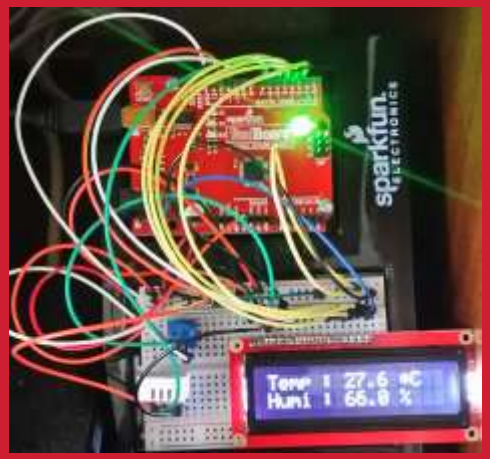
Arduino-IOT

[wk10]



Arduino + nods.js Data visualization II

Visualization of Signals using Arduino, Node.js & Storing Signals in MongoDB & Mining Data using Python

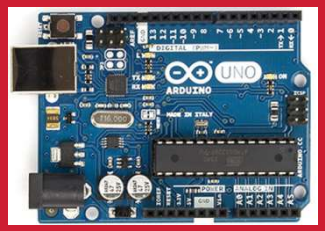
Comsi, INJE University

2nd semester, 2019

Email : chaos21c@gmail.com

My ID

ID	성명
AA01	김관용
AA02	백동진
AA03	김도훈
AA04	김희찬
AA05	류재현
AA06	문민규
AA07	박진석
AA08	이승협
AA09	표혜성
AA10	김다영
AA11	성소진
AA12	김해인
AA13	신송주
AA14	윤지훈



[Review]

- [wk09]
  - Charts by plotly
  - Complete your plotly chart project
  - Upload folder: AAnn_Rpt07

wk09 : Practice : AAnn_Rpt07

- [Target of this week]
  - Complete your works
  - Save your outcomes and upload outputs in github

제출폴더명 : **AAnn_Rpt07**

- 압축할 파일들

① AAnn_Chart_Layout.png

② AAnn_Axis_Title.png

③ AAnn_Line_Dash_Dot.png

④ AAnn_lux_Time_Series.png

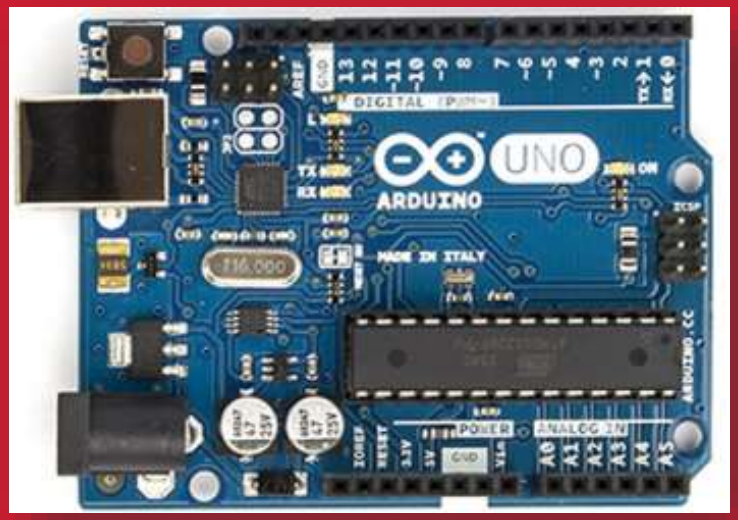
⑤ AAnn_lux_Rangeslider.png

⑥ All *.ino

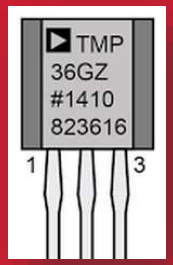
⑦ All *.js

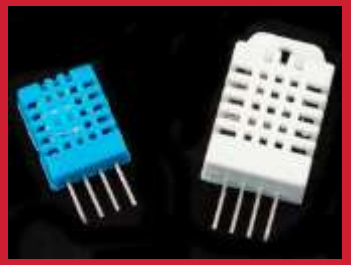
⑧ All *.html

Email : chaos21c@gmail.com

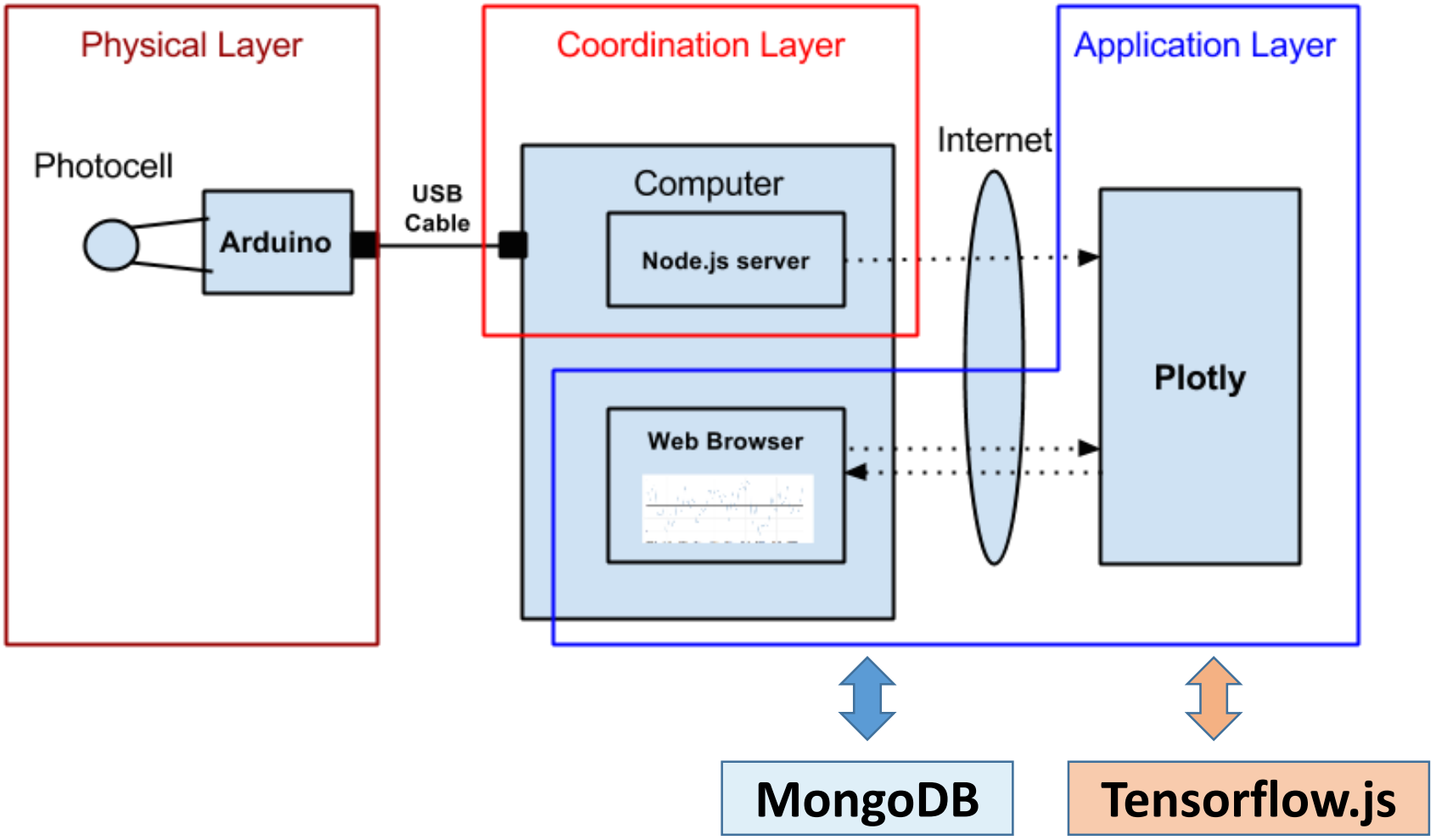


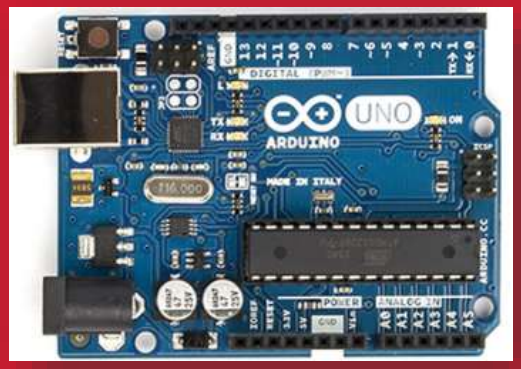
Arduino & Node.js



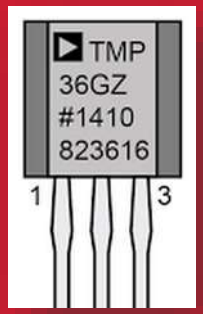


Layout [H S C]


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

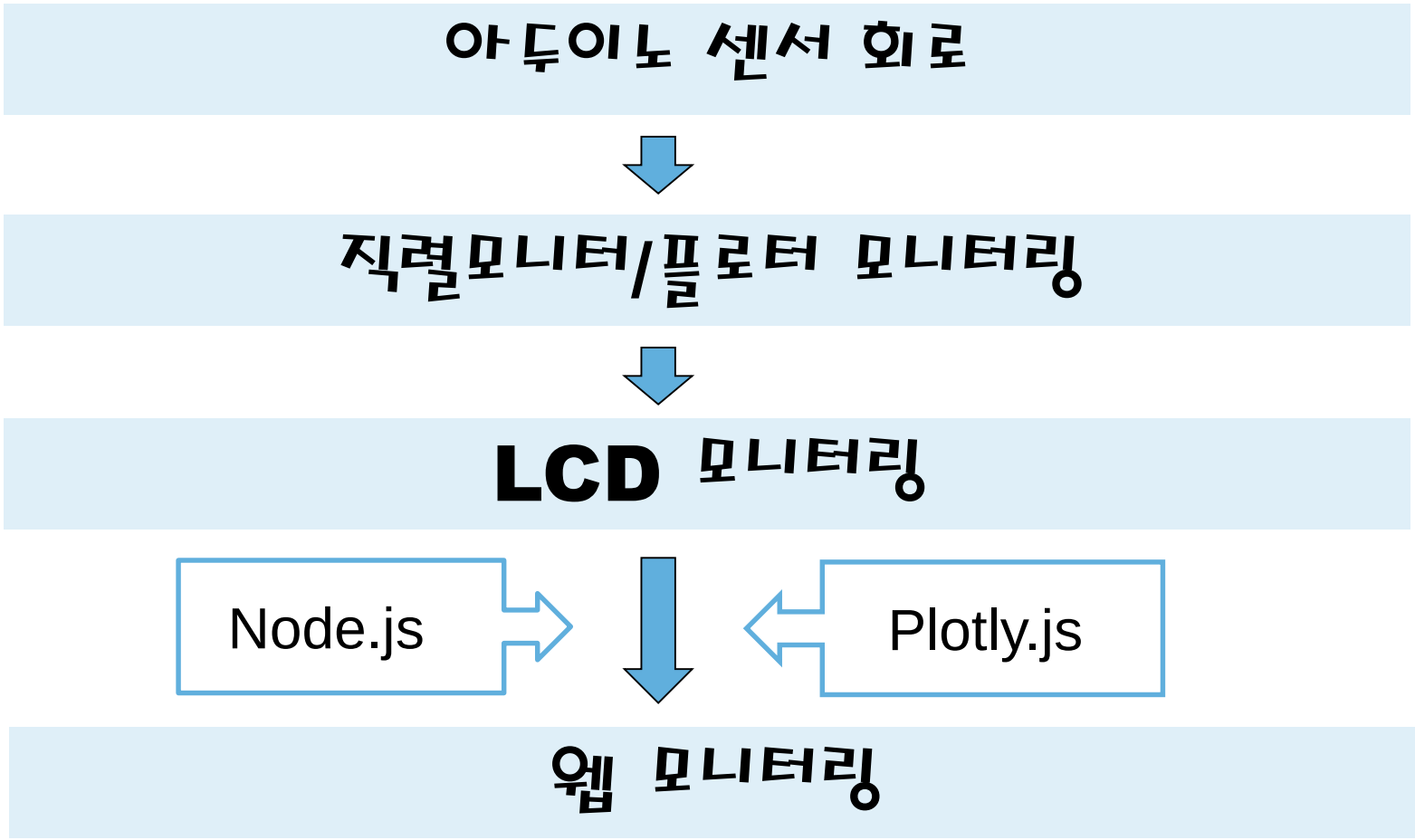


Data visualization using ploy.ly

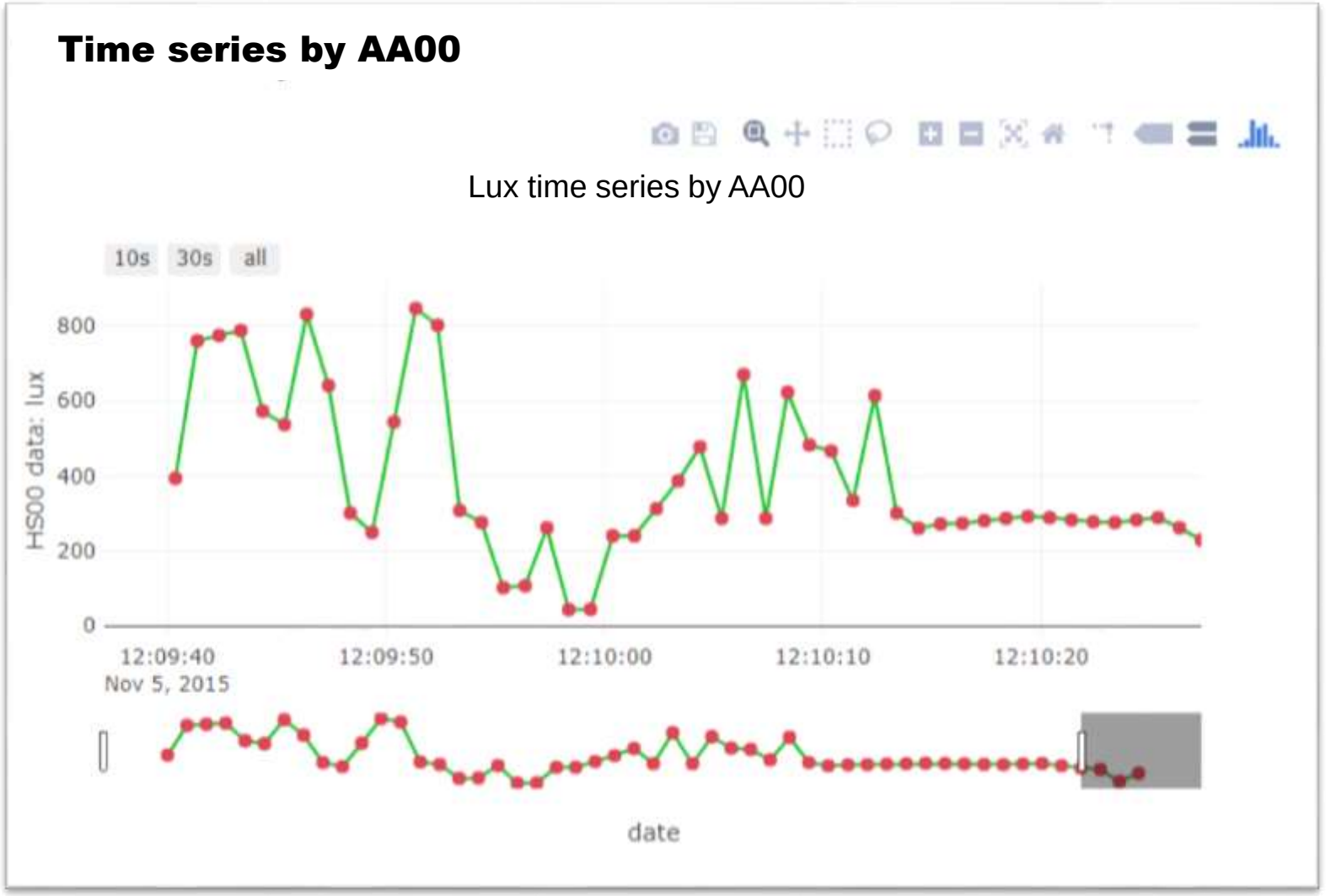


Arduino data on network socket

A5.1 Introduction to data visualization

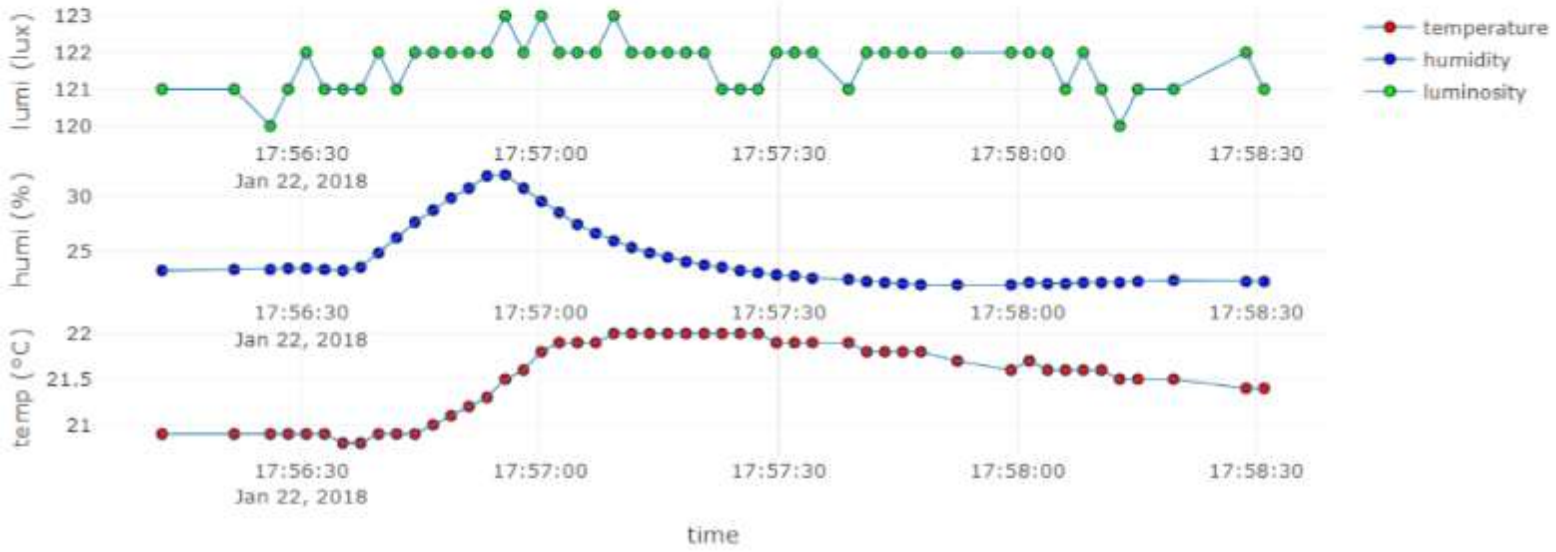


Arduino data + plotly



Real-time Weather Station from sensors

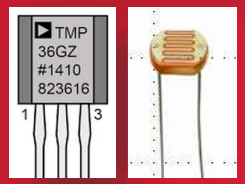
on Time: 2018-01-22 17:58:31.012

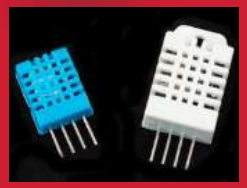


A5.1.3 plotly.js

plotly.js is Plotly's client-side, interactive JavaScript graphing library, built on top of D3.js, stack.gl.

https://plot.ly/javascript/





Data visualization using plotly.js

A5.3.4.1 plotly.js: Sensor time series

[3] Time series : my lux data

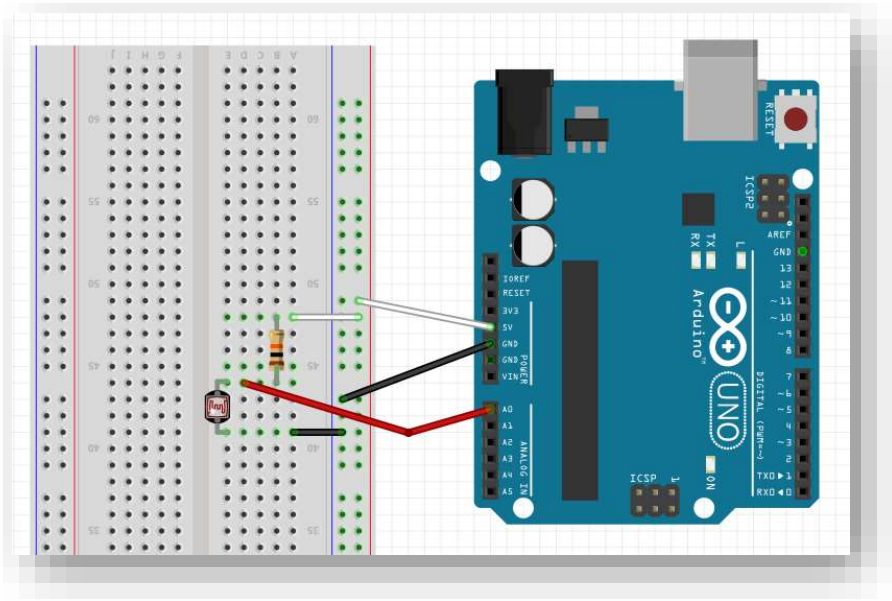
```
'2015-11-05 12:09:41.382',
'2015-11-05 12:09:42.380',
'2015-11-05 12:09:43.378',
'2015-11-05 12:09:44.377',
'2015-11-05 12:09:45.375',
'2015-11-05 12:09:46.389',
'2015-11-05 12:09:47.388',
'2015-11-05 12:09:48.386',
'2015-11-05 12:09:49.384',
'2015-11-05 12:09:50.383',
'2015-11-05 12:09:51.381',
'2015-11-05 12:09:52.380',
'2015-11-05 12:09:53.394',
'2015-11-05 12:09:54.392',
'2015-11-05 12:09:55.391',
'2015-11-05 12:09:56.389',
'2015-11-05 12:09:57.387',
'2015-11-05 12:09:58.386',
```

```
394,761,775,788,573,537,831,641,300,249,544,847,802,307,
```

```
'2015-11-05 12:10:01.397',
```

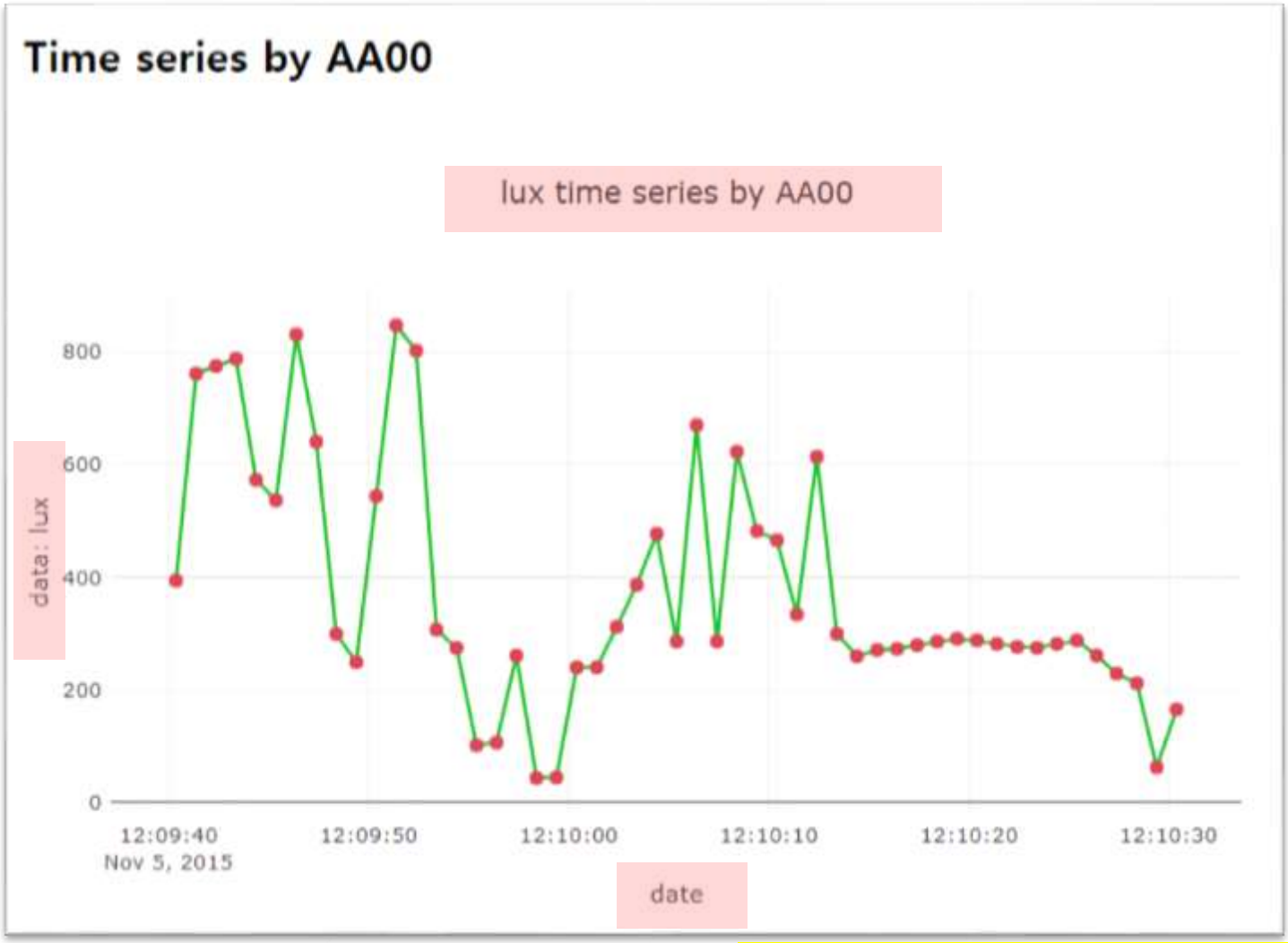
Data :

date,value



A5.3.4.3 plotly.js: Time series

[3] Time series : my lux data – [DIY] → Set title and axis title



AAnn_lux_Time_Series.png

Project: Time series with Rangeslider

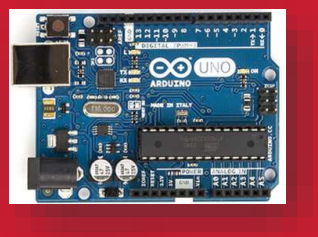
[Project-DIY] AAnn_lux_Rangelslider.html

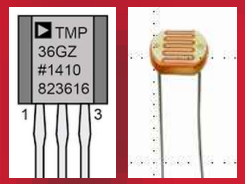
AAnn_lux_Rangelslider.png

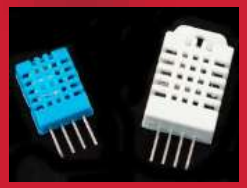
Time series with Rangeslider

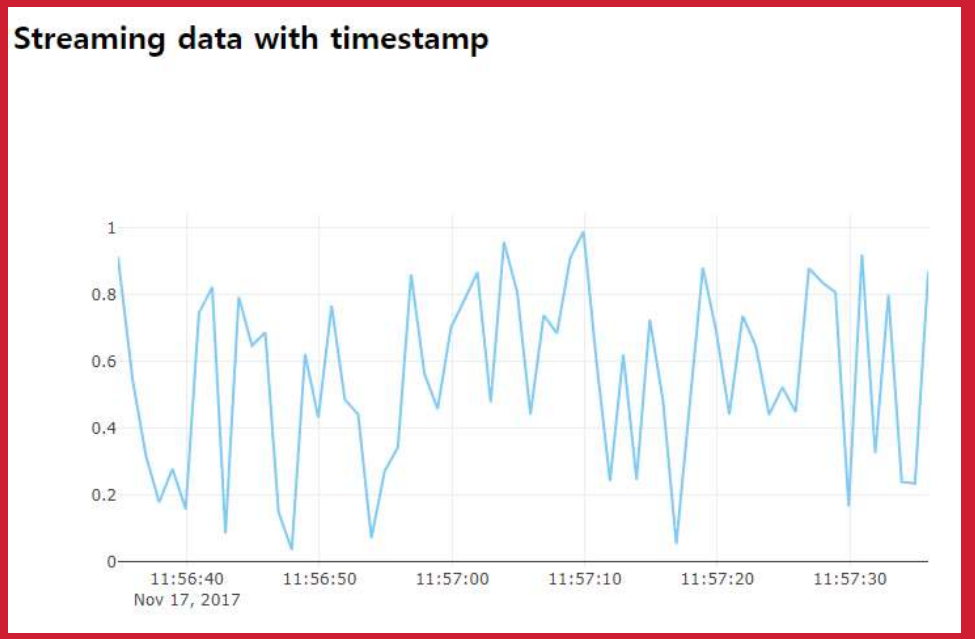
```
var layout = {
  title:'lux time series by AA00',
  width: 750, height: 500,
  margin: {
    l: 50,
    r: 50,
    b: 100,
    t: 100,
    pad: 4
  },
  xaxis: {
    title: 'date',
    autorange: true,
  range: ['2015-11-05 12:09:40.383', '2015-11-05 12:10:30.413'],
  rangeselector: {buttons: [
      {
        count: 10,
        label: '10s',
        step: 'second',
        stepmode: 'backward'
      },
      {
        count: 30,
        label: '30s',
        step: 'second',
        stepmode: 'backward'
      },
      {step: 'all'}
    ]},
  rangeslider: {range: ['2015-11-05 12:09:40.383', '2015-11-05 12:10:30.413']},
  type: 'date'
  },
  yaxis: {
    title: 'data: lux'
  }
};
```

Data Streaming using plotly.js



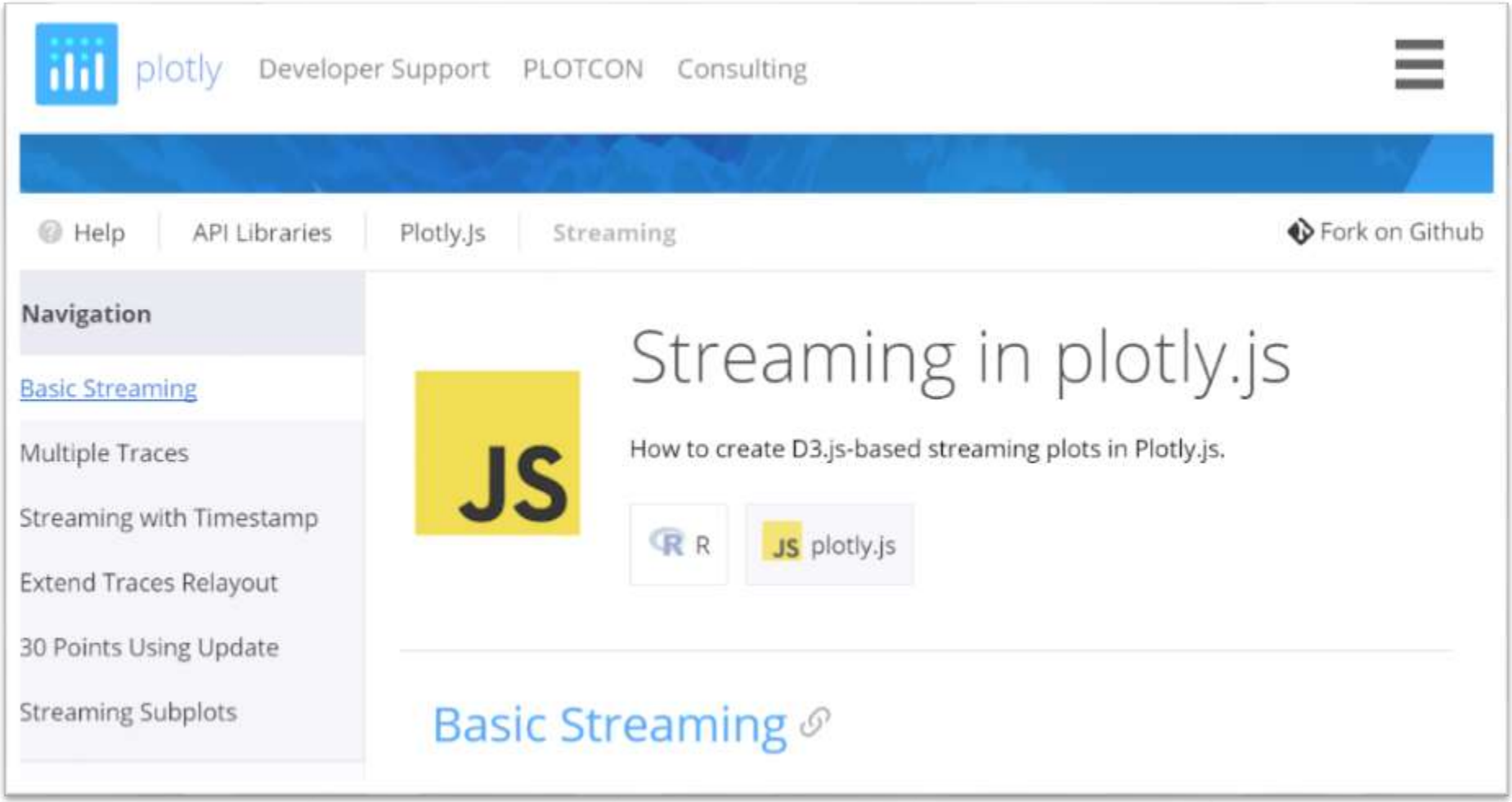






A5.4 plotly.js: Streaming data

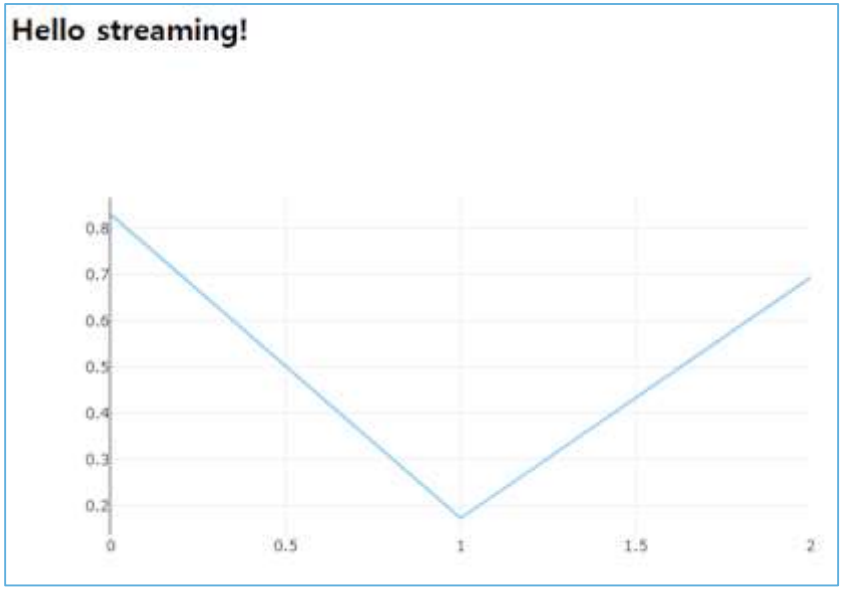
Plot.ly > Streaming



A5.4.1 plotly.js: Streaming data

[1.0] Starting chart

```
<h2>Streaming data</h2>
<div id="graph"></div>

<script>
    function rand() {
        return Math.random(); // 0.0 ~ 1.0
    }

    trace = {
        y: [1,2,3].map(rand),
        mode: 'lines',
        line: {color: '#80CAF6'}
    };

    data = [trace];

    Plotly.plot('graph', data);

    // var cnt = 0;

    // var interval = setInterval(function() {

    //   cnt++;
    //   Plotly.extendTraces('graph', {
    //       y: [[rand()]]
    //   }, [0]);

    //   if(cnt == 50) clearInterval(interval);
    // }, 1000);
</script>
```

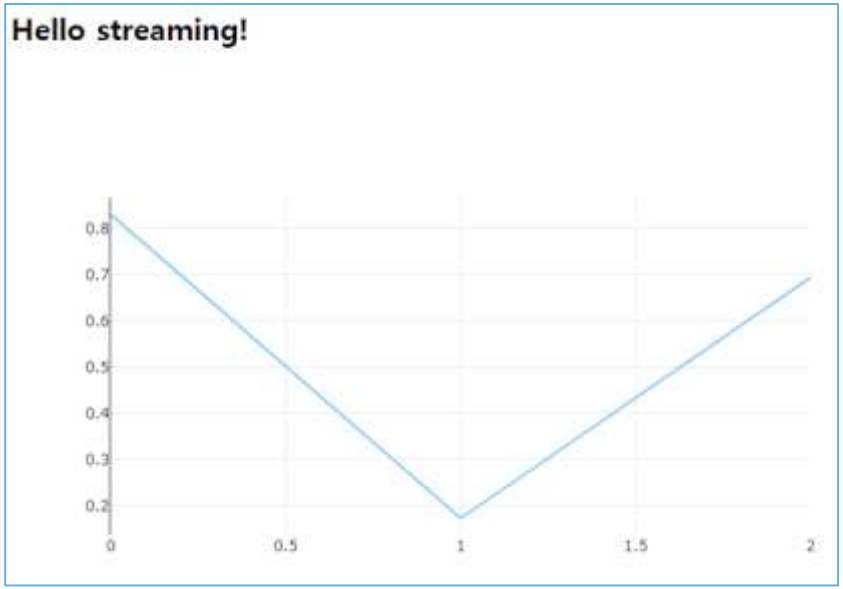


https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/map

A5.4.2.1 plotly.js: Streaming data

[1.1] Starting chart (new)

```
<h2>Hello streaming!</h2>

<div id="graph"></div>

<script>
    function rand() {
        return Math.random();
    }

    Plotly.plot('graph', [{
        y: [1,2,3].map(rand),
        mode: 'lines',
        line: {color: '#80CAF6'}
    }]);


    /*var cnt = 0;
    var interval = setInterval(function() {
        cnt++;
        Plotly.extendTraces('graph', {
            y: [[rand()]]
        }, [0]);
        if(cnt == 30) clearInterval(interval);
    }, 2000);*/
</script>
```



A5.4.2.2 plotly.js: Streaming data

[1.2] Basic streaming

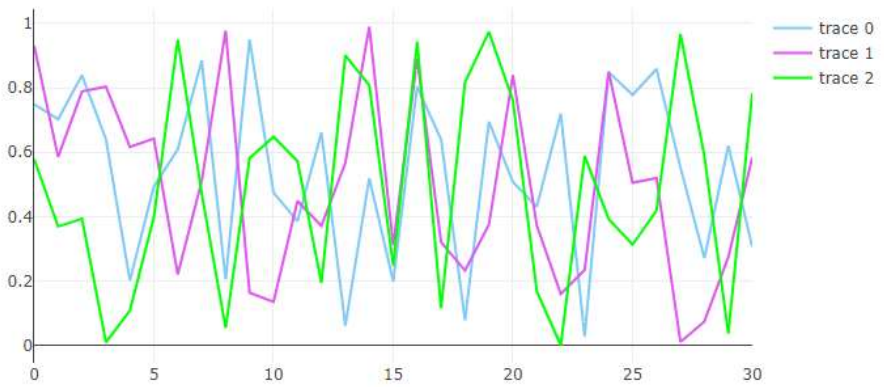
```
<h2>Streaming data!</h2>

<div id="graph"></div>

<script>
    function rand() {
        return Math.random();
    }

    Plotly.plot('graph', [{
        y: [1,2,3].map(rand),
        mode: 'lines',
        line: {color: '#80CAF6'}
    }]);

    var cnt = 0;
    var interval = setInterval(function() {
        cnt++;
        Plotly.extendTraces('graph', {
            y: [[rand()]]
        }, [0]);
        if(cnt == 30) clearInterval(interval);
    }, 2000);
</script>
```

A5.4.3.1 plotly.js: Streaming data

[2.1] Streaming multiple traces

```
function rand() {
    return Math.random();
}

// initial plot
trace1 = {
    y: [1,2,3].map(rand),
    mode: 'lines',
    line: {color: '#80CAF6'}
};
trace2 = {
    y: [1,2,3].map(rand),
    mode: 'lines',
    line: {color: '#DF56F1'}
};
trace3 = {
    y: [1,2,3].map(rand),
    mode: 'lines',
    line: {color: '#00FF00'}
};

data = [trace1, trace2, trace3];

Plotly.plot('graph', data);
```

```
// continous plot
var cnt = 0;
var interval = setInterval(function() {

    Plotly.extendTraces('graph', {
        y: [[rand()], [rand()], [rand()]]
    }, [0, 1, 2])

    cnt++;
    if(cnt === 100) clearInterval(interval);
}, 300);
```

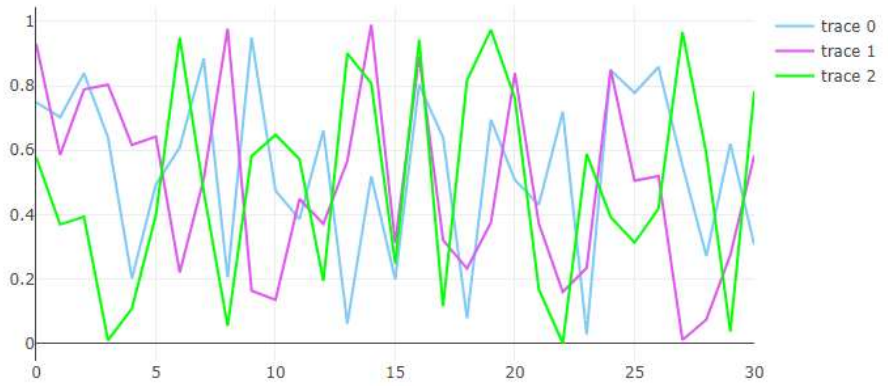


A5.4.3.2 plotly.js: Streaming data

[2.2] Streaming multiple traces (new code)

```
function rand() {
    return Math.random();
}

// initial plot
Plotly.plot('graph', [{
    y: [1,2,3].map(rand),
    mode: 'lines',
    line: {color: '#80CAF6'}
}, {
    y: [1,2,3].map(rand),
    mode: 'lines',
    line: {color: '#DF56F1'}
}, {
    y: [1,2,3].map(rand),
    mode: 'lines',
    line: {color: '#00FF00'}
}]);
```

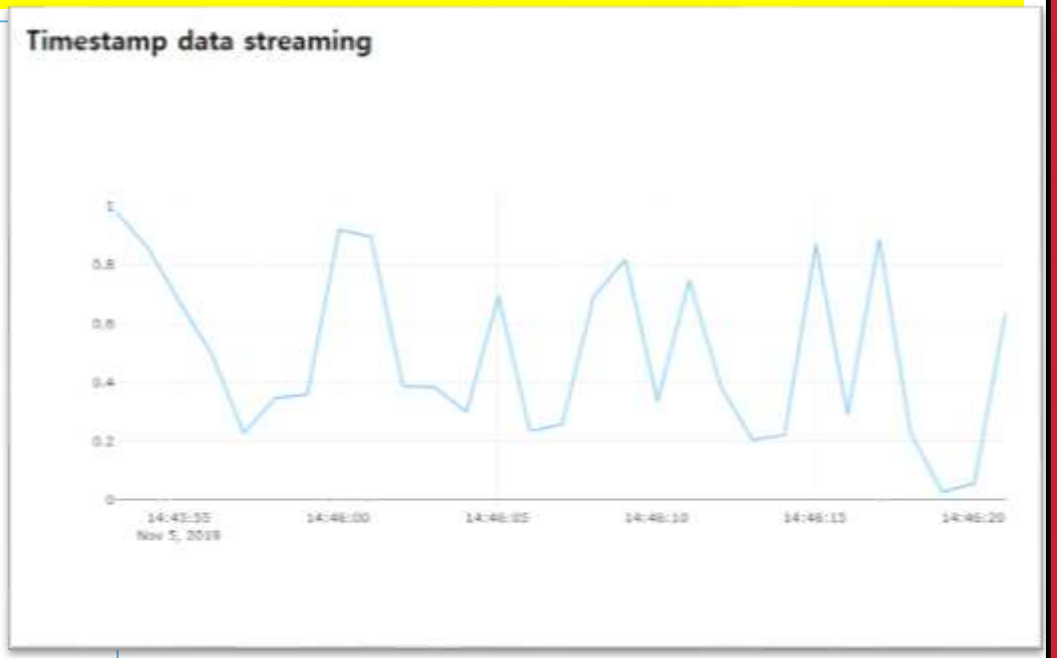
```
// continous plot
var cnt = 0;
var interval = setInterval(function() {

    Plotly.extendTraces('graph', {
        y: [[rand()], [rand()], [rand()]]
    }, [0, 1, 2])

    cnt++;

    if(cnt === 100) clearInterval(interval);
}, 300);
```



A5.4.4 plotly.js: Streaming data

[3] Streaming data with timestamp

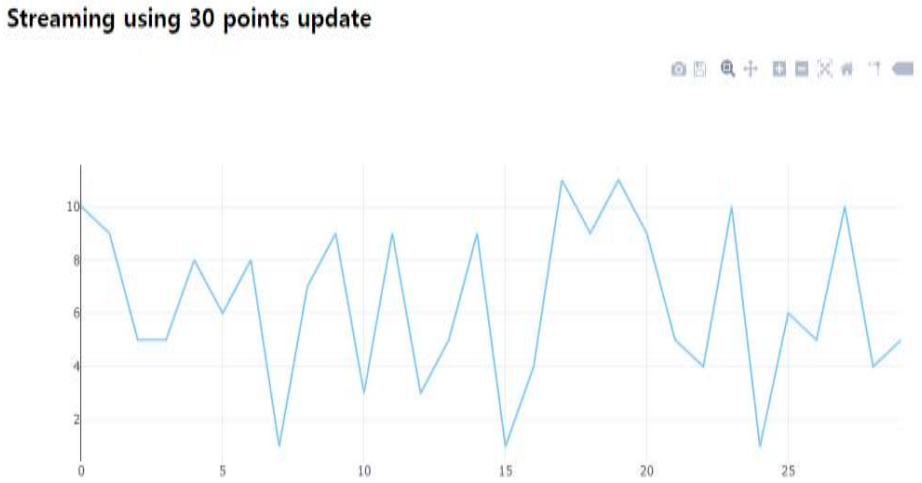
```
function rand() {
    return Math.random();
}

var time = new Date();
var data = [{
    x: [time],
    y: [rand()],
    mode: 'lines',
    line: {color: '#80CAF6'}
}]

Plotly.plot('graph', data);

var cnt = 0;

var interval = setInterval(function() {

    var time = new Date();

    var update = {
        x:  [[time]],
        y: [[rand()]]
    }

    Plotly.extendTraces('graph', update, [0])

    if(cnt === 100) clearInterval(interval);
}, 1000);
```



A5.4.5 plotly.js: Streaming data

[4] Streaming data using 30 points update

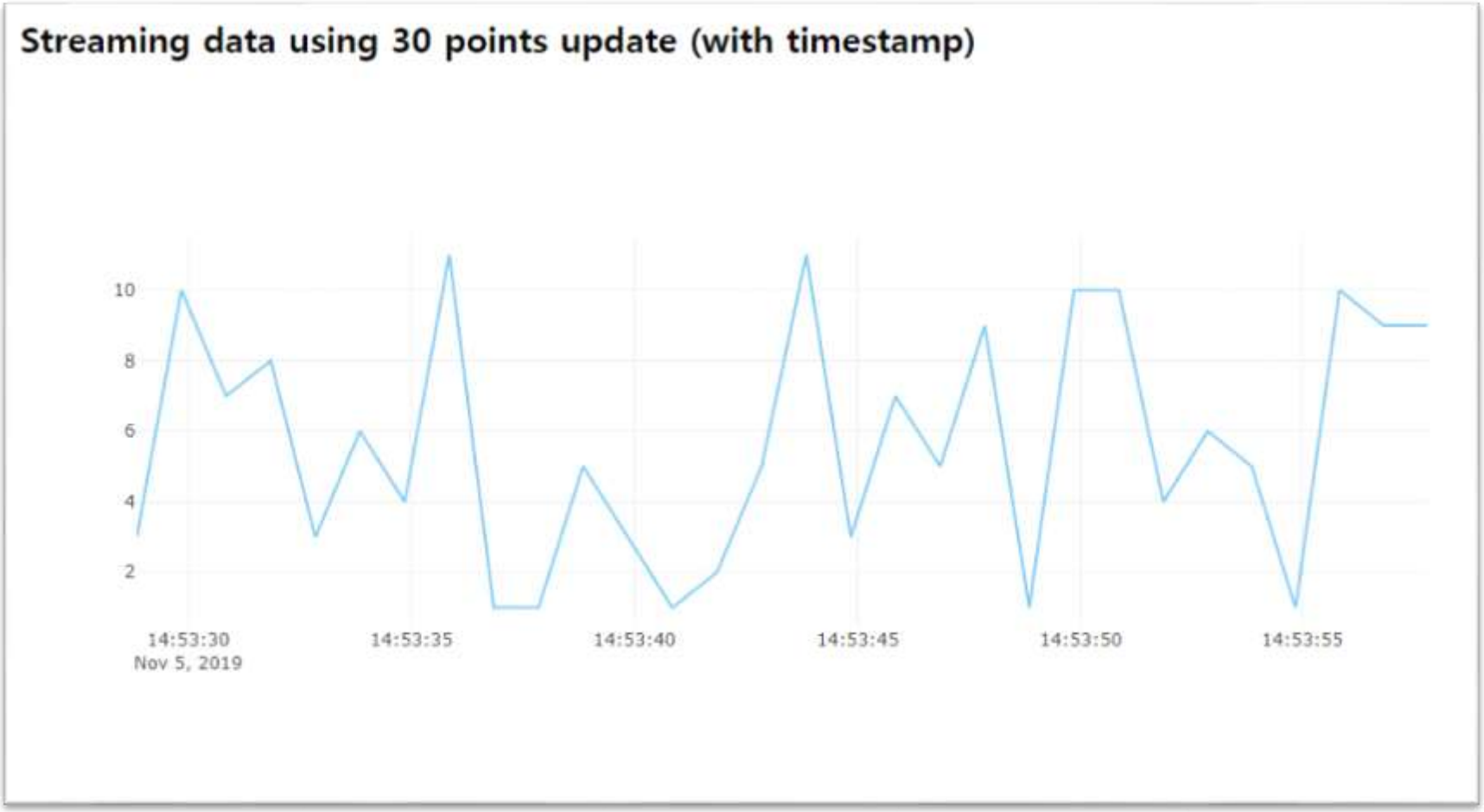
```
var arrayLength = 30
var newArray = []

// initial 30 data
for(var i = 0; i < arrayLength; i++) {
    var y = Math.round(Math.random()*10) + 1
    newArray[i] = y
}

var data = [{
    y: newArray,
    mode: 'lines',
    line: {color: '#80CAF6'}
}]

Plotly.plot('graph', data);
```




```
var cnt = 0;
var interval = setInterval(function() {

    var y = Math.round(Math.random()*10) + 1
    newArray = newArray.concat(y)
    newArray.splice(0, 1)//remove the oldest data

    var update = {
        y:  [newArray]
    }

    Plotly.update('graph', update)
    //cnt++;
    if(cnt === 50) clearInterval(interval);
}, 1000);
```

A5.4.5.1 plotly.js: Streaming data

[4.1] Streaming data using 30 points update (with timestamp)



A5.4.5.2 plotly.js: Streaming data

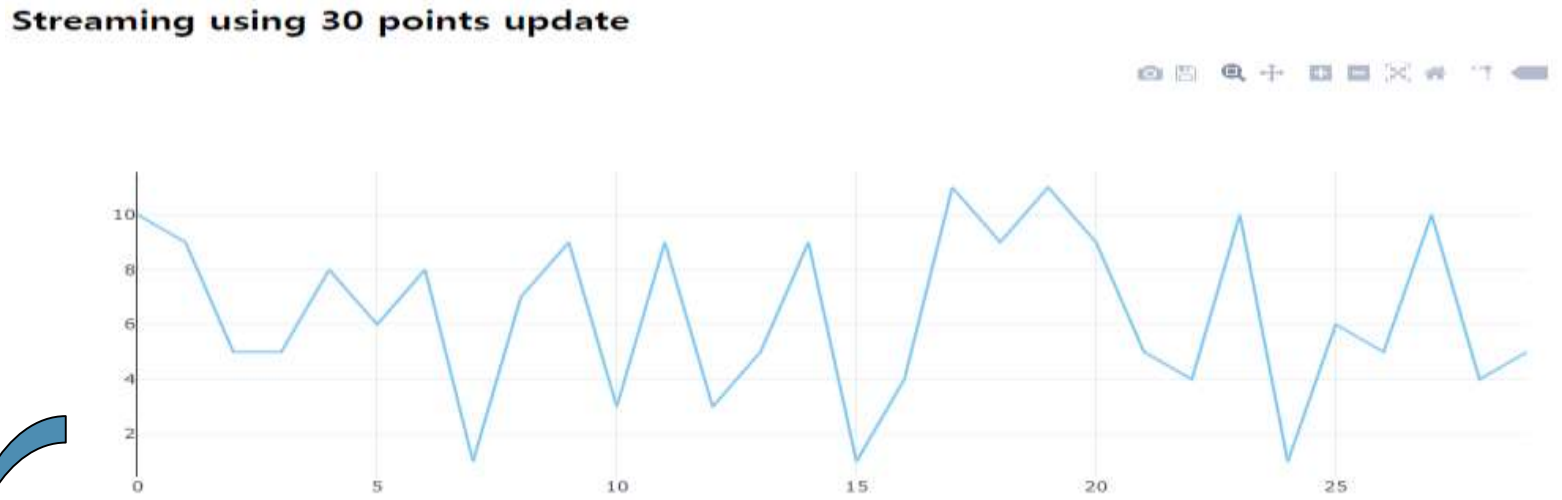
[4.2] Streaming data using 30 points update (with timestamp)

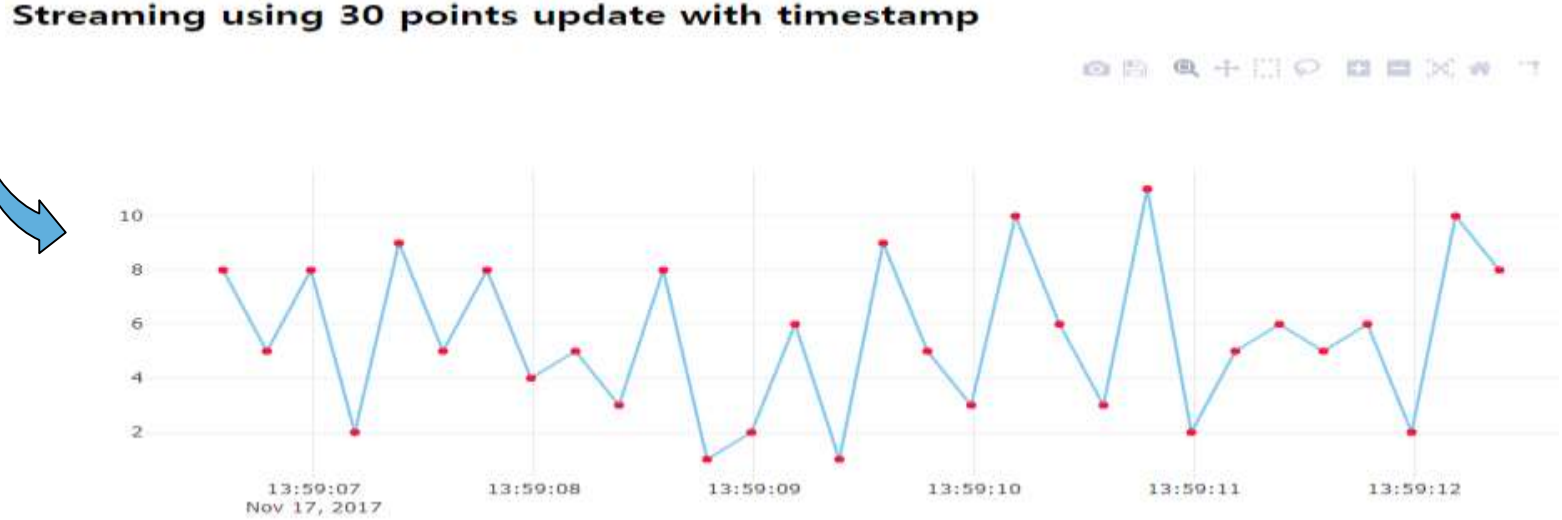
```
<h2>Streaming using 30 points update</h2>

<div id="graph"></div>

<script>
    var arrayLength = 30
    var newArray = []
    var timeArray = []
    // initial 30 data
    for(var i = 0; i < arrayLength; i++) {
        var y = Math.round(Math.random()*10) + 1
        var time = new Date();
        newArray[i] = y
        timeArray[i] = time
    }

    var data = [{
        x: timeArray,
        y: newArray,
        mode: 'lines',
        line: {color: '#80CAF6'}
    }]

    Plotly.plot('graph', data);
```

```
var cnt = 0;

var interval = setInterval(function() {

    var y = Math.round(Math.random()*10) + 1
    var time = new Date();

    timeArray = timeArray.concat(time)
    timeArray.splice(0, 1)//remove the oldest data
    newArray = newArray.concat(y)
    newArray.splice(0, 1)//remove the oldest data

    var update = {
        x:  [timeArray],
        y:  [newArray]
    }

    Plotly.update('graph', update)

    if(cnt === 100) clearInterval(interval);
}, 1000);
```

A5.4.5.3 plotly.js: Streaming data

[DIY] Streaming time series using 30 points update





AAnn_DS_30timestamps.png 로 캡처 저장.

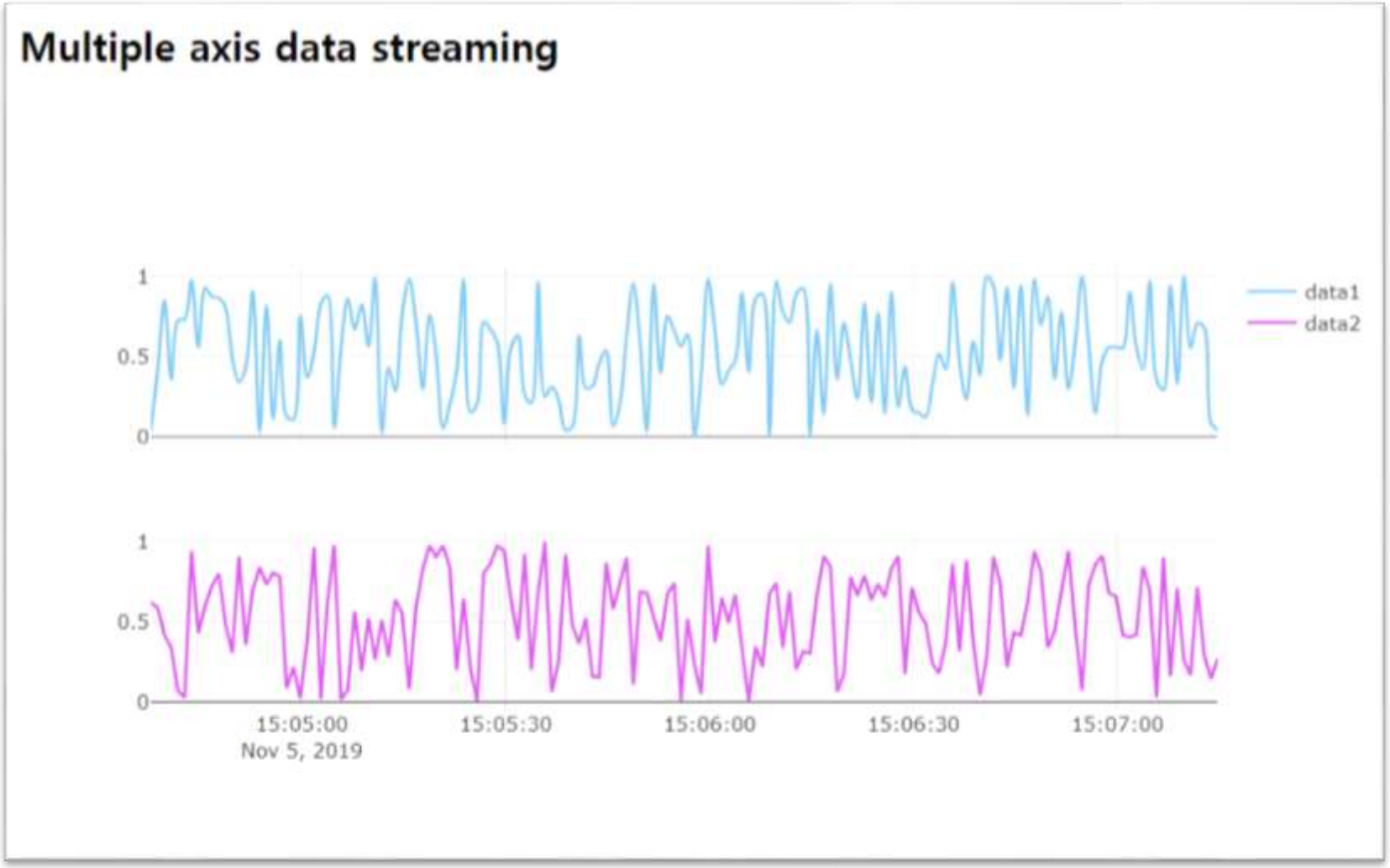
A5.4.5.4 plotly.js: Streaming data

[DIY-hint] Streaming time series using 30 points update

```
<script>
    var arrayLength = 30
    var newArray = []
    var timeArray = []

    // initial 30 data
    for(var i = 0; i < arrayLength; i++) {
        var y = Math.round(Math.random()*10) + 1
        var time = new Date();
        newArray[i] = y
        timeArray[i] = time
    }

    var data = [{
        x: timeArray,
        y: newArray,
        mode: 'lines+markers',
        line: {color: '#80CAF6'},
        marker: {color: '#FC1234'}
    }]


    Plotly.plot('graph', data);
```

A5.4.6 plotly.js: Streaming data

[5] Streaming data using multiple axis



A5.4.6.1 plotly.js: Streaming data

[5.1] Streaming data using multiple axis

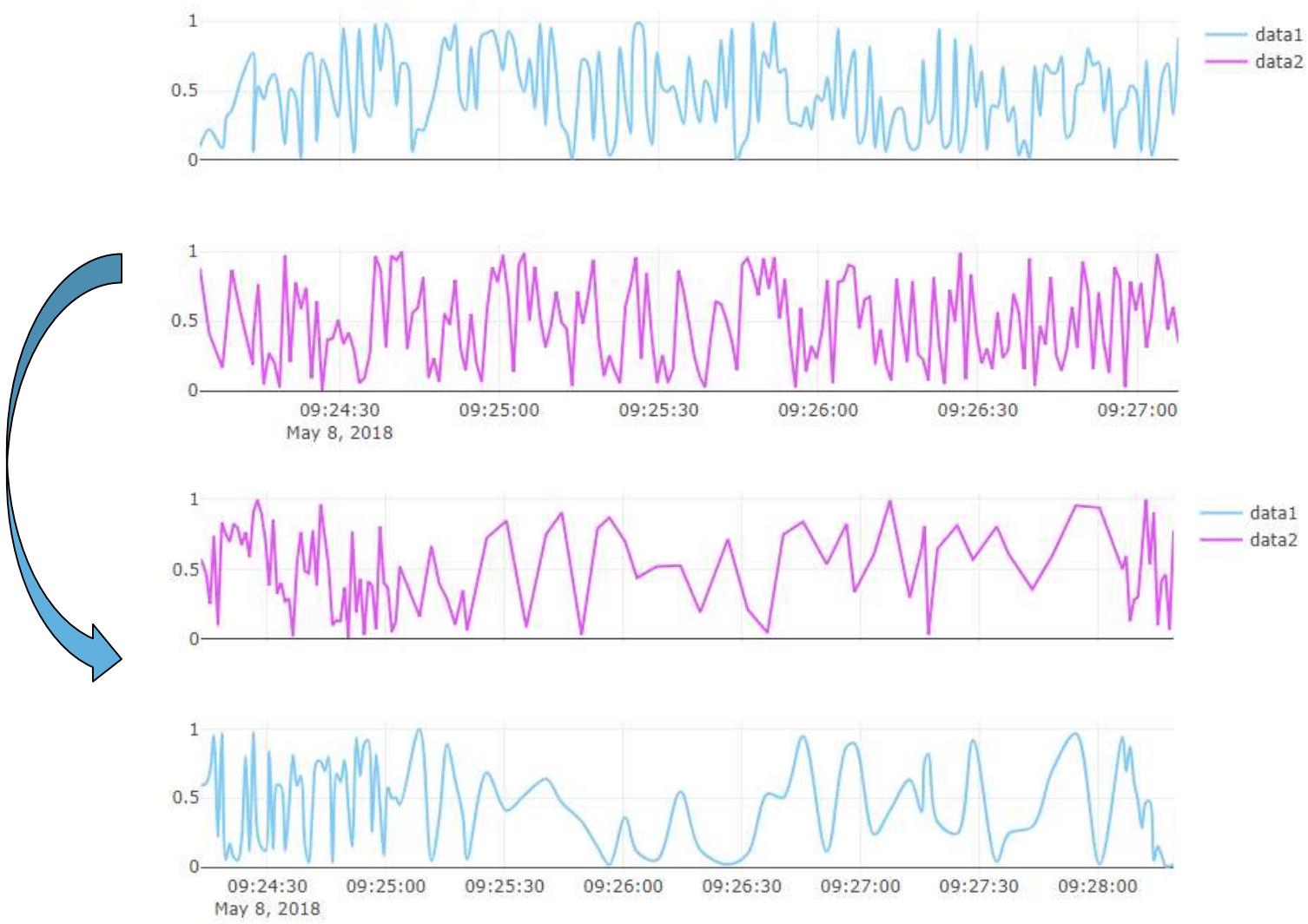
```
<h2>Multiple axis data streaming</h2>

<div id="graph"></div>

<script>
    function rand() {
        return Math.random();
    }

    var time = new Date();

    var trace1 = {
        x: [],
        y: [],
        mode: 'lines',
        line: {
            color: '#80CAF6',
            shape: 'spline'
        },
        name: 'data1'
    }
    var trace2 = {
        x: [],
        y: [],
        xaxis: 'x2',
        yaxis: 'y2',
        mode: 'lines',
        line: {color: '#DF56F1'},
        name: 'data2'
    };
```

```
var layout = {
    xaxis: {
        type: 'date',
        domain: [0, 1],
        showticklabels: false
    },
    yaxis: {domain: [0.6,1]},
    xaxis2: {
        type: 'date',
        anchor: 'y2',
        domain: [0, 1]
    },
    yaxis2: {
        anchor: 'x2',
        domain: [0, 0.4]},
    }

    var data = [trace1,trace2];
    Plotly.plot('graph', data, layout);
```

```
// streaming
var cnt = 0;
var interval = setInterval(function() {

    var time = new Date();

    var update = {
        x: [[time], [time]],
        y: [[rand()], [rand()]]
    }

    Plotly.extendTraces('graph', update, [0,1])
    // cnt++;
    if(cnt === 100) clearInterval(interval);
}, 1000);
```

A5.4.6.2 plotly.js: Streaming data

[DIY] Streaming data using multiple axis → change axis

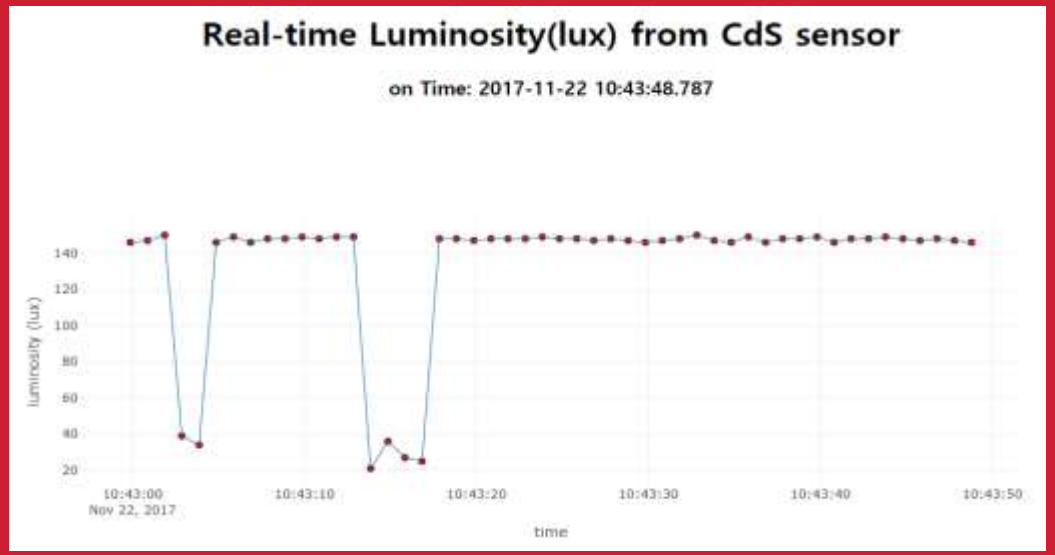


AAnn_DS_multiple_axis.png 로 캡처 저장.

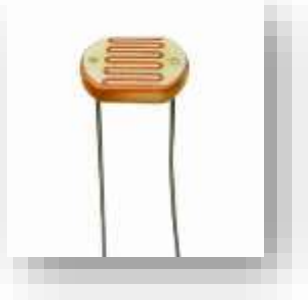
Arduino sensor data

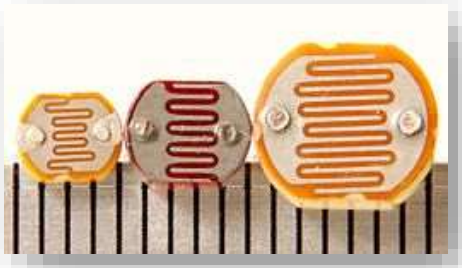
RT visualization using plotly.js

```
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
```

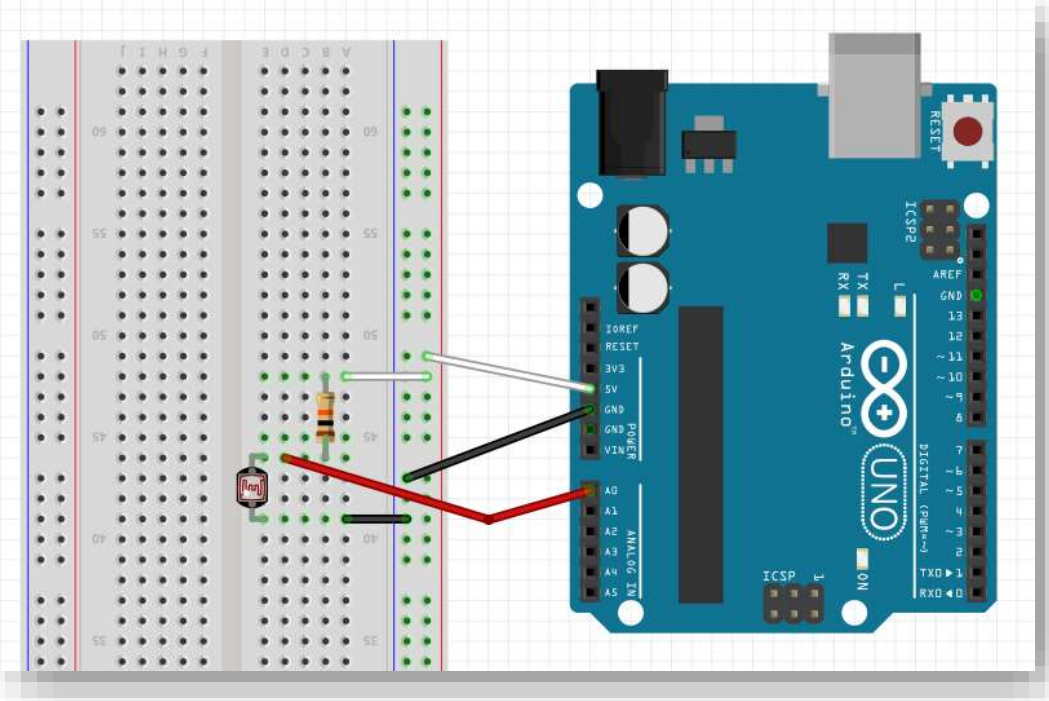


Luminosity sensor [Photocell LDR]

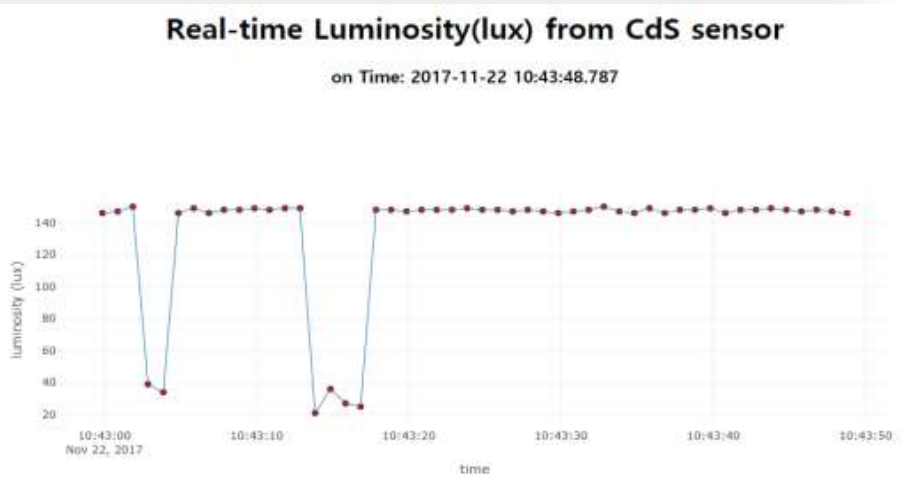




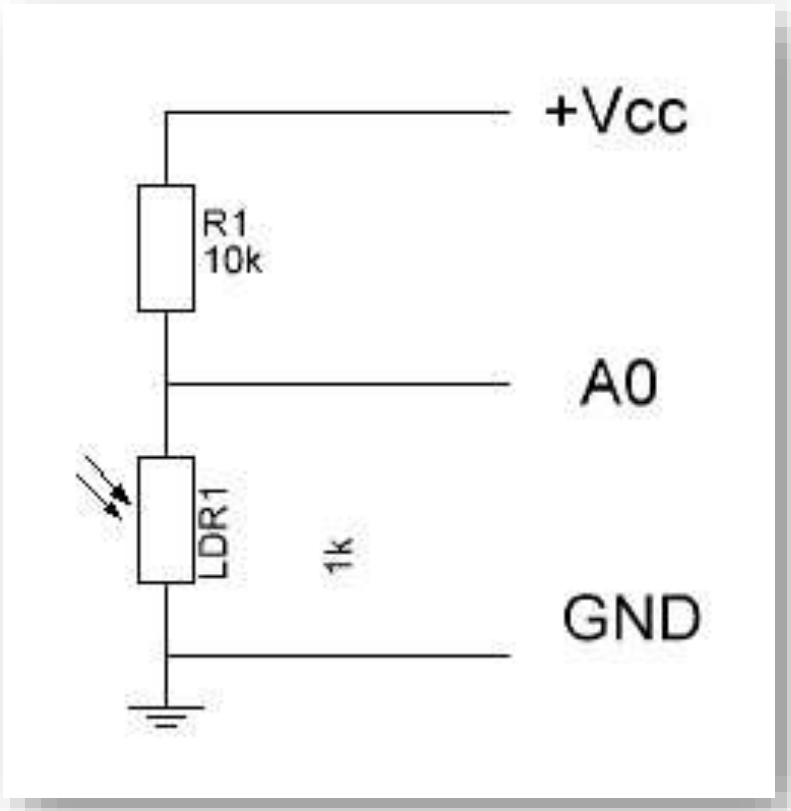
CdS




```
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
```



CdS 센서 회로 분석



Ao → Vo → lux

lux = 500 / R_{ldr}

Vo = I_{ldr} * R_{ldr}

= (5/(10 + R_{ldr}))* R_{ldr}

R_{ldr} = 10*Vo / (5 – Vo)

lux = 250/Vo – 50

Vo = 5.0 * Ao / 1023.0

```
//Voltage to Lux
double luminosity (int RawADC0){
  double Vout=RawADC0*5.0/1023.0;  // 5/1023 (Vin = 5 V)
  double lux=(2500/Vout-500)/10.0;
  // lux = 500 / Rldr, Vout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
  return lux;
}
```

A3.2.6 Luminosity sensor [Photocell LDR]

CdS 센서 회로 - 측정 2.

sketch08_CdS2

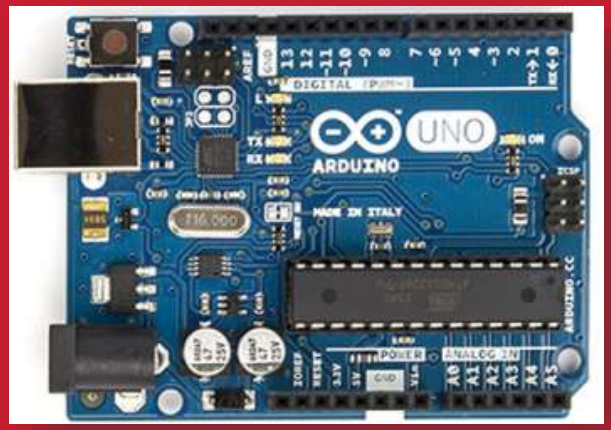
```
1  //  lux
2  #define CDS_INPUT 0
3
4  void setup() {
5  Serial.begin(9600);
6  }
7  void loop() {
8    int value = analogRead(CDS_INPUT);
9    Serial.println(int(luminosity(value)));
10   delay(1000);
11 }
12
13 //Voltage to Lux
14 double luminosity (int RawADC0){
15   double Vout=RawADC0*5.0/1023;  // 5/1023 (Vin = 5 V)
16   double lux=(2500/Vout-500)/10;
17   // lux = 500 / Rldr, Vout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
18   return lux;
19 }
```

COM11 (Arduino/Genuino Uno)

```
23
24
23
183
191
183
185
531
1169
1262
1198
1039
1063
```

밝을수록 측정 값이 커지고
어두을수록 값이 작아진다 !!!

Single sensor: CdS



CdS (LDR)

Node project

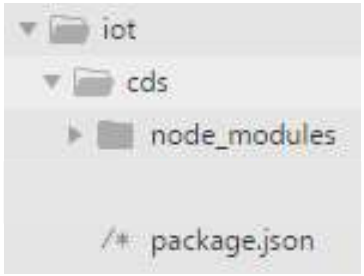
A4.2.1 Luminosity sensor [Photocell LDR]

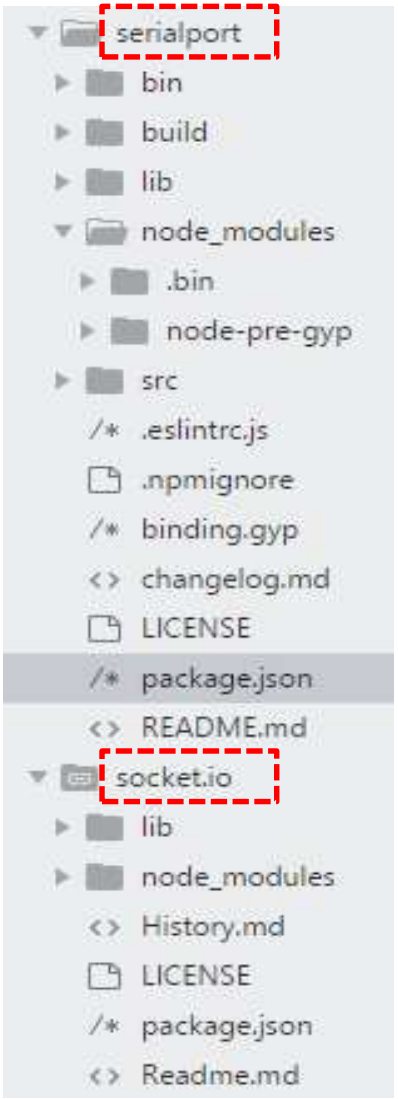
1. Make cds node project
   - ➢ md cds in iot folder
   - ➢ cd cds
2. Go to cds subfolder
   - ➢ npm init

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

A4.2.2 Luminosity sensor [Photocell LDR]

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





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

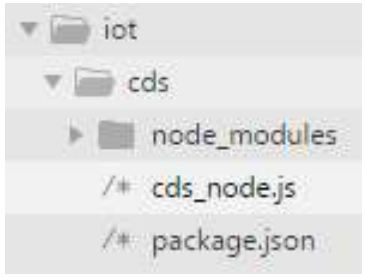
A4.2.3 Luminosity sensor [Photocell LDR]

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

package,json

```
{
  "name": "cds",
  "version": "1.0.0",
  "description": "cds-node project",
  "main": "cds_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"
  }
}
```

A4.2.4 Luminosity sensor [Photocell LDR]



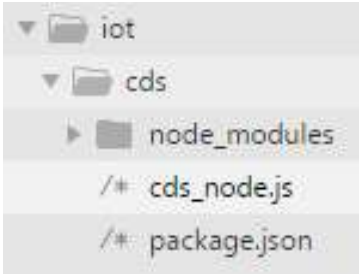
cds_node.js

```
var dStr = '';
var tdata = [];

sp.on('data', function (data) { // call back when data is received
    // raw data only
        //console.log(data);
        dStr = getDateString();
        tdata[0] = dStr;  // date
        tdata[1] = data;  // data
        console.log("AA00," + tdata);
        io.sockets.emit('message', tdata);  // send data to all clients
});

// 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;
}
```

A4.2.5 cds_ node project (실행 결과)

▸ Sublime Text 3에서 실행

```
AA00,2018-01-14 19:12:42.037,86
AA00,2018-01-14 19:12:43.035,36
AA00,2018-01-14 19:12:44.039,54
AA00,2018-01-14 19:12:45.038,175
AA00,2018-01-14 19:12:46.042,175
AA00,2018-01-14 19:12:47.041,174
```

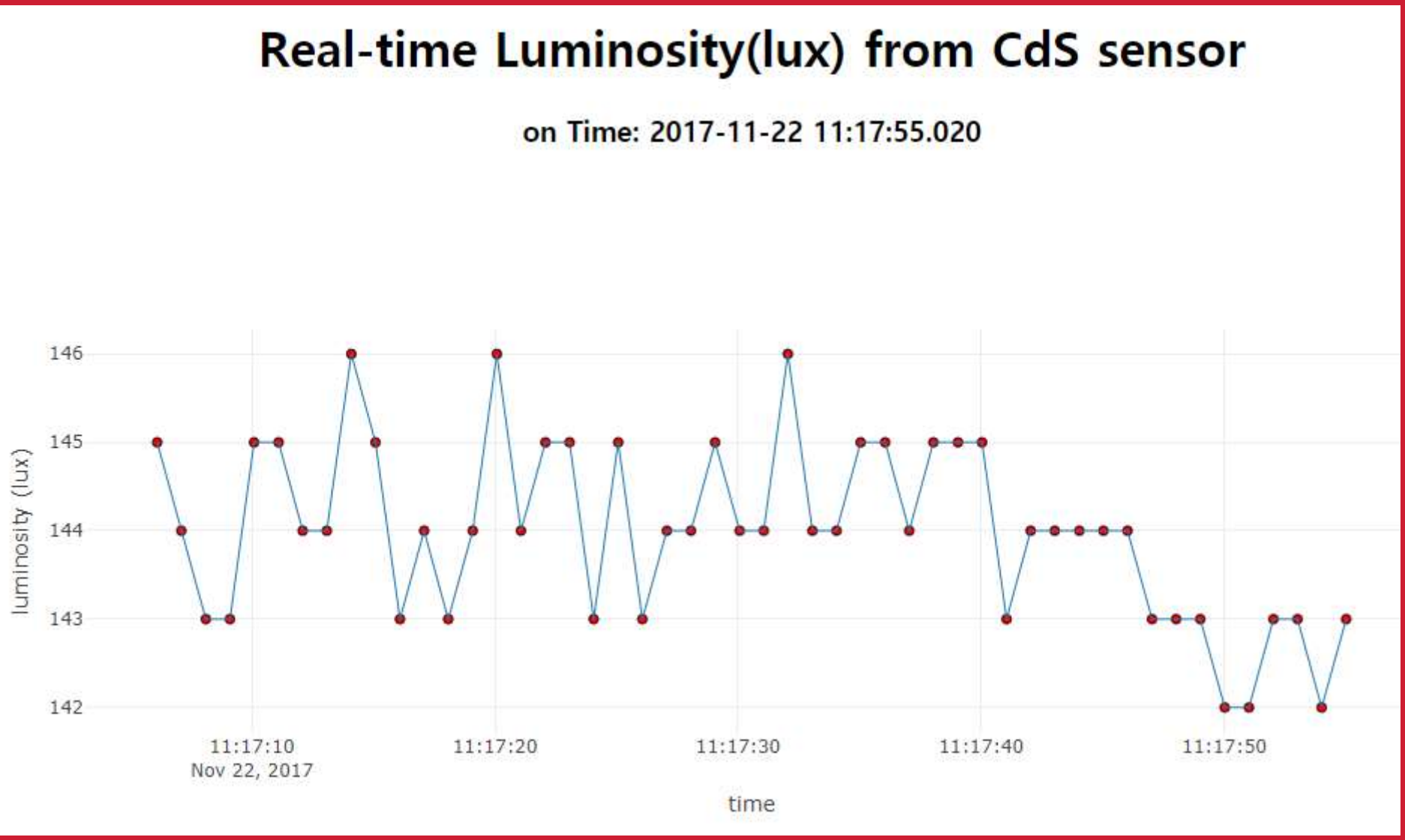


▸ Node cmd에서 실행

```
node cds_node
```

```
NodeJS - node  cds_node

D:\Portable\NodeJSPortable\Data\aa00\iot\cds>node cds_node
AA00,2018-01-14 19:15:33.602,176
AA00,2018-01-14 19:15:34.601,45
AA00,2018-01-14 19:15:35.601,35
AA00,2018-01-14 19:15:36.604,33
AA00,2018-01-14 19:15:37.604,175
```

```
io.sockets.emit('message', tdata);  // send data to all clients
```



A5.5.1 RT sensor-data streaming in Arduino

[1] Client html : client_cds.html (using socket.io.js)

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

[2] Client html : client_cds.html (global variables)

```
<body>  <!-- style="width:100%;height:100%"> -->
<!-- Plotly chart will be drawn inside this DIV -->
<h1 align="center"> Real-time Luminosity(lux) from CdS sensor </h1>

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

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

<hr>

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

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

A5.5.3 RT sensor-data streaming in Arduino

[3] Client html : client_cds.html (socket connection & handling message)

```
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]);  // lux
            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];
        nextPt();
    });
});
```

[4] Client html : client_cds.html (init() & nextPt())

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

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

function nextPt() {

    tArray.shift();
    tArray.push(dtda[0]);

    xTrack.shift();
    xTrack.push(dtda[1]);  // CdS sensor: lux

    Plotly.redraw(streamPlot);
}
```

[5] Client html : client_cds.html (data & layout)

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

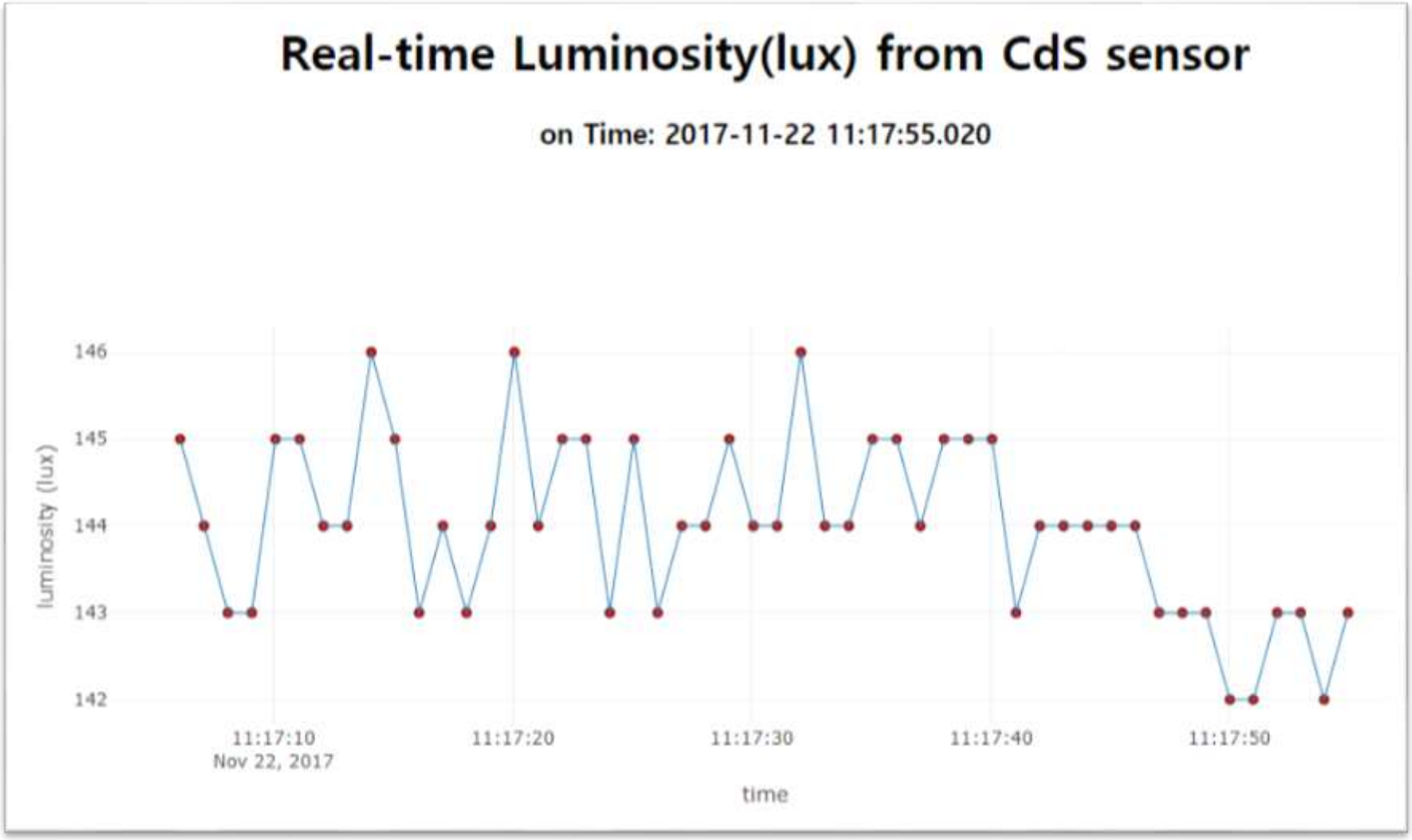
```
// layout
var layout = {
    xaxis : {
        title : 'time',
        domain : [0, 1]
    },
    yaxis : {
        title : 'luminosity (lux)',
        domain : [0, 1],
        range : [0, 500]
    }
};
```

domain: [0,1] → x 또는 y 축을 100% 사용

range: [0,500] → y 축의 범위를 0~500 설정

A5.5.6 RT sensor-data streaming in Arduino

[6] Client html : client_cds.html (real time monitoring of the luminosity)



A5.5.7.1 RT sensor-data streaming in Arduino

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

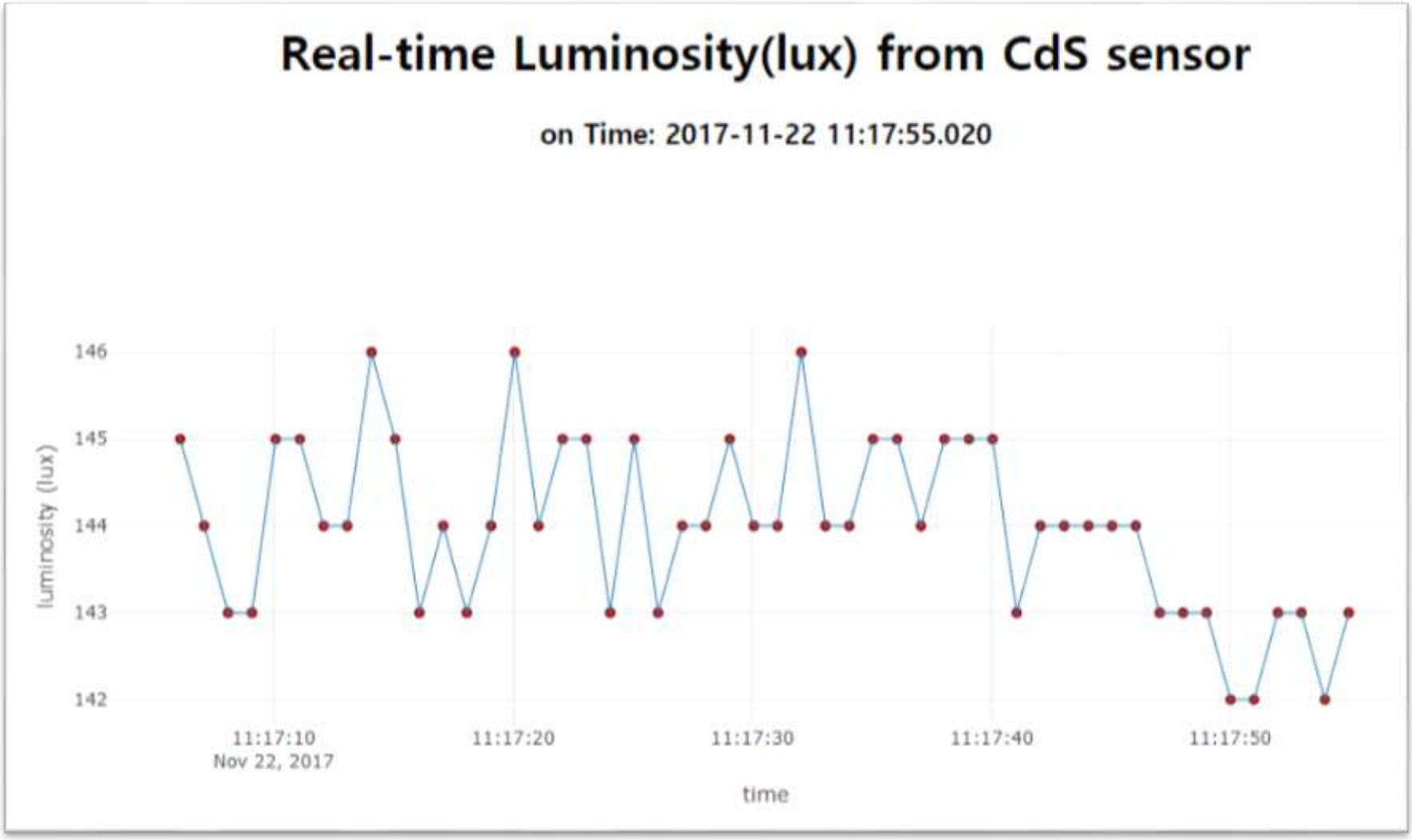
```
/* function nextPt() {

    tArray.shift();
    tArray.push(dtda[0]);

    xTrack.shift();
    xTrack.push(dtda[1]);  //

    Plotly.redraw(streamPlot);
}       */
```

nextPt() 주석 처리

```
socket.on('connect', function () {
    socket.on('message', function (msg) {
        // initial plot
        if(msg[0]!='' && initFlag){
            dtda[0]=msg[0];
            dtda[1]=parseInt(msg[1]);  // lux
            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 streami
        ctime.innerHTML = dtda[0];
        //nextPt();
        tArray = tArray.concat(dtda[0]);  // time
        tArray.splice(0,1);
        xTrack = xTrack.concat(dtda[1]);  // lux
        xTrack.splice(0,1);

        var update = {
            x: [tArray],
            y: [xTrack]
        }

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

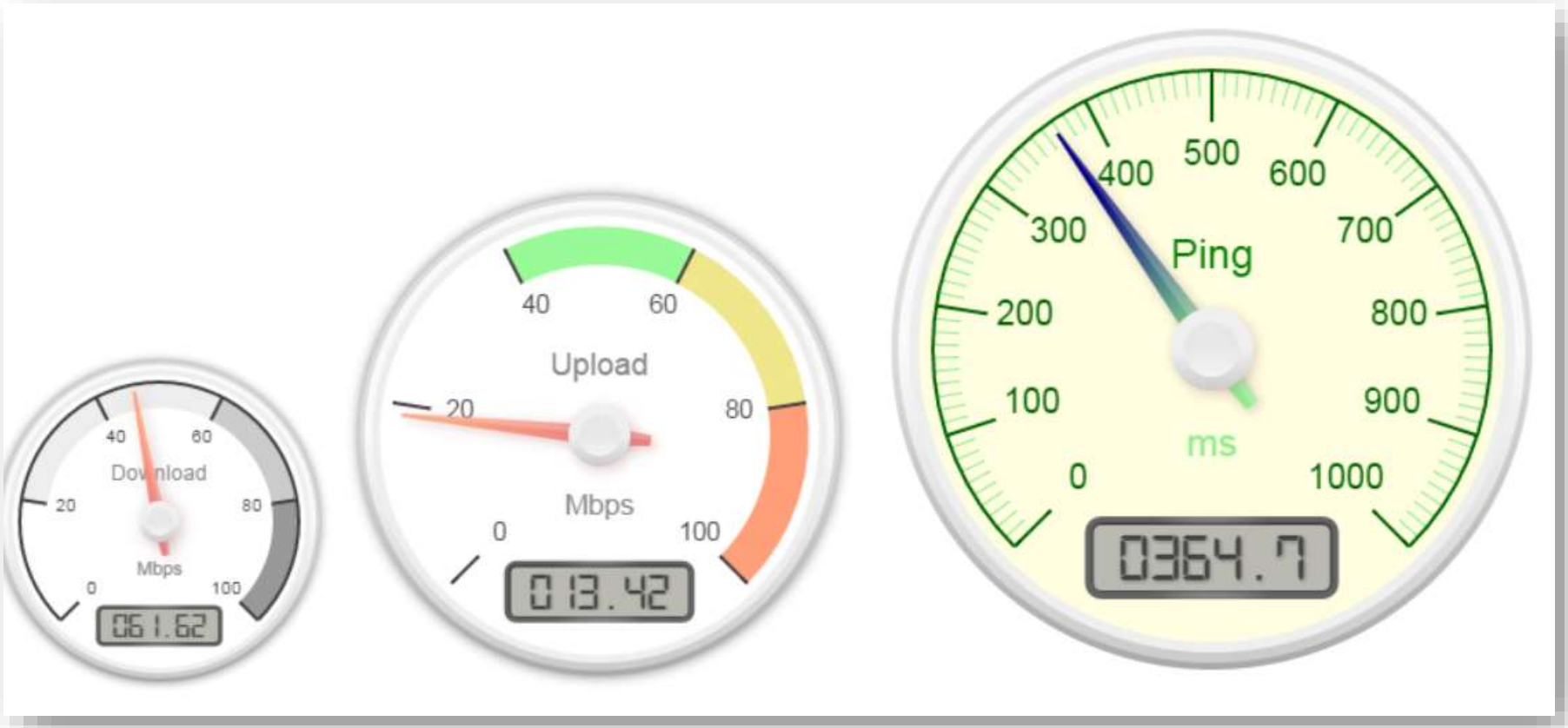
A5.5.7.2 RT sensor-data streaming in Arduino

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



Canvas Gauge

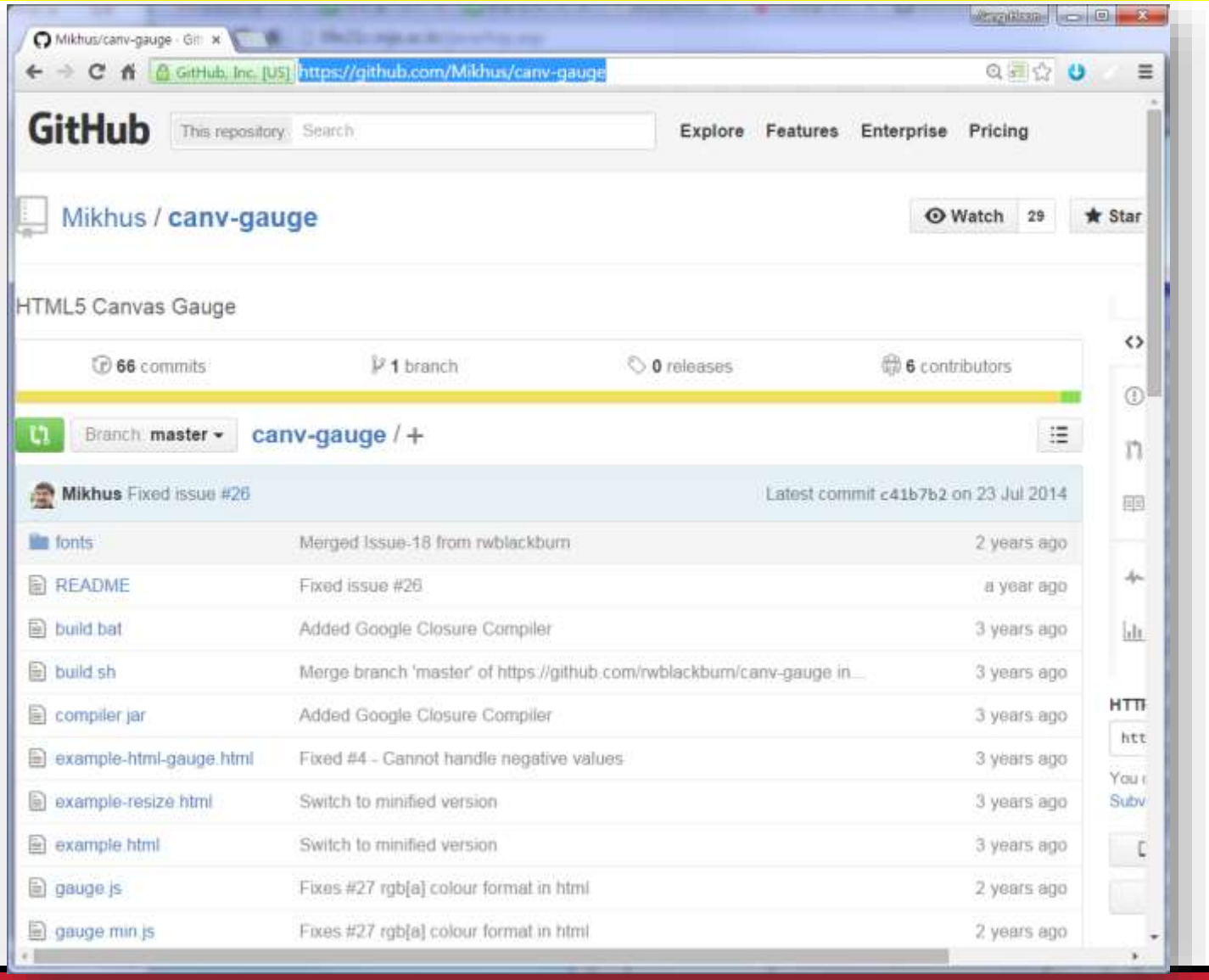
[1] Canvas gauge javascript library : example



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

Canvas Gauge

[2] Canvas gauge javascript library : gauge.js



https://github.com/Mikhus/canvas-gauges

A5.5.8.1 RT sensor-data streaming in Arduino

[DIY] Client html : client_cds_gauge.html (add Gauge)

```
<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>
```

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

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

A5.5.8.2 RT sensor-data streaming in Arduino

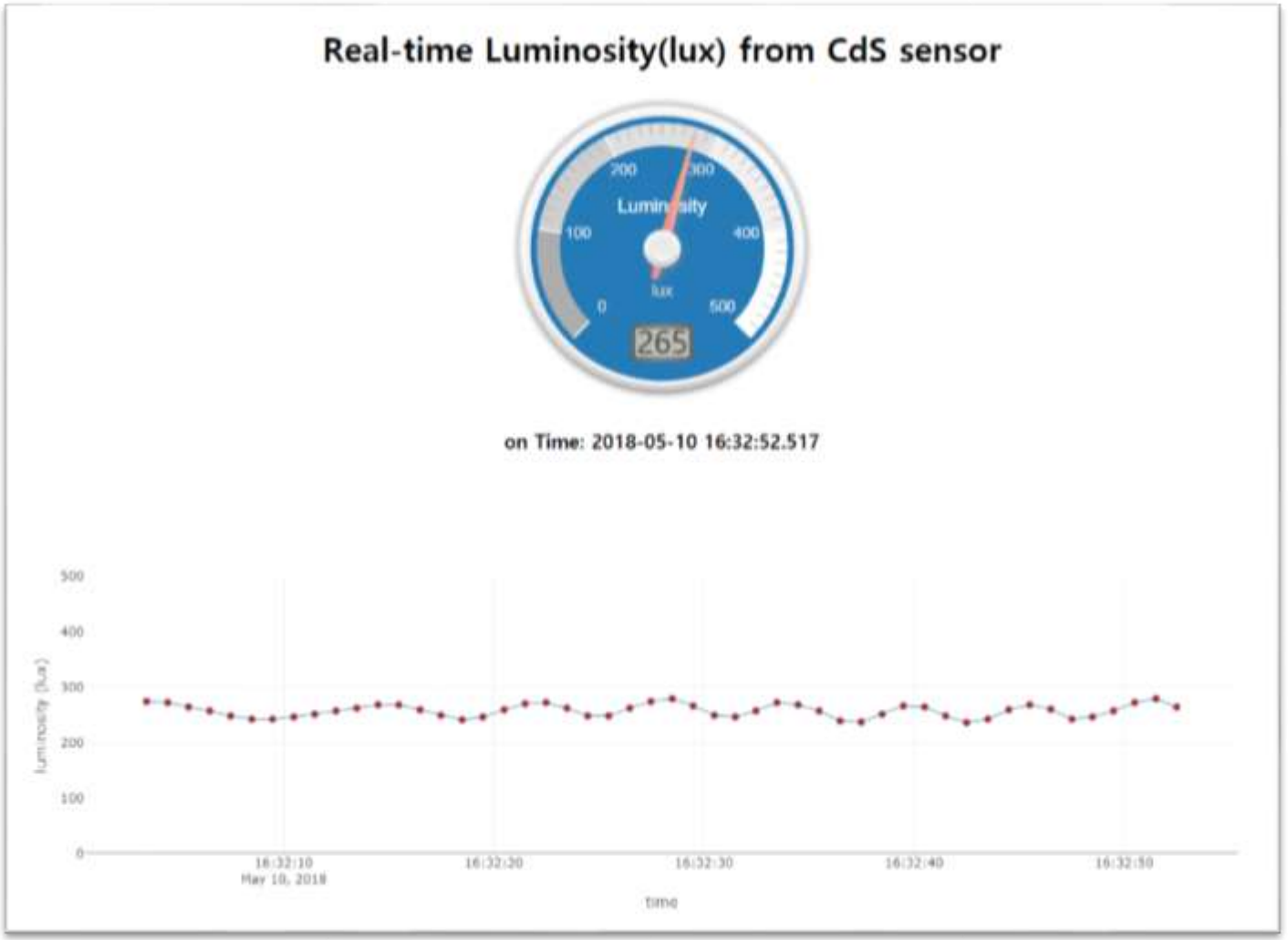
[DIY] Client html : client_cds_gauge.html (add Gauge)

```
socket.on('connect', function () {
    socket.on('message', function (msg) {
        // initial plot
        if(msg[0]!='' && initFlag){
            dtda[0]=msg[0];
            dtda[1]=parseInt(msg[1]);  // lux
            init();  // start streaming
            initFlag=false;
        }
        console.log(msg[0]);
        console.log(parseInt(msg[1])); // Conv
        dtda[0]=msg[0];
        dtda[1] = parseInt(msg[1]);

        // when new data is coming, keep on st
        ctime.innerHTML = dtda[0];
        gauge_lux.setValue(dtda[1]); // lux ga
        //nextPt();
        tArray = tArray.concat(dtda[0]);  // t
        tArray.splice(0,1);
        xTrack = xTrack.concat(dtda[1]);  // l
        xTrack.splice(0,1);

        var update = {
            x: [tArray],
            y: [xTrack]
        }

        Plotly.update(streamPlot, update);

    });
});
```

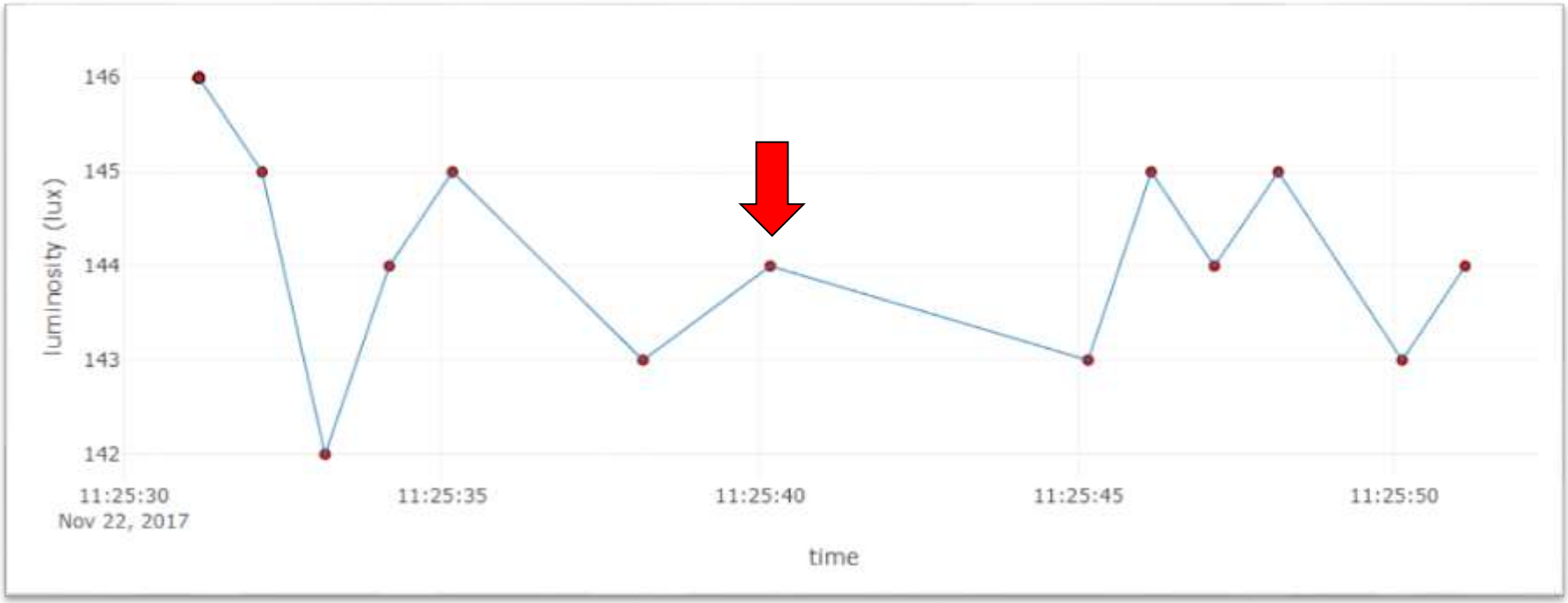
```
var gauge_lux = new Gauge({
    renderTo    : 'gauge',
    width       : 300,
    height      : 300,
    glow        : true,
    units       : 'lux',
    valueFormat : { int : 2, dec : 0 },
    title       : "Luminosity",
    minValue    : 0,
    maxValue    : 500,  // new
    majorTicks  : ['0','100','200','300','400','500'],
    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();
```

[DIY] Client html : client_cds_gauge.html (change design of Gauge)

변경된 디자인으로 된 그래프를 캡처하여 AAnn_cds_gauge.png 로 저장



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



이상 감지 (anomaly detection)

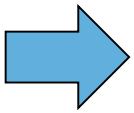
입력되는 lux 값이 변하는 경우에만 그래프를 그림.

실시간 모니터링에서 이상 감지 기능이 필요함.

밝기 값 변화의 문턱값을 설정해서 이상 감지 기능 구현

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

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

var update = {
    x: [tArray],
    y: [xTrack]
}

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

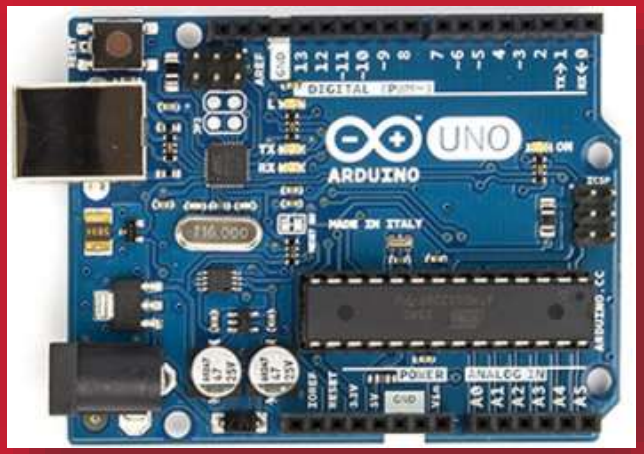


```
// Only when the value of lux is different
// from the previous one, the screen is redrawed.
if (dtda[1] != preX) {  // any change?
    preX = dtda[1];

    ctime.innerHTML = dtda[0];
    gauge_lux.setValue(dtda[1]); // lux gauge
    //nextPt();
    tArray = tArray.concat(dtda[0]);  // time
    tArray.splice(0,1);
    xTrack = xTrack.concat(dtda[1]);  // lux
    xTrack.splice(0,1);

    var update = {
        x: [tArray],
        y: [xTrack]
    }

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

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

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

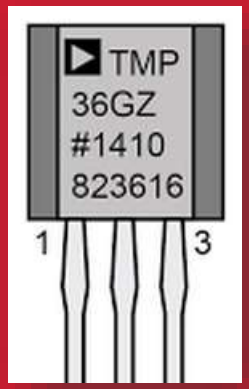
Multiple sensors



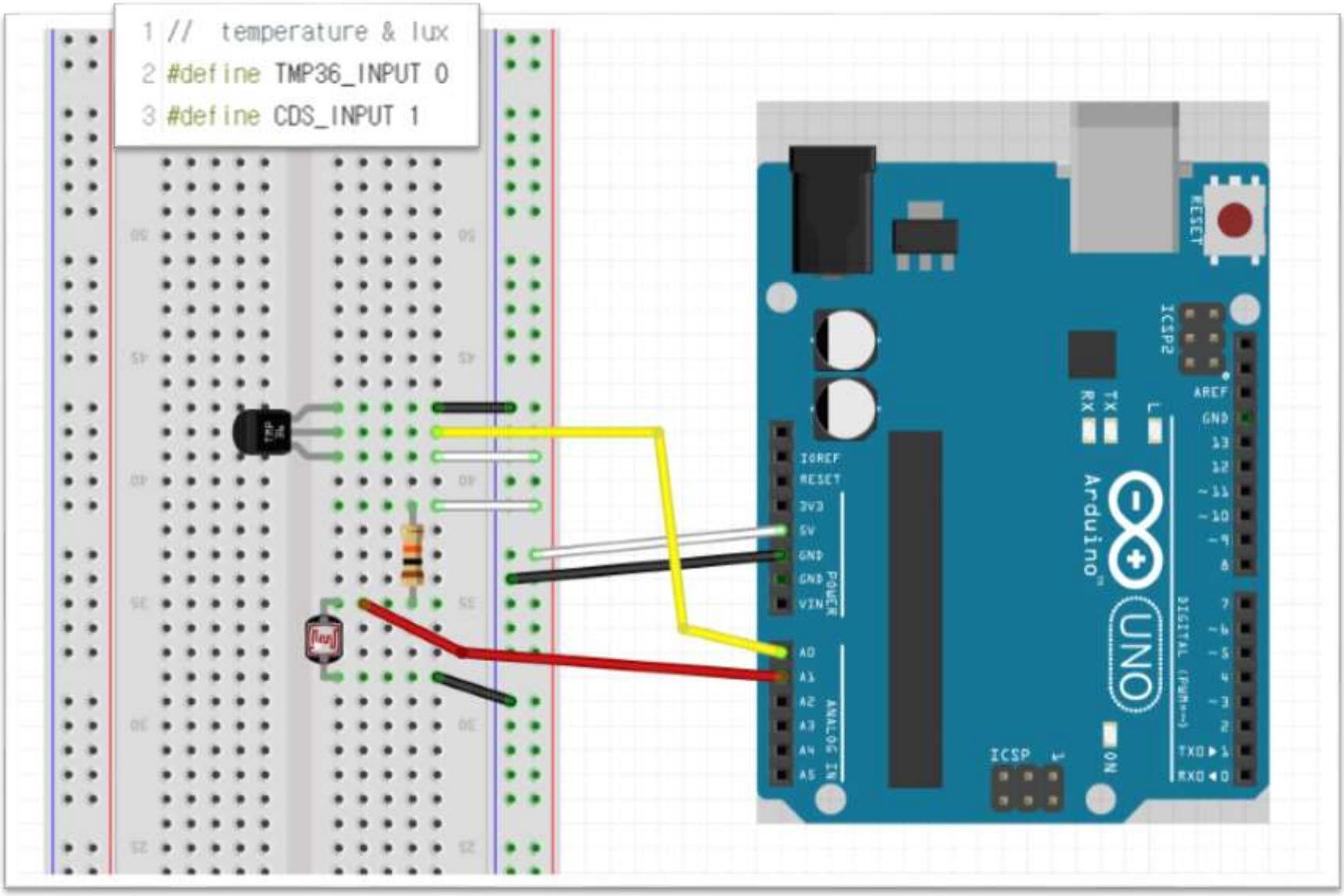
CdS + TMP36

+ plotly.js

Node project



A4.3.1 TMP36 + CdS : circuit



A4.3.2 TMP36 + CdS : code

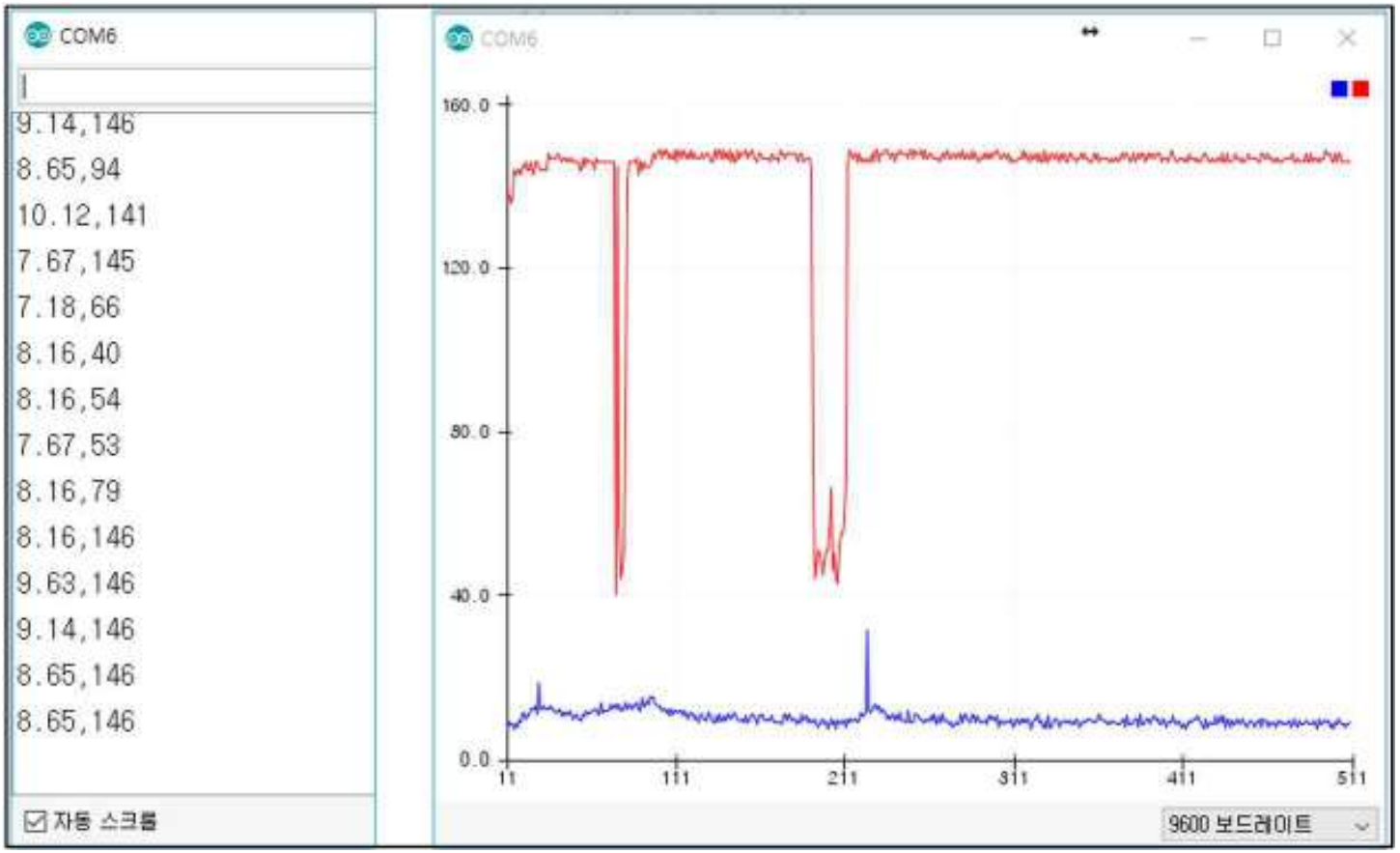
AAnn_TMP36_CdS§

```
1  //  temperature & lux
2  #define TMP36_INPUT 0
3  #define CDS_INPUT 1
4
5  void setup() {
6    Serial.begin(9600);
7  }
```

AAnn_tmp36_cds.ino

```
8  void loop() {
9    // Temperature from TMP36
10   int temp_value = analogRead(TMP36_INPUT);
11   // converting that reading to voltage
12   float voltage = temp_value * 5.0 * 1000;  // in mV
13   voltage /= 1023.0;
14   float tempC = (voltage - 500) / 10 ;
15
16   // Lux from CdS (LDR)
17   int cds_value = analogRead(CDS_INPUT);
18   int lux = int(luminosity(cds_value));
19 //  Serial.print("HSnn,");
20   Serial.print(tempC);
21   Serial.print(",");
22   Serial.println(lux);
23
24   delay(1000);
25 }
26
27 //Voltage to Lux
28 double luminosity (int RawADC0){
29   double Vout=RawADC0*5.0/1023.0;  // 5/1023 (Vin = 5 V)
30   int lux=(2500/Vout-500)/10;
31   // lux = 500 / Rldr, Vout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
32   return lux;
33 }
```

A4.3.2 TMP36 + CdS : result



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

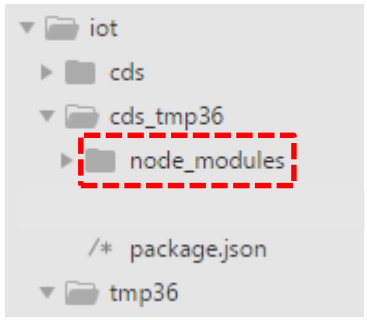
description : cds-tmp36-node project

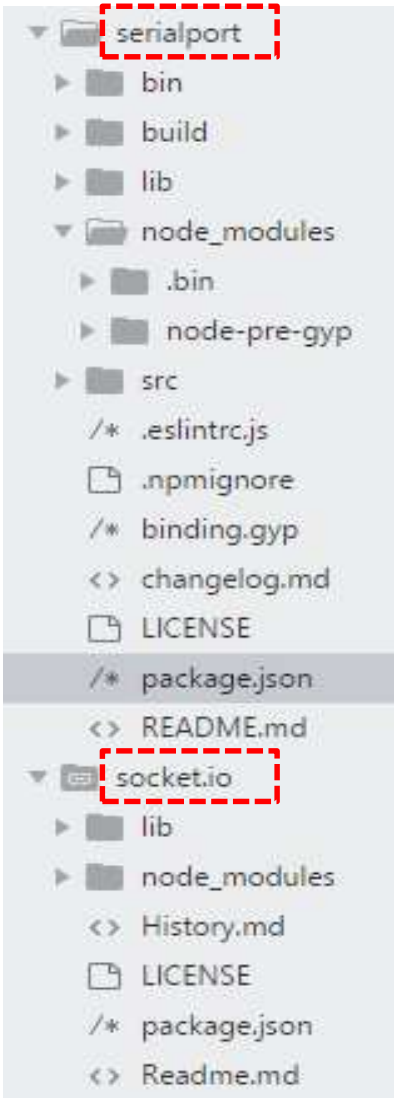
entry point : cds_tmp36_node.js

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**





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**

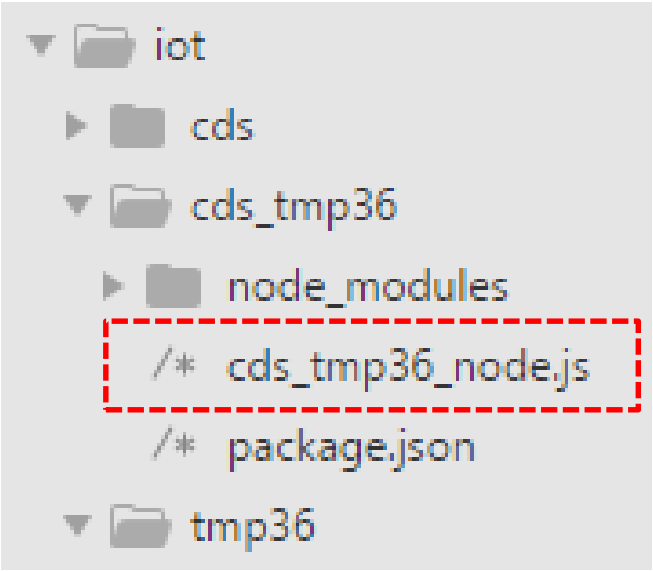
package,json

```
{
  "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



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

cds_tmp36_node.js

cds_tmp36_node.js

```
1  // cds_tmp36_node.js
2
3  var serialport = require('serialport');
4  var portName = 'COM6';  // check your COM port!!
5  var port     =   process.env.PORT || 3000;
6
7  var io = require('socket.io').listen(port);
8
9  // serial port object
10 var sp = new serialport(portName,{
11     baudRate: 9600,   // 9600  38400
12     dataBits: 8,
13     parity: 'none',
14     stopBits: 1,
15     flowControl: false,
16     parser: serialport.parsers.readline('\r\n')
17 });
```

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

cds_tmp36_node.js – parsing data

```
18 var dStr = '';
19 var readData = '';  // this stores the buffer
20 var temp ='';
21 var lux ='';
22 var mdata =[]; // this array stores date and data from multiple sensors
23 var firstcommaidx = 0;
24
25 sp.on('data', function (data) { // call back when data is received
26     readData = data.toString(); // append data to buffer
27     firstcommaidx = readData.indexOf(',');
28
29     // parsing data into signals
30     if (firstcommaidx > 0) {
31         temp = readData.substring(0, firstcommaidx);
32         lux = readData.substring(firstcommaidx + 1);
33         readData = '';
34
35         dStr = getDateString();
36         mdata[0]=dStr;  // Date
37         mdata[1]=temp;  // temperature data
38         mdata[2]=lux;   // luminosity data
39         console.log("HSnn," + mdata);
40         io.sockets.emit('message', mdata);  // send data to all clients
41
42     } else {  // error
43         console.log(readData);
44     }
45 });
```

Parsing Data

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

cds_tmp36_node.js

```
32 // helper function to get a nicely formatted date string for IOT
33 function getDateString() {
34     var time = new Date().getTime();
35     // 32400000 is (GMT+9 Korea, GimHae)
36     // for your timezone just multiply +/-GMT by 3600000
37     var datestr = new Date(time +32400000).
38     toISOString().replace(/T/, ' ').replace(/Z/, '');
39     return datestr;
40 }
41
42 io.sockets.on('connection', function (socket) {
43     // If socket.io receives message from the client browser then
44     // this call back will be executed.
45     socket.on('message', function (msg) {
46         console.log(msg);
47     });
48     // If a web browser disconnects from Socket.IO then this callback is called.
49     socket.on('disconnect', function () {
50         console.log('disconnected');
51     });
52 });
```

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

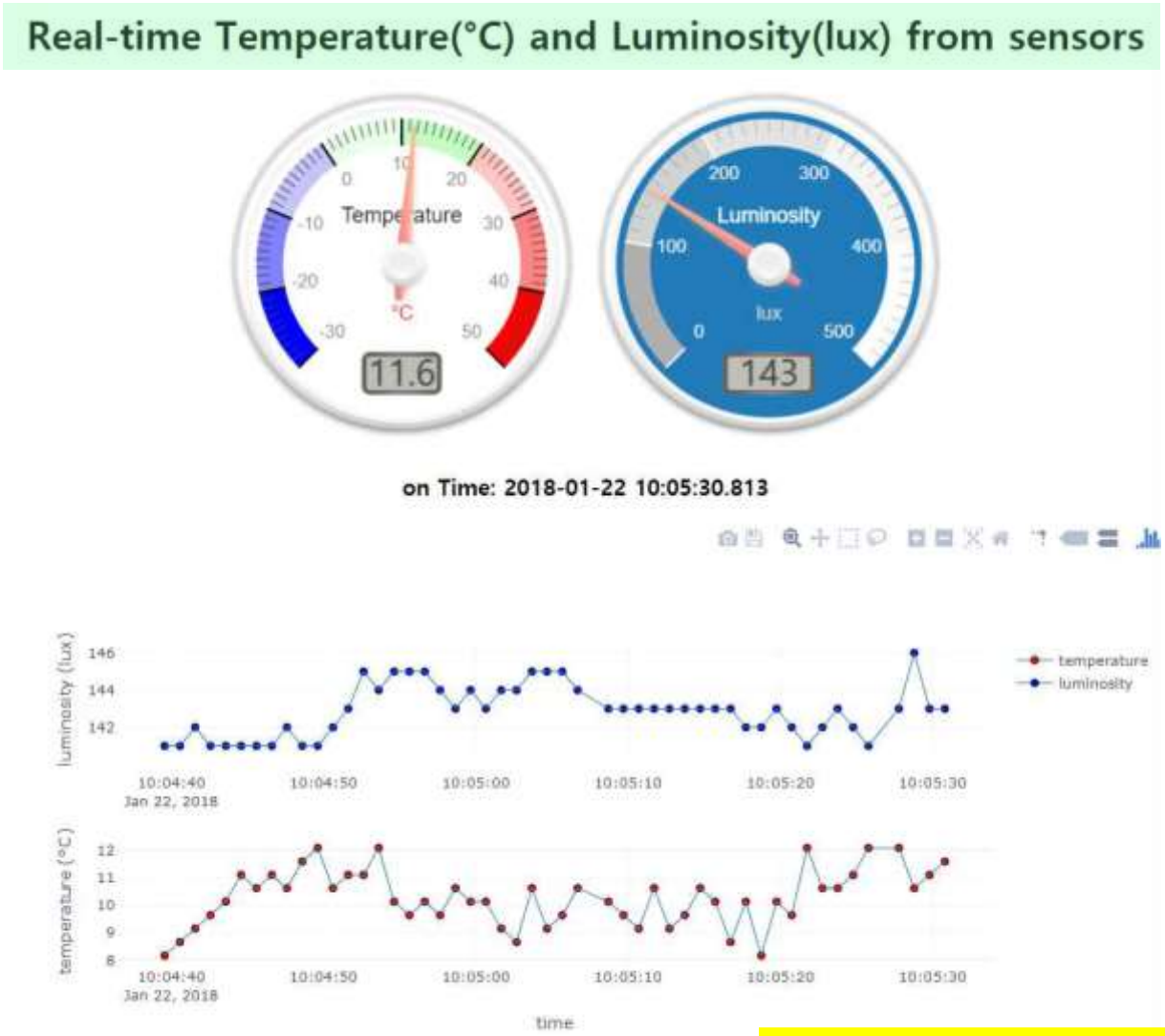
[DIY] Client html : client_cds_tmp36.html (data from multi sensors)

```
// 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)



AAnn_DS_cds_tmp36.png 로 저장

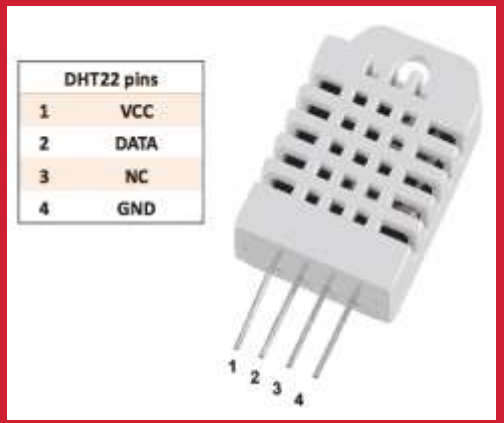
CdS + DHT22

+ plotly.js

Node project

Multi-sensors

DHT22 + CdS



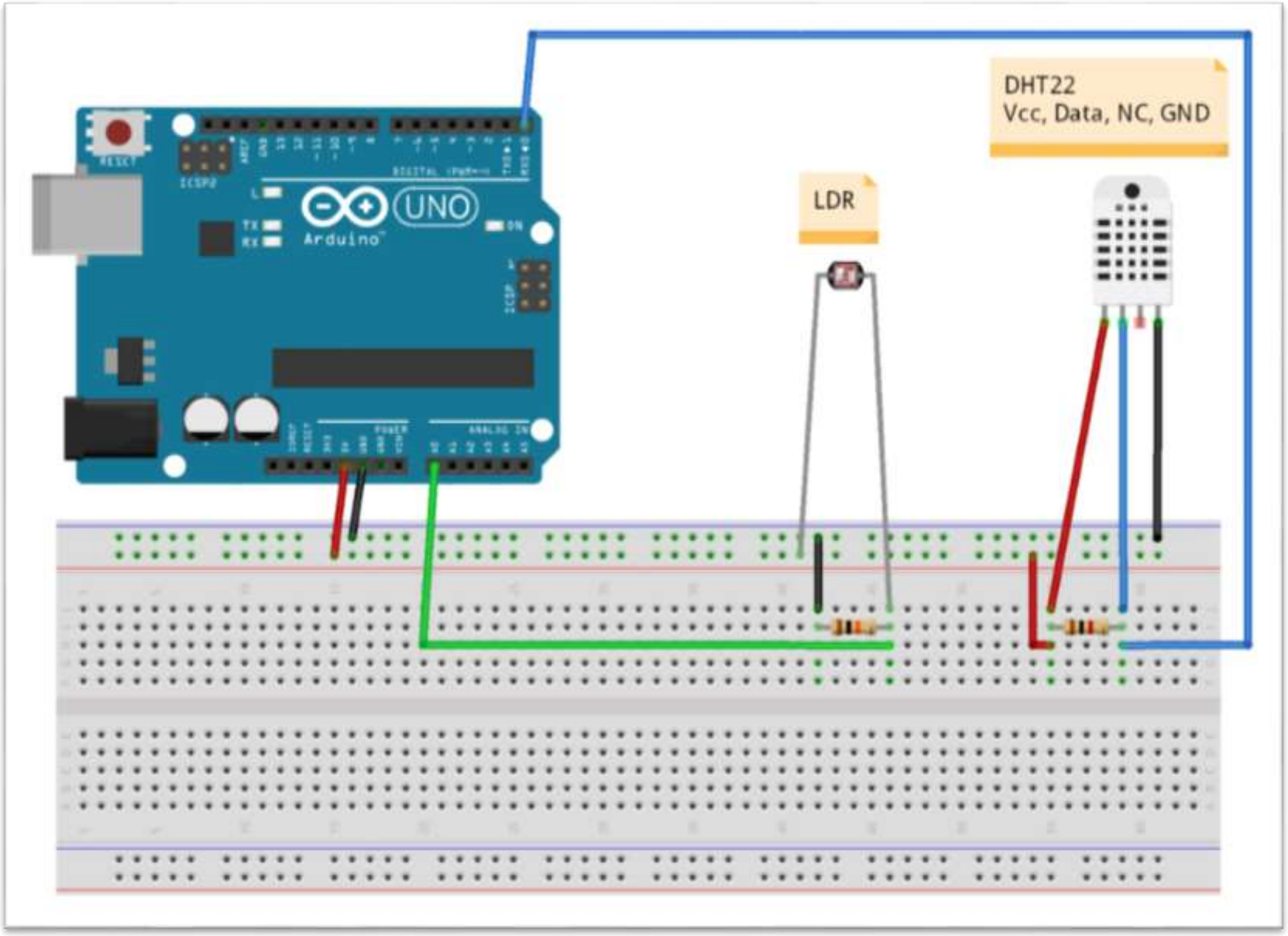
DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

그림 8-7 DHT22 pin 구조

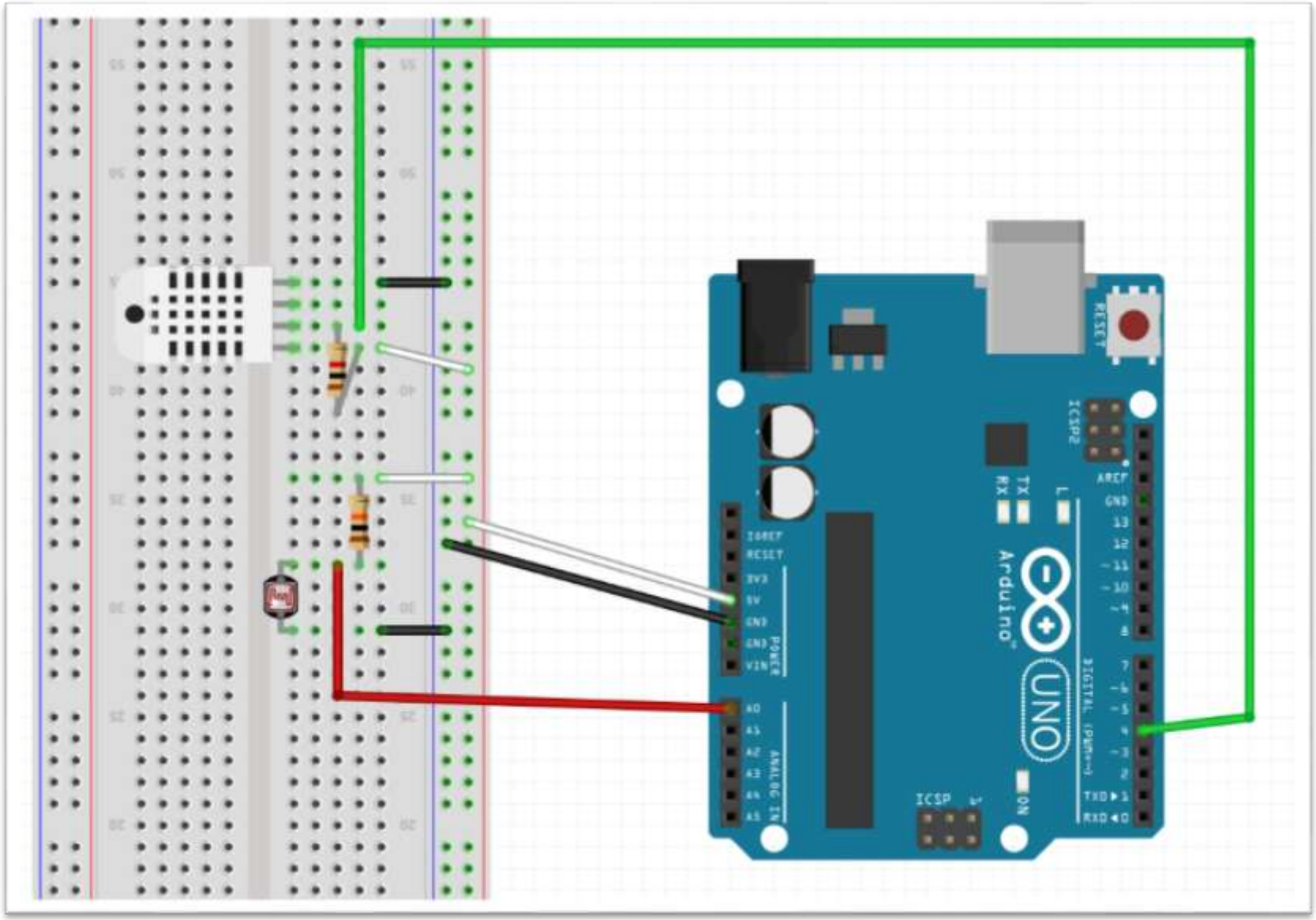
- 3 ~ 5V power and I/O
- 2.5mA max current
- [0-100%] humidity readings with 2-5% accuracy
- [-40 to 80°C] temperature readings ±0.5°C accuracy
- 0.5 Hz sampling rate

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

A5.7 DHT22 + CdS streaming project



A5.7.1 DHT22 + CdS circuit

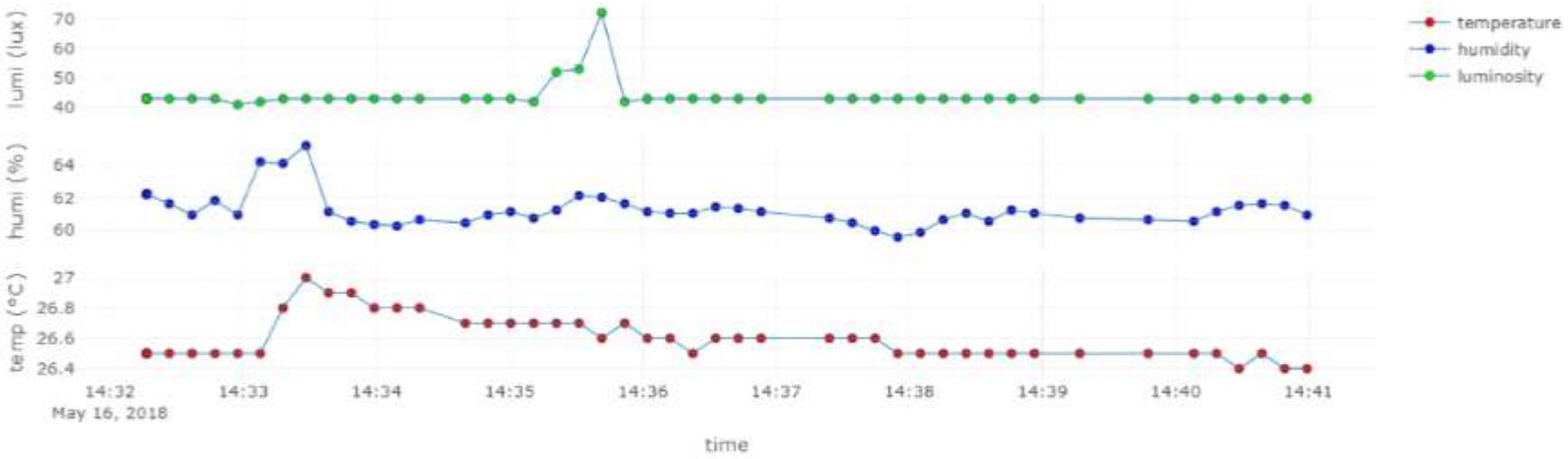


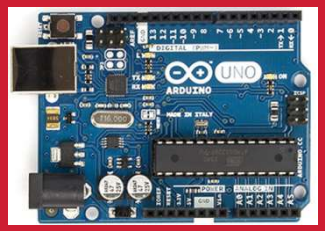
Real-time Weather Station from sensors

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

Save as

AAnn_cds_dht22.png





[Practice]

- [wk10]
  - RT Data Visualization with node.js
  - Usage of gauge.js
  - Complete your plotly-node project
  - Upload folder: AAnn_Rpt08

wk10 : Practice : AAnn_Rpt08

- [Target of this week]
  - Complete your works
  - Save your outcomes and upload outputs in github

제출폴더명 : **AAnn_Rpt07**

- 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

⑥ All *.ino

⑦ All *.js

⑧ All *.html

Email : chaos21c@gmail.com

[Upload to github]

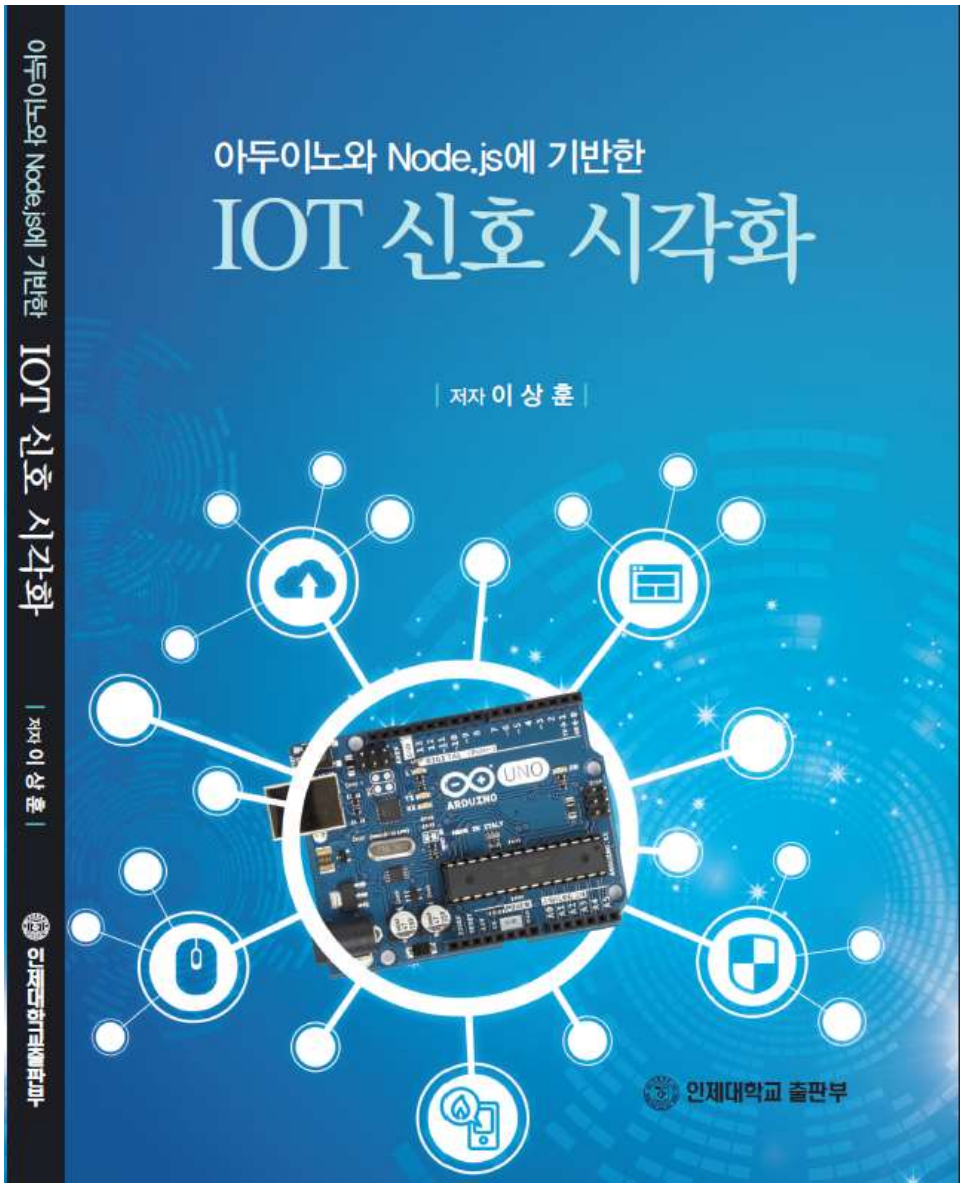
- [wk10]
  - upload all work of this week
  - Use repo “aann” in github
  - upload folder “aann_rpt08” in your github.

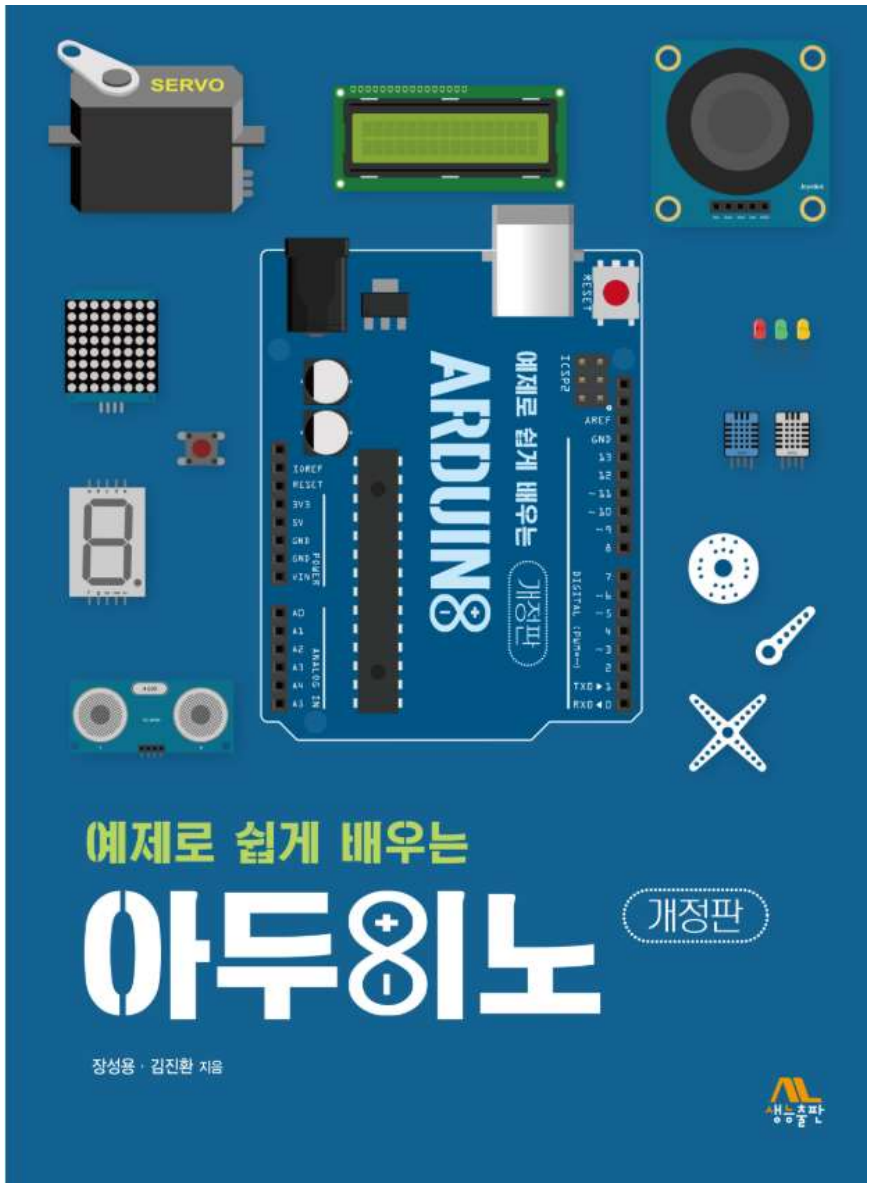
Lecture materials

References & good sites

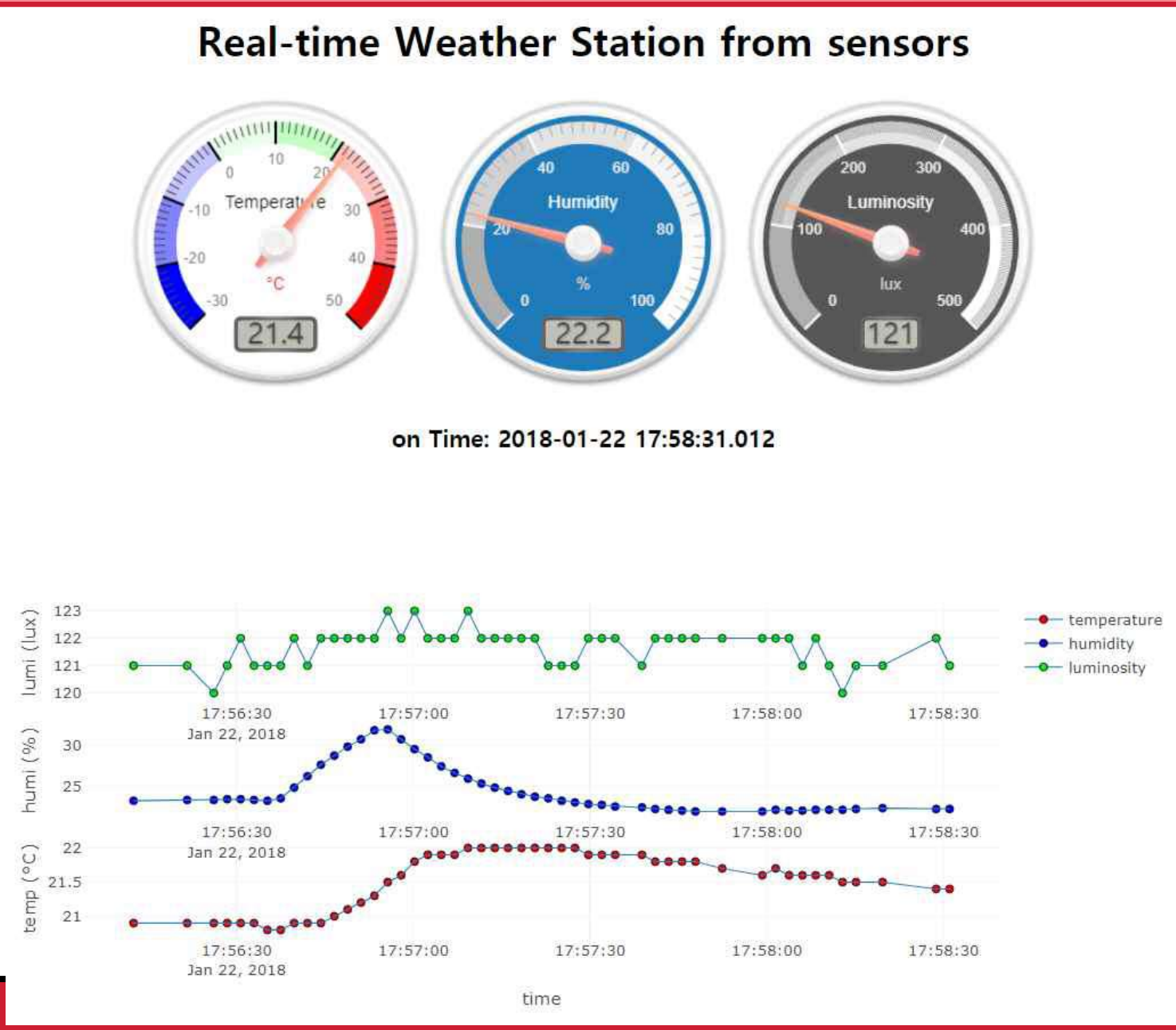
- ✓ http://www.arduino.cc Arduino Homepage
- ✓ http://www.nodejs.org/ko Node.js
- ✓ https://plot.ly/ plotly
- ✓ https://www.mongodb.com/ MongoDB
- ✓ http://www.w3schools.com By w3schools
- ✓ http://www.github.com GitHub

주교재 및 참고도서


아두이노와 Node.js에 기반한
IOT 신호 시각화
| 저자 이 상 훈 |
인제대학교 출판부
아두이노와 Node.js에 기반한 IOT 신호 시각화
| 저자 이 상 훈 |
인제대학교 출판부


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

Target of this class



Another target of this class