Lab # 11

Introduction to IoT using ESP32 Development Board

Objectives

- Create Hello world program
- Make ESP32 as web server to control data
- Get data from ESP32 on local network

Tools

- Arduino
- ESP32 Board
- 2 LEDs
- DHT11 sensor

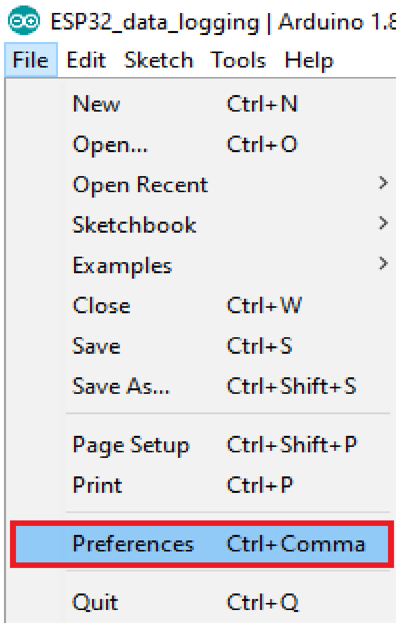
Pre Lab

Please go through the data sheet of Expressif ESP32 development board.

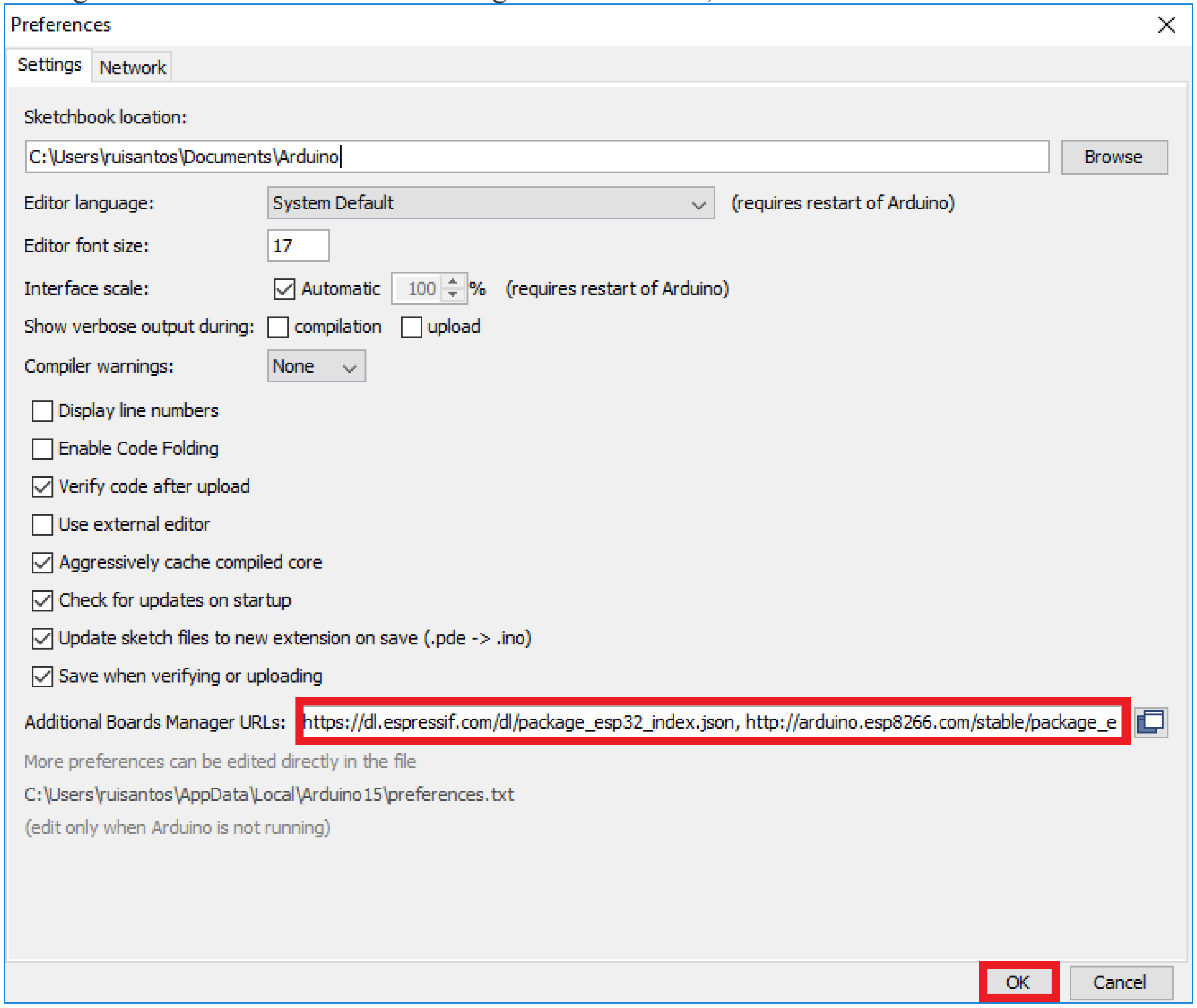
Installing ESP32 Add-on in Arduino IDE

To install the ESP32 board in your Arduino IDE, follow these next instructions:

1. In your Arduino IDE, go to **File**> **Preferences**



2. Enter **https://dl.espressif.com/dl/package_esp32_index.json** into the "Additional Board Manager URLs" field as shown in the figure below. Then, click the "OK" button:



Preferences

Settings | Network

Sketchbook location: C:\Users\ruisantos\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 17

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

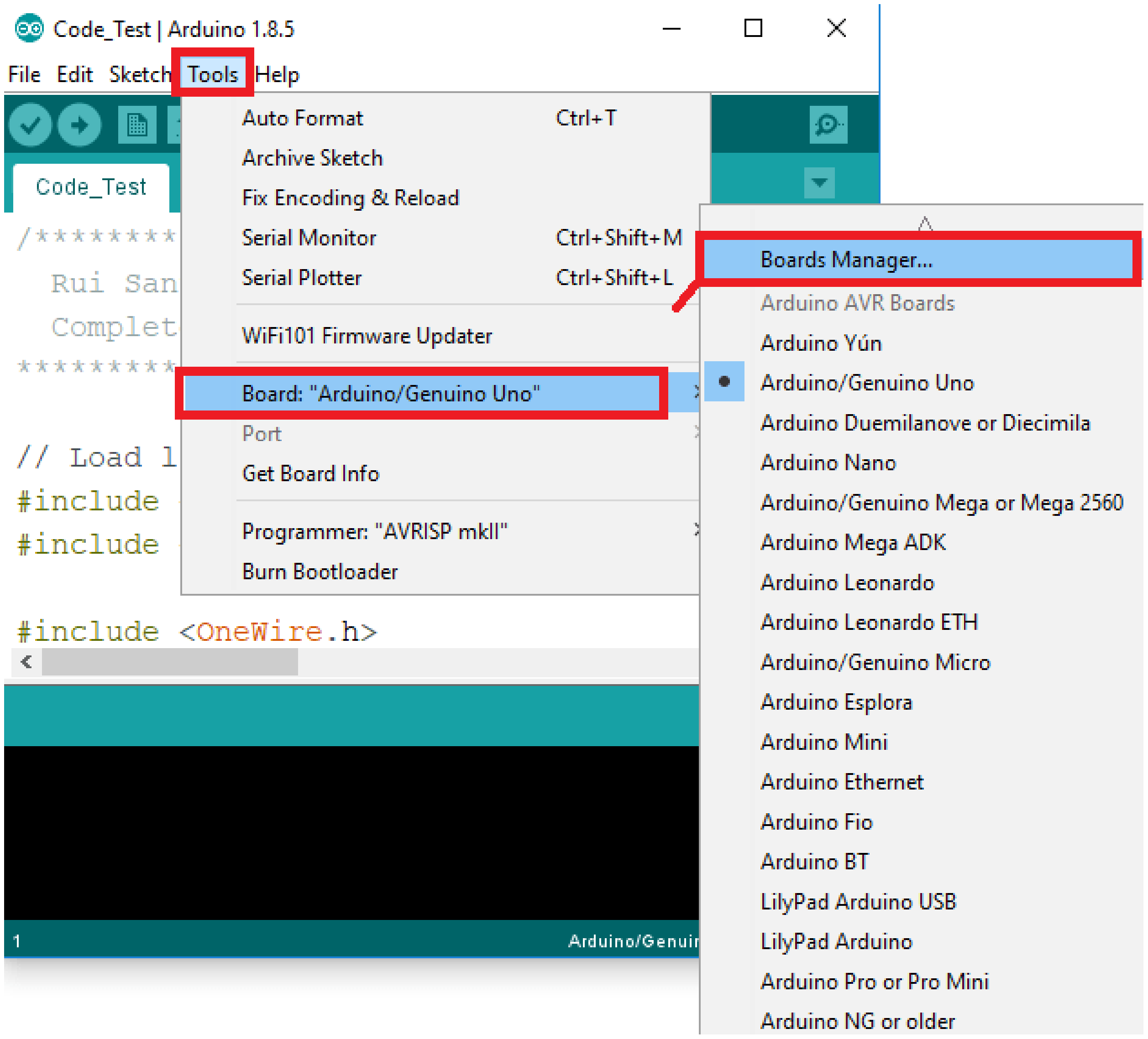
☐ Display line numbers
☐ Enable Code Folding
☑ Verify code after upload
☐ Use external editor
☑ Aggressively cache compiled core
☑ Check for updates on startup
☑ Update sketch files to new extension on save (.pde -> .ino)
☑ Save when verifying or uploading

Additional Boards Manager URLs: https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_e

More preferences can be edited directly in the file
C:\Users\ruisantos\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Note: if you already have the ESP8266 boards URL, you can separate the URLs with a comma as follows:
https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_index.json

3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**

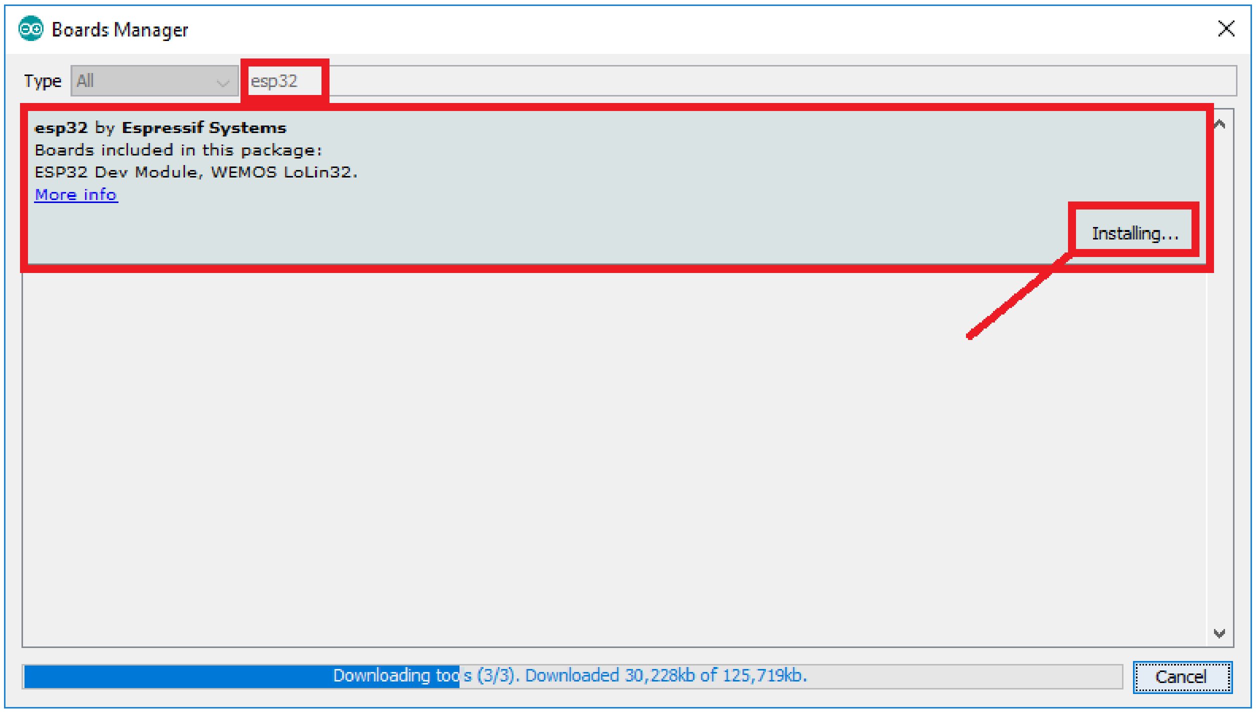


```
/********
  Rui San
  Complet
*********

// Load l
#include
#include

#include <OneWire.h>
```

4. Search for **ESP32** and press install button for the “**ESP32 by Espressif Systems**“:



esp32 by Espressif Systems
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.
More info

Downloading tools (3/3). Downloaded 30,228kb of 125,719kb.

After board installation is complete, write your first led blinking code.

Code: Pin 2 of ESP32 board is connected to LED on Pin 2 by default.

```
int ledPin = 2;
void setup()
{
  pinMode(ledPin, OUTPUT);
}
void loop()
{
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin, LOW);
  delay(500);
}
```

In-Lab Task 1:

Please ESP32 board and run your first program of led blinking on it.

In-Lab Task 2:

Control of two leds connected at pin 4 and 5 of ESP32 from local server.

Code:

```
// Load Wi-Fi library
#include <WiFi.h>

/* Put your SSID & Password */
const char* ssid = "ESP32";  // Enter SSID here
const char* password = "12345678";  //Enter Password here
// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current output state
String output4State = "off";
String output5State = "off";

// Assign output variables to GPIO pins
const int output4 = 4;
const int output5 = 5;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(output4, OUTPUT);
  pinMode(output5, OUTPUT);
  // Set outputs to LOW
  digitalWrite(output4, LOW);
  digitalWrite(output5, LOW);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available();   // Listen for incoming clients

  if (client) {                             // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");          // print a message out in the serial port
    String currentLine = "";                // make a String to hold incoming data from the client
    while (client.connected() && currentTime - previousTime <= timeoutTime) {  // loop while the
client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline character
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // turns the GPIOs on and off
            if (header.indexOf("GET /4/on") >= 0) {
              Serial.println("GPIO 5 on");
              output4State = "on";
              digitalWrite(output4, HIGH);
            } else if (header.indexOf("GET /4/off") >= 0) {
              Serial.println("GPIO 4 off");
              output4State = "off";
              digitalWrite(output4, LOW);
            } else if (header.indexOf("GET /5/on") >= 0) {
              Serial.println("GPIO 5 on");
              output5State = "on";
              digitalWrite(output5, HIGH);
            } else if (header.indexOf("GET /5/off") >= 0) {
              Serial.println("GPIO 5 off");
              output5State = "off";
              digitalWrite(output5, LOW);
            }

            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta  name=\"viewport\"  content=\"width=device-width,  initial-
scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and font-size attributes to fit your preferences
            client.println("<style>html  {  font-family:  Helvetica;  display:  inline-block;  margin:  0px
auto; text-align: center;}");
            client.println(".button { background-color: #4CAF50; border: none; color: white; padding:
16px 40px;");
            client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
            client.println(".button2 {background-color: #555555;}</style></head>");

            // Web Page Heading
            client.println("<body><h1>ESP32 Web Server</h1>");

            // Display current state, and ON/OFF buttons for GPIO 4
            client.println("<p>GPIO 4 - State " + output4State + "</p>");
            // If the output4State is off, it displays the ON button
            if (output4State=="off") {
              client.println("<p><a href=\"/4/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a                    href=\"/4/off\"><button                    class=\"button
button2\">OFF</button></a></p>");
            }

            // Display current state, and ON/OFF buttons for GPIO 5
            client.println("<p>GPIO 5 - State " + output5State + "</p>");
            // If the output5State is off, it displays the ON button
            if (output5State=="off") {
              client.println("<p><a href=\"/5/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a                    href=\"/5/off\"><button                    class=\"button
button2\">OFF</button></a></p>");
```

```
            }
            client.println("</body></html>");

            // The HTTP response ends with another blank line
            client.println();
            // Break out of the while loop
            break;
          } else { // if you got a newline, then clear currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage return character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

Post-Lab Task 3:

Critical Analysis / Conclusion

(By Student about Learning from the Lab)

Lab Assessment

Pre Lab			**/1**	**/10**
In Lab			**/5**	
Post Lab	**Data Analysis**	**/4**	**/4**	
	Data Presentation	**/4**		
	Writing Style	**/4**		
Instructor Signature and Comments				