum01

```
1 #include <DHT.h>
2 #define DHTPIN D4
3 #define DHTTYPE DHT11
4 DHT dht(DHTPIN, DHTTYPE);
5 void setup() {
6     // put your setup code here, to run once:
7     Serial.begin(115200);
8     dht.begin();
9 }
10
11 void loop() {
12     // put your main code here, to run repeatedly:
13     float temp = dht.readTemperature();
14     float humi = dht.readHumidity();
15     Serial.print("Temp: ");
16     Serial.print(temp);
17     Serial.print(" Humi: ");
18     Serial.println(humi);
19     delay(1000);
20 }
```

Done uploading.

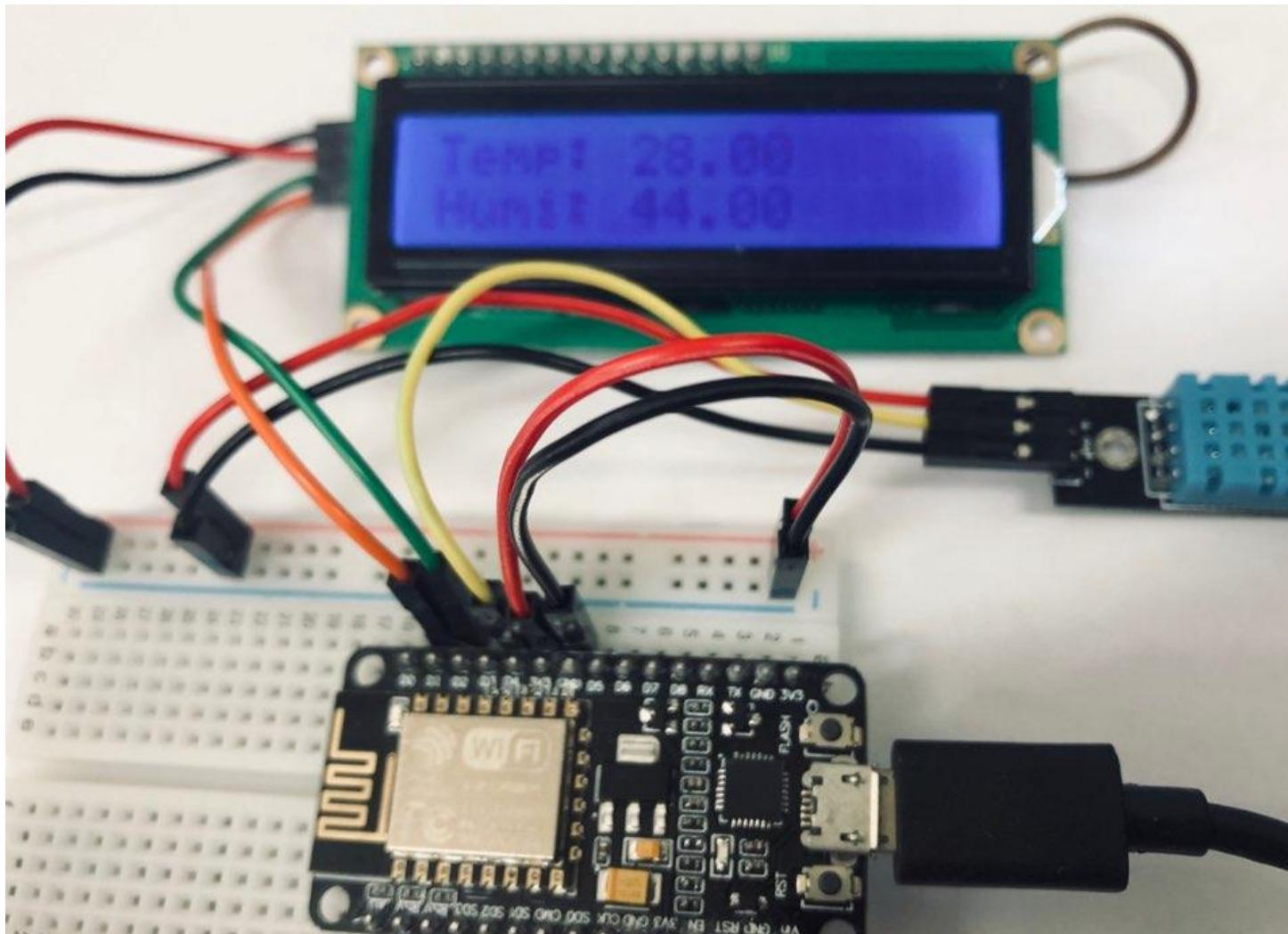
Sketch uses 265380 bytes (25%) of program storage space. Maximum is 1044464 bytes.

Global variables use 26832 bytes (32%) of dynamic memory, leaving 55088 bytes for local variables.

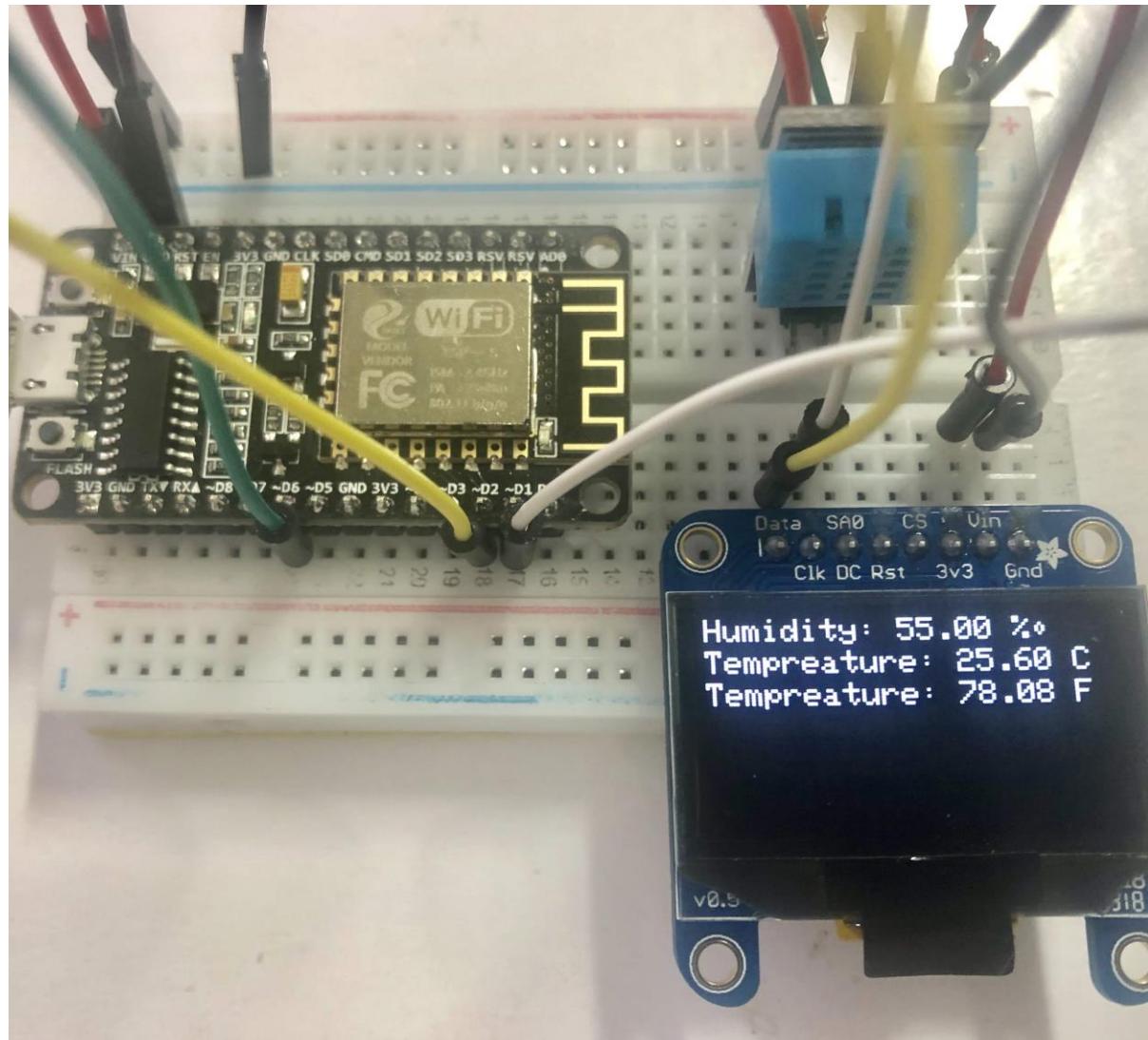
Uploading 269520 bytes from C:\Users\user\AppData\Local\Temp\arduino_build_824906\Temp_Hum01.ino to NodeMCU 1.0 (ESP-12E Module) on COM13

[30%]
[60%]
[90%]
[100%]

16x2 I2C LCD Display

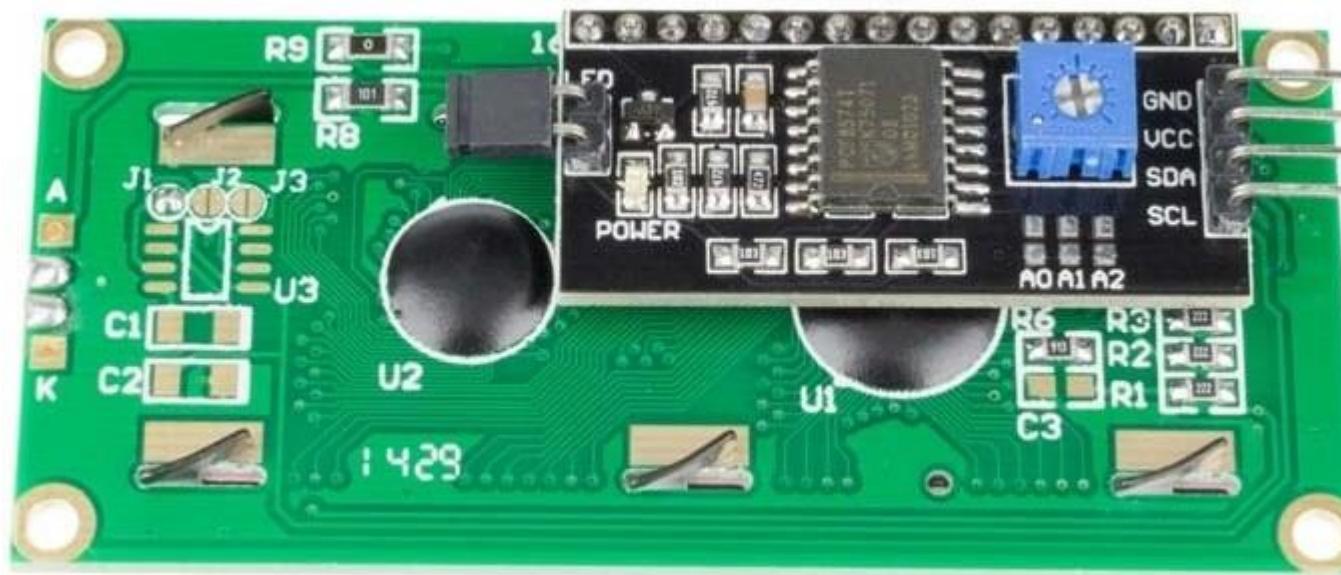


OLED Display

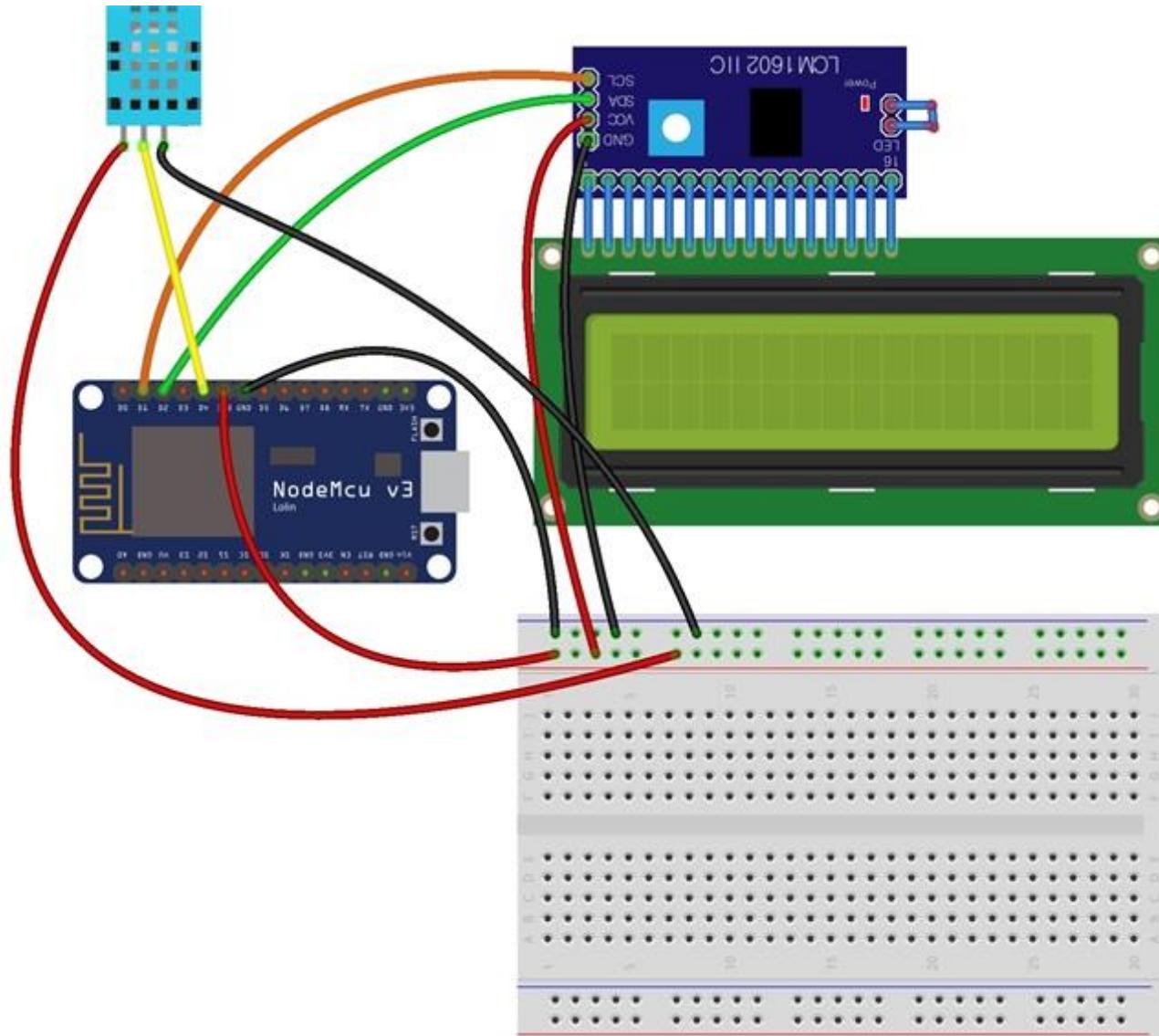


LCD I2C Pin connection

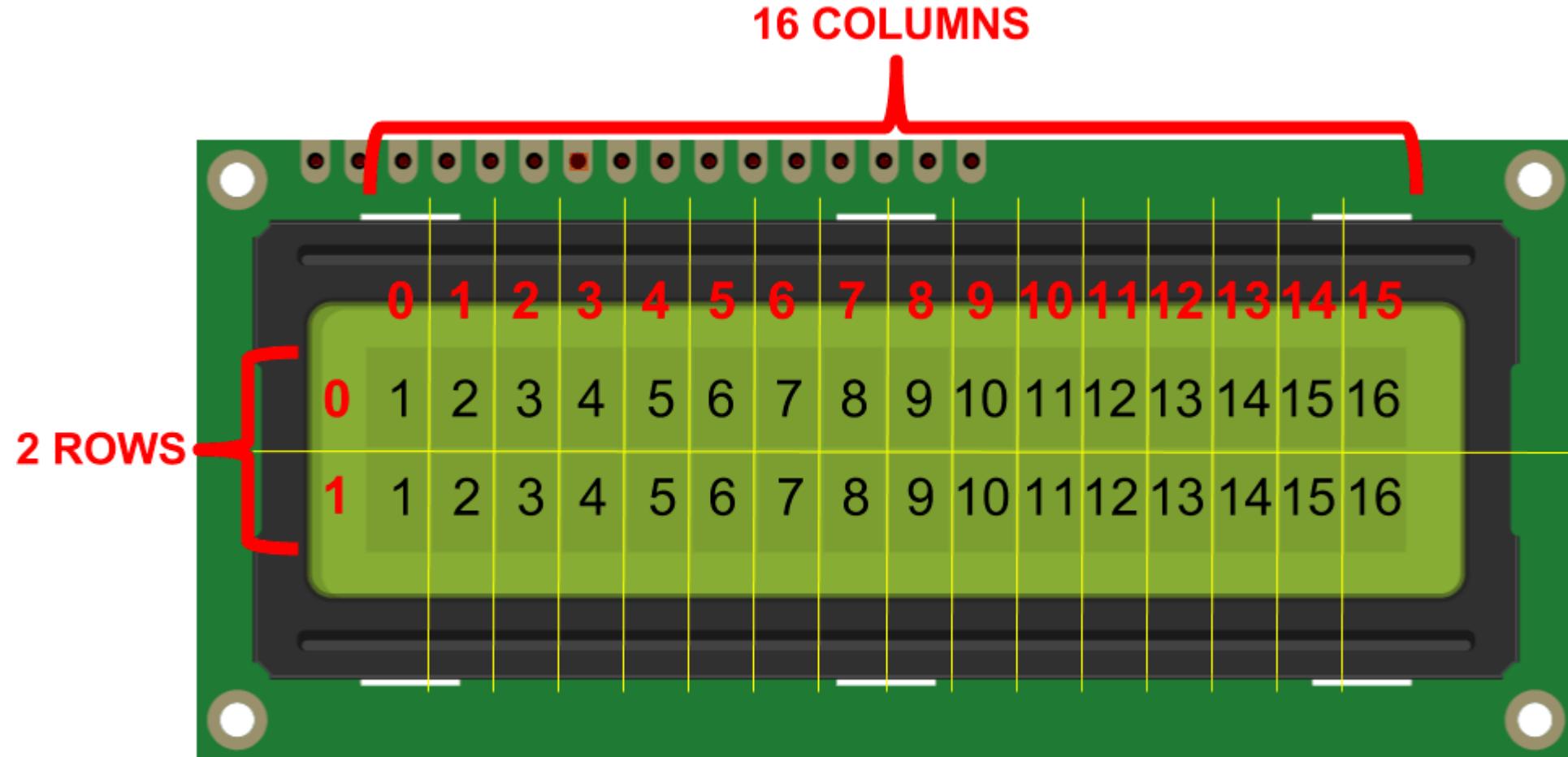
LCD	NodeMCU
GND	GND
VCC	3.3V
SDA	D2
SCL	D1



Circuit Wiring



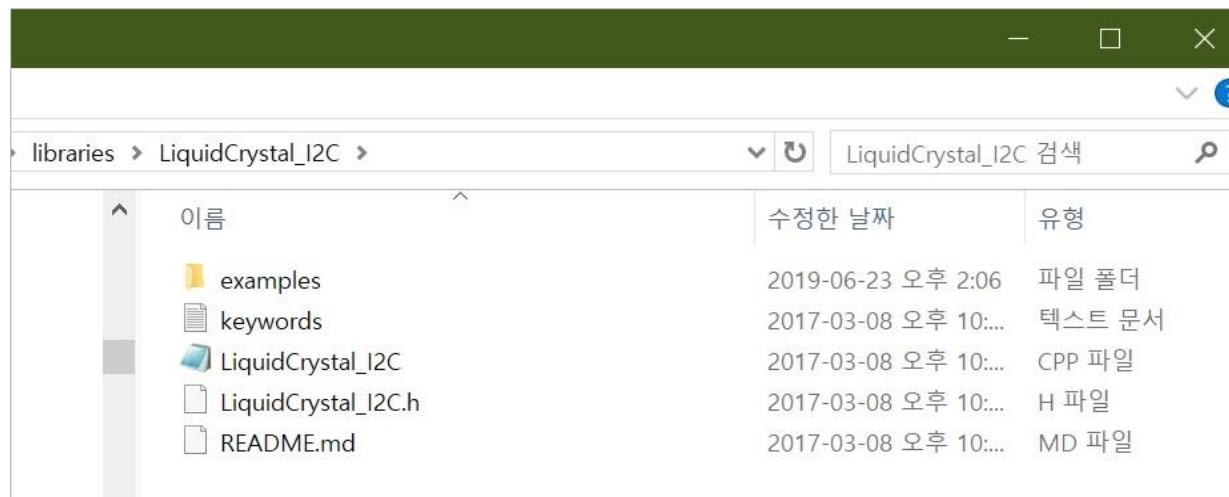
16x2 LCD Cursor Matrix



fritzing

LiquidCrystal-I2C-library

- Create a new folder called "LiquidCrystal_I2C" under the folder named "libraries" in your Arduino sketchbook folder.
- Create the folder "libraries" in case it does not exist yet. Place all the files in the "LiquidCrystal_I2C" folder.
 - <https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>



Sketch

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#define DHTPIN D4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
    Serial.begin(115200);
    dht.begin();

    // lcd init
    lcd.begin();
    // lcd back light on
    lcd.backlight();
    //lcd.print("Test");
}

}
```

```
void loop() {
    float temp = dht.readTemperature();
    float humi = dht.readHumidity();
    Serial.print("Temp: ");
    Serial.print(temp);
    Serial.print(" Humi: ");
    Serial.println(humi);

    // LCD Display
    // LCD Cursor change to (0,0)
    lcd.setCursor(0,0);
    lcd.print("Temp: ");
    lcd.print(temp);

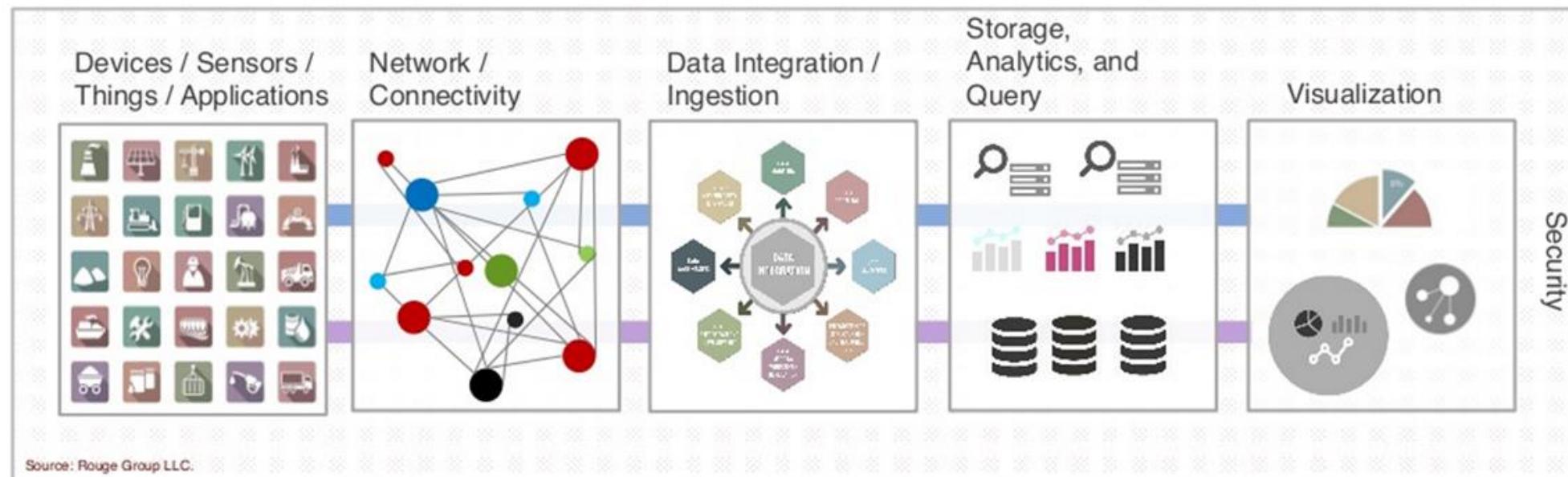
    // LCD Cursor change to (0,1)
    lcd.setCursor(0,1);
    lcd.print("Humi: ");
    lcd.print(humi);

    delay(1000);
}
```

IoT Cloud Platform Landscape

- <https://www.postscapes.com/internet-of-things-platforms/>

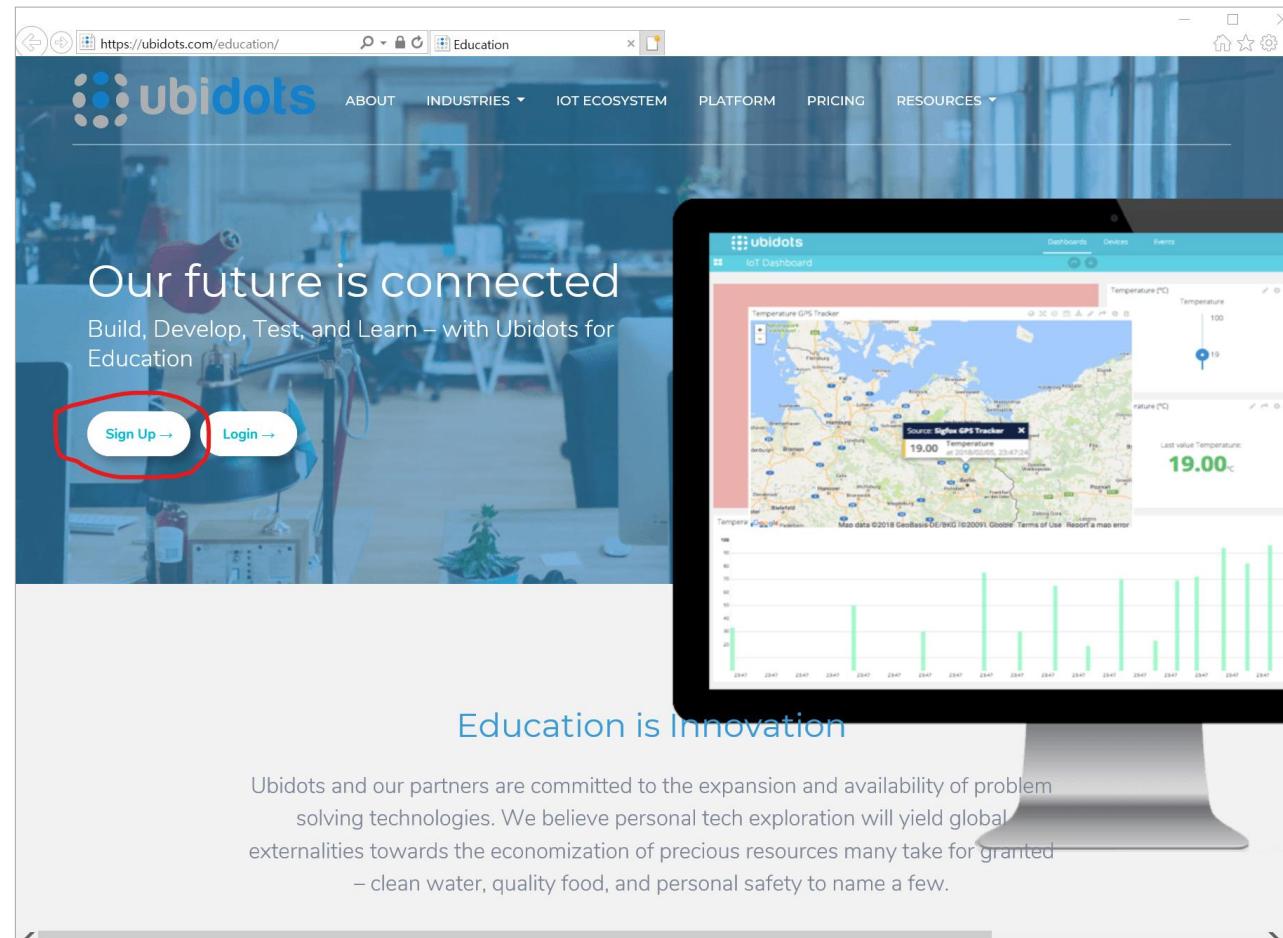
IoT Platform Reference Architecture



Lots of technologies, vendors, and approaches for each component

Ubidots : IoT Cloud Platform

- Ubidots Education : <https://ubidots.com/education/>
 - Sign Up ->



IoT and Cloud tools to build your business

Merge the physical and digital worlds into deployable, end-user ready IoT Applications that deliver insights and solve problems.

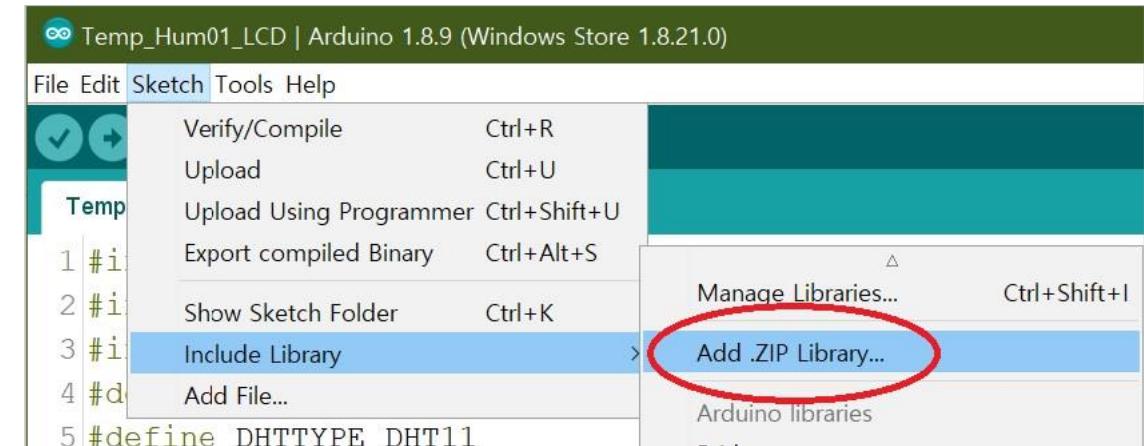
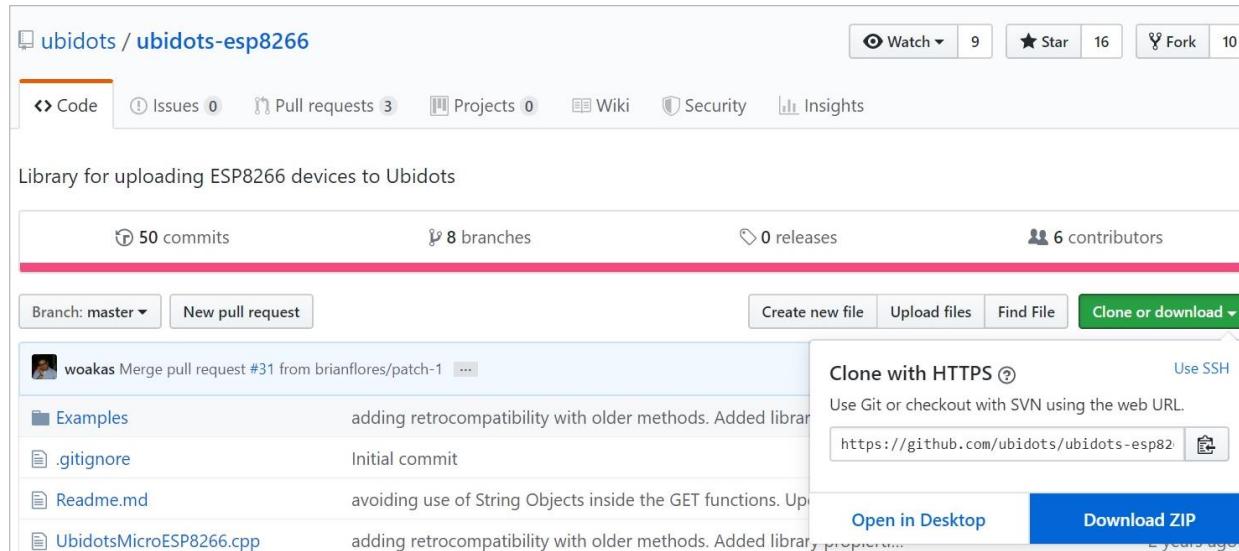


Lab 01 : IoT based Real Time Humidity/ Temperature Monitor



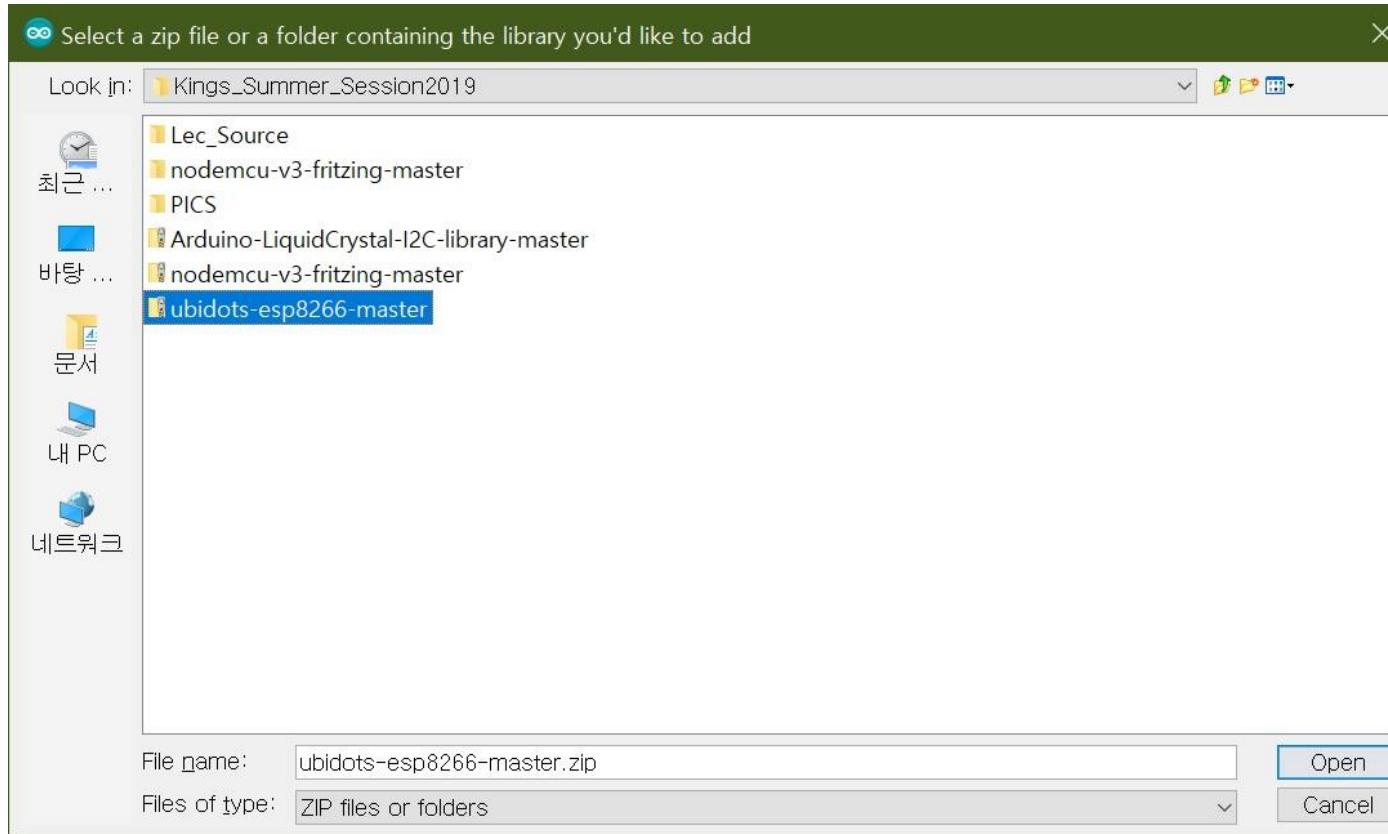
Ubidots Library(1)

- Download the UbidotsMicroESP8266 library
 - <https://github.com/ubidots/ubidots-esp8266>
- Select Download ZIP file

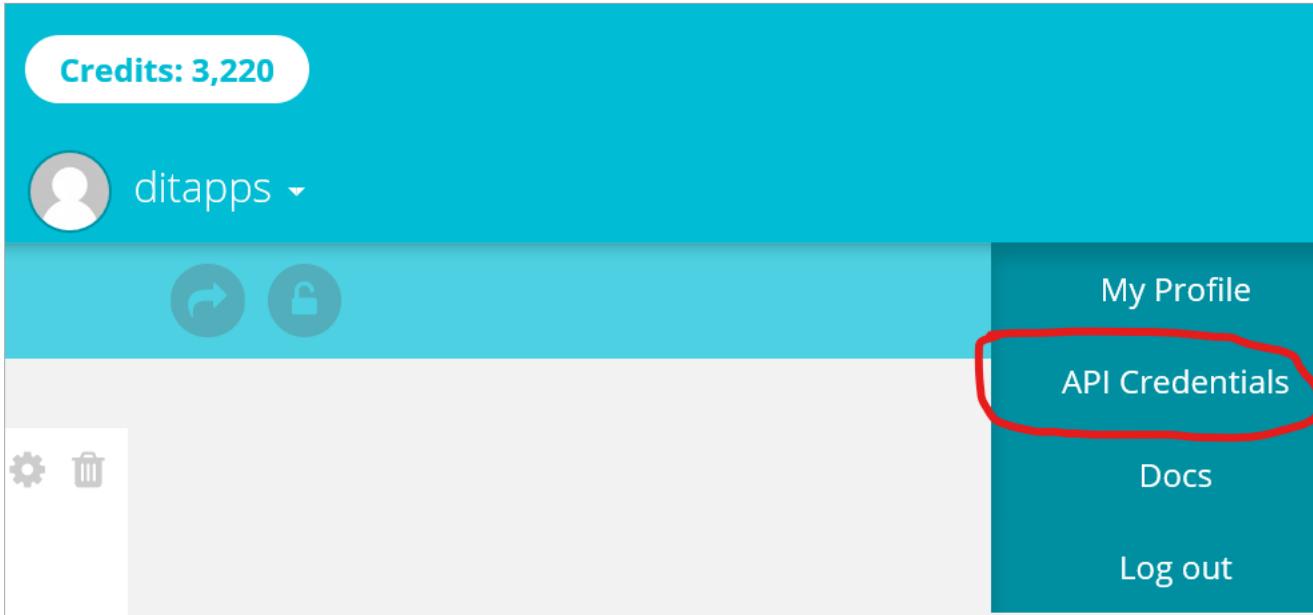


Ubidots Library(2)

- Click on Sketch -> Include Library -> Add .ZIP library
- Select the .ZIP file of UbidotsMicro8266 and the “Accept”
- Close the Arduino IDE and open it again.



Copy your Ubidots Token



A screenshot of the Ubidots Tokens section. It shows two tokens: "A1E-87fb0616fe8fea28ee3fa862b73831b88ece" and "A1E-NPfOekvBf8qTvOs3RWj3TnPrMAKC". The second token is highlighted with a red circle. Below the tokens, there's a "More" button.

Token
A1E-87fb0616fe8fea28ee3fa862b73831b88ece
A1E-NPfOekvBf8qTvOs3RWj3TnPrMAKC

Sending DHT11 Sensor Values to Ubidots Cloud

- Input code, upload and execute

```
#include <DHT.h>
#include "UbidotsMicroESP8266.h"
#define TOKEN "xxxxx" // put here your Ubidots token
#define WIFISSID "melon"
#define PASSWORD "deitcs3217"
#define DHTPIN D4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

Ubidots client(TOKEN);

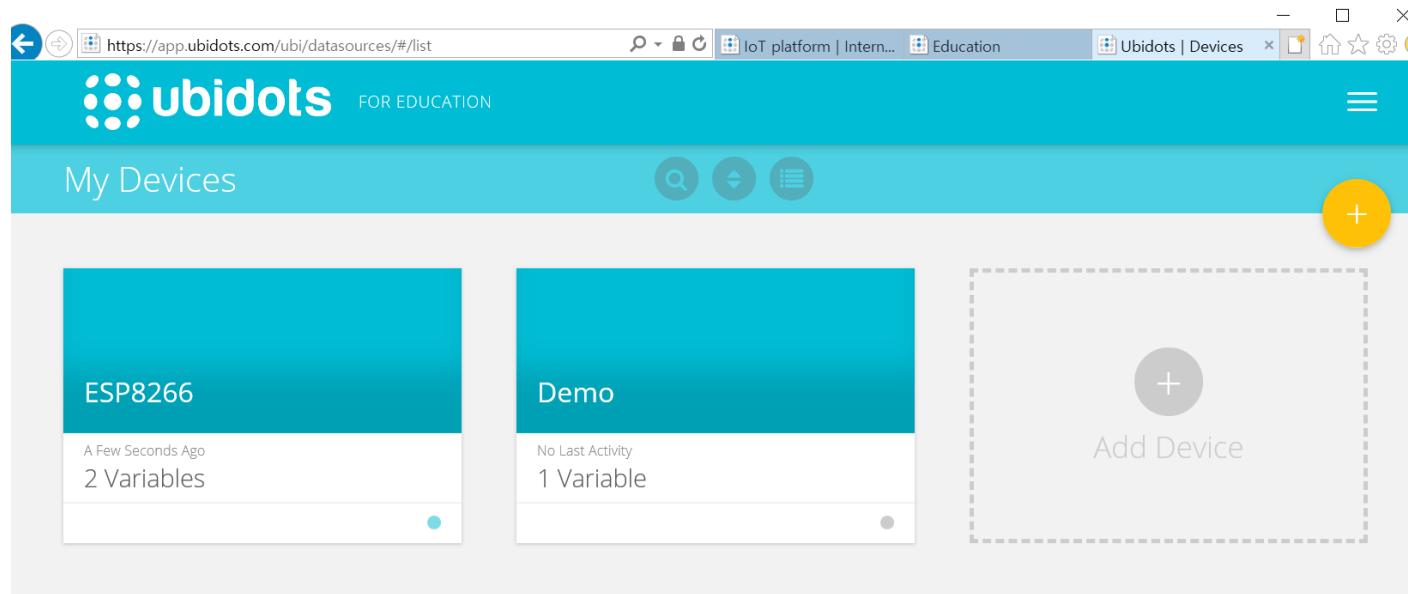
void setup() {
    Serial.begin(115200);
    dht.begin();
    client.wifiConnection(WIFISSID, PASSWORD);
}
```

```
void loop() {
    float temp = dht.readTemperature();
    float humi = dht.readHumidity();
    Serial.print("Temp: ");
    Serial.print(temp);
    Serial.print(" Humi: ");
    Serial.println(humi);

    // Temperature, Humidity is variable labels
    // of Ubidots
    client.add("Temperature", temp);
    client.add("Humidity", humi);
    client.sendAll(true);
    delay(1000);
}
```

Devices Creation : ESP8266

- Wait for the device(ESP8266) to appear



Devices Status(1)

The screenshot shows the Ubidots IoT platform interface for a device named "ESP8266".

Header: The top navigation bar includes links for Dashboards, Devices (which is the active tab), and Events. It also displays "Credits: 3,134" and a user profile icon.

Main Content: The main area features a map of South Korea with a blue callout box highlighting the location of the "ESP8266" device. Below the map, there are two yellow cards displaying real-time sensor data:

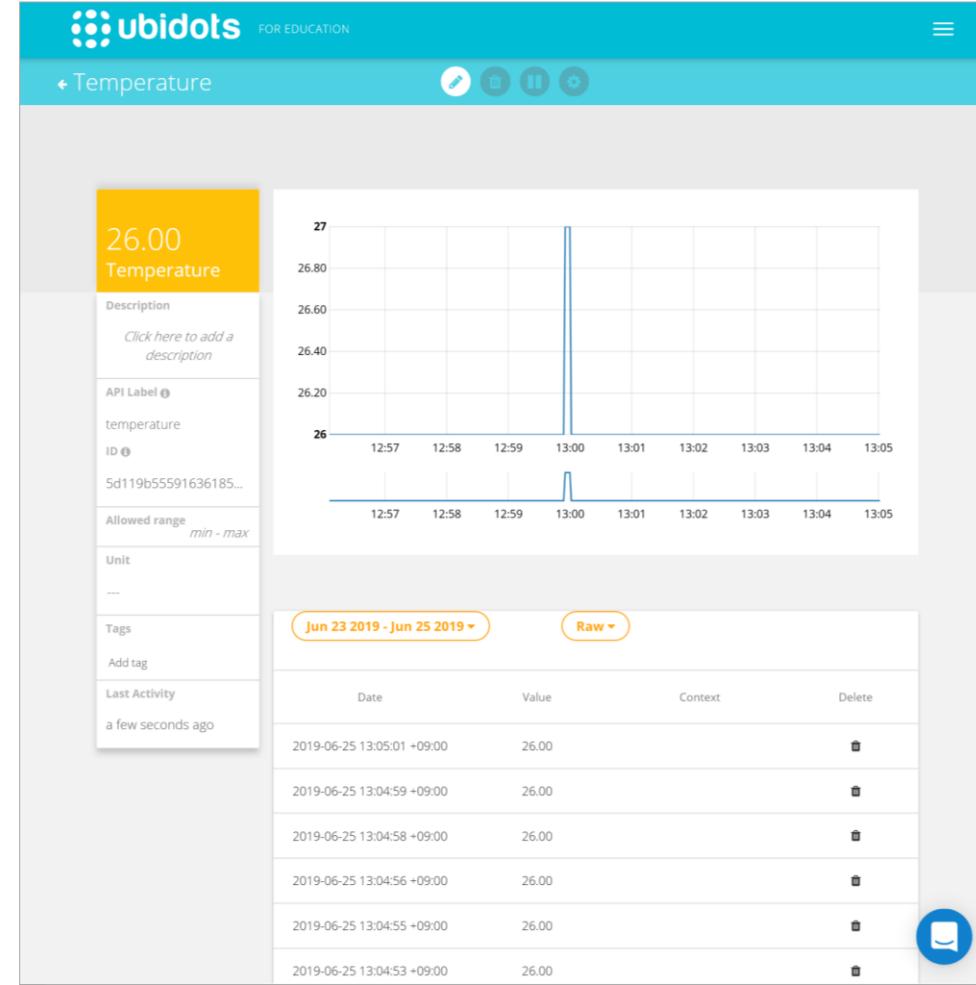
- Humidity:** Value 48.00, last activity "a day ago".
- Temperature:** Value 27.00, last activity "a day ago".

Left Sidebar: A sidebar on the left provides details about the device:

- Description:** "Click here to add a description".
- API Label:** cc50e300b96e
- ID:** 5d0f46965916361856c33e...
- Tags:** "Add tag".
- Last Activity:** "a day ago".

Bottom Right: A dashed box labeled "Add Variable" with a plus sign, and navigation arrows for "Show 50 items".

Devices Status(2)



Dashboards Creation : Add new widget

The image shows two screenshots of the Ubidots dashboard interface. The top screenshot displays the main dashboard page with a teal header containing the Ubidots logo and 'FOR EDUCATION'. Below the header is a navigation bar with a 'Dashboard' button, a refresh icon, a lock icon, and a yellow '+' button. A large central box contains the text 'Add widgets to see your data in real-time' and a sub-instruction: 'Click on the + icon to add a Widget. If you haven't sent any data to Ubidots, check out our [API Docs](#) to get started.' The bottom screenshot shows a modal window titled 'New Widget' with a teal header. Inside, it asks 'How would you like to see your data?' and lists six widget types with corresponding icons: Chart (line graph), Metric (number with 'LAST VALUE' and '27'), Map (location pin), Table (grid), Indicator (circle with dot), Control (two vertical bars), and HTML Canvas (two arrows). Each icon has its name written below it.

ubidots FOR EDUCATION

Dashboard

Add widgets to see your data in real-time

Click on the + icon to add a Widget. If you haven't sent any data to Ubidots, check out our [API Docs](#) to get started.

New Widget

How would you like to see your data?

Widget Type	Icon Description
Chart	Line graph icon
Metric	Number with 'LAST VALUE' and '27' icon
Map	Location pin icon
Table	Grid icon
Indicator	Circle with dot icon
Control	Two vertical bars icon
HTML Canvas	Two arrows icon

Dashboard Creation : Add Variable

Select a type of widget:



Line chart Double axis Scatter plot Histogram Bars

List of Variables to control

 Add Variable





Dashboard Creation: Select a Device, a Variable

Add a Variable to visualize in your Widget

Widget creation

Select a Device

ESP8266 Demo

Filter Device

Select a Variable

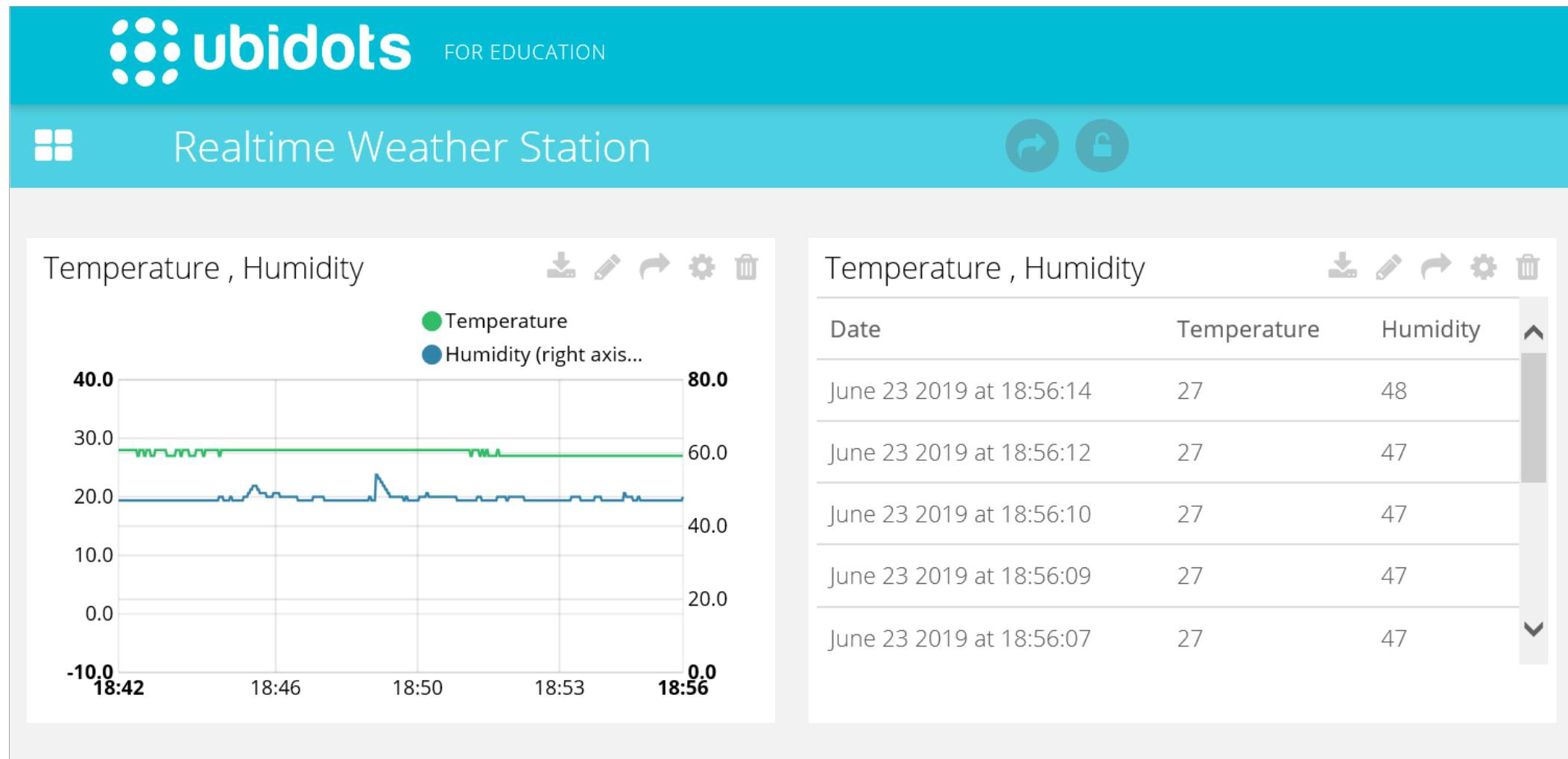
 

Humidity Temperature

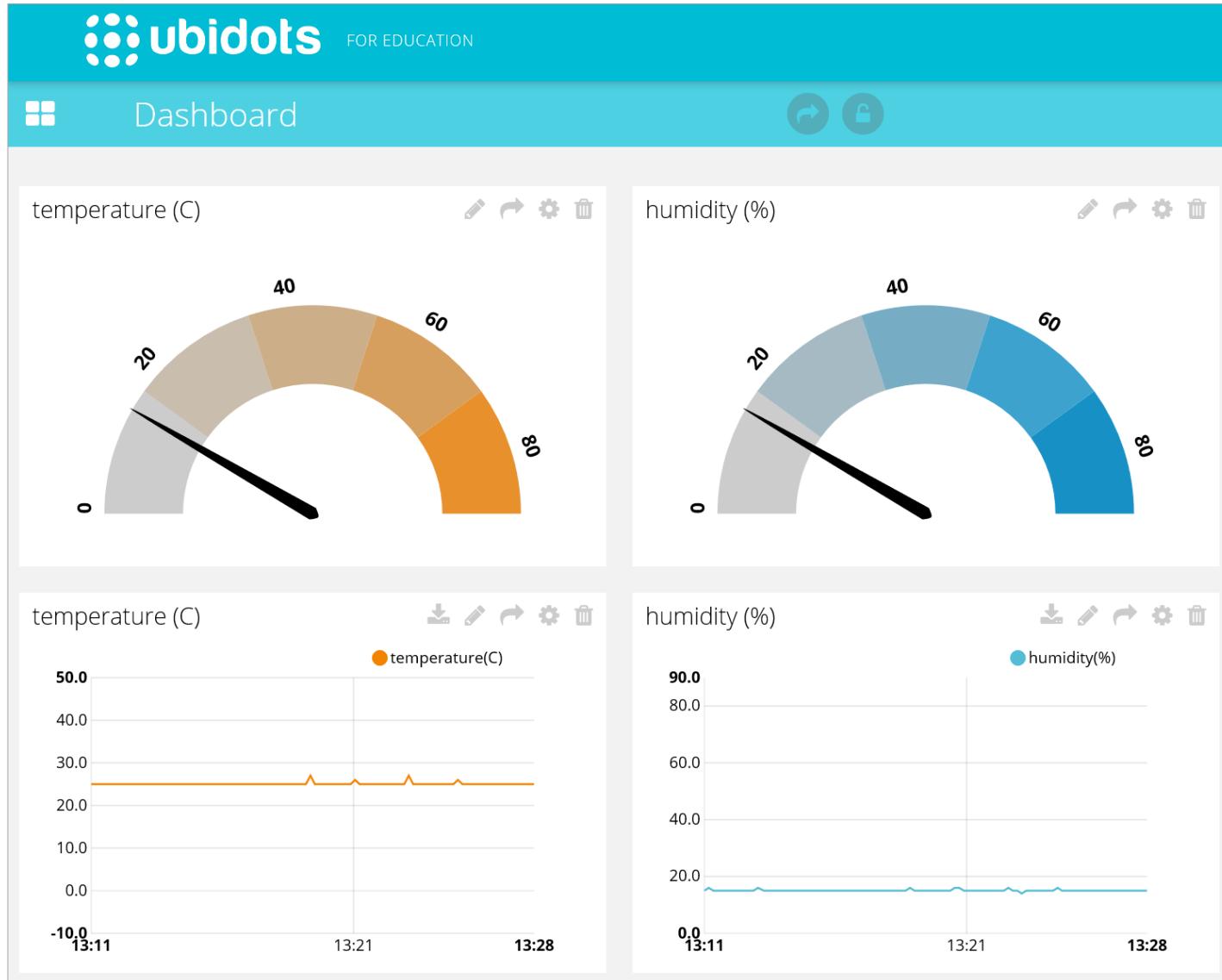
Filter Variable

Add Variable

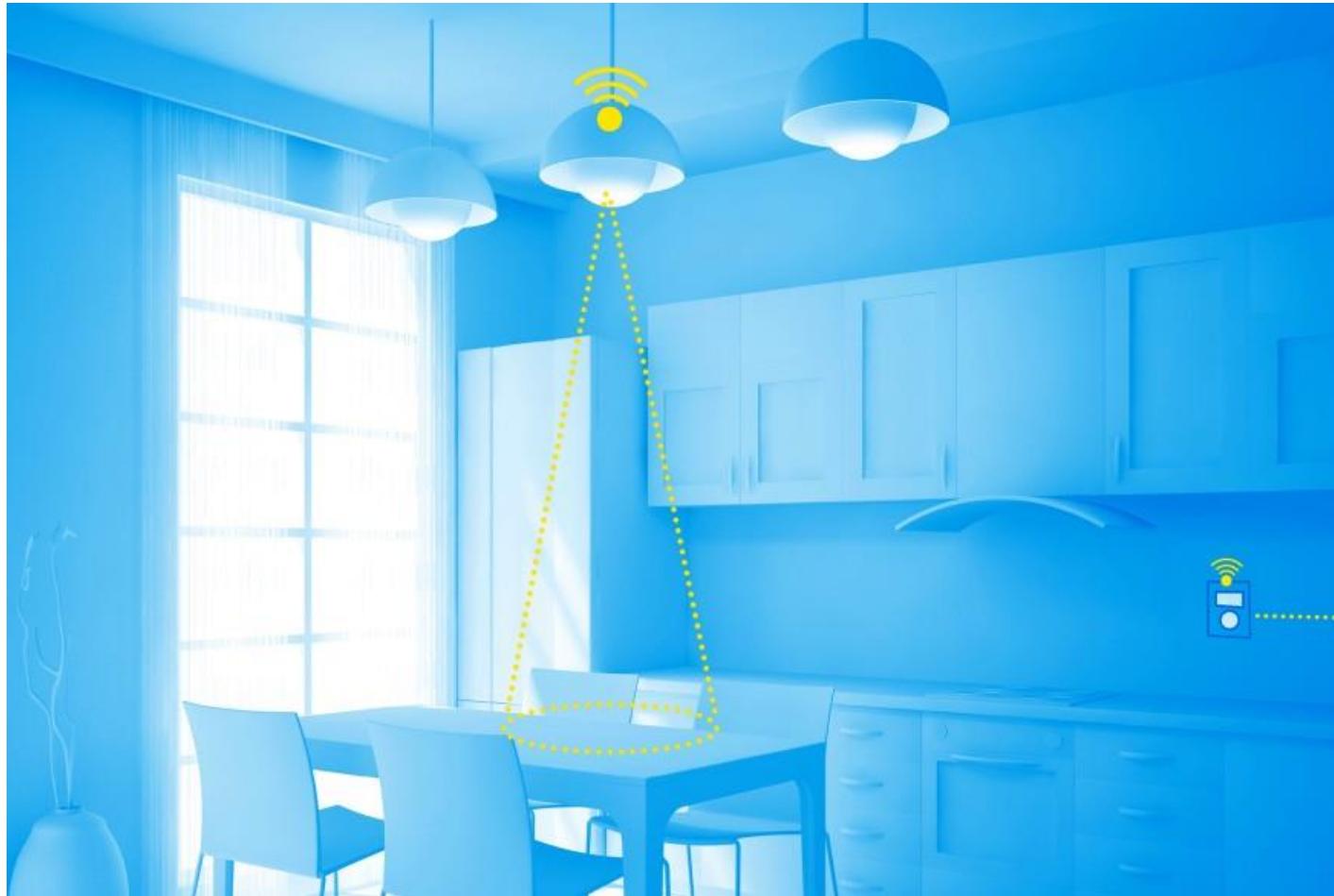
Dashboards : Chart, Table -> Historical Data



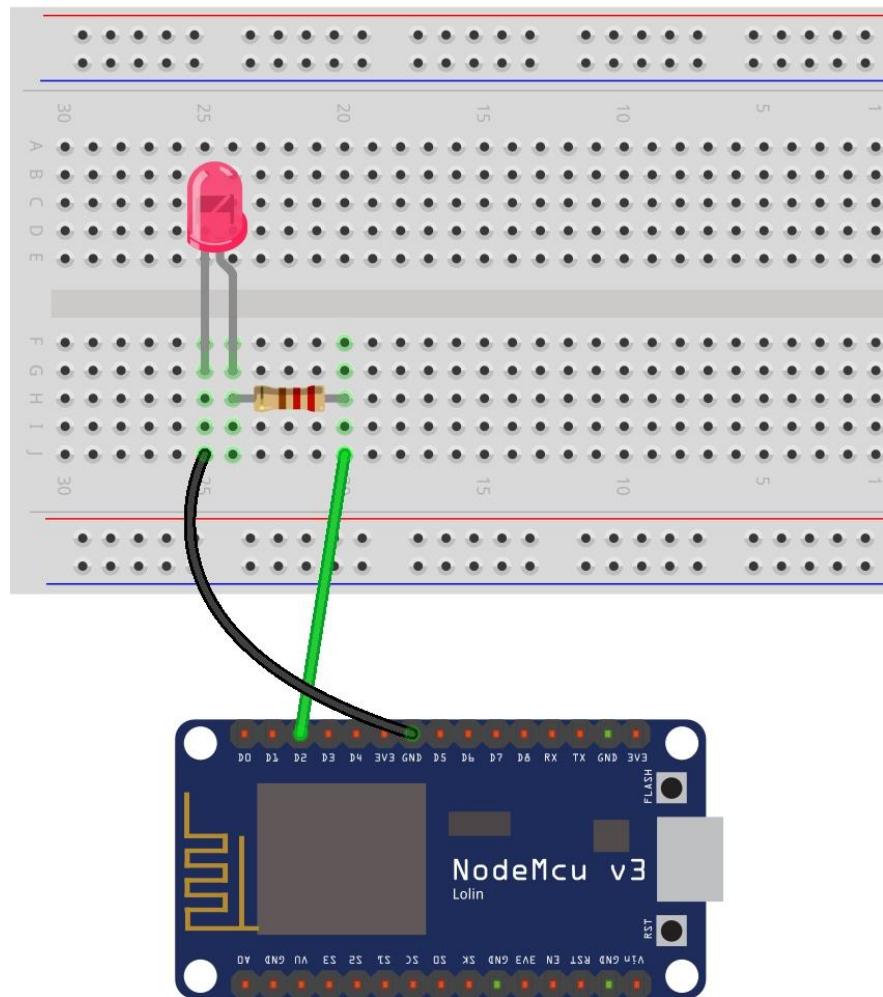
Dashboards : Indicator -> Gauge



Lab 02 : Smart Switch Controller



Circuit Wiring



Input Code, upload & Execute

```
#include "UbidotsMicroESP8266.h"

#define DEVICE  "led-control" // your Ubidots device label
#define VARIABLE  "led-value" // your Ubidots variable
label
// Put here your Ubidots TOKEN
#define TOKEN  "A1E-btnEp0we6d27oA5sRLjlexPvxL02ro"
#define WIFISSID "melon"
#define PASSWORD "deitcs3217"
#define LED_PIN D2

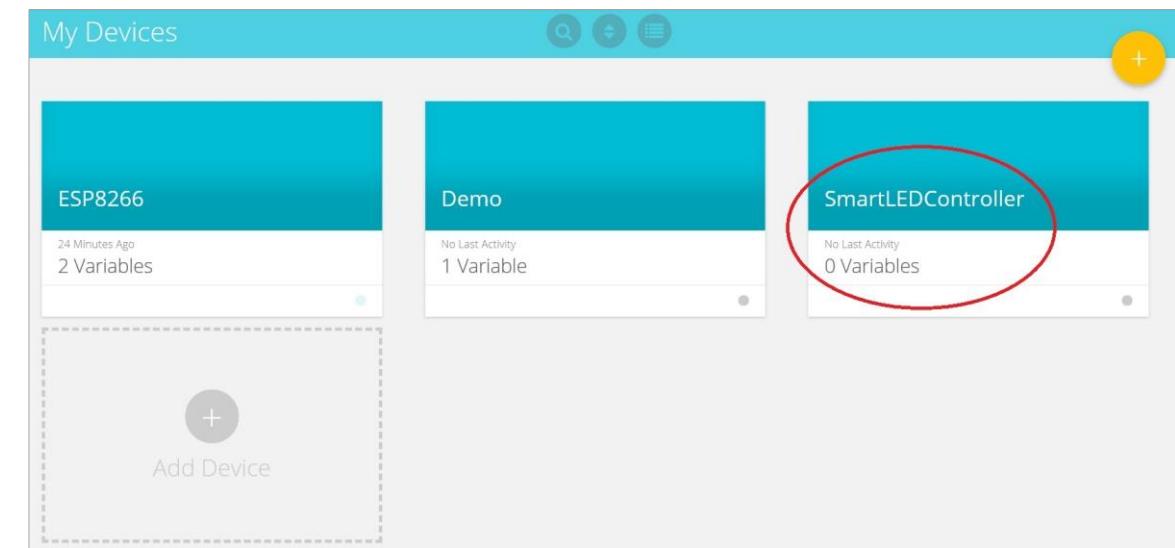
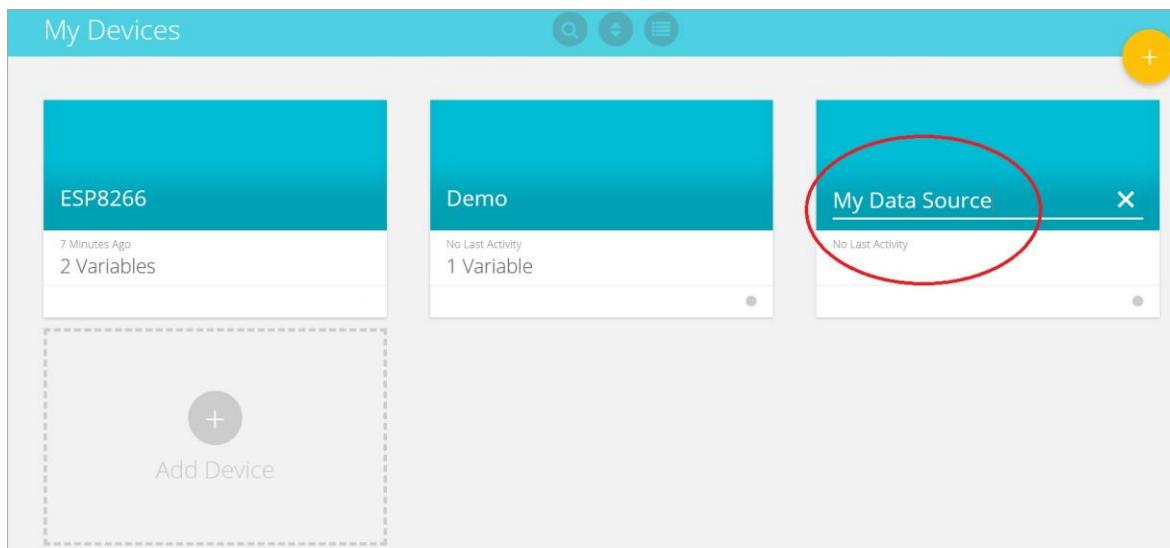
Ubidots client(TOKEN);

void setup() {
    Serial.begin(115200);
    client.wifiConnection(WIFISSID, PASSWORD);
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, 1);
}
```

```
void loop() {
    float value = client.getValueWithDevice(DEVICE, VARIABLE);
    if (value != ERROR_VALUE) {
        Serial.print("value obtained: ");
        Serial.println(value);
        if(value == 1) digitalWrite(LED_PIN, HIGH);
        else digitalWrite(LED_PIN, LOW);
    }else{
        Serial.println("Error getting value");
    }
    delay(1000);
}
```

Device Creation

- MyDevices -> My Data Source -> SmartLEDController



Setting Variables

SmartLEDController

The screenshot shows the SmartLEDController interface. At the top is a map of South Korea with various locations labeled. Below the map is a sidebar with the following sections:

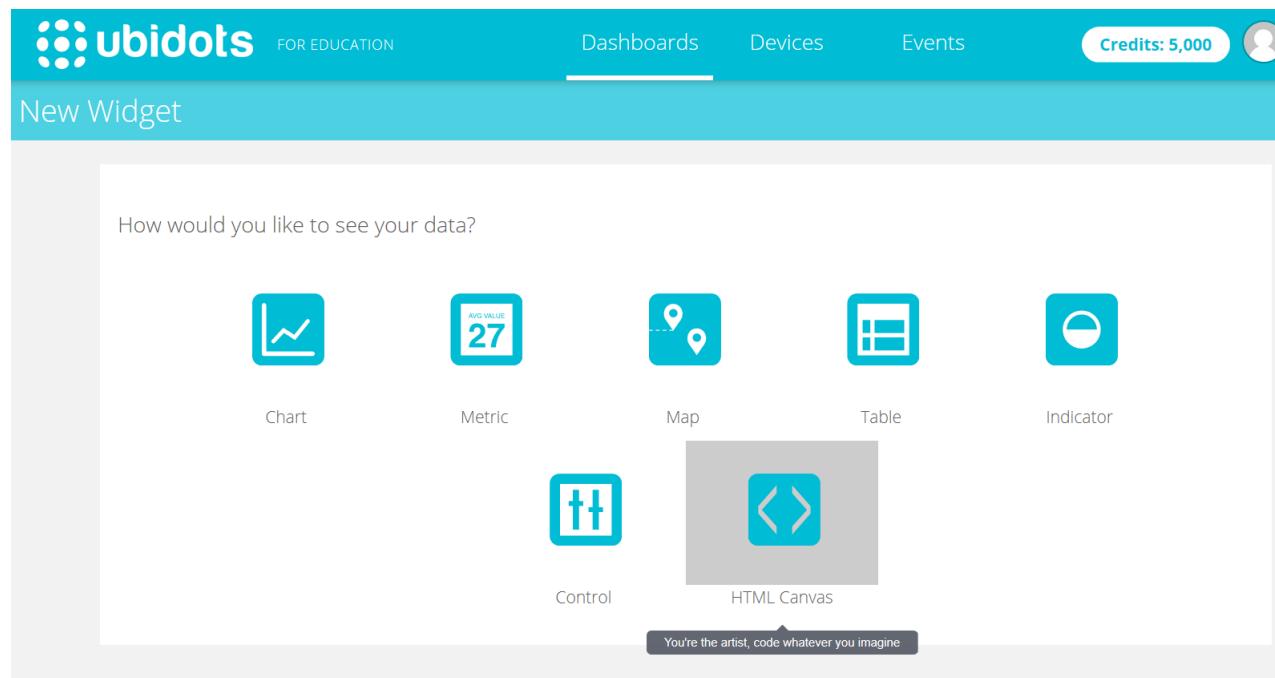
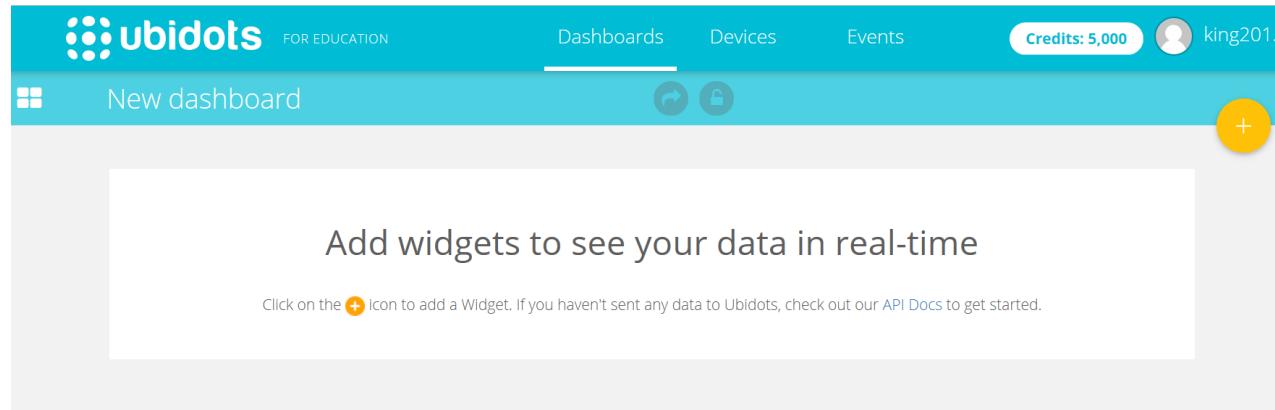
- Description: Click here to add a description
- API Label: smartledcontroller
- ID: 5d11a944c03f972d1240...
- Tags: Add tag
- Last Activity: No last activity

In the main area, there is a button labeled "Add Variable" with a plus sign. Below it is a dropdown menu with three options: "Default" (circled in red), "Derived", and "Rolling window". At the bottom of the screen are navigation buttons for "Show 50 items" and arrows.

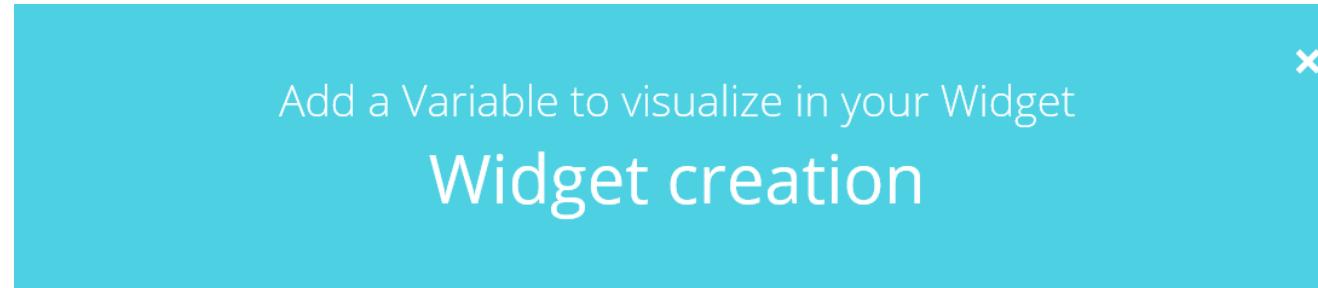
SmartLEDController

The screenshot shows the SmartLEDController interface with a variable configuration panel open. The variable is named "LedValue" and has the value "0". The panel includes a "Last activity" section stating "No last activity". The sidebar on the left is identical to the one in the first screenshot. At the bottom are navigation buttons for "Show 50 items" and arrows.

Dashboard : Add widgets



Dashboard : Select a Device, a Variable



Select a Device

Filter Device



SmartLED...



ESP8266



Demo

Select a Variable

Filter Variable



LedValue

Add Variable

Dashboard Creation : Control -> Switch

New Widget

How would you like to see your data?

Chart Metric Map Table Indicator

Control HTML Canvas

Select a type of widget:

Switch Slider

Select a Device

SmartLED ESP8266 Demo

Select a Variable

ledValue

Please provide the messages for your switch widget:

On message

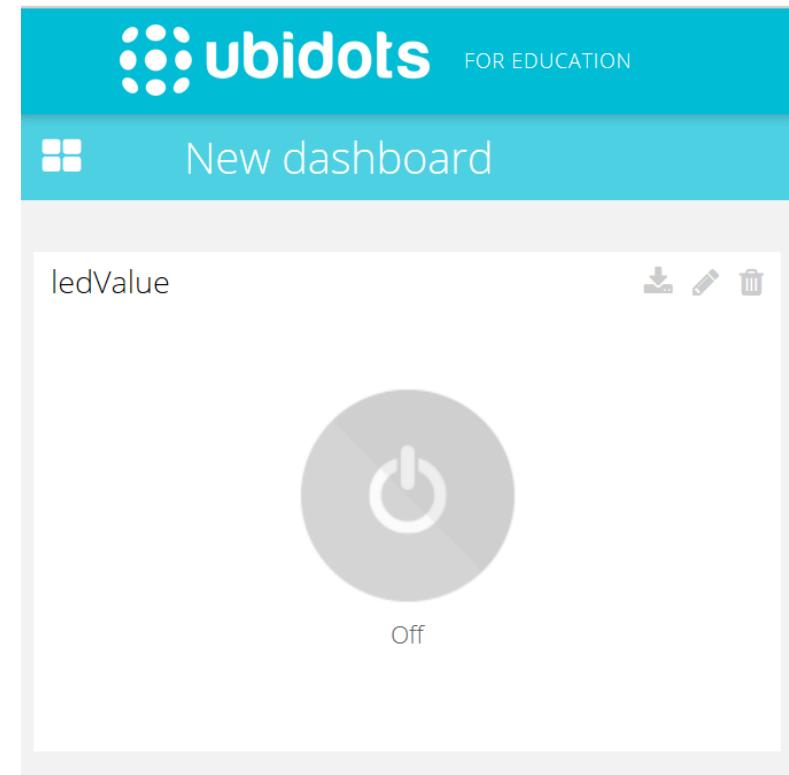
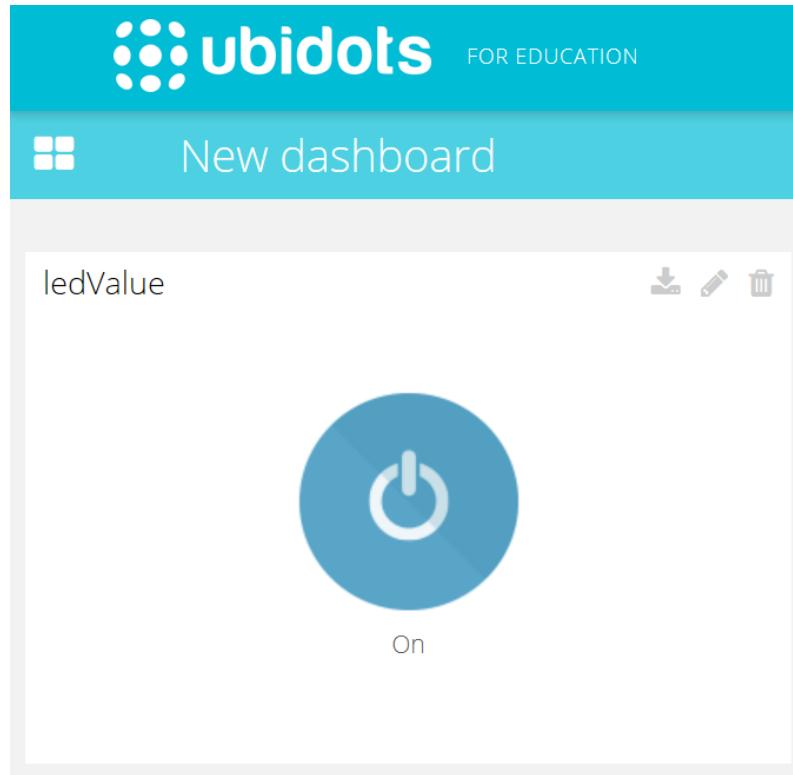
On

Off message

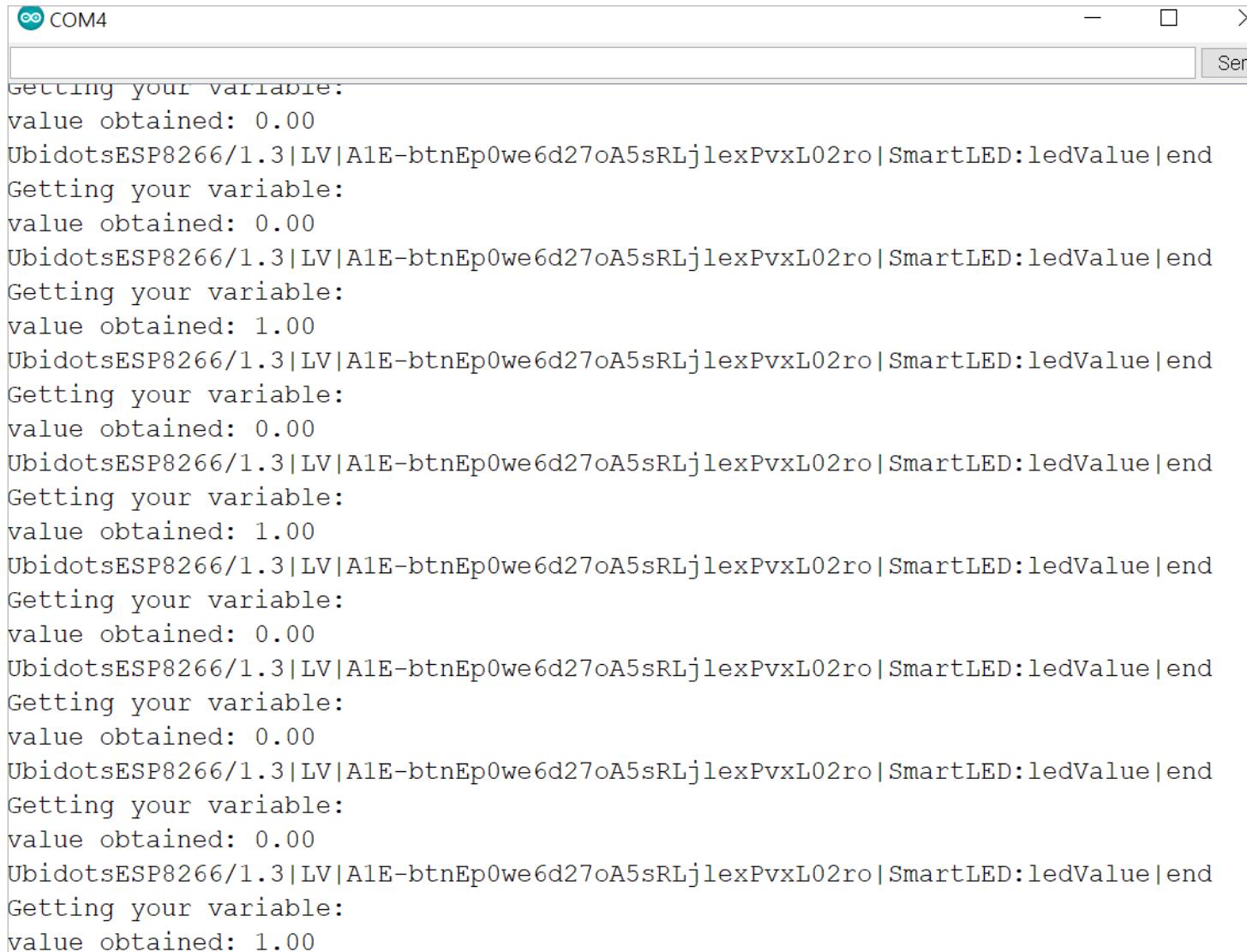
Off

Finish

Dashboard : Smart Switch Controller



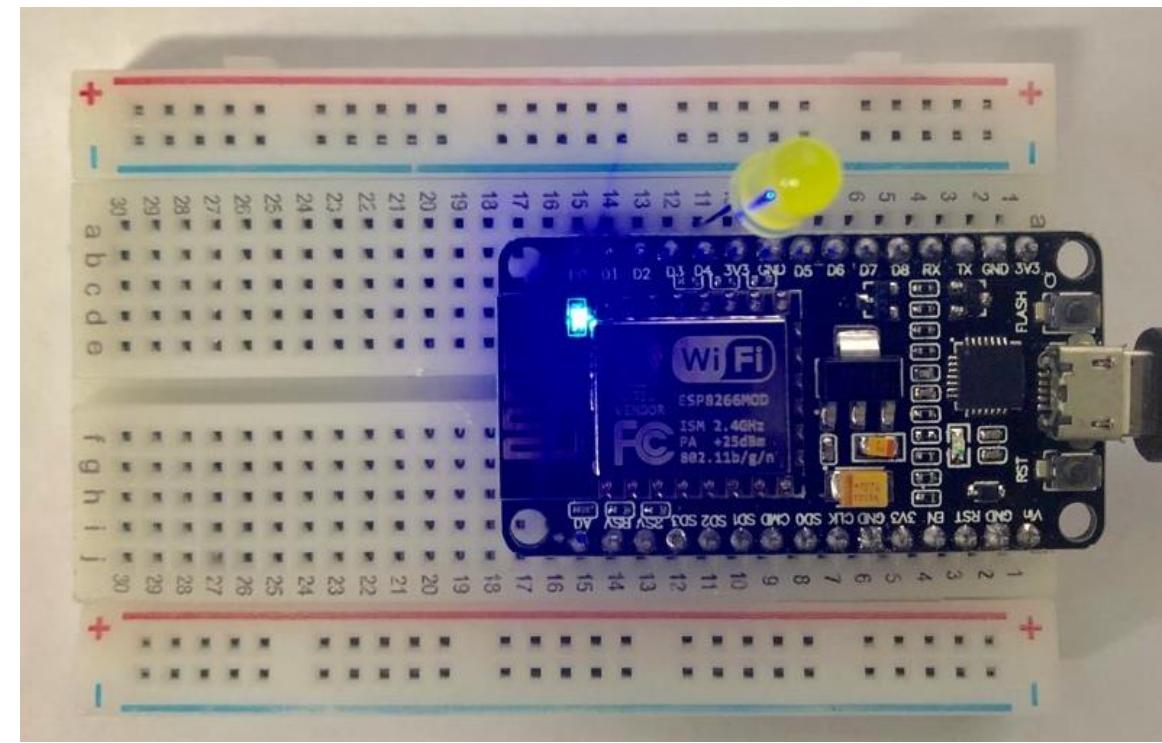
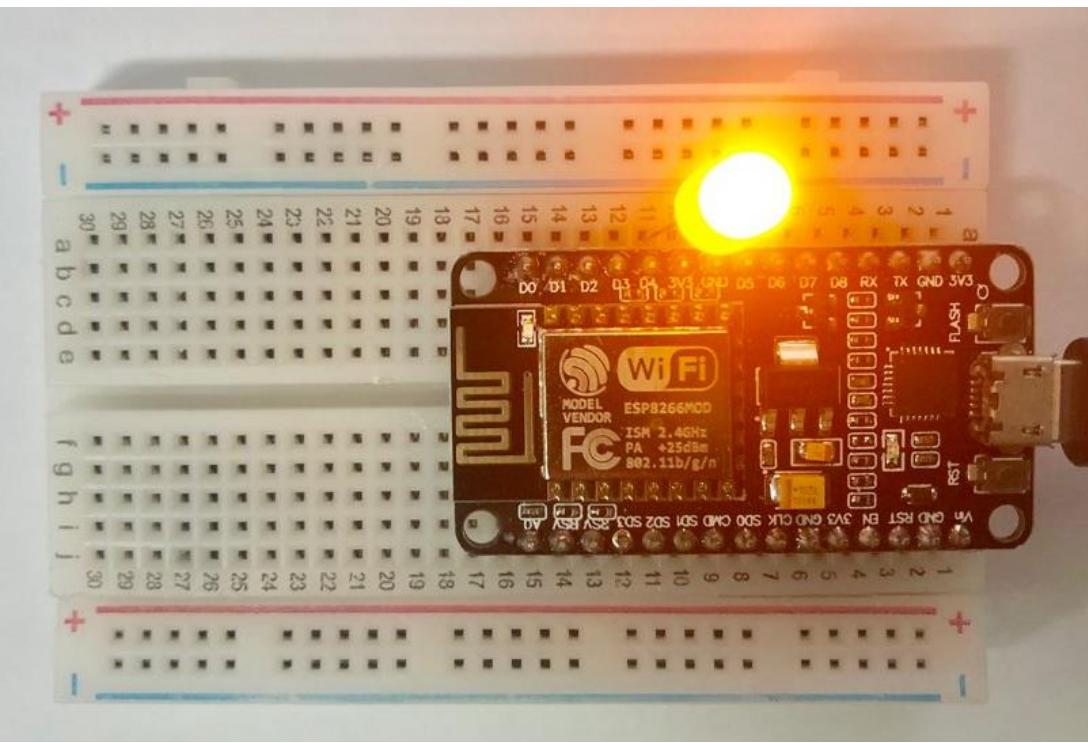
Serial Monitor of LED Control



The image shows a screenshot of a Serial Monitor window titled "COM4". The window has a standard title bar with minimize, maximize, and close buttons. Below the title bar is a text input field with a "Send" button. The main area of the window displays a continuous stream of text, which appears to be data from an ESP8266 module connected via Ubidots. The data is repeated in a loop, showing the following sequence:

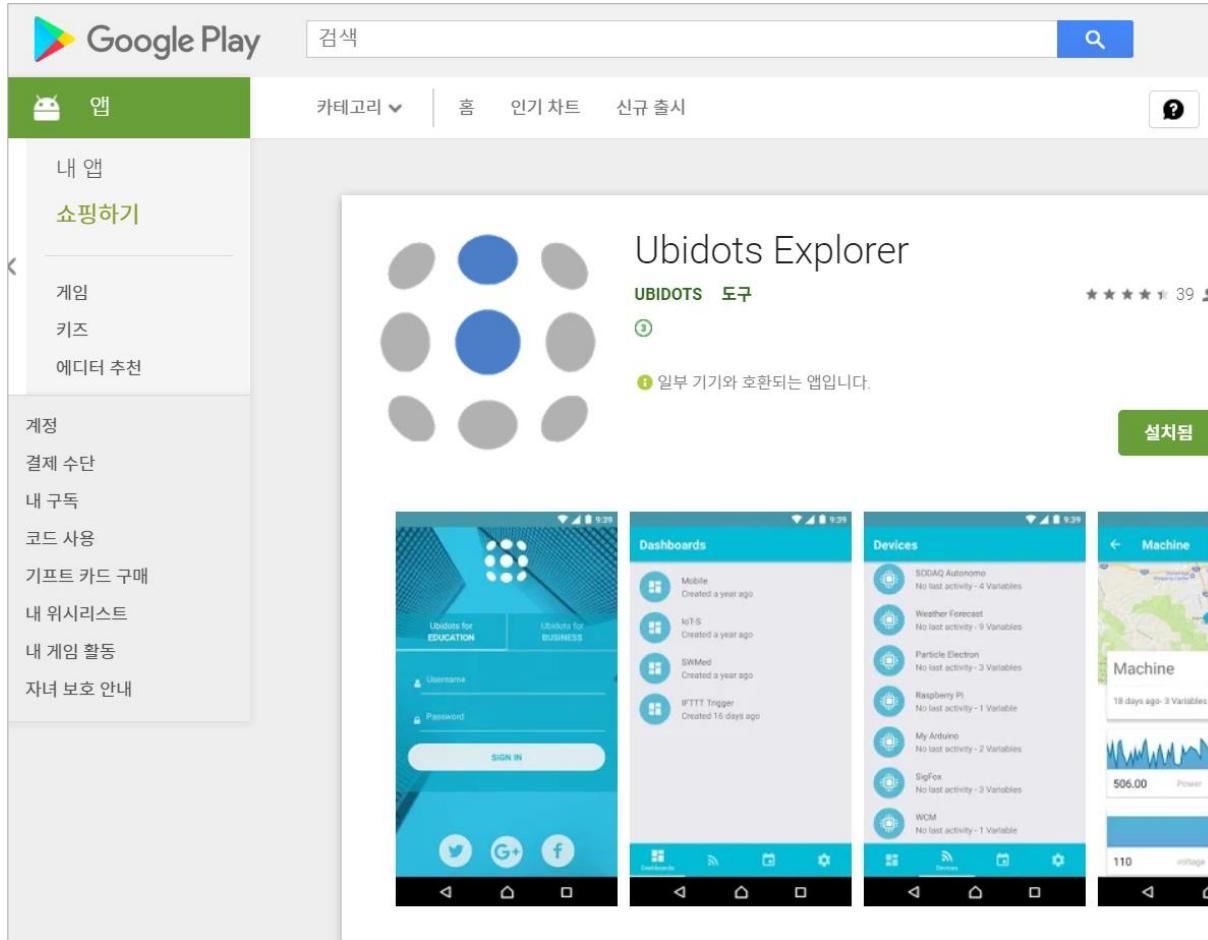
```
Getting your variable:  
value obtained: 0.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 0.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 1.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 0.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 1.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 0.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 0.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 1.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 0.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 0.00  
UbidotsESP8266/1.3|LV|A1E-btnEp0we6d27oA5sRLjlexPvxL02ro|SmartLED:ledValue|end  
Getting your variable:  
value obtained: 1.00
```

Operation of IoT Devices



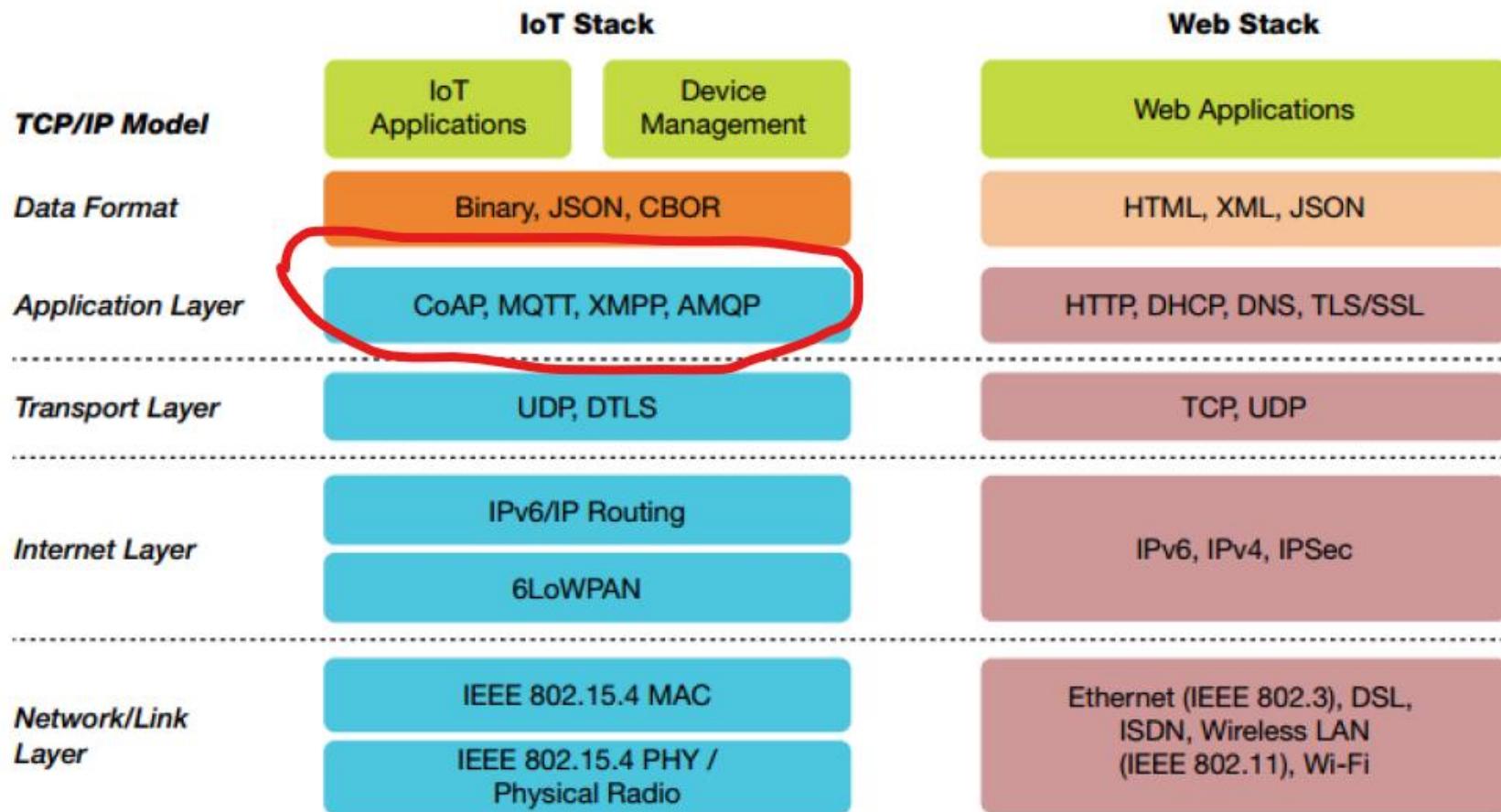
Mobile Application : Ubidots Explorer

- <https://play.google.com/store/apps/details?id=com.ubidots.ubiapp>



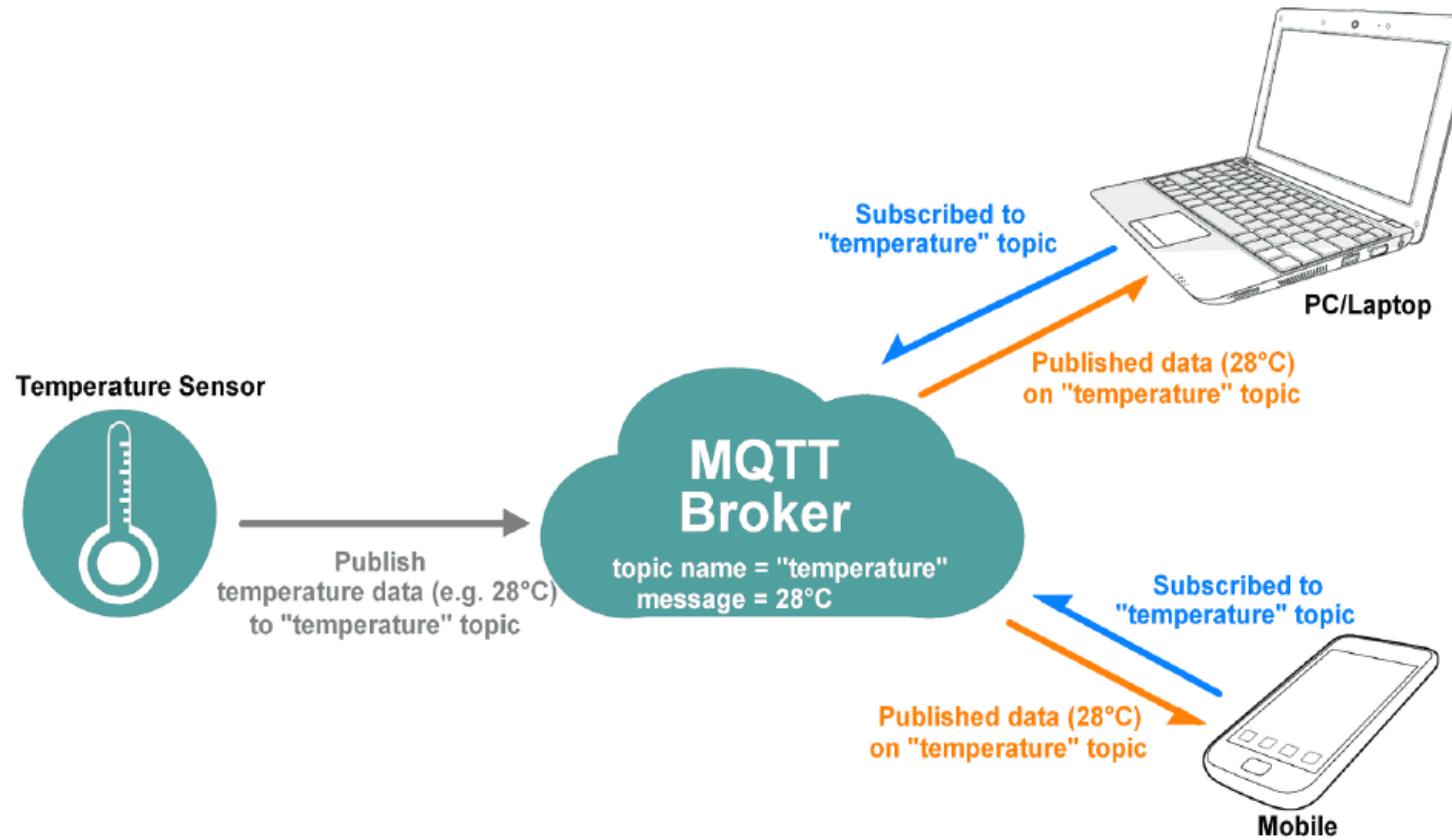
Future Study Topics

- IoT Communication Protocols



- Use Ubidots MQTT...

- <https://github.com/ubidots/ubidots-mqtt-esp>
- <https://ubidots.com/blog/temperature-control-with-ubidots/>



- Big Data Analytics, Machine Learning
- Apply IoT to field areas
 - Smart Energy :
 - Energy-As-A-Service : <http://tiny.cc/iigb9y>
 - Smart Farm
 - Smart Home
 - Smart Healthcare
 - Smart Transportation
 - Smart Factory ...

Program Sources & Libraries

- GitHub : <https://github.com/kings-iot-2019>

References

1. <https://www.theengineeringprojects.com/2018/10/introduction-to-nodemcu-v3.html>
2. <http://pdacontrolen.com/tutorial-platform-iot-ubidots-esp8266-sensor-dht11/>
3. <https://www.hackster.io/ubidots/projects>
4. <https://ubidots.com/education/>
5. <https://ubidots.com/docs/>