발간등록번호
충북교육연구정보원 2022-00

2022. 토요 정보아카데미 워크북

{ 아두이노 메이킹 }

충청북도교육연구정보원
Chungcheongbuk-do Education Research & Information Institute

아두이노 메이킹

발　행 | 2022년 8월 30일
저　자 | 박정진
발행인 | 백우정(충청북도 교육연구정보원 원장)
펴낸곳 | 충북교육연구정보원
주　소 | 충북 청주시 서원구 청남로 1931(산남동 4-3)
전　화 | 1670-8316
이메일 | cberi@cberi.co.kr

ISBN | 979-11-372-0000-0

www.cberi.go.kr

아두이노 메이킹

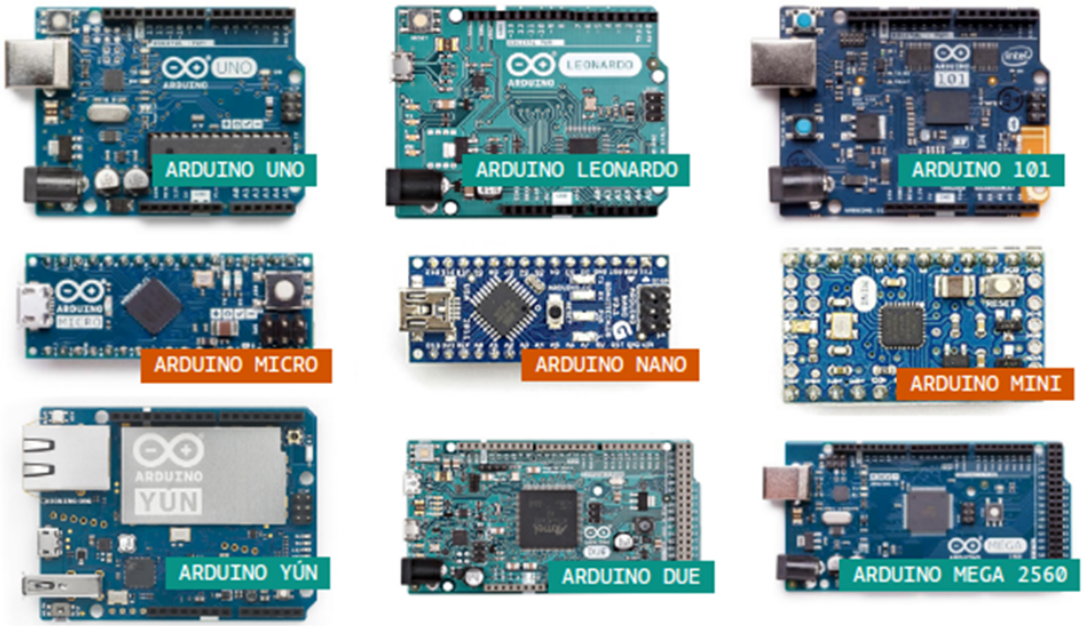
박정진 저

CONTENT

제1장 준비하기

1. 피지컬 컴퓨팅 하드웨어

가. 아두이노 시리즈



1) Arduino UNO

Atmel사의 ATmega328 칩을 사용한 마이크로 컨트롤러이다. UNO는 이탈리어어로 하나를 뜻하는 말로 첫 번째 아두이노라는 뜻이다. 가장 기본이 되는 제품으로 교육용으로 많이 쓰이며 초보자가 쓰기에 가장 좋은 기종으로 크기가 5.3cm x 7.7cm 정도로 손바닥보다 작다.

UNO의 미니 버전으로 동일한 칩을 사용하는 아두이노 나노가 있다.

2) Arduino Leonardo

Atmel사의 ATmega32u4 칩을 사용한 마이크로 컨트롤러이다. 다른 보드와는 달리 USB와 직접 연결 가능하다. 이 보드를 마우스나 키보드와 같은 HID(Human Interface Deive - 사람이 컴퓨터를 작동하는데 쓰이는 장치)로 인식시킬 수 있어 마우스나 키보드의 기능을 수행하는 것이 가능하다.

Leonardo의 미니 버전으로 동일한 칩을 사용하는 아두이노 마이크로가 있다.

3) Arduino Mega2560

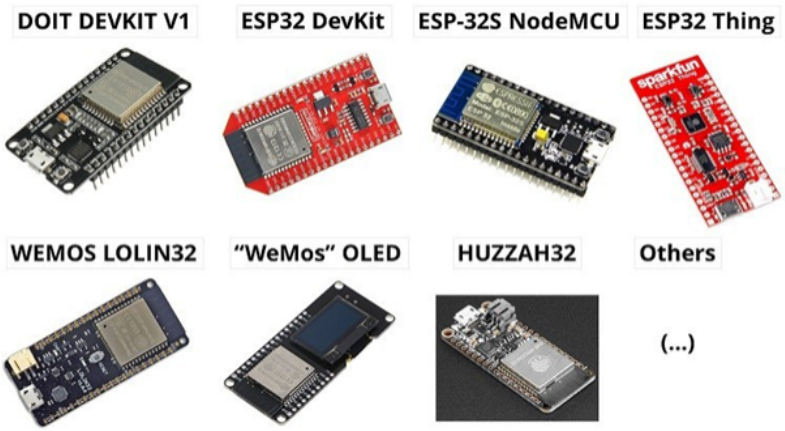
Atmel사의 ATmega2560 아두이노를 사용하다 보면 늘 아쉬운 것이 연결할 포트의 개수가 부족해진다는 점이다. 아두이노 우노는 6개, 디지털 15개로 총 20개의 연결 단자를 제공한다. 아두이노 메가는 아날로그 16개, 디지털 54개로 총 70개의 포트를 제공한다. 연결할 핀이 부족할 수 없을 정도로 많다.

나. Xtensa 시리즈

1) ESP8266

ESP8266 Node MCU는 ESP-12 모듈에 USB와 아두이노IDE환경 지원 등을 추가해서 아두이노의 한 종류처럼 개발된 모듈으로서 IOT(Internet Of Things)환경에서의 기본형 프로세서로 많이 사용된다. 즉, IOT환경을 위해서는 WIFI 통신기능이 필수인데 기존의 아두이노에 WIFI장치를 부착하는 것보다 NodeMCU 하나를 사용하는 것이 편리하다.

2) ESP32



ESP32는 중국 Espressif Systems라는 회사에서 ESP8266이 인기를 얻고나서 후속작으로 내놓은 상위 기종으로 WIFI뿐만 아니라 Bluetooth 4.2도 기본으로 지원하고 처리속도도 빨라졌다. 무엇보다도 ESP8266에서 부족했던 범용 입출력 핀이 대폭 늘어나서 활용 범위가 커진 것이 장점이다. 특히 속도가 2배 정도 빨라졌고, GPIO가 17개에서 36개로 두 배이상으로 늘어났으며 Touch센서, 온도센서 기능들이 추가되었다.

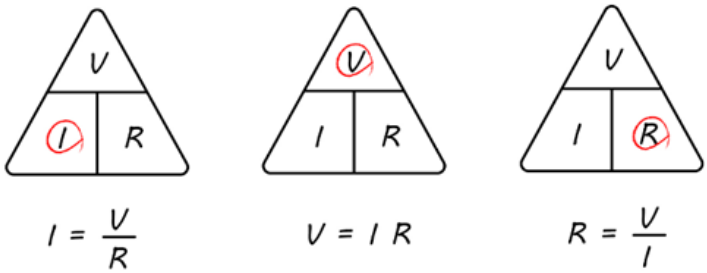
다. 개발보드 비교

구분	Arduino UNO	ESP8266	ESP32
MCU	ATmega328P 8bit	Xtensa Single-Code 32bit L106	Xtensa Dual-Core 32bit LX6
Frequency	16Mhz	80Mhz	160Mhz
Operating Voltage	5.0V	3.3V	3.3V
SRAM	2KB	160kB	512kB
FLASH	32KB	16MB	16MB
GPIO	14	17	36
802.11 b/g/n Wi-Fi	None	Yes, HT20	Yes, HT40
Bluetooth	None	None	Bluetooth 4.2 and BLE
HW / SW PWM	5/0	0/8	1/16
SPI/I2C/I2S/UART/CAN	1/1/0/1/1	2/1/2/2/0	4/2/2/2/1
ADC pins	6 (10-bit)	1 (10-bit)	18 (12bit)
DAC pins	0	0	3
Touch Sensor	None	None	Yes
Temperature Sensor	None	None	Yes

보통 Arduino를 통해 피지컬컴퓨팅을 처음 접하고 배우는 경우가 많다. Arduino는 오픈 H/W와 오픈 S/W 정책을 통해 전 세계의 수많은 사용자와 개발자로부터 전폭적인 지원을 받고 있으며 이렇게 형성된 광범위한 커뮤니티는 프로젝트 개발 중에 발생할 수 있는 온갖 종류의 문제를 해결하는 강력한 우군이 되어주기도 한다. 하지만 매우 적은 용량의 SRAM(2KB)과 느린 속도(16Hhz)은 자유로운 메이킹을 가로막는 방해요인이 되기도 하는데, 예를 들어 16x16 3색 LED 매트릭스를 연결하여 전광판이라도 만드려 한다면 한글 폰트조차 램에 올릴 수 없어 갖은 꼼수가 동원되어야 하고, Arduino를 IOT 노드로 사용하려면 갖은 통신 모듈을 덕지덕지 붙여야 목적을 달성할 수 있다. 따라서 전문적인 메이커 활동을 위해서는 아두이노만큼 저렴하면서도 고성능 낼 수 있고 통신 기능을 함께 제공하는 ESP32 등이 좋은 대안이 될 수 있다.

2. 전자회로 기본지식

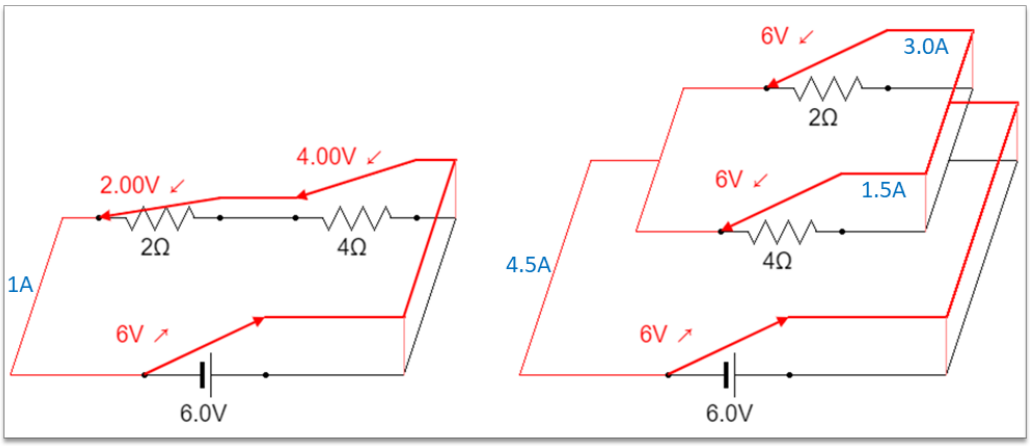
가. 옴의 법칙



$I = \frac{V}{R}$ $V = IR$ $R = \frac{V}{I}$

이 공식에서 주목해야 할 것은 전류, 전압, 저항의 관계이다. 우리가 조정할 수 있는 것은 전압과 저항이다. 그 결과로 회로에 흐르는 전하의 양이 달라집니다. 즉, 전류가 커지고 작아진다. 이를 표현하고 있는 공식이 I = V / R 이다.

나. 전압·저항·전류 관계



$R = 2 + 4 = 6$

$I = \frac{V}{R} = \frac{6}{6} = 1$

$R = \frac{2 \times 4}{2+4} = \frac{8}{6} = \frac{4}{3}$

$I = \frac{V}{R} = \frac{6}{(\frac{4}{3})} = 4.5$

1) 저항의 직렬연결

전류의 세기는 각 저항에 흐르는 전류의 세기는 회로 전체에 흐르는 전류의 세기 같으며, 각각의 저항에 걸리는 전압은 전체에 걸리는 전합의 크기와 같으며 전체 전압은 각 저항에 걸리는 전압의 합과 같다

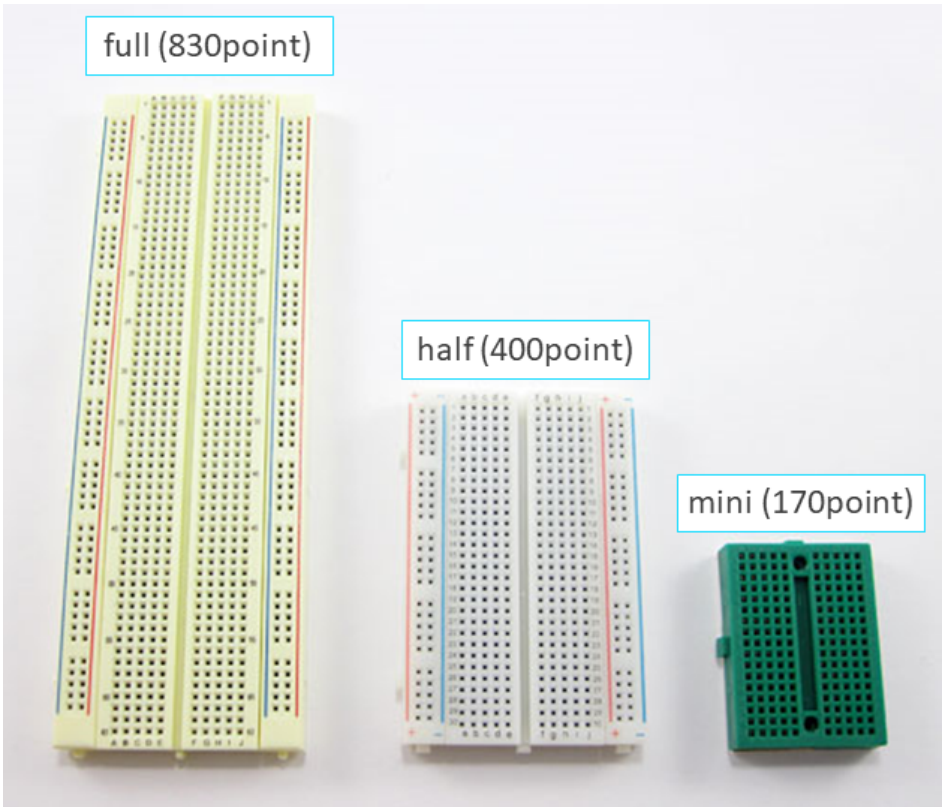
2) 저항의 병렬연결

전류가 나뉘어 흐르므로 전류의 합은 전체 전류와 동일하며, 저항이 전압원에 각각 연결되므로 각 저항에 걸리는 전압은 같고, 전체 전압은 각 저항에 걸리는 전압과 같다.

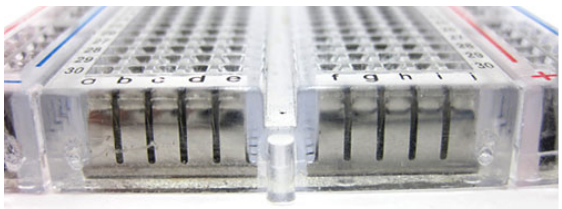
다. 브레드보드(breadboard)

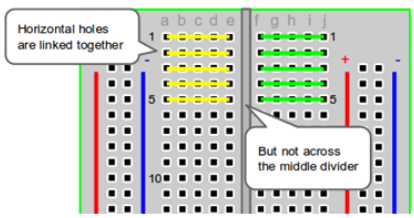
브레드보드를 사용하는 이유는 전기 인두를 이용해 납땜을 하지 않아도 전자회로를 구성할 수 있기 때문이다.

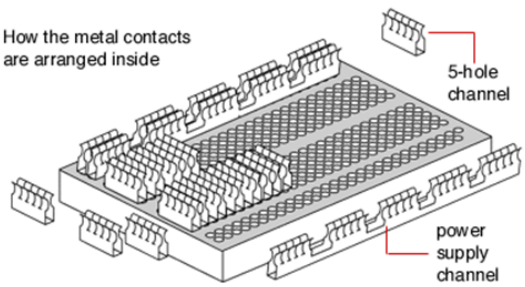
1) 크기별 구분

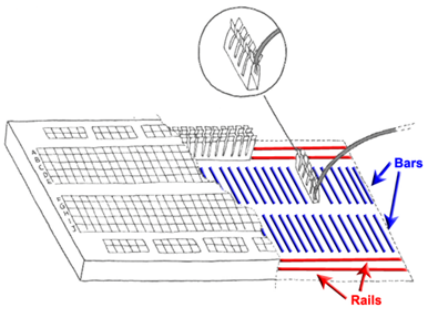


2) 내부구조



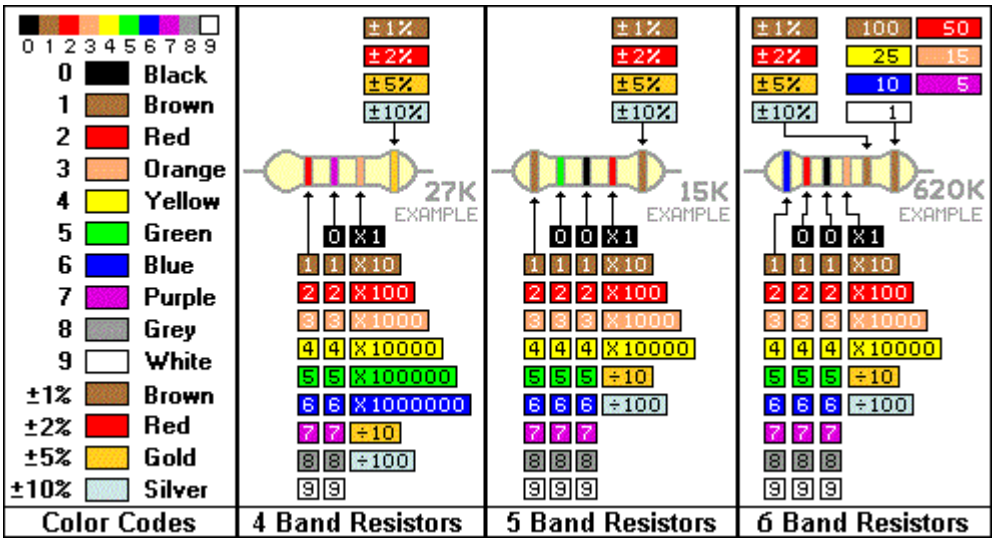






라. 저항의 컬러코드

그림과 같이 저항의 색상 코드는 4자리 색상, 5자리 색상, 6자리 색상 코드에 따라 구분된다. 이 색상 코드는 저항의 값 숫자, 배수, 오차, 온도계수 등의 정보를 담고 있다.



1) 4자리 색상 코드 읽기

4자리 색상 코드는 첫 번째 띠와 두 번째 띠는 두 자리의 숫자를 나타내고, 세 번째 띠는 배율, 네 번째 띠는 오차를 나타내는 코드이다. 보통 앞뒤를 구별하기 위해, 세 번째 띠까지 간격이 좁고, 네 번째 띠는 간격이 넓게 표시 한다. 하지만 간격이 동일한 경우, 앞뒤를 구별하기 위해 오차는 금색, 은색, 무색으로 나타내는 경우가 많다.

예) 빨강, 빨강, 검정, 금색 : 22 x 1 = 22Ω, 오차 ±5%

2) 5자리 색상 코드 읽기

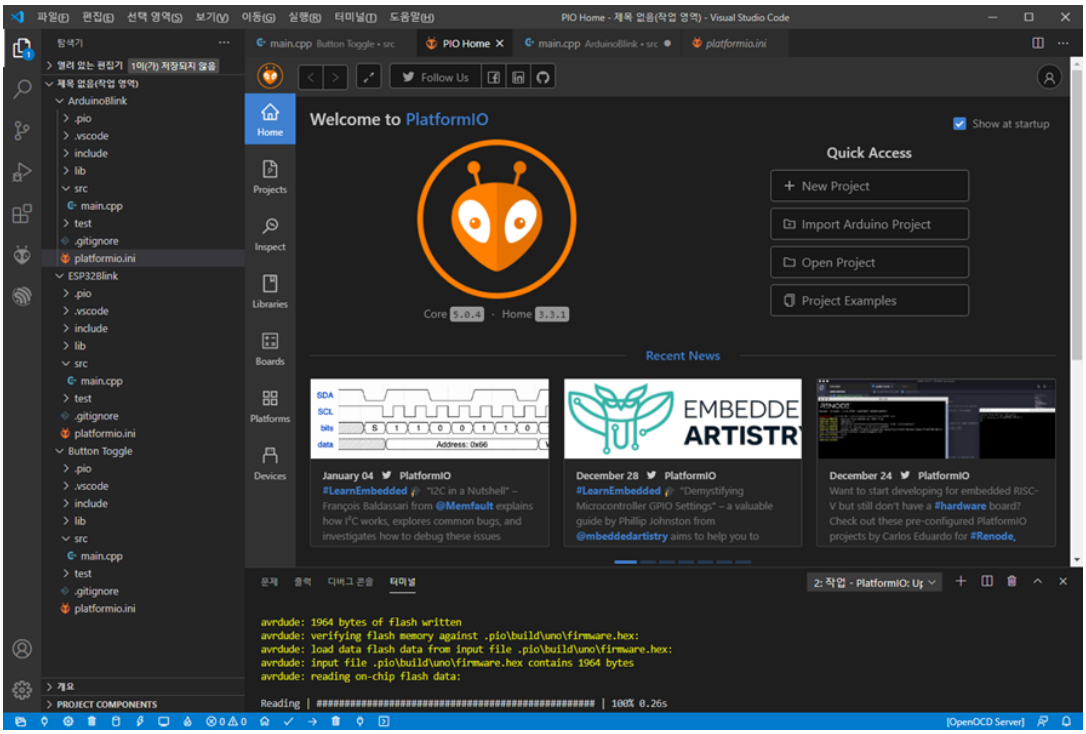
5자리 색상 코드는 정밀 저항에서 사용하는 코드로, 첫 번째, 두 번째, 세 번째 띠는 셋째 자리의 숫자이고, 네 번째 띠는 배수, 다섯 번째 띠는 오차를 나타내는 코드이다. 또한 앞뒤를 구별하기 위해, 첫 번째 띠부터 네 번째 띠까지 간격이 좁고, 다섯 번째 띠는 간격이 넓게 표시 한다.

예) 빨강, 빨강, 검정, 주황, 갈색 : 220x1000 = 220KΩ, 오차 ±1%

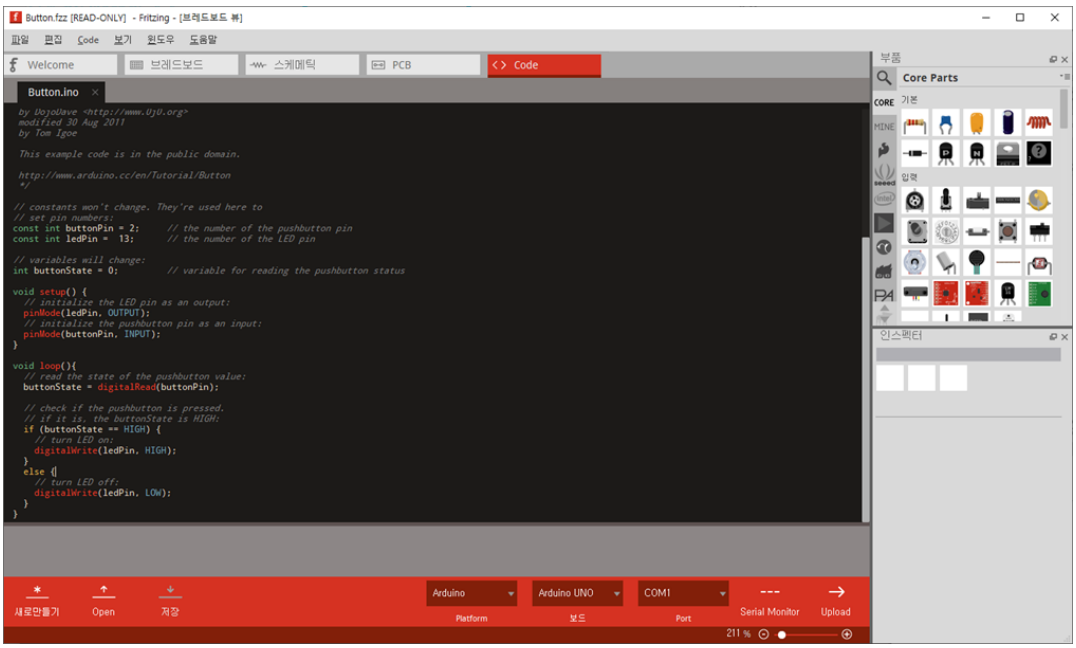
3. 개발환경

아두이노 개발 도구는 전통적인 아두이노 스케치 이외에도 다른 여러 프로그램을 사용할 수 있다.

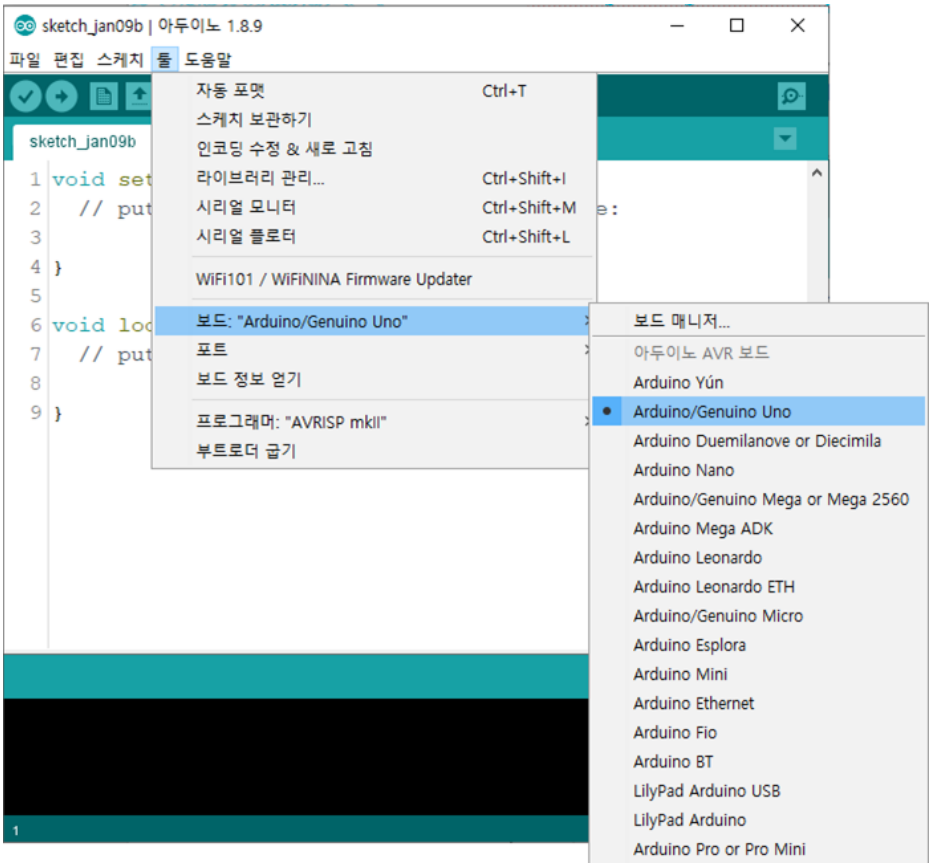
가. VScode + PlatformIO

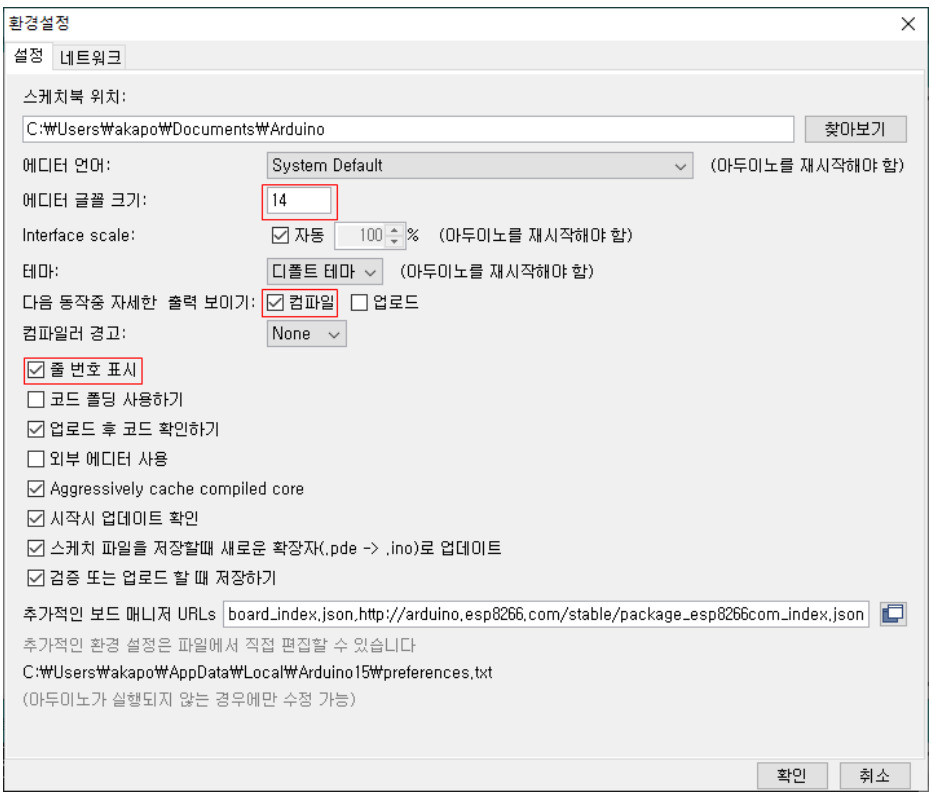


나. Fritzing



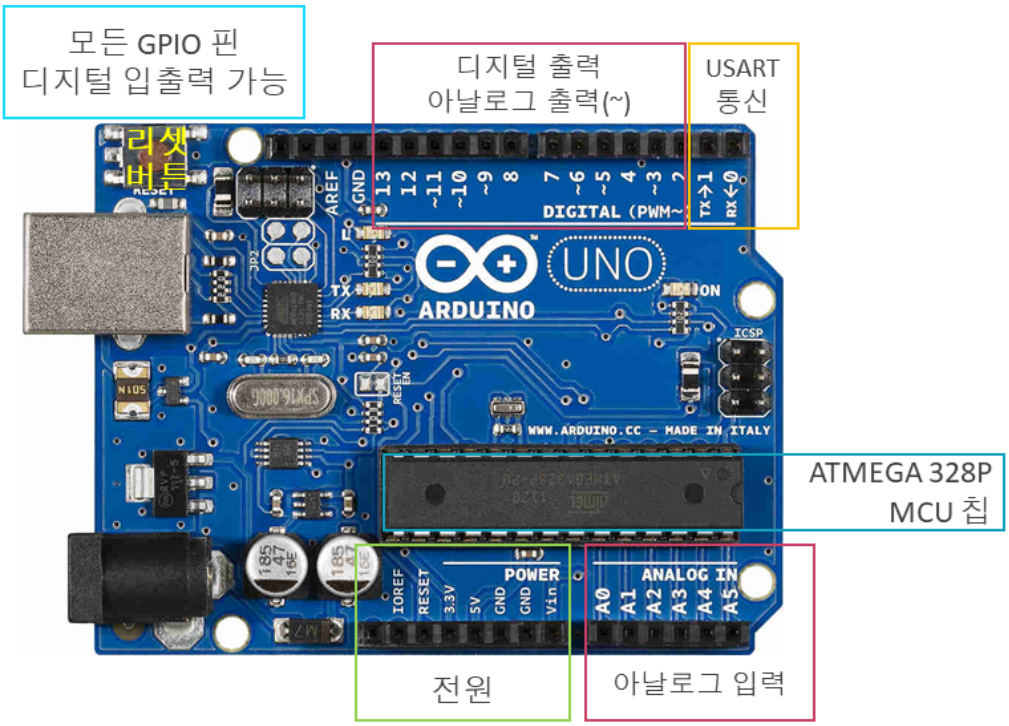
다. 아두이노 스케치


sketch_jan09b | 아두이노 1.8.9
파일 편집 스케치 툴 도움말
자동 포맷 Ctrl+T
스케치 보관하기
인코딩 수정 & 새로 고침
라이브러리 관리... Ctrl+Shift+I
시리얼 모니터 Ctrl+Shift+M
시리얼 플로터 Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
보드: "Arduino/Genuino Uno"
포트
보드 정보 얻기
프로그래머: "AVRISP mkII"
부트로더 굽기
보드 매니저...
아두이노 AVR 보드
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino/Genuino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini


환경설정
설정 네트워크
스케치북 위치:
C:₩Users₩akapo₩Documents₩Arduino
찾아보기
에디터 언어: System Default (아두이노를 재시작해야 함)
에디터 글꼴 크기: 14
Interface scale: 자동 100 % (아두이노를 재시작해야 함)
테마: 디폴트 테마 (아두이노를 재시작해야 함)
다음 동작중 자세한 출력 보이기: 컴파일 업로드
컴파일러 경고: None
줄 번호 표시
코드 폴딩 사용하기
업로드 후 코드 확인하기
외부 에디터 사용
Aggressively cache compiled core
시작시 업데이트 확인
스케치 파일을 저장할때 새로운 확장자(.pde -> .ino)로 업데이트
검증 또는 업로드 할 때 저장하기
추가적인 보드 매니저 URLs: board_index.json,http://arduino.esp8266.com/stable/package_esp8266com_index.json
추가적인 환경 설정은 파일에서 직접 편집할 수 있습니다
C:₩Users₩akapo₩AppData₩Local₩Arduino15₩preferences.txt
(아두이노가 실행되지 않는 경우에만 수정 가능)
확인
취소

4. 핀 아웃

가. 아두이노 UNO의 핀 아웃

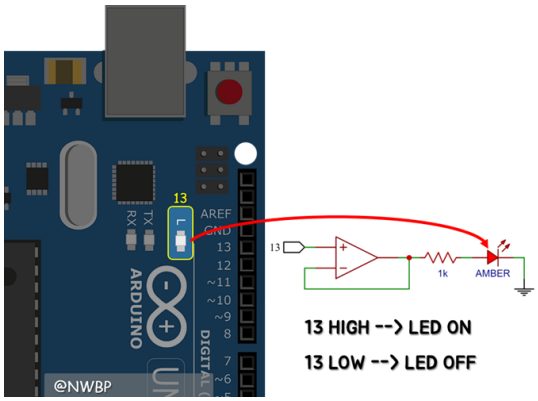


- GPIO : General Purpose Input Output
- 입력 : 외부의 자료가 아두이노 안으로 들어오는 것.
- 출력 : 아두이노 내부의 자료가 밖으로 나가는 것.
  - 디지털 출력 : 출력 값이 0(LOW, 0V), 1(HIGH, 5V) 이렇게 두 가지 상태만 있는 출력.
  - 아날로그 출력 : 출력 값을 0% ~ 100% 까지 단계별로 조절할 수 있는 출력.
- USART: Universal Synchronous Asynchronous Receiver Transmitter

제2장 디지털 출력

1. LED 켜기

가. Built in LED 제어

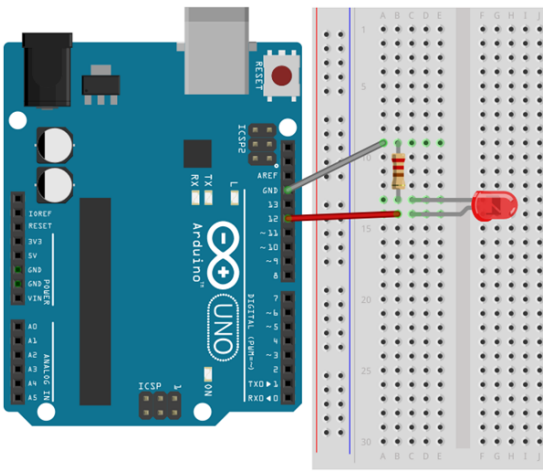



```
//#define LED_BUILTIN 13

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(500);
  digitalWrite(LED_BUILTIN, LOW);
  delay(500);
}
```

13번 포트는 아두이노 내부 기판의 LED와 연결되어 있다. 그림에서 보이는 저 LED를 내장(built in) LED라 부르며 13번 포트를 제어하여 LED를 on/off 가능하다. 아두이노에서는 LED_BUILTIN 이라는 상수로 등록되어 있으며 값은 13이다.

나. LED on/off

다음 그림은 아두이노에서 교재에서 흔하게 볼 수 있는 LED 연결 회로이다. LED는 20mA가 적정 전류이며 50mA가 넘어가면 LED에 손상이 발생하기 시작한다. 그래서 과도한 전류가 흐르는 것을 막기 위해 보통 저항과 함께 사용한다.



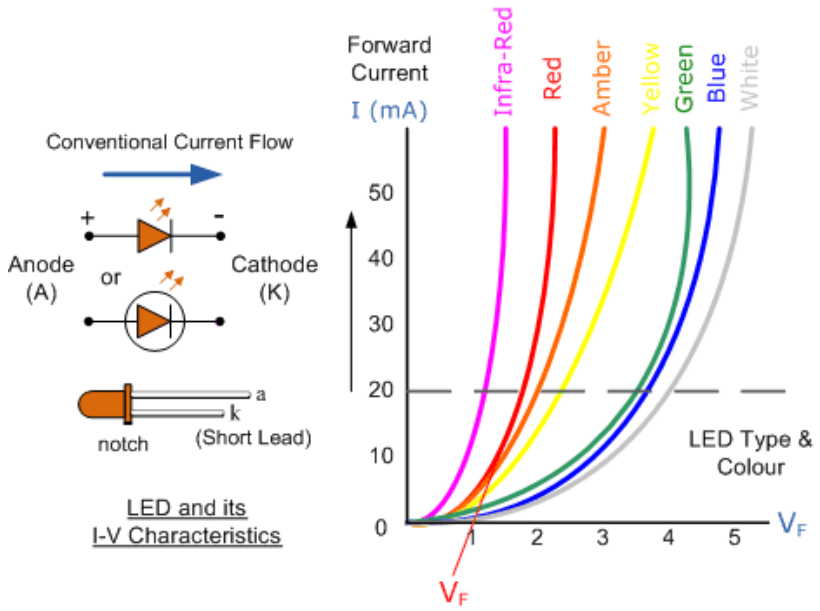
```
#define LED_R 12

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(LED_R, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  digitalWrite(LED_R, HIGH);
  delay(500);
  digitalWrite(LED_BUILTIN, LOW);
  digitalWrite(LED_R, LOW);
  delay(500);
}
```

대부분의 교재에서 220Ω에서 330Ω 정도의 저항을 사용하는데 과연 이 저항값이 어떻게 도출된 것이며, LED 개수나 색상이 바뀐다면 저항을 어떻게 바꿔줘야 하는지 LED와 연결된 저항의 정확한 사용법을 알아보자.

2. LED와 저항

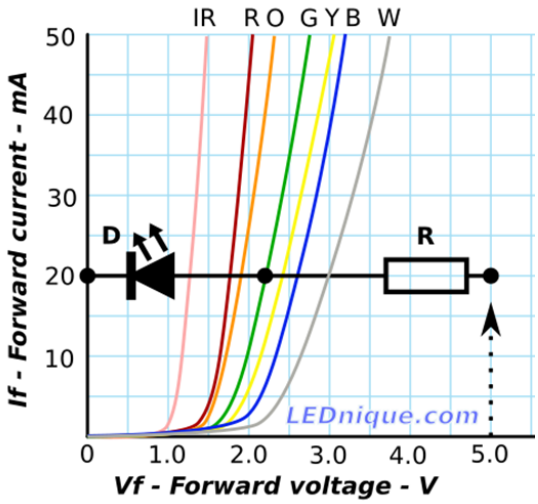
가. LED 특징

1) LED 전류 전압 그래프



LED 색상별로 LED에 전류를 흐르게 할 수 있는 V_f(순방향 전압)가 다르다. 예를 들어 20mA 전류를 기준으로 할 때 빨간색 LED는 대략 1.8V의 전압을 요구하며, 주황색, 노란색, 녹색, 파란색, 흰색 LED 순으로 더욱 높은 전압을 요구한다.

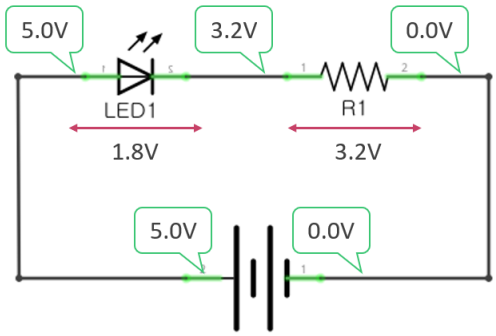
이것은 LED에 (같은 밝기의) 불을 켜기 위해 위 컬러 순서로 더 높은 전압이 필요함을 의미한다.



Color of LED	Voltage Drop (Volt)
Red	1.63 ~ 2.03
Yellow	2.10 ~ 2.18
Orange	2.03 ~ 2.10
Blue	2.48 ~ 3.7
Green	1.9 ~ 4.0
Violet	2.76 ~ 4.0
UV	3.1 ~ 4.4
White	3.2 to 3.6

나. 적정 저항 계산

1) 빨간색 LED일 경우



Color	Vf[V] at 20 mA	Material	Wavelength [nm]
Infrared	1.2	GaAs	850-940
Red	1.8	GaAsP	630-660
Amber	2	GaAsP	605-620
Yellow	2.2	GaAsP:N	585-595
Green	3.5	AlGaP	550-570
Blue	3.6	SiC	430-505
White	4	GaInN	450

▪ 적절한 저항은?

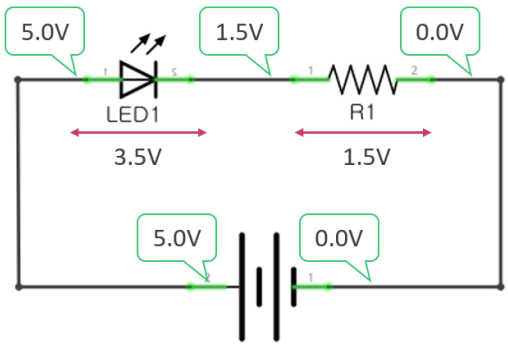
$$R = \frac{V_s - V_f}{I_f} = \frac{5.0 - 1.8}{0.02} = 160$$

▪ 40mA 흐르게 만들려면 저항은?

$$R = \frac{V_s - V_f}{I_f} = \frac{5.0 - 2.0}{0.04} = 75$$

빨간색 LED는 오른쪽 표에 따르면 V_f가 1.8V이다. 따라서 LED에서 1.8V 전압강하가 발생하고 나머지 3.2V가 저항 R1에 걸리게 된다.

2) 녹색 LED일 경우



Color	Vf[V] at 20 mA	Material	Wavelength [nm]
Infrared	1.2	GaAs	850-940
Red	1.8	GaAsP	630-660
Amber	2	GaAsP	605-620
Yellow	2.2	GaAsP:N	585-595
Green	3.5	AlGaP	550-570
Blue	3.6	SiC	430-505
White	4	GaInN	450

▪ 적절한 저항은?

$$R = \frac{V_s - V_f}{I_f} = \frac{5.0 - 3.5}{0.02} = 75$$

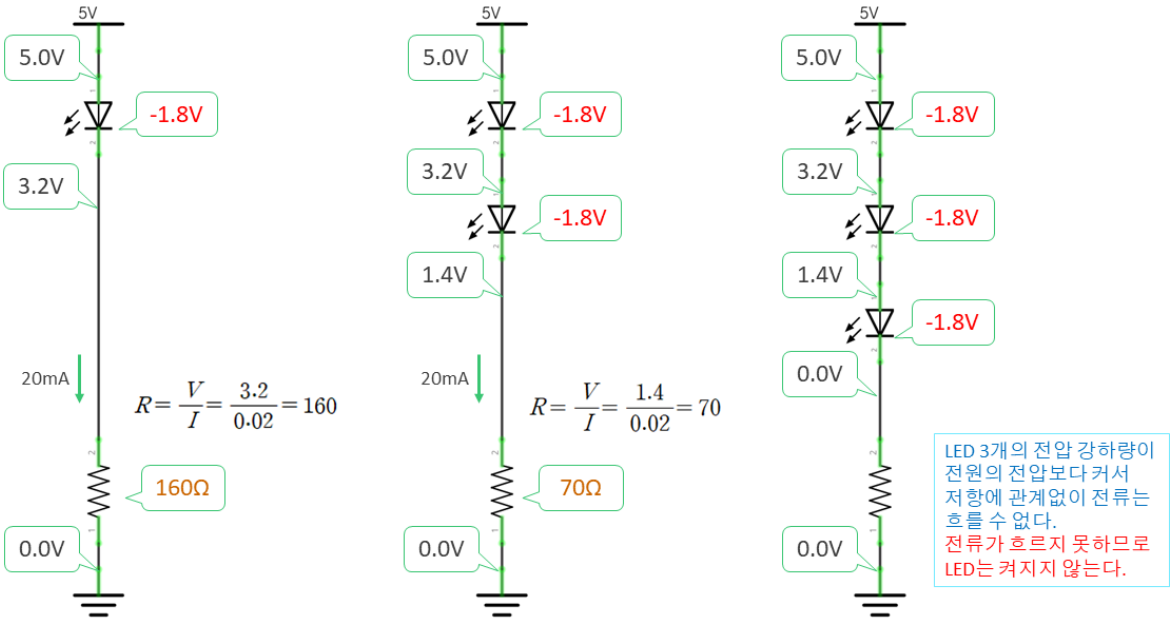
▪ 40mA 흐르게 만들려면 저항은?

$$R = \frac{V_s - V_f}{I_f} = \frac{5.0 - 4.2}{0.02} = 40$$

녹색 LED는 오른쪽 표에 따르면 V_f가 3.5V이다. 따라서 LED에서 3.5V 전압강하가 발생하고 나머지 1.5V가 저항 R1에 걸리게 된다.

3) LED 직렬 연결 시 저항

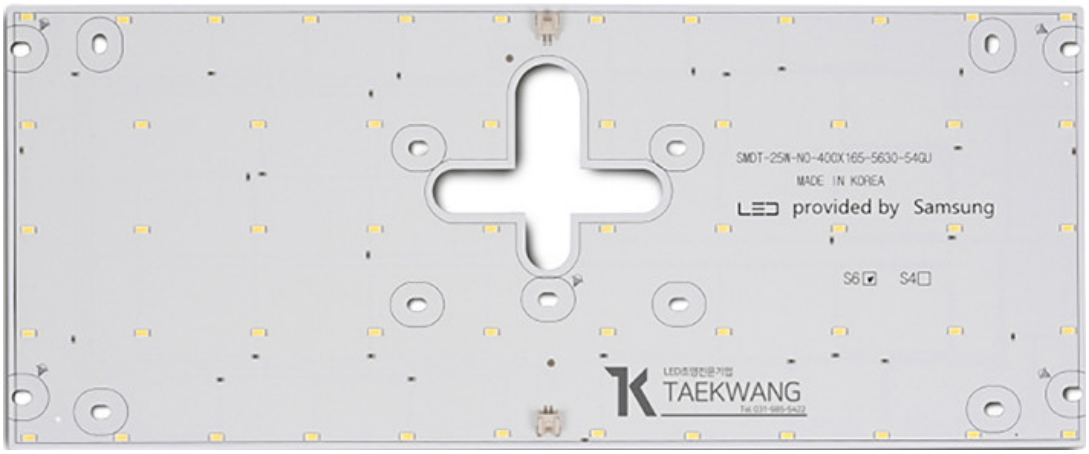
(빨간색 LED를 사용하고, 각 LED에 20㎃의 전류를 흐르도록 설계한다고 가정)



LED가 늘어날수록 전압강하가 많이 일어난다(LED를 통과할 때마다 정해진 양만큼 계속 떨어짐). 5V 회로에서는 빨간색 LED를 최대 2개까지 켤 수 있을 뿐이다. 직렬로 연결된 다수의 LED를 켜려면 높은 전압이 요구된다.

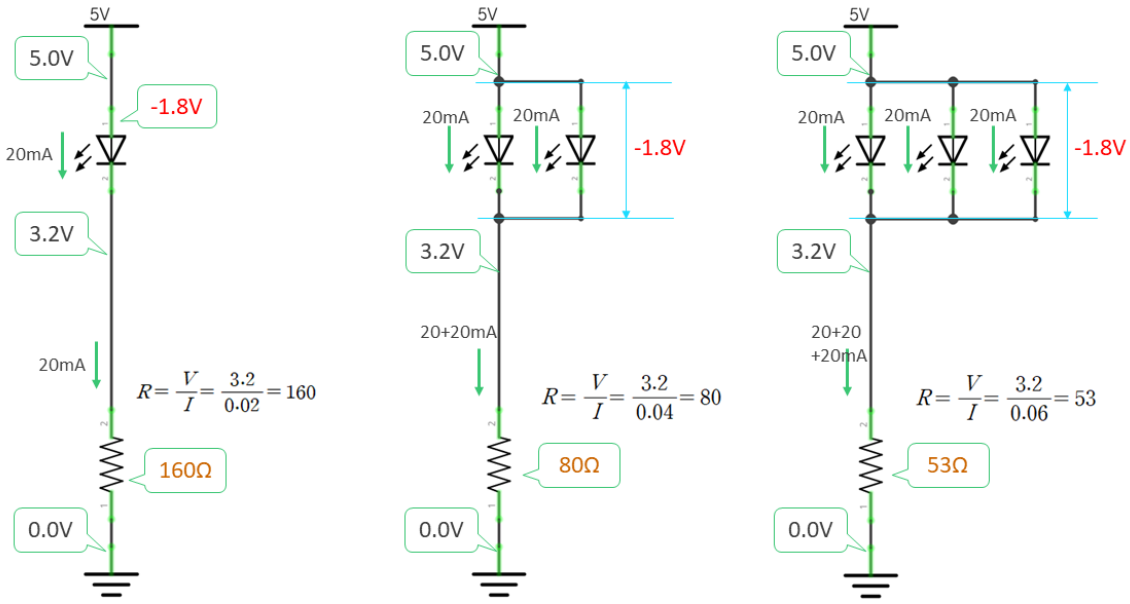
빨간 LED를 직렬로 3개 연결한 오른쪽 회로는 전원의 전압은 5V인데 LED의 전압강하 총량은 5.4V이므로 전압이 부족하여 LED를 켤 수가 없다. 이 상황에서는 저항이 문제가 아니라 전원의 전압을 더 높여 주어야 한다. 직렬로 연결된 LED 회로는 중간에 하나의 LED라도 파손(고장)되면 전체 라인의 LED가 켜지지 않는다.

실제로 우리가 사용하는 LED 조명은 직렬과 병렬을 혼합하여 사용하는데 LED 16개를 직렬로 연결한 뒤 이러한 연결을 5개 병렬로 붙이는 식이다.



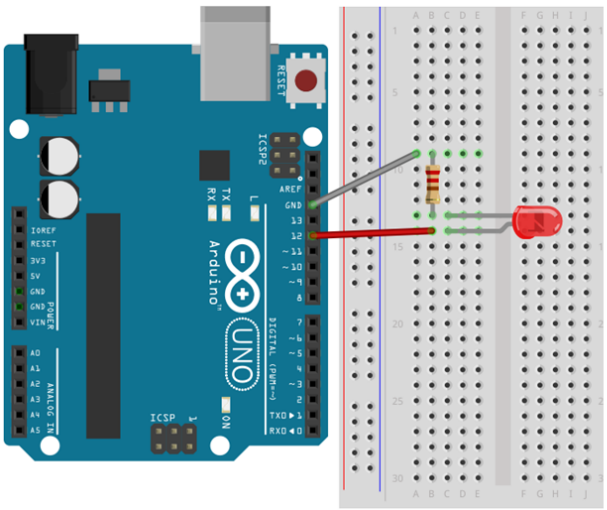
4) LED 병렬 연결 시 저항

(빨간색 LED를 사용하고, 각 LED에 20㎃의 전류를 흐르도록 설계한다고 가정)



병렬 연결 LED의 저항 계산 시 유의할 점은 각 LED를 통과하는 모든 전류가 모두 더해져서 최종 저항을 통과하게 된다는 것이다.

라. 최대 전류

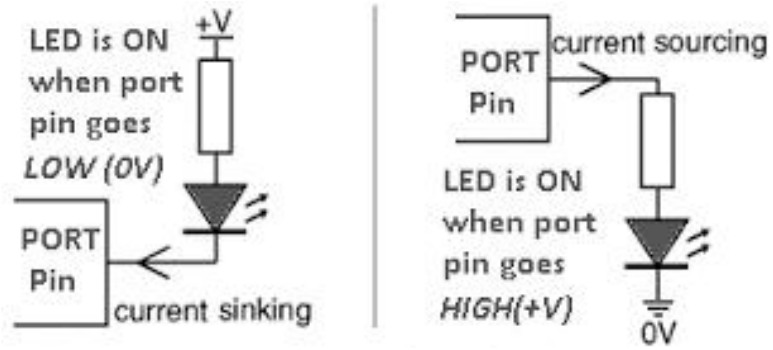


Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except RESET with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on RESET with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

저항이 없다고 무한대의 전류가 흐르는 것이 아니다. 아두이노는 각 GPIO핀 당 최대 40mA까지 전류가 나갈 수 있다. 또한 Vcc에서 GND로는 최대 200mA까지 흐를 수 있다. 이 이상의 전류가 필요하다면 별도의 외부 전원을 구축해야 한다.

3. 전류의 방향 (Source와 Sink)



Sink vs Source

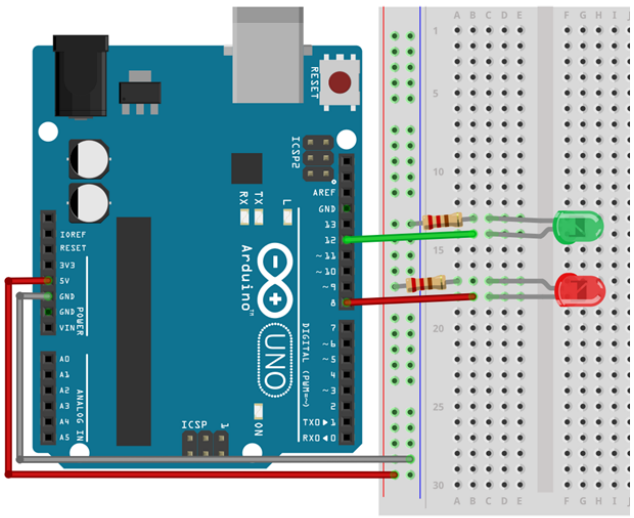
가. Source

칩에서 나가는 전류를 사용한다. HIGH일 때 ON이 되며, 이렇게 HIGH에서 활성화되는 회로를 'Active HIGH'라고 한다.

나. Sink

칩으로 들어오는 전류를 사용한다. LOW일 때 ON이 되며, 이렇게 LOW에서 활성화되는 회로를 'Active LOW'라고 한다. 전류 제한에 걸리지 않으므로 큰 전류를 필요로 할 때 사용한다.

다. 실습

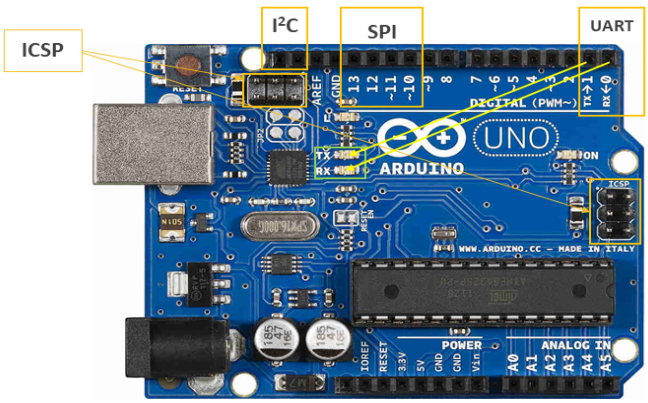


```
#define LED_R 8
#define LED_G 12

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  digitalWrite(LED_R, HIGH);
  digitalWrite(LED_G, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  digitalWrite(LED_R, LOW);
  digitalWrite(LED_G, LOW);
  delay(1000);
}
```

녹색 LED는 Source 방식으로 연결되어 12번 핀이 HIGH일 때 불이 켜진다(Active HIGH). 반면 빨간 LED는 Sink 방식으로 연결되어 8번 핀이 LOW일 때 불이 켜진다(Active LOW). 따라서 두 LED가 번갈아 가면서 불이 켜지고 꺼지고 되는데, 현재 핀의 출력 상태를 빌트인 LED가 알려주고 있다.

4. 아두이노 시리얼 통신

가. 아두이노 UNO의 통신포트



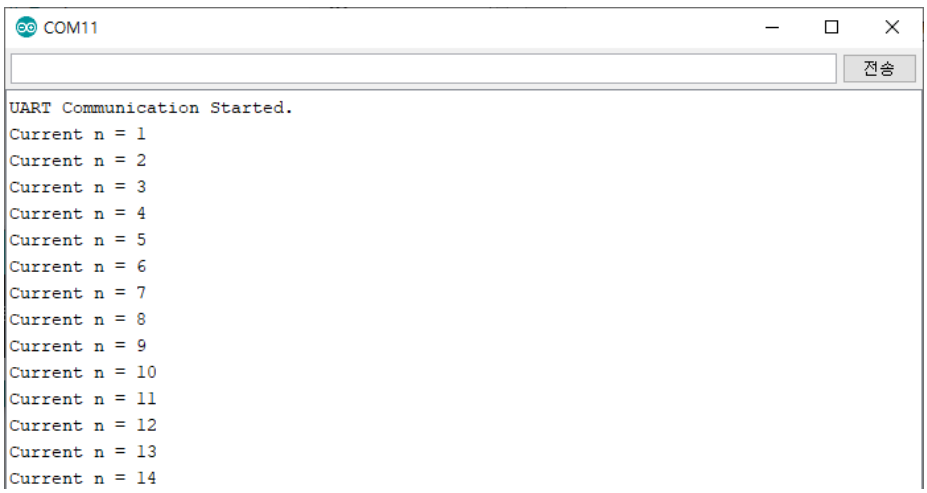
- UART : Universal Asynchronous Receiver and Transmitter
- I2C : Inter-Integrated Circuit
- SPI : Serial Peripheral Interface
- ICSP : In-Circuit Serial Programming

나. Serial 포트를 출력창으로 사용하기

1) 프로그램 소스코드

```
int n = 1;
void setup() {
    // 주의: 매번 실수가 일어나는 곳. 초기화 꼭 해야 함.
    Serial.begin(9600);
    Serial.println("UART Communication Started.");
}

void loop() {
    Serial.print("Current n = ");
    Serial.print(n++);
    Serial.print(" \n");
    delay(1000);
}
```

2) 시리얼 모니터 사용 [단축키: Ctrl+Shift+M]

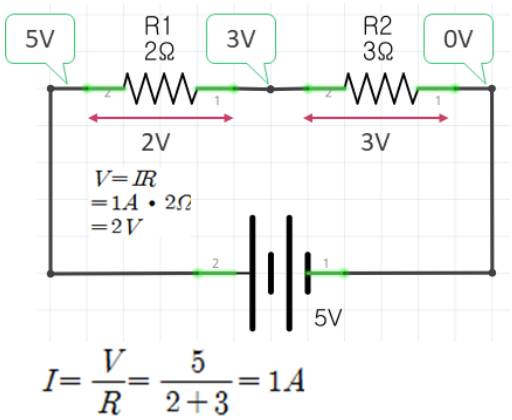


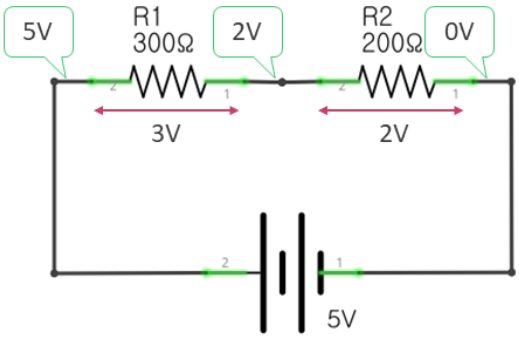
제3장 디지털 입력

1. 준비하기

가. 선수지식 - 전압강하

전류가 두 전위 사이를 흐를 때 저항을 직렬로 여러 개 연결하면 전류가 각 저항을 통과할 때마다 옴의 법칙[전압(V)= 전류(I)·저항(R)]만큼 전압이 작아져 나타나는 현상을 전압강하라 한다. 이때 전체 전압은 각각에 걸린 전압의 합이 된다.





$$I = \frac{V}{R} = \frac{5}{2+3} = 1A$$

나. pinMode(x, INPUT)

지금까지는 아두이노의 포트들을 LED를 켜는 등 회로를 동작시키기 위해 출력 모드로 사용하였다. 그러기 위해 setup() 함수에서 포트 x를 출력 모드로 설정한다는 PinMode(x, OUTPUT) 함수를 호출하였다.

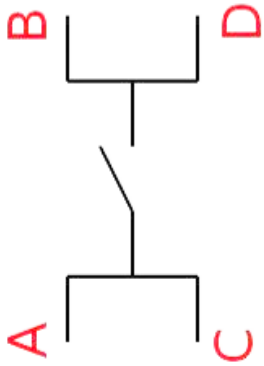
이제부터는 반대로 아두이노의 포트에서 상태(전압값)를 읽기 위해 입력 모드로 작동시켜 볼 것이다. 특정 포트(핀) x를 입력 모드로 작동시키려면 PinMode(x, INPUT) 함수를 호출하면 된다. 여기서 포트 x는 0~13까지 모든 digital 입출력 핀이 사용 가능하며 심지어 아날로그 입력을 위해 사용하는 A0~A5 핀도 디지털 입출력을 위해 사용할 수 있다.

pinMode(x, INPUT)를 호출한 뒤 digitalRead(x) 함수를 호출하여 특정 포트 x에서의 **전압값**을 읽어 디지털 신호 1 또는 0 값을 얻어낸다. 디지털 신호 1은 5V(HIGH)를 의미하며, 디지털 신호 0은 0V(LOW)를 의미한다.

여기서 유의할 점은 아두이노는 입력 모드일 때 전류를 검출하는 것이 아니라 전압을 검출한다는 것이다. 해당 핀으로 전류가 흐르면 1, 안 흐르면 0 이 입력된다고 오해하면 회로를 제대로 해석할 수 없으므로 주의하자.

2. 버튼 회로 설계

가. Tactile switch

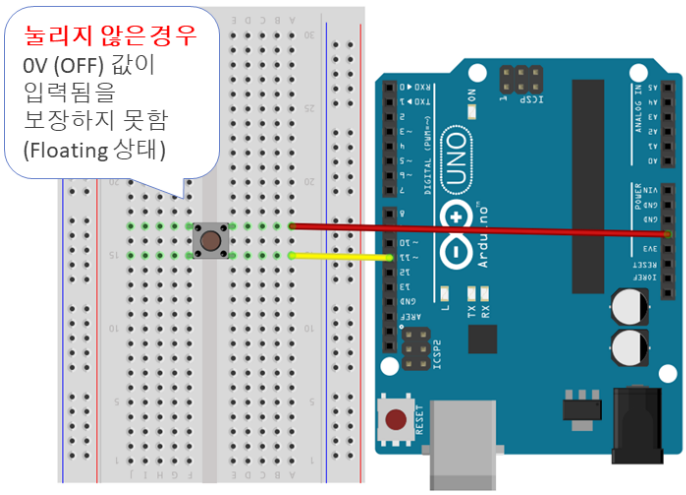


버튼을 누르면 점접 A,C가 B,D에 연결된다. 보통 접점 A,C중 한 곳과 B,D중 한 곳을 연결하는 형태로 사용한다.

나. Active HIGH 버튼 설계

설계목표: 버튼이 눌린 경우 HIGH 상태가 되고, 버튼이 떨어진 경우 LOW 상태가 되는 버튼 회로를 만들자.

1) 첫 번째 시도 - 플로팅

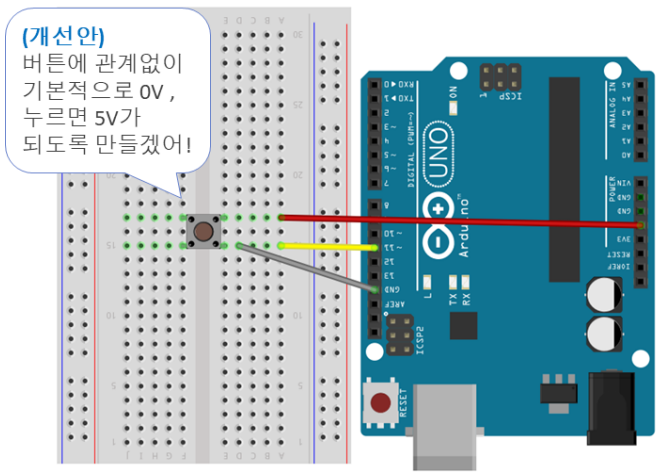


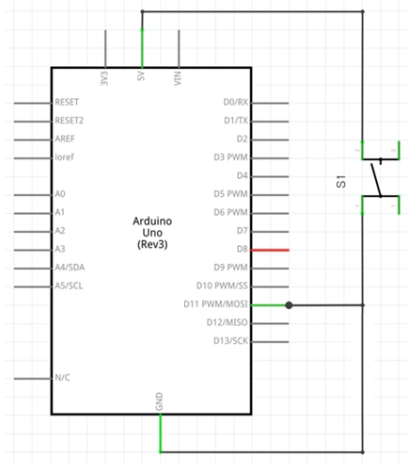

```
#define BTN 11

void setup() {
    pinMode(BTN, INPUT);
    Serial.begin(9600);
}

void loop() {
    bool r = digitalRead(BTN);
    Serial.println(r);
    delay(300);
}
```

버튼의 상태(눌렸는지 떼어졌는지)를 읽기 위한 위와 같은 시도를 고찰해 보자. 버튼이 눌리면 11번 핀은 5V 단자와 직접 연결되므로 분명 HIGH가 된다. 하지만 문제는 버튼이 눌리지 않았을 때이다. 버튼이 눌리지 않으면 digitalRead()로 읽은 값이 0 또는 1로 고정되지 않고 마구 흔들려서(floating 상태) 버튼의 상태를 정확히 알아낼 수 없다.

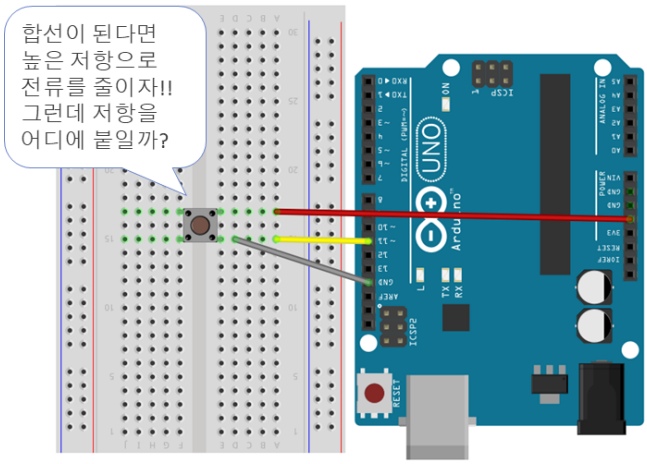
플로팅 상태에서는 정전기 등의 노이즈가 입력 값을 교란하기 때문이다. 이를 해결하려면 노이즈가 들어올 수 없도록 닫힌 회로를 구성해야 한다.

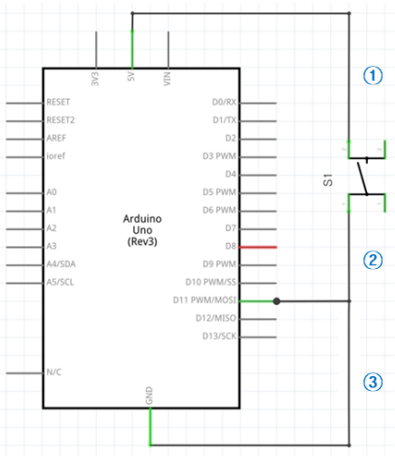
2) 두 번째 시도 – 합선





두 번째 시도는 노이즈가 영향을 줄 수 없게 완전히 닫힌 회로를 구성한 것은 평가할 만하지만 이렇게 연결하면 합선 상태(과도한 전류가 흐르게 됨)가 되어 아두이노가 다운된다. 이 상태를 벗어나려면 높은 저항을 사용하여 전류량을 확 줄여야 한다.





그래서 대략 높은 저항(10㏀)을 붙이려고 저항을 집어 들었는데 고민이 생겼다. 도대체 저항을 어디에 붙어야 하는 걸까?

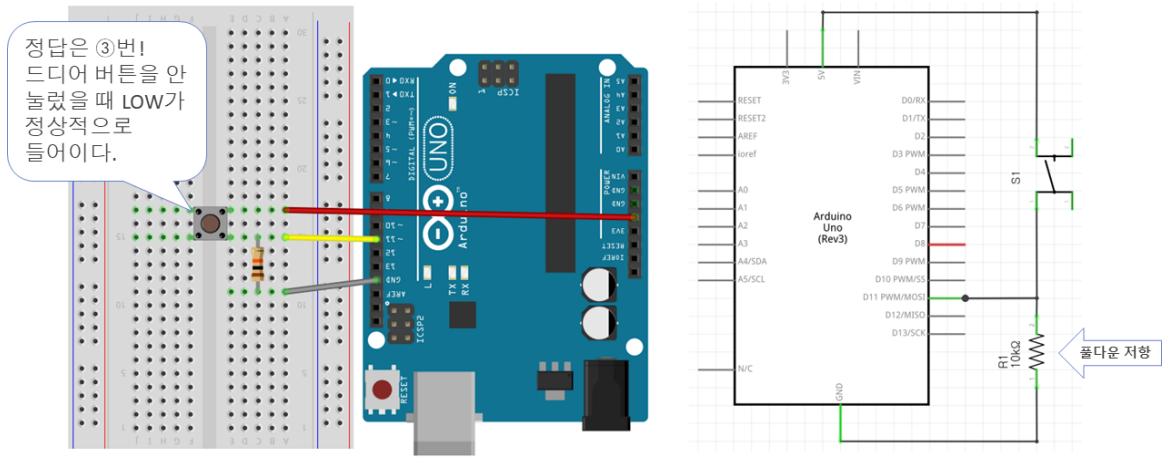
①번: 버튼 위

②번: 버튼 아래이면서 입력 단자인 11번 핀 위

③번: 입력 단자 11번 아래

왜 이런 고민을 하지? 전류량을 줄이는 것이 목적이라면 아무데나 붙이면 되는 거 아닐까? 정말 어느 위치든 상관없는 것일까? 잠시 고민해 보자.

3) 세 번째 시도 - 풀 다운 저항



※풀-다운 저항 : 스위치가 열려 있을 때, 플로팅 상태를 방지하고 LOW 값을 유지하기 위해 GND쪽에 첨부된 저항

결론부터 말하자면 정답은 ③번이다. ③번 위치에 저항을 붙여서 실험해 보면 합선 문제도 해결되었고, 버튼이 떨어져 있을 때도 LOW 값이 안정적으로 유지된다. 그런데 ① 또는 ②번 위치에 저항을 붙이는 건 왜 안되는 것일까? 합선을 막기 위한 것이라면 ①, ②, ③번 위치 어디는 상관없는 것이 아닌가?

그렇다. 단지 전류량을 줄여서 합선을 예방하기 위한 것이라면 어느 위치든 상관없다. 하지만 ① 또는 ②번 위치에 저항을 붙이게 되면 버튼 눌러도 HIGH가 아니라 LOW가 입력될 것이다. 왜냐하면 저항을 거치면서 5V의 전압강하(저항이 1개이므로 여기서 모든 전압강하가 일어남)가 일어나고 전압이 떨어진 상태에서 11번 핀이 값을 읽기 때문에 버튼을 눌렀음에도 불구하고 LOW 값이 들어오게 되는 것이다. 그러므로 위 회로에서 저항은 ③번 위치 이외에는 선택의 여지가 없다.

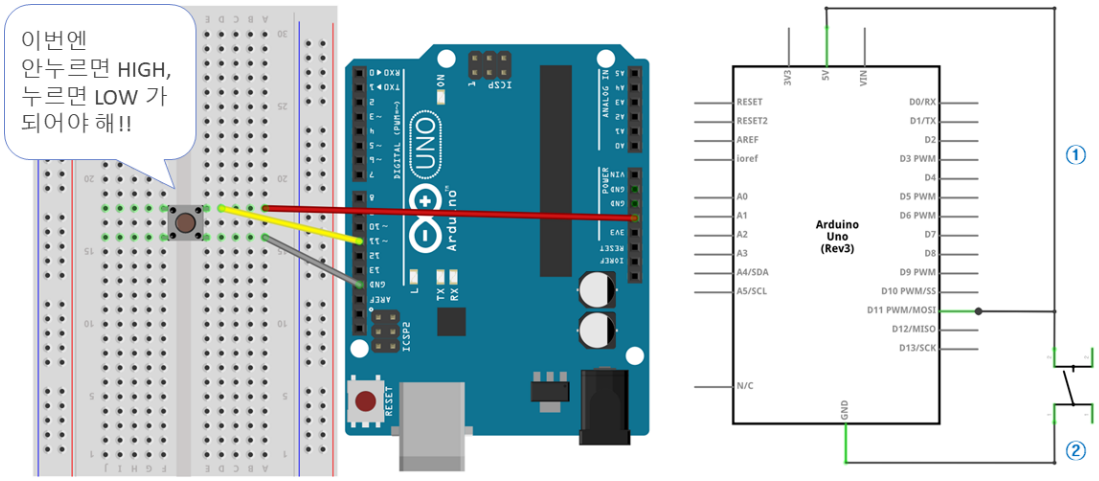
이와 같이 스위치가 열려 있을 때 전압을 결정하지 못하고 플로팅(둥둥 떠 있는)되어 있는 입력 단자를 0V로 끌어 내려주는 역할을 하는 저항이기에 '풀-다운 저항'이라는 이름이 붙게 되었다.

[깜짝 퀴즈]
아두이노의 입력 단자(GPIO 핀을 입력 모드로 전환했을 때)는 전압을 측정하는가? 아니면 전류를 측정하는가?

다. Active LOW 버튼 설계

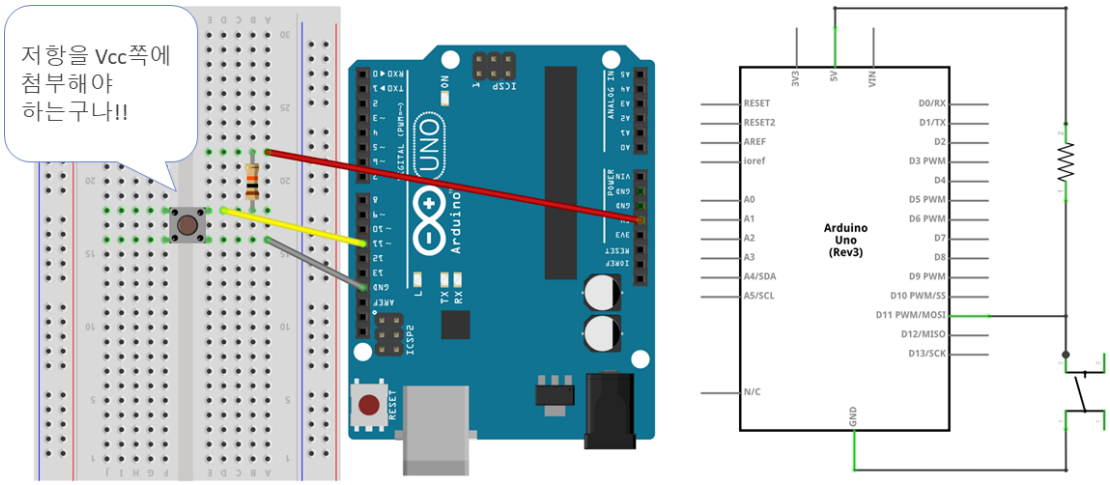
설계목표: 버튼이 눌린 경우 LOW 상태가 되고, 버튼이 떨어진 경우 HIGH 상태가 되는 버튼 회로를 만들자.

1) 첫 번째 시도



위와 같은 설계에서는 버튼을 누르게 되면 합선으로 과전류를 유발하며 아두이노는 즉시 다운된다. 이번에도 역시 저항이 첨부되어야 한다. 그런데 어느 위치에 끼워 넣어야 할까?

2) 두 번째 시도 - 풀업저항



※풀-업 저항 : 스위치가 열려있을 때, 플로팅 상태를 예방하고 HIGH 값을 유지하기 위해 Vcc쪽에 첨부된 저항

이번에는 저항을 Vcc 쪽에 붙여야 설계 목표를 달성할 수 있다. 먼저 버튼이 눌린 경우 11번 핀은 GND 단자와 직접적으로 연결되어 LOW가 인식된다. 그런데 버튼이 눌리지 않은 경우라면 어떻게 될까? 그렇게 되면 버튼이 연결된 회로는 오픈 상태가 된

다. 왜냐하면 5V 단자에서 출발하여 10㏀ 저항을 거쳐 GND로 이어지는 구간의 연결이 (버튼이 안눌려서) 끊어졌기 때문이다. 개방된 회로에서는 전압강하가 일어나지 않을 것이고, 그 결과 5V 전압원은 10㏀ 저항을 지나왔지만 11번 핀에서는 여전히 5V가 측정되는 것이다. 이로써 설계 목표는 달성되었다.

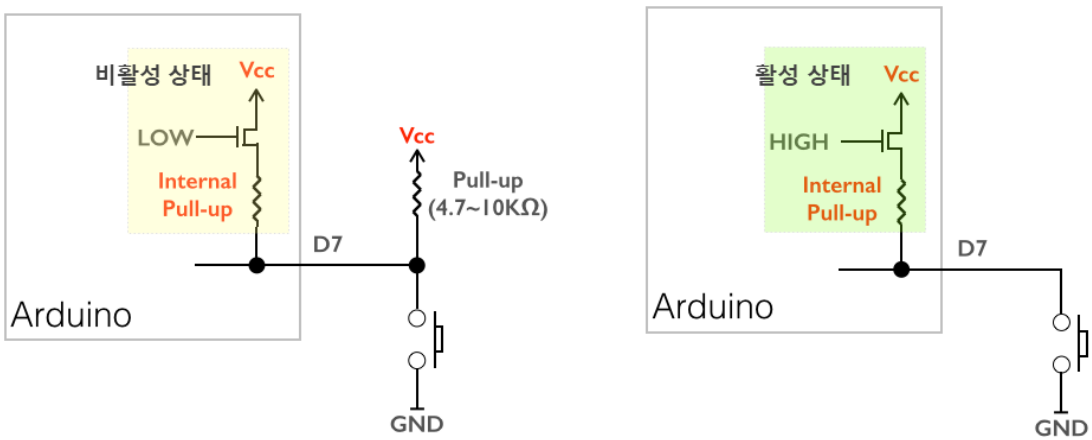
추가로 이번 설계에서 저항이 Vcc 쪽에 붙어야만 하는지도 검토해 보자. 만약 10㏀ 저항이 11번 핀 아래쪽에 첨부된 경우를 생각해 보자. 버튼이 안 눌린 상태에서는 11번 핀이 5V 전압원에 직접 연결된 상태로 HIGH가 인식된다(개방된 회로로 전압강하 요인 없음). 하지만 버튼이 눌리게 되면 어떻게 될까? 폐회로가 되면서 전압강하가 일어날 것이다. 그런데 위치가 오묘하다. 11번 핀 지점을 지나 저항이 위치하므로 11번 핀 지점은 여전히 5V이고 10㏀ 저항까지 지나고 나서야 0V로 떨어진다. 전압강하 현상은 일어났지만 전압강하 되기 이전 지점에서 측정했으므로 기대했던 LOW가 아닌 HIGH가 읽히는 황당한 일이 발생하는 것이다. 따라서 10㏀ 저항이 입력 단자 아래로 내려오면 버튼과 상관없이 항상 HIGH가 되므로 저항은 ①번 위치 이외에는 선택의 여지가 없다.

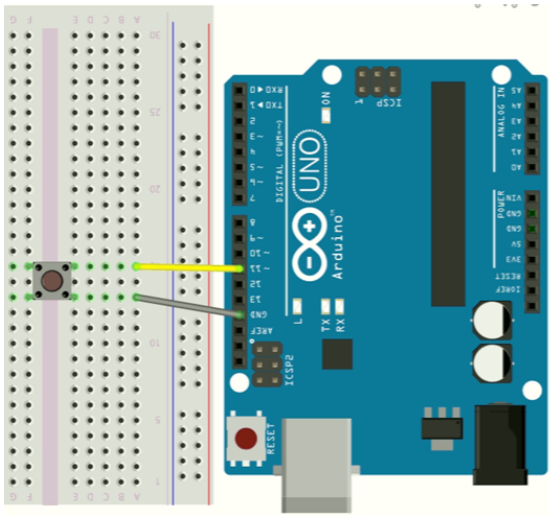
자, 이제 완성된 회로에서 잠시만 저항을 지워보자. 버튼을 누를 때는 0V가 인식되고 아무런 문제가 없다. 하지만 버튼을 열면 11번 입력 단자는 플로팅 상태가 된다(연결된 곳이 없으므로 측정값이 흔들린다). 이와 같이 스위치가 열려 있을 때 전압을 결정하지 못하고 풀로팅 되어 있는 입력 단자를 5V로 끌어 올리기 위해 사용하는 저항이기에 '풀-업 저항'이라는 이름이 붙게 되었다.

3) 세 번째 시도 - 내부 풀업(internal pull-up) 저항

버튼 상태 읽기에서 살펴본 여러 가지 상황을 고려해 보면 버튼을 정상적으로 사용하기 위해서 저항의 사용은 피할 수 없는 것으로 보인다. (풀-업 또는 풀-다운 저항 처리가 필수) 회로에 버튼을 추가할 때마다 저항을 끼워 넣으면 비용이 올라가고 회로가 복잡해지는데 저항 없이 버튼을 작동시킬 방법이 정말 없는 것일까?

다행스럽게도(?) 아두이노의 모든 GPIO핀은 내부에 풀업 저항을 내장하고 있으며 프로그램 명령을 통해 이 저항의 연결을 활성화하는 것이 가능하다.





```
#define BTN 11

void setup() {
    pinMode(BTN, INPUT_PULLUP);
    Serial.begin(9600);
}

void loop() {
    bool r = digitalRead(BTN);
    Serial.println(r);
    delay(300);
}
```

INPUT_PULLUP 키워드를 사용함으로써 아두이노 내부의 풀업 저항을 활성화 시킬 수 있고 그렇게 되면 사용자가 저항을 붙일 필요가 없어진다. 다만 내부 저항은 풀업 저항이기 때문에 ACTIVE LOW로만 작동하는 회로를 만들 때만 이용 가능하다.

라. 버튼 회로 설계 정리

구분	풀다운 저항	풀업 저항	내부 풀업 저항
설계목표	ACTIVE HIGH (누르면 HIGH, 떼면 LOW)	ACTIVE LOW (누르면 LOW, 떼면 HIGH)	ACTIVE LOW (누르면 LOW, 떼면 HIGH) 저항 없이 구현
저항위치	10㏀ GND 쪽에 부착	10㏀ Vcc 쪽에 부착	저항 불필요
pinMode	INPUT	INPUT	INPUT_PULLUP

3. 버튼 구현 실습

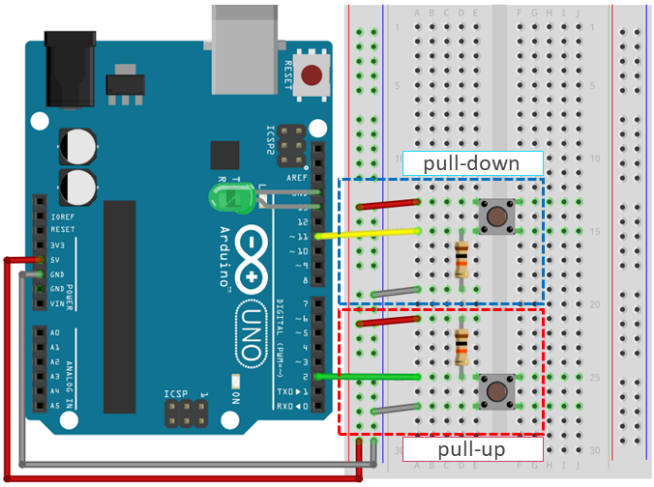
가. 버튼 입력에 LED 반응하기

1) 문제

버튼 입력상태에 따라 LED가 반응하도록 만드시오.
- 버튼이 눌리면 LED가 켜짐
- 버튼이 떨어지면 LED가 꺼짐
- 버튼 입력에 즉시 LED가 반응해야 함. 즉, 딜레이가 없어야 함.

2) 풀이

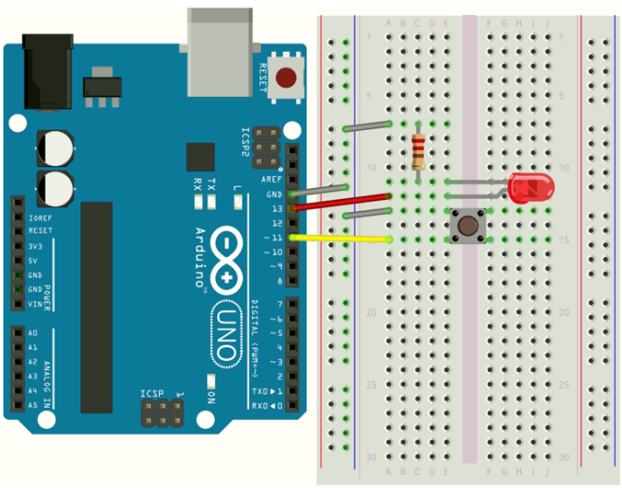
① 풀다운 또는 풀업 저항 사용 시



풀다운 저항 버튼일 때	풀업 저항 버튼일 때

#define BTN 11 // pull-down
#define LED 13
 	
void setup() {
  pinMode(BTN, INPUT);
  pinMode(LED, OUTPUT);
}
 	
void loop() {
  bool btn = digitalRead(BTN);
  digitalWrite(LED, btn);
}
	#define BTN 2 // pull-up
	#define LED 13
	
	void setup() {
	  pinMode(BTN, INPUT);
	  pinMode(LED, OUTPUT);
	}
	
	void loop() {
	  bool btn = digitalRead(BTN);
	  digitalWrite(LED, !btn);
	}

② 내부 풀업 저항 사용 시



```
#define BTN 11
#define LED 13

void setup() {
  pinMode(BTN, INPUT_PULLUP);
  pinMode(LED, OUTPUT);
}

void loop() {
  bool btn = digitalRead(BTN);
  digitalWrite(LED, !btn);
}
```

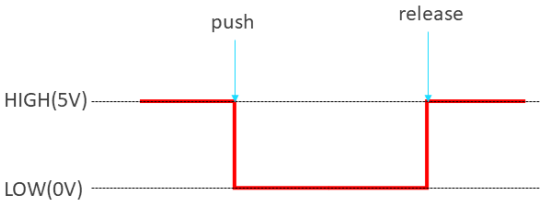
나. 토글 버튼 구현하기

1) 문제

토글 버튼을 구현하시오.
- 버튼을 한 번씩 누를 때 마다 LED의 켜짐과 꺼짐이 반전되어야 한다.
- 이전 문제(버튼 입력에 LED 반응하기) 회로를 그대로 사용할 것.
- 프로그램만 만드시오.

2) 풀이

▪ INPUT_PULLUP 버튼의 상태도



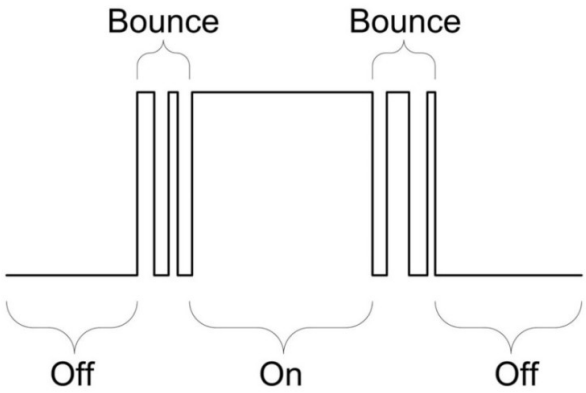

```
#define BTN 11
#define LED 13

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BTN, INPUT_PULLUP);
}

void loop() {
  static bool btn_pre = true;
  bool btn_now = digitalRead(BTN);

  if(btn_pre == LOW && btn_now == HIGH) {
    digitalWrite(LED, !digitalRead(LED));
  }
  btn_pre = btn_now;
}
```

위 소스코드는 이론적으로는 완벽하지만 실제 작동은 완벽하지 않다. 그 이유를 확인해 보기 위해 아래와 같이 소스코드를 수정해 본다.

```
#define BTN 11
#define LED 13

void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
  pinMode(BTN, INPUT_PULLUP);
}

void loop() {
  static int n=0;
  static bool btn_pre = true;
  bool btn_now = digitalRead(BTN);

  if(btn_pre == LOW && btn_now == HIGH) {
    Serial.println(n++);
    digitalWrite(LED, !digitalRead(LED));
  }
  btn_pre = btn_now;
}
```

다. 채터링 보정

1) 채터링(chattering) 현상이란?



전자회로 내의 스위치 접점이 닫히거나 열리는 순간에 기계적인 진동에 희해 매우 짧은 시간 안에 스위치가 부었다가 떨어지는 것을 반복하는 현상을 말한다.

2) 채터링 보정방법

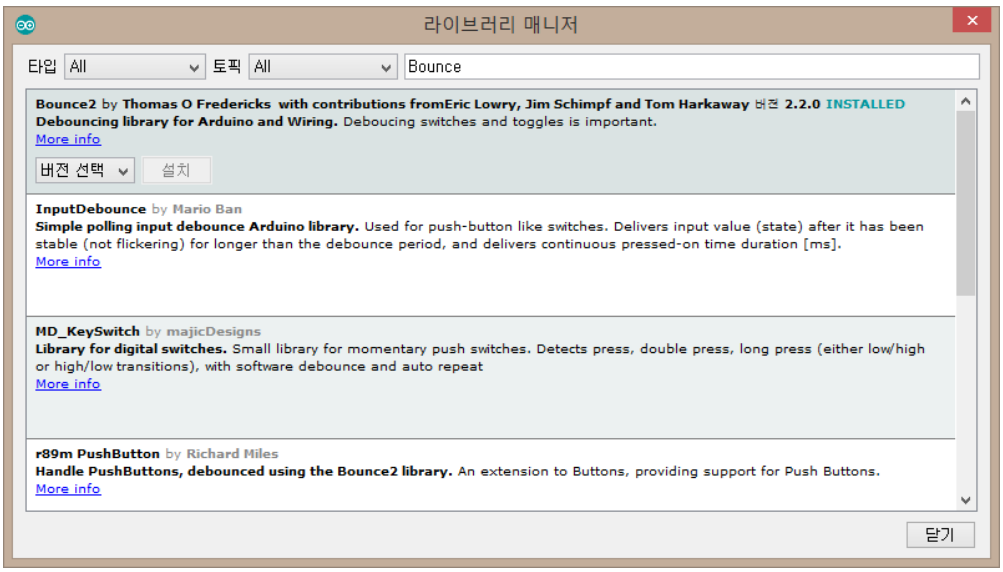
가) 하드웨어 적인 방법

버튼(스위치)에 콘덴서를 추가하여 채터링을 없앨 수 있지만 버튼의 상태 변경이 인식되는 속도가 느려지는 부작용이 발생한다.

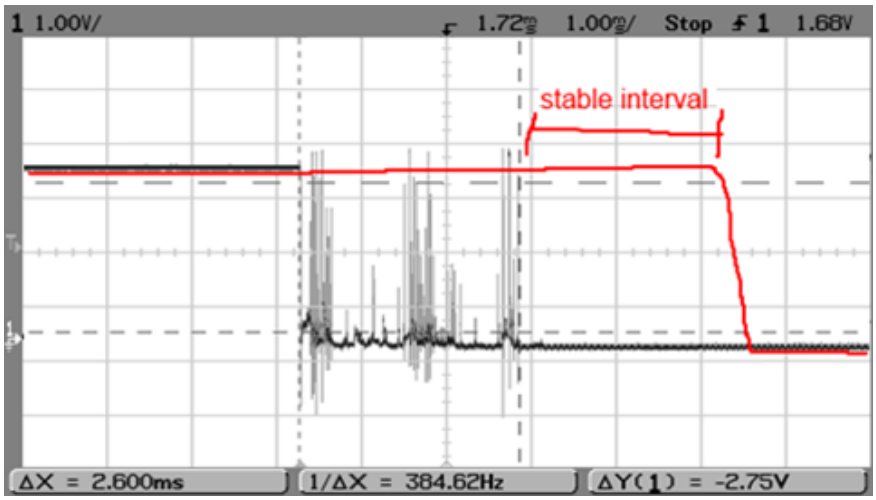
나) 소프트웨어적인 방법

버튼의 상태를 인식하는 프로그래밍에 변화를 주는 것이다. 가장 간단한 방법은 채터링이 발생하는 시간(약 20ms)을 delay 함수로 덮어 버리는 것이다. 하지만 100% 신뢰할 만한 방법이 아니기에 아두이노의 Bounce2라이브러를 활용하는 방법을 추천한다.

3) Bounce2 라이브러리 설치



4) Bounce2 라이브러리 작동 방식



11번 핀에 INPUT_PULLUP 상태로 연결된 버튼의 동작은 위 그림과 같다. 버튼을 누르지 않았을 때 HIGH를 유지하다가 버튼을 누르는 순간 LOW로 떨어지는데 그림과 같이 한 번에 떨어지지 못하고 채터링 현상이 발생하면서 전압이 위아래로 여러 차례 흔들린다. 이렇게 불안정한 상태에서는 HIGH인지 LOW인지를 판별하지 아니하고 신호가 일정 시간 동안 지속되면 그 때의 상태를 읽어내는 것이다.

5) Bounce2 메소드 알아보기

가) void interval(unsigned long interval)

변화로 인식할 신호의 지속 시간을 밀리초 단위로 지정한다.

나) void attach(int pin)

채터링 신호를 제거하고 값을 읽어낼 핀 번호를 지정한다. pinMode를 먼저 설정한 뒤 attach() 함수를 호출해야 함에 유의한다.

다) bool update()

Bounce는 인터럽트를 사용하지 않으므로 값을 읽기 전에는 update()를 호출해야만 한다. loop() 함수 안에 포함 시켜 가능하면 자주 이 함수를 호출하도록 하자 (loop 함수 내에서 한 번만 호출하면 됨). update() 메서드는 핀의 상태가 변화하면 1을 반환하고 그렇지 않으면 0을 반환한다.

라) bool read()

핀 상태의 변화를 읽어낸다.

마) bool fell()

핀의 상태가 high에서 low로 변화하였으면 true를 반환한다.

바) bool rose()

핀의 상태가 low에서 high로 변화하였으면 true를 반환한다.

```
#include <Bounce2.h>

#define BTN 11
#define LED 13

// Instantiate a Bounce object :
Bounce debouncer = Bounce();

void setup() {
  // Setup the button with an internal pull-up :
  pinMode(BTN, INPUT_PULLUP);

  // After setting up the button, setup the Bounce instance :
  debouncer.attach(BTN);
  debouncer.interval(50);

  // Setup the LED :
  pinMode(LED,OUTPUT);
}

void loop() {
   // Update the Bounce instance :
   debouncer.update();

   // Call code if Bounce fell
   //      (transition from HIGH to LOW) :
   if ( debouncer.fell() ) {
     // Toggle LED state :
     digitalWrite(LED, !digitalRead(LED));
   }
}
```

6) Bounce2를 이용한 버튼 눌린 횟수 카운팅

```
#include <Bounce2.h>

#define BTN 11
#define LED 13

// Instantiate a Bounce object :
Bounce debouncer = Bounce();

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

  // Setup the button with an internal pull-up :
  pinMode(BTN, INPUT_PULLUP);

  // After setting up the button, setup the Bounce instance :
  debouncer.attach(BTN);
```

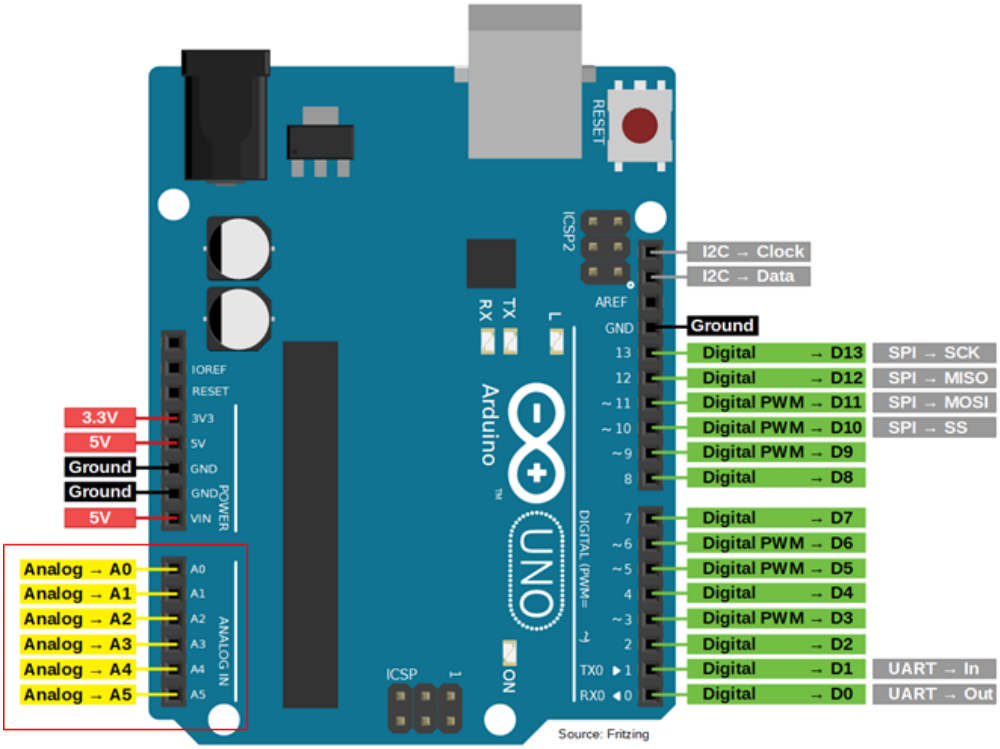
```
  debouncer.interval(50);

  // Setup the LED :
  pinMode(LED,OUTPUT);
}

void loop() {
  static int n=0;
  // Update the Bounce instance :
  debouncer.update();

  // Call code if Bounce fell
  //      (transition from HIGH to LOW) :
  if ( debouncer.fell() ) {
    Serial.print(n++);
    digitalWrite(LED, !digitalRead(LED));
  }
}
```

위와 같이 프로그래밍함으로써 채터링에 의해 비정상적으로 카운트가 올라가는 것을 막을 수 있다.

제4장 아날로그 입력

1. 아날로그 입력의 이해

가. 아두이노에서 ANALOG IN 핀 사용



analogRead() 함수는 아두이노의 아날로그 핀으로부터 값을 읽어옵니다. 아두이노 우노의 경우 6개(A0, A1, A2, A3, A4, A5)의 10bit ADC(Analog to Digital Converter)가 내장되어 있어 아날로그 핀으로 입력되는 0 ~ 5V의 신호를 2^{10}(1024) 즉 0 ~ 1023 범위의 정수 값으로 변환한다.

ADC는 아날로그 입력을 디지털 데이터로 변화하는데 일정 시간(Conversion Time)이 필요하며 아두이노에서는 아날로그 입력을 읽어오는데 약 100 마이크로초(0.0001초)의 시간이 소요된다.

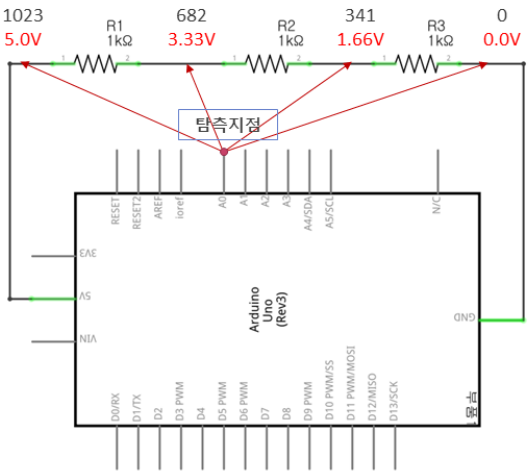
ADC는 ANALOG IN 핀 에서만 사용 가능하므로 아날로그 입력을 사용하려면 6개의 A0 ~ A5 핀을 사용해야 하며, 이 핀은 핀에서 입력된 전압의 크기를 읽어낸다. 아날로그 값을 읽을 때는 analogRead() 함수를 사용하며, 디지털 입출력 시 사전에 pinMode() 함수를 호출하는 것과는 달리 analogRead() 함수를 사용하기 전에는 pinMode(Ax, INPUT) 함수를 호출할 필요가 없다.

나. 아날로그 입력 값 취득 실험

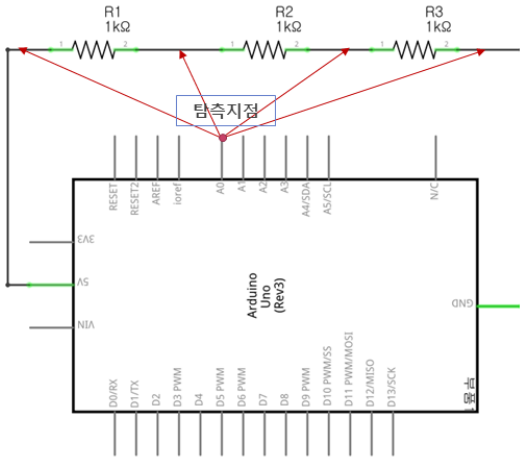
1) 실험 전 고찰

아래와 같이 회로를 구성하고 탐측 지점에서 analogRead() 함수로 취득한 값을 확인해 보자. 이론과 실험 결과가 일치하는가?

▪ 회로도

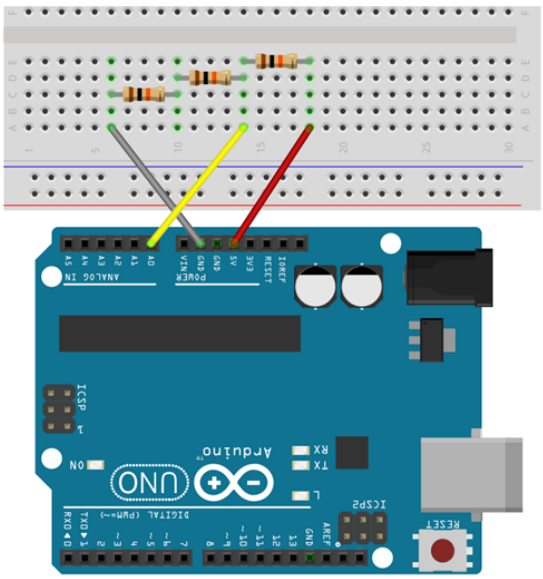


▪ 개방된 회로에서는?



2) 아두이노 전압계

아래와 같은 프로그램은 아두이노의 아날로그 포트를 전압계로 바꾸어준다. 탐측 지점을 옮겨 가면서 수치를 확인해 보자.



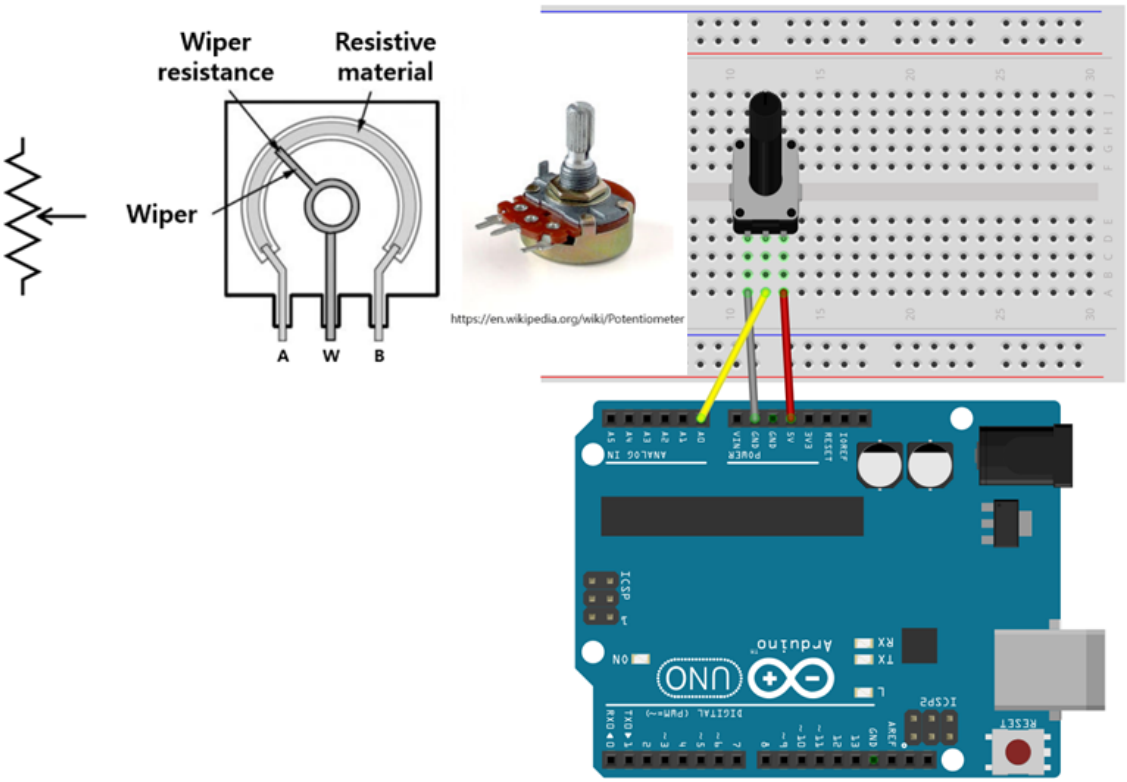
```
void setup() {
  Serial.begin(9600);
  Serial.println("a, V");
}

// A0에 연결된 선을 각 탐측지점에
// 꽂아 보면서 실험한다.
void loop() {
  int a = analogRead(A0);
  // 0~1023 범위의 값을 0~5범위로.
  double v = a/1023.0 * 5;

  Serial.print(a);
  Serial.print(", ");
  Serial.print(v);
  Serial.println("V");
  delay(200);
}
```

- 저항을 거치면서 전압강하가 일어나는 것을 확인했는가?
- 이론과 측정결과가 동일 한가?
- 개방된 회로에서는 어떠한 결과를 얻었는가?

3) 가변 저항에 연결

① 회로구성

아래와 같이 회로를 구성한 뒤 가변 저항의 레버를 돌려가면서 변화를 관찰한다.



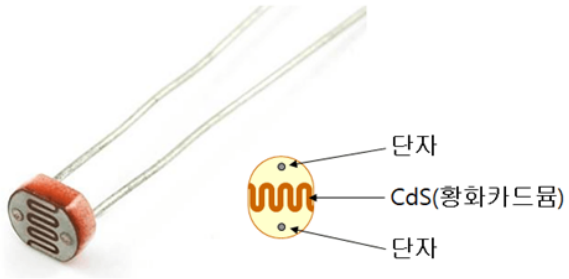
② 소스코드

```
void setup() {
  Serial.begin(9600);
  Serial.println("a, V");
}

// A0에 연결된 선을 각 탐측지점에
// 꽂아 보면서 실험한다.
void loop() {
  int a = analogRead(A0);
  // 0~1023 범위의 값을 0~5범위로.
  double v = a/1023.0 * 5;

  Serial.print(a);
  Serial.print(", ");
  Serial.print(v);
  Serial.println("V");
  delay(200);
}
```

2. 조도 센서의 사용

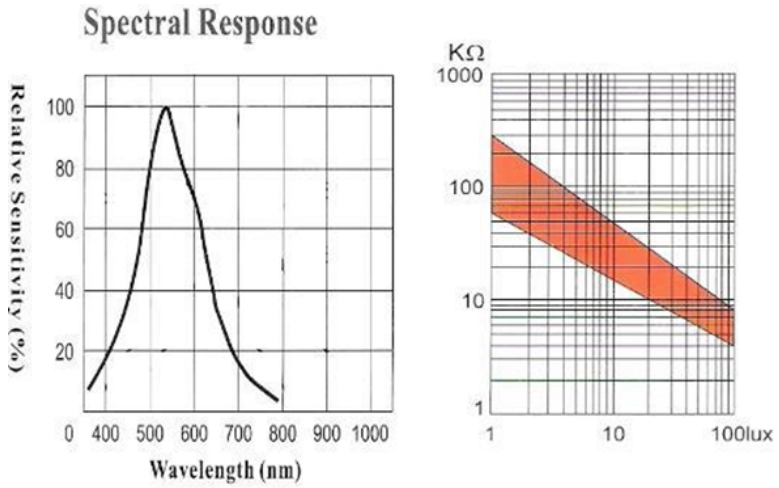
가. CdS 센서

1) 외형과 작동



Specifications

Model	Vmax (VDC)	Pmax (mW)	Ambient temp(℃)	Spectral peak (nm)	Light Resistance at 10Lux (KΩ)	Dark Resistance (MΩ)	Gamma value at 100-10Lux	Response Time (ms)	
								Rise Time	Decay time
GL5516	150	90	-30~+70	540	5-10	0.2	0.6	30	30



CDS는 광도전셀이라고도 하며 카드뮴과 유황이 화합하여 생긴 황화카드뮴 결정에 금속다리를 붙인 부품으로써 가시광선이 없는 어두운 곳에서는 절연체와 같이 전류가 흐르지 않다가 가시광선이 닿으면 도체와 같이 전류가 잘 흐르는 성질을 가지고 있다. 다시 말해서 어두운 곳에서는 높은 저항값을 갖고 있다가 빛이 밝아질수록 자체의 내부 저항값이 낮아진다.

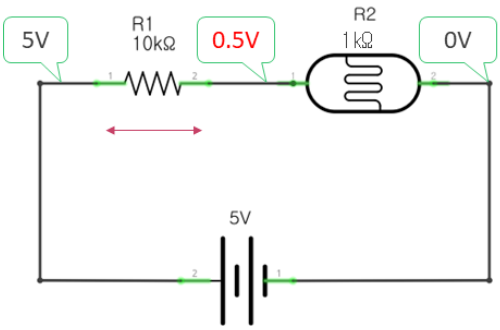
즉, CDS는 빛의 밝기에 따라 내부 저항값이 변화하여 연결된 회로를 동작시키는 일종의 가변저항기라 생각할 수 있다.

이 센서는 고감도, 소형, 저가격, 가시광선에 민감한 특징으로 자동점멸기, 노출계 등 용도가 넓다. 그러나, 응답속도가 0.1~10ms로 늦다는 것이 결점이다.

나. 조도 센서 연결 회로 설계

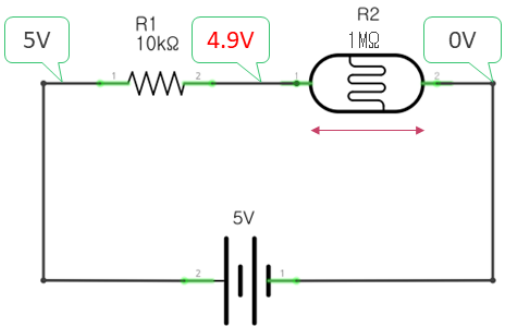
1) 저항 먼저 연결

▪ 밝은 곳



▪ 중간 지점의 전압이 낮게 나온다.

▪ 어두운 곳



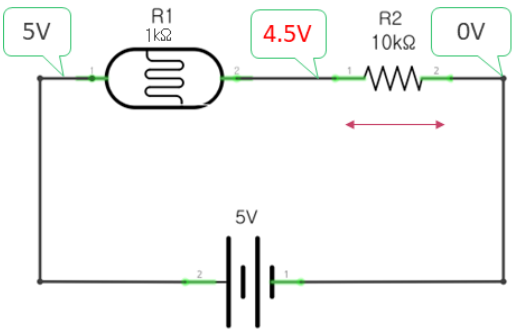
▪ 중간 지점의 전압이 높게 나온다.

저항과 조도 센서 사이에서 아두이노의 아날로그 핀으로 전압을 읽는다.

왼쪽 그림과 같이 저항을 먼저, 조도 센서를 나중에 연결한 경우를 생각해 보자. 밝은 곳에서는 조도 센서의 저항이 작아진다. 예를 들어 1㏀까지 낮아졌다고 하자. 그러면 R1의 저항이 10㏀이므로 대부분의 전압강하가 R1에서 발생한다. 따라서 중간지점에서 전압은 매우 낮은 상태가 된다. 반대로 어두운 곳에서는 조도 센서의 저항이 커진다. 예를 들어 1㏁까지 상승하면 R1은 저항값은 아주 미미하게 되고 R1에서의 전압강하량은 미미할 것이다. 즉 가운데 지점에서의 전압값은 매우 높게 측정될 것이다.

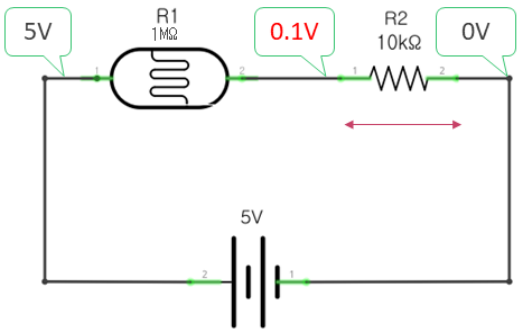
2) 센서 먼저 연결

▪ 밝은 곳



▪ 중간 지점의 전압이 높게 나온다.

▪ 어두운 곳

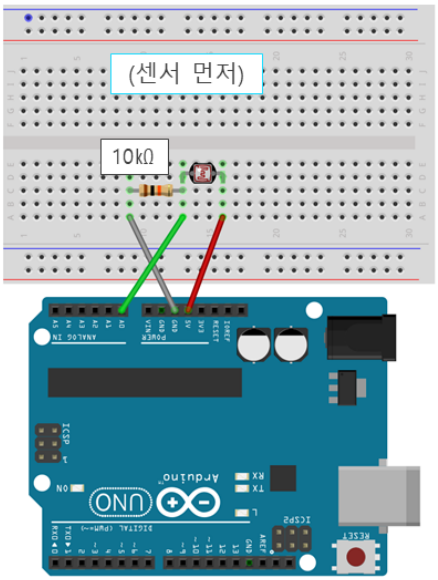


▪ 중간 지점의 전압이 낮게 나온다.

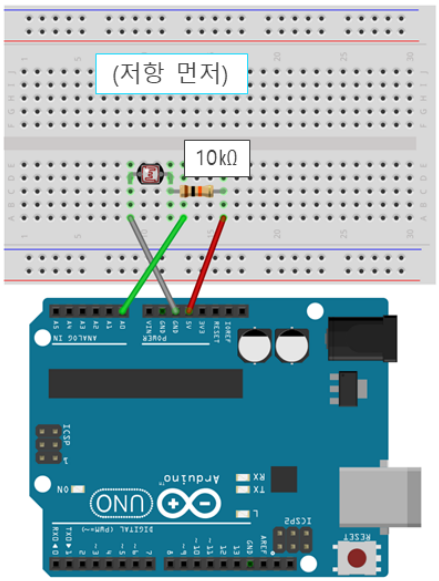
조도 센서보다 저항을 먼저 연결한 위 상황과 정확히 반대 상황이 펼쳐진다.

3) 회로 연결

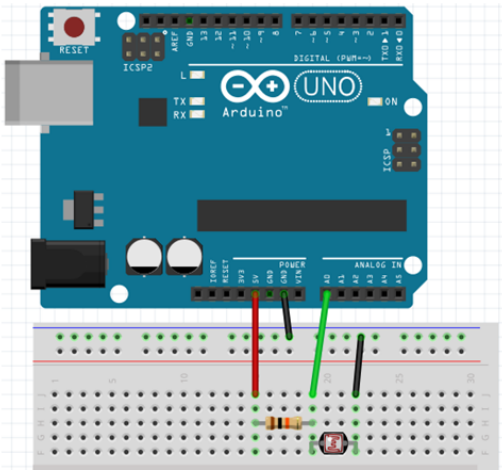
- 밝을 수록 큰 숫자 회로



- 어두울 수록 큰 숫자 회로



4) 실험



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

void loop() {
    int v = analogRead(A0);
    Serial.println(v);
    delay(1000);
}
```

위 회로는 저항을 먼저 연결한 회로인가? 조도 센서를 먼저 연결한 회로인가? 직접 실험을 통해 확인해 보자.

어두울 때 큰 숫자가 나오는가?

아니면, 밝을 때 큰 숫자가 나오는가?

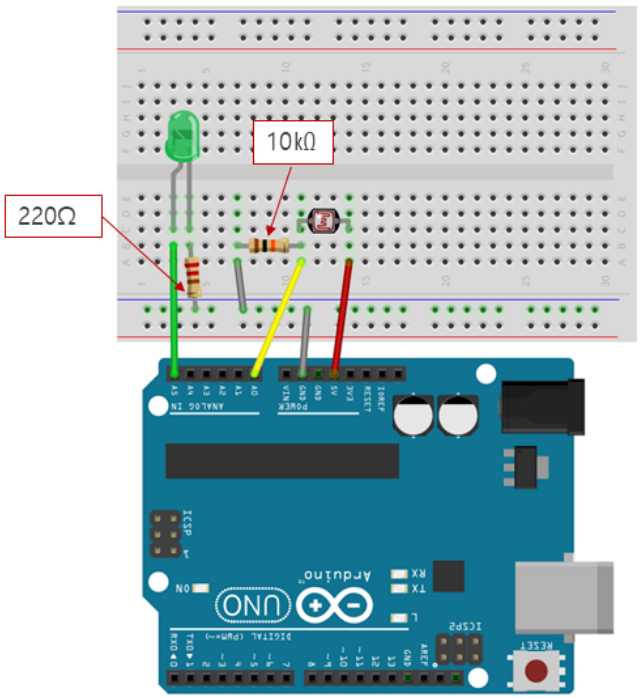
3. Analog Input 실습 과제

가. 주변 밝기에 반응하여 LED켜기

1) 문제

조도센서와 LED를 조합하여 어두워지면 저절로 켜지는 가로등을 구현하시오.
(가로등을 실제로 가져올 수 없으므로 LED를 가로등이라 가정한다)

2) 풀이




```
#define CDS A0
#define LED A5

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

void loop() {
   int v = analogRead(CDS);
   Serial.println(v);

   if(v < 550)
      digitalWrite(LED, HIGH);
   else
      digitalWrite(LED, LOW);
}
```

나. 주변 밝기에 반응하여 진짜 전등(220V) 켜기

1) 문제

조도센서를 이용하여 어두워지면 저절로 켜지는 전등을 구현하시오.
전등은 220V 전원으로 켜지므로 220V 전원을 제어해야 함.

2) 풀이

TTL신호로 220V를 제어해야 하므로 릴레이를 사용해야 한다. 릴레이 사용법은 후반부에서 다루기로 하자.

※ 220V 전원은 감전 시 매우 위험하므로 주의해야 함에 유의.

제5장 아날로그 출력

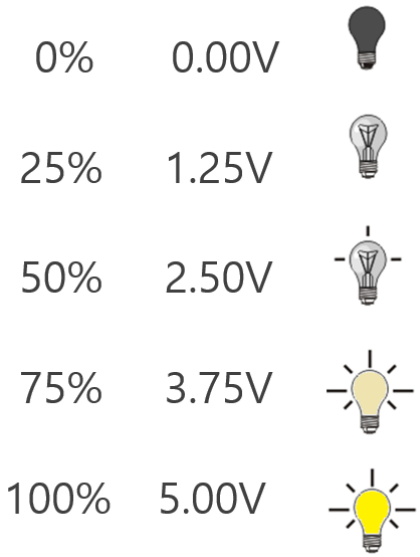
1. 아날로그 출력의 개념

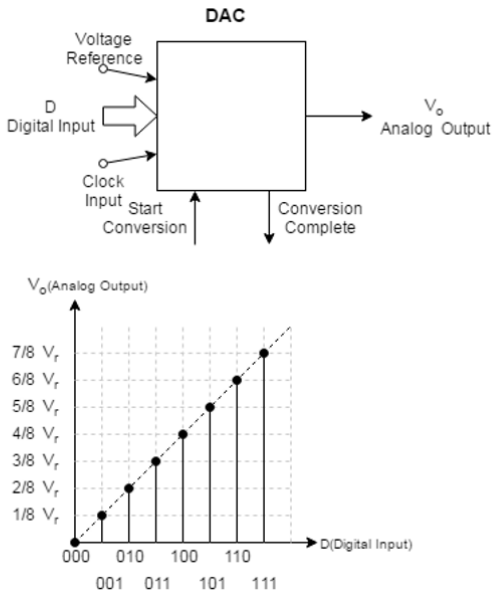
가. 출력 조절

1) 디지털 출력의 한계

아두이노는 8bit RISC 기반 마이크로컨트롤러로 기본적으로 모든 출력이 디지털로 나오게 된다. 디지털이기에 출력은 단 두 가지 즉, 0V 또는 5V의 전압 출력만 존재하며 중간 단계의 출력은 내보낼 수 없다.

2) 출력의 단계별 조절



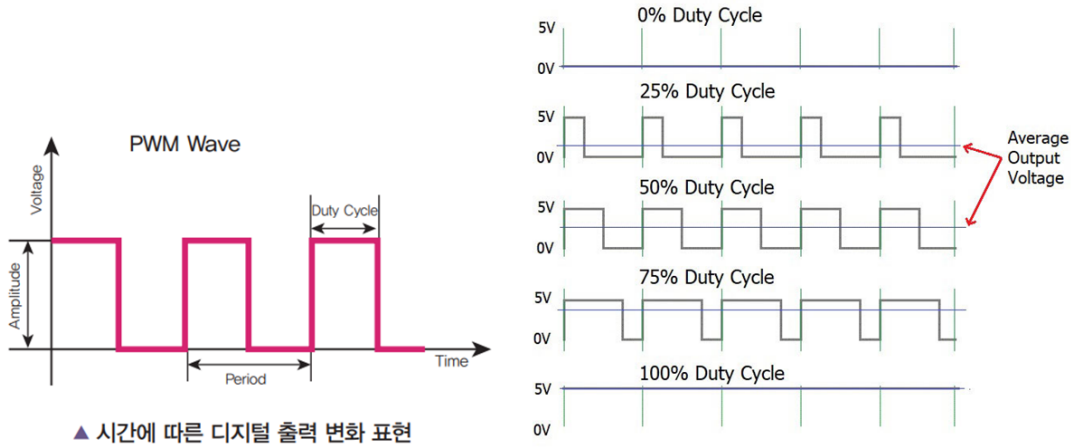


만약 위 그림과 같이 전압을 단계별로 출력해야 한다면 어떻게 해야 할까? 가능하기는 한 것일까? 모두의 예상과는 달리 아두이노에서는 위와 같은 중간 단계의 출력을 내보내는 것이 불가능하며 오직 디지털 출력만 가능하다. 아두이노에는 중간 단계의 전압을 출력할 수 있는 DAC(Digital to Analog Convert)가 내장되어 있지 않기 때문이다. (반면, ESP32 칩은 3개의 핀에서 DAC 출력을 지원한다)

그래서 DAC 기능이 없는 아두이노와 같은 MCU들은 PWM(Pulse Width Modulation)이라는 방식으로 중간 단계의 출력을 흉내 내는 방법을 사용한다. 따라서 엄밀하게 따지면 아두이노의 아날로그 출력은 진정한 아날로그 출력이 아니며 디지털 출력으로 아날로그 출력을 에뮬레이션 하는 것이다.

나. PWM

1) PWM(Pulse Code Modulation) 원리



▲ 시간에 따른 디지털 출력 변화 표현

그럼 PWM의 원리를 간략히 살펴보도록 하자. 우선 디지털 신호는 단 2가지 상태만 가질 수 있음을 명심해야 한다. 그러면 어떻게 해서 두 가지 값을 가지고 다양한 값들을 표현할 수 있을까? 이것은 일종의 눈속임이다.

예를 들어 모터에 5V의 전압을 10초 동안 걸었을 때 모터의 총 회전수가 100회라고 해보자(1초 동안 10회씩 가속된다 가정). 그렇다면 5V의 전압을 5초 동안 걸었다면 모터의 총 회전수는 50회가 될 것이다. 그러면 1초 동안 걸었다면? 모터의 회전수는 10회가 될 것이다.

이번에는 1초간 5V 전압을 걸고 1초 동안 0V, 다시 1초간 5V의 전압을 반복적으로 10초간 주었다고 하자. 결과적으로 5V의 전압을 5초간 준 셈이므로 이론상 모터의 회전수는 총 50회가 될 것이다.

그러면 이번에는 5V의 전압을 주는 시간과 0v의 전압을 주는 시간을 0.01초씩으로 하여 10초간 번갈아 주어보면 어떻게 될까? 결과적으로 앞서 예처럼 5V의 전압이 토탈 5초간 걸린 것과 동일하며 0V와 5V의 중간 값인 2.5V의 전압이 10초간 모터에 걸린 것과 같이 동작하게 된다. (모터의 회전 속도가 반으로 줄어듦)

이렇게 5V의 전압을 거는 시간과 0V의 전압을 주는 시간의 비율이 1:1면 평균 2.5V의 전압을 주는 셈이 되고 1:3이면 평균 1.25V의 전압을 주는 것처럼 눈속임을 할 수 있게 된다. PWM은 이와 같이 출력 대 비출력의 비율을 조절하여 다양한 전압을 주는 것처럼 눈속임하는 기법이다.

2) PWM 활용분야

조명의 밝기를 조절하거나 모터의 출력을 제어하고, 서보모터의 회전각을 제하는데 사용한다. 압전효과를 이용하여 부저에서 구형파를 생성함으로써 소리를 만들어 내는 것도 가능하다.

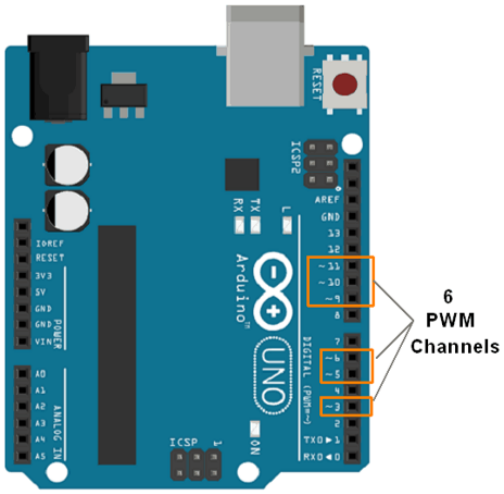
3) 아두이노의 PWM

- 아두이노 UNO의 PWM

핀번호	PWM주파수	주기
3,9,10,11	490Hz	0.00204초
5, 6	980Hz	0.00102초

- 아두이노 MEGA의 PWM

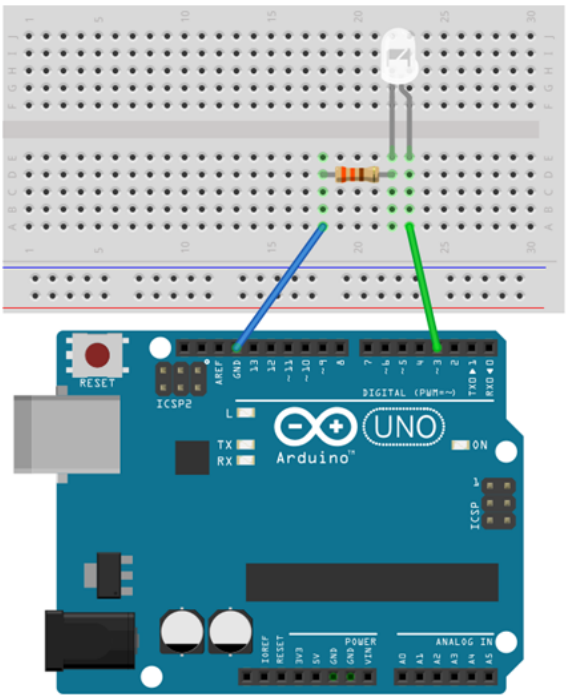
핀번호	PWM주파수	주기
2-13, 44-46	490Hz	0.00204초
4, 13	980Hz	0.00102초



아두이노 UNO에서는 디지털 단자 0~13번 핀 가운데 (~) 표시가 되어 있는 6개의 포트에서만 하드웨어 PWM출력이 가능하며 주기는 490Hz 또는 980Hz로 고정되어 있다. 명령어로 analogWrite()를 사용하며 PWM 출력 해상도는 8bit이기 때문에 출력값은 0부터 255사이의 값을 지정해야 한다.

2. 아날로그 출력 실습

가. Led fade in / fade out

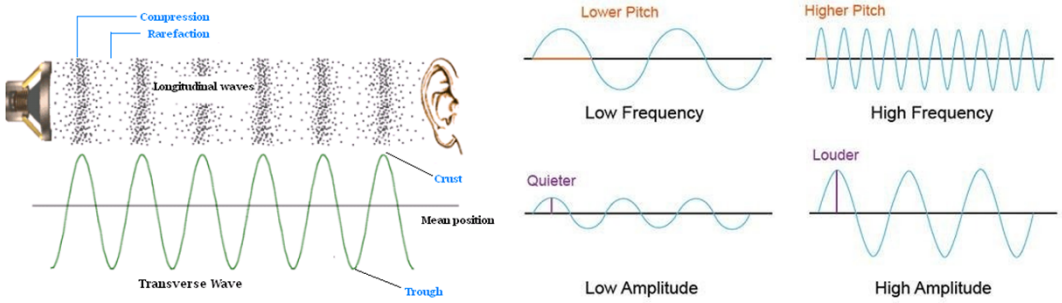


```
#define LED 3

void setup() {
      pinMode(LED, OUTPUT);
}

void loop() {
      int i;
      // fade in
      for(i=0; i<256; i+=4) {
            analogWrite(LED, b);
            delay(30);
      }
      //fade out
      for(i=255; i>=0; i-=4) {
            analogWrite(LED, b);
            delay(30);
      }
}
```

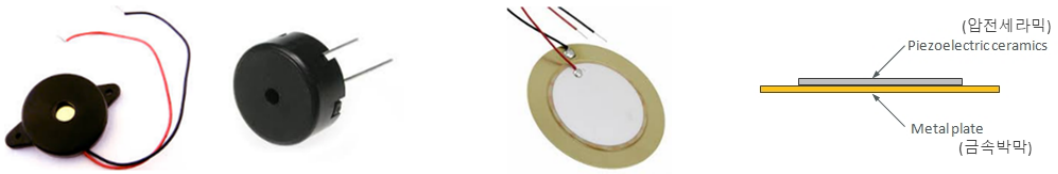
위 그림과 같이 간단하게 구현해 볼 수 있다. 30ms 단위로 4씩 증가/감소해 가면서 analogWrite() 함수를 호출하고 있다.

나. 음과 주파수

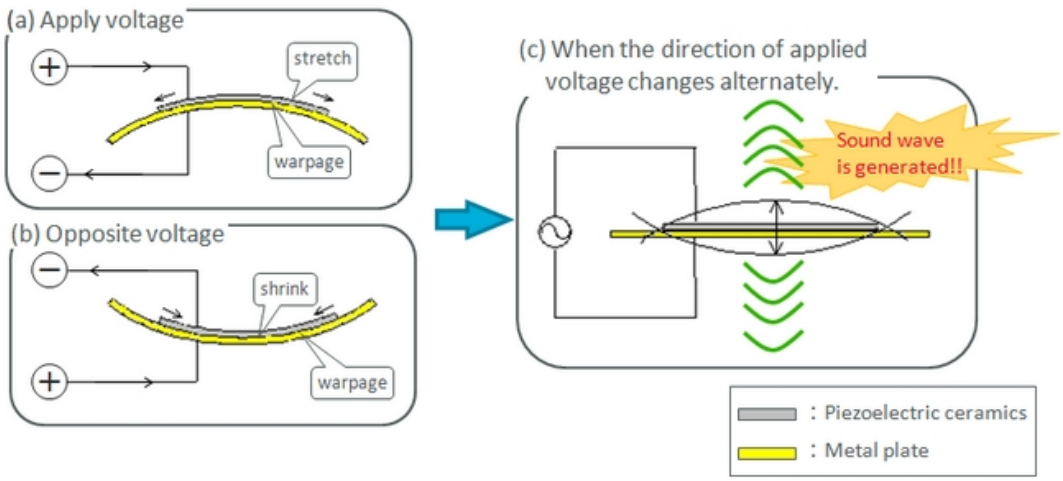
1) 음파의 특징



2) 압전부저



3) 작동원리



부저가 만들어내는 파형은 사인파가 아닌 구형파(square wave)이다. 디지털 출력이기 때문에 진폭을 변화시킬 수 없으며(소리의 크기는 바꿀 수 없다는 뜻) 단지 주파수(음조)만 변화시킬 수 있다.

4) Piezo buzzer 종류

- 액티브 부저: 전원이 인가되면 내부 발진기가 만들어 내는 톤을 발생시킨다.
- 패시브 부저: 내부 발진기(오실레이터)가 없기 때문에 직접 소리를 만들어야 한다.

다. 소리 만들기

1) tone() 함수

```
tone(int pin, uint value)

tone(int pin, uint value, ulong duration)
```

- pin: tone을 발생시킬 핀
- frequency: tone의 주파수 (Hz 단위) - unsigned int
- duration (옵션) : tone 의 지속 시간 (밀리초 단위) - unsigned long

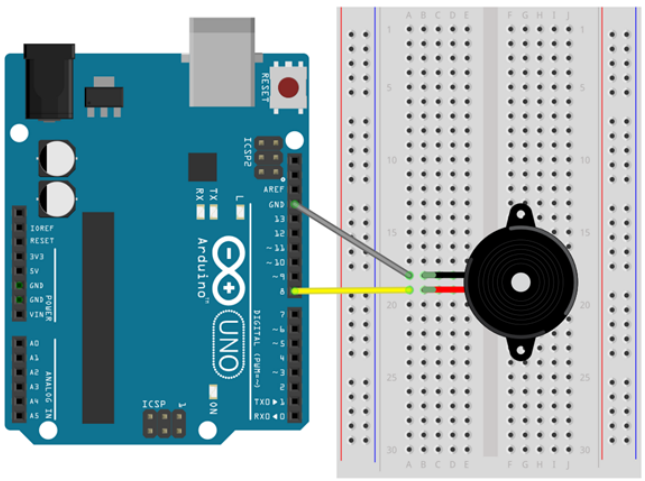
- 핀에 특정 주파수(50% 듀티 사이클)의 구형파(square wave)를 발생시킨다. 지속 시간을 정할 수 있으며, 따로 정하지 않는다면 noTone()을 부를 때까지 구형파가 지속된다. 핀을 피에조 버저 또는 스피커에 연결하여 tone을 연주할 수 있다.
- 한 번에 한 tone만 발생시킬 수 있다. 다른 핀에서 tone이 이미 연주되고 있으면, tone()을 새로 불러도 아무 일도 일어나지 않을 것이다. 같은 핀에서 tone이 연주되고 있으면, 주파수가 새로 설정된다.
- tone() 함수의 사용은 (Mega 이외의 보드에서) 3번과 11번 핀에서의 PWM 출력을 방해 한다. 31Hz보다 낮은 tone을 발생시키는 것은 불가능하다.
- tone() 함수는 기능을 수행하기 위해 하드웨어 PWM이 아닌 마이크로컨트롤러의 timer 사용한다. timer는 디지털 출력을 할 수 있는 어떤 핀에든 할당 가능하다. 따라서 tone() 함수를 사용하기 위해 굳이 PWM이 가능한 핀을 사용할 필요는 없다.

2) noTone() 함수

```
noTone(int pin)
```

- tone() 함수가 만들어 낸 구형파의 생성을 중단시킨다.

3) Passive piezo buzzer로 소리내기



```
#define LED 13
#define BUZZER 8

void setup() {
  pinMode(BUZZER, OUTPUT);
}

void loop() {
    tone(BUZZER, 440);
    delay(500);
    noTone(BUZZER);
    delay(500);

    //digitalWrite(LED, HIGH);
    tone(BUZZER, 440, 500);
    //digitalWrite(LED, LOW);
    delay(1000);
}
```

3) 다양한 소리 만들기

경고음

```
#define BUZZER 8

void setup() {
  pinMode(BUZZER, OUTPUT);
}

void loop() {
    tone(BUZZER, 600);
    delay(600);

    tone(BUZZER, 300);
    delay(600);
}
```

떨어지는 소리

```
#define BUZZER 8

void setup() {
  pinMode(BUZZER, OUTPUT);
}

void loop() {
  for(int f=20; f<5000; f+=20) {
    tone(BUZZER, f);
    delay(20);
  }
  for(int f=5000; f>20; f-=20) {
    tone(BUZZER, f);
    delay(20);
  }
}
```

외계 사운드

```
#define BUZZER 8

void setup() {
  pinMode(BUZZER, OUTPUT);
}

void loop() {
   // 외계의 소리
   int r = rand() % 5000;
   tone(BUZZER, r+20);
   delay(100);
}
```

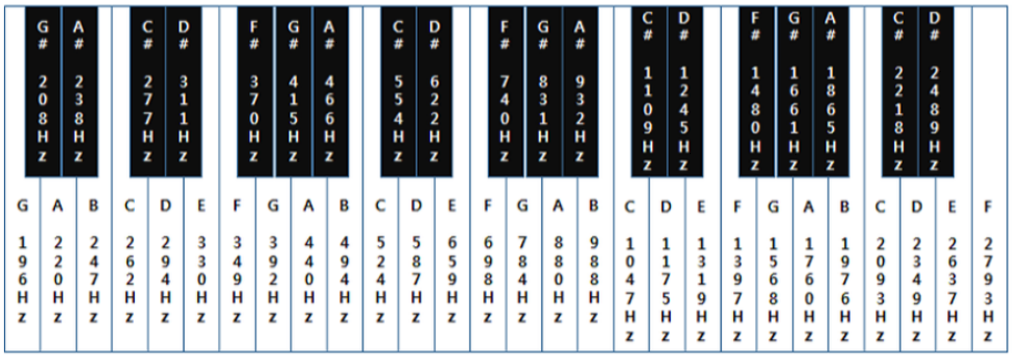
미사일 폭발

```
#define BUZZER 8

void setup() {
  pinMode(BUZZER, OUTPUT);
}

void loop() {
  // 떨어지는 소리
  for(int f=5000; f>20; f-=20) {
    tone(BUZZER, f);
    delay(20);
  }
  // 폭발 소리
  for(int i=0; i<300; i++) {
    int r=rand()%120+40;
    tone(BUZZER, r);
    delay(50);
  }
}
```

라. 음과 주파수

1) 음의 정의

- 주파수가 2배인 음의 폭 = 옥타브
- 4옥타브 라 = 440Hz



3) 음계별 주파수 계산법

총 12음계(라,라#,시,도,도#,레,레#,미,파,파#,솔,솔#,라) 존재한다.

옥타브 구간의 주파수를 동일한 간격으로 12분할하여 12음계에 배당함으로서 음계별 주파수를 계산하는 것이 가능하다.

옥타브	음계	주파수	계산
4	라	440	$440 \times 2^{0/12}$
4	라#	466	$440 \times 2^{1/12}$
4	시	494	$440 \times 2^{2/12}$
4	도	523	$440 \times 2^{3/12}$
4	도#	554	$440 \times 2^{4/12}$
4	레	587	$440 \times 2^{5/12}$
4	레#	622	$440 \times 2^{6/12}$
4	미	659	$440 \times 2^{7/12}$
4	파	698	$440 \times 2^{8/12}$
4	파#	740	$440 \times 2^{9/12}$
4	솔	784	$440 \times 2^{10/12}$
4	솔#	831	$440 \times 2^{11/12}$
5	라	880	$440 \times 2^{12/12}$
:	:	:	:

3) 음계별 주파수 계산결과

(단위 : Hz)

옥타브 / 음계	1	2	3	4	5	6	7	8
C(도)	32.7032	65.4064	130.8128	261.6256	523.2511	1046.502	2093.005	4186.009
C#	34.6478	69.2957	138.5913	277.1826	554.3653	1108.731	2217.461	4434.922
D(레)	36.7081	73.4162	146.8324	293.6648	587.3295	1174.659	2349.318	4698.636
D#	38.8909	77.7817	155.5635	311.1270	622.2540	1244.508	2489.016	4978.032
E(미)	41.2034	82.4069	164.8138	329.6276	659.2551	1318.510	2637.020	5274.041
F(파)	43.6535	87.3071	174.6141	349.2282	698.4565	1396.913	2793.826	5587.652
F#	46.2493	92.4986	184.9972	369.9944	739.9888	1479.978	2959.955	5919.911
G(솔)	48.9994	97.9989	195.9977	391.9954	783.9909	1567.982	3135.963	6271.927
G#	51.9130	103.8262	207.6523	415.3047	830.6094	1661.219	3322.438	6644.875
A(라)	55.0000	110.0000	220.0000	440.0000	880.0000	1760.000	3520.000	7040.000
A#	58.2705	116.5409	233.0819	466.1638	932.3275	1864.655	3729.310	7458.620
B(시)	61.7354	123.4708	246.9417	493.8833	987.7666	1975.533	3951.066	7902.133

4) 단순 음계 연주 코드

```
#define BUZ 8
#define MAX 48
unsigned int freqz[MAX];
bool isHalf[MAX] =
{0,1,0,0,1,0,1,0,0,1,0,1,0,1,0,0,1,0,1,0,0,1,0,1,
 0,1,0,0,1,0,1,0,0,1,0,1,0,1,0,0,1,0,1,0,0,1,0,1 };

void setup() {
        pinMode(BUZ, OUTPUT);
        pinMode(13, OUTPUT);
        Serial.begin(9600);
        for(int i=0; i<MAX; i++)  // 음계 생성
                freqz[i] = 220.0 * pow(2, i/12.0);
        for(int i=0; i<MAX; i++)  // 생성된 주파수 값 출력
                Serial.println(freqz[i], DEC);
}

void loop() {
  for(int i=0; i<MAX; i++) {
    if(isHalf[i]) continue; // 반음계는 건너뛰기

    int duration = 500;         // 500ms
    tone(BUZ, freqz[i]);
    delay(duration);
  }
}
```

5) 템포와 음표 길이를 고려하도록 계산된 코드

```
#define BUZ 8
#define MAX 48
unsigned int freqz[MAX];
bool isHalf[MAX] = // 반음계 이면 1, 아니면 0
{0,1,0,0,1,0,1,0,0,1,0,1,0,1,0,0,1,0,1,0,0,1,0,1,
 0,1,0,0,1,0,1,0,0,1,0,1,0,1,0,0,1,0,1,0,0,1,0,1 };

void setup() {
   pinMode(BUZ, OUTPUT);
   pinMode(13, OUTPUT);
   Serial.begin(9600);
   for(int i=0; i<MAX; i++)  // 음계 생성
      freqz[i] = 220.0 * pow(2, i/12.0);
   for(int i=0; i<MAX; i++)  // 생성된 주파수 값 출력
      Serial.println(freqz[i], DEC);
}

int TEMPO = 120; //(=BPM, beat per minute)
// 음 지속시간 계산 함수
// 4분 음표는 4, 8분음표는 8을 넘겨 주면 됨.
int getNoteDuration(int s) {
  // TEMPO = 60 이면, 1분에 사분음표 60개 연주
  // 따라서 사분음표 길이는 1000ms 가 됨.
  // TEMPO = 120이면, 4분음표 길이는 500ms
  int d = 60.0/TEMPO *(4000/s);
  return d;
}

void loop() {
  TEMPO = 60;
  int len = sizeof(freqz)/sizeof(int);
  for(int i=0; i<len; i++) {
    // 한 옥타브 올라갈때 마다, 2배 빠른 템포로...
    if(i%7 == 6) TEMPO=TEMPO*2;

    int duration = getNoteDuration(4);      // 4분음표
    tone(BUZ, freqz[i]);
    delay(duration);
  }
}
```

마. 악보 연주

1) 문자열로 기록된 악보를 해석하는 CNote 클래스

```
// Programmed by akapo@naver.com(박정진)
// 2022.6.24.

/* 음계명
  DO(도) RE(레) MI(미) FA(파) SO(솔) LA(라) CI(시)
    DS(도샵) RS(레샵) FS(파샵) SS(솔샵) LS(라샵)
    RF(레플랫)       SF(솔플렛) 등

  표기법 예시
  3DO4 -> 3옥타브 도4분음표
*/
#include <string.h>

class CNote {
  public:
    CNote();
    void  score(char *str);
    void  tempo(int t);
    bool  next();
    char* note();
    int   pitch();
    int   length();
    int   octave();
    int   freq(char* note);

  private:
    char *_score_origin;   // 악보 문자열 원본
    char *_score;          // 악보(플레이 하면서 오염됨)
    char *_score_head;     // 악보 시작 지점
    int  _tempo;
    bool _auto_replay;
    char *_cnote; // current_note
    const char *_delimiter =" ";
    int _note_octave;
    int _note_pitch;
    int _note_length;
};

CNote::CNote() {
  _auto_replay = true;
  _tempo = 120;
  _score_head = _score = NULL;
  _note_pitch = 0;
  _note_length = 0;
  _note_octave = 0;
}

void CNote::score(char *str) {
  _score_origin = str;
```

```
  if(_score_head != NULL)
    free(_score_head);

  _score = malloc(strlen(str)*sizeof(char));
  strcpy(_score, str);
  _score_head = _score;
  _cnote = NULL;
}

void CNote::tempo(int t) {
  _tempo = t;
}


bool CNote::next() {
  if(_cnote == NULL)   // 처음인 경우
    _cnote = strtok(_score, _delimiter);
  else
    _cnote = strtok(NULL, _delimiter);

  if(_cnote == NULL && _auto_replay == true) {
    score(_score_origin);
    return next();
  }

  if(_cnote != NULL)
    _cnote = strupr(_cnote);
  else
    return false;

  char* note = _cnote;

  if(isdigit(note[0])) {
    _note_octave = note[0]-'0';
    if(! (0 <= _note_octave && _note_octave <=9))
      _note_octave = 0;
    note++;
  }
  else if(note[0]=='-') {
    _note_octave--;
    note++;
  }
  else if(note[0]=='+') {
    _note_octave++;
    note++;
  }

  if(strlen(note) <= 0) {
    next();
    return true;
  }

  _note_pitch = freq(note);
  note += 2;
```

```
  int len = atoi(note);
  _note_length = 60.0/_tempo *(4000.0/len);

  if(note[strlen(note)-1]=='.')
    _note_length = _note_length*1.5f;

  return true;
}


int CNote::freq(char* note) {
  int pitch = 0;
  int nNote = -1;
  switch(note[0]) {
    case 'D': nNote = 3; break;
    case 'R': nNote = 5; break;
    case 'M': nNote = 7; break;
    case 'F': nNote = 8; break;
    case 'S': nNote = 10; break;
    case 'L': nNote = 12; break;
    case 'C': nNote = 14; break;
  }

  if(note[1] =='S')
    nNote++;
  else if(note[1] =='F')
    nNote--;

  if(nNote >= 0) {
    float z = ((octave()-1)*12 + nNote)/12.0;
    pitch = 27.5*pow(2, z)+0.49;  // 27.5 = 0 octave LA
  }
  else
    pitch = 0;

  return pitch;
}

char* CNote::note() {
  return _cnote;
}

int CNote::length() {
  return _note_length;
}

int CNote::pitch() {
  return _note_pitch;
}

int CNote::octave() {
  return _note_octave;
}
```

2) 악보를 문자열로 변환하기

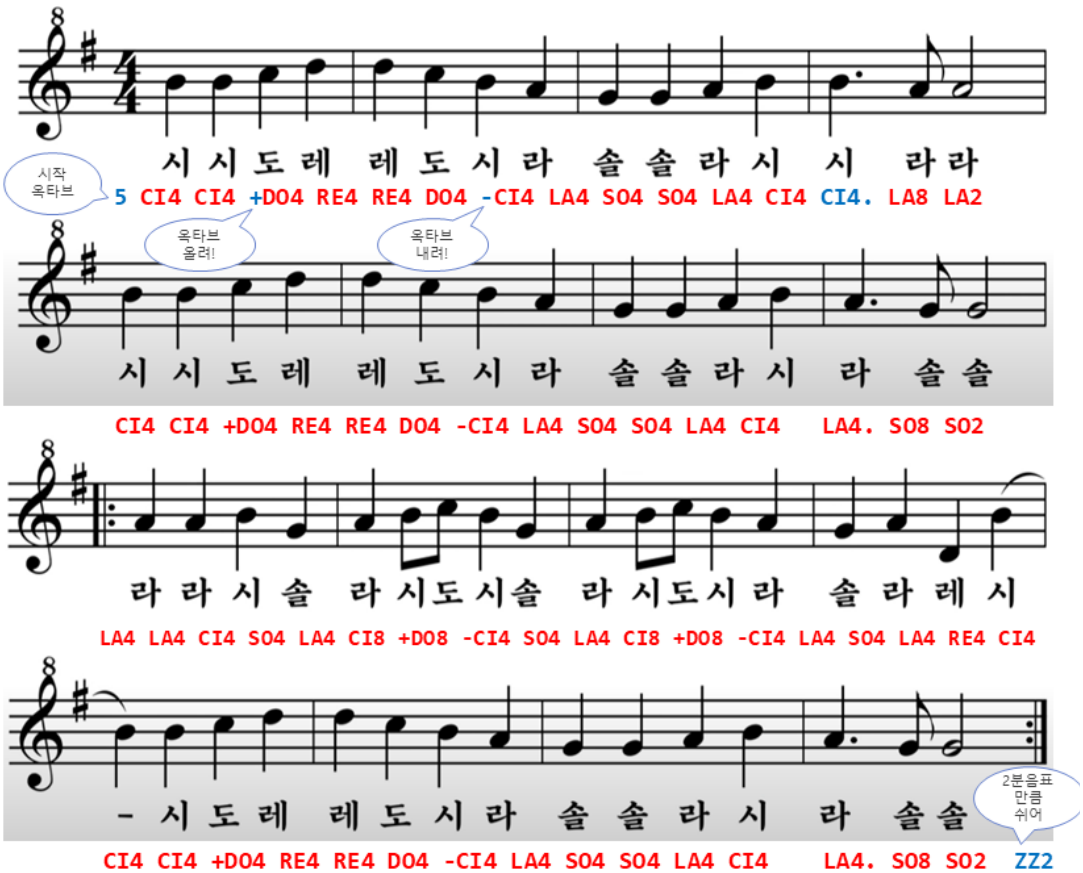
가) 음계표

계명	도	도#	레	레#	미	파	파#	솔	솔#	라	라#	시
문자열	DO	DS	RE	RS	MI	FA	FS	SO	SS	LA	LS	CI
계명		레b		미b			솔b		라b		시b	
문자열		RF		MF			SF		LF		CF	

나) 사용 예

도4분음표	레#2분음표 쩜	4분음 쉼표	한 옥타브 올려서 미8분음표쩜	한 옥타브 내려서 시8분음표	5옥타브 라4분음표
DO4	RE#2.	ZZ4	+MI8.	-CI8	5LA4

다) 변환 예시



라) 베토벤 환희의 송가 연주

```
// ※ CNote 클래스와 함께 사용해야 함.
#define BUZ  11

// 떳다 떳다 비행기
const char *AIRPLANE_SCORES =
  "5 mi8. re16 do8 re8 mi8 mi8 mi4 re8 re8 re4 mi8 mi8 mi4 mi8. \
   re16 do8 re8 mi8 mi8 mi4 re8 re8 mi8. re16 do2";

// 베토벤 환희의 송가
const char *BEETHOVEN_SCORES =
  "4 CI4 CI4 +DO4 RE4 RE4 DO4 -CI4 LA4 SO4 SO4 LA4 CI4 CI4. LA8 LA2 \
   CI4 CI4 +DO4 RE4 RE4 DO4 -CI4 LA4 SO4 SO4 LA4 CI4 LA4. SO8 SO2 \
   LA4 LA4 CI4 SO4 LA4 CI8 +DO8 -CI4 SO4 LA4 CI8 +DO8 -CI4 LA4 SO4 LA4 RE4 CI4 \
   CI4 CI4 +DO4 RE4 RE4 DO4 -CI4 LA4 SO4 SO4 LA4 CI4 LA4. SO8 SO2 \
   LA4 LA4 CI4 SO4 LA4 CI8 +DO8 -CI4 SO4 LA4 CI8 +DO8 -CI4 LA4 SO4 LA4 RE4 CI4 \
   CI4 CI4 +DO4 RE4 RE4 DO4 -CI4 LA4 SO4 SO4 LA4 CI4 LA4. SO8 SO2 ZZ2";

CNote cnote;

void setup() {
  pinMode(BUZ, OUTPUT);

  Serial.begin(9600);
  cnote = CNote();
  cnote.score(BEETHOVEN_SCORES);
  cnote.tempo(120);
}

void loop() {
  while(cnote.next())  {
    Serial.print(cnote.note());
    Serial.print(" ");
    Serial.print(cnote.pitch(), DEC);
    Serial.print(" ");
    Serial.println(cnote.length(), DEC);

    tone(BUZ, cnote.pitch());
    delay(cnote.length());
    noTone(BUZ);
  }
}
```

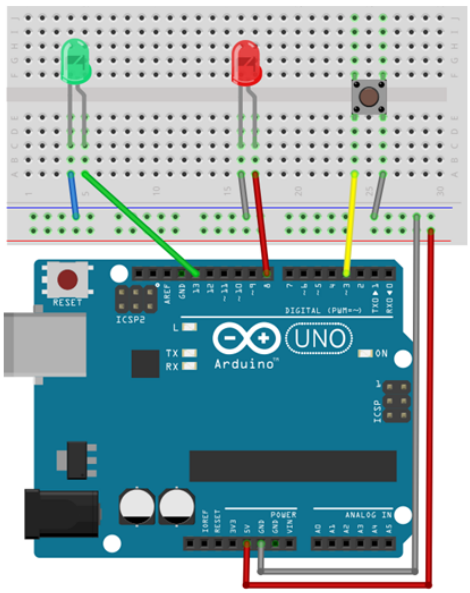
제5장 응용 프로젝트

1. 멀티태스킹

가. delay() 함수의 덫

1) 멀티태스킹 문제

- 회로 구성



- 아래와 같은 일을 하는 회로를 구성하고, 제어 프로그램을 작성하시오.
  - 13번 핀에 연결된 내장 LED는 2초 간격으로 계속 깜박인다.
  - 3번 핀에 연결된 버튼이 ON 되면 8번 포트에 연결된 RED가 즉시 켜진다.
  - 3번 핀에 연결된 버튼이 OFF 되면 8번 포트에 연결된 RED가 즉시 꺼진다.

2) 흔히 생각하는 구현 (하지만 오답)

```
#define BTN  3
#define RED  8
#define LED 13

bool isLedOn = false;

void setup() {
    pinMode(RED, OUTPUT);
    pinMode(LED, OUTPUT);
    pinMode(BTN, INPUT_PULLUP);
    // 버튼이 눌리면 LOW
    // 버튼이 떨어지면 HIGH
}
```

```
void invertLed() {
   isLedOn = !isLedOn;   // 상태 반전
   digitalWrite(LED, isLedOn);
   delay(2000);
   // 위 delay()가 실행되는 동안 먹통이 됨
}

void setRedLed() {
  bool b = digitalRead(BTN);
  digitalWrite(RED, !b);
}

void loop() {
  invertLed();
  setRedLed();
}
```

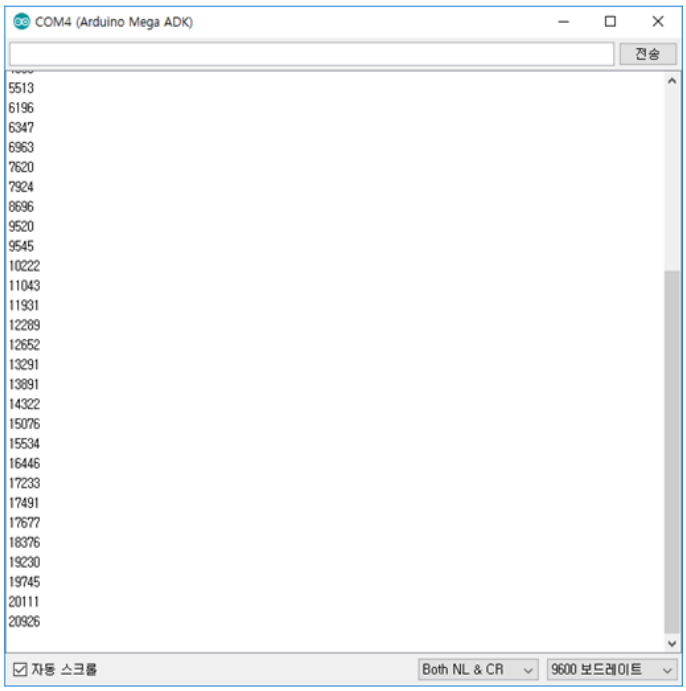
나. 서로의 작업을 방해하지 않는 멀티태스킹 구현

1) millis() 함수

millis()는 이두이노 보드가 현재 프로그램을 시작한이래 몇 밀리 초가 흘렀는지 숫자로 반환한다. 이 숫자는 대략 50일이 지나면, 오버플로 된다(다시 0으로 되돌아감). millis() 함수의 리턴 값이 unsigned long 임에 유의해야 한다. int 와 같은 작은 데이터 타입을 사용해서는 안 된다. 아래와 같은 방식으로 사용한다.

```
unsigned long currentTime= millis();
```

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

void loop() {
    unsigned long m = millis();
    Serial.println(m);
    int r = rand()%1000;
    delay(r);
}
```



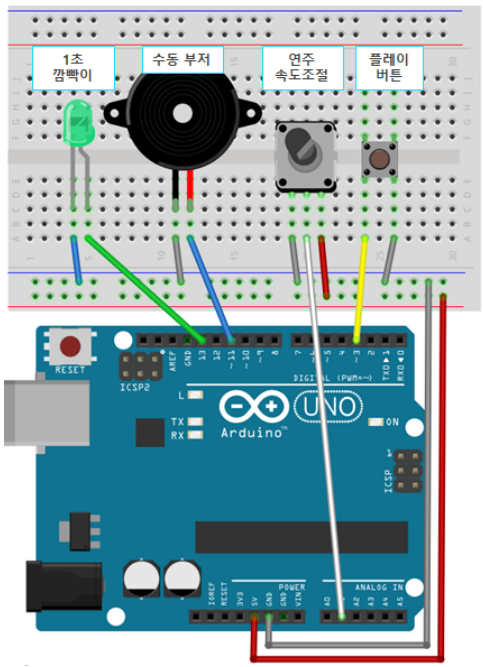
2) 올바른 구현

```
#define BTN  3
#define RED  8
#define LED 13

bool isLedOn= false;

void setup() {
    pinMode(RED, OUTPUT);
    pinMode(LED, OUTPUT);
    pinMode(BTN, INPUT_PULLUP);
    // 버튼이 눌리면 LOW
    // 버튼이 떨어지면 HIGH
}

void setRedLed() {
```

```
  bool b = digitalRead(BTN);
  digitalWrite(RED, !b);
}


void invertLed() {
  static unsigned long preTime= 0;
  unsigned long curTime= millis();  // 현재 시각은?

  // (현재시각 > 이전시각+2초)
  if(curTime> preTime+2000) {  // 2초 지났니?
    isLedOn= !isLedOn;
    digitalWrite(LED, isLedOn);

    preTime= curTime;// LED상태를 바꾼 시간을 기록
  }
}

void loop() {
  invertLed();
  setRedLed();
}
```

다. 음악 플레이어 만들기

1) 문제



1. 13번 핀에 연결된 LED가 1초에 한 번씩 반전된다.
   - 온->오프 | 오프->온
2. 플레이 버튼을 누를 때 마다 음악 연주 상태를 변경한다.
   - 음악 연주 중에 버튼을 누르면 연주 중지
   - 연주 중지 상태에서 버튼을 누르면 연주 시작

- 2. 연주속도 조절 가변 저항으로 연주 속도를 조절
  - 왼쪽으로 돌리면 연주가 느려짐
  - 오른쪽으로 돌리면 연주가 빨라짐

2) 프로그래밍

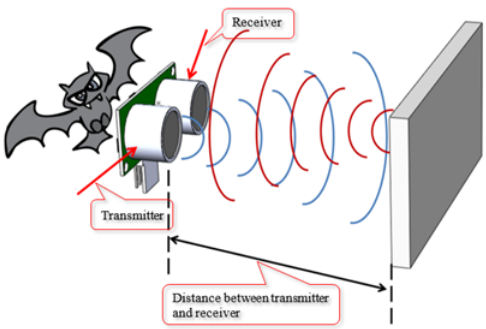
```
#define LED  13
#define BUZ  11
#define BTN   3
#define POT  A1

#include <Bounce2.h>
CNotecnote;
Bounce db_btn= Bounce();
bool isPlayable= true;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BUZ, OUTPUT);
  pinMode(BTN, INPUT_PULLUP);
  db_btn.attach(BTN);
  db_btn.interval(50);
  Serial.begin(9600);
  cnote= CNote();
  cnote.score(BEETHOVEN_SCORES);
  cnote.tempo(120);
}

// 동시에 처리해야 할 각 작업을 함수로 만드는데 delay() 함수를 사용하지 않고
// 빠르게 작업을 마무리하고 즉시 반환되도록 만들어야 한다.

void invertLed() {  // 13번 LED 1초 마다 토글
  static unsigned long preTime= 0;
  unsigned long curTime= millis();  // 현재 시각은?
  if(curTime> preTime+1000) {
    digitalWrite(LED, !digitalRead(LED));
    preTime= curTime;
  }
}

void checkBtn() {  // 연주를 시작/중단 시키는 버튼
  db_btn.update();
  if(db_btn.fell())
    isPlayable= !isPlayable;
}

void checkPot() {  // 가변 저항값으로속도 조절
  int pot = analogRead(POT);
  cnote.tempo(map(pot, 0, 1024, 60, 480));
}

void player() {
  static unsigned long prevPlayTime=0;
  static unsigned long noteLength=0;
  int pitch;
```

```
  if(! isPlayable) {
    noTone(BUZ);
    return;
  }

  // 현재 연주 중인 음표의 연주 시간이 아직 시간이 지나지 않았는가?
  if(millis() < prevPlayTime + noteLength) return;
  else noTone(BUZ);

  cnote.next();   // 다음 음표
  pitch = cnote.pitch();    // 그 음표의 음조
  noteLength = cnote.length();  // 그 음표의 길이

  if(pitch >= 31) {// tone() 함수는 31Hz 이상만 소리를 낼 수 있다.
    tone(BUZ, pitch, noteLength);
    prevPlayTime = millis();
  }
}

void loop() {
  invertLed();
  checkBtn();
  checkPot();
  player();
}
```

2. 자동차 후방감지기 만들기

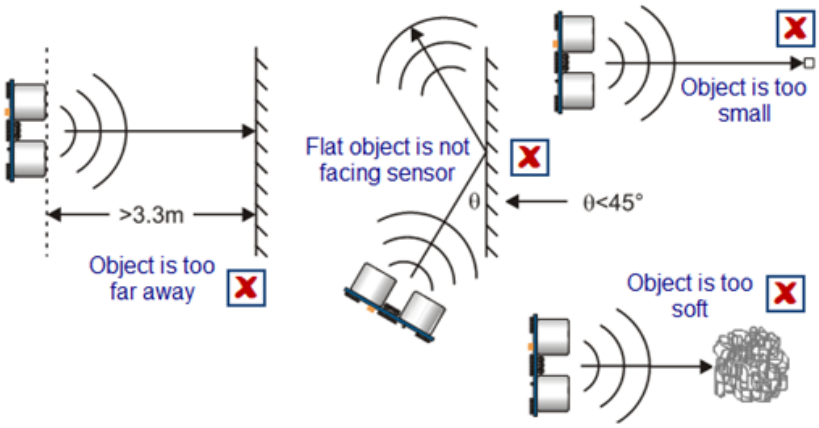
가. 초음파 센서

1) 센서 외관과 작동원리



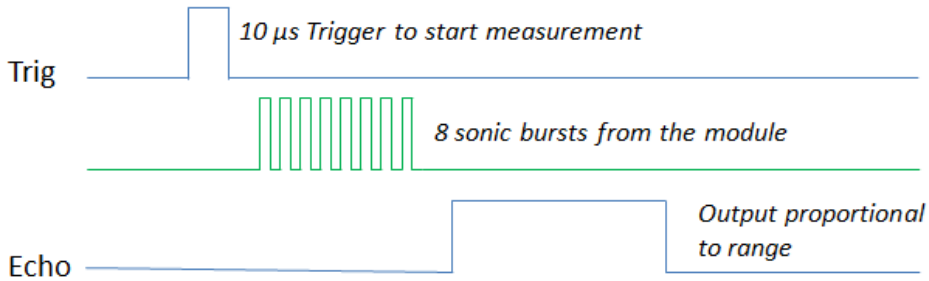
초음파 센서의 독특한 외관 때문에 아두이노로 로봇을 만들 경우 머리 부분에 배치되어 양쪽 눈처럼 활용되는 경우가 많다. 이 로봇 눈처럼 보이는 한 쌍의 배럴은 양쪽의 역할이 서로 다른데 T라고 적혀 있는 쪽이 초음파 송신부(Transmitter)로 초음파가 발사 되는 부분이며, R이라고 적혀 있는 쪽이 초음파 수신부(Reciver)로 초음파의 반향을 수신하는 부분이다.

2) 감지 한계



초음파 센서는 물체와의 거리가 너무 멀거나(3,3m), 물체가 너무 작은 경우, 그리고 물체가 너무 부드러워서 반향을 만들지 못하는 경우에는 거리 측정이 불가능할 수 있다.

3) 센서의 입출력 타이밍 다이어그램



초음파 센서는 Tigger 라인에 10㎲ 이상의 상승 에지 신호를 받으면 작동이 개시된다. 8개의 40,000Hz 음파를 송신부에서 발사한 뒤 반향이 되돌아오기까지의 시간 측정을 시작한다. 얼마 뒤 수신부에 초음파 반향이 수신되면 초음파가 왕복하는데 걸린 시간을 Echo 라인에 상승 에지로 알려준다. 바로 이 상승 에지가 지속되는 시간이 왕복 시간이다. 아두이노에서는 pulseIn() 함수를 통해 Echo 라인의 상승 에지 지속 시간을 측정할 수 있다.

4) 거리 계산 방법

음속: 340m/s = 34000cm/s = 0.034cm/us

v = d / t

d = v * t

- 2d = 0.034cm x t (d: 물체까지의 거리(㎝), t: 초음파의 왕복 거리(㎲))
- d = (0.034cm * t) / 2

5) pulseIn() 함수

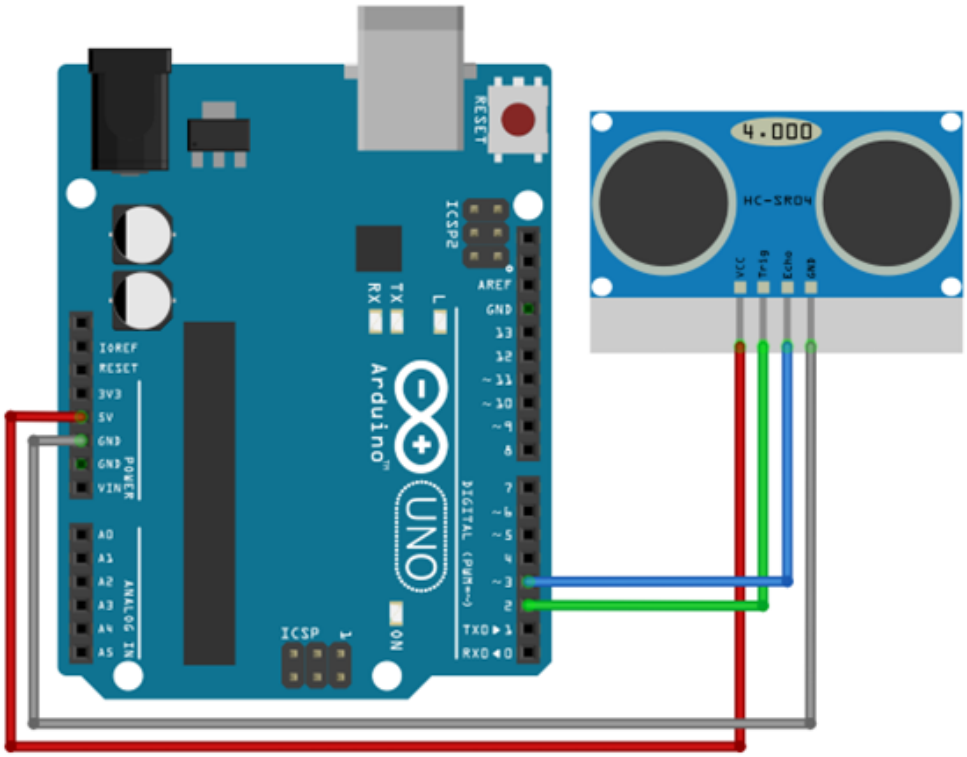
```
pulseIn(int pin, int value)
pulseIn(int pin, int value, unsigned long timeout)
```

핀에서 펄스(HIGH 또는 LOW)를 읽는다. 예를 들어, value가 HIGH이면, pulseIn()은 핀이 HIGH가 될 때까지 기다렸다가 시간을 재기 시작하고 핀이 LOW가 되면 시간 측정을 멈춘다. 이렇게 측정된 펄스의 길이를 마이크로초 단위로 반환한다. 정해진 timeout 안에 펄스가 시작되지 않으면 0을 반환한다.

나. 실습

1) 거리 측정기

가) 회로 결선



나) 프로그래밍

```
#define TRIG 2
#define ECHO 3

void setup() {
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  Serial.begin(9600);
```

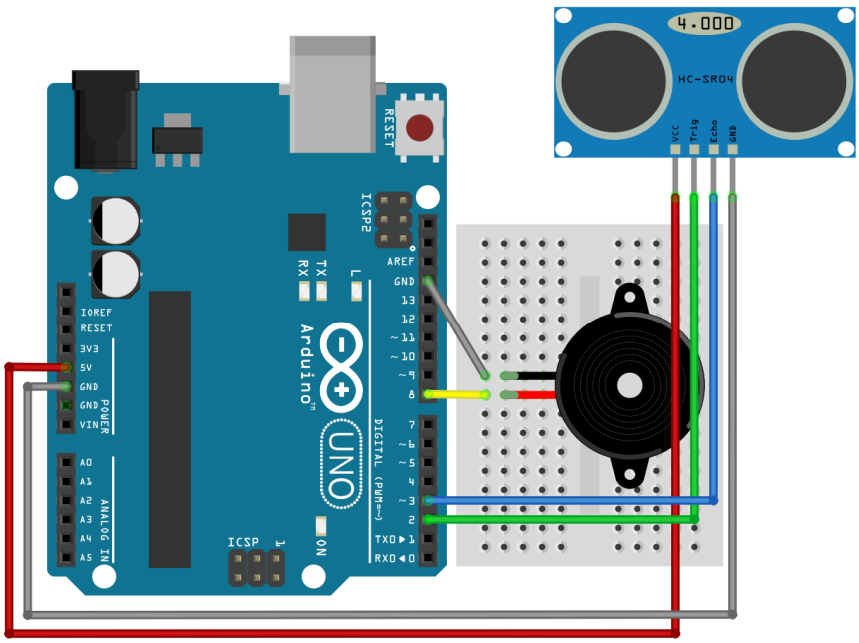
```
}

void loop() {
  long duration;
  int distanceCm;

  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  duration = pulseIn(ECHO, HIGH);
  distanceCm= 0.034*duration/2;
  Serial.print(distanceCm);
  Serial.println("cm");

  delay(300);
}
```

2) 자동차 후방 감지 센서

아두이노, 초음파 센서, BUZZER를 활용하여, 자동차 후방감지 센서와 같은 역할을 하는 장치를 만드시오.

- 물체와의 거리가 3미터 이하인 경우에만 소리로 물체까지의 거리를 알려준다.
- 물체와 거리가 가까울 수록 소리 출력의 빈도가 빨라져야 한다.

가) 회로 결선

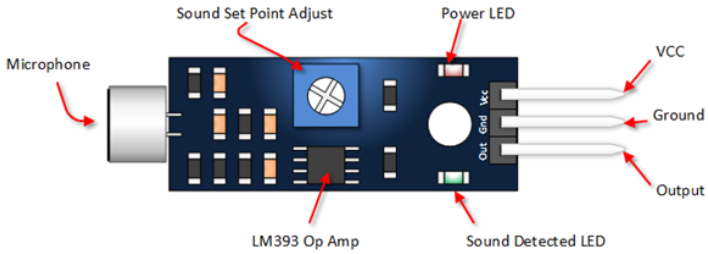


나) 프로그래밍

```
// 자동차 후방감지기
#define TRIG 2
#define ECHO 3
#define BUZZER 8

void setup() {
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  pinMode(BUZZER, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  long duration;
  int distanceCm;

  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  duration = pulseIn(ECHO, HIGH);
  distanceCm= duration*0.034/2;
  Serial.print(distanceCm);
  Serial.println("cm");

  if(distanceCm < 300) {
    int duration = 30+(distanceCm * 2.5);
    tone(BUZZER, 440);
    delay(duration);
    noTone(BUZZER);
    delay(duration);
  }
}
```

3. 사운드 센서

가. 센서의 외관

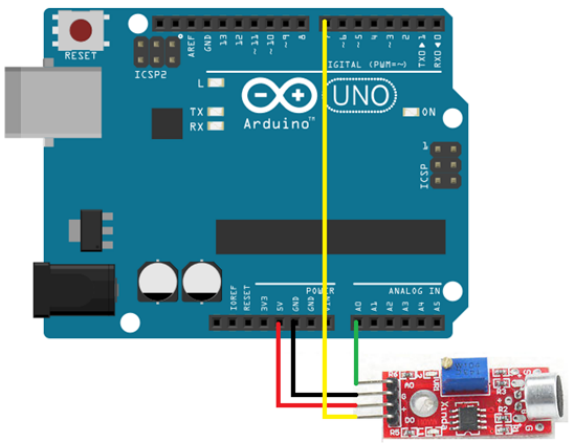


나. 센서의 단자

단자	설명
VCC	5 Vdc from your Arduino
Ground	GND from your Arduino
A0	Connect to Analog Input Pin
D0	Connect to Digital Input Pin
Power LED	Illuminates when power is applied
Sound Detection LED	Illuminates when sound is detected
Sound Set Point Adjust	CW = More Sensitive CCW = Less Sensitive

모듈에 따라 아날로그 단자(A0)가 없는 경우도 존재한다.

다. 소리 입력상태 출력



```
#define SAIN  A0
#define SDIN  7

void setup() {
  pinMode(SDIN, INPUT);
  Serial.begin(9600);
}

void loop() {
  bool sd = digitalRead(SDIN);
  int  sa = analogRead(SAIN);
  Serial.print("digital: ");
  Serial.println(sd, DEC);
  Serial.print("analog: ");
  Serial.println(sa, DEC);
  delay(300);
}
```

라. 박수 두 번으로 #13 LED 토글

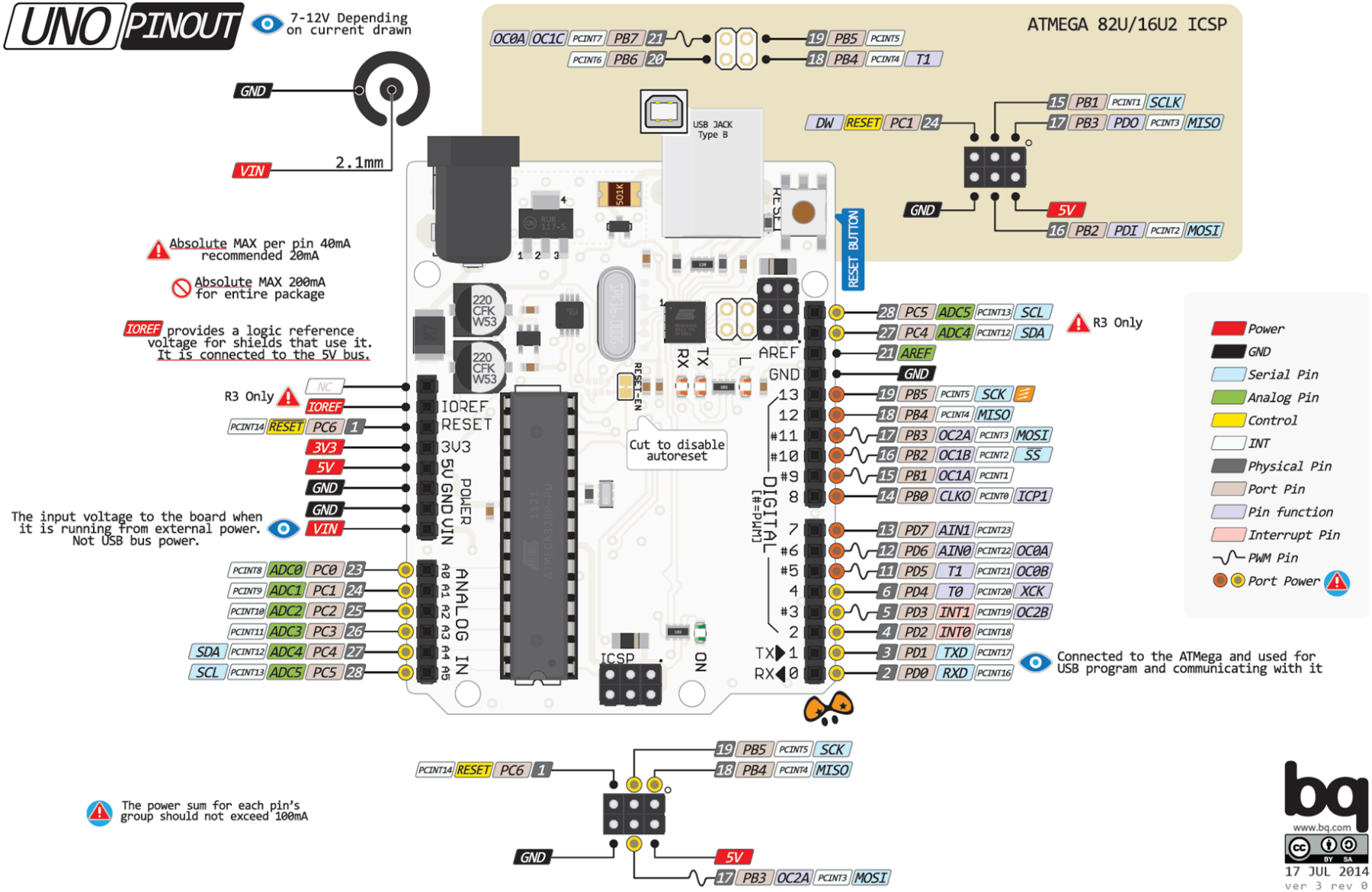
1) delay() 이용 버전

```
#define SND A0
#define LED 13
#define THREADSHOLD 100

int snd_count = 0;

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

void loop() {
  read_sound();

  if(snd_count == 2) {
    digitalWrite(LED, !digitalRead(LED));
    snd_count=0;
  }
}

void read_sound() {
  int sound = analogRead(SND);
  if(sound > THREADSHOLD) {
    snd_count++;
    Serial.print("snd_count: ");
    Serial.println(snd_count, DEC);
    // 0.2초 안에 다시 감지되지 않도록
    delay(200);
  }
}
```

2) 멀티태스킹 가능 버전

```
#define SND A0
#define LED 13
#define THREADSHOLD 100

int snd_count = 0;

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

void loop() {
  read_sound();
```

```
  if(snd_count == 2) {
    digitalWrite(LED, !digitalRead(LED));
    snd_count=0;
  }
}

void read_sound() {
  static unsigned long prev_time = -200;
  if(millis() - prev_time > 500) {
    snd_count=0;
  }
  if(millis() - prev_time > 200) {
    int sound = analogRead(SND);
    //Serial.println(sound, DEC);
    if(sound > THREADSHOLD) {
      snd_count++;
      Serial.print("snd_count: ");
      Serial.println(snd_count, DEC);
      prev_time = millis();
    }
  }
}
```


UNO PINOUT
7-12V Depending on current drawn
GND
VIN
2.1mm
ATMEGA 82U/16U2 ICSP
OC0A OC1C PCINT7 PB7 21
PCINT6 PB6 20
19 PB5 PCINT5
18 PB4 PCINT4 T1
USB JACK Type B
DW RESET PC1 24
15 PB1 PCINT1 SCLK
17 PB3 PDO PCINT3 MISO
GND
5V
16 PB2 PDI PCINT2 MOSI
RESET BUTTON
Absolute MAX per pin 40mA recommended 20mA
Absolute MAX 200mA for entire package
IOREF provides a logic reference voltage for shields that use it. It is connected to the 5V bus.
220 CFK W53
220 CFK W53
RESET-EN
Cut to disable autoreset
TX
RX
L
R3 Only
NC
IOREF
PCINT14 RESET PC6 1
3V3
5V
GND
GND
VIN
POWER
IOREF
RESET
3V3
5V GND VIN
The input voltage to the board when it is running from external power. Not USB bus power.
PCINT8 ADC0 PC0 23
PCINT9 ADC1 PC1 24
PCINT10 ADC2 PC2 25
PCINT11 ADC3 PC3 26
SDA PCINT12 ADC4 PC4 27
SCL PCINT13 ADC5 PC5 28
ANALOG IN
A0 A1 A2 A3 A4 A5
ICSP
ON
AREF
GND
13
12
#11
#10
#9
8
7
#6
#5
4
#3
2
TX 1
RX 0
DIGITAL (PWM=#)
28 PC5 ADC5 PCINT13 SCL
27 PC4 ADC4 PCINT12 SDA
21 AREF
GND
19 PB5 PCINT5 SCK
18 PB4 PCINT4 MISO
17 PB3 OC2A PCINT3 MOSI
16 PB2 OC1B PCINT2 SS
15 PB1 OC1A PCINT1
14 PB0 CLKO PCINT0 ICP1
13 PD7 AIN1 PCINT23
12 PD6 AIN0 PCINT22 OC0A
11 PD5 T1 PCINT21 OC0B
6 PD4 T0 PCINT20 XCK
5 PD3 INT1 PCINT19 OC2B
4 PD2 INT0 PCINT18
3 PD1 TXD PCINT17
2 PD0 RXD PCINT16
R3 Only
Connected to the ATMega and used for USB program and communicating with it
Power
GND
Serial Pin
Analog Pin
Control
INT
Physical Pin
Port Pin
Pin function
Interrupt Pin
PWM Pin
Port Power
PCINT14 RESET PC6 1
19 PB5 PCINT5 SCK
18 PB4 PCINT4 MISO
GND
5V
17 PB3 OC2A PCINT3 MOSI
The power sum for each pin's group should not exceed 100mA
bq
www.bq.com
17 JUL 2014
ver 3 rev 0

2022. 토요 정보아카데미 워크북

{ 아두이노 메이킹 }

발 행 인 백우정(충청북도교육연구정보원 원장)
제작총괄 신상규(충청북도교육연구정보원 정보교육부장)
제작지도 최경숙(충청북도교육연구정보원 교육연구사)

집필위원 **박정진**(흥덕고등학교)

발 행 일 2022년 8월
발 행 처 충청북도교육연구정보원 | www.cberi.go.kr
충청북도 청주시 서원구 청남로 1931(산남동 4-3)