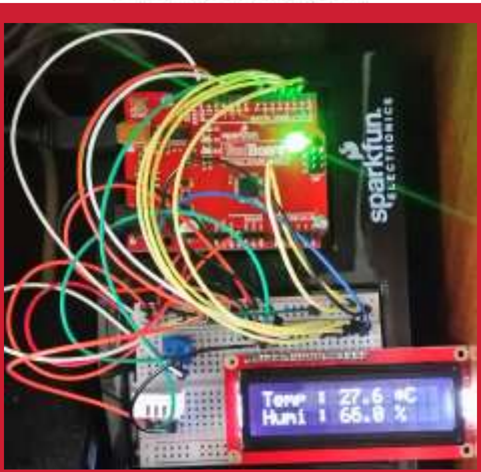
Arduino-IOT

[wk14]

Arduino + Node

Data storaging II



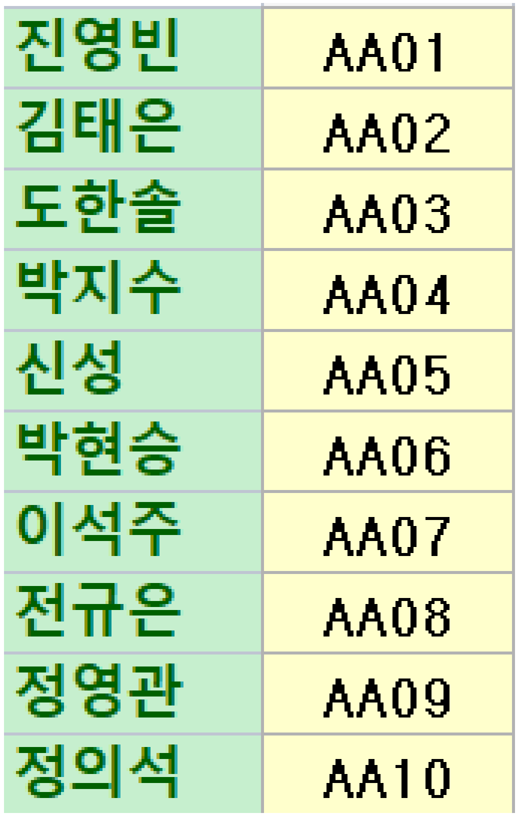
Visualization of Signals using Arduino,
Node.js & storing signals in MongoDB

Comsi, INJE University

2nd semester, 2018

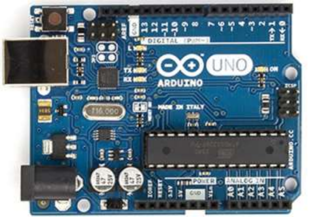
Email : chaos21c@gmail.com

My ID

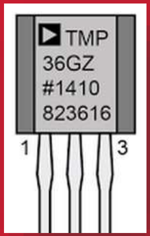


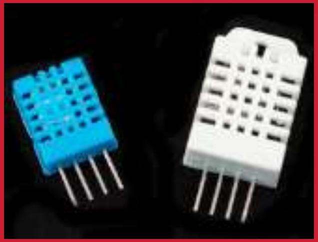
진영빈	AA01
김태은	AA02
도한솔	AA03
박지수	AA04
신성	AA05
박현승	AA06
이석주	AA07
전규은	AA08
정영관	AA09
정의석	AA10

이근재 **AA11**



[Review]





◆ [wk12]

- ➢ RT Data Visualization with node.js
- ➢ Multiple data and Usage of gauge.js
- ➢ Complete your real-time WEB charts
- ➢ Upload file name : AAnn_Rpt09.zip

[wk12] Practice-09 AAnn_Rpt09.zip

- **[Target of this week]**
  - Complete your charts
  - **Save your outcomes and compress them.**

제출파일명 : AAnn_Rpt09.zip

- 압축할 파일들

① **AAnn_DS_cds_tmp36.png**

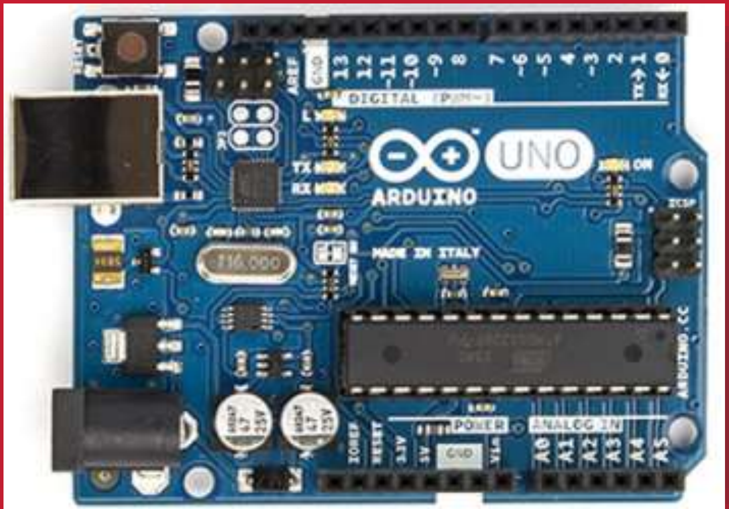
② **AAnn_cds_dht22_data.png**

③ **AAnn_cds_dht22.html**

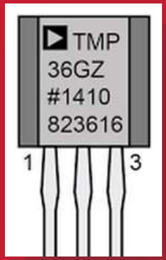
④ **AAnn_cds_dht22.png**

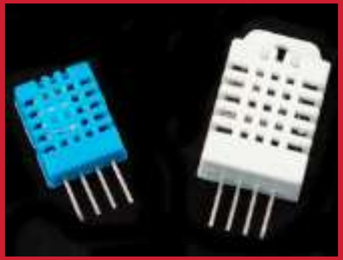
Email : chaos21c@gmail.com

[제목 : id, 이름 (수정)]



Arduino
& Node.js





IOT: HSC

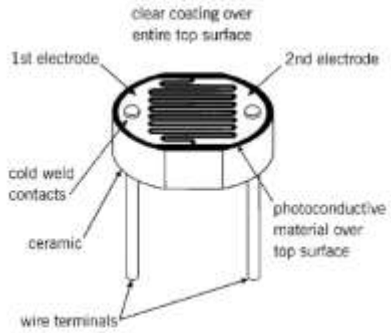
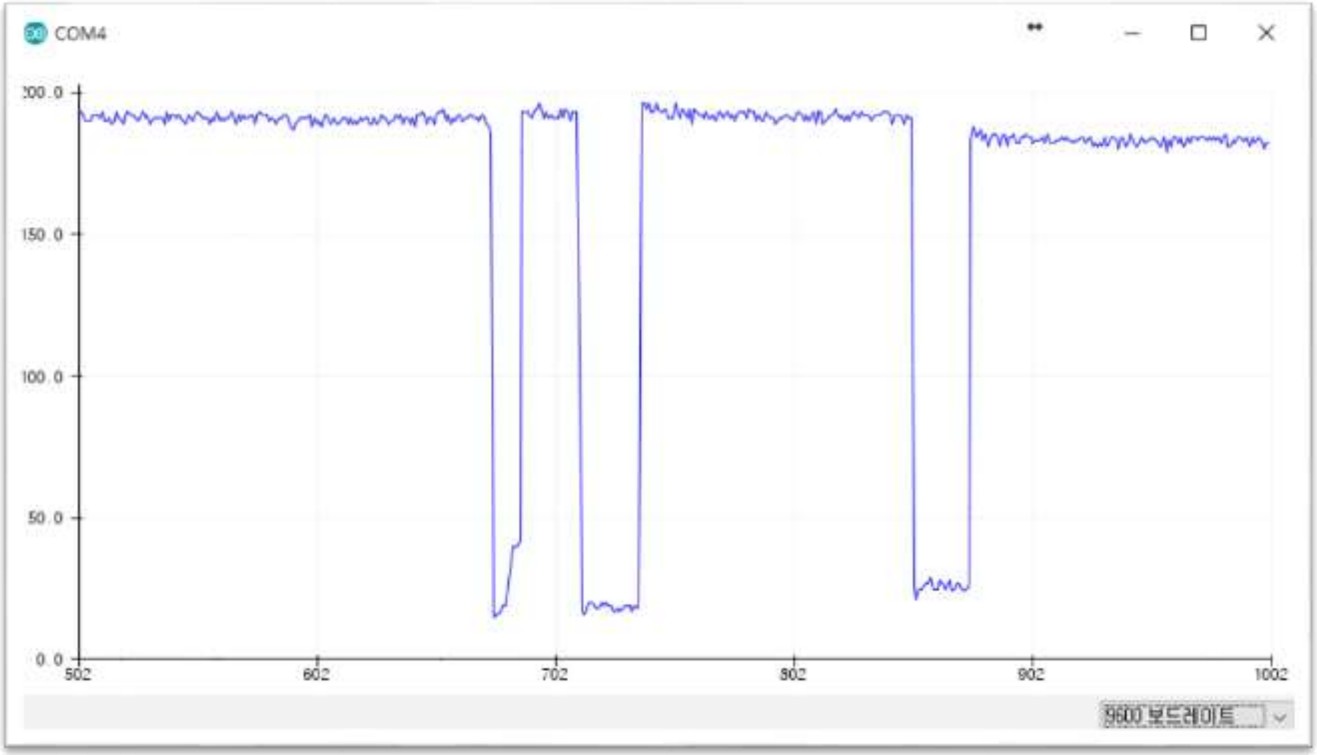
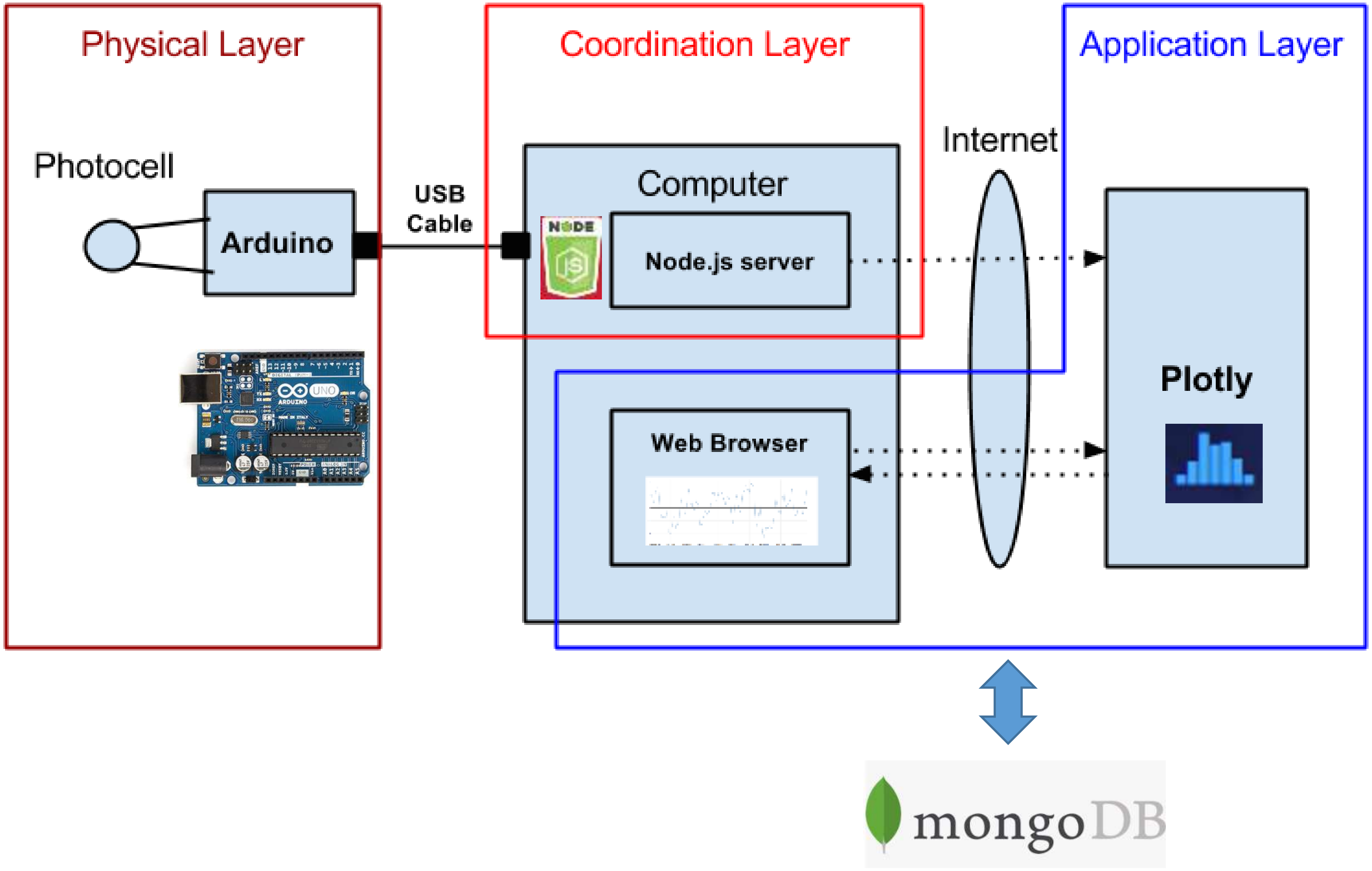


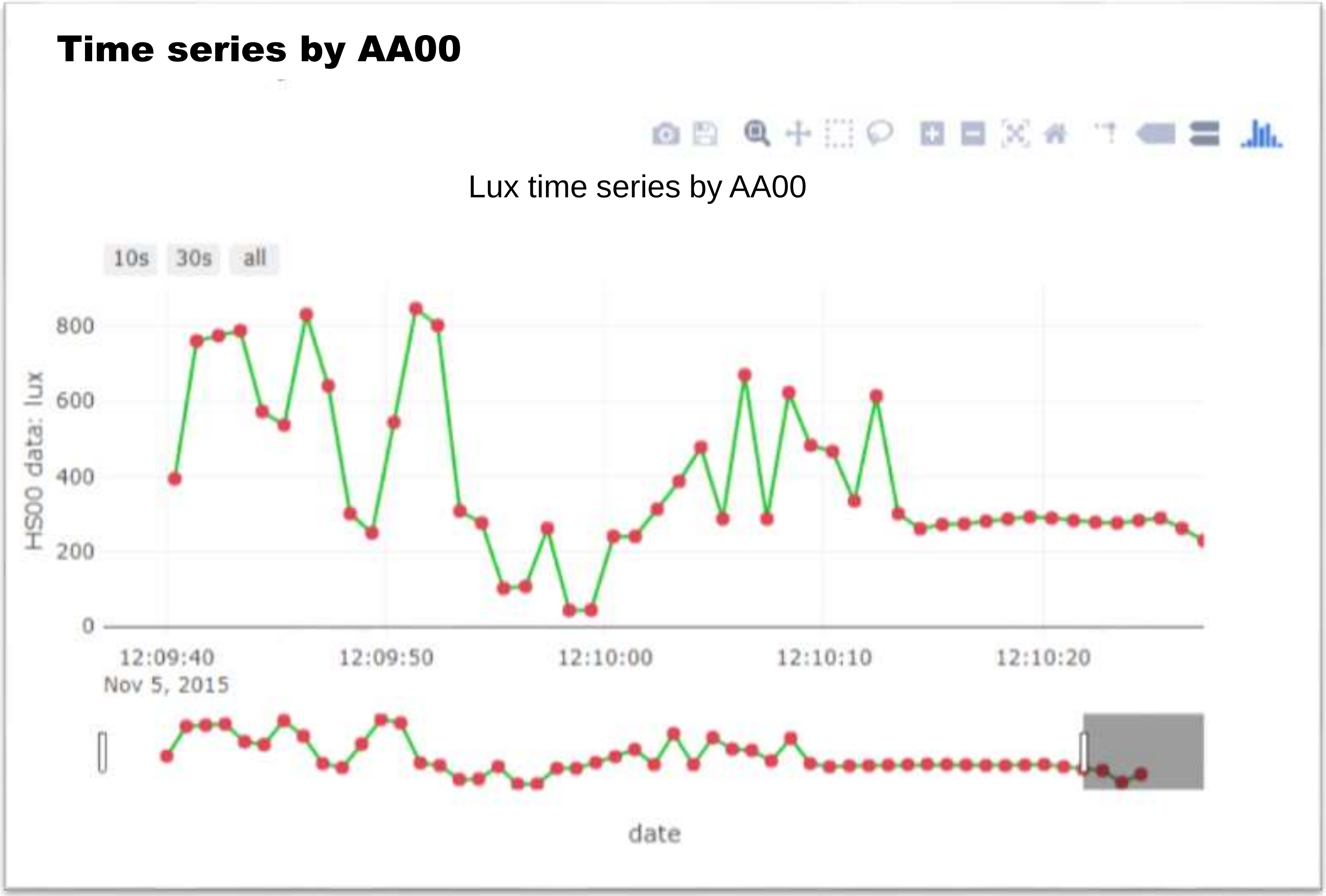
Figure 3
Typical Construction of a Plastic Coated Photocell



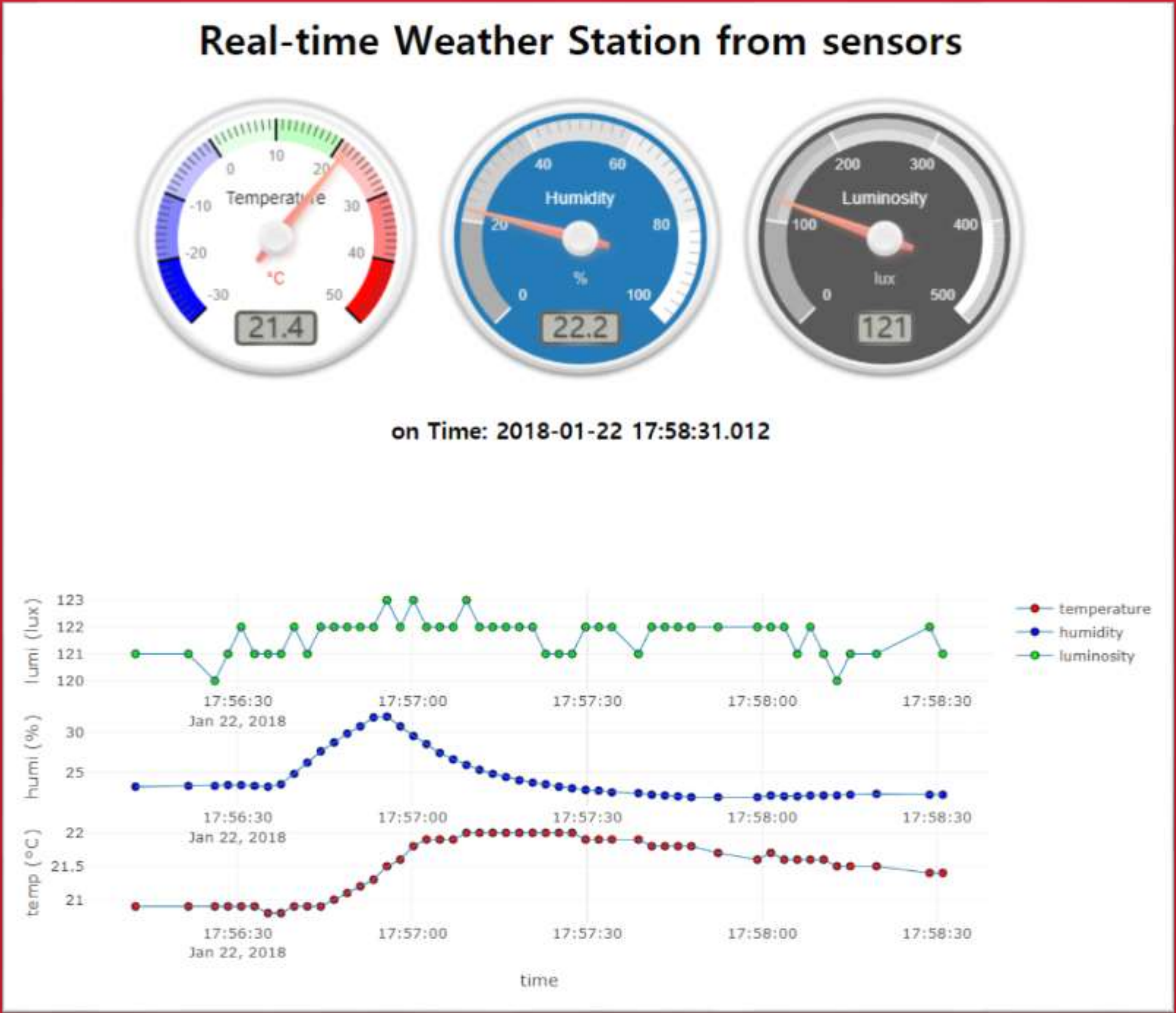
Layout [H S C]


Physical Layer
Coordination Layer
Application Layer
Photocell
Arduino
USB
Cable
Computer
NODE
JS
Node.js server
Internet
Plotly
Web Browser
mongoDB

Arduino data + plotly



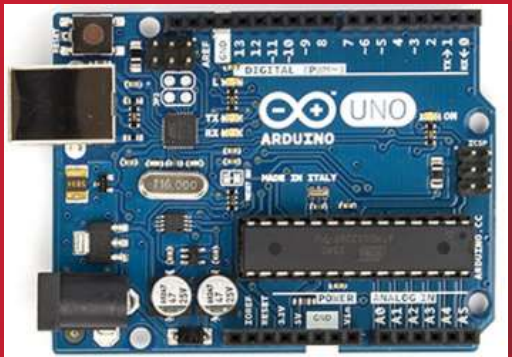
Time series by AA00

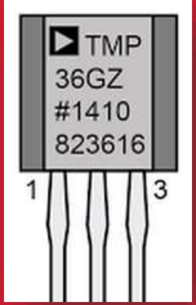


Real-time Weather Station from sensors

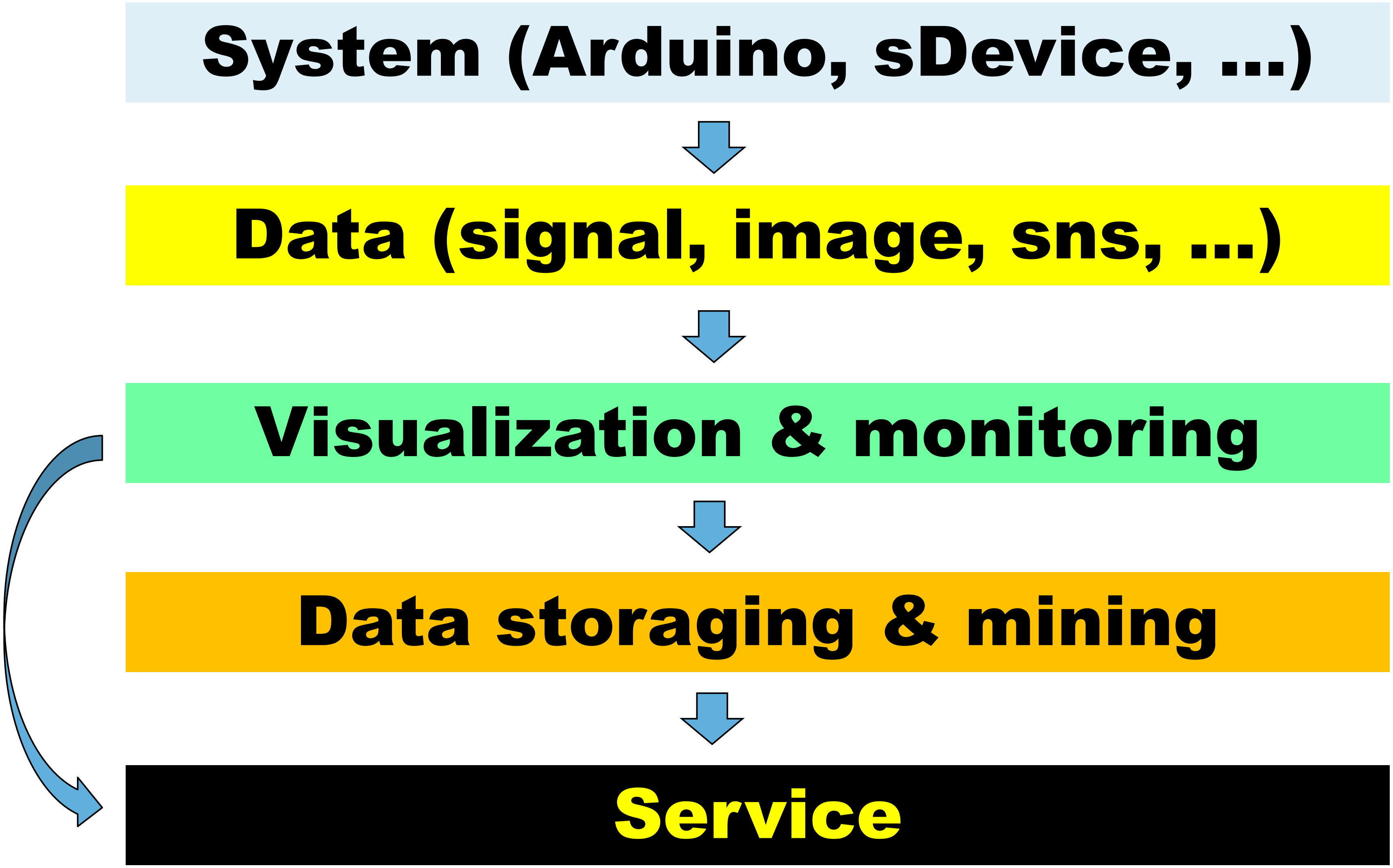
on Time: 2018-01-22 17:58:31.012

Data visualization using ploy.ly





A5. Introduction to visualization



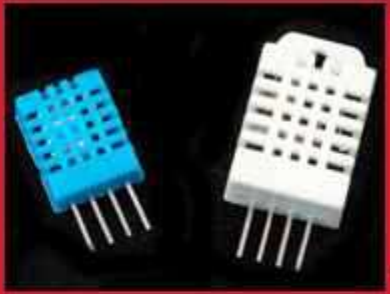
System (Arduino, sDevice, ...)

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

Visualization & monitoring

Data storaging & mining

Service



[Goal]

Arduino + Node.js

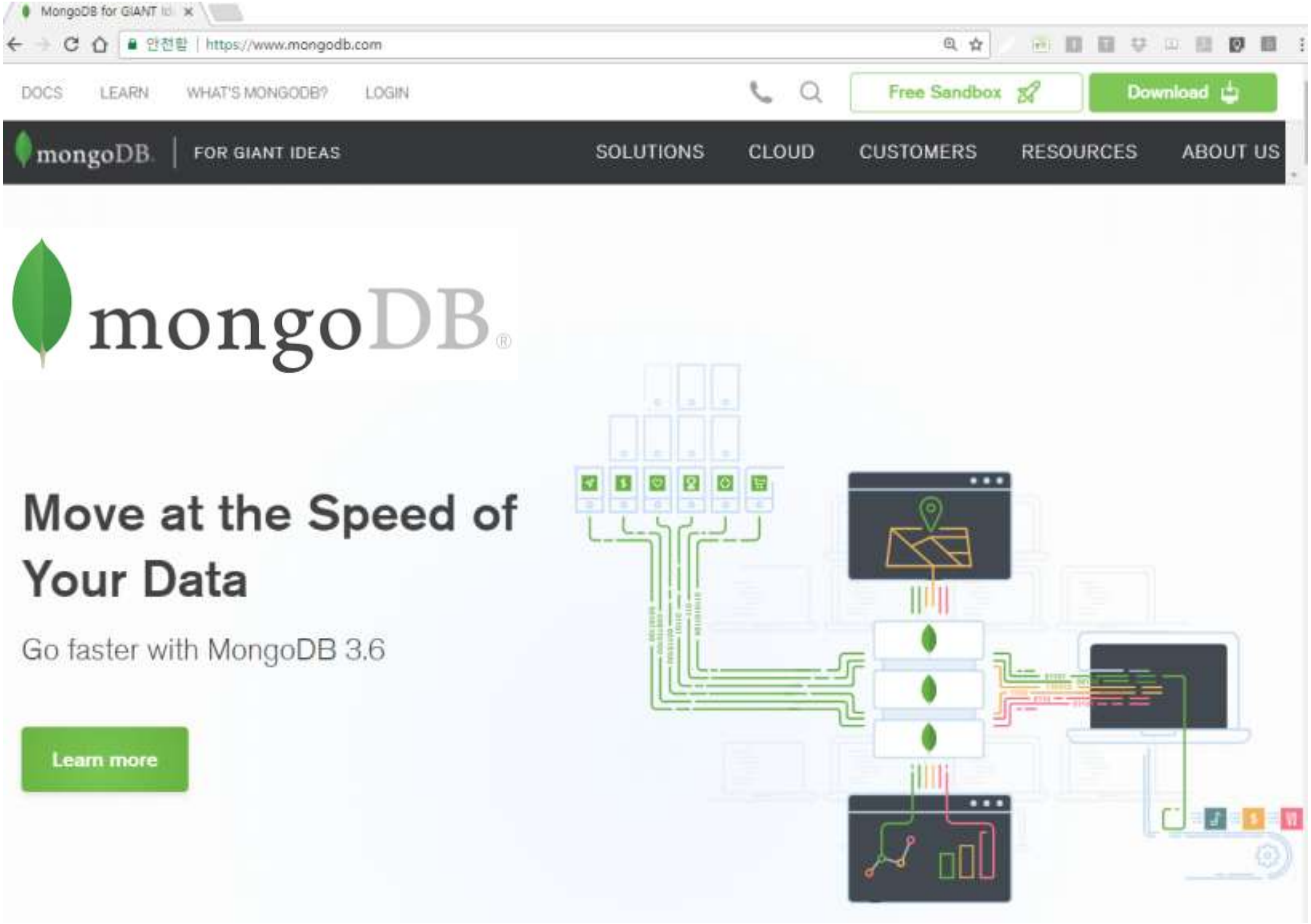
+ plotly.js

+ MongoDB

→ Data storaging

& visualization

A5.9 MongoDB



Move at the Speed of Your Data

Go faster with MongoDB 3.6

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 유지하고있습니다.

NoSQL?

흔히 NoSQL이라고 해서 아, SQL이 없는 데이터베이스구나! 라고 생각 할 수도 있겠지만, 진짜 의미는 Not Only SQL 입니다. 기존의 RDBMS의 한계를 극복하기 위해 만들어진 새로운 형태의 데이터저장소 입니다. 관계형 DB가 아니므로, RDMS처럼 고정된 스키마 및 JOIN 이 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 record 와 비슷한 개념인데요, 이의 데이터 구조는 한개이상의 key-value pair 으로 이뤄져있습니다. MongoDB 샘플 Document를 확인 해 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 document의 유일함(uniqueness)을 제공합니다.
이 값의 첫 4bytes 는현재 timestamp, 다음 3bytes는 machine id, 다음 2bytes는 MongoDB 서버의 프로세스id, 마지막 3bytes는 순차번호입니다 추가될때마다 값이 높아진다는거지요.

Document는 동적(dynamic)의 schema 를 갖고있습니다. 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 Document의 그룹입니다. Document들이 Collection내부에 위치하고 있습니다. RDMS의 table과 비슷한 개념입니다만 RDMS와 달리 schema를 따로 가지고 있지않습니다. Document 부분설명에 나와있듯이 각 Document들이 동적인 schema를 가지고 있으니까요

Database?

Database는 Collection들의 물리적인 컨테이너입니다. 각 Database는 파일시스템에 여러파일들로 저장됩니다.

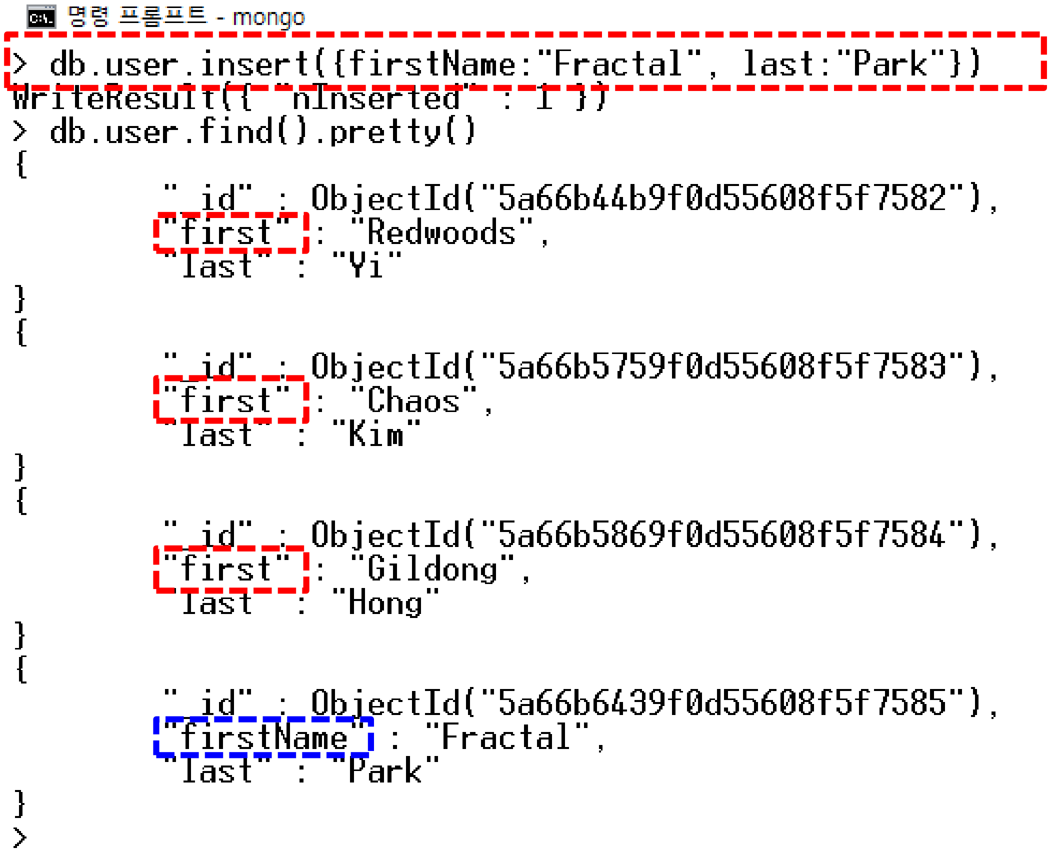
A5.9.3 MongoDB shell coding

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

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

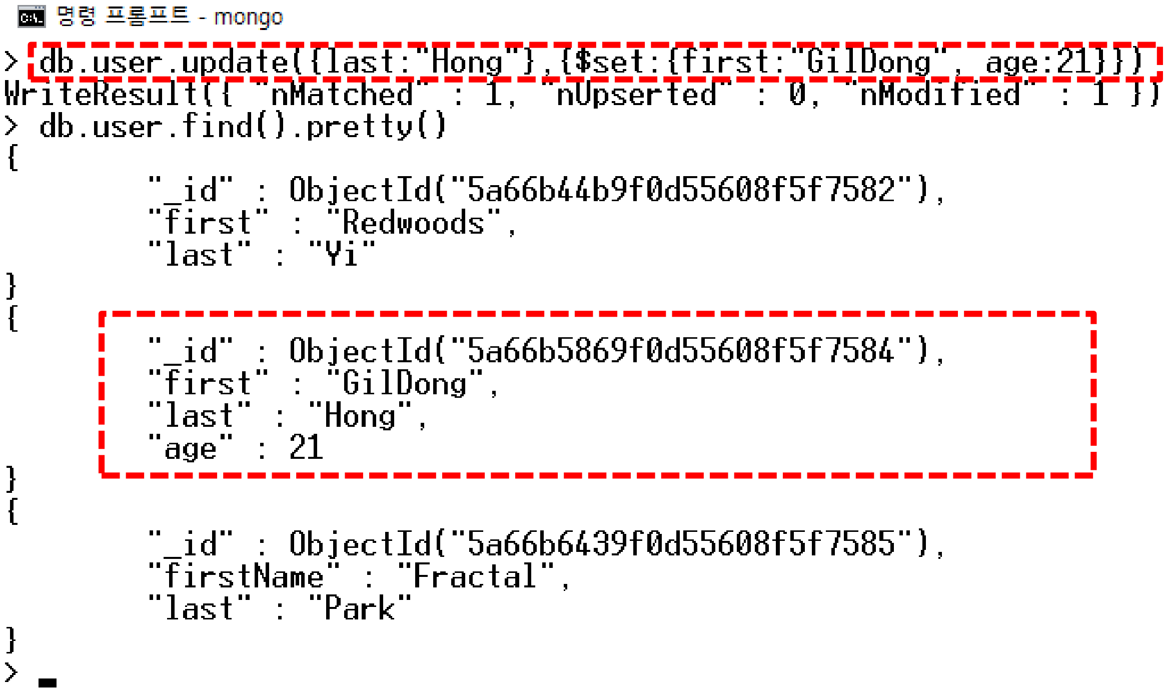
AAnn_mongo_schemas.png

A5.9.3 MongoDB shell coding

5. update a record

update record2

db.user.find().pretty()



```
> 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"
}
> _
```

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

Note that it is possible to change schema.

Save as

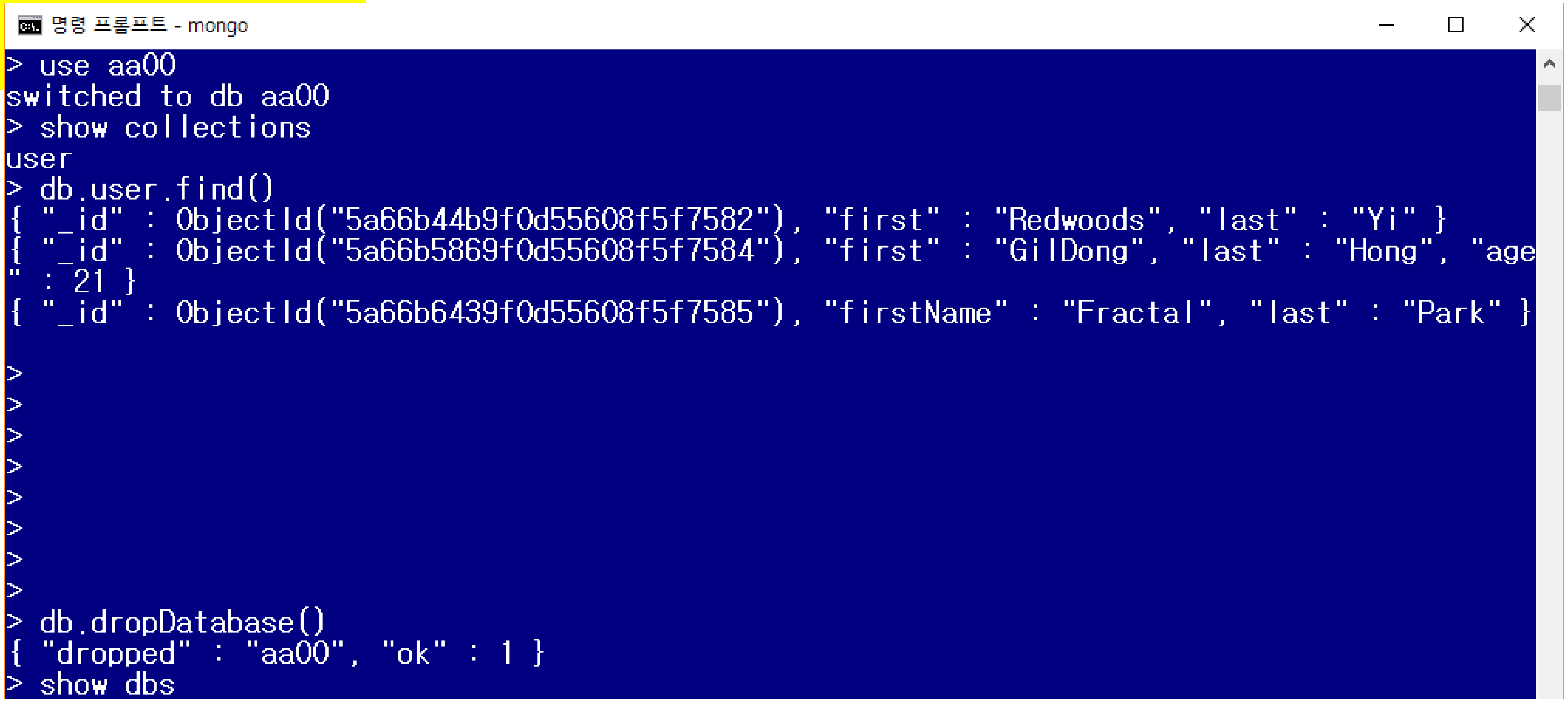
AAnn_mongo_update.png

A5.9.3 MongoDB shell coding

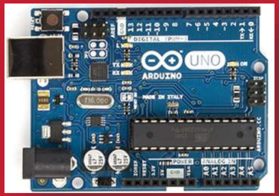
6. Delete(or remove) DB

use dbName

db.dropDatabase()



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

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

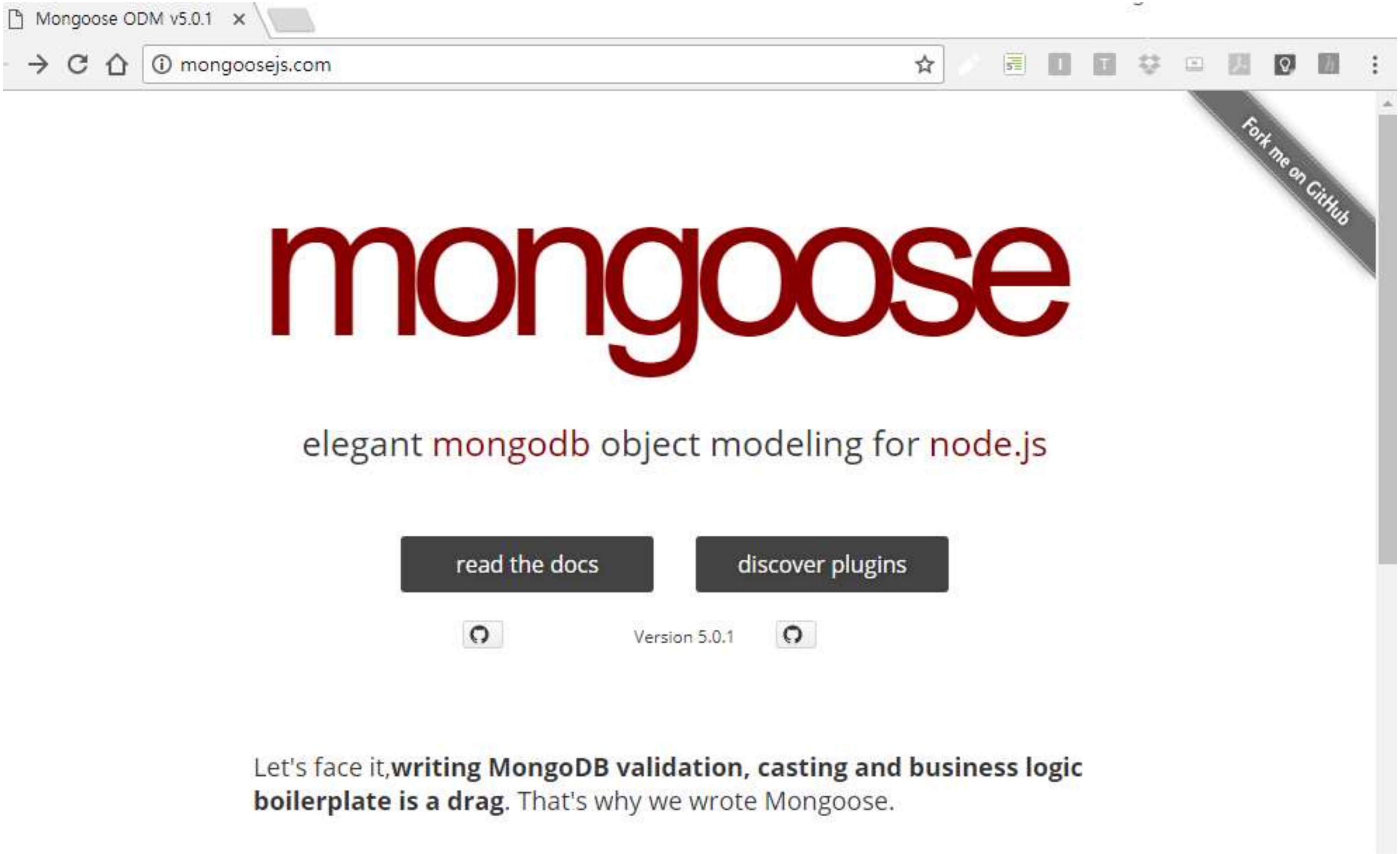


Node.js

+

MongoDB

A5.9.4 MongoDB + Node.js：mongoose



Mongoose ODM v5.0.1

mongoosejs.com

mongoose

elegant mongodb object modeling for node.js

Version 5.0.1

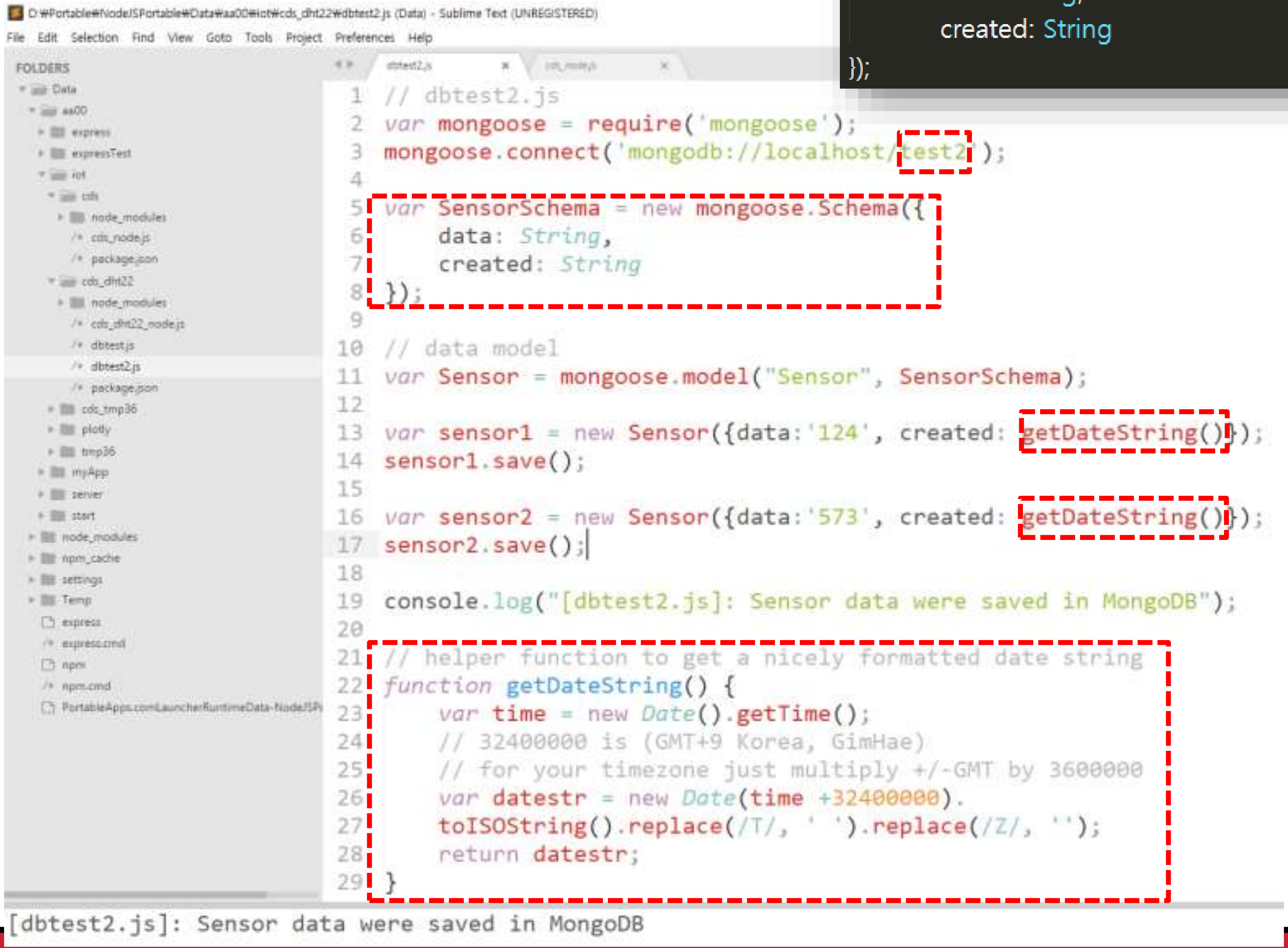
Let's face it,**writing MongoDB validation, casting and business logic boilerplate is a drag**. That's why we wrote Mongoose.

http://mongoosejs.com/

A5.9.4 MongoDB + Node.js : mongoose

4. dbtest2.js (use Sublime Text 3)

```
var SensorSchema = new mongoose.Schema({
        data: String,
        created: String
});
```



```
// dbtest2.js
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test2');

var SensorSchema = new mongoose.Schema({
    data: String,
    created: String
});

// data model
var Sensor = mongoose.model("Sensor", SensorSchema);

var sensor1 = new Sensor({data:'124', created: getDateString()});
sensor1.save();

var sensor2 = new Sensor({data:'573', created: getDateString()});
sensor2.save();

console.log("[dbtest2.js]: Sensor data were saved in MongoDB");

// helper function to get a nicely formatted date string
function getDateString() {
    var time = new Date().getTime();
    // 32400000 is (GMT+9 Korea, GimHae)
    // for your timezone just multiply +/-GMT by 3600000
    var datestr = new Date(time +32400000).
    toISOString().replace(/T/, ' ').replace(/Z/, '');
    return datestr;
}
```

```
[dbtest2.js]: Sensor data were saved in MongoDB
```

A5.9.4 MongoDB + Node.js : mongoose

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

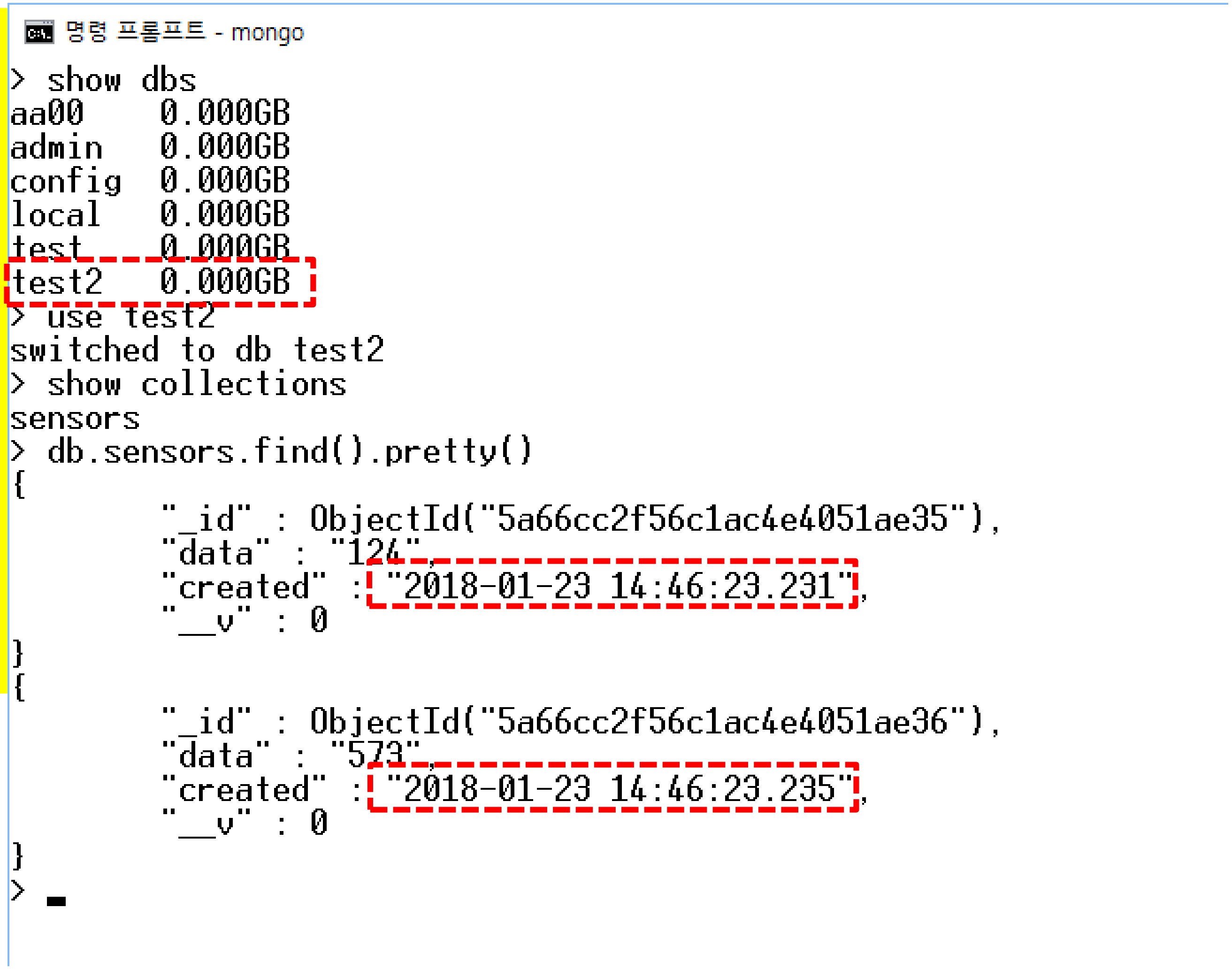
Mongo shell

> show dbs

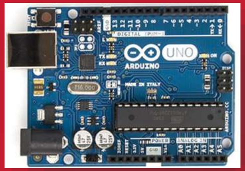
> use test2

> show collections

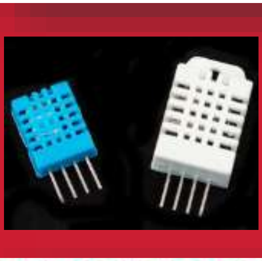
> db.sensors.find() .pretty()

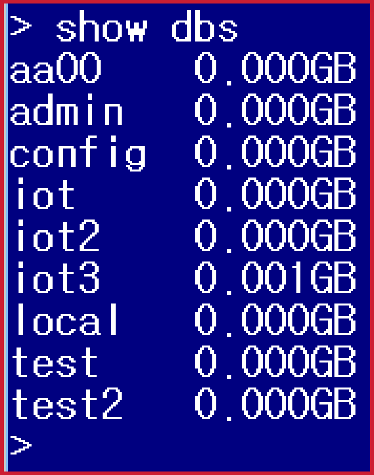


```
> show dbs
aa00     0.000GB
admin    0.000GB
config   0.000GB
local    0.000GB
test     0.000GB
test2    0.000GB
> use test2
switched to db test2
> show collections
sensors
> db.sensors.find().pretty()
{
        "_id" : ObjectId("5a66cc2f56c1ac4e4051ae35"),
        "data" : "124",
        "created" : "2018-01-23 14:46:23.231",
        "__v" : 0
}
{
        "_id" : ObjectId("5a66cc2f56c1ac4e4051ae36"),
        "data" : "573",
        "created" : "2018-01-23 14:46:23.235",
        "__v" : 0
}
> 
```

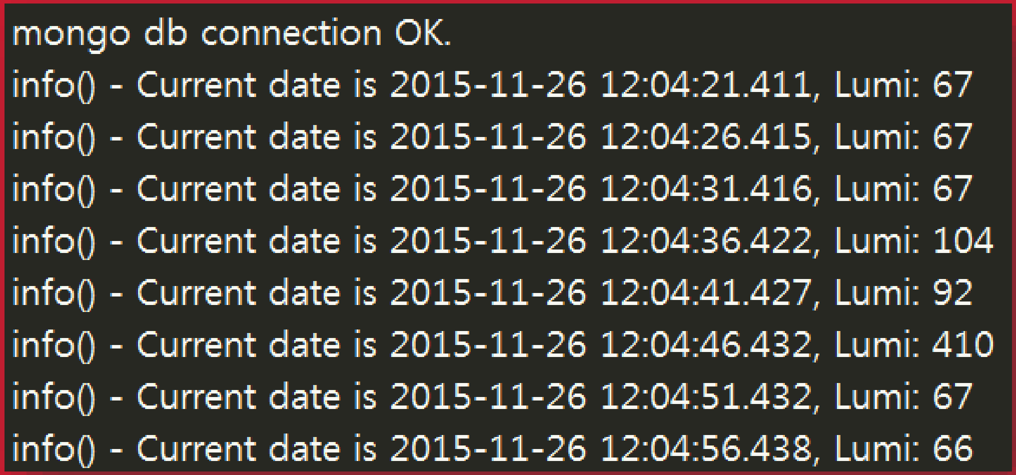


MongoDB from Arduino with node.js & mongoose



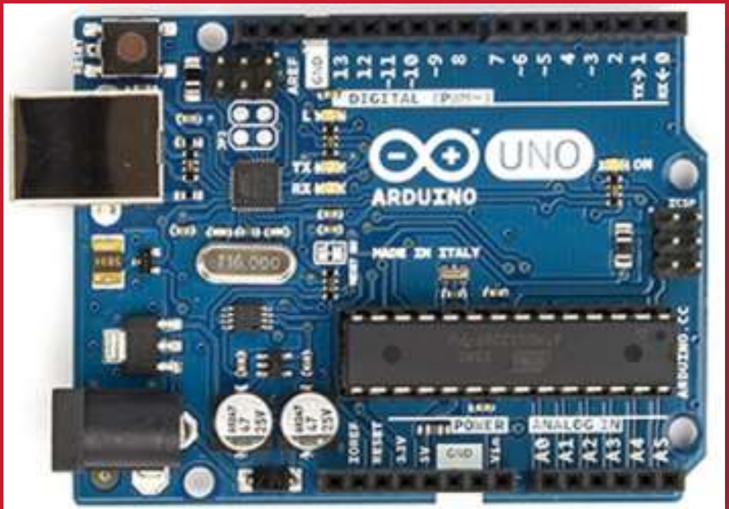


```
> show dbs
aa00     0.000GB
admin    0.000GB
config   0.000GB
iot      0.000GB
iot2     0.000GB
iot3     0.001GB
local    0.000GB
test     0.000GB
test2    0.000GB
>
```



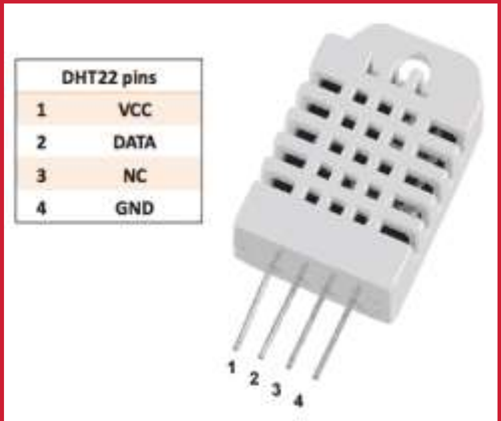
```
mongo db connection OK.
info() - Current date is 2015-11-26 12:04:21.411, Lumi: 67
info() - Current date is 2015-11-26 12:04:26.415, Lumi: 67
info() - Current date is 2015-11-26 12:04:31.416, Lumi: 67
info() - Current date is 2015-11-26 12:04:36.422, Lumi: 104
info() - Current date is 2015-11-26 12:04:41.427, Lumi: 92
info() - Current date is 2015-11-26 12:04:46.432, Lumi: 410
info() - Current date is 2015-11-26 12:04:51.432, Lumi: 67
info() - Current date is 2015-11-26 12:04:56.438, Lumi: 66
```

Arduino
& Node.js
& MongoDB



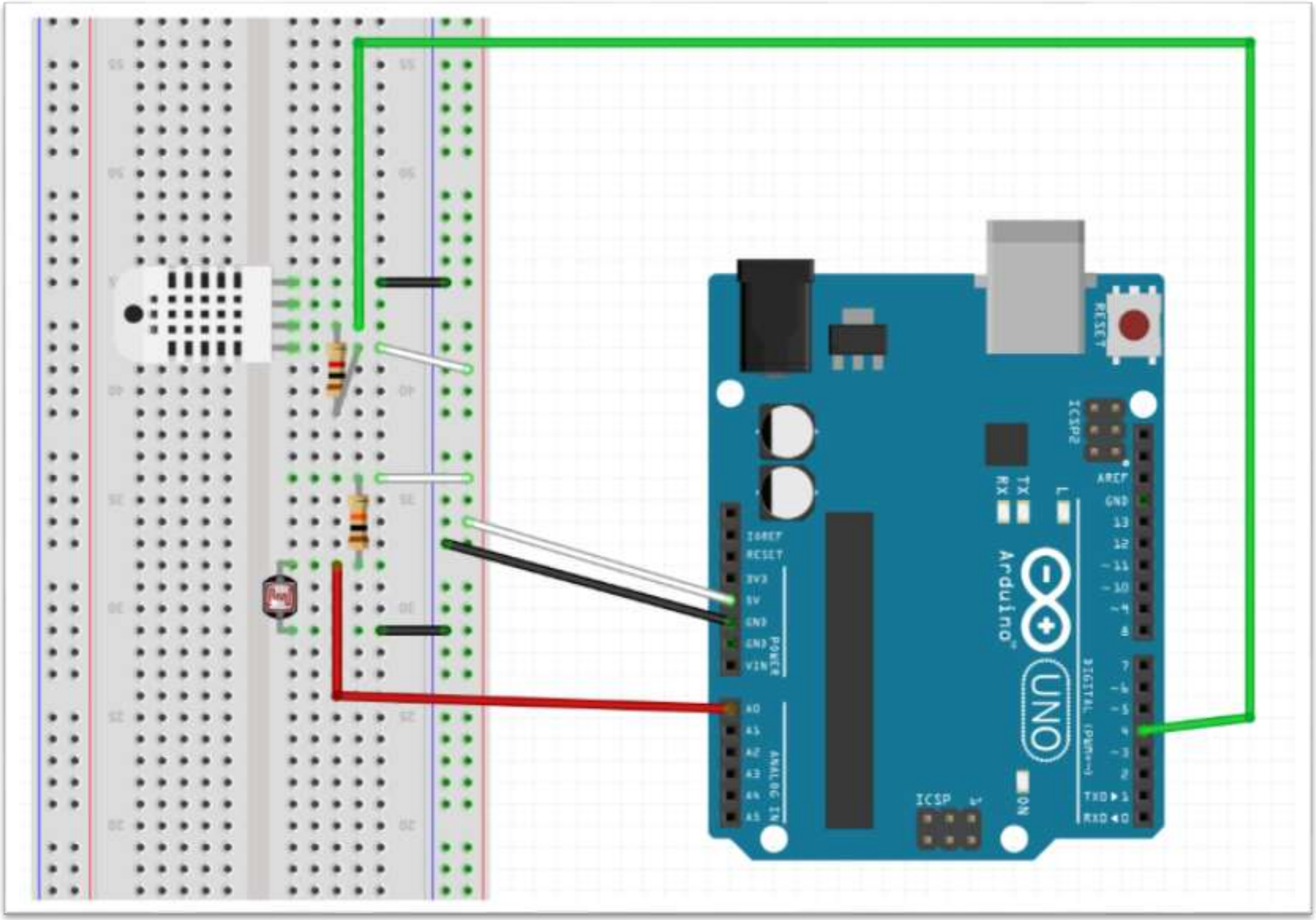
Multi-sensors

DHT22 + CdS

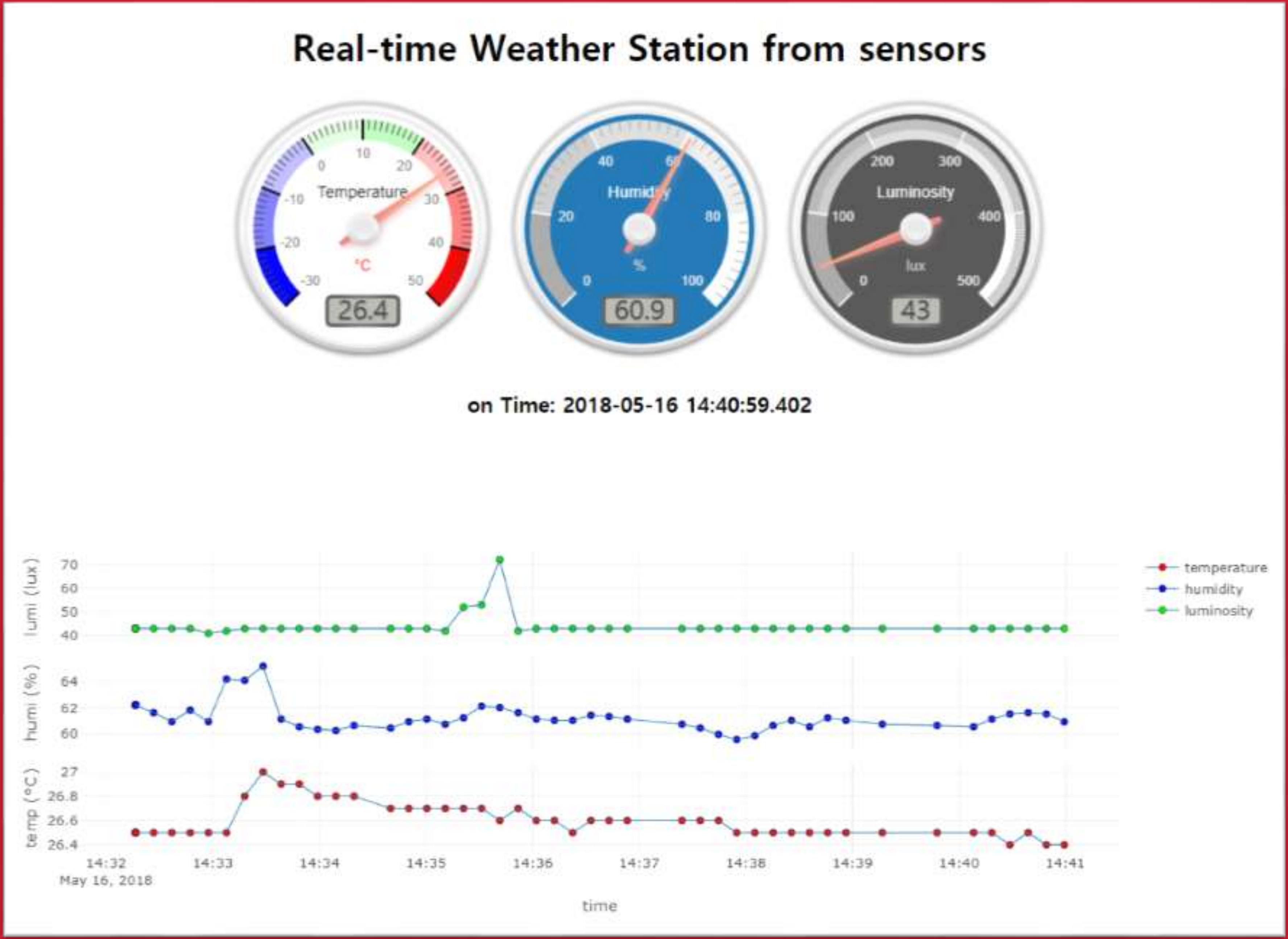


DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

DHT22 + CdS : circuit



client_CdS_DHT22.html

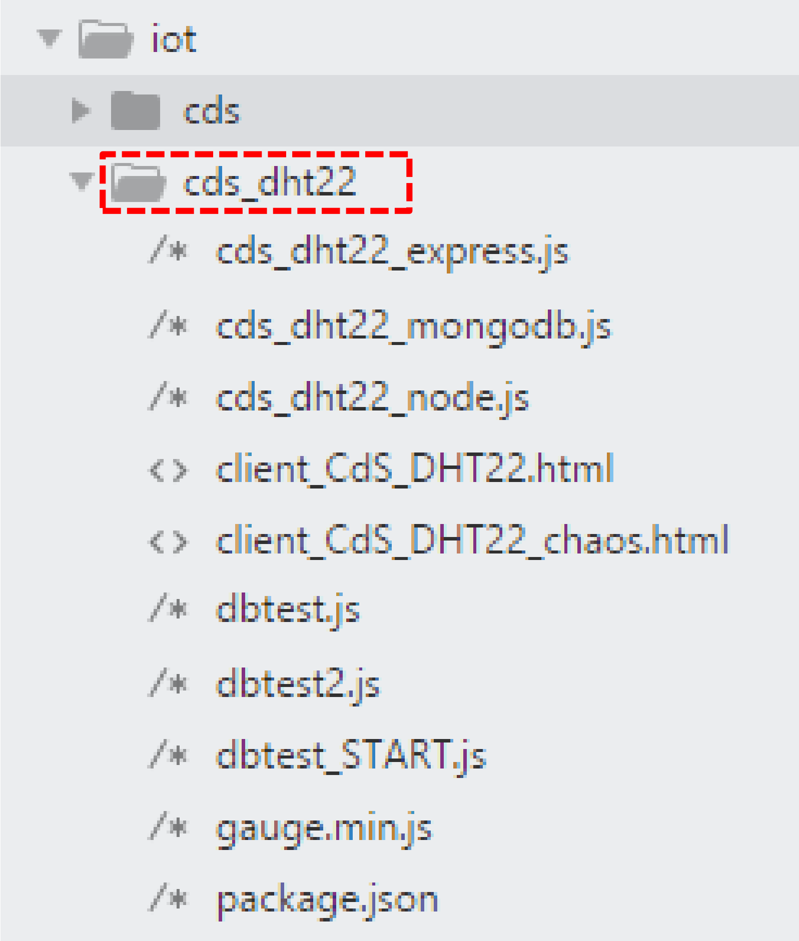


Real-time Weather Station from sensors

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

A5.9.5 DHT22 + CdS + Node.js + MongoDB

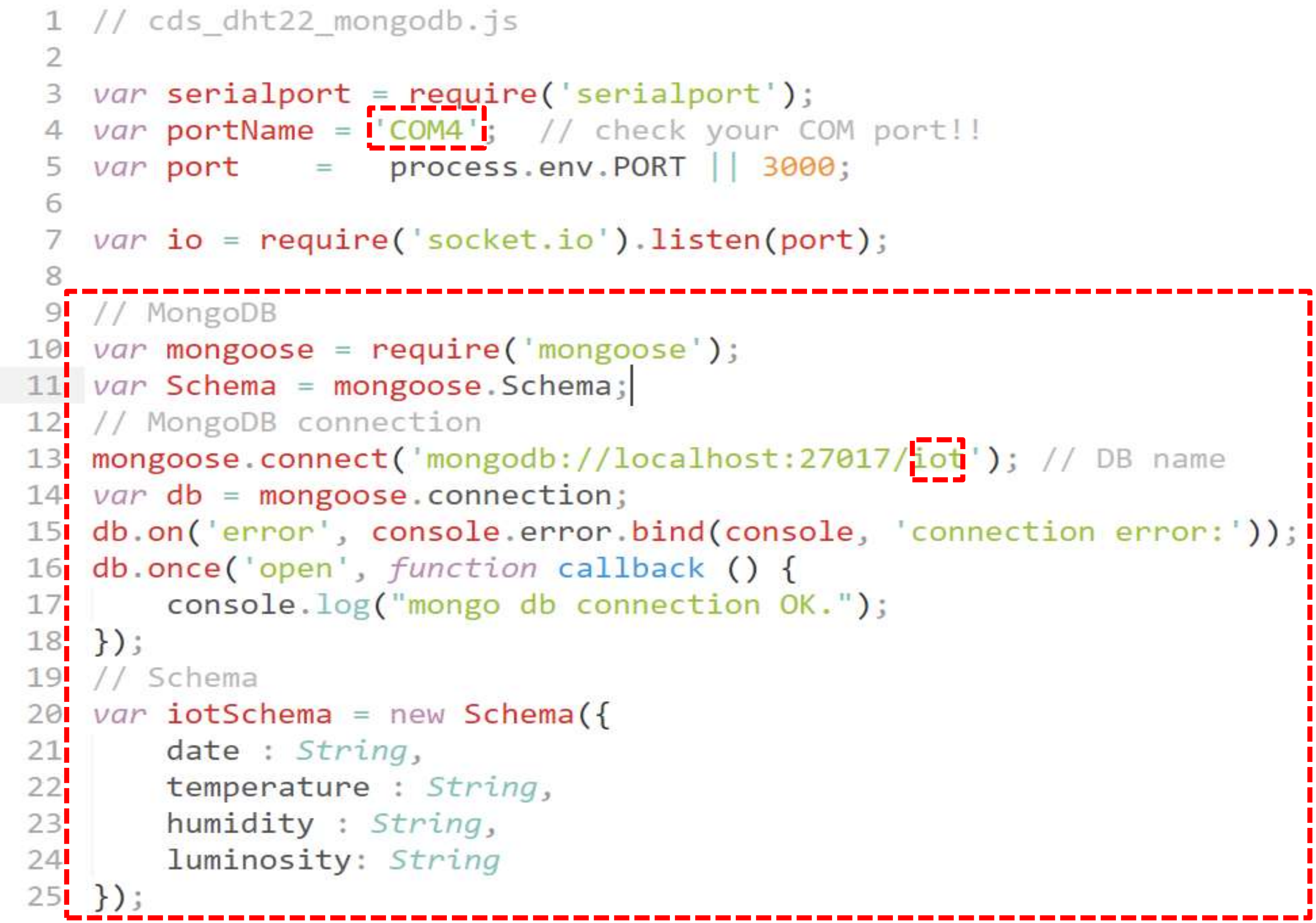
1. 작업 폴더 구조 [2018]



```
▼ iot
  ▶ cds
  ▼ cds_dht22
      cds_dht22_express.js
      cds_dht22_mongodb.js
      cds_dht22_node.js
      client_CdS_DHT22.html
      client_CdS_DHT22_chaos.html
      dbtest.js
      dbtest2.js
      dbtest_START.js
      gauge.min.js
      package.json
```

A5.9.5 DHT22 + CdS + Node.js + MongoDB

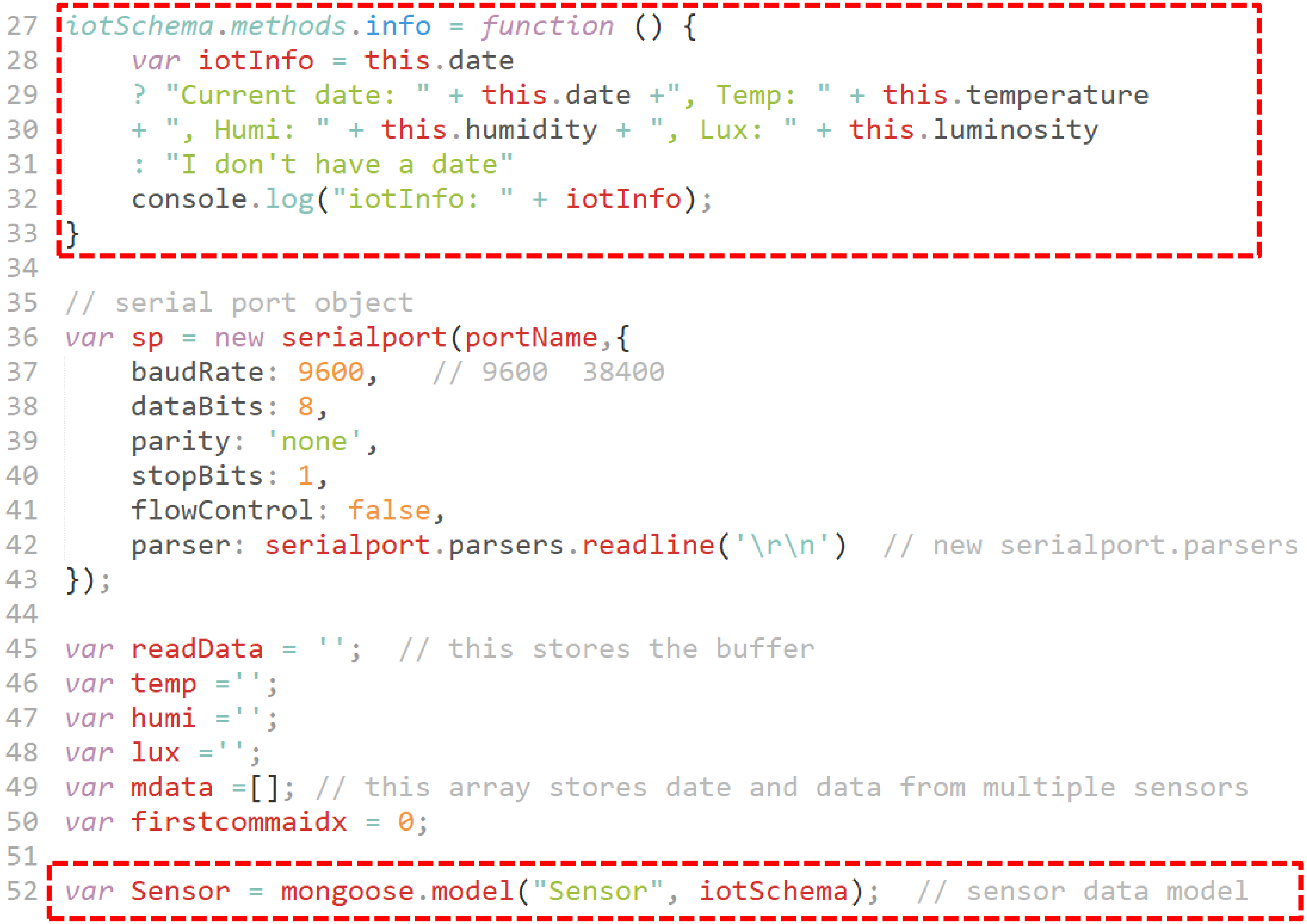
2.1 cds_dht22_mongodb.js



```javascript
// cds_dht22_mongodb.js

var serialport = require('serialport');
var portName = 'COM4';   // check your COM port!!
var port     =   process.env.PORT || 3000;

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

// MongoDB
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
// MongoDB connection
mongoose.connect('mongodb://localhost:27017/iot'); // DB name
var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function callback () {
    console.log("mongo db connection OK.");
});
// Schema
var iotSchema = new Schema({
    date : String,
    temperature : String,
    humidity : String,
    luminosity: String
});
```

A5.9.5 DHT22 + CdS + Node.js + MongoDB

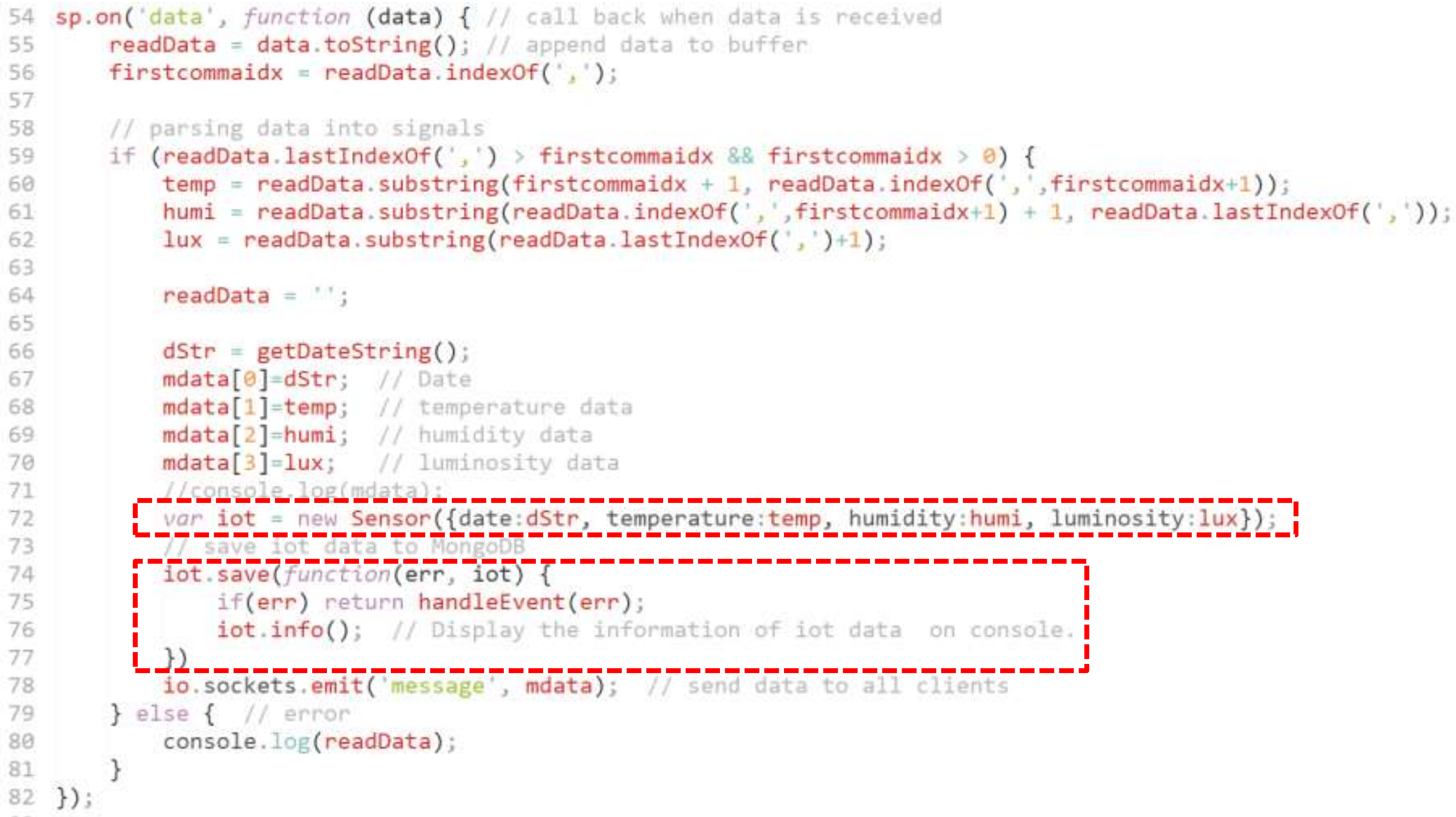
2.2 cds_dht22_mongodb.js



```
iotSchema.methods.info = function () {
    var iotInfo = this.date
    ? "Current date: " + this.date +", Temp: " + this.temperature
    + ", Humi: " + this.humidity + ", Lux: " + this.luminosity
    : "I don't have a date"
    console.log("iotInfo: " + iotInfo);
}

// serial port object
var sp = new serialport(portName,{
    baudRate: 9600,   // 9600  38400
    dataBits: 8,
    parity: 'none',
    stopBits: 1,
    flowControl: false,
    parser: serialport.parsers.readline('\r\n')  // new serialport.parsers
});

var readData = '';  // this stores the buffer
var temp ='';
var humi ='';
var lux ='';
var mdata =[]; // this array stores date and data from multiple sensors
var firstcommaidx = 0;

var Sensor = mongoose.model("Sensor", iotSchema);  // sensor data model
```

A5.9.5 DHT22 + CdS + Node.js + MongoDB

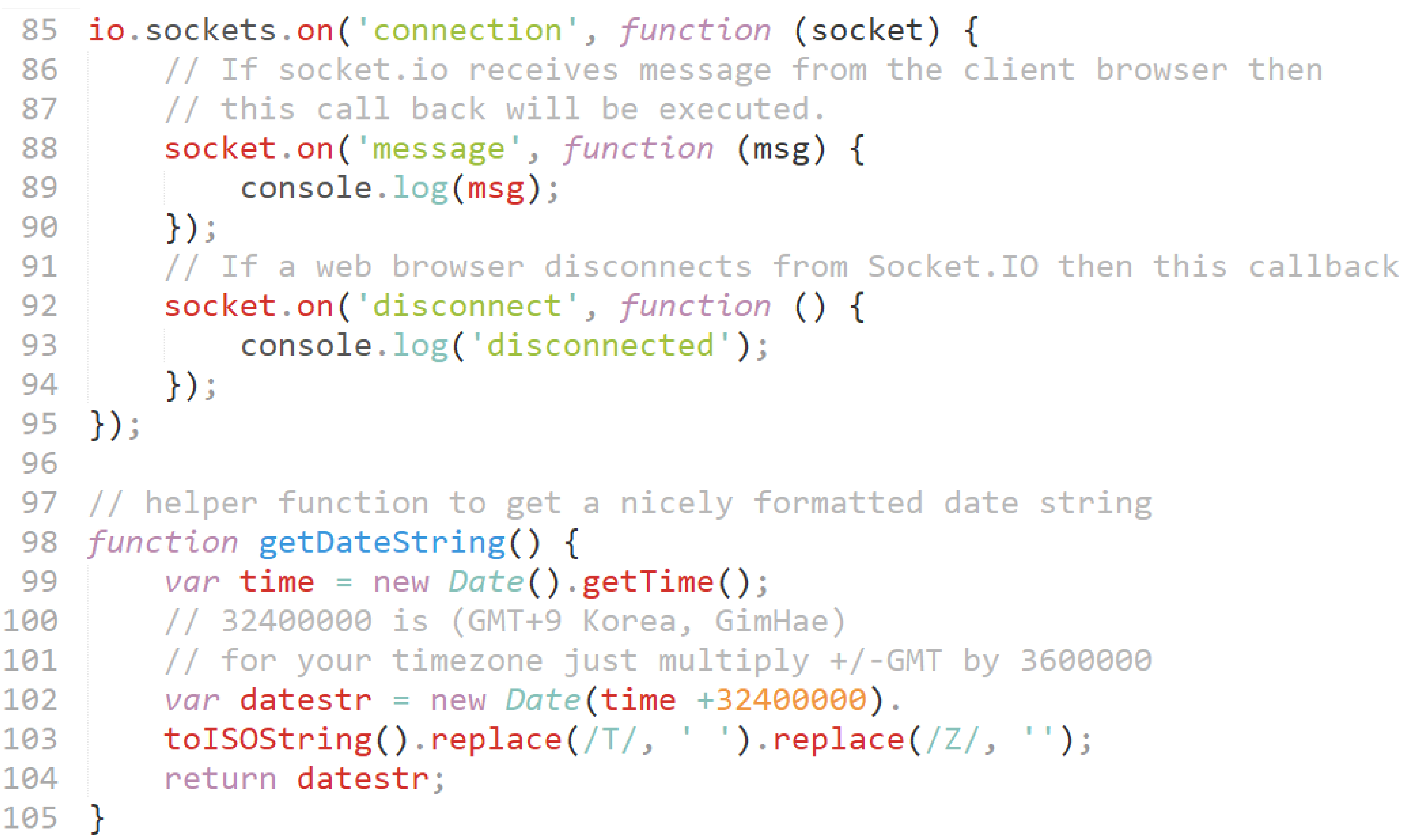
2.3 cds_dht22_mongodb.js



```
sp.on('data', function (data) { // call back when data is received
    readData = data.toString(); // append data to buffer
    firstcommaidx = readData.indexOf(',');

    // parsing data into signals
    if (readData.lastIndexOf(',') > firstcommaidx && firstcommaidx > 0) {
        temp = readData.substring(firstcommaidx + 1, readData.indexOf(',',firstcommaidx+1));
        humi = readData.substring(readData.indexOf(',',firstcommaidx+1) + 1, readData.lastIndexOf(','));
        lux = readData.substring(readData.lastIndexOf(',')+1);

        readData = '';

        dStr = getDateString();
        mdata[0]=dStr;  // Date
        mdata[1]=temp;  // temperature data
        mdata[2]=humi;  // humidity data
        mdata[3]=lux;   // luminosity data
        //console.log(mdata);
        var iot = new Sensor({date:dStr, temperature:temp, humidity:humi, luminosity:lux});
        // save iot data to MongoDB
        iot.save(function(err, iot) {
            if(err) return handleEvent(err);
            iot.info();  // Display the information of iot data  on console.
        })
        io.sockets.emit('message', mdata);  // send data to all clients
    } else {  // error
        console.log(readData);
    }
});
```

A5.9.5 DHT22 + CdS + Node.js + MongoDB

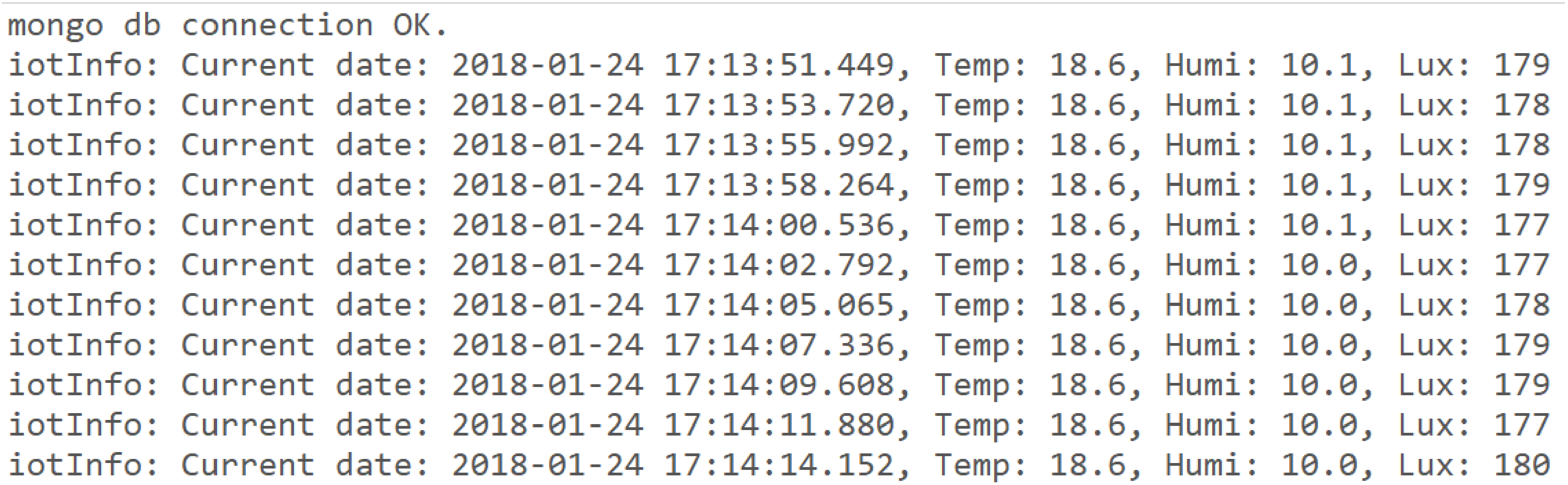
2.4 cds_dht22_mongodb.js



```
io.sockets.on('connection', function (socket) {
    // If socket.io receives message from the client browser then
    // this call back will be executed.
    socket.on('message', function (msg) {
        console.log(msg);
    });
    // If a web browser disconnects from Socket.IO then this callback
    socket.on('disconnect', function () {
        console.log('disconnected');
    });
});

// helper function to get a nicely formatted date string
function getDateString() {
    var time = new Date().getTime();
    // 32400000 is (GMT+9 Korea, GimHae)
    // for your timezone just multiply +/-GMT by 3600000
    var datestr = new Date(time +32400000).
    toISOString().replace(/T/, ' ').replace(/Z/, '');
    return datestr;
}
```

A5.9.5 DHT22 + CdS + Node.js + MongoDB

2.5 cds_dht22_mongodb.js → result (^B)



```
mongo db connection OK.
iotInfo: Current date: 2018-01-24 17:13:51.449, Temp: 18.6, Humi: 10.1, Lux: 179
iotInfo: Current date: 2018-01-24 17:13:53.720, Temp: 18.6, Humi: 10.1, Lux: 178
iotInfo: Current date: 2018-01-24 17:13:55.992, Temp: 18.6, Humi: 10.1, Lux: 178
iotInfo: Current date: 2018-01-24 17:13:58.264, Temp: 18.6, Humi: 10.1, Lux: 179
iotInfo: Current date: 2018-01-24 17:14:00.536, Temp: 18.6, Humi: 10.1, Lux: 177
iotInfo: Current date: 2018-01-24 17:14:02.792, Temp: 18.6, Humi: 10.0, Lux: 177
iotInfo: Current date: 2018-01-24 17:14:05.065, Temp: 18.6, Humi: 10.0, Lux: 178
iotInfo: Current date: 2018-01-24 17:14:07.336, Temp: 18.6, Humi: 10.0, Lux: 179
iotInfo: Current date: 2018-01-24 17:14:09.608, Temp: 18.6, Humi: 10.0, Lux: 179
iotInfo: Current date: 2018-01-24 17:14:11.880, Temp: 18.6, Humi: 10.0, Lux: 177
iotInfo: Current date: 2018-01-24 17:14:14.152, Temp: 18.6, Humi: 10.0, Lux: 180
```

A5.9.5 DHT22 + CdS + Node.js + MongoDB

3. cds_dht22_mongodb.js → Check documents in Mongo shell

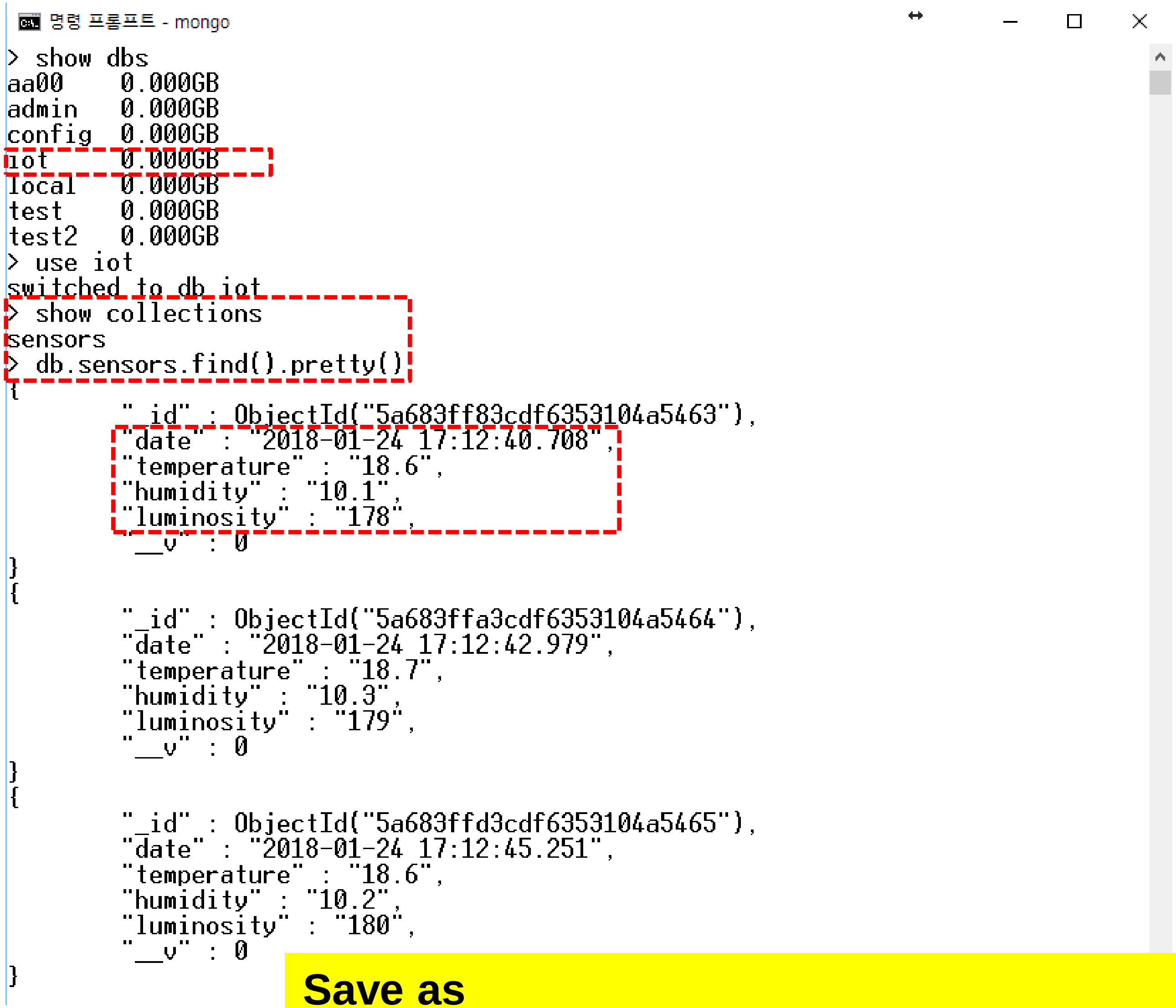
Mongo shell

> show dbs

> use iot

> show collections

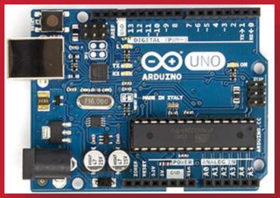
**> db.sensors.find()
.pretty()**



```
> show dbs
aa00     0.000GB
admin    0.000GB
config   0.000GB
iot      0.000GB
local    0.000GB
test     0.000GB
test2    0.000GB
> use iot
switched to db iot
> show collections
sensors
> db.sensors.find().pretty()
{
        "_id" : ObjectId("5a683ff83cdf6353104a5463"),
        "date" : "2018-01-24 17:12:40.708",
        "temperature" : "18.6",
        "humidity" : "10.1",
        "luminosity" : "178",
        "__v" : 0
}
{
        "_id" : ObjectId("5a683ffa3cdf6353104a5464"),
        "date" : "2018-01-24 17:12:42.979",
        "temperature" : "18.7",
        "humidity" : "10.3",
        "luminosity" : "179",
        "__v" : 0
}
{
        "_id" : ObjectId("5a683ffd3cdf6353104a5465"),
        "date" : "2018-01-24 17:12:45.251",
        "temperature" : "18.6",
        "humidity" : "10.2",
        "luminosity" : "180",
        "__v" : 0
}
```

Save as

AAnn_iot_mongdb.png



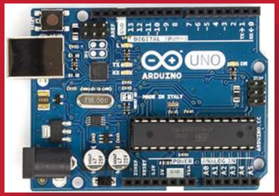
Arduino

& Node.js

& MongoDB

& Express server

NODE JS



3-servers

mongoDB

socket.io

3000

node js

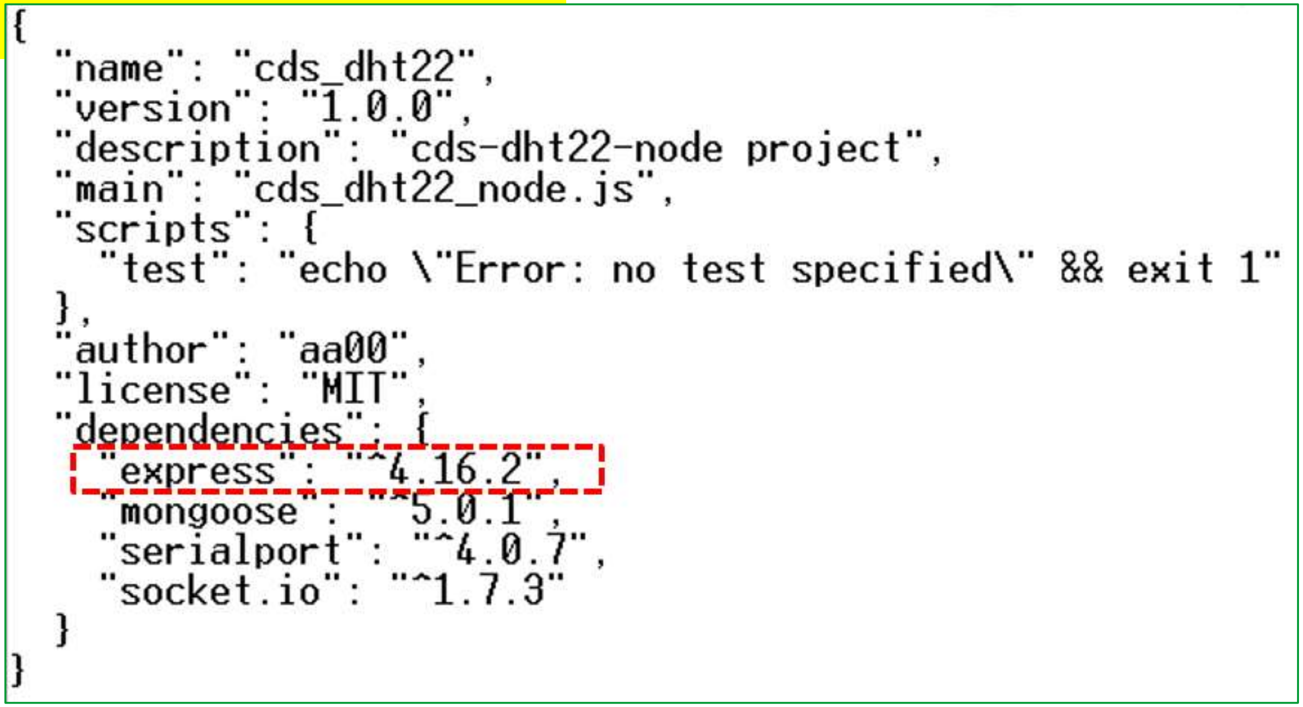
express

3030

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

1. Install express server

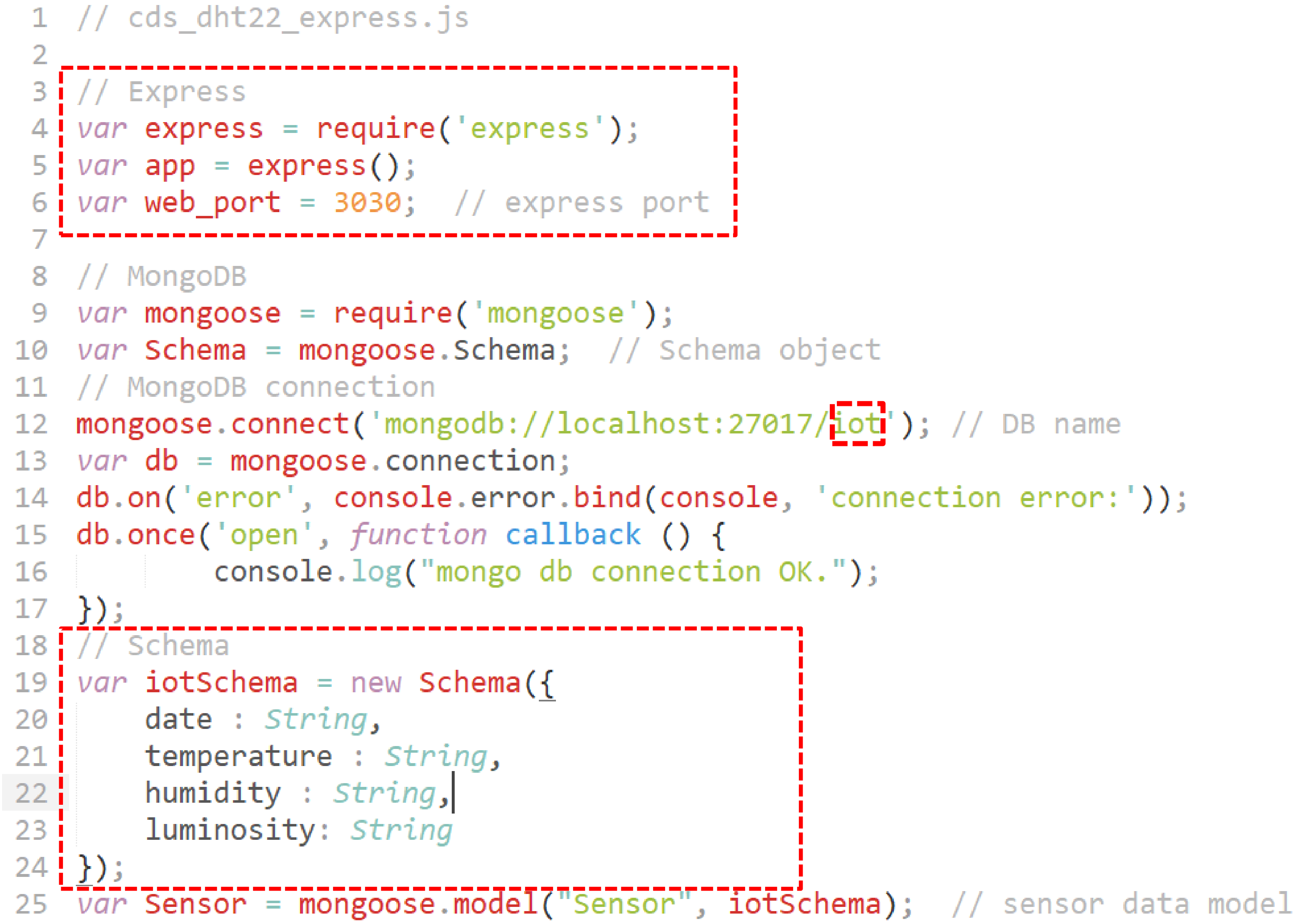
- Go to cds_dht22 project
- npm install --save express
- package.json



```
{
  "name": "cds_dht22",
  "version": "1.0.0",
  "description": "cds-dht22-node project",
  "main": "cds_dht22_node.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "aa00",
  "license": "MIT",
  "dependencies": {
    "express": "^4.16.2",
    "mongoose": "^5.0.1",
    "serialport": "^4.0.7",
    "socket.io": "^1.7.3"
  }
}
```

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

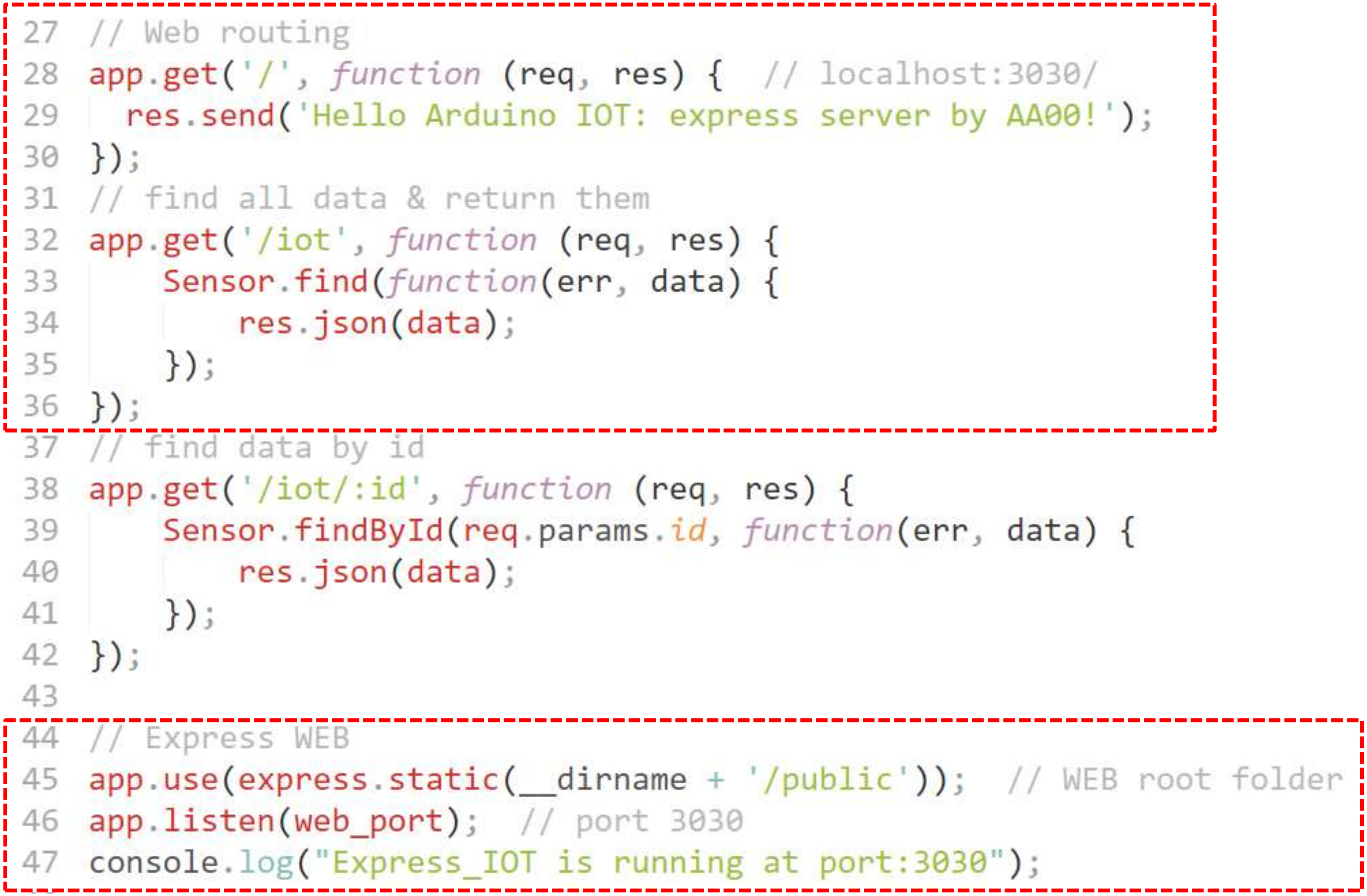
2.1 cds_dht22_express.js



```
// cds_dht22_express.js

// Express
var express = require('express');
var app = express();
var web_port = 3030;  // express port

// MongoDB
var mongoose = require('mongoose');
var Schema = mongoose.Schema;  // Schema object
// MongoDB connection
mongoose.connect('mongodb://localhost:27017/iot'); // DB name
var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function callback () {
        console.log("mongo db connection OK.");
});
// Schema
var iotSchema = new Schema({
    date : String,
    temperature : String,
    humidity : String,
    luminosity: String
});
var Sensor = mongoose.model("Sensor", iotSchema);  // sensor data model
```

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

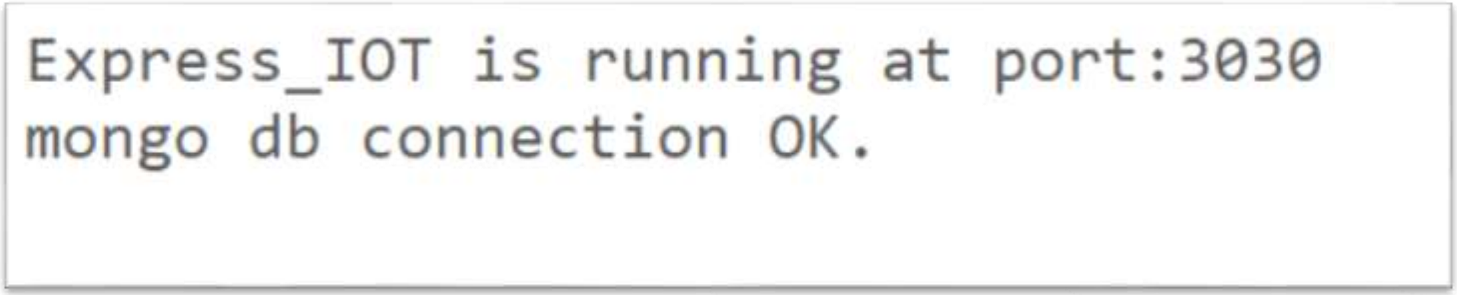
2.2 cds_dht22_express.js



```
// Web routing
app.get('/', function (req, res) {  // localhost:3030/
  res.send('Hello Arduino IOT: express server by AA00!');
});
// find all data & return them
app.get('/iot', function (req, res) {
    Sensor.find(function(err, data) {
        res.json(data);
    });
});
// find data by id
app.get('/iot/:id', function (req, res) {
    Sensor.findById(req.params.id, function(err, data) {
        res.json(data);
    });
});

// Express WEB
app.use(express.static(__dirname + '/public'));  // WEB root folder
app.listen(web_port);  // port 3030
console.log("Express_IOT is running at port:3030");
```

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

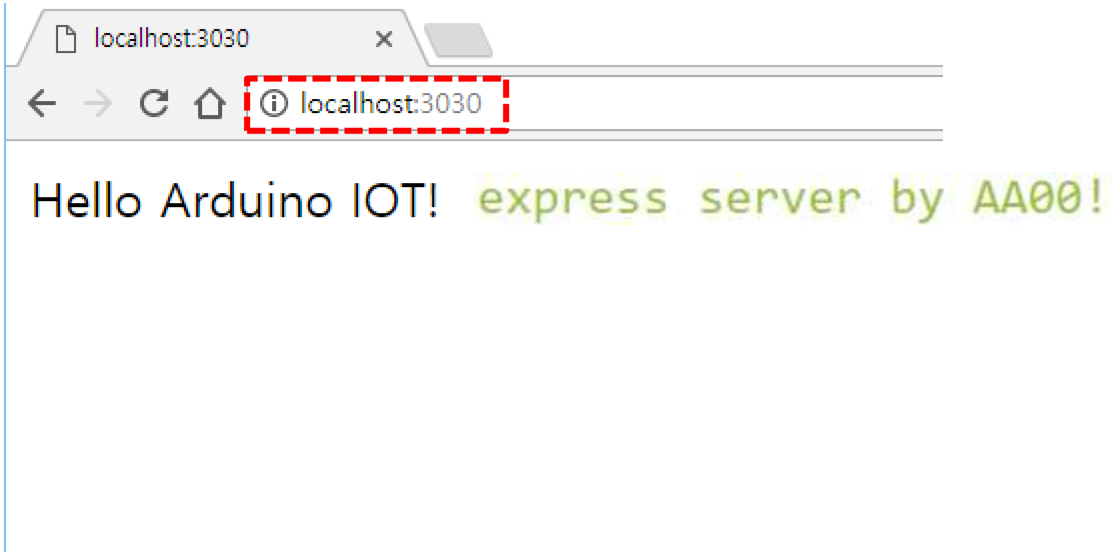
2.3 cds_dht22_express.js → Run



```
Express_IOT is running at port:3030
mongo db connection OK.
```

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

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

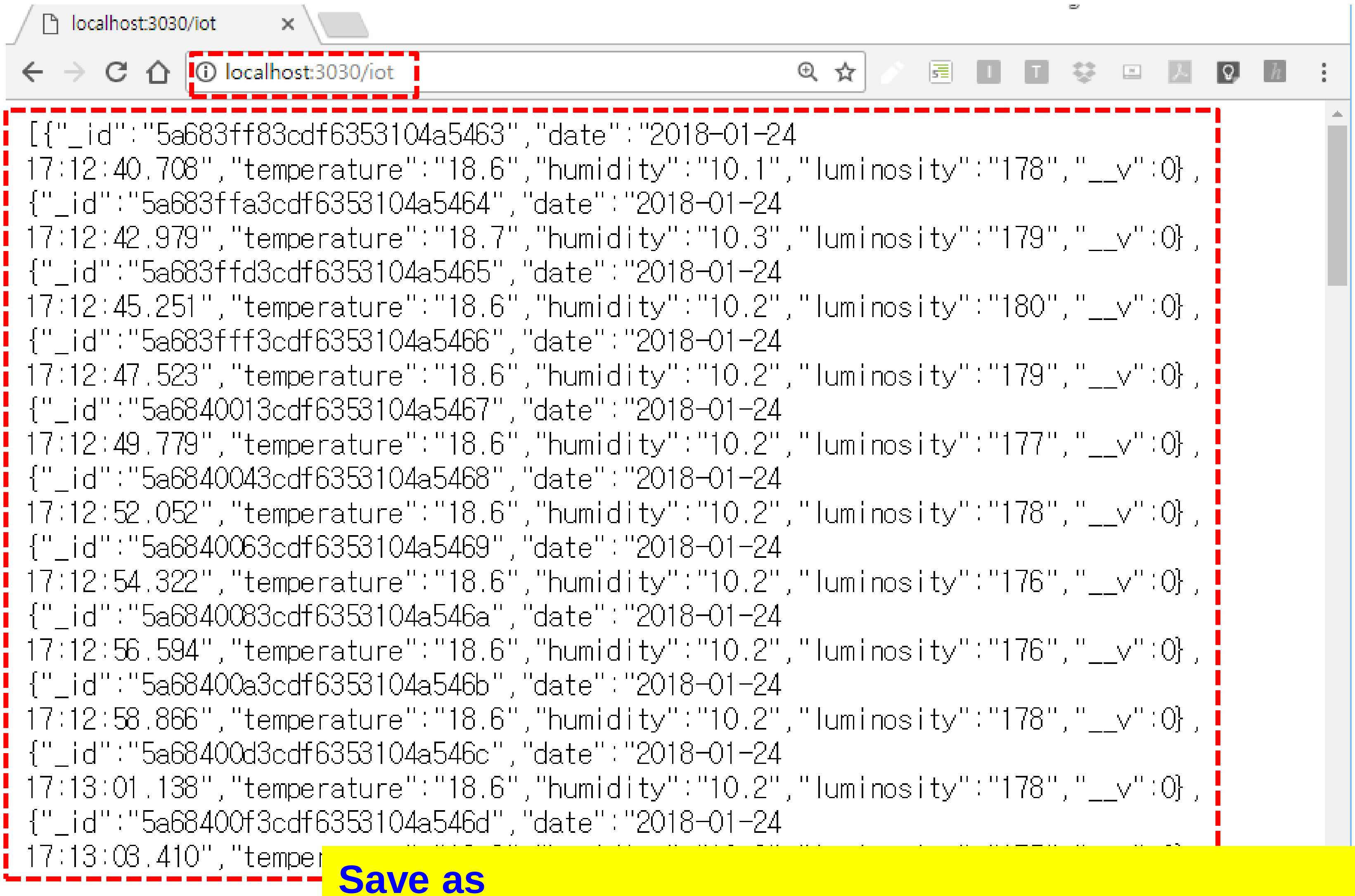


localhost:3030

Hello Arduino IOT! express server by AA00!

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

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



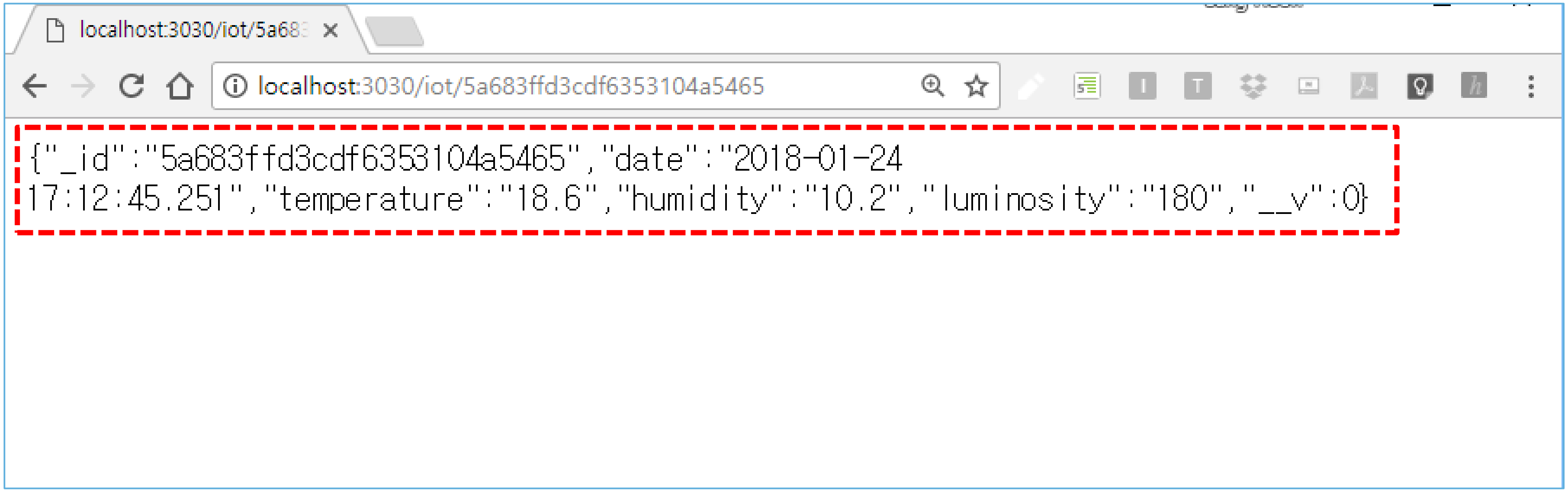
```
[{"_id":"5a683ff83cdf6353104a5463","date":"2018-01-24
17:12:40.708","temperature":"18.6","humidity":"10.1","luminosity":"178","__v":0},
{"_id":"5a683ffa3cdf6353104a5464","date":"2018-01-24
17:12:42.979","temperature":"18.7","humidity":"10.3","luminosity":"179","__v":0},
{"_id":"5a683ffd3cdf6353104a5465","date":"2018-01-24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

Save as

AAnn_iot_mongodb_web.png

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

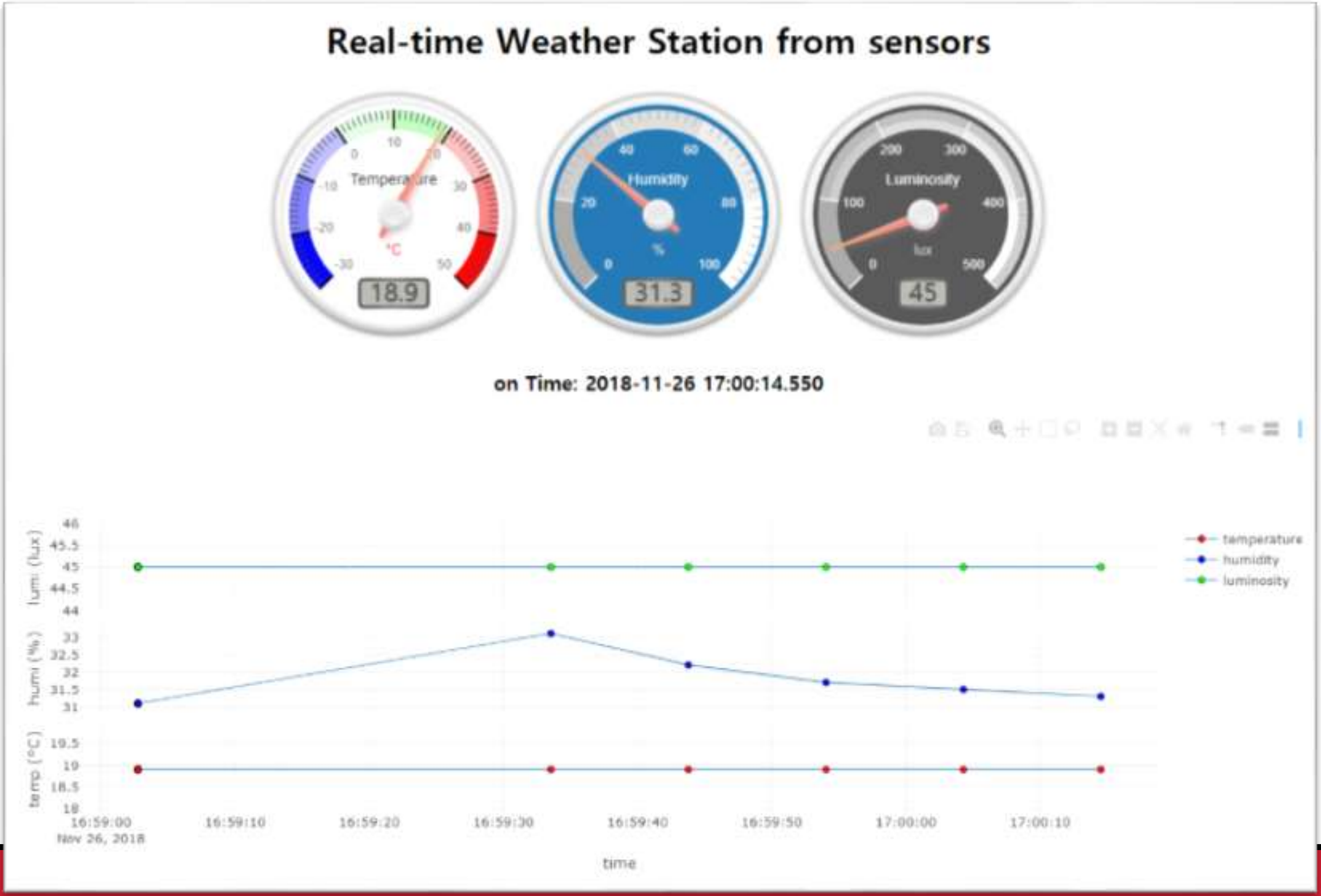
2.6 cds_dht22_express.js → routing3 http://localhost:3030/iot:id



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

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

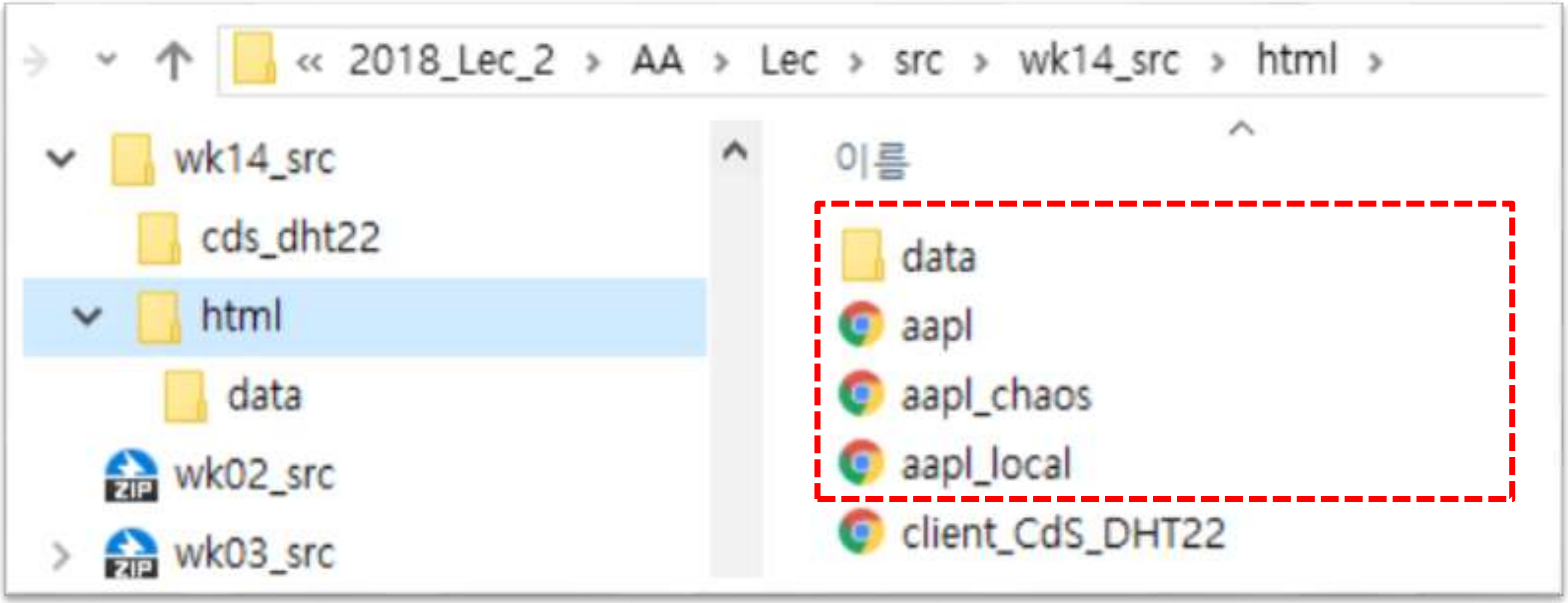
2.7 copy cds_dht22_client.html & gauge.min.js → ./public/ subfolder
http://localhost:3030/cds_dht22_client.html (web root folder)



Real-time Weather Station from sensors

on Time: 2018-11-26 17:00:14.550

2.8 CORS bug (Cross Origin Resource Sharing)



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

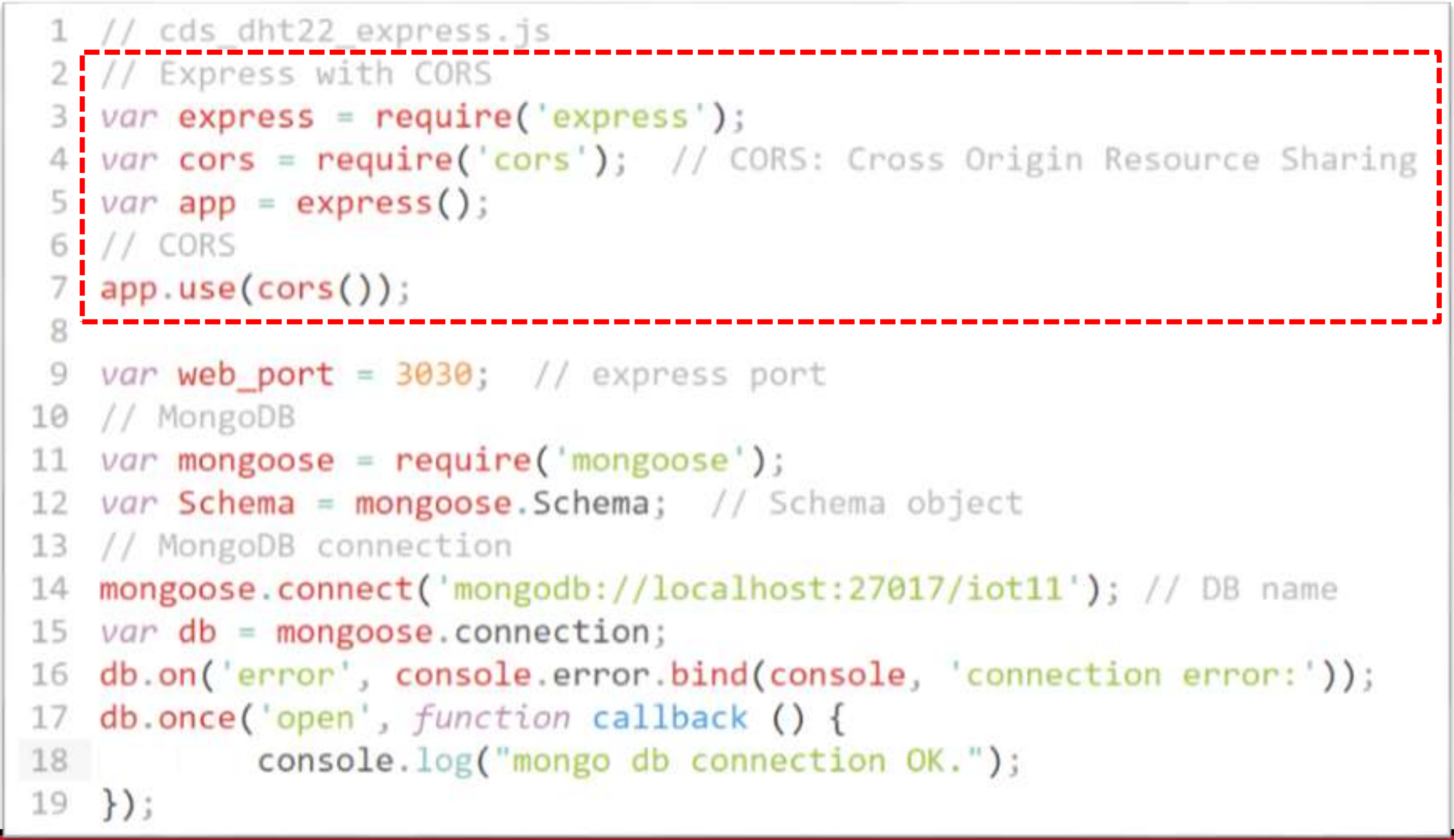
→ Local file에 접근 불허

→ CORS problem

→ public 폴더로 html,data를 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 2.84 이상)
npm install –save cors

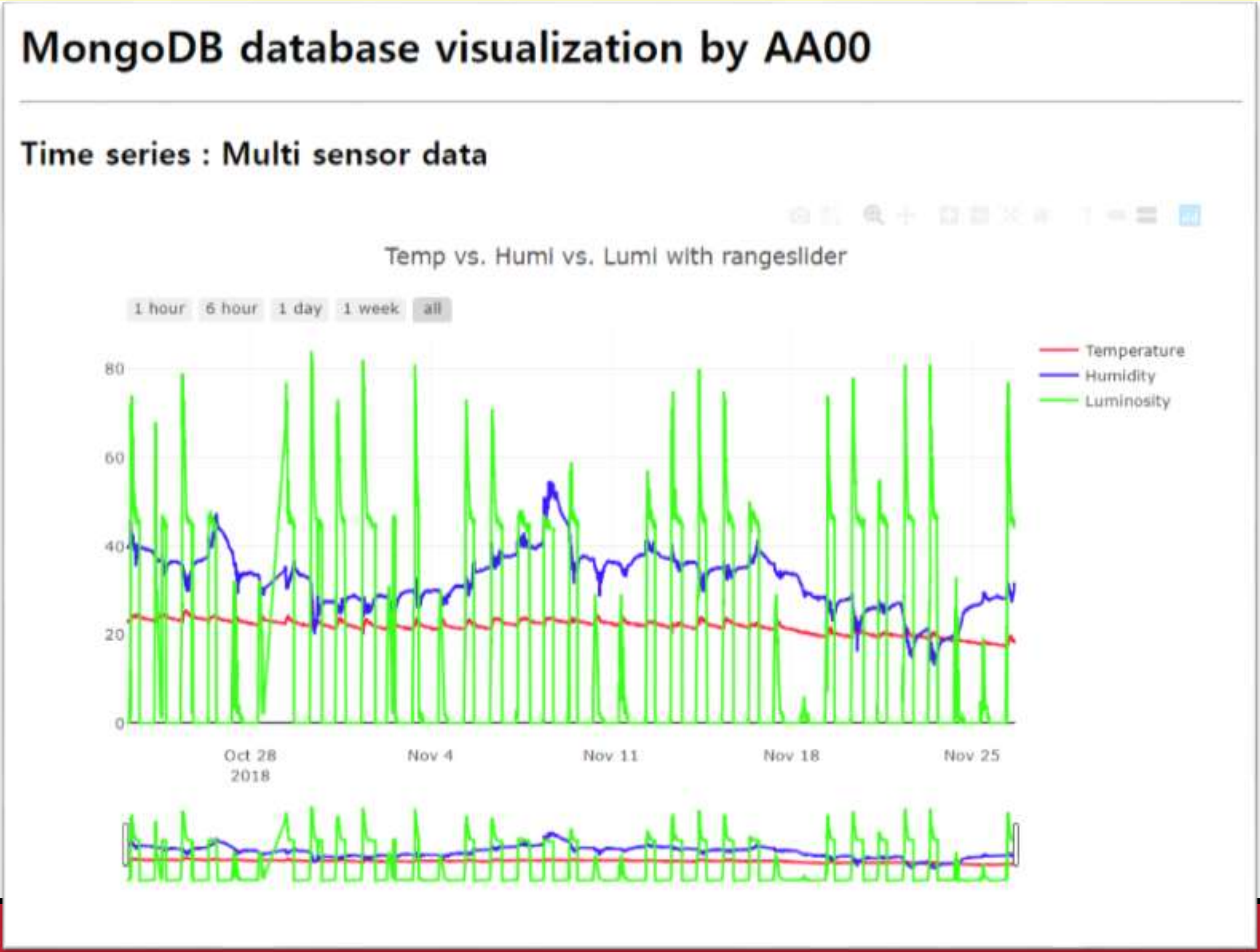


```
// cds_dht22_express.js
// Express with CORS
var express = require('express');
var cors = require('cors');  // CORS: Cross Origin Resource Sharing
var app = express();
// CORS
app.use(cors());

var web_port = 3030;  // express port
// MongoDB
var mongoose = require('mongoose');
var Schema = mongoose.Schema;  // Schema object
// MongoDB connection
mongoose.connect('mongodb://localhost:27017/iot11'); // DB name
var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function callback () {
        console.log("mongo db connection OK.");
});
```

Web monitoring

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

Web monitoring-1: month

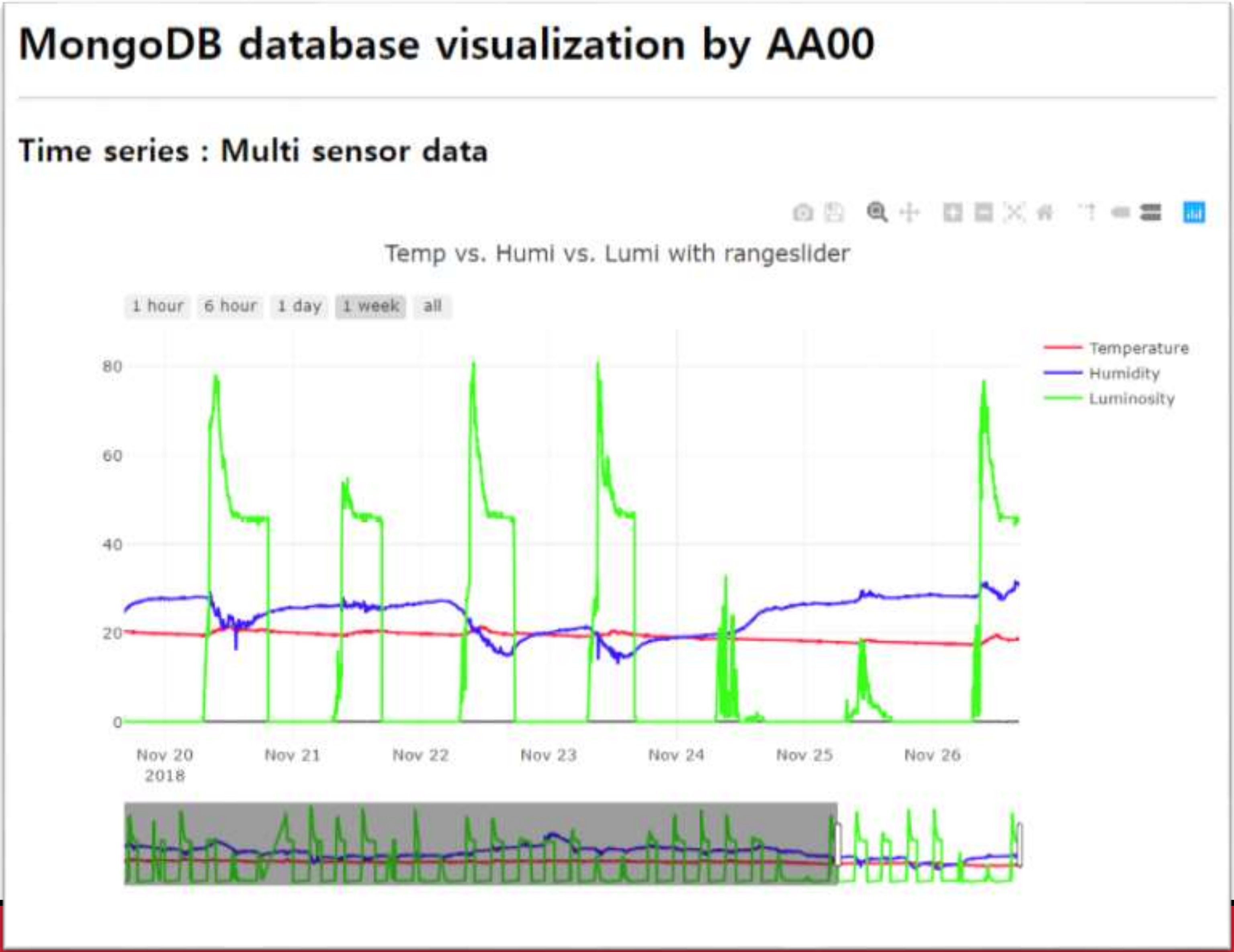


MongoDB database visualization by AA00

Time series : Multi sensor data

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

Web monitoring-2: week



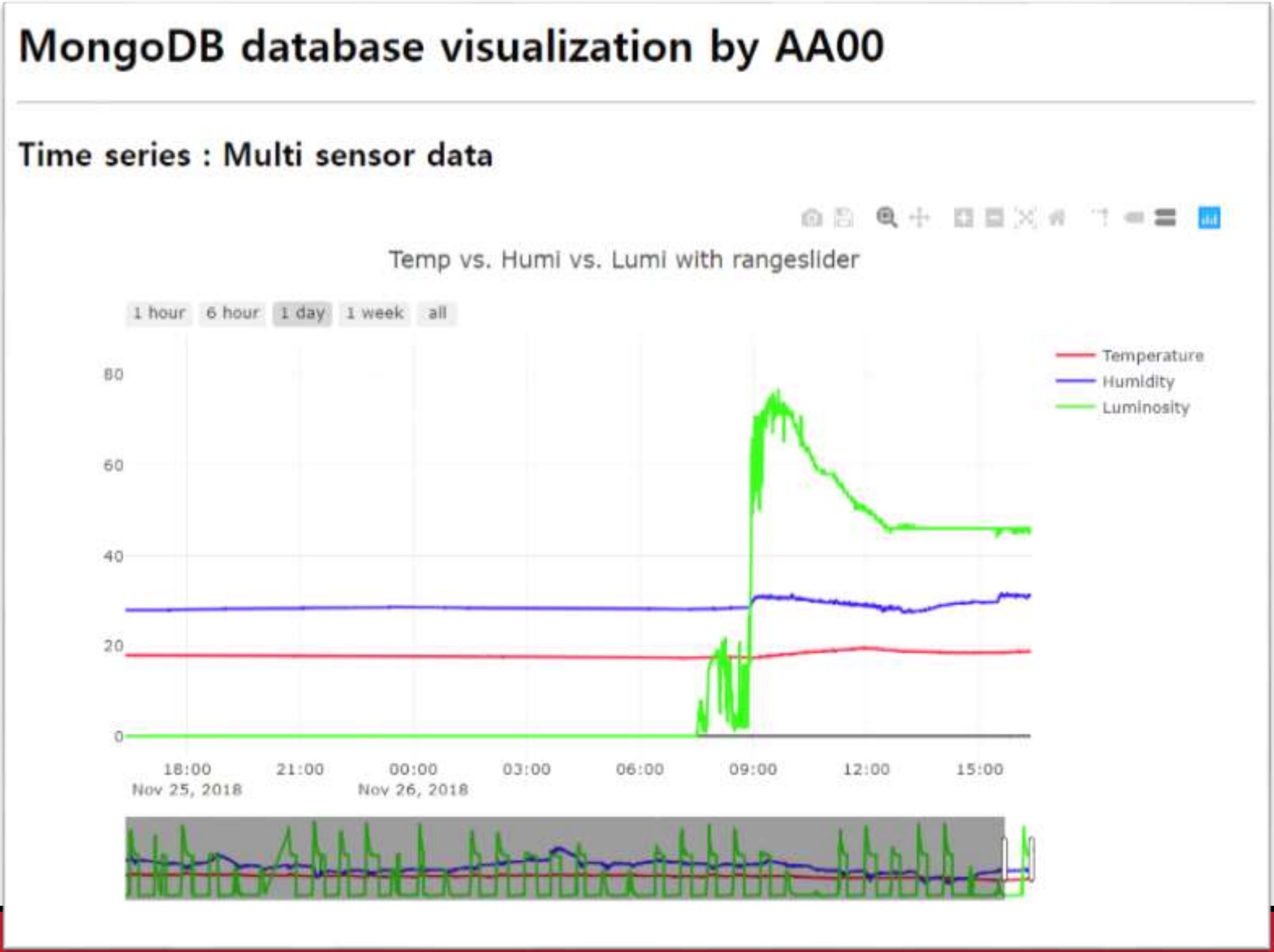
MongoDB database visualization by AA00

Time series : Multi sensor data

Temp vs. Humi vs. Lumi with rangeslider

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

Web monitoring-3: day

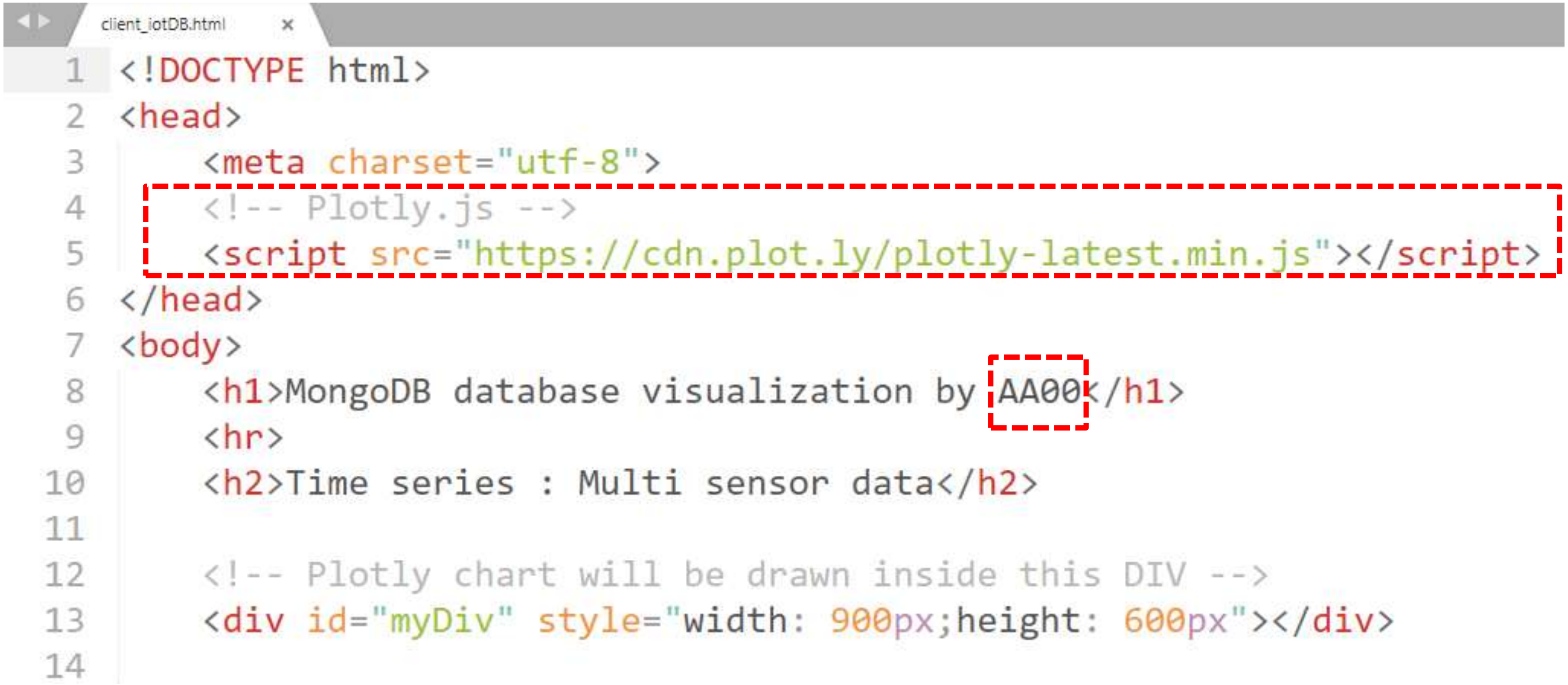


MongoDB database visualization by AA00

Time series : Multi sensor data

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

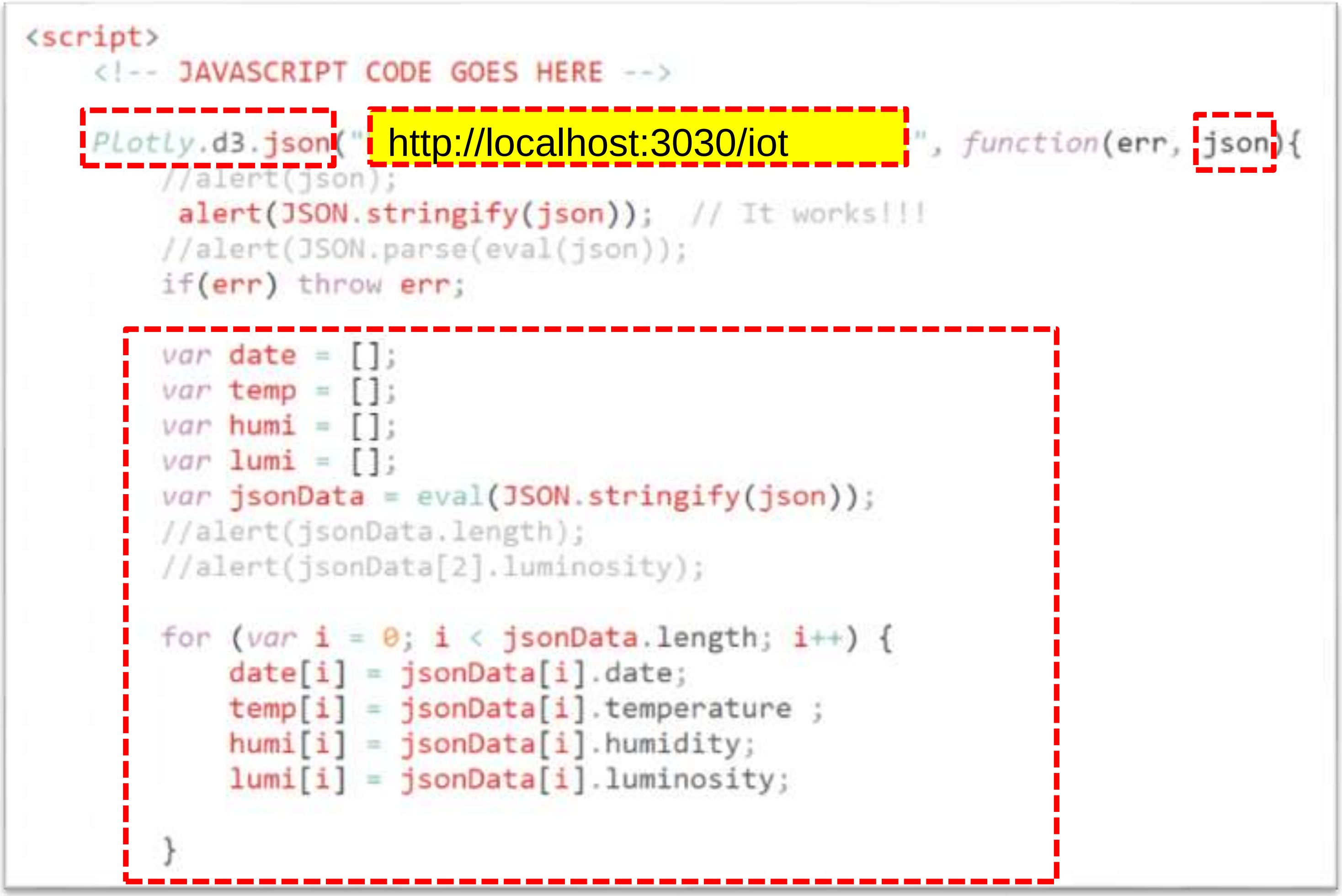
3.1 Web client: client_iotDB.html



```
<!DOCTYPE html>
<head>
    <meta charset="utf-8">
    <!-- Plotly.js -->
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
    <h1>MongoDB database visualization by AA00</h1>
    <hr>
    <h2>Time series : Multi sensor data</h2>

    <!-- Plotly chart will be drawn inside this DIV -->
    <div id="myDiv" style="width: 900px;height: 600px"></div>

```

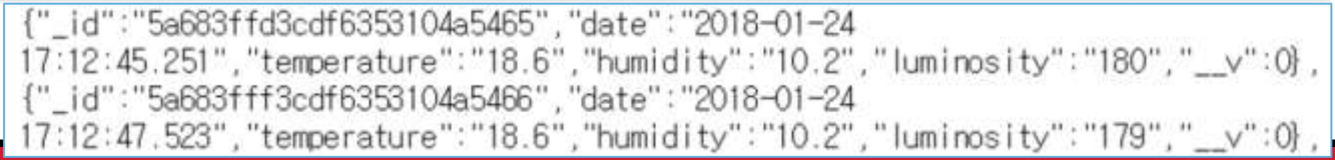
A5.9.7 DHT22 + CdS + Node.js + MongoDB

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;

        }
```

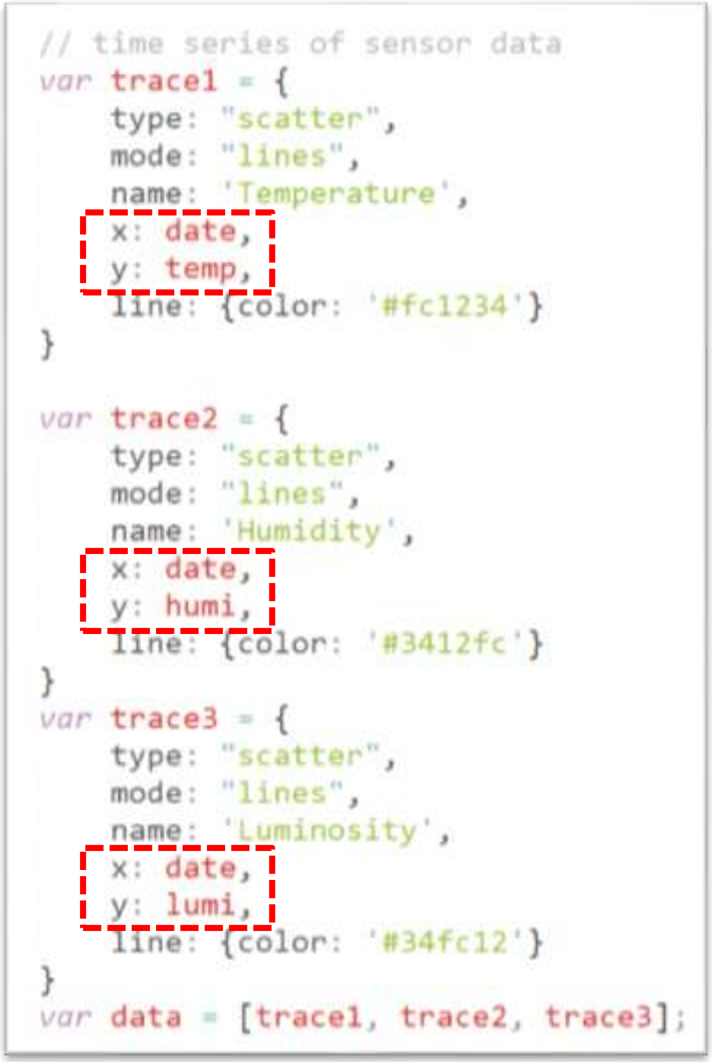
JSON file



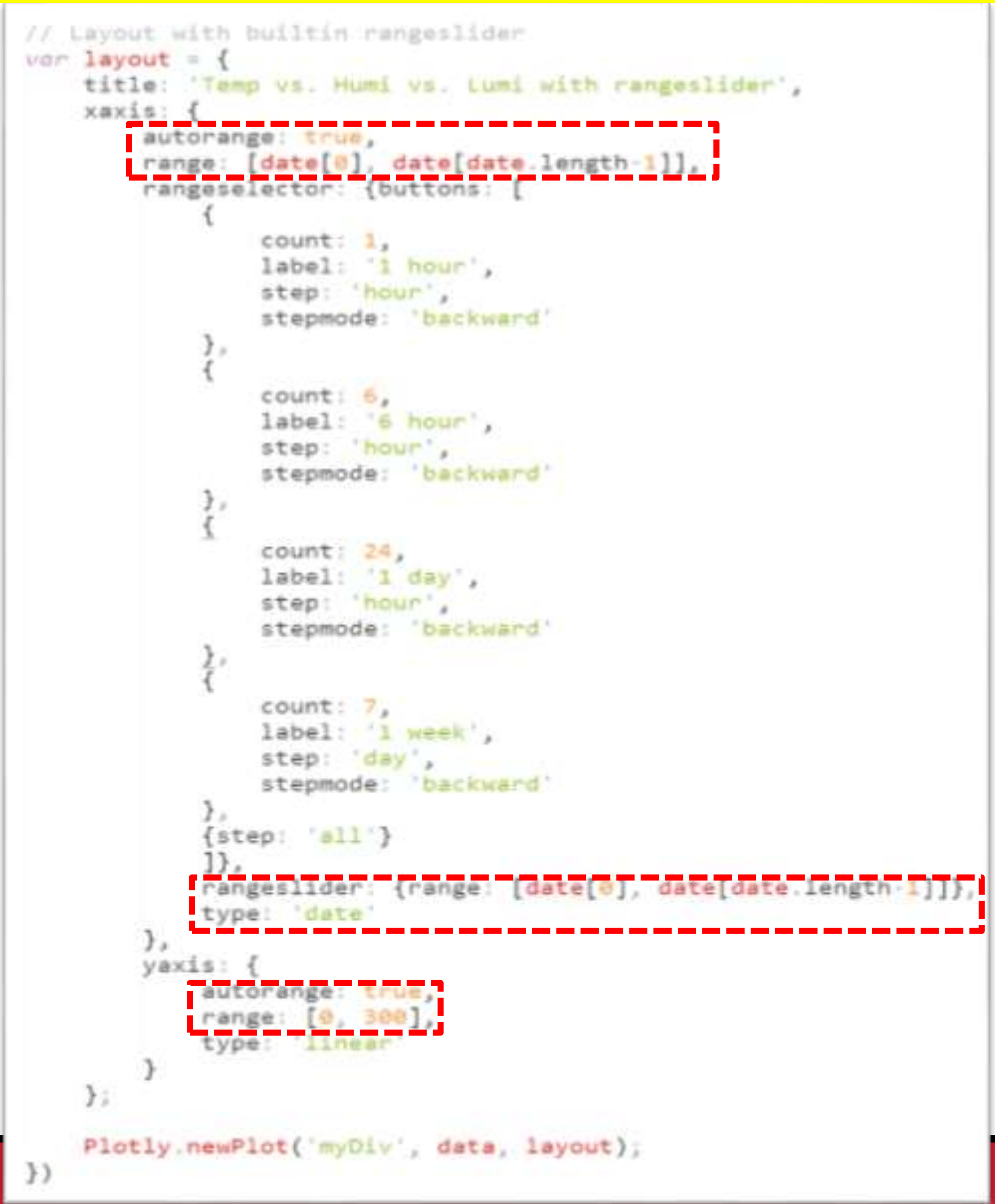
```
{"_id":"5a683ffd3cdf6353104a5465","date":"2018-01-24
17:12:45.251","temperature":"18.6","humidity":"10.2","luminosity":"180","__v":0},
{"_id":"5a683fff3cdf6353104a5466","date":"2018-01-24
17:12:47.523","temperature":"18.6","humidity":"10.2","luminosity":"179","__v":0},
```

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

3.3 Web client: client_iotDB.html – data & layout

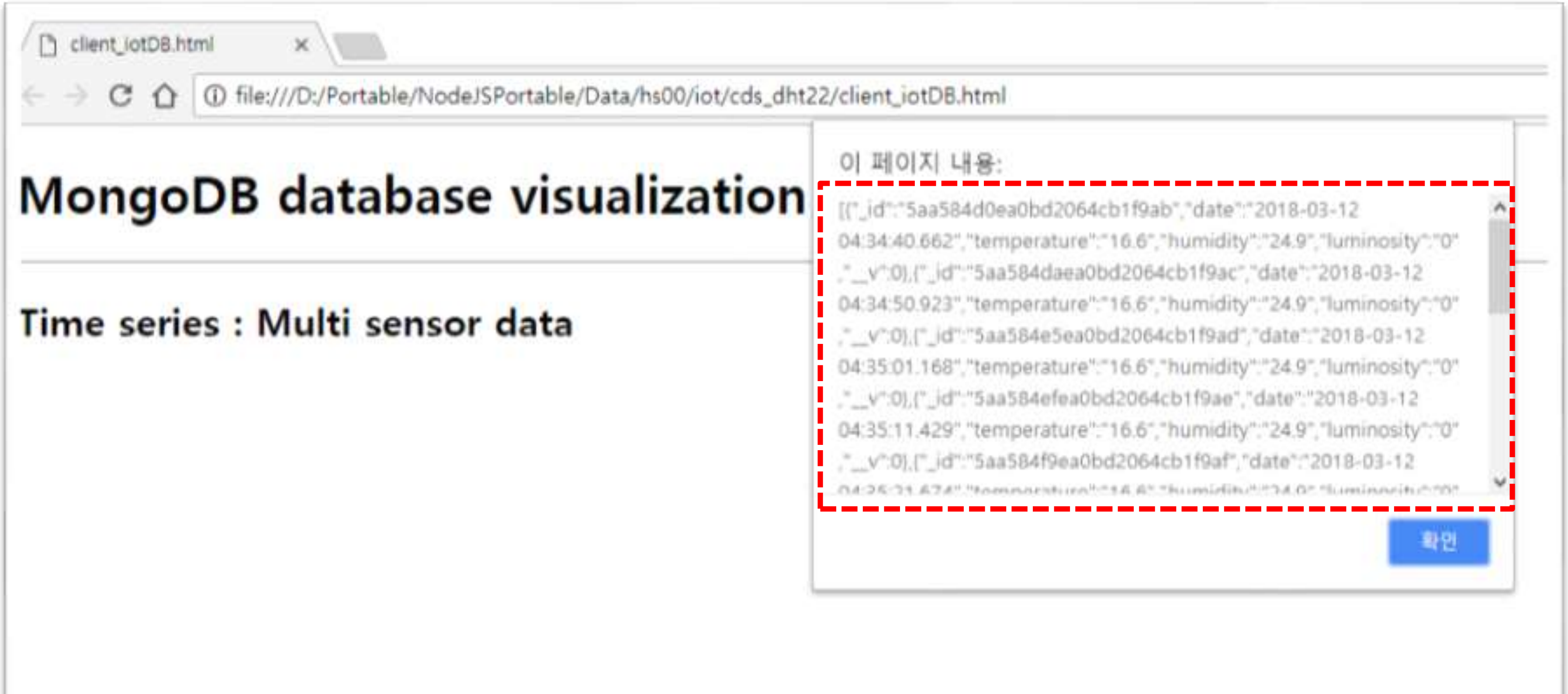


```
// time series of sensor data
var trace1 = {
    type: "scatter",
    mode: "lines",
    name: 'Temperature',
    x: date,
    y: temp,
    line: {color: '#fc1234'}
}

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'
    },
    yaxis: {
        autorange: true,
        range: [0, 300],
        type: 'linear'
    }
};

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

3.4 Web client: client_iotDB.html – load iot data in json file


client_iotDB.html
file:///D:/Portable/NodeJSPortable/Data/hs00/iot/cds_dht22/client_iotDB.html

MongoDB database visualization

Time series : Multi sensor data

이 페이지 내용:

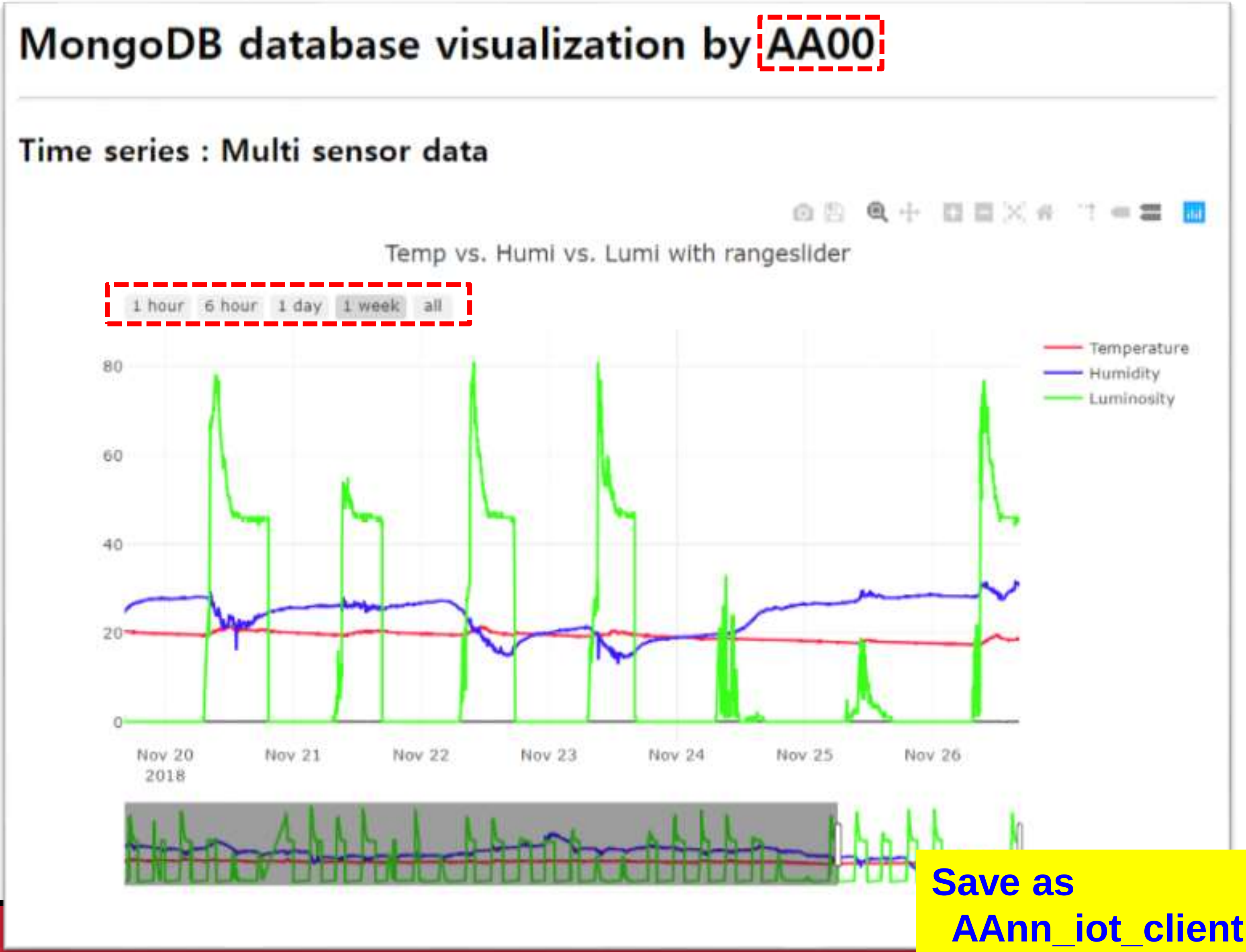
```
[{"_id":"5aa584d0ea0bd2064cb1f9ab","date":"2018-03-12
04:34:40.662","temperature":"16.6","humidity":"24.9","luminosity":"0"
,"__v":0},{"_id":"5aa584daea0bd2064cb1f9ac","date":"2018-03-12
04:34:50.923","temperature":"16.6","humidity":"24.9","luminosity":"0"
,"__v":0},{"_id":"5aa584e5ea0bd2064cb1f9ad","date":"2018-03-12
04:35:01.168","temperature":"16.6","humidity":"24.9","luminosity":"0"
,"__v":0},{"_id":"5aa584efea0bd2064cb1f9ae","date":"2018-03-12
04:35:11.429","temperature":"16.6","humidity":"24.9","luminosity":"0"
,"__v":0},{"_id":"5aa584f9ea0bd2064cb1f9af","date":"2018-03-12
04:35:21.674","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



MongoDB database visualization by AA00

Time series : Multi sensor data

Save as
AAnn_iot_client.png

mongoDB

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: “2018-11-26 22:26:05”}}) → 특정 시간 이후 document 추출

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

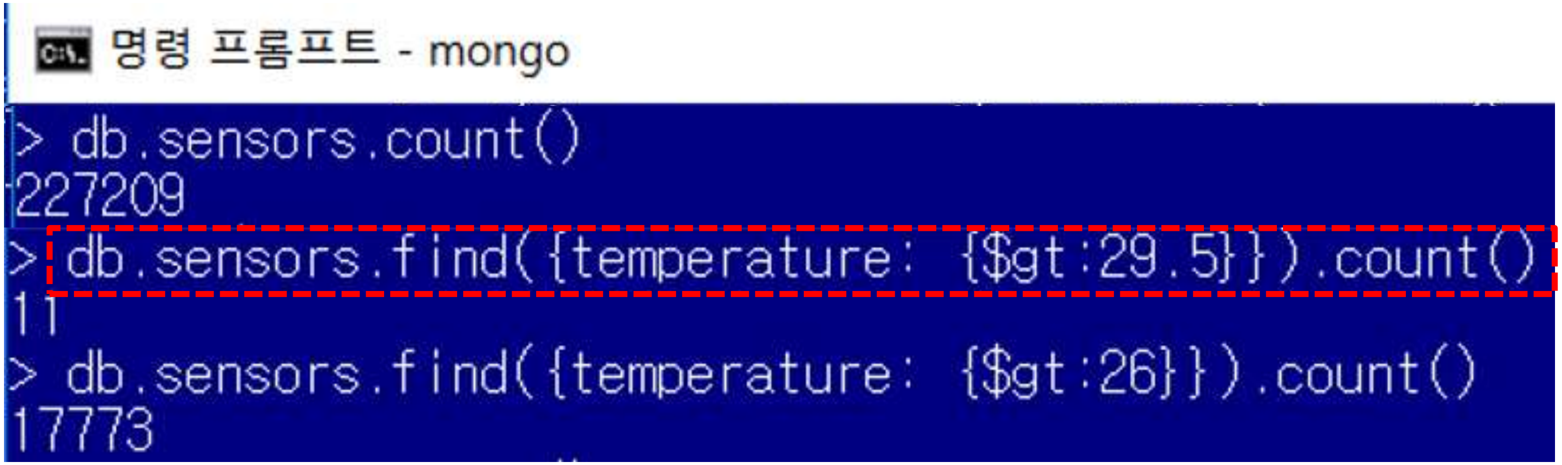
https://docs.mongodb.com/manual/tutorial/query-documents/

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

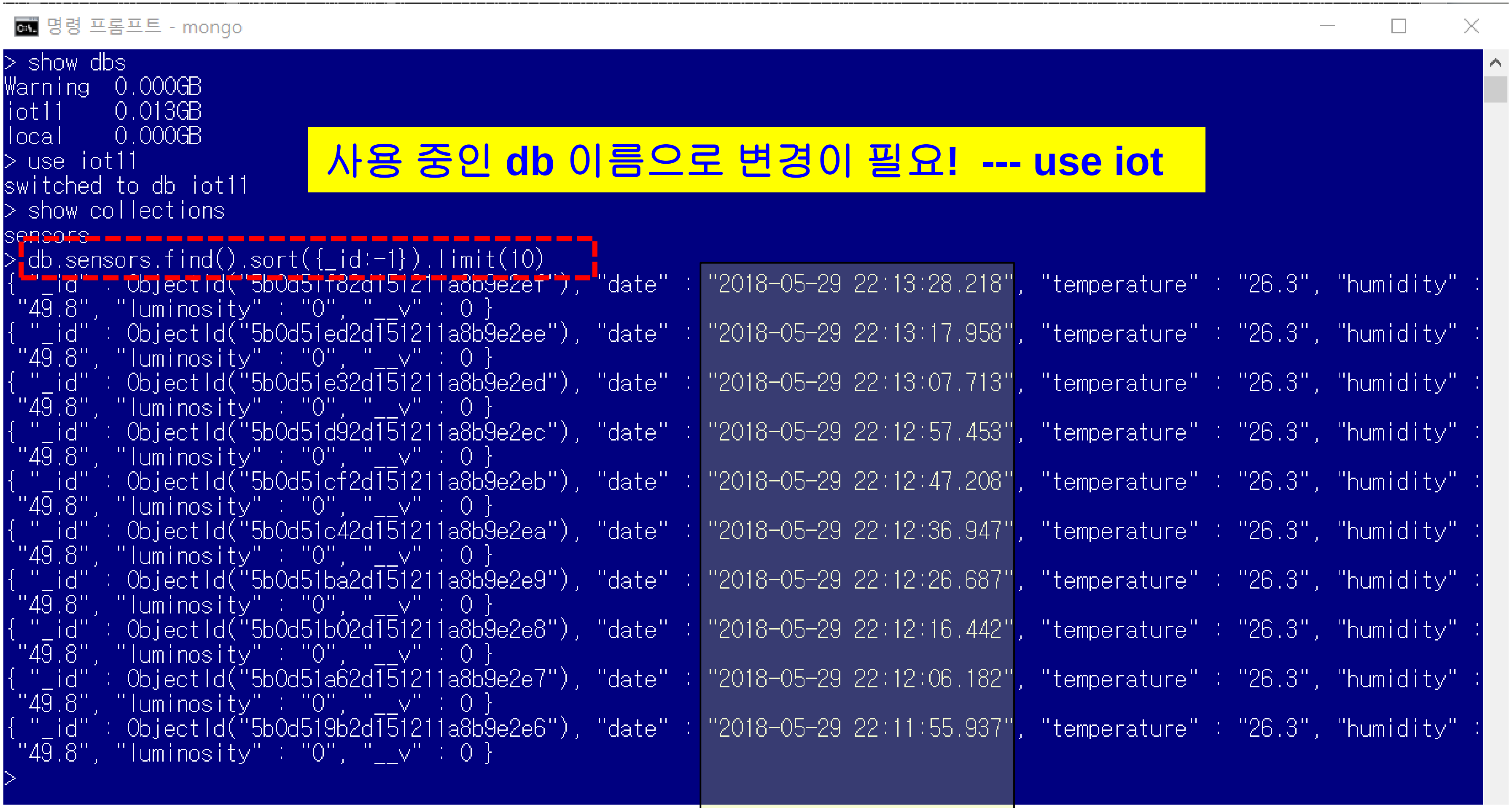


```
> db.sensors.count()
227209
> db.sensors.find({temperature: {$gt:29.5}}).count()
11
> db.sensors.find({temperature: {$gt:26}}).count()
17773
```

A5.9.8 MongoDB management

1.2 Query in Mongo shell

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



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

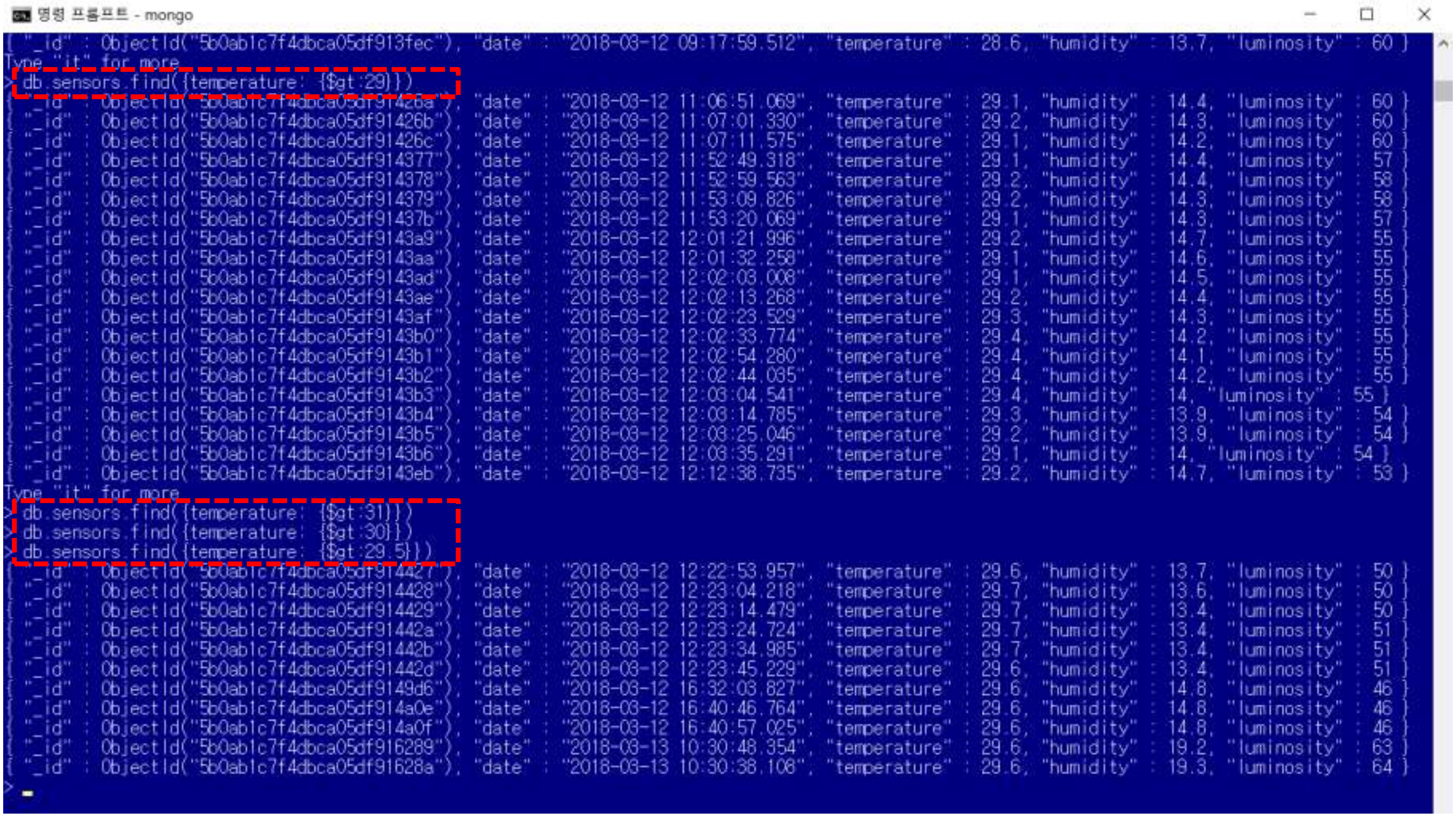
사용 중인 **db** 이름으로 변경이 필요! **--- use iot**

시간이 역순!

A5.9.8 MongoDB management

1.3 Query in Mongo shell

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

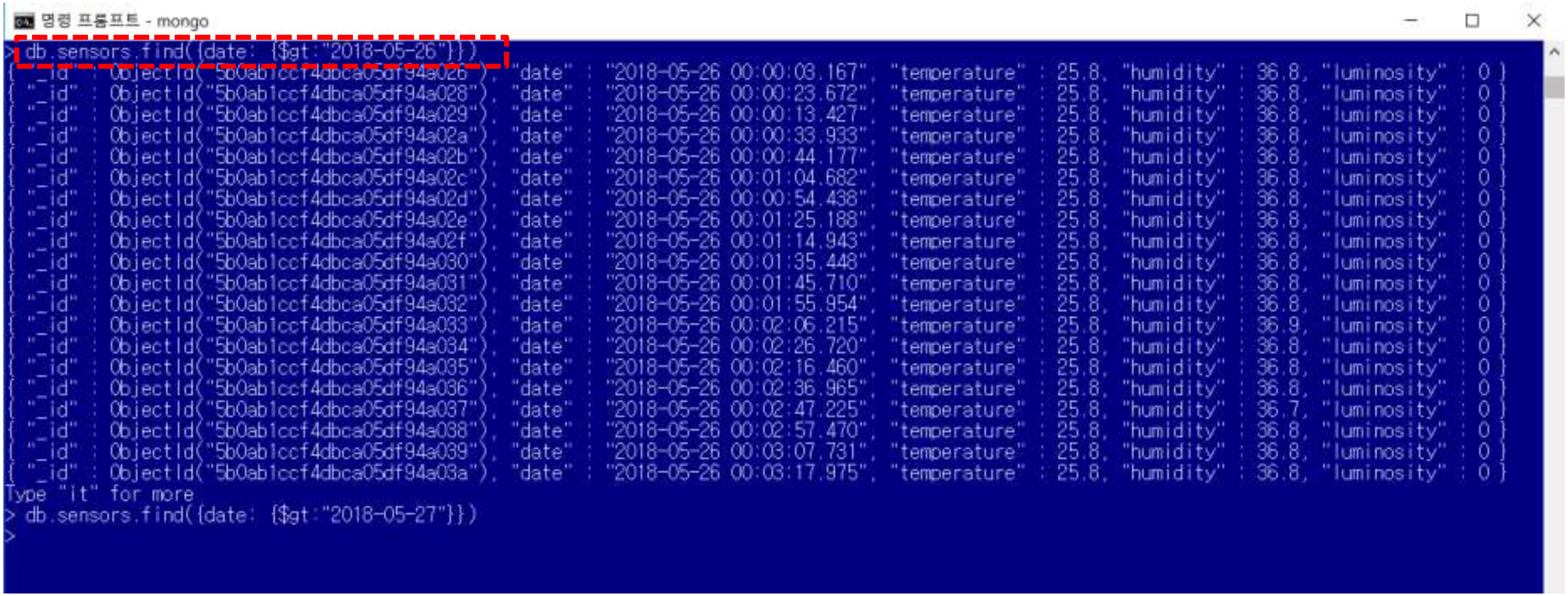


```
{ "_id" : ObjectId("5b0ab1c7f4dbca05df913fec"), "date" : "2018-03-12 09:17:59.512", "temperature" : 28.6, "humidity" : 13.7, "luminosity" : 60 }
Type "it" for more
> db.sensors.find({temperature: {$gt:29}})
{ "_id" : ObjectId("5b0ab1c7f4dbca05df91426a"), "date" : "2018-03-12 11:06:51.069", "temperature" : 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**)**



```
> 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
```

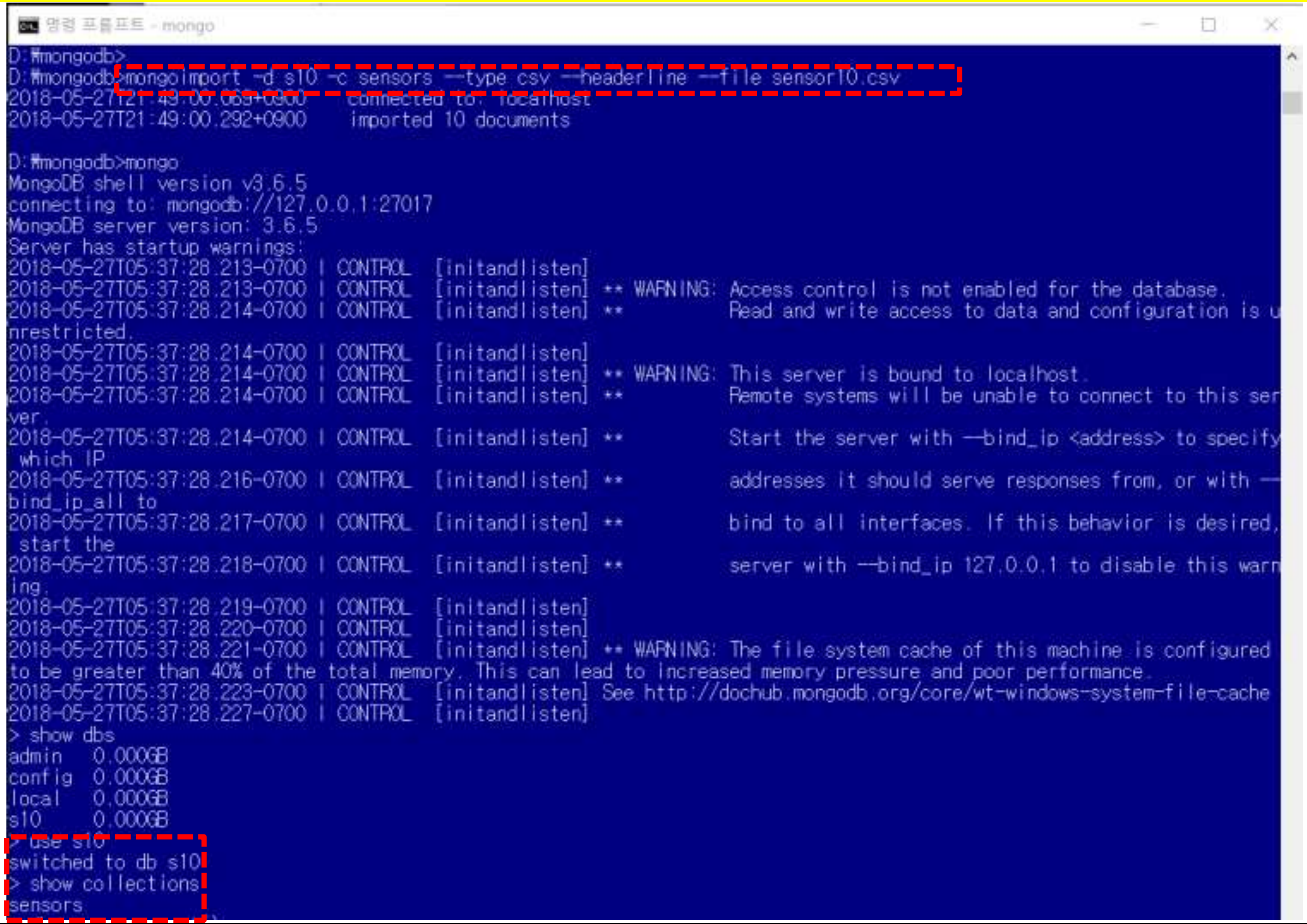
json 또는 csv 파일로 import/export

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

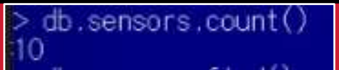
https://docs.mongodb.com/manual/reference/program/mongoexport/

2.1.1 Import MongoDB (windows cmd 창에서 실행)

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



```
D:₩mongodb>
D:₩mongodb>mongoimport -d s10 -c sensors --type csv --headerline --file sensor10.csv
2018-05-27T21:49:00.069+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 I CONTROL  [initandlisten]
2018-05-27T05:37:28.213-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-05-27T05:37:28.214-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is u
nrestricted.
2018-05-27T05:37:28.214-0700 I CONTROL  [initandlisten]
2018-05-27T05:37:28.214-0700 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2018-05-27T05:37:28.214-0700 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this ser
ver.
2018-05-27T05:37:28.214-0700 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify
 which IP
2018-05-27T05:37:28.216-0700 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --
bind_ip_all to
2018-05-27T05:37:28.217-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired,
 start the
2018-05-27T05:37:28.218-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warn
ing.
2018-05-27T05:37:28.219-0700 I CONTROL  [initandlisten]
2018-05-27T05:37:28.220-0700 I CONTROL  [initandlisten]
2018-05-27T05:37:28.221-0700 I 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 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2018-05-27T05:37:28.227-0700 I 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
```

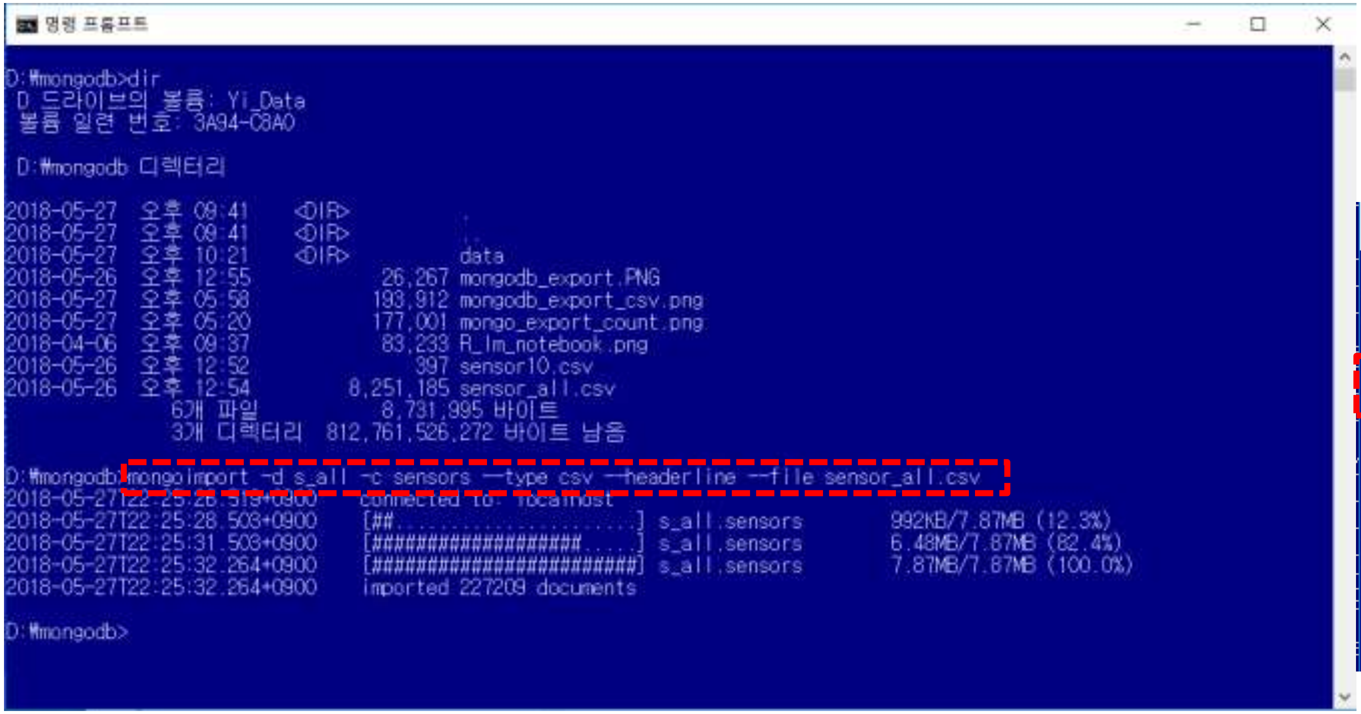


```
> db.sensors.count()
10
```

A5.9.8 MongoDB management

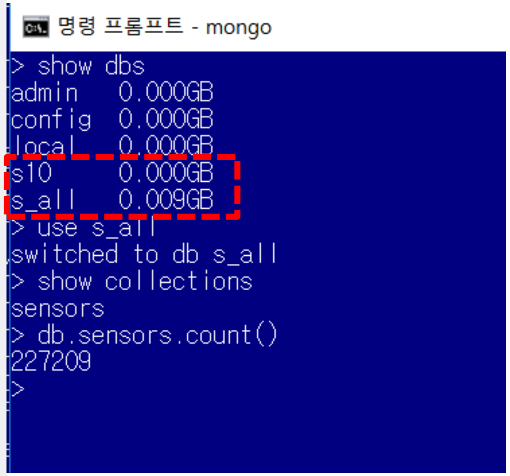
2.1.2 Import MongoDB (windows cmd 창에서 실행)

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



명령 프롬프트

```
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
               3개 디렉터리  812,761,526,272 바이트 남음

D:₩mongodb>mongoimport -d s_all -c sensors --type csv --headerline --file sensor_all.csv
2018-05-27T22:25:26.519+0900    connected to: localhost
2018-05-27T22:25:28.503+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

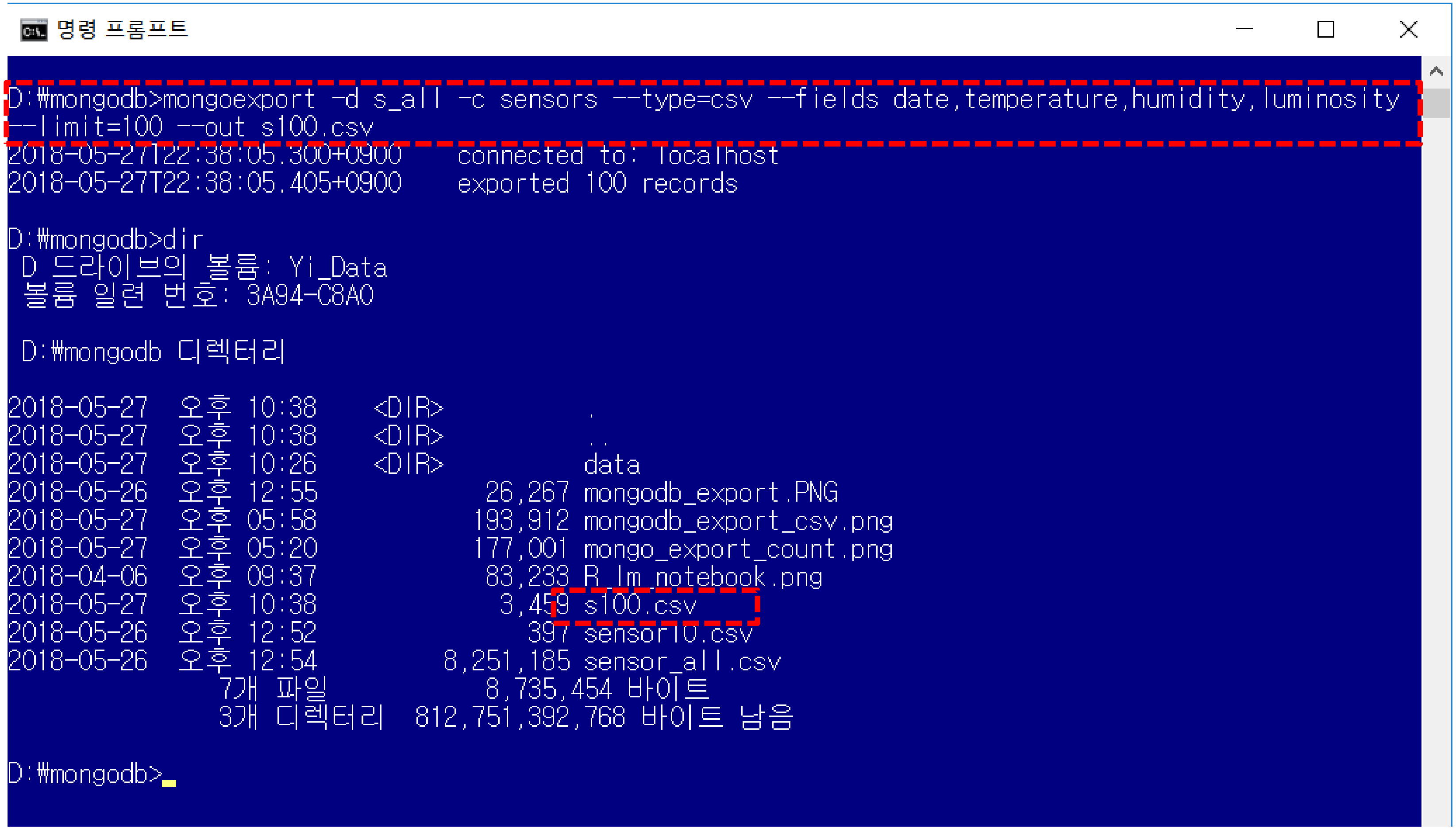
```
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
s10      0.000GB
s_all    0.009GB
> use s_all
switched to db s_all
> show collections
sensors
> db.sensors.count()
227209
>
```

[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 iot로 변경!)

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

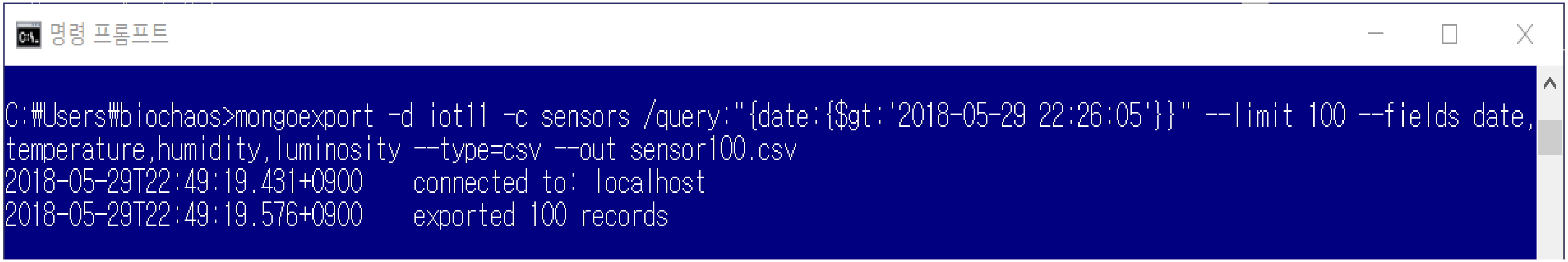


```
D:₩mongodb>mongoexport -d s_all -c sensors --type=csv --fields date,temperature,humidity,luminosity
--limit=100 --out s100.csv
2018-05-27T22:38:05.300+0900    connected to: localhost
2018-05-27T22:38:05.405+0900    exported 100 records

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)
iot11 db의 특정 시간 이후의 데이터 100개를 csv 파일 (s100.csv)로 저장

- ➢ **mongoexport -d iot11 -c sensors /query:"{date: {$gt: '2018-05-29 22:26:06'}}" --limit=100 --fields date,temperature,humidity,luminosity --type=csv --out s100.csv**



```
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
```

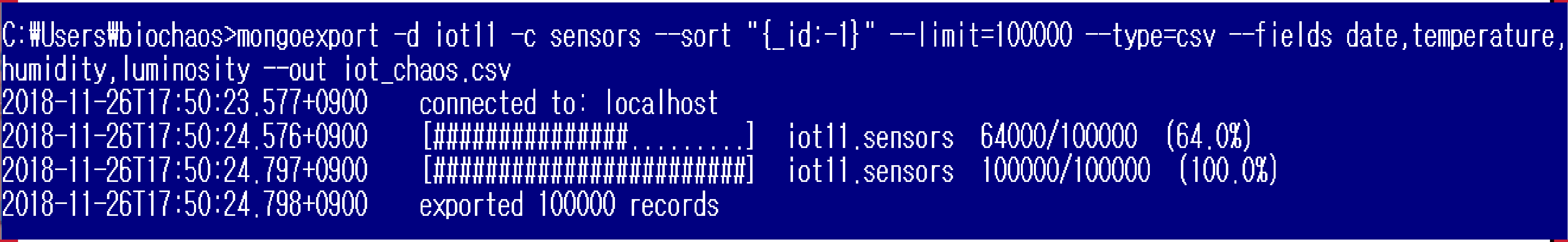
[Tip] iot db의 최근 데이터 500개를 csv 파일 (s500.csv)로 저장할 때,

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

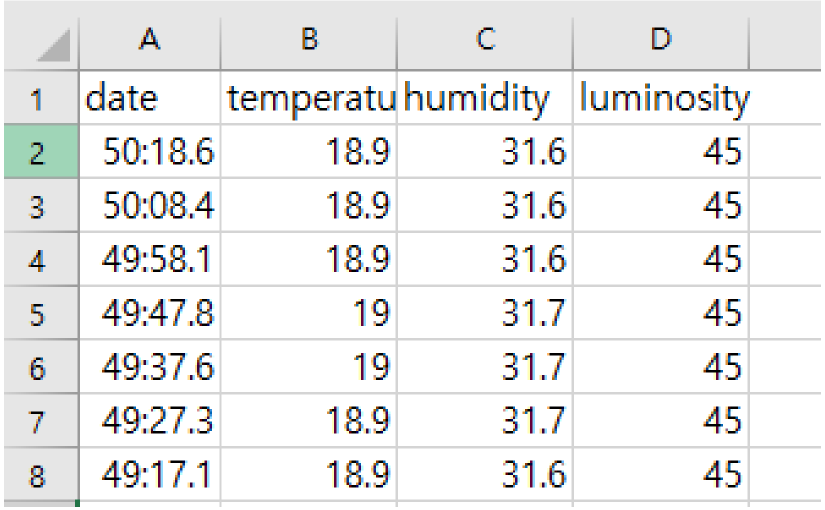
A5.9.8 MongoDB management

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

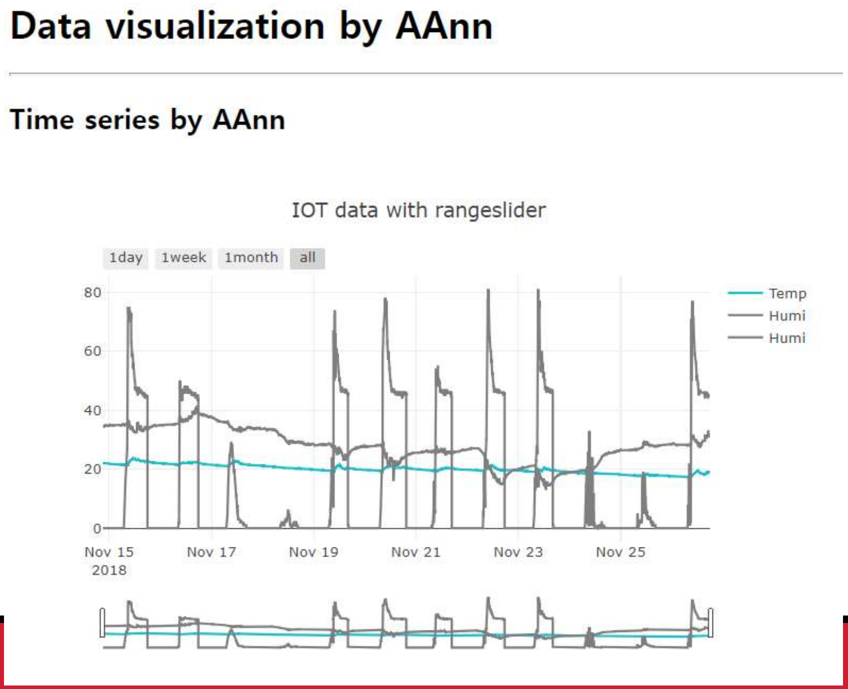
- **mongoexport –d iot -c sensors --sort “{_id: -1}” --limit=500 --fields date,temperature,humidity,luminosity --type=csv --out s500.csv**



```
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



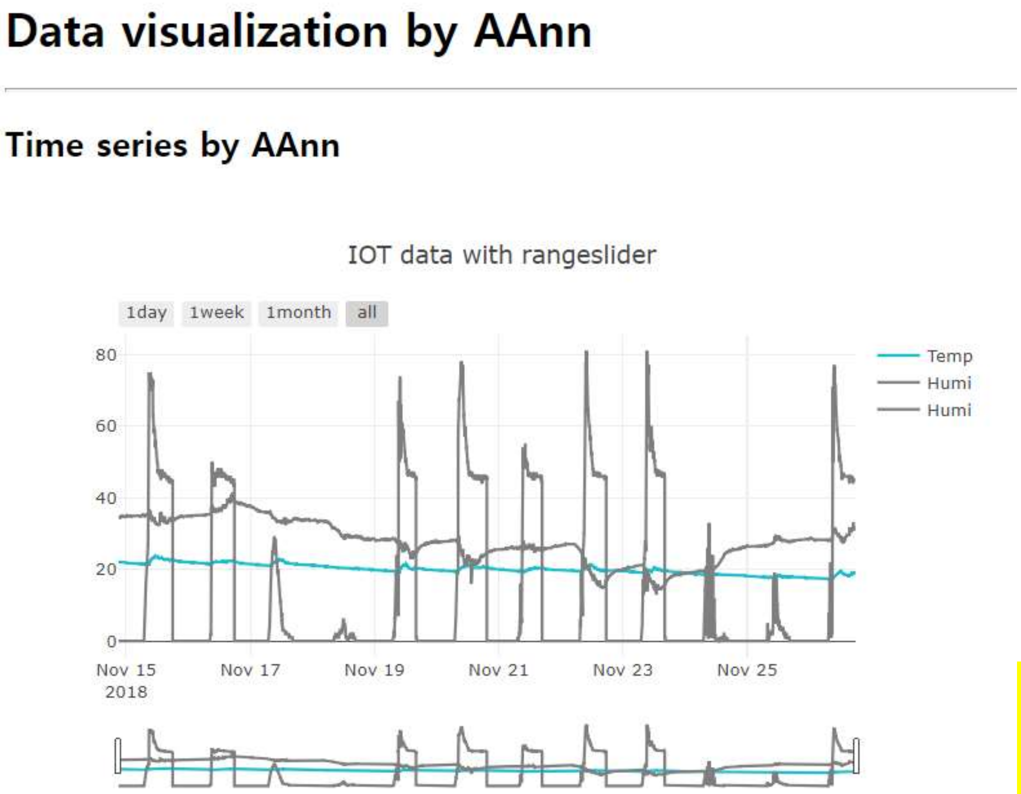
Data visualization by AAnn

Time series by AAnn

A5.9.8 MongoDB management

[DIY]

1. iot db의 최근 데이터 1000개를 csv 파일 (AAnn_s1000.csv)로 저장하시오.
2. 저장된 AAnn_s1000.csv 파일을 public/data 폴더에 복사.
2. csv 파일을 이용해서 Rangeslider가 포함된 웹 클라이언트 client_iot.html 파일을 만드시오.



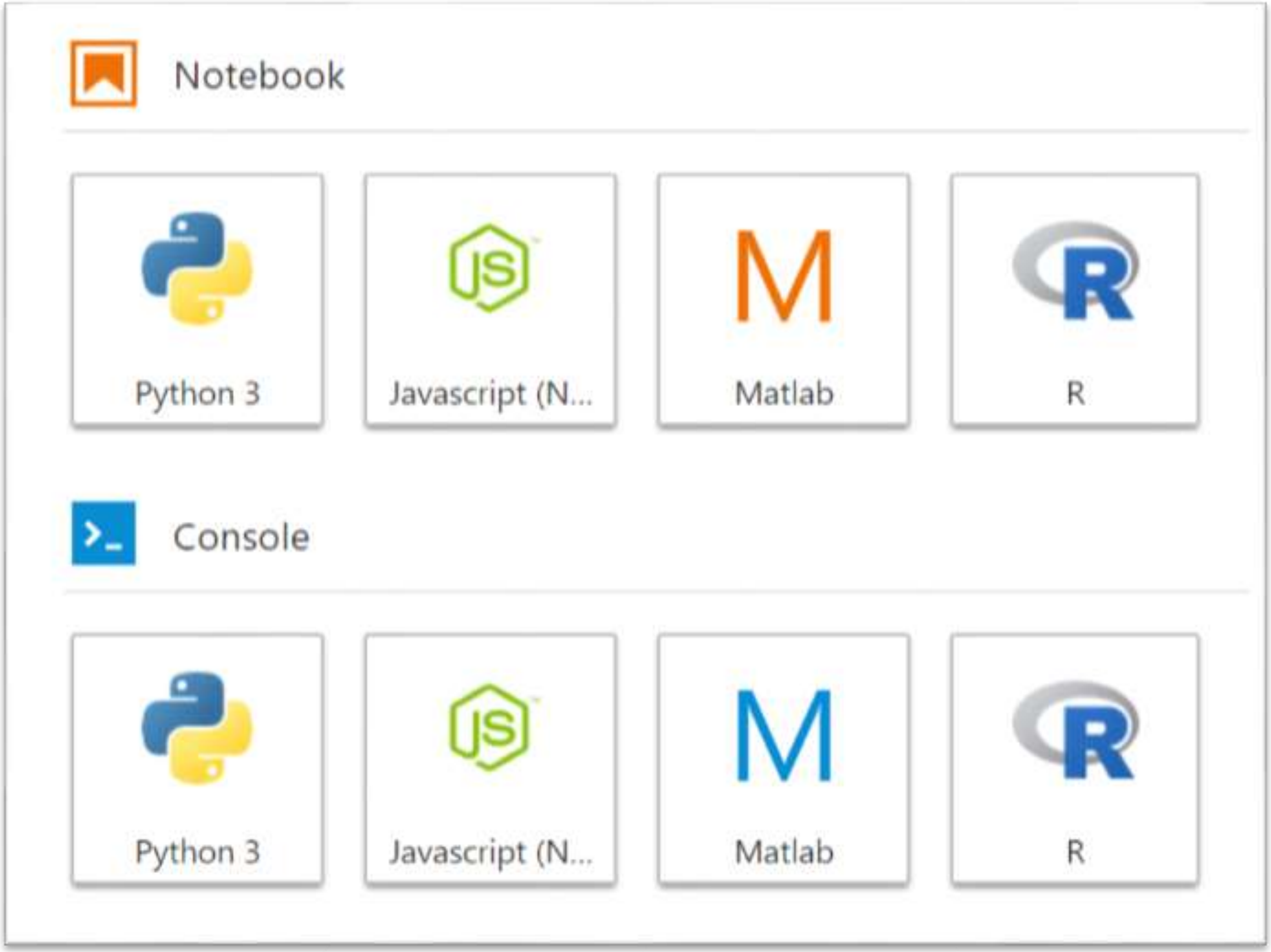
Data visualization by AAnn

Time series by AAnn

Save as AAnn_s1000.png

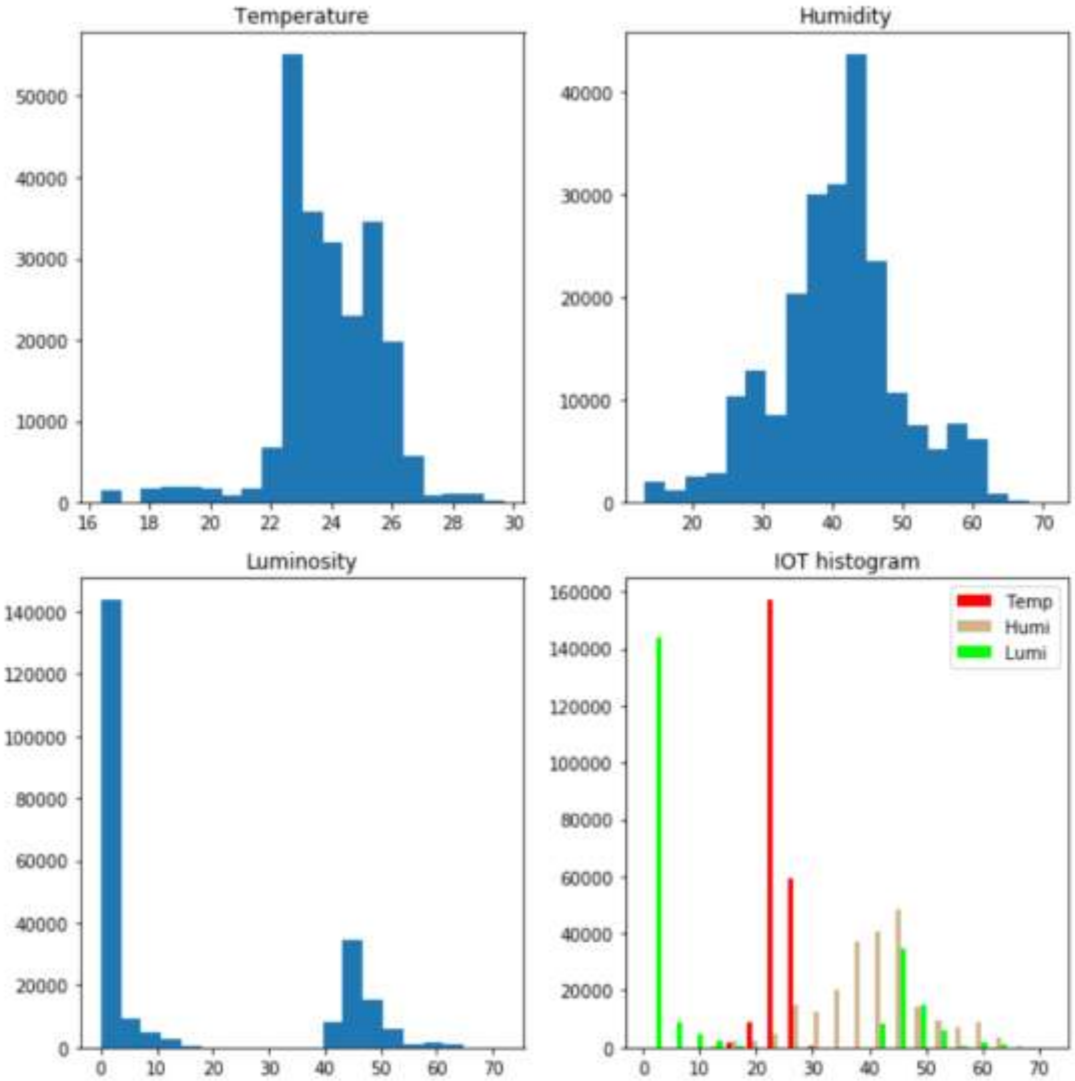
A5.9.8 iot data mining (next week)

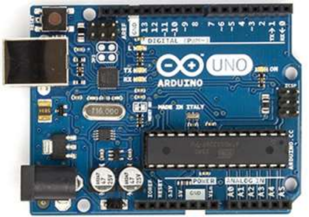
3. How to use and understand iot data? → Python(or R) in Jupyter lab



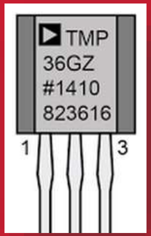
A5.9.8 MongoDB management

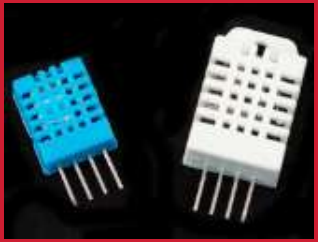
3.3 How to use and understand iot data? → csv_dht22_Py.ipynb





[Practice]





◆ [wk14]

- ➢ RT Data storaging with MongoDB
- ➢ Multi-sensor circuits(cds-dht22)
- ➢ Complete your project
- ➢ Upload file name : AAnn_Rpt10.zip

[wk14] Practice-10 AAnn_Rpt10.zip

◆ [Target of this week]

- Complete your charts
- Save your outcomes and compress them.

제출파일명 : **AAnn_Rpt10.zip**

- 압축할 파일들

① **AAnn_mongo_schemas.png**

② **AAnn_mongo_update.png**

③ **AAnn_iot_mongodb.png**

④ **AAnn_iot_mongodb_web.png**

⑤ **AAnn_iot_json.png**

⑥ **AAnn_iot_client.png**

⑦ **AAnn_s1000.csv (mongoexport file)**

⑧ **AAnn_s1000.png**

Email : chaos21c@gmail.com

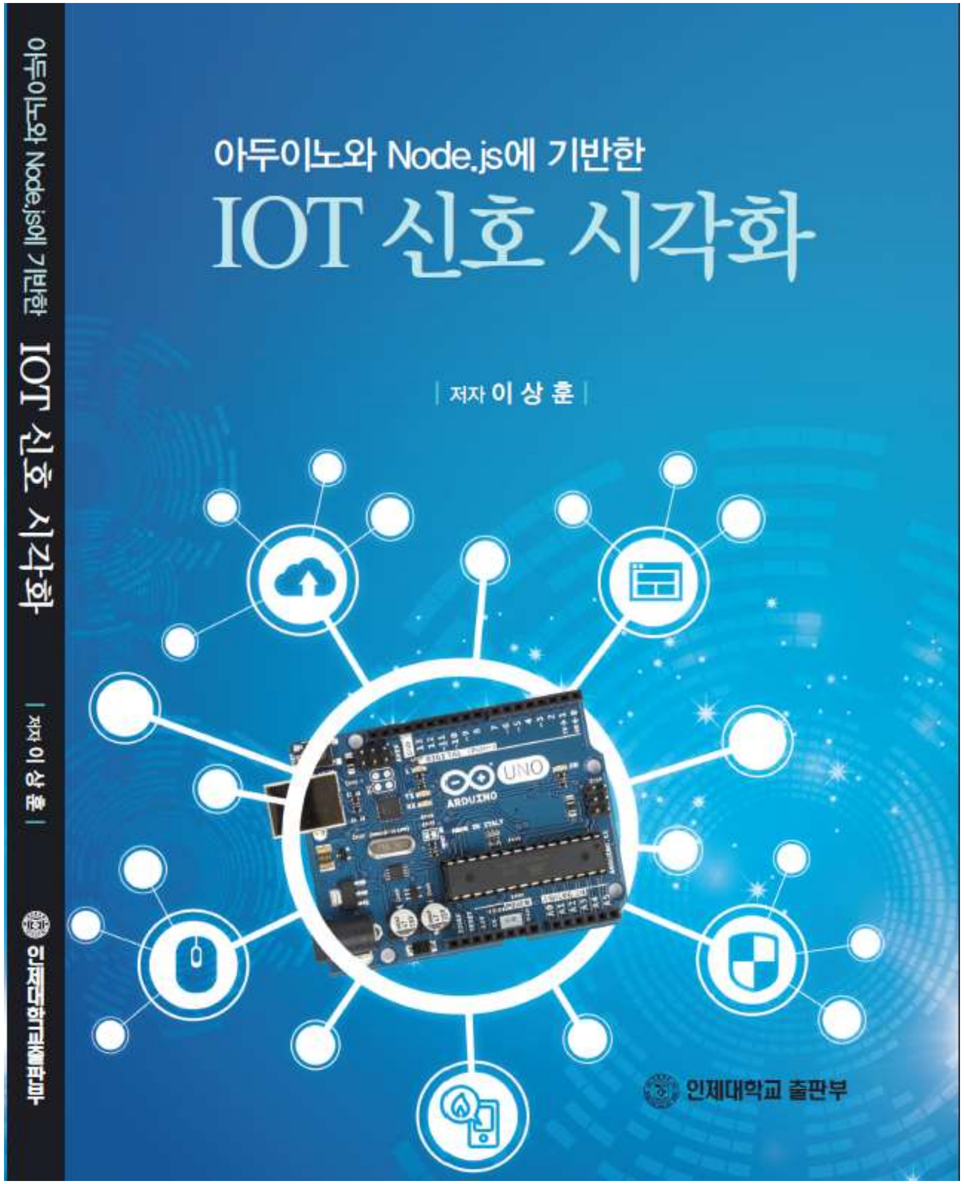
[제목 **: id,** 이름 **(**수정**)]**

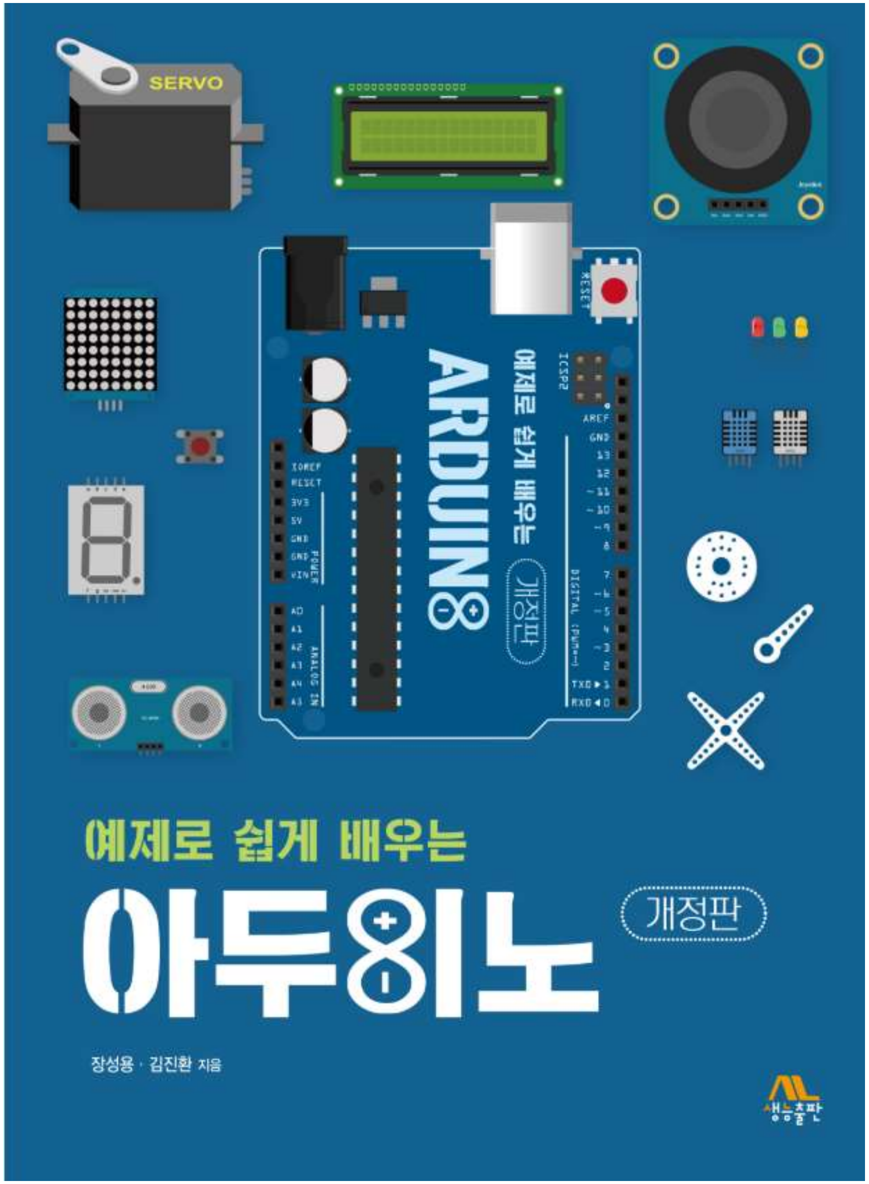
Lecture materials

References & good sites

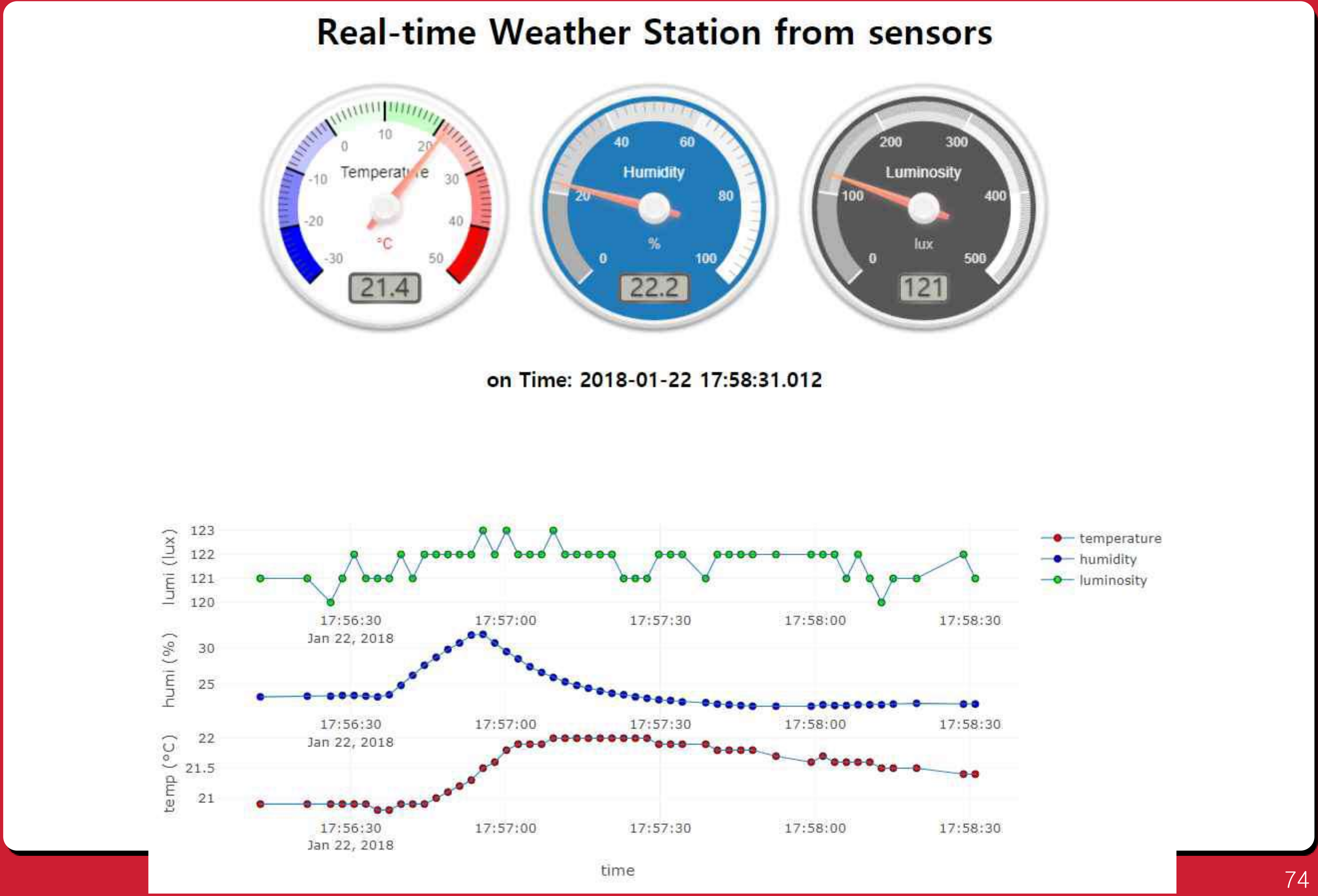
- ✓ http://www.arduino.cc Arduino Homepage
- ✓ http://www.nodejs.org/ko Node.js
- ✓ https://plot.ly/ plotly
- ✓ https://www.mongodb.com/ MongoDB
- ✓ http://www.w3schools.com By w3schools
- ✓ http://www.github.com GitHub

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
인제대학교출판부


SERVO
예제로 쉽게 배우는
ARDUINO
개정판
예제로 쉽게 배우는
아두이노
개정판
장성용 · 김진환 지음
생능출판

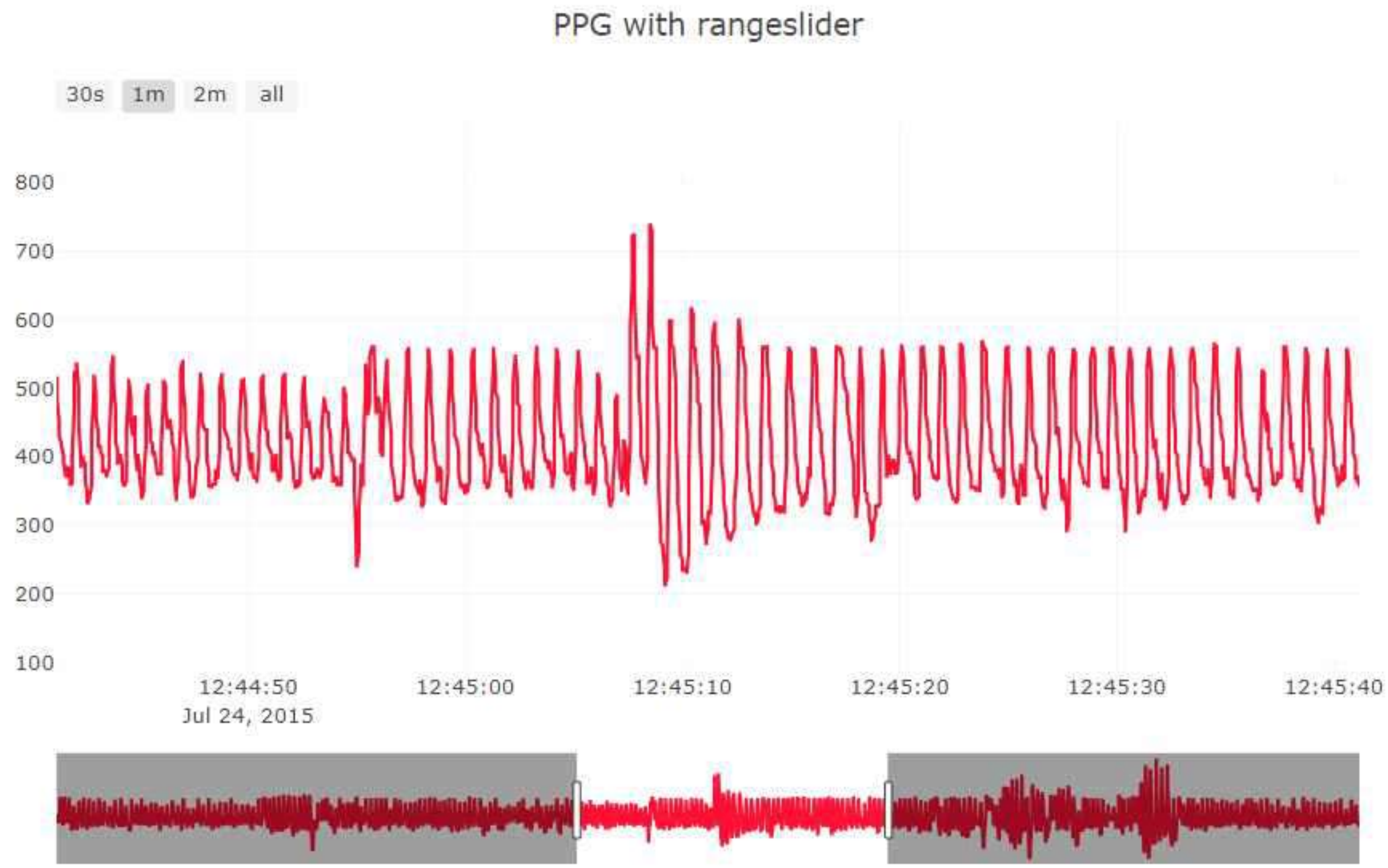
Target of this class



Real-time Weather Station from sensors

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

Another target of this class


PPG with rangeslider
30s
1m
2m
all
12:44:50
Jul 24, 2015
12:45:00
12:45:10
12:45:20
12:45:30
12:45:40