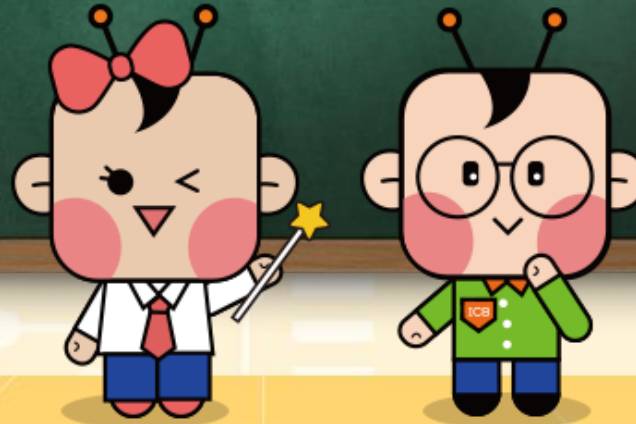


# ESP32로 만드는 IoT 월드 with 아두이노 IDE 2

**프로젝트명** Ch5.Web Server

**난이도** ★★★★★



## 학습요약

학습목표	센서값을 표시하는 웹서버를 만들어 보시다.
핵심 키워드	ESP32, IoT, 사물인터넷, 웹 서버, DH11, 온습도 센서
준비물	ESP32, ESP32 확장 실드, DH11 온습도 센서
학습 시간	2시간
학습 난이도	★★★★☆☆

ESP32 Wi-Fi 네트워크 또는 인터넷에 연결한 다음,  
ESP32에서 웹 서버를 실행하여 Client에게 웹 페이지를 제공할 수 있습니다.

## ESP32의 Wi-Fi API의 제공 기능

- ① Station 모드(STA 모드 또는 Wi-Fi Client 모드):  
기존의 Wi-Fi 네트워크에 Client로서 연결
- ② AP 모드(소프트-AP 모드 또는 액세스 포인트 모드):  
다른 장치들이 ESP32의 Wi-Fi 네트워크에 연결되어 통신
- ③ 보안 모드:  
WPA2 및 WPA3와 같은 다양한 보안 모드를 지원
- ④ 액세스 포인트 스캔:  
주변 네트워크에 대한 정보(SSID, 신호 강도, 보안 설정 등)를 수집

“Hello from ESP32!”라는 메시지를 반환하는 간단한 웹 서버 예제를 실행해 보겠습니다.

```

1  #include <WiFi.h> // Wi-Fi 라이브러리를 포함합니다.
2  #include <WebServer.h> // WebServer 라이브러리를 포함합니다.
3  const char* ssid = "YourWiFiSSID"; // 연결할 Wi-Fi의 SSID를 입력합니다.
4  const char* password = "YourWiFiPassword"; // Wi-Fi의 비밀번호를 입력합니다.
5  // 포트 번호 80을 사용하여 WebServer 객체를 생성합니다.
6  WebServer server(80);
7  void handleRoot() {
8  // "/" 경로에 대한 요청을 처리하는 핸들러 함수입니다.
9  server.send(200, "text/plain", "HellofromESP32!");
10 }
11 void setup() {
12 Serial.begin(115200); // 시리얼 통신을 초기화합니다.
13 WiFi.begin(ssid, password); // Wi-Fi에 연결합니다.
14 Serial.println("ConnectedtoWiFi");
15 while (WiFi.status() != WL_CONNECTED) {
16 delay(1000);
17 Serial.print(".");
18 }

```

※ 실습하는 WiFi환경에 맞게 ssid와 password를 설정합니다.  
ESP32는 2G대역만 지원하므로 5G대역ssid는 입력하지 않습니다.

※ ESP32웹서버에 접속했을때, 나타나는 문구를 출력해주는 함수 입니다.

※ WiFi연결을 시도합니다. ssid와 password에 오류가 있을 경우 이 코드를 벗어날 수 없습니다.

```
20 Serial.println("");
21 Serial.println("WiFiconnected.");
22 Serial.println("IPAddress:");
23 Serial.println(WiFi.localIP()); // Wi-Fi의 로컬 IP 주소를 출력합니다.
24
25 // "/" 경로에 대한 요청을 handleRoot() 함수로 처리합니다.
26 server.on("/", handleRoot);
27 server.begin(); // 서버를 시작합니다.
28 }
29 void loop() {
30   server.handleClient(); // Client의 요청을 처리합니다.
31 }
```

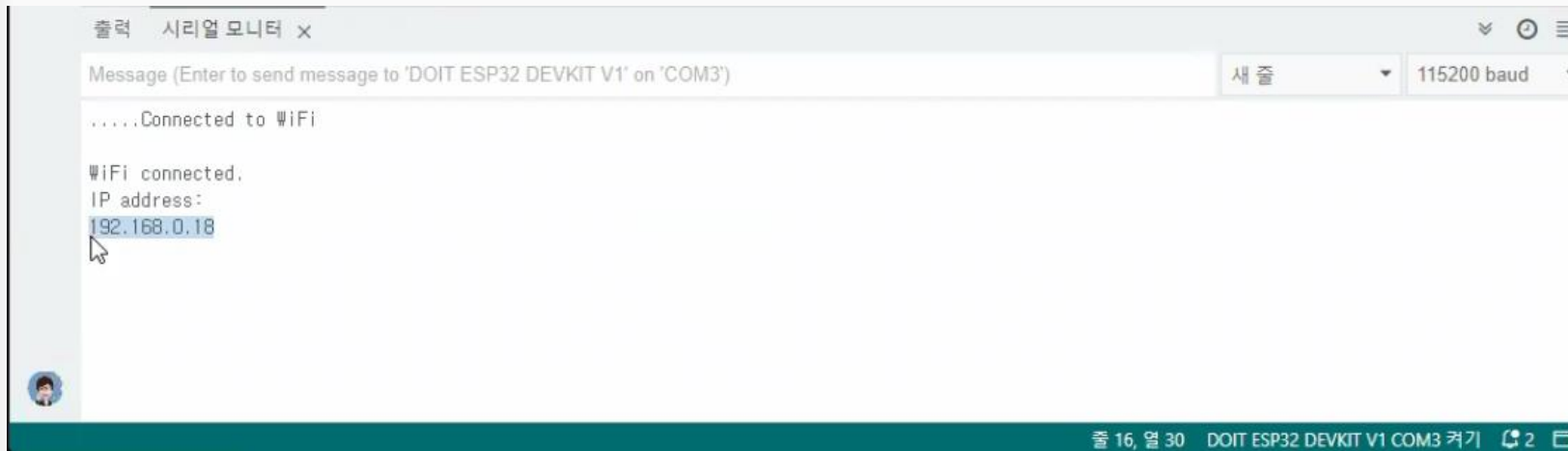
※ AP에 연결되면 ip주소를 할당 받습니다.  
이 주소를 브라우저에 입력하면 ESP32에서 Text  
를전송해줍니다.

※ 웹서버를 설정하고 시작합니다.

※ Client가 접속하기를 기다렸다가 요청을 처리합  
니다.

## 실행 결과

시리얼 모니터창을 열고 ESP32에 할당된 IP주소를 확인합니다.



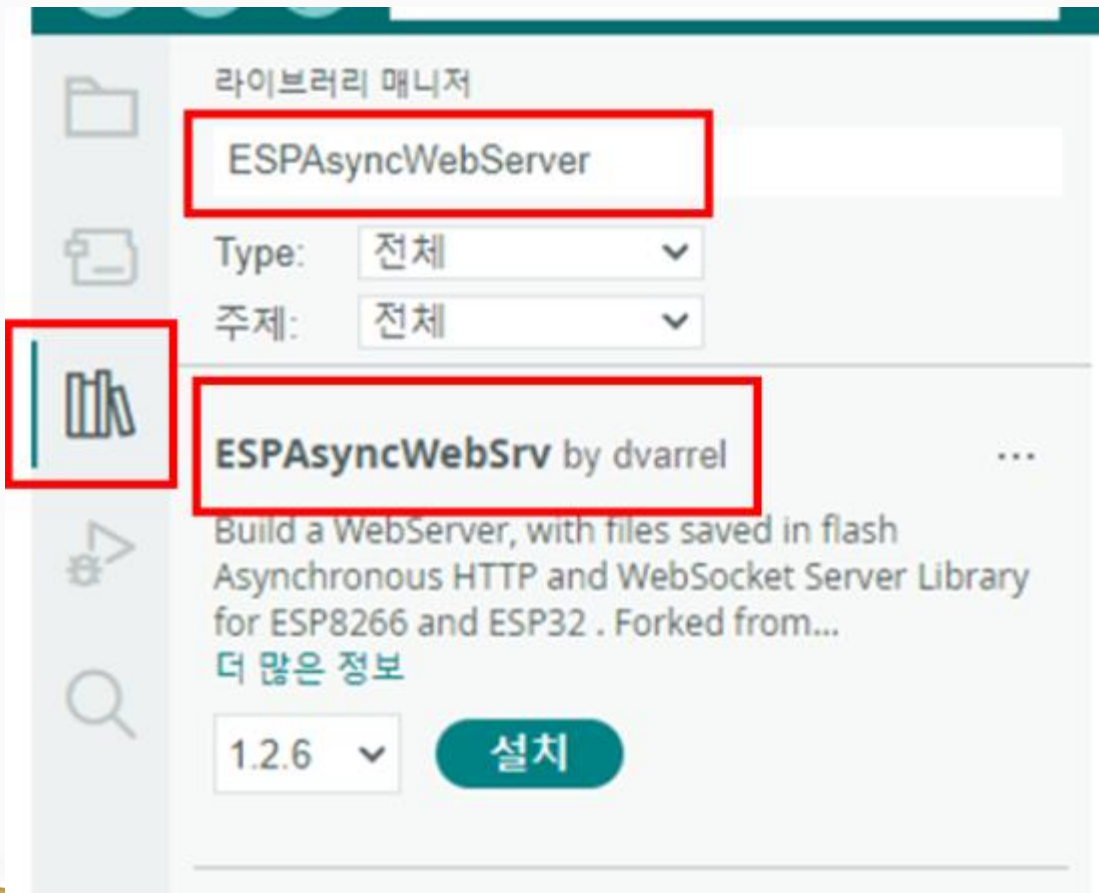
ESP32의 주소를 인터넷 브라우저에 입력합니다.



ESP32의 내장 LED를 제어하는 웹 서버 예제를 실행해 보겠습니다.

라이브러리 설치

"ESPAsyncWebSrv"를 검색하고 설치합니다.





```
1 // Import required libraries
2 #include <WiFi.h>
3 #include <AsyncTCP.h>
4 #include <ESPAsyncWebSrv.h>
5 // 자신의 네트워크 환경에 맞게 수정 필요
6 const char* ssid = "REPLACE_WITH_YOUR_SSID";
7 const char* password = "REPLACE_WITH_YOUR_PASSWORD";

14 const char index_html[] PROGMEM = R"rawliteral(
15 <!DOCTYPE html>
16 <html>
17 <head>
18
19 ...
20
21 <body>
22 <h1>ESP32 웹 서버</h1>
23 <p>GPIO 2 - 상태 %STATE%</p>
24 %BUTTON%
25 </body>
26 </html>
27 )rawliteral";
```

※ 실습환경에 맞게 SSID와 PASSWORD를 수정해 줍니다.

※ 웹서버의 내용을 표시할 html을 정의해 줍니다.

※ html body부분의 %STATE%, %BUTTON%은 LED상태에 따라 실시간으로 변경되는 부분입니다.



```
37 // Replaces placeholder with button section in your web page
38 String processor(const String& var) {
39     //Serial.println(var);
40     if (var == "STATE") {
41         return output2State;
42     }
43     if (var == "BUTTON") {
44         String buttons = "";
45         if (output2State == "off") {
46             buttons += "<p><a href=\"/2/on\">\n
47 <button class=\"button\">ON</button></a></p>";
48         } else {
49             buttons += "<p><a href=\"/2/off\">\n
50 <button class=\"button button2\">OFF</button></a></p>";
51         }
52         return buttons;
53     }
54     return String();
55 }
```

※ html body부분  
의 %STATE%, %BUTTON%을 어떤 문자열로  
채울지 정해 주는 함수 입니다.

%STATE%는 LED상태 값 "on"이나 "off"를 반환  
해 줍니다.

%BUTTON%은 LED상태에 따라 off시에는  
"ON"버튼을 on시에는 "OFF"버튼을 표시하는  
html구문을 반환합니다.

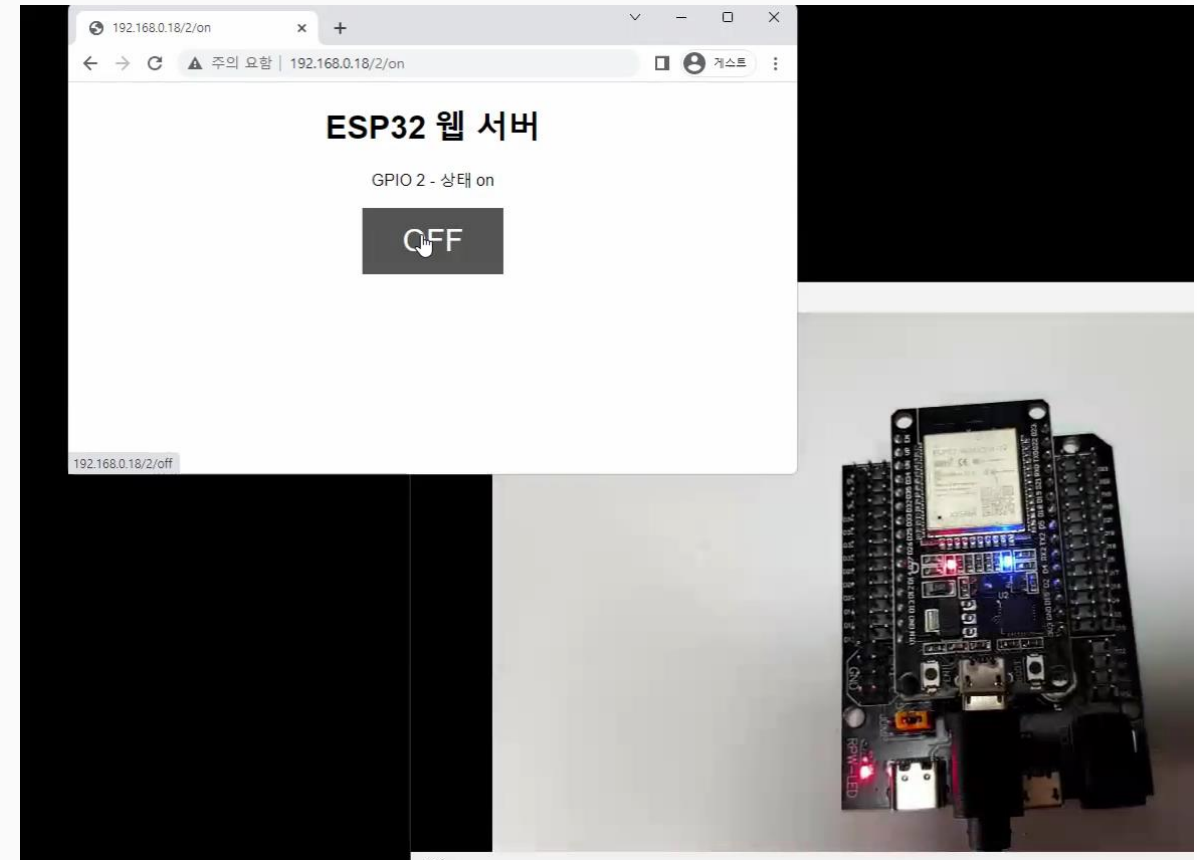
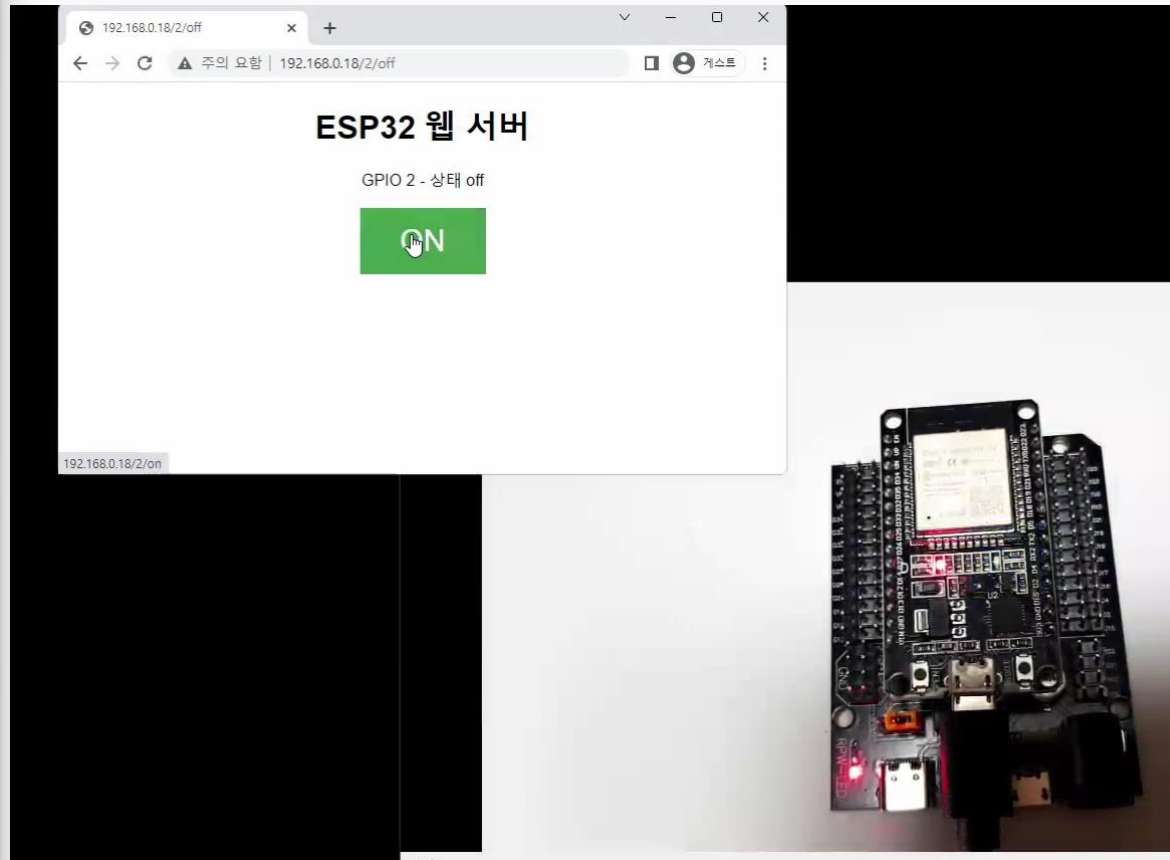
```
77 // Route for root / web page
78 server.on("/", HTTP_GET, [](AsyncWebServerRequest* request) {
79     request->send_P(200, "text/html", index_html, processor);
80 });
81 // Send a GET request to <ESP_IP>/2/on
82 server.on("/2/on", HTTP_GET, [](AsyncWebServerRequest* request) {
83     Serial.println("GPIO 2 ON");
84     output2State = "on";
85     digitalWrite(output2, HIGH);
86     request->send_P(200, "text/html", index_html, processor);
87 });
88 // Send a GET request to <ESP_IP>/2/off
89 server.on("/2/off", HTTP_GET, [](AsyncWebServerRequest* request) {
90     Serial.println("GPIO 2 OFF");
91     output2State = "off";
92     digitalWrite(output2, LOW);
93     request->send_P(200, "text/html", index_html, processor);
94 });
95 // Start server
96 server.begin();
97 }
```

※ "/" ESP32 웹서버 처음 접속시에 보여줄 html 문구와 %STATE%, %BUTTON%을 처리할 함수를 지정해 줍니다.

※ ON버튼을 클릭했을때 동작을 지정해 줍니다.  
LED상태를 바꾸어 주고 html문구와 %STATE%, %BUTTON%을 처리할 함수를 지정해 줍니다.

※ OFF버튼을 클릭했을때 동작을 지정해 줍니다.  
LED상태를 바꾸어 주고 html문구와 %STATE%, %BUTTON%을 처리할 함수를 지정해 줍니다.

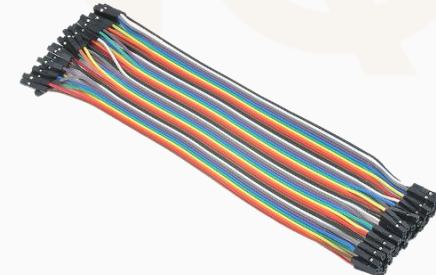
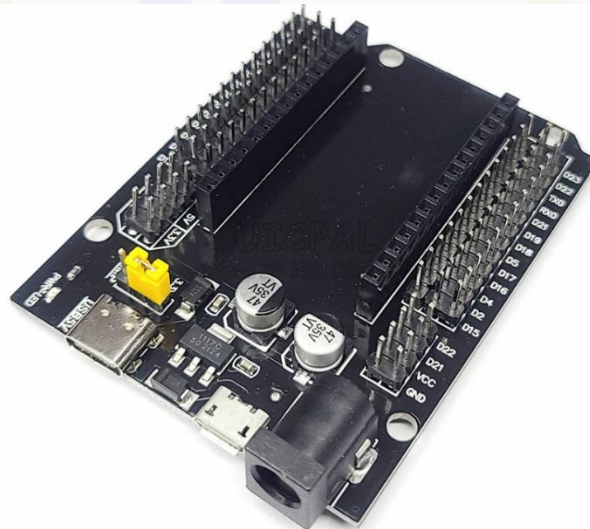
## 실행 결과



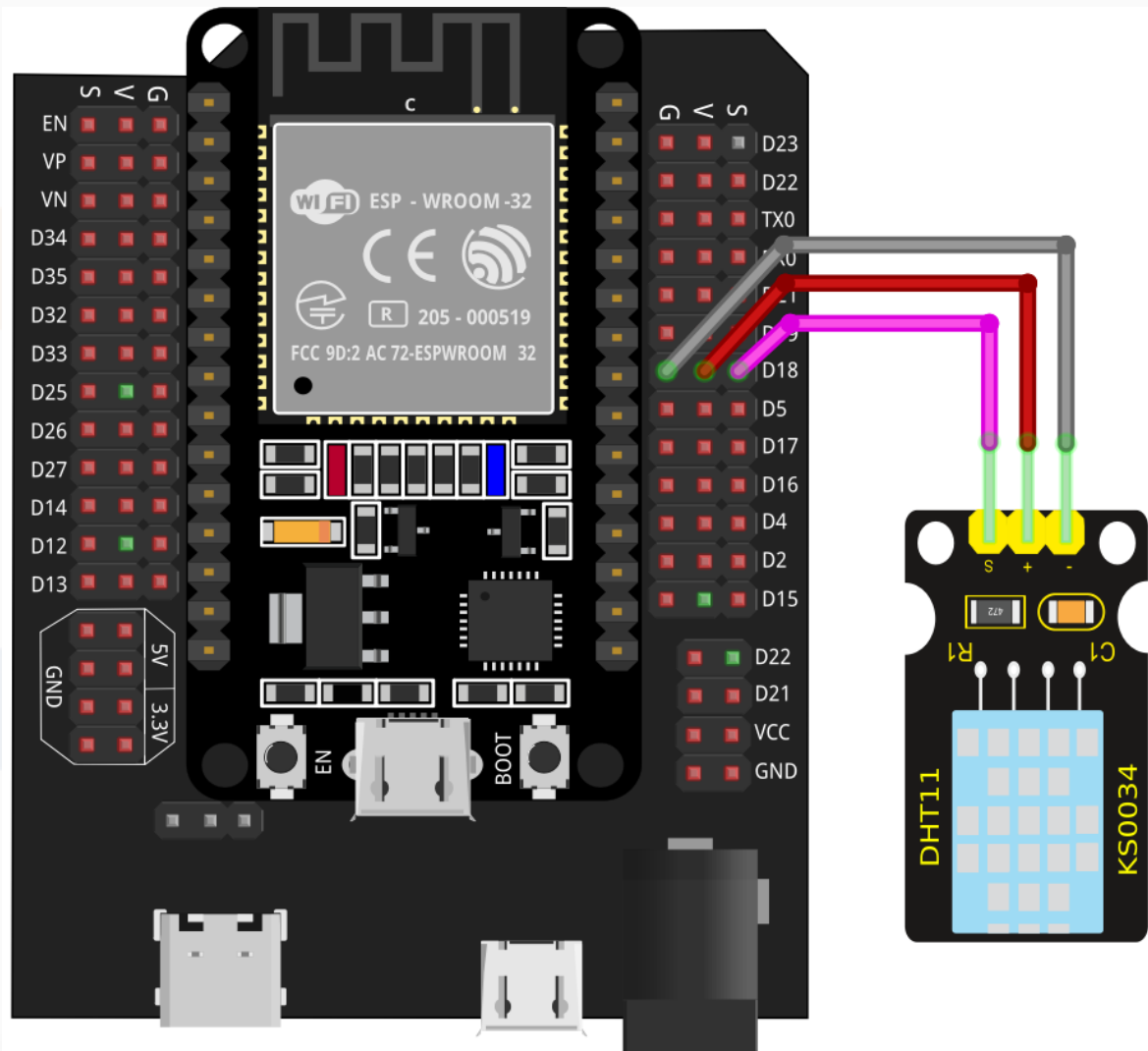
DH11 센서는 온도와 상대 습도를 측정하기 위해 사용되는 저렴한 디지털 센서입니다.

DH11모듈 사용방법을 알아보시다.

준비물 : ESP32, ESP32 확장 실드, DH11 센서 모듈



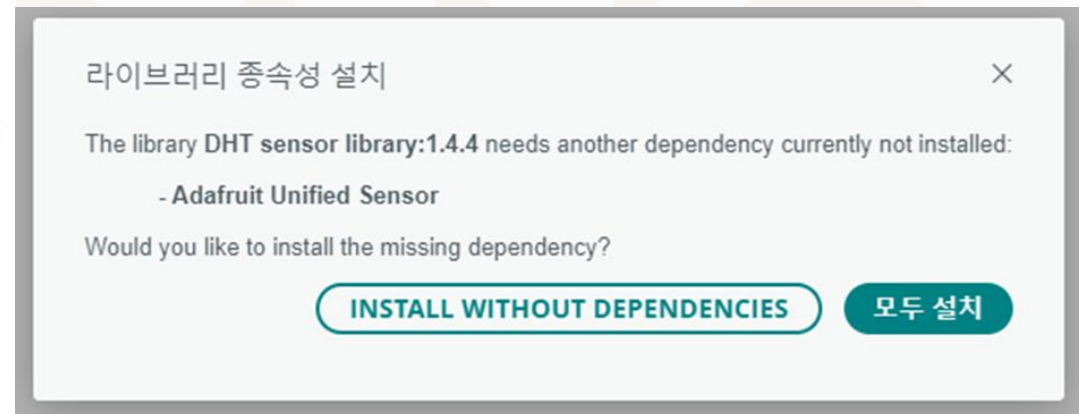
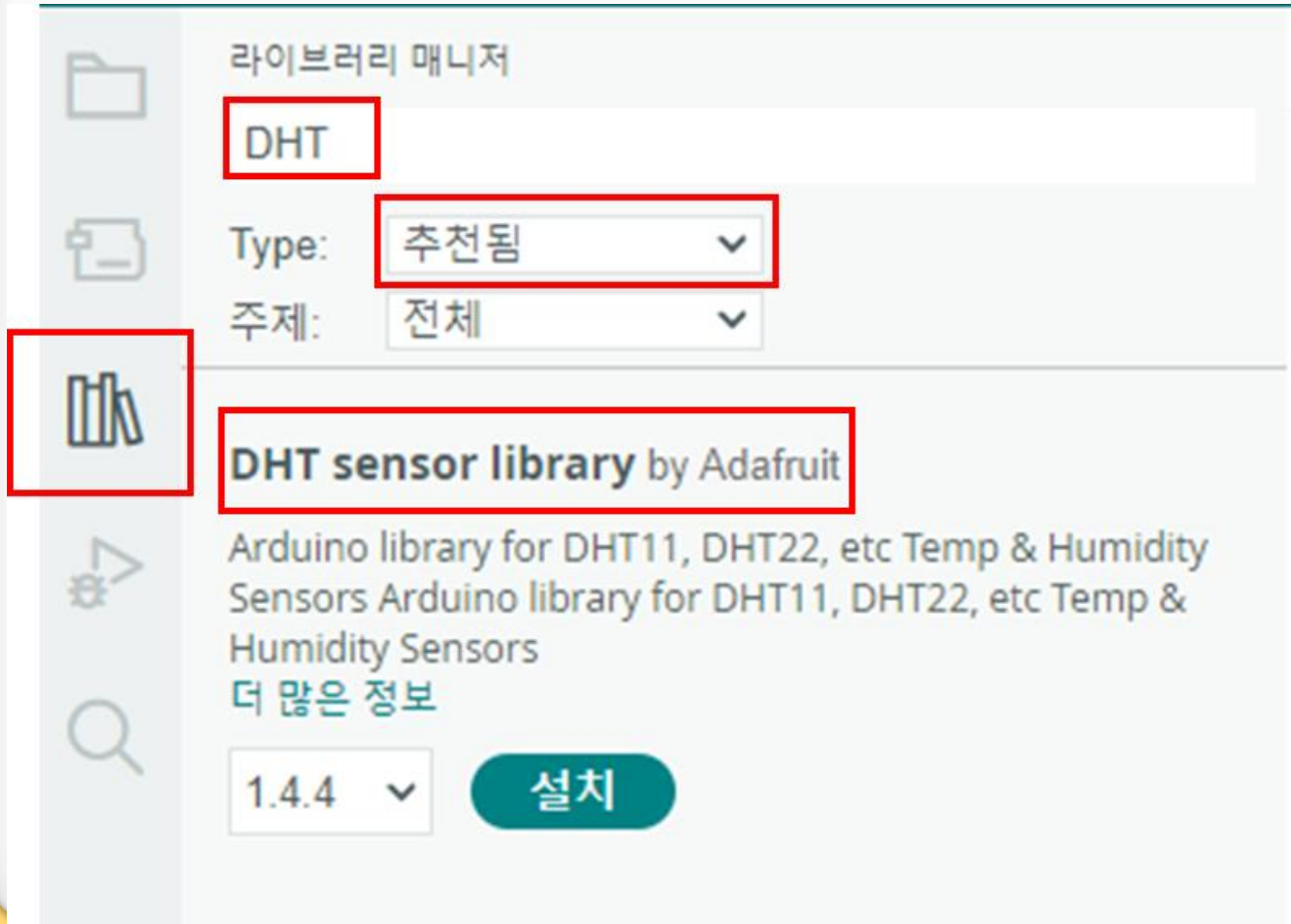
## 회로 연결하기



ESP32 쉴드	DH11 센서 모듈
S : D18	S
V	+
G	-

## 라이브러리 설치

"DHT sensor library by Adafruit"를 설치해줍니다. 연관된 "Adafruit Unified Sensor" 라이브러리도 모두 설치해주어야 합니다.



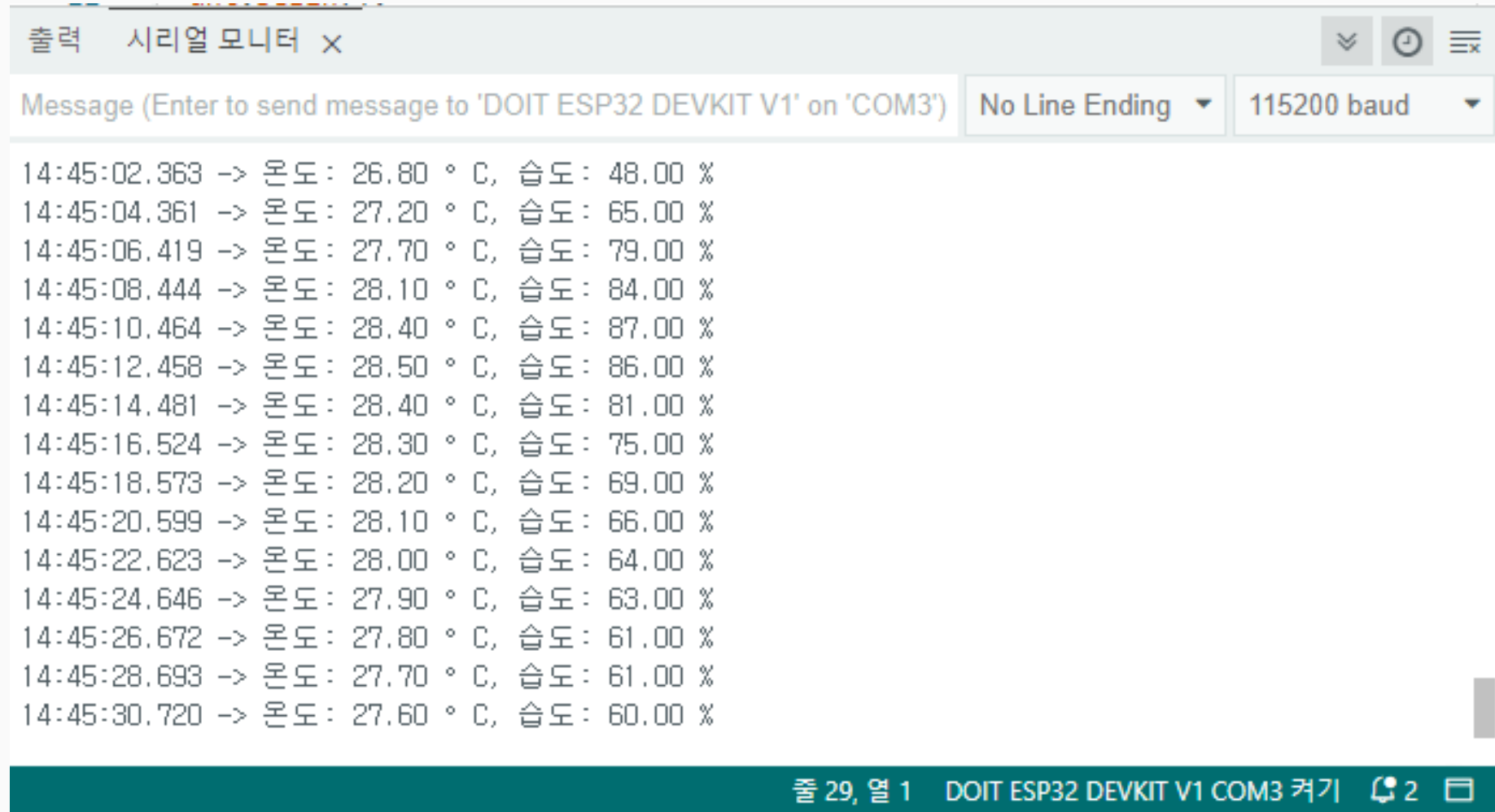


```
1  #include <DHT.h>
2
3  #define DHTPIN 18 // DHT11 센서에 연결된 핀 번호
4
5  // 사용 중인 센서 유형으로 변경하세요. DHT11, DHT22, DHT21
6  #define DHTTYPE DHT11 // DHT 센서 유형
7
8  DHT dht(DHTPIN, DHTTYPE);
9
10 void setup() {
11     Serial.begin(115200);
12     dht.begin();
13 }
14
15 void loop() {
16     delay(2000); // 2초마다 센서 값을 읽습니다.
17
18     // 온도 및 습도 값 읽기
19     float temperature = dht.readTemperature();
20     float humidity = dht.readHumidity();
21
22     // 센서 값 출력
23     Serial.print("온도: ");
24     Serial.print(temperature);
25     Serial.print(" °C, 습도: ");
26     Serial.print(humidity);
27     Serial.println(" %");
28 }
```

※ 온습도 센서값은 dht.readTemperature(), dht.readHumidity() 함수로 간단히 읽을 수 있습니다.



## 실행 결과



The screenshot shows a serial monitor window titled '출력 시리얼 모니터 x'. The message input field contains 'Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM3')'. The settings are 'No Line Ending' and '115200 baud'. The output displays a series of temperature and humidity readings over time.

Time	Temperature (°C)	Humidity (%)
14:45:02.363	26.80	48.00
14:45:04.361	27.20	65.00
14:45:06.419	27.70	79.00
14:45:08.444	28.10	84.00
14:45:10.464	28.40	87.00
14:45:12.458	28.50	86.00
14:45:14.481	28.40	81.00
14:45:16.524	28.30	75.00
14:45:18.573	28.20	69.00
14:45:20.599	28.10	66.00
14:45:22.623	28.00	64.00
14:45:24.646	27.90	63.00
14:45:26.672	27.80	61.00
14:45:28.693	27.70	61.00
14:45:30.720	27.60	60.00

At the bottom of the window, the status bar shows '줄 29, 열 1 DOIT ESP32 DEVKIT V1 COM3 켜기' and a notification icon with the number '2'.

DHT11 센서의 온도와 습도를 표시하는 웹 서버 예제를 실행해 보겠습니다.

```

82 <body>
83 <h2>ESP32 DHT Server</h2>
84 <p>
85 <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
86 <span class="dht-labels">온도</span>
87 <span id="temperature">%TEMPERATURE%</span>
88 <sup class="units">&deg;C</sup>
89 </p>
90 <p>
91 <i class="fas fa-tint" style="color:#00add6;"></i>
92 <span class="dht-labels">습도</span>
93 <span id="humidity">%HUMIDITY%</span>
94 <sup class="units">&percnt;</sup>
95 </p>
96 </body>
    
```

※ html 의 body에 ESP32로 부터 전송된 온도값을 표시하기 위한 %TEMPERATURE% 자리표시자를 지정해 줍니다.

※ html 의 body에 ESP32로 부터 전송된 습도값을 표시하기 위한 %HUMIDITY% 자리표시자를 지정해 줍니다.

```
97 <script>
98 setInterval(function() {
99 // XMLHttpRequest 객체를 사용하여 "/temperature" 엔드포인트로 GET 요청을 전송
100 var xhttp = new XMLHttpRequest();
101 xhttp.onreadystatechange = function() {
102 if (this.readyState == 4 && this.status == 200) {
103 // 요청이 완료되고 성공적인 응답을 받았을 때, 온도 값을 업데이트합니다.
104 document.getElementById("temperature").innerHTML = this.responseText;
105 }
106 };
107 xhttp.open("GET", "/temperature", true);
108 xhttp.send();
109 }, 2000); // 2초 간격으로 업데이트합니다.
110
111 setInterval(function() {
112 // XMLHttpRequest 객체를 사용하여 "/humidity" 엔드포인트로 GET 요청을 전송
113 var xhttp = new XMLHttpRequest();
114 xhttp.onreadystatechange = function() {
115 | if (this.readyState == 4 && this.status == 200) {
116 // 요청이 완료되고 성공적인 응답을 받았을 때, 습도 값을 업데이트합니다.
117 document.getElementById("humidity").innerHTML = this.responseText;
118 }
119 };
120 xhttp.open("GET", "/humidity", true);
121 xhttp.send();
122 }, 2000); //2 간격으로 업데이트합니다.
```

※ 2초마다 GET방식으로 온도값을 요청하는 script함수를 정의 해줍니다. ESP32로 부터 받은 Text를 html의 자 리표시자에 업데이트해줍니다.

※ 2초마다 GET방식으로 습값을 요청 하는 script함수를 정의 해줍니다.

```
154 // 루트(/) 웹 페이지 라우팅
155 server.on("/", HTTP_GET, [](AsyncWebServerRequest* request) {
156 | request->send_P(200, "text/html", index_html, processor);
157 | });
158
159 // "/temperature" 엔드포인트에 대한 GET 요청 처리
160 server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest* request) {
161 | request->send_P(200, "text/plain", readDHTTemperature().c_str());
162 | });
163
164 // "/humidity" 엔드포인트에 대한 GET 요청 처리
165 server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest* request) {
166 | request->send_P(200, "text/plain", readDHTHumidity().c_str());
167 | });
168
169 // 서버 시작
170 server.begin();
```

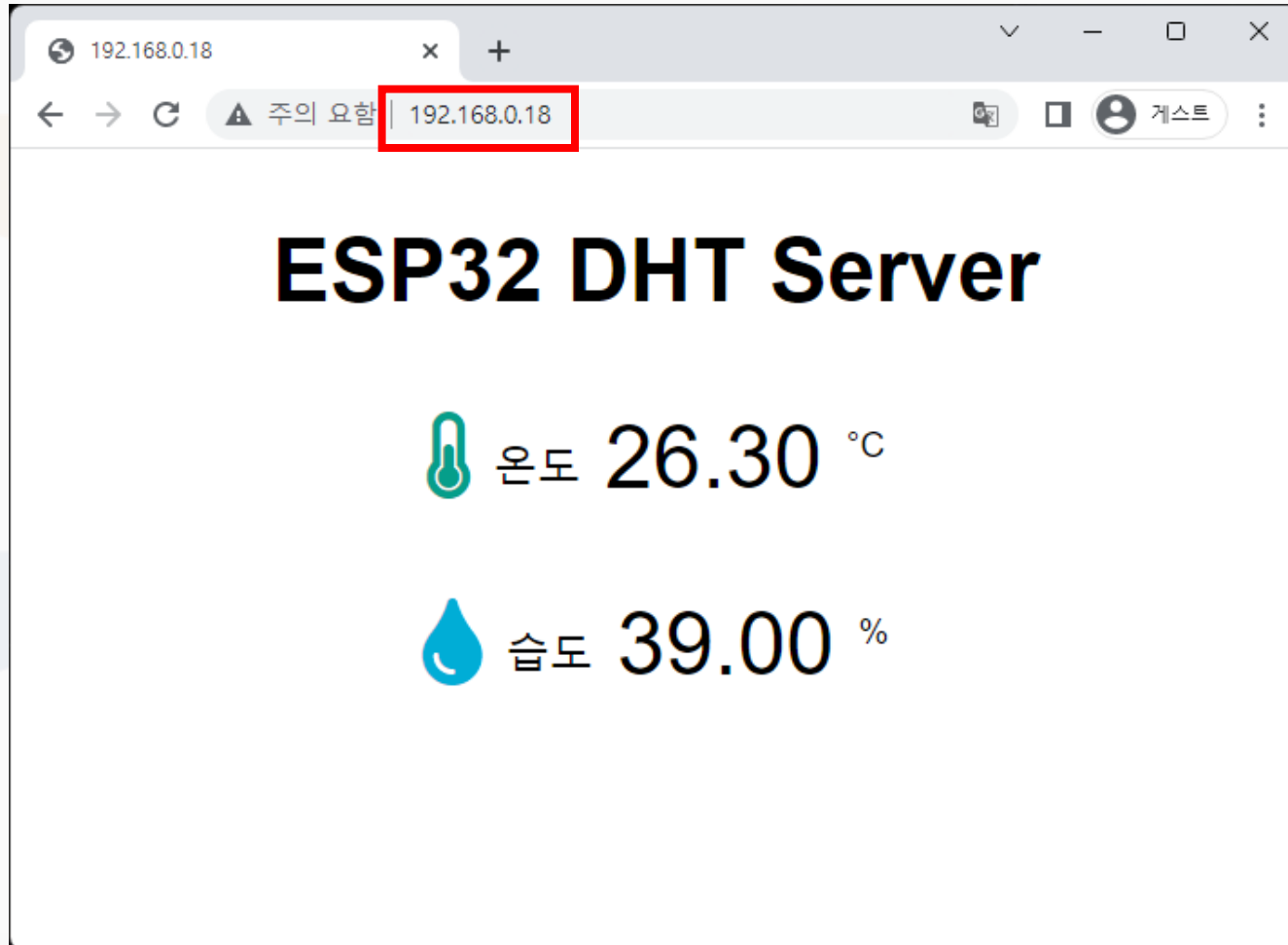
※ 브라우저에 ESP32의 주소값을 입력  
하면 html page를 회신 해줍니다./.

※ 온도값을 요청 받으면 DH11의 온도  
값을 문자열로 바꾸어 전송해 줍니다.

※ 습도값을 요청 받으면 DH11의 습도  
값을 문자열로 바꾸어 전송해 줍니다.

실행 결과 :

시리얼 모니터에서 ESP32에 할당된 IP주소를 확인한 뒤 브라우저의 주소창에 입력해 줍니다.



DHT11 센서의 온도와 습도를 표시하는 웹 서버를 인터넷에 노출하고, 외부에서 접속하는 방법을 알아봅시다.

로컬 서버를 외부 인터넷에 노출하는 다양한 방법이 있습니다.

① 포트 포워딩

공유기에서 특정 포트와 로컬 서버의 포트를 매핑,  
외부에서 공인 IP 주소와 해당 포트 번호를 통해 접속

② 터널링 기술

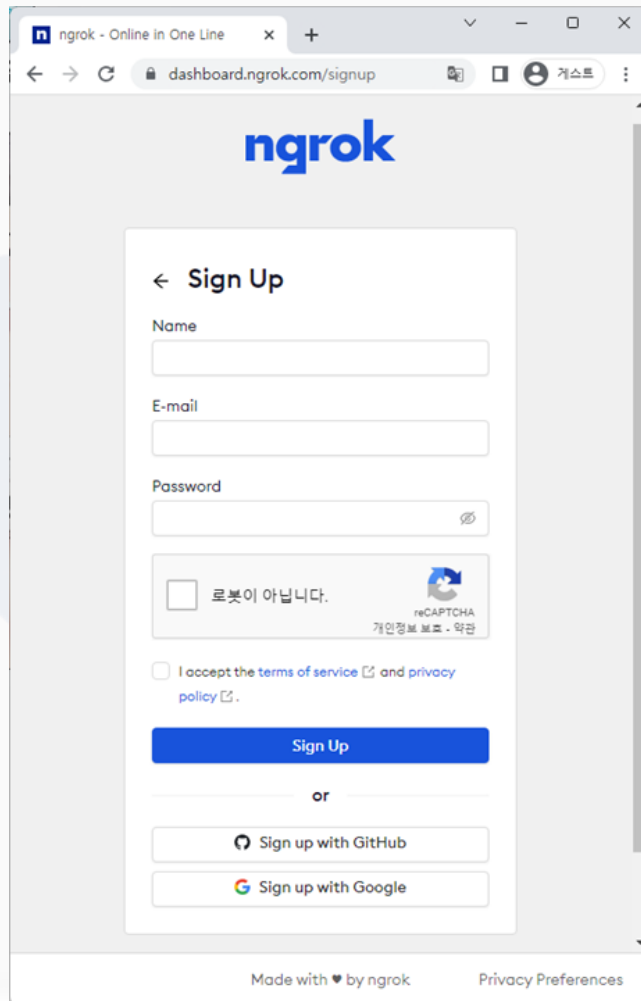
인터넷에 연결된 장치나 서버에 공인 IP와 포트를 할당하여 외부에서 접속

③ 직접 서버를 호스팅

고정 IP를 구입하여 직접 호스팅

## Ngrok가입

-<https://dashboard.ngrok.com/signup> 접속하여 가입하고 이메일 계정 인증을 합니다.



The screenshot shows the Ngrok sign-up page in a web browser. The page has a white background with the Ngrok logo at the top. Below the logo is a 'Sign Up' form with fields for Name, E-mail, and Password. There is a reCAPTCHA checkbox and a link to the terms of service and privacy policy. At the bottom of the form are buttons for 'Sign Up', 'Sign up with GitHub', and 'Sign up with Google'. The footer of the page says 'Made with ♥ by ngrok' and 'Privacy Preferences'.

## ★ Verify email address for ngrok.com

보낸사람 no-reply@ngrok.com VIP

받는사람

2023년 5월 13일 (토) 오후 10:47

영어 → 한국어 번역하기

Please verify your email address.

Use the following link to confirm your email address:

<https://dashboard.ngrok.com/email/confirmation?code=ajzjwHJxZS>

If you did not sign up for ngrok, please ignore this email.

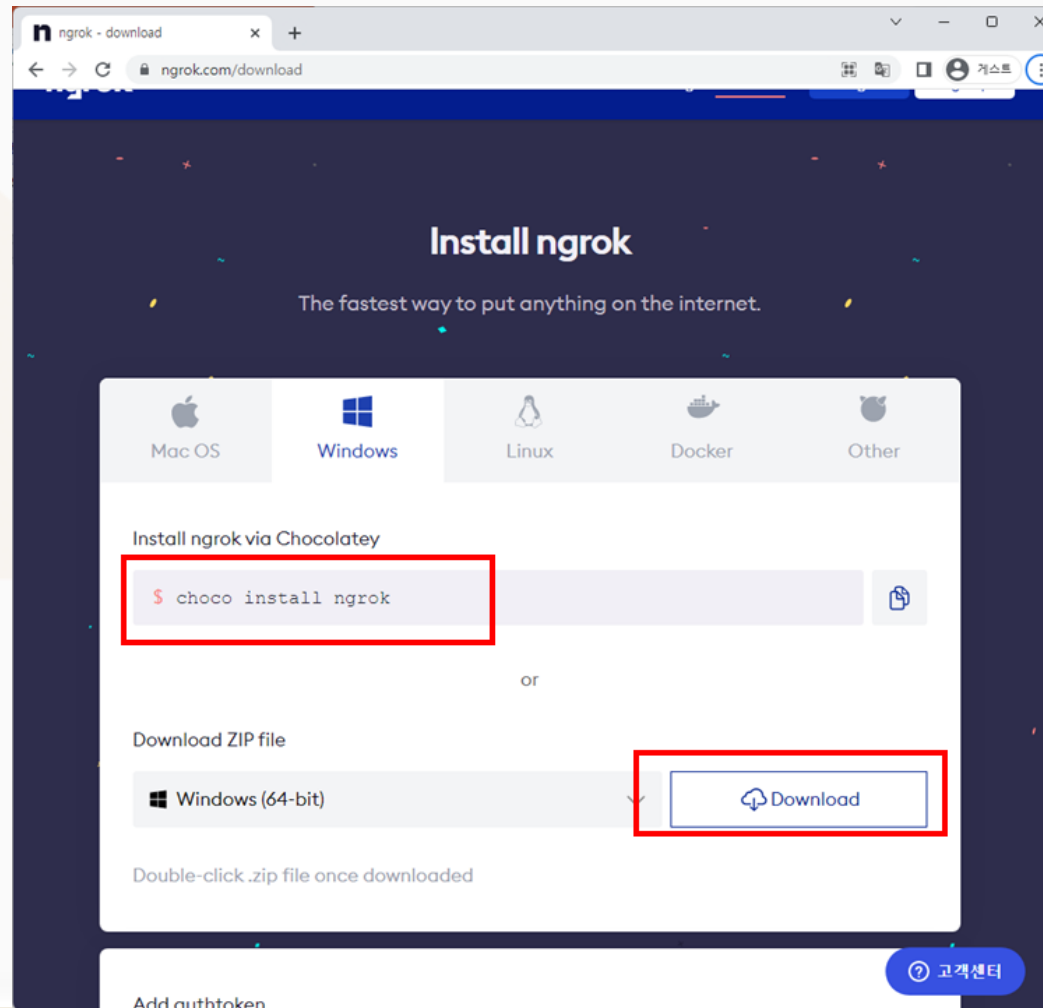
This is an automated message. Please do NOT reply to this email.

Thanks!



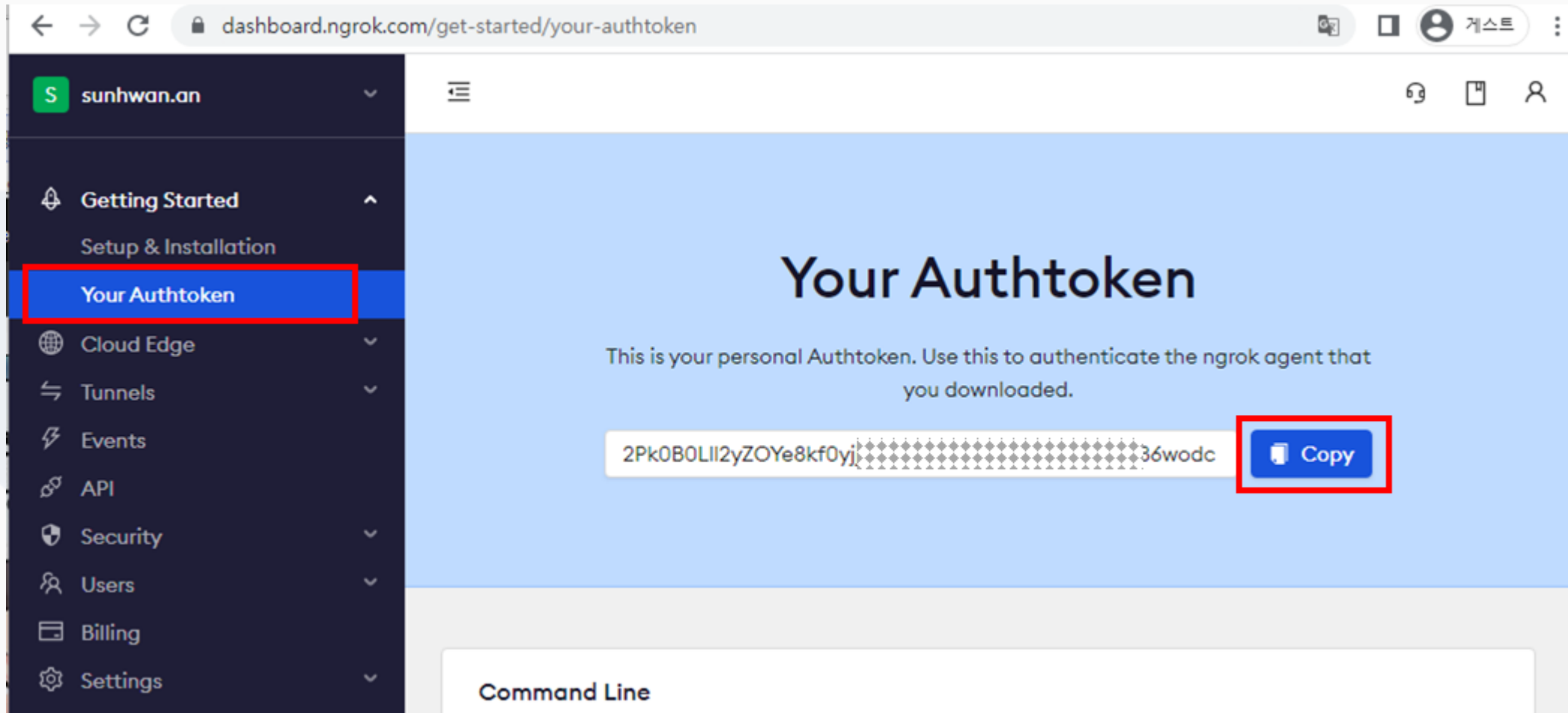
## Ngrok 설치

-<https://ngrok.com/download> 에서 프로그램을 다운 받고 압축을 풀어 둡니다.



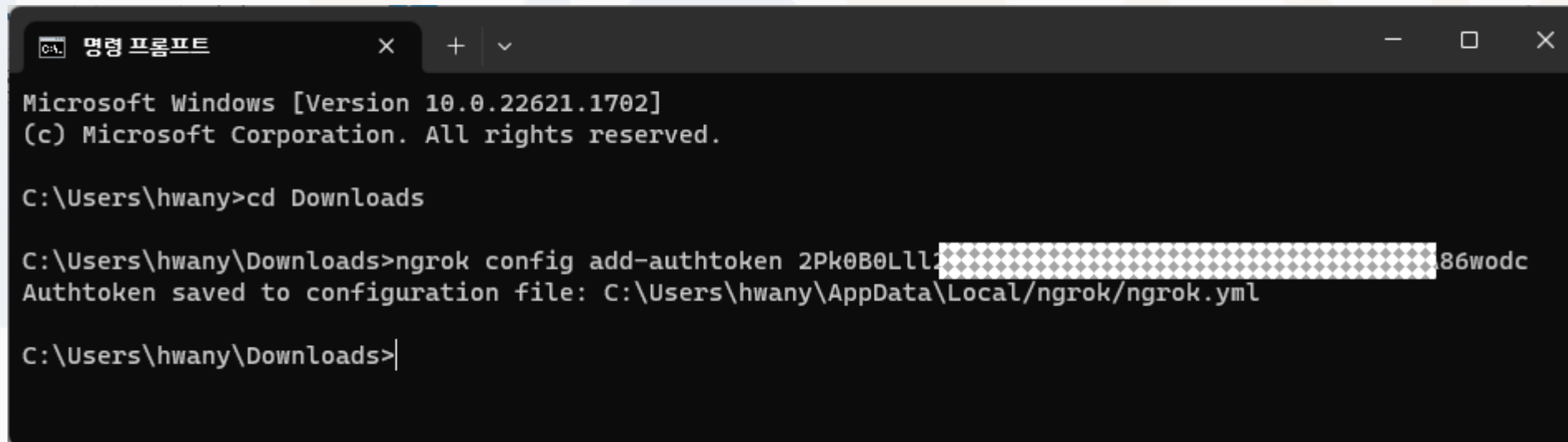
## Ngrok 인증키 받기

-https://ngrok.com에 로그인하고 “Your Authtoken” 메뉴에서 인증키를 복사해 둡니다.



## Ngrok 인증키값 저장하기

- "ngrok config add-authtoken 인증키값" 명령을 프롬프트 창에서 실행하여 인증키 값을 저장해 둡니다.



```
명령 프롬프트
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hwany>cd Downloads

C:\Users\hwany\Downloads>ngrok config add-authtoken 2Pk0B0L1l1[redacted]86wodc
Authtoken saved to configuration file: C:\Users\hwany\AppData\Local\ngrok\ngrok.yml

C:\Users\hwany\Downloads>
```

## Ngrok 실행

- "ngrok tcp 서버IP:서버포트" 명령을 프롬프트 창에서 실행하여 웹서버를 외부로 호스팅합니다.

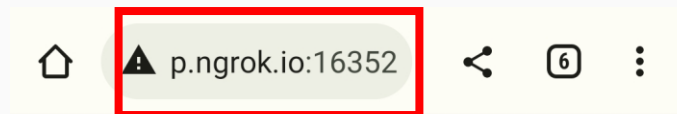
```
명령 프롬프트 - ngrok tcp 192.168.0.1:8080 x + v
ngrok (Ctrl+C to quit)
Announcing ngrok-go: The ngrok agent as a Go library: https://ngrok.com/go

Session Status      online
Account             sunhwan.an (Plan: Free)
Version             3.3.0
Region              Japan (jp)
Latency              -
Web Interface        http://127.0.0.1:4040
Forwarding            tcp://0.tcp.jp.ngrok.io:13129 -> 192.168.0.18:80

Connections
ttl    opn    rt1    rt5    p50    p90
0       0       0.00   0.00   0.00   0.00
```

## 외부 접속 확인

-프롬프트 창에서 확인한 url을 외부 네트워크에 있는 브라우저 주소창에 입력하여 결과를 확인합니다.



## ESP32 DHT Server

 온도 25.30 °C

 습도 41.00 %



[저작권 안내]

\*본 콘텐츠는 아이씨뱅크(ICBANQ)에 소유권이 있습니다. 소유권자의 허가를 받지 않고 무단으로 수정, 삭제, 배포, 상업적 사용을 할 수 없으며 위반시 민형사적 법적 처벌을 받을 수도 있습니다.