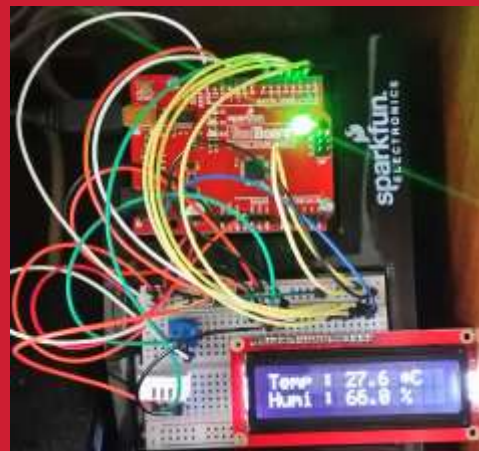




Arduino-IOT

[wk13]

Arduino + Node Data storaging II



Visualization of Signals using Arduino,
Node.js & Storing Signals in MongoDB
& Mining Data using Python



Comsi, INJE University

2nd semester, 2019

Email : chaos21c@gmail.com

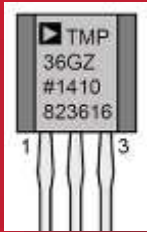


My ID

ID	성명
AA01	김관용
AA02	백동진
AA03	김도훈
AA04	김희찬
AA05	류재현
AA06	문민규
AA07	박진석
AA08	이승협
AA09	표혜성
AA10	김다영
AA11	성소진
AA12	김해인
AA13	신송주
AA14	윤지훈

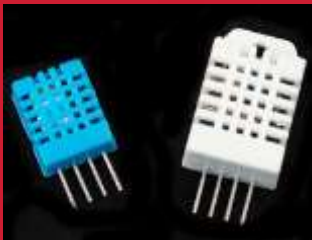


[Review]



◆ [wk12]

- RT Data storaging with MongoDB
- Multi-sensor circuits (cds-dht22)
- Complete your project
- Upload folder: AAnn_Rpt10



wk12 : Practice : AAnn_Rpt10

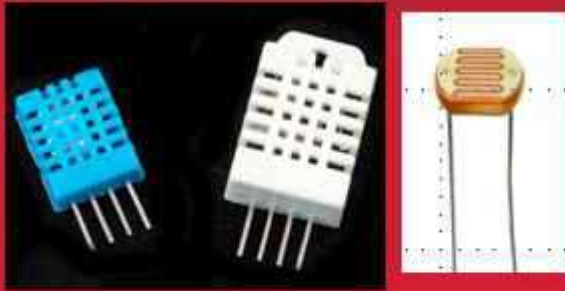
◆ [Target of this week]

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

제출폴더명 : **AAnn_Rpt10**

- 압축할 파일들

- ① **AAnn_mongo_schemas.png**
- ② **AAnn_mongo_update.png**
- ③ **AAnn_iot_mongodb.png**
- ④ **AAnn_iot_mongodb_web.png**
- ⑤ **All *.ino**
- ⑥ **All *.js**
- ⑦ **All *.html**



[Goal]

Arduino + Node.js

+ plotly.js

+ MongoDB

→ Data storaging

& visualization



A5.1 Introduction to data visualization

아두이노 센서 회로

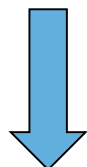


직렬모니터/플로터 모니터링



LCD 모니터링

Node.js



Plotly.js



웹 모니터링



A5. Introduction to IoT service

System (Arduino, sDevice, ...)



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



Visualization & monitoring



Data storing & mining



Service

Arduino data on network socket

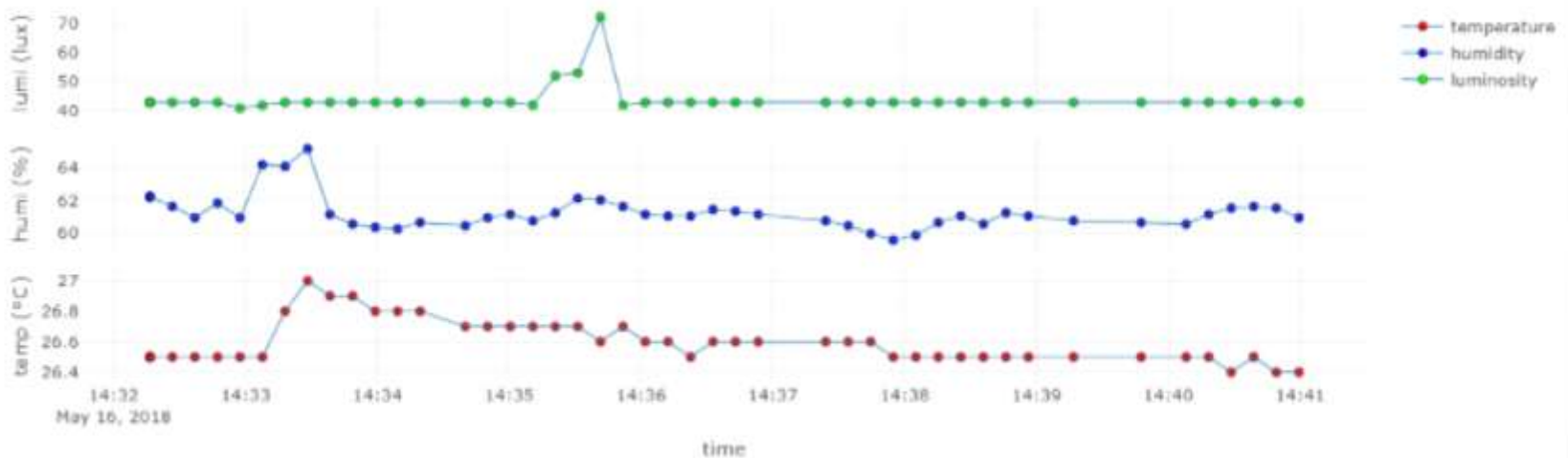


Arduino data + plotly

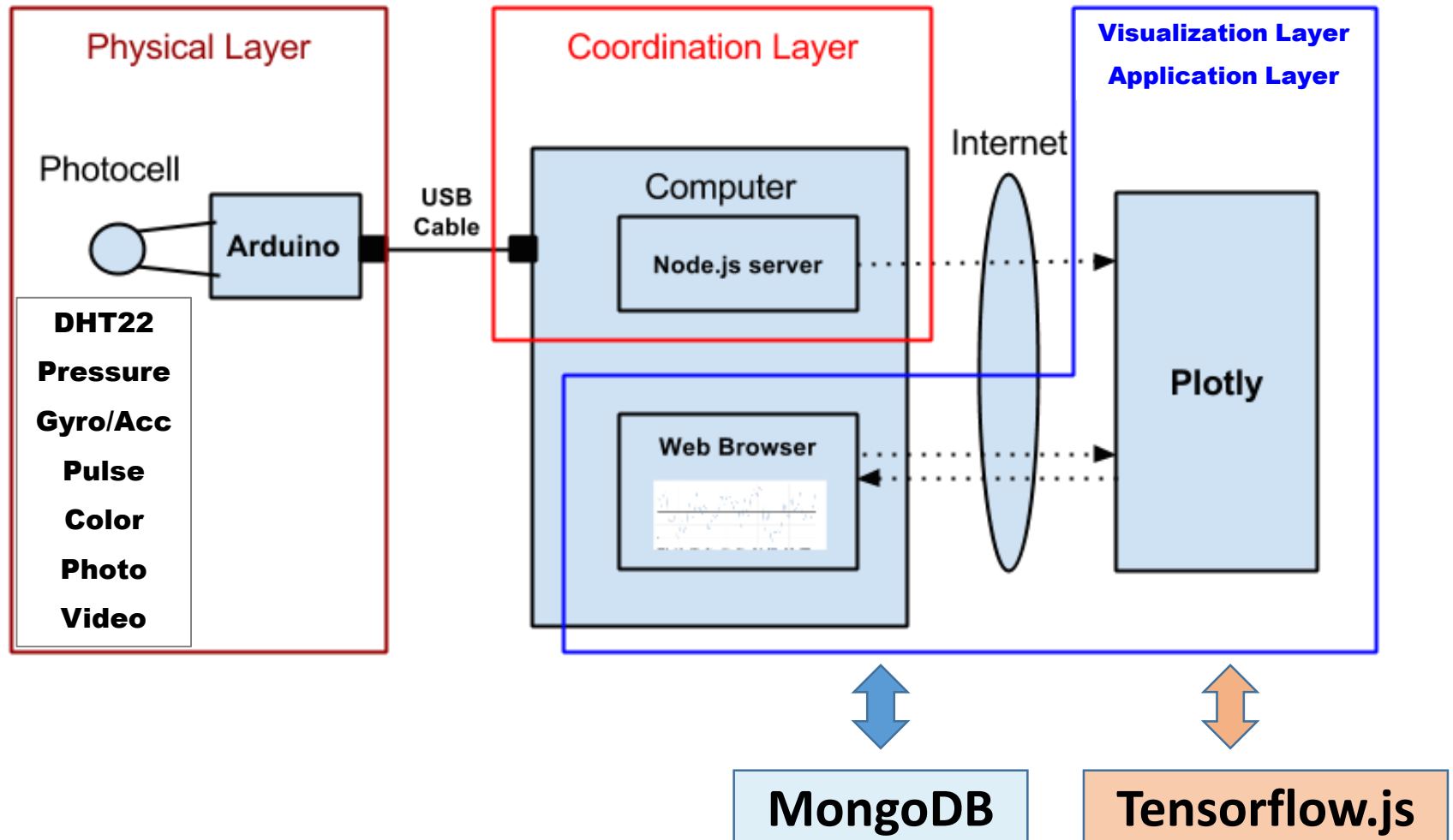
Real-time Weather Station from sensors



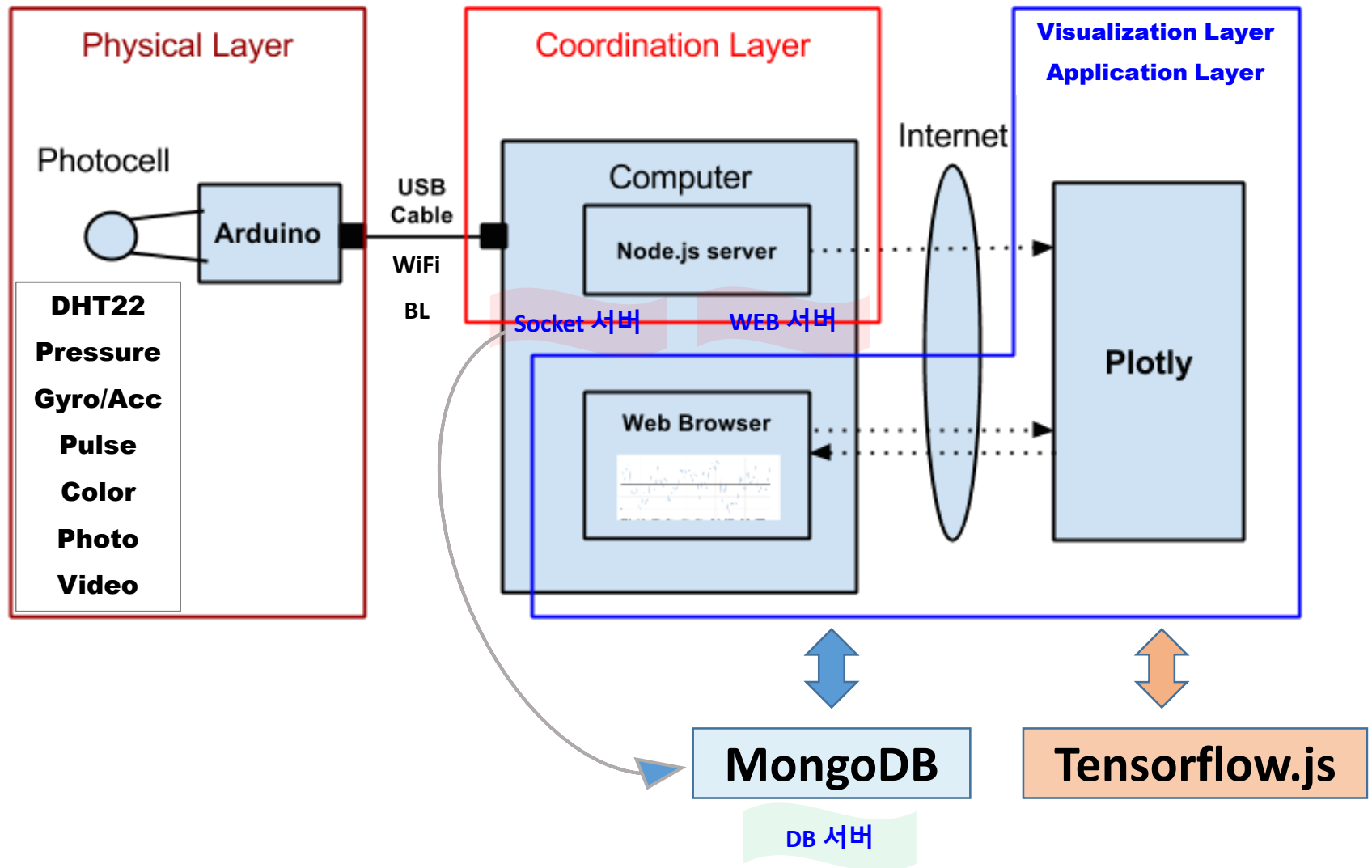
on Time: 2018-05-16 14:40:59.402



Layout [H S C]



Layout [H S C-IoT]

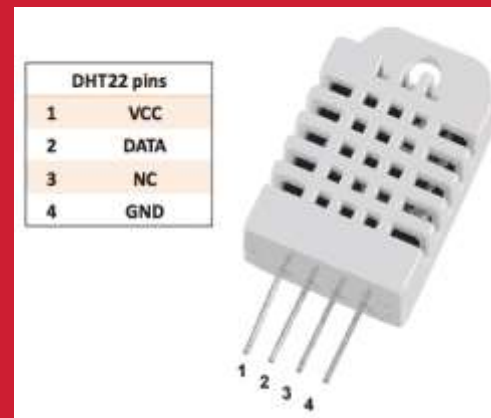




Arduino & Node.js & MongoDB

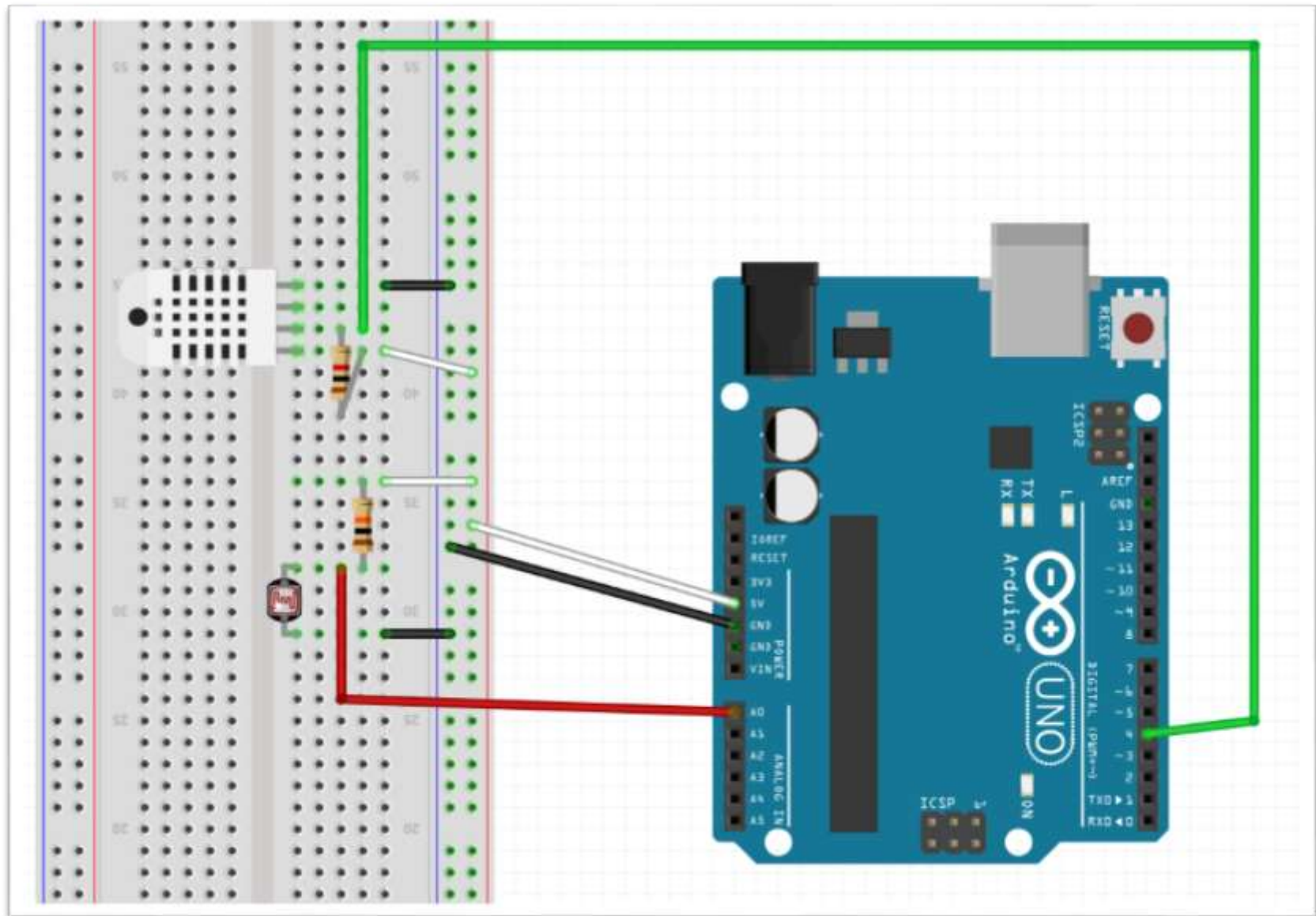


Multi-sensors
DHT22 + CdS





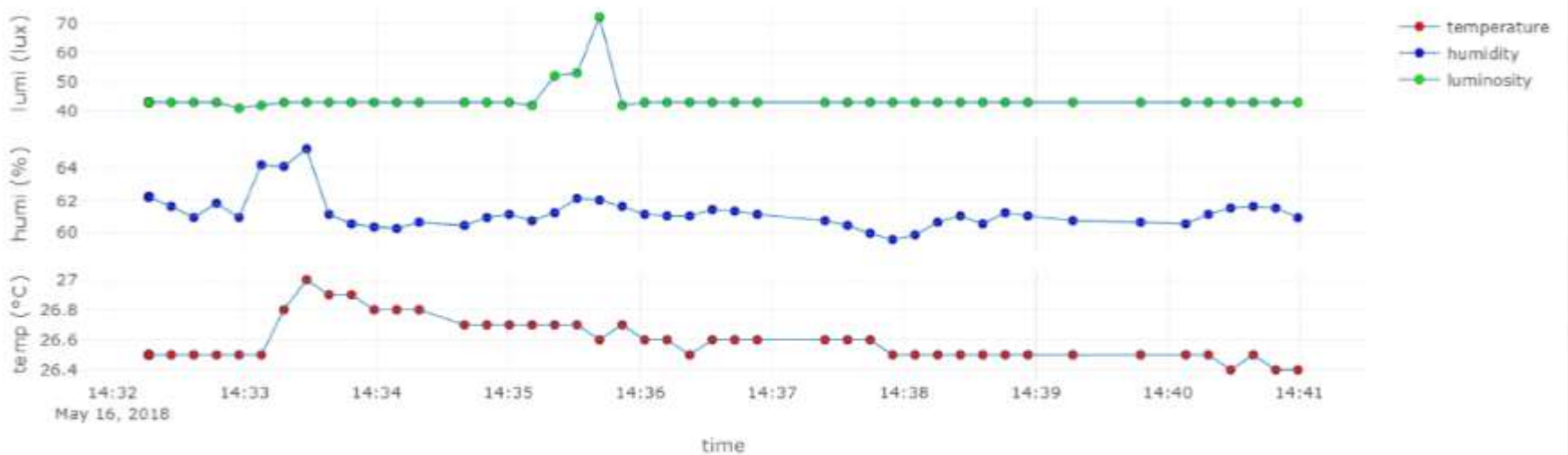
DHT22 + CdS : circuit



Real-time Weather Station from sensors



on Time: 2018-05-16 14:40:59.402





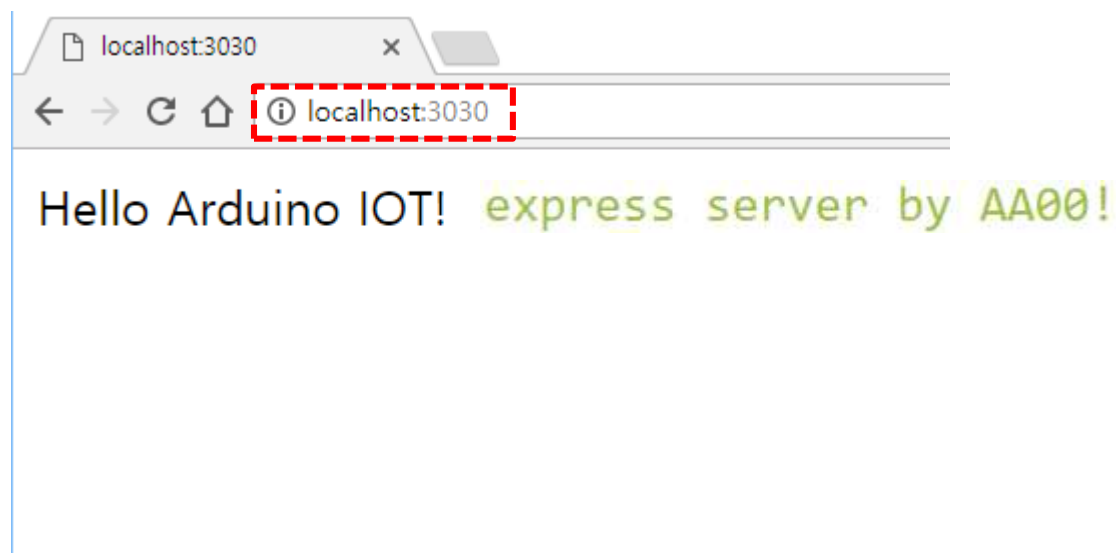
Arduino & Node.js & MongoDB & Express server





A5.9.6 DHT22 + CdS + Node.js + MongoDB

2.4 cds_dht22_express.js → routing1, <http://localhost:3030/>





A5.9.6 DHT22 + CdS + Node.js + MongoDB

2.5 cds_dht22_express.js → routing2 <http://localhost:3030/iot>

```
[{"_id": "5a683ff83cdf6353104a5463", "date": "2018-01-24", "time": "17:12:40.708", "temperature": "18.6", "humidity": "10.1", "luminosity": "178", "__v": 0}, {"_id": "5a683ffa3cdf6353104a5464", "date": "2018-01-24", "time": "17:12:42.979", "temperature": "18.7", "humidity": "10.3", "luminosity": "179", "__v": 0}, {"_id": "5a683ffd3cdf6353104a5465", "date": "2018-01-24", "time": "17:12:45.251", "temperature": "18.6", "humidity": "10.2", "luminosity": "180", "__v": 0}, {"_id": "5a683fff3cdf6353104a5466", "date": "2018-01-24", "time": "17:12:47.523", "temperature": "18.6", "humidity": "10.2", "luminosity": "179", "__v": 0}, {"_id": "5a6840013cdf6353104a5467", "date": "2018-01-24", "time": "17:12:49.779", "temperature": "18.6", "humidity": "10.2", "luminosity": "177", "__v": 0}, {"_id": "5a6840043cdf6353104a5468", "date": "2018-01-24", "time": "17:12:52.052", "temperature": "18.6", "humidity": "10.2", "luminosity": "178", "__v": 0}, {"_id": "5a6840063cdf6353104a5469", "date": "2018-01-24", "time": "17:12:54.322", "temperature": "18.6", "humidity": "10.2", "luminosity": "176", "__v": 0}, {"_id": "5a6840083cdf6353104a546a", "date": "2018-01-24", "time": "17:12:56.594", "temperature": "18.6", "humidity": "10.2", "luminosity": "176", "__v": 0}, {"_id": "5a68400a3cdf6353104a546b", "date": "2018-01-24", "time": "17:12:58.866", "temperature": "18.6", "humidity": "10.2", "luminosity": "178", "__v": 0}, {"_id": "5a68400d3cdf6353104a546c", "date": "2018-01-24", "time": "17:13:01.138", "temperature": "18.6", "humidity": "10.2", "luminosity": "178", "__v": 0}, {"_id": "5a68400f3cdf6353104a546d", "date": "2018-01-24", "time": "17:13:03.410", "temperature": "18.6", "humidity": "10.2", "luminosity": "178", "__v": 0}
```

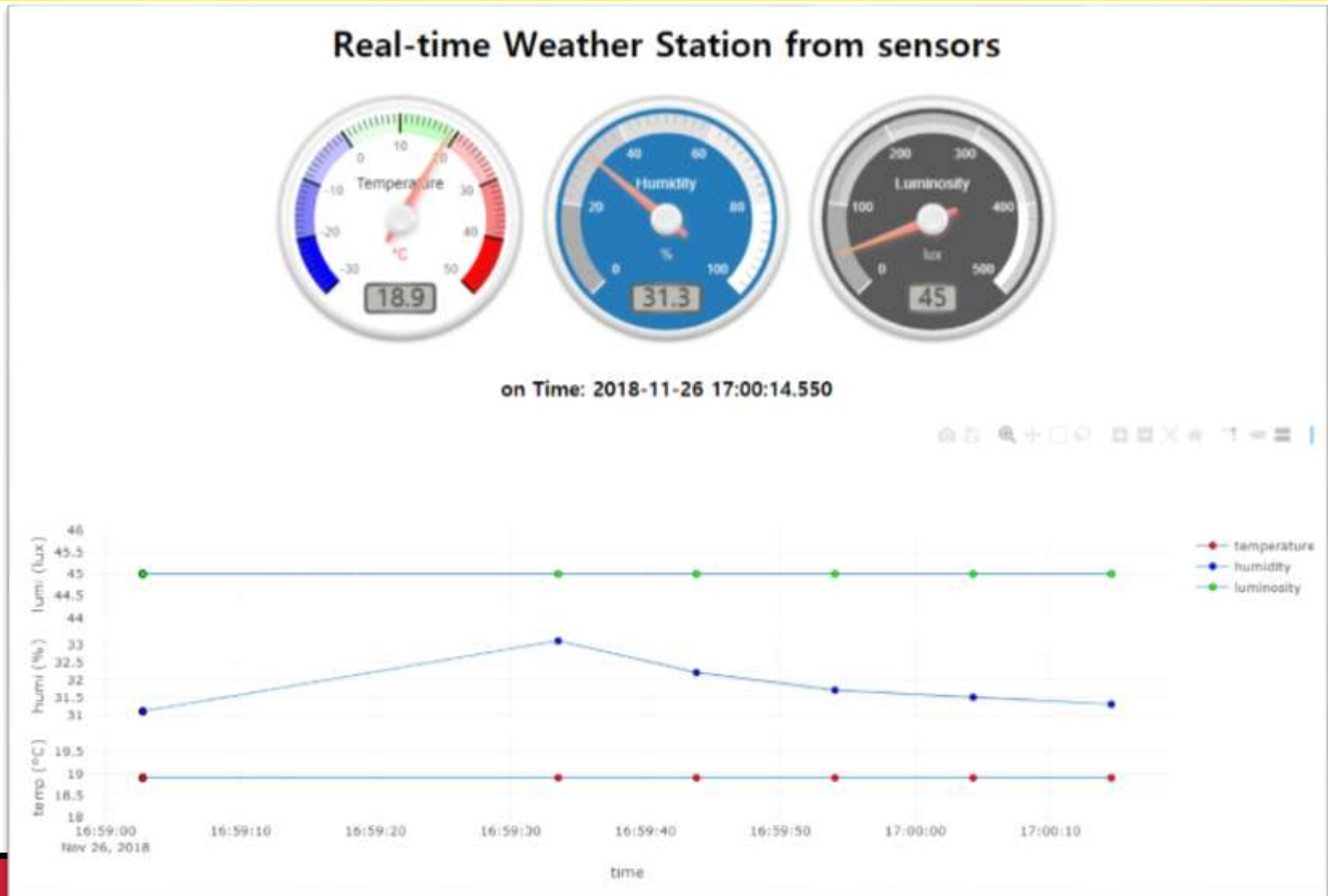
Save as

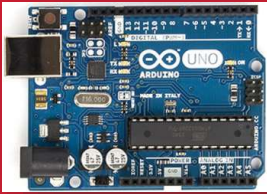
AAnn_iot_mongodb_web.png



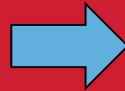
A5.9.6 DHT22 + CdS + Node.js + MongoDB

2.7 copy `cds_dht22_client.html` & `gauge.min.js` → `./public/` subfolder
http://localhost:3030/client_cds_dht22.html (web root folder)





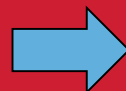
3-servers



3000

Cloud (DB)

Network-Socket

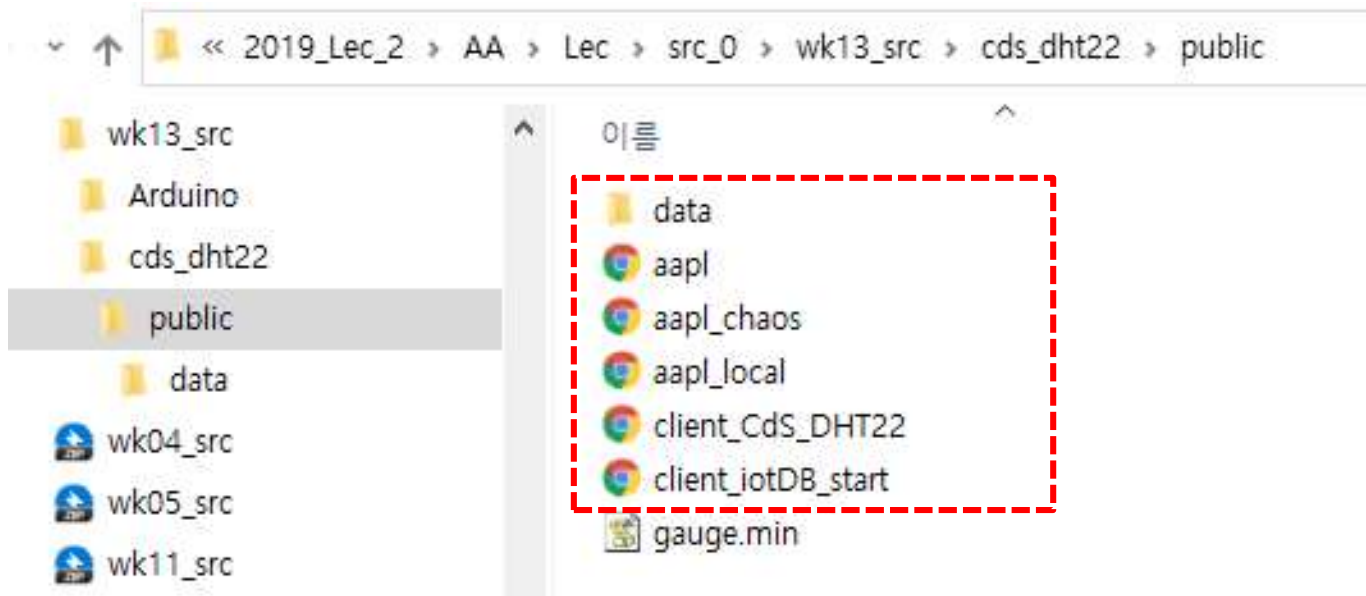


3030

Services (Client)



2.8 CORS bug (Cross Origin Resource Sharing)



Apple 사의 주가그래프를 그리는 **html client** 3개를 실행하고 결과를 비교.

- **Local file**에 접근 불허
- **CORS problem**

- **public** 폴더로 **html,data**를 복사한 후에 비교.



A5.9.6 DHT22 + CdS + Node.js + MongoDB

2.9 **CORS patch** on the express server → [cds_dht22_express.js](#)

Node cmd에서 'cors' module 설치 (version 2.8.4 이상)

npm install -save cors

```
1 // cds_dht22_express.js
2 // Express with CORS
3 var express = require('express');
4 var cors = require('cors'); // CORS: Cross Origin Resource Sharing
5 var app = express();
6 // CORS
7 app.use(cors());
8
9 var web_port = 3030; // express port
10 // MongoDB
11 var mongoose = require('mongoose');
12 var Schema = mongoose.Schema; // Schema object
13 // MongoDB connection
14 mongoose.connect('mongodb://localhost:27017/iot11'); // DB name
15 var db = mongoose.connection;
16 db.on('error', console.error.bind(console, 'connection error:'));
17 db.once('open', function callback () {
18     console.log("mongo db connection OK.");
19 });
```



DHT22 + CdS + Node.js + MongoDB

Web monitoring

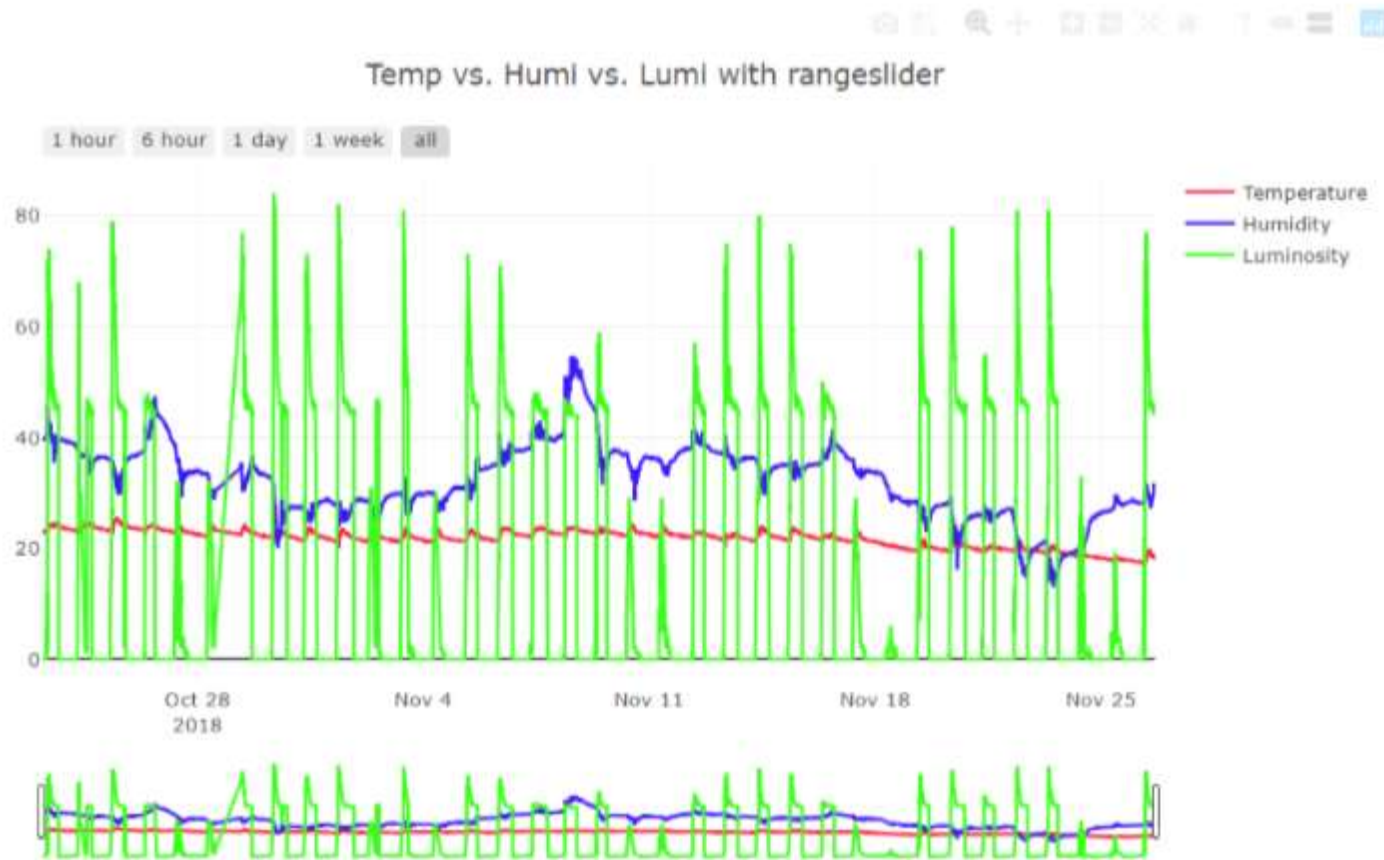


A5.9.7 DHT22 + CdS + Node.js + MongoDB

Web monitoring-1: month

MongoDB database visualization by AA00

Time series : Multi sensor data



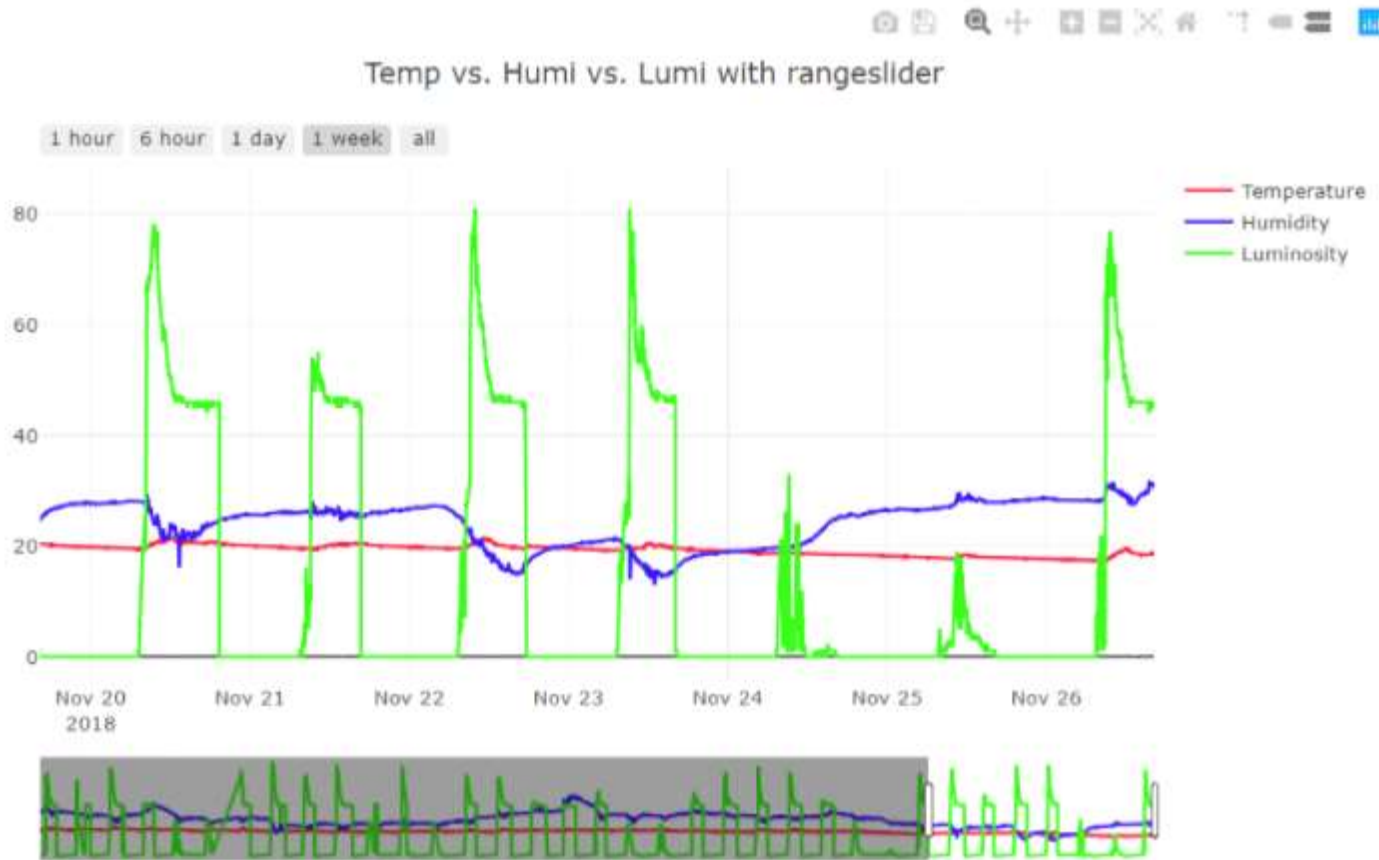


A5.9.7 DHT22 + CdS + Node.js + MongoDB

Web monitoring-2: week

MongoDB database visualization by AA00

Time series : Multi sensor data



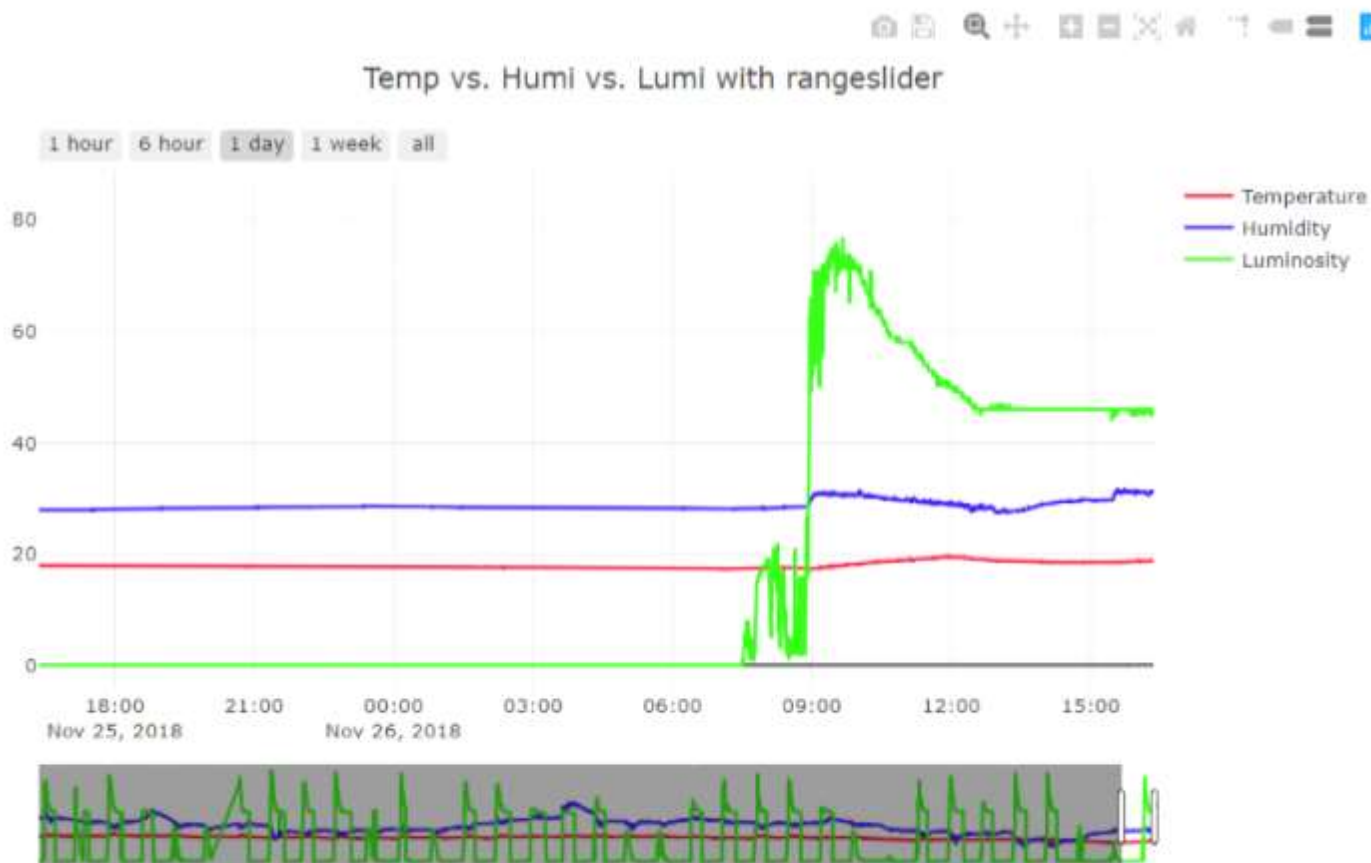


A5.9.7 DHT22 + CdS + Node.js + MongoDB

Web monitoring-3: day

MongoDB database visualization by AA00

Time series : Multi sensor data





A5.9.8 DHT22 + CdS + Node.js + MongoDB

3.1 Web client: [client_iotDB.html](#)

```
client_iotDB.html x
1 <!DOCTYPE html>
2 <head>
3   <meta charset="utf-8">
4   <!-- Plotly.js -->
5   <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
6 </head>
7 <body>
8   <h1>MongoDB database visualization by AA00</h1>
9   <hr>
10  <h2>Time series : Multi sensor data</h2>
11
12  <!-- Plotly chart will be drawn inside this DIV -->
13  <div id="myDiv" style="width: 900px; height: 600px"></div>
14
```



A5.9.7 DHT22 + CdS + Node.js + MongoDB

3.2 Web client: [client_iotDB.html](#)

```
<script>
  <!-- JAVASCRIPT CODE GOES HERE -->

  Plotly.d3.json("http://localhost:3030/iot", function(err, json){
    //alert(json);
    alert(JSON.stringify(json)); // It works!!!
    //alert(JSON.parse(eval(json)));
    if(err) throw err;

    var date = [];
    var temp = [];
    var humi = [];
    var lumi = [];
    var jsonData = eval(JSON.stringify(json));
    //alert(jsonData.length);
    //alert(jsonData[2].luminosity);

    for (var i = 0; i < jsonData.length; i++) {
      date[i] = jsonData[i].date;
      temp[i] = jsonData[i].temperature ;
      humi[i] = jsonData[i].humidity;
      lumi[i] = jsonData[i].luminosity;
    }
  }
```

**JSON
file**

```
{"_id":"5a683ffd3cdf6353104a5465","date":"2018-01-24  
17:12:45.251","temperature":"18.6","humidity":"10.2","luminosity":"180","__v":0},  
{"_id":"5a683fff3cdf6353104a5466","date":"2018-01-24  
17:12:47.523","temperature":"18.6","humidity":"10.2","luminosity":"179","__v":0},
```



A5.9.7 DHT22 + CdS + Node.js + MongoDB

3.3 Web client: [client_iotDB.html](#) – data & layout

```
// time series of sensor data
var trace1 = {
  type: "scatter",
  mode: "lines",
  name: 'Temperature',
  x: date,
  y: temp,
  line: {color: '#fc1234'}
}

var trace2 = {
  type: "scatter",
  mode: "lines",
  name: 'Humidity',
  x: date,
  y: humi,
  line: {color: '#3412fc'}
}

var trace3 = {
  type: "scatter",
  mode: "lines",
  name: 'Luminosity',
  x: date,
  y: lumi,
  line: {color: '#34fc12'}
}

var data = [trace1, trace2, trace3];
```

```
// Layout with builtin rangeslider
var layout = {
  title: 'Temp vs. Humi vs. Lumi with rangeslider',
  xaxis: {
    autorange: true,
    range: [date[0], date[date.length-1]],
    rangeselector: {buttons: [
      {
        count: 1,
        label: '1 hour',
        step: 'hour',
        stepmode: 'backward'
      },
      {
        count: 6,
        label: '6 hour',
        step: 'hour',
        stepmode: 'backward'
      },
      {
        count: 24,
        label: '1 day',
        step: 'hour',
        stepmode: 'backward'
      },
      {
        count: 7,
        label: '1 week',
        step: 'day',
        stepmode: 'backward'
      },
      {step: 'all'}
    ]},
    rangeslider: {range: [date[0], date[date.length-1]],
      type: 'date'
    },
    type: 'date'
  },
  yaxis: {
    autorange: true,
    range: [0, 300],
    type: 'linear'
  }
}

Plotly.newPlot('myDiv', data, layout);
})
```



A5.9.7 DHT22 + CdS + Node.js + MongoDB

3.4 Web client: [client_iotDB.html](#) – load iot data in json file

The screenshot shows a web browser window with the title 'client_iotDB.html'. The address bar shows the file path: file:///D:/Portable/NodeJSPortable/Data/hs00/iot/cds_dht22/client_iotDB.html. The main content area displays 'MongoDB database visualization' and 'Time series : Multi sensor data'. A right sidebar shows the page content, which includes a JSON array of sensor data points. A red dashed box highlights this JSON array.

이 페이지 내용:

```
[{"_id":"5aa584d0ea0bd2064cb1f9ab","date":"2018-03-12 04:34:40.662","temperature":"16.6","humidity":"24.9","luminosity":"0"}, {"_id":"5aa584daea0bd2064cb1f9ac","date":"2018-03-12 04:34:50.923","temperature":"16.6","humidity":"24.9","luminosity":"0"}, {"_id":"5aa584e5ea0bd2064cb1f9ad","date":"2018-03-12 04:35:01.168","temperature":"16.6","humidity":"24.9","luminosity":"0"}, {"_id":"5aa584efea0bd2064cb1f9ae","date":"2018-03-12 04:35:11.429","temperature":"16.6","humidity":"24.9","luminosity":"0"}, {"_id":"5aa584f9ea0bd2064cb1f9af","date":"2018-03-12 04:35:21.678","temperature":"16.6","humidity":"24.9","luminosity":"0"}]
```

Save as [AAnn_iot_json.png](#)

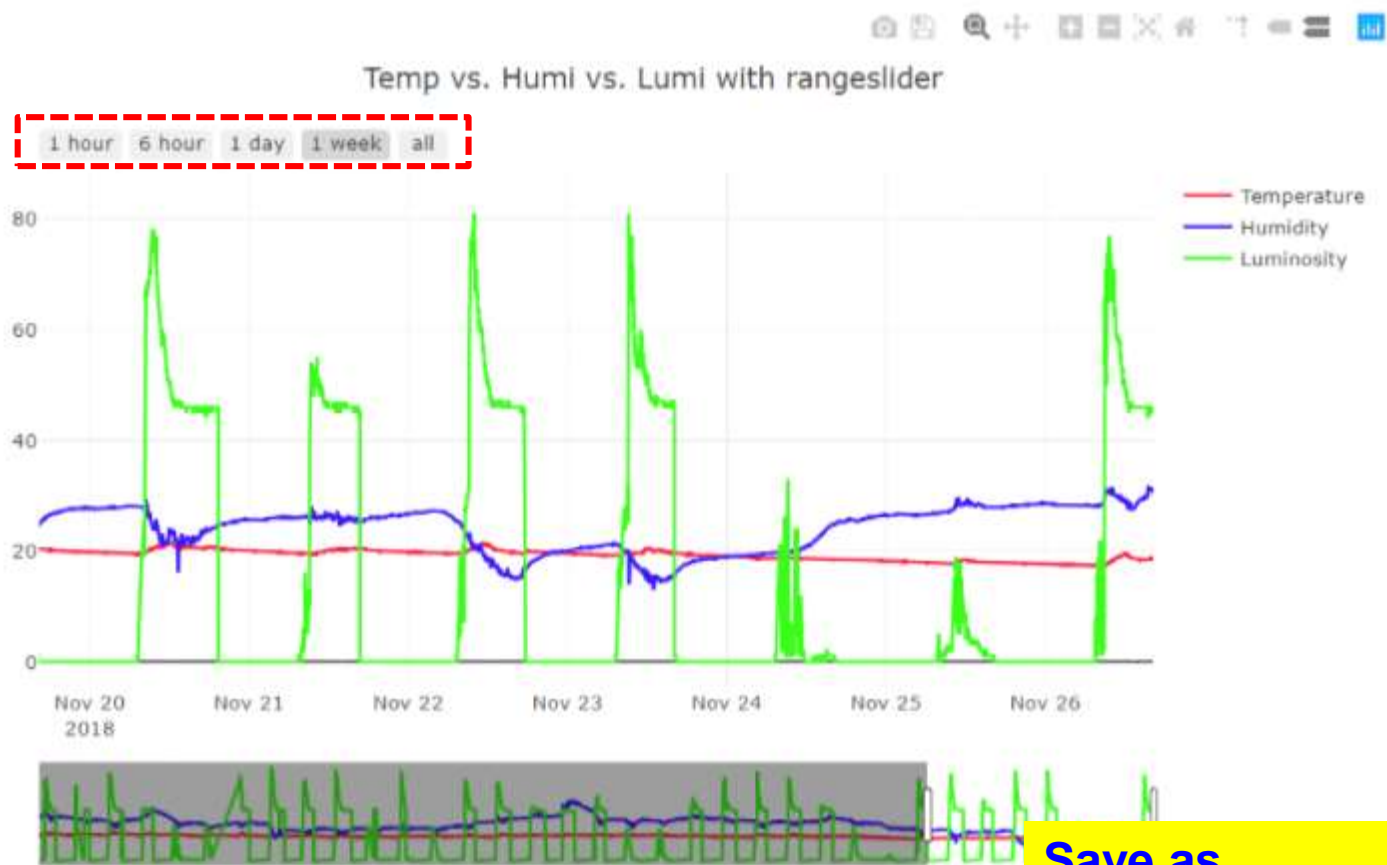


A5.9.7 DHT22 + CdS + Node.js + MongoDB

3.5 Web client: [client_iotDB.html](#) – iot DB monitoring (public 폴더에서 제공)

MongoDB database visualization by AA00

Time series : Multi sensor data



Save as
AAnn_iot_client.png



MongoDB data management

- Query in mongo shell
- Export & import MongoDB
- Using and understanding iot data with Python (or R)



A5.9.8 MongoDB management

1. Query in Mongo shell

`db.sensors.count()` → sensors collection에 있는 도큐먼트 (문서)의 수

`db.sensors.find().sort({_id: 1}).limit(10)` → 오래된 document 10개 추출

`db.sensors.find().sort({_id: -1}).limit(10)` → 최근 document 10개 추출

`db.sensors.find({date: {$gt: "2019-11-26 22:26:05"}})` → 특정 시간 이후 document 추출

`db.sensors.find({temperature: {$gt: 29}})` → 온도가 29도를 넘는 document 추출

<https://docs.mongodb.com/manual/tutorial/query-documents/>



A5.9.8 MongoDB management

1.1 Query in Mongo shell

`db.sensors.count()` → `sensors collection` 에 있는 문서의 총수

`db.sensors.find({temperature: {$gt: 29.5}}).count()`

→ `sensors collection` 에 있는 온도가 29.5를 초과하는 문서의 수

C:\ 명령 프롬프트 - mongo

```
> db.sensors.count()  
227209
```

```
> db.sensors.find({temperature: {$gt:29.5}}).count()  
11
```

```
> db.sensors.find({temperature: {$gt:26}}).count()  
17773
```



A5.9.8 MongoDB management

1.2 Query in Mongo shell

db.sensors.find().sort({_id: -1}).limit(10) → 최근 데이터 10개 추출

명령 프롬프트 - mongo

```
> show dbs
Warning: 0.000GB
iot11    0.013GB
local    0.000GB
> use iot11
switched to db iot11
> show collections
sensors
```

사용 중인 db 이름으로 변경이 필요! -- use iotxx

```
> db.sensors.find().sort({_id: -1}).limit(10)
{ "_id" : ObjectId("5b0d51f82d151211a8b9e2ef"), "date" : "2018-05-29 22:13:28.218", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
{ "_id" : ObjectId("5b0d51ed2d151211a8b9e2ee"), "date" : "2018-05-29 22:13:17.958", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
{ "_id" : ObjectId("5b0d51e32d151211a8b9e2ed"), "date" : "2018-05-29 22:13:07.713", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
{ "_id" : ObjectId("5b0d51d92d151211a8b9e2ec"), "date" : "2018-05-29 22:12:57.453", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
{ "_id" : ObjectId("5b0d51cf2d151211a8b9e2eb"), "date" : "2018-05-29 22:12:47.208", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
{ "_id" : ObjectId("5b0d51c42d151211a8b9e2ea"), "date" : "2018-05-29 22:12:36.947", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
{ "_id" : ObjectId("5b0d51ba2d151211a8b9e2e9"), "date" : "2018-05-29 22:12:26.687", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
{ "_id" : ObjectId("5b0d51b02d151211a8b9e2e8"), "date" : "2018-05-29 22:12:16.442", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
{ "_id" : ObjectId("5b0d51a62d151211a8b9e2e7"), "date" : "2018-05-29 22:12:06.182", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
{ "_id" : ObjectId("5b0d519b2d151211a8b9e2e6"), "date" : "2018-05-29 22:11:55.937", "temperature" : "26.3", "humidity" : "49.8", "luminosity" : "0", "__v" : 0 }
```

시간이 역순!



A5.9.8 MongoDB management

1.3 Query in Mongo shell

db.sensors.find({temperature: {\$gt: 29}}) → 29도 초과하는 문서추출

명령 프롬프트 - mongo

```
[{"_id" : ObjectId("5b0ab1c7f4dbca05df913fec"), "date" : "2018-03-12 09:17:59.512", "temperature" : 28.6, "humidity" : 13.7, "luminosity" : 60 }
Type "it" for more
> db.sensors.find({temperature: {$gt: 29}})
{"_id" : ObjectId("5b0ab1c7f4dbca05df91426a"), "date" : "2018-03-12 11:06:51.069", "temperature" : 29.1, "humidity" : 14.4, "luminosity" : 60 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df91426b"), "date" : "2018-03-12 11:07:01.330", "temperature" : 29.2, "humidity" : 14.3, "luminosity" : 60 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df91426c"), "date" : "2018-03-12 11:07:11.575", "temperature" : 29.1, "humidity" : 14.2, "luminosity" : 60 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df914377"), "date" : "2018-03-12 11:52:49.318", "temperature" : 29.1, "humidity" : 14.4, "luminosity" : 57 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df914378"), "date" : "2018-03-12 11:52:59.563", "temperature" : 29.2, "humidity" : 14.4, "luminosity" : 58 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df914379"), "date" : "2018-03-12 11:53:09.826", "temperature" : 29.2, "humidity" : 14.3, "luminosity" : 58 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df91437b"), "date" : "2018-03-12 11:53:20.069", "temperature" : 29.1, "humidity" : 14.3, "luminosity" : 57 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143a9"), "date" : "2018-03-12 12:01:21.996", "temperature" : 29.2, "humidity" : 14.7, "luminosity" : 55 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143aa"), "date" : "2018-03-12 12:01:32.258", "temperature" : 29.1, "humidity" : 14.6, "luminosity" : 55 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143ad"), "date" : "2018-03-12 12:02:03.008", "temperature" : 29.1, "humidity" : 14.5, "luminosity" : 55 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143ae"), "date" : "2018-03-12 12:02:13.268", "temperature" : 29.2, "humidity" : 14.4, "luminosity" : 55 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143af"), "date" : "2018-03-12 12:02:23.529", "temperature" : 29.3, "humidity" : 14.3, "luminosity" : 55 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143b0"), "date" : "2018-03-12 12:02:33.774", "temperature" : 29.4, "humidity" : 14.2, "luminosity" : 55 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143b1"), "date" : "2018-03-12 12:02:54.280", "temperature" : 29.4, "humidity" : 14.1, "luminosity" : 55 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143b2"), "date" : "2018-03-12 12:02:44.035", "temperature" : 29.4, "humidity" : 14.2, "luminosity" : 55 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143b3"), "date" : "2018-03-12 12:03:04.541", "temperature" : 29.4, "humidity" : 14, "luminosity" : 55 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143b4"), "date" : "2018-03-12 12:03:14.785", "temperature" : 29.3, "humidity" : 13.9, "luminosity" : 54 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143b5"), "date" : "2018-03-12 12:03:25.046", "temperature" : 29.2, "humidity" : 13.9, "luminosity" : 54 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143b6"), "date" : "2018-03-12 12:03:35.291", "temperature" : 29.1, "humidity" : 14, "luminosity" : 54 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9143eb"), "date" : "2018-03-12 12:12:38.735", "temperature" : 29.2, "humidity" : 14.7, "luminosity" : 53 }
Type "it" for more
> db.sensors.find({temperature: {$gt: 31}})
> db.sensors.find({temperature: {$gt: 30}})
> db.sensors.find({temperature: {$gt: 29.5}})
{"_id" : ObjectId("5b0ab1c7f4dbca05df914427"), "date" : "2018-03-12 12:22:53.957", "temperature" : 29.6, "humidity" : 13.7, "luminosity" : 50 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df914428"), "date" : "2018-03-12 12:23:04.218", "temperature" : 29.7, "humidity" : 13.6, "luminosity" : 50 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df914429"), "date" : "2018-03-12 12:23:14.479", "temperature" : 29.7, "humidity" : 13.4, "luminosity" : 50 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df91442a"), "date" : "2018-03-12 12:23:24.724", "temperature" : 29.7, "humidity" : 13.4, "luminosity" : 51 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df91442b"), "date" : "2018-03-12 12:23:34.985", "temperature" : 29.7, "humidity" : 13.4, "luminosity" : 51 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df91442d"), "date" : "2018-03-12 12:23:45.229", "temperature" : 29.6, "humidity" : 13.4, "luminosity" : 51 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df9149d6"), "date" : "2018-03-12 16:32:03.827", "temperature" : 29.6, "humidity" : 14.8, "luminosity" : 46 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df914a0e"), "date" : "2018-03-12 16:40:46.764", "temperature" : 29.6, "humidity" : 14.8, "luminosity" : 46 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df914a0f"), "date" : "2018-03-12 16:40:57.025", "temperature" : 29.6, "humidity" : 14.8, "luminosity" : 46 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df916289"), "date" : "2018-03-13 10:30:48.354", "temperature" : 29.6, "humidity" : 19.2, "luminosity" : 63 }
{"_id" : ObjectId("5b0ab1c7f4dbca05df91628a"), "date" : "2018-03-13 10:30:38.108", "temperature" : 29.6, "humidity" : 19.3, "luminosity" : 64 }
```




A5.9.8 MongoDB management

1.4 Query in Mongo shell

db.sensors.find({date: {\$gt: "2018-05-26"}})

→ 5월 26일 이후 데이터 전부 추출 (시간 변경)

```
명령 프롬프트 - mongo
> db.sensors.find( {date: {$gt: "2018-05-26"}} )
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a026"), "date" : "2018-05-26 00:00:03.167", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a028"), "date" : "2018-05-26 00:00:23.672", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a029"), "date" : "2018-05-26 00:00:13.427", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a02a"), "date" : "2018-05-26 00:00:33.933", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a02b"), "date" : "2018-05-26 00:00:44.177", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a02c"), "date" : "2018-05-26 00:01:04.682", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a02d"), "date" : "2018-05-26 00:00:54.438", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a02e"), "date" : "2018-05-26 00:01:25.188", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a02f"), "date" : "2018-05-26 00:01:14.943", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a030"), "date" : "2018-05-26 00:01:35.448", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a031"), "date" : "2018-05-26 00:01:45.710", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a032"), "date" : "2018-05-26 00:01:55.954", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a033"), "date" : "2018-05-26 00:02:06.215", "temperature" : 25.8, "humidity" : 36.9, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a034"), "date" : "2018-05-26 00:02:26.720", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a035"), "date" : "2018-05-26 00:02:16.460", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a036"), "date" : "2018-05-26 00:02:36.965", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a037"), "date" : "2018-05-26 00:02:47.225", "temperature" : 25.8, "humidity" : 36.7, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a038"), "date" : "2018-05-26 00:02:57.470", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a039"), "date" : "2018-05-26 00:03:07.731", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
{"_id" : ObjectId("5b0ab1ccf4dbca05df94a03a"), "date" : "2018-05-26 00:03:17.975", "temperature" : 25.8, "humidity" : 36.8, "luminosity" : 0 }
Type "it" for more
> db.sensors.find( {date: {$gt: "2018-05-27"}} )
>
```



A5.9.8 MongoDB management

2. Import or export MongoDB (windows cmd 창에서 실행)

- **mongoimport** -d dbName -c collectionName --type csv --headerline --file fileName.csv
- **mongoexport** -d dbName -c collectionName --fields <field1,field2,...> --limit=nn --type csv --out fileName.csv

json 또는 csv 파일로 import/export

<https://docs.mongodb.com/manual/reference/program/mongoimport/>

<https://docs.mongodb.com/manual/reference/program/mongoexport/>



A5.9.8 MongoDB management

2.1.1 Import MongoDB (windows cmd 창에서 실행)

➤ `mongoimport -d s10 -c sensors --type csv --headerline --file sensor10.csv`

```
명령 프롬프트 - mongo
D:\mongodb>
D:\mongodb>mongoimport -d s10 -c sensors --type csv --headerline --file sensor10.csv
2018-05-27T21:49:00.669+0900 connected to: localhost
2018-05-27T21:49:00.292+0900 imported 10 documents

D:\mongodb>mongo
MongoDB shell version v3.6.5
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.5
Server has startup warnings:
2018-05-27T05:37:28.213-0700 | CONTROL | [initandlisten]
2018-05-27T05:37:28.213-0700 | CONTROL | [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-05-27T05:37:28.214-0700 | CONTROL | [initandlisten] ** Read and write access to data and configuration is u
nrestricted.
2018-05-27T05:37:28.214-0700 | CONTROL | [initandlisten]
2018-05-27T05:37:28.214-0700 | CONTROL | [initandlisten] ** WARNING: This server is bound to localhost.
2018-05-27T05:37:28.214-0700 | CONTROL | [initandlisten] ** Remote systems will be unable to connect to this ser
ver.
2018-05-27T05:37:28.214-0700 | CONTROL | [initandlisten] ** Start the server with --bind_ip <address> to specify
which IP
2018-05-27T05:37:28.216-0700 | CONTROL | [initandlisten] ** addresses it should serve responses from, or with --
bind_ip all to
2018-05-27T05:37:28.217-0700 | CONTROL | [initandlisten] ** bind to all interfaces. If this behavior is desired,
start the
2018-05-27T05:37:28.218-0700 | CONTROL | [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warn
ing.
2018-05-27T05:37:28.219-0700 | CONTROL | [initandlisten]
2018-05-27T05:37:28.220-0700 | CONTROL | [initandlisten]
2018-05-27T05:37:28.221-0700 | CONTROL | [initandlisten] ** WARNING: The file system cache of this machine is configured
to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
2018-05-27T05:37:28.223-0700 | CONTROL | [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2018-05-27T05:37:28.227-0700 | CONTROL | [initandlisten]
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
s10 0.000GB
> use s10
switched to db s10
> show collections
sensors

> db.sensors.count()
10
```




A5.9.8 MongoDB management

2.1.2 Import MongoDB (windows cmd 창에서 실행)

➤ `mongoimport -d s_all -c sensors --type csv --headerline --file sensor_all.csv`

```
D:\mongodb>dir
D 드라이브의 볼륨: Yi_Data
볼륨 일련 번호: 3A94-C8A0

D:\mongodb 디렉터리

2018-05-27 오후 09:41 <DIR> .
2018-05-27 오후 09:41 <DIR> ..
2018-05-27 오후 10:21 <DIR> data
2018-05-26 오후 12:55 26,267 mongodb_export.PNG
2018-05-27 오후 05:58 193,912 mongodb_export_csv.png
2018-05-27 오후 05:20 177,001 mongo_export_count.png
2018-04-06 오후 09:37 83,233 R_lm_notebook.png
2018-05-26 오후 12:52 397 sensor10.csv
2018-05-26 오후 12:54 8,251,185 sensor_all.csv
                6개 파일      8,731,995 바이트 남음
                3개 디렉터리 812,761,526,272 바이트 남음

D:\mongodb>mongoimport -d s_all -c sensors --type csv --headerline --file sensor_all.csv
2018-05-27T22:25:26.313+0900 connected to: localhost
2018-05-27T22:25:28.513+0900 [#####] s_all.sensors 992KB/7.87MB (12.3%)
2018-05-27T22:25:31.503+0900 [#####] s_all.sensors 6.48MB/7.87MB (82.4%)
2018-05-27T22:25:32.264+0900 [#####] s_all.sensors 7.87MB/7.87MB (100.0%)
2018-05-27T22:25:32.264+0900 imported 227209 documents

D:\mongodb>
```

명령 프롬프트 - mongo

```
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
s10 0.000GB
s_all 0.009GB
> use s_all
switched to db s_all
> show collections
sensors
> db.sensors.count()
227209
>
```

[DIY] Import된 's_all' db 에 대하여 앞에서 배운 query를 테스트해서 결과를 확인한다.



A5.9.8 MongoDB management

2.2 Export MongoDB (windows cmd 창에서 실행, dbName을 iotx로 변경!)

- **mongoexport -d s_all -c sensors --type=csv --fields date,temperature,humidity,luminosity --limit=100 --out s100.csv**

```
명령 프롬프트
D:\#mongodb>mongoexport -d s_all -c sensors --type=csv --fields date,temperature,humidity,luminosity
--limit=100 --out s100.csv
2018-05-27T22:38:05.300+0900    connected to: localhost
2018-05-27T22:38:05.405+0900    exported 100 records

D:\#mongodb>dir
D 드라이브의 볼륨: Yi_Data
볼륨 일련 번호: 3A94-C8A0

D:\#mongodb 디렉터리

2018-05-27   오후 10:38   <DIR>          .
2018-05-27   오후 10:38   <DIR>          ..
2018-05-27   오후 10:26   <DIR>          data
2018-05-26   오후 12:55           26,267 mongodb_export.PNG
2018-05-27   오후 05:58       193,912 mongodb_export_csv.png
2018-05-27   오후 05:20       177,001 mongo_export_count.png
2018-04-06   오후 09:37           83,233 R_lm_notebook.png
2018-05-27   오후 10:38           3,459 s100.csv
2018-05-26   오후 12:52           397 sensor10.csv
2018-05-26   오후 12:54       8,251,185 sensor_all.csv
              7개 파일              8,735,454 바이트
              3개 디렉터리      812,751,392,768 바이트 남음

D:\#mongodb>
```




A5.9.8 MongoDB management

2.3 Advanced export with query (windows cmd 창에서 실행)

iotxx db의 특정 시간 이후의 데이터 100개를 csv 파일 (s100.csv)로 저장

- **Mongoexport** -d **iotxx** -c sensors /query:"{date: {\$gt: '2018-05-29 22:26:06'}}" --limit=100 --fields date,temperature,humidity,luminosity --type=csv --out **s100.csv**

명령 프롬프트

```
C:\Users\biochaos>mongoexport -d iot11 -c sensors /query:"{date:{$gt:'2018-05-29 22:26:05'}}" --limit 100 --fields date,temperature,humidity,luminosity --type=csv --out sensor100.csv
2018-05-29T22:49:19.431+0900    connected to: localhost
2018-05-29T22:49:19.576+0900    exported 100 records
```

[Tip] iot db의 최근 데이터 500개를 csv 파일 (s500.csv)로 저장할 때,

- **mongoexport** -d **iot** -c sensors --sort "{_id: -1}" --limit=500 --fields date,temperature,humidity,luminosity --type=csv --out **s500.csv**



A5.9.8 MongoDB management

[Tip] **iot db**의 최근 데이터 **500**개를 **csv** 파일 (**s500.csv**)로 저장할 때,

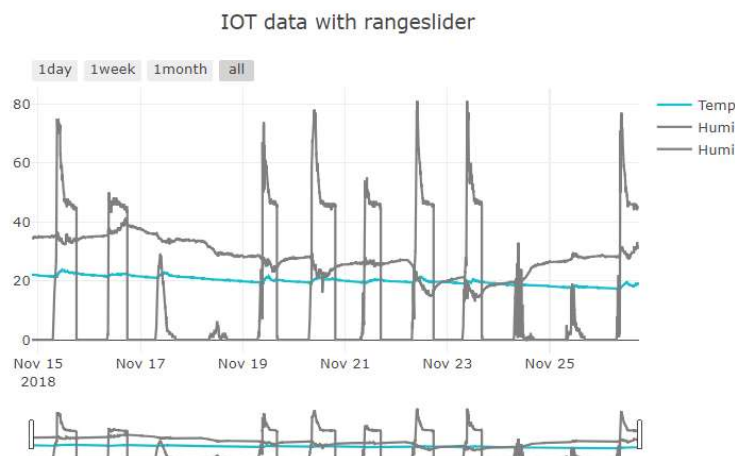
➤ **mongoexport -d iot -c sensors --sort "{_id: -1}" --limit=500 --fields date,temperature,humidity,luminosity --type=csv --out s500.csv**

```
C:\Users\biochaos>mongoexport -d iot11 -c sensors --sort "{_id:-1}" --limit=100000 --type=csv --fields date,temperature,
humidity,luminosity --out iot_chaos.csv
2018-11-26T17:50:23.577+0900    connected to: localhost
2018-11-26T17:50:24.576+0900    [#####.....] iot11.sensors 64000/100000 (64.0%)
2018-11-26T17:50:24.797+0900    [#####] iot11.sensors 100000/100000 (100.0%)
2018-11-26T17:50:24.798+0900    exported 100000 records
```

	A	B	C	D
1	date	temperatu	humidity	luminosity
2	50:18.6	18.9	31.6	45
3	50:08.4	18.9	31.6	45
4	49:58.1	18.9	31.6	45
5	49:47.8	19	31.7	45
6	49:37.6	19	31.7	45
7	49:27.3	18.9	31.7	45
8	49:17.1	18.9	31.6	45

Data visualization by AAnn

Time series by AAnn





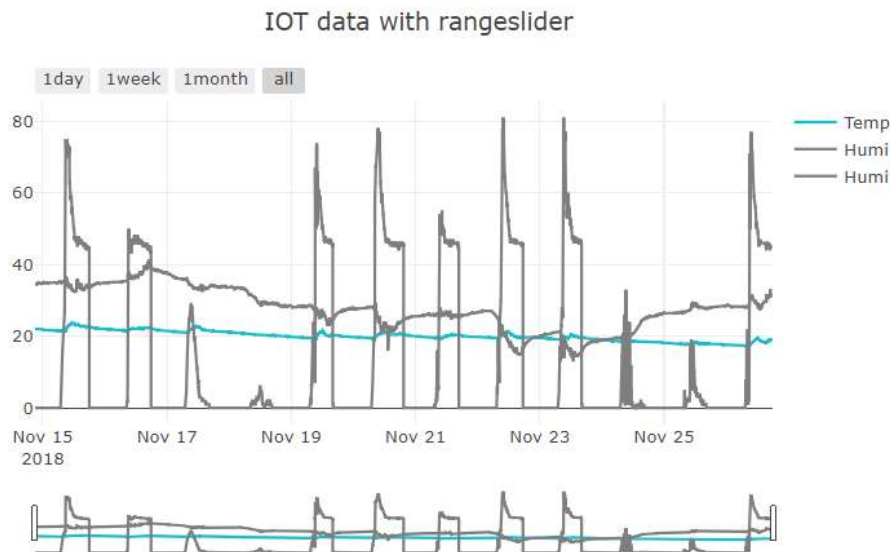
A5.9.8 MongoDB management

[DIY]

1. **iot db**의 최근 데이터 1000개를 **csv 파일 (AAnn_s1000.csv)**로 저장하시오.
2. 저장된 **AAnn_s1000.csv** 파일을 **public/data** 폴더에 복사.
3. csv 파일을 이용하는 **Rangeslider**가 포함된 웹 클라이언트 **client_iot.html** 파일을 만드시오.

Data visualization by AAnn

Time series by AAnn

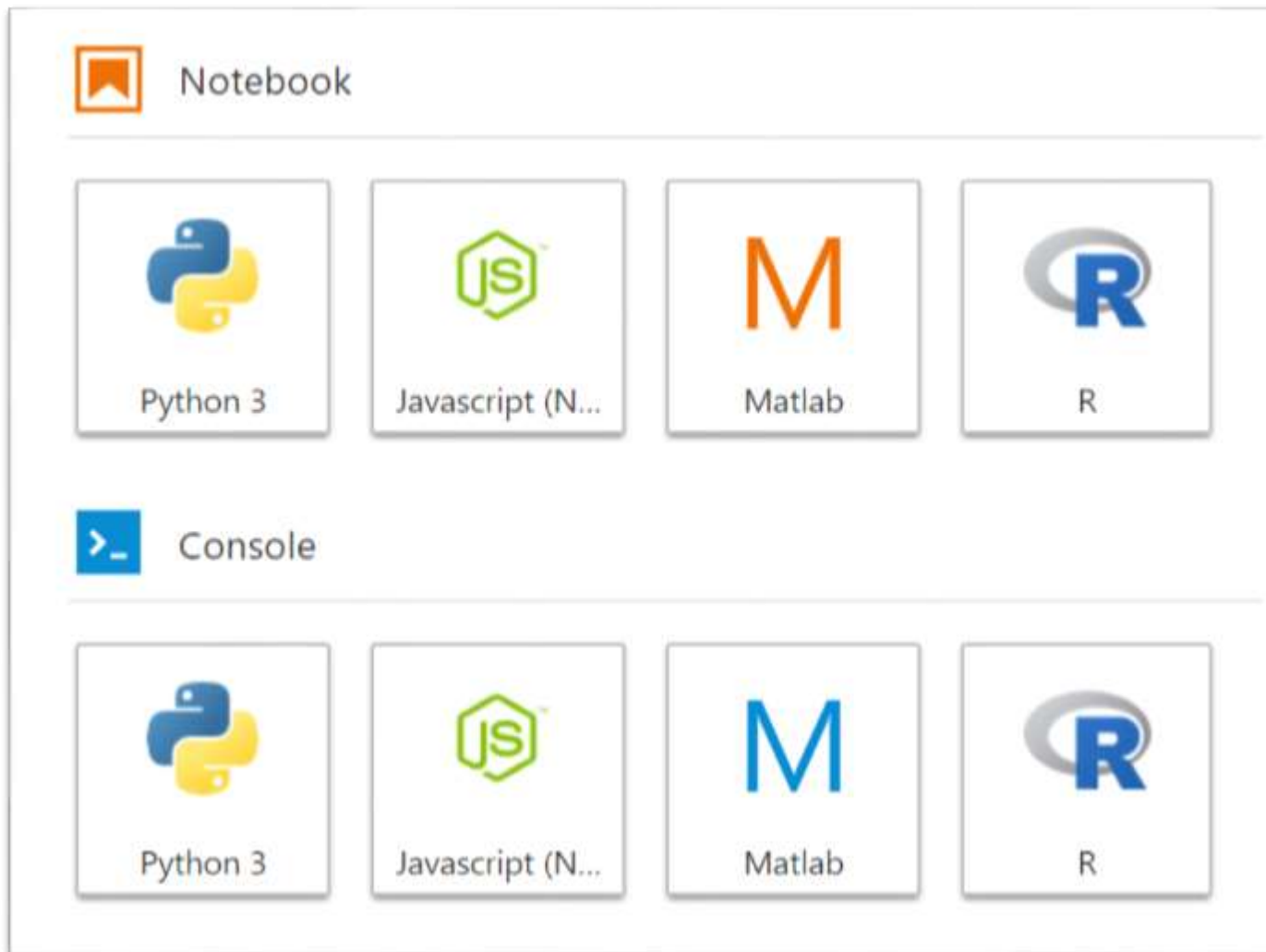


Save as
AAnn_s1000.png



IoT data mining (next week)

3. How to use and understand iot data? → Python(or R) in Colab/Jupyter lab





IoT data mining (next week)

How to use and understand iot data? → [Google Colab](#)

 [Open in Colab](#)

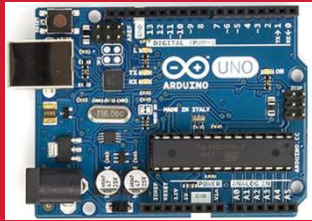
Pandas: access to the remote json from MongoDB

- The json file is generated on the fly from the express server of Node.js.
- The data stored in MongoDB are saved in the json file.
- The data are composed of three time series; temperature, humidity, and luminosity.

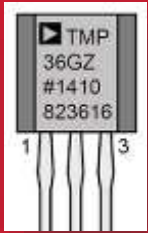
```
In [0]: import pandas as pd
```

```
In [0]: # loading json file from MongoDB via web (CORS, port=3030)
url="http://chaos.inje.ac.kr:3030/iot"
df=pd.read_json(url)
print('Large data was retrieved successfully from MongoDB!')
```

```
In [0]: df.head()
```

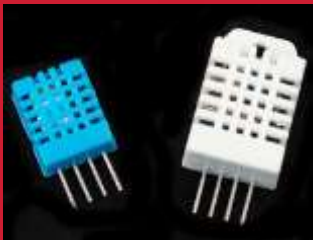


[Practice]



◆ [wk13]

- RT Data management with MongoDB
- Multi-sensor circuits(cds-dht22)
- Complete your project
- Upload folder: AAnn_Rpt11



wk13 : Practice : AAnn_Rpt11

◆ [Target of this week]

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

제출폴더명 : **AAnn_Rpt11**

- 압축할 파일들

- ① **AAnn_iot_json.png**
- ② **AAnn_iot_client.png**
- ③ **AAnn_s1000.csv** (mongoexport file)
- ④ **AAnn_s1000.png**
- ⑤ **client_IoT.html**
- ⑥ **All *.ino**
- ⑦ **All *.js**
- ⑧ **All *.html**

[Upload to github]

◆ [wk13]

- upload all work of this week
- Use your repo “aann” in github
- upload folder “aann_rpt11” in your github.

● 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



주교재 및 참고도서

아두이노와 Node.js에 기반한 IOT 신호 시각화

| 저자 이 상 훈 |

인제대학교 출판부

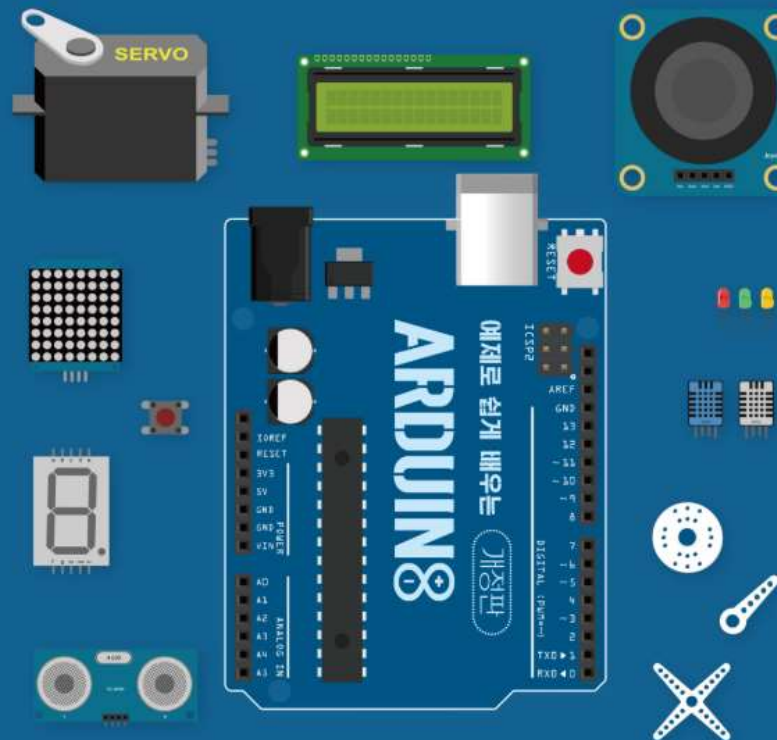
아두이노와 Node.js에 기반한

IOT 신호 시각화

| 저자 이 상 훈 |



인제대학교 출판부



예제로 쉽게 배우는

아두이노

개정판

장성용 · 김진환 지음

새로운 출판

Target of this class

Real-time Weather Station from sensors



on Time: 2018-01-22 17:58:31.012



Another target of this class

PPG with rangeslider

