



# Arduino-IoT

[wk06]

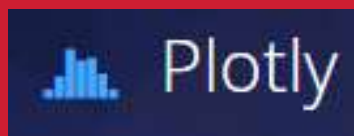
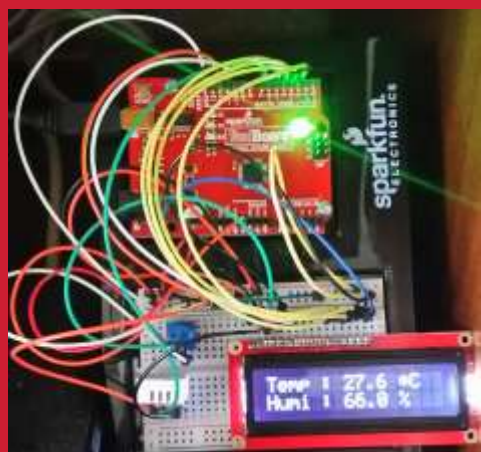
## Arduino-node.js

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

Drone-IoT-Comsi, INJE University

2<sup>nd</sup> semester, 2021

Email : chaos21c@gmail.com





# My ID

## ID를 확인하고 github에 repo 만들기

AA01	김준수	AA13	조재윤
AA02	김현서	AA14	고태승
AA03	박영훈	AA15	이한글
AA04	박윤호	AA16	장세진
AA05	성은지	AA17	장태호
AA06	손윤우	AA18	정지원
AA07	오세윤	AA19	진우태
AA08	우승철	AA20	황혁준
AA09	윤현석	AA21	장이제
AA10	이예주	AA22	박상현
AA11	강지환	AA23	정은성
AA12	성인제	AA24	김경영

위의 id를 이용해서 github에 repo를 만드시오.

Option: 아두이노응용 실습 과제 - AAnn

Public, README.md check



# [Review]

## ◆ [wk05]

- **Arduino sensors**
- **Complete your project**
- **Upload folder: aann-rpt05**
- **Use repo “aann” in github**

## ◆ [Target of this week]

- Complete your works
- Save your outcomes and upload 4 figures in github

### Upload folder : aann-rpt05

#### - 제출할 파일들

- ① **AAnn\_AnalogVoltage.png**
- ② **AAnn\_TMP36.png**
- ③ **AAnn\_LCD\_hello.png**
- ④ **AAnn\_LCD\_lux.png**
- ⑤ **All \*.ino**

# Purpose of AA

주요 수업 목표는 다음과 같다.

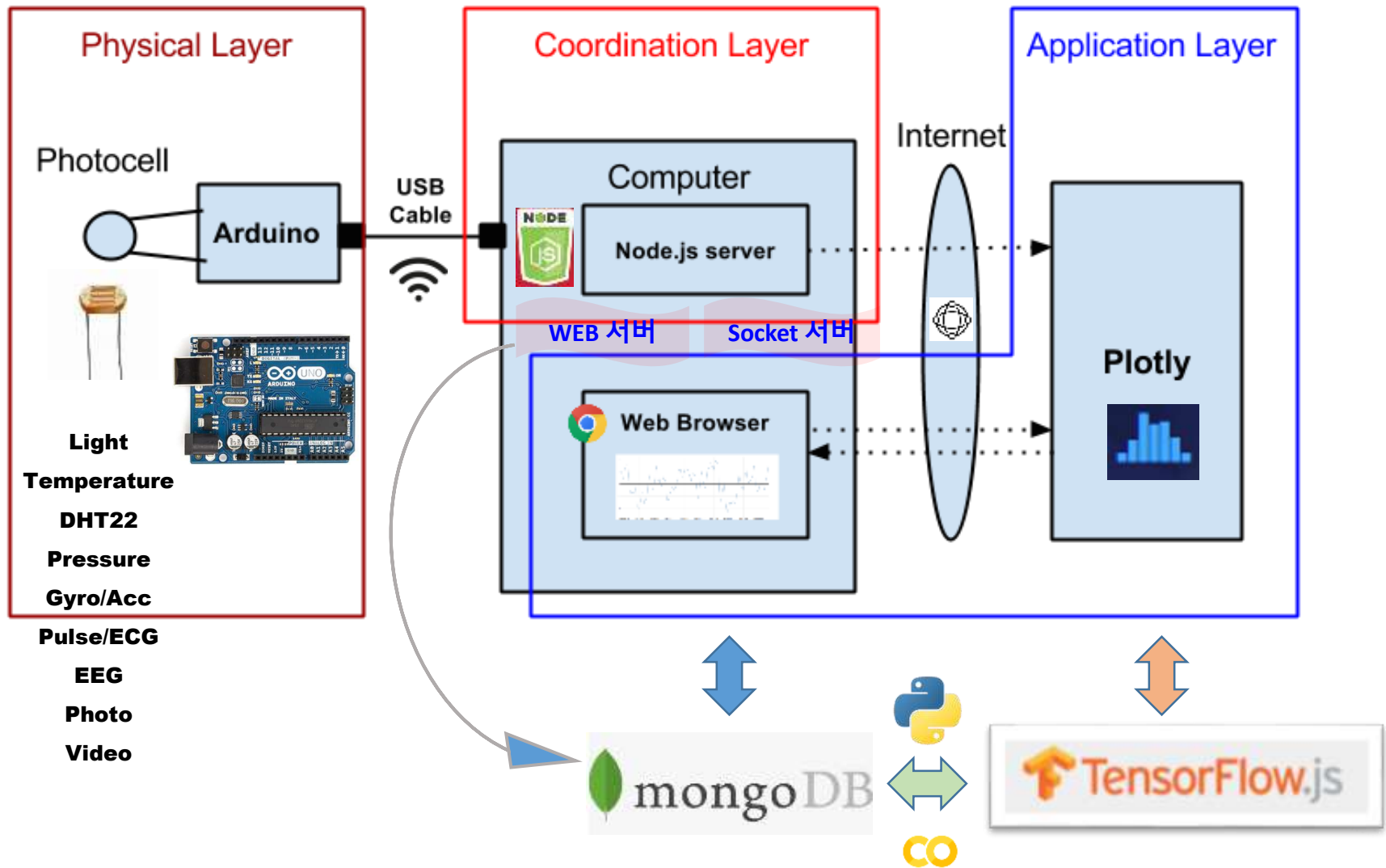
1. Node.js를 이용한 아두이노 센서 신호 처리
2. Plotly.js를 이용한 아두이노 센서 신호 시각화
3. MongoDB에 아두이노 센서 데이터 저장 및 처리



## 4. 저장된 IoT 데이터의 마이닝 (파이썬 코딩)

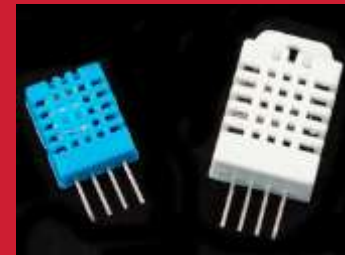
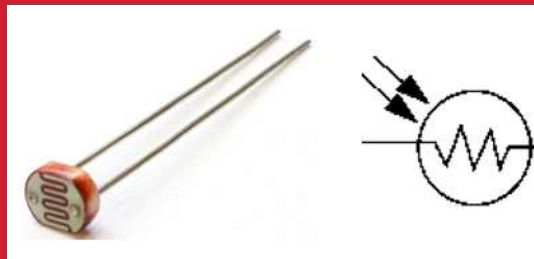
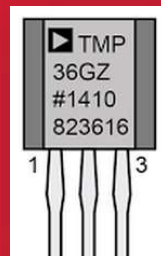
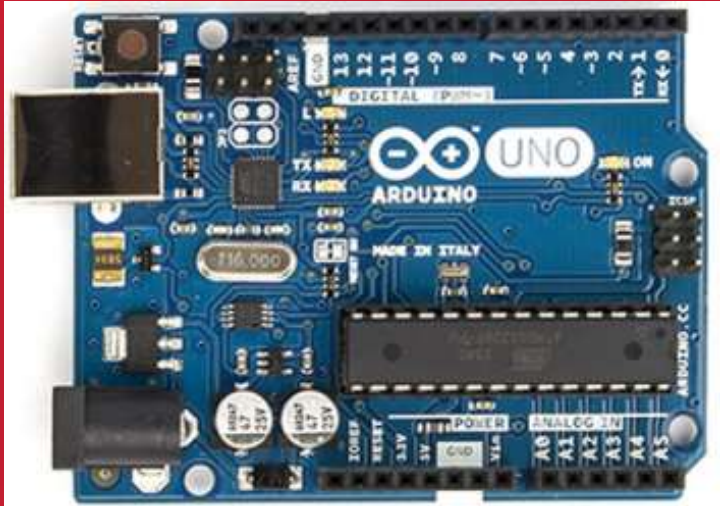


# Layout [H S C]



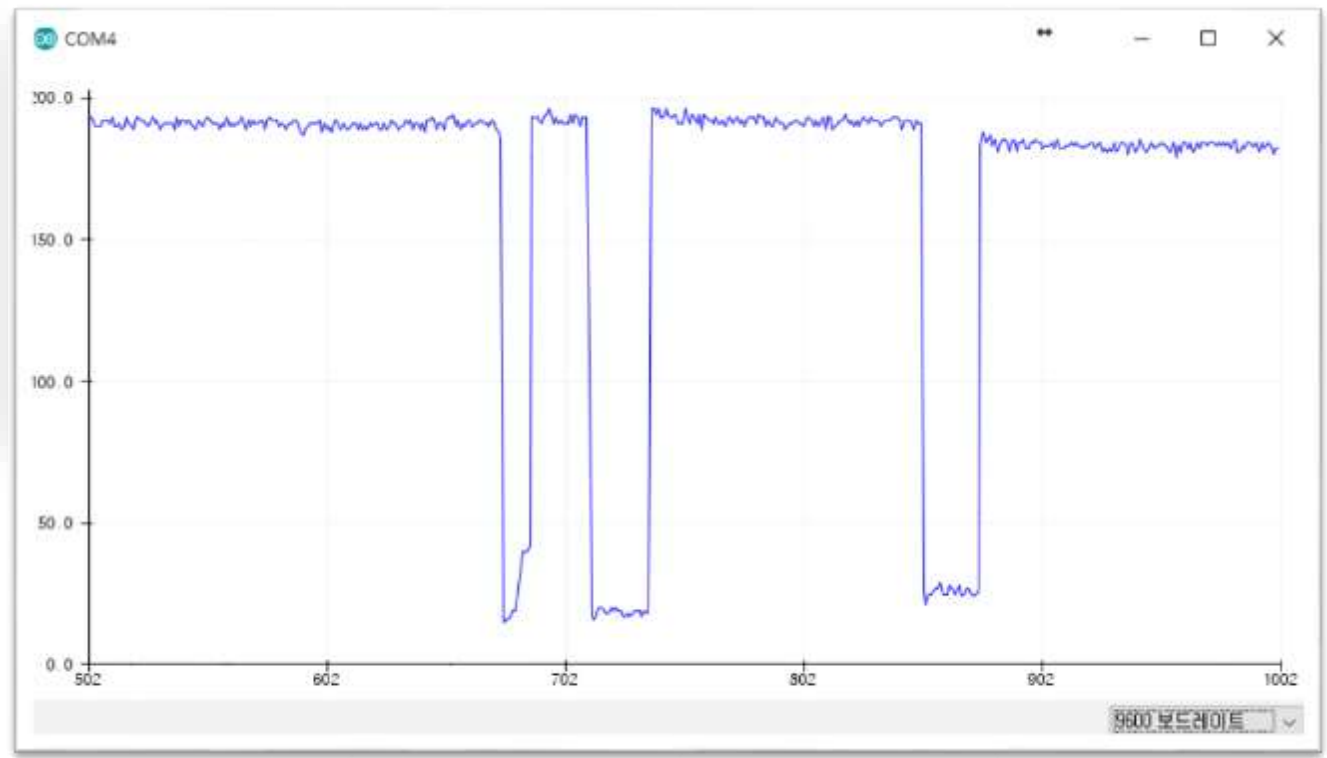
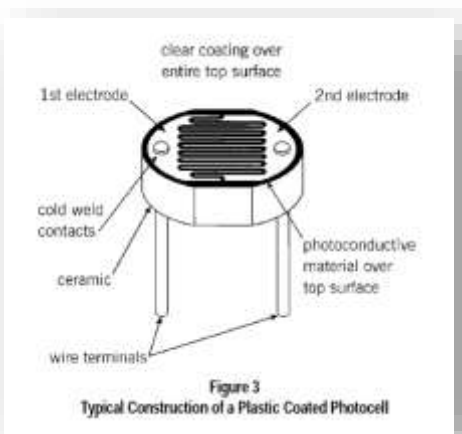


# Arduino Sensors + Node.js



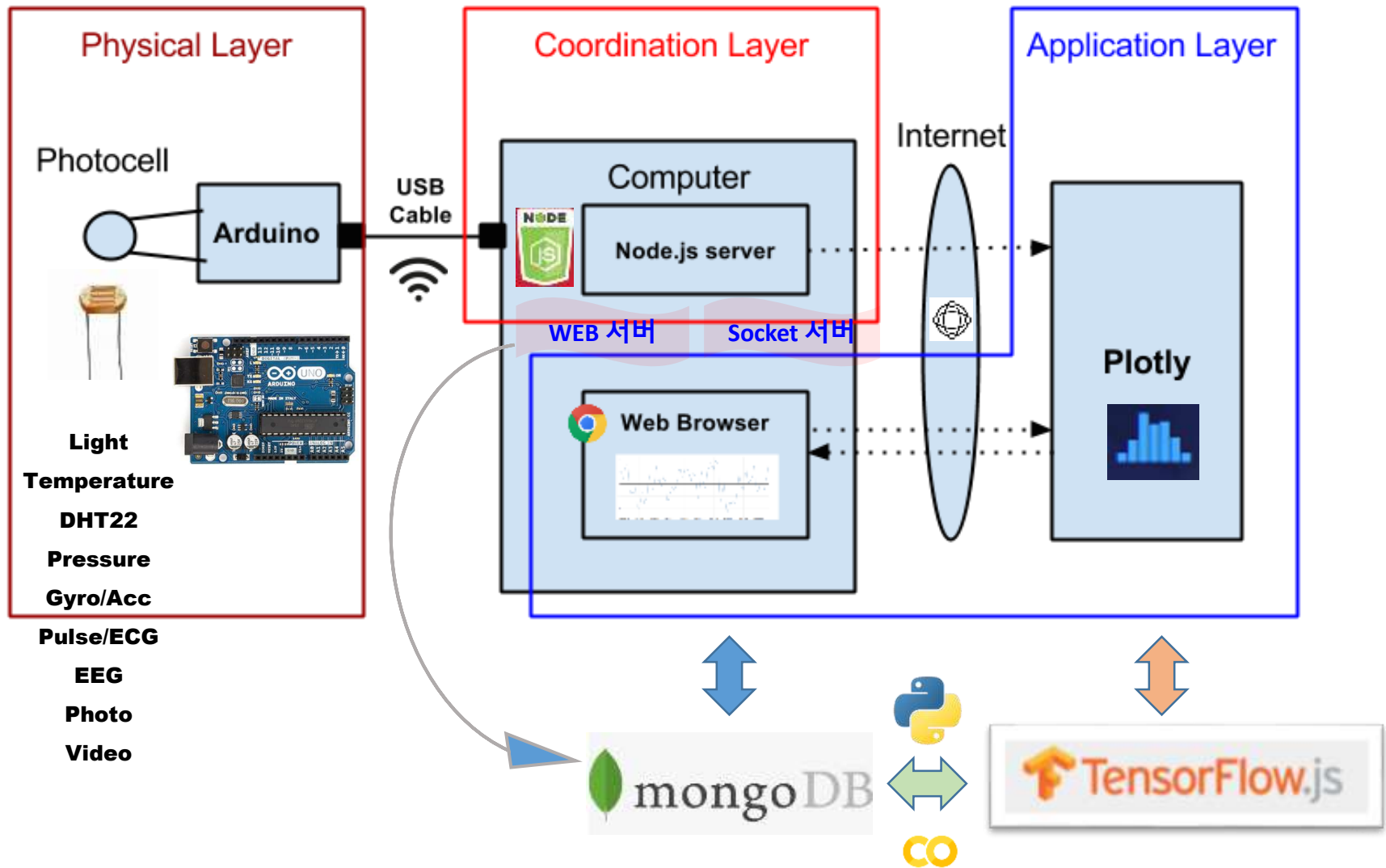


# IOT: HSC





# Layout [H S C]



on WEB monitoring Arduino data

# IoT Signal from Arduino

## Real-time Signals

---

on Time: 2021-10-06 09:49:49.818

Signals (조도, 습도, 온도) : 166,60,-5

---

[http://chaos.inje.ac.kr:3030/iot\\_multi.html](http://chaos.inje.ac.kr:3030/iot_multi.html)

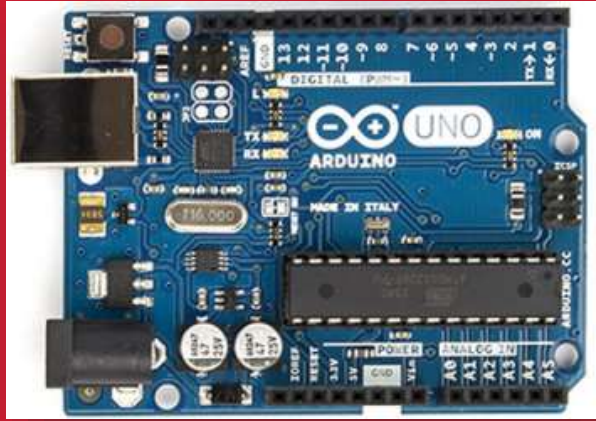
# Arduino data + plotly

## Time series by AA00



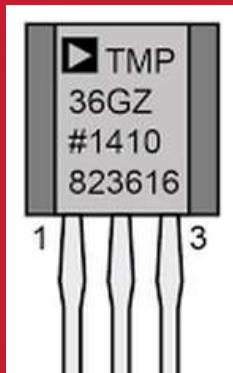


**Single sensor: tmp36**

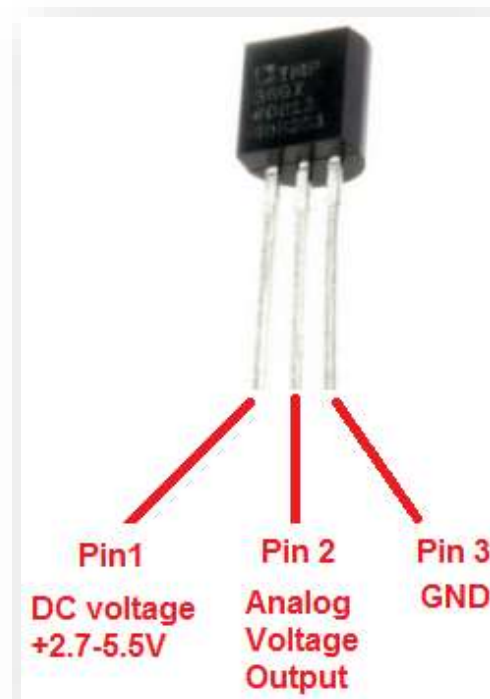
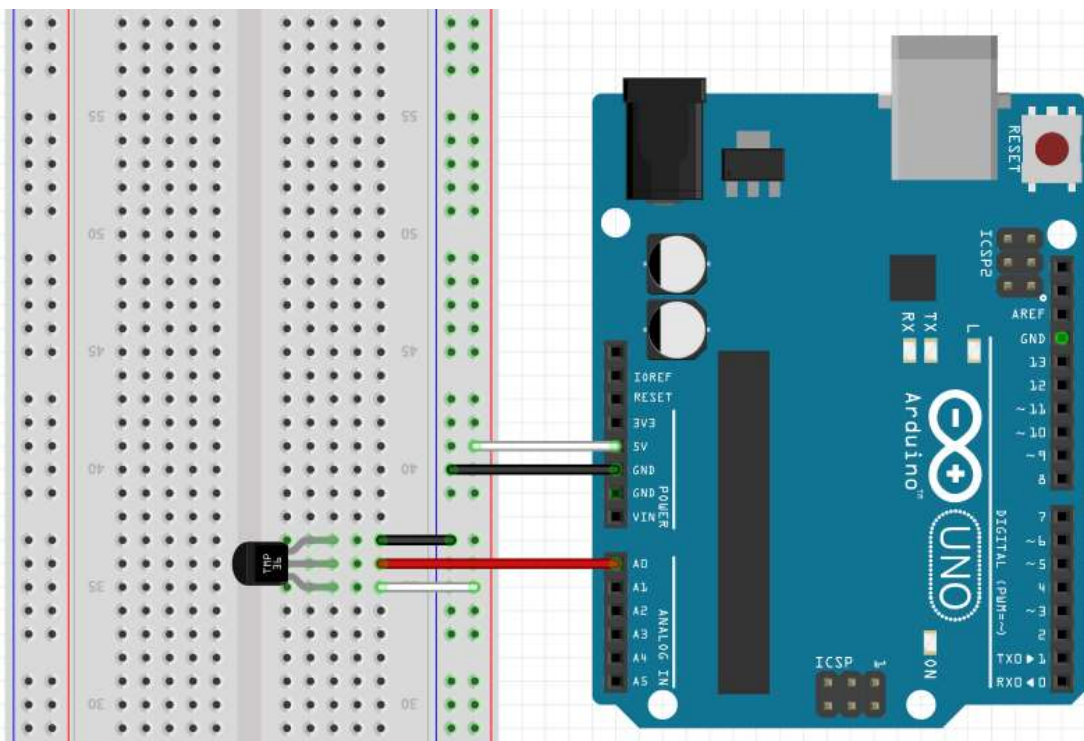


# TMP36

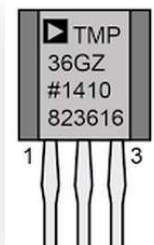
# Node project



# 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: **aann-rpt06**
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

문제

출력

디버그 콘솔

터미널

node + ▾ □ 🗑 ▾ ✕

```
D:\aann\aann-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

탐색기

열려 있는 편집기

× npm package.json aann-rpt06...

제목 없음(작업 ...)

✓ aann

- > aann-rpt01
- > aann-rpt02
- > aann-rpt03
- > aann-rpt04
- > aann-rpt05
- ✓ aann-rpt06
  - > Arduino
  - ✓ iot\ tmp36
    - npm package.json**
    - > Node
    - AA\_수업계획서.pdf

npm package.json ×

./ aann > aann-rpt06 > iot > tmp36 > npm package.json > ...

```
1 {
2   "name": "tmp36",
3   "version": "1.0.0",
4   "description": "tmp36-node project",
5   "main": "tmp36_node.js",
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 }
17
```





## A4.1.4 tmp36 node project: install modules

### npm install --save serialport

```
D:\aann\aann-rpt06\iot\tmp36>npm install --save serialport
```

```
> @serialport/bindings@9.2.4 install D:\aann\aann-rpt06\iot\tmp36\node_modules\@serialport\bindings  
> prebuild-install --tag-prefix @serialport/bindings@ || node-gyp rebuild
```

npm notice created a lockfile as package-lock.json. You should commit this file.

npm WARN tmp36@1.0.0 No repository field.

```
+ serialport@9.2.4  
added 74 packages from 45 contributors and audited 74 packages in 11.774s
```

17 packages are looking for funding  
run `npm fund` for details

found 0 vulnerabilities



# A4.1.4 tmp36 node project: install modules

## socket.io

The screenshot shows a web browser with multiple tabs open, including 'Server API | Socket.IO', 'Server Installation | Socket.IO', 'Get started | Socket.IO', and 'Node.js(Express)와 Socket.io | P...'. The address bar shows the URL 'socket.io/docs/v4/server-installation/'. The page content includes a sidebar with a navigation menu: 'Documentation' (expanded), 'Server' (expanded), 'Installation' (selected), 'Initialization', 'The Server instance', 'The Socket instance', 'Middlewares', 'Behind a reverse proxy', 'Using multiple nodes', 'Handling CORS', and 'Application structure'. The main content area is titled 'Installation' and contains the following text: 'The latest Socket.IO release is:' followed by a code block showing 'npm v4.2.0'. Below this, it says 'To install the latest release:' followed by a code block with the command '\$ npm install socket.io'. Finally, it says 'To install a specific version:' followed by a code block with the command '\$ npm install socket.io@<version>'. The browser's taskbar at the bottom shows various icons including Gmail, 네이버, YouTube, ALIOT, Android, Education, English, News, Research, 하소연, and Getting sta.

<https://socket.io/docs/v4/server-installation/>



## A4.1.4 tmp36 node project: install modules

### npm install --save socket.io@2.3.0

```
D:\aann\aann-rpt06\iot\tmp36>npm install --save socket.io@2.3.0
npm WARN tmp36@1.0.0 No repository field.
```

```
+ socket.io@2.3.0
added 52 packages from 33 contributors and audited 126 packages in 3.878s
```

```
17 packages are looking for funding
  run `npm fund` for details
```

```
found 4 vulnerabilities (2 moderate, 1 high, 1 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
```

D:\aann\aann-rpt06\iot\tmp36 디렉터리

2021-10-05	오전 10:23	<DIR>	.
2021-10-05	오전 10:23	<DIR>	.
2021-10-05	오전 10:23	<DIR>	node_modules
2021-10-05	오전 10:23		28,477 package-lock.json
2021-10-05	오전 10:23		367 package.json
	2개 파일		28,844 바이트
	3개 디렉터리		2,424,474,251,264 바이트 남음



## A4.1.4 tmp36 node project: install modules

### **npm install --save socket.io@2.3.0**

found 4 vulnerabilities (2 moderate, 1 high, 1 critical)  
run `npm audit fix` to fix them, or `npm audit` for details

```
D:\aann\aann-rpt06\iot\tmp36>npm audit fix  
npm WARN tmp36@1.0.0 No repository field.
```

+ socket.io@2.4.1

added 1 package, removed 11 packages, updated 11 packages and moved 1 package in 1.33s

17 packages are looking for funding  
run `npm fund` for details

fixed 1 of 4 vulnerabilities in 126 scanned packages  
1 package update for 3 vulnerabilities involved breaking changes  
(use `npm audit fix --force` to install breaking changes; or refer to  
`npm audit` for steps to fix these manually)



## A4.1.4 tmp36 node project: install modules

**npm install --save socket.io [N.A.]**

**4.x 버전 설치는 좀 더 검토가 필요.**

```
D:\aann\aann-rpt06\iot\tmp36>npm install --save socket.io
npm WARN tmp36@1.0.0 No repository field.
```

```
+ socket.io@4.2.0
added 20 packages from 66 contributors and audited 94 packages in 2.046s
```

```
17 packages are looking for funding
  run `npm fund` for details
```

```
found 0 vulnerabilities
```

D:\aann\aann-rpt06\iot\tmp36 디렉터리

2021-10-05	오전 10:23	<DIR>	.
2021-10-05	오전 10:23	<DIR>	.
2021-10-05	오전 10:23	<DIR>	node_modules
2021-10-05	오전 10:23		28,477 package-lock.json
2021-10-05	오전 10:23		367 package.json
	2개 파일		28,844 바이트
	3개 디렉터리		2,424,474,251,264 바이트 남음



## A4.1.4 tmp36 node project: install modules

정상 동작 버전을 설치!

**npm install --save serialport**

**npm install --save socket.io@2.4.1**

```
"author": "aa00",  
"license": "MIT",  
"dependencies": {  
  "serialport": "^9.2.4",  
  "socket.io": "^2.4.1"  
}
```



# 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

COM4 (Arduino/Genuino Uno)

```
value = 150 : 733.14 mV, 23.31 degrees C  
value = 153 : 747.80 mV, 24.78 degrees C  
value = 150 : 733.14 mV, 23.31 degrees C  
value = 150 : 733.14 mV, 23.31 degrees C  
value = 150 : 733.14 mV, 23.31 degrees C  
value = 150 : 733.14 mV, 23.31 degrees C  
value = 150 : 733.14 mV, 23.31 degrees C  
value = 150 : 733.14 mV, 23.31 degrees C  
value = 149 : 728.25 mV, 22.83 degrees C  
value = 150 : 733.14 mV, 23.31 degrees C  
value = 149 : 728.25 mV, 22.83 degrees C  
value = 150 : 733.14 mV, 23.31 degrees C  
value = 150 : 733.14 mV, 23.31 degrees C  
value = 149 : 728.25 mV, 22.83 degrees C
```

☒ 자동 스크롤 ☐ 타임스탬프 표시 line ending 없음



## 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 clients
37 });
```





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

[Terminal] `node tmp36_node.js`

```
D:\aann\aann-rpt06\iot\tmp36>node tmp36_node.js
serial port open
67.35 mV, 26.74 degrees C
67.35 mV, 26.74 degrees C
7 : 767.35 mV, 26.74 degrees C
7 : 767.35 mV, 26.74 degrees C
AA00, value = 157 : 767.35 mV, 26.74 degrees C
AA00, value = 157 : 767.35 mV, 26.74 degrees C
AA00, value = 157 : 767.35 mV, 26.74 degrees C
AA00, value = 157 : 767.35 mV, 26.74 degrees C
AA00, value = 155 : 757.58 mV, 25.76 degrees C
AA00, value = 155 : 757.58 mV, 25.76 degrees C
AA00, value = 156 : 762.46 mV, 26.25 degrees C
AA00, value = 156 : 762.46 mV, 26.25 degrees C
AA00, value = 156 : 762.46 mV, 26.25 degrees C
AA00, value = 156 : 762.46 mV, 26.25 degrees C
AA00, value = 155 : 757.58 mV, 25.76 degrees C
AA00, value = 155 : 757.58 mV, 25.76 degrees C
AA00, value = 156 : 762.46 mV, 26.25 degrees C
AA00, value = 156 : 762.46 mV, 26.25 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;  
  tdata[1] = data; // data  
  console.log("AA00," + tdata.toString());  
  io.sockets.emit("message", tdata); // send  
});
```

```
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:\aann\aann-rpt06\iot\tmp36>node tmp36_node  
serial port open  
AA00,2021-10-05 11:21:24.062,lue = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:24.062,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:24.062,value = 157 : 767.35 mV, 26.74 degrees C  
AA00,2021-10-05 11:21:24.062,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:24.063,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:24.063,value = 157 : 767.35 mV, 26.74 degrees C  
AA00,2021-10-05 11:21:25.644,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:26.648,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:27.651,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:28.651,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:29.655,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:30.658,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:31.662,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:32.661,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:33.665,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:34.669,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:35.672,value = 156 : 762.46 mV, 26.25 degrees C  
AA00,2021-10-05 11:21:36.676,value = 155 : 757.58 mV, 25.76 degrees C  
AA00,2021-10-05 11:21:37.675,value = 156 : 762.46 mV, 26.25 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

COM4 (Arduino/Genuino Uno)

```
23.31
23.80
24.29
23.80
24.29
24.78
24.29
25.27
25.27
25.27
25.27
25.27
```



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

### [Terminal] node tmp36\_node

```
D:\aann\aann-rpt06\iot\tmp36>node tmp36_node
```

```
serial port open
```

```
AA00,2021-10-05 11:31:03.941,26.25
```

```
AA00,2021-10-05 11:31:04.944,26.25
```

```
AA00,2021-10-05 11:31:05.945,26.25
```

```
AA00,2021-10-05 11:31:06.948,26.25
```

```
AA00,2021-10-05 11:31:07.951,26.25
```

```
AA00,2021-10-05 11:31:08.951,26.25
```

```
AA00,2021-10-05 11:31:09.954,25.76
```

```
AA00,2021-10-05 11:31:10.954,26.25
```

```
AA00,2021-10-05 11:31:11.958,26.25
```

```
AA00,2021-10-05 11:31:12.957,26.25
```

```
AA00,2021-10-05 11:31:13.961,26.25
```

```
AA00,2021-10-05 11:31:14.964,26.25
```

```
AA00,2021-10-05 11:31:15.964,26.25
```

시간, 온도

IOT data format

시간, data

시간, 온도

AAnn\_tmp36\_IOT\_data.png

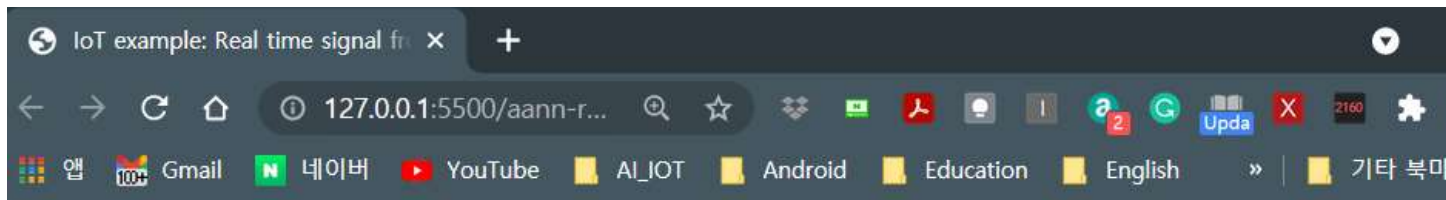
로 저장

공백없이 “,”로  
시간과 온도 구분



# A4.1.11 tmp36 node project (web monitoring)

[Web monitoring] [client\\_signal\\_tmp36.html](#)



## IoT Signal from Arduino

### Real-time Signals

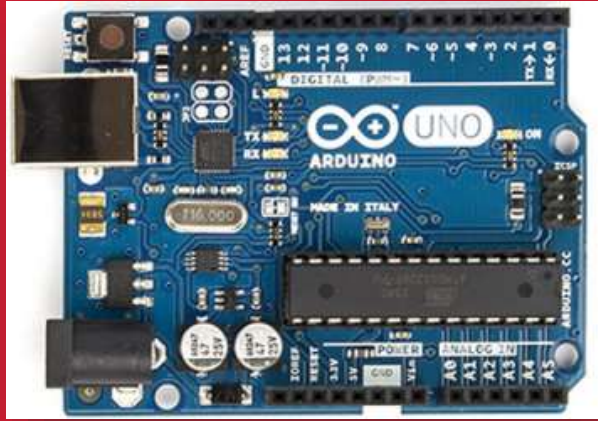
---

on Time: 2021-10-05 11:47:53.803

Signal (temp) : 25.76

---



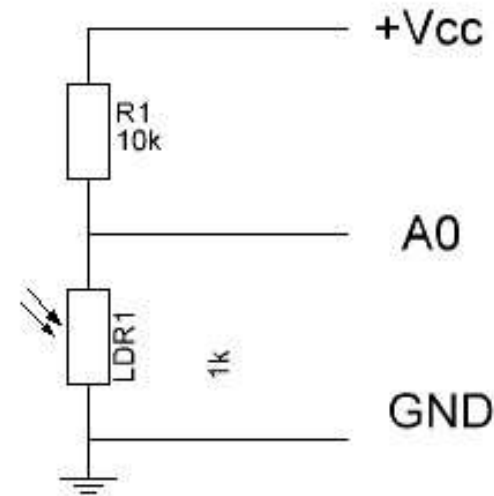
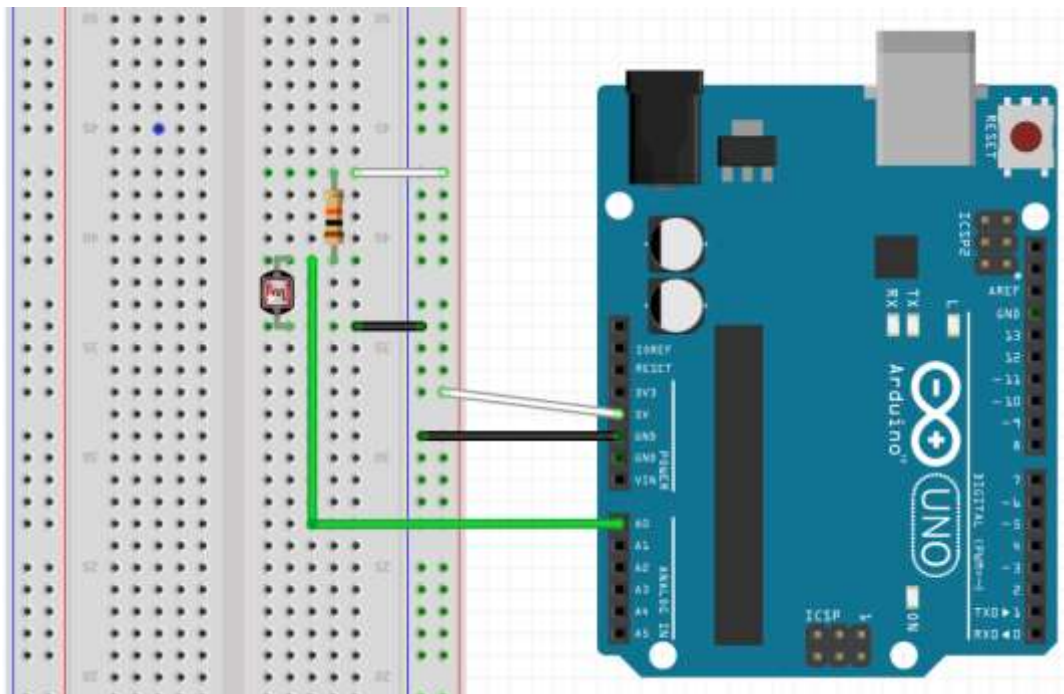


**Single sensor: CdS**

**CdS (LDR)**

**Node project**

## CdS 센서 회로



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

광센서에서의 전압 강하 값을 **A0**로 측정



## CdS 센서 회로 - 측정 2.

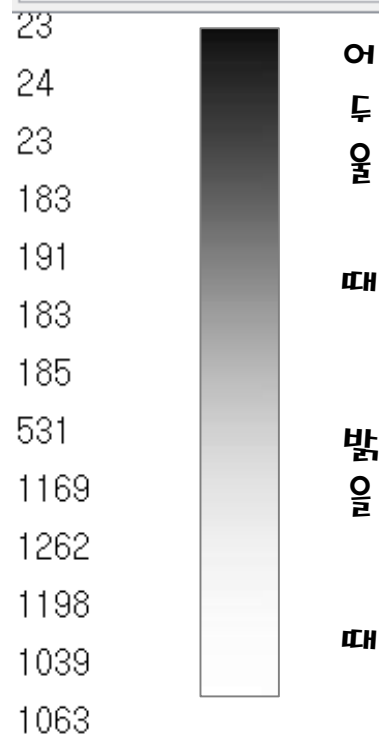
### AAnn\_cds\_start.ino

```

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 Yout=RawADC0*5.0/1023; // 5/1023 (Vin = 5 V)
16   double lux=(2500/Yout-500)/10;
17   // lux = 500 / Rldr, Yout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
18   return lux;
19 }

```

COM11 (Arduino/Genuino Uno)



밝을수록 측정 값이 커지고  
어두울수록 값이 작아진다 !!!

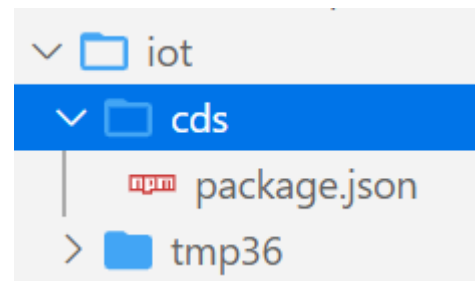




## A4.2.1 Luminosity sensor [ npm init ]

### 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]

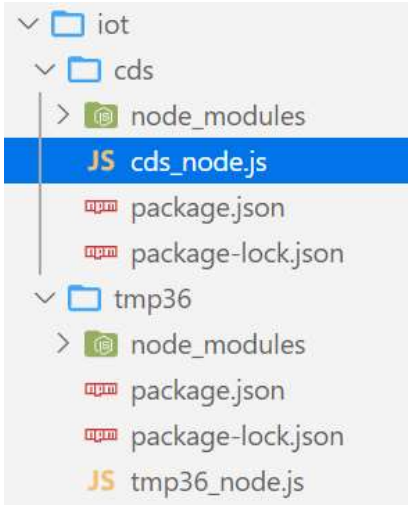
```
npm install --save serialport
```

```
npm install --save socket.io@2.4.1
```

```
1  {
2    "name": "cds",
3    "version": "1.0.0",
4    "description": "cds node project",
5    "main": "cds_node.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [
10     "cds",
11     "node"
12   ],
13   "author": "aa00",
14   "license": "MIT",
15   "dependencies": {
16     "serialport": "^9.2.4",
17     "socket.io": "^2.4.1"
18   }
19 }
```



## A4.2.3 Luminosity sensor [ node code]



Save tmp36\_node.js as **cds\_node.js**  
in cds folder  
(code 재 활용)

```
D:\aann\aann-rpt06\iot\cds>node cds_node
serial port open
AA00,2021-10-06 11:22:58.665,82
AA00,2021-10-06 11:22:59.669,83
AA00,2021-10-06 11:23:00.668,82
AA00,2021-10-06 11:23:01.672,83
AA00,2021-10-06 11:23:02.672,82
AA00,2021-10-06 11:23:03.675,82
AA00,2021-10-06 11:23:04.675,82
AA00,2021-10-06 11:23:05.678,82
AA00,2021-10-06 11:23:06.678,83
```



## A4.2.4 CdS node project (web monitoring)

[Web monitoring] [client\\_signal\\_cds.html](#)

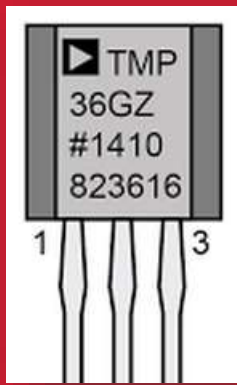
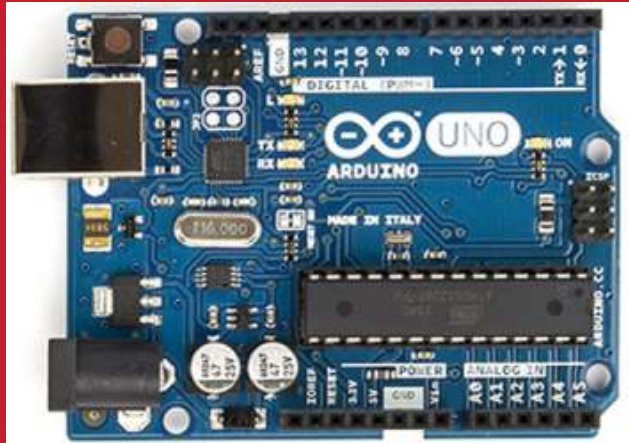




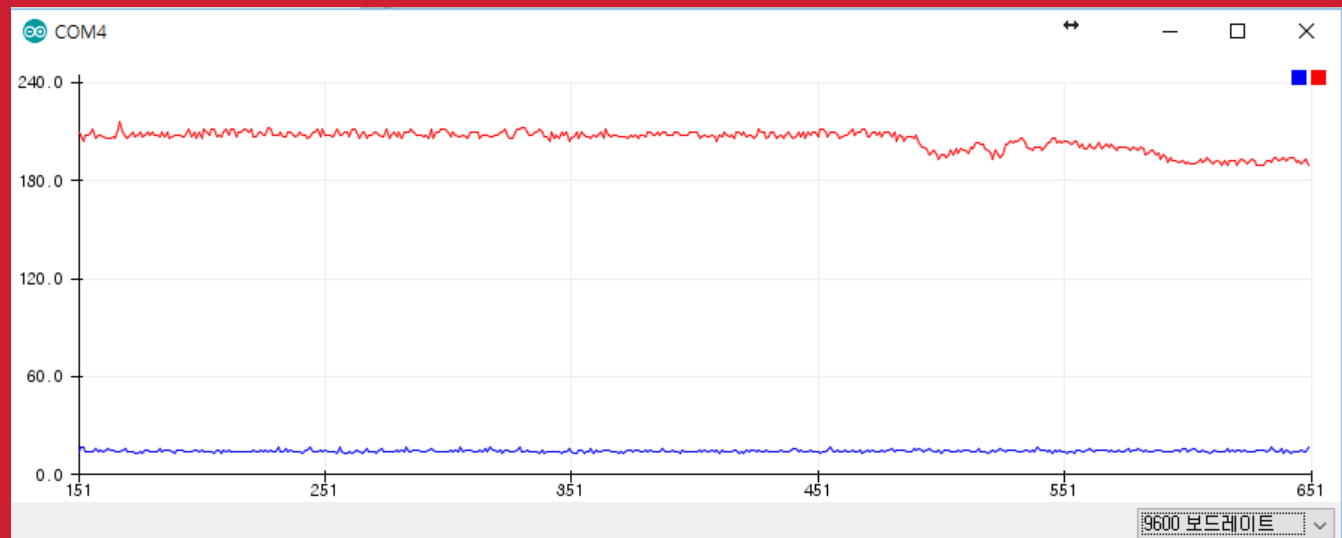
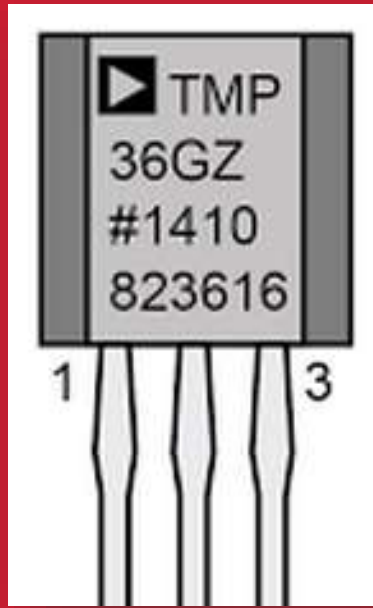
# Multiple sensors

# Arduino

# + Node.js

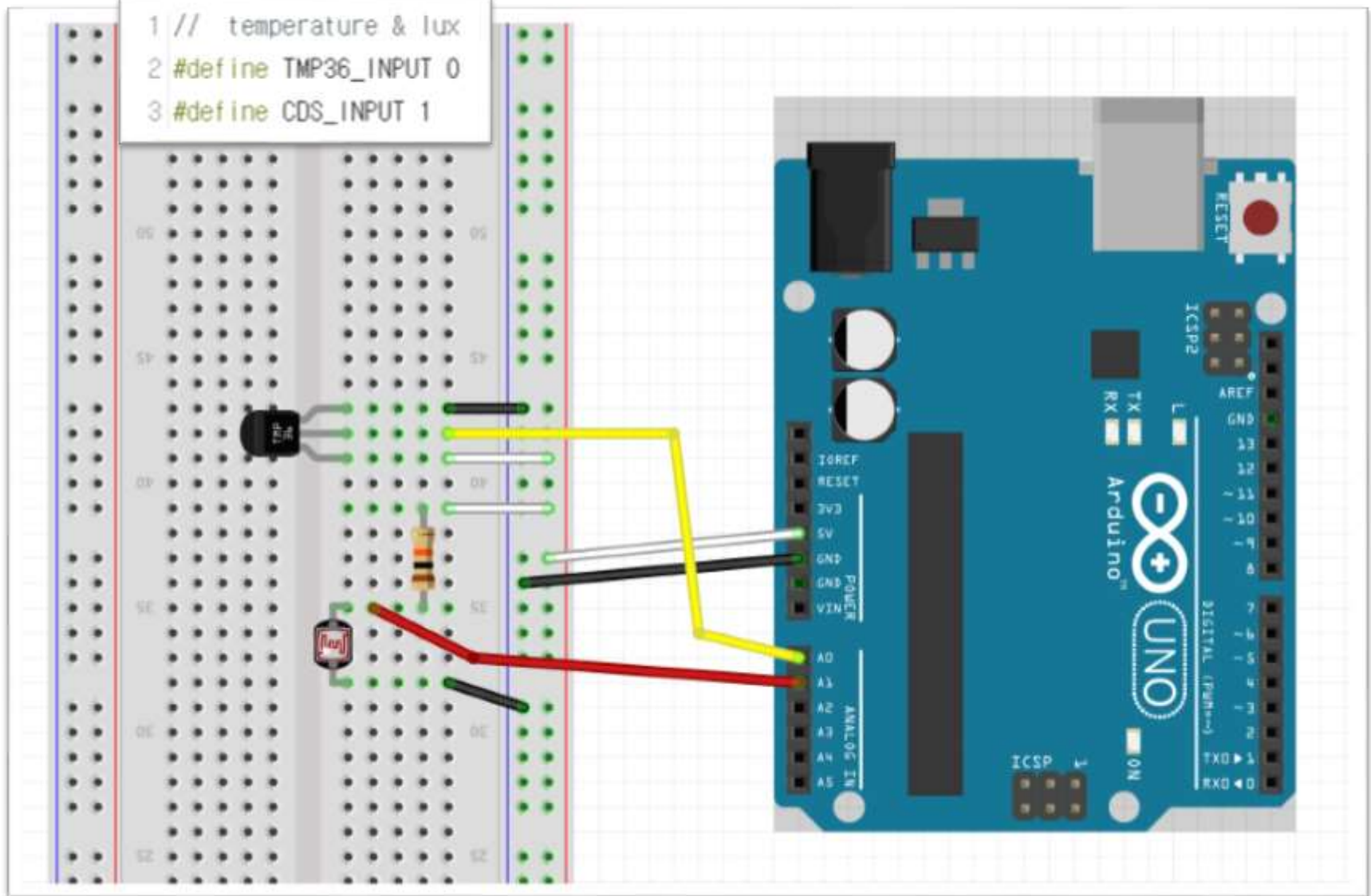


# Monitoring via Serial monitor & LCD



# A4.3.1 TMP36 + CdS : circuit

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







## A4.3.2 TMP36 + CdS : code

AAnn\_TMP36\_CdS\$

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

**AAnn\_tmp36\_cds.ino**

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

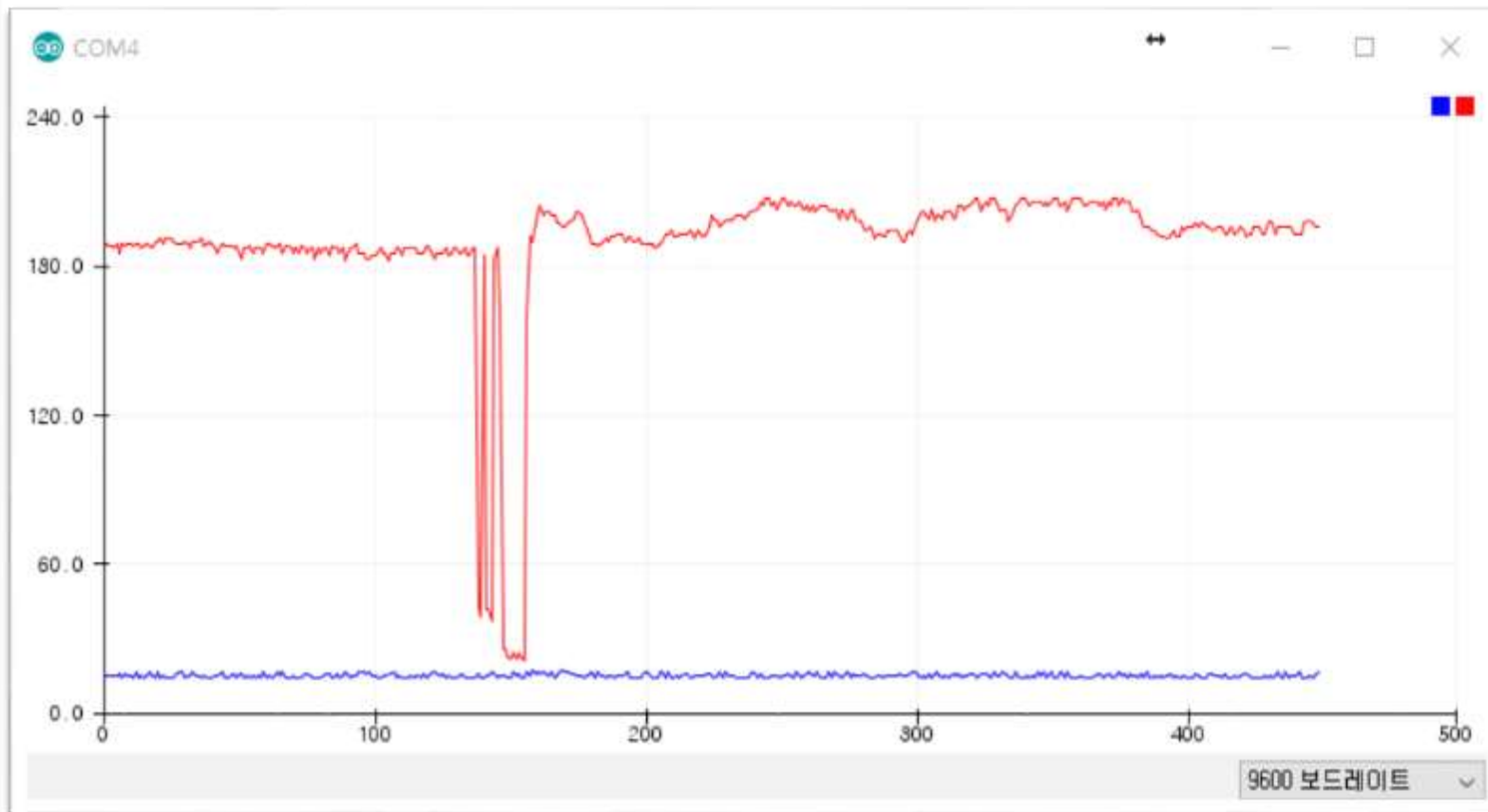




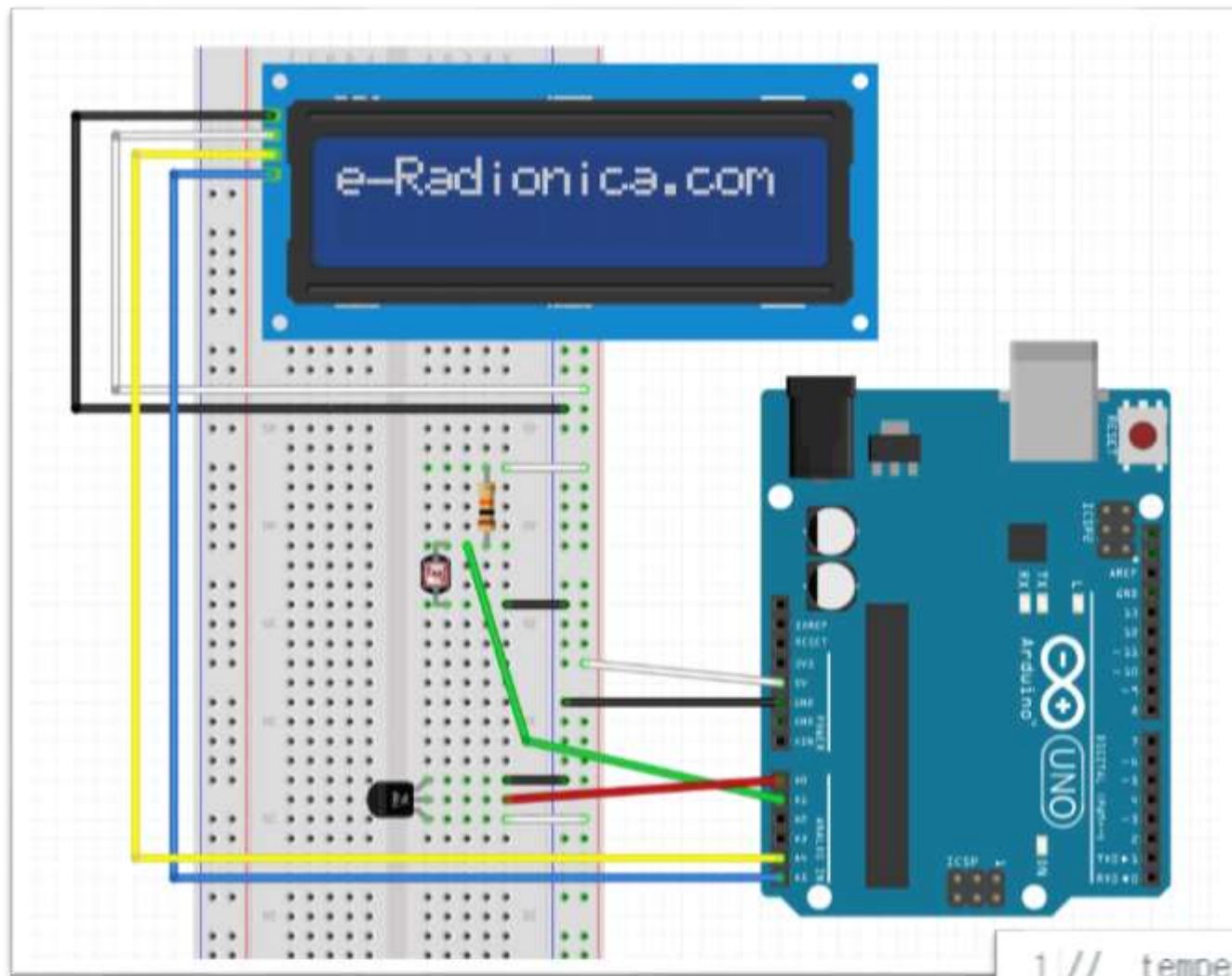
# A4.3.3 TMP36 + CdS : Monitoring

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



# A4.4.1 TMP36 + CdS + LCD : circuit



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



## A4.4.2 TMP36 + CdS + LCD : code-1

### AAnn\_tmp36\_cds\_lcd.ino

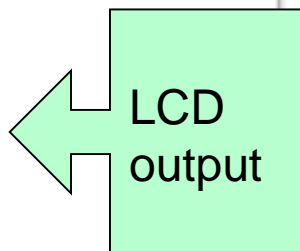
```
1 /*
2 온도, 빛 입력 LCD 모니터링 및 제어
3 */
4 // LCD 라이브러리 설정
5 #include <LiquidCrystal_I2C.h>
6 #include <Wire.h>
7 // LCD 설정
8 LiquidCrystal_I2C lcd(0x27,16,2); // 0x3F
9 // 0번 아날로그핀을 TMP36 온도 입력으로 설정한다.
10 // 1번 아날로그핀을 CdS 조도 입력으로 설정한다.
11 #define TMP36_INPUT 0 // A0
12 #define CDS_INPUT 1 // A1
13
```

```
14 void setup() {
15     Serial.begin(9600);
16     // 16X2 LCD 모듈 설정하고 백라이트를 켜다.
17     lcd.init();
18     lcd.backlight();
19     // 모든 메시지를 삭제한 뒤
20     // 숫자를 제외한 부분들을 미리 출력시킨다.
21     lcd.clear();
22     lcd.setCursor(0,0);
23     lcd.print("AA00,Temp: ");
24     lcd.setCursor(0,1);
25     lcd.print("Light: ");
26     lcd.setCursor(13,1);
27     lcd.print("lux"); //
28 }
```

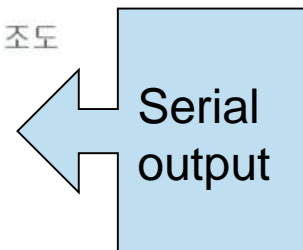


## A4.4.3 TMP36 + CdS + LCD : code-2

```
29 void loop(){
30   // Temperature from TMP36
31   int temp_value = analogRead(TMP36_INPUT);
32   // converting that reading to voltage
33   float voltage = temp_value * 5.0 * 1000; // in mV
34   voltage /= 1023.0;
35   float tempC = (voltage - 500) / 10 ;
36
37   // Lux from CdS (LDR)
38   int cds_value = analogRead(CDS_INPUT);
39   int lux = int(luminosity(cds_value));
40
41   // 전에 표시했던 내용을 지운다.
42   lcd.setCursor(12,0);
43   lcd.print(" ");
44   // 온도를 표시한다
45   lcd.setCursor(12,0);
46   lcd.print(tempC);
47   // 전에 표시했던 내용을 지운다
48   lcd.setCursor(9,1);
49   lcd.print(" ");
50   // 조도를 표시한다
51   lcd.setCursor(9,1);
52   lcd.print(lux);
```

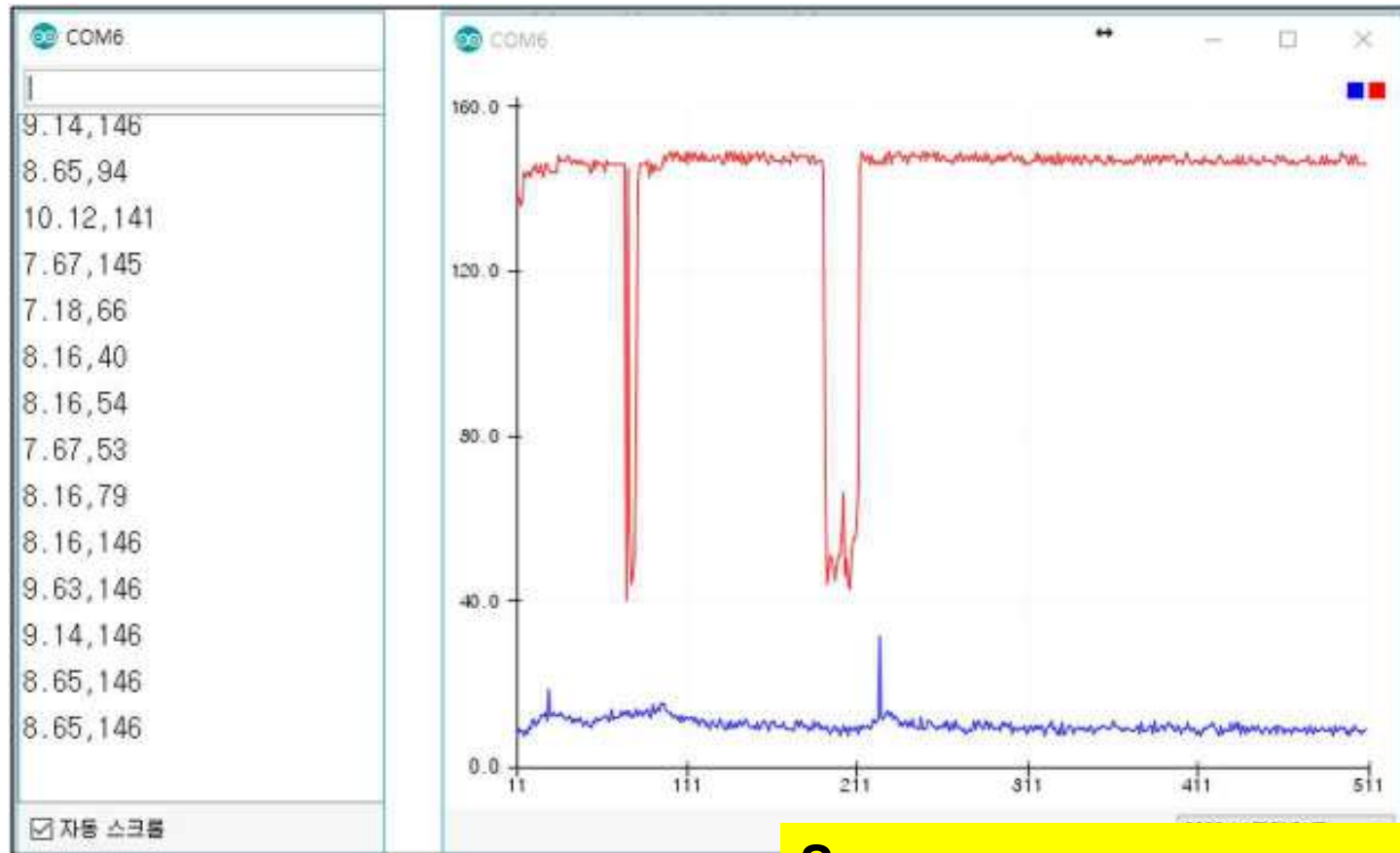


```
56   // Serial output --> 온도,조도
57   Serial.print(tempC);
58   Serial.print(",");
59   Serial.println(lux);
60   delay(1000);
61 }
62
63 //Voltage to Lux
64 double luminosity (int RawADC0){
65   double Vout=RawADC0*5.0/1023; // 5/1023 (Vin = 5 V)
66   double lux=(2500/Vout-500)/10;
67   // lux = 500 / Rldr,
68   // Vout = Ildr*Rldr = (5/(10 + Rldr))*Rldr
69   return lux;
70 }
```





## A4.4.4 TMP36 + CdS + LCD : result-1



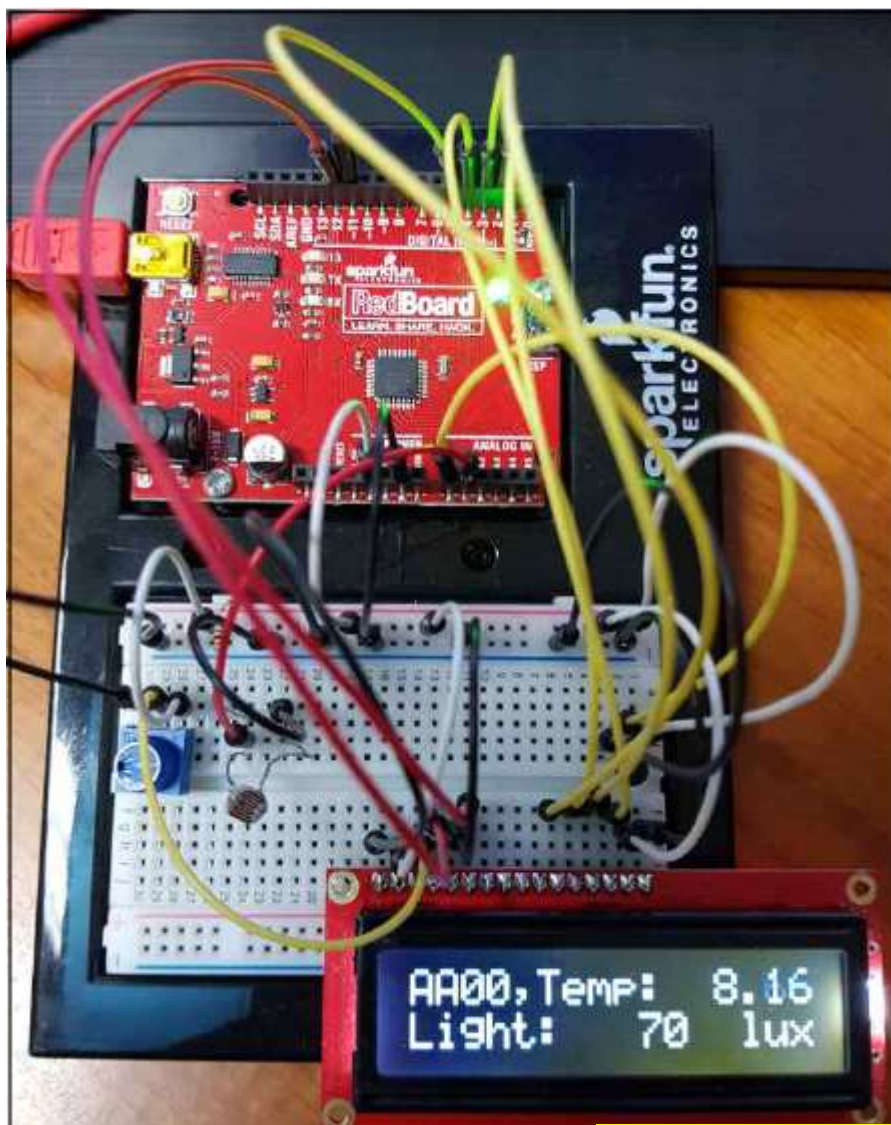
Save as

AAnn\_cds\_tmp36\_serial.png





## A4.4.5 TMP36 + CdS + LCD : result-2



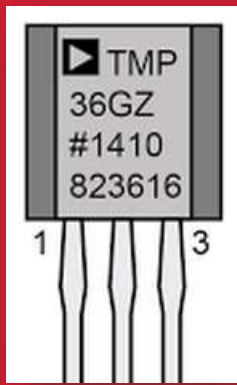
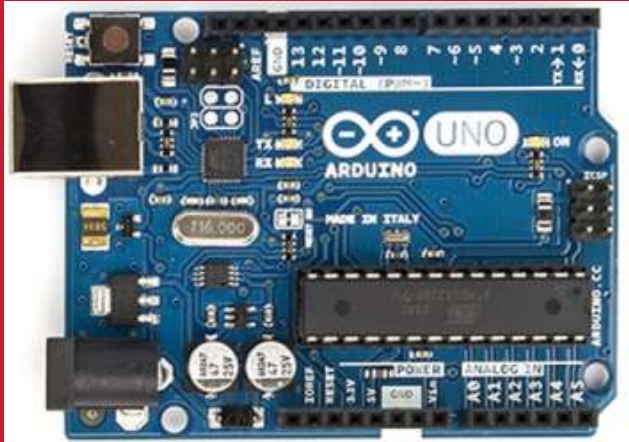
Save as  
[AAnn\\_cds\\_tmp36\\_lcd.png](#)



# 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**

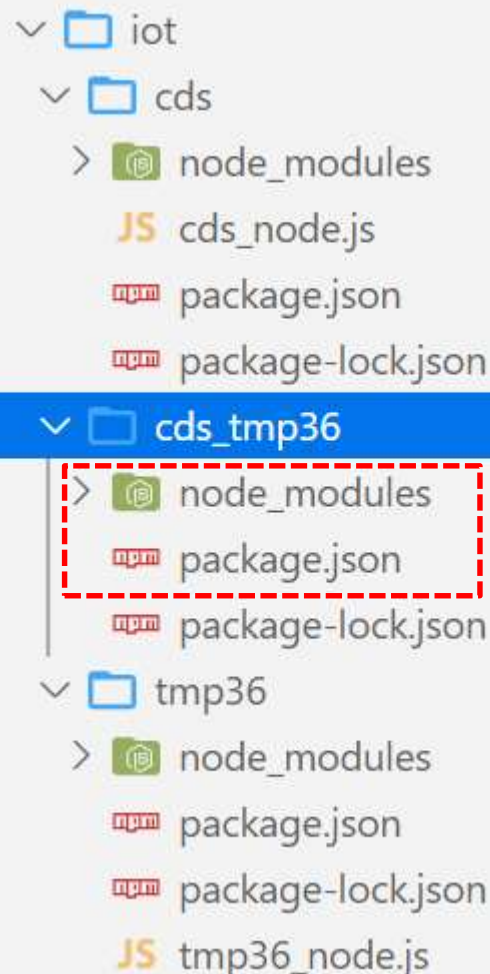
**author : hsn**



## A4.5.2 CdS + TMP36 + Node project

- **npm install --save serialport**
- **npm install --save socket.io@2.4.1**

```
"keywords": [  
  "cds",  
  "tmp36",  
  "node"  
],  
"author": "aa00",  
"license": "MIT",  
"dependencies": {  
  "serialport": "^9.2.4",  
  "socket.io": "^2.4.1"  
}
```





## 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

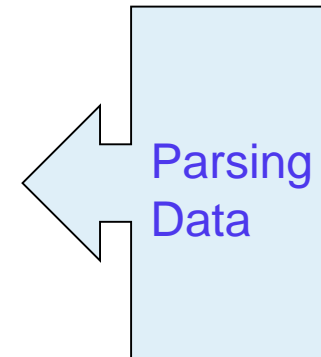
```
1  // cds_tmp36_node.js
2
3  var serialport = require("serialport");
4  var portName = "COM3"; // check your COM port!!
5  var port = process.env.PORT || 3000;
6
7  var io = require("socket.io").listen(port);
8
9  const Readline = require("@serialport/parser-readline");
10 // serial port object
11 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 });
```

## 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.toString());
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

```
D:\aann\aann-rpt06\iot\cds_tmp36>node cds_tmp36_node
serial port open
AA00,2021-10-05 13:57:38.119,25.27,84
AA00,2021-10-05 13:57:39.119,25.27,84
AA00,2021-10-05 13:57:40.122,24.78,83
AA00,2021-10-05 13:57:41.125,24.78,84
AA00,2021-10-05 13:57:42.125,24.78,84
AA00,2021-10-05 13:57:43.129,25.27,84
AA00,2021-10-05 13:57:44.132,25.27,83
AA00,2021-10-05 13:57:45.132,25.76,83
AA00,2021-10-05 13:57:46.135,24.78,84
```

IOT data format

시간, 온도, 조도

Save as

AAnn\_cds\_tmp36\_IOT.png





## A4.5.6 CdS + TMP36 + Node project : WEB

[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

---

Save as

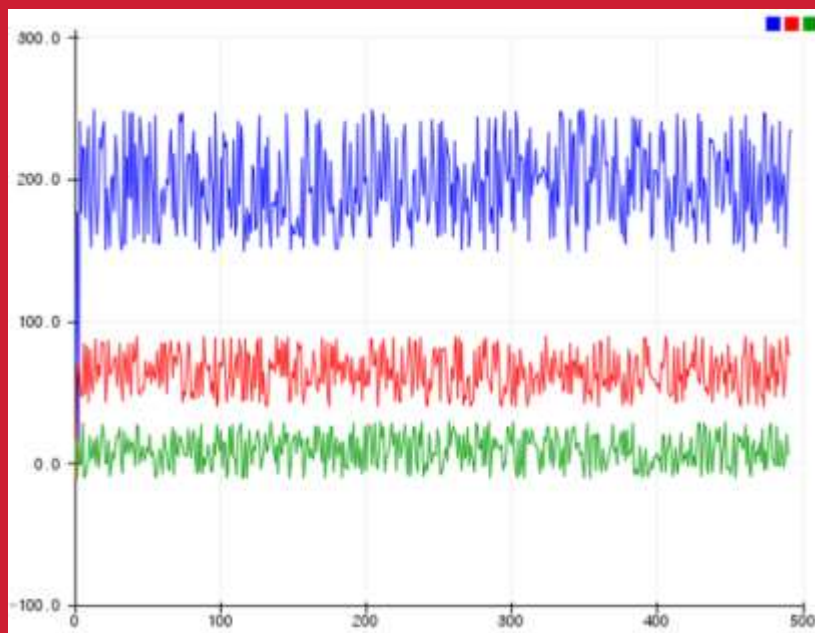
[AAnn\\_cds\\_tmp36\\_WEB.png](#)

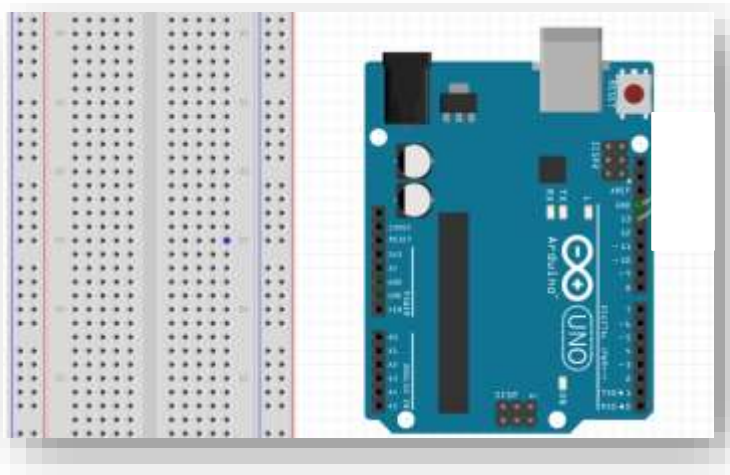


# [DIY] Multi-signals

다중신호 시뮬레이션

+ node.js





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

전자 소자 연결 없이 마구잡이 수 생성 함수를 이용해서 조도, 습도, 온도에 해당하는 **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 담은 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

```
1 /*
2   Multi Signals
3   Simulation of multiple random signals
4 */
5 // signals
6 int humi=0;
7 int temp=0;
8 int lux=0;
9
```

```
10 // the setup routine runs once when you press reset:
11 void setup() {
12   // initialize serial communication at 9600 bits per second:
13   Serial.begin(9600);
14 }
15
16 // the loop routine runs over and over again forever:
17 void loop() {
18   // Multi signals
19   humi = random(40,90);
20   temp = random(-10, 30);
21   lux = random(150,250);
22   Serial.print("AA00, Ambient lux: ");
23   Serial.print(lux);
24   Serial.print(" , Humidity: ");
25   Serial.print(humi);
26   Serial.print(" , Temperature: ");
27   Serial.println(temp);
28   delay(500);      // delay in between reads for stability
29 }
```

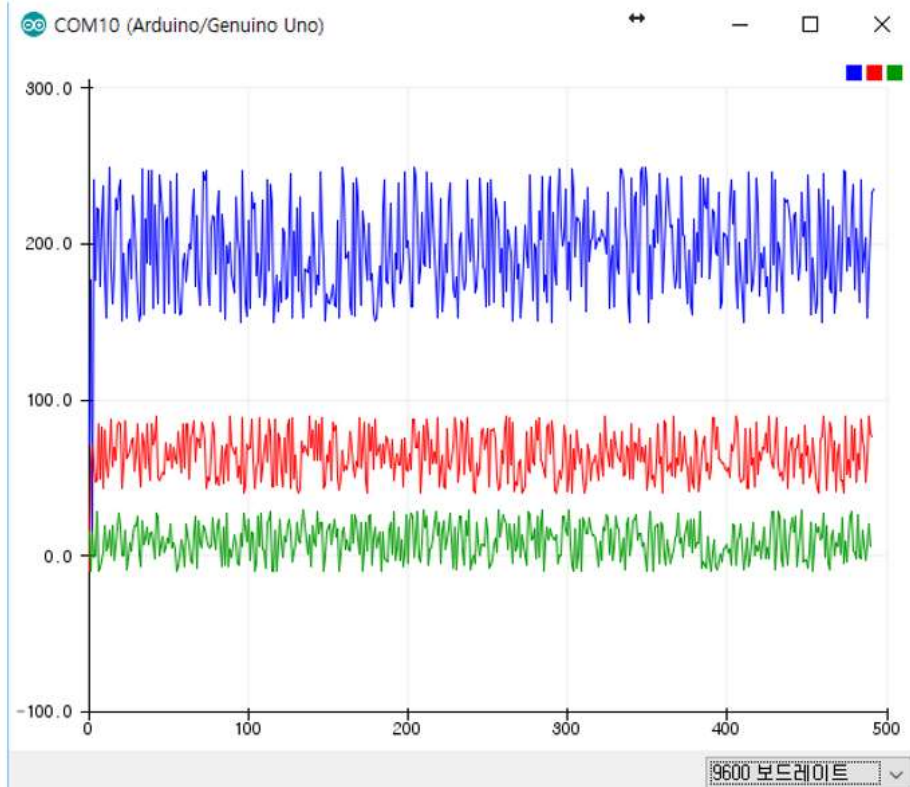


# DIY - result

## DIY 결과

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

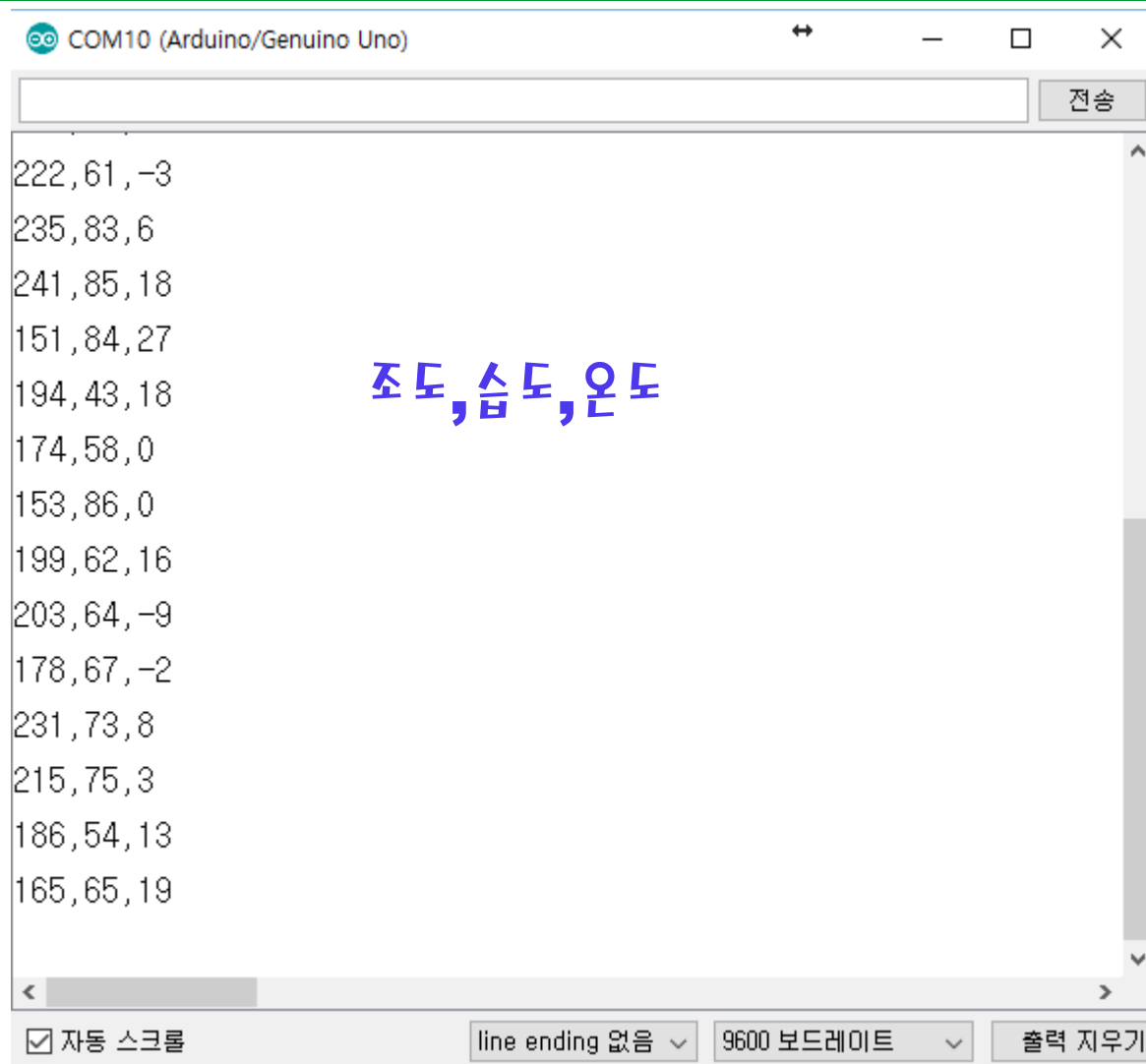
```
COM10 (Arduino/Genuino Uno)
| 전송
AA00, Ambient lux: 186 , Humidity: 54 , Temperature: 13
AA00, Ambient lux: 165 , Humidity: 65 , Temperature: 19
AA00, Ambient lux: 151 , Humidity: 84 , Temperature: 19
AA00, Ambient lux: 155 , Humidity: 57 , Temperature: 25
AA00, Ambient lux: 248 , Humidity: 44 , Temperature: 1
AA00, Ambient lux: 155 , Humidity: 78 , Temperature: -7
AA00, Ambient lux: 216 , Humidity: 72 , Temperature: 22
AA00, Ambient lux: 188 , Humidity: 56 , Temperature: 7
AA00, Ambient lux: 247 , Humidity: 84 , Temperature: 11
AA00, Ambient lux: 187 , Humidity: 61 , Temperature: 18
AA00, Ambient lux: 247 , Humidity: 48 , Temperature: 7
AA00, Ambient lux: 159 , Humidity: 84 , Temperature: 14
AA00, Ambient lux: 225 , Humidity: 71 , Temperature: 15
AA00, Ambient lux: 192 , Humidity: 75 , Tempera
< >
[x] 자동 스크롤 line ending 없음 9600 보드레이트 출력 지우기
```





# 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 센서 신호 시뮬레이션 → **조도, 습도, 온도**를 **Node.js**로 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**

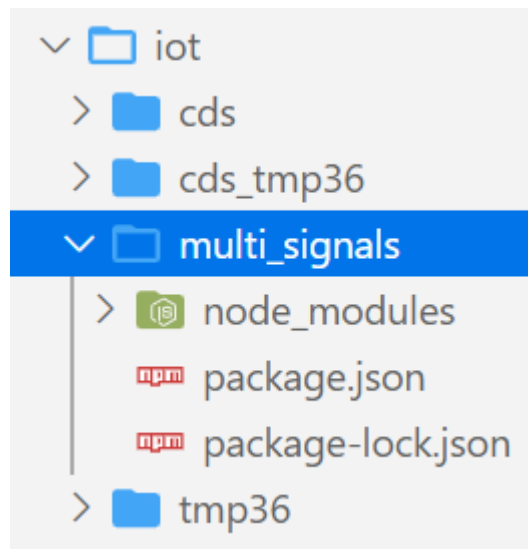
**description : multi-signals-node project**

**entry point : aann\_multi\_signals.js**

**author : aann**

### 3. Install node modules

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







## DIY – New result 2-2

DIY 결과 [2]: 가상적인 세 개의 센서신호 시뮬레이션 → **조도, 습도, 온도**를 **Node.js**로 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 센서 신호 시뮬레이션 → **조도, 습도, 온도**를 **Node.js**로 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.js**로 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, 온도**를 **Node.js**로 처리

Hint:

javascript function : **indexOf()**

[https://www.w3schools.com/jsref/jsref\\_indexof.asp](https://www.w3schools.com/jsref/jsref_indexof.asp)

Syntax

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

Parameter Values

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

javascript function : **substring()**

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

Parameter Values

Parameter	Description
<i>start</i>	Required. The position where to start the extraction. First character is at index 0
<i>end</i>	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, 온도**를 **Node.js**로 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**

Save this result as  
**AAnn\_multi\_signals\_node .png**



# A4.5.6 multi-signals + Node project : WEB

[Web monitoring] [client\\_multi\\_signals.html](#)



## IoT Signal from Arduino

### Real-time Signals

---

on Time: 2021-10-05 14:27:23.536

Signals (조도, 습도, 온도) : 161,41,22

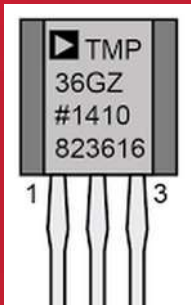
---

Save as

[AAnn\\_multi\\_signals\\_WEB.png](#)

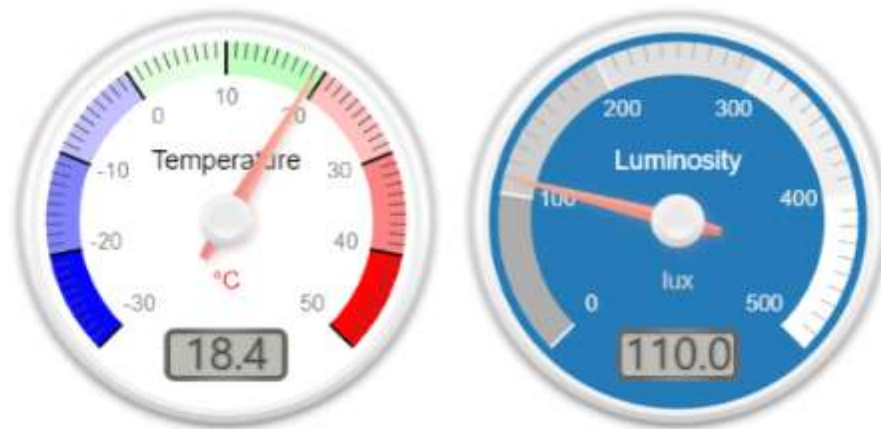


# Data visualization using **ply.ly**

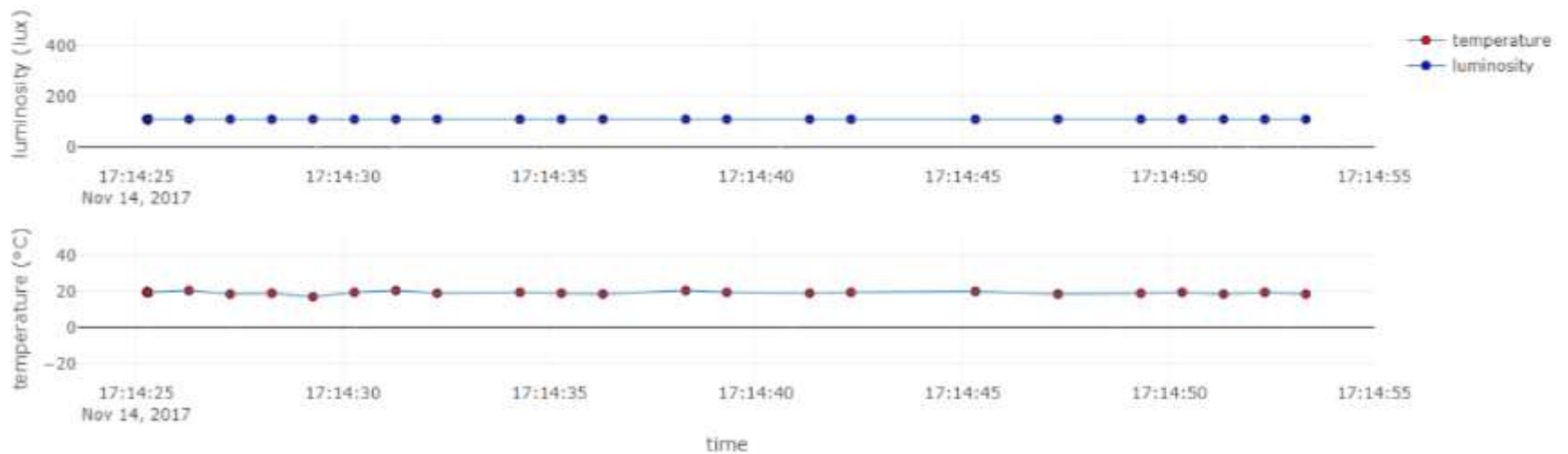


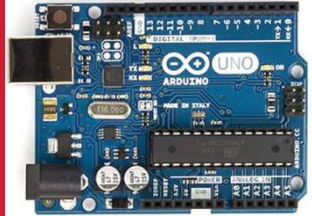


## Real-time Temperature( $^{\circ}\text{C}$ ) and Luminosity(lux) from sensors



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





# [Practice]

## ◆ [wk06]

- **Arduino sensors + Node.js**
- **Complete your project**
- **Upload folder: aann-rpt06**
- **Use repo “aann” in github**

# wk06 : Practice : aann-rpt06

## ◆ [Target of this week]

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

제출폴더명 : **aann-rpt06**

### - 압축할 파일들

- ① **AAnn\_tmp36\_message.png**
- ② **AAnn\_tmp36\_IOT\_data.png**
- ③ **AAnn\_cds\_tmp36\_serial.png**
- ④ **AAnn\_cds\_tmp36\_lcd.png**
- ⑤ **AAnn\_cds\_tmp36\_IOT.png**
- ⑥ **AAnn\_cds\_tmp36\_WEB.png**
- ⑦ **AAnn\_multi\_signals\_node.png**
- ⑧ **AAnn\_multi\_signals\_WEB.png**
- ⑨ **All \*.ino**
- ⑩ **All \*.js**
- ⑪ **NO node\_modules folder**

## ● 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

## Real-time Weather Station from nano 33 BLE sensors



on Time: 2020-09-09 10:27:17.321

