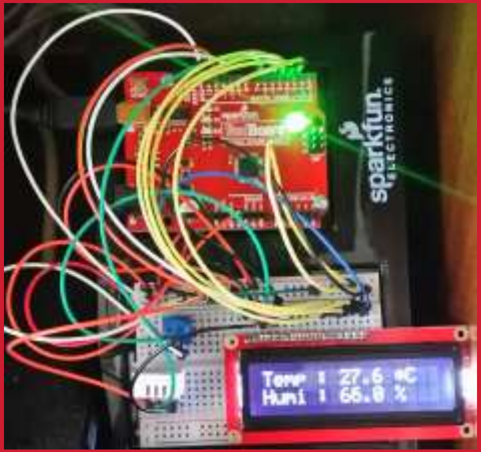
Arduino-IOT

[wk07-08]

Data Visualization - plotly.js

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



Drone-IoT-Comsi, INJE University

2nd semester, 2020

Email : chaos21c@gmail.com

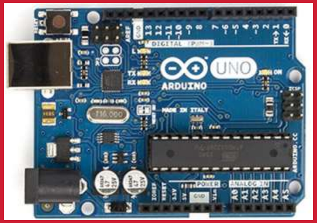
My ID

1분반-목요일 (2학년)

- AA1-01: 강서현
- AA1-02: 강태민
- AA1-03: 김세은
- AA1-04: 여수민
- AA1-05: 정영훈
- AA1-06: 차혁준
- AA1-07: 하태헌
- AA1-08: 김경욱
- AA1-09: 김민욱
- AA1-10: 김민성
- AA1-11: 김민준
- AA1-12: 김인수
- AA1-13: 김현식
- AA1-14: 장성운
- AA1-15: 전승진
- AA1-16: 정희철
- AA1-17: 조동현
- AA1-18: 전동빈
- AA1-19: 신종원

2분반-수요일 (3학년)

- AA2-01: 강민수
- AA2-02: 구병준
- AA2-03: 김종민
- AA2-04: 박성철
- AA2-05: 이승현
- AA2-06: 이창호
- AA2-07: 손성빈
- AA2-08: 안예찬
- AA2-09: 유종인
- AA2-10: 이석민
- AA2-11: 이정문
- AA2-12: 이주원
- AA2-13: 정재영
- AA2-14: 하태성
- AA2-15: 김경미
- AA2-16: 김규년
- AA2-17: 김유빈
- AA2-18: 송다은
- AA2-19: 정주은
- AA2-20: 권준표



[Review]

- [wk07/08]
  - charts by plotly
  - Complete your project
  - Upload folder: aax-nn-rpt07
  - Use repo “aax-nn” in github

wk07 : Practice : AAnn_Rpt07

◆ [Target of this week]

- Complete your works
- Save your outcomes and upload outputs in github

제출폴더명 : **aax-nn-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 *.html**

Email : chaos21c@gmail.com

IOT: HSC

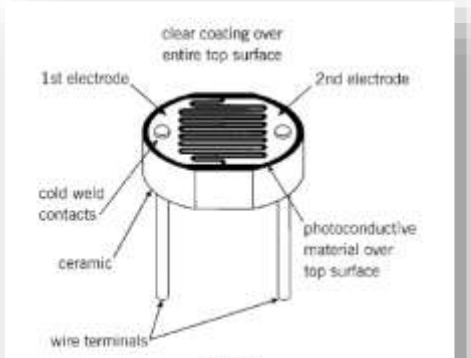
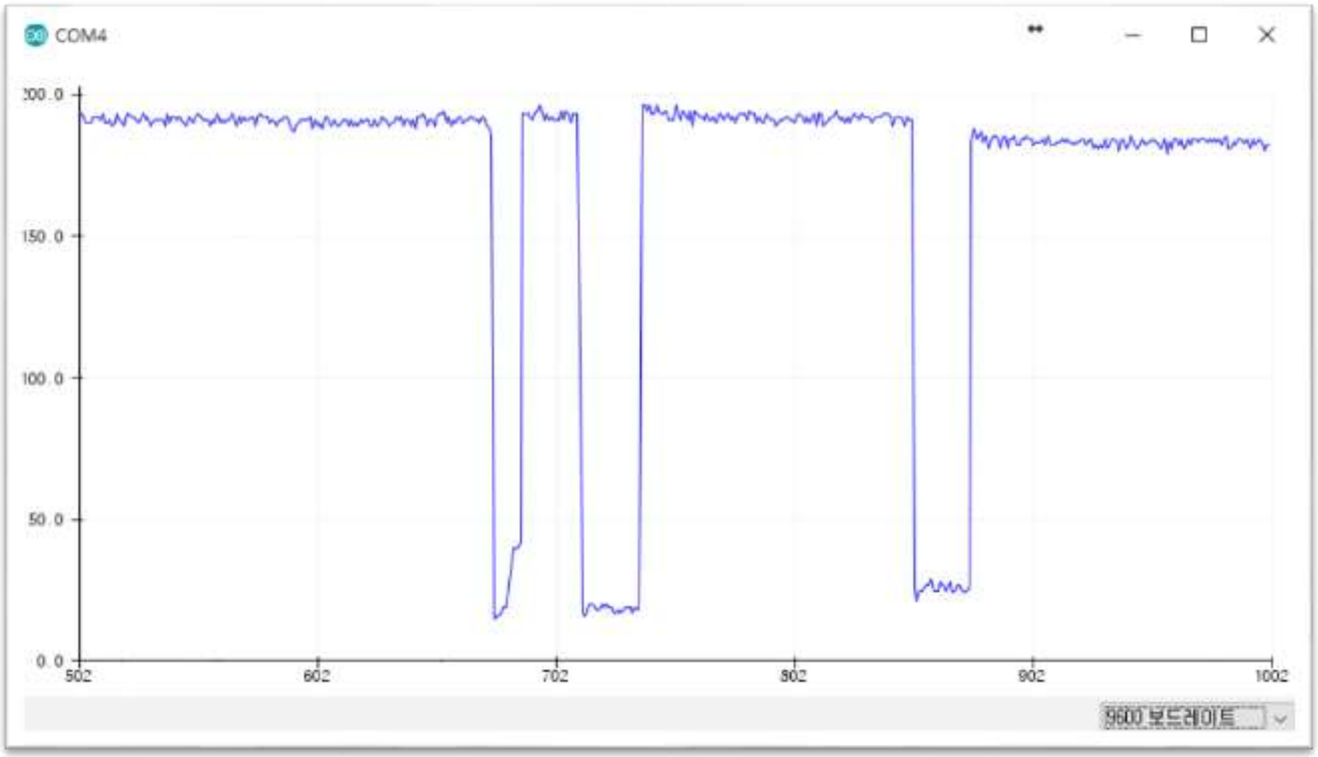
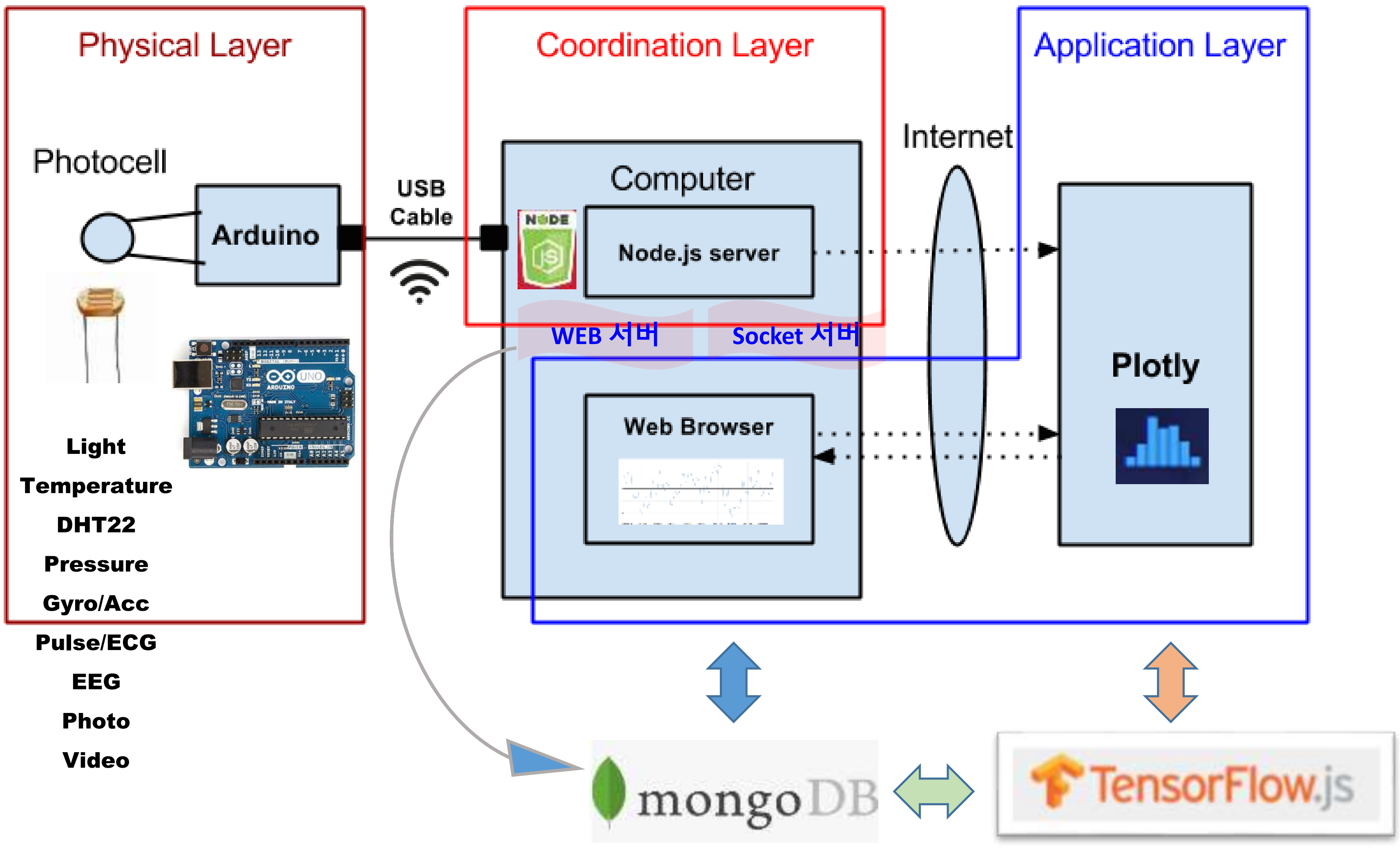

clear coating over entire top surface
1st electrode
2nd electrode
cold weld contacts
ceramic
photoconductive material over top surface
wire terminals

Figure 3
Typical Construction of a Plastic Coated Photocell

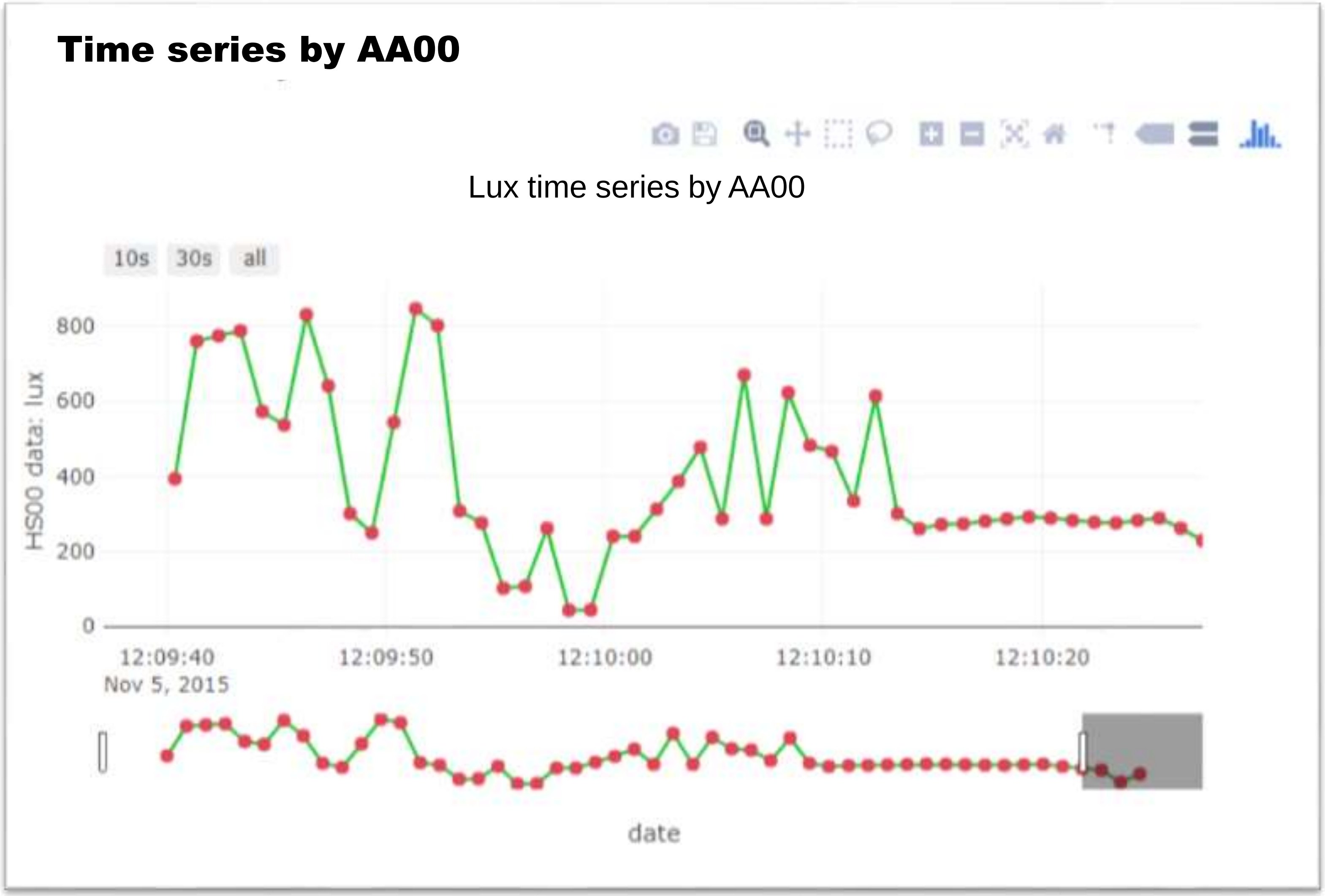


9600 보드레이트

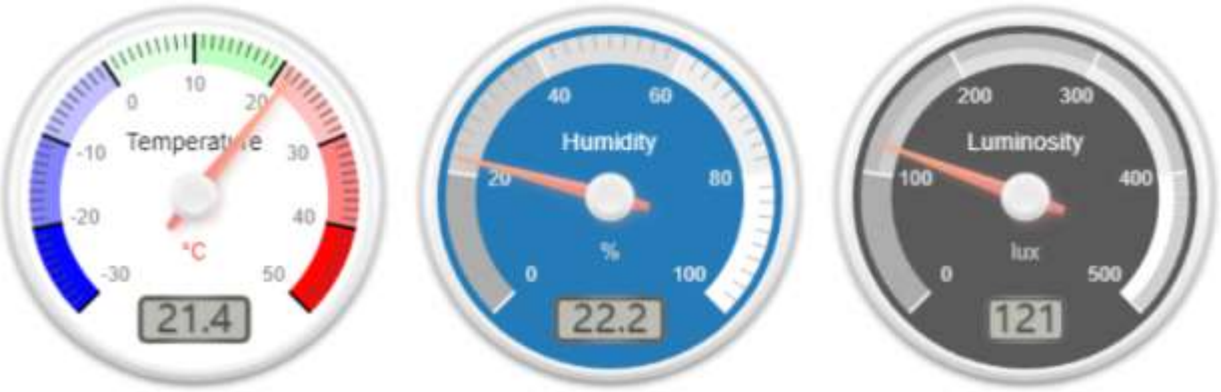
Layout [H S C]


Physical Layer
Coordination Layer
Application Layer
Photocell
Arduino
USB Cable
Computer
Node.js server
Internet
Plotly
WEB 서버
Socket 서버
Web Browser
Light
Temperature
DHT22
Pressure
Gyro/Acc
Pulse/ECG
EEG
Photo
Video
mongoDB
TensorFlow.js

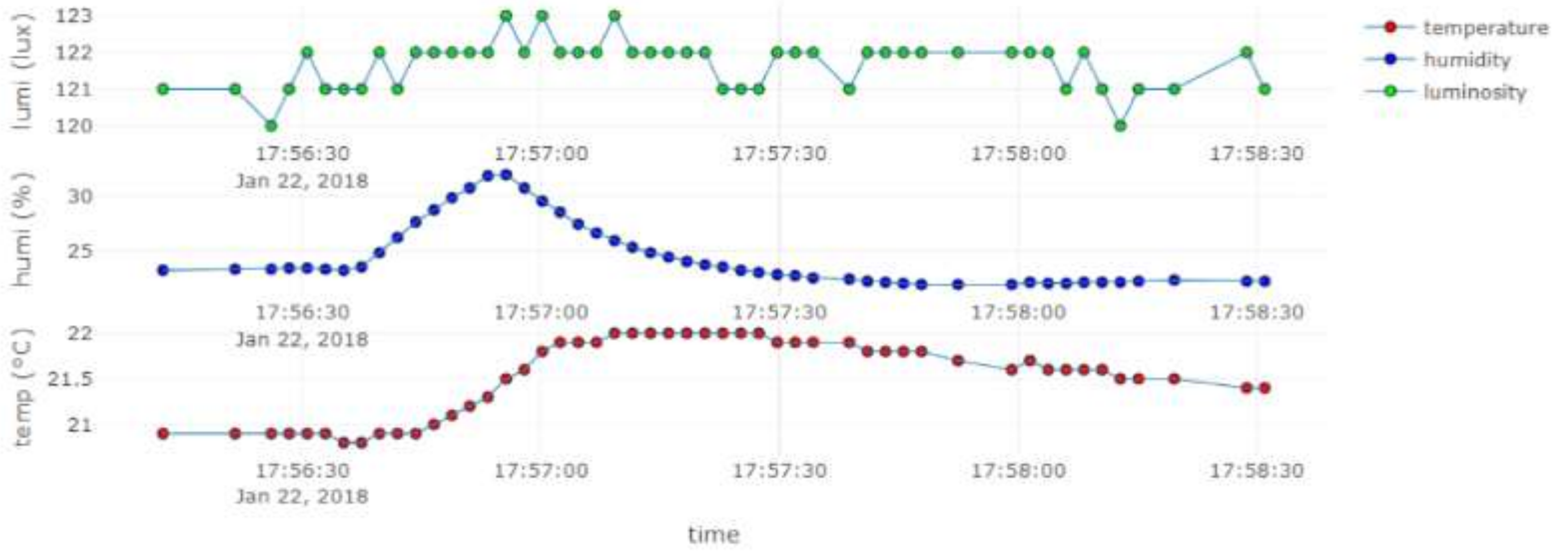
Arduino data + plotly



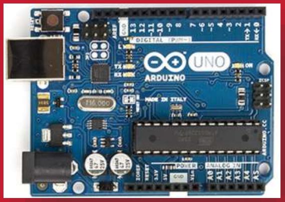
Real-time Weather Station from sensors

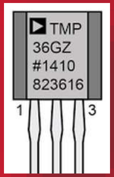


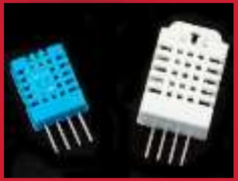
on Time: 2018-01-22 17:58:31.012

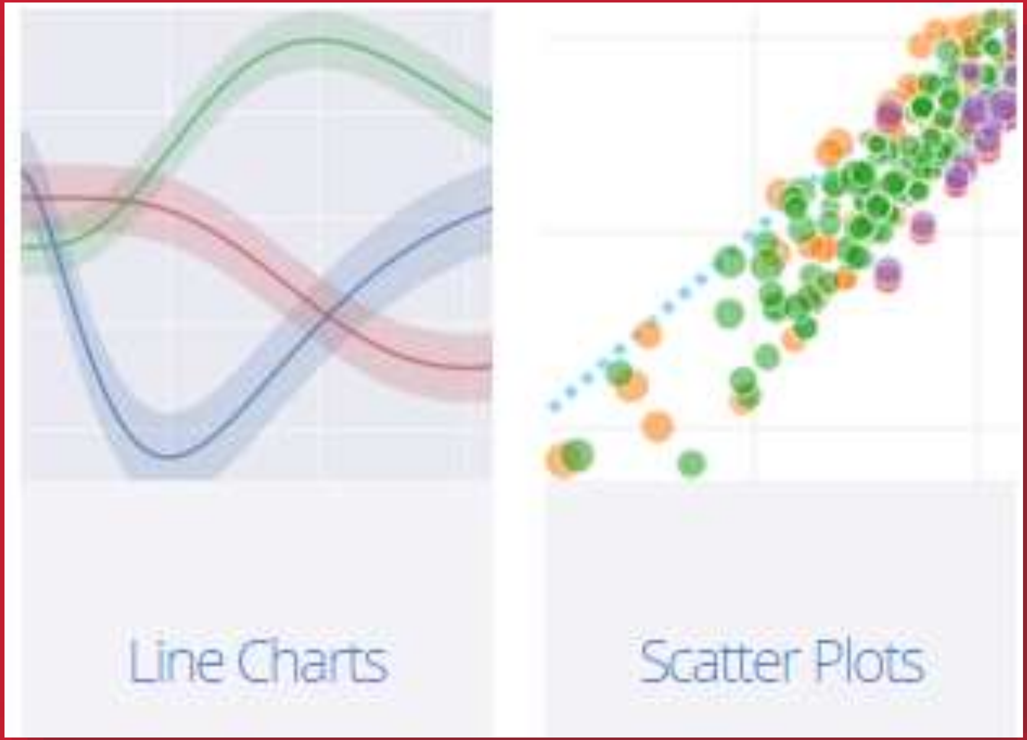


Data charts using plotly.js



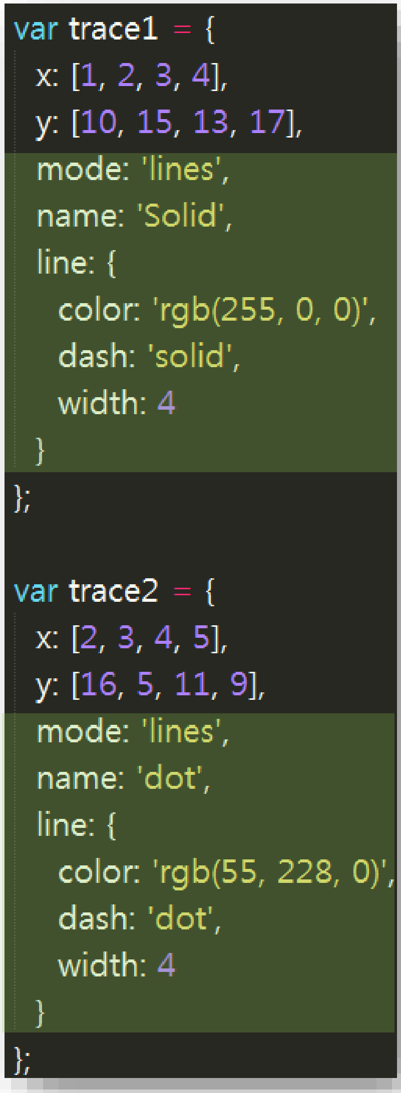




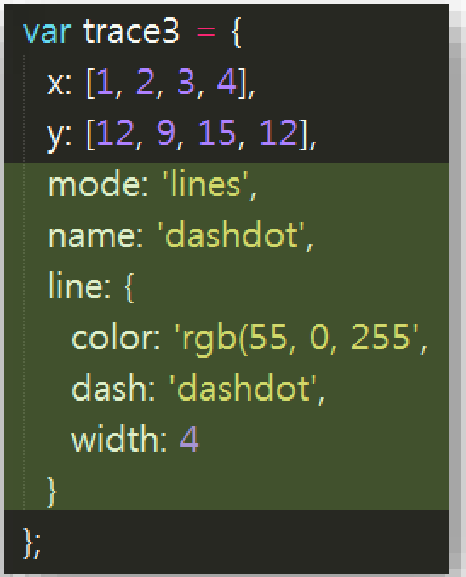


A5.2.6.6 plotly.js: Line & Scatter plot

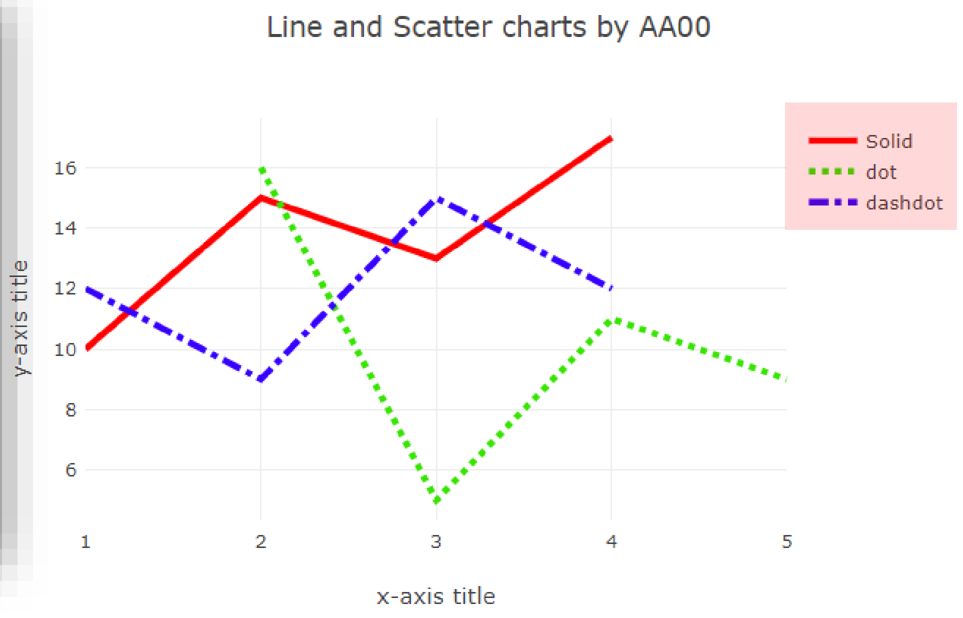
[3.5] Line & scatter plot with dash and dot



```
var trace1 = {
  x: [1, 2, 3, 4],
  y: [10, 15, 13, 17],
  mode: 'lines',
  name: 'Solid',
  line: {
    color: 'rgb(255, 0, 0)',
    dash: 'solid',
    width: 4
  }
};

var trace2 = {
  x: [2, 3, 4, 5],
  y: [16, 5, 11, 9],
  mode: 'lines',
  name: 'dot',
  line: {
    color: 'rgb(55, 228, 0)',
    dash: 'dot',
    width: 4
  }
};
```

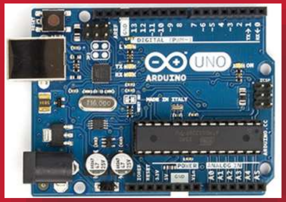


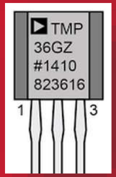
```
var trace3 = {
  x: [1, 2, 3, 4],
  y: [12, 9, 15, 12],
  mode: 'lines',
  name: 'dashdot',
  line: {
    color: 'rgb(55, 0, 255',
    dash: 'dashdot',
    width: 4
  }
};
```

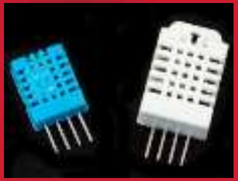


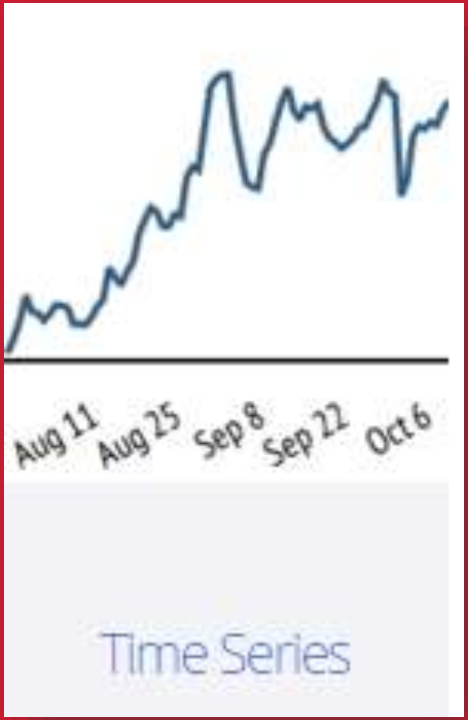
AAnn_Line_Dash_Dot.png

Data visualization using plotly.js



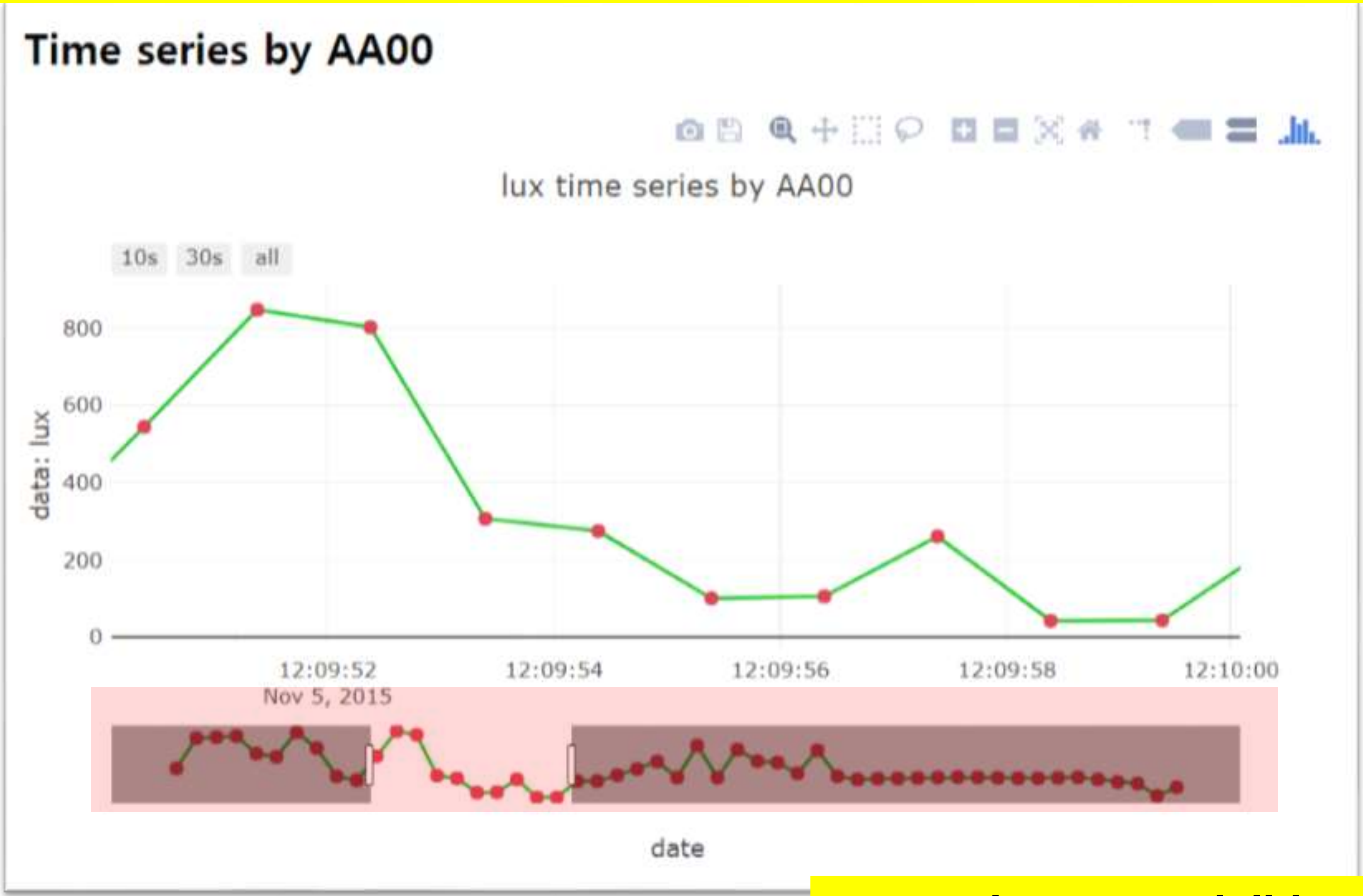






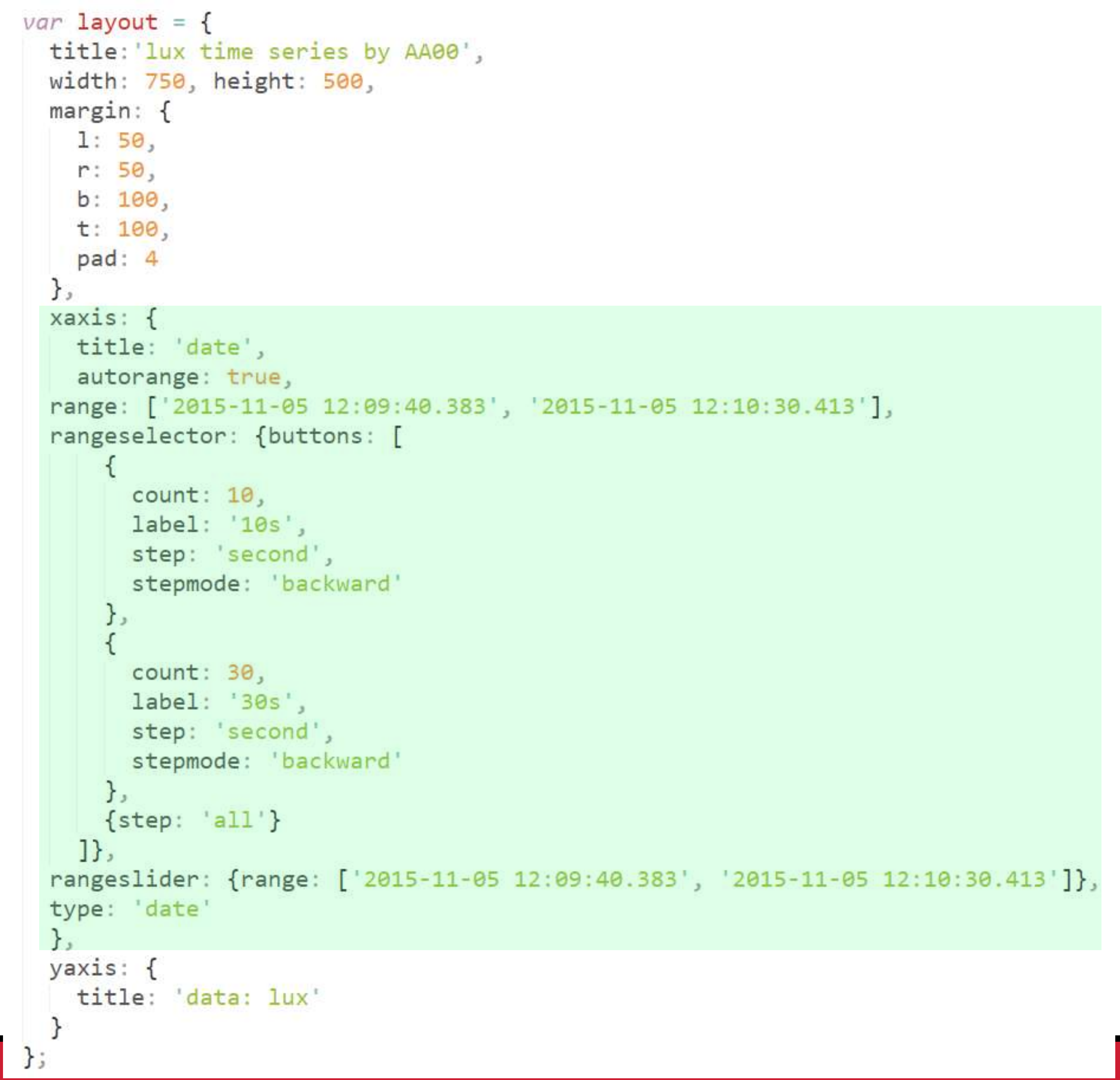
Project: Time series with Rangeslider

[Project-DIY] AAnn_lux_Rangelslider.html



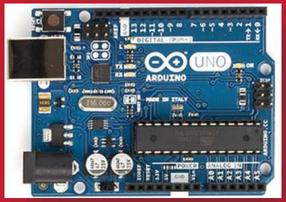
AAnn_lux_Rangelslider.png

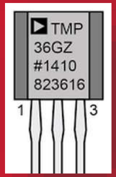
Time series with Rangeslider

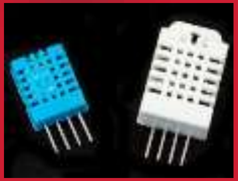


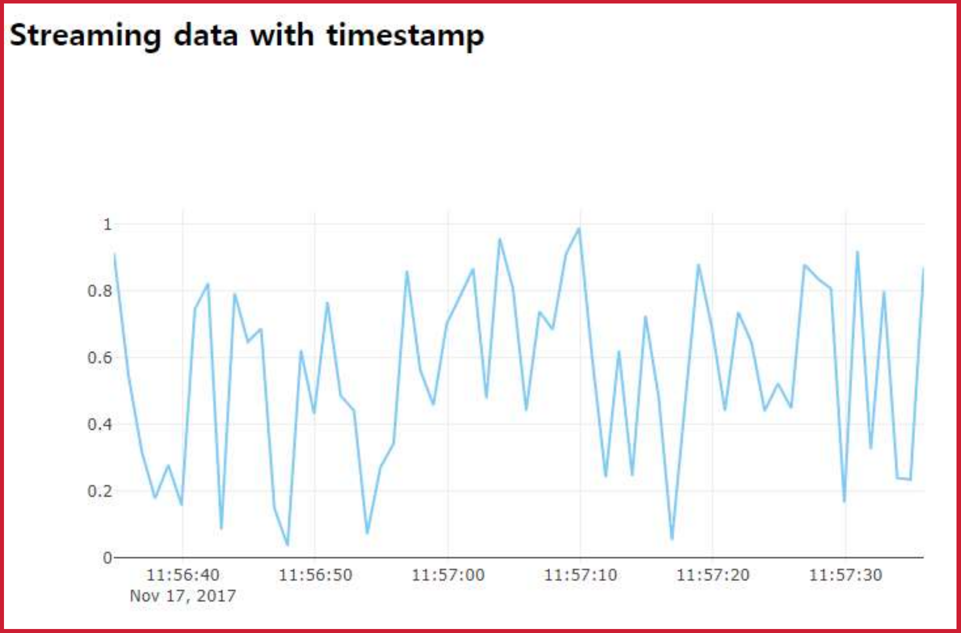
```
var layout = {
  title:'lux time series by AA00',
  width: 750, height: 500,
  margin: {
    l: 50,
    r: 50,
    b: 100,
    t: 100,
    pad: 4
  },
  xaxis: {
    title: 'date',
    autorange: true,
  range: ['2015-11-05 12:09:40.383', '2015-11-05 12:10:30.413'],
  rangeselector: {buttons: [
      {
        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



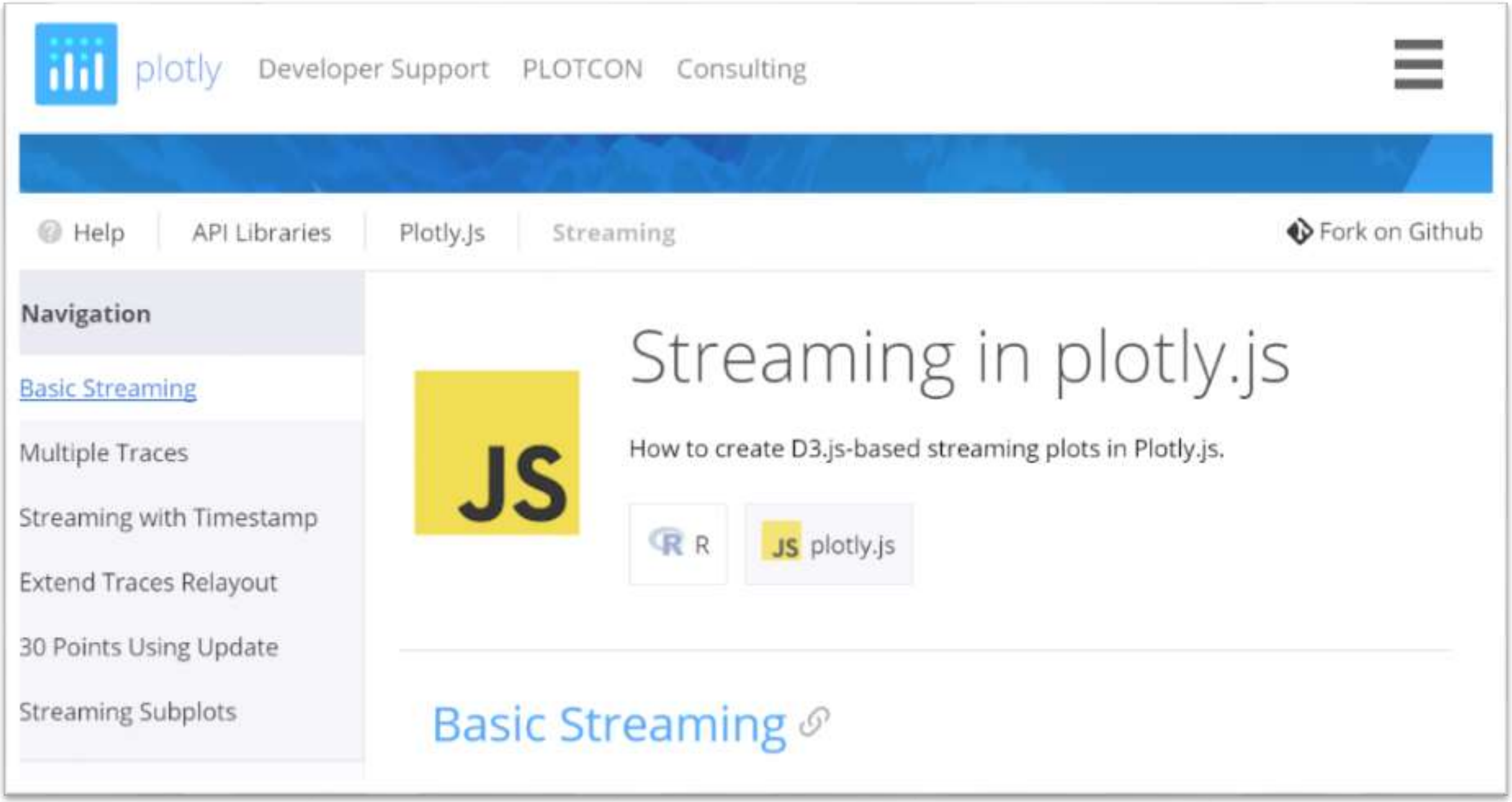






A5.4 plotly.js: Streaming data

Plot.ly > Streaming


plotly Developer Support PLOTCON Consulting

Help | API Libraries | Plotly.Js | Streaming | Fork on Github

Navigation

- Basic Streaming
- Multiple Traces
- Streaming with Timestamp
- Extend Traces Relayout
- 30 Points Using Update
- Streaming Subplots

Streaming in plotly.js

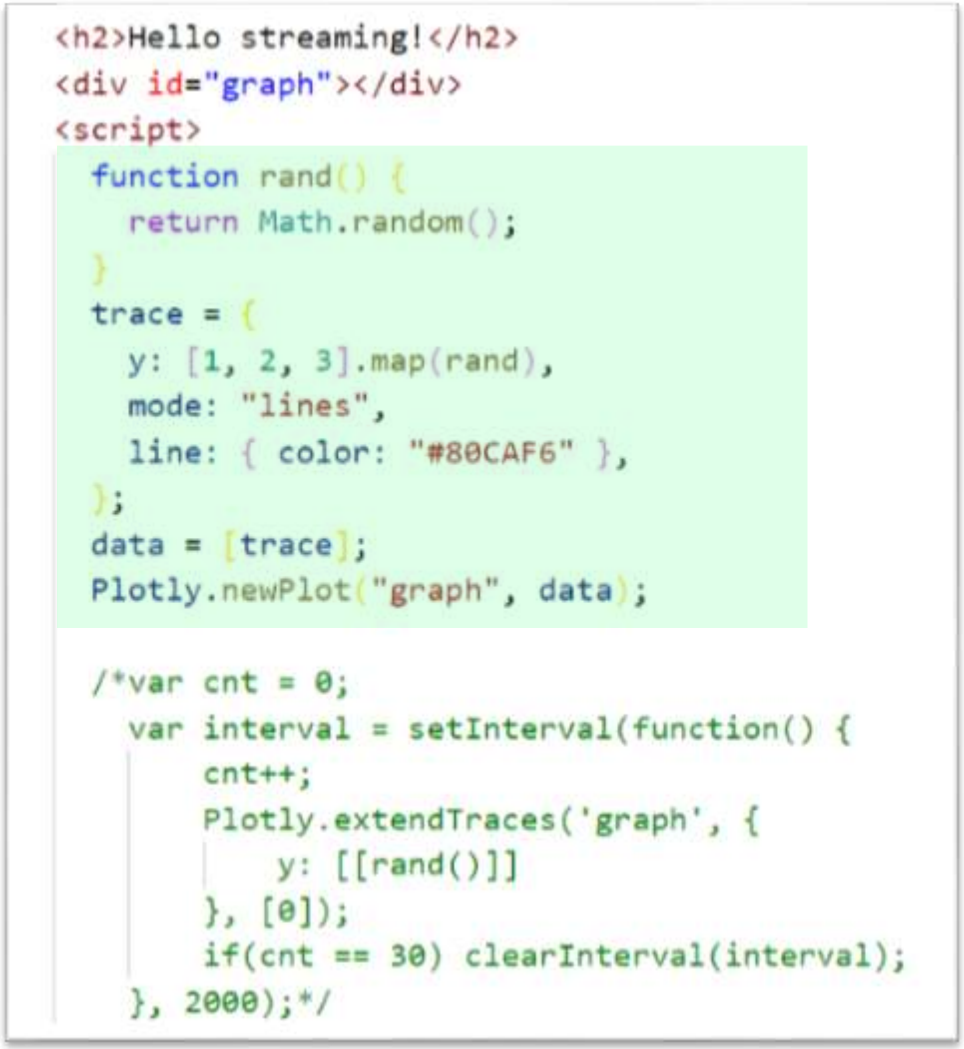
How to create D3.js-based streaming plots in Plotly.js.

R | plotly.js

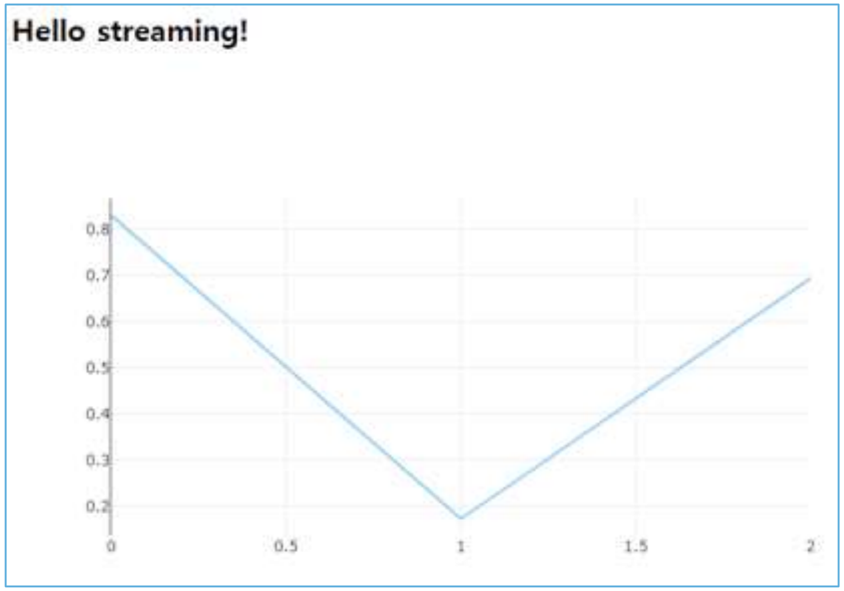
Basic Streaming

A5.4.1 plotly.js: Streaming data

[1.0] Starting chart



```
<h2>Hello streaming!</h2>
<div id="graph"></div>
<script>
  function rand() {
    return Math.random();
  }
  trace = {
    y: [1, 2, 3].map(rand),
    mode: "lines",
    line: { color: "#80CAF6" },
  };
  data = [trace];
  Plotly.newPlot("graph", data);

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



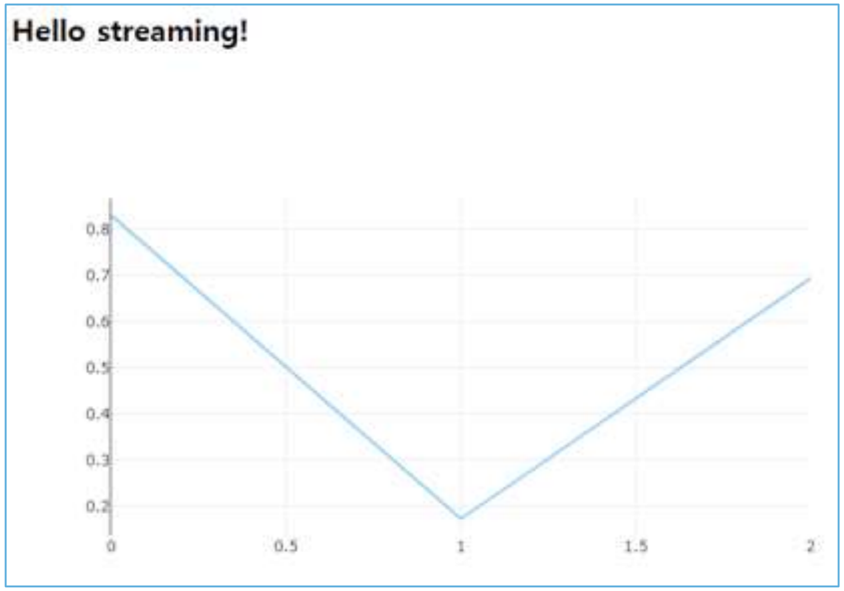
Hello streaming!

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) DV_streaming01.html

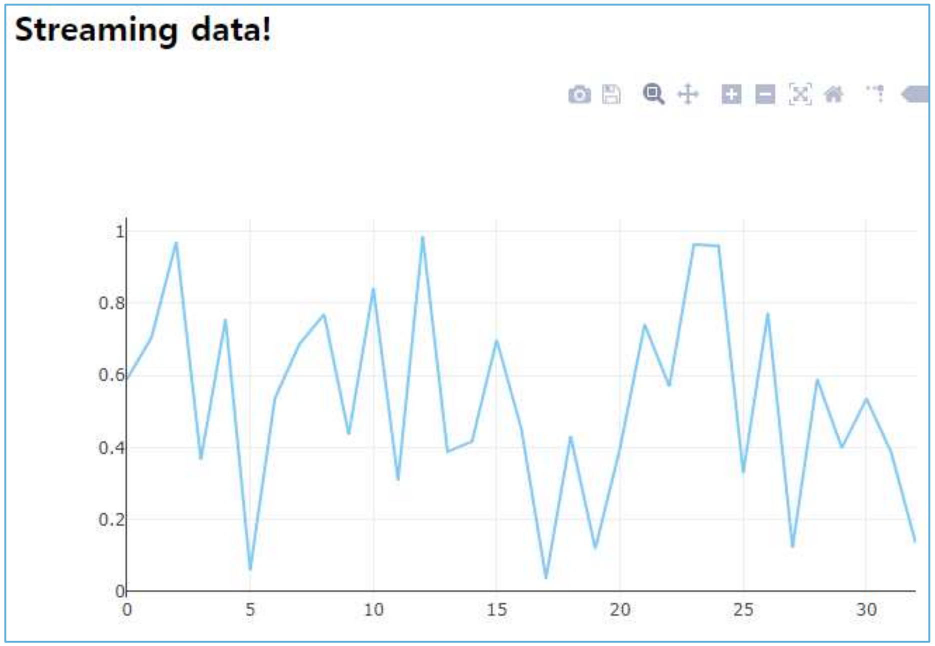
```
<h2>Hello streaming!</h2>
<div id="graph"></div>
<script>
  function rand() {
    return Math.random();
  }

  Plotly.newPlot("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>
```


Hello streaming!

A5.4.2.2 plotly.js: Streaming data

[1.2] Basic streaming DV_streaming01S.html

```
<h2>Streaming data!</h2>
<div id="graph"></div>
<script>
  function rand() {
    return Math.random();
  }
  Plotly.newPlot("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>
```

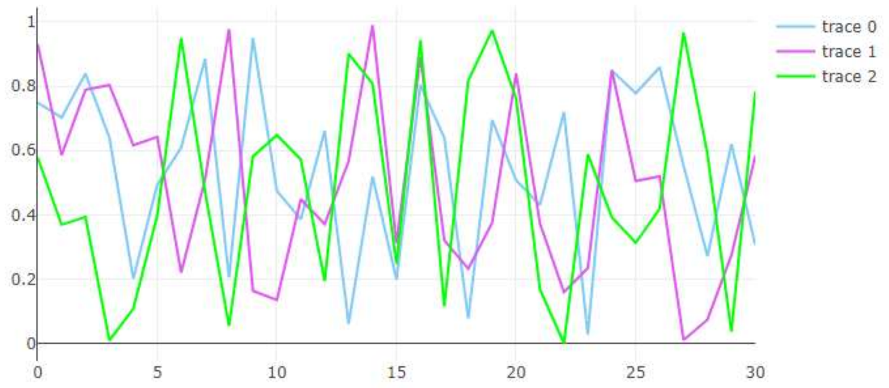

Streaming data!

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);
```

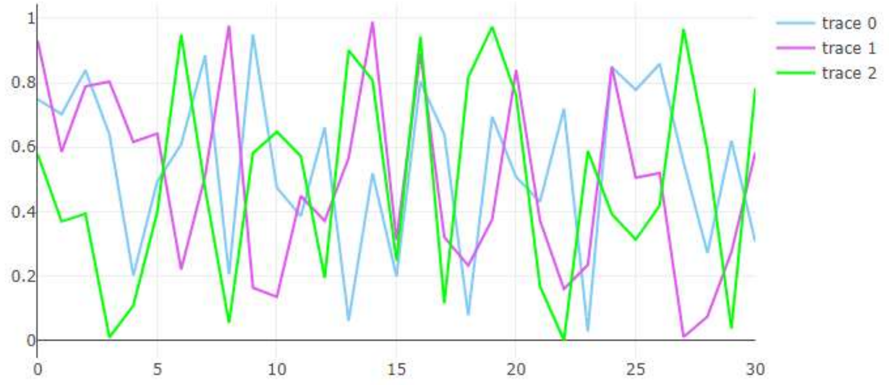


A5.4.3.2 plotly.js: Streaming data

[2.2] Streaming multiple traces (new code) **DV_streaming02.html**

```
function rand() {
  return Math.random();
}
Plotly.newPlot("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" },
  },
]);
```

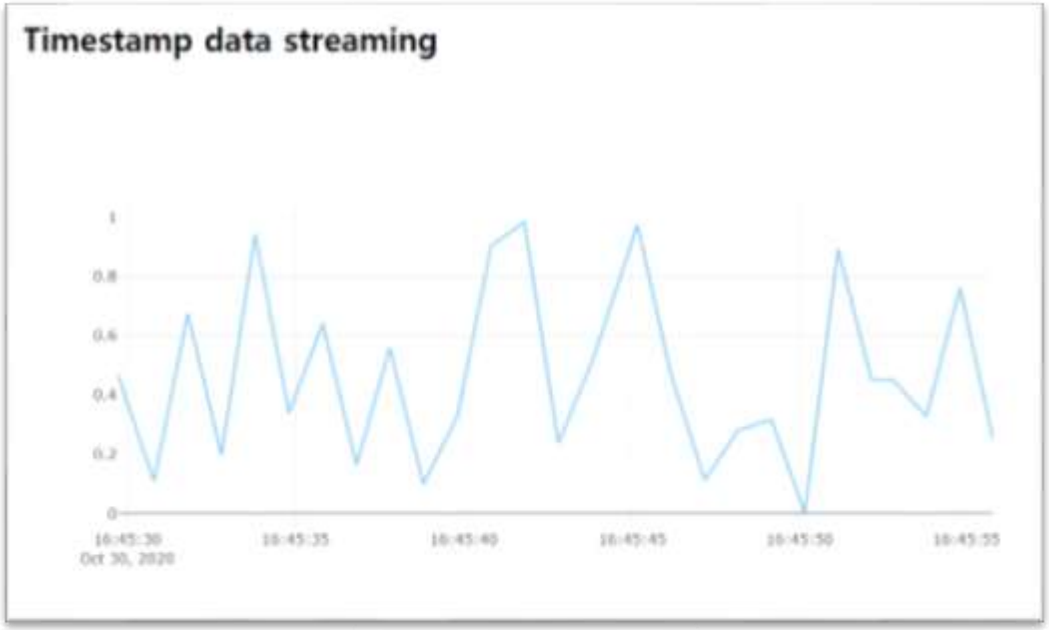
```
// continous plot
var cnt = 0;
var interval = setInterval(function() {

    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 DV_streaming03_timestamp.html

```
function rand() {
  return Math.random();
}
var time = new Date();
var data = [
  {
    x: [time],
    y: [rand()],
    mode: "lines",
    line: { color: "#80CAF6" },
  },
];
Plotly.newPlot("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);
```

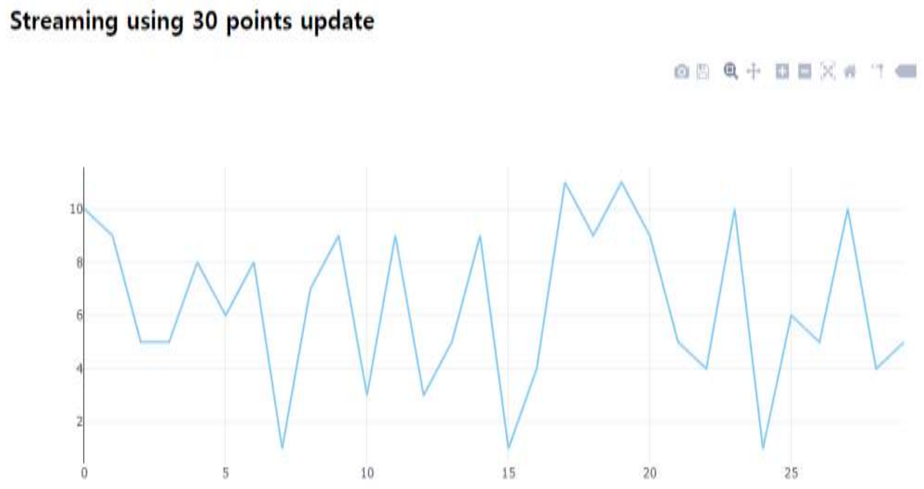


A5.4.5 plotly.js: Streaming data

[4] Streaming data using 30 points update

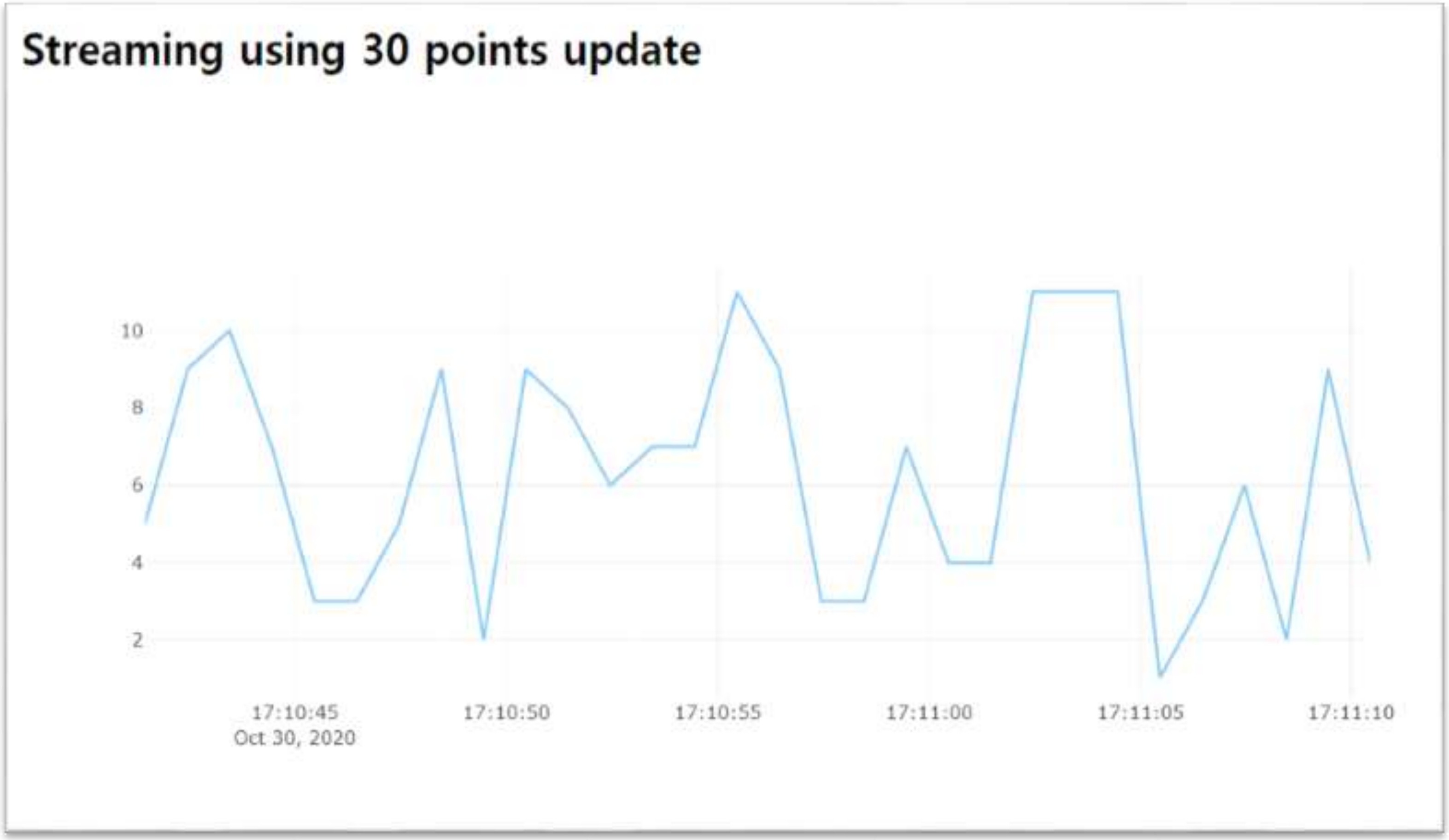
```
var arrayLength = 30;
var newArray = [];
// initial 30 data
for (var i = 0; i < arrayLength; i++) {
  var y = Math.round(Math.random() * 10) + 1;
  newArray[i] = y;
}
var data = [
  {
    y: newArray,
    mode: "lines",
    line: { color: "#80CAF6" },
  },
];
Plotly.newPlot("graph", data);
```

```
var cnt = 0;
var interval = setInterval(function () {
  var y = Math.round(Math.random() * 10) + 1;
  newArray = newArray.concat(y); // add new data
  newArray.splice(0, 1); //remove the oldest data
  var update = {
    y: [newArray],
  };
  Plotly.update("graph", update);
  cnt++;
  if (cnt === 50) clearInterval(interval);
}, 1000);
```



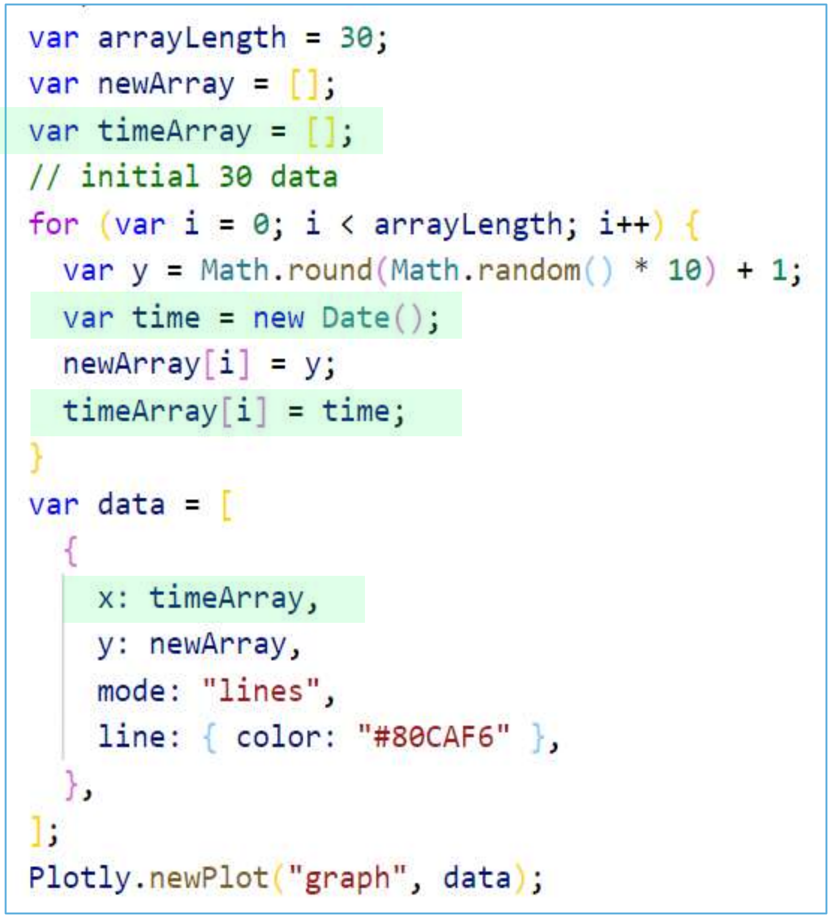
A5.4.5.1 plotly.js: Streaming data

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

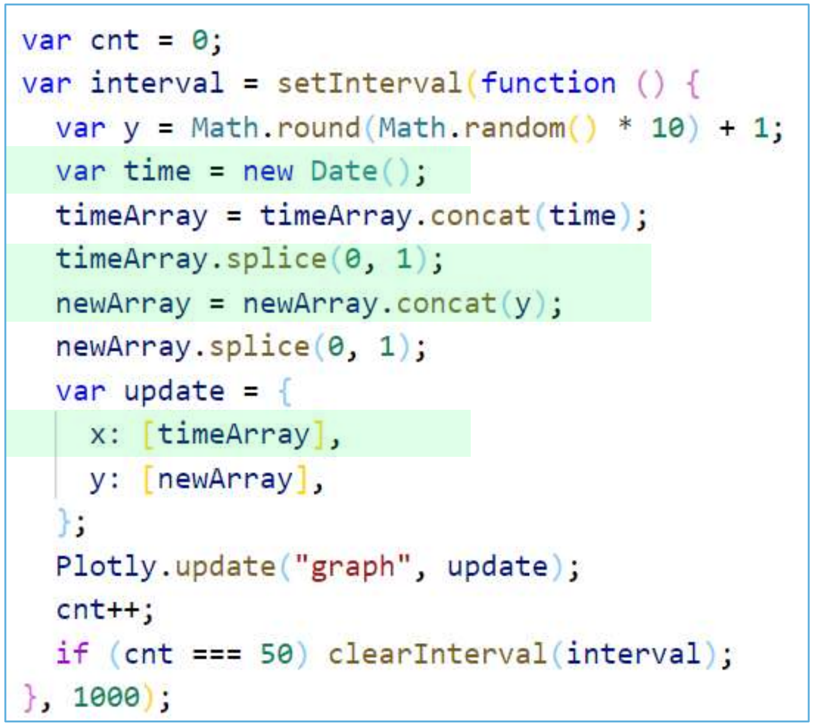


A5.4.5.2 plotly.js: Streaming data

[4.2] Streaming data using 30 points update   DV_streaming04_range.html



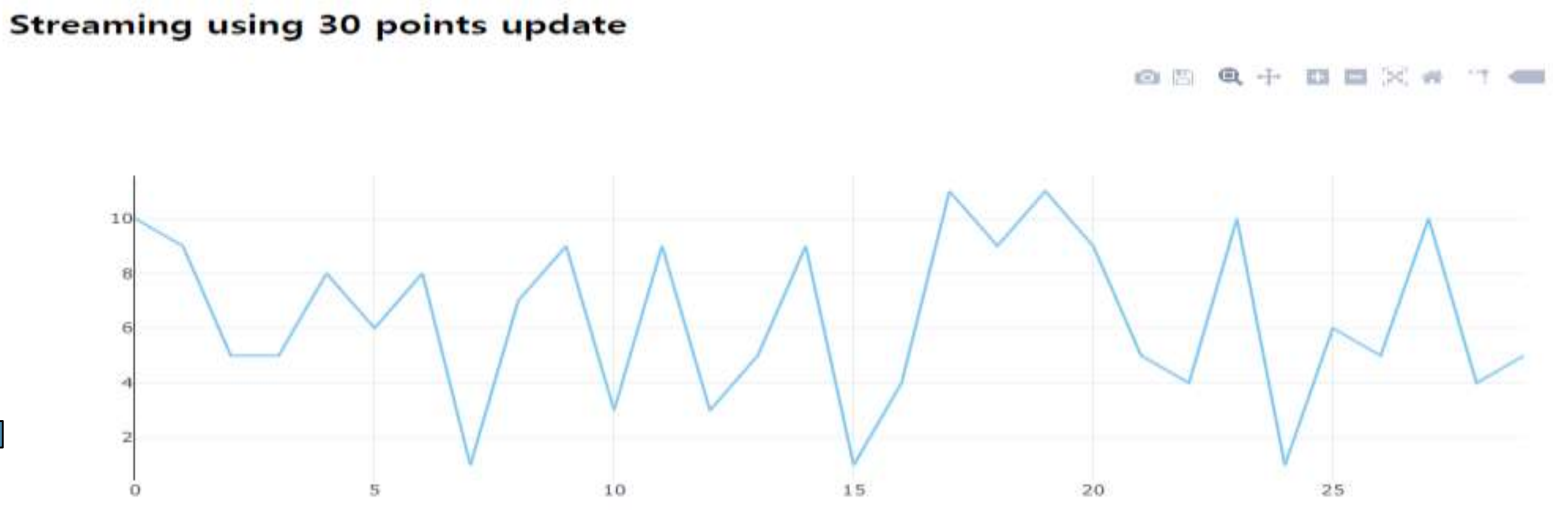
```
var arrayLength = 30;
var newArray = [];
var timeArray = [];
// initial 30 data
for (var i = 0; i < arrayLength; i++) {
  var y = Math.round(Math.random() * 10) + 1;
  var time = new Date();
  newArray[i] = y;
  timeArray[i] = time;
}
var data = [
  {
    x: timeArray,
    y: newArray,
    mode: "lines",
    line: { color: "#80CAF6" },
  },
];
Plotly.newPlot("graph", data);
```

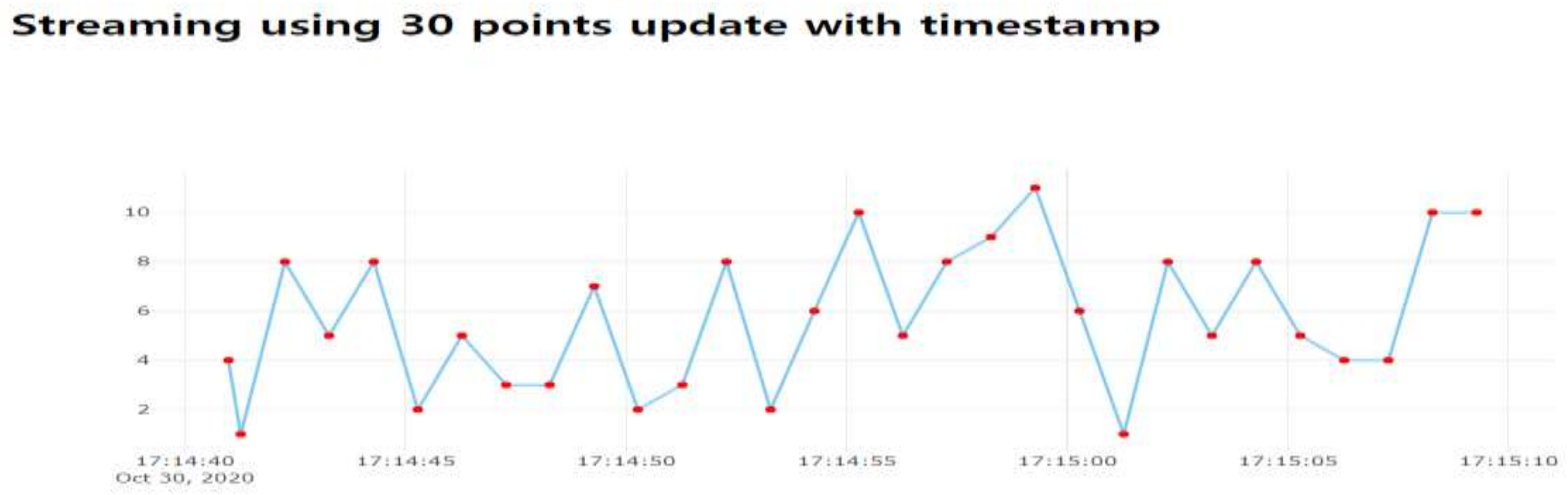


```
var cnt = 0;
var interval = setInterval(function () {
  var y = Math.round(Math.random() * 10) + 1;
  var time = new Date();
  timeArray = timeArray.concat(time);
  timeArray.splice(0, 1);
  newArray = newArray.concat(y);
  newArray.splice(0, 1);
  var update = {
    x: [timeArray],
    y: [newArray],
  };
  Plotly.update("graph", update);
  cnt++;
  if (cnt === 50) clearInterval(interval);
}, 1000);
```

A5.4.5.3 plotly.js: Streaming data

[DIY] Streaming time series using 30 points update

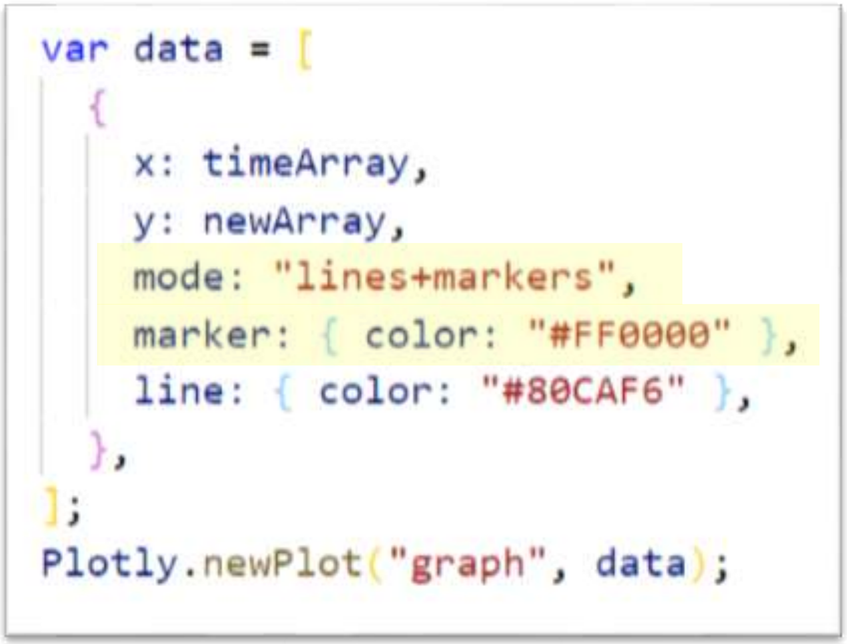




AAnn_DS_30timestamps.png 로 캡처 저장.

A5.4.5.4 plotly.js: Streaming data

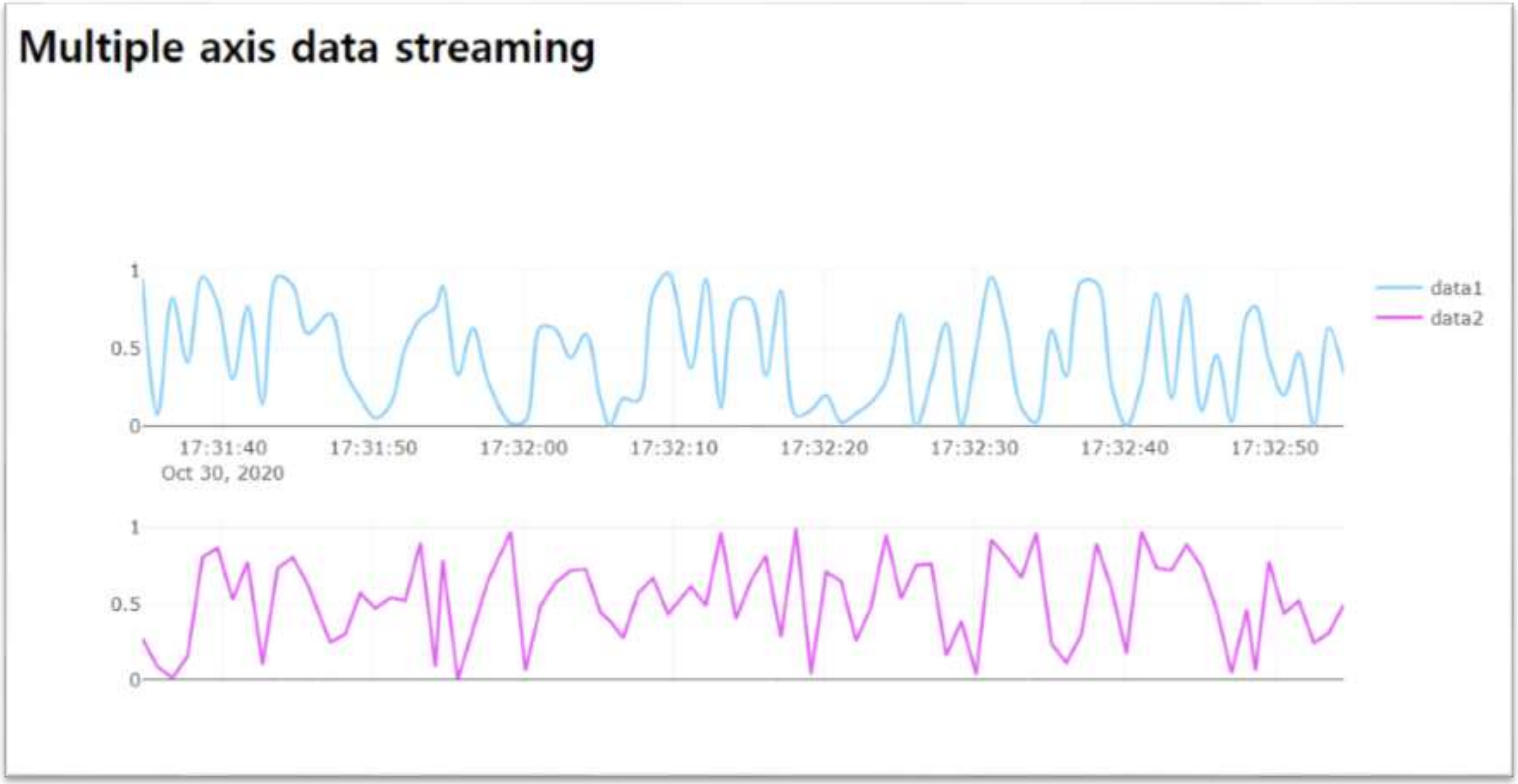
[DIY-hint] Streaming time series using 30 points update



```
var data = [
  {
    x: timeArray,
    y: newArray,
    mode: "lines+markers",
    marker: { color: "#FF0000" },
    line: { color: "#80CAF6" },
  },
];
Plotly.newPlot("graph", data);
```

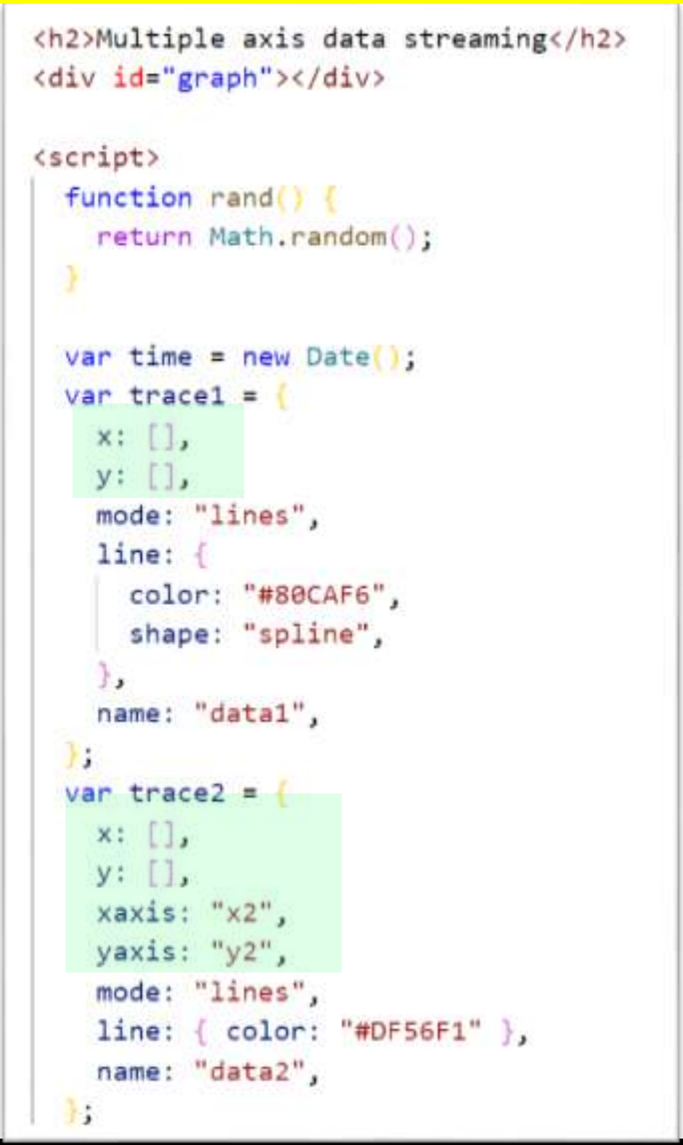
A5.4.6 plotly.js: Streaming data

[5] Streaming data using multiple axis

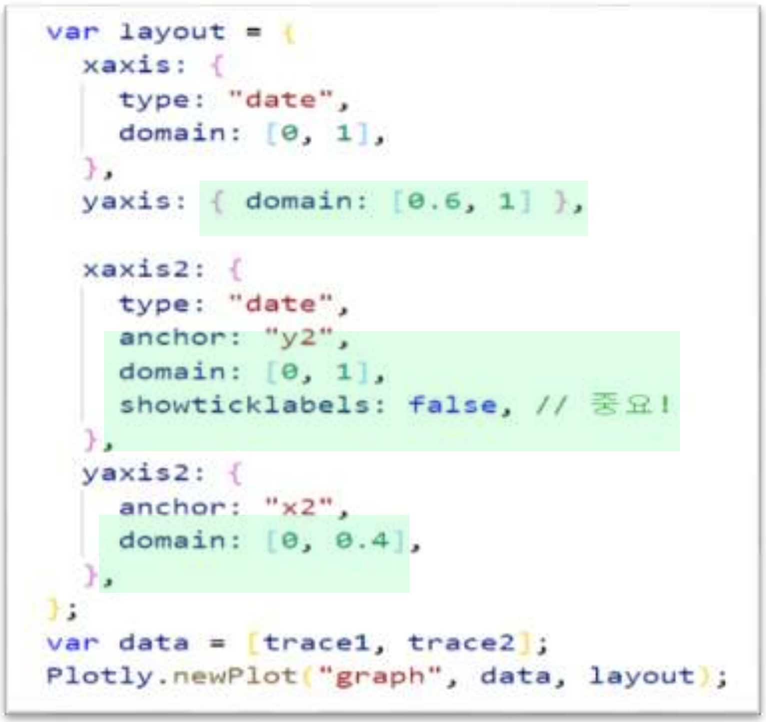


A5.4.6.1 plotly.js: Streaming data

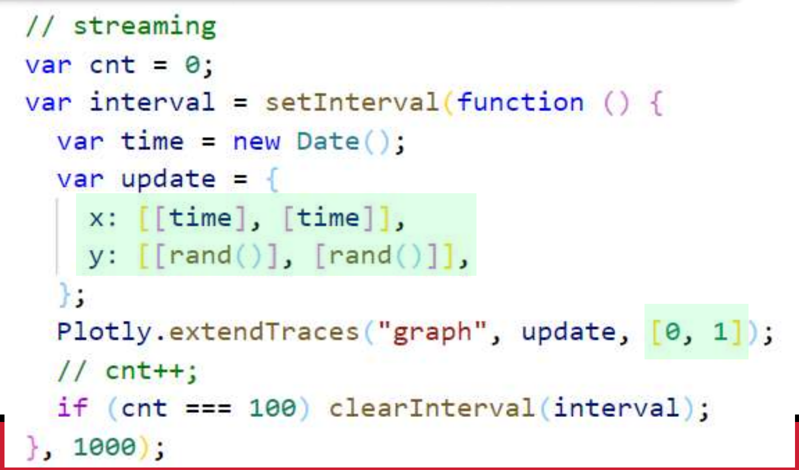
[5.1] Streaming data using multiple axis **DV_streaming05_multiple_axis.html**



```
<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",
  };
```



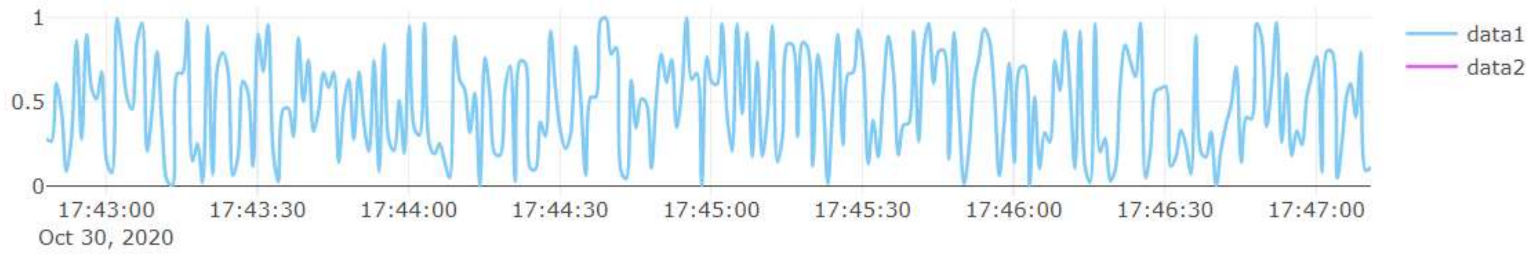
```
var layout = {
  xaxis: {
    type: "date",
    domain: [0, 1],
  },
  yaxis: { domain: [0.6, 1] },

  xaxis2: {
    type: "date",
    anchor: "y2",
    domain: [0, 1],
    showticklabels: false, // 중요!
  },
  yaxis2: {
    anchor: "x2",
    domain: [0, 0.4],
  },
};
var data = [trace1, trace2];
Plotly.newPlot("graph", data, layout);
```

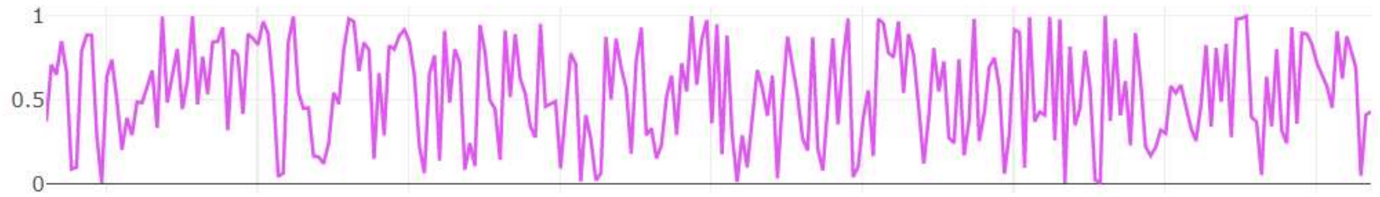


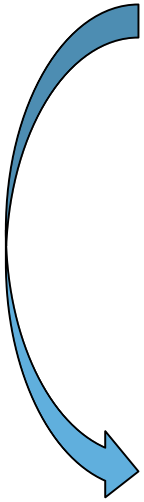
```
// streaming
var cnt = 0;
var interval = setInterval(function () {
  var time = new Date();
  var update = {
    x: [[time], [time]],
    y: [[rand()], [rand()]],
  };
  Plotly.extendTraces("graph", update, [0, 1]);
  // cnt++;
  if (cnt === 100) clearInterval(interval);
}, 1000);
```

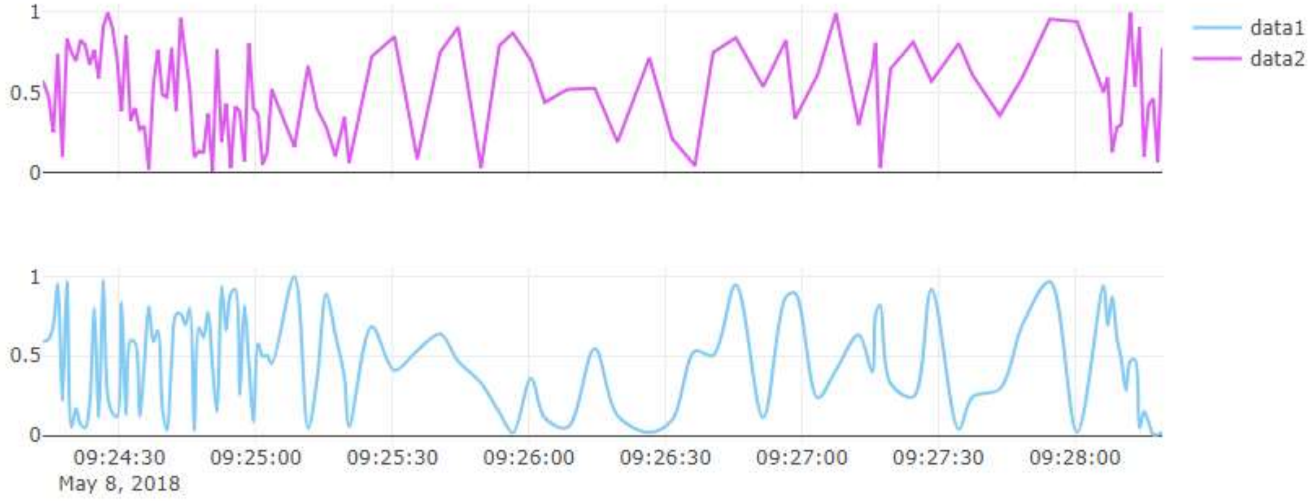
A5.4.6.2 plotly.js: Streaming data

[DIY] Streaming data using multiple axis → change axis





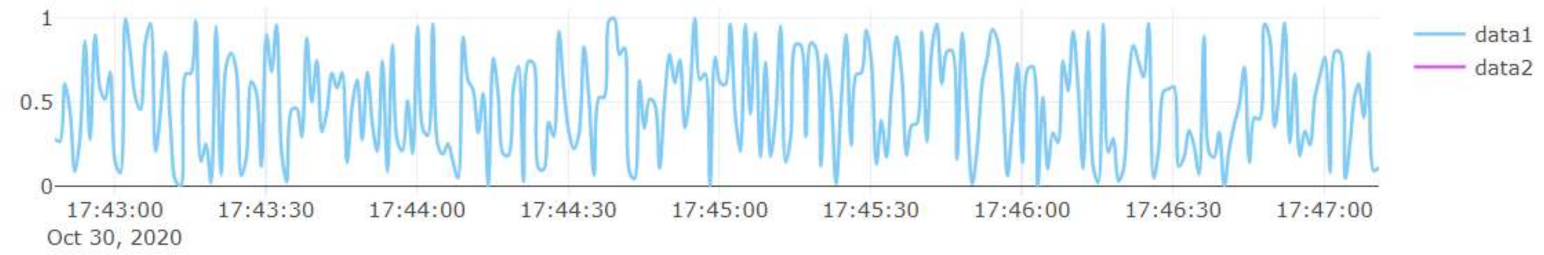


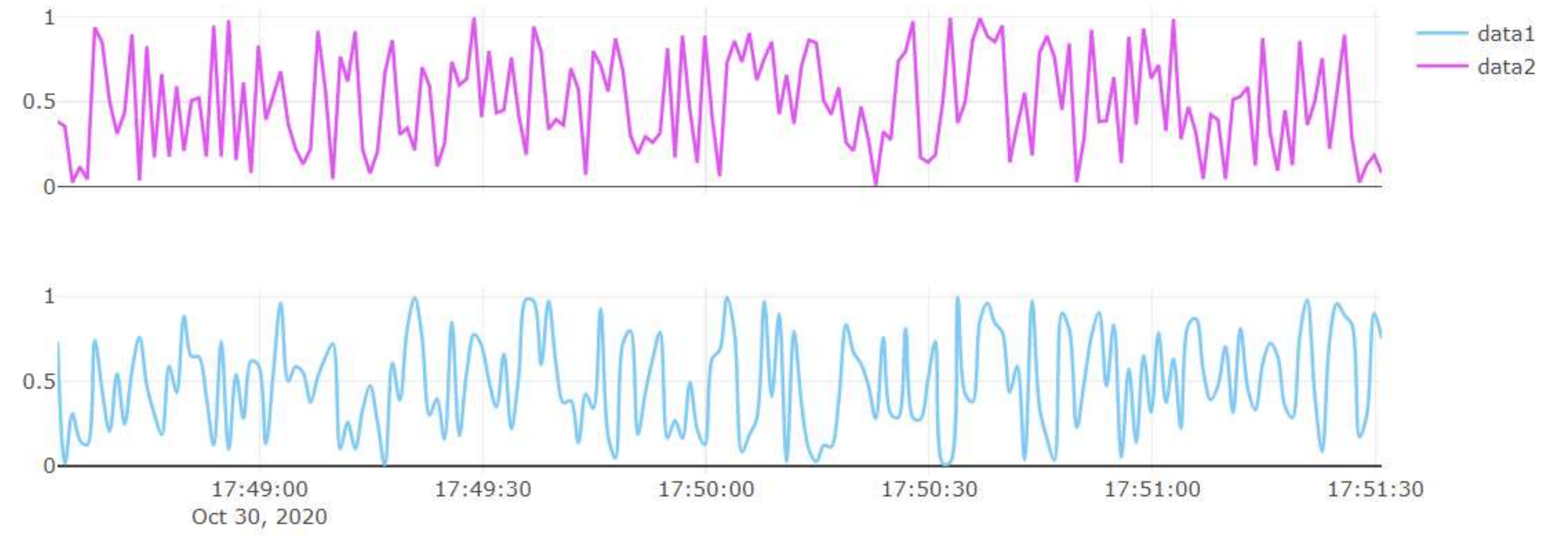


AAnn_DS_multiple_axis.png 로 캡처 저장.

A5.4.6.2 plotly.js: Streaming data

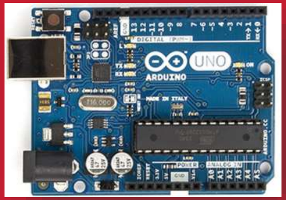
[DIY] Streaming data using multiple axis → change axis

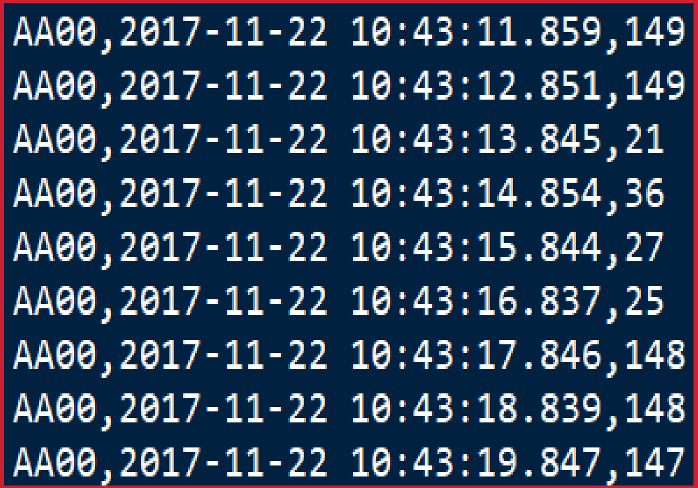




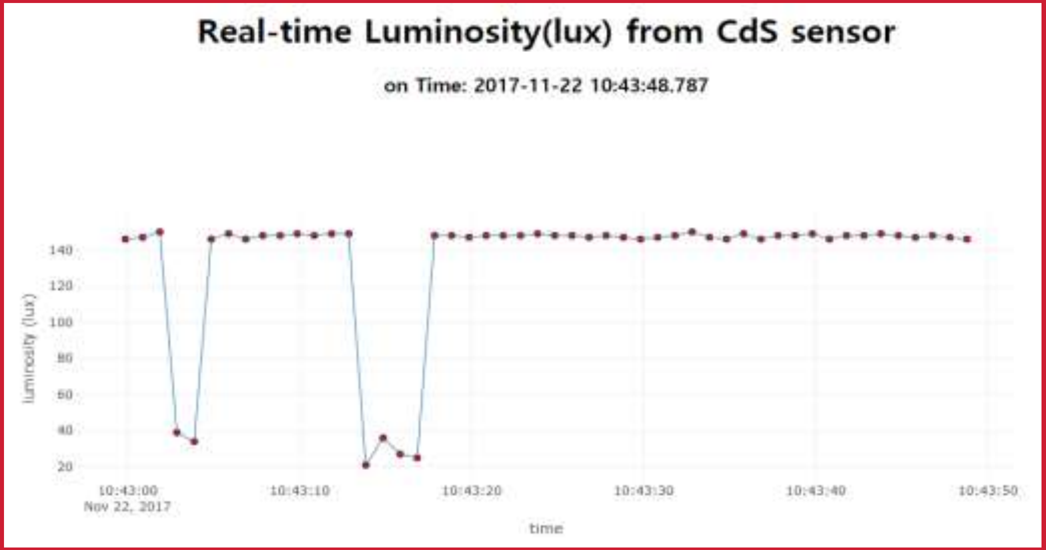
AAnn_DS_multiple_axis.png 로 캡처 저장.

Arduino sensor data RT visualization using plotly.js



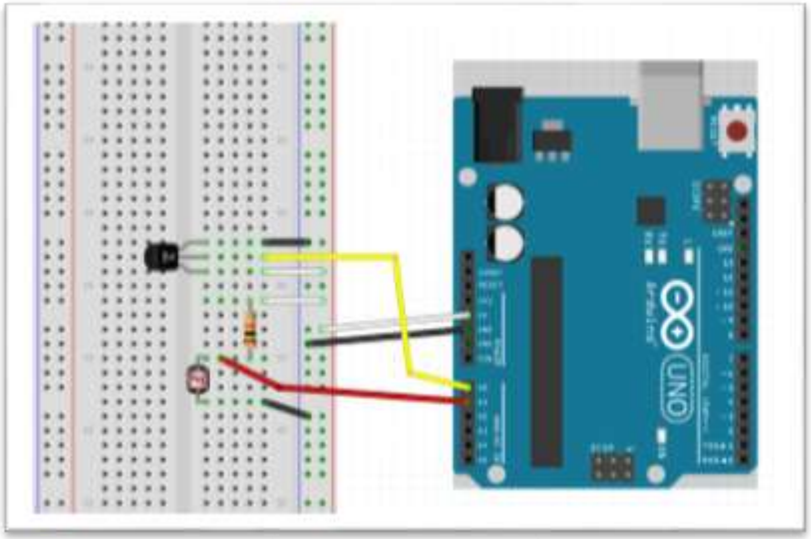


```
AA00,2017-11-22 10:43:11.859,149
AA00,2017-11-22 10:43:12.851,149
AA00,2017-11-22 10:43:13.845,21
AA00,2017-11-22 10:43:14.854,36
AA00,2017-11-22 10:43:15.844,27
AA00,2017-11-22 10:43:16.837,25
AA00,2017-11-22 10:43:17.846,148
AA00,2017-11-22 10:43:18.839,148
AA00,2017-11-22 10:43:19.847,147
```



Network socket emitting data

tmp36 + CdS circuit



```
AA00,2020-10-17 11:41:30.533,25.27,245
AA00,2020-10-17 11:41:31.535,25.27,243
AA00,2020-10-17 11:41:32.535,25.27,158
AA00,2020-10-17 11:41:33.534,24.29,40
AA00,2020-10-17 11:41:34.538,24.29,33
AA00,2020-10-17 11:41:35.537,24.78,86
AA00,2020-10-17 11:41:36.541,25.27,249
AA00,2020-10-17 11:41:37.540,25.76,245
AA00,2020-10-17 11:41:38.543,25.76,243
AA00,2020-10-17 11:41:39.543,25.27,245
```

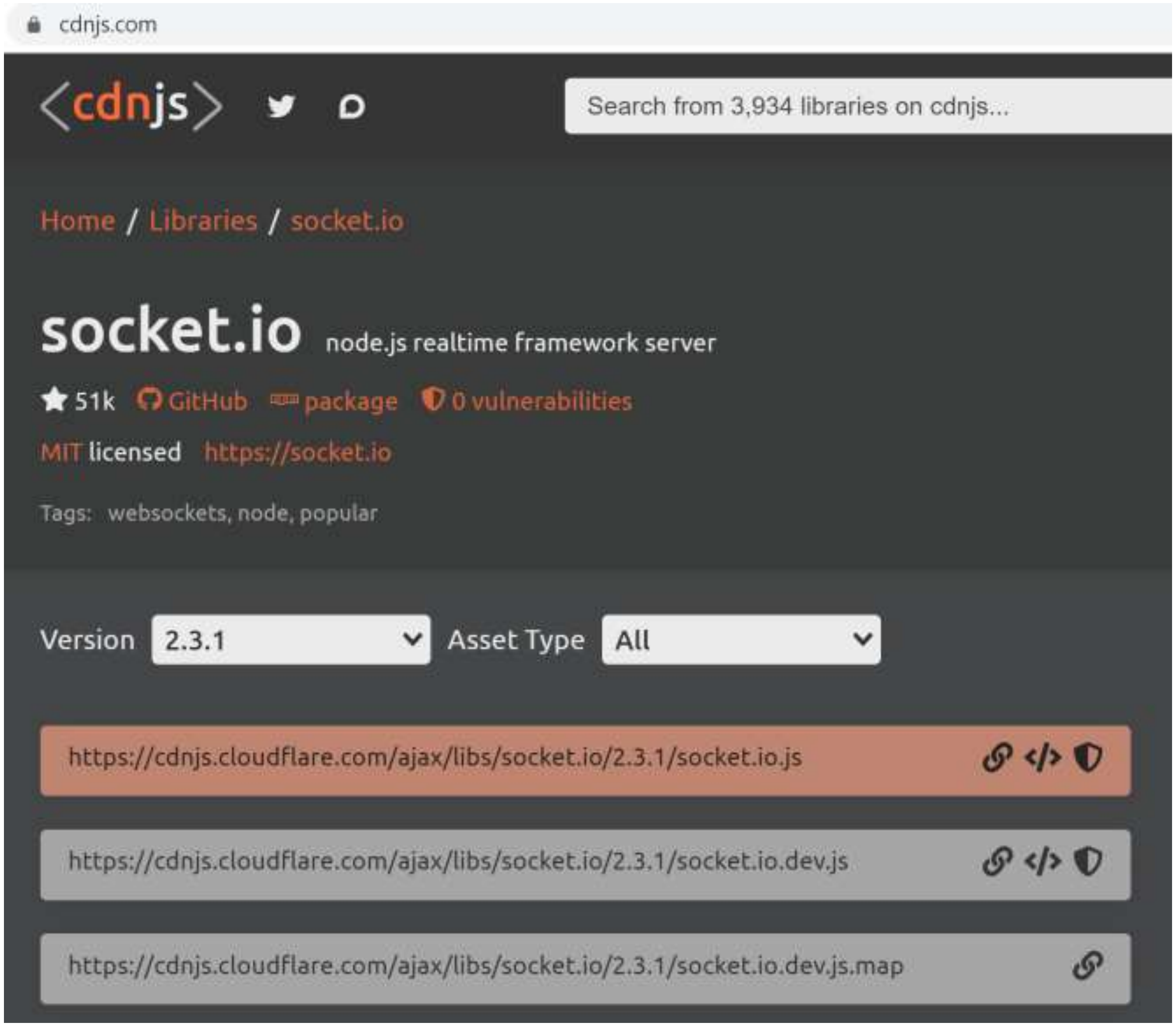
```
var readData = "";
var temp = "";
var lux = "";
var mdata = [];
var firstcommaidx = 0;

parser.on("data", (data) => {
  // call back when data is received
  readData = data.toString();
  firstcommaidx = readData.indexOf(",");
  if (firstcommaidx > 0) {
    temp = readData.substring(0, firstcommaidx);
    lux = readData.substring(firstcommaidx + 1);
    readData = "";

    dStr = getDateString();
    mdata[0] = dStr; //date
    mdata[1] = temp; //data
    mdata[2] = lux;
    console.log("AA00," + mdata);
    io.sockets.emit("message", mdata); // send data
  } else {
    console.log(readData);
  }
});
```

시간,온도,조도

Arduino data on network socket

Google search

socket.io.js cdn



cdnjs.com

<cdnjs>

Search from 3,934 libraries on cdnjs...

Home / Libraries / socket.io

socket.io node.js realtime framework server

51k GitHub package 0 vulnerabilities

MIT licensed https://socket.io

Tags: websockets, node, popular

Version: 2.3.1

Asset Type: All

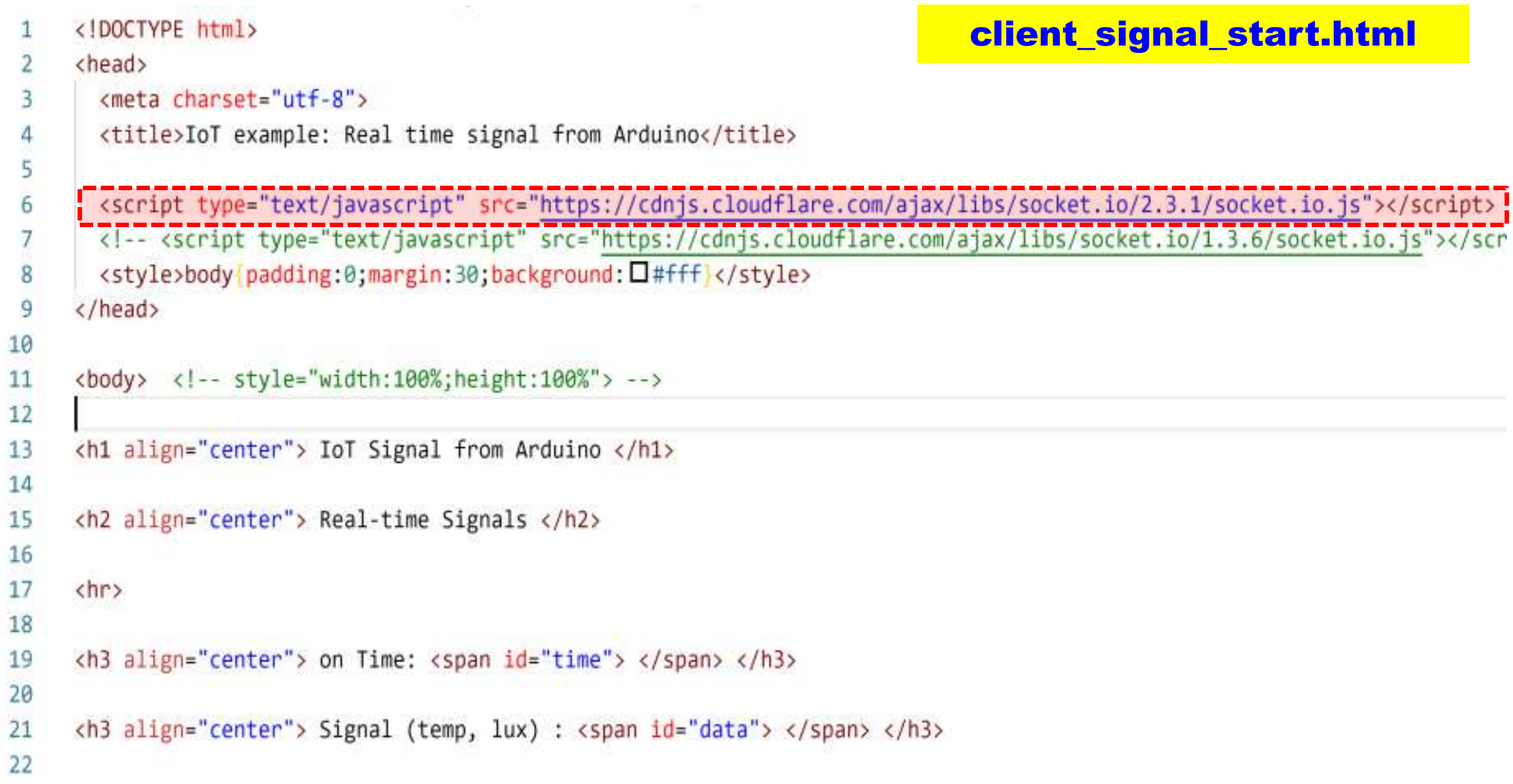
https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.3.1/socket.io.js

https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.3.1/socket.io.dev.js

https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.3.1/socket.io.dev.js.map

Arduino data on network socket

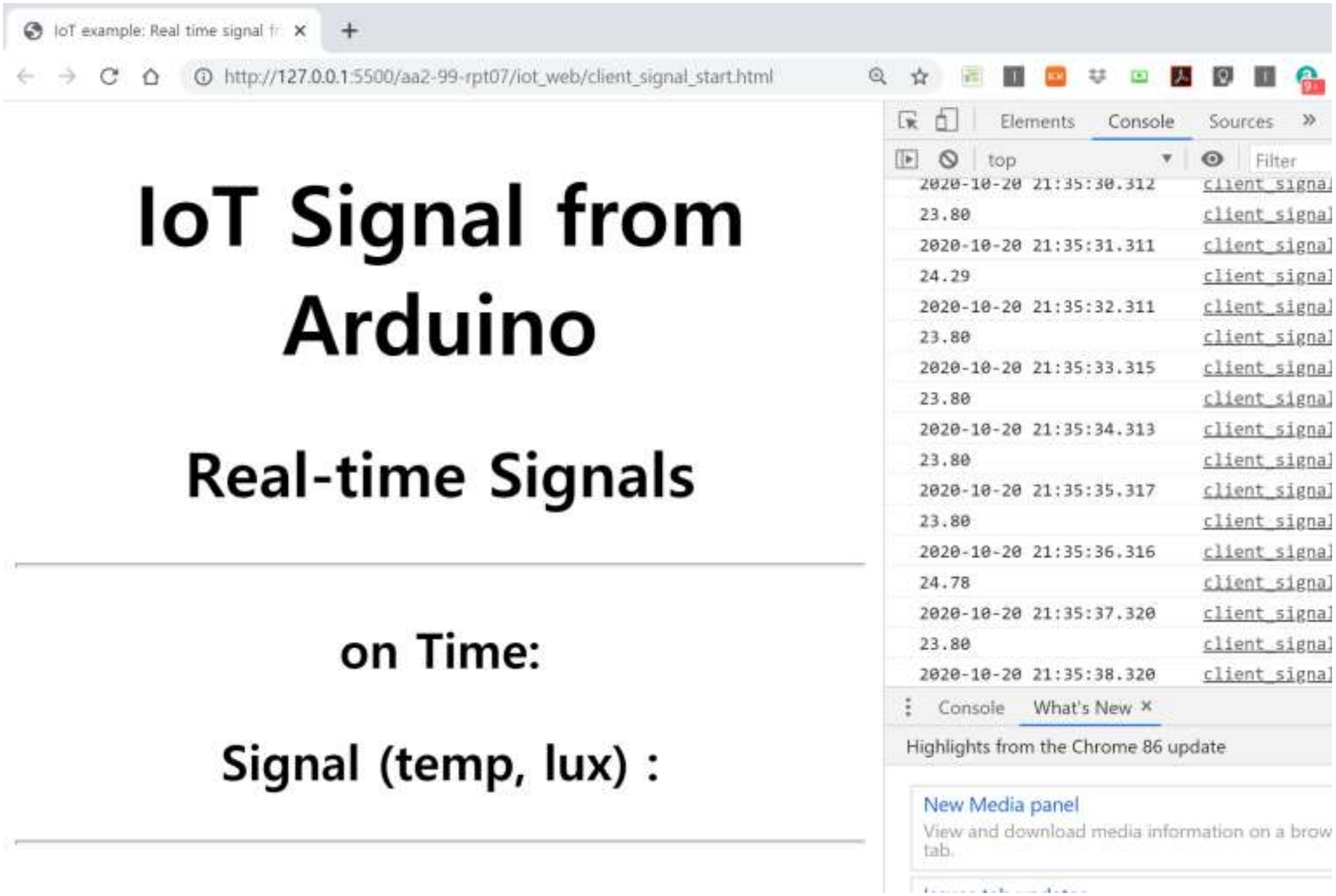
client_signal_start.html



```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <title>IoT example: Real time signal from Arduino</title>

  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.3.1/socket.io.js"></script>
  <!-- <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/socket.io.js"></scr
  <style>body{padding:0;margin:30;background:□#fff}</style>
</head>

<body>  <!-- style="width:100%;height:100%"> -->

<h1 align="center"> IoT Signal from Arduino </h1>

<h2 align="center"> Real-time Signals </h2>

<hr>

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

<h3 align="center"> Signal (temp, lux) : <span id="data"> </span> </h3>

```

Google search : socket.io.js cdn

Arduino data on network socket



IoT example: Real time signal fr

http://127.0.0.1:5500/aa2-99-rpt07/iot_web/client_signal_start.html

IoT Signal from Arduino

Real-time Signals

on Time:

Signal (temp, lux) :

```
2020-10-20 21:35:30.312    client_signal
23.80                      client_signal
2020-10-20 21:35:31.311    client_signal
24.29                      client_signal
2020-10-20 21:35:32.311    client_signal
23.80                      client_signal
2020-10-20 21:35:33.315    client_signal
23.80                      client_signal
2020-10-20 21:35:34.313    client_signal
23.80                      client_signal
2020-10-20 21:35:35.317    client_signal
23.80                      client_signal
2020-10-20 21:35:36.316    client_signal
24.78                      client_signal
2020-10-20 21:35:37.320    client_signal
23.80                      client_signal
2020-10-20 21:35:38.320    client_signal
```

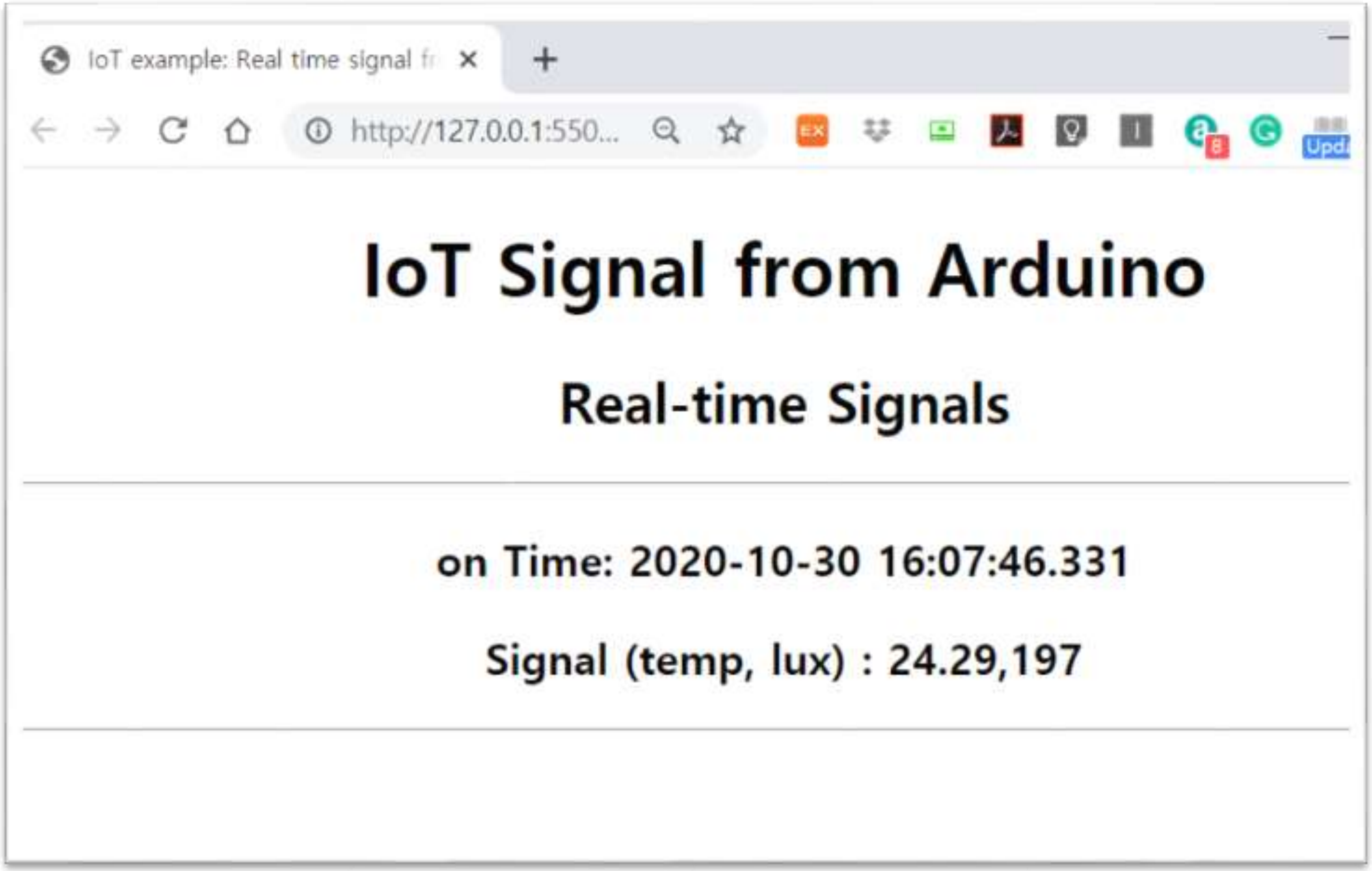
Highlights from the Chrome 86 update

New Media panel

View and download media information on a brow tab.

Real-time console showing a signal from Arduino in Chrome browser – F12

Arduino data on network socket



IoT Signal from Arduino

Real-time Signals

on Time: 2020-10-30 16:07:46.331

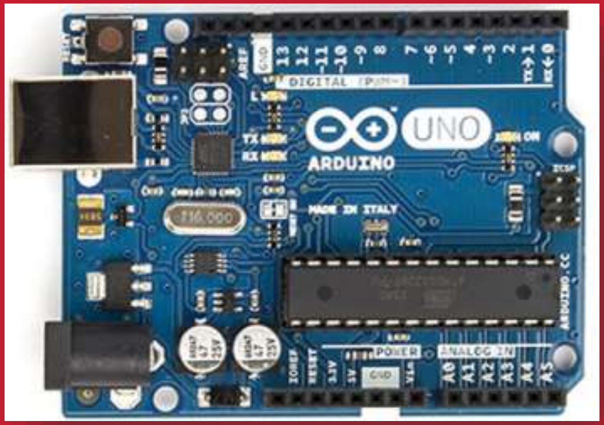
Signal (temp, lux) : 24.29,197

Real-time monitoring of a signal from Arduino
tmp36 + CdS circuit

Single sensor: CdS

CdS (LDR)

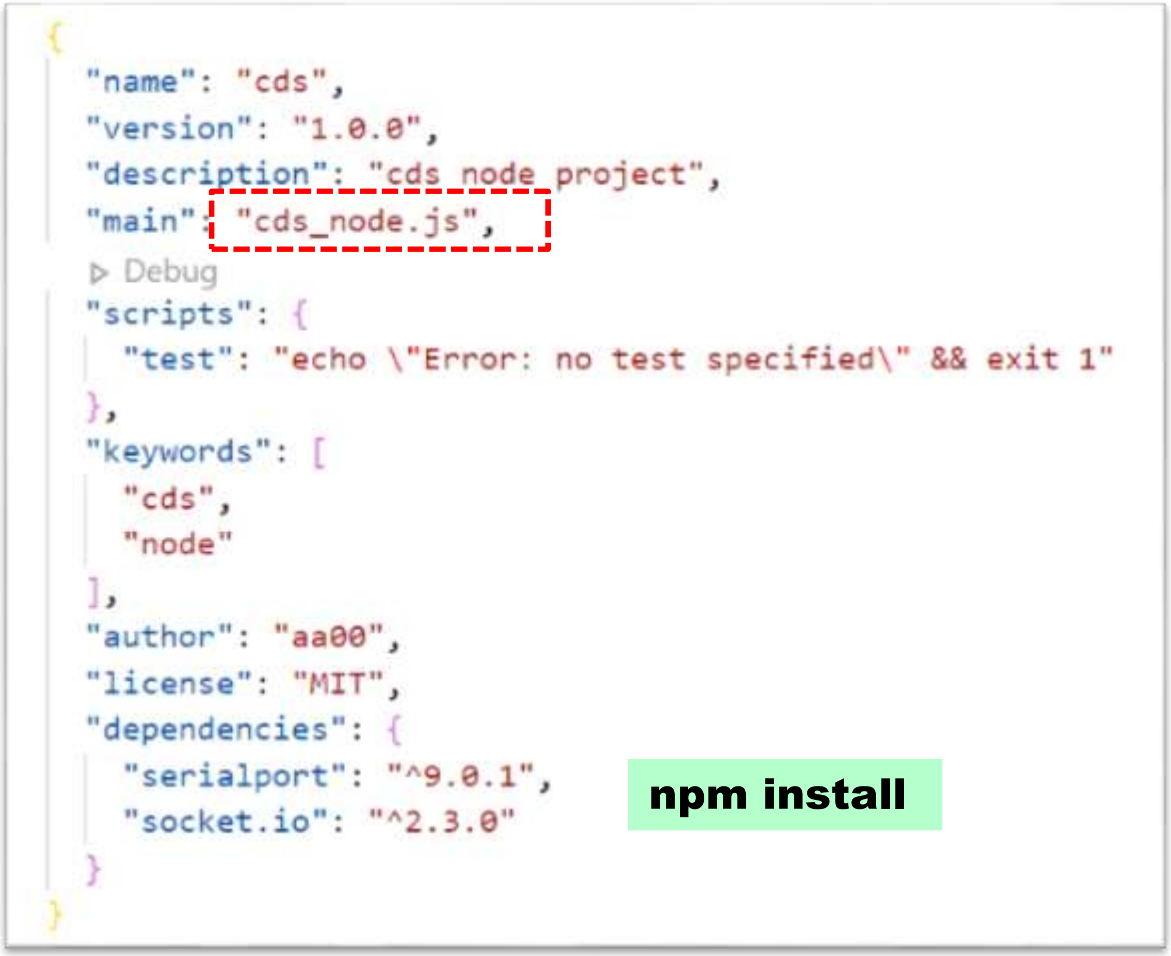
Node project



A4.2.1 Luminosity sensor [Photocell LDR]

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

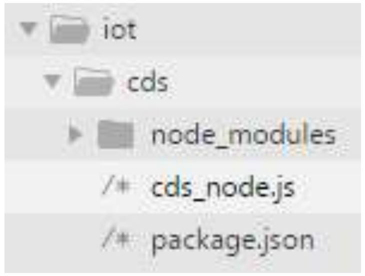
package,json



```
{
  "name": "cds",
  "version": "1.0.0",
  "description": "cds node project",
  "main": "cds_node.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "cds",
    "node"
  ],
  "author": "aa00",
  "license": "MIT",
  "dependencies": {
    "serialport": "^9.0.1",
    "socket.io": "^2.3.0"
  }
}
```

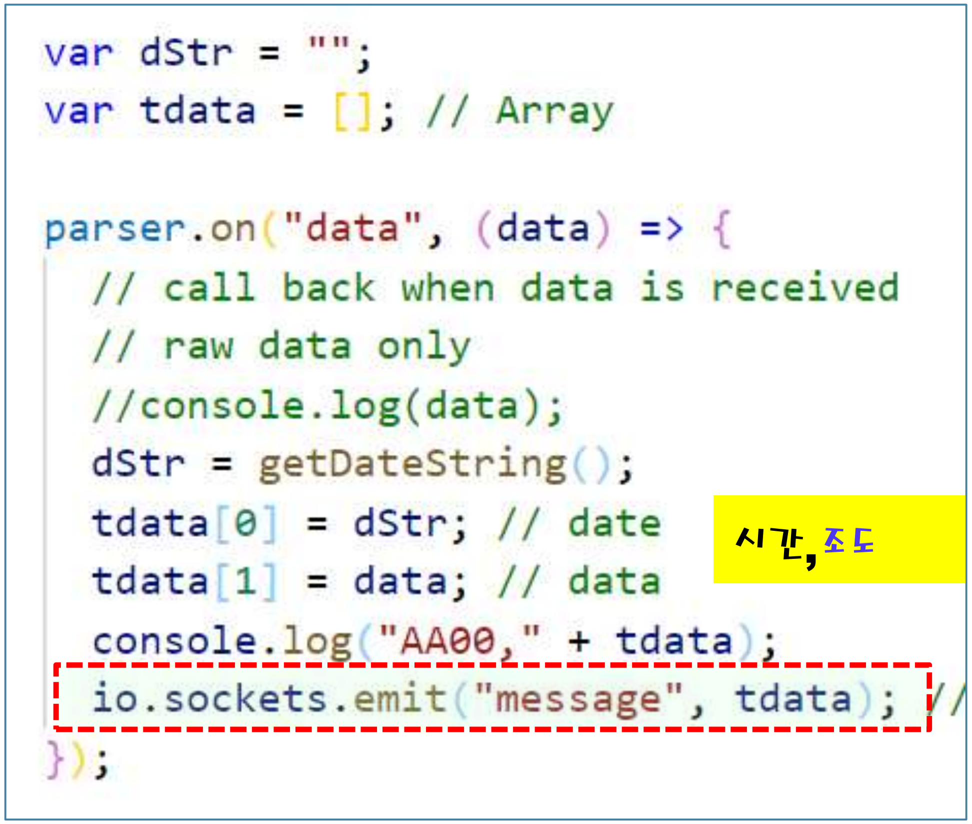
npm install

A4.2.2 Luminosity sensor [Photocell LDR]



- iot
  - cds
    - node_modules
    - cds_node.js
    - package.json

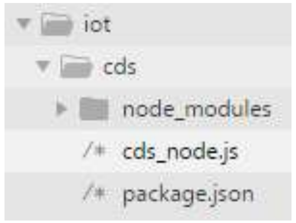
cds_node.js



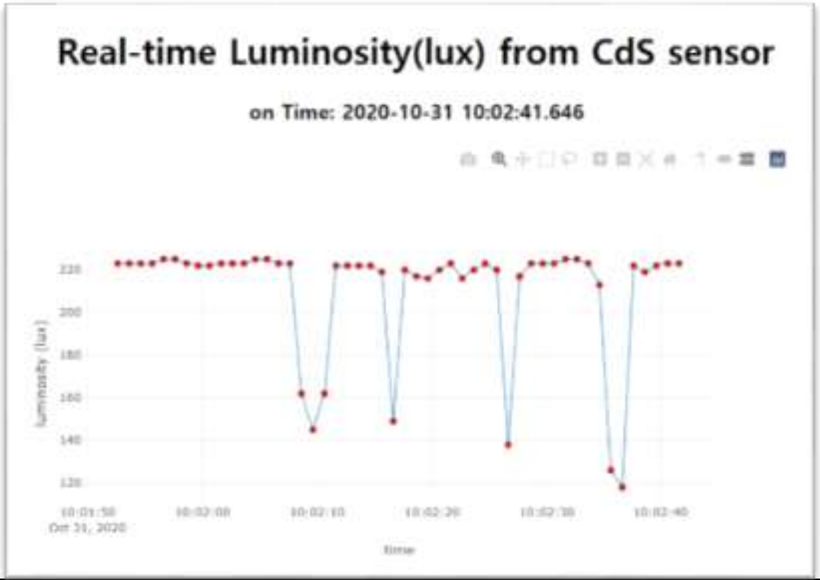
```
var dStr = "";
var tdata = []; // Array

parser.on("data", (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); //
});
```

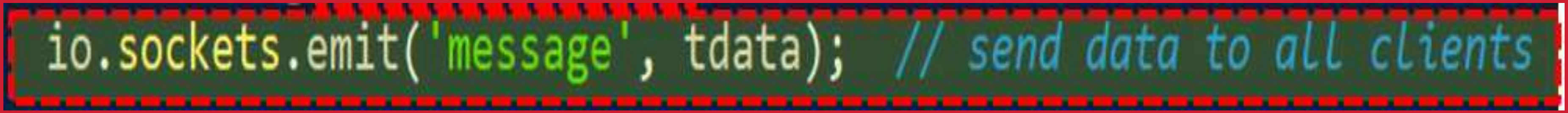
시간, 조도

A4.2.3 cds_ node project (실행 결과)

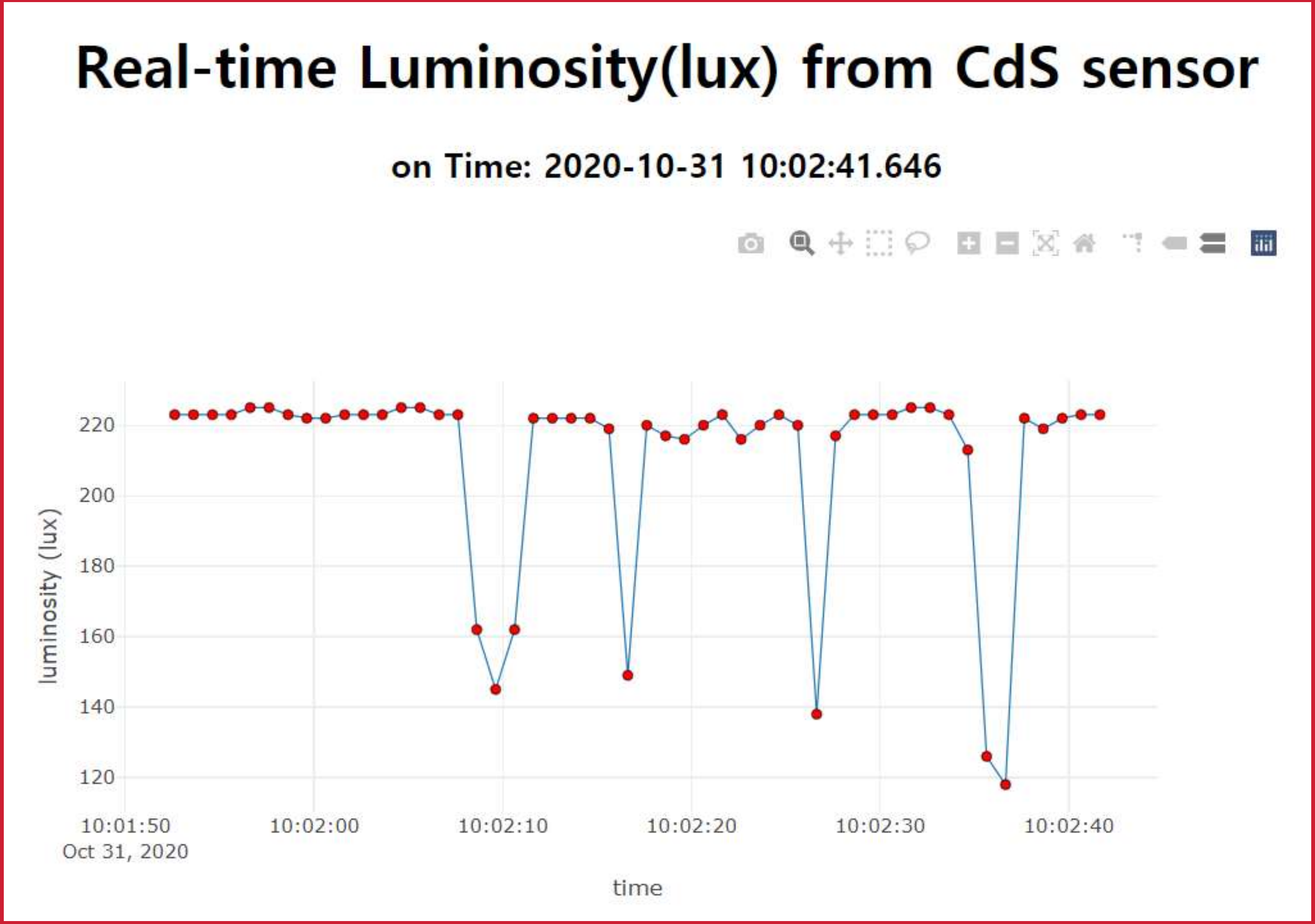

```
iot
  cds
    node_modules
    cds_node.js
    package.json
```

```
D:\Portable\vscode-portable\data\aa2-00\aa2-99-rpt08\wk11_src_start\Node\cds>node cds_node
serial port open
AA00,2020-10-31 09:40:24.912,220
AA00,2020-10-31 09:40:25.910,220
AA00,2020-10-31 09:40:26.914,220
AA00,2020-10-31 09:40:27.913,220
AA00,2020-10-31 09:40:28.912,222
AA00,2020-10-31 09:40:29.912,220
AA00,2020-10-31 09:40:30.915,220
AA00,2020-10-31 09:40:31.914,91
AA00,2020-10-31 09:40:32.914,217
AA00,2020-10-31 09:40:33.917,220
```





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

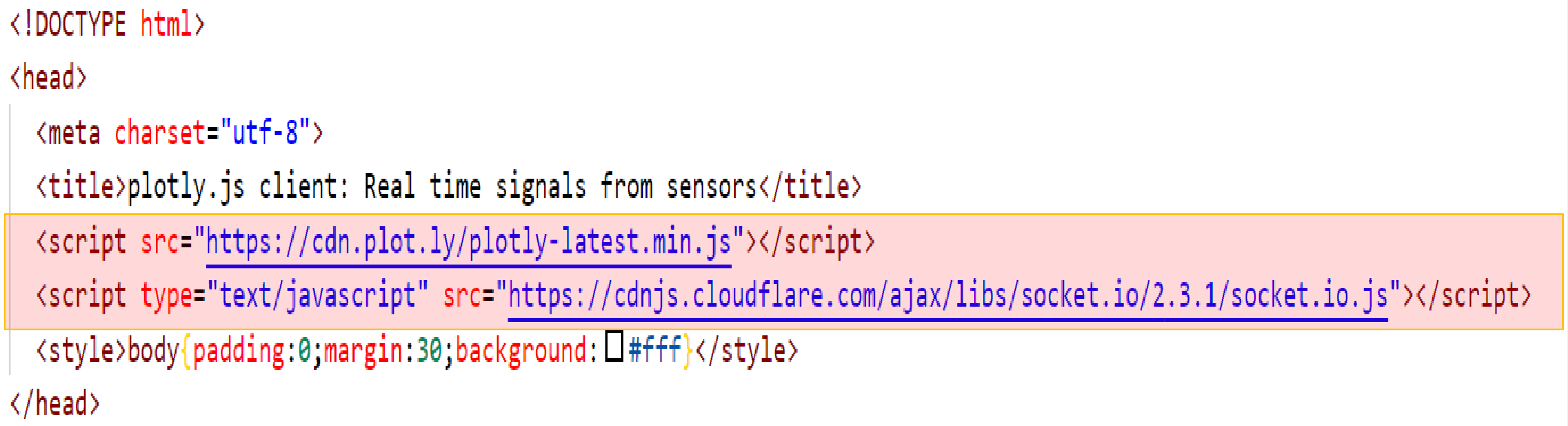


Real-time Luminosity(lux) from CdS sensor

on Time: 2020-10-31 10:02:41.646

A5.5.1 RT sensor-data streaming in Arduino

[1] Client html : client_cds.html (using socket.io.js & plotly.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/2.3.1/socket.io.js"></script>
  <style>body{padding:0;margin:30;background:□#fff}</style>
</head>
```

A5.5.2 RT sensor-data streaming in Arduino

[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();
    });
});
```

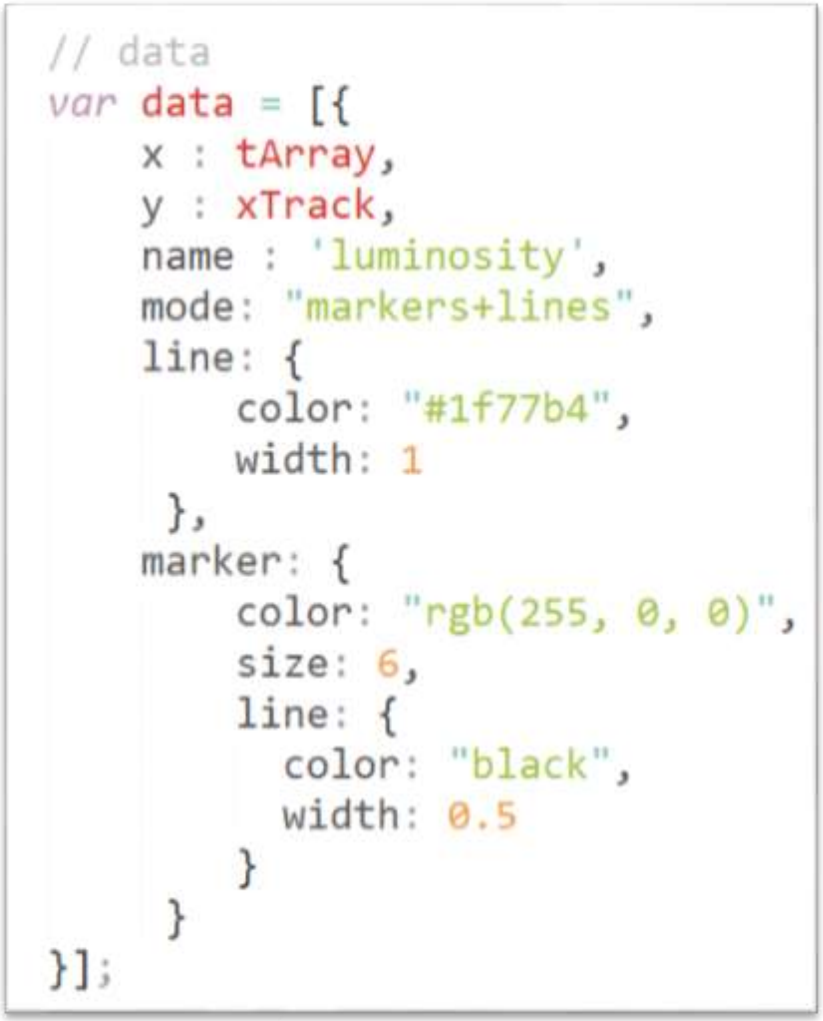
A5.5.4 RT sensor-data streaming in Arduino

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

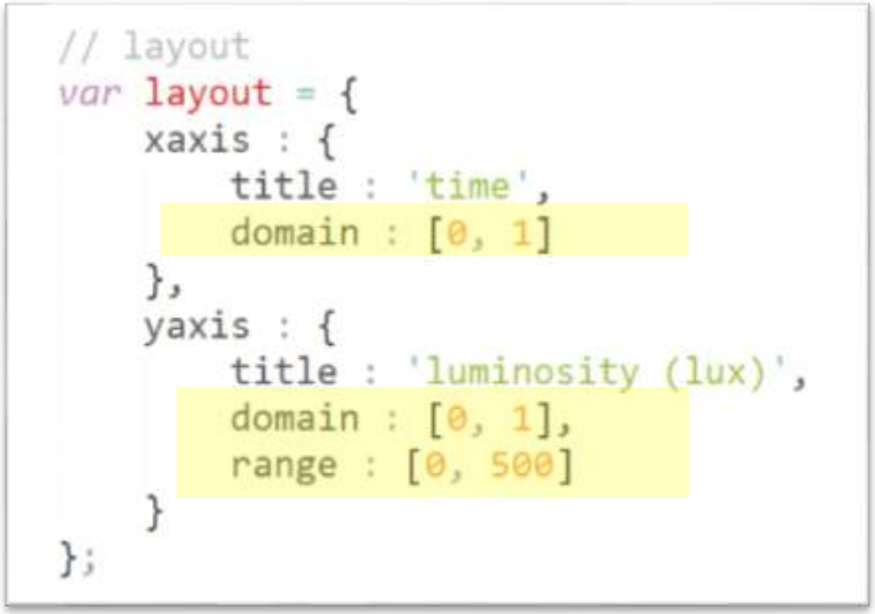
```
function init() {  // initial screen ()
    // starting point : first data (lux)
    for ( i = 0; i < numPts; i++) {
        tArray.push(dtda[0]);  // date
        xTrack.push(dtda[1]);  // CdS sensor (lux)
    }

    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);
}
```

A5.5.5 RT sensor-data streaming in Arduino

[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
        }
     }
}];
```



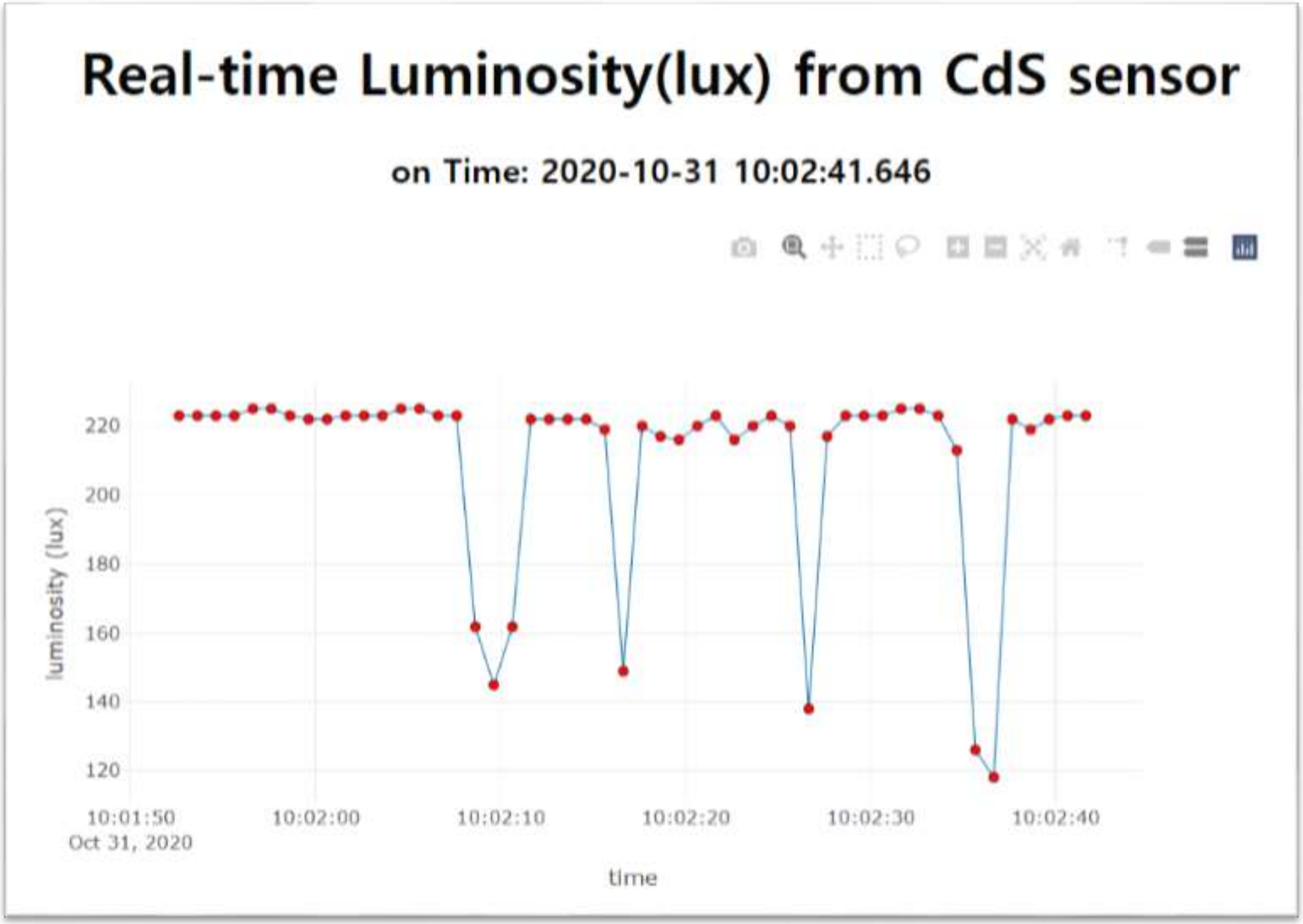
```
// layout
var layout = {
    xaxis : {
        title : 'time',
        domain : [0, 1]
    },
    yaxis : {
        title : 'luminosity (lux)',
        domain : [0, 1],
        range : [0, 500]
    }
};
```

domain: [0,1] → x 또는 y 축을 100% 사용

range: [0,500] → y 축의 범위를 0~500 설정

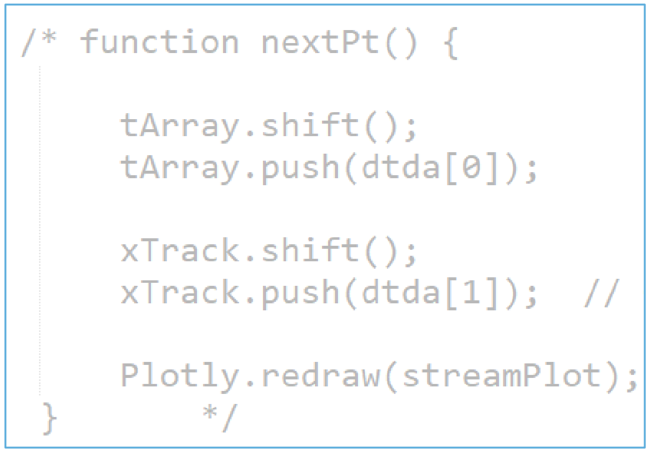
A5.5.6 RT sensor-data streaming in Arduino

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



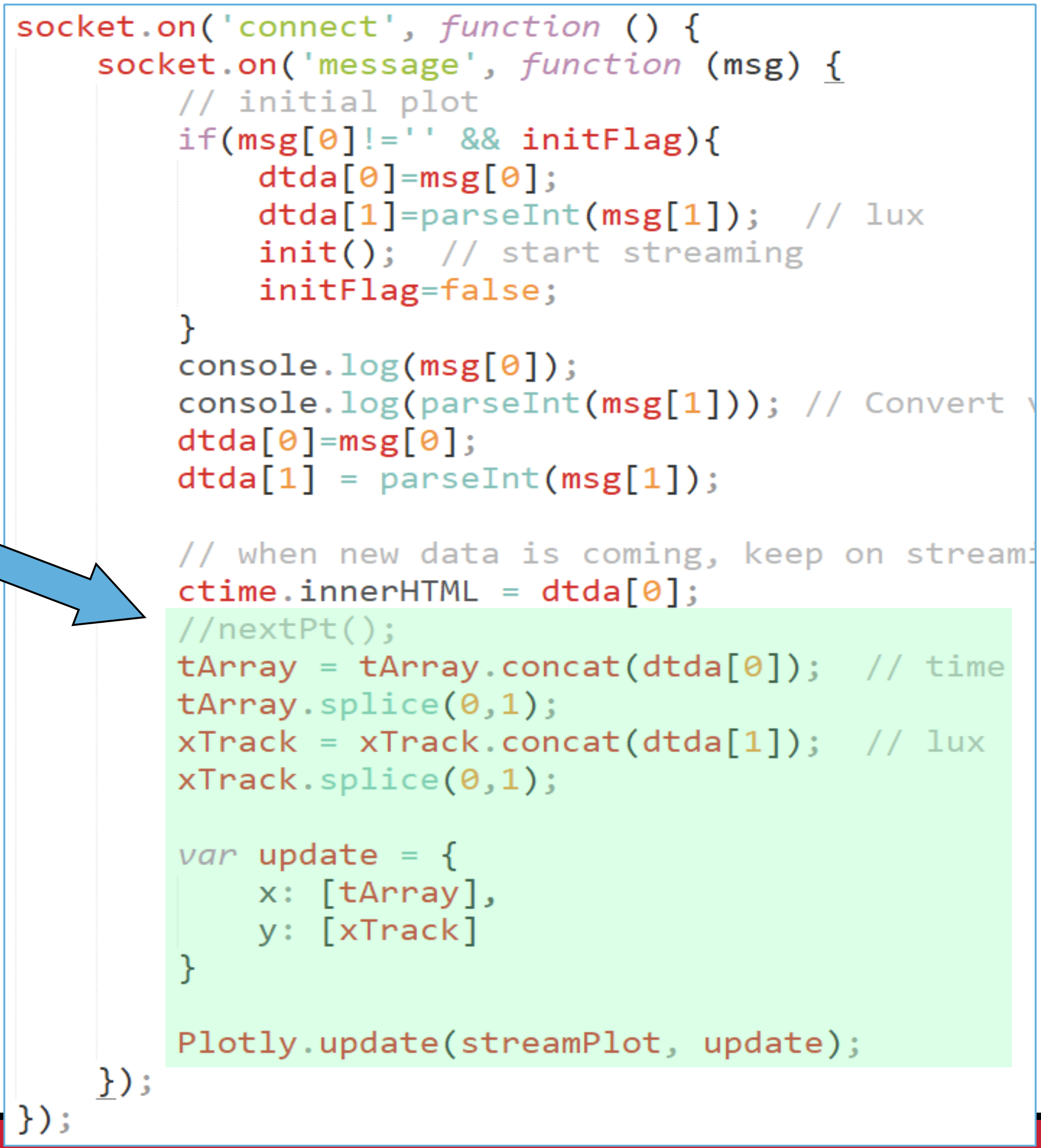
A5.5.7.1 RT sensor-data streaming in Arduino

[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);
 }      */
```

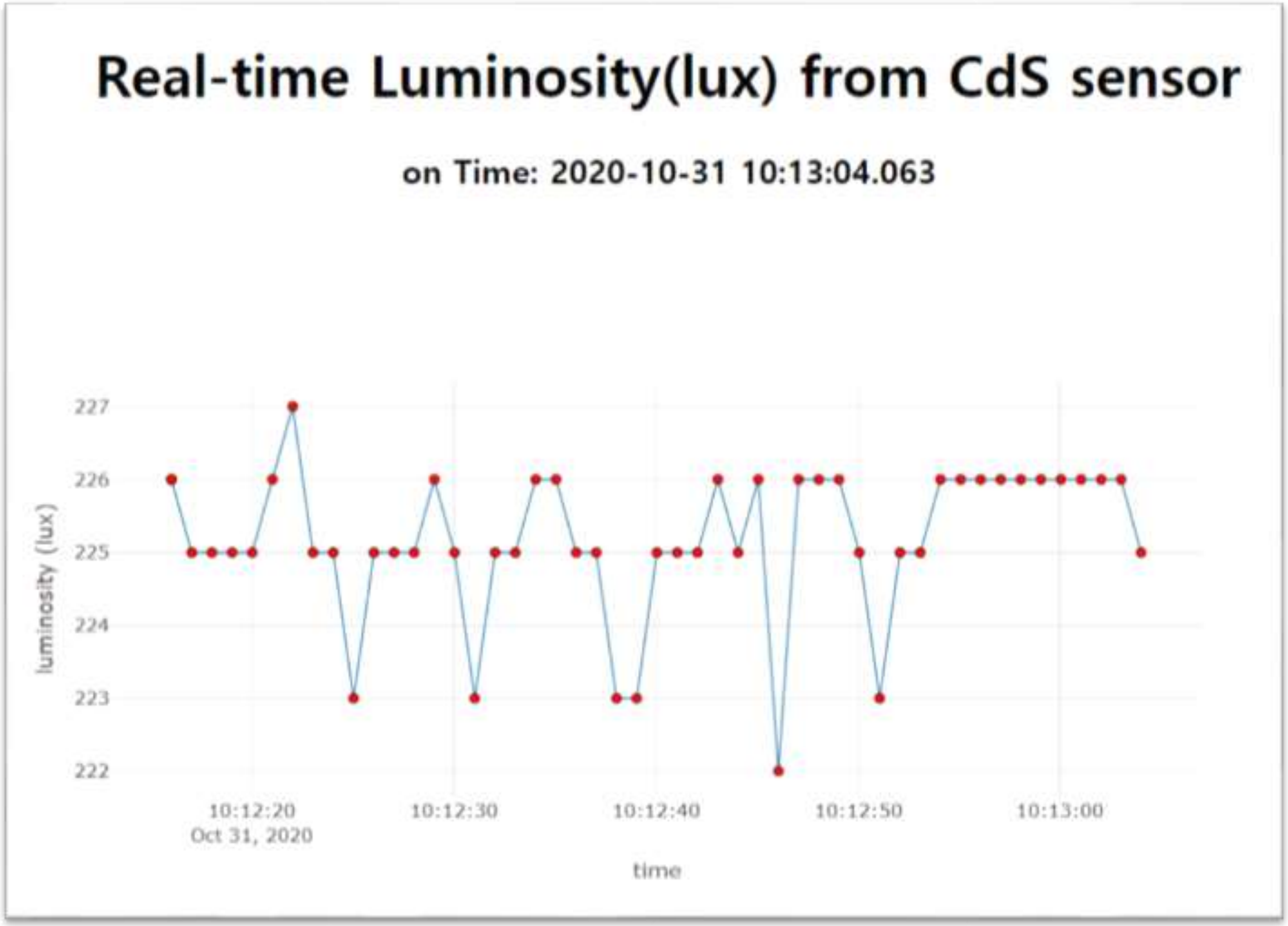
nextPt() 주석 처리



```
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 v
        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);
    });
});
```

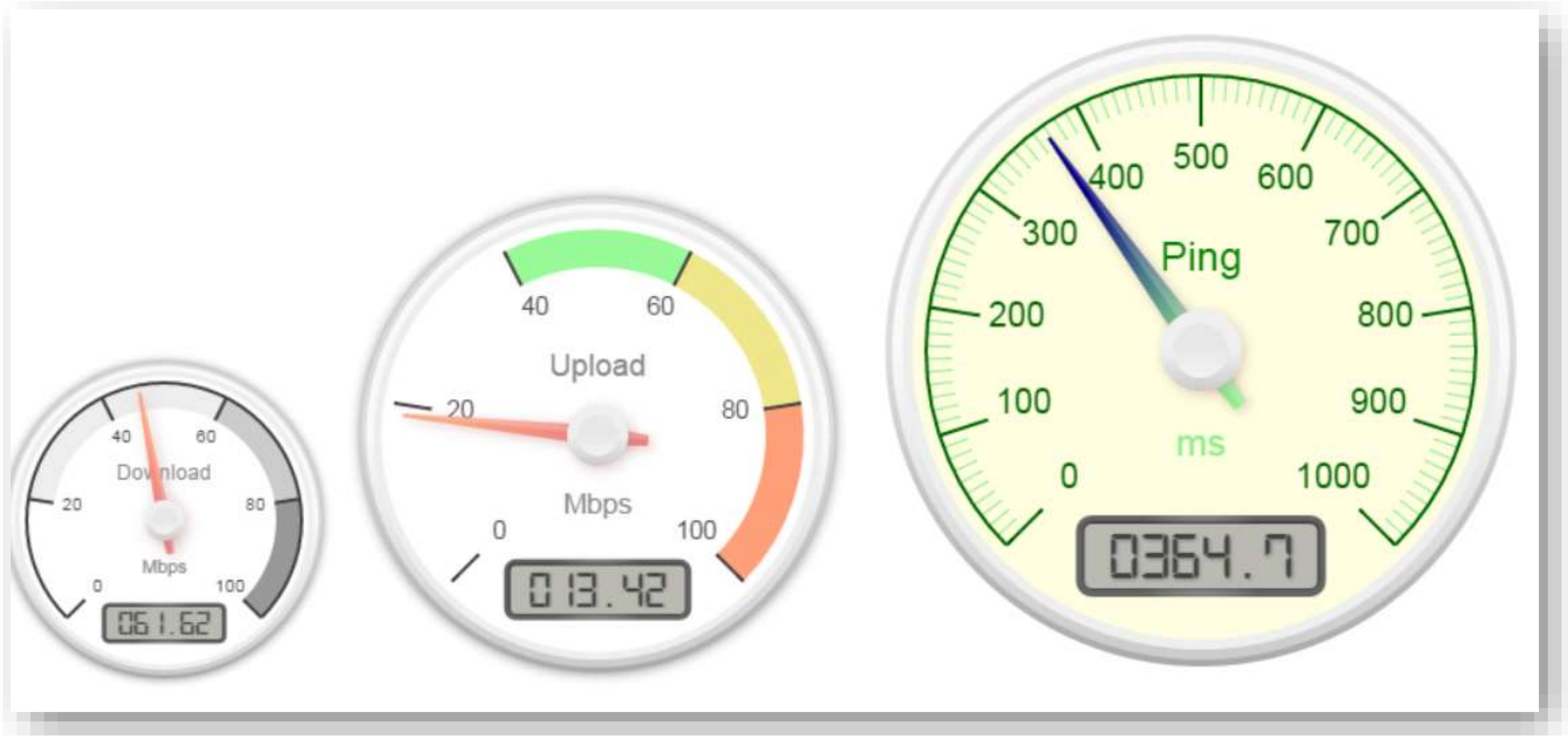
A5.5.7.2 RT sensor-data streaming in Arduino

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



Canvas Gauge

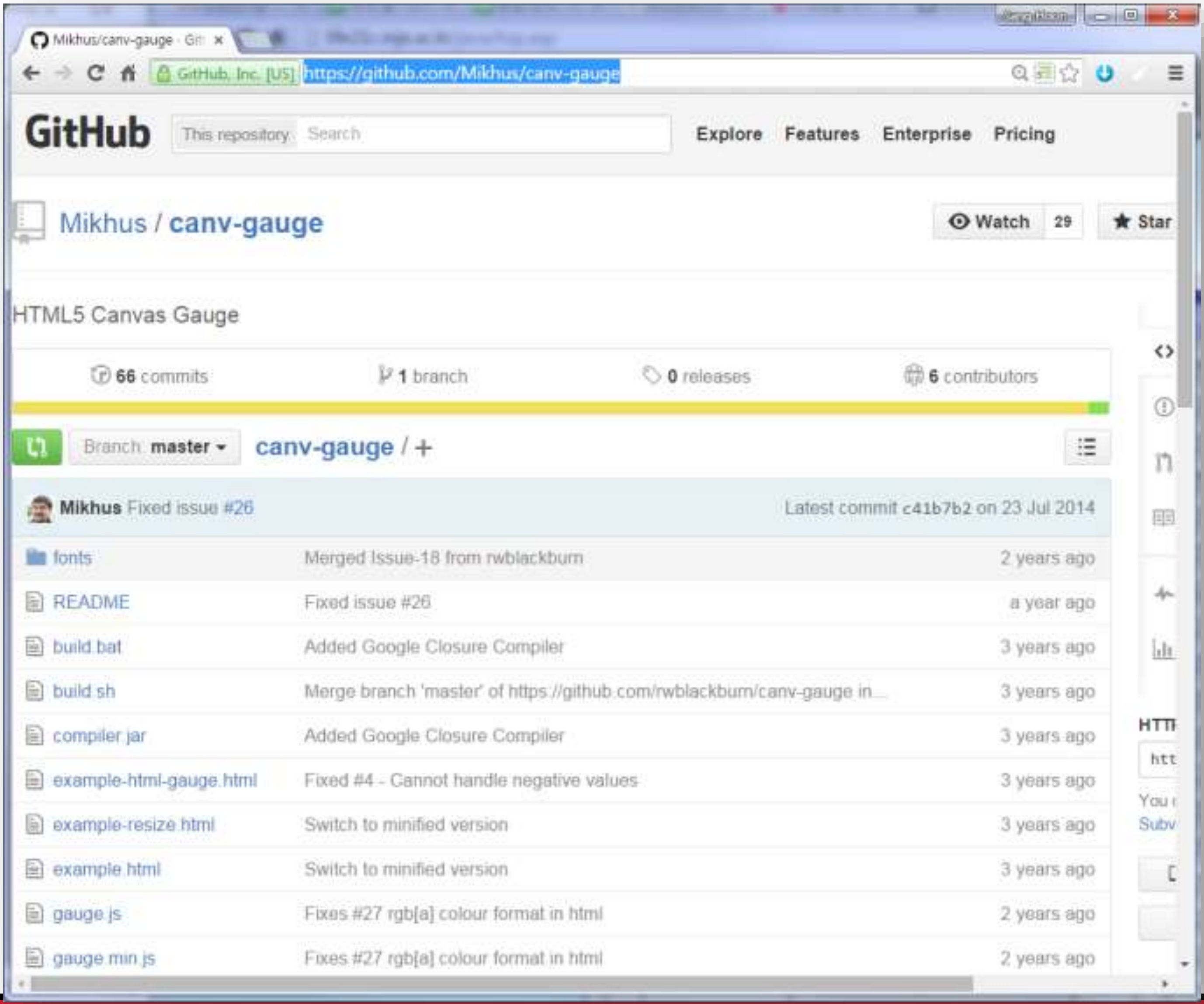
[1] Canvas gauge javascript library : example



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

Canvas Gauge

[2] Canvas gauge javascript library : gauge.js



https://github.com/Mikhus/canv-gauge

Mikhus / canv-gauge

HTML5 Canvas Gauge

66 commits · 1 branch · 0 releases · 6 contributors

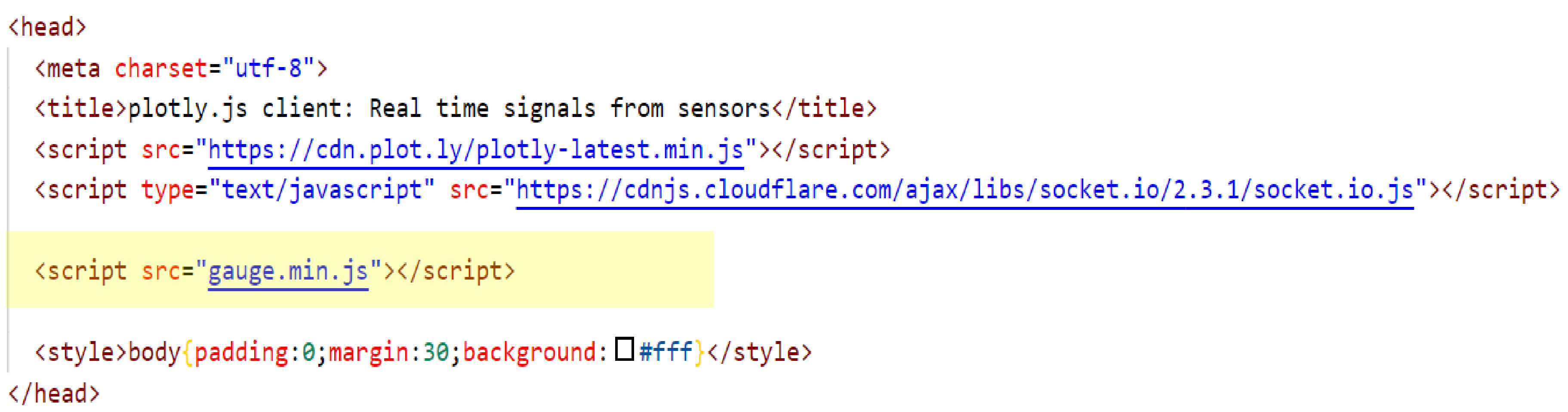
Branch: master — canv-gauge / +

Mikhus Fixed issue #26 — Latest commit c41b7b2 on 23 Jul 2014

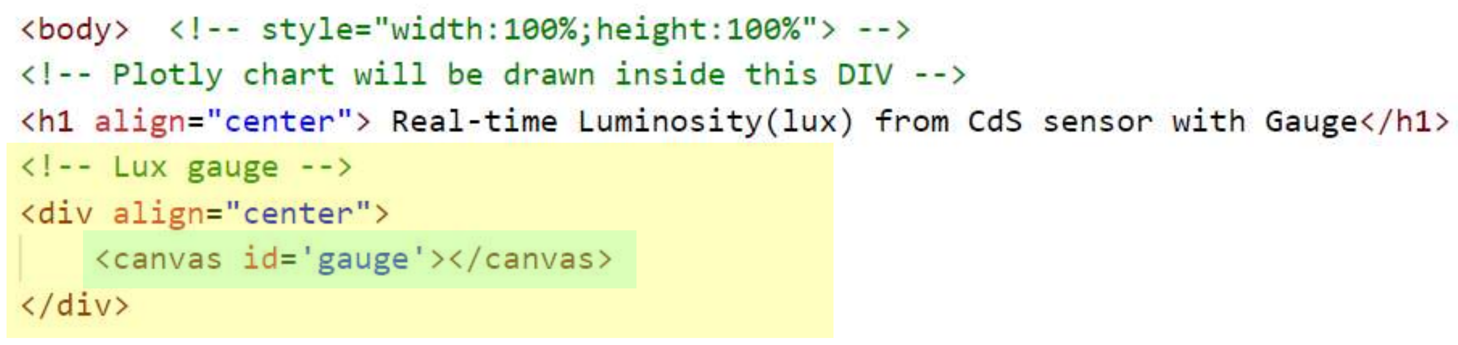
fonts	Merged Issue-18 from rwblackburn	2 years ago
README	Fixed issue #26	a year ago
build.bat	Added Google Closure Compiler	3 years ago
build.sh	Merge branch 'master' of https://github.com/rwblackburn/canv-gauge in...	3 years ago
compiler.jar	Added Google Closure Compiler	3 years ago
example-html-gauge.html	Fixed #4 - Cannot handle negative values	3 years ago
example-resize.html	Switch to minified version	3 years ago
example.html	Switch to minified version	3 years ago
gauge.js	Fixes #27 rgb[a] colour format in html	2 years ago
gauge.min.js	Fixes #27 rgb[a] colour format in html	2 years ago

A5.5.8.1 RT sensor-data streaming in Arduino

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



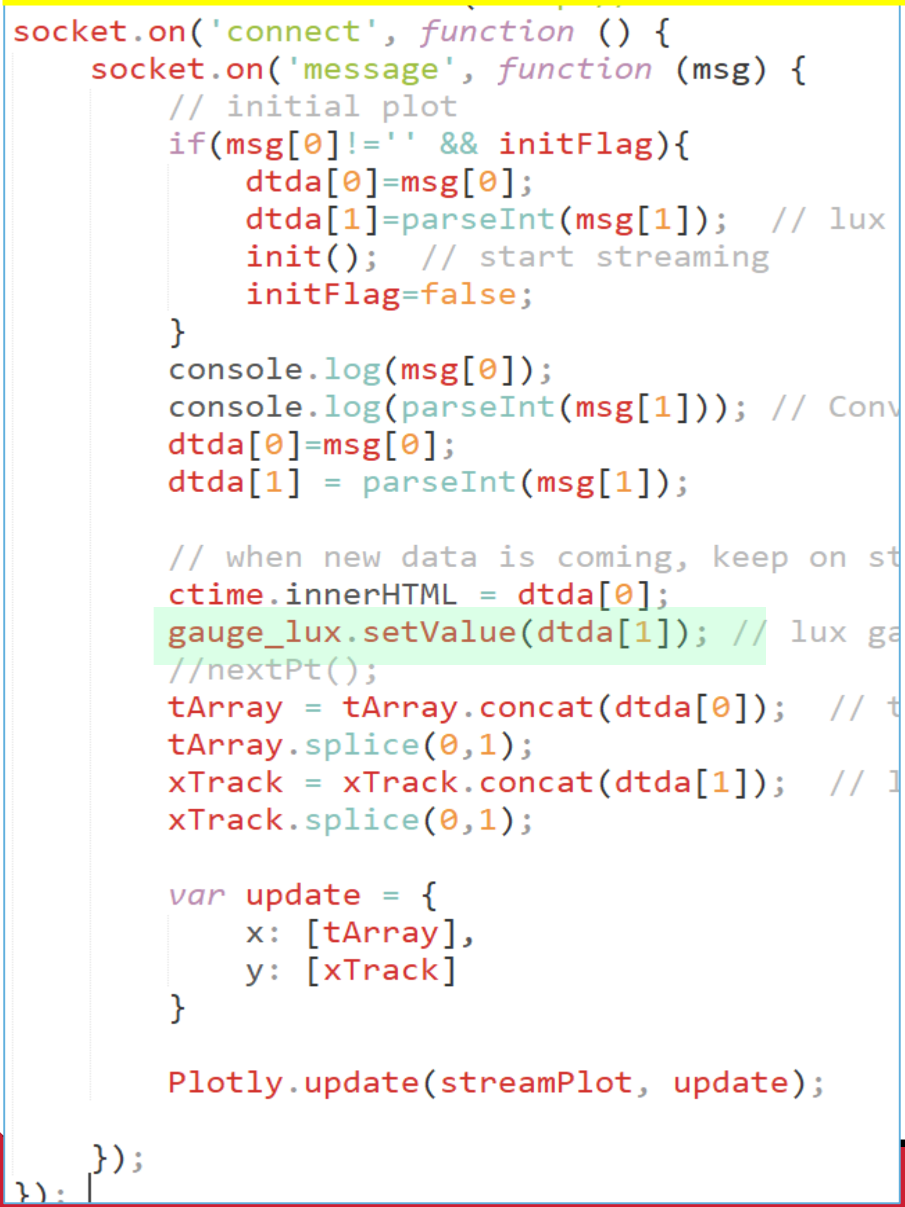
```
<head>
  <meta charset="utf-8">
  <title>plotly.js client: Real time signals from sensors</title>
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.3.1/socket.io.js"></script>

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

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



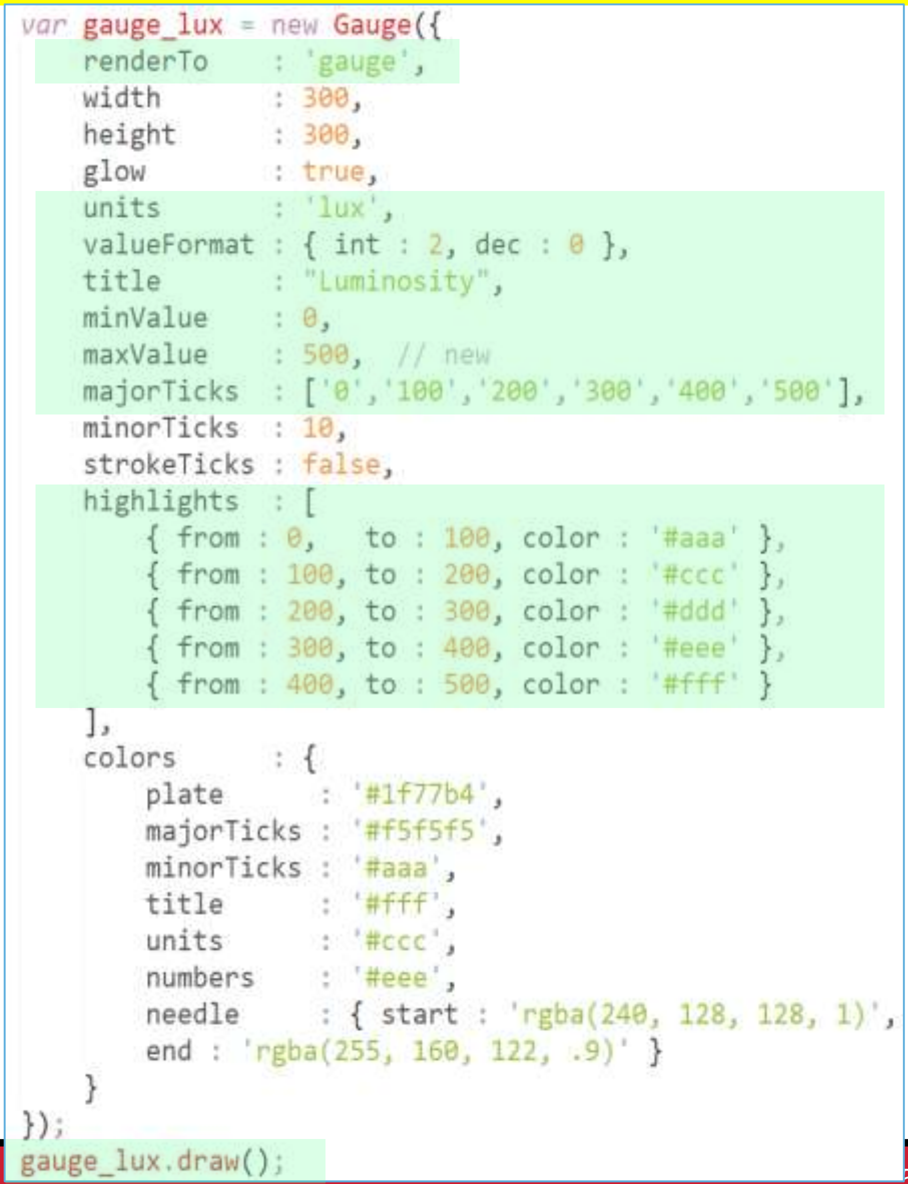
```
<body>  <!-- style="width:100%;height:100%"> -->
<!-- Plotly chart will be drawn inside this DIV -->
<h1 align="center"> Real-time Luminosity(lux) from CdS sensor with Gauge</h1>
<!-- Lux gauge -->
<div align="center">
    <canvas id='gauge'></canvas>
</div>
```

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);

    });
});
```

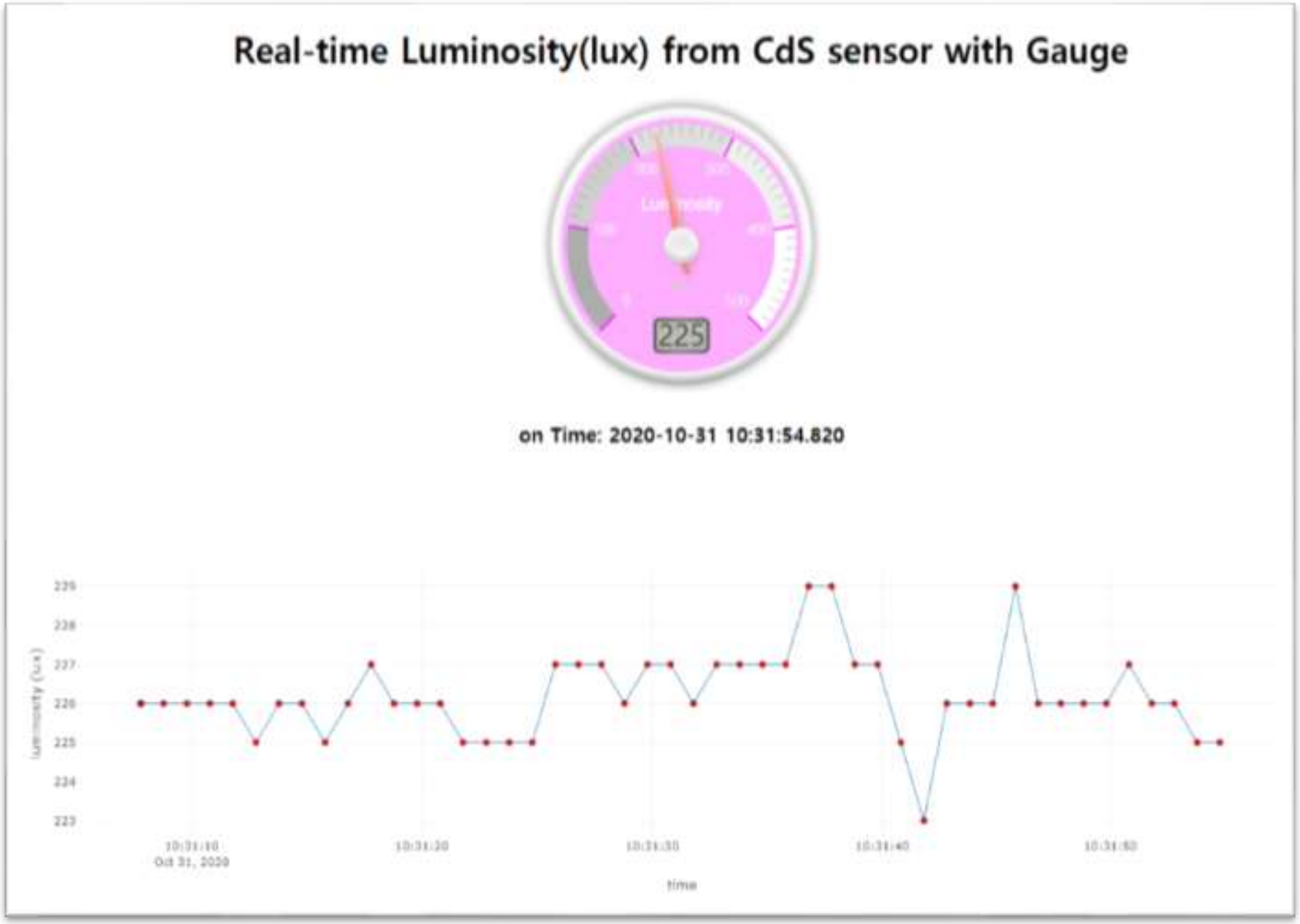


```
var gauge_lux = new Gauge({
    renderTo     : 'gauge',
    width        : 300,
    height       : 300,
    glow         : true,
    units        : 'lux',
    valueFormat  : { int : 2, dec : 0 },
    title        : "Luminosity",
    minValue     : 0,
    maxValue     : 500,  // new
    majorTicks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

A5.5.8.3 RT sensor-data streaming in Arduino

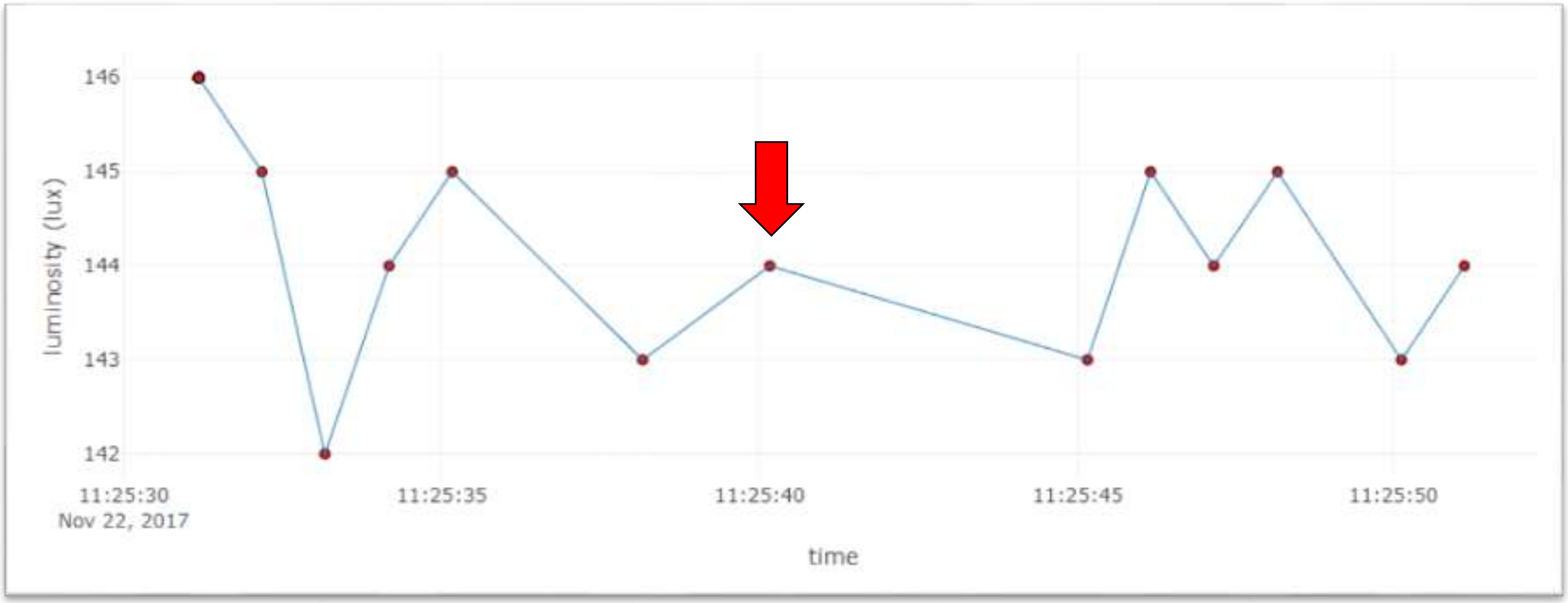
[DIY] Client html : client_cds_gauge.html (change design of Gauge)

변경된 디자인으로 된 그래프를 캡처하여 AAnn_cds_gauge.png 로 저장



A5.5.9.1 RT sensor-data streaming in Arduino

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



이상 감지 (anomaly detection)

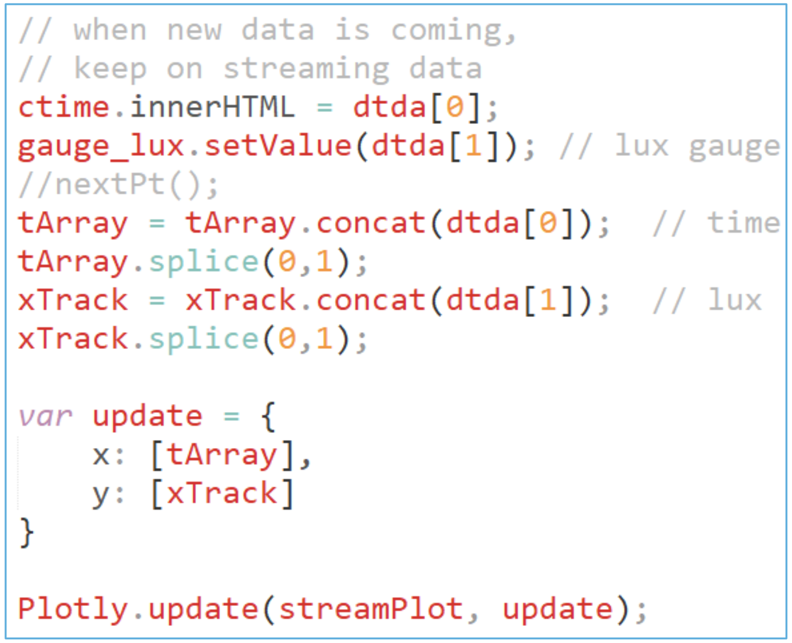
입력되는 lux 값이 변하는 경우에만 그래프를 그림.

실시간 모니터링에서 이상 감지 기능이 필요함.

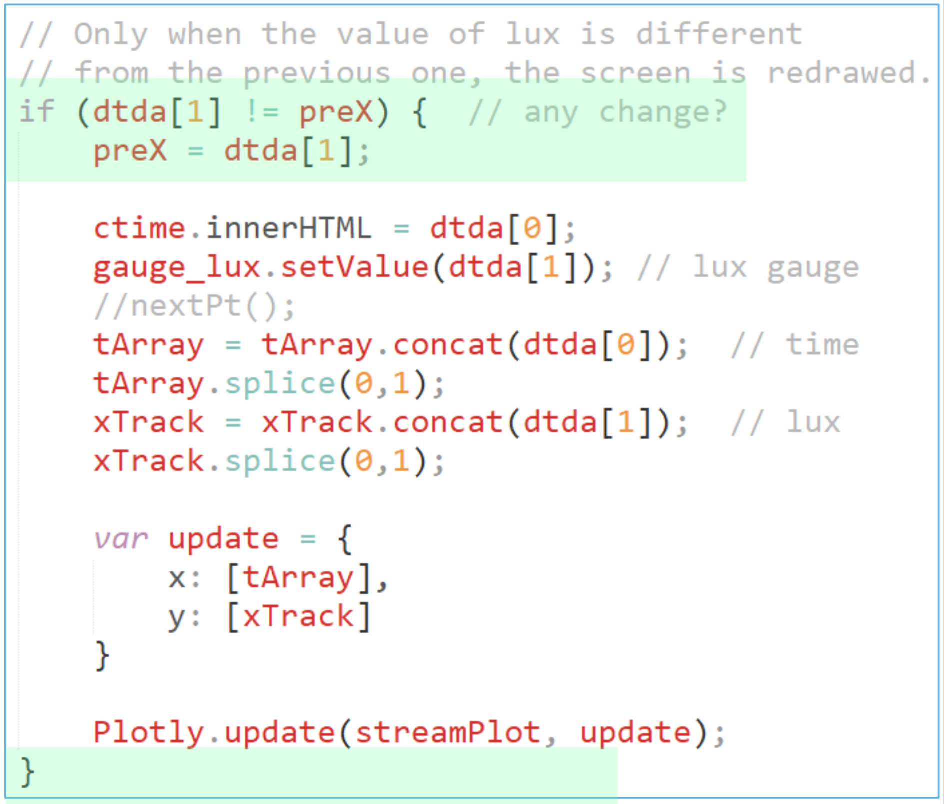
밝기 값 변화의 문턱값을 설정해서 이상 감지 기능 구현

A5.5.9.2 RT sensor-data streaming in Arduino

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



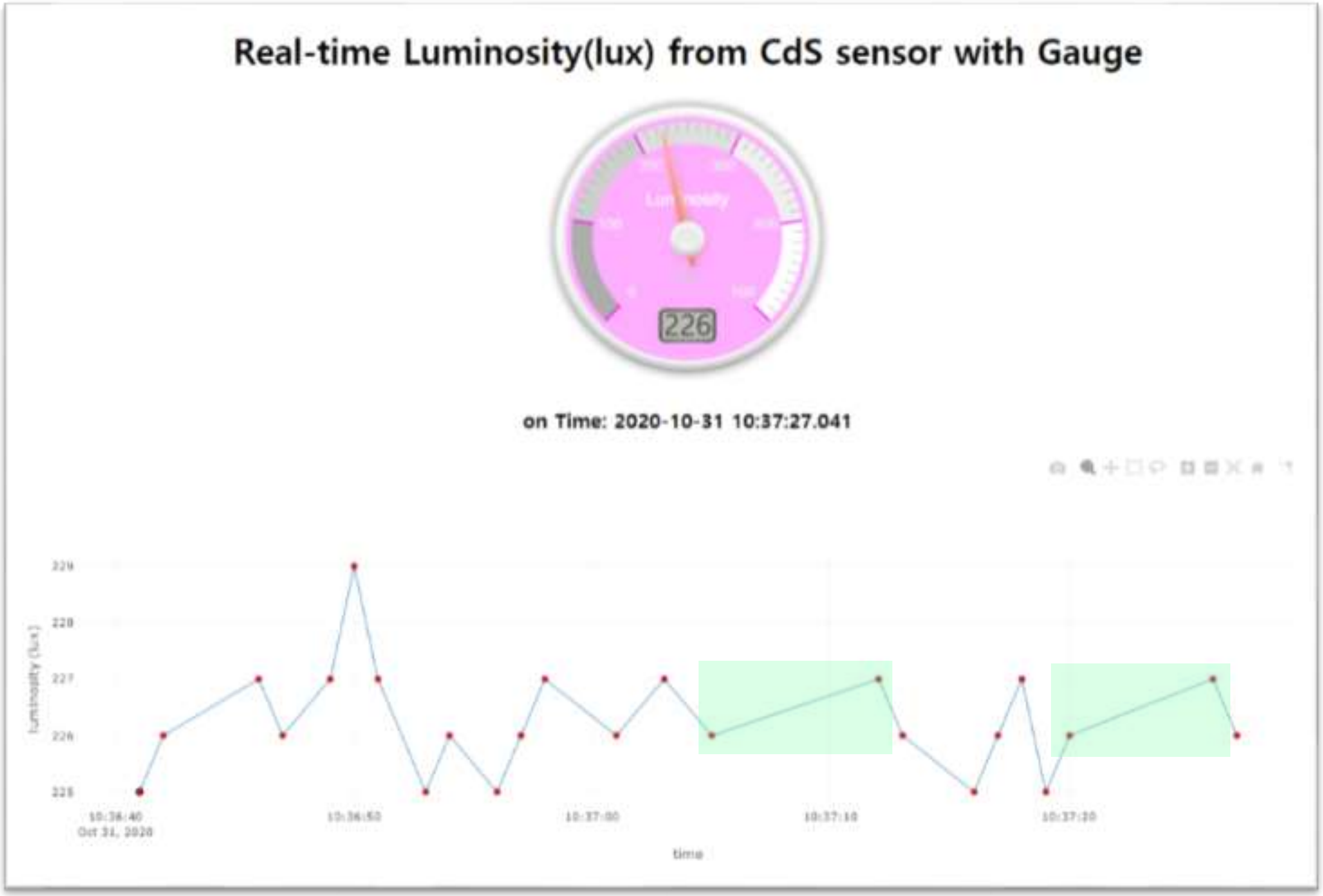
```
// when new data is coming,
// keep on streaming data
ctime.innerHTML = dtda[0];
gauge_lux.setValue(dtda[1]); // lux gauge
//nextPt();
tArray = tArray.concat(dtda[0]);  // time
tArray.splice(0,1);
xTrack = xTrack.concat(dtda[1]);  // lux
xTrack.splice(0,1);

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);
}
```

A5.5.9.3 RT sensor-data streaming in Arduino

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

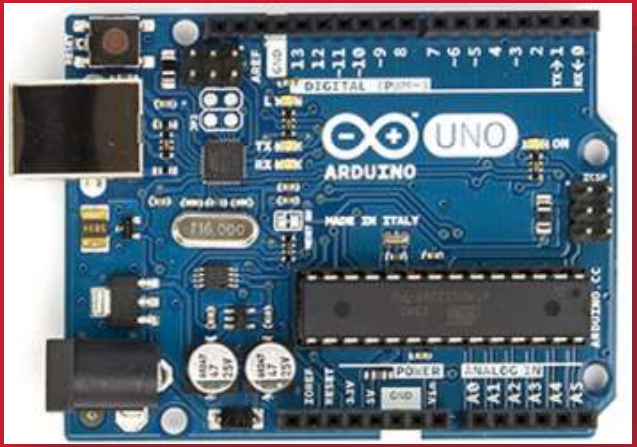

Real-time Luminosity(lux) from CdS sensor with Gauge

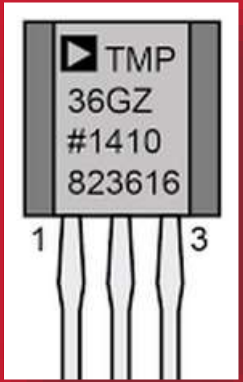
on Time: 2020-10-31 10:37:27.041

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

Multiple sensors

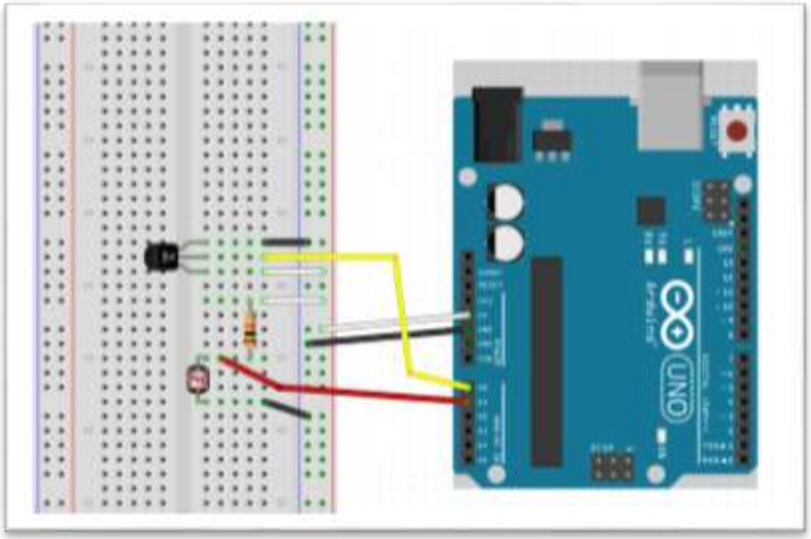
CdS + TMP36 + plotly.js Node project





CdS + TMP36 + Node.js + Plotly

tmp36 + CdS circuit



```
AA00,2020-10-17 11:41:30.533,25.27,245
AA00,2020-10-17 11:41:31.535,25.27,243
AA00,2020-10-17 11:41:32.535,25.27,158
AA00,2020-10-17 11:41:33.534,24.29,40
AA00,2020-10-17 11:41:34.538,24.29,33
AA00,2020-10-17 11:41:35.537,24.78,86
AA00,2020-10-17 11:41:36.541,25.27,249
AA00,2020-10-17 11:41:37.540,25.76,245
AA00,2020-10-17 11:41:38.543,25.76,243
AA00,2020-10-17 11:41:39.543,25.27,245
```

```
var readData = "";
var temp = "";
var lux = "";
var mdata = [];
var firstcommaidx = 0;

parser.on("data", (data) => {
  // call back when data is received
  readData = data.toString();
  firstcommaidx = readData.indexOf(",");
  if (firstcommaidx > 0) {
    temp = readData.substring(0, firstcommaidx);
    lux = readData.substring(firstcommaidx + 1);
    readData = "";

    dStr = getDateString();
    mdata[0] = dStr; //date
    mdata[1] = temp; //data
    mdata[2] = lux;
    console.log("AA00," + mdata);
    io.sockets.emit("message", mdata); // send data
  } else {
    console.log(readData);
  }
});
```

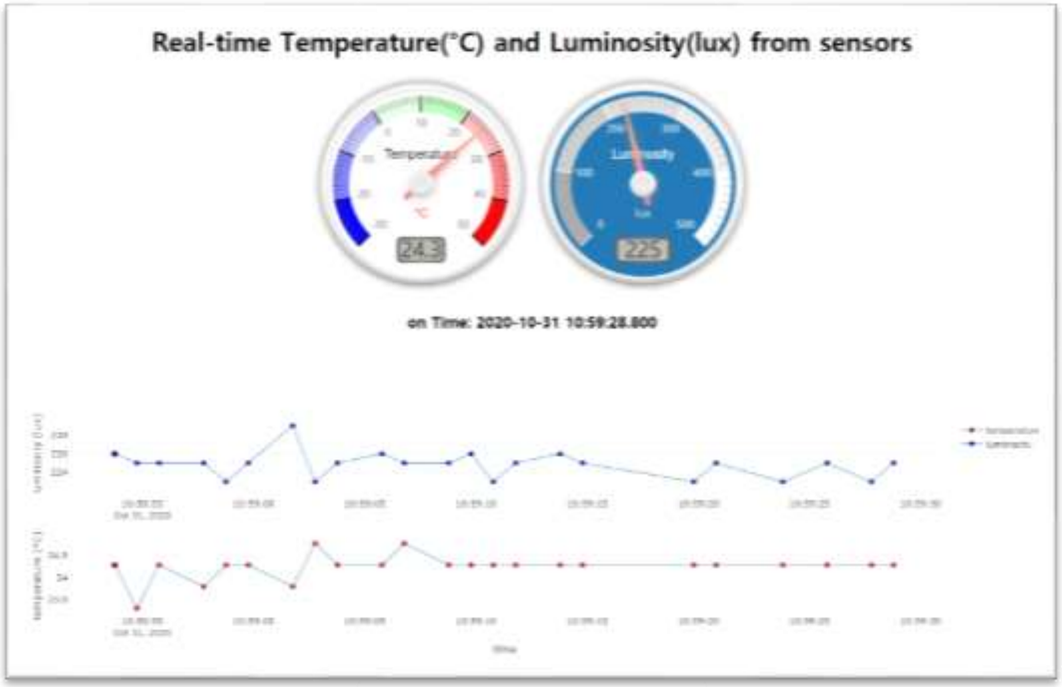
시간,온도,조도

A5.6.1 cds_ tmp36_node project (실행 결과)

```
D:\Portable\vscode-portable\data\aa2-00\aa2-99-rpt08\wk11_src_start\Node\cds_tmp36>node cds_tmp36_node
serial port open
AA00,2020-10-31 10:51:17.221,23.80,226
AA00,2020-10-31 10:51:18.220,24.29,226
AA00,2020-10-31 10:51:19.223,24.29,225
AA00,2020-10-31 10:51:20.223,24.29,225
AA00,2020-10-31 10:51:21.226,24.78,225
AA00,2020-10-31 10:51:22.226,25.27,225
AA00,2020-10-31 10:51:23.230,24.29,208
AA00,2020-10-31 10:51:24.229,25.27,213
AA00,2020-10-31 10:51:25.228,24.78,219
AA00,2020-10-31 10:51:26.232,24.29,193
AA00,2020-10-31 10:51:27.231,24.29,151
AA00,2020-10-31 10:51:28.234,24.29,225
AA00,2020-10-31 10:51:29.234,24.29,225
AA00,2020-10-31 10:51:30.237,24.29,225
AA00,2020-10-31 10:51:31.237,24.29,226
AA00,2020-10-31 10:51:32.236,24.29,226
AA00,2020-10-31 10:51:33.240,24.29,227
AA00,2020-10-31 10:51:34.239,24.29,223
AA00,2020-10-31 10:51:35.243,24.29,223
AA00,2020-10-31 10:51:36.242,24.29,225
AA00,2020-10-31 10:51:37.245,24.29,226
AA00,2020-10-31 10:51:38.245,24.29,226
█
```

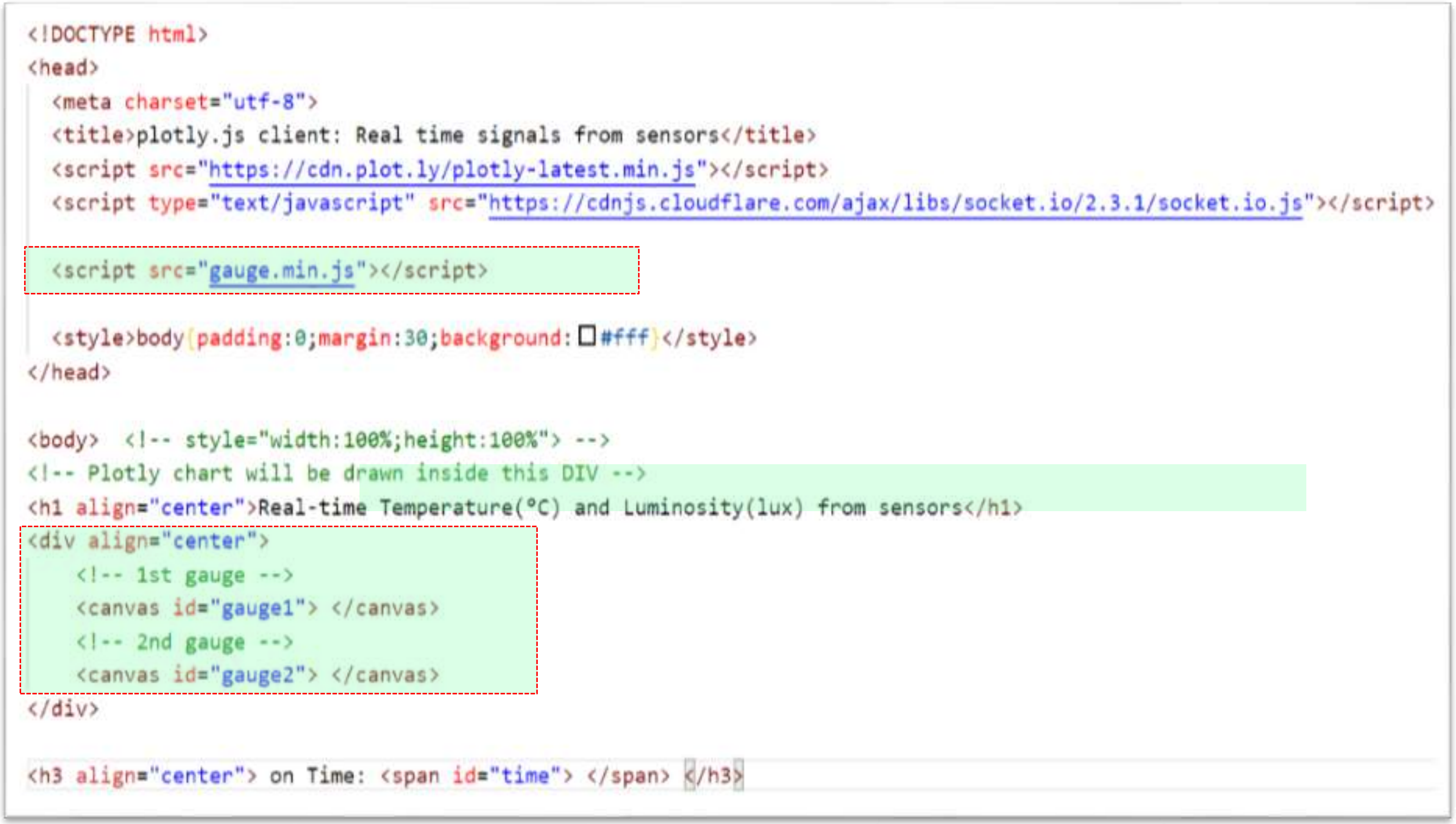
IOT data format

시간, 온도,조도


Real-time Temperature(°C) and Luminosity(lux) from sensors
Temperature
°C
24.3
Luminosity
lux
225
on Time: 2020-10-31 10:59:28.800

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/2.3.1/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>
```

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;
```

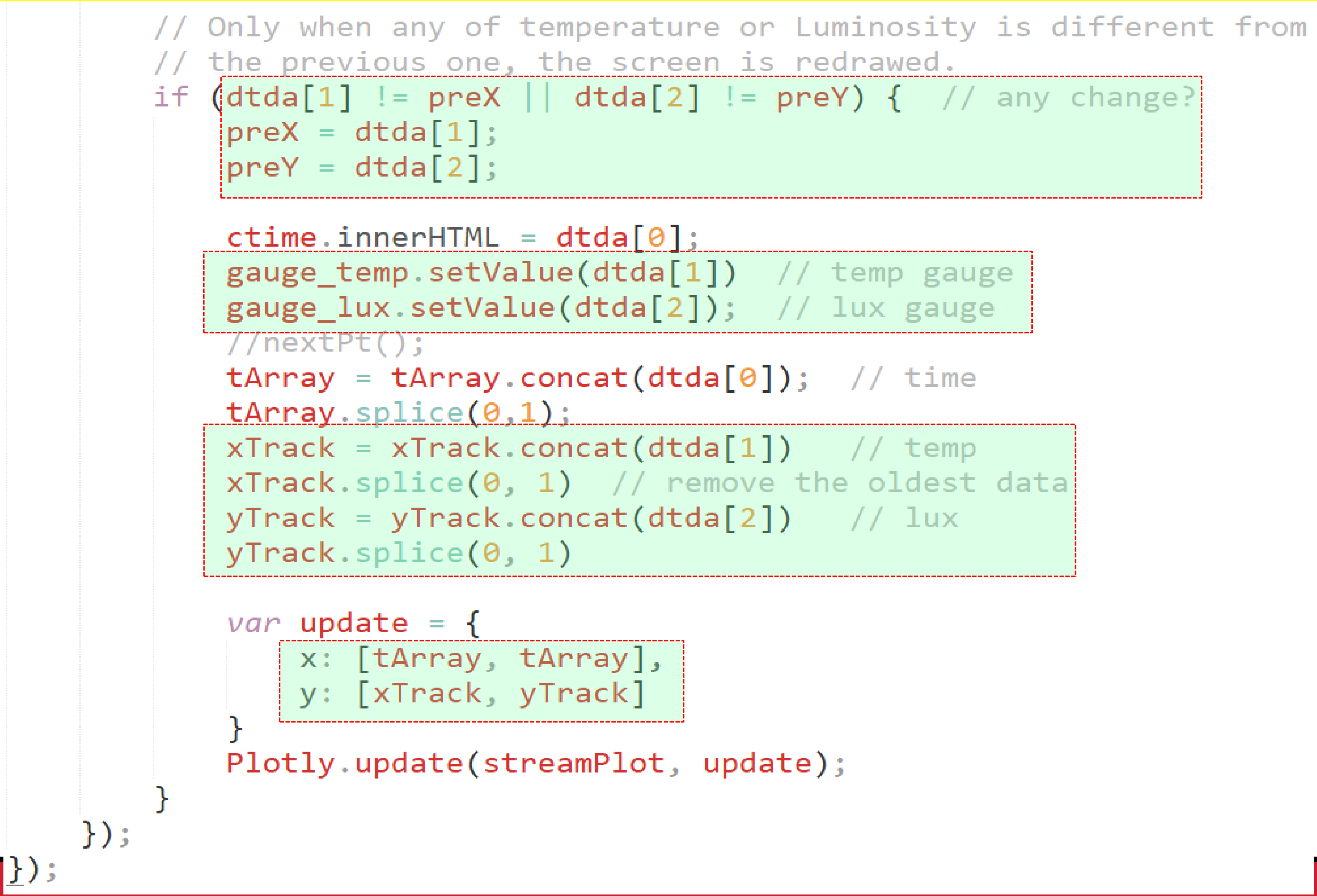
A5.6.3 TMP36 + CdS streaming project

[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

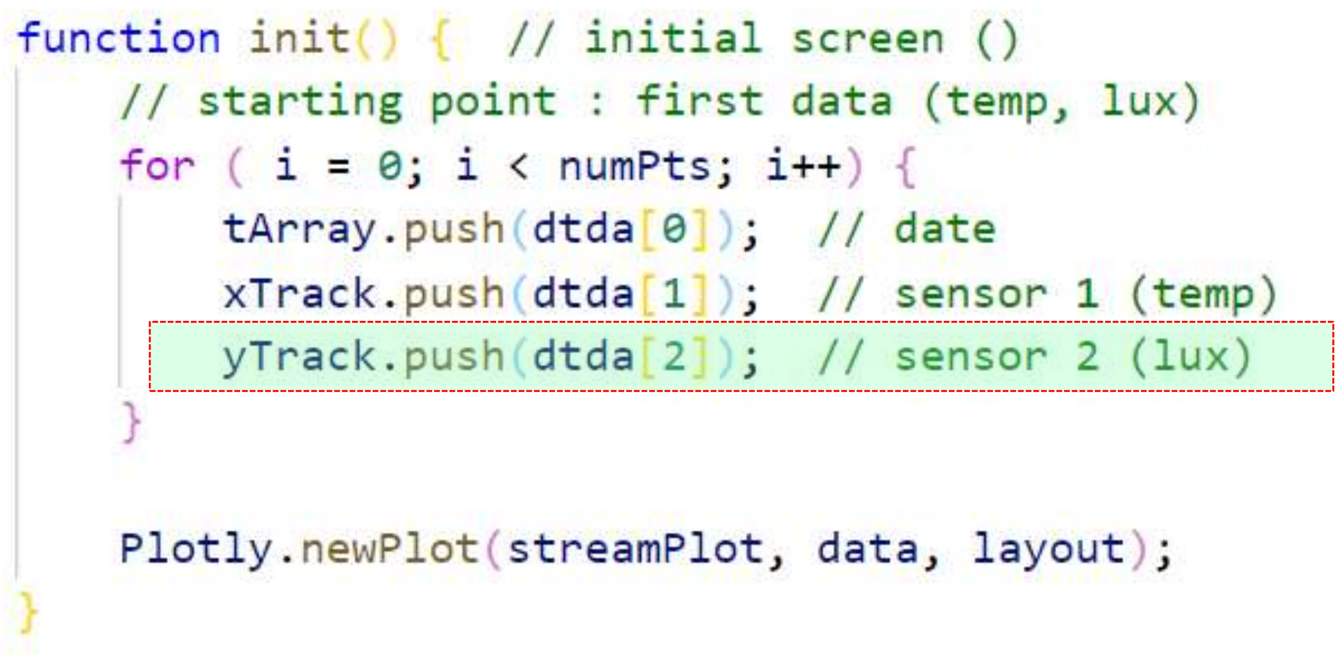
[DIY] Client html : client_cds_tmp36.html (data from multi sensors)



```
            // 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

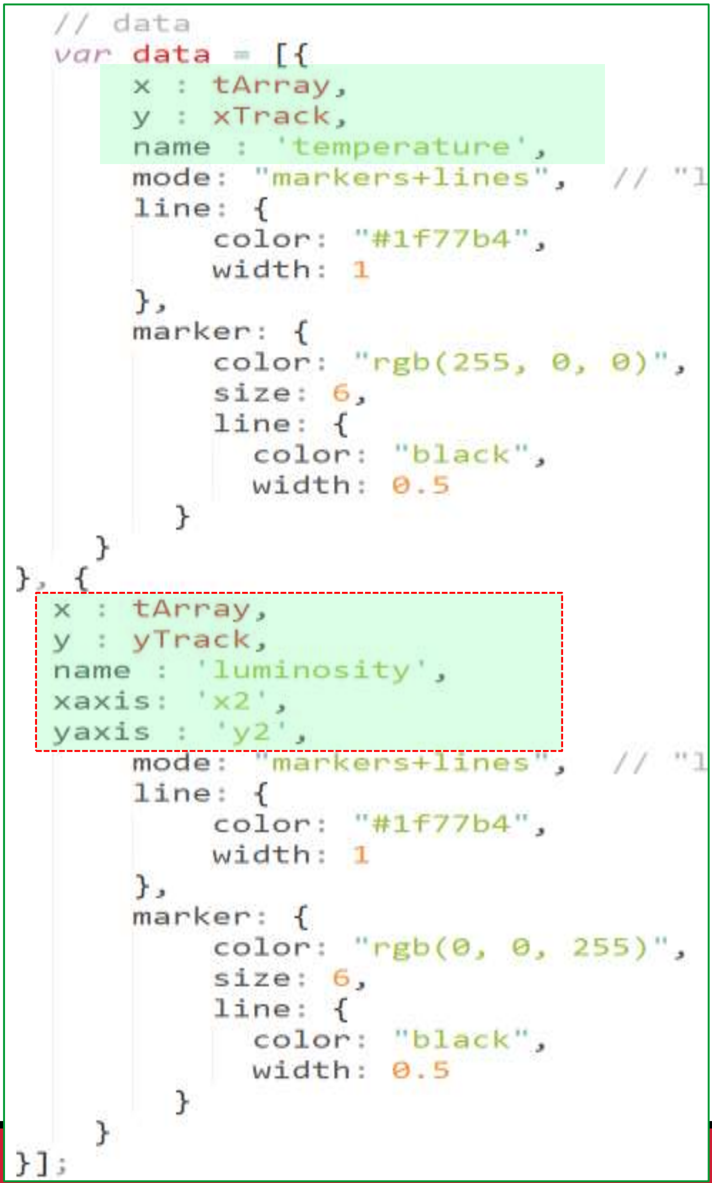
[DIY] Client html : client_cds_tmp36.html (data from multi sensors)



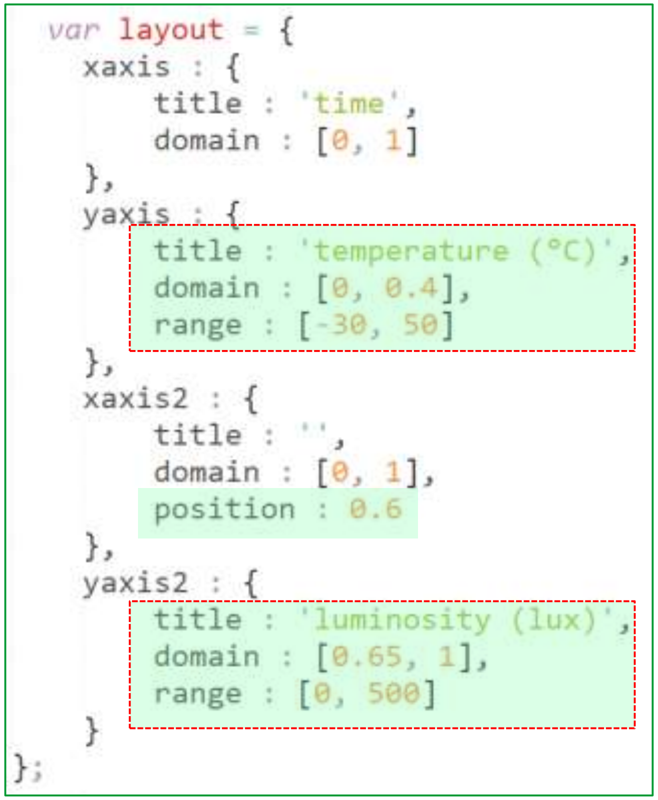
```
function init() {  // initial screen ()
    // starting point : first data (temp, lux)
    for ( i = 0; i < numPts; i++) {
        tArray.push(dtda[0]);  // date
        xTrack.push(dtda[1]);  // sensor 1 (temp)
        yTrack.push(dtda[2]);  // sensor 2 (lux)
    }

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

A5.6.6 TMP36 + CdS streaming project

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



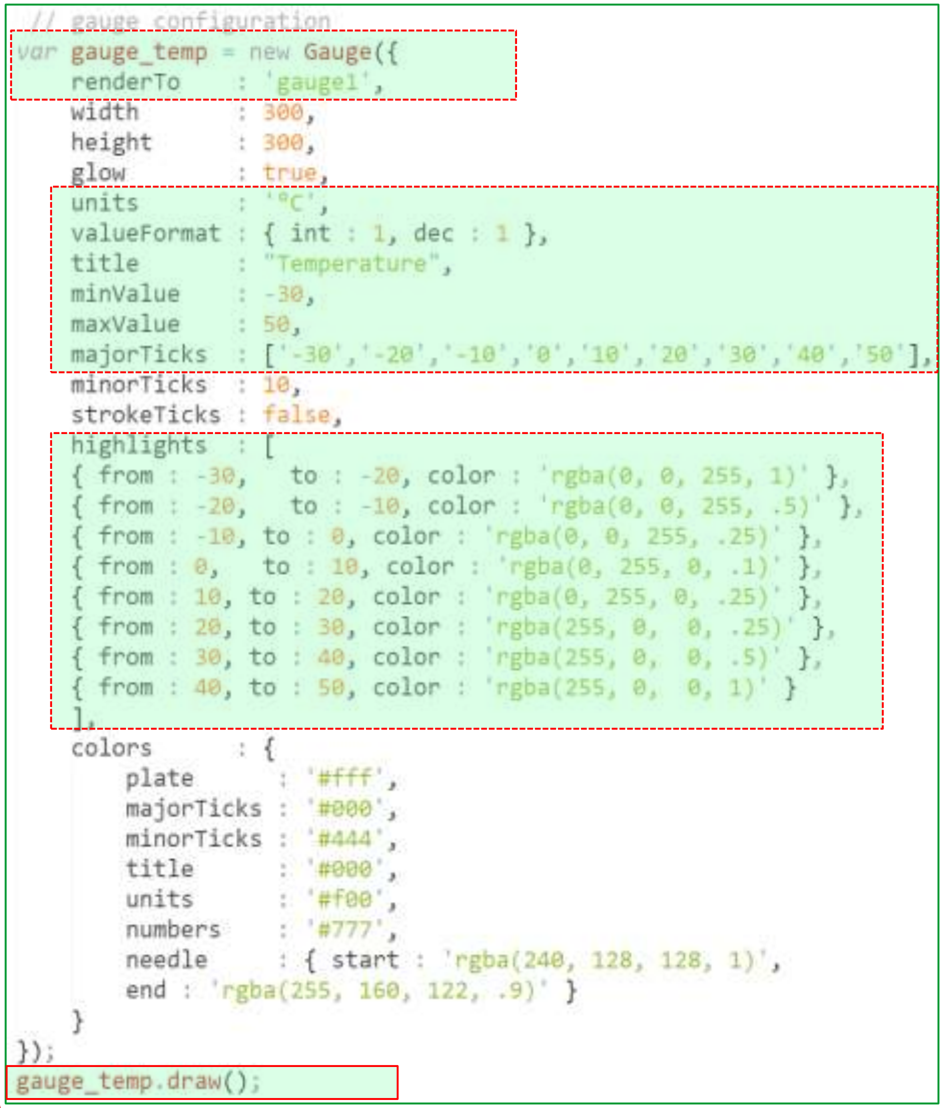
```
// data
var data = [{
        x : tArray,
        y : xTrack,
        name : 'temperature',
        mode: "markers+lines",    // "1
        line: {
            color: "#1f77b4",
            width: 1
        },
        marker: {
            color: "rgb(255, 0, 0)",
            size: 6,
          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



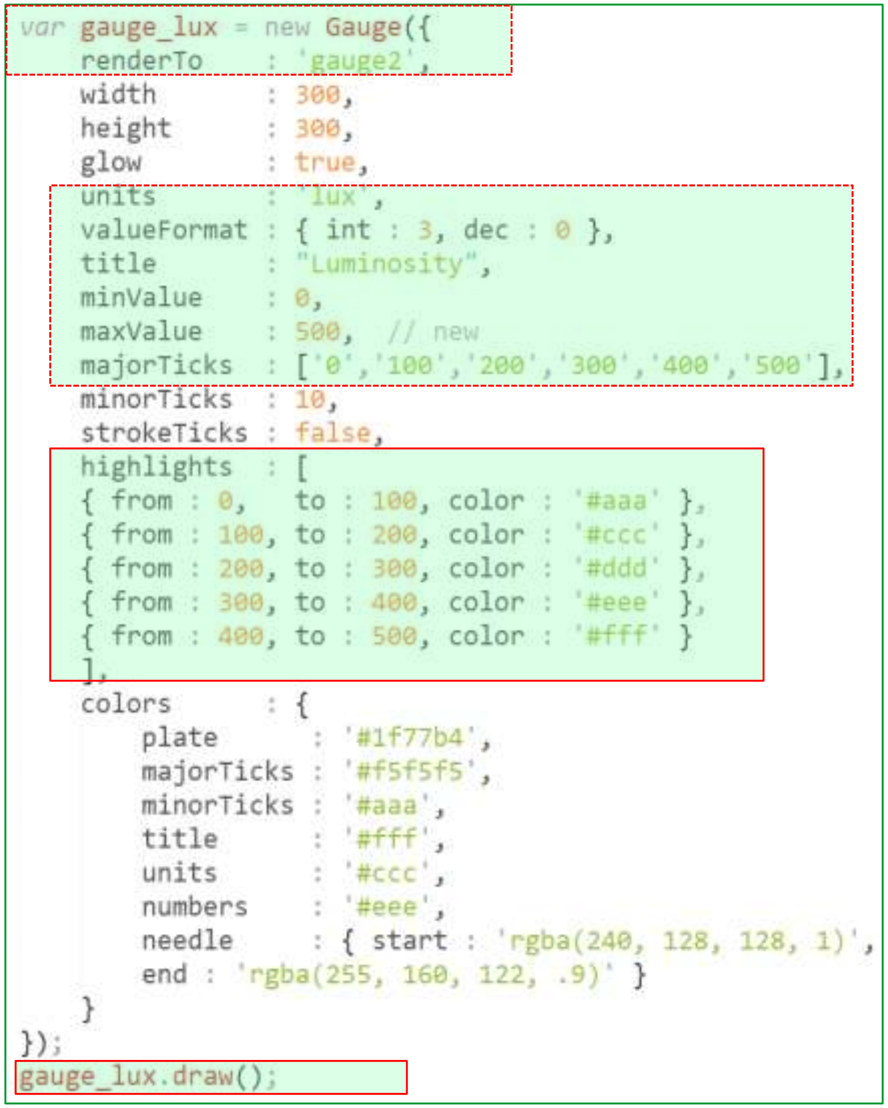
```
var layout = {
    xaxis : {
        title : 'time',
        domain : [0, 1]
    },
    yaxis : {
        title : 'temperature (°C)',
        domain : [0, 0.4],
        range : [-30, 50]
    },
    xaxis2 : {
        title : '',
        domain : [0, 1],
        position : 0.6
    },
    yaxis2 :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



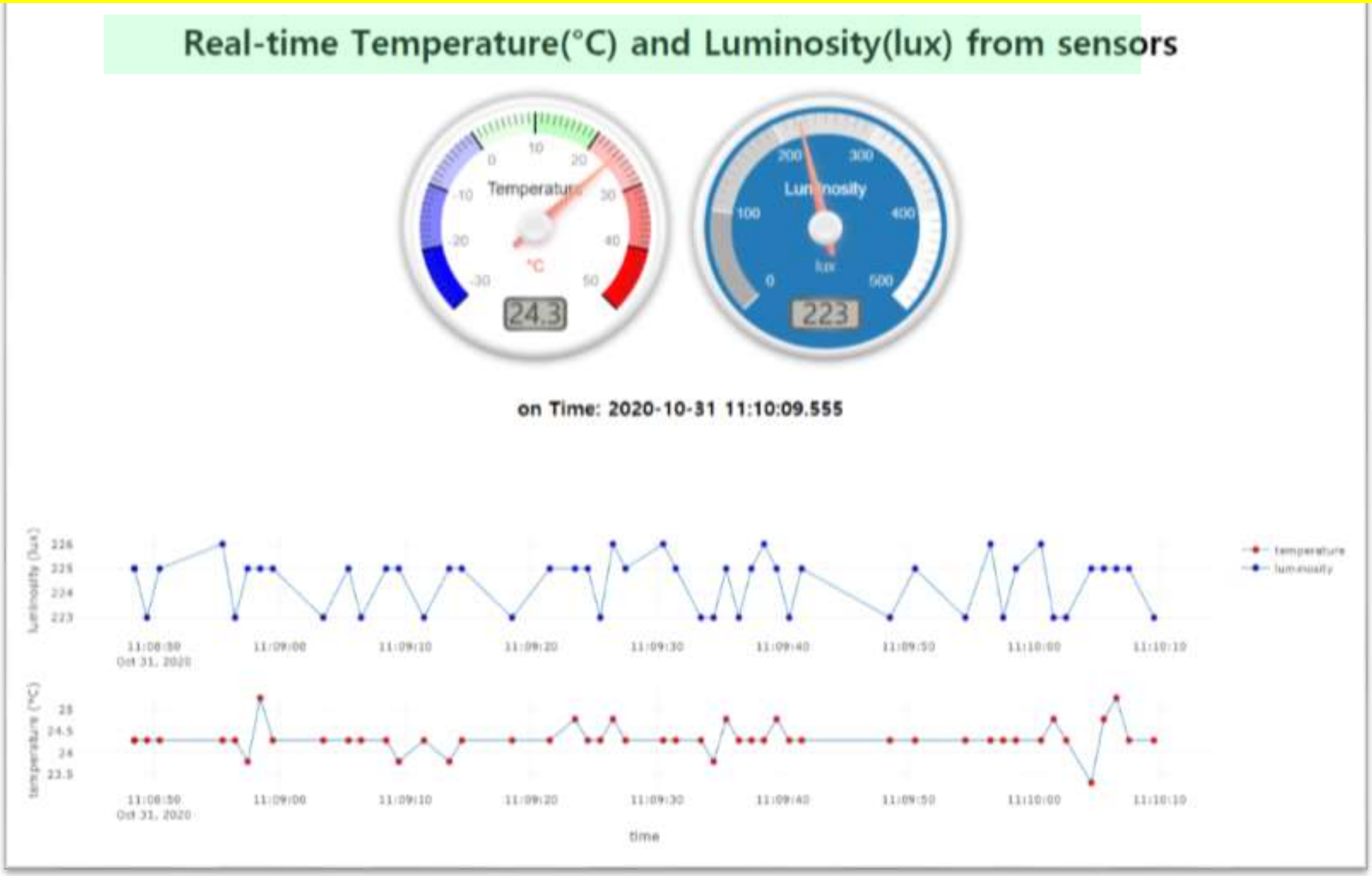
```
 // gauge configuration
var gauge_temp = new Gauge({
    renderTo    : 'gauge1',
    width       : 300,
    height      : 300,
    glow        : true,
    units       : '°C',
    valueFormat : { int : 1, dec : 1 },
    title       : "Temperature",
    minValue    : -30,
    maxValue    : 50,
    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)

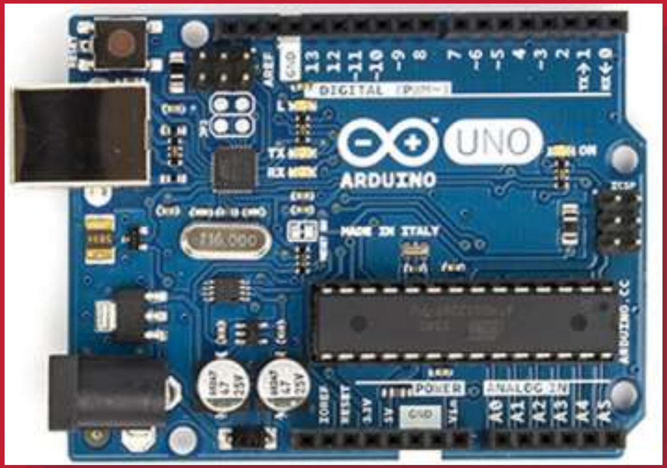


Real-time Temperature(°C) and Luminosity(lux) from sensors

on Time: 2020-10-31 11:10:09.555

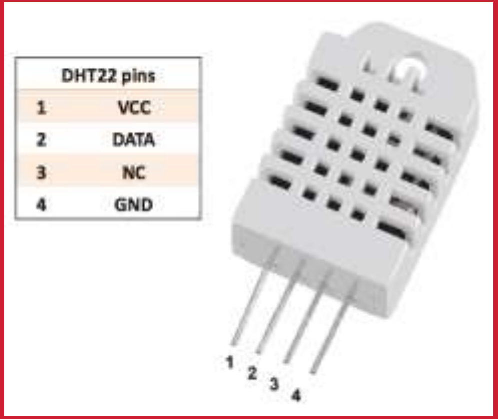
Gauge 디자인 변경한 후에, AAnn_DS_cds_tmp36.png 로 저장

CdS + DHT22 + plotly.js Node project

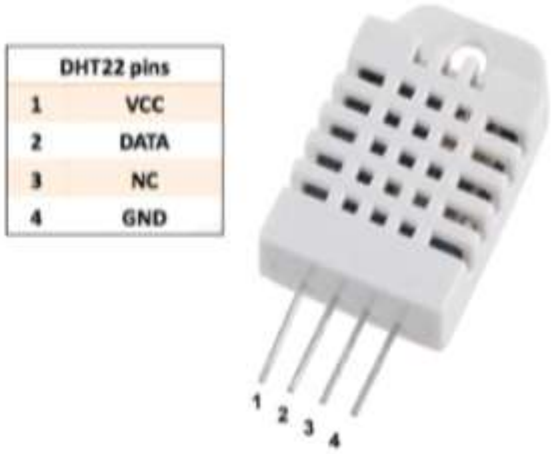


Multi-sensors

DHT22 + CdS



DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



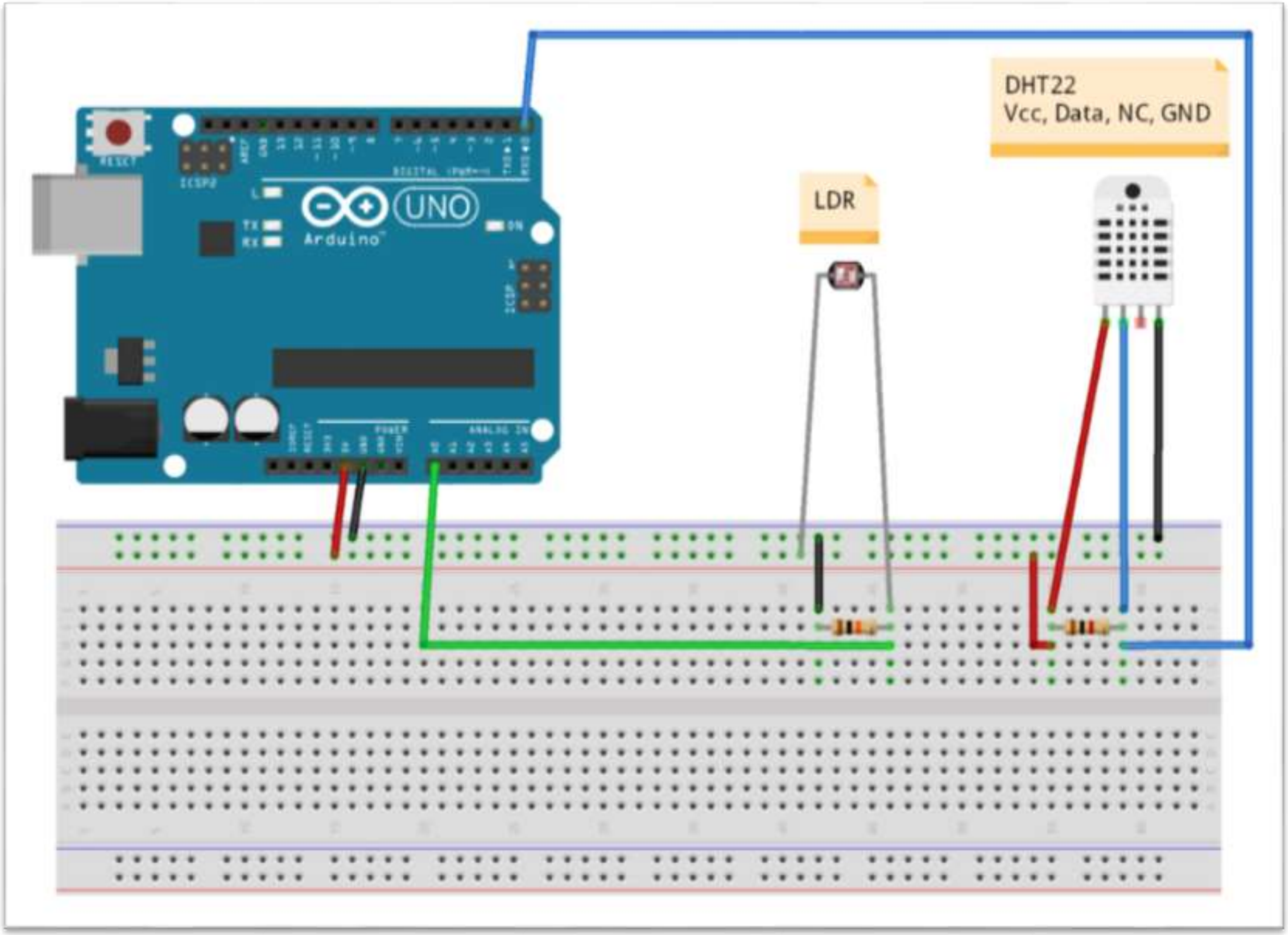
DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

그림 8-7 DHT22 pin 구조

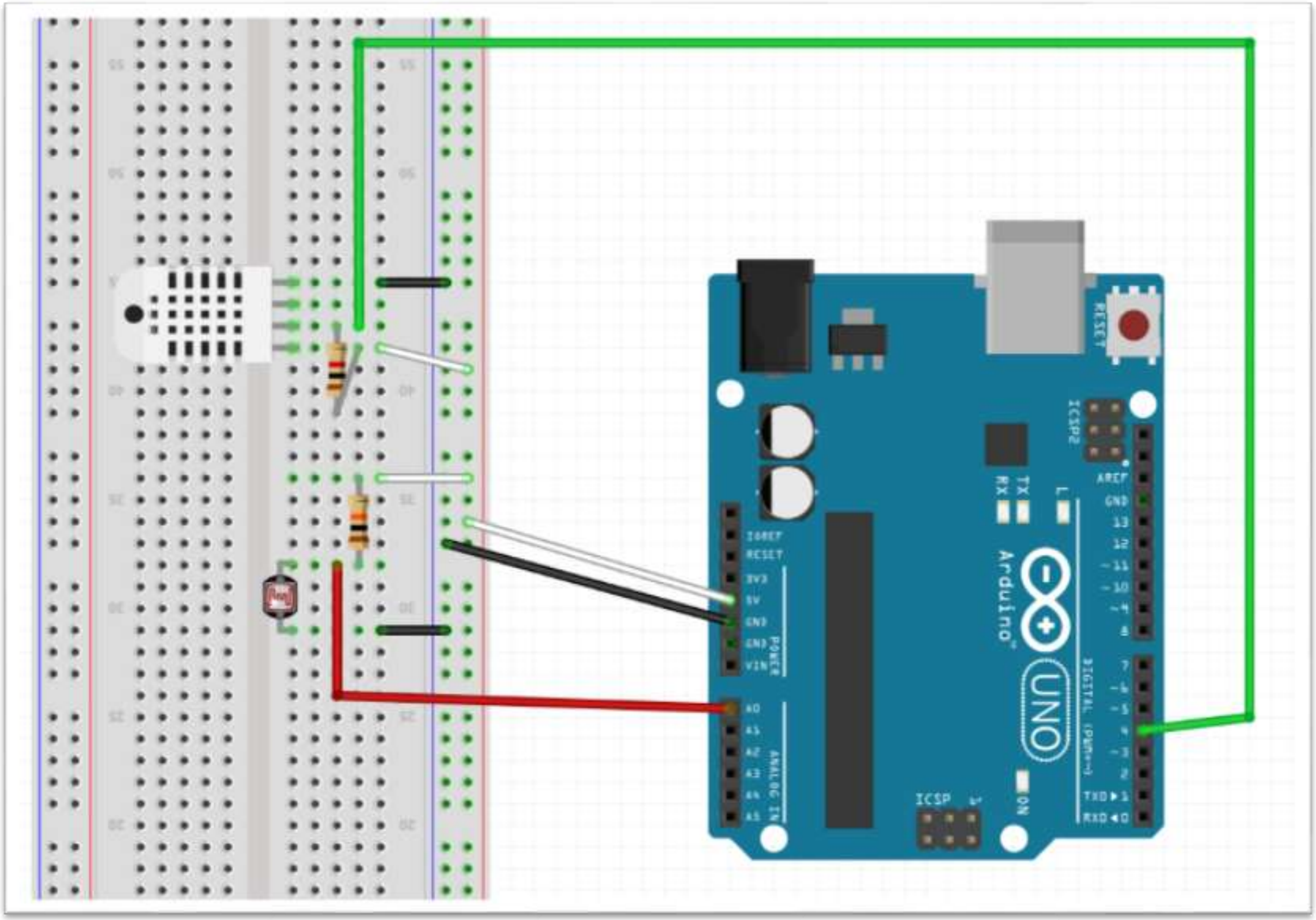
- 3 ~ 5V power and I/O
- 2.5mA max current
- [0-100%] humidity readings with 2-5% accuracy
- [-40 to 80°C] temperature readings ±0.5°C accuracy
- 0.5 Hz sampling rate

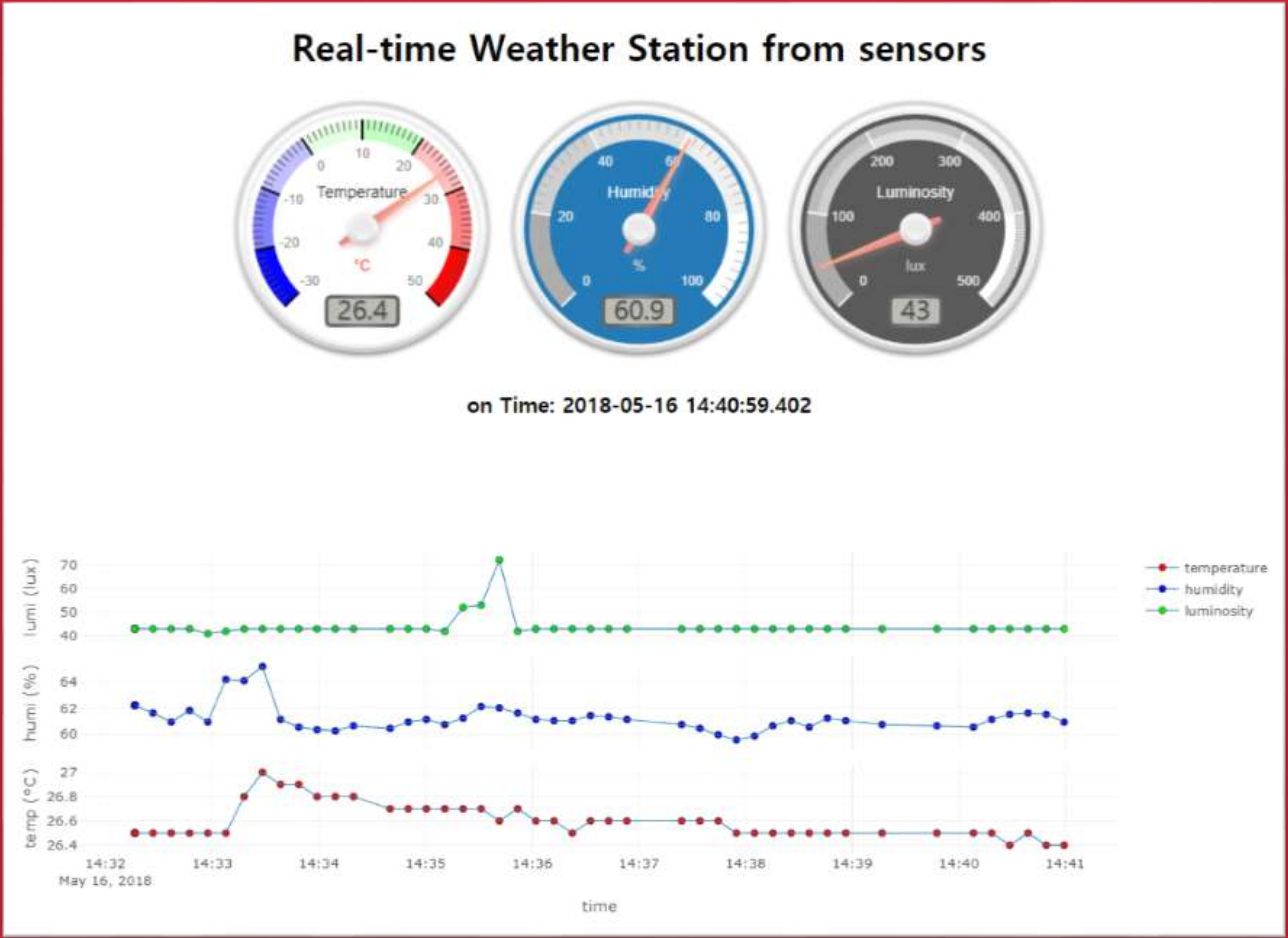
https://learn.adafruit.com/dht/overview

A5.7 DHT22 + CdS streaming project



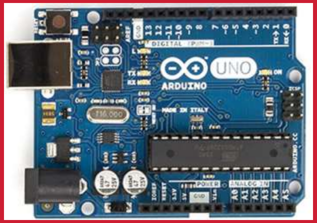
A5.7.1 DHT22 + CdS circuit





Real-time Weather Station from sensors

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



[Practice]

- [wk10]
  - RT Data Visualization with node.js
  - Usage of gauge.js
  - Complete your plotly-node project
  - Upload folder: aax-nn-rpt08
  - Use repo “aax-nn” in github

wk10 : Practice : aax-nn-rpt08

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

제출폴더명 : **aax-nn-rpt08**

- 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

Lecture materials

- **References & good sites**

  ✓ http://www.arduino.cc Arduino Homepage

  ✓ http://www.nodejs.org/ko Node.js

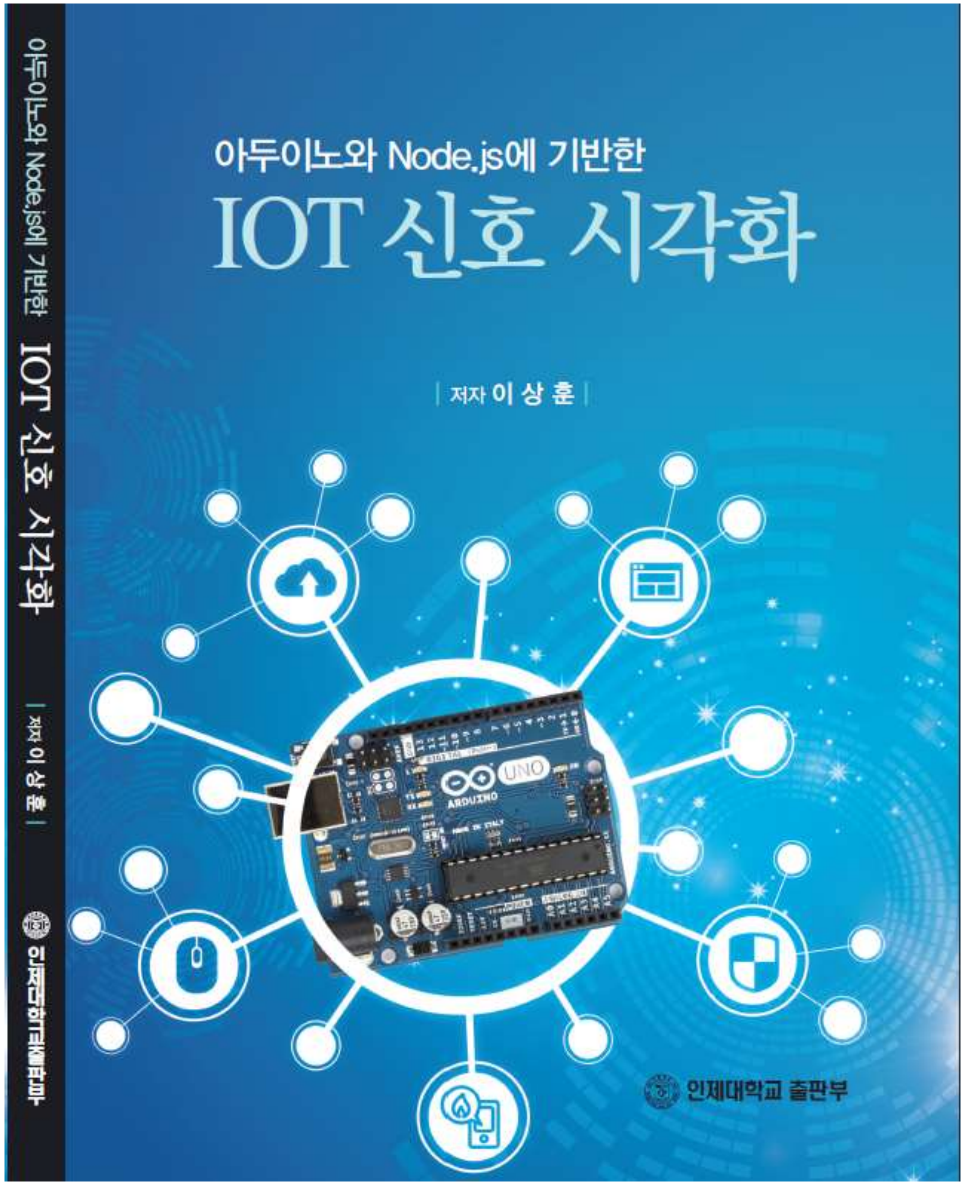
  ✓ https://plot.ly/ plotly

  ✓ https://www.mongodb.com/ MongoDB

  ✓ http://www.w3schools.com By w3schools

  ✓ http://www.github.com GitHub

주교재 및 참고도서

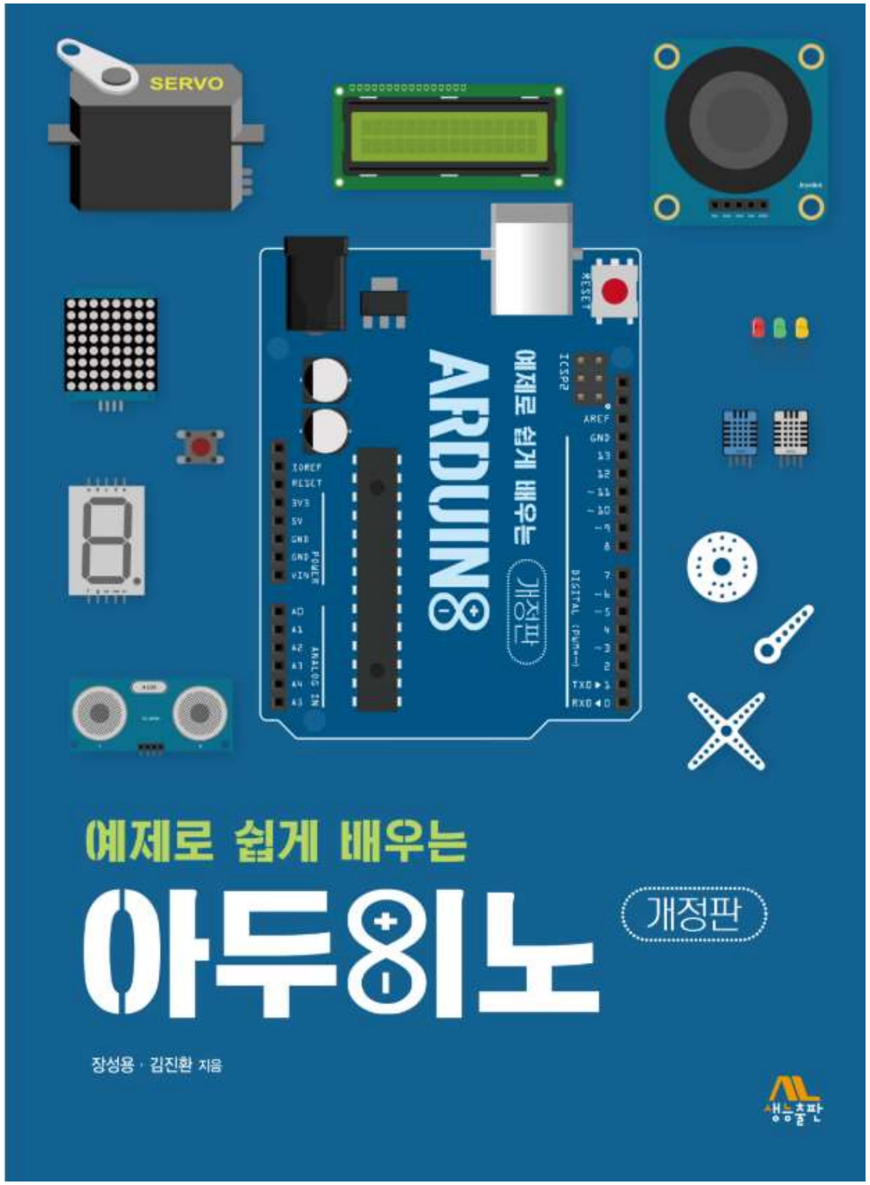


아두이노와 Node.js에 기반한

IOT 신호 시각화

| 저자 이 상 훈 |

인제대학교 출판부

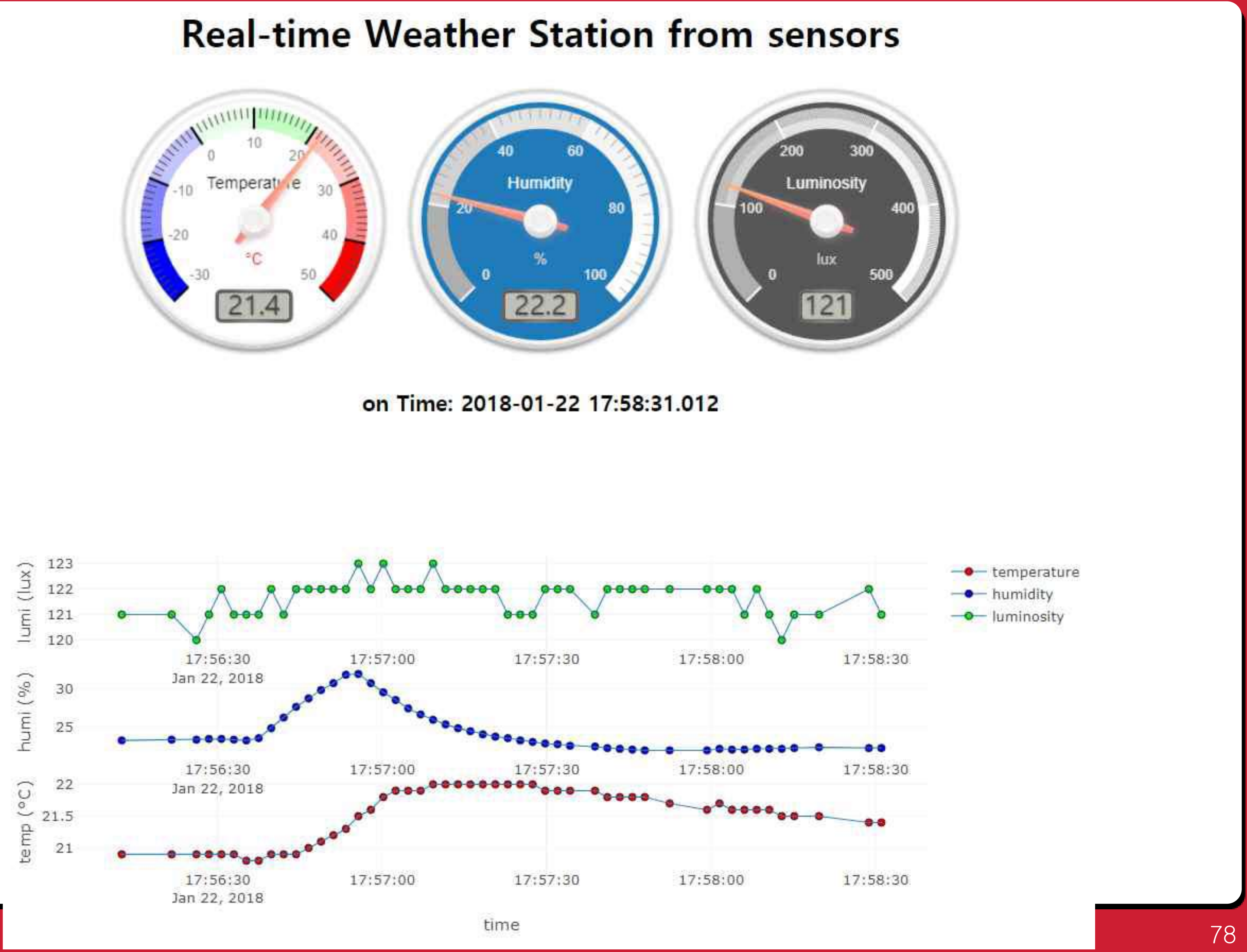


예제로 쉽게 배우는

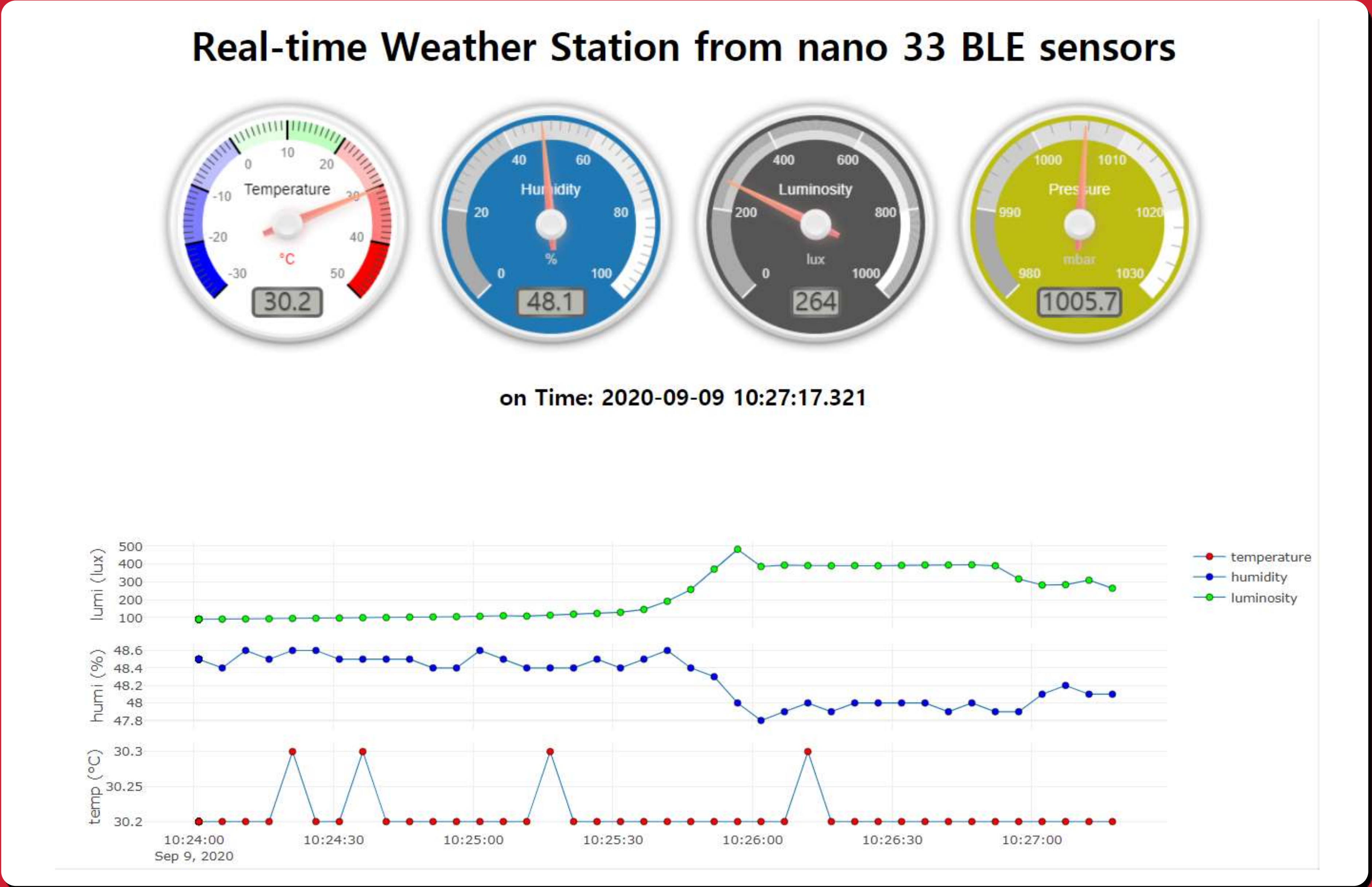
아두이노 개정판

장성용 · 김진환 지음

Target of this class

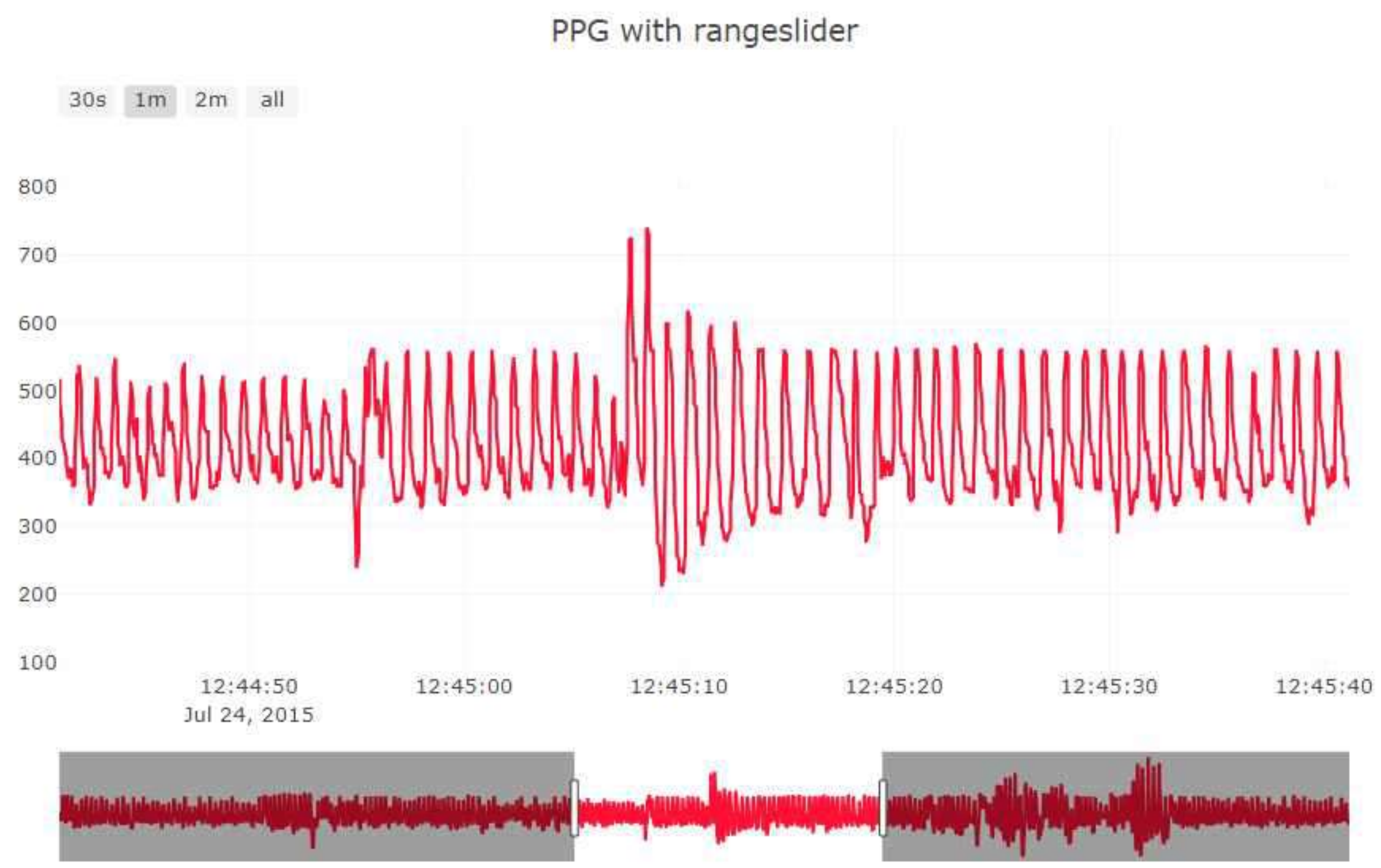


Target of this class


Real-time Weather Station from nano 33 BLE sensors

on Time: 2020-09-09 10:27:17.321

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