

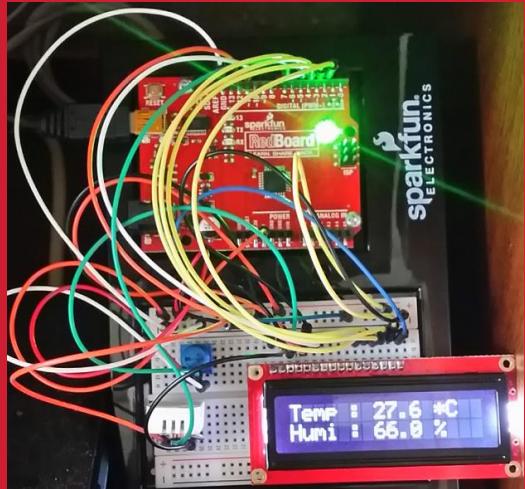


HW-SW-Connectivity

[wk14]

Arduino & NodeJS V

on Time: 2015-09-02 12:48:14.192

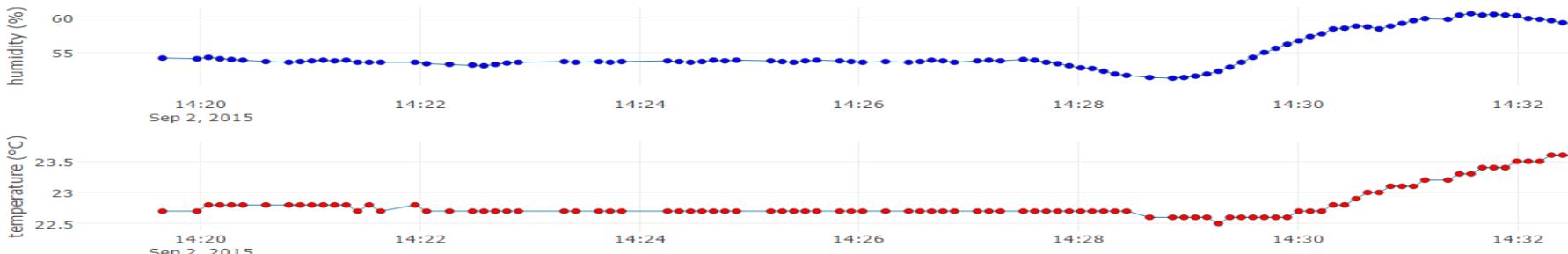


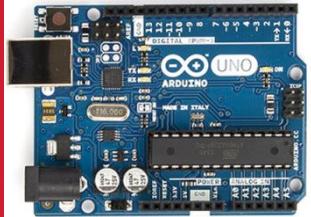
Basic HW and SW Integration using
Arduino & Javascript

COMSI, INJE University

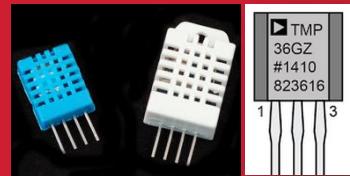
2nd semester, 2017

Email : yish@inje.ac.kr





[Practice]



◆ [wk13]

- RT Data Visualization with node.js
- Usage of gauge.js
- Complete your real-time LUX charts
- AAnn_Rpt10.zip

wk13 : Practice-10 : AAnn_Rpt10.zip

◆ [Target of this week]

- Complete your plots of real-time streaming of luminosity (lux).
- Design your own gauge and chart.
- Save your outcomes and compress 3 figures

제출파일명 : **AAnn_Rpt10.zip**

- 압축할 파일들

- ① **AAnn_DS_30timestamps.png**
- ② **AAnn_Idr_change.png**
- ③ **AAnn_Idr_gauge.png**

Email : **chaos21c@gmail.com**



results

[DIY 2] Client html : client_ldr_gauge.html (change design of Gauge)

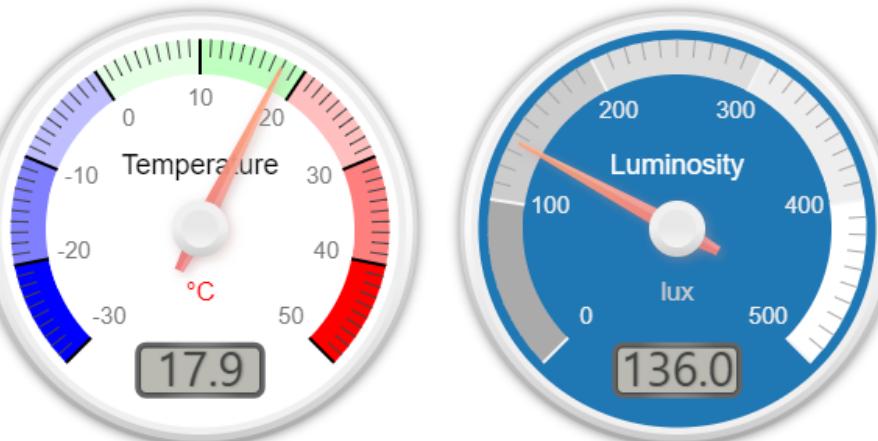


Data visualization : plotly.js

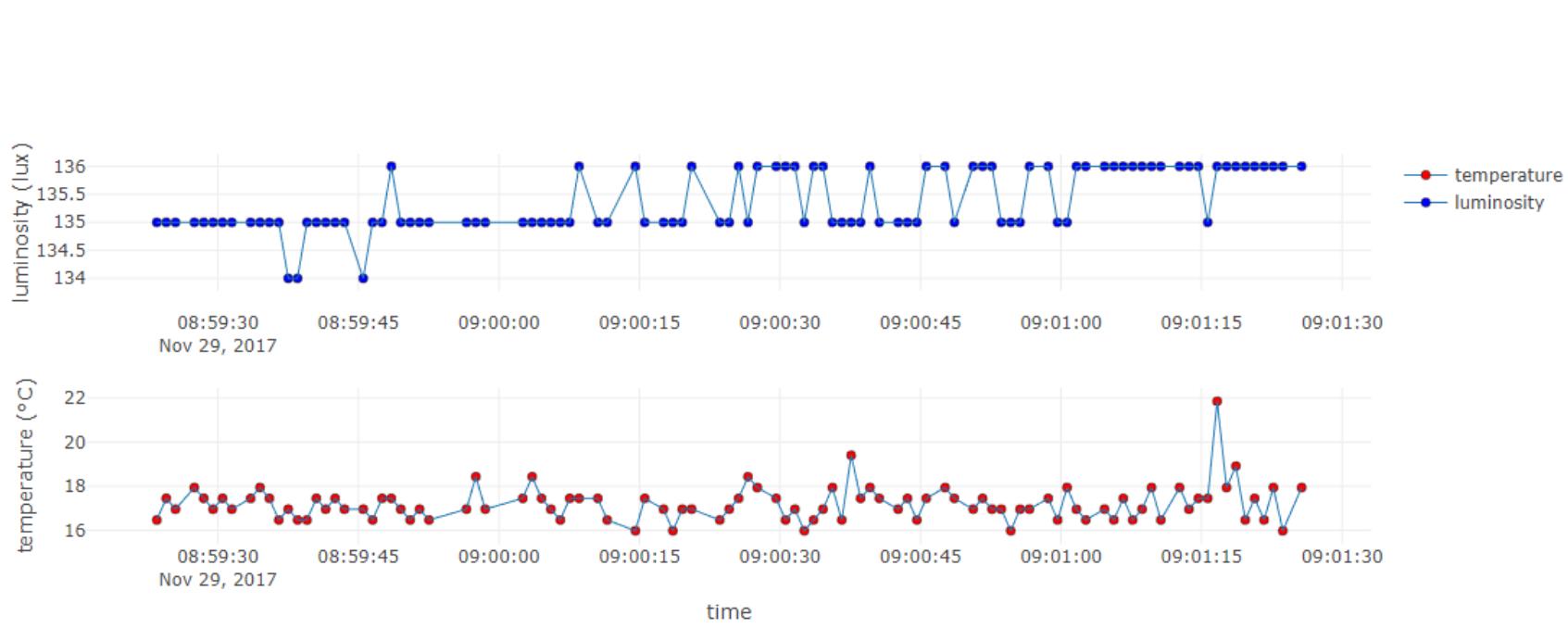
Time series by AAnn



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



on Time: 2017-11-29 09:01:25.675

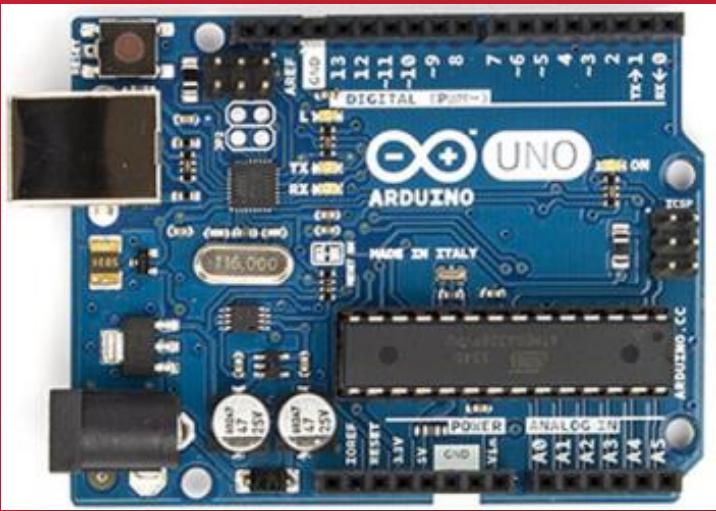




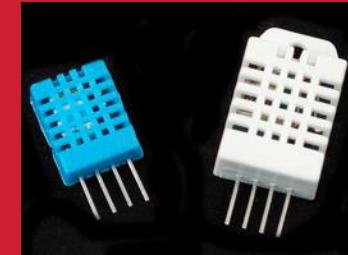
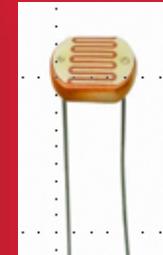
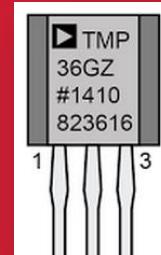
Arduino

[Home](#)[Buy](#)[Download](#)[Products](#) ▾[Learning](#) ▾[Forum](#)[Support](#) ▾[Blog](#)

<https://www.arduino.cc/>



Arduino & Node.js





A6.1. Introduction to visualization

System (Arduino, sDevice, ...)



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



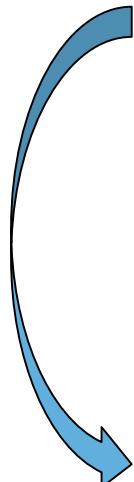
Visualization & monitoring



Data storing & mining



Service





A6.1.8 Introduction to plot.ly



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



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



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>

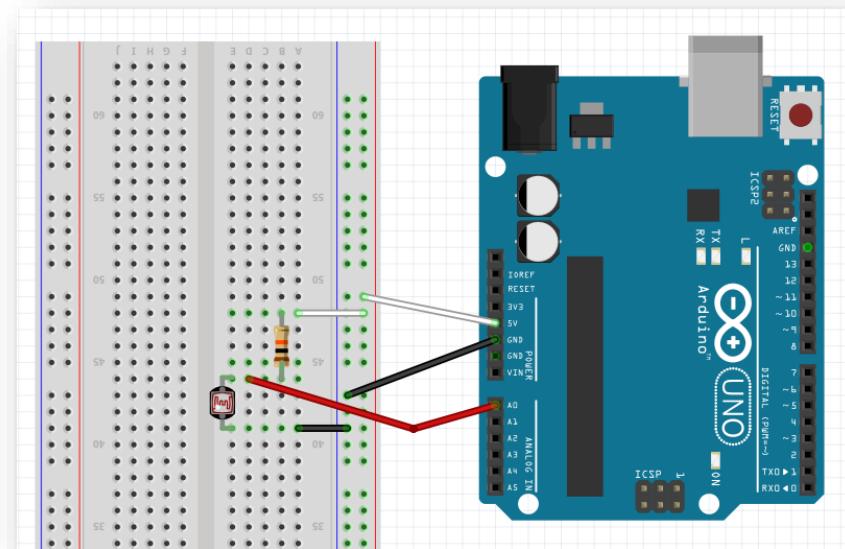


A6.3.9 plotly.js: Time series

[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



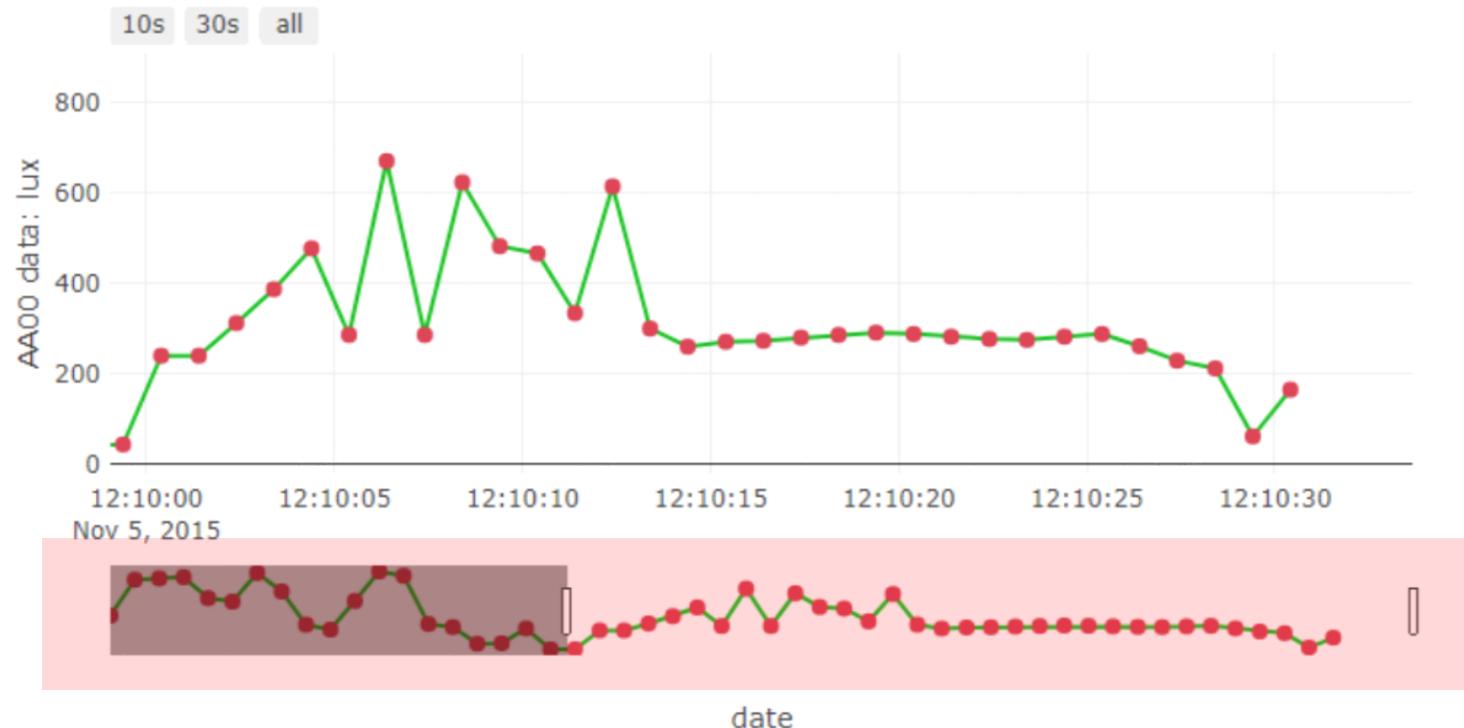


Project: Time series with Rangelslider

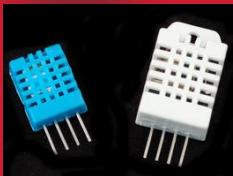
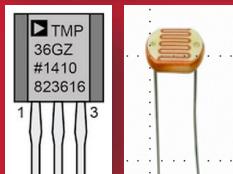
[Project-DIY] AAnn_lux_Rangelslider.html

Time series by AAnn

lux time series by AAnn



AAnn_lux_Rangelslider.html

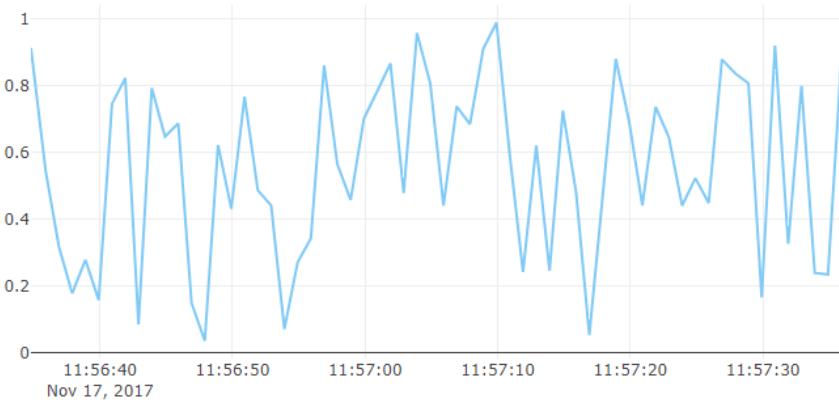


Data Streaming using ploy.ly



Streaming Charts

Streaming data with timestamp





A6.4.7 plotly.js: Streaming data

[5] Streaming data using 30 points update

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

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

```
Plotly.plot('graph', [
    y: newArray,
    mode: 'lines',
    line: {color: '#80CAF6'}
]);
```

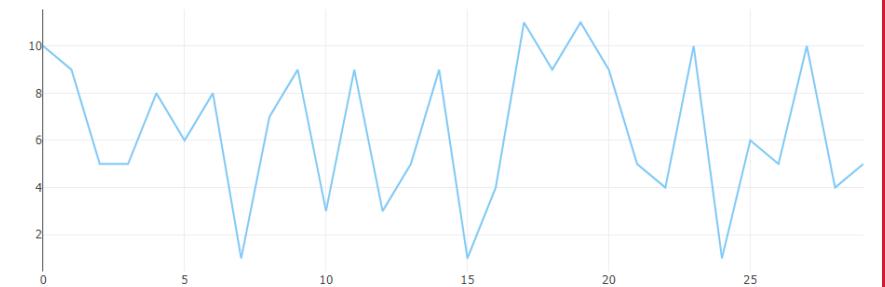
```
var cnt = 0;
var interval = setInterval(function() {

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

    var data_update = {
        y: [newArray]
    };

    Plotly.update('graph', data_update)
    //cnt++;
    if(cnt === 100) clearInterval(interval);
}, 1000);
```

Streaming using 30 points update

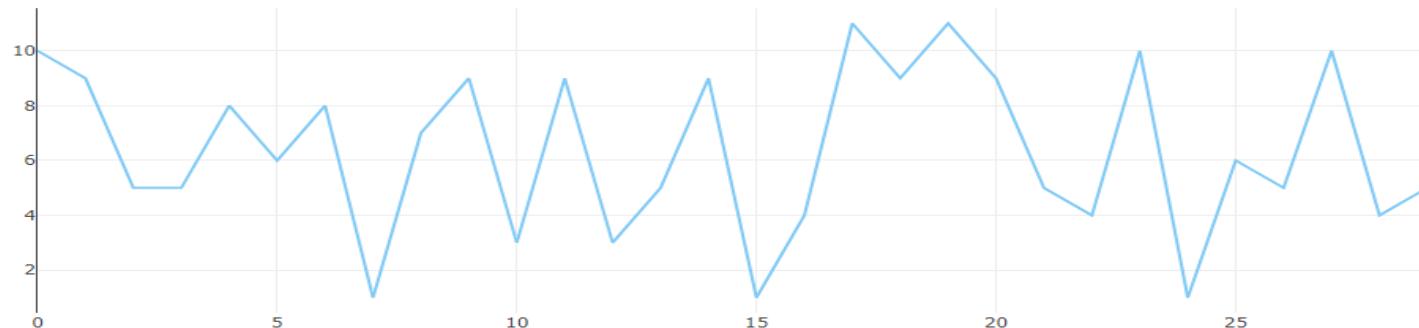




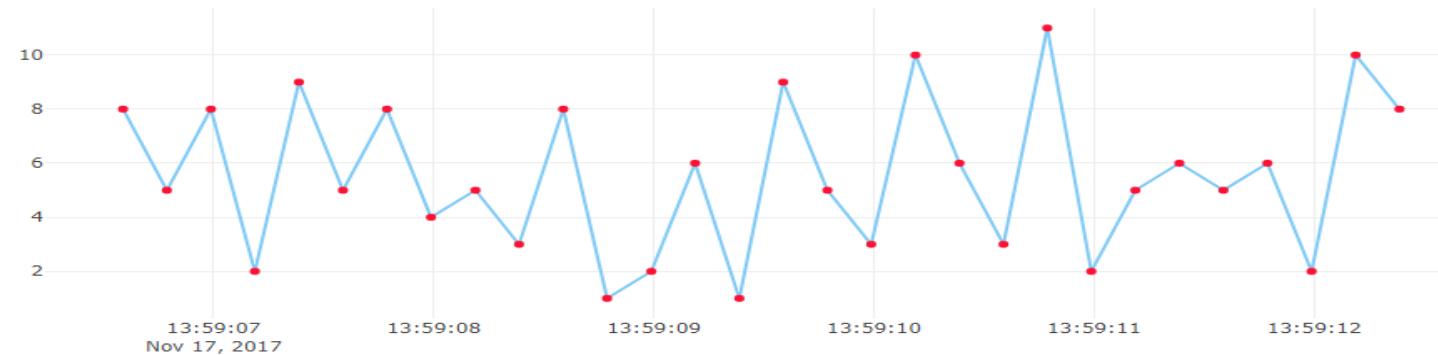
A6.4.8 plotly.js: Streaming data

[DIY] Streaming time series using 30 points update

Streaming using 30 points update



Streaming using 30 points update with timestamp



AAnn_DS_30timestamps.png 로 캡처 저장.



A6.4.9 plotly.js: Streaming data

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

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

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

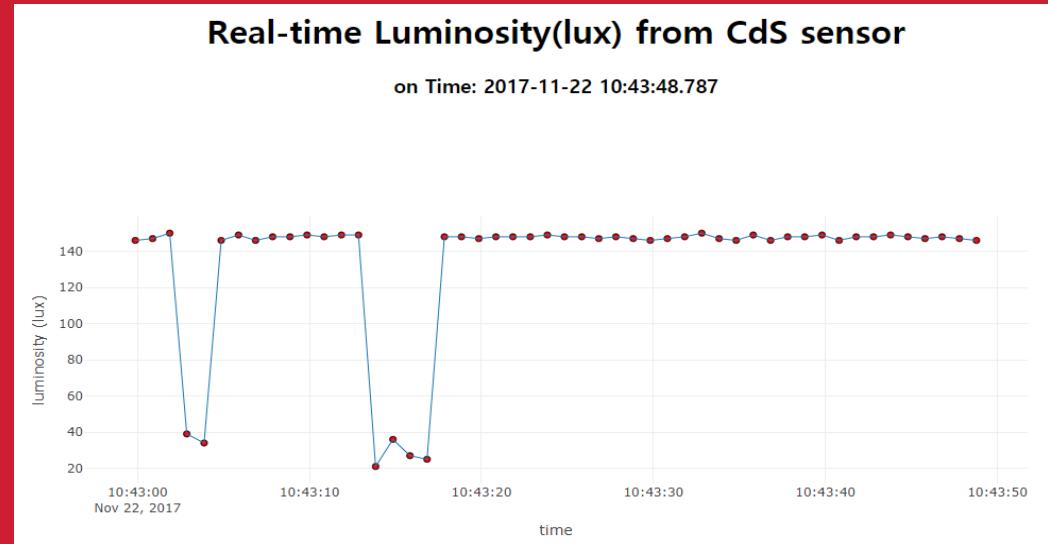
    var data = [
        {
            x: timeArray,
            y: newArray,
            mode: 'lines+markers',
            line: {color: '#80CAF6'},
            marker: {color: '#FC1234'}
        }
    ]

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



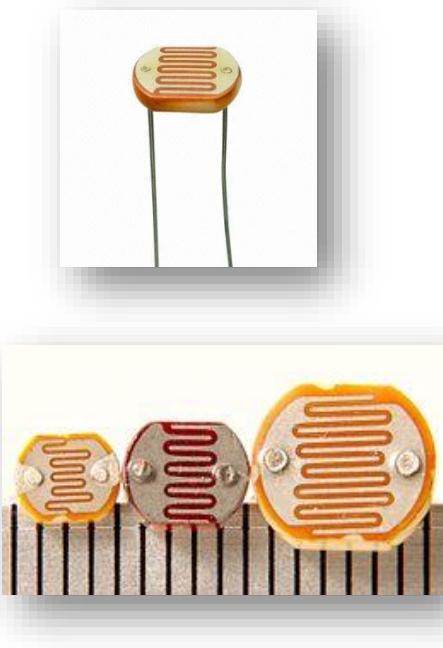
Arduino sensor data RT visualization using ploy.ly

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



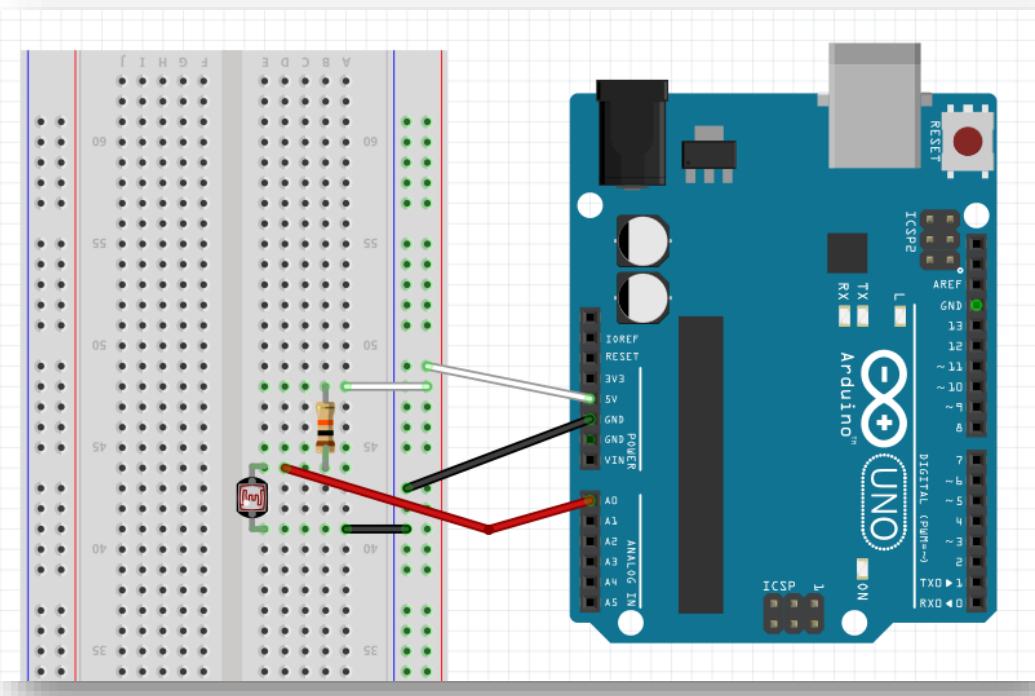


Luminosity sensor [Photocell LDR]



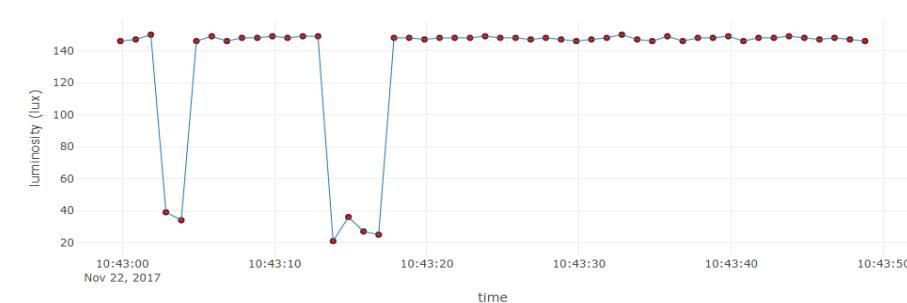
CdS

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



Real-time Luminosity(lux) from CdS sensor

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





A5.2.4 Luminosity sensor [Photocell LDR]

```
▼ └── ldr
    ├── node_modules
    └── /* ldr_node.js
        └── package.json
```

Save tmp36_node.js as ldr_node.js

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

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

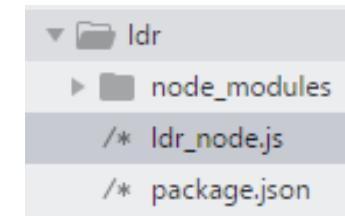
// helper function to get a nicely formatted date string
function getDateString() {
    var time = new Date().getTime();
    // 32400000 is (GMT+9 Korea, GimHae)
    // for your timezone just multiply +/-GMT by 3600000
    var datestr = new Date(time +32400000).
        toISOString().replace(/\T/, '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



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
```



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



A6.5.1 RT sensor-data streaming in Arduino

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

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <title>plotly.js Example: Real time signals from sensors</title>

  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/
  socket.io/1.3.6/socket.io.js"></script>
  <style>body{padding:0;margin:30;background:#fff}</style>
</head>
```



A6.5.2 RT sensor-data streaming in Arduino

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

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

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

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

<hr>

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

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



A6.5.3 RT sensor-data streaming in Arduino

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

```
var socket = io.connect('http://localhost:3000'); // port = 3000
socket.on('connect', function () {
    socket.on('message', function (msg) {
        // initial plot
        if(msg[0]!='' && initFlag){
            dtda[0]=msg[0];
            dtda[1]=parseFloat(msg[1]); // Lux
            init(); // start streaming
            initFlag=false;
        }
        console.log(msg[0]);
        console.log(parseFloat(msg[1]));
        // Convert value to integer
        dtda[0]=msg[0];
        dtda[1] = parseFloat(msg[1]); // integer

        // when new data is coming, keep on streaming data
        ctime.innerHTML = dtda[0];
        nextPt();
    });
});
```



A6.5.4 RT sensor-data streaming in Arduino

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

```
function init() { // initial screen ()  
  // starting point : first data (Lux)  
  for ( i = 0; i < numPts; i++) {  
    xArray.push(dtda[0]); // date  
    xTrack.push(dtda[1]); // CdS sensor (Lux)  
  }  
  
  Plotly.plot(streamPlot, data, layout);  
}  
  
function nextPt() {  
  
  xArray.shift();  
  xArray.push(dtda[0]);  
  
  xTrack.shift();  
  xTrack.push(dtda[1]); // CdS sensor: Lux  
  
  Plotly.redraw(streamPlot);  
}
```



A6.5.5 RT sensor-data streaming in Arduino

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

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

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

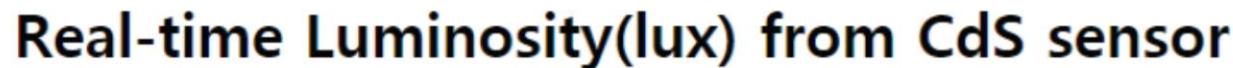
domain: [0,1] → x 또는 y 축을 100% 사용

range: [0,500] → y 축의 범위를 0~500 설정

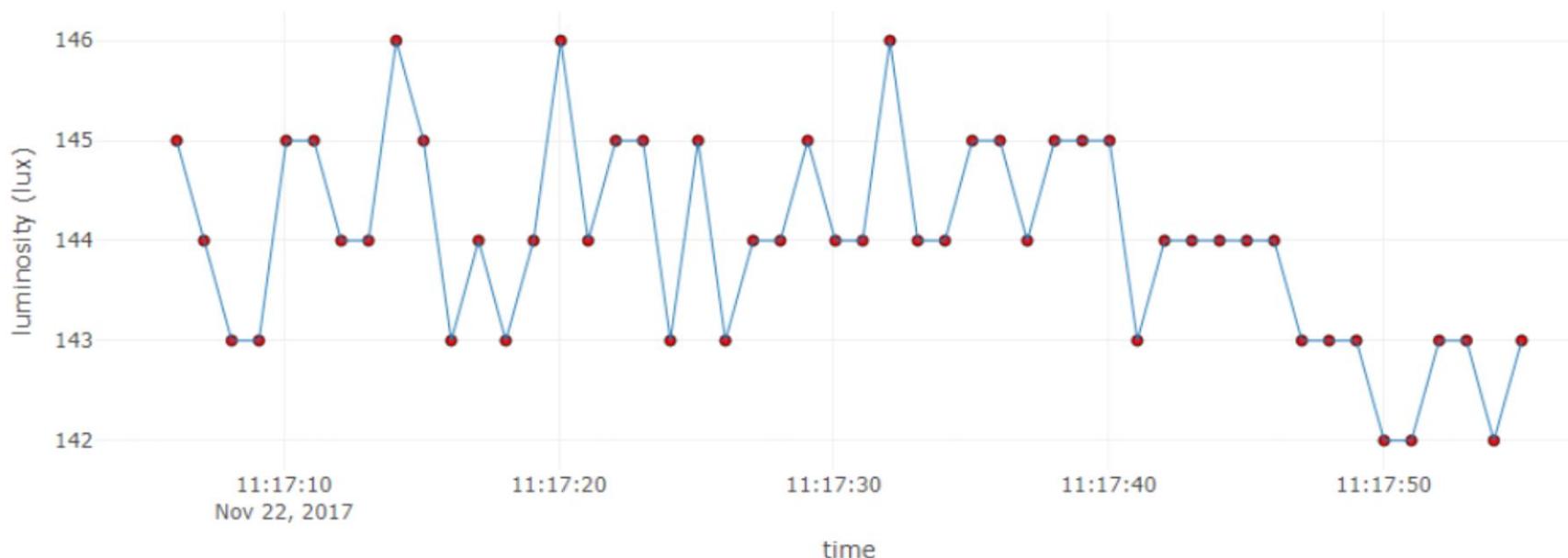


A6.5.6 RT sensor-data streaming in Arduino

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



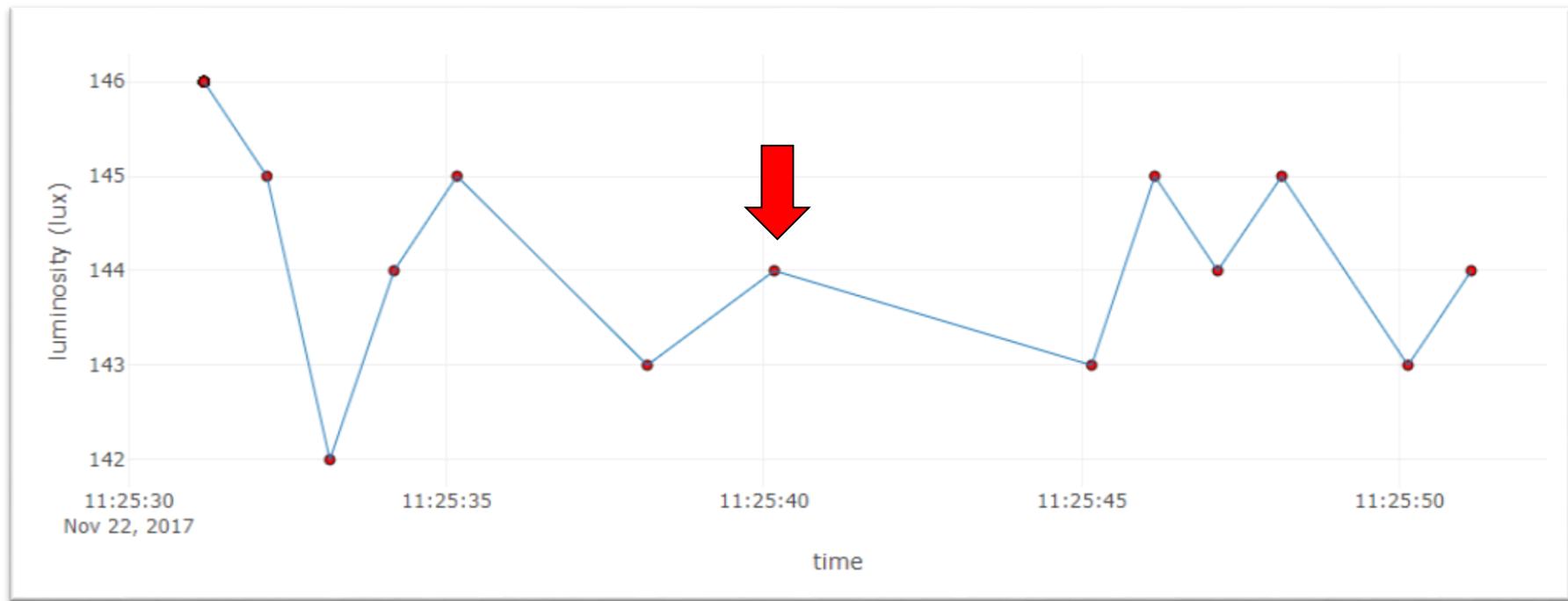
on Time: 2017-11-22 11:17:55.020





A6.5.7 RT sensor-data streaming in Arduino

[DIY 1.] Client html : client_ldr_change.html (detecting change)



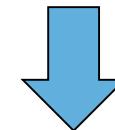
입력되는 lux 값이 변하는 경우에만 그래프를 그림.
실시간 모니터링에서 이상 감지 기능이 필요함.
밝기 값 변화의 문턱값을 설정해서 이상 감지 기능 구현



A6.5.8 RT sensor-data streaming in Arduino

[DIY 1. hint] Client html : client_Idr_change.html (detecting change)

```
// when new data is coming, keep on streaming data  
ctime.innerHTML = dtfa[0];  
nextPt();
```



```
// when new data is coming, keep on streaming data  
// Only when the value of Lux is different from the previous one,  
// the screen is redrawed.  
if (dtfa[1] != preX) { // any change?  
    preX = dtfa[1];  
  
    ctime.innerHTML = dtfa[0];  
    nextPt();  
}
```

측정되는 주변광의 밝기가 일정 시간 유지되다가 변하는
그래프를 캡처하여 [AAnn_Idr_change.png](#)로 저장



Canvas Gauge

[1] Canvas gauge javascript library : [gauge.js](#)

The screenshot shows a web browser window displaying the GitHub repository for 'canv-gauge'. The repository has 66 commits, 1 branch, 0 releases, and 6 contributors. The latest commit was on 23 Jul 2014. The repository contains files like fonts, README, build.bat, build.sh, compiler.jar, example-html-gauge.html, example-resize.html, example.html, gauge.js, and gauge.min.js.

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



A6.5.9 RT sensor-data streaming in Arduino

[DIY 2] Client html : client_ldr_gauge.html (add Gauge)

```
<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/
socket.io/1.3.6/socket.io.js"></script>

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

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

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



A6.5.10 RT sensor-data streaming in Arduino

[DIY 2] Client html : client_ldr_gauge.html (add Gauge)

```
// when new data is coming, keep on streaming data
// Only when the value of Lux is different from the previous one,
// the screen is redrawed.
if (dtda[1] != preX) { // any change?
    preX = dtda[1];

    gauge_lux.setValue(dtda[1]); // Lux gauge

    ctime.innerHTML = dtda[0];
    nextPt();
}
```



A6.5.11 RT sensor-data streaming in Arduino

[DIY 2] Client html : client_ldr_gauge.html (design of Gauge)

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



A6.5.12 RT sensor-data streaming in Arduino

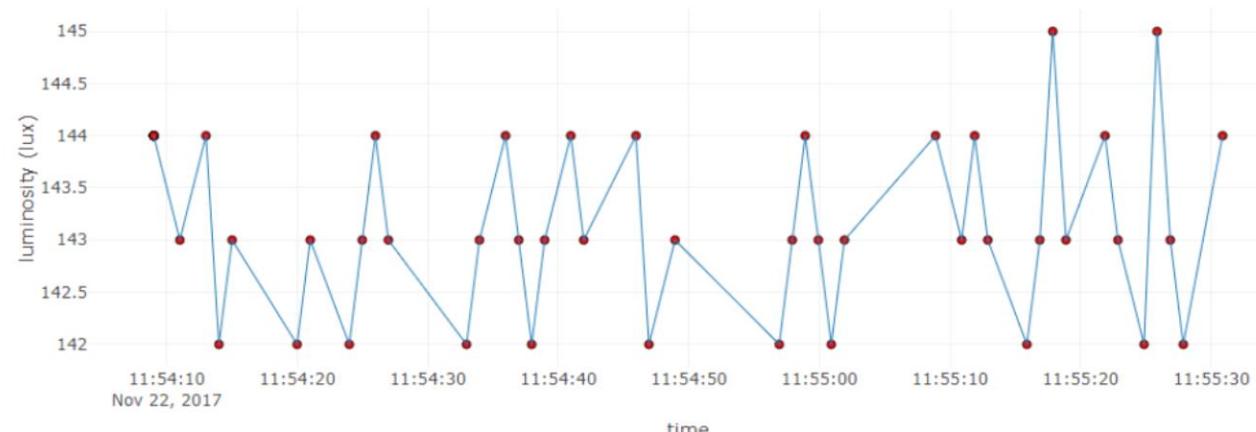
[DIY 2] Client html : client_ldr_gauge.html (change design of Gauge)

Real-time Luminosity(lux) from CdS sensor by AAnn



on Change time: 2017-11-22 11:55:30.859

변경된 디자인으로 된
그래프를 캡처하여
AAnn_ldr_gauge.
png로 저장



[DIY]

RT LDR lux monitoring
using plot.ly
data streaming



A6.5.3 RT sensor-data streaming in Arduino

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

```
var socket = io.connect('http://localhost:3000'); // port = 3000
socket.on('connect', function () {
    socket.on('message', function (msg) {
        // initial plot
        if(msg[0]!='' && initFlag){
            dtda[0]=msg[0];
            dtda[1]=parseFloat(msg[1]); // Lux
            init(); // start streaming
            initFlag=false;
        }
        console.log(msg[0]);
        console.log(parseFloat(msg[1]));
        // Convert value to integer
        dtda[0]=msg[0];
        dtda[1] = parseFloat(msg[1]); // integer

        // when new data is coming, keep on streaming data
        ctime.innerHTML = dtda[0];
        nextPt();
    });
});
```



A6.5.4 RT sensor-data streaming in Arduino

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

```
function init() { // initial screen ()  
  // starting point : first data (Lux)  
  for ( i = 0; i < numPts; i++) {  
    xArray.push(dtda[0]); // date  
    xTrack.push(dtda[1]); // CdS sensor (Lux)  
  }  
  
  Plotly.plot(streamPlot, data, layout);  
}  
  
function nextPt() {  
  
  xArray.shift();  
  xArray.push(dtda[0]);  
  
  xTrack.shift();  
  xTrack.push(dtda[1]); // CdS sensor: Lux  
  
  Plotly.redraw(streamPlot);  
}
```



A6.5.14 RT sensor-data streaming in Arduino

[6] Client html :
client_ldr_gauge_update.html
(using plotly streaming
without nextPt())

```
// function nextPt() {  
  
//     xArray.shift();  
//     xArray.push(dtda[0]);  
  
//     xTrack.shift();  
//     xTrack.push(dtda[1]); // Cds  
  
//     Plotly.redraw(streamPlot);  
// }
```

```
var socket = io.connect('http://localhost:3000'); // port = 3000  
socket.on('connect', function () {  
    socket.on('message', function (msg) {  
        // initial plot  
        if(msg[0]!='' && initFlag){  
            dtda[0]=msg[0];  
            dtda[1]=parseFloat(msg[1]); // Lux  
            init(); // start streaming  
            initFlag=false;  
        }  
        console.log(msg[0]);  
        console.log(parseFloat(msg[1]));  
        // Convert value to number  
        dtda[0]=msg[0];  
        dtda[1] = parseFloat(msg[1]); // number  
        // // when new data is coming, keep on streaming data  
        // if (dtda[1] != preX) { // any change?  
        //     preX = dtda[1];  
  
        //     gauge_Lux.setValue(dtda[1]); // Lux gauge  
        //     ctime.innerHTML = dtda[0];  
        //     nextPt();  
        // }  
        // when new data is coming, keep on streaming data  
        ctime.innerHTML = dtda[0];  
        gauge_lux.setValue(dtda[1]); // Lux gauge  
        //nextPt();  
        xArray = xArray.concat(dtda[0])  
        xArray.splice(0, 1) // remove the oldest data  
        xTrack = xTrack.concat(dtda[1])  
        xTrack.splice(0, 1) // remove the oldest data  
  
        var update = {  
            x: [xArray],  
            y: [xTrack]  
        }  
  
        Plotly.update(streamPlot, update);  
    });  
});
```



A6.5.15 RT sensor-data streaming in Arduino

[6] Client html : client_ldr_gauge_update.html (using plotly streaming without nextPt())

```
//nextPt();
xArray = xArray.concat(dtda[0])
xArray.splice(0, 1) // remove the oldest data
xTrack = xTrack.concat(dtda[1])
xTrack.splice(0, 1) // remove the oldest data

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

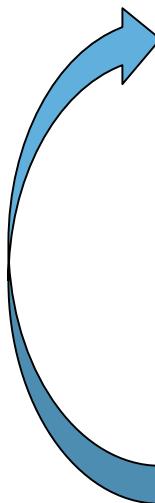
Plotly.update(streamPlot, update);
```

```
// function nextPt() {

//   xArray.shift();
//   xArray.push(dtda[0]);

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

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

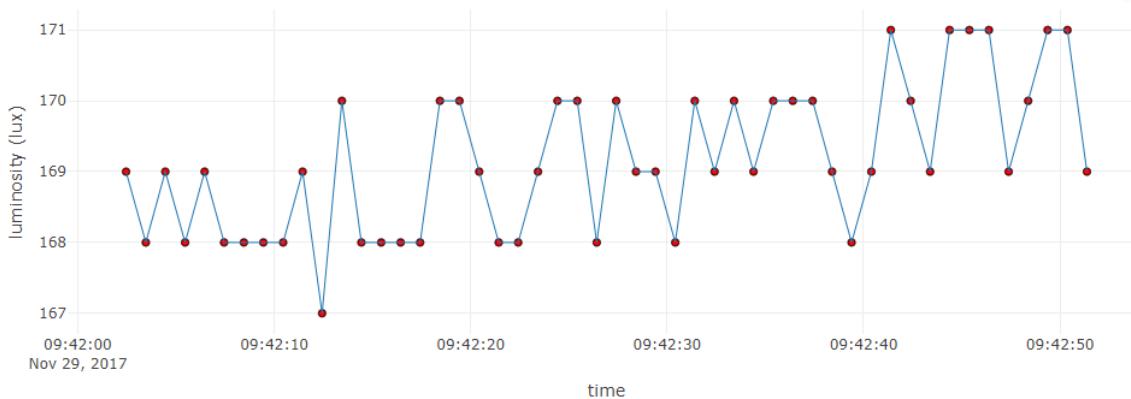


[6] Client html : client_Idr_gauge_update.html (using plotly streaming without nextPt())

Real-time Luminosity(lux) from CdS sensor by AAnn



on Change time: 2017-11-29 09:42:51.370



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

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

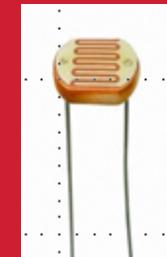
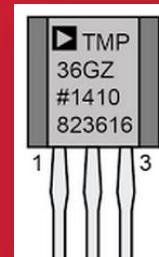
Plotly.update(streamPlot, update);
```



Arduino & Node.js

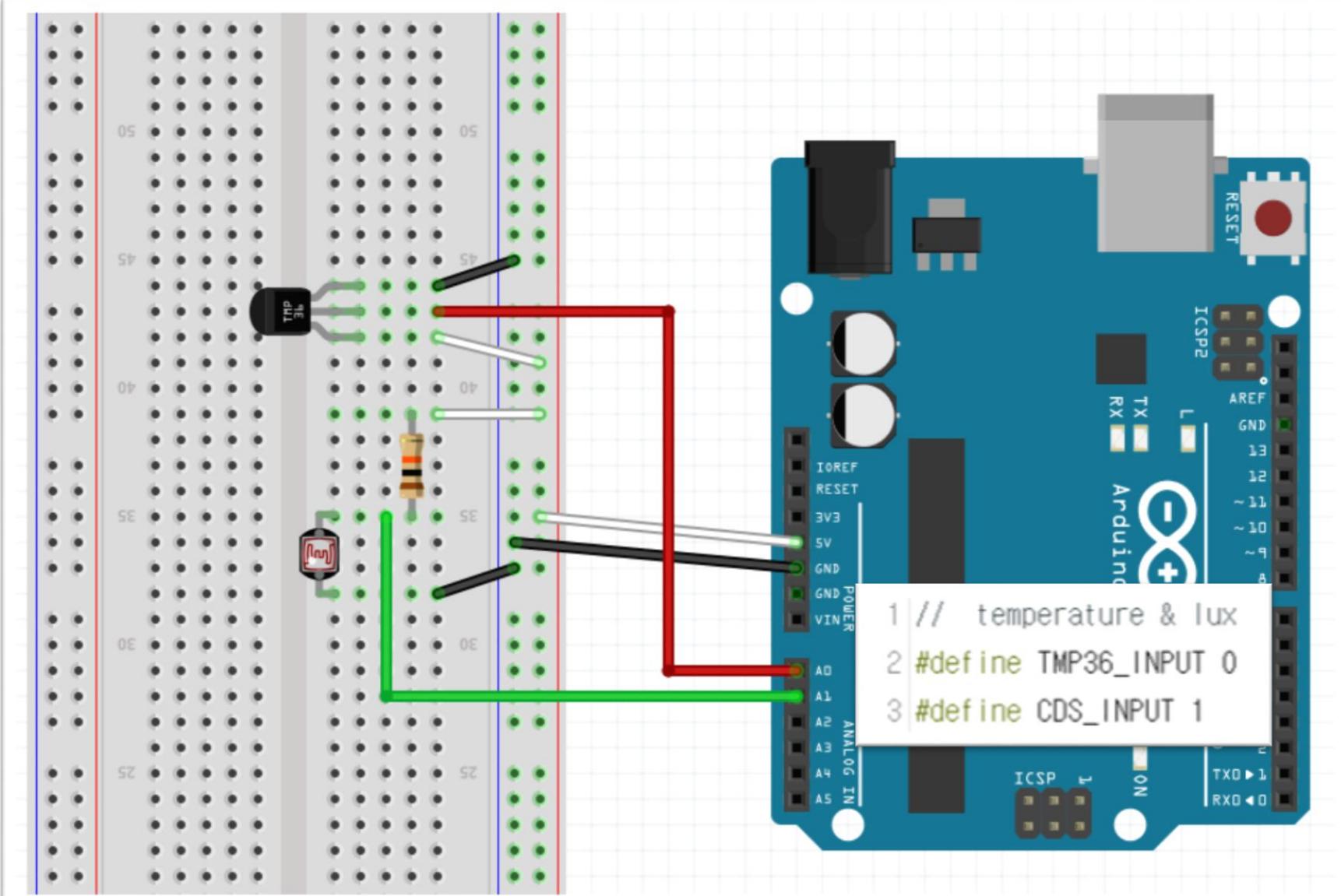
Multi-sensors

TMP36 + CdS





A6.6 TMP36 + CdS : circuit





A6.6.1 TMP36 + CdS + Node.js

[1] Arduino code: AAnn_TMP36_CdS.ino

```
AAnn_TMP36_CdS
1 // temperature & lux
2 #define TMP36_INPUT 0
3 #define CDS_INPUT 1
4
5 void setup() {
6   Serial.begin(9600);
7 }
8 void loop() {
9   // Temperature from TMP36
10  int temp_value = analogRead(TMP36_INPUT);
11  // converting that reading to voltage
12  float voltage = temp_value * 5.0 * 1000; // in mV
13  voltage /= 1023.0;
14  float tempC = (voltage - 500) / 10 ;
15
16  // Lux from CdS (LDR)
17  int cds_value = analogRead(CDS_INPUT);
18  int lux = int(luminosity(cds_value));
19  Serial.print("AAnn,");
20  Serial.print(tempC);
21  Serial.print(",");
22  Serial.println(lux);
23
24  delay(1000);
25 }
```

COM4

AAnn,11.09,156
AAnn,12.07,154
AAnn,12.07,154
AAnn,11.58,155
AAnn,11.09,67
AAnn,10.61,74
AAnn,10.61,82
AAnn,11.58,159
AAnn,12.56,163

```
27 //Voltage to LuxLux
28 double luminosity (int RawADC0){
29   double Vout=RawADC0*0.0048828125;
30   int lux=(2500/Vout-500)/10;
31   // lux = 500 / Rldr, Vout = Ildr*
32   return lux;
33 }
```



A6.6.2 TMP36 + CdS + Node.js

[2.1] NodeJS code: [tmp36_Ldr_node.js](#) ([← Ldr_node.js](#))

```
1 // tmp36_Ldr_node.js
2
3 var serialport = require('serialport');
4 var portName = 'COM4'; // 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 });
```



A6.6.3 TMP36 + CdS + Node.js

[2.2] NodeJS code: tmp36_ldr_node.js → Parsing data

```
19 var readData = ''; // this stores the buffer
20 var temp ='';
21 var lux ='';
22 var mdata =[]; // this array stores date and data from multiple sensors
23 var firstcommaidx = 0;
24
25 sp.on('data', function (data) { // call back when data is received
26     readData = data.toString(); // append data to buffer
27     firstcommaidx = readData.indexOf(',') // string.indexOf(searchvalue,start)
28     //console.log(data);
29
30     // parsing data into signals
31     if (readData.lastIndexOf(',') > firstcommaidx && firstcommaidx > 0) {
32         temp = readData.substring(firstcommaidx + 1, readData.lastIndexOf(','));
33         lux = readData.substring(readData.indexOf(',', firstcommaidx+1) + 1);
34         //console.log(firstcolonidx + "," + readData.indexOf(':', firstcolonidx+1))
35         readData = '';
36
37         dStr = getDateString();
38         mdata[0]=dStr; // Date
39         mdata[1]=temp; // temperature data
40         mdata[2]=lux; // humidity data
41         console.log(mdata);
42         io.sockets.emit('message', mdata); // send data to all clients
43
44     } else { // error
45         console.log(readData);
46     }
47
48 });

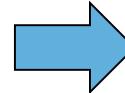
});
```



A6.6.4 TMP36 + CdS + Node.js

[3] Result: Parsed streaming data

```
COM4  
AArr,12.07,150  
AArr,12.56,152  
AArr,12.07,150  
AArr,11.09,150  
AArr,12.07,150  
AArr,12.07,150  
AArr,12.07,143  
AArr,11.09,150  
AArr,10.61,151  
AArr,10.61,42  
AArr,10.61,150  
AArr,11.09,152  
AArr,10.61,152  
AArr,10.61,152  
 자동 스크롤
```



```
D:\Portable\NodeJSPortable\Data>cd aa00  
D:\Portable\NodeJSPortable\Data\aa00>cd ldr  
D:\Portable\NodeJSPortable\Data\aa00\ldr>node tmp36_ldr_node  
[ '2017-11-29 10:05:45.264', '12.56', '153' ]  
[ '2017-11-29 10:05:46.255', '11.09', '153' ]  
[ '2017-11-29 10:05:47.247', '13.05', '153' ]  
[ '2017-11-29 10:05:48.257', '12.07', '152' ]  
[ '2017-11-29 10:05:49.248', '12.07', '152' ]  
[ '2017-11-29 10:05:50.256', '12.07', '153' ]  
[ '2017-11-29 10:05:51.250', '12.56', '153' ]  
[ '2017-11-29 10:05:52.241', '12.07', '151' ]  
[ '2017-11-29 10:05:53.249', '12.56', '152' ]  
[ '2017-11-29 10:05:54.241', '11.58', '153' ]  
[ '2017-11-29 10:05:55.250', '11.09', '153' ]  
[ '2017-11-29 10:05:56.243', '12.56', '153' ]  
[ '2017-11-29 10:05:57.234', '11.09', '153' ]  
[ '2017-11-29 10:05:58.243', '10.61', '153' ]  
[ '2017-11-29 10:05:59.234', '11.58', '152' ]  
[ '2017-11-29 10:06:00.244', '12.07', '153' ]  
[ '2017-11-29 10:06:01.235', '10.61', '153' ]  
[ '2017-11-29 10:06:02.227', '11.09', '153' ]  
[ '2017-11-29 10:06:03.236', '12.56', '153' ]  
[ '2017-11-29 10:06:04.227', '12.07', '152' ]  
[ '2017-11-29 10:06:05.237', '12.07', '153' ]  
[ '2017-11-29 10:06:06.228', '10.61', '153' ]
```

Save as AAarr_tmp36_ldr_data.png



A6.6.5 TMP36 + CdS + Node.js

[4.1] WEB client: client_tmp36_Idr.html

```
1  <!DOCTYPE html>
2  <head>
3      <meta charset="utf-8">
4      <title>plotly-basic.js Example: Real time signals from sensors</title>
5      <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
6      <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/
    socket.io/1.3.6/socket.io.js"></script>
7      <script src="gauge.min.js"></script>
8      <style>body{padding:0;margin:30;background:#fff}</style>
9  </head>
10
11 <body>  <!-- style="width:100%;height:100%"> -->
12 <!-- Plotly chart will be drawn inside this DIV -->
13 <h1 align="center"> Real-time Temperature(°C) and Luminosity(lux) from sensors </h1>
14     <!-- 1st gauge -->
15     <div align="center">
16         <canvas id="gauge1"> </canvas>
17         <!-- 2nd gauge -->
18         <canvas id="gauge2"> </canvas>
19     </div>
20     <!-- <div id="console"> </div> -->
21 <h3 align="center"> on Time: <span id="time"> </span> </h3>
22 <div id="myDiv"></div>
23 <hr>
```



A6.6.6 TMP36 + CdS + Node.js

[4.2] WEB client: client_tmp36_Idr.html

```
25 <script>
26 /* JAVASCRIPT CODE GOES HERE */
27 var streamPlot = document.getElementById('myDiv');
28 var ctime = document.getElementById('time');
29
30 var xArray = [], // time of data arrival
31     xTrack = [], // value of sensor 1 : temperature
32     yTrack = [], // value of sensor 2 : luminosity
33     numPts = 50, // number of data points in x-axis
34     dtdata = [], // 1 x 3 array : [date, data1, data2] from sensors
35     preX = -1,
36     preY = -1,
37     initFlag = true;
```



A6.6.7 TMP36 + CdS + Node.js

[4.3] WEB client: client_tmp36_ldr.html

```
39     var socket = io.connect('http://localhost:3000'); // port = 3000
40     socket.on('connect', function () {
41         socket.on('message', function (msg) {
42             // initial plot
43             if(msg[0]!='' && initFlag){
44                 dtfa[0]=msg[0];
45                 dtfa[1]=parseFloat(msg[1]); // temperature
46                 dtfa[2]=parseFloat(msg[2]); // Luminosity
47                 init();
48                 initFlag=false;
49             }
50             // console.log(msg[0]);
51             // console.log(parseFloat(msg[1]) + ', ' + parseFloat(msg[2]));
52             // Convert value to integer
53             dtfa[0]=msg[0];
54             dtfa[1] = parseFloat(msg[1]);
55             dtfa[2] = parseFloat(msg[2]);
56
57             // Only when any of temperature or Luminosity is different from the
58             // previous one, the screen is redrawn.
59             if (dtfa[1] != preX || dtfa[2] != preY) { // any change?
60                 preX = dtfa[1];
61                 preY = dtfa[2];
62
63                 gauge_temp.setValue(dtfa[1]);
64                 gauge_lux.setValue(dtfa[2]);
65                 ctime.innerHTML = dtfa[0];
66                 nextPt();
67             }
68         });
69     });
70 });
```



A6.6.8 TMP36 + CdS + Node.js

[4.4] WEB client: client_tmp36_Idr.html

```
70     function init() { // initial screen ()  
71         // starting point : first data (temp, humi)  
72         for ( i = 0; i < numPts; i++) {  
73             xArray.push(dtda[0]); // date  
74             xTrack.push(dtda[1]); // sensor 1 (temp)  
75             yTrack.push(dtda[2]); // sensor 2 (Lux)  
76         }  
77  
78         Plotly.plot(streamPlot, data, layout);  
79     }  
80  
81     function nextPt() {  
82  
83         xArray.shift();  
84         xArray.push(dtda[0]);  
85  
86         xTrack.shift();  
87         xTrack.push(dtda[1]); // sensor 1:temp  
88         yTrack.shift();  
89         yTrack.push(dtda[2]); // sensor 2:Lux  
90  
91         Plotly.redraw(streamPlot);  
92     }  
93 }
```



A6.6.9 TMP36 + CdS + Node.js

[4.5] WEB client: client_tmp36_Idr.html → data & layout

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

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



A6.6.10 TMP36 + CdS + Node.js

[4.6] WEB client: client_tmp36_Idr.html → gauges

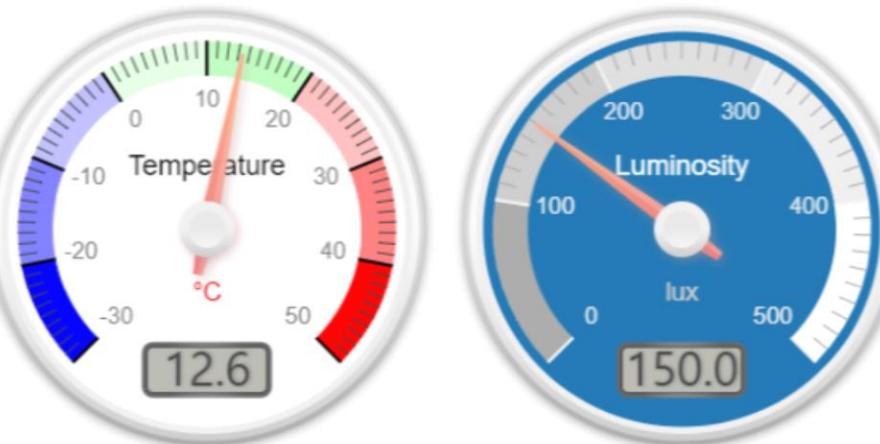
```
// gauge configuration
var gauge_temp = new Gauge({
  renderTo : 'gauge1',
  width    : 300,
  height   : 300,
  glow     : true,
  units    : '°C',
  valueFormat: { int : 2, dec : 1 },
  title    : "Temperature",
  minValue : -30,
  maxValue : 50,
  majorTicks: ['-30', '-20', '-10', '0', '10', '20', '30', '40', '50'],
  minorTicks: 10,
  strokeTicks: false,
  highlights: [
    { from : -30, to : -20, color : 'rgba(0, 0, 255, 1)' },
    { from : -20, to : -10, color : 'rgba(0, 0, 255, .5)' },
    { from : -10, to : 0, color : 'rgba(0, 0, 255, .25)' },
    { from : 0, to : 10, color : 'rgba(0, 255, 0, .1)' },
    { from : 10, to : 20, color : 'rgba(0, 255, 0, .25)' },
    { from : 20, to : 30, color : 'rgba(255, 0, 0, .25)' },
    { from : 30, to : 40, color : 'rgba(255, 0, 0, .5)' },
    { from : 40, to : 50, color : 'rgba(255, 0, 0, 1)' }
  ],
  colors    : {
    plate     : '#fff',
    majorTicks: '#000',
    minorTicks: '#444',
    title     : '#000',
    units     : '#f00',
    numbers   : '#777',
    needle    : { start : 'rgba(240, 128, 128, 1)', end : 'rgba(255, 160, 122, .9)' }
  }
});

gauge_temp.draw();
```

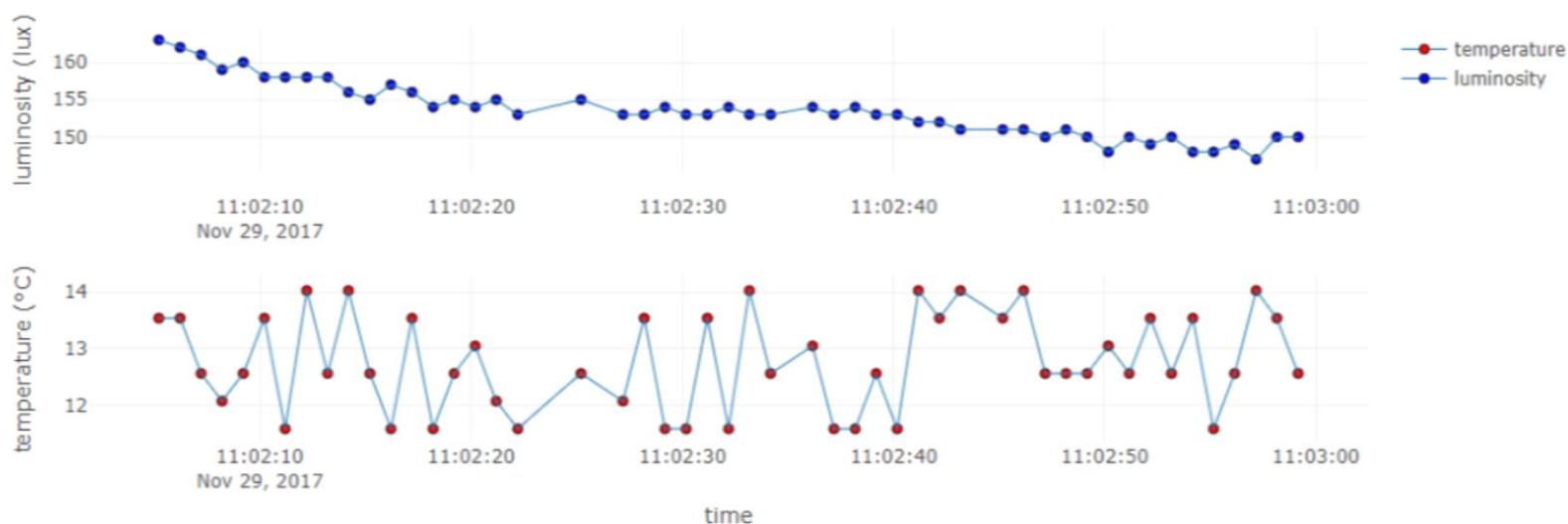
```
var gauge_lux = new Gauge({
  renderTo : 'gauge2',
  width    : 300,
  height   : 300,
  glow     : true,
  units    : 'lux',
  valueFormat: { int : 2, dec : 1 },
  title    : "Luminosity",
  minValue : 0,
  maxValue : 500,
  majorTicks: ['0', '100', '200', '300', '400', '500'],
  minorTicks: 10,
  strokeTicks: false,
  highlights: [
    { from : 0, to : 100, color : '#aaa' },
    { from : 100, to : 200, color : '#ccc' },
    { from : 200, to : 300, color : '#ddd' },
    { from : 300, to : 400, color : '#eee' },
    { from : 400, to : 500, color : '#fff' }
  ],
  colors    : {
    plate     : '#1f77b4',
    majorTicks: '#f5f5f5',
    minorTicks: '#aaa',
    title     : '#fff',
    units     : '#ccc',
    numbers   : '#eee',
    needle    : { start : 'rgba(240, 128, 128, 1)', end : 'rgba(255, 160, 122, .9)' }
  }
});

gauge_lux.draw();
```

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



on Time: 2017-11-29 11:02:59.163



[DIY]

Real-time

TMP36 & CdS monitoring

using plot.ly

→ Data streaming



[DIY] RT sensor-data streaming in Arduino

[5.1] WEB client: [client_tmp36_ldr_update.html](#) (using plotly streaming without nextPt())

Comment function `nextPt()`

```
// function nextPt() {  
  
//     xArray.shift();  
//     xArray.push(dtda[0]);  
  
//     xTrack.shift();  
//     xTrack.push(dtda[1]); // sensor 1:temp  
//     yTrack.shift();  
//     yTrack.push(dtda[2]); // sensor 2:Lux  
  
//     Plotly.redraw(streamPlot);  
// }
```



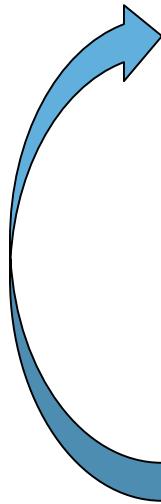
[DIY] RT sensor-data streaming in Arduino

[5.2] WEB client: [client_tmp36_ldr_update.html](#) (using plotly streaming without nextPt())

```
//nextPt();
xArray = xArray.concat(dtda[0])
xArray.splice(0, 1) // remove the oldest data
xTrack = xTrack.concat(dtda[1])
xTrack.splice(0, 1) // remove the oldest data

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

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



Complete this part of the code.

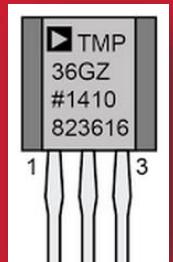
```
// function nextPt() {
//
//   xArray.shift();
//   xArray.push(dtda[0]);
//
//   xTrack.shift();
//   xTrack.push(dtda[1]); // sensor 1:temp
//   //
//   yTrack.shift();
//   yTrack.push(dtda[2]); // sensor 2:Lux
//
//   Plotly.redraw(streamPlot);
// }
```

**Save the complete
code as
[AAnn_update.html](#)**



[Practice]

◆ [wk14]



- RT Data Visualization with node.js
- TMP38 + CdS sensors
- Complete your real-time charts
- AAnn_Rpt11.zip

wk14 : Practice-11 : AAnn_Rpt11.zip

◆ [Target of this week]

- Complete your plots of real-time streaming of TMP36 & CdS
- Design your own gauge and chart.
- Save your outcomes and compress 3 figures

제출파일명 : **AAnn_Rpt11.zip**

- 압축할 파일들

- ① **AAnn_tmp36_Idr_data.png**
- ② **AAnn_update.html**

Email : **chaos21c@gmail.com**



[Tip] Using WEB browser in SB text3

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

1. **Tool -> New Plugin**을 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 - User**로 이동한 후 단축키를 할당한다.

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



[참고 : 저항 값 읽기]

Carbonfilm resistor

4 Color stripes

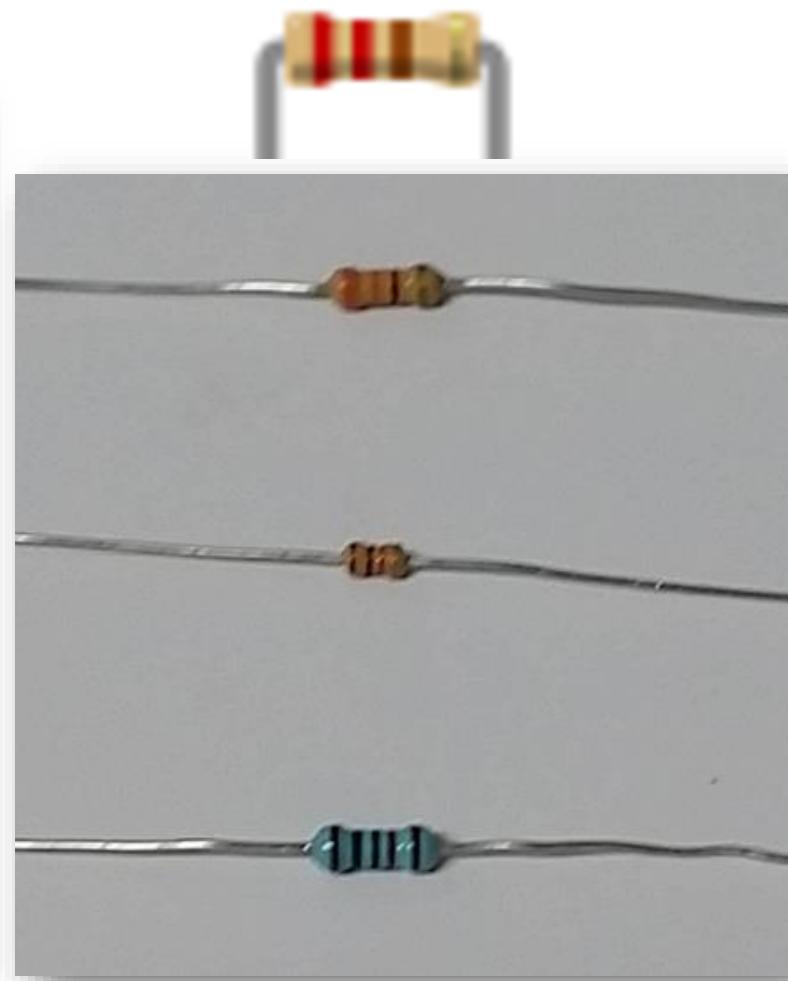
$47 \times 1000 = 47\text{KOhm } 5\%$

sm2k (c) 2006

5 Color stripes

$576 \times 1 = 576 \text{ Ohm } 1\%$

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%

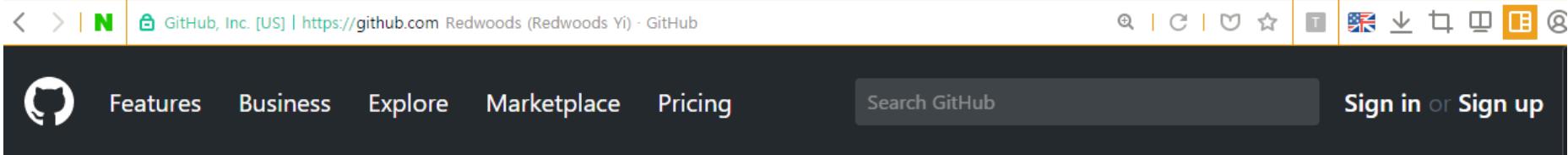


Lecture materials

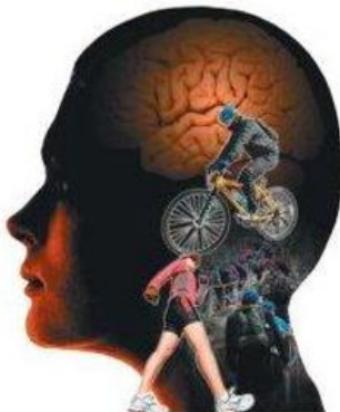


● References & good sites

- ✓ <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



A screenshot of a GitHub user profile page. At the top, there's a dark header with a navigation bar containing icons for back, forward, and search, along with links for GitHub features, business, explore, marketplace, and pricing. To the right is a search bar labeled "Search GitHub" and buttons for "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

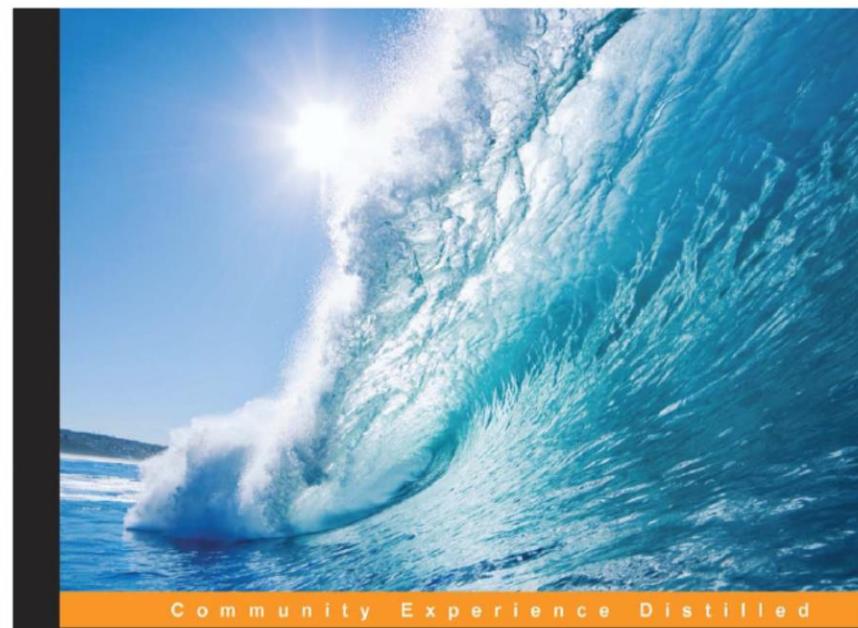
[hw-coding](#)

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

 Arduino



References

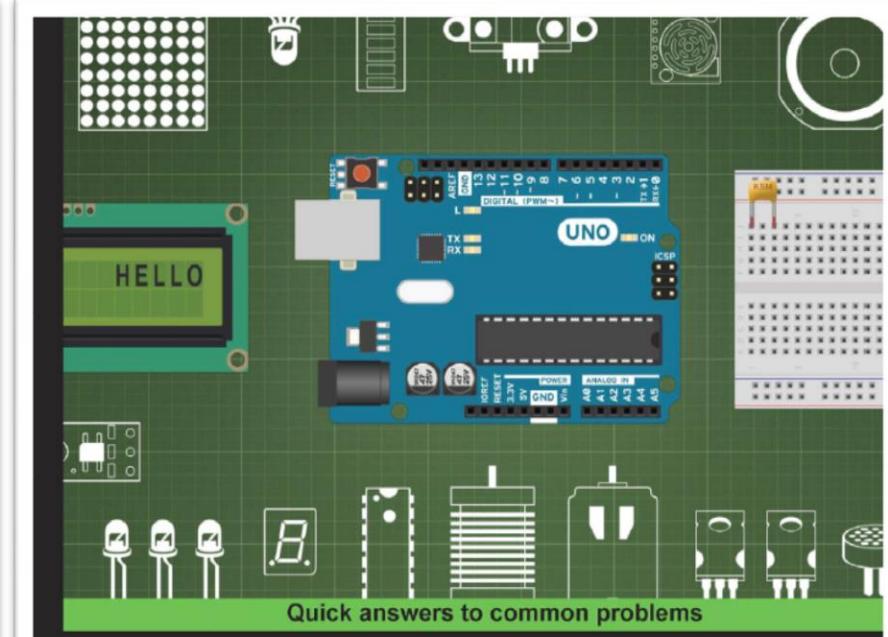


Arduino Essentials

Enter the world of Arduino and its peripherals and start creating interesting projects

Francis Perea

[PACKT]
PUBLISHING



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

www.allaboutcircuits.com