

사물인터넷 (IoT) 종결키트

-아두이노 활용편-

박 상 우, 정 동 화

메카솔루션

www.mechasolution.com

목차

1. 시작하기
 - 1) 사물인터넷 (Internet of Things)이란?
 - 2) 사물인터넷의 경제적 가치와 활용 범위
 - 3) 오픈소스 플랫폼과 인터넷 규약
 - 4) 사물인터넷 종결 키트 구성품
2. 아두이노만으로 사용하기
 - 1) 아두이노 기초
 - 2) 인터넷에 연결하기
 - (1) 유선 연결 (이더넷, Ethernet) – DHCP IP
 - (2) 무선 연결 (Wi-Fi, CC3000 칩셋) – DHCP IP
 - (3) 유선 연결 (이더넷, Ethernet) – 고정 IP
 - (4) 무선 연결 (Wi-Fi, CC3000 칩셋) – 고정 IP
 - 3) 아두이노로 인터넷 페이지에 글쓰기
 - 4) 인터넷으로 실내 상태 모니터링 하기
 - 5) LED 제어하기
 - (1) HTML 코딩 없이 하기
 - (2) HTML로 LED 켜고 끄기
 - 6) 서보모터(SG90) 제어하기
 - 7) DC모터 제어하기
 - 8) 입력받은 글자 LCD에 출력하기
3. 프로세싱과 연동하기
 - 1) 아날로그 값 그래프로 보기
 - 2) 밝기에 따라 변하는 박스
4. 참고문헌

1. 시작하기

1) 사물인터넷 (Internet of Things)이란?

사물인터넷 혹은 IoT (Internet of Things)는 모바일 플랫폼의 확산과 어디에서도 연결 가능한 인터넷의 도움으로 인해서 점점 더 시장가치가 확산되고 있으며, 전문가 혹은 비전문가들 사이에서도 큰 주목을 끌고 있습니다. 또한, 구글, 삼성, LG와 같은 거대 기업들이 사물인터넷 시장에 들어오면서 다양한 웨어러블 디바이스가 개발되었고, 아마존이 클라우드 서비스를 시행함으로써 모든 디바이스를 인터넷으로 연결하기에 보다 쉬워졌으며, 오픈소스 플랫폼인 아두이노, 라즈베리파이와 같은 하드웨어의 등장은 실로 사물인터넷의 잠재력을 더 키우게 된 계기가 되었습니다.

사물인터넷에 대한 정의는 기관별로 다르지만 ITU 표준문서에서는 “지능화된 사물들이 연결되어 형성되는 네트워크상에서 사람과 사물, 사물과 사물 간에 상호 소통하고 상황인식 기반의 지식이 결합되어 지능적인 서비스를 제공하는 글로벌 인프라”로 정의하고 있습니다. 간단히 이야기하자면, 인터넷에 연결되어 있는 디바이스들이 상호작용하면서 필요한 서비스를 제공하는 것이라고 생각할 수 있습니다.

2) 사물인터넷의 경제적 가치와 활용 범위

사물인터넷의 경제적 효용가치는 매스컴에서 이슈가 되었다시피 상당히 크다고 합니다. 예측 기관에 따라서 다르지만 IDC에서는 2020년 약 8조 8520억 달러, MarketsandMarkets에서는 1조 2000억 달러로 예측하고 있습니다. 사물인터넷 시장이 거품이라는 일각의 의견과 다르게, 벤처기업 및 대기업에서는 사물인터넷을 활용한 다양한 애플리케이션이 쏟아져 나오고 있는 것은 부인할 수 없는 현실입니다. 사물인터넷의 활용 범위의 한 예로 “난방의 신”이라 불리는 김규호 박사님의 난방수 온도 모니터링 시스템은 우리 실생활에서 발생한 이슈를 센서와 네트워크를 연결하여 얻은 데이터를 분석함으로 새로운 문제해결 방법을 고안해 낸 한 예가 될 수 있습니다. 뿐만 아니라, 심장 질환자들을 대상으로 한 심박측정 IoT 디바이스는 사물인터넷이 의료 기술에 적용된 사례이기도 합니다. 실생활에서 의료분야까지 고도화된 무선 통신 기술의 도움으로 보다 많은 사물들 사이에 네트워크로 연결이 될 것으로 전망되며, 이러한 사물인터넷의 시장의 잠재력 및 가치는 글로벌 시장을 내다보는 현 시점에서 우리로 하여금 교육 훈련과 창의적이고 융합형 인재를 육성해야 할 필요성을 느끼게 합니다.

3) 오픈소스 플랫폼과 인터넷 규약

아두이노와 라즈베리파이: 아두이노와 라즈베리파이는 오픈소스를 기반으로 그 사용자 수와 사용 가능 애플리케이션, 그리고 아두이노와 라즈베리파이를 기반으로 한 관련 혹은 호환 보드들의 종류가 기하 급수적으로 늘어나고 있습니다.

사물인터넷 디바이스를 구현하기 위해서는 인터넷과의 연결이 필수적인데, 기존의 마이크로프로

세서로 잘 알려진 ATMEGA칩들로 구성된 아두이노 시스템들이 와이파이 혹은 이더넷 모듈 제품들과 쉽게 결합이 되고 사용 가능해지면서 사물인터넷 시장에서 아두이노가 차지하는 비중이 크게 늘어나고 있습니다. 뿐만 아니라, 타이니두이노(TinyDuino), 아두이노 Yun, 혹은 피노키오(Pinoccio) 보드와 같은 아두이노를 기반으로 한 관련 제품들이 개발되면서 그 사용 영역은 점점 커지고 있습니다. 리눅스 기반의 OS를 설치하여 마치 PC와 같이 키보드와 마우스 및 각종 컴퓨터 인터페이스를 결합하여 사용할 수 있는 라즈베리파이는 아두이노보다 강력한 마이크로프로세서를 탑재하고, 카메라, 오디오, 비디오 포트가 내장되어 있는 소형 컴퓨터입니다. 내장되어 있는 이더넷 모듈을 통해 사물인터넷 디바이스로 구현할 수 있으며, 와이파이 동글을 사용해서 무선 인터넷 액세스도 가능하여 사물인터넷 디바이스를 개발하기 위한 플랫폼으로 주목 받고 있습니다. 마찬가지로 큐비보드(Cubieboard), 비글본(BeagleBone)과 같은 관련 제품들이 있습니다.



아두이노 우노 R3



타이니두이노



피노키오 보드



라즈베리파이 B형



큐비보드



비글본

다음은, 사물인터넷을 시작하기 전에 기본적으로 알아두면 좋을 규약(프로토콜)에 대해 정리를 해 보았습니다. 예컨대, 아두이노를 사용하여 사물인터넷을 구성한다고 하였을 때, 가장 먼저 떠오르는 것은 인터넷과의 연결 방법에 따른 분류입니다. 즉, 이더넷 케이블을 사용할 것인지 와이파이를 사용할 것인지 결정해야 합니다. 또한 데이터 전송 프로토콜 (TCP, UDP)과 호스트 구성 프로토콜 (BOOT, DHCP)에 따라서도 나누어 보았습니다.

연결 레이어에 따른 분류

802.3 - 이더넷(Ethernet): 간단하게 설명하면 'LAN(Local Area Network, 근거리 통신망)을 위해 개발된 컴퓨터 네트워크 기술'이라고 할 수 있습니다. 제록스에 의해 개발된 후 제록스, DEC, 인텔에 의하여 발전되었습니다. 비동기식 직렬통신방식을 사용하며 IEEE 802.3으로 표준화 되어 있습니다. 물리적으로 동축케이블과 광케이블 등 다양한 형태의 케이블을 통해 노드와 인프라스트럭처 장치(허브, 스위치, 라우터) 사이에서 이루어집니다.

802.11 - 와이파이 (WiFi): Wireless Fidelity의 약자로 IEEE 802.11 기반의 무선랜 연결과 장치 간 연결, PAN/LAN/WAN등을 지원하며 유선 LAN 형태인 이더넷의 단점을 보완하기 위해 만들어진 기술입니다. 802.11 b/g/n이라는 상용화된 규격의 b, g, n은 각각 버전을 의미하며, 시기상으로 b, g, n의 순으로 개발되었고, 802.11b는 최대 11Mbps의 속도를 지원하는데 반해, 802.11n은 600Mbps의 속도를 지원합니다. 802.11b와 802.11g의 차이는 크게 없어서 서로 호환되고 있습니다. AP에서 데이터를 전달해주며 이때 1:n통신이 이뤄집니다. 와이파이 얼라이언스의 상표명이며 여기에 인증되지 않은 기기의 경우는 라이선스 위반이나 인증되지 않았다고 해서 반드시 와이파이 기기와 호환되지 않는다고는 할 수 없습니다. 제조업체들은 이 상표를 자사 인증 제품에 채용하고 있습니다.

2G/3G/4G/LTE: 모바일 무선 통신의 규약으로 스마트폰과 함께 성능면에서 효율면에서 발전하고 있습니다. 2G는 GSM (Global System for Mobile Communications의 약자로 개인 휴대 통신 시스템)와 CDMA(Code Division Multiple Access의 약자로 부호분할다중접속 방식)를 지원하였고, 이보다 발전된 3G는 UMTS와 CDMA2000을 그리고 최근에는 4G와 LTE로 2G와 3G에 비해 월등히 빠른 네트워크를 제공하고 있습니다.

데이터 전송 프로토콜에 따른 분류

TCP: Transmission Control Protocol로 신뢰성이 요구되는 애플리케이션에서 사용하는 데이터 전송 방법입니다. 송신측에서 데이터를 전송하고, 수신측에서 잘 전송을 받았다는 응답을 통해서 통신이 이루어지게 되며, 만약 응답을 받지 못한 경우에는 재전송을 하게 됩니다. 애플리케이션으로는 데이터 손실이 있으면 안 되는 이메일 등에 사용됩니다.

UDP: User Datagram Protocol는 신뢰성을 보장하지는 않지만 간단하고 통신속도가 빠르다는 장점이 있습니다. 즉, 송신측에서는 데이터를 전송하고, 수신측에서 잘 받았는지는 신경쓰지 않습니다. 애플리케이션으로는 데이터 손실이 어느 정도 있더라도 크게 문제가 되지 않을 수 있는 음악이나 영상의 전송들에 사용이 됩니다.

호스트 구성 프로토콜에 따른 분류











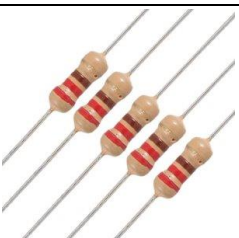

고정 IP: 고정 IP (static IP)란 고정된 IP 주소를 가지고 시간이 지나도 변하지 않습니다. 서버와 같이 보다 신뢰성이 있는 시스템에 좋지만, 보안에 취약할 수 있는 단점도 가지고 있습니다.

BOOTP (유동 IP): Bootstrap Protocol의 약자로 부트스트래핑이라는 기법을 사용한 IP 할당 방법입니다. DHCP보다 이전에 만들어진 프로토콜로 제한적인 기능을 가지고 있습니다.

DHCP (유동 IP): DHCP는 Dynamic Host Configuration Protocol의 약자로 TCP/IP를 위한 IP를 자동적으로 할당해주는 것을 말합니다. 임대방식으로 IP가 일정 시간 유효하며 서버 등에 대해서는 정적인 주소를 제공합니다. Bootp에 비해서 개선된 부분들로 인해 많이 사용되는 방식입니다.

4) 사물인터넷 종결 키트 구성품

사물인터넷 종결 키트는 다음의 구성품들로 이루어져 있습니다.

			
우노 R3 호환보드	USB 케이블	이더넷 실드 (W5100)	LAN 케이블
			
400핀 브레드보드	점퍼 와이어	SG90 미니 서보 모터	SN754410
			
DC 모터 2개	16x2 캐릭터 LCD	40핀 헤더 (male)	가변저항 10K옴
			
220옴 저항 (5개)	빨간색 LED (5개)	조도센서(CdS) 2개	1K옴 저항 (5개)

2. 아두이노만으로 사용하기

1) 아두이노 기초

아두이노란 오픈소스를 기반으로 한 마이크로컨트롤러 보드로, 용도에 따라 다양한 사이즈와 성능을 가지고 있는 보드가 있으며, 프로그래밍 개발 환경은 스케치라는 툴을 제공하고 있다. 특히, 여러 기능에 대한 라이브러리를 제공하고 있고, 오픈소스를 기반으로 많은 사용자가 참여하고 있기 때문에 다양한 소스코드와 라이브러리를 웹상에서 검색하여 사용할 수 있는 장점이 있다. 이 아두이노의 개발로 더불어 다양한 센서, 모터, 디스플레이, 그리고 무선통신 모듈과 같은 전자 소자들은 다시 빛을 발하게 되었는데 그 이유는 아두이노와의 손쉬운 연결 및 사용방법 덕분이다. 오픈소스라는 장점으로 인해서 다양한 모양과 성능의 아두이노 호환보드 혹은 관련 보드가 시중에 나오게 되었는데, 본 매뉴얼에서는 가장 많이 사용되고 있는 기본 보드인 Uno를 사용하도록 한다.

소프트웨어 설치부터 기본적인 아두이노 사용방법에 대해서는 메카솔루션의 “입문자를 위한 아두이노 종결킷 매뉴얼”을 참고하면 되며, arduino.cc 공식 웹사이트를 통해서 보다 광범위한 정보를 얻을 수 있다. PDF 매뉴얼 링크: [<http://mechasolution.com/shop/board/list.php?&id=notice>]

2) 인터넷에 연결하기

사물인터넷의 핵심은 디바이스를 인터넷에 연결하는 것이기 때문에, 두 가지 옵션인 유선(Ethernet)과 무선(Wi-Fi)으로 인터넷에 연결해보도록 합니다. 고정 IP의 경우와 DHCP IP의 경우에 따라서 각각의 소스코드가 다르기 때문에 사용하는 IP가 고정 IP인지 혹은 DHCP IP인지 확인 후에 원하는 연결 방식 (유선 혹은 무선)을 선택하도록 합니다.

(1) 유선 연결 (이더넷, Ethernet) – DHCP IP

아두이노에 이더넷 쉴드를 적층한 후에, LAN 케이블을 이더넷 쉴드의 LAN 포트에 끼우고, 다른 쪽은 인터넷 공유기 혹은 연결 가능한 LAN 포트에 끼웁니다.

DHCP로 IP를 할당 받게 될 것이며, 받은 IP를 시리얼 모니터를 통해 출력해주는 데모입니다.

아두이노 소스코드

/*

DHCP-based IP printer

This sketch uses the DHCP extensions to the Ethernet library

to get an IP address via DHCP and print the address obtained.
using an Arduino Wiznet Ethernet shield.

Circuit:

* Ethernet shield attached to pins 10, 11, 12, 13

created 12 April 2011

modified 9 Apr 2012

by Tom Igoe

*/

#include <SPI.h>

#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED } ; //mac address

EthernetClient client;

void setup() {

 // Open serial communications and wait for port to open:

 Serial.begin(9600);

 // start the Ethernet connection:

 if (Ethernet.begin(mac) == 0) {

 Serial.println("Failed to configure Ethernet using DHCP");

 // no point in carrying on, so do nothing forevermore:

 while(1) ;

 }

 // print your local IP address:

 Serial.print("My IP address: ");

 for (byte thisByte = 0; thisByte < 4; thisByte++) {

 // print the value of each byte of the IP address:

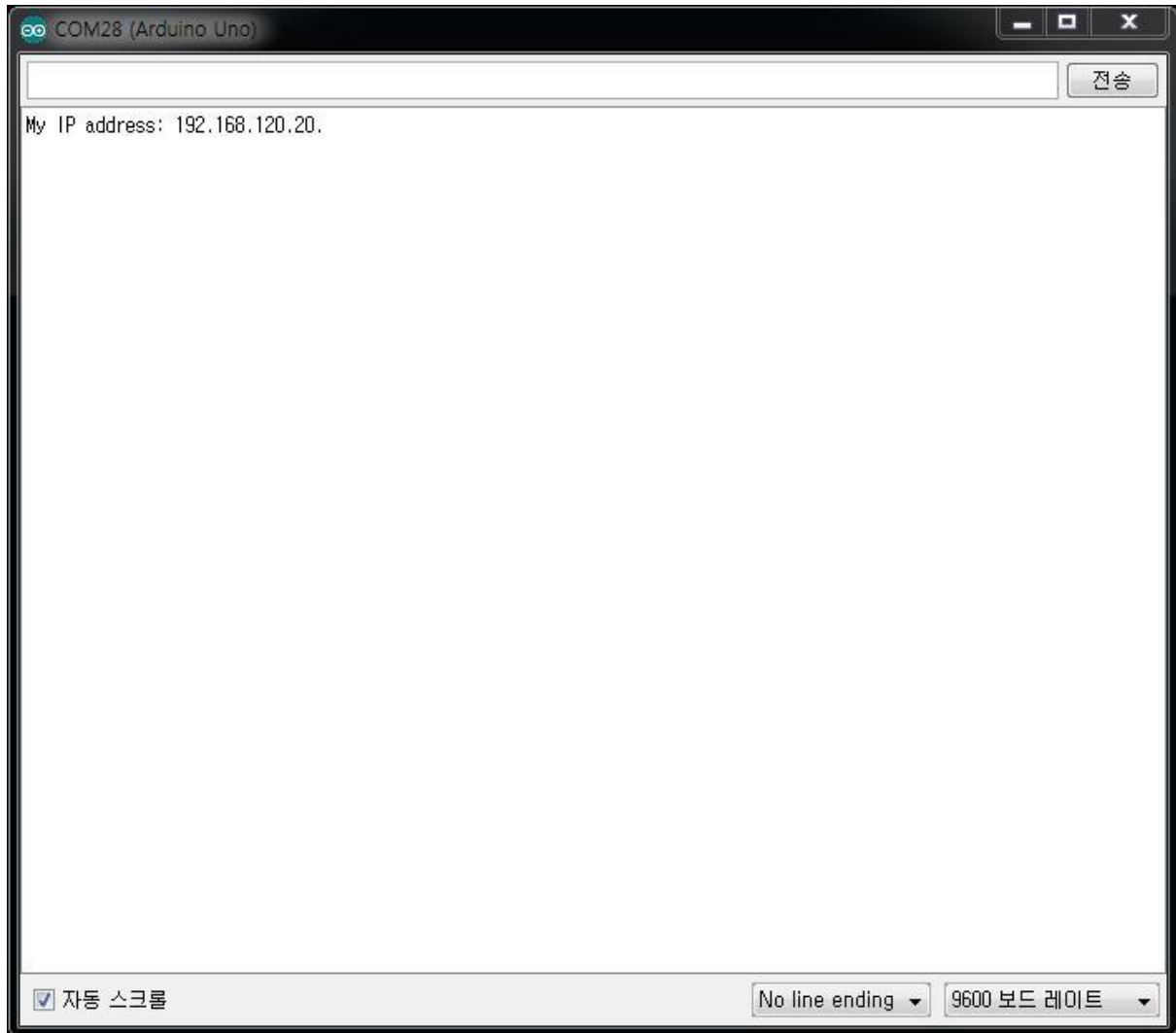
 Serial.print(Ethernet.localIP()[thisByte], DEC);

 Serial.print(".");

 }

```
}  
void loop() {  
}
```

업로드를 마친 후에 시리얼 모니터링을 통해 확인한 IP 값은 다음과 같습니다.



(2) 무선 연결 (Wi-Fi, CC3000 칩셋) – DHCP IP

스파크펀 CC3000 와이파이 쉴드를 사용하여 DHCP로 IP를 할당 받는 데모입니다. 스파크펀 와이파이 쉴드를 아두이노 우노 보드 위에 적층한 후에, 전원을 공급하면 와이파이 신호를 통해 센서 데이터를 원격으로 모니터링하거나 모터 및 디스플레이 제어가 가능하게 됩니다.

연결 코드를 사용하기 위해서는 다음의 링크에서 라이브러리 파일을 다운로드 받으신 후에, 압축을 풀어주세요. 압축이 풀린 폴더는 아두이노의 라이브러리 폴더에 넣은 후, 아두이노 소프트웨어

를 재실행하면, Examples(예제)에 ConnectionTest라는 예제 코드를 확인할 수 있습니다.

아두이노 소스코드

/*****

ConnectionTest.ino

CC3000 Connection Test

Shawn Hymel @ SparkFun Electronics

January 30, 2014

https://github.com/sparkfun/SFE_CC3000_Library

Connects to the access point given by the SSID and password and waits for a DHCP-assigned IP address. To use a static IP address, change the #define USE_DHCP from 1 to 0 and assign an IP address to static_ip_addr in the Constants section.

NOTE: Static IP is not working at this time.

The security mode is defined by one of the following:

WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA, WLAN_SEC_WPA2

Hardware Connections:

Uno Pin	CC3000 Board	Function
+5V	VCC or +5V	5V
GND	GND	GND
2	INT	Interrupt
7	EN	WiFi Enable
10	CS	SPI Chip Select
11	MOSI	SPI MOSI
12	MISO	SPI MISO
13	SCK	SPI Clock

Resources:

Include SPI.h and SFE_CC3000.h

Development environment specifics:

Written in Arduino 1.0.5

Tested with Arduino UNO R3

This code is beerware; if you see me (or any other SparkFun employee) at the local, and you've found our code helpful, please buy us a round!

Distributed as-is; no warranty is given.

*****/

```
#include <SPI.h>
```

```
#include <SFE_CC3000.h>
```

```
// Pins
```

```
#define CC3000_INT      2    // Needs to be an interrupt pin (D2/D3)
```

```
#define CC3000_EN       7    // Can be any digital pin
```

```
#define CC3000_CS      10    // Preferred is pin 10 on Uno
```

```
// IP address assignment method
```

```
#define USE_DHCP        1    // 0 = static IP, 1 = DHCP
```

```
// Connection info data lengths
```

```
#define IP_ADDR_LEN     4    // Length of IP address in bytes
```

```
#define MAC_ADDR_LEN    6    // Length of MAC address in bytes
```

```
// Constants
```

```
char ap_ssid[] = "SSID";           // SSID of network
```

```
char ap_password[] = "PASSWORD";   // Password of network
```

```
unsigned int ap_security = WLAN_SEC_WPA2; // Security of network
```

```
unsigned int timeout = 30000;       // Milliseconds
```

```
//const char static_ip_addr[] = "0.0.0.0";
```

```

// Global Variables
SFE_CC3000 wifi = SFE_CC3000(CC3000_INT, CC3000_EN, CC3000_CS);

void setup() {

    ConnectionInfo connection_info;
    int i;

    // Initialize Serial port
    Serial.begin(115200);
    Serial.println();
    Serial.println("-----");
    Serial.println("SparkFun CC3000 - Connection Test");
    Serial.println("-----");

    // Initialize CC3000 (configure SPI communications)
    if ( wifi.init() ) {
        Serial.println("CC3000 initialization complete");
    } else {
        Serial.println("Something went wrong during CC3000 init!");
    }

    #if (USE_DHCP == 1)
        // Connect using DHCP
        Serial.print("Connecting to: ");
        Serial.println(ap_ssid);
        if(!wifi.connect(ap_ssid, ap_security, ap_password, timeout)) {
            Serial.println("Error: Could not connect to AP");
        }
    #elif (USE_DHCP == 0)
        // Connect using static IP
        // ***TODO: Connect using static IP
    #endif

    // Print out connection details

```

```

if( !wifi.getConnectionInfo(connection_info) ) {
    Serial.println("Error: Could not obtain connection details");
} else {
    Serial.println("Connected!");
    Serial.println();

    // Print MAC address
    Serial.print("CC3000 MAC Address: ");
    for ( i = 0; i < MAC_ADDR_LEN; i++ ) {
        if ( connection_info.mac_address[i] < 0x10 ) {
            Serial.print("0");
        }
        Serial.print(connection_info.mac_address[i], HEX);
        if ( i < MAC_ADDR_LEN - 1 ) {
            Serial.print(":");
        }
    }
    Serial.println();

    // Print IP Address
    Serial.print("IP Address: ");
    printIPAddr(connection_info.ip_address);
    Serial.println();

    // Print subnet mask
    Serial.print("Subnet Mask: ");
    printIPAddr(connection_info.subnet_mask);
    Serial.println();

    // Print default gateway
    Serial.print("Default Gateway: ");
    printIPAddr(connection_info.default_gateway);
    Serial.println();

    // Print DHCP server address

```

```

    Serial.print("DHCP Server: ");
    printIPAddr(connection_info.dhcp_server);
    Serial.println();

    // Print DNS server address
    Serial.print("DNS Server: ");
    printIPAddr(connection_info.dns_server);
    Serial.println();

    // Print SSID
    Serial.print("SSID: ");
    Serial.println(connection_info.ssid);
    Serial.println();
}

// Disconnect
if ( wifi.disconnect() ) {
    Serial.println("Disconnected");
} else {
    Serial.println("Error: Could not disconnect from network");
}

// Done!
Serial.println("Finished connection test");
}

void loop() {

    // Do nothing
    delay(1000);

}

// Print out an IP Address in human-readable format

```

```

void printIPAddr(unsigned char ip_addr[]) {
    int i;

    for (i = 0; i < IP_ADDR_LEN; i++) {
        Serial.print(ip_addr[i]);
        if ( i < IP_ADDR_LEN - 1 ) {
            Serial.print(".");
        }
    }
}

```

(3) 유선 연결 (이더넷, Ethernet) – 고정 IP

이더넷을 통하여 고정 IP로 인터넷에 연결한 후 Mechasolution.com으로 핑을 보내 응답시간을 출력합니다.

아두이노 소스코드

```

/*
Ping Example
This example sends an ICMP pings every 500 milliseconds, sends the human-readable
result over the serial port.

Circuit:
* Ethernet shield attached to pins 10, 11, 12, 13
created 30 Sep 2010
by Blake Foster
*/

#include <SPI.h>
#include <Ethernet.h>
#include <ICMPPing.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //mac address
byte ip[] = {192, 168, 120, 20}; // ip address for ethernet shield
IPAddress pingAddr(211, 233, 50, 220); // ip address to ping

```



```

SOCKET pingSocket = 0;

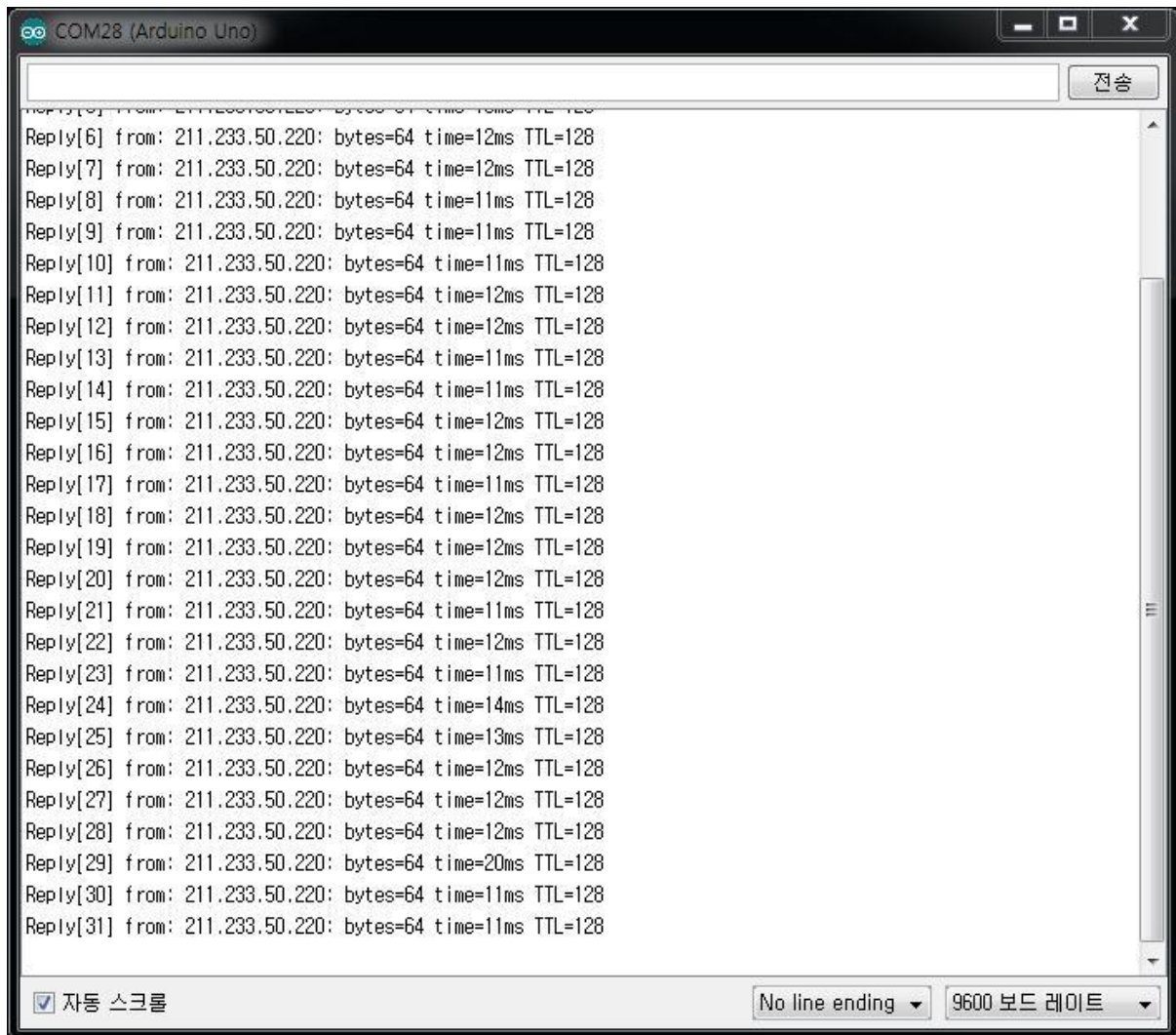
char buffer [256];
ICMPPing ping(pingSocket, (uint16_t)random(0, 255));

void setup()
{
    // start Ethernet
    Ethernet.begin(mac, ip);
    Serial.begin(9600);
}

void loop()
{
    ICMPEchoReply echoReply = ping(pingAddr, 4);
    if (echoReply.status == SUCCESS)
    {
        sprintf(buffer,
            "Reply[%d] from: %d.%d.%d.%d: bytes=%d time=%ldms TTL=%d",
            echoReply.data.seq,
            echoReply.addr[0],
            echoReply.addr[1],
            echoReply.addr[2],
            echoReply.addr[3],
            REQ_DATASIZE,
            millis() - echoReply.data.time,
            echoReply.ttl);
    }
    else
    {
        sprintf(buffer, "Echo request failed; %d", echoReply.status);
    }
    Serial.println(buffer);
    delay(500);
}

```

프로그램 업로드를 마친 후에 시리얼 모니터링을 통해 확인한 IP와 응답시간을 통해 아두이노가 인터넷에 연결되었는지 확인할 수 있습니다.



(4) 무선 연결 (Wi-Fi, CC3000 칩셋) – 고정 IP

고정 IP를 이용하여 CC3000을 통해 와이파이에 연결합니다. 기본 소스코드는 DHCP 소스코드와 같지만, 다음 소스코드의 굵은 폰트의 두 라인을 수정하시면 됩니다. `#define USE_DHCP` 값을 1에서 0으로 변환하고, `const char static_ip_addr[]` 앞의 `"/"` 주석 표시를 없애고, 0,0,0,0 대신에 사용하는 IP를 입력합니다.

아두이노 소스코드

/*****

ConnectionTest.ino

CC3000 Connection Test

Shawn Hymel @ SparkFun Electronics

January 30, 2014

https://github.com/sparkfun/SFE_CC3000_Library

Connects to the access point given by the SSID and password and waits for a DHCP-assigned IP address. To use a static IP address, change the #define USE_DHCP from 1 to 0 and assign an IP address to static_ip_addr in the Constants section.

NOTE: Static IP is not working at this time.

The security mode is defined by one of the following:

WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA, WLAN_SEC_WPA2

Hardware Connections:

Uno Pin	CC3000 Board	Function
+5V	VCC or +5V	5V
GND	GND	GND
2	INT	Interrupt
7	EN	WiFi Enable
10	CS	SPI Chip Select
11	MOSI	SPI MOSI
12	MISO	SPI MISO
13	SCK	SPI Clock

Resources:

Include SPI.h and SFE_CC3000.h

Development environment specifics:

Written in Arduino 1.0.5

Tested with Arduino UNO R3

This code is beerware; if you see me (or any other SparkFun employee) at the local, and you've found our code helpful, please buy us a round!

Distributed as-is; no warranty is given.

*****/

```
#include <SPI.h>
```

```
#include <SFE_CC3000.h>
```

```
// Pins
```

```
#define CC3000_INT      2    // Needs to be an interrupt pin (D2/D3)
```

```
#define CC3000_EN       7    // Can be any digital pin
```

```
#define CC3000_CS       10   // Preferred is pin 10 on Uno
```

```
// IP address assignment method
```

```
#define USE_DHCP        0    // 0 = static IP, 1 = DHCP
```

```
// Connection info data lengths
```

```
#define IP_ADDR_LEN     4    // Length of IP address in bytes
```

```
#define MAC_ADDR_LEN    6    // Length of MAC address in bytes
```

```
// Constants
```

```
char ap_ssid[] = "SSID";           // SSID of network
```

```
char ap_password[] = "PASSWORD";   // Password of network
```

```
unsigned int ap_security = WLAN_SEC_WPA2; // Security of network
```

```
unsigned int timeout = 30000;       // Milliseconds
```

```
const char static_ip_addr[] = "192.168.120.20";
```

```
// Global Variables
```

```
SFE_CC3000 wifi = SFE_CC3000(CC3000_INT, CC3000_EN, CC3000_CS);
```

```

void setup() {

    ConnectionInfo connection_info;
    int i;

    // Initialize Serial port
    Serial.begin(115200);
    Serial.println();
    Serial.println("-----");
    Serial.println("SparkFun CC3000 - Connection Test");
    Serial.println("-----");

    // Initialize CC3000 (configure SPI communications)
    if ( wifi.init() ) {
        Serial.println("CC3000 initialization complete");
    } else {
        Serial.println("Something went wrong during CC3000 init!");
    }

    #if (USE_DHCP == 1)
        // Connect using DHCP
        Serial.print("Connecting to: ");
        Serial.println(ap_ssid);
        if(!wifi.connect(ap_ssid, ap_security, ap_password, timeout)) {
            Serial.println("Error: Could not connect to AP");
        }
    #elif (USE_DHCP == 0)
        // Connect using static IP
        // ***TODO: Connect using static IP
    #endif

    // Print out connection details
    if( !wifi.getConnectionInfo(connection_info) ) {
        Serial.println("Error: Could not obtain connection details");
    } else {

```

```

Serial.println("Connected!");
Serial.println();

// Print MAC address
Serial.print("CC3000 MAC Address: ");
for ( i = 0; i < MAC_ADDR_LEN; i++ ) {
    if ( connection_info.mac_address[i] < 0x10 ) {
        Serial.print("0");
    }
    Serial.print(connection_info.mac_address[i], HEX);
    if ( i < MAC_ADDR_LEN - 1 ) {
        Serial.print(":");
    }
}
Serial.println();

// Print IP Address
Serial.print("IP Address: ");
printIPAddr(connection_info.ip_address);
Serial.println();

// Print subnet mask
Serial.print("Subnet Mask: ");
printIPAddr(connection_info.subnet_mask);
Serial.println();

// Print default gateway
Serial.print("Default Gateway: ");
printIPAddr(connection_info.default_gateway);
Serial.println();

// Print DHCP server address
Serial.print("DHCP Server: ");
printIPAddr(connection_info.dhcp_server);
Serial.println();

```

```

    // Print DNS server address
    Serial.print("DNS Server: ");
    printIPAddr(connection_info.dns_server);
    Serial.println();

    // Print SSID
    Serial.print("SSID: ");
    Serial.println(connection_info.ssid);
    Serial.println();
}

// Disconnect
if ( wifi.disconnect() ) {
    Serial.println("Disconnected");
} else {
    Serial.println("Error: Could not disconnect from network");
}

// Done!
Serial.println("Finished connection test");
}

void loop() {

    // Do nothing
    delay(1000);
}

// Print out an IP Address in human-readable format
void printIPAddr(unsigned char ip_addr[]) {
    int i;

```

```

for (i = 0; i < IP_ADDR_LEN; i++) {
    Serial.print(ip_addr[i]);
    if ( i < IP_ADDR_LEN - 1 ) {
        Serial.print(".");
    }
}
}

```

3) 아두이노로 인터넷 페이지에 글쓰기

이더넷шил드에 할당된 IP로 접속할 시 Hello Arduino! This web page from the Arduino server라는 글자가 보입니다. 이는 html로 코딩되었으며 여기서 IP는 192.168.120.20, 포트는 90번을 사용합니다.

아두이노 소스코드

```

#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //mac address
IPAddress ip(192, 168, 120, 20); // IP address, may need to change depending on network
EthernetServer server(90); // create a server at port 90

void setup()
{
    Ethernet.begin(mac, ip); // initialize Ethernet device
    server.begin();          // start to listen for clients
}

void loop()
{
    EthernetClient client = server.available(); // try to get client

    if (client) { // got client?
        boolean currentLineIsBlank = true;

```



```

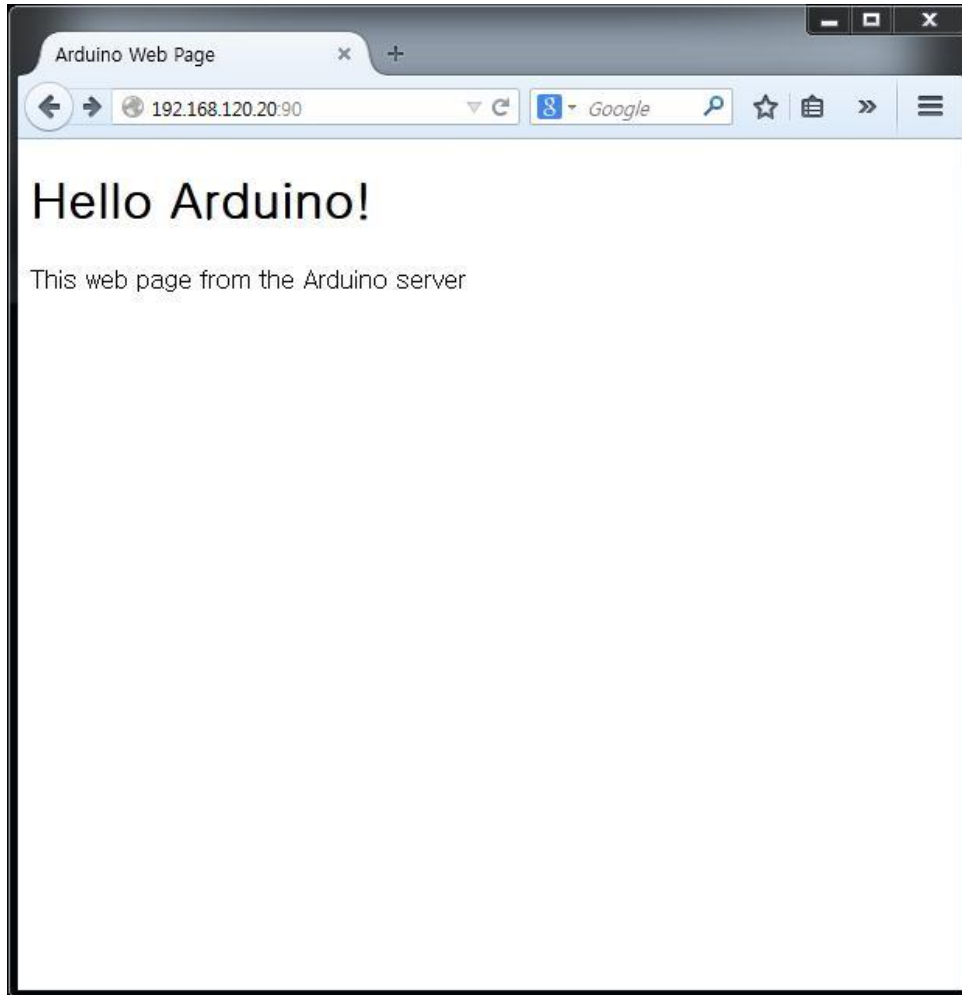
while (client.connected()) {
  if (client.available()) { // client data available to read
    char c = client.read(); // read 1 byte (character) from client
    // last line of client request is blank and ends with \n
    // respond to client only after last line received
    if (c == '\n') {
      // send a standard http response header
      client.println("HTTP/1.1 200 OK");
      client.println("Content-Type: text/html");
      client.println("Connection: close");
      client.println();
      // send web page
      client.println("<!DOCTYPE html>");
      client.println("<html>");
      client.println("<head>");
      client.println("<title>Arduino Web Page</title>");
      client.println("</head>");
      client.println("<body>");
      client.println("<h1>Hello Arduino!</h1>");
      client.println("This web page from the Arduino server");
      client.println("</body>");
      client.println("</html>");
      break;
    }
    // every line of text received from the client ends with \r\n
  } // end if (client.available())
} // end while (client.connected())
delay(1); // give the web browser time to receive the data
client.stop(); // close the connection
} // end if (client)
}

```

위의 아두이노 소스코드가 아두이노에 업로드가 된 후에는 브라우저를 열고 자신의 IP와 포트번호를 다음과 같이 주소 창에 타이핑합니다.

192.168.120.20:90

정상적으로 인터넷이 연결되었다면 다음과 같이 웹페이지에 문구가 출력되는 것을 확인할 수 있습니다. Client.println 함수를 사용하여 다른 문구로 변경하는 테스트도 해 보시기 바랍니다.

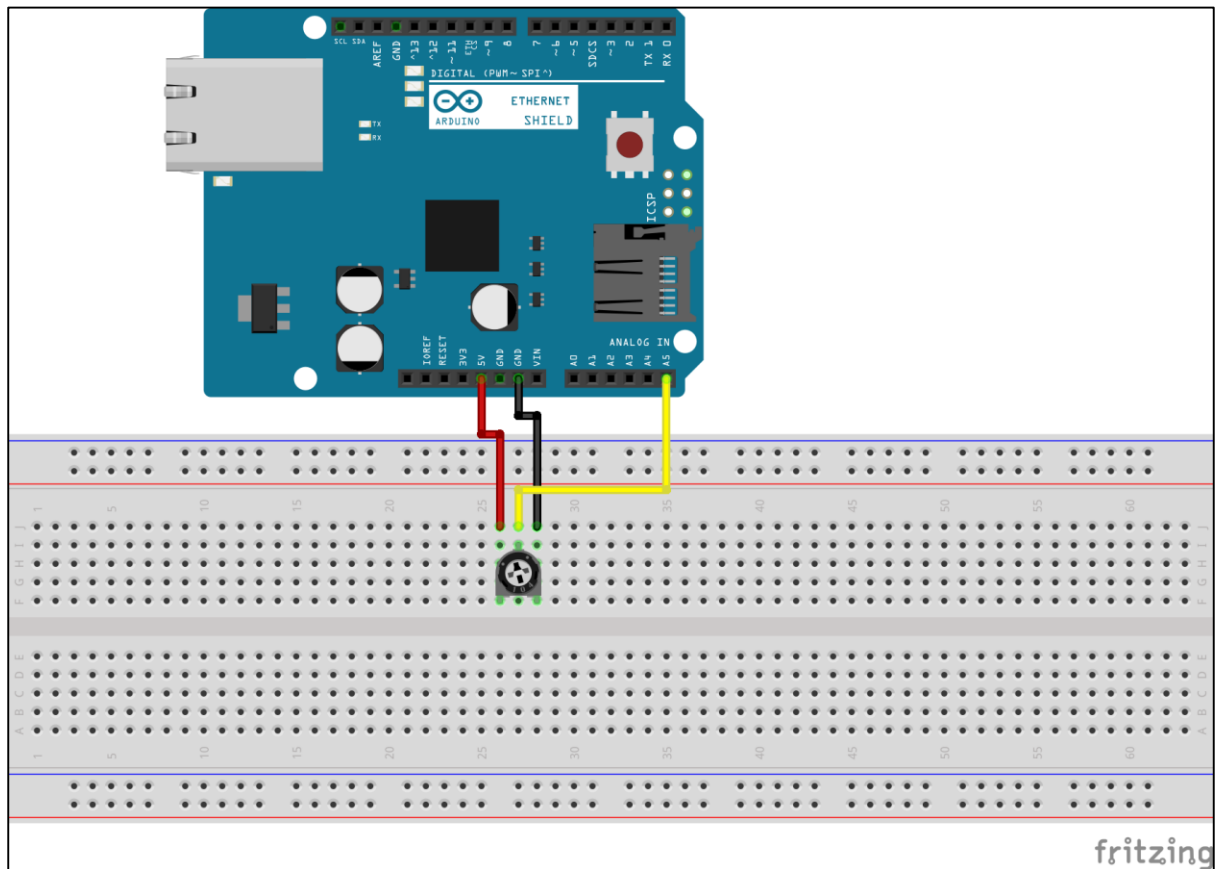


4) 인터넷으로 실내 상태 모니터링 하기

사물인터넷 프로젝트에서 빈번하게 소개되는 것이 실내 상태 모니터링입니다. 즉, 실내온도, 습도, 먼지레벨, 유해가스 검출 등을 스마트폰으로 모니터링하거나 원격에서 실시간으로 확인하기 등이 이에 해당이 됩니다. 이번 챕터에서는 아날로그 센서 대신에 가변저항을 사용하여 웹페이지에 전송하는 데모를 제작하였습니다. 다양한 아날로그 센서들을 가변저항 대신에 교체하면 원하는 상태 값을 웹페이지로 실시간 확인이 가능합니다.

준비물: 아두이노 우노, 이더넷 쉴드, LAN 케이블, USB 케이블, 점퍼 와이어, 브레드보드, 가변저항

회로구성:



아두이노 소스코드

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //mac address
IPAddress ip(192, 168, 120, 20);
EthernetServer server(90);

String test = "" ;

void setup() {
  // put your setup code here, to run once:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.begin(9600);
```

```

}

void loop() {
    // put your main code here, to run repeatedly:
    EthernetClient client = server.available() ;
    if (client) {
        boolean currentLineIsBlank = true ;
        while (client.connected()) {
            if (client.available()) {
                char c = client.read() ;
                test += c ;
                if (c == '\n' && currentLineIsBlank) {
                    client.println("HTTP/1.1 200 OK") ;
                    client.println("Content-Type: text/html") ;
                    client.println("Connection: close") ;
                    client.println() ;
                    client.println("<!DOCTYPE html>") ;
                    client.println("<html>") ;
                    client.println("<head>") ;
                    client.println("<title>anlaogRead</title>") ;
                    client.println("<meta http-equiv='refresh' content='0'/>") ;
                    client.println("</head>") ;
                    client.println("<body onload=W\"call()W\">") ;

                    String a_val = "<p>" ;
                    a_val += String(analogRead(5)) ;
                    a_val += "</p>" ;

                    client.println(a_val) ;
                    client.println("</body>") ;
                    client.println("</html>") ;
                    Serial.println(test) ;
                    test = "" ;
                    break ;
                }
            }
        }
    }
}

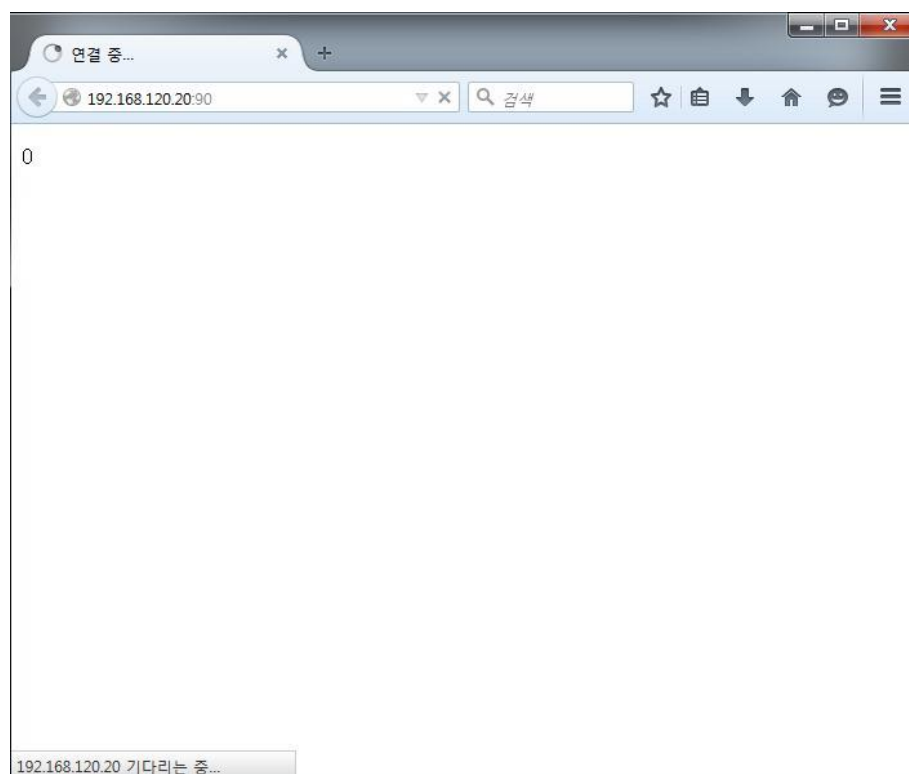
```

```

    if (c == '\n') {
        currentLineIsBlank = true ;
    }
    else if (c != '\r') {
        currentLineIsBlank = false ;
    }
}
}
delay(1) ;
client.stop() ;
}
}

```

프로그램 업로드 후, 주소창에 자신의 IP와 포트번호를 통해서 값이 변하는 것을 확인할 수 있습니다.



5) LED 제어하기

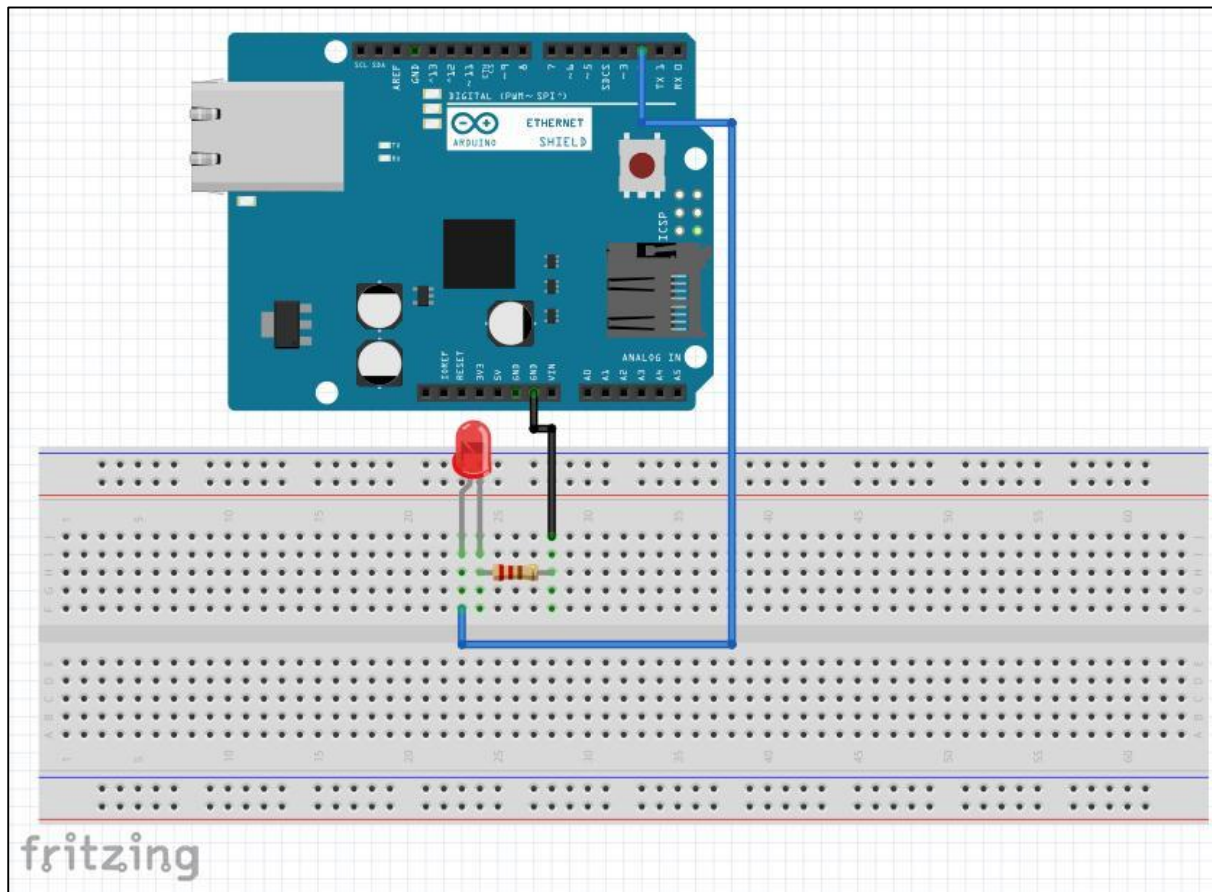
이번 챕터에서는 원격으로 아두이노에 연결된 LED를 제어하는 방법에 대해서 알아보도록 하겠습니다. 웹페이지로 LED를 제어하기 위해서 두 가지 방법을 사용하였는데, 첫번째 방법은 HTML 코

딩 없이 주소창에 컨트롤할 값을 넣어준 후에 LED를 켜고 끄는 예제이고, 두번째 방법은

(1) HTML 코딩 없이 하기

준비물: 아두이노 우노, 이더넷 실드, LAN 케이블, USB 케이블, 점퍼와이어, 빨간색 LED, 220옴 저항, 브레드보드

회로도



아두이노 소스코드

/*

Web Server Demo

thrown together by Randy Sarafan

Allows you to turn on and off an LED by entering different urls.

To turn it on:

`http://your-IP-address/$1`

To turn it off:

`http://your-IP-address/$2`

Circuit:

- * Ethernet shield attached to pins 10, 11, 12, 13
- * Connect an LED to pin D2 and put it in series with a 220 ohm resistor to ground

Based almost entirely upon Web Server by Tom Igoe and David Mellis

Edit history:

created 18 Dec 2009

by David A. Mellis

modified 4 Sep 2010

by Tom Igoe

`*/`

`#include <SPI.h>`

`#include <Ethernet.h>`

`boolean incoming = 0;`

`byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED } ; //mac address`

`IPAddress ip(192, 168, 120, 20) ;`

`EthernetServer server(80) ;`

`void setup()`

`{`

`pinMode(2, OUTPUT);`

`// start the Ethernet connection and the server:`

`Ethernet.begin(mac, ip);`

```

server.begin();
Serial.begin(9600);
}

void loop()
{
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        //reads URL string from $ to first blank space
        if (incoming && c == ' ') {
          incoming = 0;
        }
        if (c == '$') {
          incoming = 1;
        }

        //Checks for the URL string $1 or $2
        if (incoming == 1) {
          Serial.println(c);

          if (c == '1') {
            Serial.println("ON");
            digitalWrite(2, HIGH);
          }
          if (c == '2') {
            Serial.println("OFF");
            digitalWrite(2, LOW);
          }
        }
      }
    }
  }
}

```

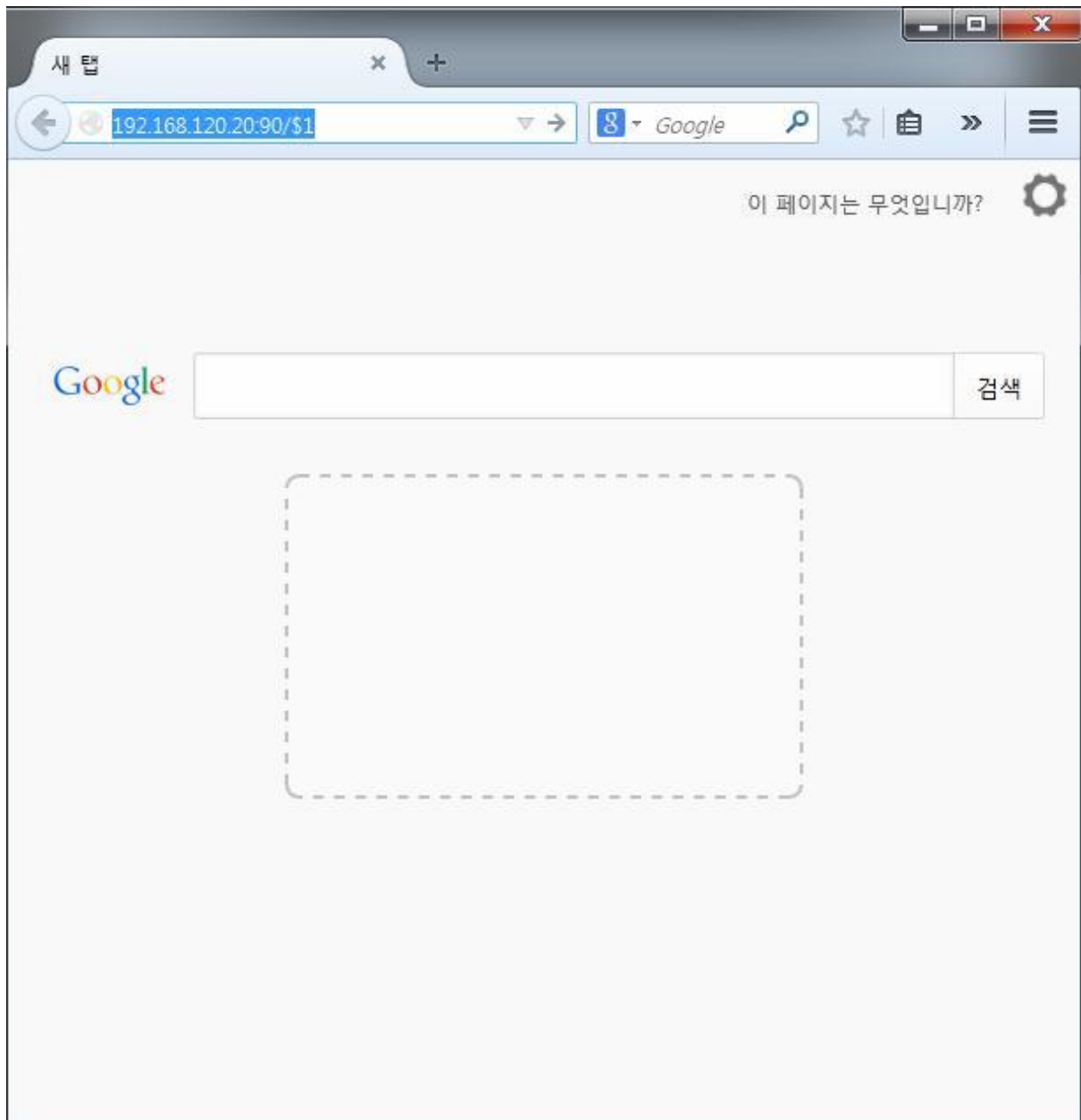


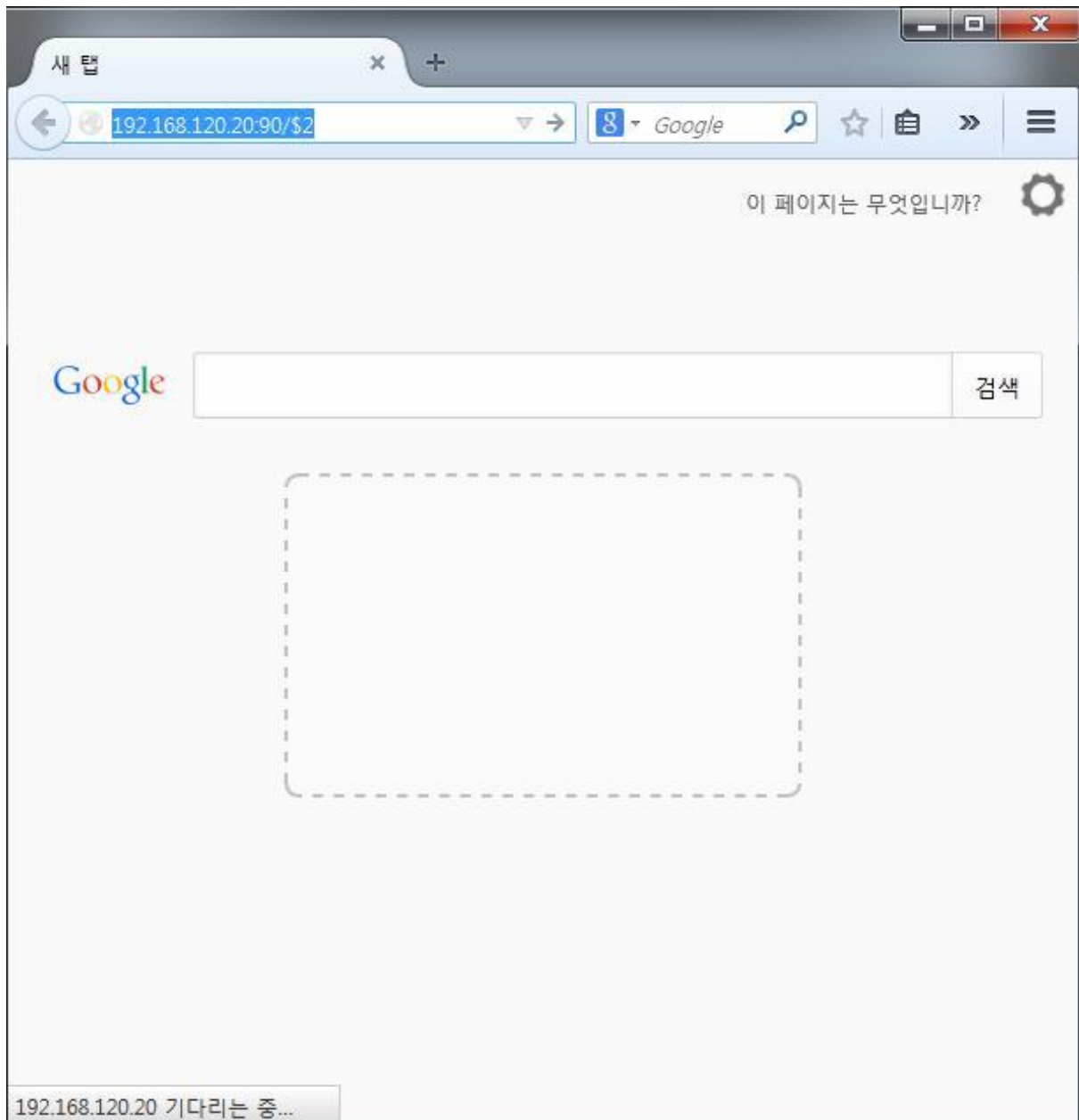
```
    }  
  }  
}  
// give the web browser time to receive the data  
delay(1);  
// close the connection:  
client.stop();  
}  
}
```

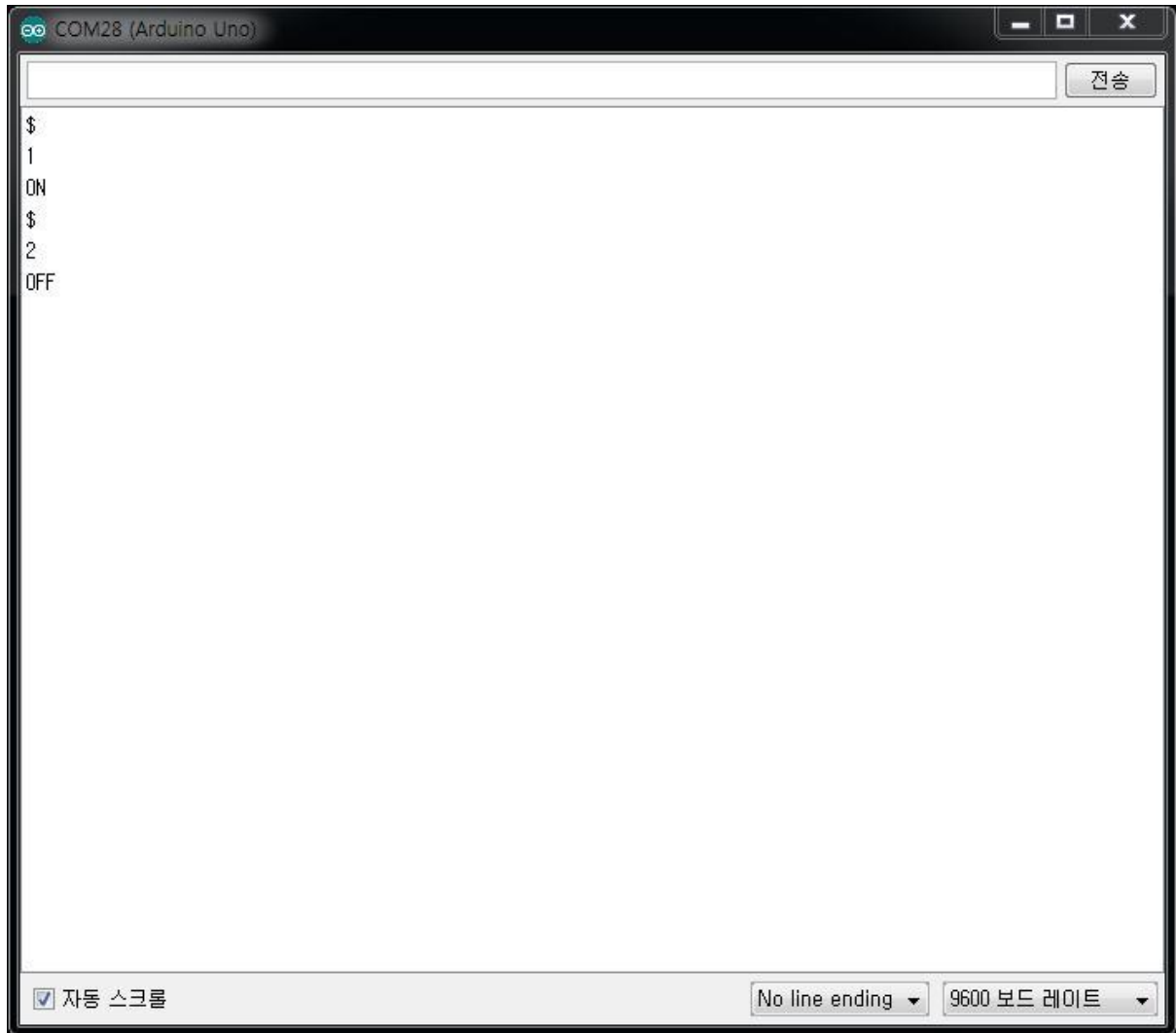
업로드가 마친 후에, 웹페이지를 열고 주소창에 자신의 IP와 포트번호(90)을 사용하여 다음과 같이 입력합니다.

192.168.120.20:90/\$1

위에서 \$1이라는 것은 소스코드의 본문에 나왔듯이, 디지털 2번 핀에 연결된 LED에 HIGH 값을 출력하여 LED를 켤 수 있게 하며, \$2를 넣게 되면 LOW 값을 출력하여 LED를 끄게 합니다. 각각의 주소를 입력한 후에 시리얼 모니터링을 통해서 웹페이지를 통해 원격으로 입력한 값이 로컬 시스템인 아두이노에 시리얼 값으로 전송이 되는지 확인하기 위해서 시리얼 모니터링을 통해 체크해보았습니다.







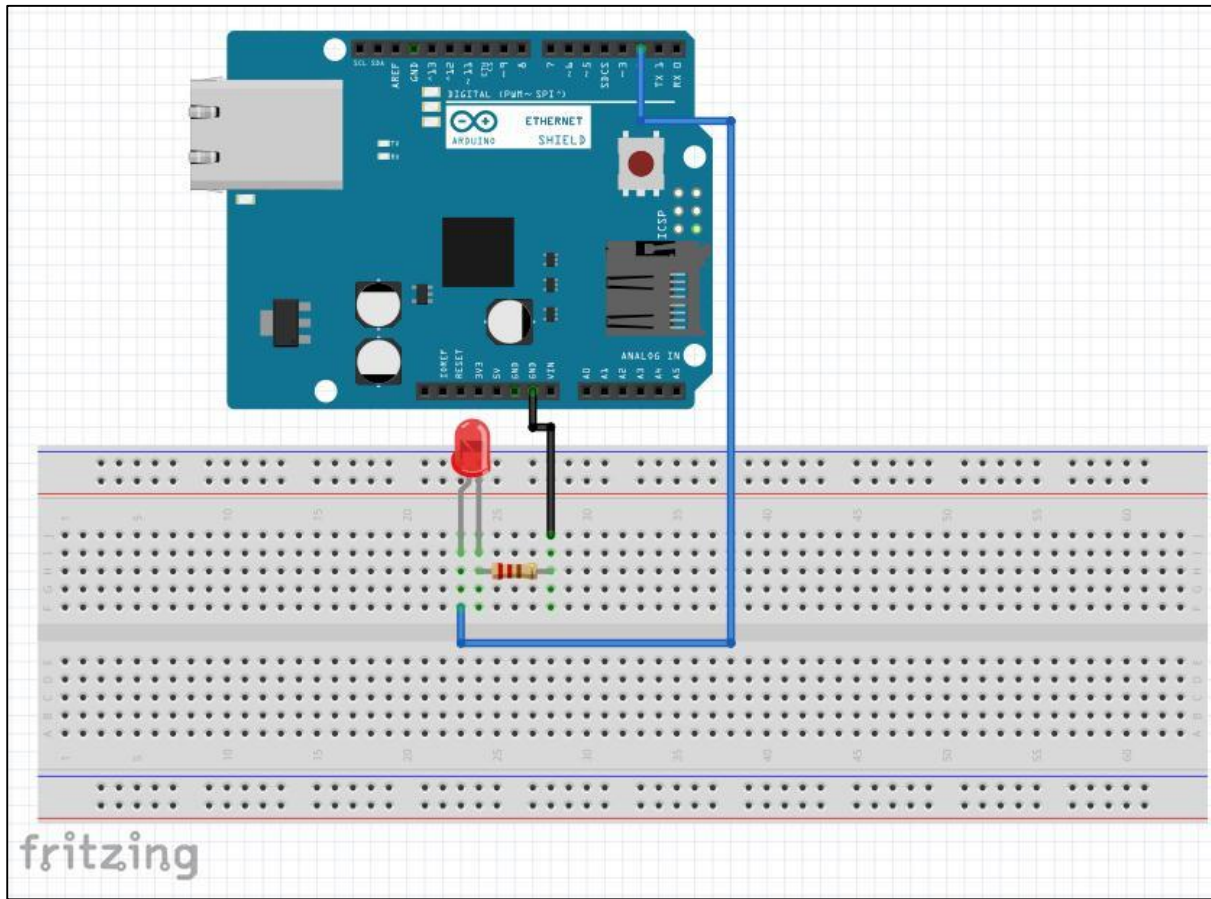
위의 시리얼 모니터링 창은, 아두이노와 컴퓨터간에 USB 케이블을 통해서 신호가 오가는 것을 보여주는 것입니다. 따라서, 웹페이지에서 보낸 값이 컴퓨터를 통해 아두이노로 전송되고 있는 것을 확인할 수 있음을 보았습니다.

(2) HTML로 LED 켜고 끄기

HTML코딩을 하여 체크박스로 LED를 ON / OFF할 수 있는 예제입니다. 위의 HTML 코딩 없이 LED를 켜고 끄는 데모와 같은 회로로 구성되며 소스코드만 변경하게 됩니다.

준비물: 아두이노 우노, 이더넷 쉴드, LAN 케이블, USB 케이블, 점퍼와이어, 빨간색 LED, 220옴 저항, 브레드보드

회로도



아두이노 소스코드

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //mac address
IPAddress ip(192, 168, 120, 20) ;
EthernetServer server(90) ;

String HTTP_req;          // stores the HTTP request
boolean LED_status = false; // state of LED, off by default

void setup()
{
  Ethernet.begin(mac, ip); // initialize Ethernet device
  server.begin();          // start to listen for clients
}
```

```

Serial.begin(9600);      // for diagnostics
pinMode(2, OUTPUT);      // LED on pin 2
}

void loop()
{
  EthernetClient client = server.available(); // try to get client
  if (client) { // got client?
    //    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) { // client data available to read
        char c = client.read(); // read 1 byte (character) from client
        HTTP_req += c; // save the HTTP request 1 char at a time
        // last line of client request is blank and ends with \n
        // respond to client only after last line received
        if (c == '\n') {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println();
          // send web page
          client.println("<!DOCTYPE html>");
          client.println("<html>");
          client.println("<head>");
          client.println("<title>Arduino LED Control</title>");
          client.println("</head>");
          client.println("<body>");
          client.println("<h1>LED</h1>");
          client.println("<p>Click to switch LED on and off.</p>");
          client.println("<form method='get'>");
          ProcessCheckbox(client);
          client.println("</form>");
          client.println("</body>");
          client.println("</html>");
        }
      }
    }
  }
}

```

```

        Serial.println(HTTP_req) ;
        HTTP_req = "";    // finished with request, empty string
        break;
    }
    } // end if (client.available())
} // end while (client.connected())
delay(1);    // give the web browser time to receive the data
client.stop(); // close the connection
} // end if (client)
}

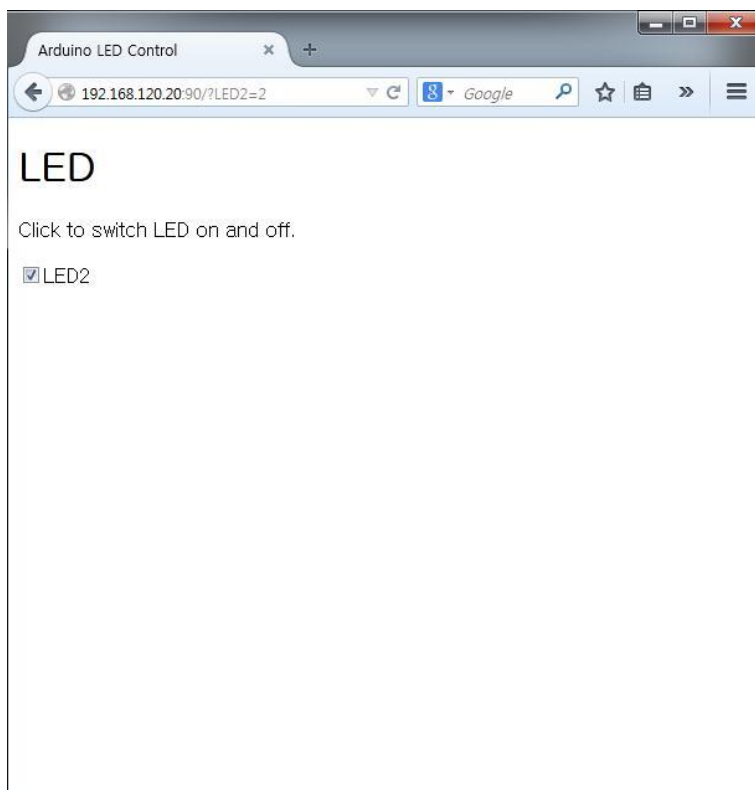
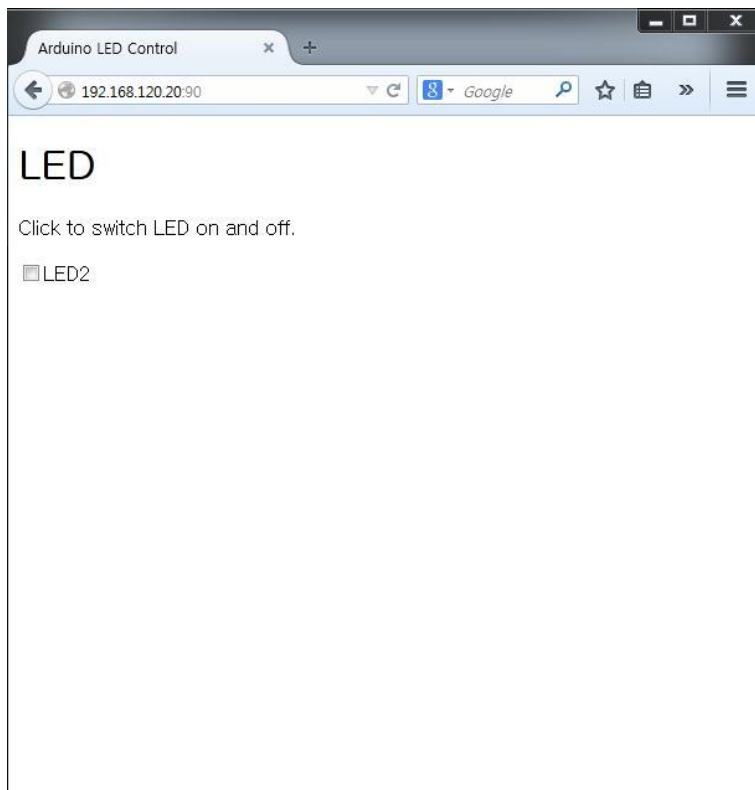
// switch LED and send back HTML for LED checkbox
void ProcessCheckbox(EthernetClient cl)
{
    if (HTTP_req.indexOf("LED2=2") != -1) LED_status = true ;
    else LED_status = false ;

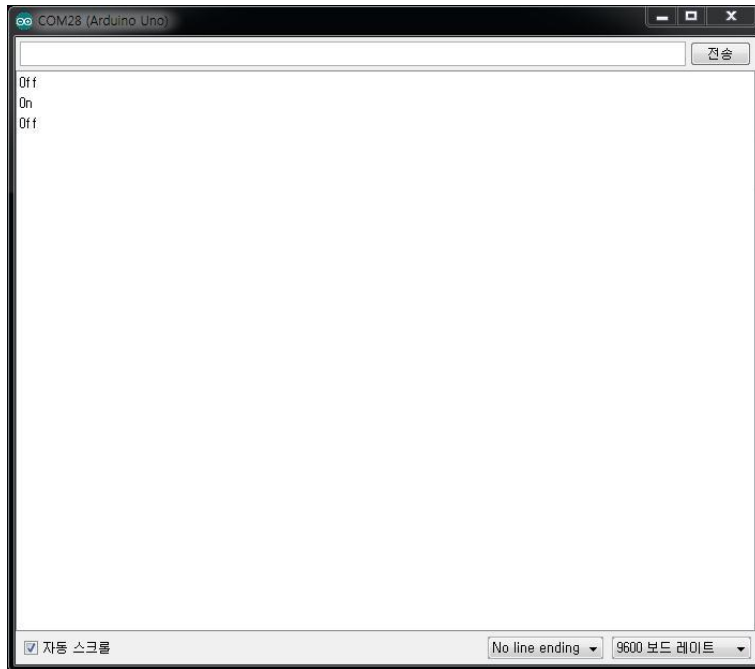
    if (LED_status) {    // switch LED on
        digitalWrite(2, HIGH);
        Serial.println("On") ;
        // checkbox is checked
        cl.println("<input type=W\"checkboxW\" name=W\"LED2W\" value=W\"2W\" W
        onclick=W\"submit();W\" checked>LED2");
    }
    else {                // switch LED off
        digitalWrite(2, LOW);
        Serial.println("Off") ;
        // checkbox is unchecked
        cl.println("<input type=W\"checkboxW\" name=W\"LED2W\" value=W\"2W\" W
        onclick=W\"submit();W\">LED2");
    }
}
}

```

프로그램이 업로드 된 후에는 웹페이지를 열어 사용하고 있는 IP와 포트번호 (90)을 주소창에 넣습니다. 그러면 다음과 같이 LED2에 체크박스가 생성된 것을 확인할 수 있으며 체크박스를 통해

LED를 켜고 끌 수 있습니다. 마찬가지로 시리얼 모니터링을 통해서 웹페이지의 컨트롤이 아두이노에 잘 전송되었는지 확인해 보았습니다.



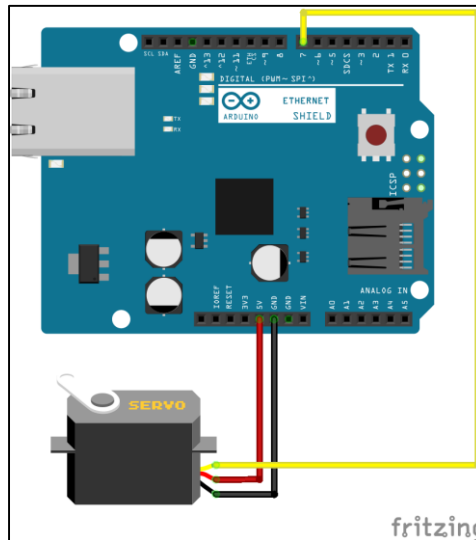


6) 서보모터(SG90) 제어하기

텍스트박스로 원하는 서보모터의 각도를 입력하고 send를 누르면 서보모터가 그 각도로 동작하는 예제입니다. 0 ~ 180도 사이에서 작동하며 이 수치는 서보모터 종류에 따라 달라질 수 있습니다. 본 데모에서는 SG90 소형 서보 모터를 사용했습니다.

준비물: 아두이노 우노, 이더넷 쉴드, SG90, LAN 케이블, USB 케이블, 점퍼와이어

회로도: 회로도 7번에 서보모터의 SIG핀이 연결되어 있습니다. 다수의 모터를 제어하는 경우에는 아두이노에서 전원을 공급하는 경우, 일시적인 과전류로 인해서 아두이노가 리셋되거나 손상될 우려가 있기 때문에 독립 전원을 사용하시길 바랍니다. 이와 관련해서는 메카솔루션 블로그 혹은 아두이노 종결키트 매뉴얼을 참고하시기 바랍니다.



아두이노 소스코드

```
//zoomkat 4-1-12
//open serial monitor to see what the arduino receives
//use the ₩ slash to escape the " in the html, or use ' instead of
//for use with W5100 based ethernet shields

#include <SPI.h>
#include <Ethernet.h>
#include <Servo.h>

Servo sg90 ; // create servo object to control a servo

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED } ; //mac address
IPAddress ip(192, 168, 120, 20) ;
EthernetServer server(90) ;

String request = "" ;

////////////////////

void setup() {
  //start Ethernet
```

```

Ethernet.begin(mac, ip);
server.begin();
sg90.attach(7); //the pin for the servo control
Serial.begin(9600);
}

void loop() {
  // Create a client connection
  EthernetClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        request += c;
        if (c == '\n') {

          client.println("HTTP/1.1 200 OK"); //send new page
          client.println("Content-Type: text/html");
          client.println();

          client.println("<!DOCTYPE html>");
          client.println("<HTML>");
          client.println("<HEAD>");
          client.println("<TITLE>Arduino Servo</TITLE>");
          client.println("</HEAD>");
          client.println("<BODY>");
          client.println("<form method=W\"getW\">");
          client.println("Servo value = <input type=W\"textW\" name=W\"valW\">");
          client.println("<input type=W\"submitW\" value=W\"sendW\">");
          client.println("</form>");
          client.println("</BODY>");
          client.println("</HTML>");

          delay(1);
          //stopping client

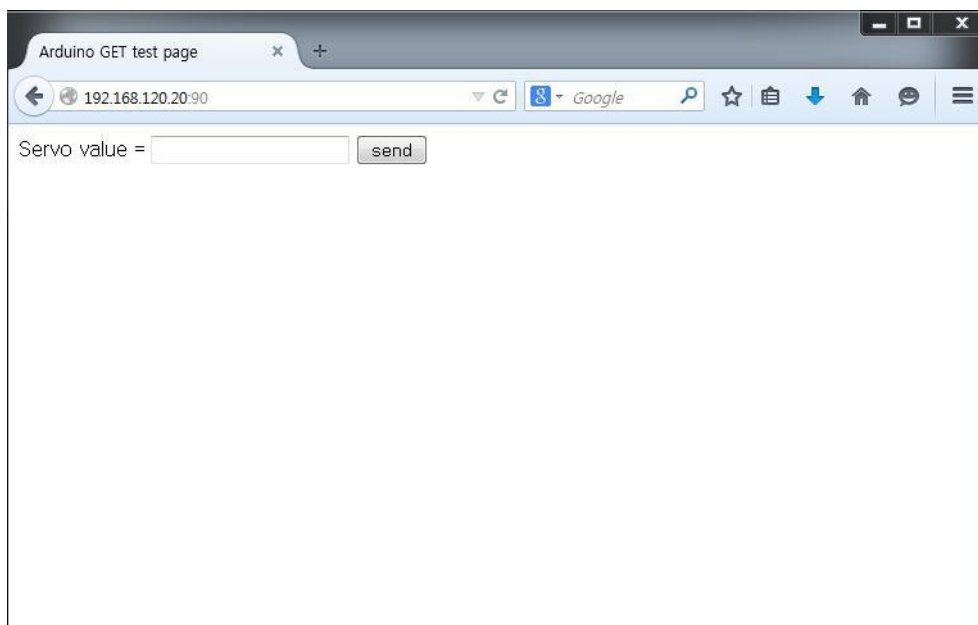
```

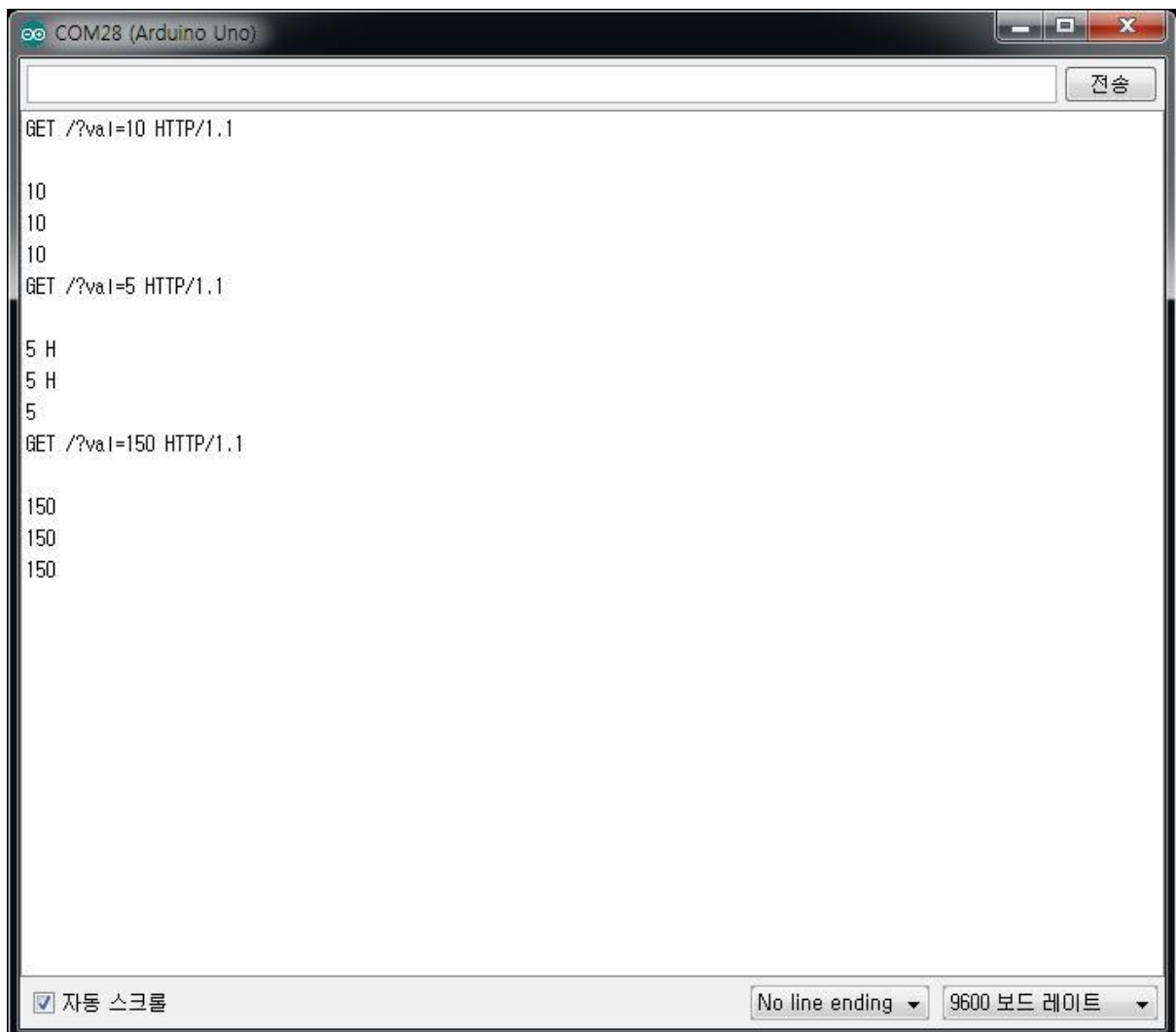
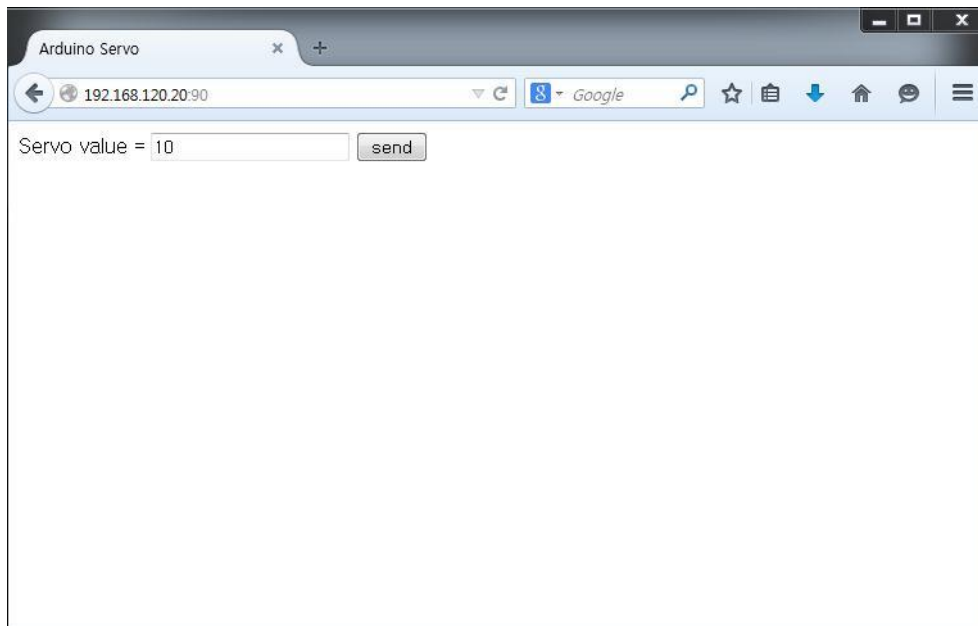
```

client.stop();

//clearing string for next read
Serial.println(request) ;
if (request.indexOf("val=") > -1) {
    request.remove(0, 10) ;
    request.remove(3) ;
    Serial.println(request) ;
    long servo_val = request.toInt() ;
    Serial.println(servo_val, DEC) ;
    sg90.write(servo_val) ;
}
request = "";
}
}
}
}
}
}

```



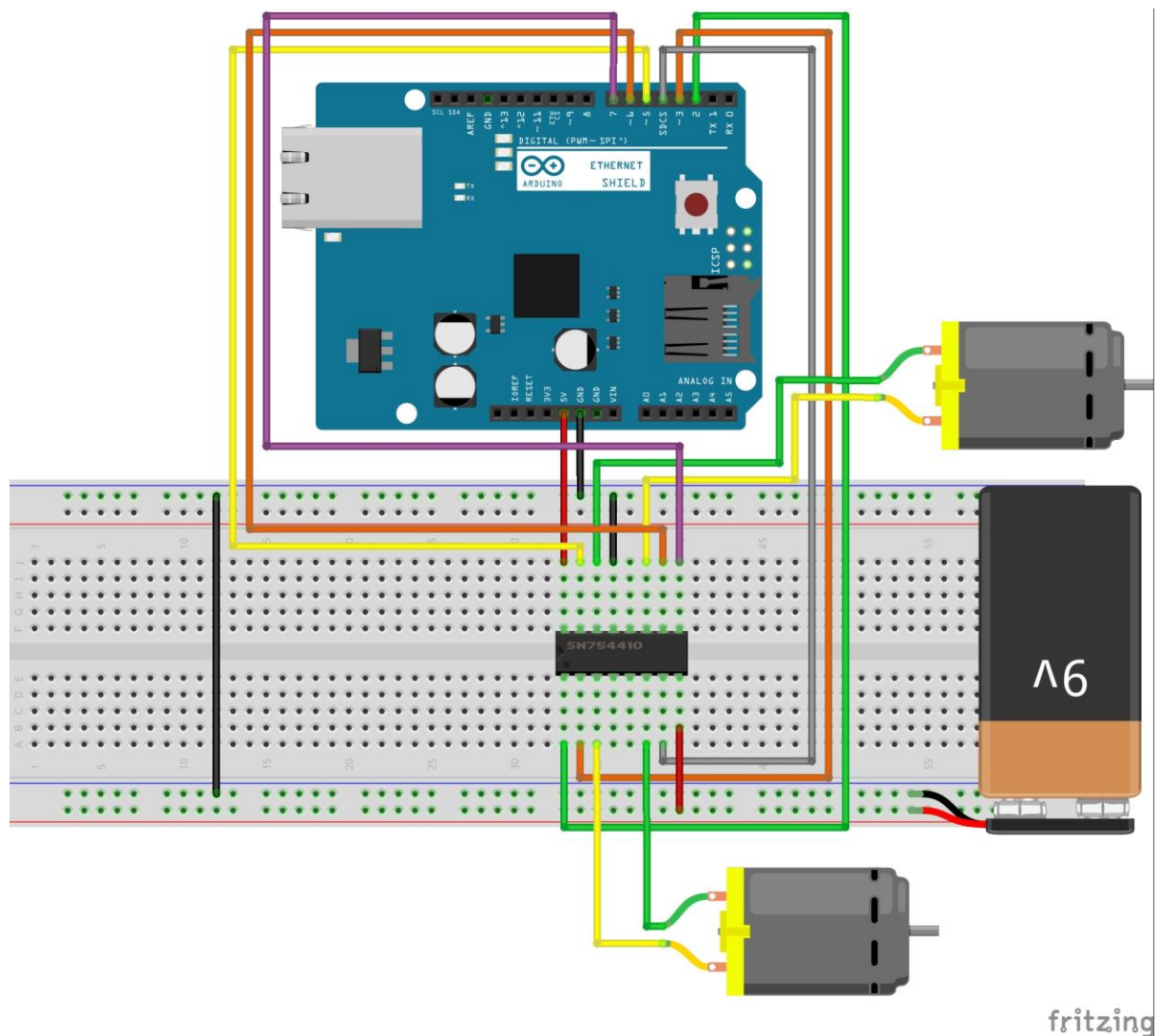


7) DC모터 제어하기

각 글자를 클릭할 때 마다 주소창의 변수가 바뀝니다. 각 링크는 1, 2, 3, 4, 5, 6이라는 값을 가지고 있으며 이를 아두이노에서 읽어 그에 해당하는 동작을 실행합니다. SN754410 IC를 사용하였으며 5V모터를 사용하였기에 별도의 외부 전원 없이 아두이노의 5V를 바로 사용하였습니다.

준비물: 아두이노 우노, 이더넷 쉴드, 브레드보드, 점퍼와이어, LAN 케이블, USB 케이블, SN754410 H-Bridge Motor Driver IC, DC모터 2개

회로도



아두이노 소스코드

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //mac address
IPAddress ip(192, 168, 120, 20) ;
EthernetServer server(90) ;

int ena = 2 ;
int in1 = 3 ;
int in2 = 4 ;
int in3 = 5 ;
int in4 = 6 ;
int enb = 7 ;

String request = "" ;

void setup() {
    // put your setup code here, to run once:
    Ethernet.begin(mac, ip) ;
    server.begin() ;
    Serial.begin(9600) ;
    for (int i = 2; i <= 7; i++) pinMode(i, OUTPUT) ;
    digitalWrite(ena, HIGH) ;
    digitalWrite(enb, HIGH) ;
}

void loop()
{
    EthernetClient client = server.available();
    if (client) {
        while (client.connected()) {
            if (client.available()) {
                char c = client.read();
```

```

request += c; // save the HTTP request 1 char at a time
if (c == '\n') {
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println();
    // send web page
    client.println("<!DOCTYPE html>");
    client.println("<html>");
    client.println("<head>");
    client.println("<title>Arduino motor</title>");
    client.println("</head>");
    client.println("<body>");
    client.println("<h1>DC MOTOR</h1>");
    client.println("<form method=W\"getW\">");

    client.println("<a href=W\"/?1W\">L Motor CW</a>&nbsp;&nbsp;&nbsp;");
    client.println("<a href=W\"/?4W\">R Motor CW</a><br />");
    client.println("<a href=W\"/?2W\">L Motor STOP</a>");
    client.println("<a href=W\"/?5W\">R Motor STOP</a><br />");
    client.println("<a href=W\"/?3W\">L Motor CCW</a>&nbsp;&nbsp;&nbsp;");
    client.println("<a href=W\"/?6W\">R Motor CCW</a><br />");

    client.println("</form>");
    client.println("</body>");
    client.println("</html>");
    delay(1); // give the web browser time to receive the data
    client.stop(); // close the connection
    Serial.println(request);
    if (request.indexOf('?') > -1) {
        request.remove(0, 6);
        request.remove(1);
        switch (request.toInt()) {
            case 0 :
                lmotor_stop();
                rmotor_stop();

```



```

        break ;
    case 1 :
        lmotor_front() ;
        break ;
    case 2 :
        lmotor_stop() ;
        break ;
    case 3 :
        lmotor_rear() ;
        break ;
    case 4 :
        rmotor_front() ;
        break ;
    case 5 :
        rmotor_stop() ;
        break ;
    case 6 :
        rmotor_rear() ;
        break ;
    }

    Serial.println(request) ;
}
request = "";    // finished with request, empty string
}
}

// every line of text received from the client ends with \r\n
} // end if (client.available())
} // end while (client.connected())
} // end if (client)

```

```

void lmotor_front()

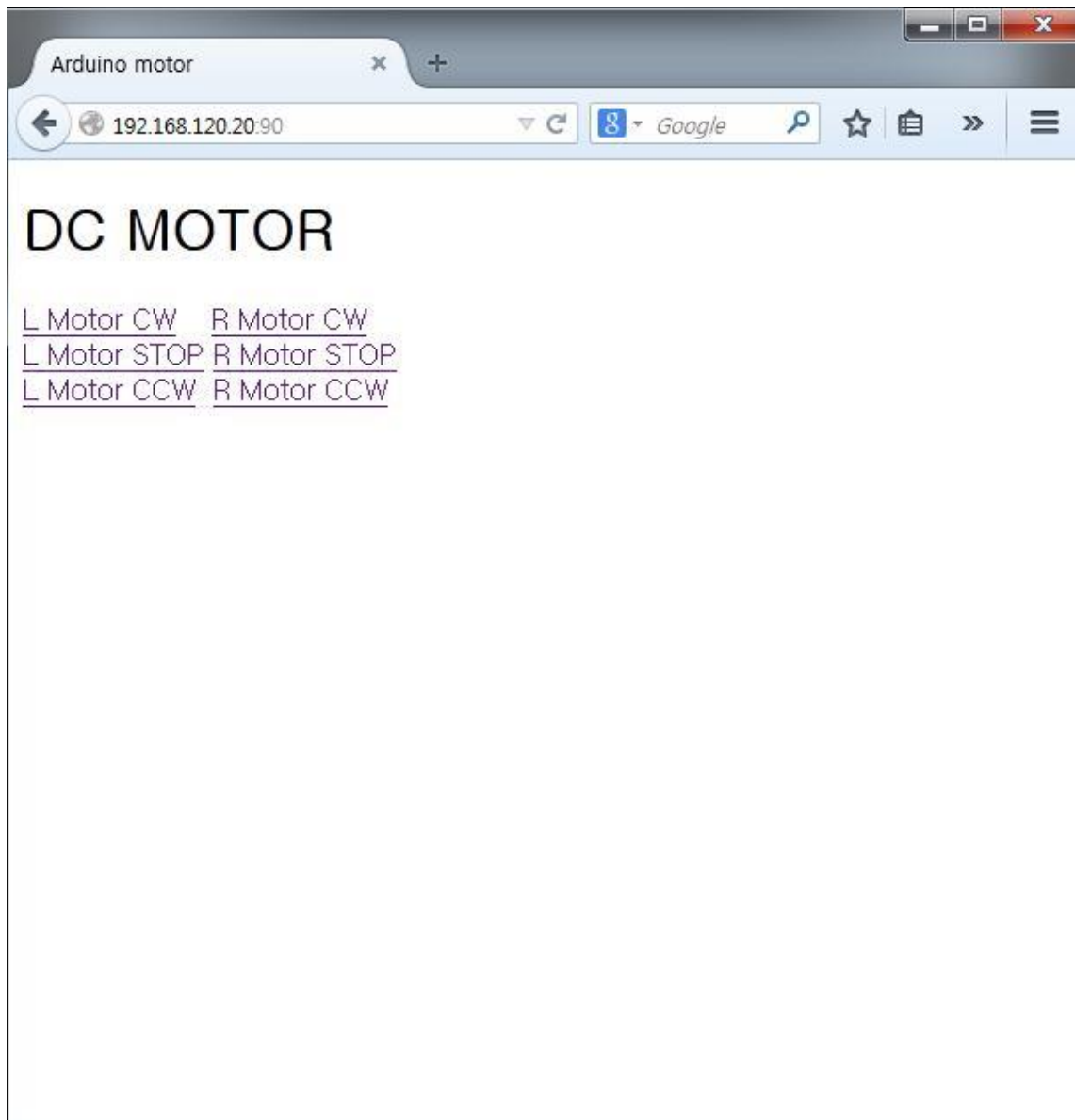
```

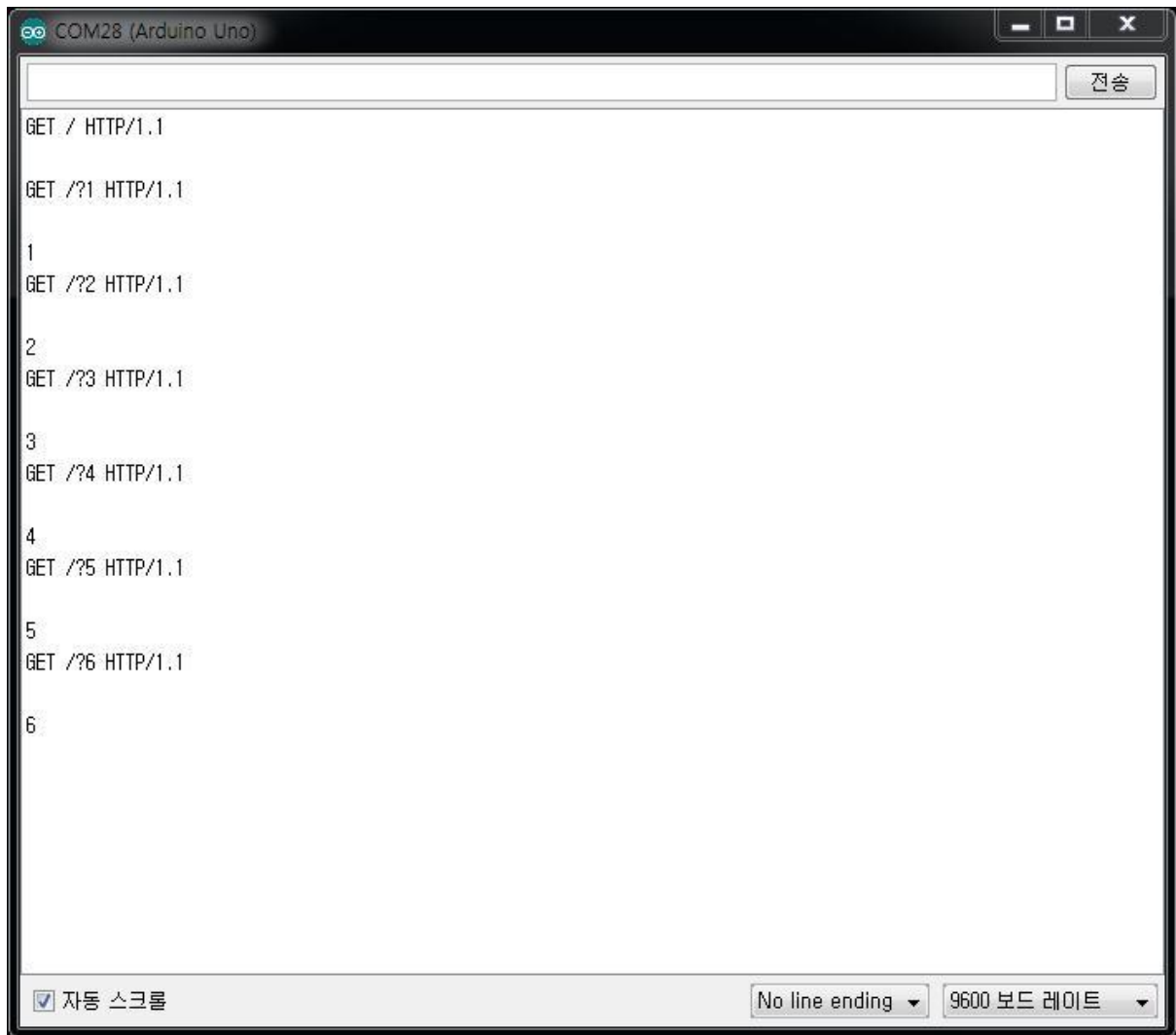
```

{
    digitalWrite(in1, HIGH) ;
    digitalWrite(in2, LOW) ;
}
void lmotor_rear()
{
    digitalWrite(in1, LOW) ;
    digitalWrite(in2, HIGH) ;
}
void lmotor_stop()
{
    digitalWrite(in1, LOW) ;
    digitalWrite(in2, LOW) ;
}
void rmotor_front()
{
    digitalWrite(in3, HIGH) ;
    digitalWrite(in4, LOW) ;
}
void rmotor_rear()
{
    digitalWrite(in3, LOW) ;
    digitalWrite(in4, HIGH) ;
}
void rmotor_stop()
{
    digitalWrite(in3, LOW) ;
    digitalWrite(in4, LOW) ;
}

```

업로드를 마친 후에 웹페이지로 접속하게 되면 다음과 같이 두 개의 DC 모터의 방향을 제어할 수 있는 텍스트가 출력된 것을 확인할 수 있으며 클릭하여서 테스트할 수 있습니다. 아두이노로 모터를 구동할 신호가 잘 전송되었는지 확인하기 위해서 시리얼 모니터링을 사용하였습니다.



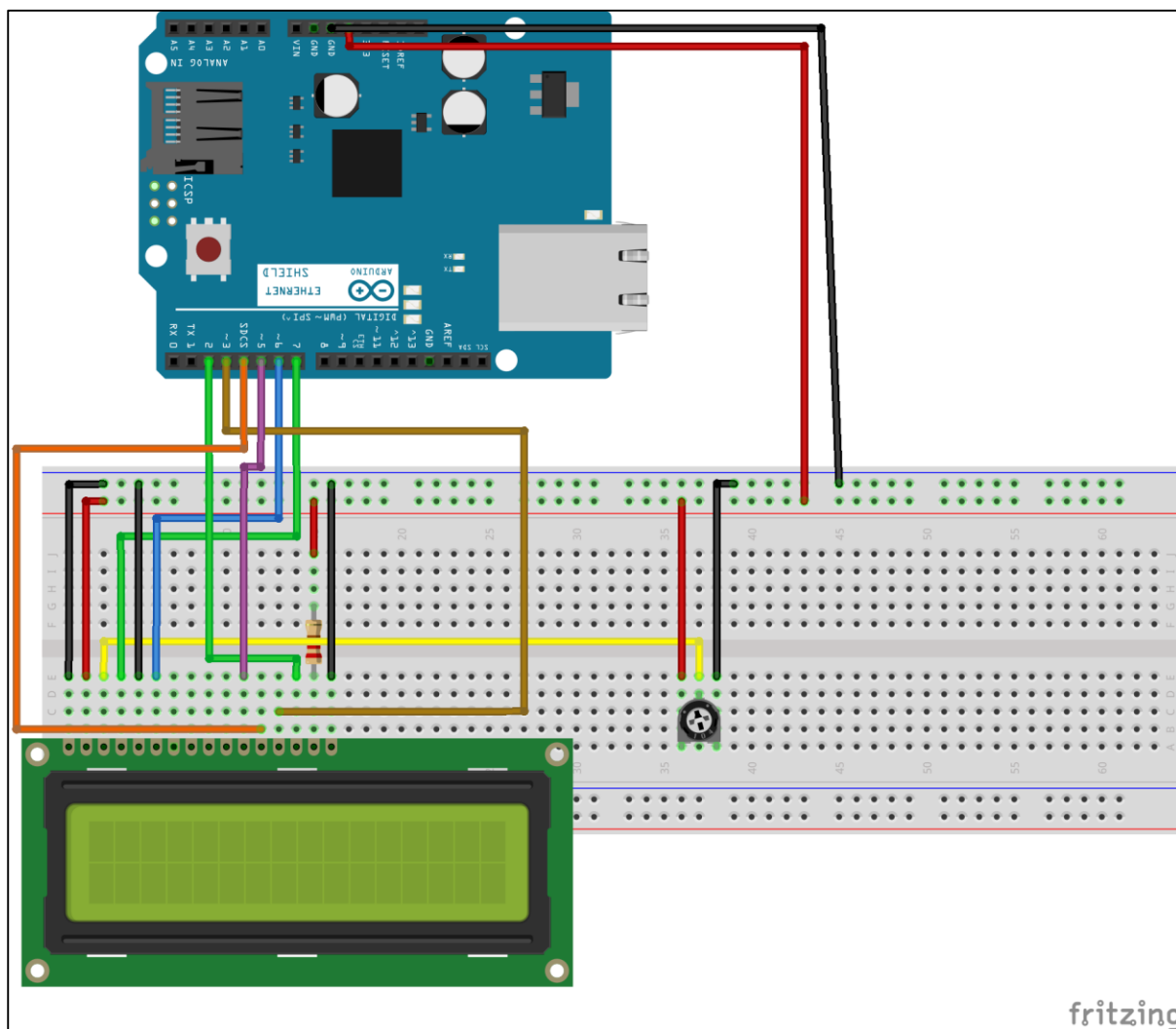


8) 입력받은 글자 LCD에 출력하기

웹페이지에서 첫째 줄과 두 번째 줄에 해당하는 텍스트를 입력할 수 있습니다. 영어, 숫자만 입력이 가능하며 최대 16자까지입니다. 초과할 시 값은 버려집니다. 가변저항으로 글자의 밝기를 조절할 수 있습니다.

준비물: 아두이노 우노, 이더넷 실드, 16x2 캐릭터 LCD, LAN 케이블, USB 케이블, 점퍼와이어, 220옴 저항, 가변저항

회로도



아두이노 소스코드

```
#include <SPI.h>

#include <Ethernet.h>

#include <LiquidCrystal.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //mac address
IPAddress ip(192, 168, 120, 20) ;

EthernetServer server(90) ;

LiquidCrystal lcd(7, 6, 5, 4, 3, 2) ;

boolean lcd_flag = false ;
```

```

String request = "" ;

void setup() {
    // put your setup code here, to run once:
    Ethernet.begin(mac, ip) ;
    server.begin() ;
    lcd.begin(16, 2) ;
    Serial.begin(9600) ;
}

void loop() {
    // put your main code here, to run repeatedly:
    EthernetClient client = server.available() ;
    if (client) {
        lcd.begin(16, 2) ;
        while (client.connected()) {
            if (client.available()) {
                char c = client.read() ;
                request += c ;
                if (c == '\n') {
                    client.println("HTTP/1.1 200 OK"); //send new page
                    client.println("Content-Type: text/html");
                    client.println();

                    client.println("<!DOCTYPE html>");
                    client.println("<HTML>");
                    client.println("<HEAD>");
                    client.println("<TITLE>Arduino LCD</TITLE>");
                    client.println("</HEAD>");
                    client.println("<BODY>");
                    client.println("<form method=W\"getW\">") ;
                    client.println("LCD  text  up  =  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input  type=W\"textW\"");
                    client.println("name=W\"val1W\"> max 16<br />") ;
                    client.println("LCD text down = <input type=W\"textW\" name=W\"val2W\"> max 16") ;
                }
            }
        }
    }
}

```

```

client.print("<br />");
for (int tem = 0; tem < 17; tem++) client.print("&nbsp;");
client.println("&nbsp;");
client.println("<input type=W\"submitW\" value=W\"sendW\">");
client.println("</form>");
client.println("</BODY>");
client.println("</HTML>");

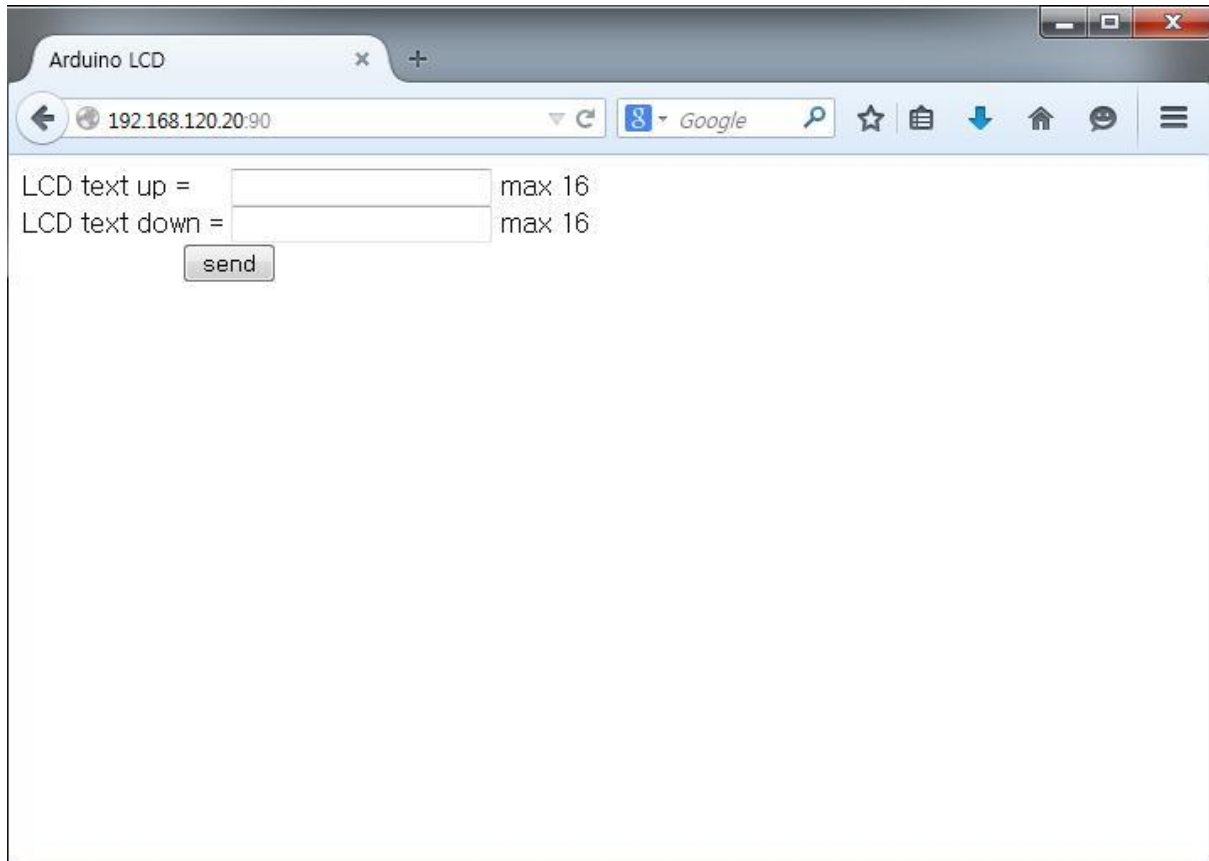
delay(1);
client.stop();

Serial.println(request);
String req_back = request;
if (request.indexOf("val1") && request.indexOf("val2")) {
    request.remove(0, 11);
    Serial.println(request);
    for (int tem = 16; tem >= 0; tem--) {
        request.remove(tem);
        if (request.lastIndexOf("&") == -1) break;
    }
    Serial.println(request);
    lcd.setCursor(0, 0);
    lcd.print(request);
    request = req_back;
    request.remove(0, 11);
    int tem = 0;
    int count = 0;
    while (1) {
        request.remove(0, tem);
        Serial.println(request);
        if (request.startsWith("&val2")) {
            Serial.print("start  :");
            Serial.println(request);
            req_back = request;
            Serial.print("req_back  :");

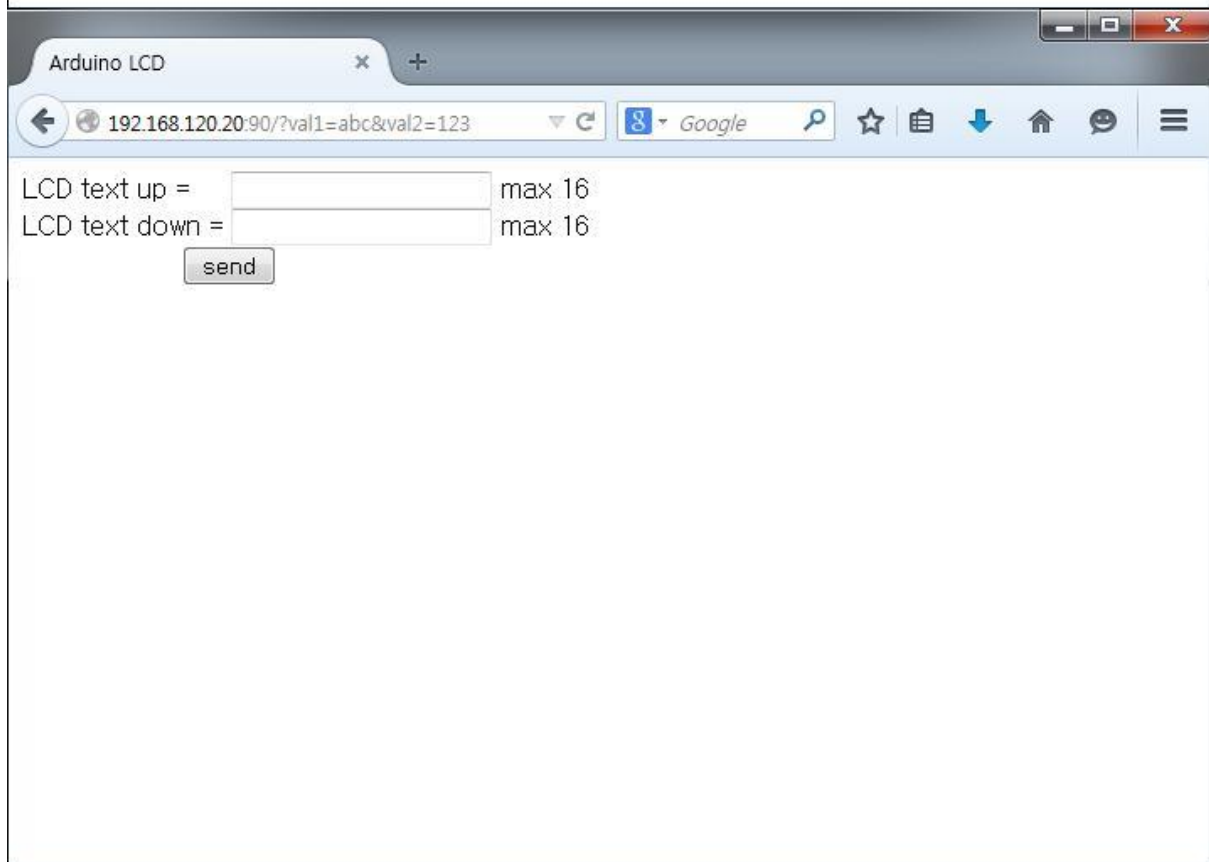
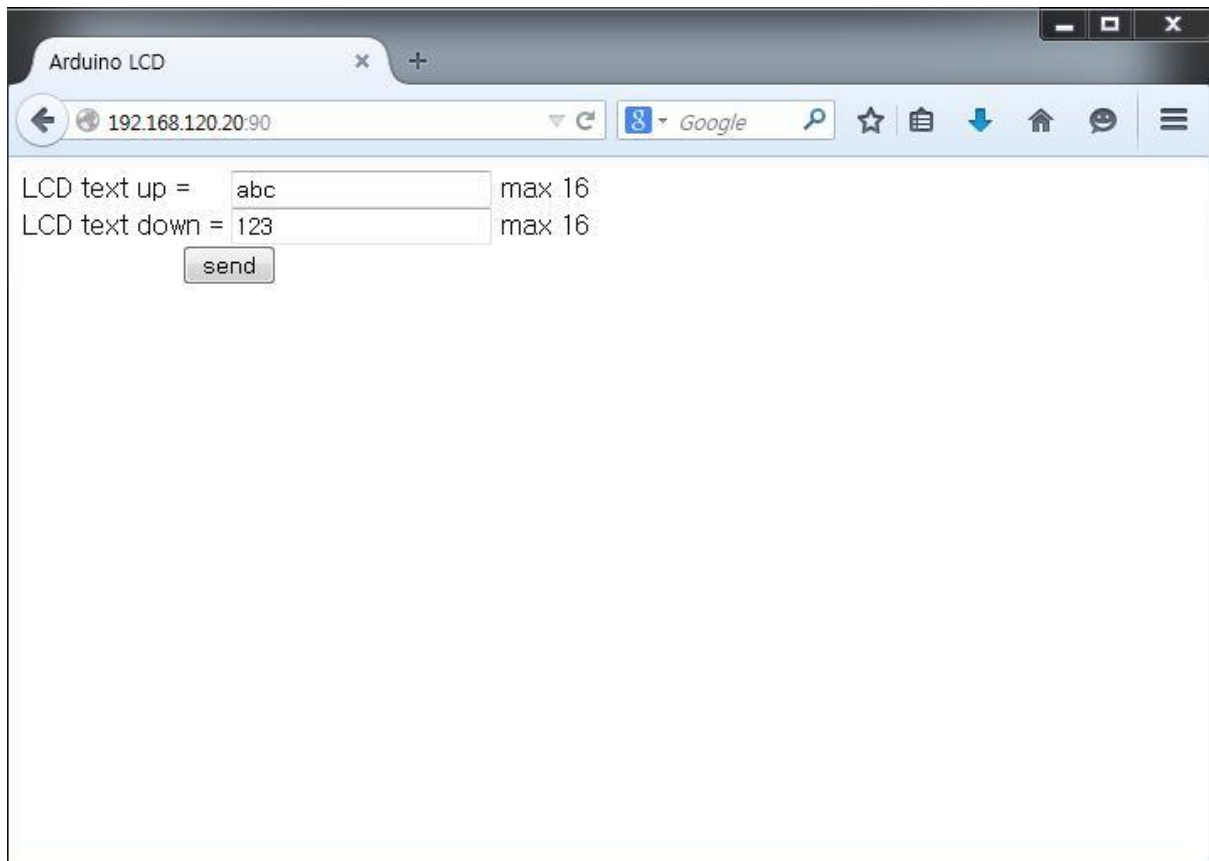
```

```
Serial.println(req_back) ;  
for (int i = 14; i < 30; i++) {  
    request.remove(i) ;  
    Serial.print("req  : ") ;  
    Serial.println(request) ;  
    if (request.lastIndexOf("HTTP/1.1") != -1) {  
        request.remove(i - 8) ;  
        break ;  
    }  
    request = req_back ;  
}  
request.remove(0, 6) ;  
Serial.print("except  : ") ;  
Serial.println(request) ;  
lcd.setCursor(0, 1) ;  
lcd.print(request) ;  
break ;  
}  
tem = 1 ;  
count++ ;  
if (request.length() == 0) {  
    lcd.setCursor(0,0) ;  
    lcd.print("                ");  
    break ;  
}  
}  
}  
request = "" ;  
}  
}  
}  
}
```


는 텍스트와 최대 16개의 문자를 넣을 수 있는 텍스트 박스가 있습니다. 한글 지원은 되지 않기 때문에 알파벳 문자 및 숫자, 기호를 사용하실 수 있으며, 입력한 후에 send 버튼을 누르면 문자가 아두이노에 전송되고 LCD 모니터를 통해서 출력됨을 확인할 수 있습니다.



The screenshot shows a web browser window with the title "Arduino LCD". The address bar displays the IP address "192.168.120.20:90". The page content includes two text input fields. The first field is labeled "LCD text up =" and has "max 16" to its right. The second field is labeled "LCD text down =" and also has "max 16" to its right. Below these fields is a button labeled "send".





3. 프로세싱과 연동하기

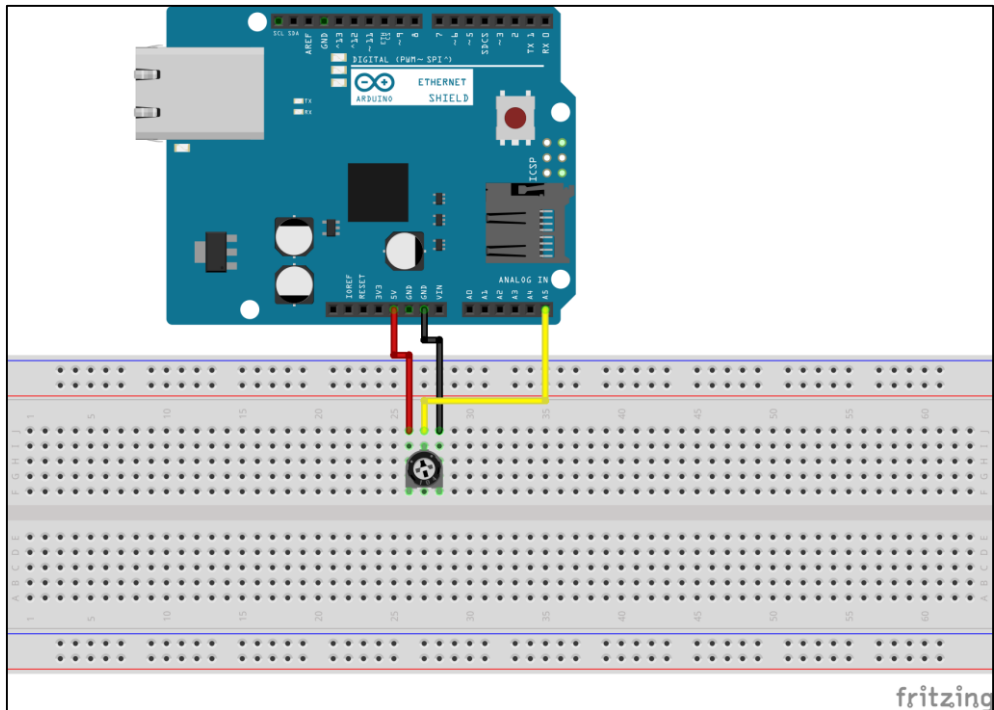
프로그래밍 언어, 개발환경, 온라인 커뮤니티를 이르는 말입니다. 프로그래밍 능력을 기르기 위해 2001년 개발되었으며 OpenGL이 통합되어 있습니다. 아두이노보단 적지만 100가지 이상의 라이브러리가 있으며 무료 그리고 오픈소스입니다. 아래 링크에서 다운받을 수 있습니다.

<https://processing.org/download/>

1) 아날로그 값 그래프로 보기

아두이노의 서버와 프로세싱을 연결하여 아두이노에서 아날로그 값을 보냅니다. 프로세싱에선 받은 아날로그 값을 그래프로 표현해줍니다. 여기서 가변저항을 사용하여 그래프가 어떻게 변하는지 알아볼 수 있습니다.

준비물: 아두이노 우노, 이더넷 실드, LAN 케이블, USB 케이블, 점퍼 와이어, 브레드보드, 가변저항 회로도



아두이노 소스코드

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //mac address
IPAddress ip(192, 168, 120, 20);
EthernetServer server(23);

void setup() {
  // initialize the ethernet device
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.begin(9600);
  Serial.print("Chat server address:");
  Serial.println(Ethernet.localIP());
}

void loop() {
  // wait for a new client:
  EthernetClient client = server.available();
```

```

if (client) {
    if (client.available() > 0) {
        char c = client.read() ;
        if (c == 'c') {
            int value = analogRead(5) ;
            client.write((value >> 8) & 0xFF) ;
            client.write(value & 0xFF) ;
        }
    }
}
}
}

```

프로세싱 소스코드

```

import processing.net.*;
Client myClient;

int resolv = 3 ;
int x_po = resolv + 11 ;
long line = 0 ;

void setup() {
    // This example will not run if you haven't
    // previously started a server on this port.
    myClient = new Client(this, "192.168.120.20", 23);
    size(1600, 600) ;
    background(0) ;
}

void draw() {
    // Change the background if the mouse is pressed
    myClient.write('c') ;
    if (myClient.available() > 1) {
        int val1 = myClient.read() << 8 ;
        int val2 = myClient.read() ;
    }
}

```

```
int val = val1 | val2 ;
stroke(255, 255, 255) ;
val = val * 512 / 1023 / 1 ;
line(x_po - resolv, line, x_po, (height - val - 44)) ;
line = (height - val - 44) ;
x_po += resolv ;
if (x_po > width) {
    background(0) ;
    x_po = resolv + 11 ;
}
}
}
```

프로세싱 실행 결과



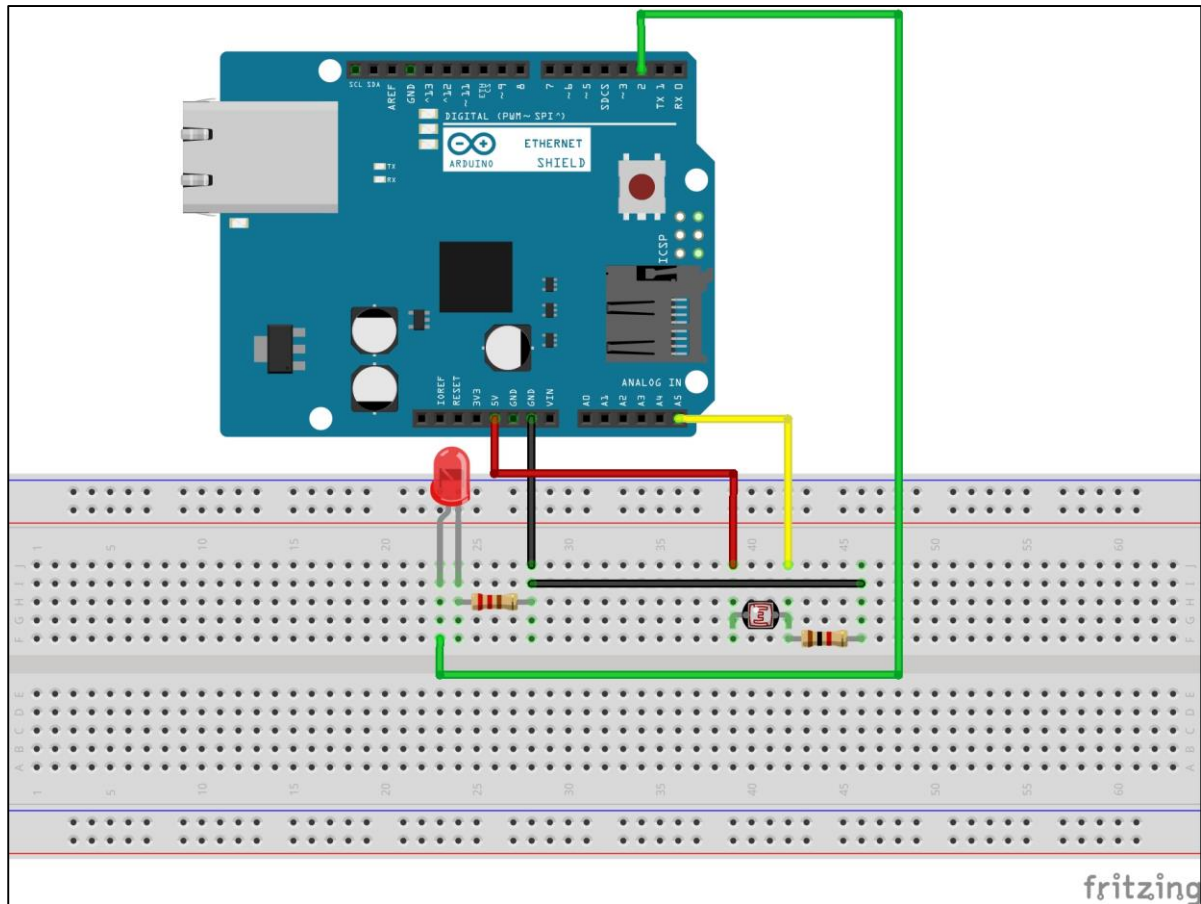
2) 밝기에 따라 변하는 박스

CdS와 LED를 복합적으로 사용하는 예제입니다. CdS는 밝기에 따라 저항값이 변하는 소자로 LED를 켜게 되면 그 빛에 영향을 받아 저항값이 변하게 됩니다. 또 햇빛의 영향도 같이 받기에 LED를 껐을 경우에도 현재 밝기에 따라 값이 변하게 됩니다. 여기서 가변저항값이 근거리에서 LED를 켜었을 때 677, 손으로 가렸을 때 68이 나왔기에 677일 경우 박스의 명도는 255, 68일 때는 0이 되도록 연산을 해 주었습니다.

준비물: 아두이노 우노, 이더넷 실드, LAN 케이블, USB 케이블, 점퍼와이어, LED, 220옴 저항, 조도

센서(CdS), 1k옴 저항

회로도



아두이노 소스코드

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //mac address
IPAddress ip(192, 168, 120, 20);
EthernetServer server(23);

int led = 2;

void setup() {
  // initialize the ethernet device
```

```

pinMode(led, OUTPUT) ;
digitalWrite(led, LOW) ;
Ethernet.begin(mac, ip);
server.begin();
Serial.begin(9600);
Serial.println(Ethernet.localIP());
}

void loop() {
    // wait for a new client:
    EthernetClient client = server.available();
    if (client) {
        if (client.available() > 0) {
            char c = client.read() ;
            if (c == 'c') {
                int value = analogRead(5) ;
                client.write((value >> 8) & 0xFF) ;
                client.write(value & 0xFF) ;
            }
            else if (c == 'n') {
                digitalWrite(led, HIGH) ;
            }
            else if (c == 'f') {
                digitalWrite(led, LOW) ;
            }
        }
    }
}

```

프로세싱 소스코드

```

import processing.net.*;
Client myClient;

boolean flag_p = false ;

```



```

int resolv = 3 ;
int x_po = resolv + 11 ;
long line = 0 ;
int mou_sta = 0 ;

void setup() {
    myClient = new Client(this, "192.168.120.20", 23);
    size(600, 600) ;
    background(0) ;
    fill(255) ;
    rect(0, 100, width, height) ;
    fill(125) ;
    rect(50, 25, 200, 50) ;
    rect(350, 25, 200, 50) ;
    textAlign(CENTER, CENTER) ;
    textSize(30) ;
    fill(255, 255, 0) ;
    text("LED ON", 150, 50) ;
    text("LED OFF", 450, 50) ;
}

void draw() {
    myClient.write('c') ;
    if (myClient.available() > 1) {
        int val1 = myClient.read() << 8 ;
        int val2 = myClient.read() ;
        int val = val1 | val2 ;
        /*
            Highest analog value is 677 and lowest is 68
            609 = highest value - lowest value
        */
        int bright = (val - 68) * 255 / 609 / 1 ;
        if (bright > 255) bright = 255 ;
        else if (bright < 0) bright = 0 ;
    }
}

```

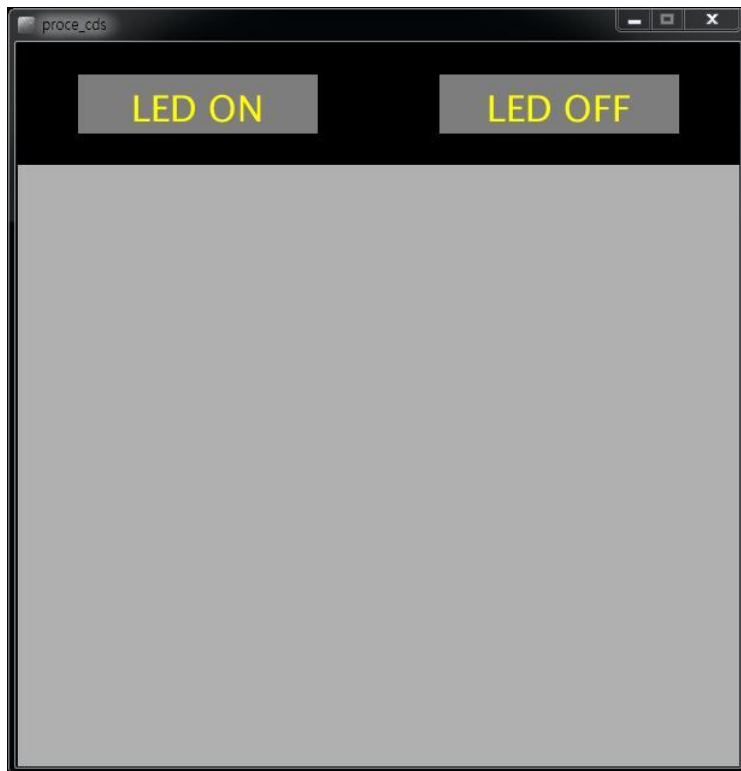
```

    fill(bright) ;
    rect(0, 100, width, height) ;
}
if (flag_p == true ) {
    flag_p = false ;
    int mo_y = mouseY / 1 ;
    if (mo_y > 25 && mo_y < 75) {
        int mo_x = mouseX / 1 ;
        if (mo_x > 50 && mo_x < 250) myClient.write('n') ;
        else if (mo_x > 350 && mo_x < 550) myClient.write('f') ;
    }
}
}

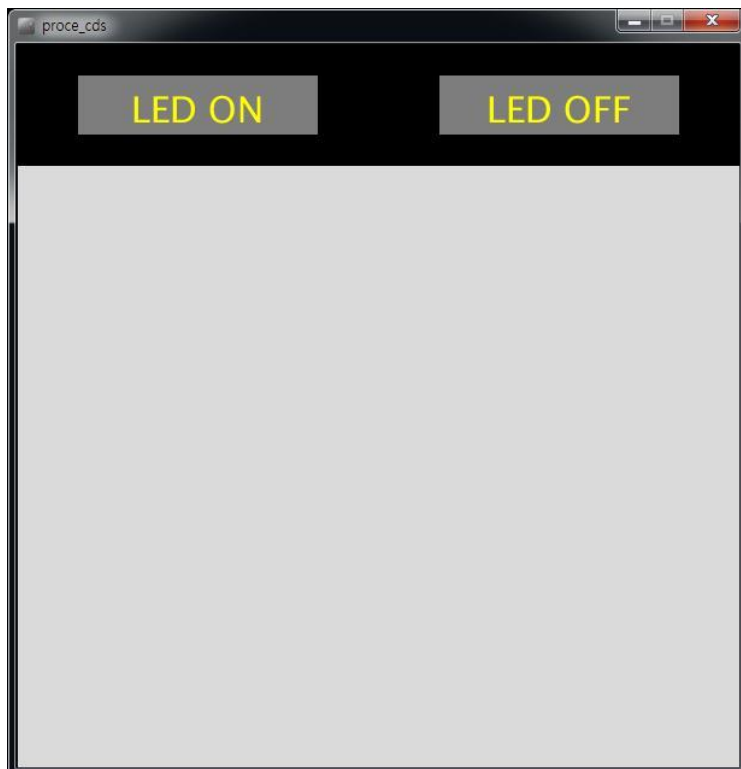
void mousePressed() {
    flag_p = true ;
}

```

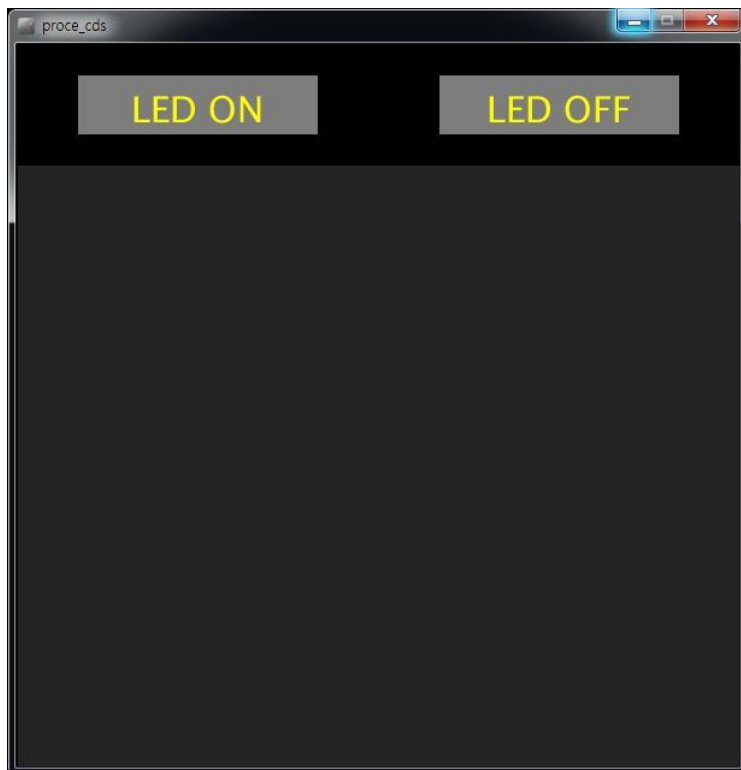
다음은 프로세싱 실행 결과입니다. 기본적으로 다음과 같은 상태로 유지되는데, 밝기가 밝아졌을 때와 어두워졌을 때에 따라 윈도우 창의 박스 색이 변하는 것을 확인할 수 있습니다.



LED를 켜올 때



손으로 가렸을 때



4. 참고문헌

- [1] ITU – T, Overview of Internet of Things, 2012, 6, 15.
- [2] IDC, <http://www.idc.com>
- [3] 아두이노 공식 웹사이트, <http://arduino.cc>
- [4] 프로세싱 공식 웹사이트, <https://processing.org>
- [5] 위키피디아, <http://wikipedia.org>
- [6] <http://www.terms.co.kr/Ethernet.htm>
- [7] http://www.ktword.co.kr/abbr_view.php?m_temp1=388
- [8] MarketandMarkets, <http://www.marketsandmarkets.com/>
- [9] <http://www.bloter.net/archives/220360>

- [10] <http://startingelectronics.com/tutorials/arduino/ethernet-shield-web-server-tutorial/basic-web-server/>
- [11] <http://www.openhomeautomation.net/tiny-wifi-temperature-arduino/>
- [12] 인스트럭터블스, <http://www.instructables.com/id/Arduino-Ethernet-Shield-Tutorial/?ALLSTEPS>
- [13] 웹서버 기반 LED 제어
<http://startingelectronics.com/tutorials/arduino/ethernet-shield-web-server-tutorial/web-server-LED-control/>
- [14] 메카솔루션 입문자를 위한 아두이노 종결 키트 매뉴얼
<http://mechasolution.com/shop/board/list.php?&id=notice>
- [15] 스파크펀, www.sparkfun.com