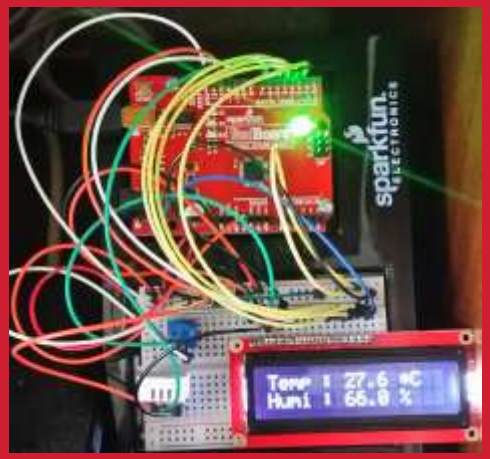



# Arduino-IOT

## [wk07] Review

# Arduino + node.js

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

Drone-IoT-Comsi, INJE University

2nd semester, 2020

Email : chaos21c@gmail.com

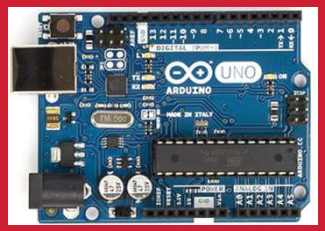
# My ID

## 1분반-목요일 (2학년)

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

## 2분반-수요일 (3학년)

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



# [Review]

- [wk06]
  - Arduino sensors + Node.js
  - Complete your project
  - Upload folder: aax-nn-rpt06
  - Use repo “aax-nn” in github

# wk06 : Practice : AAnn_Rpt06

## [Target of this week]

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

제출폴더명 : **aax-nn-rpt06**

- 압축할 파일들

① **AAnn_cds_IOT_data.png**

② **AAnn_cds_tmp36_serial.png**

③ **AAnn_cds_tmp36_lcd.png**

④ **AAnn_cds_tmp36_IOT.png**

⑤ **AAnn_multi_signals_node.png**

⑥ **All *.ino**

⑦ **All *.js**

⑧ **NO node_modules folder**

**Email : chaos21c@gmail.com**

# IOT: HSC

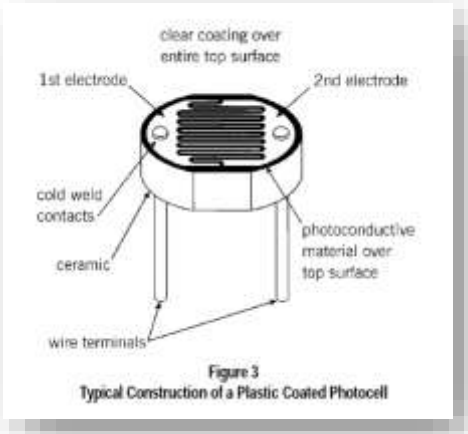
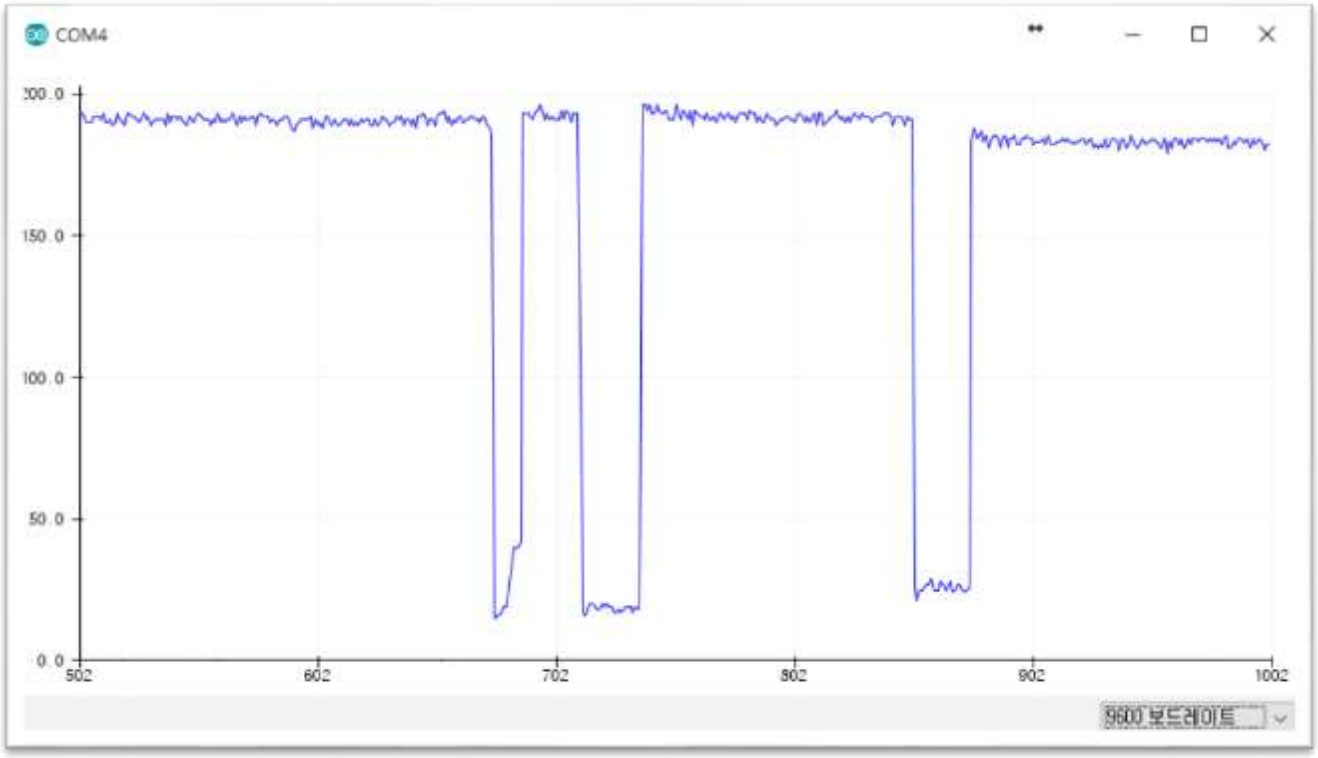
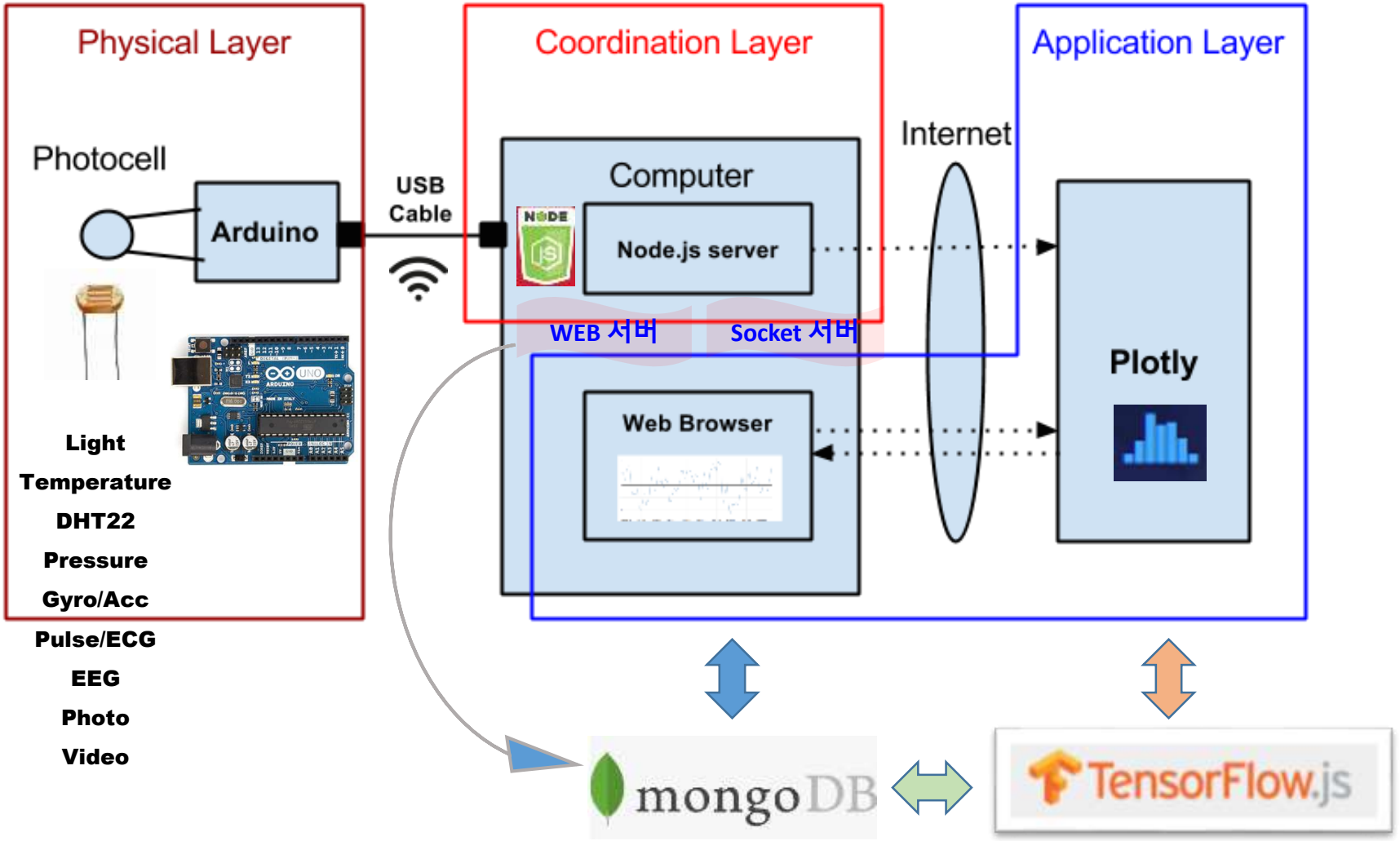


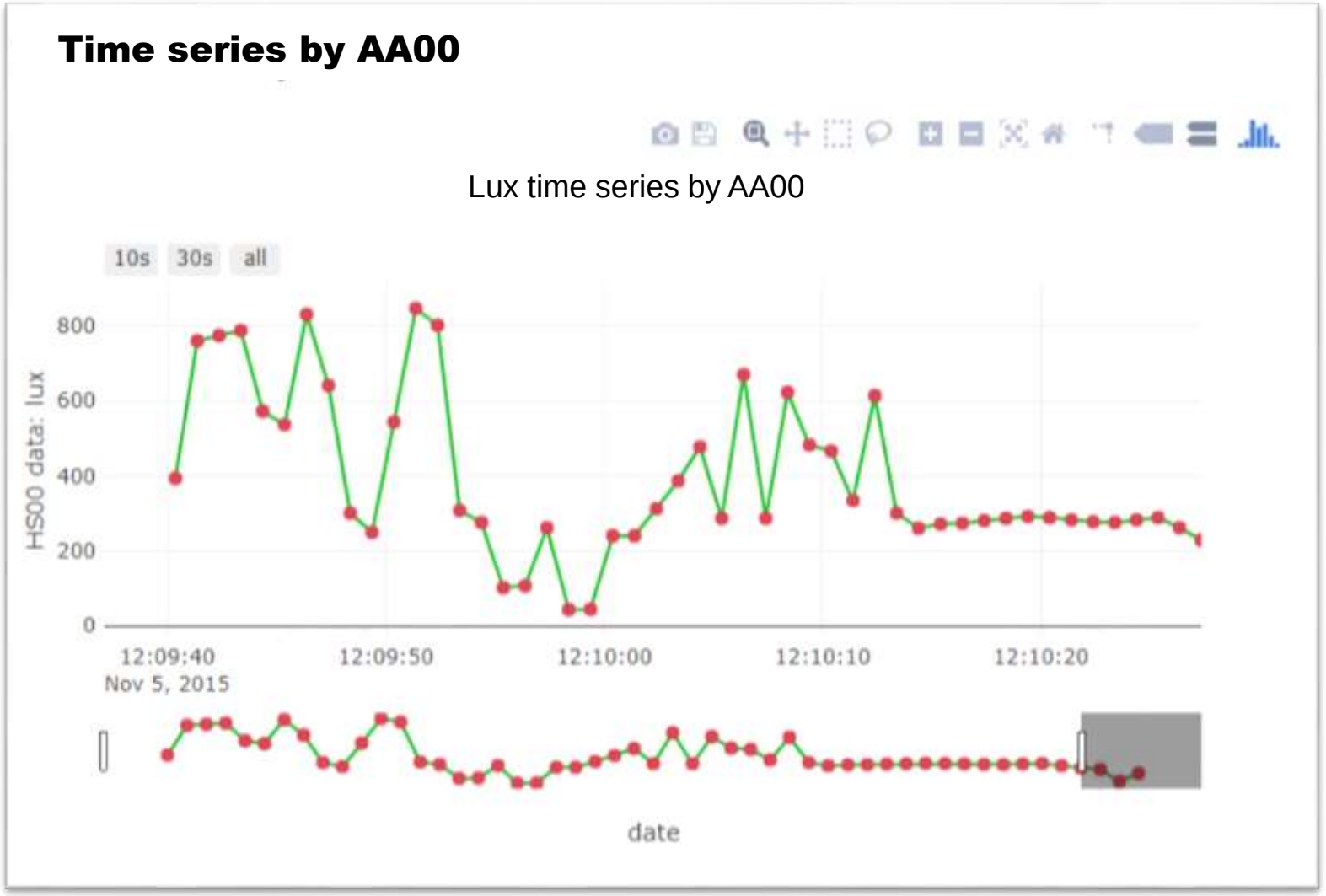
Figure 3
Typical Construction of a Plastic Coated Photocell



# Layout [H S C]

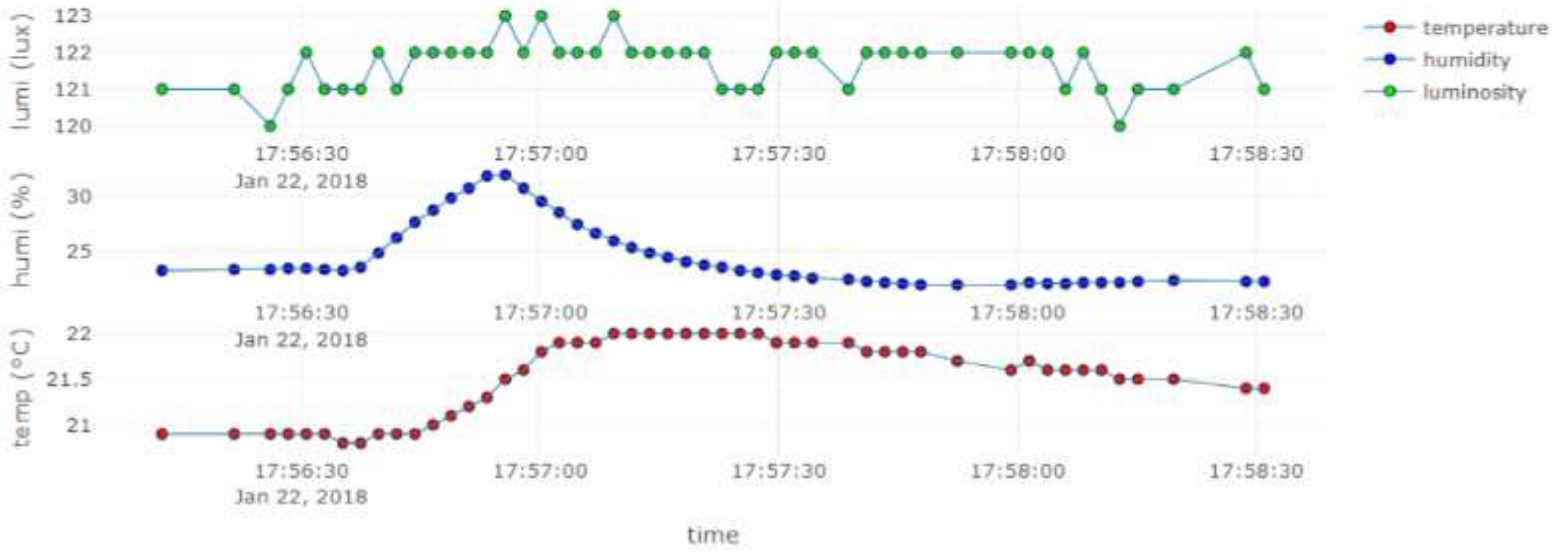

Physical Layer
Photocell
Arduino
USB Cable
Light
Temperature
DHT22
Pressure
Gyro/Acc
Pulse/ECG
EEG
Photo
Video
Coordination Layer
Computer
Node.js server
WEB 서버
Socket 서버
Web Browser
Internet
Application Layer
Plotly
mongoDB
TensorFlow.js

# Arduino data + plotly

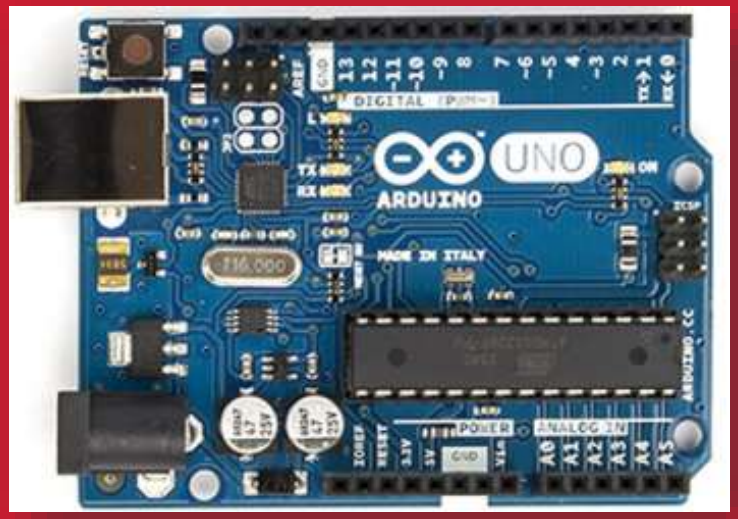


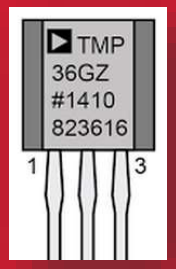
# Real-time Weather Station from sensors

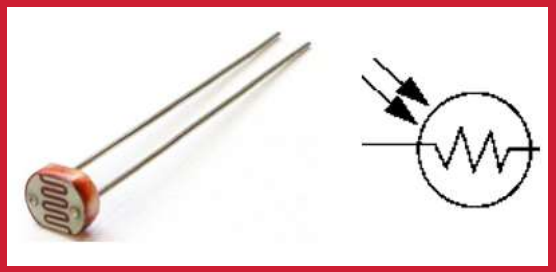
**on Time: 2018-01-22 17:58:31.012**

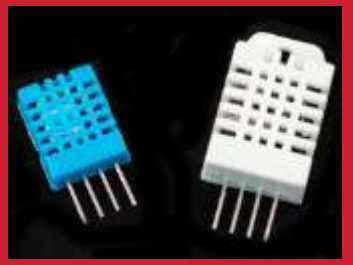


# Arduino Sensors + Node.js

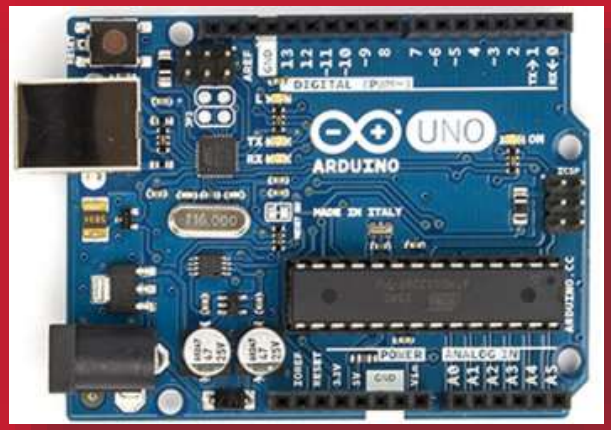


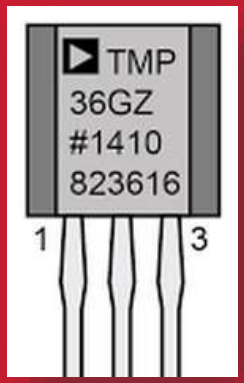


# Single sensor: tmp36

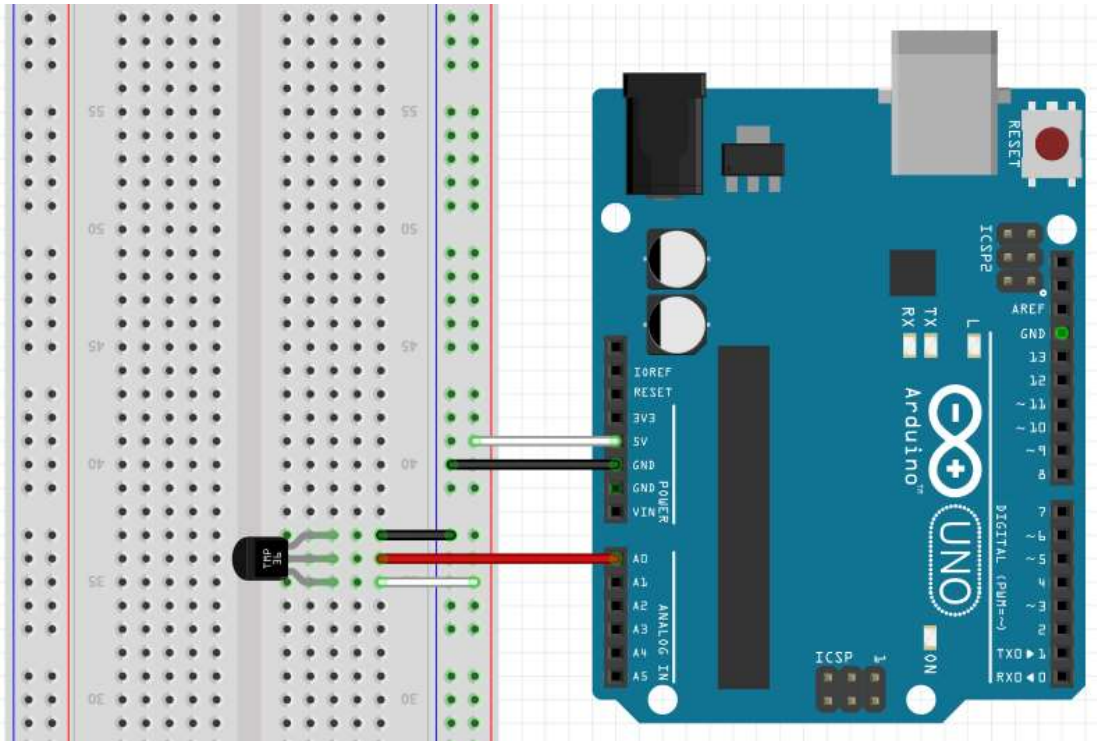


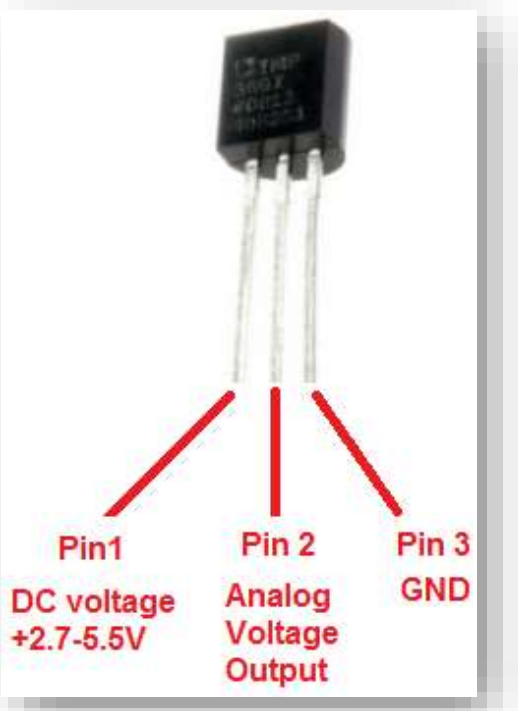

# TMP36

# Node project

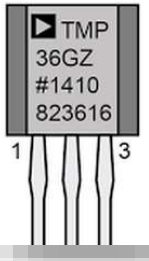


# A3.1.1 Temperature sensor [ TMP36]




## 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

# A4.1.1 tmp36 node project

**Start tmp36-node project**

1. **Go to my working folder**
2. **md iot & cd iot**
3. **md tmp36**
4. **cd tmp36**
5. **Open terminal**
6. **npm init**

# A4.1.2 tmp36 node project: npm init

```
D:\Portable\vscode-portable\data\aa2-00\aa2-99-rpt06\iot\tmp36>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (tmp36)
version: (1.0.0)
description: tmp36-node project
entry point: (index.js) tmp36_node.js
test command:
git repository:
keywords: tmp36 node arduino
author: aa00
license: (ISC) MIT
```

# A4.1.3 tmp36 node project: package.json

```
1  {
2    "name": "tmp36",
3    "version": "1.0.0",
4    "description": "tmp36-node project",
5    "main": "tmp36_node.js",
       ▷ Debug
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [
10     "tmp36",
11     "node",
12     "arduino"
13   ],
14   "author": "aa00",
15   "license": "MIT"
16 }
```

# A4.1.4 tmp36 node project: install modules

**npm install --save serialport**

**npm install --save socket.io**

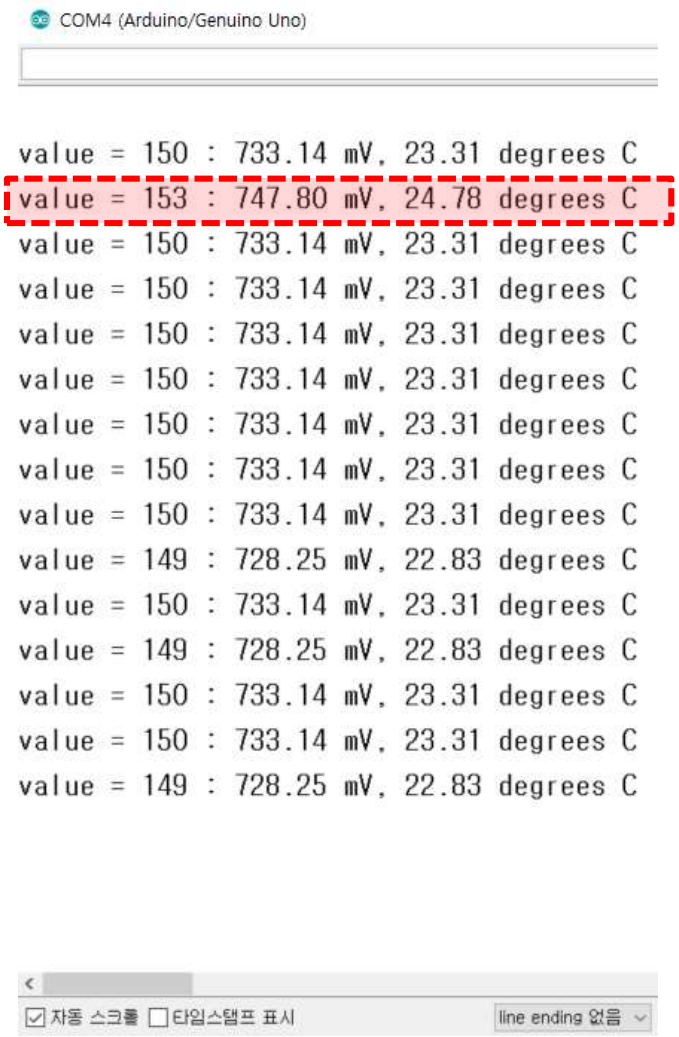
```
"author": "aa00",
"license": "MIT",
"dependencies": {
  "serialport": "^9.0.1",
  "socket.io": "^2.3.0"
}
```

# A4.1.5 tmp36 node project (Arduino code)

## AAnn_TMP36_NodeJS.ino

```
12 void loop() {
13   //getting the voltage reading from the temperature sensor
14   int value = analogRead(TEMP_INPUT);
15   Serial.print("value = ");
16   Serial.print(value);
17   Serial.print(" : ");
18
19   // converting that reading to voltage
20   float voltage = value * 5.0 * 1000;  // in mV
21   voltage /= 1023.0;
22
23   // print out the voltage
24   Serial.print(voltage);
25   Serial.print(" mV, ");
26
27   // now print out the temperature
28   float temperatureC = (voltage - 500) / 10 ;
29   Serial.print(temperatureC);
30   Serial.println(" degrees C");
31
32   delay(1000);
33 }
```

## Serial monitor



# A4.1.6 tmp36 node project (node code)

## tmp36_node_start.js

```
3   var serialport = require("serialport");
4   var portName = "COM3"; // check your COM port!!
5   var port = process.env.PORT || 3000;
6
7   var io = require("socket.io").listen(port);
8
9   const Readline = require("@serialport/parser-readline");
10  // serial port object
11  var sp = new serialport(portName, {
12    baudRate: 9600, // 9600  38400
13    dataBits: 8,
14    parity: "none",
15    stopBits: 1,
16    flowControl: false,
17    parser: new Readline("\r\n"),
18  });
```

## Node cmd

```
20  const parser = sp.pipe(new Readline({ delimiter: "\r\n" }));
21
22  // Read the port data
23  sp.on("open", () => {
24    console.log("serial port open");
25  });
26
27  var tdata = []; // Array
28
29  parser.on("data", (data) => {
30    // call back when data is received
31    // raw data only
32    //console.log(data);
33
34    tdata = data; // data
35    console.log("AA00," + tdata);
36    io.sockets.emit("message", tdata); // send data to all clie
37  });
```

# A4.1.7 tmp36 node project (node cmd message )

**[Terminal] node tmp36_node_start.js**

```
D:\Portable\vscode-portable\data\aa2-00\aa2-99-rpt00\iot\tmp36>node tmp36_node_start
serial port open
AA00,2.01 mV, 28.20 degrees C
AA00,value = 160 : 782.01 mV, 28.20 degrees C
AA00,value = 160 : 782.01 mV, 28.20 degrees C
AA00,value = 160 : 782.01 mV, 28.20 degrees C
AA00,value = 160 : 782.01 mV, 28.20 degrees C
AA00,value = 159 : 777.13 mV, 27.71 degrees C
AA00,value = 162 : 791.79 mV, 29.18 degrees C
AA00,value = 159 : 777.13 mV, 27.71 degrees C
AA00,value = 159 : 777.13 mV, 27.71 degrees C
AA00,value = 160 : 782.01 mV, 28.20 degrees C
AA00,value = 159 : 777.13 mV, 27.71 degrees C
AA00,value = 159 : 777.13 mV, 27.71 degrees C
AA00,value = 159 : 777.13 mV, 27.71 degrees C
```

# A4.1.8 tmp36 node project (all messages)

## tmp36_node.js

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

parser.on("data", (data) => {
  // call back when data is received
  // raw data only
  //console.log(data);
  dStr = getDateString();
  tdata[0] = dStr; // date
  tdata[1] = data; // data
  console.log("AA00," + tdata);
  io.sockets.emit("message", tdata); //
});

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;
}
```

## [Terminal] node tmp36_node

```
D:\Portable\vscode-portable\data\aa2-00\aa2-99-rpt06\iot\tmp36>node tmp36_node
serial port open
AA00,2020-10-13 16:11:20.497,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:21.500,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:22.499,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:23.503,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:24.507,value = 160 : 782.01 mV, 28.20 degrees C
AA00,2020-10-13 16:11:25.505,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:26.509,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:27.509,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:28.512,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:29.512,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:30.515,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:31.515,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:32.518,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:33.522,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:34.521,value = 159 : 777.13 mV, 27.71 degrees C
AA00,2020-10-13 16:11:35.524,value = 159 : 777.13 mV, 27.71 degrees C
```

**AAnn_tmp36_message.png**
로 저장

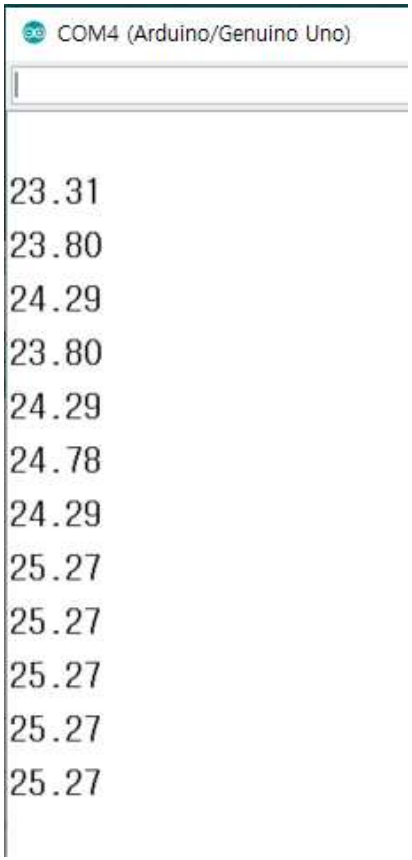
# A4.1.9 tmp36 node project (only data)

## AAnn_TMP36_NodeJS.ino 수정

AA00_TMP36_NodeJS

```
11
12 void loop() {
13   //getting the voltage reading from the temperature sensor
14   int value = analogRead(TEMP_INPUT);
15 //  Serial.print("AA00, value = ");
16 //  Serial.print(value);
17 //  Serial.print(" : ");
18
19   // converting that reading to voltage
20   float voltage = value * 5.0 * 1000;  // in mV
21   voltage /= 1023.0;
22
23   // print out the voltage
24 //   Serial.print(voltage);
25 //   Serial.print(" mV, ");
26
27   // now print out the temperature
28   float temperatureC = (voltage - 500) / 10 ;
29 //   Serial.print(" Temperature, ");
30   Serial.println(temperatureC);
31 //   Serial.println(" degrees C");
32
33   delay(1000);
34 }
```

## 실행 결과



# A4.1.10 tmp36 node project (date & data → IOT)

**[Terminal] node tmp36_node**

```
D:\Portable\vscode-portable\data\aa2-00\aa2-99-rpt06\iot\tmp36>node tmp36_node
serial port open
AA00,2020-10-13 16:20:39.684,27.71
AA00,2020-10-13 16:20:40.685,27.22
AA00,2020-10-13 16:20:41.684,27.71
AA00,2020-10-13 16:20:42.684,27.71
AA00,2020-10-13 16:20:43.688,27.71
AA00,2020-10-13 16:20:44.687,28.20
AA00,2020-10-13 16:20:45.691,27.71
AA00,2020-10-13 16:20:46.690,27.71
AA00,2020-10-13 16:20:47.689,27.22
AA00,2020-10-13 16:20:48.692,27.71
AA00,2020-10-13 16:20:49.692,27.71
AA00,2020-10-13 16:20:50.692,27.71
```

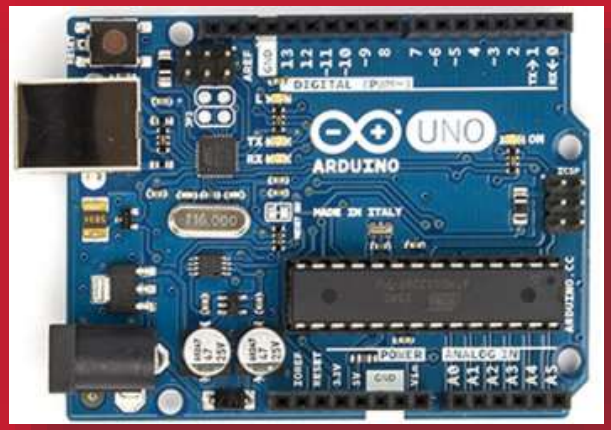
시간 , 온도

공백없이 “,”로
시간과 온도 구분

**IOT data format**
시간, **data**
시간, 온도

**AAnn_tmp36_IOT_data.png**
로 저장

# Single sensor: CdS

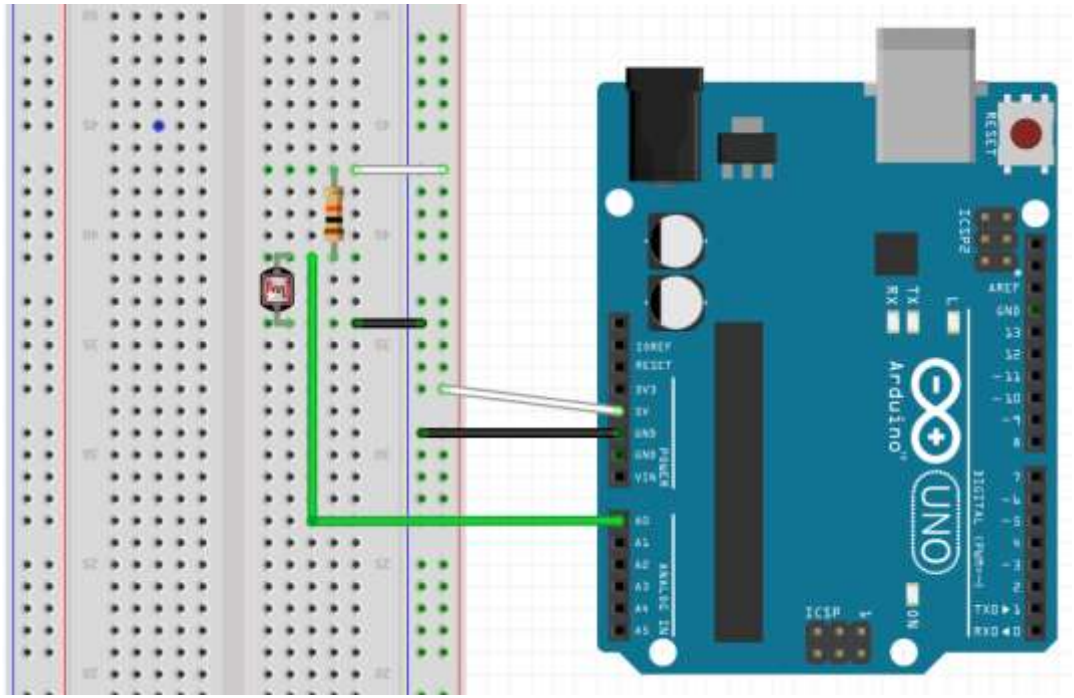


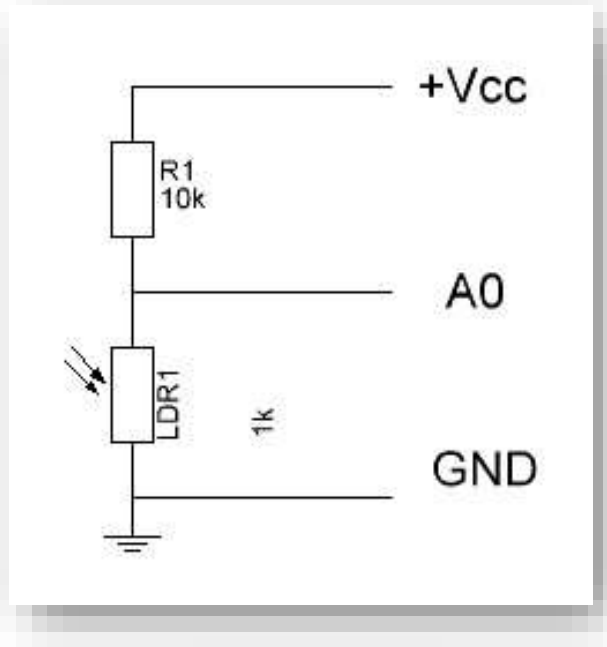

# CdS (LDR)

# Node project

# A3.2.2 Luminosity sensor [ Photocell LDR]

## CdS 센서 회로






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

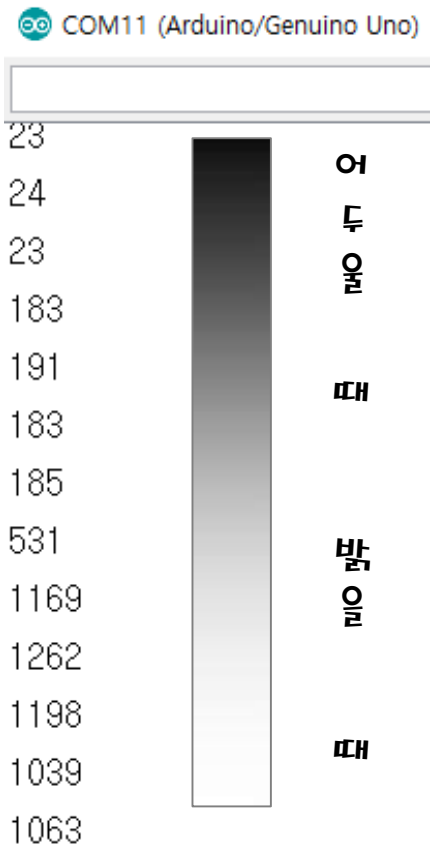
광센서에서의 전압 강하 값을 **A0**로 측정

# A3.2.6 Luminosity sensor [ Photocell LDR]

## CdS 센서 회로 - 측정 2.

**AAnn_cds_start.ino**

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

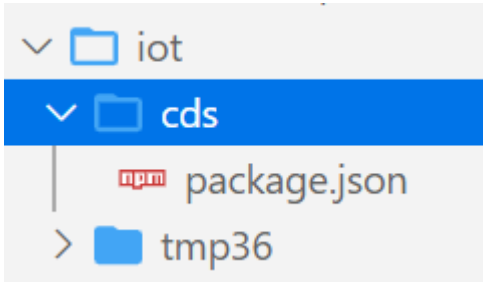



밝을수록 측정 값이 커지고
어두을수록 값이 작아진다 !!!

# A4.2.1 Luminosity sensor [ npm init ]

**Start cds-node project**

1. **Go to my working folder**
2. **Go to iot folder**
3. **md cds**
4. **cd cds**
5. **Open terminal in cds**
6. **npm init**




**"main": "cds_node.js"**
**"author": “aann"**

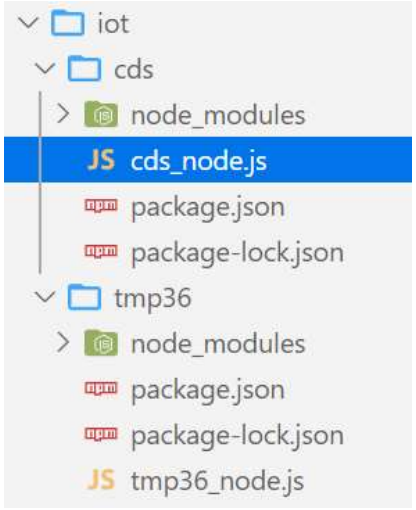
# A4.2.2 Luminosity sensor [ install node modules]

**npm install --save serialport**

**npm install --save socket.io**

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

# A4.2.3 Luminosity sensor [ node code]



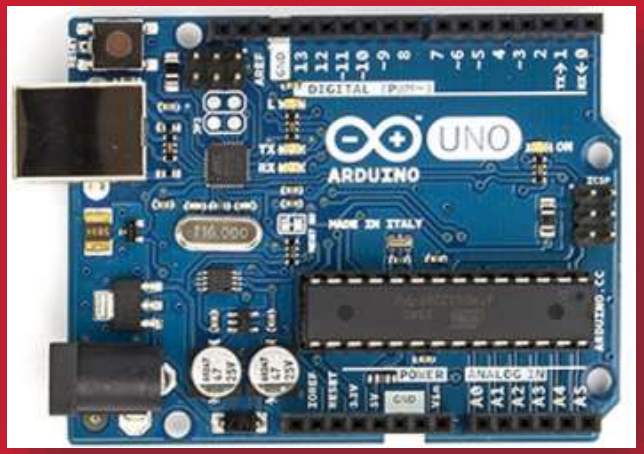

**Save tmp36_node.js as cds_node.js in cds folder**
**(code 재활용)**

```
node cds_node
```

```
NodeJS - node  cds_node

D:\Portable\NodeJSPortable\Data\aa00\iot\cds>node cds_node
AA00,2018-01-14 19:15:33.602,176
AA00,2018-01-14 19:15:34.601,45
AA00,2018-01-14 19:15:35.601,35
AA00,2018-01-14 19:15:36.604,33
AA00,2018-01-14 19:15:37.604,175
```

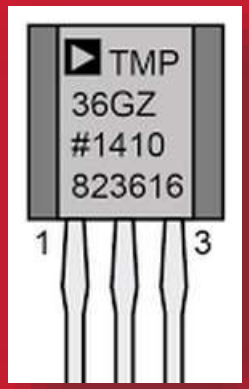
**AAnn_cds_IOT_data.png 로 저장**

# Multiple sensors



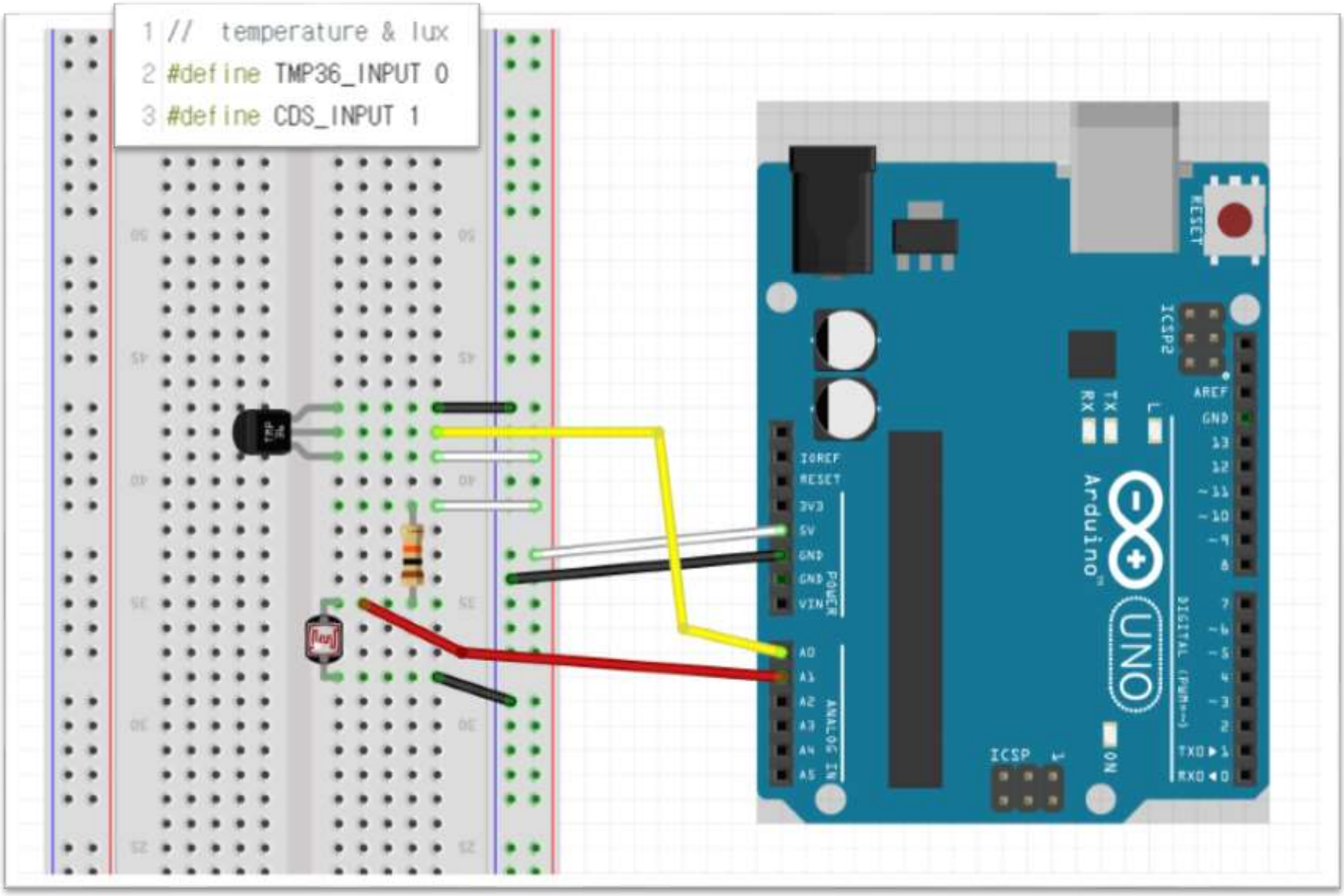

# Arduino

# + Node.js



## Monitoring via Serial monitor & LCD

# A4.3.1 TMP36 + CdS : circuit



# A4.3.2 TMP36 + CdS : code

AAnn_TMP36_CdS§

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

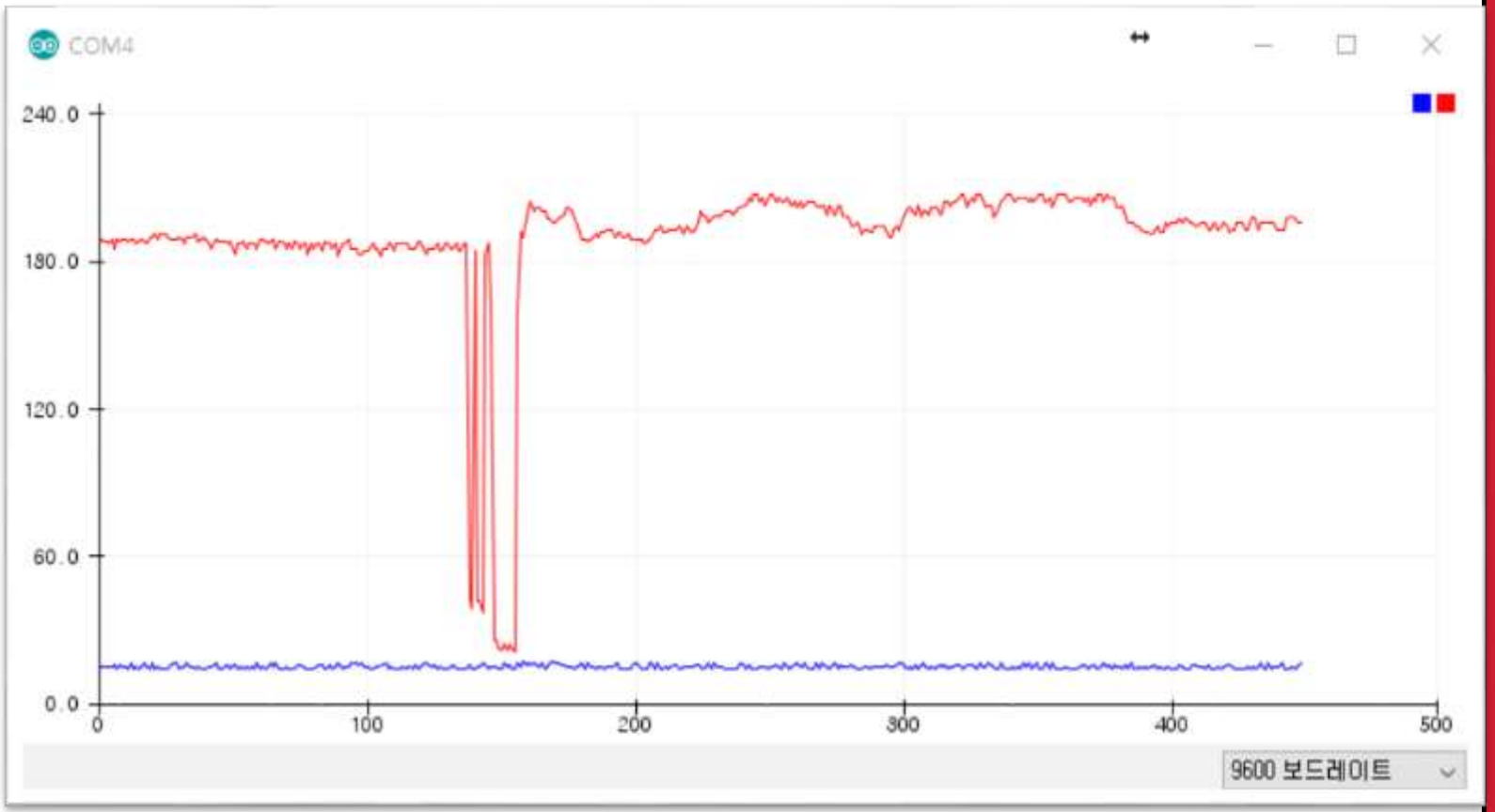
**AAnn_tmp36_cds.ino**

```
8  void loop() {
9    // Temperature from TMP36
10   int temp_value = analogRead(TMP36_INPUT);
11   // converting that reading to voltage
12   float voltage = temp_value * 5.0 * 1000;  // in mV
13   voltage /= 1023.0;
14   float tempC = (voltage - 500) / 10 ;
15
16   // Lux from CdS (LDR)
17   int cds_value = analogRead(CDS_INPUT);
18   int lux = int(luminosity(cds_value));
19 //
20   Serial.print(tempC);
21   Serial.print(",");
22   Serial.println(lux);
23
24   delay(1000);
25 }
26
27 //Voltage to Lux
28 double luminosity (int RawADC0){
29   double Vout=RawADC0*5.0/1023.0;  // 5/1023 (Vin = 5 V)
30   int lux=(2500/Vout-500)/10;
31   // lux = 500 / Rldr, Vout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
32   return lux;
33 }
```

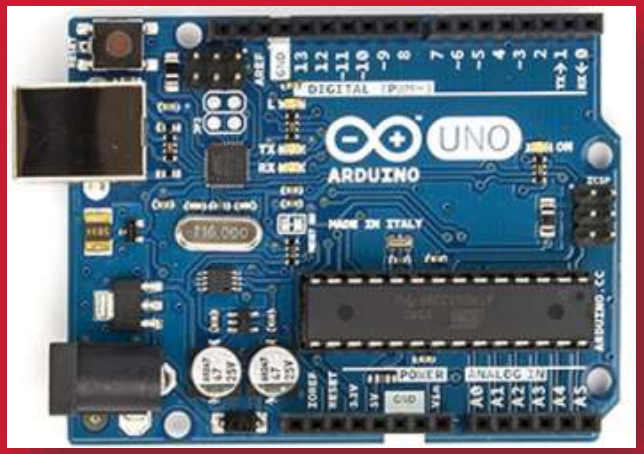
# A4.3.3 TMP36 + CdS : Monitoring

COM4

15.98,192
14.52,194
14.52,193
14.52,193
15.00,180
14.03,18
14.52,17
14.52,16
13.54,15
14.52,191
16.47,188
15.00,188
14.52,190
14.52,190

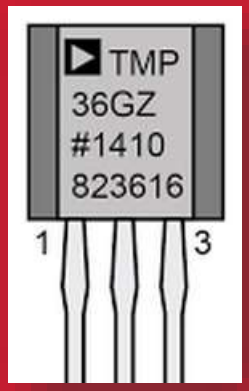


# Multiple sensors




# CdS + TMP36

## Node project



# A4.5.1 CdS + TMP36 + Node project

1. Make cds_tmp36 node project
   - md cds_tmp36 in iot folder
2. Go to cds_tmp36 subfolder
   - Start terminal
   - npm init

"main": "cds_tmp36_node.js"
"author": “aann"

name : cds_tmp36

description : cds-tmp36-node project

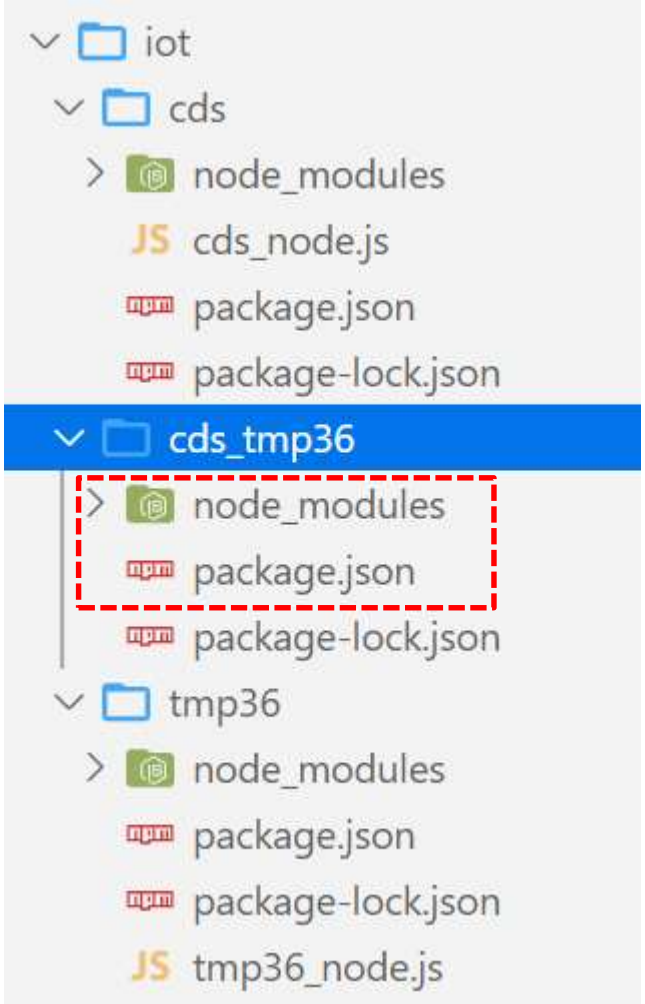
entry point : cds_tmp36_node.js

author : hsnn

# A4.5.2 CdS + TMP36 + Node project

- npm install –save serialport
- npm install –save socket.io

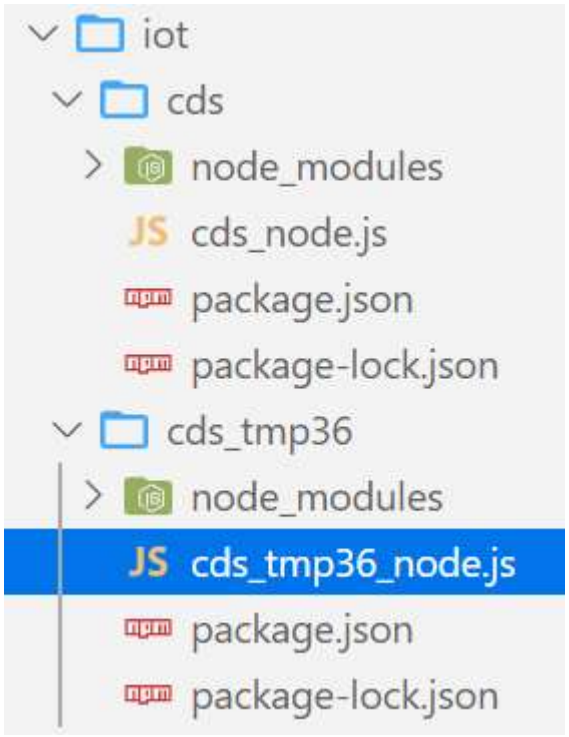
```
"keywords": [
  "cds",
  "tmp36",
  "node"
],
"author": "aa00",
"license": "MIT",
"dependencies": {
  "serialport": "^9.0.1",
  "socket.io": "^2.3.0"
}
```



# A4.5.3 CdS + TMP36 + Node project

**Recycling code:**
코드 재활용

**Save cds_node.js as**
**cds_tmp36_node.js**



# A4.5.4.1 CdS + TMP36 + Node project : code-1

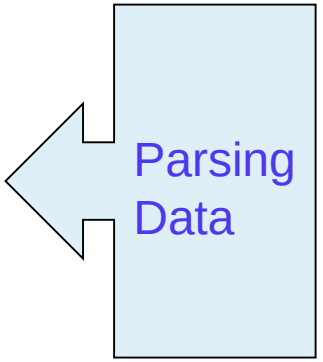
## cds_tmp36_node.js

```
1   // cds_tmp36_node.js
2
3   var serialport = require("serialport");
4   var portName = "COM3"; // check your COM port!!
5   var port = process.env.PORT || 3000;
6
7   var io = require("socket.io").listen(port);
8
9   const Readline = require("@serialport/parser-readline");
10  // serial port object
11  var sp = new serialport(portName, {
12    baudRate: 9600, // 9600  38400
13    dataBits: 8,
14    parity: "none",
15    stopBits: 1,
16    flowControl: false,
17    parser: new Readline("\r\n"),
18  });
19
20  const parser = sp.pipe(new Readline({ delimiter: "\r\n" }));
21
22  // Read the port data
23  sp.on("open", () => {
24    console.log("serial port open");
25  });
```

# A4.5.4.2 CdS + TMP36 + Node project : code-2

## cds_tmp36_node.js – parsing data

```
27  var dStr = "";
28  var readData = "";
29  var temp = "";
30  var lux = "";
31  var mdata = [];
32  var firstcommaidx = 0;
33
34  parser.on("data", (data) => {
35    // call back when data is received
36    readData = data.toString();
37    firstcommaidx = readData.indexOf(",");
38    if (firstcommaidx > 0) {
39      temp = readData.substring(0, firstcommaidx);
40      lux = readData.substring(firstcommaidx + 1);
41      readData = "";
42
43      dStr = getDateString();
44      mdata[0] = dStr; //date
45      mdata[1] = temp; //data
46      mdata[2] = lux;
47      console.log("AA00," + mdata);
48      io.sockets.emit("message", mdata); // send data to all clients
49    } else {
50      console.log(readData);
51    }
52  });
```



# A4.5.4.3 CdS + TMP36 + Node project : code-3

## cds_tmp36_node.js

```
54  io.sockets.on("connection", function (socket) {
55    // If socket.io receives message from the client browser then
56    // this call back will be executed.
57    socket.on("message", function (msg) {
58      console.log(msg);
59    });
60    // If a web browser disconnects from Socket.IO then this callback is called.
61    socket.on("disconnect", function () {
62      console.log("disconnected");
63    });
64  });
65
66  // helper function to get a nicely formatted date string for IOT
67  function getDateString() {
68    var time = new Date().getTime();
69    // 32400000 is (GMT+9 Korea, GimHae)
70    // for your timezone just multiply +/-GMT by 3600000
71    var datestr = new Date(time + 32400000)
72      .toISOString()
73      .replace(/T/, " ")
74      .replace(/Z/, "");
75    return datestr;
76  }
```

# A4.5.5 CdS + TMP36 + Node project : result

**Terminal에서 실행: node cds_tmp36_node**

```
D:\Portable\vscode-portable\data\aa2-00\aa2-99-rpt06\iot\cds_tmp36>node cds_tmp36_node
serial port open
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
```

**IOT data format**

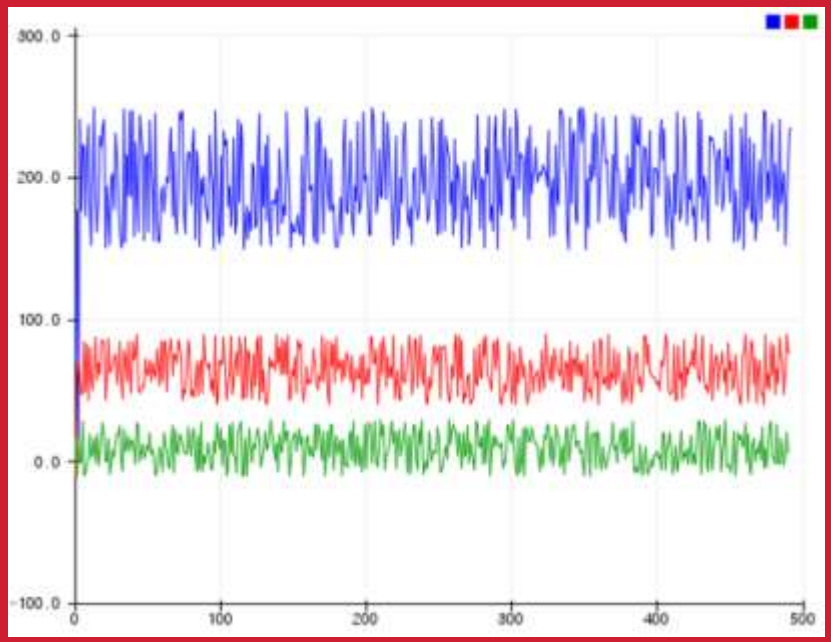
시간, 온도,조도

**Save as**
**AAnn_cds_tmp36_IOT.png**

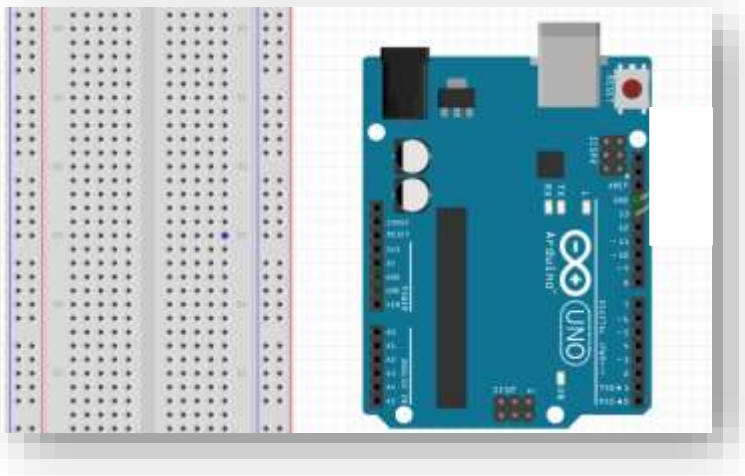
# [DIY] Multi-signals

# 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

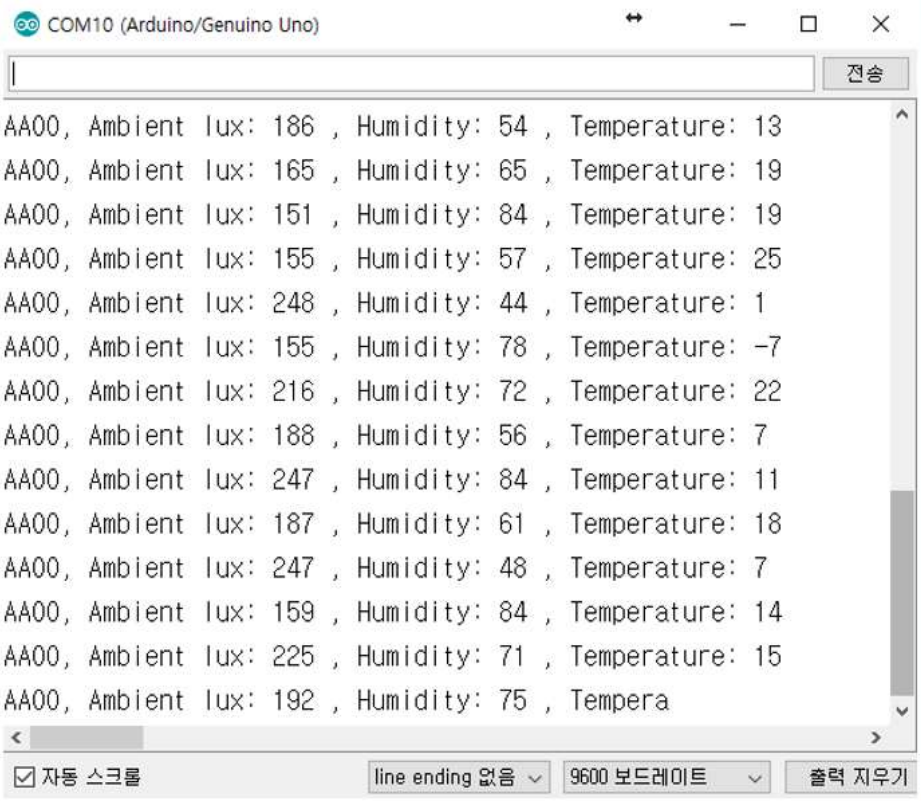
sketch05_multi_signals

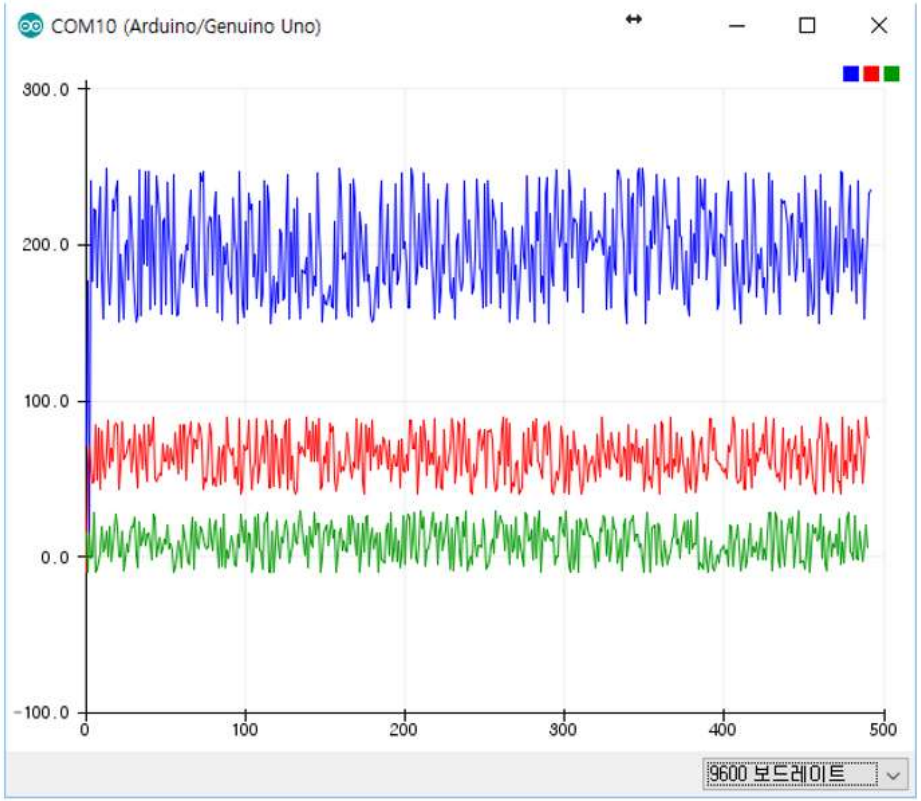
```
/*
  Multi Signals
  Simulation of multiple random signals
*/
// signals
int humi=0;
int temp=0;
int lux=0;

// 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

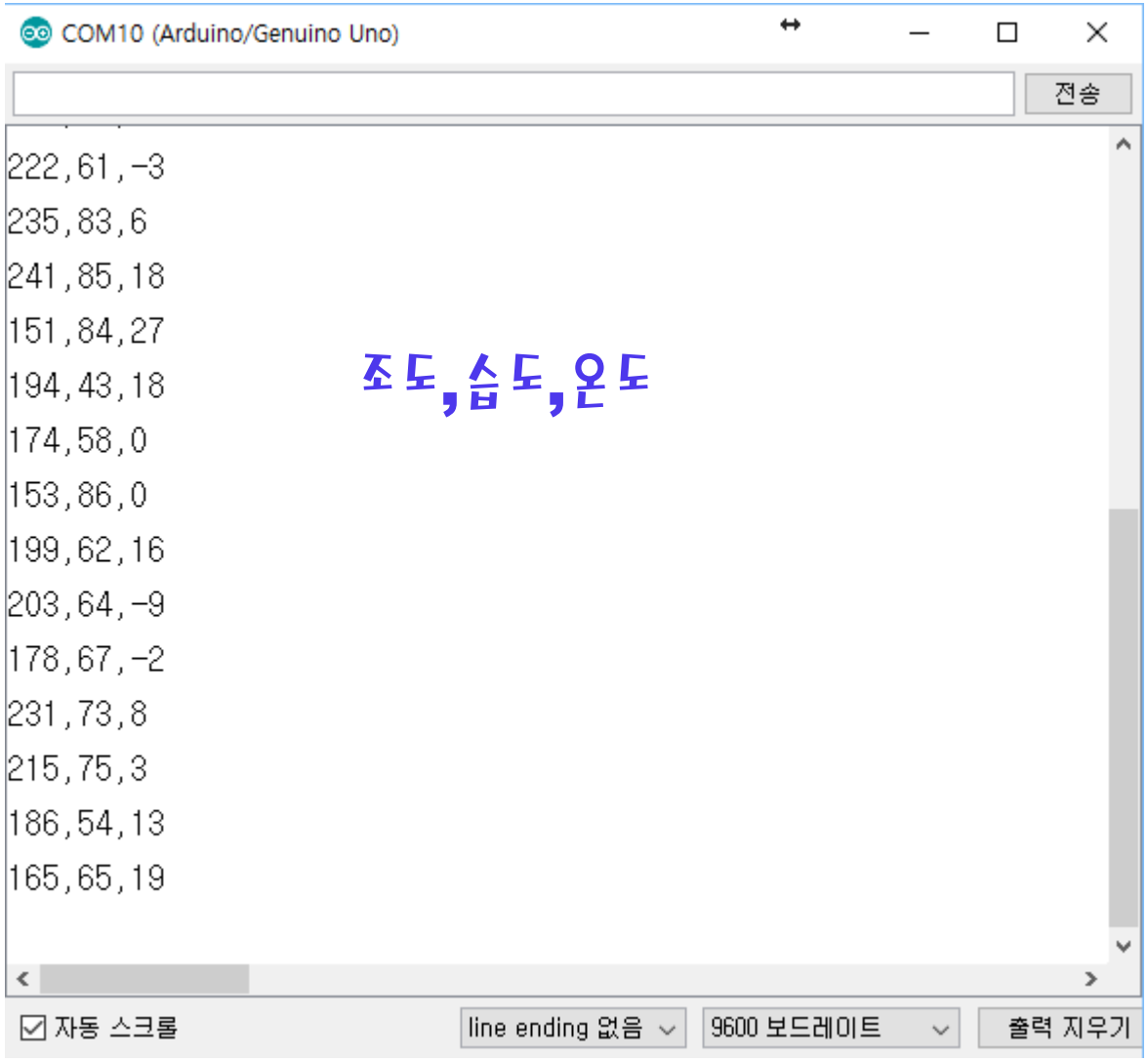
가상적인 세 개의 센서신호 시뮬레이션:조도(위), 습도(중간), 온도(아래).



# 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

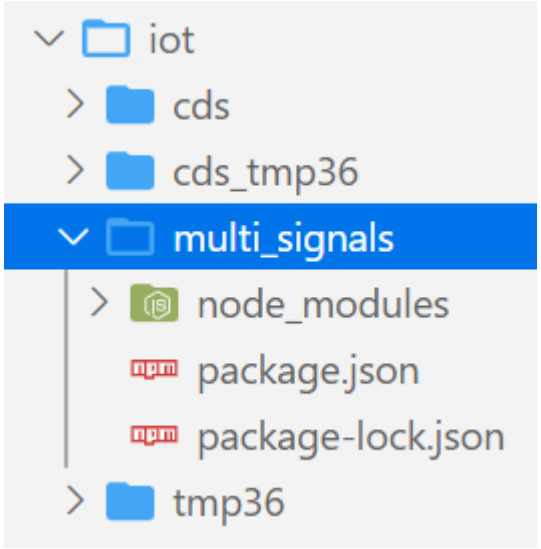
  description : multi-signals-node project

  entry point : aann_multi_signals.js

  author : aann

3. Install node modules

- npm install –save serialport
- npm install –save socket.io



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

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.

# 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);
    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:\Portable\vscode-portable\data\aa2-00\aa2-99-rpt06\iot\multi_signals>node aann_multi_signals
serial port open
AA00,2020-10-13 17:40:06.464,223,47,-1
AA00,2020-10-13 17:40:07.463,222,48,0
AA00,2020-10-13 17:40:08.466,173,84,28
AA00,2020-10-13 17:40:09.469,215,49,-10
AA00,2020-10-13 17:40:10.468,237,82,-8
AA00,2020-10-13 17:40:11.469,179,43,-3
AA00,2020-10-13 17:40:12.468,153,80,2
AA00,2020-10-13 17:40:13.471,207,59,19
AA00,2020-10-13 17:40:14.470,249,50,3
AA00,2020-10-13 17:40:15.469,185,68,6
AA00,2020-10-13 17:40:16.473,162,87,16
AA00,2020-10-13 17:40:17.472,183,57,0
AA00,2020-10-13 17:40:18.476,229,69,19
AA00,2020-10-13 17:40:19.475,222,61,-3
AA00,2020-10-13 17:40:20.475,235,83,6
```

**ID,시간,조도,습도,온도**

**Save this result as AAnn_multi_signals_node .png**

# Codes for single & multi signals

## Single

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

parser.on("data", (data) => {
  // call back when data is received
  // raw data only
  //console.log(data);
  dStr = getDateString();
  tdata[0] = dStr; // date
  tdata[1] = data; // data
  console.log("AA00," + tdata);
  io.sockets.emit("message", tdata); //
});
```

시간,온도

## Multi(2)

```
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

# Codes for multi signals

## Multi(3)

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

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

신호의 개수가 늘어나면 데이터 **parsing** 과정이 복잡해진다.

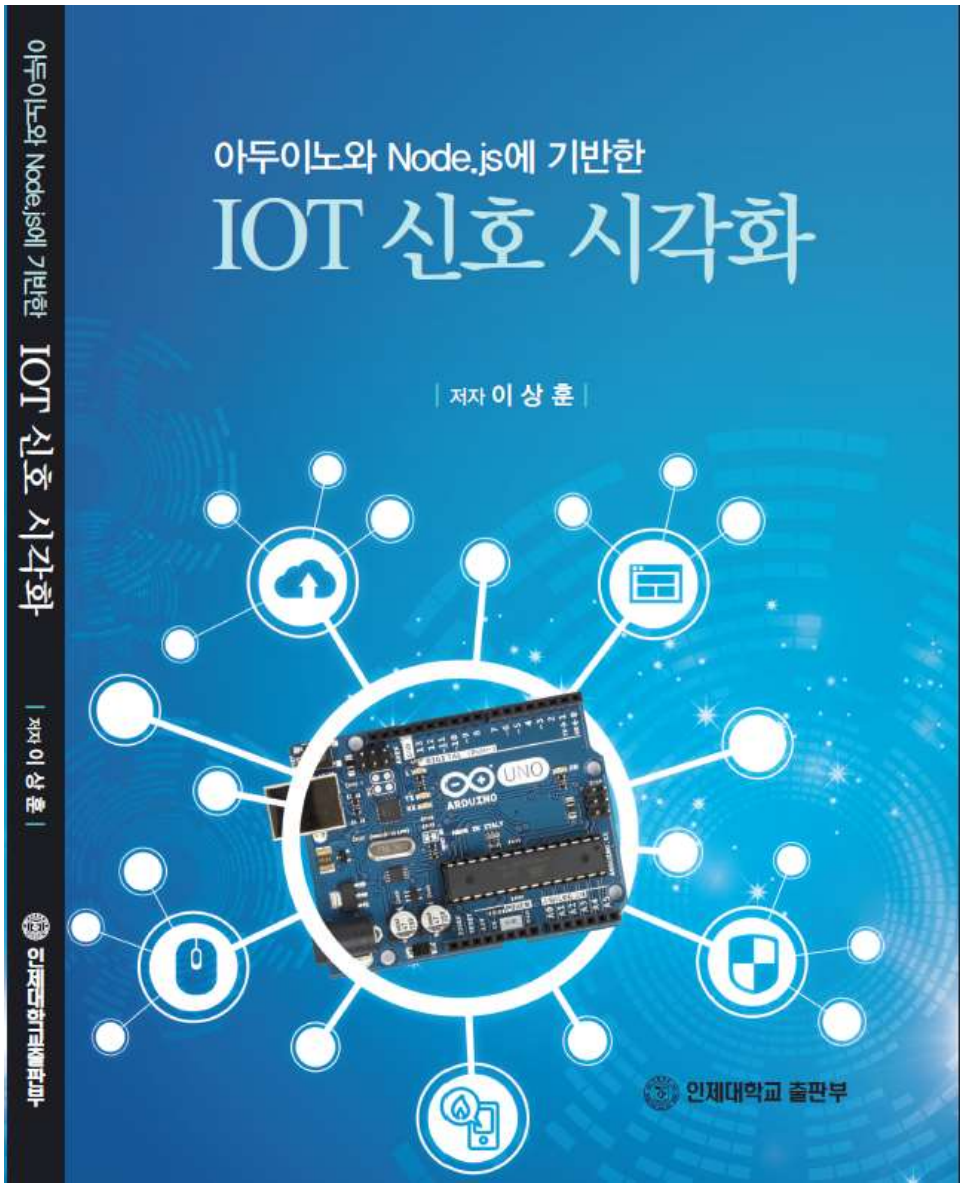
시간,조도,습도,온도

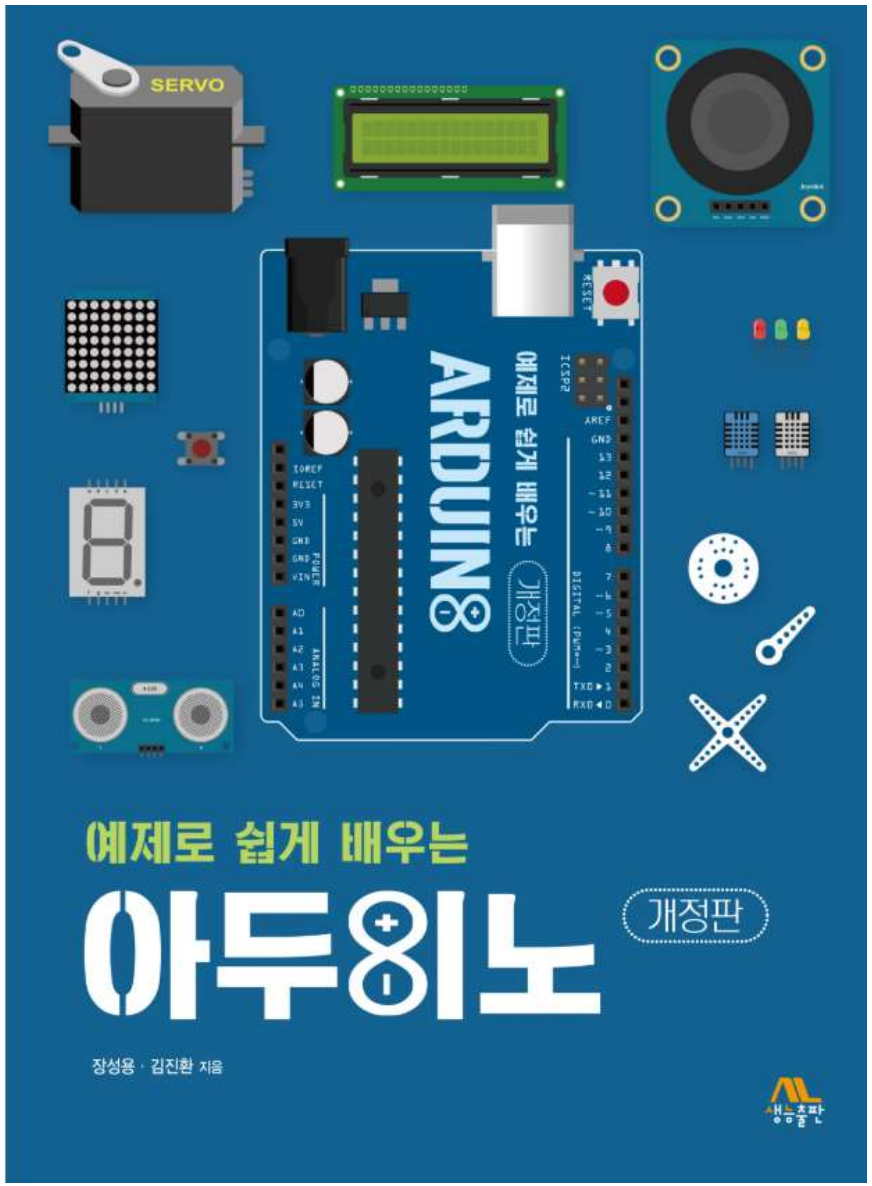
# Lecture materials

## References & good sites

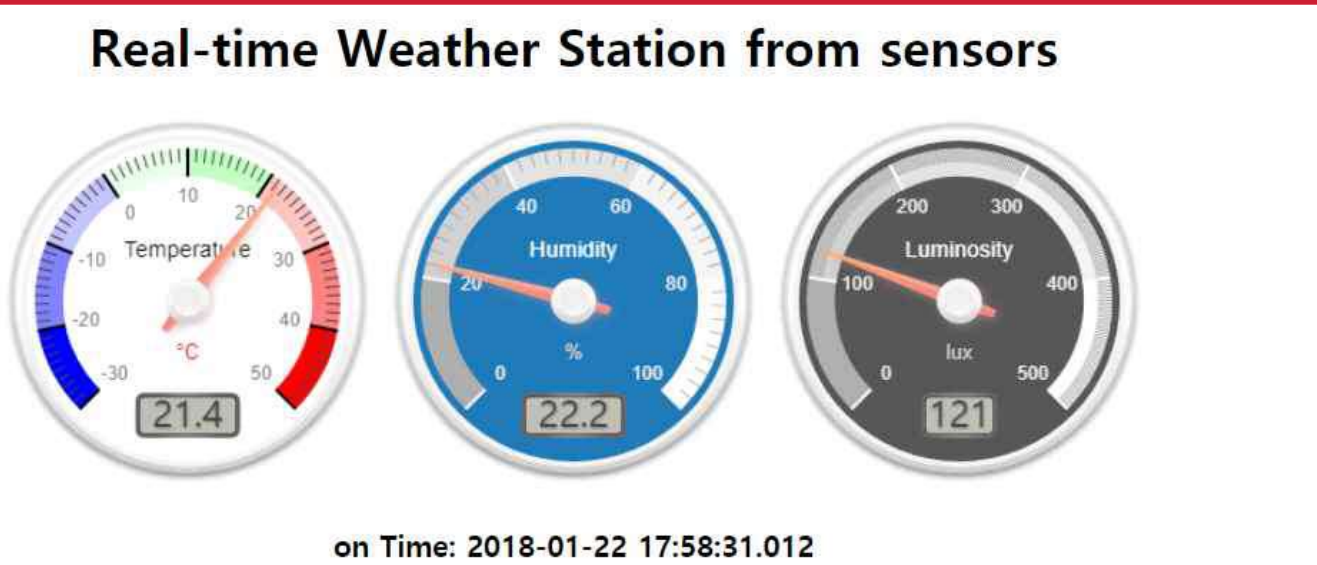
- ✓ http://www.arduino.cc Arduino Homepage
- ✓ http://www.nodejs.org/ko Node.js
- ✓ https://plot.ly/ plotly
- ✓ https://www.mongodb.com/ MongoDB
- ✓ http://www.w3schools.com By w3schools
- ✓ http://www.github.com GitHub

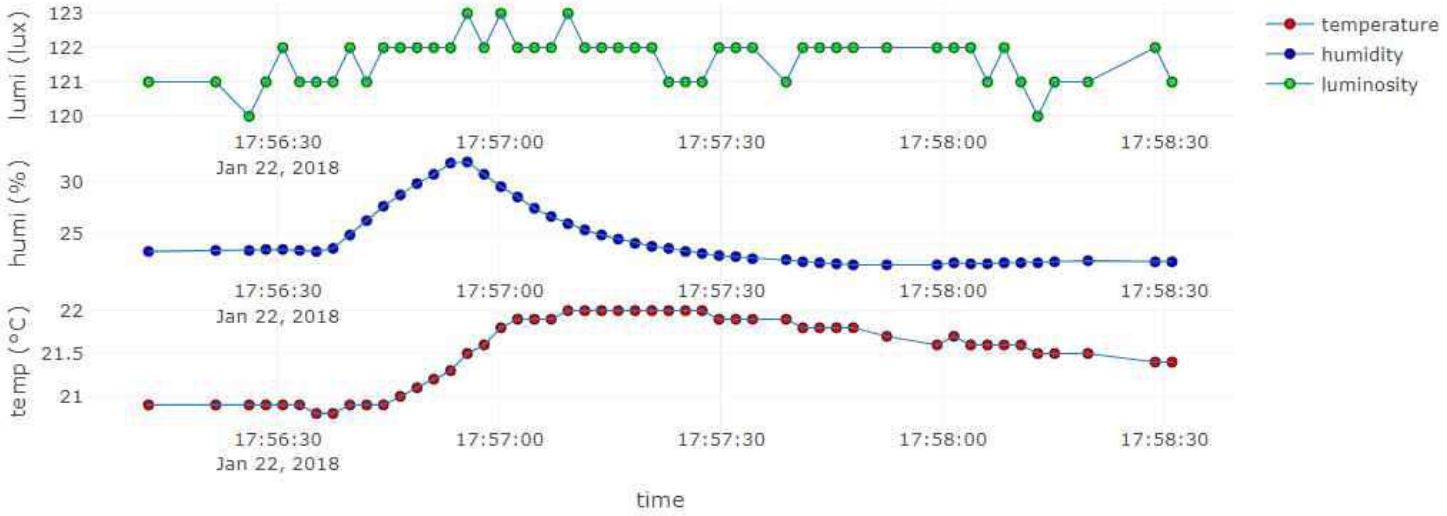
# 주교재 및 참고도서


아두이노와 Node.js에 기반한
IOT 신호 시각화
| 저자 이 상 훈 |
인제대학교 출판부
아두이노와 Node.js에 기반한 IOT 신호 시각화
| 저자 이 상 훈 |
인제대학교 출판부

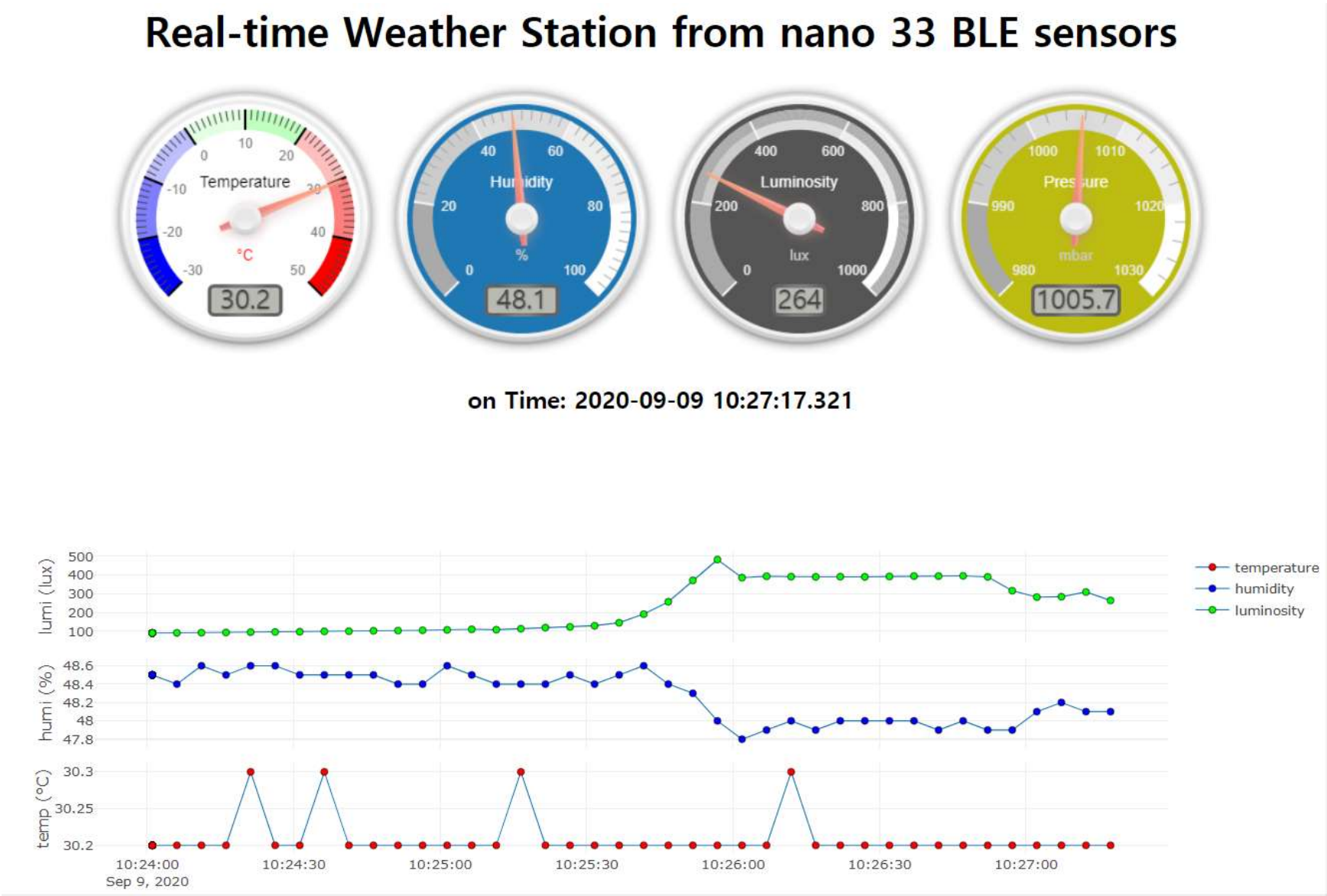
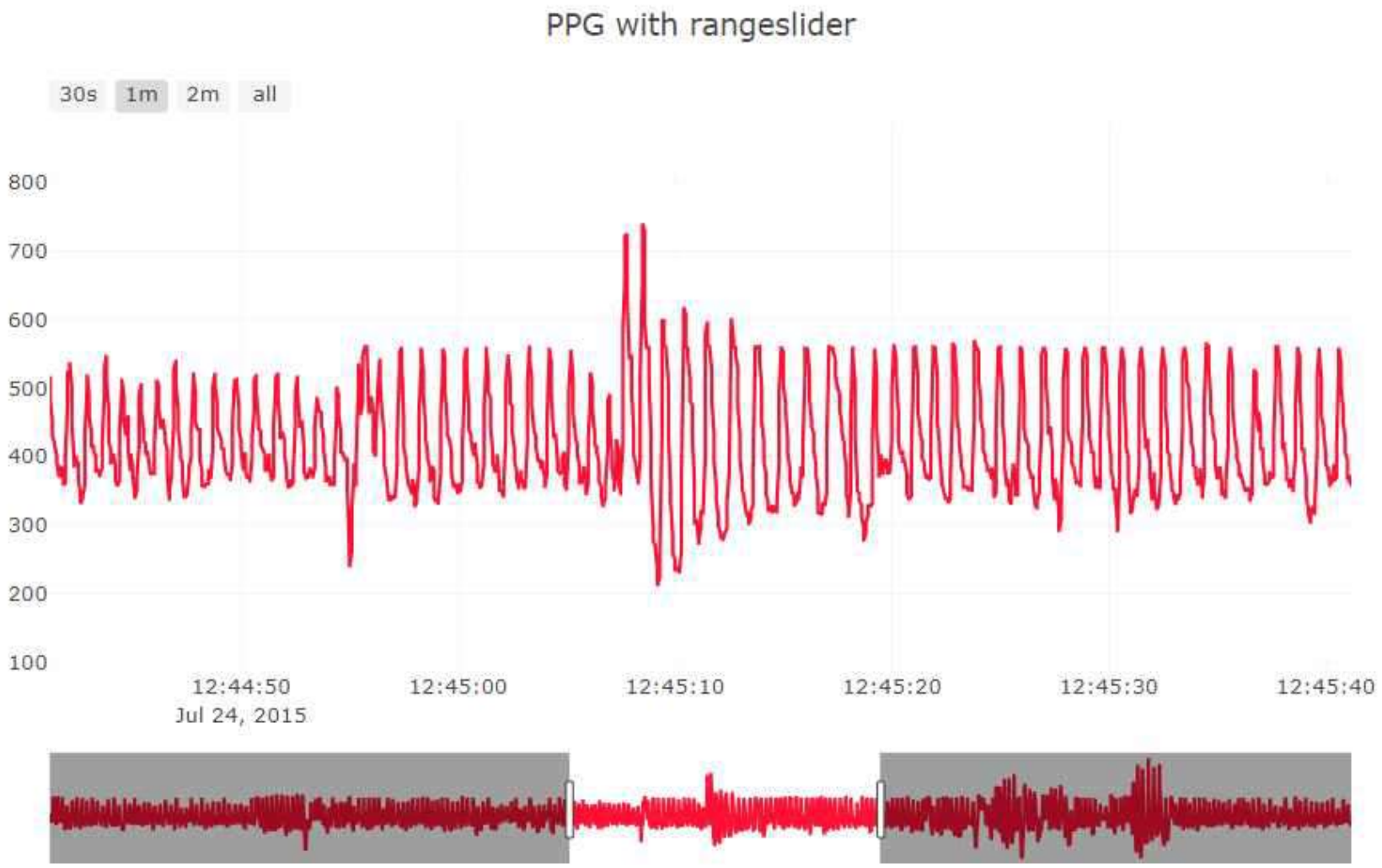

SERVO
ARDUINO
예제로 쉽게 배우는
개정판
예제로 쉽게 배우는
아두이노
개정판
장성용 · 김진환 지음

# Target of this class



# Target of this class



# Another target of this class