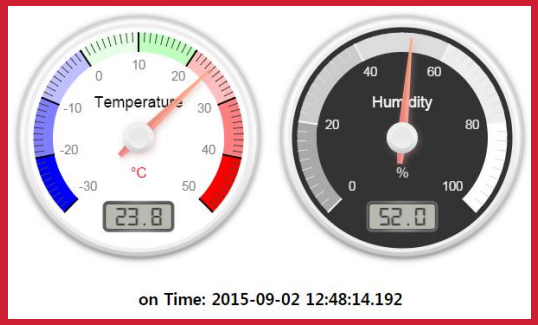
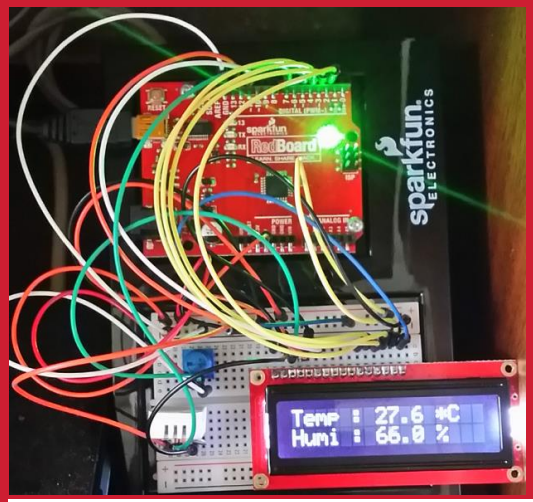


HW-SW-Connectivity

[wk12]

Arduino & NodeJS III

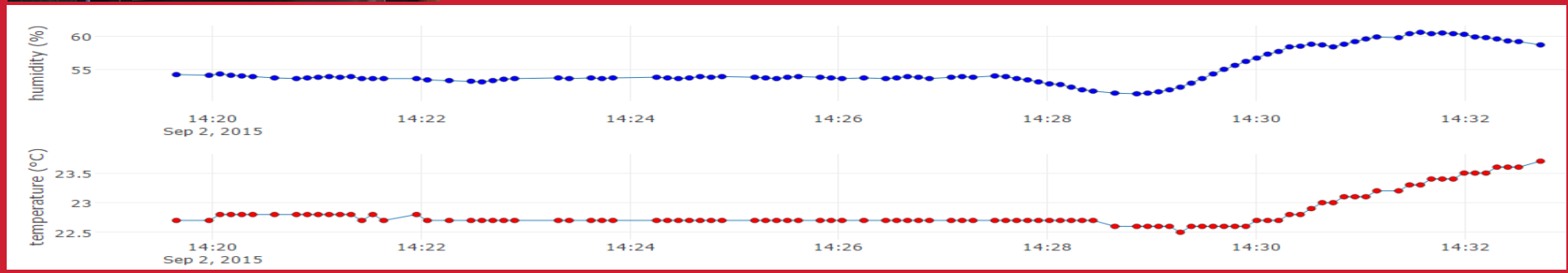


Basic HW and SW Integration using Arduino & Javascript

COMSI, INJE University

2nd semester, 2017

Email : yish@inje.ac.kr



[Practice]

- [wk11]
  - Arduino sensors & node.js
  - Complete your TMP36-CdS project
  - Upload file name : AAnn_Rpt08.zip

wk11 : Practice-08 : AAnn_Rpt08.zip

- [Target of this week]
  - Complete your projects
  - Save your outcomes and compress 4 figures

제출파일명 : **AAnn_Rpt08.zip**

- 압축할 파일들

① AAnn_lux_data.png

② AAnn_tmp36_lux_LCD.png

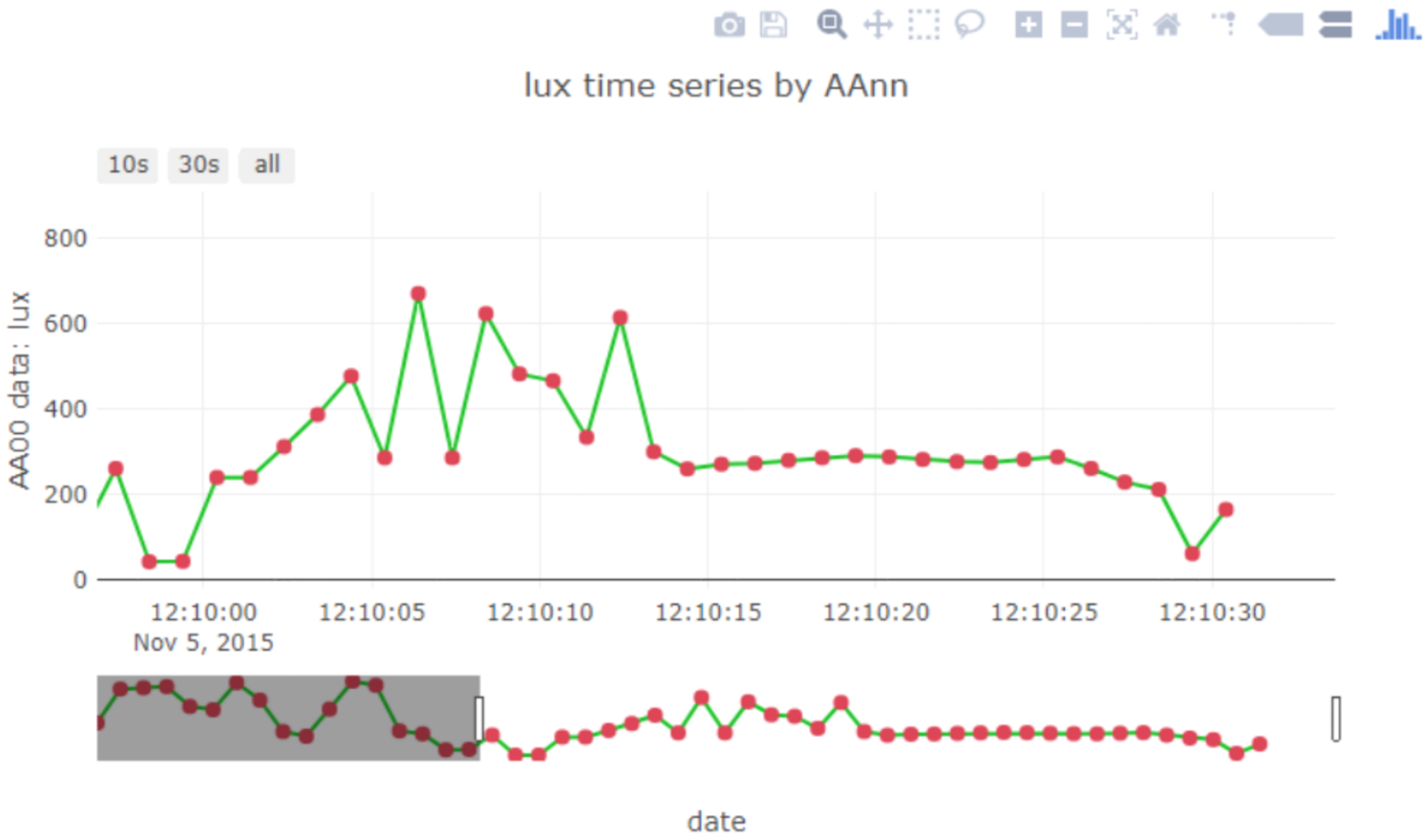
③ tmp36_ldr_node.js

④ AAnn_tmp36_lux_data.png

Email : chaos21c@gmail.com

Data visualization : plotly.js

Time series by AAnn



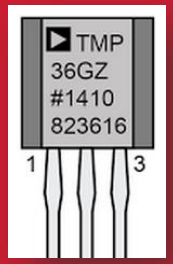
Arduino

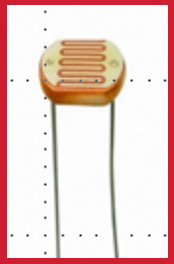
ARDUINO Genuino

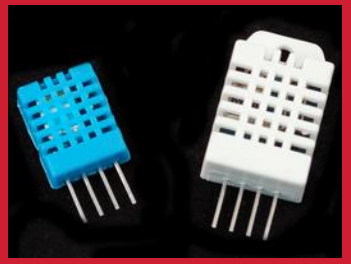
Home | Buy | Download | Products | Learning | Forum | Support | Blog

https://www.arduino.cc/

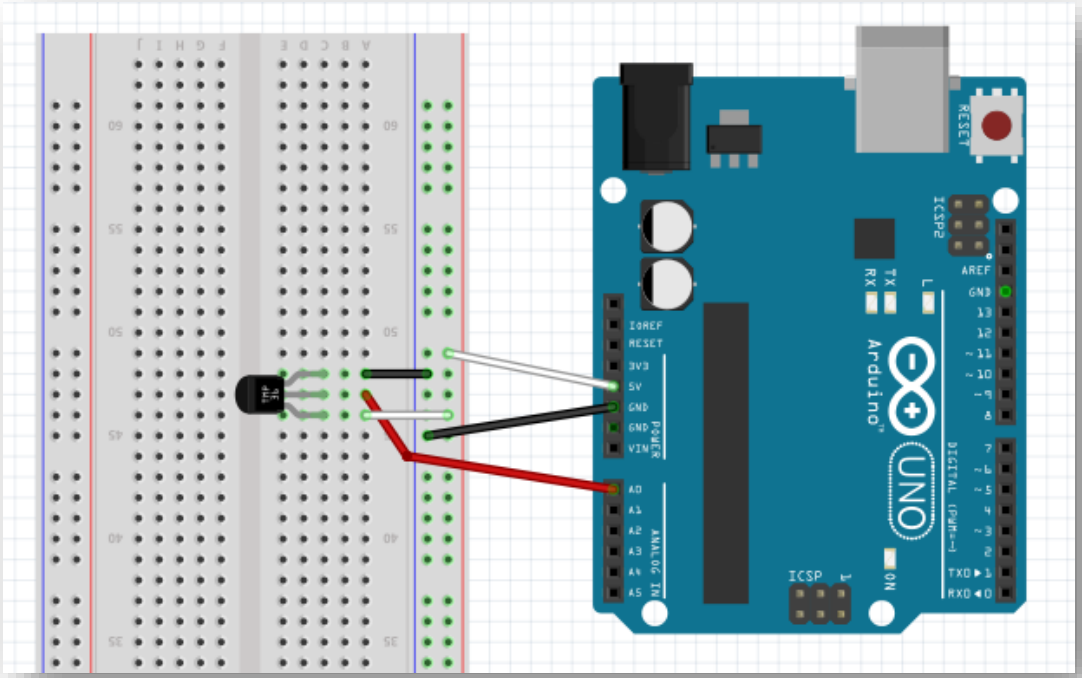
Arduino & Node.js

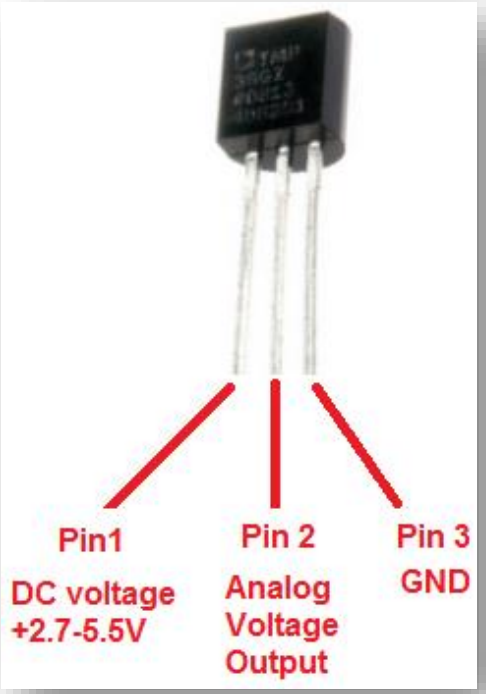




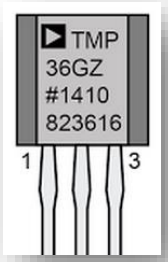


A3.1.2 Temperature sensor [TMP36]





Parts : TMP36



- **Size:** TO-92 package (about 0.2" x 0.2" x 0.2") with three leads
- **Price:** $2.00 at the Adafruit shop
- **Temperature range:** -40°C to 150°C / -40°F to 302°F
- **Output range:** 0.1V (-40°C) to 2.0V (150°C) but accuracy decreases after 125°C
- **Power supply:** 2.7V to 5.5V only, 0.05 mA current draw

A5.1.11 tmp36 node project (date & data → IOT)

tmp36_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(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;
}
```

Serial output (°C)

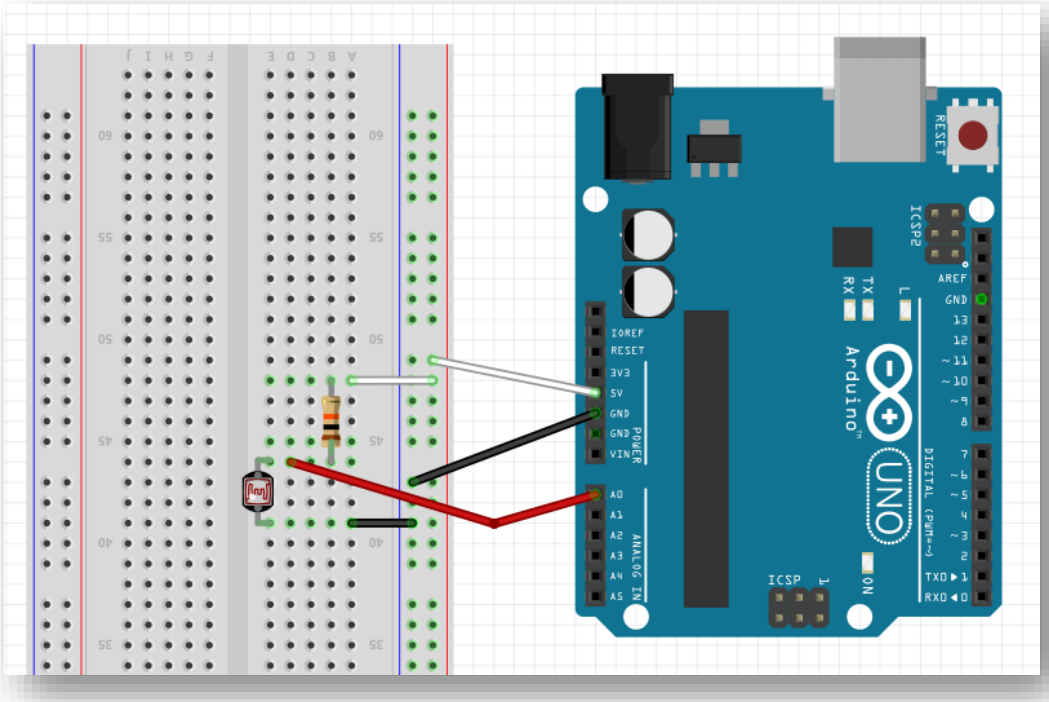
IOT data fromat

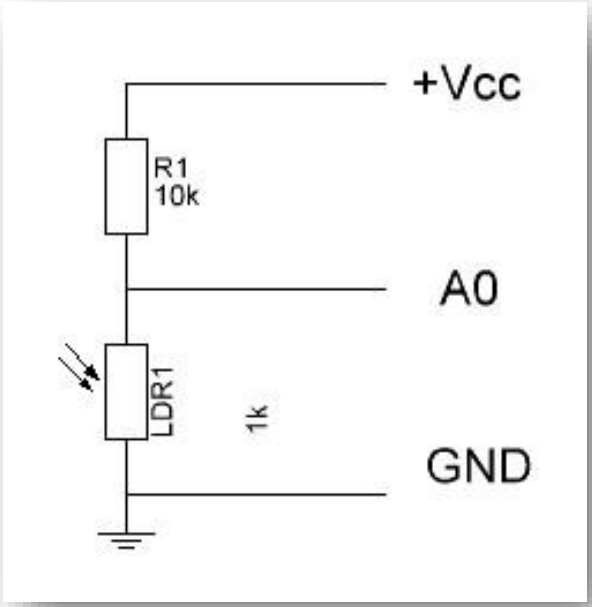
```
[ '2017-11-01 12:46:20.033', '15.49' ]
[ '2017-11-01 12:46:21.042', '15.49' ]
[ '2017-11-01 12:46:22.034', '13.54' ]
[ '2017-11-01 12:46:23.026', '14.03' ]
[ '2017-11-01 12:46:24.035', '15.00' ]
[ '2017-11-01 12:46:25.027', '14.52' ]
[ '2017-11-01 12:46:26.035', '16.47' ]
[ '2017-11-01 12:46:27.028', '15.98' ]
[ '2017-11-01 12:46:28.020', '15.98' ]
[ '2017-11-01 12:46:29.028', '15.49' ]
[ '2017-11-01 12:46:30.021', '13.05' ]
[ '2017-11-01 12:46:31.013', '15.49' ]
[ '2017-11-01 12:46:32.021', '15.00' ]
```

AAnn_tmp36_IOT_data.png
로 저장

A3.2.2 Luminosity sensor [Photocell LDR]

CdS 센서 회로





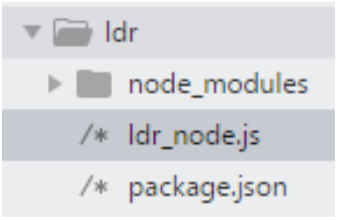
Parts : 20 mm photocell LDR, R (10 kΩ X 1)

광센서에서의 전압 강하 값을 **A0**로 측정

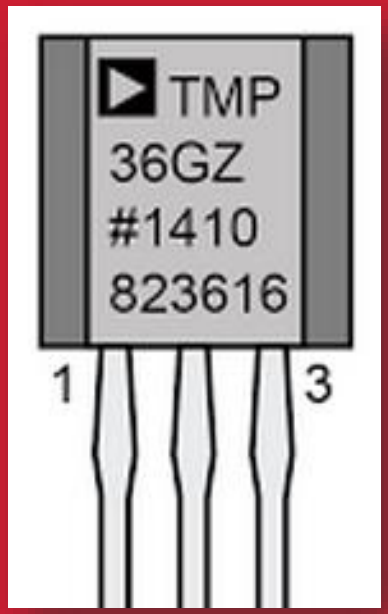
A5.2.5 Luminosity sensor [Photocell LDR]

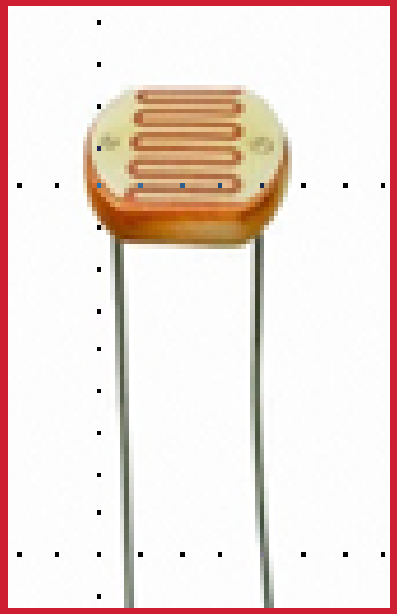
Run ldr_node.js (^B)

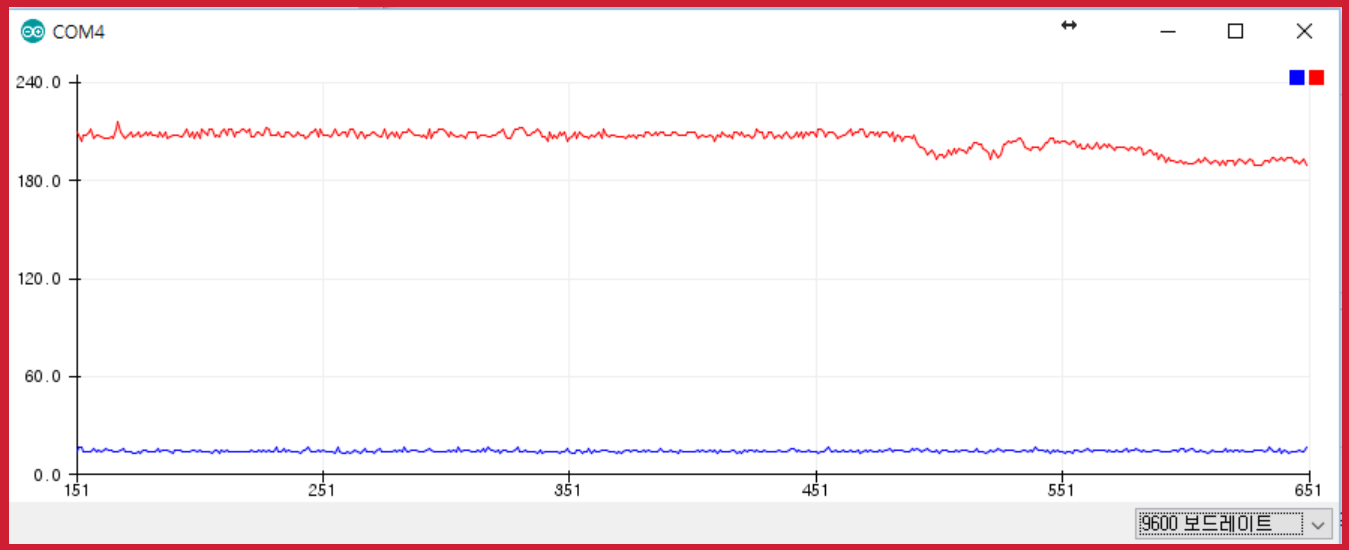
```
AA00,2017-11-08 08:49:54.597,171
AA00,2017-11-08 08:49:55.589,171
AA00,2017-11-08 08:49:56.598,173
AA00,2017-11-08 08:49:57.589,173
AA00,2017-11-08 08:49:58.596,172
AA00,2017-11-08 08:49:59.588,171
AA00,2017-11-08 08:50:00.580,173
AA00,2017-11-08 08:50:01.588,173
AA00,2017-11-08 08:50:02.579,171
AA00,2017-11-08 08:50:03.586,172
AA00,2017-11-08 08:50:04.578,173
AA00,2017-11-08 08:50:05.571,172
```



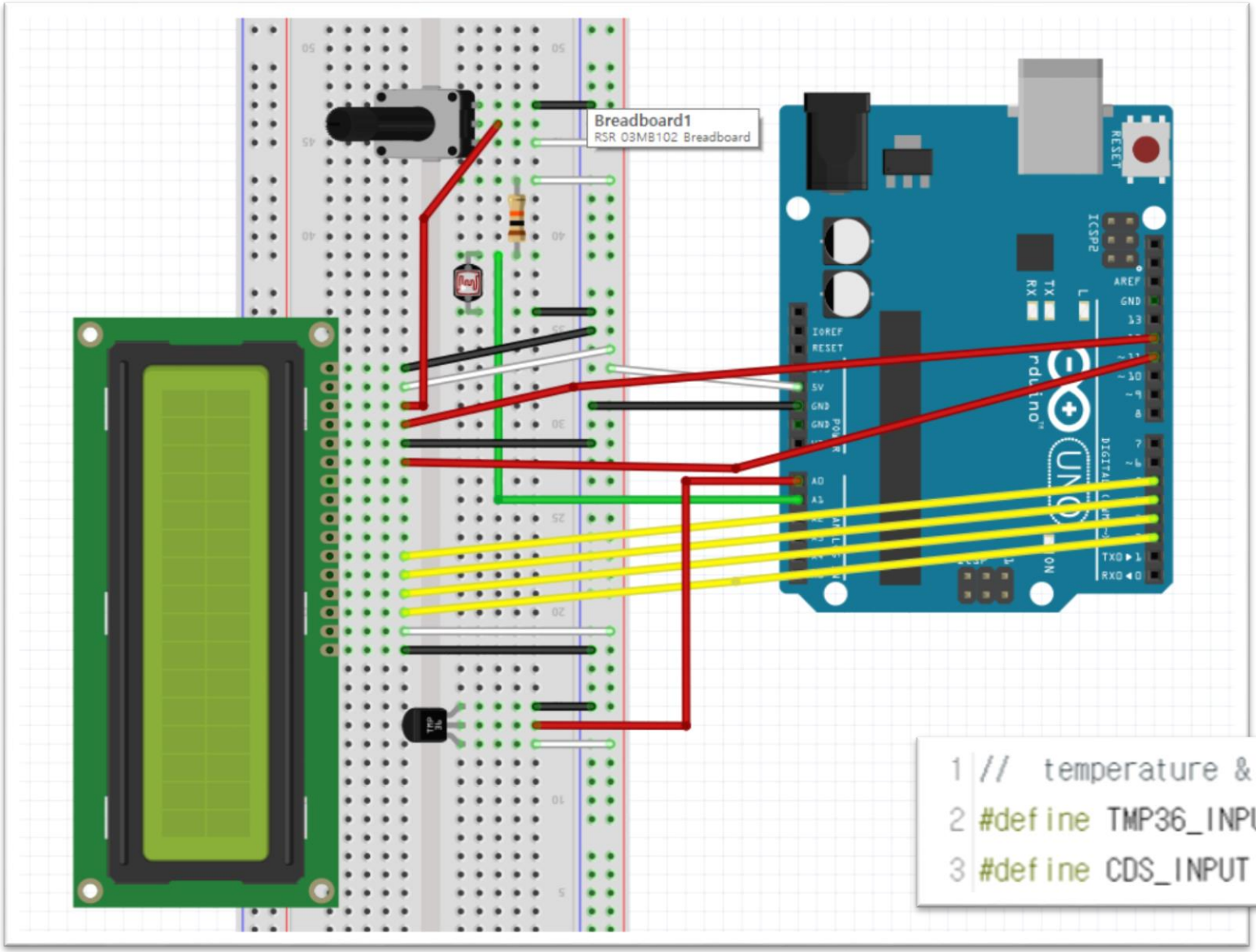
Save as
AAnn_lux_data.png


TMP
36GZ
#1410
823616
1
3




COM4
240.0
180.0
120.0
60.0
0.0
151
251
351
451
551
651
9600 보드레이트

A5.3.4 TMP36 + CdS + LCD : circuit




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

A5.3.7 TMP36 + CdS + LCD + NodeJS

Recycling code:

Save ldr_node.js as tmp36_ldr_node.js

and run in NodeJs

Save as AAnn_tmp36_lux_data2.png

NodeJS - node tmp36_ldr_node

```
D:\Portable\NodeJSPortable\Data\aa00\ldr>node tmp36_ldr_node
AA00,2017-11-08 11:17:49.040,14.52,95
AA00,2017-11-08 11:17:50.047,12.56,95
AA00,2017-11-08 11:17:51.055,14.03,95
AA00,2017-11-08 11:17:52.063,14.52,94
AA00,2017-11-08 11:17:53.070,14.03,95
AA00,2017-11-08 11:17:54.063,14.03,95
AA00,2017-11-08 11:17:55.069,13.54,95
AA00,2017-11-08 11:17:56.077,12.56,95
AA00,2017-11-08 11:17:57.084,13.54,95
AA00,2017-11-08 11:17:58.092,13.54,95
AA00,2017-11-08 11:17:59.085,12.56,95
AA00,2017-11-08 11:18:00.091,13.54,95
AA00,2017-11-08 11:18:01.098,13.05,95
AA00,2017-11-08 11:18:02.105,13.54,95
AA00,2017-11-08 11:18:03.113,13.54,95
AA00,2017-11-08 11:18:04.121,14.52,94
AA00,2017-11-08 11:18:05.112,15.00,95
AA00,2017-11-08 11:18:06.120,14.52,93
AA00,2017-11-08 11:18:07.129,13.05,92
AA00,2017-11-08 11:18:08.134,15.00,91
AA00,2017-11-08 11:18:09.142,12.56,91
AA00,2017-11-08 11:18:10.134,12.56,91
AA00,2017-11-08 11:18:11.142,13.05,91
AA00,2017-11-08 11:18:12.150,12.56,91
AA00,2017-11-08 11:18:13.158,13.05,91
AA00,2017-11-08 11:18:14.165,13.54,91
AA00,2017-11-08 11:18:15.172,13.54,91
AA00,2017-11-08 11:18:16.163,13.54,91
AA00,2017-11-08 11:18:17.170,15.00,92
AA00,2017-11-08 11:18:18.178,12.56,91
AA00,2017-11-08 11:18:19.187,14.03,91
AA00,2017-11-08 11:18:20.195,14.03,91
AA00,2017-11-08 11:18:21.186,12.56,91
AA00,2017-11-08 11:18:22.193,13.05,92
AA00,2017-11-08 11:18:23.199,13.54,91
AA00,2017-11-08 11:18:24.207,13.54,91
AA00,2017-11-08 11:18:25.216,12.56,92
AA00,2017-11-08 11:18:26.223,15.49,91
```

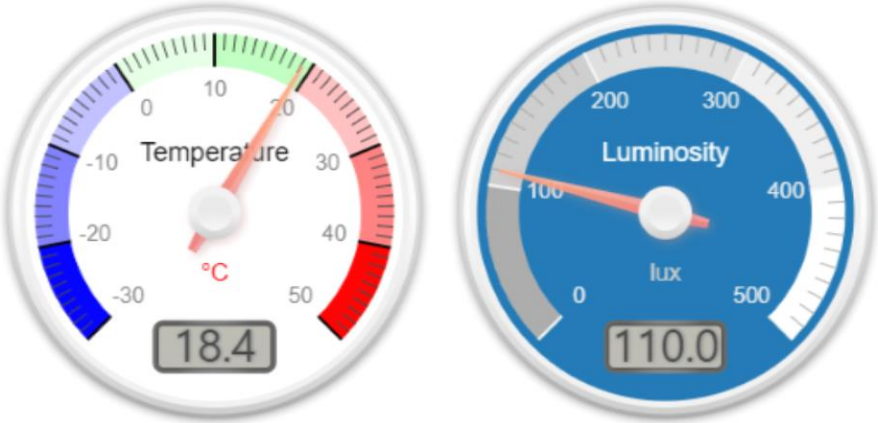
tmp36_ldr_node.js

```
1  // tmp36_ldr_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 });
```

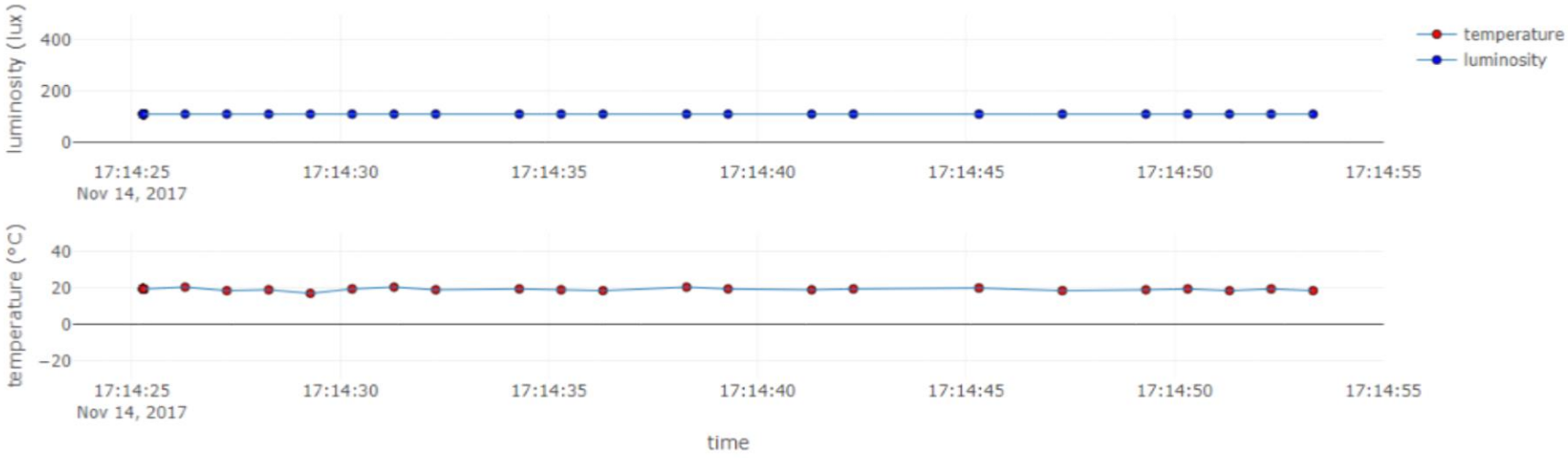
tmp36_ldr_node.js

```
19  var dStr = '';
20  var tdata = [];
21
22  sp.on('data', function (data) { // call back when data is received
23      // raw data only
24          //console.log(data);
25          dStr = getDateString();
26          tdata[0] = dStr;  // date
27          tdata[1] = data;  // data
28          console.log("AA00," + tdata);
29          io.sockets.emit('message', tdata); // send data to all clients
30  });
31
32  io.sockets.on('connection', function (socket) {
33      // If socket.io receives message from the client browser then
34      // this call back will be executed.
35      socket.on('message', function (msg) {
36          console.log(msg);
37      });
38      // If a web browser disconnects from Socket.IO then this callback i
39      socket.on('disconnect', function () {
40          console.log('disconnected');
41      });
42  });
```

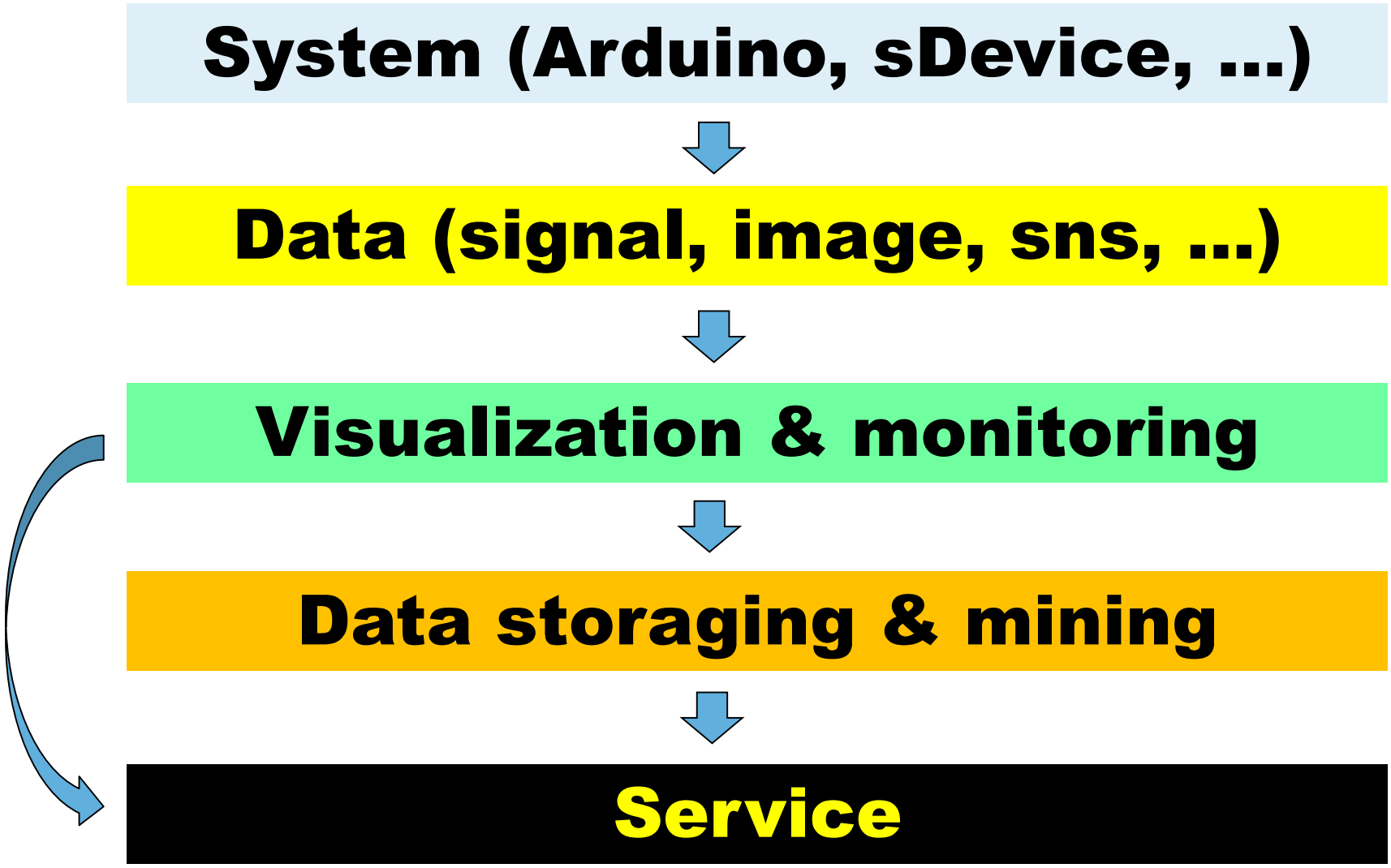
Real-time Temperature(°C) and Luminosity(lux) from sensors



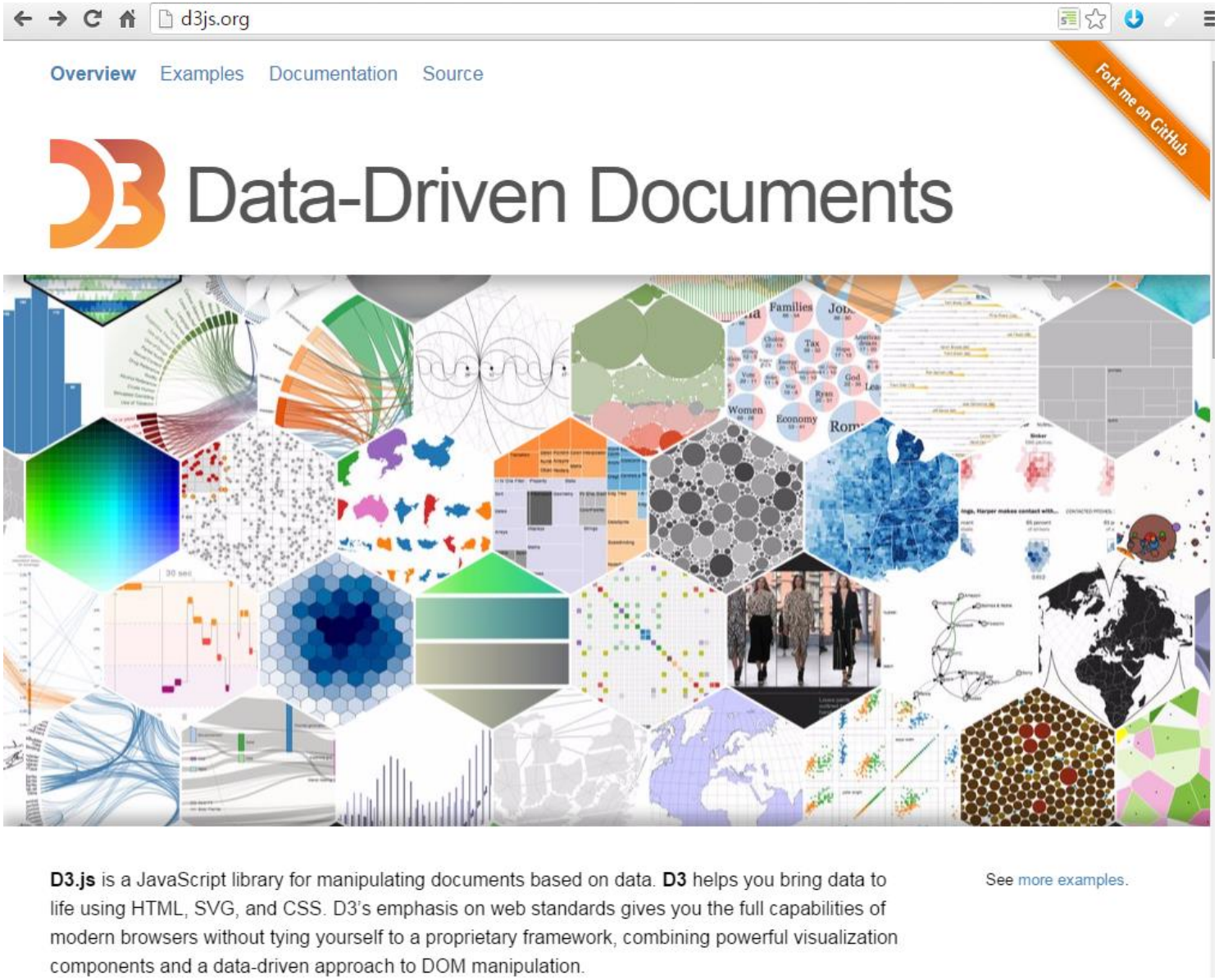
on Time: 2017-11-14 17:14:53.321



A6.1. Introduction to visualization

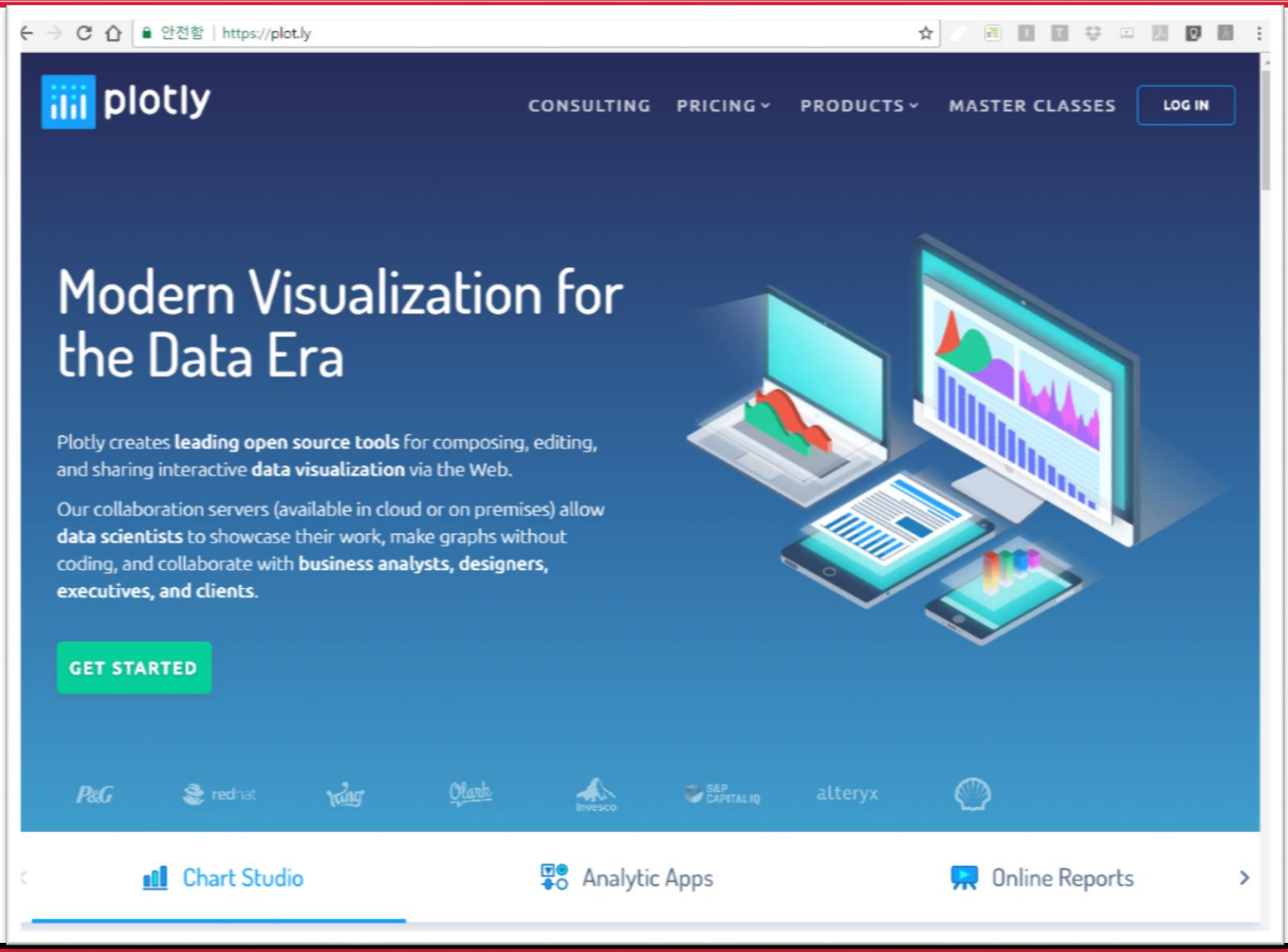


A6.1.1 Introduction to visualization



http://d3js.org/

A6.1.2 Introduction to visualization



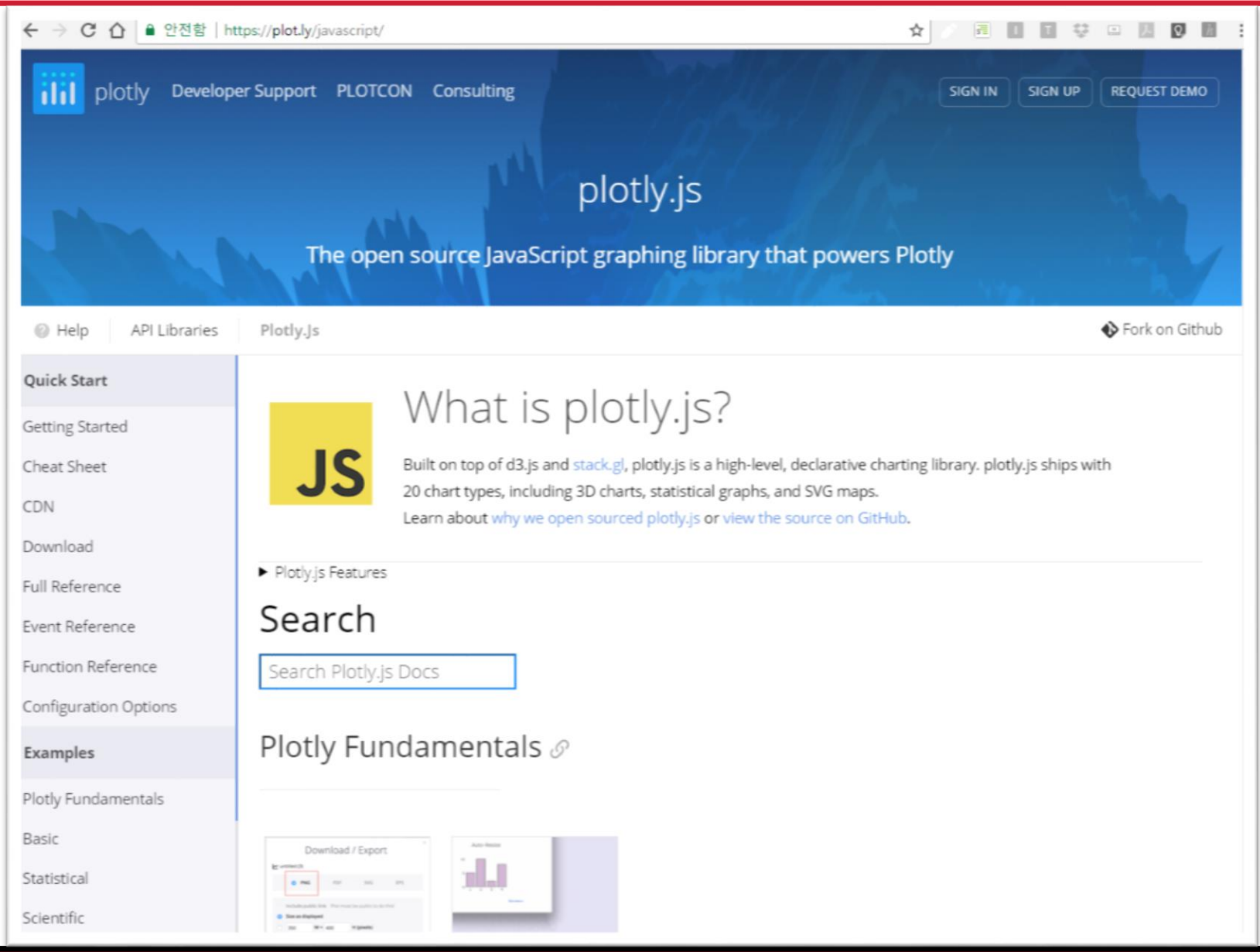
https://plot.ly/

A6.1.3 plot.ly

plotly.js is Plotly's client-side, interactive JavaScript charting library, built on top of D3.js, stack.gi, and jQuery.

https://plot.ly/javascript/

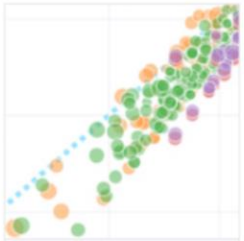
A6.1.4 Introduction to plot.ly



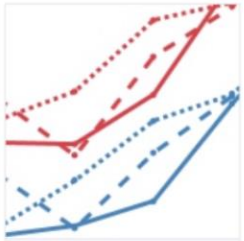
https://plot.ly/javascript/getting-started/#download

A6.1.5 Introduction to plot.ly

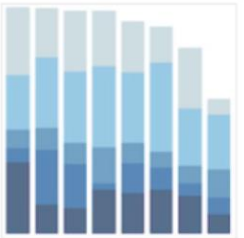
Basic Charts



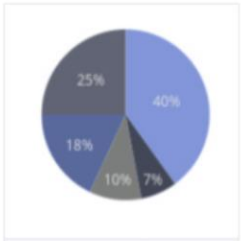
Scatter Plots



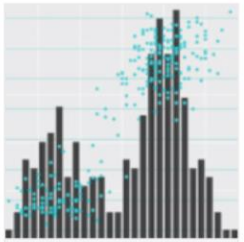
Line Charts



Bar Charts

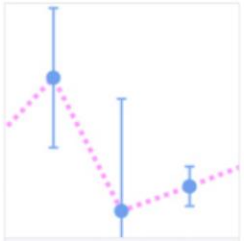


Pie Charts

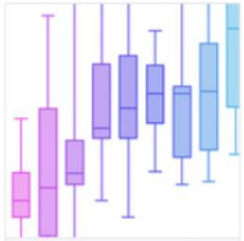


More Basic Charts

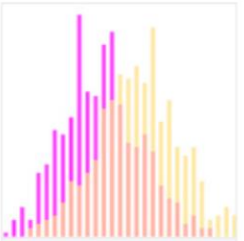
Statistical Charts



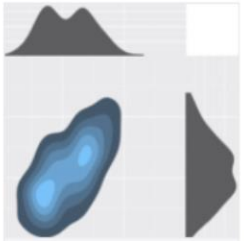
Error Bars



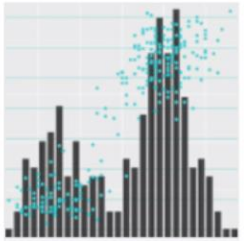
Box Plots



Histograms



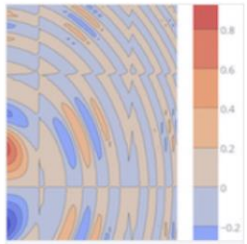
2d Density Plots



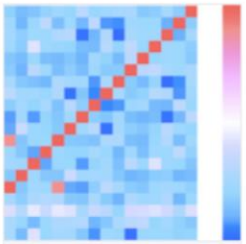
More Statistical

A6.1.6 Introduction to plot.ly

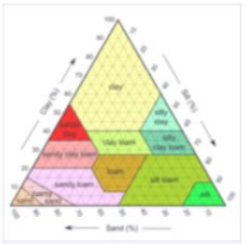
Scientific Charts



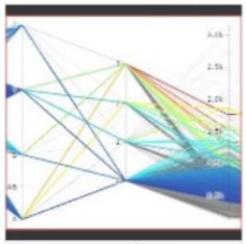
Contour Plots



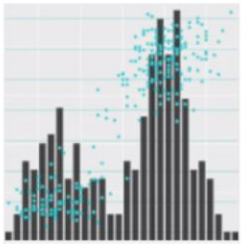
Heatmaps



Ternary Plots



Parallel Coordinates Plot

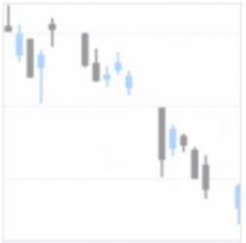


More Scientific Charts

Financial Charts

Time Series

OHLC Charts



Candlestick Charts

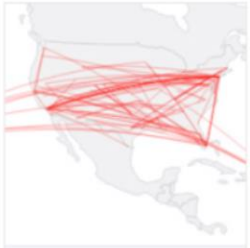
A6.1.7 Introduction to plot.ly

Maps

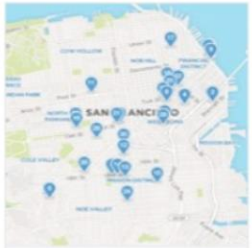
Choropleth Maps

Scatter Plots on Maps

Bubble Maps

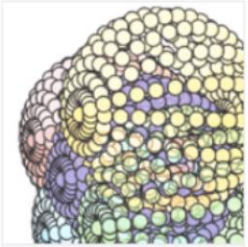


Lines on Maps

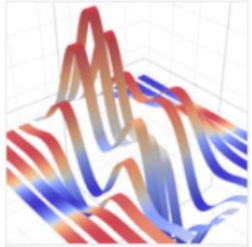


Scatter Plots on Mapbox

3D Charts

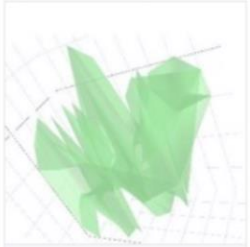


3D Scatter Plots

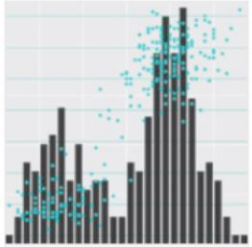


Ribbon Plots

3D Surface Plots



3D Mesh Plots



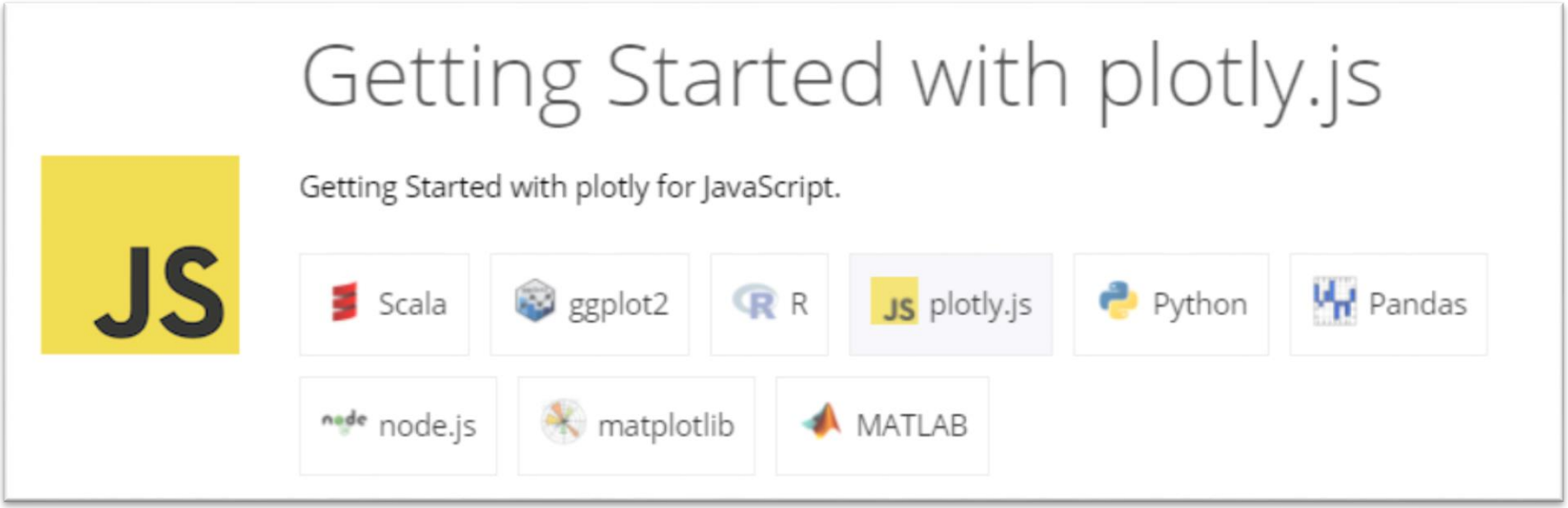
More 3D Charts

A6.1.8 Introduction to plot.ly

https://plot.ly/javascript/time-series/

https://plot.ly/javascript/streaming/

A6.1.9 Getting started: plotly.js



https://plot.ly/

A6.1.10 Getting started: plotly.js

plotly.js CDN

You can also use the ultrafast plotly.js CDN link. This CDN is graciously provided by the incredible team at Fastly.

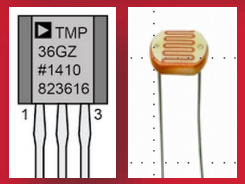
```
<head>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
```

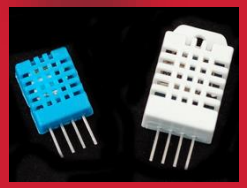
Else, if you want to get a specific version of plotly.js, say 1.2.0:

```
<head>
    <script src="https://cdn.plot.ly/plotly-1.2.0.min.js"></script>
</head>
```

<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>

https://plot.ly/javascript/getting-started/





Data visualization using ploy.ly

A6.2 Getting started: plotly.js

Navigation

- Basic Line Plot
- Line and Scatter Plot
- Adding Names to Line and Scatter Plot
- Line and Scatter Styling
- Styling Line Plot
- Colored and Styled Scatter Plot
- Line Shape Options for Interpolation
- Graph and Axes Titles
- Line Dash
- Connect Gaps Between Data
- Labelling Lines with Annotations

← Back To Plotly.Js

Line Charts in plotly.js

How to make D3.js-based line charts in JavaScript.

R | plotly.js | Python | Pandas

Basic Line Plot

```
var trace1 = {
  x: [1, 2, 3, 4],
  y: [10, 15, 13, 17],
  type: 'scatter'
};

var trace2 = {
  x: [1, 2, 3, 4],
  y: [16, 5, 11, 9],
  type: 'scatter'
};
```

A6.2.1 Getting started: plotly.js

Hello plotly data!

```
<html>
<head>
    <meta charset="utf-8">
    <!-- Plotly.js -->
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
    <h1>Data visualization by AAnn</h1>
    <hr>
    <h2>Starting graph by AAnn</h2>

    <!-- Plotly chart will be drawn inside this DIV -->
    <div id="myDiv" style="width: 500px;height: 400px"></div>
    <hr>

    <script>
        <!-- JAVASCRIPT CODE GOES HERE -->
        //var myPlot = document.getElementById('myDiv');

        var data = [
        {
            x: [1, 2, 3, 4, 5],
            y: [1, 2, 4, 8, 16],
            type: 'scatter'
        }];

        Plotly.newPlot('myDiv', data);

    </script>
</body>
</html>
```

[Tip] Using WEB browser in SB text3

[Tool] Sublime Text – 현재 작업 중인 파일을 웹브라우저로 열기

1. Tool -> New Plugin을 실행 한 후 아래 내용으로 덮어 씌운 후 **'open_browser'**으로 저장한다**.**

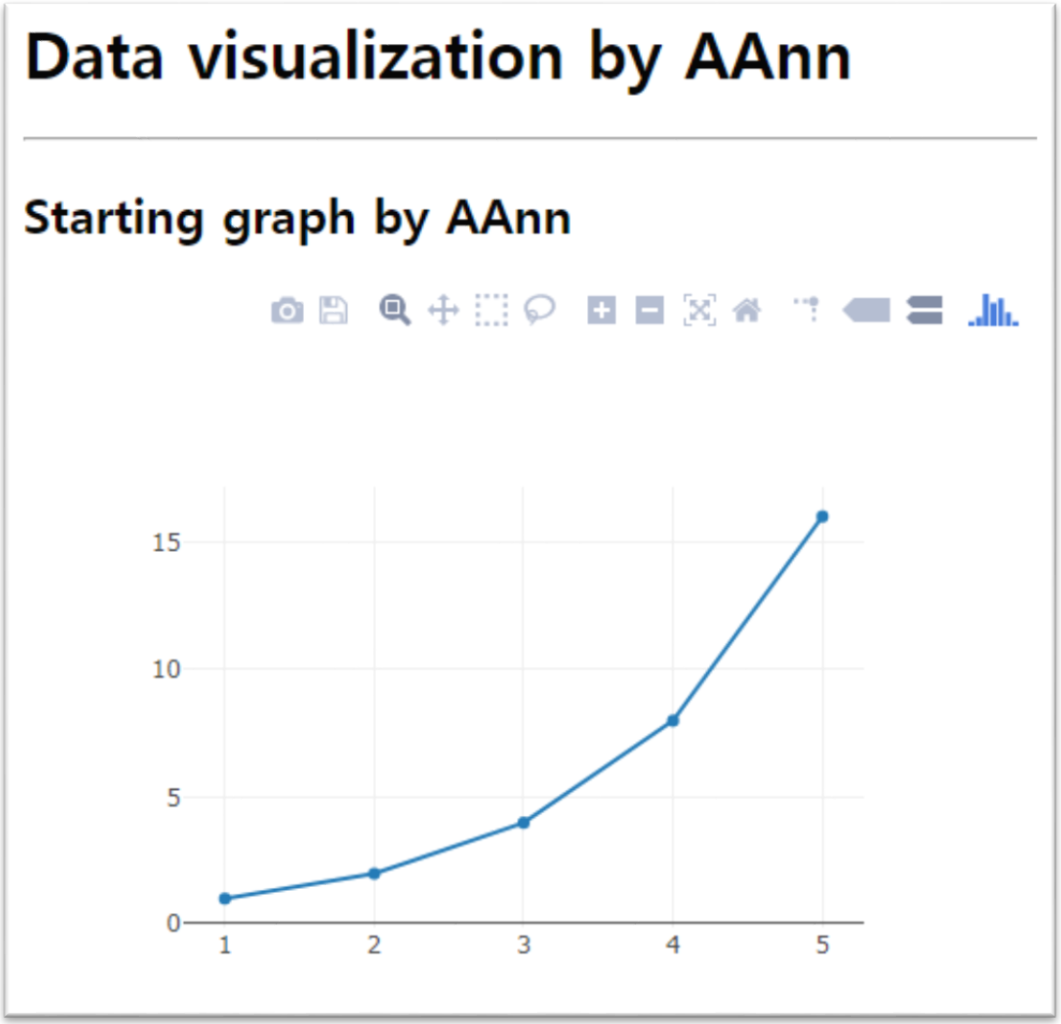
```
import sublime, sublime_plugin
import webbrowser

class OpenBrowserCommand(sublime_plugin.TextCommand):
    def run(self,edit):
        url = self.view.file_name()
        webbrowser.open_new(url)
```

2. Preferences -> Key Bindings - User로 이동한 후 단축키를 할당한다**.**

```
{ "keys": ["f10"], "command": "open_browser" }
```

A6.2.2 Getting started: plotly.js

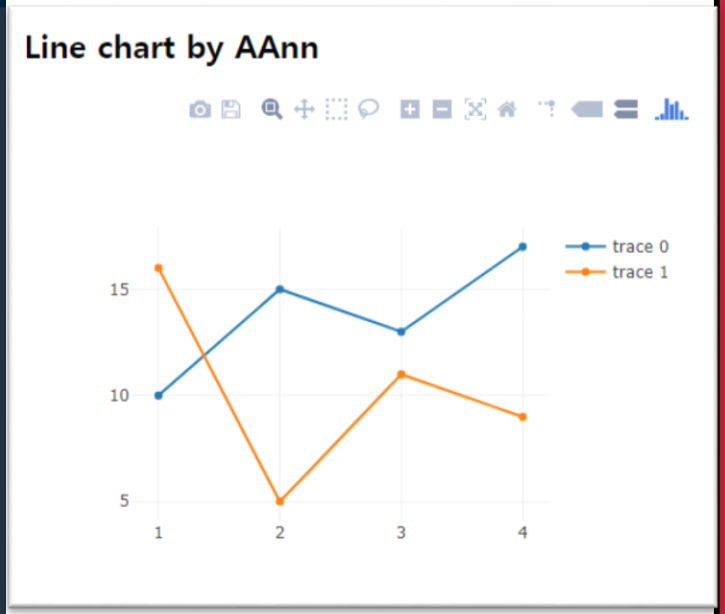
Graph : Hello plotly data!



A6.2.3 plotly.js: Line Charts

[1] Basic line charts

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

    var trace1 = {
        x: [1, 2, 3, 4],
        y: [10, 15, 13, 17],
        type: 'scatter'
    };

    var trace2 = {
        x: [1, 2, 3, 4],
        y: [16, 5, 11, 9],
        type: 'scatter'
    };

    var data = [trace1, trace2];

    Plotly.newPlot('myDiv', data);

</script>
```

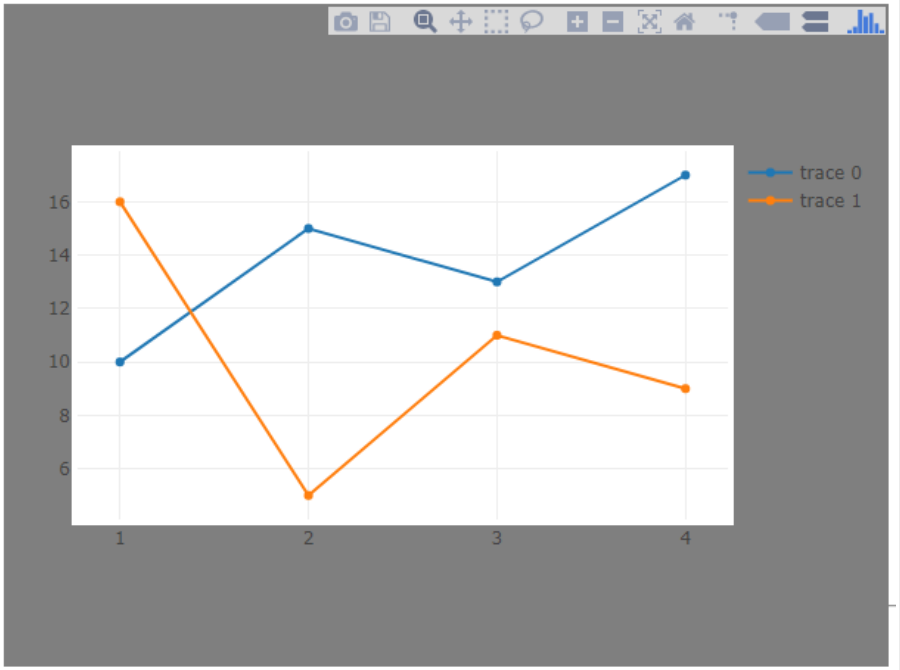


A6.2.4 plotly.js: Line Charts

[2] Basic line charts with layout

```
var layout = {
    autosize: false,
    width: 600,
    height: 450,
    margin: {
        l: 50,
        r: 50,
        b: 100,
        t: 100,
        pad: 4
    },
    paper_bgcolor: '#7f7f7f',
    plot_bgcolor: '#cacaca' //'#rrggrbb'
};

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

Line chart by AAnn



AAnn_Chart_Layout.png

A6.2.5 plotly.js: Line & Scatter plot

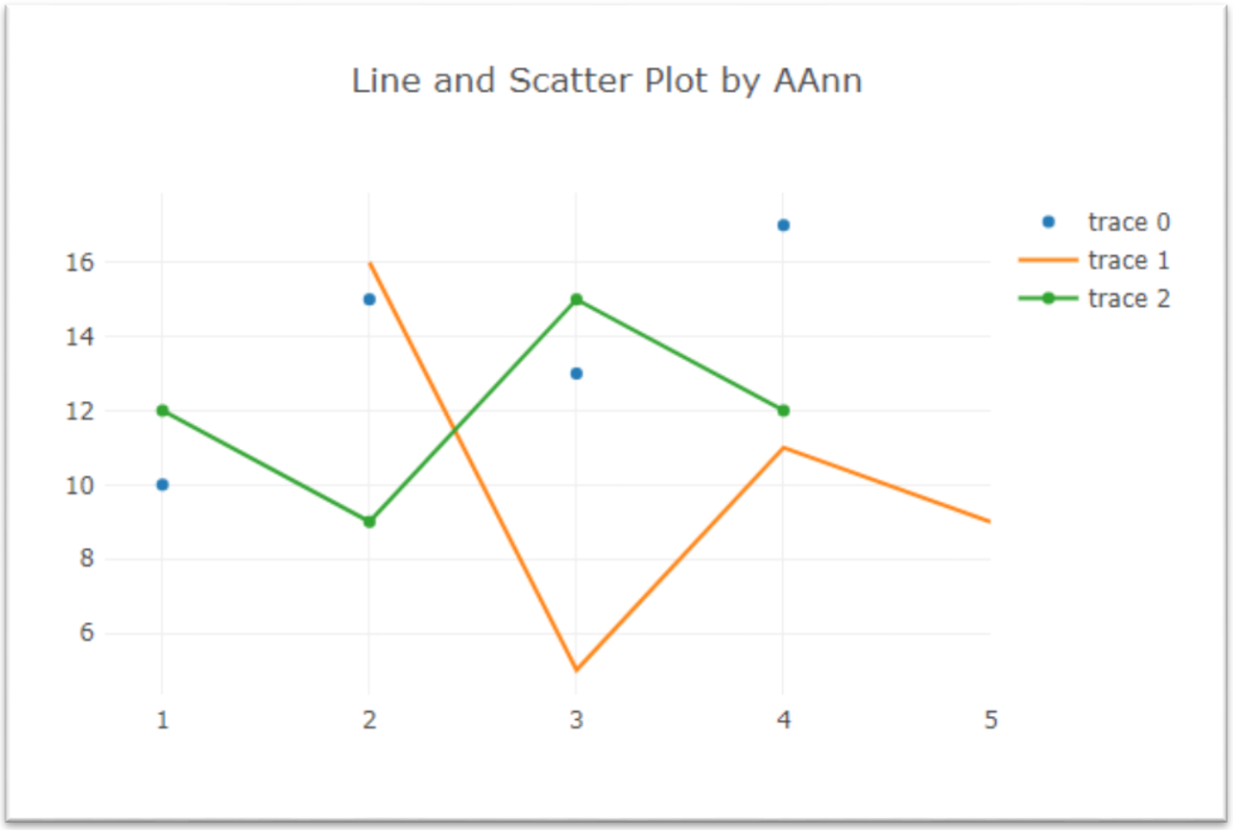
[3] Line & scatter plot

```
var trace1 = {
    x: [1, 2, 3, 4],
    y: [10, 15, 13, 17],
    mode: 'markers'
};

var trace2 = {
    x: [2, 3, 4, 5],
    y: [16, 5, 11, 9],
    mode: 'lines'
};

var trace3 = {
    x: [1, 2, 3, 4],
    y: [12, 9, 15, 12],
    mode: 'lines+markers'
};

var data = [ trace1, trace2, trace3 ];
```

```
var layout = {
    title:'Line and Scatter Plot',
    width: 600,
    height: 450,
    margin: {
        l: 50,
        r: 50,
        b: 100,
        t: 100,
        pad: 4
    },
};

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

A6.2.5 plotly.js: Line & Scatter plot

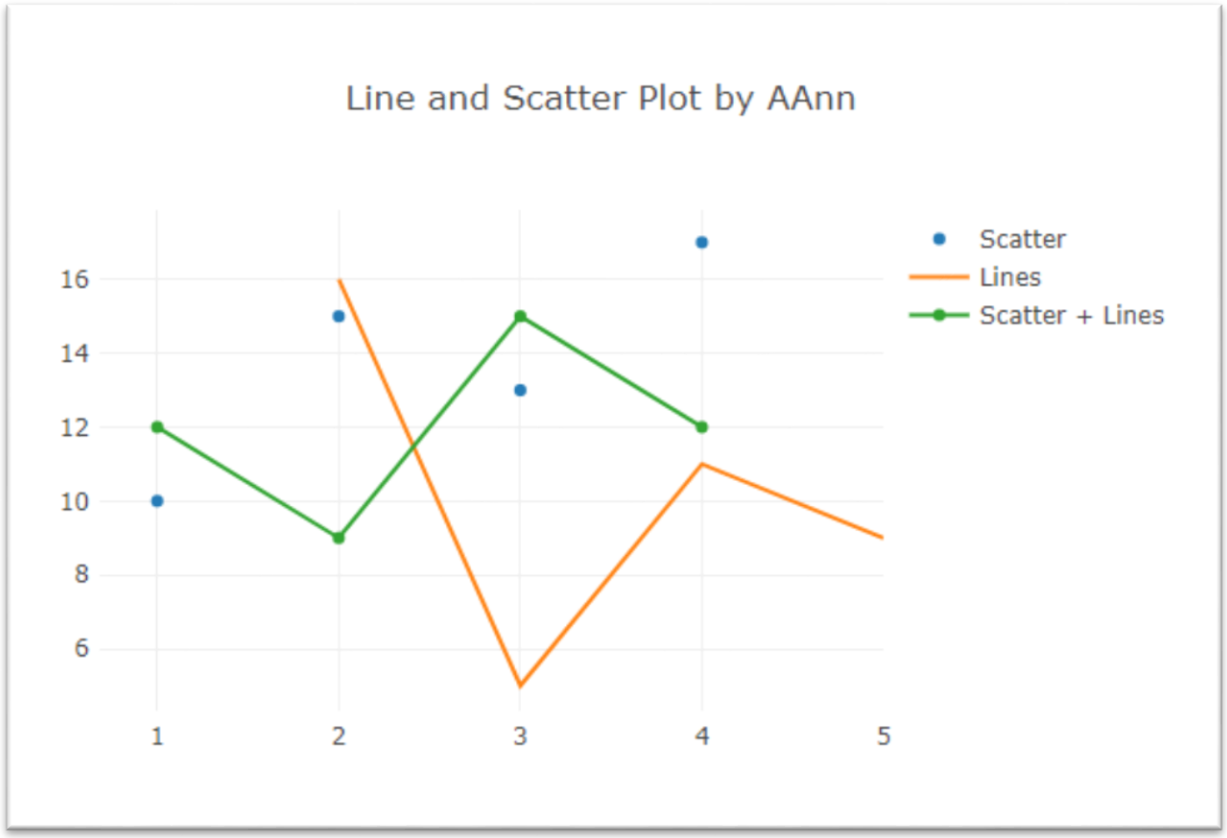
[3.1] Line & scatter plot



A6.2.6 plotly.js: Line & Scatter plot

[3.2] Line & scatter plot with axis name

```
var trace1 = {
  x: [1, 2, 3, 4],
  y: [10, 15, 13, 17],
  mode: 'markers',
  name: 'Scatter'
};

var trace2 = {
  x: [2, 3, 4, 5],
  y: [16, 5, 11, 9],
  mode: 'lines',
  name: 'Lines'
};

var trace3 = {
  x: [1, 2, 3, 4],
  y: [12, 9, 15, 12],
  mode: 'lines+markers',
  name: 'Scatter + Lines'
};
```

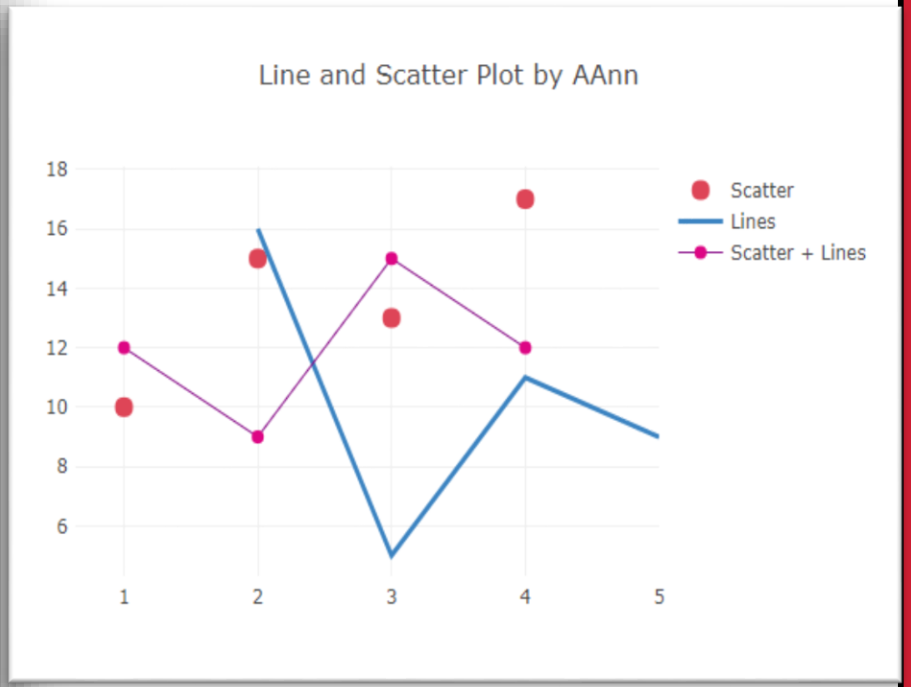


A6.2.7 plotly.js: Line & Scatter plot

[3.3] Line & scatter plot with style

```
var trace1 = {
  x: [1, 2, 3, 4],
  y: [10, 15, 13, 17],
  mode: 'markers',
  name: 'Scatter',
  marker: {
    color: 'rgb(219, 64, 82)',
    size: 12
  }
};

var trace2 = {
  x: [2, 3, 4, 5],
  y: [16, 5, 11, 9],
  mode: 'lines',
  name: 'Lines',
  line: {
    color: 'rgb(55, 128, 191)',
    width: 3
  }
};
```

```
var trace3 = {
  x: [1, 2, 3, 4],
  y: [12, 9, 15, 12],
  mode: 'lines+markers',
  name: 'Scatter + Lines',
  marker: {
    color: 'rgb(128, 0, 128)',
    size: 8
  },
  line: {
    color: 'rgb(128, 0, 128)',
    width: 1
  }
};
```

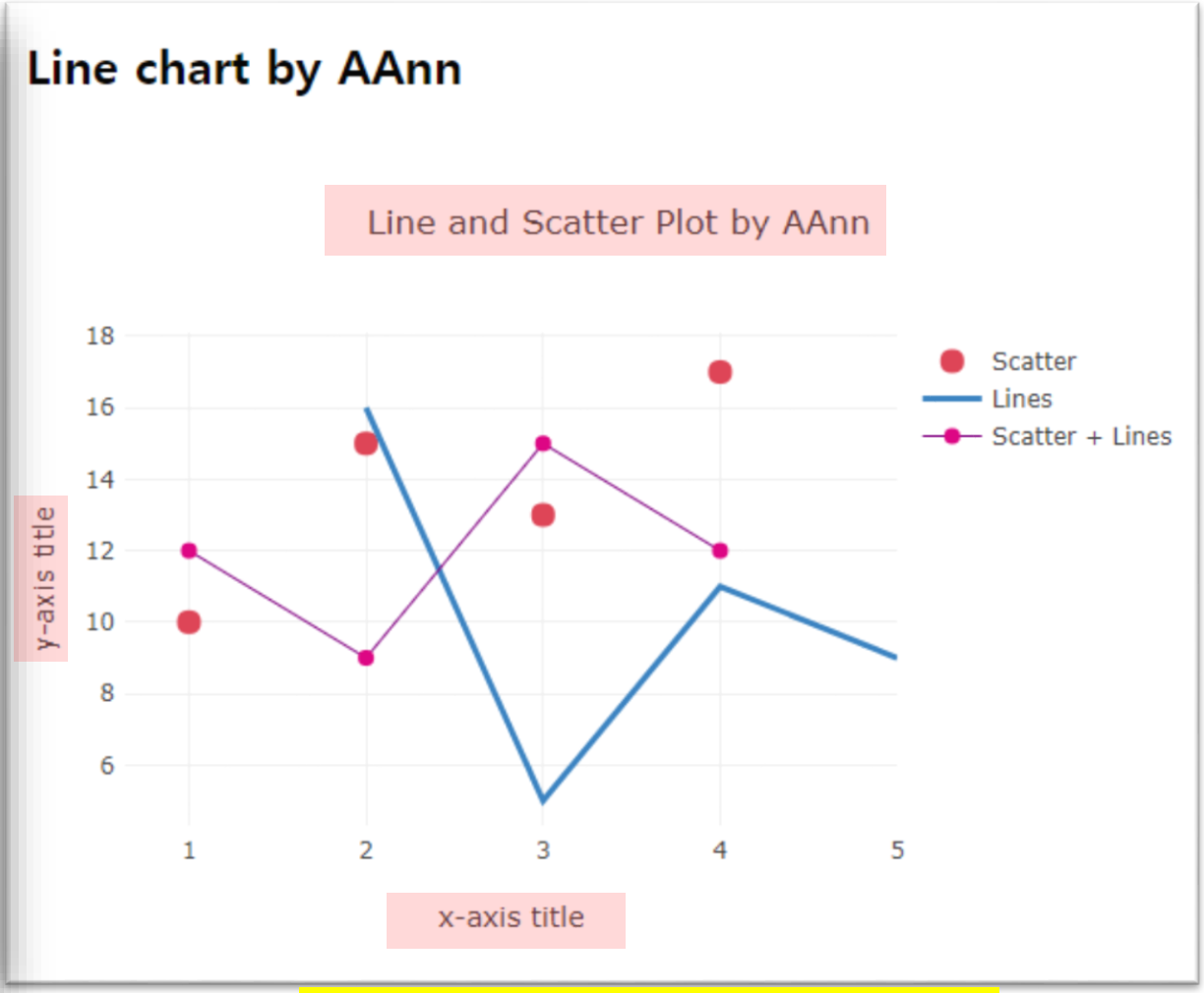


AAnn_Plot_Style.png

A6.2.8 plotly.js: Line & Scatter plot

[3.4] Line & scatter plot with axis titles

```
var layout = {
  title:'Line and Scatter Plot',
  width: 600, height: 450,
  margin: {
    l: 50,
    r: 50,
    b: 100,
    t: 100,
    pad: 4
  },
  xaxis: {
    title: 'x-axis title'
  },
  yaxis: {
    title: 'y-axis title'
  }
};
```



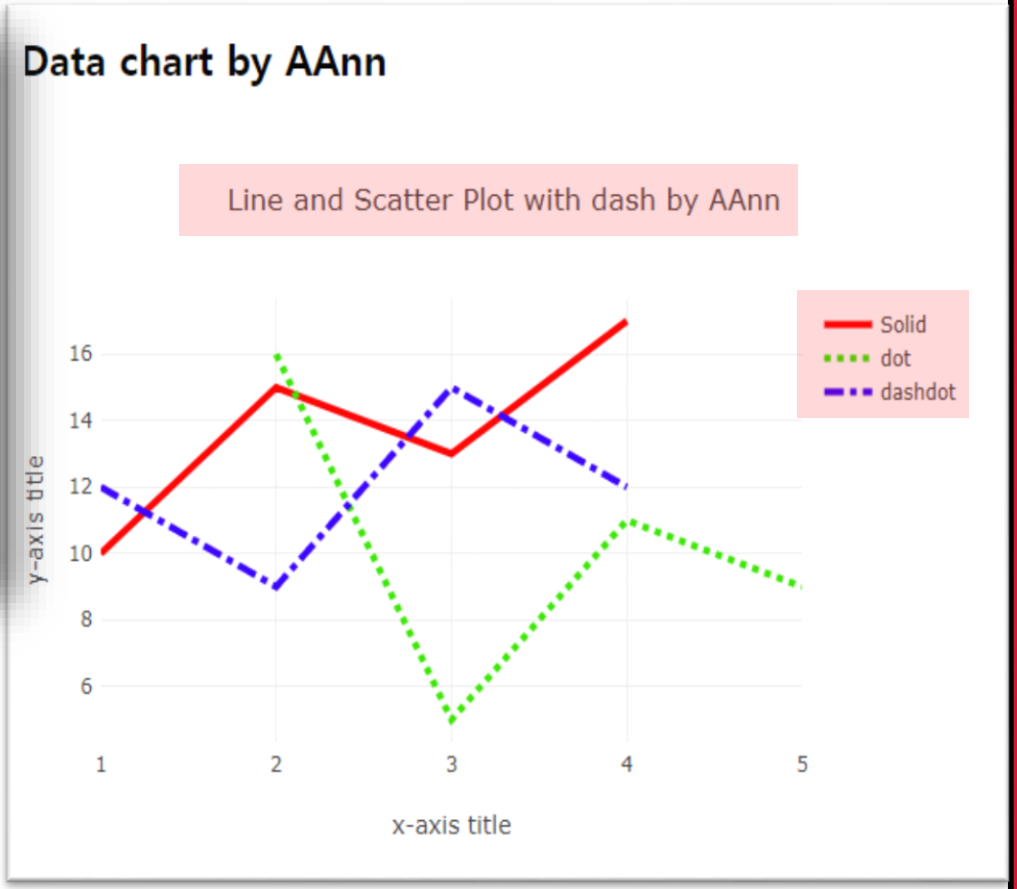
AAnn_Axis_Title.png

A6.2.9 plotly.js: Line & Scatter plot

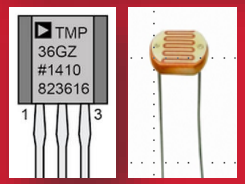
[3.5] Line & scatter plot with dash and dot

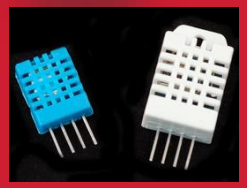
```
var trace1 = {
  x: [1, 2, 3, 4],
  y: [10, 15, 13, 17],
  mode: 'lines',
  name: 'Solid',
  line: {
    color: 'rgb(255, 0, 0)',
    dash: 'solid',
    width: 4
  }
};

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
  }
};
```

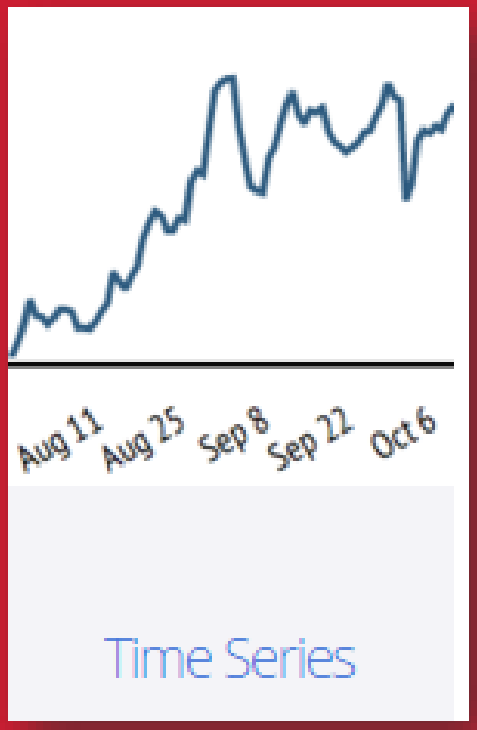


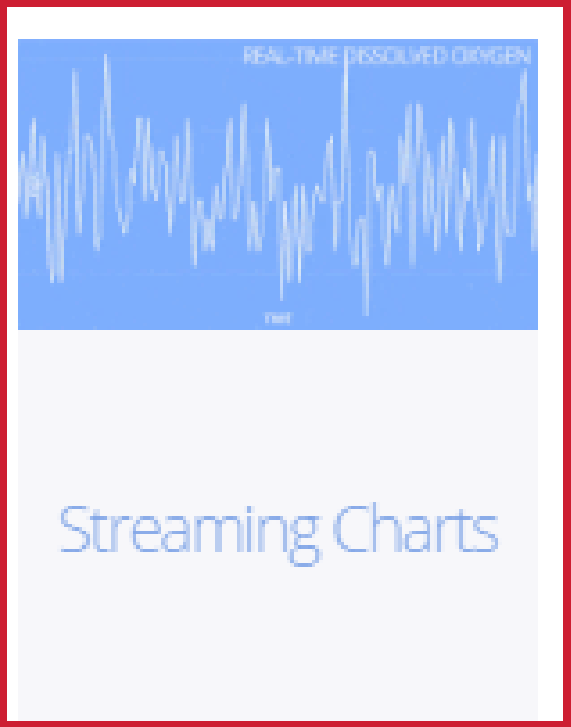
AAnn_Line_Dash.png





Data visualization using ploy.ly





A6.3. Time series

Navigation

- Date Strings
- Basic Time Series
- Manually Set Range
- Time Series with Rangeslider
- Back To Plotly.Js

Time Series in plotly.js

How to plot D3.js-based date and time in Plotly.js. An example of a time-series plot.

R | Python | matplotlib | plotly.js | Pandas | node.js | MATLAB

Date Strings

```
var data = [
  {
    x: ['2013-10-04 22:23:00', '2013-11-04 22:23:00', '2013-12-04 22:23:00'],
    y: [1, 3, 6],
    type: 'scatter'
  }
];

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

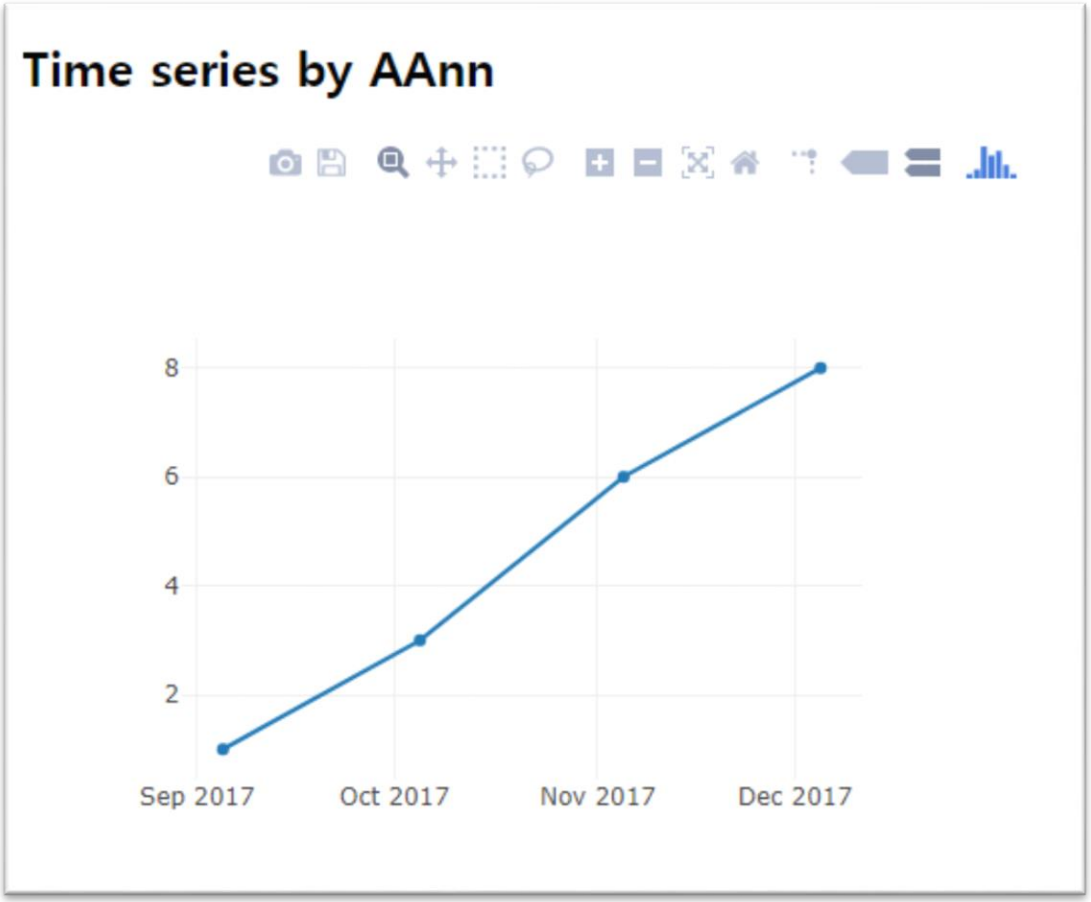
A6.3.1 plotly.js: Time series

[1] Time series : data strings

```
<!-- Plotly chart will be drawn inside this DIV -->
<div id="myDiv" style="width: 500px;height: 400px"></div>

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

    var data = [
    {
        x: ['2017-9-04 22:23:00',
        '2017-10-04 22:23:00',
        '2017-11-04 22:23:00',
        '2017-12-04 22:23:00'],
        y: [1, 3, 6, 8],
        type: 'scatter'
    }
    ];

    Plotly.newPlot('myDiv', data);

</script>
```

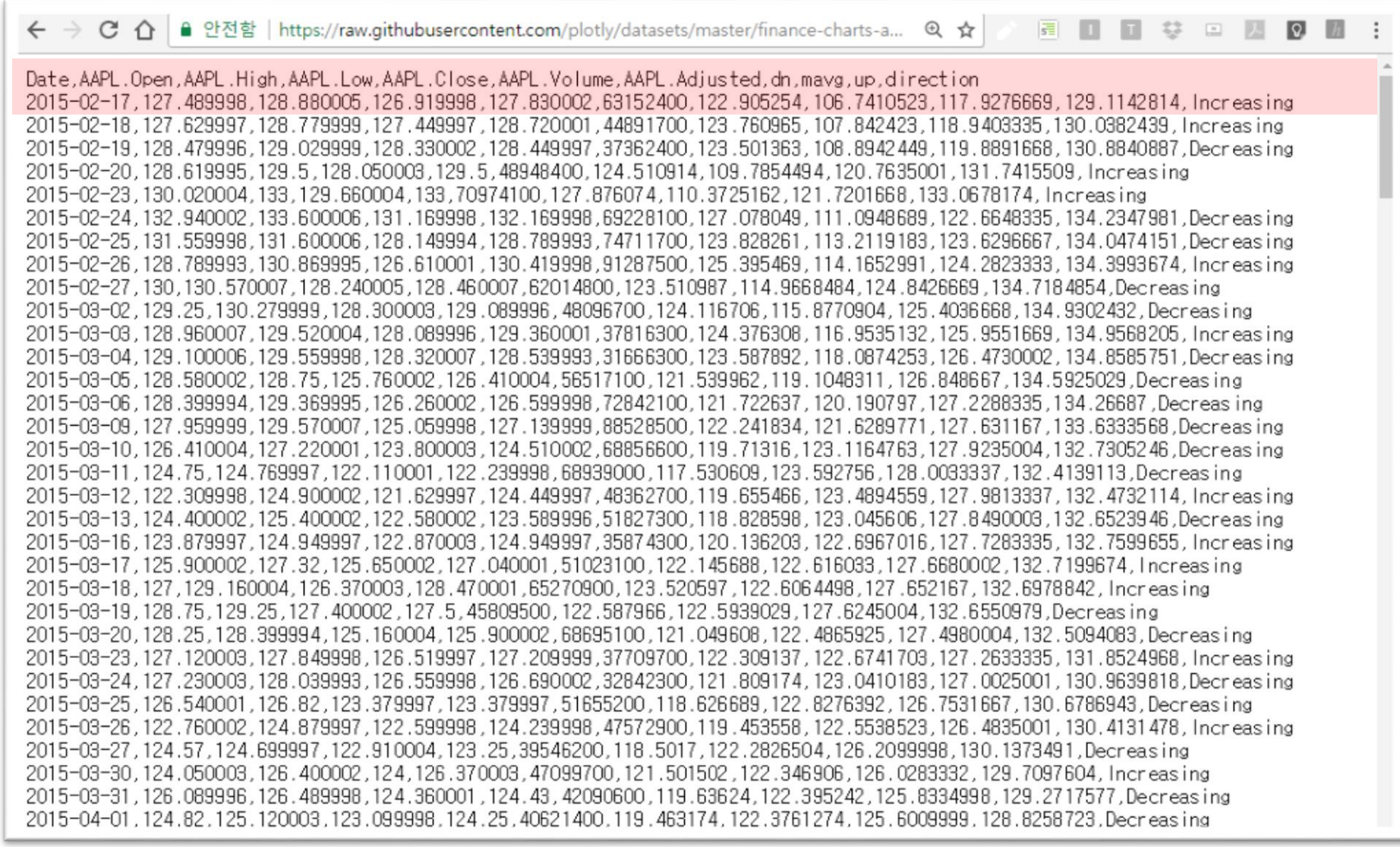
A6.3.2 plotly.js: Time series

Time series : data strings – result



A6.3.3 plotly.js: Time series

[2] Time series : financial data strings – AAPL stock price




```
Date,AAPL.Open,AAPL.High,AAPL.Low,AAPL.Close,AAPL.Volume,AAPL.Adjusted,dn,mavg,up,direction
2015-02-17,127.489998,128.880005,126.919998,127.830002,63152400,122.905254,106.7410523,117.9276669,129.1142814,Increasing
2015-02-18,127.629997,128.779999,127.449997,128.720001,44891700,123.760965,107.842423,118.9403335,130.0382439,Increasing
2015-02-19,128.479996,129.029999,128.330002,128.449997,37362400,123.501363,108.8942449,119.8891668,130.8840887,Decreasing
2015-02-20,128.619995,129.5,128.050003,129.5,48948400,124.510914,109.7854494,120.7635001,131.7415509,Increasing
2015-02-23,130.020004,133,129.660004,133,70974100,127.876074,110.3725162,121.7201668,133.0678174,Increasing
2015-02-24,132.940002,133.600006,131.169998,132.169998,69228100,127.078049,111.0948689,122.6648335,134.2347981,Decreasing
2015-02-25,131.559998,131.600006,128.149994,128.789993,74711700,123.828261,113.2119183,123.6296667,134.0474151,Decreasing
2015-02-26,128.789993,130.869995,126.610001,130.419998,91287500,125.395469,114.1652991,124.2823333,134.3993674,Increasing
2015-02-27,130,130.570007,128.240005,128.460007,62014800,123.510987,114.9668484,124.8426669,134.7184854,Decreasing
2015-03-02,129.25,130.279999,128.300003,129.089996,48096700,124.116706,115.8770904,125.4036668,134.9302432,Decreasing
2015-03-03,128.960007,129.520004,128.089996,129.360001,37816300,124.376308,116.9535132,125.9551669,134.9568205,Increasing
2015-03-04,129.100006,129.559998,128.320007,128.539993,31666300,123.587892,118.0874253,126.4730002,134.8585751,Decreasing
2015-03-05,128.580002,128.75,125.760002,126.410004,56517100,121.539962,119.1048311,126.848667,134.5925029,Decreasing
2015-03-06,128.399994,129.369995,126.260002,126.599998,72842100,121.722637,120.190797,127.2288335,134.26687,Decreasing
2015-03-09,127.959999,129.570007,125.059998,127.139999,88528500,122.241834,121.6289771,127.631167,133.6333568,Decreasing
2015-03-10,126.410004,127.220001,123.800003,124.510002,68856600,119.71316,123.1164763,127.9235004,132.7305246,Decreasing
2015-03-11,124.75,124.769997,122.110001,122.239998,68939000,117.530609,123.592756,128.0033337,132.4139113,Decreasing
2015-03-12,122.309998,124.900002,121.629997,124.449997,48362700,119.655466,123.4894559,127.9813337,132.4732114,Increasing
2015-03-13,124.400002,125.400002,122.580002,123.589996,51827300,118.828598,123.045606,127.8490003,132.6523946,Decreasing
2015-03-16,123.879997,124.949997,122.870003,124.949997,35874300,120.136203,122.6967016,127.7283335,132.7599655,Increasing
2015-03-17,125.900002,127.32,125.650002,127.040001,51023100,122.145688,122.616033,127.6680002,132.7199674,Increasing
2015-03-18,127,129.160004,126.370003,128.470001,65270900,123.520597,122.6064498,127.652167,132.6978842,Increasing
2015-03-19,128.75,129.25,127.400002,127.5,45809500,122.587966,122.5939029,127.6245004,132.6550979,Decreasing
2015-03-20,128.25,128.399994,125.160004,125.900002,68695100,121.049608,122.4865925,127.4980004,132.5094083,Decreasing
2015-03-23,127.120003,127.849998,126.519997,127.209999,37709700,122.309137,122.6741703,127.2633335,131.8524968,Increasing
2015-03-24,127.230003,128.039993,126.559998,126.690002,32842300,121.809174,123.0410183,127.0025001,130.9639818,Decreasing
2015-03-25,126.540001,126.82,123.379997,123.379997,51655200,118.626689,122.8276392,126.7531667,130.6786943,Decreasing
2015-03-26,122.760002,124.879997,122.599998,124.239998,47572900,119.453558,122.5538523,126.4835001,130.4131478,Increasing
2015-03-27,124.57,124.699997,122.910004,123.25,39546200,118.5017,122.2826504,126.2099998,130.1373491,Decreasing
2015-03-30,124.050003,126.400002,124,126.370003,47099700,121.501502,122.346906,126.0283332,129.7097604,Increasing
2015-03-31,126.089996,126.489998,124.360001,124.43,42090600,119.63624,122.395242,125.8334998,129.2717577,Decreasing
2015-04-01,124.82,125.120003,123.099998,124.25,40621400,119.463174,122.3761274,125.6009999,128.8258723,Decreasing
```

A6.3.4 plotly.js: Time series

[2] Time series : financial data strings – AAPL stock price

```
Plotly.d3.csv("https://raw.githubusercontent.com/plotly/datasets/master/
    finance-charts-apple.csv", function(err, rows){

    function unpack(rows, key) {
        return rows.map(function(row) { return row[key]; });
    }


    var trace1 = {
        type: "scatter",
        mode: "lines",
        name: 'AAPL High',
        x: unpack(rows, 'Date'),
        y: unpack(rows, 'AAPL.High'),
        line: {color: '#17BECF'}
    }

    var trace2 = {
        type: "scatter",
        mode: "lines",
        name: 'AAPL Low',
        x: unpack(rows, 'Date'),
        y: unpack(rows, 'AAPL.Low'),
        line: {color: '#7F7F7F'}
    }

    var data = [trace1,trace2];
```

A6.3.5 plotly.js: Time series

[2] Time series : financial data strings – AAPL stock price

```
var data = [trace1,trace2];

var layout = {
    title: 'AAPL Price Time Series',
};

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

A6.3.6 plotly.js: Time series

[2] Time series : financial data strings – set range

```
var data = [trace1,trace2];

var layout = {
    title: 'AAPL Price Time Series',
    xaxis: {
        range: ['2016-07-01', '2016-12-31'],
        type: 'date'
    },
    yaxis: {
        autorange: true,
        range: [86.8700008333, 138.870004167],
        type: 'linear'
    }
};

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

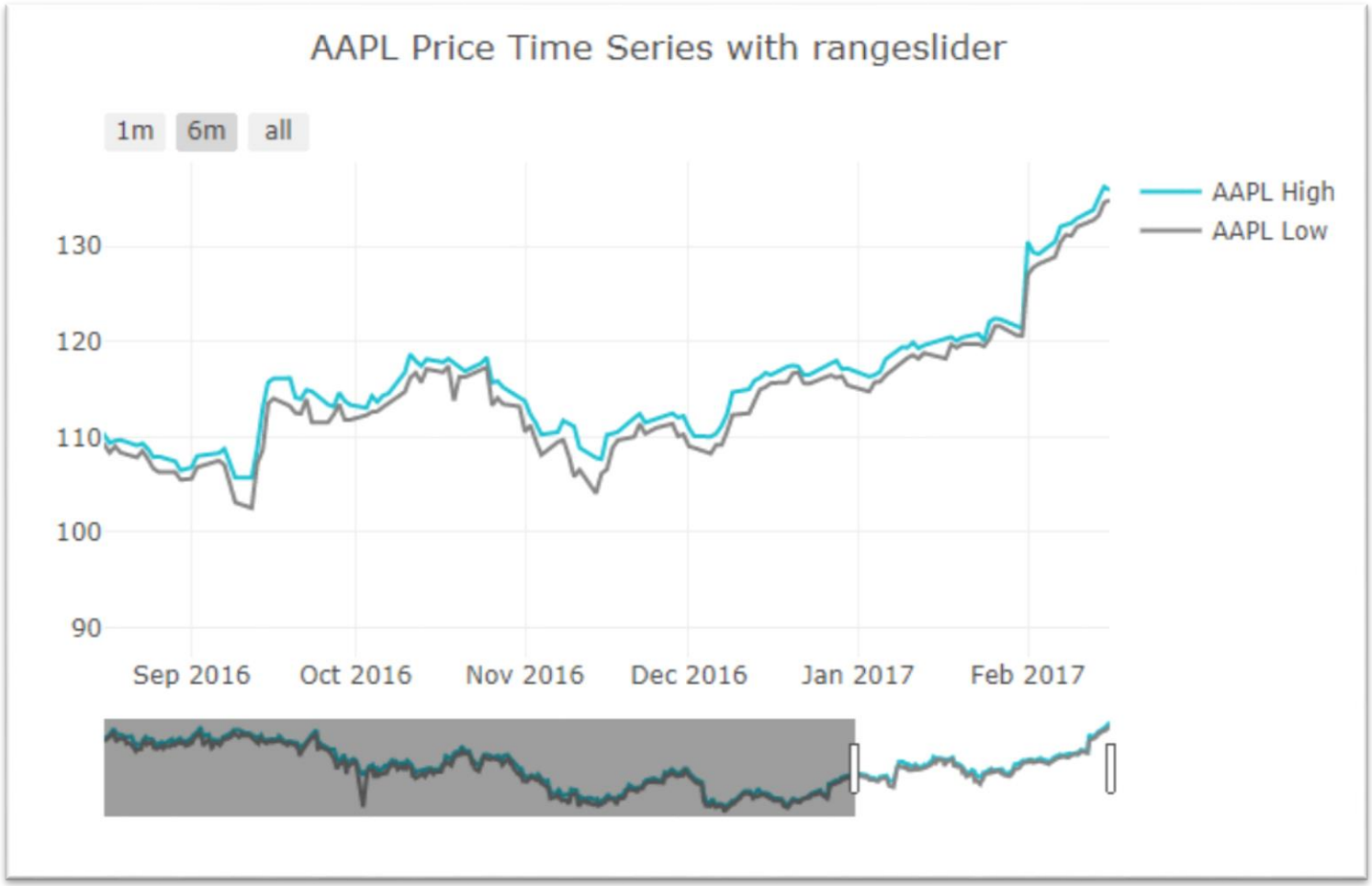
A6.3.7 plotly.js: Time series

[2] Time series : financial data strings – Range slider

```
var layout = {
    title: 'AAPL Price Time Series with rangeslider',
    xaxis: {
        autorange: true,
        range: ['2015-02-17', '2017-02-16'],
        rangeselector: {buttons: [
            {
                count: 1,
                label: '1m',
                step: 'month',
                stepmode: 'backward'
            },
            {
                count: 6,
                label: '6m',
                step: 'month',
                stepmode: 'backward'
            },
            {step: 'all'}
            ]},
            rangeslider: {range: ['2015-02-17', '2017-02-16']},
            type: 'date'
        },
        yaxis: {
            autorange: true,
            range: [86.8700008333, 138.870004167],
            type: 'linear'
        }
    };
```

A6.3.8 plotly.js: Time series

[2] Time series : financial data strings –range slider

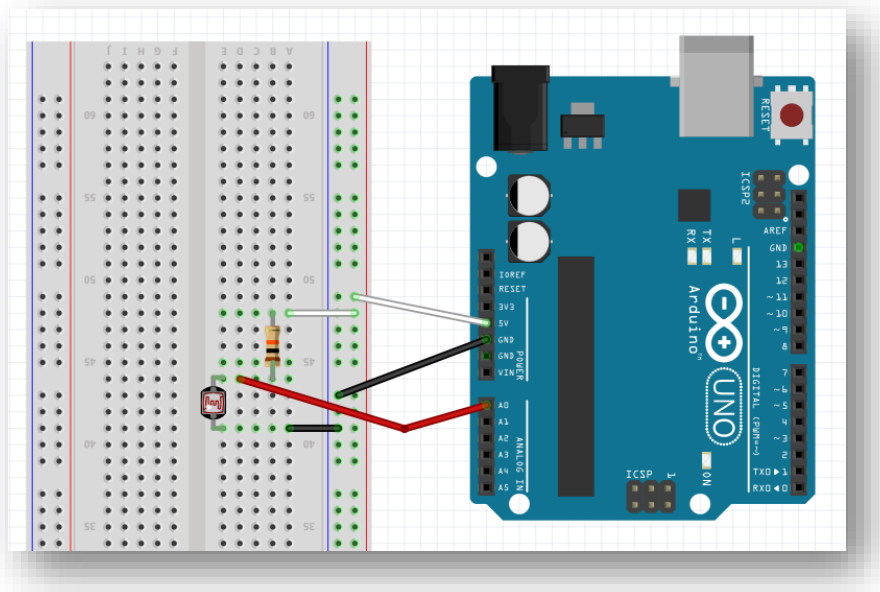


A6.3.9 plotly.js: Time series

[3] Time series : my lux data

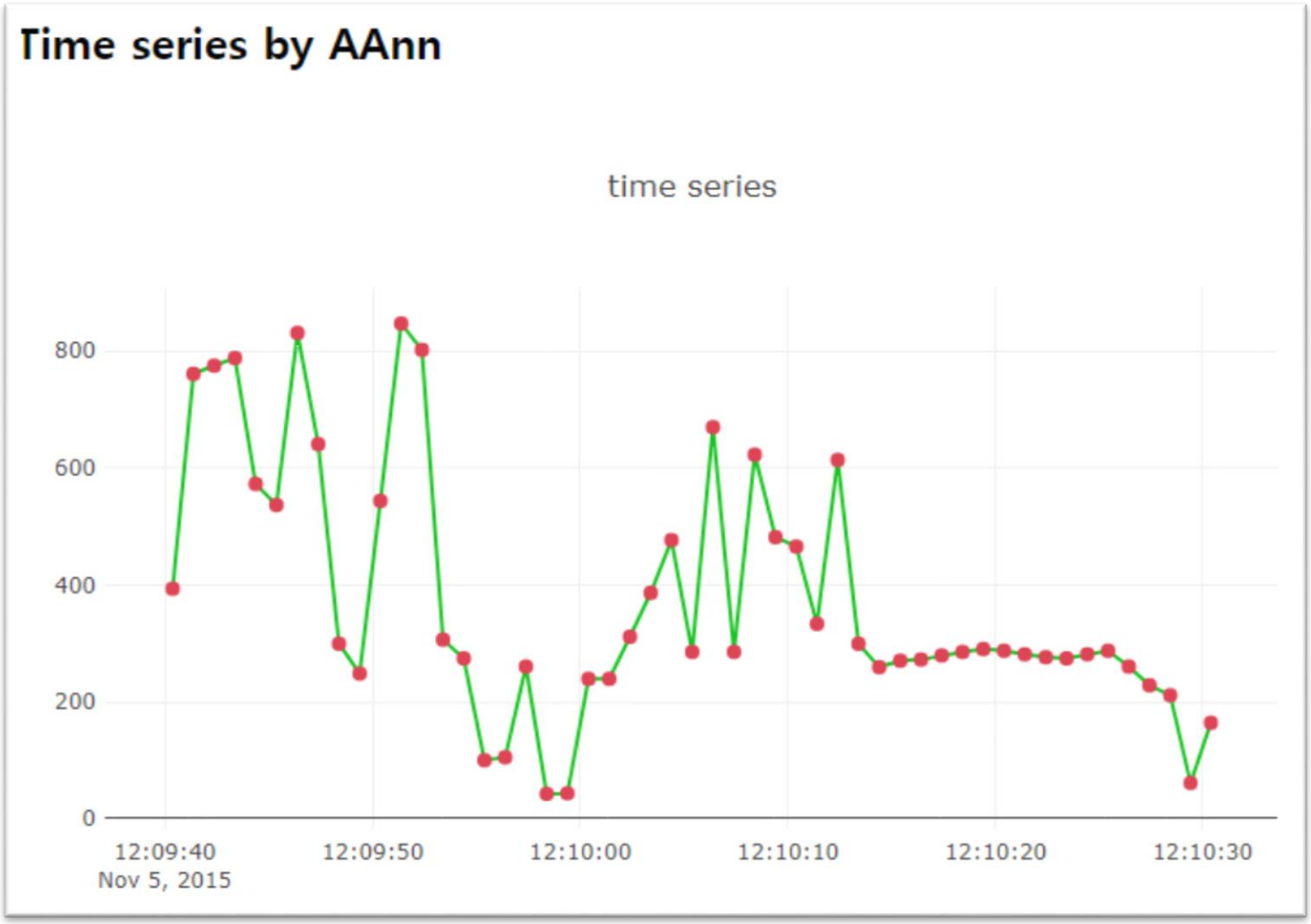
```
'2015-11-05 12:09:41.382',
'2015-11-05 12:09:42.380',
'2015-11-05 12:09:43.378',
'2015-11-05 12:09:44.377',
'2015-11-05 12:09:45.375',
'2015-11-05 12:09:46.389',
'2015-11-05 12:09:47.388',
'2015-11-05 12:09:48.386',
'2015-11-05 12:09:49.384',
'2015-11-05 12:09:50.383',
'2015-11-05 12:09:51.381',
'2015-11-05 12:09:52.380',
'2015-11-05 12:09:53.394',
'2015-11-05 12:09:54.392',
'2015-11-05 12:09:55.391',
'2015-11-05 12:09:56.389',
'2015-11-05 12:09:57.387',
'2015-11-05 12:09:58.386',
'2015-11-05 12:09:59.384',
'2015-11-05 12:10:00.398',
'2015-11-05 12:10:01.397',
```

Data :
date,value



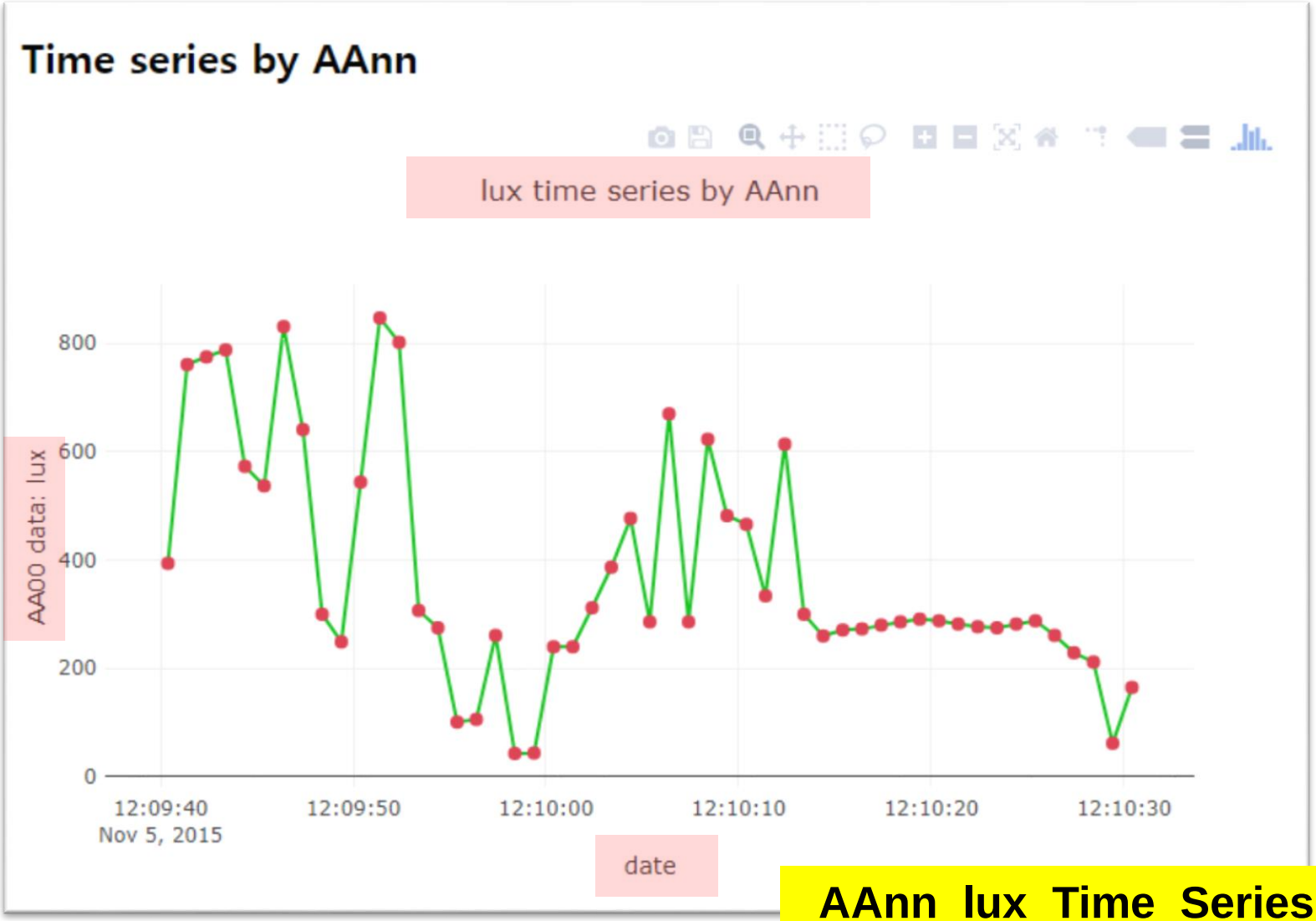
A6.3.10 plotly.js: Time series

[3] Time series : my lux data



A6.3.11 plotly.js: Time series

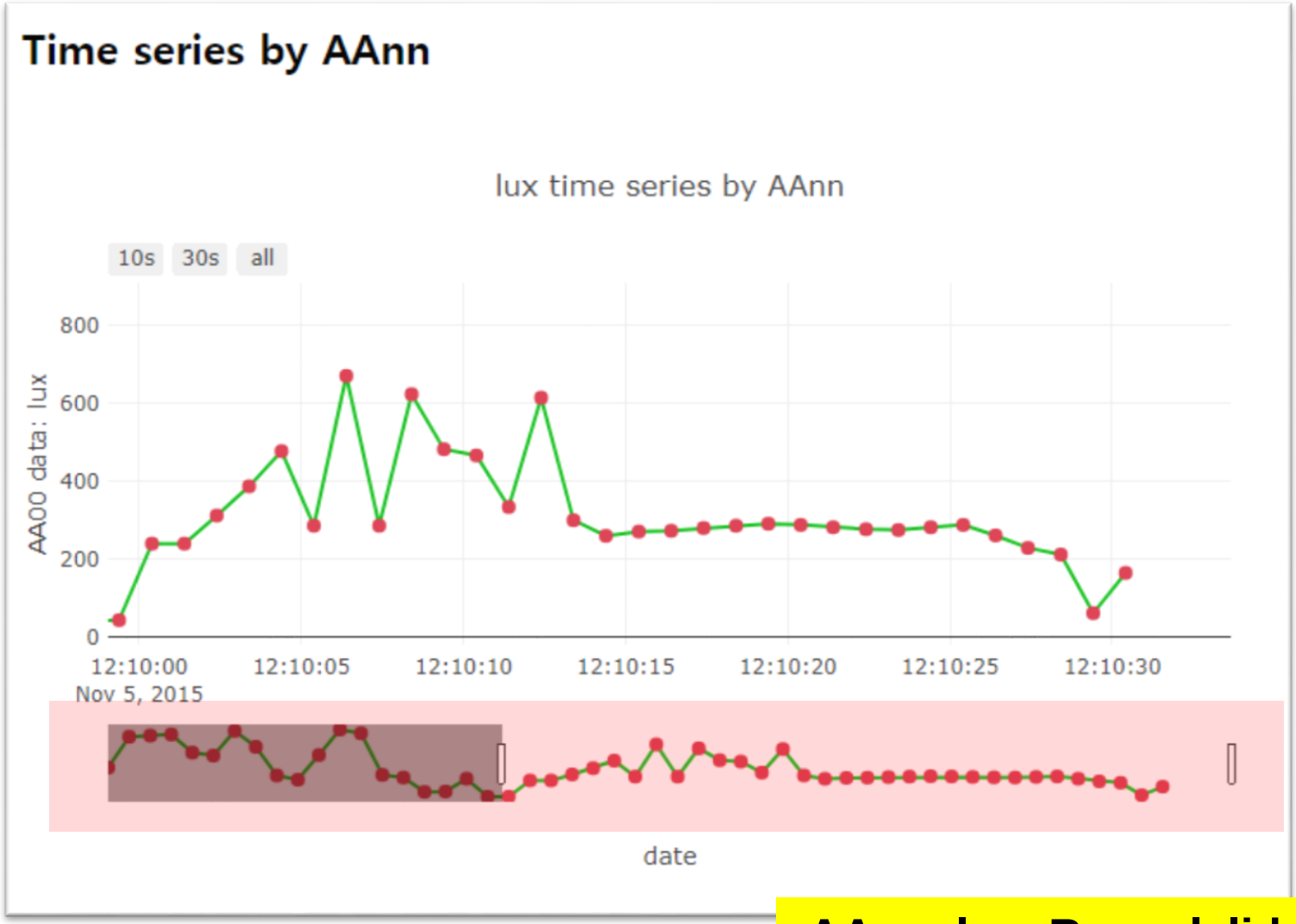
[3] Time series : my lux data – [DIY]



AAnn_lux_Time_Series.png

Project: Time series with Rangeslider

[Project-DIY] AAnn_lux_Rangelslider.html



AAnn_lux_Rangelslider.html

[Practice]

- [wk12]
  - Charts by plotly
  - Complete your plotly chart project
  - Upload file name : AAnn_Rpt09.zip

wk12 : Practice-09 : AAnn_Rpt09.zip

- [Target of this week]
  - Complete your charts
  - Save your outcomes and compress 5 figures & one html.

제출파일명 : **AAnn_Rpt09.zip**

- 압축할 파일들

① AAnn_Chart_Layout.png

② AAnn_Plot_Style.png

③ AAnn_Axis_Title.png

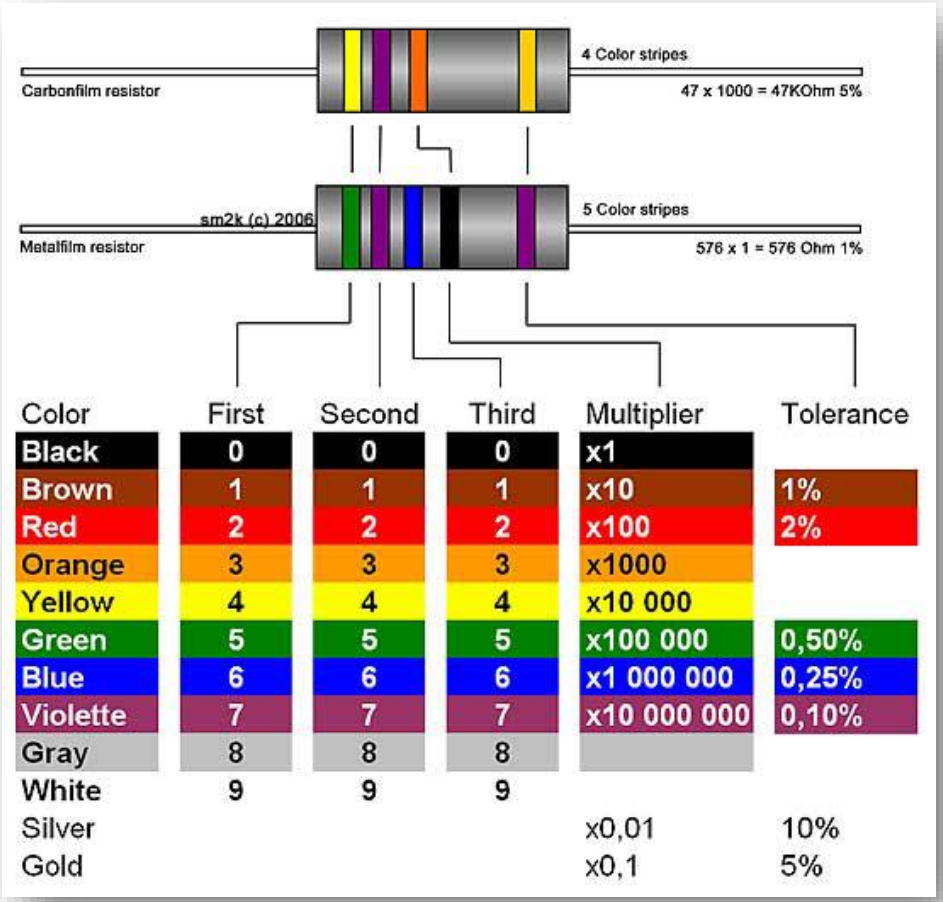
④ AAnn_Line_Dash.png

⑤ AAnn_lux_Time_Series.png

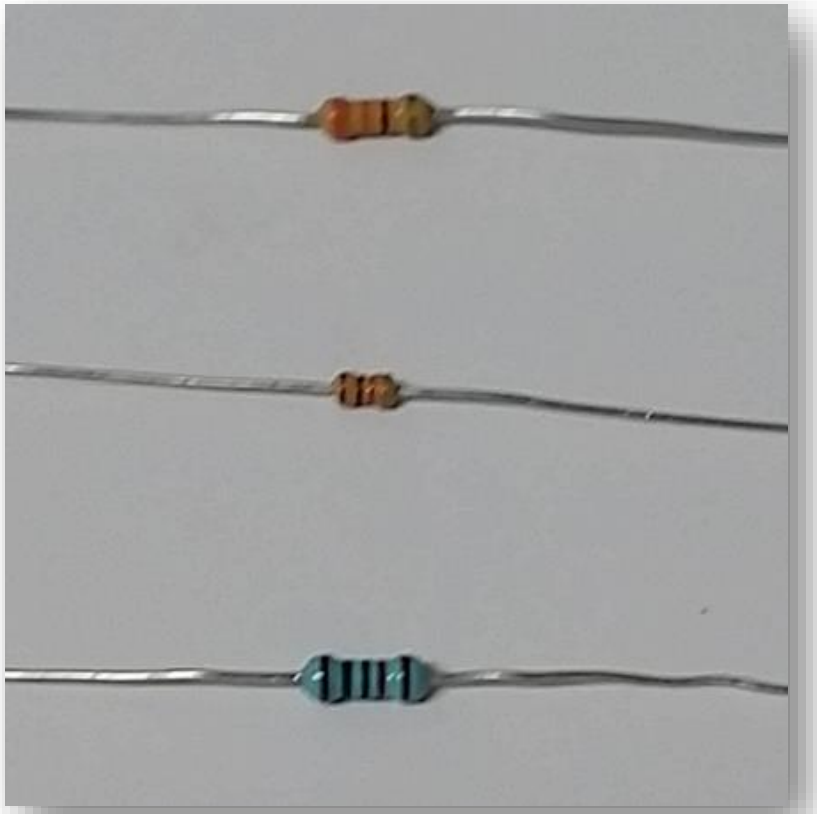
⑥ AAnn_lux_Rangeslider.html

Email : chaos21c@gmail.com

[참고 : 저항 값 읽기]



Color	First	Second	Third	Multiplier	Tolerance
Black	0	0	0	x1	
Brown	1	1	1	x10	1%
Red	2	2	2	x100	2%
Orange	3	3	3	x1000	
Yellow	4	4	4	x10 000	
Green	5	5	5	x100 000	0,50%
Blue	6	6	6	x1 000 000	0,25%
Violette	7	7	7	x10 000 000	0,10%
Gray	8	8	8		
White	9	9	9		
Silver				x0,01	10%
Gold				x0,1	5%



Lecture materials

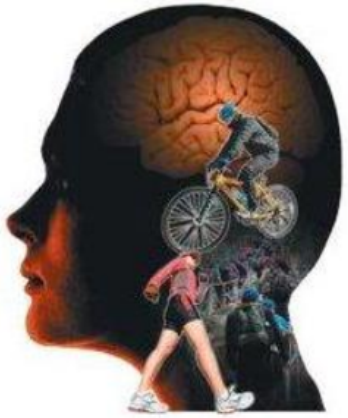
References & good sites

- ✓ http://www.nodejs.org/ko Node.js
- ✓ http://www.arduino.cc Arduino Homepage
- ✓ http://www.w3schools.com By w3schools
- ✓ http://www.github.com GitHub
- ✓ http://www.google.com Googling

Github.com/Redwoods

GitHub, Inc. [US] | https://github.com Redwoods (Redwoods Yi) · GitHub

Features Business Explore Marketplace Pricing Search GitHub Sign in or Sign up



Redwoods Yi

Redwoods

Block or report user

GimHae, Republic of Korea

Overview Repositories 5 Stars 2 Followers 0 Following 0

Pinned repositories

dht22-iot-project

Iot project to monitor data streaming from DHT22 wired at Arduino.

HTML

Lec

All lectures by Redwoods in Inje University

arduino-nodejs-plotly-streaming

This repo introduces a simple and efficient way to plot the streaming data from Arduino with Easy Pulse ppg sensor or DHT11 sensor.

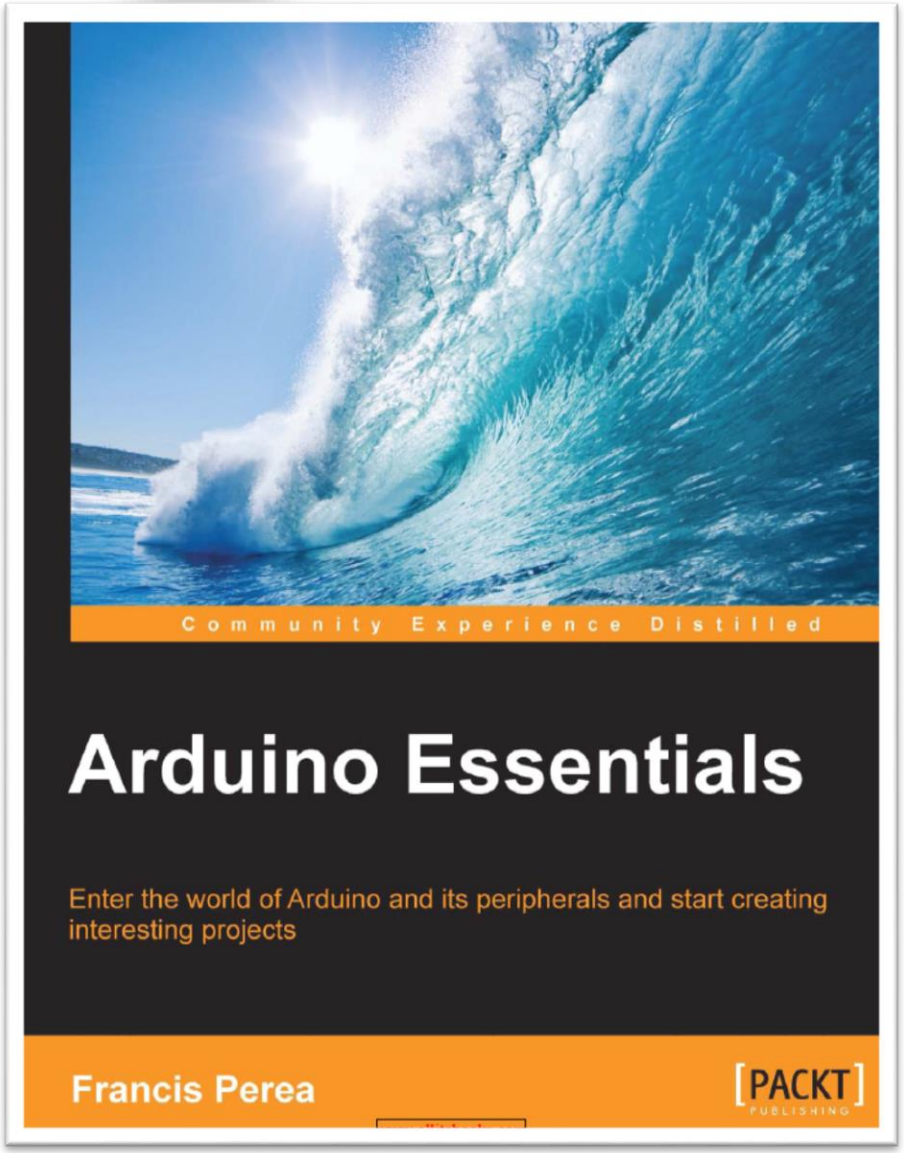
HTML

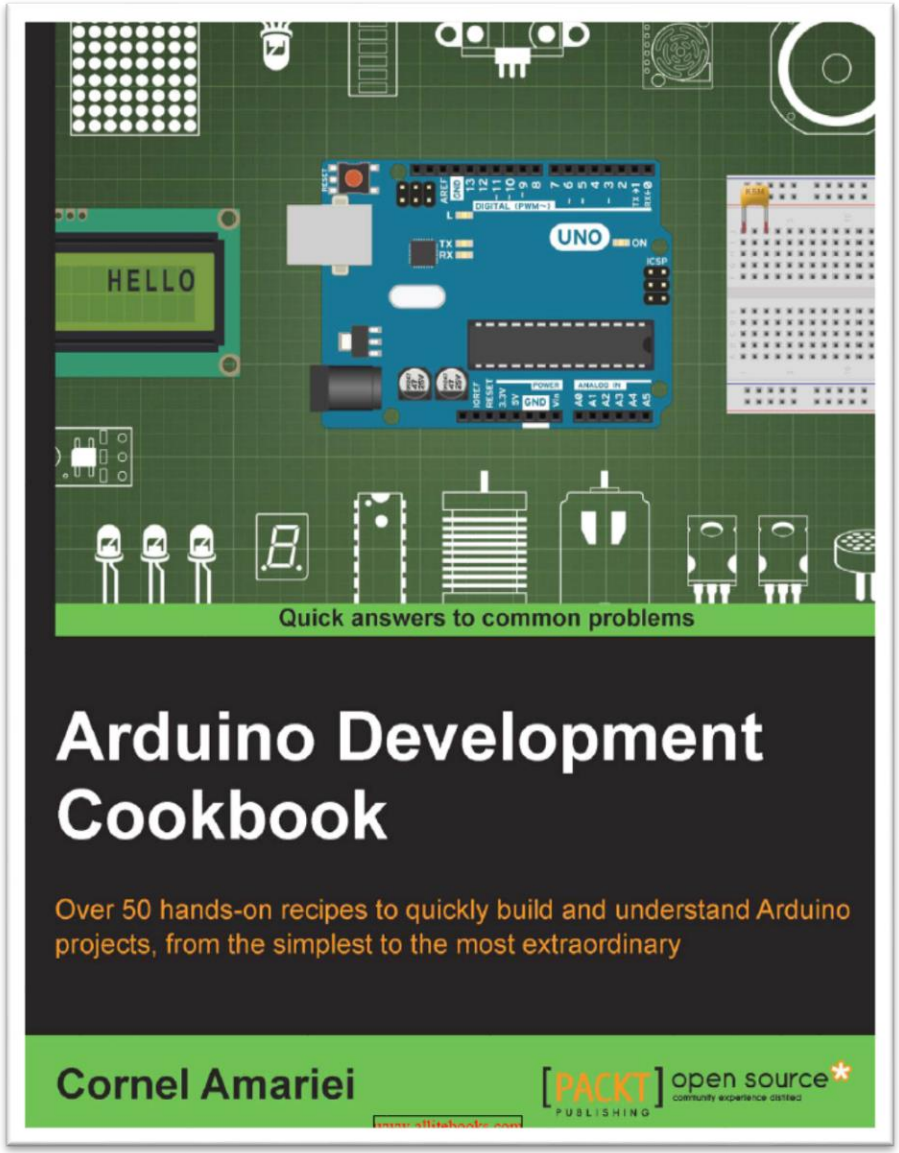
hw-coding

Resource for lecture of Hardware Programming (2017, Inje university)

Arduino

References


Community Experience Distilled
Arduino Essentials
Enter the world of Arduino and its peripherals and start creating interesting projects
Francis Perea
[PACKT]
PUBLISHING


HELLO
UNO
Quick answers to common problems
Arduino Development Cookbook
Over 50 hands-on recipes to quickly build and understand Arduino projects, from the simplest to the most extraordinary
Cornel Amariei
[PACKT] open source
PUBLISHING