

**Zero to
Hero**

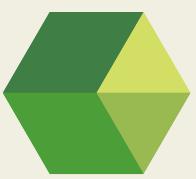
Basic Website Development For ESP32

Progress of
Zero to Hero 2023

Presented By
Chollasak Anuwareepong

Supported by
Embedded System Lab



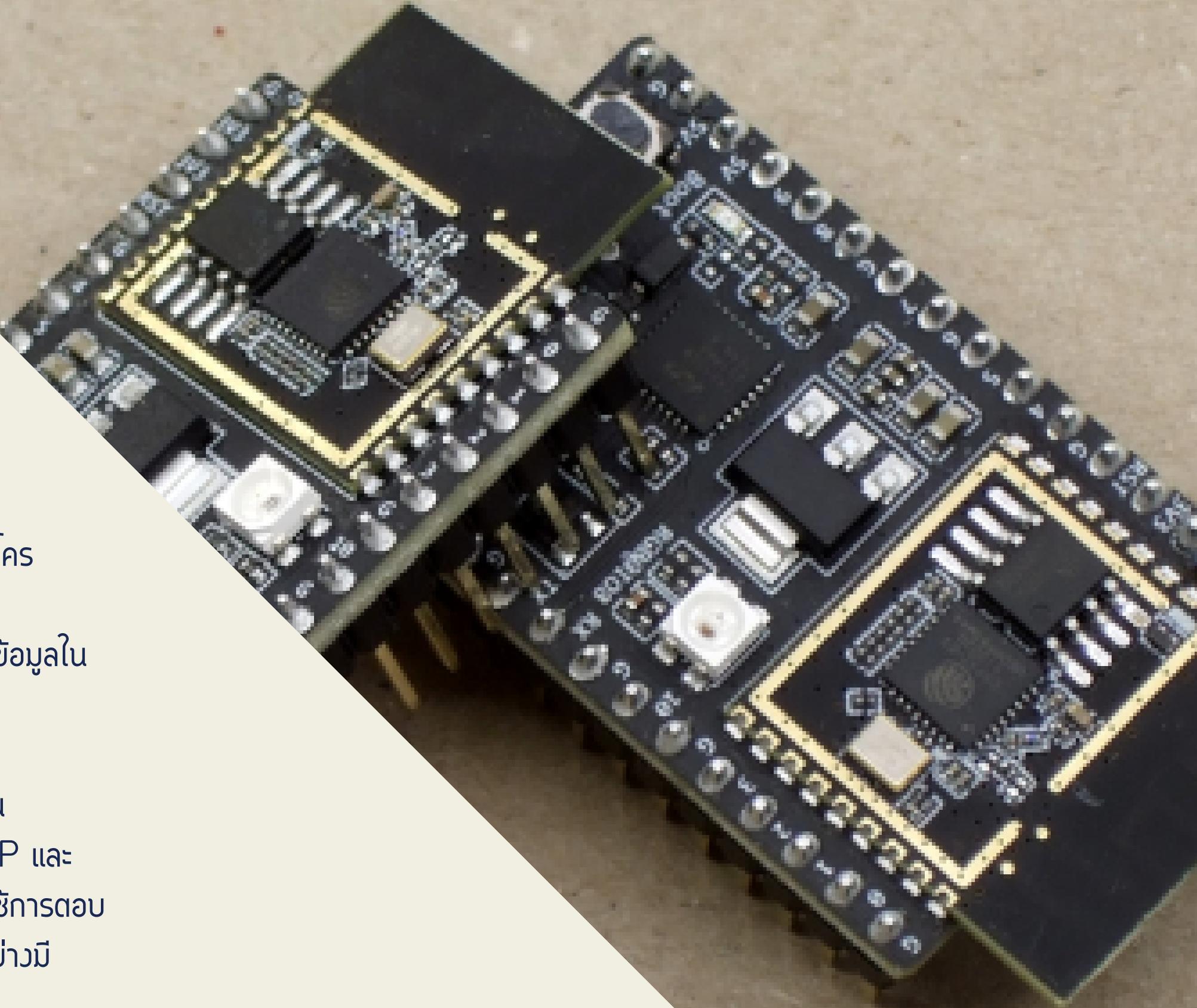


Zero to
Hero

Introduction

ในบทเรียนนี้เราจะเข้าสู่โลกของการสร้างเว็บเซิร์ฟเวอร์ด้วย ESP32 ซึ่งเป็นไมโครคอนโทรลเลอร์ที่ทรงพลังและมีความสามารถที่น่าทึ่ง การสร้างเว็บเซิร์ฟเวอร์บน ESP32 จะเปิดโอกาสให้คุณสร้างแอปพลิเคชันเว็บที่สามารถเชื่อมต่อและแสดงข้อมูลในเครือข่ายได้อย่างง่ายดาย

ในบทเรียนนี้คุณจะได้เรียนรู้เกี่ยวกับขั้นตอนพื้นฐานในการสร้างเว็บเซิร์ฟเวอร์บน ESP32 รวมถึงการกำหนดค่า Wi-Fi, การจัดการรับและส่งข้อมูลผ่าน HTTP และการสร้างหน้าเว็บสำหรับแสดงข้อมูล นอกจากนี้เราจะสอนคุณเกี่ยวกับการปรับใช้การต่อสานของเว็บ เพื่อให้ผู้ใช้สามารถเข้าถึงและติดต่อกับเว็บแอปพลิเคชันของคุณได้อย่างมีประสิทธิภาพ

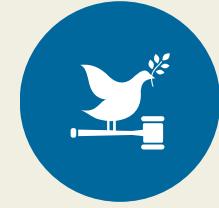


Step of Development



ตรวจสอบ **SENSORS**

ตรวจสอบเซนเซอร์ที่ใช้ รวมถึง กดลงรับค่าจากเซนเซอร์ให้แม่น ย้ำก่อนที่จะทำค่าดังกล่าวไป Monitor uu Website



พัฒนา **MOCKUP WEBSITE** จำลอง

เพื่อความสะดวกสบายในการตรวจสอบและพัฒนา จะได้ไม่เสียเวลาในการแก้ไขใน Arduino IDE จึงทำการพัฒนา เว็บไซต์บน VS code sageon จำลองค่าจากเซนเซอร์ว่าควรอยู่ ตำแหน่งไหน



นำส่วนทั้งสองมาร่วมกัน ส่วนสุดท้ายคือการนำค่าที่เราอ่านได้ จากเซนเซอร์ มาขึ้นบนเว็บไซต์

Example Website

The screenshot displays the Emoplanter website interface. On the left, a dark sidebar contains the text "CEKMITL" at the top, followed by the Emoplanter logo (two overlapping green and blue heart shapes) and the text "Emoplanter THE IOT PLANTER". Below the logo is a navigation menu with links: Home, Work Samples, Realtime Emotion (which is highlighted in blue), Developers, and Contact. At the bottom of the sidebar, there is a reference to "01076108 Circuits and Electronics in Practice of Computer Engineering". The main content area has a light gray background. At the top, the title "Realtime Emotion" is displayed above a horizontal orange bar. A sub-header below the bar reads "These values are being read from sensors those inside our planter." To the right of this header, two colored boxes are visible: a dark blue box labeled "Quality Values" and a teal box labeled "Realtime Data From Planter". The "Realtime Data From Planter" box contains four data points: "Soil Moisture" with a value of "23 %", "Temperature" with a value of "26 Celsius", "Light Intensity" with a value of "3241 Lumens", and "Planter's emotion" with a "Happy" status indicated by a yellow emoji face.

CEKMITL

Emoplanter
THE IOT PLANTER

Home

Work Samples

Realtime Emotion

Developers

Contact

01076108 Circuits and
Electronics in Practice of
Computer Engineering

Realtime Emotion

These values are being read from sensors those inside our planter.

Quality Values	Realtime Data From Planter
Soil Moisture	23 %
Temperature	26 Celsius
Light Intensity	3241 Lumens
Planter's emotion	Happy

<https://chollsak.github.io>

ESP32 Web Server – Arduino IDE

บทเรียนนี้เราจะนำคลาส **WIFISERVER** ใน **LIBRALY WIFI.H** มาสร้าง **WEBSERVER** ที่สามารถใช้วิค่าจากเซ็นเซอร์ไปยังบท **WEBSITE** ของเราได้

หลักการทำงานพื้นฐานของคลาส

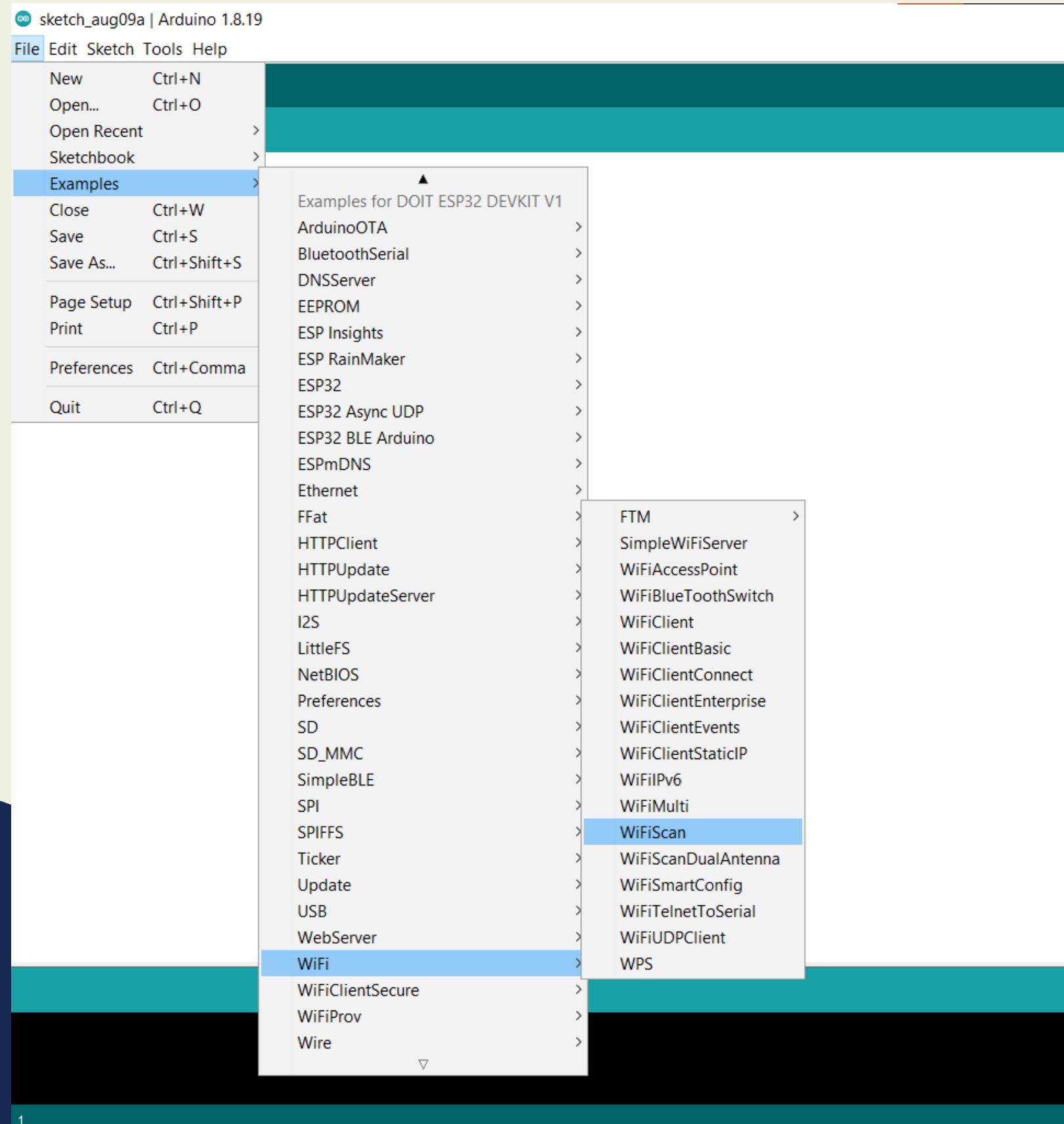
- คลาส WiFiServer ในไลบรารี WiFi.h ใน Arduino ใช้สำหรับสร้างเซิร์ฟเวอร์ (Server) ที่เชื่อมต่อผ่าน WiFi สำหรับการติดต่อแบบโต้ตอบ (TCP) หรือการแลกเปลี่ยนข้อมูลผ่านเครือข่าย WiFi ระหว่างอุปกรณ์ Arduino หรือ NodeMCU กับอุปกรณ์อื่น ๆ ที่เป็นไปได้ เช่น เครื่องคอมพิวเตอร์ โทรศัพท์มือถือ หรืออุปกรณ์อื่น ๆ ที่รองรับการเชื่อมต่อ WiFi แบบ TCP/IP.



ตัวอย่างโปรเจคที่ใช้งานอุปกรณ์ดังกล่าว
สามารถดูข้อมูลต่างๆได้ที่ README

[https://github.com/chollsak/Smart-IoT-Silkworm-House?
utm_source=canva&utm_medium=iframely](https://github.com/chollsak/Smart-IoT-Silkworm-House?utm_source=canva&utm_medium=iframely)

ตรวจสอบ Wifi.h ด้วย การสแกนคืนหา Wifi ในระยะที่ตรวจจับได้



เมื่อได้ตัวอย่างแล้ว ให้ลองอัพโหลด
ตรวจสอบว่า อุปกรณ์ของเรา
สามารถตรวจจับไวไฟได้หรือไม่

The screenshot shows the Arduino Serial Monitor window titled "COM3". The output text is as follows:

```
Scan start
Scan done
3 networks found
Nr | SSID           | RSSI | CH | Encryption
1  | Choll_2.4G     | -48  | 7  | WPA2
2  | marisi 2.4GHz  | -81  | 6  | WPA2
3  | Pause_2G        | -90  | 3  | WPA2
```

At the bottom of the window, there are checkboxes for "Autoscroll" and "Show timestamp", and buttons for "Newline", "115200 baud", and "Clear output".

จากการทดลองพบว่า ESP32 ของเราสามารถค้นหาและ
เชื่อมต่อไวไฟ ได้อย่างแน่นอน

Start!

นำโค้ดตัวอย่างด้านล่างมาใช้งานใน Arduino IDE

https://drive.google.com/drive/folders/1qpqA0JLC0fyX5Q19YAlTKUnB4ismAiqq?usp=drive_link



sketch_aug09a | Arduino 1.8.19

File Edit Sketch Tools Help

sketch_aug09a §

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

//Your sensors input
int sensorInput;

// Replace with your network credentials
const char* ssid = "Wifi SSID";
const char* password = "Password";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// 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);

  // 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) {

Done uploading.

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

ตัวแปรที่ไว้เก็บค่าข้อมูลจากเซนเซอร์ที่น้องๆใช้ในบทเรียน Basic ESP32

ข้อมูล WiFi ที่ใช้งาน

86 DOIT ESP32 DEVKIT V1, 80MHz, 921600, None, Disabled on COM3

```
void loop() {  
  
    sensorInput = 27; //please change your sensor code and argorithms here  
    WiFiClient client = server.available(); // Listen for incoming clients  
  
    if (client) {  
        currentTime = millis();  
        previousTime = currentTime;  
        Serial.println("New Client.");  
        String currentLine = "";  
        while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the client's connected  
            currentTime = millis();  
            if (client.available()) {  
                char c = client.read();  
                Serial.write(c);  
                header += c;  
                if (c == '\n') {  
                    // 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)  
                    }  
                }  
            }  
        }  
    }  
}
```

นำ Code เชนเซอร์ของน้องๆ มาแทนที่ 27 ที่พื้นที่ MOCKUP ไว้



sketch_aug09a §

```
// 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();

// Display the HTML web page
```

}

นำ HTML มาใส่ในส่วนนี้ และต้องมี Client.println ทุกบรรทัด

```
// Html End Here

// 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");
```

Done uploading.

Leaving...

Hard resetting via RTS pin...



Tips! เพื่อความสะดวกสบาย โดยที่ไม่ต้องพิมพ์ client.println ทุกบรรทัด

A

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h3>Basic Website Development For ESP32 [ZERO TO HERO 2023]</h3>
  <hr>
  <h1>ESP32 Web Server</h1>
  <p>Realtime Value:</p>
</body>
</html> add client.println for this code in every single line
```



Html example code

TITLE	LAST MODIFIED
arduino.txt	3:32 am
html.txt	3:49 am

เมื่ออัพโหลดโปรแกรม เปิด Serial Monitor และเมื่อเชื่อมต่อไฟสำเร็จ จะพบข้อมูลดังภาพ

```
COM3
|
Connecting to Choll_2.4G
.....
WiFi connected.
IP address:
192.168.1.117 <- นำเลข IP Address ไปค้นหาที่ Search Engine
New Client.
GET / HTTP/1.1
Host: 192.168.1.117
```

จะได้ผลลัพธ์ดังนี้

Basic Website Development For ESP32 [ZERO TO HERO 2023]

ESP32 Web Server

Realtime Value

Temperature:

Basic Website Development For ESP32 [ZERO TO HERO 2023]

ESP32 Web Server

Realtime Value

Temperature: <- เราจะนำค่าที่อ่านจากเซนเซอร์ มาแสดงตรงนี้อย่างไร?



```
// Display the HTML web page
client.println("<!DOCTYPE html>");
client.println("<html lang=\"en\">");
client.println("<head>");
client.println("    <meta charset=\"UTF-8\">");
client.println("    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">");
client.println("    <title>Document</title>");
client.println("</head>");
client.println("<body>");
client.println("    <h3>Basic Website Development For ESP32 [ZERO TO HERO 2023]</h3>");
client.println("    <hr>");
client.println("    <h1>ESP32 Web Server</h1>");
client.println("    <p>Realtime Value </p>");
client.println("    <p>Temperature: " + String(sensorInput) + "</p>" );
client.println("</body>");
client.println("</html>");
// Html End Here
```





การเขียนของ Code

The first thing you need to do is to include the WiFi library.

```
#include <WiFi.h>
```

As mentioned previously, you need to insert your ssid and password in the following lines inside the double quotes.

```
const char* ssid = "";  
const char* password = "";
```

Then, you set your web server to port 80.

```
WiFiServer server(80);
```

The following line creates a variable to store the header of the HTTP request:

```
String header;
```

setup()

Now, let's go into the `setup()`. First, we start a serial communication at a baud rate of 115200 for debugging purposes.

```
Serial.begin(115200);
```

The following lines begin the Wi-Fi connection with `WiFi.begin(ssid, password)`, wait for a successful connection and print the ESP IP address in the Serial Monitor.

```
// 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();
```

loop()

In the `loop()` we program what happens when a new client establishes a connection with the web server.

The ESP32 is always listening for incoming clients with the following line:

```
WiFiClient client = server.available(); // Listen for incoming clients
```

When a request is received from a client, we'll save the incoming data. The while loop that follows will be running as long as the client stays connected. We don't recommend changing the following part of the code unless you know exactly what you are doing.

```
if (client) { // If a new client connects,
    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()) { // loop while the client's connected
        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();
                }
            }
        }
    }
}
```

Displaying the HTML web page

The next thing you need to do, is creating the web page. The ESP32 will be sending a response to your browser with some HTML code to build the web page.

The web page is sent to the client using this expressing `client.println()`. You should enter what you want to send to the client as an argument.

The first thing we should send is always the following line, that indicates that we are sending HTML.



HTML and CSS code example: <https://raw.githubusercontent.com/chollsak/KMITL-Circuit-and-Electronics-Projects-1D/main/src/main.cpp> (Real Project)

The Website: <https://chollsak.github.io> (Mocked Up)

More HTML Templates: https://www.w3schools.com/w3css/w3css_templates.asp

Activity

ให้น้องๆ ตกลงแต่งเว็บไซต์ของตนเอง
รวมถึงแสดงค่าจากเซนเซอร์ให้ชัดเจน

T₁ H₄ A₁ N₁ K₅ S₁