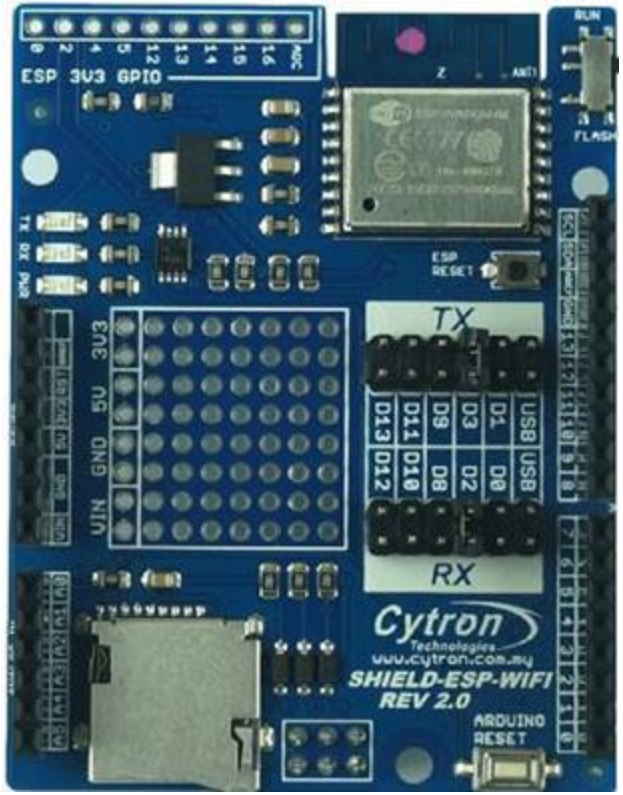
실험 6
WiFi 무선 통신

2024. 10. 09. (Wed)

실험 전 공지사항

- 모듈 수령 확인
  - WiFi Shield
- 교재 변경
  - 모듈 변경으로 인해 교재에 있는 실험 관련 자료는 사용하지 않음
  - 이 발표자료를 기준으로 실험
    - 코드 관련 문제는 먼저 구글링 후 질문



실험 목표

- WiFi shield를 이용한 무선 통신 이해
- 서버를 이용하여 간단한 IoT 시스템 구축

WiFi 기술

- 무선으로 전자기기들을 연결할 수 있도록 한 표준 기술. IEEE에서 제정하여 전세계적으로 널리 사용되며, 이 표준 기술 규격의 브랜드명인 'Wireless Fidelity'를 줄여서 Wi-Fi라고 부른다.
- WiFi 연결 방식
  - **Infrastructure 모드**: 하나의 Access Point(AP)에 다수의 기기가 연결하여 상호 소통. WiFi 신호를 송신하는 AP (E.g 무선 공유기) 가 필요하며, AP에 연결된 기기 간 데이터 송수신이 가능
  - **Ad hoc 모드**: 단말기끼리 직접 접속 (E.g WiFi 기반 무선 스피커)

관련 용어

- IP (Internet Protocol) Address
  - 인터넷에 연결된 모든 기기들에게 할당된 고유 주소. 인터넷 상에서 존재하는 많은 기기들 중 필요한 기기를 찾기 위해 IP 주소를 사용한다. 32bit가 할당되며, 8bit씩 나누어 표시된다.

- Port (포트)
  - 하나의 IP 주소를 가지는 기기를 공유하여 다양한 어플리케이션을 동시에 실행시키기 위해서 사용하는 가상 주소.
  - 자주 사용되는 어플리케이션들은 포트 번호가 지정되어 있다. (e.g. HTTP: 80번, FTP: 21번, POP3 메일 수신: 110번, telnet: 23번)
    - 이번 실험은 **HTTP**를 이용하여 진행하므로, 각 실험에서 port가 **80**번으로 제대로 설정되었는지 확인

관련 용어

- DHCP (Dynamic Host Configuration Protocol)
  - 인터넷을 사용하기 위한 IP 주소를 동적으로 할당 받기 위한 프로토콜. DHCP 서버는 단말기가 일정기간 동안 IP 주소를 사용할 수 있도록 임대를 해주고, 임대기간이 지나면 IP 주소를 반납한다.
    - 본 실험에서는 WiFi의 AP 기능을 수행하는 스마트폰이 DHCP를 처리 한다. 즉, 스마트폰이 DHCP를 요청하는 Arduino 보드에 대응하여 IP 주소를 생성하고 Arduino 보드에 알려준다.

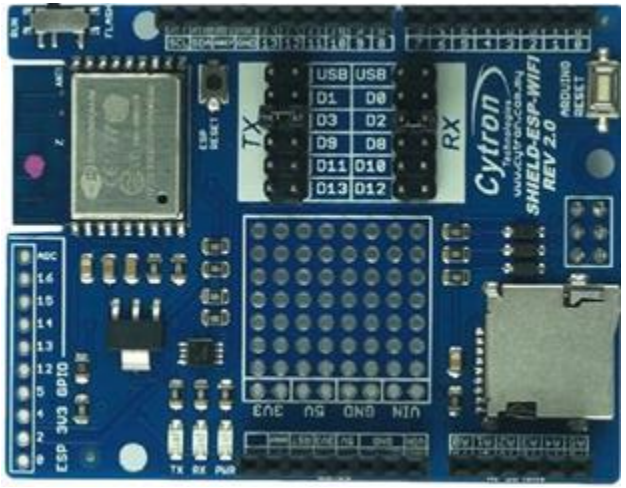
- HTTP (HyperText Transfer Protocol)
  - 인터넷상에서 주로 웹서비스를 위한 데이터 전송을 하는 프로토콜. 주로 웹페이지를 표현하기 위한 언어인 HTML (Hyper Text Markup Language)로 표현된 문서를 주고받는다.

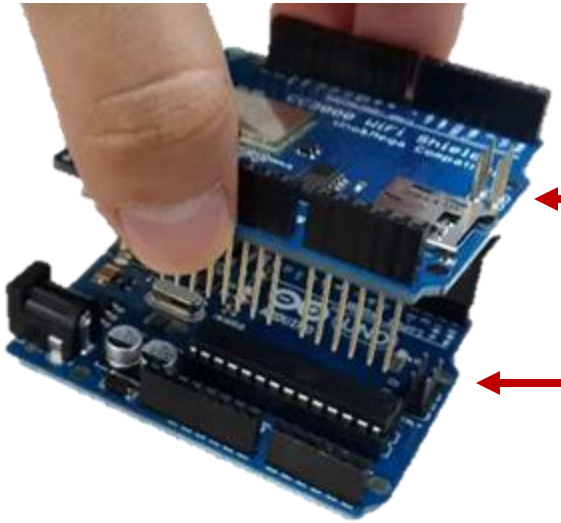
관련 용어

- Telnet
  - 인터넷을 통하여 원격 컴퓨터를 접속하여 자신의 컴퓨터처럼 사용할 수 있는 원격 접속 서비스. 인터넷이 사용되던 초창기에서 쓰이던 네트워크 서비스로서 인터넷이 보급화 된 90년대 이후
    보안이 중요해지면서 현재는 보안에 좀더 강인한 ssh 접속으로 많이 대체되었다.

WiFi Shield

- 아두이노 보드의 기능을 확장하기 위해서 'Shield' 라는 별도의 보드를 장착한다.
- WiFi 기술을 이용한 무선 통신을 위해 WiFi Shield를 사용한다. 우측 사진처럼 기존에 사용하던 Arduino 보드 위에 장착하여 사용한다.





WiFi Shield

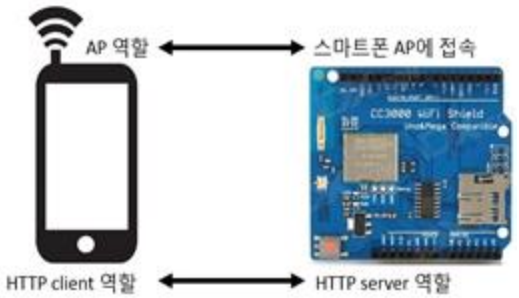
Arduino 보드

실험 내용

- 실험 1: HTTP server
- 실험 2: HTML코딩
- 실험 3: HTML코딩을 통한 링크 생성
- 실험 4: 광센서(analog)를 이용한 LED 제어
- 실험 5: HTTP 서버를 활용한 원격 LED 제어

실험 1: HTTP server

- 본 실험에서는 스마트폰 하나로 , AP 역할과 클라이언트 역할을 모두 수행하도록 한다.



- 1. Cytron ESPWiFi Shield library를 설치한다.
  - 스케치 - 라이브러리 포함하기 - 라이브러리 관리 - "cytron wifi" 검색 후 "Cytron ESPWiFi Shield" 3.0.2 버전 설치

- 2. 접속할 WiFi를 스마트폰 테더링을 통해 생성한다.
  - WPA2 방식으로 암호 설정. (iPhone의 경우 WPA2로 자동 설정됨)
  - 네트워크 이름/암호는 이후 실험에 사용되므로 반드시 기억
    - 네트워크 이름에 한글, 공백, 작은 따옴표 등이 포함되는 경우 Connection error가 발생할 수 있으므로 이에 유의하여 생성.

실험 1: HTTP server

- 3. Cytron ESPWiFi Shield의 예제 "CytronWiFiDemo"를 실행한다. (파일-예제- Cytron ESPWiFi Shield –CytronWiFiDemo)
  - 해당 파일 맨 윗 부분의 ssid, pass에 앞서 설정한 테더링 network ID와 Password 정보를 입력한다.
  - 이때 통신 port가 80으로 되어있는지 확인한다.

```
const char *ssid = "...";
const char *pass = "...";
//IPAddress ip(192, 168, 1 ,242);
ESP8266Server server(80);
```

본 실험의 경우 아두이노 보드는 스마트폰이 생성한 IP주소를 사용하므로 밑줄 코드는 필요X

오류 발생시 해당 코드 주석처리 후 실행

```
if(!wifi.begin(2, 3))
{
  Serial.println(F("Error talking to shield"
  while(1);
}
Serial.println(wifi.firmwareVersion());
Serial.print(F("Mode: "));Serial.println(wi
Serial.println(F("Setup wifi config"));
//wifi.config(ip);
Serial.println(F("Start wifi connection"));
```

실험 1: HTTP server

- 4. void setup() 내부를 살펴본다.

```
void setup() {

  // put your setup code here, to run once:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  if(!wifi.begin(2, 3))
  {
    Serial.println(F("Error talking to shield"));
    while(1);
  }
  Serial.println(wifi.firmwareVersion());
  Serial.print(F("Mode: "));Serial.println(wifi.getMode());// 1-
  // Uncomment these 2 lines if you are using static IP Address
  // Serial.println(F("Setup wifi config"));
  // wifi.config(ip);
  Serial.println(F("Start wifi connection"));
  if(!wifi.connectAP(ssid, pass))
  {
    Serial.println(F("Error connecting to WiFi"));
    while(1);
  }
  Serial.print(F("Connected to "));Serial.println(wifi.SSID());
  Serial.println(F("IP address: "));
  Serial.println(wifi.localIP());
  wifi.updateStatus();
  Serial.println(wifi.status()); //2- wifi connected with ip, 3- s
                                 //4- disconnect with clients or s
  clientTest();
  espblink(100);
  server.begin();
}
```

- Begin → firmware version print → wifi mode print → connect to AP → IP address print → client test → server begin 순으로 initialize 진행.
  - 시리얼 모니터를 통해 위와 같은 순서로 출력되며 시리얼 모니터에 출력된 IP 주소(IP address)를 기록해두고, 다음 실험에서 사용.

```
AT version:0.52.0.0(Jan  7 2016 18:44:24)
SDK version:1.5.1(e67da894)
compile time:Jan  7 2016 19:03:11
Mode: 1
Start wifi connection
Connected to [redacted]
IP address:
172.20.[redacted]
2
```

실험 1: HTTP server

- 5. void loop()에서 serverTest를 call하고 있으므로 void serverTest() 내부를 살펴본다.

```
void serverTest()
{
  ESP8266Client client = server.available();

  if(client.available()>0)
  {
    String req = client.readStringUntil('\r');
    // First line of HTTP request looks like "GET /path HTTP/1.1"
    // Retrieve the "/path" part by finding the spaces
    int addr_start = req.indexOf(' ');
    int addr_end = req.indexOf(' ', addr_start + 1);
    if (addr_start == -1 || addr_end == -1) {
      Serial.print(F("Invalid request: "));
      Serial.println(req);
      return;
    }
    req = req.substring(addr_start + 1, addr_end);
    Serial.print(F("Request: "));
    Serial.println(req);
    client.flush();

    if(req.equals("/"))
    {
      IPAddress ip = wifi.localIP();
      String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' +
      client.print(htmlHeader);
      String htmlBody = "Hello from ESP8266 at ";
      htmlBody += ipStr;
      htmlBody += "</html>\r\n\r\n";
      client.print(htmlBody);
    }
```

client.readString();

수정!!

- 웹서버에 접근하기 위해 브라우저 주소창에 앞 슬라이드의 IP 주소를 적고 enter를 입력한다.

Hello from ESP8266 at 172.20.

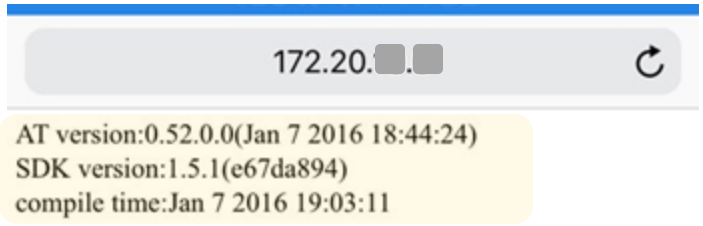
시리얼 모니터 화면

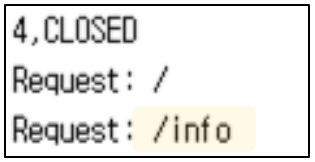
- 접속오류가 뜨는 경우 IP 주소 앞에 "http://" 추가.
- Client에서 enter키를 입력하면, IP주소 이후("/"부터)의 string을 request로 보내고, request string 값에 따라 해당 정보를 출력해준다.

실험 1: HTTP server

- serverTest 함수 내 analog, info가 예제로 제공되어 있다.
- Path에 다양하게 문자열을 넣은 후 브라우저와 시리얼 모니터의 화면을 확인한다.

172.20.xx.x/info

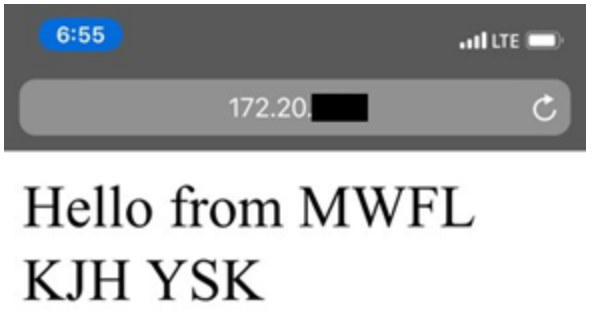




- **6. path "/"에서 "htmlBody" 변수를 변경하여, 아래 예시와 같이 조 번호와 조원 이름이 출력되도록 한다.**
  - "
"을 이용하여 줄 바꿈 수행.

```
htmlBody += "<br>";
```

```
if(req.equals("/"))
{
  IPAddress ip = wifi.localIP();
  String ipStr = String(ip[0]) + '.' +
  client.print(htmlHeader);
  String htmlBody = "Hello from MWFL";
  htmlBody += "<br>";
  htmlBody += "write_your_initial";
  htmlBody += "</html>\r\n\r\n";
  client.print(htmlBody);
}
```



실험 2: HTML코딩

- HTML 형식으로 Client에 다양한 정보를 전달해본다.
- 1. 실험 1의 코드를 바탕으로 IP 주소 뒤에 "/team_info"를 입력하면 새로운 Path로 이동하도록 else if 추가, 아래 코드를 적는다.
  - HTML에서 사용하는 명령어의 집합을 '태그'라고 하며, **여는 태그 < >**로 시작하여 **닫는 태그 </ >**로 끝을 맺는다.
    - HTML은 head와 body로 구성되며, head 안에 title이 존재한다.

```
const char htmlHeader[] = "HTTP/1.1 200 OK₩r₩n"
                          "Content-Type: text/html₩r₩n"
                          "Connection: close₩r₩n₩r₩n"
                          "<!DOCTYPE HTML>₩r₩n"
                          "<html>₩r₩n";
```

코드 상단부에 htmlHeader가 이미 정의되어 있으며, **이 정보로 HTML 형식임을 명시함.**

```
else if(req.equals("/team_info"))
{
  client.print(htmlHeader);
  client.print("<head>");
  client.print("<title>");

  client.print("</title>");
  client.print("</head>");
  client.print("<body>");

  client.print("</body>");
  client.print("</html>");
}
```

실험 2: HTML코딩

- 2. Title 태그 사이에 페이지의 제목으로 "HTML example page"을 추가한다.
  - 페이지의 제목은 브라우저 상단 혹은 열린 탭의 목록에서 확인.

```
client.print("<title>");
client.print("HTML example page"); // 페이지 제목 추가
client.print("</title>");
client.print("</head>");
```

- 3. 같은 방법으로 Body 태그 사이에 본문 내용(조 번호와 조원 이름)을 추가한다.

```
client.print("<body>");
client.print("Team MWFL");
client.print("<br>");
client.print("put_your_inital");
client.print("</body>");
```

- 4. 작성한 코드를 업로드한 후 브라우저 주소창에 "IP 주소/team_info"를 입력하여 아래와 같이 출력되는 것을 확인한다.

페이지 제목 (Title)

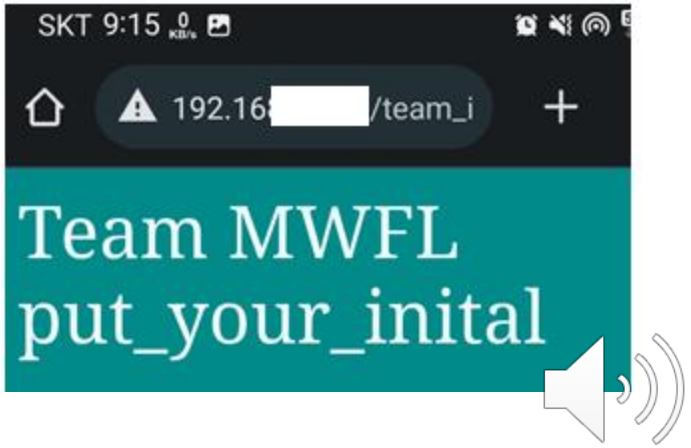
페이지 본문(Body)

실험 2: HTML코딩

- 5. HTML의 다양한 태그를 통해서 페이지를 꾸민다.
  - 배경 색: <body> 태그 속에 bgcolor 속성을 넣는다.
    - 색상표는 인터넷에 'HTML 색상표'를 검색
  - 글씨: <font> 태그를 이용하여, 태그 속에 size 속성을 넣으면 글씨 크기를, color 속성을 넣으면 글씨 색을 변경할 수 있다.
  - E.g.

```
client.print("<body>");
client.print("<body bgcolor=#008B8B>");
client.print("<font size=5 color=00A500>");
client.print("Team MWFL");
client.print("<br>");
client.print("put_your_inital");
client.print("</body>");
```

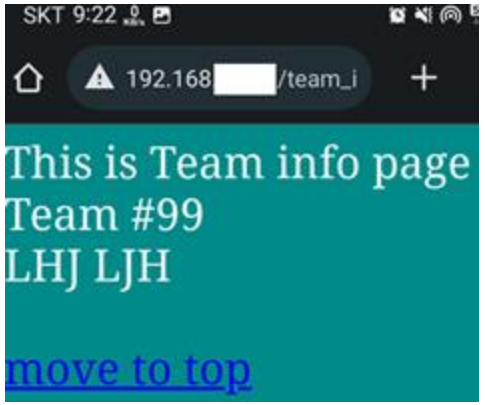
- 6. 추가된 코드 업로드 후 페이지에 접속하면 다음과 같은 출력 화면을 확인할 수 있다.



실험 3: HTML코딩을 통한 링크 생성

- 실험 2 코드를 응용하여, HTML을 이용해 링크를 생성한 뒤 링크를 통해 Path 이동을 해본다.
- 1. 먼저 링크를 만들어 다른 path로 이동이 가능한지 확인한다. "team_info" 페이지 본문 아래에 “move to top”을 프린트하게 하고, <a> 태그를 통해 추가된 문자에 링크를 건다.

```
client.print("This is Team info page");
client.print("<br>");
client.print("Team #99");
client.print("<br>");
client.print("LHJ LJH");
client.print("<br>");
client.print("<br>");
client.print("<a href=./>");
client.print("move to top");
client.print("</a>");
client.print("</body>");
```

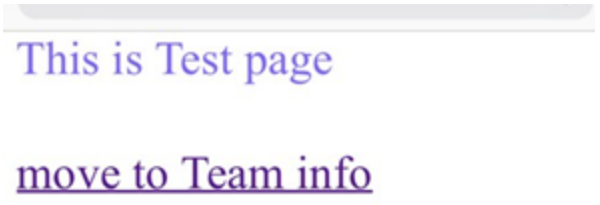


- <a href=./> : IP/으로 이동하는 링크 생성
- **<a href=./team_info> : IP/team_info로 이동하는 링크 생성**

실험 3: HTML코딩을 통한 링크 생성

- 2. 새로운 Path인 "/test"를 만들고, "/team_info" 페이지와 "/test" 페이지가 서로 링크를 이용해서 이동할 수 있게 한다.
  - 이때 페이지마다 서로 다른 내용, 배경색 등을 display하도록 작성해본다.

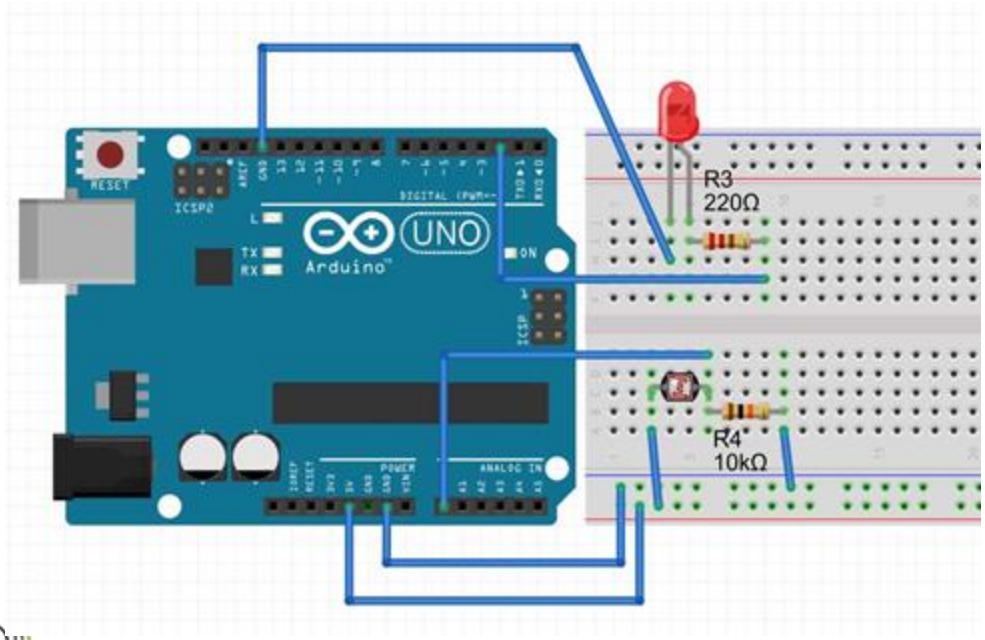
```
else if(req.equals("/team_info"))

else if(req.equals("/test"))
```



실험 4: 조도 센서를 이용한 LED 제어

- 조도 센서로 입력 받은 값을 바탕으로 LED 조명을 제어

- 1. 아래와 같이 회로를 구성한다. (WiFi shield를 장착한 상태에서 진행)
  - LED는 긴 쪽이 (+). 조도센서는 극성이 없음
  - 그림과 다르게 LED pin → D2 대신 **D5 연결필요!**



Arduino (WiFi Shield 장착한 상태)	
D5	LED (+)
GND	LED (-), 저항R4
5V	조도센서
A0	조도센서, 저항R4

실험 4: 조도 센서를 이용한 LED 제어

- 2. 조도 센서를 이용한 LED 제어 프로그램을 작성한다.

```
const int LED_PIN = 5;
const int LED_ON_THRESHOLD = 500; // should be modified according to surrounding

void setup() {
  Serial.begin(9600); //Begin serial communication
  pinMode( LED_PIN, OUTPUT );
}

void loop() {
  int sensed_light = analogRead(A0);
  if( sensed_light > LED_ON_THRESHOLD)
    digitalWrite(LED_PIN, HIGH);
  else
    digitalWrite(LED_PIN, LOW);

  Serial.println(sensed_light);
  delay(500);
}
```

각 수강생의 실험 환경에 따라 시리얼 모니터 확인 후 LED_ON_THRESHOLD 값 수정

- 3. 코드를 업로드 하고 조도 센서로의 입력에 따른 LED의 결과가 올바른지 확인한다.
  - 센서에 입력되는 빛이 (**밝은**/어두운) 상태에서는 serial monitor에 (**높은**/낮은) 값을 출력하고, LED에는 빛이 (**꺼진다**/ 켜진다).

실험 5: HTTP 서버를 활용한 원격 LED 제어

- HTTP 서버를 활용하여 원격으로 조도 센서의 상태를 확인하고, HTML 링크를 응용하여 LED를 제어
- 실험 3의 코드+실험 4의 회로 사용
  실험 3에서 사용했던 HTML 링크 페이지코드에서 시작

- 1. 사용될 변수들을 추가로 선언하고, void setup(void) 내부에도 LED의 pinMode설정을 위한 코드를 추가한다.

```
const char htmlHeader[] = "HTTP/1.1 200 OK₩r₩n"
                          "Content-Type: text/html₩r₩n"
                          "Connection: close₩r₩n₩r₩n"
                          "<!DOCTYPE HTML>₩r₩n"
                          "<html>₩r₩n";
```

```
const int LED_PIN = 5;
const int LED_ON_THRESHOLD = 500;

void setup() {

  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
```

기존 코드를 지우지 않도록 주의!

실험 5: HTTP 서버를 활용한 원격 LED 제어

- 2. void serverTest()내부에도 조도 센서와 LED 관련 코드를 추가한다.
  - Client가 valid request를 요청한 경우에 해당 값을 사용하므로, "if (addr_start == -1 || addr_end == -1) ~" 이후 코드를 추가
  - 값을 확인하기 위해 시리얼 모니터로 intensity 값을 print한다.

```
void serverTest()
{
  ESP8266Client client = server.available();

  if(client.available()>0)
  {
    String req = client.readStringUntil('\r');
    // First line of HTTP request looks like "GET /path HTTP/1.1"
    // Retrieve the "/path" part by finding the spaces
    int addr_start = req.indexOf(' ');
    int addr_end = req.indexOf(' ', addr_start + 1);
    if (addr_start == -1 || addr_end == -1) {
      Serial.print(F("Invalid request: "));
      Serial.println(req);
      return;
    }

    int sensed_light = analogRead(A0);
    Serial.print("Current intensity: ");
    Serial.println(sensed_light);

    req = req.substring(addr_start + 1, addr_end);
    Serial.print(F("Request: "));
    Serial.println(req);
    client.flush();
```

client.readString();

수정!!

실험 5: HTTP 서버를 활용한 원격 LED 제어

- 3. LED 조명 제어를 위해 HTML Link를 응용한다.

  - **home(/), /on, /off** 총 세 개의 path를 만들고, 각 페이지에서 **REFRESH, ON, OFF** Link를 통해 각각 home(/), /on, /off path로 이동할 수 있도록 링크를 설정한다.

  - 이때, /on path로 들어가면 LED를 켜고, /off path로 들어가면 LED를 끄고, Home이나 다른 path에서는 LED 상태는 변화하지 않는다.

실험 5: HTTP 서버를 활용한 원격 LED 제어

```
client.print(htmlHeader);
client.print("<a href=./>"); // home
client.print("REFRESH");
client.print("</a>");
client.print("<br>");
client.print("<a href=./on>"); // on
client.print("ON");
client.print("</a>");
client.print("<br>");
client.print("<a href=./off>"); // off
client.print("OFF");
client.print("</a>");
client.print("<br>");
client.print("<br>");
```

```
if(req.equals("/"))
{
  client.print("controlled nothing");
  client.print("</html>");
}

else if (req.equals("/on"))
{
  client.print("LED is ON");
  digitalWrite(LED_PIN, HIGH);
  client.print("</html>");
}

else if (req.equals("/off"))
{
  client.print("LED is OFF");
  digitalWrite(LED_PIN, LOW);
  client.print("</html>");
}
```

실험 5: HTTP 서버를 활용한 원격 LED 제어

- 4. home(/), /on, /off path의 각 페이지에서 조도 센서의 상태가 나타나도록 코드를 수정한다.
  - HTML 태그를 이용해 배경색과 글씨 색을 변경하고, 본문에는 조번호와 함께 "Room is currently (**BRIGHT**/DARK)"라는 메시지 출력.
    - 밝은 상태 : 밝은 배경 / 어두운 글씨 / "Room is currently BRIGHT“
    - 어두운 상태 : 어두운 배경 / 밝은 글씨 / “Room is currently DARK”

```
client.print(htmlHeader);

if(sensed_light > LED_ON_THRESHOLD){
  client.print("<body bgcolor=#F0FFFF>");
  client.print("<font size=5 color=000000>");
  client.print("Room is currently BRIGHT");
  client.print("<br>");
}
else {
  client.print("<body bgcolor=#000000>");
  client.print("<font size=5 color=F0F8FF>");
  client.print("Room is currently DARK");
  client.print("<br>");
}

client.print("<a href=./>"); // home
client.print("REFRESH");
client.print("</a>");
client.print("<br>");
```

추가

실험 5: HTTP 서버를 활용한 원격 LED 제어

```
if(req.equals("/"))
{
  //client.print(htmlHeader);
  client.print(F("controlled nothing"));
  client.print("</font>");
  client.print("</body>");
  client.print("</html>");

}

  else if(req.equals("/on"))
{

  //client.print(htmlHeader);
  client.print(F("LED is ON"));
  digitalWrite(LED_PIN, HIGH);
  client.print("</font>");
  client.print("</body>");
  client.print("</html>");
}

  else if(req.equals("/off"))
{

  //client.print(htmlHeader);
  client.print(F("LED is OFF"));
  digitalWrite(LED_PIN, LOW);
  client.print("</font>");
  client.print("</body>");
  client.print("</html>");
}
```

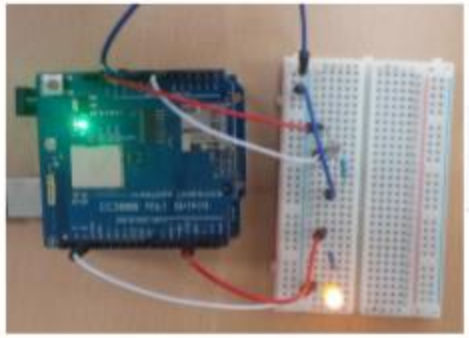
추가

추가

추가

- 조도센서의 intensity를 조절하며 링크를 눌렀을 때 LED의 반응을 확인한다.

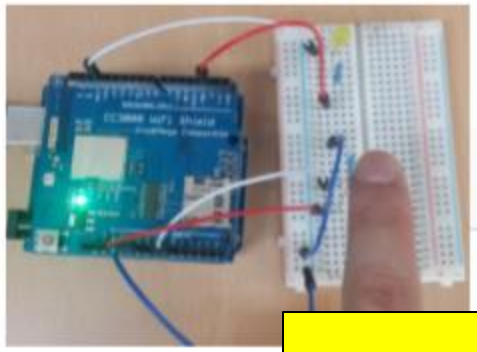
/on path에서 LED가 켜짐



192.168.43.180/on

Light is on

/off path에서 LED가 꺼짐



192.168.43.180/off

Light is off

주의사항: 오래 걸림!

실험 5: HTTP 서버를 활용한 원격 LED 제어

- (참고 - favicon)
  - Favicon은 website의 icon으로 web browser에서 볼 수 있다.
  - HTTP 웹을 print하면 자동으로 favicon을 요청하고 받는 경우가 있으므로, path가 favicon.ico일 때의 조건을 삽입하여 이를 생략할 수 있다.

실험 검사 항목

- 실험 1: HTTP server
  - Internet browser에서 조 번호와 조원 이름 출력
- 실험 2: HTML코딩
  - Internet browser에 HTML 형식으로 디자인된 페이지를 출력
- 실험 3: HTML코딩을 통한 링크 생성
  - internet browser에서 path간 이동이 원활하게 되고, path에 따른 페이지를 올바르게 출력
- 실험 4: 조도 센서(analog)를 이용한 LED 제어
  - LED가 조도 센서로 제어가 되는지 확인
  - 실험 5번과 같이 검사
- 실험 5: HTTP 서버를 활용한 원격 LED 제어
  - 5번 항목의 모든 경우가 올바르게 구현되었는지 확인

결과보고서 및 추가 실험

- 실험시간 내에 검사 받지 못한 항목도 보고서에 포함.
- 배경 지식 & 실험에 대한 고찰

- 추가 실험 A, B
  - 교재 내의 추가 실험 2개 수행
    - 추가 실험 A : HTML 코드를 이미지 삽입
      - client.print("<img src='이미지 주소'>");
      - 위 코드 '이미지 주소' 란에 원하는 이미지의 주소를 입력하여 body에 삽입
    - 추가 실험 B : HTTP 서버를 활용하여 서로 다른 LED 제어

- 각 실험 항목에 대해서 간단한 실험 설명, 코드 설명, 회로 사진, 결과를 보여줄 수 있는 사진 첨부할 것.