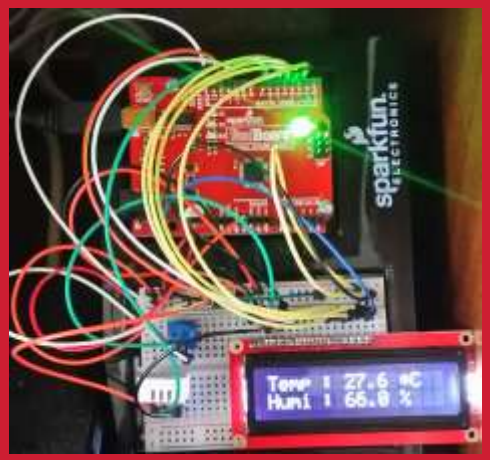


Arduino-IOT

[wk08]

Data Visualization I - plotly.js charts

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

Drone-IoT-Comsi, INJE University

2nd semester, 2023

Email : chaos21c@gmail.com

My ID

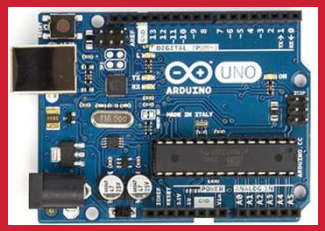
ID를 확인하고 github에 repo 만들기

ID	성명
AA01	강동하
AA02	고서진
AA03	김민재
AA04	김예원
AA05	김주호
AA06	김창욱
AA07	김현서
AA08	박종혁
AA09	서명진
AA10	유동기
AA11	
AA12	이근보
AA13	정호기

위의 id를 이용해서 github에 repo를 만드시오.

Option: 아두이노응용 실습 과제 – AAnn

Public, README.md check

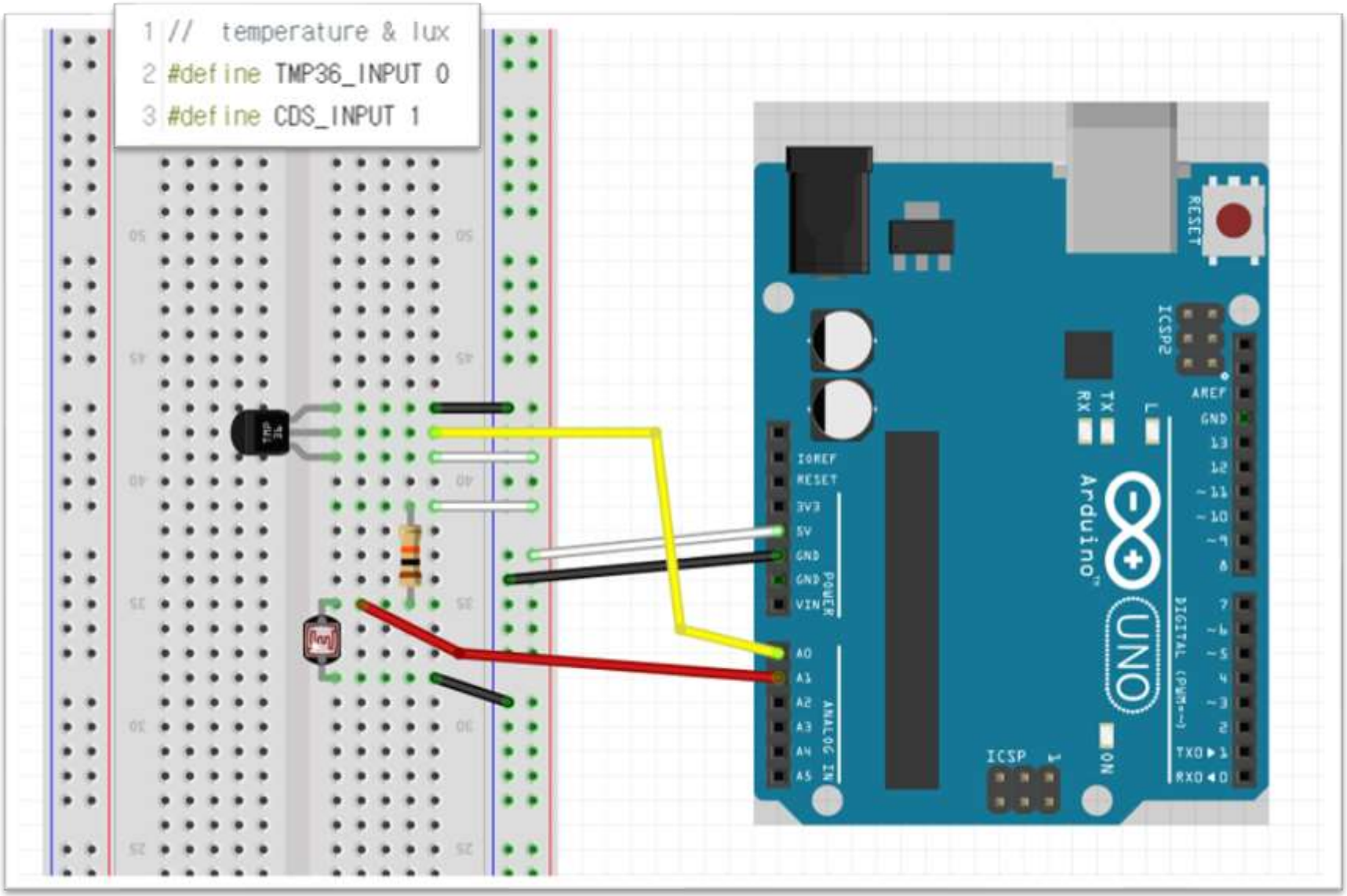


[Mid-Exam]

[wk07]

- cds + tmp36 회로 구성 → 조도,온도
- 습도(humi)를 40~90 범위로 simulation (cds_tmp36_humi.ino)
- 조도,습도,온도 순서로 3개의 신호 처리 (cds_tmp36_humi.js)
- 조도,습도,온도 웹 모니터링 (client_cds_tmp36_humi.html)

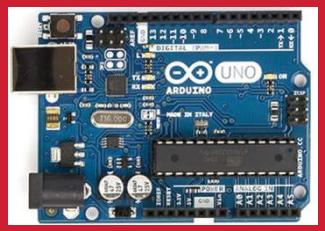
A4.3.1 TMP36 + CdS : circuit



A4.5.6 multi-signals + Node project : WEB

[Web monitoring] client_cds_tmp36_humi.html

Save as
AAnn_cds_tmp36_humi_WEB.png



[Practice]

- [wk07]
  - Arduino sensors + Node.js
  - Complete your project
  - Upload folder: aann-rpt07
  - Use repo “aann” in github

wk07 : Mid-Exam : aann-rpt07

[Target of this week]

- Complete your works & update your repo.
- Save your outcomes and upload outputs in github repo.

제출폴더명 : **aann-rpt07**

- 압축할 파일들

① cds_tmp36_humi.ino

② cds_tmp36_humi.js

③ cds_tmp36_humi_WEB.png

④ client_cds_tmp36_humi.html

⑤ NO node_modules folder

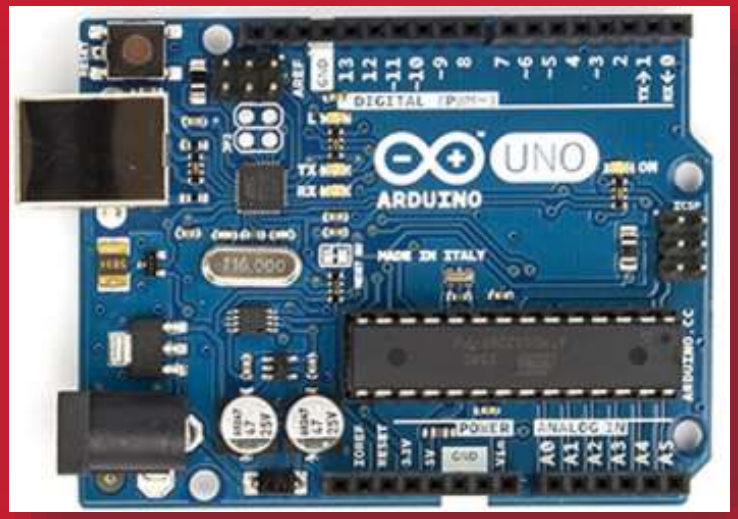
Purpose of AA

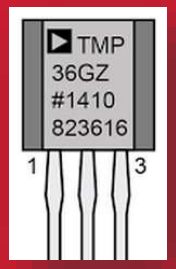
주요 수업 목표는 다음과 같다.

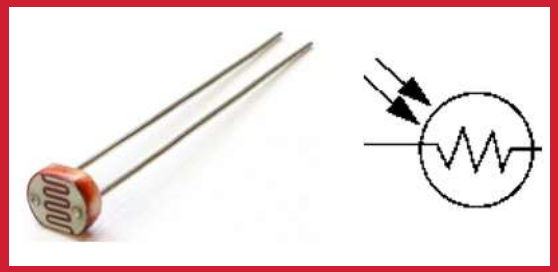
1. Node.js를 이용한 아두이노 센서 신호 처리
2. Plotly.js를 이용한 아두이노 센서 신호 시각화
3. MongoDB에 아두이노 센서 데이터 저장 및 처리

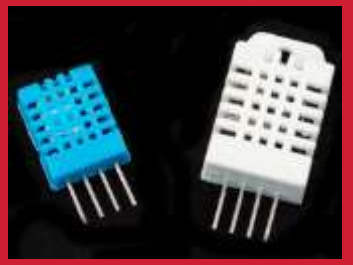
4. 저장된 IoT 데이터의 마이닝 (파이썬 코딩)

Arduino Sensors + Node.js

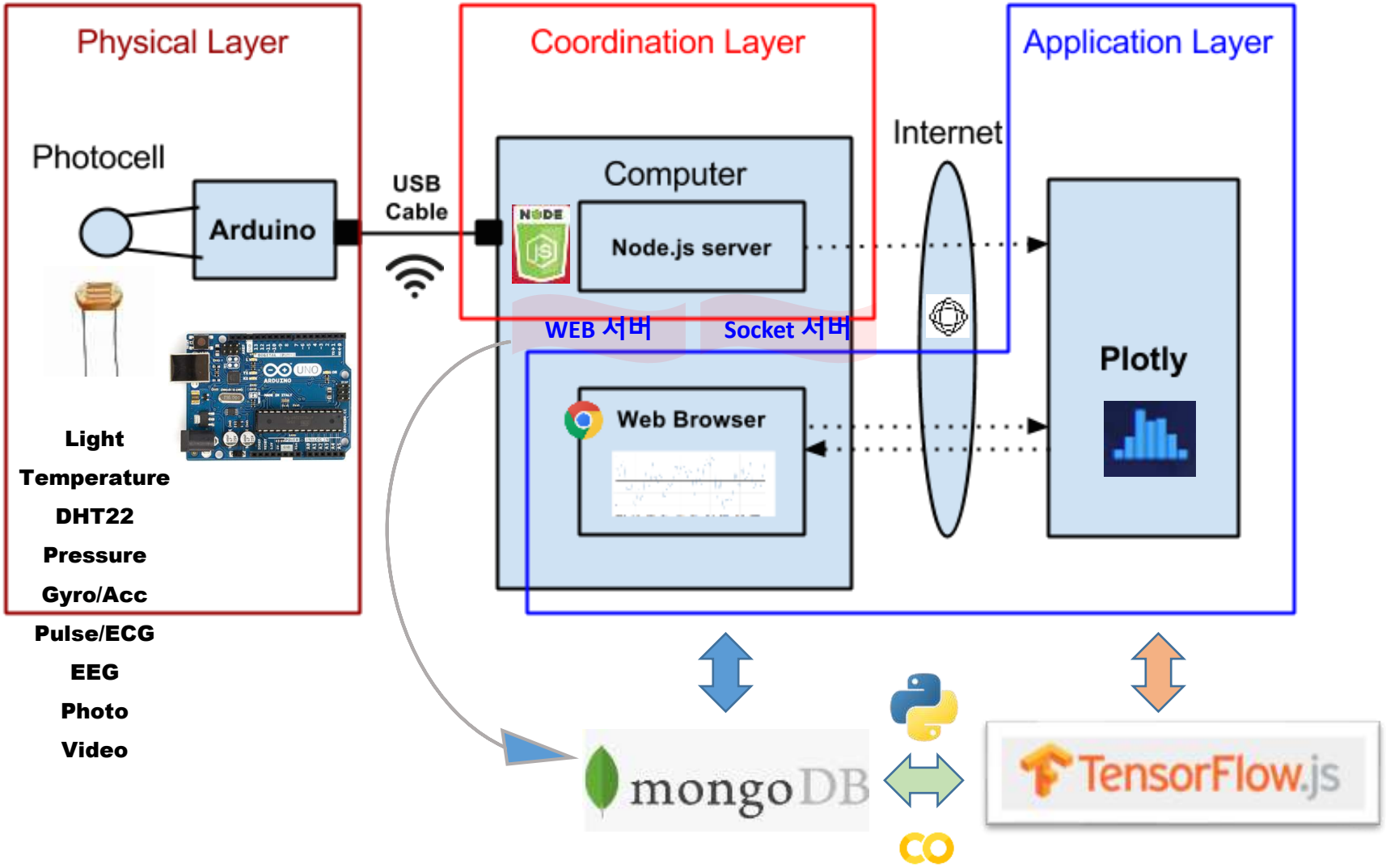




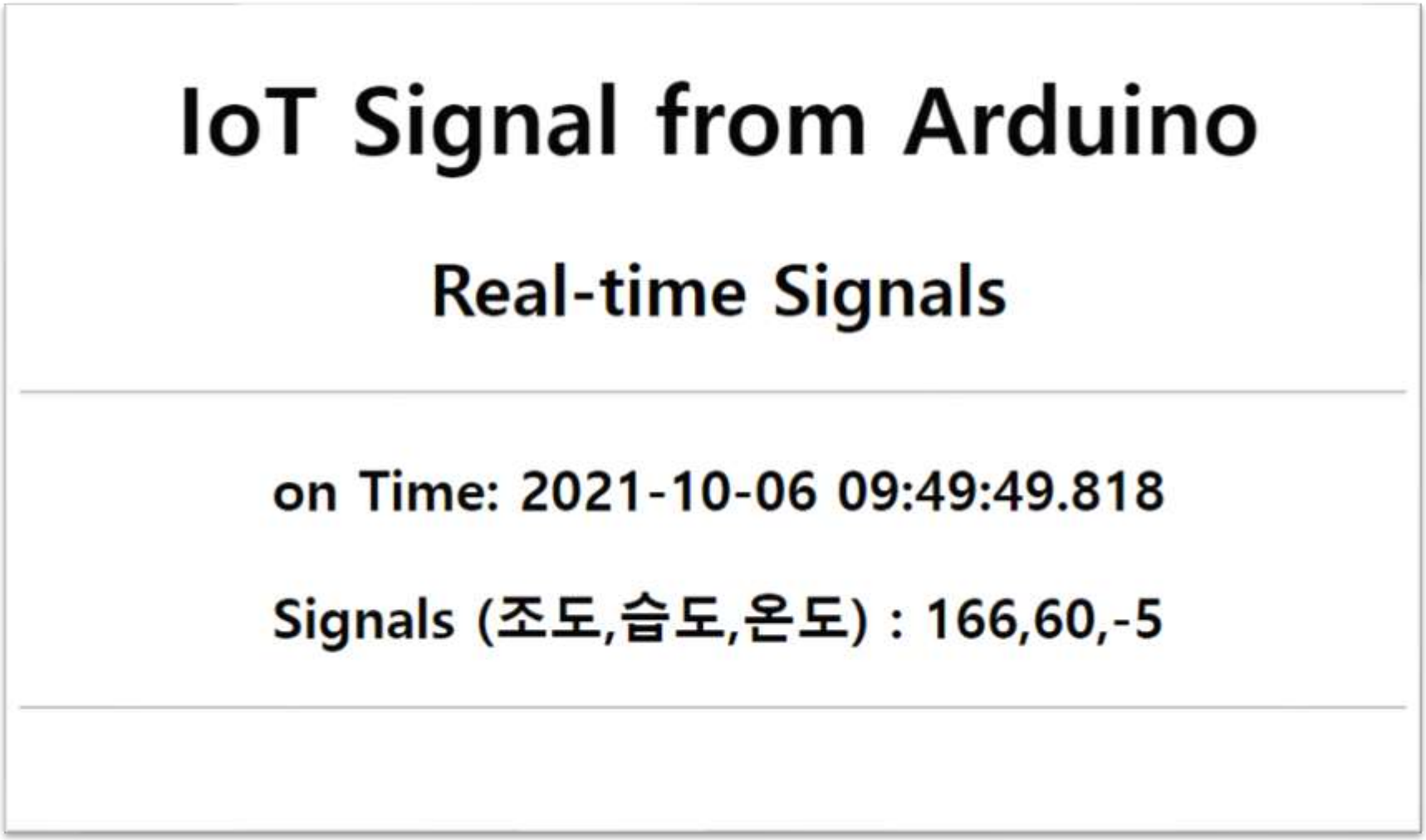




Layout [H S C]


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

on WEB monitoring Arduino data

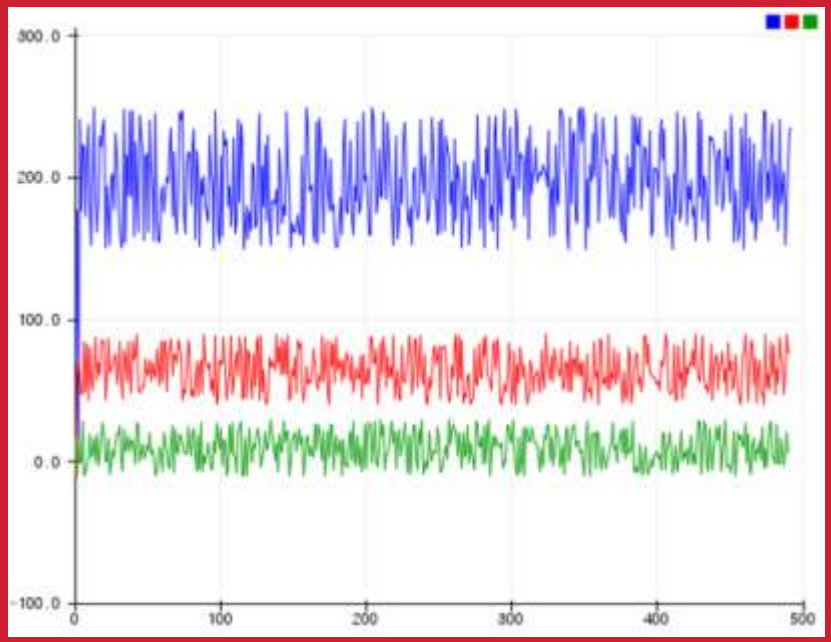


http://chaos.inje.ac.kr:3030/iot_multi.html

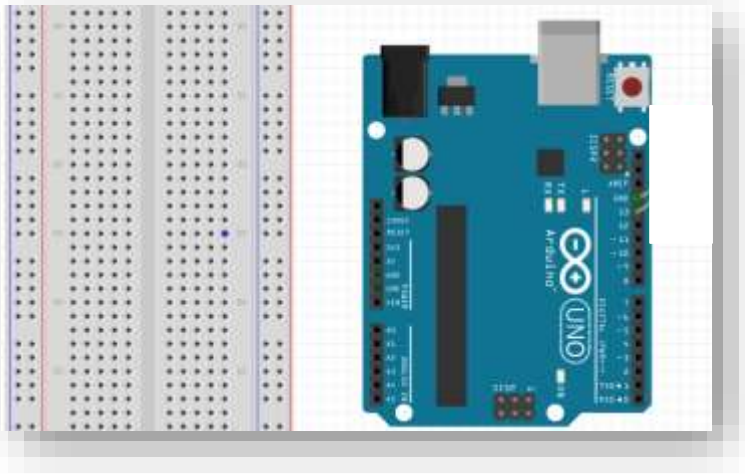
[DIY] Multi-signals

다중신호 시뮬레이션

+ node.js



DIY - 스케치



아두이노에서 **LED**와 저항을 모두 제거하고 **USB**만 컴퓨터와 연결한다.

전자 소자 연결 없이 마구잡이 수 생성 함수를 이용해서 조도, 습도, 온도에 해당되는 **3**개의 신호를 만든다.

온도는 값의 범위를 **–10 ~ 30,** 습도는 **40 ~ 90,** 그리고 조도는 **150 ~ 250** 으로 가상적 으로 설정한다.

직렬통신 모니터링을 이용해서 세 개의 신호의 변화를 모니터링 하는 코드를 만들어 결과를 확인한다.

▶ 스케치 구성

1. 3 개의 신호를 담을 변수를 초기화한다.
2. setup()에서 직렬 통신 속도를 9600 bps 로 설정하고 컴퓨터와 연결한다.
3. loop()에서 마구잡이 수를 세 개 발생시켜서 직렬 통신으로 3 개의 pwm 값을 각각 컴퓨터로 전송한다.

DIY - code

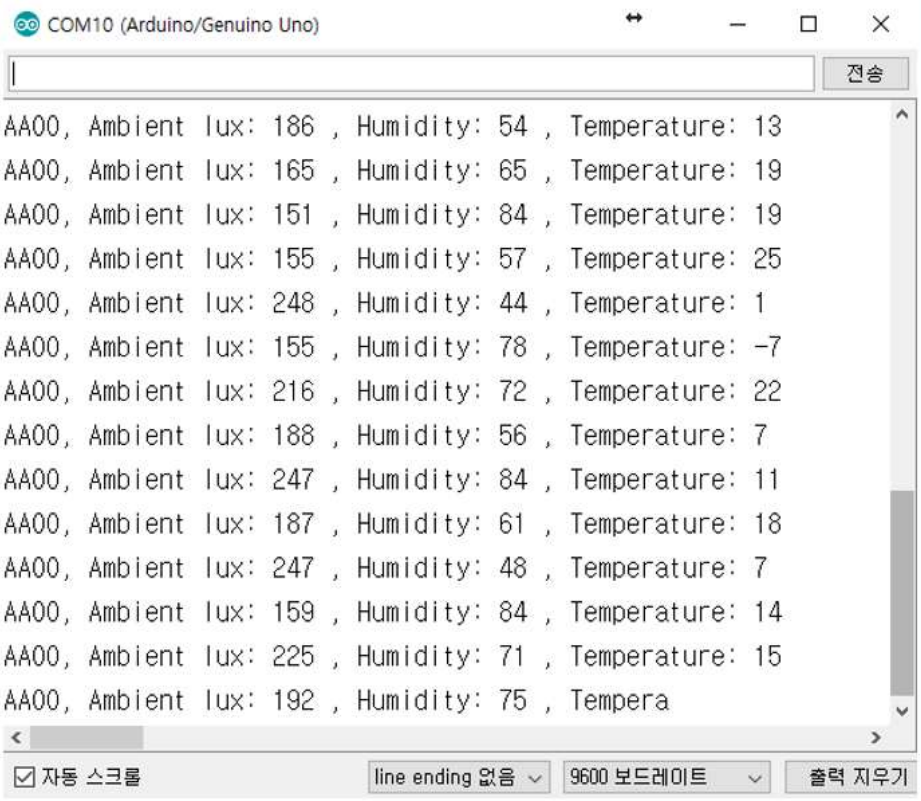
sketch05_multi_signals

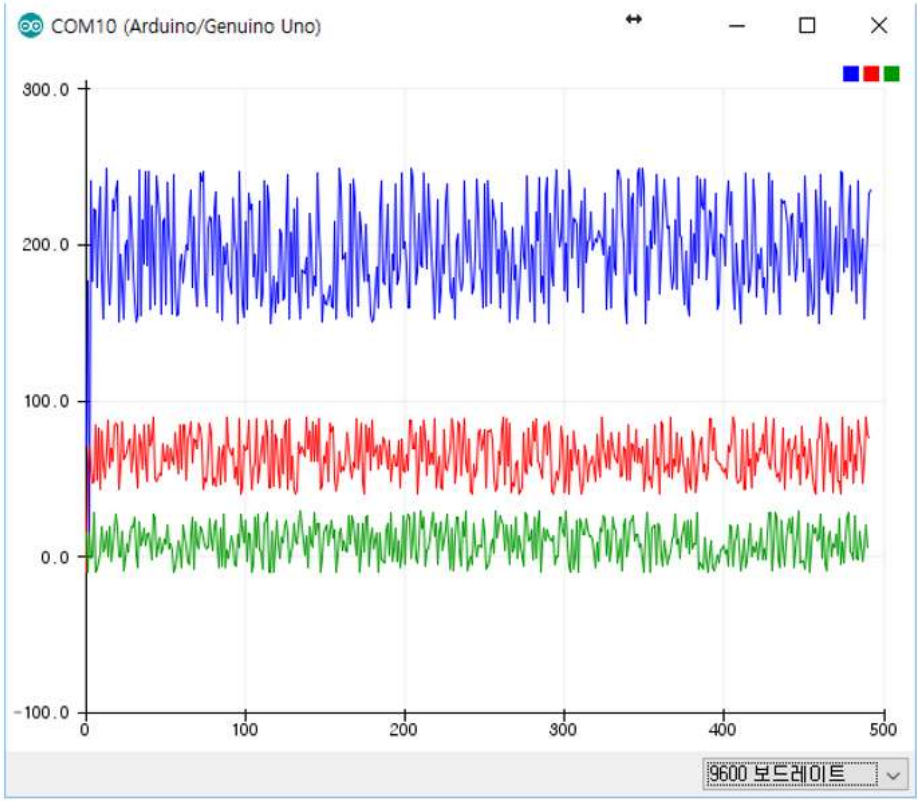
```
/*
  Multi Signals
  Simulation of multiple random signals
*/
// signals
int humi=0;
int temp=0;
int lux=0;

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // Multi signals
  humi = random(40,90);
  temp = random(-10, 30);
  lux = random(150,250);
  Serial.print("AA00, Ambient lux: ");
  Serial.print(lux);
  Serial.print(" , Humidity: ");
  Serial.print(humi);
  Serial.print(" , Temperature: ");
  Serial.println(temp);
  delay(500);          // delay in between reads for stability
}
```

DIY - result

DIY 결과

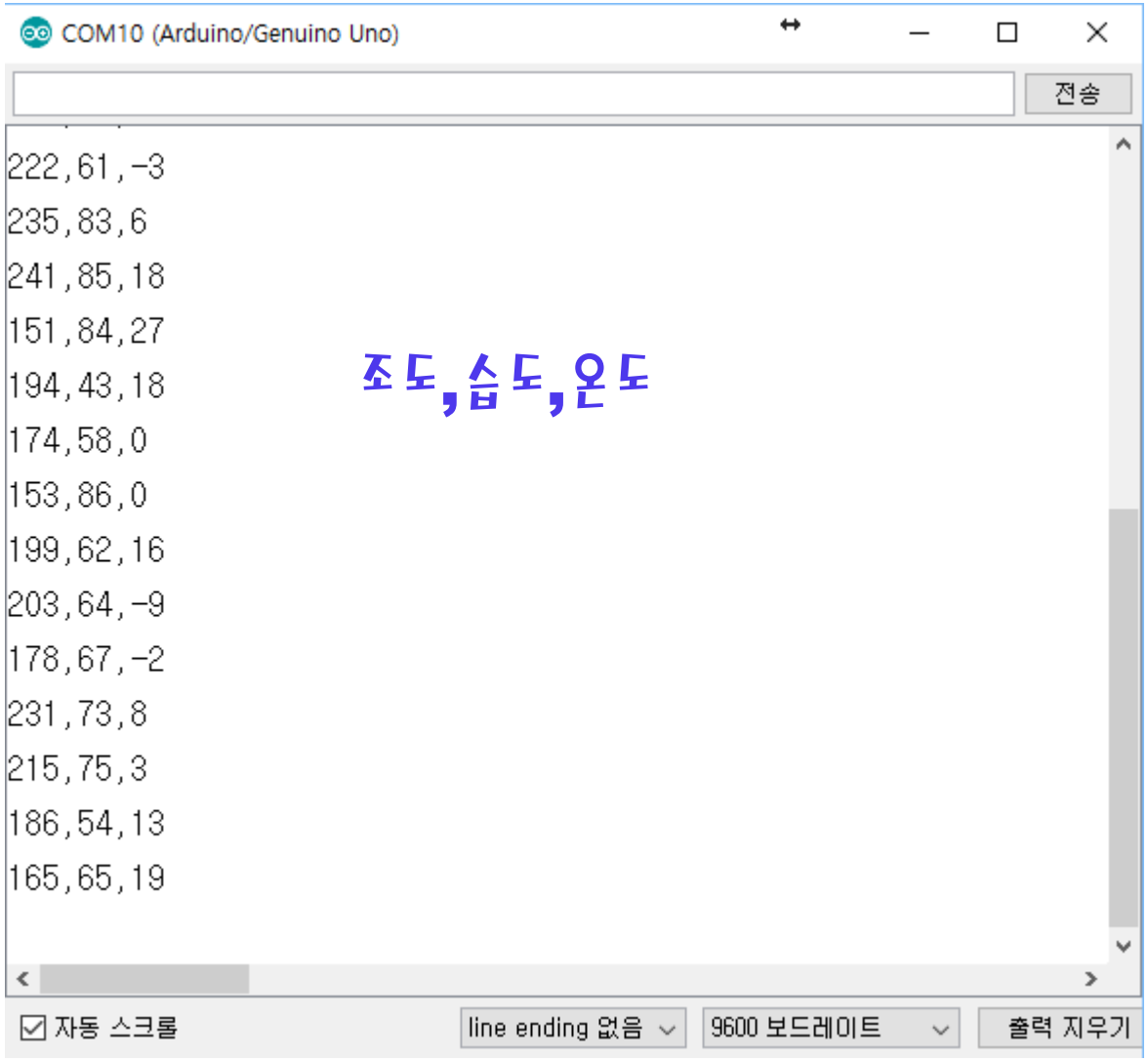
가상적인 세 개의 센서신호 시뮬레이션:조도(위), 습도(중간), 온도(아래).





DIY – New result 1

DIY 결과 [1] : 가상적인 세 개의 센서신호 시뮬레이션 → 조도,습도,온도



DIY – New result 2-1

DIY 결과 [2] : 가상적인 세 개의 센서신호 시뮬레이션 → 조도,습도,온도를 Node.js로 처리

[1 단계] Node cmd

1. Make multi_signals node project

- md multi_signals in iot folder
- cd multi_signals

2. Go to multi_signals subfolder

- npm init

  name : multi_signals

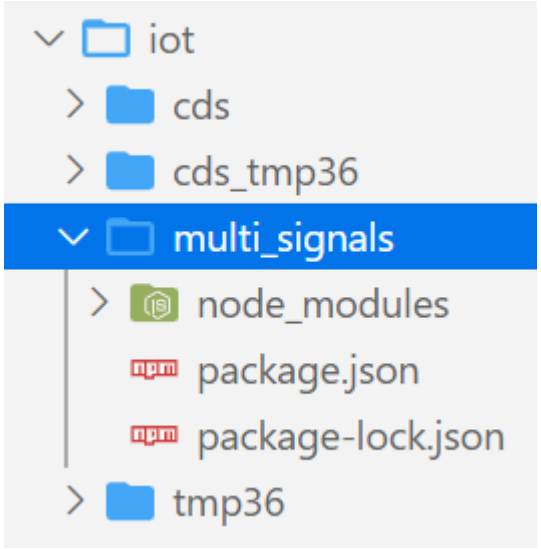
  description : multi-signals-node project

  entry point : aann_multi_signals.js

  author : aann

3. Install node modules

- npm install –save serialport@9.2.4
- npm install –save socket.io@2.4.1



DIY – New result 2-2

DIY 결과 [2] : 가상적인 세 개의 센서신호 시뮬레이션 → 조도,습도,온도를 Node.js로 처리

Recycling code:
Save cds_tmp36_node.js as
aann_multi_signals.js in multi_signals subfolder
Update code

```
var dStr ='';
var readData='';
var temp='';
var humi='';
var lux='';
var mdata=[];
var firstcommaidx = 0;
var secondcommaidx= 0;
```

DIY – New result 2-3

DIY 결과 [2] : 가상적인 세 개의 센서신호 시뮬레이션 → 조도,습도,온도를 Node.js로 처리

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

아두이노가 직렬통신으로 전송하는 2 개의 comma (,)로 구분된

조도,습도,온도 데이터 메시지를 **parsing**하여 **mdata** 배열에 담는 코드를 완성하시오.

substring() 함수에서 firstcommaidx, secondcommaidx를 잘 이용하시오.

```
    console.log("AA00," + mdata);
    io.sockets.emit("message", mdata); // send data to all clients
  } else {
    console.log(readData);
  }
});
```

DIY – New result 2-3

DIY 결과 [2] : 가상적인 세 개의 센서신호 시뮬레이션 → 조도,습도,온도를 Node.js로 처리

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

    dStr = getDateString();
    mdata[0] = dStr; //date
    mdata[1] = lux; //data
    mdata[2] = humi;
    mdata[3] = temp;
    console.log("AA00," + mdata.toString());
    io.sockets.emit("message", mdata); // send data to all clients
  } else {
    console.log(readData);
  }
});
```

DIY – New result 2-4 : js functions

DIY 결과 [2] : 가상적인 세 개의 센서신호 시뮬레이션 → 조도,습도,온도를 Node.js로 처리

Hint:

javascript function : indexOf()

https://www.w3schools.com/jsref/jsref_indexof.asp

Syntax

```
string.indexOf(searchvalue, start)
```

Parameter Values

Parameter	Description
searchvalue	Required. The string to search for
start	Optional. Default 0. At which position to start the search

javascript function : substring()

```
string.substring(start, end)
```

Parameter Values

Parameter	Description
start	Required. The position where to start the extraction. First character is at index 0
end	Optional. The position (up to, but not including) where to end the extraction. If omitted, it extracts the rest of the string

DIY – New result 2-5

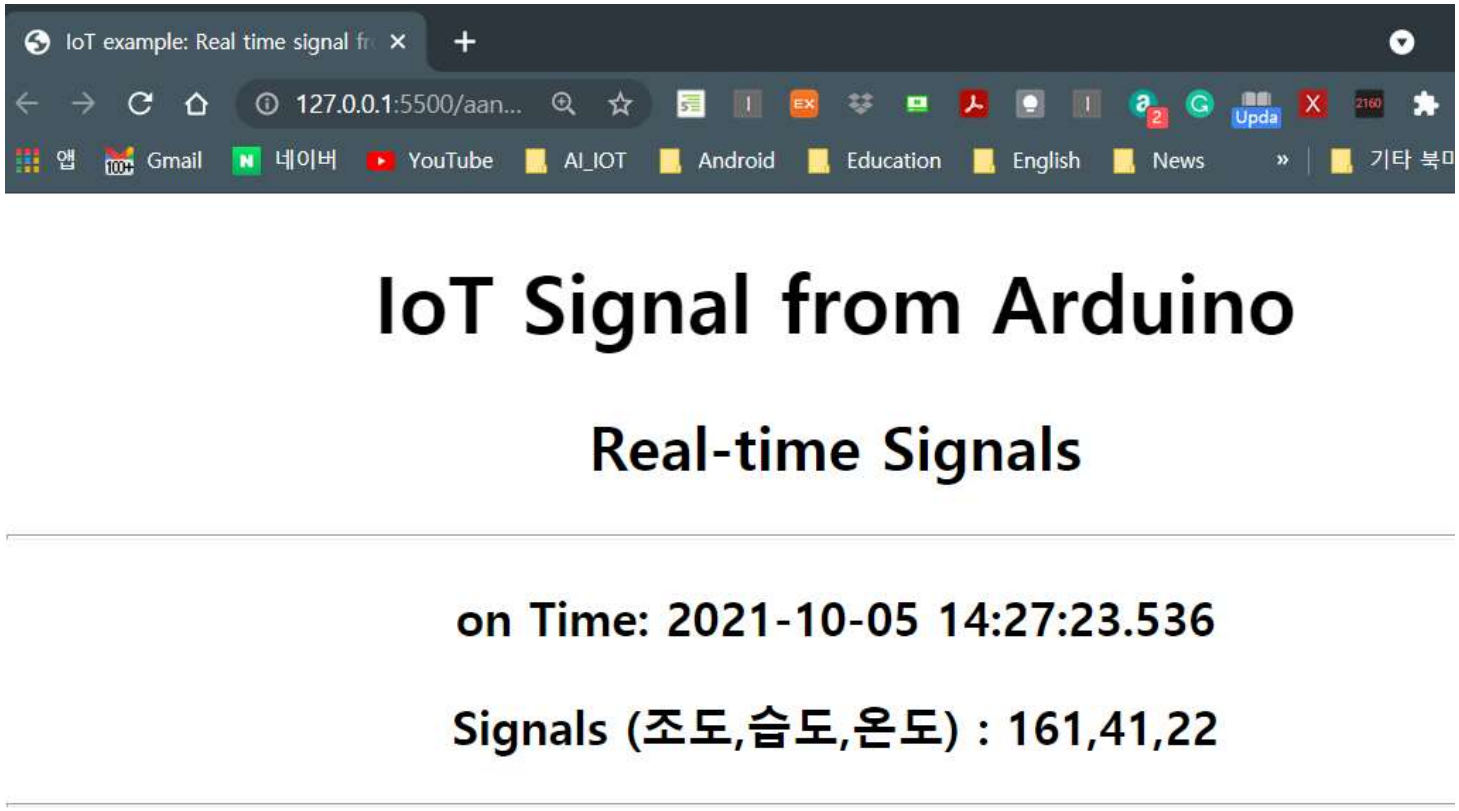
DIY 결과 [2] : 가상적인 세 개의 센서신호 시뮬레이션 → 조도,습도,온도를 Node.js로 처리

```
D:\aann\aann-rpt06\iot\multi_signals>node aann_multi_signals
serial port open
AA00,2021-10-05 14:21:10.805,223,47,-1
AA00,2021-10-05 14:21:11.804,222,48,0
AA00,2021-10-05 14:21:12.808,173,84,28
AA00,2021-10-05 14:21:13.811,215,49,-10
AA00,2021-10-05 14:21:14.811,237,82,-8
AA00,2021-10-05 14:21:15.815,179,43,-3
AA00,2021-10-05 14:21:16.814,153,80,2
AA00,2021-10-05 14:21:17.818,207,59,19
AA00,2021-10-05 14:21:18.817,249,50,3
AA00,2021-10-05 14:21:19.821,185,68,6
AA00,2021-10-05 14:21:20.820,162,87,16
```

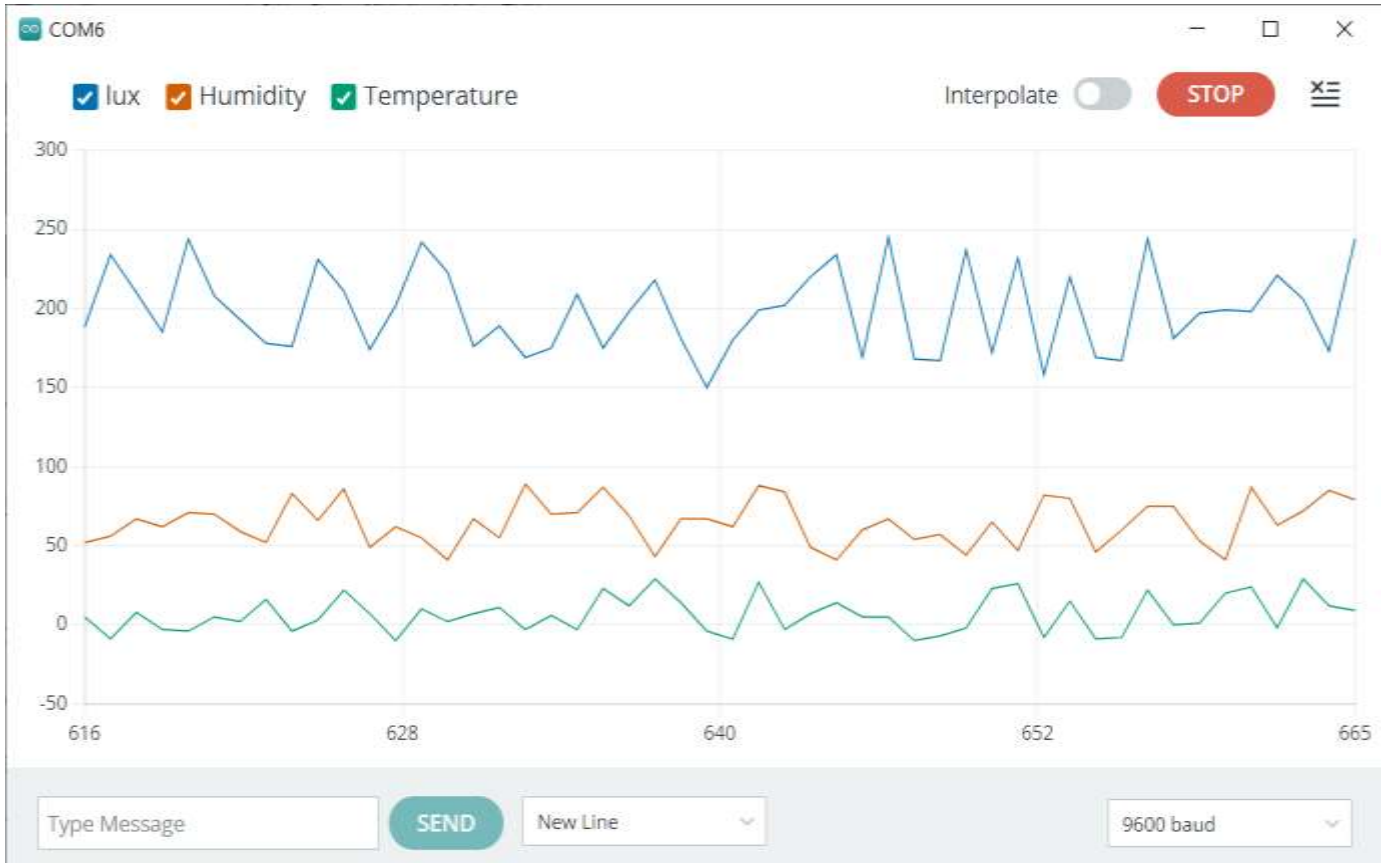
ID,시간,조도,습도,온도

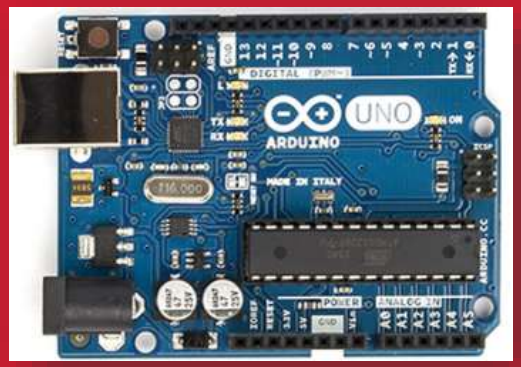
A4.5.6 multi-signals + Node project : WEB

[Web monitoring] client_multi_signals.html

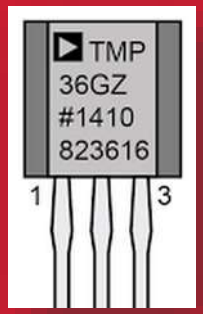


[Web monitoring] client_multi_signals_streaming.html



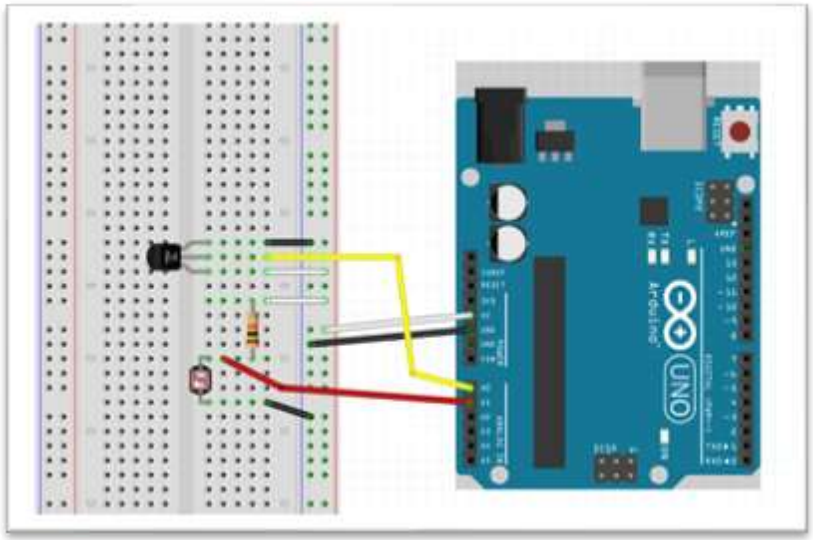


Data visualization using plot.ly



Network socket emitting data

tmp36 + CdS circuit



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

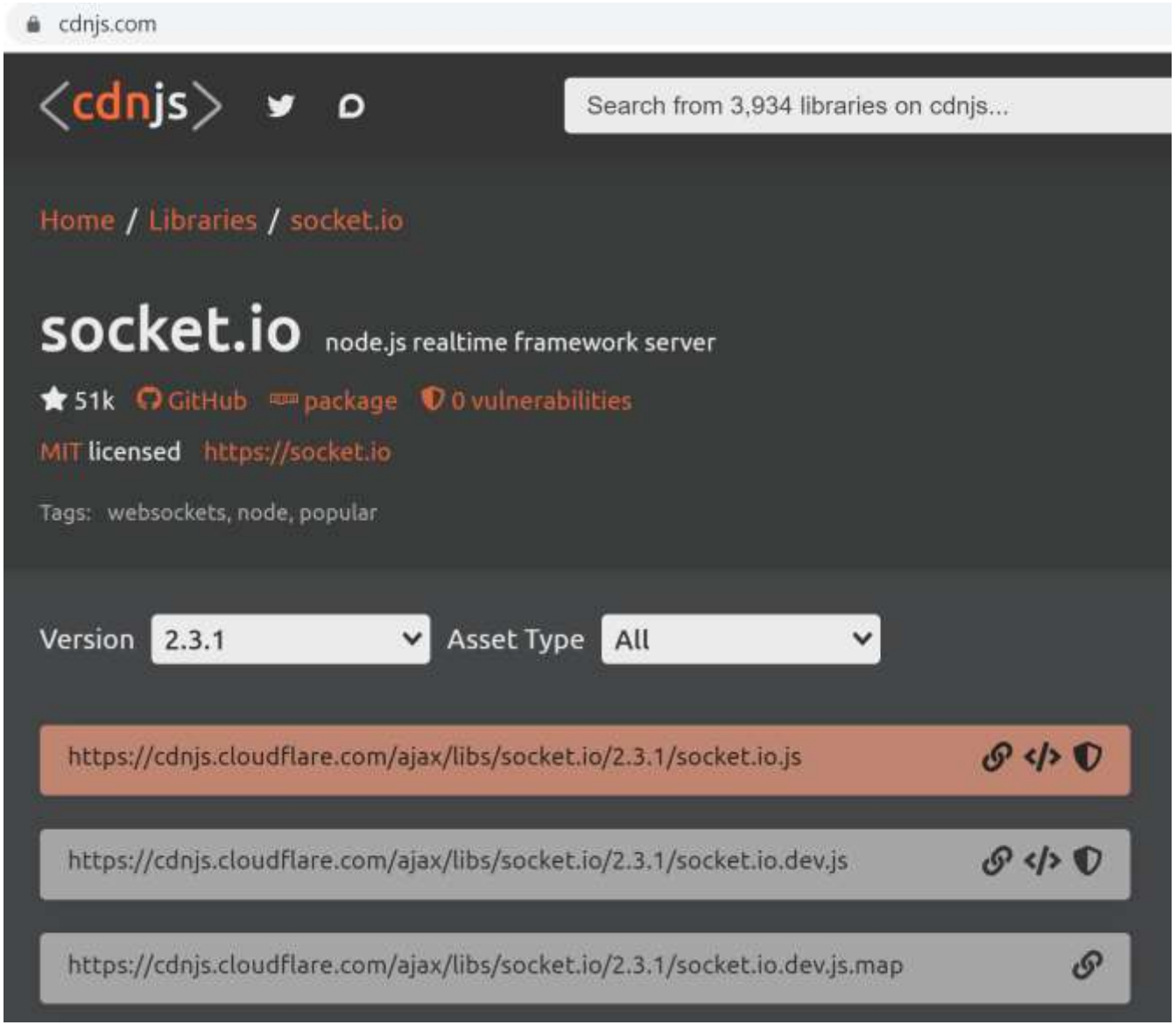
```
var readData = "";
var temp = "";
var lux = "";
var mdata = [];
var firstcommaidx = 0;

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

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

시간,온도,조도

Arduino data on network socket

Google search

socket.io.js cdn



Arduino data on network socket

client_signal_start.html

```
1  <!DOCTYPE html>
2  <head>
3    <meta charset="utf-8">
4    <title>IoT example: Real time signal from Arduino</title>
5
6    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.3.1/socket.io.js"></script>
7    <!-- <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/socket.io.js"></scr
8    <style>body{padding:0;margin:30;background:#fff}</style>
9  </head>
10
11 <body>  <!-- style="width:100%;height:100%"> -->
12
13 <h1 align="center"> IoT Signal from Arduino </h1>
14
15 <h2 align="center"> Real-time Signals </h2>
16
17 <hr>
18
19 <h3 align="center"> on Time: <span id="time"> </span> </h3>
20
21 <h3 align="center"> Signal (temp, lux) : <span id="data"> </span> </h3>
22
```

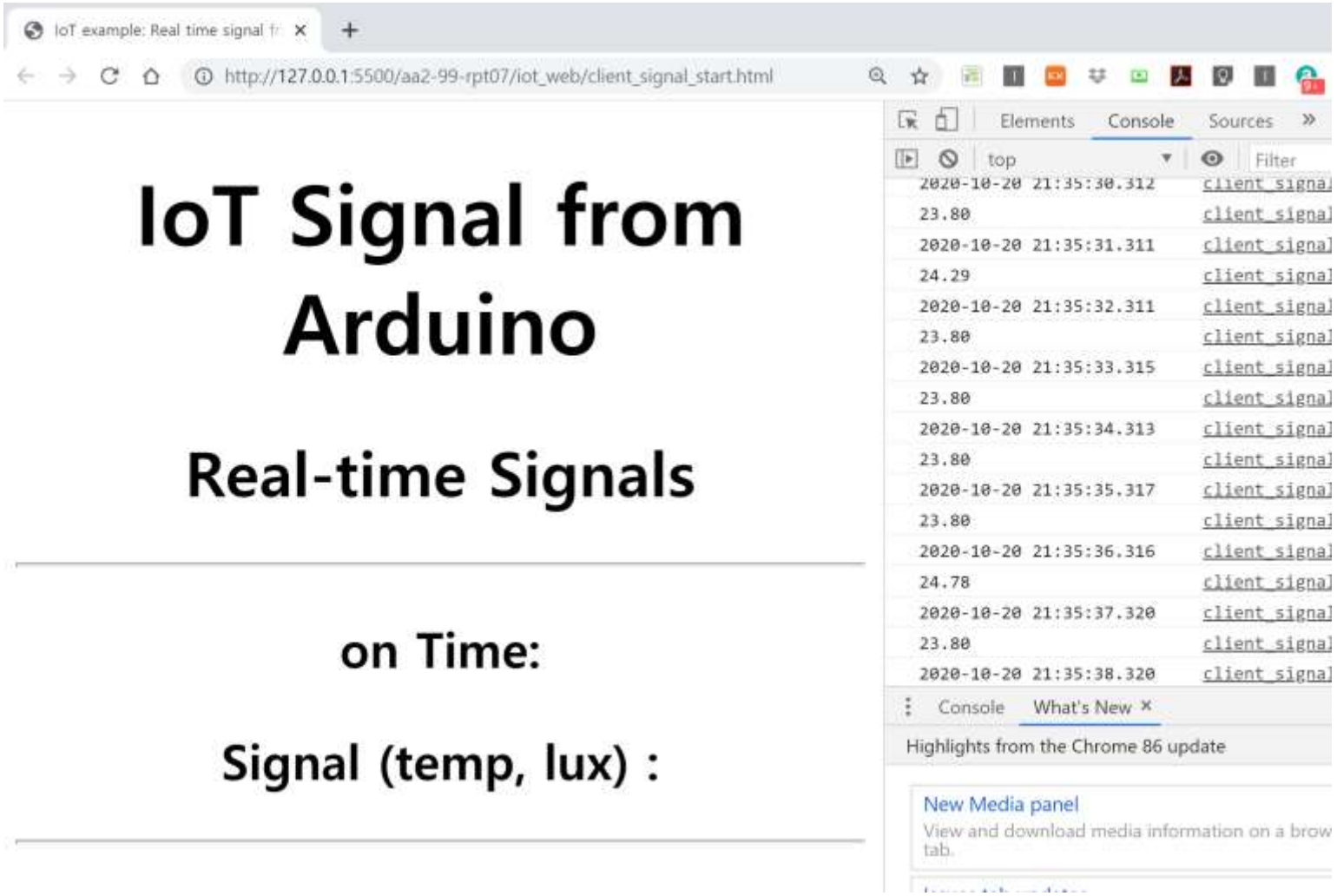
Google search : socket.io.js cdn

Arduino data on network socket

client_signal_start.html

```
25  <script>
26  /* JAVASCRIPT CODE GOES HERE */
27    var ctime = document.getElementById('time');
28    var cdata = document.getElementById('data');
29
30    // Get message from network socket
31    var socket = io.connect('http://localhost:3000'); // port = 3000
32    socket.on('connect', function () {
33        socket.on('message', function (msg) {
34            // message on Socket (TCP server)
35            // Check the console by F12 in Chrome browser
36            console.log(msg[0]); // mdata[0] => dStr
37            console.log(msg[1]); // mdata[1] => msg[1]
38            console.log(msg[2]);
39            //console.log(msg[3]);
40
41            // Display signal on web browser
42            ctime.innerHTML = msg[0];
43            cdata.innerHTML = msg[1] + "," + msg[2]; // + "," + msg[3];
44        });
45    });
46
47  </script>
```

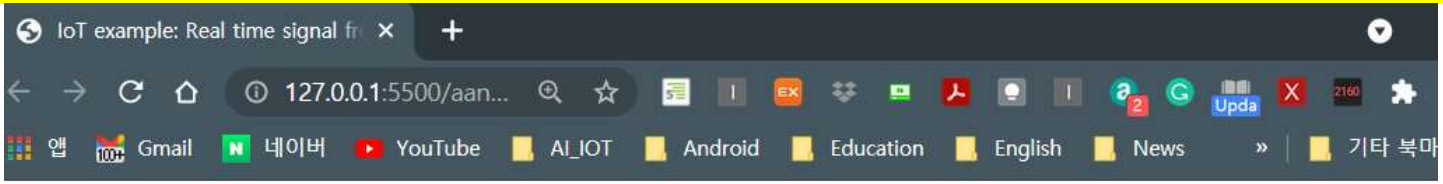
Arduino data on network socket



Real-time console showing signals from Arduino in Chrome browser

Arduino data on network socket

[Web monitoring] client_signal_cds_tmp36.html



IoT Signal from Arduino

Real-time Signals

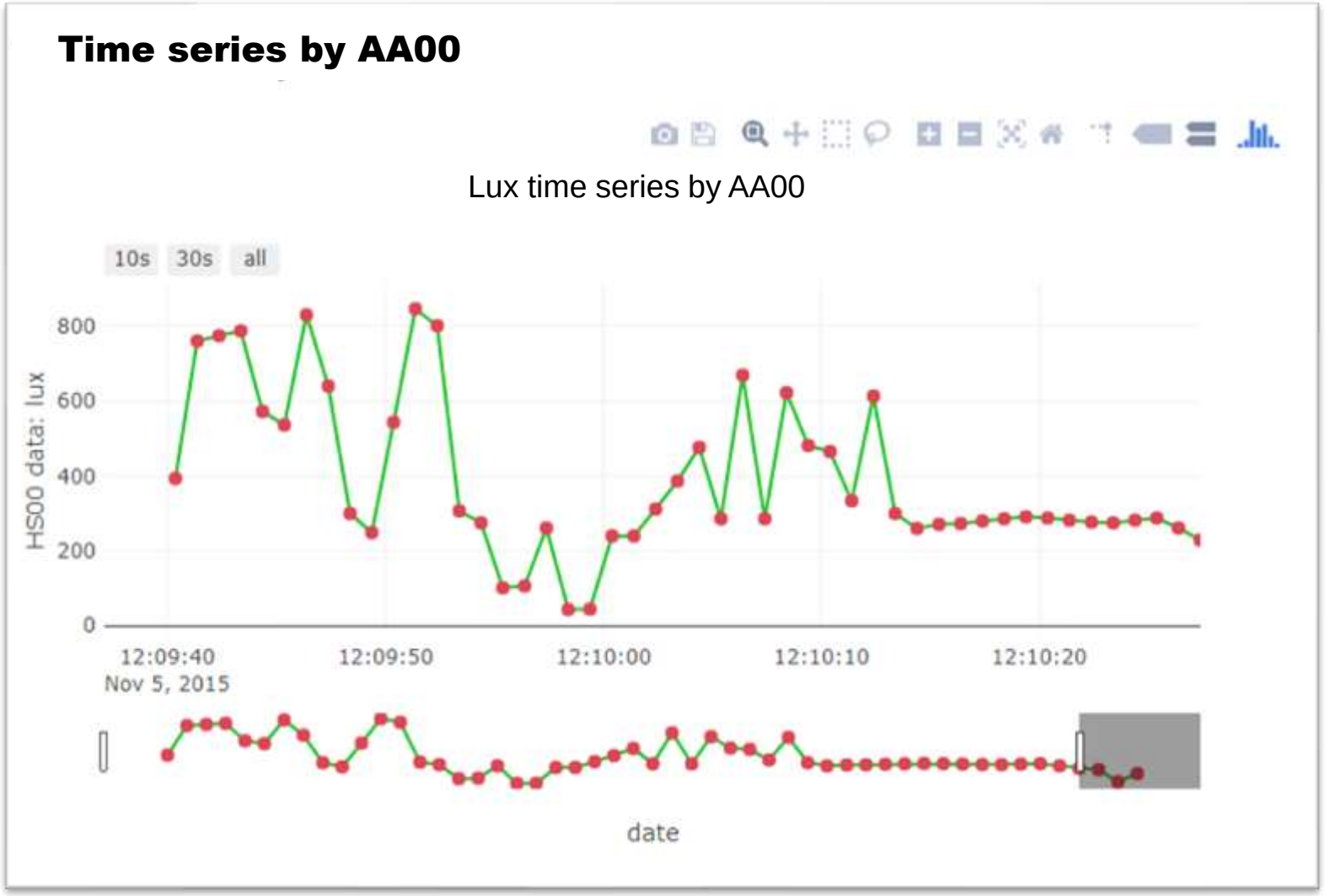
---

on Time: 2021-10-05 14:02:26.657

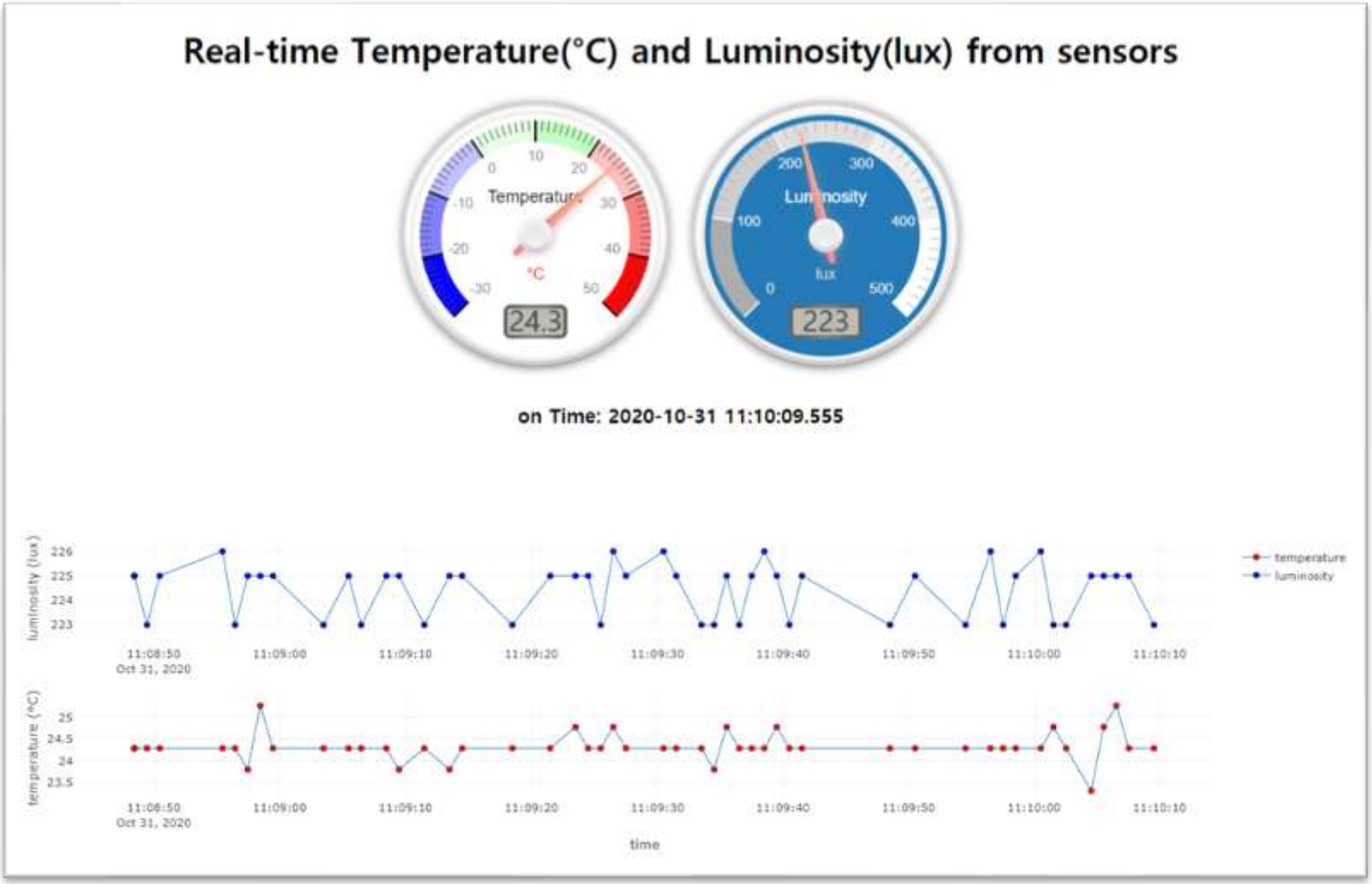
Signal (temp,lumi) : 25.27,84

Real-time monitoring of signals from Arduino tmp36 + CdS circuit

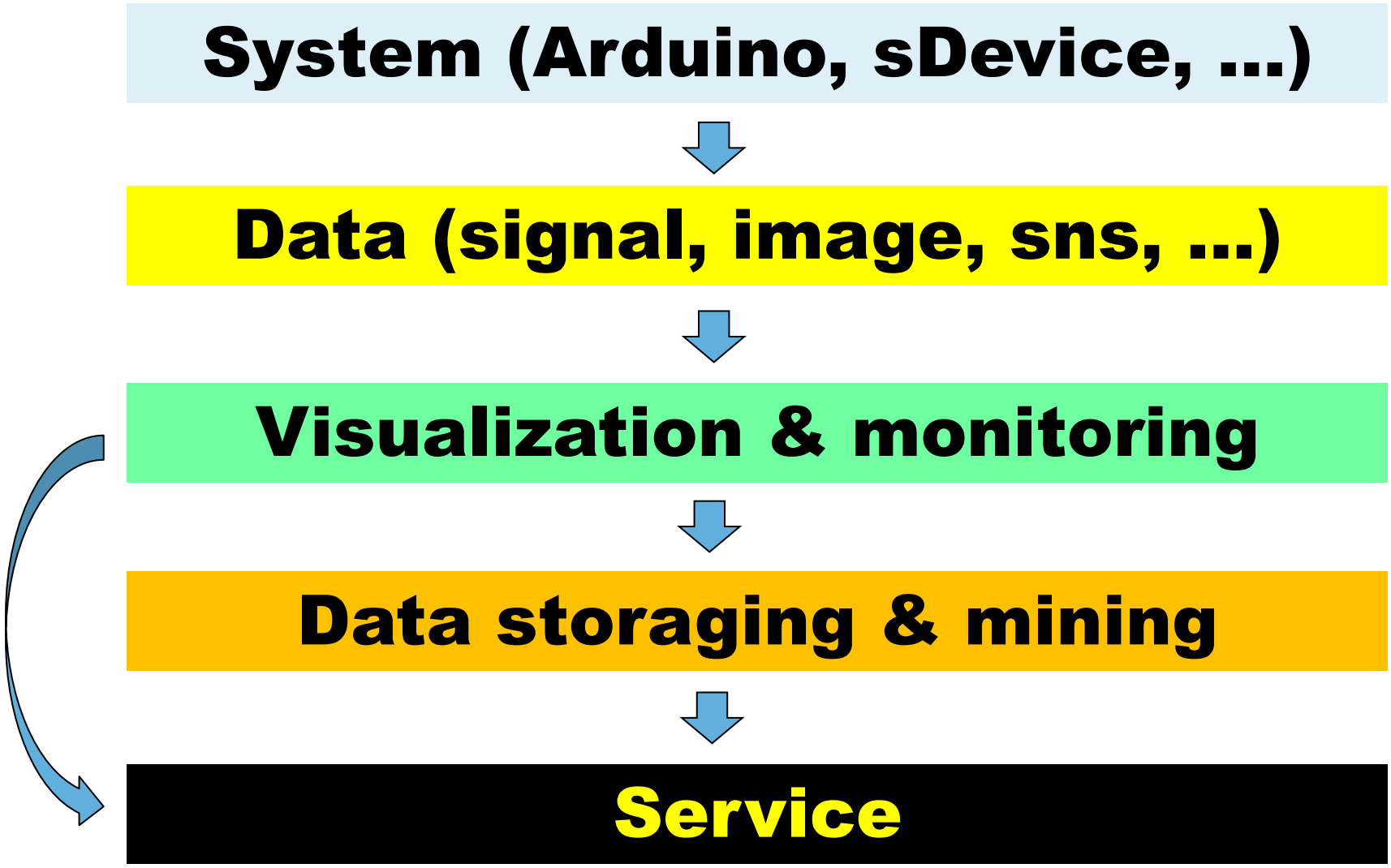
Arduino data + plotly.js



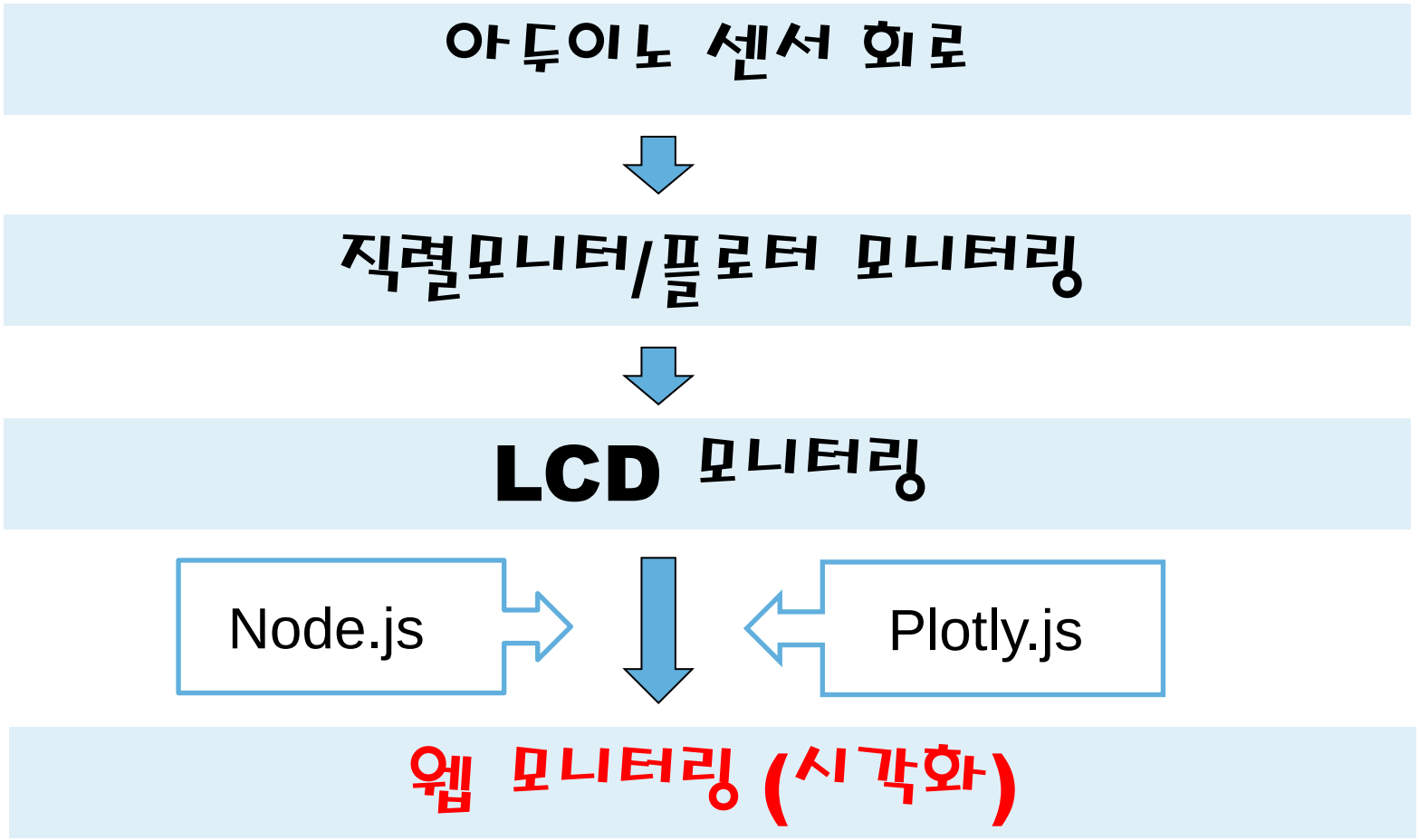
Arduino data + plotly.js + gauge.js



A5. Introduction to visualization



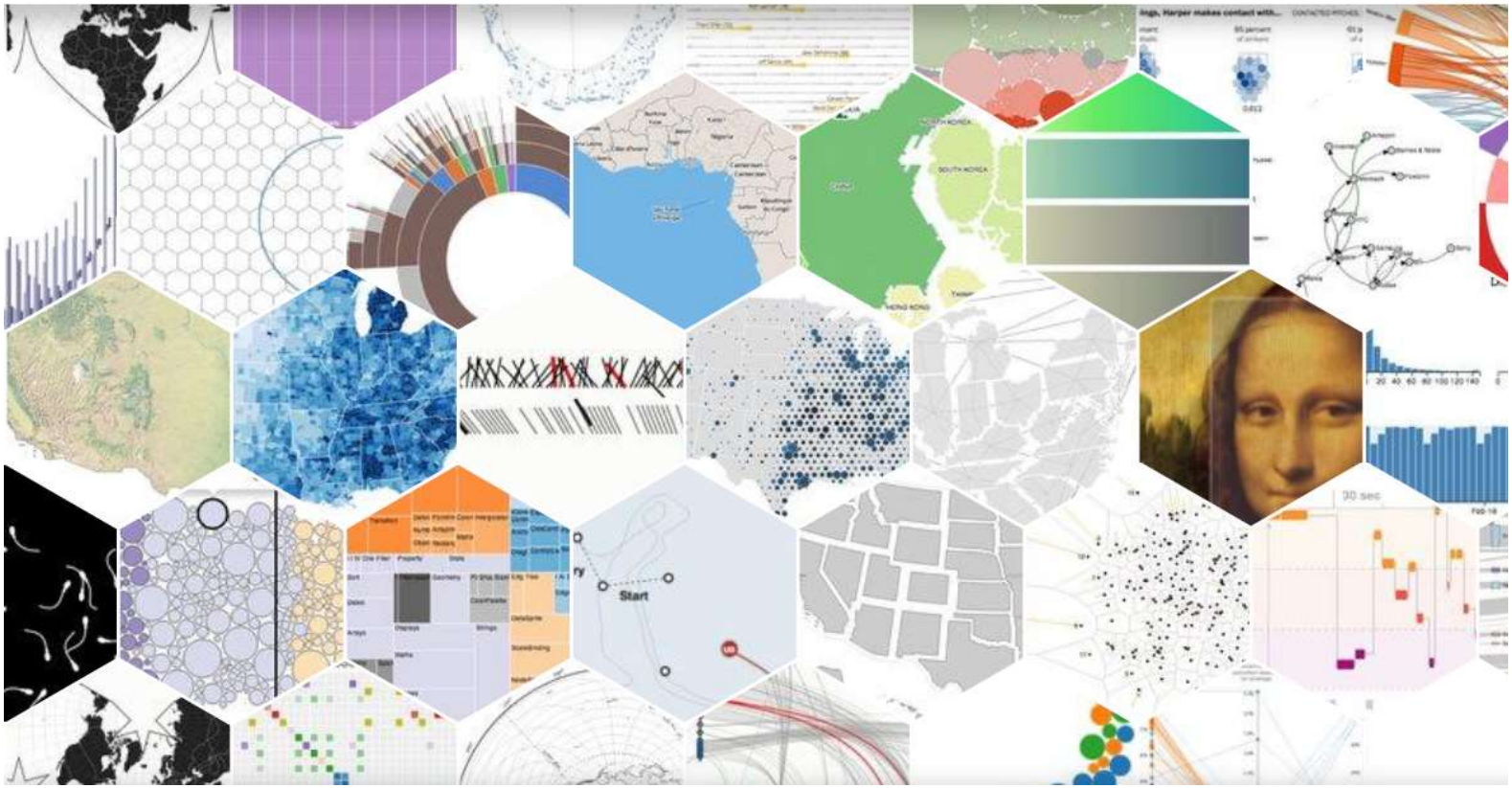
A5.1 Introduction to data visualization



A5.1.1 D3.js

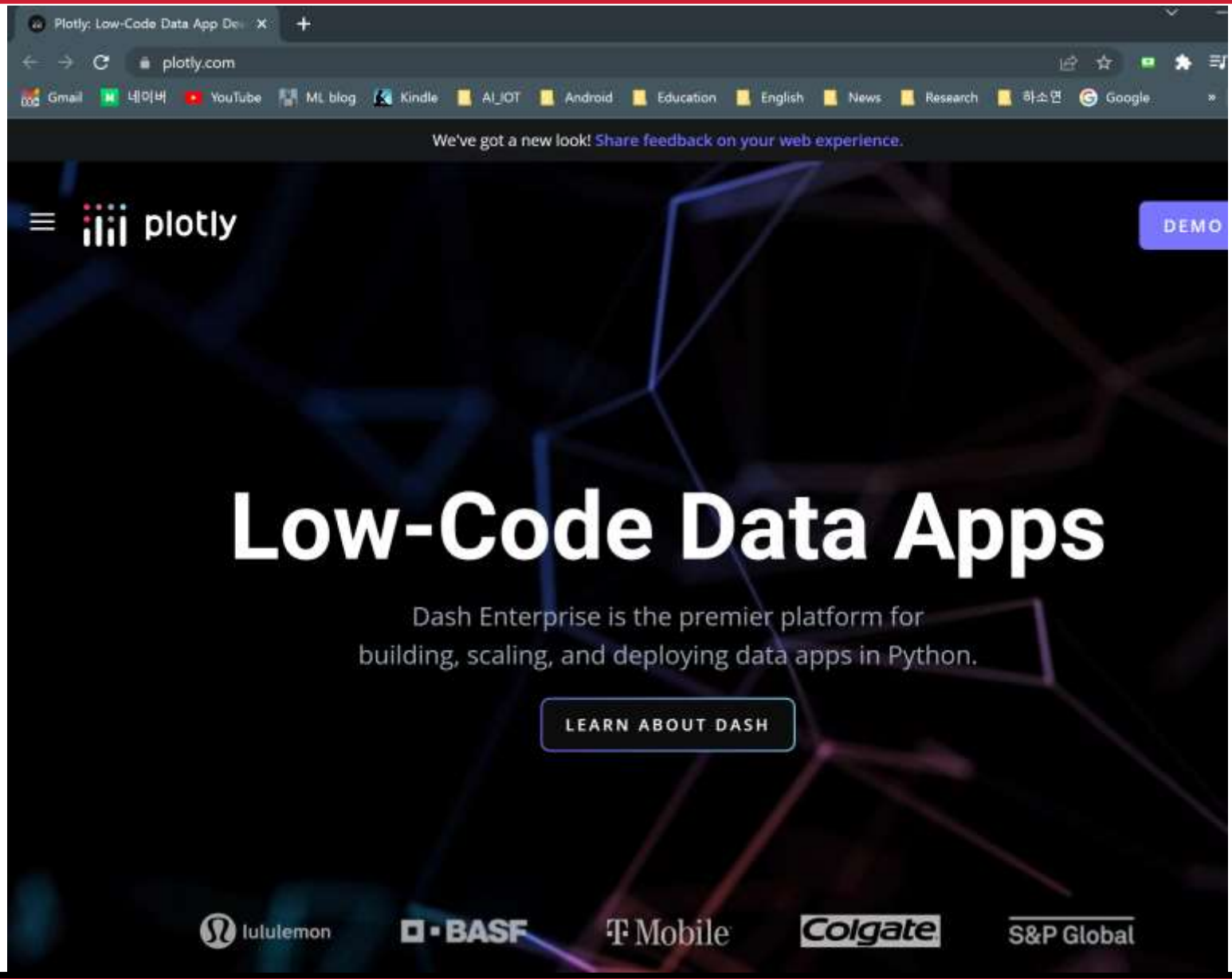
Overview Examples Documentation API Source

Data-Driven Documents



http://d3js.org/

A5.1.2 plotly



https://plotly.com/

A5.1.3 plotly.js

Built on top of d3.js and stack.gl,

Plotly.js is a high-level, declarative charting library.

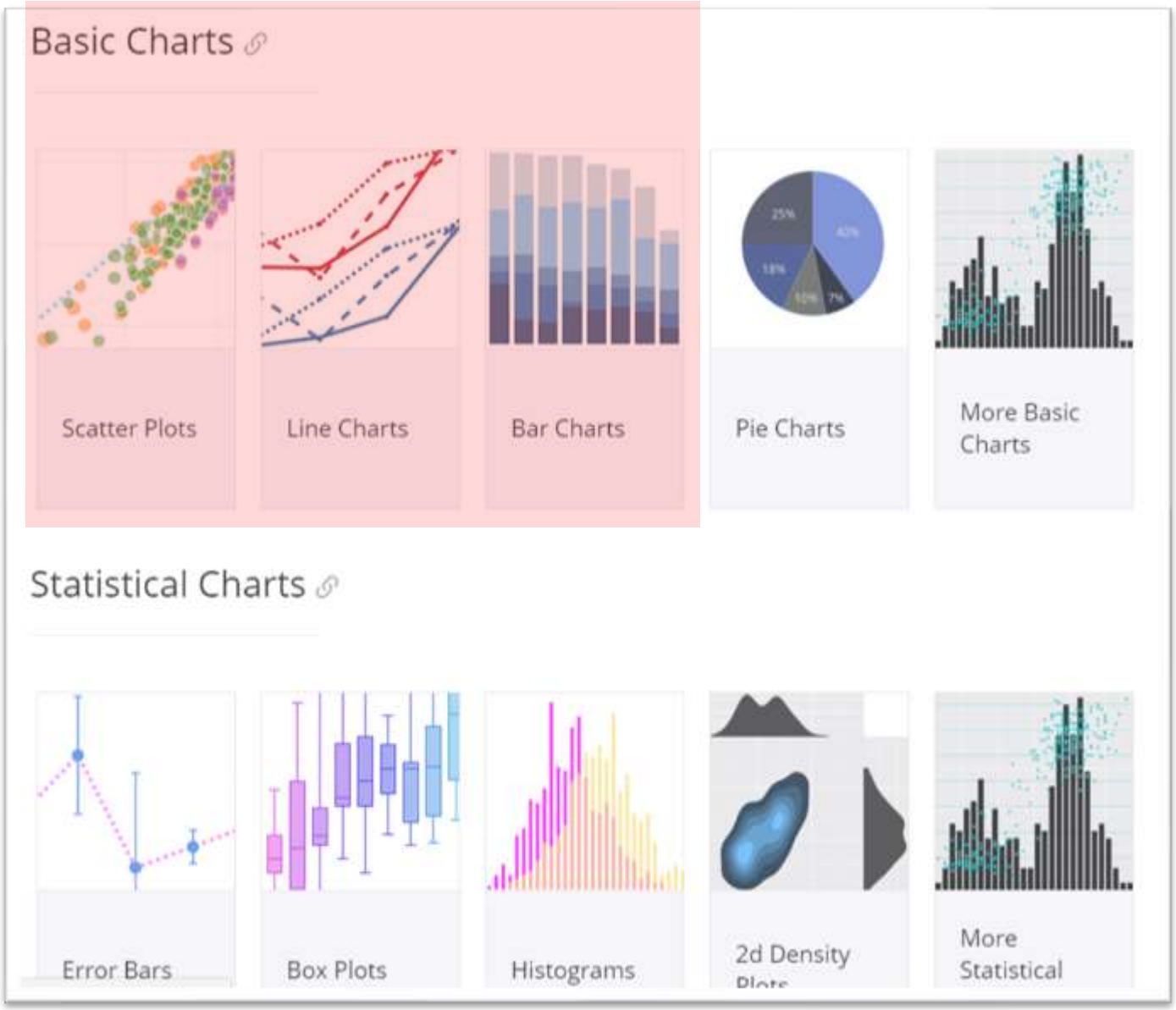
plotly.js ships with over 40 chart types, including 3D charts, statistical graphs, and SVG maps.

https://plot.ly/javascript/

A5.1.4 Introduction to plotly.js

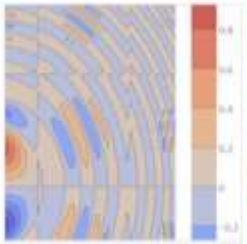
https://plotly.com/javascript/

A5.1.5 Introduction to plotly.js charts

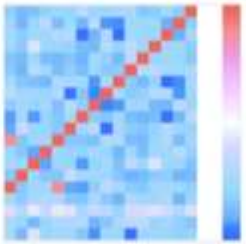


A5.1.6 Introduction to plotly.js charts

Scientific Charts

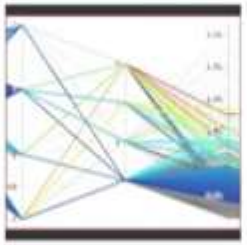


Contour Plots



Heatmaps

Ternary Plots

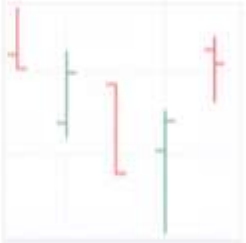


Parallel Coordinates Plot

More Scientific Charts

Financial Charts

Time Series



OHLC Charts

Candlestick Charts

A5.1.7 Introduction to plotly.js charts

Maps

Choropleth Maps

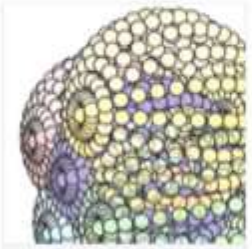
Scatter Plots on Maps

Bubble Maps

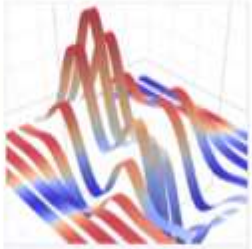
Lines on Maps

Scatter Plots on Mapbox

3D Charts



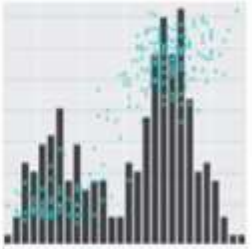
3D Scatter Plots



Ribbon Plots

3D Surface Plots

3D Mesh Plots



More 3D Charts

A5.1.8 plotly.js: time series & streaming

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

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

A5.1.9 Getting started: plotly.js

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

https://plotly.com/

A5.1.10 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.

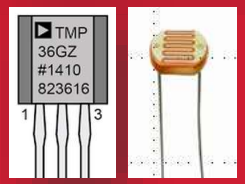
```
<head>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
```

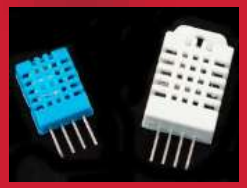
Else, if you want to get a specific version of plotly.js, say 1.2.0:

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

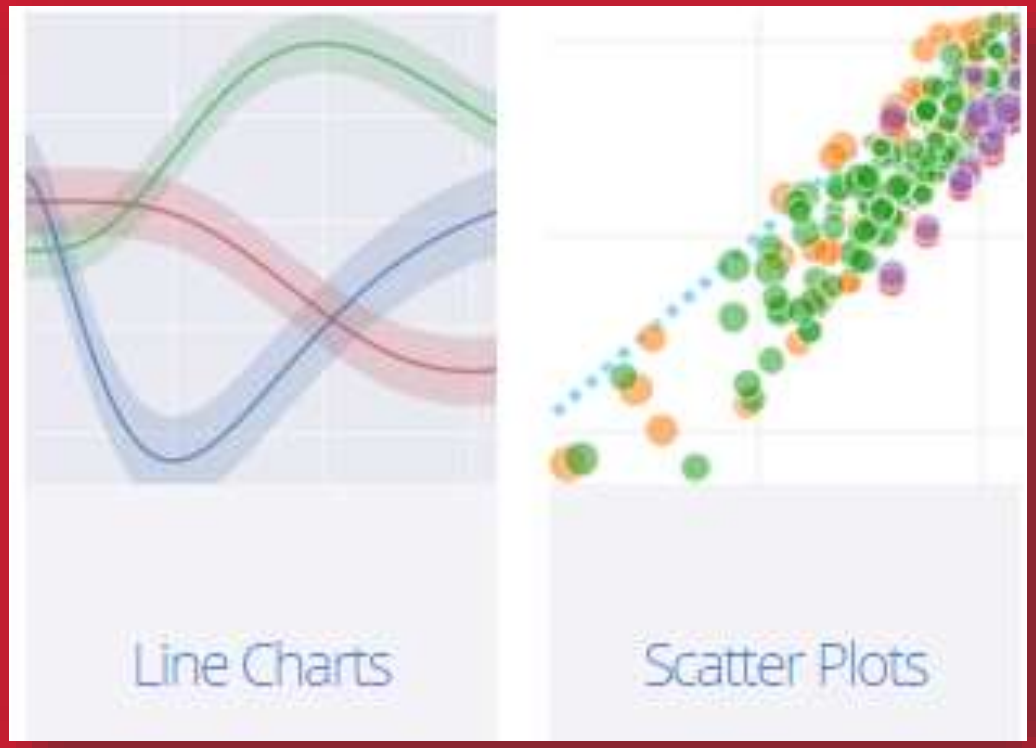
```
<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
```

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

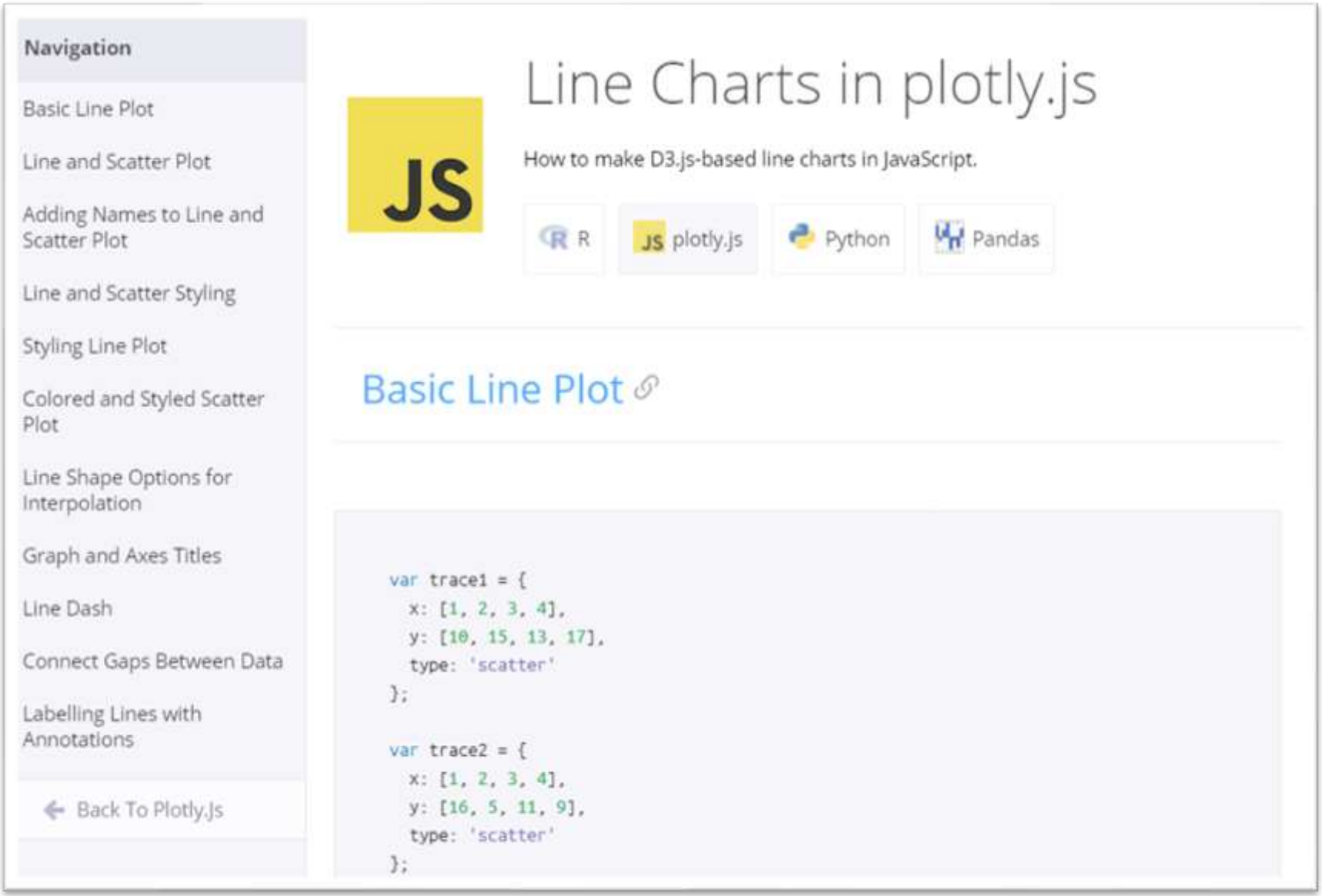




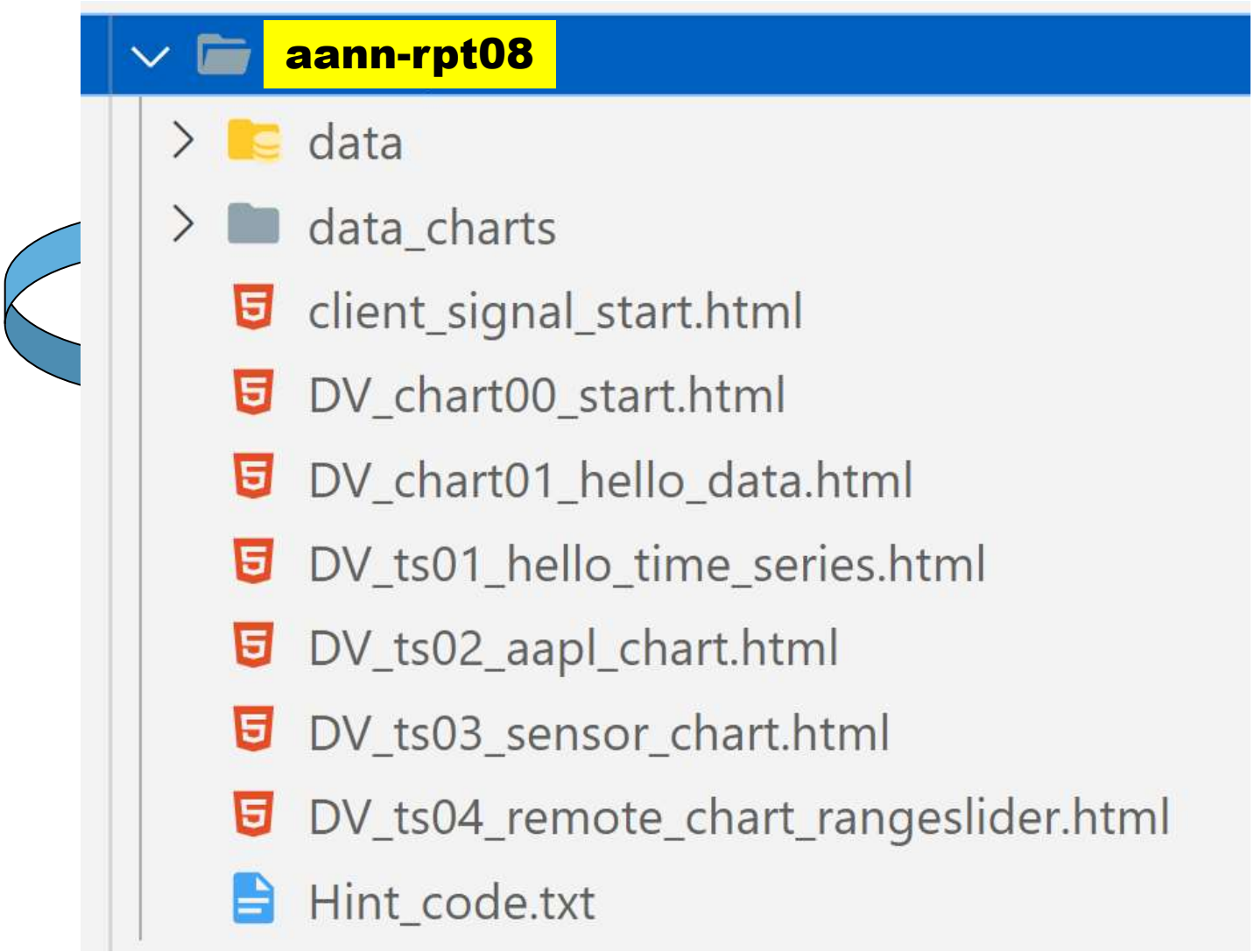
Data charts using plotly.js



A5.2 Data charts



A5.2.1 Working folders



A5.2.2.1 Starting plotly basic chart

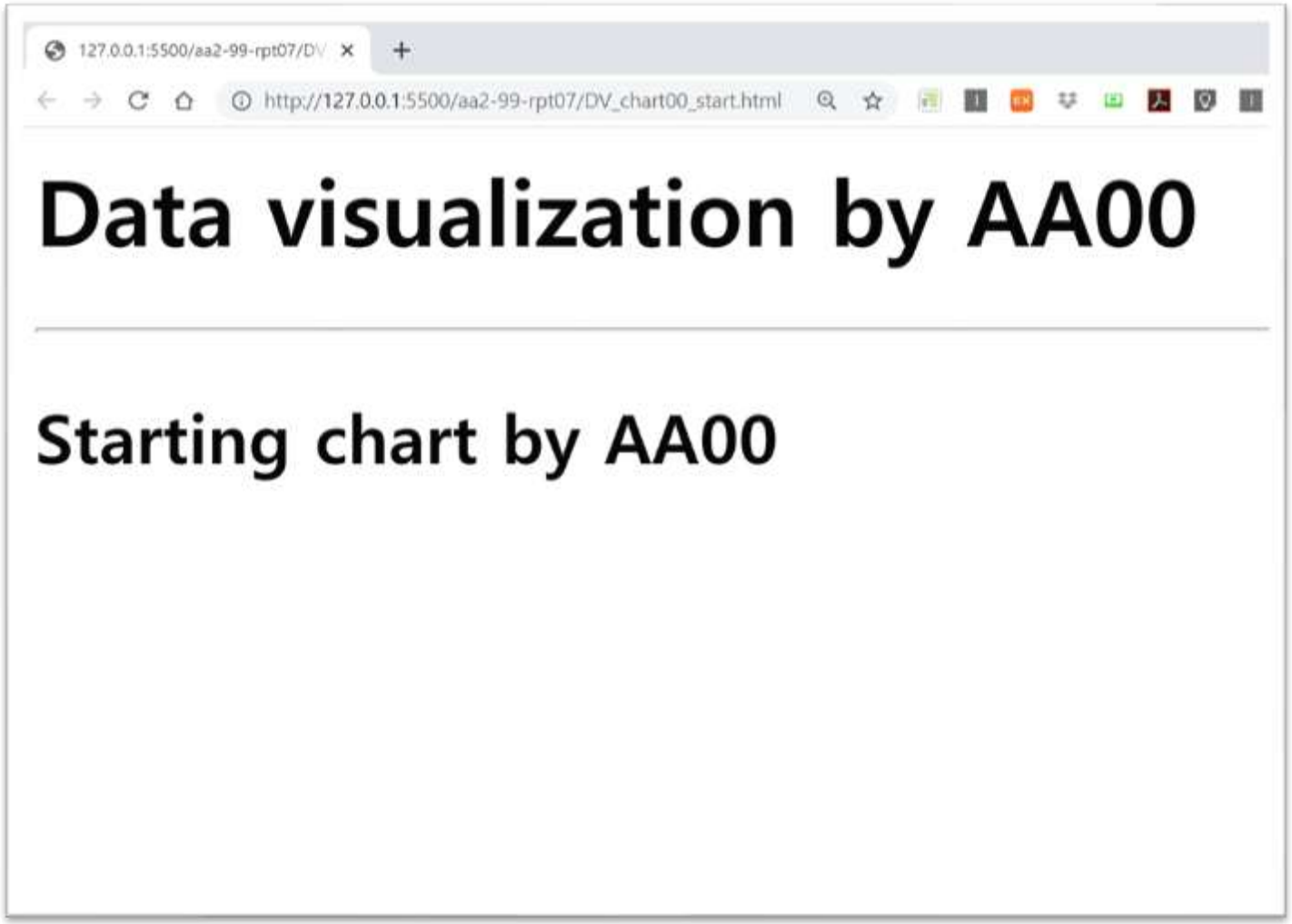
DV_chart00_start.html

Starting chart!

```
1  <html>
2  <head>
3      <meta charset="utf-8">
4      <!-- Plotly.js -->
5      <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
6  </head>
7  <body>
8      <h1>Data visualization by AA00</h1>
9      <hr>
10     <h2>Starting graph by AA00</h2>
11
12     <!-- Plotly chart will be drawn inside this DIV -->
13     <div id="myDiv" style="width: 500px;height: 300px"></div>
14
15     <script>
16         <!-- JAVASCRIPT CODE GOES HERE -->
17
18
19     </script>
20 </body>
21 </html>
22
```

A5.2.2.2 Starting plotly basic chart

VSCode, live server



A5.2.3.1 Hello plotly basic chart

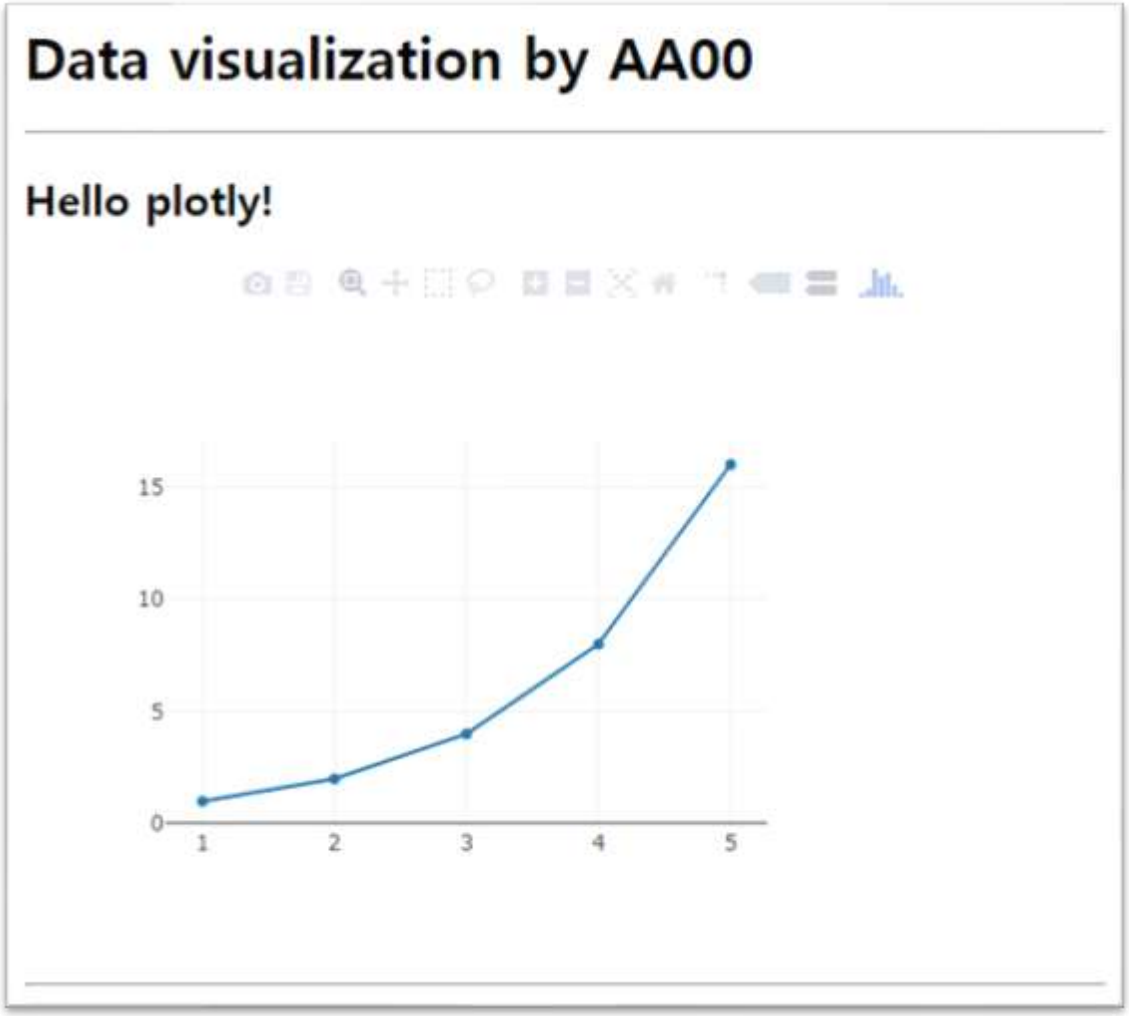
Hello plotly data chart!
DV_chart01_hello_data.html

```
1  <html>
2  <head>
3      <meta charset="utf-8">
4      <!-- Plotly.js -->
5      <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
6  </head>
7  <body>
8      <h1>Data visualization by AA00</h1>
9      <hr>
10     <h2>Hello plotly!</h2>
11     <!-- Plotly chart will be drawn inside this DIV -->
12     <div id="myDiv" style="width: 500px;height: 400px"></div>
13     <hr>
14     <script>
15         <!-- JAVASCRIPT CODE GOES HERE -->
16         var data = [
17         {
18             x: [1, 2, 3, 4, 5],
19             y: [1, 2, 4, 8, 16],
20             type: 'scatter'
21         }];
22
23         Plotly.newPlot('myDiv', data);
24
25     </script>
26 </body>
27 </html>
```

data는 무엇?
그래프 객체들의 구조,
데이터 배열

A5.2.3.2 Hello plotly basic chart

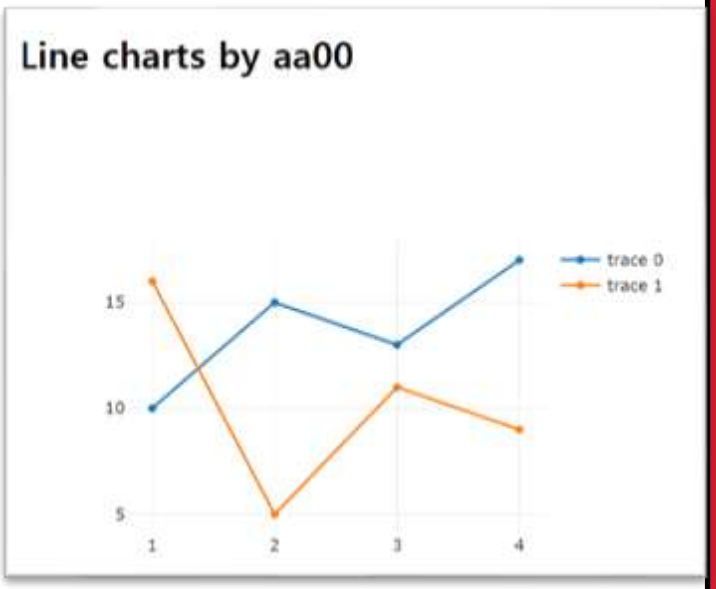
Graph : Hello plotly chart!



A5.2.4 plotly.js: Line Charts

[1] Basic multi-line charts ➔ DV_chart02_two_traces.html

```
<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>
```

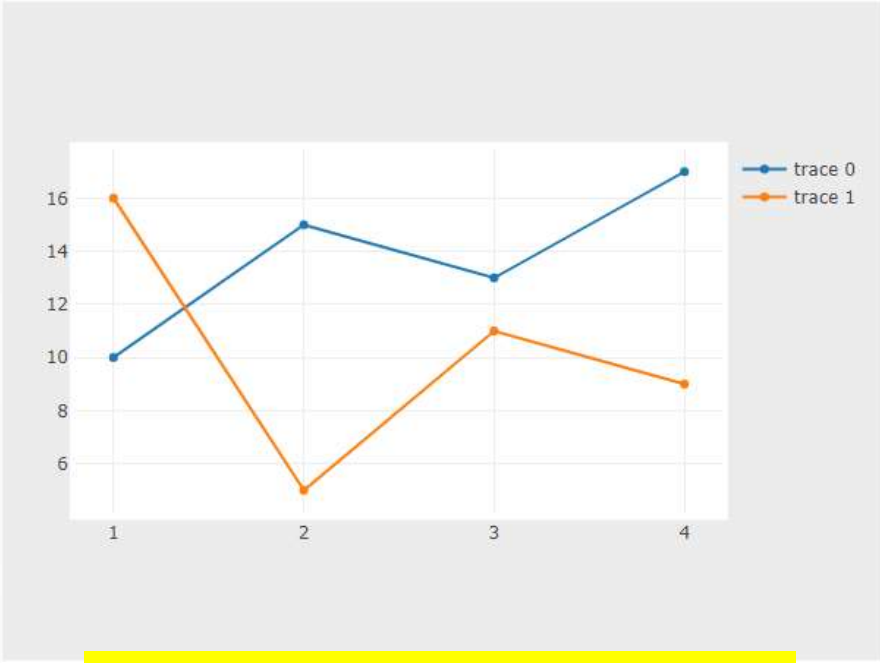


A5.2.5 plotly.js: Line Charts

[2] Basic line charts with layout ➔ DV_chart03_layout.html

```
var layout = {
    autosize: false,
    width: 600,
    height: 450,
    margin: {
        l: 50,  // left
        r: 50,  // right
        b: 100, // bottom
        t: 100, // top
        pad: 4  // padding
    },
    paper_bgcolor: '#ececec',
    plot_bgcolor: '#ffffff' //'#rrggbb'
};

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

Line charts with layout by AA00



Test: pad → 40

Save
AAnn_Chart_Layout.png

A5.2.6.1 plotly.js: Line & Scatter plot

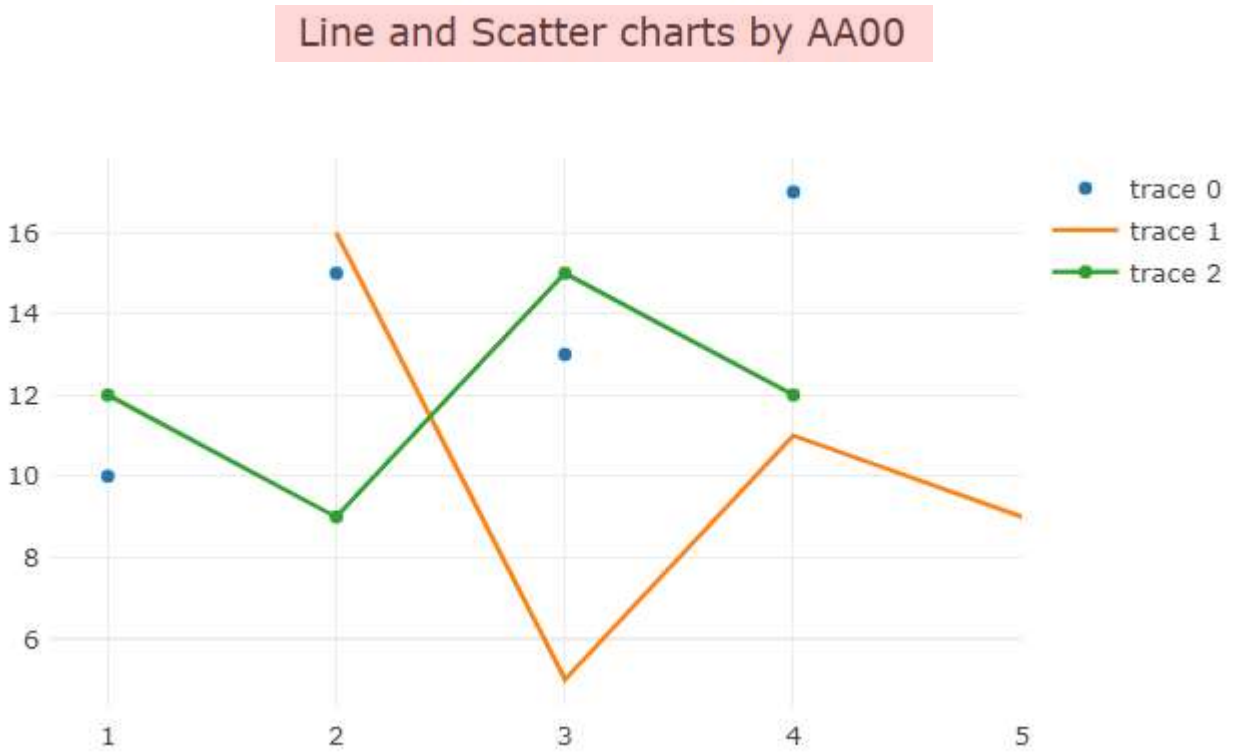
[3] Line & scatter plot : setting mode ➔ DV_chart04_modes.html

```
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 charts by AA00',
    width: 600,
    height: 450,
    margin: {
        l: 50,
        r: 50,
        b: 100,
        t: 100,
        pad: 4
    },
};

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

A5.2.6.2 plotly.js: Line & Scatter plot

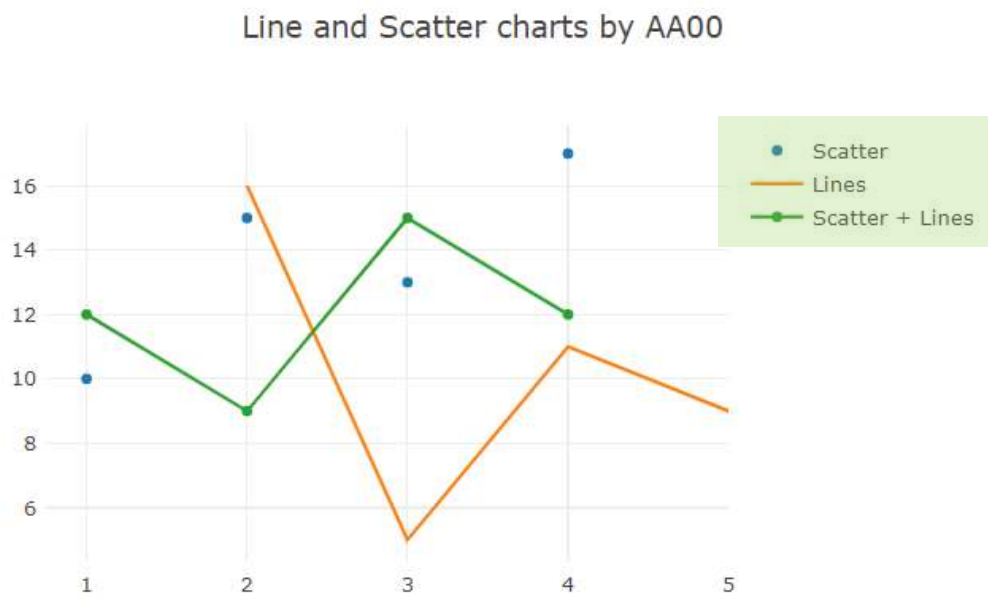
[3.1] Line & scatter plot with mode & title



A5.2.6.3 plotly.js: Line & Scatter plot

[3.2] Line & scatter plot with axis name ➔ DV_chart05_axis_name.html

```
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'
};
```

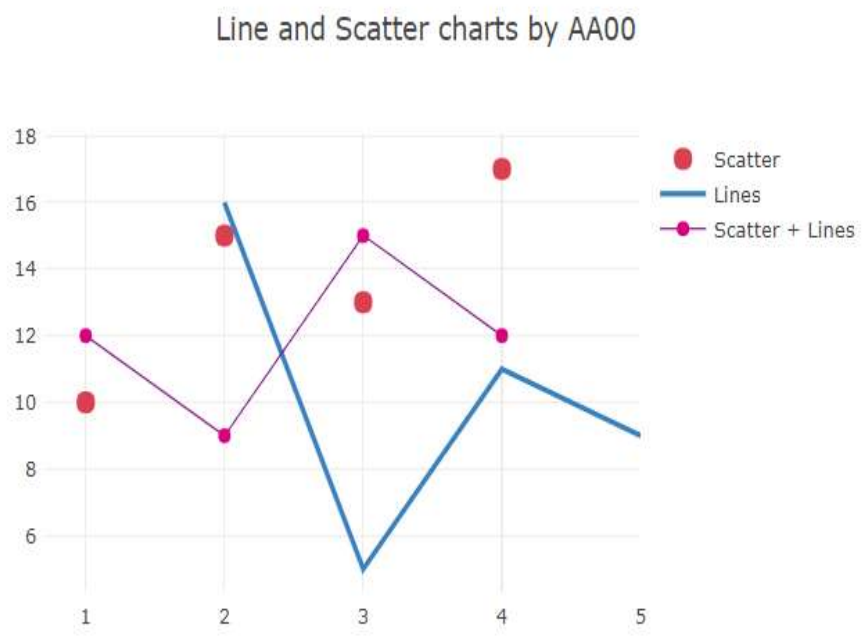


A5.2.6.4 plotly.js: Line & Scatter plot

[3.3] Line & scatter plot with style ➔ DV_chart06_style.html

```
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
  }
};
```

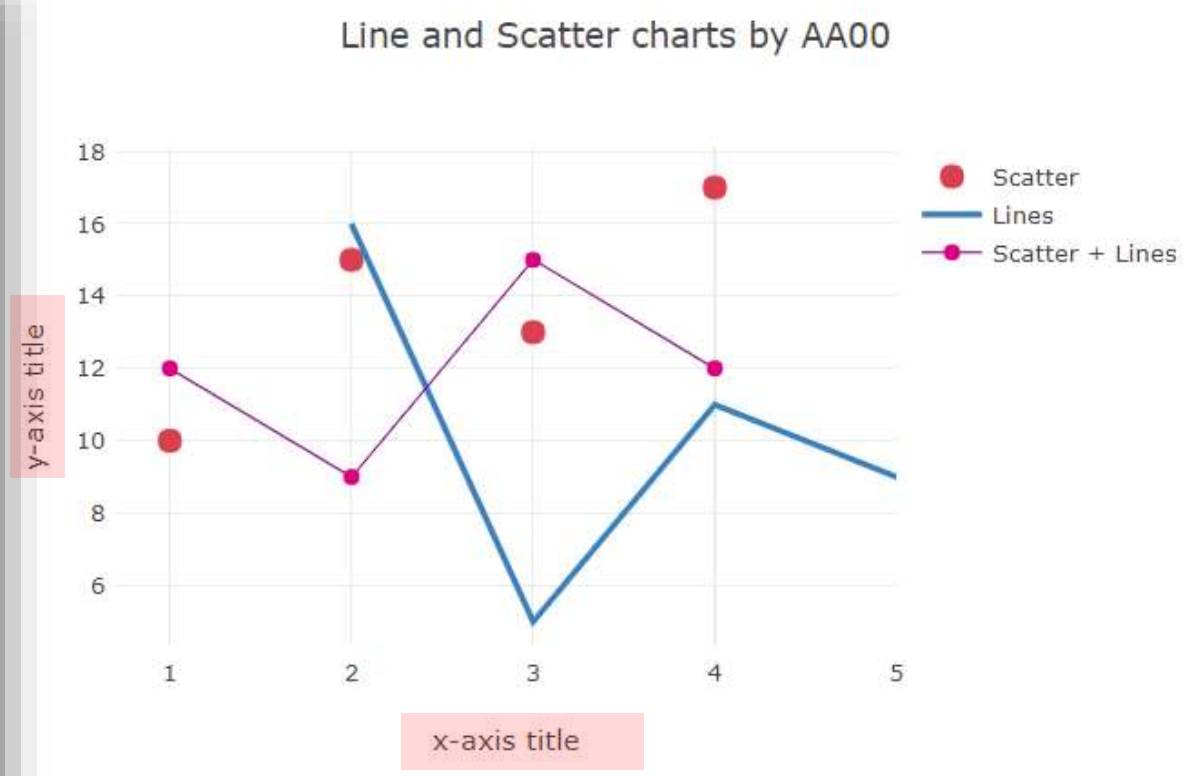
```
var trace3 = {
  x: [1, 2, 3, 4],
  y: [12, 9, 15, 12],
  mode: 'lines+markers',
  name: 'Scatter + Lines',
  marker: {
    color: 'rgb(128, 0, 128)',
    size: 8
  },
  line: {
    color: 'rgb(128, 0, 128)',
    width: 1
  }
};
```



A5.2.6.5 plotly.js: Line & Scatter plot

[3.4] Line & scatter plot with axis titles ➔ DV_chart07_axis_title.html

```
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'
  }
};
```



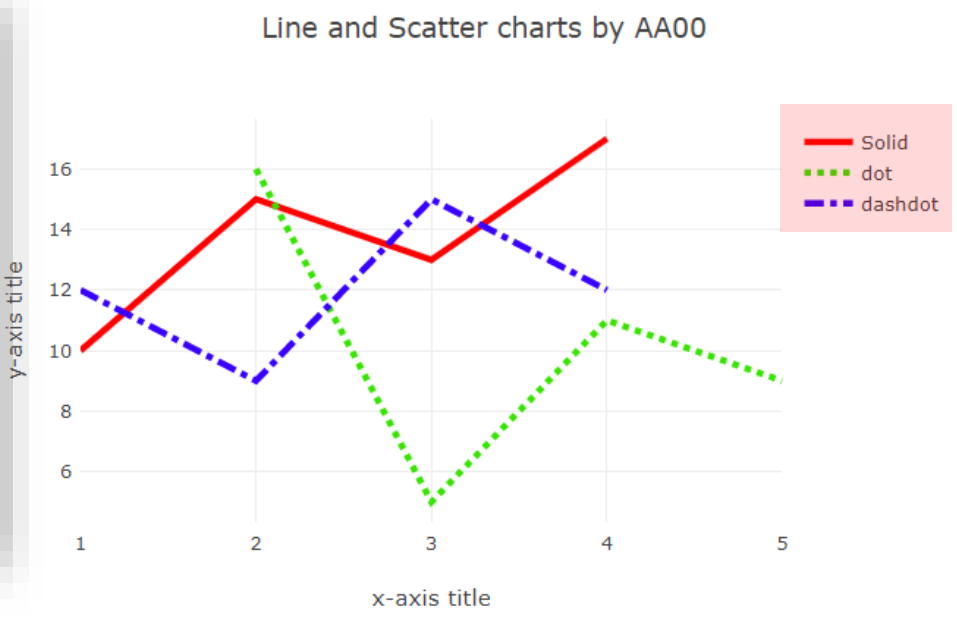
Save
AAnn_Axis_Title.png

A5.2.6.6 plotly.js: Line & Scatter plot

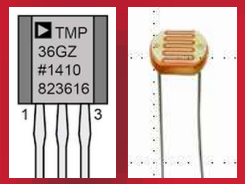
[3.5] Line & scatter plot with dash and dot ➔ DV_chart08_line_style.html

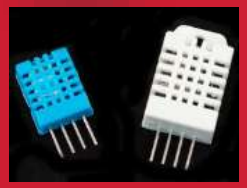
```
var trace1 = {
  x: [1, 2, 3, 4],
  y: [10, 15, 13, 17],
  mode: 'lines',
  name: 'Solid',
  line: {
    color: 'rgb(255, 0, 0)',
    dash: 'solid',
    width: 4
  }
};

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
  }
};
```



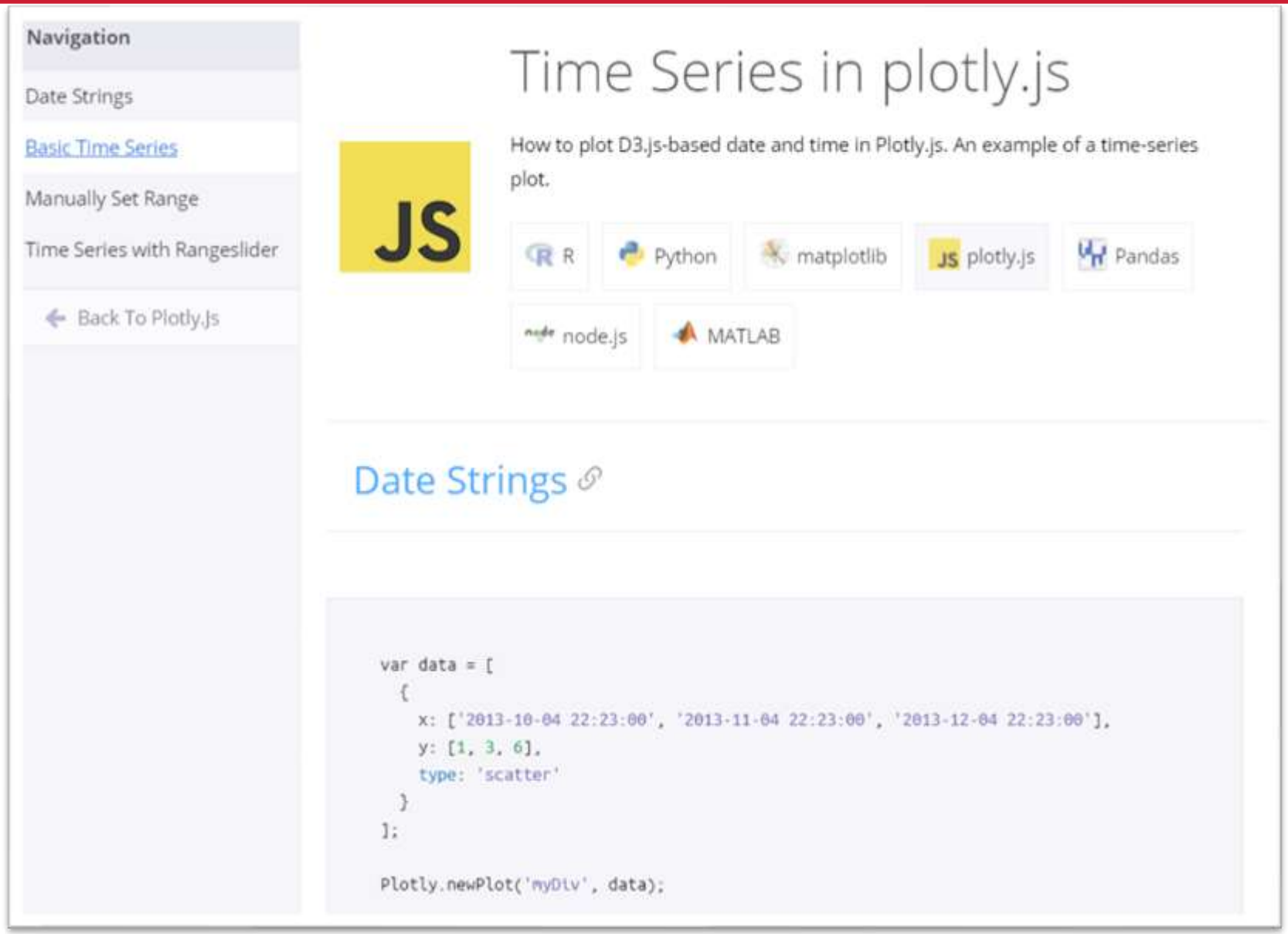
Save
AAnn_Line_Dash_Dot.png





Data visualization using plotly.js

A5.3. Time series



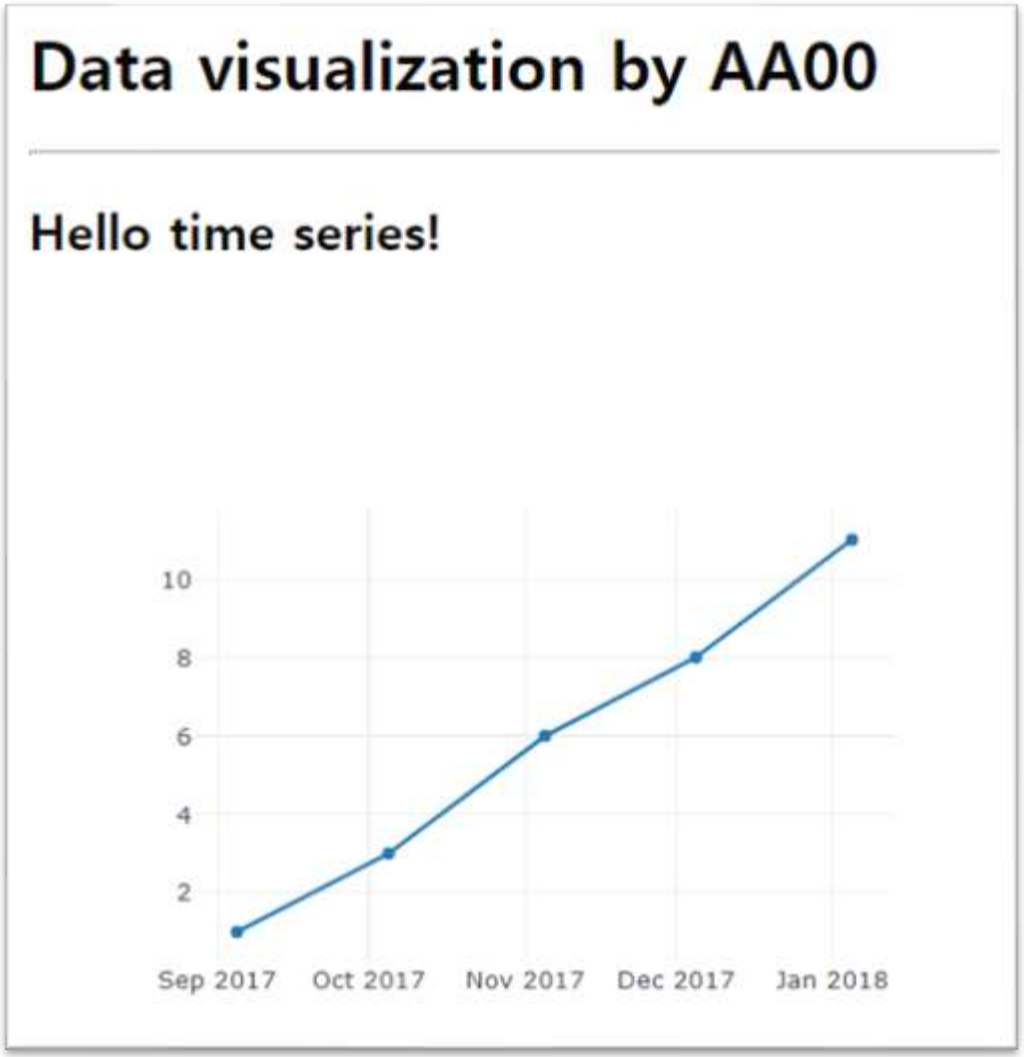
A5.3.1 plotly.js: Time series

[1] Time series : date 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',
        '2018-01-04 22:23:00'],
        y: [1, 3, 6, 8, 11],
        type: 'scatter'
    }
    ];

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

A5.3.2 plotly.js: Time series

Time series : date strings ➔ DV_ts01_hello_time_series.html



[DIY]

오늘 날자와 데이터를 추가

A5.3.3.1 plotly.js: Time series

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

안전함 | https://raw.githubusercontent.com/plotly/datasets/master/finance-charts-a...

```
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
```

A5.3.3.2 plotly.js: Time series

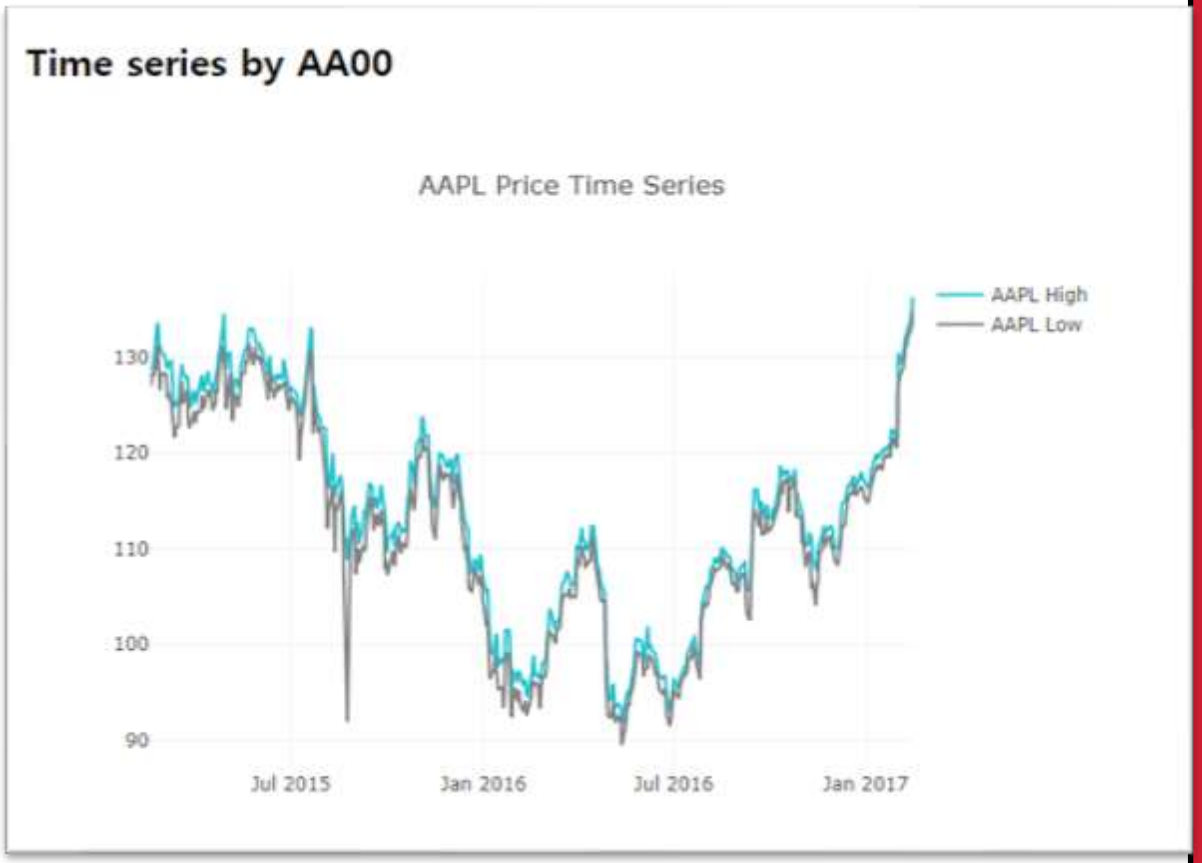
[2] Time series : financial data strings – AAPL stock price

```
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];
```

A5.3.3.3 plotly.js: Time series

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

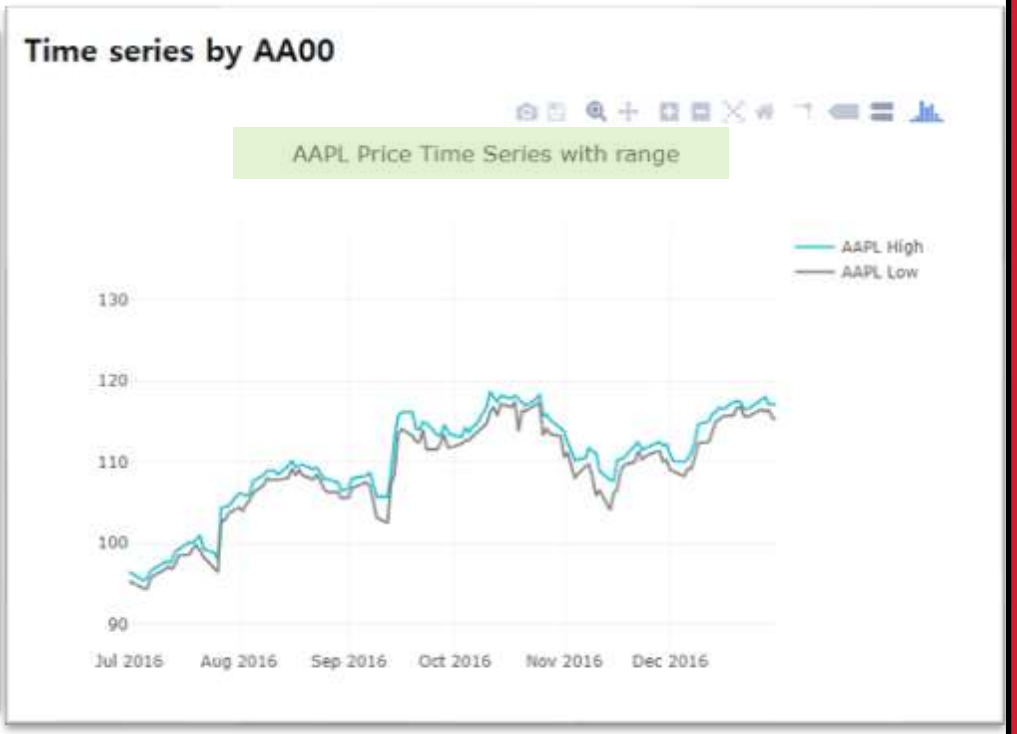
```
var data = [trace1,trace2];

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

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



A5.3.3.4 plotly.js: Time series

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

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

var layout = {
    title: 'AAPL Price Time Series with range',
    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);
```



날짜와 주가의 범위를 지정

➔ 2016년 하반기

A5.3.3.5 plotly.js: Time series

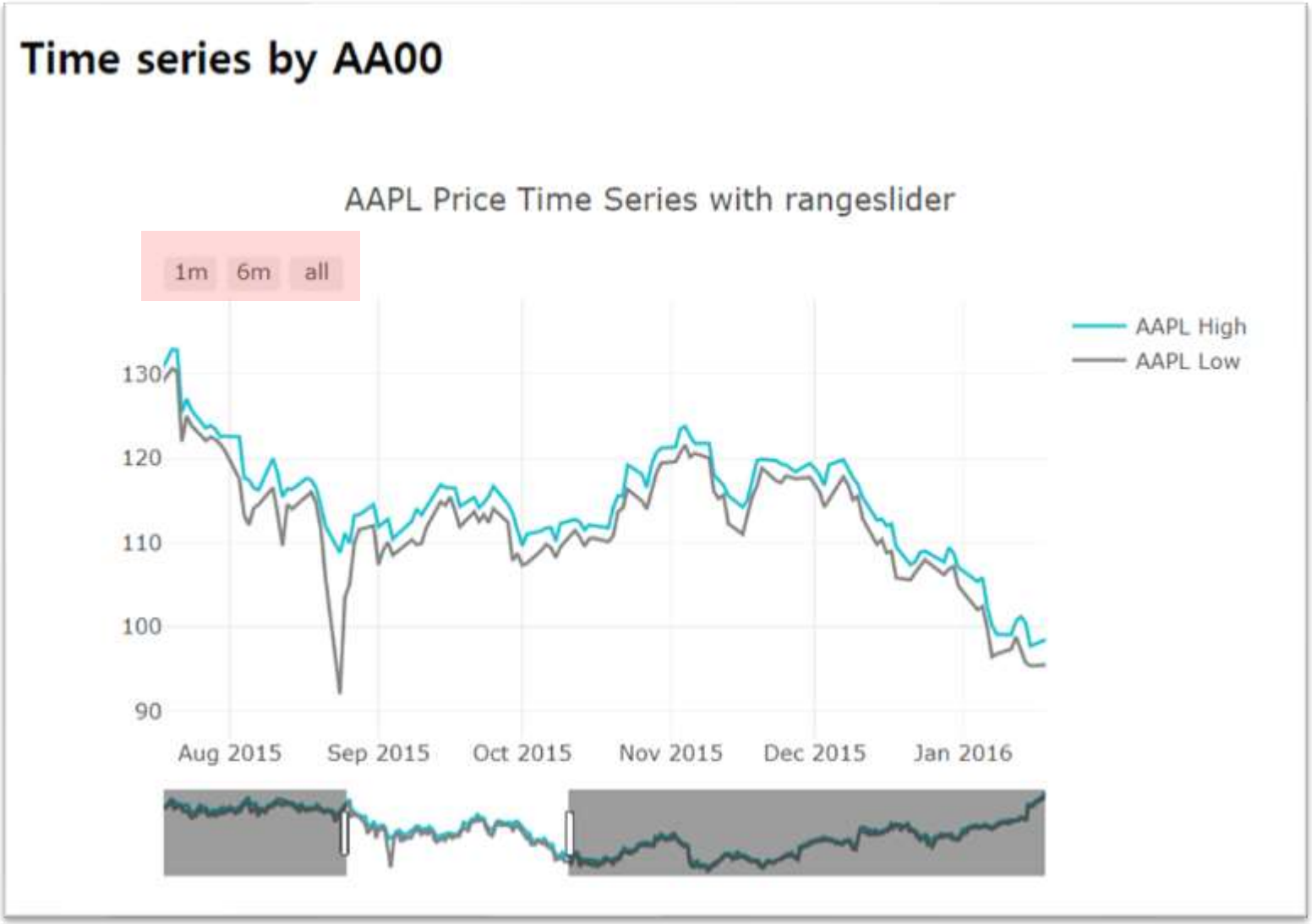
[2] Time series : financial data strings – Range slider

➔ DV_ts03_aapl_chart_range_slider.html

```
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'
        }
    };
```

A5.3.3.6 plotly.js: Time series

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

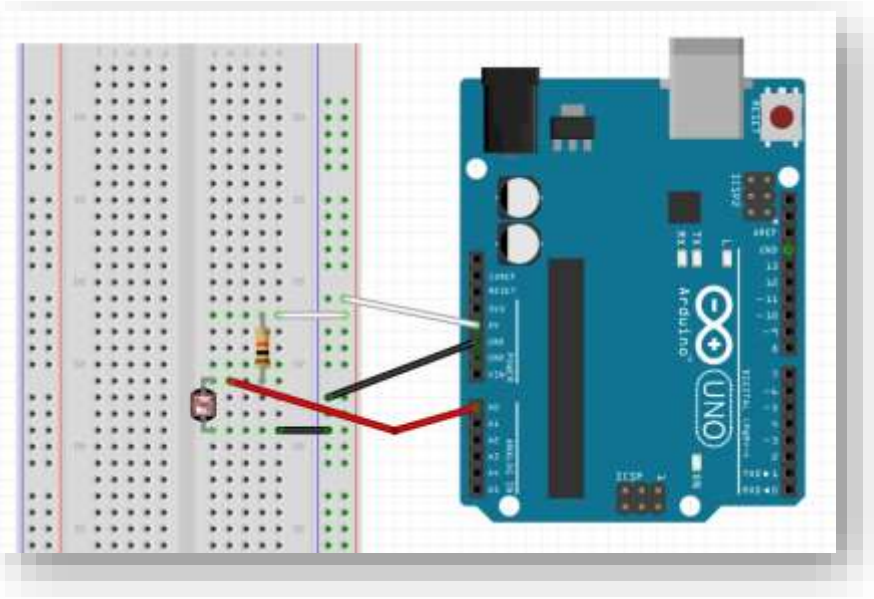


A5.3.4.1 plotly.js: Sensor time series

[3] Time series : my lux data

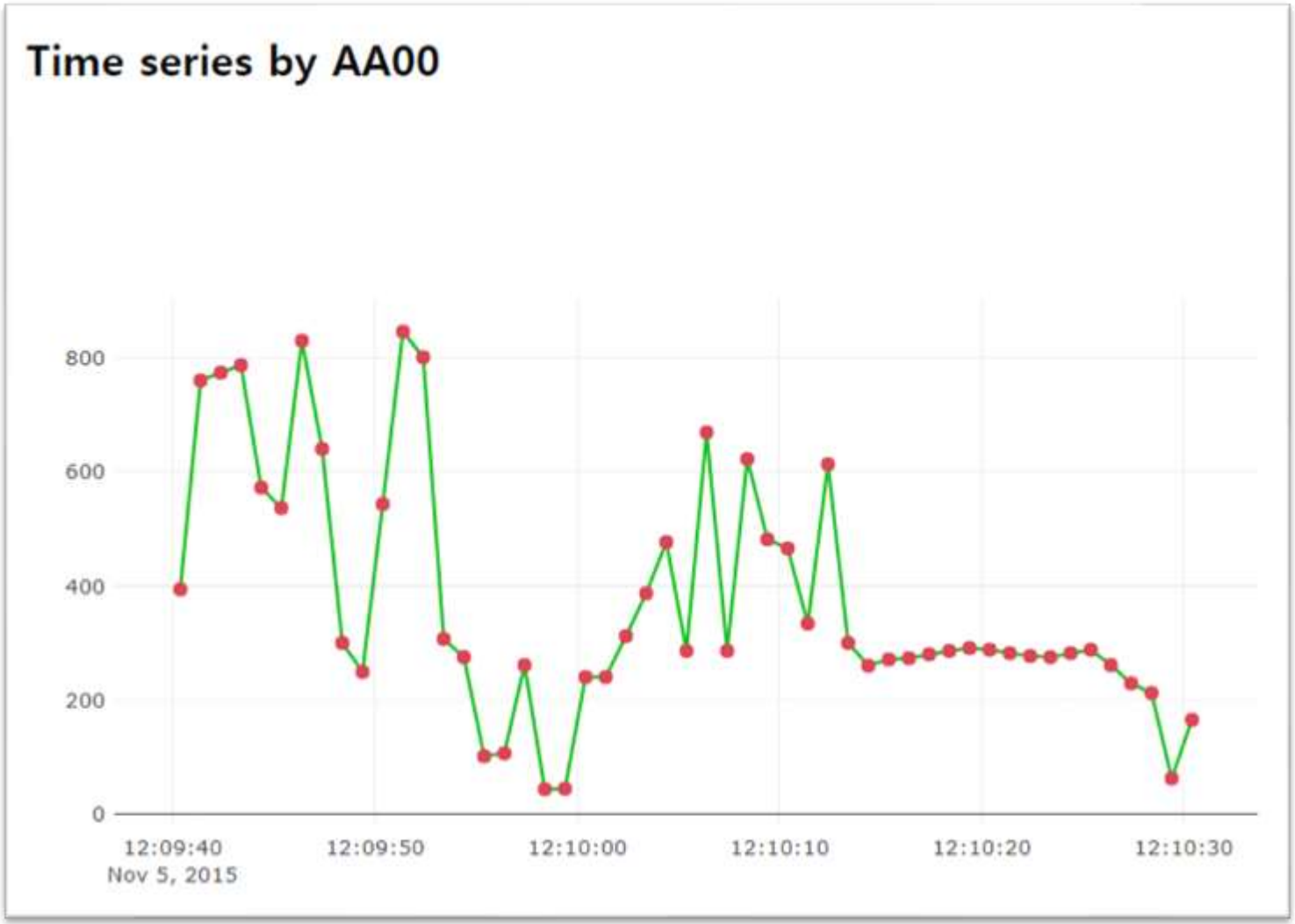
```
'2015-11-05 12:09:41.382', 394,
'2015-11-05 12:09:42.380', 761,
'2015-11-05 12:09:43.378', 775,
'2015-11-05 12:09:44.377', 788,
'2015-11-05 12:09:45.375', 573,
'2015-11-05 12:09:46.389', 537,
'2015-11-05 12:09:47.388', 831,
'2015-11-05 12:09:48.386', 641,
'2015-11-05 12:09:49.384', 300,
'2015-11-05 12:09:50.383', 249,
'2015-11-05 12:09:51.381', 544,
'2015-11-05 12:09:52.380', 847,
'2015-11-05 12:09:53.394', 802,
'2015-11-05 12:09:54.392', 307,
'2015-11-05 12:09:55.391', 275,
'2015-11-05 12:09:56.389', 101,
'2015-11-05 12:09:57.387', 106,
'2015-11-05 12:09:58.386', 261,
'2015-11-05 12:09:59.384', 43 ,
'2015-11-05 12:10:00.398', 44 ,
'2015-11-05 12:10:01.397', 240,
'2015-11-05 12:10:02.395', 240,
'2015-11-05 12:10:03.394', 312,
```

Data :
date,value



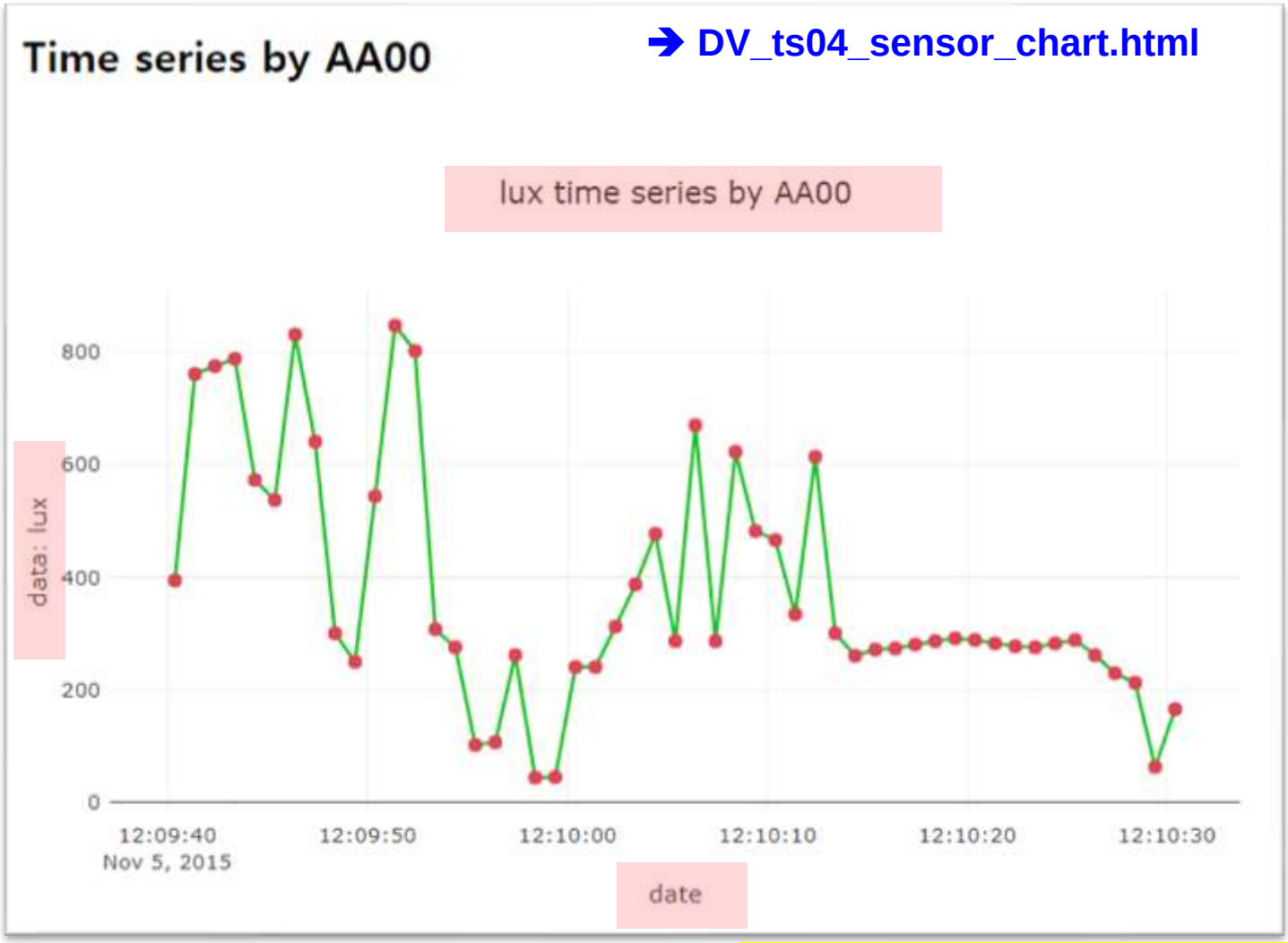
A5.3.4.2 plotly.js: Time series

[3] Time series : my lux data → DV_ts03_sensor_chart.html



A5.3.4.3 plotly.js: Time series

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



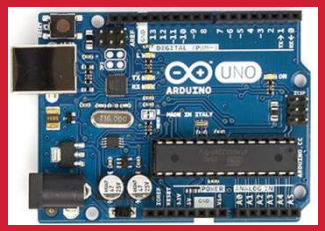
→ DV_ts04_sensor_chart.html

AAnn_lux_Time_Series.png

Project: Time series with Rangeslider

[Project-DIY] ➔ DV_ts05_sensor_Rangeslider.html

AAnn_lux_Rangelslider.png



[Practice]

- [wk08]
  - charts by plotly
  - Complete your project
  - Upload folder: aann-rpt08
  - Use repo “aann” in github

wk08 : Practice : aann-rpt08

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

제출폴더명 : **aann-rpt08**

- 압축할 파일들

① AAnn_Chart_Layout.png

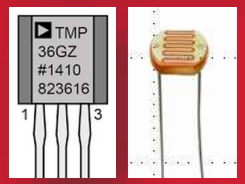
② AAnn_Axis_Title.png

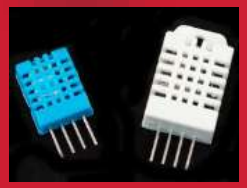
③ AAnn_Line_Dash_Dot.png

④ AAnn_lux_Time_Series.png

⑤ AAnn_lux_Rangeslider.png

⑥ All *.html in data_charts folder





Data visualization using plotly.js

Lecture materials

References & good sites

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

Target of this class