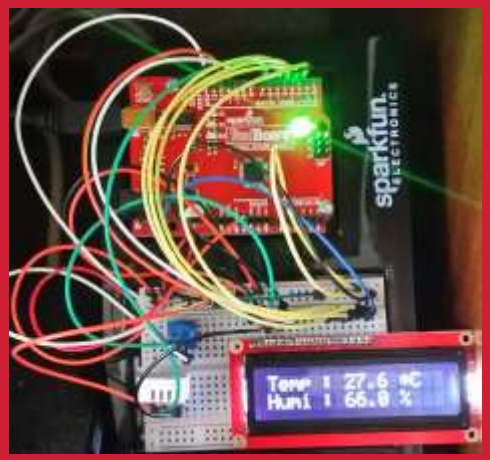
Healthcare-IOT
[wk13]



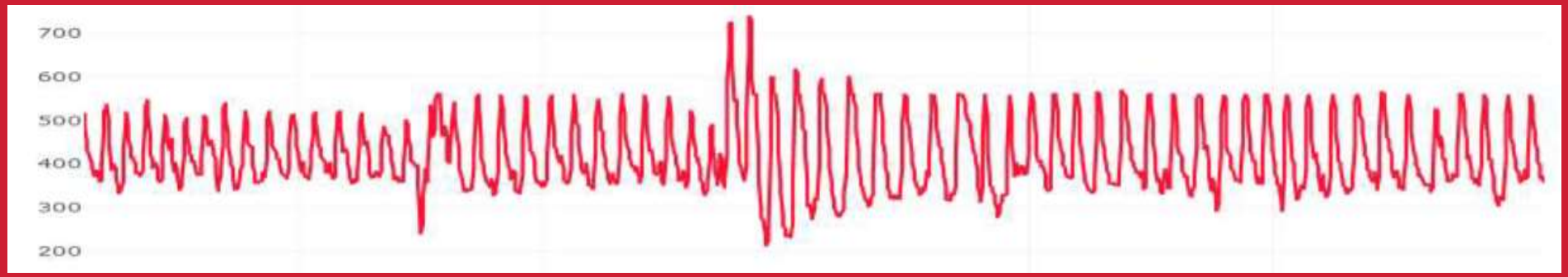
Data storaging using MongoDB II.

Visualization of Healthcare Signals using Arduino & Node.js

HCit, INJE University

1st semester, 2018

Email : chaos21c@gmail.com



My ID

오전

성명	ID
김민선	HS01
김영걸	HS02
김주란	HS03
김주현	HS04
김태민	HS05
여준하	HS06
이수민	HS07
정민지	HS08
정유현	HS09
정재은	HS10
주하영	HS11
한준영	HS12

오후

성명	ID
신영주	HS21
오가영	HS22
윤민수	HS23
윤진아	HS24
이진영	HS25
임상은	HS26
임재형	HS27
최민영	HS28
황유빈	HS29

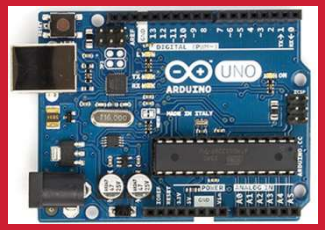
주간계획서

주간계획서			
주차	수업방법	수업내용	과제물
1	강의/실습	수업 및 실습 안내 -포터블 소프트웨어 설치	
2	강의/실습	Node.js I - Node.js 코드의 기본 구조 - 기초 Node 서버 및 클라이언트	실습확인
3	강의/실습	Node.js II - Node.js Express 서버	실습확인
4	강의/실습/발표	Arduino I - 아날로그 신호 회로 - LCD를 이용한 센서 신호 모니터링	실습확인
5	강의/실습	Arduino II - 단일 센서 회로와 Node.js 연결 - 다중 센서 회로와 Node.js 연결	실습확인
6	강의/실습	프로젝트1 - 생체 센서 회로와 Node.js 연결 - 생체 신호 소개	프로젝트1
7	강의/실습/발표	IOT 데이터 시각화 I (Plotly.js) - 데이터 및 시계열 차트 - 데이터 스트리밍	실습확인
8	시험	중간고사	
9	강의/실습	IOT 데이터 시각화 II (Plotly.js) - 다중 센서 데이터 시각화 - 다중 센서 데이터 스트리밍	실습확인
10	강의/실습/발표	프로젝트II - 생체 센서 데이터 시각화 - 생체 센서 데이터 스트리밍	프로젝트II
11	강의/실습	IOT 데이터 저장과 처리 - MongoDB 설치 및 Mongo shell - MongoDB와 Node.js 연결 및 데이터 저장	실습확인
12	강의/실습	프로젝트III - MongoDB에 IOT 데이터 저장 및 모니터링 - 생체 센서 데이터 저장 및 시각화	프로젝트III
13	강의/실습	IOT 데이터 마이닝 - 아두이노에서 발생된 데이터 관리 - 데이터마이닝 소개	실습확인
14	강의/실습/발표	프로젝트IV - 생체 센서 데이터 관리 - 생체 센서 데이터 마이닝	프로젝트IV
15	시험	기말고사	

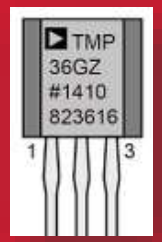
Purpose of HS

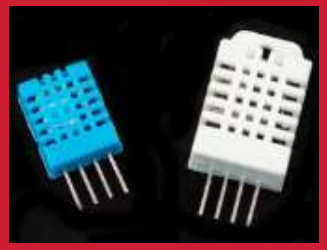
주요 수업 목표는 다음과 같다.

1. Node.js를 이용한 아두이노 센서 신호 처리
2. Plotly.js를 이용한 아두이노 센서 신호 시각화
3. MongoDB에 아두이노 센서 데이터 저장 및 처리
4. 생체 센서 발생 신호 처리, 시각화 및 저장
5. 생체 센서 발생 신호 저장 및 분석
6. 생체 신호 장비 활용 능력



[Review]





- [wk12]
  - RT Data storaging with MongoDB
  - Multi-sensor circuits
  - Complete your project
  - Upload file name : HSnn_Rpt10.zip

[wk12] Practice-10 HSnn_Rpt10.zip

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

제출파일명 : **HSnn_Rpt10.zip**

- 압축할 파일들

① **HSnn_mongo_schemas.png**

② **HSnn_mongo_update.png**

③ HSnn_iot_mongodb.png

④ HSnn_iot_mongodb_web.png

Email : chaos21c@gmail.com

[제목 : id, 이름 (수정)]

[My working folder – wk12]

PC › DATA (D:) › Portable › arduino-1.8.5 › hs00

- hcit
- HS00_AnalogRead_fmap
- HSnn_CdS_DHT22
- HSnn_CdS_TMP36_LCD
- HSnn_multi_signals
- HSnn_TMP36_CdS
- HSnn_TMP36_NodeJS
- HSnn_TMP36_start
- libraries
- sketch01_blink
- sketch01_start
- sketch02_pwm_led
- sketch03_pwm_led_serial
- sketch04_pwm_3_leds
- sketch05_multi_signals
- sketch06_analog_read
- sketch06_analog_read_start
- sketch07_tmp36
- sketch07_tmp36_start
- sketch08_CdS
- sketch08_CdS_start
- sketch08_CdS2
- sketch09_hello_LCD
- sketch09_hello_LCD_start
- sketch10_CdS_LCD_lux
- sketch10_CdS_LCD_lux_start
- sketch12_CdS_TMP36_LCD

hs00

- express
- expressTest
- hs00App
- iot
- myApp
- server
- start

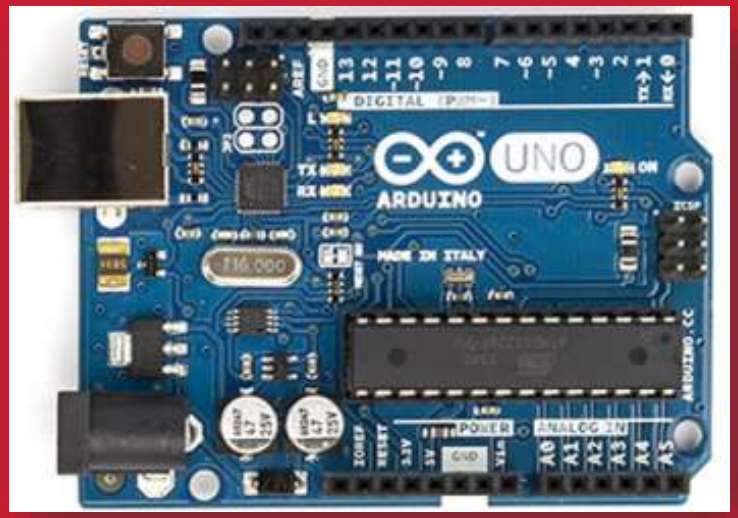
이름

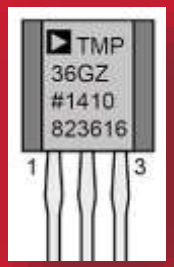
- cds
- cds_dht22
- cds_tmp36
- multi_signals
- plotly
- tmp36

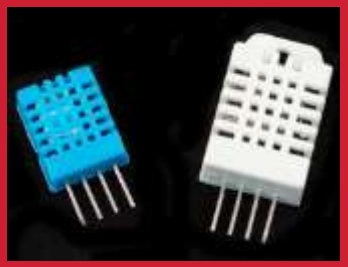
Arduino

+ Node.js

+ plotly.js







IOT: HSC

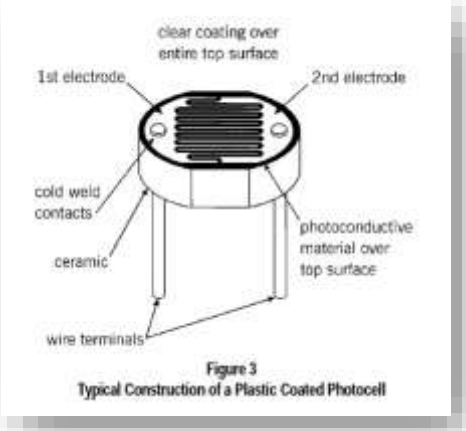
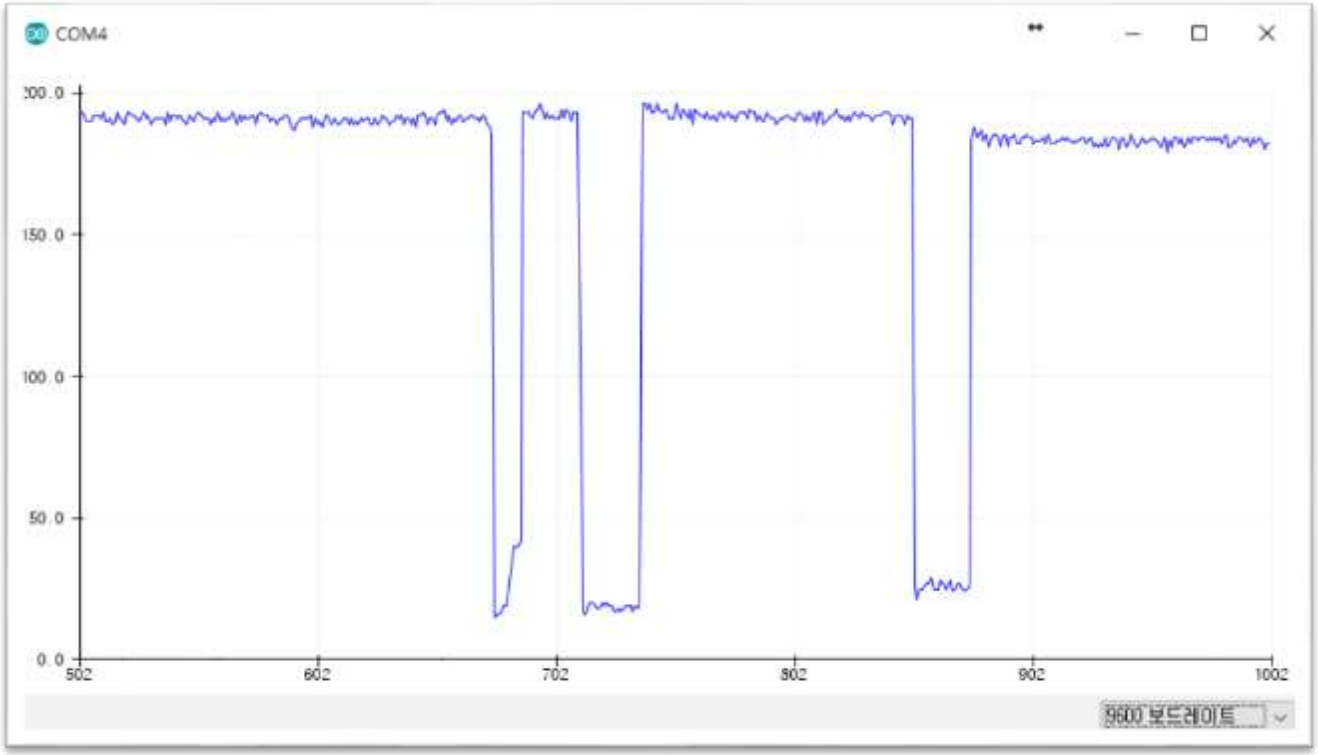
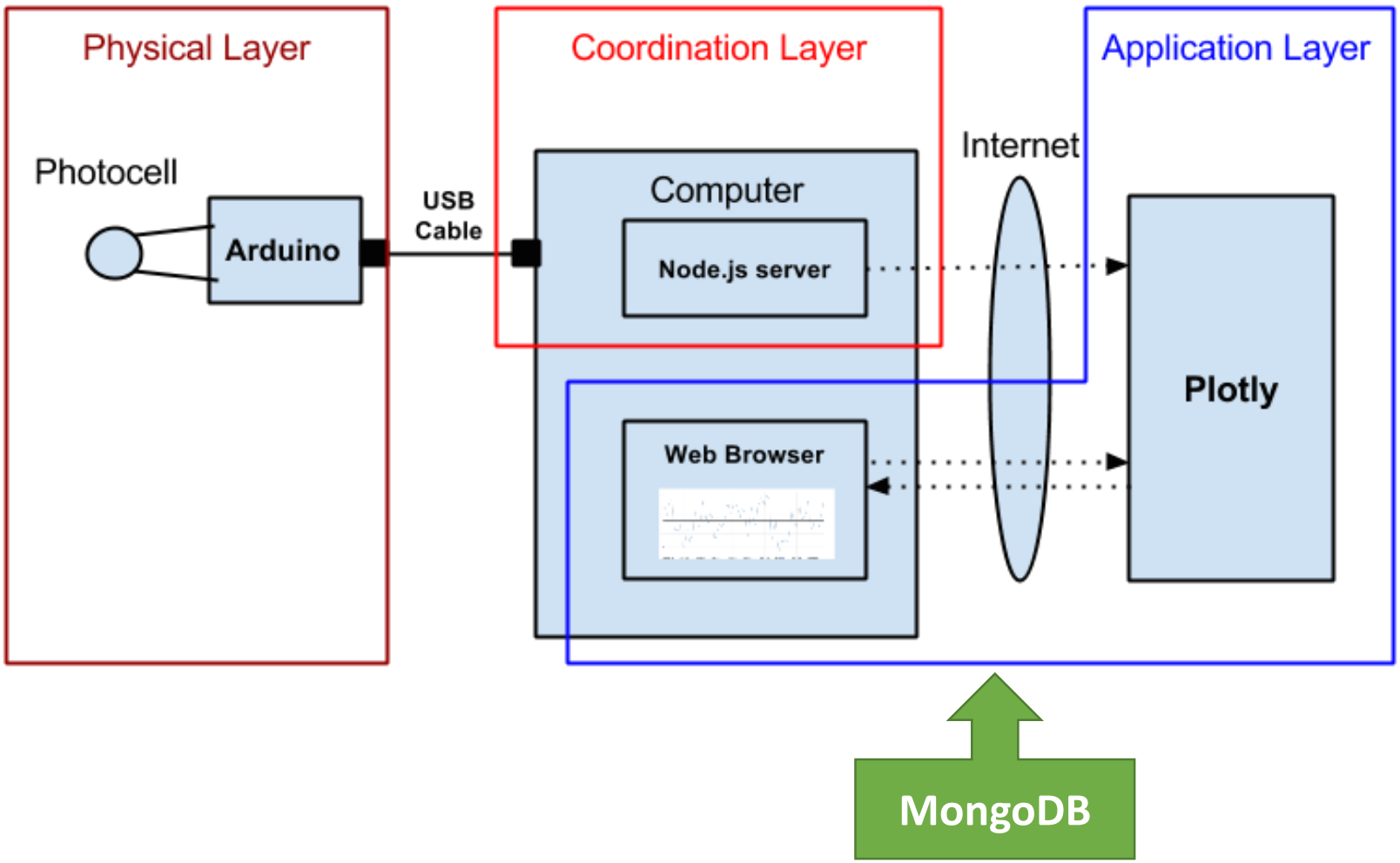


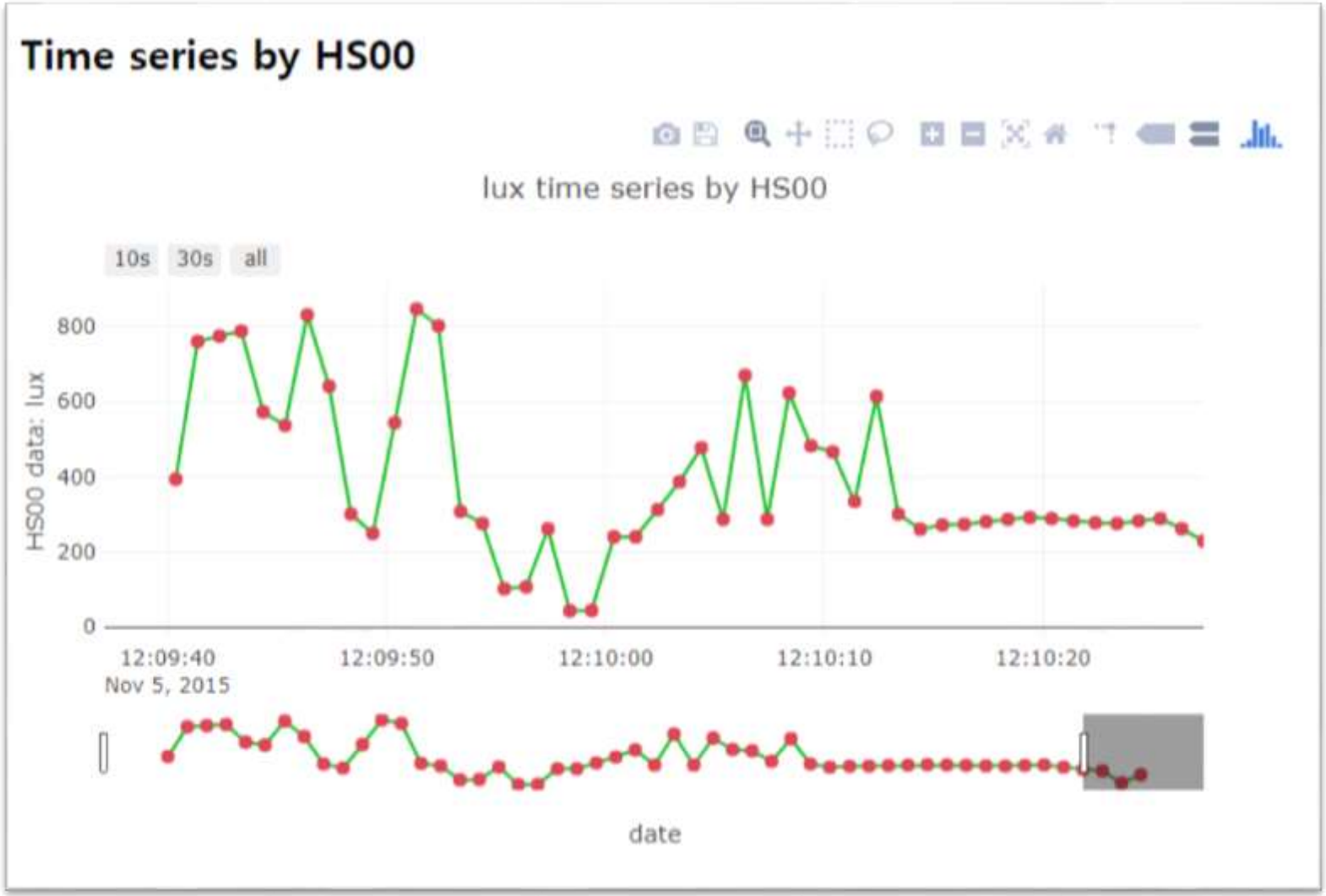
Figure 3
Typical Construction of a Plastic Coated Photocell



Layout [H S C]

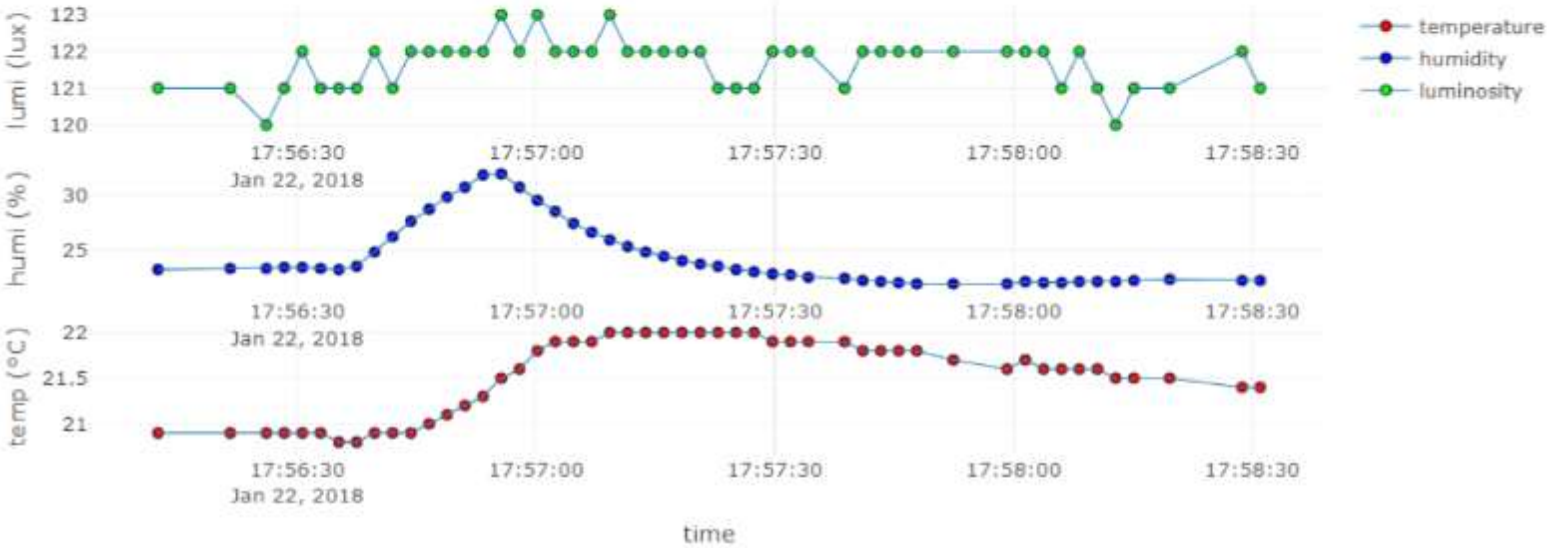

Physical Layer
Photocell
Arduino
USB
Cable
Coordination Layer
Computer
Node.js server
Web Browser
Internet
Application Layer
Plotly
MongoDB

Arduino data + plotly

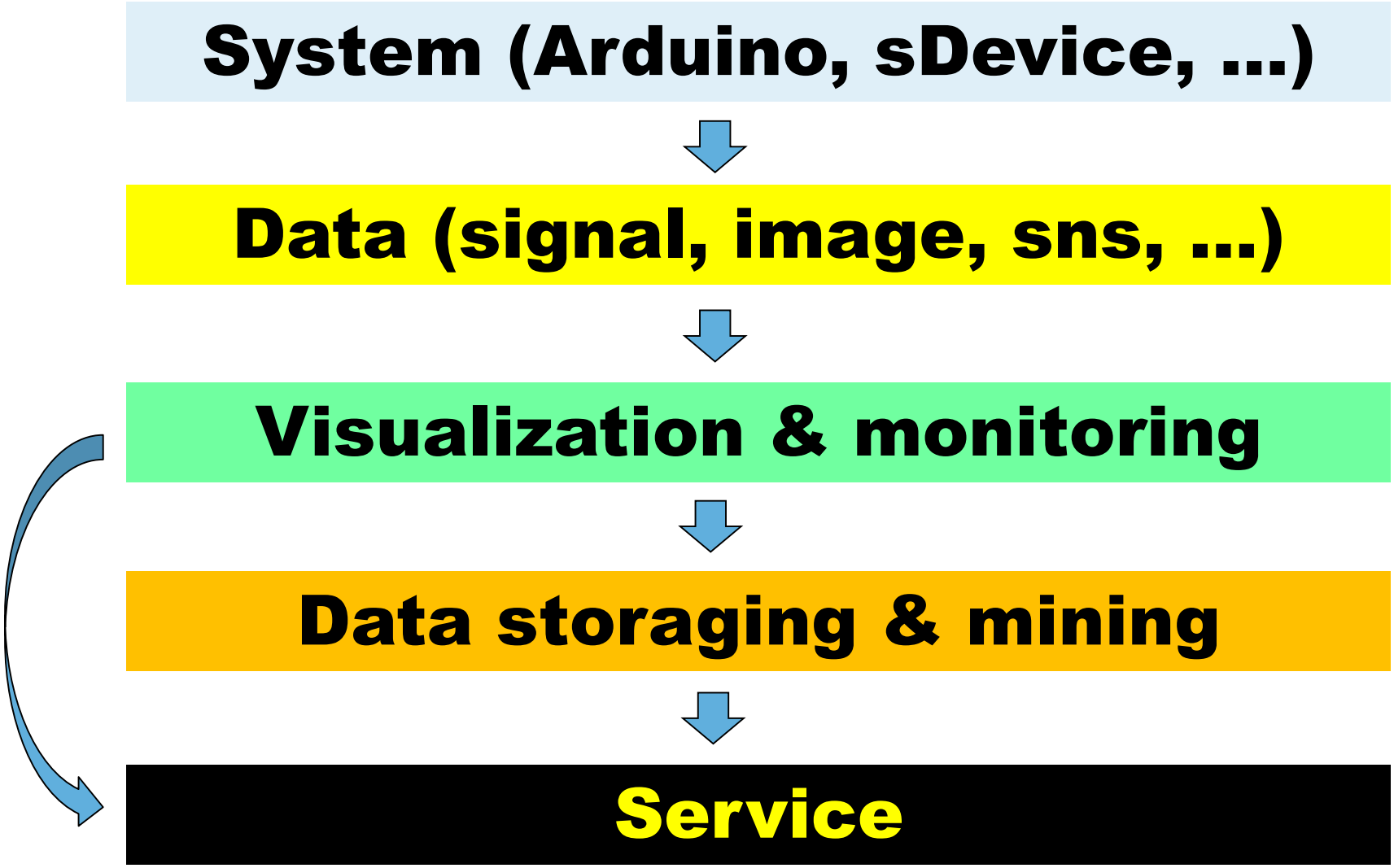


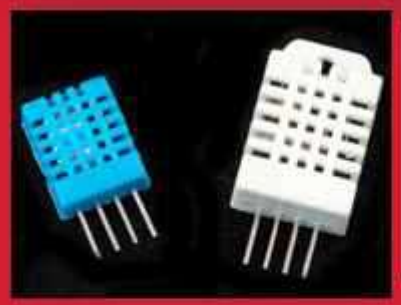
Real-time Weather Station from sensors

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



A5. Introduction to visualization





[Goal]

Arduino + Node.js

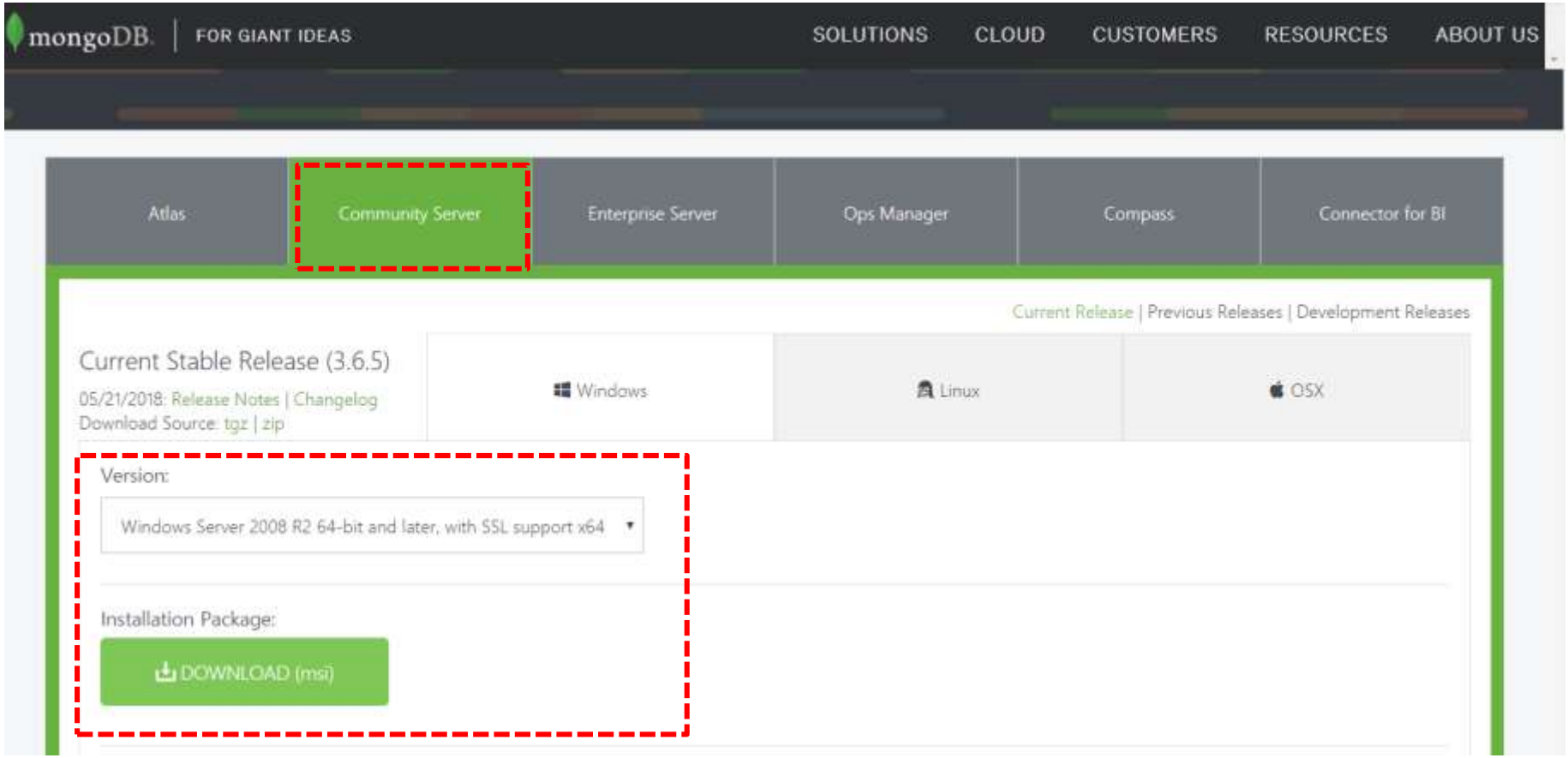
+ plotly.js

+ MongoDB

→ Data storaging

& visualization

A5.9 MongoDB



https://www.mongodb.com/download-center#community

A5.9.1 MongoDB install – 3

MongoDB Compass Community - Connect

Connect View Help

NEW CONNECTION

FAVORITES

RECENTS

LIMITATION, ANY CAUSE OF ACTION SOUNDING IN CONTRACT, TORT, OR STRICT LIABILITY, SHALL NOT EXCEED ONE HUNDRED DOLLARS (U.S. $100.00). THE FOREGOING LIMITATIONS WILL APPLY NOTWITHSTANDING THE FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

10. ESSENTIAL BASIS OF AGREEMENT. The Parties acknowledge and agree that the disclaimers, exclusions and limitations of liability set forth in Section 9 form an essential basis of this Agreement, and that, absent any of such disclaimers, exclusions or limitations of liability, the terms of this Agreement, including, without limitation, the economic terms, would be substantially different.

11. GENERAL. This Agreement shall be governed by and interpreted in accordance with the laws of the State of New York, without regard to conflicts of law principles thereof or to the United Nations Convention on the International Sale of Goods. For purposes of all claims brought under this agreement, each of the parties hereby irrevocably submits to the exclusive jurisdiction of the state and federal courts located within the State of New York. Company may assign this Agreement, in whole or in part, at any time with or without notice to You. You may not assign this Agreement, or any part of it, to any other party. Any attempt by You to do so is null and void. If any provision of this Agreement is held to be unenforceable, that provision will be enforced to the extent permissible by law and the remaining provisions will remain in full force. No waiver under this Agreement shall be valid or binding unless set forth in writing and duly executed by the party against whom enforcement of such waiver is sought. Any such waiver shall constitute a waiver only with respect to the specific matter described therein and shall in no way impair the rights of the party granting such waiver in any other respect or at any other time. Any delay or forbearance by either party in exercising any right hereunder shall not be deemed a waiver of that right. This Agreement is the complete and exclusive statement of the agreement between us and supersedes any proposal or prior agreement, oral or written, and any other communications between You and Company in relation to the subject matter of this Agreement.

If You have any questions regarding this Agreement or the Software, please direct all correspondence to:legal@mongodb.com.

Agree Disagree

CONNECT

https://www.mongodb.com/download-center#community

A5.9.1 MongoDB install – 5.

윈도우10: 설정 > 시스템 > 정보

[중요] 시스템 환경변수 : **PATH** 에 경로 추가

C:\Program Files\MongoDB\Server\3.6\bin

3. Run MongoDB by using mongod.exe

➢ **mongod –dbpath c:\mongodb\data**

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

➢ **mongod –dbpath c:\mongodb\data**

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

사용 PC 환경에 맞게 실행 (특히, 경로 지정)

A5.9.2 MongoDB shell – 4

4. Run mongo shell : mongo.exe [use new cmd]

- **mongo**

Run new cmd

mongo

명령 프롬프트 - mongo

```
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.2
Server has startup warnings:
2018-01-22T19:27:33.549-0700 I CONTROL  [initandlisten]
2018-01-22T19:27:33.549-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabl
ed for the database.
2018-01-22T19:27:33.550-0700 I CONTROL  [initandlisten] **          Read and write access to da
ta and configuration is unrestricted.
2018-01-22T19:27:33.550-0700 I CONTROL  [initandlisten]
2018-01-22T19:27:33.554-0700 I CONTROL  [initandlisten] ** WARNING: This server is bound to loc
alhost.
2018-01-22T19:27:33.557-0700 I CONTROL  [initandlisten] **          Remote systems will be unab
le to connect to this server.
2018-01-22T19:27:33.559-0700 I CONTROL  [initandlisten] **          Start the server with --bin
d_ip <address> to specify which IP
2018-01-22T19:27:33.561-0700 I CONTROL  [initandlisten] **          addresses it should serve r
esponses from, or with --bind_ip_all to
2018-01-22T19:27:33.563-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If
this behavior is desired, start the
2018-01-22T19:27:33.564-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0
.0.1 to disable this warning.
2018-01-22T19:27:33.566-0700 I CONTROL  [initandlisten]
2018-01-22T19:27:33.567-0700 I CONTROL  [initandlisten]
2018-01-22T19:27:33.569-0700 I CONTROL  [initandlisten] ** WARNING: The file system cache of th
is machine is configured to be greater than 40% of the total memory. This can lead to increased
 memory pressure and poor performance.
2018-01-22T19:27:33.570-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-w
indows-system-file-cache
2018-01-22T19:27:33.571-0700 I CONTROL  [initandlisten]
>
```

A5.9.3 MongoDB shell coding

5. update a record

update record2

db.user.find().pretty()

명령 프롬프트 - mongo

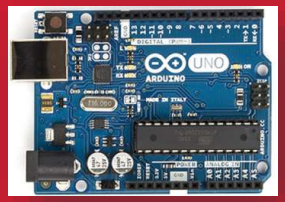
```
> db.user.update({last:"Hong"},{$set:{first:"GilDong", age:21}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.user.find().pretty()
{
        "_id" : ObjectId("5a66b44b9f0d55608f5f7582"),
        "first" : "Redwoods",
        "last" : "Yi"
}
{
        "_id" : ObjectId("5a66b5869f0d55608f5f7584"),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

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

Note that it is possible to change schema.

Save as

HSnn_mongo_update.png



Node.js

+

MongoDB

A5.9.4 MongoDB + Node.js：mongoose

Mongoose ODM v5.0.1

mongoosejs.com

Fork me on GitHub

mongoose

elegant mongodb object modeling for node.js

read the docs | discover plugins

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 : mongooseJS

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

- Go to cds_dht22 project
- npm install --save mongoose

```
NodeJS
D:\Portable\NodeJSPortable\Data\aa00\iot\cds_dht22>npm install -s mongoose
loadRequestedDeps → fetch ? ????????????????????????????????????????????????????????????
loadRequestedDeps → fetch ? ????????????????????????????????????????????????????????????
loadRequestedDeps → netwo ? ????????????????????????????????????????????????????????????
loadRequestedDeps → fetch ? ????????????????????????????????????????????????????????????
loadDep:sliced → request  ? ????????????????????????????????????????????????????????????
loadDep:sliced → 200      ? ????????????????????????????????????????????????????????????
loadDep:sliced → fetch    ? ????????????????????????????????????????????????????????????
loadDep:sliced → headers  ? ????????????????????????????????????????????????????????????
loadDep:sliced → fetch    ? ????????????????????????????????????????????????????????????
loadDep:sliced → fetch    ? ????????????????????????????????????????????????????????????
loadDep:sliced → get      ? ????????????????????????????????????????????????????????????
loadDep:sliced → afterAdd ? ????????????????????????????????????????????????????????????
extract:mongoose → gunzla ? ????????????????????????????????????????????????????????????
extract:mongoose → gently ? ????????????????????????????????????????????????????????????
finalize:sliced → finaliz ? ????????????????????????????????????????????????????????????
build:resolve-from → link ? ????????????????????????????????????????????????????????????
cds_dht22@1.0.0 D:\Portable\NodeJSPortable\Data\aa00\iot\cds_dht22
└──mongoose@5.0.1  extraneous


D:\Portable\NodeJSPortable\Data\aa00\iot\cds_dht22>
```

A5.9.4 MongoDB + Node.js : mongoose

4. dbtest2.js (use Sublime Text 3)

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

D:\Portable\NodeJSPortable\Data\aa00\iot\cds_dht22\dbtest2.js (Data) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS
- Data
  - aa00
    - express
    - expressTest
    - iot
      - cds
        - node_modules
        - cds_node.js
        - package.json
      - cds_dht22
        - node_modules
        - cds_dht22_node.js
        - dbtest.js
        - dbtest2.js
        - package.json
      - cds_tmp36
      - plotly
      - tmp36
    - myApp
    - server
    - start
  - node_modules
  - npm_cache
  - settings
  - Temp
  - express
  - express.cmd
  - npm
  - npm.cmd
  - PortableApps.comLauncherRuntimeData-NodeJSPo

dbtest2.js

```
// 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
```

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

Mongo shell

> show dbs

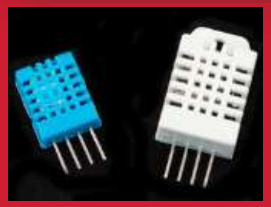
> use test2

> show collections

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

명령 프롬프트 - mongo

```
> 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

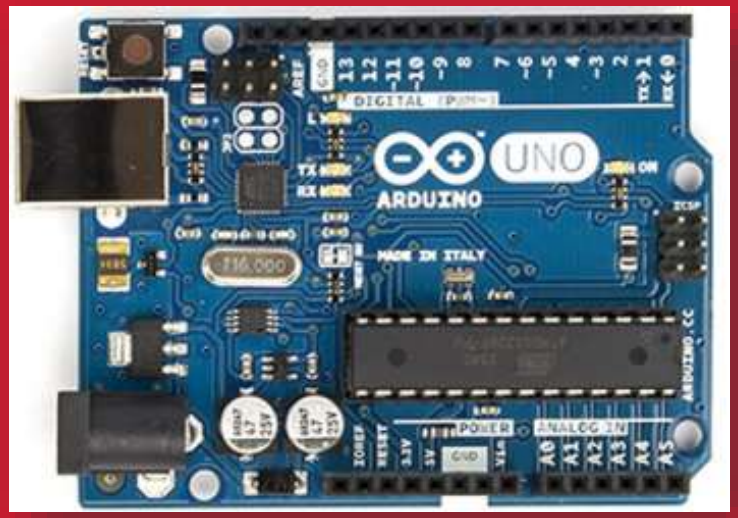
```
> show dbs
aa00     0.000GB
admin    0.000GB
config   0.000GB
iot      0.000GB
iot2     0.000GB
iot3     0.001GB
local    0.000GB
test     0.000GB
test2    0.000GB
>
```

```
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

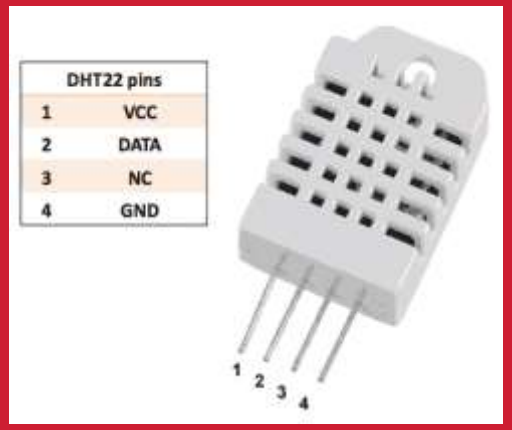
& Node.js

& MongoDB

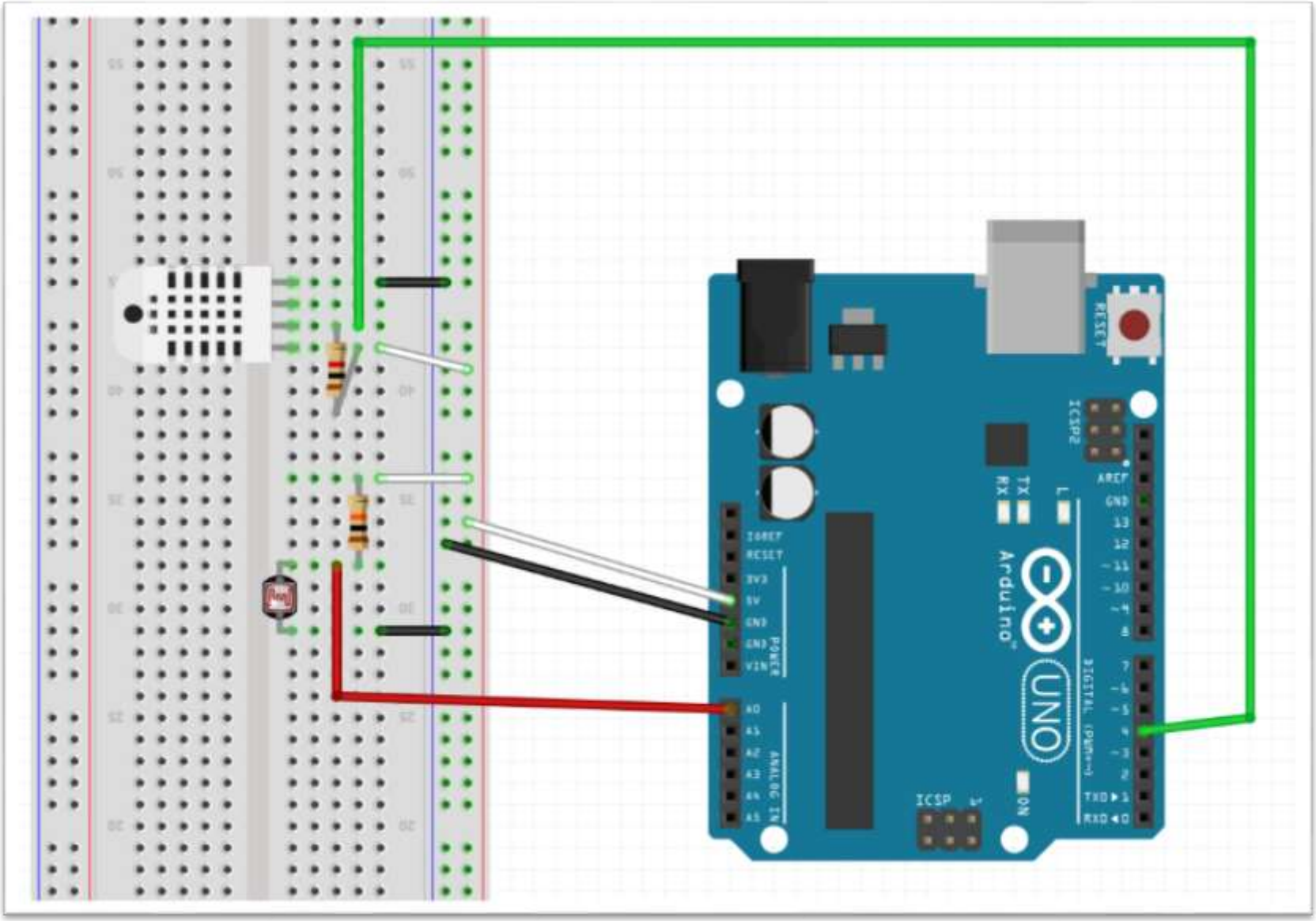


Multi-sensors

DHT22 + CdS



DHT22 + CdS : circuit



A5.7.4 DHT22 + CdS : circuit

[1] Arduino code: HSnn_CdS_DHT22.ino

HSnn_CdS_DHT22

```
// DHT22
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
// CdS (LDR)
#define CDS_INPUT 0

void setup() {
  dht.begin();
  Serial.begin(9600);
}
```

```
//Voltage to Lux
double luminosity (int RawADC0){
  double Vout=RawADC0*5.0/1023.0;  // 5/1023
  double lux=(2500/Vout-500)/10;
  // lux = 500 / Rldr,
  // Vout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
  return lux;
}
```

```
void loop() {
  int cds_value, lux;
  float temp, humi;
  // Lux from CdS (LDR)
  cds_value = analogRead(CDS_INPUT);
  lux = int(luminosity(cds_value));
  // Reading temperature or humidity takes a given interval!
  // Sensor readings may also be up to 2 seconds 'old'
  humi = dht.readHumidity();
  // Read temperature as Celsius (the default)
  temp = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(humi) || isnan(temp) || isnan(lux)) {
    Serial.println("Failed to read from DHT sensor or CdS!");
    return;
  }
  else {
    Serial.print("HS00,");
    Serial.print(temp,1);  // temperature, float
    Serial.print(",");
    Serial.print(humi,1);  // humidity, float
    Serial.print(",");
    Serial.println(lux);   // luminosity, int
  }
  delay(2000);  // 2000 msec, 0.5 Hz
}
```

A5.7.10 DHT22 + CdS + Node.js

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

COM4

```
AA00,20.9,21.9,117
AA00,20.9,21.8,117
AA00,20.9,21.8,118
AA00,20.9,21.8,118
AA00,20.9,21.8,119
AA00,20.9,21.8,118
AA00,20.9,21.8,118
AA00,20.9,21.8,118
AA00,20.9,21.9,118
AA00,20.9,21.9,118
AA00,20.8,21.9,118
AA00,20.9,22.0,118
AA00,20.9,22.0,118
AA00,20.8,21.8,119
```

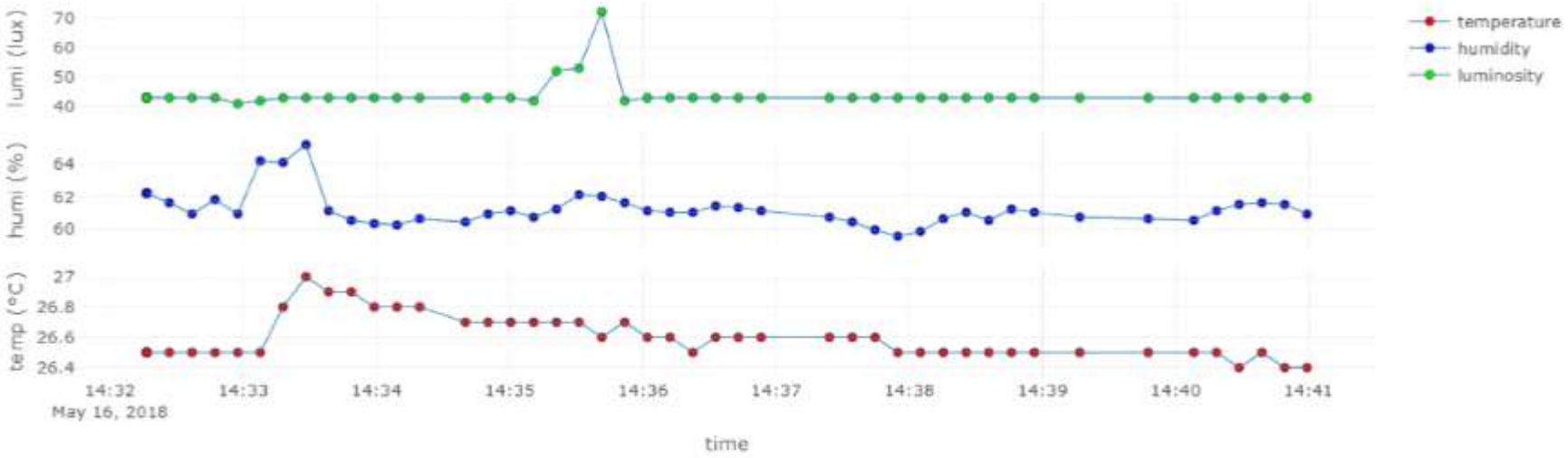
☑ 자동 스크롤

NodeJS - node cds_dht22_node

```
D:\Portable\NodeJSPortable\Data\aa00\iot\cds_dht22>node cds_dht22_node
[ '2018-01-22 17:22:47.683', '20.7', '23.2', '118' ]
[ '2018-01-22 17:22:49.954', '20.6', '23.2', '116' ]
[ '2018-01-22 17:22:52.227', '20.7', '23.2', '117' ]
[ '2018-01-22 17:22:54.486', '20.7', '23.2', '116' ]
[ '2018-01-22 17:22:56.757', '20.6', '23.2', '117' ]
[ '2018-01-22 17:22:59.031', '20.7', '23.3', '117' ]
[ '2018-01-22 17:23:01.306', '20.7', '23.3', '117' ]
[ '2018-01-22 17:23:03.577', '20.7', '23.3', '117' ]
[ '2018-01-22 17:23:05.851', '20.7', '23.3', '118' ]
[ '2018-01-22 17:23:08.109', '20.6', '23.2', '115' ]
[ '2018-01-22 17:23:10.381', '20.6', '23.2', '113' ]
[ '2018-01-22 17:23:12.655', '20.7', '23.5', '114' ]
[ '2018-01-22 17:23:14.928', '20.7', '23.7', '38' ]
[ '2018-01-22 17:23:17.201', '20.6', '23.9', '117' ]
[ '2018-01-22 17:23:19.475', '20.7', '24.5', '117' ]
[ '2018-01-22 17:23:21.732', '20.7', '25.9', '73' ]
[ '2018-01-22 17:23:24.004', '20.7', '34.2', '118' ]
[ '2018-01-22 17:23:26.277', '21.3', '55.5', '117' ]
[ '2018-01-22 17:23:28.553', '21.0', '68.1', '117' ]
[ '2018-01-22 17:23:30.825', '20.9', '76.1', '117' ]
[ '2018-01-22 17:23:33.083', '21.0', '74.0', '116' ]
[ '2018-01-22 17:23:35.355', '21.0', '65.7', '117' ]
[ '2018-01-22 17:23:37.628', '21.0', '57.7', '116' ]
[ '2018-01-22 17:23:39.901', '21.0', '51.2', '116' ]
[ '2018-01-22 17:23:42.175', '21.0', '45.9', '117' ]
[ '2018-01-22 17:23:44.448', '21.0', '41.6', '117' ]
[ '2018-01-22 17:23:46.706', '21.0', '38.3', '116' ]
[ '2018-01-22 17:23:48.979', '21.0', '35.8', '118' ]
```

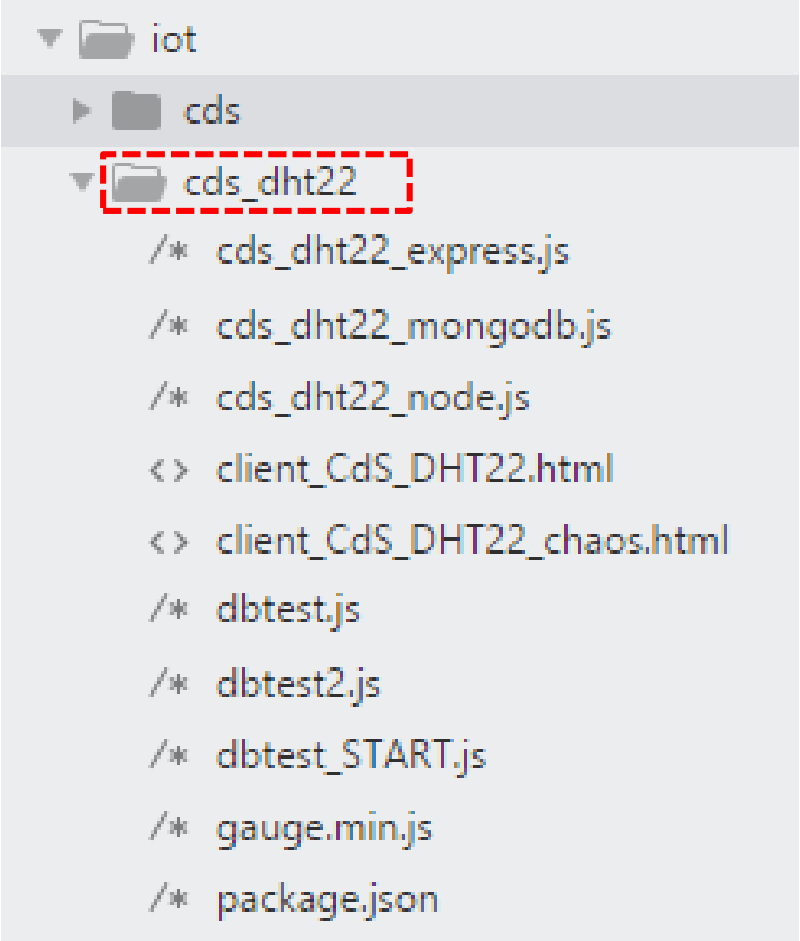
Real-time Weather Station from sensors

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



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

1. 작업 폴더 구조 [2018]



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

2.1 cds_dht22_mongodb.js (cds_dht22_node.js 를 MongoDb 용으로 변경)

```
1  // cds_dht22_mongodb.js
2
3  var serialport = require('serialport');
4  var portName = 'COM4';  // check your COM port!!
5  var port     =   process.env.PORT || 3000;
6
7  var io = require('socket.io').listen(port);
8
9  // MongoDB
10 var mongoose = require('mongoose');
11 var Schema = mongoose.Schema;
12 // MongoDB connection
13 mongoose.connect('mongodb://localhost:27017/iot'); // DB name
14     var db = mongoose.connection;
15     db.on('error', console.error.bind(console, 'connection error:'));
16     db.once('open', function callback () {
17         console.log("mongo db connection OK.");
18     });
19 // Schema
20 var iotSchema = new Schema({
21     date : String,
22     temperature : String,
23     humidity : String,
24     luminosity: String
25 });
```

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

2.2 cds_dht22_mongodb.js

```
27 iotSchema.methods.info = function () {
28     var iotInfo = this.date
29     ? "Current date: " + this.date +", Temp: " + this.temperature
30     + ", Humi: " + this.humidity + ", Lux: " + this.luminosity
31     : "I don't have a date"
32     console.log("iotInfo: " + iotInfo);
33 }
34
35 // serial port object
36 var sp = new serialport(portName,{
37     baudRate: 9600,   // 9600  38400
38     dataBits: 8,
39     parity: 'none',
40     stopBits: 1,
41     flowControl: false,
42     parser: serialport.parsers.readline('\r\n')  // new serialport.parsers
43 });
44
45 var readData = '';  // this stores the buffer
46 var temp ='';
47 var humi ='';
48 var lux ='';
49 var mdata =[]; // this array stores date and data from multiple sensors
50 var firstcommaidx = 0;
51
52 var Sensor = mongoose.model("Sensor", iotSchema);  // sensor data model
```

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

2.3 cds_dht22_mongodb.js

```
54 sp.on('data', function (data) { // call back when data is received
55     readData = data.toString(); // append data to buffer
56     firstcommaidx = readData.indexOf(',');
57
58     // parsing data into signals
59     if (readData.lastIndexOf(',') > firstcommaidx && firstcommaidx > 0) {
60         temp = readData.substring(firstcommaidx + 1, readData.indexOf(',',firstcommaidx+1));
61         humi = readData.substring(readData.indexOf(',',firstcommaidx+1) + 1, readData.lastIndexOf(','));
62         lux = readData.substring(readData.lastIndexOf(',')+1);
63
64         readData = '';
65
66         dStr = getDateString();
67         mdata[0]=dStr;  // Date
68         mdata[1]=temp;  // temperature data
69         mdata[2]=humi;  // humidity data
70         mdata[3]=lux;   // luminosity data
71         //console.log(mdata);
72         var iot = new Sensor({date:dStr, temperature:temp, humidity:humi, luminosity:lux});
73         // save iot data to MongoDB
74         iot.save(function(err, iot) {
75             if(err) return handleEvent(err);
76             iot.info();  // Display the information of iot data  on console.
77         })
78         io.sockets.emit('message', mdata);  // send data to all clients
79     } else {  // error
80         console.log(readData);
81     }
82 });
```

2.4 cds_dht22_mongodb.js

```
85  io.sockets.on('connection', function (socket) {
86      // If socket.io receives message from the client browser then
87      // this call back will be executed.
88      socket.on('message', function (msg) {
89          console.log(msg);
90      });
91      // If a web browser disconnects from Socket.IO then this callback
92      socket.on('disconnect', function () {
93          console.log('disconnected');
94      });
95  });
96
97  // helper function to get a nicely formatted date string
98  function getDateString() {
99      var time = new Date().getTime();
100     // 32400000 is (GMT+9 Korea, GimHae)
101     // for your timezone just multiply +/-GMT by 3600000
102     var datestr = new Date(time +32400000).
103     toISOString().replace(/T/, ' ').replace(/Z/, '');
104     return datestr;
105 }
```

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

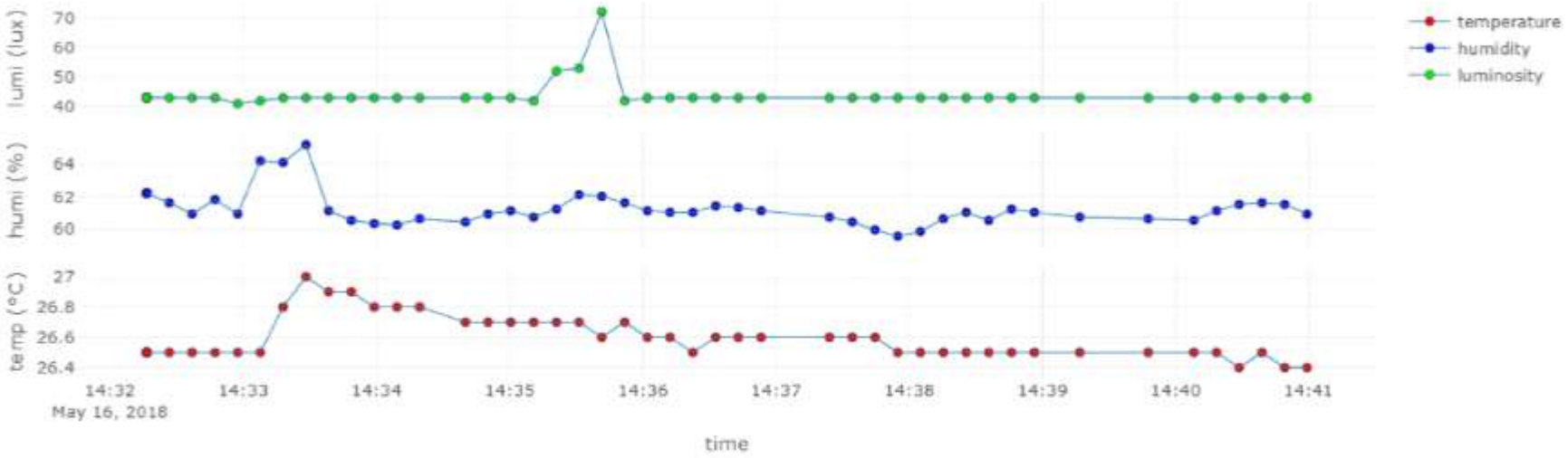
2.5 cds_dht22_mongodb.js → result (^B)

```
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
```

동작 중인 **MongoDB**에 **Sensor** 객체에 담긴 데이터**(iot)**를 저장하면서, 동시에 **socket**으로 데이터 배열 **(mdata)**를 네트워크에 전파한다.

Real-time Weather Station from sensors

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



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

3. cds_dht22_mongodb.js → Check documents in Mongo shell

Mongo shell

> show dbs

> use iot

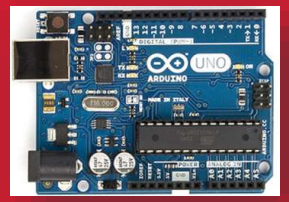
> show collections

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

명령 프롬프트 - mongo

```
> 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
HSnn_iot_mongdb.png



Arduino

& Node.js

& MongoDB

& Express server

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

```
1  // cds_dht22_express.js
2
3  // Express
4  var express = require('express');
5  var app = express();
6  var web_port = 3030;  // express port
7
8  // MongoDB
9  var mongoose = require('mongoose');
10 var Schema = mongoose.Schema;  // Schema object
11 // MongoDB connection
12 mongoose.connect('mongodb://localhost:27017/iot'); // DB name
13 var db = mongoose.connection;
14 db.on('error', console.error.bind(console, 'connection error:'));
15 db.once('open', function callback () {
16         console.log("mongo db connection OK.");
17 });
18 // Schema
19 var iotSchema = new Schema({
20     date : String,
21     temperature : String,
22     humidity : String,
23     luminosity: String
24 });
25 var Sensor = mongoose.model("Sensor", iotSchema);  // sensor data model
```

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

2.2 cds_dht22_express.js

```
27 // Web routing addrebss
28 app.get('/', function (req, res) {  // localhost:3030/
29   res.send('Hello Arduino IOT!');
30 });
31 // find all data & return them
32 app.get('/iot', function (req, res) {
33     Sensor.find(function(err, data) {
34         res.json(data);
35     });
36 });
37 // find data by id
38 app.get('/iot/:id', function (req, res) {
39     Sensor.findById(req.params.id, function(err, data) {
40         res.json(data);
41     });
42 });
43
44 // Express WEB
45 app.use(express.static(__dirname + '/public'));  // WEB root folder
46 app.listen(web_port);  // port 3030
47 console.log("Express_IOT is running at port:3030");
```

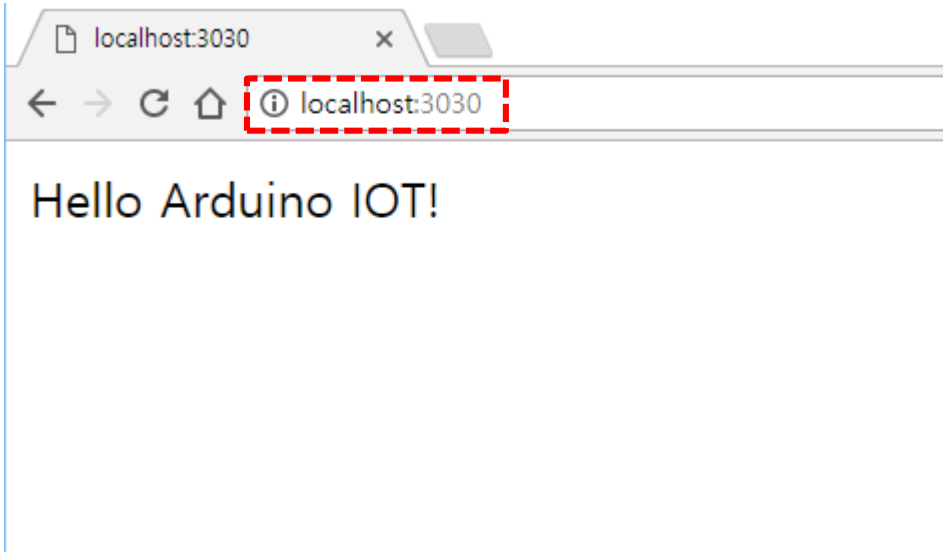
A5.9.6 DHT22 + CdS + Node.js + MongoDB

2.3 cds_dht22_express.js → Run (새로운 Node cmd에서 추가로 실행)

node cds_dht22_express

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

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

localhost:3030/iot

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","tempe
```


Save as

HSnn_iot_mongodb_web.png

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

2.6 cds_dht22_express.js → routing2 http://localhost:3030/iot:id

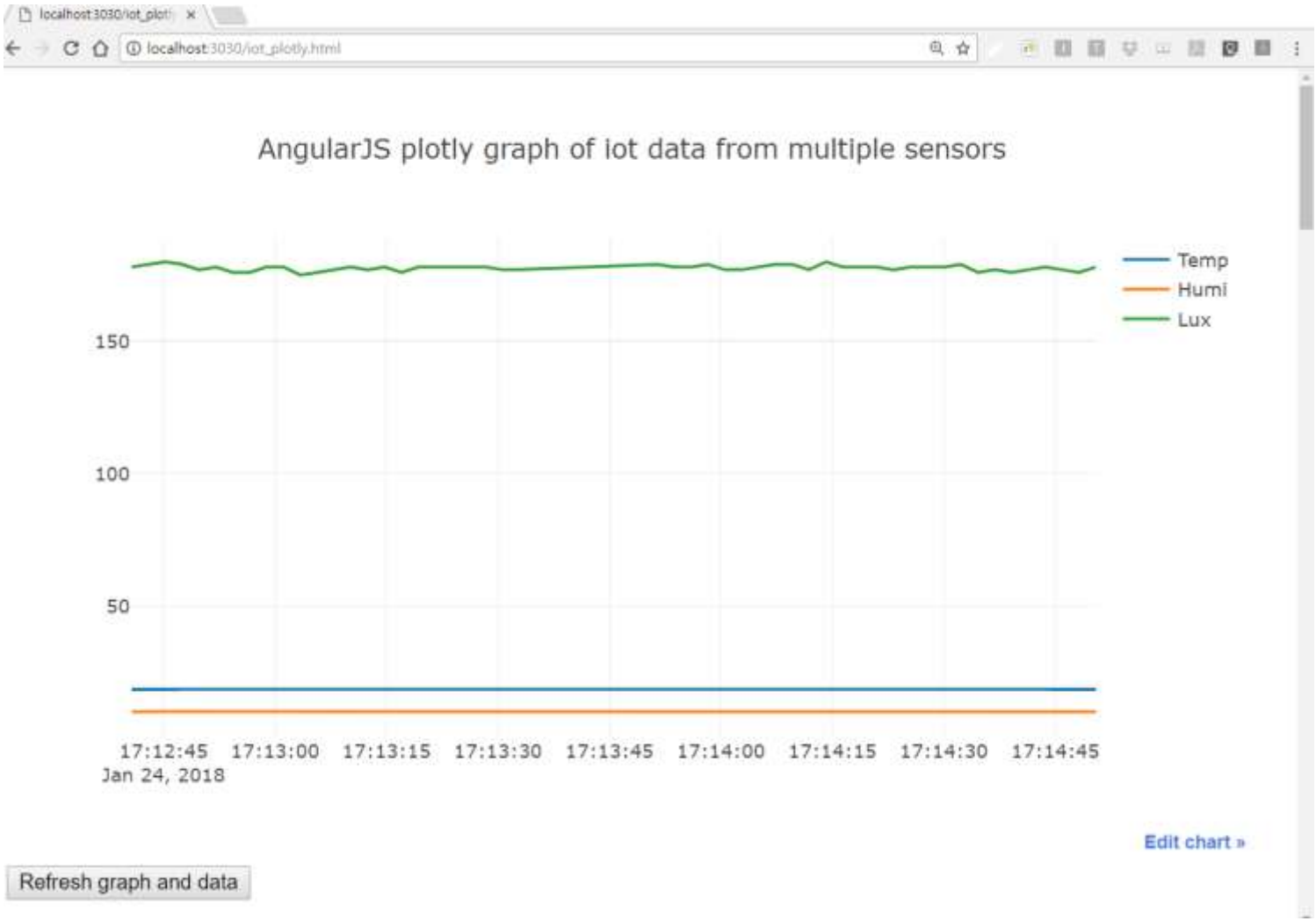
localhost:3030/iot/5a68

localhost:3030/iot/5a683ffd3cdf6353104a5465

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

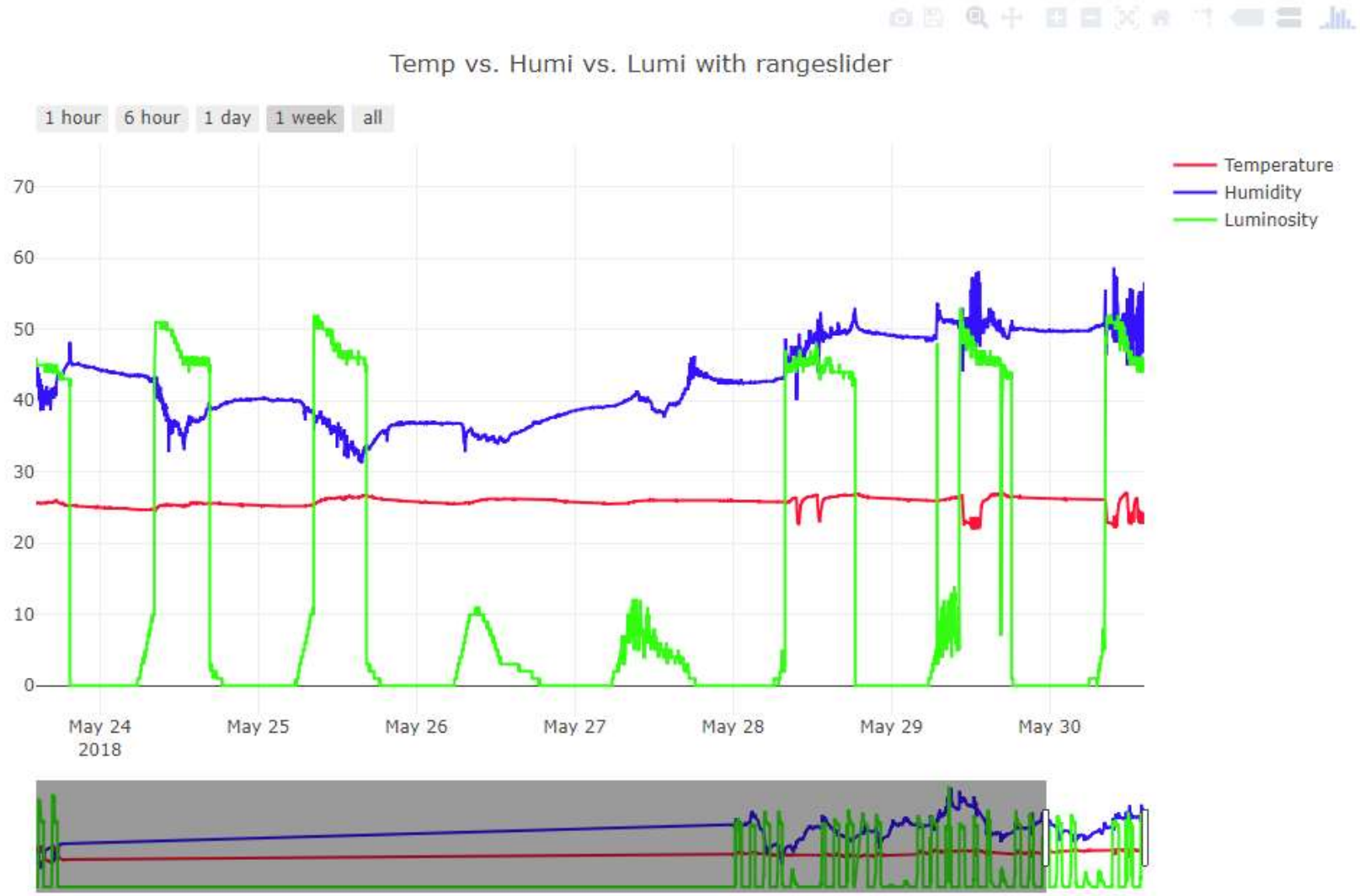
DHT22 + CdS + Node.js + MongoDB

Web monitoring (Old version using AngularJS & more)



MongoDB database visualization by HS00

Time series : Multi sensor data



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

3.1 Web client: client_iotDB.html

client_iotDB.html

```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 HS00</h1>
    <hr>
    <h2>Time series : Multi sensor data</h2>

    <!-- Plotly chart will be drawn inside this DIV -->
    <div id="myDiv" style="width: 1000px;height: 700px"></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;

        }
```

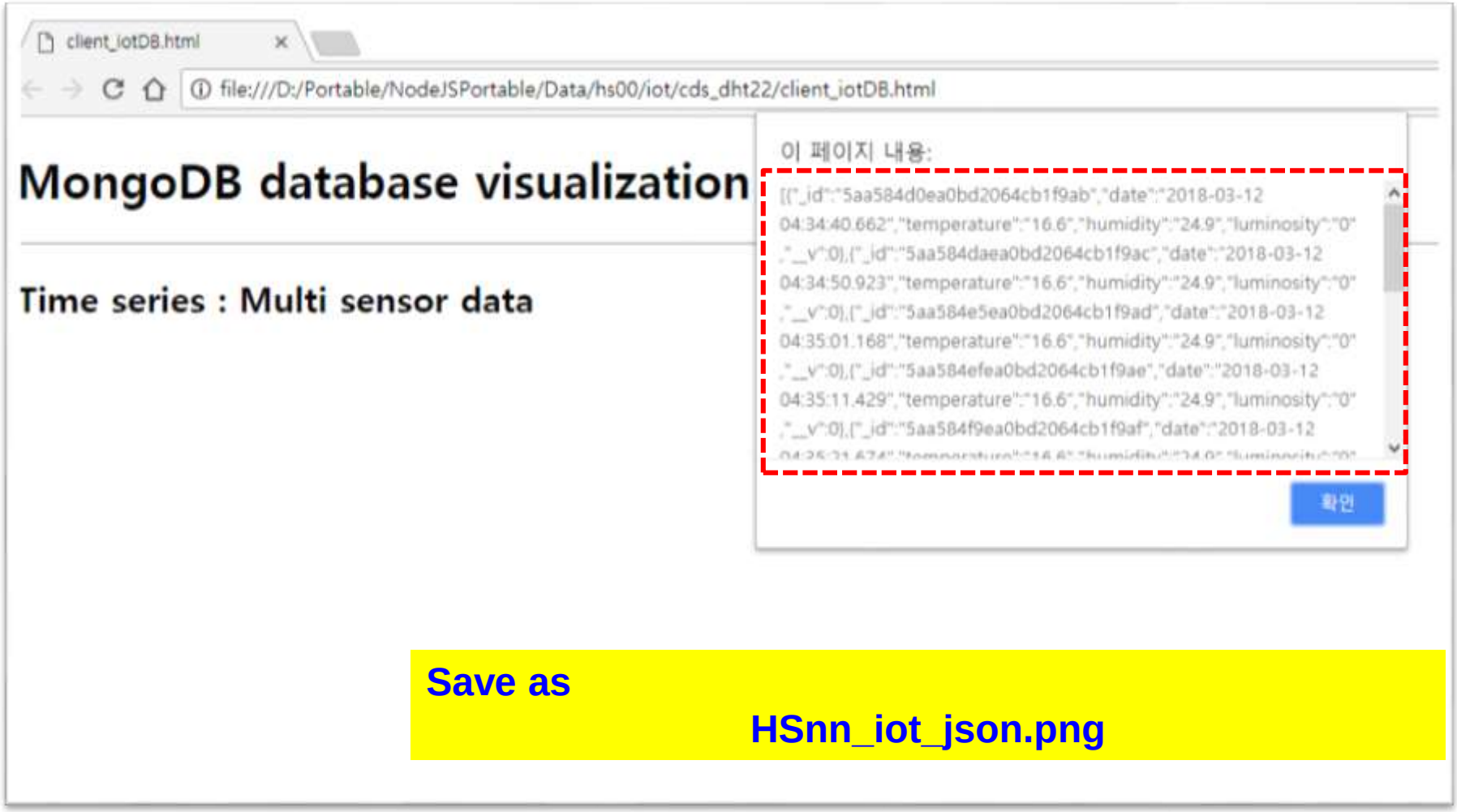
3.3 Web client: client_iotDB.html – data & layout

```
// 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);
})
```

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

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

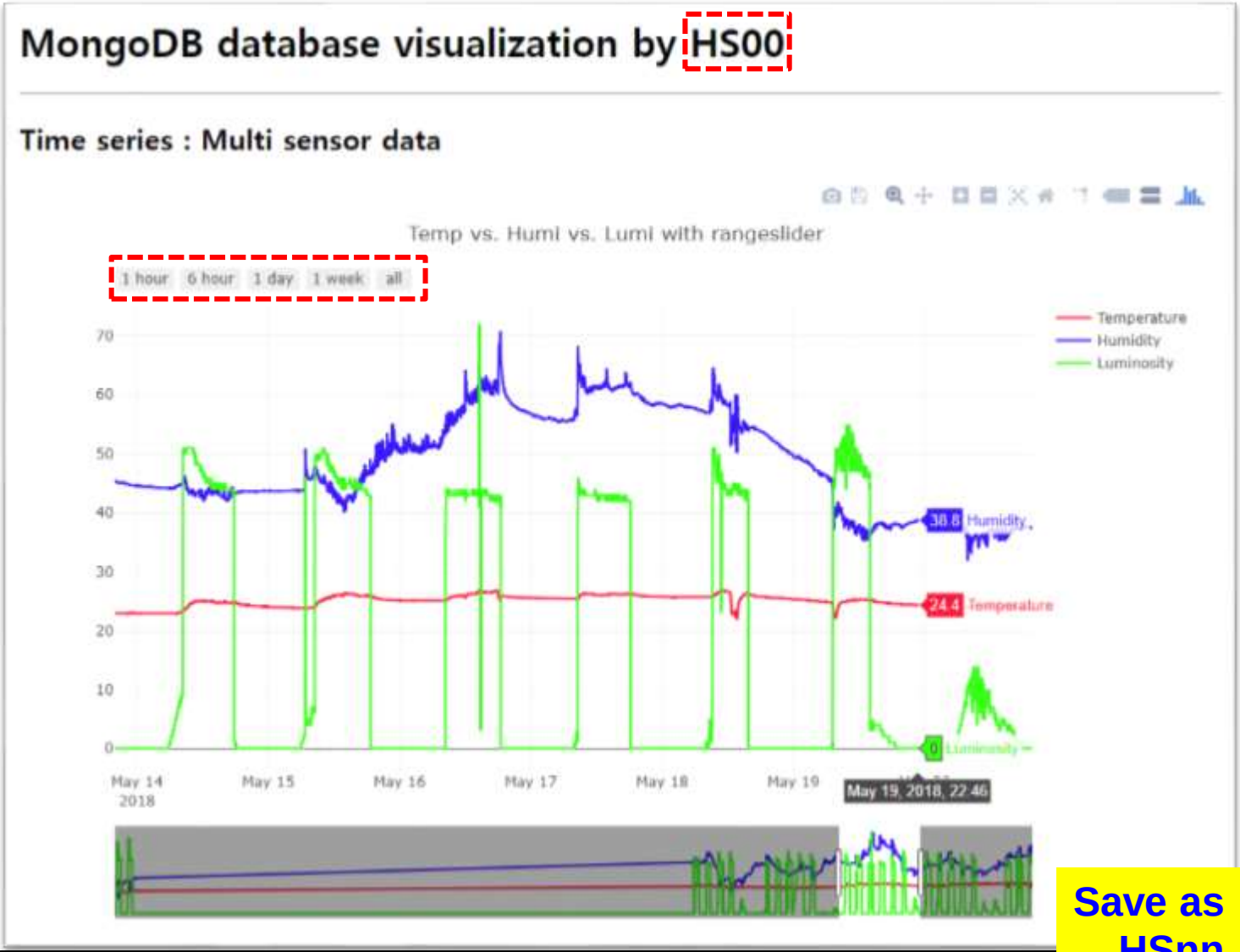


Save as

HSnn_iot_json.png

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

3.5 Web client: client_iotDB.html – iot DB monitoring



Save as HSnn_iot_DB.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: "2018-05-29 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

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 **10**개 추출

명령 프롬프트 - mongo

```
> 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**

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

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
2018-05-27T21:49:00.083+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
>
```

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**

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)
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**

명령 프롬프트

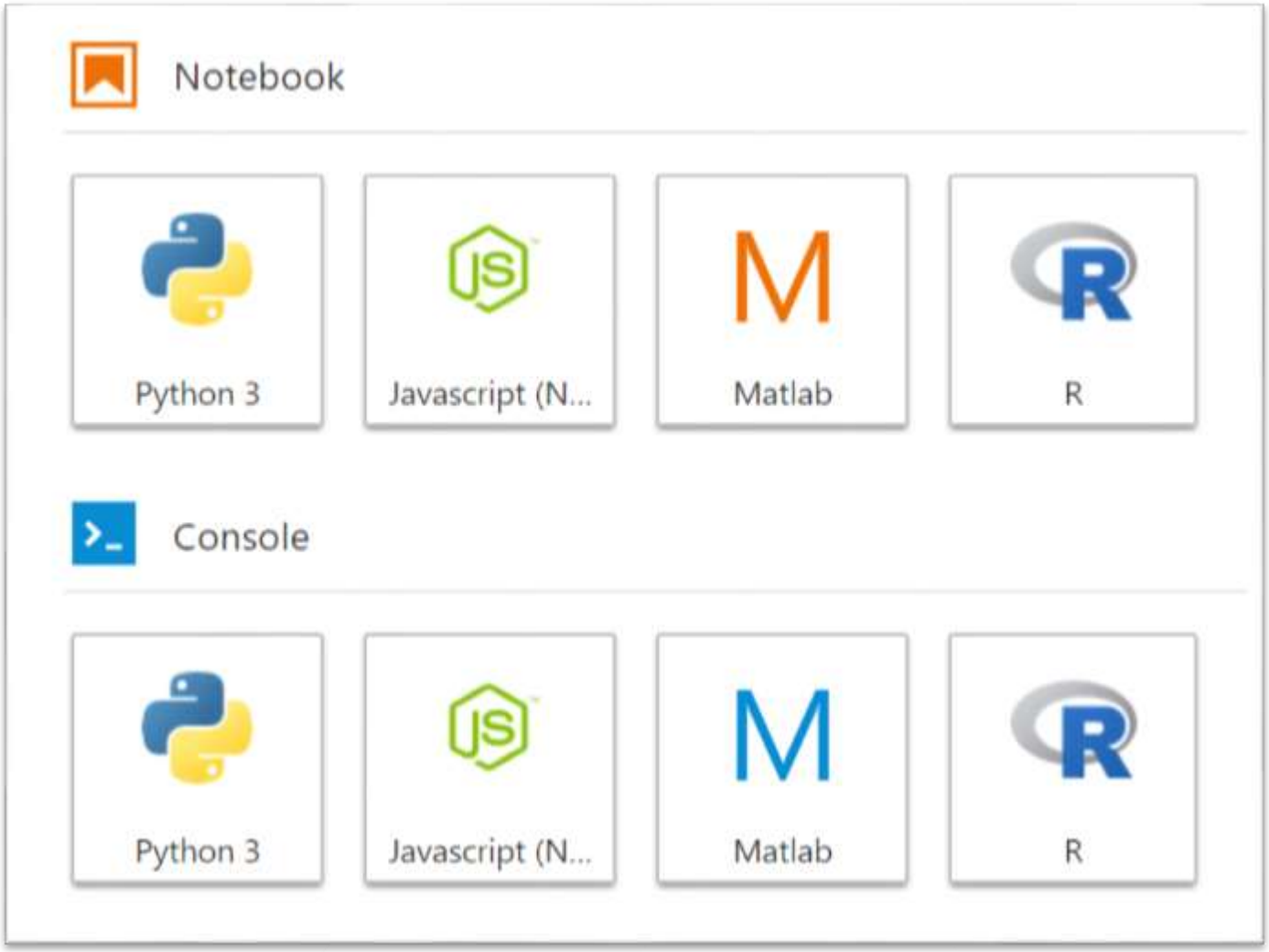
```
C:₩Users₩biochaos>mongoexport -d iot11 -c sensors /query:"{date:{$gt:'2018-05-29 22:26:05'}}" --limit 100 --fields date,
temperature,humidity,luminosity --type=csv --out sensor100.csv
2018-05-29T22:49:19.431+0900    connected to: localhost
2018-05-29T22:49:19.576+0900    exported 100 records
```

[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

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



A5.9.8 MongoDB management

3.1 How to use and understand iot data? → csv_dht22_R.ipynb

Redwoods / Lec

Code | Issues 0 | Pull requests 0 | Projects 0 | Insights

Branch: master | Lec / healthcare-signal-iot / src / iot_data /

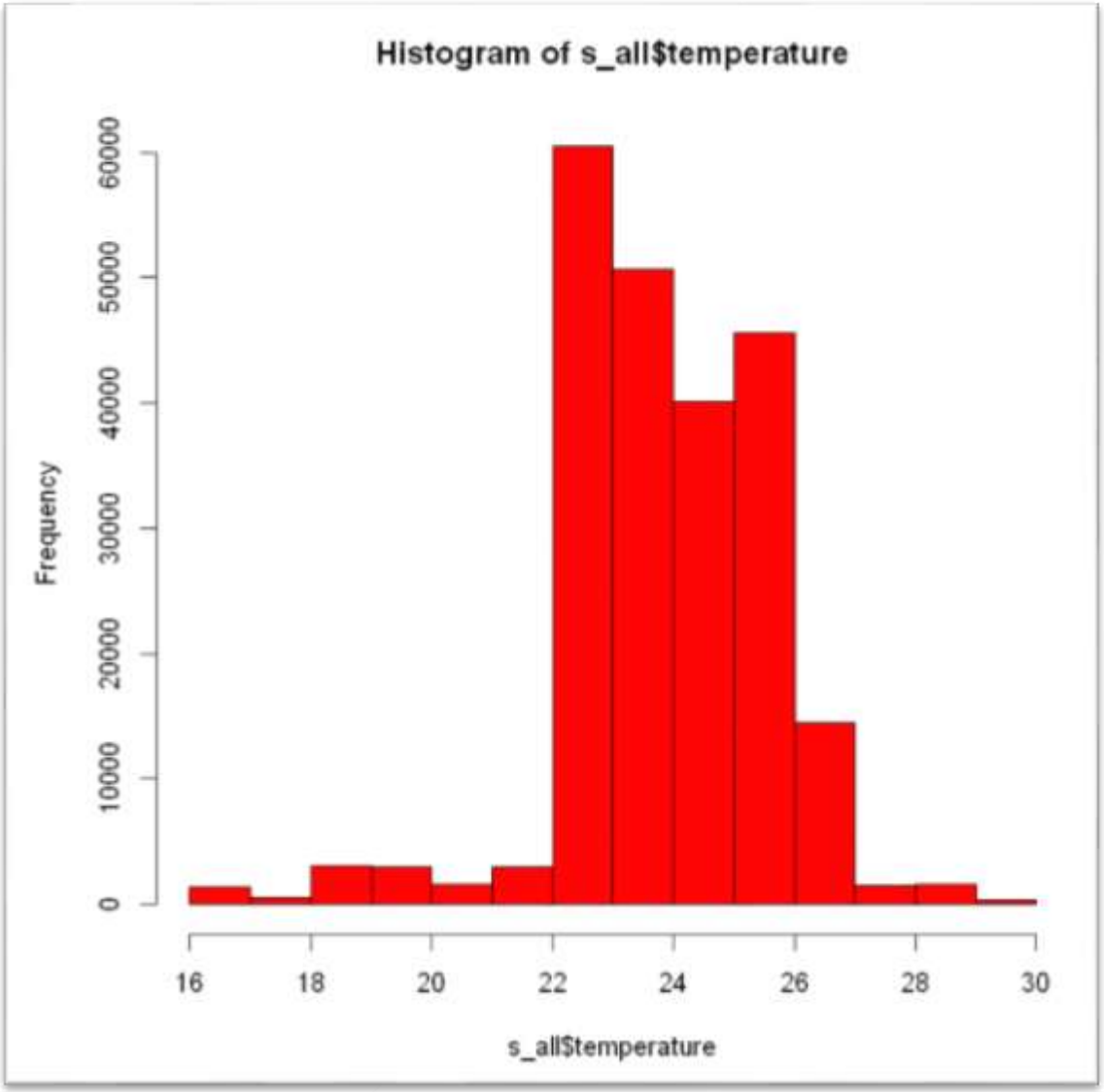
Redwoods ensor iot data analysis-1

..

data	sensor iot data analysis-1
csv_dht22_Py.ipynb	sensor iot data analysis-1
csv_dht22_R.ipynb	ensor iot data analysis-1

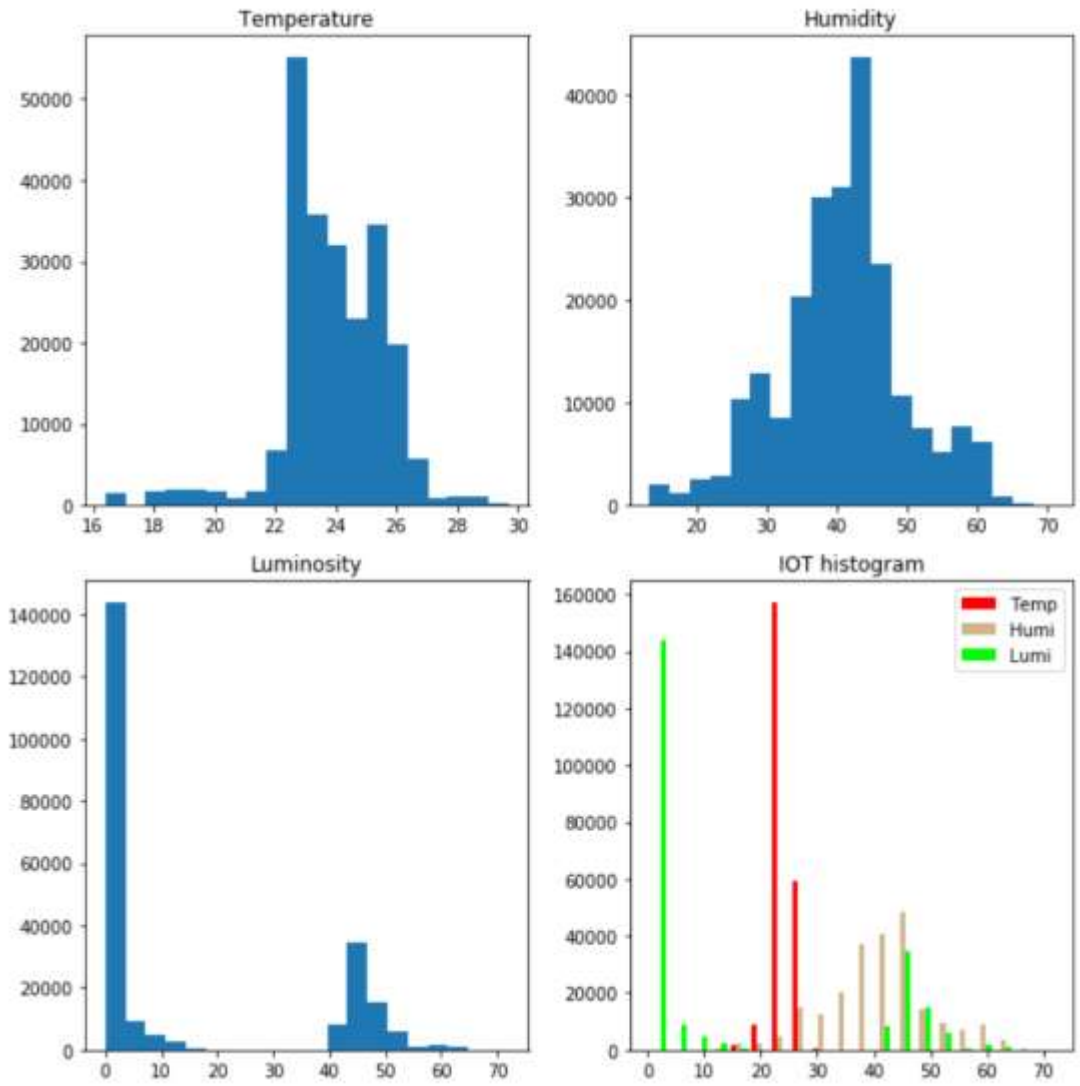
A5.9.8 MongoDB management

3.2 How to use and understand iot data? → csv_dht22_R.ipynb



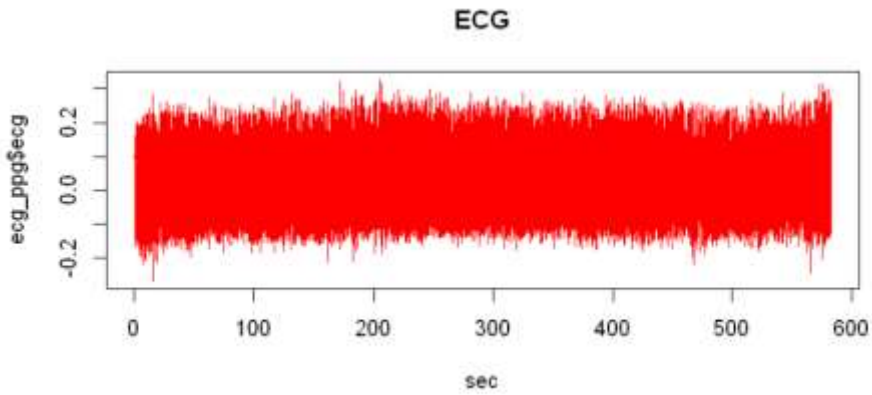
A5.9.8 MongoDB management

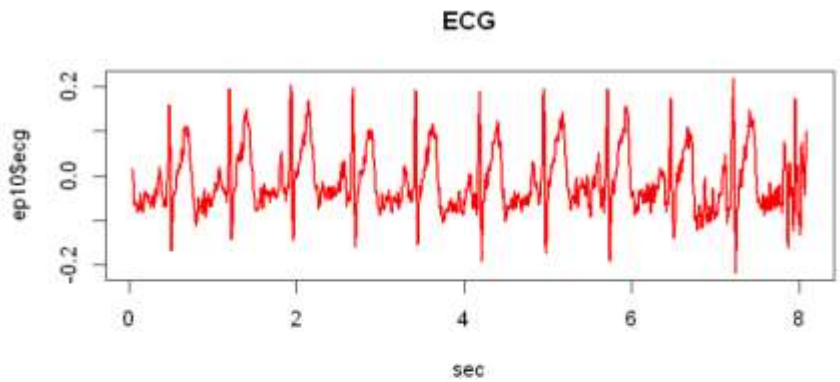
3.3 How to use and understand iot data? → csv_dht22_Py.ipynb

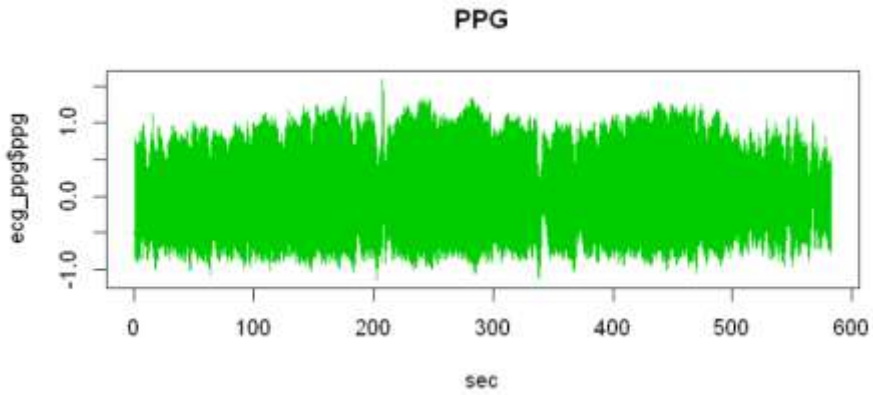


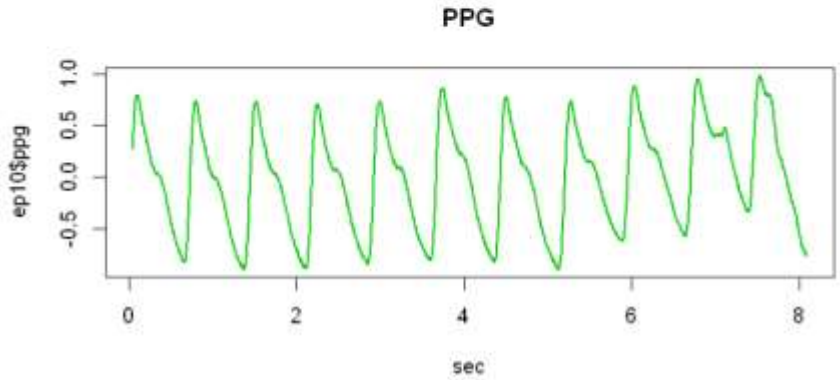
A5.9.8 MongoDB management

3.4 How to use and understand iot data? → ecg_ppg_R.ipynb



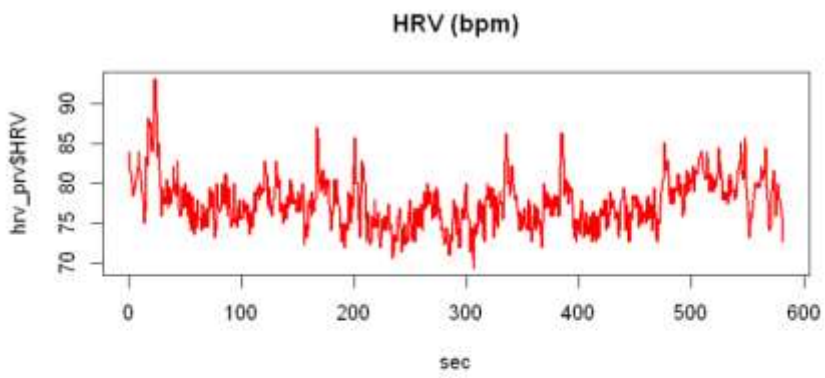


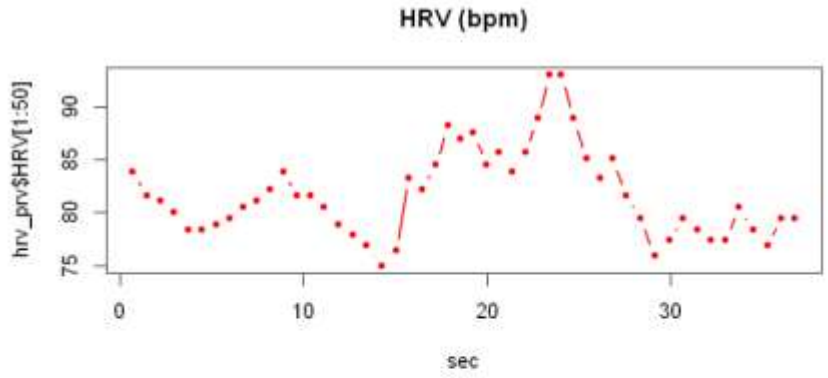


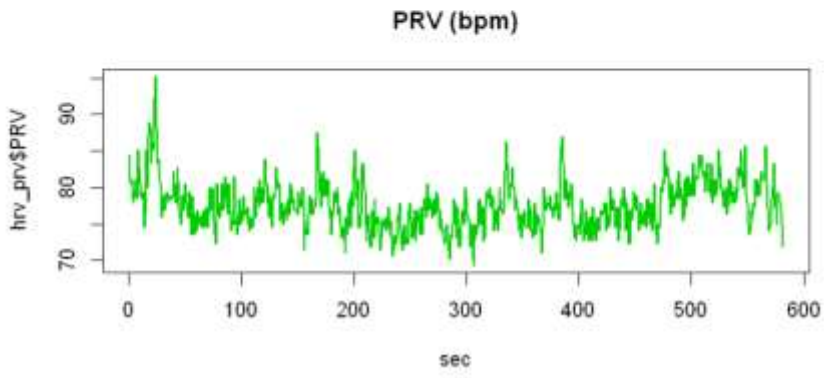


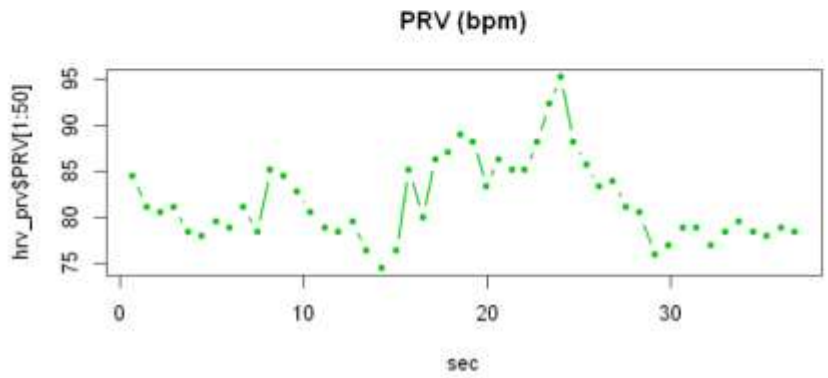
A5.9.8 MongoDB management

3.4 How to use and understand iot data? → ecg_ppg_R.ipynb



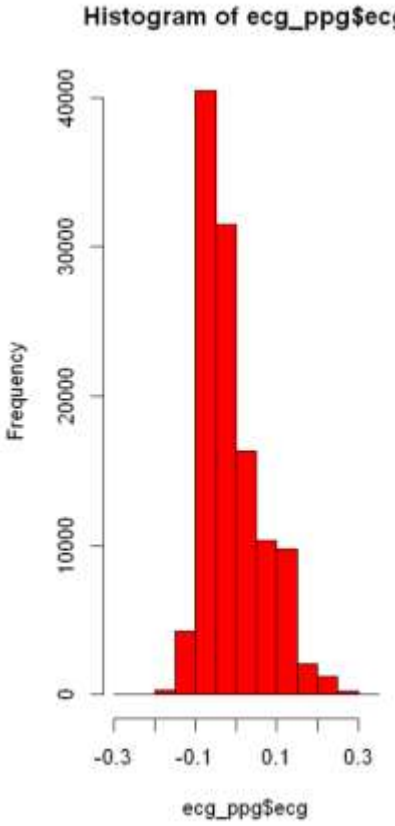


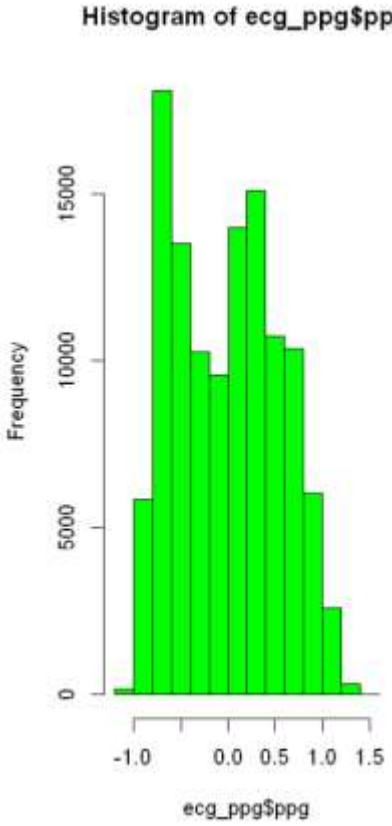


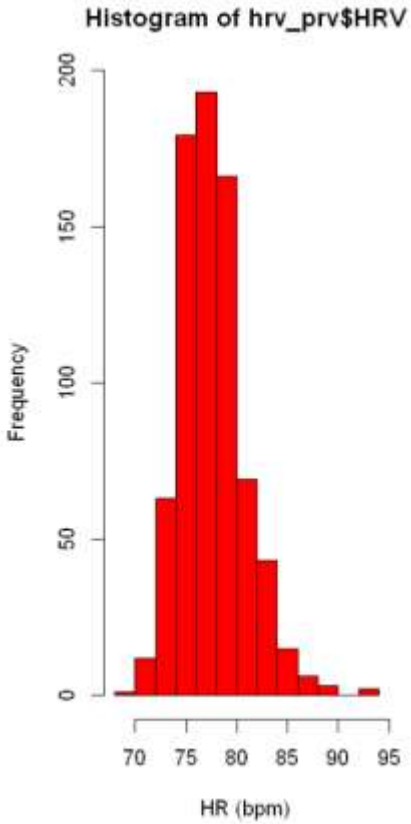


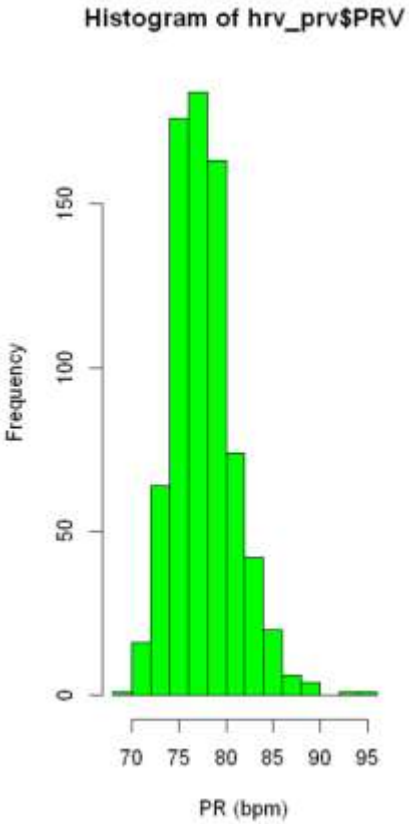
A5.9.8 MongoDB management

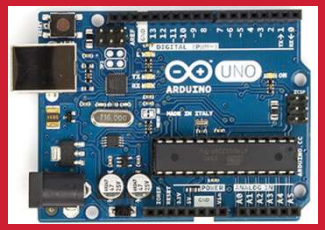
3.4 Which is meaningful? → ecg_ppg_R.ipynb



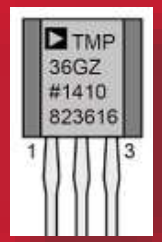


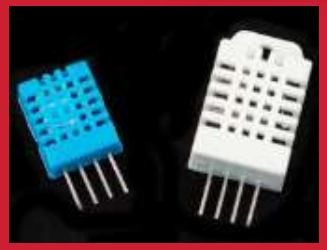






[Practice]





- [wk13]
  - RT Data management with MongoDB
  - Multi-sensor circuits
  - Complete your project
  - Upload file name : HSnn_Rpt11.zip

[wk13] Practice-11 HSnn_Rpt11.zip

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

제출파일명 : **HSnn_Rpt11.zip**

- 압축할 파일들

① HSnn_iot_mongodb.png

② HSnn_iot_mongodb_web.png

③ HSnn_iot_json.png

④ HSnn_iot_DB.png

⑤ HSnn.csv (mongoexport file)

Email : chaos21c@gmail.com

[제목 : **id,** 이름 (수정)]

Lecture materials

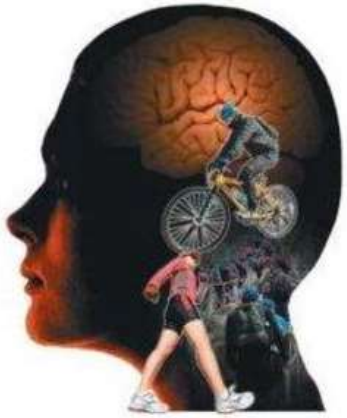
References & good sites

- ✓ http://www.nodejs.org/ko Node.js
- ✓ http://www.arduino.cc Arduino Homepage
- ✓ http://www.w3schools.com By w3schools
- ✓ http://www.github.com GitHub
- ✓ http://www.google.com Googling

Github.com/Redwoods

GitHub, Inc. [US] | https://github.com Redwoods (Redwoods Yi) - GitHub

Features Business Explore Marketplace Pricing Search GitHub Sign in or Sign up



Redwoods Yi

Redwoods

Block or report user

GimHae, Republic of Korea

Overview Repositories 5 Stars 2 Followers 0 Following 0

Pinned repositories

dht22-iot-project

Iot project to monitor data streaming from DHT22 wired at Arduino.

HTML

Lec

All lectures by Redwoods in Inje University

arduino-nodejs-plotly-streaming

This repo introduces a simple and efficient way to plot the streaming data from Arduino with Easy Pulse ppg sensor or DHT11 sensor.

HTML

hw-coding

Resource for lecture of Hardware Programming (2017, Inje university)

Arduino

Github.com/Redwoods/healthcare-signal-iot

Redwoods / Lec

Unwatch 1

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

All lectures by Redwoods in Inje University

Add topics

81 commits | 1 branch | 0 releases

Branch: master | New pull request | Create new file | Upload files

Redwoods 2018 wk01 upload

advanced-Arduino-iot	wk16 exam upload
ev3	wk16 final exam. answers
healthcare-signal-iot	2018 wk01 upload
html5-basic	2018 wk01 upload
html5-mobile-simulation	wk15 lec upload
Lec.Rproj	2018 wk01 upload
README.md	wk03 upload and fix links

Github.com/Redwoods/healthcare-signal-iot

This repository Search Pull requests Issues Marketplace Explore

Redwoods / Lec

Unwatch 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master Lec / healthcare-signal-iot / Create new file Upload

Redwoods 2018 wk03 upload Latest

src	2018 wk03 upload
README.md	2018 wk01 upload
wk01_hs_Intro.pdf	2018 wk01 upload-2
wk01_hs_Intro.pptx	2018 wk01 upload-2
wk02_hs_nodejs.pdf	2018 wk02 upload
wk03_hs_node_express.pdf	2018 wk03 upload

README.md

Lec : Introductionto Healthcare Signal Visualization

All lectures by Redwoods in Inje University from 2018 and 2017.

1.0 What is node.js?

안전함 | https://www.w3schools.com/nodejs/nodejs_intro.asp

HTML CSS JAVASCRIPT SQL PHP BOOTSTRAP HOW TO JQUERY MORE

Node.js Tutorial

- Node.js HOME
- Node.js Intro
- Node.js Get Started
- Node.js Modules
- Node.js HTTP Module
- Node.js File System
- Node.js URL Module
- Node.js NPM
- Node.js Events
- Node.js Upload Files
- Node.js Email

Node.js MySQL

- MySQL Get Started
- MySQL Create Database
- MySQL Create Table

Node.js Introduction

‹ Previous

What is Node.js?

- Node.js is an open source server framework
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

Why Node.js?

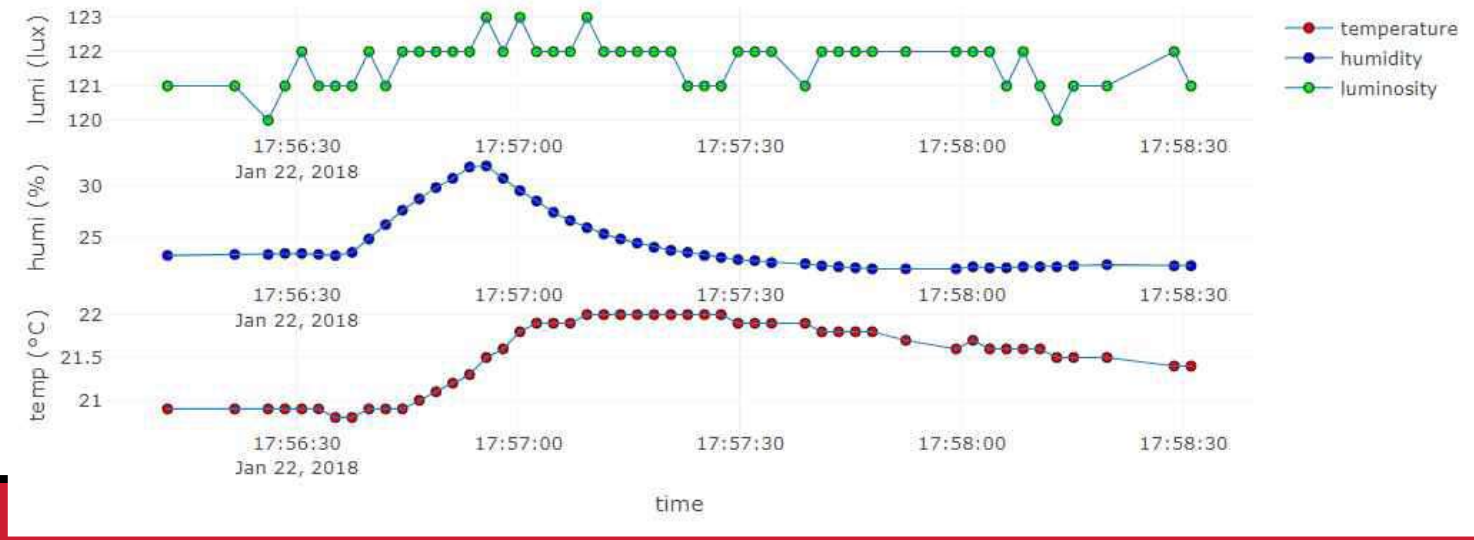
Node.js uses asynchronous programming!

https://www.w3schools.com/nodejs/nodejs_intro.asp

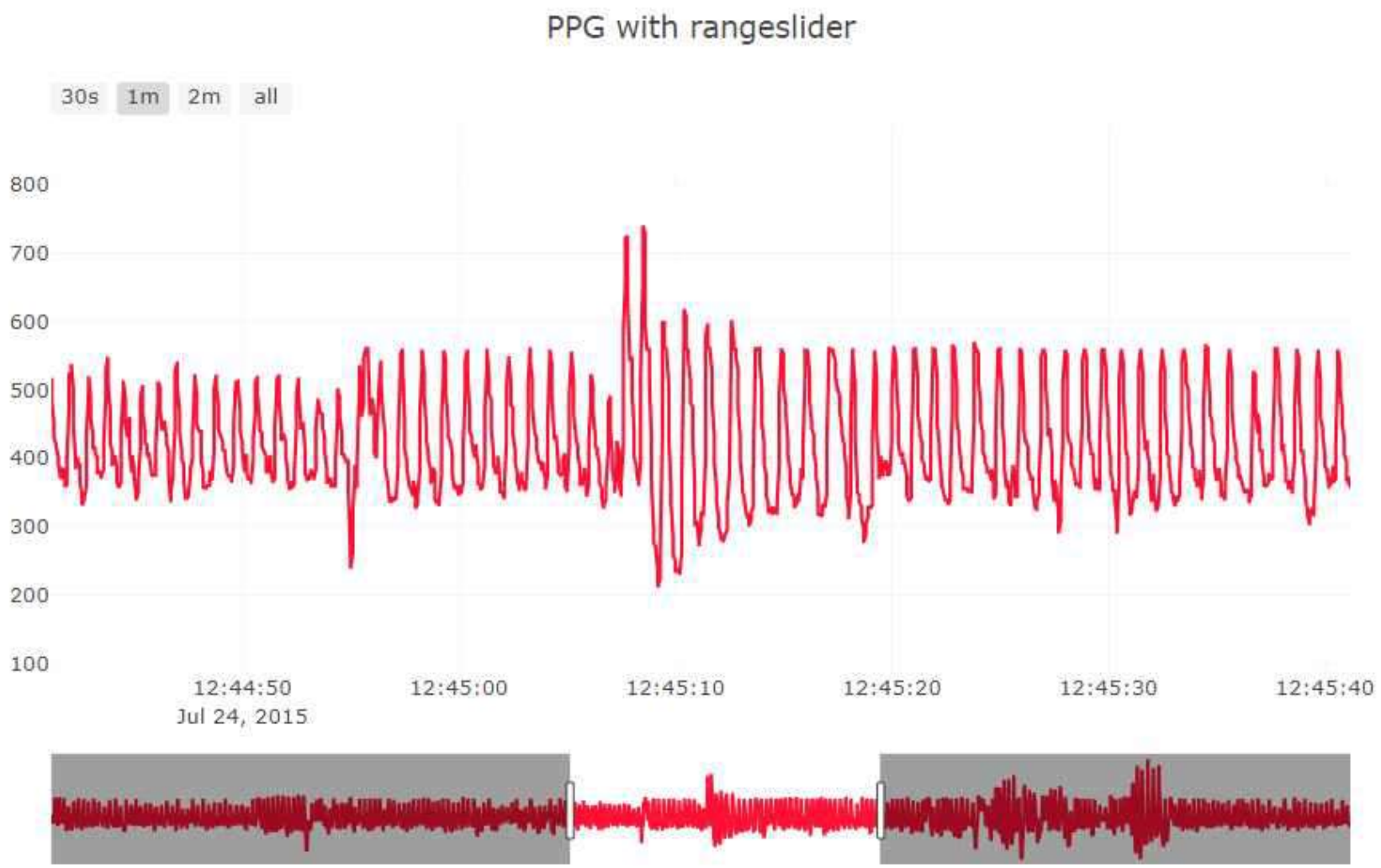
Target of this class

Real-time Weather Station from sensors

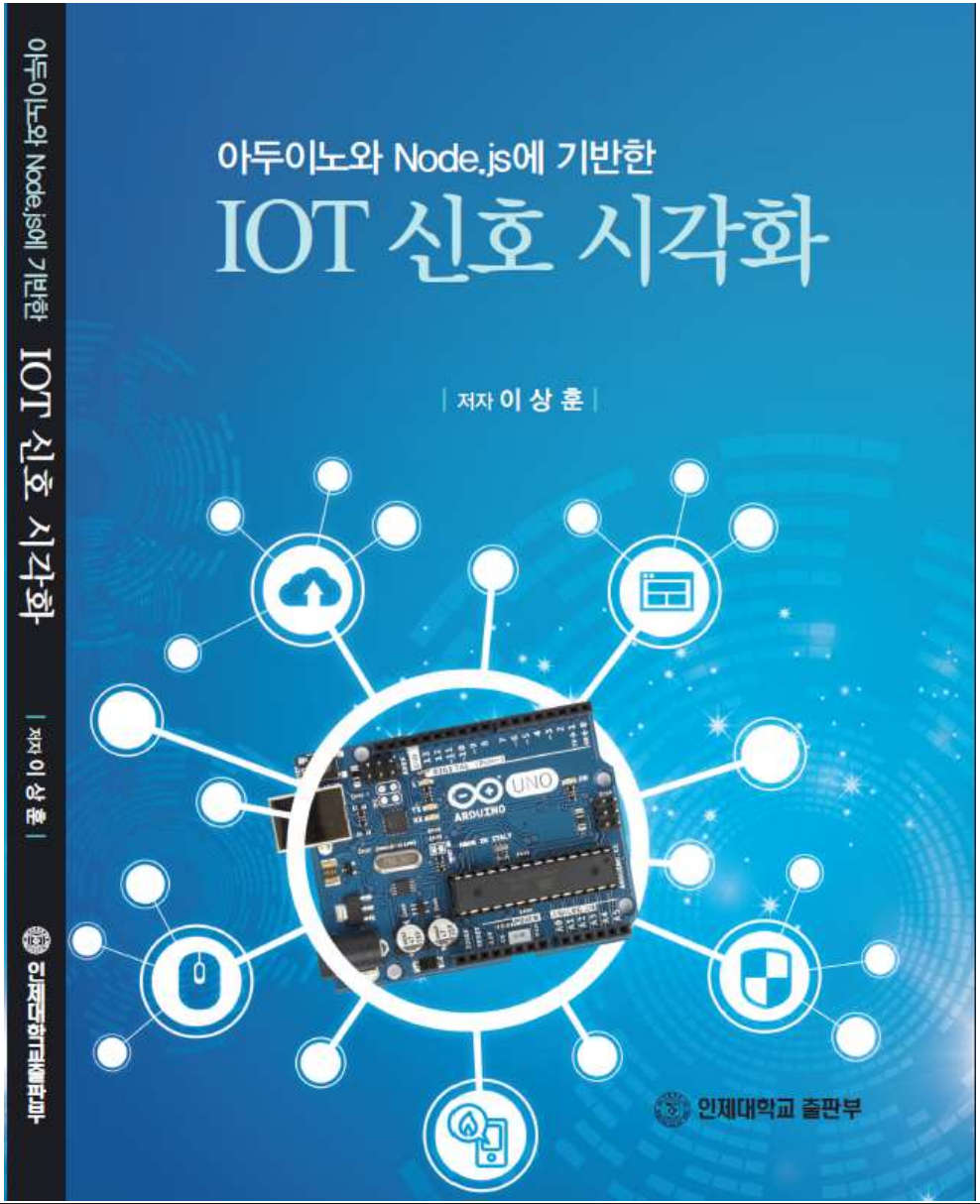
on Time: 2018-01-22 17:58:31.012



Project of this class



주교재


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