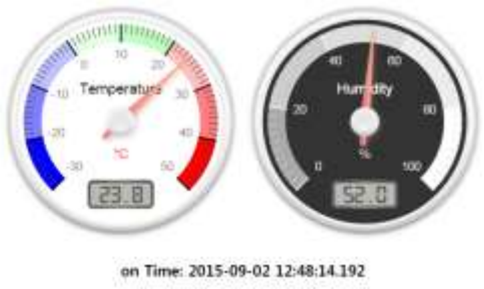
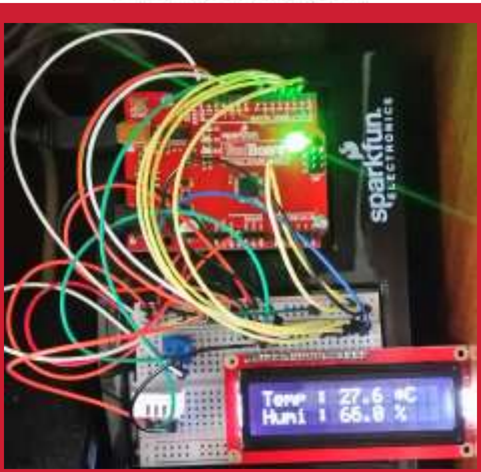
Arduino-IOT

[wk12]

Arduino + Node

Data visualization III





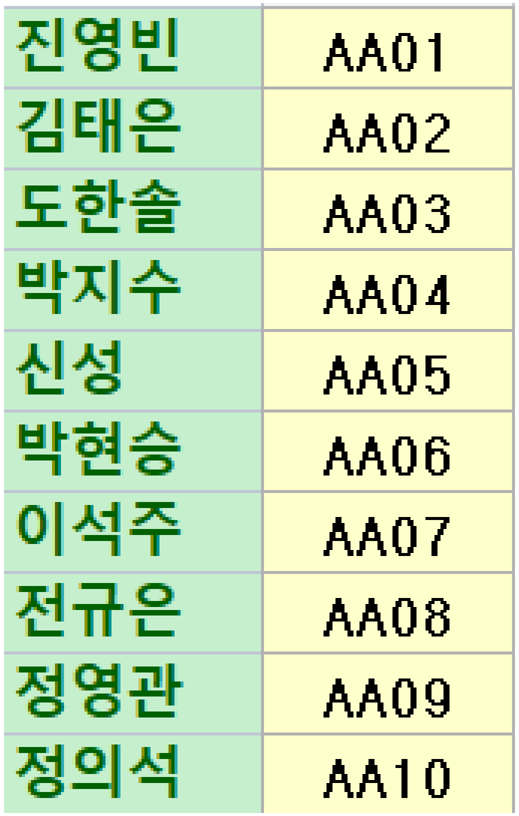
Visualization of Signals using Arduino,
Node.js & storing signals in MongoDB

Comsi, INJE University

2nd semester, 2018

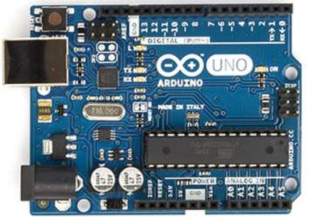
Email : chaos21c@gmail.com

My ID

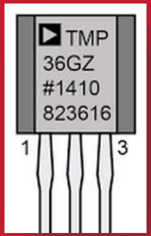


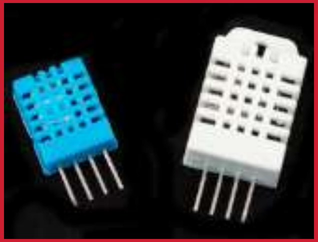
진영빈	AA01
김태은	AA02
도한솔	AA03
박지수	AA04
신성	AA05
박현승	AA06
이석주	AA07
전규은	AA08
정영관	AA09
정의석	AA10

이근재 **AA11**



[Review]





◆ [wk11]

- ➢ RT Data Visualization with node.js
- ➢ Usage of gauge.js
- ➢ Complete your real-time WEB charts
- ➢ Upload file name : AAnn_Rpt08.zip

[wk11] Practice-08 AAnn_Rpt08.zip

◆ **[Target of this week]**

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

제출파일명 : **AAnn_Rpt08.zip**

- 압축할 파일들

① **AAnn_DS_30timestamps.png**

② **AAnn_DS_multiple_axis.png**

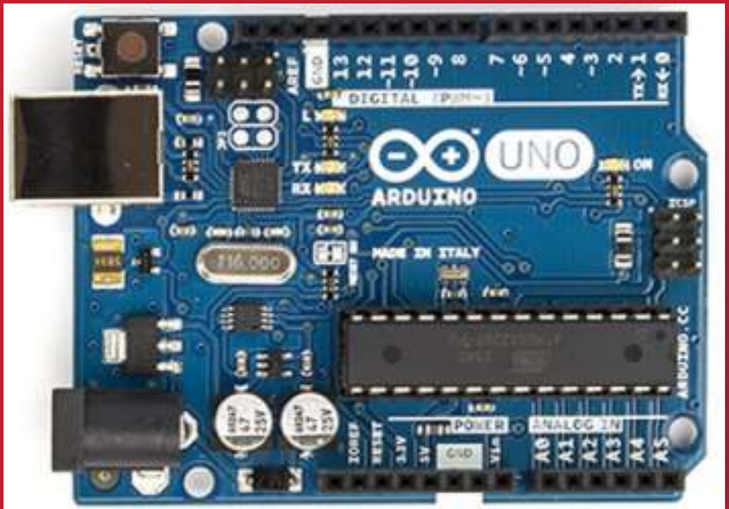
③ **AAnn_cds_gauge.png**

④ **AAnn_cds_change.png**

⑤ AAnn_DS_cds_tmp36.png

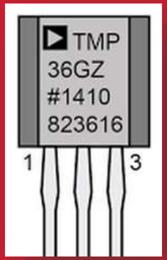
Email : chaos21c@gmail.com

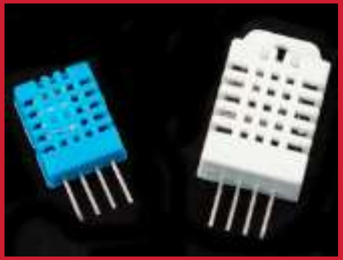
[제목 **:** **id,** 이름 **(**수정**)]**



Arduino

& Node.js





IOT: HSC

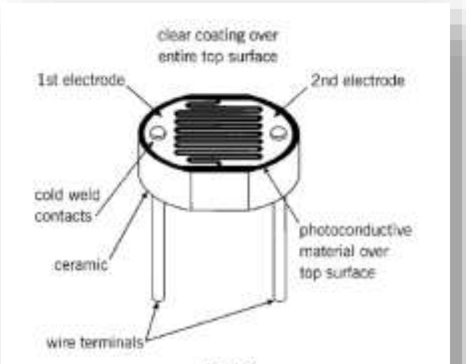
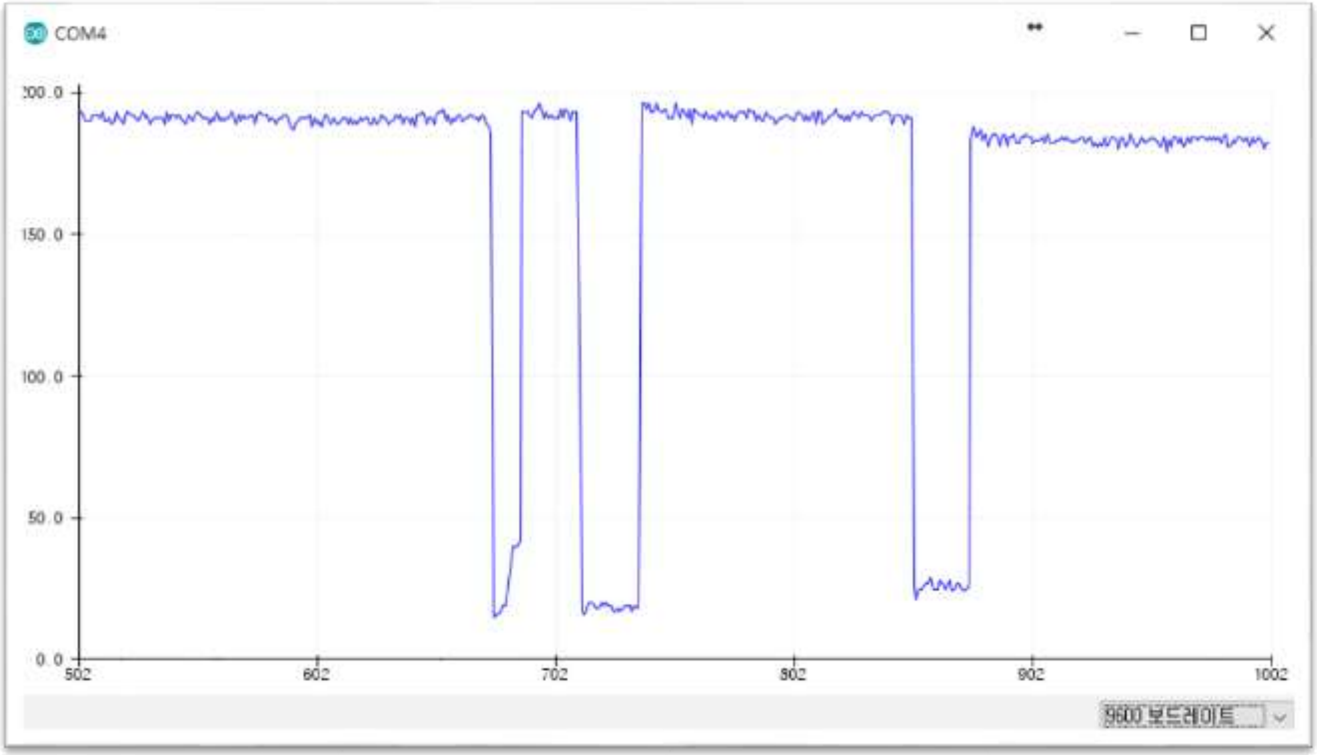
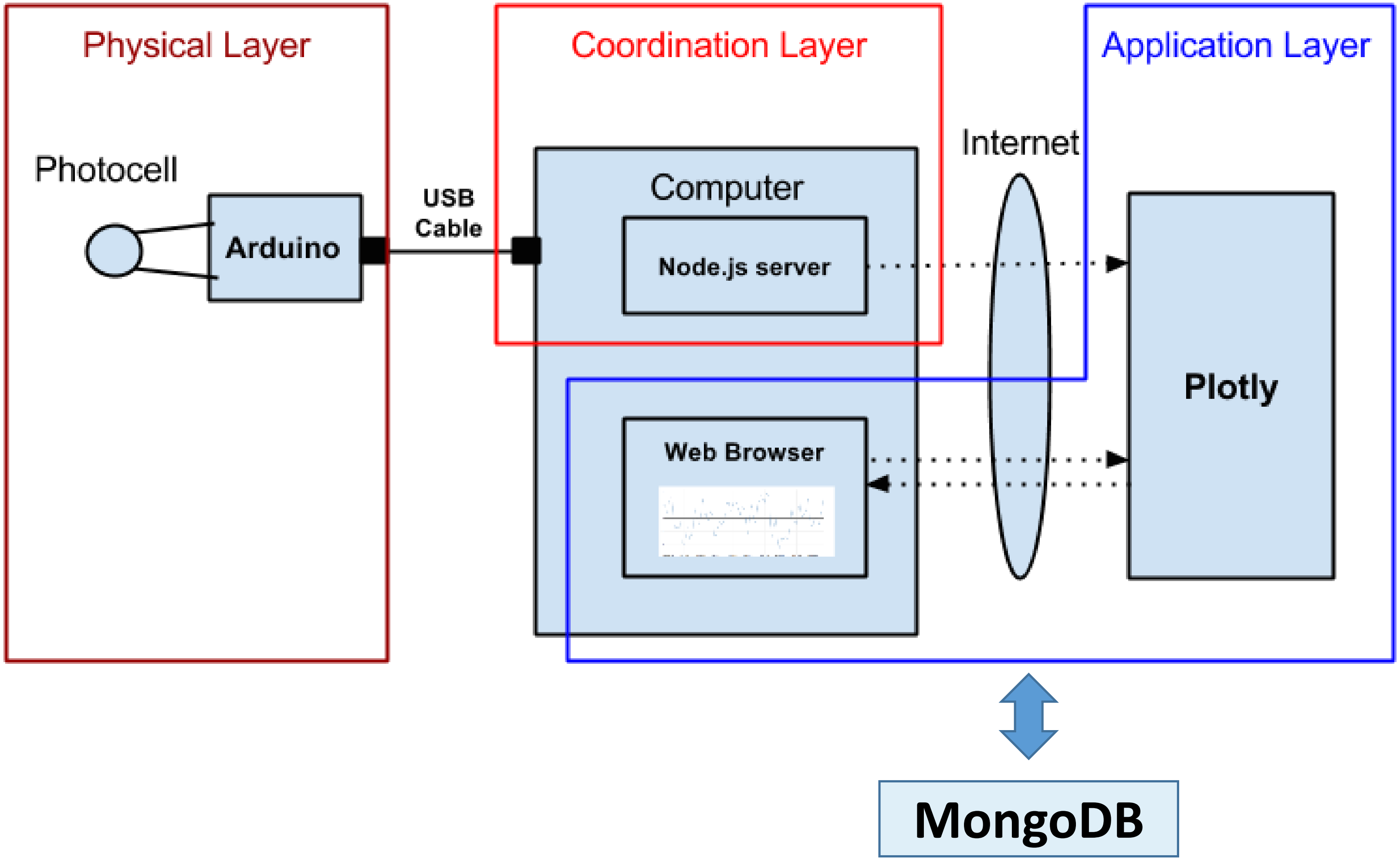


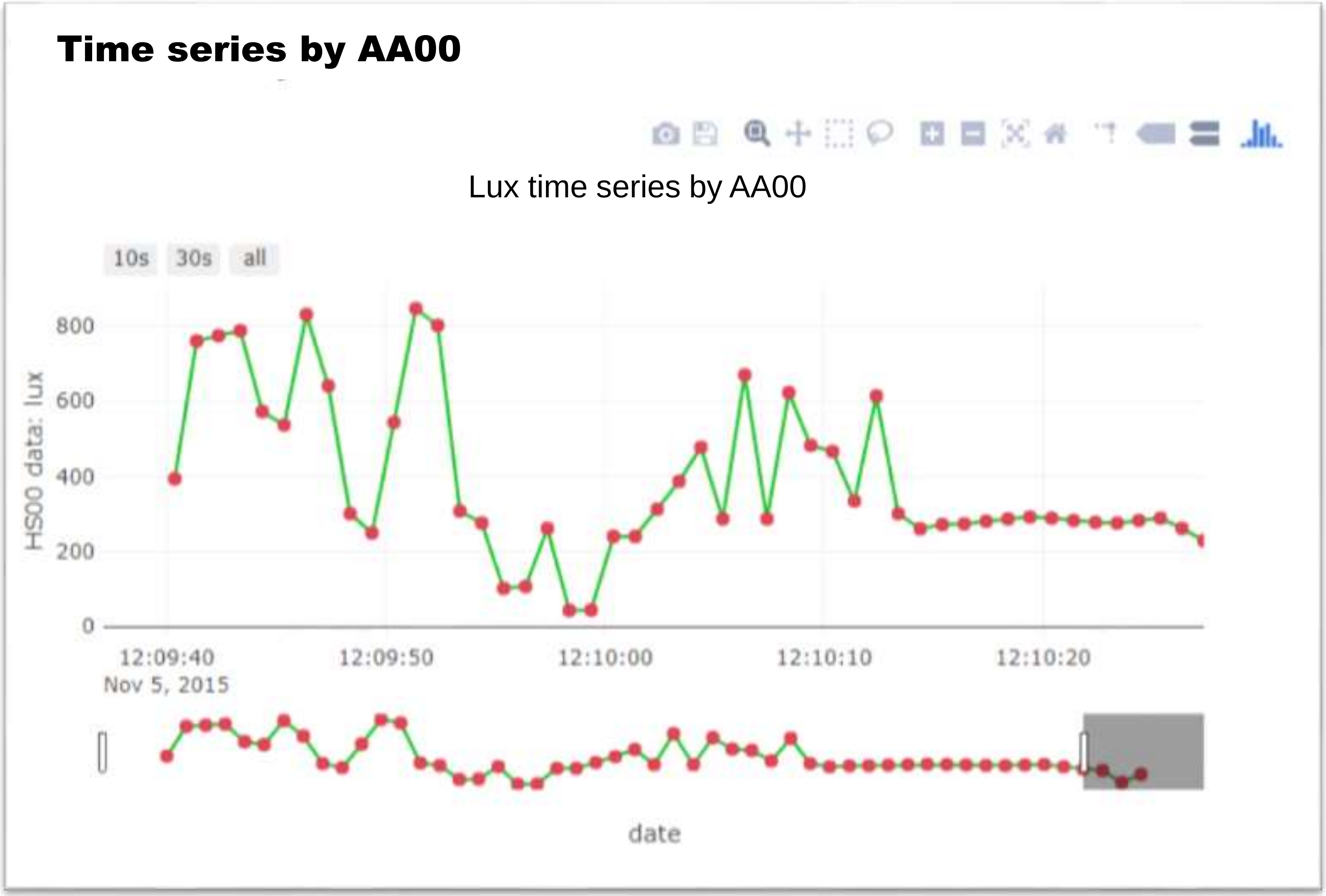
Figure 3
Typical Construction of a Plastic Coated Photocell

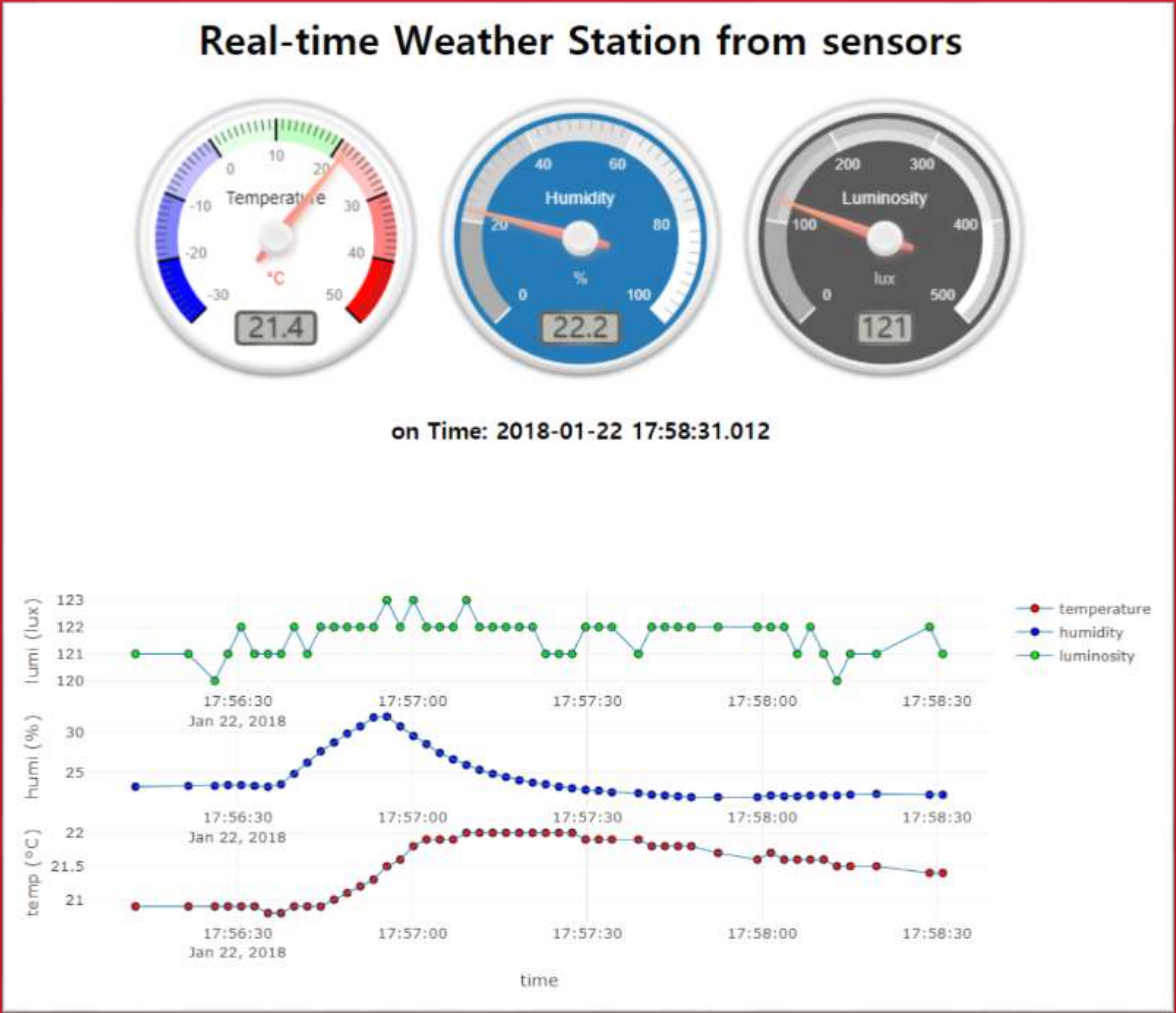


Layout [H S C]


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

Arduino data + plotly

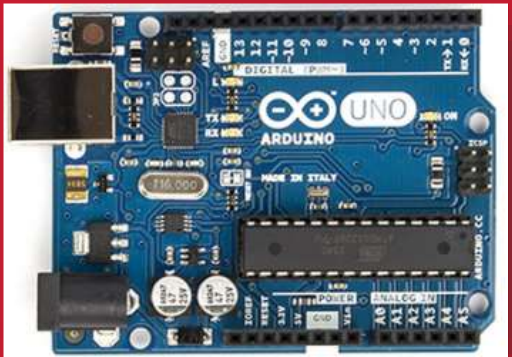


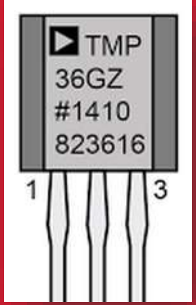


Real-time Weather Station from sensors

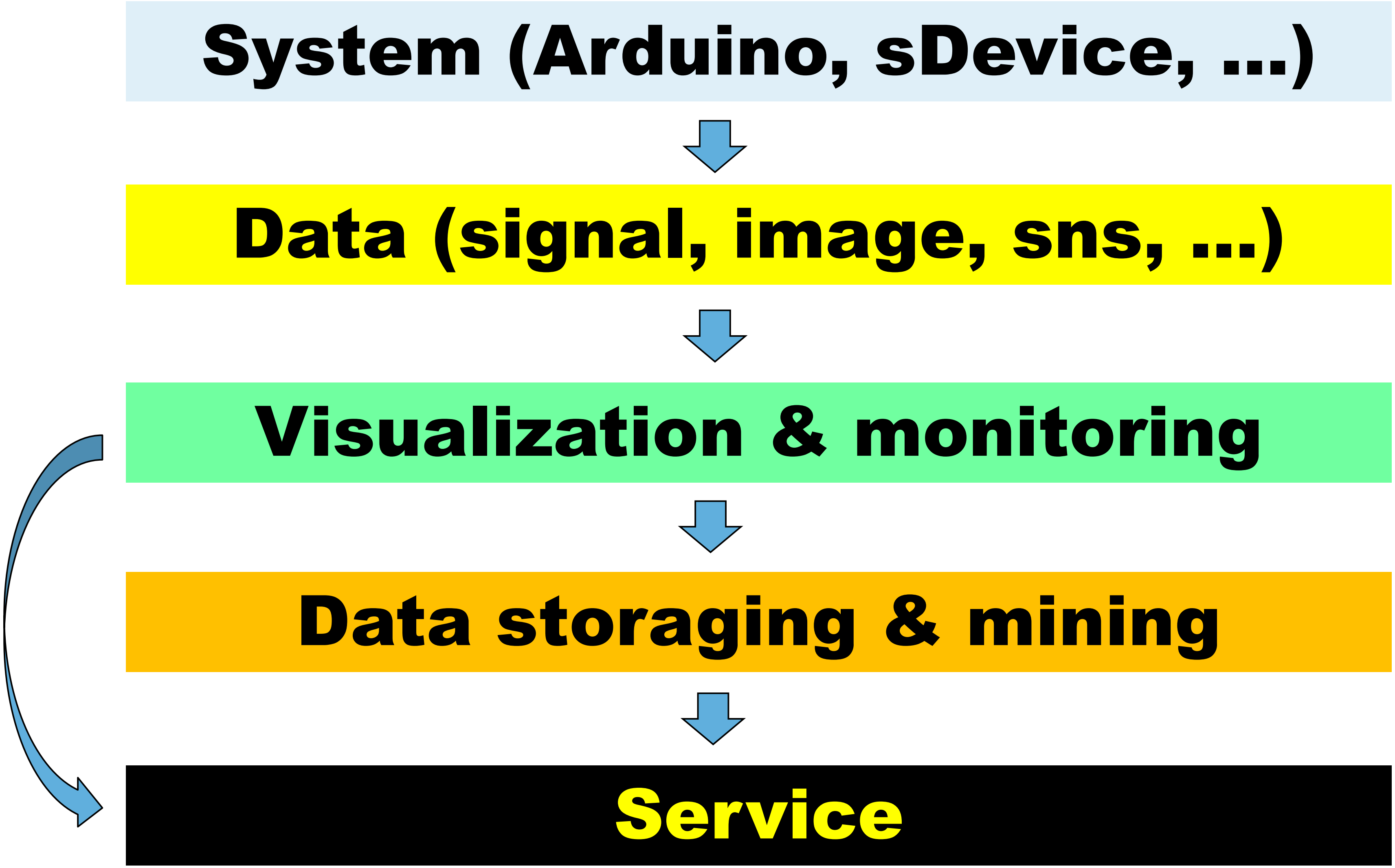
on Time: 2018-01-22 17:58:31.012

Data visualization using ploy.ly





A5. Introduction to visualization



System (Arduino, sDevice, ...)

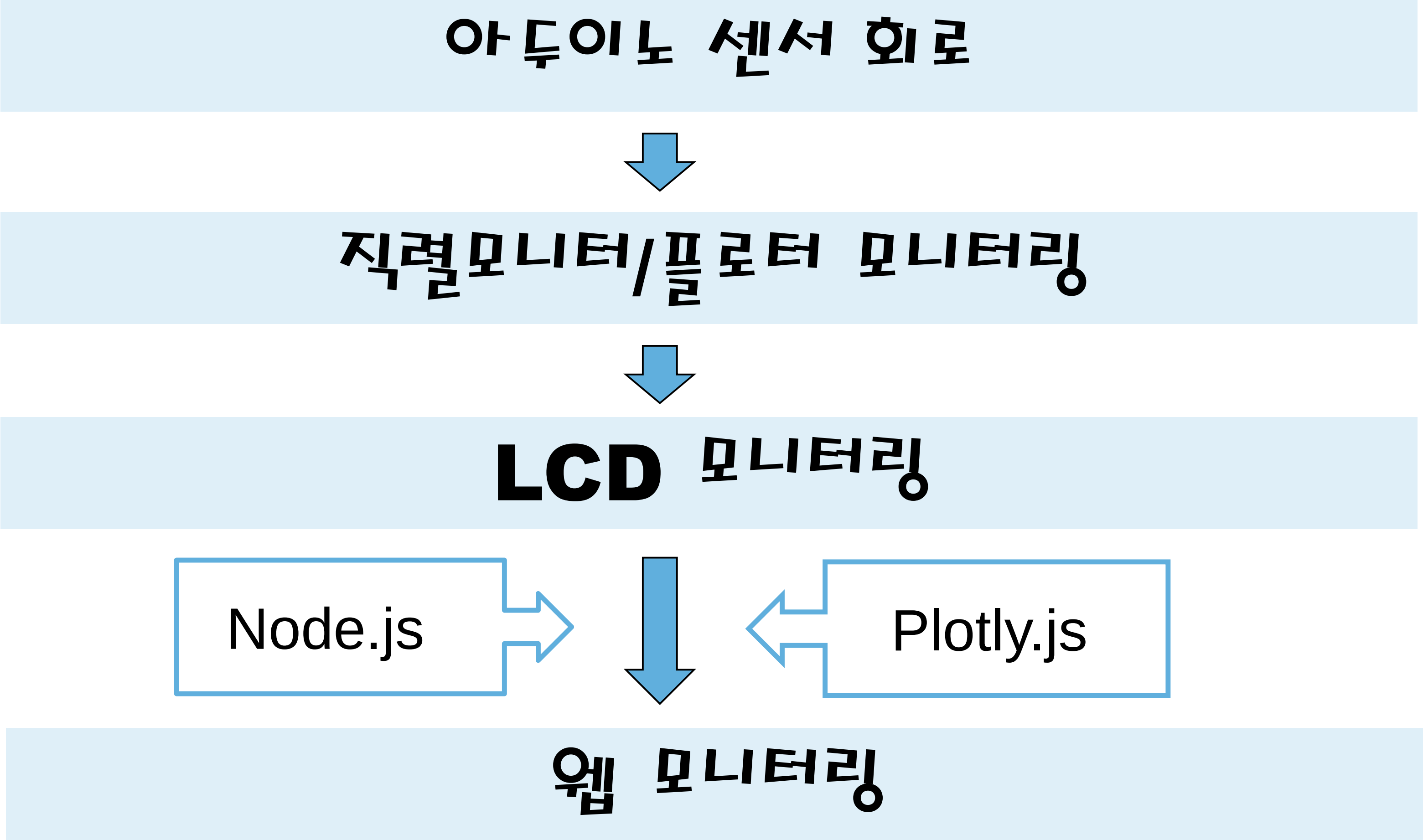
Data (signal, image, sns, ...)

Visualization & monitoring

Data storaging & mining

Service

A5.1 Introduction to data visualization



아두이노 센서 회로

직렬모니터/플로터 모니터링

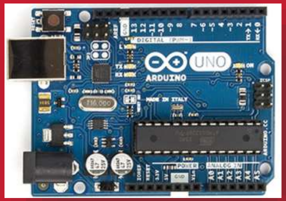
LCD 모니터링

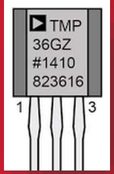
Node.js

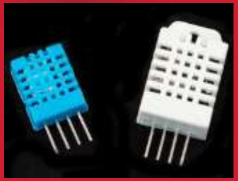
Plotly.js

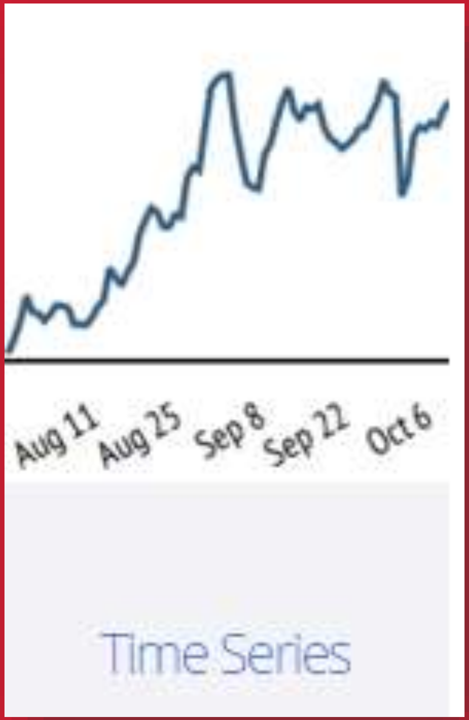
웹 모니터링

Data visualization using plotly.js

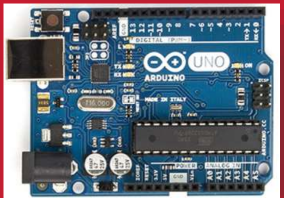


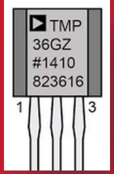


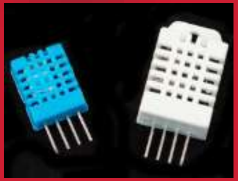


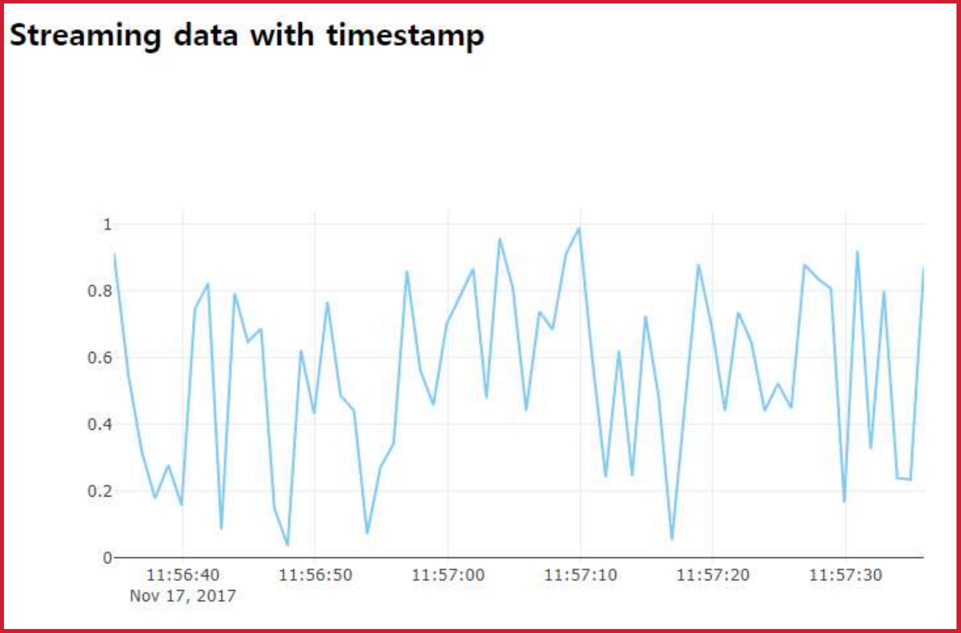


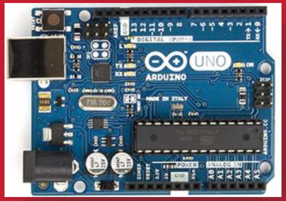
Data Streaming using plotly.js







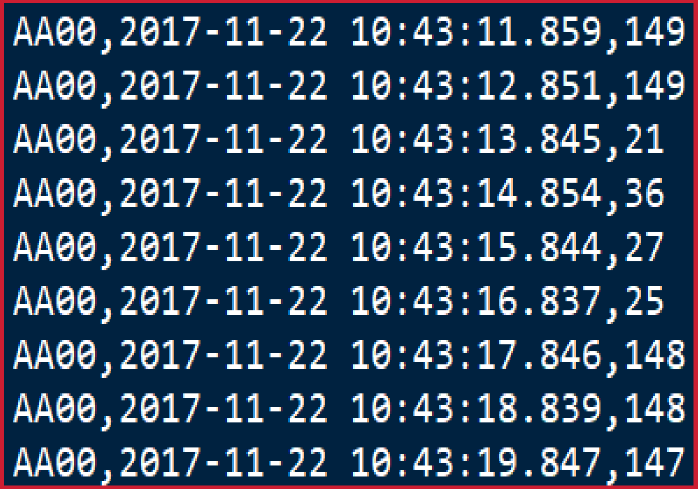




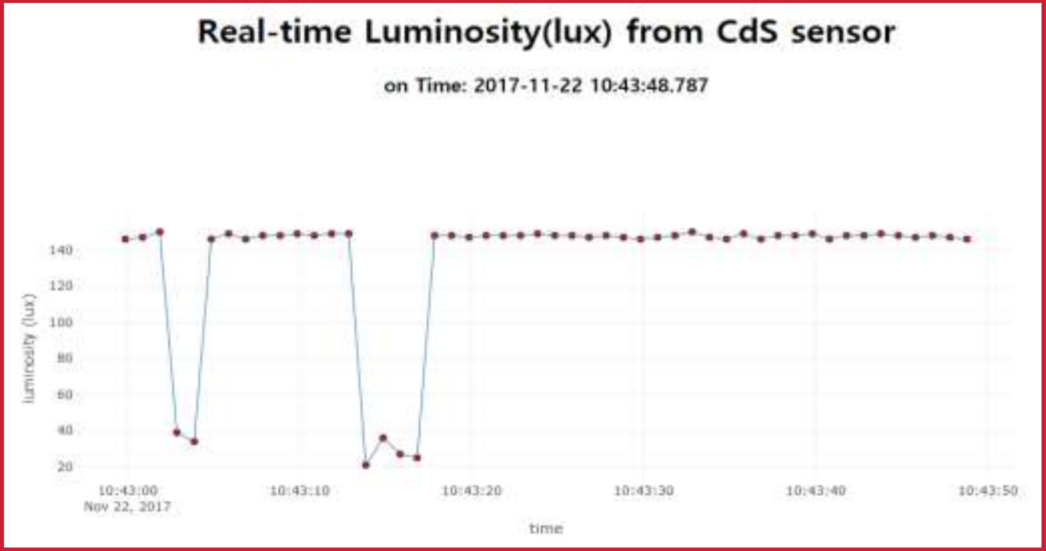
Arduino sensor data

RT visualization

using plotly.js

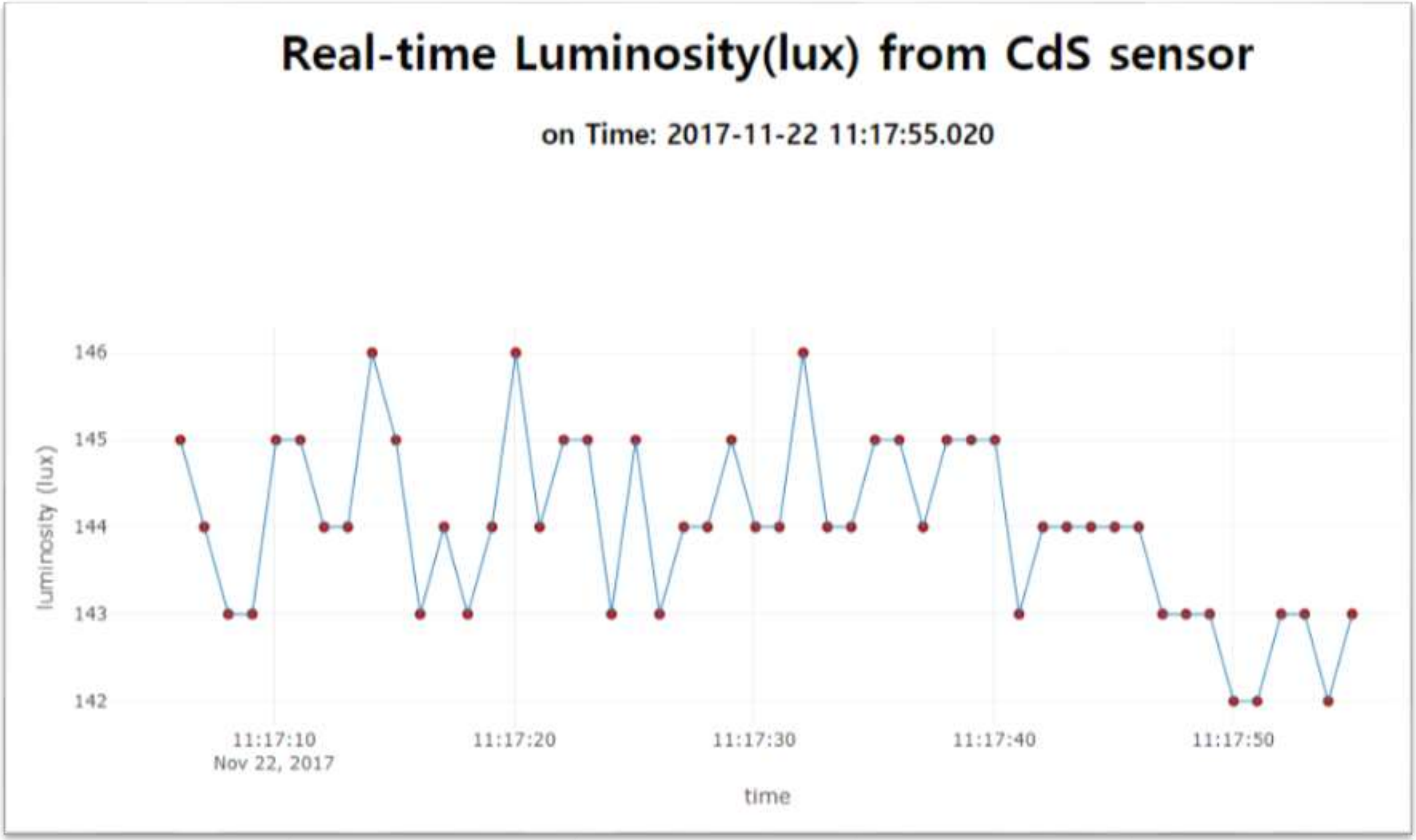


```
AA00,2017-11-22 10:43:11.859,149
AA00,2017-11-22 10:43:12.851,149
AA00,2017-11-22 10:43:13.845,21
AA00,2017-11-22 10:43:14.854,36
AA00,2017-11-22 10:43:15.844,27
AA00,2017-11-22 10:43:16.837,25
AA00,2017-11-22 10:43:17.846,148
AA00,2017-11-22 10:43:18.839,148
AA00,2017-11-22 10:43:19.847,147
```



A5.5.7.2 RT sensor-data streaming in Arduino

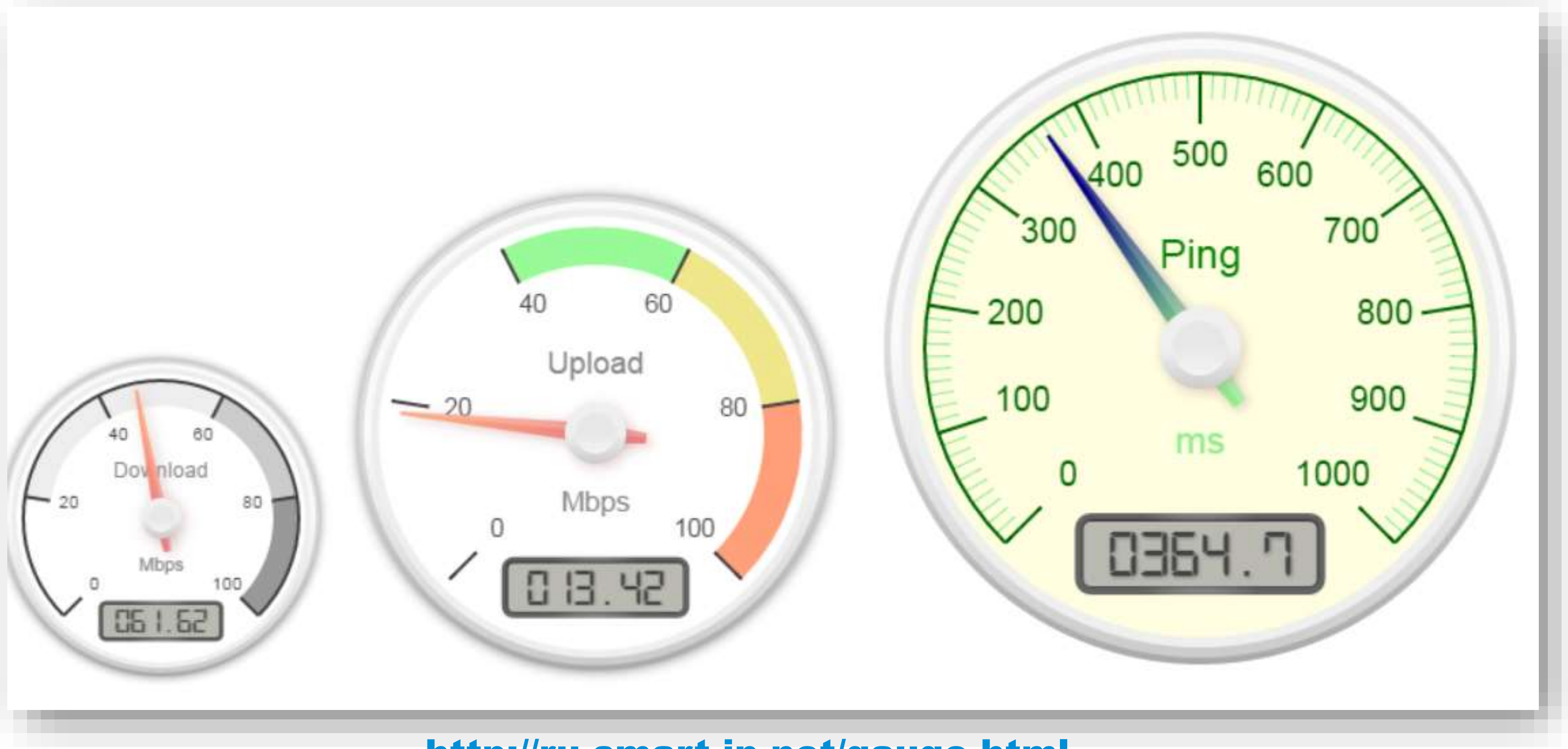
[7.2] Client html : client_cds2.html (using plotly streaming without nextPt())


Real-time Luminosity(lux) from CdS sensor

on Time: 2017-11-22 11:17:55.020

Canvas Gauge

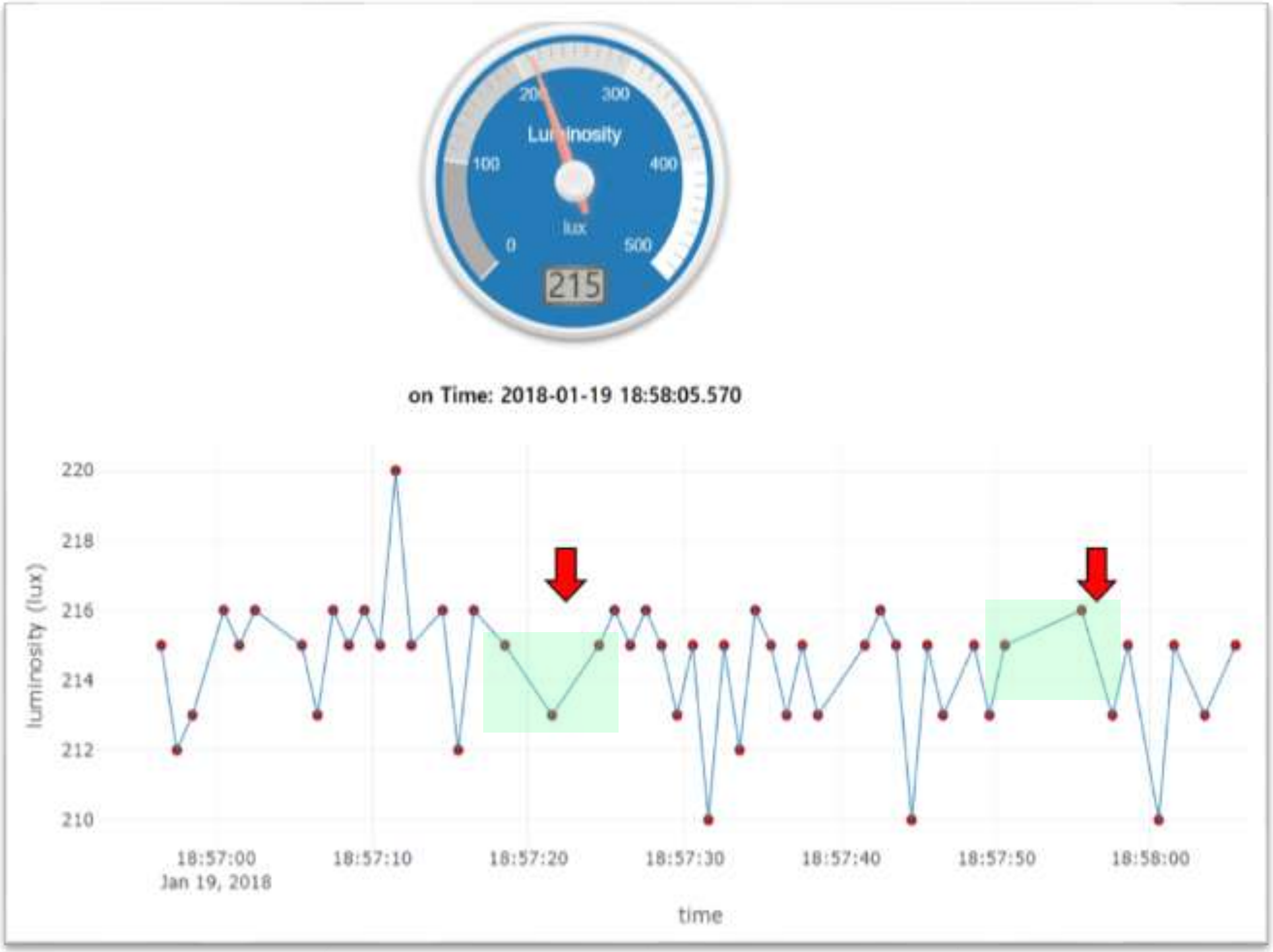
[1] Canvas gauge javascript library : example



http://ru.smart-ip.net/gauge.html

A5.5.9.3 RT sensor-data streaming in Arduino

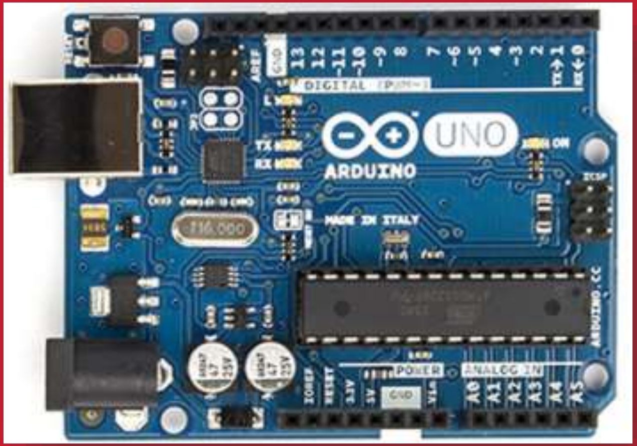
[DIY] Client html : client_cds_change.html (detecting change)



on Time: 2018-01-19 18:58:05.570

측정되는 주변광의 밝기가 일정 시간 유지되다가 변하는 그래프를 캡처하여 AAnn_cds_change.png 로 저장

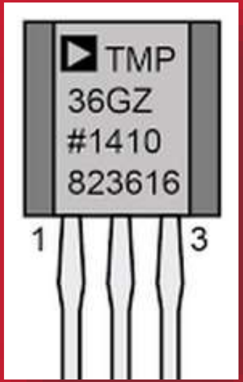
Multiple sensors



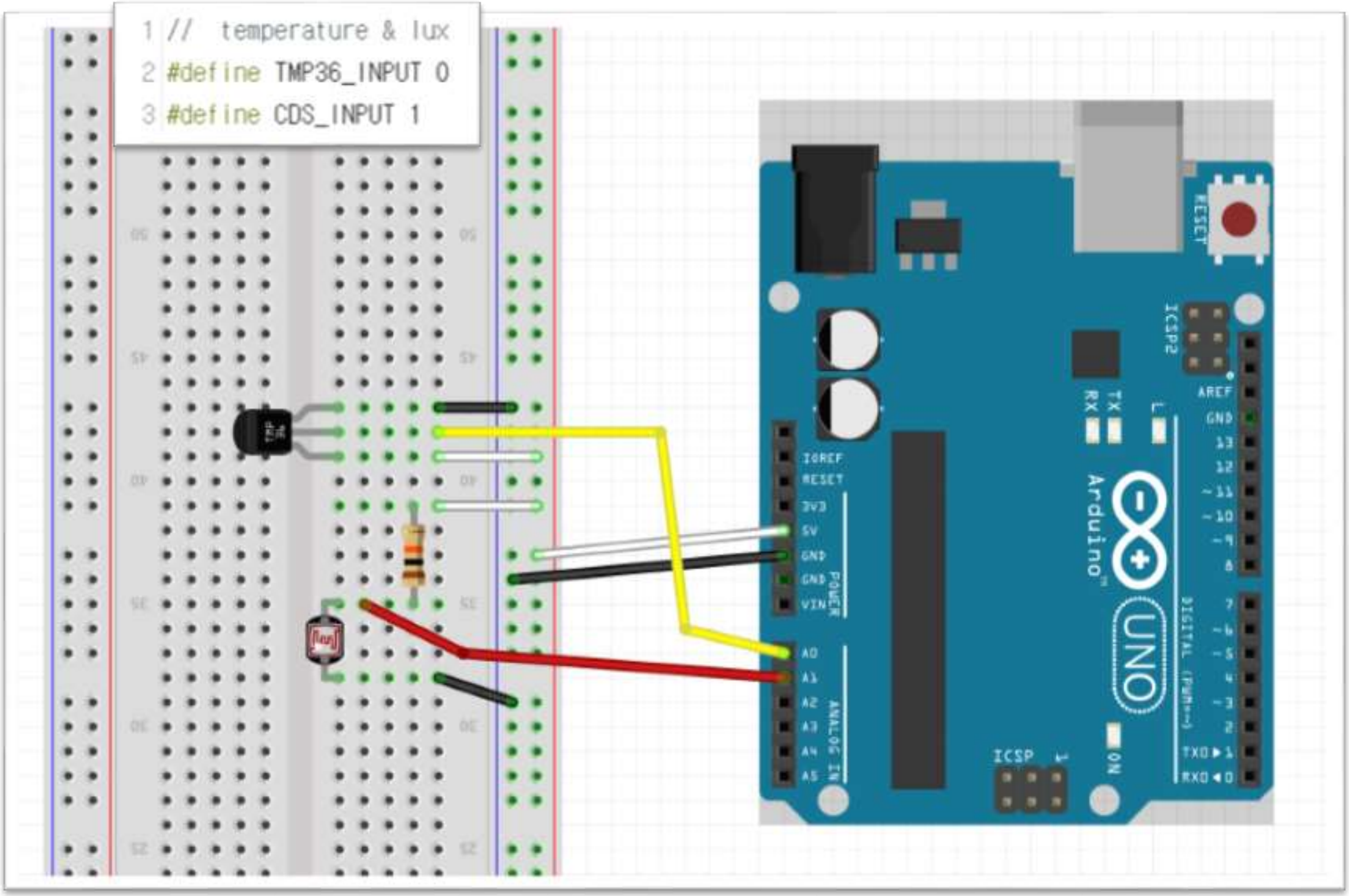
CdS + TMP36

+ plotly.js

Node project

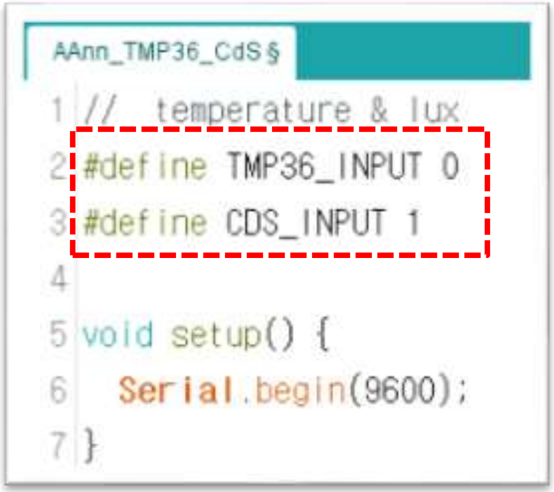


A4.3.1 TMP36 + CdS : circuit



```
// temperature & lux
#define TMP36_INPUT 0
#define CDS_INPUT 1
```

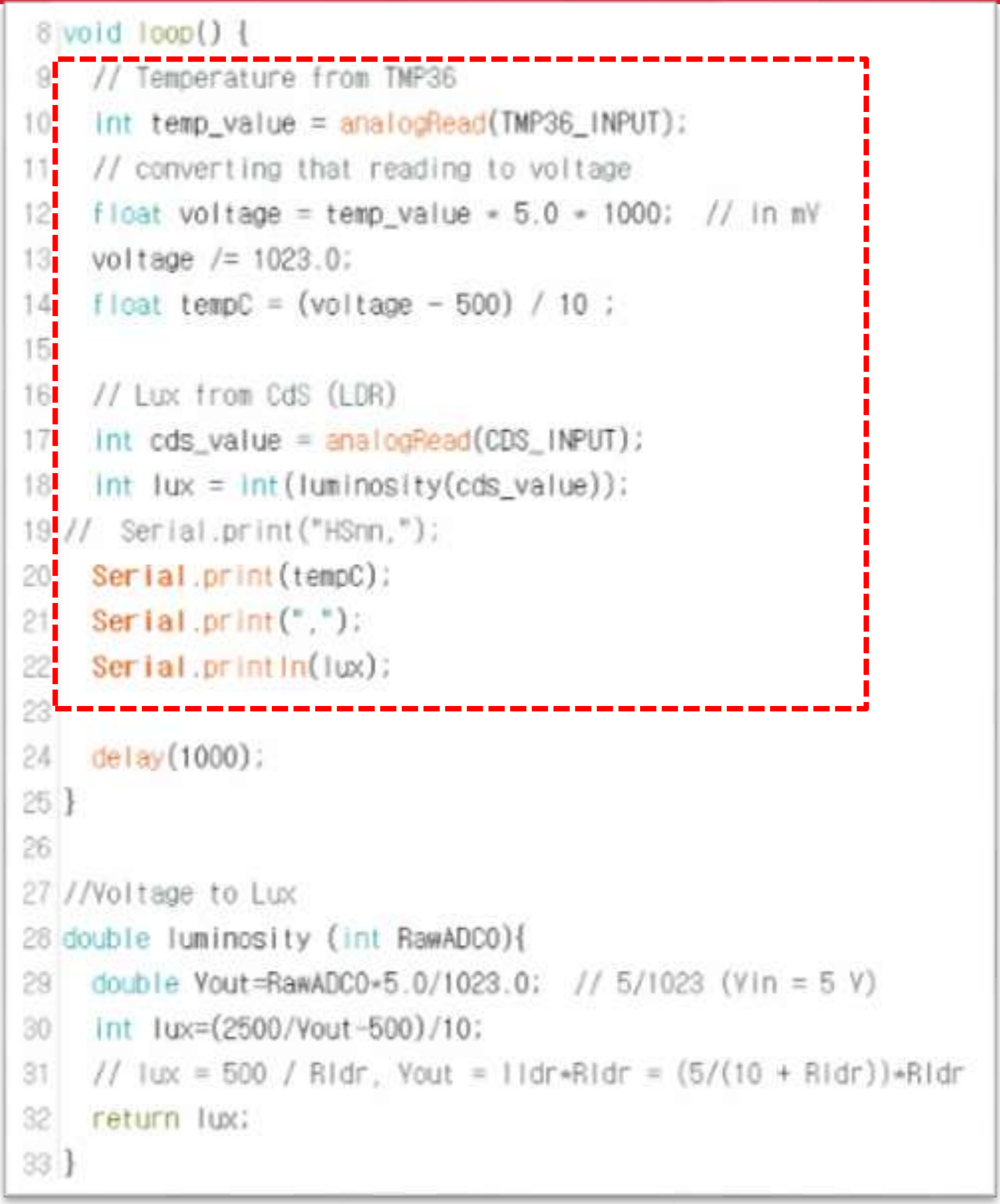
A4.3.2 TMP36 + CdS : code



AAnn_TMP36_CdS §

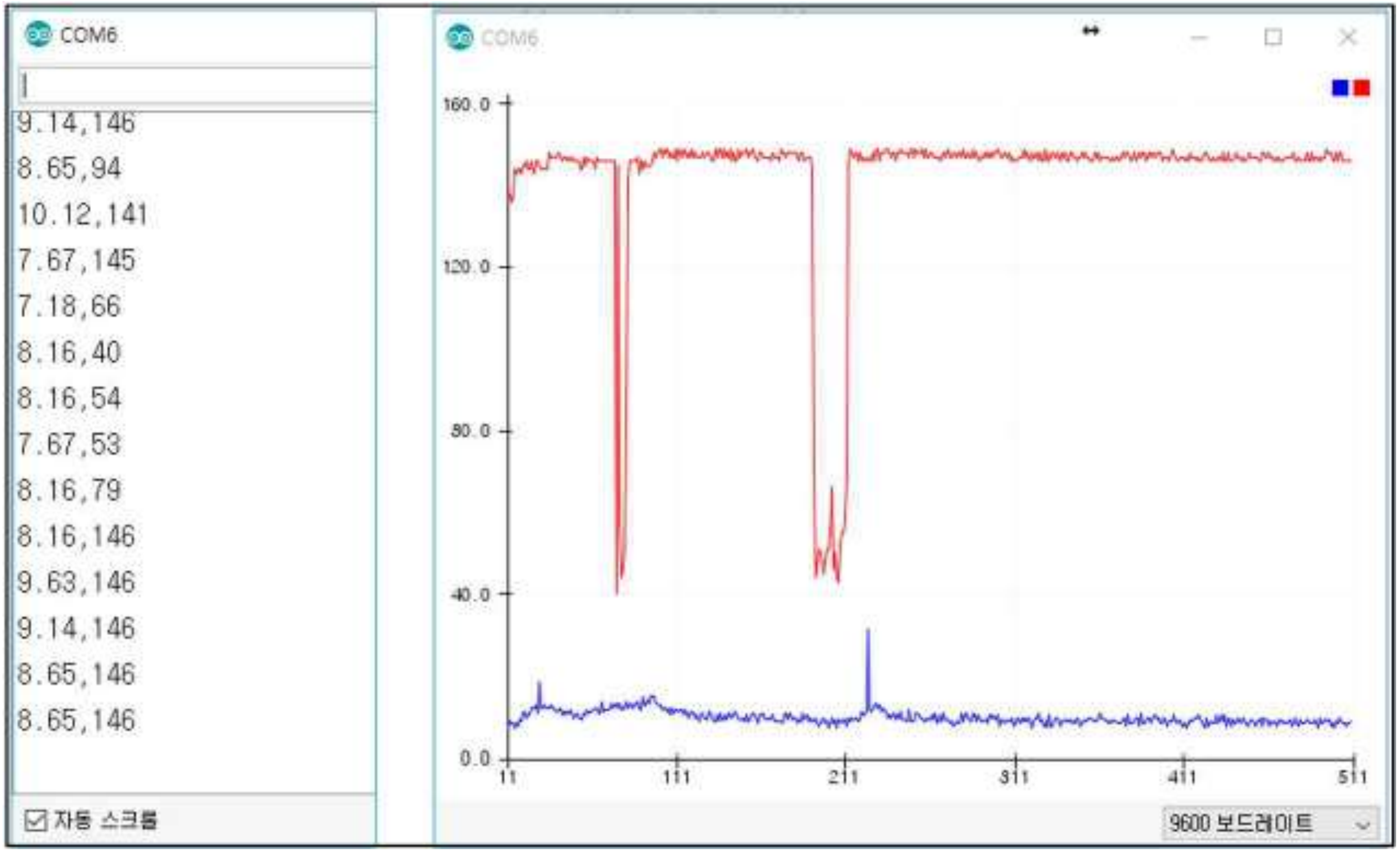
```
//  temperature & lux
#define TMP36_INPUT 0
#define CDS_INPUT 1

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

AAnn_cds_tmp36.ino



```
void loop() {
  // Temperature from TMP36
  int temp_value = analogRead(TMP36_INPUT);
  // converting that reading to voltage
  float voltage = temp_value * 5.0 * 1000;   // in mV
  voltage /= 1023.0;
  float tempC = (voltage - 500) / 10 ;

  // Lux from CdS (LDR)
  int cds_value = analogRead(CDS_INPUT);
  int lux = int(luminosity(cds_value));
//  Serial.print("HSnn,");
  Serial.print(tempC);
  Serial.print(",");
  Serial.println(lux);

  delay(1000);
}

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

A4.3.2 TMP36 + CdS : result



COM6

```
9.14,146
8.65,94
10.12,141
7.67,145
7.18,66
8.16,40
8.16,54
7.67,53
8.16,79
8.16,146
9.63,146
9.14,146
8.65,146
8.65,146
```

☑ 자동 스크롤

A4.5.1 CdS + TMP36 + Node project

1. Make cds_tmp36 node project

- ➢ md cds_tmp36 in iot folder
- ➢ cd cds_tmp36

2. Go to cds_tmp36 subfolder

- ➢ npm init

"main":
"cds_tmp36_node.js"
"author": “aann"

name : cds_tmp36

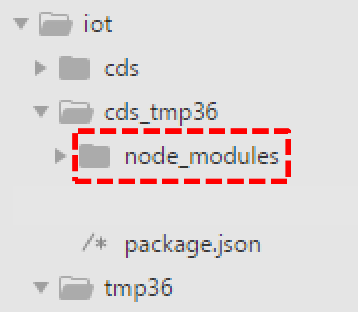
description : cds-tmp36-node project

entry point : cds_tmp36_node.js

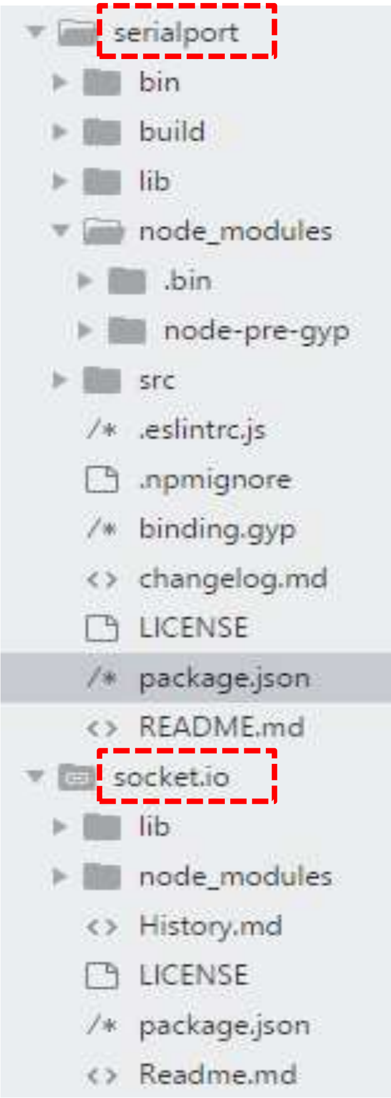
author : hsnn

A4.5.2 CdS + TMP36 + Node project

1. Make cds_tmp36 node project
   - md cds_tmp36 in iot folder
   - cd cds_tmp36
2. Go to cds_tmp36 subfolder
   - npm init
   - **npm install –save serialport@4.0.7**
   - **npm install –save socket.io@1.7.3**



```
iot
  cds
  cds_tmp36
    node_modules
    package.json
  tmp36
```



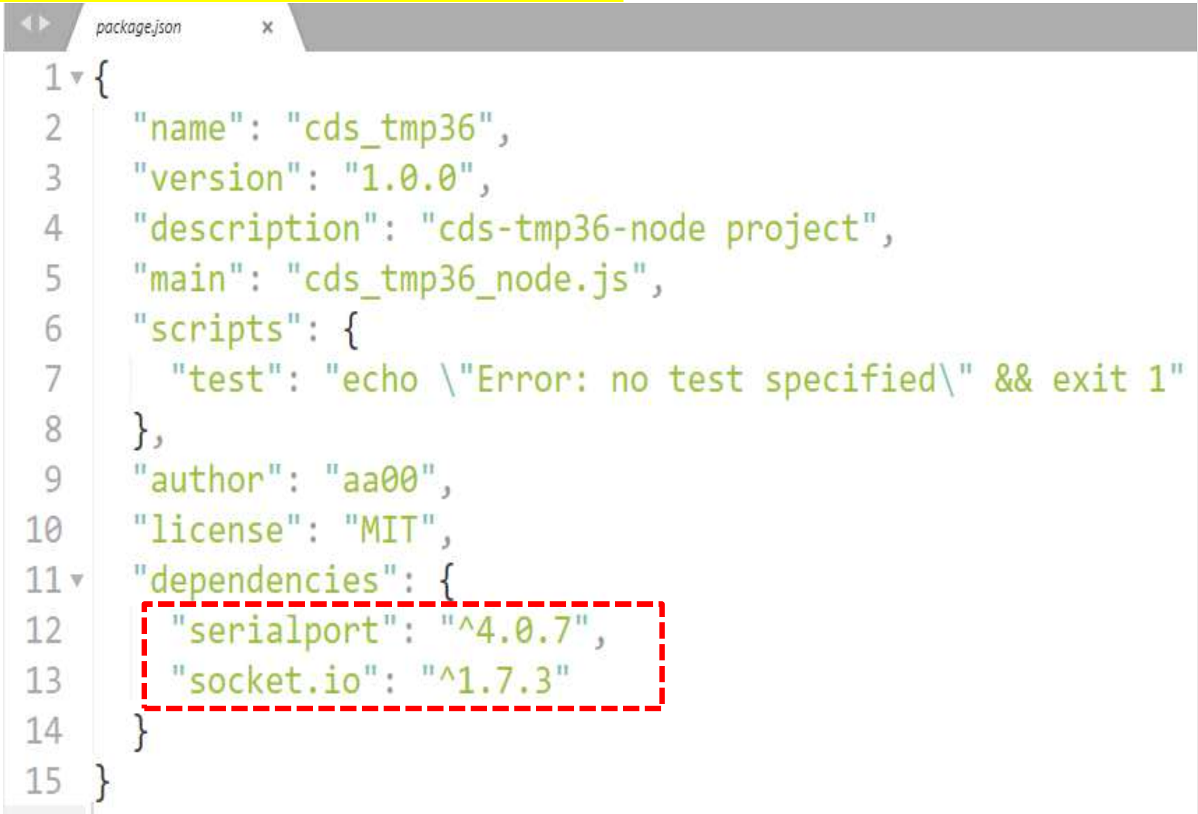
```
serialport
  bin
  build
  lib
  node_modules
    .bin
    node-pre-gyp
  src
  .eslintrc.js
  .npmignore
  binding.gyp
  changelog.md
  LICENSE
  package.json
  README.md
socket.io
  lib
  node_modules
  History.md
  LICENSE
  package.json
  Readme.md
```

You can check version of each module by browing package.json in each module subfolder.

A4.5.3 CdS + TMP36 + Node project

1. Make cds_tmp36 node project
   - ➢ md cds_tmp36
   - ➢ cd cds_tmp36
2. Go to cds_tmp36 subfolder
   - ➢ npm init
   - ➢ npm install –save serialport@4.0.7
   - ➢ npm install –save socket.io@1.7.3

package,json

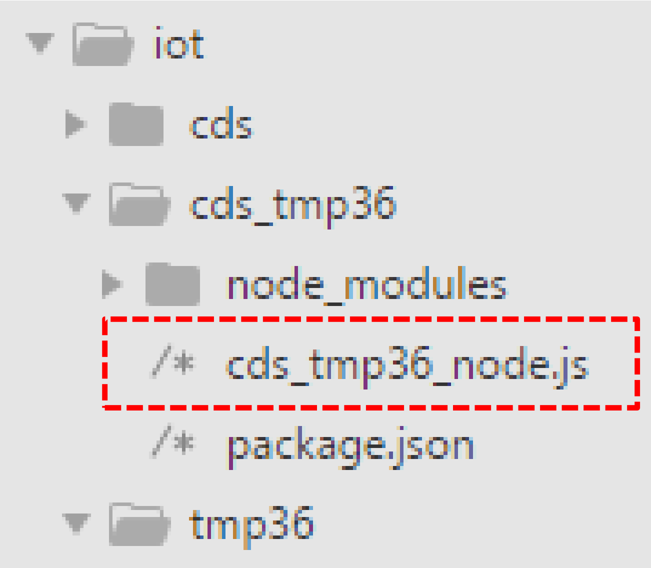


```
{
  "name": "cds_tmp36",
  "version": "1.0.0",
  "description": "cds-tmp36-node project",
  "main": "cds_tmp36_node.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "aa00",
  "license": "MIT",
  "dependencies": {
    "serialport": "^4.0.7",
    "socket.io": "^1.7.3"
  }
}
```

A4.5.4 CdS + TMP36 + Node project

Recycling code:

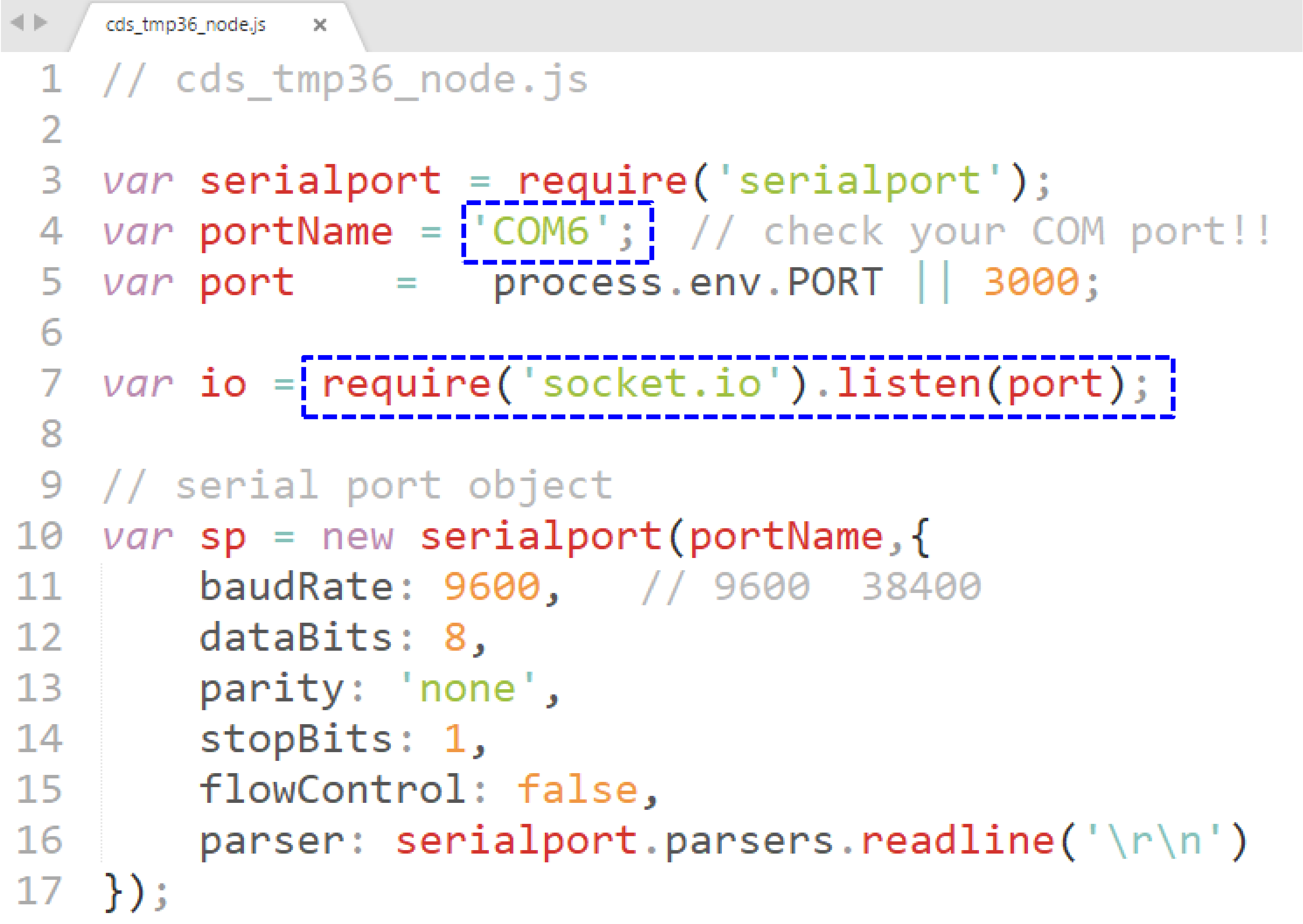
Save cds_node.js as
cds_tmp36_node.js



- iot
  - cds
  - cds_tmp36
    - node_modules
    - cds_tmp36_node.js
    - package.json
  - tmp36

A4.5.5.1 CdS + TMP36 + Node project : code-1

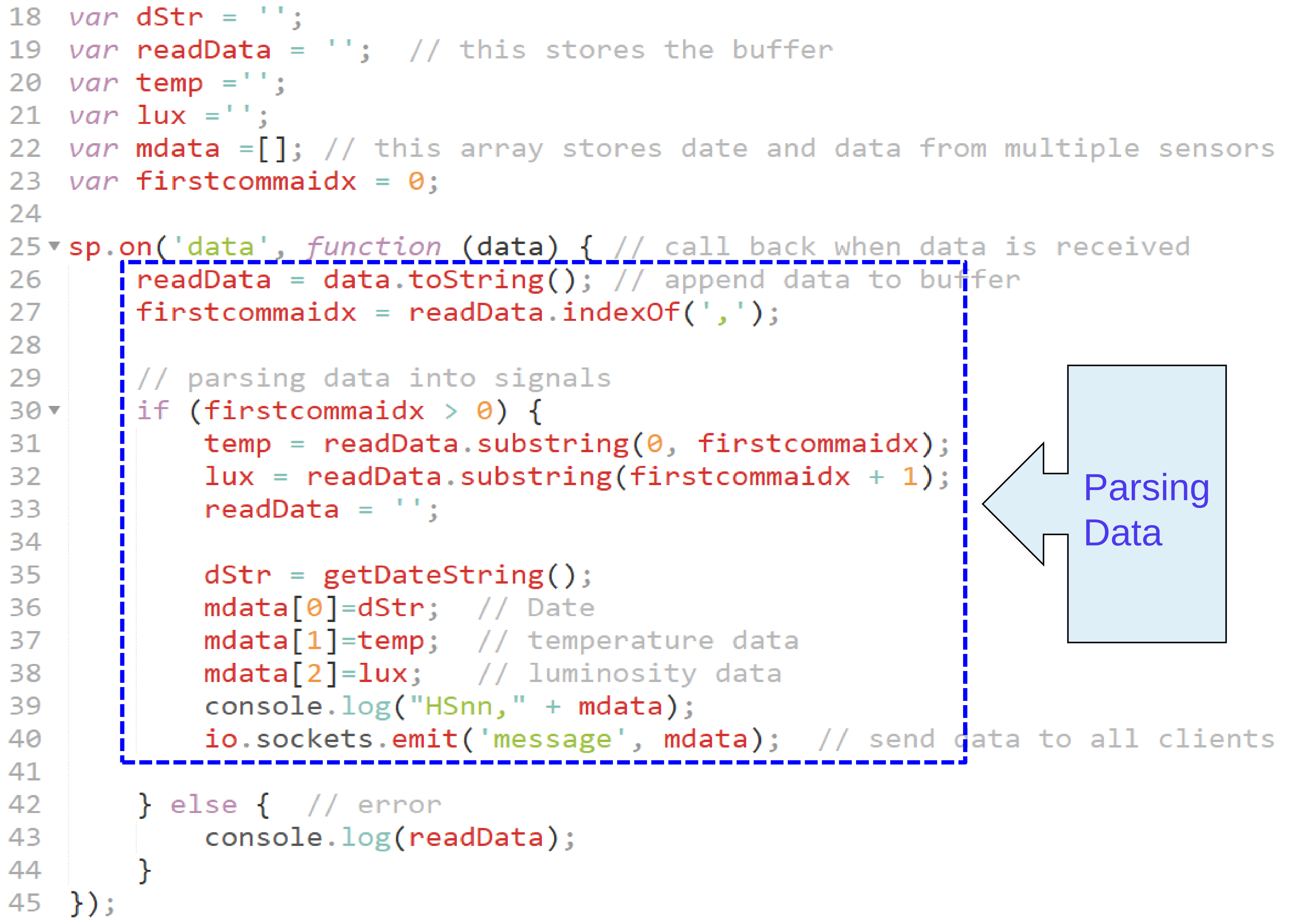
cds_tmp36_node.js



```
// cds_tmp36_node.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);

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

A4.5.5.2 CdS + TMP36 + Node project : code-2

cds_tmp36_node.js – parsing data (온도,조도)

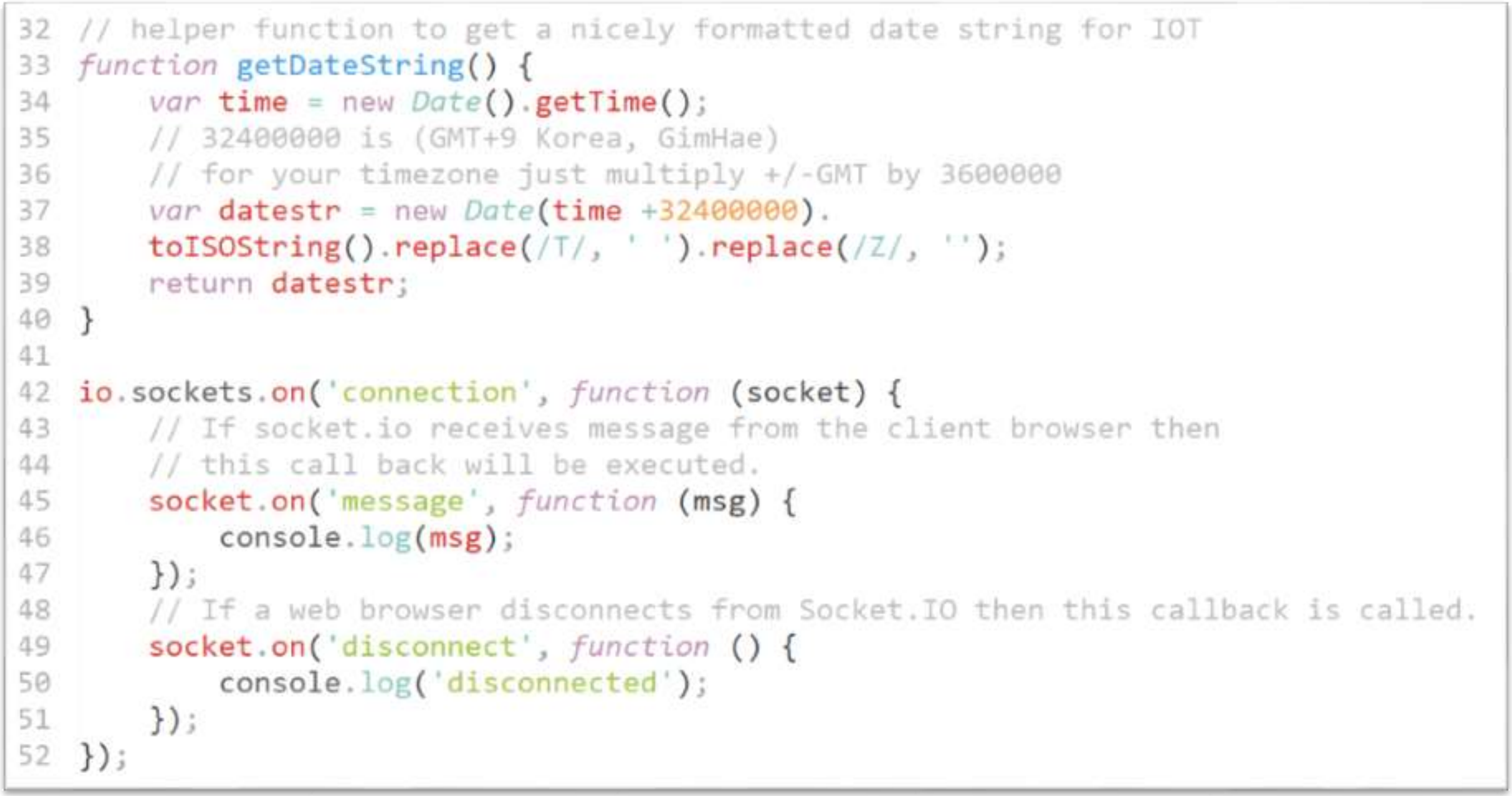


```
var dStr = '';
var readData = '';  // this stores the buffer
var temp ='';
var lux ='';
var mdata =[]; // this array stores date and data from multiple sensors
var firstcommaidx = 0;

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

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

        dStr = getDateString();
        mdata[0]=dStr;  // Date
        mdata[1]=temp;  // temperature data
        mdata[2]=lux;   // luminosity data
        console.log("HSnn," + mdata);
        io.sockets.emit('message', mdata);  // send data to all clients

    } else {  // error
        console.log(readData);
    }
});
```

Parsing Data

A4.5.5.3 CdS + TMP36 + Node project : code-3

cds_tmp36_node.js

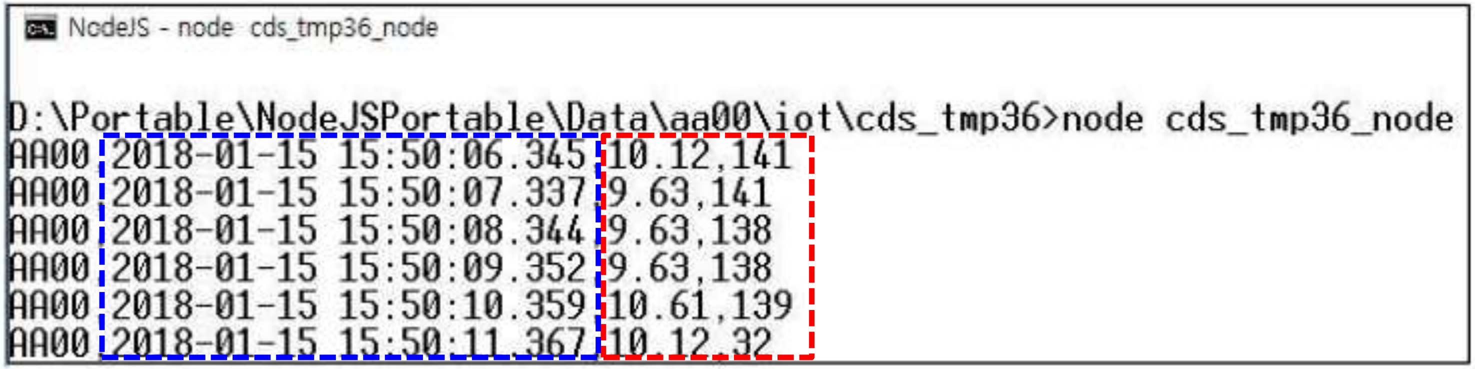


```
// helper function to get a nicely formatted date string for IOT
function getDateString() {
    var time = new Date().getTime();
    // 32400000 is (GMT+9 Korea, GimHae)
    // for your timezone just multiply +/-GMT by 3600000
    var datestr = new Date(time +32400000).
    toISOString().replace(/T/,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

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

A4.5.6 CdS + TMP36 + Node project : result

Node cmd 에서 실행

```
node cds_tmp36_node
```



NodeJS - node cds_tmp36_node

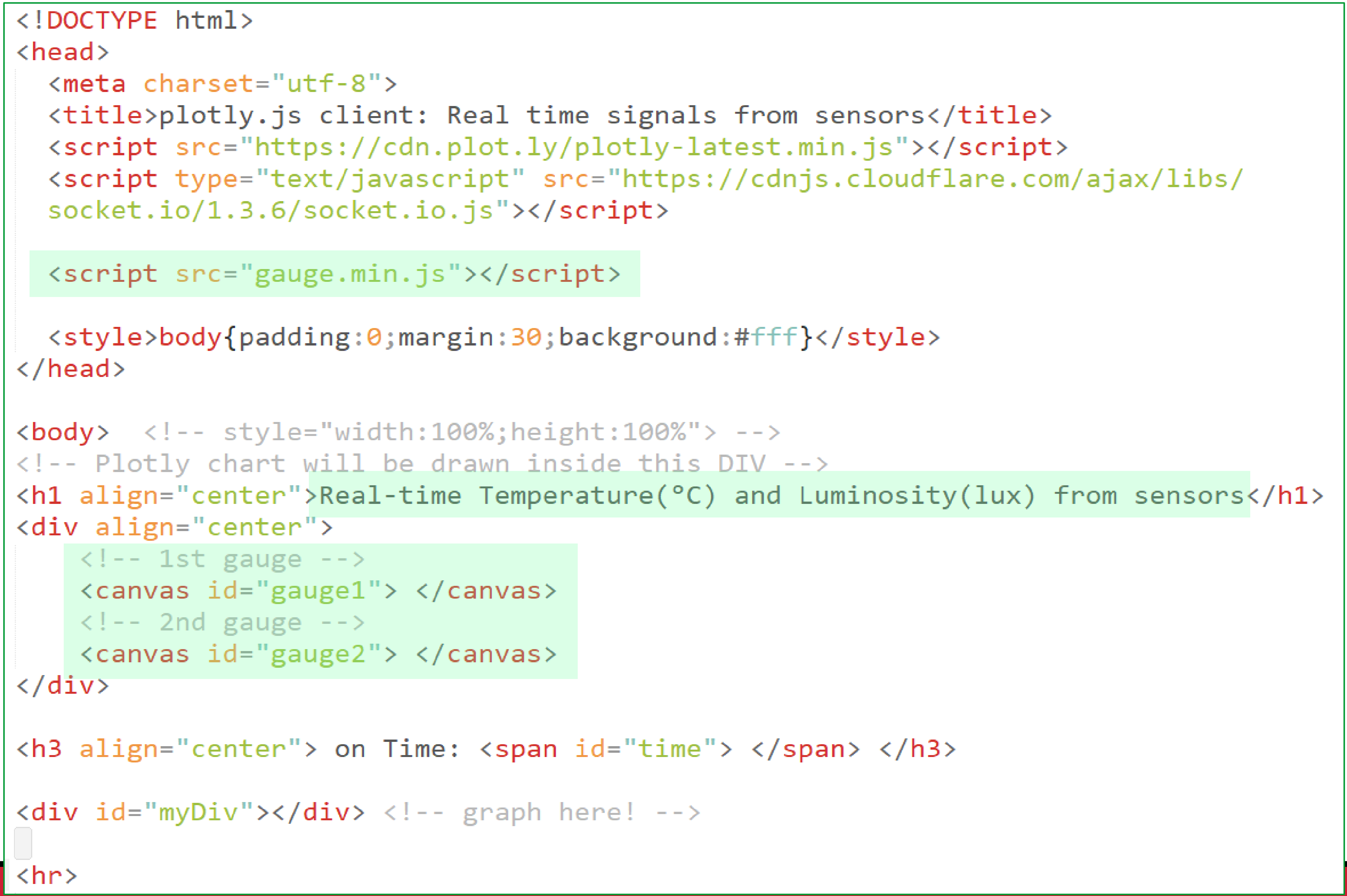
```
D:\Portable\NodeJSPortable\Data\aa00\iot\cds_tmp36>node cds_tmp36_node
AA00 2018-01-15 15:50:06.345 10.12,141
AA00 2018-01-15 15:50:07.337 9.63,141
AA00 2018-01-15 15:50:08.344 9.63,138
AA00 2018-01-15 15:50:09.352 9.63,138
AA00 2018-01-15 15:50:10.359 10.61,139
AA00 2018-01-15 15:50:11.367 10.12,32
```

IOT data format

시간, 온도,조도

A5.6.1 TMP36 + CdS streaming project

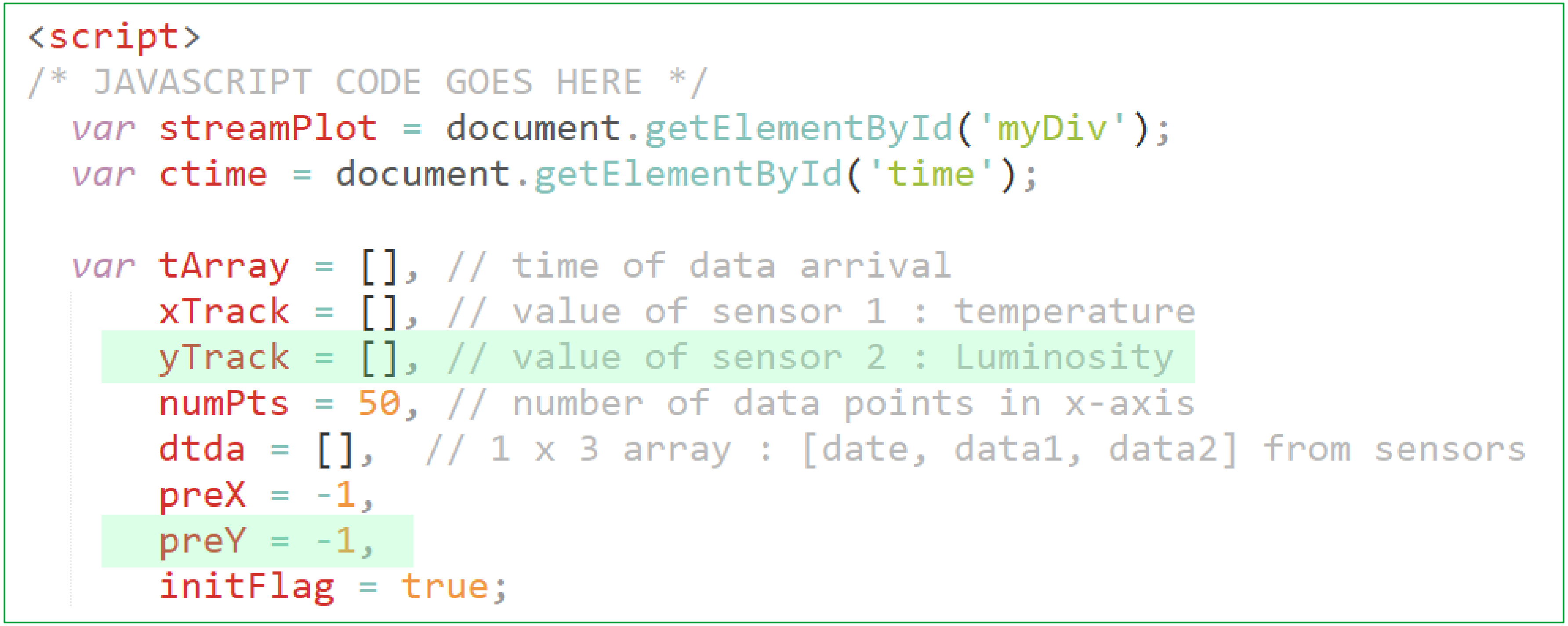
[DIY] Client html : client_cds_tmp36.html (data from multi sensors)



```
<!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 Temperature(°C) and Luminosity(lux) from sensors</h1>
<div align="center">
    <!-- 1st gauge -->
    <canvas id="gauge1"> </canvas>
    <!-- 2nd gauge -->
    <canvas id="gauge2"> </canvas>
</div>

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

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

<hr>
```

A5.6.2 TMP36 + CdS streaming project

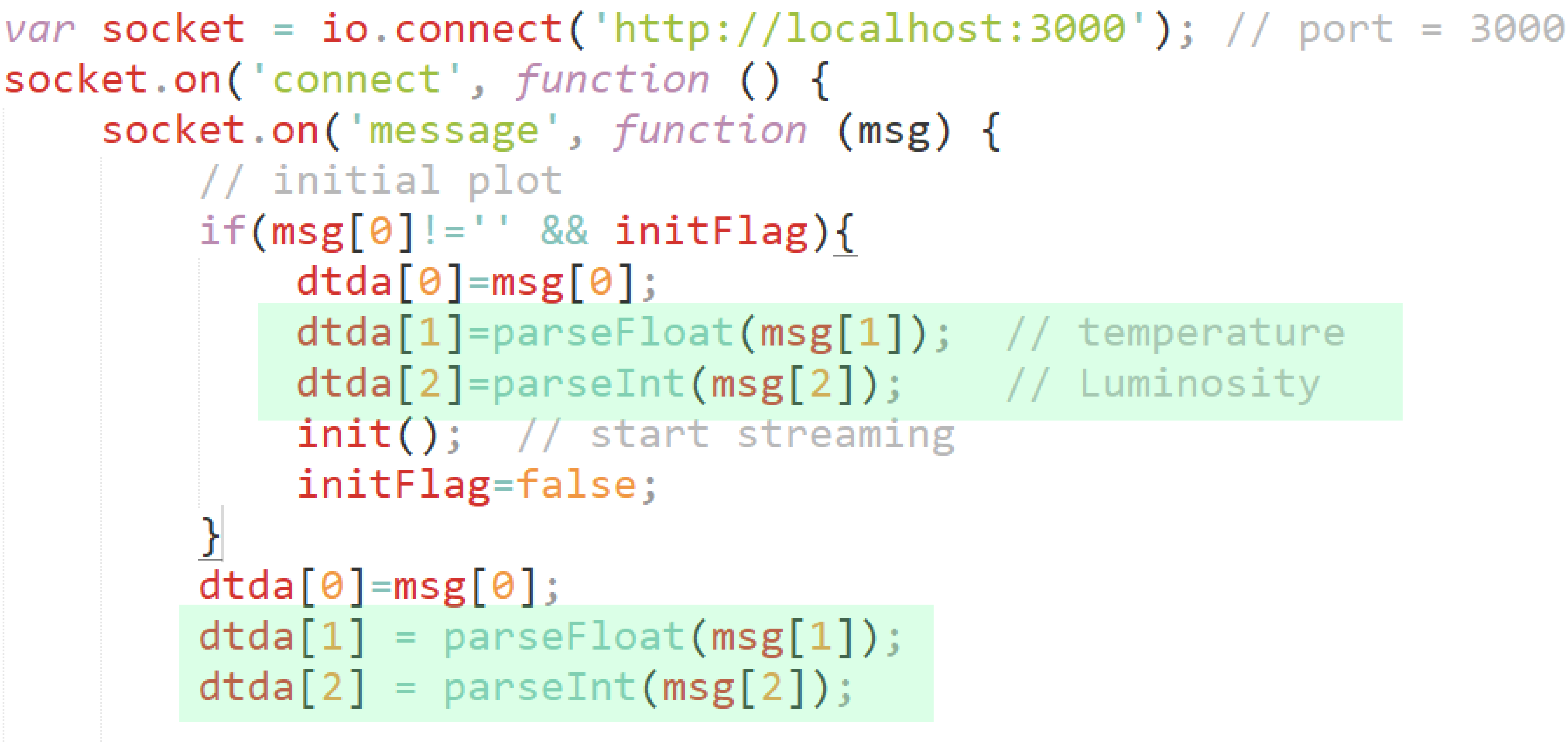
[DIY] Client html : client_cds_tmp36.html (data from multi sensors)



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

  var tArray = [], // time of data arrival
      xTrack = [], // value of sensor 1 : temperature
      yTrack = [], // value of sensor 2 : Luminosity
      numPts = 50, // number of data points in x-axis
      dtda = [],  // 1 x 3 array : [date, data1, data2] from sensors
      preX = -1,
      preY = -1,
      initFlag = true;
```

A5.6.3 TMP36 + CdS streaming project

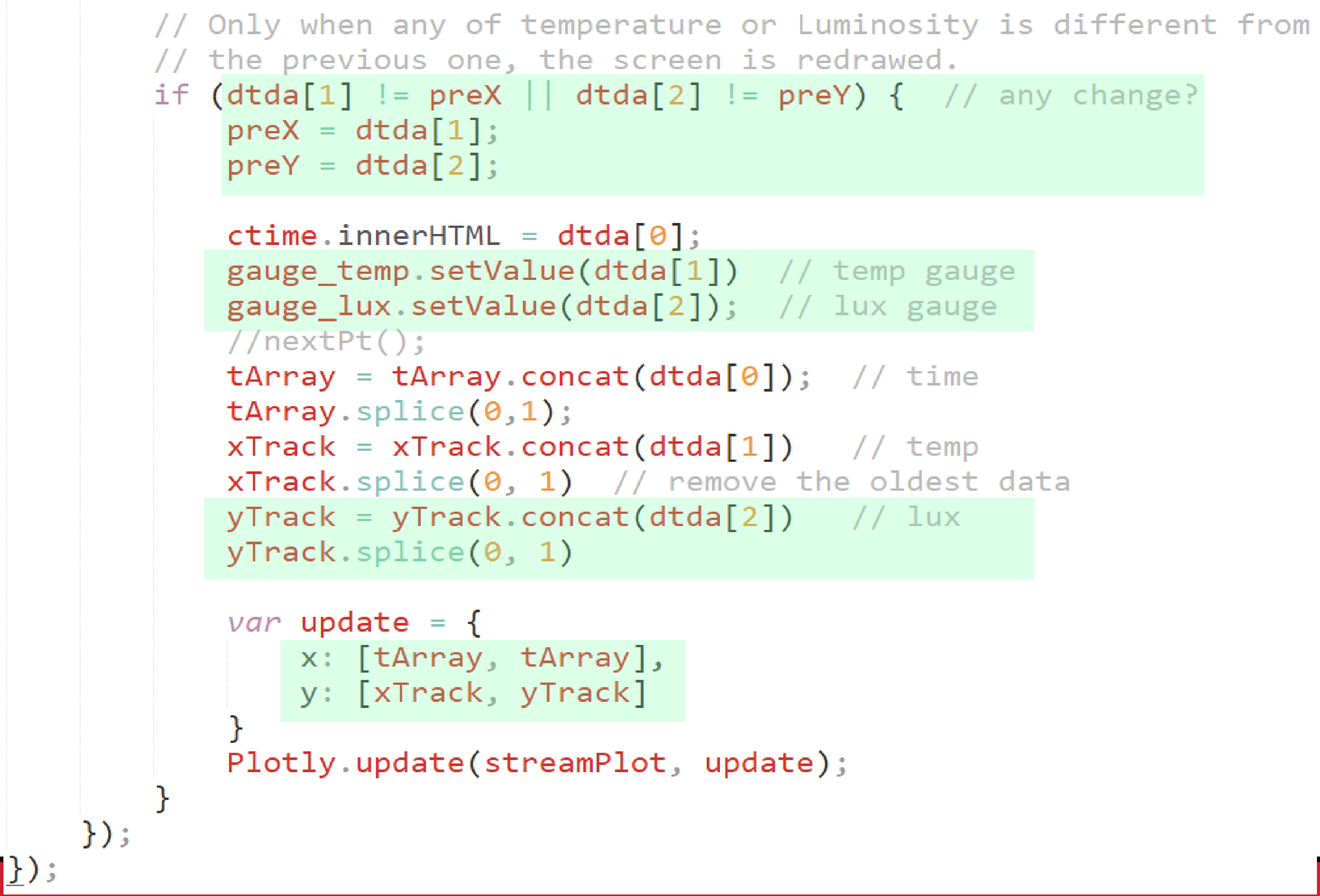
[DIY] Client html : client_cds_tmp36.html (data from multi sensors)



```
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]=parseFloat(msg[1]);  // temperature
            dtda[2]=parseInt(msg[2]);    // Luminosity
            init();  // start streaming
            initFlag=false;
        }
        dtda[0]=msg[0];
        dtda[1] = parseFloat(msg[1]);
        dtda[2] = parseInt(msg[2]);
```

A5.6.4 TMP36 + CdS streaming project

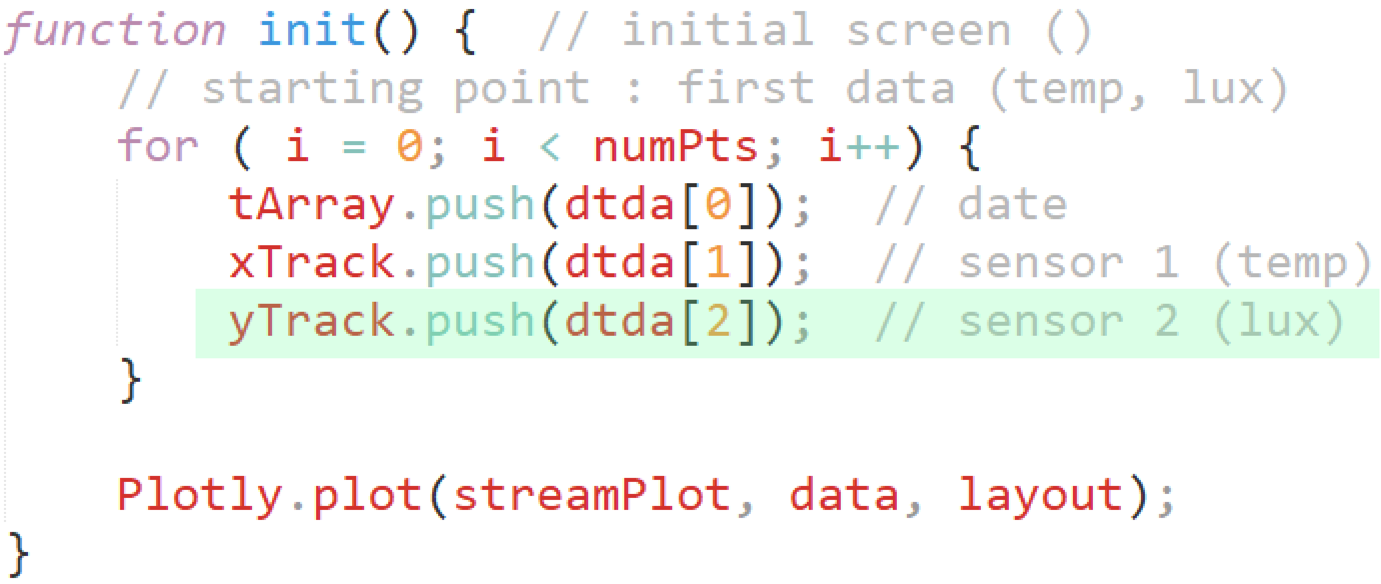
[DIY] Client html : client_cds_tmp36.html (data from multi sensors)



```
        // Only when any of temperature or Luminosity is different from
        // the previous one, the screen is redrawed.
        if (dtda[1] != preX || dtda[2] != preY) {  // any change?
            preX = dtda[1];
            preY = dtda[2];

            ctime.innerHTML = dtda[0];
            gauge_temp.setValue(dtda[1])  // temp gauge
            gauge_lux.setValue(dtda[2]);  // lux gauge
            //nextPt();
            tArray = tArray.concat(dtda[0]);  // time
            tArray.splice(0,1);
            xTrack = xTrack.concat(dtda[1])   // temp
            xTrack.splice(0, 1)  // remove the oldest data
            yTrack = yTrack.concat(dtda[2])   // lux
            yTrack.splice(0, 1)

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

A5.6.5 TMP36 + CdS streaming project

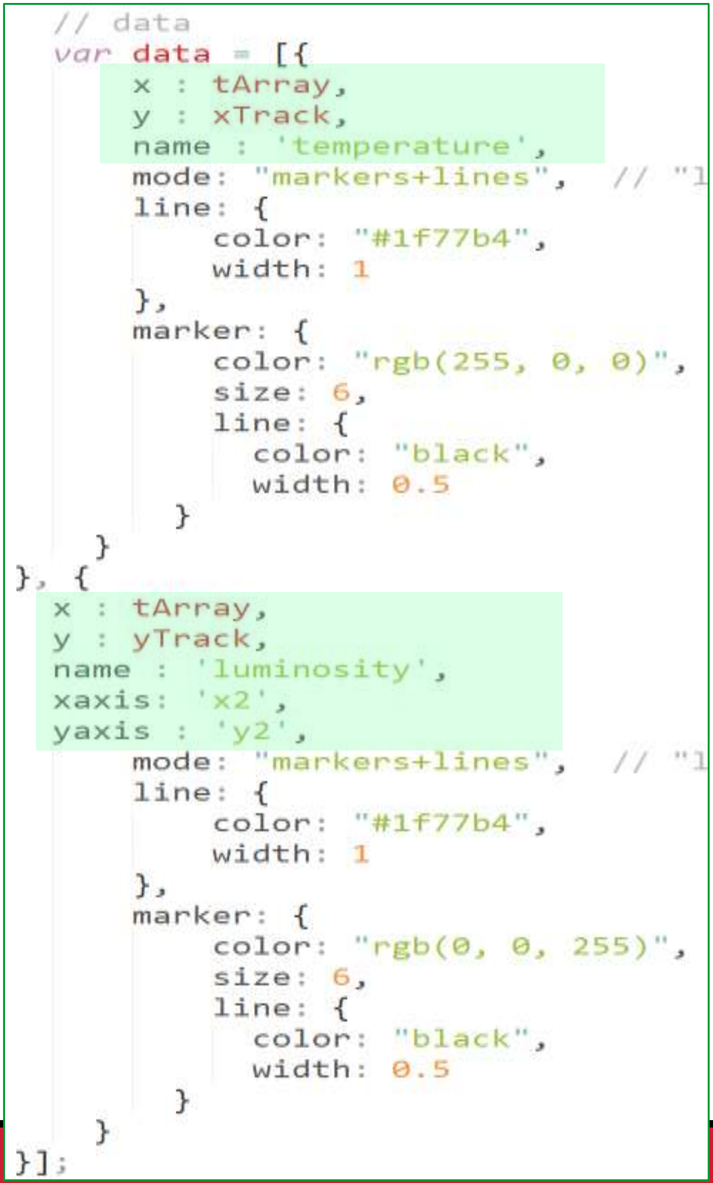
[DIY] Client html : client_cds_tmp36.html (data from multi sensors)



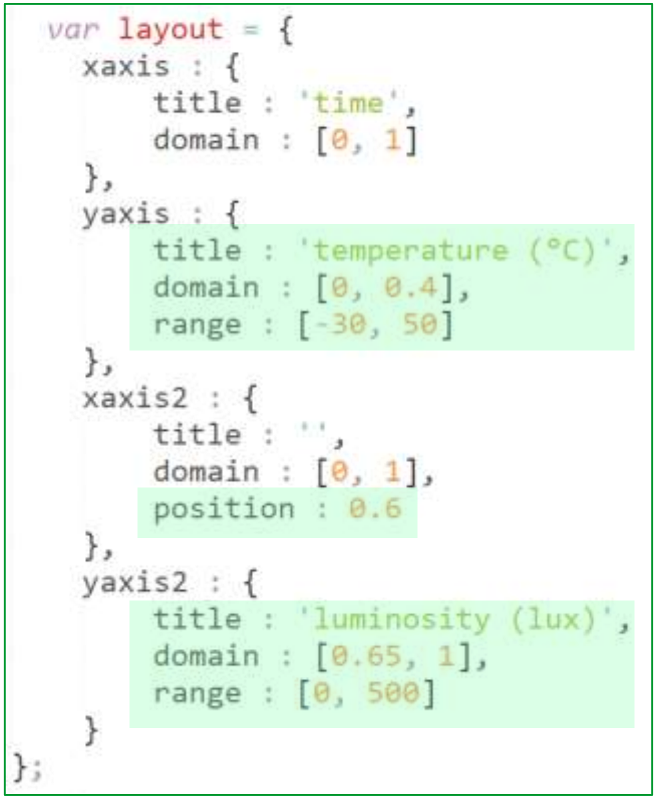
```
function init() {  // initial screen ()
    // starting point : first data (temp, lux)
    for ( i = 0; i < numPts; i++) {
        tArray.push(dtda[0]);  // date
        xTrack.push(dtda[1]);  // sensor 1 (temp)
        yTrack.push(dtda[2]);  // sensor 2 (lux)
    }

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

A5.6.6 TMP36 + CdS streaming project

[DIY] Client html : client_cds_tmp36.html (data from multi sensors)



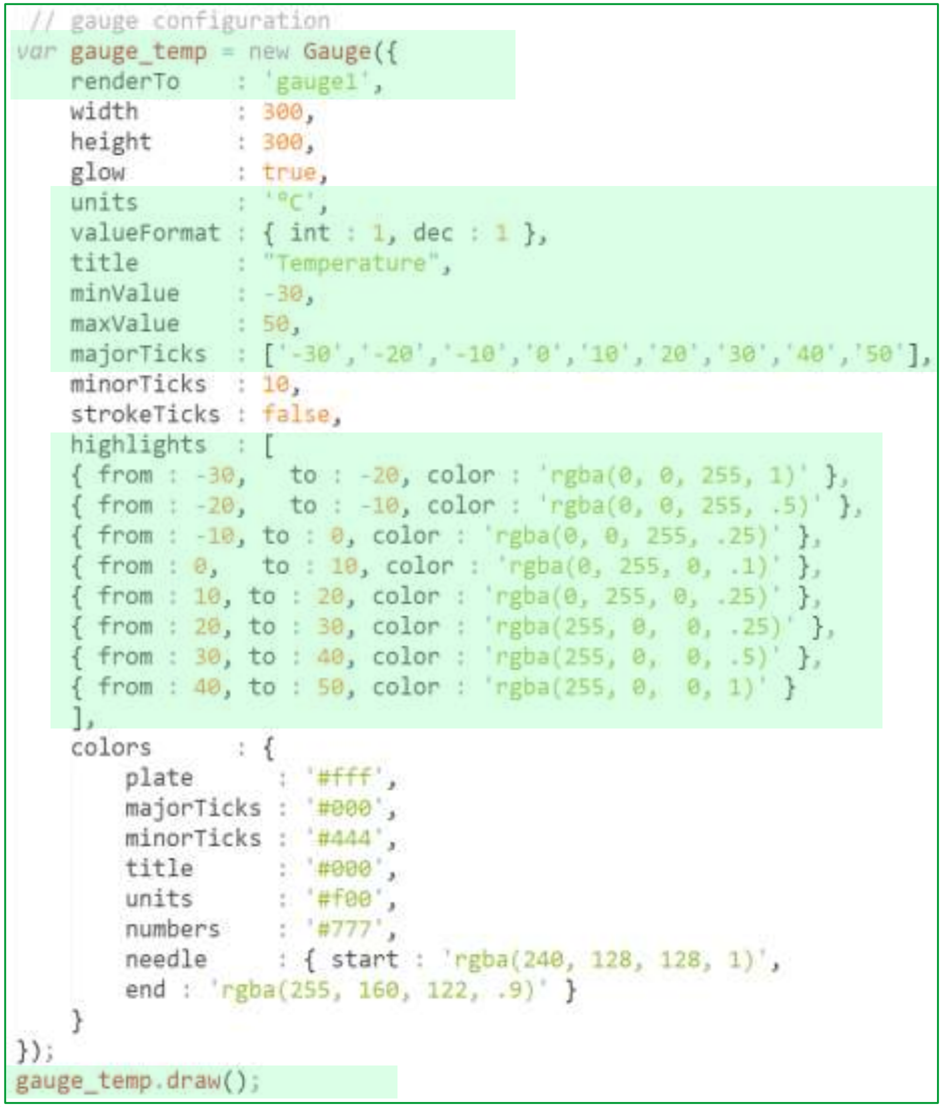
```
  // data
  var data = [{
      x : tArray,
      y : xTrack,
      name : 'temperature',
      mode: "markers+lines",   // "l
      line: {
          color: "#1f77b4",
          width: 1
      },
      marker: {
          color: "rgb(255, 0, 0)",
          size: 6,
          line: {
            color: "black",
            width: 0.5
          }
      }
}, {
  x : tArray,
  y : yTrack,
  name : 'luminosity',
  xaxis: 'x2',
  yaxis : 'y2',
      mode: "markers+lines",   // "l
      line: {
          color: "#1f77b4",
          width: 1
      },
      marker: {
          color: "rgb(0, 0, 255)",
          size: 6,
          line: {
            color: "black",
            width: 0.5
          }
      }
}];
```



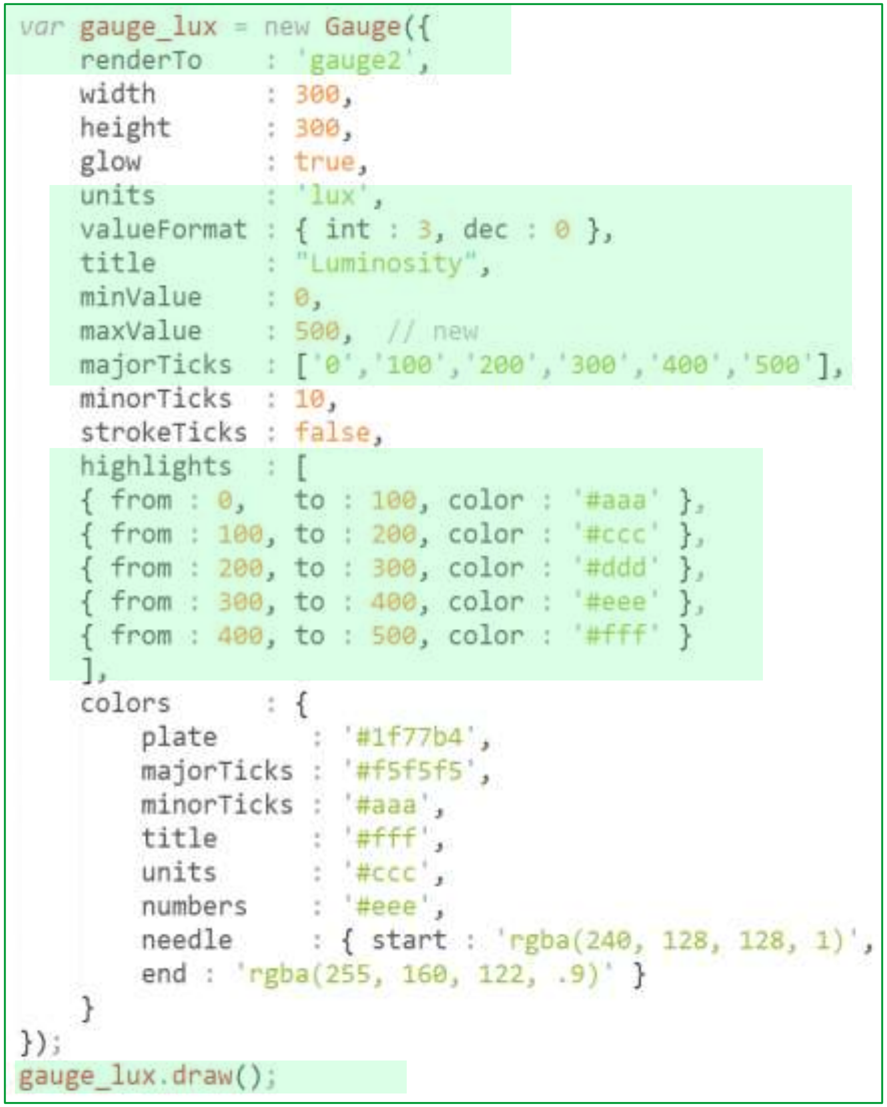
```
  var layout = {
    xaxis : {
        title : 'time',
        domain : [0, 1]
    },
    yaxis : {
        title : 'temperature (°C)',
        domain : [0, 0.4],
        range : [-30, 50]
    },
    xaxis2 : {
        title : '',
        domain : [0, 1],
        position : 0.6
    },
    yaxis2 : {
        title : 'luminosity (lux)',
        domain : [0.65, 1],
        range : [0, 500]
    }
};
```

A5.6.7 TMP36 + CdS streaming project

[DIY] Client html : client_cds_tmp36.html (data from multi sensors)



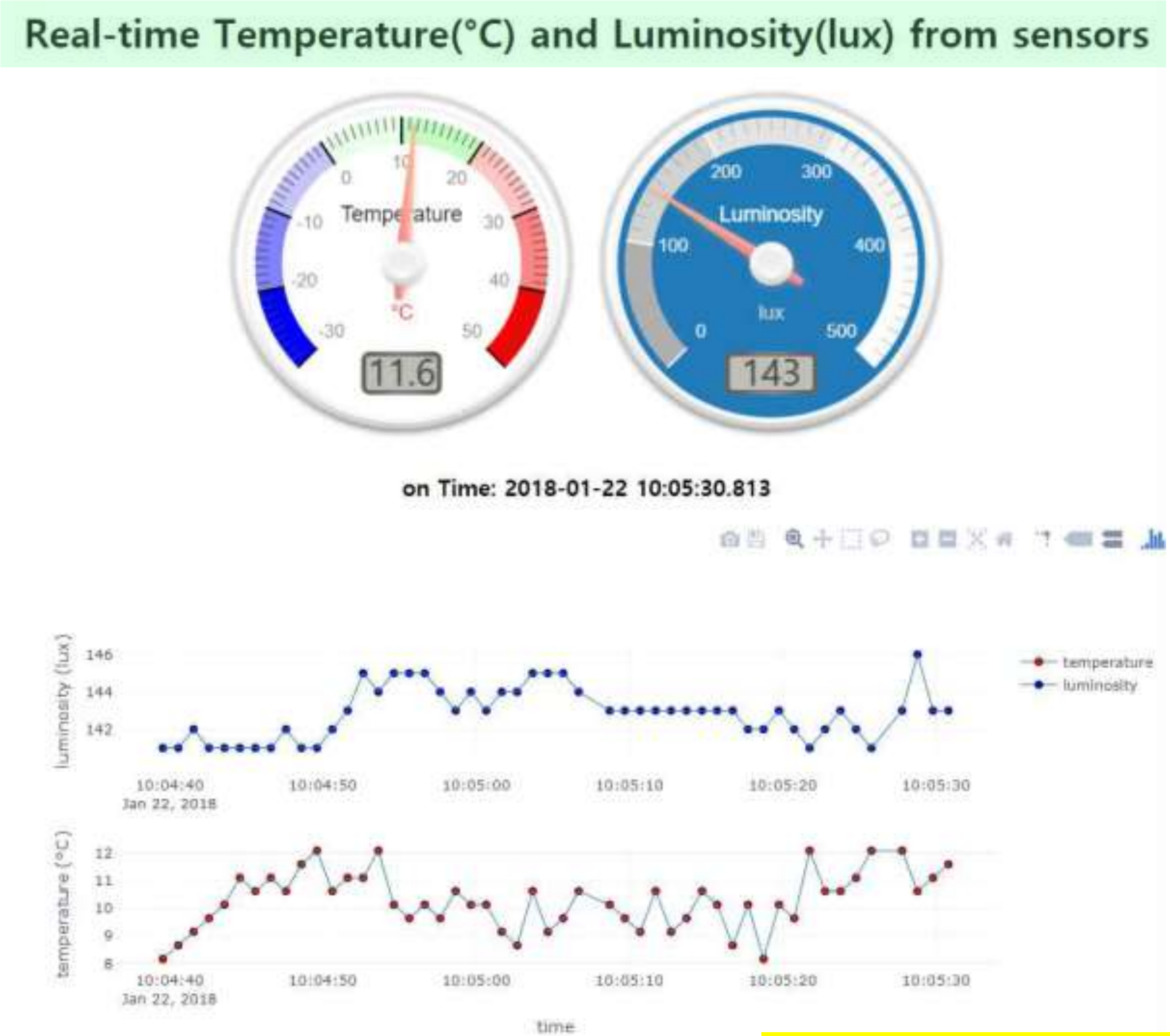
```
// gauge configuration
var gauge_temp = new Gauge({
    renderTo    : 'gauge1',
    width       : 300,
    height      : 300,
    glow        : true,
    units       : '°C',
    valueFormat : { int : 1, dec : 1 },
    title       : "Temperature",
    minValue    : -30,
    maxValue    : 50,
    majorTicks  : ['-30','-20','-10','0','10','20','30','40','50'],
    minorTicks  : 10,
    strokeTicks : false,
    highlights  : [
    { from : -30,   to : -20, color : 'rgba(0, 0, 255, 1)' },
    { from : -20,   to : -10, color : 'rgba(0, 0, 255, .5)' },
    { from : -10, to : 0, color : 'rgba(0, 0, 255, .25)' },
    { from : 0,   to : 10, color : 'rgba(0, 255, 0, .1)' },
    { from : 10, to : 20, color : 'rgba(0, 255, 0, .25)' },
    { from : 20, to : 30, color : 'rgba(255, 0,  0, .25)' },
    { from : 30, to : 40, color : 'rgba(255, 0,  0, .5)' },
    { from : 40, to : 50, color : 'rgba(255, 0,  0, 1)' }
    ],
    colors      : {
        plate      : '#fff',
        majorTicks : '#000',
        minorTicks : '#444',
        title      : '#000',
        units      : '#f00',
        numbers    : '#777',
        needle     : { start : 'rgba(240, 128, 128, 1)',
        end : 'rgba(255, 160, 122, .9)' }
    }
});
gauge_temp.draw();
```



```
var gauge_lux = new Gauge({
    renderTo     : 'gauge2',
    width        : 300,
    height       : 300,
    glow         : true,
    units        : 'lux',
    valueFormat  : { int : 3, dec : 0 },
    title        : "Luminosity",
    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       : '#1f77b4',
        majorTicks  : '#f5f5f5',
        minorTicks  : '#aaa',
        title       : '#fff',
        units       : '#ccc',
        numbers     : '#eee',
        needle      : { start : 'rgba(240, 128, 128, 1)',
        end : 'rgba(255, 160, 122, .9)' }
    }
});
gauge_lux.draw();
```

A5.6.8 TMP36 + CdS streaming project

[DIY] Client html : client_cds_tmp36.html (result)



Real-time Temperature(°C) and Luminosity(lux) from sensors

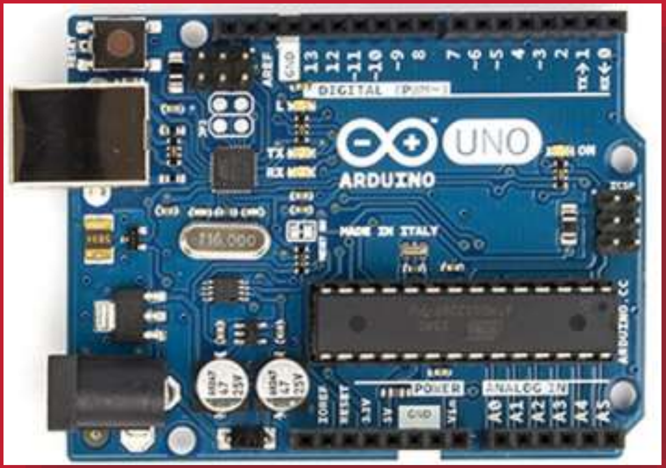
on Time: 2018-01-22 10:05:30.813

AAnn_DS_cds_tmp36.png 로 저장

CdS + DHT22

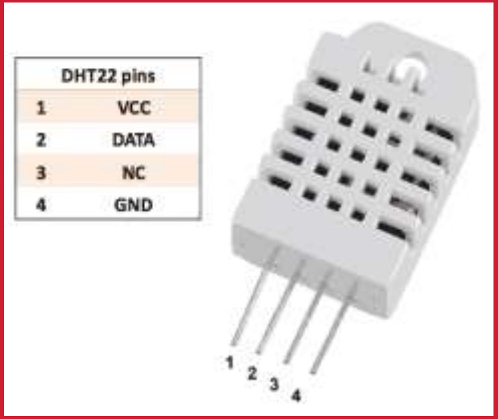
+ plotly.js

Node project

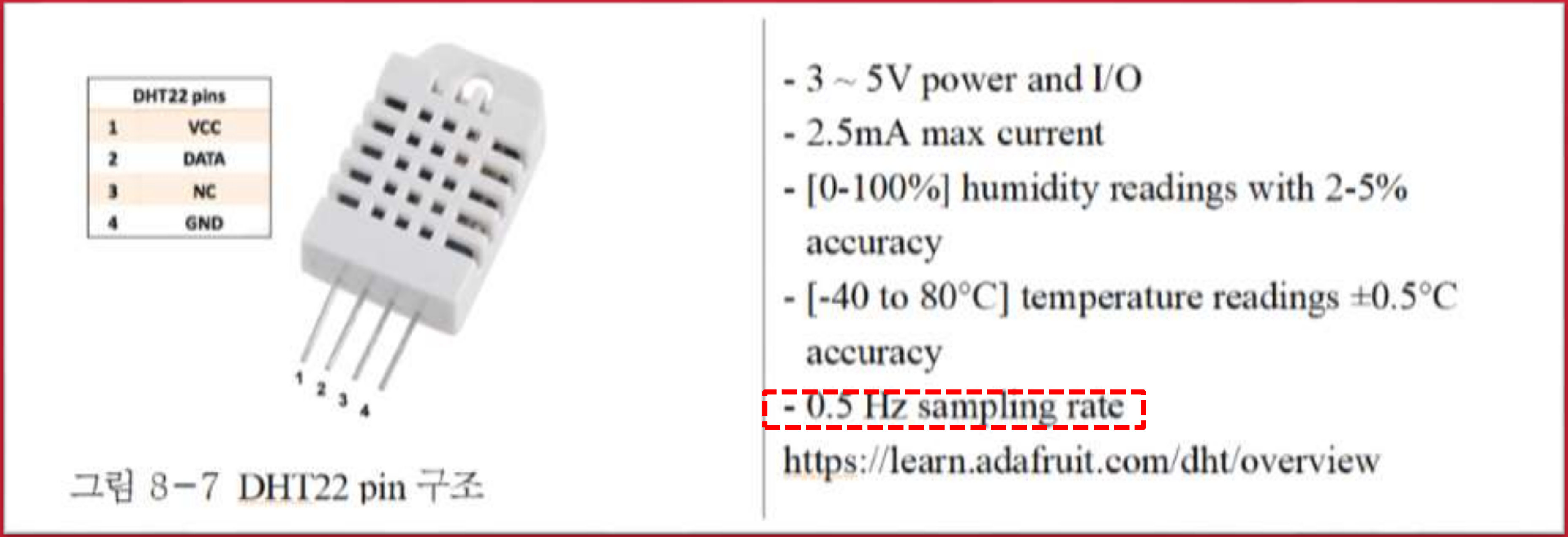


Multi-sensors

DHT22 + CdS



DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



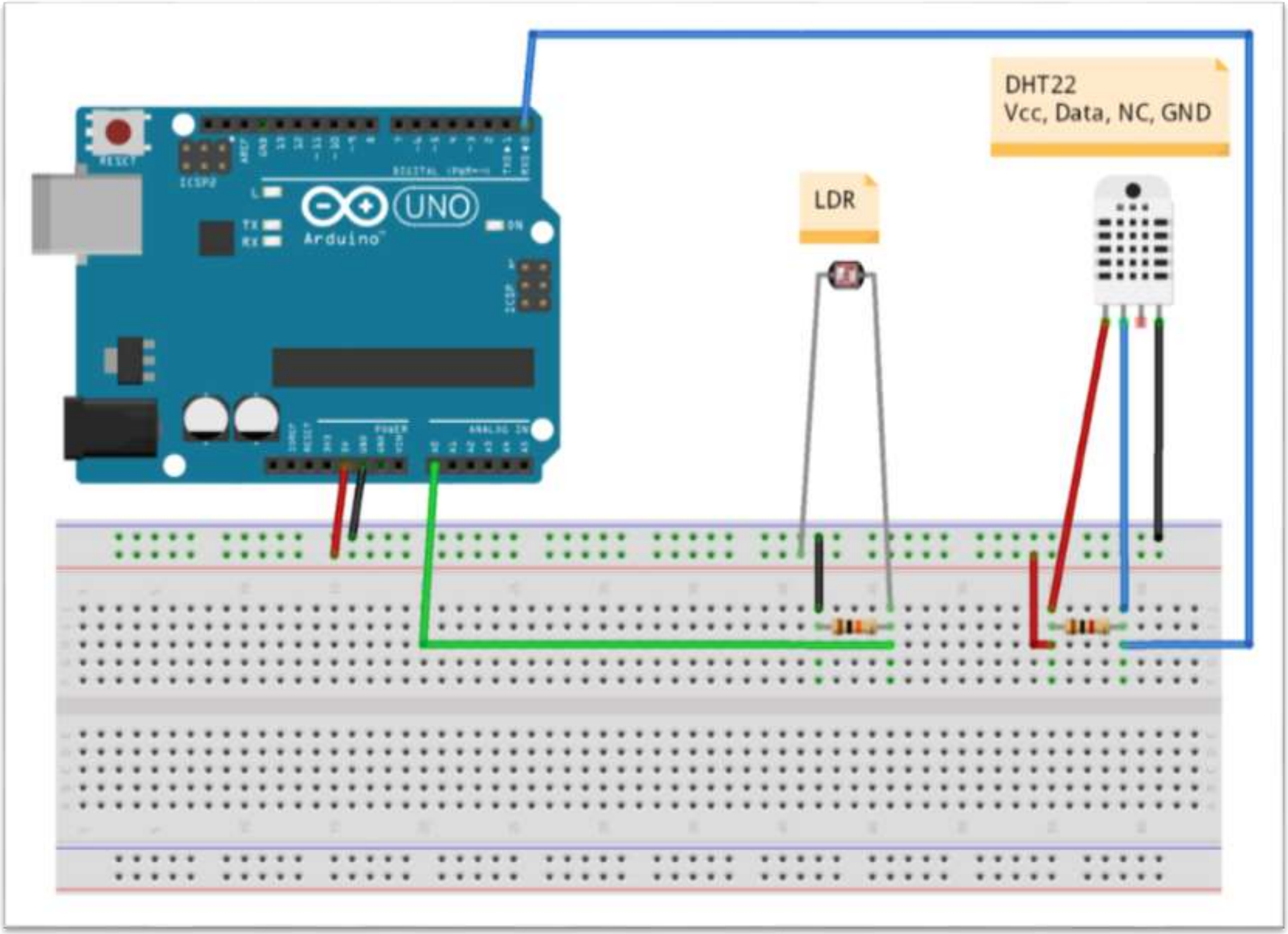
DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

그림 8-7 DHT22 pin 구조

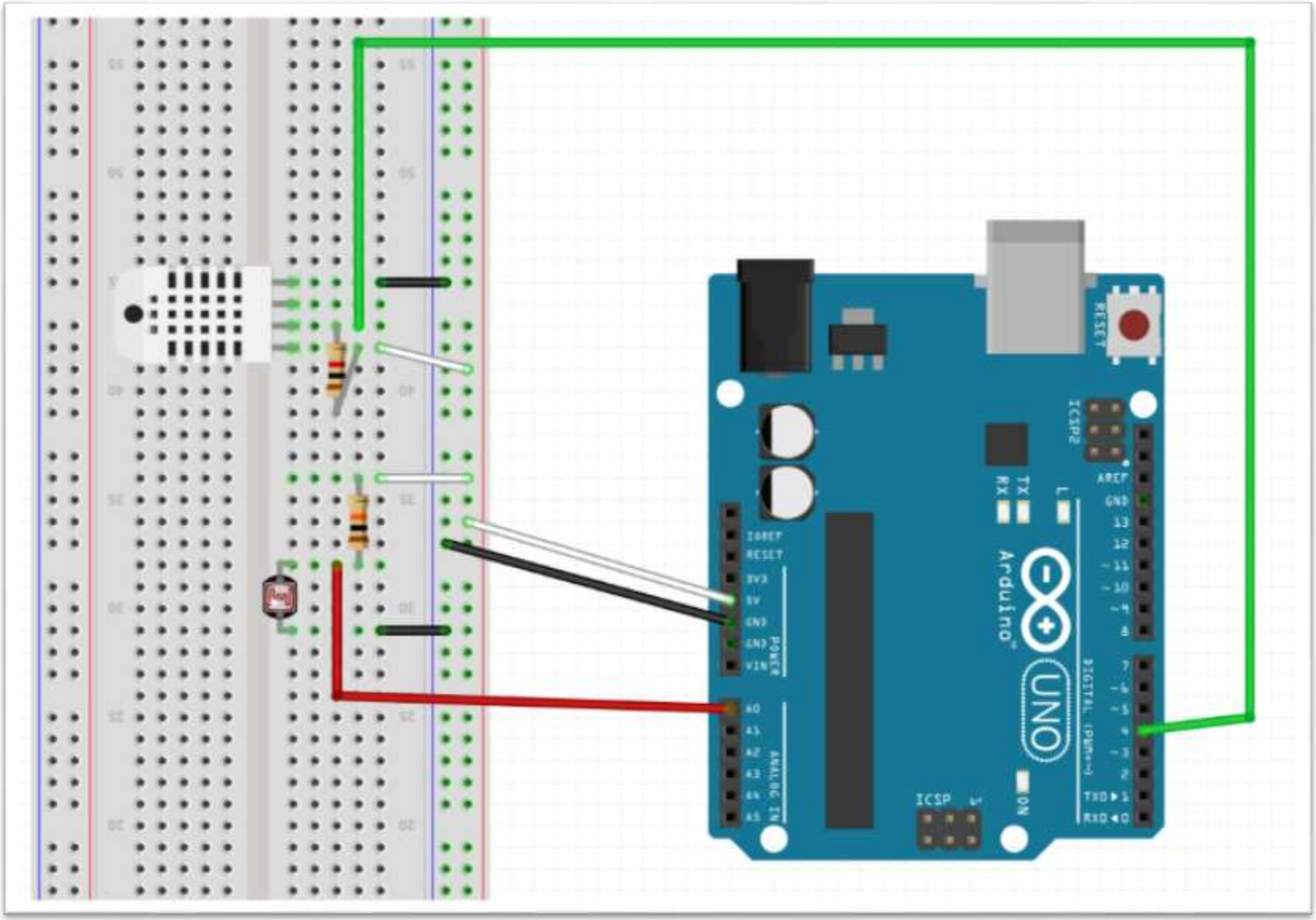
- 3 ~ 5V power and I/O
- 2.5mA max current
- [0-100%] humidity readings with 2-5% accuracy
- [-40 to 80°C] temperature readings ±0.5°C accuracy
- 0.5 Hz sampling rate

https://learn.adafruit.com/dht/overview

A5.7 DHT22 + CdS streaming project

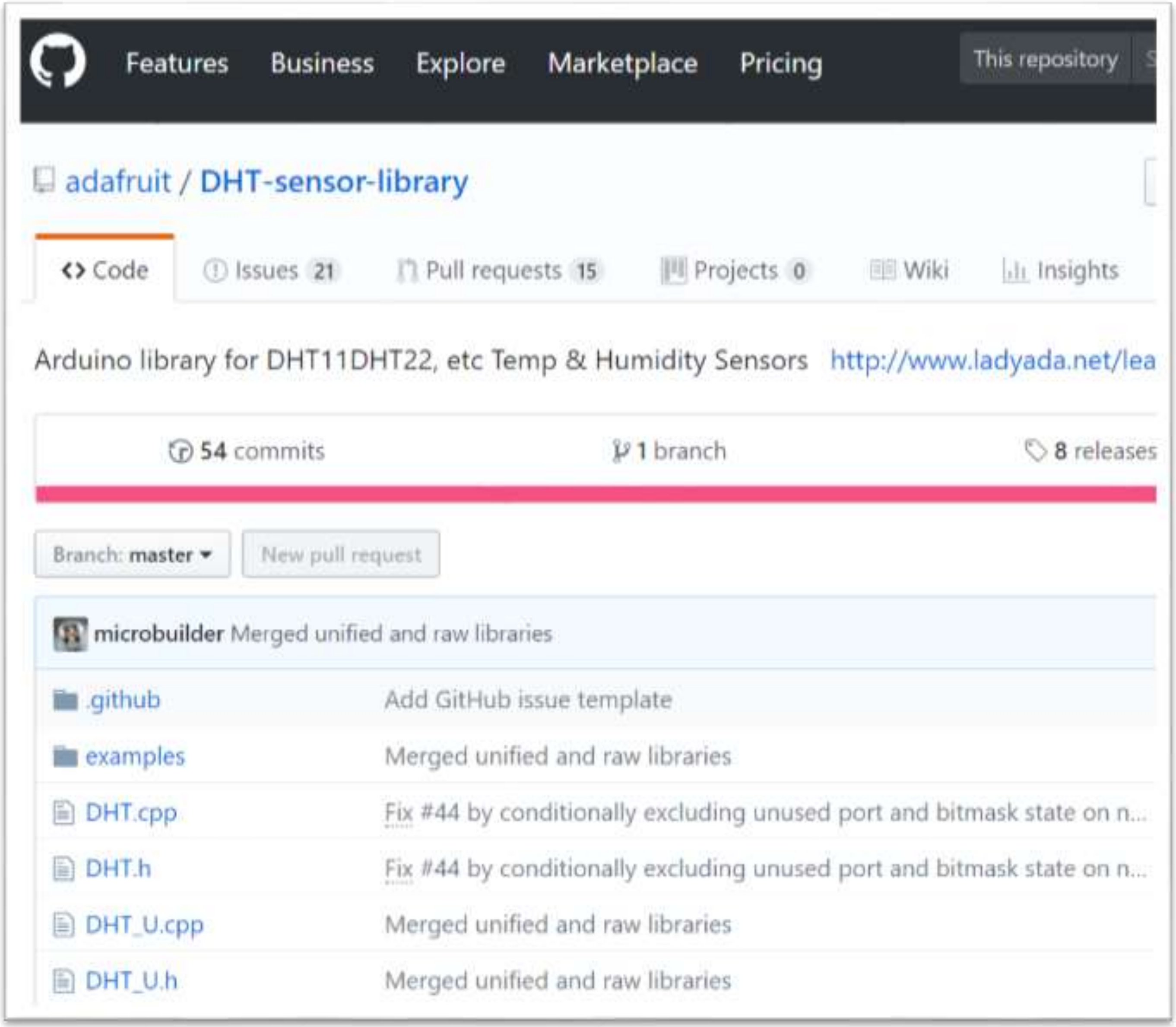


A5.7.1 DHT22 + CdS circuit



DHT22 + 1 kΩ,   CdS + 10 k Ω

A5.7.2 DHT22 + CdS : DHT library



adafruit / DHT-sensor-library

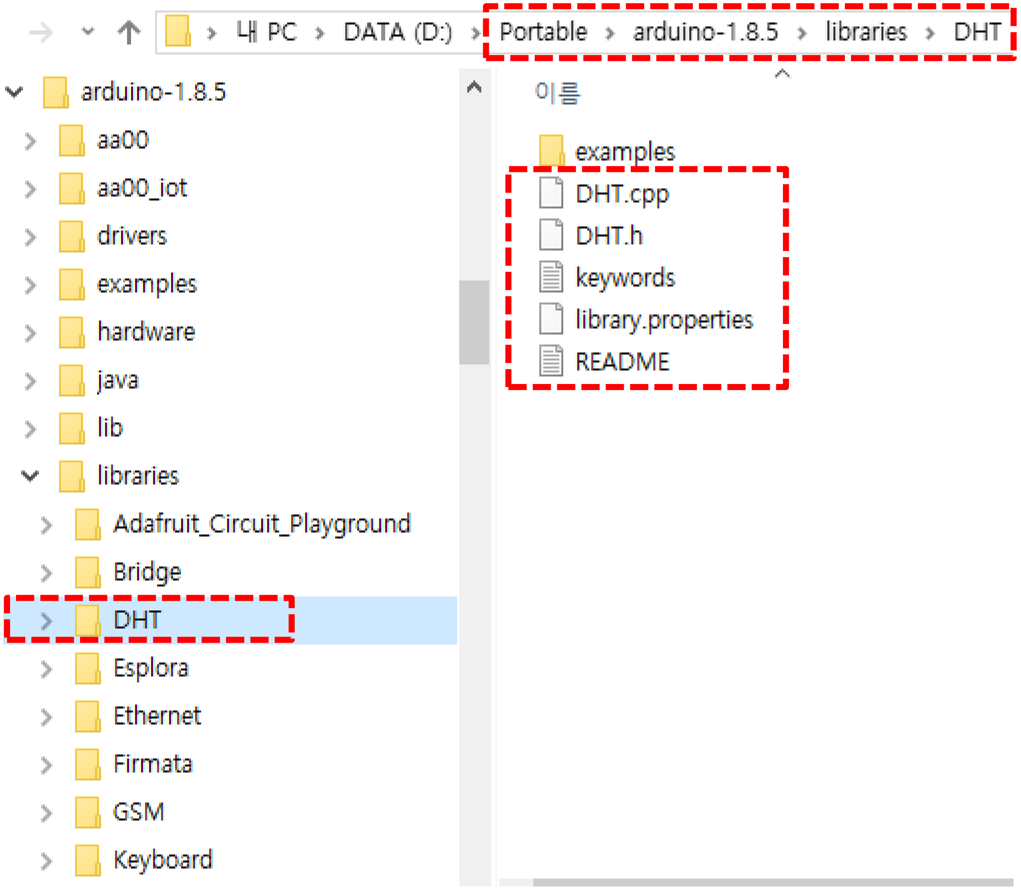
Arduino library for DHT11DHT22, etc Temp & Humidity Sensors http://www.ladyada.net/lea

54 commits 1 branch 8 releases

microbuilder Merged unified and raw libraries

.github	Add GitHub issue template
examples	Merged unified and raw libraries
DHT.cpp	Fix #44 by conditionally excluding unused port and bitmask state on n...
DHT.h	Fix #44 by conditionally excluding unused port and bitmask state on n...
DHT_U.cpp	Merged unified and raw libraries
DHT_U.h	Merged unified and raw libraries

A5.7.3 DHT22 + CdS : DHT library



내 PC › DATA (D:) › Portable › arduino-1.8.5 › libraries › DHT

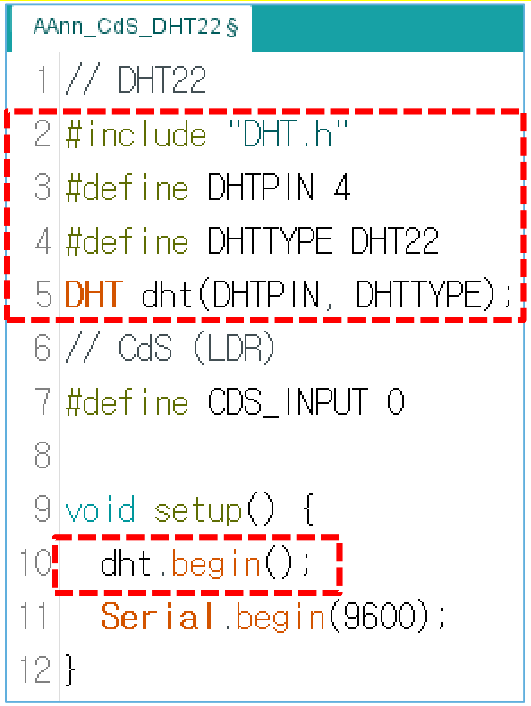
- arduino-1.8.5
  - aa00
  - aa00_iot
  - drivers
  - examples
  - hardware
  - java
  - lib
  - libraries
    - Adafruit_Circuit_Playground
    - Bridge
    - DHT
    - Esplora
    - Ethernet
    - Firmata
    - GSM
    - Keyboard

이름

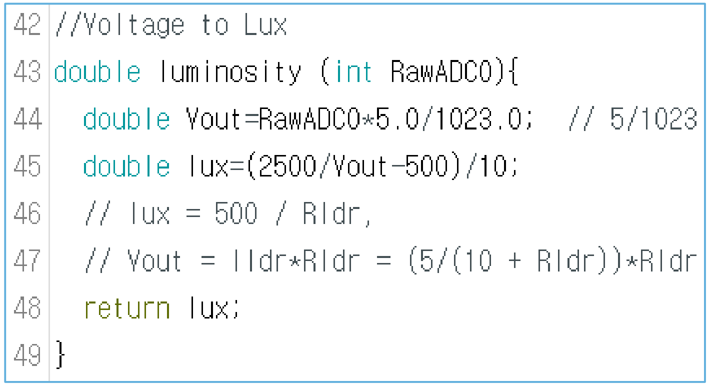
- examples
- DHT.cpp
- DHT.h
- keywords
- library.properties
- README

A5.7.4 DHT22 + CdS : circuit

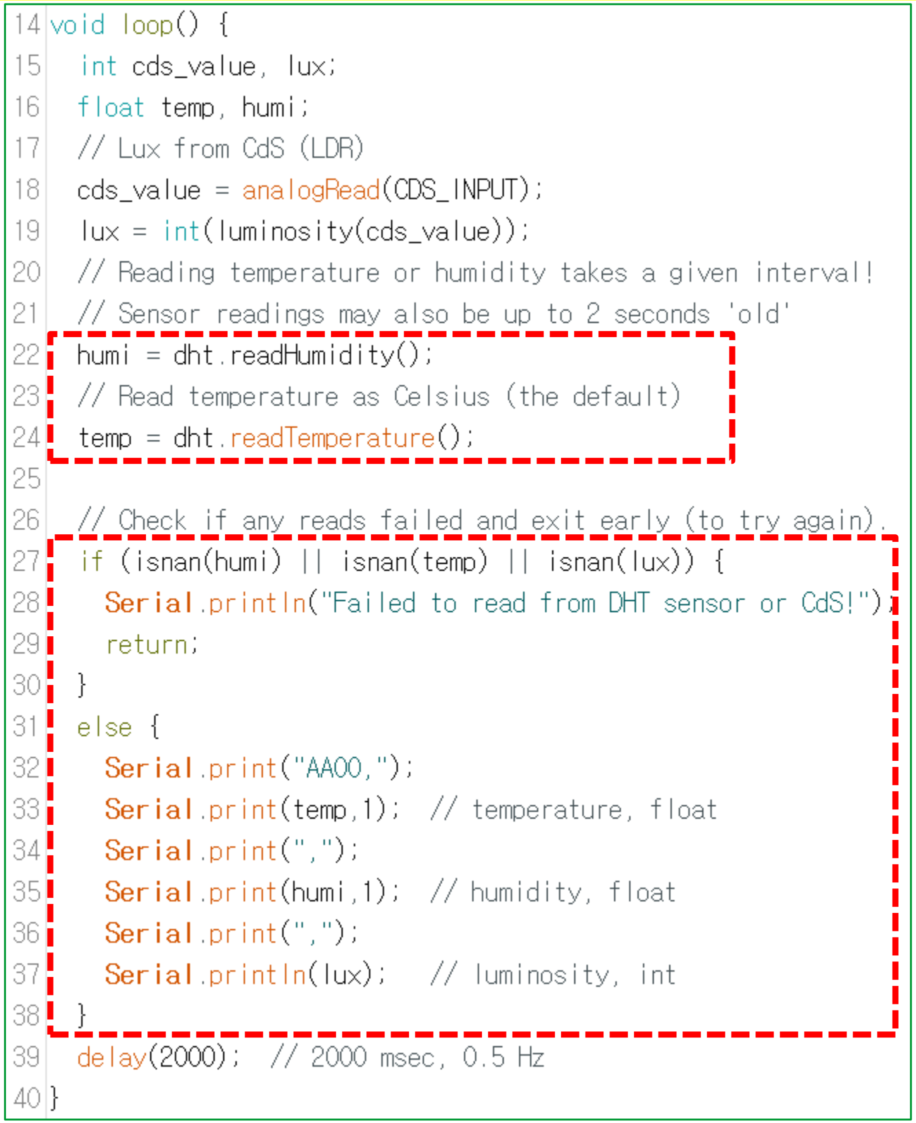
[1] Arduino code: AAnn_CdS_DHT22.ino



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

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



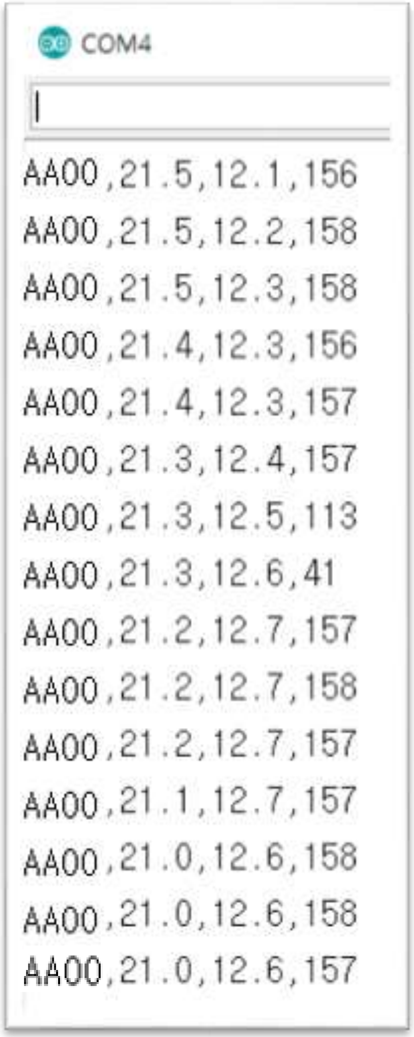
```
//Voltage to Lux
double luminosity (int RawADC0){
  double Vout=RawADC0*5.0/1023.0;  // 5/1023
  double lux=(2500/Vout-500)/10;
  // lux = 500 / Rldr,
  // Vout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
  return lux;
}
```



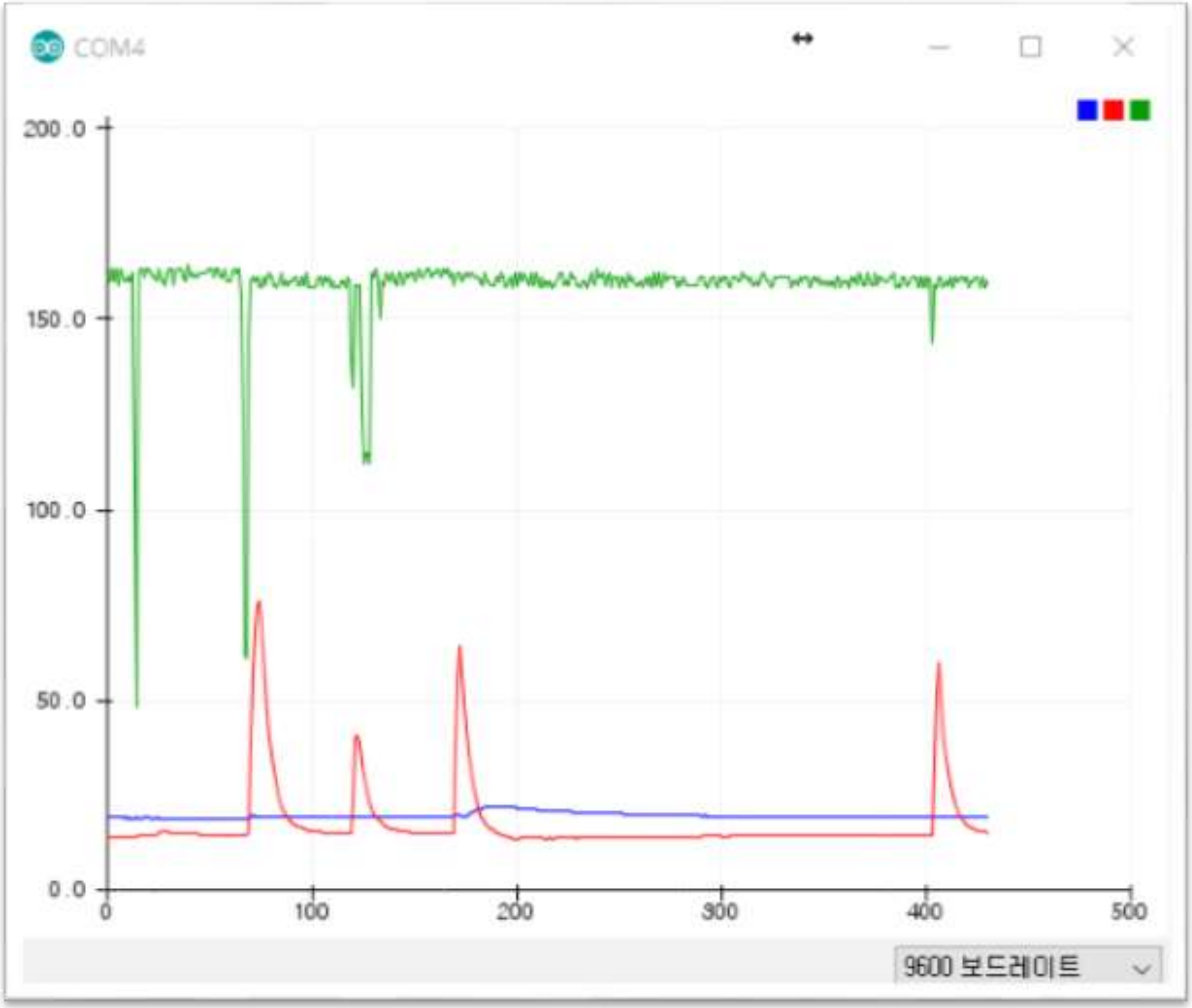
```
void loop() {
  int cds_value, lux;
  float temp, humi;
  // Lux from CdS (LDR)
  cds_value = analogRead(CDS_INPUT);
  lux = int(luminosity(cds_value));
  // Reading temperature or humidity takes a given interval!
  // Sensor readings may also be up to 2 seconds 'old'
  humi = dht.readHumidity();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

  // Check if any reads failed and exit early (to try again).
  if (isnan(humi) || isnan(temp) || isnan(lux)) {
    Serial.println("Failed to read from DHT sensor or CdS!");
    return;
  }
  else {
    Serial.print("AA00,");
    Serial.print(temp,1);  // temperature, float
    Serial.print(",");
    Serial.print(humi,1);  // humidity, float
    Serial.print(",");
    Serial.println(lux);   // luminosity, int
  }
  delay(2000);  // 2000 msec, 0.5 Hz
}
```

A5.7.5 DHT22 + CdS : Serial monitor

[1] Arduino code: AAnn_CdS_DHT22.ino

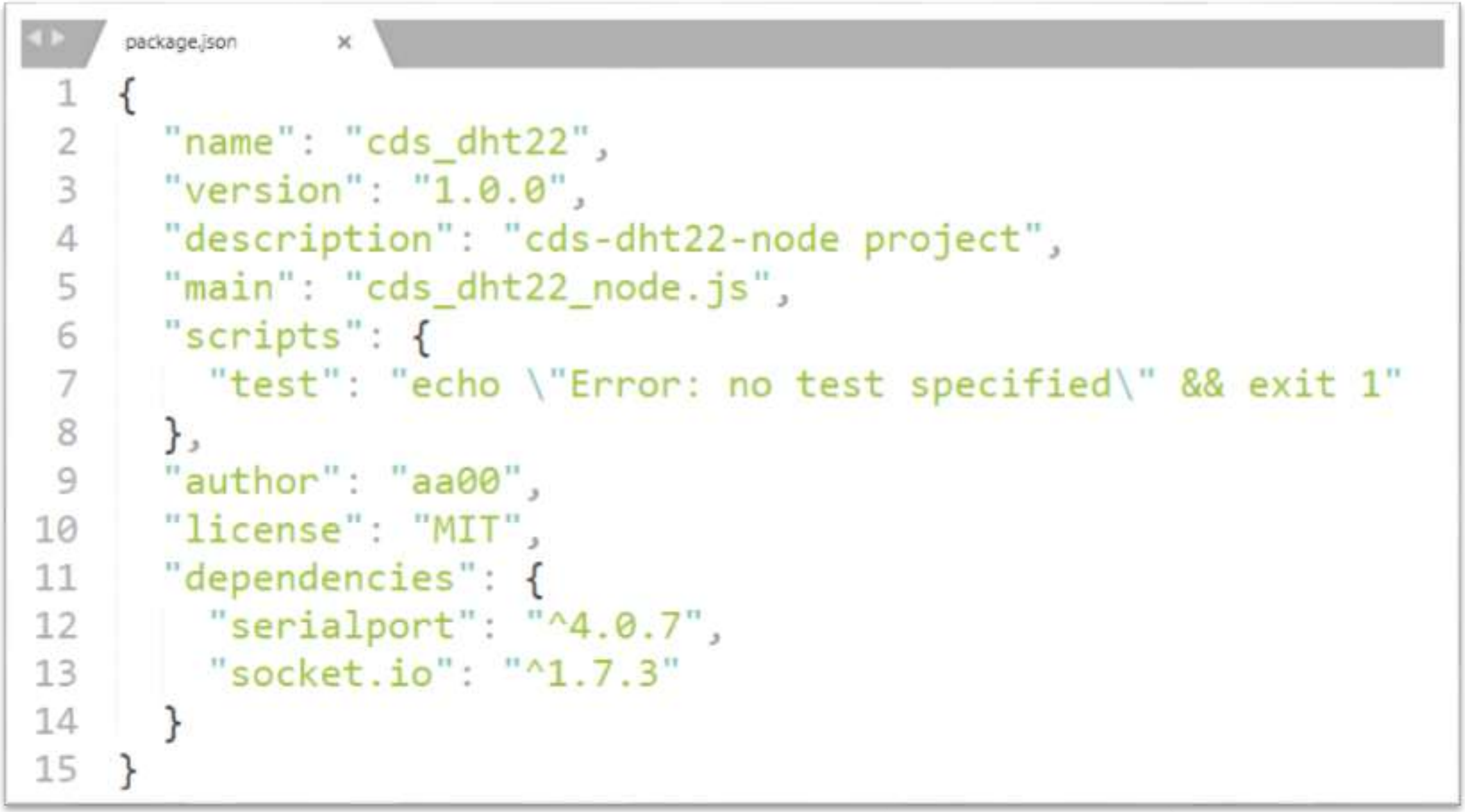


```
AA00,21.5,12.1,156
AA00,21.5,12.2,158
AA00,21.5,12.3,158
AA00,21.4,12.3,156
AA00,21.4,12.3,157
AA00,21.3,12.4,157
AA00,21.3,12.5,113
AA00,21.3,12.6,41
AA00,21.2,12.7,157
AA00,21.2,12.7,158
AA00,21.2,12.7,157
AA00,21.1,12.7,157
AA00,21.0,12.6,158
AA00,21.0,12.6,158
AA00,21.0,12.6,157
```



A5.7.6 DHT22 + CdS + Node.js

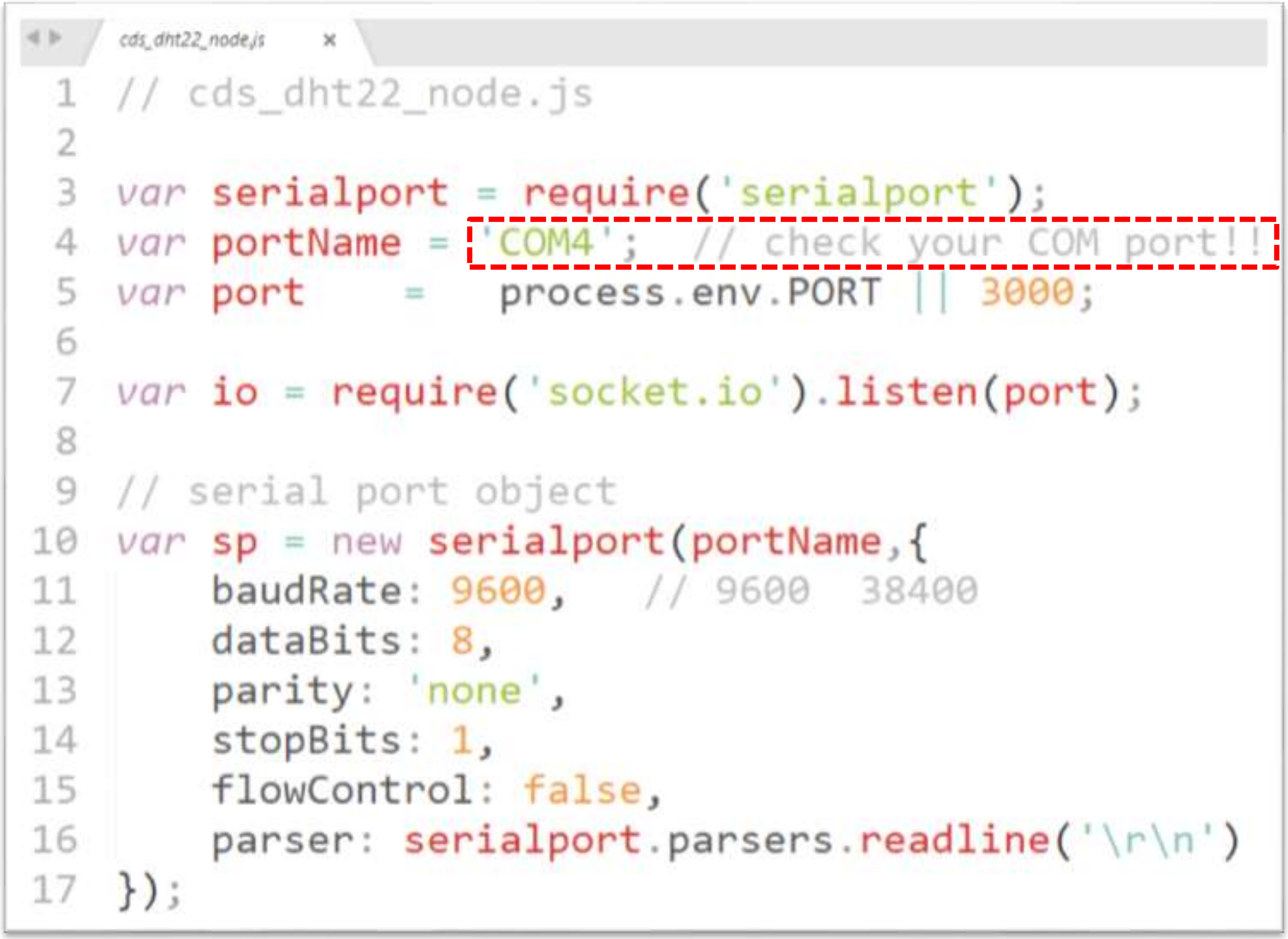
[2.1] NodeJS project: "cds-dht22-node project“ → package.json



```
{
  "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": {
    "serialport": "^4.0.7",
    "socket.io": "^1.7.3"
  }
}
```

A5.7.7 DHT22 + CdS + Node.js

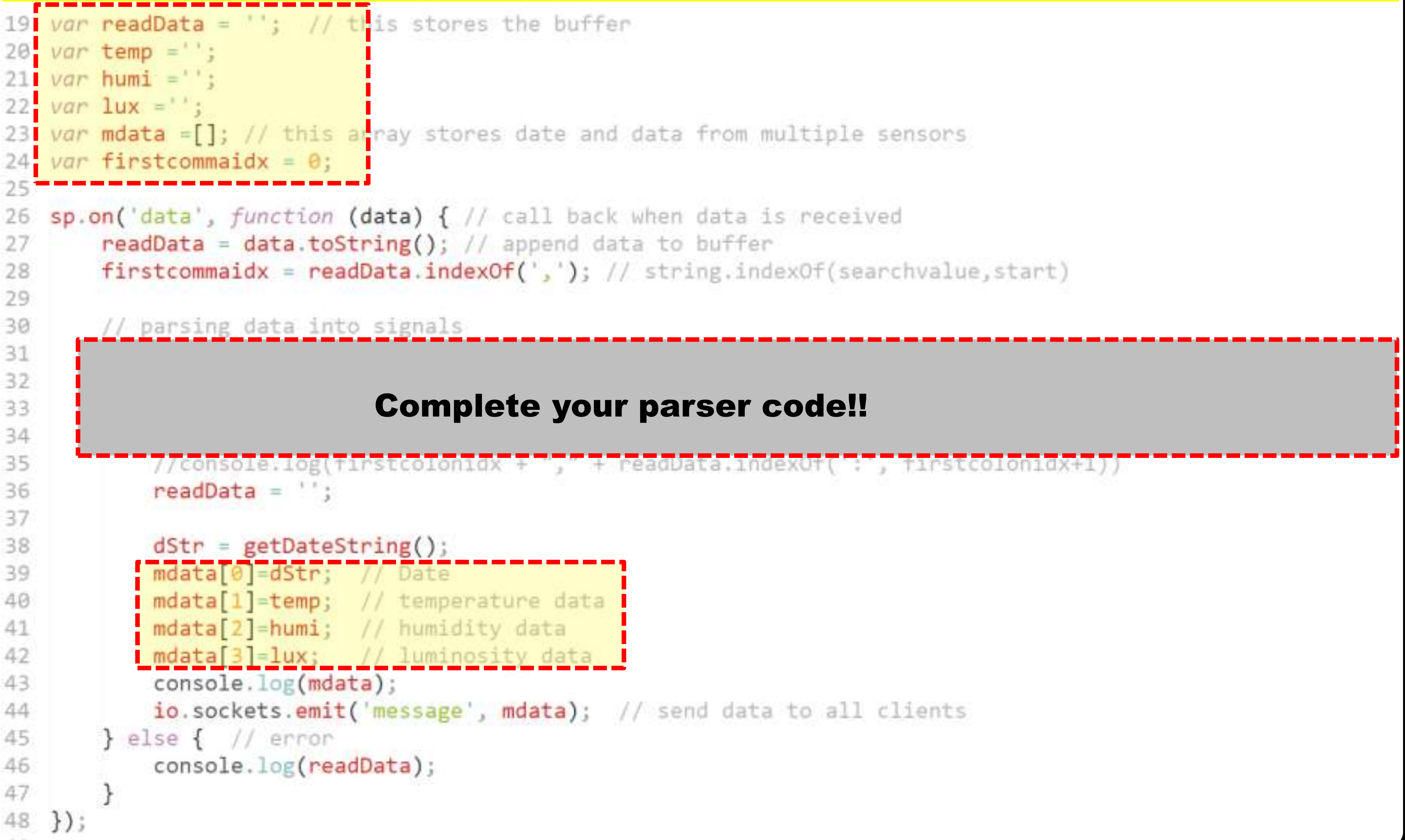
[2.2] NodeJS code: cds_dht22_node.js (← cds_tmp36_node.js)



```
// cds_dht22_node.js

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

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

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

A5.7.8 DHT22 + CdS + Node.js

[2.3] NodeJS code: cds_dht22_node.js (Complete your parser code)



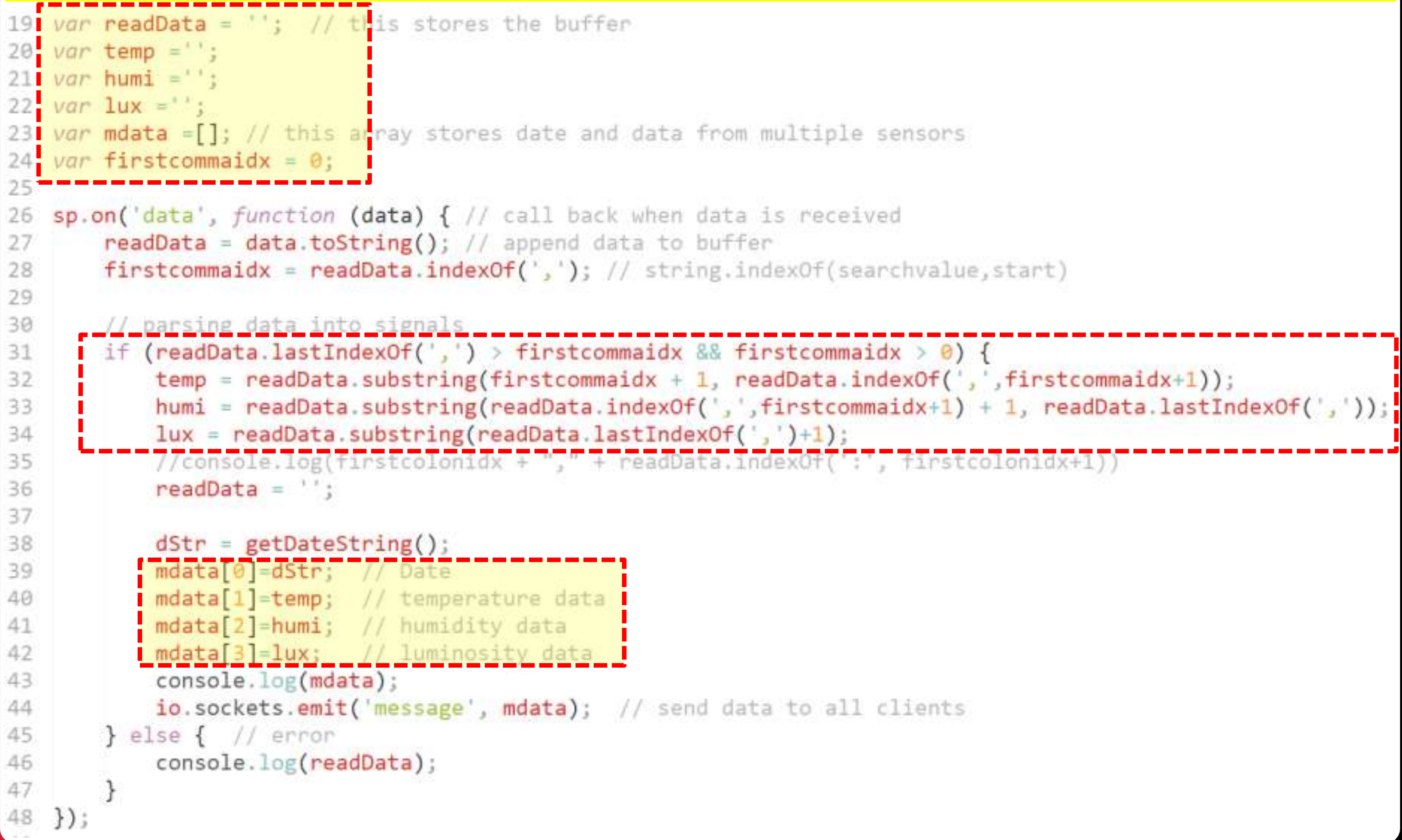
```
var readData = '';  // this stores the buffer
var temp ='';
var humi ='';
var lux ='';
var mdata =[]; // this array stores date and data from multiple sensors
var firstcommaidx = 0;

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

    // parsing data into signals
```

Complete your parser code!!

```
        //console.log(firstcolonidx + "," + readData.indexOf(':', firstcolonidx+1))
        readData = '';

        dStr = getDateString();
        mdata[0]=dStr;   // Date
        mdata[1]=temp;   // temperature data
        mdata[2]=humi;   // humidity data
        mdata[3]=lux;    // luminosity data
        console.log(mdata);
        io.sockets.emit('message', mdata);  // send data to all clients
    } else {  // error
        console.log(readData);
    }
});
```

A5.7.9 DHT22 + CdS + Node.js

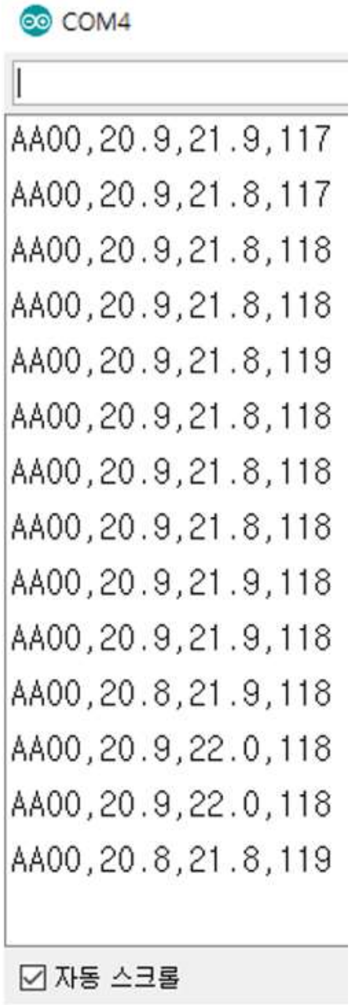
[2.3] NodeJS code: cds_dht22_node.js (Complete your parser code)



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

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

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

        dStr = getDateString();
        mdata[0]=dStr;   // Date
        mdata[1]=temp;   // temperature data
        mdata[2]=humi;   // humidity data
        mdata[3]=lux;    // luminosity data
        console.log(mdata);
        io.sockets.emit('message', mdata);  // send data to all clients
    } else {  // error
        console.log(readData);
    }
});
```

A5.7.10 DHT22 + CdS + Node.js

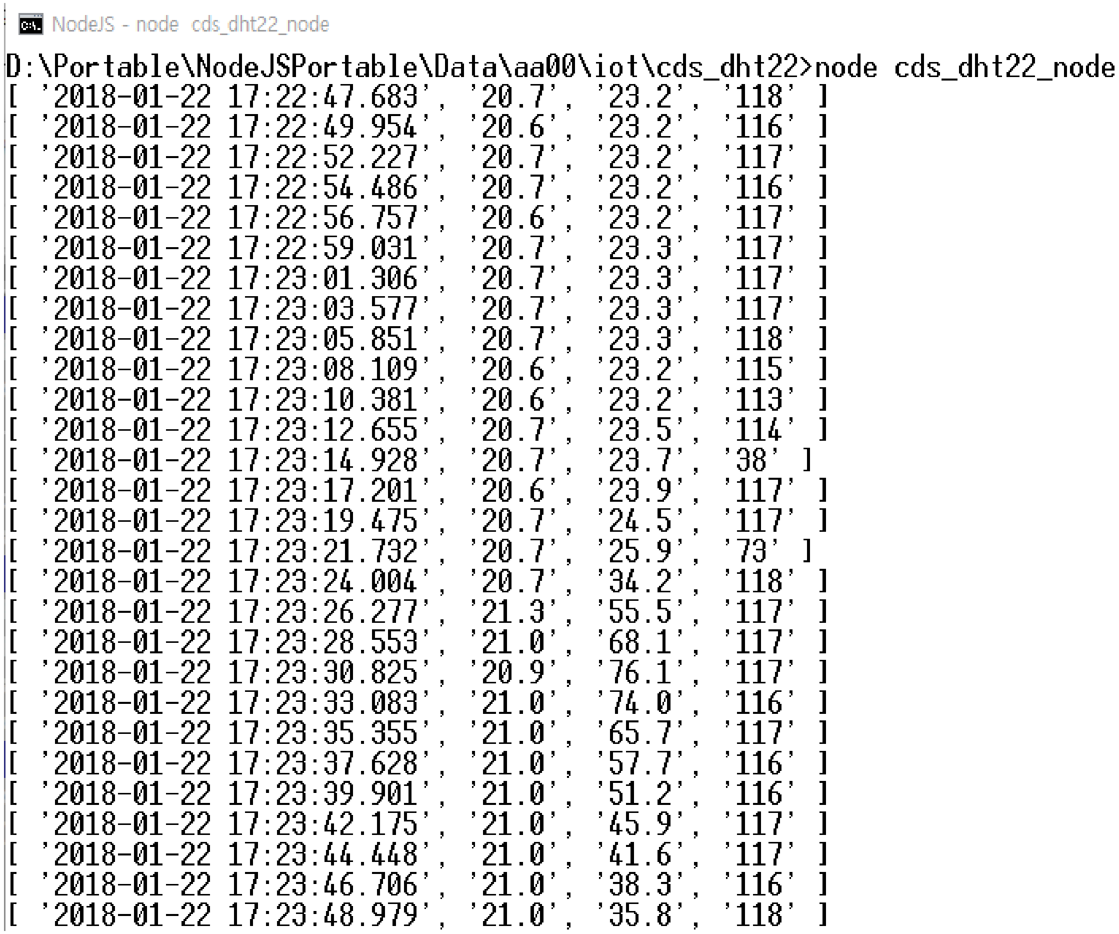
[3] Result: Parsed streaming data from dht22 & CdS (Run in Node cmd)



COM4

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

☑ 자동 스크롤

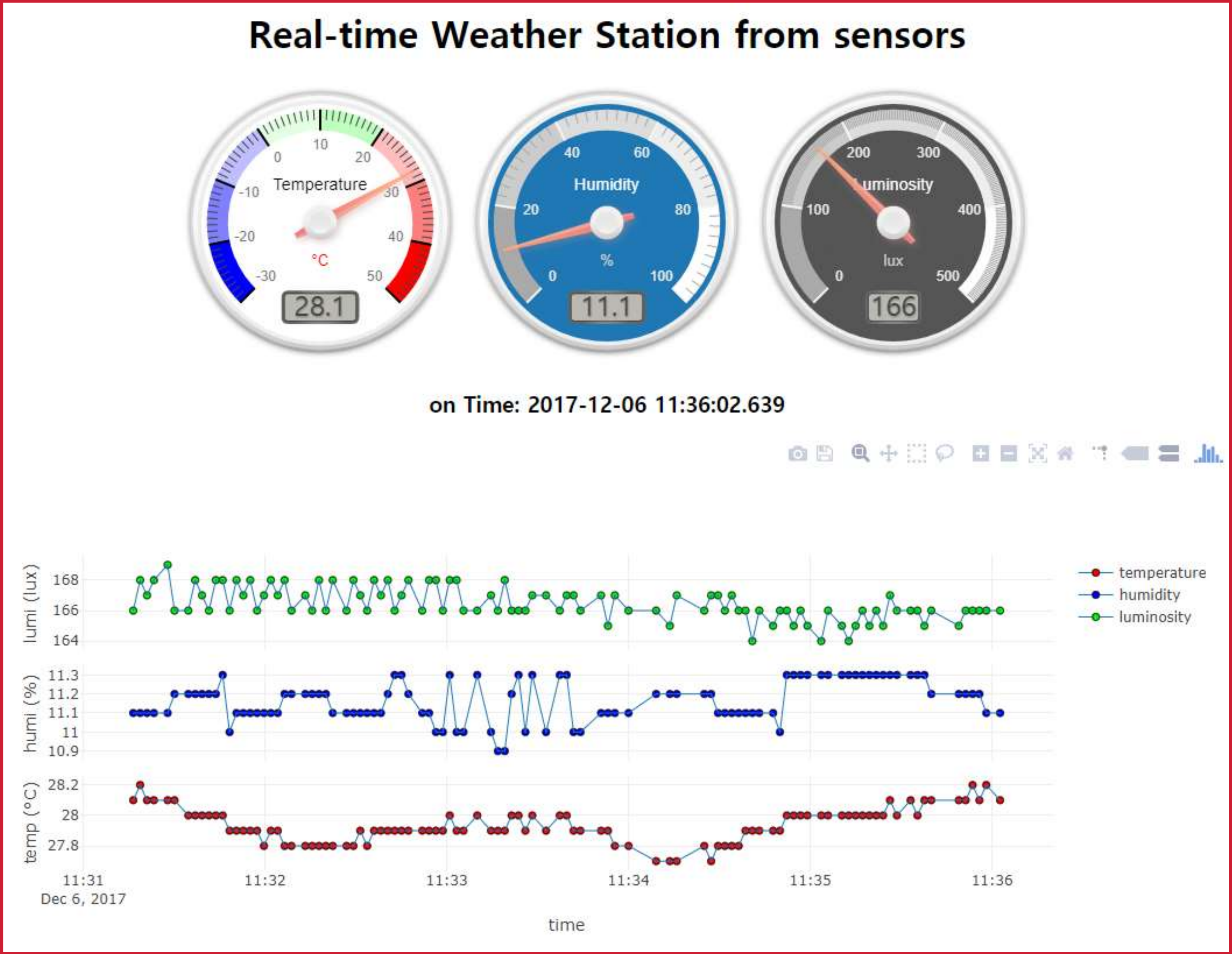


NodeJS - node cds_dht22_node

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

Save as
AAnn_cds_dht22_data.png

WEB client： client_cds_dht22.html

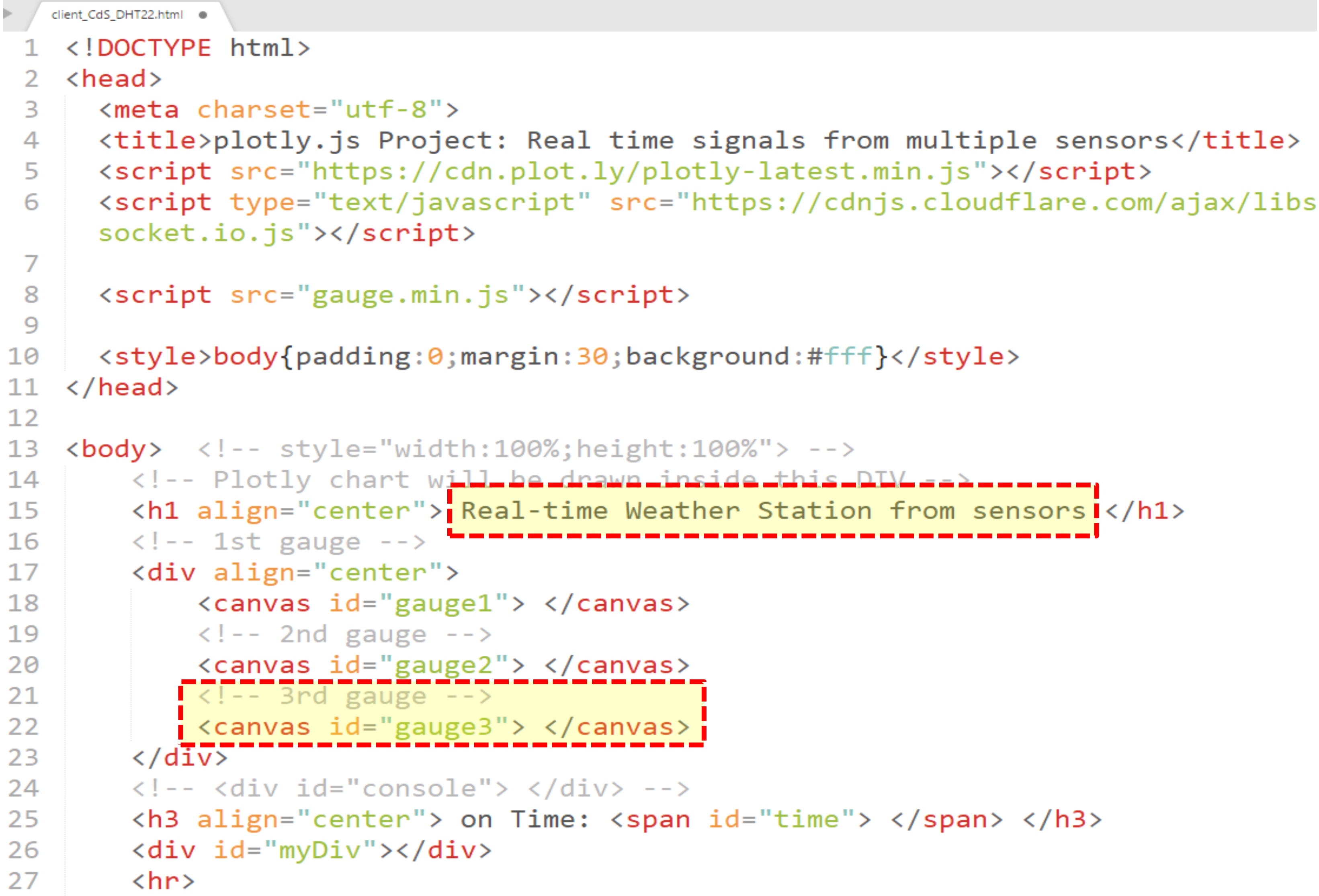


Real-time Weather Station from sensors

on Time: 2017-12-06 11:36:02.639

A5.8.1 DHT22 + CdS + Node.js

[4.1] WEB client: client_cds_dht22.html

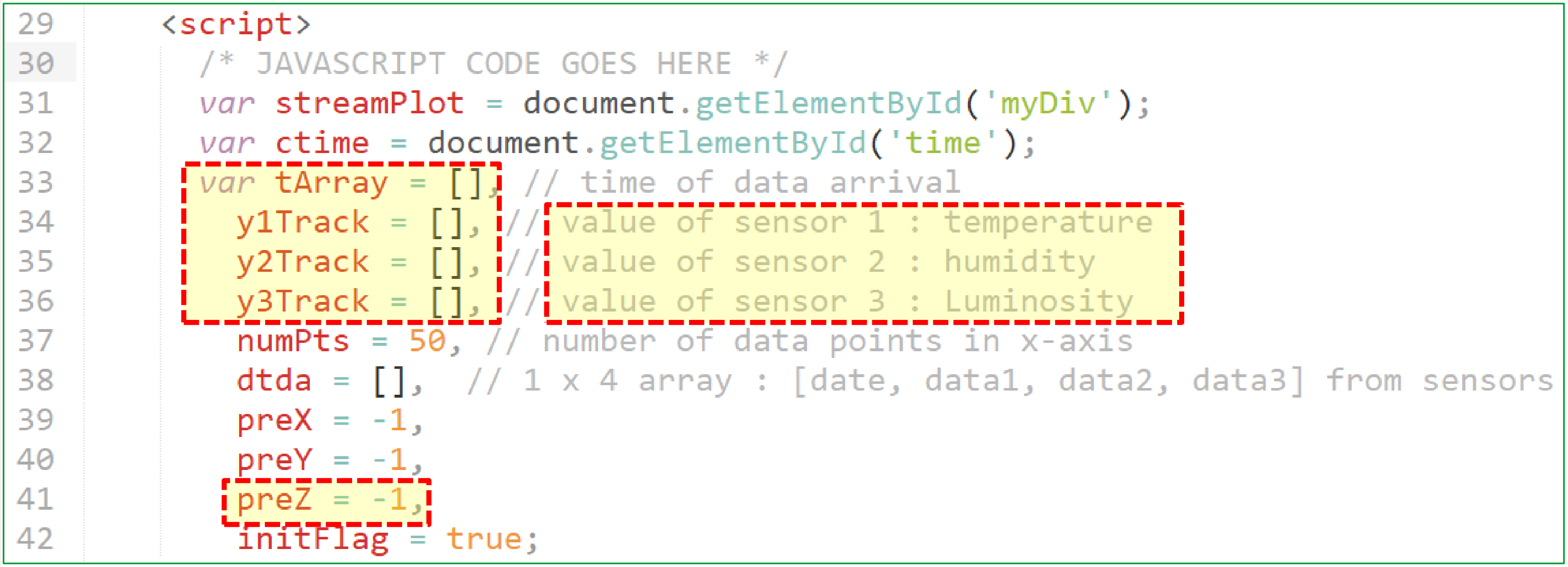


client_CdS_DHT22.html

```html
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <title>plotly.js Project: Real time signals from multiple 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.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 Weather Station from sensors </h1>
    <!-- 1st gauge -->
    <div align="center">
        <canvas id="gauge1"> </canvas>
        <!-- 2nd gauge -->
        <canvas id="gauge2"> </canvas>
        <!-- 3rd gauge -->
        <canvas id="gauge3"> </canvas>
    </div>
    <!-- <div id="console"> </div> -->
    <h3 align="center"> on Time: <span id="time"> </span> </h3>
    <div id="myDiv"></div>
    <hr>
```

A5.8.2 DHT22 + CdS + Node.js

[4.2] WEB client: client_cds_dht22.html



```
<script>
  /* JAVASCRIPT CODE GOES HERE */
  var streamPlot = document.getElementById('myDiv');
  var ctime = document.getElementById('time');
  var tArray = [], // time of data arrival
    y1Track = [], // value of sensor 1 : temperature
    y2Track = [], // value of sensor 2 : humidity
    y3Track = [], // value of sensor 3 : Luminosity
    numPts = 50, // number of data points in x-axis
    dtda = [],  // 1 x 4 array : [date, data1, data2, data3] from sensors
    preX = -1,
    preY = -1,
    preZ = -1,
    initFlag = true;
```

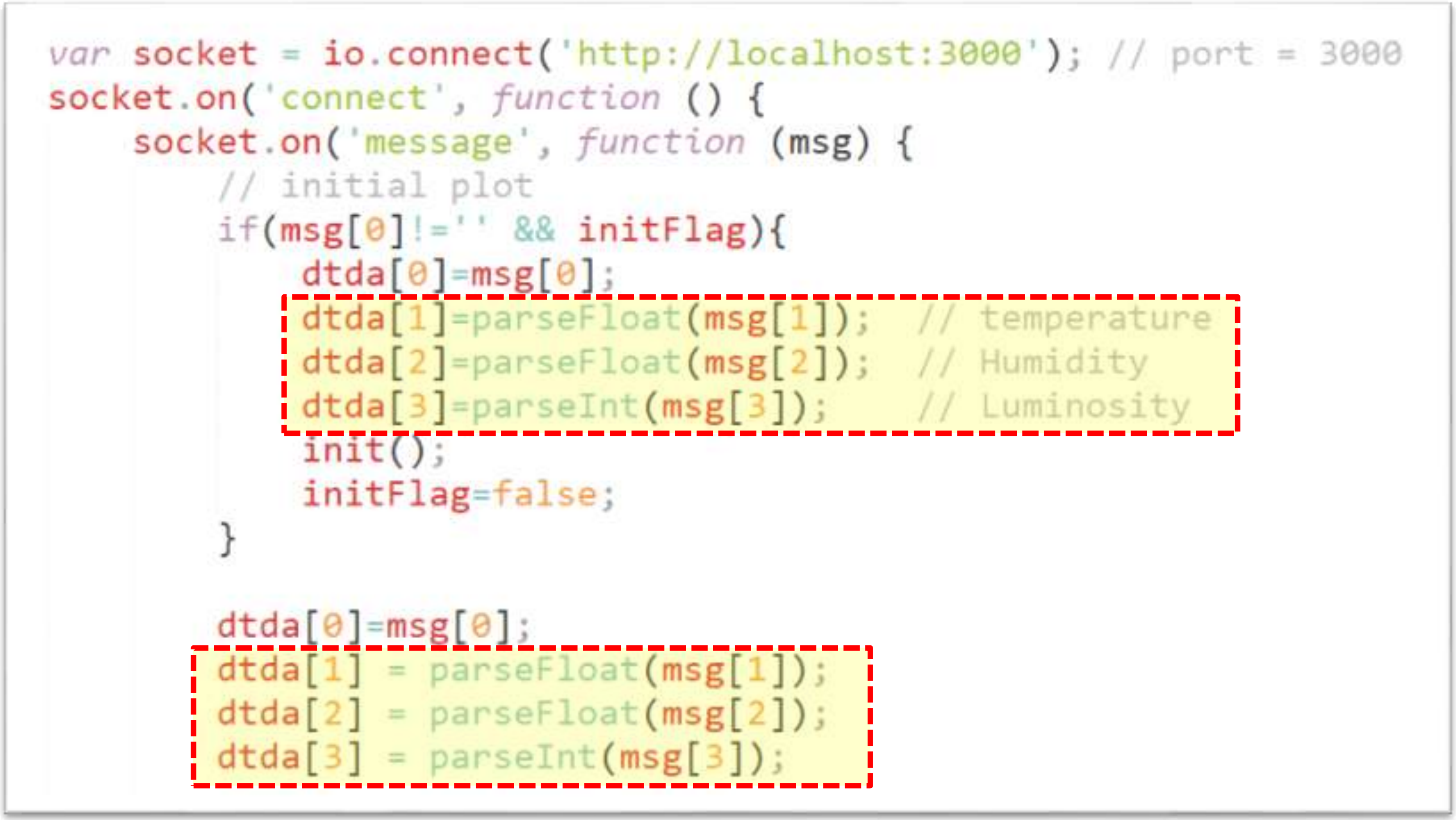
Check points: tArray

xTrack → y1Track, yTrack → y2Track

& add y3Track & preZ

A5.8.3 DHT22 + CdS + Node.js

[4.3] WEB client: client_cds_dht22.html



```
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]=parseFloat(msg[1]);  // temperature
            dtda[2]=parseFloat(msg[2]);  // Humidity
            dtda[3]=parseInt(msg[3]);    // Luminosity
            init();
            initFlag=false;
        }

        dtda[0]=msg[0];
        dtda[1] = parseFloat(msg[1]);
        dtda[2] = parseFloat(msg[2]);
        dtda[3] = parseInt(msg[3]);
```

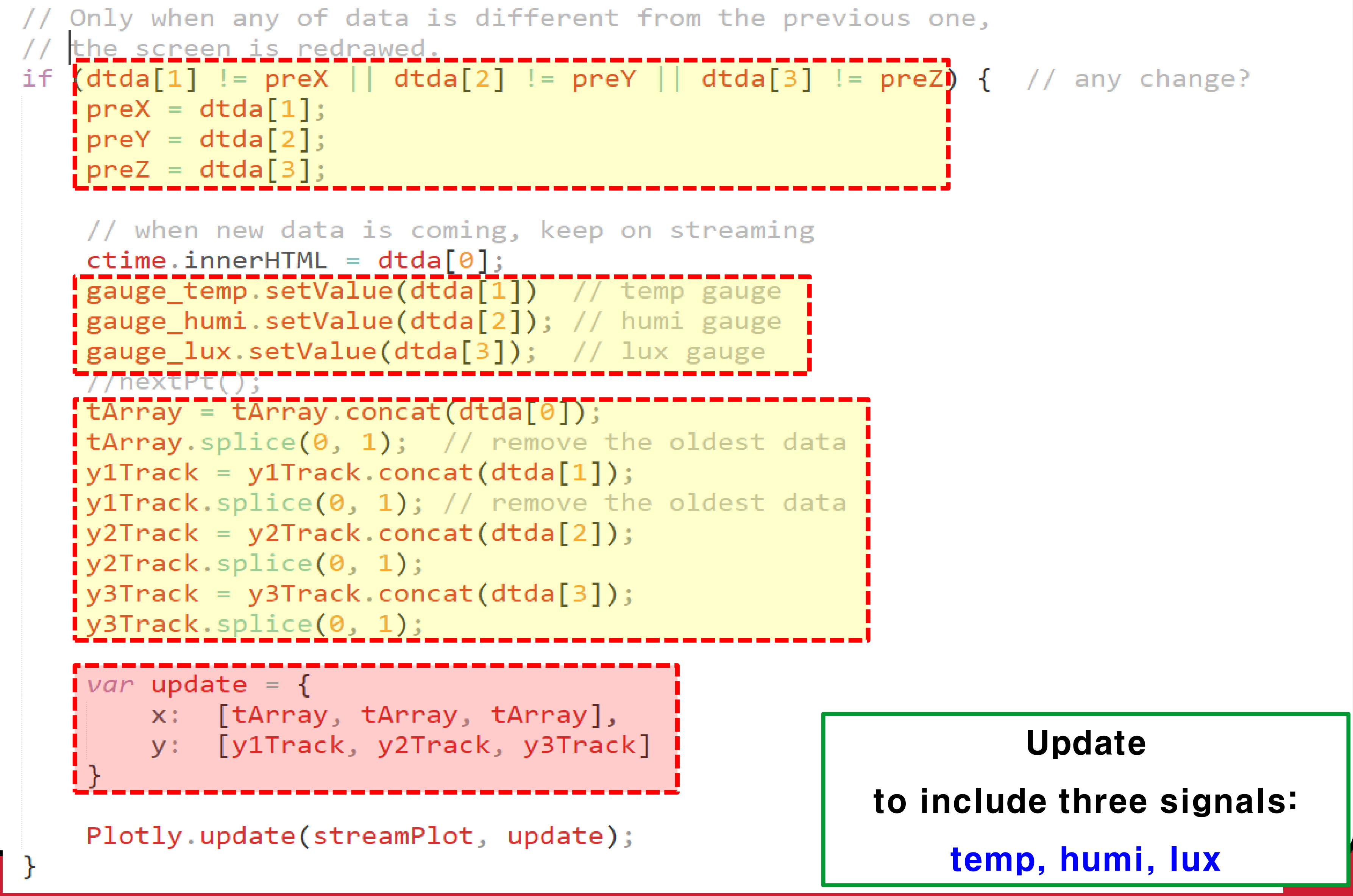
Update

to include three signals:

temp, humi, lux

A5.8.4 DHT22 + CdS + Node.js

[4.4] WEB client: client_cds_dht22.html



```
// Only when any of data is different from the previous one,
// the screen is redrawed.
if (dtda[1] != preX || dtda[2] != preY || dtda[3] != preZ) {  // any change?
    preX = dtda[1];
    preY = dtda[2];
    preZ = dtda[3];

    // when new data is coming, keep on streaming
    ctime.innerHTML = dtda[0];
    gauge_temp.setValue(dtda[1])  // temp gauge
    gauge_humi.setValue(dtda[2]); // humi gauge
    gauge_lux.setValue(dtda[3]);  // lux gauge
    //nextPt();
    tArray = tArray.concat(dtda[0]);
    tArray.splice(0, 1);  // remove the oldest data
    y1Track = y1Track.concat(dtda[1]);
    y1Track.splice(0, 1); // remove the oldest data
    y2Track = y2Track.concat(dtda[2]);
    y2Track.splice(0, 1);
    y3Track = y3Track.concat(dtda[3]);
    y3Track.splice(0, 1);

    var update = {
        x:  [tArray, tArray, tArray],
        y:  [y1Track, y2Track, y3Track]
    }

    Plotly.update(streamPlot, update);
}
```

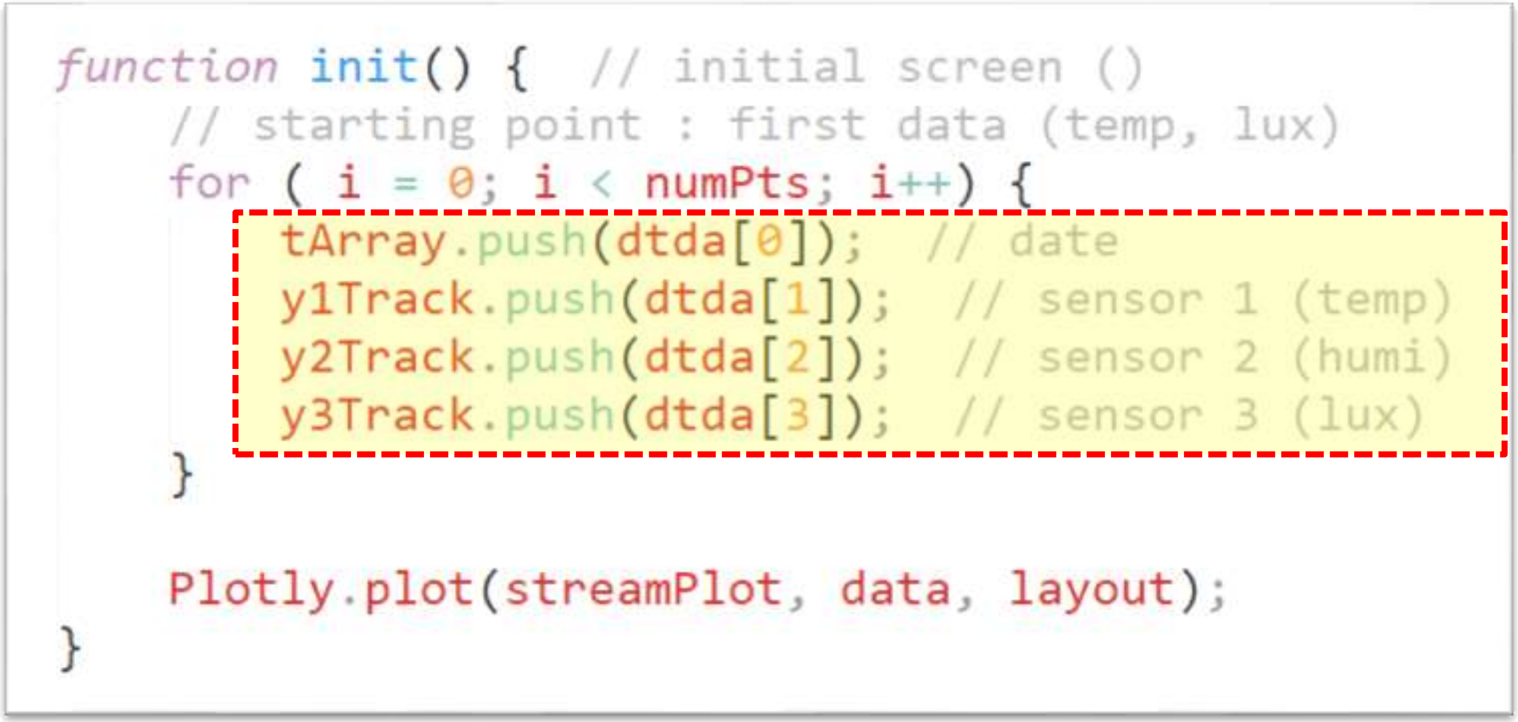
Update

to include three signals:

temp, humi, lux

A5.8.5 DHT22 + CdS + Node.js

[4.5] WEB client: client_dht22_ldr.html → init()



```
function init() {  // initial screen ()
    // starting point : first data (temp, lux)
    for ( i = 0; i < numPts; i++) {
        tArray.push(dtda[0]);  // date
        y1Track.push(dtda[1]);  // sensor 1 (temp)
        y2Track.push(dtda[2]);  // sensor 2 (humi)
        y3Track.push(dtda[3]);  // sensor 3 (lux)
    }

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

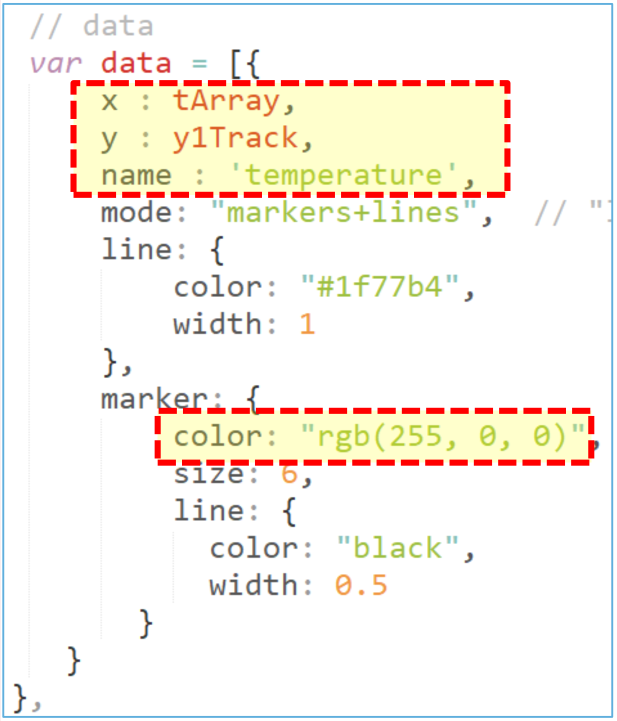
Update

to include three signals:

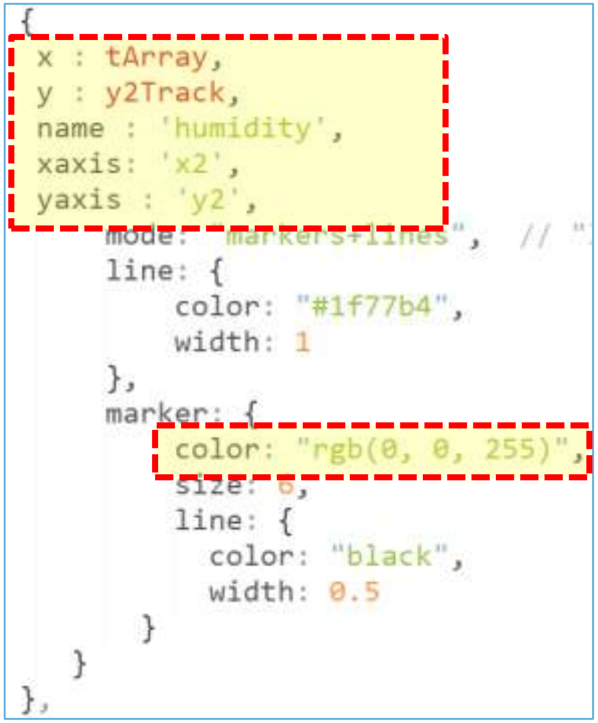
temp, humi, lux

A5.8.6 DHT22 + CdS + Node.js

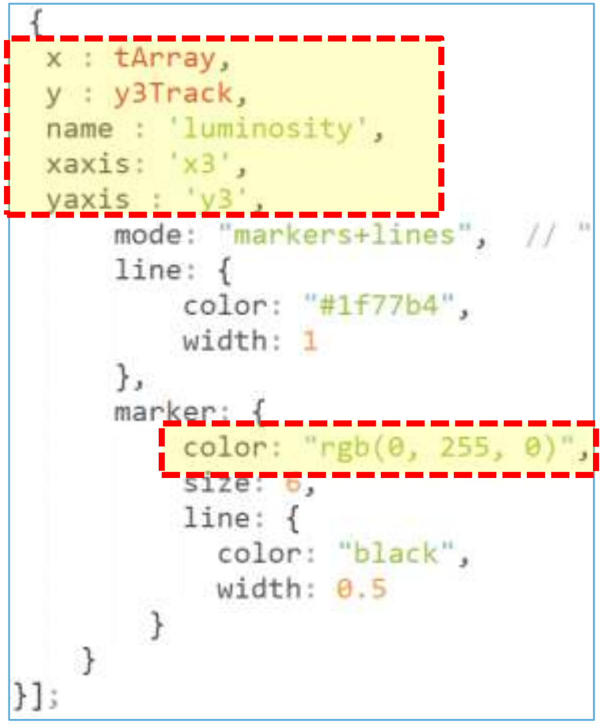
[4.6] WEB client: client_cds_dht22.html - data



```
// data
var data = [{
    x : tArray,
    y : y1Track,
    name : 'temperature',
    mode: "markers+lines",   // "
    line: {
        color: "#1f77b4",
        width: 1
    },
    marker: {
        color: "rgb(255, 0, 0)",
        size: 6,
        line: {
          color: "black",
          width: 0.5
        }
    }
},
```



```
{
    x : tArray,
    y : y2Track,
    name : 'humidity',
    xaxis: 'x2',
    yaxis : 'y2',
        mode: "markers+lines",   // "
        line: {
            color: "#1f77b4",
            width: 1
        },
        marker: {
            color: "rgb(0, 0, 255)",
            size: 6,
            line: {
              color: "black",
              width: 0.5
            }
        }
},
```



```
{
    x : tArray,
    y : y3Track,
    name : 'luminosity',
    xaxis: 'x3',
    yaxis : 'y3',
        mode: "markers+lines",   // "
        line: {
            color: "#1f77b4",
            width: 1
        },
        marker: {
            color: "rgb(0, 255, 0)",
            size: 6,
            line: {
              color: "black",
              width: 0.5
            }
        }
}];
```

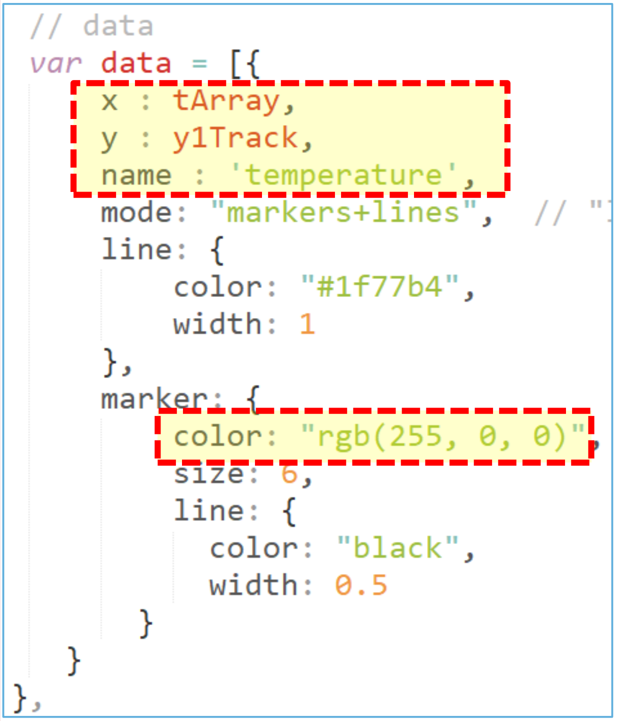
Update data

to include three signals:

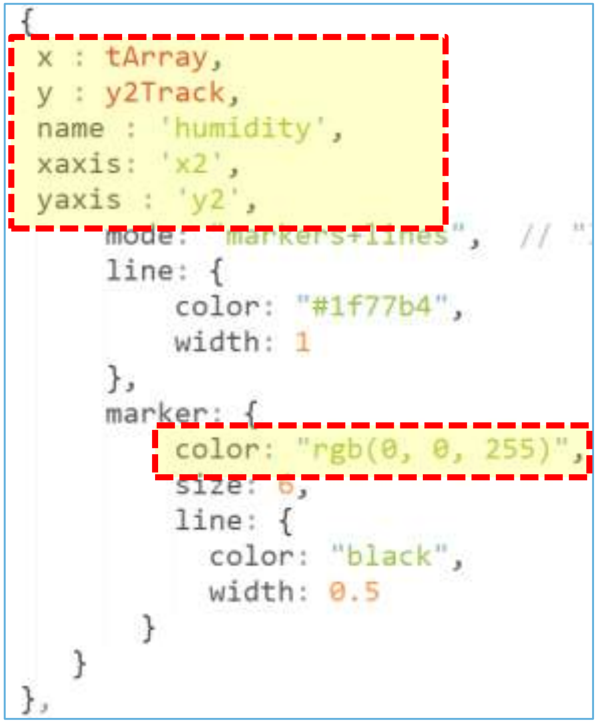
temp, humi, lux

A5.8.7 DHT22 + CdS + Node.js

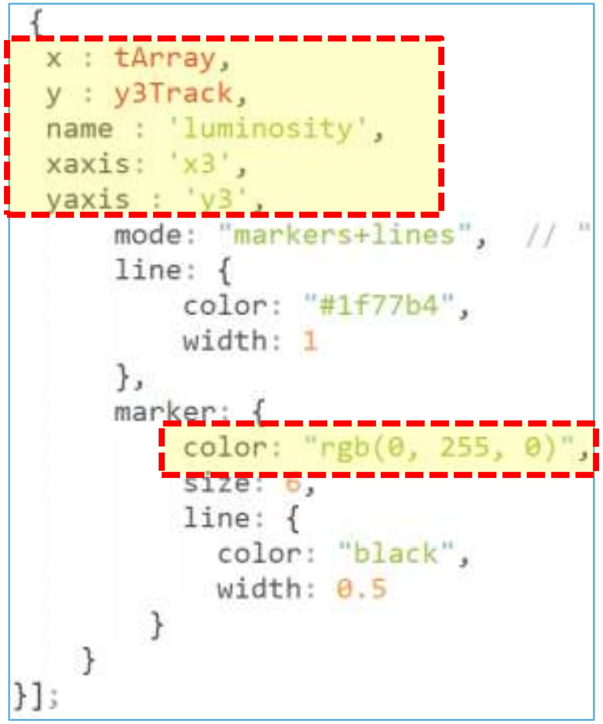
[4.6] WEB client: client_cds_dht22.html - data



```
// data
var data = [{
    x : tArray,
    y : y1Track,
    name : 'temperature',
    mode: "markers+lines",   // "
    line: {
        color: "#1f77b4",
        width: 1
    },
    marker: {
        color: "rgb(255, 0, 0)",
        size: 6,
        line: {
          color: "black",
          width: 0.5
        }
    }
},
```



```
{
    x : tArray,
    y : y2Track,
    name : 'humidity',
    xaxis: 'x2',
    yaxis : 'y2',
    mode: "markers+lines",   // "
    line: {
        color: "#1f77b4",
        width: 1
    },
    marker: {
        color: "rgb(0, 0, 255)",
        size: 6,
        line: {
          color: "black",
          width: 0.5
        }
    }
},
```



```
{
    x : tArray,
    y : y3Track,
    name : 'luminosity',
    xaxis: 'x3',
    yaxis : 'y3',
    mode: "markers+lines",   // "
    line: {
        color: "#1f77b4",
        width: 1
    },
    marker: {
        color: "rgb(0, 255, 0)",
        size: 6,
        line: {
          color: "black",
          width: 0.5
        }
    }
}];
```

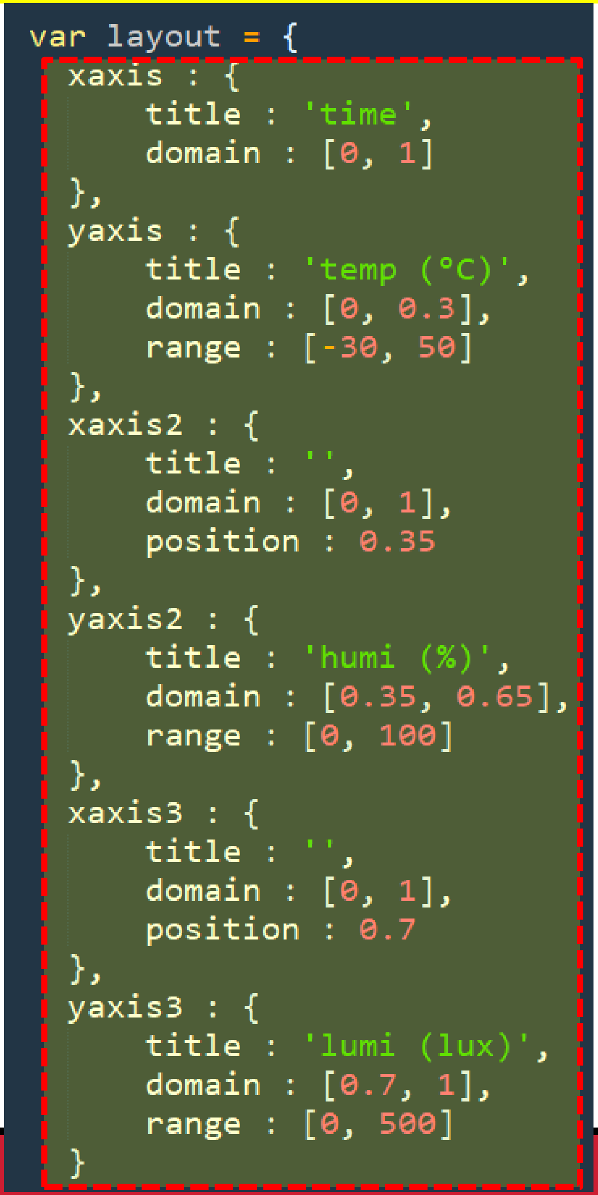
Update data

to include three signals:

temp, humi, lux

A5.8.7 DHT22 + CdS + Node.js

[4.7] WEB client: client_cds_dht22.html - layout



```
var layout = {
    xaxis : {
        title : 'time',
        domain : [0, 1]
    },
    yaxis : {
        title : 'temp (°C)',
        domain : [0, 0.3],
        range : [-30, 50]
    },
    xaxis2 : {
        title : '',
        domain : [0, 1],
        position : 0.35
    },
    yaxis2 : {
        title : 'humi (%)',
        domain : [0.35, 0.65],
        range : [0, 100]
    },
    xaxis3 : {
        title : '',
        domain : [0, 1],
        position : 0.7
    },
    yaxis3 : {
        title : 'lumi (lux)',
        domain : [0.7, 1],
        range : [0, 500]
    }
```

1. Update layout

   to include three signals:

   temp, humi, lux.

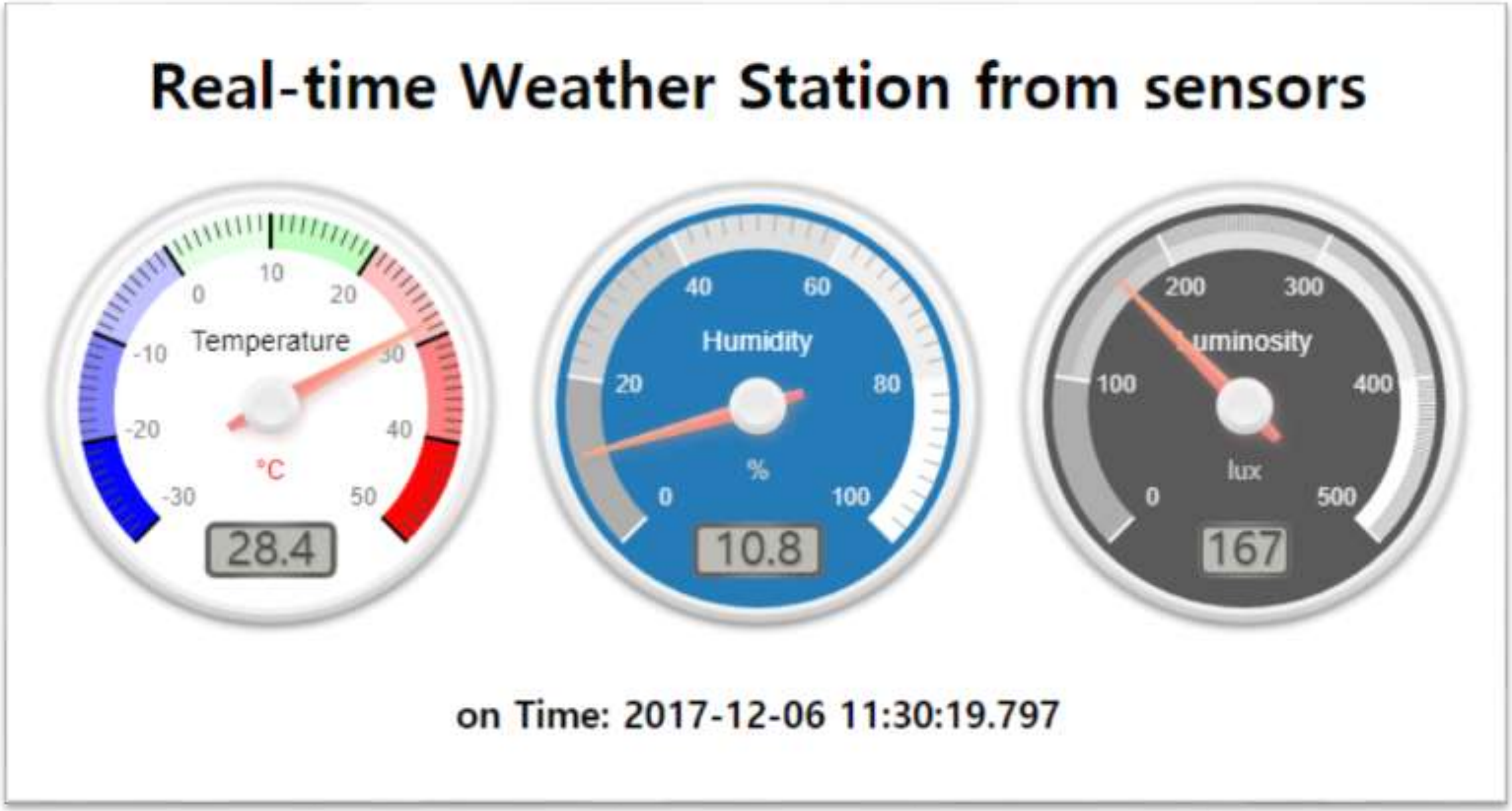
2. Check the domain &

   position.

Save the complete code as AAnn_cds_dht22.html

A5.8.8 DHT22 + CdS + Node.js

[4.8] WEB client: client_dht22_ldr.html – Design your gauges



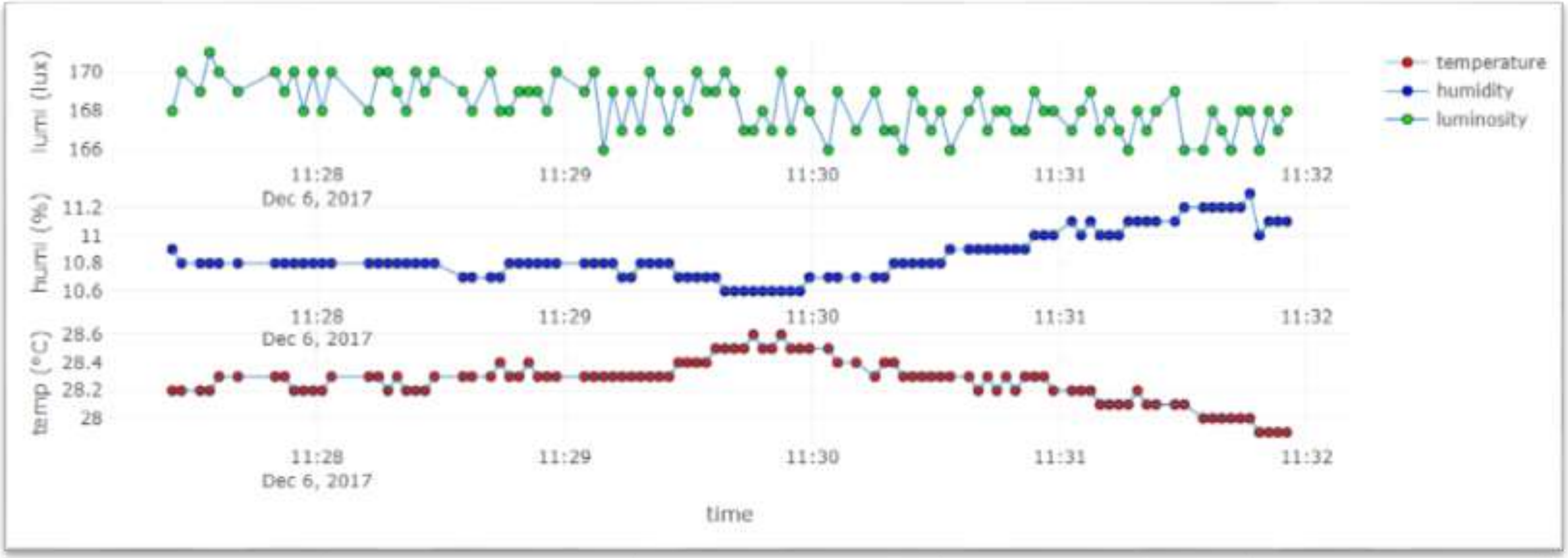
Real-time Weather Station from sensors

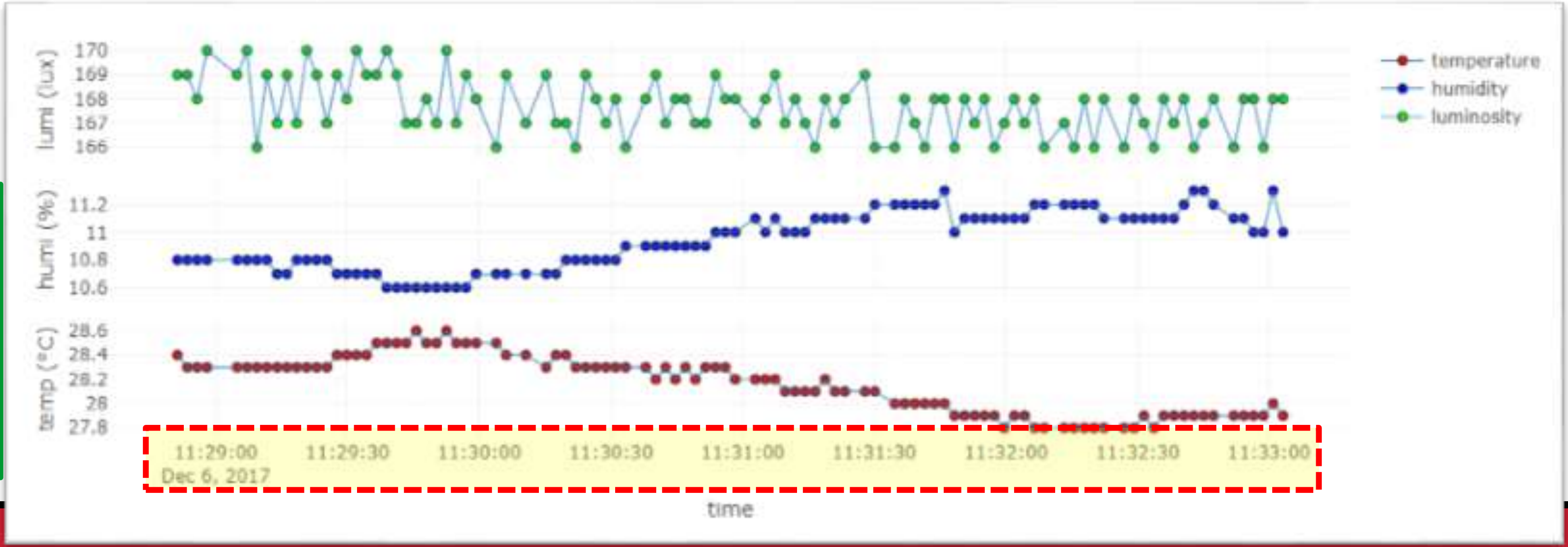
on Time: 2017-12-06 11:30:19.797

Save the complete code as AAnn_cds_dht22.html

A5.8.9 DHT22 + CdS + Node.js

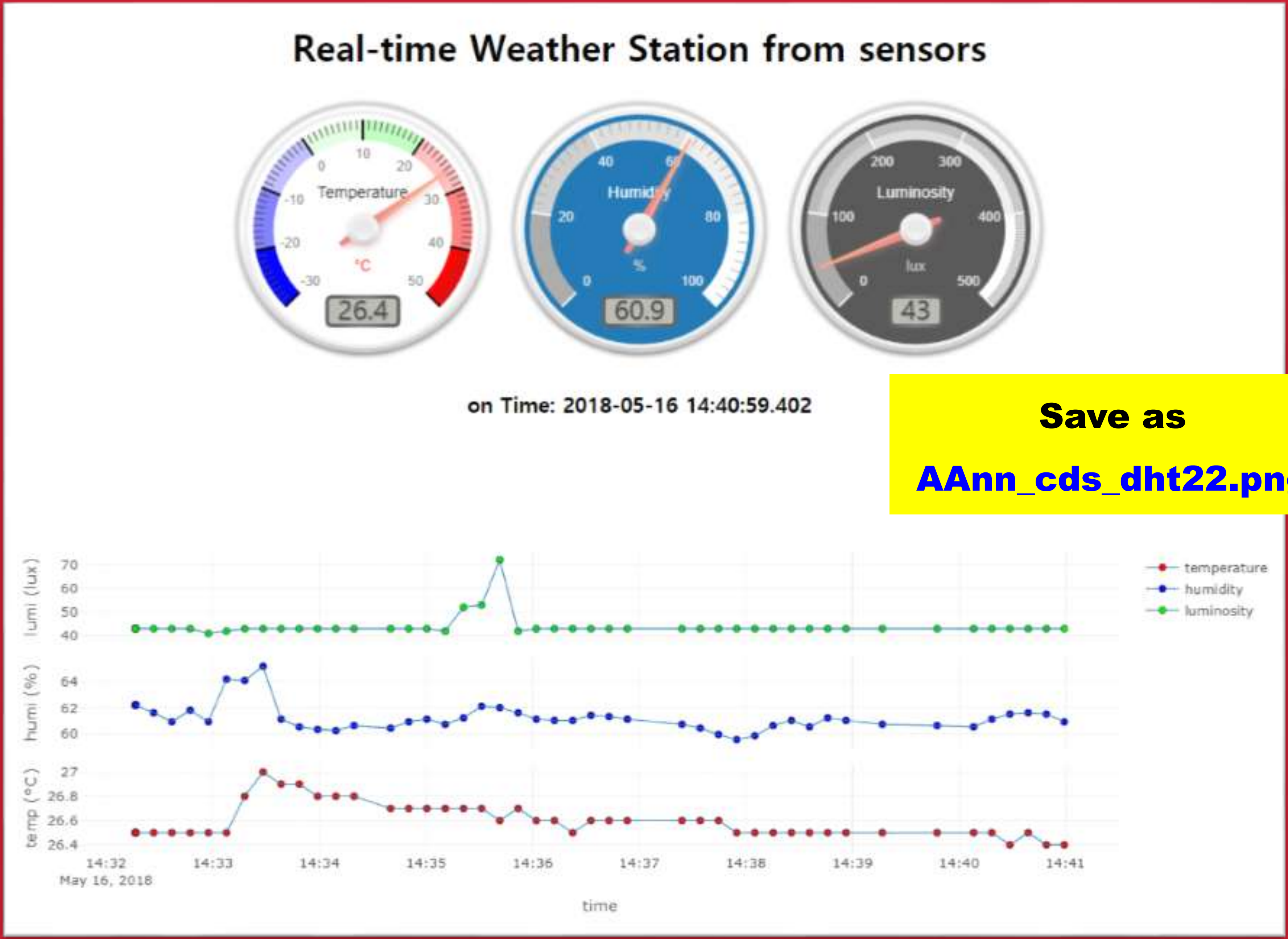
[4.9] WEB client: Design layout (show date at lower axis)





[Hint]

Plot.ly

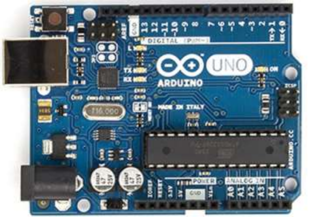


Real-time Weather Station from sensors

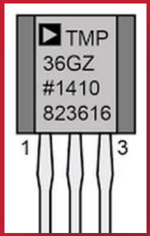
on Time: 2018-05-16 14:40:59.402

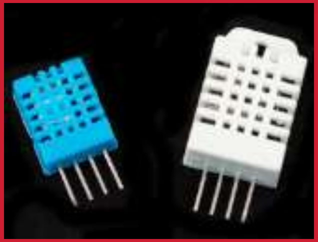
Save as

AAnn_cds_dht22.png



[Practice]





◆ [wk12]

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

[wk12] Practice-09 AAnn_Rpt09.zip

◆ [Target of this week]

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

제출파일명 : **AAnn_Rpt09.zip**

- 압축할 파일들

① **AAnn_DS_cds_tmp36.png**

② **AAnn_cds_dht22_data.png**

③ **AAnn_cds_dht22.html**

④ **AAnn_cds_dht22.png**

Email : chaos21c@gmail.com

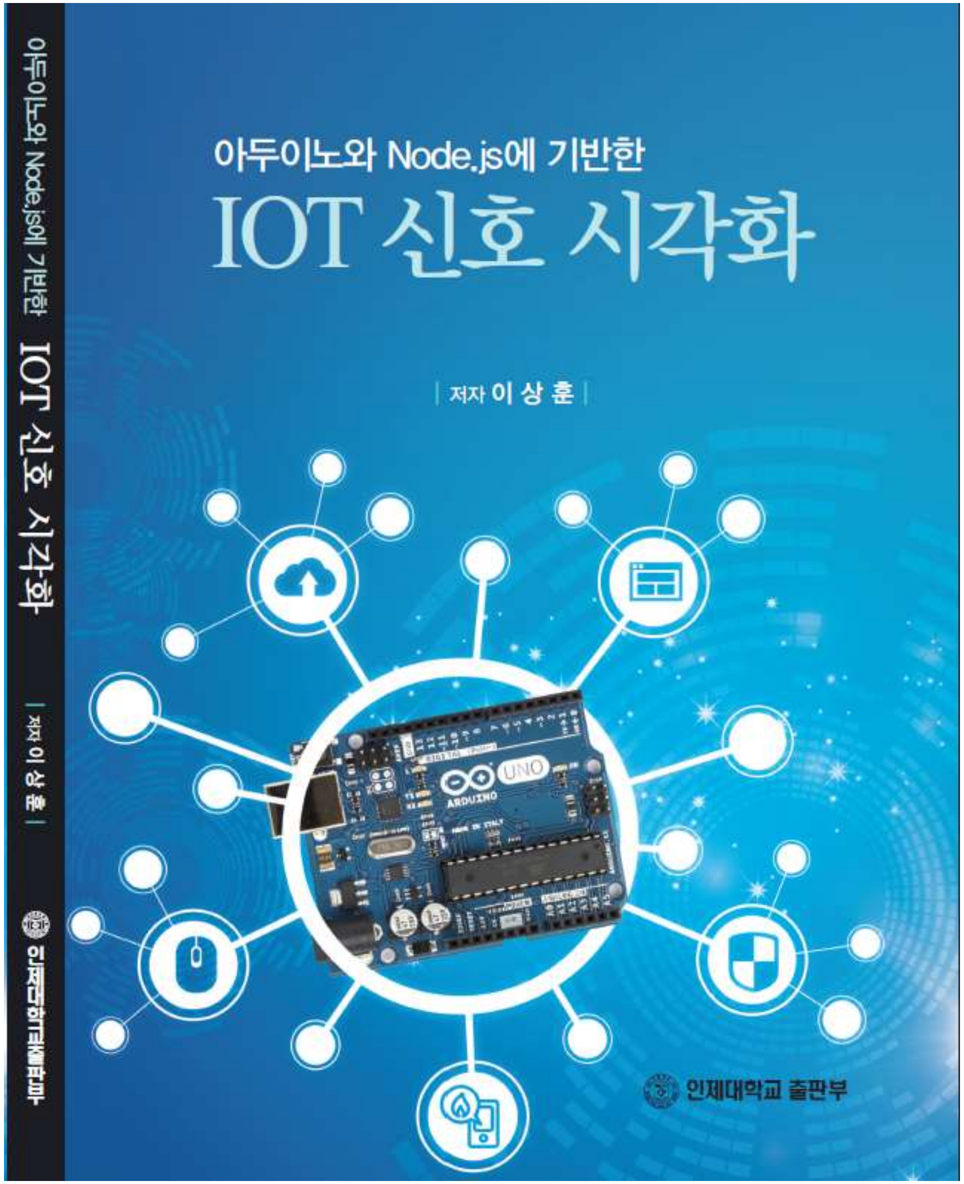
[제목 **: id,** 이름 **(**수정**)]**

Lecture materials

References & good sites

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

주교재 및 참고도서

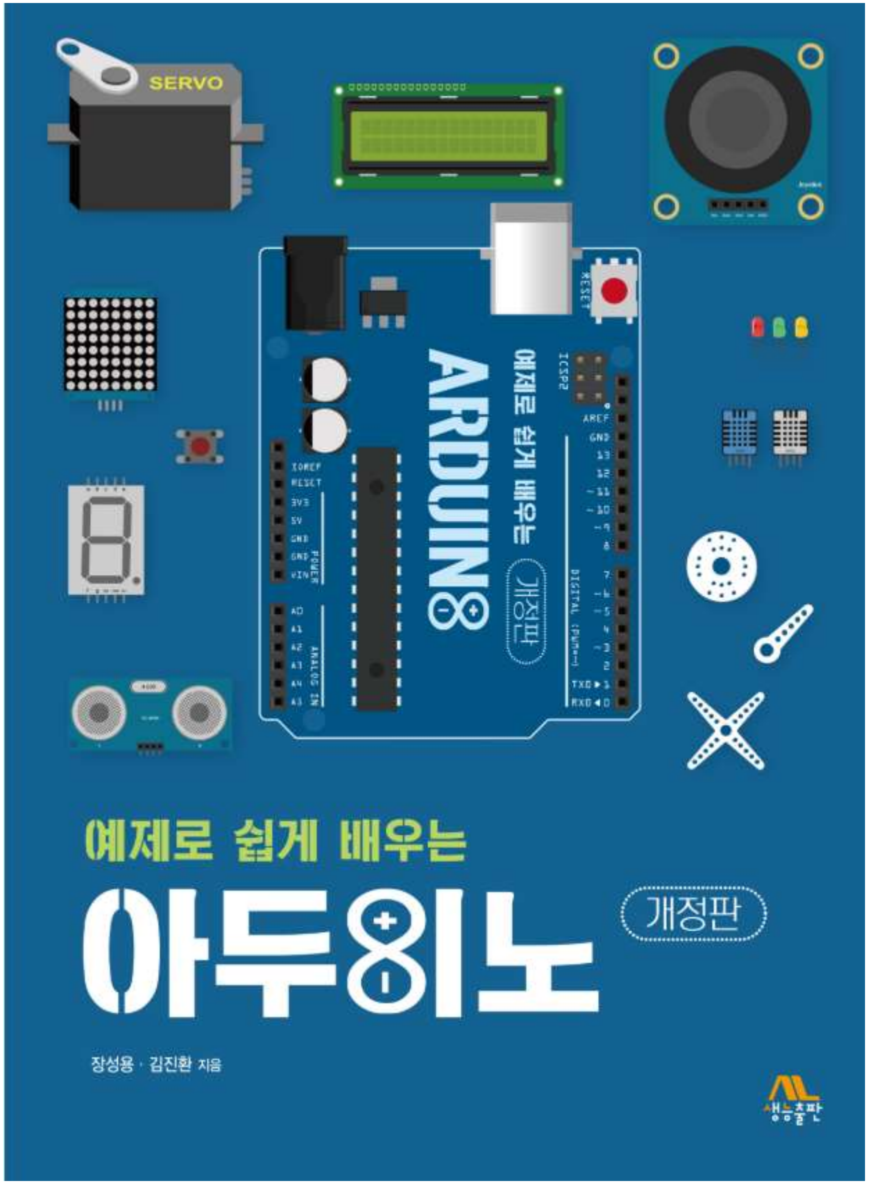


아두이노와 Node.js에 기반한

IOT 신호 시각화

| 저자 이 상 훈 |

인제대학교 출판부



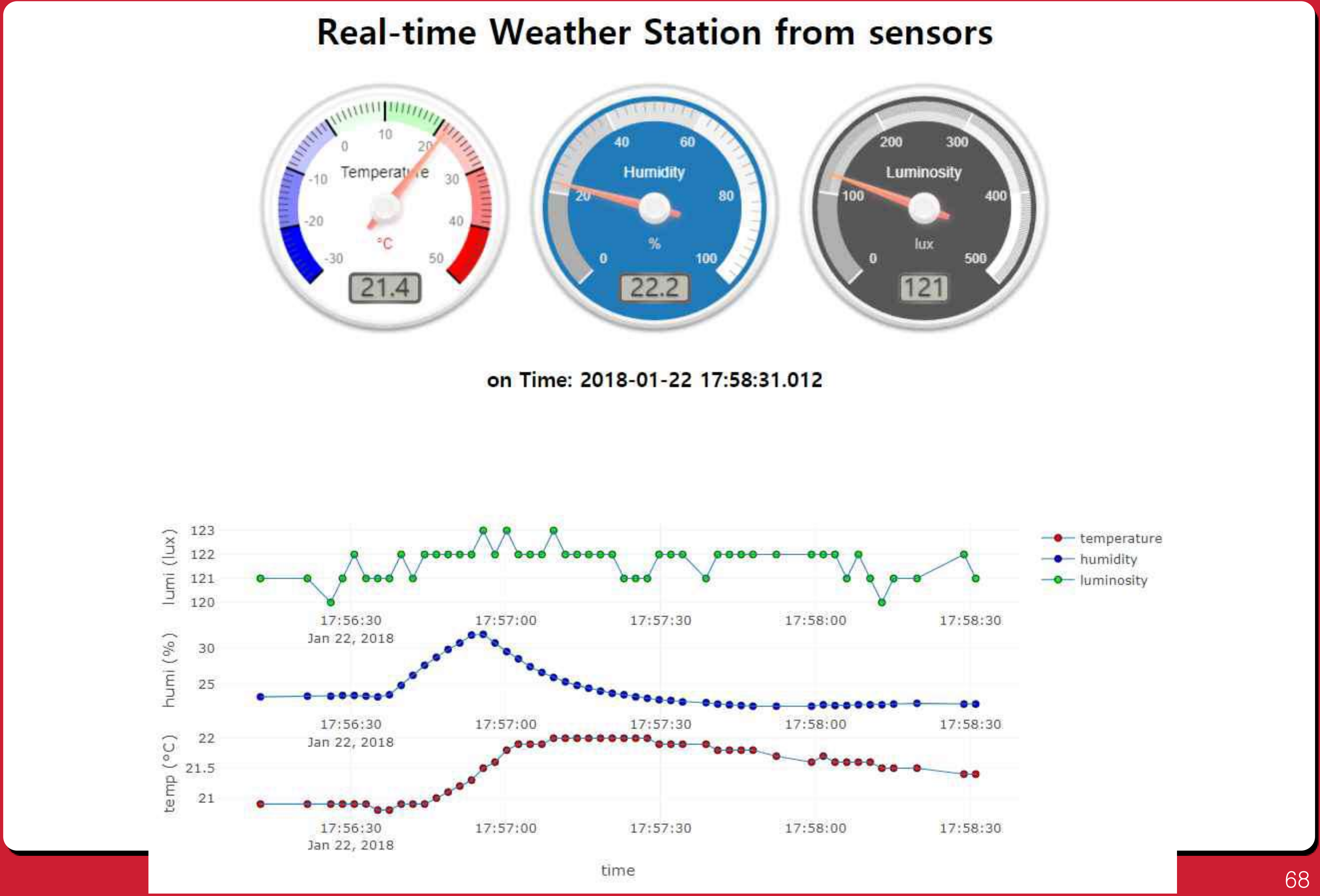
예제로 쉽게 배우는

아두이노 (개정판)

장성용 · 김진환 지음

생능출판

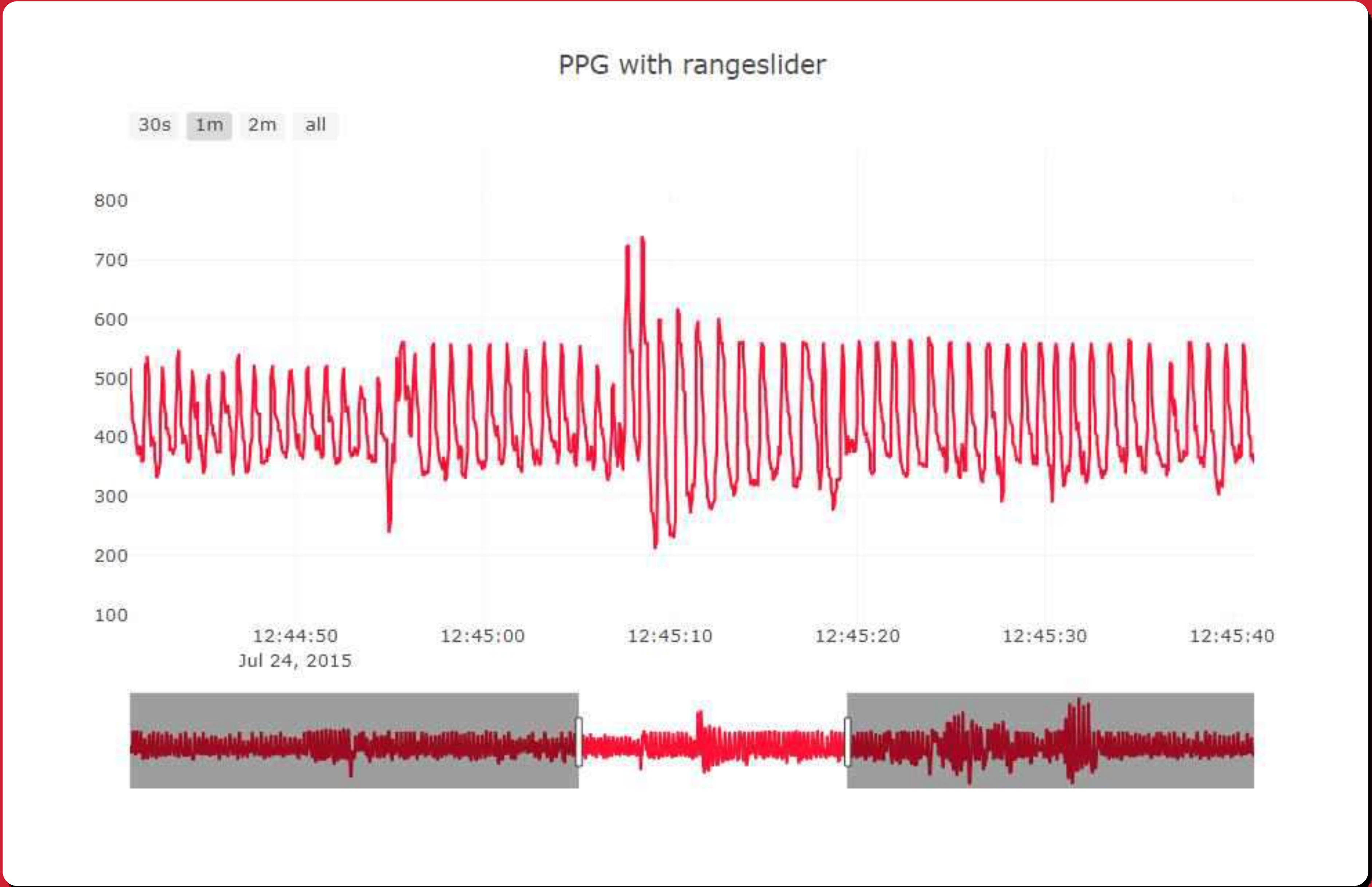
Target of this class



Real-time Weather Station from sensors

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

Another target of this class


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