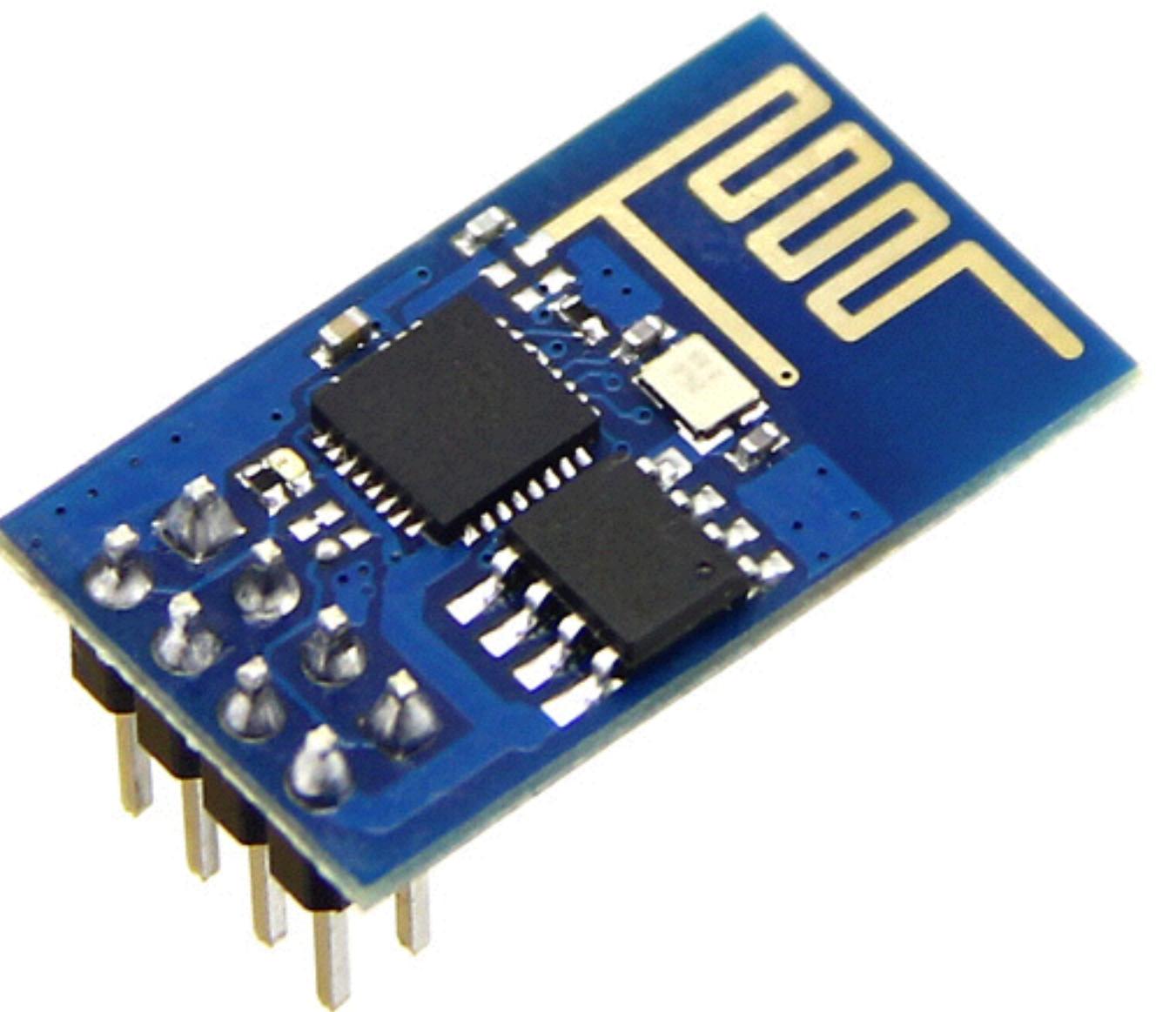
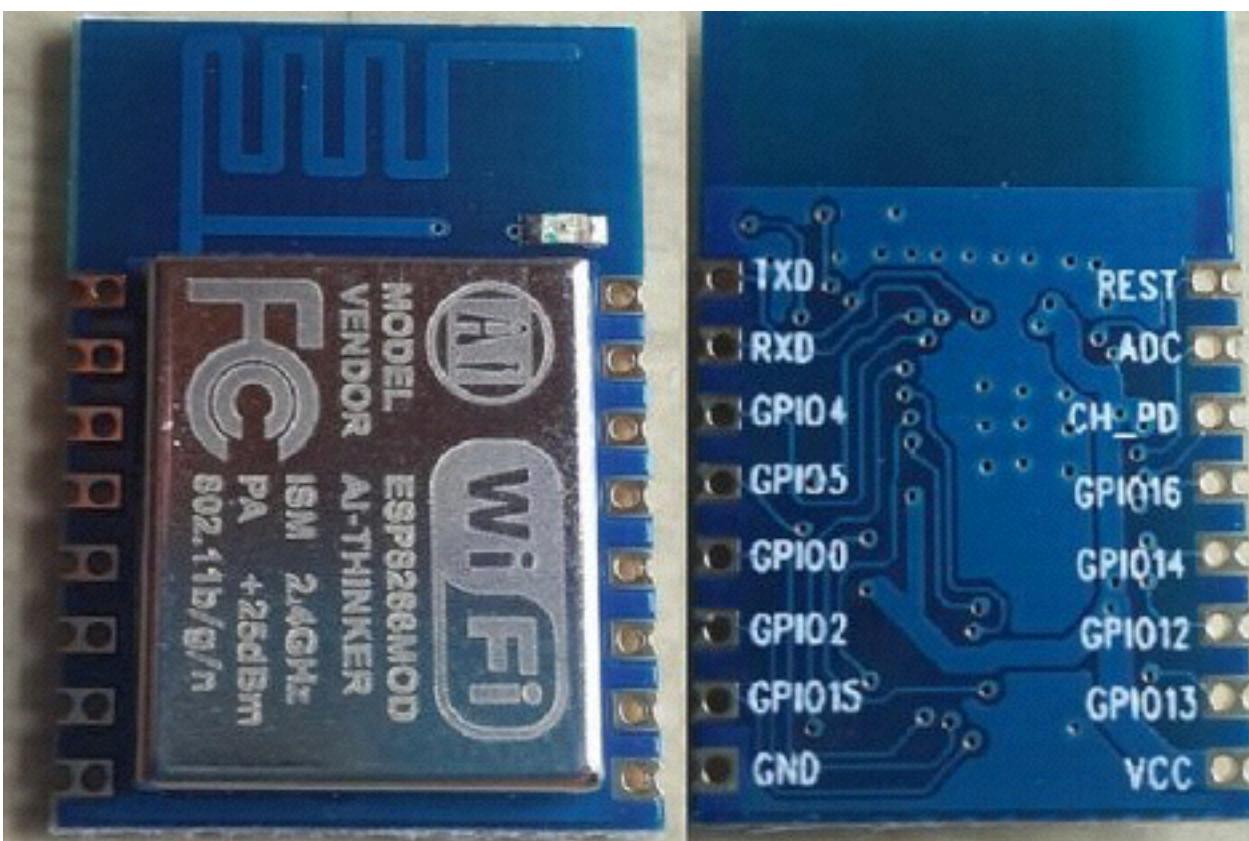
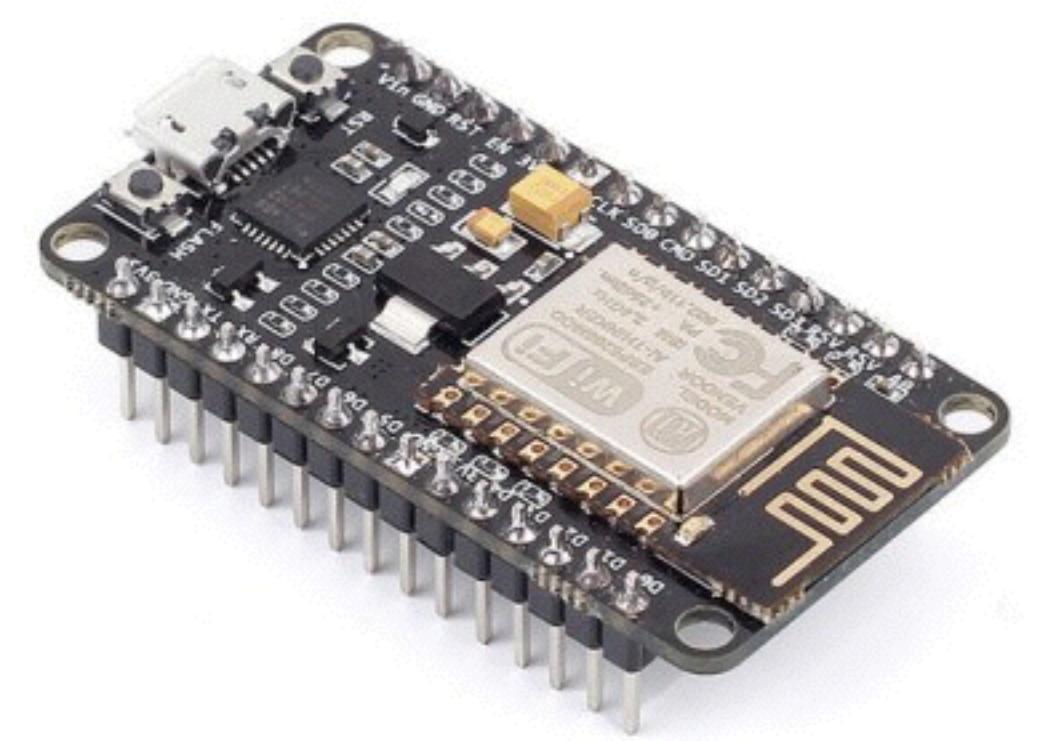


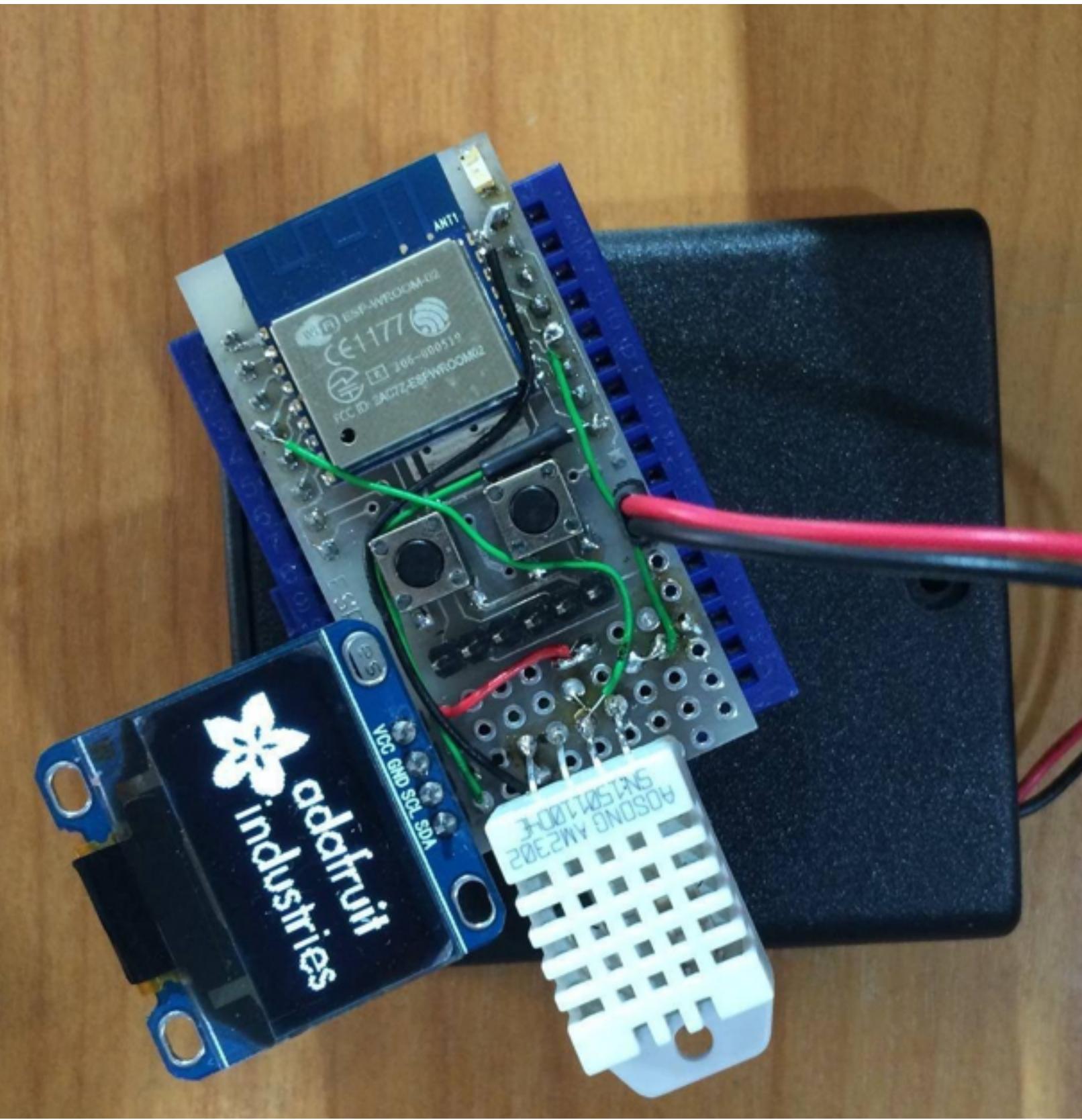
ESP8266 & Arduino IDE

The ESPresso Lite

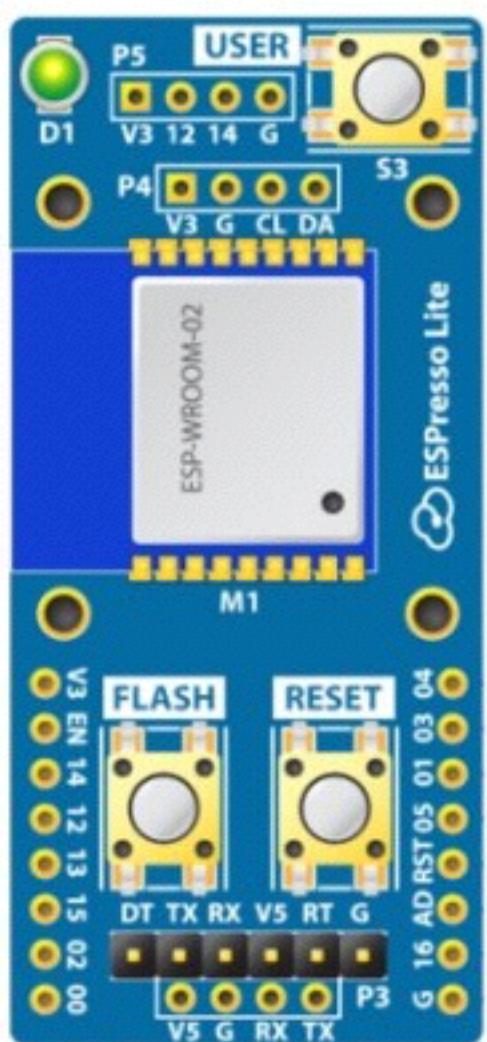




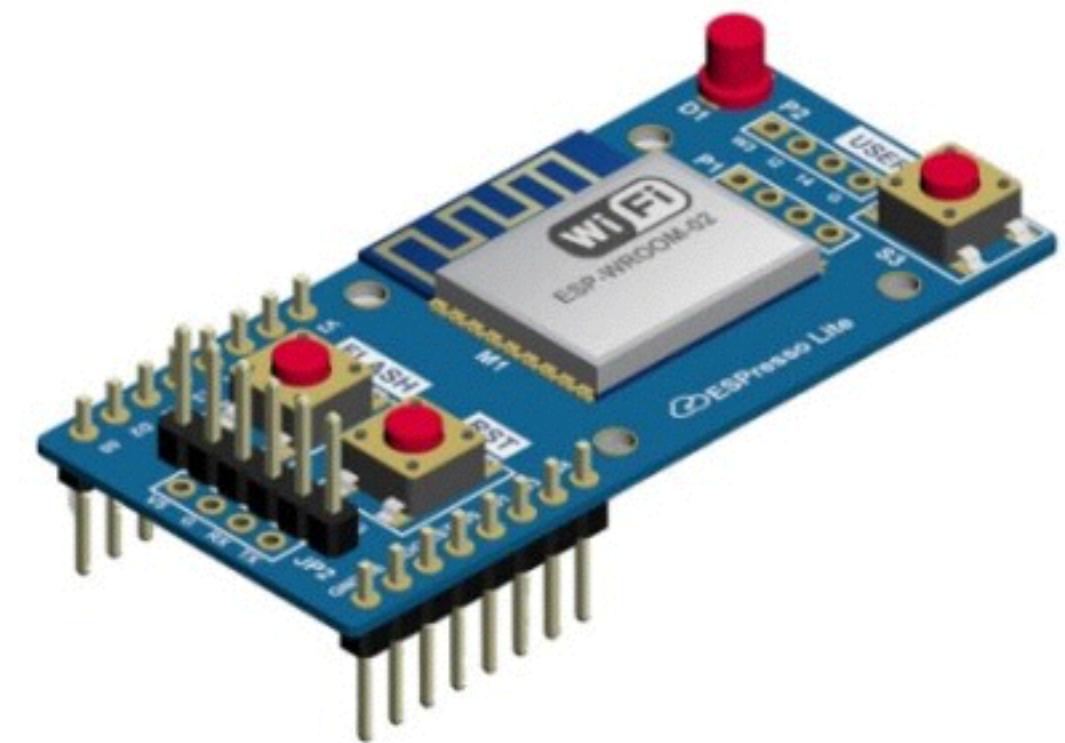


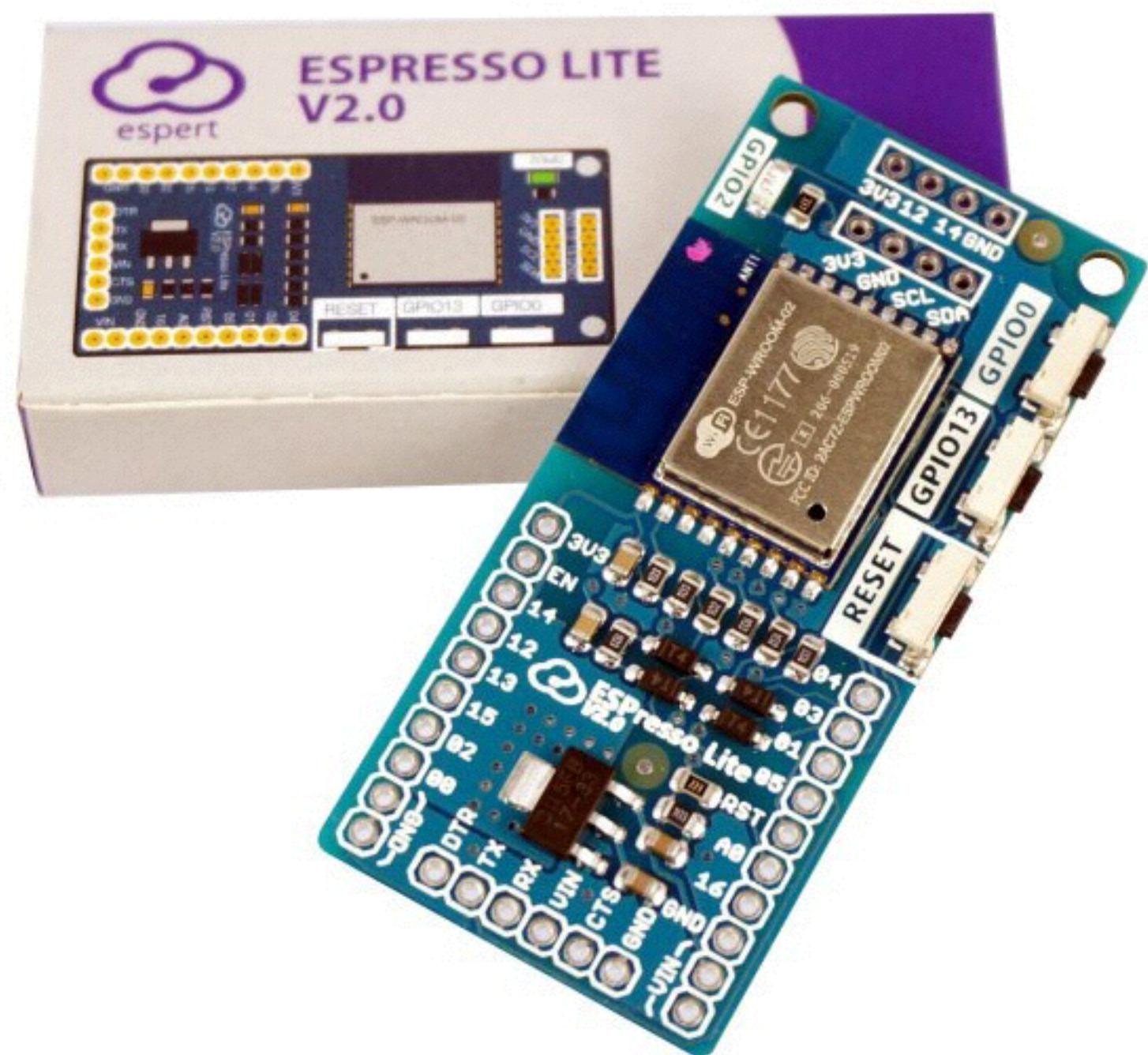


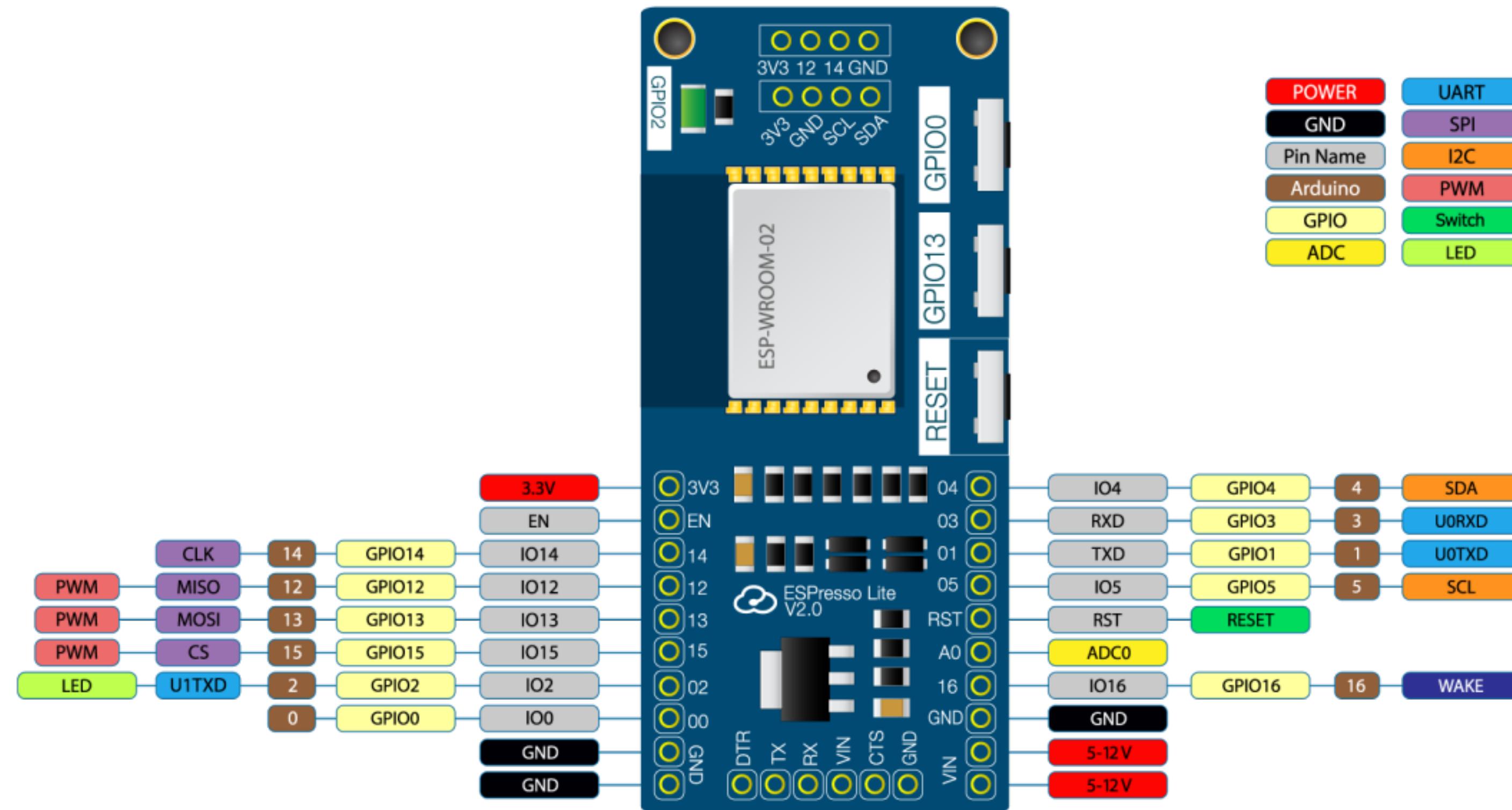
ESPresso Lite



Latest Arduino-compatible,
WiFi-enabled (ESP8266)
development board







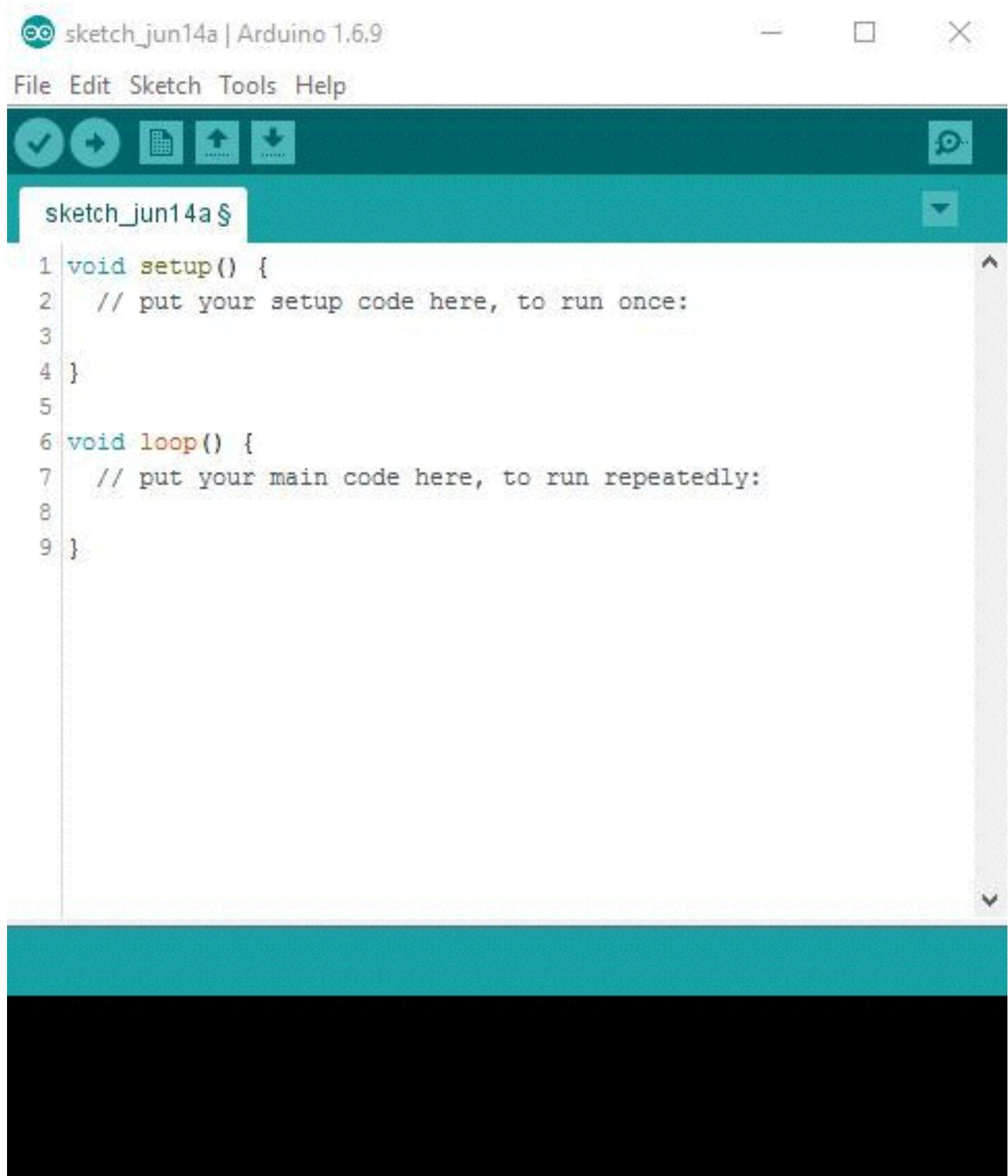
ESPresso Lite V2.0

- Espressif's certified ESP-WROOM-02 Wi-Fi module
(which houses the popular 32-bit 80 Mhz ESP8266 SoC with 64kb RAM & 4Mb flash),
- Two user-programmable buttons (connected to pin 0 & 13) and a reset button,
- Green SMD LED indicator,
- On board 3.3V Voltage Regulator with maximum current of 800mA continuous, 1A peak,
- Input voltage Vin: 5 - 12 VDC; operating voltage at 3.3VDC,
- Supports the Arduino IDE with own board manager and libraries,
- Auto program loading from Arduino IDE; no manual resetting required,
- Custom-arranged I2C pads for I2C-compliant sensors or OLED LC display,
- Breakout pins are breadboard-friendly .

Download the Arduino Software

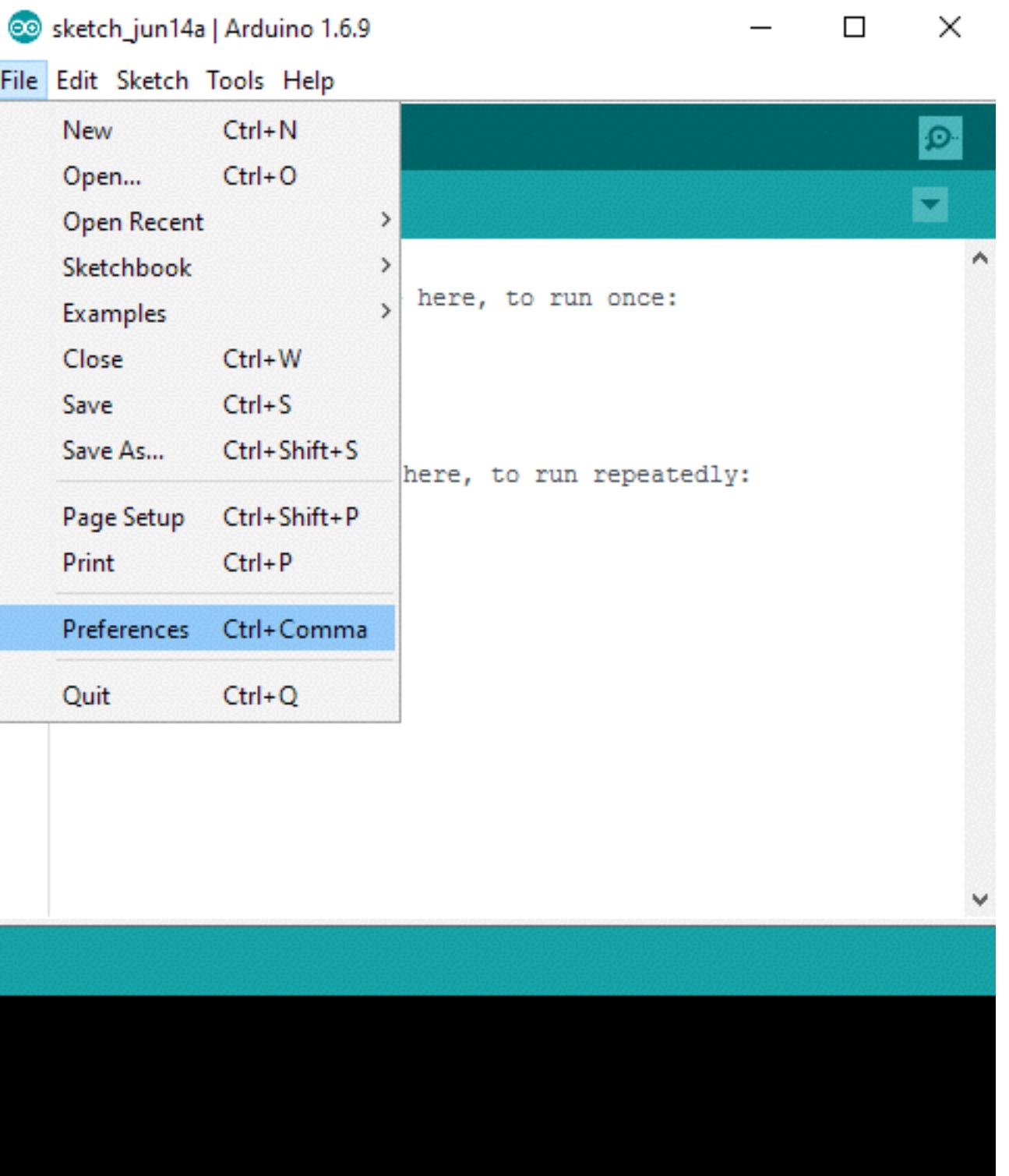


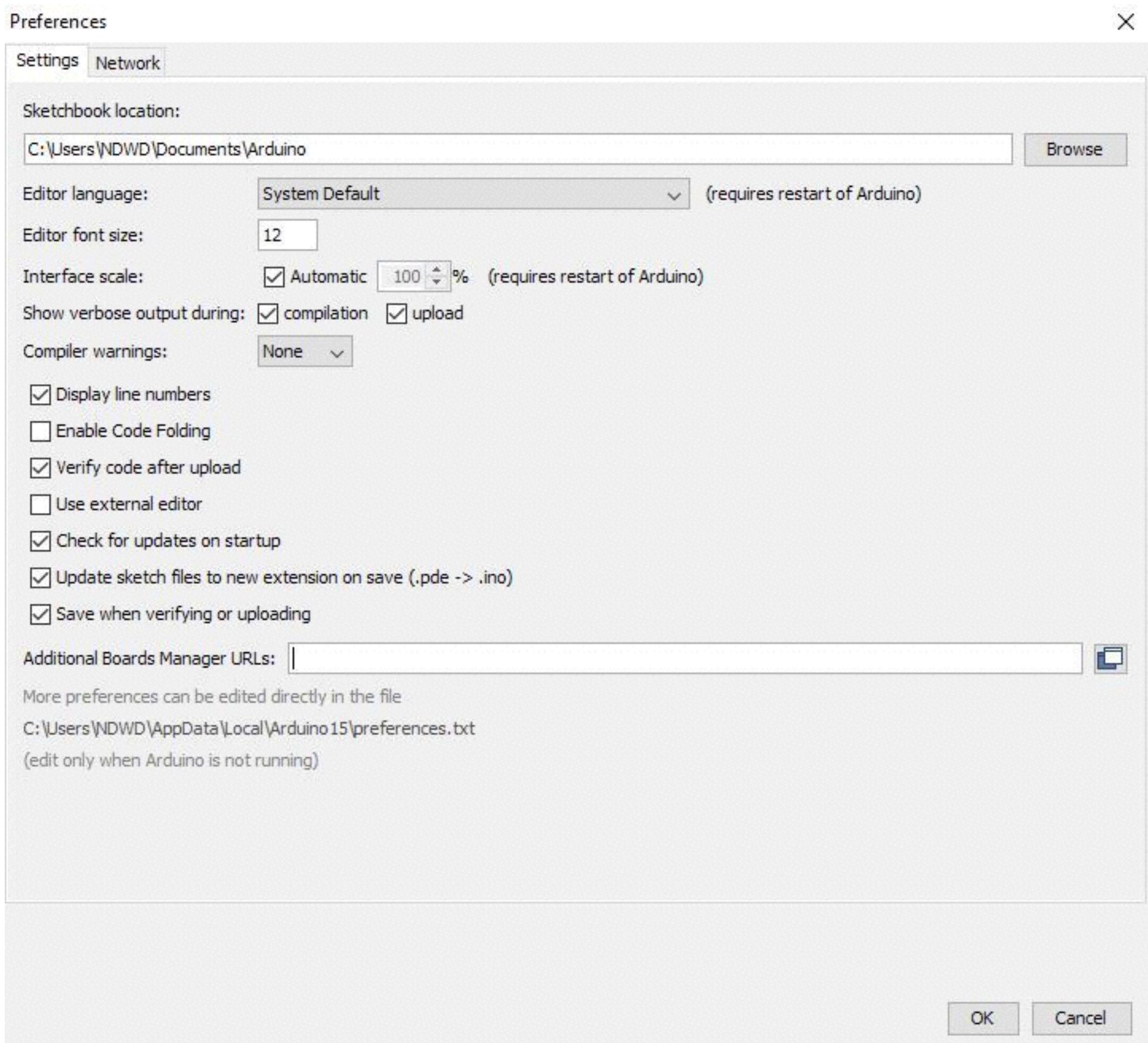


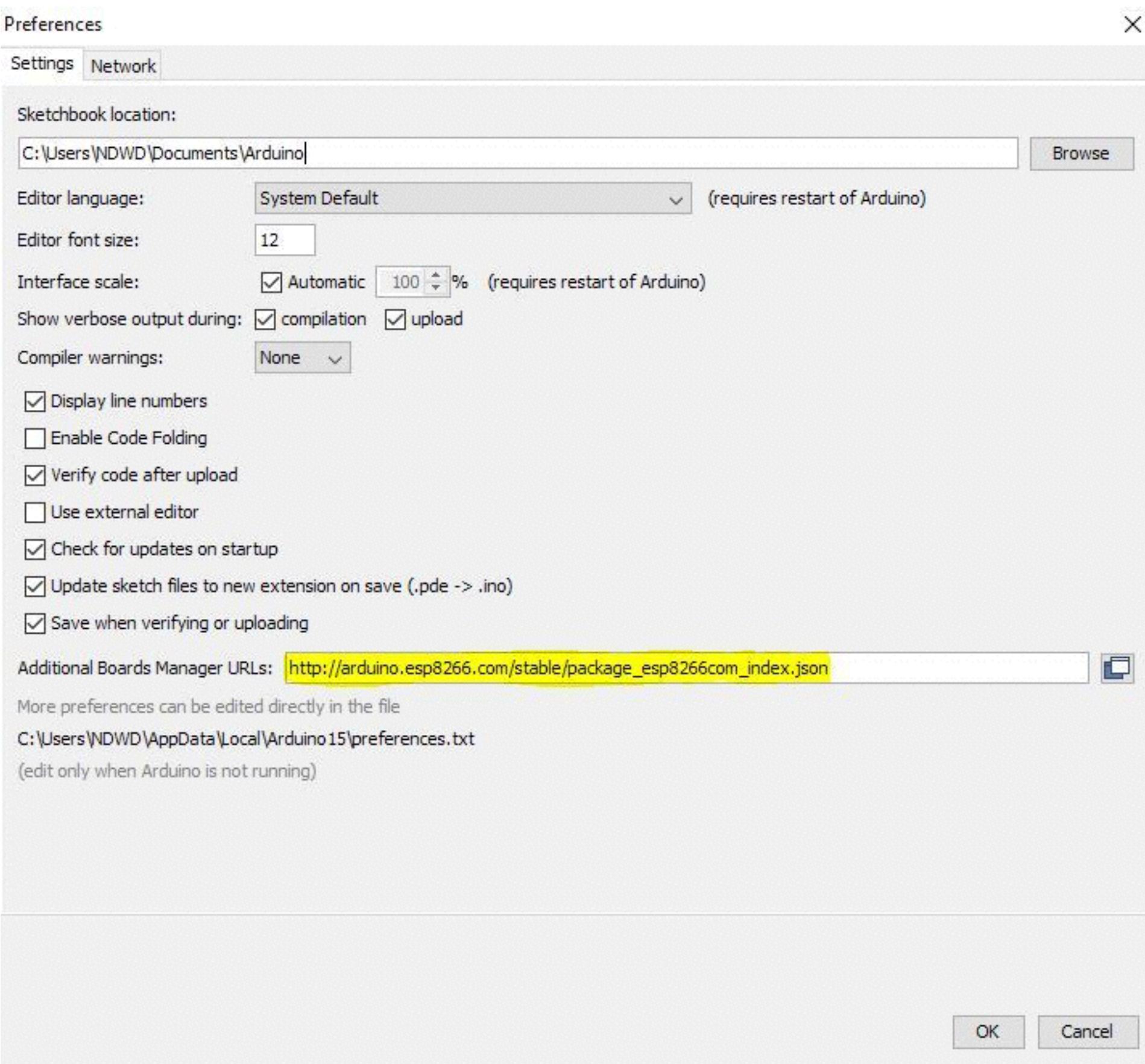


The screenshot shows the Arduino IDE interface. The title bar reads "sketch_jun14a | Arduino 1.6.9". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu is a toolbar with icons for file operations like Open, Save, and Print. The main workspace is titled "sketch_jun14a §" and contains the following Arduino code:

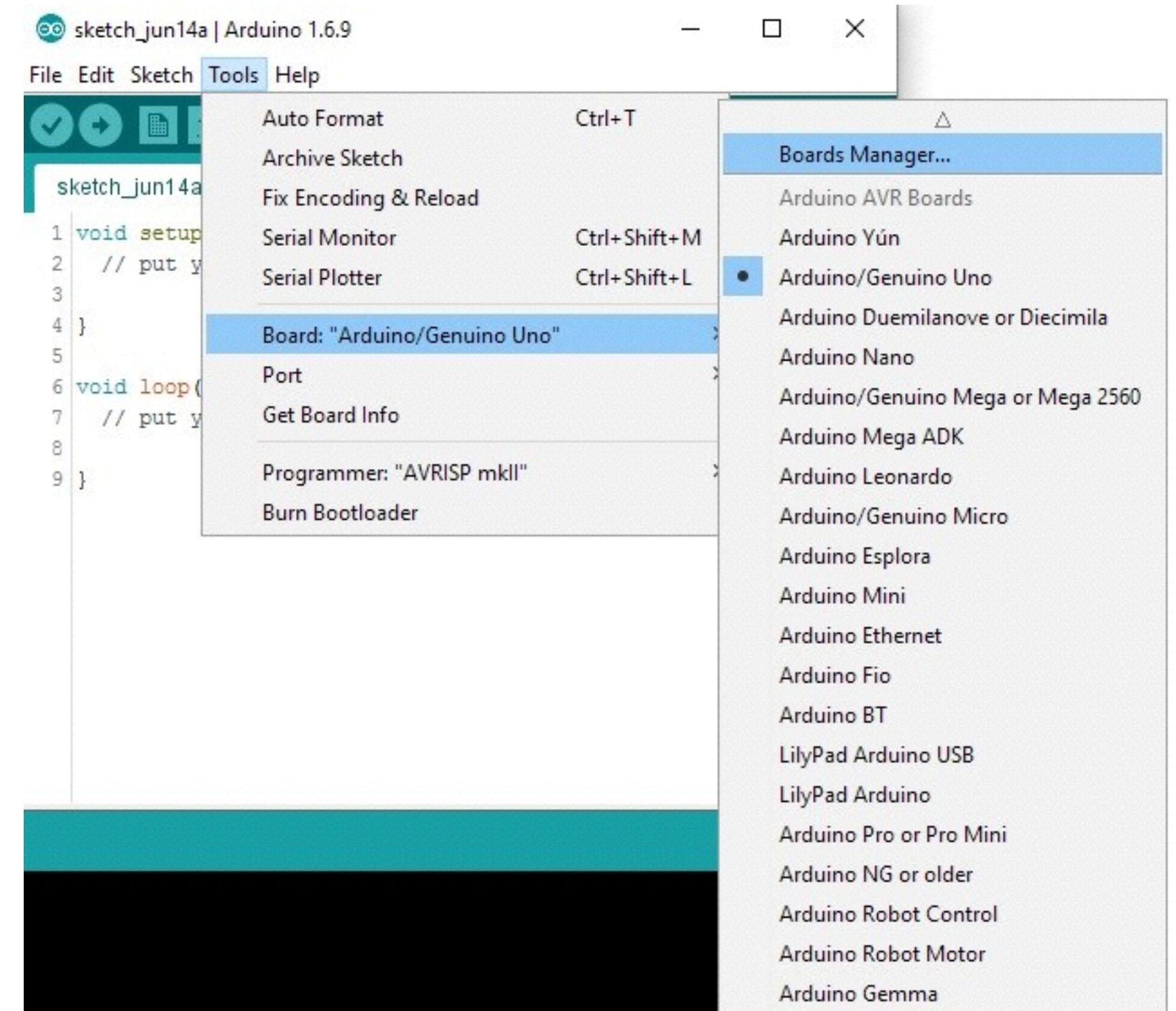
```
1 void setup() {  
2   // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8 }  
9 }
```

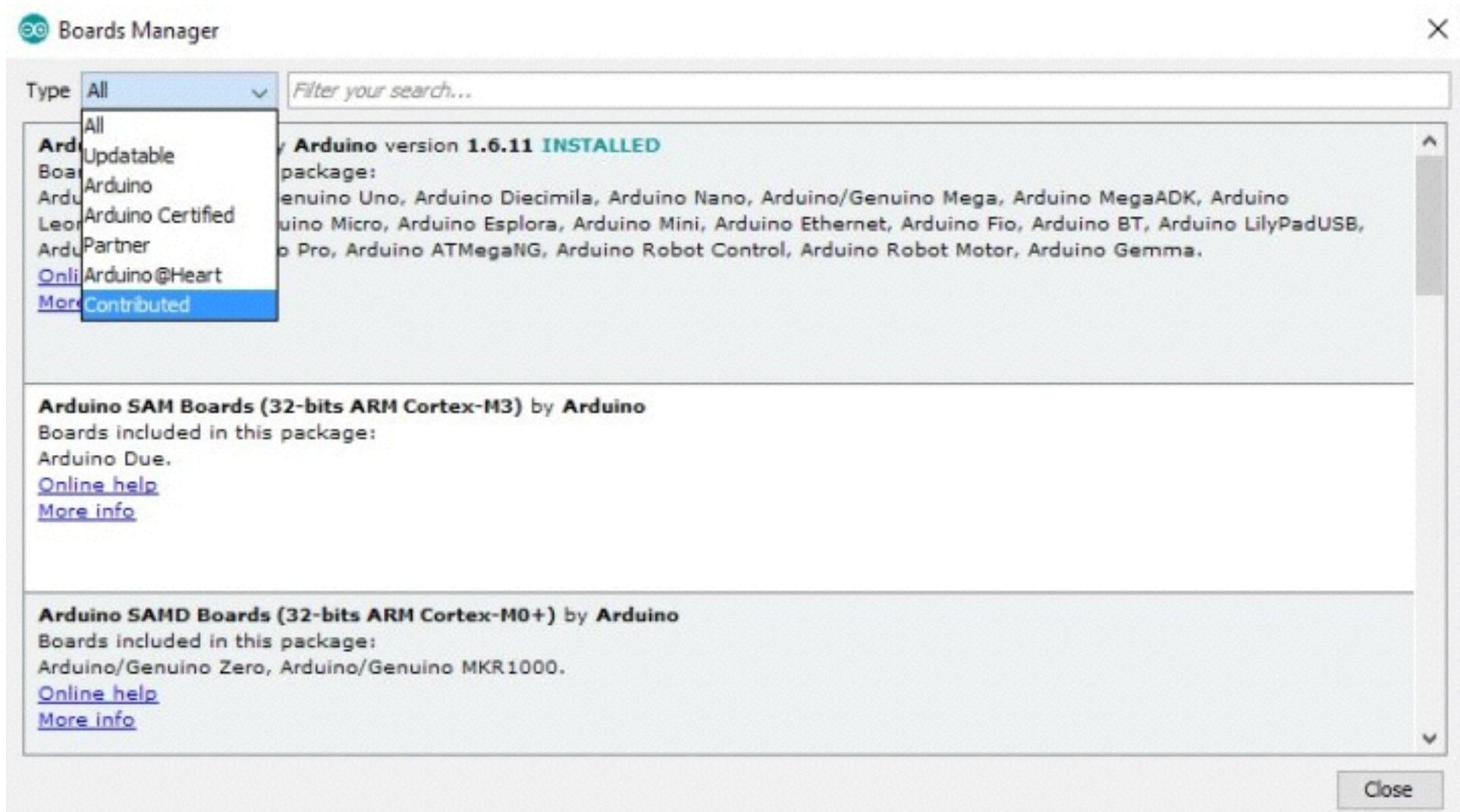


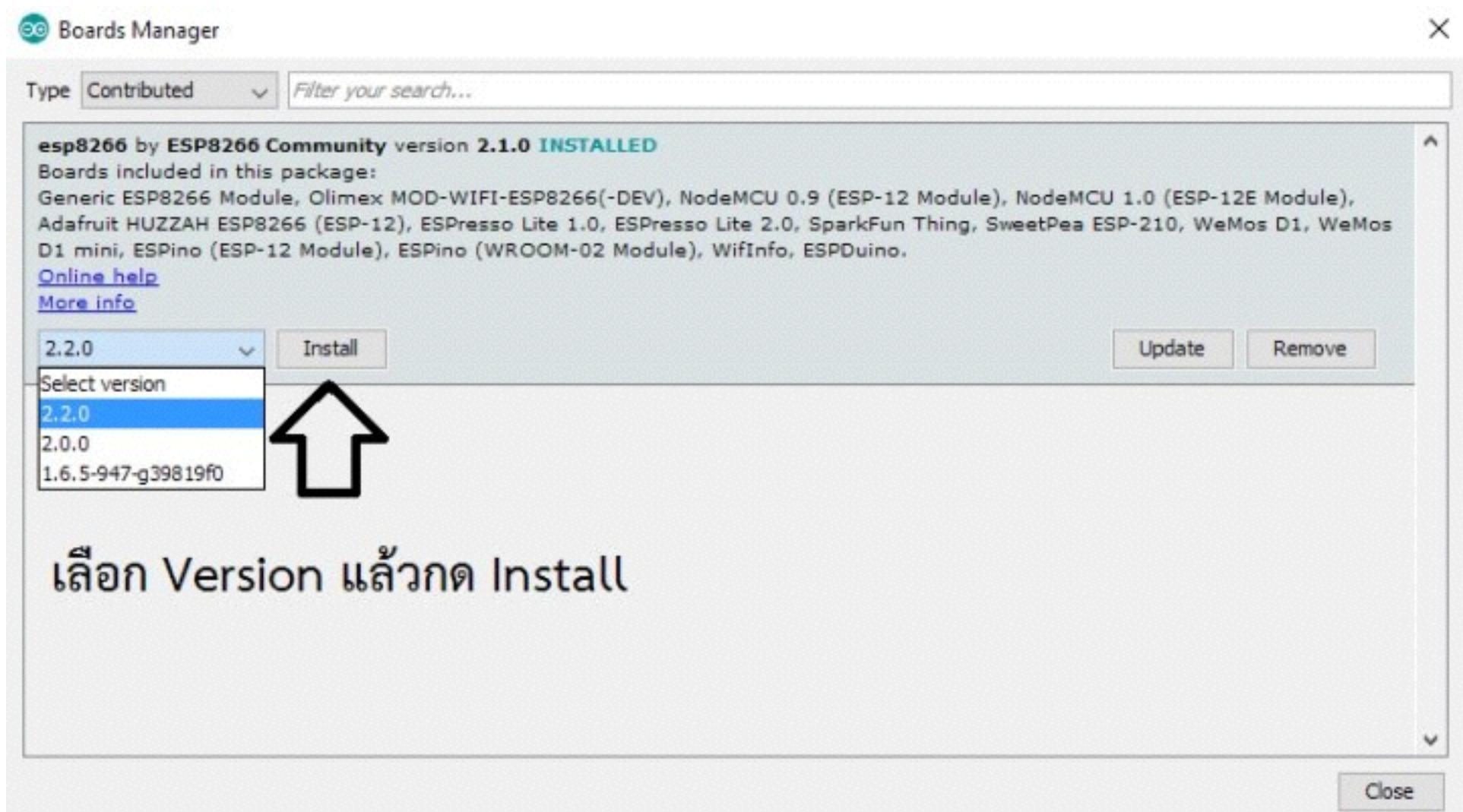




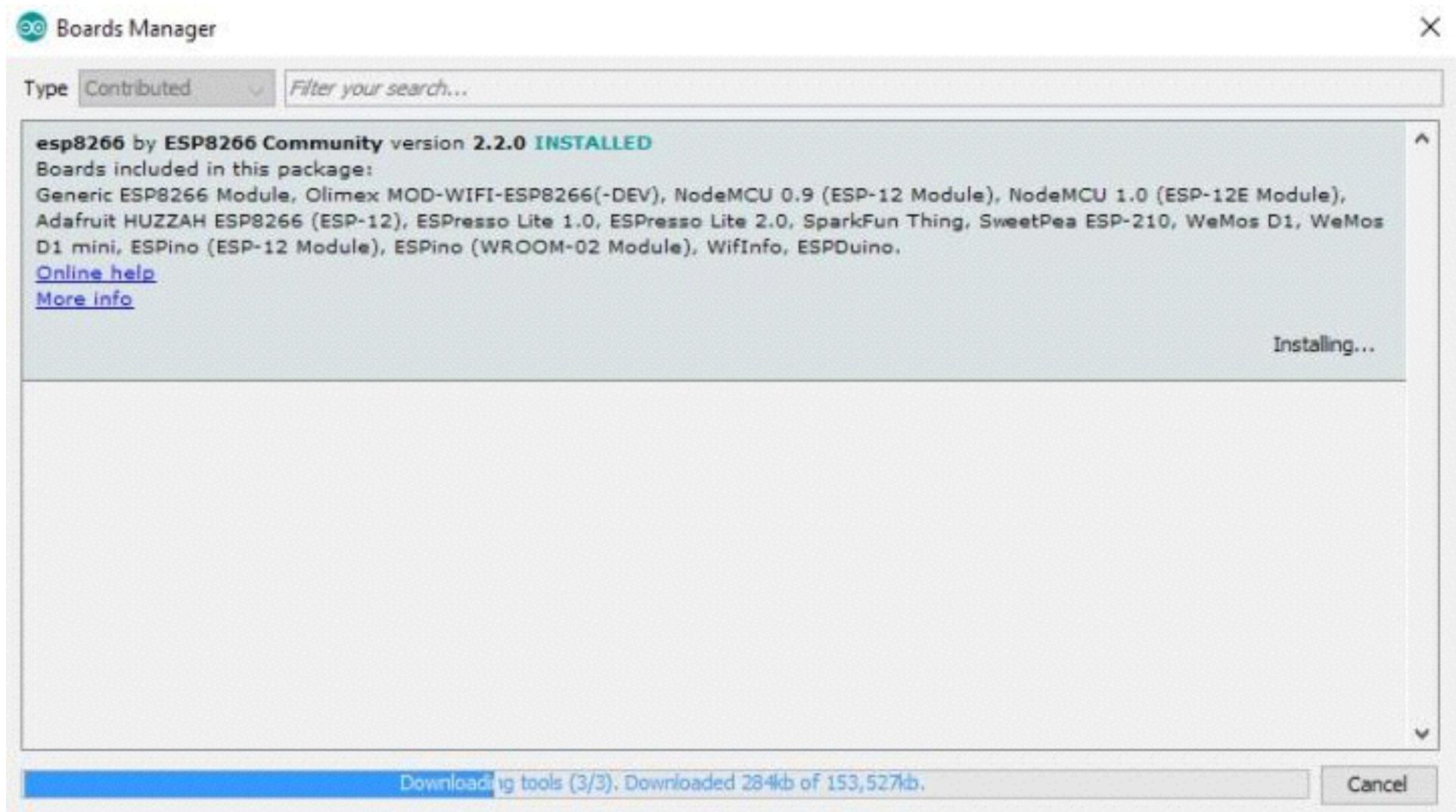
<https://github.com/esp8266/Arduino>

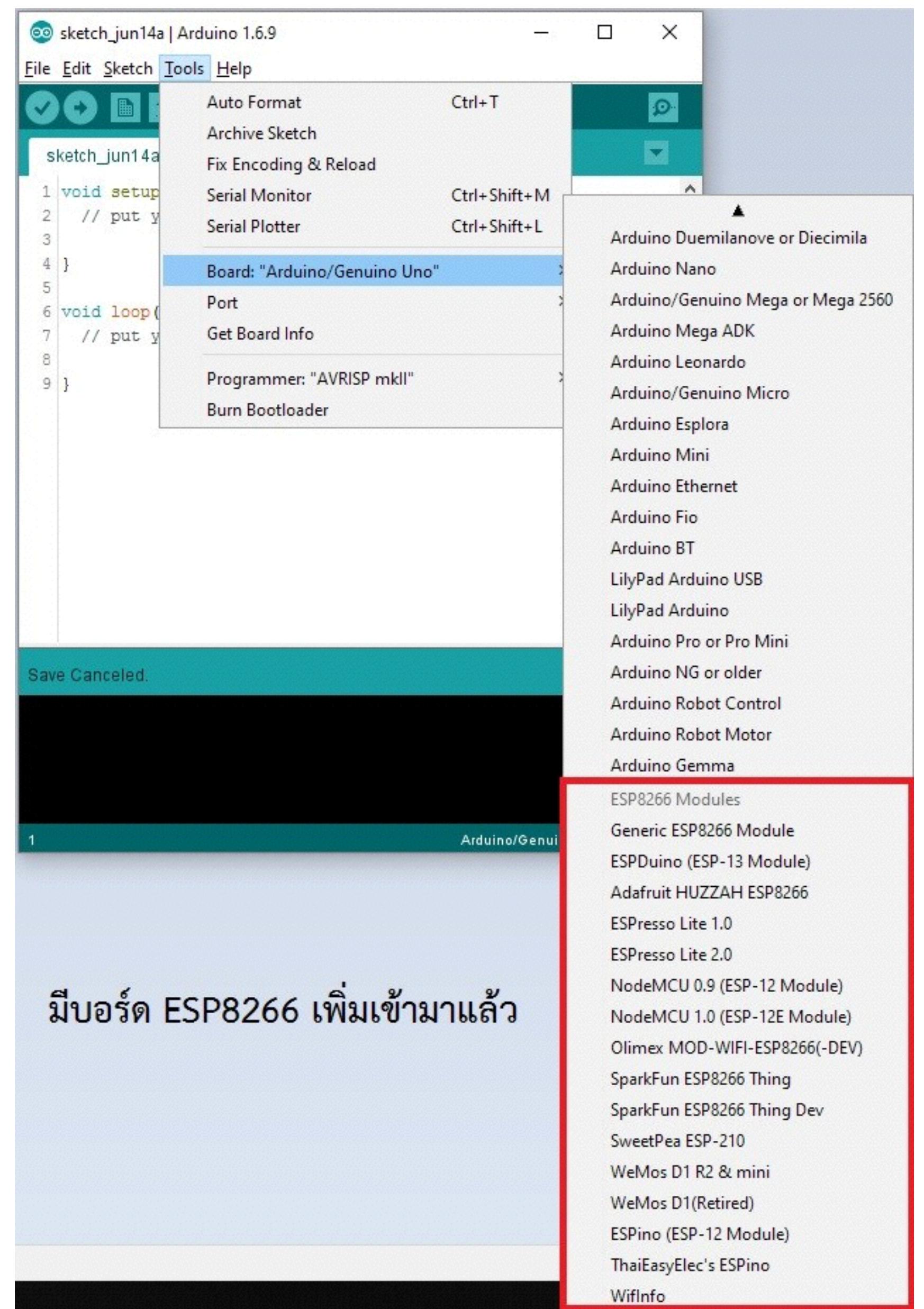




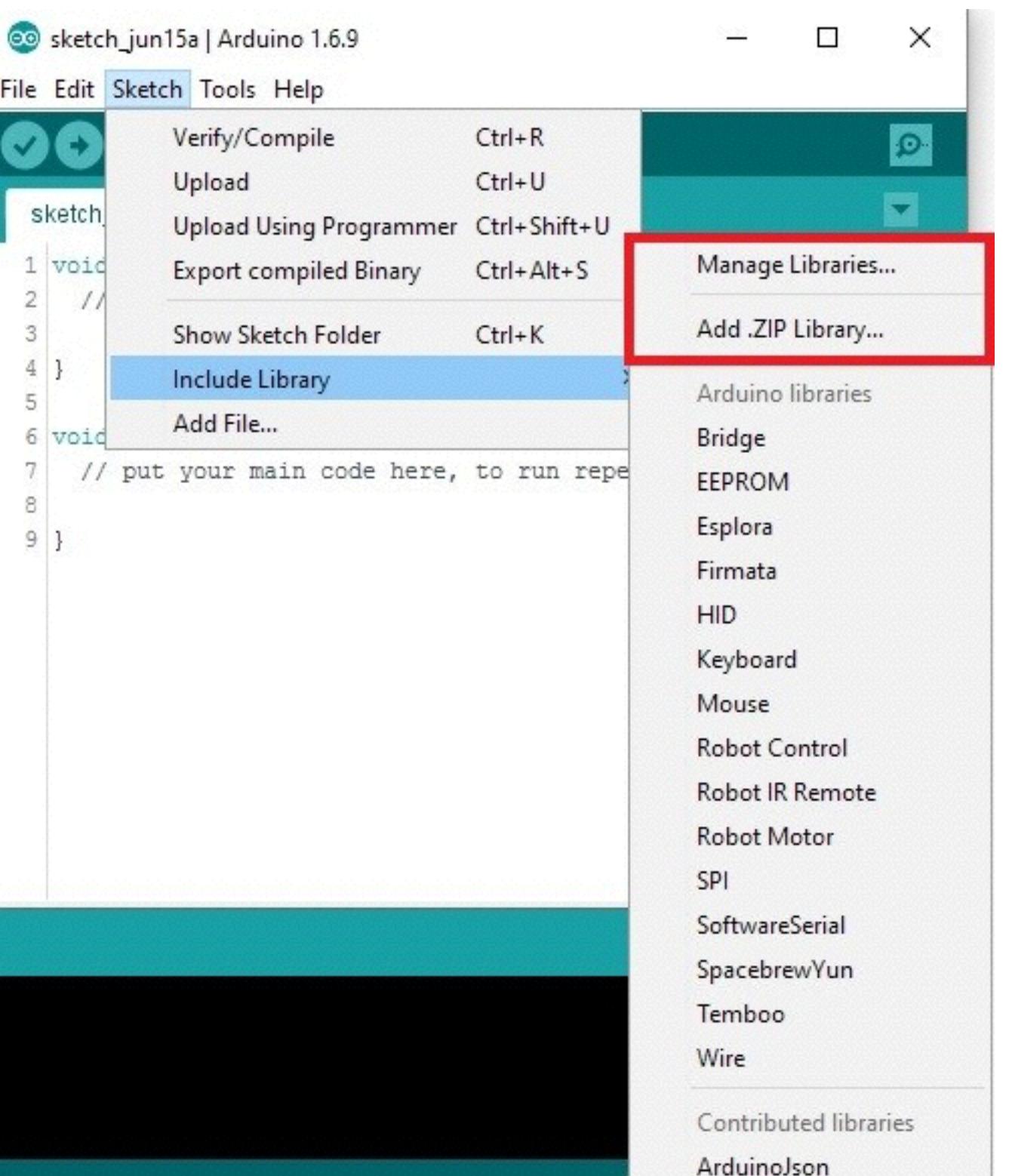


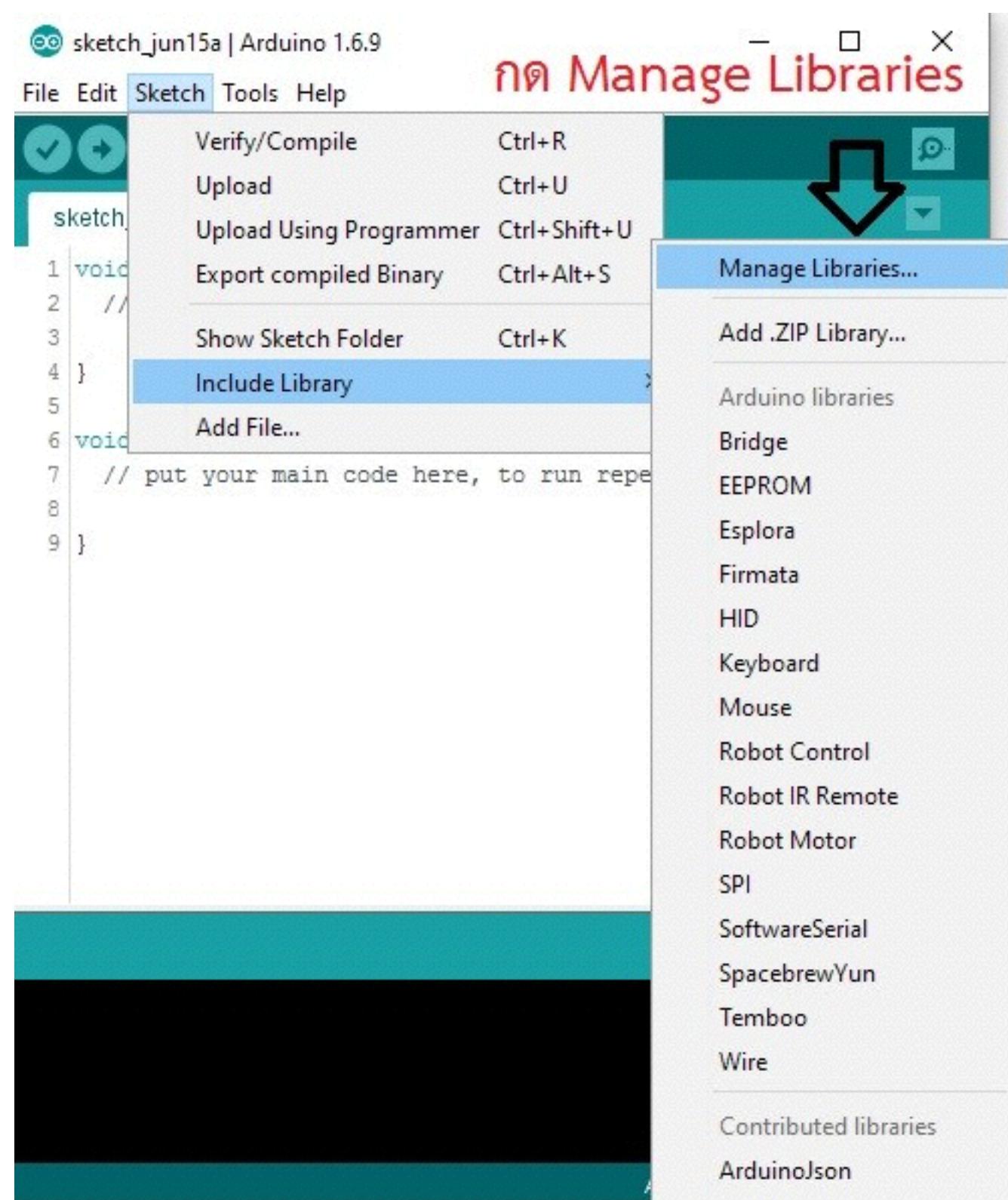
เลือก Version และกด Install

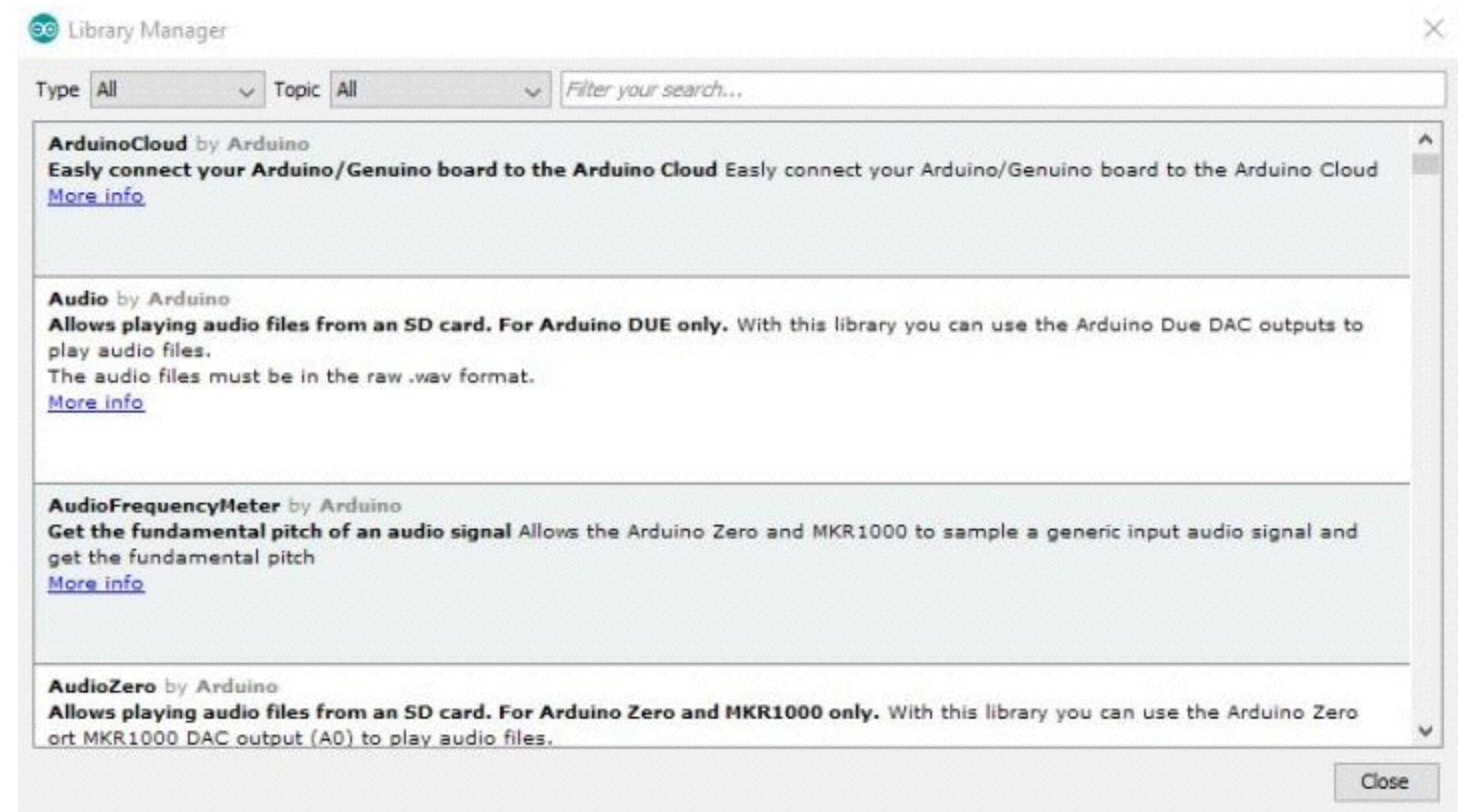


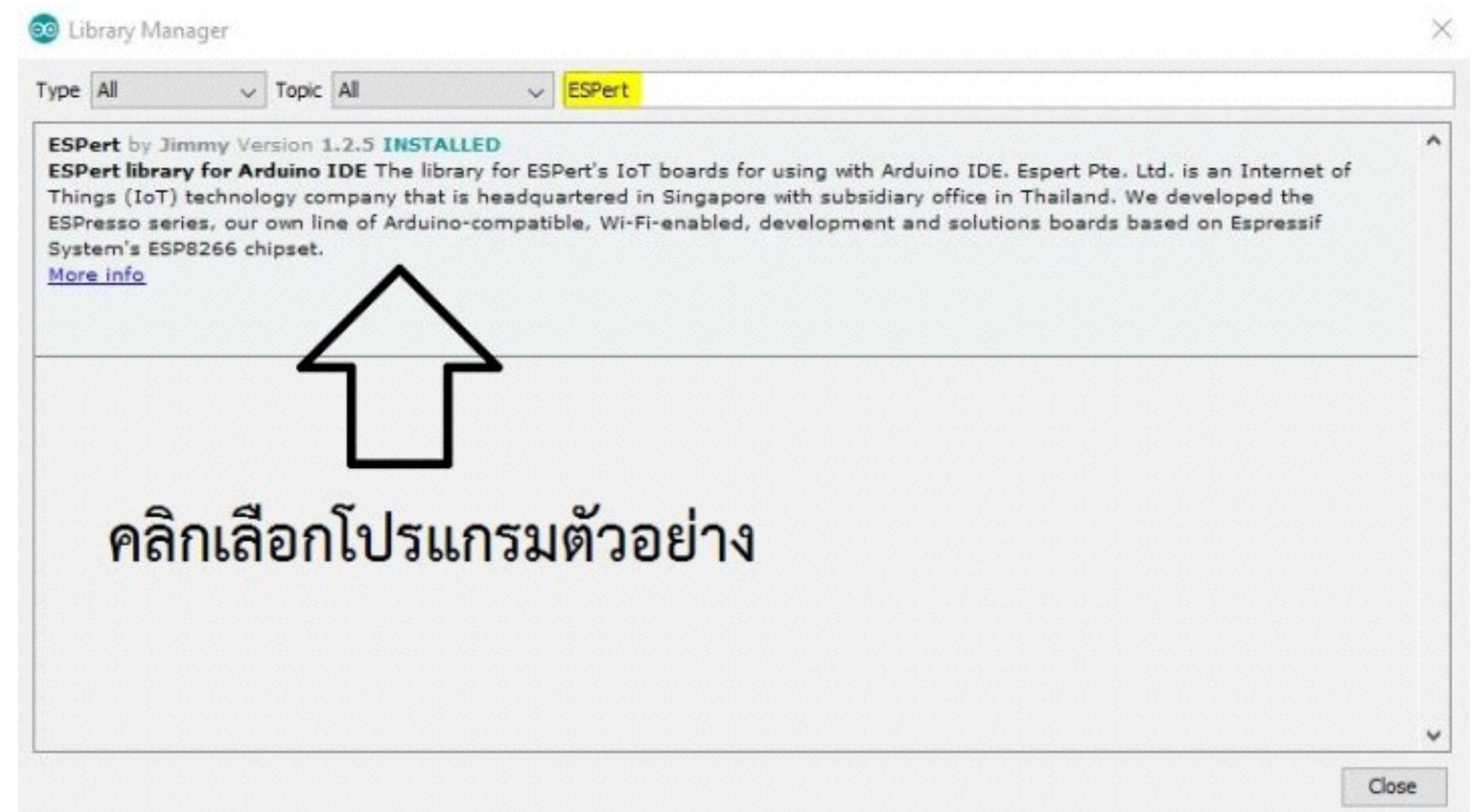


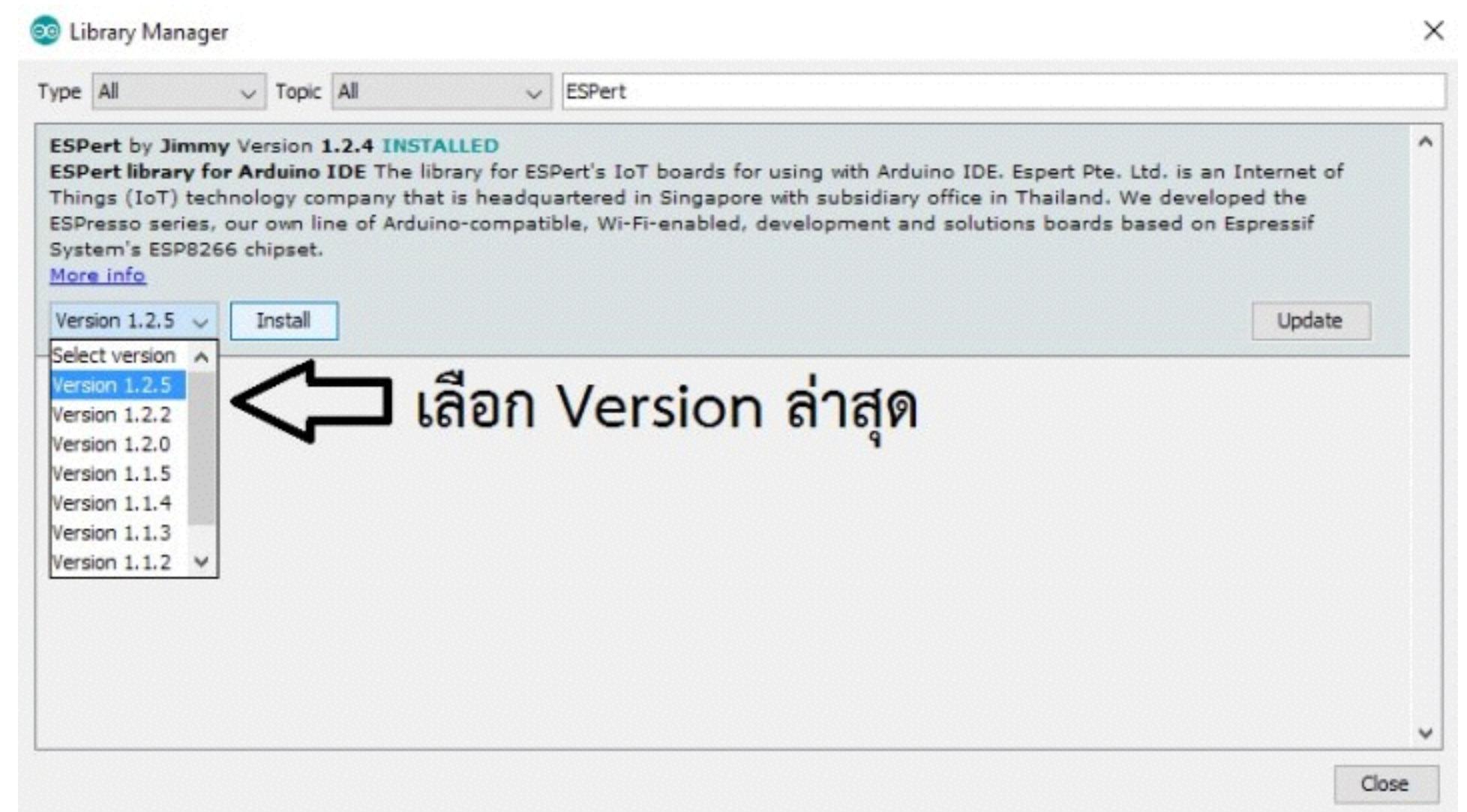
มีบอร์ด ESP8266 เพิ่มเข้ามาแล้ว

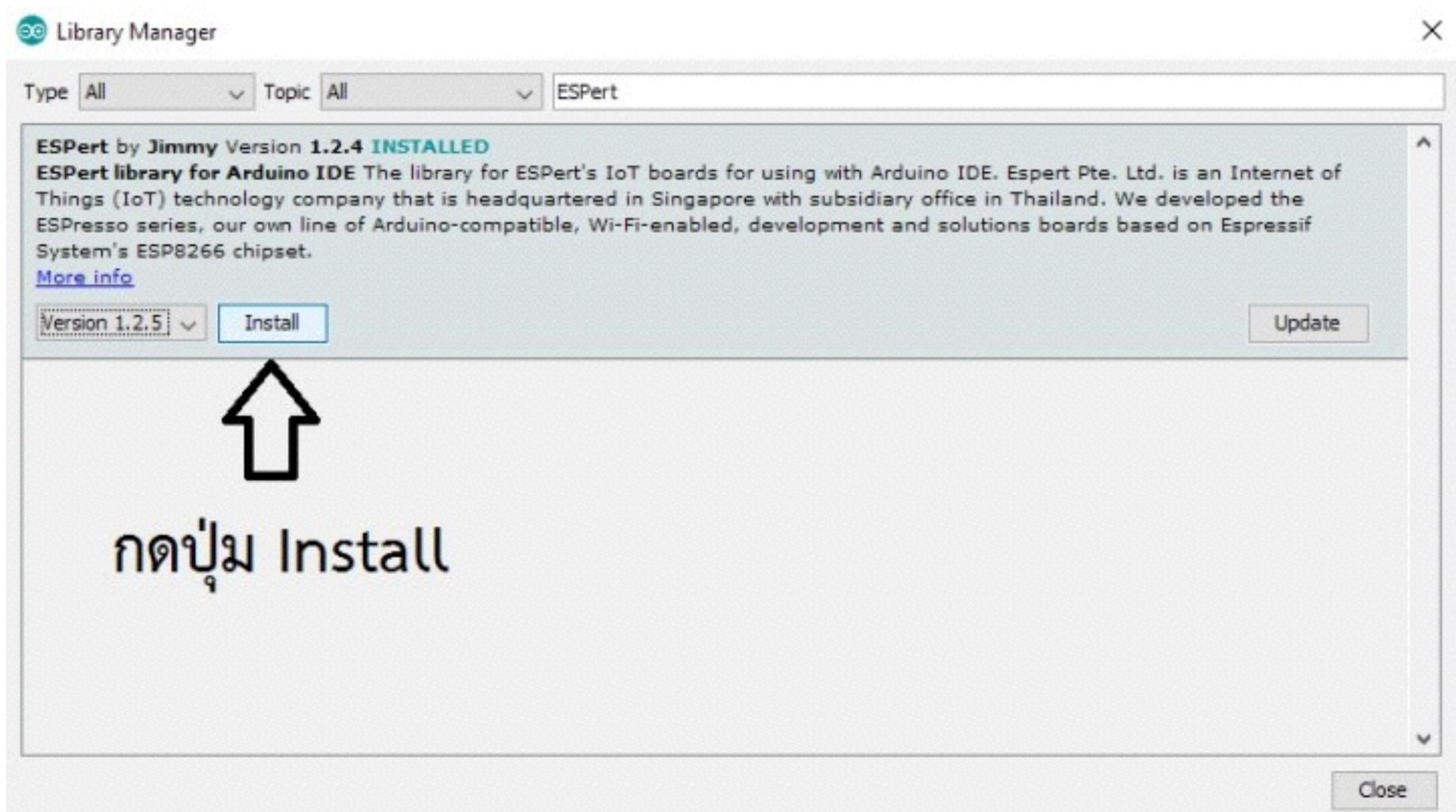


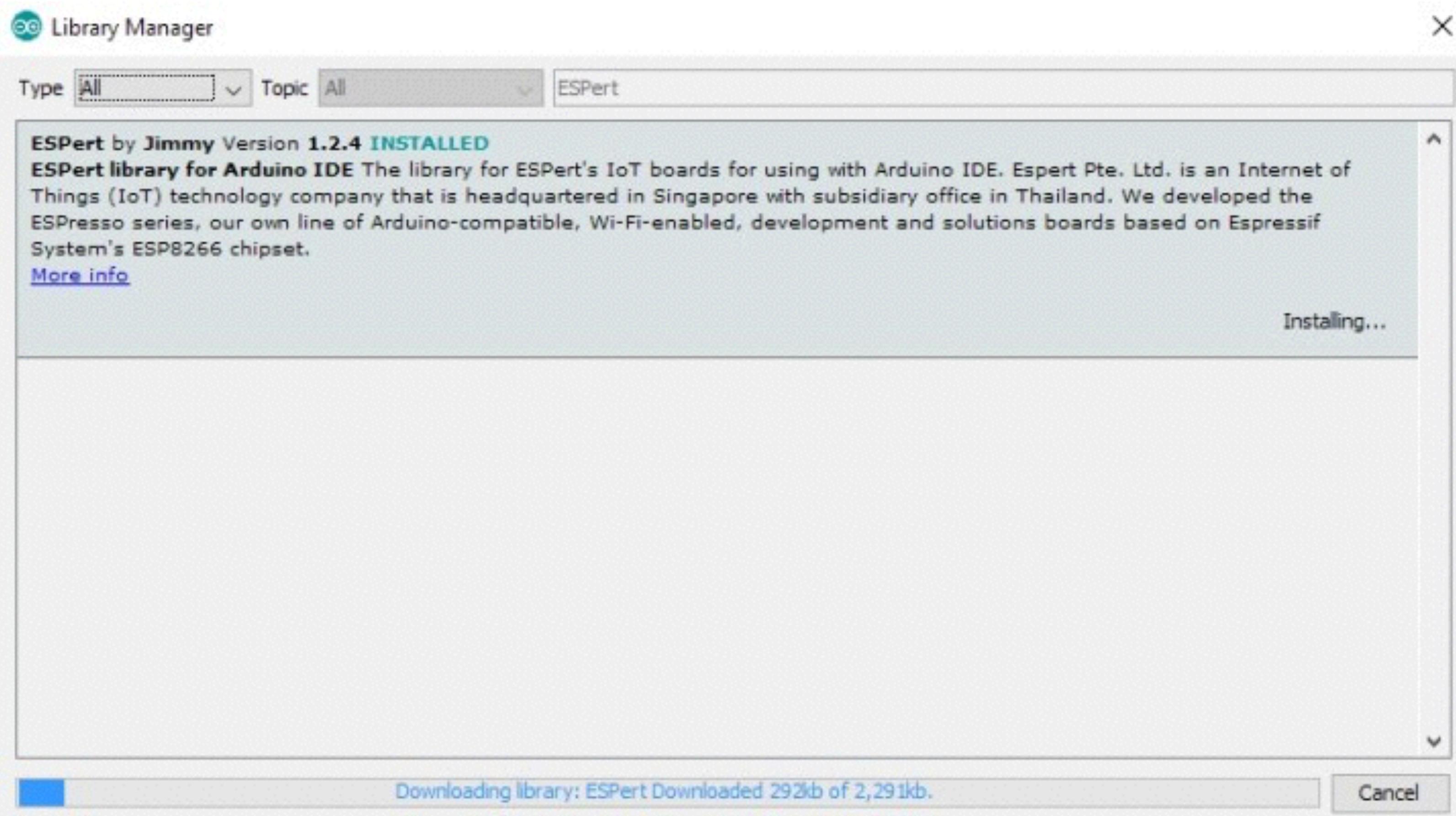


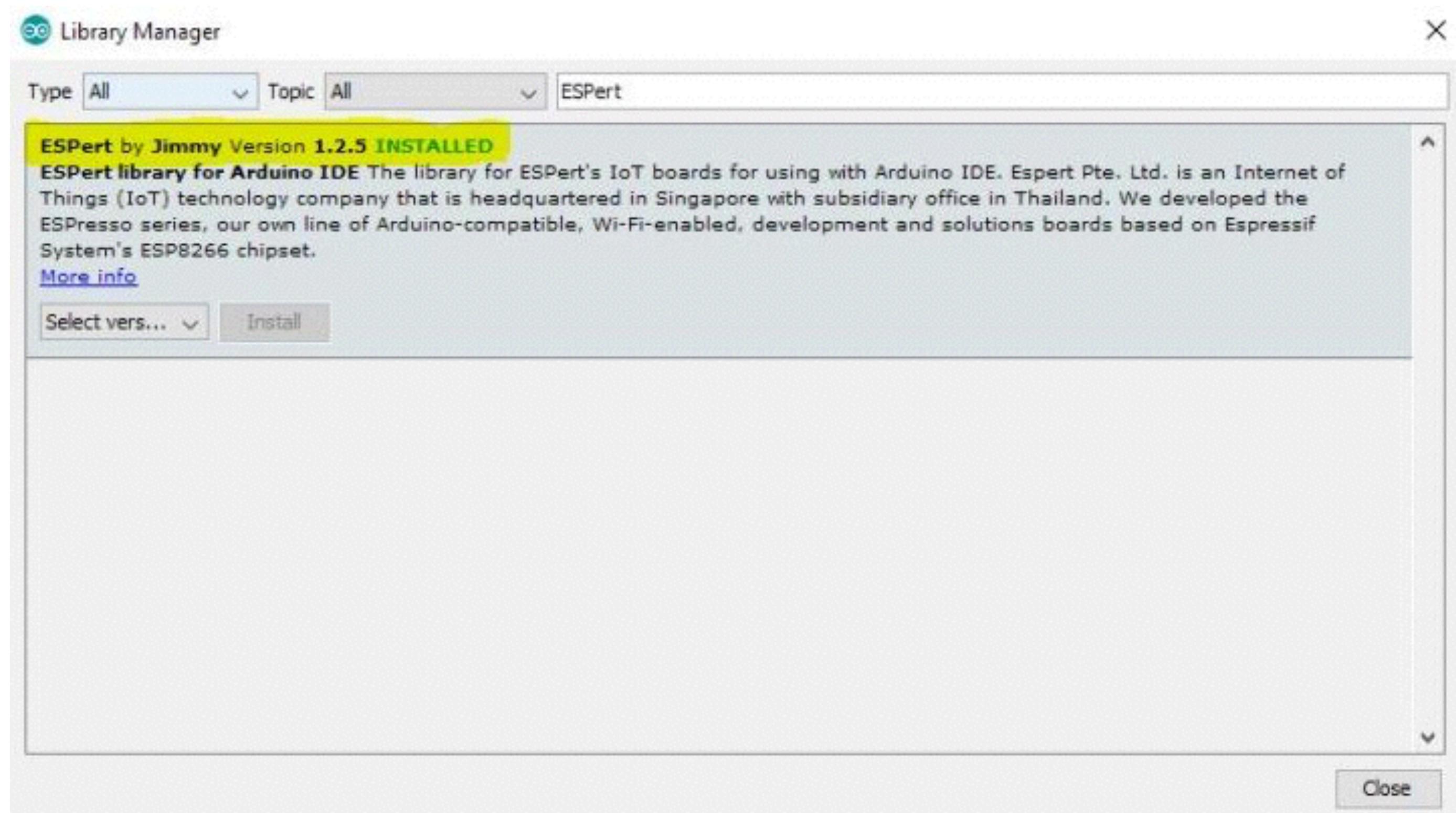












ESP8266

Arduino Framework

Connect to WiFi

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

- Title Bar:** esp8266_sta | Arduino 1.6.9
- Code Editor:** The code is named `esp8266_sta`. It includes #includes for `ESP8266WiFi.h` and `WiFiClient.h`, defines `ssid` and `password`, sets up pins, and performs a loop to connect to WiFi and print the IP address.
- Serial Monitor:** Shows the message "Done compiling." followed by the library path "Using library ESP8266WiFi at version 1.0 in folder /Users/luke/Downloads/packages/ESP8266/". Below that, it displays memory usage: "Sketch uses 225,873 bytes (21%) of program storage space. Maximum is 1,044,464 bytes. Global variables use 31,800 bytes (38%) of dynamic memory, leaving 50,120 bytes for local variables. Memory use is 225,873 bytes." The monitor also indicates the board is an "Espresso Lite 2.0, 80 MHz, 921600, 4M (3M SPIFFS), nodemcu, Disabled, None on /dev/cu.usbserial-DR00N13E".

STA WebServer



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

- Title Bar:** esp8266_webserver | Arduino 1.6.9
- Toolbar:** Standard Arduino toolbar with icons for file operations (New, Open, Save, Print, Find, Copy, Paste).
- Code Editor:** The main area displays the C++ code for an ESP8266 Webserver. The code includes setup and loop functions for connecting to WiFi, starting a MDNS responder, and handling HTTP requests. Lines of code are numbered from 33 to 72.

```
33     digitalWrite(led, 0);
34 }
35
36 void setup(void){
37     pinMode(led, OUTPUT);
38     digitalWrite(led, 0);
39     Serial.begin(115200);
40     WiFi.begin(ssid, password);
41     Serial.println("");
42
43     // Wait for connection
44     while (WiFi.status() != WL_CONNECTED) {
45         delay(500);
46         Serial.print(".");
47     }
48     Serial.println("");
49     Serial.print("Connected to ");
50     Serial.println(ssid);
51     Serial.print("IP address: ");
52     Serial.println(WiFi.localIP());
53
54 if (MDNS.begin("esp8266")) {
55     Serial.println("MDNS responder started");
56 }
57
58 server.on("/", handleRoot);
59
60 server.on("/inline", [](){
61     server.send(200, "text/plain", "this works as well");
62 });
63
64 server.onNotFound(handleNotFound);
65
66 server.begin();
67 Serial.println("HTTP server started");
68 }
69
70 void loop(void){
71     server.handleClient();
72 }
```

- Status Bar:** At the bottom left, it says "Done Saving."

Done Saving.

AP WebServer

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

- Title Bar:** esp8266_webserver | Arduino 1.6.9
- Toolbar:** Standard Arduino toolbar with icons for file operations (New, Open, Save, Print, Find, Copy, Paste, Undo, Redo).
- Code Editor:** The main area displays the C++ code for the `esp8266_webserver` sketch. The code includes setup for a LED, WiFi connection, MDNS responder, and an HTTP server, along with a loop to handle client connections.

```
esp8266_webserver §

33   digitalWrite(led, 0);
34 }
35
36 void setup(void){
37   pinMode(led, OUTPUT);
38   digitalWrite(led, 0);
39   Serial.begin(115200);
40   WiFi.begin(ssid, password);
41   Serial.println("");
42
43   // Wait for connection
44   while (WiFi.status() != WL_CONNECTED) {
45     delay(500);
46     Serial.print(".");
47   }
48   Serial.println("");
49   Serial.print("Connected to ");
50   Serial.println(ssid);
51   Serial.print("IP address: ");
52   Serial.println(WiFi.localIP());
53
54   if (MDNS.begin("esp8266")) {
55     Serial.println("MDNS responder started");
56   }
57
58   server.on("/", handleRoot);
59
60   server.on("/inline", [](){
61     server.send(200, "text/plain", "this works as well");
62   });
63
64   server.onNotFound(handleNotFound);
65
66   server.begin();
67   Serial.println("HTTP server started");
68 }
69
70 void loop(void){
71   server.handleClient();
72 }
```

- Status Bar:** At the bottom left, it says "Done Saving."

Basic HTTP Get

esp8266_basic_httpget | Arduino 1.6.9

```
27 Serial.println("");
28 Serial.print("Connected to ");
29 Serial.println(ssid);
30 Serial.print("IP address: ");
31 Serial.println(WiFi.localIP());
32
33 }
34
35 void loop() {
36   HTTPClient http;
37
38   Serial.print("[HTTP] begin...\n");
39   http.begin("http://192.168.1.12/test.html"); //HTTP
40
41   Serial.print("[HTTP] GET...\n");
42   // start connection and send HTTP header
43   int httpCode = http.GET();
44
45   // httpCode will be negative on error
46   if (httpCode > 0) {
47     // HTTP header has been send and Server response header has been handled
48     Serial.printf("[HTTP] GET... code: %d\n", httpCode);
49
50     // file found at server
51     if (httpCode == HTTP_CODE_OK) {
52       String payload = http.getString();
53       Serial.println(payload);
54     }
55   } else {
56     Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
57   }
58
59   http.end();
60
61
62   delay(10000);
63 }
```

No valid hardware definitions found in folder esp8266com.
WARNING: Error loading hardware folder /Users/Nat/Documents/Arduino/hardware/esp8266com
No valid hardware definitions found in folder esp8266com.
/Users/Nat/labesp8266/basic-esp8266/esp8266_basic_httpget/esp8266_basic_httpget.ino

20 ESPRESSO LITE 2.0, 80 MHZ, 115200, 4M (3M SPIFFS), ck, Disabled, None on /dev/cu.usbserial-A7CX2S3 CHANG MAI MAKERCLUB

LAB 1

cmmc.io/netpie/freeboard

netpie-freeboard(html5)

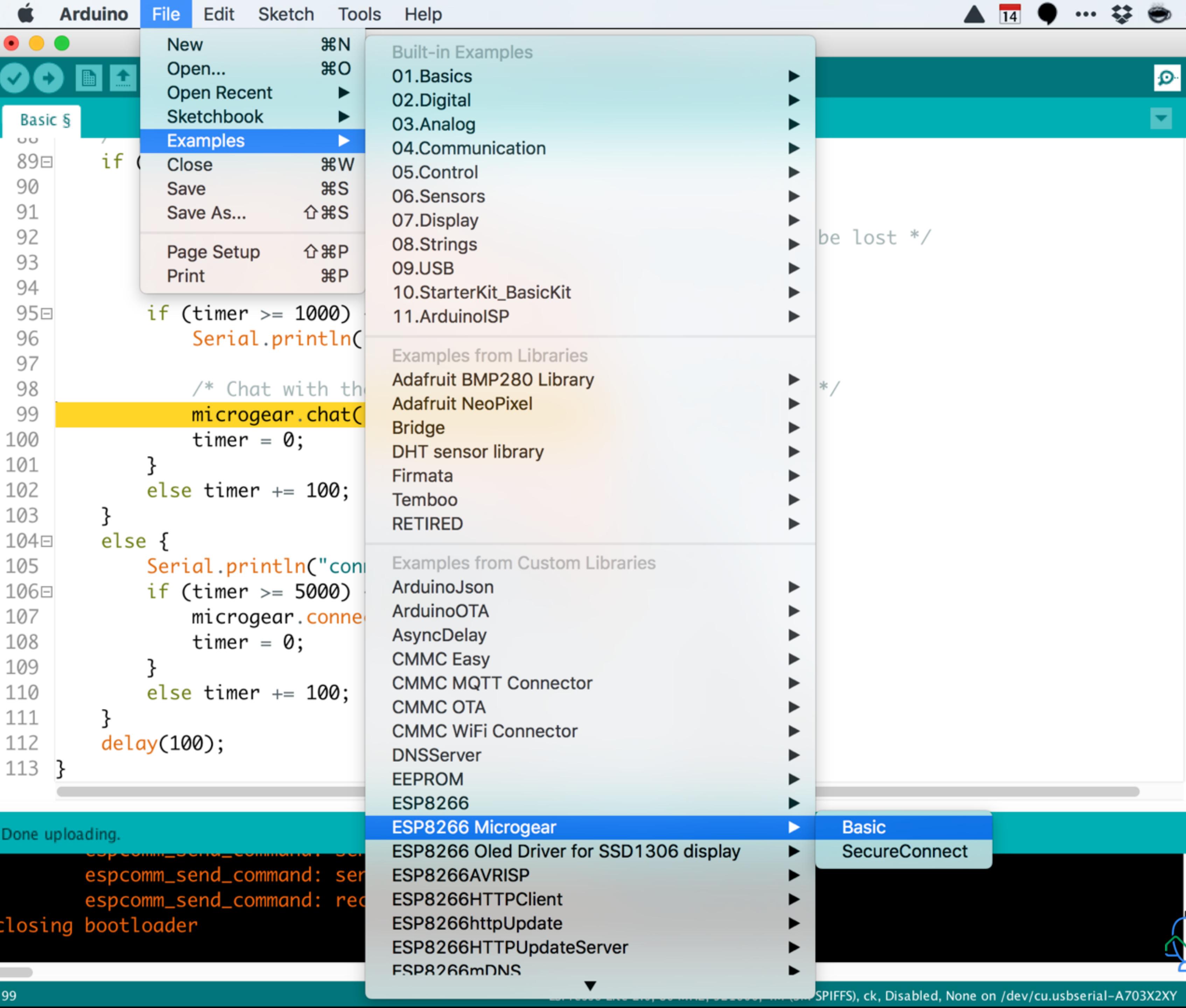
APPID : HelloNETPIE

Key : 2BrGVTbajdZFOSz

Secret : xIzR8yQ9RRp13T7KkPMt33sn7

```
microgear[“MR”].chat(“plug001”, “ON”);
```

CODING!



netpie (devicekey)

APPID : HelloNETPIE

Key : 60Rk0Ct2daXfarJ

Secret : X6Cnwb4cQCHBItijCoBre8SKn

Basic | Arduino 1.6.9

```
// To check if the microgear is still connected
89if (microgear.connected()) {
90    Serial.println("connected");
91
92    /* Call this method regularly otherwise the connection may be lost */
93    microgear.loop();
94
95if (timer >= 1000) {
96    Serial.println("Publish...");
97
98    /* Chat with the microgear named ALIAS which is myself */
99    microgear.chat("NAT/TEST", "Hello");
100   timer = 0;
101 }
102 else timer += 100;
103 }
104else {
105    Serial.println("connection lost, reconnect...");
106if (timer >= 5000) {
107    microgear.connect(APPID);
108    timer = 0;
109 }
110 else timer += 100;
111 }
112 delay(100);
113 }
```

Done uploading.

```
espcomm_serial_command: sending command header
espcomm_send_command: sending command payload
espcomm_send_command: receiving 2 bytes of data
closing bootloader
```

99

ESPRESSO Lite 2.0, 80 MHz, 921600, 4M (3M SPIFFS), ck, Disabled, None on /dev/cu.usbserial-A703X2XY



Basic | Arduino 1.6.9

```
Basic §
12 //SECRET - XCHMB4EQRHBTJGQBPE5K
13 #define ALIAS "NAT-DEVICE"
14
15 WiFiClient client;
16
17 int timer = 0;
18 MicroGear microgear(client);
19
20 /* If a new message arrives, do this */
21 void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen) {
22     Serial.print("Incoming message --> ");
23     msg[msglen] = '\0';
24     Serial.println((char *)msg);
25 }
26
27 void onFoundgear(char *attribute, uint8_t* msg, unsigned int msglen) {
28     Serial.print("Found new member --> ");
29     for (int i=0; i<msglen; i++)
30         Serial.print((char)msg[i]);
31     Serial.println();
32 }
33
34 void onLostgear(char *attribute, uint8_t* msg, unsigned int msglen) {
35     Serial.print("Lost member --> ");
36     for (int i=0; i<msglen; i++)
37         Serial.print((char)msg[i]);
```

Done uploading.

```
espcomm_serial_command: sending command header
espcomm_send_command: sending command payload
espcomm_send_command: receiving 2 bytes of data
closing bootloader
```