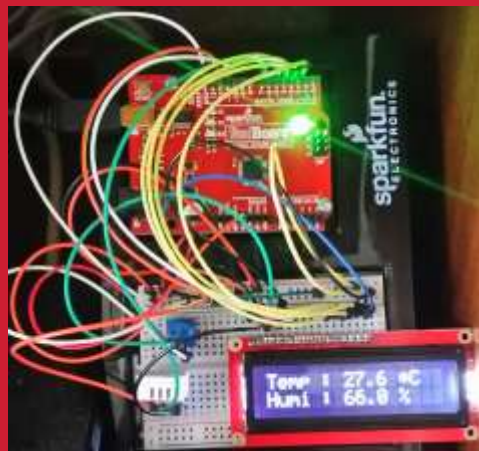




# Arduino-IOT

[wk13]

## Arduino + Node Data visualization III



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



Comsi, INJE University

2<sup>nd</sup> semester, 2018

Email : chaos21c@gmail.com



# My ID

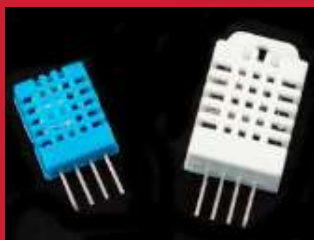
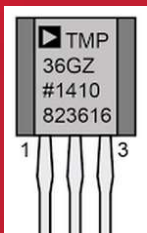
진영빈	AA01
김태은	AA02
도한솔	AA03
박지수	AA04
신성	AA05
박현승	AA06
이석주	AA07
전규은	AA08
정영관	AA09
정의석	AA10

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

## ◆ [Target of this week]

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

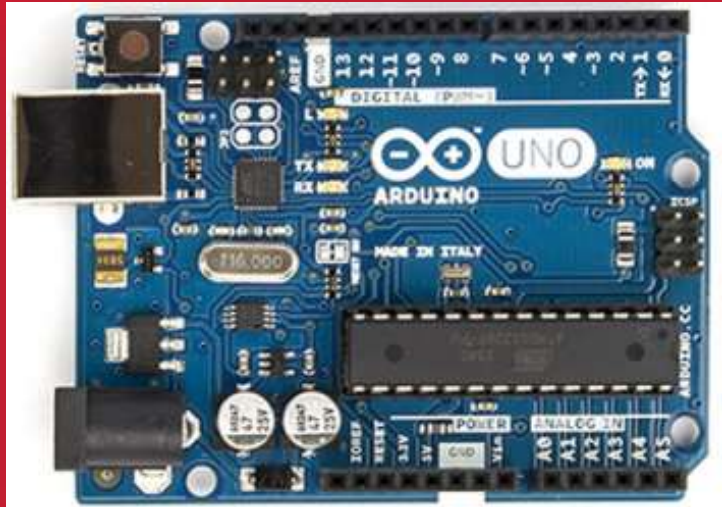
**제출파일명 : AAnn\_Rpt09.zip**

▪ 압축할 파일들

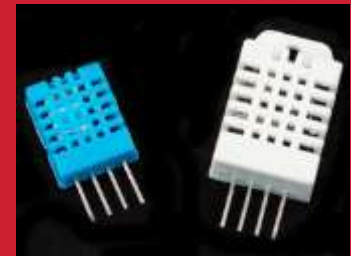
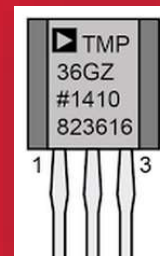
- ① **AAnn\_DS\_cds\_tmp36.png**
- ② **AAnn\_cds\_dht22\_data.png**
- ③ **AAnn\_cds\_dht22.html**
- ④ **AAnn\_cds\_dht22.png**

**Email : [chaos21c@gmail.com](mailto:chaos21c@gmail.com)**

**[ 제목 : id, 이름 (수정) ]**

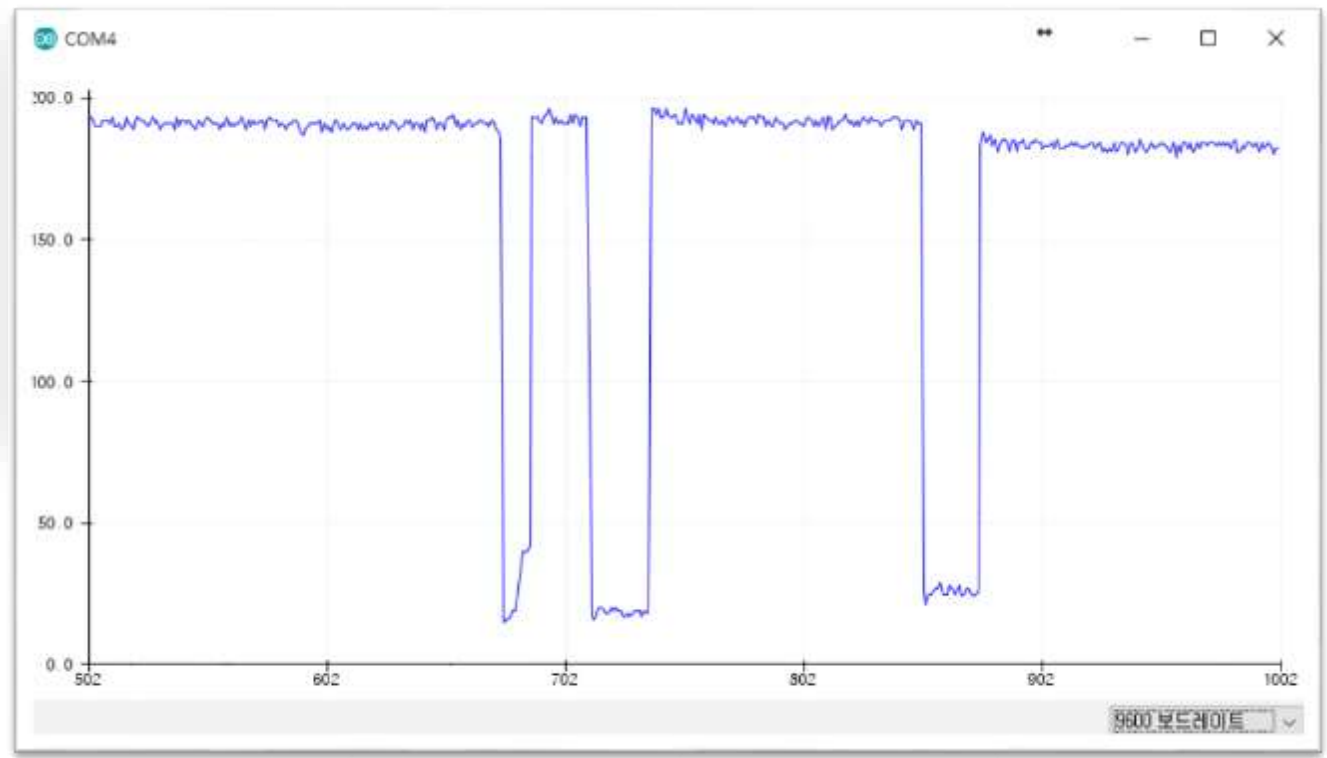
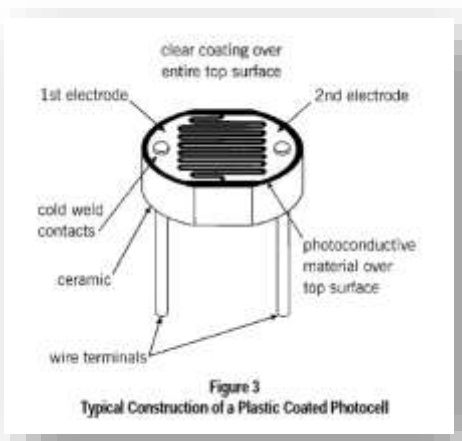


# Arduino & Node.js

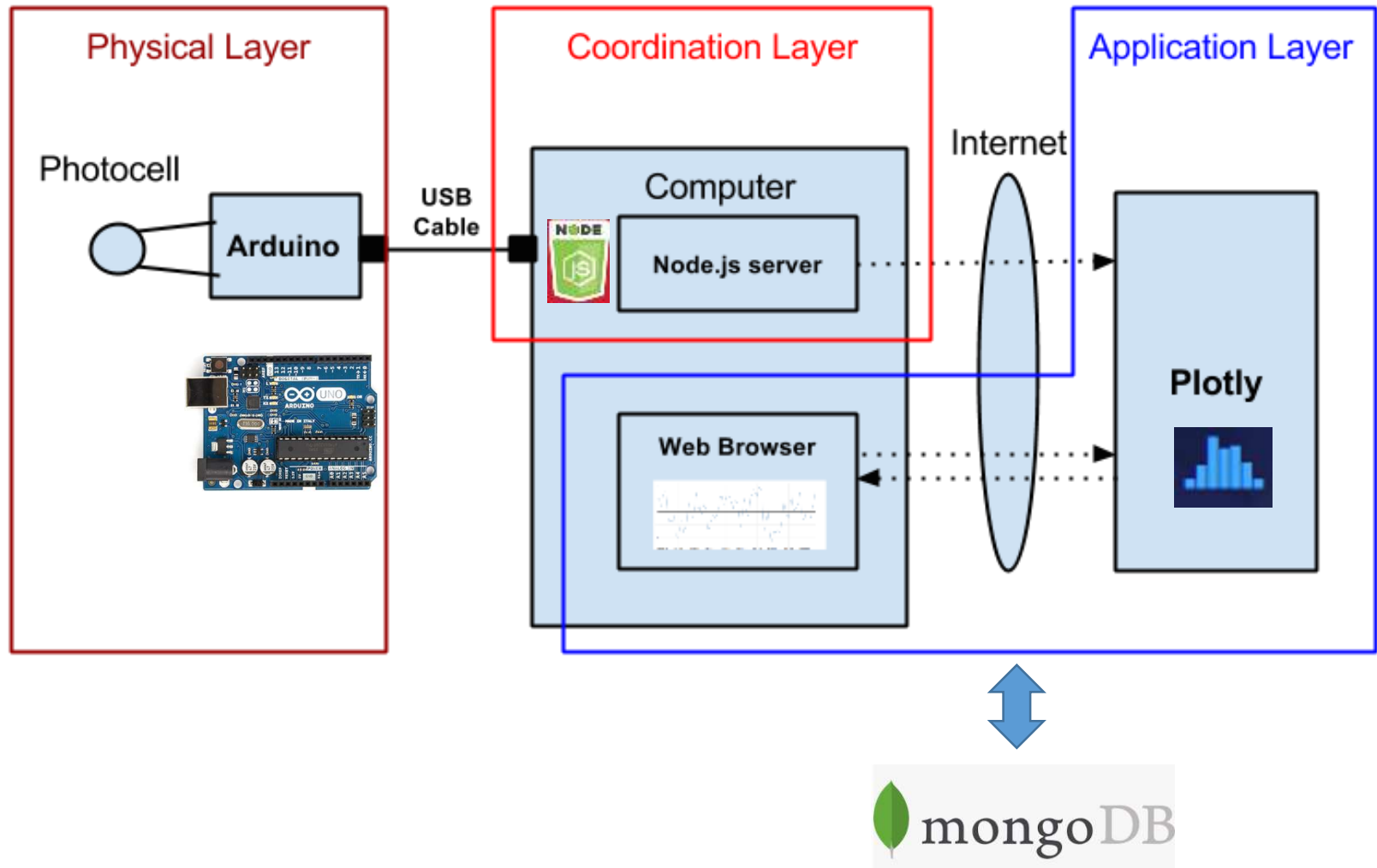




# IOT: HSC

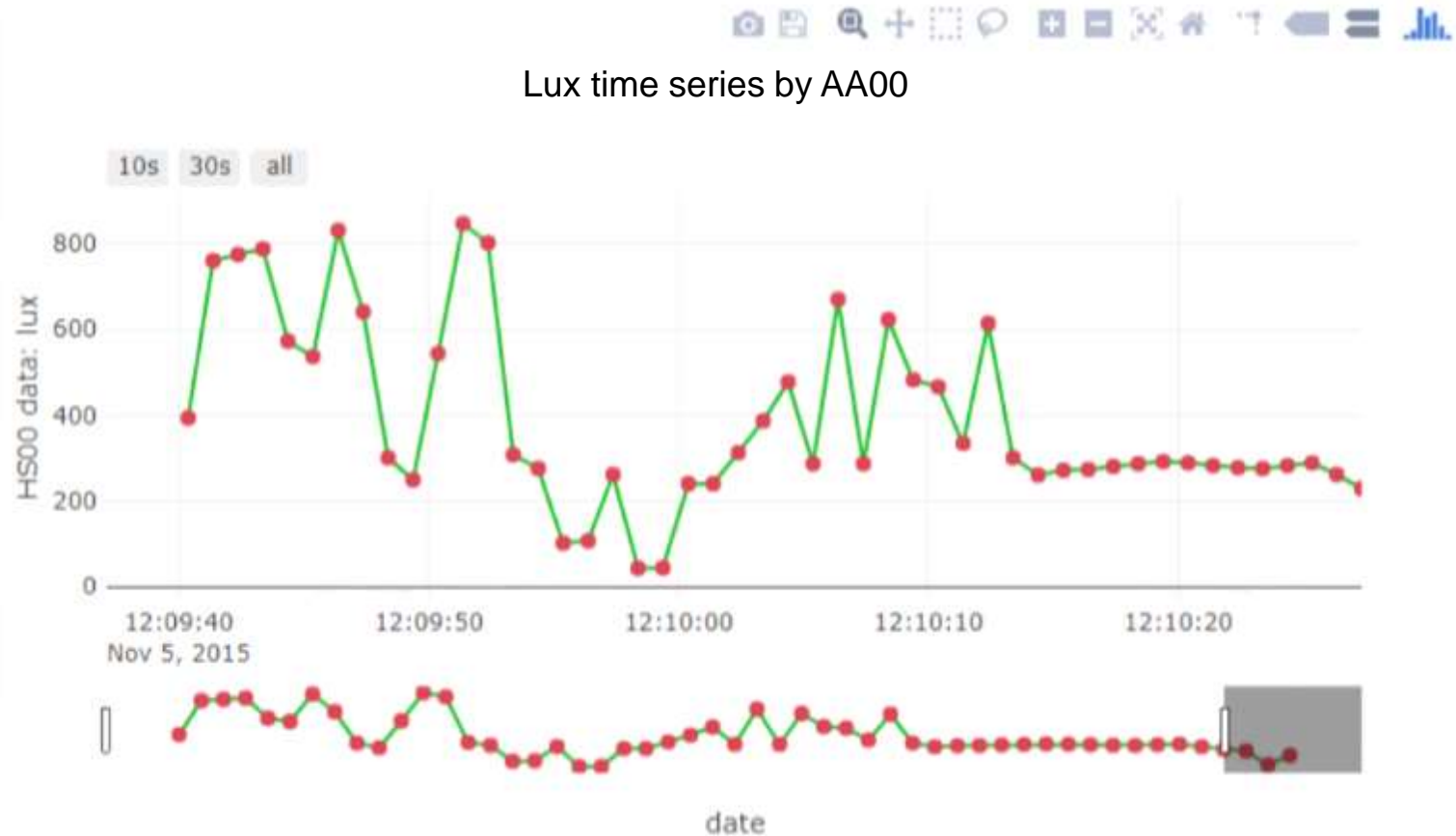


# Layout [H S C]



# Arduino data + plotly

## Time series by AA00

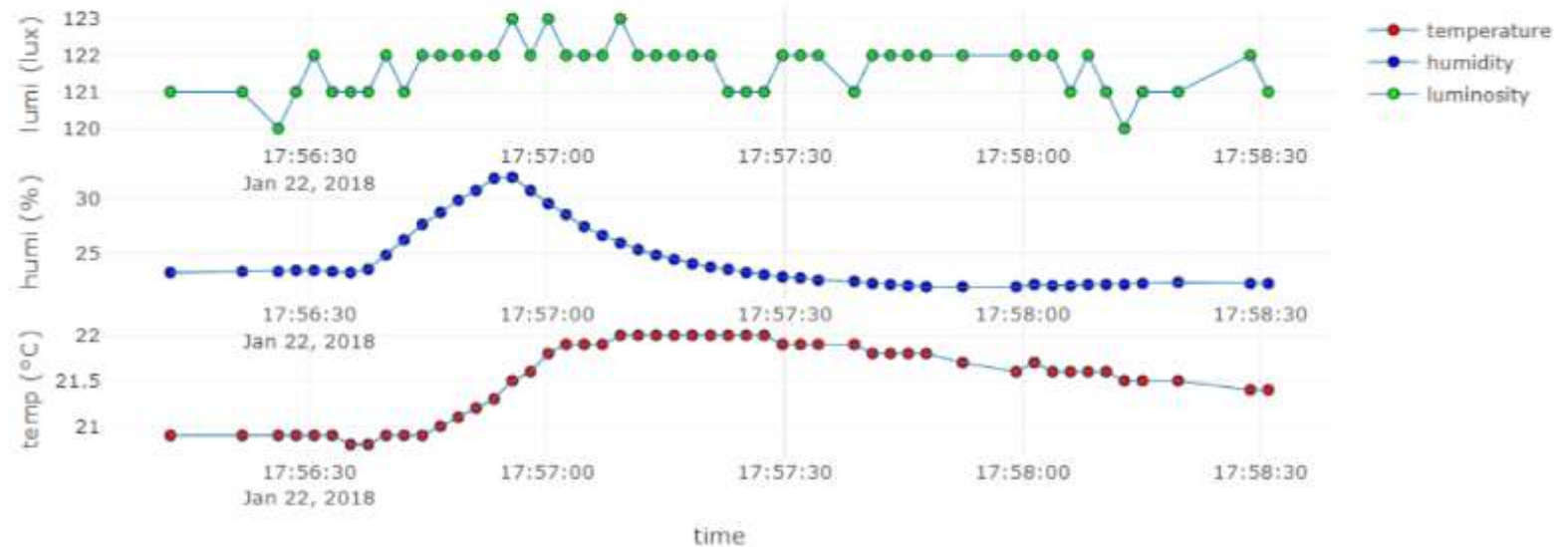


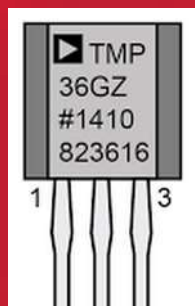
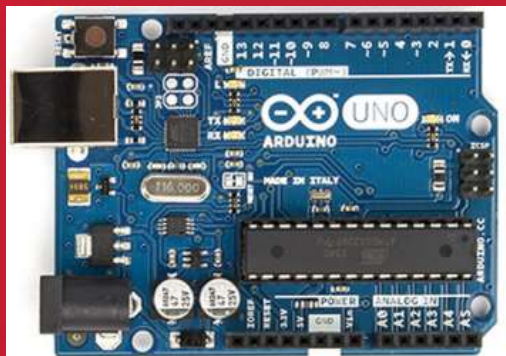


# Real-time Weather Station from sensors

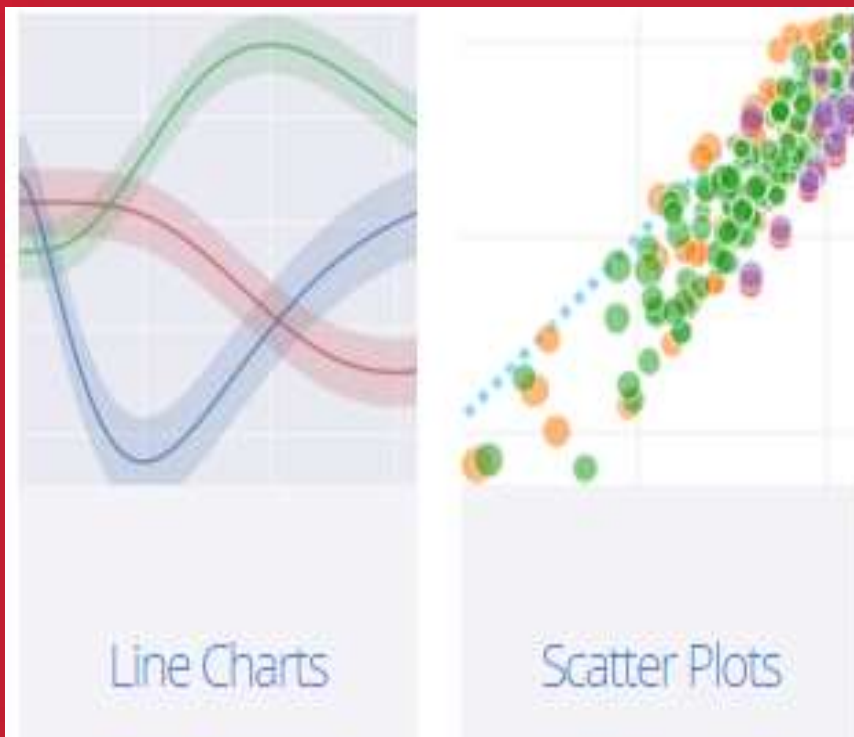


on Time: 2018-01-22 17:58:31.012





# Data visualization using **play.ly**





## A5. Introduction to visualization

**System (Arduino, sDevice, ...)**



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



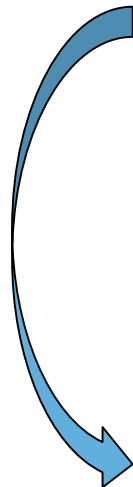
**Visualization & monitoring**

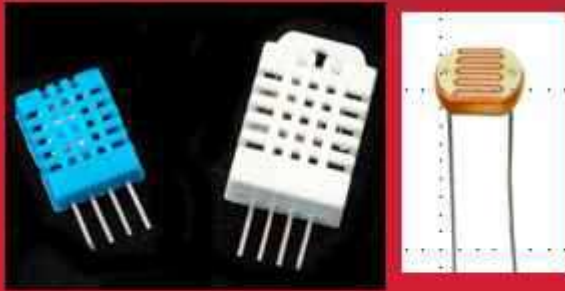


**Data storing & mining**



**Service**





[Goal]

Arduino + Node.js

+ plotly.js

+ MongoDB

→ Data storaging

& visualization



# A5.9 MongoDB




MongoDB for GIANT id x

← → ↻ 🏠 | 안전함 | https://www.mongodb.com

DOCS LEARN WHAT'S MONGODB? LOGIN

📞 🔍 [Free Sandbox](#) [Download](#)

**mongoDB.** | FOR GIANT IDEAS SOLUTIONS CLOUD CUSTOMERS RESOURCES ABOUT US

 **mongoDB®**

**Move at the Speed of Your Data**

Go faster with MongoDB 3.6

[Learn more](#)



# A5.9 MongoDB



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

## NoSQL?

흔히 **NoSQL**이라고 해서 아, **SQL**이 없는 데이터베이스구나! 라고 생각 할 수도 있겠지만, 진짜 의미는 **Not Only SQL** 입니다. 기존의 **RDBMS**의 한계를 극복하기 위해 만들어진 새로운 형태의 데이터저장소 입니다. 관계형 **DB**가 아니므로, **RDMS**처럼 고정된 스키마 및 **JOIN**이 존재하지 않습니다.

## Document?

**Document Oriented** 데이터베이스라는데.. 여기서 말하는 **Document**가 뭘까요? 문서? 이게 그냥 '문서'로 번역해버리면 조금은 애매합니다. 문서라고 하면 보통 워드/엑셀에 사용되는 그런 문서가 떠오르는데요, 그것과는 다릅니다. **Document**는 **RDMS**의 **record**와 비슷한 개념인데요, 이의 데이터 구조는 한개이상의 **key-value pair**으로 이루어져 있습니다. **MongoDB** 샘플 **Document**를 확인해 볼까요?

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



# A5.9 MongoDB



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

**\_id** 는 12bytes의 hexadecimal 값으로서, 각 document의 유일함(uniqueness)을 제공합니다.

이 값의 첫 4bytes 는 현재 timestamp, 다음 3bytes 는 machine id, 다음 2bytes 는 MongoDB 서버의 프로세스id, 마지막 3bytes 는 순차번호입니다 추가될때마다 값이 높아진다는거지요.

**Document** 는 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?

**Collection** 은 MongoDB Document의 그룹입니다. Document들이 **Collection** 내부에 위치하고 있습니다. RDMS의 table과 비슷한 개념입니다만 RDMS와 달리 schema를 따로 가지고 있지않습니다. Document 부분설명에 나와있듯이 각 Document들이 동적인 schema를 가지고 있으니까요

## Database?

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



## MongoDB 3.6

Move at the Speed of your Data

MongoDB 3.6 introduces innovations that make you more productive with less code and operations, whether it's rapidly delivering cutting-edge applications to market, ensuring an exceptional experience on a global scale, or unlocking the intelligence you need for your next move.

[Try it now](#)

[Download the Guide to MongoDB 3.6](#)

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





# A5.9 MongoDB



mongoDB | FOR GIANT IDEAS

SOLUTIONS CLOUD CUSTOMERS RESOURCES ABOUT US

Atlas **Community Server** Enterprise Server Ops Manager Compass Connector for BI

Current Release | Previous Releases | Development Releases

Current Stable Release (3.6.5)  
05/21/2018: Release Notes | Changelog  
Download Source: [tgz](#) | [zip](#)

Windows Linux OSX

Version:  
Windows Server 2008 R2 64-bit and later, with SSL support x64 ▼

Installation Package:  
[DOWNLOAD \(msi\)](#)

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

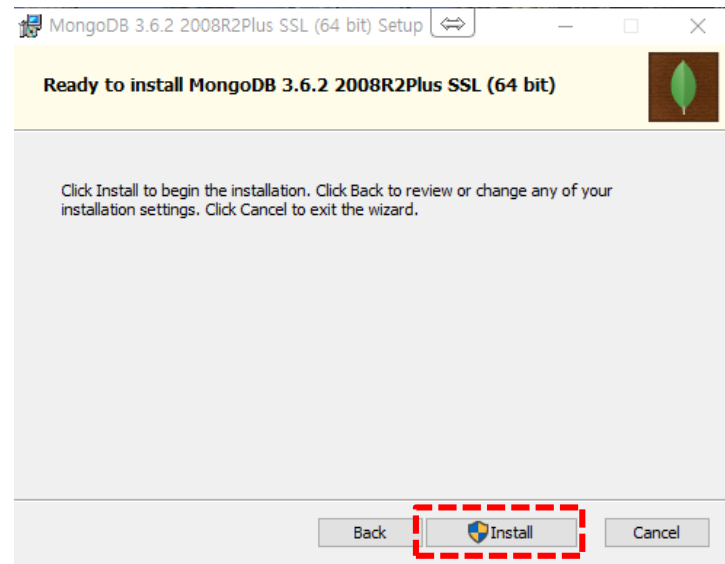
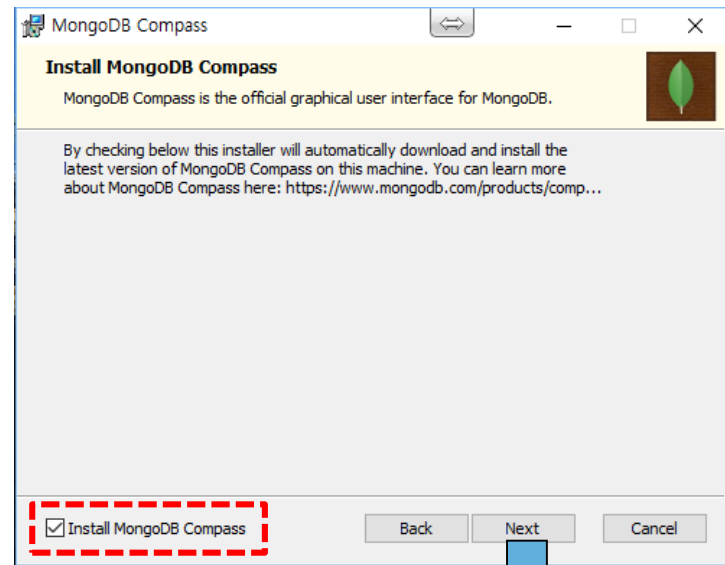
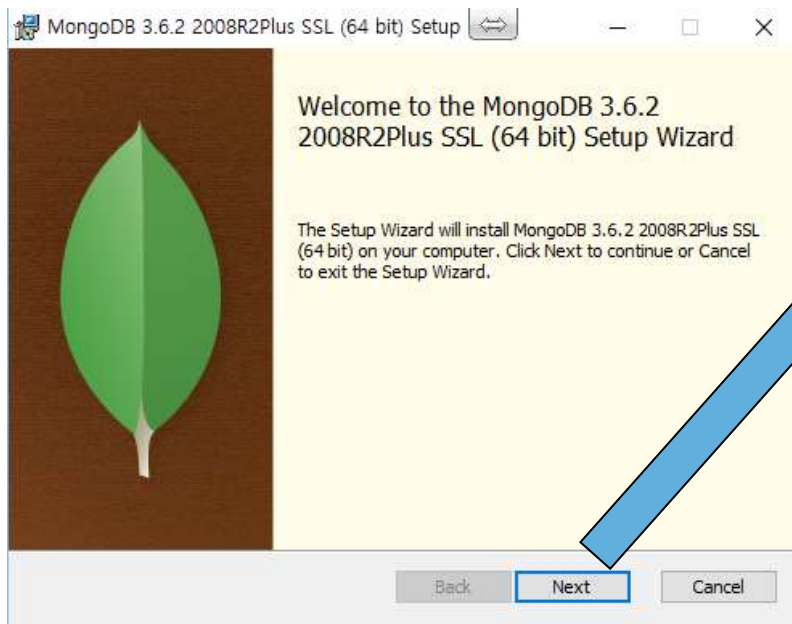
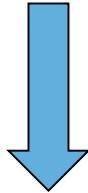


# A5.9.1 MongoDB install – 1

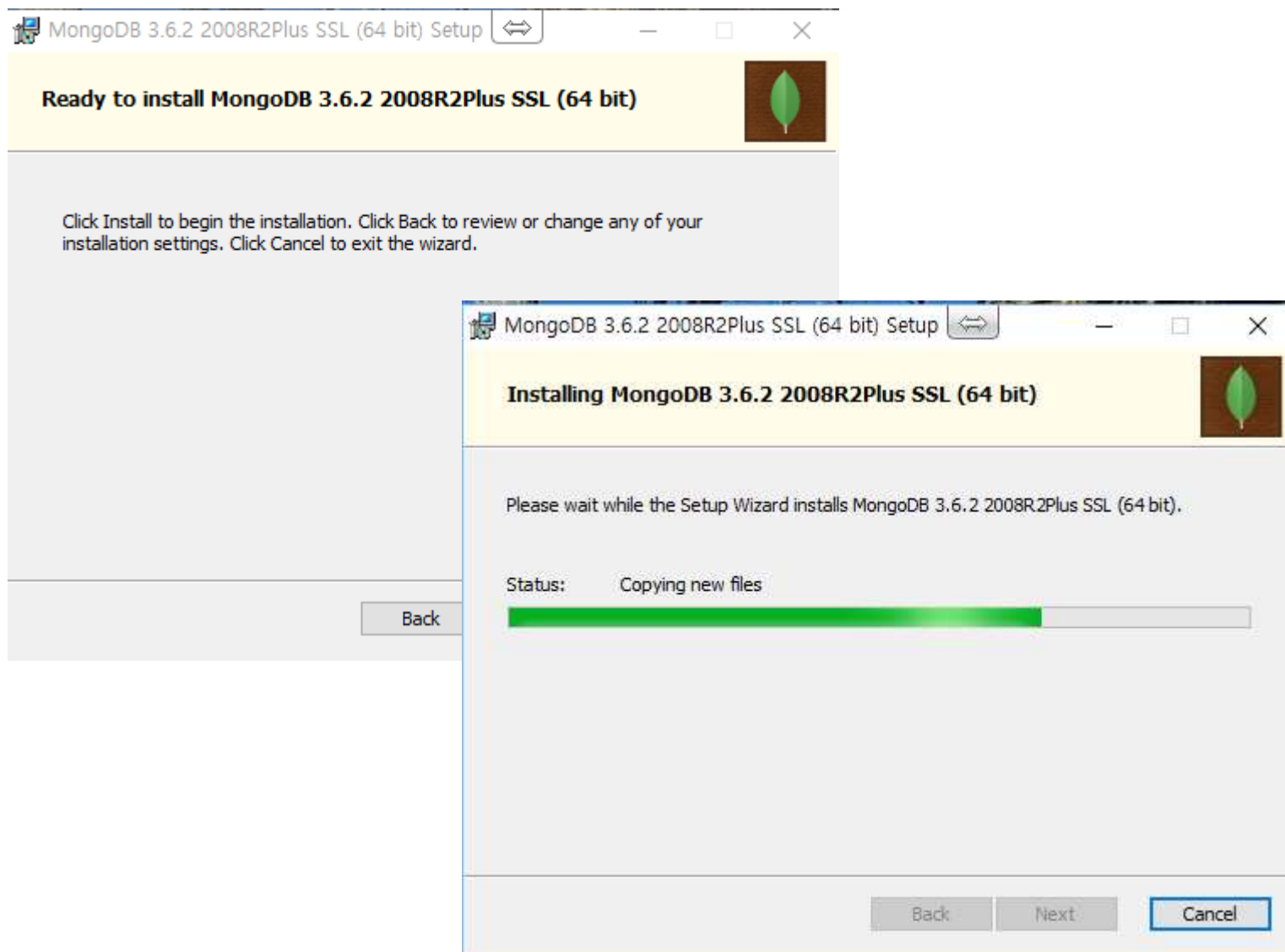
**mongodb-win32-x86\_64-2008plus-ssl-3.6.2-signed**

NodeJSPortable\_5.7.0.paf

Sublime Text Build 3143 x64

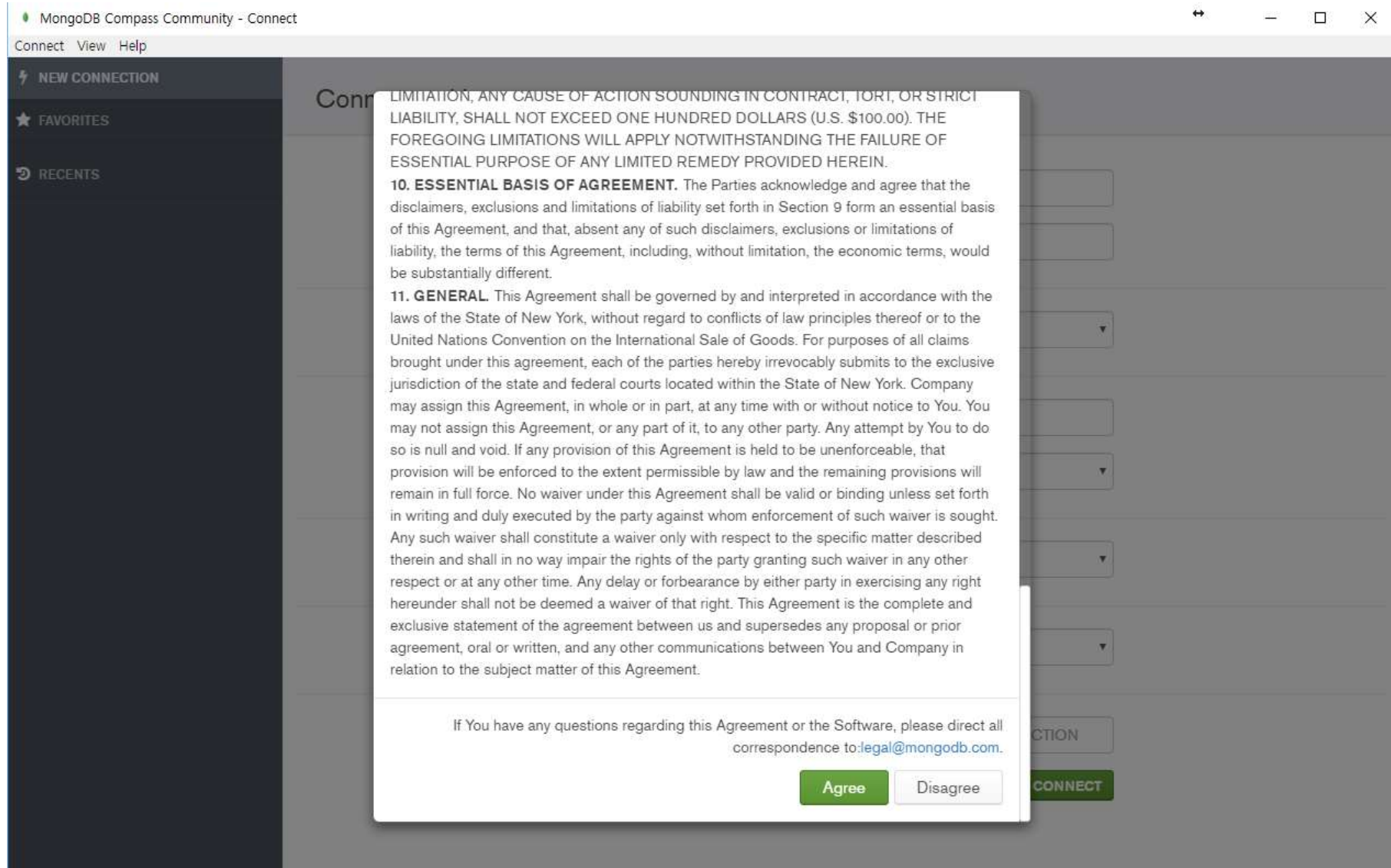


# A5.9.1 MongoDB install – 2





# A5.9.1 MongoDB install – 3





# A5.9.1 MongoDB install – 4

## Privacy Settings

To enhance the user experience, Compass can integrate with 3rd party services, which requires external network requests. Please choose from the settings below:

- ☒ **Enable Crash Reports**  
Allow Compass to send crash reports containing stack traces and unhandled exceptions.
- ☒ **Enable Usage Statistics**  
Allow Compass to send anonymous usage statistics.
- ☒ **Enable Automatic Updates**  
Allow Compass to periodically check for new updates.

With any of these options, none of your personal information or stored data will be submitted.  
Learn more: [MongoDB Privacy Policy](#)

Start Using Compass



## A5.9.1 MongoDB install – 5.



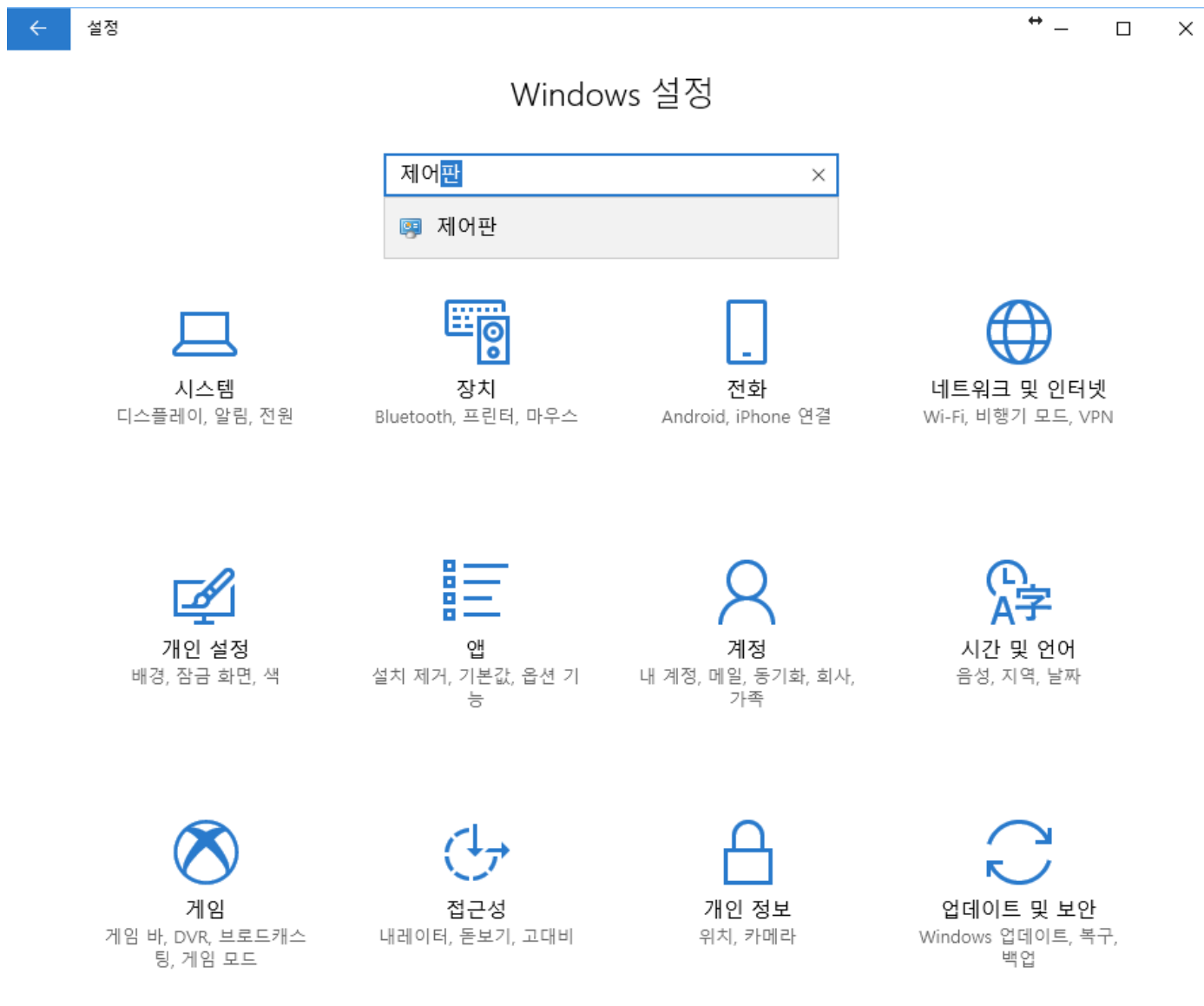
윈도우10: 설정 > 시스템 > 정보

[중요] 시스템 환경변수 : **PATH** 에 경로 추가

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



# A5.9.1 MongoDB install – 6.





# A5.9.1 MongoDB install – 7.

윈도우10: 설정 > ‘제어판’ 검색 > 모든 제어판 항목에서 ‘시스템’ 선택  
> 고급 시스템 설정

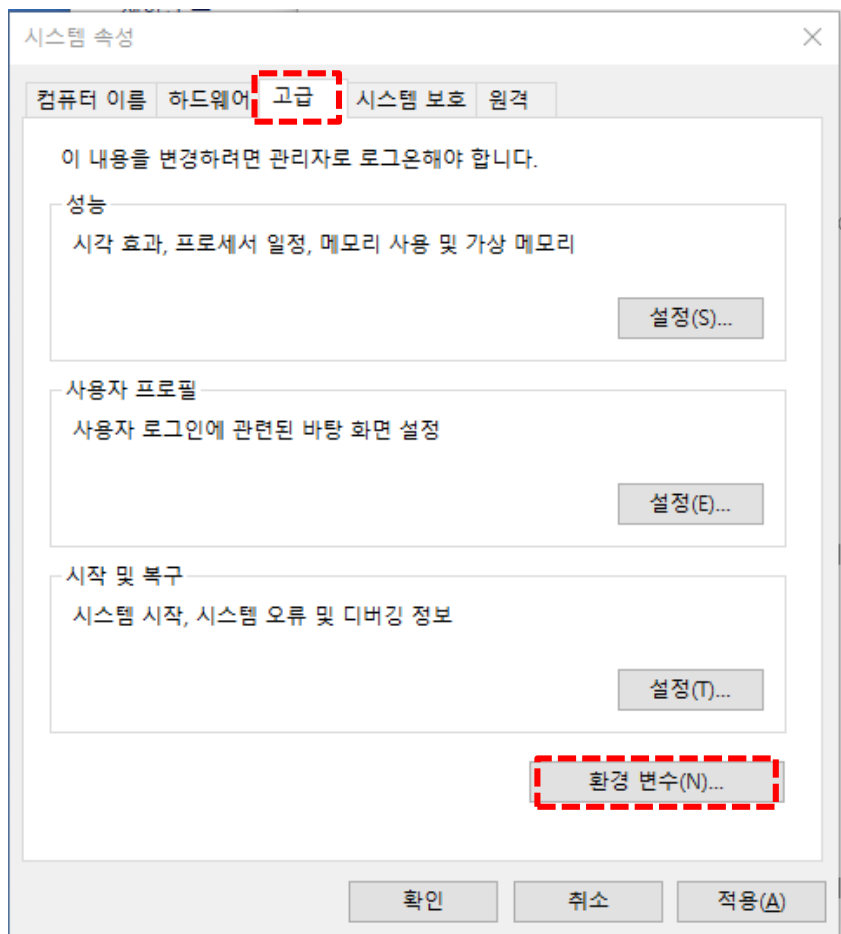
The screenshot shows the Windows 10 Settings application. The left sidebar contains the following options: '시스템' (System), '장치 관리자' (Device Manager), '원격 설정' (Remote Settings), '시스템 보호' (System Protection), and '고급 시스템 설정' (Advanced System Settings), which is highlighted with a red dashed box. The main content area is titled '컴퓨터에 대한 기본 정보 보기' (View basic information about your computer). It displays the following information:

- Windows 버전** (Windows version): Windows 10 Home, © 2017 Microsoft Corporation. All rights reserved.
- 시스템** (System):
  - 프로세서: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz 3.40 GHz
  - 설치된 메모리(RAM): 32.0GB
  - 시스템 종류: 64비트 운영 체제, x64 기반 프로세서
  - 펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.
- 컴퓨터 이름, 도메인 및 작업 그룹 설정** (Computer name, domain, and workgroup settings):
  - 컴퓨터 이름: yish-HCit
  - 전체 컴퓨터 이름: yish-HCit
  - 컴퓨터 설명:
  - 작업 그룹: WORKGROUP
- Windows 정품 인증** (Windows digital license):
  - Windows 정품 인증을 받았습니다. [Microsoft 소프트웨어 사용 조건 읽기](#)
  - 제품 ID: 00325-96080-39821-AAOEM

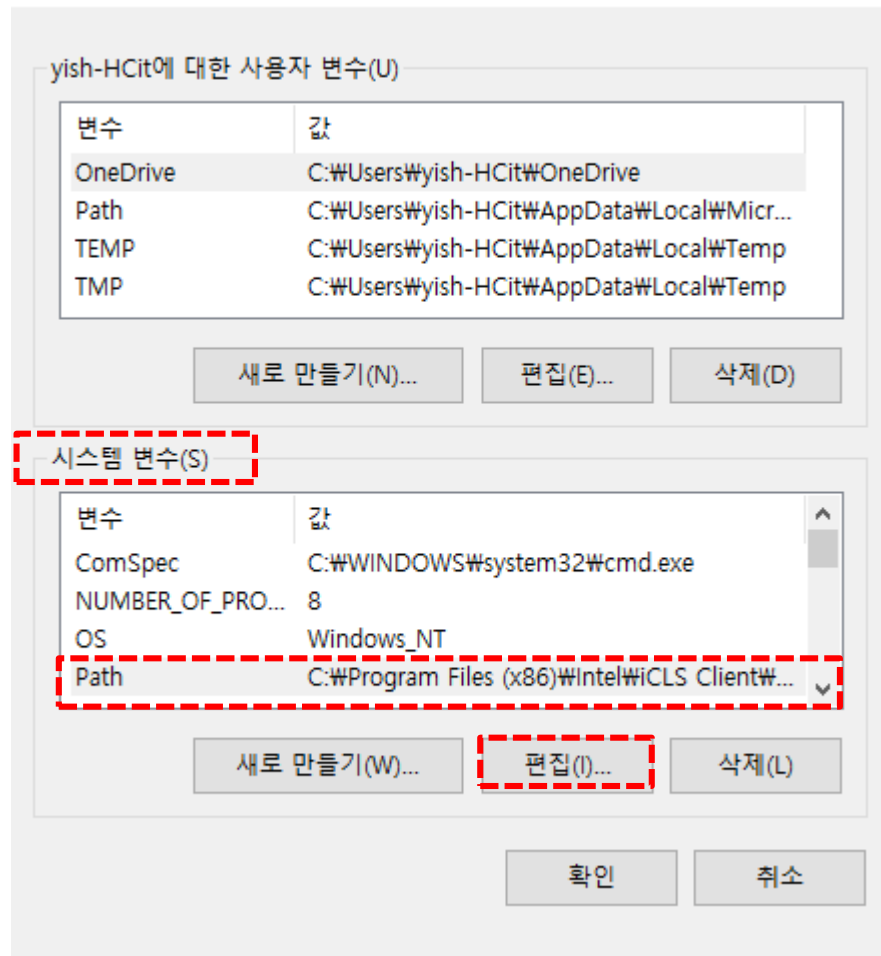
At the bottom left, there are links for '참고 항목' (Related items) and '보안 및 유지 관리' (Security and maintenance).



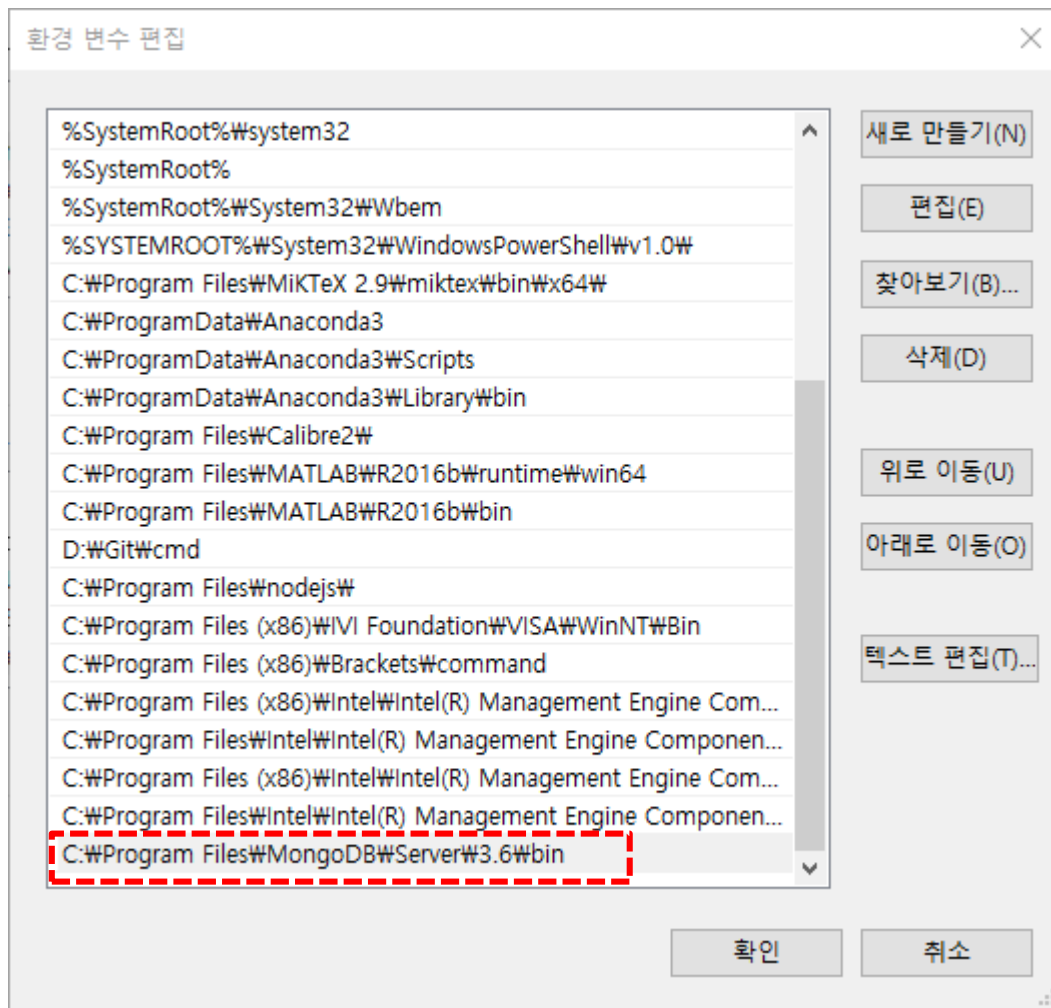
## 환경 변수 설정



환경 변수



## 환경 변수 추가





## A5.9.2 MongoDB shell - 1

### 1. Mongo shell 실행

> mongo

CA 명령 프롬프트

— □ ×

```
Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\yish-HCIt>cd C:\Program Files\MongoDB\Server\3.6\bin

C:\Program Files\MongoDB\Server\3.6\bin>mongo
MongoDB shell version v3.6.2
connecting to: mongodb://127.0.0.1:27017
2018-11-16T15:53:11.687+0900 W NETWORK [thread1] Failed to connect to 127.0.0.1:27017 after 5000ms milliseconds, giving up.
2018-11-16T15:53:11.687+0900 E QUERY [thread1] Error: couldn't connect to server 127.0.0.1:27017, connection attempt failed :
connect@src/mongo/shell/mongo.js:251:13
@(connect):1:6
exception: connect failed

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

**Connect failed... Why?**

## 2. MongoDB 저장소 만들기 → D drive

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

```

관리자: 명령 프롬프트

C:\#>md mongodb

C:\#>cd mongodb

C:\#mongodb>md data

C:\#mongodb>dir
C 드라이브의 볼륨: SYSTEM
볼륨 일련 번호: 3E92-DE79

C:\#mongodb 디렉터리

2018-05-23 오후 12:17 <DIR> .
2018-05-23 오후 12:17 <DIR> ..
2018-05-23 오후 12:17 <DIR> data
0개 파일 0 바이트
3개 디렉터리 97,830,256,640 바이트 남음

C:\#mongodb>
  
```

사용 **PC** 환경에 맞게 실행 (특히, 경로 지정)

실습실 환경에 맞추어서 **D:**에 **mongoDB data** 폴더 지정

## 3. Run MongoDB by using **mongod.exe**

➤ **mongod -dbpath d:\mongodb\data**

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

D:\mongodb>md data

D:\mongodb>mongod -dbpath d:\mongodb\data

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

사용 **PC** 환경에 맞게 실행 (특히, 경로 지정)



## A5.9.2 MongoDB shell - 4

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

➤ **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 enabled for the database.
2018-01-22T19:27:33.550-0700 I CONTROL [initandlisten] **          Read and write access to data 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 localhost.
2018-01-22T19:27:33.557-0700 I CONTROL [initandlisten] **          Remote systems will be unable to connect to this server.
2018-01-22T19:27:33.559-0700 I CONTROL [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2018-01-22T19:27:33.561-0700 I CONTROL [initandlisten] **          addresses it should serve responses 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 this machine is configured to be greater than 40% of the total memory. This can lead to increased
2018-01-22T19:27:33.570-0700 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/working-memory-pressure
2018-01-22T19:27:33.571-0700 I CONTROL [initandlisten]
```



## A5.9.2 MongoDB shell - 5

### 5. mongo shell :

Run new cmd

mongo

show dbs

use local

show  
collections

help

cmd 창 프로그래밍 - mongo

```
> show dbs
admin 0.000GB
local 0.000GB
> use local
switched to db local
> show collections
startup_log
> help
```

```
db.help()
db.mycoll.help()
sh.help()
rs.help()
help admin
help connect
help keys
help misc
help mr
```

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

```
show dbs
show collections
show users
show profile
show logs
show log [name]
```

```
show database names
show collections in current database
show users in current database
show most recent system.profile entries with time >= 1ms
show the accessible logger names
prints out the last segment of log in memory, 'global' is
```

default

```
use <db_name>
db.foo.find()
db.foo.find( { a : 1 } )
it
DBQuery.shellBatchSize = x
exit
```

```
set current database
list objects in collection foo
list objects in foo where a == 1
result of the last line evaluated; use to further iterate
set default number of items to display on shell
quit the mongo shell
```



## A5.9.3 MongoDB shell coding

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

use as00

show collections

insert record with new collection "user"

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

show collections

→ "user"

show dbs

**db.user.find()**

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





## A5.9.3 MongoDB shell coding

### 2. insert more records with different schema & show records

insert record2

insert record3

show collections

db.user.find()

db.user.find().pretty()

명령 프롬프트 - mongo

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

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



## A5.9.3 MongoDB shell coding

### 3. insert more records with different schema & show records

insert record4  
with firstName key

`db.user.find()`

`db.user.find().pretty()`

```
명령 프롬프트 - mongo
> 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

### 4. remove one of records (or documents)

remove record3

`db.user.find().pretty()`

명령 프롬프트 - mongo

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

## 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

[AAnn\\_mongo\\_update.png](#)





# Node.js



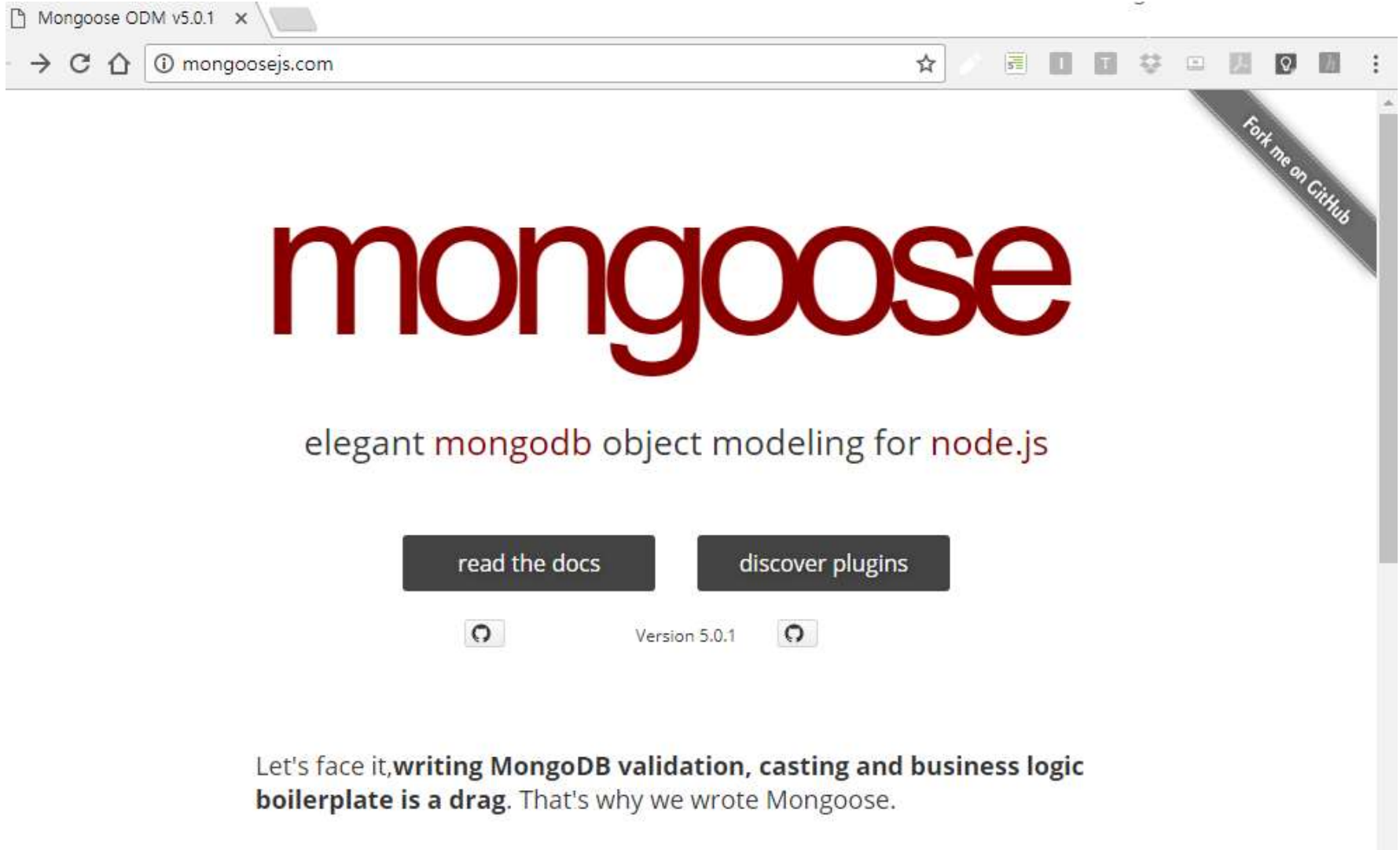
+

# MongoDB






## A5.9.4 MongoDB + Node.js : mongoose




<http://mongoosejs.com/>






# A5.9.4 MongoDB + Node.js : mongoose

 [Features](#) [Business](#) [Explore](#) [Marketplace](#) [Pricing](#) [This repository](#)


Automatic / mongoose 




[Code](#) [Issues 250](#) [Pull requests 2](#) [Projects 0](#) [Wiki](#) [Insights](#)

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

 8,362 commits  14 branches  420 releases

Branch: master ▾ [New pull request](#)

 vkarpov15 Merge branch '4.x'

 .github	fix typo
 benchmarks	style: remove unused `BlogPost` variable
 docs	Merge branch '4.x'

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



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

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

```
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

### 2. node.js project using mongoose (use Sublime Text 3)

- cds\_dht22 project in SBT3
- New file: dbtest.js
- ^B (run dbtest.js)

The screenshot shows the Sublime Text 3 editor with a file named `dbtest.js` open. The file contains the following code:

```
1 // dbtest.js
2 var mongoose = require('mongoose');
3 mongoose.connect('mongodb://localhost:27024/test');
4
5 var SensorSchema = new mongoose.Schema({
6   data: String,
7   created: Date
8 });
9
10 // data model
11 var Sensor = mongoose.model("Sensor", SensorSchema);
12
13 var sensor1 = new Sensor({data: '124', created: new Date()});
14 sensor1.save();
15
16 var sensor2 = new Sensor({data: '573', created: new Date()});
17 sensor2.save();
18
19 console.log("Sensor data were saved in MongoDB");
20
```

The output of the program is displayed in the console at the bottom of the editor:

```
Sensor data were saved in MongoDB
```

Sensor data were saved in MongoDB



## A5.9.4 MongoDB + Node.js : mongoose

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

#### Mongo shell

> show dbs

> use test

> show collections

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

```
명령 프롬프트 - mongo
> show dbs
aa00      0.000GB
admin     0.000GB
config    0.000GB
local     0.000GB
test      0.000GB
> use test
switched to db test
> show collections
sensors
> db.sensors.find().pretty()
...
2018-01-23T14:31:32.959+0900 E QUERY   [thread1] SyntaxError: identifier starts immediately af
ter numeric literal @(<shell>):1:16
> use test
switched to db test
> show collections
sensors
> db.sensors.find().pretty()
{
  "_id" : ObjectId("5a66c84d000e8f1630e176e9"),
  "data" : "124",
  "created" : ISODate("2018-01-23T05:29:49.973Z"),
  "__v" : 0
}
{
  "_id" : ObjectId("5a66c84d000e8f1630e176ea"),
  "data" : "573",
  "created" : ISODate("2018-01-23T05:29:49.977Z"),
  "__v" : 0
}
```



# A5.9.4 MongoDB + Node.js : mongoose

## 4. dbtest2.js (use Sublime Text 3)

D:\Portable\Node\SPortable\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.com\LauncherRuntimeData-Node\SP

```
1 // dbtest2.js
2 var mongoose = require('mongoose');
3 mongoose.connect('mongodb://localhost/test2');
4
5 var SensorSchema = new mongoose.Schema({
6   data: String,
7   created: String
8 });
9
10 // data model
11 var Sensor = mongoose.model("Sensor", SensorSchema);
12
13 var sensor1 = new Sensor({data: '124', created: getDateString()});
14 sensor1.save();
15
16 var sensor2 = new Sensor({data: '573', created: getDateString()});
17 sensor2.save();
18
19 console.log("[dbtest2.js]: Sensor data were saved in MongoDB");
20
21 // helper function to get a nicely formatted date string
22 function getDateString() {
23   var time = new Date().getTime();
24   // 32400000 is (GMT+9 Korea, GimHae)
25   // for your timezone just multiply +/-GMT by 3600000
26   var datestr = new Date(time + 32400000).
27     toISOString().replace(/T/, ' ').replace(/Z/, '');
28   return datestr;
29 }
```

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

[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

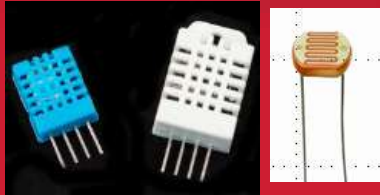
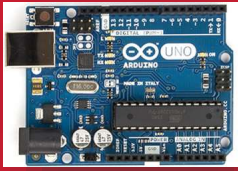
> show dbs

> use test2

> show collections

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

```
cmd. 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
>
```

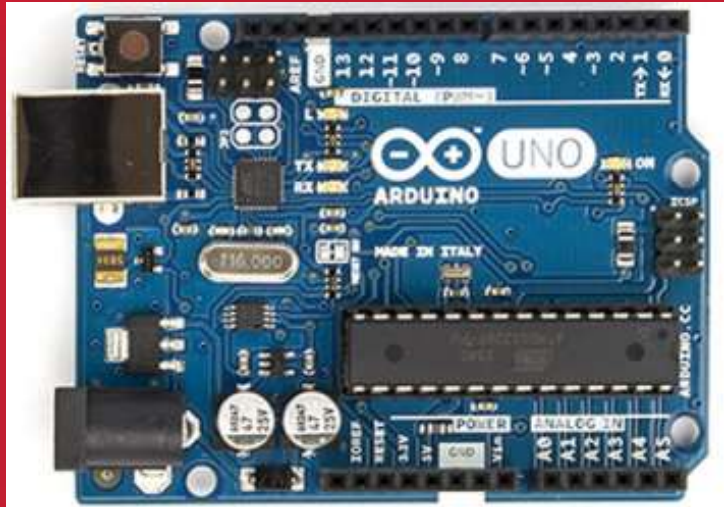
mongo db connection OK.

```
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
```

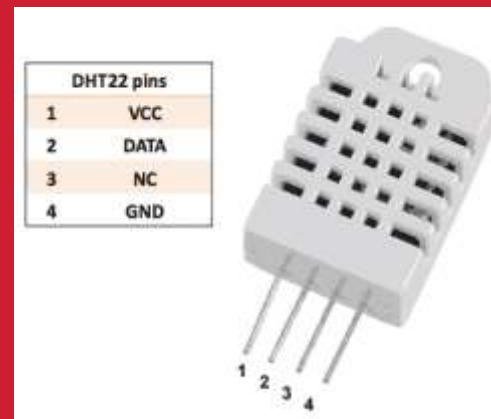




# Arduino & Node.js & MongoDB

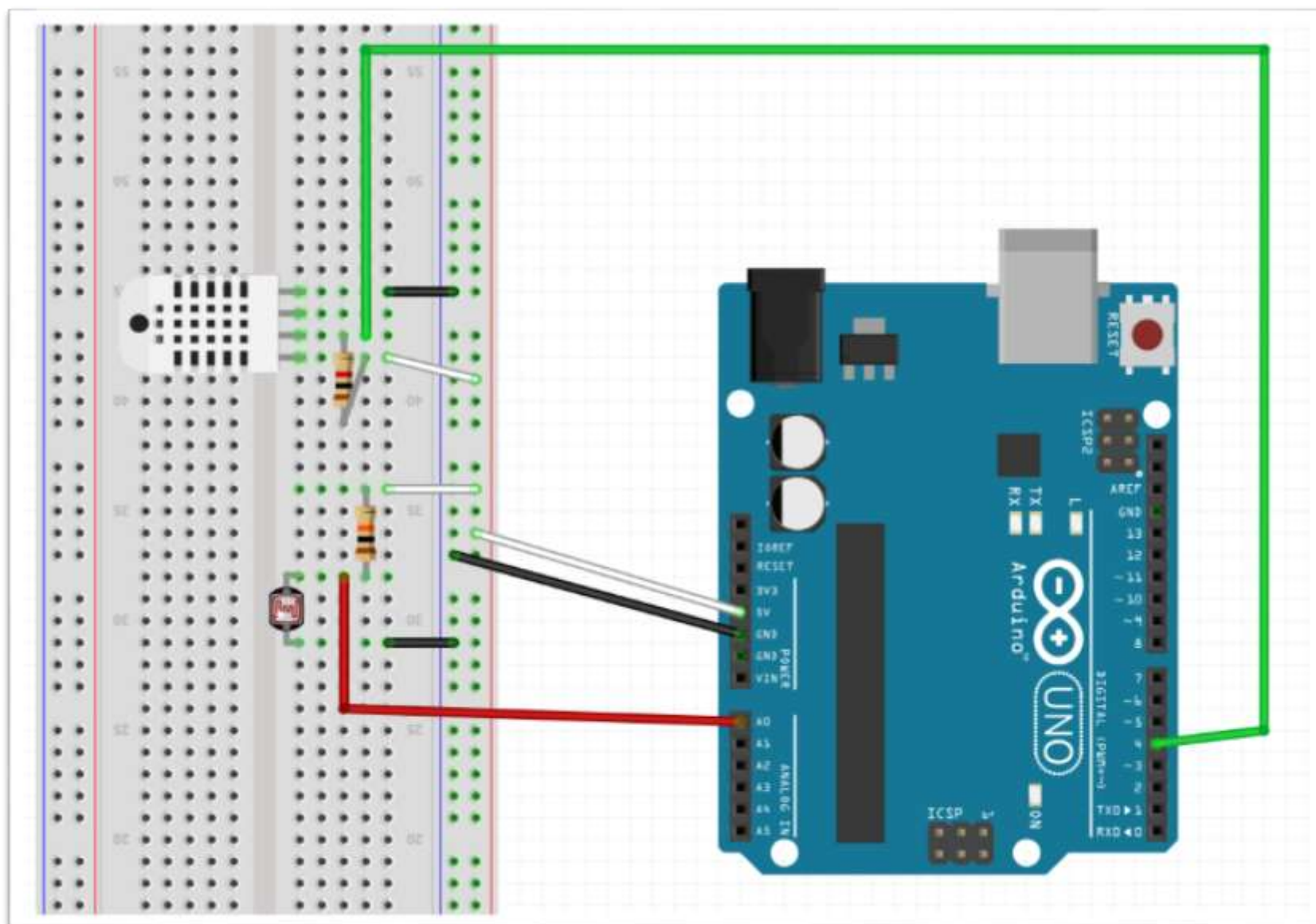


**Multi-sensors**  
**DHT22 + CdS**





# DHT22 + CdS : circuit

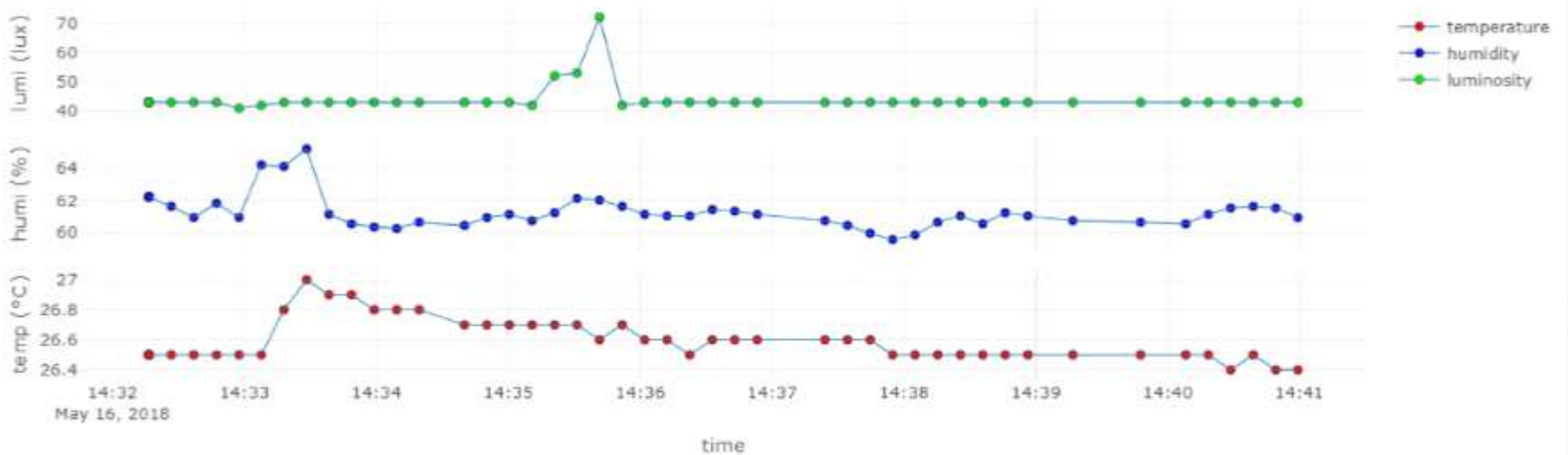




## Real-time Weather Station from sensors



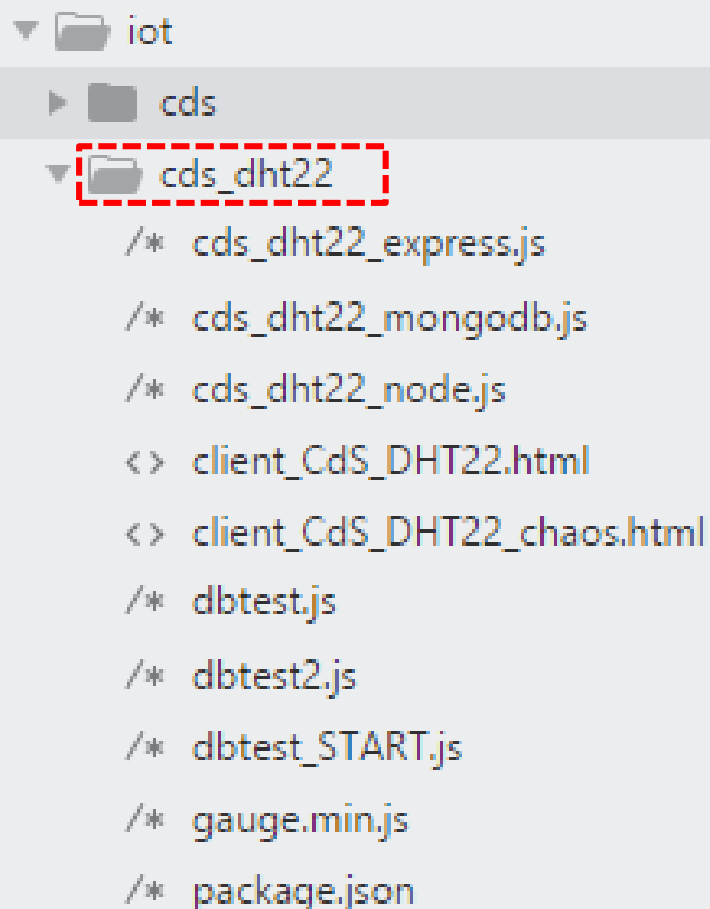
on Time: 2018-05-16 14:40:59.402





## 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

```
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 handleError(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 });
```



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

### 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
```





# 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

AAnn\_iot\_mongodb.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 address
28 app.get('/', function (req, res) { // localhost:3030/
29   res.send('Hello Arduino IOT: express server by AA00!');
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

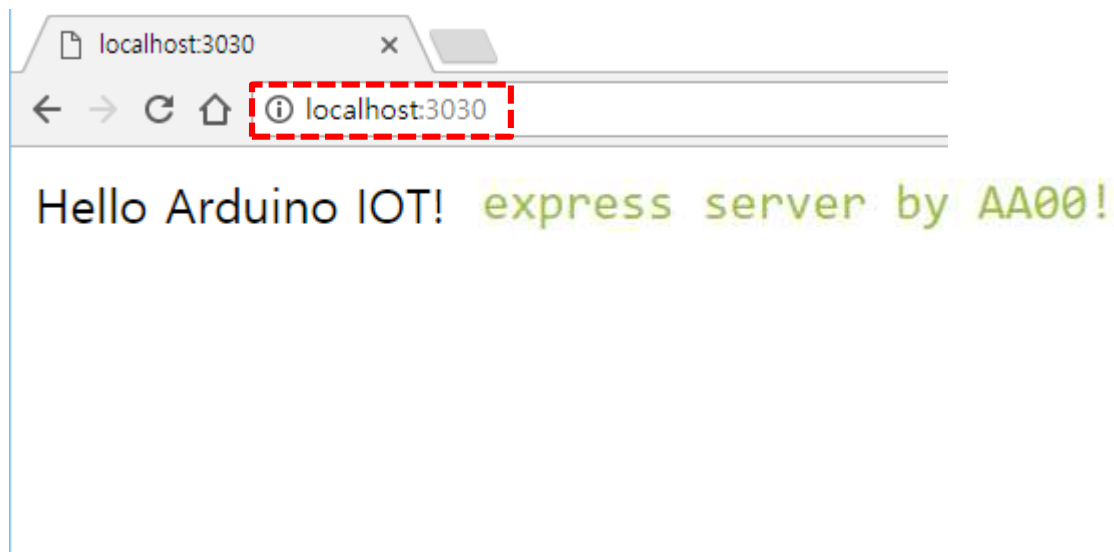
### 2.3 cds\_dht22\_express.js → Run

```
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/>





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

### 2.5 cds\_dht22\_express.js → routing2 <http://localhost:3030/iot>

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

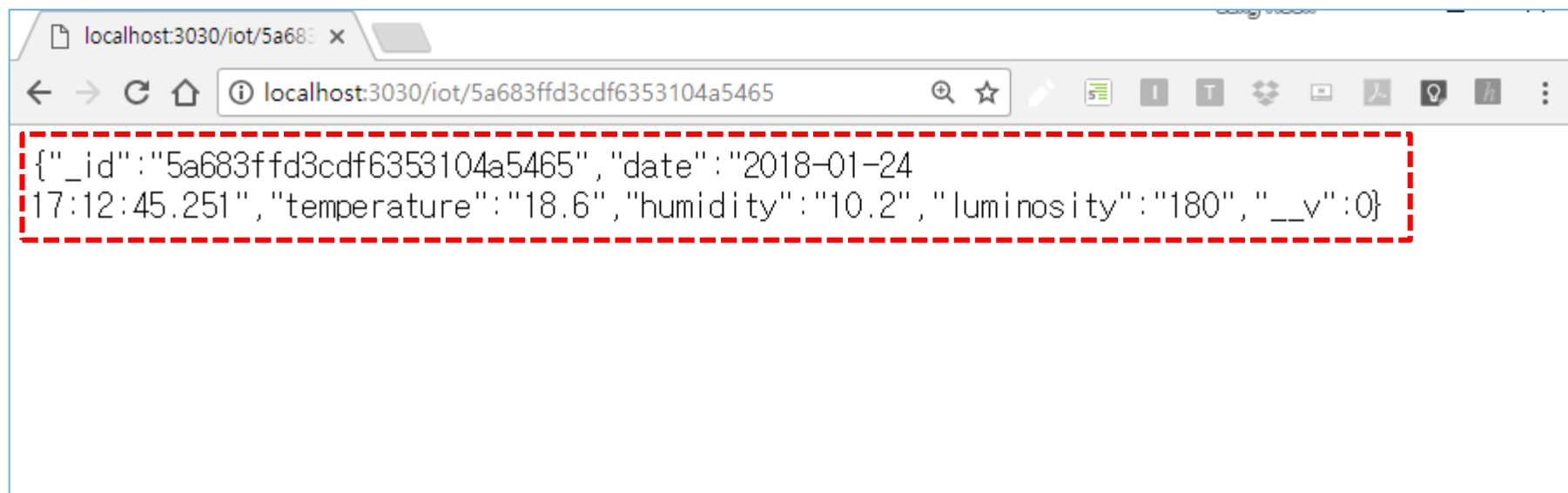
Save as

AAnn\_iot\_mongodb\_web.png



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

2.6 cds\_dht22\_express.js → routing2 <http://localhost:3030/iot:id>



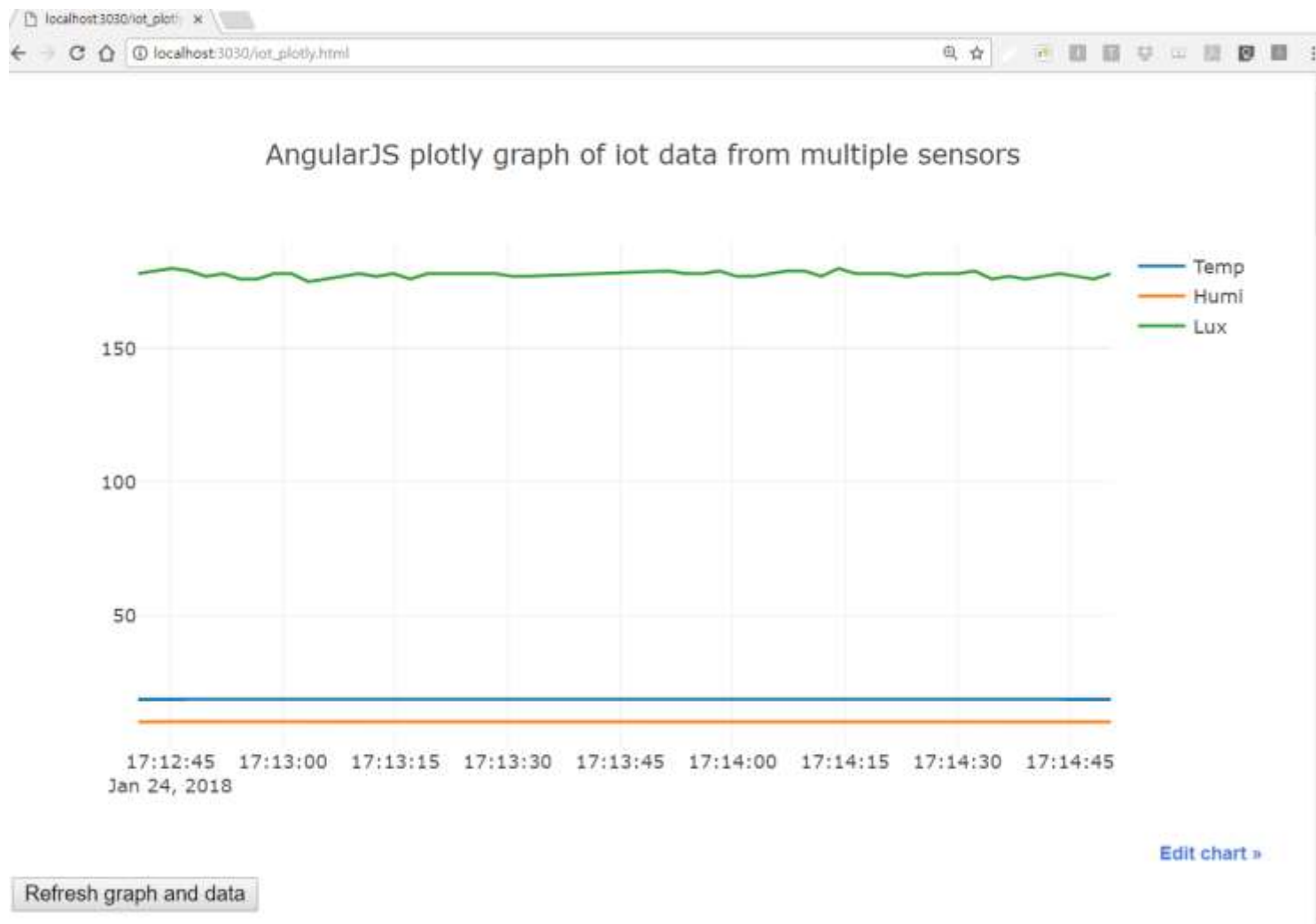
```
{ "_id": "5a683ffd3cdf6353104a5465", "date": "2018-01-24  
17:12:45.251", "temperature": "18.6", "humidity": "10.2", "luminosity": "180", "__v": 0 }
```





# DHT22 + CdS + Node.js + MongoDB

## [Next week] Web monitoring





# DHT22 + CdS + Node.js + MongoDB

[Next week] Web monitoring

chaos.inje.ac.kr:3030/iot3 x

← → ↻ 🏠 ⓘ chaos.inje.ac.kr:3030/iot3.html 🔍 ☆

## MongoDB datab

---

## Time series : Multi ser

chaos.inje.ac.kr:3030 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", "__v":0}]
```

확인

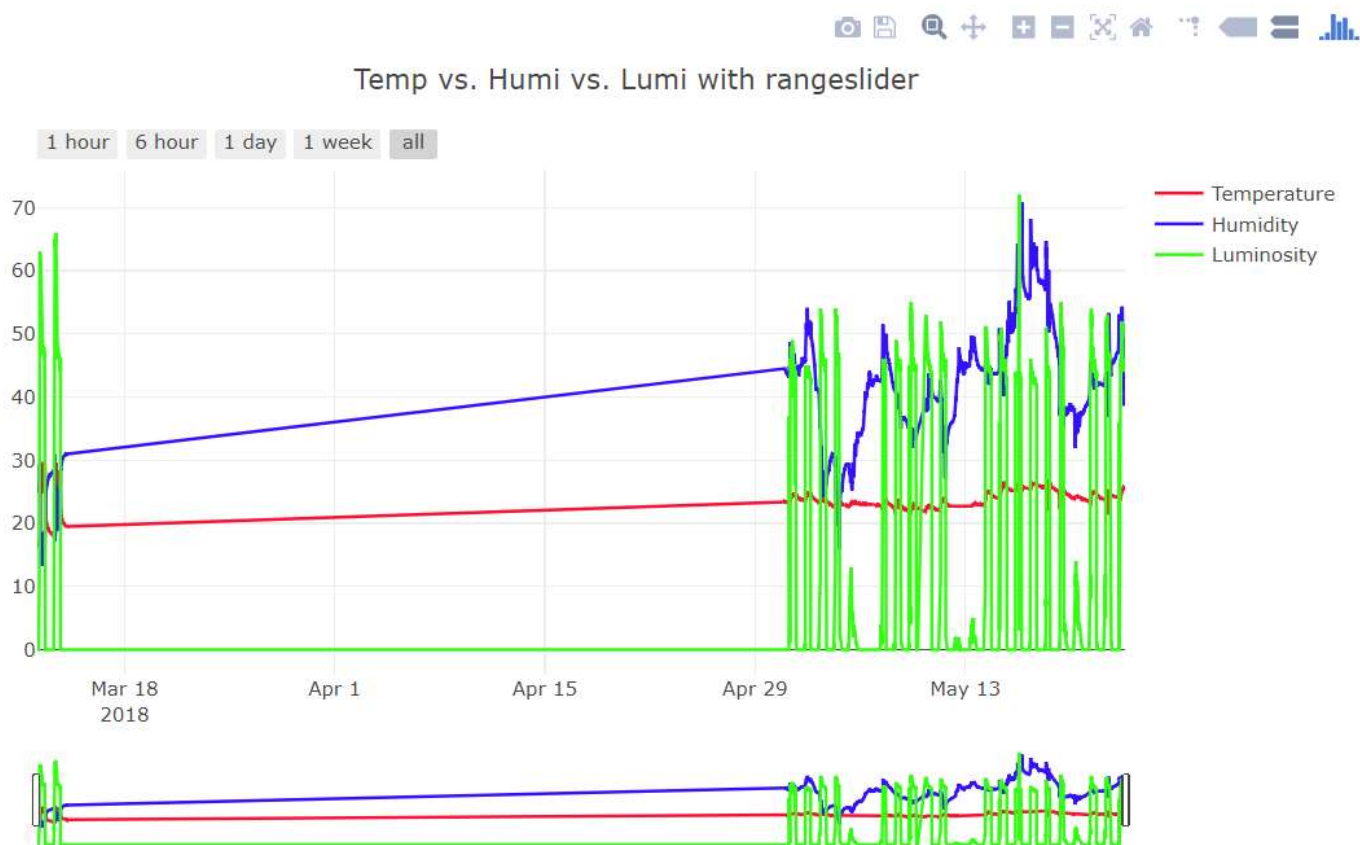


# DHT22 + CdS + Node.js + MongoDB

## [Next week] Web monitoring

### MongoDB database visualization by AA00

Time series : Multi sensor data

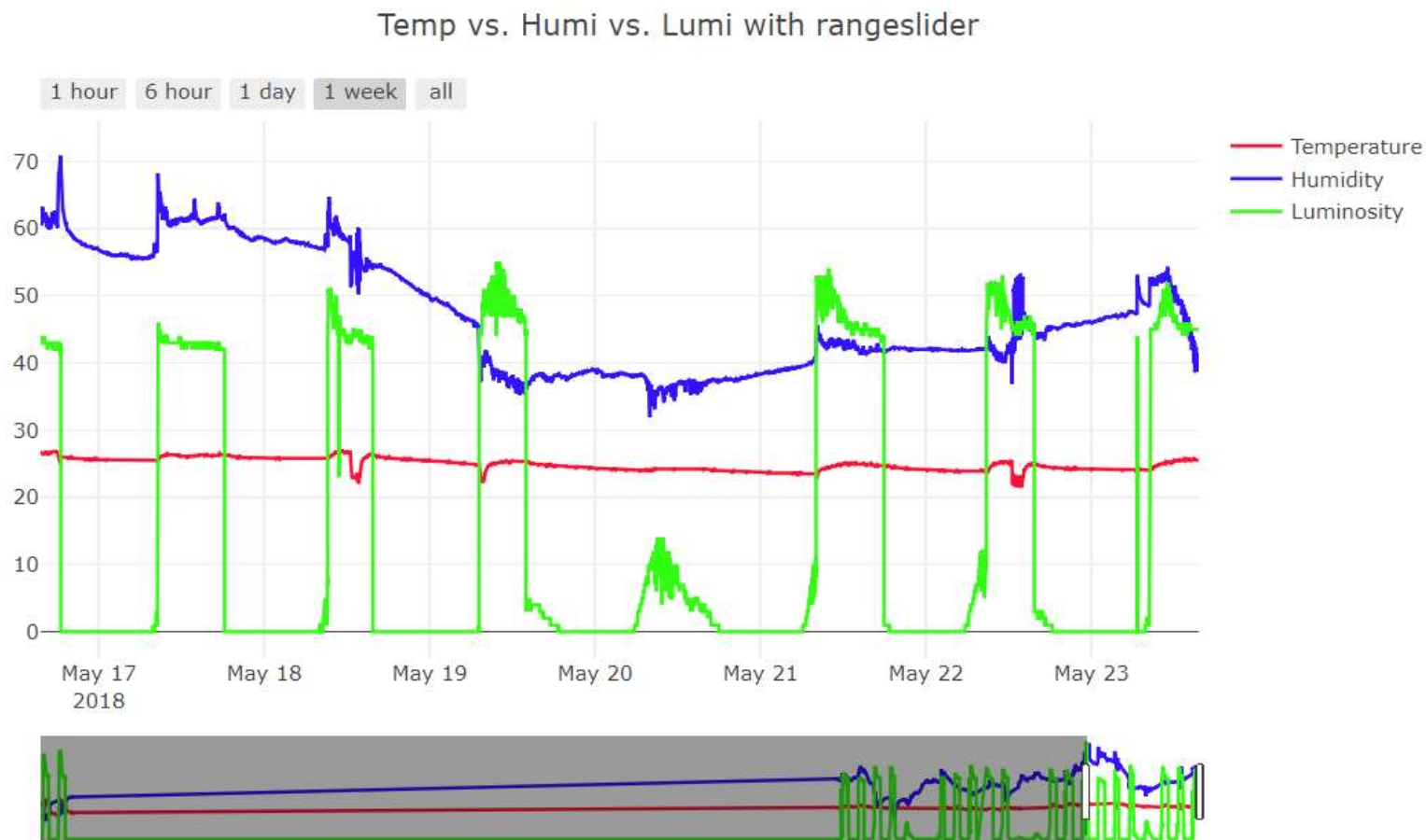


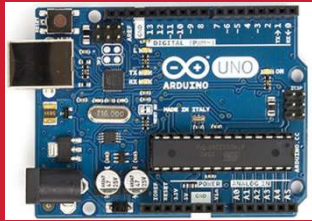


# DHT22 + CdS + Node.js + MongoDB

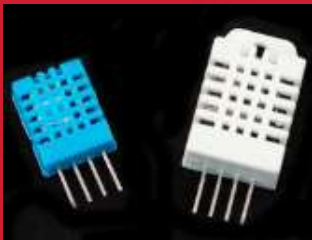
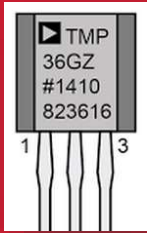
## [Next week] Web monitoring

### Time series : Multi sensor data





# [Practice]



## ◆ [wk13]

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

**Email : [chaos21c@gmail.com](mailto:chaos21c@gmail.com)**

**[ 제목 : id, 이름 (수정) ]**

## ● References & good sites

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





# 주교재 및 참고도서

아두이노와 Node.js에 기반한 IOT 신호 시각화

| 저자 이 상 훈 |

인제대학교 출판부

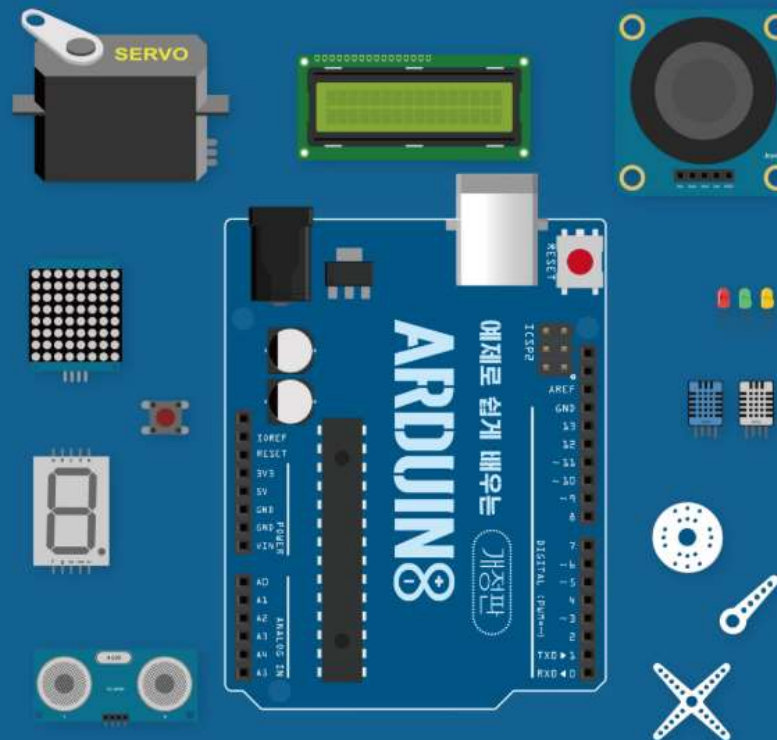
아두이노와 Node.js에 기반한

## IOT 신호 시각화

| 저자 이 상 훈 |



인제대학교 출판부



예제로 쉽게 배우는

## 아두이노

개정판

장성용 · 김진환 지음

인제대학교 출판부

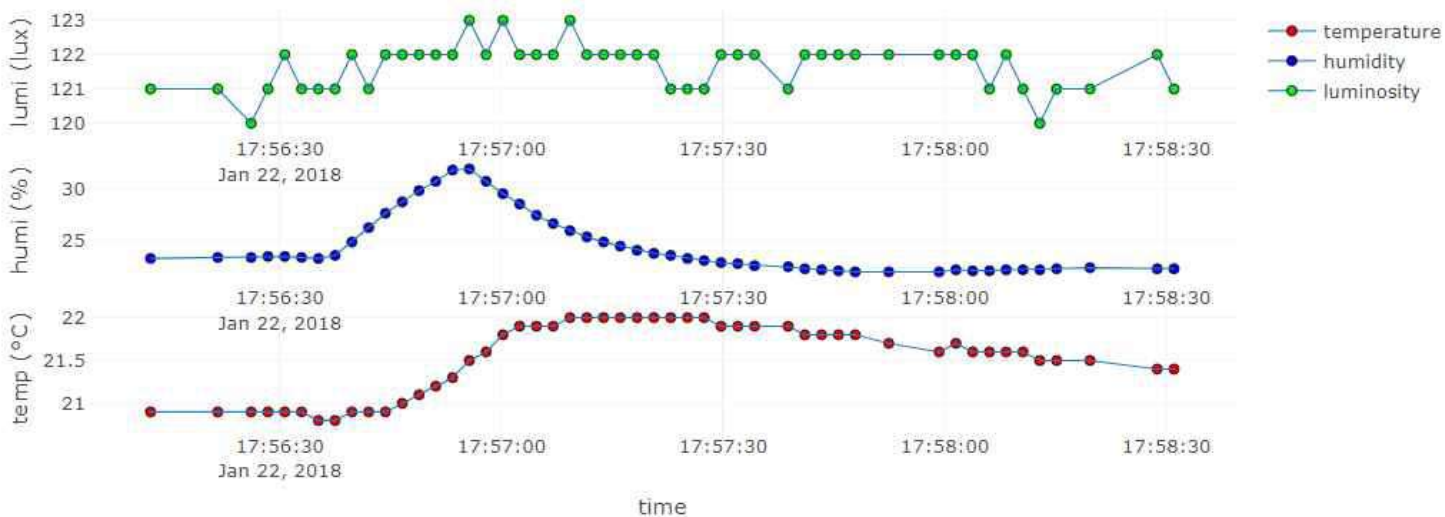


# Target of this class

## Real-time Weather Station from sensors



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



# Another target of this class

PPG with rangeslider

