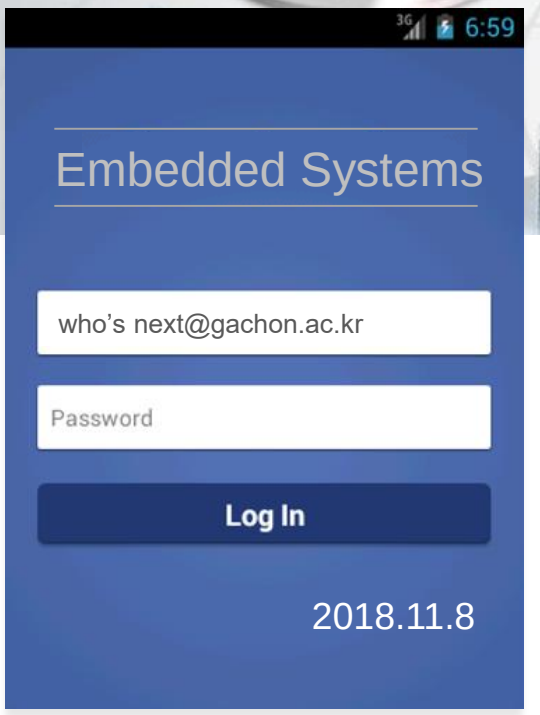


임베디드 시스템

컴퓨터공학과 이병문

강의일정(후반부)

09	**Node**기반 웹프레임워크 기술
10	웹기반 임베디드시스템 제어**1**
11	웹기반 임베디드시스템 제어**2**
12	웹기반 임베디드시스템 제어**3**
13	**REST API**기반 임베디드 연동 **1**
14	**REST API**기반 임베디드 연동 **2**
15	임베디드 무선연동기술
16	기말고사

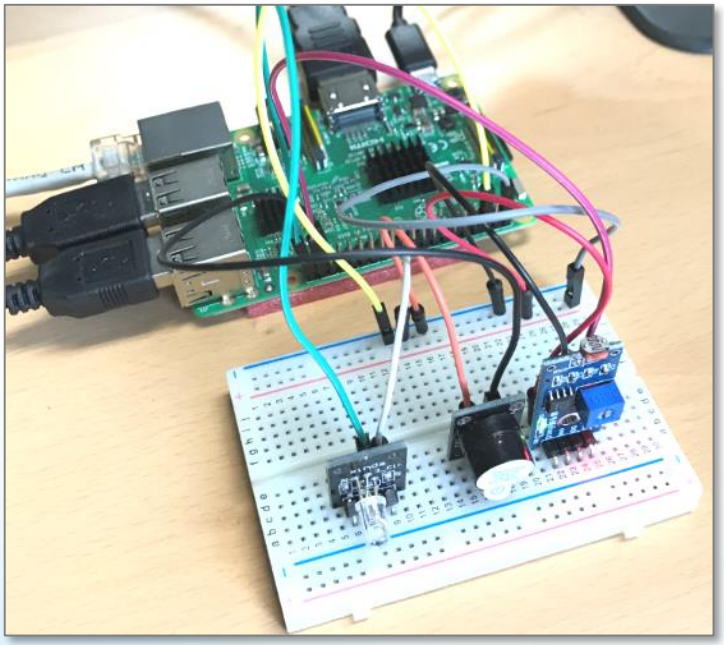
- **웹기반 제어 환경설정**
- **웹기반 임베디드 제어**
- **아날로그 사운드센서모듈**

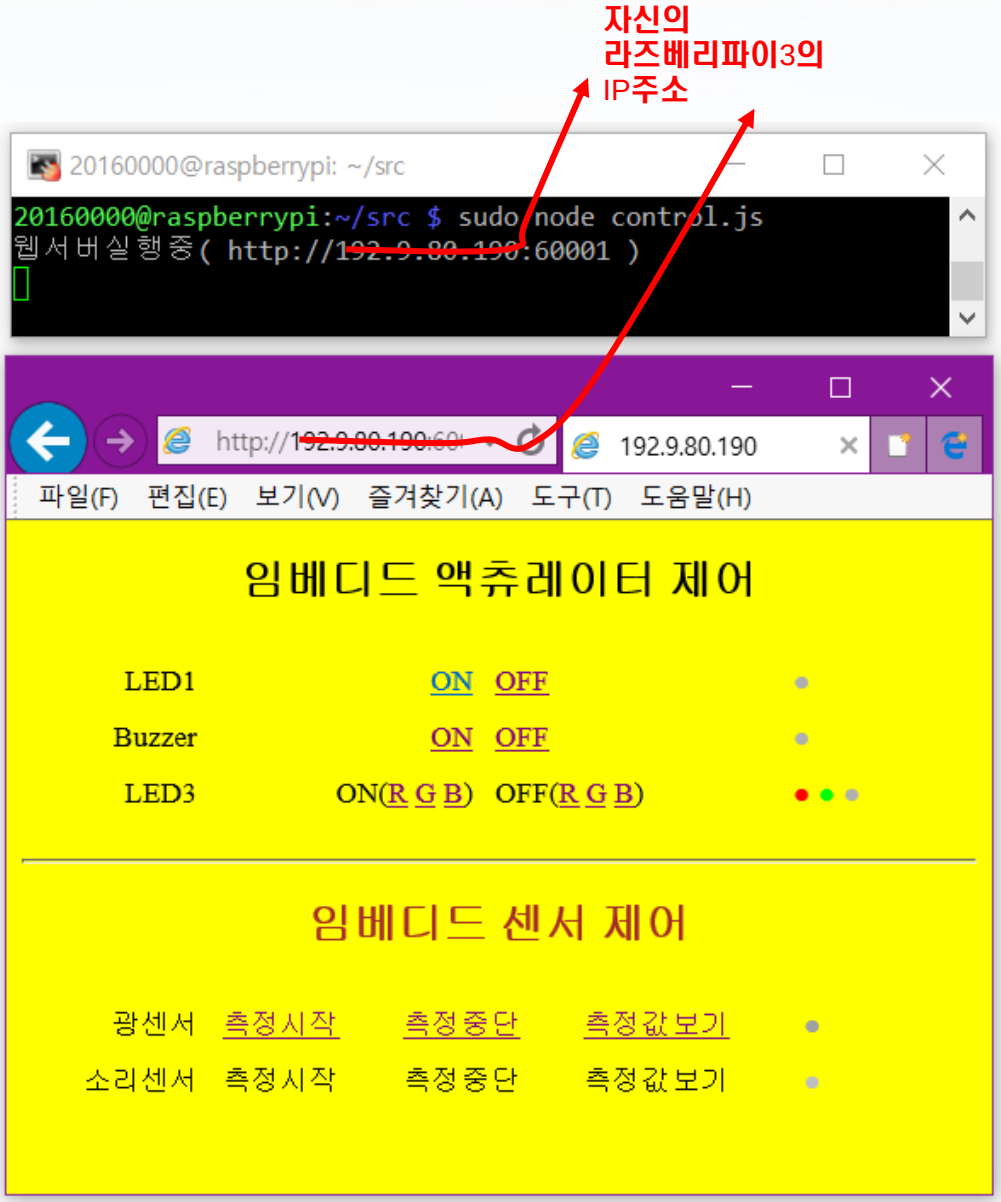
웹기반 임베디드시스템 제어

■ 웹제어 구성

☑ 2개의 액츄레이터모듈(LED, 부져)과 1개의 센서모듈(광센서)을 활용

☑ 웹(node.js, express, ejs, wiring-pi)를 활용한 제어



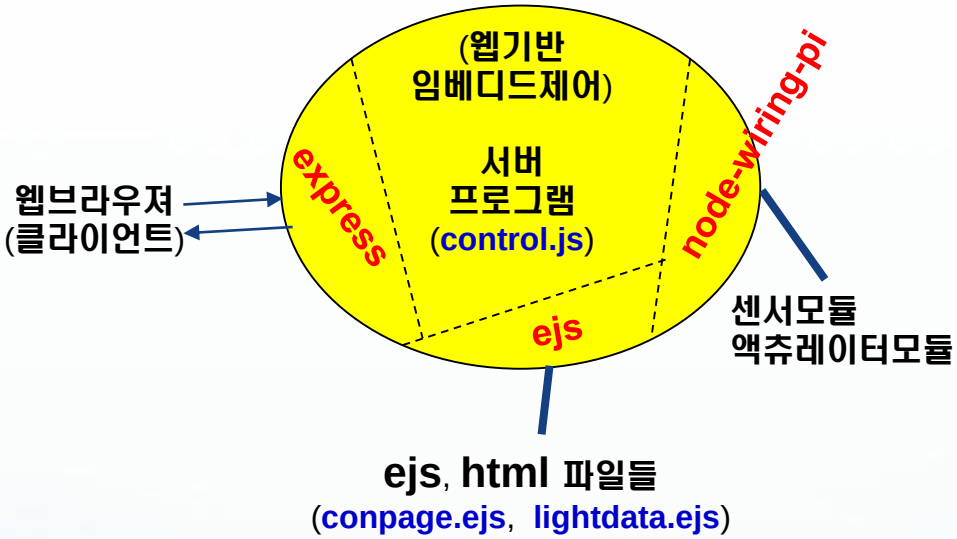


■ 웹 개발환경

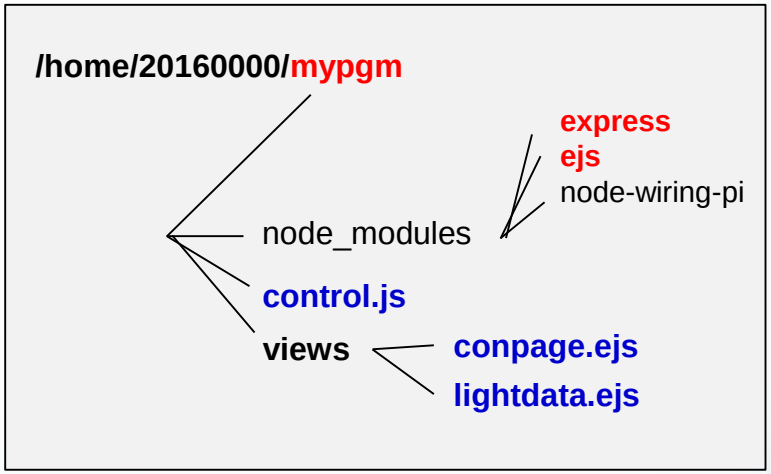
☑ 웹에서 node.js (추가적으로 필요한) 외부모듈설치 (express, ejs)를 활용한 제어

```
$ cd  mypgm
$ npm  install  express
```

웹프레임워크 외부모듈 설치

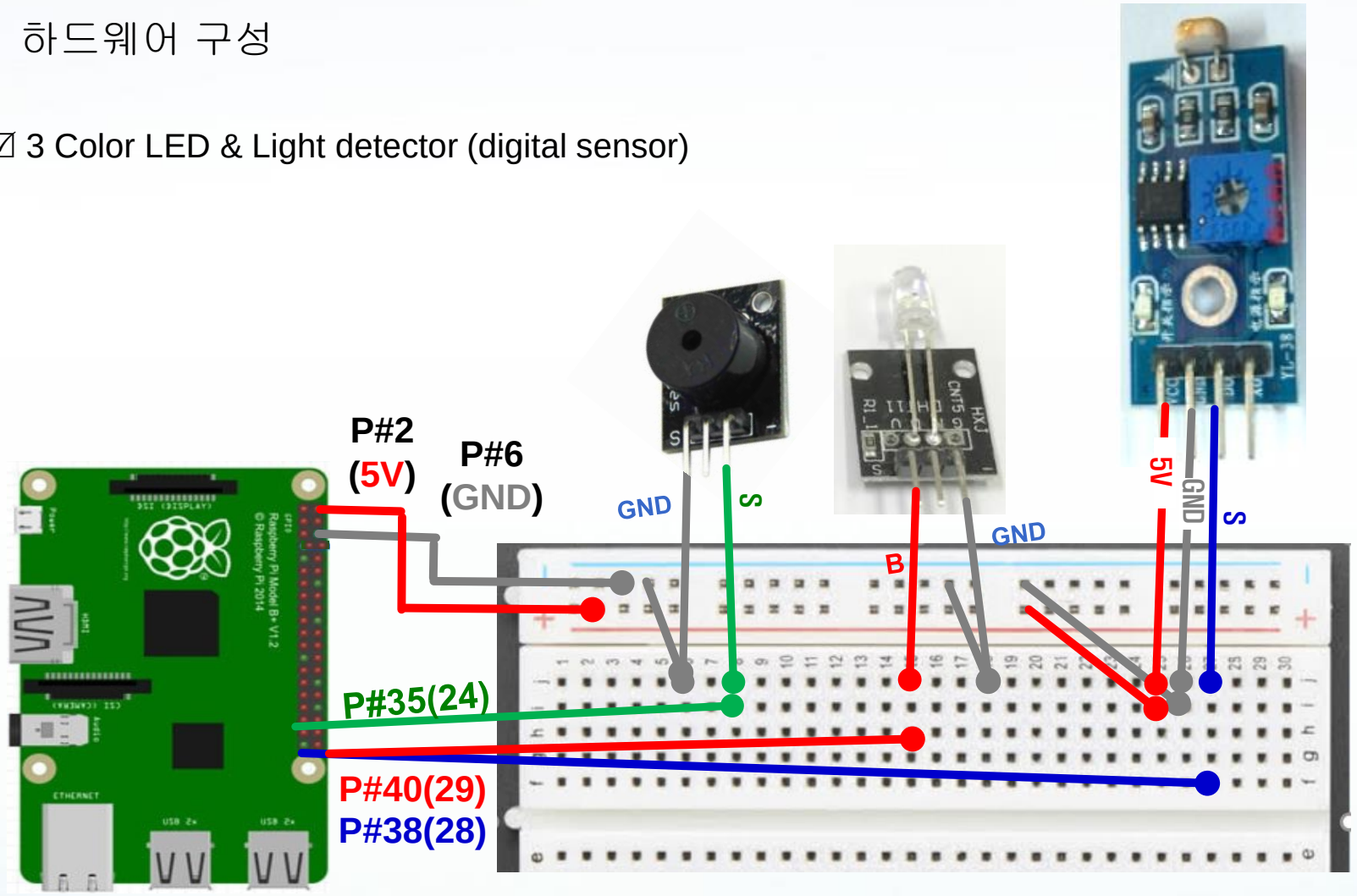


디렉터리 구조



■ 하드웨어 구성

☑ 3 Color LED & Light detector (digital sensor)



웹기반 임베디드시스템 제어

■ 프로그램 코드

☑ 웹에서 LED와 부져를 ON/OFF 시키고, 광센서 측정시작/중지/측정값 조회하는 control.js

예제1 (control.js, views/contpage.ejs views/lightdata.ejs)

```
const   gpio = require('node-wiring-pi'), ejs = require('ejs');
const   express = require('express'),  fs = require('fs');
const   app = express();

const   BLUELED = 29,  BUZZER = 24,  LIGHT = 28;
var   timerid,   index,  value=[ ];   // value 배열에 데이터저장
var   led1state = '#b0b0b0' ;       // OFF상태색(회색)
var   buzzerstate = '#b0b0b0';
var   lightstate = '#b0b0b0';

app.get('/', (req, res) => {
       fs.readFile('views/contpage.ejs','utf8',(error, data) => {
        if (error)
          res.send(500);
        else
           res.send(ejs.render(data, {
                 led1color:led1state,
                 buzzercolor:buzzerstate,
                 lightcolor:lightstate }));
      });
});

app.listen(60001, () => {
       gpio.wiringPiSetup();
       gpio.pinMode(BLUELED, gpio.OUTPUT);
       gpio.pinMode(BUZZER, gpio.OUTPUT);
       gpio.pinMode(LIGHT, gpio.INPUT);
       console.log("웹서버실행중( http://192.9.80.190:60001 ) ");
});
```

자신의 라즈베리파이3의 IP주소

views/contpage.ejs

```
app.get('/led1/1', (req, res) => {
       console.log("LED를 ON시킵니다");
       gpio.digitalWrite(BLUELED, 1);
       led1state = '#0000ff';   // 웹 컬러셋팅(파랑)
       res.redirect('/');
});

app.get('/led1/0', (req, res) => {
       console.log("LED를 OFF합니다.");
       gpio.digitalWrite(BLUELED, 0);
       led1state = '#b0b0b0'; // 웹 컬러셋팅(회색)
       res.redirect('/');
});

app.get('/buzzer/1', (req, res) => {
       console.log("부져를 켭니다");
       gpio.digitalWrite(BUZZER, 1);
       buzzerstate = '#0000ff';
       res.redirect('/');
});

app.get('/buzzer/0', (req, res) => {
       console.log("부져를 끕니다");
       gpio.digitalWrite(BUZZER, 0);
       buzzerstate = '#b0b0b0';
       res.redirect('/');
});
```

■ 프로그램 코드

```
const   lightctl = ( ) => {
        if (index < 500) {
             value[index++] = gpio.digitalRead(LIGHT);
             console.log('value:' + value[index-1]);
        }
        else
            index = 0;
        timerid = setTimeout(lightctl, 1000);
}
app.get('/light/1', (req, res) => {   // 조도센서 측정시작(활성화)
        console.log(" 광센서로 측정을 시작합니다...");
        timerid = setTimeout(lightctl, 100);
        lightstate = '#0000ff';
        res.redirect('/');
});
app.get('/light/2', (req, res) => {  // 조도센서 측정값 조회
        console.log(" 그동안 측정된 값들을 보여줍니다.");
        fs.readFile('views/lightdata.ejs','utf8',(error, data) => {
         if (error)     res.send(500);
         else   res.send(ejs.render(data, { lightdata: value }));
        });
});
app.get('/light/0', (req, res) => {  // 조도센서 측정종료(비활성화)
        console.log(" 광센서의 측정을 중지하였습니다...");
        clearTimeout(timerid);
        lightstate = '#a0a0a0';
        res.redirect('/');
});
```

예제1 (control.js, views/contpage.ejs
views/lightdata.ejs)

```
<!DOCTYPE html>
<html>

   <head>
      <title>광센서 측정결과</title>
   </head>
   <body>
      <h2> 광센서 측정값 조회 </h2>
       <p>측정시작부터 측정된 값을 보여줍니다.
            (0은 밝음, 1은 어두움) </p>
      <hr>
       <table>
       <tr>
        <% lightdata.forEach( function(element, i) { %>
        <td color=red> <%= element %> </td>
        <%  }); %>
        </tr> </table>
   </body>
</html>
```

■ 프로그램 코드

예제1 (control.js, **views/contpage.ejs**, views/lightdata.ejs)

```
<!doctype html>
<html>
 <head>
 </head>
 <body bgcolor=yellow>
   <h2 align=center> 임베디드 액츄레이터 제어 </h2>
   <p>  <table border=0 width=500 cellspacing=10>
     <tr>
       <td align=right> LED1 </td>
       <td align=right> <a href="led1/1"> ON </a></td>
       <td> <a href="led1/0"> OFF </a></td>
       <td> <font color= <%= led1color%> > ● </font></td>
     </tr>
     <tr>
       <td align=right> Buzzer </td>
       <td align=right> <a href="buzzer/1"> ON </a></td>
       <td> <a href="buzzer/0"> OFF </a></td>
       <td> <font color= <%= buzzercolor %> > ● </font></td>
     </tr>
     <tr>
       <td align=right> LED3 </td>
       <td align=right> ON(<a href="led3/11">R</a>
                        <a href="led3/10">G</a>
                        <a href="led3/21">B</a>)</td>
       <td> OFF(<a href="led3/20">R</a>
              <a href="led3/31">G</a>
              <a href="led3/30">B</a>)  </td>
       <td> <font color=#ff0000>●</font>
            <font color=#00ff00>●</font>
            <font color=#b0b0b0>●</font> </td>
     </tr>
   </table>
   </p>
<hr>
   <h2 align=center>
       <font color=brown> 임베디드 센서 제어 </font>
   </h2>
   <p>  <table border=0 width=450 cellspacing=10>
     <tr>
       <td align=right> 광센서 </td>
       <td> <a href="light/1"> 측정시작 </a></td>
       <td> <a href="light/0"> 측정중단 </a></td>
       <td> <a href="light/2" target=_new> 측정값보기 </a></td>
       <td> <font color= <%= lightcolor %> > ● </font></td>
     </tr>
     <tr>
       <td align=right> 소리센서  </td>
       <td> 측정시작 </td>
       <td> 측정중단 </td>
       <td> 측정값보기 </td>
       <td> <font color=#bfbfbf> ● </font></td>
     </tr>
   </table>
   </p>
 </body>
</html>
```

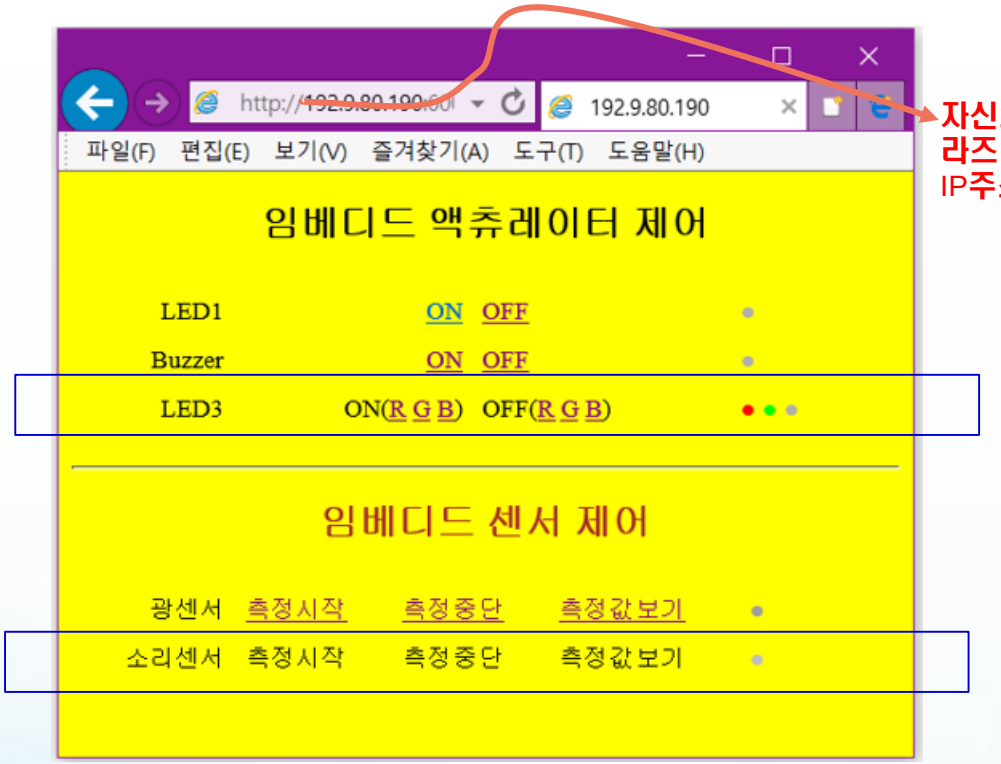
bmlee made

■ 예제2)

☑ 예제1에서 미완성된 부분(3색 LED, 사운드센서)를 완성하여라.

단, 소스코드를 복사하여 control2.js 와 contpage2.ejs, 그리고 soundata.ejs 를 작성하여라

즉, (control.js -> control2.js, contpage.ejs -> contpage2.ejs)



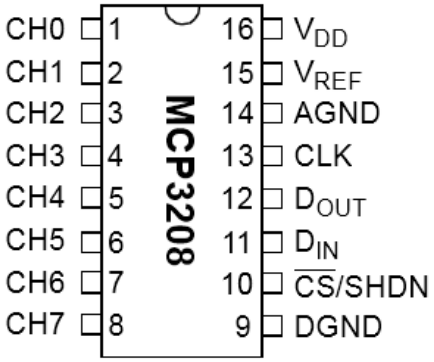
자신의
라즈베리파이3의
IP주소와 포트주소

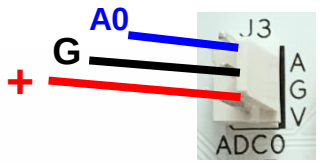
동일 네트워크에 있다면
스마트폰에서도 접속해보세요~

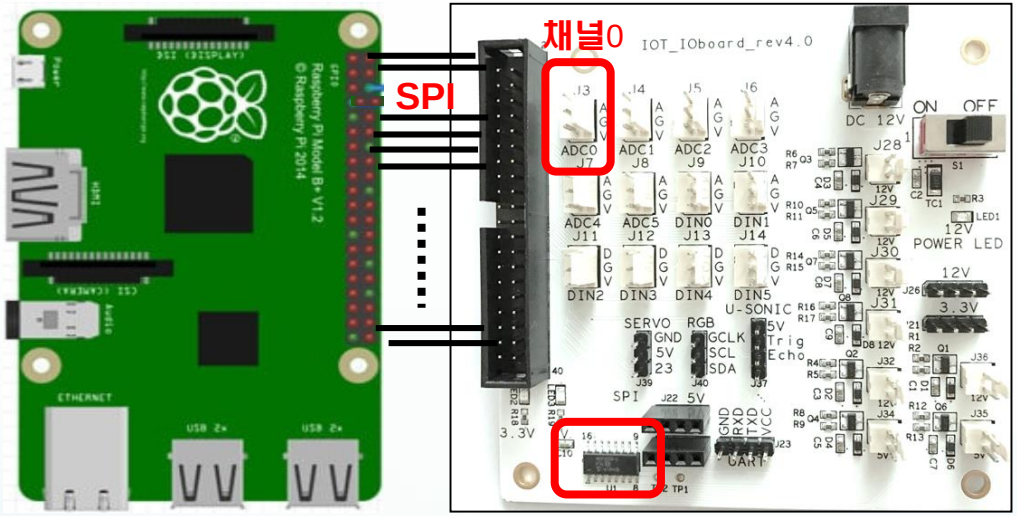
■ 아날로그 센서모듈 제어

☑ 아날로그 사운드(소리)센서모듈

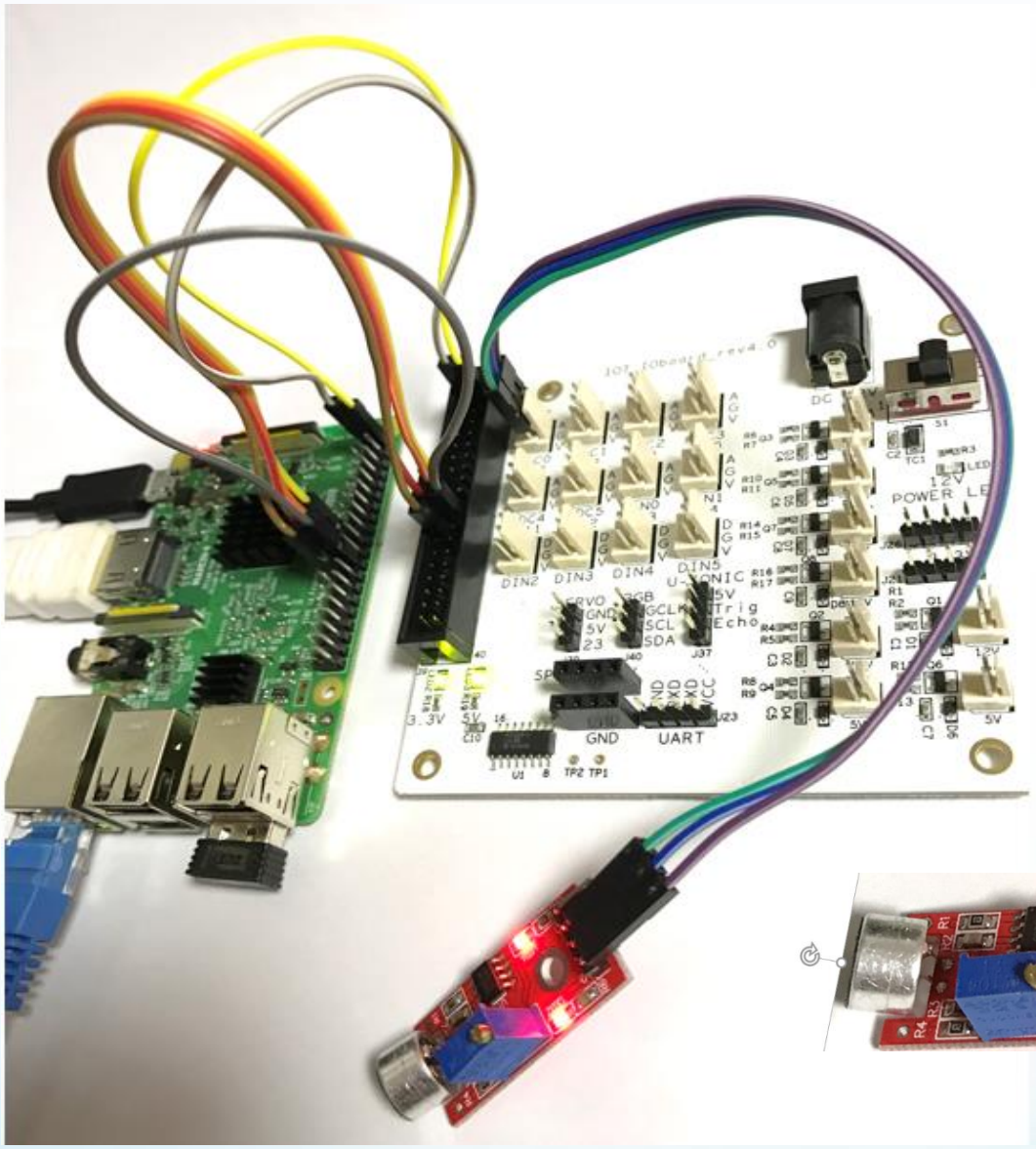
ADC (Analog to Digical Converter)



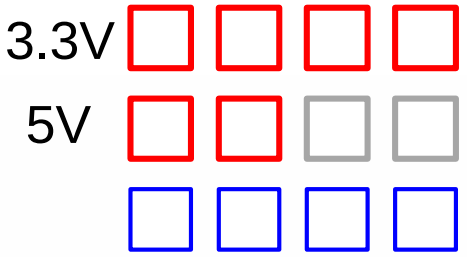




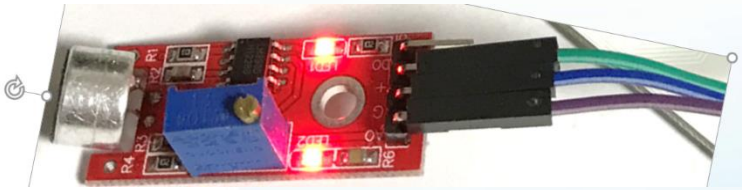
■ 하드웨어 구성



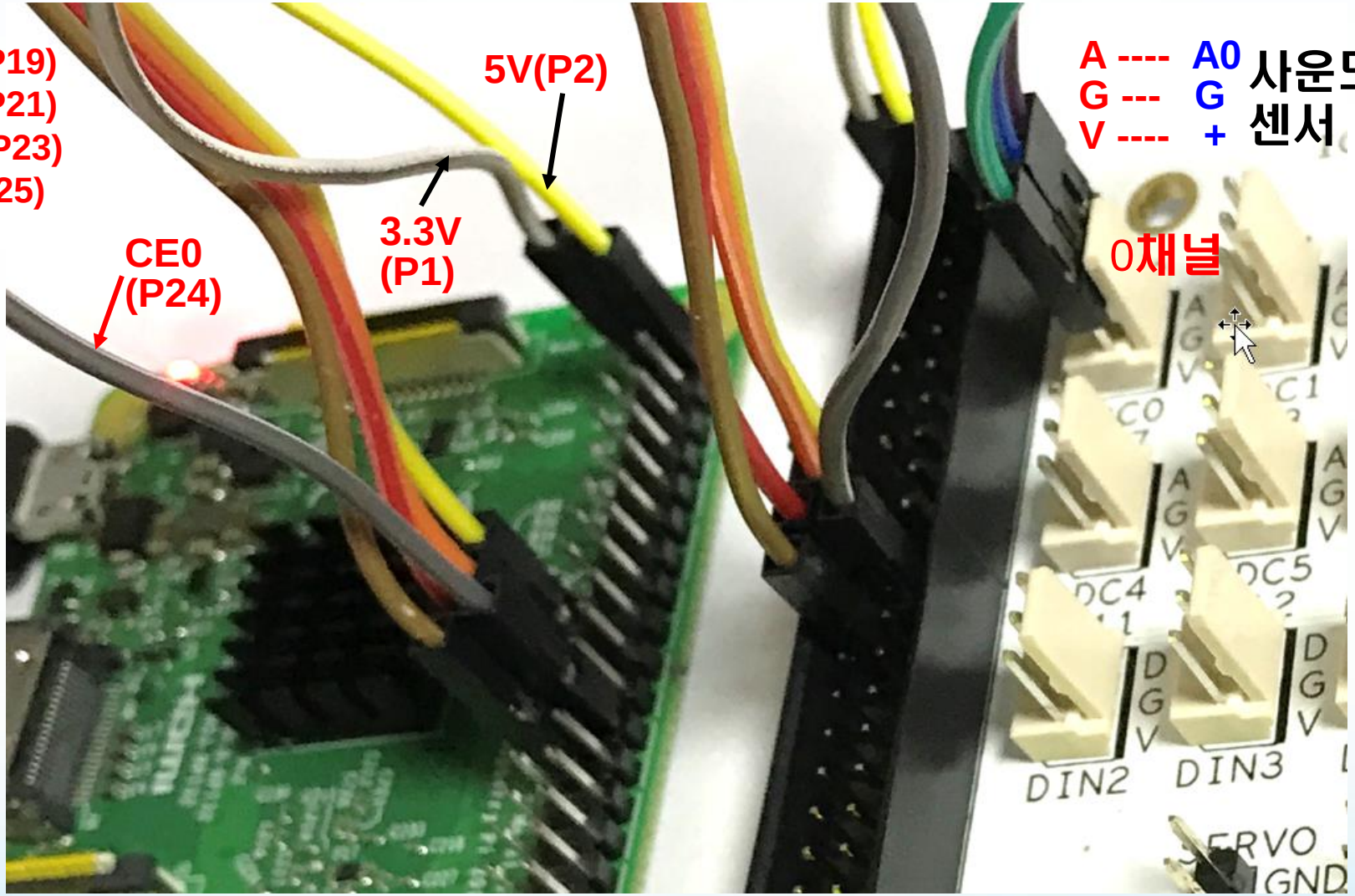
상단의 4개채널은 3.3V전원
하단의 4개채널은 5V전원



MCP3208은 8채널을 지원하나
I/O보드에서는 5채널만 연결되어 있음

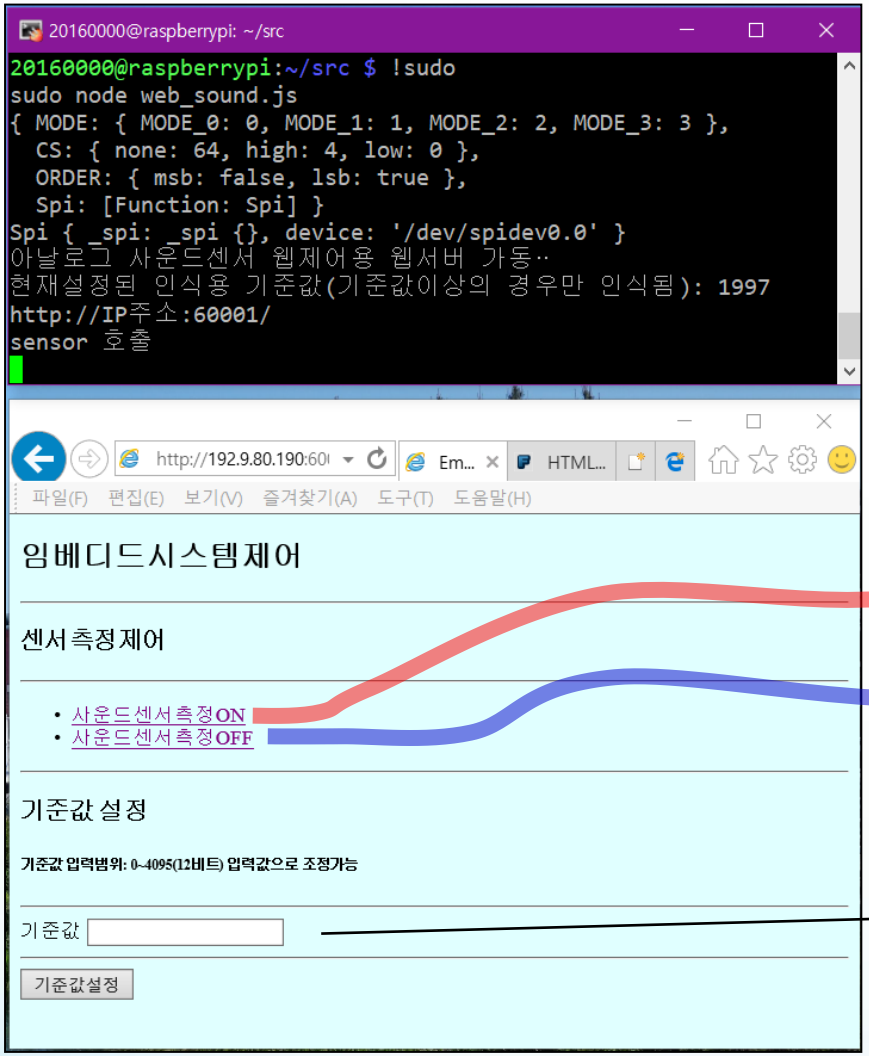


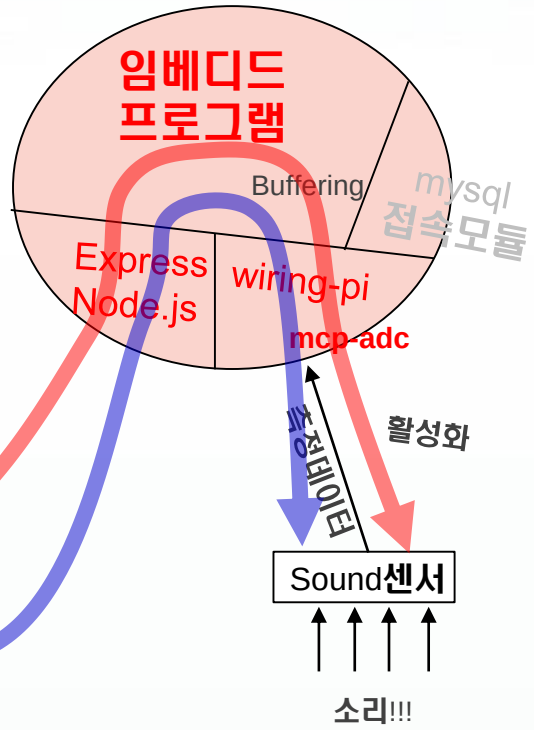
■ 하드웨어 구성 (점퍼케이블 – 핀연결)



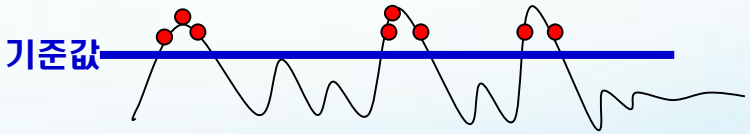
■ Example code

☑ 아날로그 소리센서 측정 샘플코드





아날로그 소리데이터: 0~ 2^{12}-1 (4095)



■ Example code

☑ 웹기반 소리센서 제어 샘플코드 **예제**3 (web_sound.js)

```
const   express = require('express');
const   fs = require('fs');
const   app = express();
const   gpio = require('node-wiring-pi');
const   mcpadc = require('mcp-adc');
const   soundsensor = new  mcpadc.Mcp3208();
const   bodyParser = require('body-parser');

const   CS_MCP3208 = 10      // Chip Enable(CE0) is set
const   SPI_CHANNEL  = 0     // ADC 0번째 채널선택=아날로그센서
const   SPI_SPEED  = 1000000 // 1Mhz
var     QuietSound = 1997;   //  사운드센서 필터링 기준값
var     sid;                 //  TimeOut핸들러 id 값 (타임아웃 취소할때 필요함)
```

기준값은 주변소음/상황에 따라 적절한 값을 찾아 설정하여야 함!! (또는, 가변저항기를 드라이버로 돌려서 기준값을 설정할수도 있음)

```
app.use(bodyParser.urlencoded({ extended:false }));    // body parser 환경설정

const    SoundDetect = function()  {
   soundsensor.readRawValue(SPI_CHANNEL, function(value) {
        if (value > QuietSound)       //  필터링 기준값 이상크기이면, 출력
             console.log("기준값:(%d), 아날로그 측정값(%d)",  QuietSound, value);
        else
             console.log("인식안함");
    });
    sid  =  setTimeout(SoundDetect,  200);   // 타임아웃취소용 sid값 저장
}
```

임베디드 액츄레이터/센서 제어

■ Example code

☑ 웹기반 소리센서 제어 샘플코드

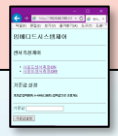
```
process.on('SIGINT', function() {
    console.log("Program Exit...");
    process.exit();
});

app.get('/', (req, res) => {
      console.log("sensor 호출");
       fs.readFile('sen.html','utf8',(error, data) => {
         if (!error)
              res.send( data );
      });
});

app.post('/', (req, res) => {
   let   body = req.body;

   console.log('다음값으로 설정됩니다: ');
   console.log(' ==> : ' + body.threshold);
   QuietSound = body.threshold;
   res.redirect('/');
});
```

sen.html



예제3 (web_sound.js)

```
app.get('/1', (req, res) => {
      console.log("sound sensor ON 호출");
      sid = setTimeout(SoundDetect, 200);   // 활성화
      res.redirect('/');
});

app.get('/0', (req, res) => {
      console.log(" sound sensor OFF 호출");
      clearTimeout(sid);      // 타임아웃취소(비활성화)
      res.redirect('/');
});

app.listen(60001, () => {
      gpio.wiringPiSetup();
      gpio.wiringPiSPISetup(SPI_CHANNEL, SPI_SPEED);
      gpio.pinMode(CS_MCP3208, gpio.OUTPUT);
      console.log("아날로그 사운드센서 제어용 웹서버 …");
      console.log('기준값: ' + QuietSound);
      console.log("http://IP주소:60001/");
});
```

■ Example code

☑ 웹기반 소리센서 제어 샘플코드

예제 3 (sen.html)

```
<!DOCTYPE html>
<html>

   <head>
      <title>Embedded App</title>
   </head>

   <body bgcolor="lightcyan">
      <h2>임베디드시스템제어</h2>
      <hr>
      <h3>센서측정제어</h3>
      <hr>
      <ul>
         <li>
            <a href="/1">사운드센서측정ON</a>
         </li>
         <li>
            <a href="/0">사운드센서측정OFF</a>
         </li>
      </ul>
      <hr>
      <h3>기준값 설정</h3>
      <h6>기준값 입력범위: 0~4095(12비트)
         입력값으로 조정가능</h6>
      <hr>
      <form name="soundon" method="POST">
         <div class="input">
            <span class="label">기준값</span>
            <input type="text" name="threshold">
         </div>
         <hr>
         <div class="input">
            <input type="submit" value="기준값설정">
         </div>
      </form>
   </body>

</html>
```

■ 실습일지