*J. Myanmar Acad. Arts Sci. 2021* **Vol. XIX. No.2**

# CONTROLLING THE APPLIANCES AND MONITORING DATA FROM LOCAL WEB SERVER USING ESP32 AND HTML

Nu Nu Swe[1], Mi Yin Sa Nine[2], Hnin Aye Khine[3], Ohnmar Soe[4]

## Abstract

Physical data can be sending (writing) to or receiving (reading) from the web server using ESP32 which is a device of combination of microcontroller and Wi-Fi, generally. Physical data such as temperature, humidity and voltage are detected (read condition) from the DHT11 sensor and potentiometer by using microcontroller and these are monitored on web page. And then, the home appliance can be controlled by the touching or clicking the word or symbol on the web page on laptop or smart phone as writing action. In this research, a LED is controlled instead of the home appliance and the potentiometer is used behalf of voltage source to be measured. ESP32 is compatible to Arduino IDE but ESP32 board is not included in Arduino IDE as default. The board is installed in the IDE. Coding composes of data writing to / reading from web, connecting to web server and web page management. Web page is created by using hypertext markup language (HTML).

**Keywords:** ESP 32, Wi-Fi, Read/write, client, web server, HTML.

## Introduction

The data can be writing to or read on the web server with the help of ESP32. Arduino IDE can be compatible to ESP32. There are no ESP boards in the default Arduino IDE. The ESP boards are needed to include in Arduino IDE. The physical data (such as voltage, temperature, humidity) are detected from the potentiometer and DHT 11 and then data are visualized on the web page using the local Wi-Fi router or Wi-Fi hot spot of phone. By the writing section, the electrical appliance can be controlled from the web page on the specified webserver. Programmer code is written by Arduino language and HTML.

## Materials Required

In this research, ESP32 is used as the main processing unit and DHT11, potentiometer and electrical appliance (such as LED) are used to implement the data reading /writing action.

### Web Server

A web server processes incoming network requests over Hypertext Transfer Protocol (HTTP) and several other related protocols. The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP). Web servers can frequently be found embedded in printers, routers and webcams which are serving only a local area network (LAN). The web server may then be used as a part of a system for monitoring or administering the device. This usually means that additional software not to be installed on the client computer since only a web browser which is included the most required operating system.

[1] Dr, Associate Professor, Department of Physics, Yangon University of Education
[2] Dr, Lecturer, Department of Physics, Myeik University
[3] Dr, Lecturer, Department of Physics, Yangon University of Education
[4] Dr, Lecturer, Department of Physics, Yangon University of Education

## ESP32

ESP32 development board is equipped with ESP-WROOM-32 module which contains dual-core 32-bit microprocessor attached with Wi-Fi and dual mode Bluetooth. Moreover, consists of two CPU cores which can be controlled individually. It operates at 80 MHz to 240MHz adjustable clock speed and performs 600 Dhrystone million instructions per second (DMIPS).

ESP32, is the various memory types which are 448KB for ROM, 520KB of SRAM and 4MB of flash memory. In sleep mode condition, 8KB memory can be used for real time controller fast/ slow SRAM. These memories are sufficient to run with the string that make web pages, JSON/XML data is utilized in Internet of Things(IoT)devices. ESP32 integrates Wi-Fi transceiver, classic Bluetooth and Bluetooth low energy (BLE) so that it can connect to Wi-Fi network and interact with internet. ESP32 can set up its own network which can directly be connected by other devices.

ESP32 possesses the multiplexed general-purpose input output (GPIO)pins which are shown in Table 1.

**Table 1 Peripheral Input /Output of ESP32**

| No | Pin Name | Function |
|---|---|---|
| 1. | Touch pad | Capacitance touch sensing |
| 2. | ADC | (Analog-to-converter) channels of 12- bit SAR-ADC. The ADC range can be varied 0-1V,0-1.4V,0-2V and 0-4V. |
| 3. | DAC | (Digital-to-analog converter) channels of 8-bit |
| 4. | I2C | (inter-integrated circuit) uses to interface the sensors |
| 5. | UART | (Universal asynchronous receiver/transmitter). One is used to load serially. |
| 6. | CAN2.0 | Controller area network |
| 7. | SPI | (Serial peripheral interface) use to interface the sensors. |
| 8. | I2S | (inter-integrated sound) use to interface sound |
| 9. | RMII | (Reduced media-independent interface) |
| 10. | PWM | (Pulse width modulation) pins for controlling device |

ESP32 is highly integrated with built-in antenna switches, RF, power amplifier, low-noise receive amplifier, filters, and power management modules. The internal block diagram is shown in Figure1.

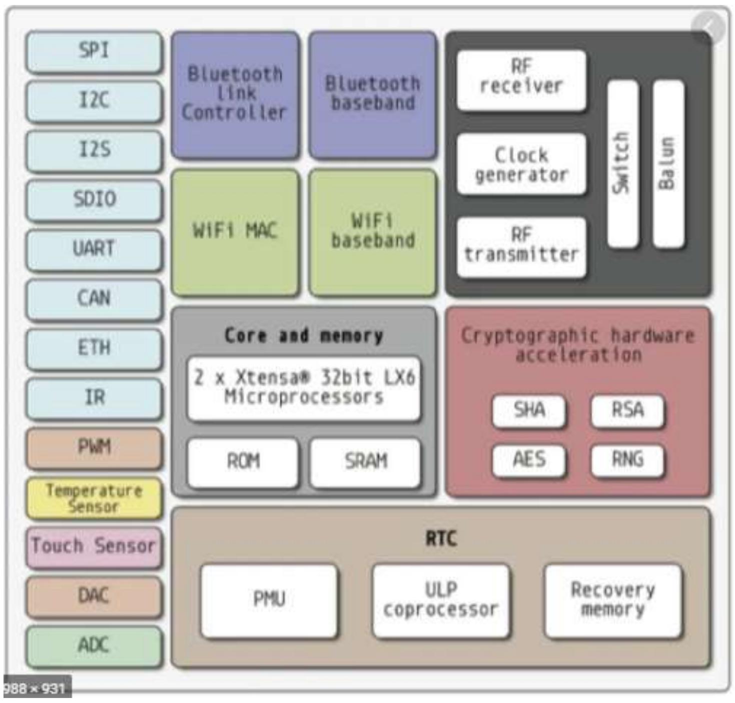


**Figure 1** Function Block Diagram of ESP32

## DHT 11

The DHT11 is a digital temperature and humidity sensor as shown in Figure 2. It is composed of a capacitive humidity sensor and a thermistor to measure the surrounding air, and a digital signal is produced on the data pin. Table 1 shows the pin description of DHT11. Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor. It is about 4ms for one communication process. The data consists of decimal and integral parts. A complete data transmission is 40bit, and the sensor sends higher data bit first. Data format is the sum of 8bit integral RH data, 8bit decimal RH data, 8bit integral T data, 8bit decimal T data and 8bit check sum. If the data transmitted correctly, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data". Data transmission is expressed in Figure 3.

**Table 2 Pin Description of DHT 11**

| Pin No | Pin Name | Pin Description |
|---|---|---|
| 1 | VCC | Power supply 3.3 to 5.5 Volt DC |
| 2 | DATA | Digital output pin |
| 3 | NC | Not in use |
| 4 | GND | Ground |

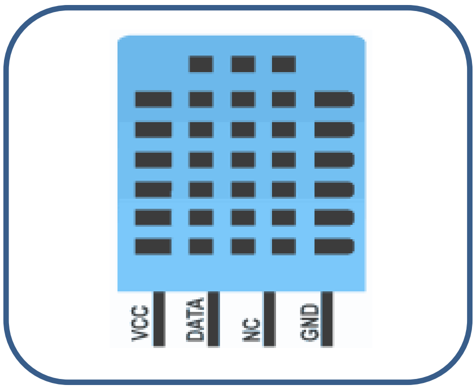


**Figure 2** The Photo of DHT 11

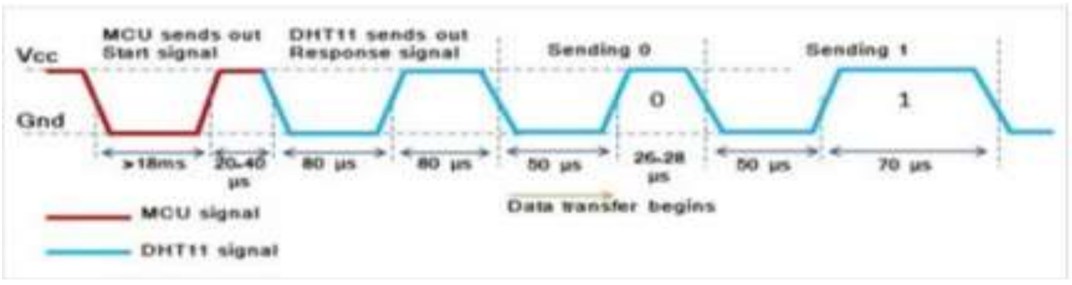


**Figure 3** Data Transmission

## Operation

In this research, the temperature and humidity are detected from DHT11 sensor and the voltage is detected from potentiometer by ESP32. The data are transferred to the webserver via Wi-Fi which is associated with ESP32. The web page style can be designed in option. There are two main sections hardware preparation and software preparation for implementation of data monitoring on web server.

## Hardware Preparation

There is more GPIOs in ESP32 which is more functionalities compared to the ESP8266. For the ESP32, one can decide which pins are UART, I2C, or SPI – it is needed to set that on the code. This is possible due to the ESP32 chip's multiplexing feature that allows to assign multiple functions to the same pin. If the pins are not set them on the code, the pins will be used as default as shown in Figure 2. DHT 11 is a sensor which produces the digital output. The data pin of DHT 11 is connected to GPIO 23 pin of ESP32, the wiper pin of potentiometer is connected to GPIO 34 pin to read the value of voltage. The LED is wired to GPIO 2 pin to be controlled by clicking or touching the switch or symbol on web page. The schematic circuit diagram is shown in Figure 4. The Figure 5 shows the block diagram of the working principle of data monitoring on web server via Wi-Fi.

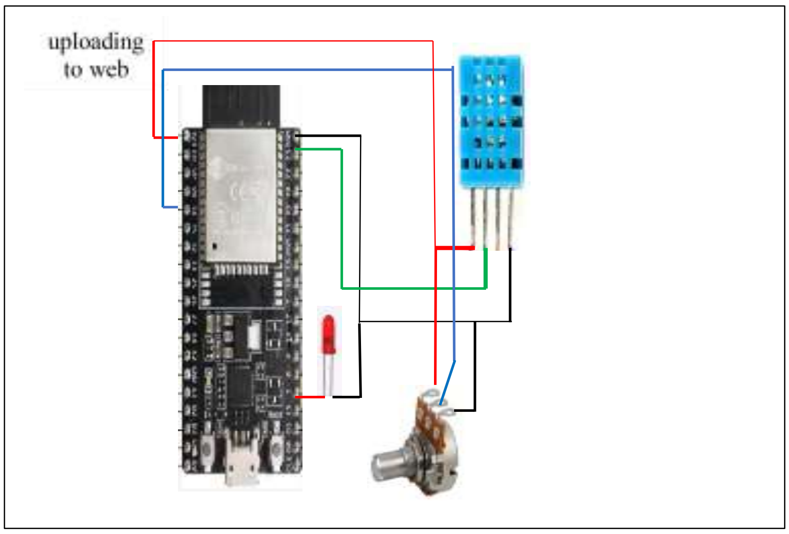


**Figure 4** Schematic Circuit Connection

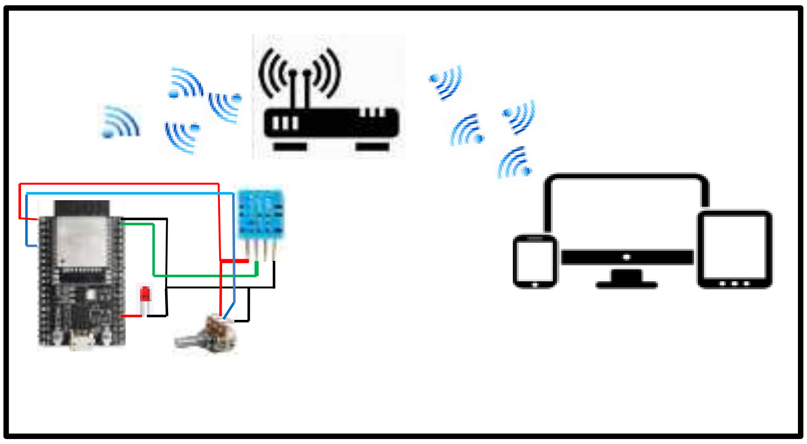
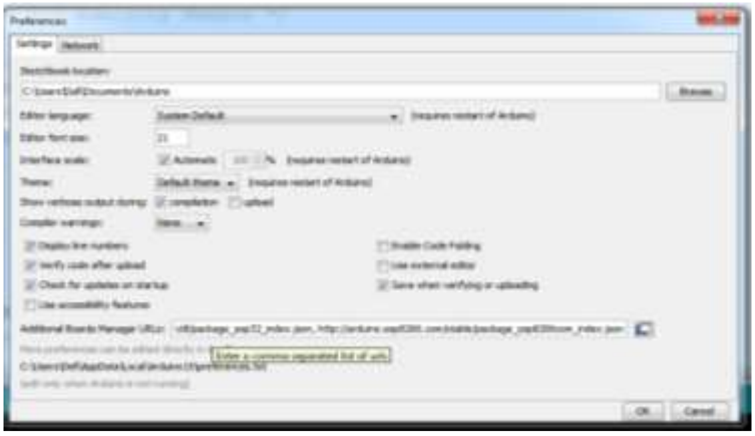


**Figure 5** Block Diagram of Data Read/ Write

## Software Preparation

First of all, ESP32 library and ESP32 board must be added in Arduino IDE by the following procedures.

1) In Arduino IDE, go to File> Preferences.
2) Https://dl.espressif.com/dl/package_esp32_index.json enter into the "Additional Board Manager URLs" field as shown in the figure 6. Then, Click the "OK" Button.
3) Open the board manager: tool>board>board manger and install ESP32 as in Figure 7.



Preferences

Settings | Network

Sketchbook location: [illegible]

Editor language: System Default (requires restart of Arduino)

Editor font size: 21

Interface scale: ✓ Automatic [illegible] % (requires restart of Arduino)

Theme: Default theme (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

✓ Check for updates on startup ✓ Save when verifying or uploading

☐ Use accessibility features

Additional Boards Manager URLs: [illegible]package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

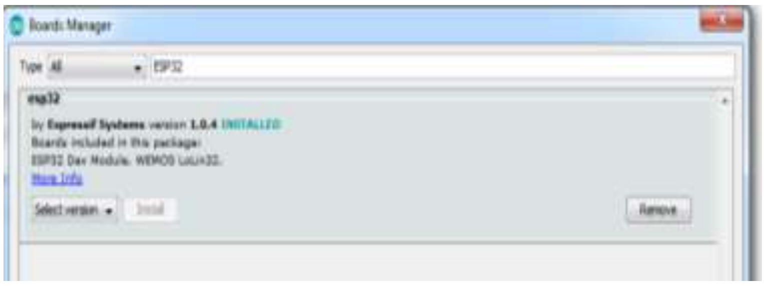
Enter a comma separated list of urls

[illegible]preferences.txt

(edit only when Arduino is not running)

OK | Cancel

**Figure 6** Including ESP32



Boards Manager

Type All ESP32

esp32

by **Espressif Systems** version **1.0.4** INSTALLED

Boards included in this package:

ESP32 Dev Module, WEMOS LoLin32.

More Info

Select version | Install | Remove

**Figure 7** Installing ESP32

Software coding composes of four sections; 1) accessing web server, 2) data reading, 3) appliance controlling(writing) and 4) web page preparation.

DHT11 library is used to detect the temperature and humidity from sensor.

```
#include "DHT.h"
#define DHTPIN 23
#define DHTTYPE DHT11
```

Temperature and humidity data are obtained by using the following commands.

```
 DHT dht(DHTPIN, DHTTYPE);
 dht.readTemperature( )
 dht.readHumidity( )
```

To get the value of analog voltage

```
  #define ANALOG_PIN_0 34
  int analog_value = 0;
  float volt=0;
  analog_value =0;
  analogRead(ANALOG_PIN_0);
   volt= analog_value*5/4096;
```

To connect Wi-Fi set point, the following instructions are used;

```
#include <Wi-Fi.h>
const char* ssid     = "xxxxxxxxxxxx";
const char* password = "xxxxxxxxxxx";
Wi-FiServer server (80);
```

To connect Web server and find out the IP address,

```
Wi-Fi.begin(ssid, password);
while (Wi-Fi.status() != WL_CONNECTED)
{
delay (500);
Serial.print(".");
}
Serial.println("");
Serial.println("Wi-Fi connected.");
Serial.println("IP address: ");
Serial.println(Wi-Fi.localIP());
  server.begin();
}
```

Sending data to web server and web page style preparation are as follows;

```
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:,\">");
client.println("</style></head><body><h1>ESP32 WEb Server data sending and
receiving</h1>");
client.println("<h2>Dr Nu Nu Swe, Associate Professor, YUOE.</h2>");
client.print("Current Temperature  is: ");
client.print(localTemp);
client.print("  oC<br>");
client.print("Current Humidity is    : ");
client.print(localHum);
client.print(" % <br>");
client.print("<br>");
client.print("Voltage measured:     ");
client.print(volt);
client.print("V");
        client.print("<br>");
        client.print("<br>");
```

Receiving data to control the appliance (LED) is as;

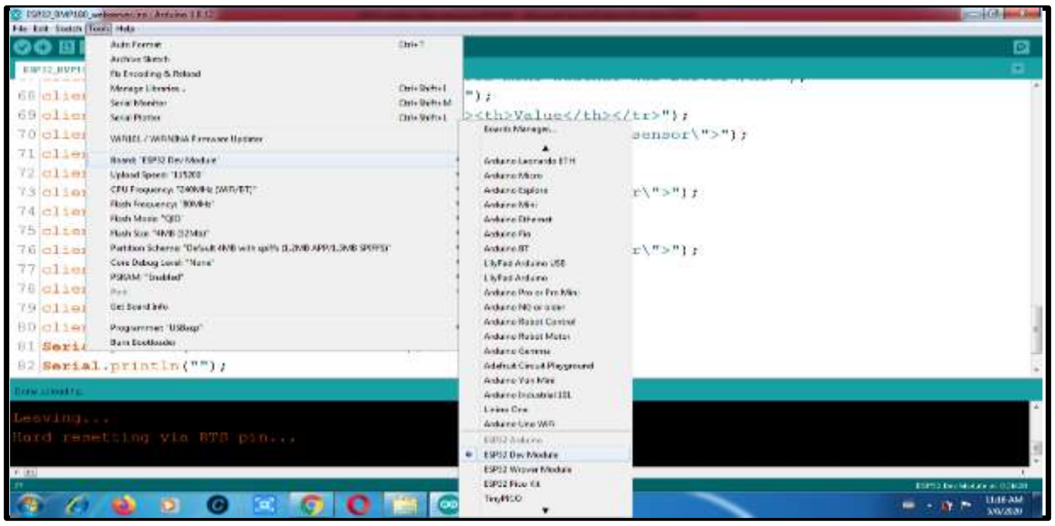
```
client.print("Click <a href=\"/H\">ON</a> to turn the LED on.<br>");
 client.print("Click <a href=\"/L\">OFF</a> to turn the LED off.<br>");
```

## Results

After the completion of circuit connection, the program is uploaded as following procedures;

1) **Tools > Board** and select ESP**32 Dev Module** in Figure 8**.**
2) **Tools > Port** and select the COM port which ESP32 to be connected. Click the upload button.
3) Press the ESP32 on board BOOT button after seeing “connecting” dots on debugging window in Figure 9.

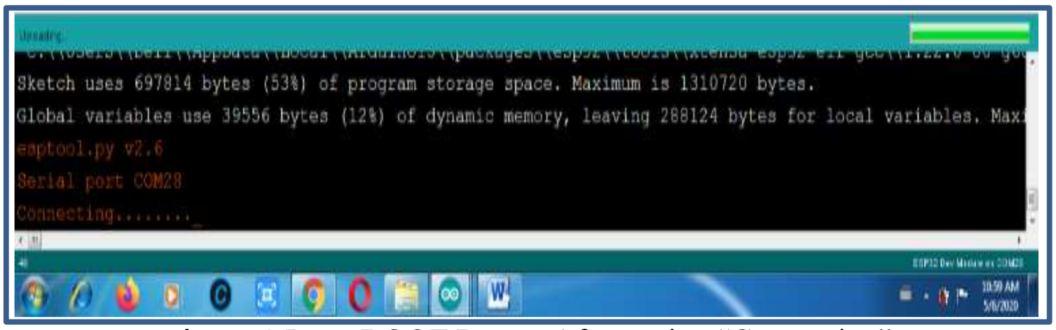
After uploading. Then, serial monitor is opened and press RST button to obtain IP address as shown in Figure 10.



```
Board: "ESP32 Dev Module"
Upload Speed: "115200"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Programmer: "USBasp"
```

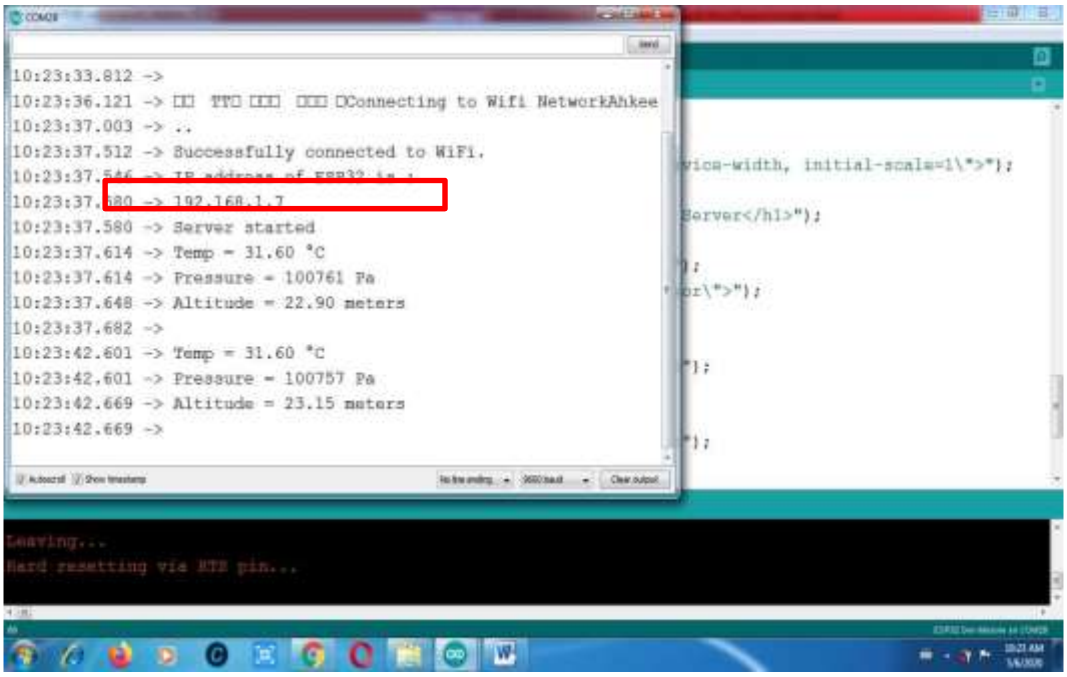
```
Serial.println("");

Leaving...
Hard resetting via RTS pin...
```

**Figure 8** Selection Board



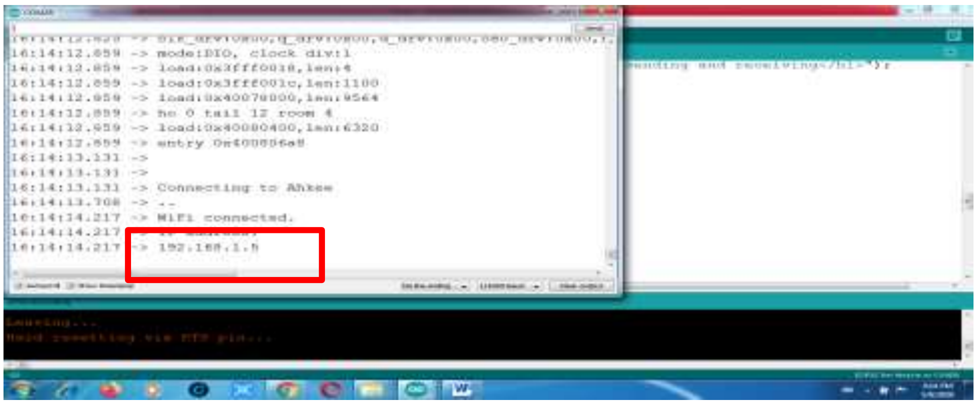
```
Uploading...
Sketch uses 697814 bytes (53%) of program storage space. Maximum is 1310720 bytes.
Global variables use 39556 bytes (12%) of dynamic memory, leaving 288124 bytes for local variables. Maxi
esptool.py v2.6
Serial port COM28
Connecting........_
```

**Figure 9** Press BOOT Button After seeing “Connection”



```
10:23:33.812 ->
10:23:36.121 -> □□  TTO □□□  □□□ □Connecting to Wifi NetworkAhkee
10:23:37.003 -> ..
10:23:37.512 -> Successfully connected to WiFi.
10:23:37.546 -> IP address of ESP32 is :
10:23:37.580 -> 192.168.1.7
10:23:37.580 -> Server started
10:23:37.614 -> Temp = 31.60 *C
10:23:37.614 -> Pressure = 100761 Pa
10:23:37.648 -> Altitude = 22.90 meters
10:23:37.682 ->
10:23:42.601 -> Temp = 31.60 *C
10:23:42.601 -> Pressure = 100757 Pa
10:23:42.669 -> Altitude = 23.15 meters
10:23:42.669 ->

Leaving...
Hard resetting via RTS pin...
```

**Figure 10** Pressing RST Button to get IP Address

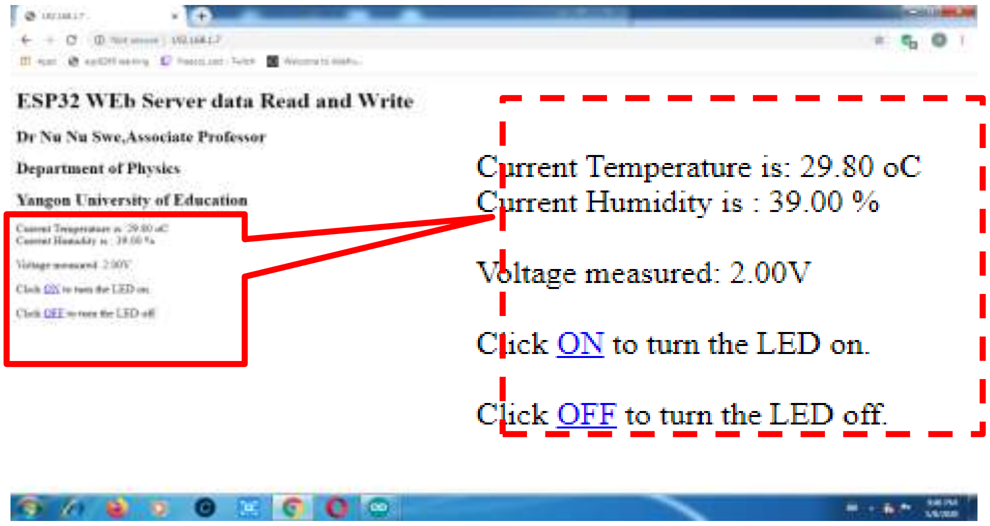


```
16:14:12.859 -> mode:DIO, clock div:1
16:14:12.859 -> load:0x3fff0018,len:4
16:14:12.859 -> load:0x3fff001c,len:1100
16:14:12.859 -> load:0x40078000,len:9564
16:14:12.859 -> ho 0 tail 12 room 4
16:14:12.859 -> load:0x40080400,len:6320
16:14:12.859 -> entry 0x400806a8
16:14:13.131 ->
16:14:13.131 ->
16:14:13.131 -> Connecting to Ahkee
16:14:13.708 -> ..
16:14:14.217 -> WiFi connected.
16:14:14.217 -> IP address:
16:14:14.217 -> 192.168.1.5
```

```
Leaving...
Hard resetting via RTS pin...
```

**Figure 11** Obtaining IP Address

One can access web server on the specific local network. By typing the IP address of http://192.168.1.7 on a browser, the data can be visualized on the PC as well as on Serial monitor as illustrated in Figure 12 and 13. The data can be also visualized on the web page on the smart phone as shown in Figure 14.



ESP32 WEb Server data Read and Write

Dr Nu Nu Swe,Associate Professor

Department of Physics

Yangon University of Education

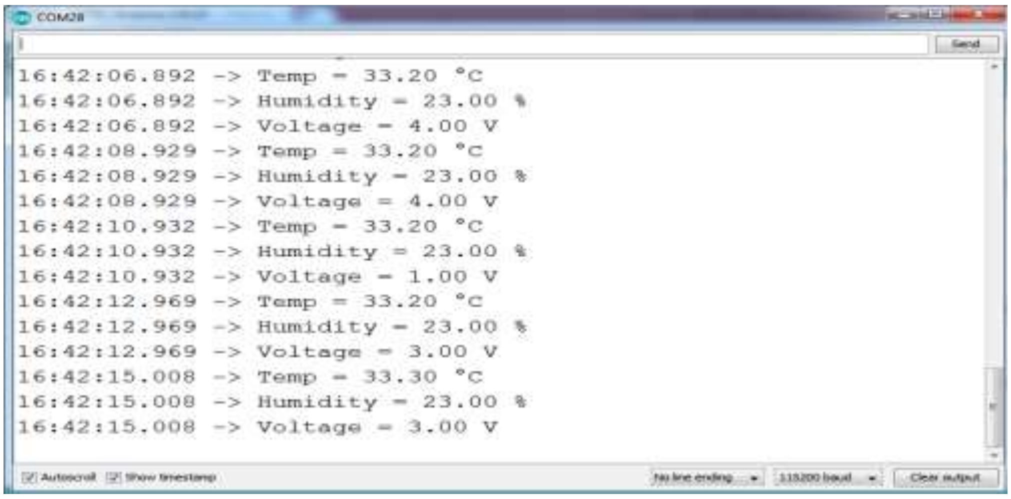
Current Temperature is: 29.80 oC
Current Humidity is : 39.00 %

Voltage measured: 2.00V

Click ON to turn the LED on.

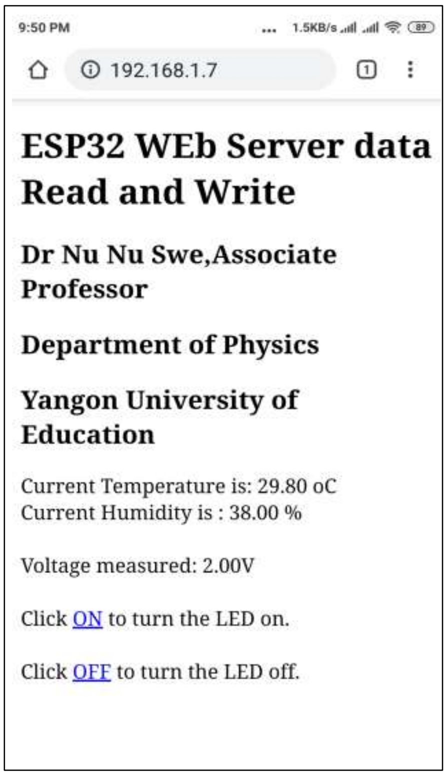
Click OFF to turn the LED off.

**Figure 12** Monitoring data and controlling LED from PC.



```
16:42:06.892 -> Temp = 33.20 °C
16:42:06.892 -> Humidity = 23.00 %
16:42:06.892 -> Voltage = 4.00 V
16:42:08.929 -> Temp = 33.20 °C
16:42:08.929 -> Humidity = 23.00 %
16:42:08.929 -> Voltage = 4.00 V
16:42:10.932 -> Temp = 33.20 °C
16:42:10.932 -> Humidity = 23.00 %
16:42:10.932 -> Voltage = 1.00 V
16:42:12.969 -> Temp = 33.20 °C
16:42:12.969 -> Humidity = 23.00 %
16:42:12.969 -> Voltage = 3.00 V
16:42:15.008 -> Temp = 33.30 °C
16:42:15.008 -> Humidity = 23.00 %
16:42:15.008 -> Voltage = 3.00 V
```

**Figure 13** Data on Serial Monitor



9:50 PM

192.168.1.7

**ESP32 WEb Server data Read and Write**

**Dr Nu Nu Swe,Associate Professor**

**Department of Physics**

**Yangon University of Education**

Current Temperature is: 29.80 oC
Current Humidity is : 38.00 %

Voltage measured: 2.00V

Click ON to turn the LED on.

Click OFF to turn the LED off.

**Figure 14** Monitoring Data at http://192.168.1.7,and controlling LED from Phone

## Discussion

There are a few problems when the new code (sketch) is uploaded to ESP32. The first problem is “COM port is not available”. It might be seen when the USB driver is missing or USB cable without data wires. If USB driver is missing CP2102 driver is installed. The another problem is “Failed to connect to ESP32: Timed out Connecting” It will occur when ESP32 is failing to connect with IDE, it means that ESP32 is not in flashing/uploading mode. It can be corrected as the hold-down the “**BOOT**” button in your ESP32 board while the uploading the code. After seeing the “**Connecting….**” message in Arduino IDE, the finger should release from the “**BOOT**” button.

## Conclusion

The local Wi-Fi router or phone Wi-Fi or fiber network are used as internet service provider (ISP). IP address of the website which is specified in this research is 192.168.1.7. It can be used in the local area network (LAN). If the public domain is available, the data on web page can be visualized from any other place with internet access. This system can be used to send secure message. This system can be used as home appliances monitoring and controlling system using the smoke sensor, flame sensor, gas sensor and current sensors. But it can be used in the Wi-Fi coverage area about thirty meters of radius. If the system can be monitoring sensor and controlling appliances at anywhere, it will integrate the software such as “ngrok”, IoT cloud server and hardware such as GSM module.

## Acknowledgements

I would like to thank Pro-rectors, Yangon University of Education, Dr Pyone Pyone Aung and Dr Kay Thwe Hlaing for their kind permission to take part in this research work.

I also thankful to Professor Dr Khin Swe Oo, Head of Department of Physics, Yangon University of Education, for giving advice and encouragement through the research period.

I deeply grateful to Professor Dr Ohn Than (Retired), Head of Department of Physics, Kyaukse University who supports the continuous guideline about the Arduino hardware and software.

Also, Thankful all my colleagues for their participation in this research work.

Finally, I pay my thank and respect to my parents.

## References

https://en.wikipedia.org/wiki/Web_server

https://en.wikipedia.org/wiki/ESP32

https://www.waveshare.com/wiki/DHT11_Temperature-Humidity_Sensor

https://www.mouser.com/datasheet/2/891/esp-wroom-32_datasheet_en-1223836.pdf

Rui Santos,”Learn ESP32 withArduino IDE” Random Nerd Tutorial Lab,2020

Purdun J, “Beginning C for Arduino” Apress, NY, 2012

Rui Santos,”ESP 32 troublesshooting guide” Random Nerd Tutorial Lab,2020