

## 2~3 Lecture Contents

- Setting up the NodeMCU programming environment

Hello World! Output

Digital I/O Function, Analog I/O Function

LED Control Basics

Drawing Schematics with Fritzing

Exercises

2 LED blink

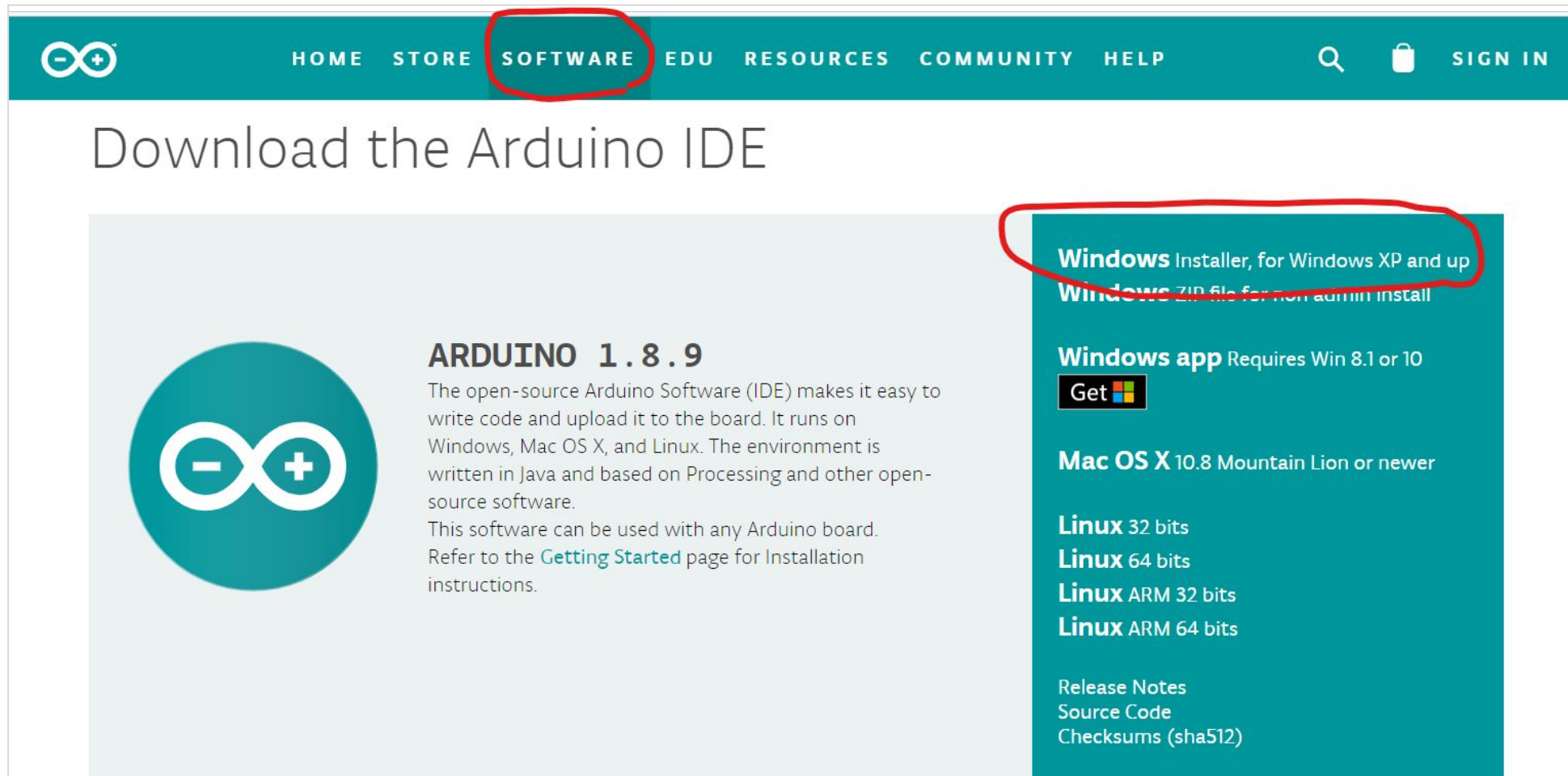
ADC (Analog Digital Converter) : A0 Analog Pin

LED brightness control with PWM

LED brightness control using photoresistors (light sensors)

# NodeMCU Arduino Programming Settings

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



The screenshot shows the Arduino website's 'SOFTWARE' page. The navigation bar at the top includes links for HOME, STORE, SOFTWARE (highlighted with a red circle), EDU, RESOURCES, COMMUNITY, and HELP. Below the navigation bar, the heading 'Download the Arduino IDE' is displayed. On the left, there is a large circular logo with the Arduino infinity symbol. To the right of the logo, the text 'ARDUINO 1.8.9' is shown, followed by a description of the IDE and its compatibility with Windows, Mac OS X, and Linux. On the right side of the page, a teal sidebar contains links for downloading the IDE on various operating systems. The 'Windows' section is highlighted with a red circle and includes links for 'Windows Installer, for Windows XP and up' and 'Windows ZIP file for non-admin install'. Other links in the sidebar include 'Windows app', 'Mac OS X', 'Linux 32 bits', 'Linux 64 bits', 'Linux ARM 32 bits', 'Linux ARM 64 bits', 'Release Notes', 'Source Code', and 'Checksums (sha512)'.

HOME STORE **SOFTWARE** EDU RESOURCES COMMUNITY HELP

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 \(sha512\)](#)

# Download and install the NodeMCU drive

- NodeMCU USB to UART Bridge Chip

For CP102: The chip we use

FOR CH340

CP102 Drive Download

<https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers#windows>

Unzip the zip file and select x64 if your PC has more than 4GB of memory, or x86 if less than

In the case of the NodeMCU for the CH340 chip, the relevant drive must be downloaded and installed.

# Download and install NodeMCU CP102 driver


## Download for Windows 10 Universal (v10.1.3)

Platform	Software	Release Notes
 Windows 10 Universal	<a href="#">Download VCP (2.3 MB)</a>	<a href="#">Download VCP Revision History</a>

## Download for Windows 7/8/8.1 (v6.7.6)

Platform	Software	Release Notes
 Windows 7/8/8.1	<a href="#">Download VCP (5.3 MB) (Default)</a>	<a href="#">Download VCP Revision History</a>
 Windows 7/8/8.1	<a href="#">Download VCP with Serial Enumeration (5.3 MB)</a> <a href="#">Learn More »</a>	<a href="#">Download VCP Revision History</a>

## Download for Windows XP/Server 2003/Vista/7/8/8.1 (v6.7)

Platform	Software	Release Notes
 Windows XP/Server 2003/Vista/7/8/8.1	<a href="#">Download VCP (3.66 MB)</a>	<a href="#">Download VCP Revision History</a>

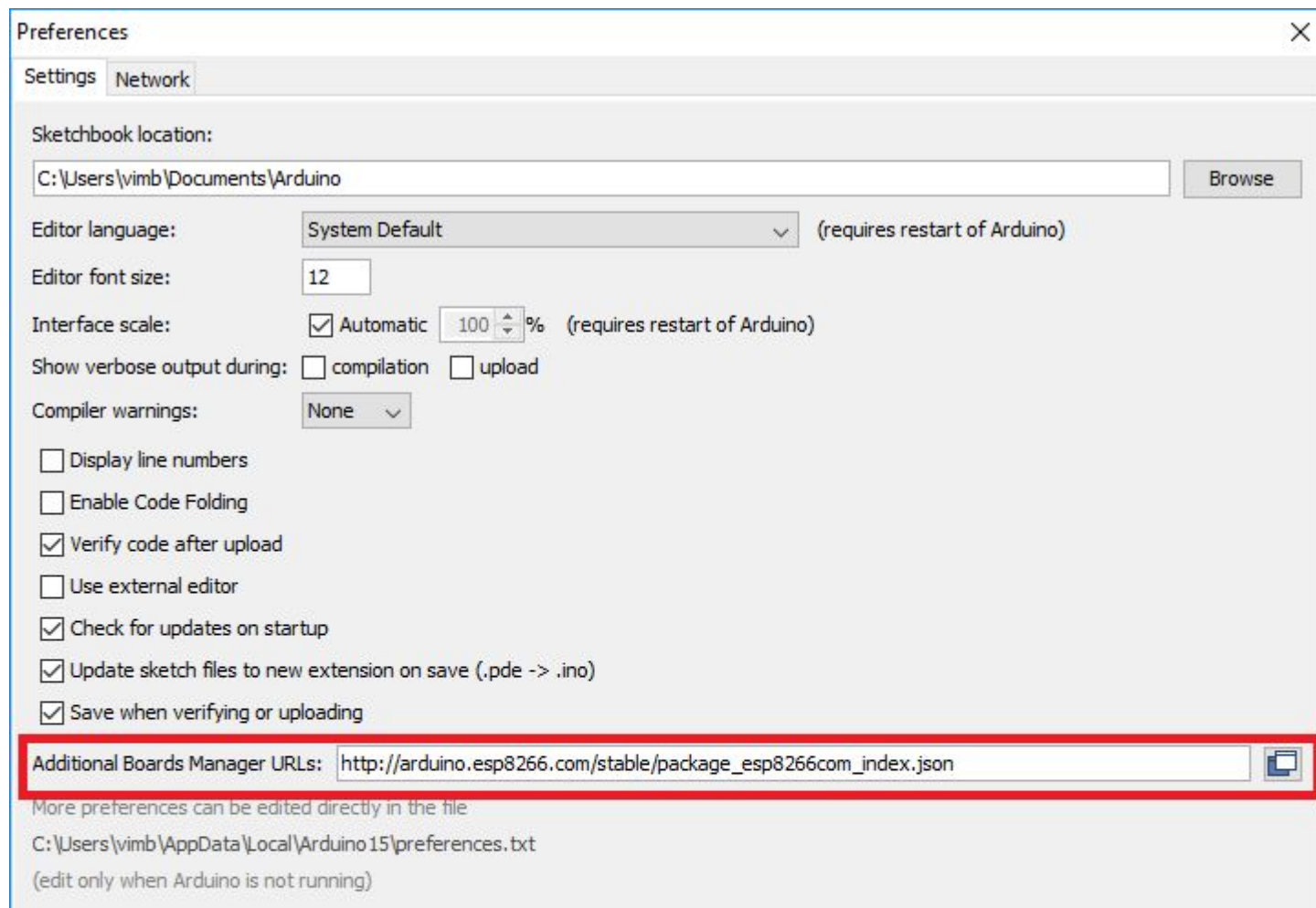
## Download for Windows 2K (v6.3a)

Important information regarding the Silicon Labs website: this site uses cookies to improve user experience and stores information on your computer. By continuing to use our site, you consent to our [Cookie Policy](#). If you do not want to enable cookies, review our policy and learn how they can be disabled. Note that disabling cookies will disable some features of the site.

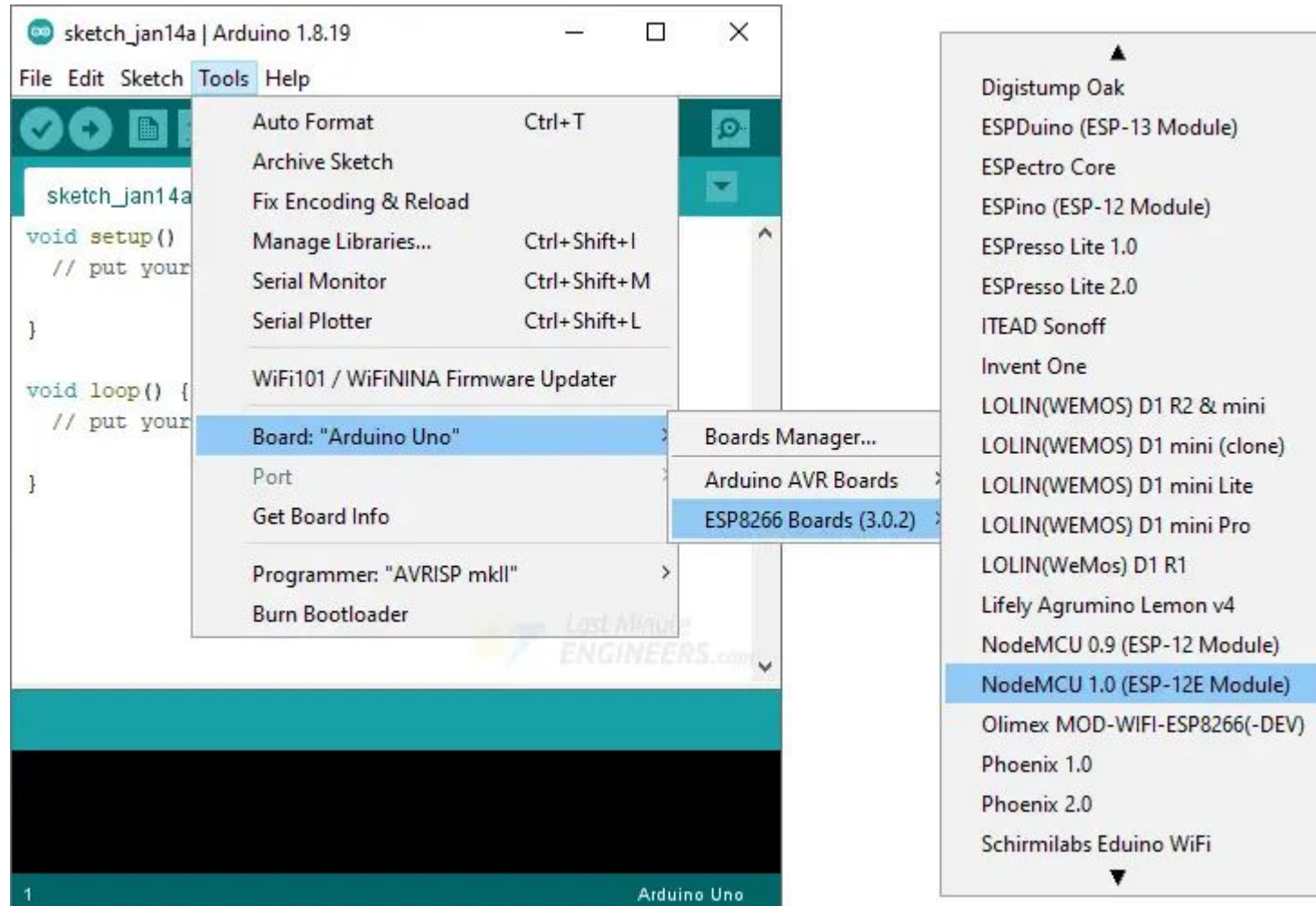
Accept

# 보드 관리자 URLs 설정

- Preferences – > Add additional board manager URLs  
“[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)”

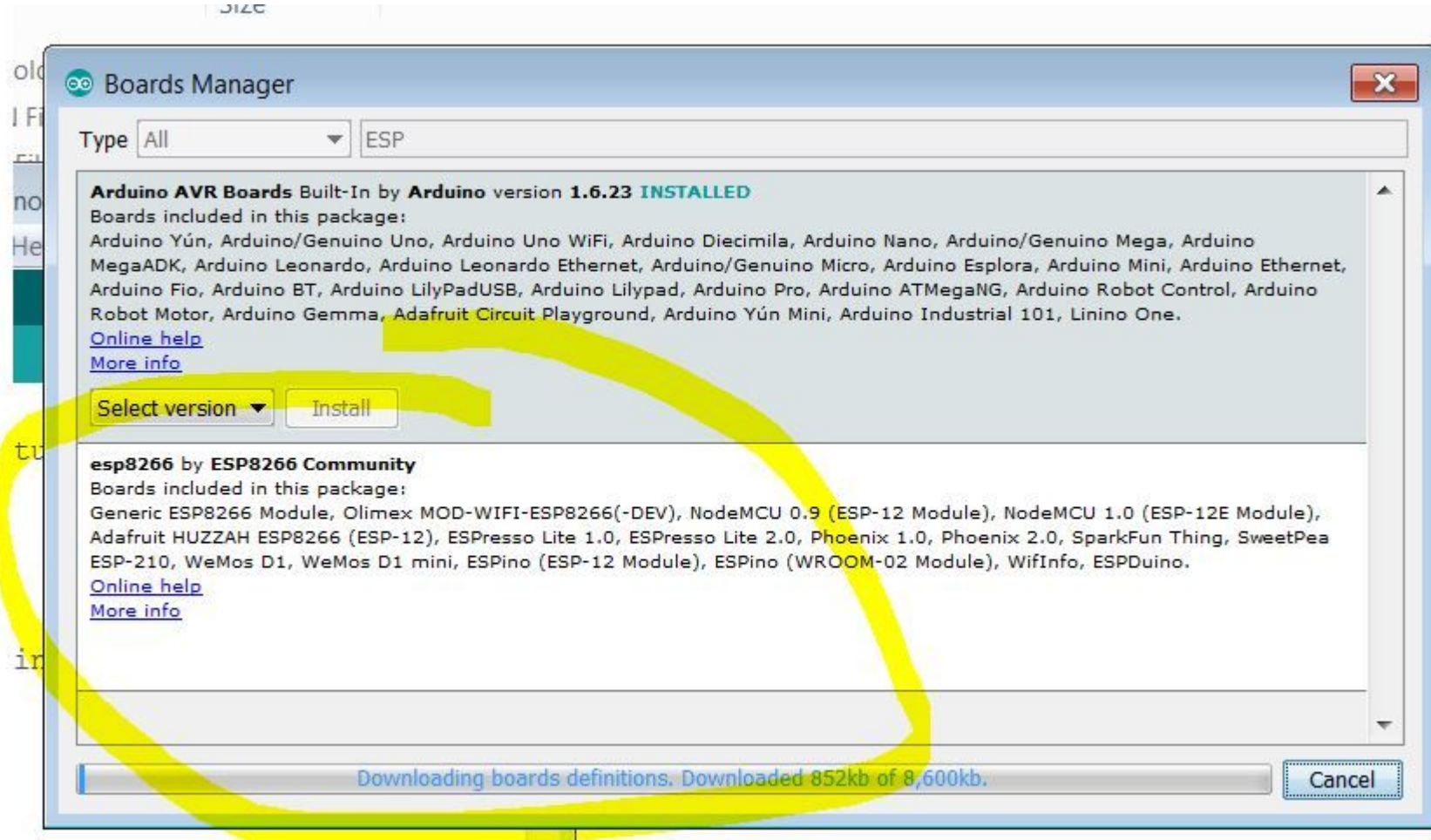


# Board Manager Settings: ESP8266 Search

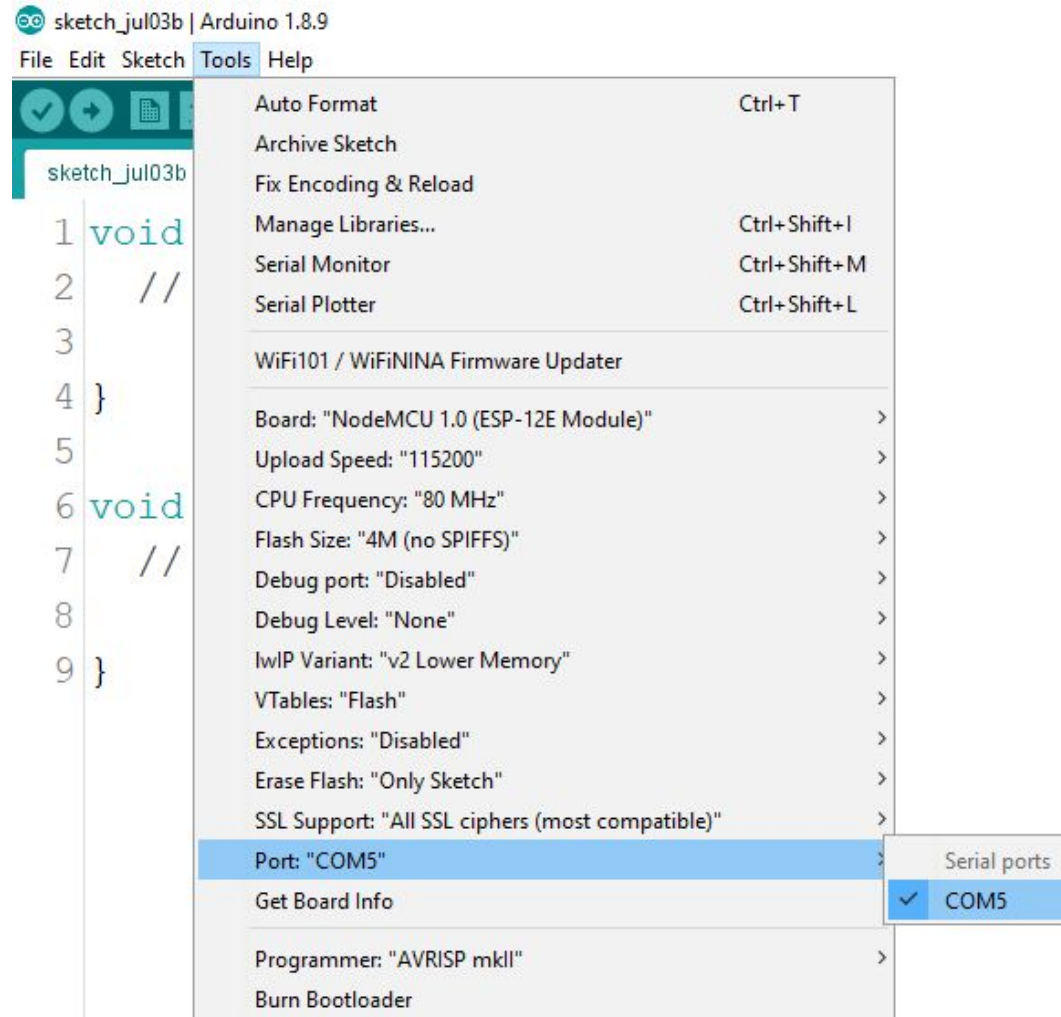




# Board admin settings: ESP8266 search

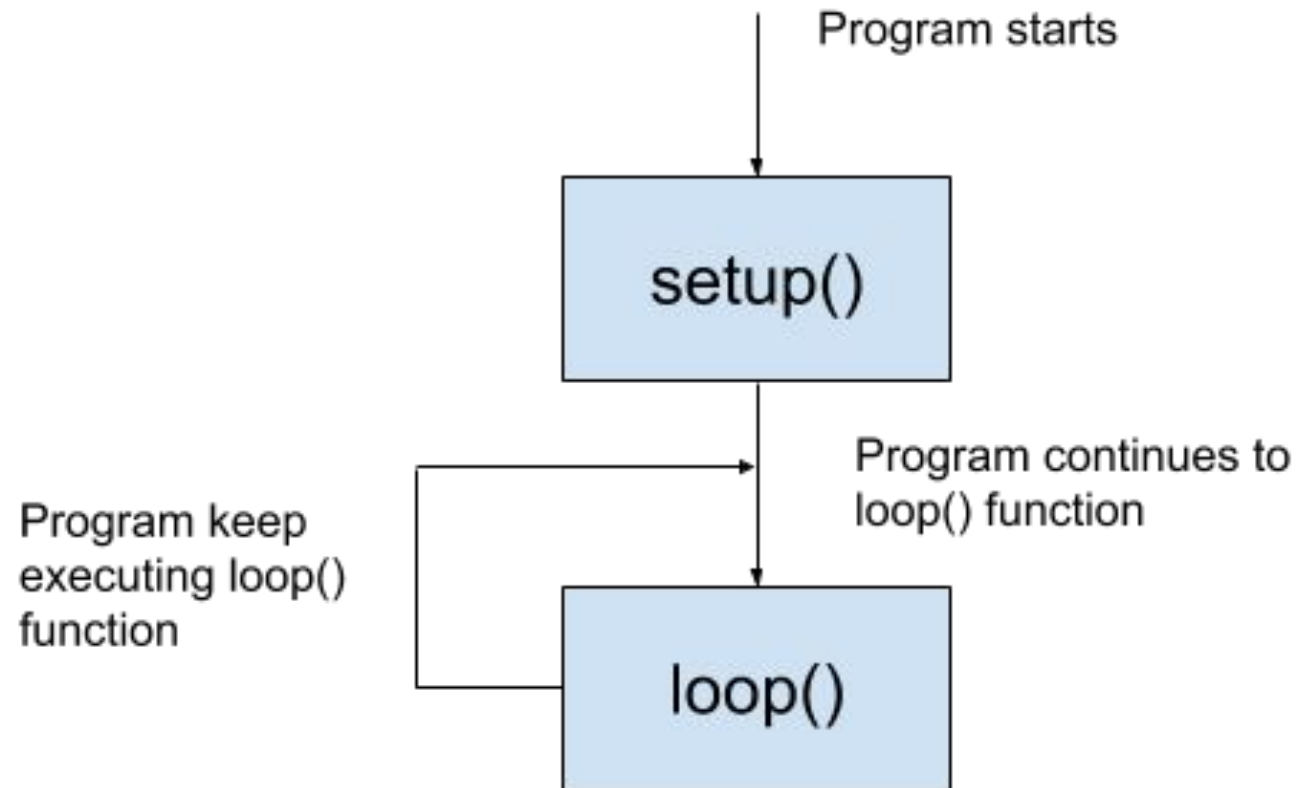


# Serial Port Settings



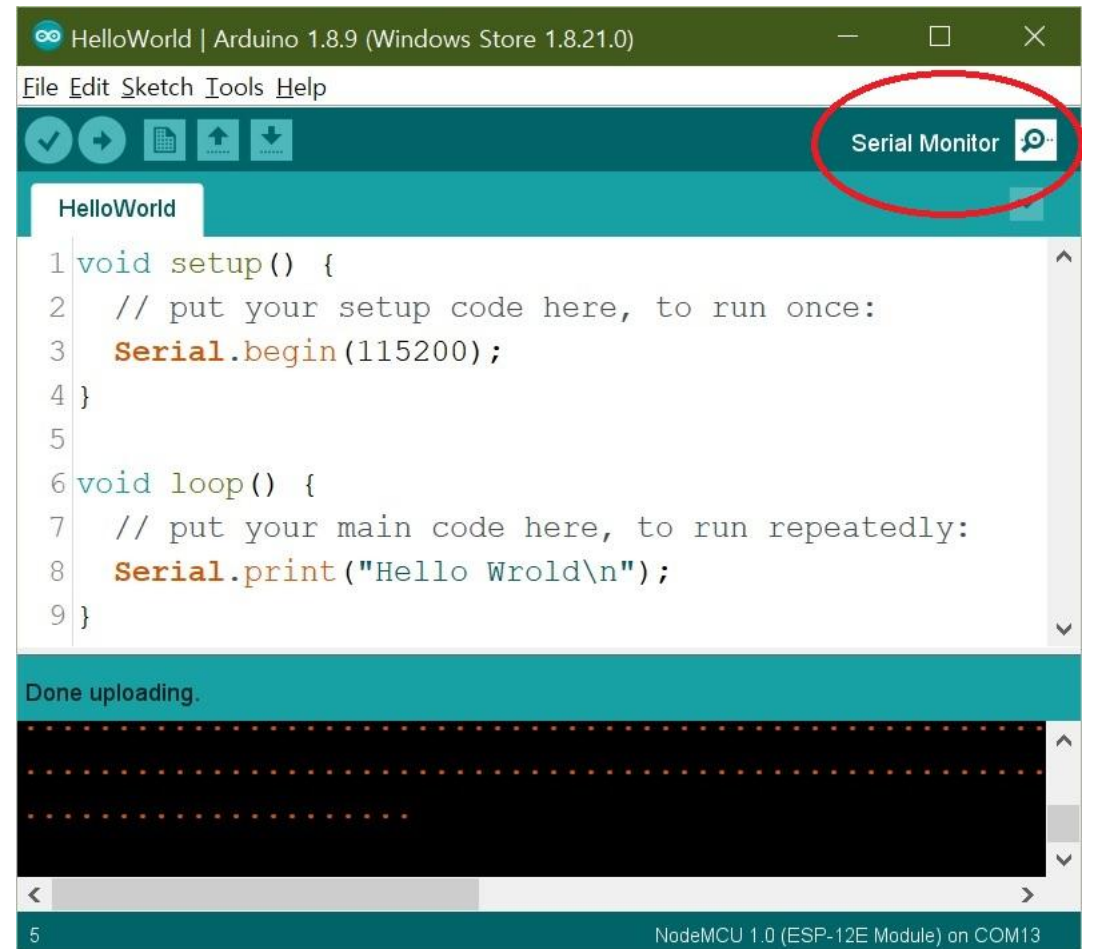


# Arduino Programming Architecture



# 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");  
}
```



# Digital I/O Functions

- **pinMode(pin, mode)**

- Assigned Pin Number(**pin**) as input or mode.
- **pin** : GPIO Pin Number
- **mode** : **INPUT**, **OUTPUT**, **INPUT\_PULLUP**, **INPUT\_PULLDOWN**

- **digitalWrite(pin, value)**

- On the specified digital pin **HIGH** (5V or 3.3V) or **LOW** (0V or ground) Writing Values

- **delay(ms)**

- Specifying the program's latency (milliseconds) Arduino Function Note:

<https://www.arduino.cc/reference/en/>

# Analog input/output functions

- **`pinMode(pin, mode)`**

- Configure a specified pin number as input or output (mode) output.

- **`analogRead(pin)`**

- Read analog values (0~1023) with analog pins10 bit ADC :  $2^{10} = 1024$

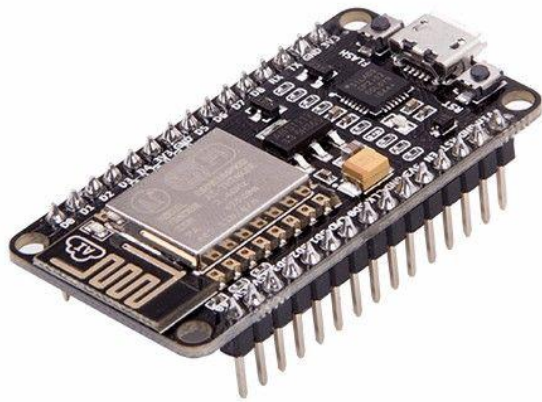
- **`analogWrite(pin, value)`**

- The NodeMCU has one analog pin at A0 Arduino Functions Note:

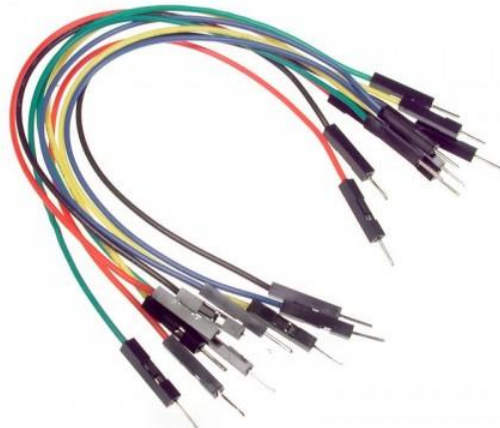
<https://www.arduino.cc/reference/en/>

# NodeMCU LED Control Fundamentals

- What to prepare



NodeMCU



Jump  
Cable

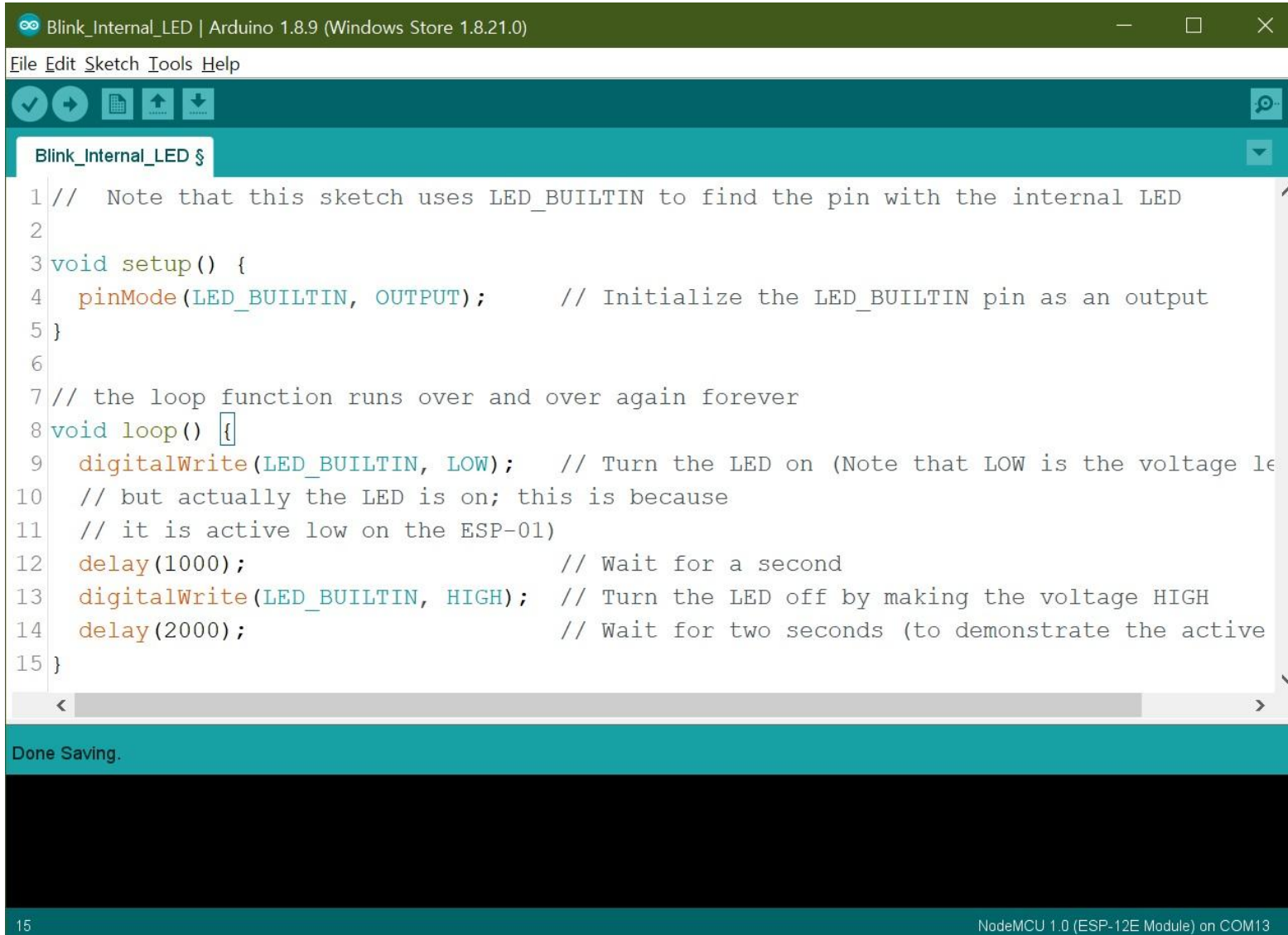


LE  
D



Resistor  
(220  $\Omega$ )

# LED ON/OFF : LED\_BUILTIN(D0)



The screenshot shows the Arduino IDE interface. The title bar reads "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 bar is a toolbar with icons for saving, running, and uploading. The main text area shows the following code:

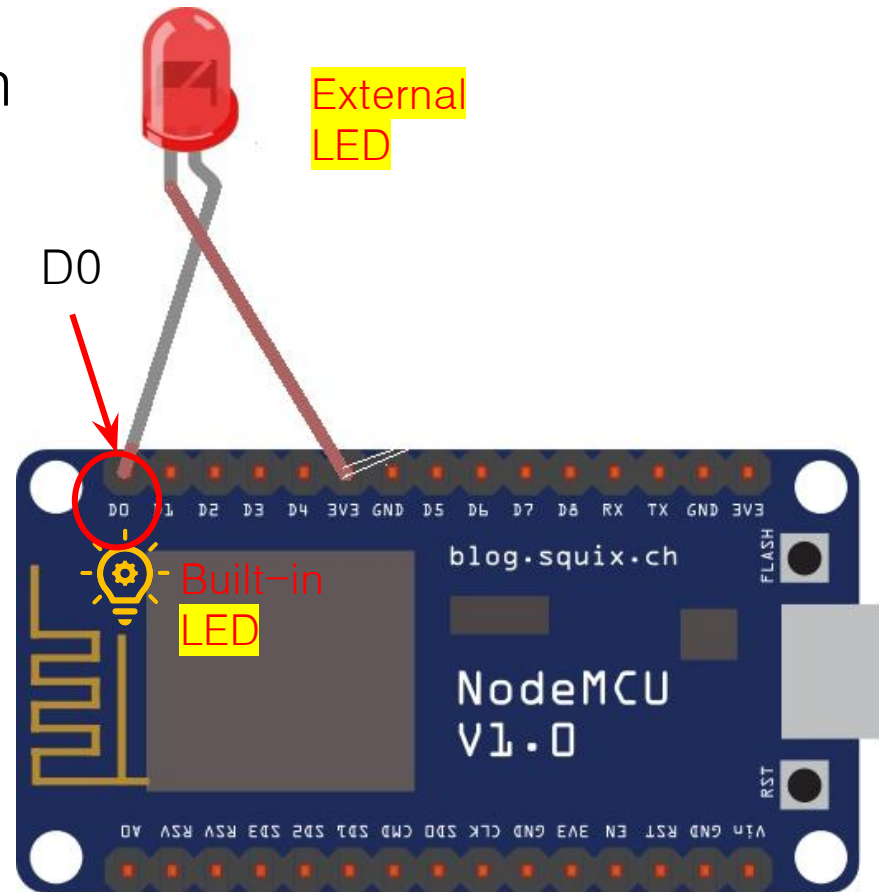
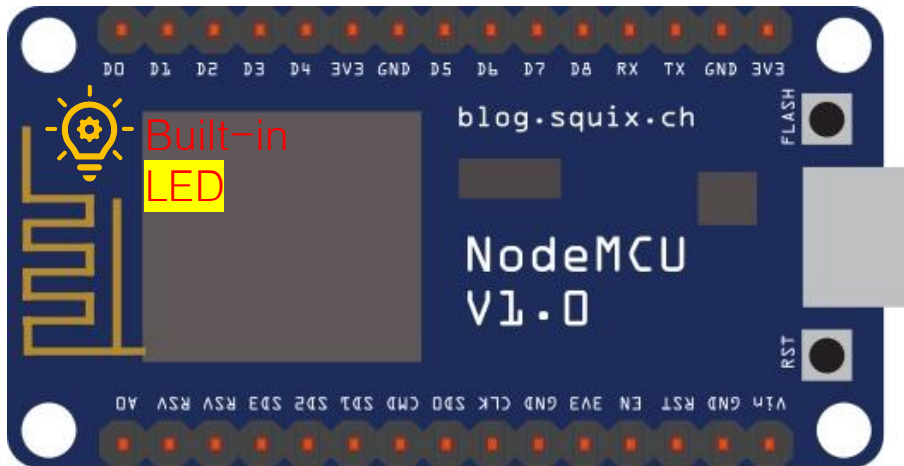
```
Blink_Internal_LED $
1 // Note that this sketch uses LED_BUILTIN to find the pin with the internal LED
2
3 void setup() {
4   pinMode(LED_BUILTIN, OUTPUT);    // Initialize the LED_BUILTIN pin as an output
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9   digitalWrite(LED_BUILTIN, LOW);  // Turn the LED on (Note that LOW is the voltage level
10  // but actually the LED is on; this is because
11  // it is active low on the ESP-01)
12   delay(1000);                     // Wait for a second
13   digitalWrite(LED_BUILTIN, HIGH); // Turn the LED off by making the voltage HIGH
14   delay(2000);                     // Wait for two seconds (to demonstrate the active
15 }
```

Below the code editor, a status bar shows "Done Saving." and a black area for serial output. At the bottom, the status bar indicates "NodeMCU 1.0 (ESP-12E Module) on COM13".



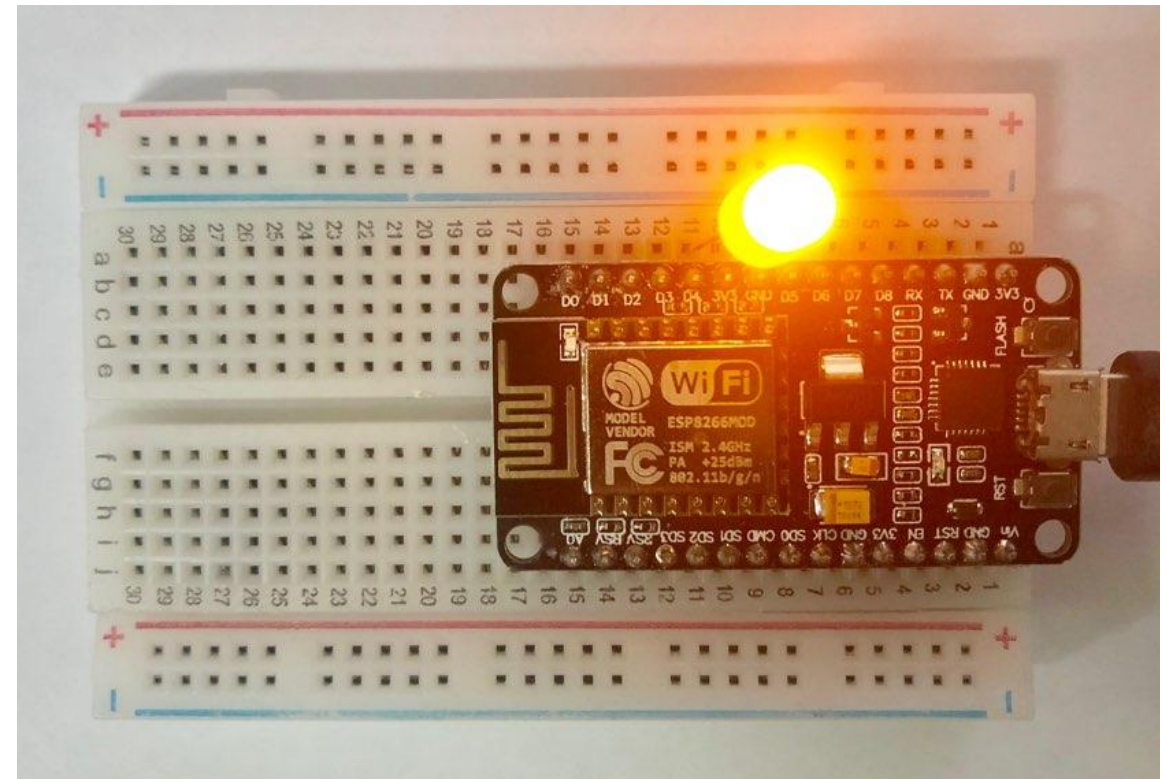
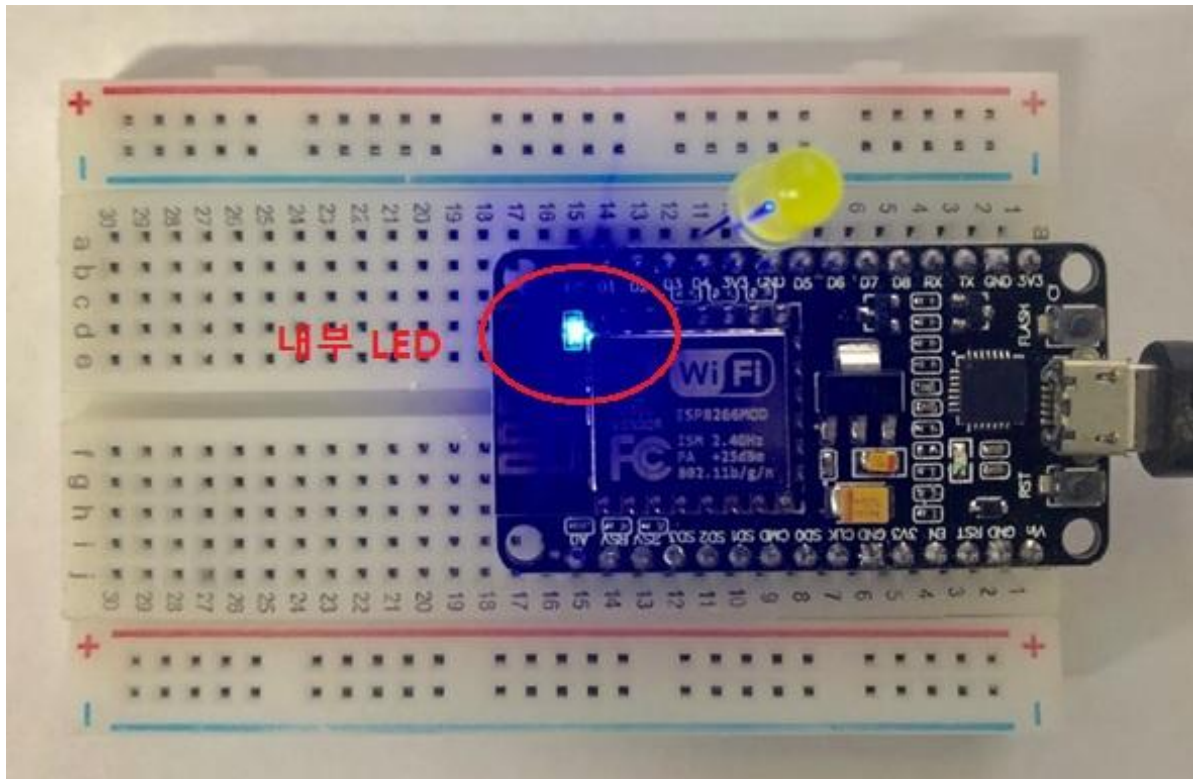
# Drawing a schematic: Fritzing

- Built-in LED(**LED\_BUILTIN**) D0 pin

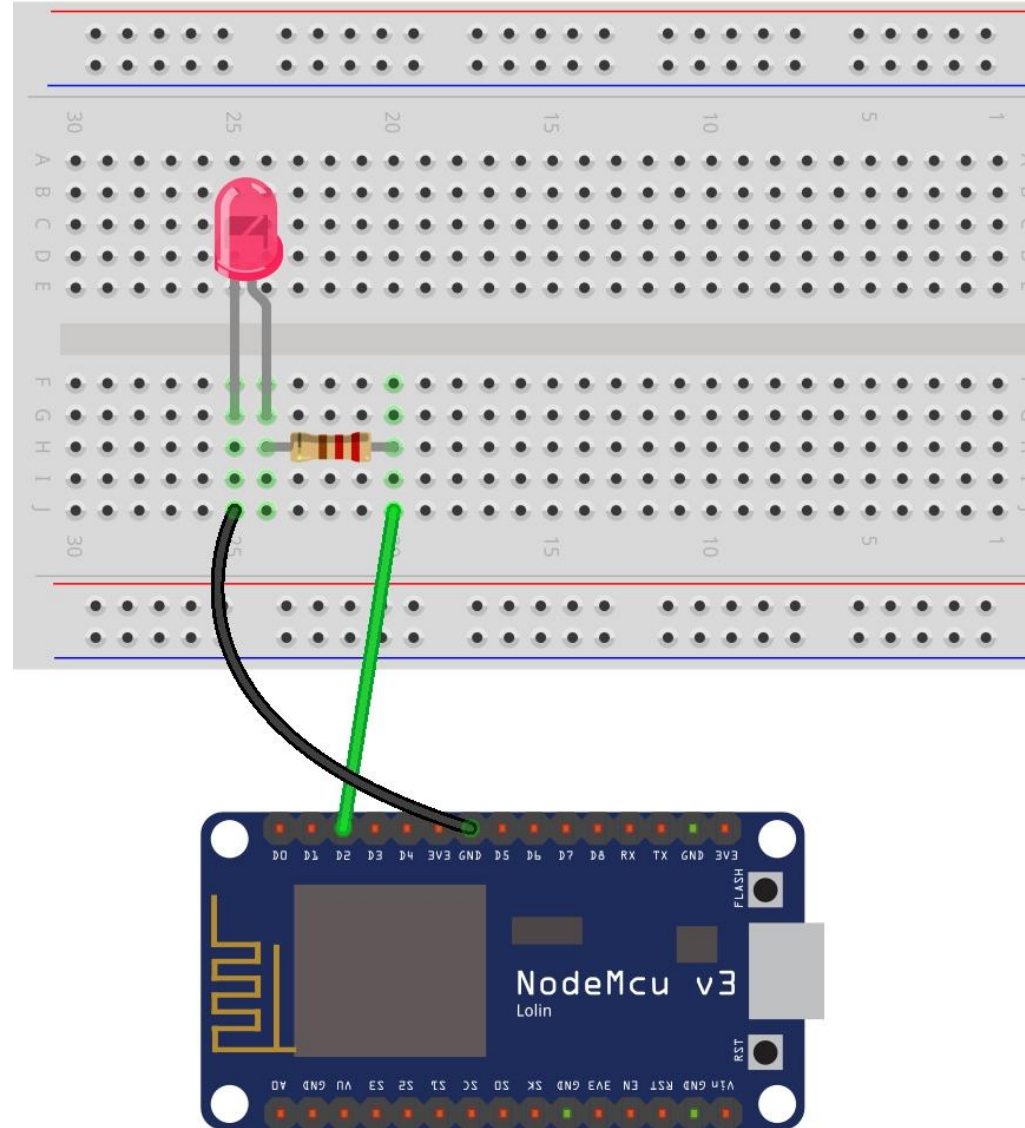


# Built-in LED action confirmation : D0(GPIO4) pin

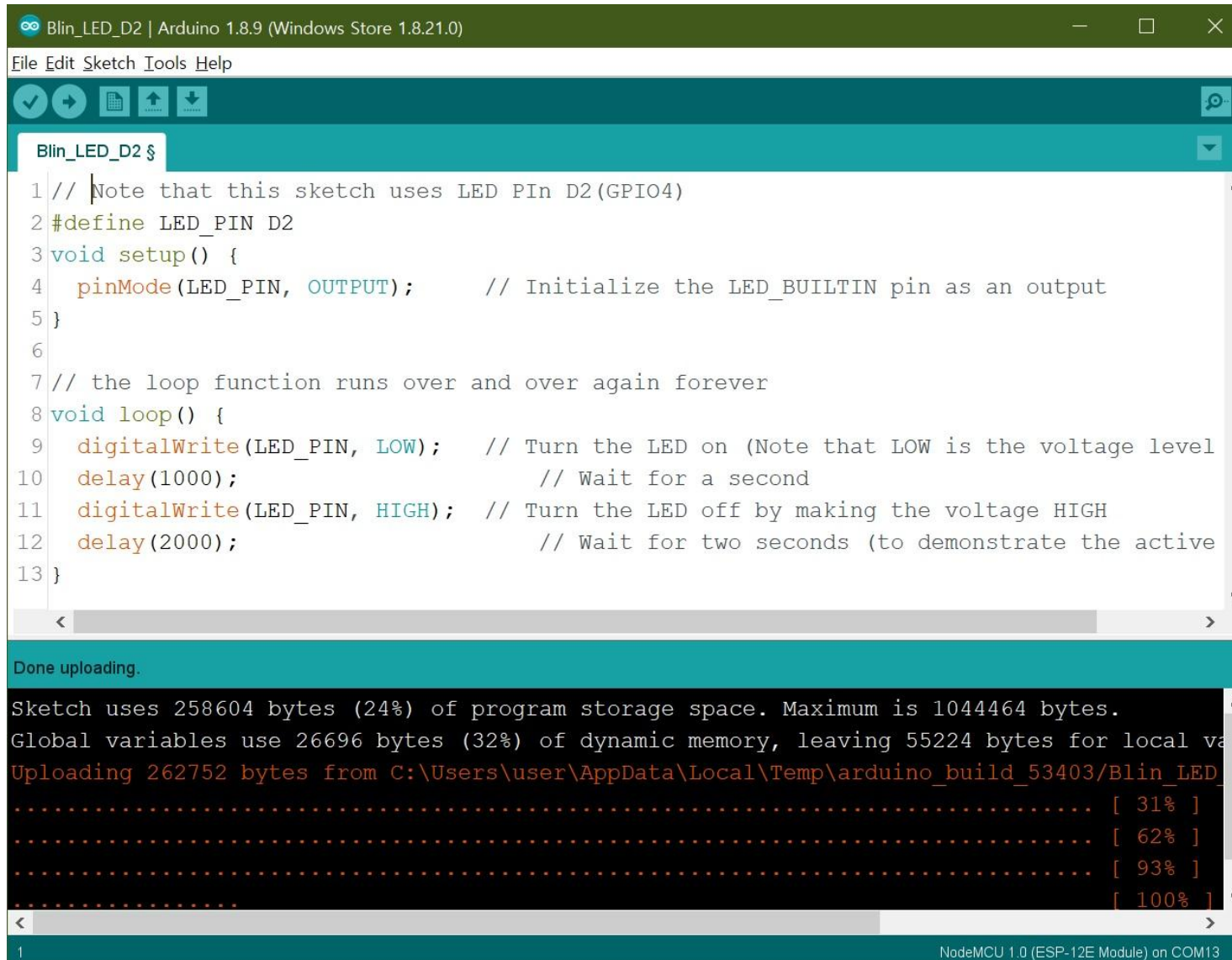
- There is no current resistance (220  $\Omega$ ), so if you leave it on for a long time, the LED will heat up due to overload (caution!!)



# LED control : D2(GPIO4) pin



# sketch : D2(GPIO4) pin



```
Blin_LED_D2 | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help

Blin_LED_D2 $
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 }
```

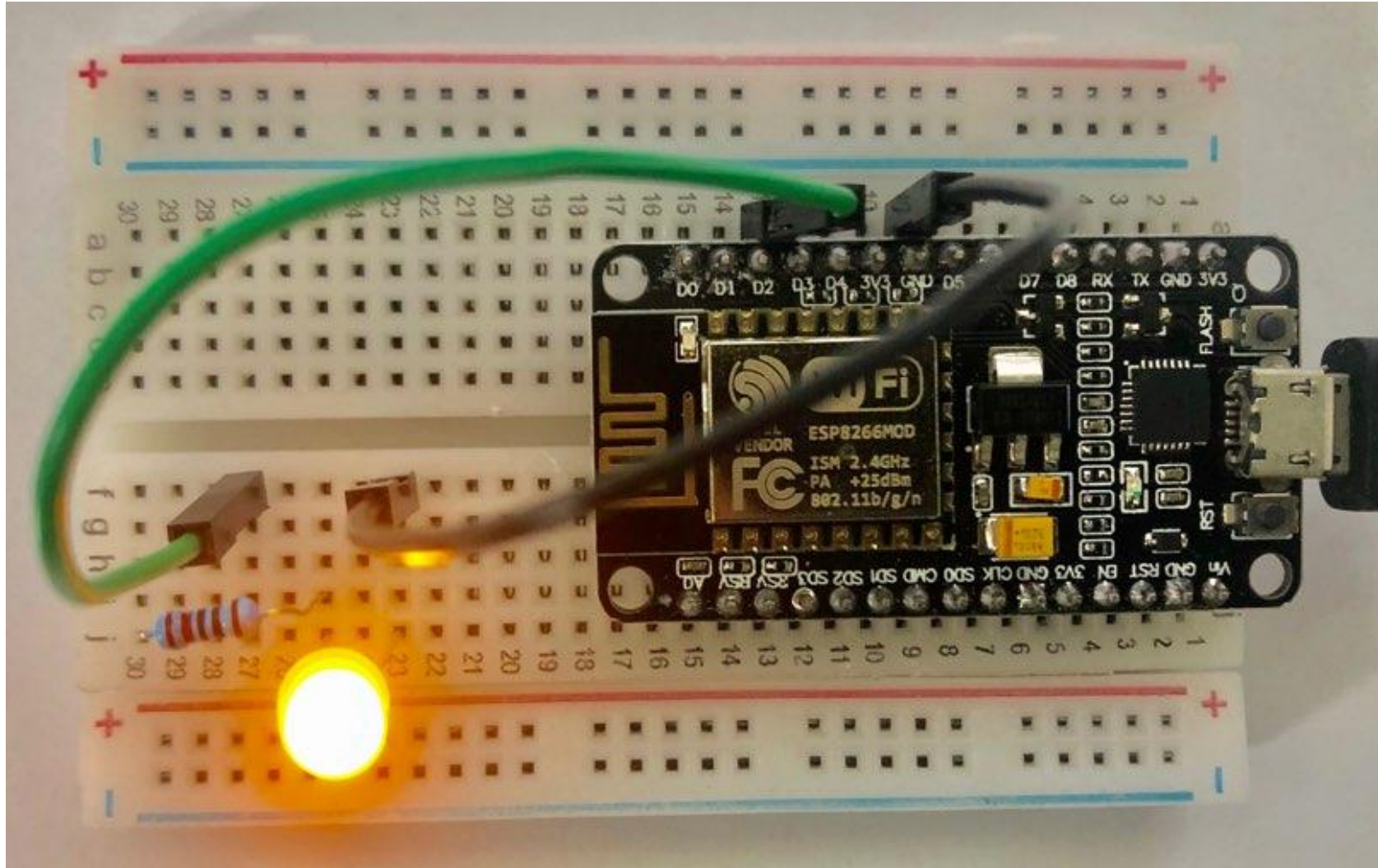
Done uploading.

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 variables.  
Uploading 262752 bytes from C:\Users\user\AppData\Local\Temp\arduino\_build\_53403\Blin\_LED\_D2.ino to COM13  
..... [ 31% ]  
..... [ 62% ]  
..... [ 93% ]  
..... [ 100% ]

1 NodeMCU 1.0 (ESP-12E Module) on COM13

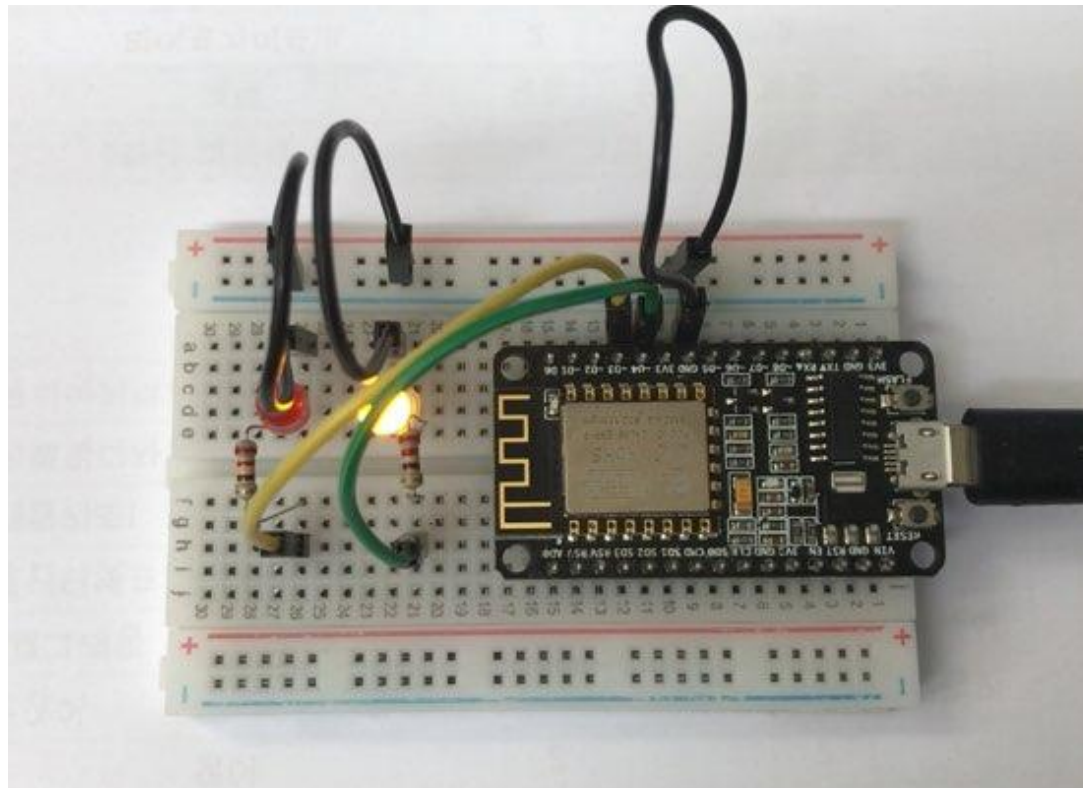


LED action : D2 pin, 220  $\Omega$  resistance



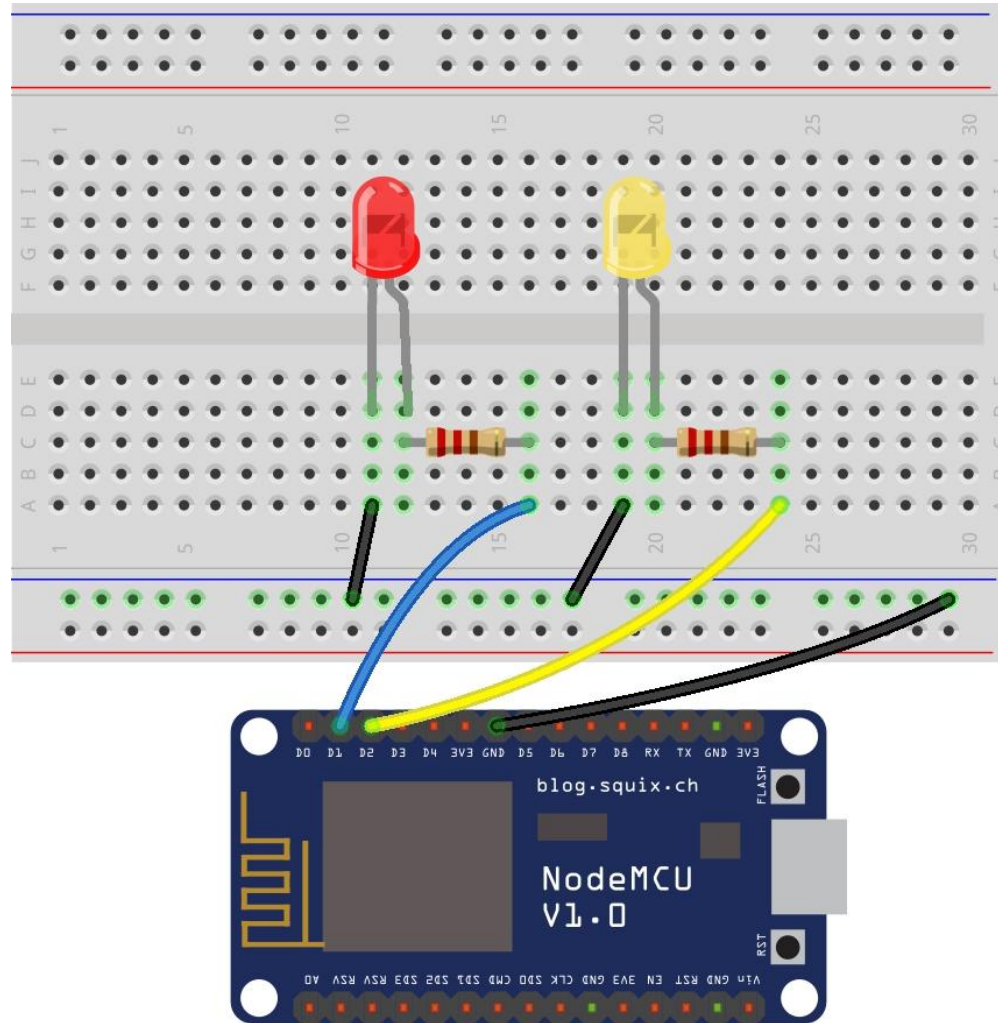
# Lab 01

- Use 2 LEDs to blink at 0.5 sec intervals (red→yellow→red→yellow....) `digitalWrite()`, 220  $\Omega$  resistance



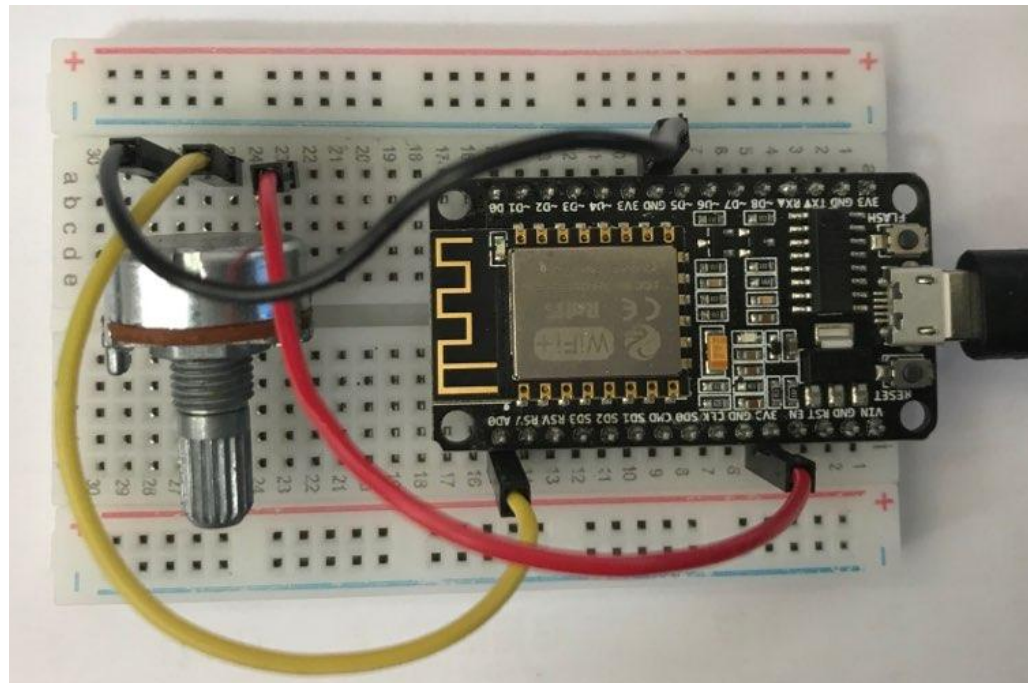


# Practice Exercises 01 : Circuit Wiring

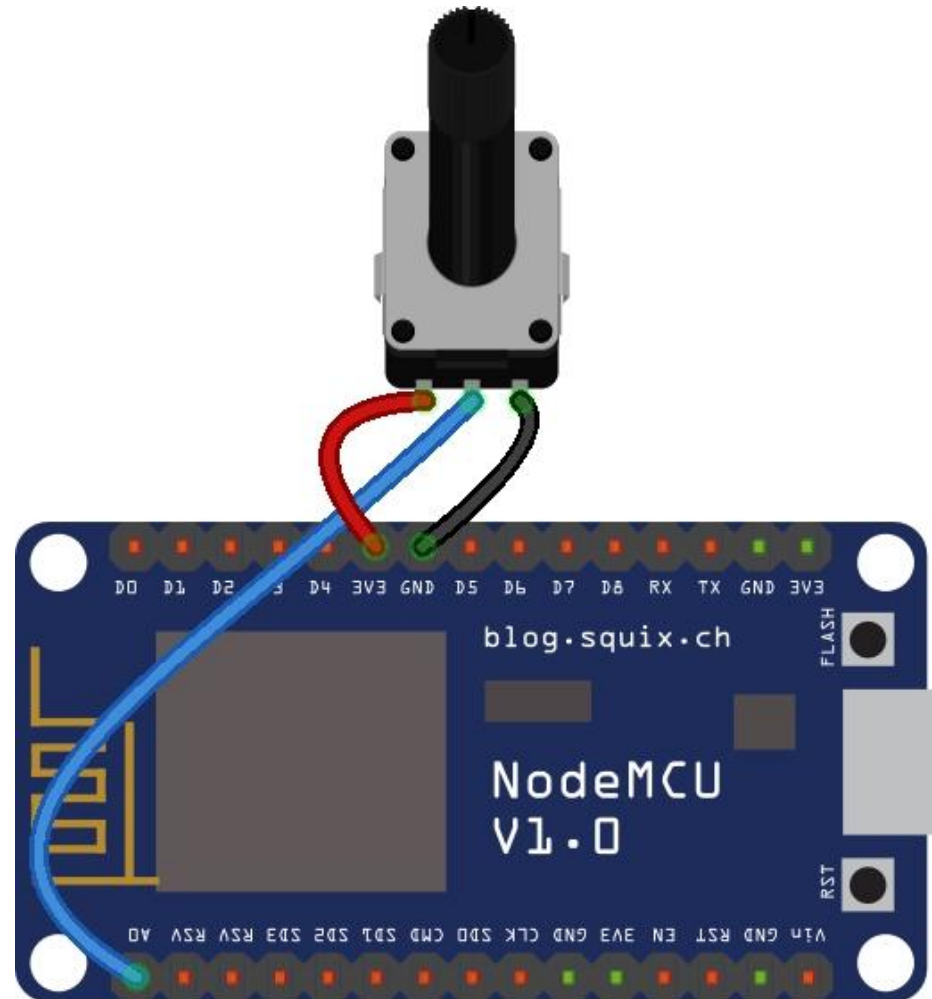


# Practice Exercises02

- ADC : Analog Digital Converter
  - Converting the external power value (analog value) of the rheostat to the digital value (0 ~ 1023) and output it analog pin (A0),  
`analogRead()`

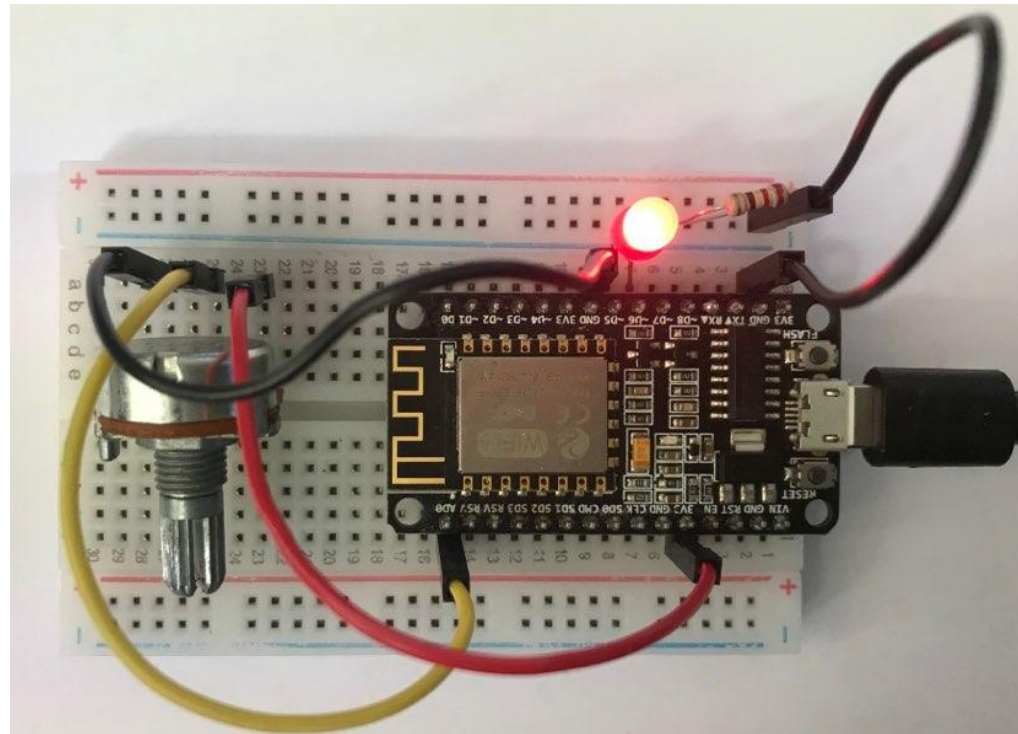


# Practice Exercises 02 : CircuitWiring



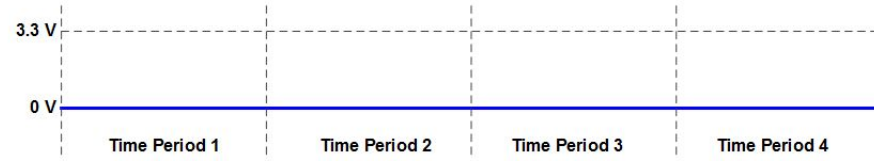
# Practice Exercises 03

- PWM(Pulse Width Modulation), LED brightness with rheostats Adjust
  - Analog (A0) Pin, analogRead(), analogWrite()

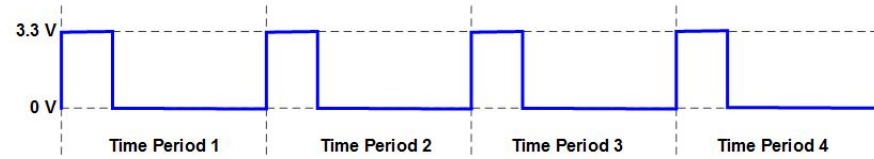


# PWM(Pulse Width Modulation)

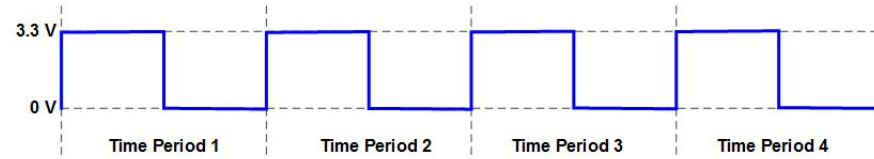
```
analogWrite(LedPin, 0);
```



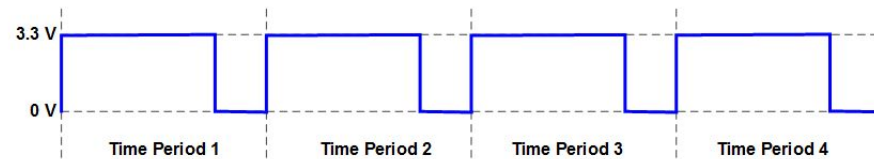
```
analogWrite(LedPin, 255);
```



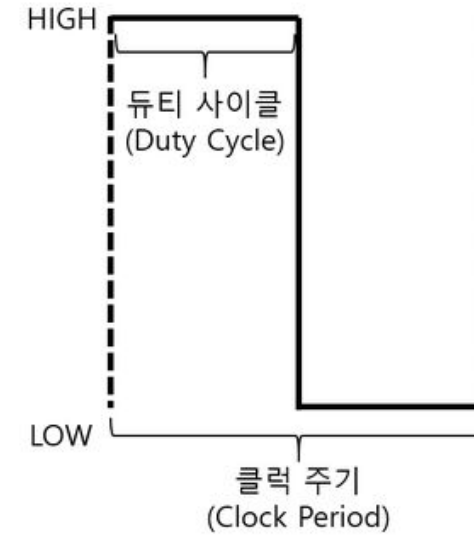
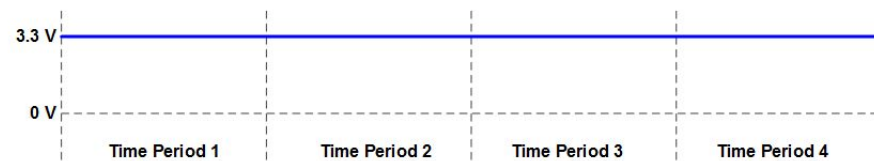
```
analogWrite(LedPin, 512);
```



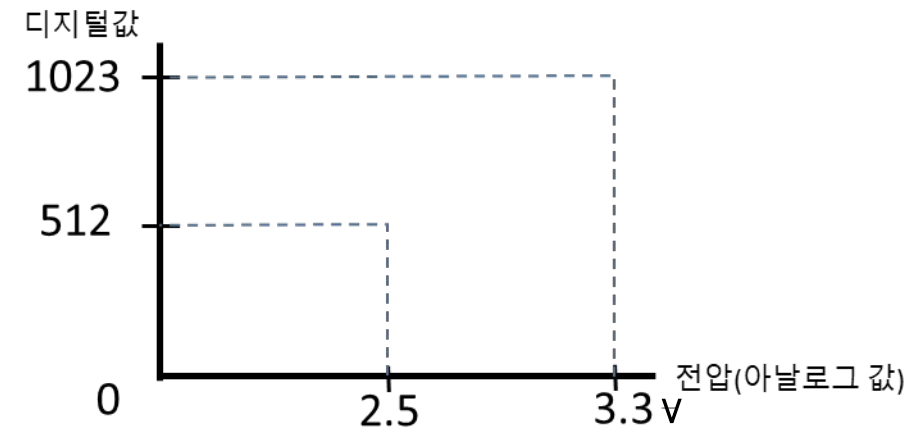
```
analogWrite(LedPin, 767);
```



```
analogWrite(LedPin, 1023);
```

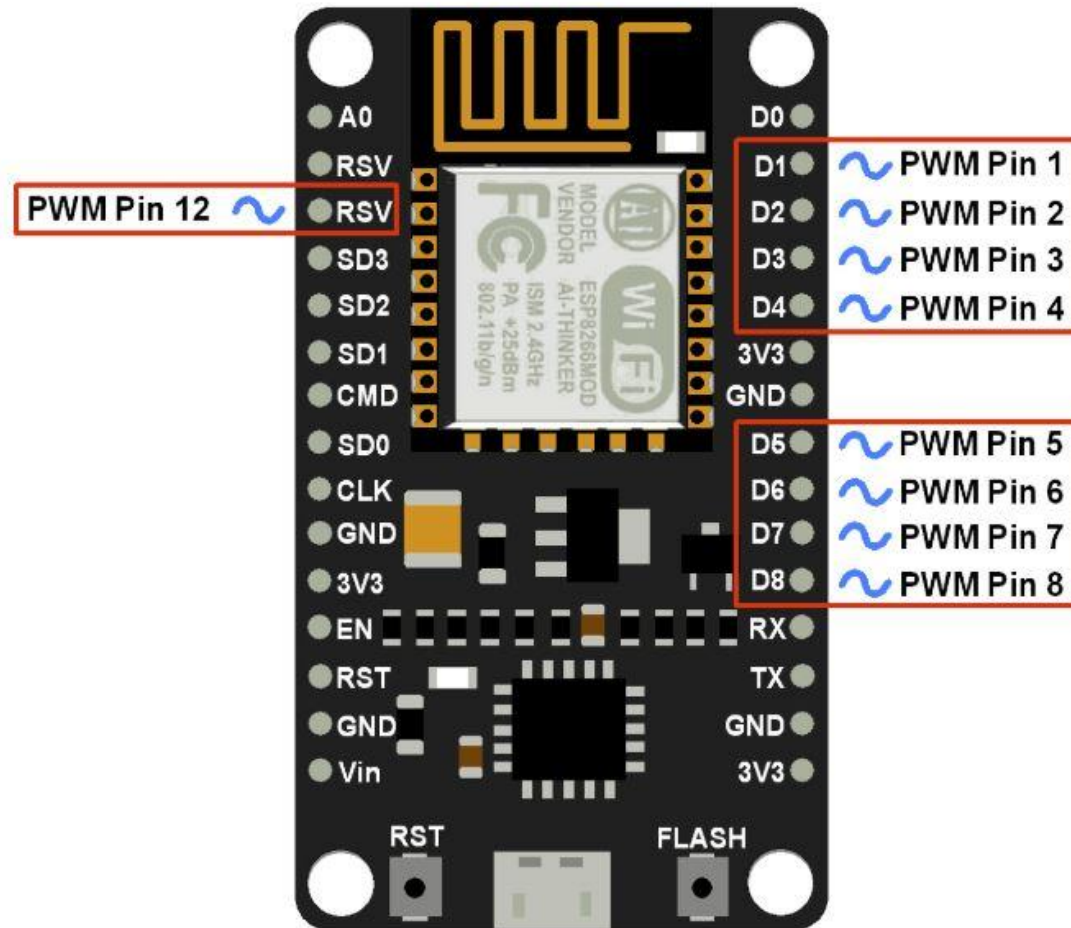


$$\text{디지털 값} = \frac{3.3}{1023} \times \text{아날로그 핀 값 (Duty Cycle)}$$





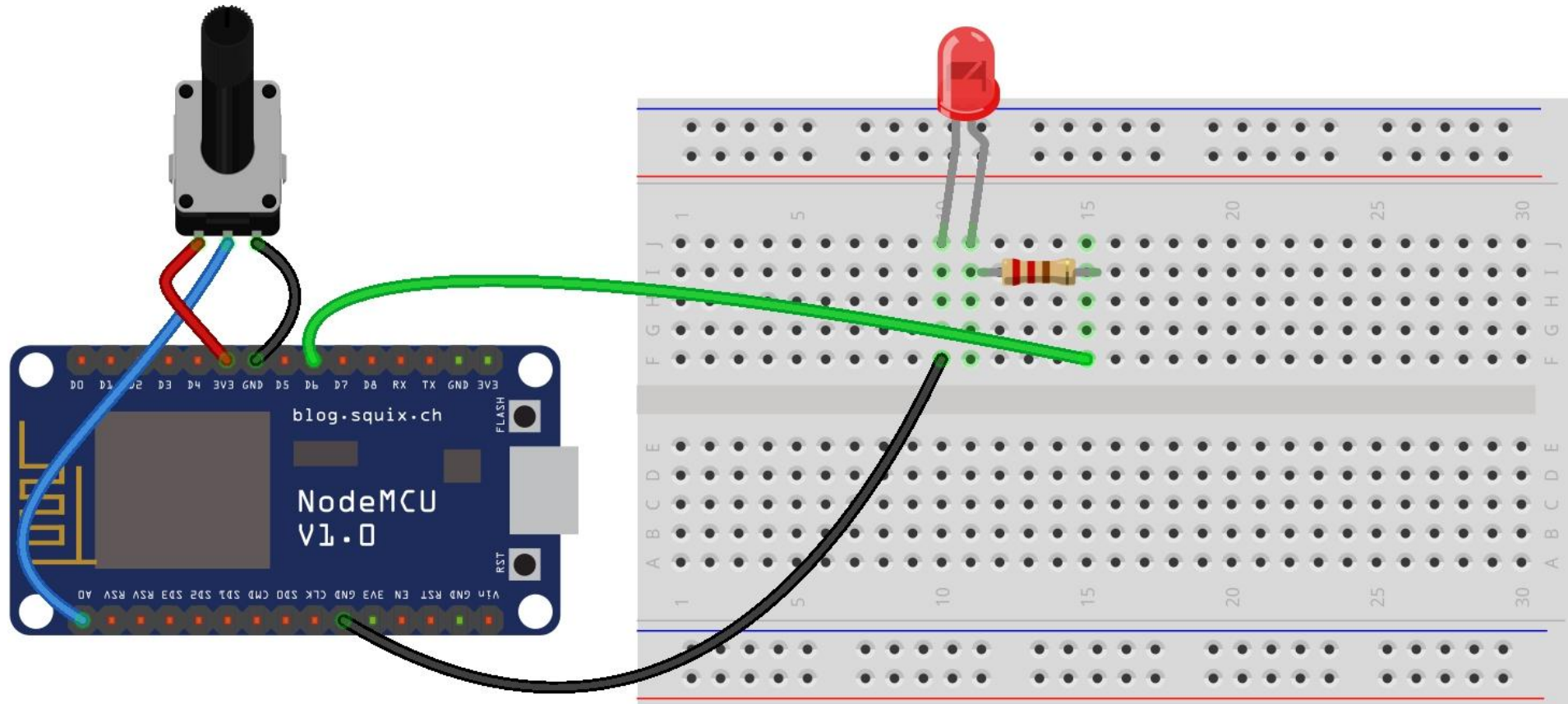
# Node MCU PWM pin





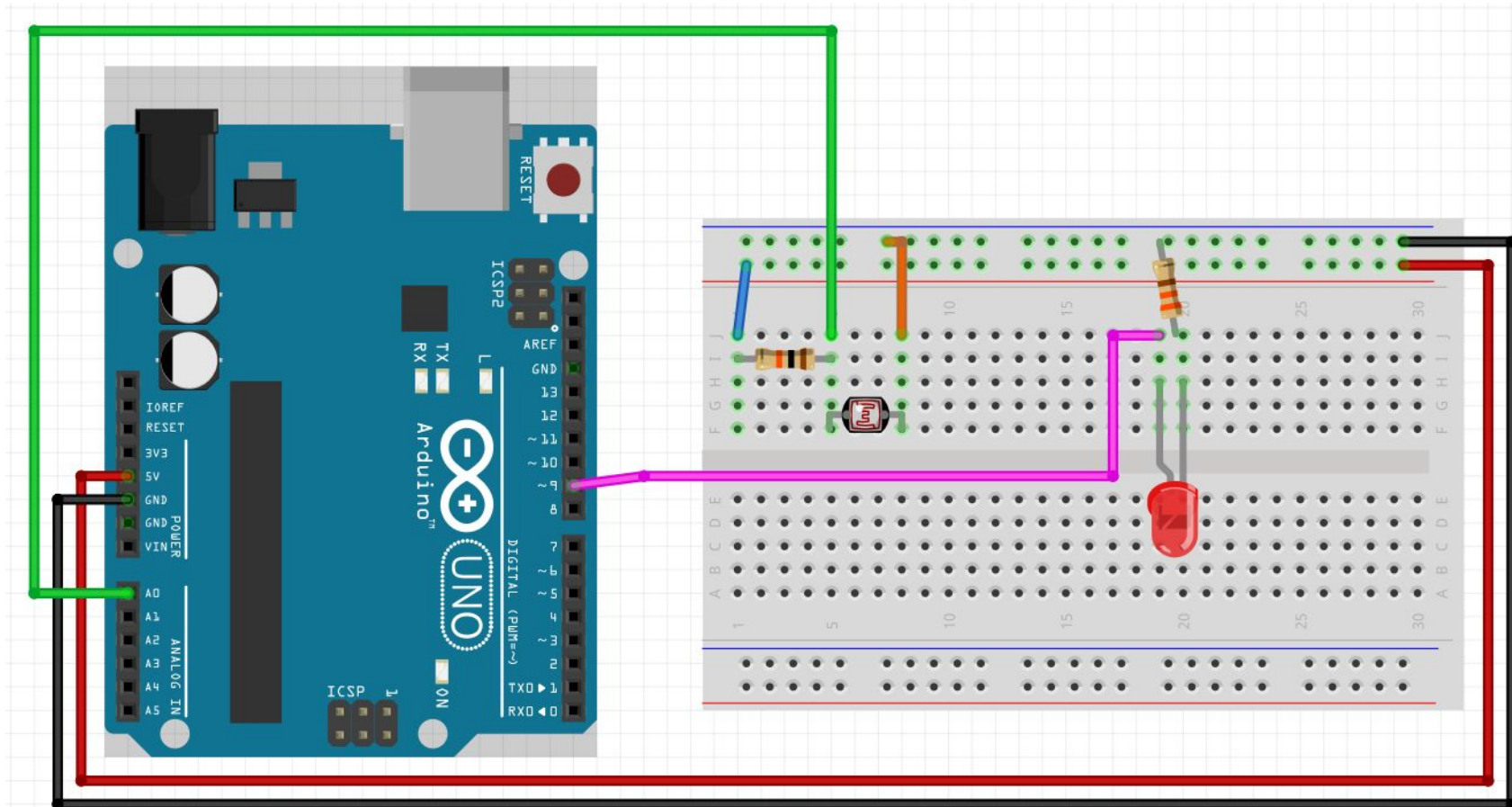
# Practice Exercises03 : Circuit Wiring

- A0 : Analog Pin
- D6(GPIO12) : PWM



# Practice Lesson 04: Adjusting LED Brightness Based on Light Intensity

- under Fritzing(Ardunio UNO) to NodeMCU Design
  - Photoresistor, LED, 2 resistors



# Reference videos

- NodeMCU Build a development environment  
<https://youtu.be/YNOSQTk29DE>
- NodeMCU Take control
  - <https://youtu.be/nBc-2Wb49wl>
  - <https://developer.ibm.com/kr/cloud/internet-of-things/2017/07/30/esp8266-iot-arduino-ide-nodemcu-basic/>
  - <https://www.instructables.com/id/NodeMCU-Basic-Project-Blink-a-LED/>