

와이파이 기반 아두이노 IoT

(weMos weather station)

지도교수 : 이상훈 교수님
20141203 박현승
20141217 정영관

CONTENTS

CONTENTS A

개발 도구

CONTENTS C

소스 설명

CONTENTS E

향후 계획

CONTENTS B

부품 설명

CONTENTS D

시연 & 테스트

CONTENTS F

Q & A

| 개발 환경 & 개발 도구



Bracket



Plotly.js



Node.js



Arduino

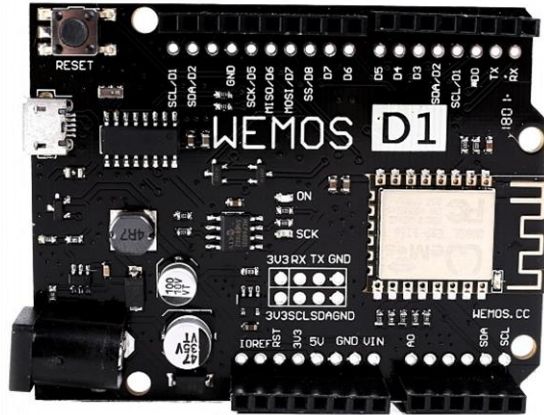


Windows 10



Fritzing

부품 설명



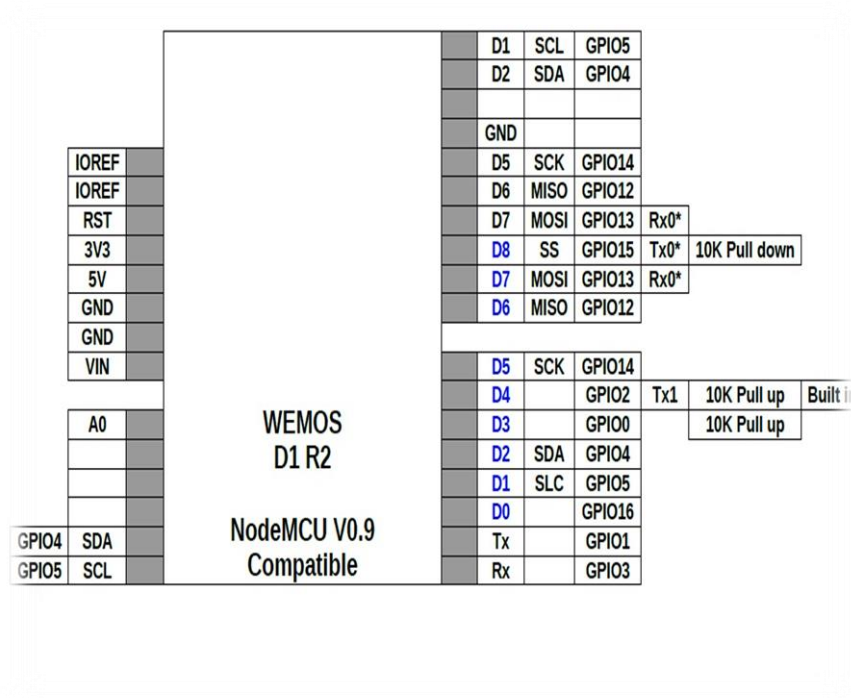
- 펌웨어 개발 없이 WIFI 사용가능
- 가격이 저렴하다
- 5v 마이크로 USB 케이블 이용

VS



- 현재 상용화 되지 않음
- 가격대가 비쌘
- AB형 USB케이블 이용

weMos 란?



weMos R1 D2란 ?

아두이노 우노 보드에 ESP-8266이 결합된 보드

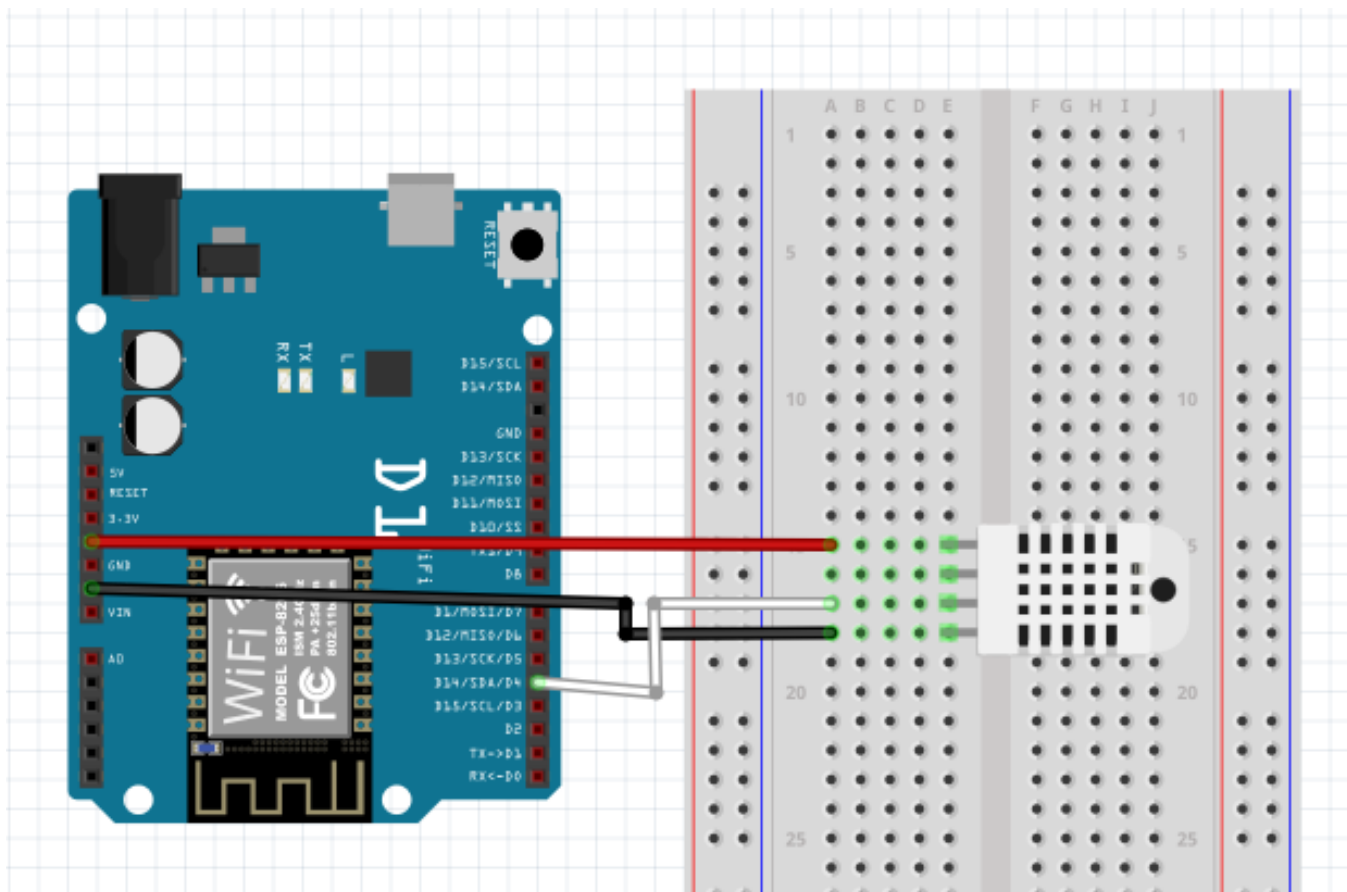
- 아두이노 IDE 지원
- 16개의 GPIO를 가지고 있음

GPIO

<General-Purpose Input/Output>

프로세서나 컨트롤러 등에서 입력을 받거나 출력으로 특정 장치를 제어하여 사용하도록 준비된 입출력 포트

회로도



아두이노 소스 (전역 변수)

```
#include <ESP8266WiFi.h>
#include "DHT.h"
#define DHTTYPE DHT22

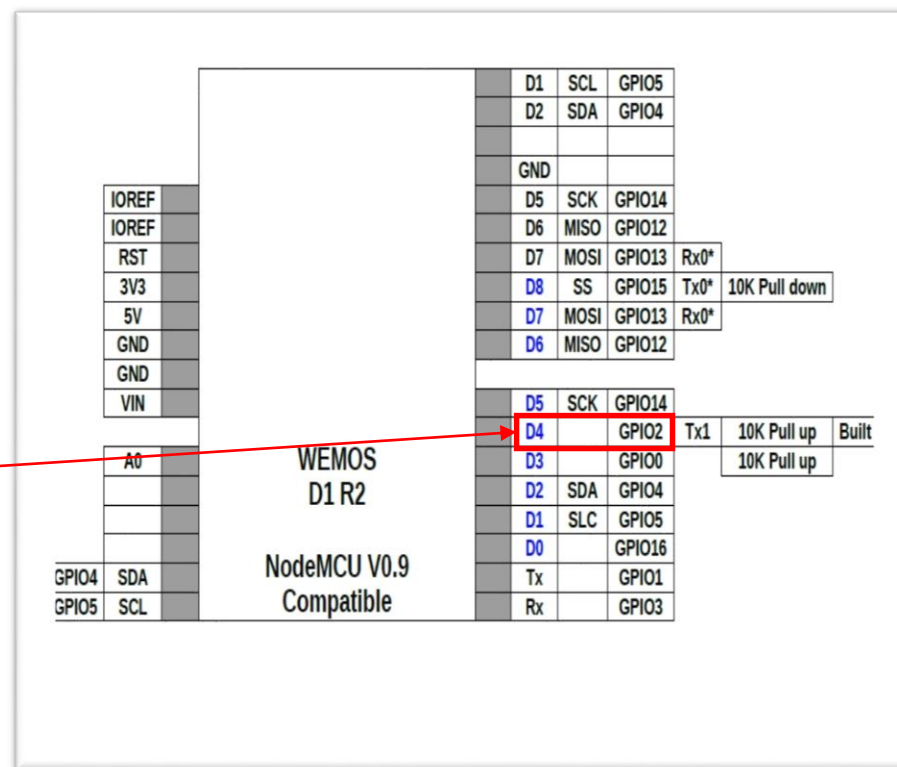
const char* ssid = "SSID";
const char* password = "Password";

WiFiServer server(80);

const int DHTPin = 2; // weMos 핀맵에 따라 D4에 연결한다.

DHT dht(DHTPin, DHTTYPE);

static char celsiusTemp[7];
static char fahrenheitTemp[7];
static char humidityTemp[7];
```



- 각 데이터가 담길 배열을 선언해준다

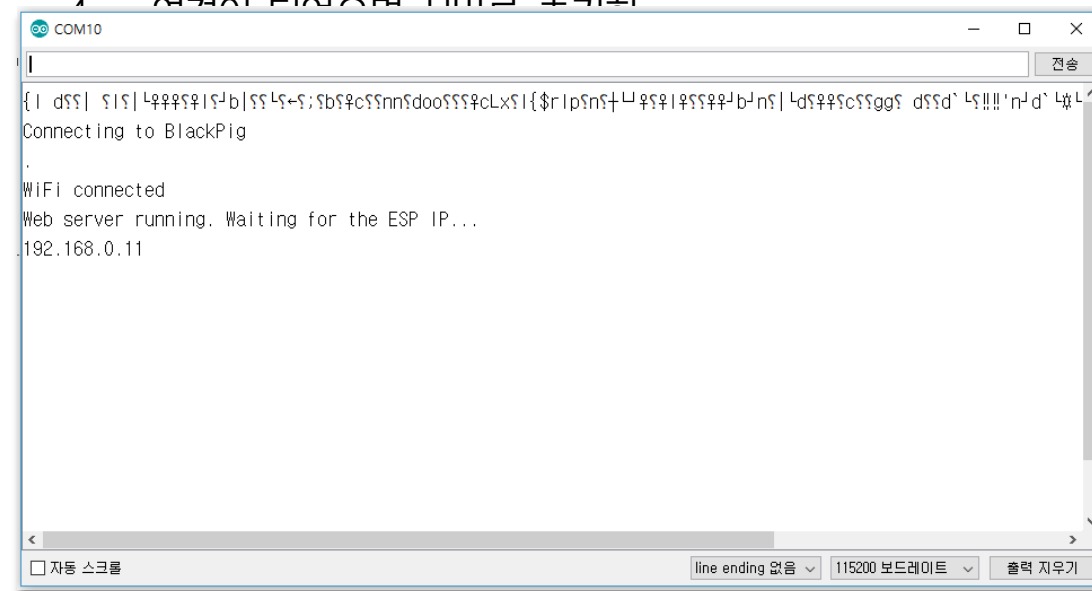
를 불러온다

아두이노 소스 (Set up)

```
void setup() {  
  Serial.begin(115200);  
  delay(10);  
  
  dht.begin();  
  
  Serial.println();  
  Serial.print("Connecting to ");  
  Serial.println(ssid);  
  delay(10000);  
  
  WiFi.begin(ssid, password);  
  
  while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
  }  
  Serial.println("");  
  Serial.println("WiFi connected");  
  
  server.begin();  
  Serial.println("Web server running. Waiting for the ESP IP...");  
  delay(10000);  
  
  Serial.println(WiFi.localIP());  
}
```

1. dht 온습도 센서를 초기화
2. 공유기에 접속을 시도
3. 연결이 되지 않으면 "."을 출력

4. 연결이 되었으면 IP를 출력



아두이노 소스 (loop)

```
void loop() {
  WiFiClient client = server.available();

  if (client) {
    Serial.println("New client");

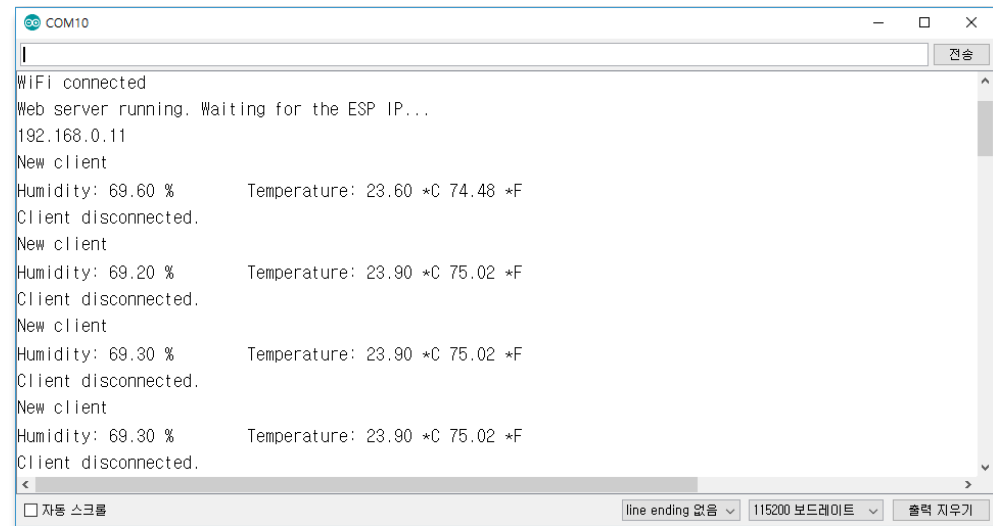
    boolean blank_line = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();

        if (c == '\n' && blank_line) {

          float h = dht.readHumidity();
          float t = dht.readTemperature();
          float f = dht.readTemperature(true);

          if (isnan(h) || isnan(t) || isnan(f)) {
            Serial.println("Failed to read from DHT sensor!");
            strcpy(celsiusTemp, "Failed");
            strcpy(fahrenheitTemp, "Failed");
            strcpy(humidityTemp, "Failed");
          }
          else{
            dtostrf(t, 6, 2, celsiusTemp);
            dtostrf(f, 6, 2, fahrenheitTemp);
            dtostrf(h, 6, 2, humidityTemp);

            Serial.print("Humidity: ");
            Serial.print(h);
            Serial.print(" %\n\t Temperature: ");
            Serial.print(t);
            Serial.print(" *C ");
            Serial.print(f);
            Serial.println(" *F");
          }
        }
      }
    }
  }
}
```



1. 서버가 연결되었다면 변수 h,t,h에 온습도 데이터를 담는다
2. isnan(not a number)함수를 사용하여 각 데이터값이 제대로 들어왔는지 확인.
3. 제대로 들어온 데이터를 dtostrf 함수를 사용하여 배열에 담는다
4. 시리얼 모니터로 각각의 값을 확인

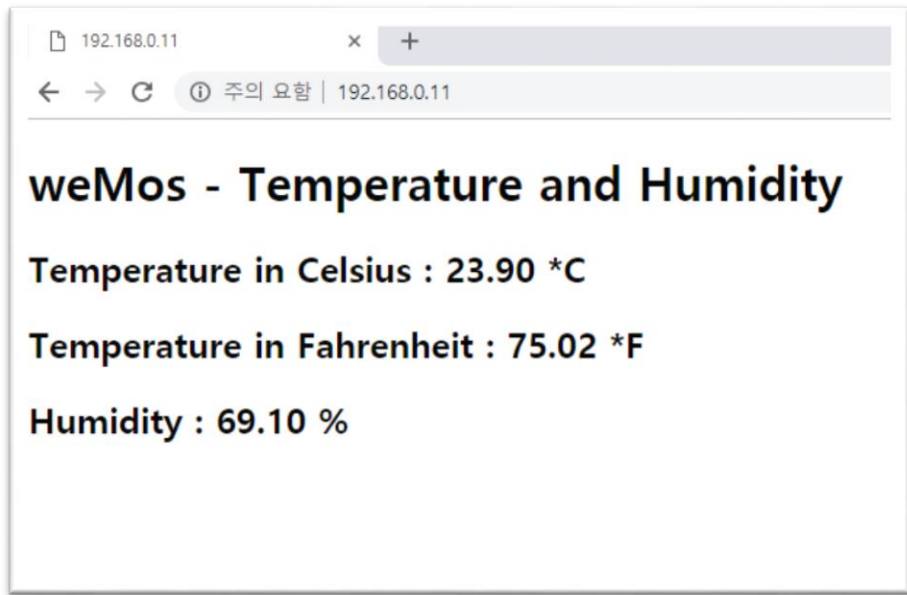
아두이노 소스 (loop)

```

client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("Connection: close");
client.println("Refresh: 2");
client.println();

client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<head></head>");
client.print("<body><h1>weMos - Temperature and Humidity</h1>");
client.print("<dl id='data'><dt class='Celsius'>");
client.print("<h2>Temperature in Celsius : ");
client.print(celsiusTemp);
client.print(" *C</h2></dt><dt class='Fahrenheit'>");
client.print("<h2>Temperature in Fahrenheit : ");
client.print(fahrenheitTemp);
client.print(" *F</h2></dt><dt class='Humidity'>");
client.print("<h2>Humidity : ");
client.print(humidityTemp);
client.print(" %</h2></dt></dl>");
client.print("</body>");
client.println("</html>");
break;
}
if (c == '\n') {
    blank_line = true;
}
else if (c != '\r') {
    blank_line = false;
}
}
}
delay(1);
client.stop();
Serial.println("Client disconnected.");
}
}

```



2. HTTP 프로토콜 셋팅을 하고 웹 페이지가 2초마다 refresh되도록 설정
3. 클라이언트로 접속했을 때 생성된 웹 페이지와 데이터 값을 확인할 수 있다

Node.js 소스 (package)

```
8  —  "dependencies": {
9      "cookie-parser": "~1.4.3",
10     "debug": "~2.6.9",
11     "http-errors": "~1.6.2",
12     "jade": "~1.11.0",
13     "morgan": "~1.9.0",
14     "request": "^2.88.0",
15     "cheerio": "^1.0.0-rc.2",
16     "cors": "^2.8.4",
17     "express": "^4.15.2",
18     "socket.io": "^1.7.3"
19   }
20 }
```

- Node.js 사용
- cheerio, cors, express, socket.io 등 필요한 모듈 설치

Node.js 소스 (index.js)

```
1 var cheerio = require('cheerio');
2 var request = require('request');
3 var io = require('socket.io').listen(3000, function (req, res) {
4   console.log('Listening on port 3000');
5 });
6
7 var dht22data = [];
8 var dateStr = '';
9 var CelsiusData = '';
10 var FahrenheitData = '';
11 var HumidityData = '';
12
13
14 var url = 'http://192.168.0.11/';
15
16 var Celsius = '';
17
18 function getCelsius() {
19
20   request(url, function (err, res, body) {
21
22     var $ = cheerio.load(body);
23
24     $('#data .Celsius').each(function () {
25       Celsius = $(this).text().substring(26, 31);
26     });
27   });
28   return Celsius;
29 }
```

- 모듈 require
- Socket 포트 설정
- 웹 스크리핑

```
<dt id="data">
  <dt class="Celsius">
    <h2>Temperature in Celsius : celsiusTemp *C</h2>
  </dt>
```

Node.js 소스 (index.js)

```
79 - io.sockets.on('connection', function (socket) {  
80  
81 -     socket.on('message', function (msg) {  
82         console.log(msg);  
83     });  
84  
85     socket.on('disconnect', function () {});  
86 });  
--  
88 - setInterval(function () {  
89     dateStr = getDateString();  
90     CelsiusData = getCelsius();  
91     FahrenheitData = getFahrenheit();  
92     HumidityData = getHumidity();  
93     dht22data[0] = dateStr; // Date  
94     dht22data[1] = CelsiusData; // temperature data  
95     dht22data[2] = FahrenheitData; // Fahrenheit data  
96     dht22data[3] = HumidityData; // humidity data  
97     io.sockets.emit('message', dht22data);  
98     console.log("COMSI," + dht22data);  
99 }, 2000);  
100  
101 });  
102
```

- Socket 기본 설정
- setInterval 함수 사용

웹 클라이언트 소스 (html)

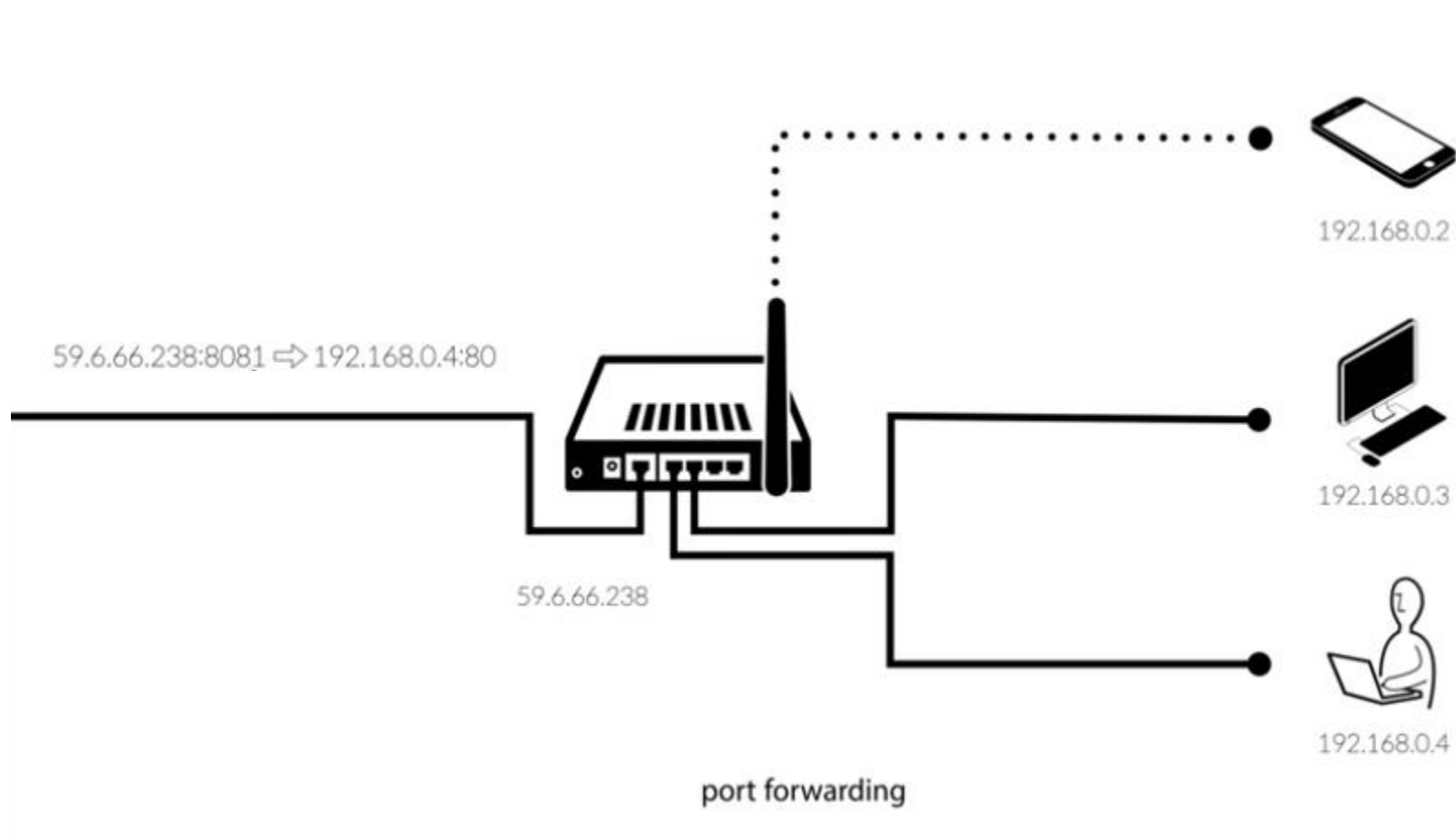
```
37 - <script>
38     var streamPlot = document.getElementById('myDiv');
39     var ctime = document.getElementById('time');
40     var tArray = [], // time of data arrival
41         y1Track = [], // value of sensor 1 : temperature
42         y2Track = [], // value of sensor 2 : fahrenheit
43         y3Track = [], // value of sensor 3 : humidity
44         numPts = 50, // number of data points in x-axis
45         dtda = [], // 1 x 4 array : [date, data1, data2, data3] from sensors
46         preX = -1,
47         preY = -1,
48         preZ = -1,
49         initFlag = true;
```

```
51     var socket = io.connect('http://175.199.157.52:3000'); // port = 3000
52 -     socket.on('connect', function() {
53 -         socket.on('message', function(msg) {
54             // initial plot
55 -             if (msg[0] != '' && initFlag) {
56                 dtda[0] = msg[0];
57                 dtda[1] = parseFloat(msg[1]); // temperature
58                 dtda[2] = parseFloat(msg[2]); // fahrenheit
59                 dtda[3] = parseFloat(msg[3]); // humidity
60                 init();
61                 initFlag = false;
62             }
63
64             dtda[0] = msg[0];
65             dtda[1] = parseFloat(msg[1]);
66             dtda[2] = parseFloat(msg[2]);
67             dtda[3] = parseFloat(msg[3]);
68
```

- 데이터 변수 선언

- 소켓서버 연결 및 데이터 파싱

포트 포워딩이란?



웹 클라이언트 소스 (html)

```

69 // Only when any of temperature or Luminosity is different
70 // from the previous one, the screen is redrawn.
71 if (dtdda[1] != preX || dtdda[2] != preY || dtdda[3] != preZ) { // any change?
72     preX = dtdda[1];
73     preY = dtdda[2];
74     preZ = dtdda[3];
75
76 // when new data is coming, keep on streaming
77 ctime.innerHTML = dtdda[0];
78 gauge_temperature.setValue(dtdda[1]) // temperature gauge
79 gauge_fahrenheit.setValue(dtdda[2]); // fahrenheit gauge
80 gauge_humidity.setValue(dtdda[3]); // humidity gauge
81
82 tArray = tArray.concat(dtdda[0]);
83 tArray.splice(0, 1); // remove the oldest data
84 y1Track = y1Track.concat(dtdda[1]);
85 y1Track.splice(0, 1); // remove the oldest data
86 y2Track = y2Track.concat(dtdda[2]);
87 y2Track.splice(0, 1);
88 y3Track = y3Track.concat(dtdda[3]);
89 y3Track.splice(0, 1);
90
91 var update = {
92     x: [tArray, tArray, tArray],
93     y: [y1Track, y2Track, y3Track]
94 }
95
96 Plotly.update(streamPlot, update);
97
98 }
99
100 });

```

- 데이터 업데이트

```

102 function init() { // initial screen ()
103     // starting point : first data (temp, lux)
104     for (i = 0; i < numPts; i++) {
105         tArray.push(dtdda[0]); // date
106         y1Track.push(dtdda[1]); // sensor 1 (temp)
107         y2Track.push(dtdda[2]); // sensor 2 (humi)
108         y3Track.push(dtdda[3]); // sensor 3 (lux)
109     }
110
111     Plotly.plot(streamPlot, data, layout);
112 }

```

- 그래프 초기화

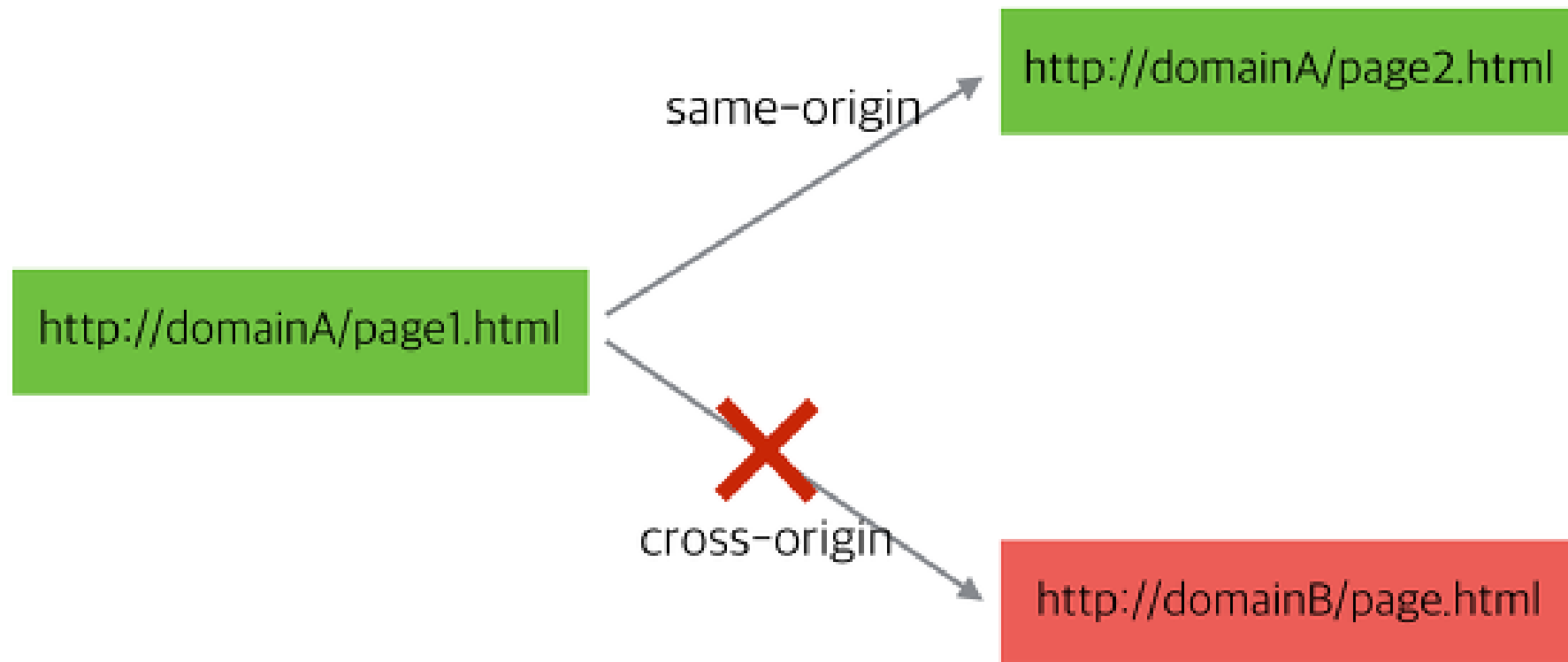
Node.js 소스 (server)

```
7 var indexRouter = require('./routes/index');
8 var app = express();
9
10 var cors = require('cors')();
11 app.use(cors); //Cross-Origin Resource Sharing
12
13 app.use(express.static(path.join(__dirname, 'public')));
14 app.use('/', indexRouter);
```

```
15 var port = normalizePort(process.env.PORT || '3030');
16 app.set('port', port);
17
```

- Cors 설정
- Express 포트 설정

| Cors(Cross-Origin Resource Sharing) 란?



Node.js 소스 (server)

The screenshot displays two windows. The top window is a terminal running 'npm start' in a directory 'C:\Users\rhks1\NodeScrapeGen\scrape'. It shows the output of 'node ./bin/www', including GET requests for 'plotly.html', 'gauge.min.js', and 'Fonts/digital-7-mono.ttf' with their respective response times and status codes.

The bottom window is the PM2 Dashboard. It shows a process list with one process named 'index' using 40 MB of memory. The Global Logs section displays a series of 'index > COMS!' log entries with timestamps and IP addresses. The Custom metrics section shows Event Loop Latency (0.77ms), Active handles (9), and Active requests (0). The Metadata section provides details about the application, including its name ('index'), restarts (14), uptime (62s), script path ('C:\Users\rhks1\NodeScrape\index.js'), and script arguments (N/A).

```
npm
C:\Users\rhks1\NodeScrapeGen\scrape>npm start
> scrape@0.0.0 start C:\Users\rhks1\NodeScrapeGen\scrape
> node ./bin/www

GET /plotly.html 304 4.401 ms --
GET /gauge.min.js 304 0.822 ms --
GET /Fonts/digital-7-mono.ttf 404 471.842 ms - 1172
```

PM2 Dashboard

Process list

id	name	memory
0	index	Mem: 40 MB

Global Logs

```
index > COMS!,2018-11-13 23:45:19.343,22.70,72.86,40.80
index > COMS!,2018-11-13 23:45:22.344,22.70,72.86,40.90
index > COMS!,2018-11-13 23:45:25.346,22.70,72.86,40.90
index > COMS!,2018-11-13 23:45:28.347,22.70,72.86,41.00
index > COMS!,2018-11-13 23:45:31.348,22.70,72.86,41.00
index > COMS!,2018-11-13 23:45:34.349,22.70,72.86,41.10
index > COMS!,2018-11-13 23:45:37.351,22.70,72.86,41.00
index > COMS!,2018-11-13 23:45:40.352,22.70,72.86,40.90
index > COMS!,2018-11-13 23:45:43.352,22.70,72.86,40.80
index > COMS!,2018-11-13 23:45:46.354,22.70,72.86,40.80
index > COMS!,2018-11-13 23:45:49.355,22.70,72.86,40.80
index > COMS!,2018-11-13 23:45:52.355,22.80,73.04,40.70
index > COMS!,2018-11-13 23:45:55.356,22.70,72.86,40.60
index > COMS!,2018-11-13 23:45:58.357,22.80,73.04,40.70
index > COMS!,2018-11-13 23:46:01.358,22.80,73.04,40.70
index > COMS!,2018-11-13 23:46:04.359,22.80,73.04,40.70
index > COMS!,2018-11-13 23:46:07.360,22.80,73.04,40.70
index > COMS!,2018-11-13 23:46:10.361,22.80,73.04,40.80
index > COMS!,2018-11-13 23:46:13.363,22.80,73.04,40.70
```

Custom metrics

Metric	Value
Event Loop Latency	0.77ms
Active handles	9
Active requests	0

Metadata

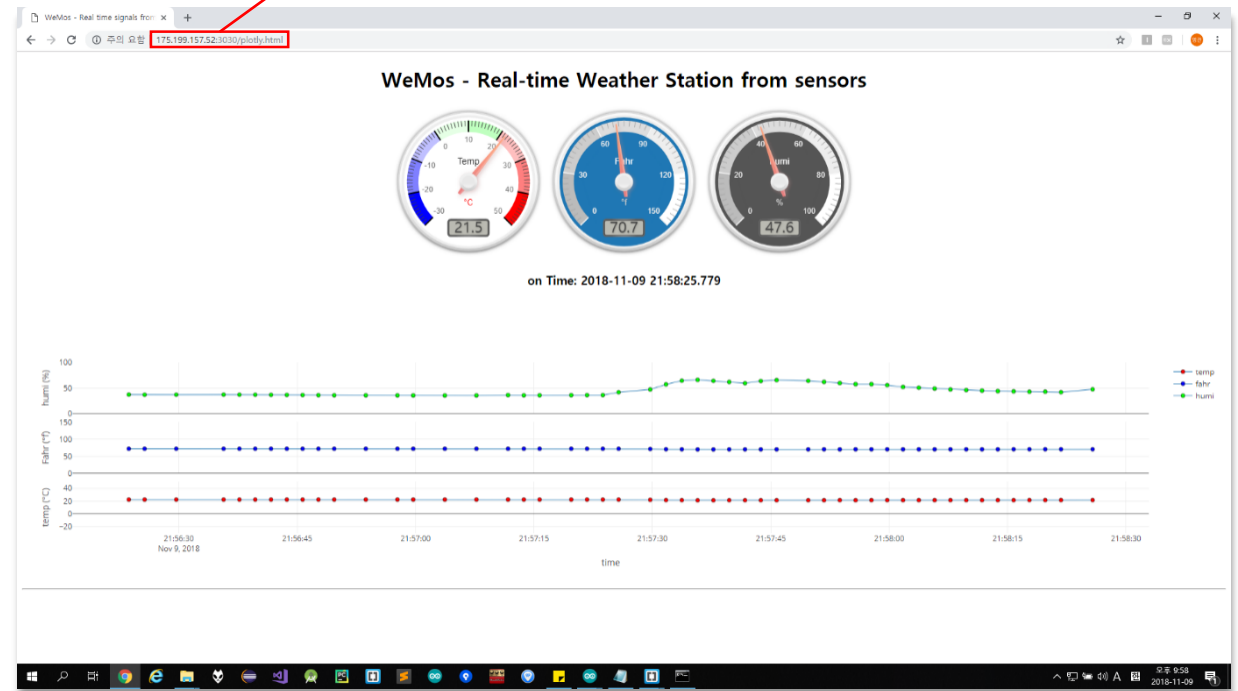
Key	Value
App Name	index
Restart	14
Uptime	62s
Script path	C:\Users\rhks1\NodeScrape\index.js
Script args	N/A

left/right: switch boards | up/down/mouse: scroll | Ctrl-C: exit To go further check out <https://pm2.io/>

결과 화면



<http://175.199.157.52:3030/plotly.html>



I 시연 & 테스트

<http://chaos.inje.ac.kr:3030/wifi1.html>

<http://chaos.inje.ac.kr:3030/wifi2.html>

<http://chaos.inje.ac.kr:3030/wemos1.html>

<http://chaos.inje.ac.kr:3030/wemos2.html>



| 향후 계획



| GitHub 오픈소스

github.com/swarthyPig/WeMos_DHT22_Wireless

The screenshot shows the GitHub repository page for `swarthyPig / WeMos_DHT22_Wireless`. The page includes a navigation bar with links to Features, Business, Explore, Marketplace, and Pricing, along with a search bar and a sign-in/sign-up button. Below the navigation bar, the repository name is displayed with icons for Watch (0), Star (0), and Fork (0). The main content area shows a banner for "Join GitHub today" with a "Sign up" button. Below the banner, there is a section for "No description, website, or topics provided." and a table of commits. The table lists the following commits:

Commit	Message	Time
swarthyPig	Update README.md	Latest commit e775d8f 5 days ago
NodeScape	Update plotly.html	5 days ago
NodeScapeGenerator(new version)/scrape	Add files via upload	5 days ago
image	Add files via upload	5 days ago
DHT22WeMos.ino	Add files via upload	6 days ago
README.md	Update README.md	5 days ago
exam.html	Add files via upload	16 days ago

Below the table, the README.md file is displayed, showing the title "WeMos_DHT22_Wireless" and the date "18/11/09".

The background of the image is a dark gray field filled with numerous white question marks of varying sizes and orientations, creating a sense of depth and inquiry. In the center of the image, there is a semi-transparent dark gray circle. Inside this circle, the text "Q & A" is written in a clean, white, sans-serif font.

Q & A



THANK YOU

(weMos weather station)