저자 소개

오재관

공주사범대학 졸업

전주대학교 교육학 석사(수학교육 전공)

전주대학교 전기전자통신공학과 박사과정(정보공학 전공) 수료

전) 대학수학능력시험 프로그래밍교과 출제위원

현) 중등학교 교사

홈페이지 http://ojk.kr

표지 디자인

김향희

서양화가

전북대학교 미술학과 졸업

조선대학교 대학원 졸업

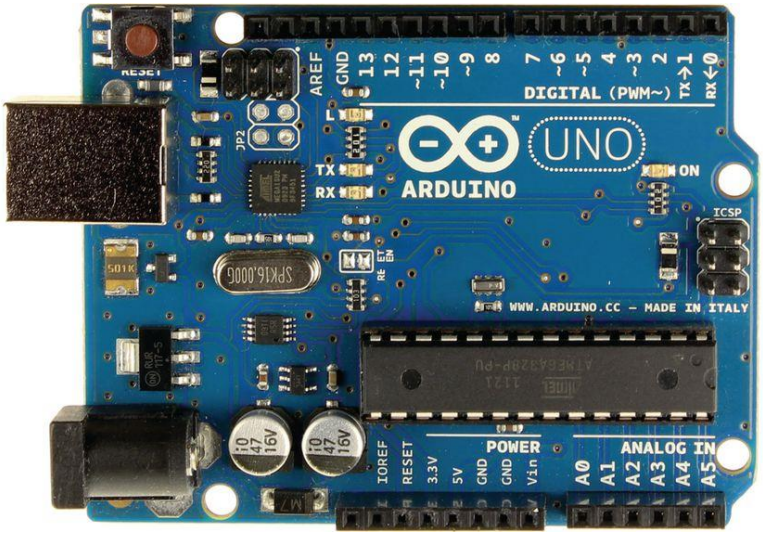
광주여자대학교 대학원 졸업

전남대학교 대학원 미술교육과 재학중

블로그 https://blog.naver.com/greentea22

Chapter 1. 아두이노 들어가기

1. 아두이노란



우리가 사용하는 전자제품은 어떤 기능을 하기 위해서 그 기능을 실행하고 제어하는 장치가 필요합니다. 이러한 장치는 성질상 그 크기가 크지 않고 저장할 수 있는 용량도 작기 때문에 마이크로 컨트롤러(Micro Controller)라고 합니다. 아두이노(이탈리아어: Arduino 아르두이노)는 이러한 마이크로 컨트롤러 중의 하나로서 오픈 소스를[1] 기반으로 한 단일 보드 마이크로컨트롤러이며 이를 이용하여 필요로 하는 마이크로컨트롤러 기능을 제작할 수 있는 개발 도구 및 환경입니다. 쉽게 말하면 아두니노를 사용하여 제작자가 원하는 새로운 마이크로컨트롤러를 만들 수 있습니다. 예를 들면 RC 카나 드론을 제어하는 컨트롤로를 제작할 수 있는 개발도구입니다.

2005 년 이탈리아의 IDII(Interaction Design Institutelvera)에서 하드웨어에 익숙지 않은 학생들이 자신들의 디자인 작품을 손쉽게 제어할 수 있게 하려고 고안되었고, 처음에 AVR을 기반으로 만들

[1] 하드웨어의 제작 방법이 공개되어 있어, 누구가 그 설계도에 의해서 제품을 제작할 수 있는 것을 말합니다.

어졌으며, 현재는 아트멜 AVR 계열의 보드가 가장 많이 사용되고 있습니다. 아두이노 우노 R3 보드에 사용되는 마이크로 컨트롤러 칩은 ATMEGA328 이라는 칩으로 Atmel 사에서 개발한 8Bit 칩입니다. ARM 계열의 Cortex-M0(Arduino M0 Pro)과 Cortex-M3(Arduino Due)를 이용한 제품도 존재합니다.

아두이노는 다수의 스위치나 센서로부터 값을 받아들이고, LED나 모터와 같은 외부 전자 장치들을 통제함으로써 환경과 상호작용이 가능한 장치를 만들어 낼 수 있습니다. 임베디드 시스템 중의 하나로 쉽게 개발할 수 있는 환경을 이용하여, 장치를 제어할 수 있고, 요즘들어 각광 받고 있는 사물인터넷을 만들어 낼 수 있습니다.

아두이노는 제어 방식이 프로그래밍 방식이므로 프로그래밍을 하여 이를 컴파일하여 펌웨어로 바꾼 다음에 아두이노에 저장하여 실행하는 방식입니다. 이를 위해서 아두이노는 아두이노 통합 개발 환경(IDE)을 제공하며, 소프트웨어 개발과 실행코드 업로드도 제공합니다. 또한프로세싱, Max/MSP와 같은 소프트웨어와 연동할 수 있고 , 오픈소스이기 때문에 아두이노를 기반으로 여러 가지 프로젝트를 수행할 수 있습니다.

아두이노의 가장 큰 장점은 마이크로컨트롤러를 쉽게 동작시킬 수 있다는 점입니다.[2] 아두이노는 컴파일된 펌웨어를 USB를 통해 쉽게 업로드 할 수 있고, 다른 모듈에 비해 비교적 저렴하며, 윈도우를 비롯한 맥 OS X, 리눅스와 같은 여러 OS를 모두 지원합니다. 아두이노 보드의 회로도가 CCL에 따라 공개되어 있으므로, 누구나 직접 보드를 만들고 수정할 수 있다.

아두이노가 인기를 끌면서 이를 비즈니스에 활용하는 기업들도 늘어나고 있습니다. 장난감 회사 레고는 자사의 로봇 장난감과 아두이노를 활용한 로봇 교육 프로그램을 학생과 성인을 대상으로 북미 지역에서 운영하고 있습니다. 자동차회사 포드는 아두이노를 이용해 차량용 하드웨어와 소프트웨어를 만들어 차량과 상호작용을 할 수 있는 오픈XC라는 프로그램을 선보이기도 했습니다.

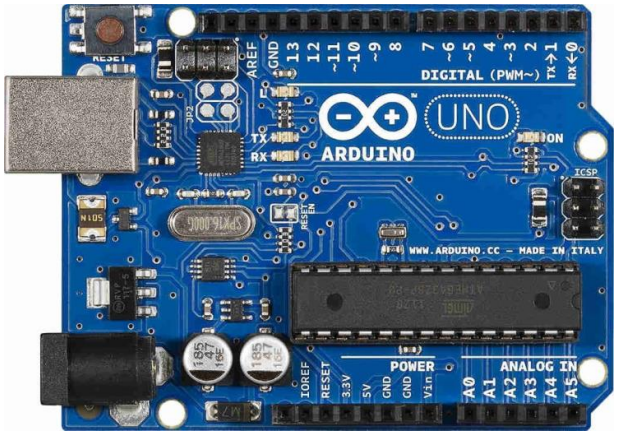
국내에서도 수 많은 교육기관에서 아두이노에서 제공하는 C언어 또는 스크래치 또는 스크래치 기반의 블록프로그래밍 언어를 이용하여 아두이노를 제어하는 교육시스템을 제공하고 있습니다.

---

[2] 일반적으로 AVR 프로그래밍은 AVRStudio(Atmel Studio[6]로 변경, ARM 도구 추가됨)와 WinAVR(avr-gcc)의 결합으로 컴파일하거나 IAR E.W.나 코드비전(CodeVision)등으로 개발하여, 별도의 ISP 장치를 통해 업로드를 해야하는 번거로운 과정을 거쳐야 합니다.

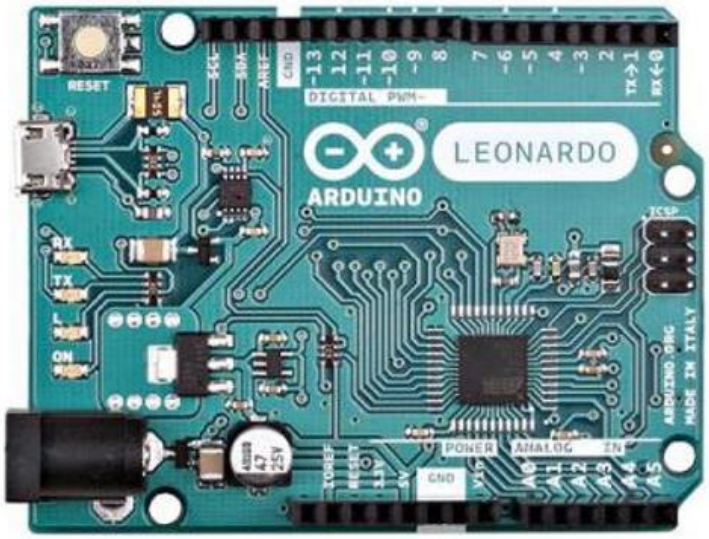
2. 아두이노의 종류

【1】아두이노 우노 (Arduino UNO)



가장 기본적인 아두이노 보드입니다. 크기가 크고, 여러 입출력단자에 맞게 나온 확장 보드(쉴드, shield)가 다양해서 초보자들이 연습하기엔 딱 좋습니다. 대부분의 책 또는 예제에 제일 많이 쓰입니다. USB-B타입 커넥터가 달려 있고요, 크기가 커서 작게 뭔가를 만들기엔 제약이 많습니다.

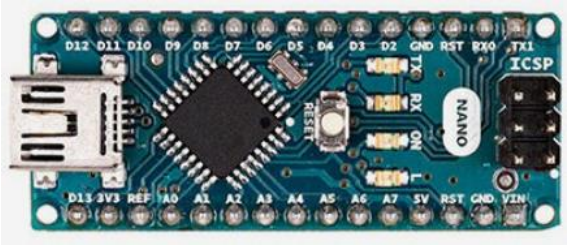
【2】Arduino Leonardo with Headers



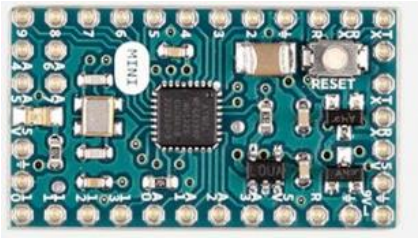
【3】Arduino Micro

【4】아두이노 나노 (Arduino Nano)

예전 MP3 플레이어 등에 많이 쓰였던 USB mini B 타입 커넥터가 달려 있고, 외부 전원 잭이 제거된 작은 아두이노입니다. 기능적으로는 아두이노 우노와 동일한 작은 보드죠.



【5】아두이노 미니



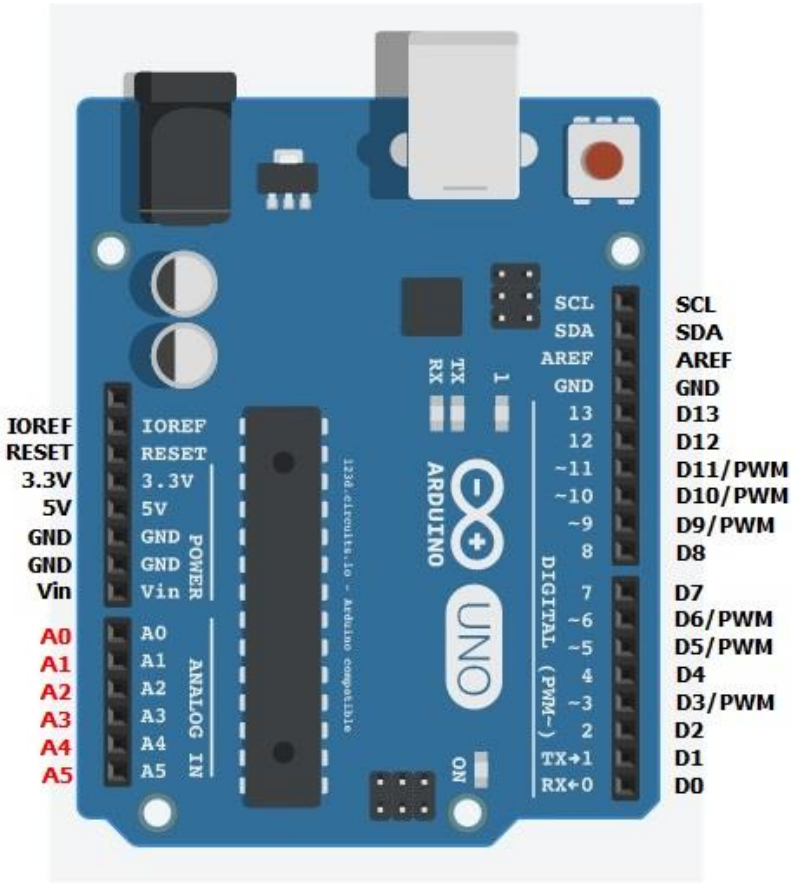
【6】아두이노 프로 미니 (Arduino Pro Mini)

아두이노 나노 보다도 크기가 작습니다. USB 포트를 제거해 버린 겁니다. CPU에서 나오는 직렬 포트를 USB로 변환해주는 회로까지 제거하여 소형화한 것인데요, 이를 컴퓨터와 연결하기 위해서는

USB TTL Serial 케이블이 있어야 합니다. 마침 저희 회사의 장비 중에 이 케이블이 필요한 곳이 있어서 써본 경험이 있어서 시도해 볼 수 있었는데, 아주 만족스러워서 지금은 제일 자주 쓰고 있습니다. 앞선 장치들이 Atmel사의 ATmega328P라는 프로세서 기반에 5V I/O인데 반해, 아두이노 프로 미니 3.3V I/O 기반 버전도 있어 좀 다른 구성도 해볼 수 있습니다. 물론 5V 버전도 있고요, 각 전압 버전에 따라 USB TTL Serial케이블도 다른 걸써야 합니다. 물론 두가지 전압 모두 지원하는 어댑터도 판매되고 있습니다.

【7】아두이노 제로

3. 아두이노 GPIO



【1】GPIO(General Purpose Input Output)

여러 가지 용도로 사용되는 입출력 포트

【2】전원

3.3V , 5V, GND(Ground, -) 3 개

【3】아날로그 입출력

A0, A1, A2, A3, A4, A5 총 6 개가 있고 입력으로도 사용할 수 있고 출력으로도 사용할 수 있습니다.

따라서 입력 또는 출력을 사용하기 전에 지정하여야 합니다.

【4】 디지털 출력

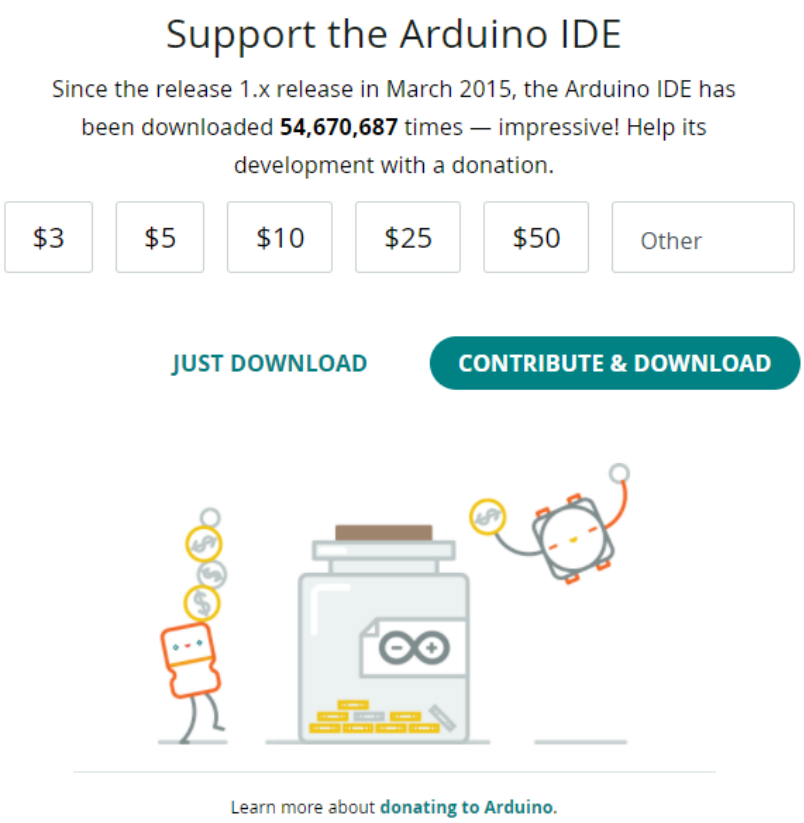
4. 프로그래밍 툴

【1】Windows 스케치

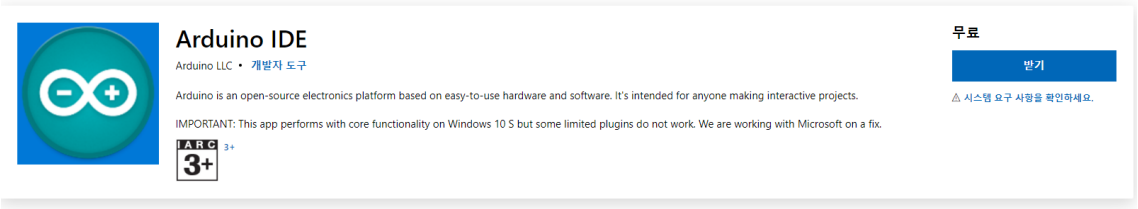
① 다운로드와 설치

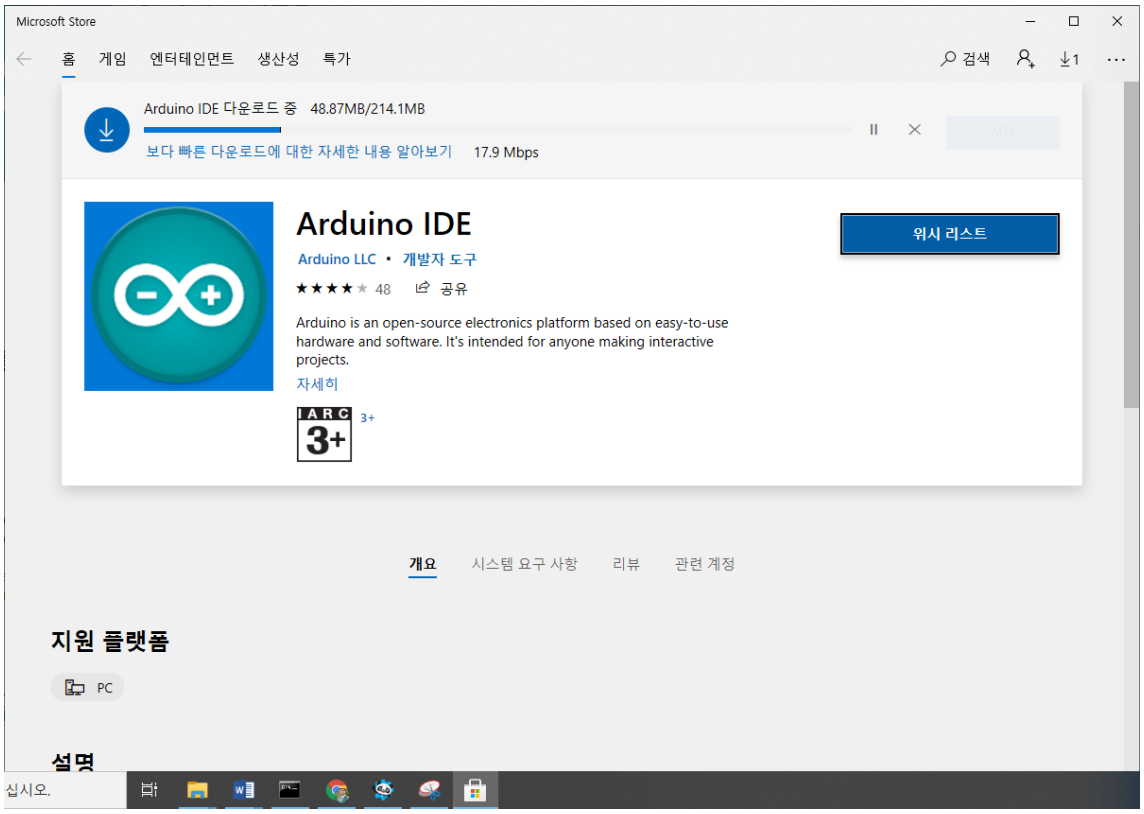
https://www.arduino.cc/ 에 접속하여 프로그램을 다운로드 합니다.

[JUST DOWNLOAD] 를 클릭합니다.

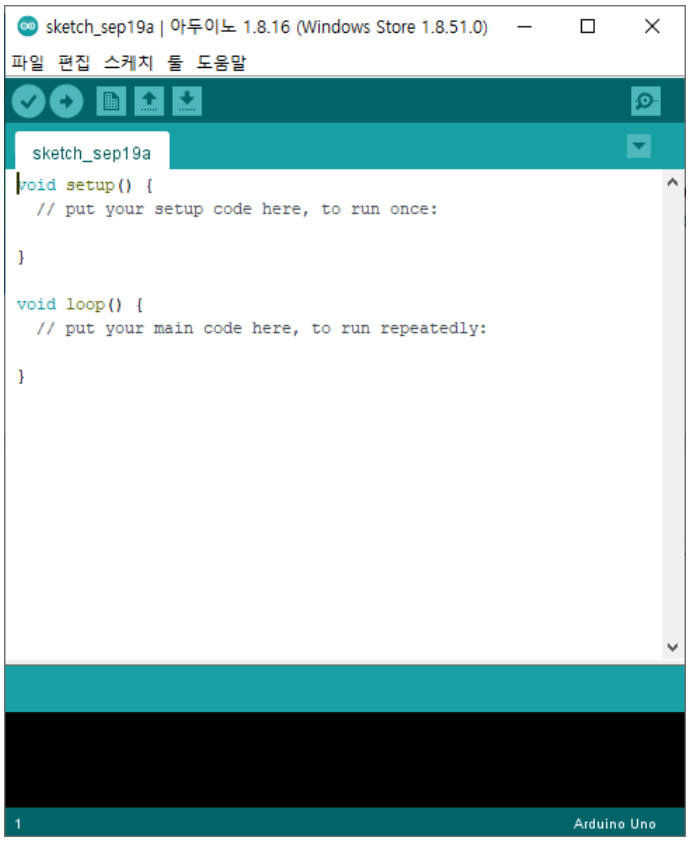

Support the Arduino IDE
Since the release 1.x release in March 2015, the Arduino IDE has been downloaded 54,670,687 times — impressive! Help its development with a donation.
$3
$5
$10
$25
$50
Other
JUST DOWNLOAD
CONTRIBUTE & DOWNLOAD
Learn more about donating to Arduino.

[받기]를 클릭합니다.


Arduino IDE
Arduino LLC • 개발자 도구
Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.
IMPORTANT: This app performs with core functionality on Windows 10 S but some limited plugins do not work. We are working with Microsoft on a fix.
IARC 3+
무료
받기
⚠ 시스템 요구 사항을 확인하세요.

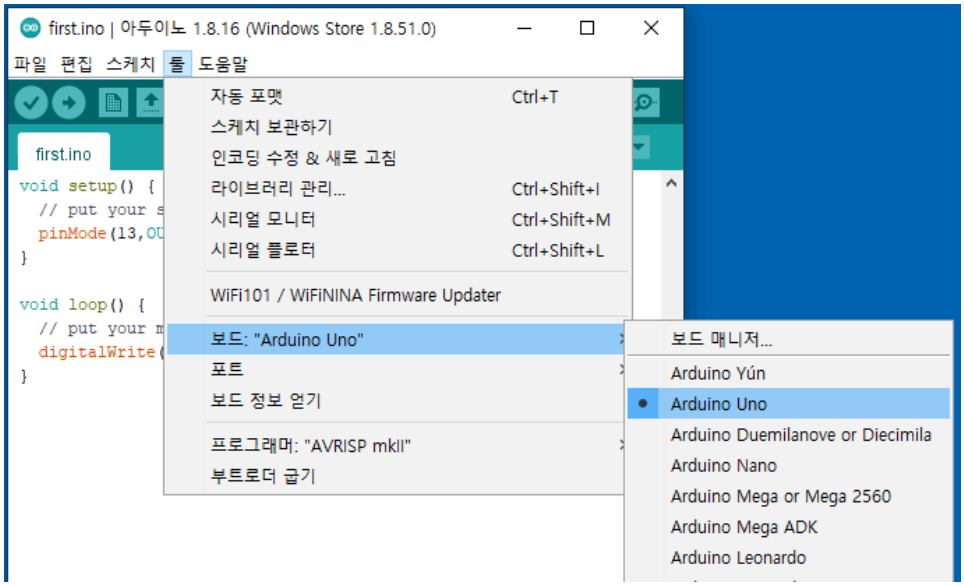

Microsoft Store
홈
게임
엔터테인먼트
생산성
특가
검색
Arduino IDE 다운로드 중 48.87MB/214.1MB
보다 빠른 다운로드에 대한 자세한 내용 알아보기
17.9 Mbps
Arduino IDE
Arduino LLC • 개발자 도구
48
공유
Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.
자세히
3+
위시 리스트
개요
시스템 요구 사항
리뷰
관련 계정
지원 플랫폼
PC
설명
십시오.


Microsoft Store
홈
게임
엔터테인먼트
생산성
특가
검색
이 제품은 설치되어 있습니다.
시작
Arduino IDE
Arduino LLC • 개발자 도구
48
공유
Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.
자세히
3+
위시 리스트

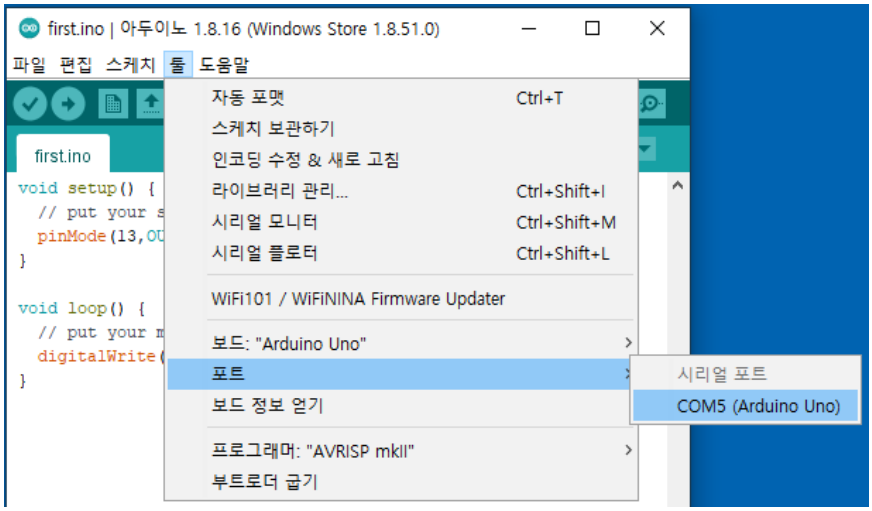


② 설정

[메뉴-툴-보드]에서 현재 사용하는 아두이노 종류를 선택합니다. 여기에서는 Arduino Uno를 선택합니다.



앞에서 확인한 것처럼 현재 아두이노가 COM5 에 연결되어 있으므로 [메뉴-툴-포트]에서 COM(Arduino Uno)를 선택합니다.



【2】Linux 스케치

다음과 같은 에러 메시지를 나오면

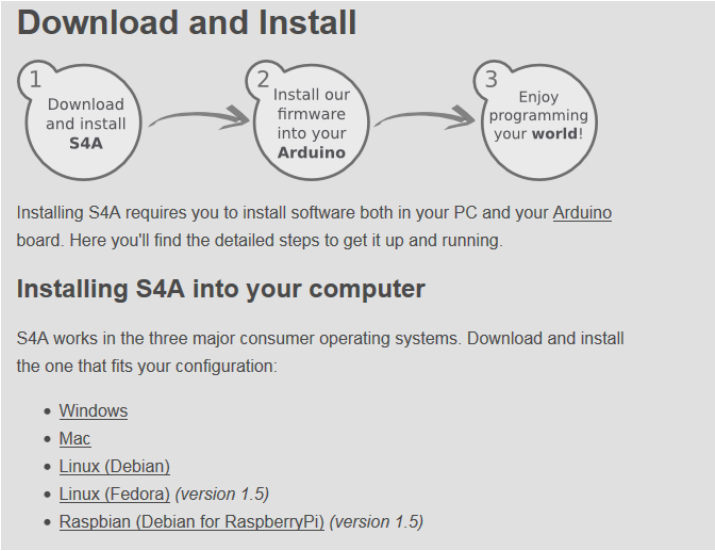
```
스케치는 프로그램 저장 공간 1,046 바이트(3%)를 사용. 최대 32,256 바이트.
전역 변수는 동적 메모리 9 바이트(0%)를 사용, 2,039 바이트의 지역변수가
남음.  최대는 2,048 바이트.
avrdude: ser_open(): can't open device "/dev/ttyACM0": Permission denied
ioctl("TIOCMGET"): Inappropriate ioctl for device
```

/dev/ttyACMO 를 일반 사용자도 접근할 수 있도록 퍼미션을 조정한다.

```
root@ojk:/home/ojk# cd /dev
root@ojk:/dev# ls ttyACM0
```

```
ttyACM0
root@ojk:/dev# ls -l ttyACM0
crw-rw---- 1 root dialout 166, 0  5 월 25 16:07 ttyACM0
root@ojk:/dev# chmod 766 ttyACM0
root@ojk:/dev# ls -l ttyACM0
crwxrw-rw- 1 root dialout 166, 0  5 월 25 16:07 ttyACM0
```

S4A설치 싸이트(http://s4a.cat)에서 다운로드 합니다.



자신의 운영체제에 맞는 프로그램을 다운로드합니다.

Windows를 클릭하여 프로그램을 다운로드 하여 압축을 풀고 설치합니다.

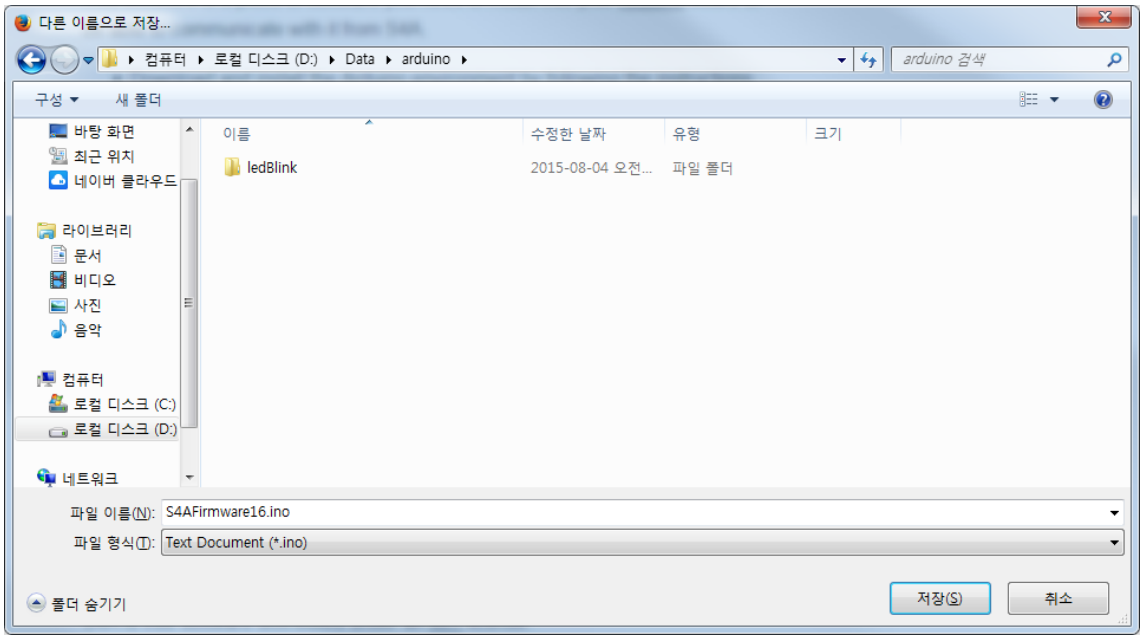
Installing the Firmware into your Arduino

This firmware is a piece of software you need to install into your Arduino board to be able to communicate with it from S4A.

- Download and install the Arduino environment by following the instructions on http://arduino.cc/en/Main/Software. Take in account Arduino Uno requires at least version 0022.
- Download our firmware from here
- Connect your Arduino board to a USB port in your computer
- Open the firmware file (S4AFirmware16.ino) from the Arduino environment
- In the Tools menu, select the board version and the serial port where the board is connected
- Load the firmware into your board through File > Upload

Down our firmware from here

여기를 마우스 오른쪽 버튼 클릭하여 다른이름으로 저장합니다.

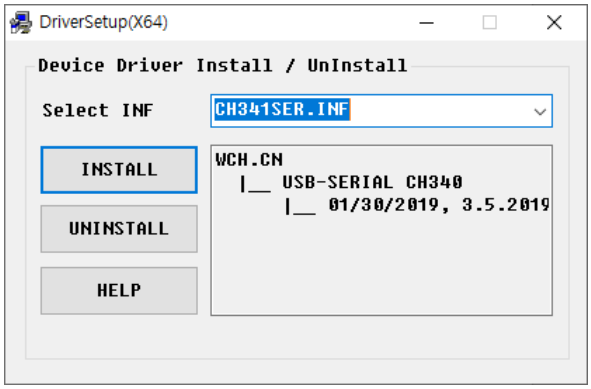


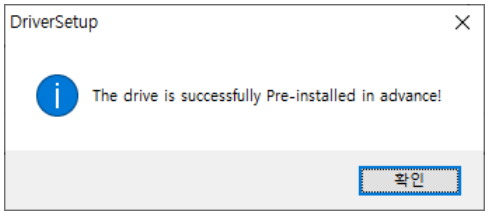
다운받은 S4AFirmware16.ino를 아두이노 보드에 전송합니다. 여기에서는 어쩔 수 없이 아두이노

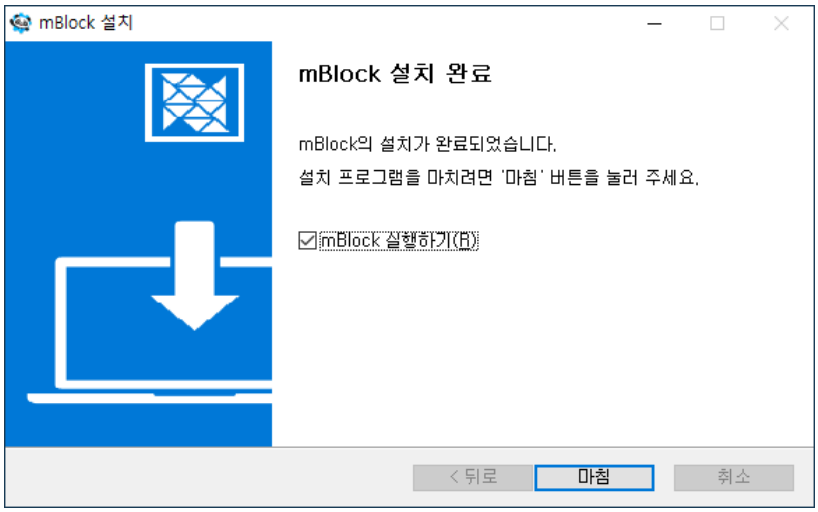
회사에서 제공하는 스케치 프로그램을 사용합니다.

【3】MBlock

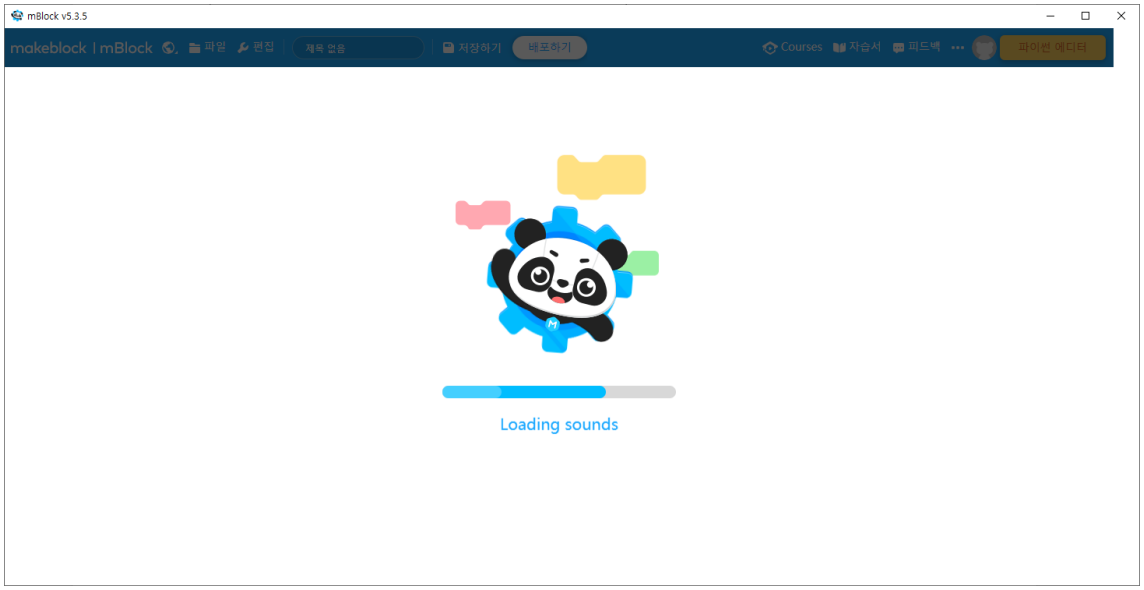
① MBlock 설치



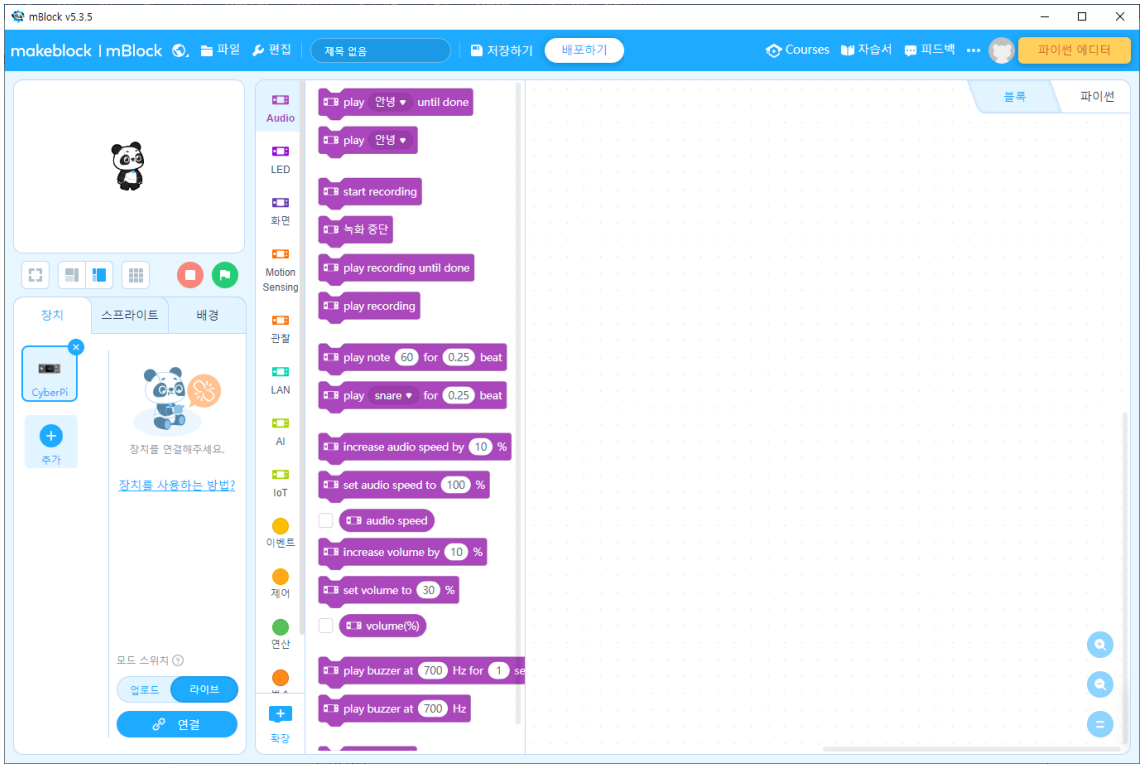




처음 실행 시에는 실행하는데에 상당한 시간이 소요되므로 인내를 가지고 기다립니다.

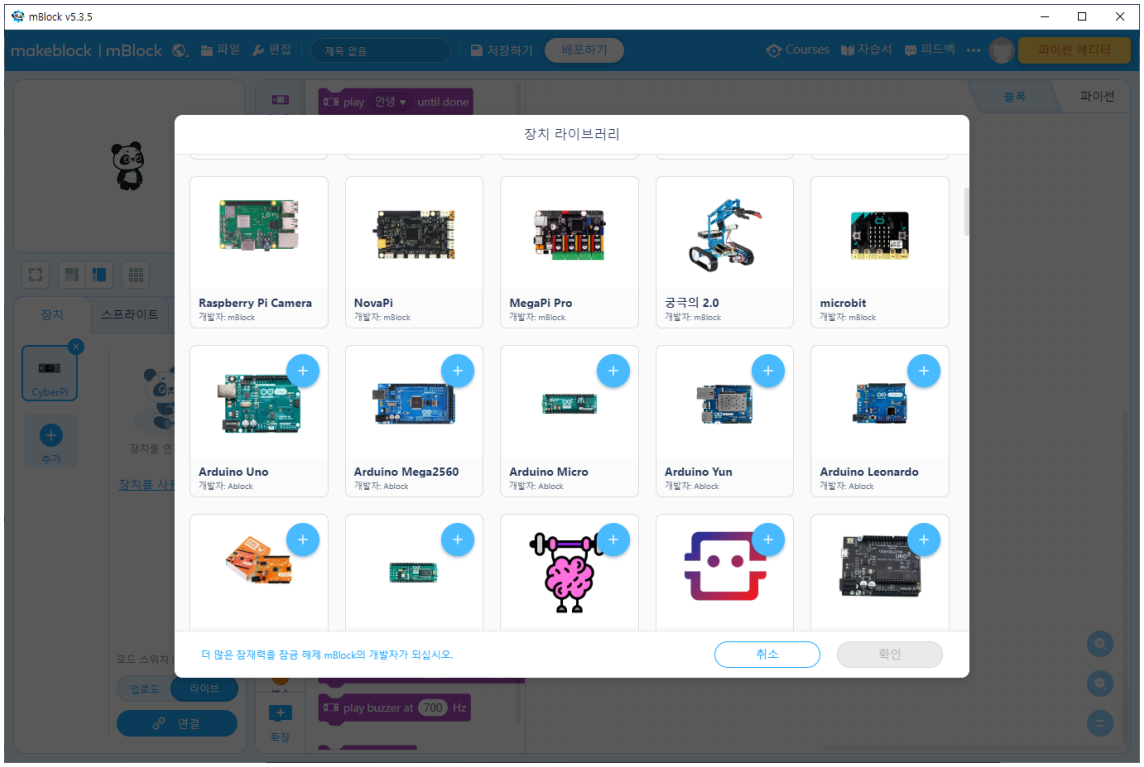


다음과 같은 화면이 나오면 mblock 설치와 실행이 완료된 것입니다.

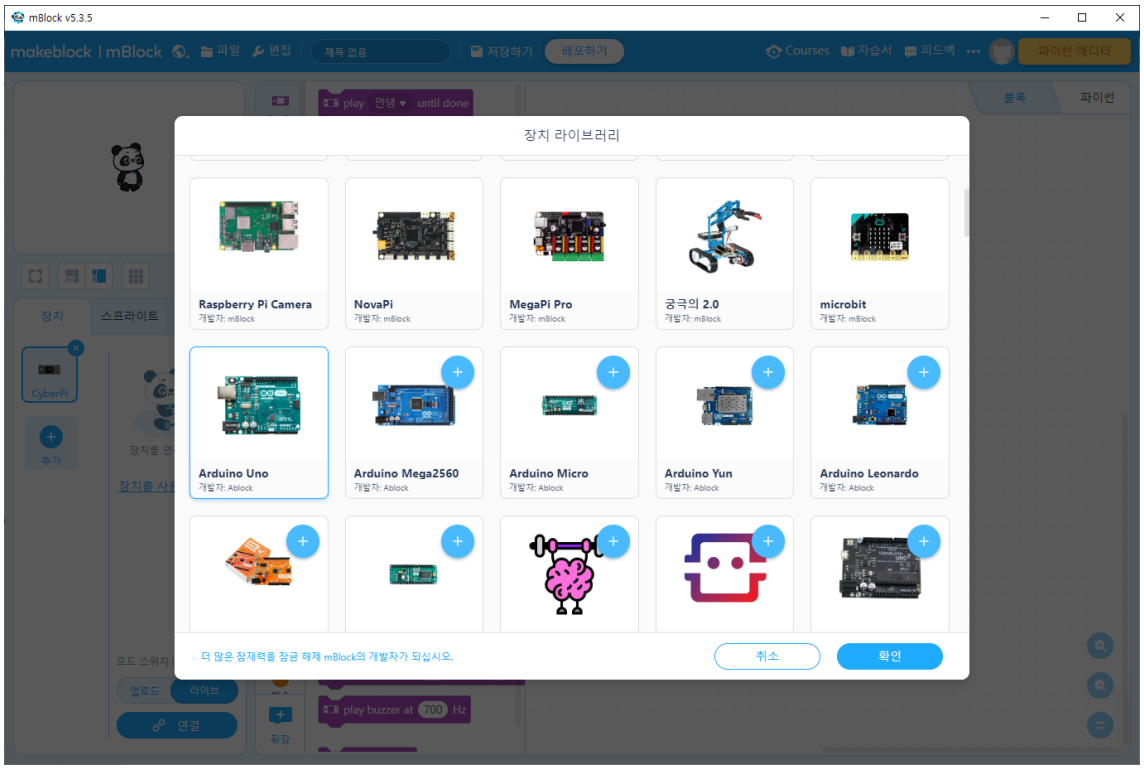


② 드라이버 설치

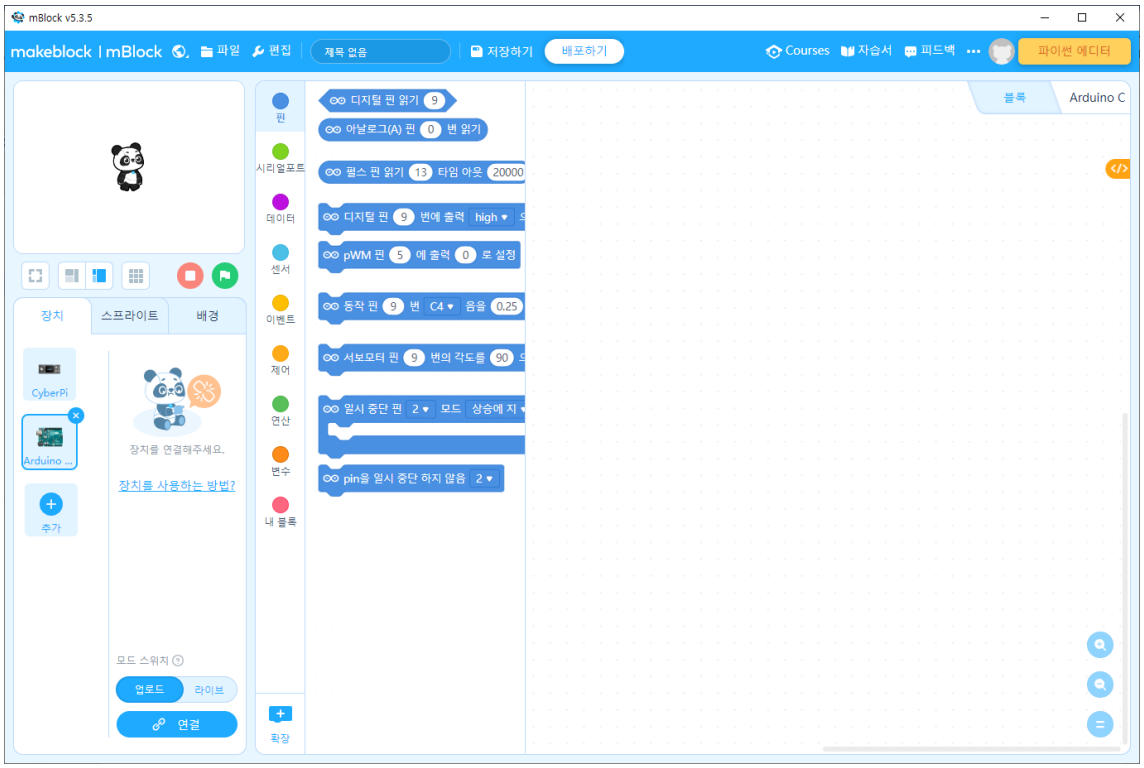
Arduino Uno 를 선택하고 [+]를 클릭하면 Arduino Uno 드라이버를 다운도르 합니다.

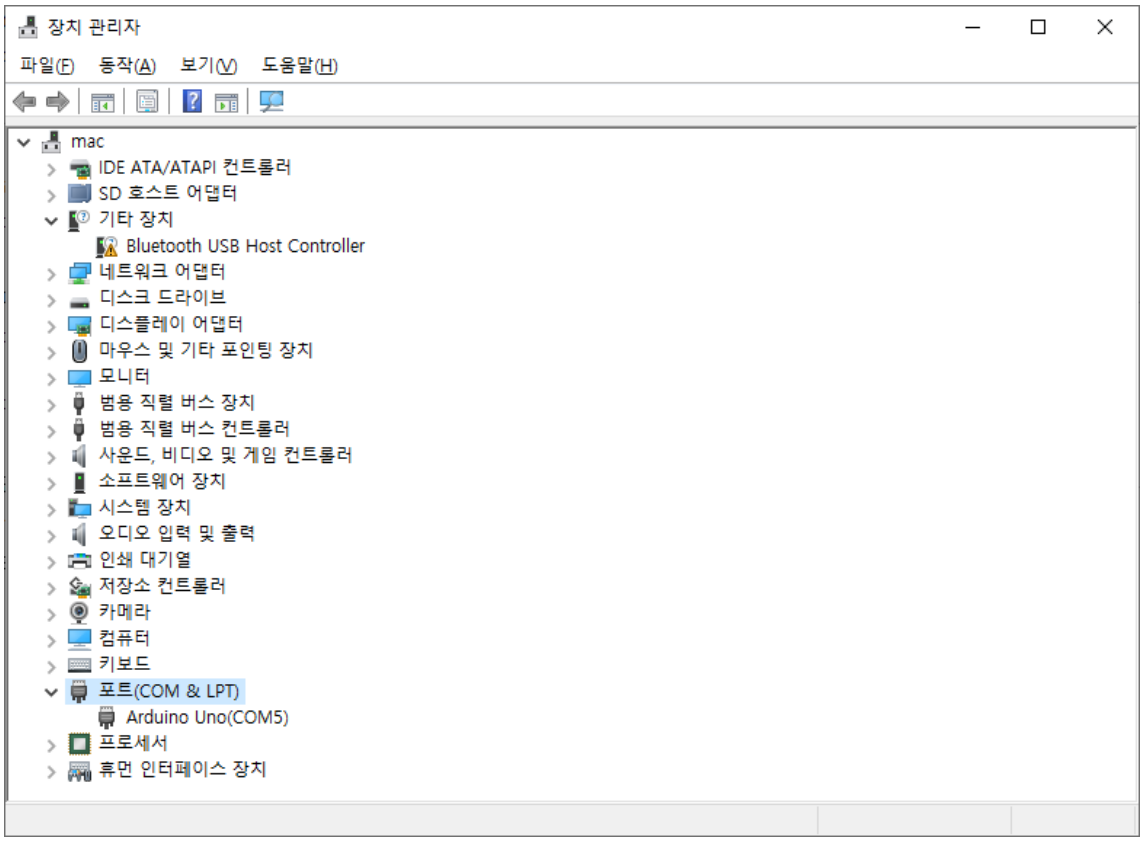


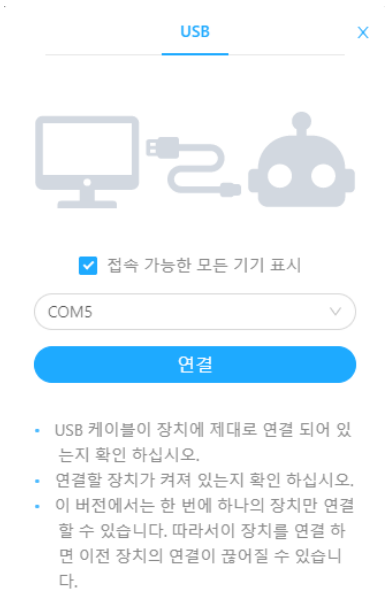
Arduino Uno 를 선택하고 [확인]을 클릭합니다.



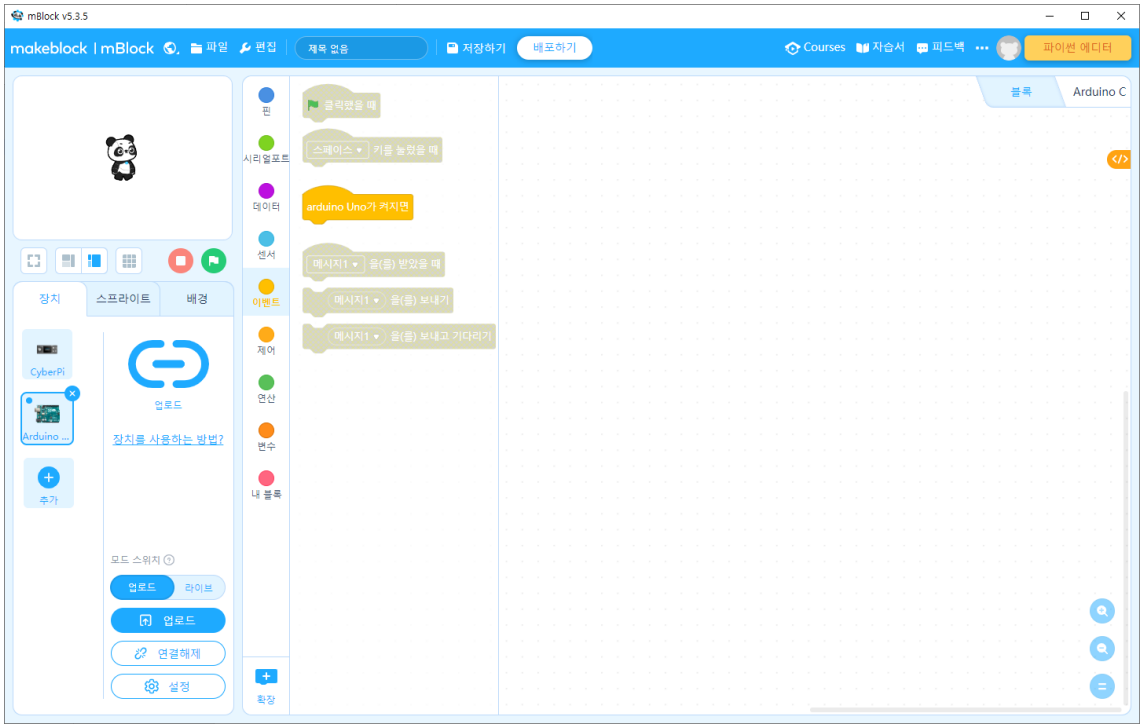
화면이 아두이노를 프로그래밍할 수 있도록 바뀝니다.




장치 관리자
파일(F) 동작(A) 보기(V) 도움말(H)
mac
IDE ATA/ATAPI 컨트롤러
SD 호스트 어댑터
기타 장치
Bluetooth USB Host Controller
네트워크 어댑터
디스크 드라이브
디스플레이 어댑터
마우스 및 기타 포인팅 장치
모니터
범용 직렬 버스 장치
범용 직렬 버스 컨트롤러
사운드, 비디오 및 게임 컨트롤러
소프트웨어 장치
시스템 장치
오디오 입력 및 출력
인쇄 대기열
저장소 컨트롤러
카메라
컴퓨터
키보드
포트(COM & LPT)
Arduino Uno(COM5)
프로세서
휴먼 인터페이스 장치


USB
접속 가능한 모든 기기 표시
COM5
연결
USB 케이블이 장치에 제대로 연결 되어 있는지 확인 하십시오.
연결할 장치가 켜져 있는지 확인 하십시오.
이 버전에서는 한 번에 하나의 장치만 연결할 수 있습니다. 따라서이 장치를 연결하면 이전 장치의 연결이 끊어질 수 있습니다.

아두이노와 컴퓨터가 연결이 된 상태입니다.



Chapter 2. C언어 기초

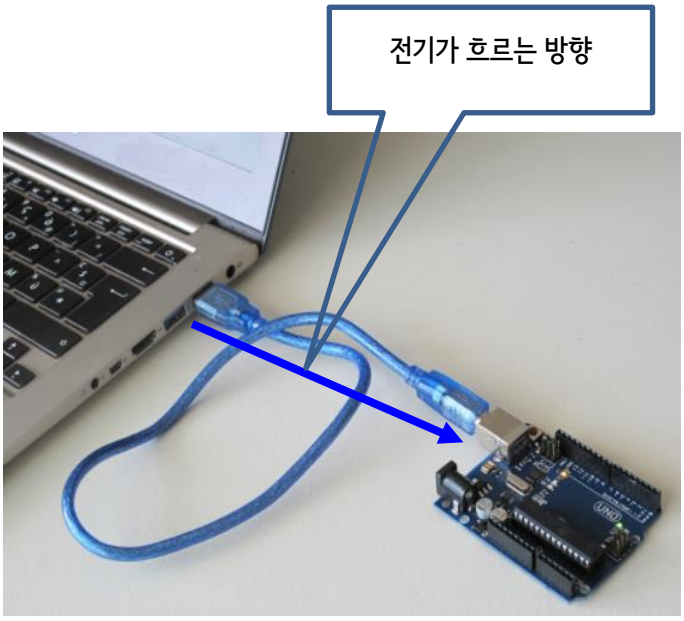
Chapter 3. MBlock 기초

Chapter 4. 아두이노 기초

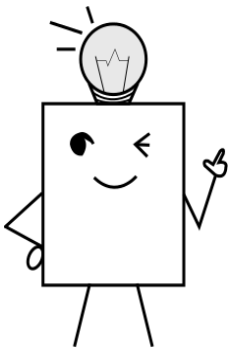
1. 전원 공급

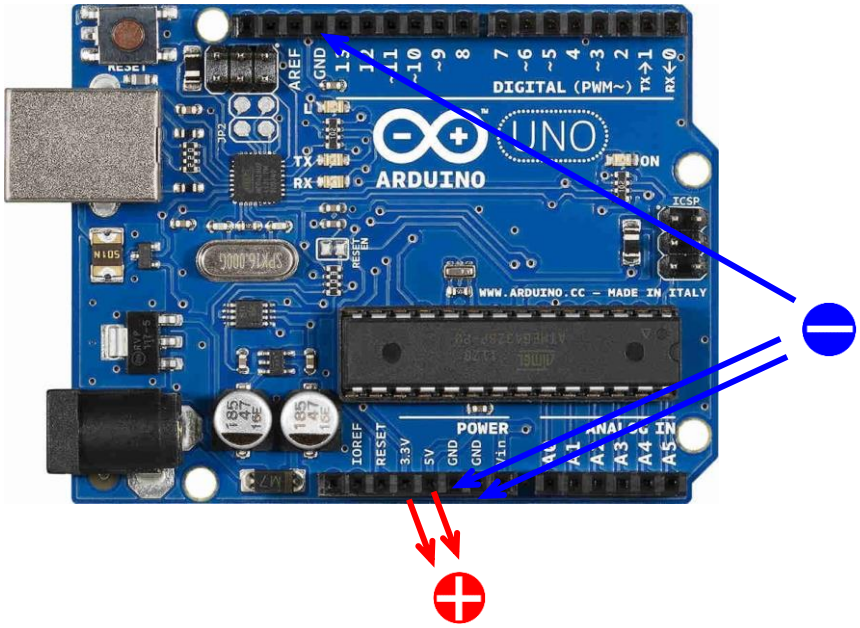
【1】 내부 전원 공급

컴퓨터와 아두이노를 USB로 연결하면 컴퓨터에서 사용하는 전기가 USB를 통하여 흘러나와 아두이노 USB를 통하여 들어가서 아두이노에 전원이 공급됩니다.



아두이노는 기본적으로 5V와 3.3V를 사용합니다. 5V 핀이 1 개, 3.3V 핀이 한 개가 있고 그라운드가 3 개가 있습니다. USB를 통하여 전원을 공급하면 5V에서 나와 GND로 들어가거나 3.5V에서 나와 GND로 들어가도록 설계가 되어있습니다. 따라서 USB로 컴퓨터로 연결하면 프로그래밍 없이 5V, 3.3V 핀에서 전류가 나와서 그라운드도 들어가게 회로를 구성할 수 있습니다.



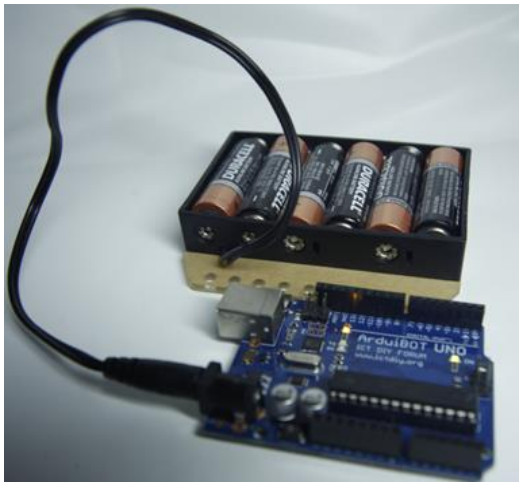


【2】 외부전원 공급

아두이노에 외부 전원을 공급하는 것은 절대적으로 필요합니다. 컴퓨터와 연결이 되어 있는 상태에서 아두이노를 다루는 경우에, 모터 사용의 경우처럼 특별한 경우가 아니면 외부 전원의 공급 없이 컴퓨터와 아두이노가 연결된 상태에서 프로그래밍하고, 이를 아두이노에 전송하여 실행을 합니다.

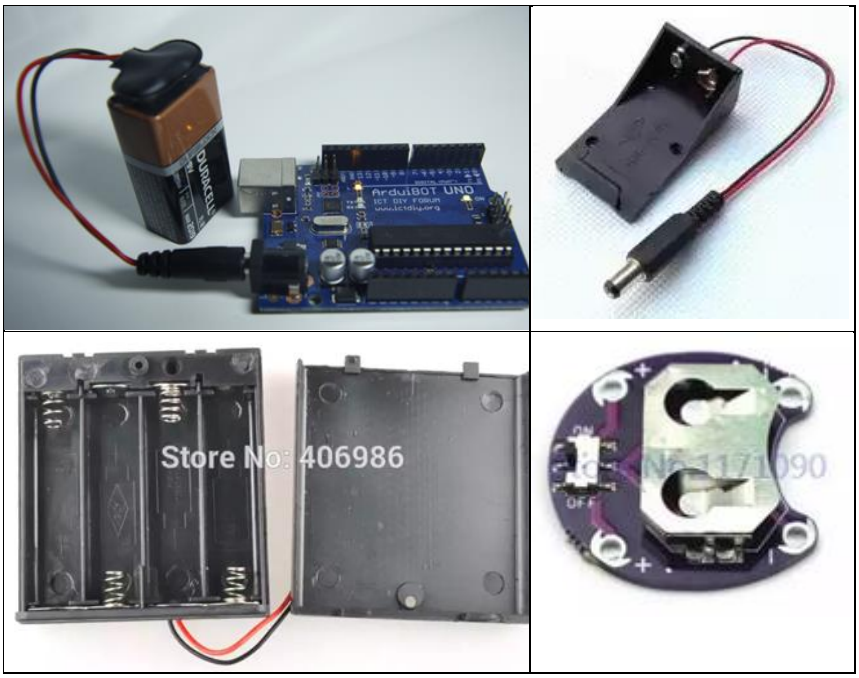
그러나 대부분의 경우에 프로그래밍할 때는 컴퓨터와 아두이노를 연결하여 프로그래밍하고 작성된 프로그램을 아두이노에 전송하고, 실행할 때에는 컴퓨터와 아두이노를 분리하여 실행합니다. 따라서 아두이노에서 외부 전원의 공급은 필수적입니다.

Arduino의 전원 공급 방식은 매우 다양한데, 건전지로 전기를 공급하는 몇몇 배터리 홀더(battery holder) 등을 살펴보겠습니다. 대개 모터나 대용량의 LED등을 구동하기 위해서는 매우 큰 전원이 필요할 수 있으나, 간단하게 디스플레이나 buzzer, 센서 등을 붙이는데는 간단한 건전지 전원만으로도 충분합니다. 흔히 판매되는건 다음과 같습니다.



참고로 Arduino의 외부 전원부는 7V~12V입력이 추천 범위이고 최대는 6V~20V를 수용합니다.

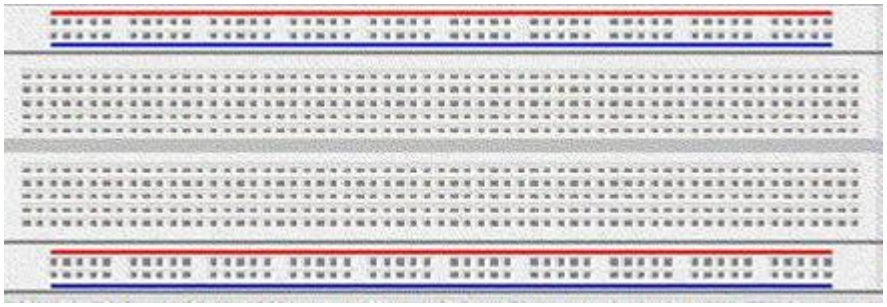
배터리를 이용한 최고의 간단한 조합 중 하나는 오른쪽과 같이 9v 건전지를 사용하는 것입니다. 9v 홀더는 몇백원으로 구매가능하고, 9v 배터리는 개당 1~2 천원 정도로 구매 가능합니다.



2. 브레드 보드

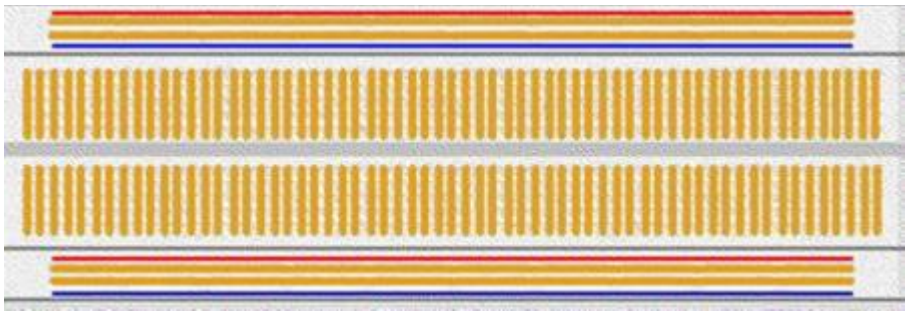
전자회로를 구성하기 위해서는 전선을 연결하고 납땜을 하여야 합니다. 그러나 실습용이나 초기 개발을 할 경우에는 시간과 비용이 많이 소모됩니다. 따라서 납땜을 대신하여 회로를 구성할 수 있는 방법이 필요한데 이러한 목적으로 만들어 진 것이 그림과 같은 브레드 보드(Bread Board) 또는 빵판이라고 합니다. 우리가 먹는 빵을 만들 때 사용하는 기구와 모양새가 비슷하다고 해서 브레드 보드라고 합니다.

[빵을 만들 때 사용하는 브레드 보드]



[전기 회로를 구성하는 브레드 보드] [앞면]

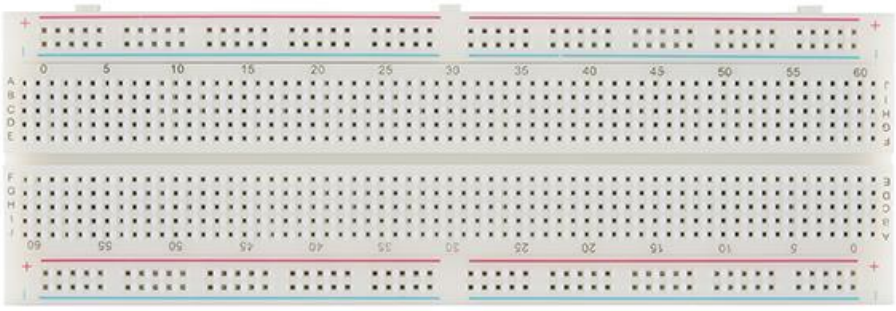
브레드보드에는 핀을 꼽을 수 있는 구멍들이 있고 이 구멍에 점퍼 와이어 또는 전자부품을 바로 꼽아서 납땜한 것과 같은 효과를 낼 수 있는 보드입니다.



[전기 회로를 구성하는 브레드 보드] [뒷면]

브레드보드 뒷면을 뜯어서 안을 들여다 보면 같은 철심 위에 있는 구멍들이 서로 연결되어 있습니다. 가로로 긴 줄은 그림처럼 가로로 연결이 되어 있습니다. 세로로 되어 있는 짧은 줄은 세로로 연결이 되어 있습니다. 그러므로 같은 줄에 전선이니 부품을 꽂으면 연결이 되므로 납땜한 것과 같은 회로를 구성하게 됩니다.

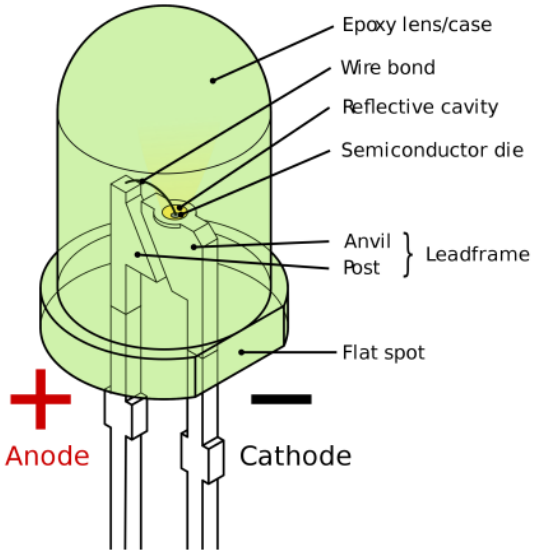
다음 그림과 같이 브레드 보드 종류에 따라서 가로로 긴 부분이 2 개의 부분으로 나누어져 있어서 가로로 긴 줄이 서로 연결이 되지 않는 브레드 보드도 있으므로 연결할 때 주의하여야 합니다.



일반적으로 긴 두 줄에서 빨간 줄에는 전원을 연결하고 파란 줄에는 그라운드(Ground,)를 연결합니다. 이런 구조를 가진 이유는 플러스와 마이너스를 더 많이 연결하기 위함입니다.

3. LED

아두이노에서 전류가 흘러 나와서 아두이노로 들어가면서 LED를 켜는 회로를 구성하여 봅시다.

실제	회로도(fritzing)	구성도
		
길이가 긴 쪽이	구부러진 쪽이	상세도

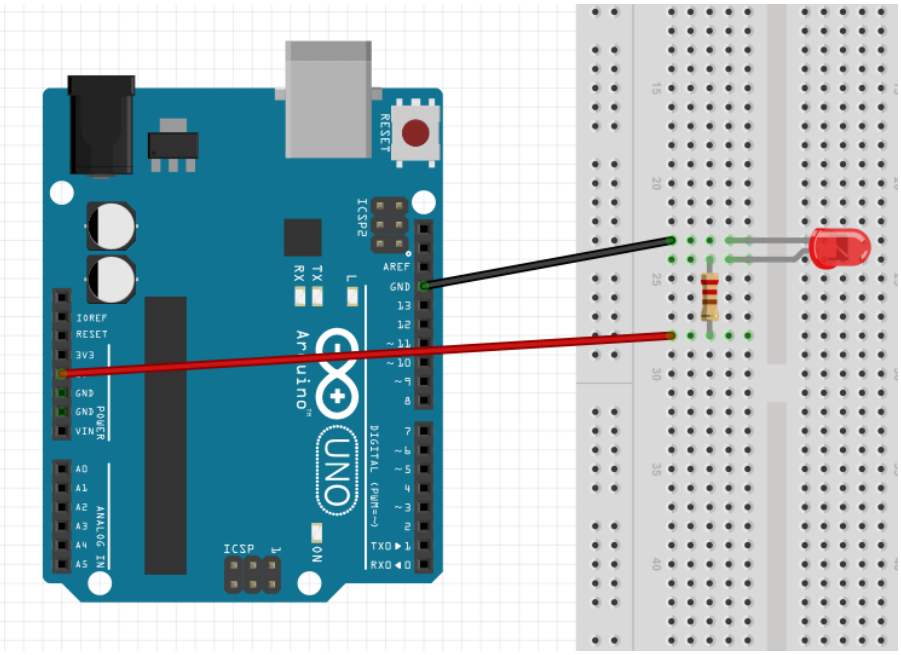
【1】기본 회로 구성

① 5V와 GND

5V와 GND를 이용하여 LED를 켜봅시다. 아두이노에는 3.3V 1 개, 5V 1 개, GND 3 개 있으므로 5V를 로 삼고 GND를 로 설정하면 5V 핀에서 전기가 흘러 나와 GND로 흘러 들어갑니다. 이 중간에 저항과 LED를 연결하면 기본 회로가 구성이 됩니다. 이렇게 구성을 하고 아두이노 USB 를 컴퓨터에 연결하면 아두이노에 전원이 공급되므로 LED가 켜질 것입니다.

저항은 크게 1K~ 10K 중의 하나를 사용합니다. 저항은 반드시 연결하여야 한다는 것을 잊어서는 안 됩니다. 5V 전원을 사용하므로 부품에 별 문제가 없을 것이라고 생각하면 오산입니다. 5V 전원에도 회로를 구성하는 부품이 손상이 있을 수 있으므로 주의하여야 합니다.

다음 회로도와 같이 회로를 구성합니다. LED를 연결할 때는 5V 핀에 연결된 쪽이 길이가 긴 부분에 오게 연결합니다. 저항은 LED앞에 위치시키나 뒤에 위치 시키나 관계 없으나 일반적으로 LED 앞에 연결합니다.



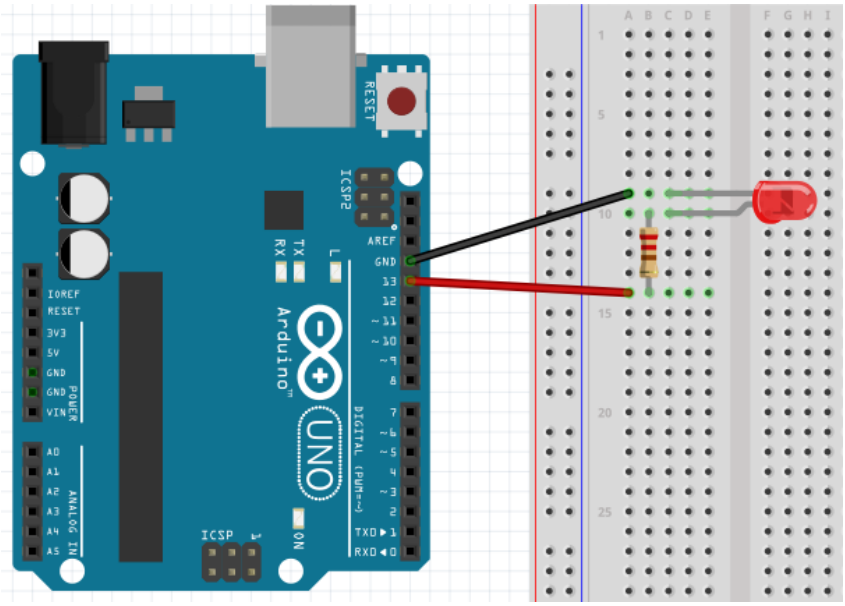
아두이노와 브레드보드를 사용하여 위와 같이 회로를 구성한 다음 컴퓨터와 아두이노를 연결하여 봅시다. 프로그램의 구성 없이 컴퓨터 USB를 통하여 아두이노에 전기가 흘러가고 그 전기가 브레드 보드를 통해 다시 아두이노로 흘러 들어 옴으로써 LED 에 불이 들어옵니다.

② 핀출력과 GND

이제는 프로그램을 사용하여 LED를 제어해 봅시다. 아두이노에는 디지털 입출력핀과 아날로그 입

출력핀이 있습니다.

13 핀에서 전기를 보내서 LED 를 켜 봅시다. 그러기 위해서는 13 핀이 ⊕가 되고 GND가 ⊖가 됩니다. 13 핀은 디지털 핀이므로 ON과 OFF만 존재합니다. 다음과 같이 회로를 구성합니다.



【Sketch】

다음과 같이 코드를 작성합니다.

13 번 핀으로 전원을 보내기 위해서는 2 가지를 해야 합니다.

- ✓ setup()에서 13 번 핀을 디지털 출력핀으로 지정
- ✓ setup()에서 13 번 핀으로 전기를 보냄

【형식】

```
pinMode(핀번호,OUTPUT);
```

【예】

```
pinMode(13,OUTPUT);
```

13 핀으로 전기를 보내는 방법은 digitalWrite를 사용합니다.

【형식】

```
digitalWrite(핀번호, 신호);
```

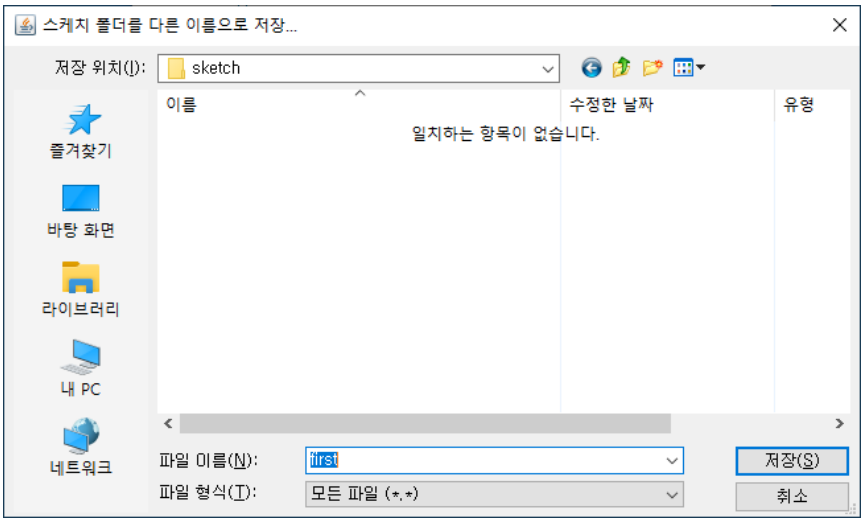
신호는 ON인 경우에는 1 또는 HIGH, off 인 경우에는 0 또는 LOW를 사용합니다.

【예】

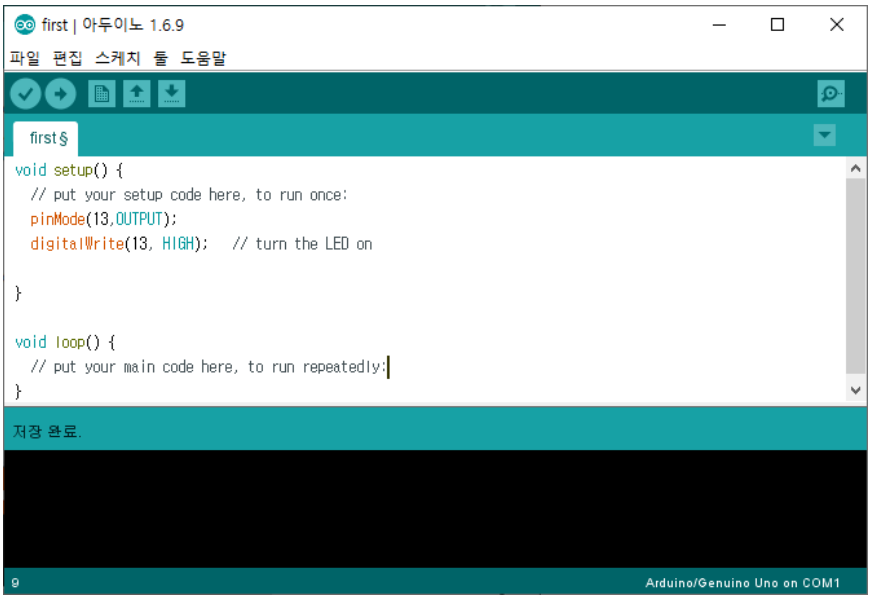
```
digitalWrite(13,1);
```

【소스 프로그램】

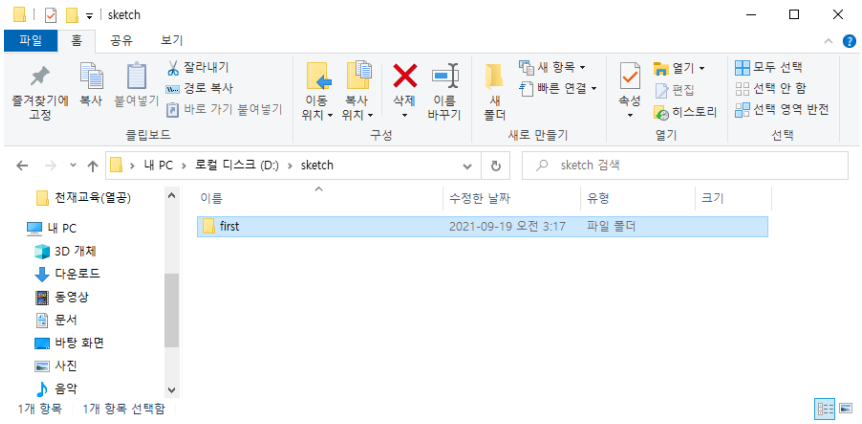
```
void setup() {
  // put your setup code here, to run once:
 pinMode(13,OUTPUT);
  digitalWrite(13,1);

}

void loop() {
  // put your main code here, to run repeatedly:
}
```



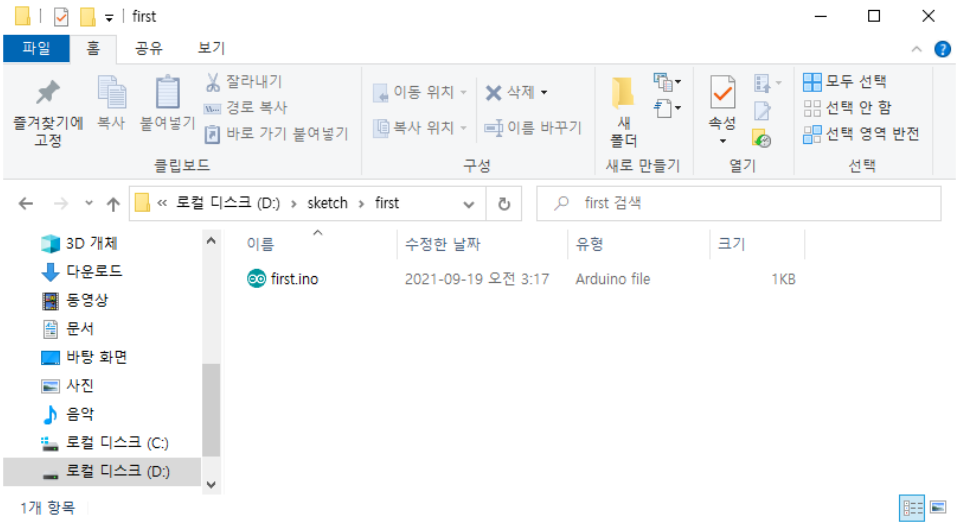
d:\sketch\first 이름으로 저장하였습니다. 이제 확인하여 보면



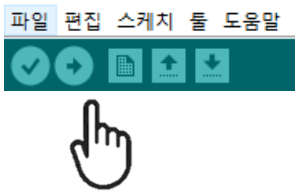
저장이 완료되었습니다. sketch 폴더에 가보면 first 파일이 있는 것이 아니라 first 폴더가 있습니다.



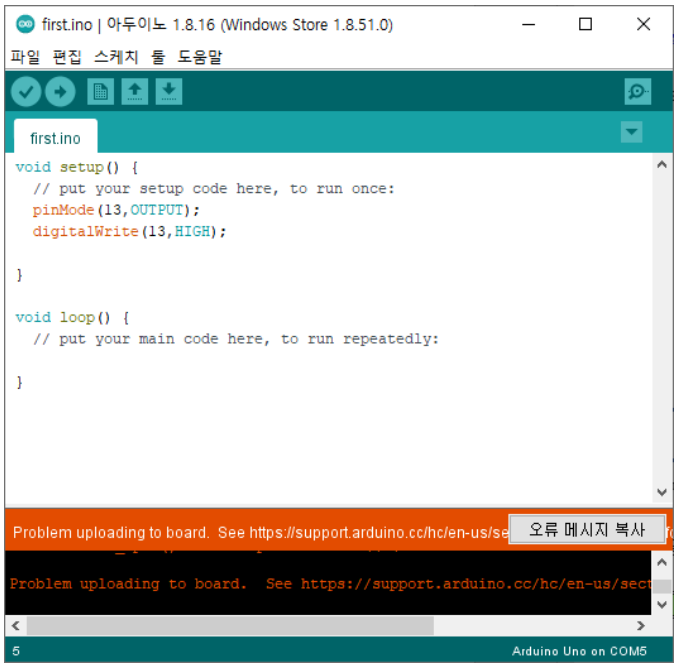
first 폴더로 들어가 봅시다.



first 폴더에 first.ino 파일이 있다는 것을 알 수 있습니다. 스케치에서는 파일을 저장하면 파일만 저장되는 것이 아니라 그 위치에 지정한 파일이름으로 폴더가 생성되고 그 폴더 안에 지정된 파일명.ino 이름으로 파일이 저장됩니다. 작성된 코드를 컴파일하고 아두이노에 전송하기 위해서 오른쪽 화살표 모양의 업로드 버튼을 클릭합니다.



다음 그림과 같이 에러가 발생하는 경우는 여러가지가 있겠지만, 보통 초보자에게는 접촉 불량인 경우가 많으므로 접촉 상태를 확인한 다음에 다시 업로드 합니다.



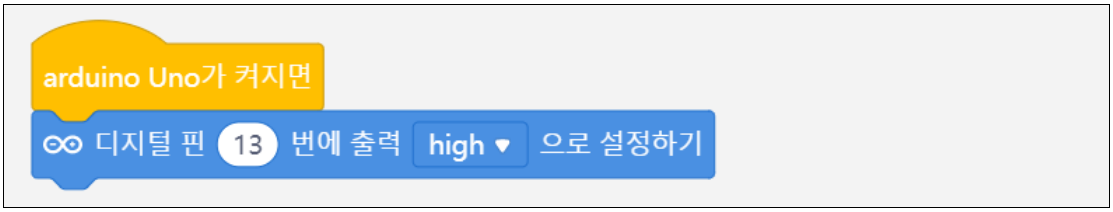
다음 화면이 나오면 성공적으로 아두이노에 업로드가 된 것입니다. 이런 상태에서는 외부 전원이 공급이 된다면 컴퓨터와 분리되어도 LED에 불이 들어올 것입니다.

【mblock】

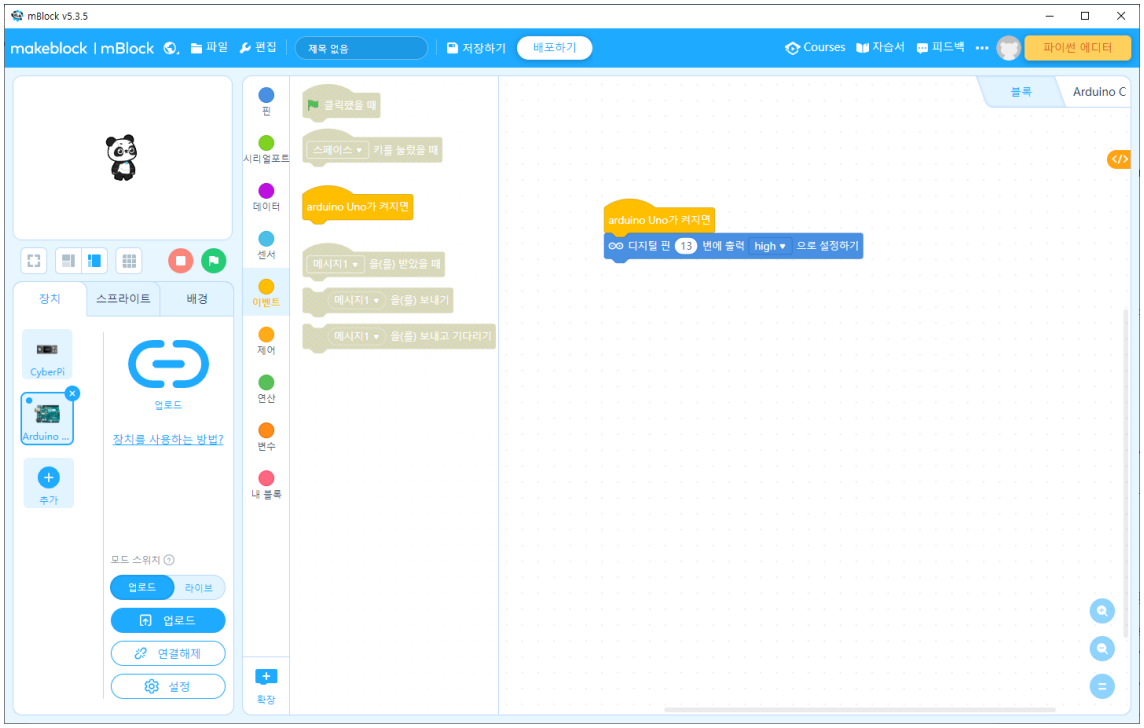
[이벤트-arduino Uno가 켜지면] 을 선택하여 코드창에 끌어다 놓습니다.

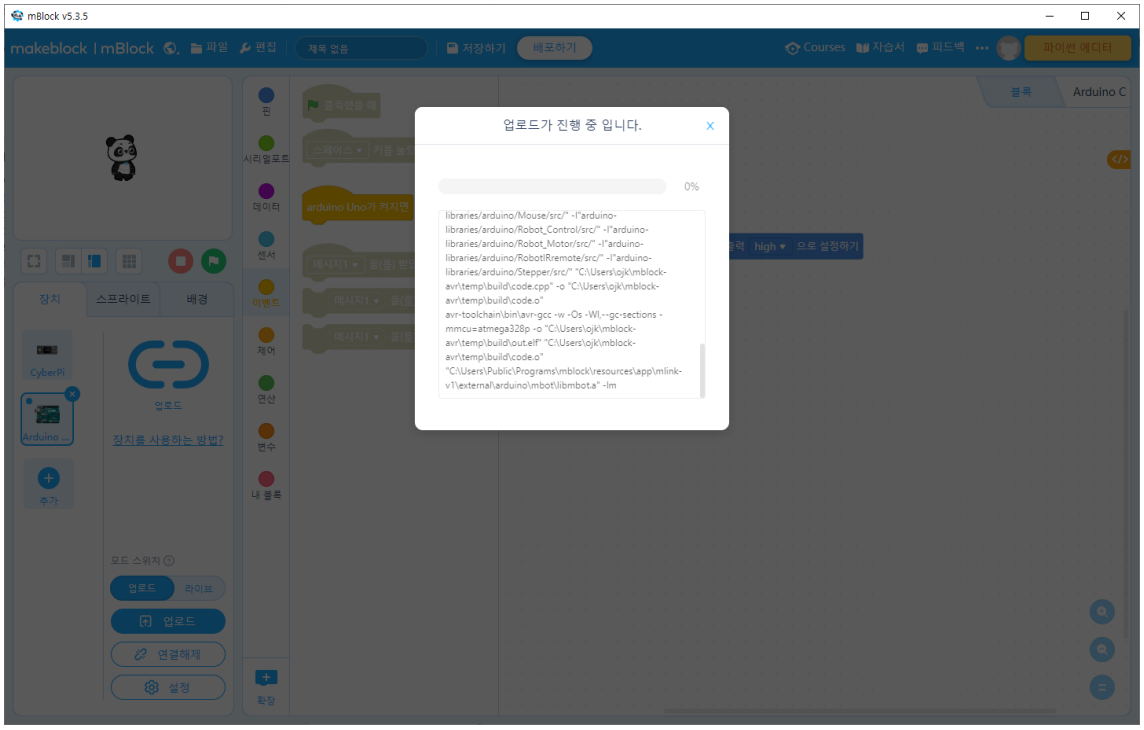
[핀-디지털 핀 9 번에 출력….]을 끌어나 놓습니다.

【코드 블록】



[업로드]를 클릭합니다.




mBlock v5.3.5
makeblock | mBlock
파일
편집
제목 없음
저장하기
배포하기
Courses
자습서
피드백
파이썬 에디터
블록
Arduino C
업로드가 진행 중 입니다.
0%
libraries/arduino/Mouse/src/" -I"arduino-libraries/arduino/Robot_Control/src/" -I"arduino-libraries/arduino/Robot_Motor/src/" -I"arduino-libraries/arduino/RobotIRremote/src/" -I"arduino-libraries/arduino/Stepper/src/" "C:\Users\ojk\mblock-avr\temp\build\code.cpp" -o "C:\Users\ojk\mblock-avr\temp\build\code.o"
avr-toolchain\bin\avr-gcc -w -Os -Wl,--gc-sections -mmcu=atmega328p -o "C:\Users\ojk\mblock-avr\temp\build\out.elf" "C:\Users\ojk\mblock-avr\temp\build\code.o" "C:\Users\Public\Programs\mblock\resources\app\mlink-v1\external\arduino\mbot\libmbot.a" -lm
장치
스프라이트
배경
CyberPi
업로드
장치를 사용하는 방법?
추가
모드 스위치
업로드
라이브
업로드
연결해제
설정
이벤트
제어
연산
변수
내 블록
확장
arduino Uno가 켜지면
high
으로 설정하기

③ LED ON/OFF

13 핀으로 LED 를 켜고 5 초 후에 꺼봅시다. LED를 켠 후에 5 초가 지나고 LED를 끕니다.

【Sketch】

스케치에서 시간의 경과는 delay()함수를 사용합니다. ( ) 안에 들어가는 숫자의 단위는 milesecond 입니다. 쉽게 말하면 delay(1000)은 1 초입니다.

【형식】

```
delay( 경과시간);
```

【예】

```
delay(5000);
```

LED를 끄는 명령은

digitalWrite(13, LOW); 또는 digitalWrite(13, 0);

입니다.

【소스 프로그램】

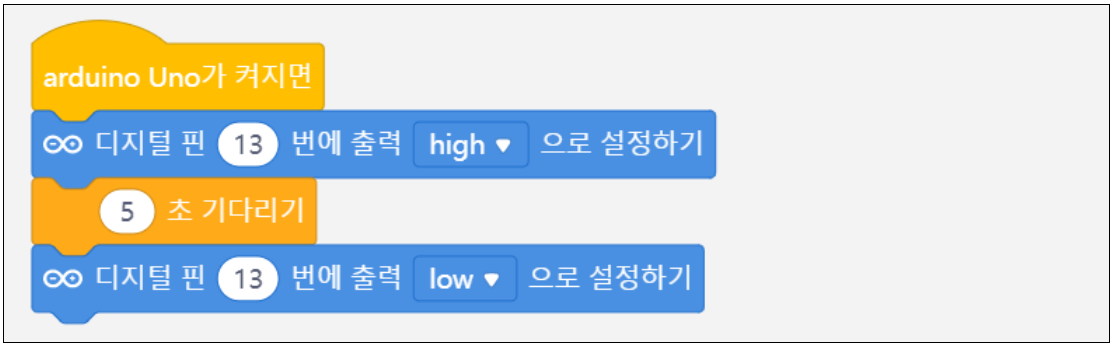
```
void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
  digitalWrite(13, HIGH);   // turn the LED on
  delay(5000);               // wait for 5 second
 digitalWrite(13, LOW);   // turn the LED off
 }

void loop() {
  // put your main code here, to run repeatedly:
  }
```

【mblock】

5 초 경과는 시간을 제어하는 것이므로 [제어] 블록에 있습니다. [제어-1 초 기다리기]를 끌어다 놓고 시간을 5 초로 변경합니다.

[핀-디지털 핀….]을 가져다 놓고 low로 설정합니다.

【코드 블록】



④ LED 점멸

13 핀으로 LED 를 켜고 1 초 후에 끄는, 깜박임 동작을 10 번 하고 끄는 동작을 구현해봅시다.

【Sketch】

【소스 프로그램】

```
//File Name : ex002_blink_01.ino
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  int i;
  for(i=0;i<5;i++){
    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage
level)
    delay(1000);              // wait for a second
    digitalWrite(13, LOW);    // turn the LED off by making the voltage
LOW
    delay(1000);
 }
 digitalWrite(13, LOW);
}
```

【mblock】

[제어-10 번 반복하기]를 끌어다 놓고

그 사이에 출력을 ON으로 하고, 1 초 기다리기

그 사이에 출력을 OFF으로 하고, 1 초 기다리기

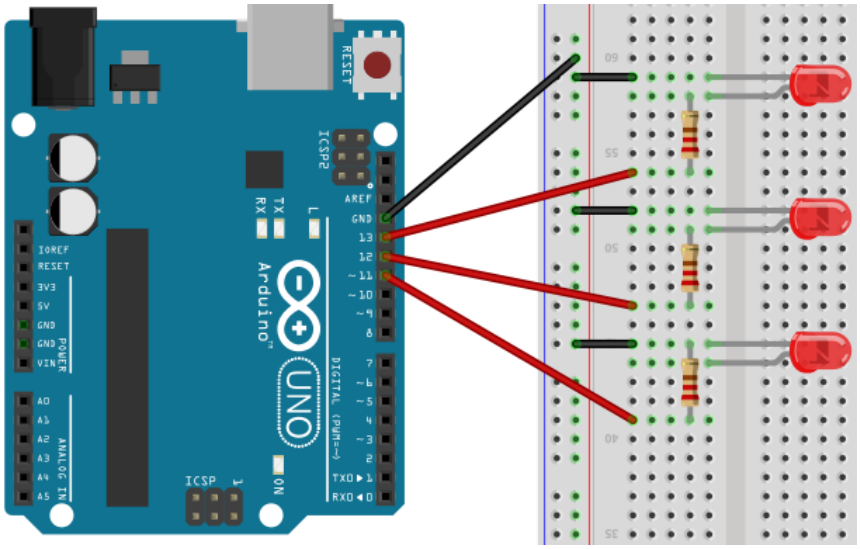
를 넣은 다음

마지막에 출력을 OFF으로 하는 블록을 넣습니다.

【코드 블록】

【2】신호등

신호등을 만들어 봅시다. 3 색 LED를 사용하여 신호등을 만드는 것이 타당하지만, 여기에서는 LED 3 개를 각각 빨강, 노랑, 파랑이라고 가정하고 빨강색을 5 초 ON, 녹색을 2 초 ON, 파랑색을 5 초 ON 하는 방식으로 프로그래밍하여 봅시다.



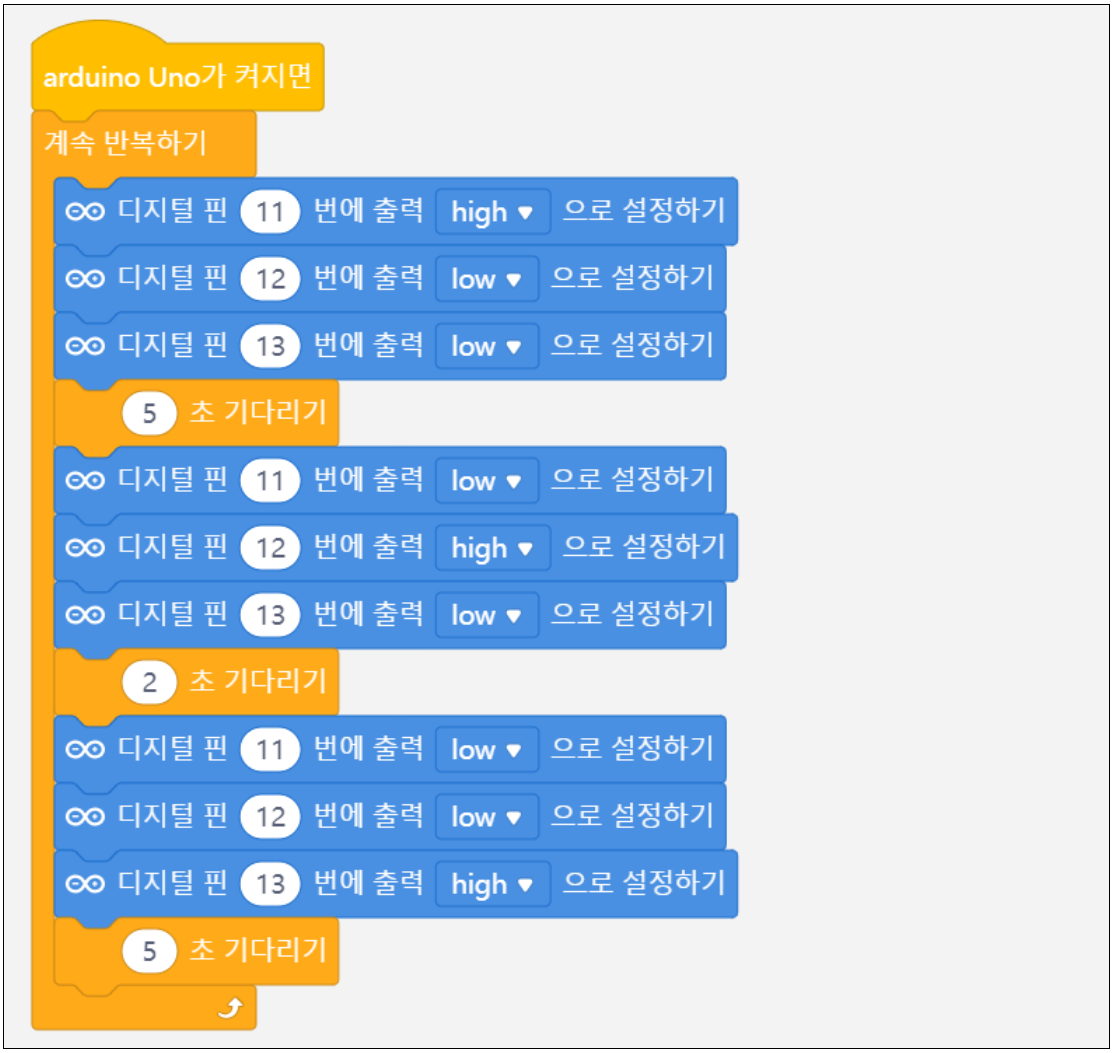
알고리즘

- 빨강 ON, 녹색 OFF, 파랑 OFF
- 5 초 대기
- 빨강 OFF, 녹색 ON, 파랑 OFF
- 2 초 대기
- 빨강 OFF, 녹색 OFF, 파랑 ON

스크레치 코드 삽입

【mblock】

【코드 블록】



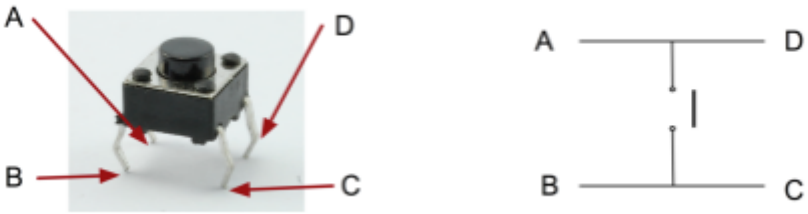
실행 동영상 : https://www.youtube.com/watch?v=Yij4CKb5GxM&list=PL0-gQtm6qnGIIb5cssjOg0uYkqEmyevNx&index=2

【3】LED 밝기 조절하기


Arduino Program
i 을(를) 0 로 정하기
5 번 반복하기
i = 255 까지 반복하기
set pwm pin 11 output as i
i 을(를) 1 만큼 바꾸기
0.01 초 기다리기
i = 0 까지 반복하기
set pwm pin 11 output as i
i 을(를) -1 만큼 바꾸기
0.01 초 기다리기


Arduino Program
i 을(를) 0 로 정하기
5 번 반복하기
i = 255 까지 반복하기
set pwm pin 11 output as i
i 을(를) 1 만큼 바꾸기
0.01 초 기다리기
i = 0 까지 반복하기
set pwm pin 11 output as i
i 을(를) -1 만큼 바꾸기
0.01 초 기다리기

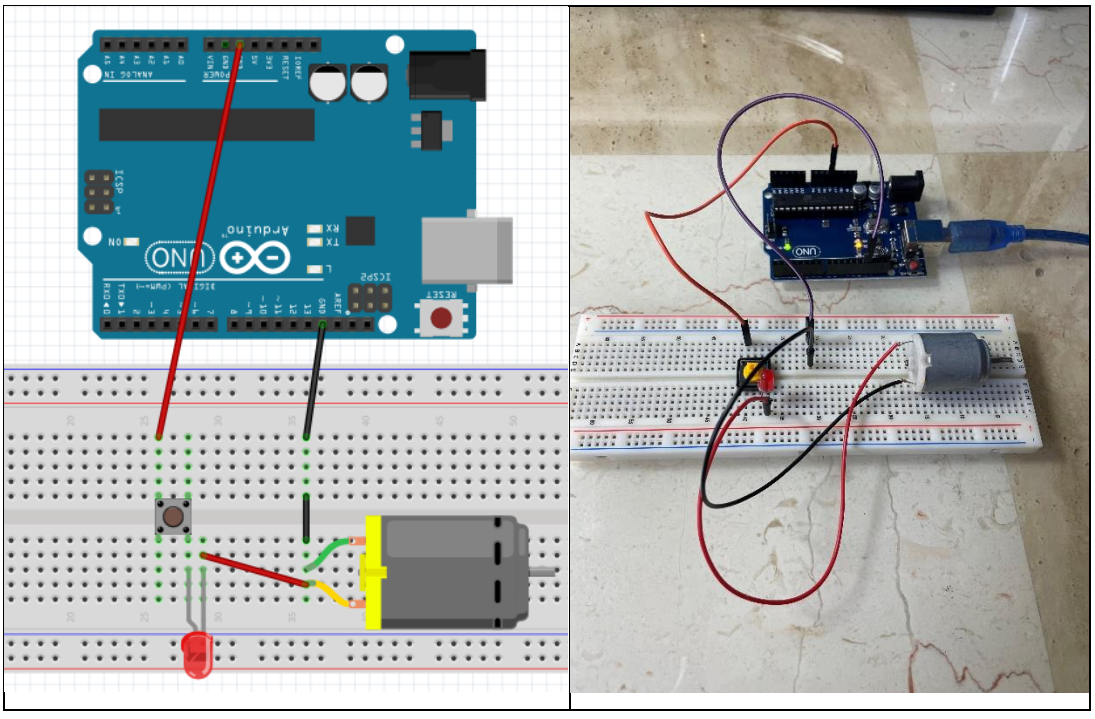
4. 푸시버튼



푸시 버튼은 마주 보는 핀(안 쪽으로 휘어져 있는 핀)끼리 서로 연결되어 있어, 버튼을 누르면 4 개가 다 연결이 되고 버튼을 떼면 같은 방향이니 대각선의 핀은 연결이 끊깁니다. 따라서 버튼을 눌렀을 때 전기가 흐르게 하려면 같은 방향에 있는 것끼리 연결하거나 대각선으로 연결하여야 합니다.

【1】자동차 시동 버튼

자동차는 전기를 넣어서 모터를 회전 시키고 그 힘으로 엔진을 가동시킵니다. 이를 시뮬레이션하는 회로를 만들어 봅시다. 푸시버튼을 누르면 LED가 켜지고, 모터가 회전하는 회로를 만들면 됩니다.

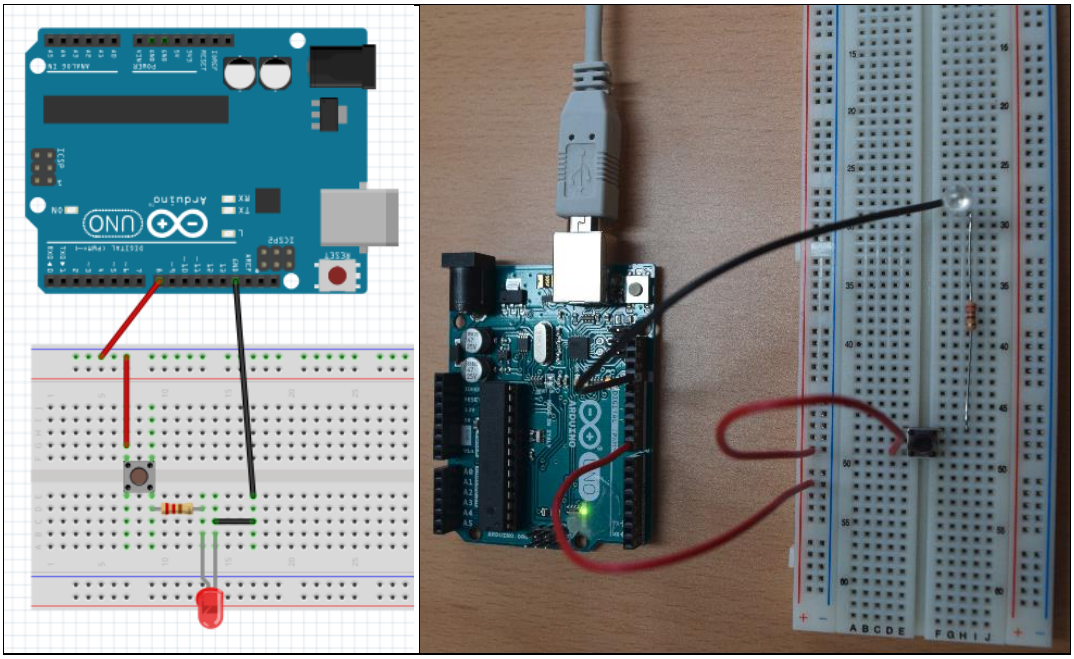


여기에서 사용하는 모터는 1.5V로도 충분히 구동되는 모터이어야 합니다. 일반적으로 사용하는 직류모터를 장착하는 경우에는 전기 부족으로 회전하지 않습니다.

실행 동영상 : https://www.youtube.com/watch?v=i5Q7mmuoDa4&list=PL0-gQtm6qnGIIb5cssjOg0uYkqEmyevNx&index=1

【2】 버튼 하나

8 번 핀으로 전기를 출력한 상태에서 버튼을 누르면 LED가 켜지고, 버튼에서 손을 떼면 꺼지는 프로그램을 만들어 봅시다.



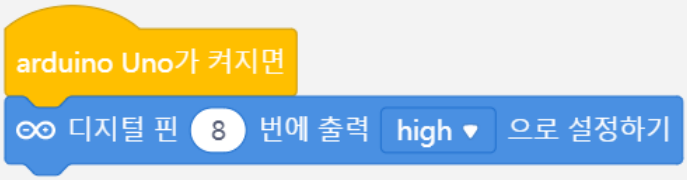
회로를 구성하고 8 번 핀으로 출력하도록 프로그래밍 하면 됩니다.

【Sketch】

```
// File Name : button_01.ino
void setup() {
  // initialize digital pin 8 as an output.
  pinMode(8, OUTPUT);
}
```

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(8, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
}
```

【mblock】



【3】 버튼 둘

버튼을 2 개를 설치하여, 버튼 하나를 누르면 켜지고 다른 버튼을 누르면 꺼지는 프로그램을 작성하여 봅시다.

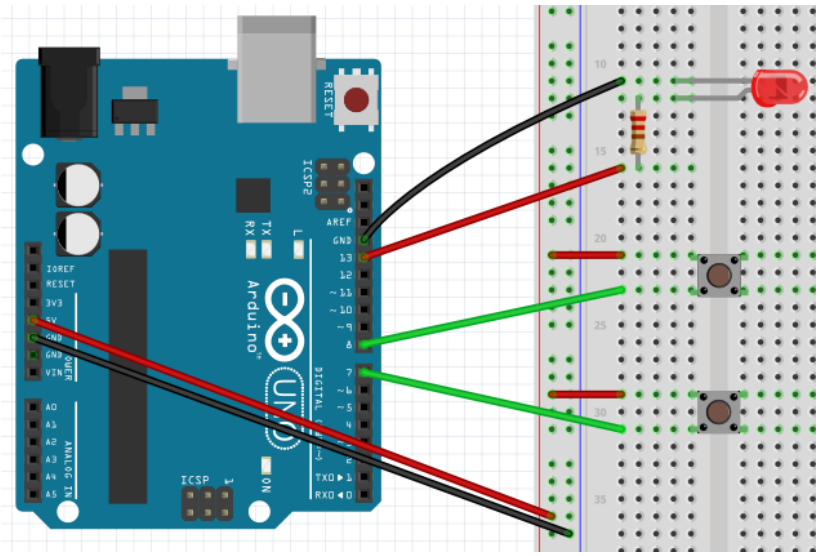
버튼 1 개는 7 번 핀에 다른 하나는 8 번 핀에 연결하여 입력을 받고, LED는 13 번 핀에 연결하여 8 번 핀에 연결된 버튼을 누르면 LED가 켜지고, 7 번 핀에 연결된 버튼을 누르면 LED가 꺼지게 만들어 봅시다.

전기는 5V 핀에서 흘러나와 버튼을 거쳐서 8 번핀과 7 번 핀으로 들어가게 회로를 구성하면 버튼을 클릭하면 8 번핀과 7 번핀에 전기가 흘러들어갑니다. 이를 감지하여

8 번 핀에 전기가 감지되면 13 번 핀으로 ON을 보내고

7 번 핀에 전기가 감지되면 13 번 핀으로 OFF를 보내면 됩니다.

이를 위해서 8 번과 7 번 핀은 디지털 입력핀으로 설정하고, 13 번 핀은 출력으로 설정하여야 합니다.



위와 같이 회로를 구성하면 정상적으로 작동할 것으로 예상됩니다.

【Sketch】

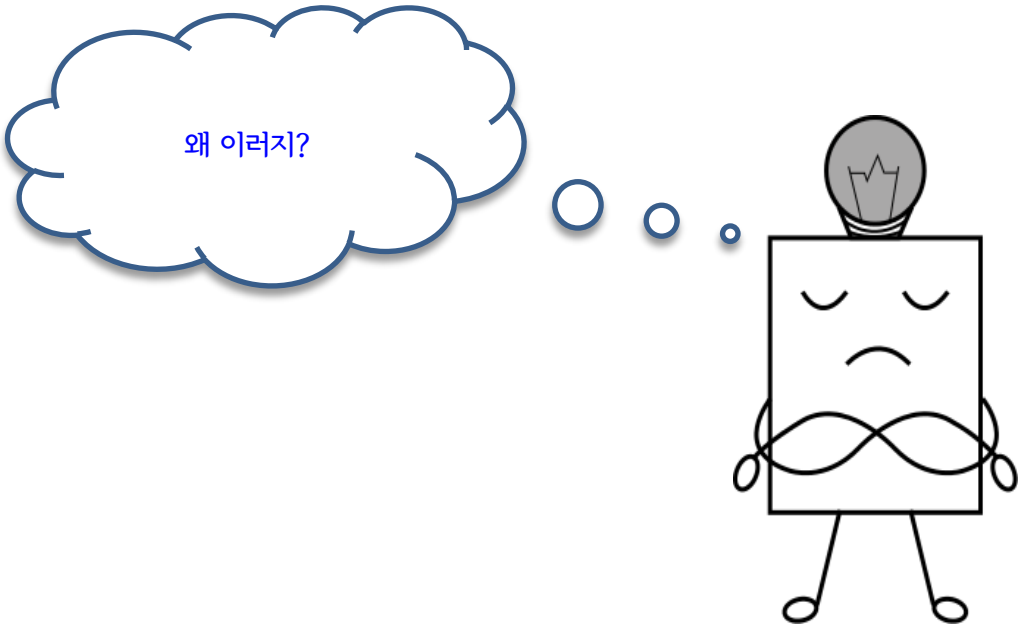
【소스 프로그램】

```
void setup() {
   pinMode(7,INPUT);
   pinMode(8,INPUT);
   pinMode(13,OUTPUT);
   digitalWrite(13,0);
}

void loop() {
   if(digitalRead(8)==1){
   digitalWrite(13,1);
   delay(1000);
   }

  if(digitalRead(7)==1){
   digitalWrite(13,0);
   delay(1000);
  }
}
```

【코드 블록】

정상적으로 실행될 것으로 예상하고, 실제로 실행하면…

그러나… 버튼의 눌러짐 여부에 상관 없이 LED에 불이 들어오거나 들어오지 않거나 합니다.

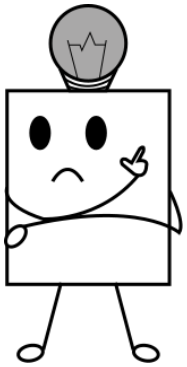


핀을 입력으로 설정하는 경우에는, 핀 주위에는 항상 미량의 전기가 흐르기 때문에(전자회로에서 어

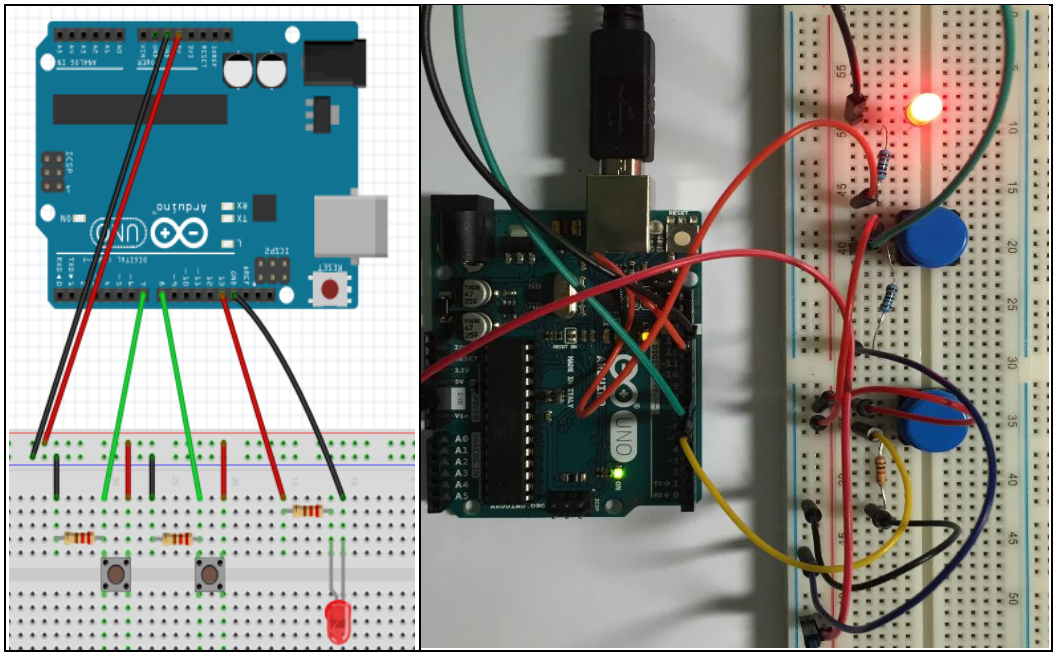
쩔 수 없이 발생하는 현상) 실제적으로 5V가 입력이 되지 않더라고 입력이 된 것으로 인식할 수 있습니다. 이런 현상을 **『플로팅 현상』**이라고 합니다.

이런 현상을 해결하기 위해서는 미량의 전기를 제거하여야 합니다. 회로에 흐르는 미량의 전기를 제거하는 방법은 이를 GND에 흘려보내는 방법이 있습니다. 따라서 입력핀을 GND에 연결하면 해결이 됩니다. 그러나 이렇게 하면 실제로 버튼을 눌렀을 때 8 번 핀이나 7 번 핀으로 전기가 흐르지 않고 GND에 곧바로 흘러가버리기 때문에 목적을 달성할 수 없습니다.

따라서 입력핀과 GND사이에 저항을 두면 해결이 됩니다. 버튼이 눌러지지 않았을 때는 입력 핀 주변의 미량의 전기가 GND로 흘러들어 가고, 그래서 입력 핀의 상태가 OFF로 됩니다.



버튼을 눌렀을 때는 5V에서 출발한 전류가 버튼을 통과한 다음 두갈래 길에 서게 됩니다. 하나는 저항이 없는 입력 핀으로 들어가는 길이고 하나는 저항이 있는 GND로 들어가는 길입니다. 당연히 저항이 없는 입력 핀으로 들어가는 길을 선택하게 됩니다. 그래서 다음과 같이 회로를 구성하면 됩니다.

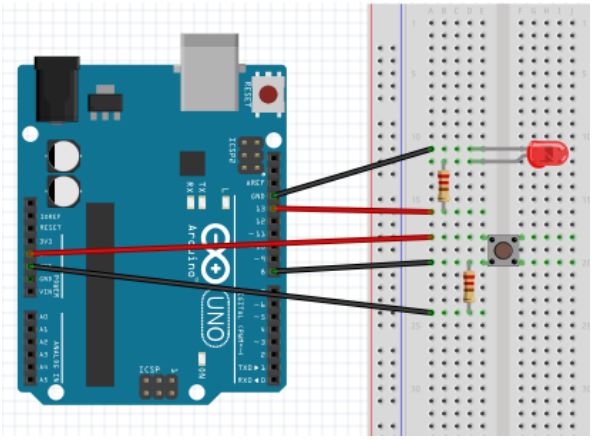


이러한 방법을 『풀다운』이라고 합니다. 풀다운을 하는데 보통 사용하는 저항은 10kΩ입니다. 이와 같이 풀다운은 입력 핀의 전압을 평상시에 LOW로 고정하는 것을 말합니다.

풀다운
입력핀과 GND 핀 사이에 저항을 둔다.

【4】토글 버튼

버튼 하나로 한 번 누르면 켜지고 한 번 더 누르면 꺼지는 프로그램을 작성하여 봅시다. 하나의 버튼으로 ON, OFF가 반복되는 버튼을 토글 버튼이라 합니다.



푸시버튼은 누르면 연결이 되고 떼면 연결이 해제되므로, 하나의 버튼으로 ON,OFF를 변경하는 것은 불가능합니다. 따라서 이를 소프트웨어로 구현하여야 합니다.

- 버튼 상태를 OFF로 지정하고, 이 상태에서
- 버튼을 한 번 누르면 버튼 상태 OFF를 참조하여,

  LED를 ON하는 명령을 주고 버튼 상태를 ON으로 바꾸어 줍니다.
- 버튼을 한 번 더 눌렀을 때는 버튼 상태(ON)을 참고하여,

  LED를 OFF하는 명령을 주고 버튼 상태를 OFF로 바꾸어 줍니다.

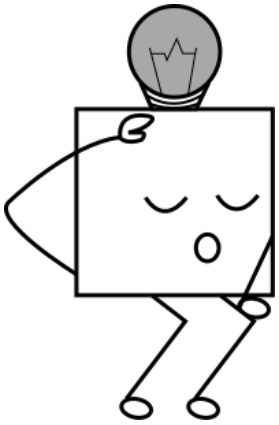
이러한 동작을 반복하면 LED가 켜져 있는 경우에는 버튼 상태가 ON이 되고, LED가 꺼져 있는 경우에는 버튼 상태가 OFF가 됩니다. 이를 코드로 바꾸면 다음이 됩니다.

【Sketch】

```
if( (digitalRead(8)==1) & (status==0)){
        digitalWrite(13,1);
        status = 1;
 }
```

```
    if( (digitalRead(8)==1) & (status==1)){
        digitalWrite(13,0);
        status = 0;
}
```

실행을 시키면 원하는 대로 동작이 되지 않는다는 것을 알 수 있습니다.



두개의 블록은 아주 빠르게 실행이 됩니다.

푸시버튼을 누르면

```
if( (digitalRead(8)==1) & (status==0)){
        digitalWrite(13,1);
        status = 1;
```

명령이 아주 빠르게 실행됩니다. 푸시 버튼을 누르고 바로 떼면 사람은 곧바로 뗀다고 생각하지만 사람이 떼기 이전에

```
 if( (digitalRead(8)==1) & (status==1)){
        digitalWrite(13,0);
        status = 0;
 }
```

명령이 실행되고 stuatus가 0 으로 설정되고, 푸시 버튼을 떼기 이전에

```
if( (digitalRead(8)==1) & (status==0)){
        digitalWrite(13,1);
        status = 1;
```

명령이 실행되고 stuatus가 1 으로 설정되고, 푸시 버튼을 떼기 이전에 또 앞의 명령이 실행됩니다. 따라서 푸시 버튼을 한 번 누르면 LED가 켜기고 또 한 번 누르면 LED가 꺼지게 하는 것이 되지 않습니다. 그러므로 사람이 버튼을 누르고 떼는 시간 만큼 프로그램 실행이 멈추고 기다려야 하여야 합니다. 그래서 delay(1000);을 주어야 합니다.

【소스 프로그램】

```
int status = 0;
void setup(){
    pinMode(8,INPUT);
    pinMode(13,OUTPUT);
}

void loop(){
    if( (digitalRead(8)==1) & (status==0)){
        digitalWrite(13,1);
        status = 1;
        delay(1000);
    }
    if( (digitalRead(8)==1) & (status==1)){
        digitalWrite(13,0);
        status = 0;
        delay(1000);
    }
}
```

물론 delay(1000)이므로 푸시 버튼을 누른 다음 다시 한 번 누를 때는 적어도 1 초가 지나야 작동이 되겠죠. LED를 켠 다음 1 초가 되기 전에 한 번 더 누르면 꺼지지 않습니다.

한 편

```
if( (digitalRead(8)==1) & (status==0)){
        digitalWrite(13,1);
        status = 1;
        delay(1000);
    }
    if( (digitalRead(8)==1) & (status==1)){
        digitalWrite(13,0);
        status = 0;
        delay(1000);
    }
```

는 피지컬 프로그래밍에서 큰 가치를 발휘하는 3 항 연산자를 사용하면 다음과 같이 바꿀 수 있습니다.

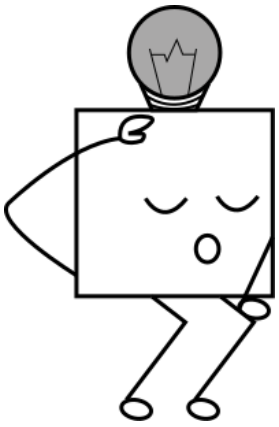
```
if(digitalRead(8)==1){
    status =    (status==0) ? 1 : 0;
    digitalWrite(13,status);
    delay(1000);
  }
```

연산자 우선 순위가 3 항연산자가 먼저이고 대입연산자가 그 다음에 실행되기 때문에 status 값이 무엇인지를 판단하고 0 이면 3 항연산자의 값을 1 로 하고 0 이 아니면 그 값을 그대로 0 으로 둡니다. 그런 다음 1 또는 0 을 status에 대입하기 때문에 status에 기억되는 값이 0 일 때는 1 로 바뀌고 1 일 때는 0 으로 바뀝니다. 결국 이러한 방식을 사용하면 변수값을 토글할 수 있습니다.

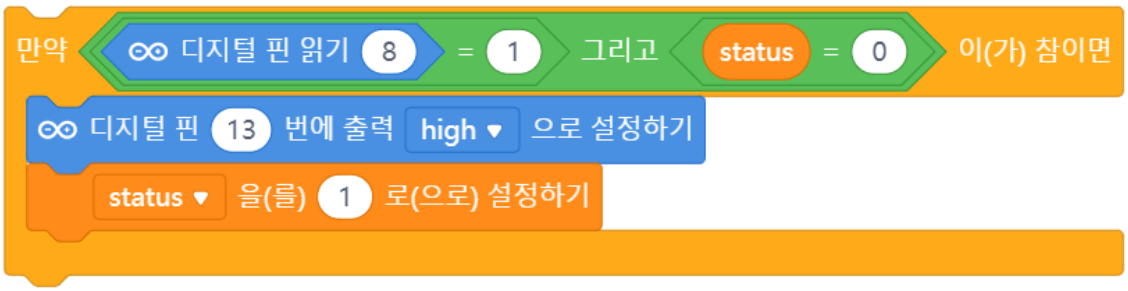
다시 한 번 순차적으로 살펴보면 status 초기값이 0 이기 때문에 이 명령이 실행되면 status 값이 1 로 바뀌고 digitalWrite(13,1); 명령이 실행됩니다. 그 다음에 한 번 더 실행되면 status 값이 1 이기 때문에 조건이 거짓이 되어서 3 항연산자의 결과는 0 으로 바뀌고 digitalWrite(13,0);이 실행되어 LED는 꺼집니다.

【mblock】

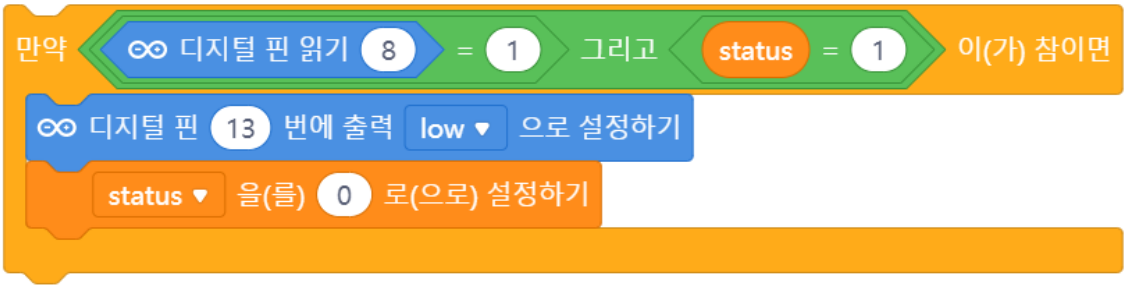
실행을 시키면 원하는 대로 동작이 되지 않는다는 것을 알 수 있습니다.



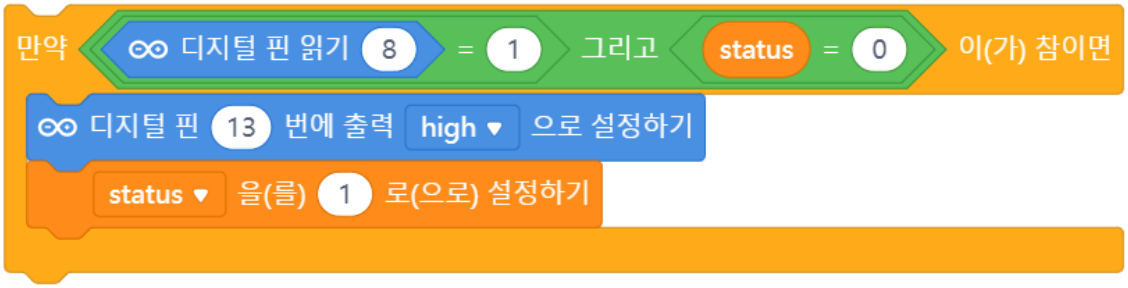
두개의 블록은 아주 빠르게 실행이 됩니다. 푸시버튼을 누르면



명령이 아주 빠르게 실행됩니다. 푸시 버튼을 누르고 바로 떼면 사람은 곧바로 뗀다고 생각하지만 사람이 떼기 이전에



명령이 실행되고 stuatus가 0 으로 설정되고, 푸시 버튼을 떼기 이전에

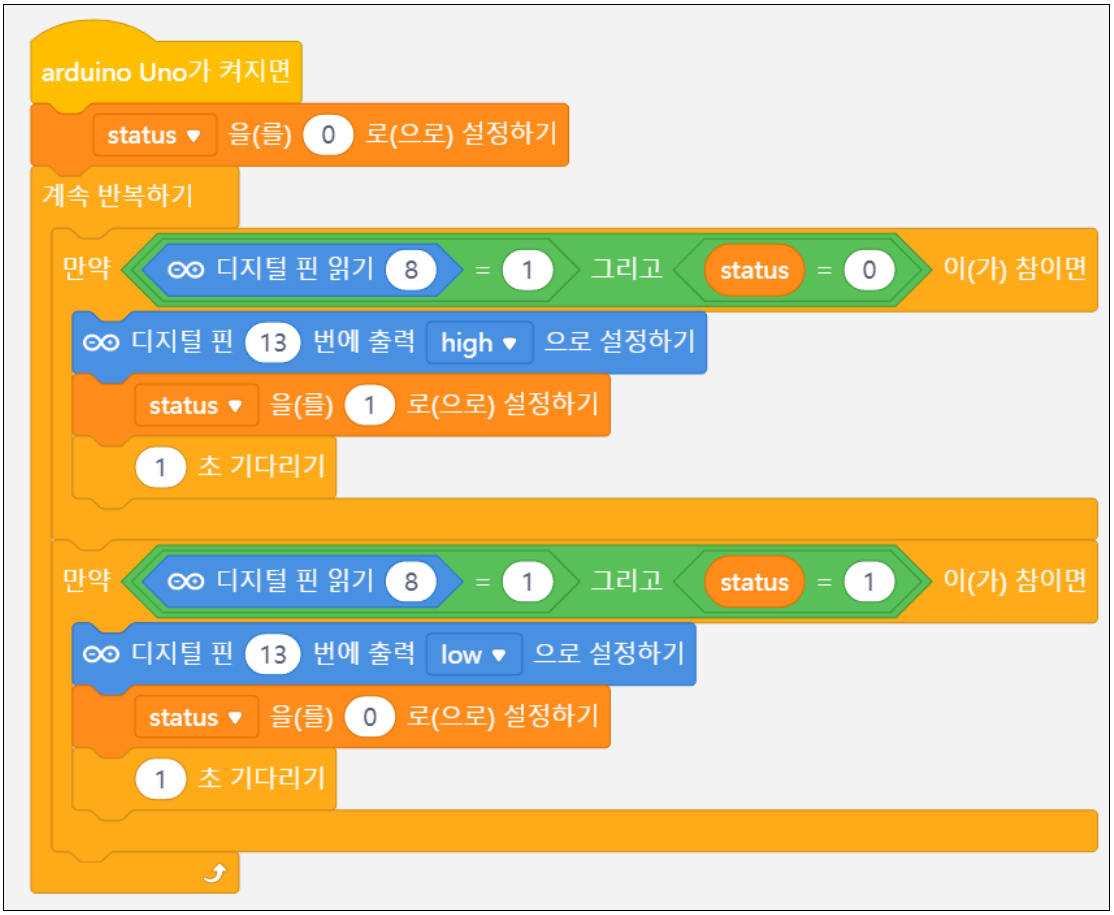


명령이 실행되고 stuatus가 1 으로 설정되고, 푸시 버튼을 떼기 이전에 또 앞의 명령이 실행됩니다. 따라서 푸시 버튼을 한 번 누르면 LED가 켜기고 또 한 번 누르면 LED가 꺼지게 하는 것이 되지 않습니다. 그러므로 사람이 버튼을 누르고 떼는 시간 만큼 프로그램 실행이 멈추고 기다려야 하여야 합니다.

따라서 

를 주어야 합니다.

【코드 블록】


arduino Uno가 켜지면
status ▾ 을(를) 0 로(으로) 설정하기
계속 반복하기
만약 디지털 핀 읽기 8 = 1 그리고 status = 0 이(가) 참이면
디지털 핀 13 번에 출력 high ▾ 으로 설정하기
status ▾ 을(를) 1 로(으로) 설정하기
1 초 기다리기
만약 디지털 핀 읽기 8 = 1 그리고 status = 1 이(가) 참이면
디지털 핀 13 번에 출력 low ▾ 으로 설정하기
status ▾ 을(를) 0 로(으로) 설정하기
1 초 기다리기


arduino Uno가 켜지면
status ▾ 을(를) 0 로(으로) 설정하기
계속 반복하기
만약 디지털 핀 읽기 8 = 1 그리고 status = 0 이(가) 참이면
디지털 핀 13 번에 출력 high ▾ 으로 설정하기
status ▾ 을(를) 1 로(으로) 설정하기
만약 디지털 핀 읽기 8 = 1 그리고 status = 1 이(가) 참이면
디지털 핀 13 번에 출력 low ▾ 으로 설정하기
status ▾ 을(를) 0 로(으로) 설정하기

5. 시리얼

Serial communication on pins TX/RX uses TTL logic levels (5V or 3.3V depending on the board). Don't connect these pins directly to an RS232 serial port; they operate at +/- 12V and can damage your Arduino board.
Serial is used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): **Serial**. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().

The Arduino Mega has three additional serial ports: **Serial1** on pins 19 (RX) and 18 (TX), **Serial2** on pins 17 (RX) and 16 (TX), **Serial3** on pins 15 (RX) and 14 (TX). To use these pins to communicate with your personal computer, you will need an additional USB-to-serial adaptor, as they are not connected to the Mega's USB-to-serial adaptor. To use them to communicate with an external TTL serial device, connect the TX pin to your device's RX pin, the RX to your device's TX pin, and the ground of your Mega to your device's ground.

The Arduino Due has three additional 3.3V TTL serial ports: **Serial1** on pins 19 (RX) and 18 (TX); **Serial2** on pins 17 (RX) and 16 (TX), **Serial3** on pins 15 (RX) and 14 (TX). Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip, which is connected to the USB debug port. Additionally, there is a native USB-serial port on the SAM3X chip, SerialUSB'.

The Arduino Leonardo board uses **Serial1** to communicate via TTL (5V) serial on pins 0 (RX) and 1 (TX). **Serial** is reserved for USB CDC communication. For more information, refer to the Leonardo getting started page and hardware page.

시리얼 설정

```
Serial.begin(baud rate);
```

예)

```
Serial.begin(9600);
```

시리얼에 문자 출력

```
Serial.println("문자열");
```

예)

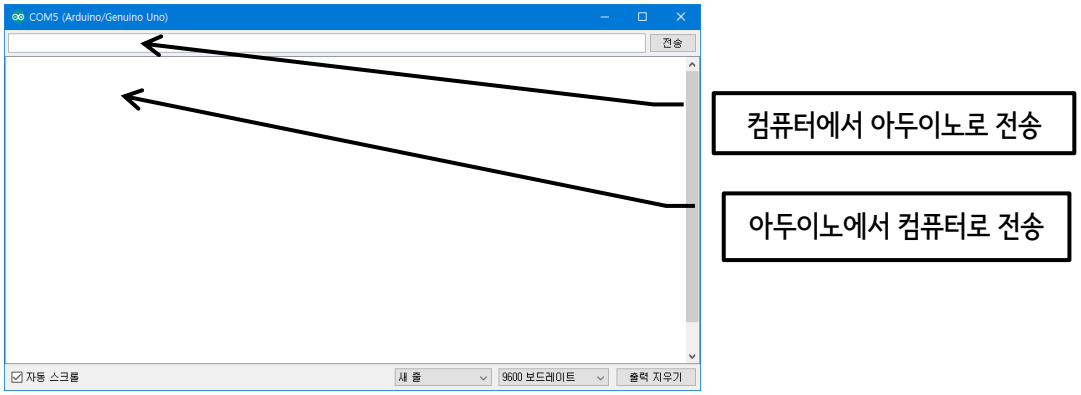
```
Serial.println("LED Off");
```

시리얼에 변수값 출력

```
Serial.println(변수);
```

```
Serial.println(dice);
```

시리얼에 변수값을 출력하는 것은 아두이노가 컴퓨터에 실행되고 있는 스케치 프로그램의 시리얼 창에 데이터를 전송하는 것입니다. 컴퓨터가 아두이노에 전송하는 것이 아닙니다. 컴퓨터에서 실행되고 있는 스케치 프로그램의 시리얼 창에서 아두이노에 데이터를 전송하려면 다음과 같이 시리얼 창의 텍스트박스에 입력하여야 합니다.



13 핀에 LED를 연결하여 3 초 간격으로 점멸하고 시리얼에 LED on, LED off 를 표시하는 프로그램을 만들어 봅시다.

```
#define LED13 13

void setup() {
  pinMode(LED13, OUTPUT);
  Serial.begin(9600);
}


void loop() {

  digitalWrite(LED13, HIGH);
  Serial.println("LED On");
  delay(3000);

  digitalWrite(LED13, LOW);
  Serial.println("LED Off");
  delay(3000);

}
```

arduino_multibyte_serial_example_1.pde

```
/* ---------------------------------------------
* SERIAL COM - HANDELING MULTIPLE BYTES inside ARDUINO - 01_simple
version
* by beltran berrocal
*
* this prog establishes a connection with the pc and waits for it to send
him
* a long string of characters like "hello Arduino!".
* Then Arduino informs the pc that it heard the whole sentence
*
* this is the first step for establishing sentence long conversations
```

```
between arduino and the pc.
* serialRead() reads one byte at a time from the serial buffer.
* so in order to print out the whole sentence at once
* (it is actually still printing one byte at a time but the pc will
receive it
* not interupted by newLines or other printString inside you loop)
* You must loop untill there are bytes in the serial buffer and
* and print right away that byte you just read.
* after that the loop can continue it's tasks.
*
* created 15 Decembre 2005;
* copyleft 2005 Progetto25zero1  <http://www.progetto25zero1.com>
*
* ---------------------------------------------------- */

int  serIn; //var that will hold the bytes in read from the serialBuffer

void setup() {
  Serial.begin(9600);

}

//auto go_to_the_line function
//void printNewLine() {
//  Serial.print(13, BYTE);
//  Serial.print(10, BYTE);
//}


void loop () {
  //simple feedback from Arduino  Serial.println("Hello World");

  // only if there are bytes in the serial buffer execute the following
code
  if(Serial.available()) {
    //inform that Arduino heard you saying something
    Serial.print("Arduino heard you say: ");
```

```
    //keep reading and printing from serial untill there are bytes in the
serial buffer
     while (Serial.available()>0){
        serIn = Serial.read();    //read Serial
        Serial.print(serIn, BYTE);     //prints the character just read
     }

    //the serial buffer is over just go to the line (or pass your
favorite stop char)
    Serial.println();
  }

  //slows down the visualization in the terminal
  delay(1000);
}

 arduino_multibyte_serial_example_2.pde
/* ---------------------------------------------
* SERIAL COM - HANDELING MULTIPLE BYTES inside ARDUINO - 01_simple
version
* by beltran berrocal
*
* this prog establishes a connection with the pc and waits for it to send
him
* a long string of characters like "hello Arduino!".
* Then Arduino informs the pc that it heard the whole sentence
*
* this is the first step for establishing sentence long conversations
between arduino and the pc.
* serialRead() reads one byte at a time from the serial buffer.
* so in order to print out the whole sentence at once
* (it is actually still printing one byte at a time but the pc will
receive it
* not interupted by newLines or other printString inside you loop)
* You must loop untill there are bytes in the serial buffer and
```

```
* and print right away that byte you just read.
* after that the loop can continue it's tasks.
*
* created 15 Decembre 2005;
* copyleft 2005 Progetto25zero1  <http://www.progetto25zero1.com>
*
* ---------------------------------------------------- */

int  serIn; //var that will hold the bytes in read from the serialBuffer

void setup() {
  Serial.begin(9600);

}

//auto go_to_the_line function
//void printNewLine() {
//  Serial.print(13, BYTE);
//  Serial.print(10, BYTE);
//}


void loop () {
  //simple feedback from Arduino  Serial.println("Hello World");

  // only if there are bytes in the serial buffer execute the following
code
  if(Serial.available()) {
    //inform that Arduino heard you saying something
    Serial.print("Arduino heard you say: ");

    //keep reading and printing from serial untill there are bytes in the
serial buffer
     while (Serial.available()>0){
        serIn = Serial.read();    //read Serial
        Serial.print(serIn, BYTE);     //prints the character just read
     }
```

```
    //the serial buffer is over just go to the line (or pass your
favorite stop char)
    Serial.println();
  }

  //slows down the visualization in the terminal
  delay(1000);
}
 arduino_multibyte_serial_example_3.cpp
/* ---------------------------------------------
* SERIAL COM - HANDELING MULTIPLE BYTES inside ARDUINO - 03_function
development
* by beltran berrocal
*
* this prog establishes a connection with the pc and waits for it to send
him
* a long string of characters like "hello Arduino!".
* Then Arduino informs the pc that it heard the whole sentence
*
* the same as examlpe 02 but it deploys 2 reusable functions.
* for doing the same job.
* readSerialString() and  printSerialString()
* the only problem is that they use global variables instead of getting
them passed
* as parameters. this means that in order to reuse this code you should
also copy
* the 4 variables instantiated at the beginning of the code.
* Another problem is that if you expect more than one string at a time
* you will have to duplicate and change names to all variables as well as
the functions.
* Next version should have the possibility to pass the array as a
parameter to the function.
*
* created 15 Decembre 2005;
* copyleft 2005 Progetto25zero1  <http://www.progetto25zero1.com>
```

```
*
* ------------------------------------------------ */

int  serIn;             // var that will hold the bytes-in read from the
serialBuffer
char serInString[100];  // array that will hold the different bytes
100=100characters;
                        // -> you must state how long the array will be
else it won't work.
int  serInIndx  = 0;    // index of serInString[] in which to insert the
next incoming byte
int  serOutIndx = 0;    // index of the outgoing serInString[] array;


/*read a string from the serial and store it in an array
//you must supply the array variable and the index count
void readSerialString (char *strArray, int indx) {
    int sb;                                 //declare local serial byte
before anything else
    Serial.print("reading Serial String: ");
    if(serialAvailable()) {
       while (serialAvailable()){
          sb = serialRead();
          strArray[indx] = sb;
          indx++;
          serialWrite(sb);
       }
    }
    Serial.println();
}
*/

//read a string from the serial and store it in an array
//this func uses globally set variable so it's not so reusable
//I need to find the right syntax to be able to pass to the function 2
parameters:
// the stringArray and (eventually) the index count
```

```
void readSerialString () {
    int sb;
    if(Serial.available()) {
       //Serial.print("reading Serial String: ");     //optional
confirmation
       while (Serial.available()){
          sb = Serial.read();
          serInString[serInIndx] = sb;
          serInIndx++;
          //serialWrite(sb);                         //optional
confirmation
       }
       //Serial.println();
    }
}

//print the string all in one time
//this func as well uses global variables
void printSerialString() {
   if( serInIndx > 0) {
      Serial.print("Arduino memorized that you said: ");
      //loop through all bytes in the array and print them out
      for(serOutIndx=0; serOutIndx < serInIndx; serOutIndx++) {
          Serial.print( serInString[serOutIndx] );    //print out the
byte at the specified index
          //serInString[serOutIndx] = "";            //optional: flush
out the content
      }
      //reset all the functions to be able to fill the string back with
content
      serOutIndx = 0;
      serInIndx  = 0;
      Serial.println();
   }

}
```

```
void setup() {
  Serial.begin(9600);
  Serial.println("Hello World");
}

void loop () {
  //simple feedback from Arduino

  //read the serial port and create a string out of what you read
  //readSerialString(serInString, serInIndx);
  readSerialString();

  //do somenthing else perhaps wait for other data or read another Serial
string
  Serial.println ("------------ arduino is doing somenthing else ");

  //try to print out collected information. it will do it only if there
actually is some info.
  printSerialString();

  //slows down the visualization in the terminal
  delay(2000);
}
```

시리얼에 'on'을 입력하면 10 번 핀에 연결된 LED가 켜지고 시리얼에 'off'를 입력하면 LED가 커지는 프로그램을 만들어 보자.

[소스프로그램]

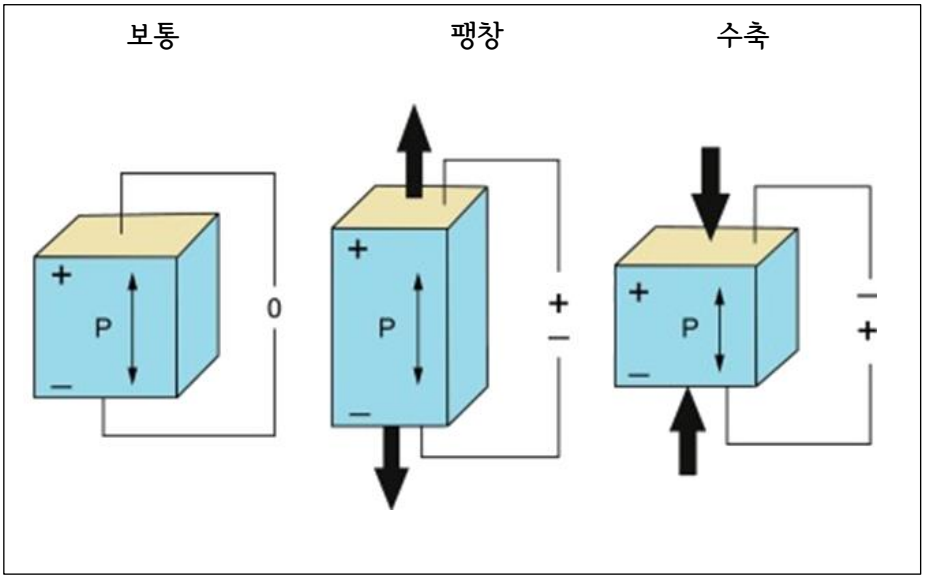
```
#define LED10 10

void uf_ledOn(int delayTime);
void uf_ledOff(int delayTime);

void setup() {
  pinMode(LED10, OUTPUT);
  Serial.begin(9600);

  digitalWrite(10,0);
}

void loop() {
  int i, j;
  char inputChar[10];
  int delayTimes=3000;
  Serial.println("Hello OJK");


  for(i=0;i<10;i++){    //serial read
     inputChar[i]=Serial.read();
  }

 if( inputChar[0]=='o' &&  inputChar[1]=='n' )
 {
    Serial.println("turn on LED");
    uf_ledOn(delayTimes);
  }

 if( inputChar[0]=='o' &&  inputChar[1]=='f' &&  inputChar[2]=='f' )
 {
    Serial.println("turn off LED");
    uf_ledOff(delayTimes);
 }
```

```
} // End of void loo()

void uf_ledOn(int delayTime){
  digitalWrite(LED10,HIGH);
  delay(delayTime);
}

void uf_ledOff(int delayTime){
  digitalWrite(LED10,LOW);
  delay(delayTime);
}
```

6. 피에조 스피커

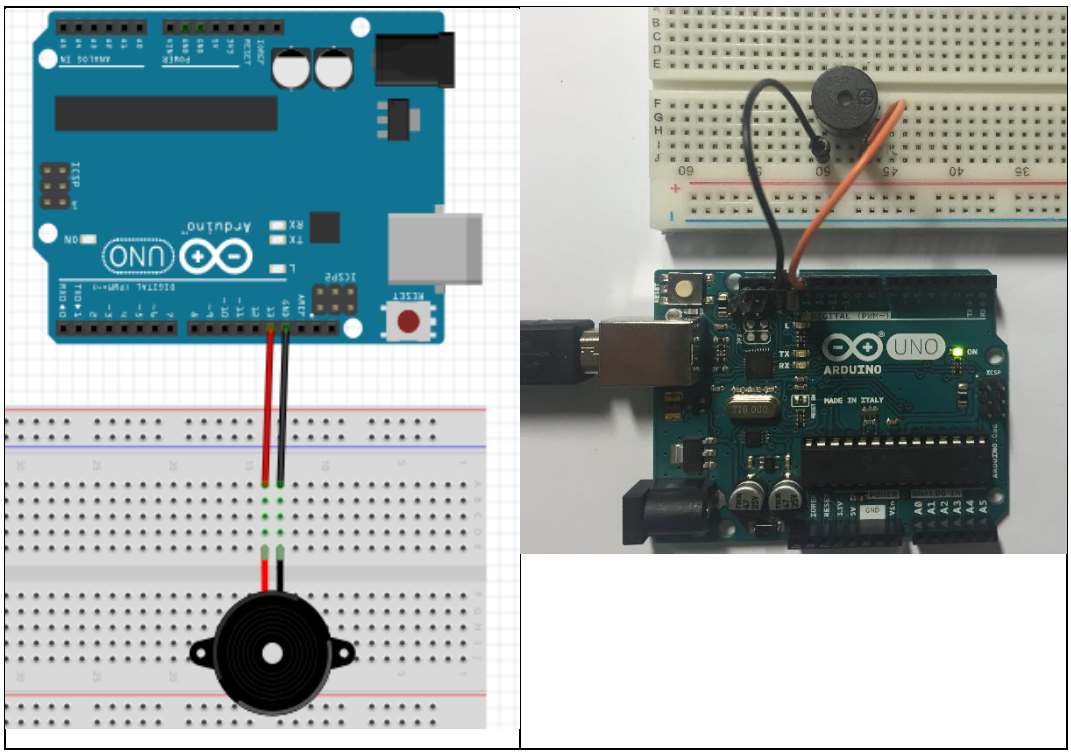
피에조(Piezo) 스피커는 2 개의 단자로 구성되면, 한쪽 단자는 양극이고 다른 쪽 단자는 음극입니다. 보통 단자의 길이가 긴 것이 양극 짧은 것이 음극이며, 스피커 위쪽에 [十] 기호가 표시되어 있습니다.

피에조 스피커는 압력을 가하면 전기를 발생시키는 압전효과(Piezoelectric Effect)를 역으로 이용하여 소리를 발생시킵니다. 압전소자에 전기신호를 주어 수축과 팽창을 반복적으로 수행하도록 함으로써 진동에 의해서 소리를 발생시키는 장치입니다.



【1】기본 소리 내기

13 번 핀으로 전류를 보내어 피에조 스피커에서 소리가 나고 1 초 후에 소리를 꺼봅시다.



알고리즘은 간단합니다. 13 번 핀으로 전류를 ON하고 1 초 후에 다시 OFF를 보내면 됩니다.

【Sketch】

【소스 프로그램】

```
//File Name: ex018_Piezo_02.ino
void setup() {
 pinMode(13,OUTPUT);
  digitalWrite(13, HIGH);   // turn the LED on
  delay(1000);                    // wait for 1 second
  digitalWrite(13, LOW);   // turn the LED off }
}

void loop() {
```

```
}
```

【mblock】

13 번 핀으로 전류를 보내어 피에조 스피커에서 1 초 동안 소리가 나고 소리를 끄고, 다시 3 초 후에 1 초 동안 소리가 나는 동작이 반복되게 프로그래밍하여 봅시다.

【Sketch】

【소스 프로그램】

```
//File Name: ex020_Piezo_04.ino
void setup() {
  pinMode(13,OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);      // turn the LED on
  delay(1000);                 // wait for 1 second
  digitalWrite(13, LOW);       // turn the LED off }
  delay(3000);
}
```

【mblock】

【코드 블록】

1 초동안 부저가 울리고 멈추는 것을 3 회하고, 0.5 초 동안 부저가 울리고 멈추는 것을 5 회하는 프로그램을 작성하여 봅시다.

【Sketch】

【소스 프로그램】

```
//File Name: ex021_Piezo_05.ino
int longNum=3;
int longTime=1;
int shortNum=5;
double shortTime=0.5;
int i;

void setup() {
  pinMode(13,OUTPUT);
  for(i=1;i<=longNum;i++)
  {
    digitalWrite(13,HIGH);
    delay(longTime*1000);
    digitalWrite(13,LOW);
    delay(longTime*1000);
  }
```

```
  for(i=1;i<=shortNum;i++)
  {
    digitalWrite(13,HIGH);
    delay(shortTime*1000);
    digitalWrite(13,LOW);
    delay(shortTime*1000);
  }
}
void loop() {
}
```

【mblock】

【코드 블록】

변수명을 한글로 지정하면 out of memory 에러가 나올 수 있으므로 변수명은 영어로 지정

【2】 멜로디 연주

음정과 해당 주파수

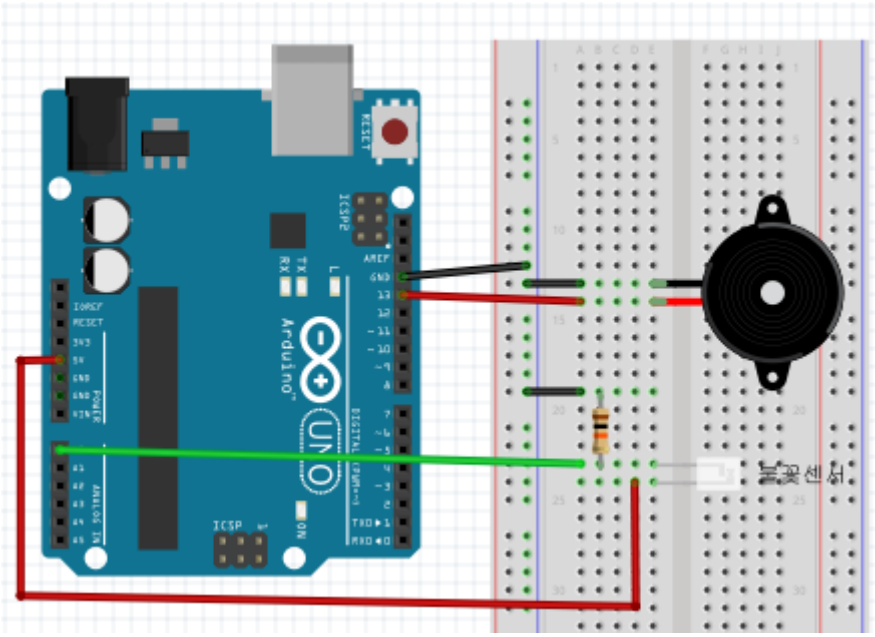
【소스 프로그램】

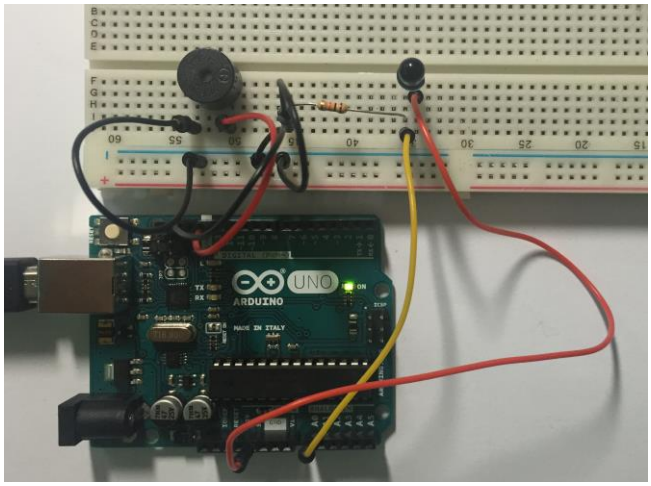
```
//int melody[]={262, 294, 330, 349, 392, 440, 440, 494, 523};
int melody[]={523, 587, 659, 740, 784, 831, 880, 988,1047};
void setup() {
  for (int i=0;i<8;i++){
```

```
    tone(13, melody[i], 250);
    delay(400);
    noTone(13);

  }


}

void loop() {
  // put your main code here, to run repeatedly:

}
```

7. 불꽃감지 센서

불꽃감지 센서는 포토 트랜지스터라고 불린다. 기본적으로 빛을 받으면 전기에너지로 변환시켜 증폭하는 작용을하며 빛을 많이 받을 수록 생성되는 전기 에너지가 커진다. 불꽃도 결국은 빛 이므로 이를 감지하여 화재여부를 감지하는 센서로 사용이 된다. 긴 쪽이 이미터 짧은 쪽이 콜렉터이다. 핀의 길이로 알 수 없는 경우에는 머리(?) 부분에서 넓은 쪽이 컬렉터 얇은 쪽이 이미터입니다. 불꽃이 감지되어 수광부로 받아지게 될 빛의 양에 따라 변화하는 전압 값을 이미터 부분에서 나오게 되므로 콜렉터는 5V단자에 이미터에는 아날로그 입력핀으로 연결합니다. 풀다운을 위해 아날로그 입력핀과 이미터는 10 ㏀의 저항을 사용하여 GND단자에 연결한다.



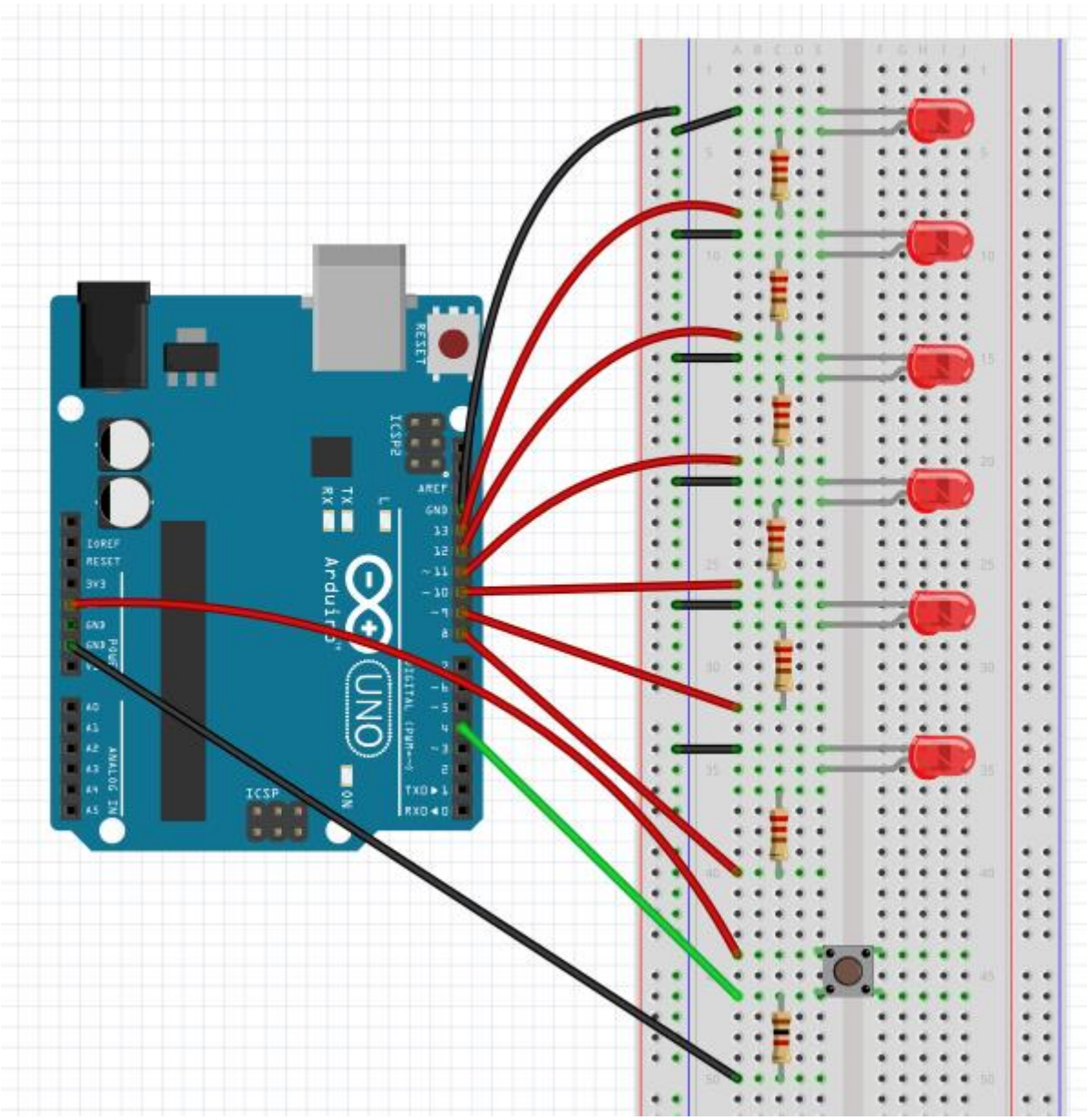


【소스 프로그램】

```
int val = 0;    //센서출력값 저장 변수

void setup() {
  pinMode(13, OUTPUT);   //피에조부저 출력설정
  pinMode(A0,INPUT);    //포토트랜지스터 입력설정
  Serial.begin(9600);     //시리얼모니터 설정
}

void loop() {
  val = analogRead(A0);  //포토트랜지스터에서 값을 읽어옴
  Serial.println(val);     //포토트랜지스터 입력값 시리얼모니터로 출력
  if(val >= 800)           //포토트랜지스터 입력값이 1000 이상이면
BEEP 실행, 환경에 따라 적절한 값으로 조절필요
  {
  digitalWrite(13,HIGH);  //피에조 부저 BEEP
  }else{
  digitalWrite(13,LOW);   //피에조 부저 OFF
  }
  delay(500);
}
```

8. LED 주사위 만들기

버튼을 클릭할 때마다 주사위 처럼 LED가 켜지는 프로그램



【1】난수발생기

【형식】

```
random(max)
random(min, max)
```

Parameters

min - lower bound of the random value, inclusive *(optional)*

max - upper bound of the random value, exclusive

Returns

a random number between min and max-1 (*long*)

1 에서 100 까지의 자연수 난수 발생

```
random(100)+1;
```

5 에서 70 까지 자연수 난수 발생

```
random(5, 70)+1;
```

randomSeed(seed)

Description

randomSeed() initializes the pseudo-random number generator, causing it to start at an arbitrary point in its random sequence.

예) 1 에서 9 까지의 난수를 발생

```
randomSeed(A0);
random(9)+1;
```

【소스 프로그램】

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int randomNumber;
  randomSeed(analogRead(A0));
  randomNumber=random(9)+1;
  Serial.println(randomNumber);
  delay(1000);
}
```

주사위 시뮬레이션 프로그램

【소스 프로그램】

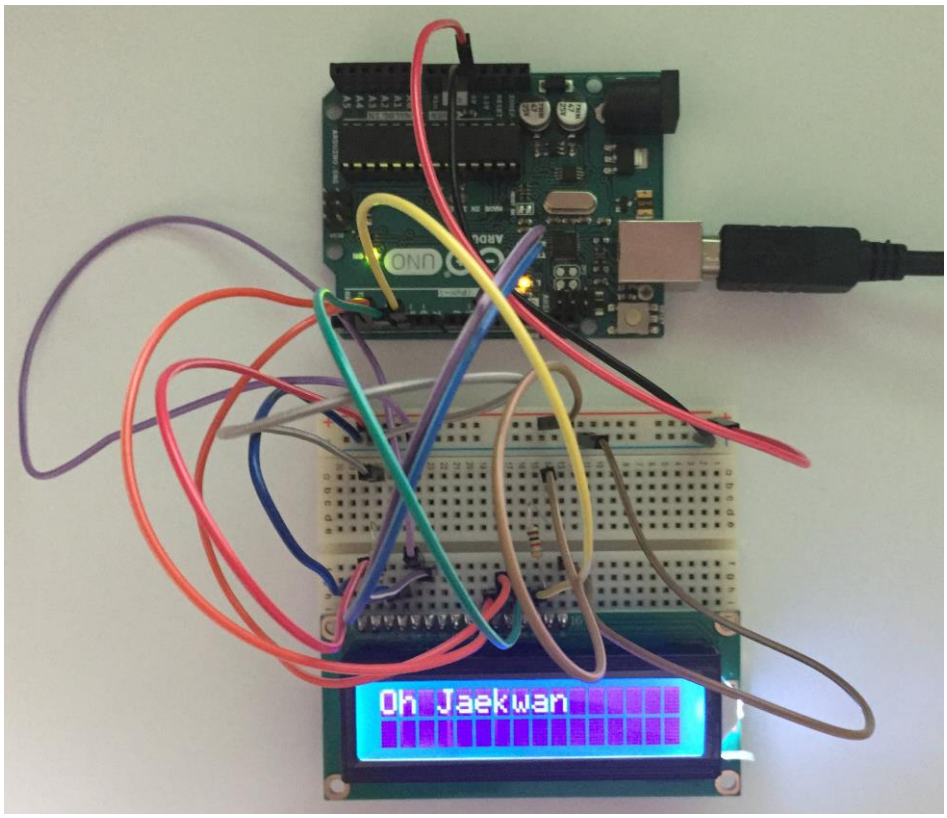
```
#define LED13 13
#define LED12 12
#define LED11 11
#define LED10 10
#define LED9 9
#define LED8 8

int delayTime=1000;
int dice;
void led_01();
void led_02();
void led_03();
void led_04();
void led_05();
void led_06();

void setup() {
  pinMode(LED13, OUTPUT);
  pinMode(LED12, OUTPUT);
  pinMode(LED11, OUTPUT);
  pinMode(LED10, OUTPUT);
  pinMode(LED9, OUTPUT);
  pinMode(LED8, OUTPUT);
 pinMode(4,INPUT);

 randomSeed(analogRead(A2));

 Serial.begin(9600);
}

void loop() {
  if (digitalRead(4)==1) {
    dice=random(6)+1;
```

```
    Serial.print("Dice Number =");
    Serial.println(dice);

    switch(dice){
      case 1: led_01(); break;
      case 2 : led_02(); break;
      case 3 : led_03(); break;
      case 4 : led_04(); break;
      case 5 : led_05(); break;
      case 6 : led_06(); break;
      default: led_off(); break;
    }
    delay(delayTime);
    led_off();
    delay(delayTime);
  }// End of if (digitalRead(4)==1)
} // End of void loop()

void led_01(){
  digitalWrite(LED13, HIGH);
}

void led_02(){
  digitalWrite(LED13, HIGH);
  digitalWrite(LED12, HIGH);
}

void led_03(){
  digitalWrite(LED13, HIGH);
  digitalWrite(LED12, HIGH);
  digitalWrite(LED11, HIGH);
}

void led_04(){
  digitalWrite(LED13, HIGH);
  digitalWrite(LED12, HIGH);
  digitalWrite(LED11, HIGH);
```

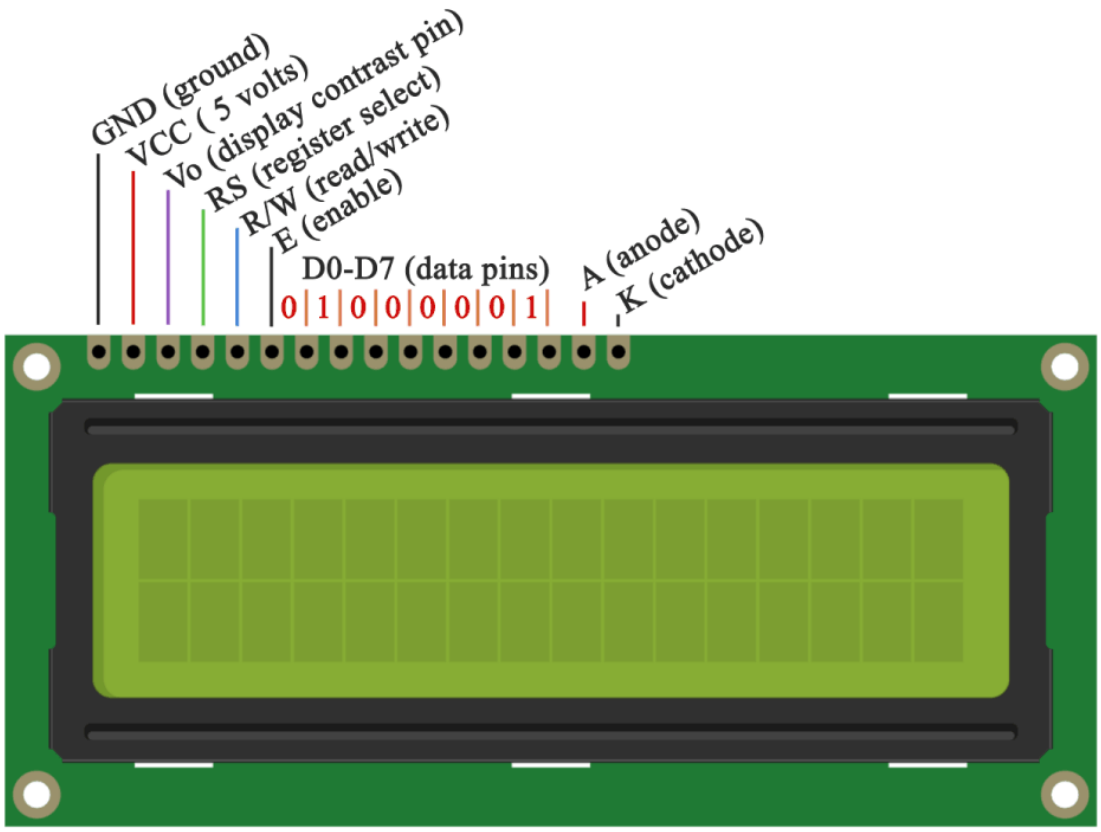
```
  digitalWrite(LED10, HIGH);
}

void led_05(){
  digitalWrite(LED13, HIGH);
  digitalWrite(LED12, HIGH);
  digitalWrite(LED11, HIGH);
  digitalWrite(LED10, HIGH);
  digitalWrite(LED9, HIGH);
}

void led_06(){
  digitalWrite(LED13, HIGH);
  digitalWrite(LED12, HIGH);
  digitalWrite(LED11, HIGH);
  digitalWrite(LED10, HIGH);
  digitalWrite(LED9, HIGH);
  digitalWrite(LED8, HIGH);
}

void led_off(){
  digitalWrite(LED13, LOW);
  digitalWrite(LED12, LOW);
  digitalWrite(LED11, LOW);
  digitalWrite(LED10, LOW);
  digitalWrite(LED9, LOW);
  digitalWrite(LED8, LOW);
}
```

9. LCD

【1】완성작



【2】LCD 구조


GND (ground)
VCC (5 volts)
Vo (display contrast pin)
RS (register select)
R/W (read/write)
E (enable)
D0-D7 (data pins)
0 1 0 0 0 0 0 1
A (anode)
K (cathode)

핀	이름	기능	비고
1	Vss	GND	0V
2	Vcc	5V	5V
3	Vo	백라이트 밝기 조절, 글자의 진하기 설정	
4	RS	Register Select	명령: 0, 데이터 : 1
5	RW	Read / Write	Read:1, Write : 0
6	E	Enable	
7	Data Bit 0(DB0)	데이터 입출력	
8	Data Bit 1(DB1)	데이터 입출력	
9	Data Bit 2(DB2)	데이터 입출력	
10	Data Bit 3(DB3)	데이터 입출력	
11	Data Bit 4(DB4)	데이터 입출력	
12	Data Bit 5(DB5)	데이터 입출력	
13	Data Bit 6(DB6)	데이터 입출력	
14	Data Bit 7(DB7)	데이터 입출력	
15	A(Anode)	LED 백라이트 + 전원	5V
16	K(Cathod)	LED 백라이트 - 전원	0V

【3】 설정

① 전원 설정

pin 2 Vcc로 전기가 흘러 들어가고 pin 1 Vss로 전기가 흘러 나온다. 따라서 Vcc는 5v에 연결하고 Vss는 GND에 연결한다.

② 글자 설정

pin 3 Vo는 글자 색을 진하게 할지 연하게 할지 결정한다. 저항을 연결하여 밝기를 조절하므로

글자 색을 진하게 하기 위해서는 적은 저항을 연결하고 연하게 하기 위해서는 큰 저항을 연결한다.

pin 4 RS는 LCD 내에 있는 저장 공간의 어느 곳에 저장할 것인지를 선택한다.

pin 5 RW는 LCD에서 읽어 올 것인지 쓸 것인지를 결정한다. 0 으로 설정하면 쓴다는 의미하고 1 로 설정하면 읽어 온다는 것을 의미한다.

활성화 여부

pin 6 E는 LCD에 값을 쓸 수 있도록 할 것인지 말 것인지를 결정한다.

데이터의 입출력

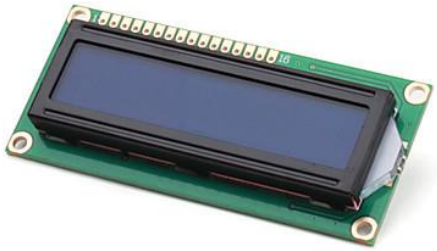
pin 7 ~ pin 14 는 데이터를 실제로 입출력할 때 사용하는 핀이다.

백라이트 전원

pin 15 는 백라이트 + 전원이므로 5V에 연결하고 pin 16 은 백라이트 – 전원이므로 GND에 연결한다.

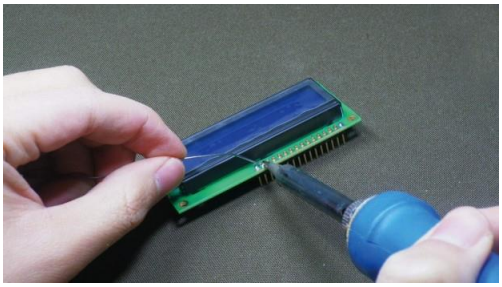
【4】LCD 설치

LCD를 구입하면 다음 그림의 형태로 되어 있다.



이 부품을 어떻게 연결할 것 인가 막막하다. 다음 그림과 같은 접속하는 핀이 꽂혀 있고 납땜이 되어 있는 부품으로 구입하면

간단하겠지만 그렇지 않은 경우에는 납땜을 하여야 한다.



왼쪽 그림과 같은 핀 헤더를 핀의 개수 16 를 잘라서 오른 쪽 그림과 같이 LCD에 길이가 짧은 핀 쪽을 아래에서부터 끼우고 납땜을 조심스럽게 하여야 한다.

LCD 좌표

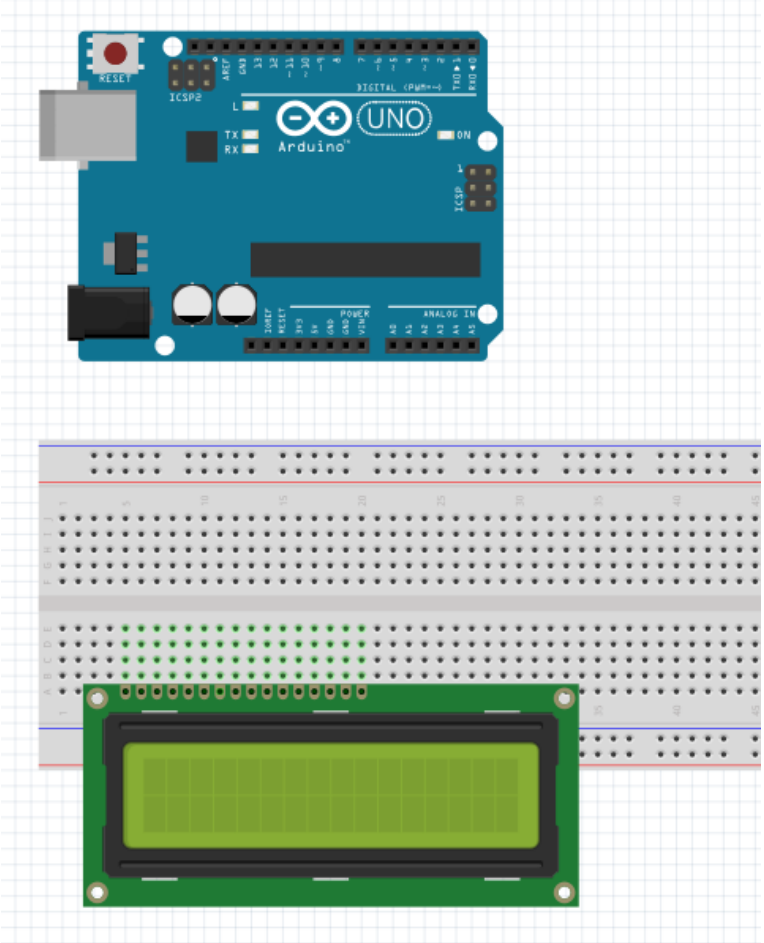
좌표를 지정할 때 (x,y) 순이고 LCD도 마찬가지이다. 좌표평면으로 말하면 제 4 사분면을 양수로 표기하는 것이다.

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)	(8,0)	(9,0)	(10,0)	(11,0)	(12,0)	(13,0)	(14,0)	(15,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)	(8,1)	(9,1)	(10,1)	(11,1)	(12,1)	(13,1)	(14,1)	(15,1)

이제 LCD에 영어로 “Oh Jaekwan” 라고 적어보자

1. LCD 장착

LCD를 브레드보드f~j 또는 a~e에 끼운다.



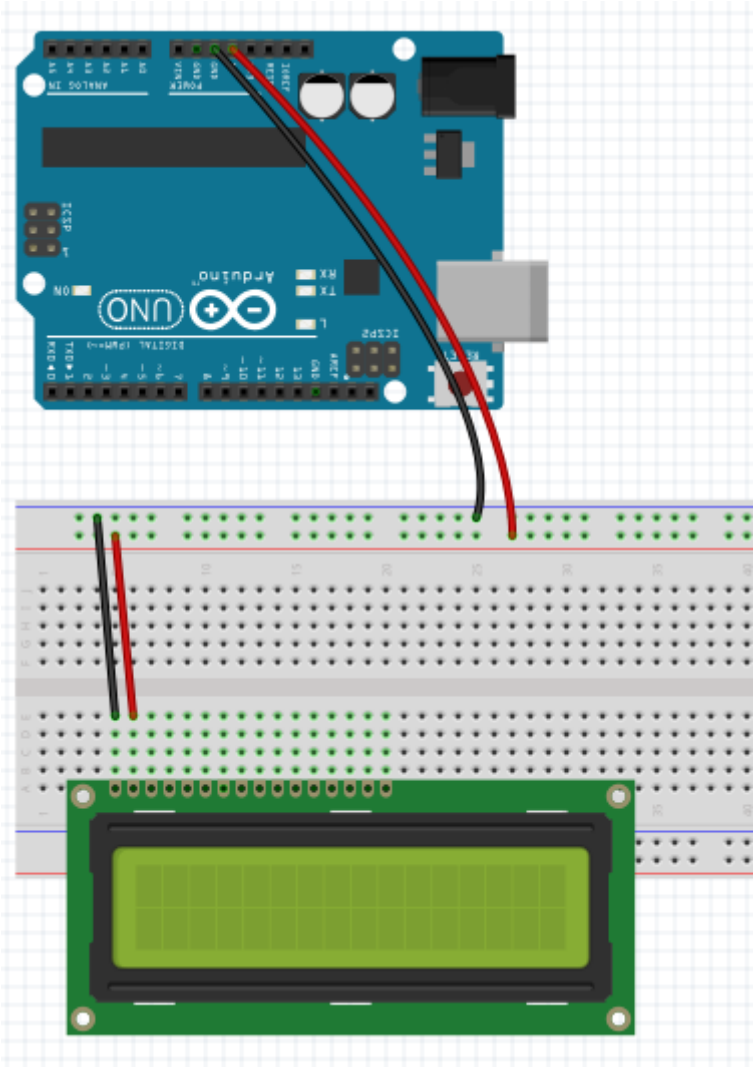
2. LCD 전원 연결

LCD에 전원을 공급하기 위해서 pin1 Vss를 GND에 pin2 를 5V에 연결한다. 백라이트 전원을 사용하지 않는다면 아두이노 보드에서 직접 연결하여도 되지만 백라이트 전원을 사용하여야 하므로 5V, GND가 2 개가 필요하다. 따라서

브레드보드 긴 파란색 +와 긴 빨간 색 -를 사용하여 5V를 긴 파란색 +에 연결하고 GND를 빨간 색 -에 연결한 다음 긴 파란색 +와 빨간 색 -에서 LCD에 연결되어 있는 브레드 보드에 연결한다.

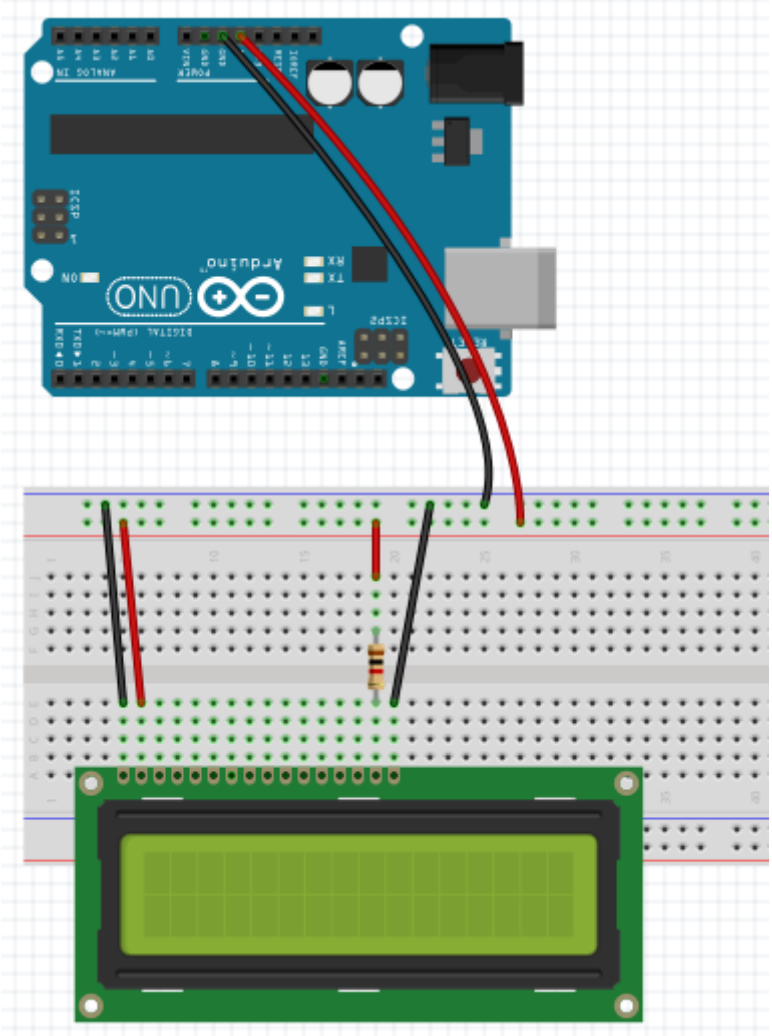
【주의!】

브레드 보드에 있는 긴 파란색과 긴 빨간색이 위치에 주의하여야 한다. 브레드 보드는 한 쪽은 바깥 쪽이 빨간 색이고 다른 쪽은 바깥 쪽이 빨간색이다.



3. 백라이트 전원 연결

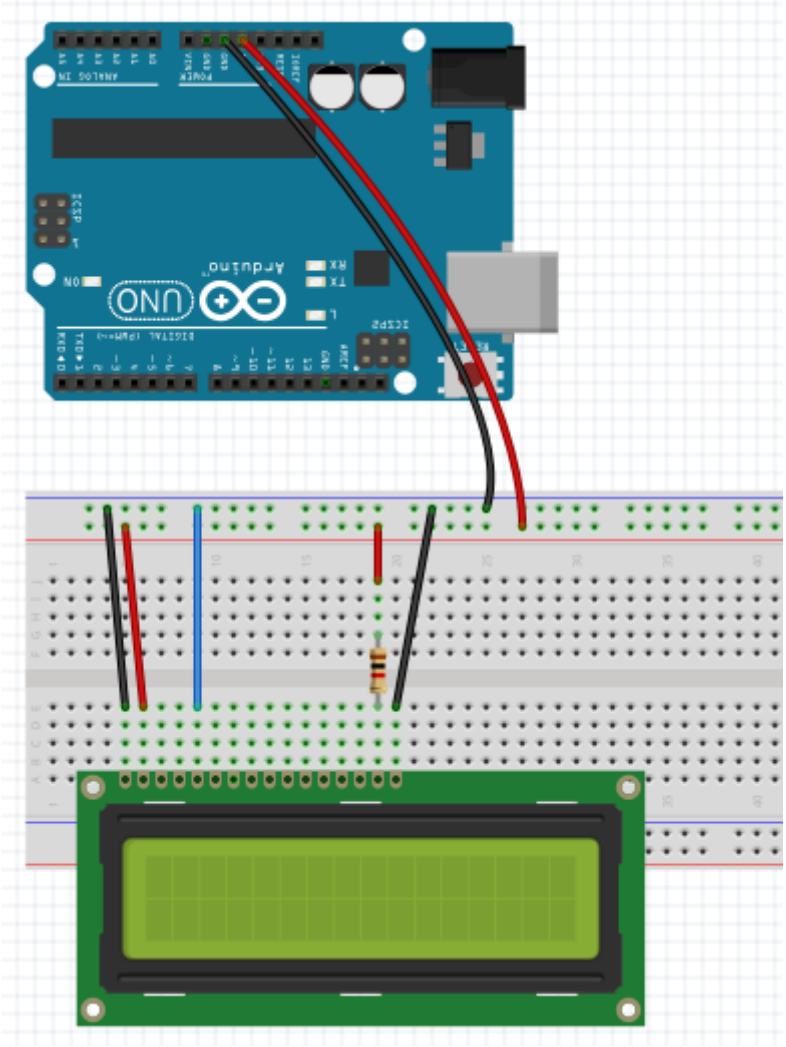
백라이트 전원을 연결하기 위해서 pin 15 A를 긴 빨간색 +에 pin 16 K를 긴 파란색 – 에 연결한다. 긴 빨간색 +에 연결할 때 중간에 1KΩ 저항을 둔다.



4. Read/Write 설정

LCD에 데이터를 Write하여야 하므로 Write 값을 0 으로 설정하여야 한다. 0 을 설정하기 위해서는 GND에 연결하여야 하므로

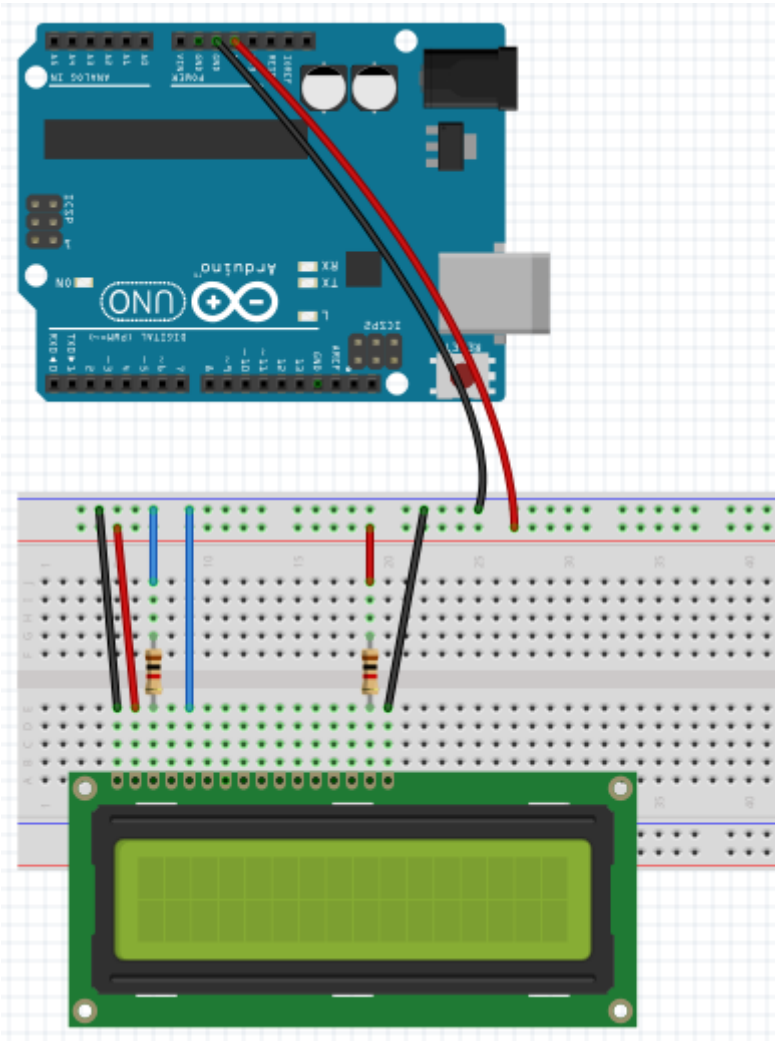
pin 5 E를 긴 파란색 -에 연결한다.



5. 글자의 농도(백라이트 밝기) 설정

글자를 어느 정도 진하게 할 것인지를 설정한다. 글자의 농도를 설정하는 것은 pin 3 Vo이므로 pin 3 Vo의 전압을 정도를 설정하면 된다. 전압이 낮을 수록 진해지고 전압이 높을 수록 연해진다. 가장 진하게 설정하기 위해서는 전압을 가장 낮게 설정하여야 하므로 GND에 연결하면 된다. 따라서

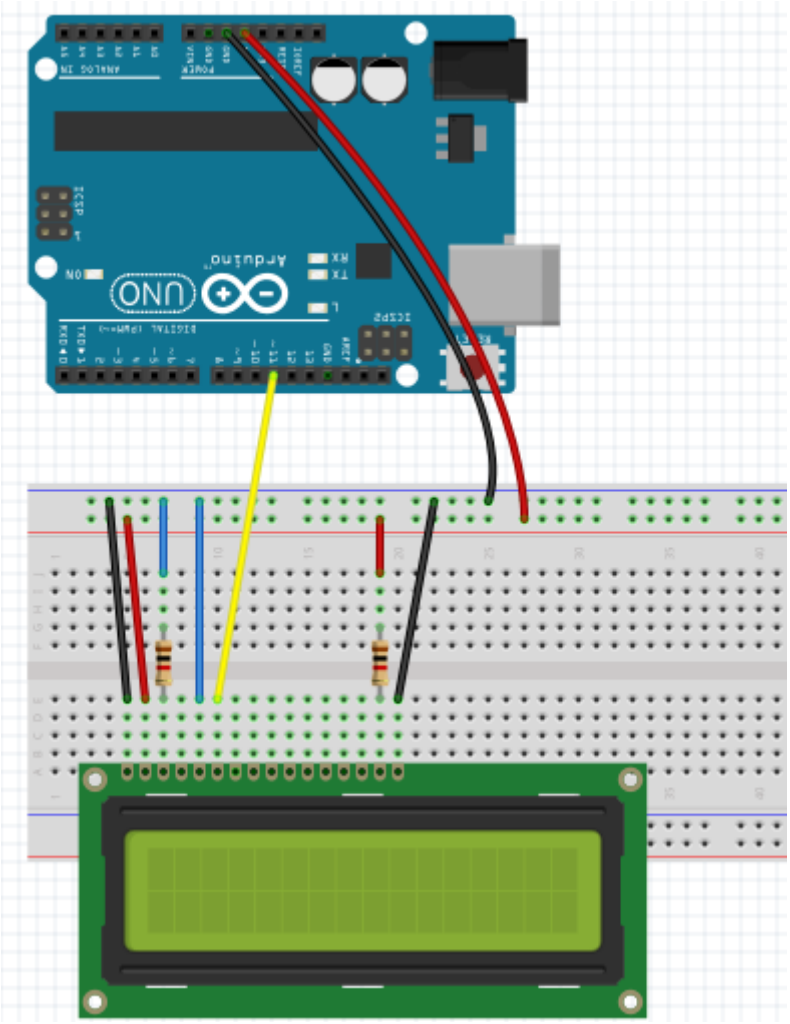
pin 3 Vo를 중간에 1KΩ 저항을 두고 긴 파란색 -에 연결한다.



6. Enable 여부 설정

pin 6 E는 LCD사용 여부를 결정한다. 왜 이런 기능이 필요한가 의문을 가질 수도 있지만 시스템 리소스를 최대로 아끼지 위해서 LCD가 연결되어 있더라도 필요에 따라서 사용하고 안 하고를 설정하여 사용하는 것이 바람직하므로 이런 기능이 있어야 한다. 우리는 여기에서 LCD를 당연히 사용하므로 Enable을 설정하여야 한다. Enable를 부여하기 위해서는 pin 6 E에 펄스를 주어야 한다. 따라서 제어가능한 아두이노 보드의 핀에 연결하여야 한다. LCD를 초기화 할 때에 이 핀을 지정하여야 한다. 여기에서는 아두이노 보드 11 번을 사용하고 연결하고 프로그램에서는 아두이노 11 번 핀을 사용하여 enable한다는 것을 지정하여야 한다.

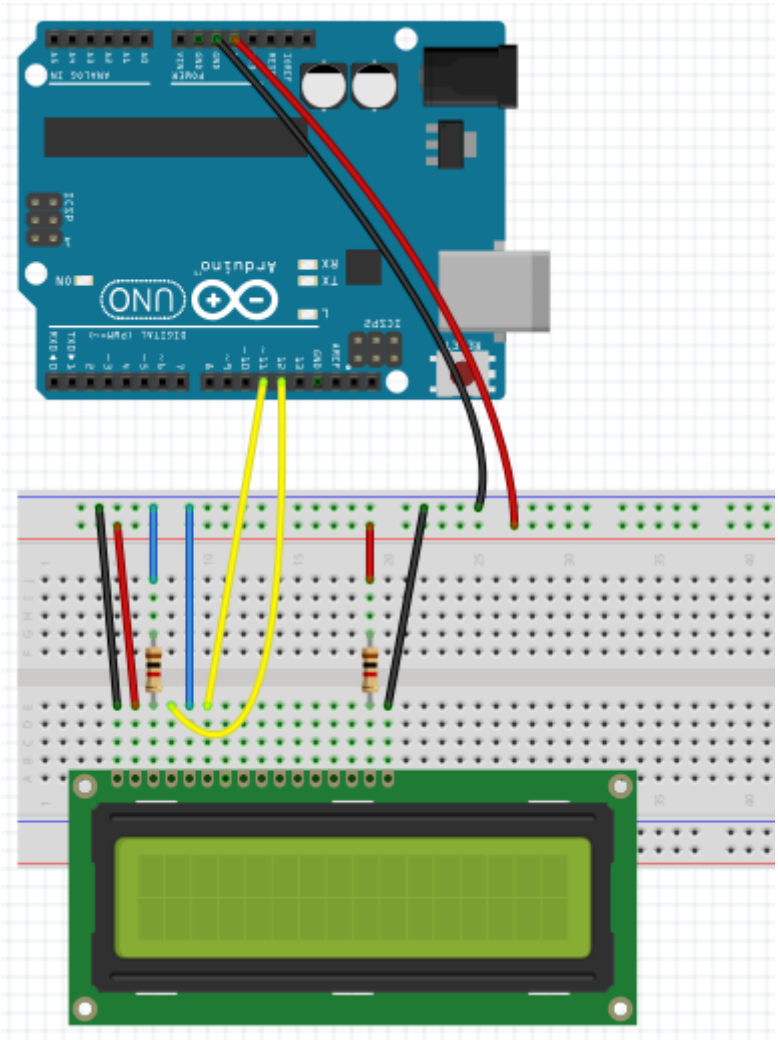
pin 6 E를 아두이노 핀 11 에 연결한다.


UNO
Arduino

7. 레지스터 선택

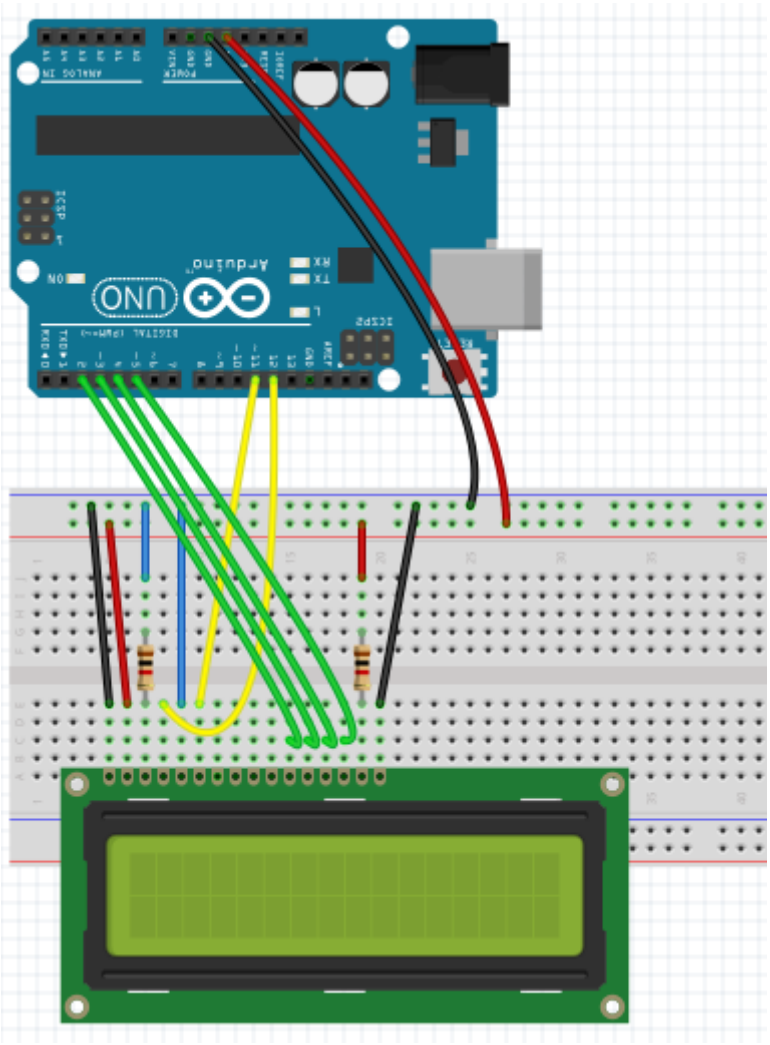
pin 4 RS의 값이 0 이면 명령이고 1 이면 데이터이다. 이 것은 아두이노 핀에서 직접 보내야 하므로 아두이노 핀 중 하나를 지정한다. 여기에서는 12 번 핀을 사용한다.

pin 4 RS를 아두이노 핀 12 에 연결한다.



8. 데이터 전송 선 연결

보통 데이터를 전송할 때 DB4, DB5, DB6, DB7 을 사용하므로 pin 11 DB4 를 아두이노 2 번 핀에, pin 12 DB5 를 아두이노 3 번 핀에, pin 13 DB6 을 아두이노 4 번 핀에, pin 14 DB7 를 아두이노 5 번 핀에 각각 연결한다.


UNO
Arduino

LiquidCrystal lcd(RS,Enable,DB4,DB5,DB6,DB7);
ex) LiquidCrystal lcd(12,11,2,3,4,5);

lcd.begin(column , row); 사용할 컬럼과 행 수를 지정
ex) lcd.begin(16,2);

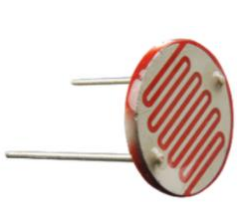
lcd.clear(); LCD를 지운다.

lcd.print("String xx"); LCD에 문자열을 출력한다.
ex) lcd.print("Oh Jaekwan");

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(12,11,2,3,4,5);
void setup() {
  lcd.begin(16,2);
}

void loop() {
  // put your main code here, to run repeatedly:
  lcd.clear();
  lcd.print("Oh Jaekwan");
  delay(1000);
}
```

```
#include< LiquidCrystal.h>
LiquidCrystal lcd(13,12,11,10,9,8);      //INITIALIZING LCD IN 4-BIT MODE
char c[15]={"USMAN ALI BUTT"};
String s;
void setup()
{
lcd.begin(16 , 2);
lcd.clear();
}
void loop()
{
int i,j,k=0,l=0;
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Moving Text!!!");
delay(1000);
for(i=14;i>=0;i--)
{
  k=0;
  for(j=i;j<=13;j++)
  {
  lcd.setCursor(j,1);
  lcd.print(c[k]);
  delay(90);
  k++;
  if(i==0 && j==13)
  goto xyz;              //IF string reaches the first matrix drop each character one by one
                         //so it feels that the string is terminating
  }
}
xyz:

if(i==0 && j==13)
      {
        while(k!=0)
           {
        lcd.setCursor(0,1);
        for(i=k;i>=2;i--)
        {
          l=15-i;
        lcd.print(c[l]);
        delay(90);
        }
        lcd.setCursor(0,0);
        lcd.clear();
        lcd.print("Moving Text!!!");
          k--;
        }
      }
}
```

10. 조도 센서



조도 센서(照渡 Sensor, Photo Resistor)란 주위의 밝기를 감지하는 센서입니다. 빛을 받으면 내부에 움직이는 전자가 발생하여 전도율이 변화는 광전효과를 가지는 소자를 사용합니다. 황화카드뮴(Cds)를 소자로 사용하기 때문에 Cds 센서라고도 부릅니다.

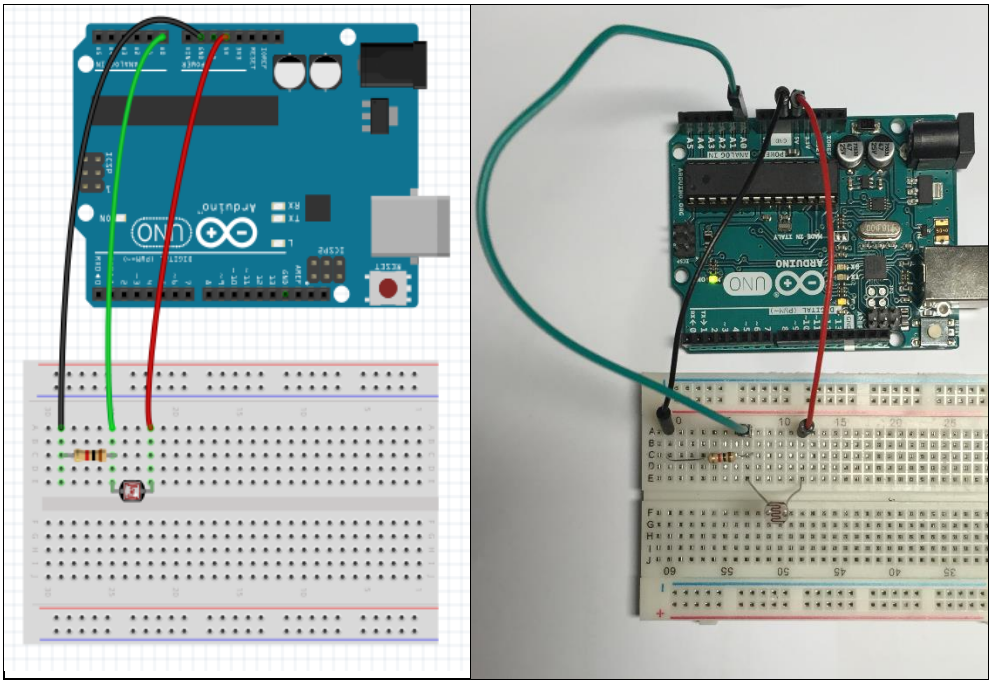
LED 처럼 3mm, 5mm, 12mm 등 다양합니다. 조도센서는 일종의 저항으므로 주위가 밝으면 저항이 감소하고 주위가 어두우면 저항이 증가한다. 우리가 밝은 데에 가면 눈이 부셔서 움추리는 것처럼 조도센서도 밝은 데에서는 움추려서 저항이 감소한다고 기억하면 된다. 저항이 감소한다는 것은 전류량이 증가한다는 것이고 저항이 증가한다는 것은 전류량이 감소한다는 것을 의미합니다.

따라서 조도센서는 밝은 데에서는 전류량이 증가하고 어두운 데에서는 전류량이 감소합니다. 즉 조도센서의 감지값이 크면 밝고 작으면 어둡다는 것을 알 수 있습니다. 감지에서 사용하는 명령어는 analogRead 이고 analogRead의 최솟값은 0 이고 최댓값은 1023 이므로 조도센서에 연결되어 있는 analogRead값이 0 에 가까우면 어둡고 1023 에 가까우면 밝다는 것을 알 수 있다.

조도센서 감지값이 0 에 가까우면 어둡고 1023 에 가까우면 밝습니다.

【1】조도 감지

조도센서로 주위의 빛의 양을 감지하여 표시하여 봅시다.

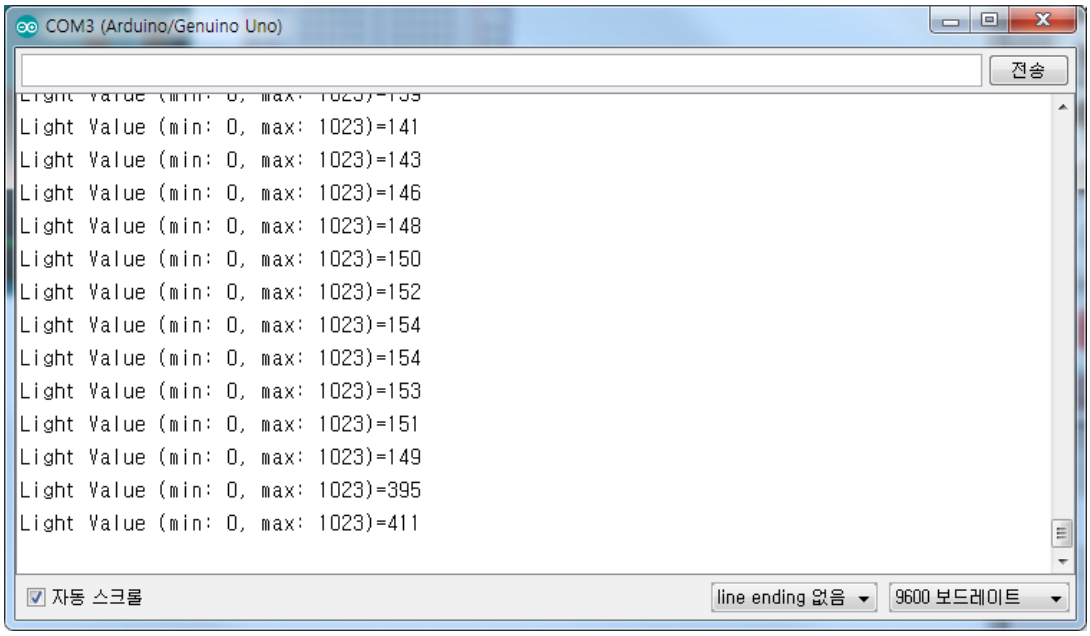


평상시 낮의 실내 조도가 10 미만으로 나오면 센서의 오동작이므로 센서를 교체하여야 합니다.

【Sketch】

【소스 프로그램】

```
//File Name: ex021_photoResistor01
void setup() {

}

void loop() {
    int photoValue;
    photoValue=analogRead(A0);
    Serial.begin(9600);
    Serial.print("Light Value (min: 0, max: 1023)=");
    Serial.println(photoValue);
    delay(1000);
 }
```

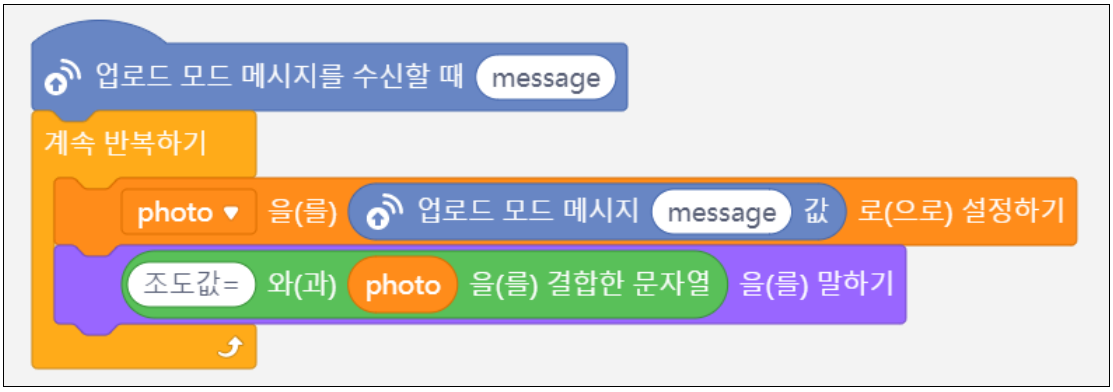


【mblock】

【코드 블록】

장치

스프라이트



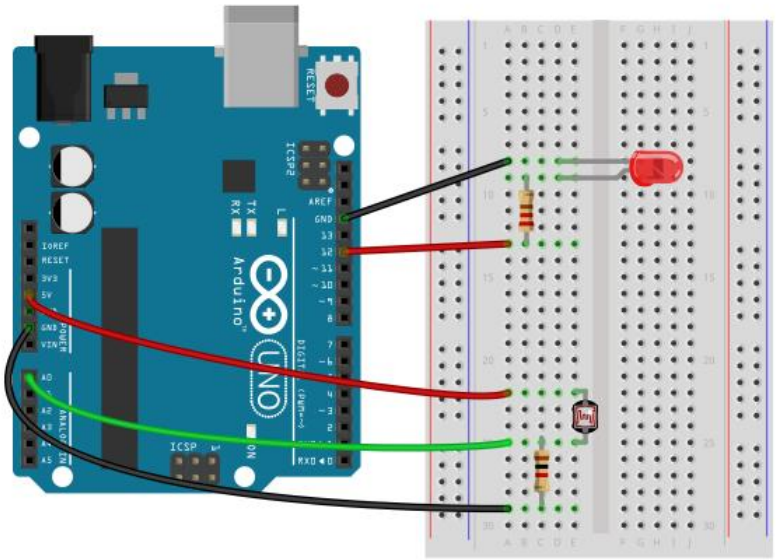
【무대】

조도값이 300 이상일 때	조도값이 300 미만일 때
	

【2】자동 조명

자동 조명은 어두울 때면 조명이 들어오고, 밝으면 조명이 꺼지는 장치입니다. 그 원리는 사실 간단합니다. 조도 센서로 주변 밝기를 감지하여 밝기가 어떤 값 미만이면 조명이 켜지고, 그 값 이상이면 조명이 꺼지게 하면 됩니다.

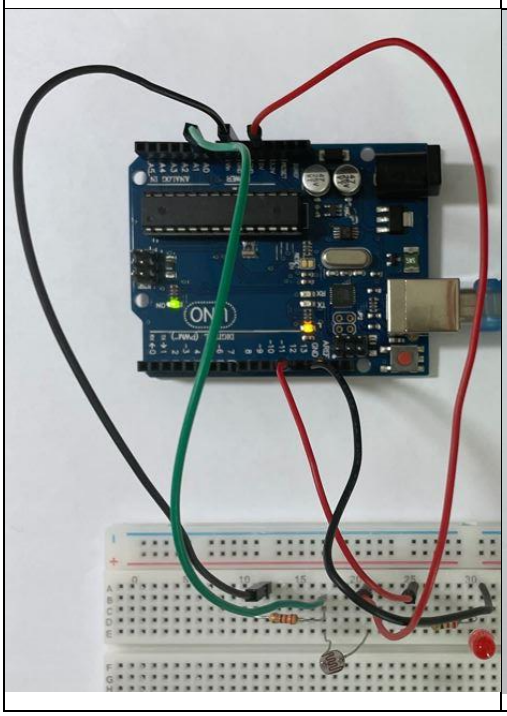
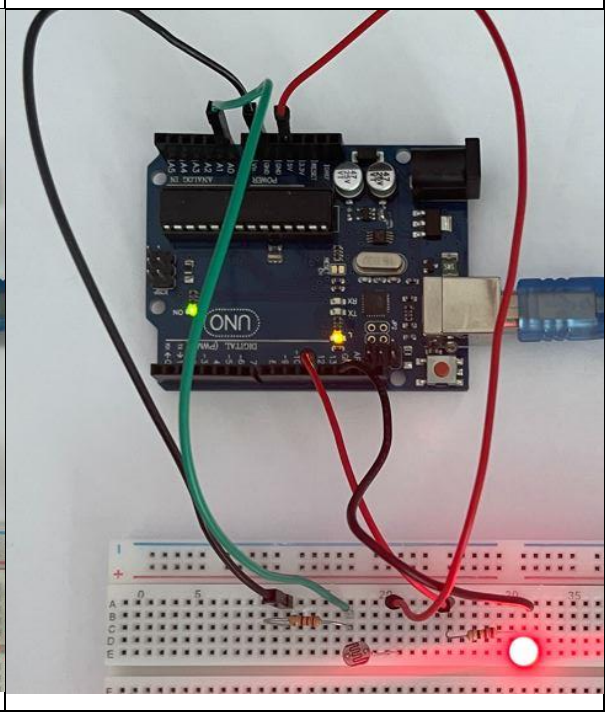
조명을 LED를 사용하여 조도 센서 값이 300 미만이면 LED 가 켜지고 300 이상이면 LED가 꺼지게 만들어 봅시다.



【mblock】

【코드 블록】

【실행 결과】

조도값이 300 이상일 때	조도값이 300 미만일 때
	

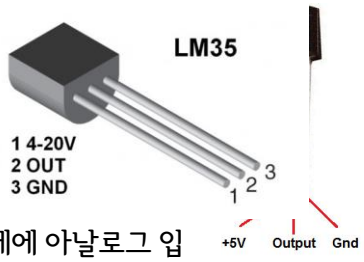
실행 동영상: https://www.youtube.com/watch?v=ts9tArNSmqQ&list=PL0-gQtm6qnGIIb5cssjOg0uYkqEmyevNx&index=3

11. 온도 센서

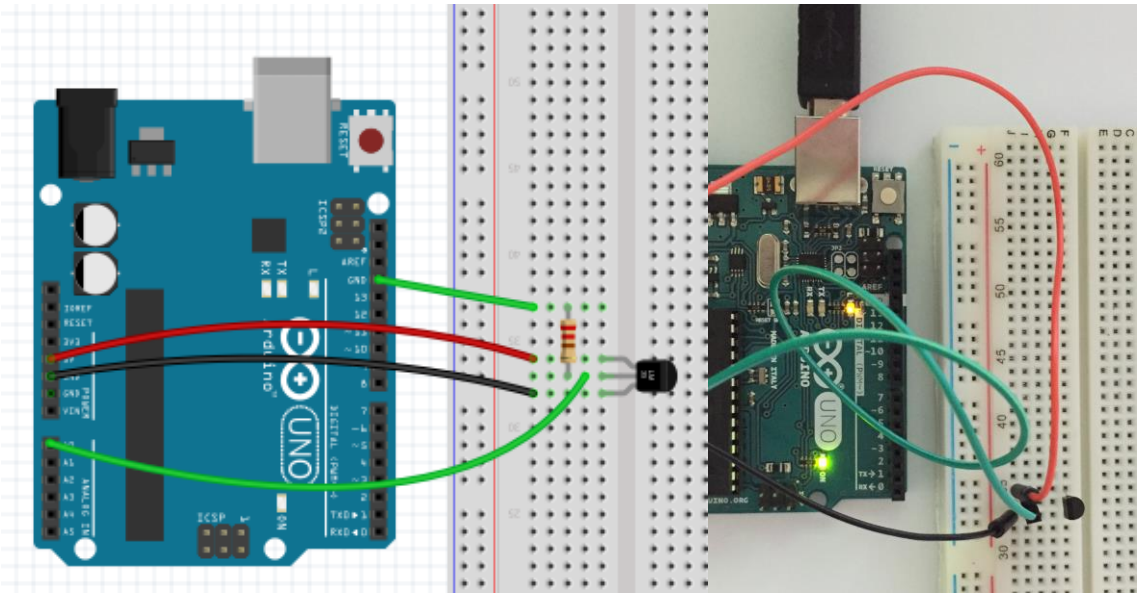
온도 센서로 감지한 값은 LCD와 시리얼에 표시하기

온도센서란 온도에 따라서 다른 전압을 출력하는 센서이다. 따라서 온도센서에서 출력되는 전압을 측정하여 온도로 변환시켜서 온도를 측정한다.

여기에서 사용하는 온도 센서는 LM35 이다. LM35 는 0 도에서 100 도까지 측정가능하고 동작 전원 범위가 4V~20V이고 출력 신호는 10mV 단위의 아날로그 이다. 누설 전류도 60mA 이하로 적어서 많이 활용되는 센서이지만 빠른 온도 변화에서는 측정 신뢰가 떨어지는 센서이다.



오른 쪽 그림처럼 만들어져 있으므로 아두이노에서 사용하기 위해서는 왼편에 5V를 연결하고 오른 쪽에 GND를 연결하고 가운데에 아날로그 입력 단자 A0~A5 중의 하나를 연결한다.



아날로그 입력 값을 온도로 변환하는 공식은 제품마다 다르며 아두이노 공식 싸이트에서 제시한 공

식은 다음과 같다.

```
LM35 = (5.0 * reading * 100.0) / 1024.0;
TMP36 = (((reading * 5.0) / 1024.0) - 0.5) * 100
```

LM35 센서는 0^0C~100^0C까지 측정 가능하고 출력 전압은 0 V ~ 1 V이다. 즉 0^0C에서는 0V를 출력하고 100^0C에서는 1V를 출력한다. 1^0C 당 0.01V를 출력하므로 LM35 센서가 0.25V를 출력한다면 25^0C이다. 따라서 LM35 센서에서 출력되는 전압을 측정하여 여기에 100 을 곱해서 온도로 변환하면 된다.

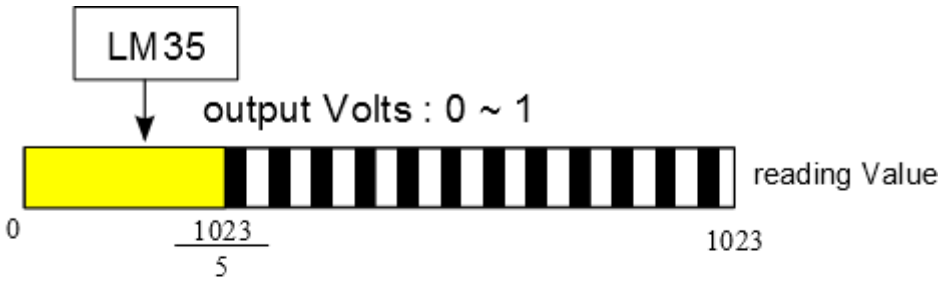
온도 = 전압 * 100

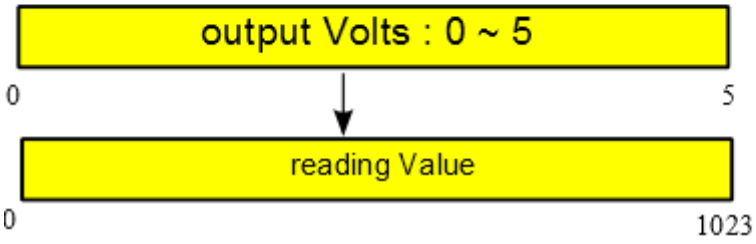
그러나 센서에서 출력되는 값은 전압이지만 analogRead 명령을 통해서 읽는 값은 0 ~ 1023 인 숫자이다. 따라서 출력되는 전압을 0~1023 숫자로 읽어서 이를 전압으로 변환한 다음 다시 온도로 변환하여야 한다.

analogRead 값 → 전압 → 온도

analogRead() 에 대한 Arduino Reference를 보면 analogRead() 는 0V일 때 0 을 5V일 때 1023 을 출력한다. analogRead 값이 0 이면 0V가 아날로그 입력핀으로 들어오고 analogRead 값이 1023 이면 5V가 아날로그 입력핀으로 들어온다는 것을 알 수 있다.

우리의 목표는 전압을 온도로 변환하는 것이다. 다음 그림에서 볼 수 있는 것처럼 analogRead는 5V 를 기준으로 값을 읽은 것이므로 LM35 가 최대 출력인 1V를 출력하더라도 analogRead은 1023/5 의 값을 읽게 된다. 그러므로 LM35 를 사용하는 경우에는 analogRead의 최댓값 1023 을 얻기 위해 5 를 곱하여야 한다.





analogRead 값에 5 를 곱하였으므로 1023 일 때 LM35 는 1v가 된다. 그러므로 analogRead 값에 5 를 곱한 값을 전압으로 바꾸기 위해서는 analogRead 값에 5 를 곱한 값을 1024 로 나누어 주면 된다. 왜냐하면 전압의 폭은 1 이고 analogRead 값의 폭은 1024 이기 때문이다.

전압 = 5 * analogRead 값 / 1024

이제 전압을 온도로 바꾸어 보자. LM35 에서는 온도 = 전압 * 100 이므로

온도 = 전압 *100 = 5 * analogRead 값 / 1024 * 100

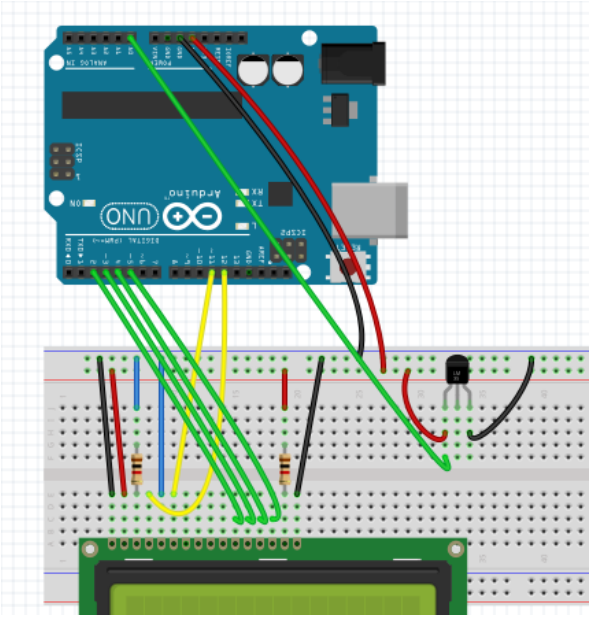
이다. 이 식을 간단히 정리하면

온도 = (5 * analogRead 값 *100) / 1024

가 된다.

```
#define LM35 A0

int reading = 0;
float temperature = 0;


void setup() {

}

void loop() {
  reading = analogRead(LM35);
  temperature = ( 5.0 *reading * 100.0) / 1024.0;

  Serial.begin(9600);
  Serial.print("LM35 = ");
  Serial.println(temperature);
  delay(1000);
```

12. LCD에 온도 표시



```
#include<LiquidCrystal.h>
#define LM35 A0

int val = 0;
float temp = 0;
LiquidCrystal lcd(12,11,2,3,4,5);
```

```
void setup() {
  lcd.begin(16,2);
}

void loop() {
  // put your main code here, to run repeatedly:
  val = analogRead(LM35);
  celsius = val * 0.48876;          //(5.0 * reading * 100.0) / 1024.0; 오차 있음.
  Serial.begin(9600);
  Serial.print("LM35 = ");
  Serial.print(celsius);
  Serial.println("C");
  delay(1000);
  lcd.clear();
  lcd.print(celsius);

  delay(1000);
}
```

```
#include<LiquidCrystal.h>
#define LM35 A0

int reading = 0;
float temperature = 0;
LiquidCrystal lcd(12,11,2,3,4,5);

void setup() {
  lcd.begin(16,2);
}

void loop() {
  reading = analogRead(LM35);
  temperature = 5 *reading * 100 / 1024;
  Serial.begin(9600);
```

```
  Serial.print("LM35 = ");
  Serial.println(temperature);
  delay(1000);
```

Arduino Reference

analogRead()

Description

Reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().

http://playground.arduino.cc/

The LM35 is a common TO-92 temperature sensor. It is often used with the equation

temp = (5.0 * analogRead(tempPin) * 100.0) / 1024;

However, this does not yield high resolution. This can easily be avoided, however. The LM35only produces voltages from 0 to +1V. The ADC uses 5V as the highest possible value. This is wasting 80% of the possible range. If you change aRef to 1.1V, you will get almost the highest resolution possible.

The original equation came from taking the reading, finding what percentage of the range (1024) it is, multiplying that by the range itself(aRef, or 5000 mV), and dividing by ten (10 mV per degree Celcius, according to the datasheet:http://www.ti.com/lit/ds/symlink/lm35.pdf

However, if you use 1.1V as aRef, the equation changes entirely. If you divide 1.1V over 1024, each step up in the analog reading is equal to approximately 0.001074V = 1.0742 mV. If 10mV is equal to 1 degree Celcius, 10 / 1.0742 = ~9.31. So, for every change of 9.31 in the analog reading, there is one degree of temperature change.

To change aRef to 1.1V, you use the command "analogReference(INTERNAL);"

Here's an example sketch using 1.1 as aRef:

float tempC;

```
int reading;
int tempPin = 0;

void setup()
{
analogReference(INTERNAL);
Serial.begin(9600);
}

void loop()
{
reading = analogRead(tempPin);
tempC = reading / 9.31;
Serial.println(tempC);
delay(1000);
}
```

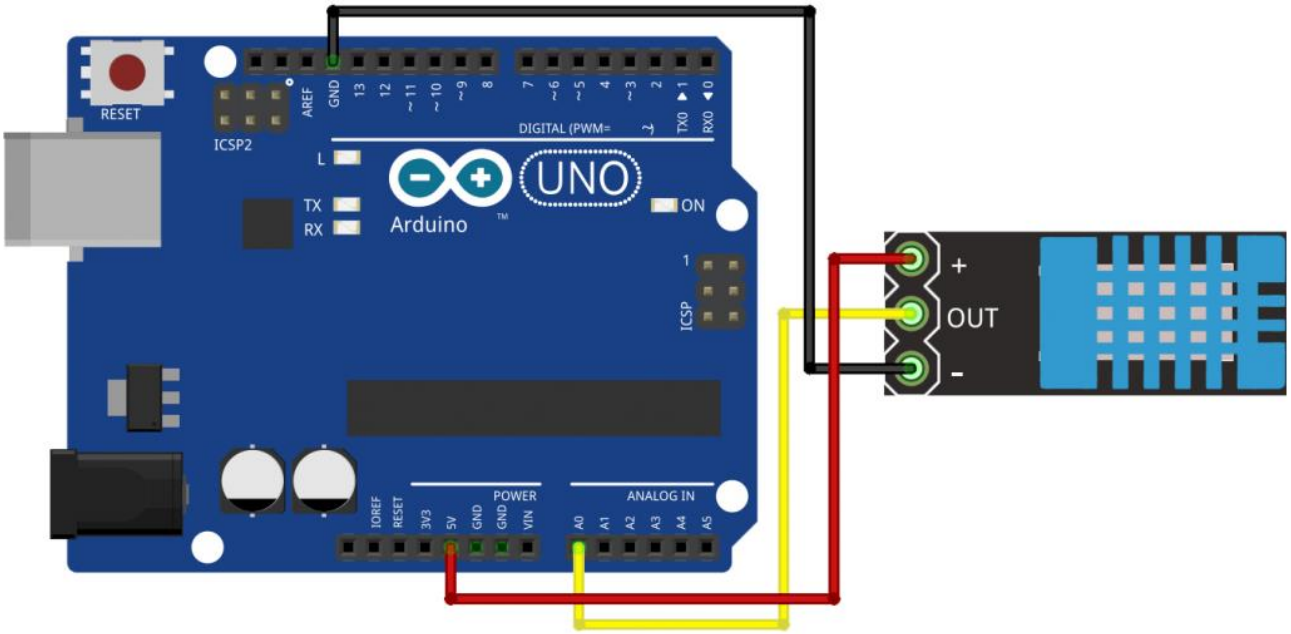
```
void loop()
{
int rawvoltage= analogRead(outputpin);
float millivolts= (rawvoltage/1024.0) * 5000;
float celsius= millivolts/10;
Serial.print(celsius);
Serial.print(" degrees Celsius, ");

Serial.print((celsius * 9)/5 + 32);
Serial.println(" degrees Fahrenheit");
delay(1000);
}
```

13. 온도 습도센서

온도습도 센서로 감지한 값은 시리얼에 표시하기

온도와 습도를 동시에 측정하는 온도습도 센서 중 가장 많이 사용하는 DHT11 을 이용하여 보자.

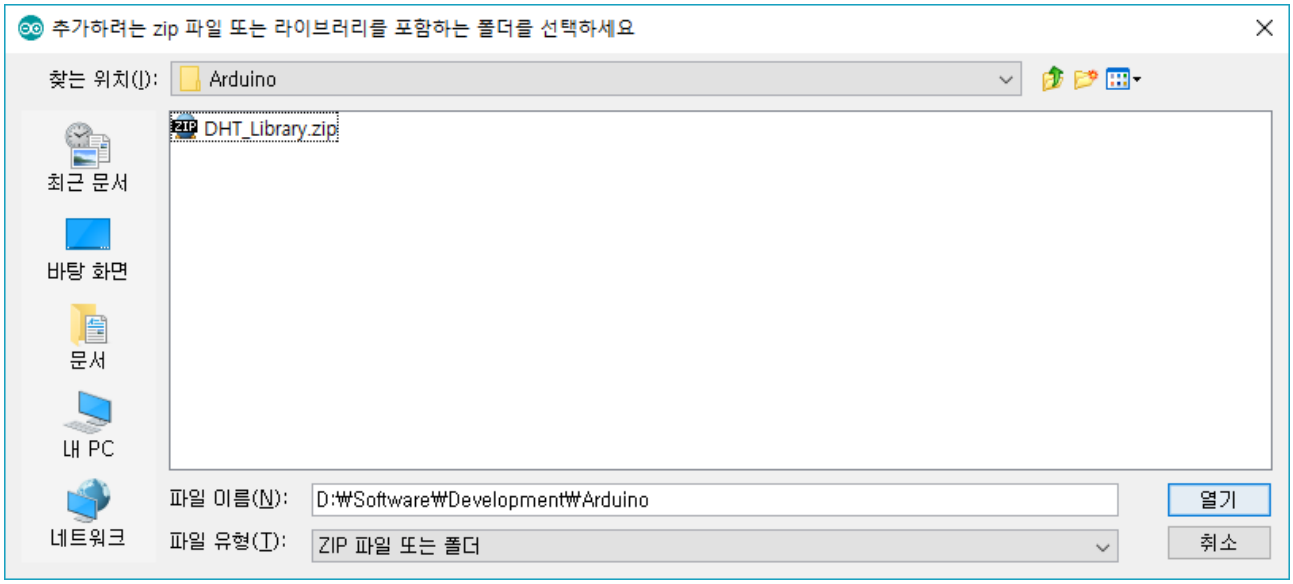
다음과 같이 연결하면 된다.



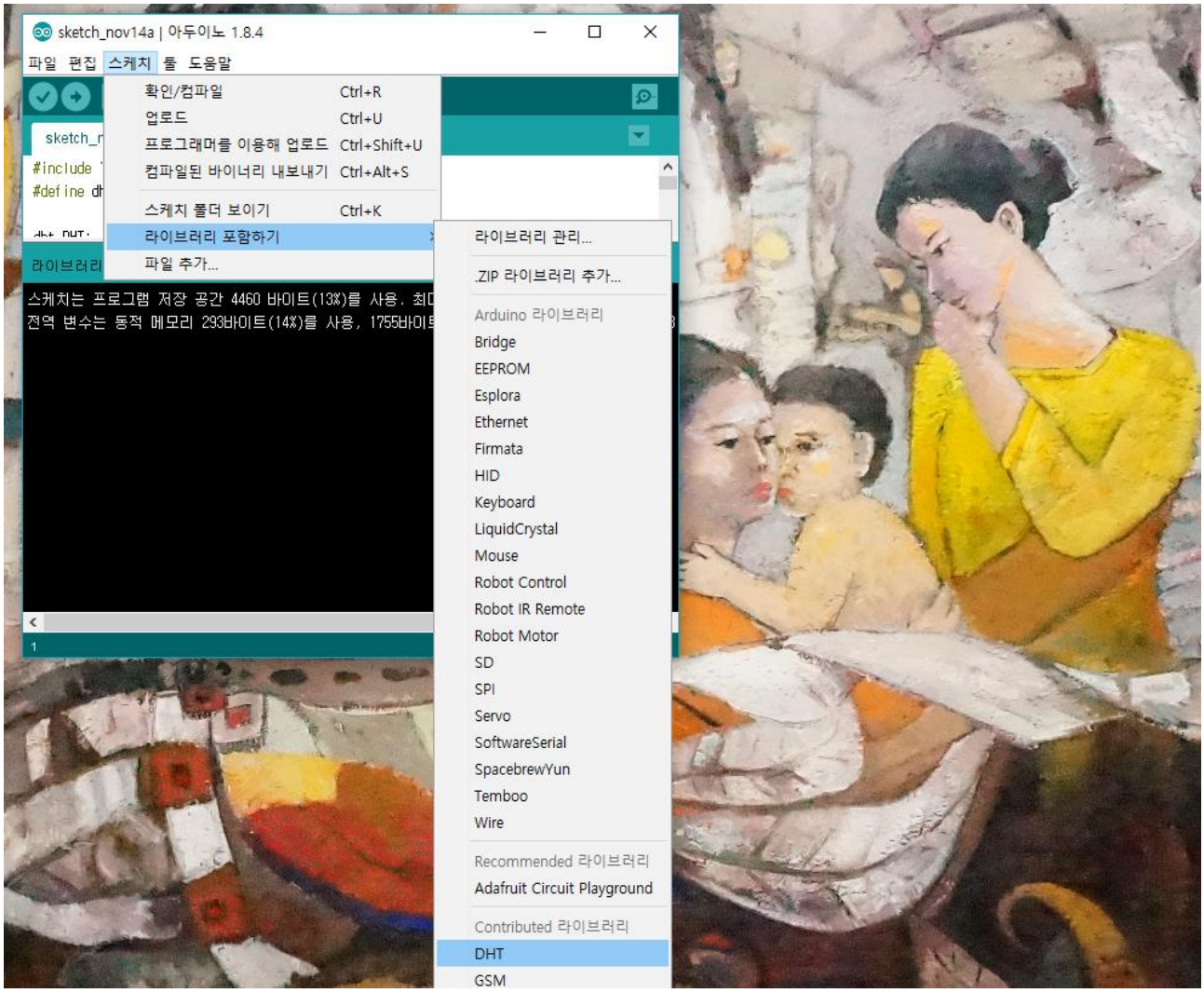
이를 프로그래밍하기 위해서는 DHT11 라이브러리를 설치하여야 한다.

DHT11.zip을 다운받아서

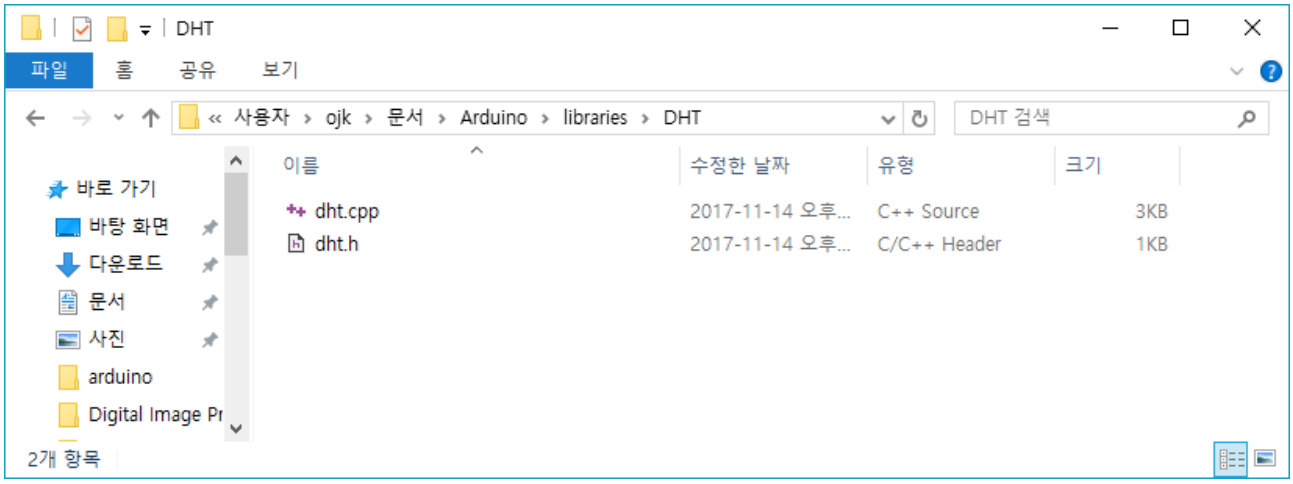
스케치 메뉴 – 스케치 – 라이브러리 포함하기 - .ZIP 라이브러치 추가를 클릭하여 추가한다.



메뉴-스케치-라이브러리 포함하기에 보면 다음 그림과 같이 DHT가 들어가 있다.



이 파일은 다음 그림과 같이 사용자의 홈디렉토리 아두이노 폴더에 들어 있다.


DHT
파일
홈
공유
보기
« 사용자 › ojk › 문서 › Arduino › libraries › DHT
DHT 검색
바로 가기
바탕 화면
다운로드
문서
사진
arduino
Digital Image Pr
이름
수정한 날짜
유형
크기
dht.cpp
2017-11-14 오후...
C++ Source
3KB
dht.h
2017-11-14 오후...
C/C++ Header
1KB
2개 항목

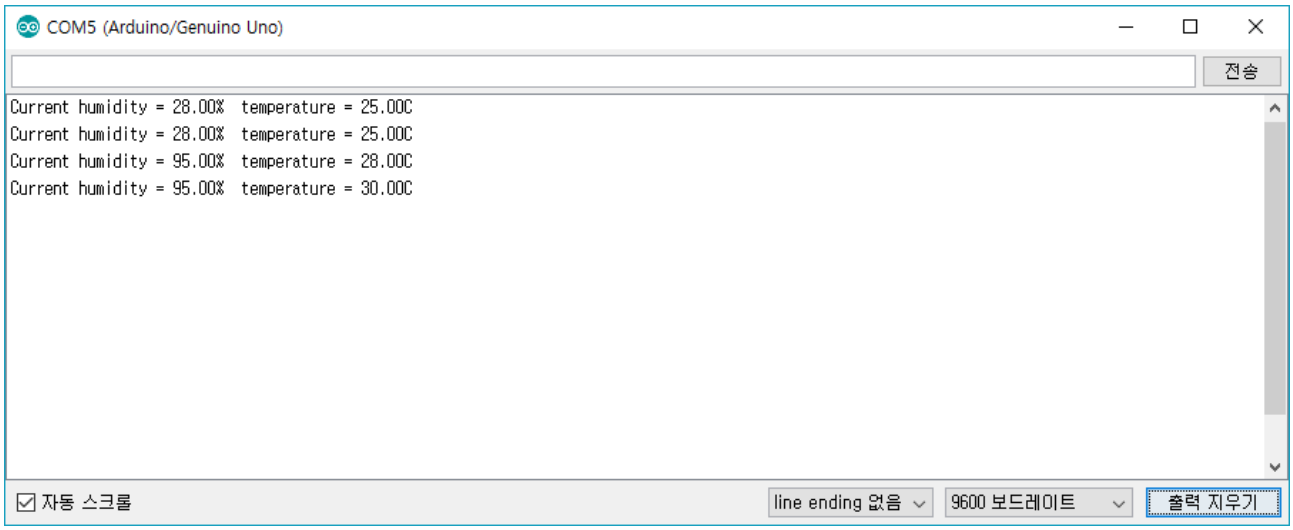
이제 프로그래밍을 해 보자.

라이브러리가 "dht.h"로 되어 있으므로

#include "dht.h" 를 사용하여 include하여야 한다.

```
#include "dht.h"

dht DHT;

void setup(){

  Serial.begin(9600);
  delay(500);   //Delay to let system boot
  Serial.println("DHT11 Humidity & temperature Sensor\n");
  delay(1000);  //Wait before accessing Sensor

}//end "setup()"

void loop(){

    DHT.read11(A0);

    Serial.print("Current humidity = ");
    Serial.print(DHT.humidity);
    Serial.print("%  ");
    Serial.print("temperature = ");
    Serial.print(DHT.temperature);
    Serial.println(" C  ");

    delay(5000);  //Wait 5 seconds before accessing sensor again.

  //  Fastest should be once every two seconds.

}// end loop()
```

처음 측정 값이 나와 있고, 습도와 온도를 올리기 위해 입으로 센서를 불게 되면 당연히 습도는 급격히 상승할 것이고, 온도은 약간 오를 것이다. 다음 그림은 그 결과이다.




```
//
//    FILE: dht.h
// VERSION: 0.1.00
// PURPOSE: DHT Temperature & Humidity Sensor library for Arduino
//
//     URL: http://arduino.cc/playground/Main/DHTLib
//
// HISTORY:
// see dht.cpp file
//

#ifndef dht_h
#define dht_h
```

```
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

#define DHT_LIB_VERSION "0.1.00"

class dht
{
public:
   int read11(uint8_t pin);
        int read22(uint8_t pin);
   double humidity;
   double temperature;

private:
   uint8_t bits[5];  // buffer to receive data
   int read(uint8_t pin);
};
#endif
//
// END OF FILE
```

```
//    FILE: dht22.cpp
// VERSION: 0.1.00
// PURPOSE: DHT22 Temperature & Humidity Sensor library for Arduino
//
// DATASHEET:
//
// HISTORY:
// 0.1.0 by Rob Tillaart (01/04/2011)
// inspired by DHT11 library
//
```

```
#include "dht.h"

#define TIMEOUT 10000

////////////////////////////////////////////////
//
// PUBLIC
//


// return values:
//  0 : OK
// -1 : checksum error
// -2 : timeout
int dht::read11(uint8_t pin)
{
  // READ VALUES
  int rv = read(pin);
  if (rv != 0) return rv;

  // CONVERT AND STORE
  humidity    = bits[0];  // bit[1] == 0;
  temperature = bits[2];  // bits[3] == 0;

  // TEST CHECKSUM
  uint8_t sum = bits[0] + bits[2]; // bits[1] && bits[3] both 0
  if (bits[4] != sum) return -1;

  return 0;
}

// return values:
//  0 : OK
// -1 : checksum error
// -2 : timeout
int dht::read22(uint8_t pin)
```

```
{
  // READ VALUES
  int rv = read(pin);
  if (rv != 0) return rv;

  // CONVERT AND STORE
  humidity    = word(bits[0], bits[1]) * 0.1;

  int sign = 1;
  if (bits[2] & 0x80) // negative temperature
  {
       bits[2] = bits[2] & 0x7F;
       sign = -1;
  }
  temperature = sign * word(bits[2], bits[3]) * 0.1;


  // TEST CHECKSUM
  uint8_t sum = bits[0] + bits[1] + bits[2] + bits[3];
  if (bits[4] != sum) return -1;

  return 0;
}

/////////////////////////////////////////////////
//
// PRIVATE
//

// return values:
//  0 : OK
// -2 : timeout
int dht::read(uint8_t pin)
{
  // INIT BUFFERVAR TO RECEIVE DATA
  uint8_t cnt = 7;
  uint8_t idx = 0;
```

```
// EMPTY BUFFER
for (int i=0; i< 5; i++) bits[i] = 0;

// REQUEST SAMPLE
pinMode(pin, OUTPUT);
digitalWrite(pin, LOW);
delay(20);
digitalWrite(pin, HIGH);
delayMicroseconds(40);
pinMode(pin, INPUT);

// GET ACKNOWLEDGE or TIMEOUT
unsigned int loopCnt = TIMEOUT;
while(digitalRead(pin) == LOW)
     if (loopCnt-- == 0) return -2;

loopCnt = TIMEOUT;
while(digitalRead(pin) == HIGH)
     if (loopCnt-- == 0) return -2;

// READ THE OUTPUT - 40 BITS => 5 BYTES
for (int i=0; i<40; i++)
{
     loopCnt = TIMEOUT;
     while(digitalRead(pin) == LOW)
          if (loopCnt-- == 0) return -2;

     unsigned long t = micros();

     loopCnt = TIMEOUT;
     while(digitalRead(pin) == HIGH)
          if (loopCnt-- == 0) return -2;

     if ((micros() - t) > 40) bits[idx] |= (1 << cnt);
     if (cnt == 0)   // next byte?
```

```
        {
             cnt = 7;
             idx++;
        }
        else cnt--;
   }

   return 0;
}
//
// END OF FILE
```

14. 서보 모터

다음 그림과 같이 서보모터는 축을 원하는 각도로 회전하고 그대로 멈출 수 있는 액추에이터입니다.

아두이노에 연결하여 사용하는 서보 모터는 보통 9g 서보모터입니다. 9g 서보모터의 경우 0 도에서 180 도까지 원하는 각도로 축을 회전시킬 수 있지만 180 도까지 회전하지 않는 서보모터도 있기 때문에 보통 120~ 130 도 정도까지만 회전하는 것으로 프로그래밍을 합니다.

서보모터는 선이 3 개입니다. 주황색, 빨강, 갈색인 경우에는 주황색이 디지털 핀, 빨강이 전원, 갈색이 그라운드이고 선이 노랑, 빨강, 검정인 경우에는 노랑이 디지털 핀, 빨강이 전원, 검정이 그라운드 선입니다.

【1】서보모터의 사용

서보모터를 사용하기 위해서는 Servo.h 헤더 파일을 인클루드하여야 하고 서보모터를 선언하여햔 합니다.

[라이브러리 사용]

```
#include <Servo.h>   // S 가 대문자인 것에 주의
Servo 서보모터이름;
```

ex)

```
#include <Servo.h>
Servero myServo;
```

[서보모터 연결]

```
서보모터이름.attach(디지털 핀번호)
```

ex)

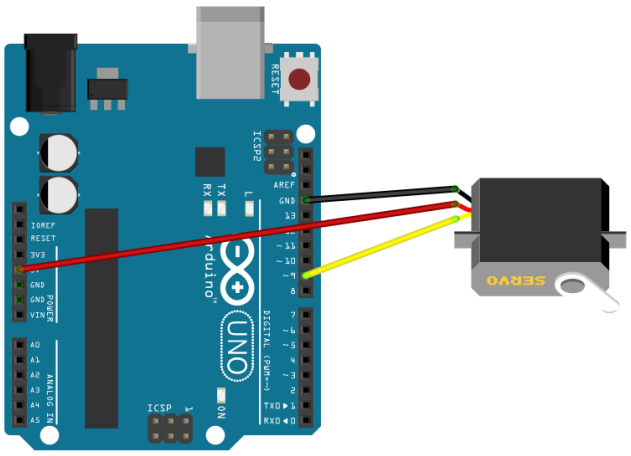
```
#include <Servo.h>
Servero myServo;
myServo.attach(9);   //디지털 9 번 핀에 연결
```

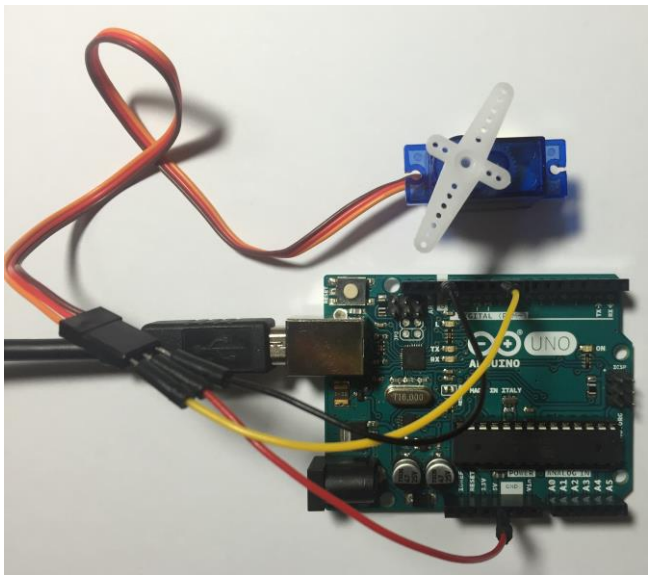
서보 모터의 회전

서보모터이름.write(각도) ;

ex)

```
myServo.write(100);    // 100 도 회전
myServo.write(i);    // 변수 i 에 저장되어 있는 수만큼 회전
```





처음위치에서 지정된 값의 각도까지 회전합니다.

myservo.write(90);
myservo.write(120);

위 2 명령을 실행하면 처음 위치에서 90 도 회전하고 다시 120 도 회전하는 것이 아니라 처음 위치에서 90 도 회전하고 다시 120 도까지 회전하므로 2 번째에서는 30 도만 회전하는 것이 됩니다. 따라서 이런 상황에서는 처음 명령문이 필요없고 단지 2 번째 명령으로 충분합니다.

서보모터를 130 도 회전시키고 다시 원위치로 오게 하자.

```
void setup() {
  // put your setup code here, to run once:
  myServo.attach(9);
  myServo.write(0);
  delay(1000);
}

void loop() {
```

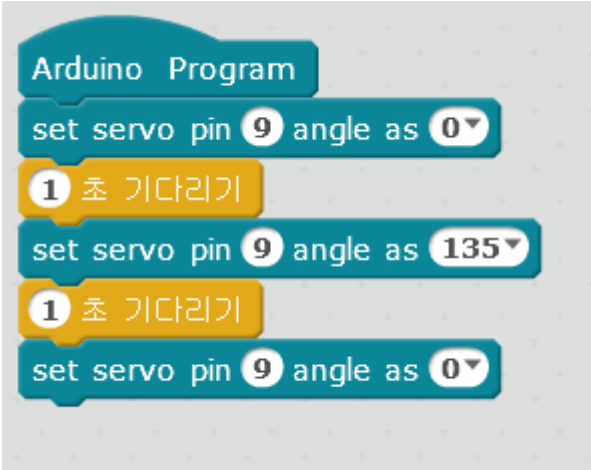
```
  myServo.write(130);
  delay(1000);
  myServo.write(0);
  delay(3000);
}
```

이제 서보 모터의 회전 속도를 제어하여 보자. 서보 모터의 회전 속도를 제어하는 방법은 약간 회전하고 delay하고 좀더 회전하는 방식을 사용한다. 여기에서 delay시간을 조절하면 회전 속도가 제어된다.

```
#include<Servo.h>
Servo myServo;
int speed=10;
int i;
int angle=130;
void setup() {
  // put your setup code here, to run once:
  myServo.attach(9);
  myServo.write(0);
  delay(1000);
}

void loop() {
  for (i=0;i<angle;i++){
    myServo.write(i);
    delay(speed);
  }
  delay(1000);

  myServo.write(0);
  delay(3000);
}
```

이제는 원래 위치로 돌아오는 속도도 같은 속도로 지정하여 보자.

```
#include<Servo.h>
Servo myServo;
int speed=10;
int i;
int angle=130;
void setup() {
  // put your setup code here, to run once:
  myServo.attach(9);
  myServo.write(0);
  delay(1000);
}

void loop() {
  for (i=0;i<angle;i++){
    myServo.write(i);
    delay(speed);
  }
  delay(1000);
  for (i=angle;i>0;i--){
    myServo.write(i);
    delay(speed);
  }
 delay(3000);
}
```


Arduino Program
set servo pin 9 angle as 0
1 초 기다리기
set servo pin 9 angle as 135
1 초 기다리기
set servo pin 9 angle as 0

map(value, fromLow, fromHigh, toLow, toHigh)

map( ) 함수는 숫자의 범위를 재설정해 준다. 즉, value 값의 fromLow는 toLow로, fromHigh는 toHigh로 바뀌며, 그 사이의 값은 비례배분한 값으로 바뀐다.

그러나 변환값은 상하한 값으로 한정되지는 않는데, 이렇게 범위를 벗어나는 것이 유용한 경우도 있기 때문이다. 따라서 상하한 값을 제한하기 위해서는 constrain( ) 함수를 map( ) 함수 전 또는 후에 사용해야 한다.

한가지 주의할 것은 fromLow가 fromHigh보다 크거나, toLow가 toHigh보다 클 수도 있다. 이렇게 하면 다음 예시와 같이 map( ) 함수를 이용하여 값을 역전시킬 수도 있다.

```
y = map(x, 1, 50, 50, 1); // (x,y) = (1, 50) .... (50, 1)
```

또한 음수 역시 가능한데, 다음 예시 역시 완벽하게 작동한다.

```
y = map(x, 1, 50, 50, -100); // (x,y) = (1, 50), (2, 47) .... (49, -96), (50, -100)
```

map( ) 함수는 정수형 산술을 하기 때문에 소수점은 다룰 수 없으며, 소수 이하는 버린다.

예시

```
void loop() {
  int val = analogRead(0);                // 아날로그 센서의 값을
읽는다. 0 ~ 1023 사이의 값
  val = map(val, 0, 1023, 0, 255);    // 아날로그 센서의 값을 0 ~ 255
사이의 값으로 변환
  analogWrite(9, val);                      // PWM 출력
 }
```

```
void loop() {
  int val = analogRead(0);                // 아날로그 센서의 값을
읽는다. 0 ~ 1023 사이의 값
  rotateAngle = map(val, 0, 1023, 0, 150);    // val 값을 0 ~ 150 사이의
값으로 변환
  myservoWrite(rotateAngle);             // 아날로그 센서의 값만큼
0 도에서 150 도까지 회전
}
```

map( ) 함수는 내부적으로 다음과 같이 계산한 값을 반환한다.

```
long map ( long x, long in_min, long in_max, long out_min, long out_max) {
```

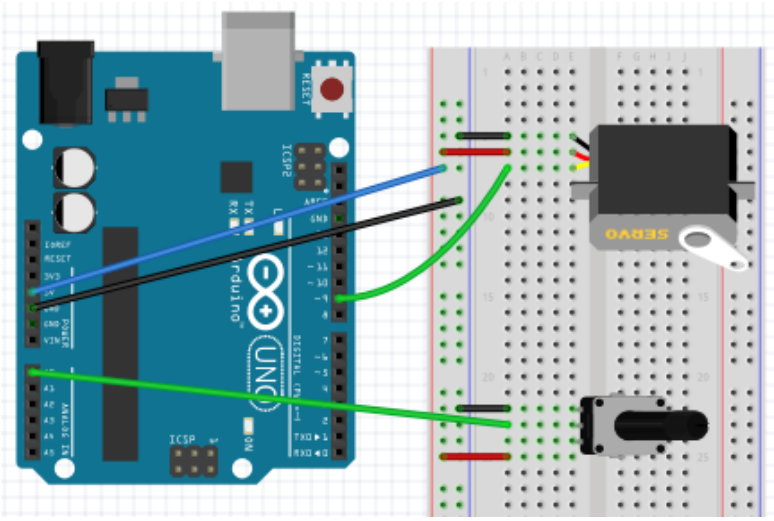
```
  retrun (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;

}
```

15. 가변 저항

가변 저항을 그림과 같이 다리가 3 개 이다. 양쪽 다리 2 개가 전원 또는 그라운드이고 가운데 다리가 저항 값에 따라서 전압이 변하는 것이다. 따라서 가운데 다리를 아날로그 입력 핀에 연결하면 된다. 가변 저항은 저항이므로 극성이 없다. 그러나 돌림에 따라서 저항이 변하므로 회전방향을 선택을 잘 해야 한다. 한 쪽이 전원이고 한 쪽이 그라운드이므로 전원 쪽으로 돌리는 경우에 저항이 적어지고 전압이 세진다. 오디오 볼륨 조절 장치를 생각해 보자. 보통 왼쪽에서 시계방향으로 돌려 볼륨을 키운다. 따라서 전원을 가변 저항의 오른쪽에 연결하고 왼쪽에는 그라운드를 연결하는 것이 합리적이다. 반대로 연결하면 당연히 방향이 바뀔 것이다.

가변 저항의 연결

왼쪽 다리 : 전원, 중간 : 아날로그 입력, 오른쪽 다리: 그라운드

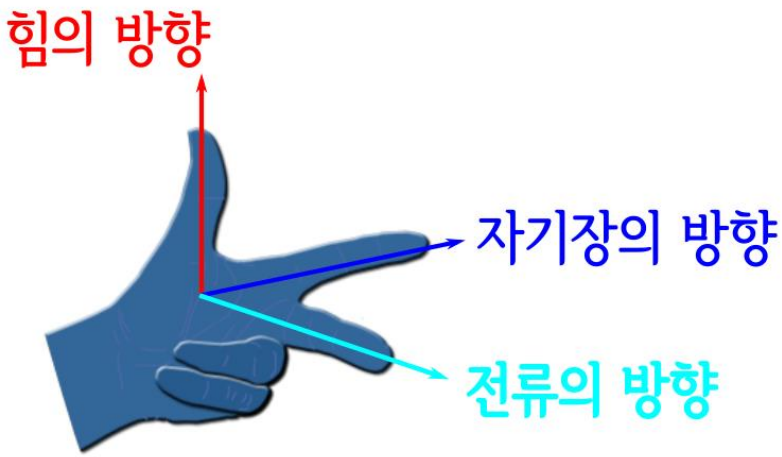


가변 저항이 왼쪽 끝에 있을 때 회전각도가 0 도, 오른쪽 끝에 있을 때 회전각도가 130 도가 되게 프로그래밍 해보자.

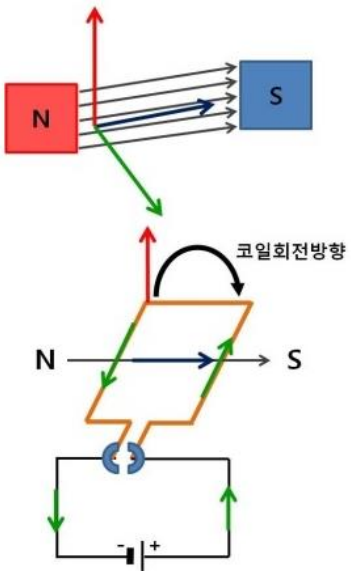
```
#include<Servo.h>
Servo myServo;
int speed=50, i, angle;
int reading;


void setup() {
  Serial.begin(9600);
  myServo.attach(9);
  myServo.write(0);
  delay(1000);
}

void loop() {
  reading = analogRead(A0);
  angle = map(reading, 0, 1023, 0, 130);
  Serial.println(angle);

  for (i = 0; i < angle; i++) {
    myServo.write(i);
    delay(speed);
  }
  delay(1000);
  for (i = angle; i > 0; i--) {
    myServo.write(i);
    delay(speed);
  }
  delay(3000);
}
```

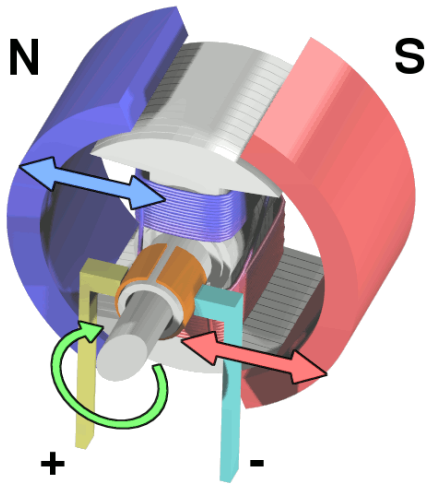
16. 직류 모터

다음은 플레밍의 왼손 법칙을 나타낸 그림입니다.

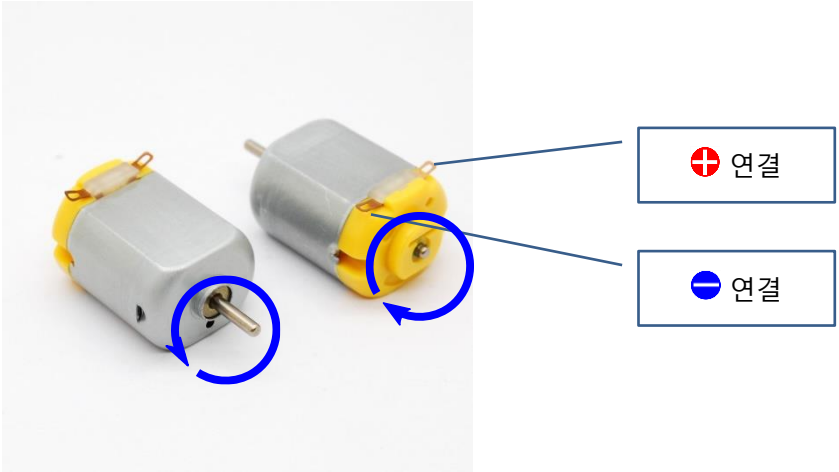


이러한 원리를 이용하여 직류 모터를 다음 그림과 같이 구성할 수 있습니다. 자기장의 방향에 맞게 자석을 배치하고 코일을 그 안에 두면, 전기를 가했을 때 시계방향 또는 반시계 방향으로 회전하게 됩니다.





직류 모터의 구조는 오른쪽 그림과 같습니다. 오른쪽 그림과 같이 제작하고 전원을 넣으면 시계 방향으로 회전합니다. 물론 +와 –전원을 바꾸어 넣으면 반시계 방향으로 회전합니다.

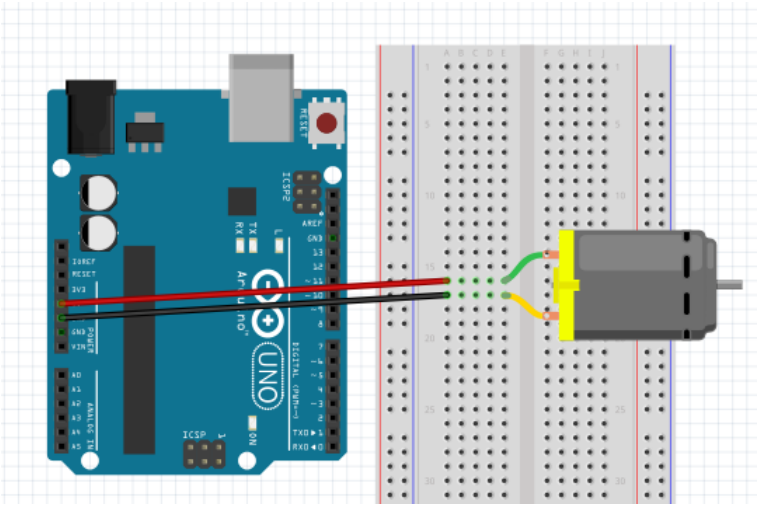


모터 속에는 코일이 들어 있습니다. 코일에 통했던 전기를 끊으면 갑자기 강한 전기가 생기면서 불꽃이 튕깁니다. 이 불꽃은 높은 전압의 전기(역기전력)입니다. 높은 전압의 전기가 아두이노 본체에 들어가면 아두이노가 망가질 수 있고, 이것을 막기위해 다이오드를 사용합니다.

다이오드는 전기가 한 쪽 방향으로만 흐르므로 반대 쪽으로 전기가 흐르는 것을 막아줍니다. 이 대신에 트랜지스터나 IC칩을 사용합니다.

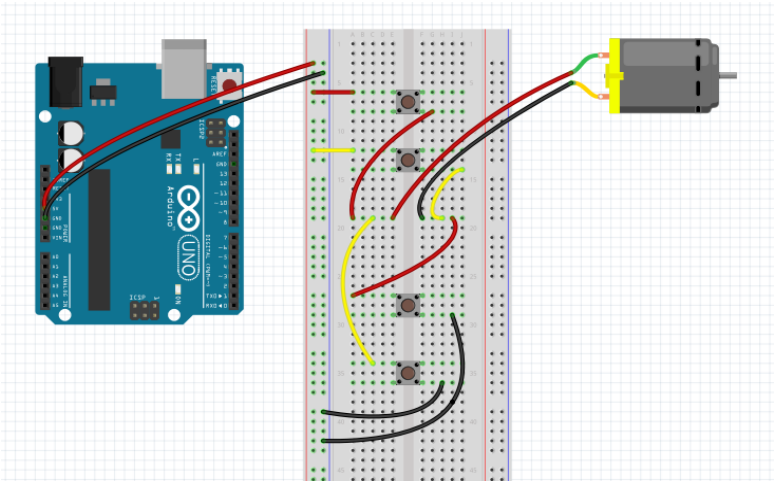
직류모터는 전기가 들어가면 회전하고 전기가 들어 가지 않으면 멈춥니다. 물론 관성의 법칙 때문에 전기가 들어가지 않더라고 더 회전하다가 천천히 멈추게 됩니다.

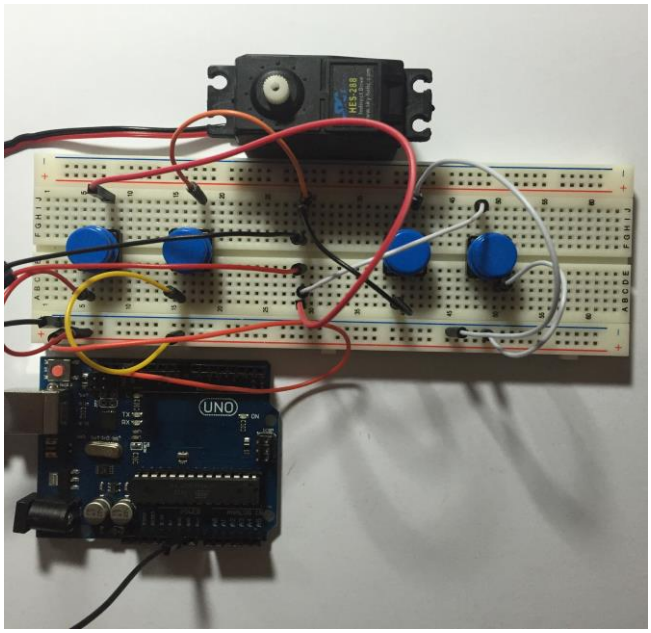
회전 방향을 바꾸기 위해서는 전기의 방향을 거꾸로 주면 됩니다. 다음 회로도처럼 전기의 방향을 바꾸어 봅시다.



회전 방향을 바꾸기 위해서 배선을 다시 하면 되지만 배선을 다시 하지 않고 다음 처럼 스위치 4 개를 사용하여 직류모터 방향 전환을 할 수 있습니다.

1 번째와 3 번째 스위치를 누르면 정방향으로 회전하고 2 번째와 4 번째를 누르면 역방향으로 회전합니다.





이런 방법 말고 다른 방법을 없을까요? 물론 있습니다. 전류의 방향을 바꾸어 주는 회로를 만들고 전류의 세기를 바꾸어 주는 회로를 사용하면 회전 방향과 회전 세기를 조절할 수 있다.

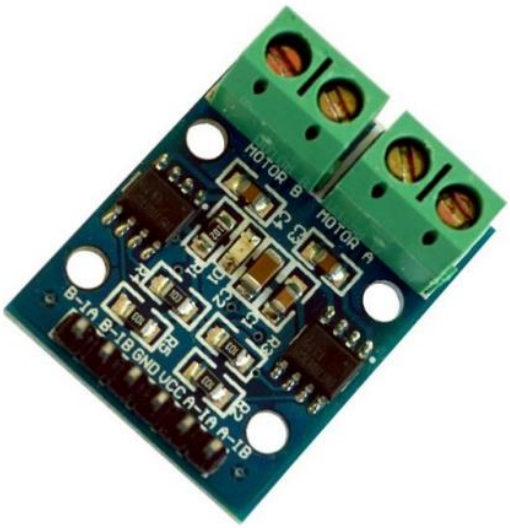
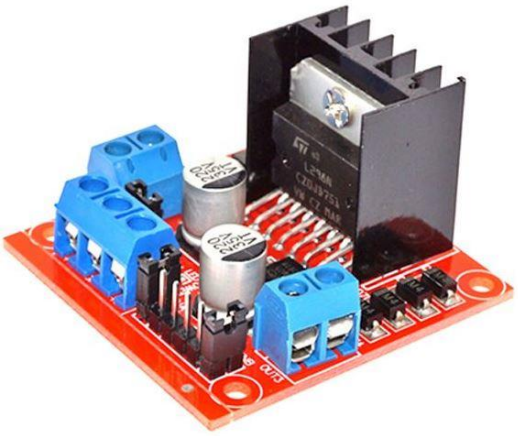
모터는 성격 상 직접 제어할 수 없기 때문에 다이오드와 트랜지스터를 사용하는 방법, 모터 드라이버 모듈을 사용하는 방법, 모터 드라이버 쉴드를 사용하는 방법 등이 있습니다. 물론 모터 쉴드를 사용하는 방법이 가장 편리하고 정확하나 개발과 학습을 위해서는 첫 번째와 두 번째 방법을 사용하는 것도 나쁘지 않습니다.

【1】 모터 드라이버 모듈

① 모터 드라이버 모듈의 원리와 연결

모터 드라이버의 종류

- **2핀으로 제어**
- **3핀으로 제어**

2 핀으로 제어하는 모듈	3 핀으로 제어하는 모듈
	

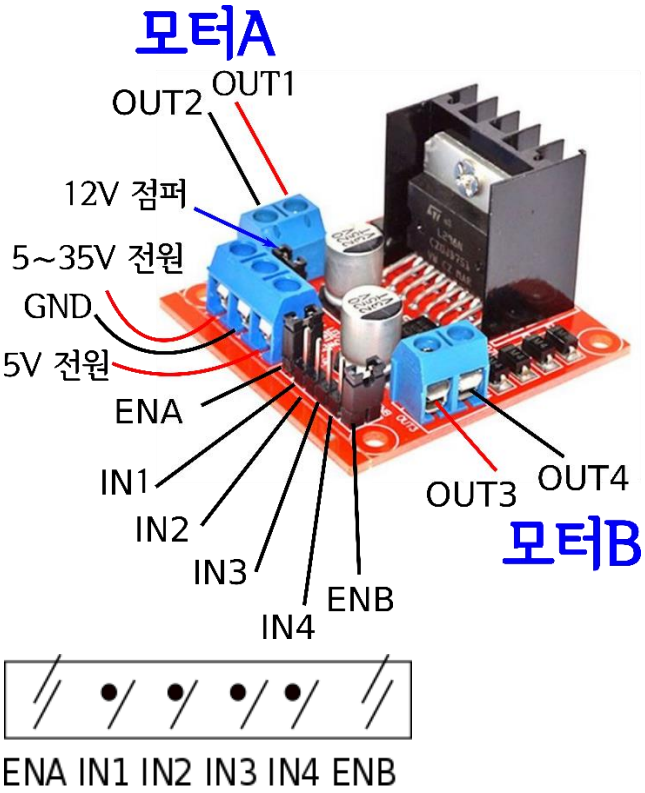
하나의 모터를 2 개의 핀으로 제어하는 모듈은 1 개의 핀으로 방향을 제어하고 1 개의 핀으로 속도를 제어합니다.

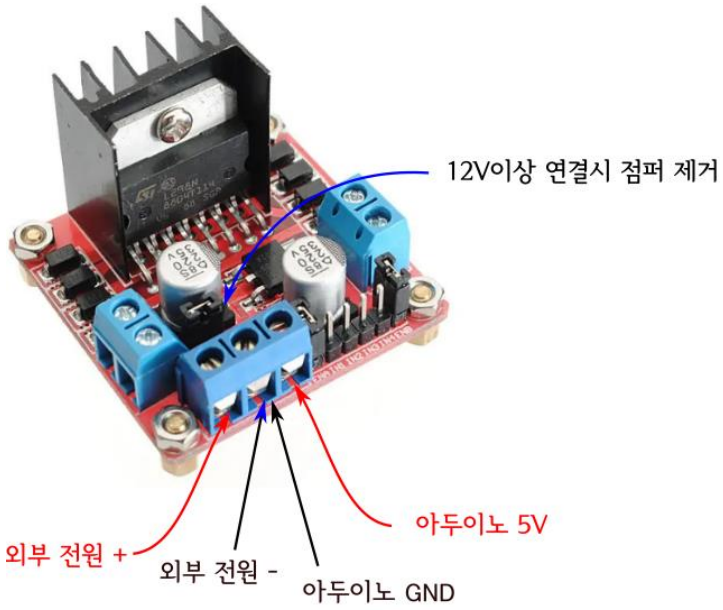
하나의 모터를 3 개의 핀으로 제어하는 모듈은 2 개의 핀으로 방향을 제어하고 1 개의 핀으로 속도를 제어합니다. 2 개의 핀으로 방향 제어를 할 때는 digitalWrite로 제어하고 속도는 analogWrite로 제어합니다. 따라서 속도를 제어하는 핀은 아나로그 출력이 가능한 핀으로 하여야 합니다.

속도와 방향 제어
방향 제어:디지털 핀1
방향 제어:디지털 핀1
속도 제어:아날로그 핀1

보통 모터드라이브 모듈은 모터 하나를 제어할 때는 방향 제어용으로 4, 5 번 핀과 속도 제어용으로 10 번 이상의 핀을 사용합니다.

직류모터 모듈(L298N, L298 Dual H Bridge IC) 사용





주의: 2 번째에는 2개를 같이 연결

외부 전원을 연결하지 않는 경우에는【아두이노 5V】와【아두이노 GND】를 연결하고, 외부 전원을 연결하는 경우에는 4 개를 동시에 연결합니다. 외부 전원이 있는 경우에는 아두이노 5V는 연결하지 않아도 되지만, 아두이노 GND는 반드시 연결하여야 합니다. 그러나 컴퓨터와 아두이노를 분리하는 경우에는 아두이노에도 전원을 공급하여야 하므로 4 개의 선을 전부 연결하여야 합니다.

12V 전원을 연결하는 것이 좋고 12V를 초과하는 경우에는 위 그림에 표시한 점퍼를 제거하여야 모터 드라이브 모듈의 손상을 막을 수 있습니다. 12V점퍼에 점퍼가 꽂혀 있으면 5V Linear Regulator에서 5V로 전환하여 내부 제어에 사용합니다.

ENA와 ENB는 점퍼가 꽂혀있는데 점퍼가 연결되어 있으면 항상 풀스피트(255)이므로 점퍼를 제거하면 속도의 제어(0~255:아날로그)가 가능합니다.

모터가 용량이 작은 경우에는 아두이노 전원 만으로 모터를 구동하고 제어할 수 있지만, 실제로는 아두이노 전원 만으로 구동할 수 있는 모터는 용량이 극히 작은 모터이므로 좀더 용량이 큰 모터를 사용하기 위해서는 외부 전원을 이용하여야 합니다.

결론적으로는 쓸 만한 모터를 사용하기 위해서는 외부 전원을 사용하여야 합니다. 아두이노에 전원을 공급하여야 하는 경우가 대부분이기 때문에 9V이상의 전원을 공급할 것을 권장합니다.

하지만 왼쪽 그림과 같은 건전지는 1.5V 건전지를 6 개를 연결한 것과 전압은 같으나 힘은 다르므로 권장하지 않습니다.

.

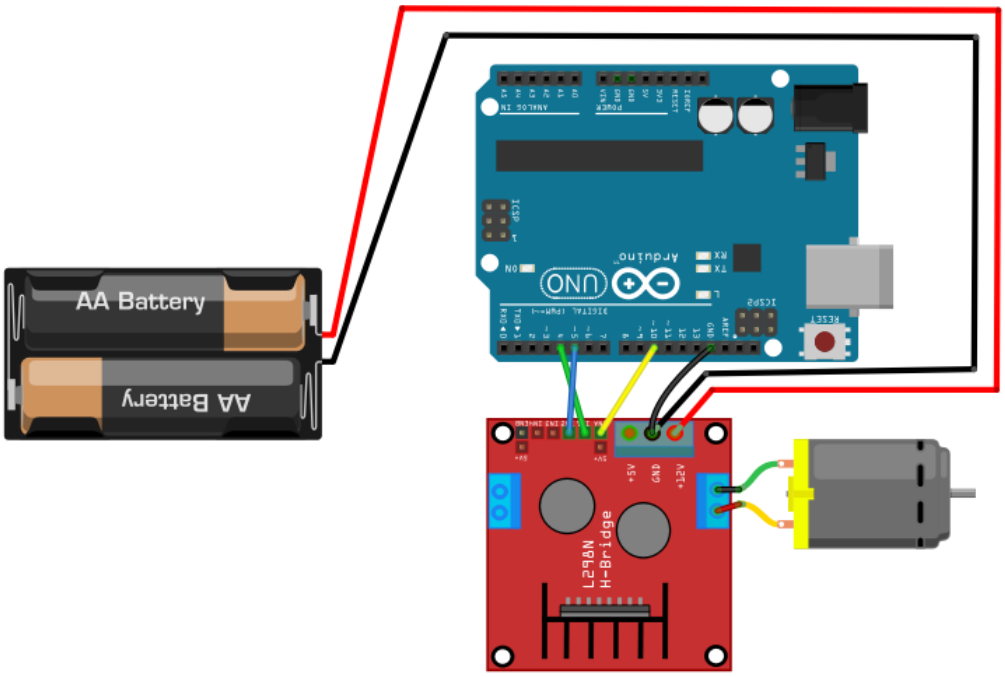
② 회전과 정지(기어드 모터 1 개)

연결과 회전 방향(3 핀 제어)

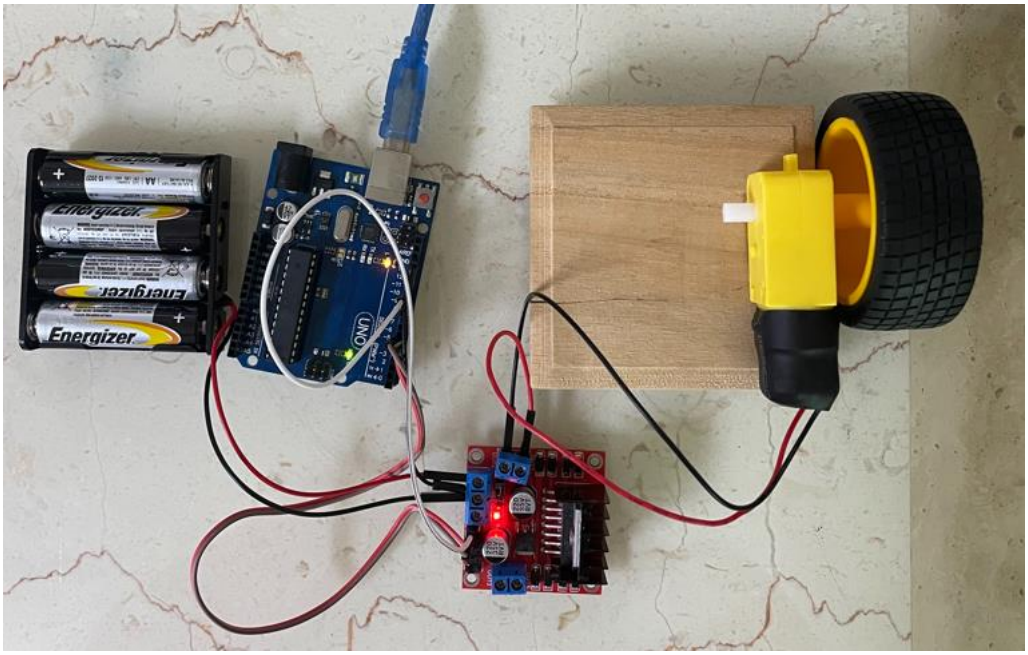
- IN1, IN2 에 A 모터 방향을 제어하는 디지털 출력을 연결하고
- ENA 에 꽂혀 있는 점퍼를 제거하고 A 모터의 속도를 제어하는 아날로그 핀을 연결합니다.
- IN1 에 HIGH 를 IN2 에 LOW 를 출력하면 A 모터 방향이 정방향(CW)이 되고,
- IN1 에 LOW 를 IN2 에 HIGH 를 출력하면 A 모터 방향이 역방향(CCW)이 됩니다.

물론 모터드라이브와 모터를 연결할 때 OUT1 에 ⊕을 연결하고 OUT2 에 ⊖를 연결하는 것이 일반적이지만, 이를 반대로 연결하면 모터는 당연히 반대로 회전합니다. 그렇지만 혼돈이 되기 때문에 일반적인 방법으로 연결하는 것이 합리적입니다. 불량 모터의 경우에는(제조자가 납땜을 잘못하는 경우) 반대로 회전하는 경우도 있겠지만 그런 경우는 거의 없습니다.

모터 하나를 사용하는 경우에는 IN1 : 4 번, IN2 : 5 번, ENA : 10 번 를 연결합니다.



모터는 녹색 선이 ⊕, 노랑색 선이 ⊖



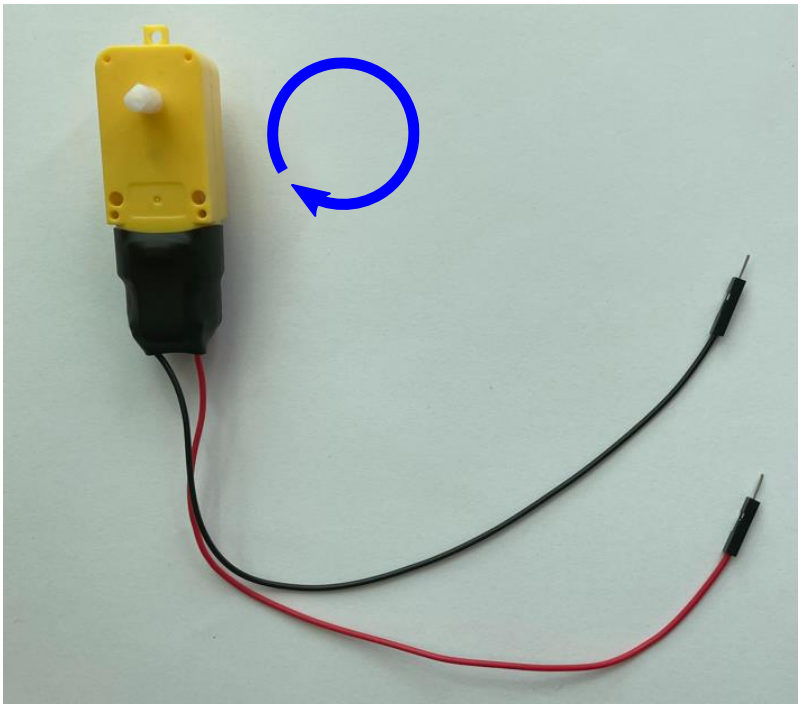
모터 1 개를 제어하는데 4 번 핀과 5 번 핀으로 방향을 제어하고, 10 번 핀으로 PWM 출력으로 속도를 제어합니다. 따라서 4 번 핀을 IN1, 5 번 핀을 IN2, 10 번 핀을 ENA로 설정합니다.

바퀴를 연결하여 회전할 수 있도록 제작되어 있는 모터를 기어드 모터라고 하는데, 기어모터와 바퀴

의 연결은 원하는 회전 방향으로 회전하도록 설치하여야 합니다.

장착

기어드 모터는 전원 선이 위에 위치한 기준으로 봤을 때(빨간 선이 오른 쪽, 검은 선이 왼쪽), 시계 방향으로 회전합니다.



이 기어드 모터를 전선이 밖으로 가도록(검은 선이 위, 빨간 선이 아래) 오른 쪽 바퀴에 설치하는 것이 원래의 설치 방법이고, 이렇게 설치하면 밖에서 봤을 때 정방향 전원을 주면 시계방향으로 회전하여 전진하게 됩니다.

3 초간 시계 방향으로 회전하여 전진하고, 2 초간 정지한 후에 다시 전진하는 것을 반복하여 봅시다.

【Sketch】

전진

【형식】

```
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
analogWrite(ENA, 255);
```

정지

【형식】

```
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
analogWrite(ENA, 0);
```

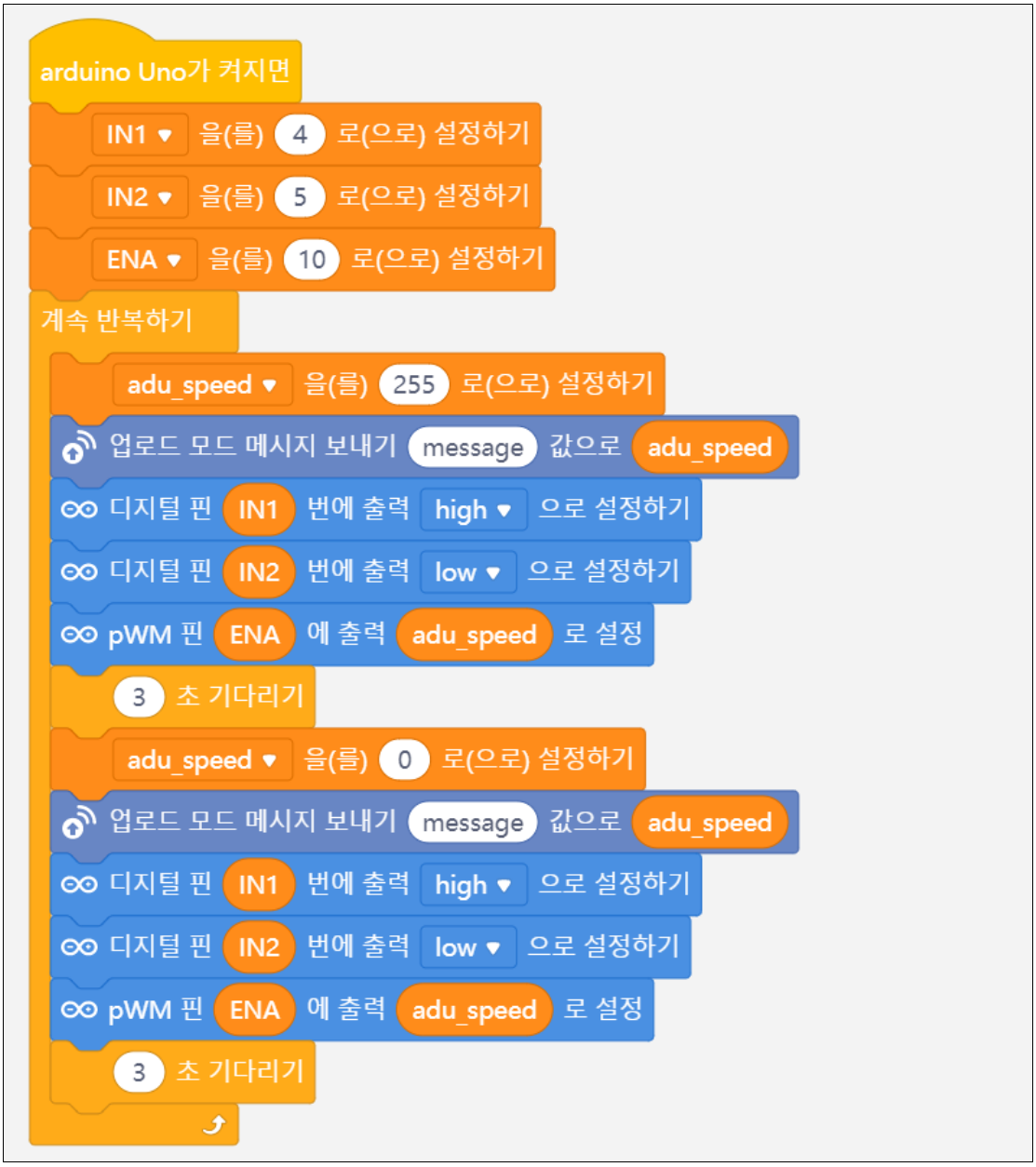
```
// File Name: ex100_Motor Module_01.ino
#define IN1 4
#define IN2 5
#define  ENA 10

void setup() {
 // Motor Driver Module L298N
Serial.begin(9600);
 pinMode(IN1, OUTPUT);   // Direction  Control
 pinMode(IN2, OUTPUT);    // Direction  Control
 pinMode(ENA, OUTPUT);   // Speed Control
 }

void loop() {
 Serial.println("CW :speed=255, Max");
 digitalWrite(IN1, HIGH);  digitalWrite(IN2, LOW);    // CW
 analogWrite(ENA, 255);
 delay(3000);

 Serial.println("CW :speed=0, Stop");
```

```
 digitalWrite(IN1, HIGH);  digitalWrite(IN2, LOW);    // CW
 analogWrite(ENA, 0);
 delay(2000);
}
```

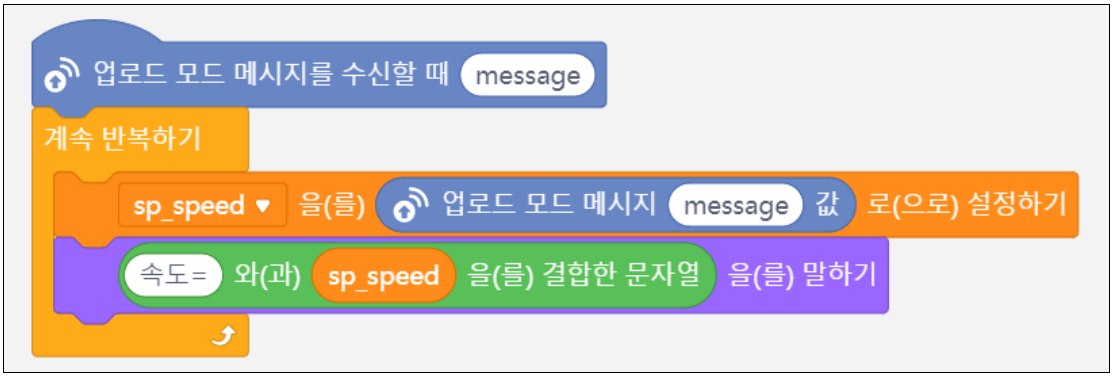
【mblock】

【코드 블록】

장치


arduino Uno가 켜지면
IN1 을(를) 4 로(으로) 설정하기
IN2 을(를) 5 로(으로) 설정하기
ENA 을(를) 10 로(으로) 설정하기
계속 반복하기
adu_speed 을(를) 255 로(으로) 설정하기
업로드 모드 메시지 보내기 message 값으로 adu_speed
디지털 핀 IN1 번에 출력 high 으로 설정하기
디지털 핀 IN2 번에 출력 low 으로 설정하기
pWM 핀 ENA 에 출력 adu_speed 로 설정
3 초 기다리기
adu_speed 을(를) 0 로(으로) 설정하기
업로드 모드 메시지 보내기 message 값으로 adu_speed
디지털 핀 IN1 번에 출력 high 으로 설정하기
디지털 핀 IN2 번에 출력 low 으로 설정하기
pWM 핀 ENA 에 출력 adu_speed 로 설정
3 초 기다리기

스프라이트



③ 전진과 정지(기어드 모터 2 개)

자동차가 전진하려면 오른 쪽 바퀴는 자동차 밖에서 봤을 때 시계방향으로 회전(안에서 봤을 때는 반시계 방향)하여야 하고, 왼쪽 바퀴는 자동차 밖에서 봤을 때 반시계 방향(안에서 봤을 때는 시계 방향)으로 회전하여야 합니다. 즉 오른 쪽 바퀴와 왼쪽 바퀴는 반대 방향으로 회전하여야 합니다.

오른 쪽 바퀴와 왼 쪽 바퀴는 반대방향으로 회전

이렇게 만드는 방법은 2 가지가 있습니다.

모터 설치과 신호 부여

모터 설치: 반대, 신호: 동일

모터 설치: 동일, 신호: 반대

첫 번째 방법

모터를 장착할 때 왼 쪽 모터와 오른 쪽 모터를 다른 방향으로 설치하고, 회전 시킬 때 신호를 같은 방식으로 주는 경우입니다.

좌우	모터설치		비고	방향

왼쪽	검은 선 위, 빨간 선 아래	IN1	HIGH	전진
		IN2	LOW	
오른쪽	검은 선 위, 빨간 선 아래	IN3	HIGH	전진
		IN4	LOW	

두 번째 방법

모터를 설치할 때 왼 쪽 모터와 오른 쪽 모터를 같은 방향으로 장착하고, 회전 시킬 때 신호를 반대로 주는 경우입니다.

좌우	모터설치	제어핀	신호	방향
왼쪽	검은 선 위, 빨간 선 아래	IN1	HIGH	전진
		IN2	LOW	
오른쪽	빨간 선 위, 검은 선 아래	IN3	LOW	전진
		IN4	HIGH	

어느 방법이 더 좋다고 말할 수는 없습니다.

첫 번째 방법은 장착을 반대로 하기 때문에 장착할 때는 혼돈이 오지만, 프로그래밍할 때는 왼 쪽과 오른 쪽을 동일하게 프로그래밍하므로 편리합니다.

두 번째 방식은 장착할 때는 편리하지만, 프로그래밍 할 때는 왼 쪽과 오른 쪽을 다르게 프로그래밍 하여야 하기 때문에 혼돈이 와서 프로그래밍이 더 어려워집니다. 일반적으로 두번째 방식을 많이 사용하고, 제작자 편의에 따라서 선택하면 됩니다.

먼저 첫 번째 방식으로 설치하여 보겠습니다. 각 바퀴는 다음 그림처럼 회전합니다.

왼쪽	오른쪽
	

다음 그림과 같이

오른 쪽 바퀴는 전선이 밖으로 위치하게(검은 선이 위로, 빨간 선이 아래로)모터와 바퀴를 연결하고

왼 쪽 바퀴는 전선이 안으로 위치하게(검은 선이 위로, 빨간 선이 아래로)모터와 바퀴를 연결하고

왼쪽과 오른쪽에 똑같이 정방향으로 신호를 주면 전진하고, 역방향으로 신호를 주면 후진합니다.

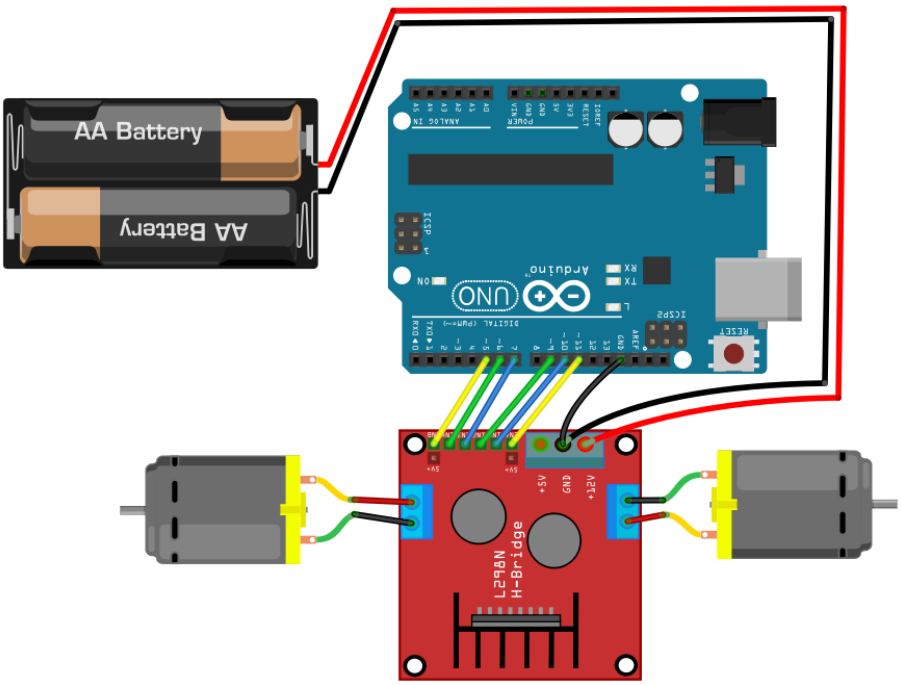
왼쪽 바퀴	오른쪽 바퀴
	

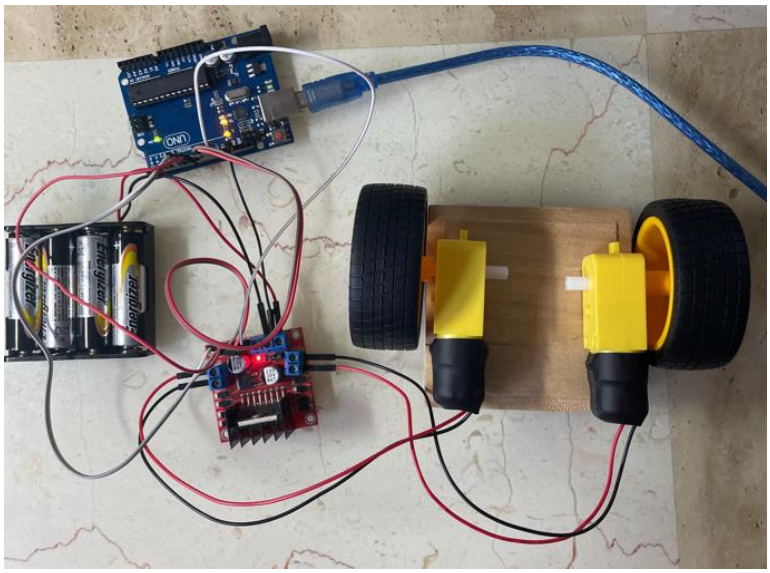
연결과 회전 방향

- IN1, IN2 에 A 모터 방향을 제어하는 디지털 출력을 연결하고
- IN3, IN4 에 B 모터 방향을 제어하는 디지털 출력을 연결합니다.
- ENA 에 꽂혀 있는 점퍼를 제거하고 A 모터의 속도를 제어하는 아날로그 핀을 연결합니다.
- ENB 에 꽂혀 있는 점퍼를 제거하고 B 모터의 속도를 제어하는 아날로그 핀을 연결합니다.
- IN1 에 HIGH 를 IN2 에 LOW 를 출력하면, A 모터 방향이 정방향(CW)이 되고,

- IN1 에 LOW 를 IN2 에 HIGH 를 출력하면, A 모터 방향이 역방향(CCW)이 됩니다
- IN3 에 HIGH 를 IN4 에 LOW 를 출력하면, B 모터 방향이 정방향(CW)이 되고,
- IN4 에 LOW 를 IN3 에 HIGH 를 출력하면, B 모터 방향이 역방향(CCW)이 됩니다.

L298N 모듈	아두이노 보드	비고
ENA	D11	PWM 포트
IN1	D10	디지털 포트
IN2	D9	디지털 포트
IN3	D7	디지털 포트
IN4	D6	디지털 포트
ENB	D5	PWM 포트





3 초간 시계 방향으로 회전하여 전진하고, 2 초간 정지한 후에 다시 전진하는 것을 반복하여 봅시다.

【Sketch】

【소스 프로그램】

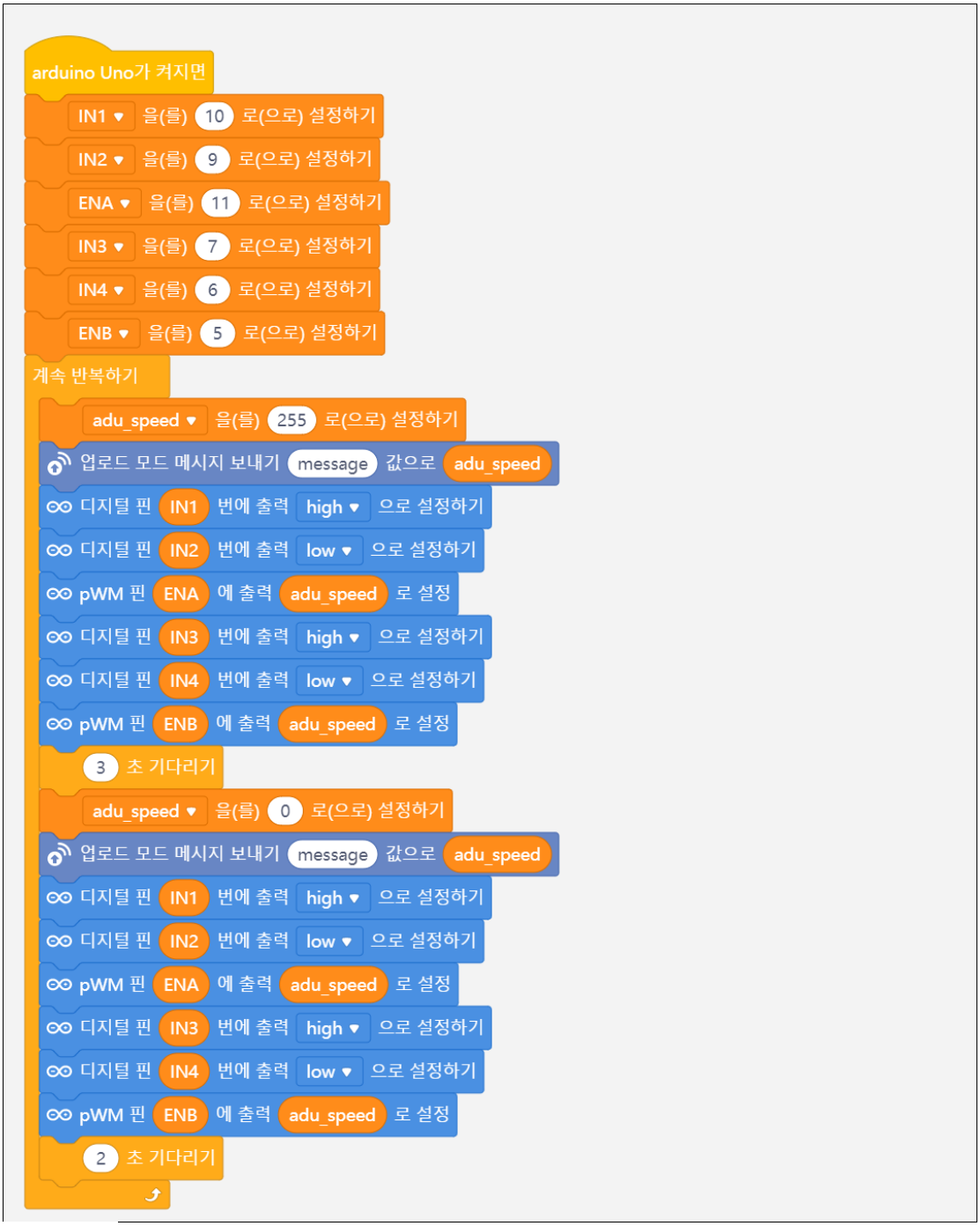
```
// File Name: ex103_Motor_Module_04.ino
#define IN1 10
#define IN2 9
#define IN3 7
#define IN4 6
#define ENA 11
#define ENB 5


void setup() {
  // Motor Module L298N
  Serial.begin(9600);
  pinMode(IN1, OUTPUT);      // Direction  Control
  pinMode(IN2, OUTPUT);      // Direction  Control
  pinMode(ENA, OUTPUT);      // Speed Control
  pinMode(IN1, OUTPUT);      // Direction  Control
  pinMode(IN2, OUTPUT);      // Direction  Control
  pinMode(ENA, OUTPUT);      // Speed Control
}

void loop() {
  Serial.println("CW :speed=255, Right Foreward");    // Right and Left
Foreward
  digitalWrite(IN1, HIGH);  digitalWrite(IN2, LOW);
  analogWrite(ENA, 255);
  Serial.println("CW :speed=255, Left Foreward");
  digitalWrite(IN3, HIGH);  digitalWrite(IN4, LOW);
  analogWrite(ENB, 255);
  delay(3000);

  Serial.println("CW :speed=0, Right Stop");    // Right and Left Stop
  digitalWrite(IN1, HIGH);  digitalWrite(IN2, LOW);
  analogWrite(ENA, 0);
  Serial.println("CW :speed=255, Left Stop");
  digitalWrite(IN3, HIGH);  digitalWrite(IN4, LOW);
  analogWrite(ENB, 0);
```

```
  delay(2000);
}
```

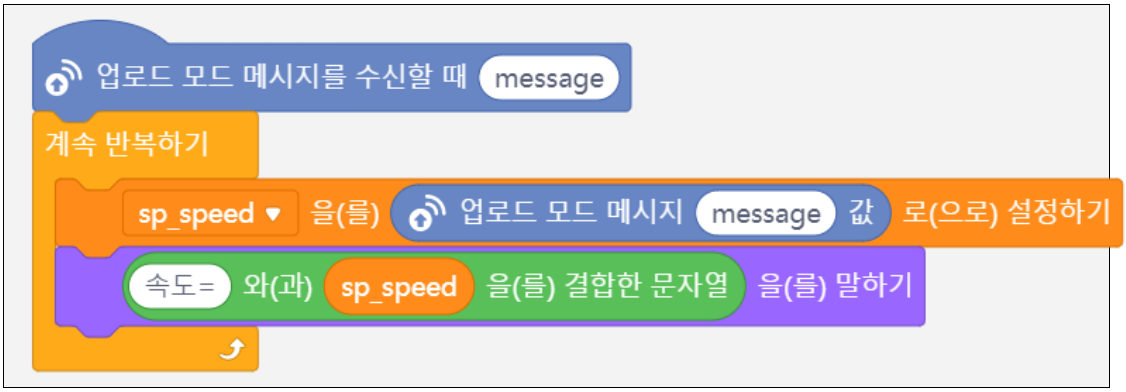
【mblock】

【코드 블록】

장치


arduino Uno가 켜지면
IN1 을(를) 10 로(으로) 설정하기
IN2 을(를) 9 로(으로) 설정하기
ENA 을(를) 11 로(으로) 설정하기
IN3 을(를) 7 로(으로) 설정하기
IN4 을(를) 6 로(으로) 설정하기
ENB 을(를) 5 로(으로) 설정하기
계속 반복하기
adu_speed 을(를) 255 로(으로) 설정하기
업로드 모드 메시지 보내기 message 값으로 adu_speed
디지털 핀 IN1 번에 출력 high 으로 설정하기
디지털 핀 IN2 번에 출력 low 으로 설정하기
pWM 핀 ENA 에 출력 adu_speed 로 설정
디지털 핀 IN3 번에 출력 high 으로 설정하기
디지털 핀 IN4 번에 출력 low 으로 설정하기
pWM 핀 ENB 에 출력 adu_speed 로 설정
3 초 기다리기
adu_speed 을(를) 0 로(으로) 설정하기
업로드 모드 메시지 보내기 message 값으로 adu_speed
디지털 핀 IN1 번에 출력 high 으로 설정하기
디지털 핀 IN2 번에 출력 low 으로 설정하기
pWM 핀 ENA 에 출력 adu_speed 로 설정
디지털 핀 IN3 번에 출력 high 으로 설정하기
디지털 핀 IN4 번에 출력 low 으로 설정하기
pWM 핀 ENB 에 출력 adu_speed 로 설정
2 초 기다리기

스프라이트

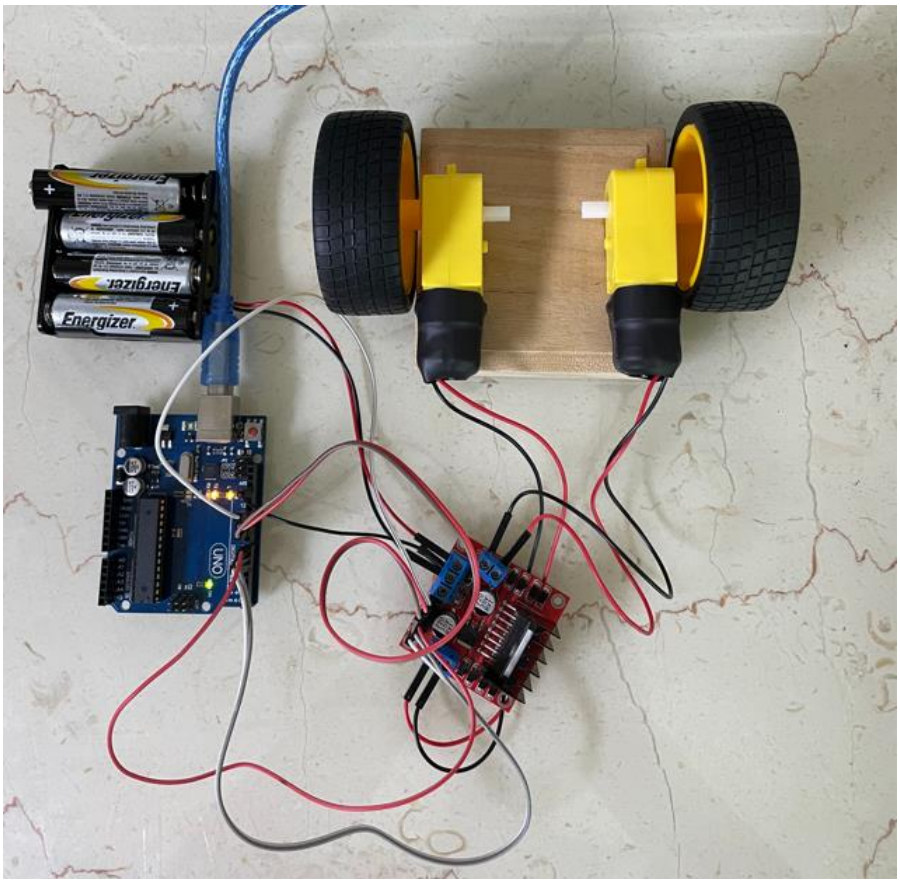


이제 두 번째 방식으로 장착하고 프로그래밍을 하여 봅시다.

다음 그림과 같이

오른 쪽 바퀴는 전선이 밖으로 위치하게(검은 선이 위로, 빨간 선이 아래로)모터와 바퀴를 연결하고

왼 쪽 바퀴도 전선이 밖으로 위치하게(빨간 선이 위로, 검은 선이 아래로)모터와 바퀴를 연결하고



오른쪽에는 정방향 신호를 주고 왼쪽에는 역방향 신호을 주면 전진하고,

오른쪽에는 역방향 신호를 주고 왼쪽에는 정방향 신호을 주면 후진합니다.

【Sketch】

【소스 프로그램】

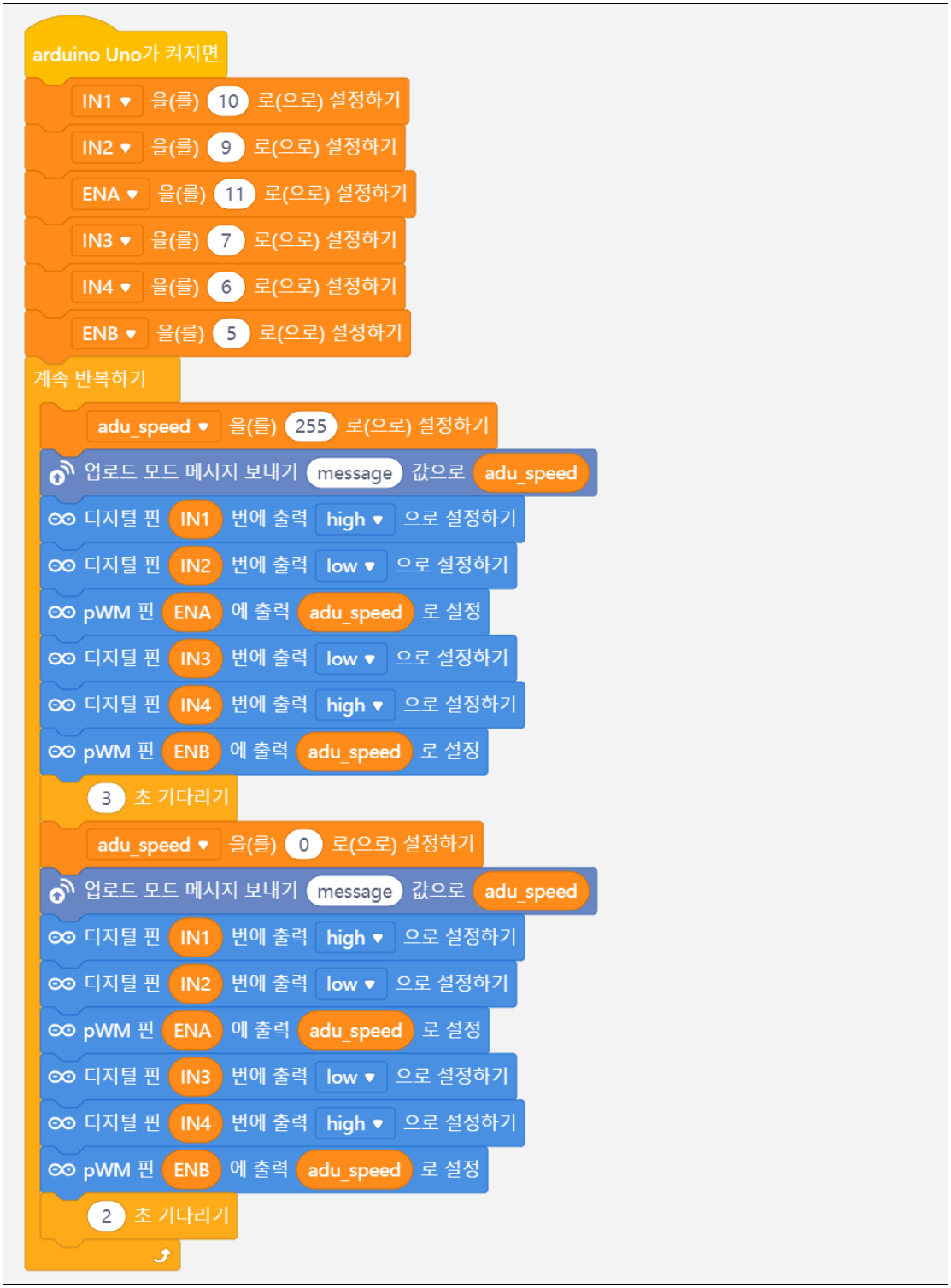
```
// File Name: ex104_Motor_Module_05.ino
#define IN1 10
#define IN2 9
#define IN3 7
#define IN4 6
#define ENA 11
#define ENB 5

void setup() {
  // Motor Module L298N
  Serial.begin(9600);
  pinMode(IN1, OUTPUT);      // Direction  Control
  pinMode(IN2, OUTPUT);      // Direction  Control
  pinMode(ENA, OUTPUT);      // Speed Control
  pinMode(IN1, OUTPUT);      // Direction  Control
  pinMode(IN2, OUTPUT);      // Direction  Control
  pinMode(ENA, OUTPUT);      // Speed Control
}

void loop() {
  Serial.println("CW :speed=255, Right Foreward");    // Right and Left
Foreward
  digitalWrite(IN1, HIGH);  digitalWrite(IN2, LOW);
  analogWrite(ENA, 255);
  Serial.println("CW :speed=255, Left Foreward");
  digitalWrite(IN3, LOW);  digitalWrite(IN4, HIGH);
  analogWrite(ENB, 255);
  delay(3000);

  Serial.println("CW :speed=0, Right Stop");    // Right and Left Stop
  digitalWrite(IN1, HIGH);  digitalWrite(IN2, LOW);
```

```
  analogWrite(ENA, 0);
  Serial.println("CW :speed=255, Left Stop");
  digitalWrite(IN3, LOW);  digitalWrite(IN4, HIGH);
  analogWrite(ENB, 0);
  delay(2000);
}
```

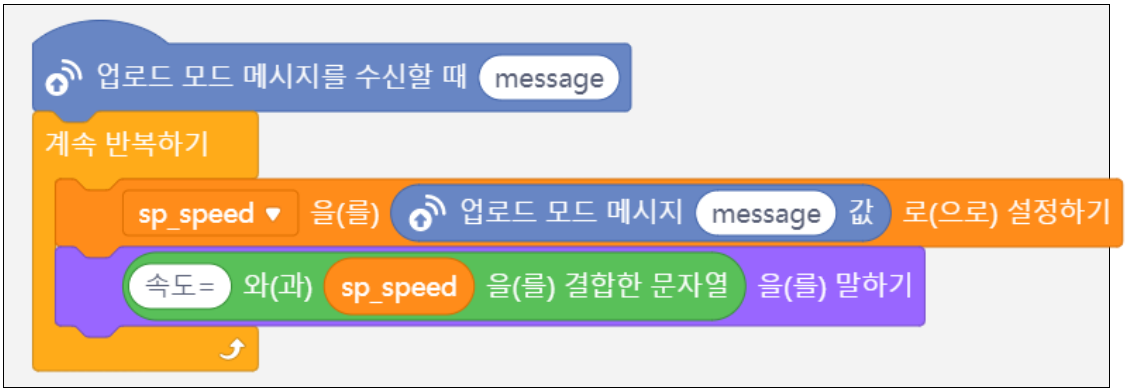
【mblock】

【코드 블록】

장치

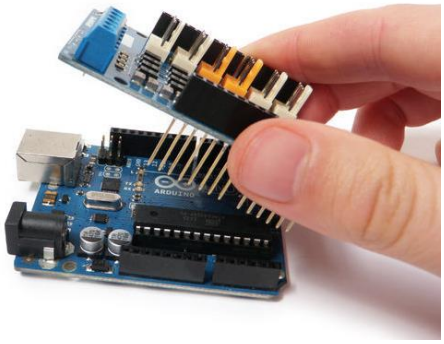

arduino Uno가 켜지면
IN1 을(를) 10 로(으로) 설정하기
IN2 을(를) 9 로(으로) 설정하기
ENA 을(를) 11 로(으로) 설정하기
IN3 을(를) 7 로(으로) 설정하기
IN4 을(를) 6 로(으로) 설정하기
ENB 을(를) 5 로(으로) 설정하기
계속 반복하기
adu_speed 을(를) 255 로(으로) 설정하기
업로드 모드 메시지 보내기 message 값으로 adu_speed
디지털 핀 IN1 번에 출력 high 으로 설정하기
디지털 핀 IN2 번에 출력 low 으로 설정하기
pWM 핀 ENA 에 출력 adu_speed 로 설정
디지털 핀 IN3 번에 출력 low 으로 설정하기
디지털 핀 IN4 번에 출력 high 으로 설정하기
pWM 핀 ENB 에 출력 adu_speed 로 설정
3 초 기다리기
adu_speed 을(를) 0 로(으로) 설정하기
업로드 모드 메시지 보내기 message 값으로 adu_speed
디지털 핀 IN1 번에 출력 high 으로 설정하기
디지털 핀 IN2 번에 출력 low 으로 설정하기
pWM 핀 ENA 에 출력 adu_speed 로 설정
디지털 핀 IN3 번에 출력 low 으로 설정하기
디지털 핀 IN4 번에 출력 high 으로 설정하기
pWM 핀 ENB 에 출력 adu_speed 로 설정
2 초 기다리기

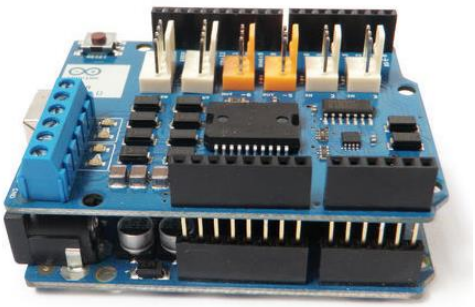
스프라이트


업로드 모드 메시지를 수신할 때
message
계속 반복하기
sp_speed
을(를)
업로드 모드 메시지
message
값
로(으로) 설정하기
속도=
와(과)
sp_speed
을(를) 결합한 문자열
을(를) 말하기

【2】모터 드라이버 쉴드

모듈은 아두이노와 전선으로 연결하여 사용하는 장치이지만 쉴드는 아두이노의 핀에 맞게 제작되어 있어서 아두이노에 끼워서 사용하는 장치입니다. 모터 드라이버 쉴드는 모터드리이버 모듈과 같은 기능을 가지고 있지만 쉴드이기 때문에 다음 그림처럼 아두이노에 끼워서 사용하는 장치입니다.





모터 드라이버 쉴드의 종류(핀 제어 방식)
2개의 핀으로 제어
3개의 핀으로 제어

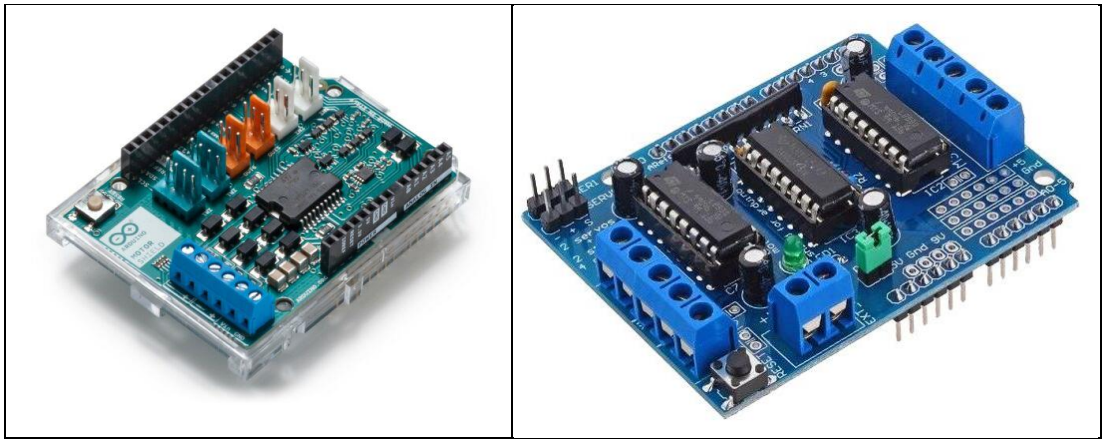
모터 드라이버 쉴드의 종류(제어 가능 모터 개수)
모터 2개 제어 가능
모터 4개 제어 가능

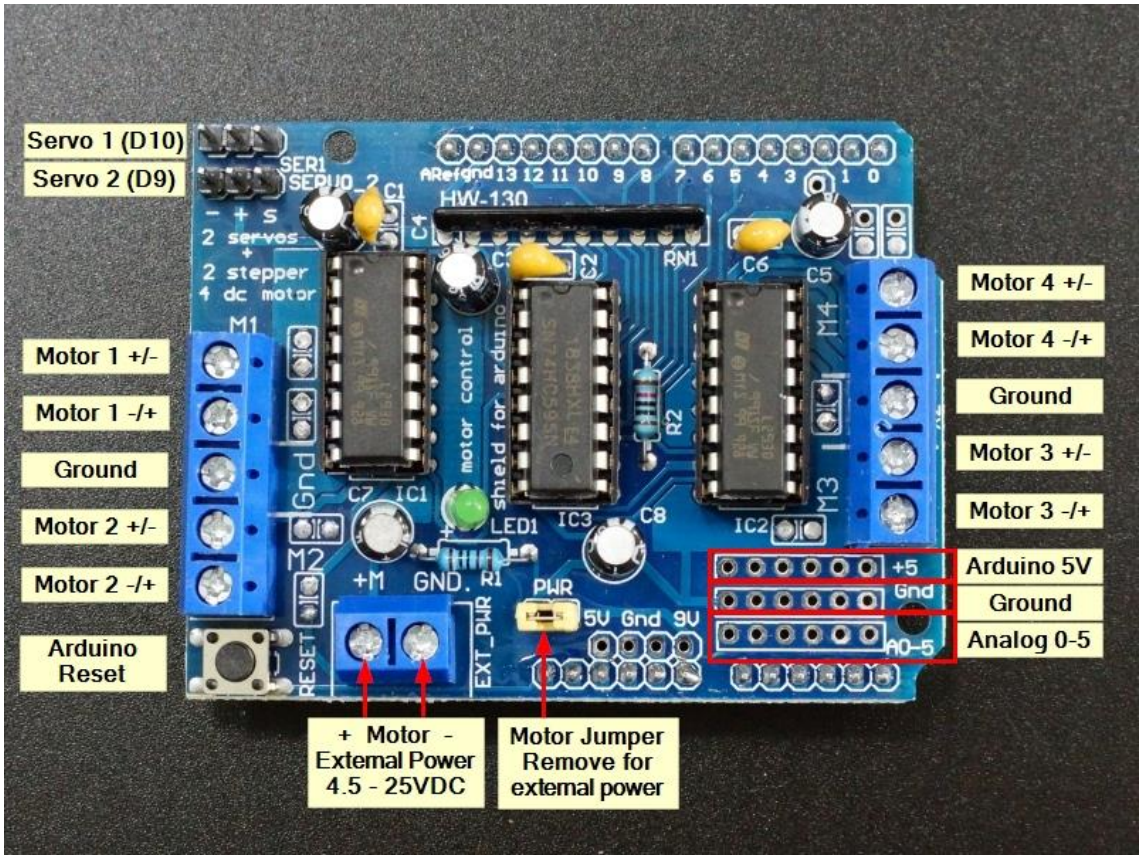
2 개의 핀으로 제어하는 쉴드는 1 개의 핀으로는 방향을 제어하고, 나머지 1 개 의 핀으로 속도를 제어합니다. 당연히 속도를 제어하는 핀은 아날로그 출력이 가능하여 합니다.

3 개의 핀으로 모터를 제어하는 모터쉴드는 2 개의 핀으로는 방향을 제어하고, 나머지 1 개의 핀으로 속도를 제어합니다.

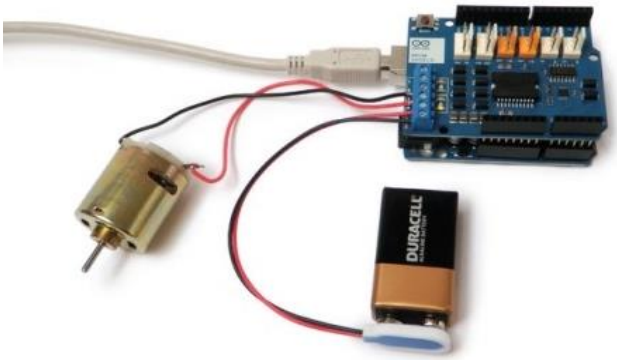
또한 모터 1 개만 제어하는 쉴드는 없고, 보통 2 개 또는 4 개의 모터를 제어 가능하게 제작되어 있습니다.

2 개 제어 가능	4 개 제어 가능

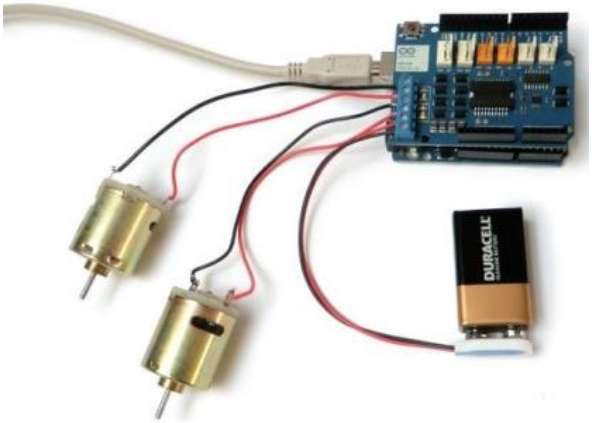



Servo 1 (D10)
Servo 2 (D9)
Motor 1 +/-
Motor 1 -/+
Ground
Motor 2 +/-
Motor 2 -/+
Arduino Reset
Motor 4 +/-
Motor 4 -/+
Ground
Motor 3 +/-
Motor 3 -/+
Arduino 5V
Ground
Analog 0-5
+ Motor - External Power 4.5 - 25VDC
Motor Jumper Remove for external power
HW-130
2 servos
2 stepper
4 dc motor
motor control shield for arduino
M1
M2
M3
M4
LED1
PWR
RESET
EXT_PWR

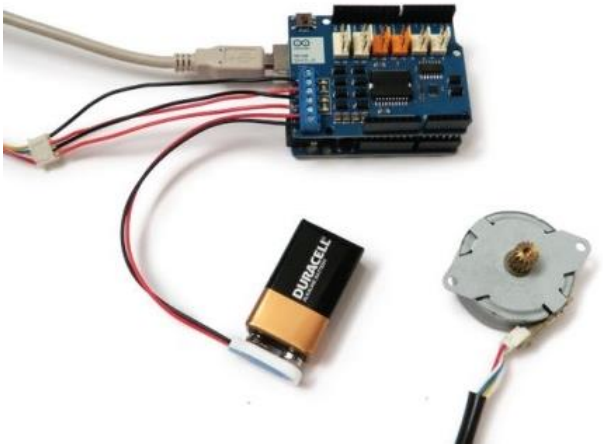
외부 전원을 연결한다. 모터 쉴드에 전원의 극성이 표시되어 있다.



모터 2 개까지 연결하여 제어할 수 있습니다.



스텝 모터도 제어 가능합니다. 모터의 속도제어는 PWM방식으로 할 수 있고 alalogWrite 함수를 사용하여 다양한 회전 속도를 만들어 낼 수 있습니다.



3 개의 핀으로 제어

3 개의 핀으로 모터를 제어하는 모터쉴드는 2 개의 핀으로 방향을 제어하고, 나머지 1 개의 핀으로 속도를 제어한다.

ENA 에 아날로그 신호를 출력하여 속도를 제어하고

IN1 에 HIGH, IN2 에 LOW를 출력하면 CW로 회전하고

IN1 에 LOW, IN2 에 HIGH를 출력하면 CCW로 회전한다.

ENB 에 아날로그 신호를 출력하여 속도를 제어하고

IN3 에 HIGH, IN4 에 LOW를 출력하면 CW로 회전하고

IN3 에 LOW, IN4 에 HIGH를 출력하면 CCW로 회전한다.

IN1 에 HIGH, IN2 에 HIGH 를 출력하면 정지하고

IN3 에 HIGH, IN4 에 HIGH 를 출력하면 정지하고

아두이노 우노 R3 & L298N 제어 포트 연결

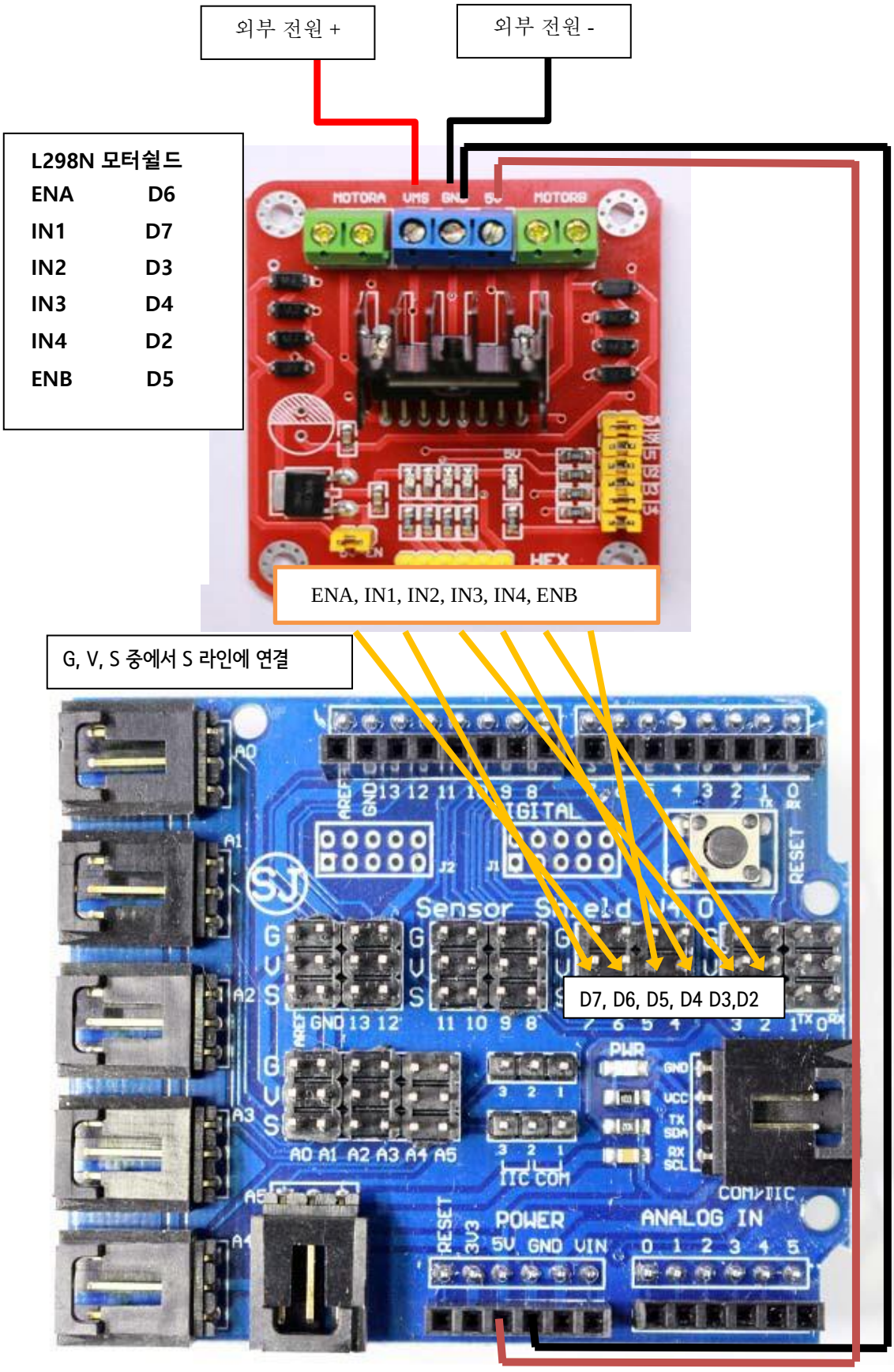
운행 방향, 속도 제어하기 위해 L298N 제어 보드의 포트와 연결해야 합니다. ENA, IN1, IN2, (D3, D4, D5), ENB IN3, IN4, ENB (D9, D8, D7) 순서로 연결 합니다.

물론, 아두이노 보드와 연결 시 같은 성격의 해당 핀에 연결하여도 무방합니다. ENA, ENB 는 PWM 지원 포트에 연결합니다.

L298N 모듈	아두이노 보드	비고
ENA	D6	PWM 포트
IN1	D7	디지털 포트
IN2	D3	디지털 포트
IN4	D2	디지털 포트
IN3	D4	디지털 포트
ENB	D5	PWM 포트

4WD 스마트 로봇 자동차를 만들기 위한 초음파 센서, 서보모터, 적외선 수신모듈, L298N 제어보드와 연결됩니다. 연결하기 편한 GVS 포트를 사용하여도 되며, 기존 아두이노핀 배열 형태의 Female 헤더도 지원됩니다.

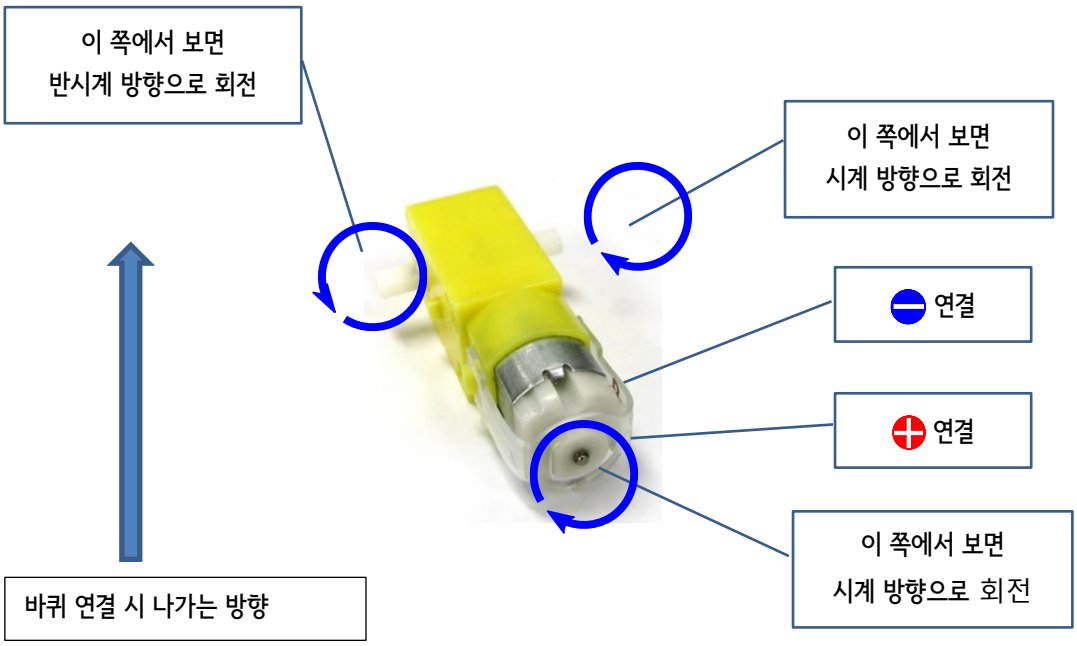
당연히 속도를 제어하는 핀은 아날로그 출력이 가능하여 한다.


외부 전원 +
외부 전원 -
L298N 모터쉴드
ENA D6
IN1 D7
IN2 D3
IN3 D4
IN4 D2
ENB D5
ENA, IN1, IN2, IN3, IN4, ENB
G, V, S 중에서 S 라인에 연결
D7, D6, D5, D4 D3,D2

ENA, EN1, EN2(6, 7, 3) : 오른쪽 바퀴(MOTORA), ENB, EN4, EN3 : 왼쪽 바퀴(MOTORB)

기어드 모터

기어드 모터는 그림과 같이 바퀴를 연결하여 사용할 수 있는 모터이다. 직류 모터로 바퀴를 돌리기에는 구조상 불편하기 때문에 이러한 기어가 추가된 모터를 사용한다.


이 쪽에서 보면
반시계 방향으로 회전
이 쪽에서 보면
시계 방향으로 회전
연결
연결
이 쪽에서 보면
시계 방향으로 회전
바퀴 연결 시 나가는 방향

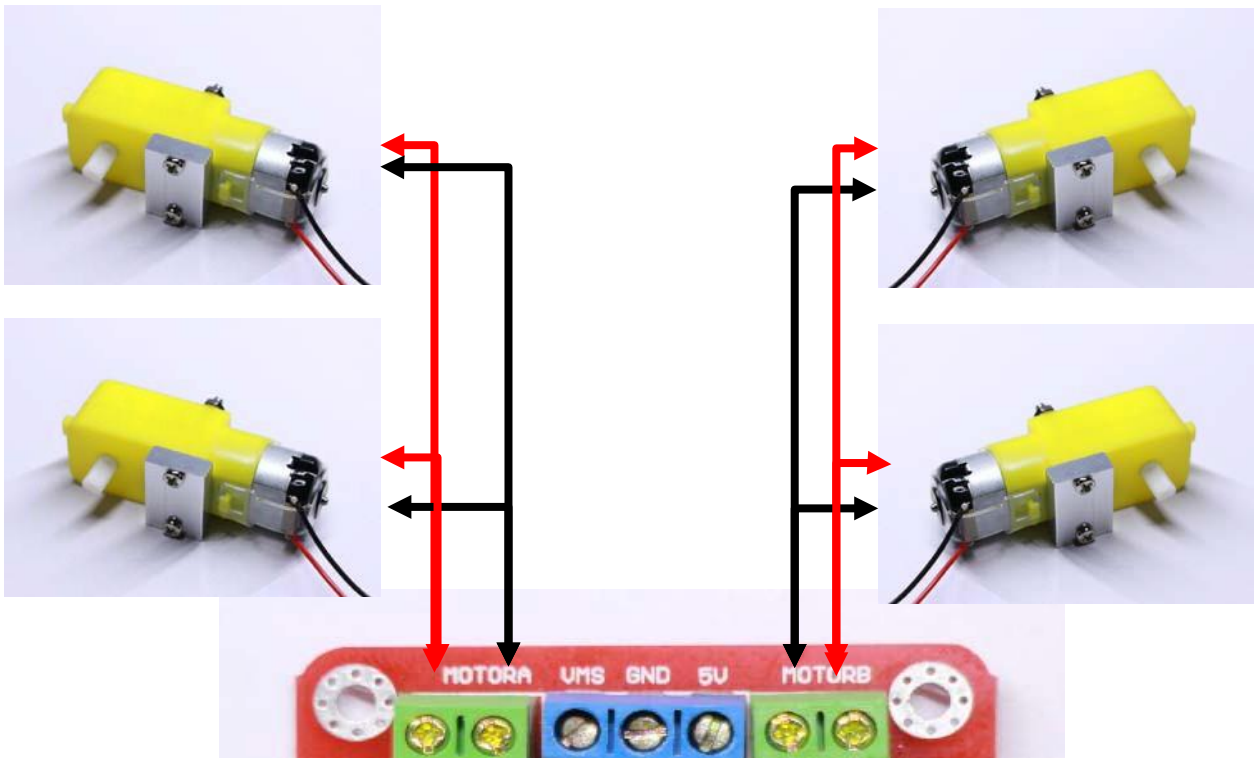
모든 DC 모터는 정방향 전원을 공급하면 시계 방향(CW)으로 회전하고,
역방향으로 전원을 공급하면 반시계 방향(CCW)으로 회전하게 설계되어 있다.

처음 조립할 때에 정방향 전원을 공급하면 왼쪽은 시계방향으로 회전하고
오른쪽은 반시계 방향으로 회전하게 조립하였다.

RC카를 조립하는 경우에 일반적으로 정방향 전원을 공급하면, 위에서 봤을 때
왼쪽은 밖에서 봤을 때, 시계 방향 회전 앞으로 가게하고
오른쪽은 밖에서 봤을 때, 반시계 방향 회전하여 뒤로 가고

따라서 RC카를 앞으로 나가게 하려면 왼쪽은 정방향, 오른쪽은 역방향 전원을 주어야 한다.

RC카를 조립하는 경우에 일반적으로 정방향 전원을 공급하면, 위에서 봤을 때
왼쪽은 앞으로 가게하고(시계 방향 회전)
오른쪽은 뒤로 가게 한다(반시계 방향 회전)



MOTORA에 오른쪽 바퀴를 연결하고 MOTORB에 왼쪽 바퀴를 연결했다.

따라서 전진하기 위해서는

왼쪽 바퀴가 연결되어 있는 모터쉴드는 정방향을 주고, 오른쪽 방향은 역방향을 주어야 하고

후진하기 위해서는

왼쪽 바퀴가 연결되어 있는 모터쉴드는 역방향을 주고, 오른쪽 방향은 정방향을 주어야 하고

현재 4 륜 모터는 왼쪽을 IN3, IN4, ENB(D2, D4, D5)로 제어하고

오른쪽을 ENA, IN1, IN2(D6, D7, D3)로 제어한다.

ENA, ENB에 가까운 쪽으로 LOW, HIGH 를 주면 정회전

HIGH, LOW 를 주면 역회전

ENA 에 아날로그 신호를 출력하여 속도를 제어하고

IN1 에 LOW, IN2 에 HIGH를 출력하면 CW로 회전하고

IN1 에 HIGH, IN2 에 LOW를 출력하면 CCW로 회전한다.

ENB 에 아날로그 신호를 출력하여 속도를 제어하고

IN4 에 LOW, IN3 에 HIGH를 출력하면 CW로 회전하고

IN4 에 HIGH, IN3 에 LOW를 출력하면 CCW로 회전한다.

		오른쪽 바퀴			왼쪽 바퀴		
모터 모듈		ENA	IN1	IN2	IN3	IN4	ENB
아두이노 핀		D6	D7	D3	D4	D2	D5
방향	정방향	전원	LOW	HIGH	HIGH	LOW	전원
	역방향	전원	HIGH	LOW	LOW	HIGH	전원

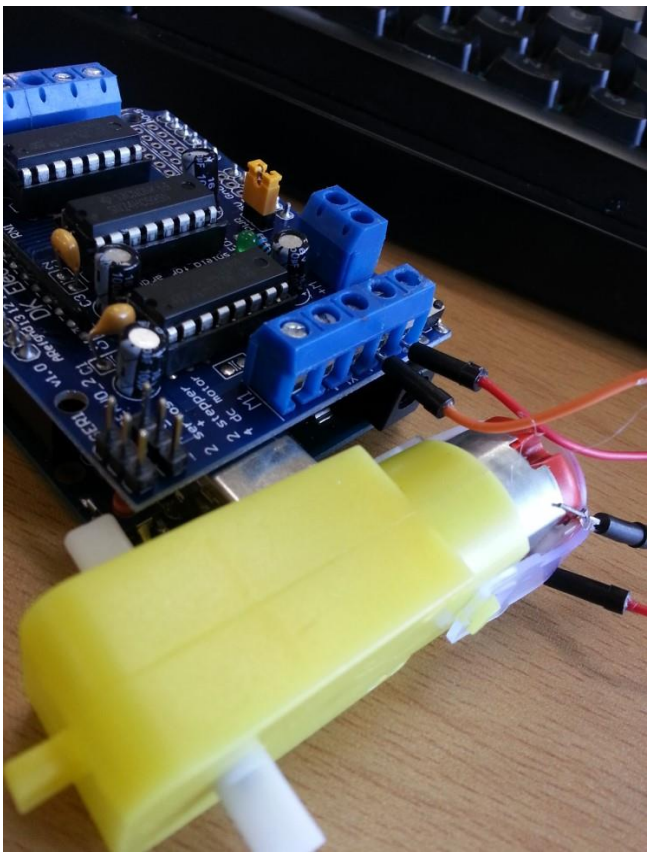
따라서 전진하기 위해서는 왼쪽은 정방향, 오른쪽은 역방향을 주어야 하므로

오른쪽 바퀴			왼쪽 바퀴		
ENA	IN1	IN2	IN3	IN4	ENB
D6	D7	D3	D4	D2	D5
전원	HIGH	LOW	HIGH	LOW	전원

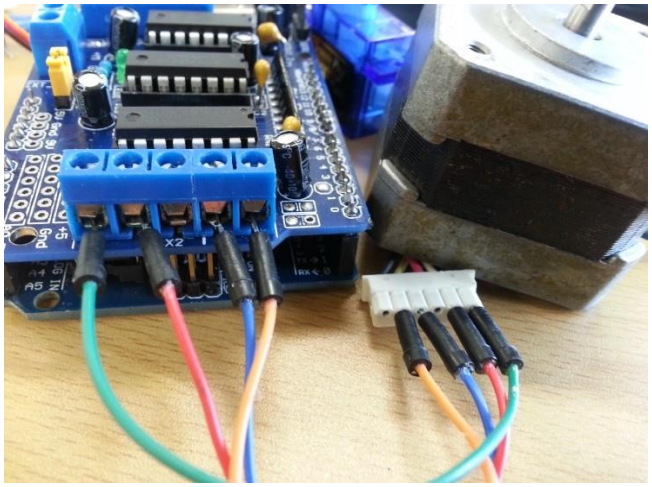

아두이노 프로그램
leftSpeed 을(를) 5 로 정하기
leftWard1 을(를) 2 로 정하기
leftWard2 을(를) 4 로 정하기
rightSpeed 을(를) 6 로 정하기
rightWard1 을(를) 7 로 정하기
rightWard2 을(를) 3 로 정하기
왼쪽 : 5, 2, 4
오른쪽 : 6, 7, 3
무한 반복하기
PWM leftSpeed 핀에 255 보내기
디지털 leftWard1 핀에 꺼짐 보내기
디지털 leftWard2 핀에 켜짐 보내기
PWM rightSpeed 핀에 255 보내기
디지털 rightWard1 핀에 켜짐 보내기
디지털 rightWard2 핀에 꺼짐 보내기
왼쪽 바퀴 전진(CW)
IN4: LOW, IN3: HIGH
오른쪽 바퀴 전진(CCW)
IN1: HIGH, IN2: LOW

서보모터 : 최대 2 개, DC 모터 : 최대 4 개, 스텝모터 : 최대 2 개 연결 가능

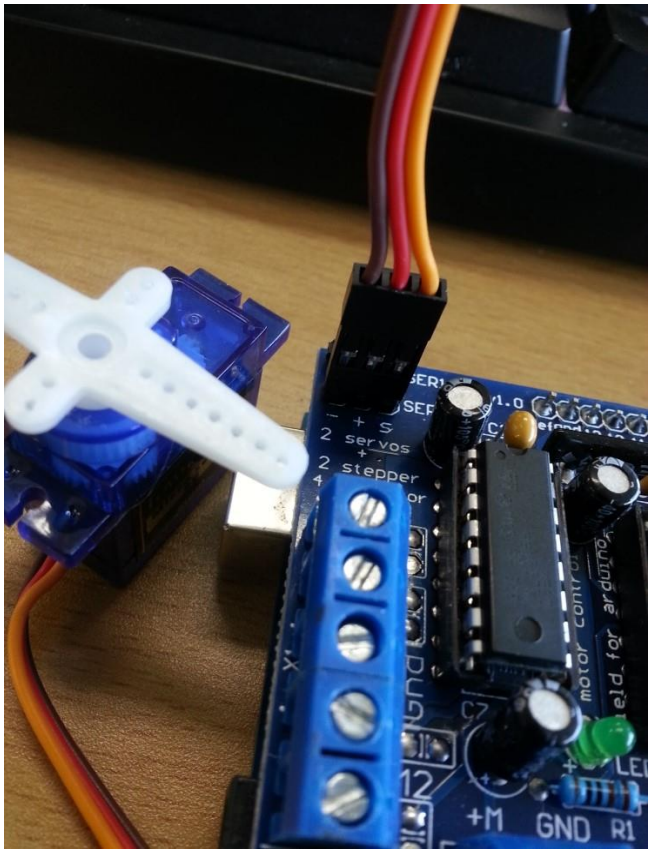
DC 모터 연결



스텝모터 연결

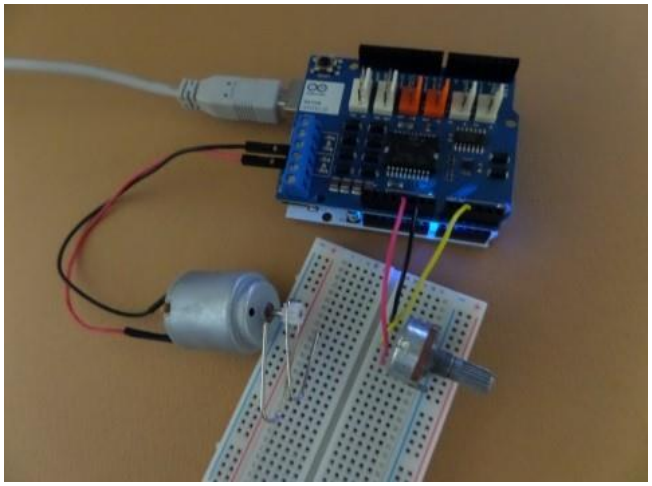


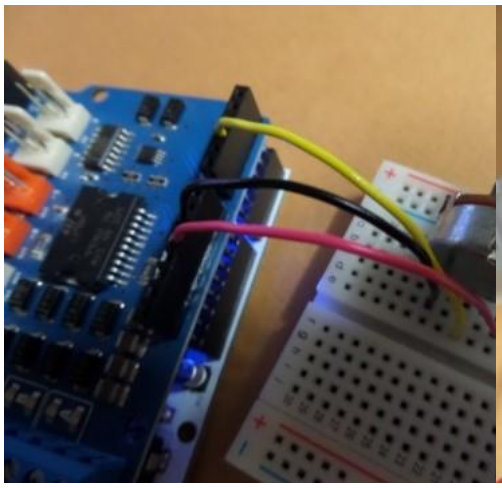
서보모터 연결

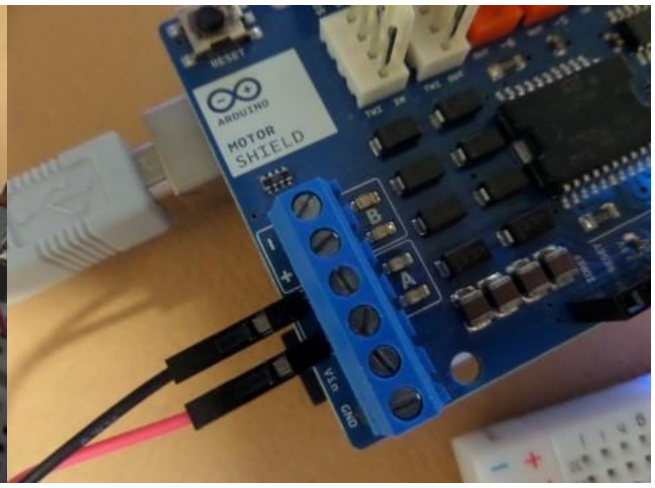


모터쉴드와 포텐셔미터로 DC 모터 제어하기

포텐셔미터의 양쪽 핀에 5V 전원과 접지(GND)를 연결한다. 그리고 중간핀을 아두이노 아날로그핀 A2 에 연결한다.





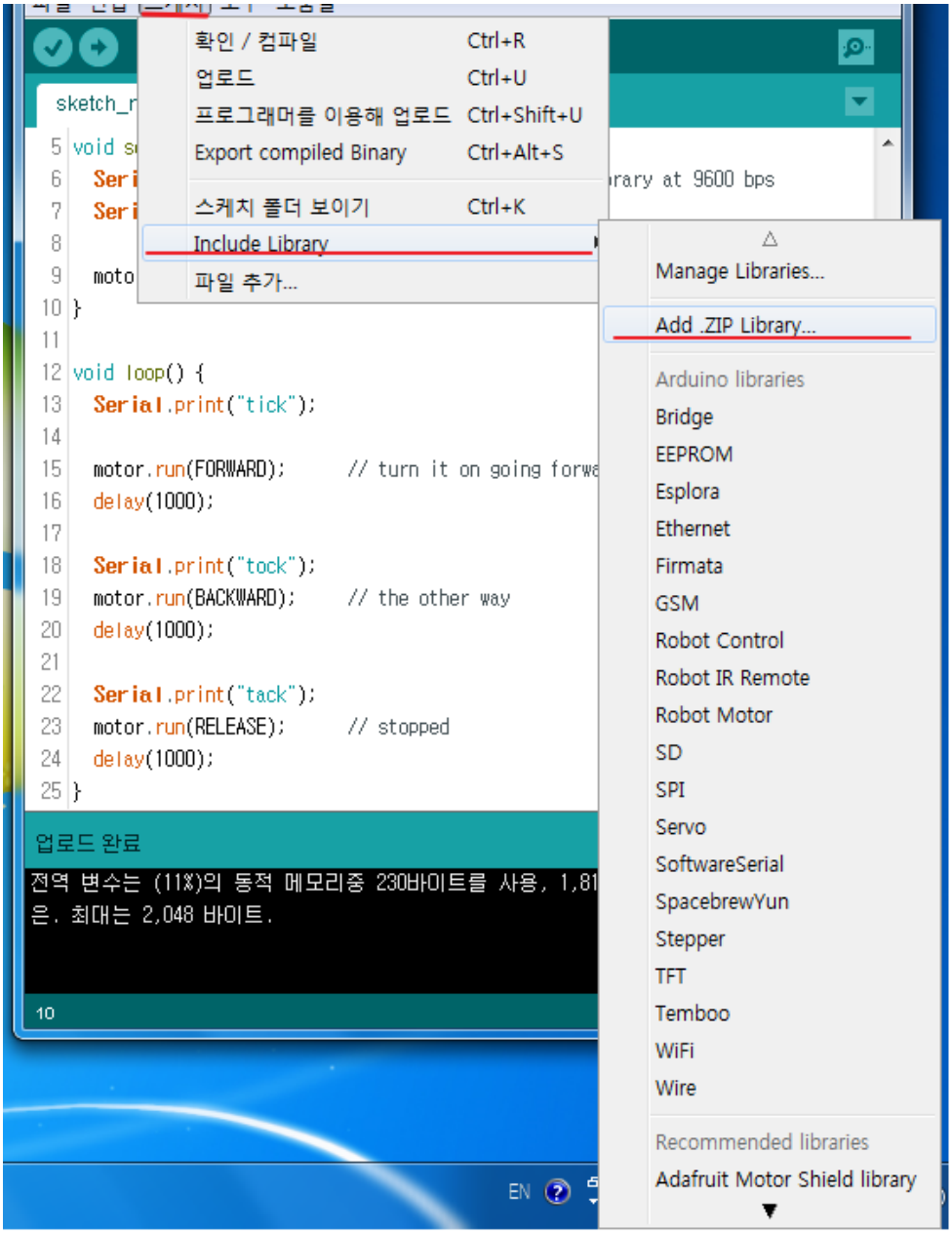



```
int dc_direction = 12;
int dc_pwm = 3;
int potMeter = 0;
```

```
int moTorVal = 0;
void setup() {
  pinMode(dc_direction, OUTPUT);
  pinMode(dc_pwm, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  potMeter = analogRead(A2);
  moTorVal = map(potMeter, 0, 1023, 0, 255);

  Serial.print(potMeter);
  Serial.print(" ¦ ");
  Serial.println(moTorVal);

  digitalWrite(dc_direction, HIGH);
  analogWrite(dc_pwm, moTorVal);
  delay(100);
}
```

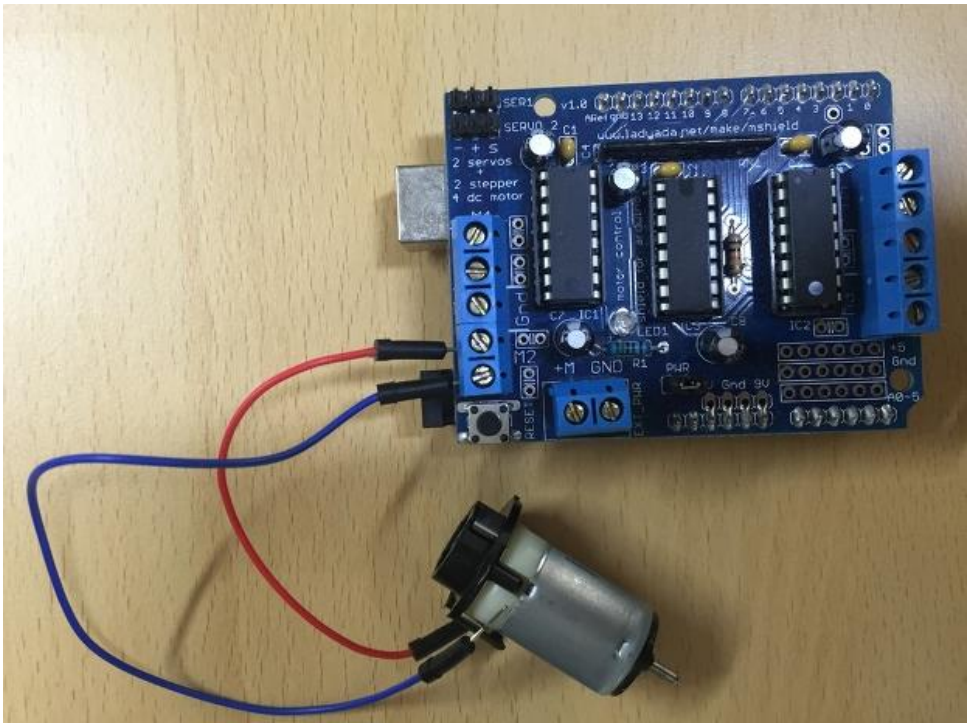
[라이브러리 다운받는 곳] - ZIP파일 다운로드
https://github.com/adafruit/Adafruit-Motor-Shield-library

[라이브러리 추가하는 방법] - 다운로드 한 곳을 확인하여 ZIP 파일 라이브러리 추가



```
#include <AFMotor.h>

AF_DCMotor motor(2, MOTOR12_64KHZ); // create motor #2, 64KHz pwm

void setup() {
  Serial.begin(9600);         // set up Serial library at 9600 bps
  Serial.println("Motor test!");

  motor.setSpeed(200);    // set the speed to 200/255
}

void loop() {
  Serial.print("tick");

  motor.run(FORWARD);     // turn it on going forward
  delay(1000);

  Serial.print("tock");
  motor.run(BACKWARD);    // the other way
  delay(1000);

  Serial.print("tack");
  motor.run(RELEASE);     // stopped
  delay(1000);
}
```

실행하면 시계방향, 반시계방향으로 왔다갔다 합니다.


SER1
v1.0
SERVO 2
C1
2 servos
2 stepper
4 dc motor
M4
Gnd
M2
RESET
motor control
C7
IC1
LED1
C8
IC2
+M
GND
R1
PWR
Gnd 9V
+5
Gnd
A0-5

analogWrite(IN1, 127); analogWrite(IN2, 0); // CW 속도 127

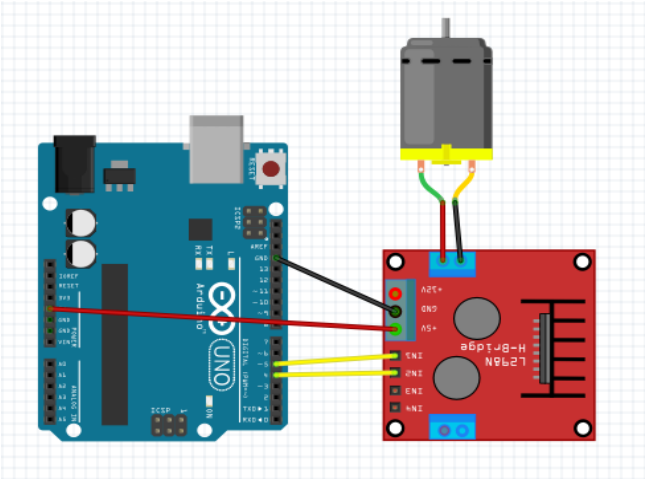
analogWrite(IN1, 0); analogWrite(IN2, 127); // CCW 속도 127

nalogWrite(IN1, 255); analogWrite(IN2, 0); // CW 속도 255

analogWrite(IN1, 0); analogWrite(IN2, 255); // CCW 속도 255

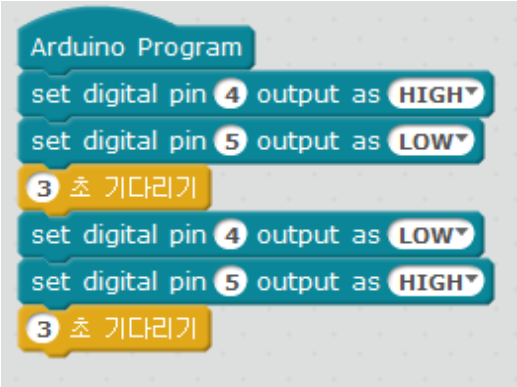
analogWrite(IN1, 0); analogWrite(IN2, 0); // 멈춤

digitalWrite(IN1, 0); digitalWrite(IN2, 0); // 급정거



```
int IN1=3;
int IN2=5;

void setup() {
  // Motor Module L298N
Serial.begin(9600);
```

```
 pinMode(IN1, OUTPUT);   // Direction  Control
 pinMode(IN2, OUTPUT);   // Speed Control
}

void loop() {
analogWrite(IN1, 127);    analogWrite(IN2, 0);
Serial.println("CW :speed=127");
delay(2000);

analogWrite(IN1, 0); analogWrite(IN2, 127);
Serial.println("CCW :speed=127");
delay(2000);

 analogWrite(IN1, 255);  analogWrite(IN2, 0);
 Serial.println("CW :speed=255");
 delay(2000);

 analogWrite(IN1, 0);   analogWrite(IN2, 255);
 Serial.println("CCW :speed=255");
 delay(2000);

digitalWrite(IN1, HIGH); digitalWrite(IN2,HIGH);  // stop
Serial.println("STOP");
delay(3000);
```



정방향과 역방향 설정

```
void setup() {
// Motor Module L298N
 pinMode(5,OUTPUT);
 pinMode(4,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(4,HIGH);
digitalWrite(5,LOW);
delay(3000);
digitalWrite(4,LOW);
digitalWrite(5,HIGH);
delay(3000);
}
```

정방향과 역방향, 모터 속도 제어 설정

```
void setup() {
// Motor Shield L298
Serial.begin(9600);
 pinMode(4, OUTPUT);   // Direction  Control
 pinMode(5, OUTPUT);   // Speed Control
}

void loop() {
  // put your main code here, to run repeatedly:
 digitalWrite(4, LOW);  // CW
 analogWrite(5, 100);
 Serial.println("CW :speed=100");
 delay(3000);
 analogWrite(5, 255);
 Serial.println("CW :speed=255");
 delay(3000);
 analogWrite(5, 0);
 Serial.println("CW :speed=0");
 delay(5000);

 digitalWrite(4, HIGH);  //CCW
 delay(3000);
 analogWrite(5, 100);
 Serial.println("CCW :speed=100");
 delay(3000);
 analogWrite(5, 255);
 Serial.println("CCW :speed=255");
 delay(3000);
 analogWrite(5, 0);
 Serial.println("CCW :speed=0");
 delay(5000);
}
```

아두이노에 외부전원 주는방법은 아두이노 보드 보시면 vin 핀과 gnd 핀이 있습니다.

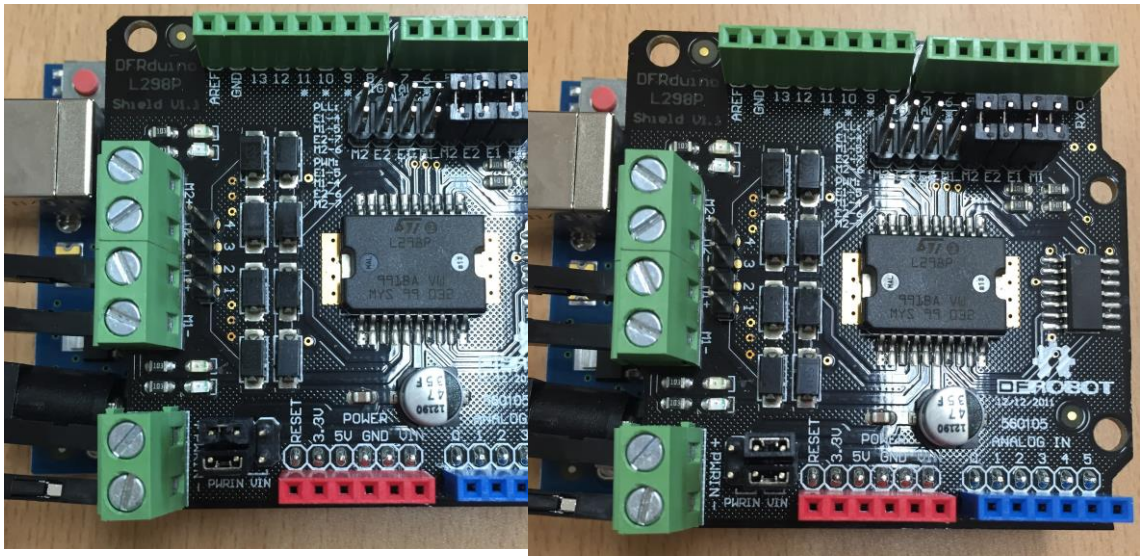
vin에다가 외부전원 +를 꽂으시고

gnd에다가 외부전원 -를 꽂으시면 됩니다.

모터 쉴드 외부 전원 사용법

외부 전원 핀에 연결하고 외부 전원을 사용함을 알려줘여 한다. 점퍼를 바꾸는데 그림과 같이 점퍼가 VIN에 연결되어 있는 것을 PWRIN 으로 바꾸어야 한다.

외부 전원 사용 아두이노 전원 사용


DFRduino
L298P
Shield V1.1
DFROBOT
POWER
ANALOG IN

```
void setup() {
  // Motor Shield L298
Serial.begin(9600);
 pinMode(4, OUTPUT);   // Direction  Control
 pinMode(5, OUTPUT);   // Speed Control
}

void loop() {
  // put your main code here, to run repeatedly:
 digitalWrite(4, LOW);  // CW
 analogWrite(5, 100);
 Serial.println("CW :speed=100");
 delay(3000);
 analogWrite(5, 255);
 Serial.println("CW :speed=255");
 delay(3000);
 analogWrite(5, 0);
 Serial.println("CW :speed=0");
 delay(5000);

 digitalWrite(4, HIGH);  //CCW
 delay(3000);
 analogWrite(5, 100);
 Serial.println("CCW :speed=100");
 delay(3000);
 analogWrite(5, 255);
 Serial.println("CCW :speed=255");
 delay(3000);
 analogWrite(5, 0);
 Serial.println("CCW :speed=0");
 delay(5000);
}
```

```
void setup() {
  // Motor Shield L298
Serial.begin(9600);
 pinMode(4, OUTPUT);   // Direction  Control
 pinMode(5, OUTPUT);   // Speed Control
}

void loop() {
  // put your main code here, to run repeatedly:
 digitalWrite(4, LOW);  // CW
 analogWrite(5, 100);
 Serial.println("CW :speed=100");
 delay(3000);

 analogWrite(5, 255);
 Serial.println("CW :speed=255");
 delay(3000);
 analogWrite(5, 0);
 Serial.println("CW :speed=0");
 delay(5000);

 digitalWrite(4, HIGH);  //CCW
 delay(3000);
 analogWrite(5, 100);
 Serial.println("CCW :speed=100");
 delay(3000);
 analogWrite(5, 255);
 Serial.println("CCW :speed=255");
 delay(3000);
 analogWrite(5, 0);
 Serial.println("CCW :speed=0");
 delay(5000);
}
```

Chapter 5. 아두이노 한 걸음 더

1. 시간 함수

【1】millis()

아두이노 보드가 현재 프로그램을 실행한 이후의 시간을 밀리초로 반환한다. 프로그램이 약 50 일이 지나면 오버플로워가 발생하여 0 으로 돌아갑니다.

【형식】

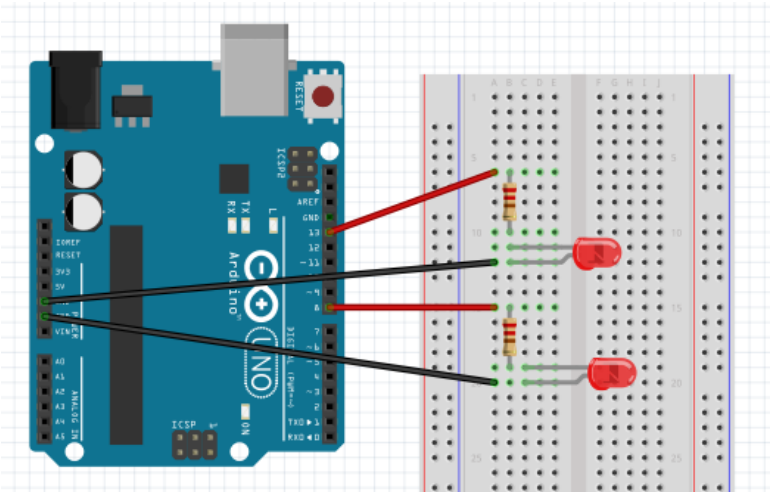
```
millis();
```

반환 값의 자료형은 unsigned long입니다.

【소스 프로그램】

```
unsigned long time;

void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Time: ");
  time = millis();
  //prints time since program started
  Serial.println(time);
  // wait a second so as not to send massive amounts of data
  delay(1000);
}
```

이 함수를 사용하여 2 개의 LED를 독립적으로 깜박거리게 만들어 보자.



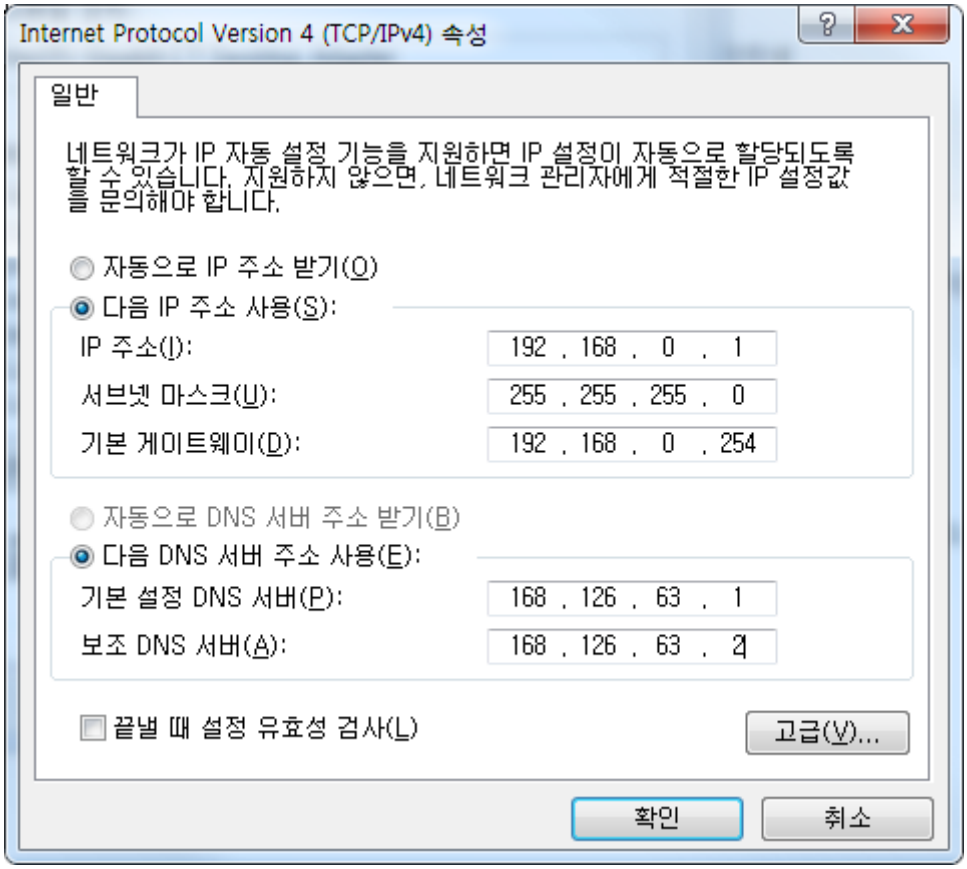
```
const int LedPin01=13;
int LedOnOff01=LOW;
unsigned long preMillils01=0;
const long interval01=1013;

const int LedPin02=8;
int LedOnOff02=LOW;
unsigned long preMillils02=0;
const long interval02=509;

void setup() {
  pinMode(LedPin01, OUTPUT);
  pinMode(LedPin02, OUTPUT);
}

void loop() {
  unsigned long curMillils=millis();

  if( curMillils - preMillils01 > interval01){
    LedOnOff01=  (LedOnOff01==LOW)? HIGH: LOW;
    digitalWrite(LedPin01, LedOnOff01);
    preMillils01=curMillils;
  }
```

```
  if( curMillils - preMillils02 > interval02){
    LedOnOff02=  (LedOnOff02==LOW)? HIGH: LOW;
    digitalWrite(LedPin02, LedOnOff02);
    preMillils02=curMillils;
  }

}
```

실제 사진

2. 이더넷 쉴드

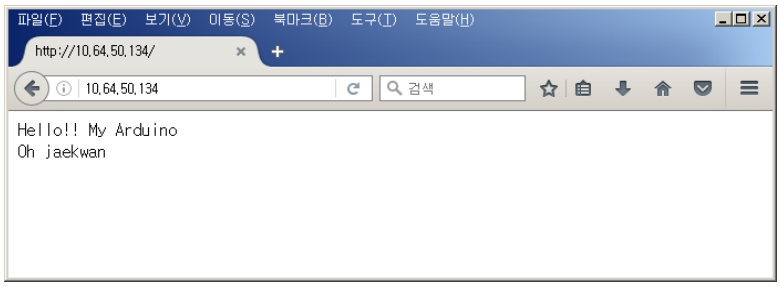
네트워크



http 프로토콜

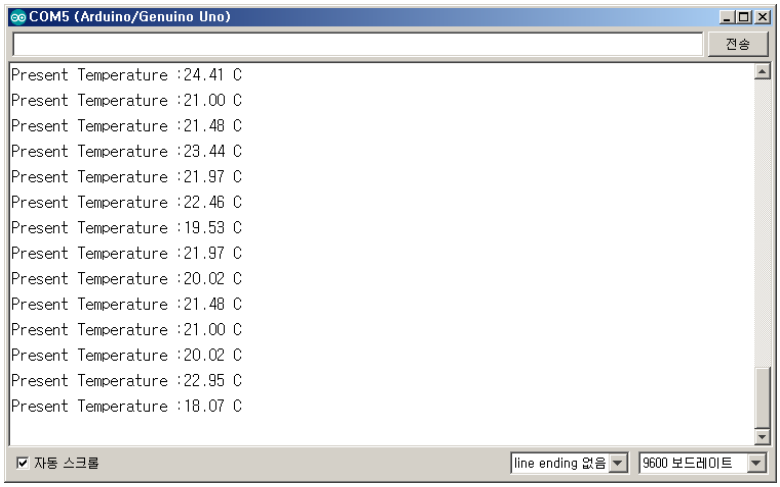
ftp

port




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

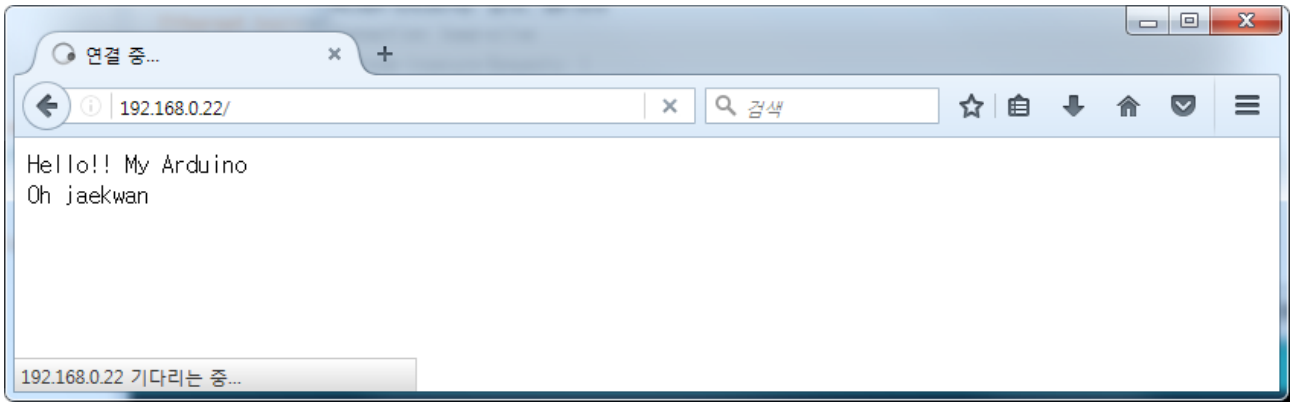
```
#include <Ethernet.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; //mac 주소 16 진수
입력
IPAddress ip(10,64,50,134);                         //로컬 네트워크에서
유일한 IP 를 사용해야 합니다.
IPAddress gateway( 10, 64, 50, 254 );
IPAddress subnet( 255, 255, 255, 0 );
IPAddress myDns(168,126, 63, 1);                    // DNS Server IP
Address

EthernetServer server(80); //80 은 포트번호



void setup() {
  Ethernet.begin(mac, ip, myDns, gateway, subnet);  //이더넷 디바이스
초기화
  server.begin();  //서버 연결 시작
 }

void loop() {
  EthernetClient client= server.available();
  client.println("Hello!! My Arduino");
  client.println("Oh jaekwan");
  client.stop();
}
```

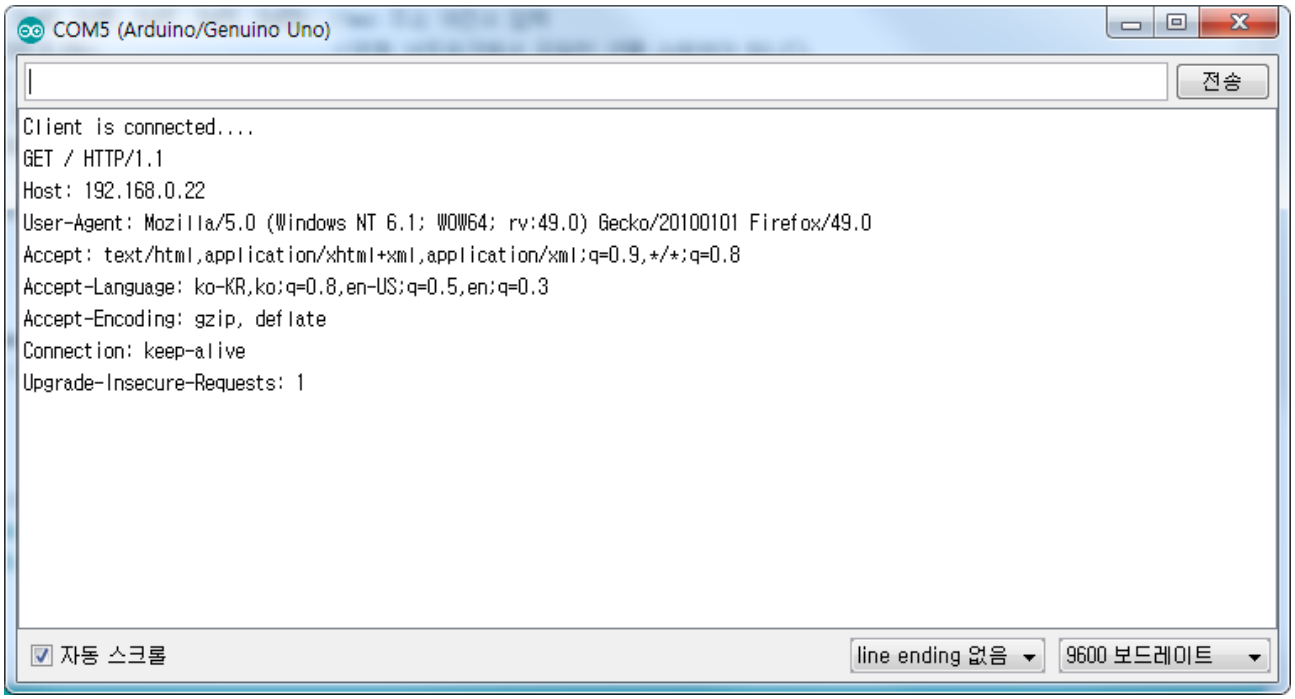



```
#include <SPI.h>
```

```
#include <Ethernet.h>
#define LM35 A3

int reading = 0;
float temperature = 0;

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; //mac 주소 16 진수
입력
IPAddress ip(10,64,50,135);                        //로컬 네트워크에서
유일한 IP 를 사용해야 합니다.
IPAddress gateway( 10, 64, 50, 254 );
IPAddress subnet( 255, 255, 255, 0 );
IPAddress myDns(168,126, 63, 1);                   // DNS Server IP
Address

EthernetServer server(80); //80 은 포트번호



void setup() {
  Ethernet.begin(mac, ip, myDns, gateway, subnet);  //이더넷 디바이스
초기화
  server.begin();  //서버 연결 시작
  Serial.begin(9600);
 }

void loop() {
  EthernetClient client= server.available();

  reading = analogRead(LM35);
  temperature = ( 5.0 *reading * 100.0) / 1024.0;
  Serial.print("Present Temperature :");
  Serial.print(temperature);
  Serial.println(" C");

  client.print("Present Temperature :");
  client.print(temperature);
  client.println(" C");
```

```
  delay(3000);
  client.stop();
}
```





서버에서 데이터를 읽어 오는 방법

GET 방식 : URL 끝에 data를 붙여서 가져옴(보임)

FORM 방식 : 메시지 방식으로 가져옴. (보이지 않음)

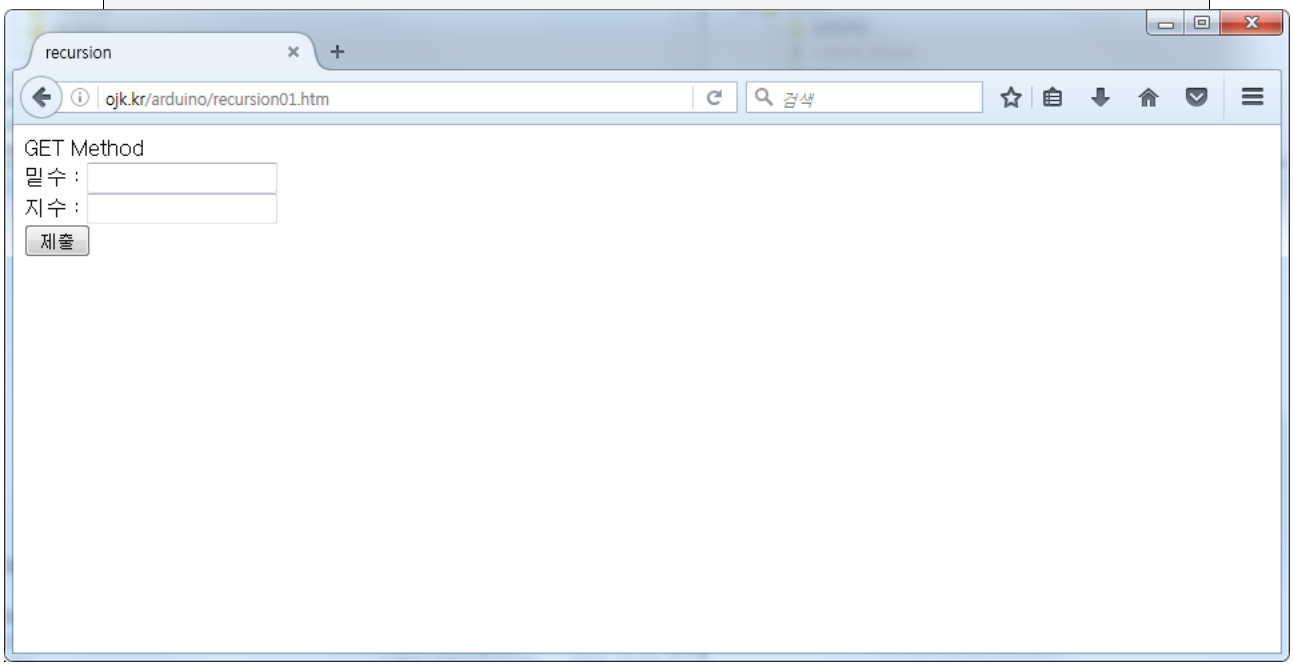
recursion01.htm

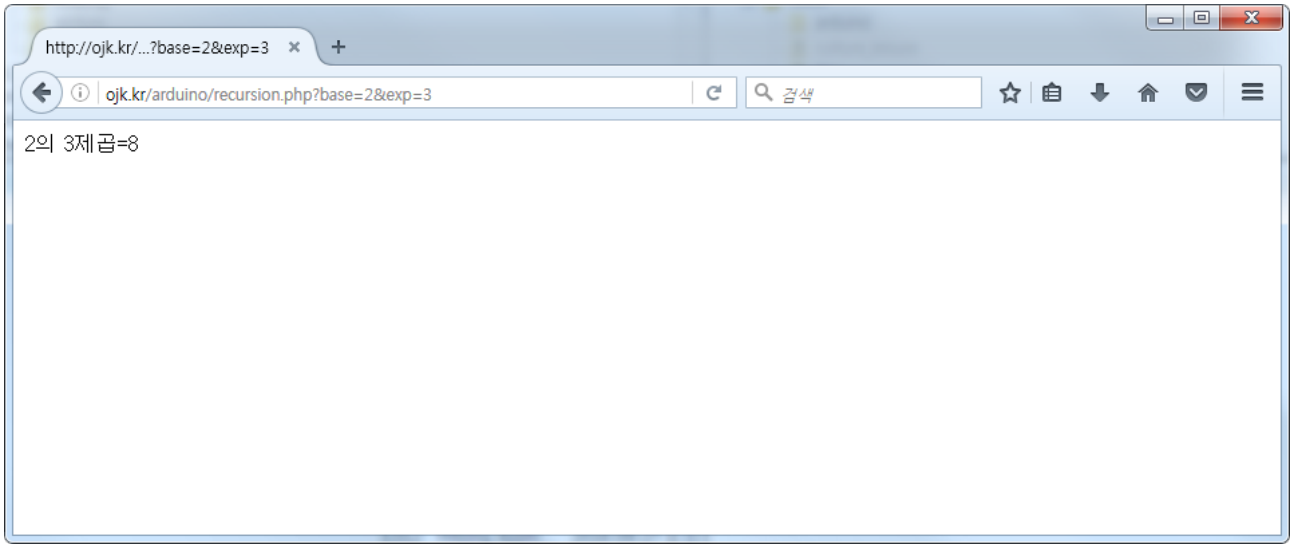
```
<html>
<head><title>recursion</title></head>
<body>
GET Method<br>
<form method="get" action="./recursion.php">

밑수 : <input type="text" name="base"><br>
지수 : <input type="text" name="exp"><br>

<input type="submit" value="제출"><br>
</form>
</body>
</html>
```

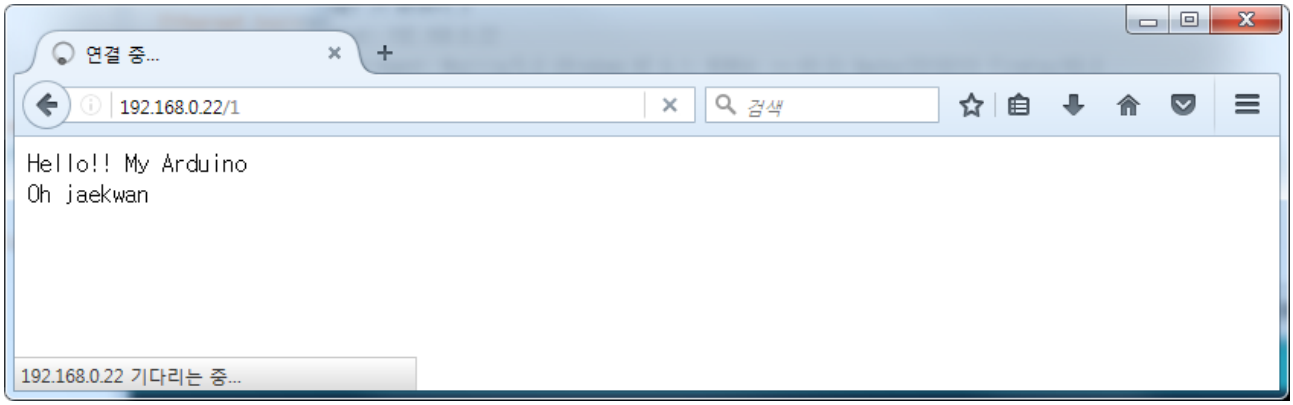
recursion.php

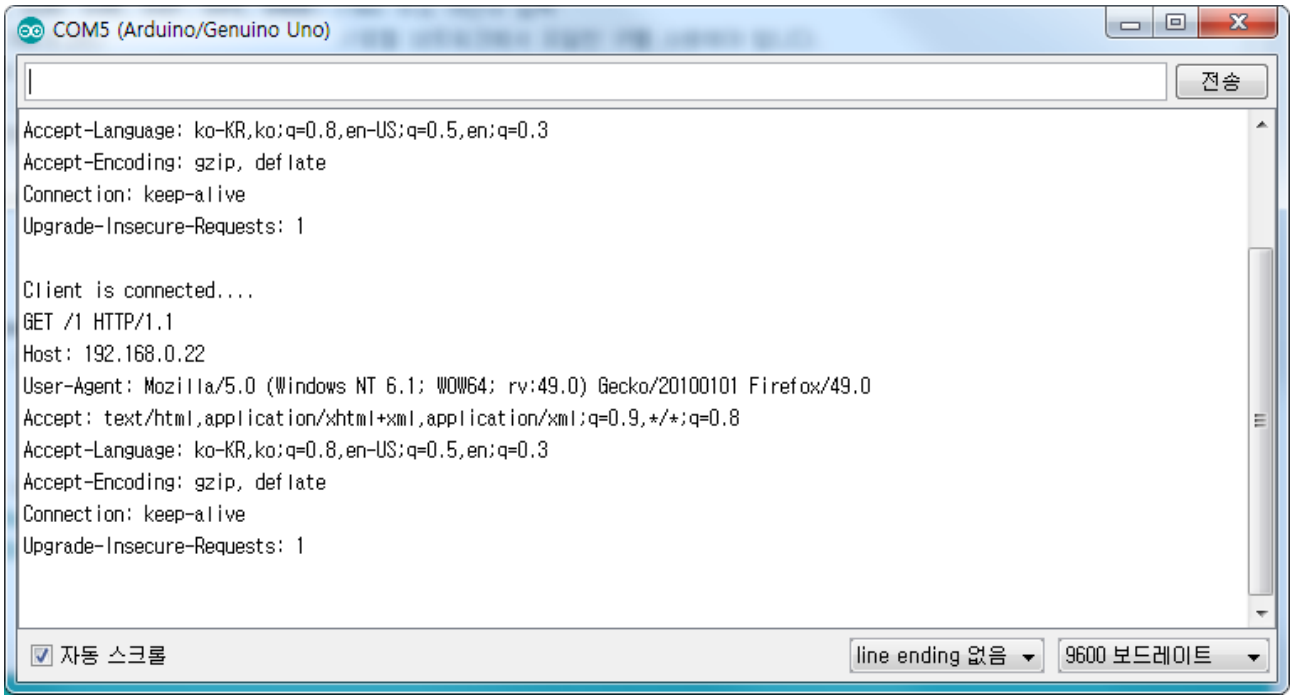
```
<?
function power($base,$exp){
  if($exp){
       return $base*power($base,$exp-1);
  }

  return 1;
```

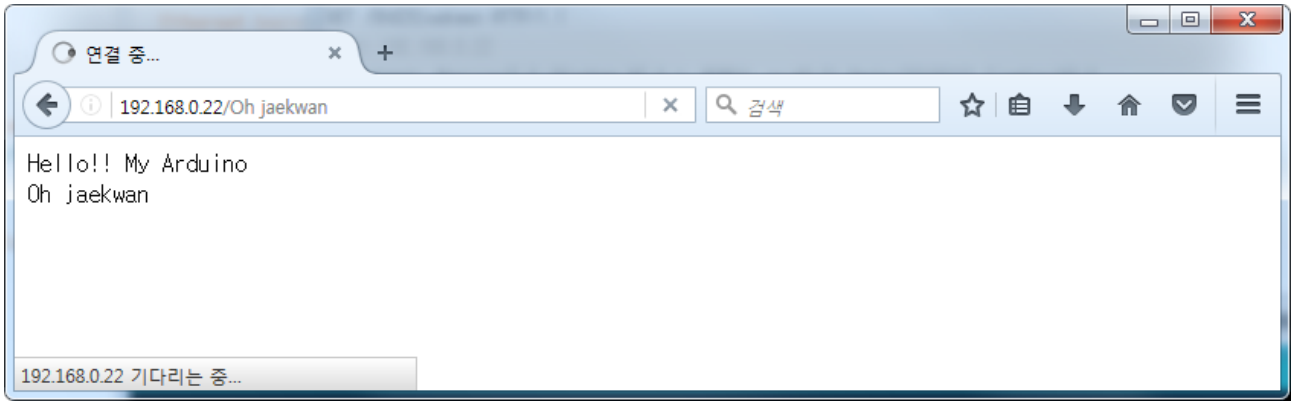
```
}

echo($base ."의 " . $exp ."제곱=");
echo(power($base,$exp));
?>
```

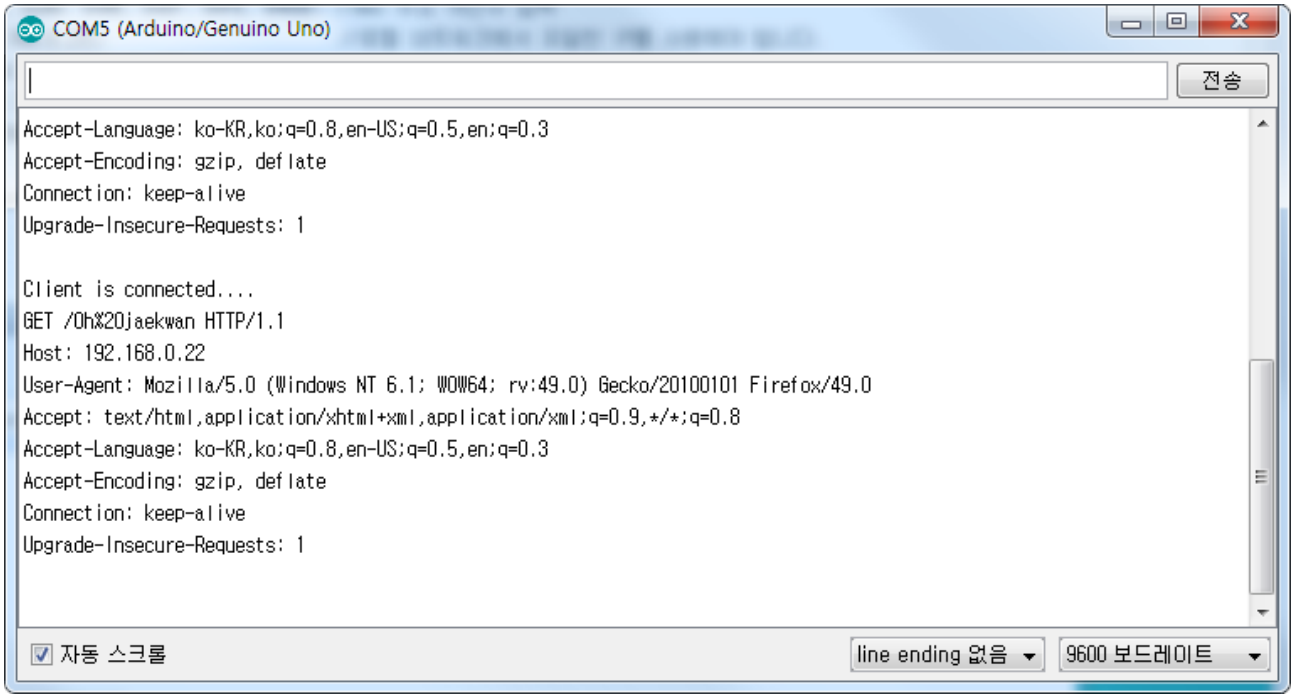



http://ojk.kr/...?base=2&exp=3
ojk.kr/arduino/recursion.php?base=2&exp=3
검색
2의 3제곱=8


연결 중...
192.168.0.22/1
검색
Hello!! My Arduino
Oh jaekwan
192.168.0.22 기다리는 중...


COM5 (Arduino/Genuino Uno)
전송
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Client is connected....
GET /1 HTTP/1.1
Host: 192.168.0.22
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
자동 스크롤
line ending 없음
9600 보드레이트


연결 중...
192.168.0.22/Oh jaekwan
검색
Hello!! My Arduino
Oh jaekwan
192.168.0.22 기다리는 중...


COM5 (Arduino/Genuino Uno)
전송
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Client is connected....
GET /Oh%20jaekwan HTTP/1.1
Host: 192.168.0.22
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
자동 스크롤
line ending 없음
9600 보드레이트

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

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED } ; //mac address
IPAddress ip(192, 168, 0, 22) ;
EthernetServer server(80) ;

char readChar, controlVal;



void setup(){
 pinMode(2, OUTPUT);
 Ethernet.begin(mac, ip);
 server.begin();
 Serial.begin(9600);
}

void loop(){
 EthernetClient client=server.available();
 client.println("Welcome Arduino Web Server");

 if(client) {
  if(client.connected()) Serial.println("Client is connected....");
```

```
  while ( client.connected()){
    if(client.available()){
      readChar = client.read();
      Serial.print(readChar);
      if (readChar == '$'){
        controlVal = client.read();
        if (controlVal == '1') {
          Serial.println("ON");
          digitalWrite(2, HIGH);
        }

        if (controlVal == '0') {
          Serial.println("OFF");
          digitalWrite(2, LOW);
        }
        delay(10);
      }  //if (readChar == '$')
    } //if(client.available())
  }  // while
  client.stop();   // close the connection:
 }  //if(client)
}   // void loop()
```

recursion02.htm

```
<html>
<head><title>recursion</title></head>
<body>
POST Methdod<br>
<form method="post" action="./recursion.php">

밑수 : <input type="text" name="base"><br>
지수 : <input type="text" name="exp"><br>
```

```
<input type="submit" value="제출"><br>
</form>
</body>
</html>
```

각자 실습 : 1.php 1_get.htm 1_form.htm

주소 : ojk.kr id :ojk0527 password : alex1225

arduino 폴더에 업로드

3. 초음파 센서

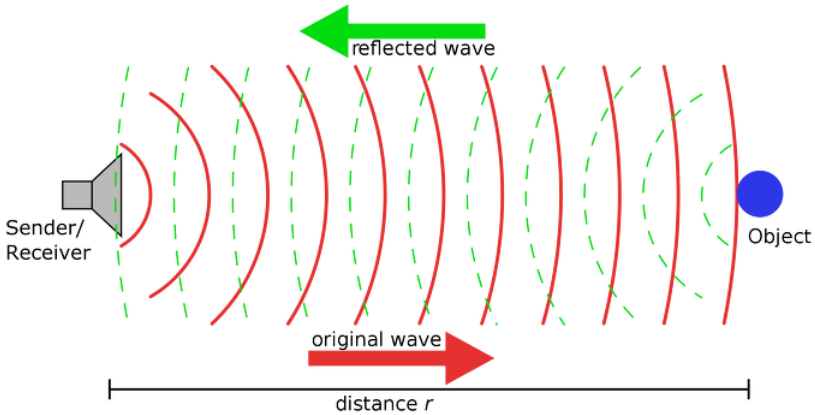
【알게되는 것들】

- 일차식을 사용하여 실생활 문제를 해결할 수 있다.
- 음파의 속도를 이용하여 거리를 계산할 수 있다.
- 초음파 센서를 거리 계산에 이용할 수 있다.

【1】초음파 센서란

초음파 센서란 초음파를 쏘아서 물체에 반사되어 되돌아 오는 시간을 계산하는 센서입니다. 이를 이용하면 음파의 속도와 되돌아 오는 시간을 이용하여 물체까지의 거리를 알아낼 수 있습니다. 초음파 센서의 종류에는 여러가지가 있지만 많이 사용하는 센서가 다음 그림과 같은 HC-R04 입니다. 이 센서는 측정거리 : 2cm ~ 5m, 측정각도 : 15', 측정 해상도 : 3mm 의 성능을 갖습니다.

아래 그림에서와 같이 Trig 에서 초음파을 쏘아서(Trig 에 전원을 넣어서) Echo에 되돌아 오는 것을 감지하여(Echo에 전원이 감지된다) Trig 시간에서 Echo 시간을 빼서 되돌아 오는 시간을 계산합니다.



음속이 340m/s 정도 되니까 센서를 통해 응답이 오는 시간만 알면 초음파 센서 앞에 있는 사물까지

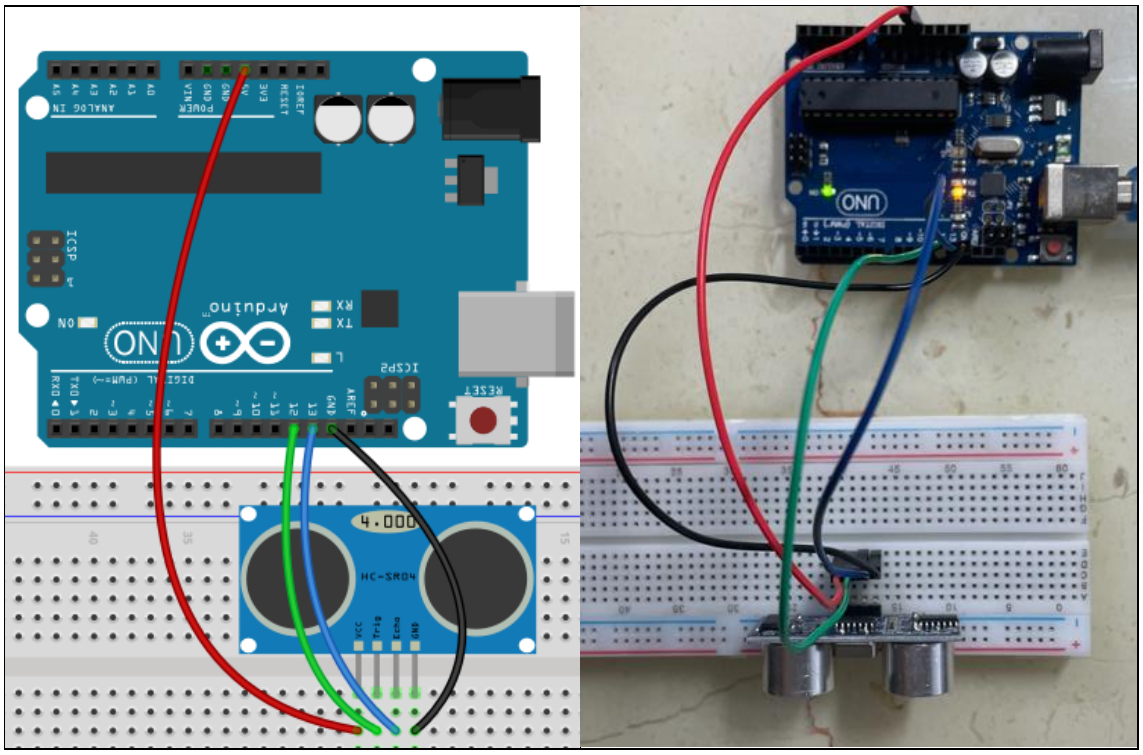
의 거리를 잴 수 있습니다.

【2】 연결방법

초음파 센서를 보시면 4 개의 핀이 있습니다. VCC, Trig, Echo, GND 인데, 아두이노에 다음과 같이 연결합니다.

초음파 센서	아두이노
VCC	5V
Trig	D12
Echo	D13
GND	GND

Trig(트리거) 핀으로 ON을 입력하면 초음파 모듈에서 40KHz 음파를 발사합니다.(10us 이상 HIGH 유지를 권장) 이때부터 Echo 핀은 High 상태가 되고, 음파가 되돌아와 수신되면 echo 핀이 다시 Low 상태가 됩니다.



【3】거리 계산 방법

$$\text{거리} = \text{속력} \times \text{시간}$$

보통 소리는 1 초에 340 미터를 이동합니다. 즉, 음속 = 340 m/sec . 아두이노에서는 시간을 초단위가 아니라 미리초 단위를 사용하므로 미리초 단위로 고침니다.

$$\text{음속} = 340m/sec = 34000cm/sec = \frac{34000}{1000} cm / \frac{1}{1000} sec$$

$$= 34cm / \frac{1}{1000} sec = 34cm/ms$$

$$= \frac{34}{1000} cm / \frac{1}{1000} ms = \frac{34}{1000} cm/\mu s = \frac{340}{10000} cm/\mu s$$

$$\text{초음파의 속력} = \frac{340}{10000} cm/\mu s$$

$$\text{초음파 왕복거리} = \text{초음파의 속력} \times \text{시간} = \frac{340}{10000} cm/\mu s \times time$$

$$\text{물체와의 거리} = \frac{\text{초음파의 왕복거리}}{2}$$

$$= \frac{340}{10000} cm/\mu s \times time \times \frac{1}{2}$$

$$= \frac{340}{10000} \times \frac{1}{2} \times time\ cm/\mu s$$

따라서 아두이노에서는 초음파를 이용한 거리 계산은 다음 식 중 하나를 사용합니다. 물론 단위는 $cm/\mu s$ 입니다.

$$distance = \frac{340}{10000} \times time \times \frac{1}{2}$$

$$distance = 0.017 \times time$$

첫 번째 식은 계산을 한 번 더 하는 식이므로 두 번째 식이 계산 상으로는 간단하므로 0.017 이라는

숫자가 어떻게 나온 수인지 궁금할 수 있으므로 첫 번째 식을 사용하는 것이 바람직합니다. 초음파 센서를 사용하여 거리를 측정하여 봅시다. time은 어떻게 계산할까요?

trig에 계속적으로 펄스를 주면, 되돌아 오는 시간이 언제 발사한 신호가 들어오는지 모르게 때문에 펄스를 한 번 주고 되돌아 오는 펄스를 감지하여야 합니다.

- trig 에 0.01 초 동안 펄스를 줌(전기를 ON)
- trig 를 OFF 합니다.
- Echo 에 되돌아 온 신호가 들어오기를 기다립니다.
- Echo 에 신호가 도착하면 그 시간차를 빼어서 시간을 계산
- 앞의 설명한 식에 의해서 거리를 계산합니다.

【4】 거리 계산과 출력

【Sketch】

【형식】

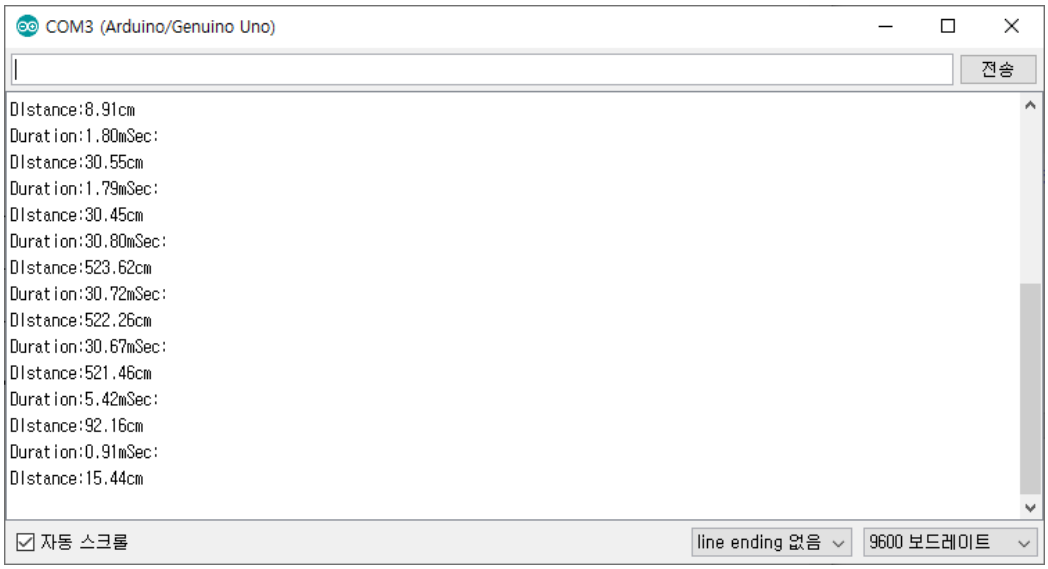
```
digitalWrite(trig, HIGH);
delay(10);
digitalWrite(trig, LOW);
duration = pulseIn(echoPin, HIGH);
```

【소스 프로그램】

```
//File Name:ex031_supersonnic01.ino
const int Trig = 12; //Trig pin
const int Echo = 13; //Echo pin
long duration = 0;
double distance;


void setup() {
  pinMode(Trig, OUTPUT); // Trigger is an output pin
```

```
  pinMode(Echo, INPUT); // Echo is an input pin
  Serial.begin(9600); // Serial Output
}

void loop() {
  digitalWrite(Trig, LOW);
  delay(10);
  digitalWrite(Trig, HIGH); // Trigger pin to HIGH
  delay(10);                //  10us high
  digitalWrite(Trig, LOW);  // Trigger pin to LOW
  duration = pulseIn(Echo, HIGH);
  distance = (340.0 / 10000.0 * duration) / 2.0;
  Serial.print("Duration:");
  Serial.print(duration/1000.0);
  Serial.println("mSec:");
  Serial.print("DIstance:");
  Serial.print(distance);
  Serial.println("cm");
  delay(1000);
}
```

【실행 결과】



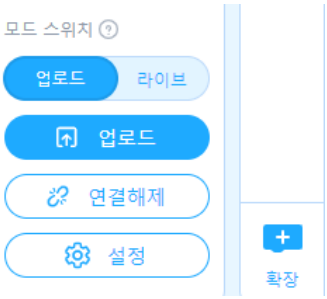
【mblock】

초음파 센서 이용을 위하여 Trig에서 신호을 보내고 이를 Echo로 받아서 거리를 계산하는 모든 작업을 엠블록에서는 

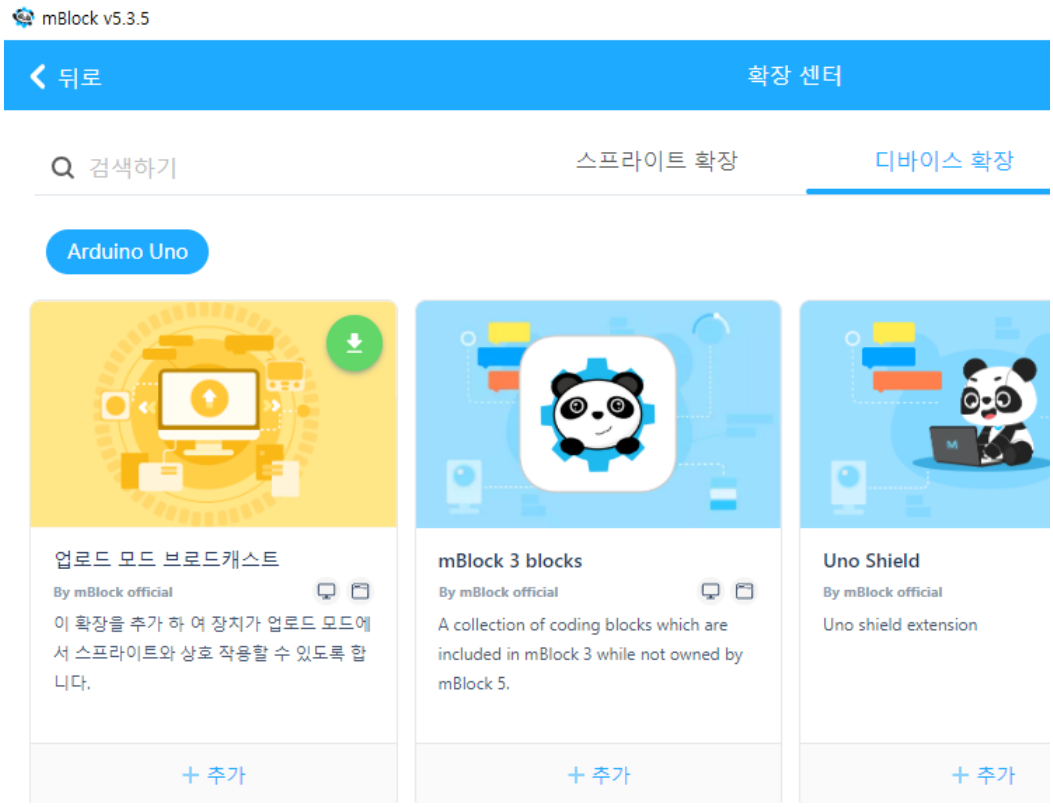
블록하나로 해줍니다. 따로 계산을 위한 블록을 작성할 필요가 없습니다. 따라서 이 블록으로 얻은 값을 컴퓨터에 전송하여 그 값을 처리하면 됩니다.

아두이노에서 전송되어 오는 값을 컴퓨터(MBlock)에서 수신 받으려면 블록을 확장하여야 합니다. 오른 쪽 그림의 

을 클릭하고

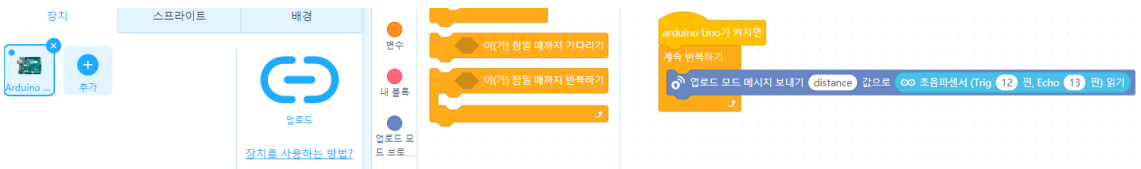


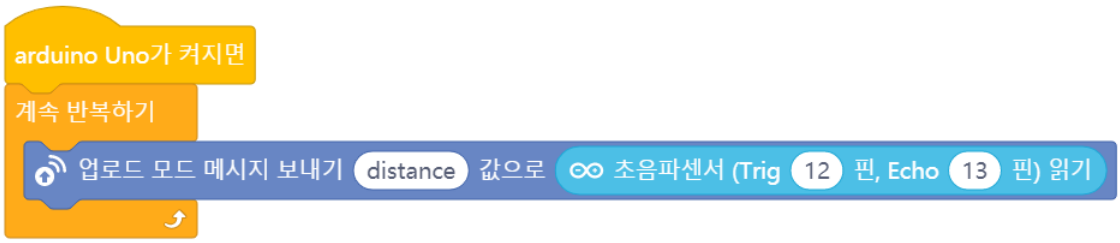
업로드 모드 브로드캐스트를 추가합니다. 이 블록은 스크래치에서 사용하는 "방송하기"와 같은 블록입니다. 즉, 아두이노에서 측정한 값을 컴퓨터로 방송하여(브로드캐스트) 컴퓨터에서는 이를 수신하여 말하기 블록으로 설정하면 아두이노에서 측정한 값을 컴퓨터에 표시할 수 있습니다.



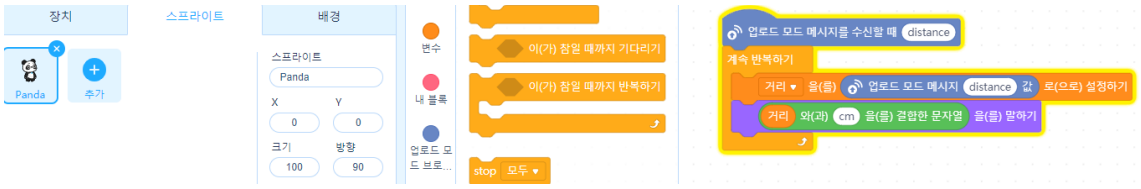
"업로드 모드 브로드캐스트"를 추가하면 다음과 같이 블록이 변경됩니다.

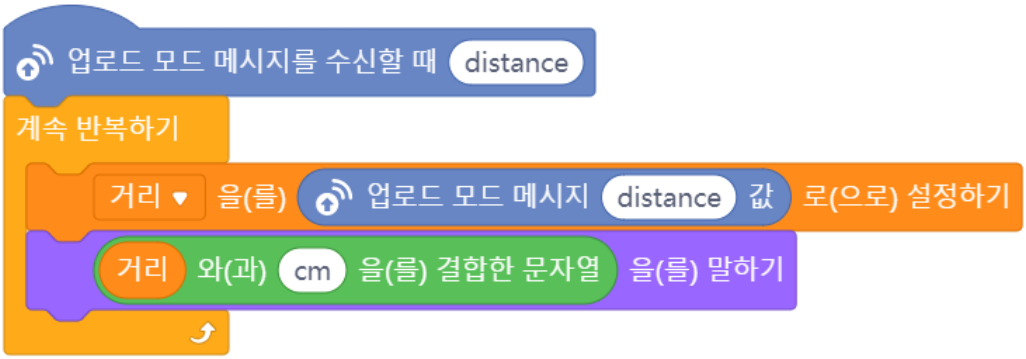
이제 아두이노에서 값을 엠블록에 보내면 됩니다. 그림처럼【장치】에서 블록을 추가합니다.



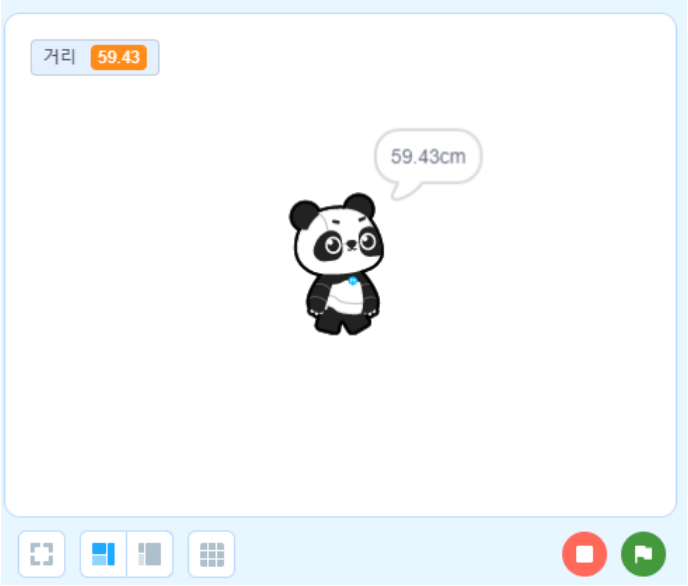


이제 컴퓨터에서 전송받은 값을 처리하여야 합니다. 지금 하고자 하는 것은 거리를 출력하여야 하므로 다음 블록을 【스프라이트】에 추가합니다.





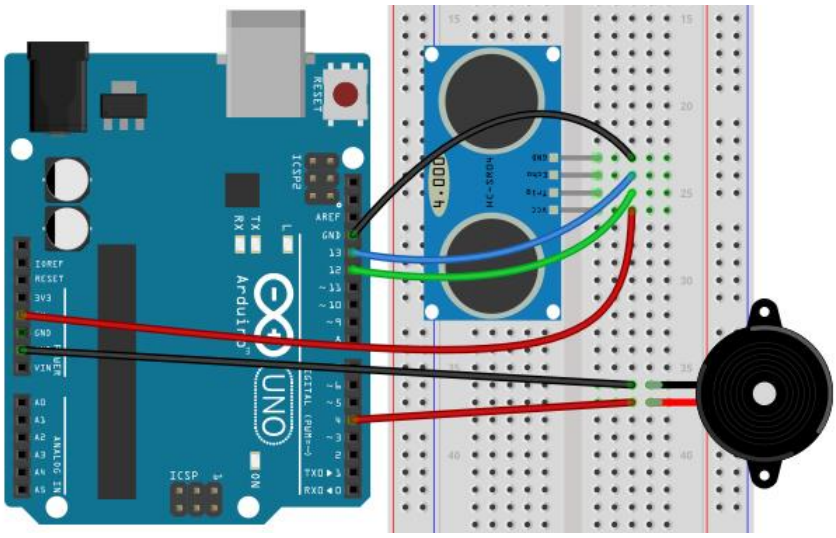
다음은 거리를 출력한 화면입니다. 1 초에 한 번씩 값을 출력하고 싶겠지만 좀처럼 되지 않습니다. 좀 더 세세한 출력 설정을 원한다면 C언어로 프로그래밍하여야 합니다. 엠블록은 훌륭한 프로그래밍 언어이기는 하지만 교육용 프로그래밍 언어이므로 한계가 있습니다.

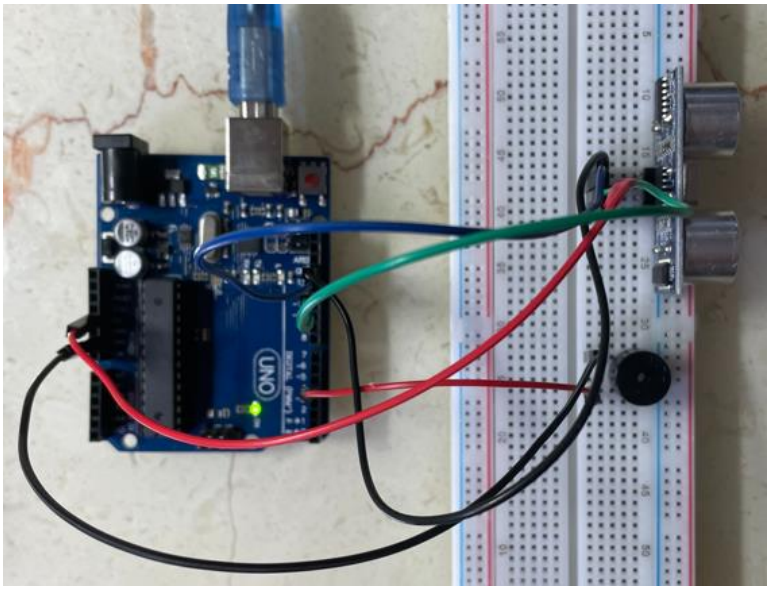


【5】후방 감지기

자동차 후진을 하면 뒤에 있는 물체를 감지하여 소리를 출력해 주는 후방감지기를 만들어 봅시다. 초음파 센서를 이용하여 거리를 측정한 다음에 거리가 50cm 미만이면 1 초 간격으로 부저가 울리고 거리가 화면에 출력이 되도록 만들어 봅시다.

.

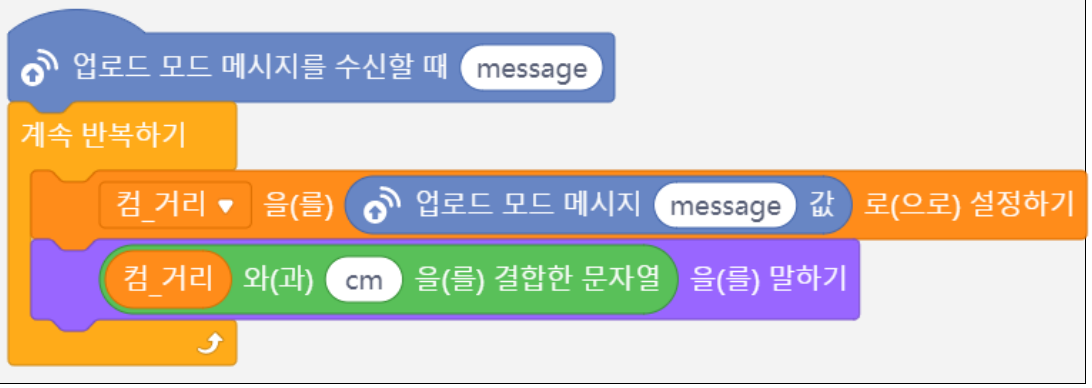




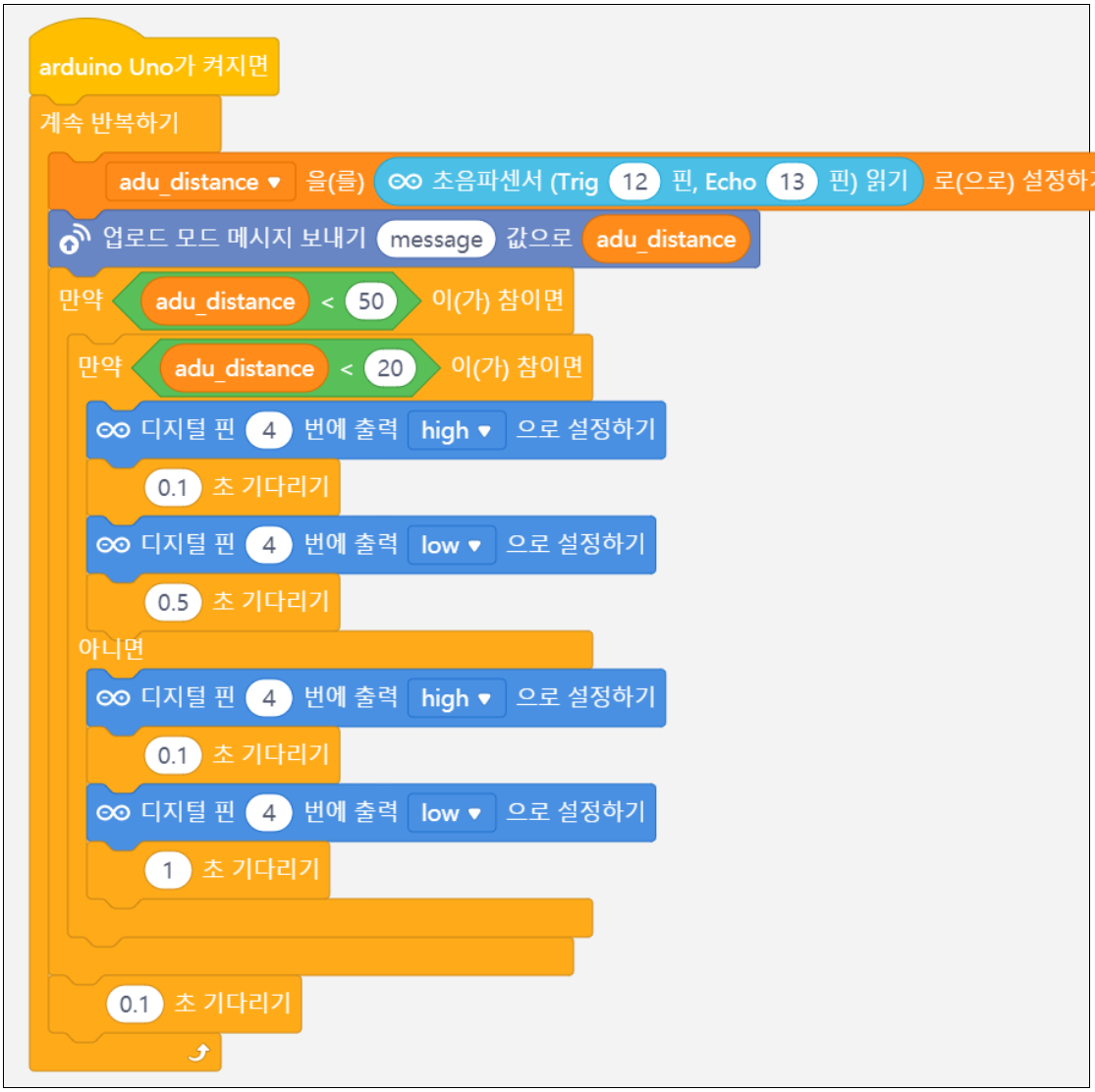
【코드 블록】

【 장치 】

【 스프라이트 】



거리가 50cm 미만이면 1 초마다 부저가 울리고 거리가 20cm미만이면 0.5 초마다 부저가 울리도록 만들어 봅시다.


arduino Uno가 켜지면
계속 반복하기
adu_distance ▾ 을(를) 초음파센서 (Trig 12 핀, Echo 13 핀) 읽기 로(으로) 설정하기
업로드 모드 메시지 보내기 message 값으로 adu_distance
만약 adu_distance < 50 이(가) 참이면
만약 adu_distance < 20 이(가) 참이면
디지털 핀 4 번에 출력 high ▾ 으로 설정하기
0.1 초 기다리기
디지털 핀 4 번에 출력 low ▾ 으로 설정하기
0.5 초 기다리기
아니면
디지털 핀 4 번에 출력 high ▾ 으로 설정하기
0.1 초 기다리기
디지털 핀 4 번에 출력 low ▾ 으로 설정하기
1 초 기다리기
0.1 초 기다리기

【Sketch】

측정 거리를 시리얼에 표시

```
// File Name : ex071_supersonic01.ino
int trigPin = 13;
int echoPin = 12;

void setup(){
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop(){
  float duration, distance;
  digitalWrite(trigPin, HIGH);
  delay(10);
  digitalWrite(trigPin, LOW);

 duration = pulseIn(echoPin, HIGH);
 distance = (340.0 / 10000.0 * duration) / 2.0;

 Serial.print("Duration:");
 Serial.print(duration/1000.0);
 Serial.println("mSec:");
 Serial.print("DIstance:");
 Serial.print(distance);
 Serial.println("cm");
 delay(1000);
}
```

```
// File Name : ex072_supersonic02.ino
int trigPin = 13;
int echoPin = 12;
int buzzer=4;

void setup(){
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(4 , OUTPUT);
}

void loop(){
  float duration, distance;
  digitalWrite(trigPin, HIGH);
  delay(10);
  digitalWrite(trigPin, LOW);

 duration = pulseIn(echoPin, HIGH);
 distance = (340.0 / 10000.0 * duration) / 2.0;

 if (distance < 20){
    digitalWrite(buzzer,HIGH);
    delay(50);
    digitalWrite(buzzer,LOW);
    delay(50);
 }

 Serial.print("Duration:");
 Serial.print(duration/1000.0);
 Serial.println("mSec:");
 Serial.print("DIstance:");
 Serial.print(distance);
 Serial.println("cm");
 delay(1000);
```

```
}
```

거리가 20cm 이내이면 부저가 울리고, 10cm 이내이면 부저가 급하게 울리는 프로그램

```
// File Name : ex073_supersonic03.ino
int trigPin = 13;
int echoPin = 12;
int buzzer=4;

void setup(){
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(4 , OUTPUT);
}

void loop(){
  float duration, distance;
  digitalWrite(trigPin, HIGH);
  delay(10);
  digitalWrite(trigPin, LOW);

 duration = pulseIn(echoPin, HIGH);
 distance = (340.0 / 10000.0 * duration) / 2.0;

 if (distance <= 10){
    //while(distance >= 10 ){
      digitalWrite(buzzer,HIGH);
      delay(100);
      digitalWrite(buzzer,LOW);
      delay(50);
   // }
  } else if (distance >10 && distance < 20){
    digitalWrite(buzzer,HIGH);
    delay(100);
    digitalWrite(buzzer,LOW);
    delay(500);
```

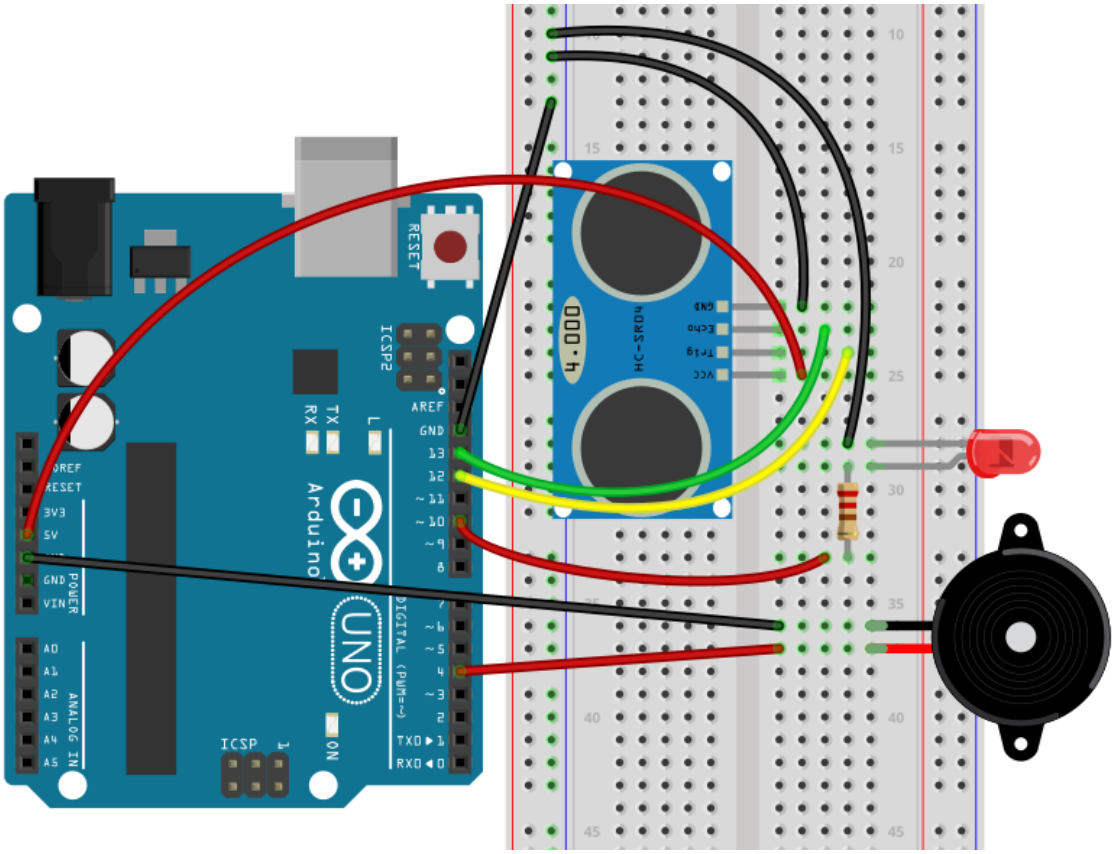
```
 }

 Serial.print("Duration:");
 Serial.print(duration/1000.0);
 Serial.println("mSec:");
 Serial.print("DIstance:");
 Serial.print(distance);
 Serial.println("cm");
 delay(1000);
}
```

알고리즘

거리가 30~40 이면 LED on

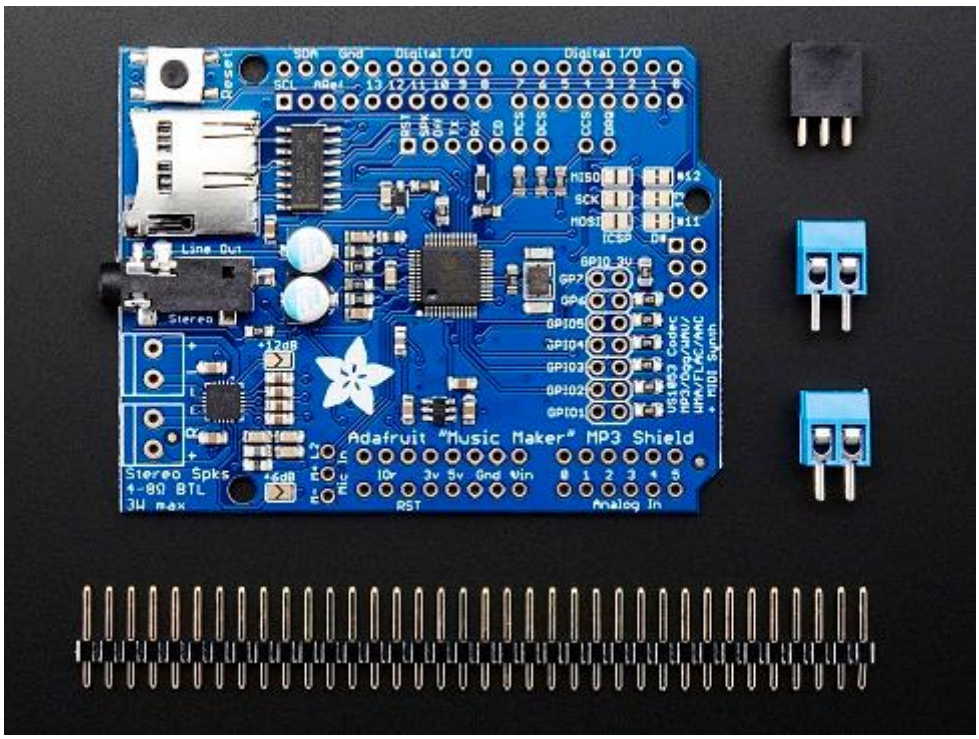
거리가 20~30 이면 부저 울림

거리가 20 이하 이면 차단막 내려옴

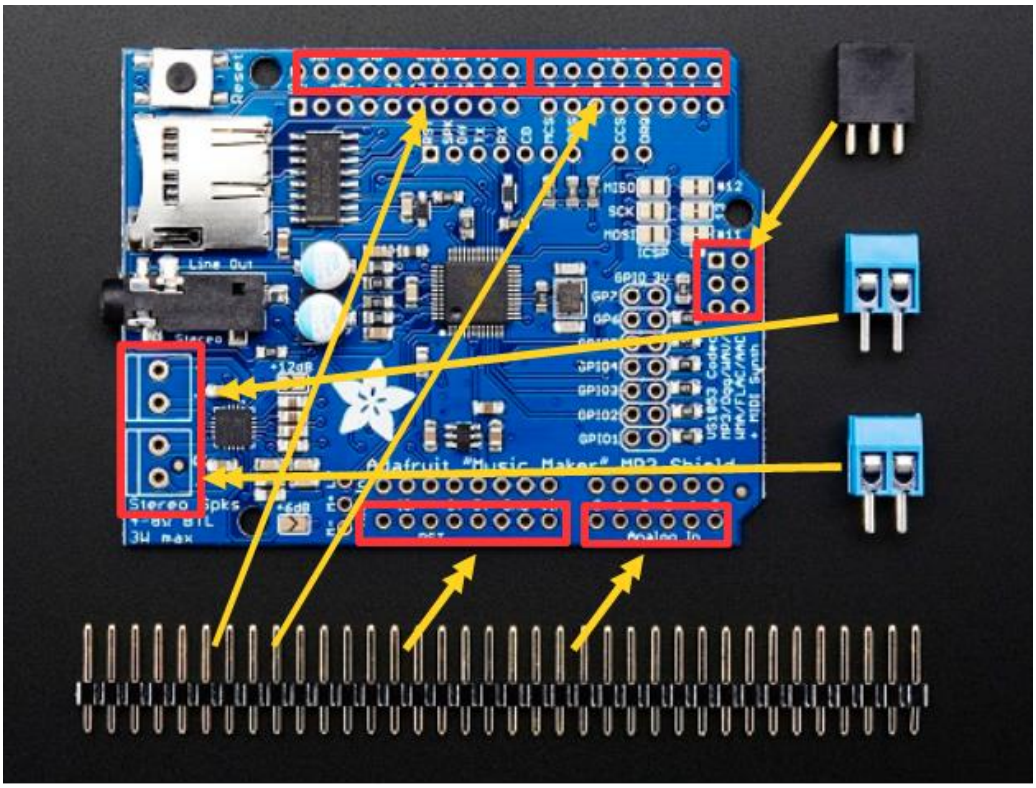


4. 뮤직메이커 쉴드

Adafruit -Music Maker- MP3 Shield for Arduino 를 다음과 같이 납 땜을 하여 사용하여야 한다.



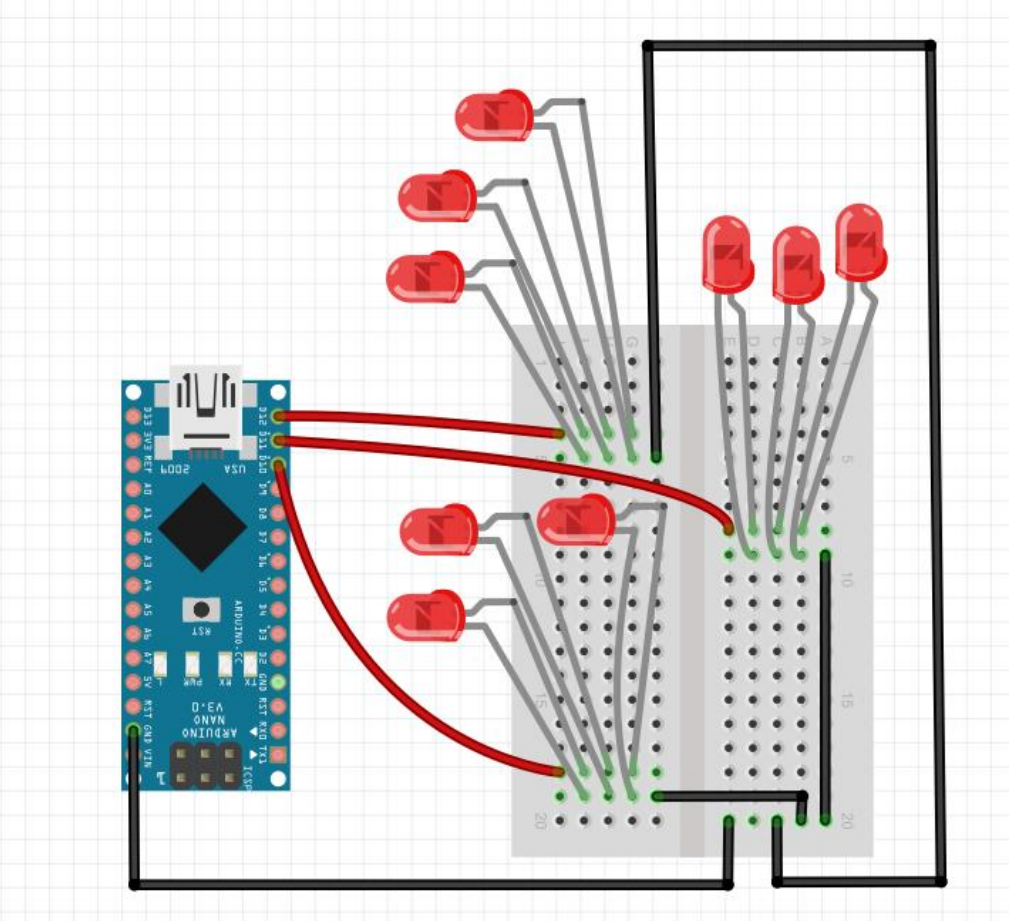
아래 그림과 같이 납 땜을 한다.



VS1053B은 Ogg Vorbis, MP3/MP2/MP1, MP4, AAC, WMA, FLAC, WAV/PCM, MIDI. Encodes Ogg or WAV/PCM 파일 포맷을 지원한다. GPIO 7 핀은 아두이노 라이브러리를 통해 버튼이나 LED를 밝히는데 사용되고, 신디사이저 및 드럼 소리가 내장되어 있다.

MicroSD 카드 소켓 - FAT16/FAT32 포멧, 64MB 이상의 사양이어야 한다.

5. 크리스마스 트리 만들기



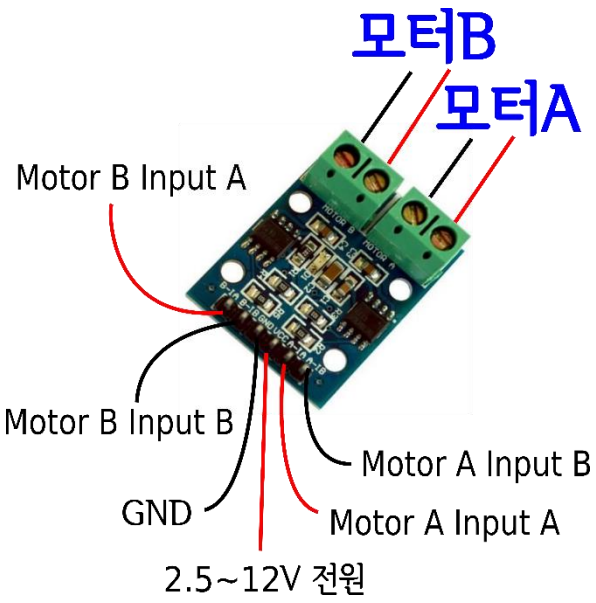
Chapter 6. 아두이노 RC 카

1. 2 륜 구동 RC카

【1】모터 드라이버 모듈

하나의 모터를 2 핀으로 제어하는 모듈은 1 개의 핀으로 방향을 제어하고, 1 개의 핀으로는 속도를 제어합니다.

다음은 L9110 모터 드라이버 모듈입니다. 이 모듈은 방향을 제어하는 핀과 속도를 제어하는 핀 모두를 PWM으로 제어합니다. 아두이노에 연결하는 것은 따로 구분하지 않고 전부 디지털 핀에 연결하면 됩니다.

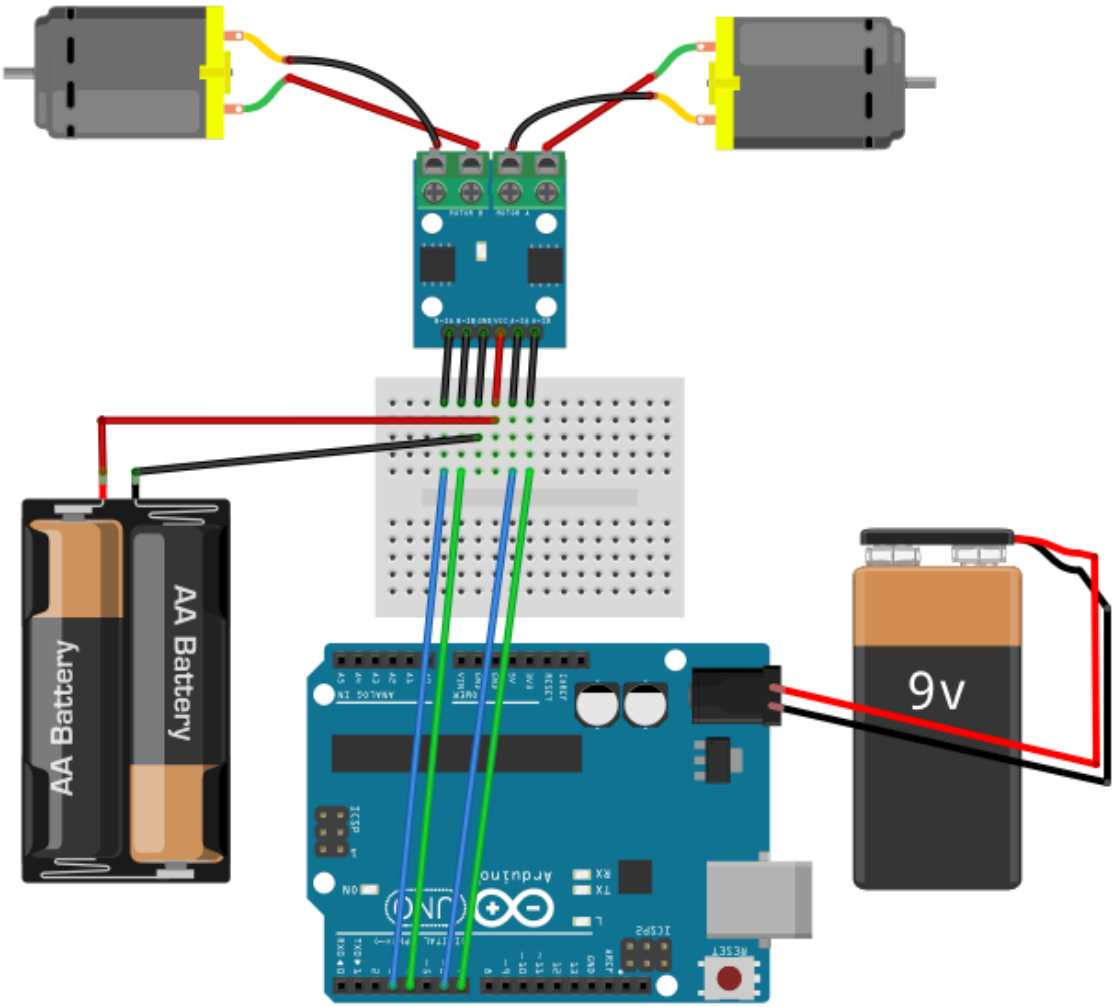


① 아두이노와 연결

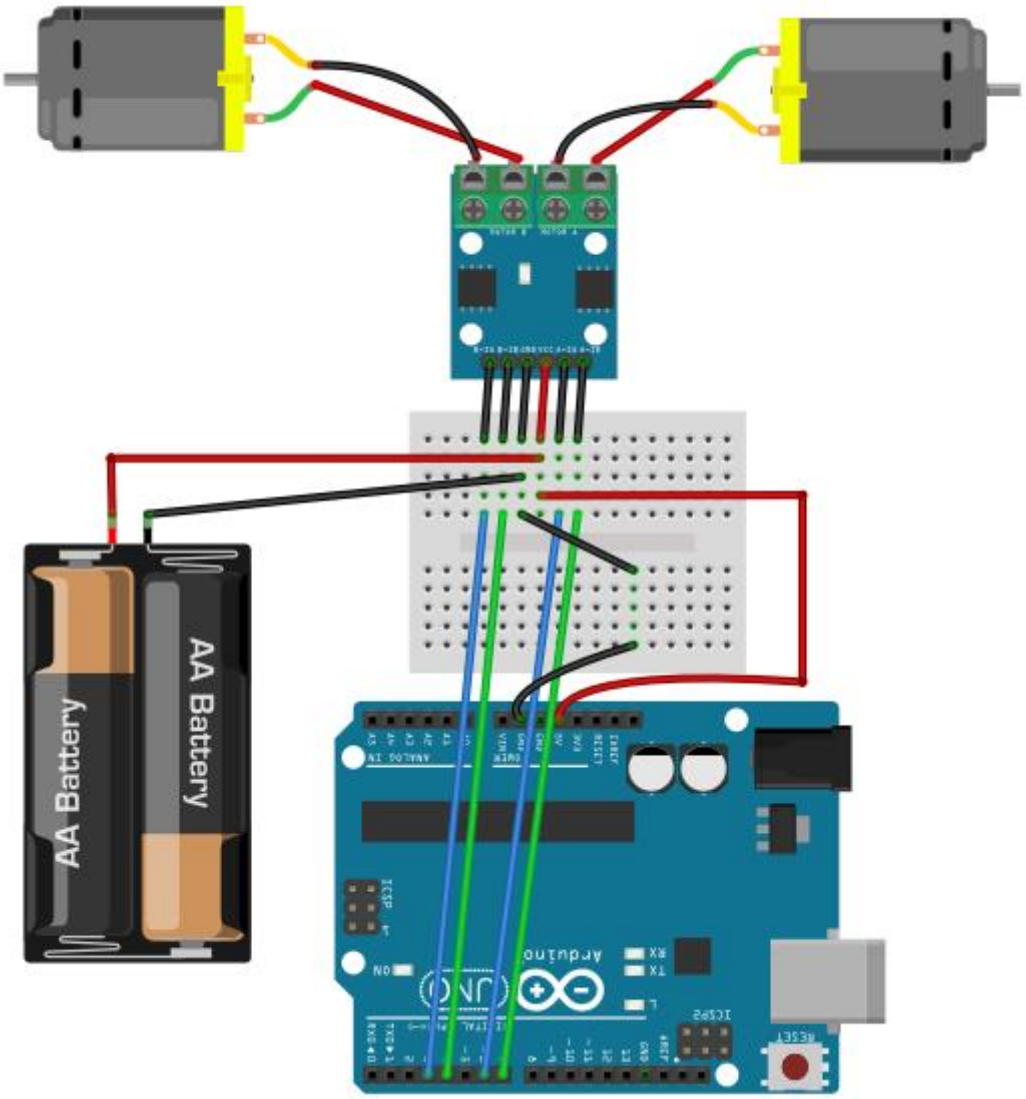
이름	약자	연결 핀	비고
Motor A Input A	AIA	6	디지털
Motor A Input B	AIB	7	디지털
Motor B Input A	BIA	3	디지털
Motor B Input B	BIB	4	디지털
모터 A			왼쪽 바퀴
모터 B			오른쪽 바퀴

중요: VCC외부전원의 ⊕와 아두이노 5V동시에 연결하여야 하고, GND는 외부 전원의 ⊖와 아두이노 GND를 동시에 연결하여야 합니다.

② 회로도



아두이노에 전원을 공급하기 위해서 위와 같이 회로를 구성합니다.



아두이노에도 전원을 공급하기 위해서 앞에서처럼 따로 전원을 공급하여도 되지만, 외부전원에서 모터 드라이브 모듈에도 연결하고 아두이노에도 연결하면 하나의 외부 전원으로 2 개의 장치를 다 사용할 수 있습니다.

【2】주행

① 방향과 속도 제어

전진할 때는 IA에 전진하고자 하는 속도($0 < 값 \leq 255$)를 설정하고, IB 에 0 을 부여합니다.

후진할 때는 IA에 0 을 부여하고, IB에는 후진하고자 하는 속도 $0 < 값 \leq 255$ 를 부여합니다.

정지하고자 할 때는 IA와 IB에 전부 0 을 부여합니다.

이름	AIA	AIB	BIA	BIB	비고
전진	255	0	255	0	최대속도
정지	0	0	0	0	
후진	0	255	0	255	최대속도

중간 속도(150)로 전진하고 너 늦은 속도(50)으로 후진하려면

이름	AIA	AIB	BIA	BIB	비고
전진	150	0	150	0	
정지	0	0	0	0	
후진	0	50	0	50	

로 지정하면 됩니다.

【Sketch】

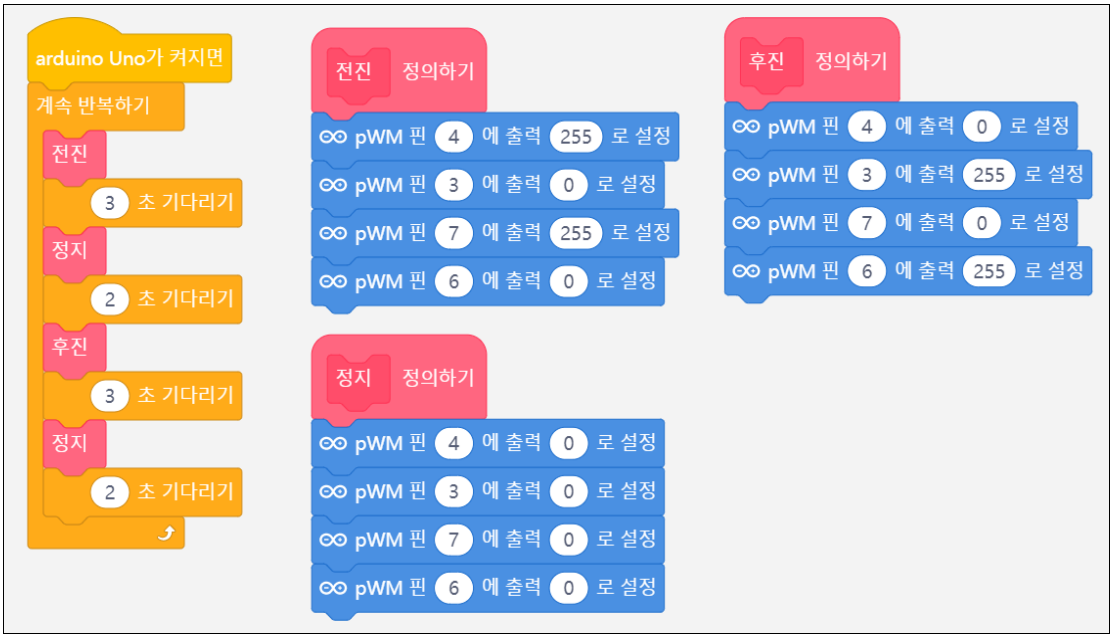
【소스 프로그램】

```
// File Name: L9110_01.ino
const int AIA = 4;
const int AIB = 3;
const int BIA = 7;
const int BIB = 6;

void setup() {
pinMode(AIA, OUTPUT);
pinMode(AIB, OUTPUT);
pinMode(BIA, OUTPUT);
pinMode(BIB, OUTPUT);
}
```

```
void loop() {
  uf_forward();
  delay(2000);
  uf_stop();
  delay(1000);
  uf_backward();
  delay(2000);
  uf_stop();
  delay(1000);
}

void uf_forward()
{
  analogWrite(AIA, 255);
  analogWrite(AIB, 0);
  analogWrite(BIA, 255);
  analogWrite(BIB, 0);
}

void uf_stop()
{
  analogWrite(AIA, 0);
  analogWrite(AIB, 0);
  analogWrite(BIA, 0);
  analogWrite(BIB, 0);
}

void uf_backward()
{
  analogWrite(AIA, 0);
  analogWrite(AIB, 255);
  analogWrite(BIA, 0);
  analogWrite(BIB, 255);
}
```

【mblock】

【코드 블록】



② 좌회전과 우회전

조타 장치를 장착한 경우에는 바퀴를 움직여서 좌회전과 우회전을 하지만, 조타 장치(핸들)가 없는 경우에는 순수하게 모터의 회전 속도로만 조정을 합니다.

좌회전을 하려면, 왼쪽 모터는 천천히 회전시키고 오른쪽 모터는 빨리 회전시키면 됩니다.

우회전을 하려면, 왼쪽 모터는 빠르게 회전시키고 오른쪽 모터는 천천히 회전시키면 됩니다.

【Sketch】

【소스 프로그램】

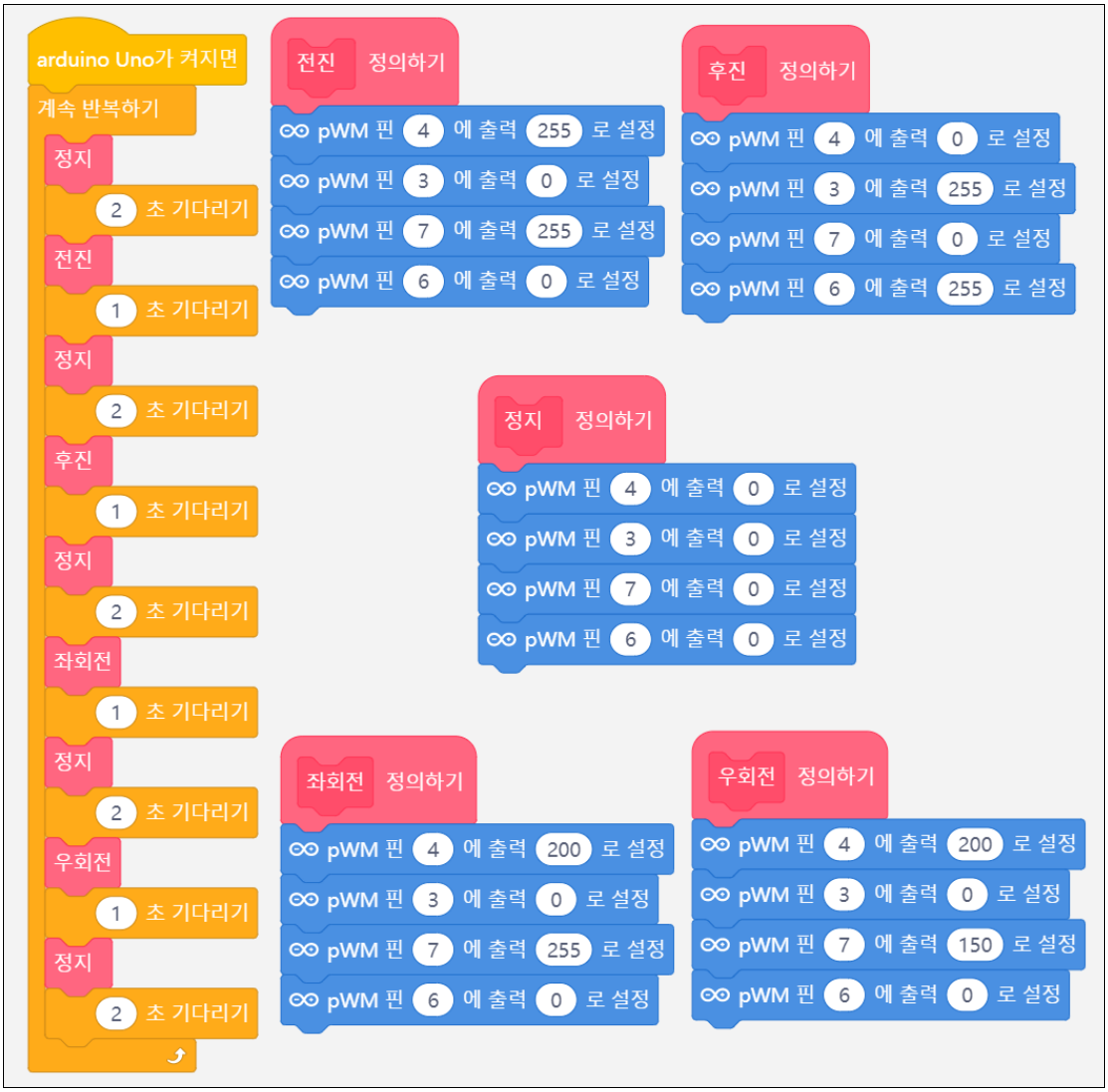
```
// File Name: L9110_02.ino
const int AIA = 6;
```

```
const int AIB = 7;
const int BIA = 3;
const int BIB = 4;

void setup() {
pinMode(AIA, OUTPUT);
pinMode(AIB, OUTPUT);
pinMode(BIA, OUTPUT);
pinMode(BIB, OUTPUT);
}

void loop() {
  uf_forward();
  delay(1000);
  uf_stop();

  uf_backward();
  delay(1000);
  uf_stop();

  uf_leftward();
  delay(1000);
  uf_stop();

  uf_rightward();
  delay(1000);
  uf_stop();

}

void uf_forward()
{
  analogWrite(AIA, 255);
  analogWrite(AIB, 0);
  analogWrite(BIA, 255);
  analogWrite(BIB, 0);
```

```
}

void uf_stop()
{
  analogWrite(AIA, 0);
  analogWrite(AIB, 0);
  analogWrite(BIA, 0);
  analogWrite(BIB, 0);
  delay(2000);
}

void uf_backward()
{
  analogWrite(AIA, 0);
  analogWrite(AIB, 255);
  analogWrite(BIA, 0);
  analogWrite(BIB, 255);
}

void uf_leftward()
{
  analogWrite(AIA, 190);
  analogWrite(AIB, 0);
  analogWrite(BIA, 255);
  analogWrite(BIB, 0);
}

void uf_rightward()
{
  analogWrite(AIA, 255);
  analogWrite(AIB, 0);
  analogWrite(BIA, 190);
  analogWrite(BIB, 0);
}
```

【mblock】

【코드 블록】



③ 문제 해결

모터와 모터모듈과 ⊕, ⊖의 연결이 맞게 되어 있는가 확인합니다.

AIA, AIB, BIA, BIB와 아두이노 핀의 연결이 맞게 되어 있는지 확인합니다.

바닥이 미끄러우면 마찰력의 감소로 원하는 방향으로 방향전환이 되지 않으므로 왼쪽 모터의 값과 오른쪽 모터의 값의 차이를 작게 합니다.

예를 들어

```
void uf_leftward()
{
  analogWrite(AIA, 190);
  analogWrite(AIB, 0);
  analogWrite(BIA, 255);
  analogWrite(BIB, 0);
}

void uf_rightward()
{
  analogWrite(AIA, 255);
  analogWrite(AIB, 0);
  analogWrite(BIA, 190);
  analogWrite(BIB, 0);
}
```

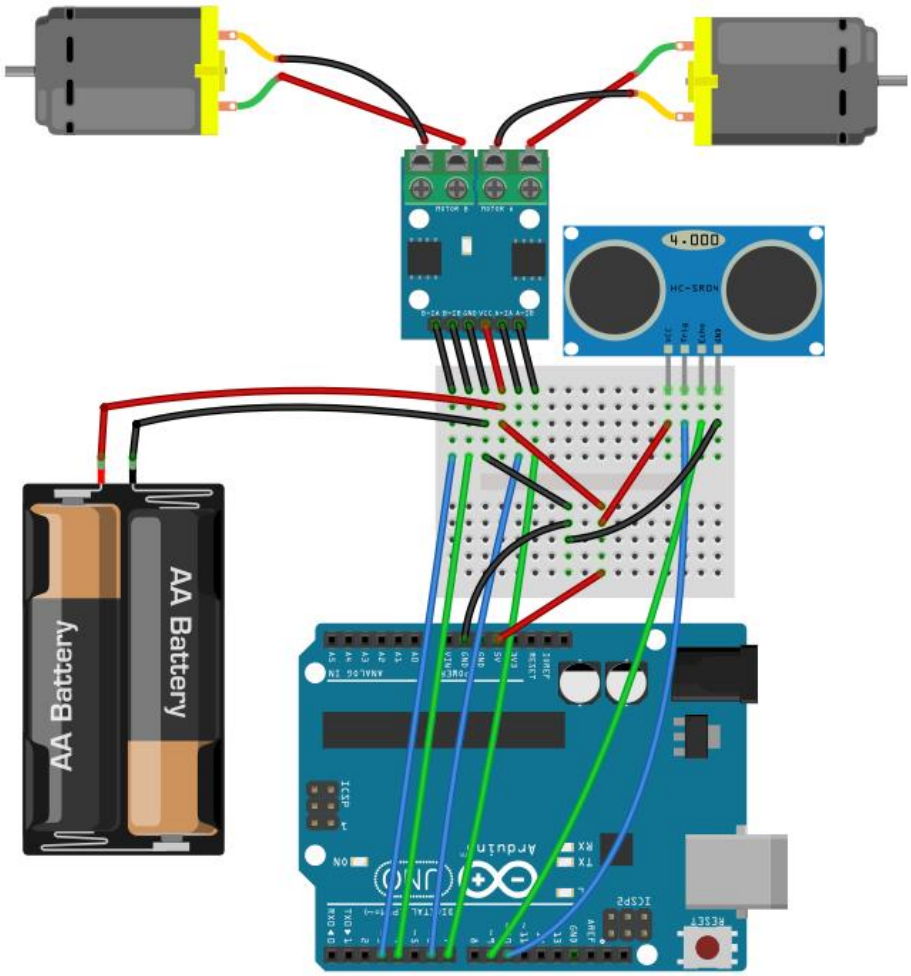
바닥이 미끄러우면 190 보다 큰 값을 주어서 255 와의 간격을 줄입니다.

【3】 충돌 방지

자동차 주행 중 장애물이 발견되는 경우에는 장애물을 피하거나 정지하여야 합니다. 인간은 보거나 듣고서 이를 해결하지만 아두이노는 이를 초음파 센서를 활용하여 장애물을 감지하여 대응합니다.

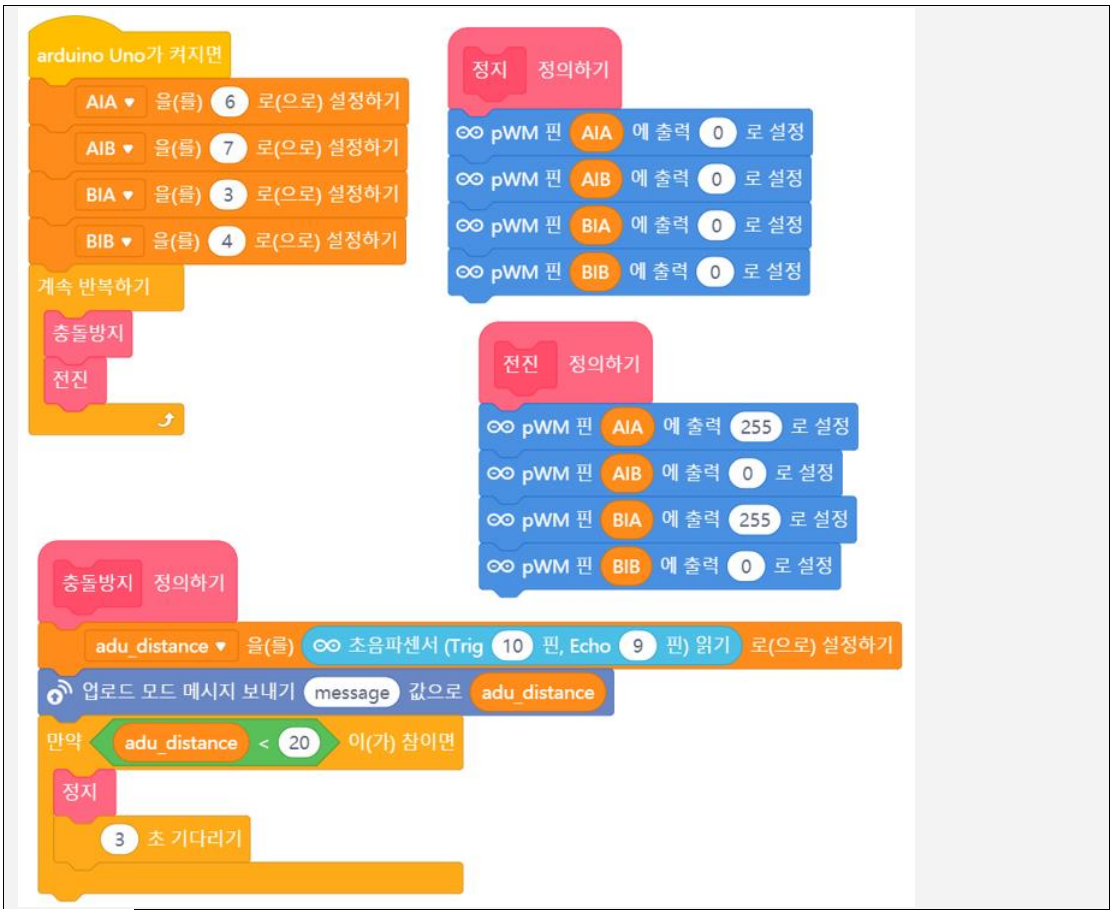
우선 초음파 센서를 추가하여야 합니다. 초음파 센서의 연결을 다시 한 번 살펴봅시다.

초음파 센서	아두이노
VCC	5V
Trig	D10
Echo	D9
GND	GND



20cm 이내에 장애물이 있으면 자동차를 정지시키는 프로그램을 만들어 봅시다.

장치

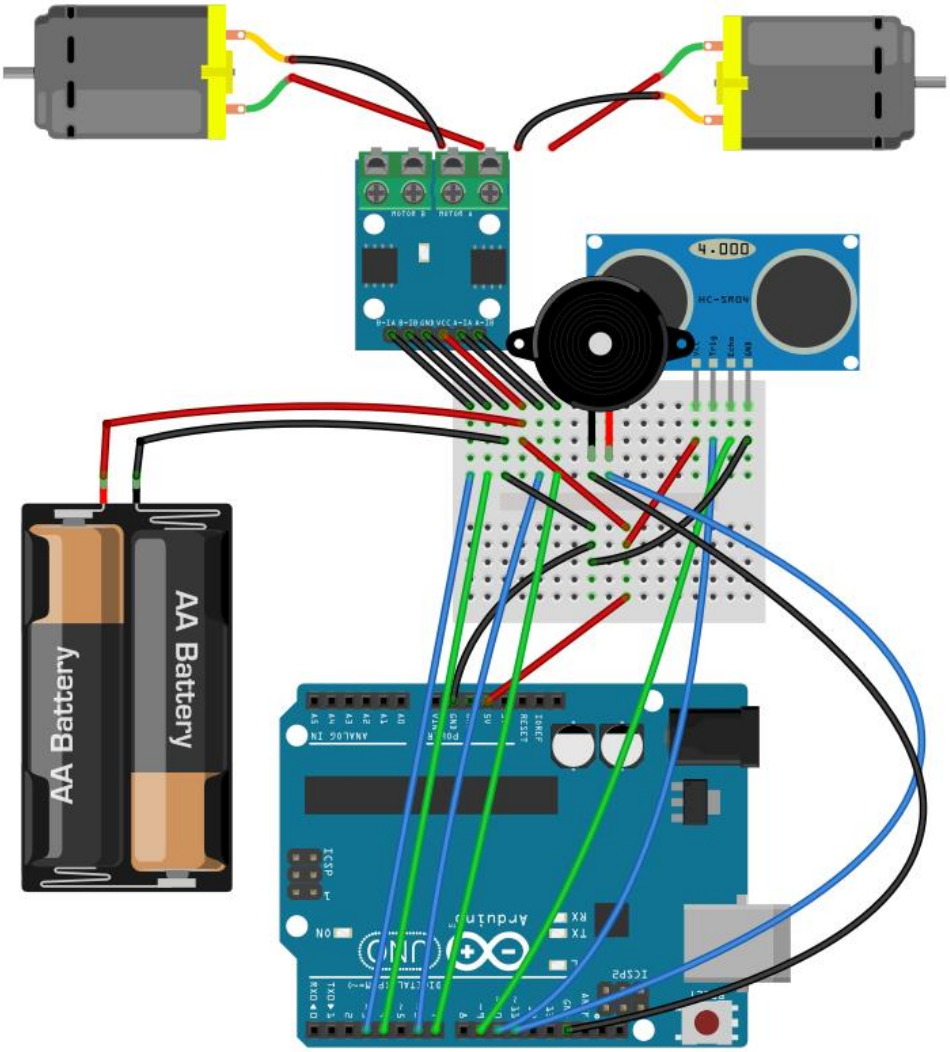


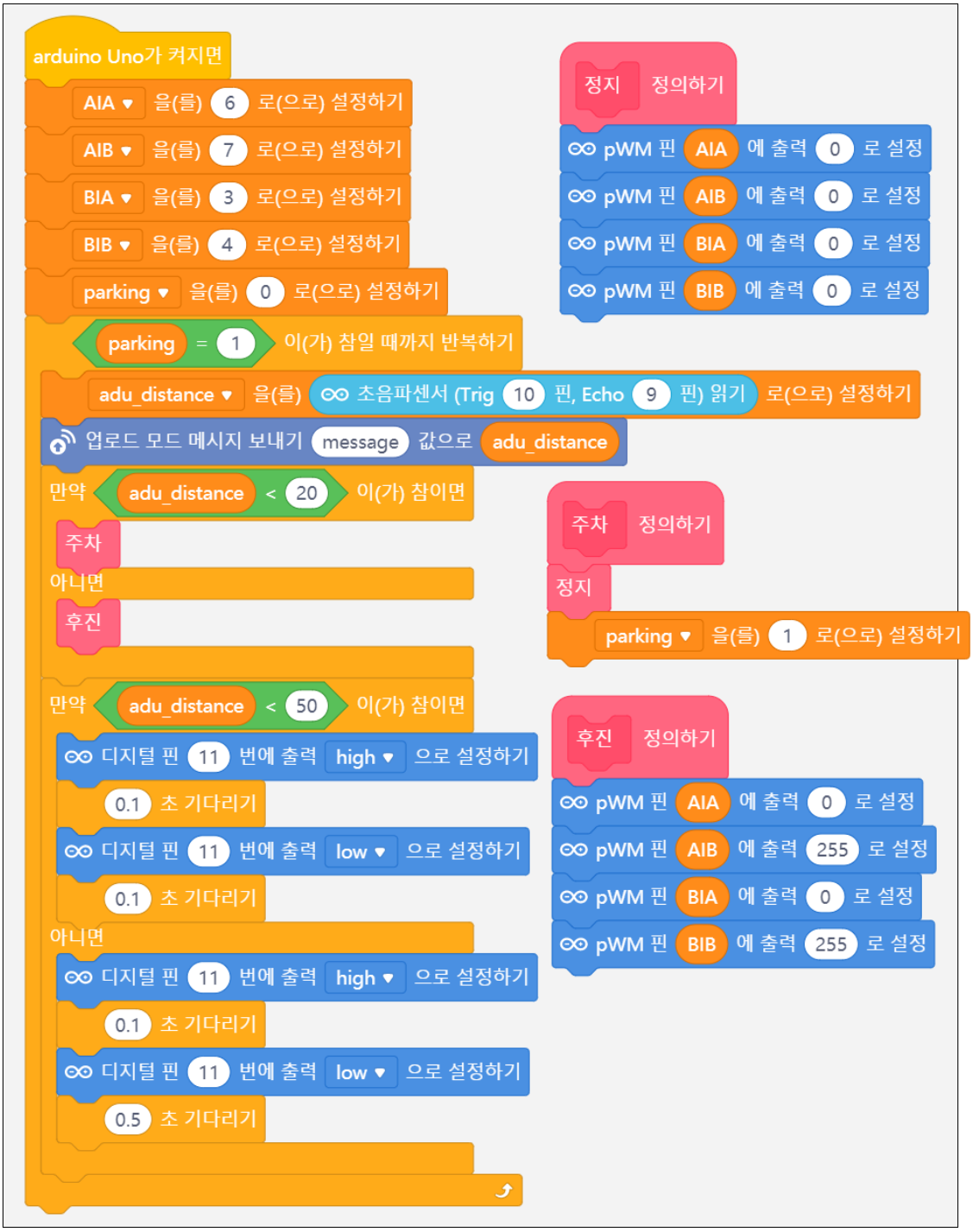
스프라이트

실제로 동작을 시켜보면 정지한 후에도 잠깐씩 이동하는 경우가 왕왕 있는데, 이는 초음파 센서가 일시적으로 잘못 측정하기 때문에 발생하는 현상입니다. 이 문제를 해결하기 위해서는 에러 없는 초음파 센서를 사용하거나 소프트웨어 기법으로 보정하는 고급 기법이 필요합니다.

【4】주차

주차는 보통 후진을 하면서 이루어 집니다. 여기서는 후진할 때 50cm 이상 일정 거리가 되면 부저가 울리고 더 가까워지면 부저가 빠른 속도로 울리면서 거리가 20cm 미만이 되면 정지하는 프로그램을 만들어 봅시다.




arduino Uno가 켜지면
AIA ▾ 을(를) 6 로(으로) 설정하기
AIB ▾ 을(를) 7 로(으로) 설정하기
BIA ▾ 을(를) 3 로(으로) 설정하기
BIB ▾ 을(를) 4 로(으로) 설정하기
parking ▾ 을(를) 0 로(으로) 설정하기
parking = 1 이(가) 참일 때까지 반복하기
adu_distance ▾ 을(를) 초음파센서 (Trig 10 핀, Echo 9 핀) 읽기 로(으로) 설정하기
업로드 모드 메시지 보내기 message 값으로 adu_distance
만약 adu_distance < 20 이(가) 참이면
주차
아니면
후진
만약 adu_distance < 50 이(가) 참이면
디지털 핀 11 번에 출력 high ▾ 으로 설정하기
0.1 초 기다리기
디지털 핀 11 번에 출력 low ▾ 으로 설정하기
0.1 초 기다리기
아니면
디지털 핀 11 번에 출력 high ▾ 으로 설정하기
0.1 초 기다리기
디지털 핀 11 번에 출력 low ▾ 으로 설정하기
0.5 초 기다리기
정지 정의하기
pWM 핀 AIA 에 출력 0 로 설정
pWM 핀 AIB 에 출력 0 로 설정
pWM 핀 BIA 에 출력 0 로 설정
pWM 핀 BIB 에 출력 0 로 설정
주차 정의하기
정지
parking ▾ 을(를) 1 로(으로) 설정하기
후진 정의하기
pWM 핀 AIA 에 출력 0 로 설정
pWM 핀 AIB 에 출력 255 로 설정
pWM 핀 BIA 에 출력 0 로 설정
pWM 핀 BIB 에 출력 255 로 설정

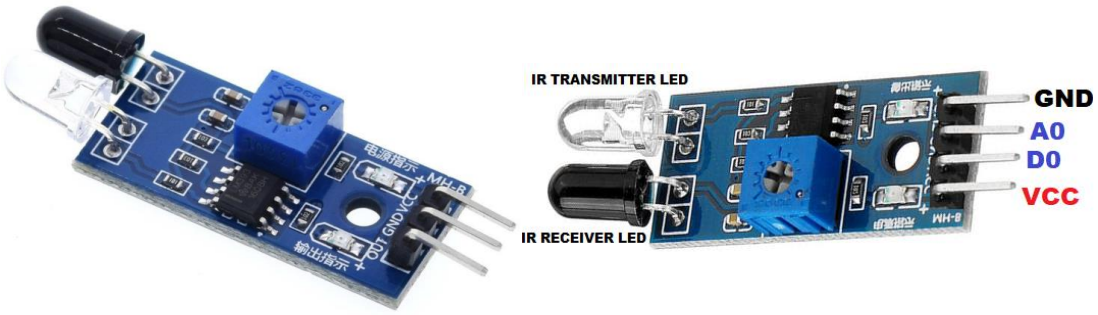
【5】라인 트레이셔

① 라인 트레이셔란?

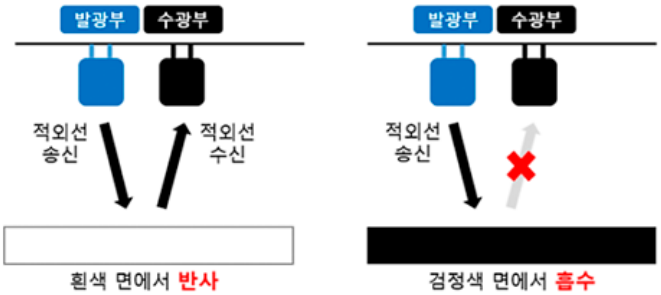
일명 라인트레이서 (Line Tracer) 센서라 불리는 센서는 적외선 센서로 발광부에서 적외선을 발사하여 물체에 닿아 반사되는 들어오는 신호를 수광부에서 수신하여 수신되는 양을 계산하여 반사된 물체를 식별하는 센서입니다.

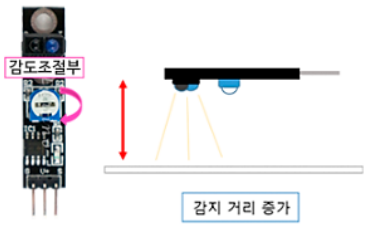
빛의 성질상 다른 색은 물체 고유의 색을 반사하지만, 검은 색은 흡수하므로 검은 색을 인식하여 조작을 하는데 유용하게 사용됩니다.

만일 도로가 검정색이라면 이 검정색 길을 쭉 따라서 주행할 수 있는 장치를 만들 수 있습니다. 따라서 검정색 선을 따라가는 것을 라인트레이셔라고 합니다. 다음 그림과 같은 센서가 라인트레이서 센서입니다.



② 라인트레이셔 센서의 원리





[감도 조절부를 시계 방향으로 돌렸을 때]

[감도 조절부를 반시계 방향으로 돌렸을 때]

센서의 탐지 거리는 감도 조절을 통해 조절할 수 있습니다.
센서의 민감도 (Sensitivity)를 조절할 수 있는 부분이 '감도 조절부'이며,
시계방향으로 돌리면 센서의 탐지 거리를 늘려주며,
반시계방향으로 돌리면 센서의 탐지 거리를 줄여줍니다.

SPEC
라인트레이서 센서 모듈 스펙

크기	6.1 * 1.85 cm
용도	라인트레이서와 같은 라인(선)을 인식하는 용도
사용전압	3.3V ~ 5V
감도 조절 방법	가변저항을 이용하여 감도 조절 가능
측정범위	12mm 내외
특징	- OUT핀을 통해 아날로그 값과 디지털 값으로 출력 가능 - 흰색~검은색 감지가능 - 센서 감도는 센서에 부착된 가변저항을 이용해 조정

라인트레이셔는 센서가 1 개인가 2 개인가에 따라서 알고리즘이 다릅니다. 센서가 2 개인 경우를 살

펴봅시다.

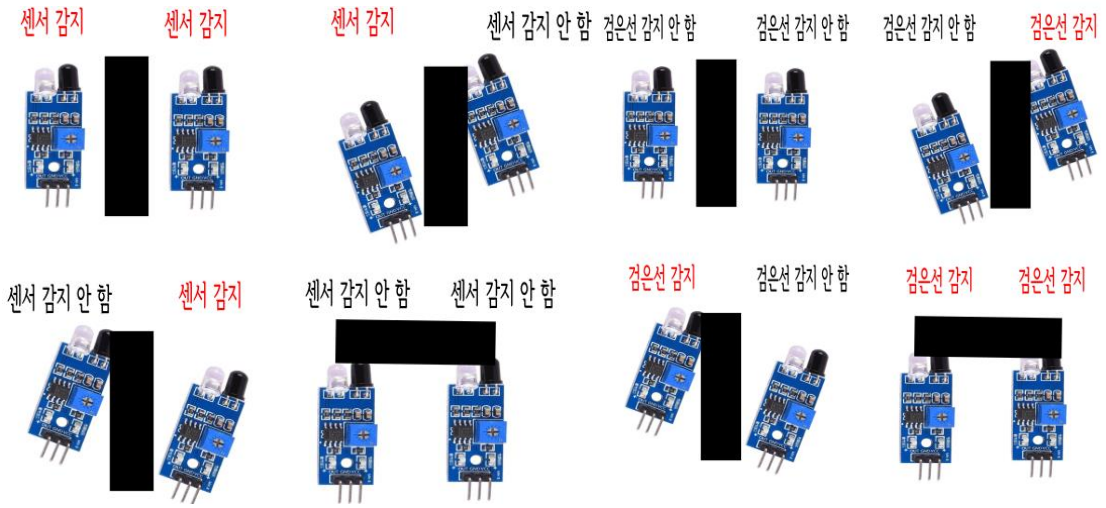
라인트레이서 센서는 검은 색은 감지하지 않고, 다른 색은 감지합니다.

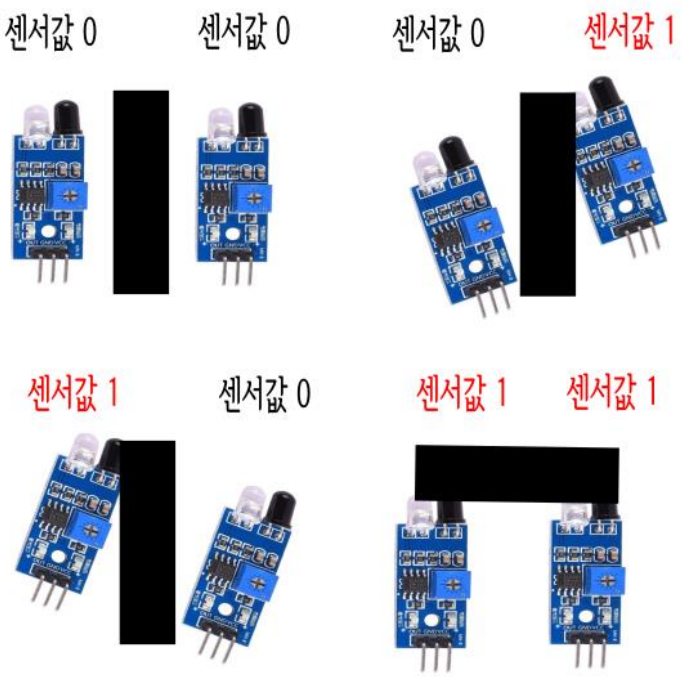
감지하면 불이 들어오고, 감지하지 않으면 불이 들어오지 않습니다.

센서에 불이 들어 온다는 말은 센서값이 1 이고, 센서에 불이 들어 오지 않는다는 것은 센서값이 0 이라는 의미라고 생각할 수 있지만, 라인트레이서 센서의 목표는 검정색을 감지하느냐 하지 않느냐이므로 검정색을 감지하였을 때 센서 값을 1 로 두고, 검정 색이 아닌 다른 값을 감지한 경우에는 검정색을 감지하지 못한 것으로 간주하여 센서 값을 0 으로 둡니다.

이처럼 감지를 했느냐 하지 않았느냐는 실제 센서가 "적외선을 수신하여 감지를 했느냐? 하지 않았느냐?" 와 "검정색을 감지 했느냐? 하지 않았느냐?"와 혼돈이 오므로 센서값을 기준으로 처리하는 것이 합리적입니다.

감지와 센서값
검정색을 감지하면 센서값이 1
검정색을 감지하지 않으면 센서값이 0

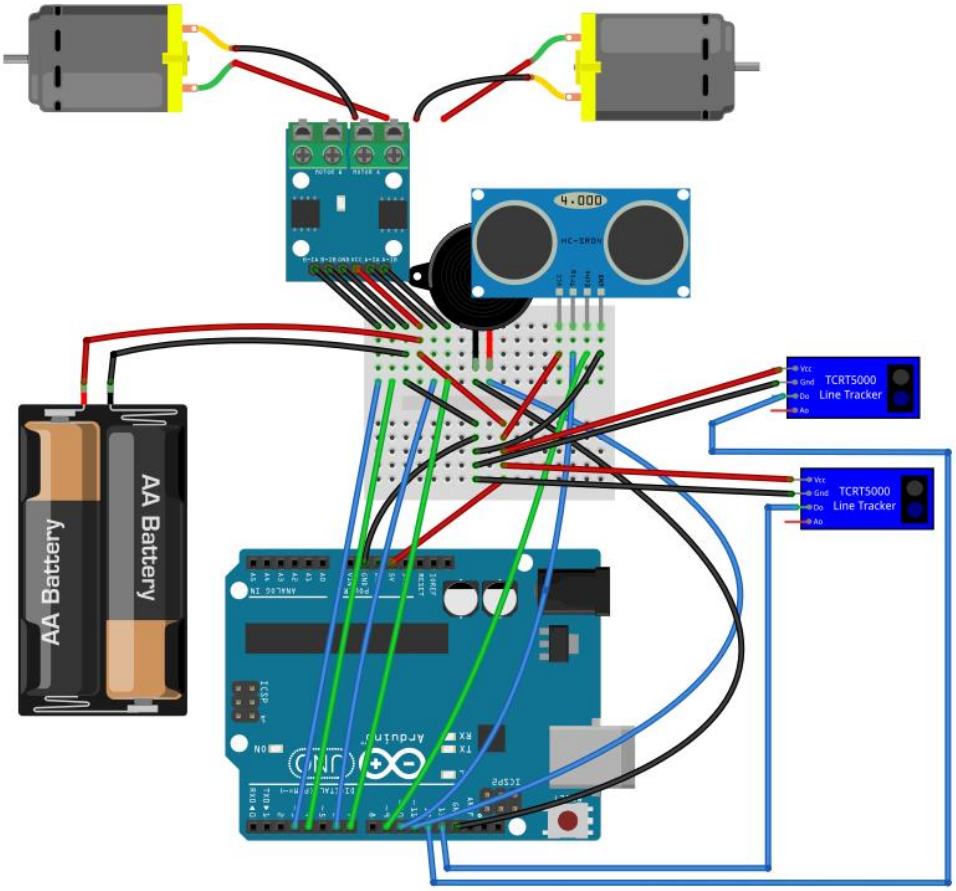




따라서 센서를 왼쪽과 오른쪽 2 개를 사용하는 경우에

(왼쪽 0) & (오른쪽 0)	직 진
(왼쪽 0) & (오른쪽 1)	우회전
(왼쪽 1) & (오른쪽 0)	좌회전
(왼쪽 1) & (오른쪽 1)	정 지

③ 회로도



④ 프로그램

【Sketch】

【소스 프로그램】

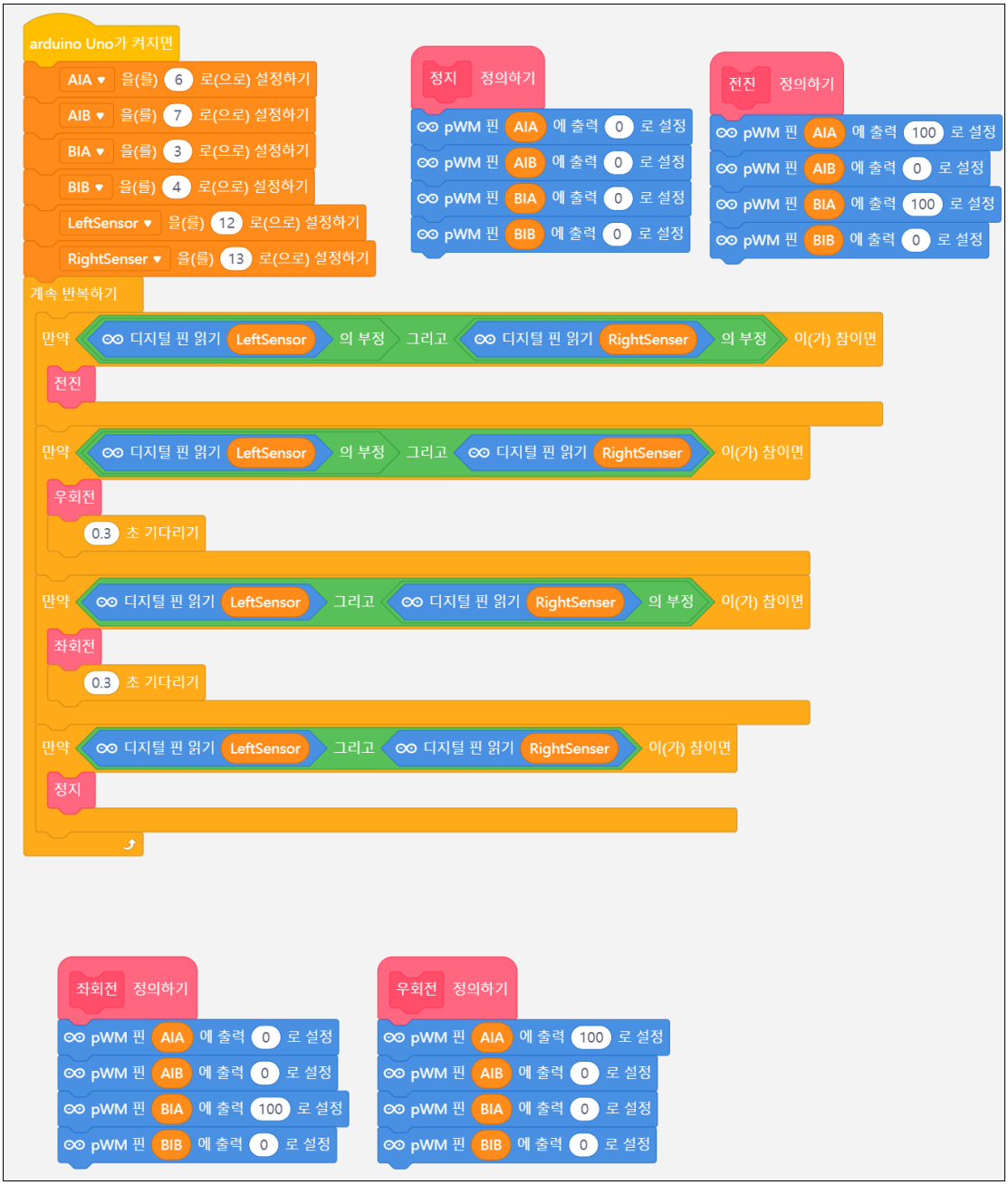
```
// File Name: ex200_LineTracer_01.ino
const int AIA = 6;
const int AIB = 7;
const int BIA = 3;
const int BIB = 4;
const int leftLineSensor =  12;
const int rightLineSensor = 13;
```

```
int leftValue=0;
int rightValue=0;

void setup() {
pinMode(AIA, OUTPUT);   //Left Motor
pinMode(AIB, OUTPUT);
pinMode(BIA, OUTPUT);   //Right Motor
pinMode(BIB, OUTPUT);

pinMode(leftLineSensor, INPUT);
pinMode(rightLineSensor, INPUT);
Serial.begin(9600);
}

void loop() {
   leftValue=digitalRead(leftLineSensor);
   rightValue=digitalRead(rightLineSensor);

   if (leftValue==0 && rightValue==0) {
    uf_forward();
   }
  if (leftValue==0 && rightValue==1) {
    uf_right(); delay(300);
  }

  if (leftValue==1 && rightValue==0) {
    uf_left(); delay(300);
  }
  if (leftValue==1 && rightValue==1) {
    uf_stop();
  }

  //Serial.print("Left=" & digitalRead(leftLineSensor);
  //Serial.print("Right=" & digitalRead(rightLineSensor);
  Serial.print("Left="); Serial.println(digitalRead(leftLineSensor));
  Serial.print("Right="); Serial.println(digitalRead(rightLineSensor));
  //Serial.print("\n");
```

```
  delay(500);
  //Serial.println("Right=",digitalRead(rightLineSensor));
}

void uf_forward()
{
  analogWrite(AIA, 150);   analogWrite(AIB, 0);
  analogWrite(BIA, 150);   analogWrite(BIB, 0);
}

void uf_stop()
{
  analogWrite(AIA, 0);  analogWrite(AIB, 0);
  analogWrite(BIA, 0);  analogWrite(BIB, 0);
  delay(2000);
}

void uf_left()
{
  analogWrite(AIA, 0);    analogWrite(AIB, 0);
  analogWrite(BIA, 150);  analogWrite(BIB, 0);
}

void uf_right()
{
  analogWrite(AIA, 150);    analogWrite(AIB, 0);
  analogWrite(BIA, 0);      analogWrite(BIB, 0);
}
```

【mblock】

【코드 블록】


arduino Uno가 켜지면
AIA ▾ 을(를) 6 로(으로) 설정하기
AIB ▾ 을(를) 7 로(으로) 설정하기
BIA ▾ 을(를) 3 로(으로) 설정하기
BIB ▾ 을(를) 4 로(으로) 설정하기
LeftSensor ▾ 을(를) 12 로(으로) 설정하기
RightSenser ▾ 을(를) 13 로(으로) 설정하기
계속 반복하기
만약 디지털 핀 읽기 LeftSensor 의 부정 그리고 디지털 핀 읽기 RightSenser 의 부정 이(가) 참이면
전진
만약 디지털 핀 읽기 LeftSensor 의 부정 그리고 디지털 핀 읽기 RightSenser 이(가) 참이면
우회전
0.3 초 기다리기
만약 디지털 핀 읽기 LeftSensor 그리고 디지털 핀 읽기 RightSenser 의 부정 이(가) 참이면
좌회전
0.3 초 기다리기
만약 디지털 핀 읽기 LeftSensor 그리고 디지털 핀 읽기 RightSenser 이(가) 참이면
정지
정지 정의하기
pWM 핀 AIA 에 출력 0 로 설정
pWM 핀 AIB 에 출력 0 로 설정
pWM 핀 BIA 에 출력 0 로 설정
pWM 핀 BIB 에 출력 0 로 설정
전진 정의하기
pWM 핀 AIA 에 출력 100 로 설정
pWM 핀 AIB 에 출력 0 로 설정
pWM 핀 BIA 에 출력 100 로 설정
pWM 핀 BIB 에 출력 0 로 설정
좌회전 정의하기
pWM 핀 AIA 에 출력 0 로 설정
pWM 핀 AIB 에 출력 0 로 설정
pWM 핀 BIA 에 출력 100 로 설정
pWM 핀 BIB 에 출력 0 로 설정
우회전 정의하기
pWM 핀 AIA 에 출력 100 로 설정
pWM 핀 AIB 에 출력 0 로 설정
pWM 핀 BIA 에 출력 0 로 설정
pWM 핀 BIB 에 출력 0 로 설정

Input A에

```
/*
 *
 * Smart Robot Car V3
 * - Auto driving
 * - Obstacle avoidance(장애물 회피)
 */

////////////////////////////////////////////////
// Ultrasonic sensor
//
#define Sonic_TRIG    12      // Ultrasonic Trigger
#define Sonic_ECHO    13      // Ultrasonic Echo

#define MAX_DISTANCE  200

int cmFront = 0;
int cmF_old = 0;
int cmF_avg = 0;

////////////////////////////////////////////////
// Infrared sensor(FC-51)
////////////////////////////////////////////////
// Obstacle = LOW,  Nothing  = HIGH
//
#define pin_InfraR    8      // Right Infrared sensor
#define pin_InfraL    9      // Left Infrared sensor

int InfraL = HIGH;
int InfraR = HIGH;
int InfraL_Old = HIGH;
int InfraR_Old = HIGH;
```

```
int obstacle_cnt = 0;
int obstacle_try = 0;
long duration, cm, avg_cm;

//////////////////////////////////////////////////////////
// Note:  ENA and ENB must be connected to PWD supported pins
//
#define ENA   6    // PWD
#define EN1   7
#define EN2   3

#define EN3   4
#define EN4   2
#define ENB   5    // PWD

//////////////////////////////////
// Car direction
//
#define CAR_DIR_FW  0   // forward
#define CAR_DIR_BK  1   // backward
#define CAR_DIR_LT  2   // left turn
#define CAR_DIR_RT  3   // right turn
#define CAR_DIR_ST  4   // stop

//////////////////////////////////
// Car Speed : 0 ~ 255
//
#define CAR_SPEED_DEFAULT 150

//////////////////////////////////
// Default direction and speed
//
```

```
int g_carDirection = CAR_DIR_ST;
int g_carSpeed_L;
int g_carSpeed_R;

////////////////////////////////////////////////
// Note : confirm HIGH/LOW for correct movement
//
void car_forward()
{
  digitalWrite(EN1, HIGH);
  digitalWrite(EN2, LOW);
  analogWrite(ENA, g_carSpeed_R);

  digitalWrite(EN3, HIGH);
  digitalWrite(EN4, LOW);
  analogWrite(ENB, g_carSpeed_L);
}

void car_backward()
{
  digitalWrite(EN1, LOW);
  digitalWrite(EN2, HIGH);
  analogWrite(ENA, g_carSpeed_R);

  digitalWrite(EN3, LOW);
  digitalWrite(EN4, HIGH);
  analogWrite(ENB, g_carSpeed_L);
}

void car_left()
{
  digitalWrite(EN1, LOW);
  digitalWrite(EN2, HIGH);
```

```
  analogWrite(ENA, g_carSpeed_R);

  digitalWrite(EN3, HIGH);
  digitalWrite(EN4, LOW);
  analogWrite(ENB, g_carSpeed_L);
}

void car_right()
{
  digitalWrite(EN1, HIGH);
  digitalWrite(EN2, LOW);
  analogWrite(ENA, g_carSpeed_R);

  digitalWrite(EN3, LOW);
  digitalWrite(EN4, HIGH);
  analogWrite(ENB, g_carSpeed_L);
}

void car_stop()
{
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
}

////////////////////////////
// Execute car moving
//
void update_Car()
{
  switch ( g_carDirection ) {
    case CAR_DIR_FW:
        car_forward();
        break;
```

```
    case CAR_DIR_BK:
      car_backward();
      break;
    case CAR_DIR_LT:
      car_left();
      break;
    case CAR_DIR_RT:
      car_right();
      break;
    case CAR_DIR_ST:
      car_stop();
      break;
    default :
      ;
  }
  return;
}

void setup() {

  // for debug
  Serial.begin(9600);
  delay(100);
  Serial.println("Auto Driving(Obstacle avoidance) >> Start");

  //
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(EN1, OUTPUT);
  pinMode(EN2, OUTPUT);
  pinMode(EN3, OUTPUT);
  pinMode(EN4, OUTPUT);
```

```
  pinMode(Sonic_TRIG,OUTPUT);
  pinMode(Sonic_ECHO,INPUT);
  pinMode(pin_InfraR,INPUT);
  pinMode(pin_InfraL,INPUT);

}

void loop() {

  //
  GetDistance_UltraSonic();
  DetectObstacle_InfraRed();

  g_carSpeed_L = CAR_SPEED_DEFAULT;
  g_carSpeed_R = CAR_SPEED_DEFAULT;

  Serial.print("cmFront -> ");
  Serial.println(cmFront);

  if ( (InfraL == LOW)||(InfraR == LOW) )     //좌측이나 우측에 장애물이 있다면
  {

    Serial.println("Left or Right : obstacle");
    //Serial.print("cmFront -> ");
    //Serial.println(cmFront);
    if( cmFront < 30 )        //전방센서 근접에 장애물이 있다면 약간 후진
    {
      Serial.println("Front(LR) : obstacle -> BACK");
      g_carDirection = CAR_DIR_BK;
      update_Car();
      delay(100);
    }
    else {
```

```
 if( InfraL < InfraR )       //좌측에 장애물이 있다면 우측으로 회피
 {
   Serial.println("LEFT : obstacle -> Turn Right");
   g_carSpeed_R = CAR_SPEED_DEFAULT;
   g_carDirection = CAR_DIR_RT;  //우측으로 선회 회피
   update_Car();
   delay(10);
   obstacle_cnt += 1;
 }
 else if( InfraL > InfraR ){   //우측에 장애물이 있다면 좌측으로 회피
   Serial.println("RIGHT : obstacle -> Turn Left");
   g_carSpeed_L = CAR_SPEED_DEFAULT;
   g_carDirection = CAR_DIR_LT;  //좌측으로 선회 회피
   update_Car();
   delay(10);
   obstacle_cnt += 1;
 }
 else {
   Serial.println("No : obstacle - ERROR?");
   g_carDirection = CAR_DIR_FW;
   update_Car();
   delay(100);
 }
}

//g_carSpeed_L = CAR_SPEED_DEFAULT;
//g_carSpeed_R = CAR_SPEED_DEFAULT;

//좌우 측면에 장애물이 계속 감지 된다면 코너에 있는 것으로 간주
if (obstacle_cnt>5) {
```

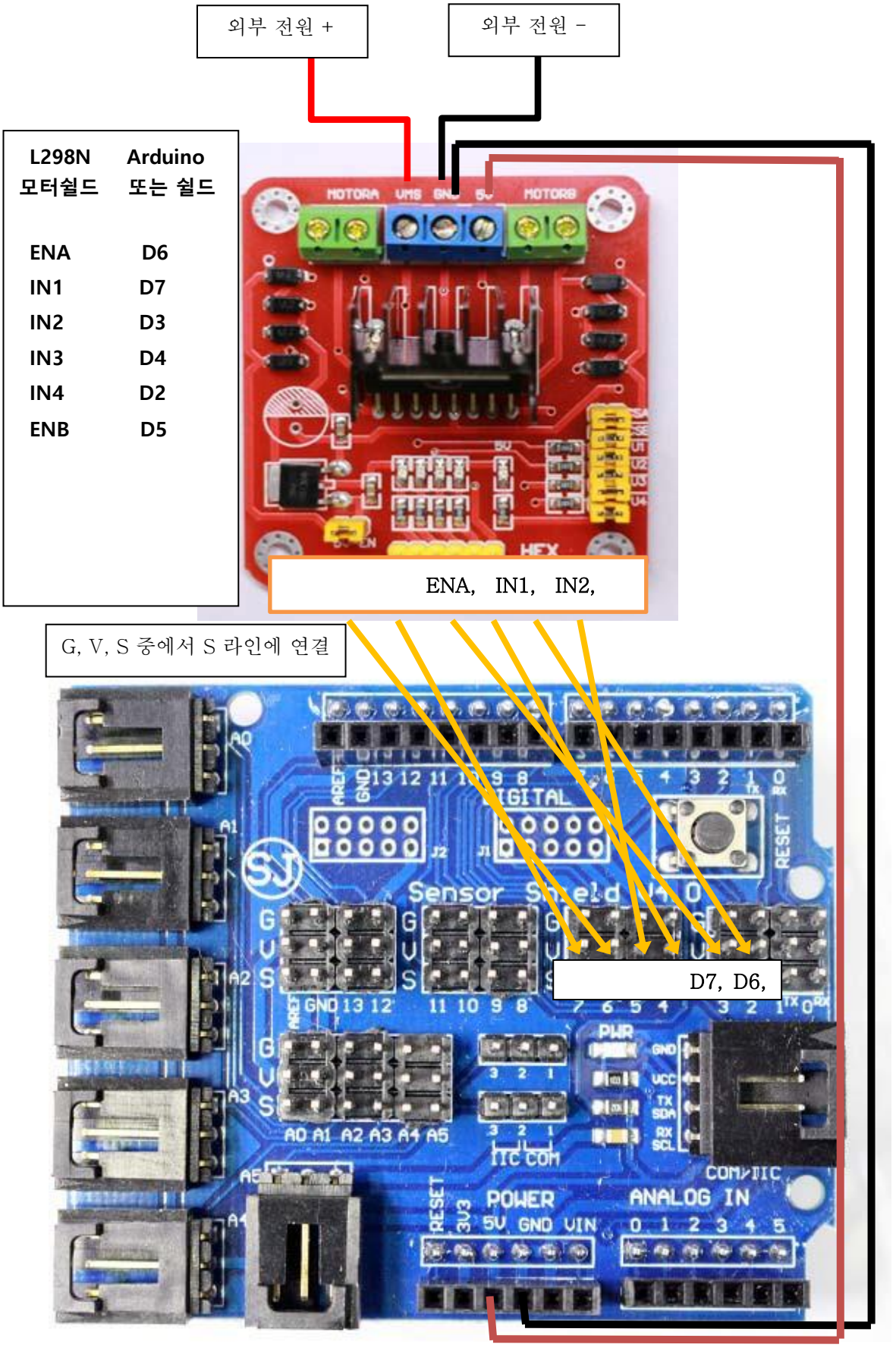
```
do{

 if( InfraL < InfraR ) {      //좌측에 장애물이 있다면
  g_carDirection = CAR_DIR_RT;  //우측으로 회피
 } else {              //우측에 장애물이 있다면
  g_carDirection = CAR_DIR_LT;  //좌측으로 회피
 }
 update_Car();
 delay(100);

 GetDistance_UltraSonic();   //전방(Ultrasonice) 장애물 감지

 if ( cmFront < 15) { //전방에 장애물이 있으면 후진.
  g_carDirection = CAR_DIR_BK;
  update_Car();
  delay(100);
 }

 if ( cmFront > 40 ){  //트인공간이 있으면 탈출
  g_carDirection = CAR_DIR_FW;
  update_Car();
  delay(100);
  obstacle_cnt = 0;
  break;
 }

 if (obstacle_try > 30) {    //탈출하지 못하면 약간 후진.
  g_carDirection = CAR_DIR_BK;
  update_Car();
  delay(50);
  obstacle_cnt = 0;
```

```
        obstacle_try = 0;
        break;
      }
      obstacle_try+=1;

      Serial.println("obstacle count : Corner??");

    }while(true);

  }

}
else {       //좌우 측면에 장애물이 없다면

  obstacle_try = 0;
  obstacle_cnt = 0;
  g_carSpeed_L = CAR_SPEED_DEFAULT;
  g_carSpeed_R = CAR_SPEED_DEFAULT;

  if( cmFront >= 30 ) {

    g_carDirection = CAR_DIR_FW;

//    if( cmFront >= 70 ) { //장애물이 70cm 밖에 있으면 직진
//      g_carSpeed_L = CAR_SPEED_DEFAULT;
//      g_carSpeed_R = CAR_SPEED_DEFAULT;
//    }
    update_Car();

    Serial.println("Nothing at front -> go FW");
  }
  else if ( cmFront < 30 ) {
```

```
      Serial.println("Front 30 obstacle -> turn RIGHT(15<>30)");
      g_carDirection = CAR_DIR_RT;  //우측으로 선회 회피
      update_Car();
      delay(10);
    }
    else if ( cmFront < 15 ) {  //장애물이 15cm 이내이면 후진
     Serial.println("Front 15 obstacle -> BACK");
     g_carDirection = CAR_DIR_BK;
     update_Car();
     delay(100);
    }
    else {
     // nothing
     ;
    }

  } // no Left/Right obstacle

  //
  // For debug : possible to check out direction with delay()
  //
  //

  delay(2000);

}  // loop

long microsecondsToCentimeters(long microseconds)
{
  return microseconds/29/2;
}

void GetDistance_UltraSonic(){
```

```
  //HC-SR04 Ultrasonic sensor(0~400cm)
  digitalWrite(Sonic_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(Sonic_TRIG,LOW);
  duration = pulseIn(Sonic_ECHO, HIGH);
  cmFront = microsecondsToCentimeters(duration);

  if ( cmFront > MAX_DISTANCE ) { cmFront = MAX_DISTANCE; }

  cmF_avg = (cmF_old + cmFront) / 2;
  cmF_old = cmFront;
  cmFront = cmF_avg;
  delay(10);

  Serial.println( cmFront );

}


void DetectObstacle_InfraRed()
{
  //FC-51 Infrared sensor
  InfraL = digitalRead( pin_InfraL );
  InfraR = digitalRead( pin_InfraR );

  Serial.print(InfraL);
  Serial.print(" -- I -- ");
  Serial.println(InfraR);
}
```

2. 4 륜 구동 자동차


외부 전원 +
외부 전원 -
L298N 모터쉴드
Arduino 또는 쉴드
ENA D6
IN1 D7
IN2 D3
IN3 D4
IN4 D2
ENB D5
MOTORA
MOTORB
ENA, IN1, IN2,
G, V, S 중에서 S 라인에 연결
D7, D6,
Sensor Shield V4.0
DIGITAL
ANALOG IN
POWER
RESET
PWR
IIC COM
COM/IIC

앞에서 기술한 것처럼

현재 4 륜 모터는 왼쪽을 IN3, IN4, ENB(D2, D4, D5)로 제어하고

오른쪽을 ENA, IN1, IN2(D6, D7, D3)로 제어한다.

ENA, ENB에 가까운 쪽으로 LOW, HIGH 를 주면 정회전

HIGH, LOW 를 주면 역회전

ENA 에 아날로그 신호를 출력하여 속도를 제어하고

IN1 에 LOW, IN2 에 HIGH를 출력하면 CW로 회전하고

IN1 에 HIGH, IN2 에 LOW를 출력하면 CCW로 회전한다.

ENB 에 아날로그 신호를 출력하여 속도를 제어하고

IN4 에 LOW, IN3 에 HIGH를 출력하면 CW로 회전하고

IN4 에 HIGH, IN3 에 LOW를 출력하면 CCW로 회전한다.

		오른쪽 바퀴			왼쪽 바퀴		
모터 모듈		ENA	IN1	IN2	IN3	IN4	ENB
아두이노 핀		D6	D7	D3	D4	D2	D5
방향	정방향	전원	LOW	HIGH	HIGH	LOW	전원
	역방향	전원	HIGH	LOW	LOW	HIGH	전원

따라서 전진하기 위해서는 왼쪽은 정방향, 오른쪽은 역방향을 주어야 하므로

오른쪽 바퀴			왼쪽 바퀴		
ENA	IN1	IN2	IN3	IN4	ENB
D6	D7	D3	D4	D2	D5
전원	HIGH	LOW	HIGH	LOW	전원

【1】전진

최대속력으로 전진하여 보자.

최대속력으로 전진하기 위해서는 출력값을 255 를 주어야 하고

방향은 EN와 가까운 쪽을 기준으로 왼쪽은 LOW, HIGH 오른 쪽은 HIGH, LOW을 주어야 한다.

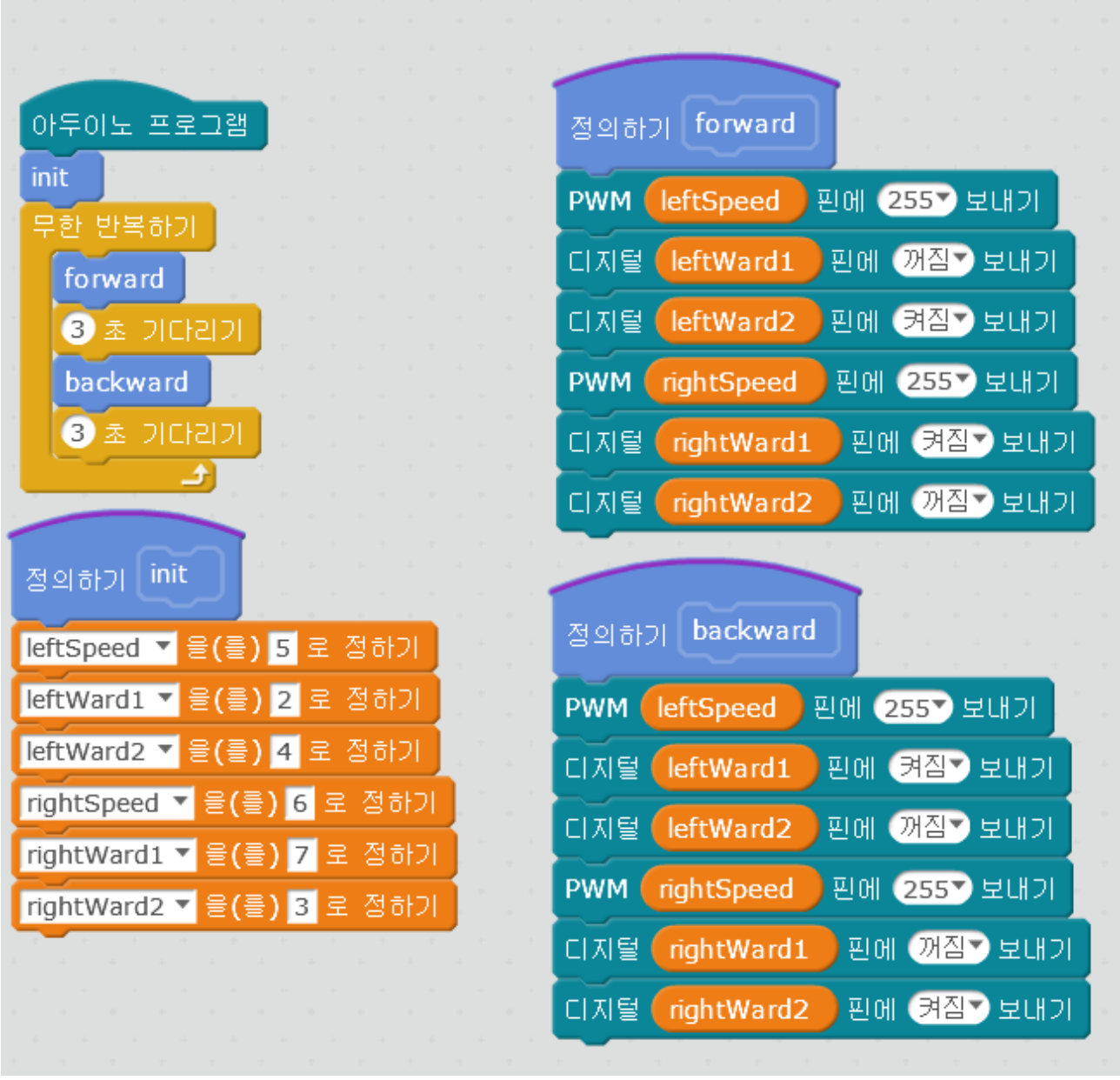
따라서 D5=255, D2=LOW, D4=HIGH, D6=255, D7=HIGH, D3=LOW로 설정하여야 한다.

```
int leftSpeed;
int leftWard1;
int leftWard2;
int rightSpeed;
int rightWard1;
int rightWard2;

void setup(){
    leftSpeed = 5;
    leftWard1 = 2;
    leftWard2 = 4;
    rightSpeed = 6;
    rightWard1 = 7;
    rightWard2 = 3;

    pinMode(leftSpeed,OUTPUT);
    pinMode(leftWard1,OUTPUT);
    pinMode(leftWard2,OUTPUT);
    pinMode(rightSpeed,OUTPUT);
    pinMode(rightWard1,OUTPUT);
    pinMode(rightWard2,OUTPUT);
}

void loop(){
    analogWrite(leftSpeed,255);
    digitalWrite(leftWard1,0);
    digitalWrite(leftWard2,1);
```

```
    analogWrite(rightSpeed,255);
    digitalWrite(rightWard1,1);
    digitalWrite(rightWard2,0);

}
```

【2】 전진과 후진

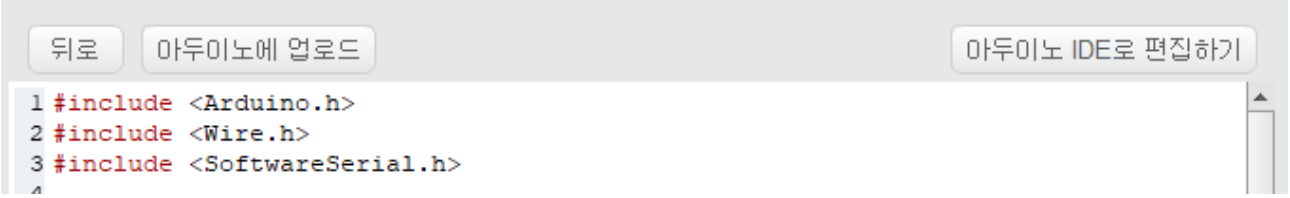
3 초 전진하고 곧 바로 3 초 후진하여 보자.



이와 같이 하면 초기화하고 전진하고 후진하고자 반복될 것 처럼 보입니다. 이 것을 아두이노에 업로드하면 다음과 같은 에러가 발생합니다.

```
 (x86)\mBlock\Arduino/project_forwardBackward5_3.ino:21: first defined
here
collect2.exe: error: ld returned 1 exit status
哪颇老 坷幅 惯积.
```

스크래치 기반의 mblock로 프로그래밍을 하면 그 에러 원인을 찾아 볼 수 가 없을 것입니다.



[아두이노 IDE로 편집하기]를 클릭하여 본래 C언어 소스프로그램을 살펴보면

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>


double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
void init();
double leftSpeed;
double leftWard1;
double leftWard2;
```

```
double rightSpeed;
double rightWard1;
double rightWard2;
void forward();
void backward();


void init()
{
    leftSpeed = 5;
    leftWard1 = 2;
    leftWard2 = 4;
    rightSpeed = 6;
    rightWard1 = 7;
    rightWard2 = 3;
}

void forward()
{
    analogWrite(leftSpeed,255);
    digitalWrite(leftWard1,0);
    digitalWrite(leftWard2,1);
    analogWrite(rightSpeed,255);
    digitalWrite(rightWard1,1);
    digitalWrite(rightWard2,0);
}

void backward()
{
    analogWrite(leftSpeed,255);
    digitalWrite(leftWard1,1);
    digitalWrite(leftWard2,0);
    analogWrite(rightSpeed,255);

    digitalWrite(rightWard1,0);
    digitalWrite(rightWard2,1);
```

```
}


void setup(){
    init();

    pinMode(leftSpeed,OUTPUT);
    pinMode(leftWard1,OUTPUT);
    pinMode(leftWard2,OUTPUT);
    pinMode(rightSpeed,OUTPUT);
    pinMode(rightWard1,OUTPUT);
    pinMode(rightWard2,OUTPUT);
}

void loop(){

    forward();
    _delay(3);
    backward();
    _delay(3);

    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){

}
```

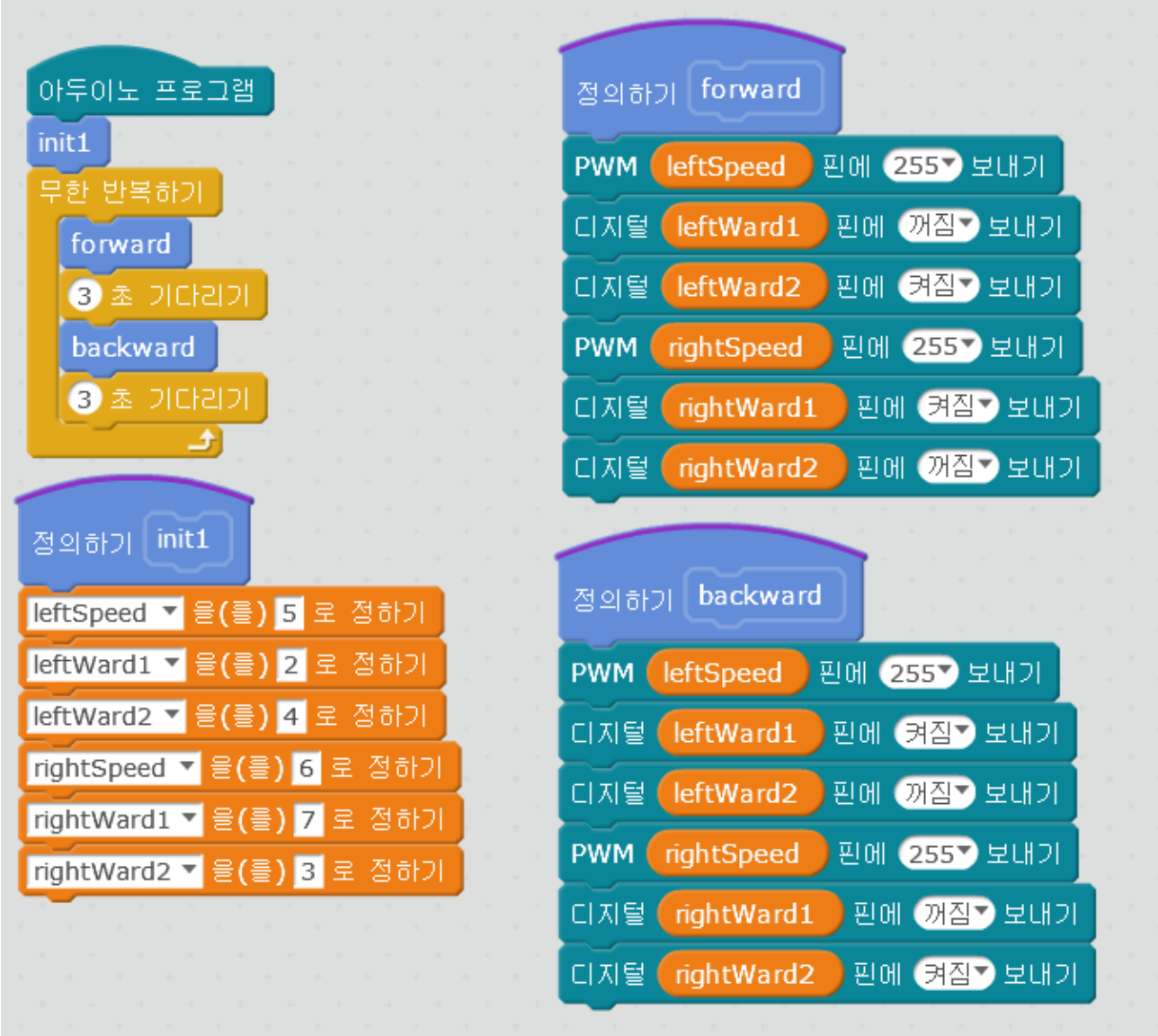
컴파일 하면 다음과 같은 에러가 발생합니다. 특히 multiple definition of `init' 에러를 보면 init가 중복해서 사용했다는 것을 알 수 있습니다. 즉, 우리가 작성한 프로그램에서는 init가 보이지 않지만 이미 아두이노에서는 init라는 함수명이 들어 있다는 것을 알 수 있습니다. 따라서 init함수명은 쓸 수 없다고 생각하면 됩니다.

```
C:\Users\ojk\AppData\Local\Temp\build5613543336795115882.tmp/core.a(wirin
g.c.o): In function `init':
C:\Program Files
(x86)\mBlock\Arduino\hardware\arduino\avr\cores\arduino/wiring.c:247:
multiple definition of `init'
project_forwardBackward5_3.cpp.o:C:\Program Files
(x86)\mBlock\Arduino/project_forwardBackward5_3.ino:21: first defined
here
c:/program files
(x86)/mblock/arduino/hardware/tools/avr/bin/../lib/gcc/avr/4.8.1/../../..
/../avr/bin/ld.exe: Disabling relaxation: it will not work with multiple
definitions
collect2.exe: error: ld returned 1 exit status
컴파일 오류 발생.
```

이제 init 이름을 init_01 로 바꾸어서 아두이노에 업로드합니다.

```
void init1();
void forward();
void backward();

double leftSpeed;
double leftWard1;
double leftWard2;
double rightSpeed;
double rightWard1;
double rightWard2;


void init1()
{
    leftSpeed = 5;
    leftWard1 = 2;
    leftWard2 = 4;
    rightSpeed = 6;
    rightWard1 = 7;
    rightWard2 = 3;
}

void forward()
{
    analogWrite(leftSpeed,255);
    digitalWrite(leftWard1,0);
    digitalWrite(leftWard2,1);
    analogWrite(rightSpeed,255);
    digitalWrite(rightWard1,1);
    digitalWrite(rightWard2,0);
}

void backward()
```

```
{
    analogWrite(leftSpeed,255);
    digitalWrite(leftWard1,1);
    digitalWrite(leftWard2,0);
    analogWrite(rightSpeed,255);
    digitalWrite(rightWard1,0);
    digitalWrite(rightWard2,1);
}

void setup(){
    init1();
    pinMode(leftSpeed,OUTPUT);
    pinMode(leftWard1,OUTPUT);
    pinMode(leftWard2,OUTPUT);
    pinMode(rightSpeed,OUTPUT);
    pinMode(rightWard1,OUTPUT);
    pinMode(rightWard2,OUTPUT);
}


void loop(){
  forward();
  delay(3000);
  backward();
  delay(3000);
}
```


아두이노 프로그램
init1
무한 반복하기
forward
3 초 기다리기
backward
3 초 기다리기
정의하기 init1
leftSpeed 을(를) 5 로 정하기
leftWard1 을(를) 2 로 정하기
leftWard2 을(를) 4 로 정하기
rightSpeed 을(를) 6 로 정하기
rightWard1 을(를) 7 로 정하기
rightWard2 을(를) 3 로 정하기
정의하기 forward
PWM leftSpeed 핀에 255 보내기
디지털 leftWard1 핀에 꺼짐 보내기
디지털 leftWard2 핀에 켜짐 보내기
PWM rightSpeed 핀에 255 보내기
디지털 rightWard1 핀에 켜짐 보내기
디지털 rightWard2 핀에 꺼짐 보내기
정의하기 backward
PWM leftSpeed 핀에 255 보내기
디지털 leftWard1 핀에 켜짐 보내기
디지털 leftWard2 핀에 꺼짐 보내기
PWM rightSpeed 핀에 255 보내기
디지털 rightWard1 핀에 꺼짐 보내기
디지털 rightWard2 핀에 켜짐 보내기