

Basic C programming with ESP8266 and Arduino IDE

Lecture 5: 01205479 IoT for EE
Dr.Varodom (Dew) Toochinda

Outline

- Overview
- Arduino-based Hardware Examples
- Arduino IDE
- ESP8266 Arduino Core Installation
- Additional Libraries used in the course
- CP210x USB Driver Installation
- Indoor Greenhouse Regulator (IGR) Hardware
- ESP8266 Labs

Some Arduino-based Hardware Examples

Arduino UNO

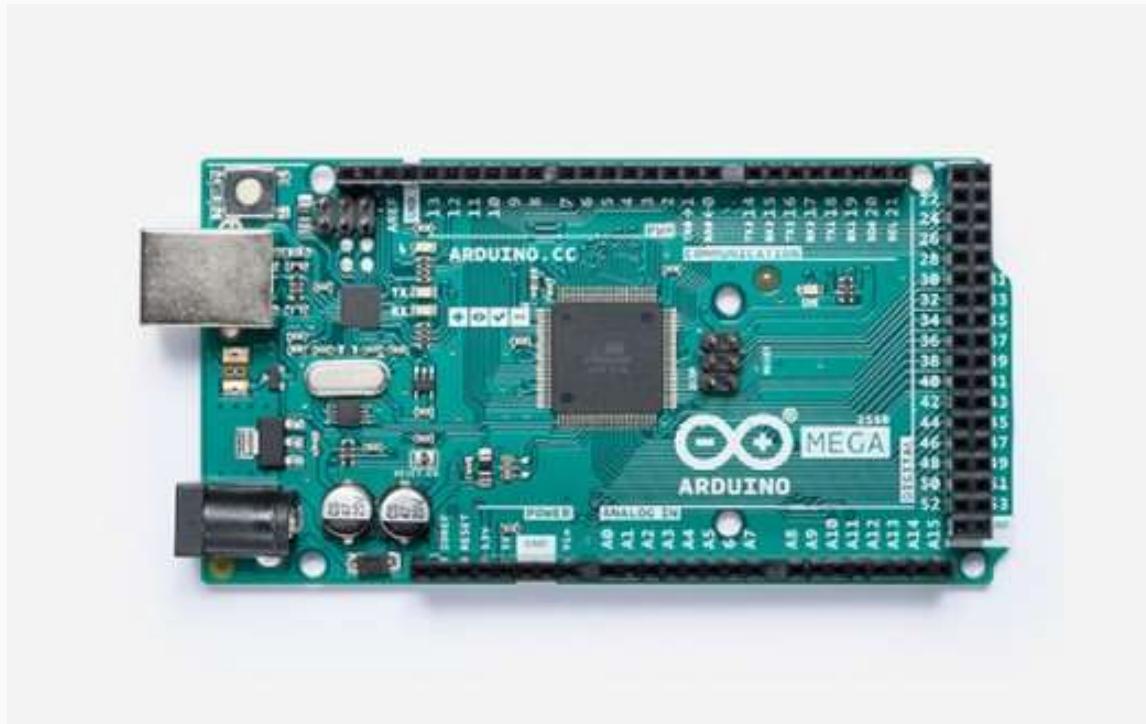
- MCU: ATmega328P (8-bit)
- Low performance (16 MHz)
- Limited resources : 32 KB Flash, 2 KB SRAM
- Require communication shield for IoT applications
 - WiFi
 - NB-IoT



Operating Voltage: 5 Volts

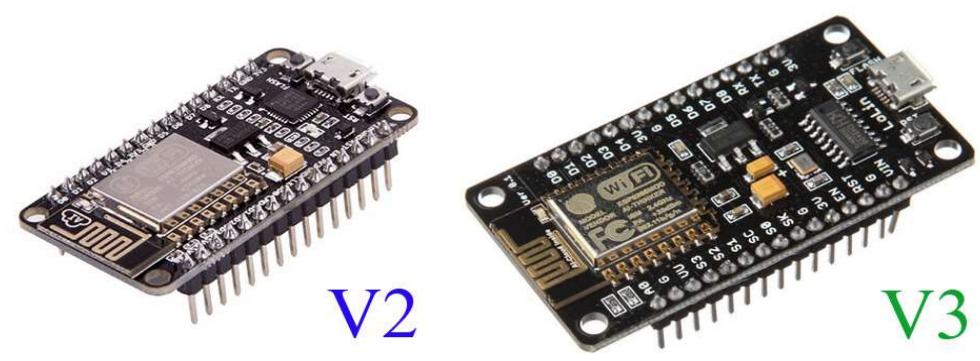
Arduino MEGA

- MCU: ATmega2560 (8-bit)
- More resources than UNO
- Still low performance (16 MHz clock)
- Still require additional hardware for IoT



NodeMCU (ESP8266)

- Espressif L106 (32-bit) based on Tensilica/Xtensa
- Based clock speed 80 MHz
- 4 MB Flash
- 16 GPIOs
- 1 ADC (max. voltage 1 Volt)
- WiFi on chip (IEEE 802.11 b/g/n)



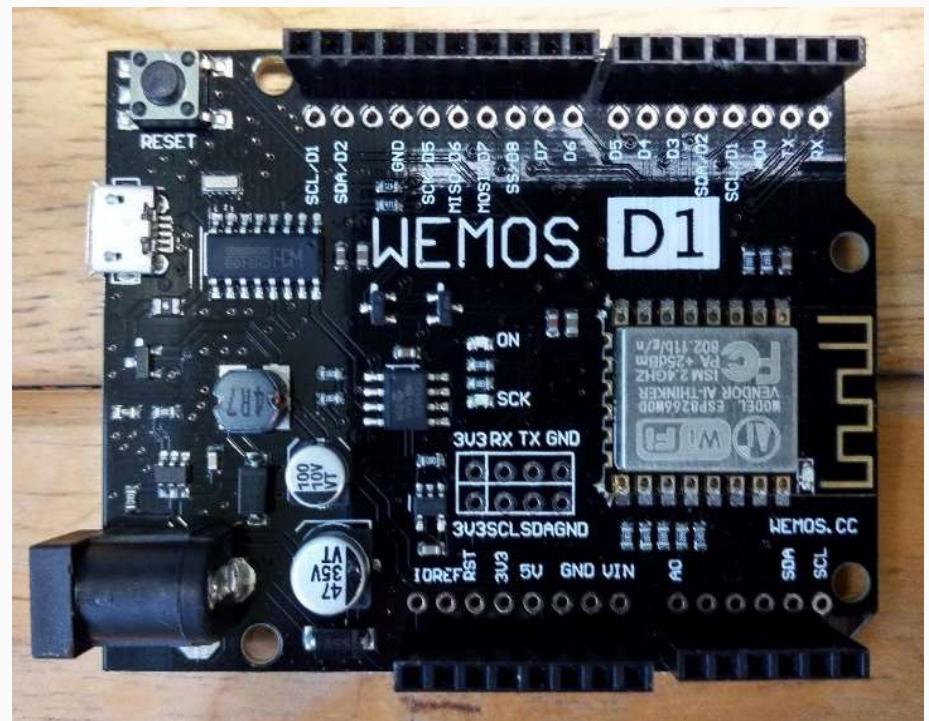
V2

V3

Operating Voltage : 3.3 Volts

WEMOS D1 R2

- ESP8266 in Arduino Form Factor
- Pins are NOT completely compatible with Arduino UNO!



ESP32

- Espressif LX6 (32-bit, 2 processing cores) based on Tensilica/Xtensa
- Clock speed 160 MHz/240 MHz
- 34 GPIOs
- 18 ADC pins
- 2 DAC pins (8-bit)
- WiFi + Bluetooth on chip
- Touch and hall sensors



ESP32 Dev Kit
V1

NodeMCU
ESP-32S

WEMOS
Lolin 32

Arduino IDE

Software for Embedded and IoT Programming

Download and install latest Arduino IDE

Downloads



Arduino IDE 2.3.2

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

[SOURCE CODE](#)

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

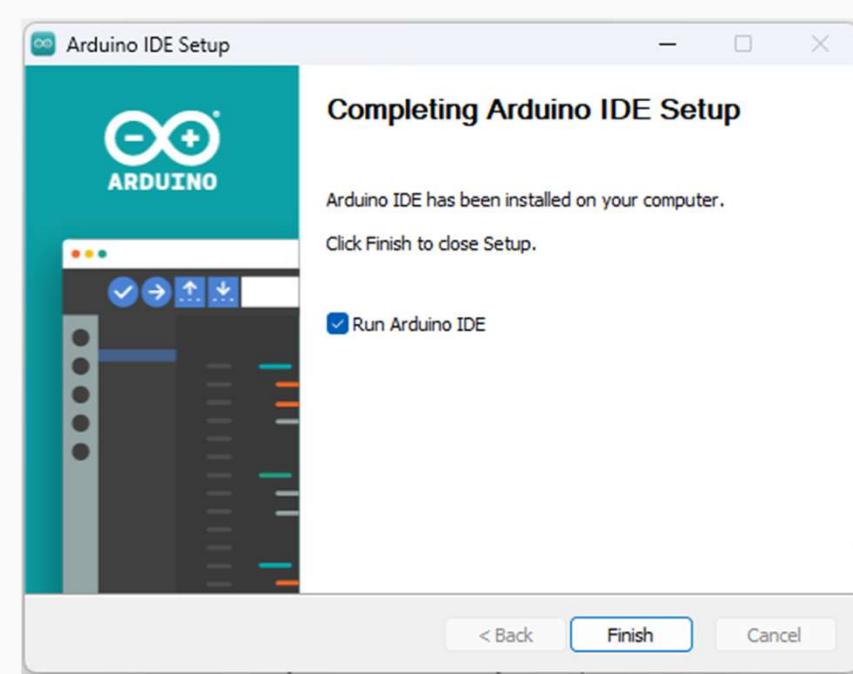
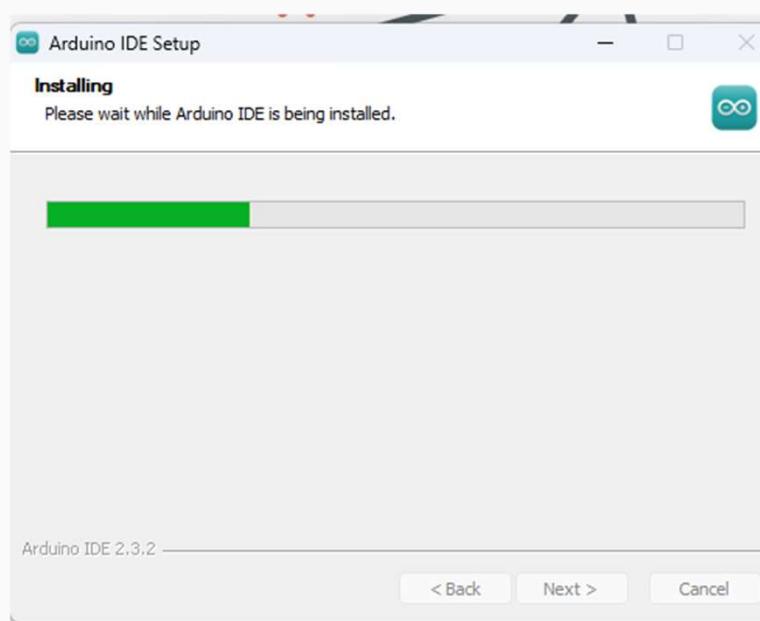
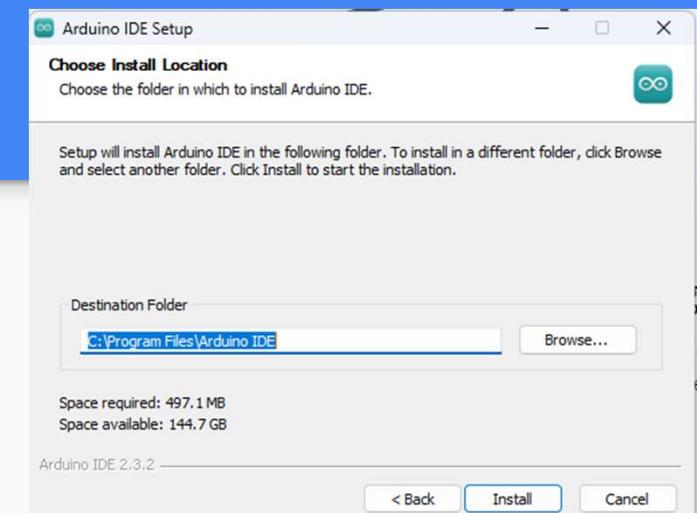
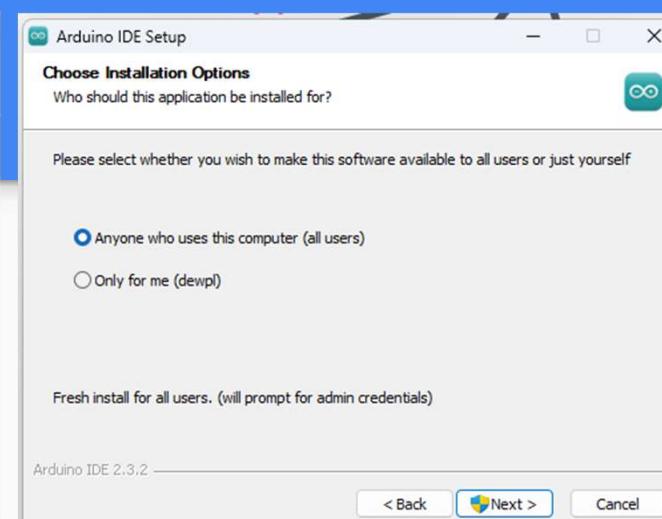
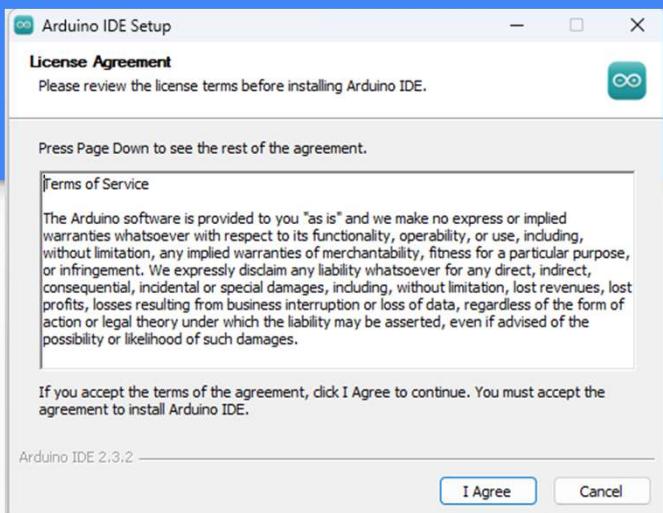
macOS Intel, 10.15: "Catalina" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

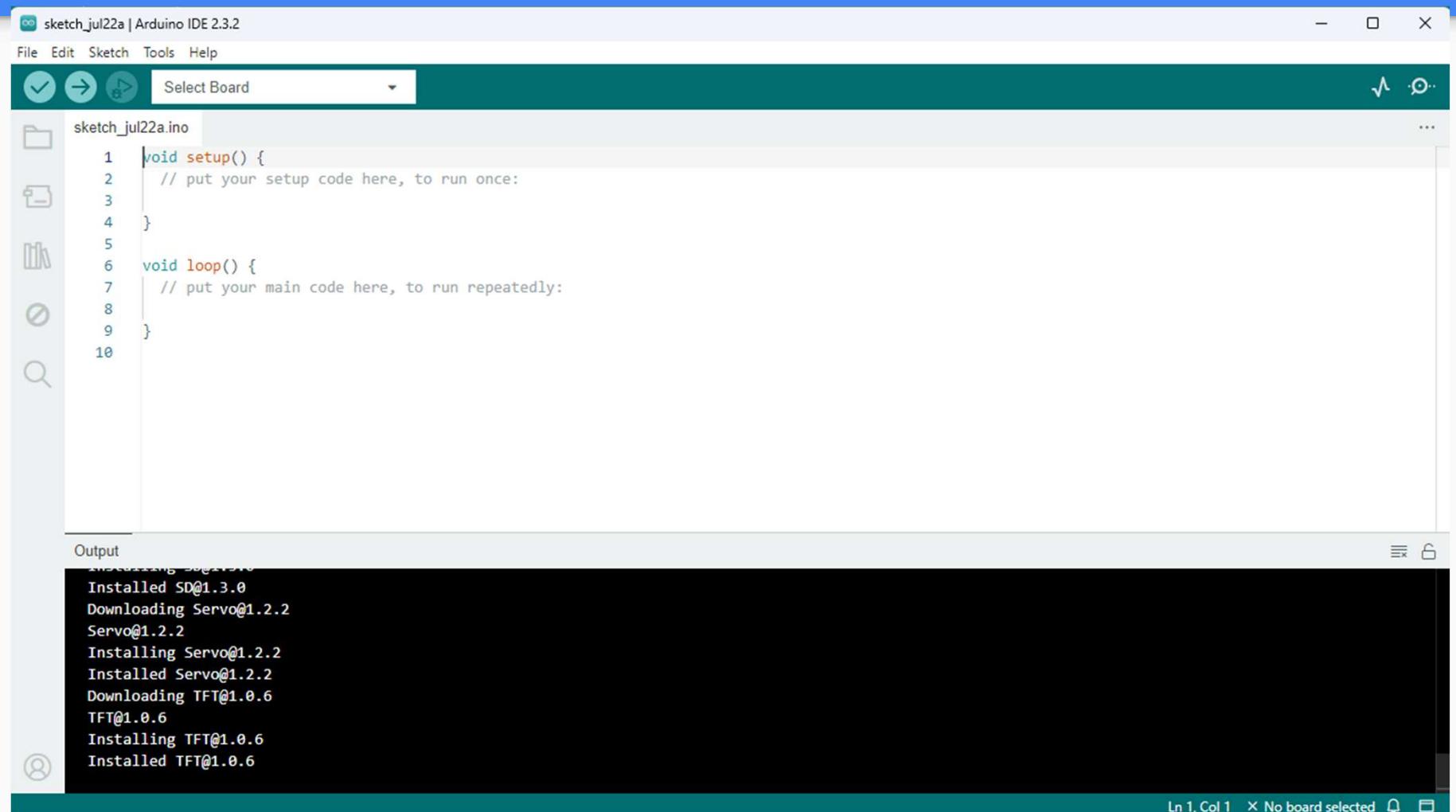
<https://www.arduino.cc/en/software>

As of July 2024

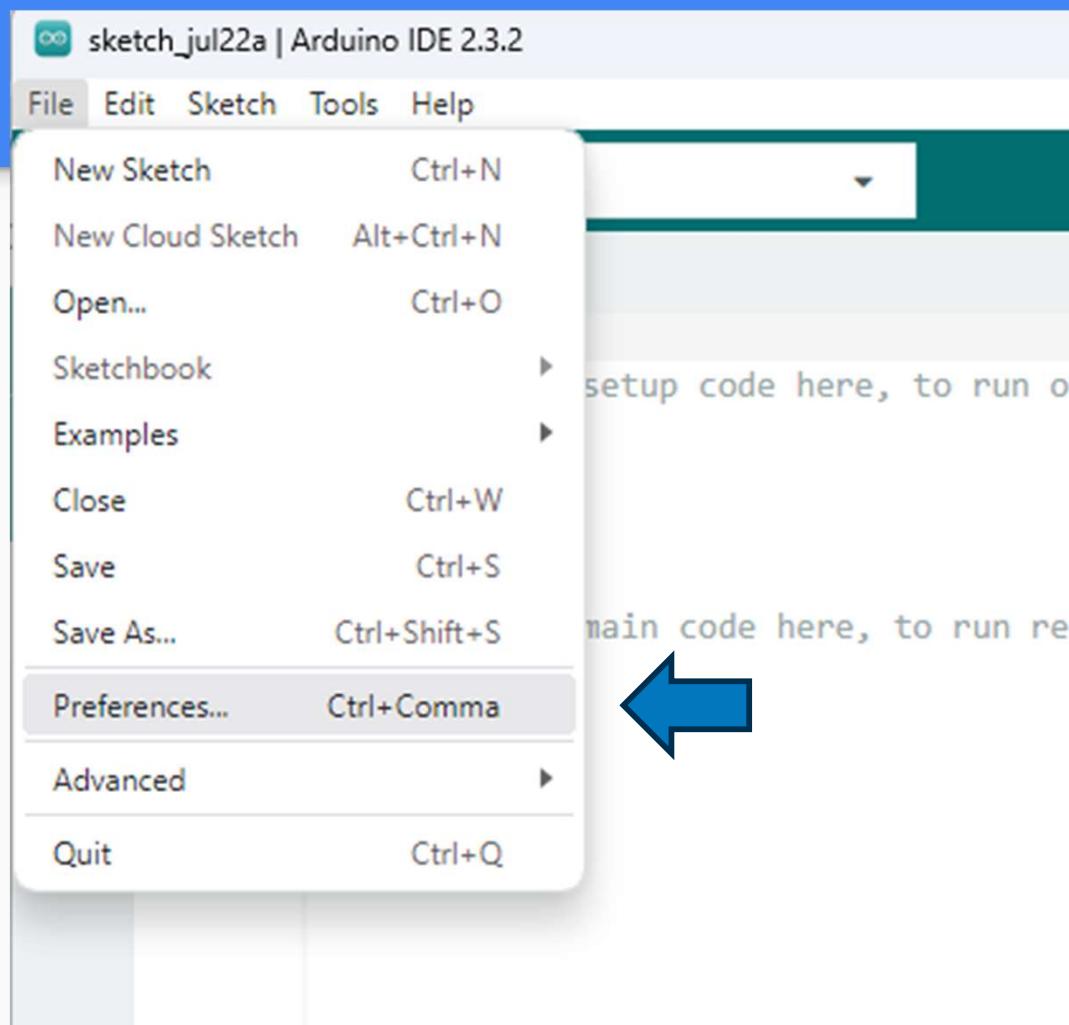
Windows installation screens



Launching Arduino IDE for the first time



Install ESP8266 Arduino Core



Install ESP8266 Arduino Core

Preferences X

Settings Network

Sketchbook location: BROWSE

Show files inside Sketches

Editor font size:

Interface scale: Automatic %

Theme:

Language: (Reload required)

Show verbose output during compile upload

Compiler warnings

Verify code after upload
 Auto save
 Editor Quick Suggestions

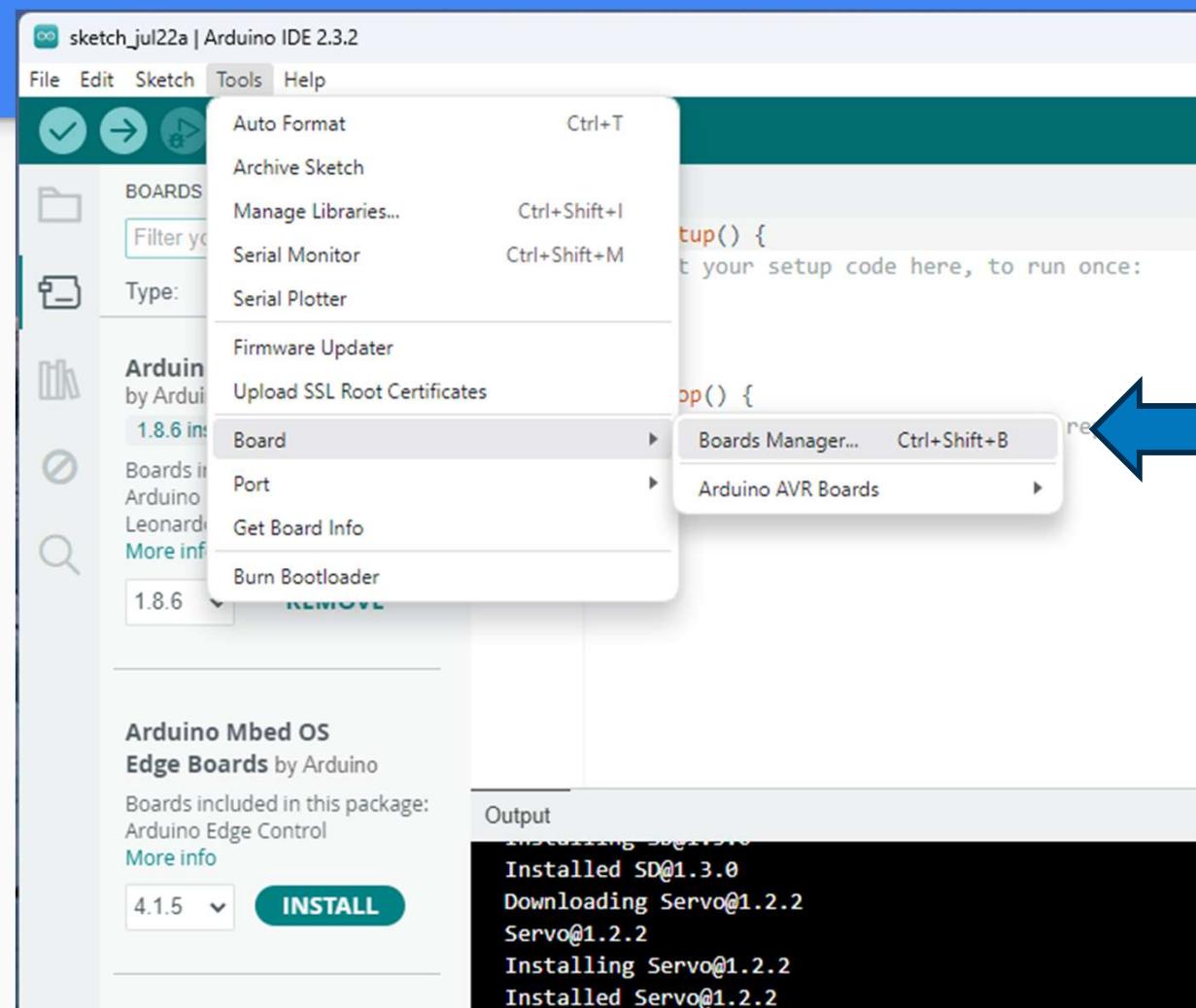
Additional boards manager URLs: +

CANCEL OK

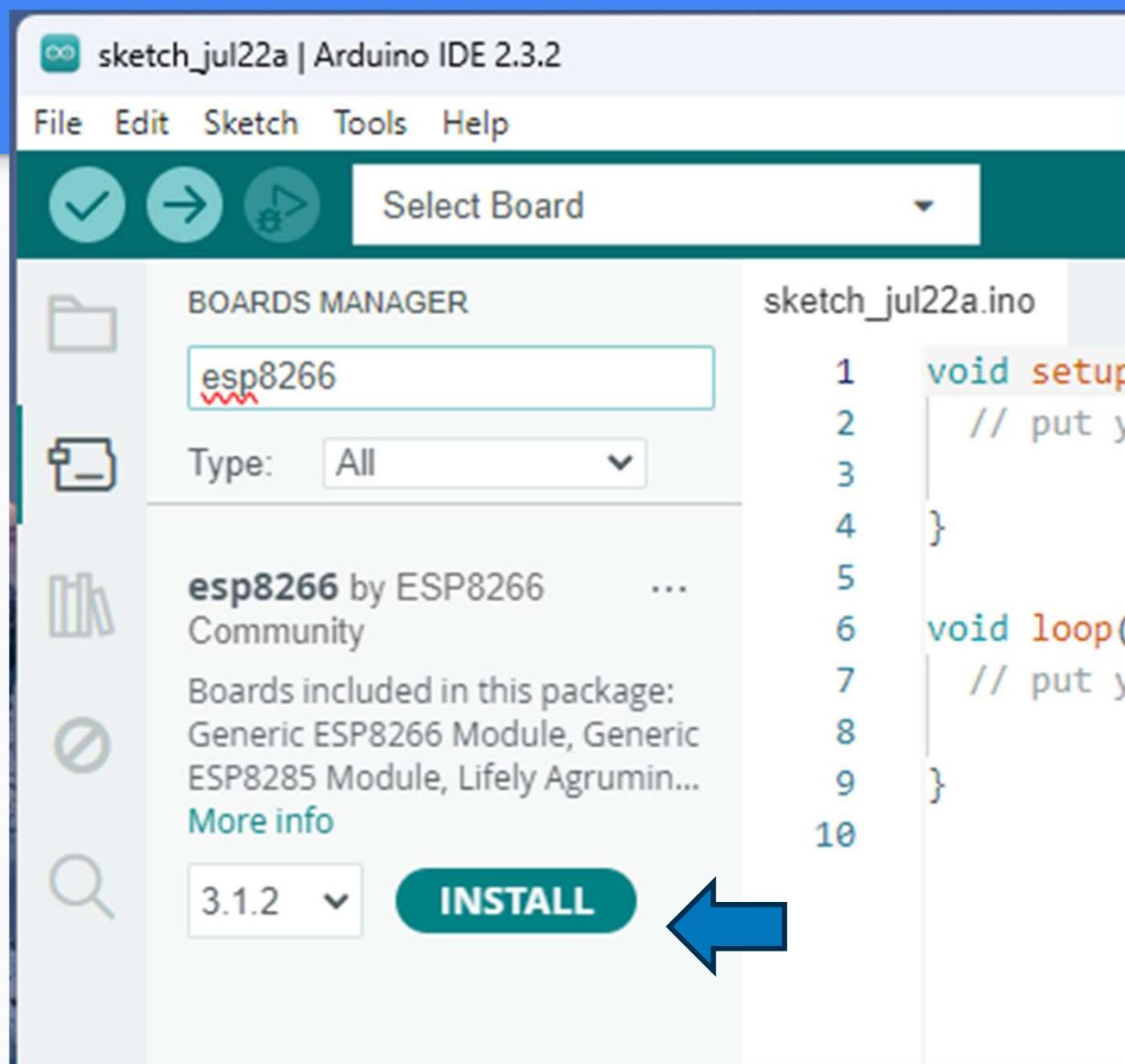


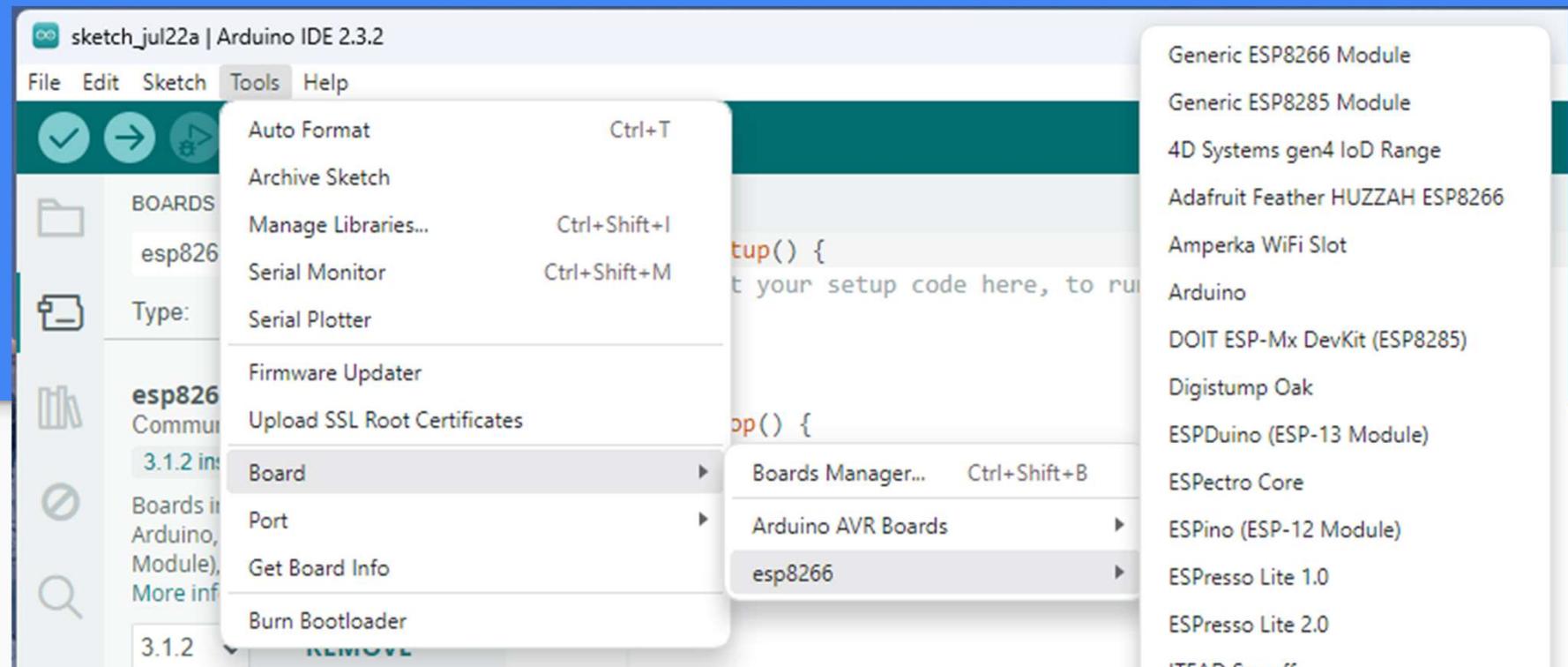
http://arduino.esp8266.com/stable/package_esp8266com_index.json

Install ESP8266 Arduino Core



Install ESP8266 Arduino Core





Select NodeMCU ESP8266 Board

```
Output
Installing esp8266:mklittlefs@3.1.0-gcc10.3
Configuring tool.
esp8266:mklittlefs@3.1.0-gcc10.3-e5f9fec
Installing esp8266:python3@3.7.2-post1
Configuring tool.
esp8266:python3@3.7.2-post1 installed
Installing platform esp8266:esp8266@3.1.2
Configuring platform.
Platform esp8266:esp8266@3.1.2 installed
```

Install Additional Libraries

- DHT Library and Adafruit Unified Sensor
- Adafruit GFX Library
- Adafruit SSD1306
- BH1750FVI by PeterEmbedded

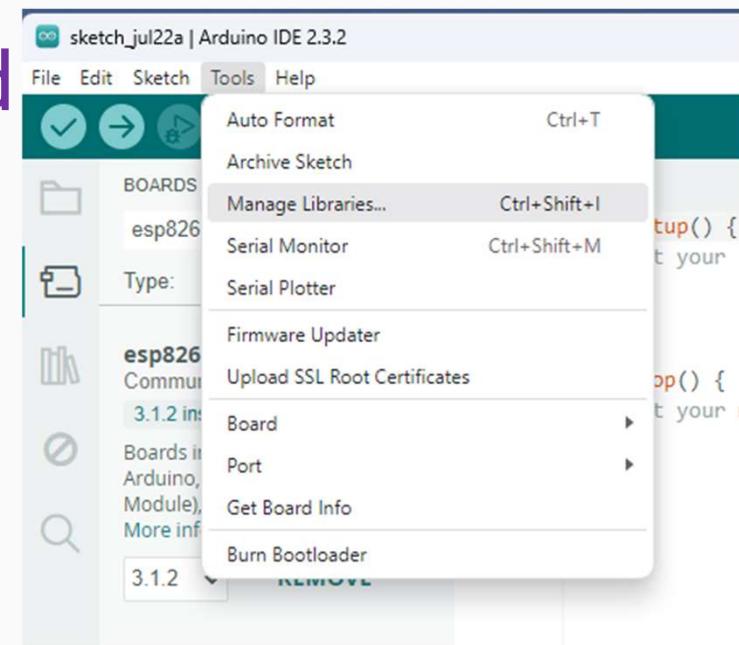
For light sensor



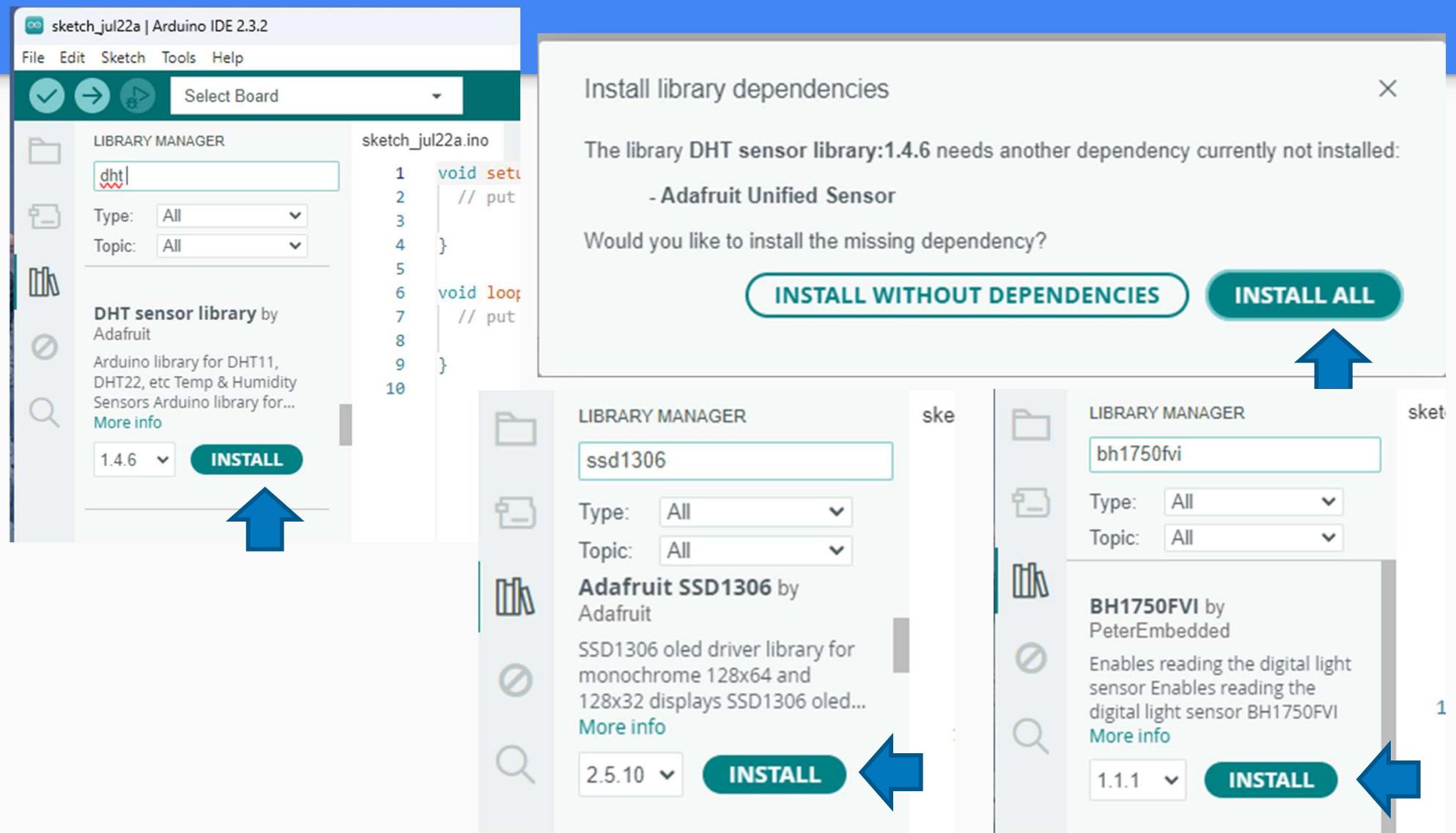
For DHT sensors



For OLED display



Install Additional Libraries



Install CP2102 USB Driver

<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads>

Today			
 CP210xVCPIinstaller_x64	7/22/2024 9:52 AM	Application	1,026 KB
 CP210xVCPIinstaller_x86	7/22/2024 9:52 AM	Application	901 KB
 dpinst.xml	7/22/2024 9:52 AM	xmlfile	12 KB
 ReleaseNotes	7/22/2024 9:52 AM	TXT File	11 KB
 SLAB_License_Agreement_VCP_Windows	7/22/2024 9:52 AM	TXT File	9 KB
 slabvcp	7/22/2024 9:52 AM	Security Catalog	12 KB
 slabvcp	7/22/2024 9:52 AM	Setup Information	5 KB
 x64	7/22/2024 9:52 AM	File folder	
 x86	7/22/2024 9:52 AM	File folder	



Structure of Arduino Sketch

sketch_jul22a | Arduino IDE 2.3.2

File Edit Sketch Tools Help

Select Board

LIBRARY MANAGER

bh1750fvi

Type: All

Topic: All

BH1750FVI by PeterEmbedded
Enables reading the digital light sensor
Enables reading the digital light sensor BH1750FVI
More info

1.1.1 **INSTALL**

BH1750 by Christopher Laws
Arduino library for the digital light sensor breakout boards containing the BH1750FVI IC...
More info

1.3.0 **INSTALL**

sketch_jul22a.ino

```
1 int i = 0;           ← Global variables
2
3 void setup() {
4     // put your setup code here, to run once:
5
6 }
7
8 void loop() {
9     // put your main code here, to run repeatedly:
10
11 }
12
13 // other user functions, if any
14 int myfunc()
15 {
16
17 }
```

Output

Example : addmult.ino

- Use Serial Monitor to accept input and print output
- Commands
 - add=x : add x to result
 - mult=y : mult y to result

The screenshot shows the Arduino IDE interface. The main window displays the code for `addmult.ino`. The code reads commands from the serial port, processes them, and prints the results to the serial monitor. The serial monitor window shows the following interaction:

```
mult=2.4
add previous result by 30.00
New result = 30.00
multiply previous result by 4.00
New result = 120.00
add previous result by 450.00
New result = 570.00
```

The code itself includes functions for handling commands like `add` and `mult`, and a `cmdInt` function for interpreting the received strings.

```
addmult
float parmvalfloat;
bool newcmd = 0; // flag when new command
int sepIndex; // index of separator
bool noparm = 0; // flag if no parameter
float Result = 0;

void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
  if (Serial.available() > 0) { // data available
    rcvdstring = Serial.readString();
    newcmd = 1;
  }
  if (newcmd) { // execute this part
    cmdInt(); // invoke the interpretation
    Serial.print("New result = ");
    Serial.println(Result);
    newcmd = 0;
  }
  delay(100);
}

void cmdInt(void)
{
  rcvdstring.trim(); // remove leading&trailing whitespace, if any
  // find index of separator (blank character)
  sepIndex = rcvdstring.indexOf('=');
}
```

Note: appearance changes slightly in newer versions

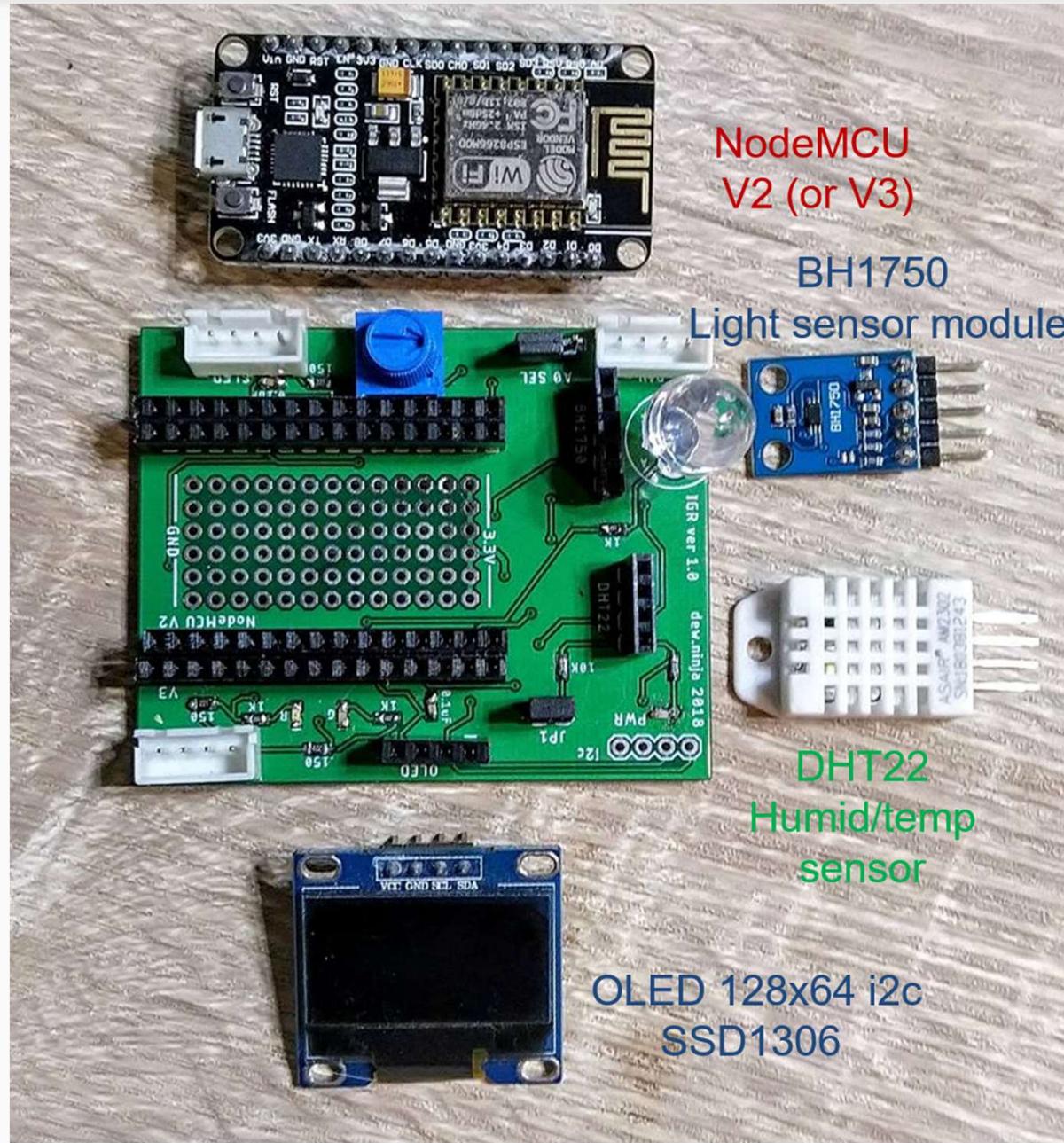
Wokwi diagram :

<https://wokwi.com/projects/404633623147534337>

Exercise 1

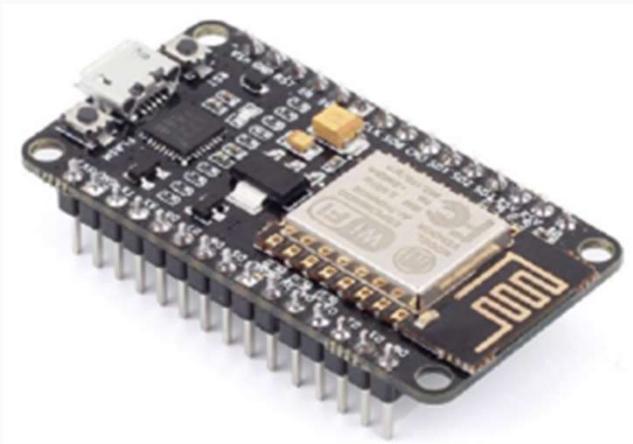
- Save addmult.ino to addmultdiv.ino
- Append command div=z to divide previous result by z
- Fix divide by zero problem
- Add a command to turn off/on LED at D5

Indoor Greenhouse Regulator (IGR) hardware



NodeMCU-12E (NodeMCU V2)

ESP8266-12E (ESP8266EX)



- MCU 32 BIT
- 4 Mbyte flash memory
- 112 kByte RAM
- Onboard WIFI module
- CP2109 USB 2 Serial
- Power supplied from USB
- Program upload via USB
- 16 GPIO's
- 1 Analog pin

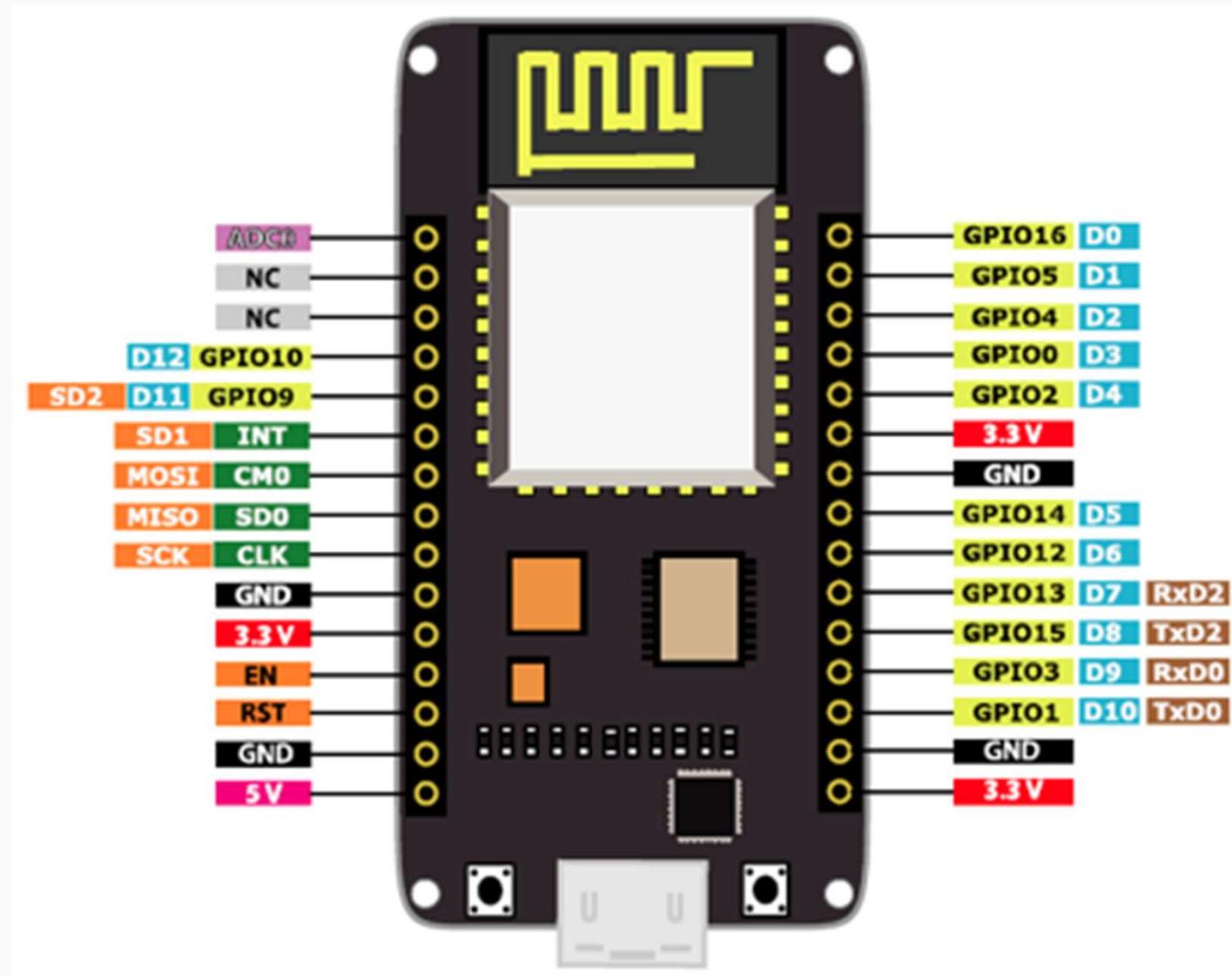
DHT11

DHT11 (Humidity and Temperature Sensor)

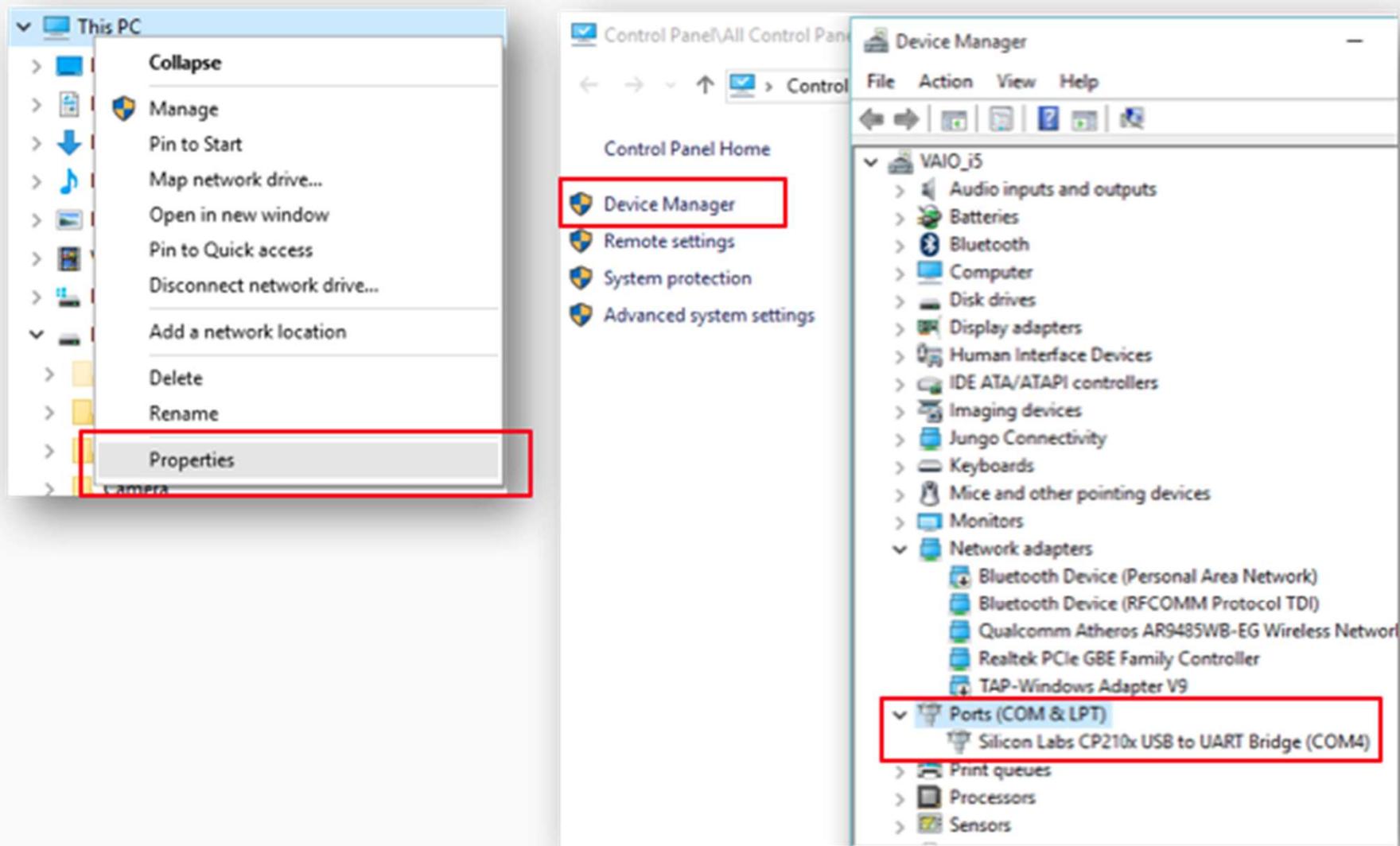


- Humidity range 20-90% RH with +/-5% RH accuracy, 1 % resolution, 8-bit output
- Temperature 0-50 Celsius +/-2 C accuracy, 1 C resolution, 8-bit output
- Consume 0.5-2.5 mA current (during measurement) at 3 - 5.5 VDC
- 1 second sampling period (min)

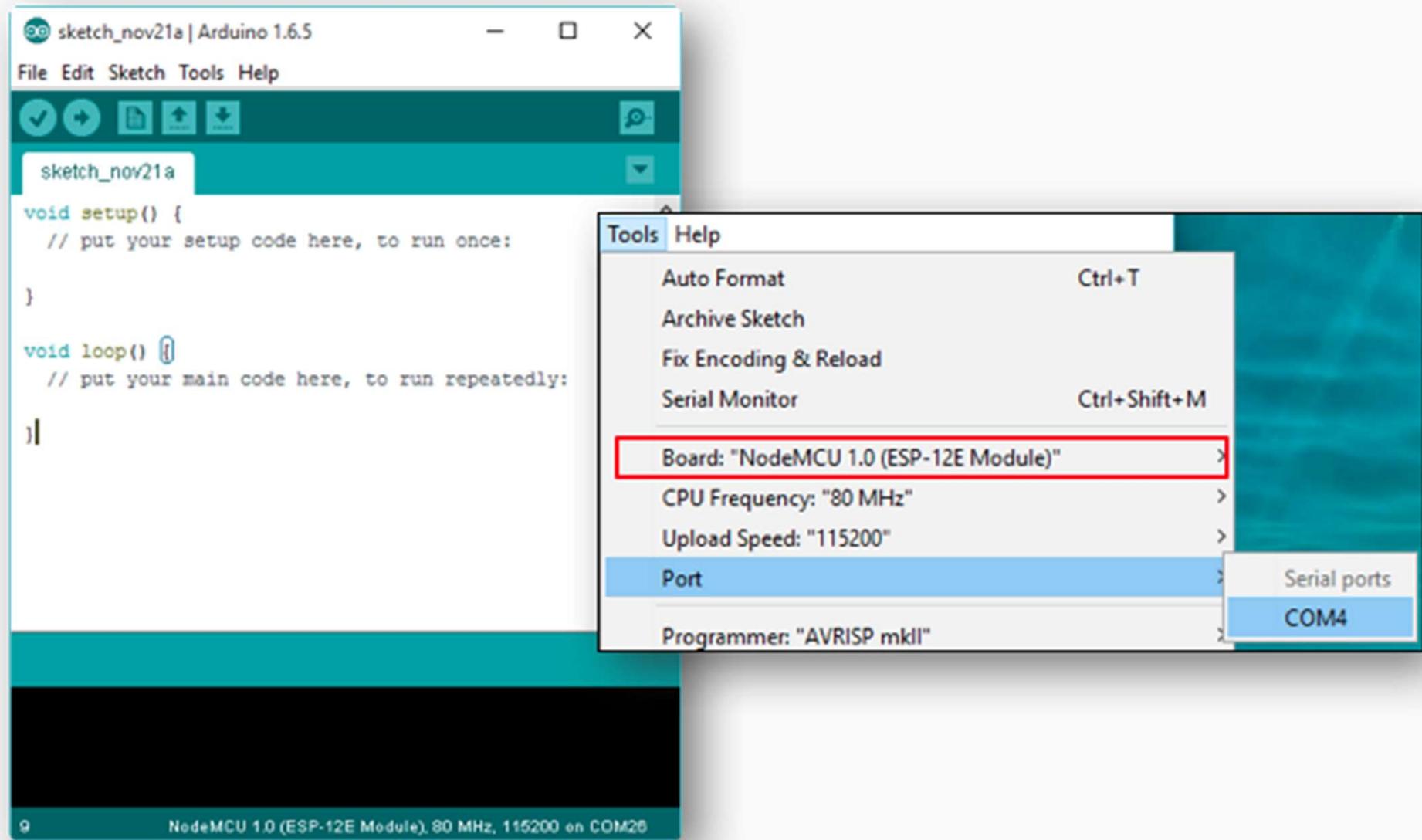
NodeMCU V2 pinout



Check connection



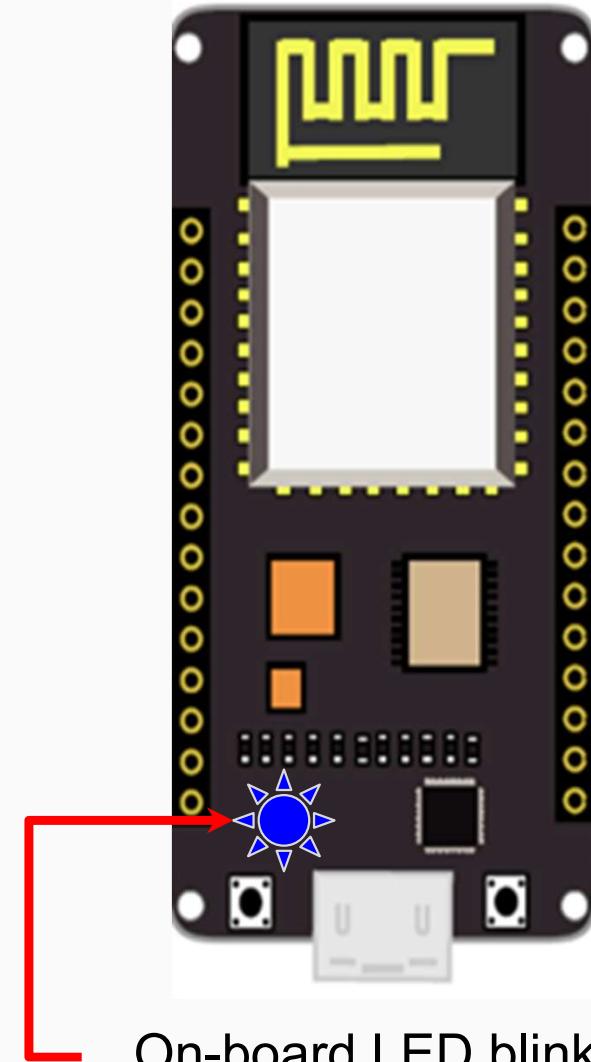
Arduino IDE



Note: appearance changes slightly in newer versions

LAB 5-1-Blink

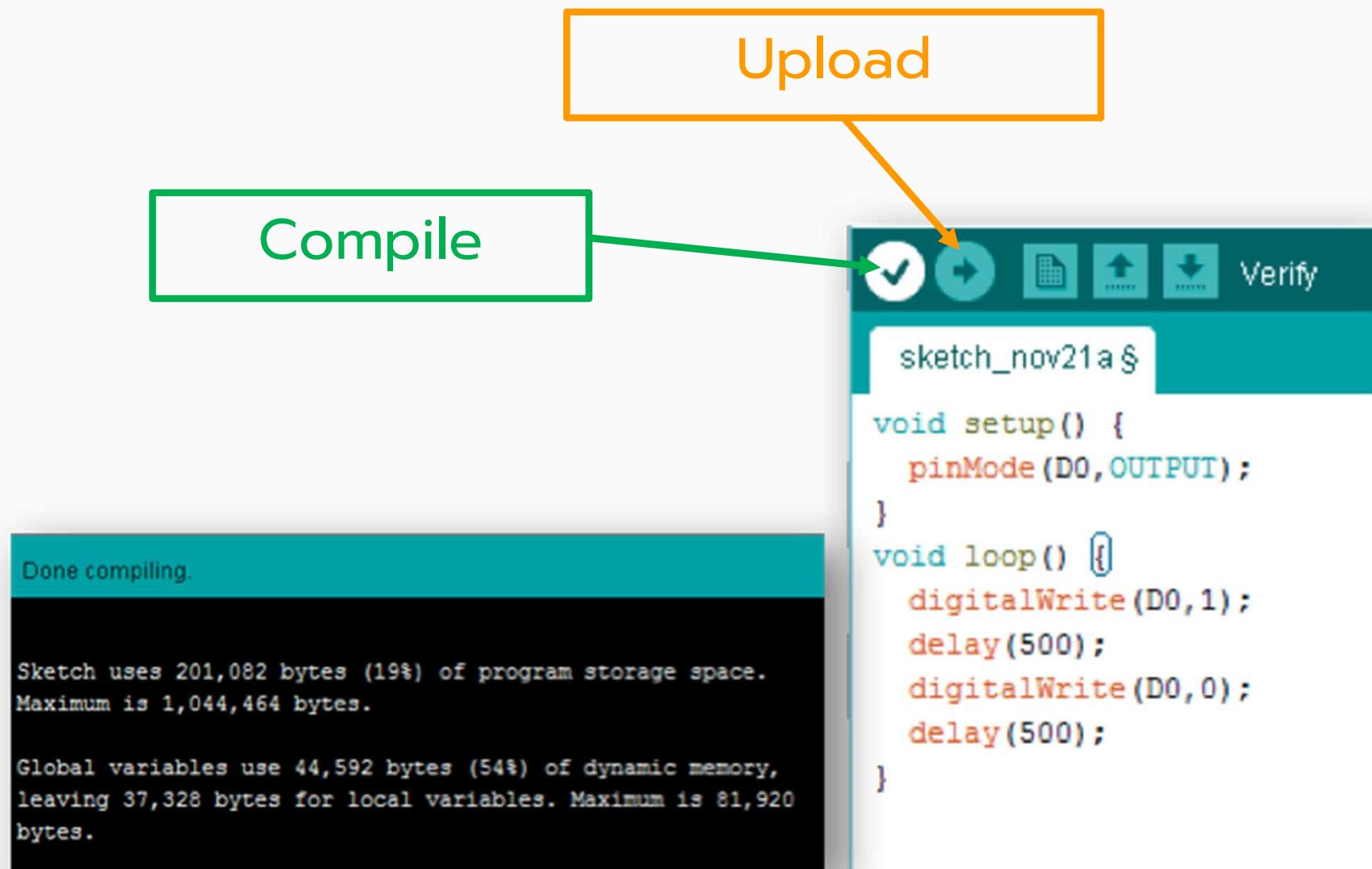
```
void setup() {  
    //LED_BUILTIN คือตัวแปรค่าเริ่มต้น สำหรับ LED  
    //ประจำแต่ละบอร์ด  
    pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(LED_BUILTIN, HIGH);  
    delay(500);  
    digitalWrite(LED_BUILTIN, LOW);  
    delay(500);  
}
```



On-board LED blinking

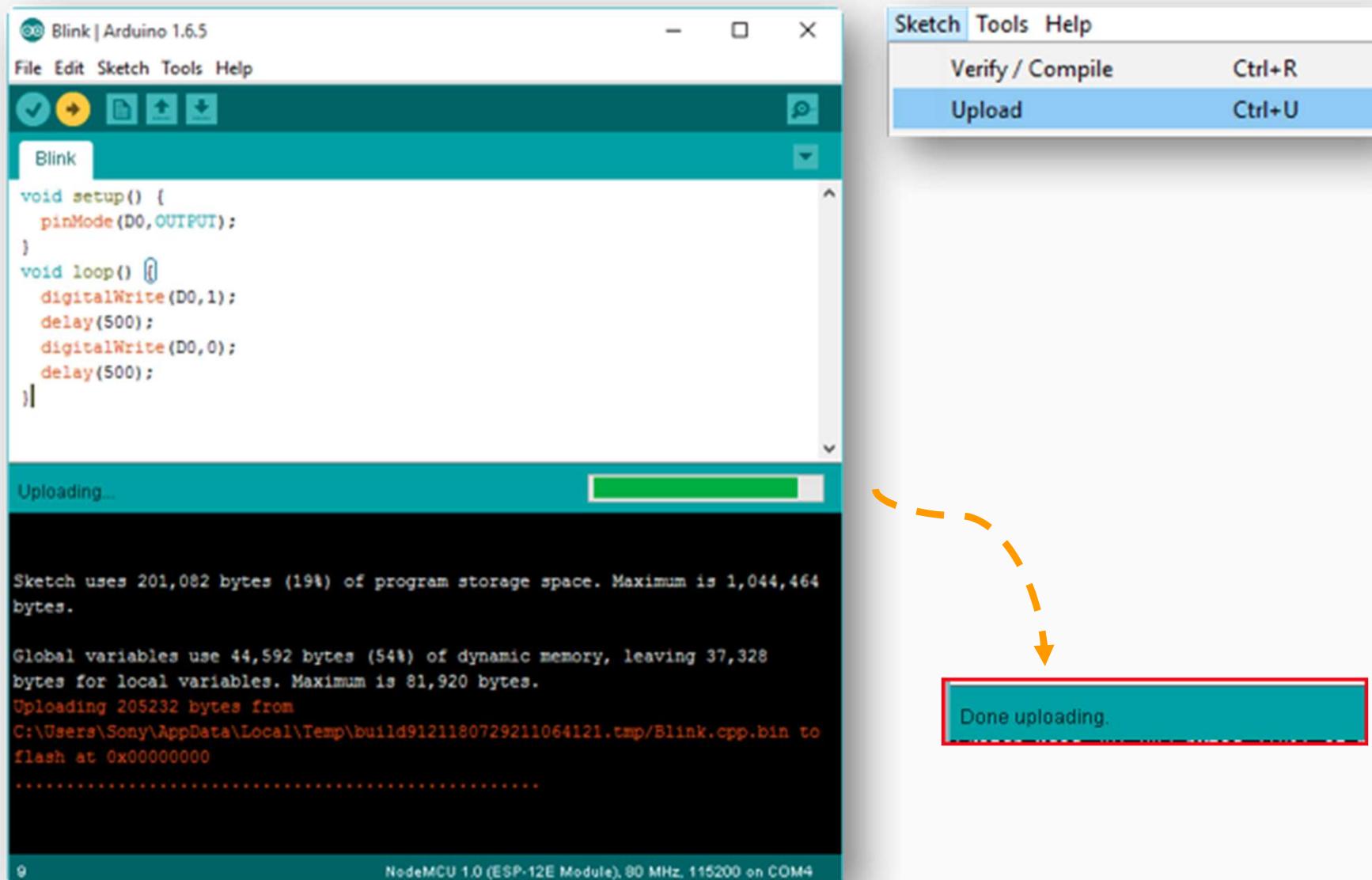
Remark: for NodeMCU V3
Onboard LED connected to D4

Program Compile and Upload



Note: appearance changes slightly in newer versions

Program Upload



Note: appearance changes slightly in newer versions

Input/output pin assignment

pinMode (PIN, MODE) ;

PIN : D0 - D10 (or use GPIO number)

MODE

OUTPUT : set pin to output

INPUT : set pin to input

INPUT_PULLUP : set pin to input

with internal pullup resistors

Example :

pinMode (D0, OUTPUT) ;

Send digital output

digitalWrite(PIN, Logic);

PIN : pin number

Logic

HIGH : logic 1 (3.3 Volts)

LOW : logic 0 (GND)

Example :

digitalWrite(D0, HIGH);

Delay period

delay (time);

time : period specified in milliseconds

Example :

delay (3000);

wait 3 seconds

Note : delay() blocks execution

Analog output

analogWrite (pin,Value);

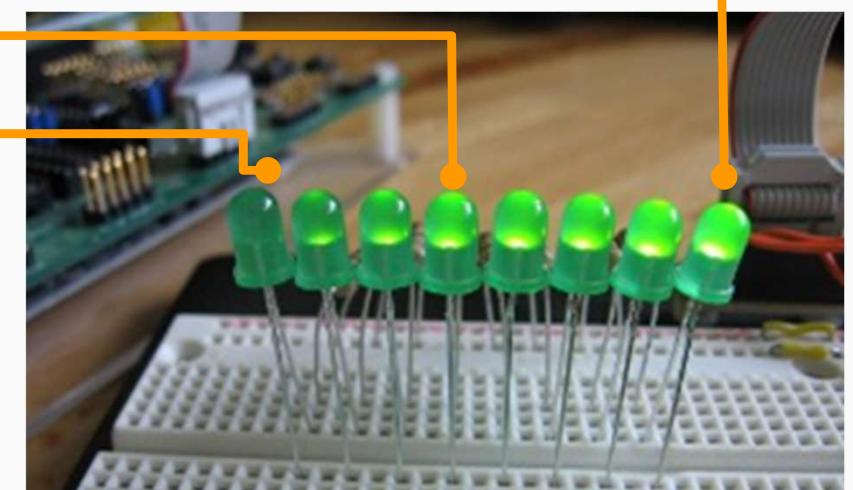
pin : pin number

Value : integer between 0 - 1023

analogWrite (D5,1023);

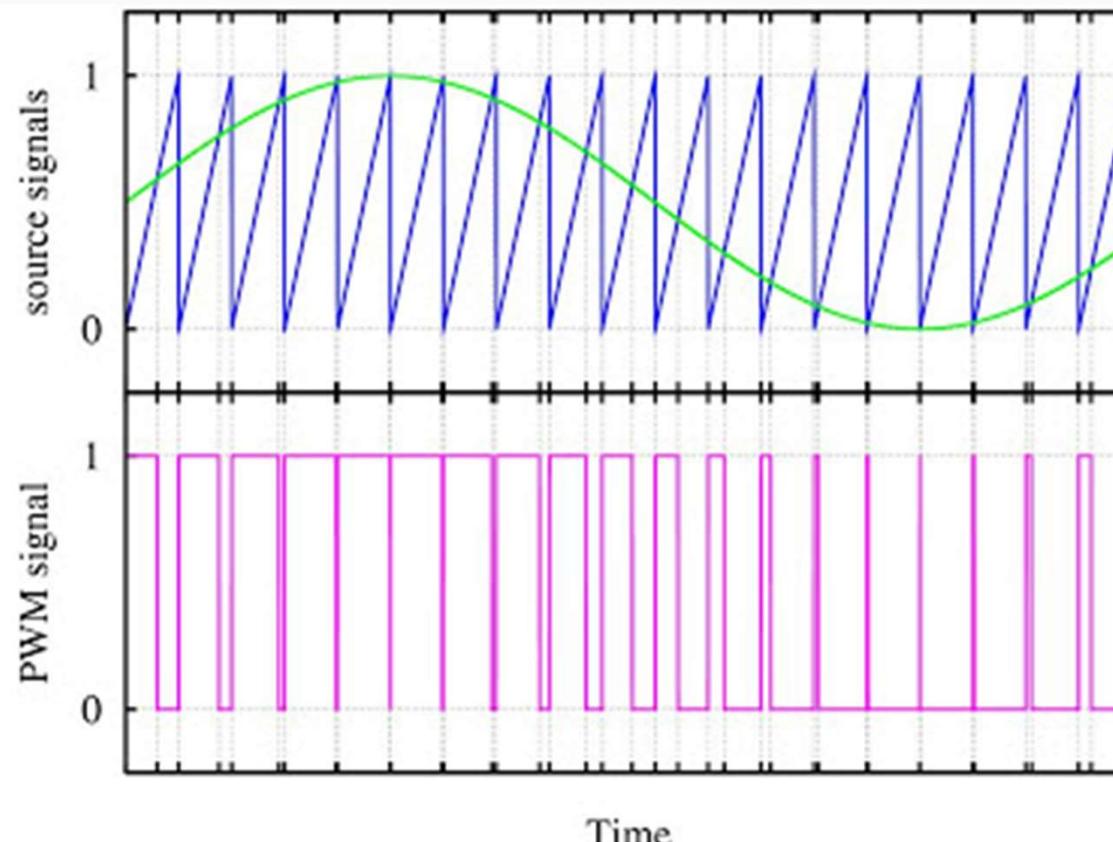
analogWrite (D5,512);

analogWrite (D5,0);

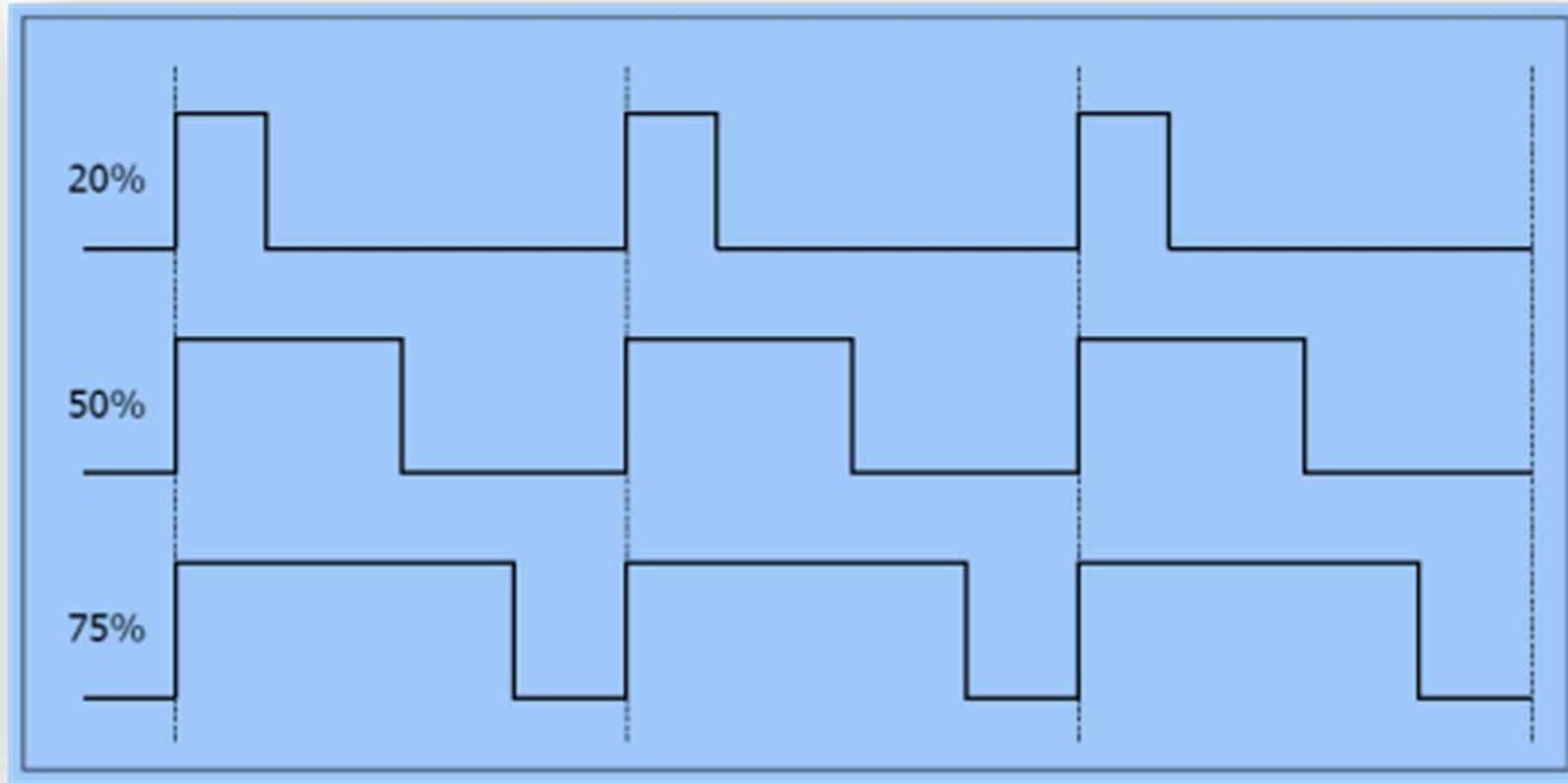


PWM (Pulse Width Modulation)

PWM is generated by comparing an analog output signal with sawtooth wave



PWM (Pulse Width Modulation)



analogWrite controls duty cycle of output.
Can be used with any NodeMCU pin with PWM capability.

Debugging with Serial port

```
Serial.begin(BAUDRATE);
```

Called once in `setup()` to configure serial port

`BAUDRATE` : baud rate is the speed of Serial Port in bits/sec

```
Serial.print(DATA);
```

```
Serial.println(DATA);
```

Used to print variable value or string via serial monitor



Conditional statement

Format:

```
if (condition1) {  
    statement 1; // when condition 1 is true  
}  
  
else if (condition2) {  
    statement 2; // when condition 2 is true  
}  
  
else {  
    statement 3; // when all conditions are false  
}
```

LAB 5-2 : Dim (LED brightness adjustment)

```
#define LED D2

int brightness = 0;
int fadeAmount = 10;

void setup() {
    pinMode(LED, OUTPUT);
}

void loop() {
    analogWrite(LED, brightness);
    brightness = brightness + fadeAmount;
    if (brightness <= 0 || brightness >= 1023) {
        fadeAmount = -fadeAmount;
    }
    delay(30);
}
```

LAB 5-3 : AnalogRead

```
int value = 0;

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

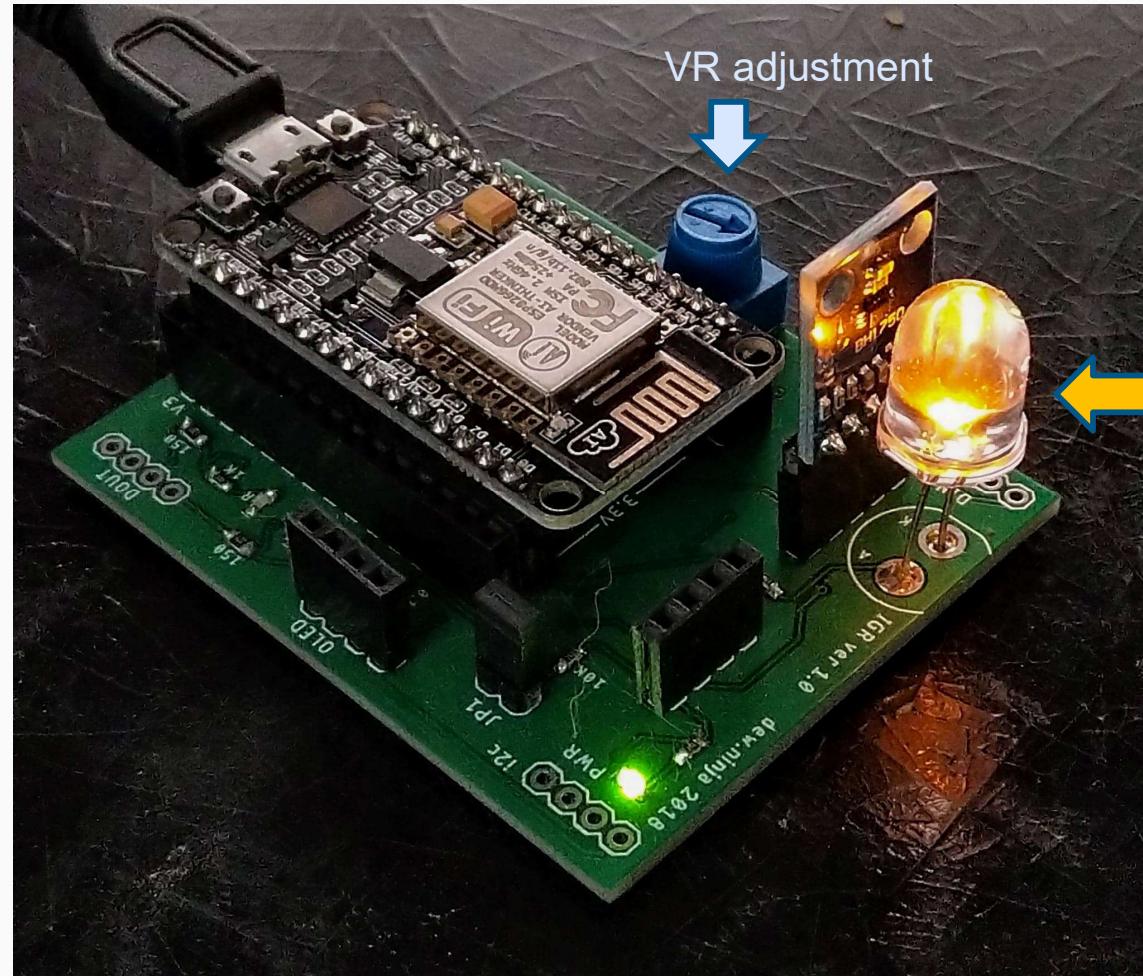
void loop() {
    value = analogRead(A0);
    Serial.print("analog value = ");
    Serial.println(value);
    delay(500);
}
```



open Serial Monitor. Turn on-board VR and observe the readouts

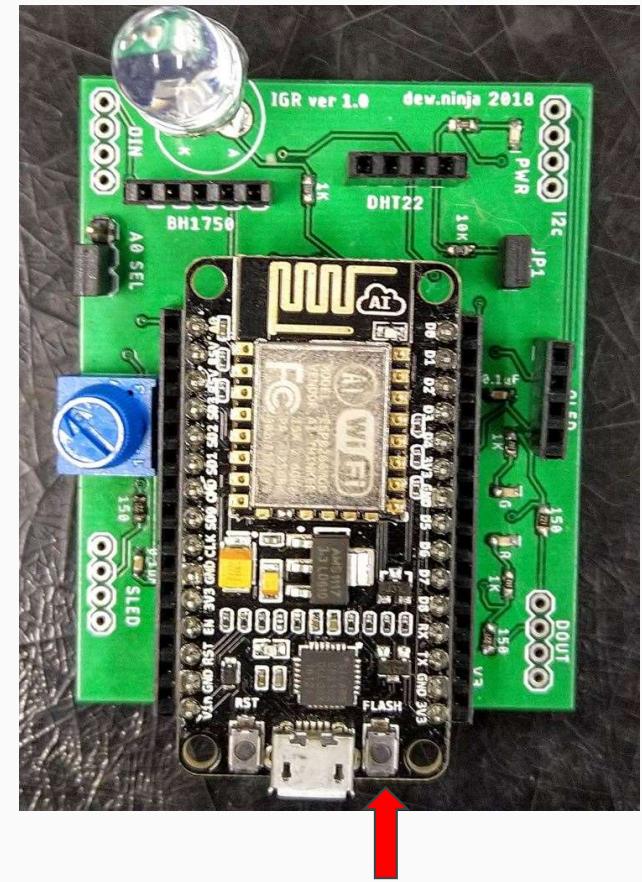
Exercise 2

Adjust LED brightness with VR



LAB 5-4 : Switch

```
#define SWPIN D3  
  
#define LEDPIN LED_BUILTIN  
  
void setup() {  
    pinMode(LEDPIN, OUTPUT); // Set as output  
    pinMode(SWPIN, INPUT); // Set as input  
}  
  
void loop() {  
    if (digitalRead(SWPIN) == HIGH) {  
        digitalWrite(LEDPIN, LOW);  
    }  
    else{  
        digitalWrite(LEDPIN, HIGH);  
    }  
}
```



On-board switch
connected to D3

LAB 5-5 : ToggleSwitch

```
#define SWPIN D3
#define LEDPIN LED_BUILTIN
int buttonState = 0;
int ledState = 0;

void setup() {
    pinMode(LEDPIN, OUTPUT);
    pinMode(SWPIN, INPUT);
}

void loop() {
    int reading = digitalRead(SWPIN);
    if(buttonState != reading) {
        delay(100);
        buttonState = reading;
    }
}
```

```
    if (buttonState == HIGH) {
        ledState = !ledState;
    }
}
digitalWrite(LEDPIN, ledState);
}
```

LAB 5-6 : BlinkWithoutDelay

```
#define INTERVAL 1000

int ledState = LOW;
long previousMillis = 0;

void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    long currentMillis = millis();
    if(currentMillis - previousMillis > INTERVAL) {
        previousMillis = currentMillis;

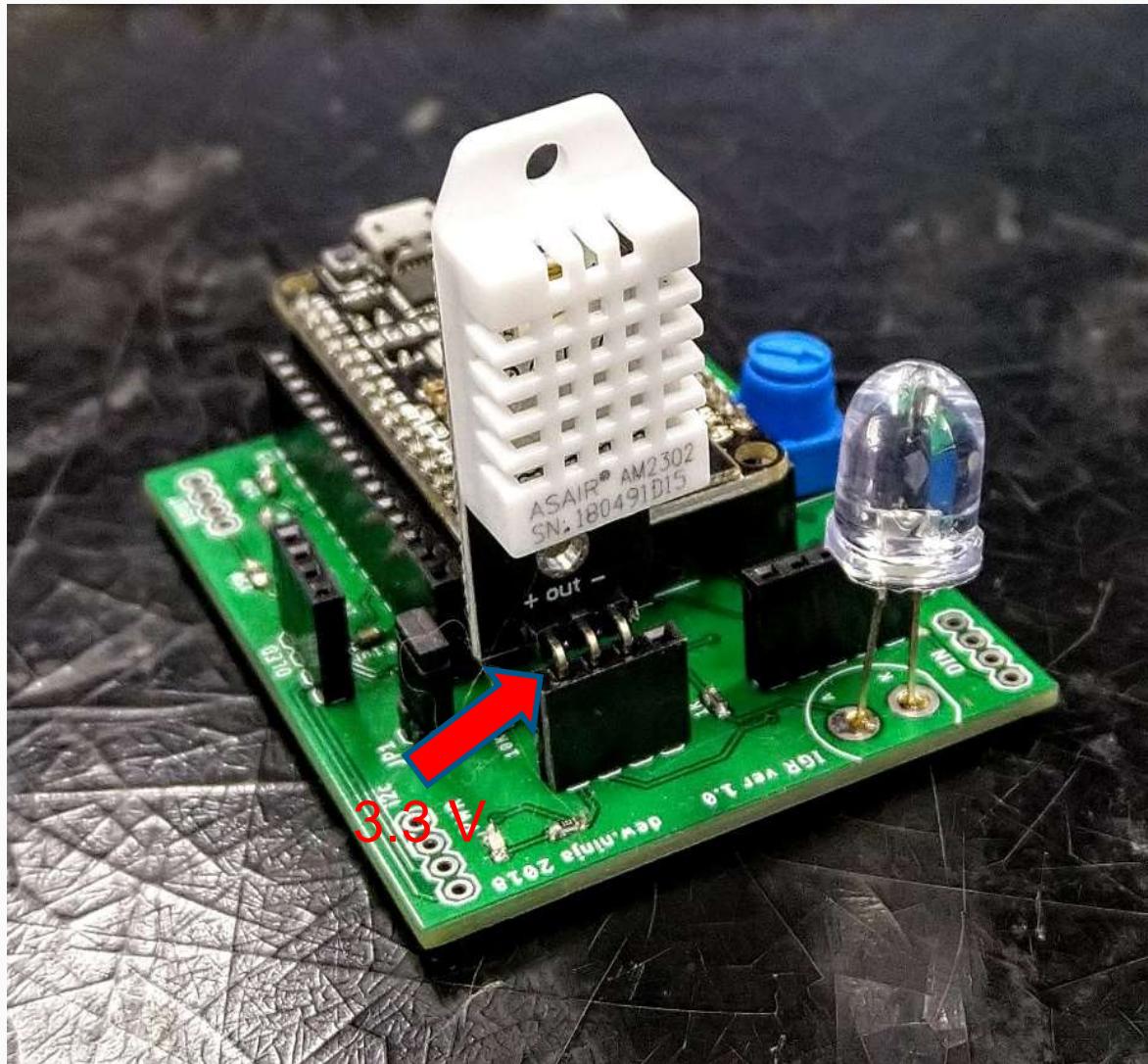
        if (ledState == LOW)
            ledState = HIGH;
        else
            ledState = LOW;
    }
}
```

```
    digitalWrite(LED_BUILTIN, ledState);
}
}
```

Exercise 3

- Modify LAB 2-6 to blink the orange LED (D5) with different rate from on-board LED.
- For example, yellow LED rate = 0.1 Hz, on-board LED = 1 Hz

Connect DHT sensor



DHT Library Installation

- Select Sketch menu
- Include Library → Manage Libraries...
- Search `dht`
- Select DHT sensor library version `xxx` by Adafruit

- Search Adafruit unified sensor
- Select Adafruit Unified Sensor version `xxx` by Adafruit

Using DHT Library

Include DHT library

```
#include <DHT.h>
```

Create DHT object named dht

```
#define DHTTYPE DHT22 // (or DHT11)  
#define DHTPIN D4  
DHT dht(DHTPIN, DHTTYPE);
```

DHT Library Usage

In setup(),

```
dht.begin();
```

Command to verify read status

```
dht.read();
```

Return value

TRUE : read success

FALSE : read failure

DHT Library Usage

Relative humidity reading

```
dht.readHumidity();
```

return value in unit of %RH

Temperature reading

```
dht.readTemperature();
```

Return value in °C unit (use argument
true for °F)

*wait at least 1 second between each reading

LAB 5-7 : DHT

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

DHT dht(DHTPIN, DHTTYPE, 15);

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

void loop() {
    float humid = dht.readHumidity();
    float temp = dht.readTemperature();
    Serial.print("Temp: ");
    Serial.print(temp);
    Serial.print(" , Humid: ");
    Serial.println(humid);
    delay(1000);
}
```

LAB 5-8 : OLED

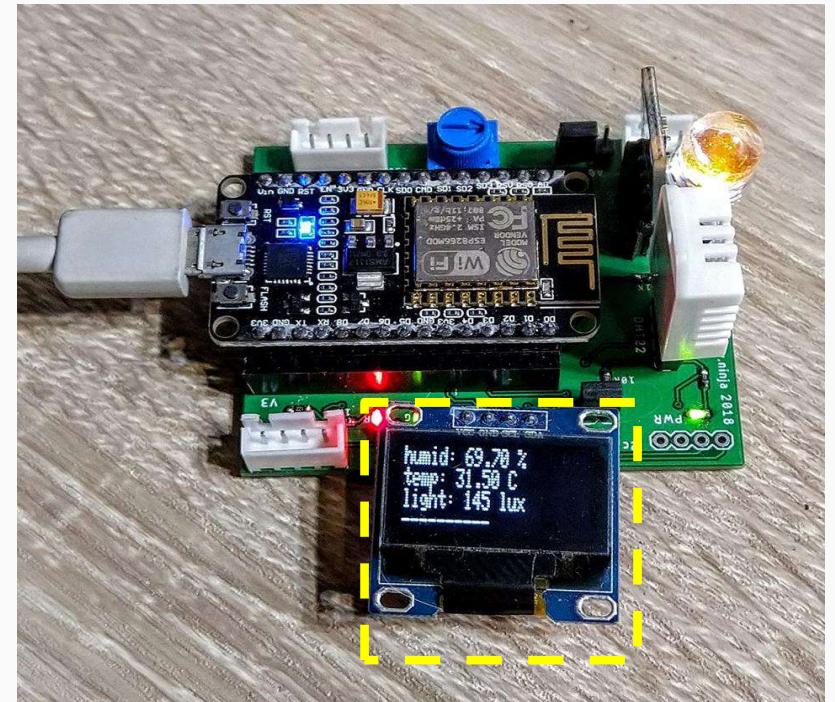
```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET -1
Adafruit_SSD1306 OLED(OLED_RESET);

void setup() {
    Serial.begin(115200);
    OLED.begin(SSD1306_SWITCHCAPVCC, 0x3C);

    OLED.clearDisplay();
    OLED.setTextColor(WHITE);
    OLED.setCursor(0, 0);
    OLED.setTextSize(1);
    OLED.println("Hello NETPIE!");
    OLED.display();
}

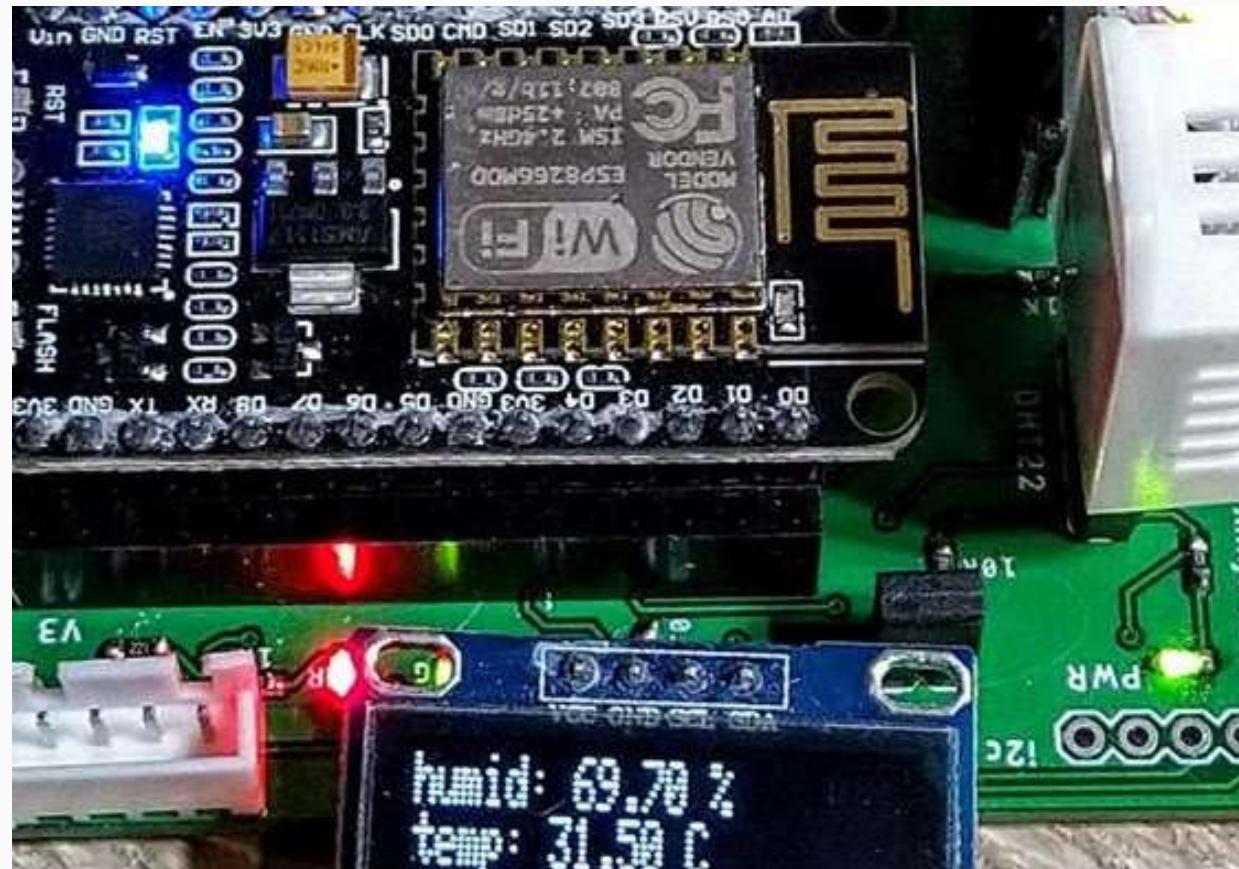
void loop() {}
```



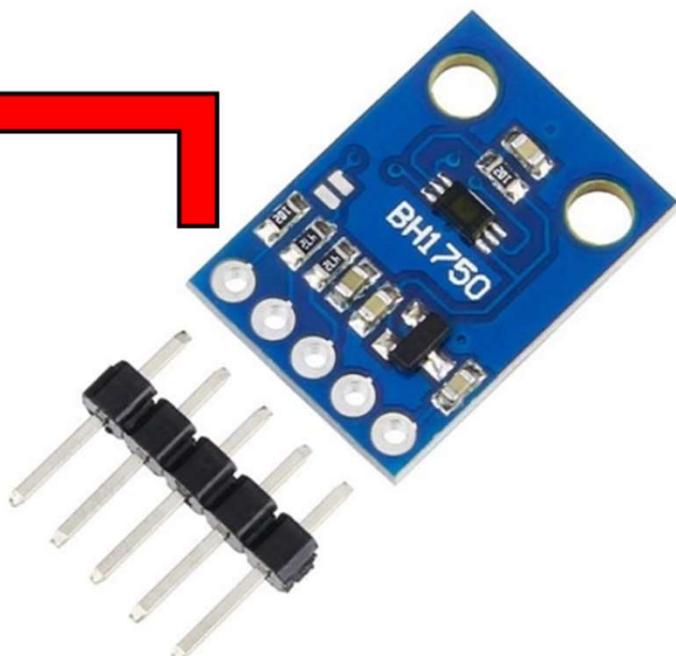
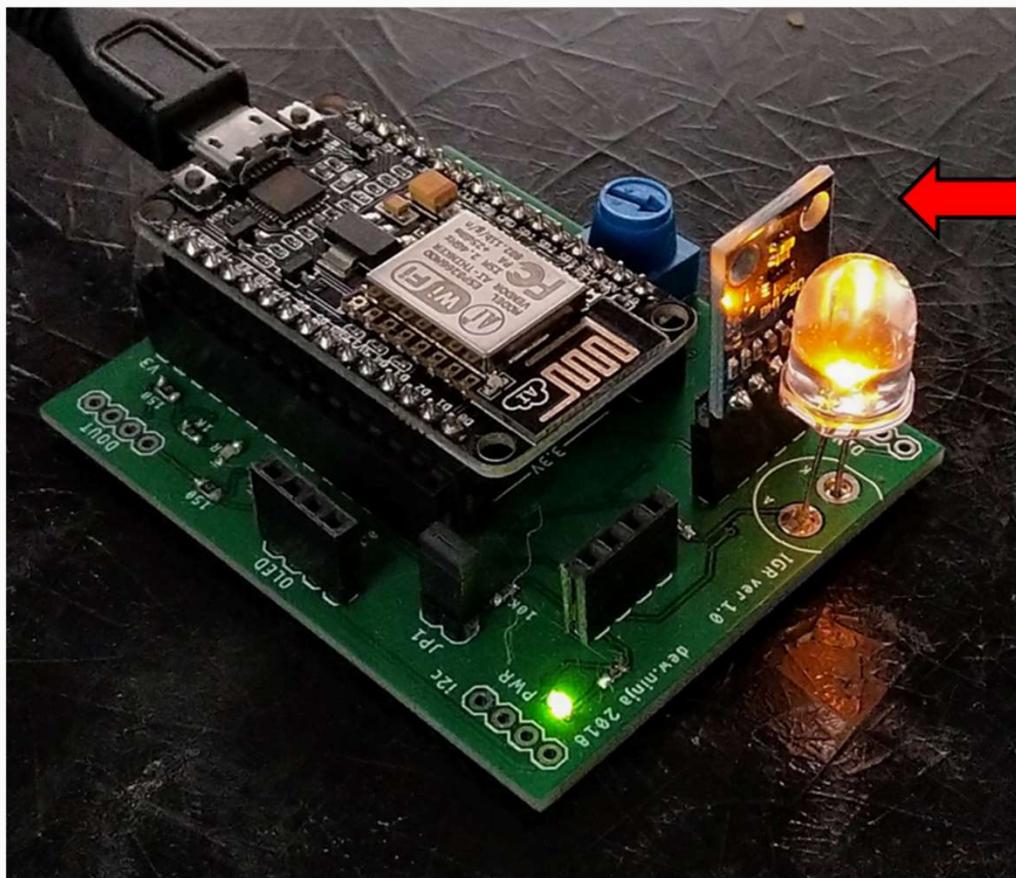
OLED panel

Exercise 4

Show humidity and
Temperature values on OLED



LAB 5-9 : digital light sensor



LAB 5-9 : digital light sensor

```
#include <Wire.h>
#include <BH1750FVI.h>

int lampvalue=0;

uint8_t ADDRESSPIN = 15;      // D8, not connected
BH1750FVI::eDeviceAddress_t DEVICEADDRESS = BH1750FVI::k_DevAddress_L;
BH1750FVI::eDeviceMode_t DEVICEMODE = BH1750FVI::k_DevModeContHighRes;

// Create the Lightsensor instance
BH1750FVI LightSensor(ADDRESSPIN, DEVICEADDRESS, DEVICEMODE);

void setup()
{
    pinMode(D5, OUTPUT);
    Serial.begin(115200);
    LightSensor.begin();
}
```

LAB 5-9 : digital light sensor (continued)

```
void loop()
{
    lampvalue = analogRead(A0);
    analogWrite(D5, lampvalue);
    uint16_t lux = LightSensor.GetLightIntensity();
    Serial.printf("Light: %d lux\n", lux);
    delay(500);
}
```

Adjust LED brightness to see change in sensor reading



```
COM5
Light: 23 lux
Light: 21 lux
Light: 20 lux
Light: 18 lux
Light: 15 lux
Light: 12 lux
Light: 12 lux
Light: 15 lux
Light: 19 lux
Light: 20 lux
Light: 20 lux
Light: 20 lux
Light: 25 lux
Light: 20 lux
Light: 18 lux
Light: 14 lux
Light: 11 lux
```

Autoscroll Show timestamp

Newline 115200 baud Clear output

D1 = SCL, D2 = SDA

Connect NodeMCU to internet

In order to use NodeMCU as station or WiFi access point, **ESP8266WiFi.h** must be included as header file

```
#include <ESP8266WiFi.h>
```

WiFi configuration

Configure ssid and password for WiFi connection

```
WiFi.begin(char* ssid);  
WiFi.begin(char* ssid, char* pass);
```

Example

```
WiFi.begin("HOMEAP", "12345678");
```

Can omit if no password is needed for access point

```
WiFi.begin("HOMEAP");
```

Check IP address

IP address request

```
IPAddress localIP();
```

return value

IP address as 4-byte array corresponding to
the assigned IP address for NodeMCU

Command used to print IP address via serial

```
Serial.println(WiFi.localIP());
```

LAB 5-10 : Wifi

```
#include <ESP8266WiFi.h>
const char* ssid      = "NETPIE1";
const char* password = "netpie1234";
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
void loop() {}
```

.....
WiFi connected
IP address:
192.168.1.3

Wifi Status

WIFI status may be requested by command

```
Wl_status_t WiFi.status();
```

return values

WL_IDLE_STATUS (0)	= connection in progress
WL_NO_SSID_AVAIL (1)	= access point not found
WL_SCAN_COMPLETED (2)	= network scanning completed
WL_CONNECTED (3)	= access point connection completed
WL_CONNECT_FAILED (4)	= access point connection failed
WL_CONNECTION_LOST (5)	= access point connection lost
WL_DISCONNECTED (6)	= disconnected

LAB 5-11: WebClient

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
const char* ssid      = "netpie1";
const char* password = "netpie1234";
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
void loop() {
    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        http.begin("http://cdn.netpie.io/test.html");
        Serial.print("[HTTP] Send request\n");
        int httpCode = http.GET();
        if(httpCode > 0) {
            Serial.printf("[HTTP] Reply code: %d\n", httpCode);
            if(httpCode == HTTP_CODE_OK) {
                String payload = http.getString();
                Serial.println(payload);
            }
        } else {
            Serial.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
        }
        http.end();
    }
    delay(10000);
}
```

The code above is a C++ sketch for an ESP8266 microcontroller. It uses the WiFi library to connect to a network with SSID "netpie1" and password "netpie1234". Once connected, it prints the IP address to the Serial monitor. Then, it sends an HTTP GET request to the URL "http://cdn.netpie.io/test.html". If the request is successful (HTTP_CODE_OK), it prints the received payload. If there's an error, it prints the error message. Finally, it ends the connection and waits for 10 seconds before repeating the process.

Value to text conversion

String can be created from a value with **String**

Format :

String String(val)

Such as

```
String myString;  
int x = 13;  
myString String(x);
```

```
myString = "13"
```

LAB 5-12 : save WiFi data to EEPROM

EEwifi_config.ino

```
#include <EEPROM.h>
#define EEWIFISSIDADDR 0
#define EEWIFIPWDADDR 20
void setup() {
    EEPROM.begin(512);
    EEPROM.put(EEWIFISSIDADDR, "mywifissid");
    EEPROM.put(EEWIFIPWDADDR, "mywifipwd");
    delay(10);
    EEPROM.commit();
}
void loop() { }
```

LAB 5-12 : save WiFi data to EEPROM

```
EEPROM.begin(512);
EEPROM.get(EEWIFISSIDADDR,ssid);
EEPROM.get(EEWIFIPWDADDR,password);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    if (dotdisplay) Serial.print(".");
    if (Serial.available() > 0) { // detect new input
        rcvdstring = Serial.readString();
        if (dotdisplay) {
            dotdisplay=0; // stop displaying dots
            Serial.println("");
        }
        newcmd = 1;
    }
    if (newcmd) { // execute this part only when new command received
        cmdInt(); // invoke the interpreter
        newcmd = 0;
    }
}
```

LAB 5-12 : save WiFi data to EEPROM

```
void cmdInt(void)
{
    rcvdstring.trim(); // remove leading&trailing whitespace, if any
    // find index of separator (blank character)
    sepIndex = rcvdstring.indexOf('=');
    if (sepIndex===-1) {
        cmdstring = rcvdstring;
        noparm = 1;
    }
    else {
        // extract command and parameter
        cmdstring = rcvdstring.substring(0, sepIndex);
        cmdstring.trim();
        parmstring = rcvdstring.substring(sepIndex+1);
        parmstring.trim();
        noparm = 0;
    }
}
```

LAB 5-12 : save WiFi data to EEPROM

```
// check if received command string is a valid command
if (cmdstring.equalsIgnoreCase("wifissid"))    {
    if (noparm==1)    {
        Serial.print("Current Wifi SSID = ");
        Serial.println(ssid);
    }
    else    {
        parmstring.toCharArray(ssid, parmstring.length()+1);
        Serial.print("New SSID = ");
        Serial.println(ssid);
    }
}

else if (cmdstring.equalsIgnoreCase("wifipassword"))    {
    if (noparm==1)    {
        Serial.print("Current Wifi password = ");
        Serial.println(password);
    }
    else    {
        parmstring.toCharArray(password, parmstring.length()+1);
        Serial.print("New Wifi password = ");
        Serial.println(password);
    }
}
```

LAB 5-12 : save WiFi data to EEPROM

```
else if (cmdstring.equalsIgnoreCase("save")) {  
    EEPROM.put(EEWIFISSIDADDR,ssid);  
    EEPROM.put(EEWIFIPWDADDR,password);  
    delay(10);  
    EEPROM.commit();  
    Serial.println("New WiFi config saved. Press reset button.");  
}  
}
```

LAB 5-12 : save WiFi data to EEPROM

To save new WiFi AP info, type commands in Serial Monitor

wifissid=<your ssid>

wifipassword=<your pwd>



The image shows the Arduino IDE interface with the sketch file "L2_12-Wificonfig" open. The code in the editor handles WiFi configuration commands. A large blue curved arrow points from the text "wifipassword=<your pwd>" above to the Serial Monitor window, which displays a command prompt and a series of dots indicating input. The Arduino IDE toolbar at the top includes File, Edit, Sketch, Tools, Help, and various icons.

```
L2_12-Wificonfig | Arduino 1.8.8
File Edit Sketch Tools Help
L2_12-Wificonfig
85     else if (cmdstring.equalsIgnoreCase("wifipassword")) {
86         if (noparam==1)
87             Serial.print("C")
88         Serial.println();
89     }
90     else
91         parmstring.toCharArray();
92         Serial.print("I");
93         Serial.println();
94     }
95     }
96     }
97     }
98 }
99 else if (cmdstring.equalsIgnoreCase("EEPROM"))
100     EEPROM.put(EEWIFIS);
101     EEPROM.put(EEWIFI);
102     delay(10);
103     EEPROM.commit();
104     Serial.println("N");
105 }
```

Serial Monitor window content:

```
COM8
|.....
```

Serial Monitor settings:

- Autoscroll: checked
- Show timestamp: unchecked
- Baud rate: 115200 baud
- Clear output button

Note: appearance changes slightly in newer versions

LAB 5-12 : save WiFi data to EEPROM

