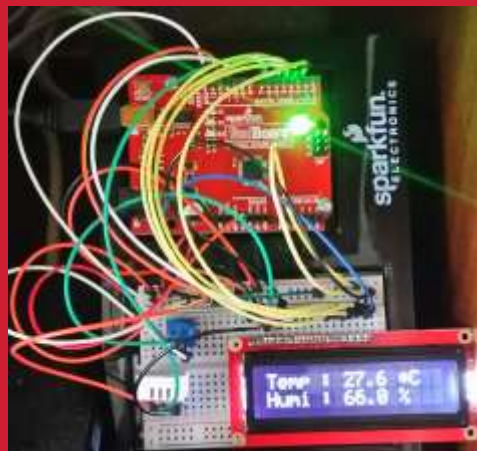




Arduino-IOT

[wk11]

Arduino + Node Data visualization II



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



Comsi, INJE University

2nd semester, 2018

Email : chaos21c@gmail.com

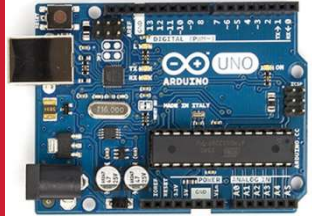


My ID

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

이근재

AA11



[Review]

◆ [wk10]

- Charts by plotly
- Complete your plotly chart project
- Upload file name : AAnn_Rpt07.zip

◆ [Target of this week]

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

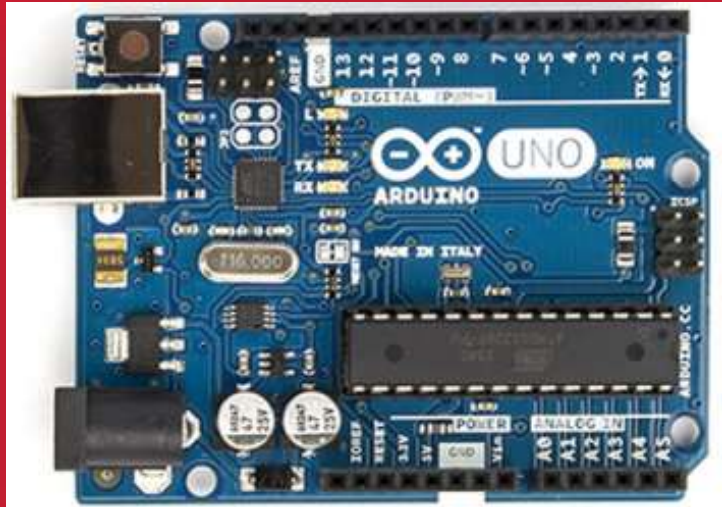
제출파일명 : **AAnn_Rpt07.zip**

- 압축할 파일들

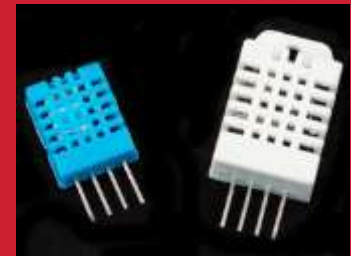
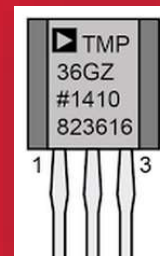
- ① **AAnn_multi_signals_node.png**
- ② **AAnn_Chart_Layout.png**
- ③ **AAnn_Axis_Title.png**
- ④ **AAnn_Line_Dash_Dot.png**
- ⑤ **AAnn_lux_Time_Series.png**
- ⑥ **AAnn_lux_Rangeslider.png**

Email : chaos21c@gmail.com

[제목 : id, 이름 (수정)]



Arduino & Node.js





IOT: HSC

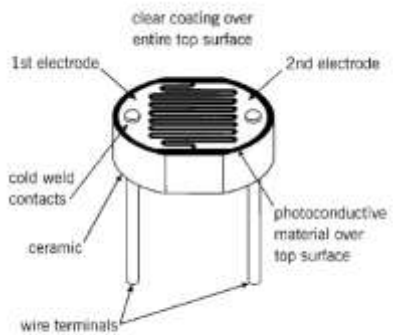
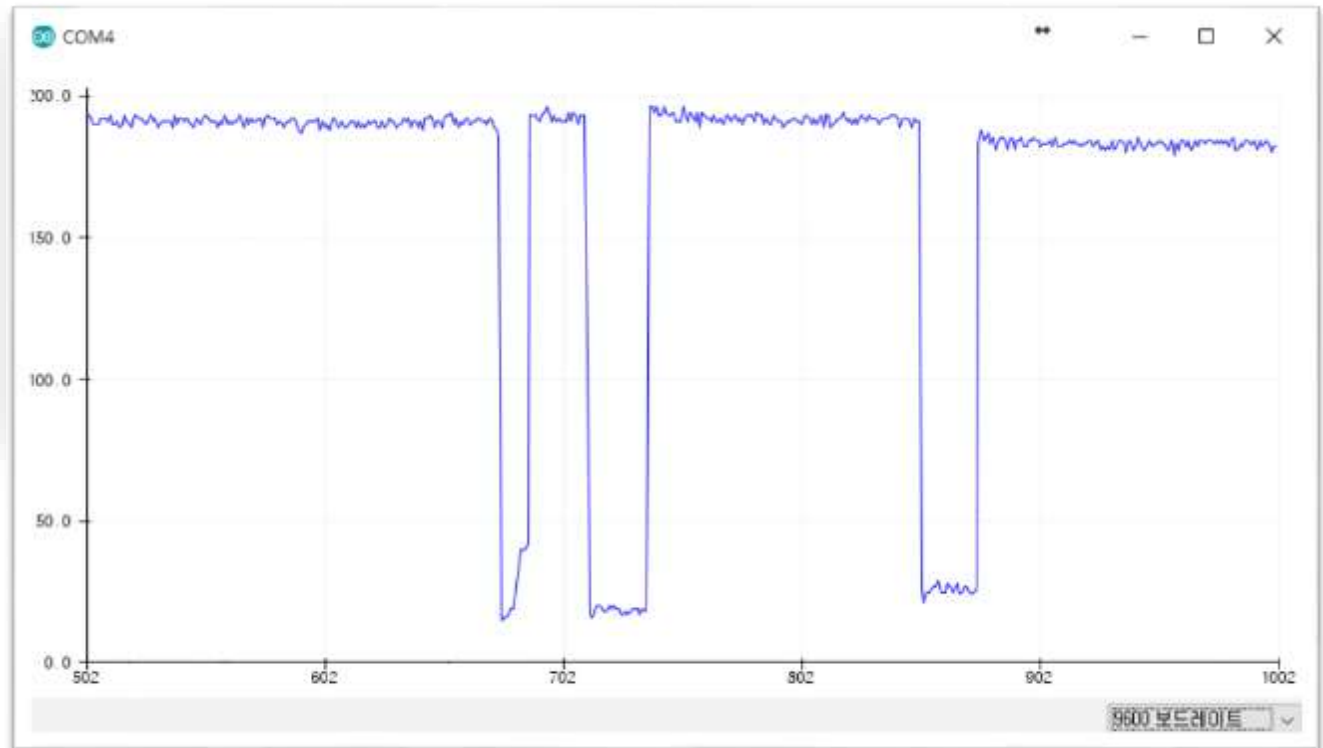
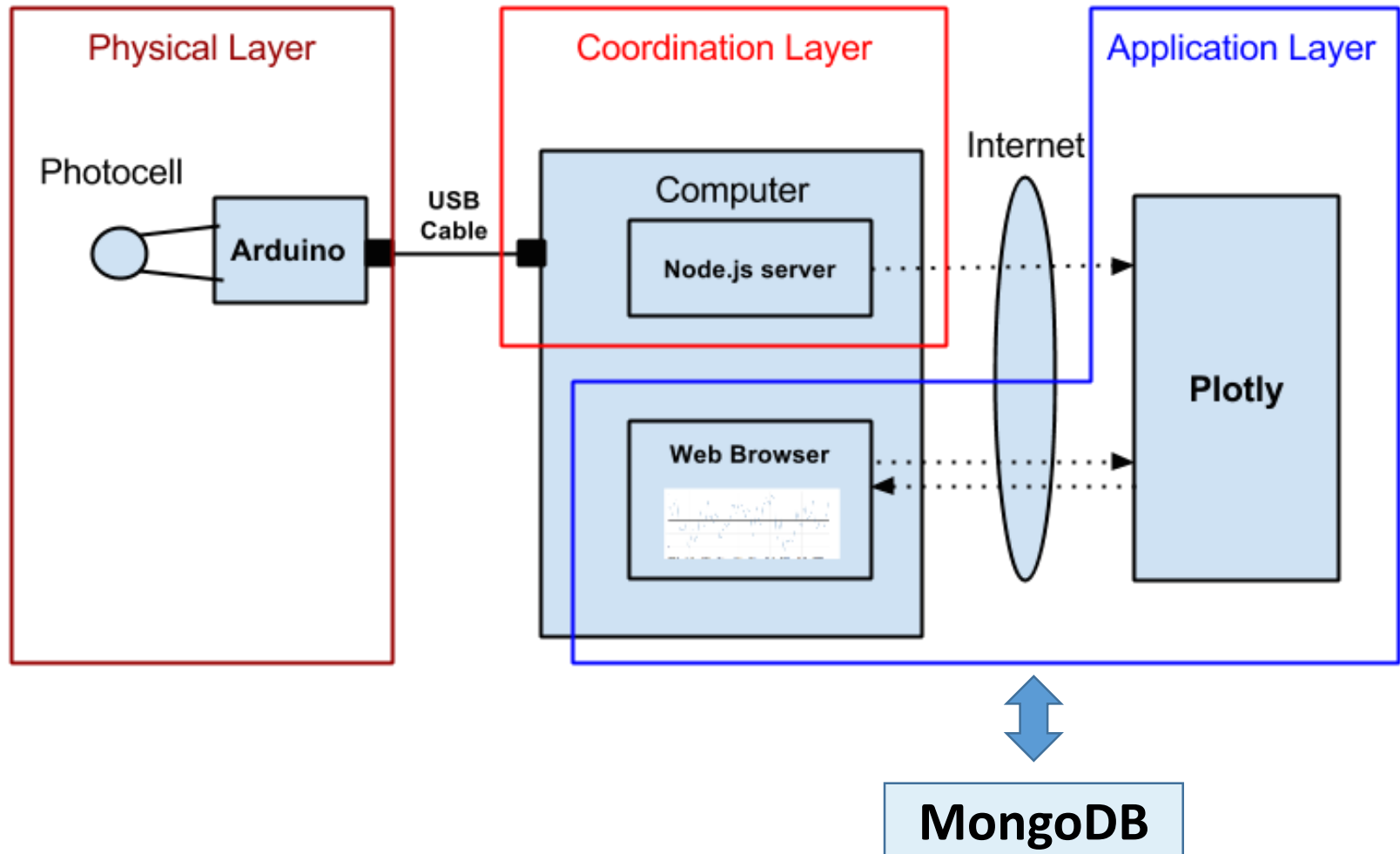


Figure 3
Typical Construction of a Plastic Coated Photocell

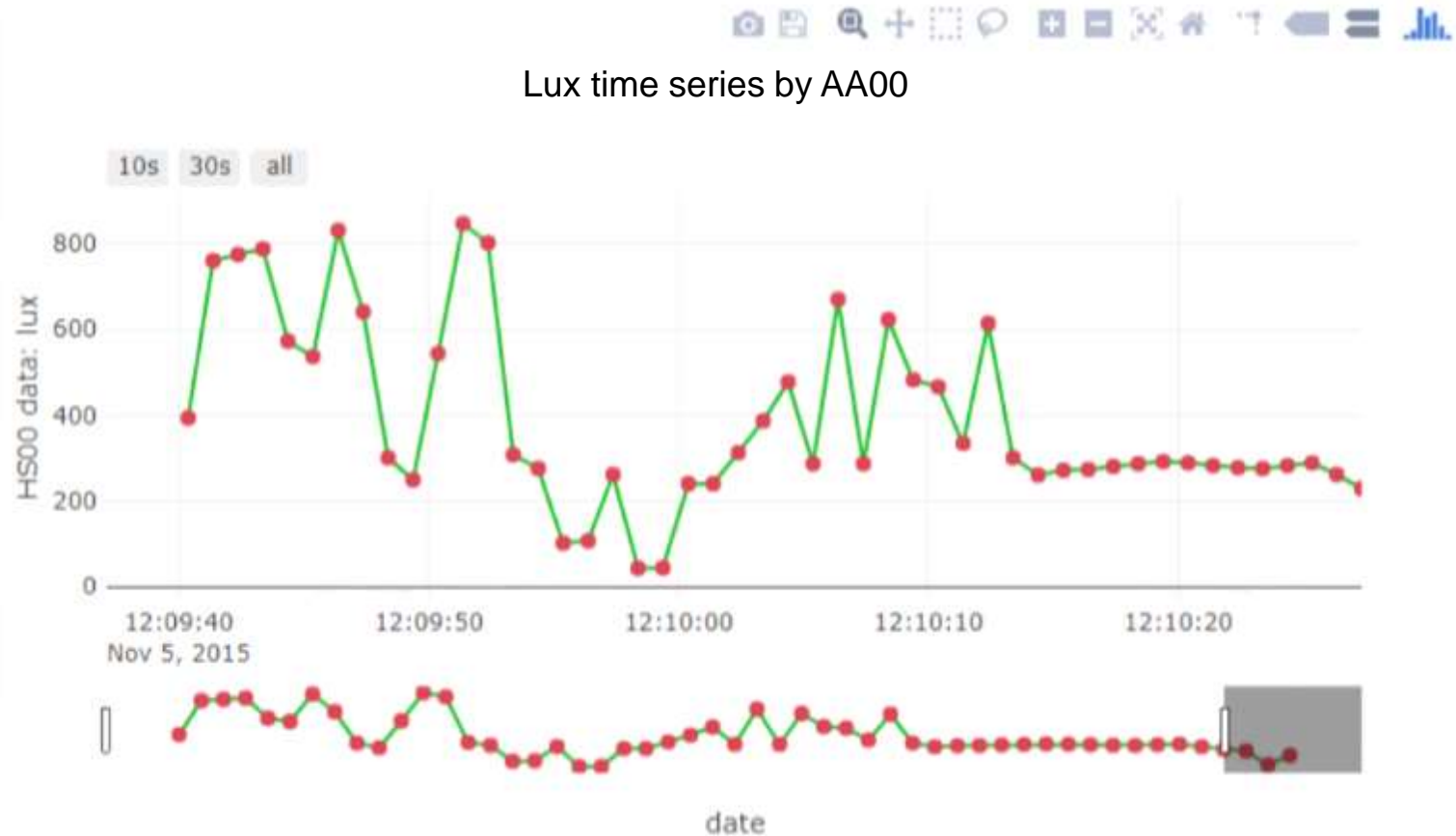


Layout [H S C]



Arduino data + plotly

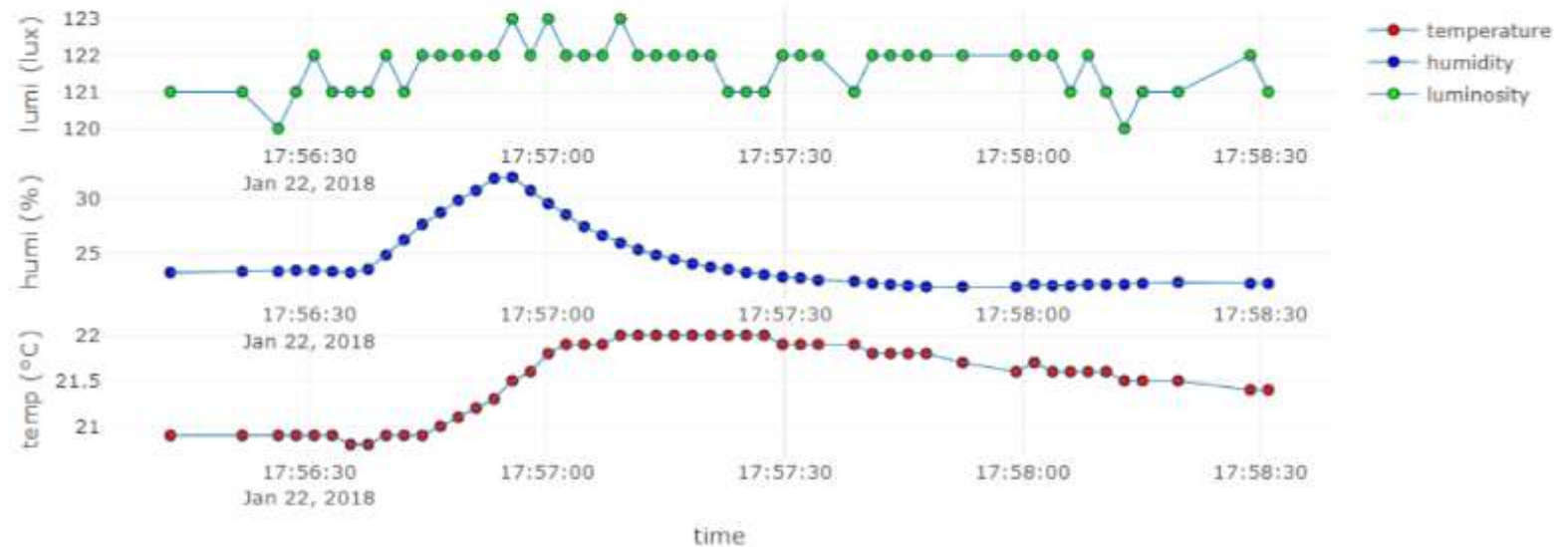
Time series by AA00

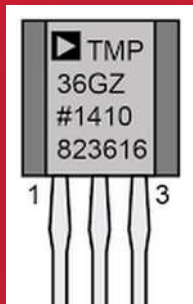
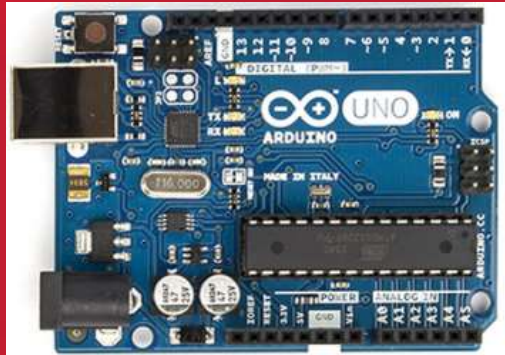


Real-time Weather Station from sensors



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





Data visualization using **play.ly**





A5. Introduction to visualization

System (Arduino, sDevice, ...)



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



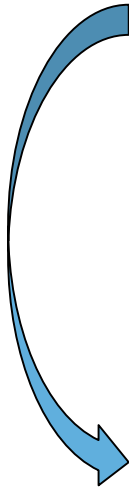
Visualization & monitoring



Data storing & mining



Service





A5.1 Introduction to data visualization

아두이노 센서 회로

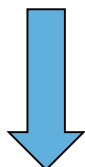


직렬모니터/플로터 모니터링



LCD 모니터링

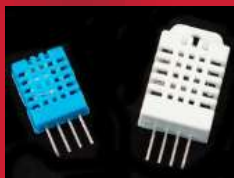
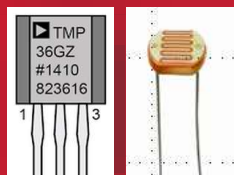
Node.js



Plotly.js



웹 모니터링

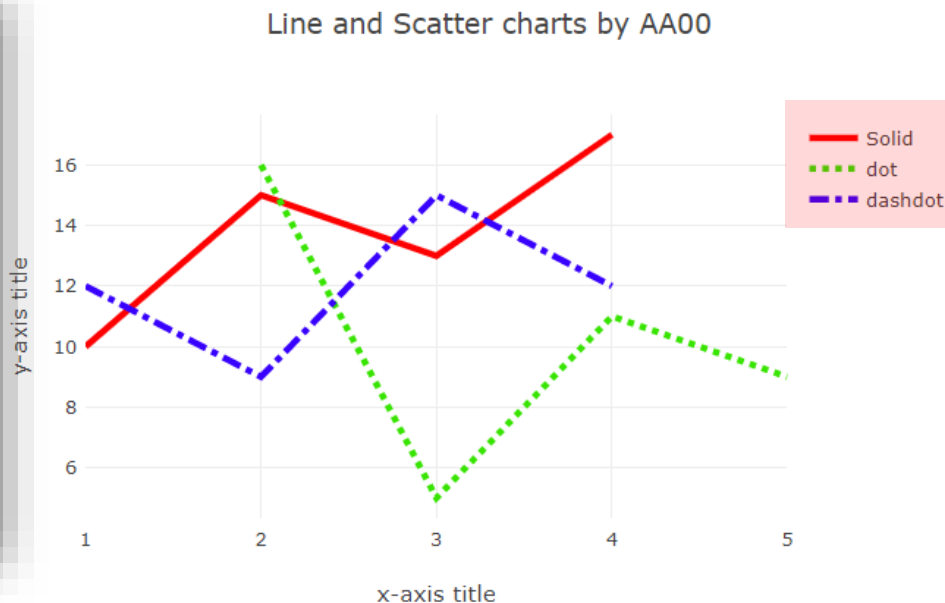


[3.5] Line & scatter plot with dash and dot

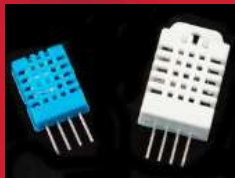
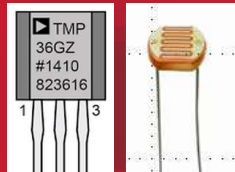
```
var trace1 = {
  x: [1, 2, 3, 4],
  y: [10, 15, 13, 17],
  mode: 'lines',
  name: 'Solid',
  line: {
    color: 'rgb(255, 0, 0)',
    dash: 'solid',
    width: 4
  }
};
```

```
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
  }
};
```

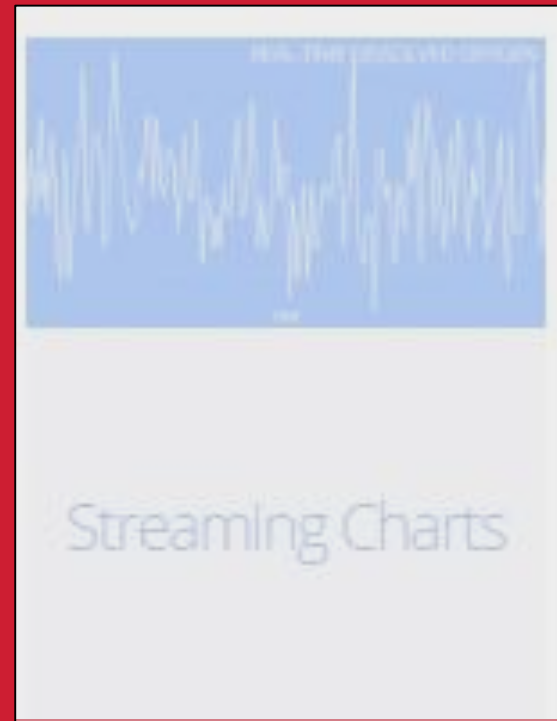
```
var trace3 = {
  x: [1, 2, 3, 4],
  y: [12, 9, 15, 12],
  mode: 'lines',
  name: 'dashdot',
  line: {
    color: 'rgb(55, 0, 255)',
    dash: 'dashdot',
    width: 4
  }
};
```



AAnn_Line_Dash_Dot.png



Data visualization using **plotly.js**





Project: Time series with Rangelslider

[Project-DIY] AAnn_lux_Rangelslider.html

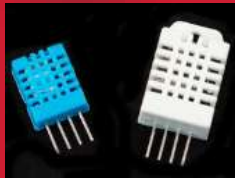
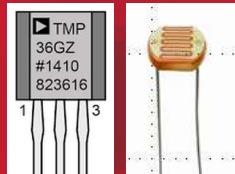
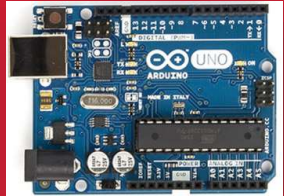


AAnn_lux_Rangelslider.png



Time series with Rangeslider

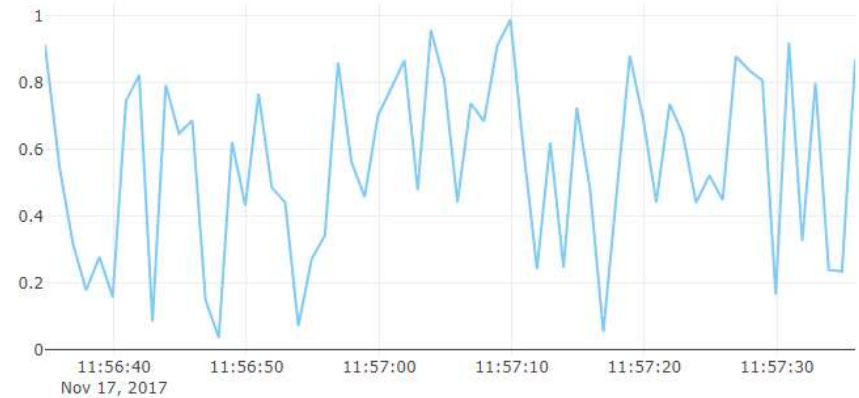
```
var layout = {  
  title: 'lux time series by AA00',  
  width: 750, height: 500,  
  margin: {  
    l: 50,  
    r: 50,  
    b: 100,  
    t: 100,  
    pad: 4  
  },  
  xaxis: {  
    title: 'date',  
    autorange: true,  
    range: ['2015-11-05 12:09:40.383', '2015-11-05 12:10:30.413'], 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



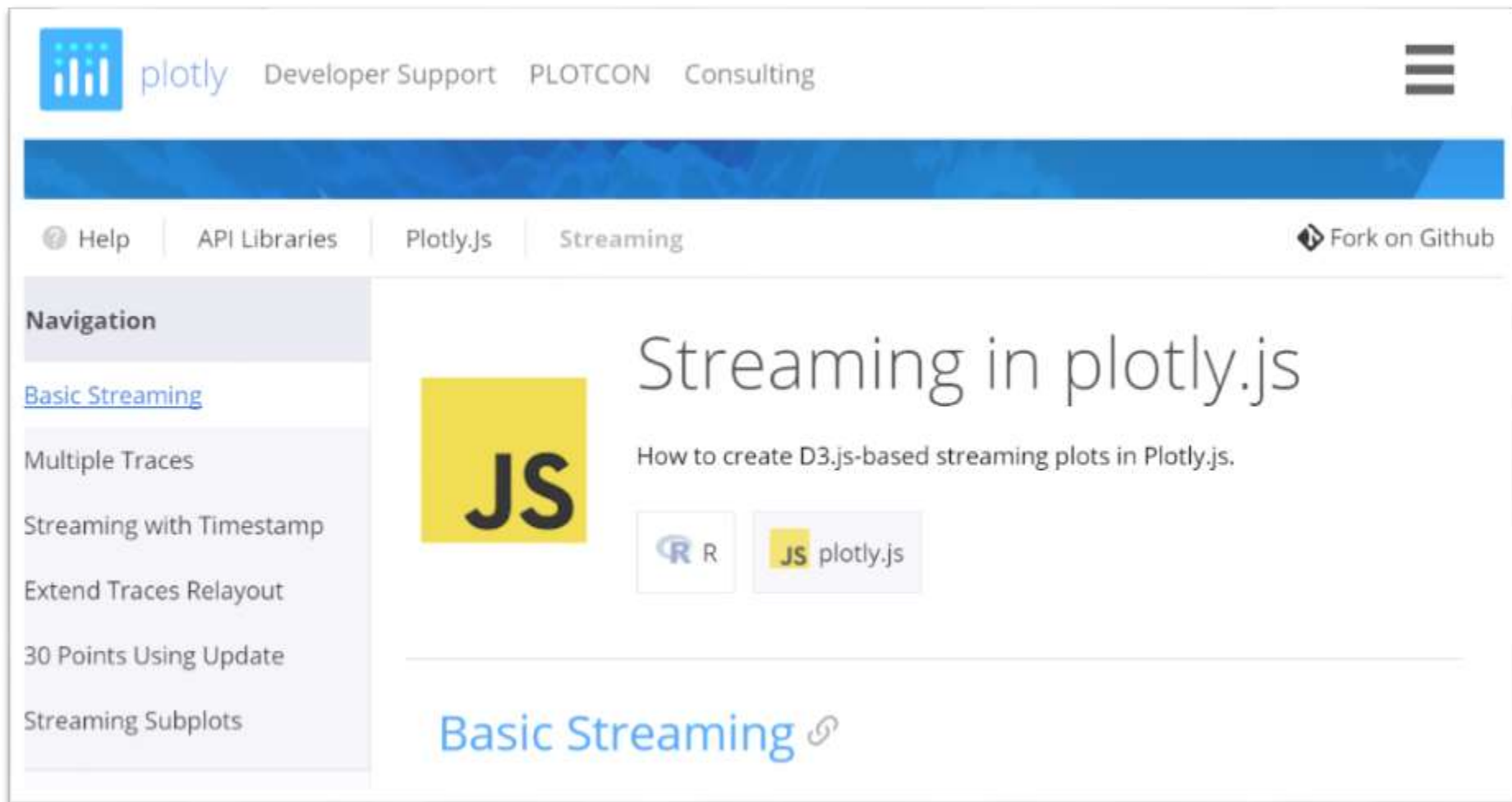
Streaming data with timestamp





A5.4 plotly.js: Streaming data

Plot.ly > Streaming



The screenshot shows the Plotly.js website's 'Streaming' page. The header includes the Plotly logo, navigation links for 'Developer Support', 'PLOTCON', and 'Consulting', and a hamburger menu icon. Below the header is a blue banner. The main navigation bar contains links for 'Help', 'API Libraries', 'Plotly.js', and 'Streaming', along with a 'Fork on Github' button. A left sidebar titled 'Navigation' lists various topics, with 'Basic Streaming' highlighted. The main content area features a large yellow 'JS' logo, the title 'Streaming in plotly.js', and a subtitle 'How to create D3.js-based streaming plots in Plotly.js.'. Below this are two buttons: one for 'R' and one for 'JS plotly.js'. At the bottom, there is a link for 'Basic Streaming' with an external link icon.

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 JS plotly.js

[Basic 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;

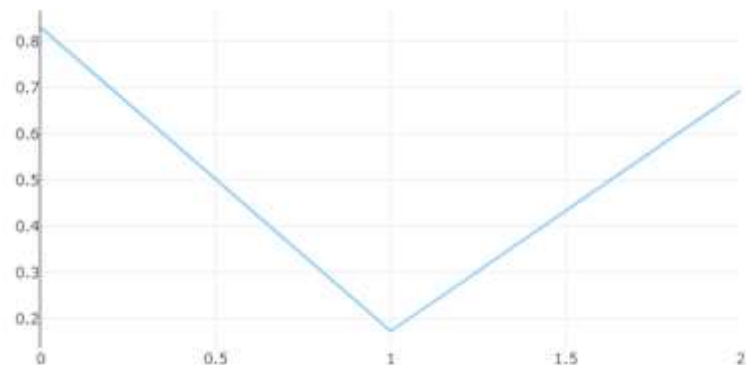
  // var interval = setInterval(function() {

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

  //   if(cnt == 50) clearInterval(interval);
  // }, 1000);

</script>
```

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)

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

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

```
<script>
```

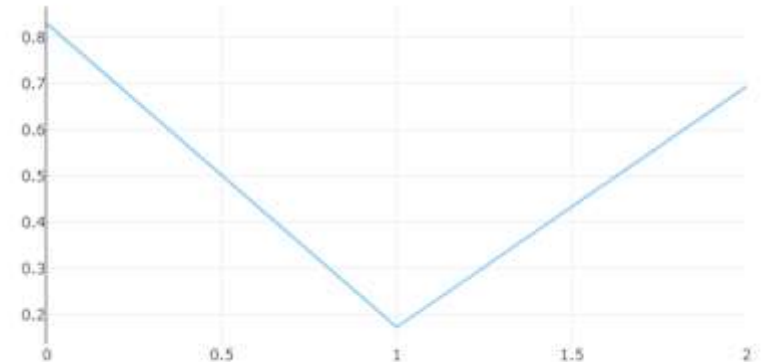
```
function rand() {  
  return Math.random();  
}
```

```
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>
```

Hello streaming!





A5.4.2.2 plotly.js: Streaming data

[1.2] Basic streaming

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

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

```
<script>
```

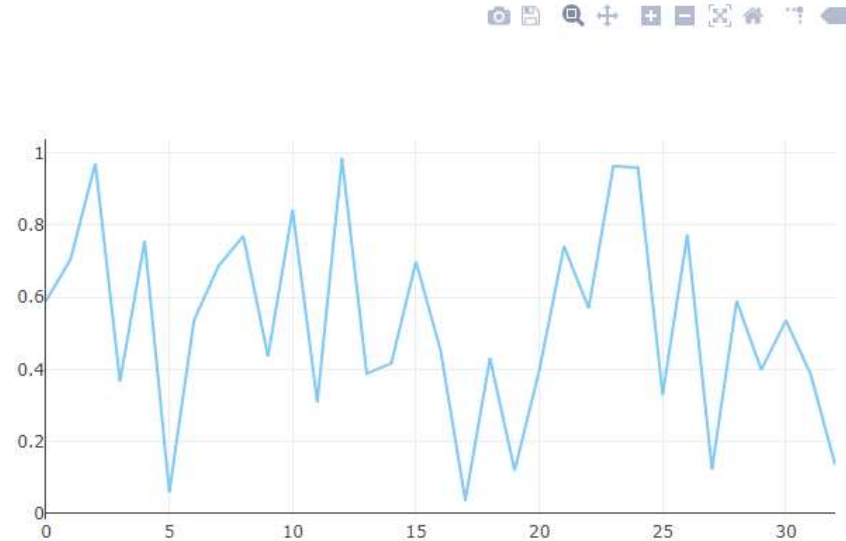
```
function rand() {  
  return Math.random();  
}
```

```
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>
```

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;
```

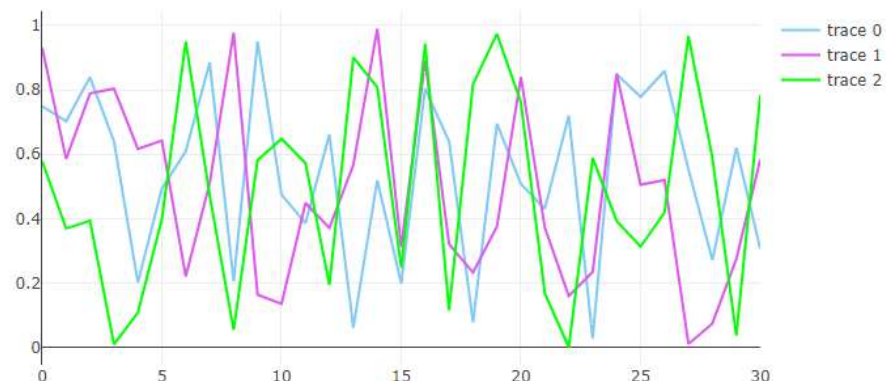
```
var interval = setInterval(function() {
```

```
    Plotly.extendTraces('graph', {
```

```
        y: [[rand()], [rand()], [rand()]]  
    }, [0, 1, 2])
```

```
    cnt++;
```

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



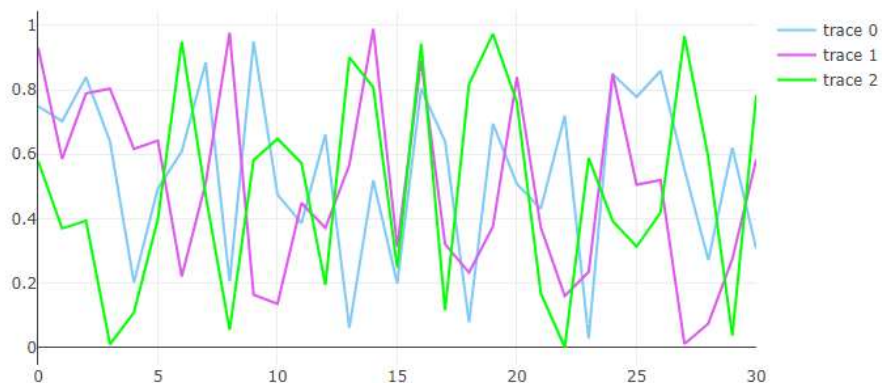


A5.4.3.2 plotly.js: Streaming data

[2.2] Streaming multiple traces (new code)

```
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);
```

Timestamp data streaming



[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);
```

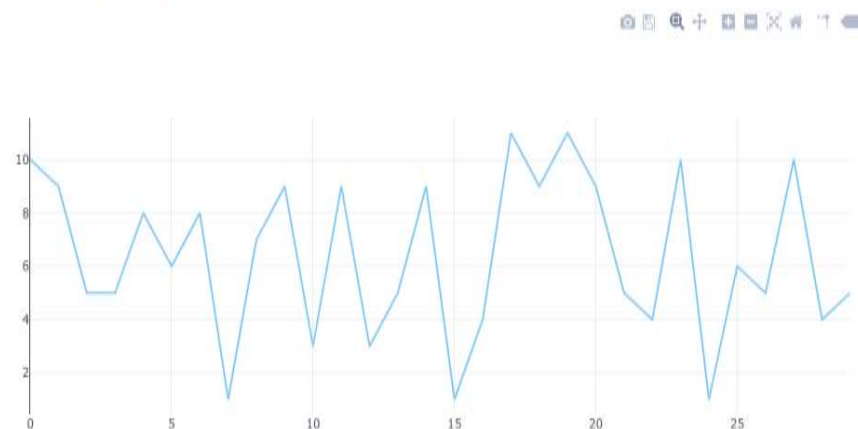
```
var cnt = 0;
var interval = setInterval(function() {

  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);
```

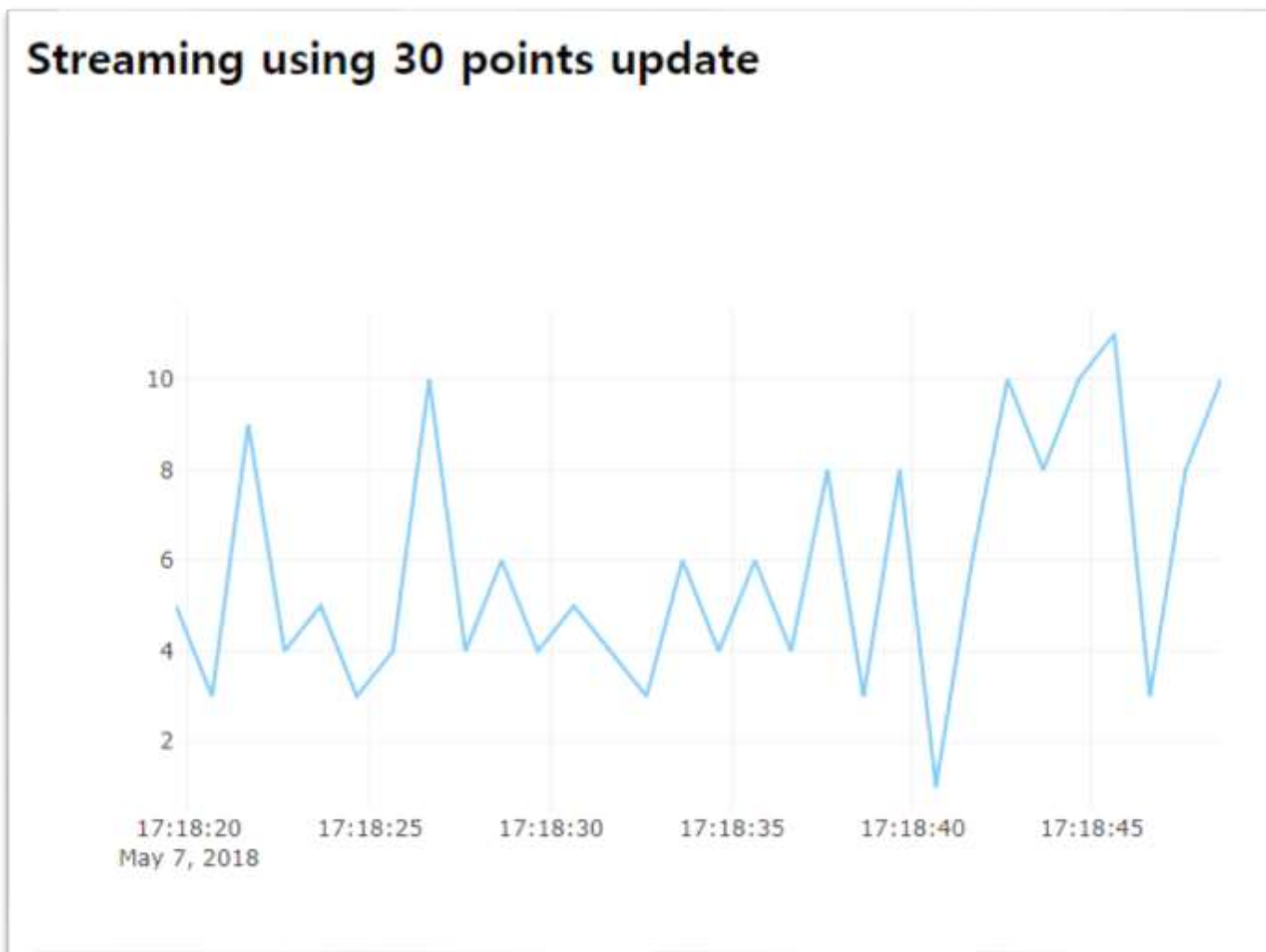
Streaming using 30 points update





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

```
<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

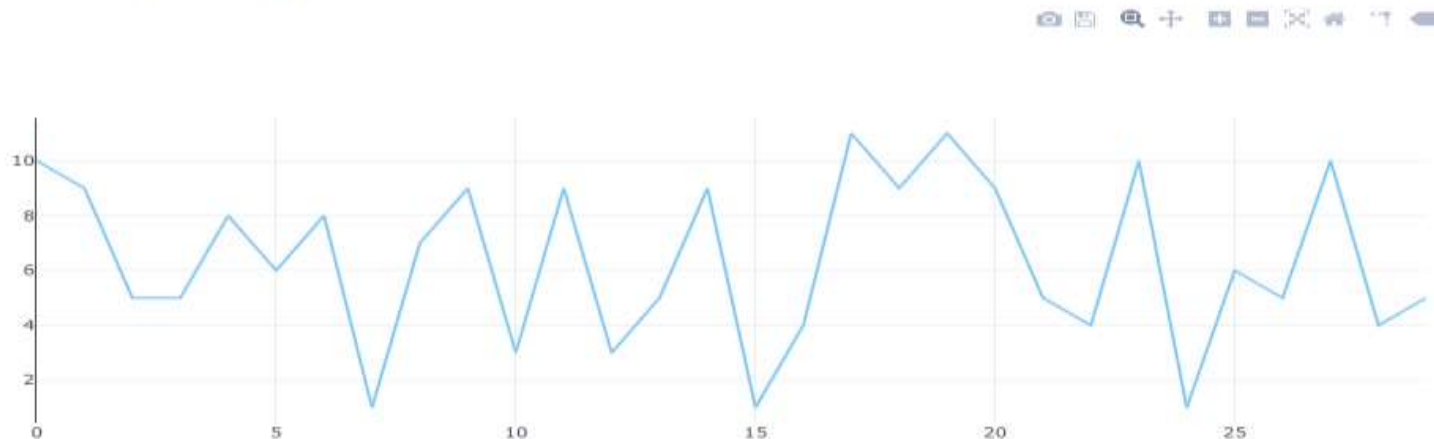
  var update = {
    x: [timeArray],
    y: [newArray]
  }

  Plotly.update('graph', update)

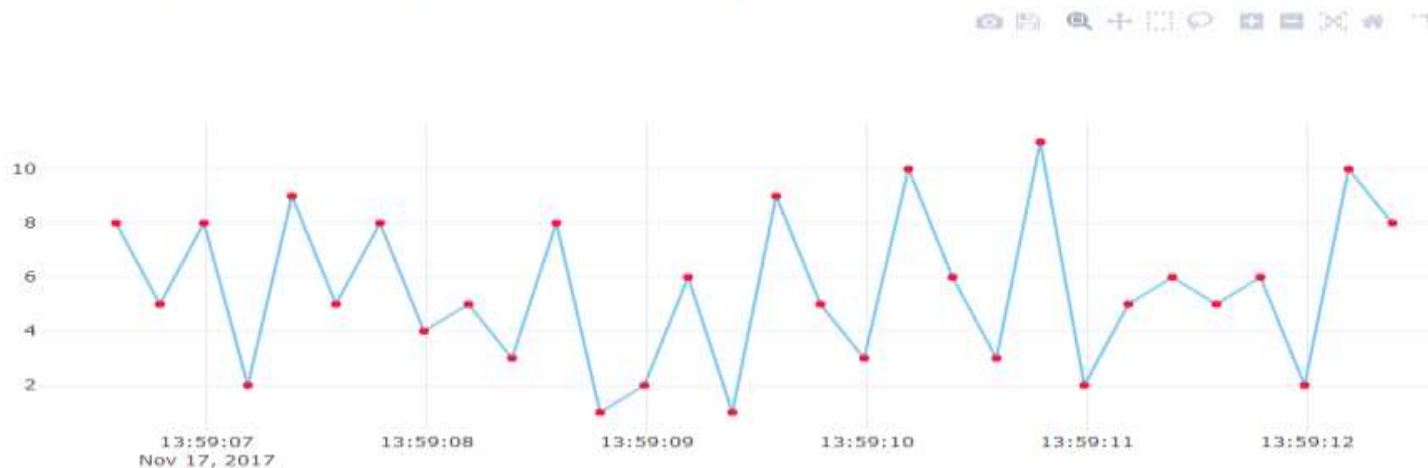
  if(cnt === 100) clearInterval(interval);
}, 1000);
```

[DIY] Streaming time series using 30 points update

Streaming using 30 points update



Streaming using 30 points update with timestamp



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);
```

[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'
  };
</script>
```

```
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);
```

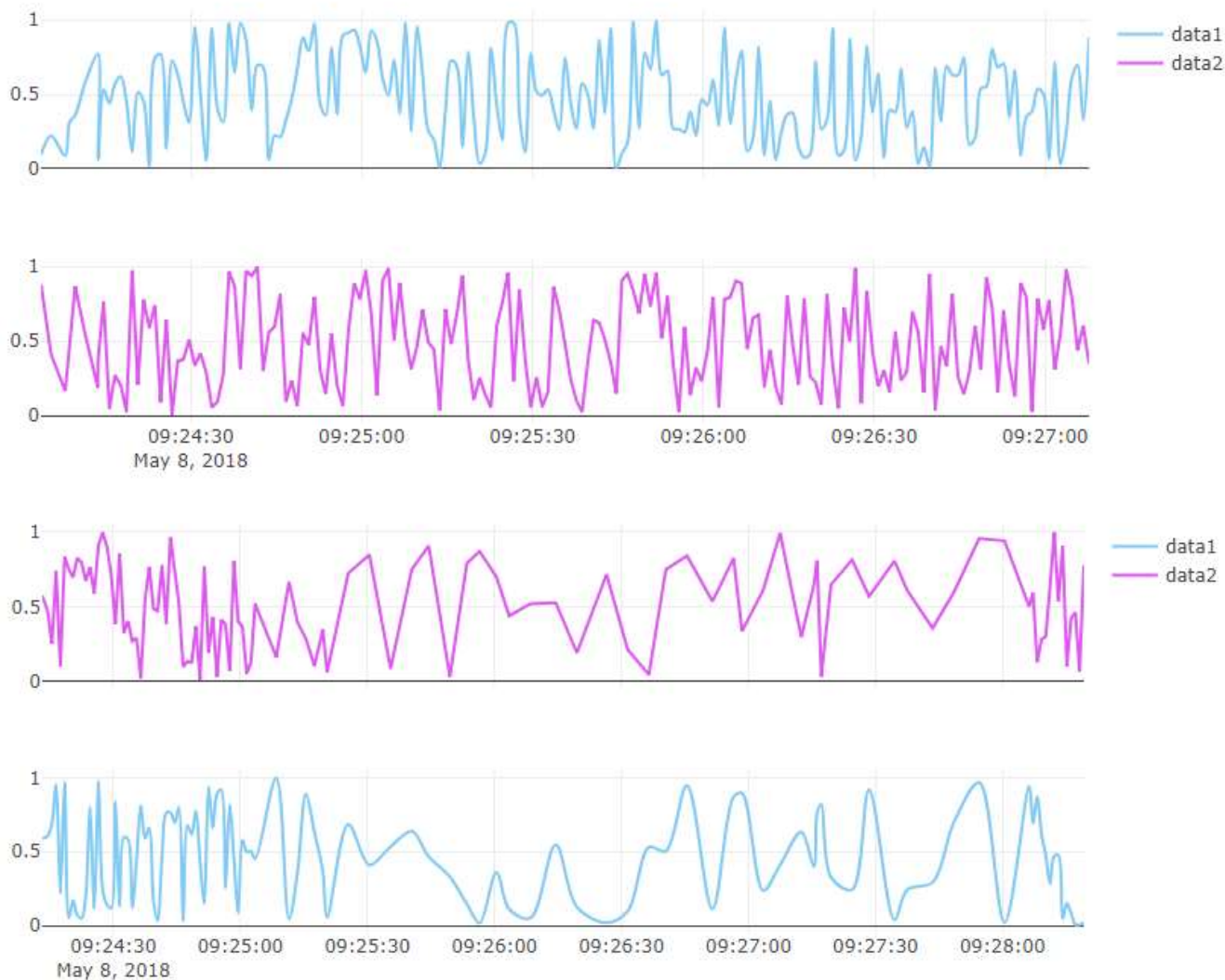
```
// streaming
var cnt = 0;
var interval = setInterval(function() {

  var time = new Date();

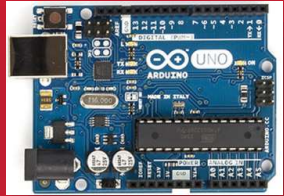
  var update = {
    x: [[time], [time]],
    y: [[rand()], [rand()]]
  }

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


[DIY] Streaming data using multiple axis → change axis

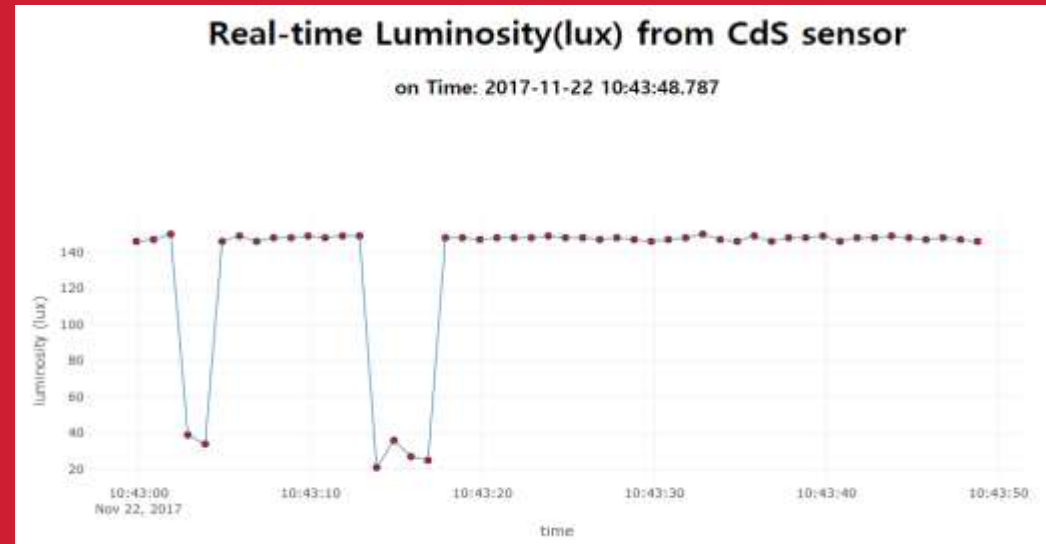


AAnn_DS_multiple_axis.png 로 캡처 저장.

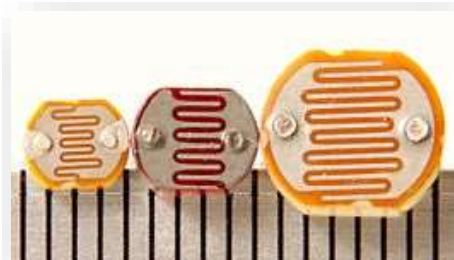


Arduino sensor data RT visualization using **plotly.js**

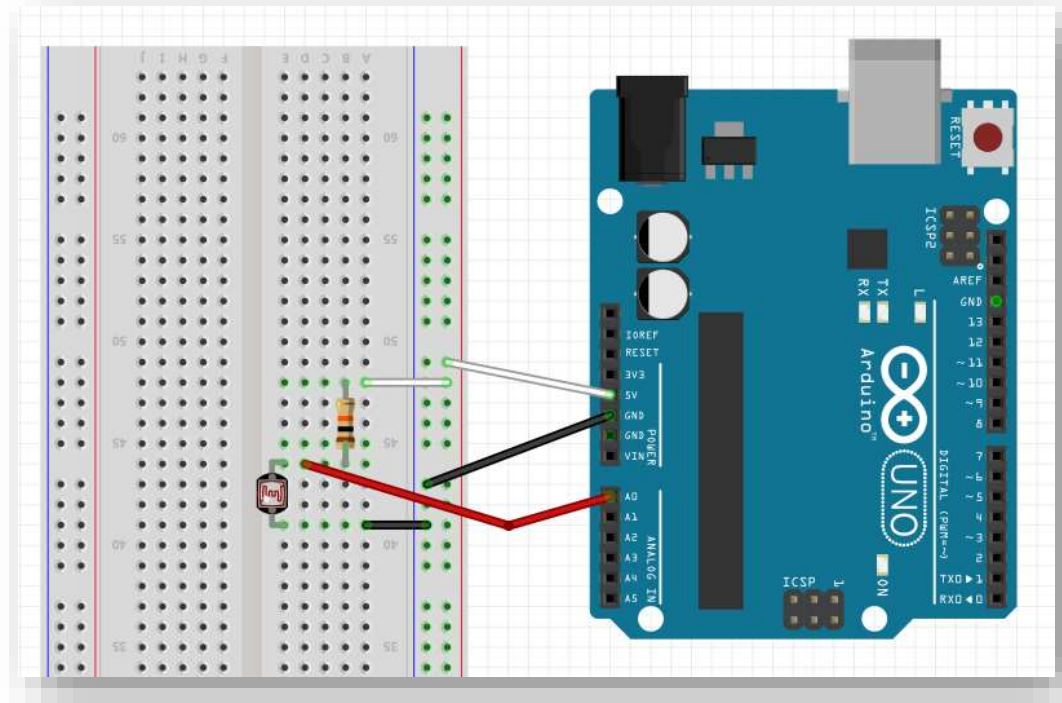
```
AA00,2017-11-22 10:43:11.859,149
AA00,2017-11-22 10:43:12.851,149
AA00,2017-11-22 10:43:13.845,21
AA00,2017-11-22 10:43:14.854,36
AA00,2017-11-22 10:43:15.844,27
AA00,2017-11-22 10:43:16.837,25
AA00,2017-11-22 10:43:17.846,148
AA00,2017-11-22 10:43:18.839,148
AA00,2017-11-22 10:43:19.847,147
```



Luminosity sensor [Photocell LDR]



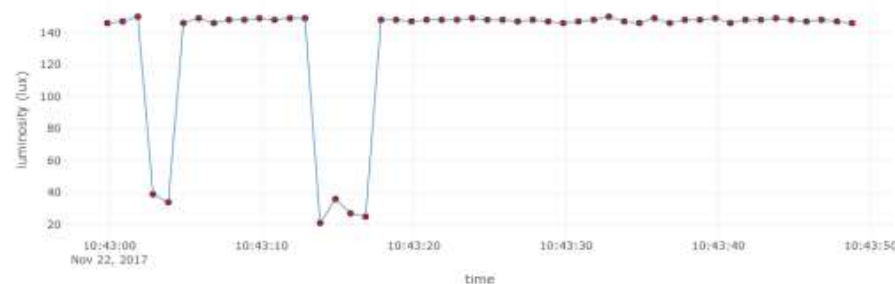
CdS



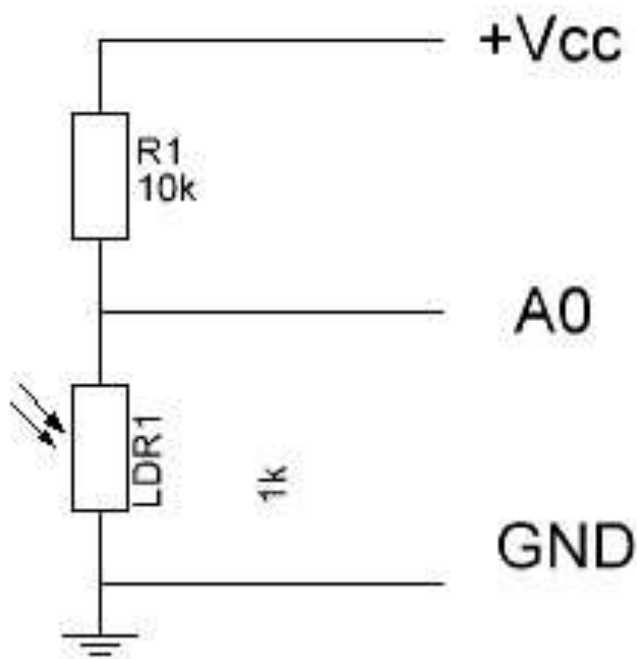
Real-time Luminosity(lux) from CdS sensor

on Time: 2017-11-22 10:43:48.787

```
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



$$A_o \rightarrow V_o \rightarrow \text{lux}$$

$$\text{lux} = 500 / R_{ldr}$$

$$V_o = I_{ldr} * R_{ldr}$$

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

$$R_{ldr} = 10 * V_o / (5 - V_o)$$

$$\text{lux} = 250 / V_o - 50$$

$$V_o = 5.0 * A_o / 1023.0$$

```
//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;
}
```

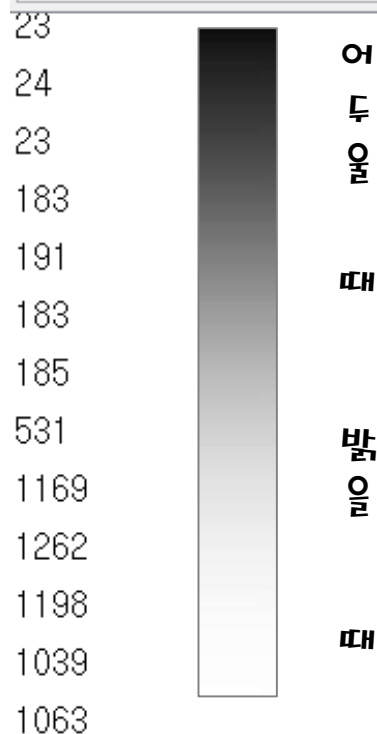
CdS 센서 회로 - 측정 2.

```

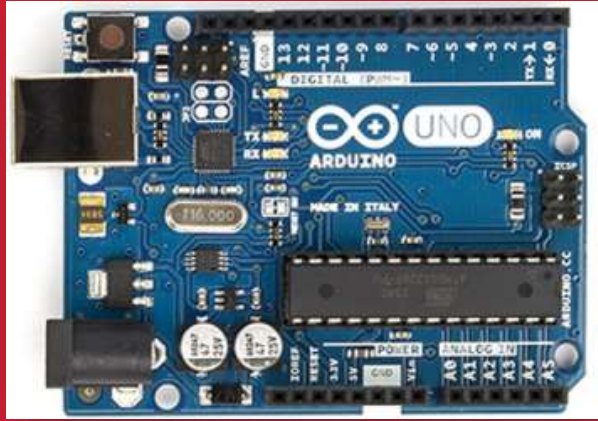
sketch08_CdS2
1 // lux
2 #define CDS_INPUT 0
3
4 void setup() {
5   Serial.begin(9600);
6 }
7 void loop() {
8   int value = analogRead(CDS_INPUT);
9   Serial.println(int(luminosity(value)));
10  delay(1000);
11 }
12
13 //Voltage to Lux
14 double luminosity (int RawADC0){
15   double Yout=RawADC0*5.0/1023; // 5/1023 (Vin = 5 V)
16   double lux=(2500/Yout-500)/10;
17   // lux = 500 / Rldr, Yout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
18   return lux;
19 }

```

COM11 (Arduino/Genuino Uno)



밝을수록 측정 값이 커지고
어두울수록 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"

D:\Portable\NodeJS\Portable\Data\aa00\iot\cds\package.json (Data) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

- Data
 - aa00
 - express
 - expressTest
 - iot
 - cds
 - /* package.json
 - tmp36
 - myApp
 - server
 - start
 - node_modules
 - npm_cache
 - settings
 - Temp
 - express
 - /* express.cmd

```
1 {
2   "name": "cds",
3   "version": "1.0.0",
4   "description": "cds-node project",
5   "main": "cds_node.js",
6   "scripts": {
7     "test": "echo \\"Error: no test specified\\" && exit 1"
8   },
9   "author": "aa00",
10  "license": "MIT"
11 }
```



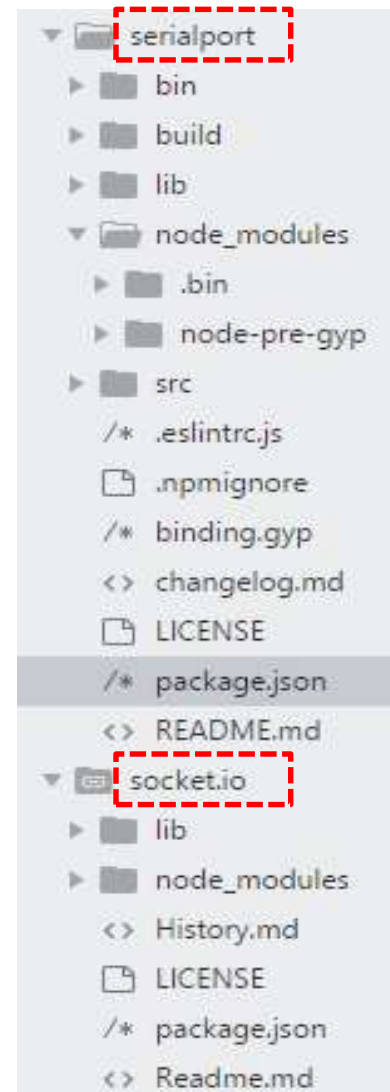
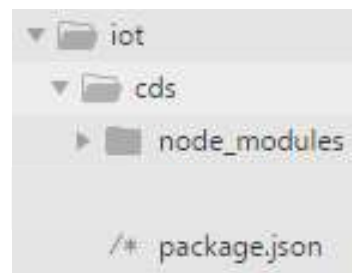
A4.2.2 Luminosity sensor [Photocell LDR]

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 browsing 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

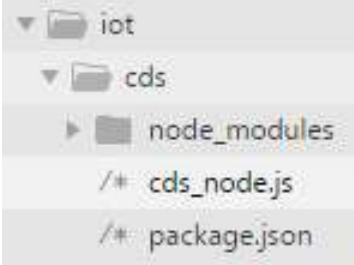
- npm init
- npm install --save serialport@4.0.7
- npm install --save socket.io@1.7.3

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
});

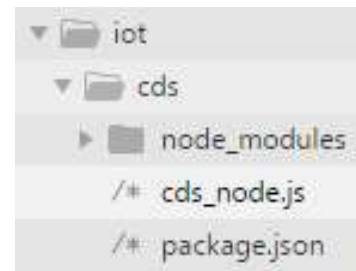
// helper function to get a nicely formatted date string
function getDateString() {
  var time = new Date().getTime();
  // 32400000 is (GMT+9 Korea, GimHae)
  // for your timezone just multiply +/-GMT by 3600000
  var datestr = new Date(time + 32400000).
    toISOString().replace(/T/, ' ').replace(/Z/,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

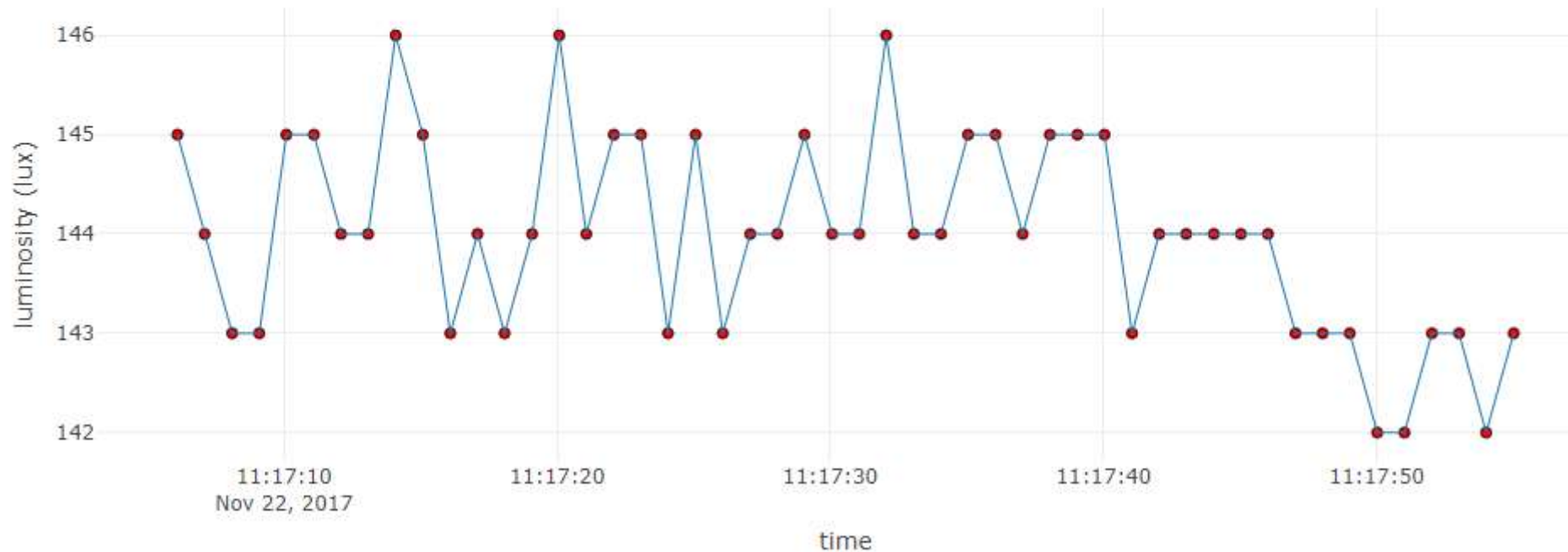
```
0% NodeJS - node cds_node
```

```
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
```

Real-time Luminosity(lux) from CdS sensor

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





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>
```



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);
}
```


[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]
  }
};
```

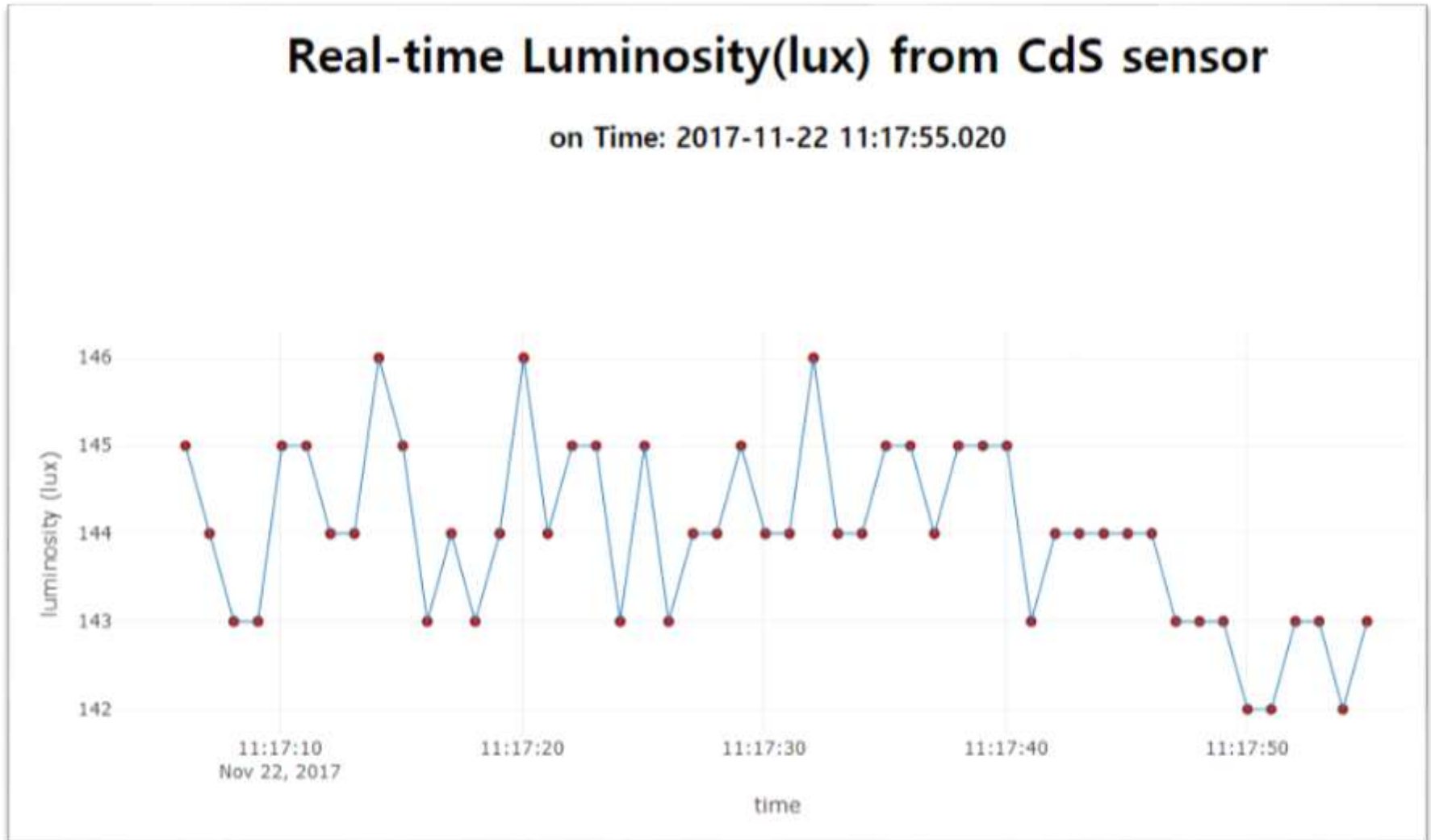
domain: [0,1] → x 또는 y 축을 100% 사용

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(dtdata[0]);  
  
    xTrack.shift();  
    xTrack.push(dtdata[1]); //  
  
    Plotly.redraw(streamPlot);  
}  
*/
```

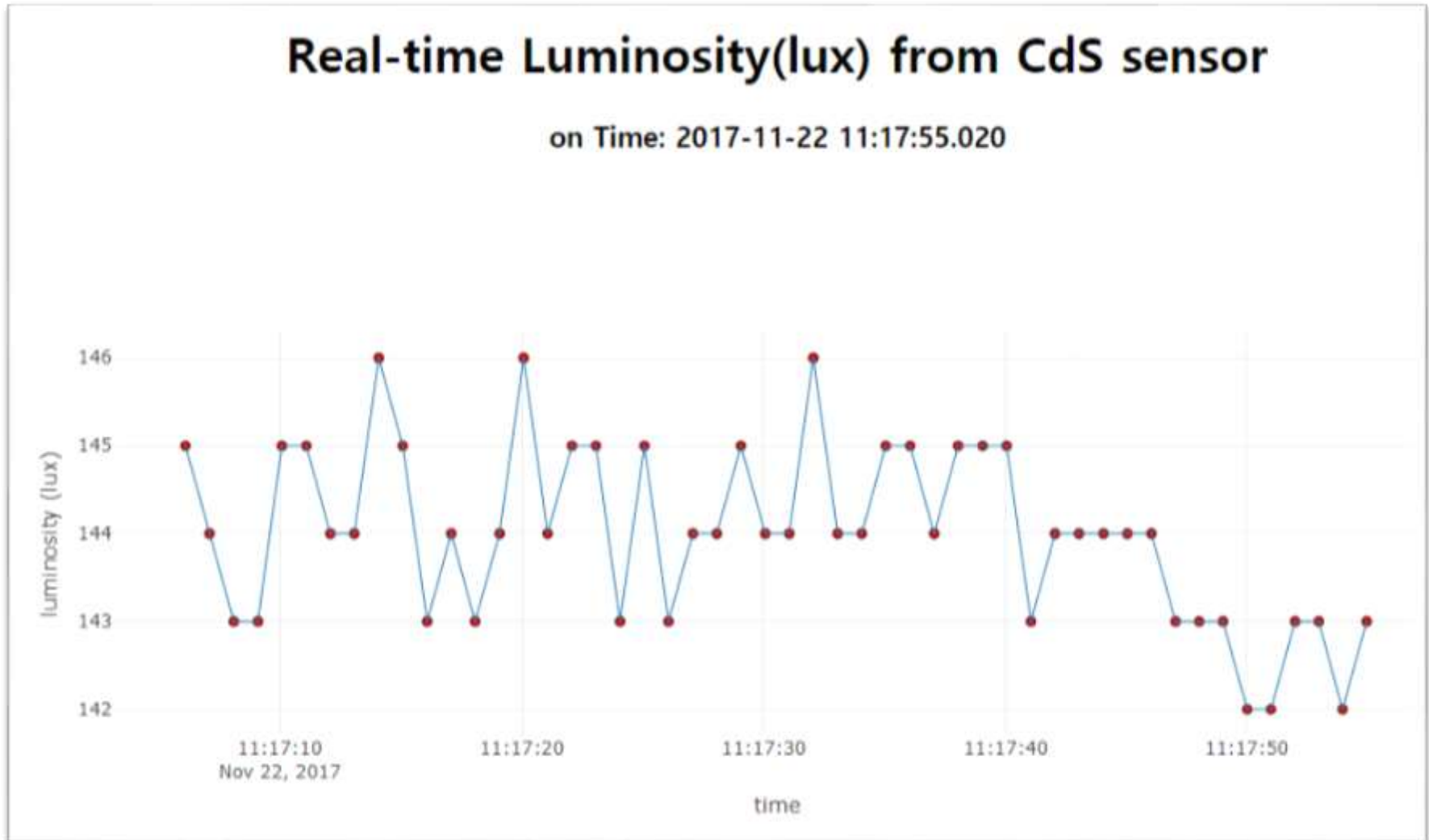
nextPt() 주석 처리

```
socket.on('connect', function () {  
    socket.on('message', function (msg) {  
        // initial plot  
        if(msg[0]!='' && initFlag){  
            dtdata[0]=msg[0];  
            dtdata[1]=parseInt(msg[1]); // lux  
            init(); // start streaming  
            initFlag=false;  
        }  
        console.log(msg[0]);  
        console.log(parseInt(msg[1])); // Convert  
        dtdata[0]=msg[0];  
        dtdata[1] = parseInt(msg[1]);  
  
        // when new data is coming, keep on stream  
        ctime.innerHTML = dtdata[0];  
        //nextPt();  
        tArray = tArray.concat(dtdata[0]); // time  
        tArray.splice(0,1);  
        xTrack = xTrack.concat(dtdata[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

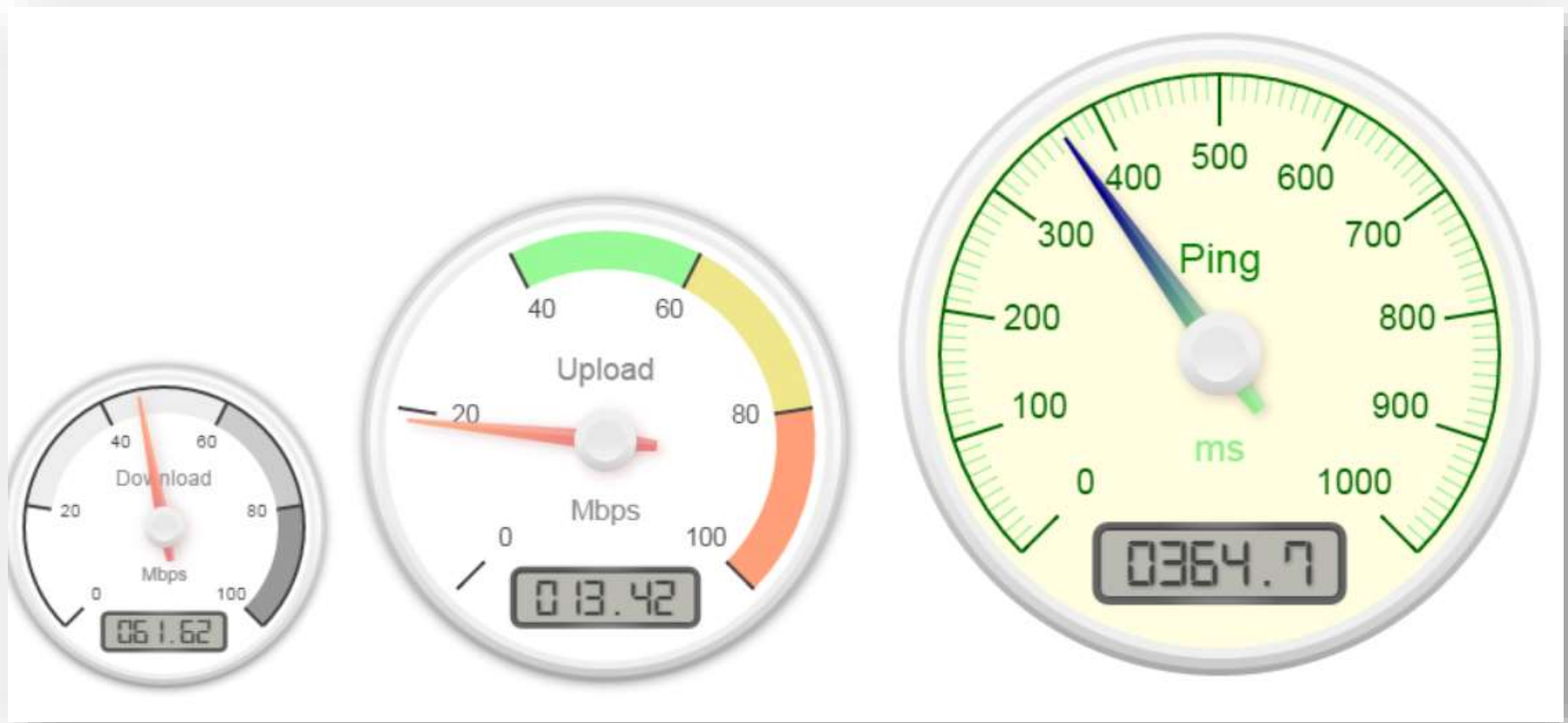
[7.2] Client html : **client_cds2.html** (using plotly streaming without nextPt())





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)

GitHub

This repository: Search

Explore Features Enterprise Pricing

Mikhus / **canv-gauge** Watch 29 Star

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 rblackburn	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/rblackburn/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**)

```
<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]); // 1
    xTrack.splice(0,1);

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

    Plotly.update(streamPlot, update);

  });
});
```

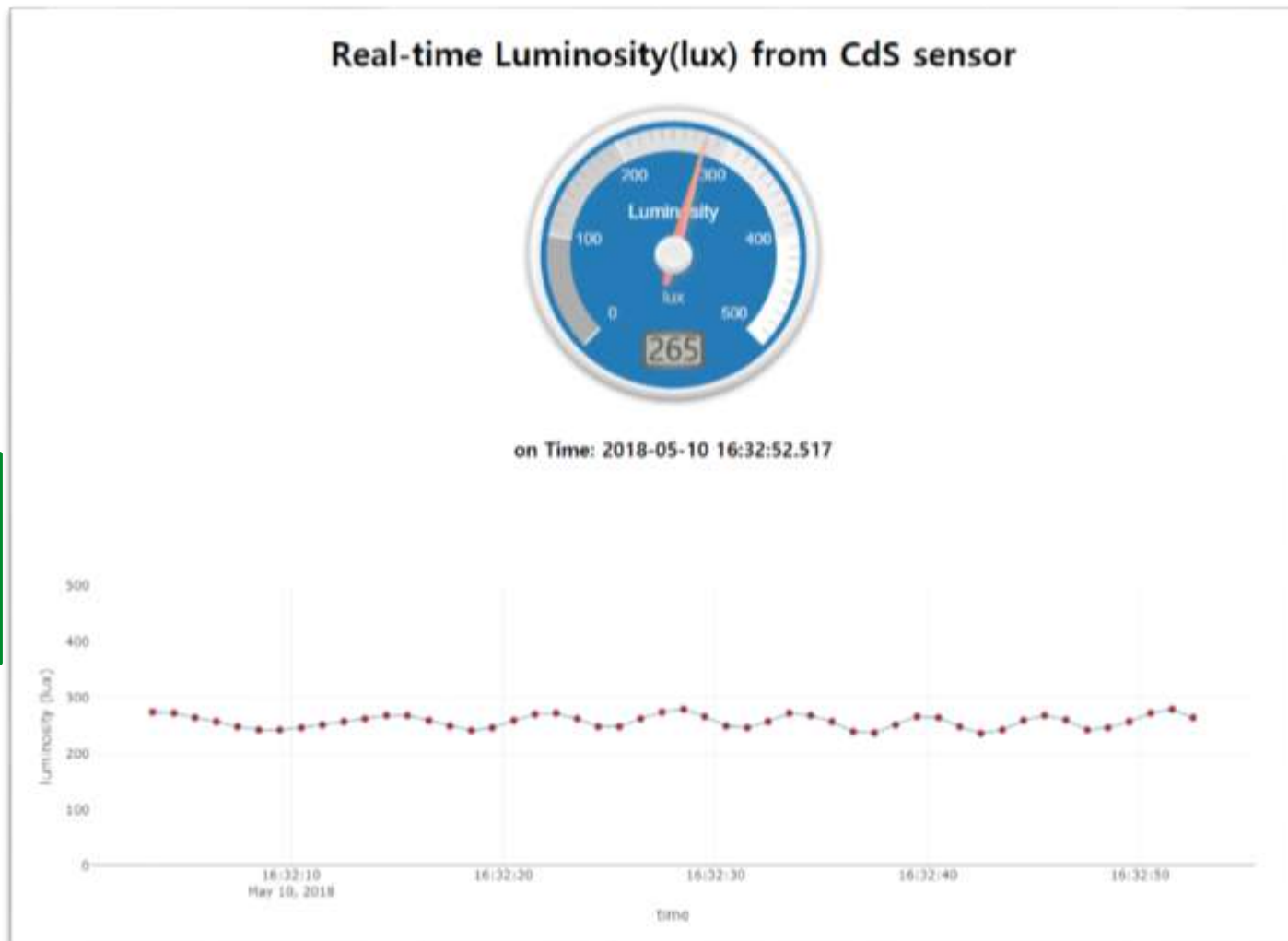
```
var gauge_lux = new Gauge({
  renderTo : 'gauge',
  width : 300,
  height : 300,
  glow : true,
  units : 'lux',
  valueFormat : { int : 2, dec : 0 },
  title : "Luminosity",
  minValue : 0,
  maxValue : 500, // new
  majorTicks : ['0','100','200','300','400','500'],
  minorTicks : 10,
  strokeTicks : 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();
```



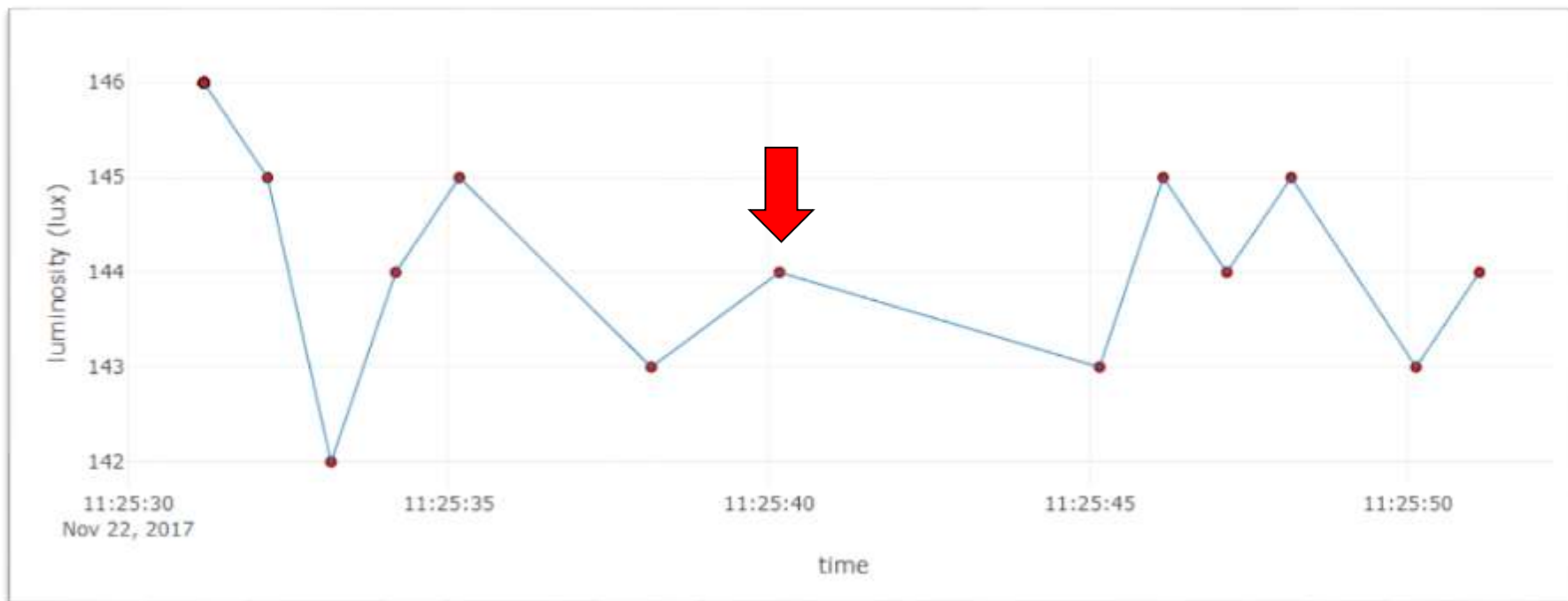
A5.5.8.3 RT sensor-data streaming in Arduino

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

변경된 디자인으로 된
그래프를 캡처하여
[AAnn_cds_gauge.png](#)로 저장



[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.

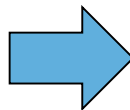
밝기 값 변화의 문턱값을 설정해서 이상 감지 기능 구현



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 = dtdata[0];  
gauge_lux.setValue(dtdata[1]); // lux gauge  
//nextPt();  
tArray = tArray.concat(dtdata[0]); // time  
tArray.splice(0,1);  
xTrack = xTrack.concat(dtdata[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 (dtdata[1] != preX) { // any change?  
  preX = dtdata[1];  
  
  ctime.innerHTML = dtdata[0];  
  gauge_lux.setValue(dtdata[1]); // lux gauge  
  //nextPt();  
  tArray = tArray.concat(dtdata[0]); // time  
  tArray.splice(0,1);  
  xTrack = xTrack.concat(dtdata[1]); // lux  
  xTrack.splice(0,1);  
  
  var update = {  
    x: [tArray],  
    y: [xTrack]  
  }  
  
  Plotly.update(streamPlot, update);  
}
```

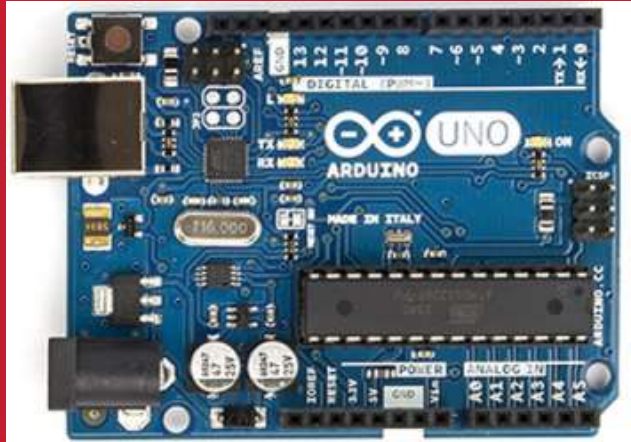
[DIY] Client html : **client_cds_change.html** (detecting change)



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



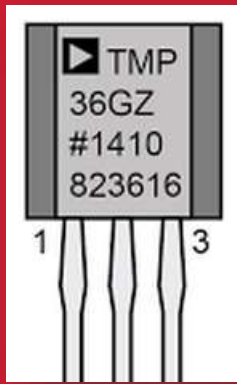
Multiple sensors



CdS + TMP36

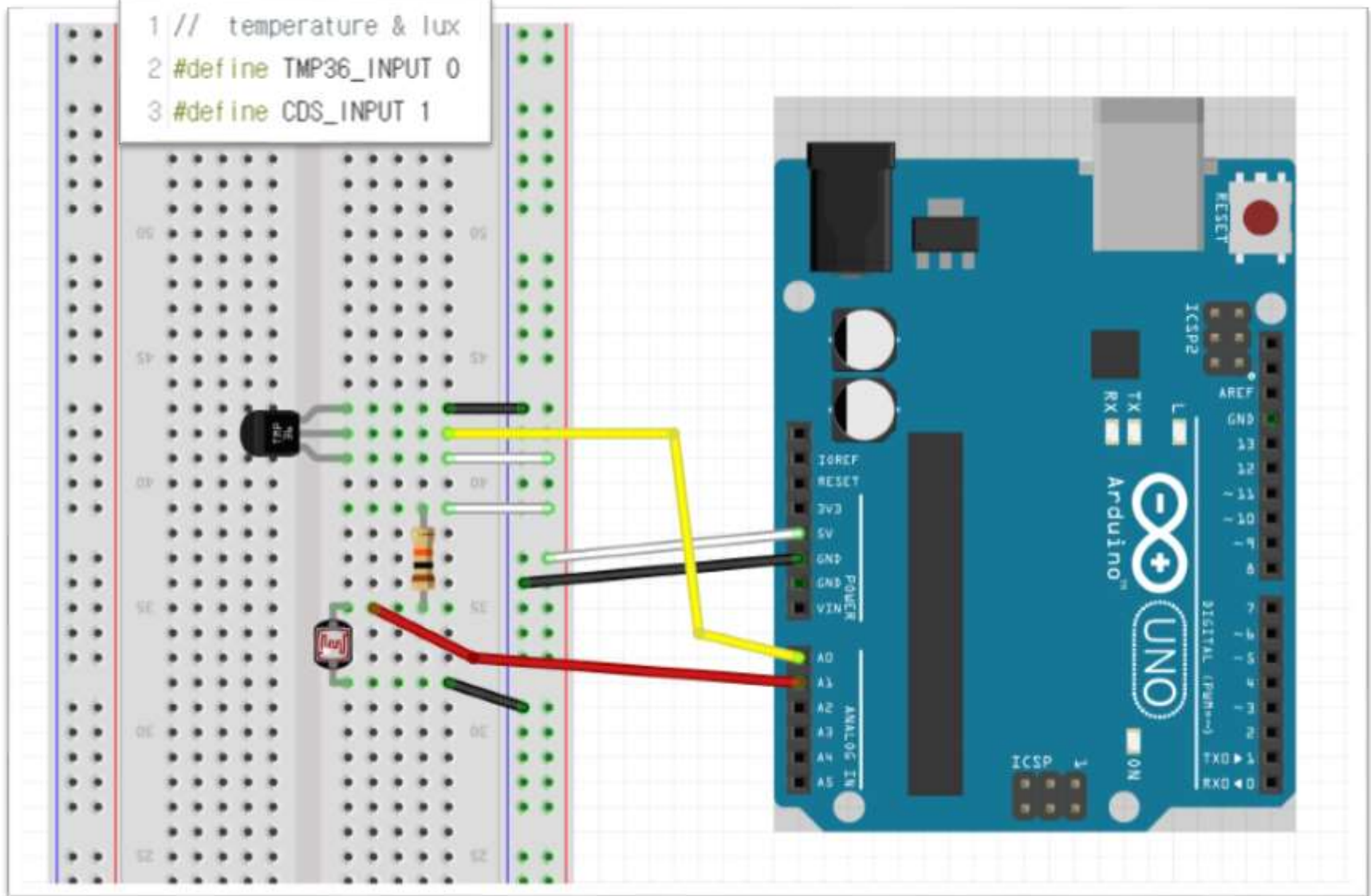
+ plotly.js

Node project



A4.3.1 TMP36 + CdS : circuit

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





A4.3.2 TMP36 + CdS : code

AAnn_TMP36_CdS\$

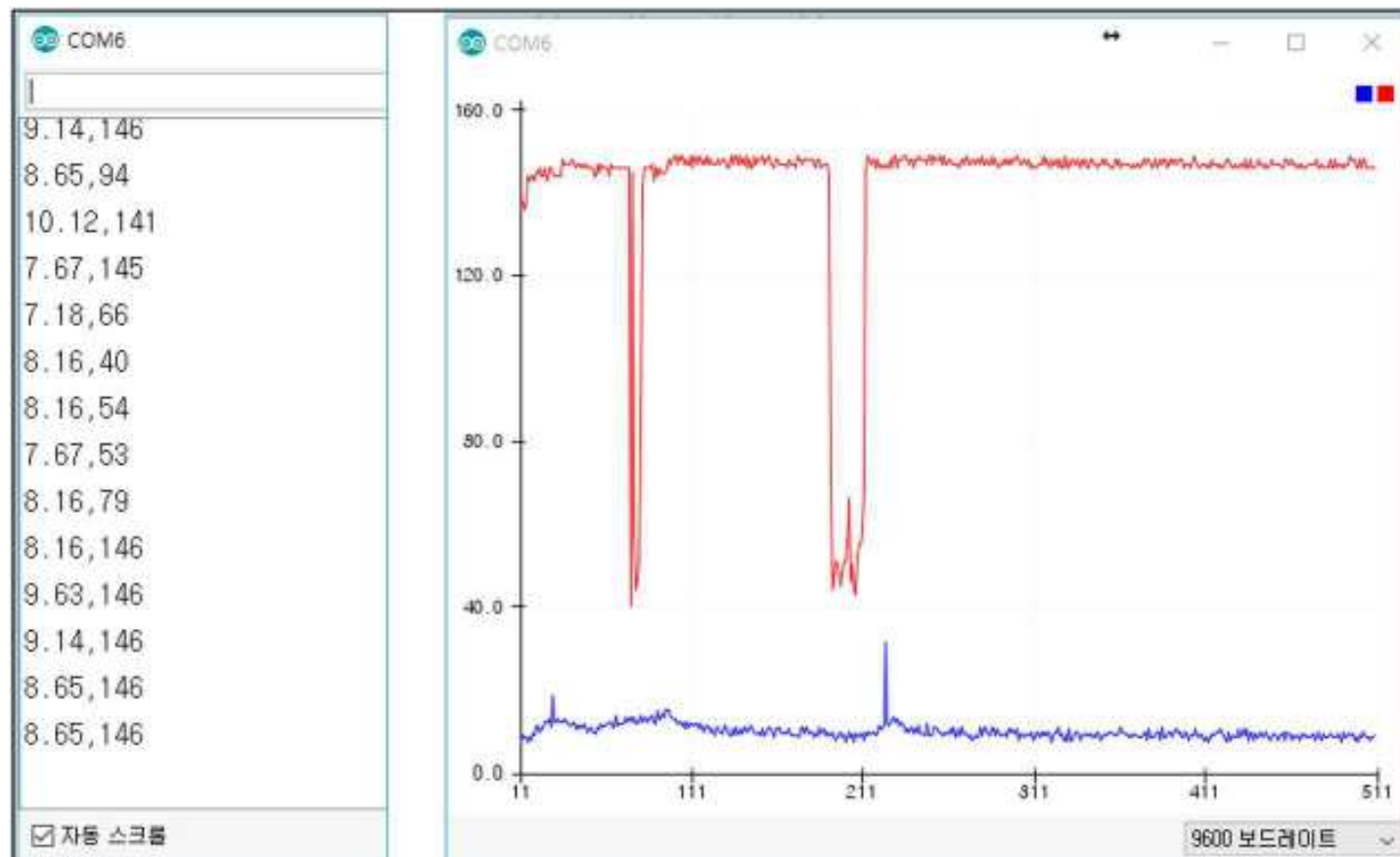
```
1 // temperature & lux
2 #define TMP36_INPUT 0
3 #define CDS_INPUT 1
4
5 void setup() {
6   Serial.begin(9600);
7 }
```

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 Yout=RawADC0*5.0/1023.0; // 5/1023 (Vin = 5 V)
30   int lux=(2500/Yout-500)/10;
31   // lux = 500 / Rldr , Yout = 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": "hsnn"
```

name : cds_tmp36

description : cds-tmp36-node project

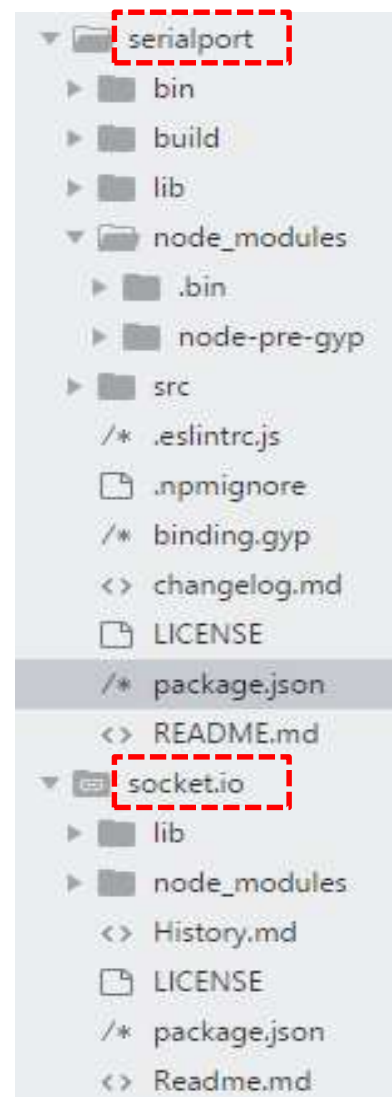
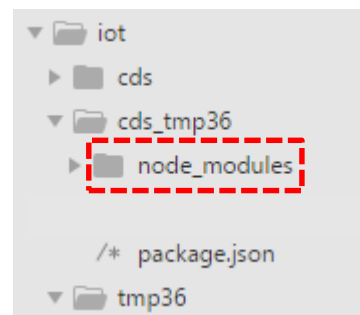
entry point : cds_tmp36_node.js

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 browsing 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"
  },
  "keywords": [
    "cds",
    "tmp36",
    "node"
  ],
  "author": "hs00",
  "license": "MIT",
  "dependencies": {
    "serialport": "^4.0.7",
    "socket.io": "^1.7.3"
  }
}
```



A4.5.4 CdS + TMP36 + Node project

Recycling code:

Save `cds_node.js` as
`cds_tmp36_node.js`

```
▼ iot
  ► cds
  ▼ cds_tmp36
    ► node_modules
    /* cds_tmp36_node.js
    /* package.json
  ▼ tmp36
```



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

cds_tmp36_node.js

```
cds_tmp36_node.js x
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 });
```


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 (dtdda[1] != preX || dtdda[2] != preY) { // any change?
    preX = dtdda[1];
    preY = dtdda[2];

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

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

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

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

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



A5.6.7 TMP36 + CdS streaming project

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

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

```
var gauge_lux = new Gauge({
  renderTo    : 'gauge2',
  width       : 300,
  height      : 300,
  glow        : true,
  units       : 'lux',
  valueFormat : { int : 3, dec : 0 },
  title       : "Luminosity",
  minValue    : 0,
  maxValue    : 500, // new
  majorTicks  : [ '0', '100', '200', '300', '400', '500' ],
  minorTicks  : 10,
  strokeTicks : false,
  highlights  : [
    { from : 0, to : 100, color : '#aaa' },
    { from : 100, to : 200, color : '#ccc' },
    { from : 200, to : 300, color : '#ddd' },
    { from : 300, to : 400, color : '#eee' },
    { from : 400, to : 500, color : '#fff' }
  ],
  colors      : {
    plate      : '#1f77b4',
    majorTicks : '#f5f5f5',
    minorTicks : '#aaa',
    title      : '#fff',
    units      : '#ccc',
    numbers    : '#eee',
    needle     : { start : 'rgba(240, 128, 128, 1)',
                  end   : 'rgba(255, 160, 122, .9)' }
  }
});
gauge_lux.draw();
```



A5.6.8 TMP36 + CdS streaming project

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

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



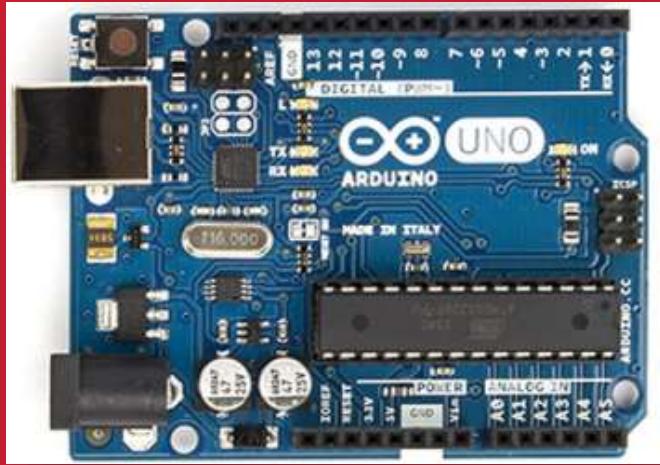
on Time: 2018-01-22 10:05:30.813



AAnn_DS_cds_tmp36.png 로 저장



CdS + DHT22

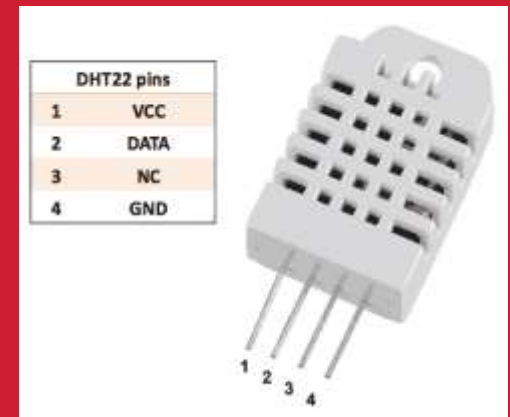


+ **plotly.js**

Node project

Multi-sensors

DHT22 + CdS



DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



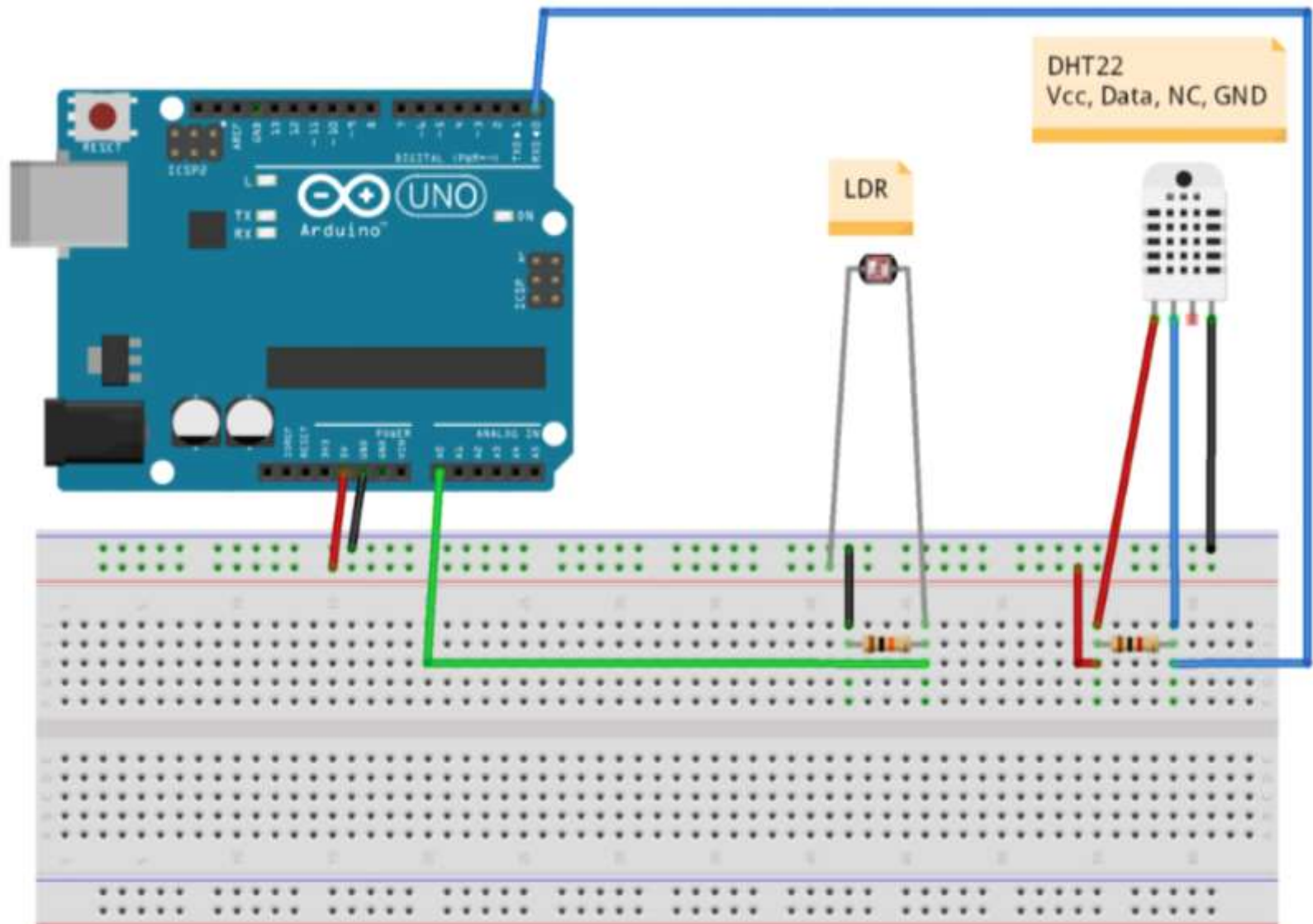
그림 8-7 DHT22 pin 구조

- 3 ~ 5V power and I/O
- 2.5mA max current
- [0-100%] humidity readings with 2-5% accuracy
- [-40 to 80°C] temperature readings $\pm 0.5^{\circ}\text{C}$ accuracy
- 0.5 Hz sampling rate

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

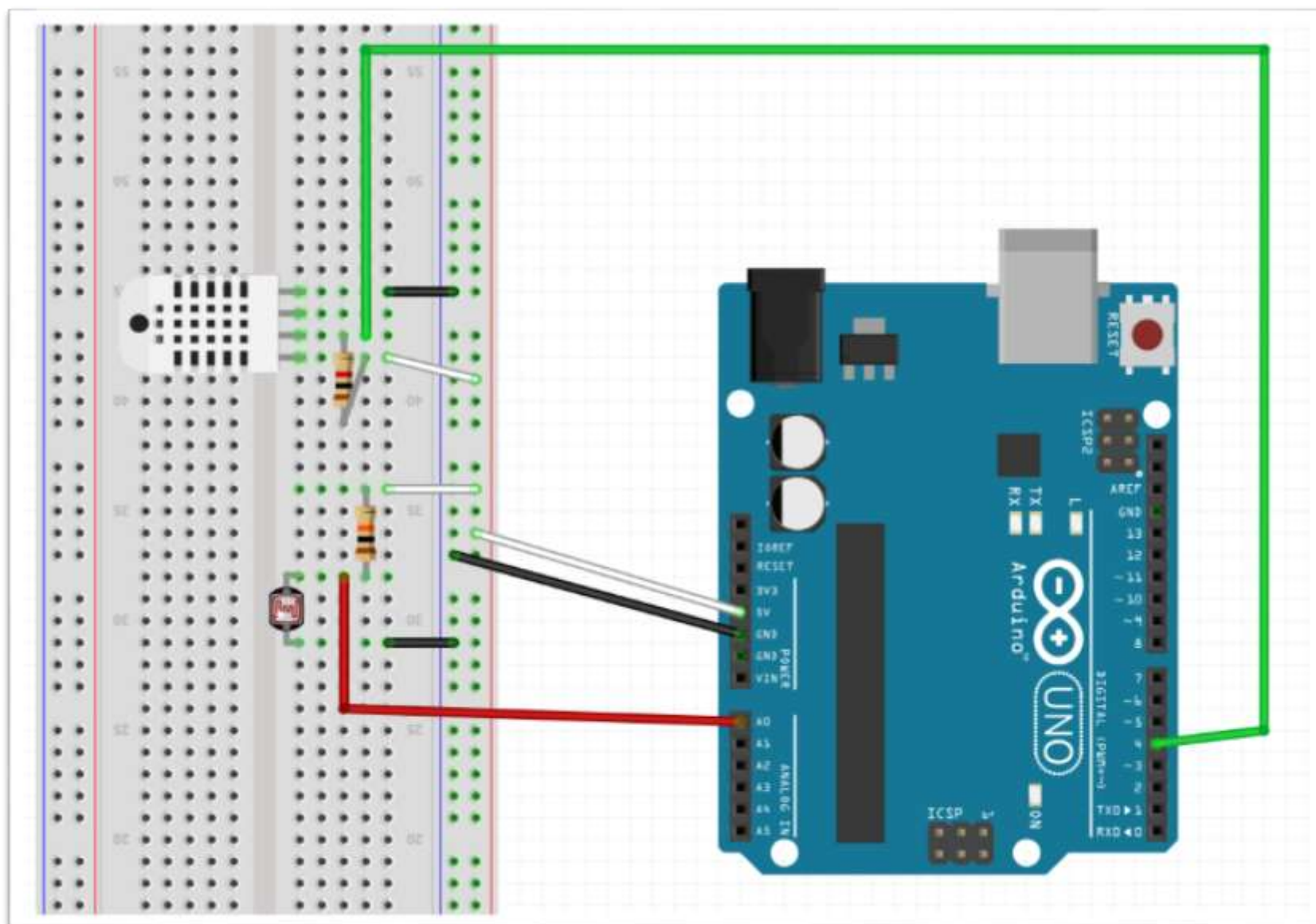


A5.7 DHT22 + CdS streaming project





A5.7.1 DHT22 + CdS circuit



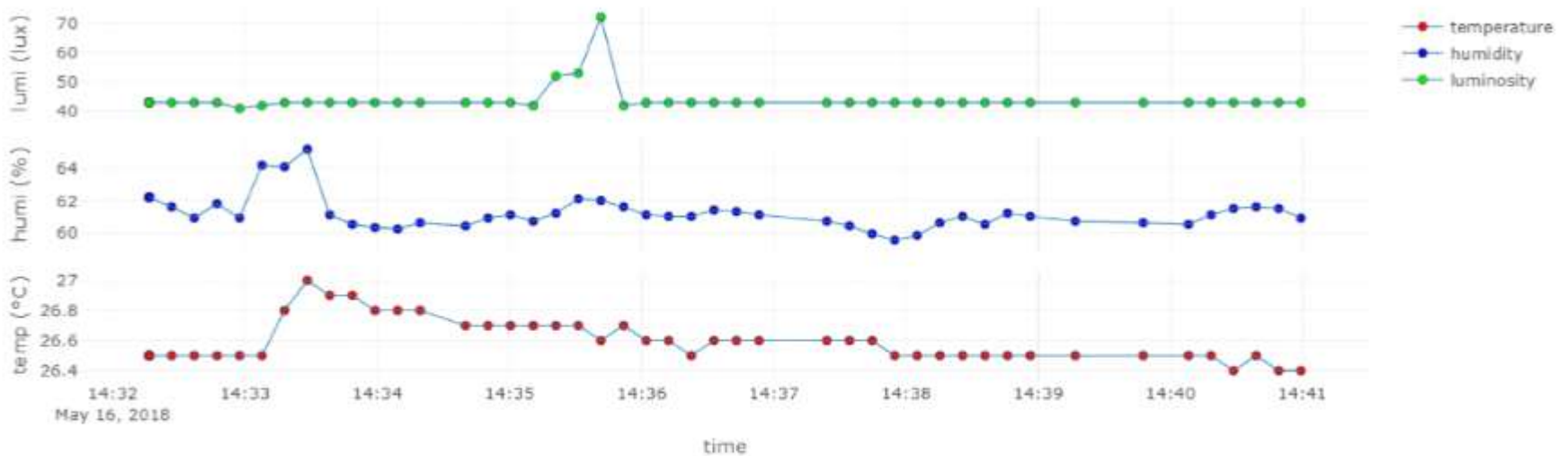
Real-time Weather Station from sensors

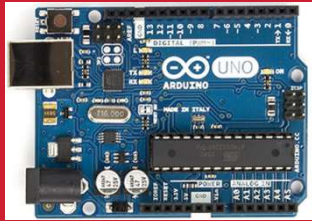


on Time: 2018-05-16 14:40:59.402

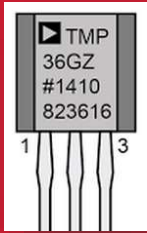
Save as

[AAnn_cds_dht22.png](#)



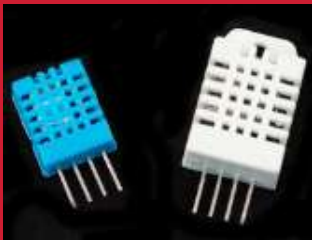


[Practice]



◆ [wk11]

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



◆ [Target of this week]

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

제출파일명 : AAnn_Rpt08.zip

- 압축할 파일들

- ① **AAnn_DS_30timestamps.png**
- ② **AAnn_DS_multiple_axis.png**
- ③ **AAnn_cds_gauge.png**
- ④ **AAnn_cds_change.png**
- ⑤ **AAnn_DS_cds_tmp36.png**

Email : chaos21c@gmail.com

[제목 : id, 이름 (수정)]

● References & good sites

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



주교재 및 참고도서

아두이노와 Node.js에 기반한 IOT 신호 시각화

| 저자 이 상 훈 |

인제대학교 출판부

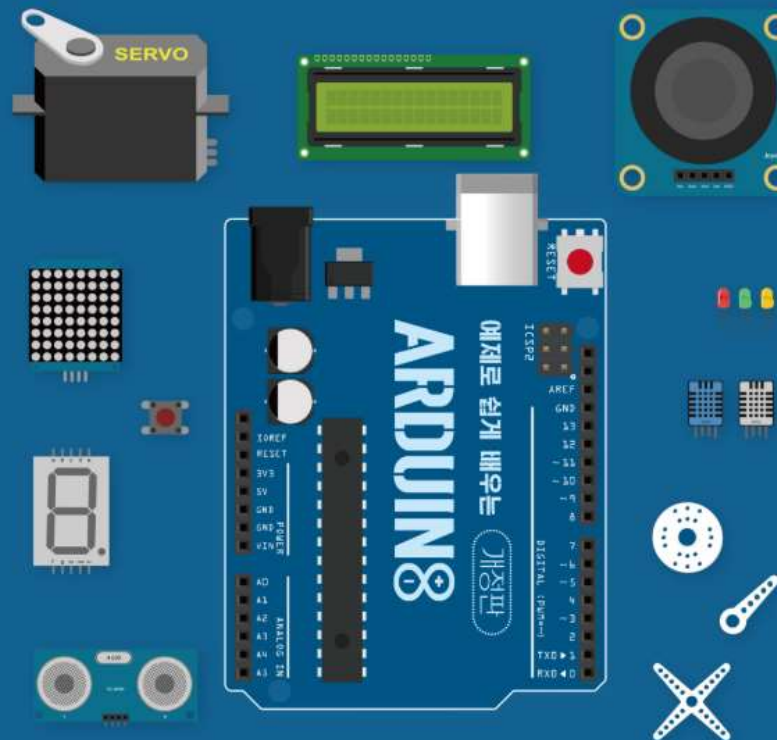
아두이노와 Node.js에 기반한

IOT 신호 시각화

| 저자 이 상 훈 |



인제대학교 출판부



예제로 쉽게 배우는

아두이노

개정판

장성용 · 김진환 지음

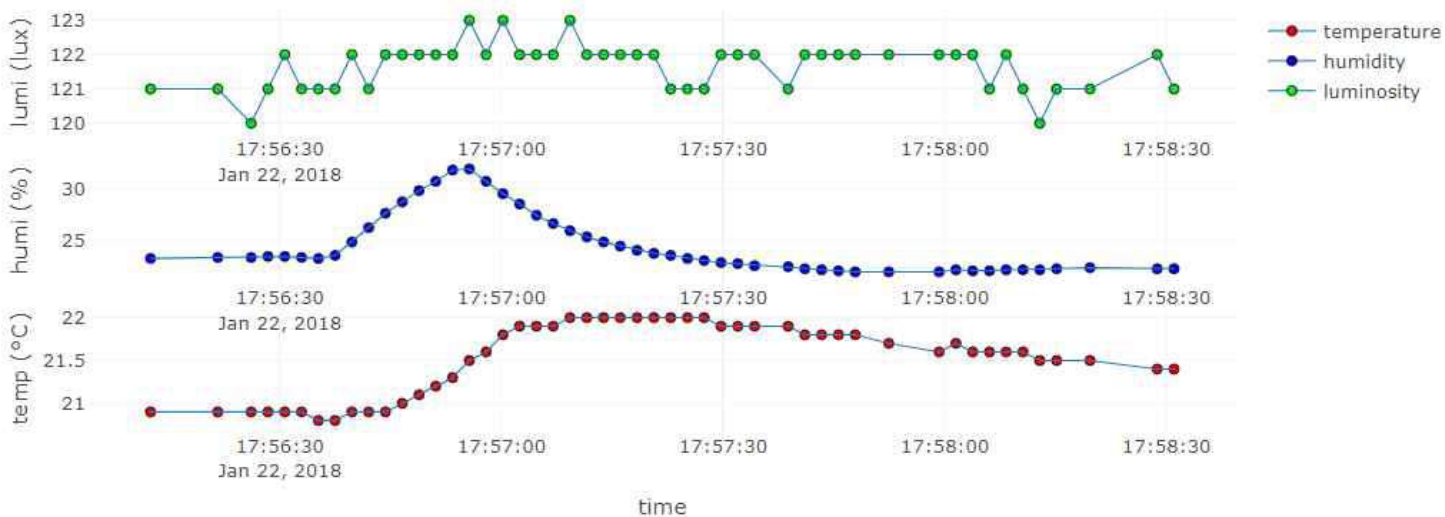
새로운 출판

Target of this class

Real-time Weather Station from sensors



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



Another target of this class

PPG with rangeslider

