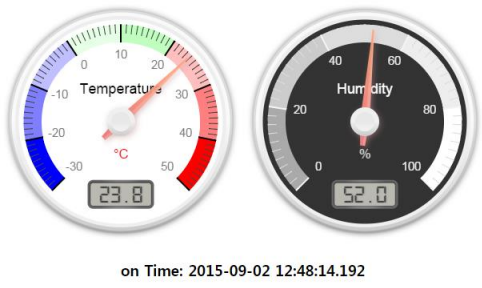
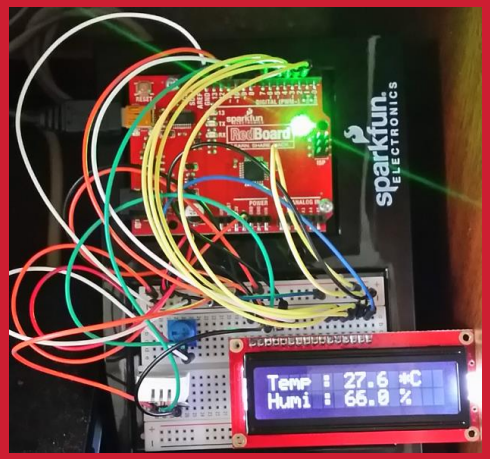
Healthcare-IOT
[wk14]





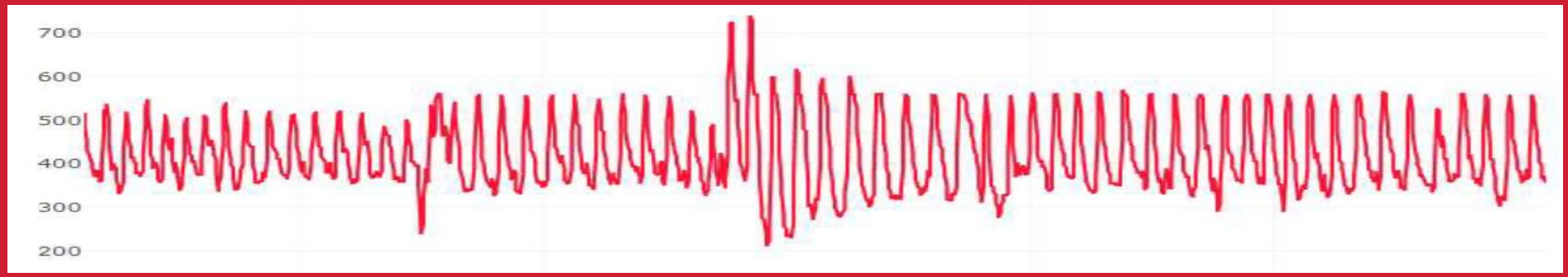
HealthCare Data -ECG/HR- Project

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

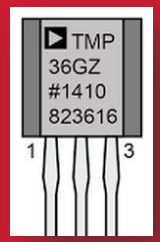
주간계획서			
주차	수업방법	수업내용	과제물
1	강의/실습	수업 및 실습 안내 -포터블 소프트웨어 설치	
2	강의/실습	Node.js I - Node.js 코드의 기본 구조 - 기초 Node 서버 및 클라이언트	실습확인
3	강의/실습	Node.js II - Node.js Express 서버	실습확인
4	강의/실습/발표	Arduino I - 아날로그 신호 회로 - LCD를 이용한 센서 신호 모니터링	실습확인
5	강의/실습	Arduino II - 단일 센서 회로와 Node.js 연결 - 다중 센서 회로와 Node.js 연결	실습확인
6	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	

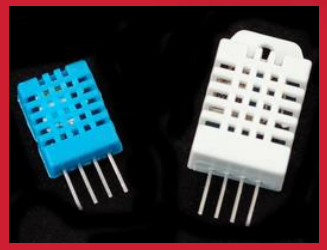
Purpose of HS

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]





- [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

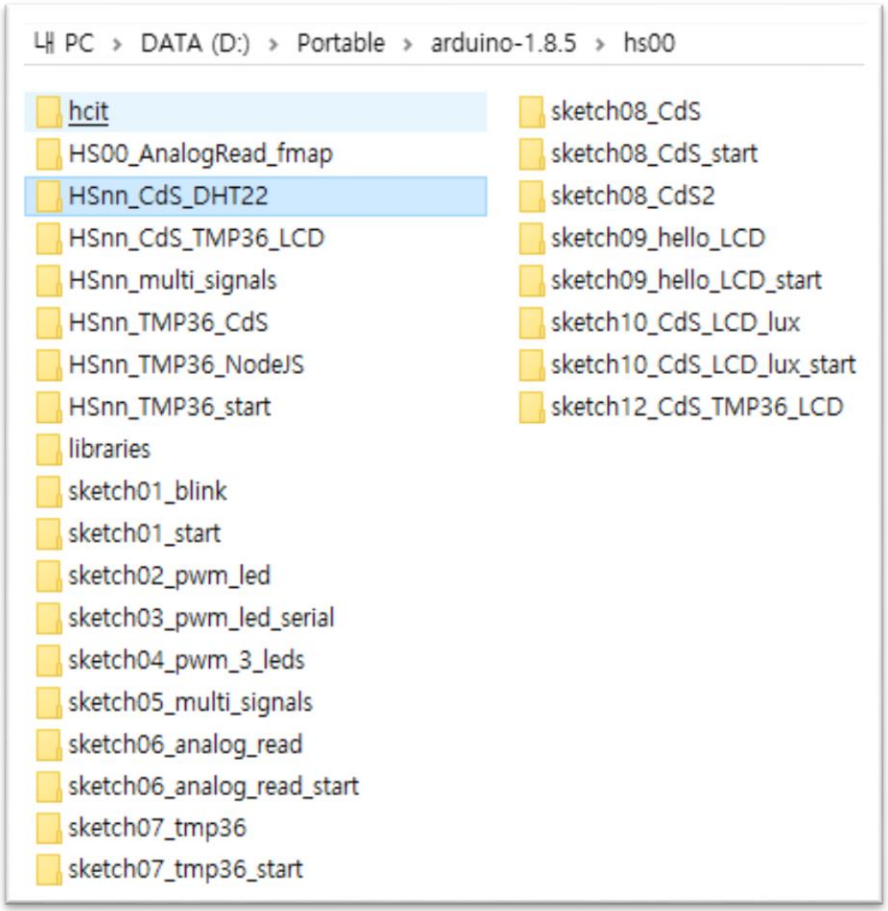
④ HSnn_iot_DB.png

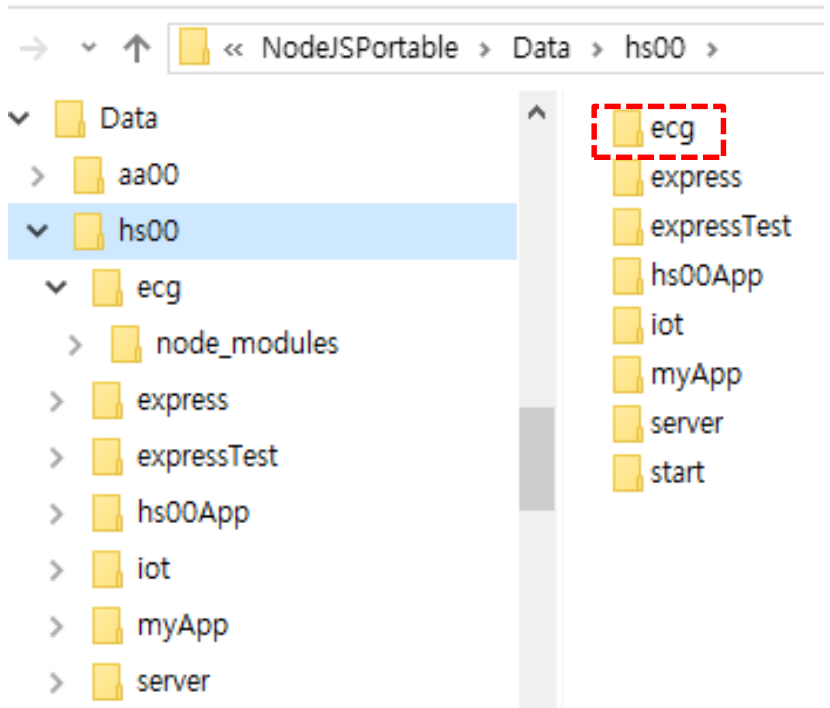
⑤ HSnn.csv (mongoexport file)

Email : chaos21c@gmail.com

[제목 : id, 이름 (수정)]

[My working folder – wk14]



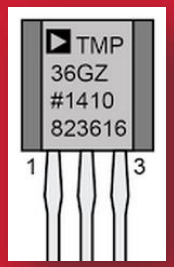


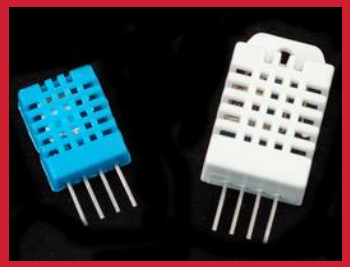
Arduino

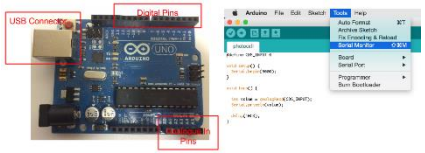
+ Node.js

+ plotly.js

+ MongoDB







IOT: HSC

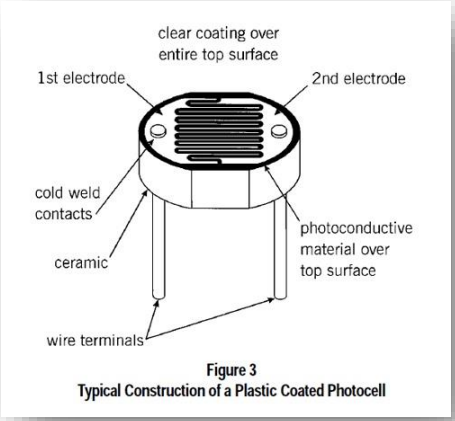
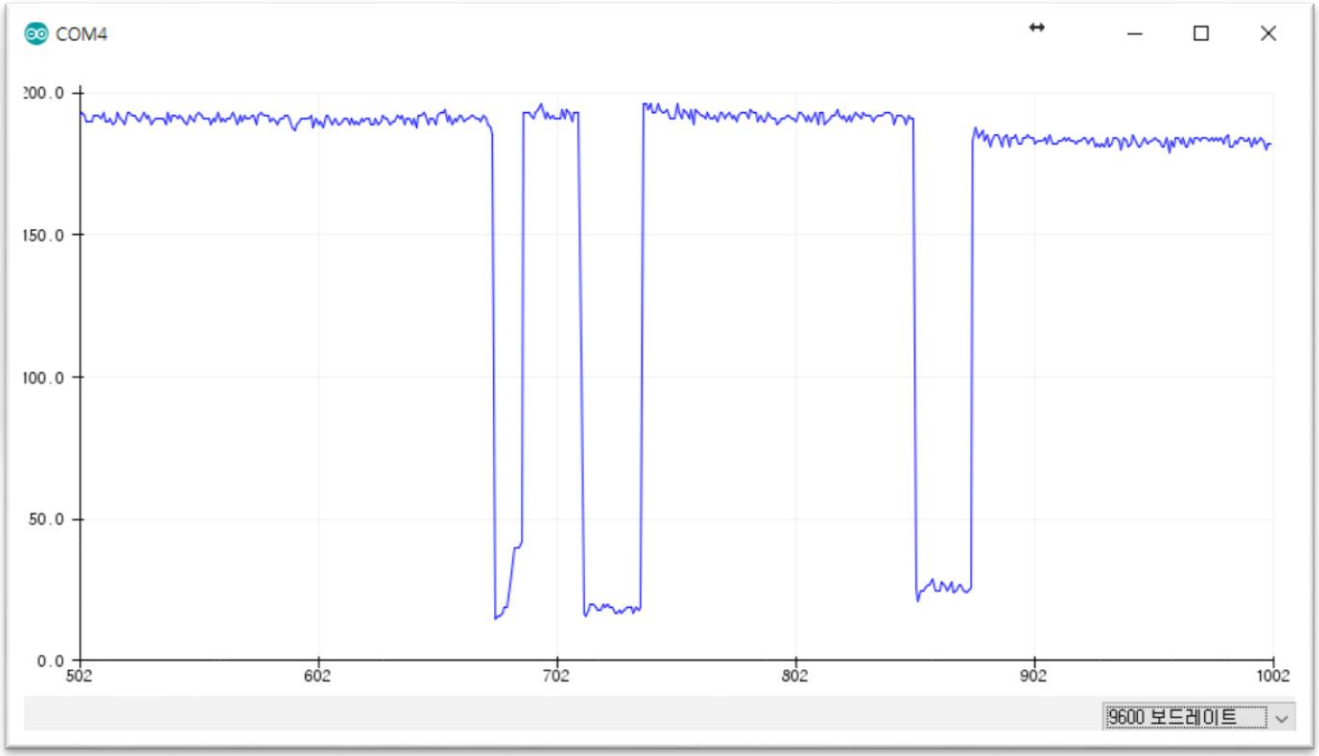
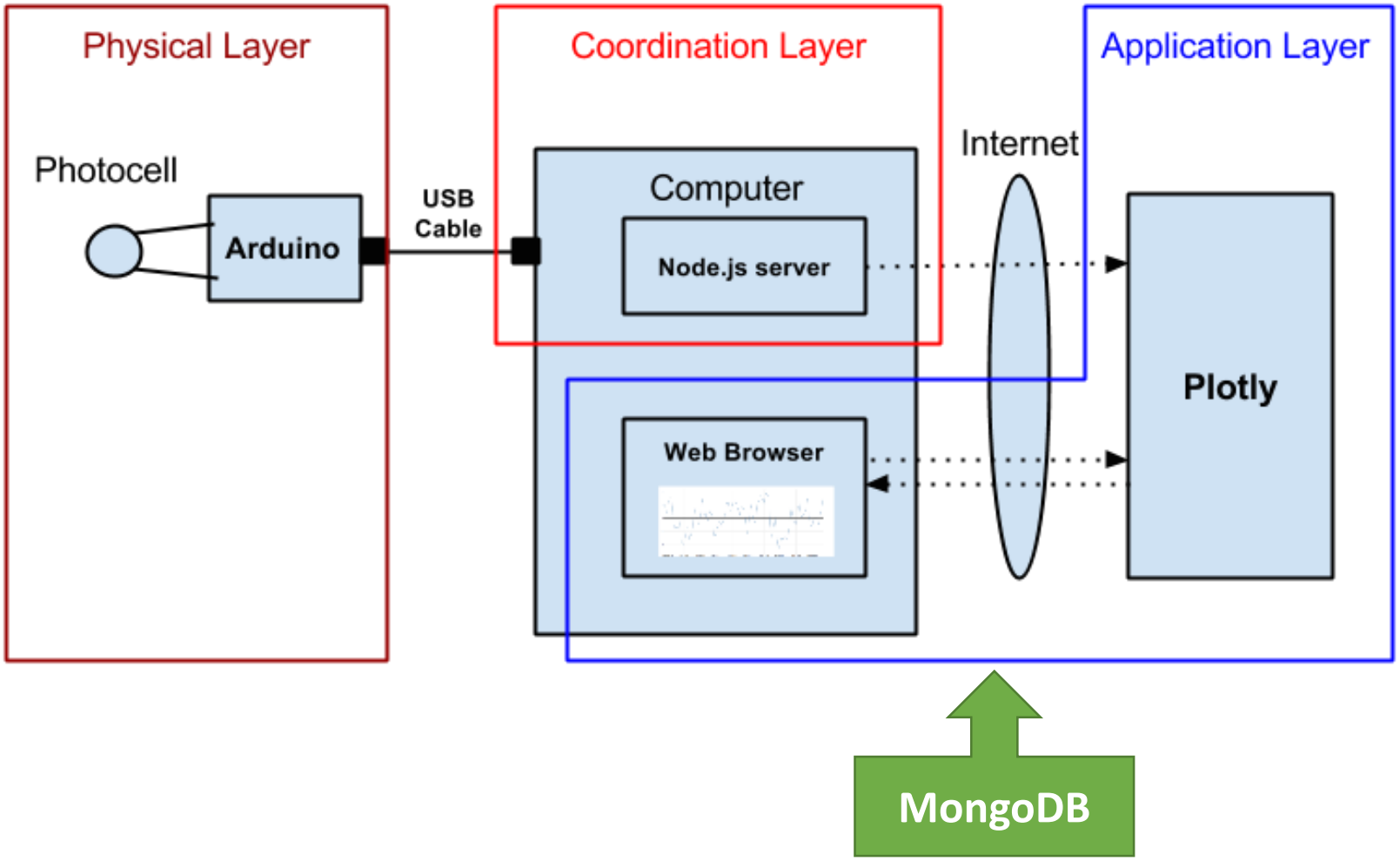


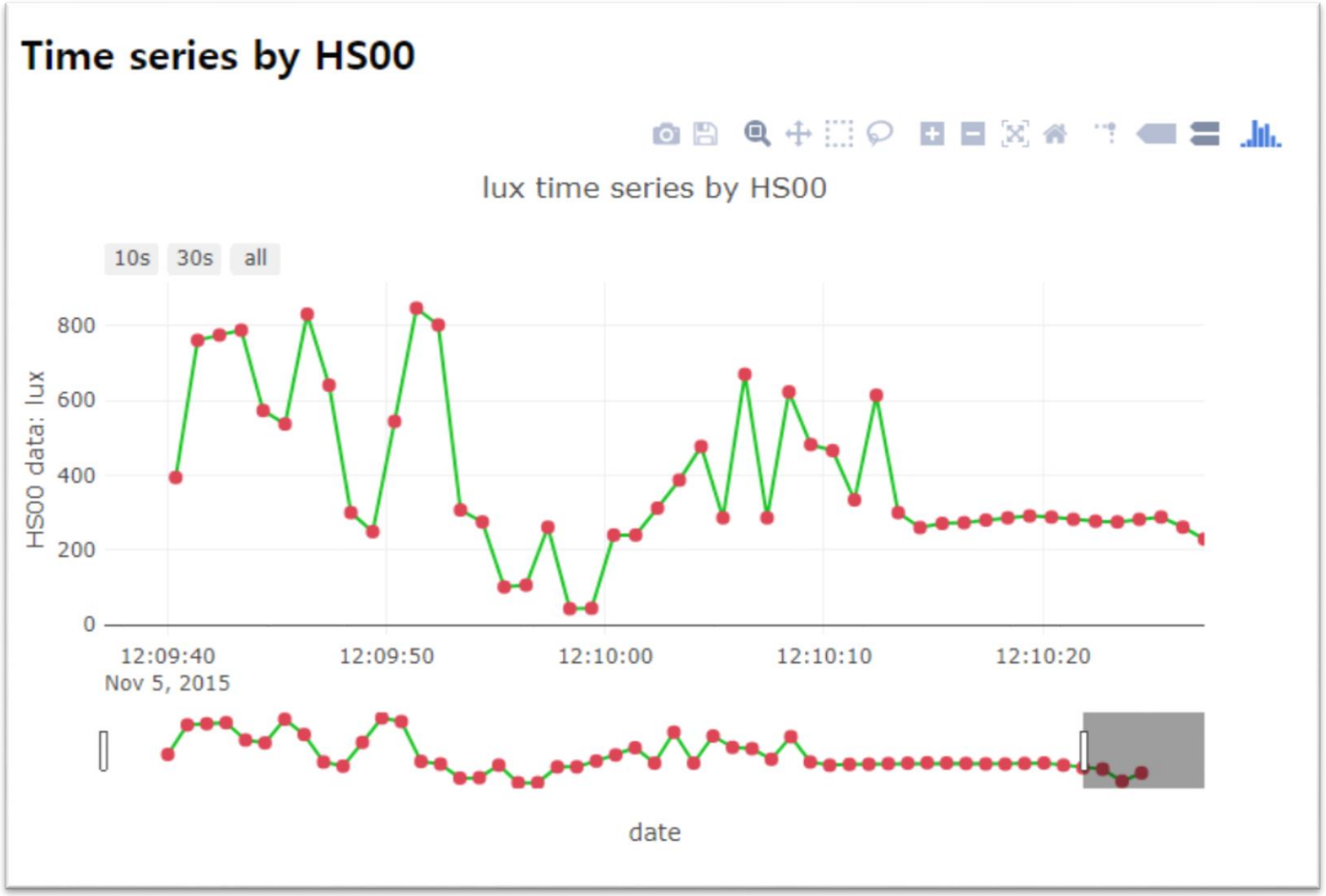
Figure 3
Typical Construction of a Plastic Coated Photocell



Layout [H S C]

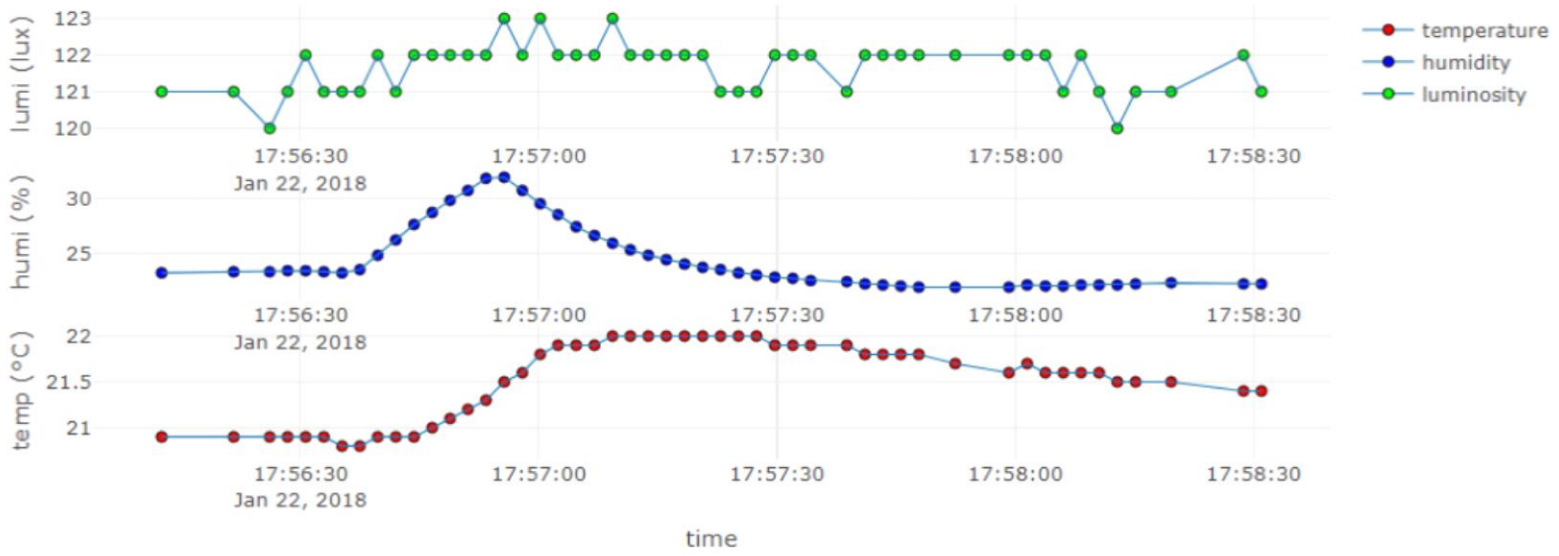

Physical Layer
Photocell
Arduino
USB Cable
Coordination Layer
Computer
Node.js server
Web Browser
Internet
Application Layer
Plotly
MongoDB

Arduino data + plotly



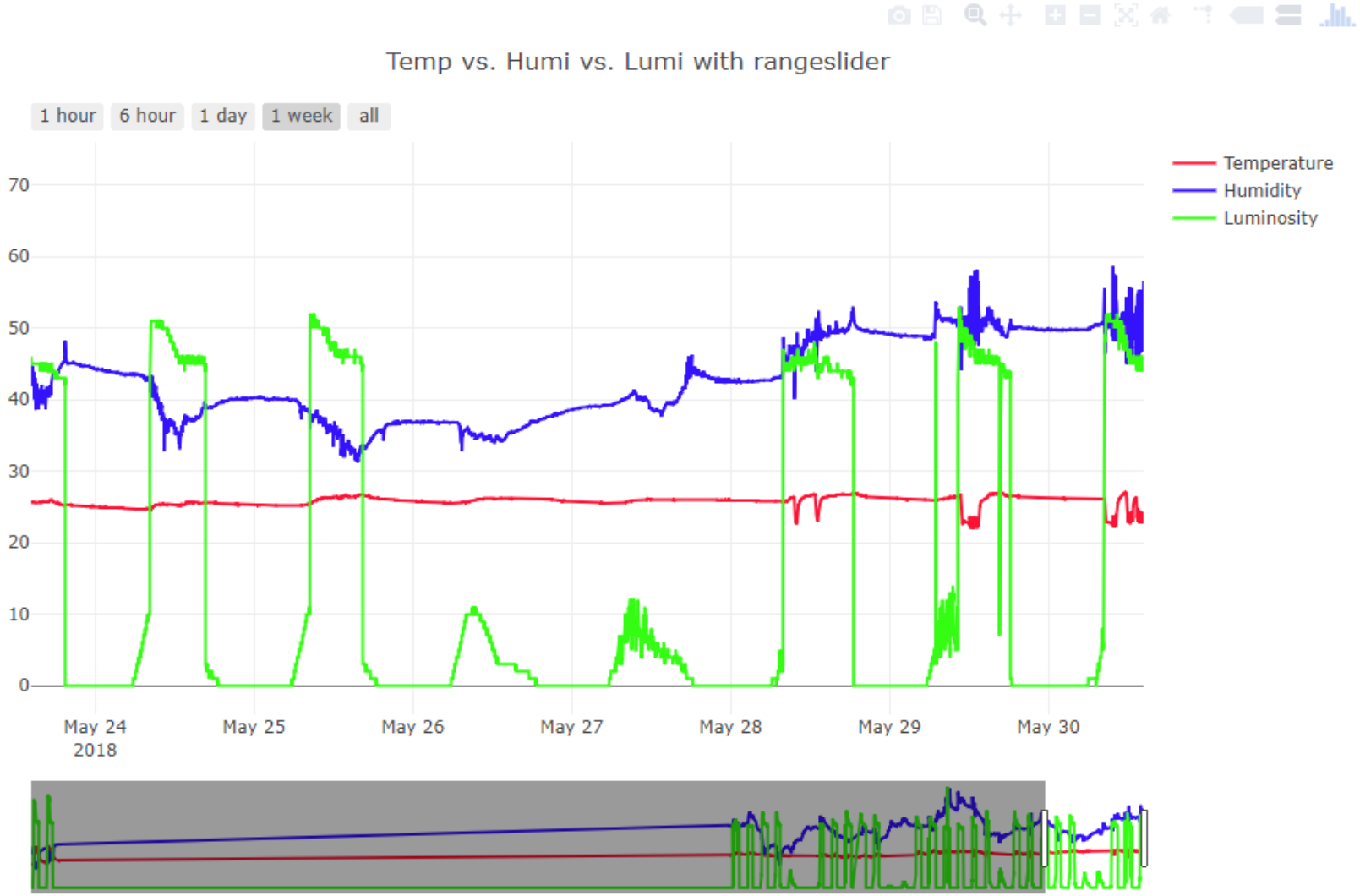
Real-time Weather Station from sensors

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



MongoDB database visualization by HS00

Time series : Multi sensor data

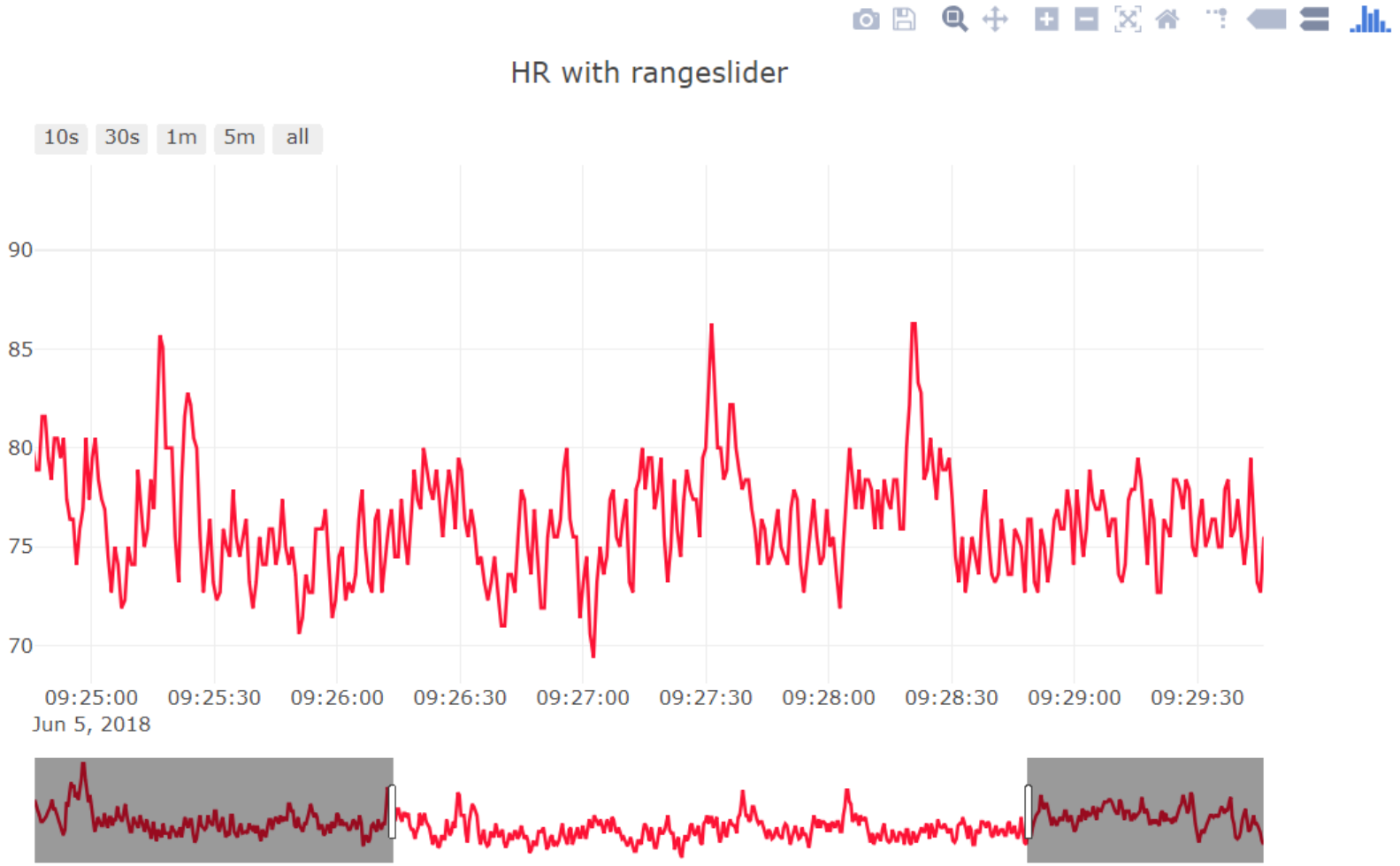


Data visualization by HS00

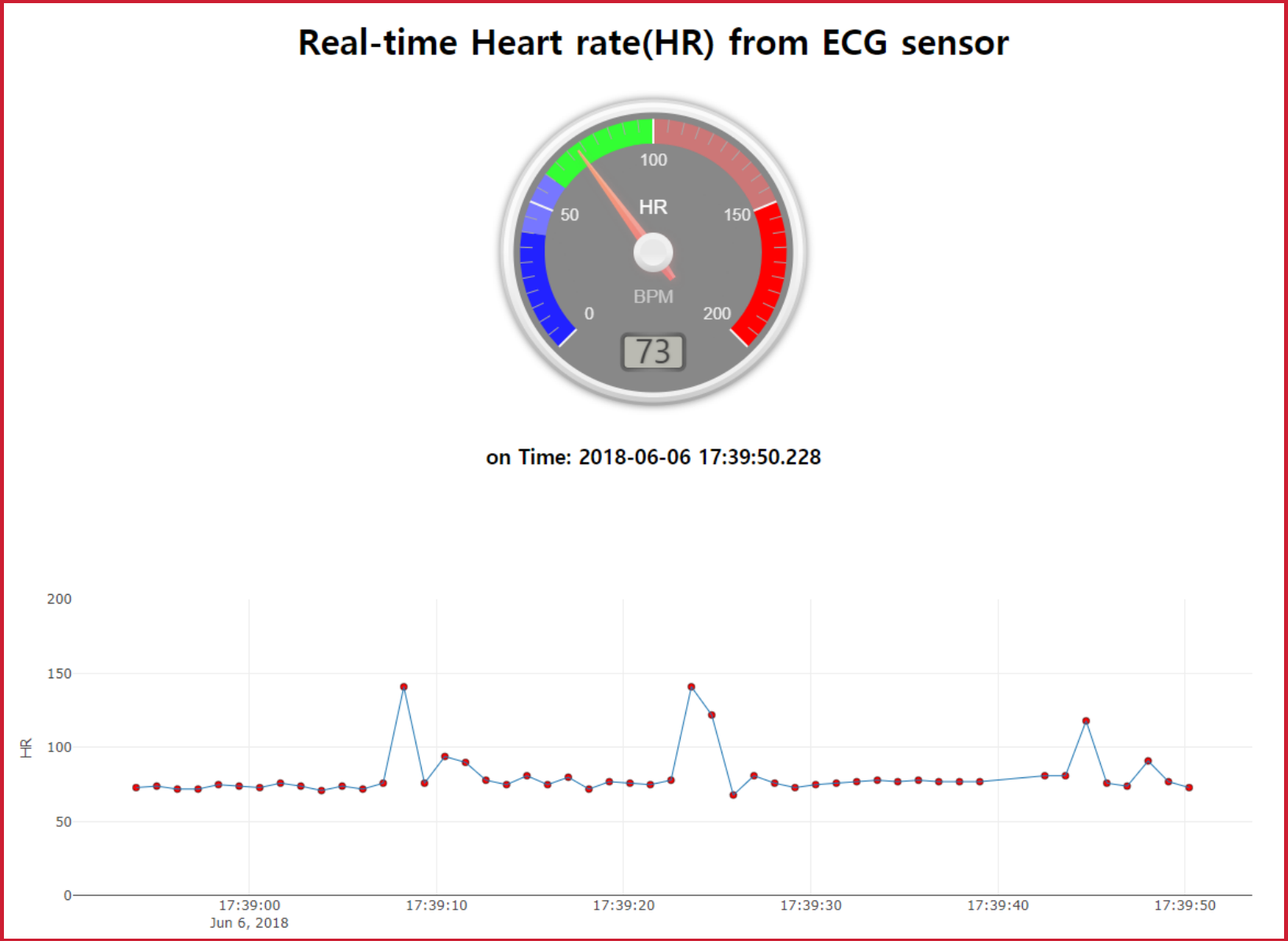
Time series : ECG & PPG

Data visualization by Biochaos

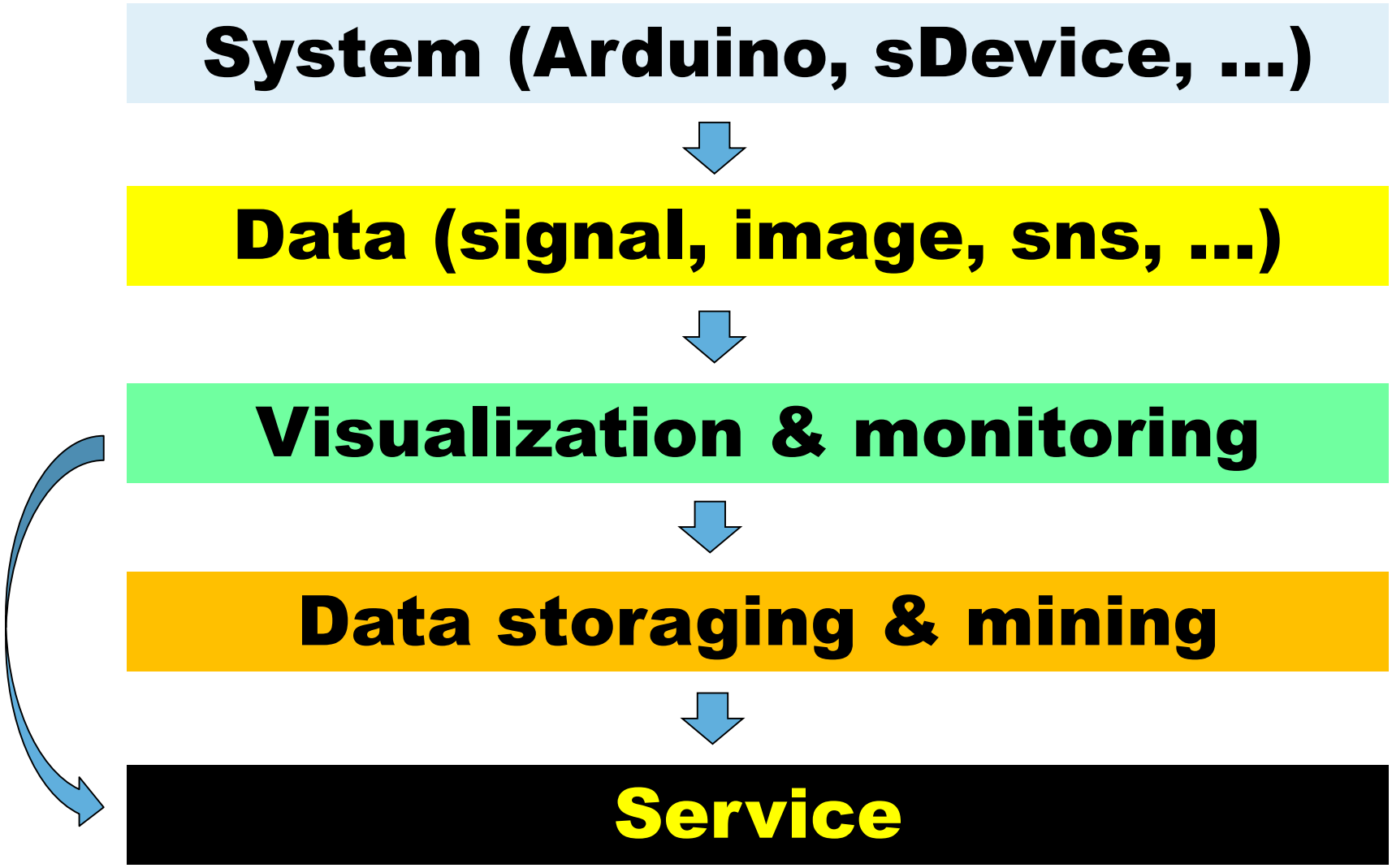
Time series : HR & PR

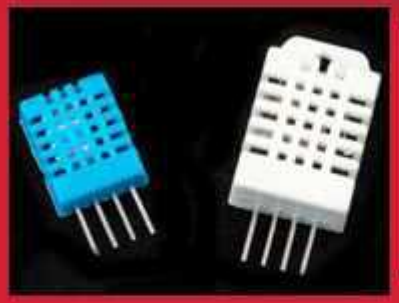


HR: Web client→ Real-time streaming



A5. Introduction to visualization





[Goal of project]

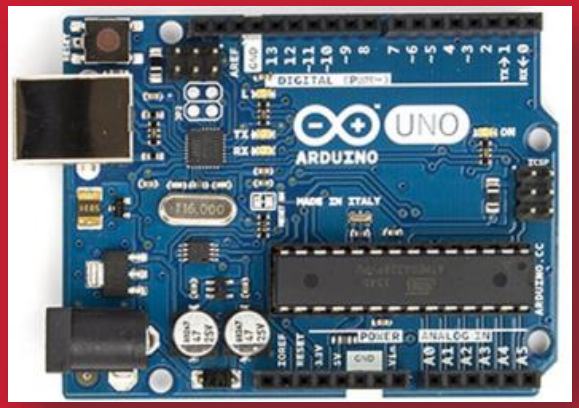
Arduino + Node.js

+ plotly.js

+ MongoDB

→ Data storaging

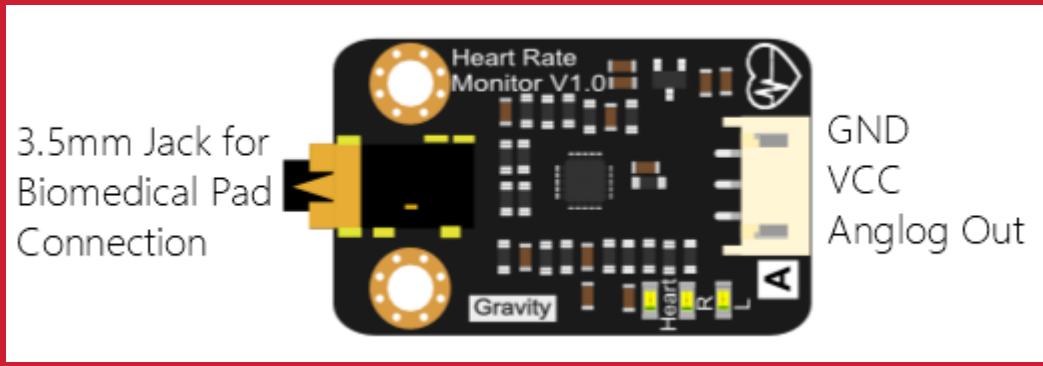
& visualization



Arduino circuit

ECG sensor Module

SEN0213

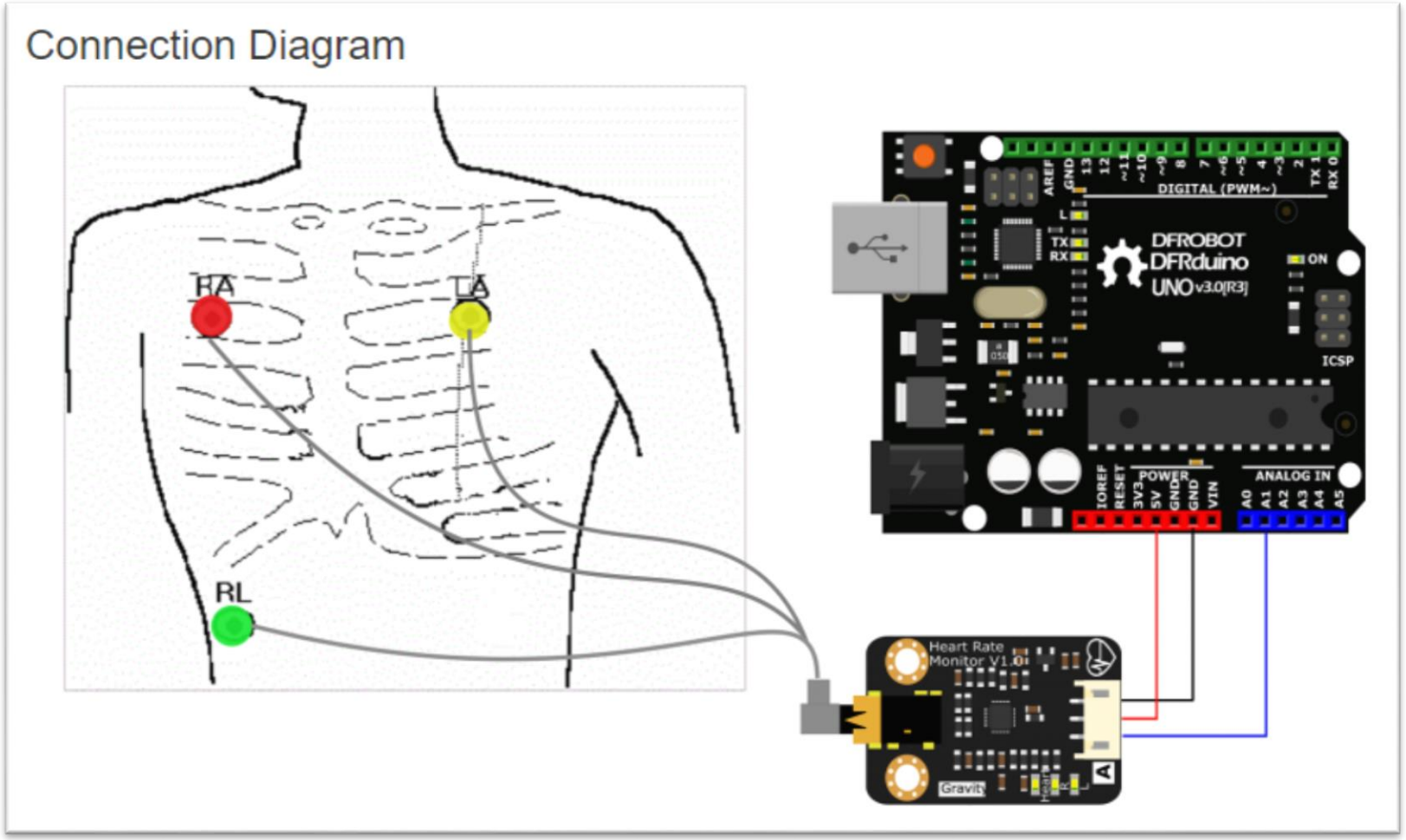


ECG & HR: sensor module – SEN0213

Heart Rate Monitor Sensor SKU: SEN0213

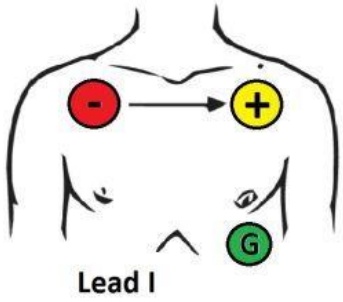
https://www.dfrobot.com/product-1510.html

ECG & HR: circuit

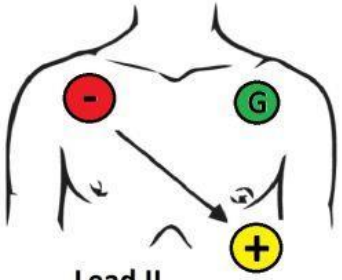


https://www.dfrobot.com/wiki/index.php/Heart_Rate_Monitor_Sensor_SKU:_SEN0213

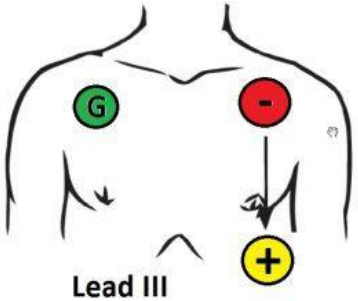
ECG: standard wave form – lead I, II, III



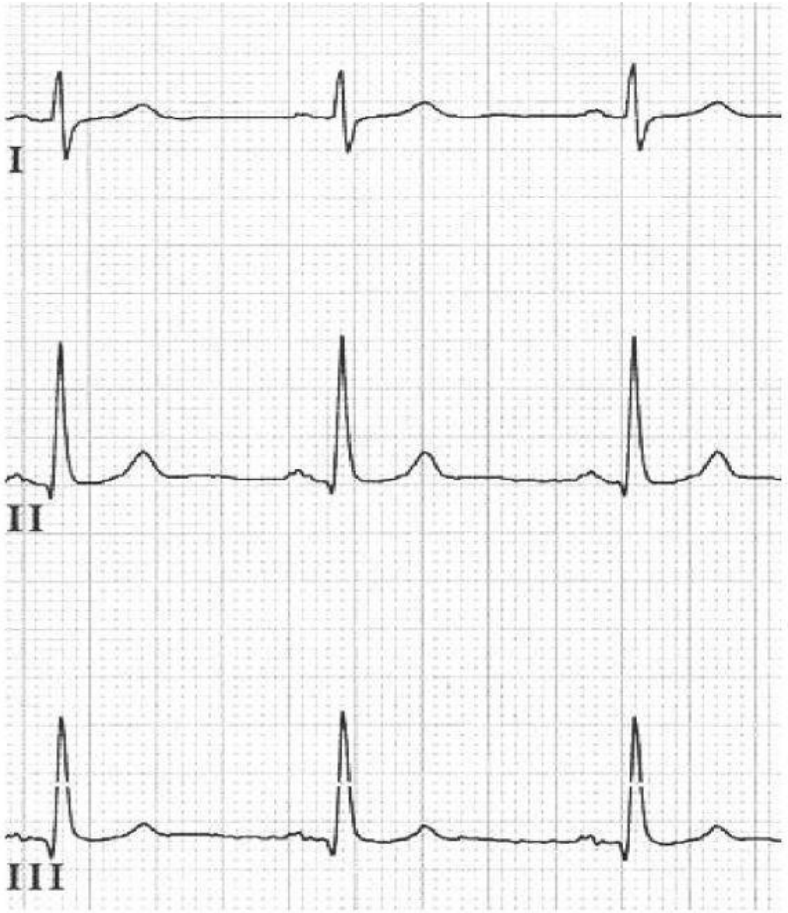
Lead I



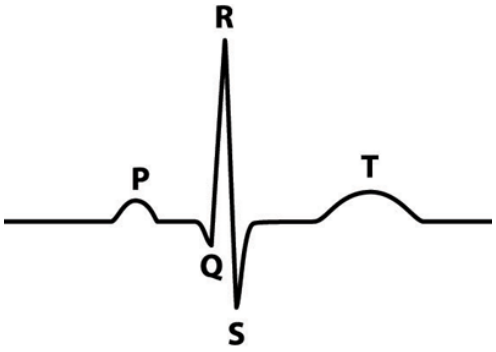
Lead II



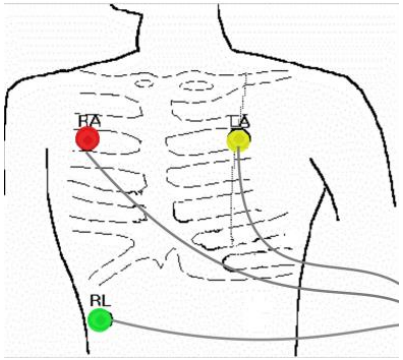
Lead III

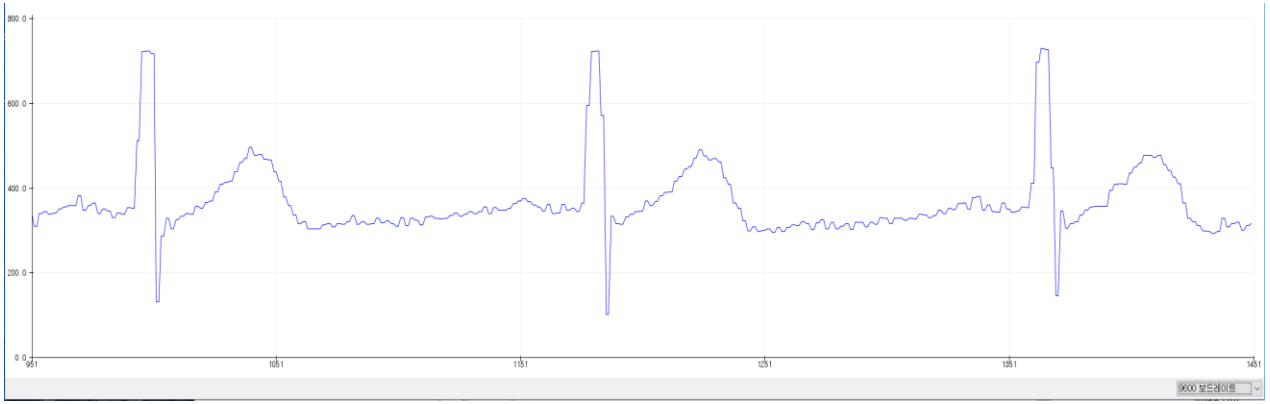


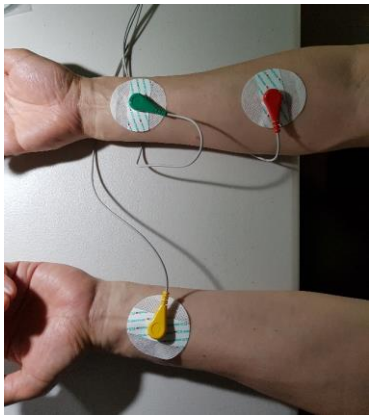
Normal ECG

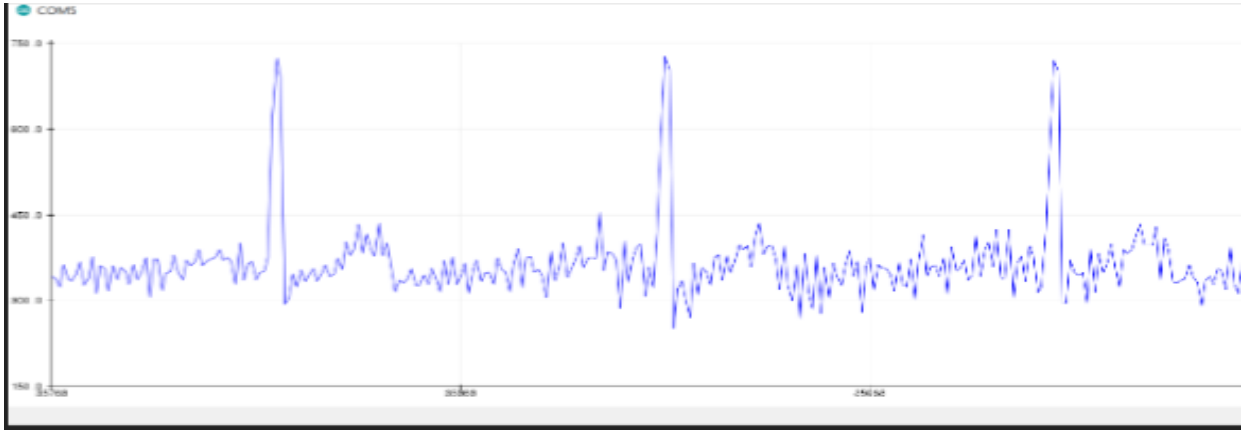


ECG: wave form





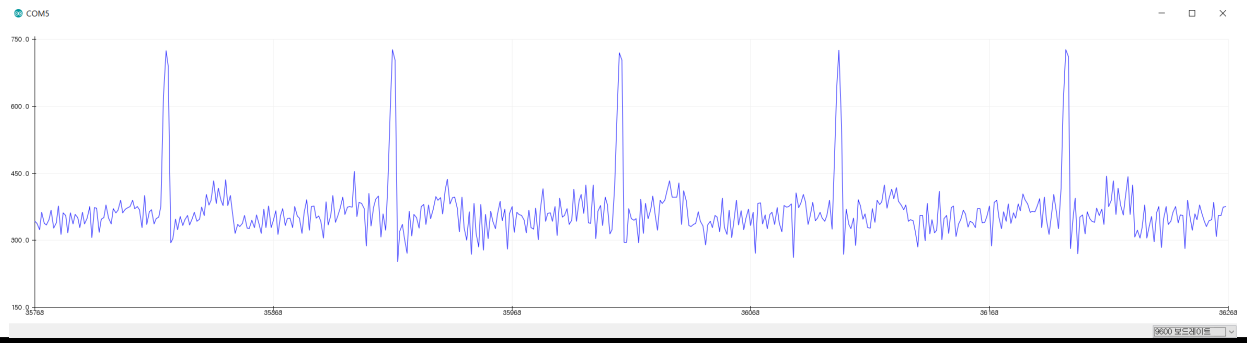




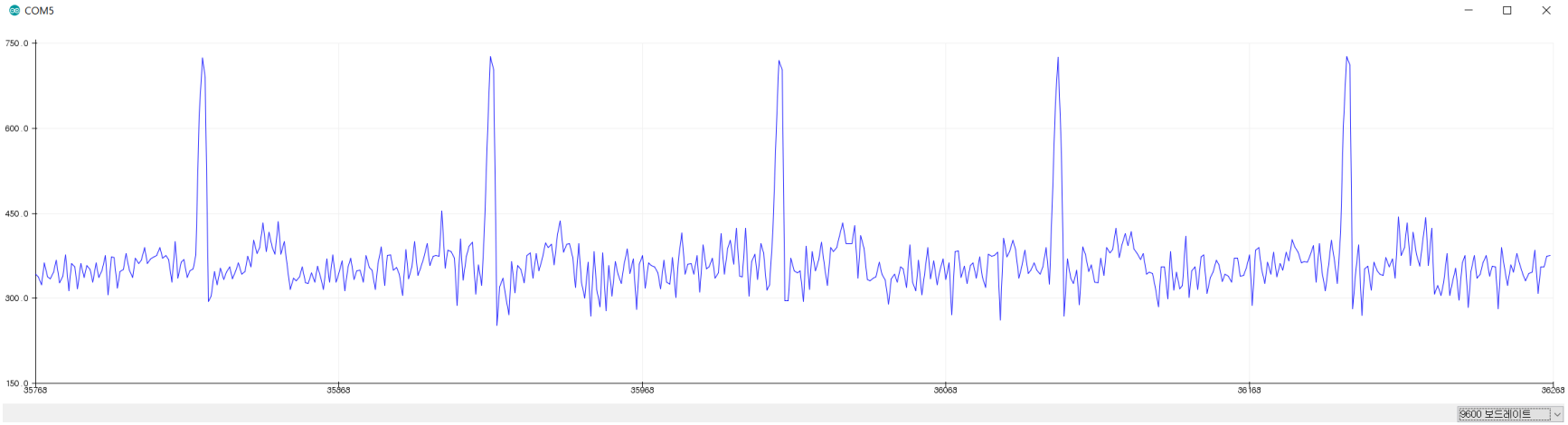
ECG: Arduino code – 1

```
11 const int heartPin = A1;
12
13 void setup() {
14   Serial.begin(19200);  // 9600, 115200
15 }
16
17 void loop() {
18 int heartValue = analogRead(heartPin);
19 Serial.println(heartValue);
20 delay(10);  // fs = 100 Hz (sampling period = 10 sec)
21 }
```

delay(10) → fs = 100 Hz



ECG: Arduino code - 1



작은 파형들은 전기적 잡음이며

전선이 흔들리면 서서히 변하는 잡음이 발생.

HR: Arduino code - 2

hcnn_ecg_HR §

```
14 #include "HeartSpeed.h"
15
16 HeartSpeed heartspeed(A1);            ///<The serial port for observing pulse.
17 //HeartSpeed heartspeed(A1,RAW_DATA); ///<The serial port observation of ECG wave.
18
19 /* Print the position result */
20 void mycb(uint8_t rawData, int value)  // call-back function
21 {
22   if(rawData){
23     Serial.println(value);
24   }else{
25     Serial.print("HeartRate Value = ");
26     Serial.print(value);
27     Serial.println(" (bpm)");
28   }
29 }
30 void setup() {
31   Serial.begin(115200);
32   heartspeed.setCB(mycb);    ///Callback function.
33   heartspeed.begin();///The pulse test.
34 }
35
36 void loop() {
37
38 }
```

ECG

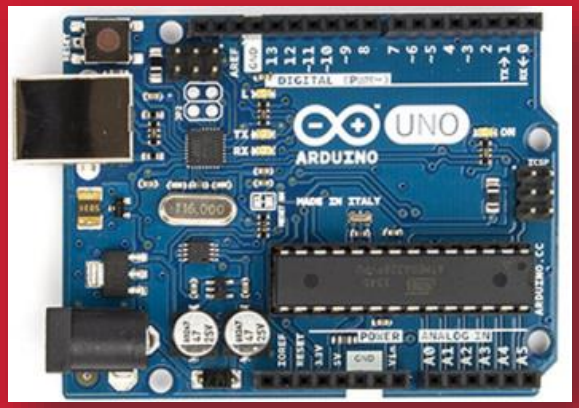
HR

HR

COM9 (Arduino Uno)

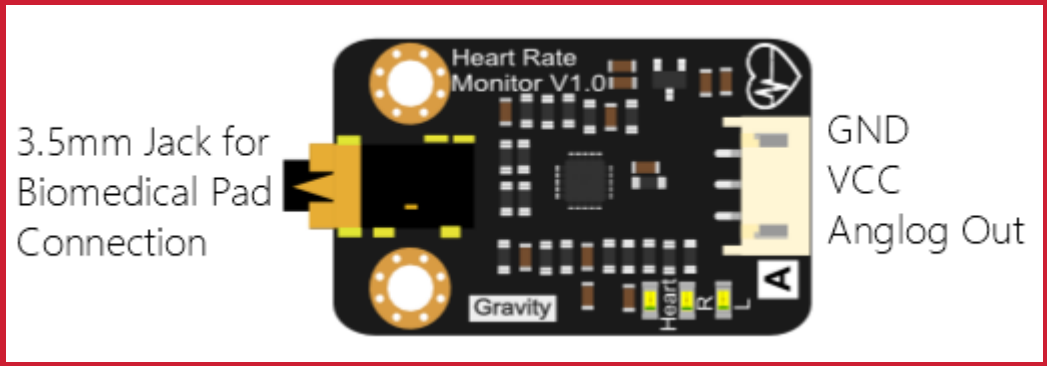
```
HeartRate Value = 69
HeartRate Value = 71
HeartRate Value = 72
HeartRate Value = 74
HeartRate Value = 75
HeartRate Value = 75
HeartRate Value = 74
HeartRate Value = 71
HeartRate Value = 62
HeartRate Value = 71
HeartRate Value = 68
HeartRate Value = 68
HeartRate Value = 78
HeartRate Value = 74
HeartRate Value = 77
```

Arduino & Node.js & MongoDB



ECG sensor Module

SEN0213



ECG: Node.js – package.json

package.json

```
{
  "name": "ecg_hr",
  "version": "1.0.0",
  "description": "ecg-hr-node project",
  "main": "ecg_node.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "hs00",
  "license": "MIT",
  "dependencies": {
    "cors": "^2.8.4",
    "express": "^4.16.2",
    "mongoose": "^5.0.1",
    "serialport": "^4.0.7",
    "socket.io": "^1.7.3"
  }
}
```

ECG: Node.js – npm install

NodeJS

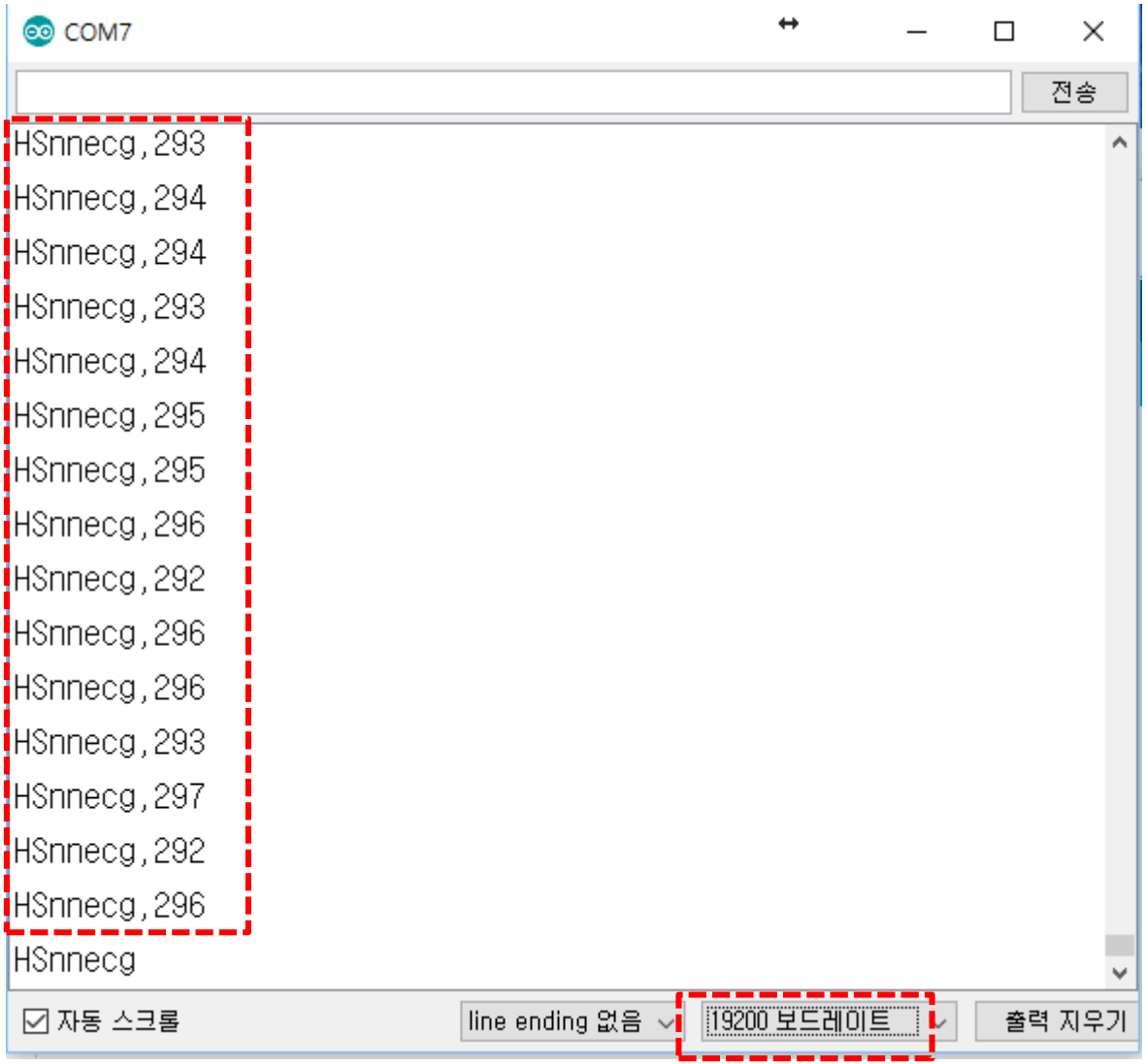
```
D:\Portable\NodeJSPortable\Data\hs00\ecg>dir
 D 드라이브의 볼륨: DATA
 볼륨 일련 번호: 7A01-106A

 D:\Portable\NodeJSPortable\Data\hs00\ecg 디렉터리

2018-06-05  오전 11:13    <DIR>          .
2018-06-05  오전 11:13    <DIR>          ..
2018-06-05  오전 11:13               393 package.json
               1개 파일                 393 바이트
               2개 디렉터리  867,170,316,288 바이트 남음

D:\Portable\NodeJSPortable\Data\hs00\ecg>npm install
```

```
 D:\Portable\NodeJSPortable\Data\hs00\ecg 디렉터리

2018-06-05  오전 11:22    <DIR>          .
2018-06-05  오전 11:22    <DIR>          ..
2018-06-05  오전 11:22    <DIR>          node_modules
2018-06-05  오전 11:13               393 package.json
               1개 파일                 393 바이트
               3개 디렉터리  867,142,000,640 바이트 남음
```

ECG: Arduino code – 1, new

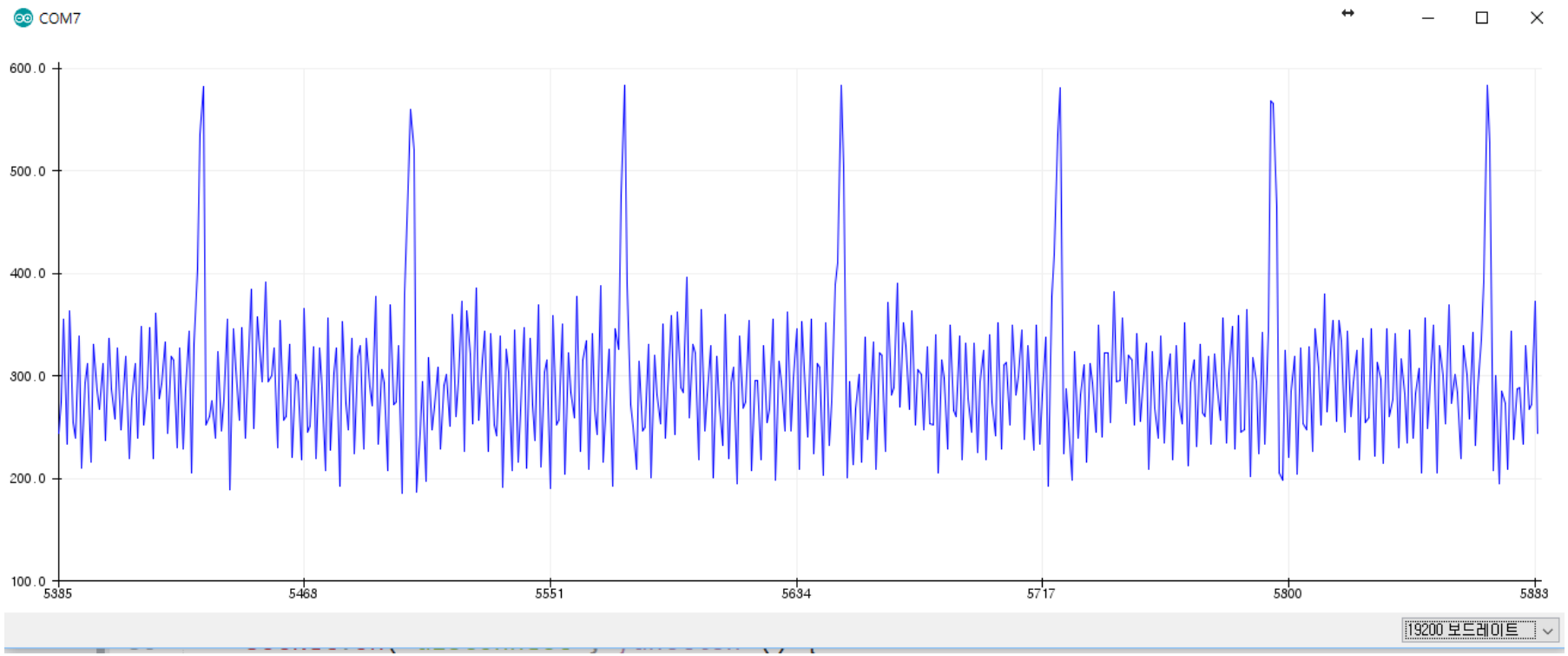
```
11 const int heartPin = A1;
12
13 void setup() {
14   Serial.begin(19200);  // 9600, 115200
15 }
16
17 void loop() {
18 int heartValue = analogRead(heartPin);
19 Serial.print("HSnnecg,");
20 Serial.println(heartValue);
21 delay(10);  // fs = 100 Hz (sampling period = 10 sec)
22 }
```

ECG: Arduino code – 1, new



직렬 통신 속도를 일치시켜야 신호 확인!

ECG: Arduino code – 1, new



전기적 잡음이 과도함...

ECG: Node.js – ecg_node_mongodb.js

ecg_node_mongodb.js

```
// ecg_node_mongodb.js

var serialport = require('serialport');
var portName = 'COM6';  // check your COM port!!
var port     =   process.env.PORT || 3000;

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

// MongoDB
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
// MongoDB connection
mongoose.connect('mongodb://localhost:27017/ecg'); // DB name
    var db = mongoose.connection;
    db.on('error', console.error.bind(console, 'connection error:'));
    db.once('open', function callback () {
        console.log("mongo db connection OK.");
});
// Schema
var iotSchema = new Schema({
    date : String,
    ecg : String
});
// Display data on console in the case of saving data.
iotSchema.methods.info = function () {
    var iotInfo = this.date
    ? "Current date: " + this.date +", ECG: " + this.ecg
    : "I don't have a date"
    console.log("iotInfo: " + iotInfo);
}
```

ECG: Node.js – ecg_node_mongodb.js

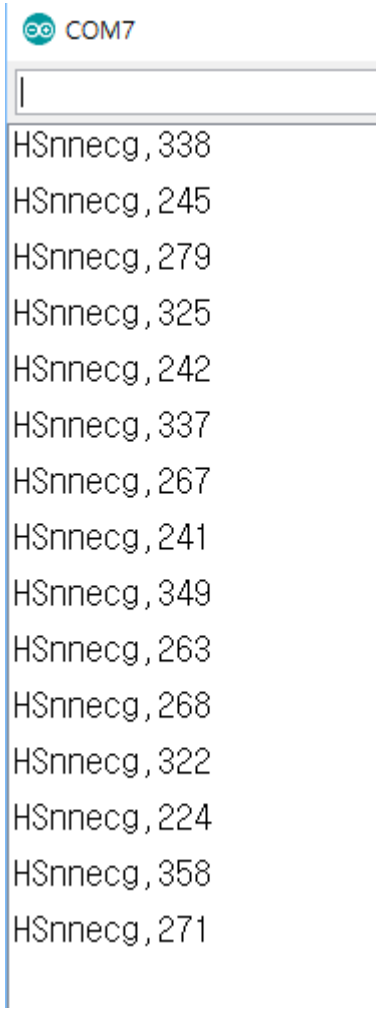
```
34 var sp = new serialport(portName,{
35     baudRate: 9600,    // 9600  38400
36     dataBits: 8,
37     parity: 'none',
38     stopBits: 1,
39     flowControl: false,
40     parser: serialport.parsers.readline('\r\n')
41 });
42 var readData = '';  // this stores the buffer
43 var ecgwave ='';
44 var mdata =[]; // this array stores date and data from multiple sensors
45 var firstcommaidx = 0;
46 var Sensor = mongoose.model("Sensor", iotSchema);  // sensor data model
47 sp.on('data', function (data) { // call back when data is received
48     readData = data.toString(); // append data to buffer
49     firstcommaidx = readData.indexOf(',');
50     // parsing data into signals
51     if (firstcommaidx > 0) {
52         ecgwave = readData.substring(firstcommaidx + 1);
53         readData = '';
54         dStr = getDateString();
55         mdata[0]=dStr;  // Date
56         mdata[1]=ecgwave; // ecg data
57         var iot = new Sensor({date:dStr,ecg:ecgwave});
58         // save iot data to MongoDB
59         iot.save(function(err, iot) {
60             if(err) return handleEvent(err);
61             iot.info();  // Display the information of iot data  on console.
62         })
63         io.sockets.emit('message', mdata);  // send data to all clients
64     } else {  // error
65         console.log(readData);
66     }
67 });
```

ECG: Node.js – ecg_node_mongodb.js

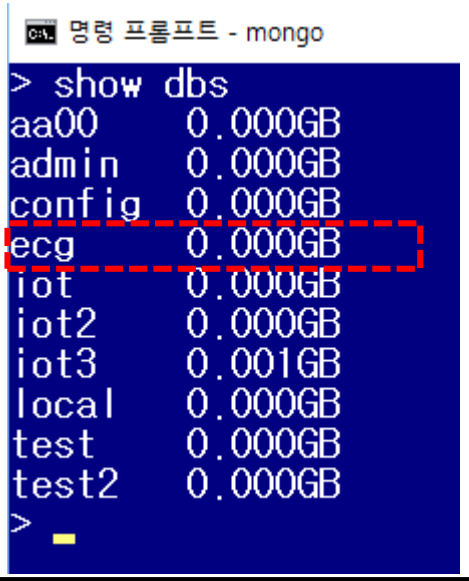
```
69  // helper function to get a nicely formatted date string for IOT
70  function getDateString() {
71      var time = new Date().getTime();
72      // 32400000 is (GMT+9 Korea, GimHae)
73      // for your timezone just multiply +/-GMT by 3600000
74      var datestr = new Date(time +32400000).
75      toISOString().replace(/T/, ' ').replace(/Z/, '');
76      return datestr;
77  }
78
79  io.sockets.on('connection', function (socket) {
80      // If socket.io receives message from the client browser then
81      // this call back will be executed.
82      socket.on('message', function (msg) {
83          console.log(msg);
84      });
85      // If a web browser disconnects from Socket.IO then this callback is called.
86      socket.on('disconnect', function () {
87          console.log('disconnected');
88      });
89  });
90
```

ECG: Node.js – ecg_node_mongodb.js

Run: node ecg_node_mongodb.js (Run in Node cmd)

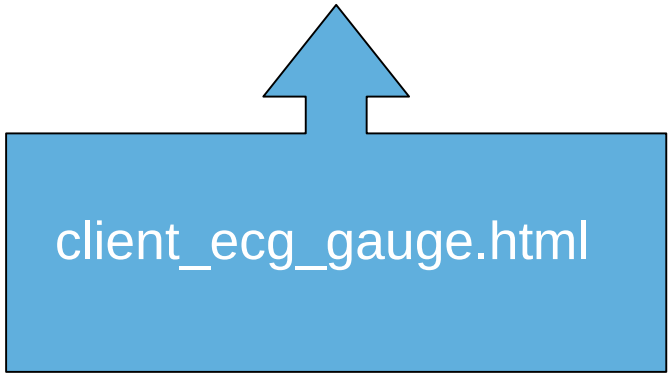



```
D:\Portable\NodeJSPortable\Data\hs00\ecg>node ecg_node_mongodb
mongo db connection OK.
iotInfo: Current date: 2018-06-05 12:37:32.257, ECG: 311
iotInfo: Current date: 2018-06-05 12:37:32.301, ECG: 308
iotInfo: Current date: 2018-06-05 12:37:32.303, ECG: 225
iotInfo: Current date: 2018-06-05 12:37:32.311, ECG: 323
iotInfo: Current date: 2018-06-05 12:37:32.311, ECG: 208
iotInfo: Current date: 2018-06-05 12:37:32.323, ECG: 226
iotInfo: Current date: 2018-06-05 12:37:32.355, ECG: 513
iotInfo: Current date: 2018-06-05 12:37:32.371, ECG: 454
iotInfo: Current date: 2018-06-05 12:37:32.387, ECG: 284
iotInfo: Current date: 2018-06-05 12:37:32.339, ECG: 506
```



[DIY]

Real-time WEB client to monitor ECG



ECG: Web client→ client_ecg_gauge.html – 1

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <title>plotly.js client: Real time signals from sensors</title>
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/
  socket.io.js"></script>

  <script src="gauge.min.js"></script>

  <style>body{padding:0;margin:30;background:#fff}</style>
</head>

<body>  <!-- style="width:100%;height:100%"> -->
<!-- Plotly chart will be drawn inside this DIV -->
<h1 align="center">Real-time Electrocardiogram(ecg) from ECG sensor</h1>
<!-- Lux gauge -->
<div align="center">
    <canvas id="gauge"> </canvas>
</div>

<h3 align="center"> on Time: <span id="time"> </span> </h3>

<div id="myDiv"></div> <!-- graph here! -->

<hr>
```

ECG: Web client→ client_ecg_gauge.html – 2

```
<script>
/* JAVASCRIPT CODE GOES HERE */
  var streamPlot = document.getElementById('myDiv');
  var ctime = document.getElementById('time');

  var tArray = [], // time of data arrival
      xTrack = [], // value of ECG sensor 1
      numPts = 400, // number of data points
      dtda = [],  // 1 x 2 array : [date, ecg]
      preX = -1,  // check change in data
      initFlag = true;
```

```
var socket = io.connect('http://localhost:3000'); // port = 3000
socket.on('connect', function () {
    socket.on('message', function (msg) {
        // initial plot
        if(msg[0]!='' && initFlag){
            dtda[0]=msg[0];
            dtda[1]=parseInt(msg[1]);  // ecg
            init();  // start streaming
            initFlag=false;
        }
        console.log(msg[0]);
        console.log(parseInt(msg[1])); // Convert value to integer
        dtda[0]=msg[0];
        dtda[1] = parseInt(msg[1]);

        // when new data is coming, keep on streaming data
        ctime.innerHTML = dtda[0];
        gauge_ecg.setValue(dtda[1]); // ecg gauge
        //nextPt();
        tArray = tArray.concat(dtda[0]);  // time
        tArray.splice(0,1);
        xTrack = xTrack.concat(dtda[1]);  // ecg
        xTrack.splice(0,1);

        var update = {
            x: [tArray],
            y: [xTrack]
        }
        Plotly.update(streamPlot, update);
    });
});
```

ECG: Web client→ client_ecg_gauge.html – 3

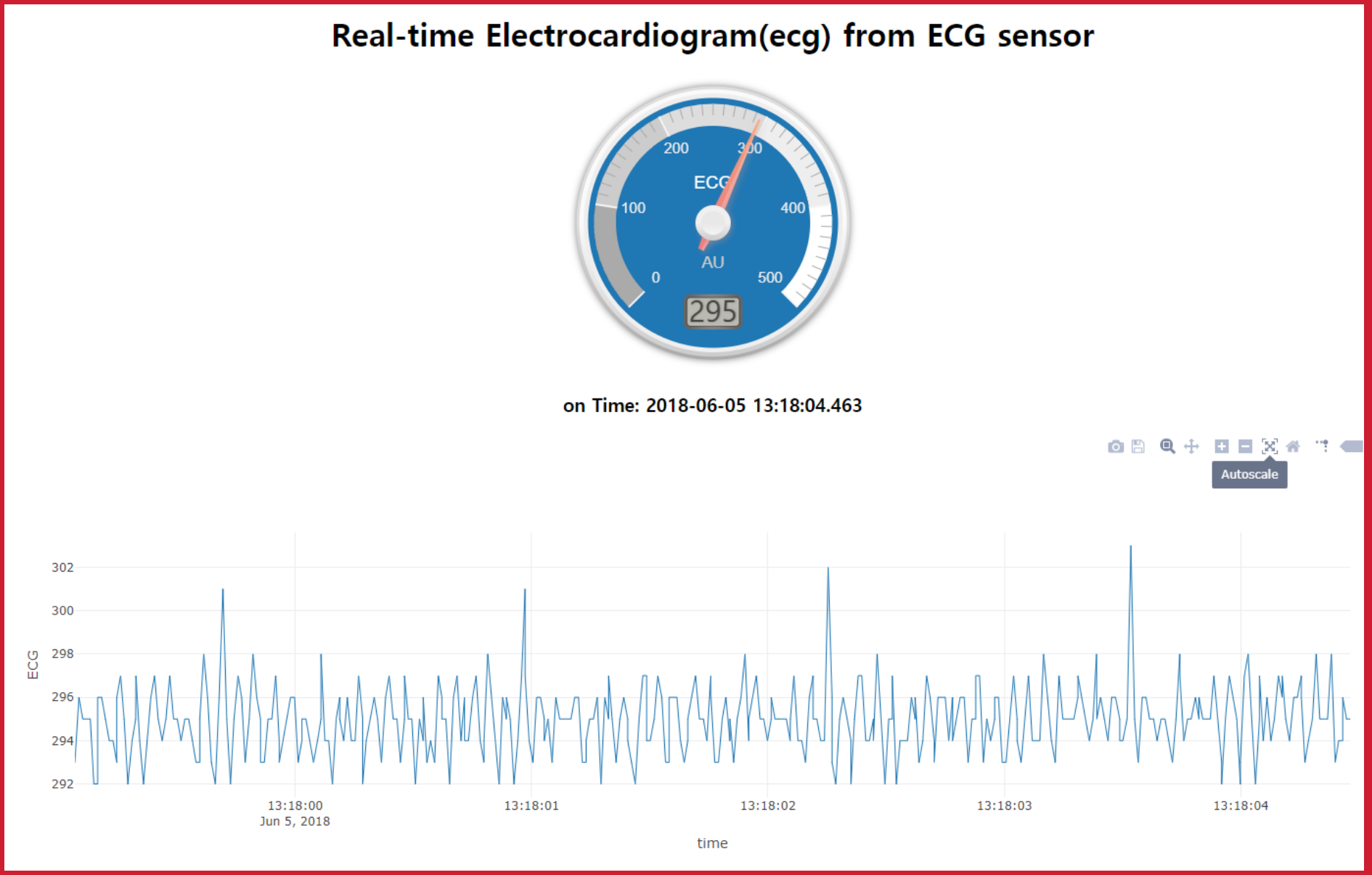
```
function init() {  // initial screen ()
    // starting point : first data (lux)
    for ( i = 0; i < numPts; i++) {
        tArray.push(dtda[0]);  // date
        xTrack.push(dtda[1]);  // ecg
    }

    Plotly.plot(streamPlot, data, layout);
}

// data
var data = [{
    x : tArray,
    y : xTrack,
    name : 'ECG',
    mode: "lines",  //markers+lines
    line: {
        color: "#1f77b4",
        width: 1
     }//,
    // marker: {
    //     color: "rgb(255, 0, 0)",
    //     size: 6,
    //     line: {
    //       color: "black",
    //       width: 0.5
    //     }
    //  }
}];
```

```
// layout
var layout = {
    xaxis : {
        title : 'time',
        domain : [0, 1]
    },
    yaxis : {
        title : 'ECG',
        domain : [0, 1],
        range : [0, 500]
    }
};
```

ECG: Web client→ client_ecg_gauge.html – 4

```
// gauge configuration
var gauge_ecg = new Gauge({
    renderTo    : 'gauge',
    width       : 300,
    height      : 300,
    glow        : true,
    units       : 'AU',
    valueFormat : { int : 2, dec : 0 },
    title       : "ECG",
    minValue    : 0,
    maxValue    : 500,  // new
    majorTicks  : ['0','100','200','300','400','500'],
    minorTicks  : 10,
    strokeTicks : false,
    highlights  : [
        { from : 0,   to : 100, color : '#aaa' },
        { from : 100, to : 200, color : '#ccc' },
        { from : 200, to : 300, color : '#ddd' },
        { from : 300, to : 400, color : '#eee' },
        { from : 400, to : 500, color : '#fff' }
    ],
    colors      : {
        plate      : '#df2578',
        majorTicks : '#f5f5f5',
        minorTicks : '#aaa',
        title      : '#fff',
        units      : '#ccc',
        numbers    : '#eee',
        needle     : { start : 'rgba(240, 128, 128, 1)',
        end : 'rgba(255, 160, 122, .9)' }
    }
});
gauge_ecg.draw();
```

ECG: Web client→ Real-time streaming



ECG: Node.js – ecg_express.js

ecg_express.js

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

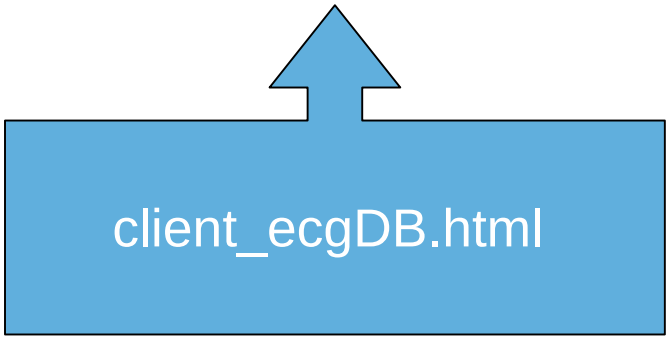
ECG: Node.js – ecg_express.js

```
27  // Web routing address
28  app.get('/', function (req, res) {  // localhost:3030/
29    res.send('Hello Arduino-ECG IOT!');
30  });
31  // find all data & return them
32  app.get('/ecg', function (req, res) {
33      Sensor.find(function(err, data) {
34          res.json(data);
35      });
36  });
37  // find data by id
38  app.get('/ecg/: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_ECG_IOT is running at port:3030");
```

ECG: Node.js – ecg_node_mongodb.js

[DIY]

Real-time WEB client to monitor ECG from MongoDB



ECG: Web client→ client_ecgDB.html – 1

```
<!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 : ECG data</h2>

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

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

ECG: Web client→ client_ecgDB.html – 2

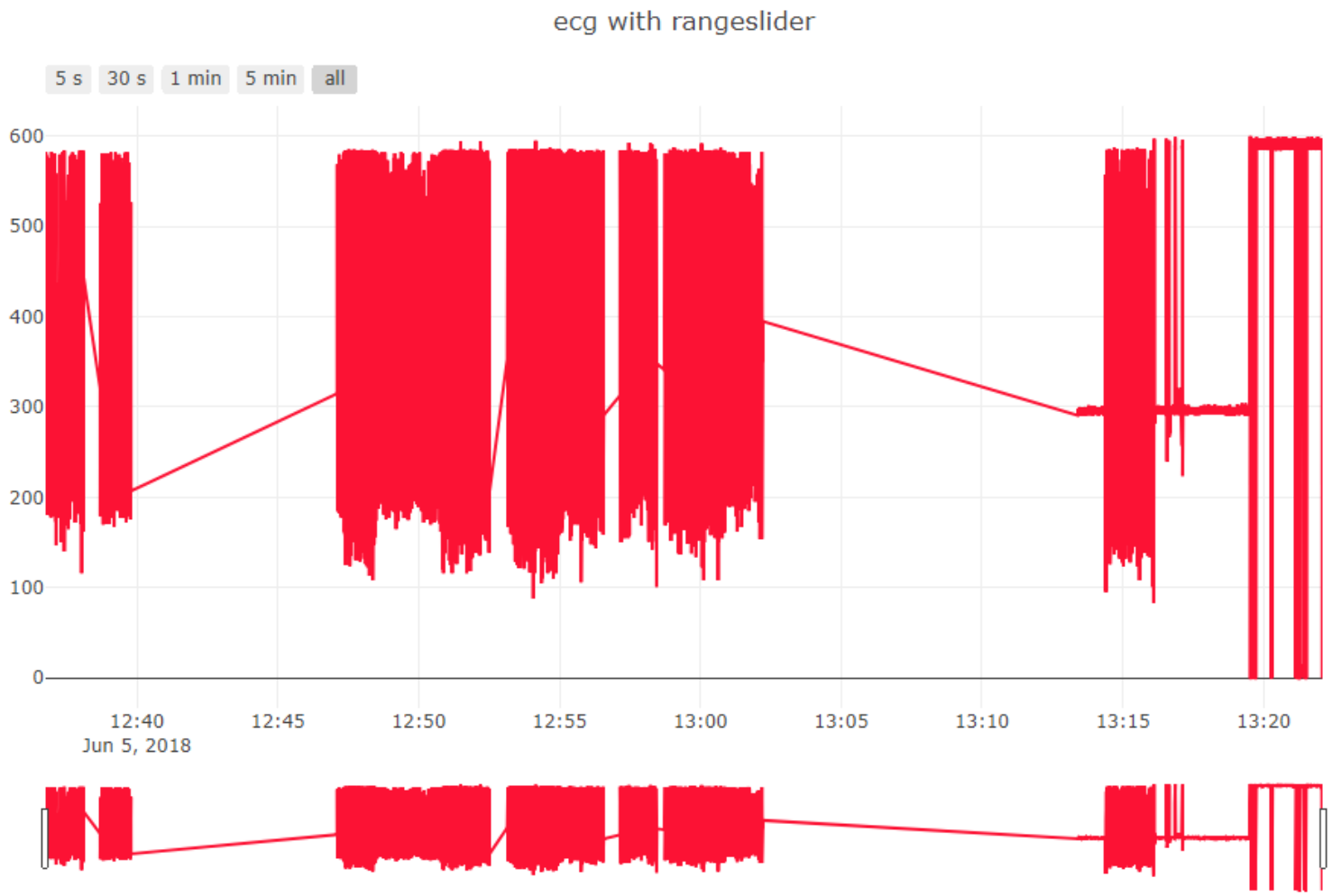
```
Plotly.d3.json("http://localhost:3030/ecg", function(err, json){
    //alert(JSON.stringify(json));  // It works!!!
    //alert(JSON.parse(eval(json));
    if(err) throw err;

    var date = [];
    var ecg = [];
    var jsonData = eval(JSON.stringify(json));

    for (var i = 0; i < jsonData.length; i++) {
        date[i] = jsonData[i].date;
        ecg[i] = jsonData[i].ecg;
    }

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

    var data = [trace1];
```

ECG: Web client→ client_ecgDB.html – 3

```
// Layout with builtin rangeslider
var layout = {
    title: 'ecg with rangeslider',
    xaxis: {
        autorange: true,
        range: [date[0], date[date.length-1]],
        rangeselector: {buttons: [
            {
                count: 5,
                label: '5 s',
                step: 'second',
                stepmode: 'backward'
            },
            {
                count: 30,
                label: '30 s',
                step: 'second',
                stepmode: 'backward'
            },
            {
                count: 1,
                label: '1 min',
                step: 'minute',
                stepmode: 'backward'
            },
            {
                count: 5,
                label: '5 min',
                step: 'minute',
                stepmode: 'backward'
            },
            {step: 'all'}
            ]},
            rangeslider: {range: [date[0], date[date.length-1]]},
            type: 'date'
    },
    yaxis: {
        autorange: true,
        range: [0, 500],
        type: 'linear'
    }
};

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

MongoDB database visualization by HS00

Time series : ECG data



MongoDB database visualization by HS00

Time series : ECG data

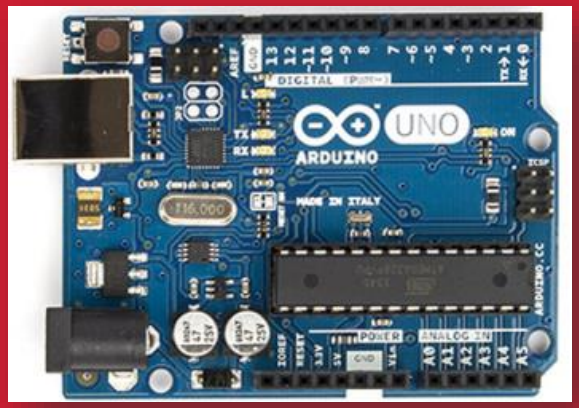
Save as
HSnn_ecg_web.png

Save your ECG

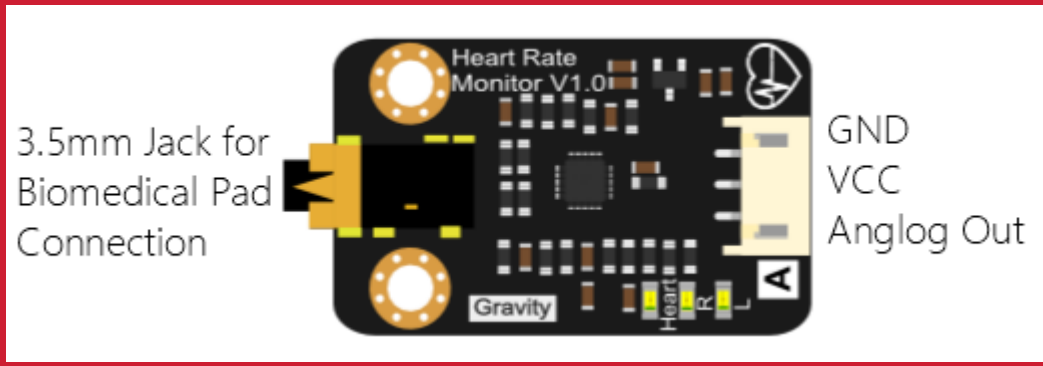
from MongoDB ‘ecg’

Filename :
HSnn_ecg.csv

Arduino & Node.js & MongoDB



HR sensor Module SEN0213



HR: Node.js – hr_node_mongodb.js - 1

hr_node_mongodb.js

```
// hr_node_mongodb.js

var serialport = require('serialport');
var portName = 'COM7';  // check your COM port!!
var port     =   process.env.PORT || 3000;

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

// MongoDB
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
// MongoDB connection
mongoose.connect('mongodb://localhost:27017/hrv'); // DB name
    var db = mongoose.connection;
    db.on('error', console.error.bind(console, 'connection error:'));
    db.once('open', function callback () {
        console.log("mongo db connection OK.");
});
// Schema
var hrSchema = new Schema({
    date : String,
    hr : String
});
// Display data on console in the case of saving data.
hrSchema.methods.info = function () {
    var hrInfo = this.date
    ? "Current date: " + this.date +", HR: " + this.hr
    : "I don't have a date"
    console.log("hrInfo: " + hrInfo);
}
```

HR: Node.js – hr_node_mongodb.js - 2

```
// serial port object
var sp = new serialport(portName,{
    baudRate: 115200,    // 9600  19200  38400
    dataBits: 8,
    parity: 'none',
    stopBits: 1,
    flowControl: false,
    parser: serialport.parsers.readline('\r\n')
});

var readData = '';  // this stores the buffer
var hrv ='';
var mdata =[]; // this array stores date and data
var firstcommaidx = 0;
var Sensor = mongoose.model("Sensor", hrSchema);
```

```
sp.on('data', function (data) { // call back when dat
    readData = data.toString(); // append data to bu
    firstcommaidx = readData.indexOf(',');
    // parsing data into signals
    if (firstcommaidx > 0) {
        hrv = readData.substring(firstcommaidx + 1);
        readData = '';
        dStr = getDateString();
        mdata[0]=dStr;  // Date
        mdata[1]=hrv;  // hr data
        var iot = new Sensor({date:dStr, hr:hrv});
        // save iot data to MongoDB
        iot.save(function(err, iot) {
            if(err) return handleEvent(err);
            iot.info();  // Display the information
        })
        io.sockets.emit('message', mdata);  // send
    } else {  // error
        console.log(readData);
    }
});
```

HR: Node.js – hr_node_mongodb.js - 3

```
// 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;
}

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

HR: Node.js – hr_node_mongodb.js

Run: node hr_node_mongodb.js (Run in Node cmd)

NodeJS - node hr_node_mongodb

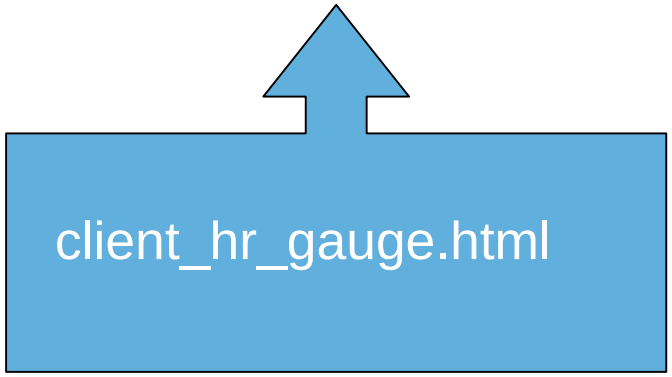
```
D:\Portable\NodeJSPortable\Data\hs00\ecg>node hr_node_mongodb
mongo db connection OK.
hrInfo: Current date: 2018-06-05 15:10:52.937, HR: 141
hrInfo: Current date: 2018-06-05 15:10:53.944, HR: 141
hrInfo: Current date: 2018-06-05 15:10:54.985, HR: 141
hrInfo: Current date: 2018-06-05 15:10:56.090, HR: 141
hrInfo: Current date: 2018-06-05 15:10:57.195, HR: 109
hrInfo: Current date: 2018-06-05 15:10:58.332, HR: 89
hrInfo: Current date: 2018-06-05 15:10:59.435, HR: 92
hrInfo: Current date: 2018-06-05 15:11:01.726, HR: 94
hrInfo: Current date: 2018-06-05 15:11:02.829, HR: 95
hrInfo: Current date: 2018-06-05 15:11:03.950, HR: 98
hrInfo: Current date: 2018-06-05 15:11:05.056, HR: 85
hrInfo: Current date: 2018-06-05 15:11:07.297, HR: 79
hrInfo: Current date: 2018-06-05 15:11:08.385, HR: 86
hrInfo: Current date: 2018-06-05 15:11:18.889, HR: 80
hrInfo: Current date: 2018-06-05 15:11:20.008, HR: 80
```

명령 프롬프트 - mongo

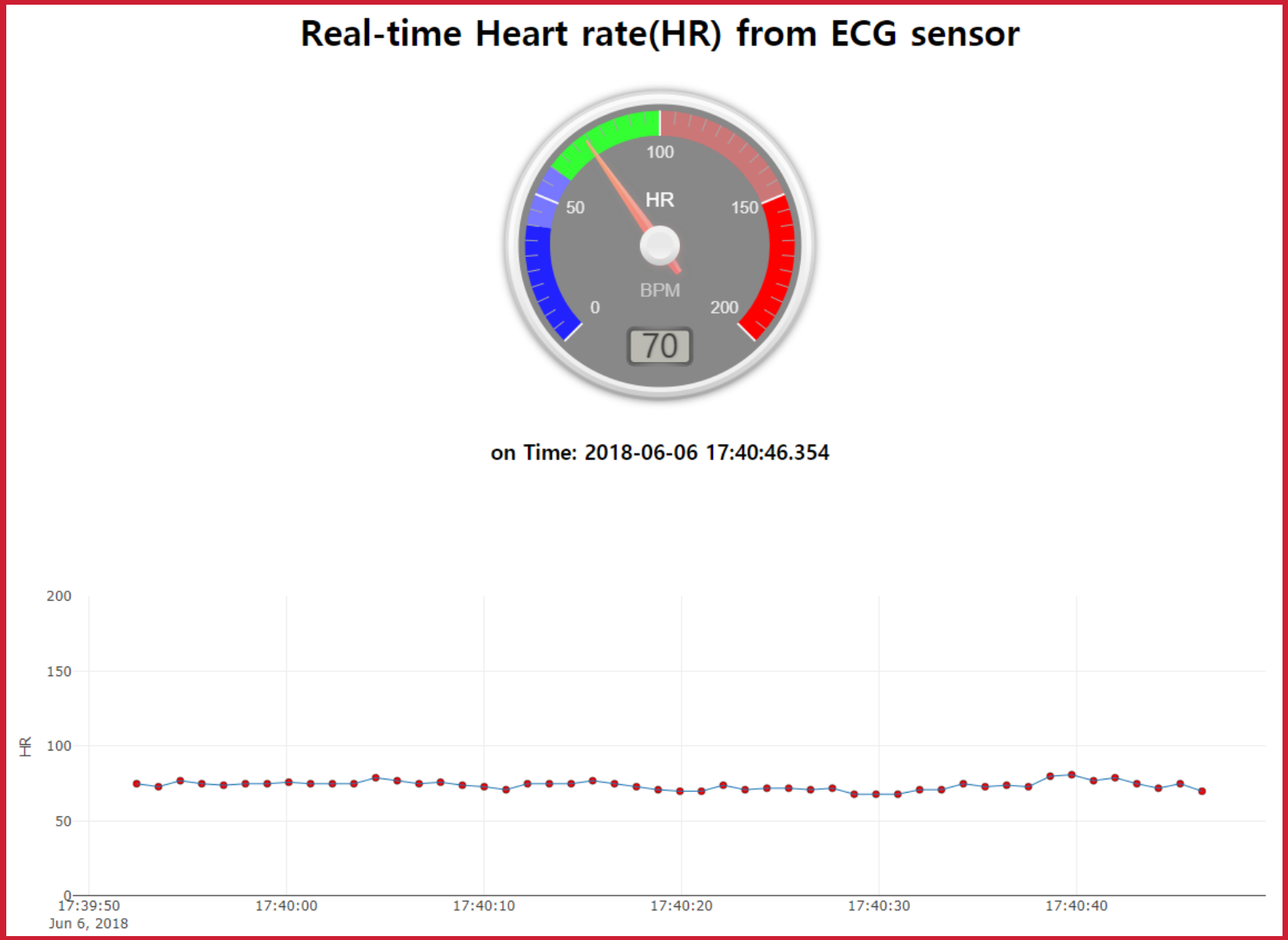
```
> show dbs
aa00    0.000GB
admin   0.000GB
config  0.000GB
ecg     0.004GB
hrv     0.000GB
iot     0.000GB
iot2    0.000GB
iot3    0.001GB
local   0.000GB
test    0.000GB
test2   0.000GB
>
```

[DIY]

Real-time WEB client to monitor HR



HR: Web client→ Real-time streaming



HR: Web client→ client_hr_gauge.html – 1

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <title>plotly.js client: Real time signals from sensors</title>
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/
  socket.io.js"></script>

  <script src="gauge.min.js"></script>

  <style>body{padding:0;margin:30;background:#fff}</style>
</head>

<body>  <!-- style="width:100%;height:100%"> -->
<!-- Plotly chart will be drawn inside this DIV -->
<h1 align="center">Real-time Heart rate(HR) from ECG sensor</h1>
<!-- Lux gauge -->
<div align="center">
    <canvas id="gauge"> </canvas>
</div>

<h3 align="center"> on Time: <span id="time"> </span> </h3>

<div id="myDiv"></div> <!-- graph here! -->

<hr>
```

HR: Web client→ client_hr_gauge.html – 2

```
<script>
/* JAVASCRIPT CODE GOES HERE */
  var streamPlot = document.getElementById('myDiv');
  var ctime = document.getElementById('time');

  var tArray = [], // time of data arrival
      xTrack = [], // value of HR
      numPts = 50, // number of data points
      dtda = [],  // 1 x 2 array : [date, hr] from ECG
      preX = -1,  // check change in data
      initFlag = true;
```

```
var socket = io.connect('http://localhost:3000');
socket.on('connect', function () {
    socket.on('message', function (msg) {
        // initial plot
        if(msg[0]!='' && initFlag){
            dtda[0]=msg[0];
            dtda[1]=parseInt(msg[1]);  // hr
            init();  // start streaming
            initFlag=false;
        }
        console.log(msg[0]);
        console.log(parseInt(msg[1])); // Convert
        dtda[0]=msg[0];
        dtda[1] = parseInt(msg[1]);

        // when new data is coming, keep on stream
        ctime.innerHTML = dtda[0];
        gauge_hr.setValue(dtda[1]); // hr gauge
        //nextPt();
        tArray = tArray.concat(dtda[0]);  // time
        tArray.splice(0,1);
        xTrack = xTrack.concat(dtda[1]);  // hr
        xTrack.splice(0,1);

        var update = {
            x: [tArray],
            y: [xTrack]
        }
        Plotly.update(streamPlot, update);
    });
});
```

HR: Web client→ client_hr_gauge.html – 3

```
function init() {  // initial screen ()
    // starting point : first data (lux)
    for ( i = 0; i < numPts; i++) {
        tArray.push(dtda[0]);  // date
        xTrack.push(dtda[1]);  // hr
    }

    Plotly.plot(streamPlot, data, layout);
}

// data
var data = [{
    x : tArray,
    y : xTrack,
    name : 'Heart rate',
    mode: "lines",  //markers+lines
    line: {
        color: "#1f77b4",
        width: 1
     }//,
    // marker: {
    //     color: "rgb(255, 0, 0)",
    //     size: 6,
    //     line: {
    //       color: "black",
    //       width: 0.5
    //     }
    //  }
}];
```

```
// layout
var layout = {
    xaxis : {
        title : 'time',
        domain : [0, 1]
    },
    yaxis : {
        title : 'HR',
        domain : [0, 1],
        range : [0, 200]
    }
};
```

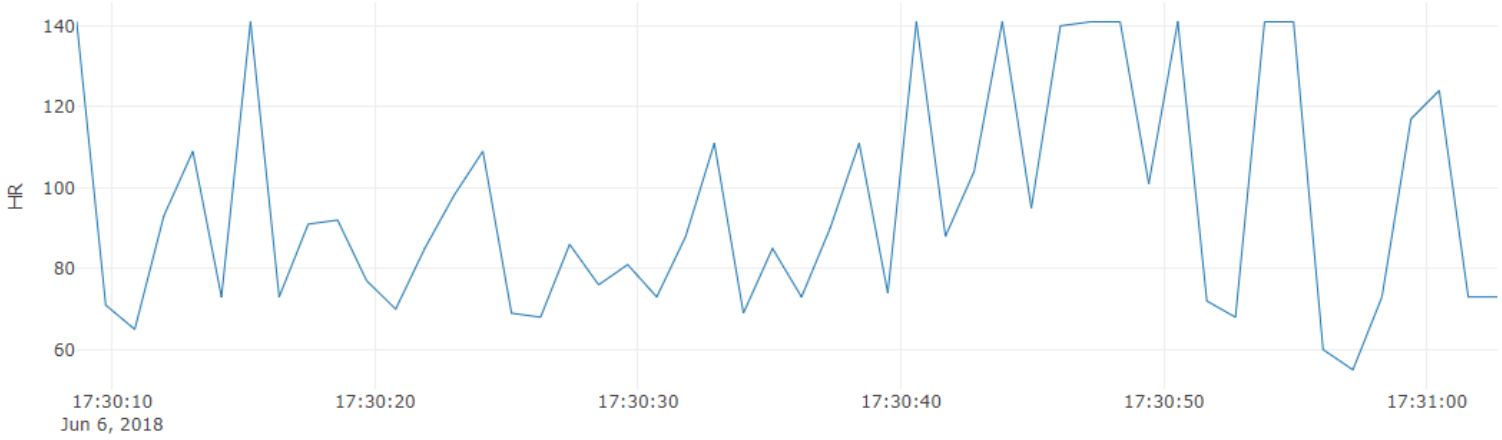
HR: Web client→ client_hr_gauge.html – 4

```
// gauge configuration
var gauge_hr = new Gauge({
    renderTo    : 'gauge',
    width       : 300,
    height      : 300,
    glow        : true,
    units       : 'BPM',
    valueFormat : { int : 2, dec : 0 },
    title       : "HR",
    minValue    : 0,
    maxValue    : 200,  // new
    majorTicks  : ['0','50','100','150','200'],
    minorTicks  : 10,
    strokeTicks : false,
    highlights  : [
        { from : 0,   to : 40, color : '#22f' },
        { from : 40,  to : 60, color : '#77f' },
        { from : 60,  to : 100, color : '#3f3' },
        { from : 100, to : 150, color : '#c77' },
        { from : 150, to : 200, color : '#f00' }
    ],
    colors      : {
        plate       : '#888888',
        majorTicks  : '#f5f5f5',
        minorTicks  : '#aaa',
        title       : '#fff',
        units       : '#ccc',
        numbers     : '#eee',
        needle      : { start : 'rgba(240, 128, 128, 1)',
        end : 'rgba(255, 160, 122, .9)' }
    }
});
gauge_hr.draw();
```

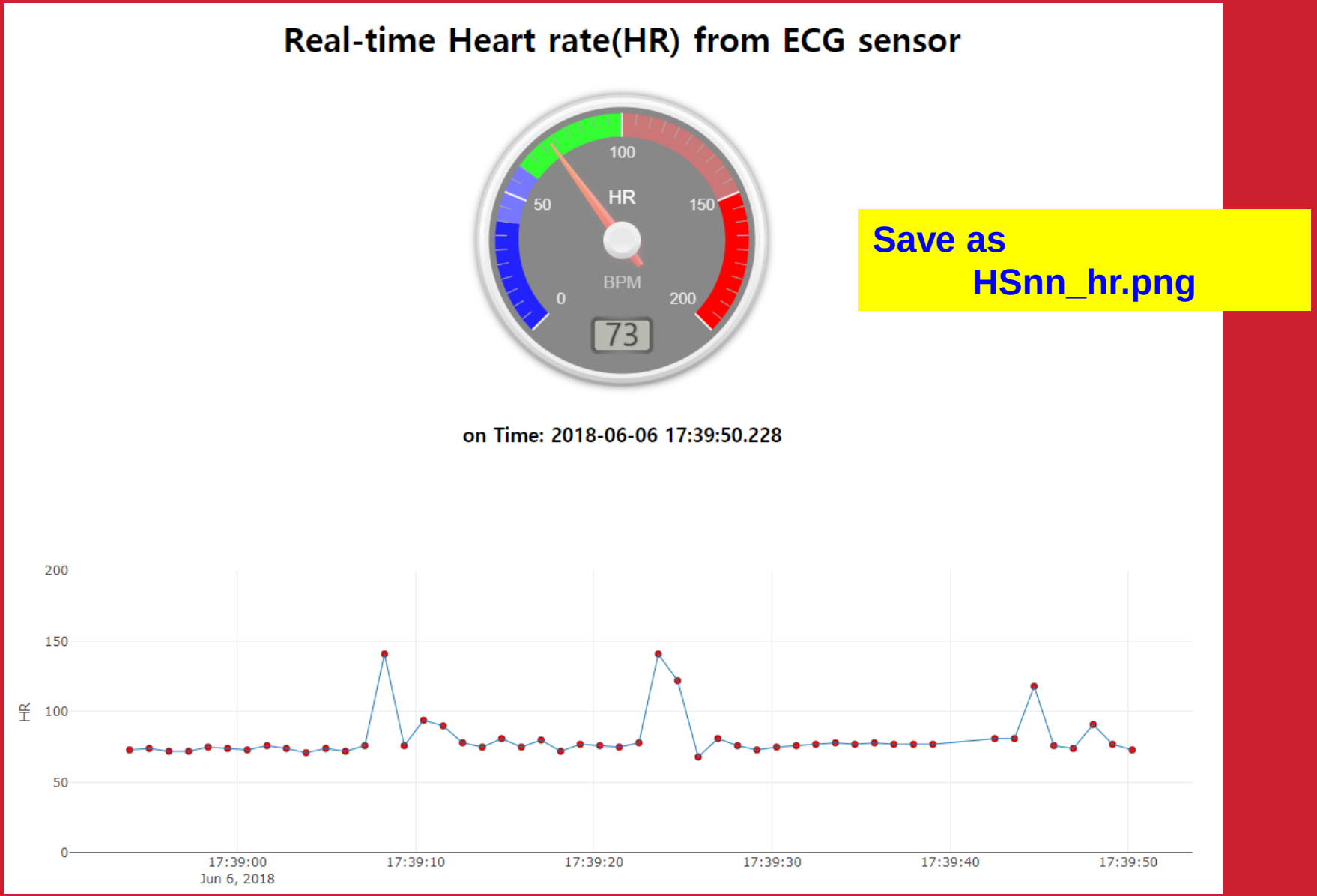
HR: Web client→ Real-time streaming

Real-time Heart rate(HR) from ECG sensor

on Time: 2018-06-06 17:31:02.684



HR: Web client→ Real-time streaming



Save as
HSnn_hr.png

HR: Node.js – hr_express.js - 1

```
// hr_express.js
// Express with CORS
var express = require('express');
var cors = require('cors');  // CORS: Cross Origin Resource Sharing
var app = express();
// CORS
app.use(cors());
var web_port = 3030;  // express port

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

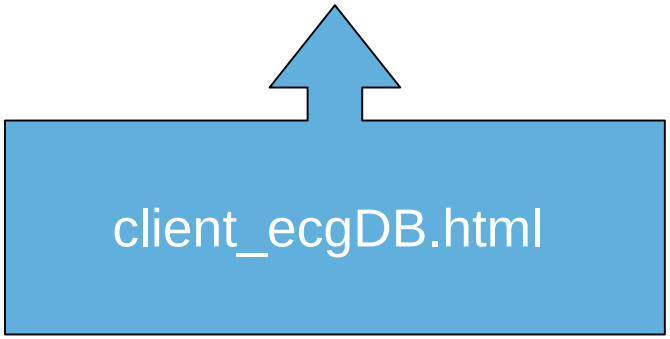
HR: Node.js – hr_express.js – 2

```
// Schema
var hrSchema = new Schema({
    date : String,
    hr : String
});
var Sensor = mongoose.model("Sensor", hrSchema); // sensor data model

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

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

HR: Node.js – hr_node_mongodb.js

[DIY]

Real-time WEB client to monitor HR from MongoDB



MongoDB database visualization by HS00

Time series : HR data

HR: Web client→ client_hrDB.html – 1

```
<!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 : HR data</h2>

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

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

HR: Web client→ client_hrDB.html – 2

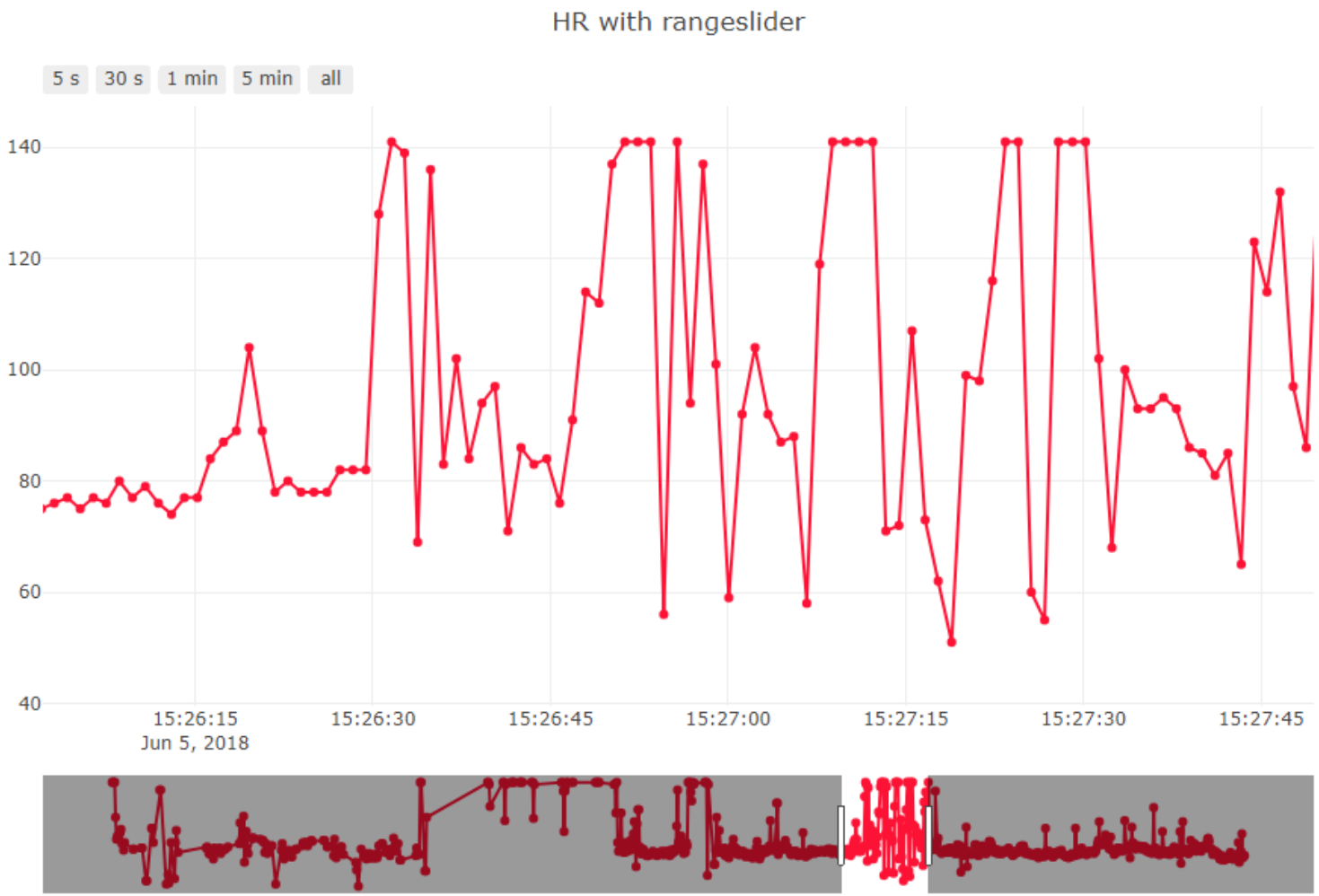
```
Plotly.d3.json("http://localhost:3030/hrv", function(err, json){
    //alert(JSON.stringify(json));  // It works!!!
    //alert(JSON.parse(eval(json));
    if(err) throw err;

    var date = [];
    var hrv = [];
    var jsonData = eval(JSON.stringify(json));

    for (var i = 0; i < jsonData.length; i++) {
        date[i] = jsonData[i].date;
        hrv[i] = jsonData[i].hr;
    }
    // time series of sensor data
    var trace1 = {
        type: "scatter",
        mode: "lines+markers",
        name: 'Heart rate',
        x: date,
        y: hrv,
        line: {color: '#fc1234'}
    }

    var data = [trace1];
```

HR: Web client→ client_hrDB.html – 3

```
// Layout with builtin rangeslider
var layout = {
    title: 'HR with rangeslider',
    xaxis: {
        autorange: true,
        range: [date[0], date[date.length-1]],
        rangeselector: {buttons: [
            {
                count: 5,
                label: '5 s',
                step: 'second',
                stepmode: 'backward'
            },
            {
                count: 30,
                label: '30 s',
                step: 'second',
                stepmode: 'backward'
            },
            {
                count: 1,
                label: '1 min',
                step: 'minute',
                stepmode: 'backward'
            },
            {
                count: 5,
                label: '5 min',
                step: 'minute',
                stepmode: 'backward'
            },
            {step: 'all'}
            ]},
            rangeslider: {range: [date[0], date[date.length-1]]},
            type: 'date'
        },
        yaxis: {
            autorange: true,
            range: [0, 200],
            type: 'linear'
        }
    };

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

MongoDB database visualization by HS00

Time series : HR data



HR data in MongoDB

NodeJS - node hr_node_mongodb

```
hrInfo: Current date: 2018-06-05 15:23:46.916, HR: 89
hrInfo: Current date: 2018-06-05 15:23:48.006, HR: 82
hrInfo: Current date: 2018-06-05 15:23:49.095, HR: 81
hrInfo: Current date: 2018-06-05 15:23:50.200, HR: 76
hrInfo: Current date: 2018-06-05 15:23:51.288, HR: 80
hrInfo: Current date: 2018-06-05 15:23:52.394, HR: 77
hrInfo: Current date: 2018-06-05 15:23:53.483, HR: 75
hrInfo: Current date: 2018-06-05 15:23:54.588, HR: 75
hrInfo: Current date: 2018-06-05 15:23:55.691, HR: 76
hrInfo: Current date: 2018-06-05 15:23:56.780, HR: 78
hrInfo: Current date: 2018-06-05 15:23:57.886, HR: 76
hrInfo: Current date: 2018-06-05 15:23:58.973, HR: 78
hrInfo: Current date: 2018-06-05 15:24:00.079, HR: 77
hrInfo: Current date: 2018-06-05 15:24:01.168, HR: 75
hrInfo: Current date: 2018-06-05 15:24:02.273, HR: 73
hrInfo: Current date: 2018-06-05 15:24:03.377, HR: 75
hrInfo: Current date: 2018-06-05 15:24:04.465, HR: 75
hrInfo: Current date: 2018-06-05 15:24:05.569, HR: 74
hrInfo: Current date: 2018-06-05 15:24:06.674, HR: 76
hrInfo: Current date: 2018-06-05 15:24:07.779, HR: 75
hrInfo: Current date: 2018-06-05 15:24:08.867, HR: 73
hrInfo: Current date: 2018-06-05 15:24:09.972, HR: 72
hrInfo: Current date: 2018-06-05 15:24:11.077, HR: 73
hrInfo: Current date: 2018-06-05 15:24:12.182, HR: 72
hrInfo: Current date: 2018-06-05 15:24:13.272, HR: 76
hrInfo: Current date: 2018-06-05 15:24:14.374, HR: 76
hrInfo: Current date: 2018-06-05 15:24:15.479, HR: 81
hrInfo: Current date: 2018-06-05 15:24:16.570, HR: 87
hrInfo: Current date: 2018-06-05 15:24:17.657, HR: 81
hrInfo: Current date: 2018-06-05 15:24:18.747, HR: 74
hrInfo: Current date: 2018-06-05 15:24:19.852, HR: 71
hrInfo: Current date: 2018-06-05 15:24:20.956, HR: 74
hrInfo: Current date: 2018-06-05 15:24:22.060, HR: 75
hrInfo: Current date: 2018-06-05 15:24:23.148, HR: 78
hrInfo: Current date: 2018-06-05 15:24:24.254, HR: 78
hrInfo: Current date: 2018-06-05 15:24:25.342, HR: 80
hrInfo: Current date: 2018-06-05 15:24:26.446, HR: 78
hrInfo: Current date: 2018-06-05 15:24:27.536, HR: 77
hrInfo: Current date: 2018-06-05 15:24:28.639, HR: 80
```

명령 프롬프트 - mongo

```
}
> db.sensors.find().pretty().limit(7)
{
        "_id" : ObjectId("5b16296cb3dea26bb80a926c"),
        "date" : "2018-06-05 15:10:52.937",
        "hr" : "141",
        "__v" : 0
}
{
        "_id" : ObjectId("5b16296db3dea26bb80a926d"),
        "date" : "2018-06-05 15:10:53.944",
        "hr" : "141",
        "__v" : 0
}
{
        "_id" : ObjectId("5b16296eb3dea26bb80a926e"),
        "date" : "2018-06-05 15:10:54.985",
        "hr" : "141",
        "__v" : 0
}
{
        "_id" : ObjectId("5b162970b3dea26bb80a926f"),
        "date" : "2018-06-05 15:10:56.090",
        "hr" : "141",
        "__v" : 0
}
{
        "_id" : ObjectId("5b162971b3dea26bb80a9270"),
        "date" : "2018-06-05 15:10:57.195",
        "hr" : "109",
        "__v" : 0
}
{
        "_id" : ObjectId("5b162972b3dea26bb80a9271"),
        "date" : "2018-06-05 15:10:58.332",
        "hr" : "89",
        "__v" : 0
}
{
        "_id" : ObjectId("5b162973b3dea26bb80a9272"),
        "date" : "2018-06-05 15:10:59.435",
        "hr" : "92",
        "__v" : 0
}
>
```

Save your HR

from MongoDB ‘hrv’

Filename :
HSnn_hr.csv

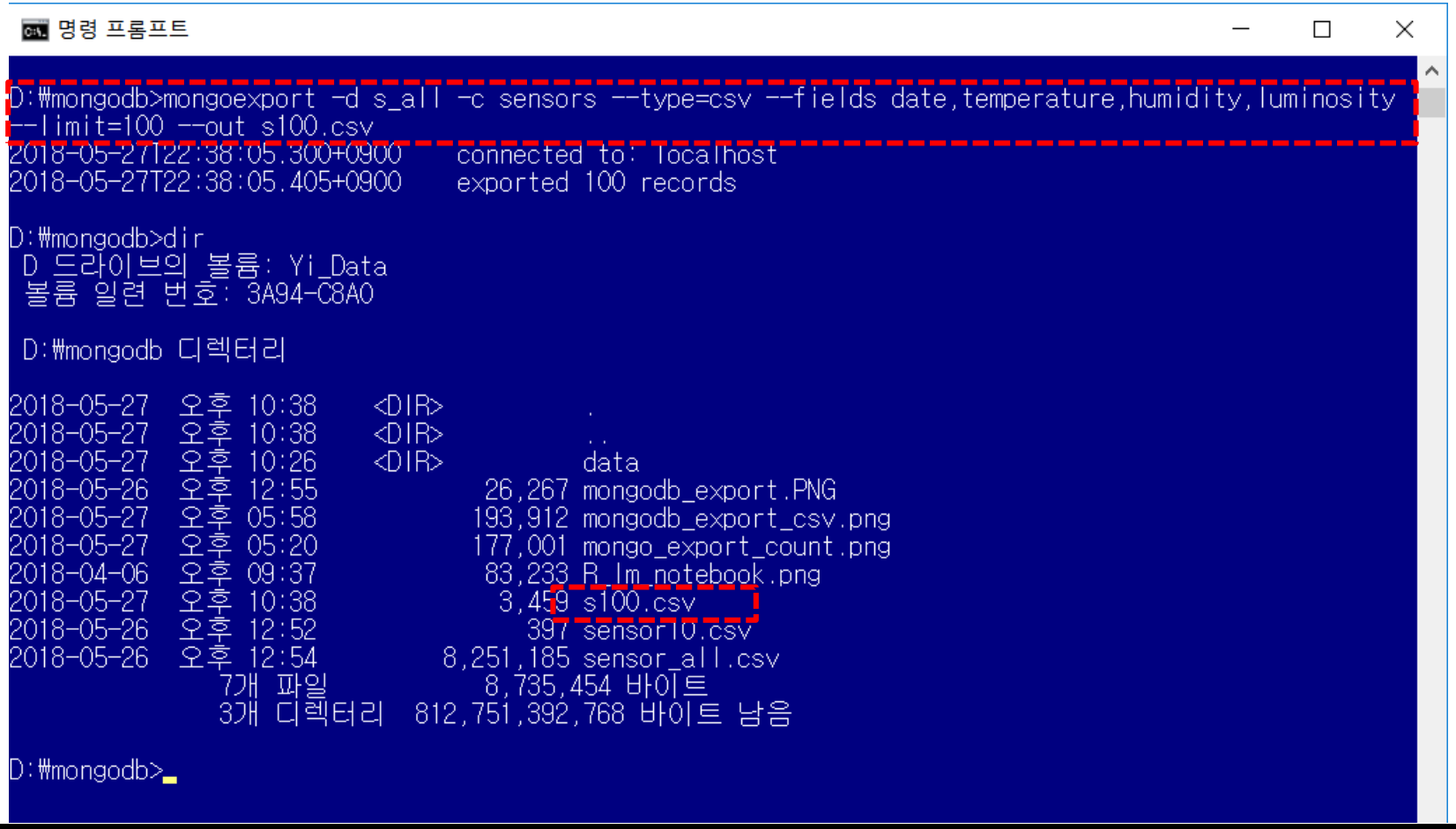
MongoDB data management

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

A5.9.8 MongoDB management

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

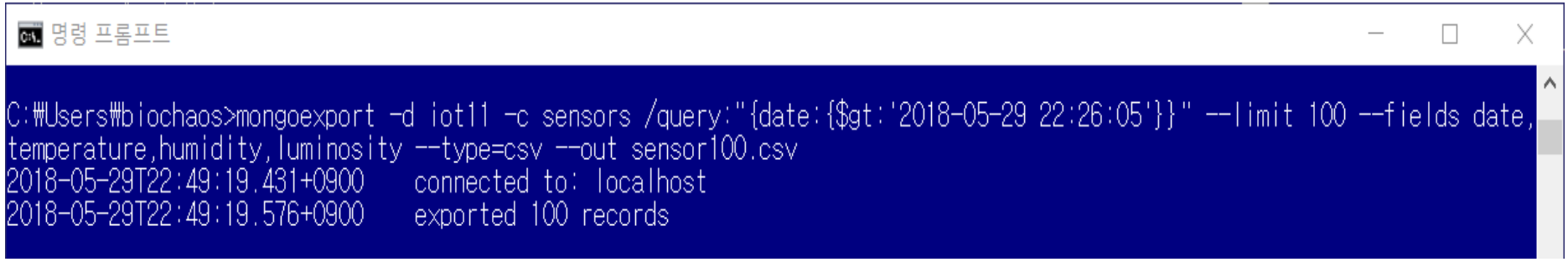
- **mongoexport -d s_all -c sensors --type=csv --fields date,temperature,humidity,luminosity --limit=100 --out s100.csv**



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

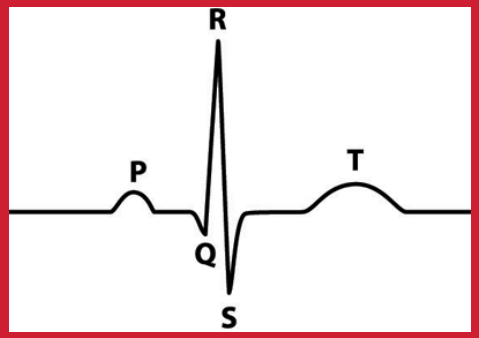
- **mongoexport -d iot11 -c sensors /query:"{date: {$gt: '2018-05-29 22:26:06'}}" --limit=100 --fields date,temperature,humidity,luminosity --type=csv --out s100.csv**

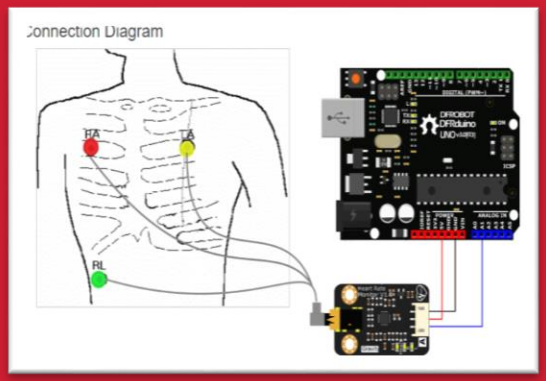


[Tip] iot db의 최근 데이터 500개를 csv 파일 (s500.csv)로 저장할 때,

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

[Practice]





- [wk14]
  - ECG Data management with MongoDB
  - ECG-HR monitoting
  - Complete your project
  - Upload file name : HSnn_Prj2.zip

[wk14] Project-2 HSnn_Prj2.zip

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

제출파일명 : **HSnn_Prj2.zip**

- 압축할 파일들

① HSnn_ecg_web.png

② HSnn_ecg.csv

③ HSnn_hr.png

④ HSnn_hr.csv

Email : chaos21c@gmail.com

[제목 : id, 이름 (수정)]

기말고사
Arduino & Node & MongoDB

[1] 과목명 : 헬스케어신호처리개론 (헬스케어IT 학과 2학년)
[2] 6월 14일(목), 5교시: 오후 1시~2시 (장소 : F1002)
[3] 필기 시험 (20점)
- Arduino, NodeJS, MongoDB Coding에서 20문제
  (선다형/단답형/서술형)
- 수업 시간에 배운 Code의 이해와 활용 능력 평가
- github > src > core_code_final 폴더를 참조
  (cds_dht222, ecg_hr, MongoDB)

**** 잘 준비해서 웃기를 ... ^_^

Lecture materials

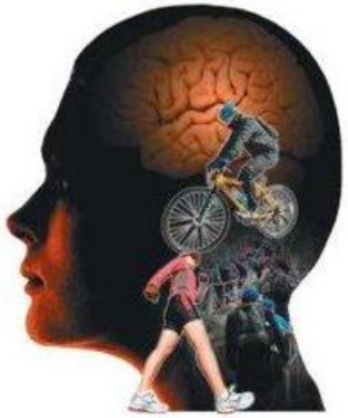
References & good sites

- ✓ 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

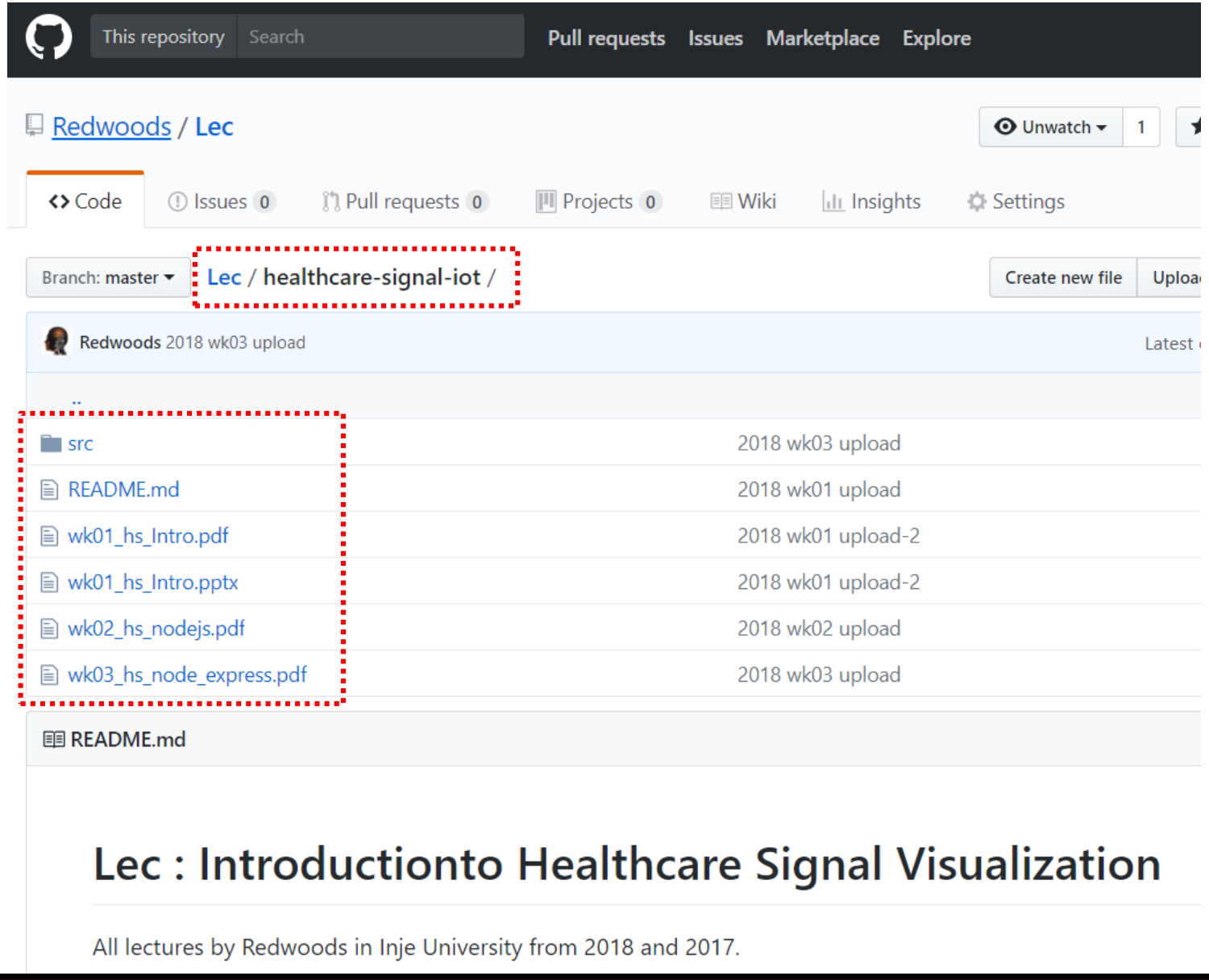
81 commits | 1 branch | 0 releases

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

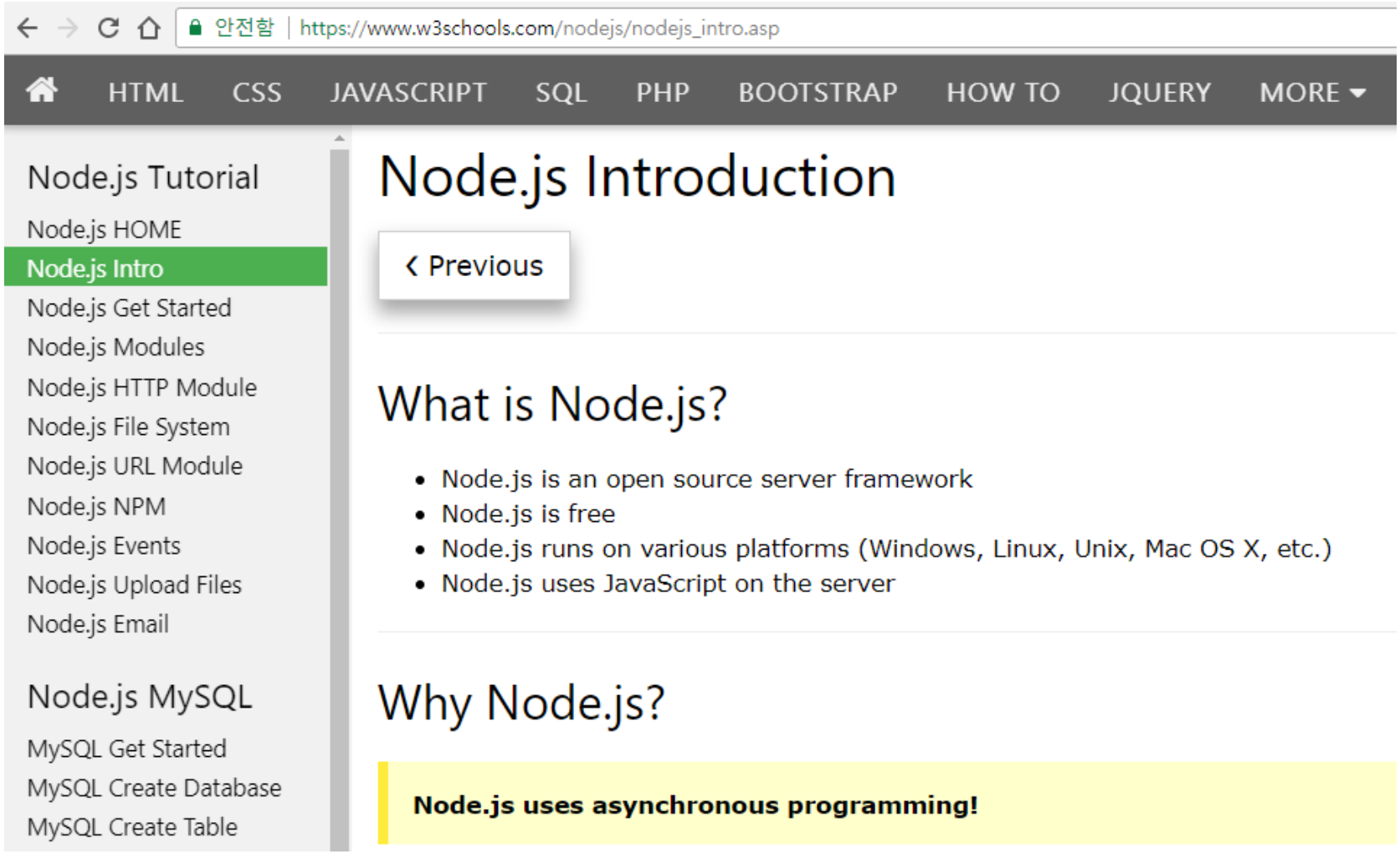
Redwoods 2018 wk01 upload

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

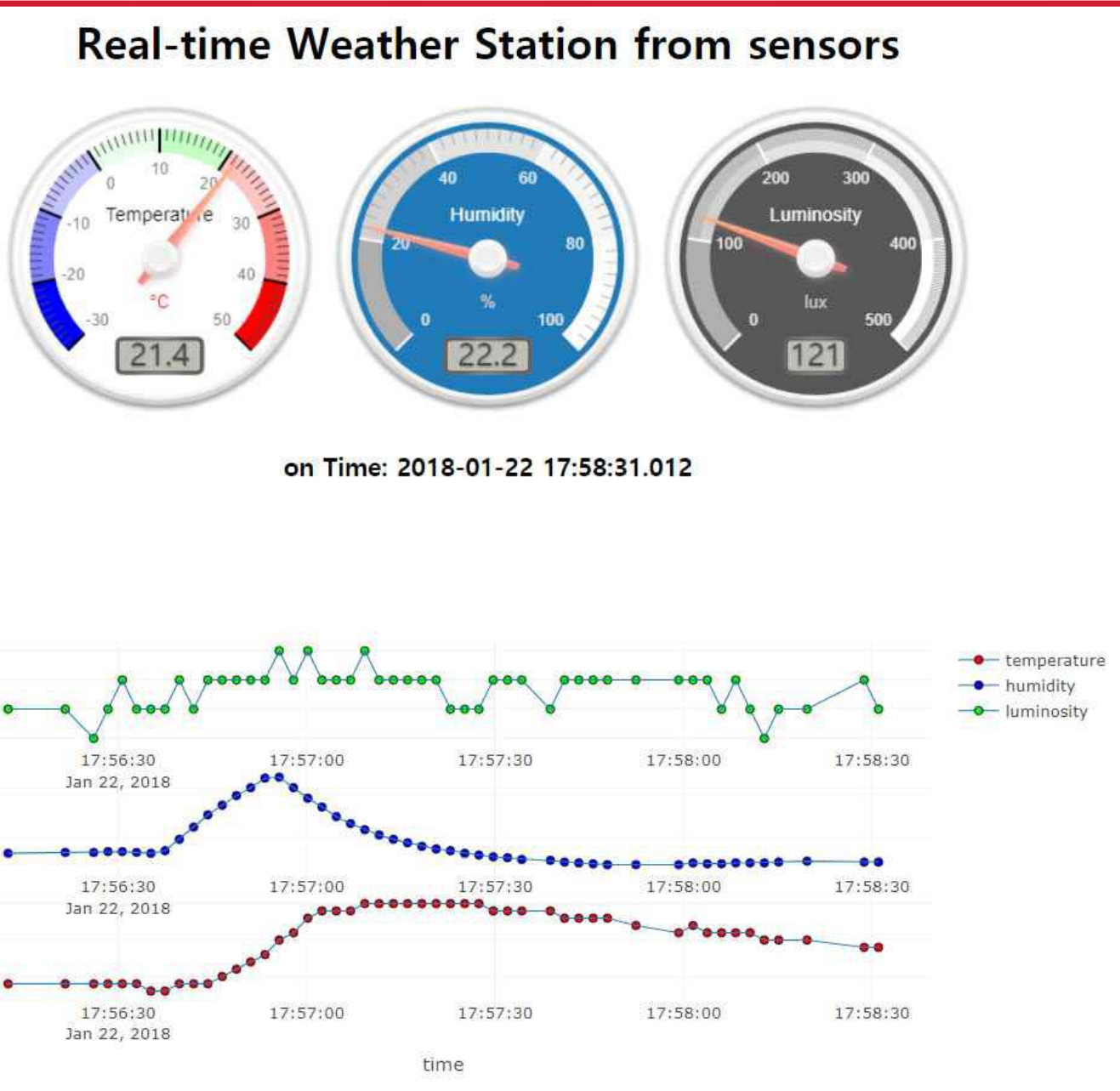


1.0 What is node.js?

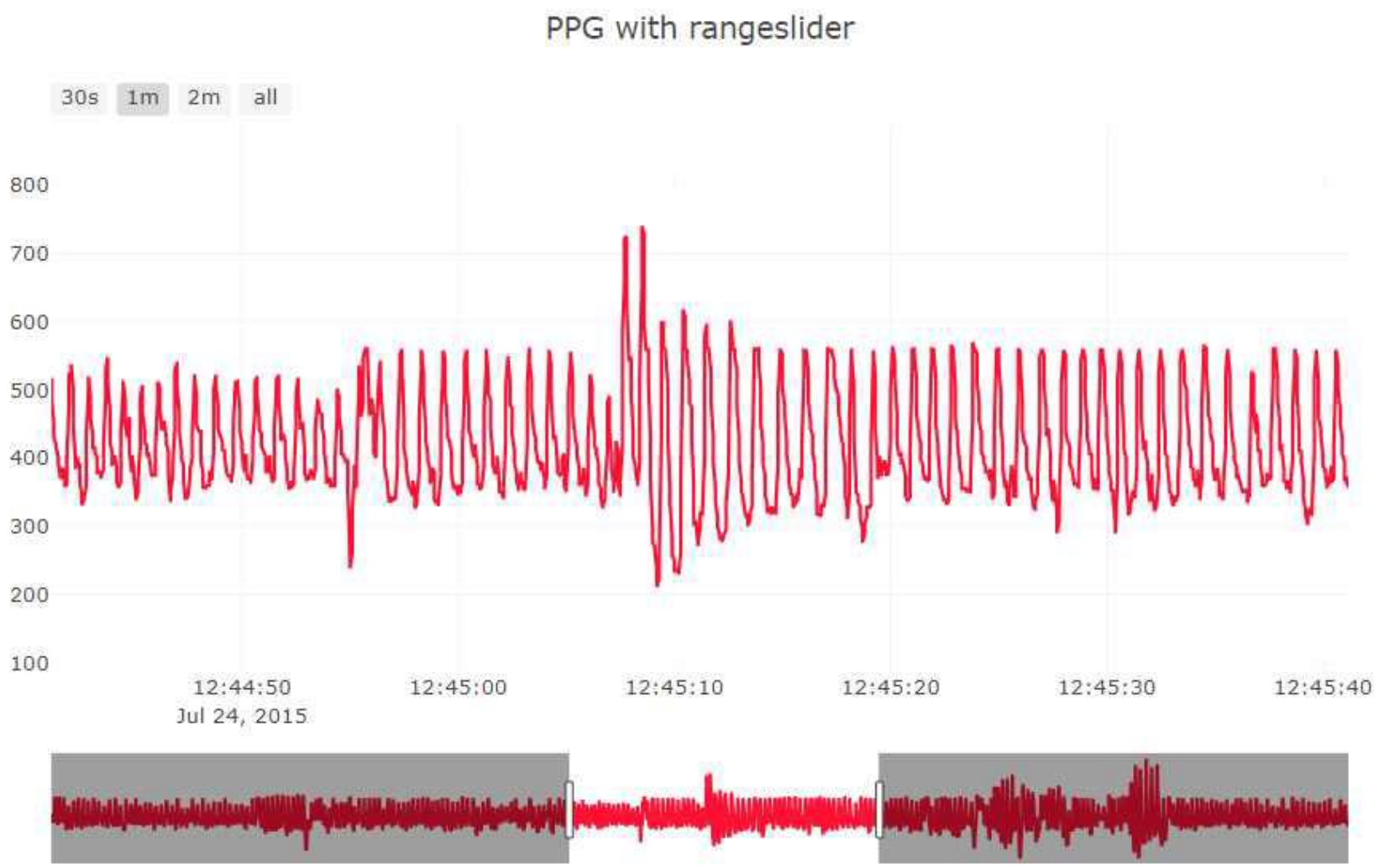


https://www.w3schools.com/nodejs/nodejs_intro.asp

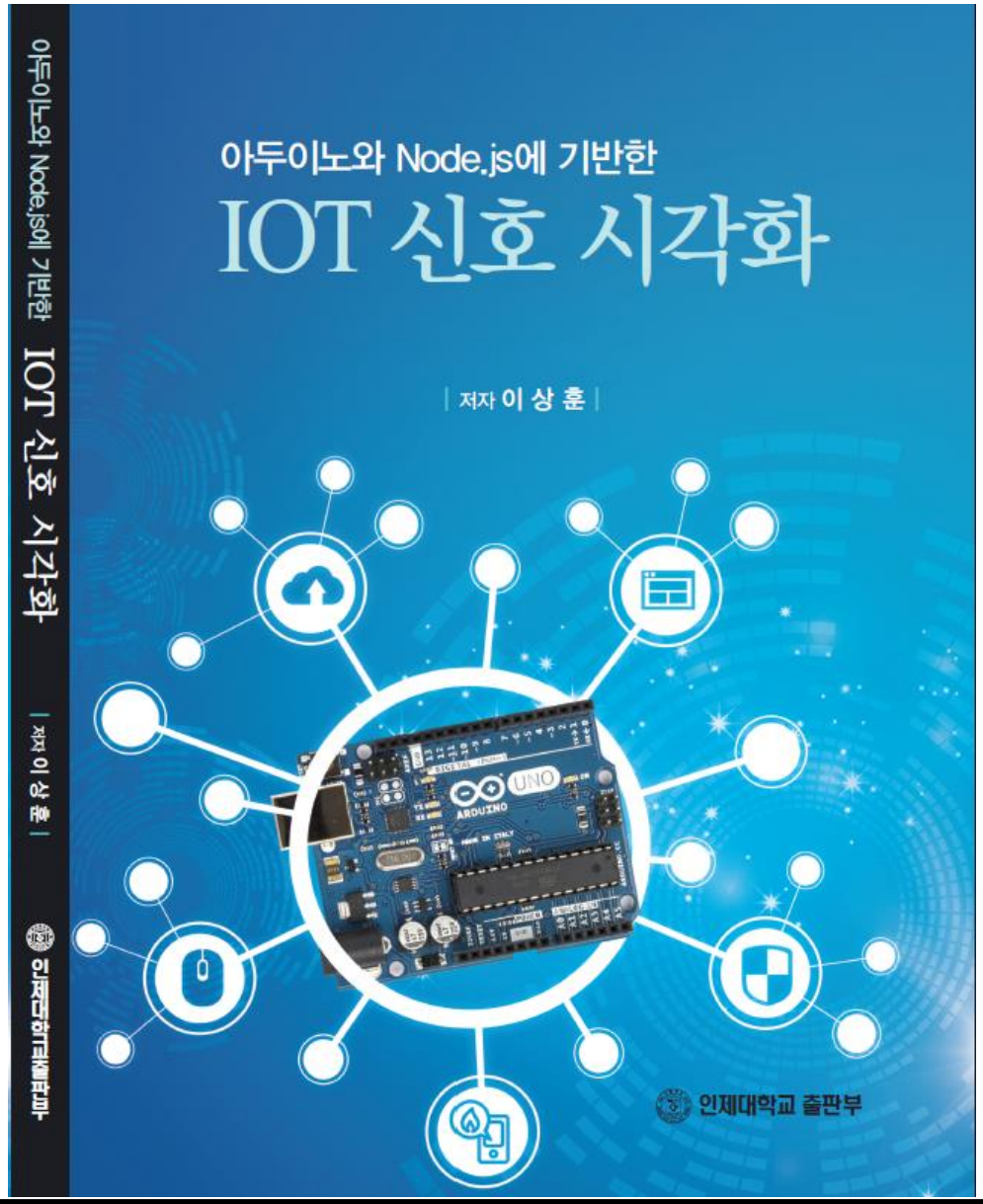
Target of this class



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 |