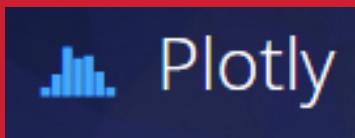


Arduino-IOT

[wk09]

cds_dht22 + node
MongoDB-I

Visualization of Signals using Arduino,
Node.js & storing signals in MongoDB
& mining data using Python



Drone-IoT-Comsi, INJE University
2nd semester, 2022

Email : chaos21c@gmail.com



My ID

ID를 확인하고 github에 repo 만들기

AA01	강대진	AA13	박제홍
		AA14	심준혁
AA03	김성우	AA15	이상혁
AA04	김정현	AA16	이승무
		AA17	이승준
AA06	김창연	AA18	이준희
AA07	김창욱	AA19	이현준
AA08	김태화	AA20	임태형
AA09	남승현	AA21	정동현
AA10	류재환		
AA11	박세훈	AA23	정희서
AA12	박신영	AA24	최재형

위의 id를 이용해서 github에 repo를 만드시오.

Option: 아두이노용 실습 과제 – AAnn

Public, README.md check



[Practice]

◆ [wk07-08]

- **RT Data Visualization with node.js**
- **Usage of gauge.js**
- **Complete your plotly-node project**
- **Upload folder: aann-rpt08**
- **Use repo “aann” in github**

Wk07/08 : Practice : aann-rpt08

◆ [Target of this week]

- Complete your works
- Save your outcomes and upload outputs in github

제출폴더명 : **aann-rpt08**

- 암축할 파일들

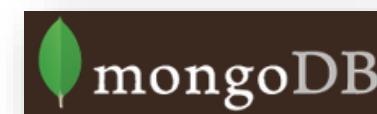
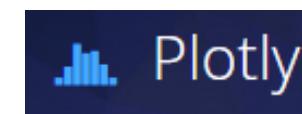
- ① **AAnn_DS_30timestamps.png**
- ② **AAnn_DS_multiple_axis.png**
- ③ **AAnn_cds_gauge.png**
- ④ **AAnn_cds_change.png**
- ⑤ **AAnn_DS_cds_tmp36.png**
- ⑥ **All *.ino**
- ⑦ **All *.js**
- ⑧ **All *.html**



Purpose of AA

주요 수업 목표는 다음과 같다.

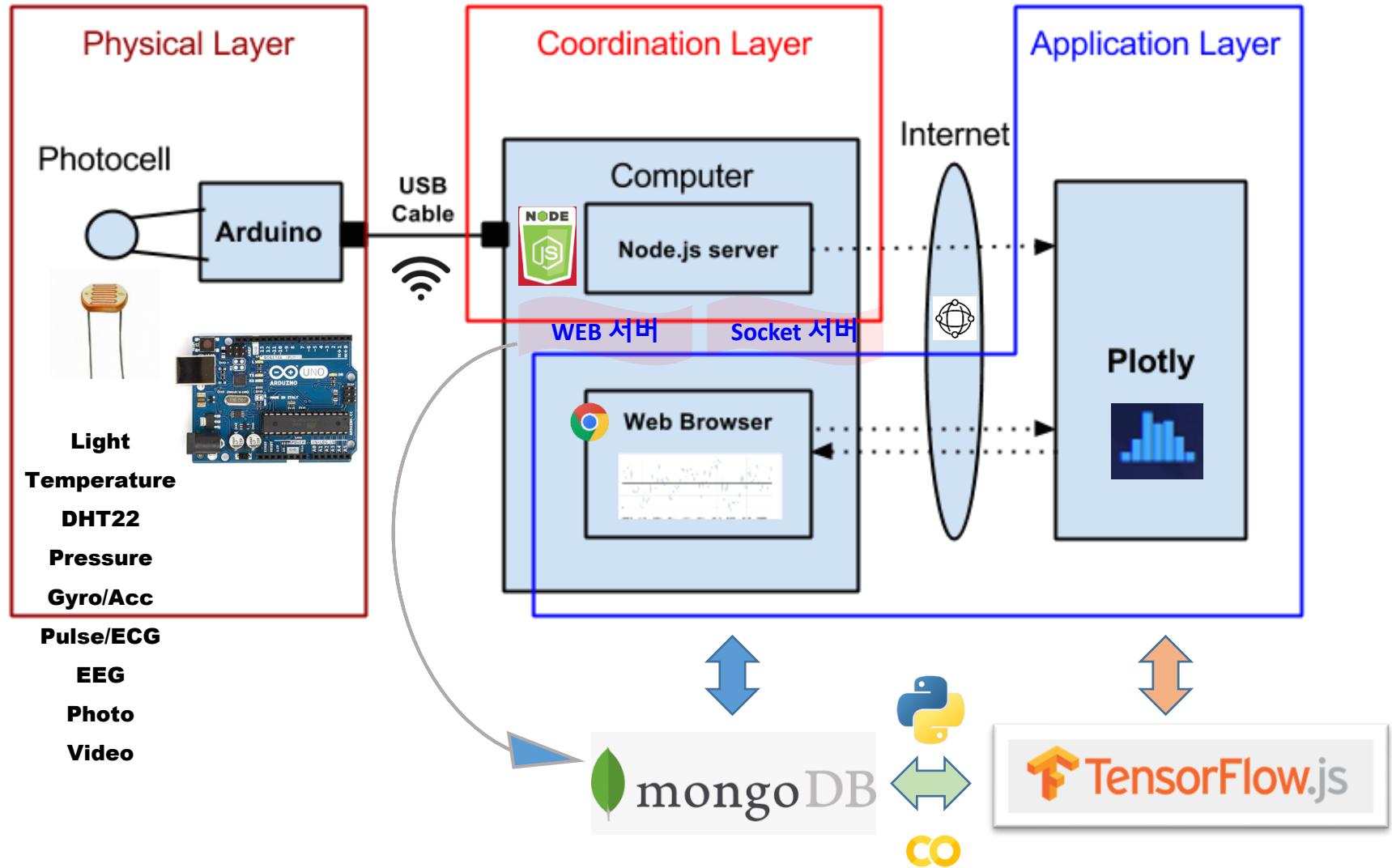
1. Node.js를 이용한 아두이노 센서 신호 처리
2. Plotly.js를 이용한 아두이노 센서 신호 시각화
3. MongoDB에 아두이노 센서 데이터 저장 및 처리



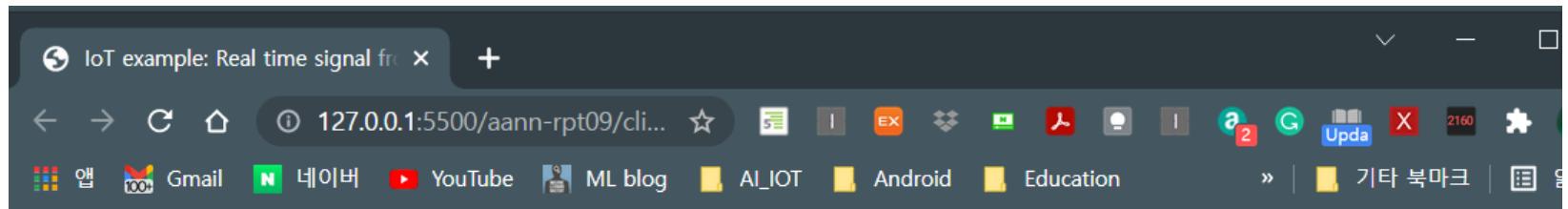
4. 저장된 IoT 데이터의 마이닝 (파이썬 코딩)



Layout [H S C]



on WEB monitoring Arduino data



IoT Signal from Arduino Weather Station

Real-time Signals

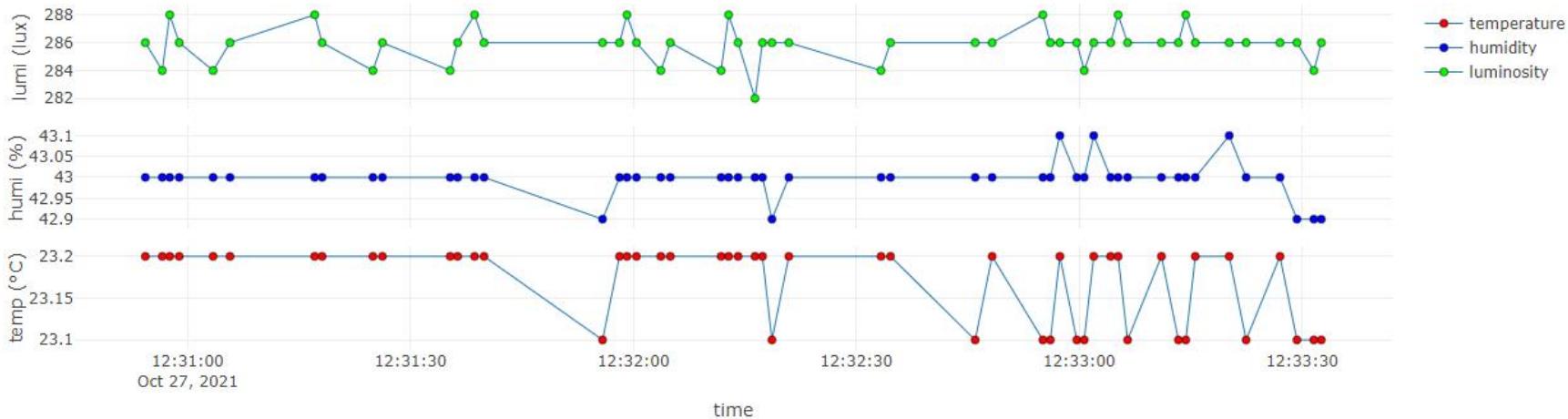
on Time: 2021-10-27 11:54:48.997

Signals (온도,습도,조도) : 23.4,42.6,286

Real-time Weather Station from sensors

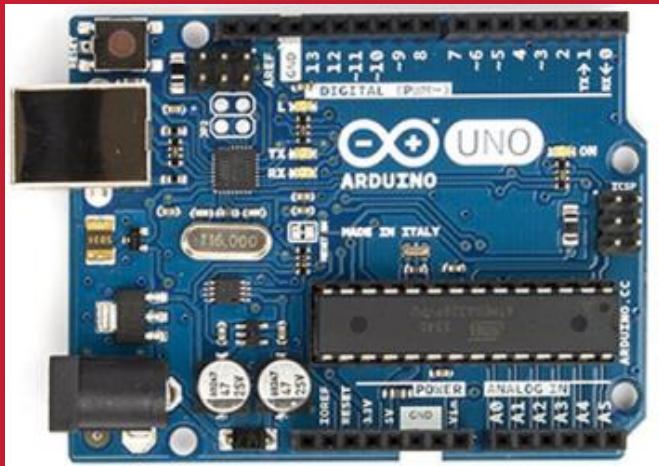


on Time: 2021-10-27 12:33:32.600





CdS + DHT22

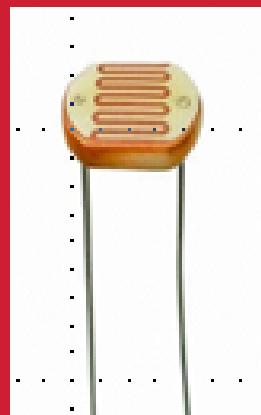


+ `plotly.js`

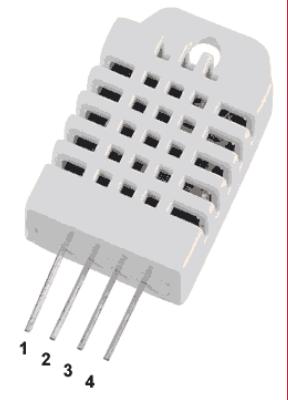
Node project

Multi-sensors

CdS + DHT22



DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND





DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

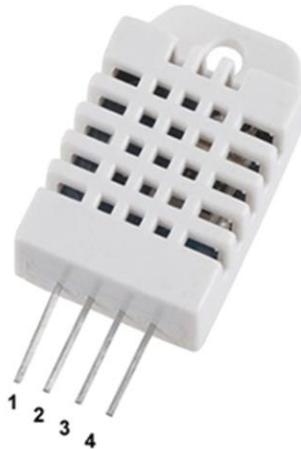


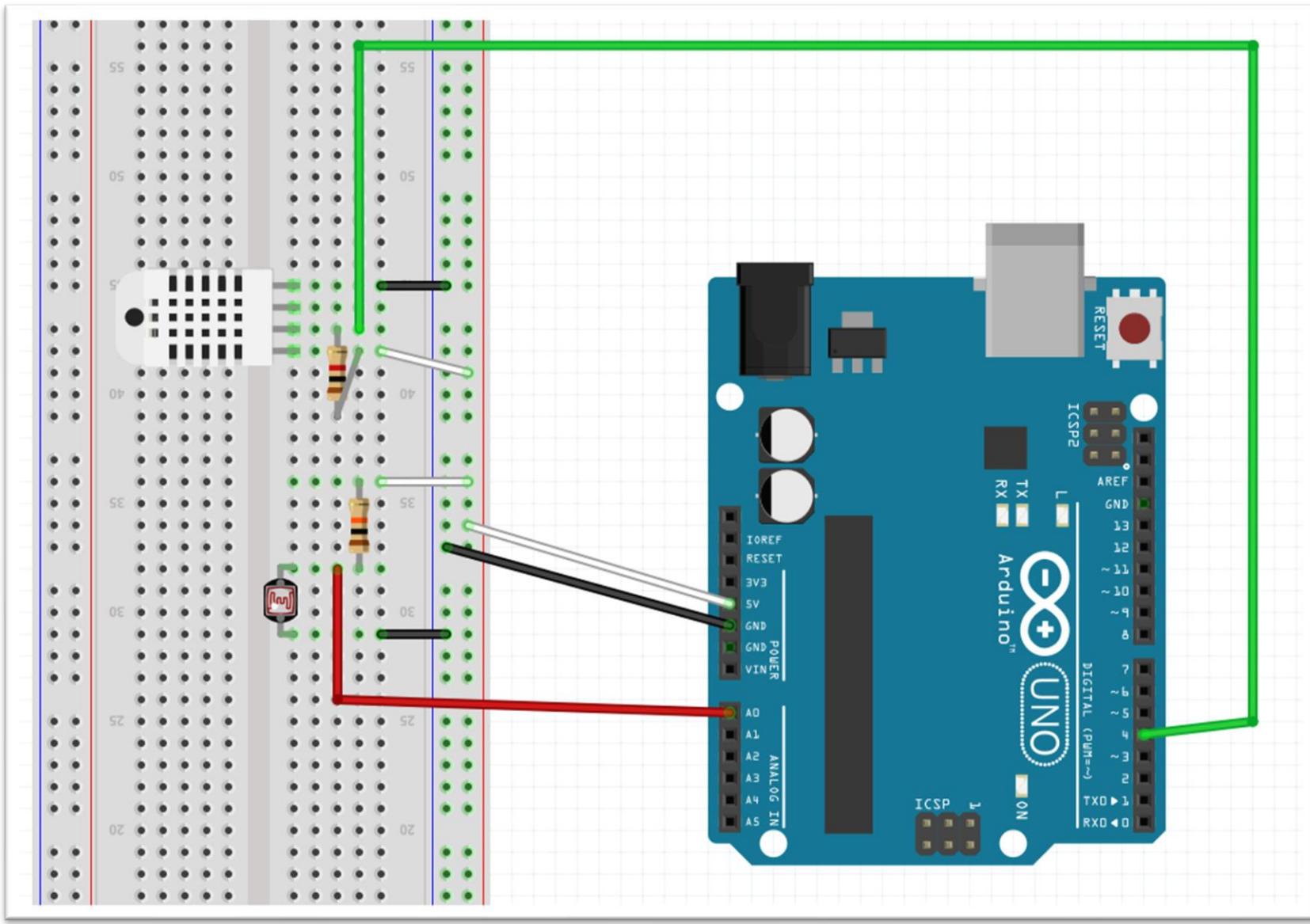
그림 8-7 DHT22 pin 구조

- 3 ~ 5V power and I/O
- 2.5mA max current
- [0-100%] humidity readings with 2-5% accuracy
- [-40 to 80°C] temperature readings $\pm 0.5^{\circ}\text{C}$ accuracy
- 0.5 Hz sampling rate

<https://learn.adafruit.com/dht/overview>



A5.7.1 DHT22 + CdS circuit



DHT22[D4] + 1 kΩ, CdS[A0] + 10 k Ω



A5.7.2 DHT22 + CdS : DHT library

Features Business Explore Marketplace Pricing This repository S

adafruit / DHT-sensor-library

Code Issues 21 Pull requests 15 Projects 0 Wiki Insights

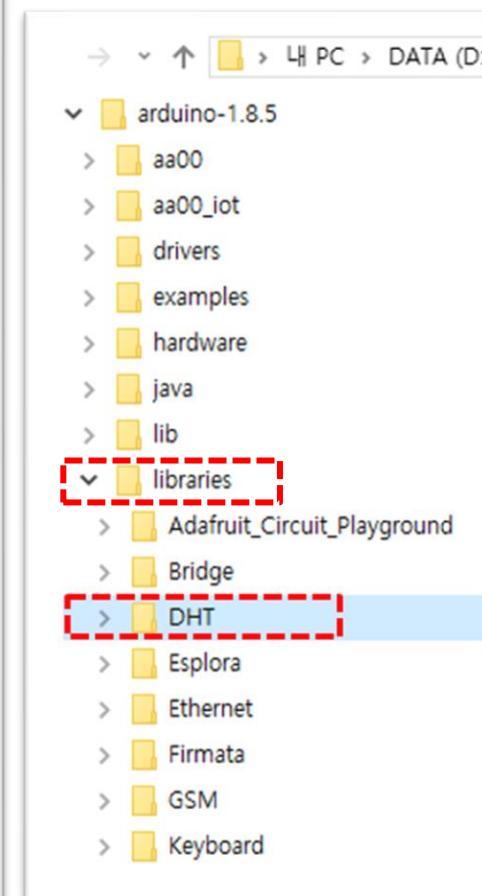
Arduino library for DHT11DHT22, etc Temp & Humidity Sensors <http://www.ladyada.net/learn/>

54 commits 1 branch 8 releases

Branch: master New pull request

microbuilder Merged unified and raw libraries

File	Description
.github	Add GitHub issue template
examples	Merged unified and raw libraries
DHT.cpp	Fix #44 by conditionally excluding unused port and bitmask state on n...
DHT.h	Fix #44 by conditionally excluding unused port and bitmask state on n...
DHT_U.cpp	Merged unified and raw libraries
DHT_U.h	Merged unified and raw libraries





A5.7.4 DHT22 + CdS : circuit

[1] Arduino code: AAnn_CdS_DHT22.ino

```
AAnn_CdS_DHT22.ino
1 // DHT22
2 #include "DHT.h"
3 #define DHTPIN 4
4 #define DHTTYPE DHT22
5 DHT dht(DHTPIN, DHTTYPE);
6 // CdS (LDR)
7 #define CDS_INPUT 0
8
9 void setup() {
10 dht.begin();
11 Serial.begin(9600);
12 }
```

```
42 //Voltage to Lux
43 double luminosity (int RawADC0){
44     double Vout=RawADC0*5.0/1023.0; // 5/1023
45     double lux=(2500/Vout-500)/10;
46     // lux = 500 / Rldr,
47     // Vout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
48     return lux;
49 }
```

```
14 void loop() {
15     int cds_value, lux;
16     float temp, humi;
17     // Lux from CdS (LDR)
18     cds_value = analogRead(CDS_INPUT);
19     lux = int(luminosity(cds_value));
20     // Reading temperature or humidity takes a given interval!
21     // Sensor readings may also be up to 2 seconds 'old'
22     humi = dht.readHumidity();
23     // Read temperature as Celsius (the default)
24     temp = dht.readTemperature();
25
26     // Check if any reads failed and exit early (to try again).
27     if (isnan(humi) || isnan(temp) || isnan(lux)) {
28         Serial.println("Failed to read from DHT sensor or CdS!");
29         return;
30     }
31     else {
32         Serial.print("AA00,") // 주석 처리
33         Serial.print(temp,1); // temperature, float
34         Serial.print(",");
35         Serial.print(humi,1); // humidity, float
36         Serial.print(",");
37         Serial.println(lux); // luminosity, int
38     }
39     delay(2000); // 2000 msec, 0.5 Hz
40 }
```



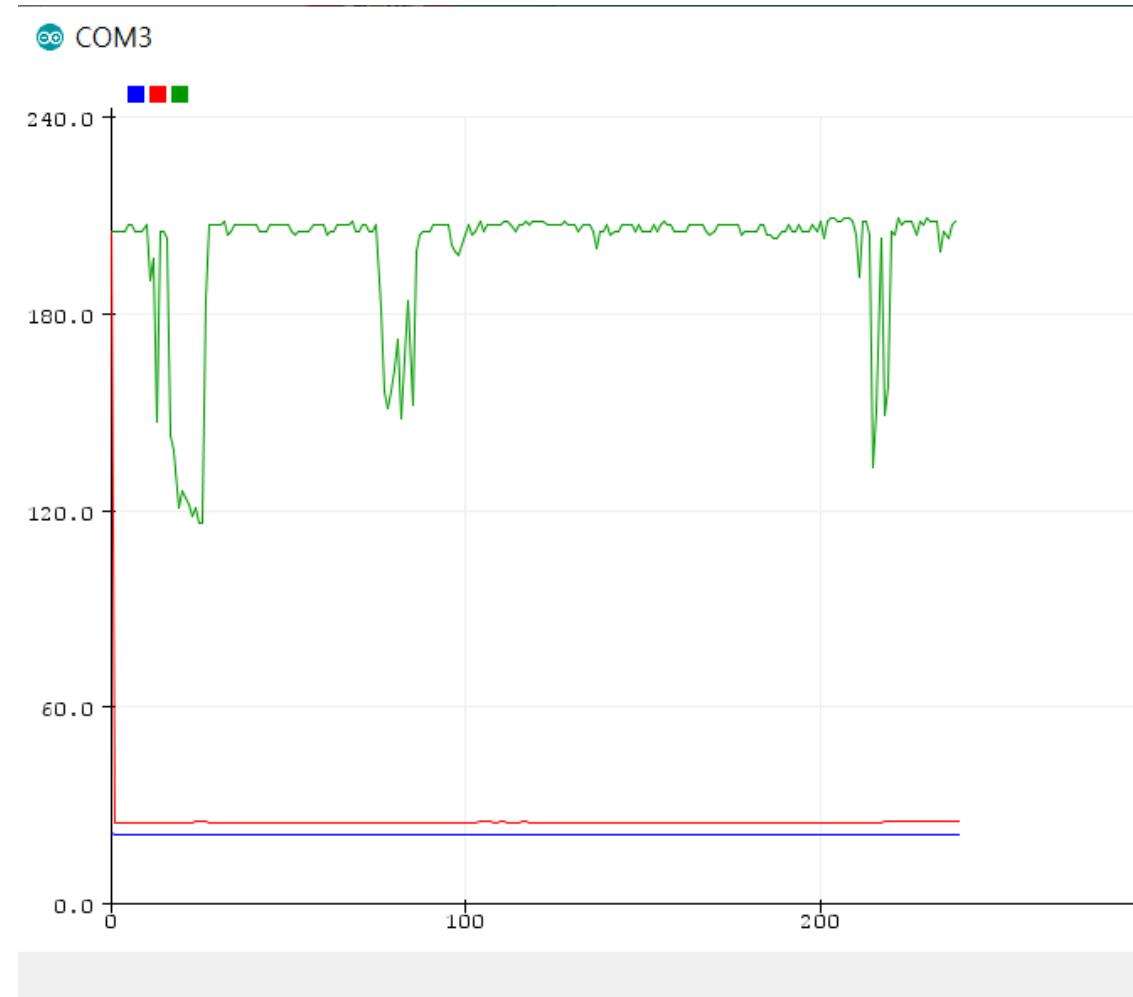
A5.7.5 DHT22 + CdS : Serial monitor

[1] Arduino code: [AAnn_CdS_DHT22.ino](#)

COM3

```
21.0,24.7,205
21.0,24.7,207
21.0,24.7,205
21.0,24.7,152
21.0,24.7,167
20.9,24.6,166
20.9,24.6,204
21.0,24.8,204
21.0,24.8,152
21.0,24.8,173
21.0,24.8,191
21.0,24.8,203
21.0,24.8,207
21.0,24.9,204
21.0,24.9,204
```

온도,습도,조도 템프





A5.7.6 DHT22 + CdS + Node.js

[2.1] NodeJS project: "cds-dht22-node project" → package.json

```
./ aann > aann-rpt09 > Node > cds_dht22 > npm package.json > ...
1  {
2      "name": "cds_tmp36",
3      "version": "1.0.0",
4      "description": "cds-dht22-node project",
5      "main": "cds_dht22_node.js",
6          ▷ 디버그
7      "scripts": {
8          "test": "echo \\\"Error: no test specified\\\" && exit 1"
9      },
10     "keywords": [
11         "cds",
12         "dht22",
13         "node",
14         "arduino"
15     ],
16     "author": "aa00",
17     "license": "MIT",
18     "dependencies": {
19         "serialport": "^9.2.4",
20         "socket.io": "^2.4.1"
21     }
}
```



A5.7.7 DHT22 + CdS + Node.js

[2.2] NodeJS code: [cds_dht22_node.js](#) ([← cds_tmp36_node.js](#) ⇒ rename)

```
// cds_dht22_node.js

var serialport = require("serialport");
var portName = "COM3"; // check your COM port!!
var port = process.env.PORT || 3000;

var io = require("socket.io").listen(port);

const Readline = require("@serialport/parser-readline");
// serial port object
var sp = new serialport(portName, {
  baudRate: 9600, // 9600 38400
  dataBits: 8,
  parity: "none",
  stopBits: 1,
  flowControl: false,
  parser: new Readline("\r\n"),
});

const parser = sp.pipe(new Readline({ delimiter: "\r\n" }));

// Read the port data
sp.on("open", () => {
  console.log("serial port open");
});
```



A5.7.8 DHT22 + CdS + Node.js

[2.3] NodeJS code: [cds_dht22_node.js](#) (Complete your parser code)

```
var dStr = "";
var readData = ""; // Date string
var temp = "";
var humi = "";
var lux = "";
var mdata = []; // this array will store the data
var firstcommaidx = 0;
```

```
parser.on("data", (data) => {
```

Complete your parser code!!

```
    readData = "";
    dStr = getDateString();
    mdata[0] = dStr; // Date
    mdata[1] = temp; // temperature data
    mdata[2] = humi; // humidity data
    mdata[3] = lux; // luminosity data
    console.log("AAnn," + mdata);
    io.sockets.emit("message", mdata); // send data to all clients
} else {
    // error
    console.log(readData);
}
```



A5.7.10 DHT22 + CdS + Node.js

[3] Result: Parsed streaming data from dht22 & CdS ([Run in Terminal](#))

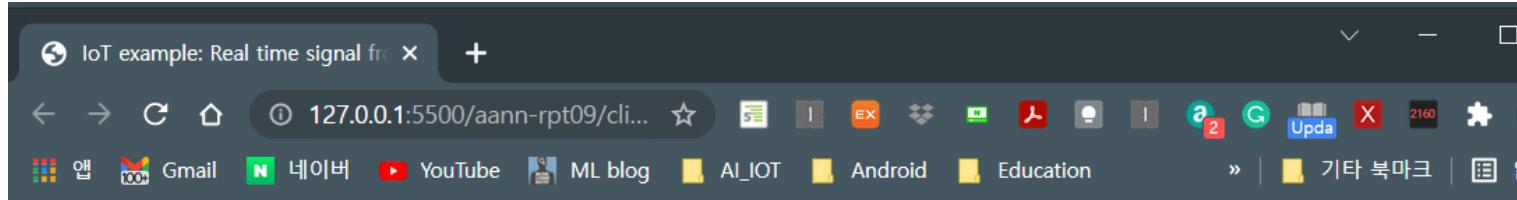
```
COM3  
=====  
21.0,24.7,205  
21.0,24.7,207  
21.0,24.7,205  
21.0,24.7,152  
21.0,24.7,167  
20.9,24.6,166  
20.9,24.6,204  
21.0,24.8,204  
21.0,24.8,152  
21.0,24.8,173  
21.0,24.8,191  
21.0,24.8,203  
21.0,24.8,207  
21.0,24.9,204  
21.0,24.9,204  
  
 자동 스크롤  타임스탬프
```



```
문제    출력    디버그 콘솔    터미널    JUPYTER    node  
=====  
AAnn,2021-10-27 11:53:01.872,23.4,42.6,286  
AAnn,2021-10-27 11:53:02.872,23.4,42.6,286  
AAnn,2021-10-27 11:53:04.150,23.4,42.6,286  
AAnn,2021-10-27 11:53:05.154,23.4,42.6,286  
AAnn,2021-10-27 11:53:06.428,23.4,42.6,286  
AAnn,2021-10-27 11:53:07.431,23.4,42.6,286  
AAnn,2021-10-27 11:53:08.709,23.4,42.6,286  
AAnn,2021-10-27 11:53:09.713,23.4,42.6,286  
AAnn,2021-10-27 11:53:10.987,23.4,42.6,286  
AAnn,2021-10-27 11:53:11.990,23.4,42.6,286  
AAnn,2021-10-27 11:53:13.269,23.4,42.6,284  
AAnn,2021-10-27 11:53:14.268,23.4,42.6,284  
AAnn,2021-10-27 11:53:15.546,23.4,42.6,286  
AAnn,2021-10-27 11:53:16.550,23.4,42.6,284  
AAnn,2021-10-27 11:53:17.824,23.4,42.6,286  
AAnn,2021-10-27 11:53:18.827,23.4,42.6,286  
■
```

Save as
AAnn_cds_dht22_data.png

Arduino data on network socket



IoT Signal from Arduino Weather Station

Real-time Signals

on Time: 2021-10-27 11:54:48.997

Signals (온도,습도,조도) : 23.4,42.6,286

Save as [AAnn_signals_cds_dht22.html](#)

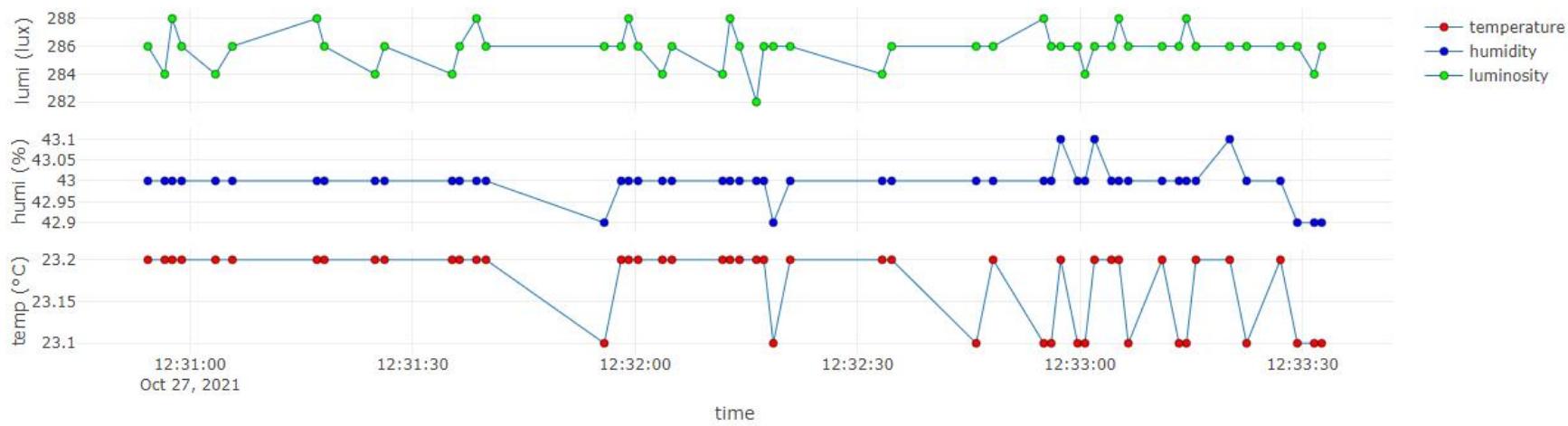
**Real-time monitoring of signals from Arduino
CdS + DHT22 circuit**

WEB client : client_cds_dht22.html

Real-time Weather Station from sensors



on Time: 2021-10-27 12:33:32.600





A5.8.1 DHT22 + CdS + Node.js

[4.1] WEB client: client_cds_dht22.html

```
client_CdS_DHT22.html •
1 <!DOCTYPE html>
2 <head>
3   <meta charset="utf-8">
4   <title>plotly.js Project: Real time signals from multiple sensors</title>
5   <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
6   <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/
socket.io.js"></script>
7
8   <script src="gauge.min.js"></script>
9
10  <style>body{padding:0;margin:30;background:#fff}</style>
11 </head>
12
13 <body>  <!-- style="width:100%;height:100%" -->
14  <!-- Plotly chart will be drawn inside this DIV -->
15  <h1 align="center">Real-time Weather Station from sensors</h1>
16  <!-- 1st gauge -->
17  <div align="center">
18    <canvas id="gauge1"> </canvas>
19    <!-- 2nd gauge -->
20    <canvas id="gauge2"> </canvas>
21    <!-- 3rd gauge -->
22    <canvas id="gauge3"> </canvas>
23  </div>
24  <!-- <div id="console"> </div> -->
25  <h3 align="center"> on Time: <span id="time"> </span> </h3>
26  <div id="myDiv"></div>
27  <hr>
```



A5.8.2 DHT22 + CdS + Node.js

[4.2] WEB client: client_cds_dht22.html

```
29 <script>
30 /* JAVASCRIPT CODE GOES HERE */
31 var streamPlot = document.getElementById('myDiv');
32 var ctime = document.getElementById('time');
33 var tArray = [], // time of data arrival
34     y1Track = [], // value of sensor 1 : temperature
35     y2Track = [], // value of sensor 2 : humidity
36     y3Track = [], // value of sensor 3 : Luminosity
37     numPts = 50, // number of data points in x-axis
38     dtfa = [], // 1 x 4 array : [date, data1, data2, data3] from sensors
39     preX = -1,
40     preY = -1,
41     preZ = -1,
42     initFlag = true;
```

Check points: tArray

xTrack → y1Track, yTrack → y2Track

& add y3Track & preZ



A5.8.3 DHT22 + CdS + Node.js

[4.3] WEB client: client_cds_dht22.html

```
var socket = io.connect('http://localhost:3000'); // port = 3000
socket.on('connect', function () {
  socket.on('message', function (msg) {
    // initial plot
    if(msg[0]!='' && initFlag){
      dtda[0]=msg[0];
      dtda[1]=parseFloat(msg[1]); // temperature  
dtda[2]=parseFloat(msg[2]); // Humidity  
dtda[3]=parseInt(msg[3]); // Luminosity
      init();
      initFlag=false;
    }

    dtda[0]=msg[0];
    dtda[1] = parseFloat(msg[1]);  
dtda[2] = parseFloat(msg[2]);  
dtda[3] = parseInt(msg[3]);
  }
})
```

**Update
to include three signals:
temp, humi, lux**



A5.8.4 DHT22 + CdS + Node.js

[4.4] WEB client: client_cds_dht22.html

```
// Only when any of data is different from the previous one,  
// the screen is redrawed.  
if (dtda[1] != preX || dtda[2] != preY || dtda[3] != preZ) { // any change?  
    preX = dtda[1];  
    preY = dtda[2];  
    preZ = dtda[3];  
  
    // when new data is coming, keep on streaming  
    ctime.innerHTML = dtda[0];  
    gauge_temp.setValue(dtda[1]); // temp gauge  
    gauge_humi.setValue(dtda[2]); // humi gauge  
    gauge_lux.setValue(dtda[3]); // lux gauge  
    //nextPt();  
    tArray = tArray.concat(dtda[0]);  
    tArray.splice(0, 1); // remove the oldest data  
    y1Track = y1Track.concat(dtda[1]);  
    y1Track.splice(0, 1); // remove the oldest data  
    y2Track = y2Track.concat(dtda[2]);  
    y2Track.splice(0, 1);  
    y3Track = y3Track.concat(dtda[3]);  
    y3Track.splice(0, 1);  
  
    var update = {  
        x: [tArray, tArray, tArray],  
        y: [y1Track, y2Track, y3Track]  
    }  
  
    Plotly.update(streamPlot, update);  
}
```

Update
to include three signals:
temp, humi, lux



A5.8.5 DHT22 + CdS + Node.js

[4.5] WEB client: client_dht22_ldr.html → init()

```
function init() { // initial screen ()  
    // starting point : first data (temp, lux)  
    for ( i = 0; i < numPts; i++) {  
        tArray.push(dtda[0]); // date  
        y1Track.push(dtda[1]); // sensor 1 (temp)  
        y2Track.push(dtda[2]); // sensor 2 (humi)  
        y3Track.push(dtda[3]); // sensor 3 (lux)  
    }  
  
    Plotly.plot(streamPlot, data, layout);  
}
```

**Update
to include three signals:
temp, humi, lux**



A5.8.6 DHT22 + CdS + Node.js

[4.6] WEB client: client_cds_dht22.html - data

```
// data
var data = [
  {
    x : tArray,
    y : y1Track,
    name : 'temperature',
    mode: "markers+lines", // "
    line: {
      color: "#1f77b4",
      width: 1
    },
    marker: {
      color: "rgb(255, 0, 0)",
      size: 6,
      line: {
        color: "black",
        width: 0.5
      }
    }
  },
  {
    x : tArray,
    y : y2Track,
    name : 'humidity',
    xaxis: 'x2',
    yaxis : 'y2',
    mode: "markers+lines", // "
    line: {
      color: "#1f77b4",
      width: 1
    },
    marker: {
      color: "rgb(0, 0, 255)",
      size: 6,
      line: {
        color: "black",
        width: 0.5
      }
    }
  },
  {
    x : tArray,
    y : y3Track,
    name : 'luminosity',
    xaxis: 'x3',
    yaxis : 'y3',
    mode: "markers+lines", // "
    line: {
      color: "#1f77b4",
      width: 1
    },
    marker: {
      color: "rgb(0, 255, 0)",
      size: 6,
      line: {
        color: "black",
        width: 0.5
      }
    }
  }
];
```

**Update data
to include three signals:
temp, humi, lux**



A5.8.7 DHT22 + CdS + Node.js

[4.7] WEB client: client_cds_dht22.html - layout

```
var layout = {  
   .xaxis : {  
        title : 'time',  
        domain : [0, 1]  
    },  
   .yaxis : {  
        title : 'temp (°C)',  
        domain : [0, 0.3],  
        range : [-30, 50]  
    },  
   .xaxis2 : {  
        title : '',  
        domain : [0, 1],  
        position : 0.35  
    },  
   .yaxis2 : {  
        title : 'humi (%)',  
        domain : [0.35, 0.65],  
        range : [0, 100]  
    },  
   .xaxis3 : {  
        title : '',  
        domain : [0, 1],  
        position : 0.7  
    },  
   .yaxis3 : {  
        title : 'lumi (lux)',  
        domain : [0.7, 1],  
        range : [0, 500]  
    }  
}
```

1. Update layout
to include three signals:
temp, humi, lux.

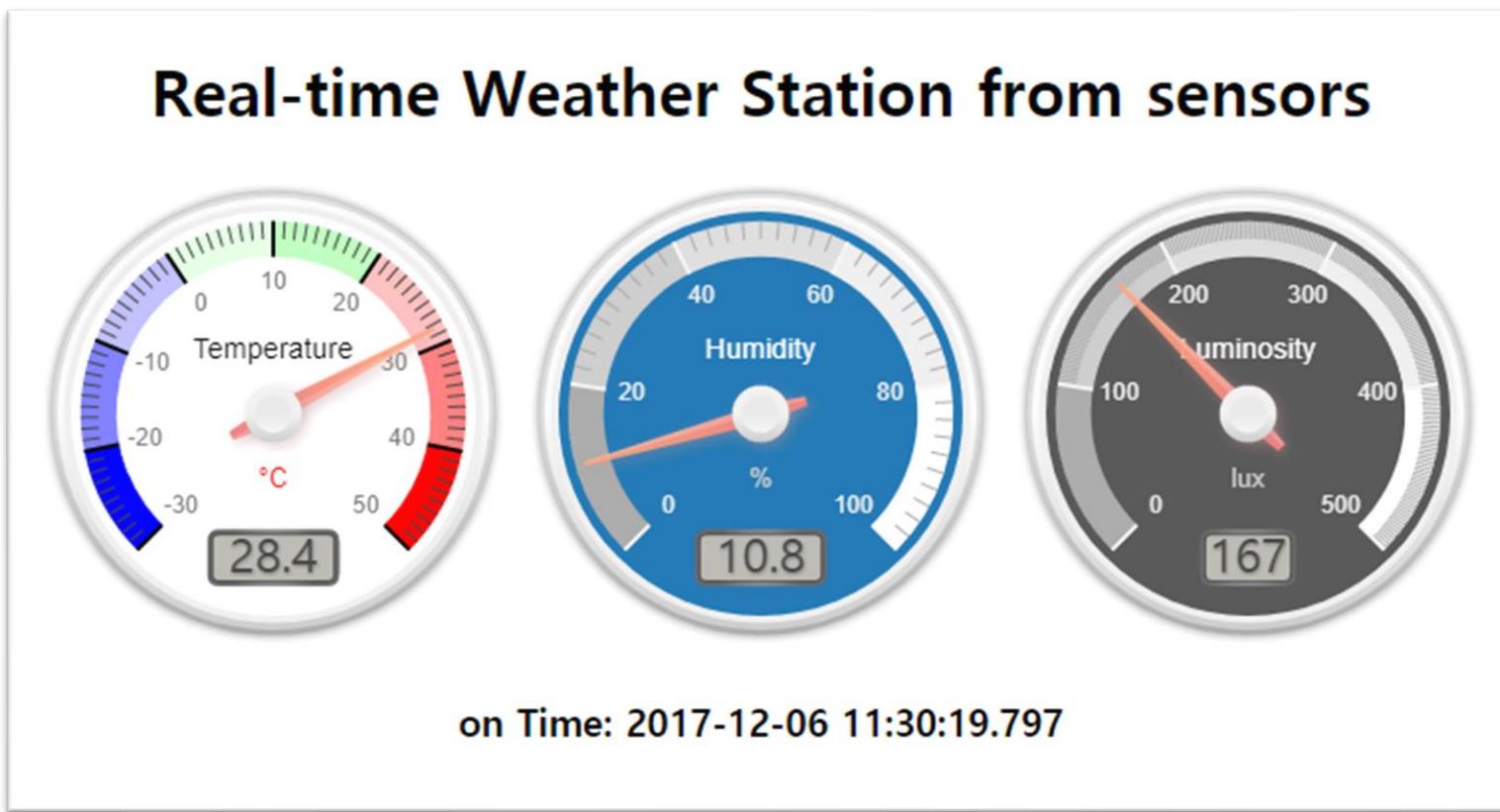
2. Check the domain & position.

Save the complete code as
AAnn_cds_dht22.html



A5.8.8 DHT22 + CdS + Node.js

[4.8] WEB client: client_dht22_ldr.html – [Design your gauges](#)



Save the complete

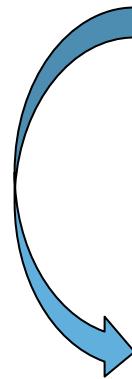
code as

[**AAnn_cds_dht22.html**](#)



A5.8.9 DHT22 + CdS + Node.js

[4.9] WEB client: Design layout (show date at lower axis)



[Hint]

[Plot.ly](#)

WEB client : client_cds_dht22.html

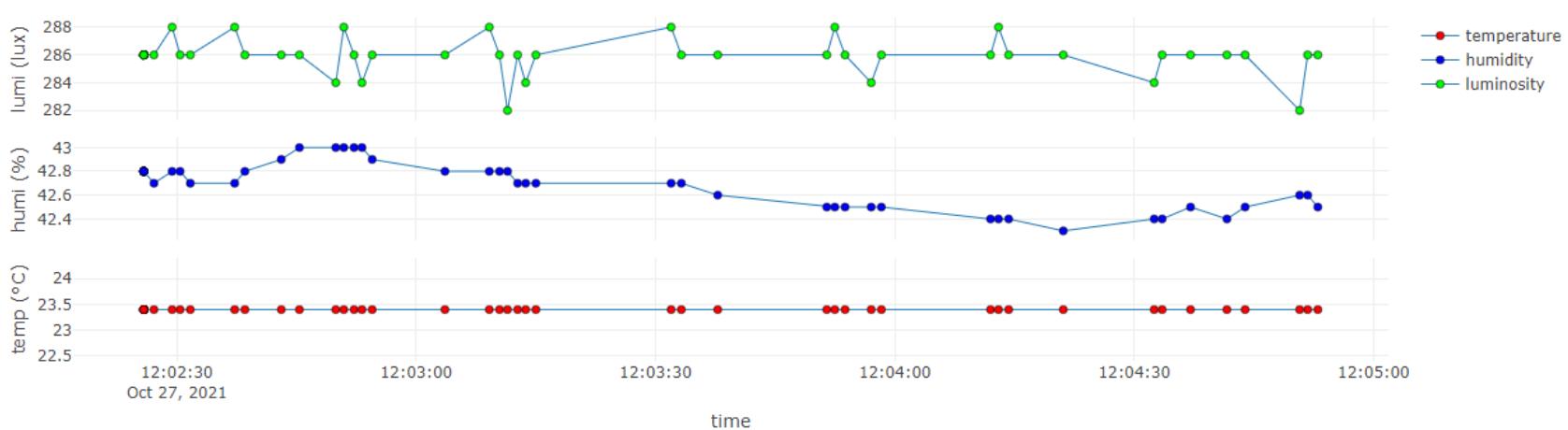
Real-time Weather Station from sensors

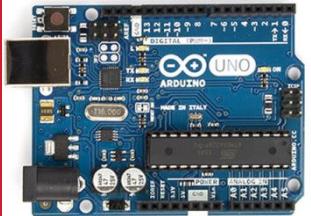


on Time: 2021-10-27 12:04:53.016

Save as

AAnn_cds_dht22.png





[Practice]

- ◆ [wk09: cds_dht22]
 - RT Data Visualization with node.js
 - Multiple data and Usage of gauge.js
 - Complete your real-time WEB charts
 - Upload folder: aann-rpt09
 - Use repo “aann” in github

wk09 : Practice : aann-rpt09

◆ [Target of this week]

- Complete your works : **cds_dht22 project.**
- Save your outcomes and upload outputs in github

제출폴더명 : **aann-rpt09**

- 제출할 파일들

- ① **AAnn_cds_dht22_data.png**
- ② **AAnn_signals_cds_dht22.html**
- ③ **AAnn_cds_dht22.html**
- ④ **AAnn_cds_dht22.png**
- ⑤ **All *.ino**
- ⑥ **All *.js**
- ⑦ **All *.html**



A5. Introduction to IoT service

System (Arduino, sDevice, ...)



Data (signal, image, sns, ...)



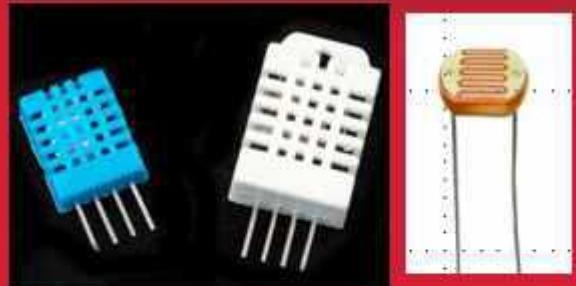
Visualization & monitoring



Data storing & mining



Service



[Goal]

Arduino + Node.js

+ plotly.js

+ MongoDB

→ Data storaging

& visualization



A5.9 MongoDB



mongoDB®



A5.9 MongoDB



The most popular database for [...x](#)

[mongodb.com](#)

Event Interested in speaking at MongoDB World 2022? Become a speaker >>

MongoDB.

NEW Introducing native support for time series data – [Learn more >](#)

Build faster. Build smarter.

Get your ideas to market faster with an application data platform built on the leading modern database. With built-in search, analytics, and edge support, accessible with a common query API and built on the data model developers love, MongoDB makes working with data easy – for any use case.

[Start Free](#) [Questions? Talk to us >](#)



A5.9 MongoDB



MongoDB는 C++로 작성된 오픈소스 문서지향(Document-Oriented) 적 Cross-platform 데이터베이스이며, 뛰어난 확장성과 성능을 자랑합니다. 또한, 현존하는 NoSQL 데이터베이스 중 인지도 1위를 유지하고 있습니다.

NoSQL?

흔히 NoSQL이라고 해서 아, SQL이 없는 데이터베이스구나!라고 생각 할 수도 있겠지만, 진짜 의미는 Not Only SQL입니다. 기존의 RDBMS의 한계를 극복하기 위해 만들어진 새로운 형태의 데이터저장소입니다. 관계형 DB가 아니므로, RDMS처럼 고정된 스키마 및 JOIN이 존재하지 않습니다.

Document?

Document Oriented 데이터베이스라는데.. 여기서 말하는 Document가 뭘까요? 문서? 이게 그냥 ‘문서’로 번역해버리면 조금은 애매합니다. 문서라고 하면 보통 워드/엑셀에 사용되는 그런 문서가 떠오르는데요, 그것과는 다릅니다. Document는 RDMS의 record와 비슷한 개념인데요, 이의 데이터 구조는 한개이상의 key-value pair으로 이루어져있습니다. MongoDB 샘플 Document를 확인해 볼까요?

```
{ "_id": ObjectId("5099803df3f4948bd2f98391"),
  "username": "velopert",
  "name": { first: "M.J.", last: "Kim" } }
```



A5.9 MongoDB



여기서 **_id, username, name** 은 **key** 이고 그 오른쪽에 있는 값들은 **value** 입니다.

_id 는 12bytes의 hexadecimal 값으로서, 각 **document**의
유일함(uniqueness)을 제공합니다.
이 값의 첫 4bytes 는 현재 **timestamp**, 다음 3bytes는 **machine id**, 다음
2bytes는 **MongoDB** 서버의 프로세스 **id**, 마지막 3bytes는 순차번호입니다 추가될때마다
값이 높아진다는 거지요.

Document는 동적(dynamic)의 **schema**를 갖고 있습니다. 같은 **Collection** 안에
있는 **Document**끼리 다른 **schema**를 갖고 있을 수 있는데요, 쉽게 말하면 서로 다른
데이터 (즉 다른 **key**) 들을 가지고 있을 수 있습니다.

Collection?

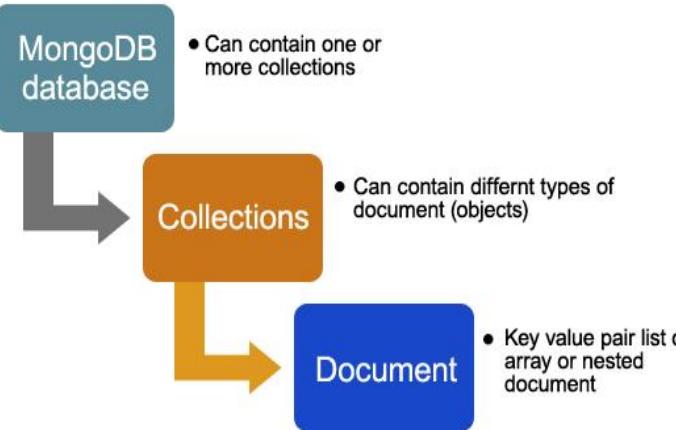
Collection은 **MongoDB Document**의 그룹입니다. **Document**들이
Collection 내부에 위치하고 있습니다. **RDMS**의 **table**과 비슷한 개념입니다만 **RDMS**와
달리 **schema**를 따로 가지고 있지 않습니다. **Document** 부분설명에 나와있듯이 각
Document들이 동적인 **schema**를 가지고 있으니까요

Database?

Database는 **Collection**들의 물리적인 컨테이너입니다. 각 **Database**는 파일시스템에
여러파일들로 저장됩니다.



A5.9 MongoDB



<https://cdn.educba.com/academy/wp-content/uploads/2019/04/MongoDB-chart2.jpg>

<https://i.imgur.com/Att4uVC.png>



A5.9 MongoDB: download



MongoDB Community Download

mongodb.com/try/download/community

Atlas
MongoDB as a service

On-premises
MongoDB locally

Tools
Boost productivity

Mobile & Edge
Realm Datastore

MongoDB Enterprise Server

MongoDB Community Server

The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and real-time aggregations to provide powerful ways to access and analyze your data.

The database is also offered as a fully-managed service with [MongoDB Atlas](#). Get access to advanced functionality such as auto-scaling, serverless instances (in preview), full-text search, and data distribution across regions

Available Downloads

Version: 4.2.17

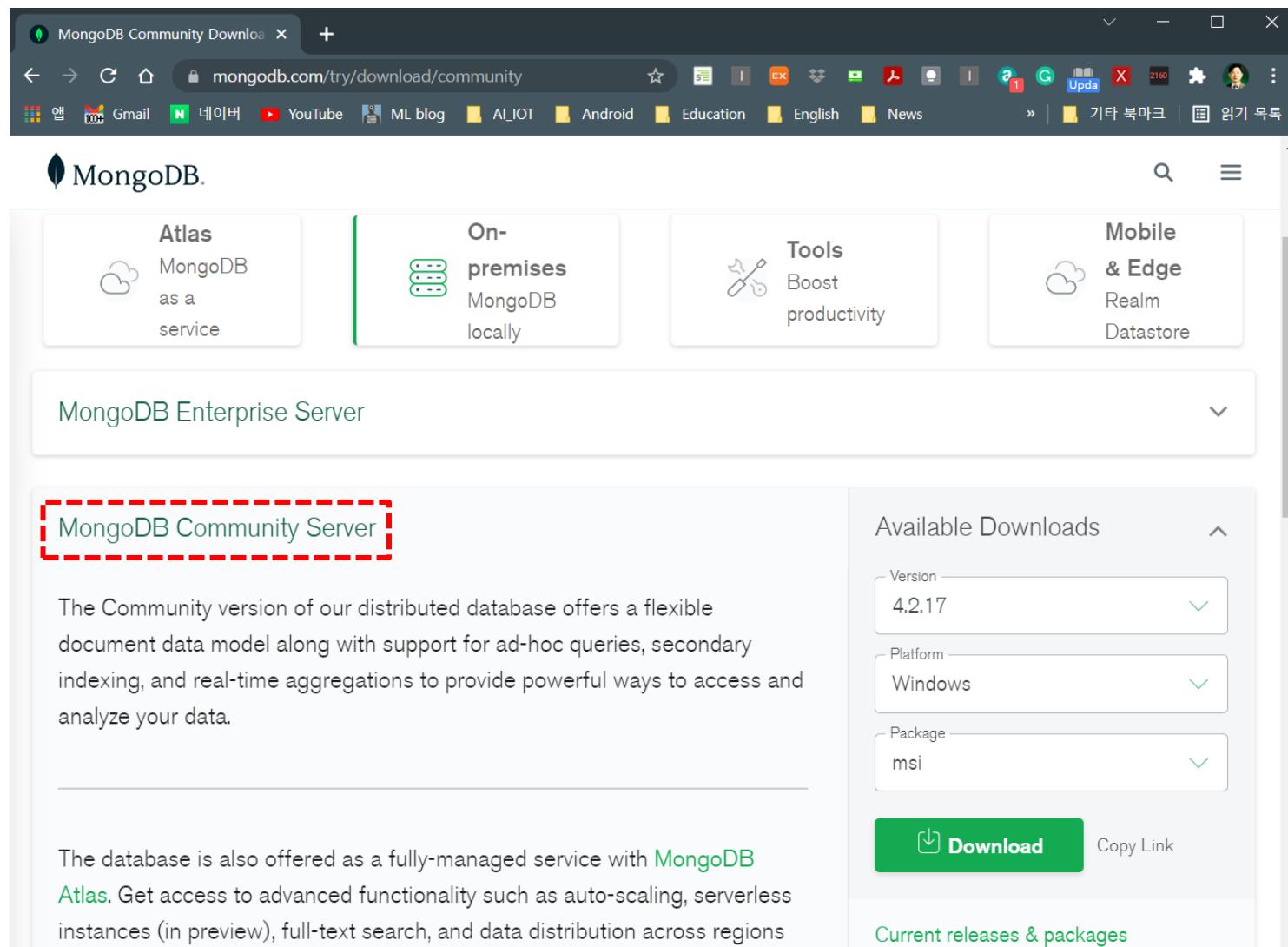
Platform: Windows

Package: msi

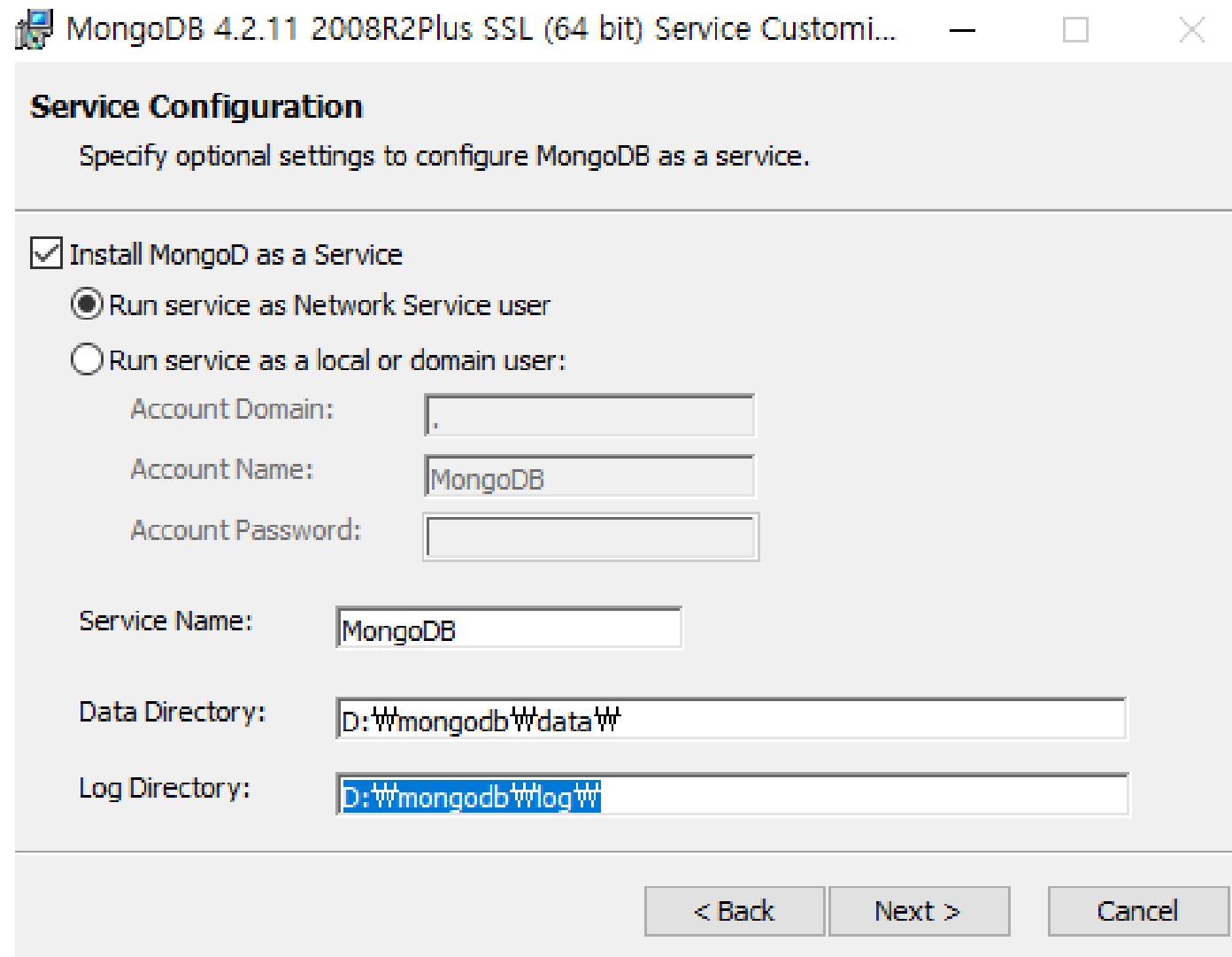
Download

Copy Link

Current releases & packages



<https://www.mongodb.com/try/download/community>





A5.9.1 MongoDB install - 1



원도우10: 설정 > 시스템 > 정보

[중요] 시스템 환경변수 : PATH 에 경로 추가

C:\Program Files\MongoDB\Server\4.2\bin



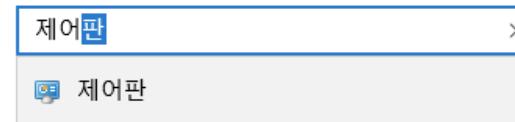
A5.9.1 MongoDB install – 2



설정



Windows 설정



시스템
디스플레이, 알림, 전원



장치
Bluetooth, 프린터, 마우스



전화
Android, iPhone 연결



네트워크 및 인터넷
Wi-Fi, 비행기 모드, VPN



개인 설정
배경, 잠금 화면, 색



앱
설치 제거, 기본값, 옵션 기능



계정
내 계정, 메일, 등기화, 회사, 가족



시간 및 언어
음성, 지역, 날짜



게임
게임 바, DVR, 브로드캐스팅, 게임 모드



접근성
내레이터, 듣보기, 고대비



개인 정보
위치, 카메라



업데이트 및 보안
Windows 업데이트, 복구, 백업



A5.9.1 MongoDB install – 3

제어판 홍

제어판에 대한 기본 정보 보기

Windows 버전

Windows 10 Home
© 2017 Microsoft Corporation. All rights reserved.

시스템

프로세서: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz 3.40 GHz
설치된 메모리(RAM): 32.0GB
시스템 종류: 64비트 운영 체제, x64 기반 프로세서
펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.

컴퓨터 이름, 도메인 및 작업 그룹 설정

컴퓨터 이름: yish-HCIt
전체 컴퓨터 이름: yish-HCIt
컴퓨터 설명:
작업 그룹: WORKGROUP

설정 변경

Windows 정품 인증

Windows 정품 인증을 받았습니다. Microsoft 소프트웨어 사용 조건 읽기

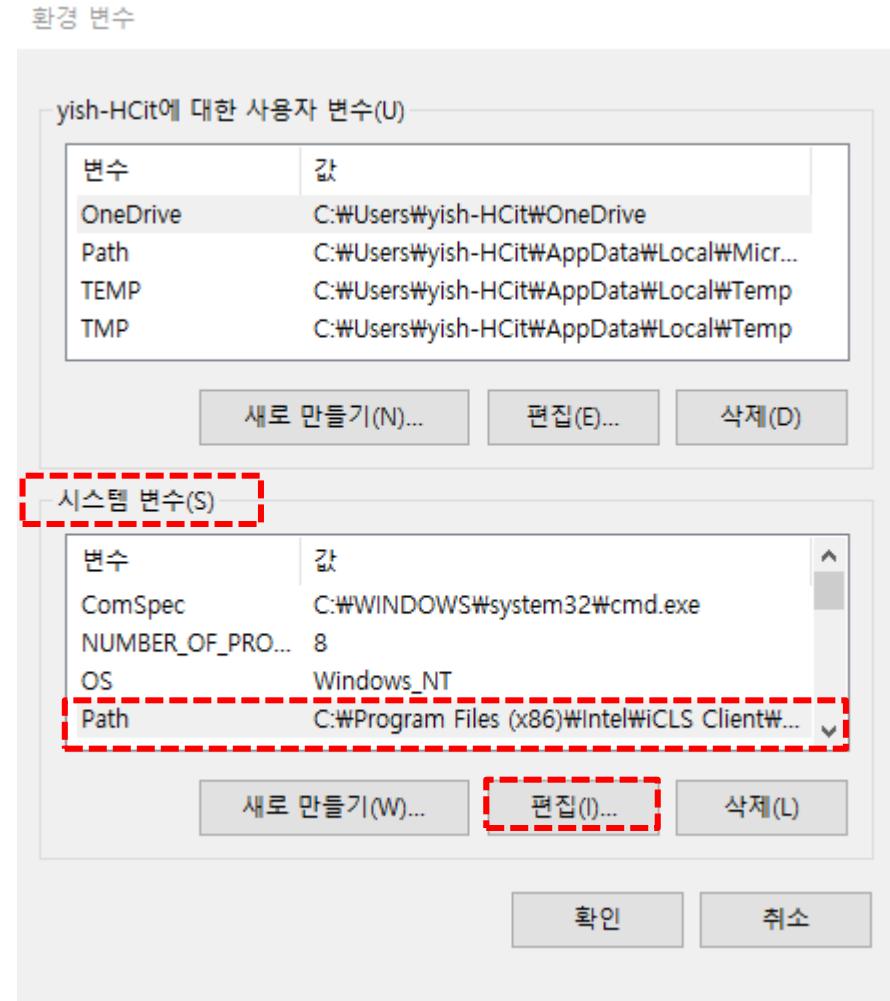
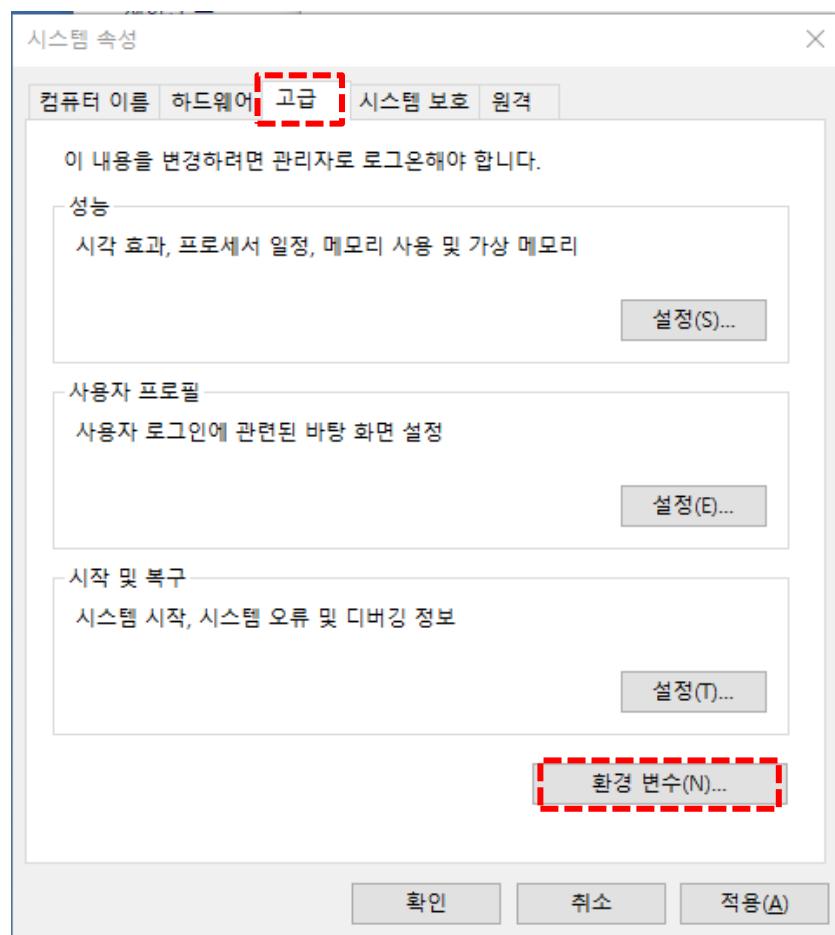
제품 ID: 00325-96080-39821-AAOEM

제품 키 변경



A5.9.1 MongoDB install – 4

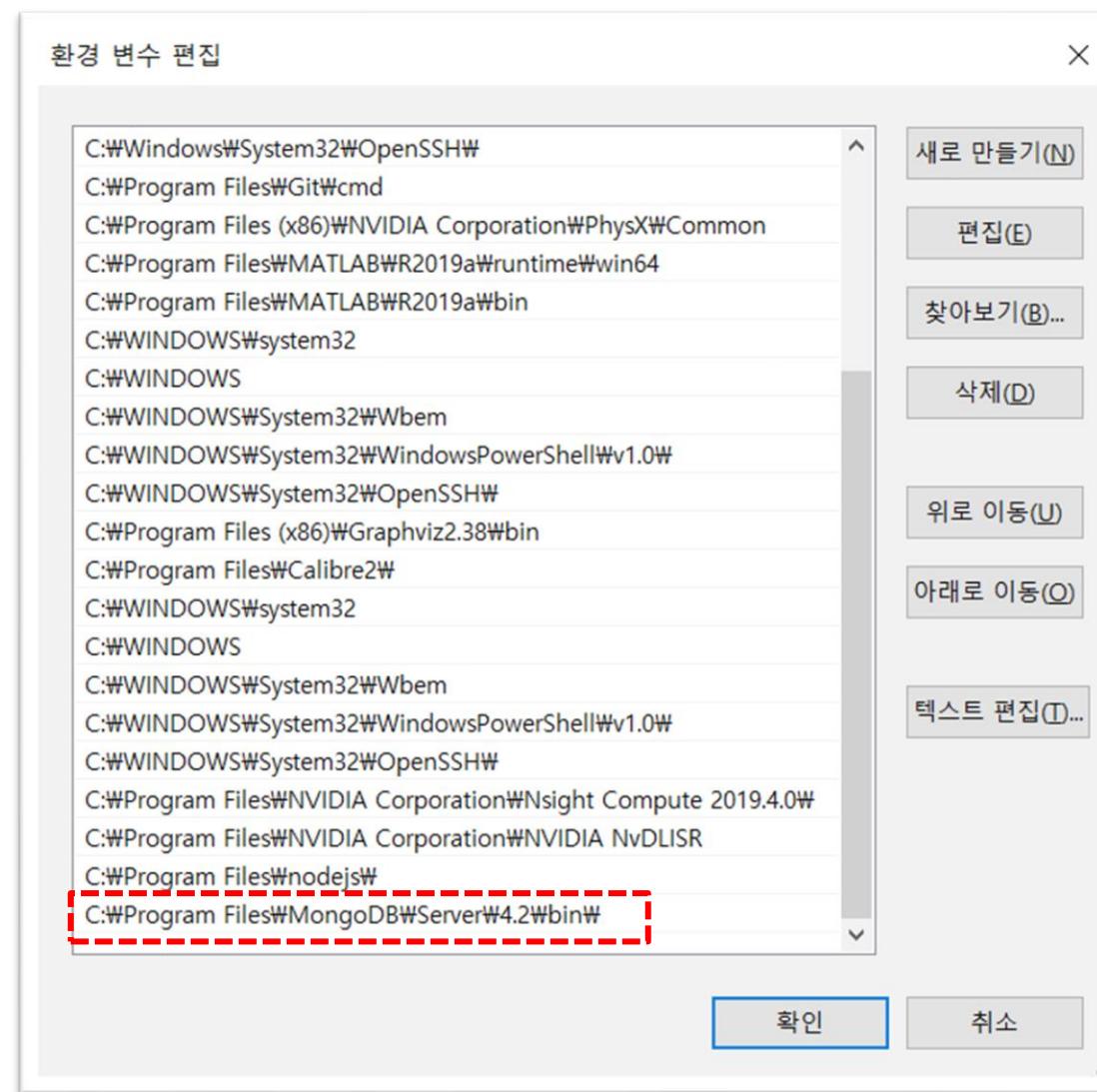
환경 변수 설정





A5.9.1 MongoDB install – 5

환경 변수 추가





A5.9.2 MongoDB shell - 1

1. Mongo shell 실행

> mongo (or mongosh)

```
명령 프롬프트 - mongo
D:\W\mongodb>mongo
MongoDB shell version v4.2.11-rc1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("862dc45b-18a7-4f26-8006-b28d58799e64") }
MongoDB server version: 4.2.11-rc1
Server has startup warnings:
2020-11-18T09:43:57.884+0900 I CONTROL [initandlisten]
2020-11-18T09:43:57.884+0900 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-11-18T09:43:57.884+0900 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-11-18T09:43:57.885+0900 I CONTROL [initandlisten]
---
```

If, Connect failed... → DB 데몬 실행



A5.9.2 MongoDB shell - 2

2. MongoDB 저장소 만들기 → D drive

- **md mongodb**
- **cd mongodb**
- **dir**
- **md data**
- **md log**
- **dir**

```
D:\mongodb>dir
D 드라이브의 볼륨: DATA
볼륨 일련 번호: 82D1-4852

D:\mongodb 디렉터리

2020-11-18 오전 09:39 <DIR> .
2020-11-18 오전 09:39 <DIR> ..
2020-11-18 오전 10:11 <DIR> data
2020-11-18 오전 09:39 <DIR> log
                           0개 파일
                           4개 디렉터리  2,332,408,369,152 바이트 남음

D:\mongodb>
```

The screenshot shows a Windows Command Prompt window titled "명령 프롬프트". The command `dir` is run from the directory `D:\mongodb`. The output lists four subdirectories: `.` (current directory), `..` (parent directory), `data`, and `log`. A red dashed box highlights the `data` and `log` directories.

사용 PC 환경에 맞게 실행 (특히, 경로 지정)

실습실 환경에 맞춰서 D:에 MongoDB data 폴더 지정



A5.9.2 MongoDB shell - 3

3. Run MongoDB by using `mongod.exe`

➤ `mongod --dbpath d:\mongodb\data`

명령 프롬프트 - mongod -dbpath d:\mongodb\data

D:\mongodb>md data

```
D:\mongodb>mongod --dbpath d:\mongodb\data
2018-01-22T19:27:32.931-0700 I CONTROL [initandlisten] MongoDB starting : pid=18820 port=27017
dbpath=d:\mongodb\data 64-bit host=yish-HCit
2018-01-22T19:27:32.931-0700 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2
008 R2
2018-01-22T19:27:32.932-0700 I CONTROL [initandlisten] db version v3.6.2
2018-01-23T11:27:33.699+0900 I COMMAND [initandlisten] setting featureCompatibilityVersion to
3.6
2018-01-23T11:27:33.706+0900 I STORAGE [initandlisten] createCollection: local.startup_log wit
h generated UUID: 06b3b7cb-62fe-4be5-a929-2a7478650a9b
2018-01-23T11:27:34.211+0900 I FTDC [initandlisten] Initializing full-time diagnostic data
capture with directory 'd:/mongodb/data/diagnostic.data'
2018-01-23T11:27:34.215+0900 I NETWORK [initandlisten] waiting for connections on port 27017
```

사용 PC 환경에 맞게 실행 (특히, 경로 지정)



A5.9.2 MongoDB shell - 4

4. Run mongo shell : [mongo.exe](#)

[use new cmd or Terminal]

➤ **mongo**

Run in Terminal

mongo

D:\aann\aann-rpt10>**mongo**
MongoDB shell version v4.2.11-rc1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("ef9fb07b-cc3b-49ed-8a49-2df43ec1200a") }
MongoDB server version: 4.2.11-rc1
Server has startup warnings:
2021-10-24T18:59:43.599+0900 I CONTROL [initandlisten]
2021-10-24T18:59:43.599+0900 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2021-10-24T18:59:43.599+0900 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2021-10-24T18:59:43.599+0900 I CONTROL [initandlisten]

Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()



A5.9.2 MongoDB shell - 5

5. mongo shell :

Run new cmd

mongo

show dbs

use local

show
collections

help

exit

```
명령 프롬프트 - mongo
>[show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
>use local
switched to db local
>show collections
startup_log
> help
    db.help()
    db.mycoll.help()
    sh.help()
    rs.help()
    help admin
    help connect
    help keys
    help misc
    help mr

    show dbs
    show collections
    show users
    show profile

    lms
        show logs
        show log [name]
    is default
        use <db_name>
        db.foo.find()
        db.foo.find( { a : 1 } )
        it

    rate
        DBQuery.shellBatchSize = x
        exit
> -
```

help on db methods
help on collection methods
sharding helpers
replica set helpers
administrative help
connecting to a db help
key shortcuts
misc things to know
mapreduce

show database names
show collections in current database
show users in current database
show most recent system.profile entries with time >=

show the accessible logger names
prints out the last segment of log in memory, 'global'

set current database
list objects in collection foo
list objects in foo where a == 1
result of the last line evaluated; use to further iterate

set default number of items to display on shell
quit the mongo shell



A5.9.3 MongoDB shell coding

1. make my own db (aann) & insert one record (document)

`use aa00`

`show collections`

insert record with new collection “user”

`db.user.insert({first:"Redwoods", last:"Yi"})`

`show collections`

→ “user”

`show dbs`

`db.user.find()`

```
C:\ 명령 프롬프트 - mongo
> use aa00
switched to db aa00
> show collections
> db.user.insert({first:"Redwoods", last:"Yi"})
WriteResult({ "nInserted" : 1 })
> show collections
user
> show dbs
aa00    0.000GB
admin   0.000GB
config  0.000GB
local   0.000GB
> -
```



A5.9.3 MongoDB shell coding

2. insert more records with different schema & show records

insert record2

insert record3

show collections

db.user.find()

db.user.find().pretty()

```
명령 프롬프트 - mongo
> db.user.insert({first:"Chaos", last:"Kim"})
WriteResult({ "nInserted" : 1 })
> db.user.insert({first:"Gildong", last:"Hong"})
WriteResult({ "nInserted" : 1 })
> show collections
USER
> db.user.find()
[{"_id": ObjectId("5a66b44b9f0d55608f5f7582"), "first": "Redwoods", "last": "Yi"}, {"_id": ObjectId("5a66b5759f0d55608f5f7583"), "first": "Chaos", "last": "Kim"}, {"_id": ObjectId("5a66b5869f0d55608f5f7584"), "first": "Gildong", "last": "Hong"}]
> db.user.find().pretty()
{
    "_id" : ObjectId("5a66b44b9f0d55608f5f7582"),
    "first" : "Redwoods",
    "last" : "Yi"
}
{
    "_id" : ObjectId("5a66b5759f0d55608f5f7583"),
    "first" : "Chaos",
    "last" : "Kim"
}
{
    "_id" : ObjectId("5a66b5869f0d55608f5f7584"),
    "first" : "Gildong",
    "last" : "Hong"
}
```

_id 는 12bytes의 hexadecimal 값으로서, 각 document의
유일함(uniqueness)을 제공합니다.
이 값의 첫 4bytes는 현재 timestamp, 다음 3bytes는
machine id, 다음 2bytes는 MongoDB 서버의 프로세스 id,
마지막 3bytes는 순차번호입니다.



A5.9.3 MongoDB shell coding

3. insert more records with different schema & show records

insert record4
with firstName key

db.user.find()

db.user.find().pretty()

```
> db.user.insert({firstName:"Fractal", last:"Park"})
writeResult({ "nInserted": 1 })
> db.user.find().pretty()
{
    "_id" : ObjectId("5a66b44b9f0d55608f5f7582"),
    "first" : "Redwoods",
    "last" : "Yi"
}
{
    "_id" : ObjectId("5a66b5759f0d55608f5f7583"),
    "first" : "Chaos",
    "last" : "Kim"
}
{
    "_id" : ObjectId("5a66b5869f0d55608f5f7584"),
    "first" : "Gildong",
    "last" : "Hong"
}
{
    "_id" : ObjectId("5a66b6439f0d55608f5f7585"),
    "firstName" : "Fractal",
    "last" : "Park"
}>
```

**Dynamic
schema**

동적스키마

Note that there are two kinds of schemas in JSON.
Save as

[AAnn_mongo_schemas.png](#)



A5.9.3 MongoDB shell coding

4. remove one of records (or documents)

remove record3

db.user.find().pretty()

```
C:\ 명령 프롬프트 - mongo
> db.user.remove({last:"Kim"})
WriteResult({ "nRemoved" : 1 })
> db.user.find().pretty()
{
    "_id" : ObjectId("5a66b44b9f0d55608f5f7582"),
    "first" : "Redwoods",
    "last" : "Yi"
}
{
    "_id" : ObjectId("5a66b5869f0d55608f5f7584"),
    "first" : "Gildong",
    "last" : "Hong"
}
{
    "_id" : ObjectId("5a66b6439f0d55608f5f7585"),
    "firstName" : "Fractal",
    "last" : "Park"
}
>
```



A5.9.3 MongoDB shell coding

5. update a record

```
db.user.update({last:"Hong"},{$set:{first:"GilDong", age:21}})
```

update record2

db.user.find().pretty()

```
C:\ 명령 프롬프트 - mongo
> db.user.update({last:"Hong"},{$set:{first:"GilDong", age:21}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.user.find().pretty()
{
    "_id" : ObjectId("5a66b44b9f0d55608f5f7582"),
    "first" : "Redwoods",
    "last" : "Yi"
}
{
    "_id" : ObjectId("5a66b5869f0d55608f5f7584"),
    "first" : "GilDong",
    "last" : "Hong",
    "age" : 21
}
{
    "_id" : ObjectId("5a66b6439f0d55608f5f7585"),
    "firstName" : "Fractal",
    "last" : "Park"
}
> -
```

Note that it is possible to change schema.
Save as

AAnn_mongo_update.png



A5.9.3 MongoDB shell coding

6. Delete(or remove) DB

use dbName

db.dropDatabase()

show dbs

```
명령 프롬프트 - mongo
> use aa00
switched to db aa00
> show collections
user
> db.user.find()
{ "_id" : ObjectId("5a66b44b9f0d55608f5f7582") , "first" : "Redwoods" , "last" : "Yi" }
{ "_id" : ObjectId("5a66b5869f0d55608f5f7584") , "first" : "Gi IDong" , "last" : "Hong" , "age"
" : 21 }
{ "_id" : ObjectId("5a66b6439f0d55608f5f7585") , "firstName" : "Fractal" , "last" : "Park" }

>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
> db.dropDatabase()
{ "dropped" : "aa00" , "ok" : 1 }
> show dbs
```

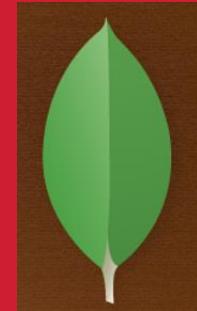


Node.js



+

MongoDB





A5.9.4 MongoDB + Node.js : mongoose

Fork me on GitHub

mongoose

elegant **mongodb** object modeling for **node.js**

[read the docs](#)

[discover plugins](#)



Version 6.0.12



Let's face it, **writing MongoDB validation, casting and business logic boilerplate is a drag**. That's why we wrote Mongoose.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/test');

const Cat = mongoose.model('Cat', { name: String });
```

<http://mongoosejs.com/>



A5.9.4 MongoDB + Node.js : mongoose

Features Business Explore Marketplace Pricing This repository

Automattic / mongoose

Code Issues 250 Pull requests 2 Projects 0 Wiki Insights

MongoDB object modeling designed to work in an asynchronous environment. <http://mon>

8,362 commits 14 branches 420 releases

Branch: master ▾ New pull request

vkarlov15 Merge branch '4.x'

.github fix typo

benchmarks style: remove unused `BlogPost` variable

docs Merge branch '4.x'

<https://github.com/Automattic/mongoose>



A5.9.4 MongoDB + Node.js : mongooseJS

1. Install mongoose in node.js project <http://mongoosejs.com/>

- Go to cds_dht22 project
- npm install --save mongoose (버전 : 6.0.12)

The screenshot shows a terminal window with the following interface elements:

- Top bar: 문제 (Issues), 출력 (Output), 디버그 콘솔 (Debug Console), 터미널 (Terminal) (highlighted in blue), cmd, +, -, ^, X.
- Output area:

```
D:\aann\aann-rpt10\cds_dht22>npm install --save mongoose
npm notice created a lockfile as package-lock.json. You should commit this
file.
npm WARN cds_dht22@1.0.0 No repository field.

+ mongoose@6.0.12
added 24 packages from 60 contributors, removed 10 packages and audited 68
packages in 2.941s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



A5.9.4 MongoDB + Node.js : mongoose

2. node.js project using mongoose (use VSCode)

- [cds_dht22 project](#)
- [New file: dbtest.js](#)
- [Run:
node dbtest](#)

```
./ aann > aann-rpt10 > cds_dht22 > JS dbtest.js > ...
1 // dbtest.js
2 var mongoose = ...
3 mongoose.connect("mongodb://localhost/test", {
4   useNewUrlParser: true,
5   useUnifiedTopology: true,
6 });
7
8 var SensorSchema = new mongoose.Schema({
9   data: String,
10  created: Date,
11});
12
13 // data model
14 var Sensor = mongoose.model("Sensor", SensorSchema);
15
16 var sensor1 = new Sensor({ data: "124", created: new Date() });
17 sensor1.save();
18
19 var sensor2 = new Sensor({ data: "573", created: new Date() });
20 sensor2.save();
21
22 console.log("Sensor data were saved in MongoDB");
```

Sensor data were saved in MongoDB



A5.9.4 MongoDB + Node.js : mongoose

3. node.js project using mongoose (mongo shell)

Mongo shell

> show dbs

> use test

> show collections

> db.sensors.find()
.pretty()

```
C:\ mongo   ●  C:\ cmd  JS dbtest.js  + ▾ □ ...  
> show dbs  
admin 0.000GB  
config 0.000GB  
iot 0.000GB  
local 0.000GB  
test 0.000GB  
test2 0.000GB  
> use test  
switched to db test  
> show collections  
sensors  
> db.sensors.find()  
> db.sensors.find()  
{ "_id" : ObjectId("6180e44fbc986ac8a5297195"), "data" : "124", "created" : ISODate("2021-11-02T07:10:07.689Z"), "__v" : 0 }  
{ "_id" : ObjectId("6180e44fbc986ac8a5297196"), "data" : "573", "created" : ISODate("2021-11-02T07:10:07.692Z"), "__v" : 0 }  
>
```



A5.9.4 MongoDB + Node.js : mongoose

4. dbtest2.js

./ aann > aann-rpt10 > cds_dht22 > JS dbtest2.js > getDateString

```
1 // dbtest2.js
2 var mongoose = require("mongoose");
3 mongoose.connect("mongodb://localhost/test2", {
4   useNewUrlParser: true,
5   useUnifiedTopology: true,
6 });
7 var SensorSchema = new mongoose.Schema({
8   data: String,
9   created: String,
10 });
11 // data model
12 var Sensor = mongoose.model("Sensor", SensorSchema);
13
14 var sensor1 = new Sensor({ data: "124", created: getDateString() })
15 sensor1.save();
16 var sensor2 = new Sensor({ data: "573", created: getDateString() })
17 sensor2.save();
18
19 console.log("[dbtest2.js]: Sensor data were saved in MongoDB");
20 // helper function to get a nicely formatted date string
21 function getDateString() {
22   var time = new Date().getTime();
23   // 32400000 is (GMT+9 Korea, GimHae)
24   // for your timezone just multiply +/-GMT by 3600000
25   var datestr = new Date(time + 32400000)
26     .toISOString()
27     .replace(/\T/, " ")
28     .replace(/\Z/, "");
29   return datestr;
30 }
```



A5.9.4 MongoDB + Node.js : mongoose

5. dbtest2.js (change Schema & check using mongo shell)

Mongo shell

> show dbs

```
> show dbs
admin 0.000GB
config 0.000GB
iot 0.000GB
local 0.000GB
test 0.000GB
test2 0.000GB
```

> use test2

```
> use test2
```

```
switched to db test2
```

> show collections

```
> db.sensors.find().pretty()
```

> db.sensors.find()
.pretty()

```
{
  "_id" : ObjectId("6181d4c4501d6bc94cfa6903"),
  "data" : "124",
  "created" : "2021-11-03 09:16:04.448",
  "__v" : 0
}
{
  "_id" : ObjectId("6181d4c4501d6bc94cfa6904"),
  "data" : "573",
  "created" : "2021-11-03 09:16:04.451",
  "__v" : 0
}
>
```



[Practice]

- ◆ [wk09: mongoDB test]
 - Insert documents to test mongodb
 - Upload folder: aann-rpt09
 - Use repo “aann” in github

wk09 : Practice : aann-rpt09

◆ [Target of this week]

- Complete your works : **mongoDB test.**
- Save your outcomes and upload outputs in github

제출폴더명 : **aann-rpt09** 에 아래 파일을 추가

- 제출할 파일들

- ① **AAnn_mongo_schemas.png**
- ② **AAnn_mongo_update.png**
- ③ **dbtest.js (cds_dht22 folder)**
- ④ **dbtest2.js (cds_dht22 folder)**

Lecture materials



● References & good sites

- ✓ <http://www.arduino.cc> Arduino Homepage
- ✓ <http://www.nodejs.org/ko> Node.js
- ✓ <https://plot.ly/> plotly
- ✓ <https://www.mongodb.com/> MongoDB
- ✓ <http://www.w3schools.com> By w3schools
- ✓ <http://www.github.com> GitHub

Target of this class

Real-time Weather Station from nano 33 BLE sensors



on Time: 2020-09-09 10:27:17.321

