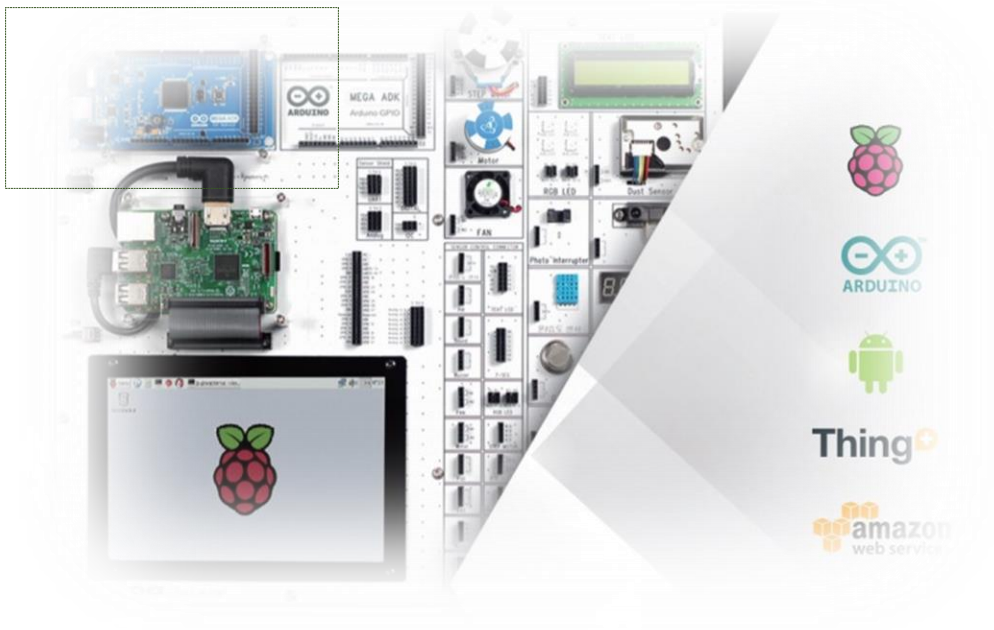
마이크로프로세서
(강의자료 #6)



교과목명 : 마이크로프로세서 (01)

담당교수 : 이 수 형

E-mail : soohyong@uu.ac.kr

교재명 : 스마트기기 개발을 위한 아두이노, 이수형. (LINC+ 사업단 배포)

수강생 안내

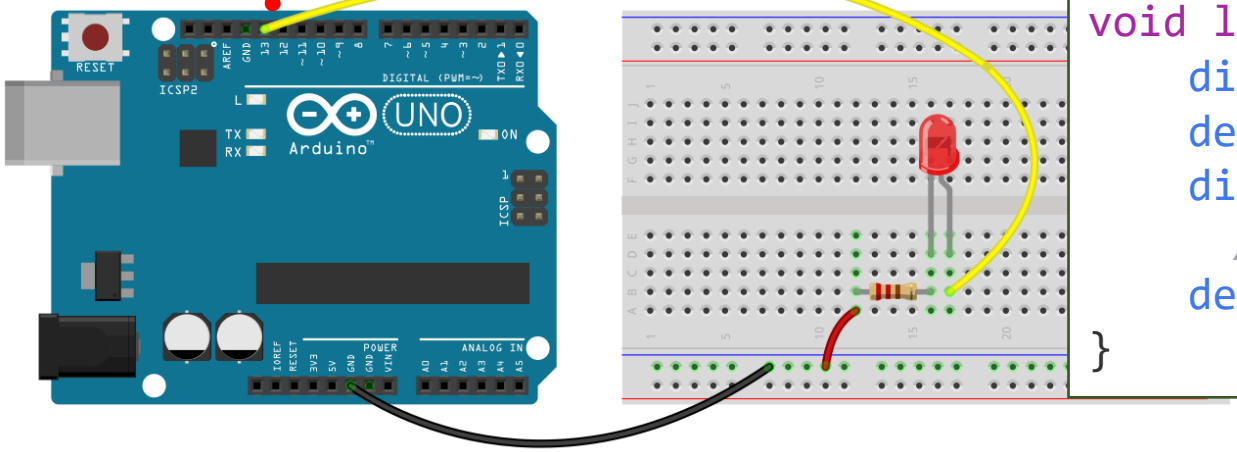
- 본 교과목은 기본적으로 “대면”수업입니다.
- ‘코로나바이러스감염증-19’ 등의 이유로 대면수업을 받지 못하는 학생들을 위해서 ‘비대면 실시간 수업’을 병행합니다. 기본적으로 대면 수업이므로 대면 수업에 참석한 학생들은 교실에서 출석을 체크하면 되며, 참석하지 못하고 ‘비대면 수업’을 듣는 학생들은 ‘실시간 온라인 강의’에 참석하면 출석이 반영됩니다.
- 수강생 여러분들은 강의 자료를 온라인 배포 및 게재 등의 행위를 할 시 저작권 문제가 발생할 수 있사오니 이에 유념하여 강의 자료를 공유하는 행위는 삼가 바랍니다.
- 교재는 “스마트기기 개발을 위한 아두이노”이며, 나누어준 키트는 각자 잘 관리하면서 실습실(대면) 및 집(비대면)에서 실험/실습을 수행할 수 있도록 진행합니다.

지난시간에는?

4.1 LED를 이용한 디지털 신호 출력

Review

- 회로 구성



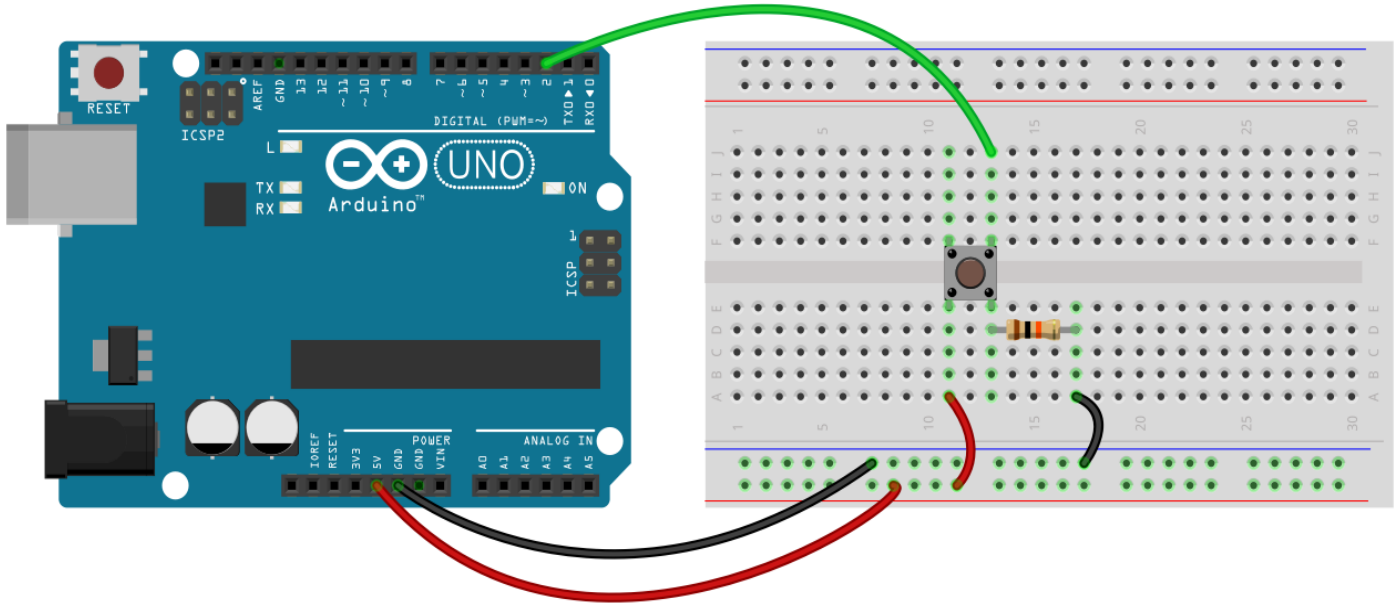
예제 4-1. 첫번째 스케치

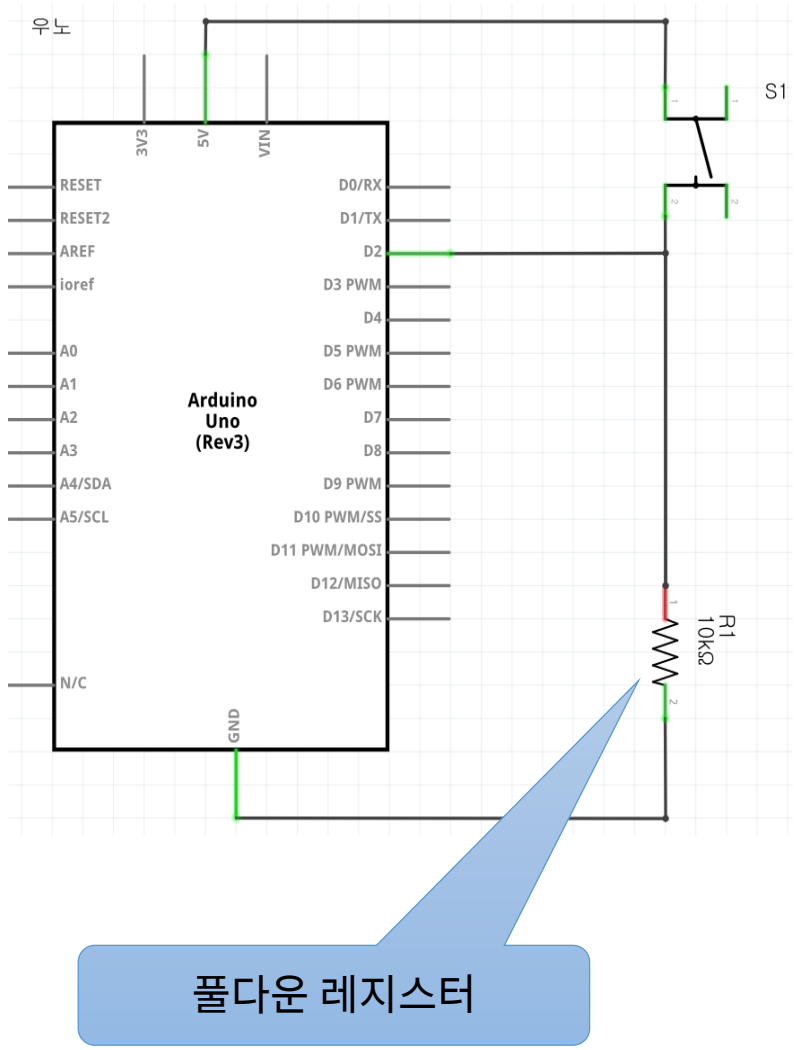
```
int led = 13;

// 처음 시작시 한번만 수행하는 함수, 설정을 담당한다
void setup() {
    pinMode(led, OUTPUT);      // 13번핀 출력설정
}

// 반복해서 호출되는 함수
void loop() {
    digitalWrite(led, HIGH);   // 13번 핀으로 5V출력
    delay(1000);               // 1000 ms = 1초 대기
    digitalWrite(led, LOW);
      // 13번 핀으로 0V출력
    delay(1000);               // 1초를 기다린다.
}
```

4.3 스위치를 이용한 디지털 입력

Review

- 회로도
  - 저항의 역할 : 풀다운(pull-down) 저항
  - 스위치가 Off인 경우에 GND로 연결 → LOW
  - 스위치 On인 경우에는 5V 인가 → HIGH





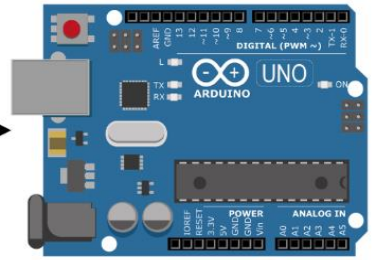
UART 통신

- 시리얼 통신
  - 아두이노와 PC와의 통신, Serial 객체 사용
  - UART(Universal Asynchronous Receiver and Transmittor)
  - 아두이노 : H/W 방식(디지털 포트 0번과 1번), 추가 시리얼통신은 S/W 방식
- 방법
  - 초기화
    - `Serial.begin(9600);`
  - 전송 (PC ← 아두이노)
    - `Serial.print(전송값);`
    - `Serial.println(...);`
  - 전송 (PC → 아두이노)
    - `if(Serial.available()) { char data = Serial.read(); ... }`

아두이노 IDE & 시리얼 모니터

USB 연결

시리얼 통신



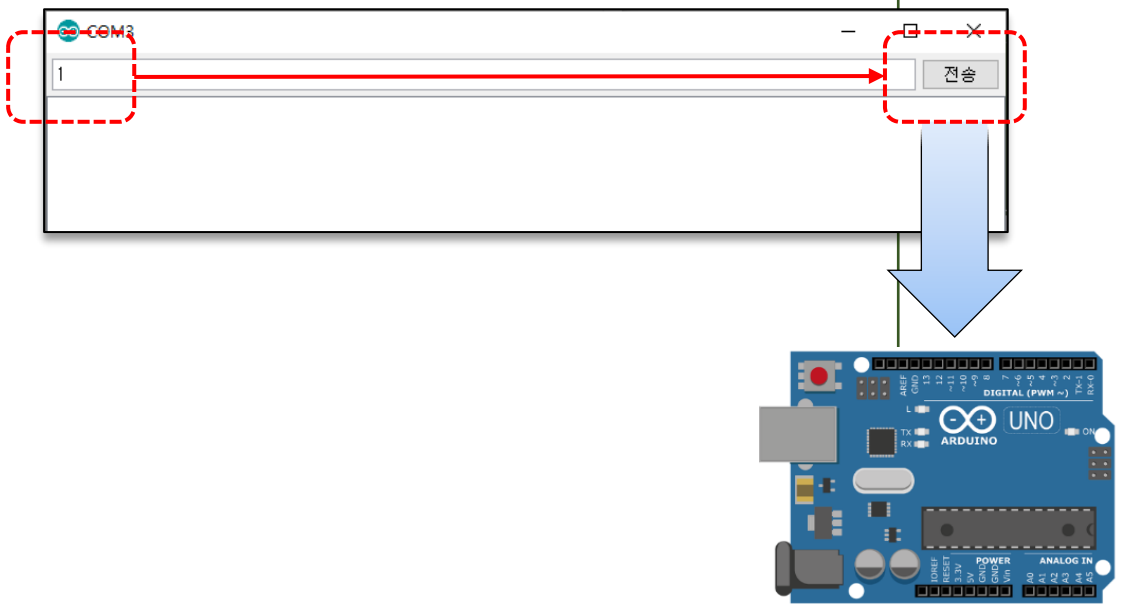
Serial 객체 사용

예제 4-3. 디지털 입력 프로그램

```
// 디지털 입/출력 핀 2번을 스위치의 입력으로 설정한다.
int pushButton = 2;

void setup() {
  // 시리얼 통신을 위하여 초기화하는 과정, 9600보레이트(baud-rate)로 설정한다.
  Serial.begin(9600);
  // 버튼 스위치가 연결된 2번 핀을 입력 모드로 설정한다.
  pinMode(pushButton, INPUT);
}

void loop() {
  // 현재 버튼이 연결되어 있는 핀의 값을 읽어들인다.
  int buttonState = digitalRead(pushButton);
  // 입력받은 값을 시리얼 통신을 통해서 PC로 전송한다.
  Serial.println(buttonState);
    // 입력 상태를 안정화하기 위하여 1미리초 동안 대기한다
  delay(1);
}
```

예제: 시리얼 입력을 통한 LED 제어

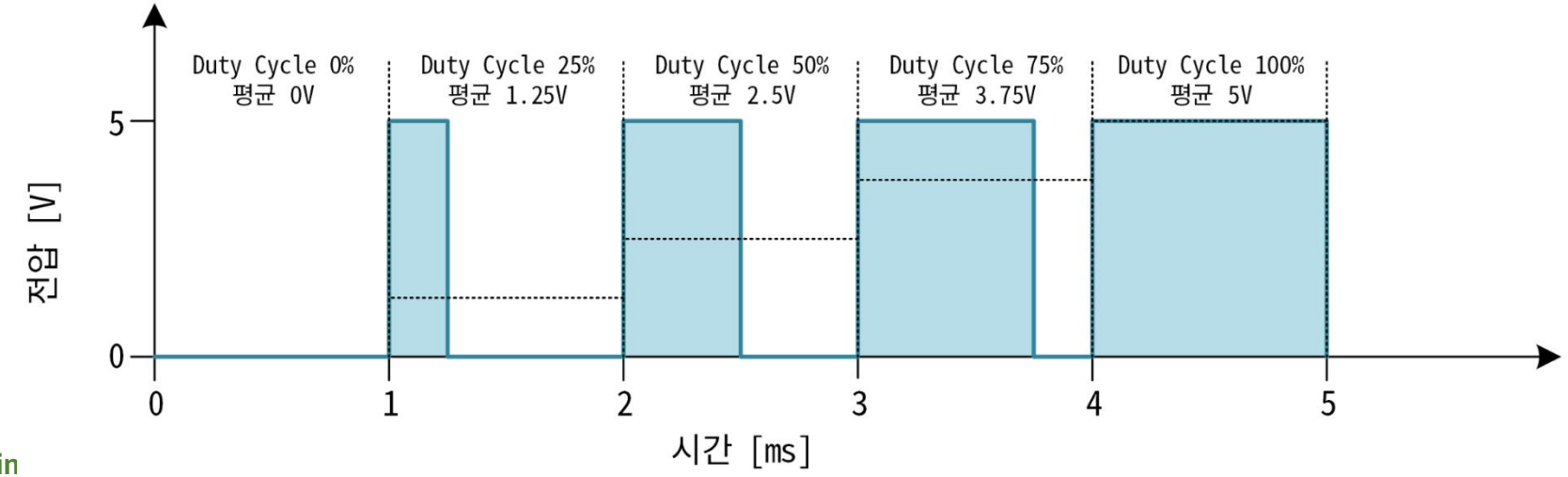
```
void setup() {
  // LED_BUILTIN : 내장된 LED 포트 번호 == 13
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  // 시리얼 통신 검사 후 데이터 읽기
  if(Serial.available()) {
    char ch = Serial.read();
    if(ch == '1')
      digitalWrite(LED_BUILTIN, HIGH);
    else if(ch == '2')
      digitalWrite(LED_BUILTIN, LOW);
  }
  delay(10);
}
```



5.1 아날로그 출력

Review

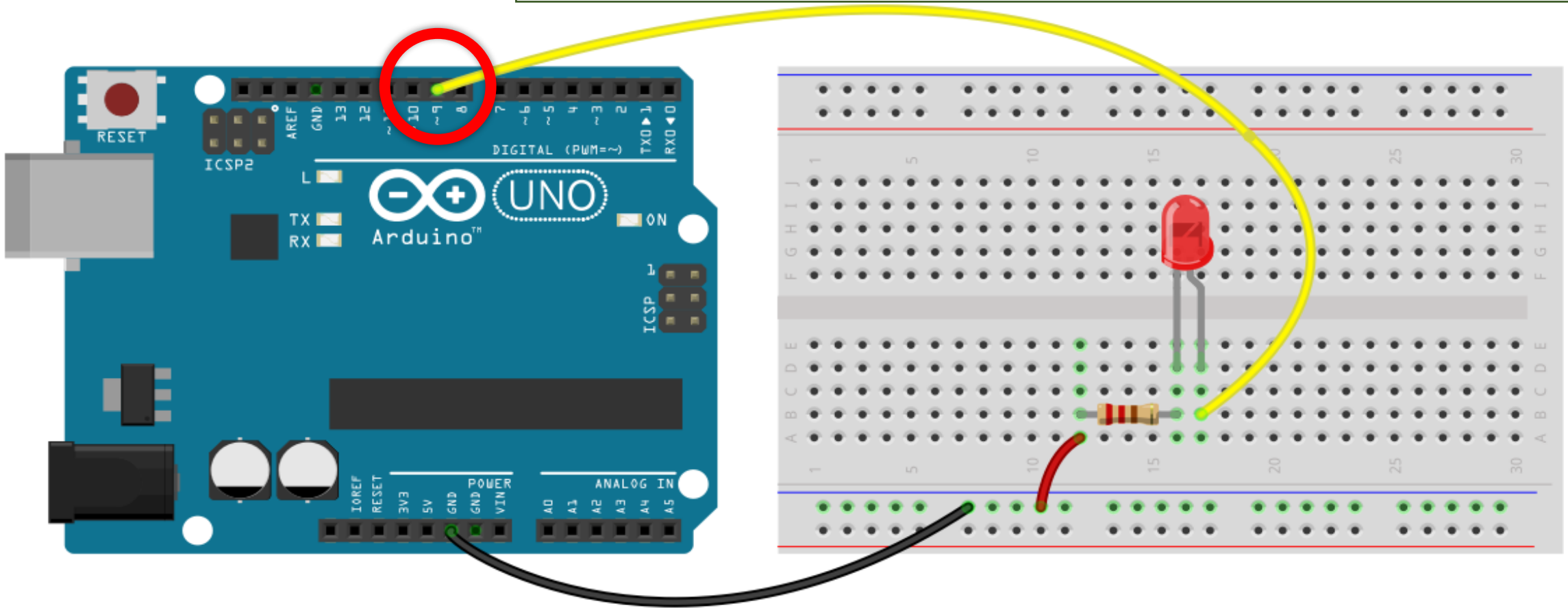
- 아두이노 우노의 입/출력 기능
  - 14개의 디지털 입/출력 기능, 6개의 아날로그 입력 기능
  - 아날로그 출력 기능 → PWM(Pulse Width Modulation) 출력으로 해결
  - PWM(Pulse Width Modulation)
    - ➢ 주기적인 사각형 펄스의 5V, 0V의 발생 시간을 조절 하면서 평균전압을 제어
    - ➢ 5V, 0V의 발생 시간의 비 : 듀티 사이클 (duty cycle)
    - ➢ 아두이노 : 6개의 디지털 출력핀이 가능 (~기호로 표시)
    - ➢ analogWrite(9, 127); → 9번 핀으로 50%듀티 사이클 출력



아날로그 입력

- 회로 구성

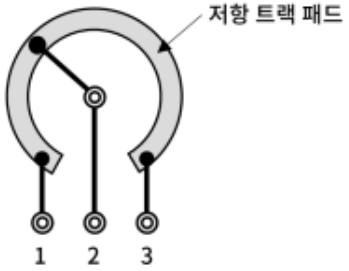
```
digitalWrite(9, HIGH); →  5V digital signal
analogWrite(9, 0);     →  0% Duty Cycle Pulse (≈ 0V)
analogWrite(9, 128);   →  50% Duty Cycle Pulse (≈ 2.5V)
analogWrite(9, 255);   →  100% Duty Cycle Pulse (≈ 5V)
```

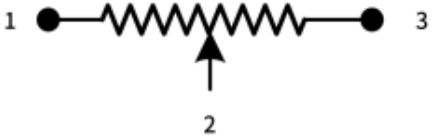


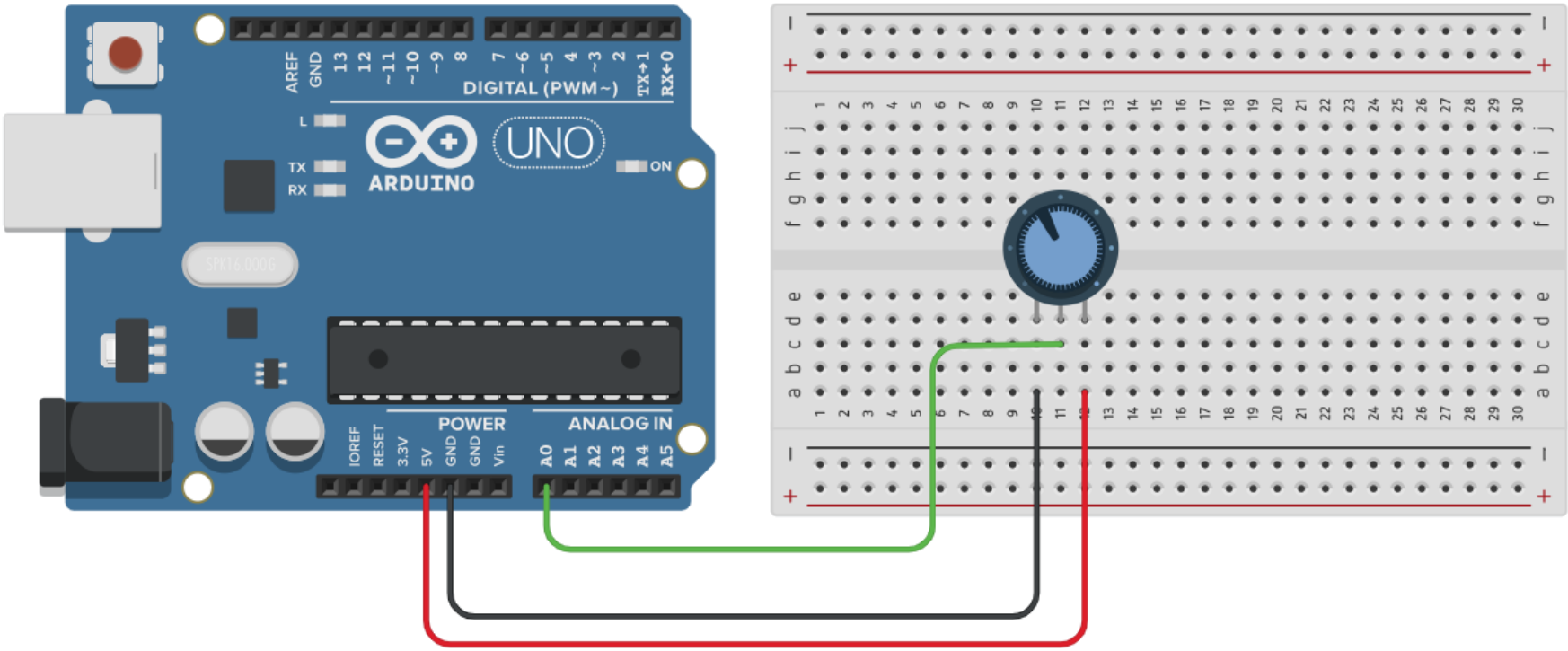
5.2 포텐쇼미터를 이용한 아날로그 입력

Review

- 포텐쇼미터(potentiometer)
  - 가변저항
- 회로구성

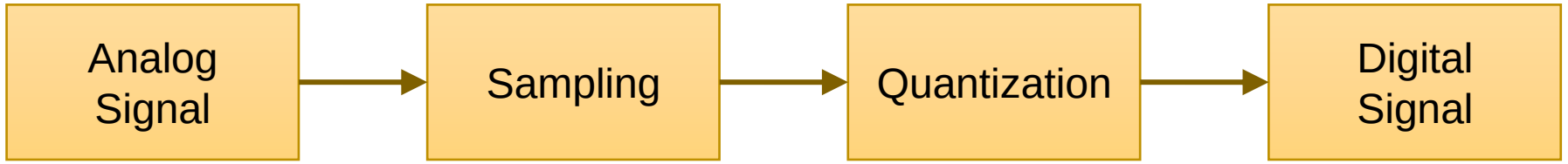






5.2 아날로그 신호 입력

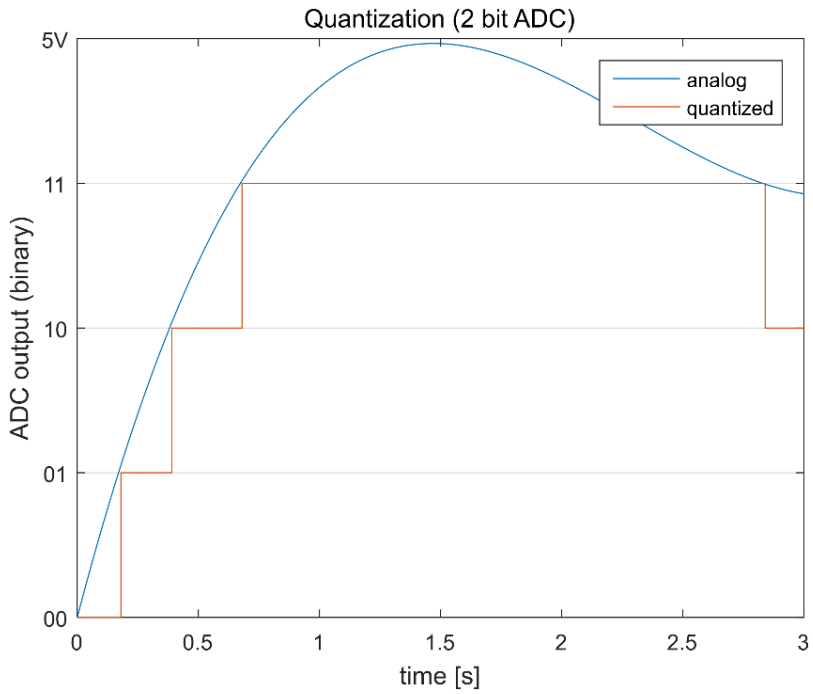
- 아날로그(자연계의 물리 정보, 연속적) 신호 → 디지털 신호

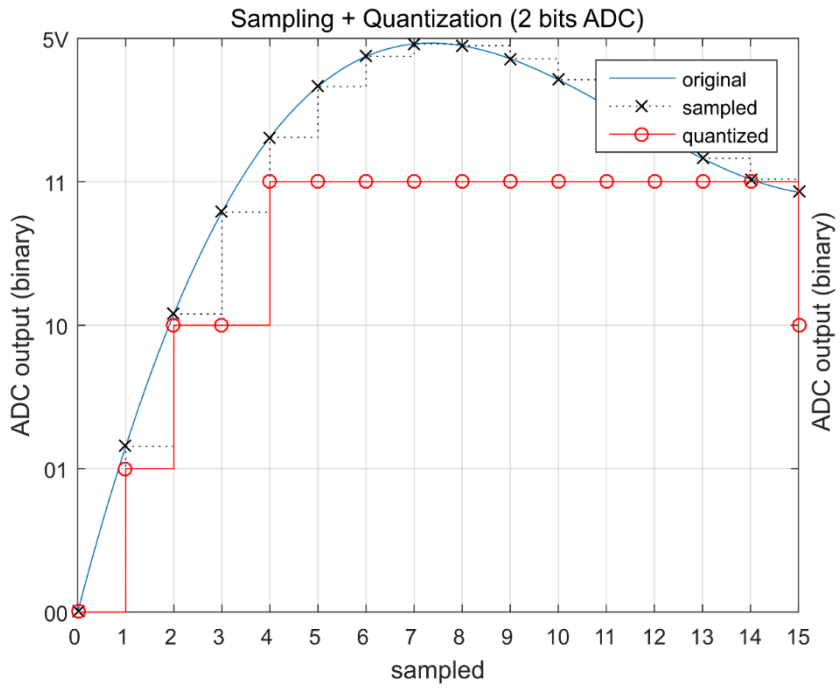


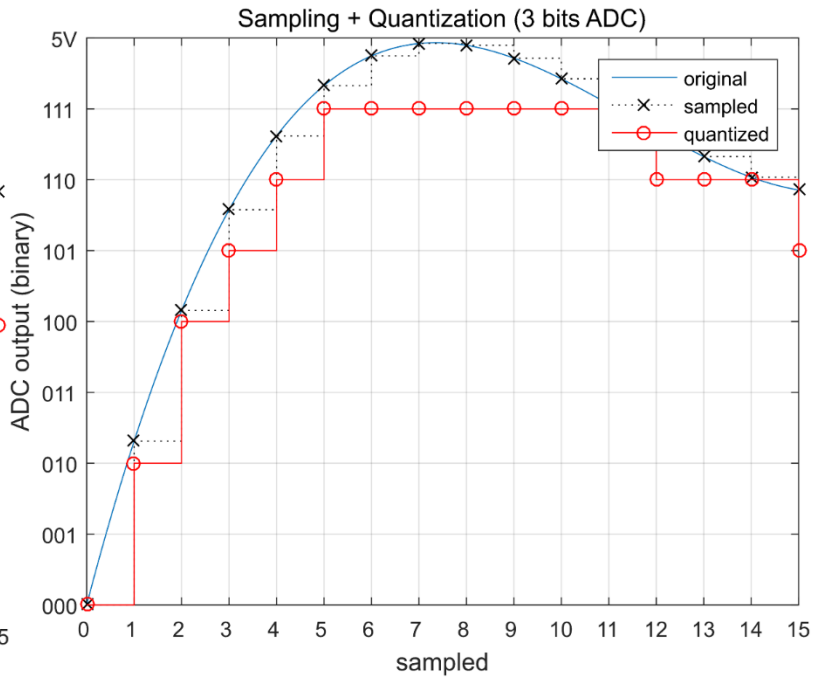
- 샘플링 (sampling)
  - 샘플링 주기에 따른 값만 사용 (예: 1초당 44100번)
- 양자화 (quantization)
  - 연속적인 물리량의 값을 일정한 레벨 로 나누어서 변환하는 과정
  - ADC(Analog to Digital Converter) 사용
  - 내부회로의 비교기의 수에 따라 해상도가 결정됨 (3bit → 3번 비교 → 2^3레벨)

아날로그 신호 입력

- 아날로그 → 디지털 (샘플링 된 값의 양자화)
  - 아두이노 양자화 : 10bit ADC 사용 ⇒ $2^{10} = 1024$ 레벨







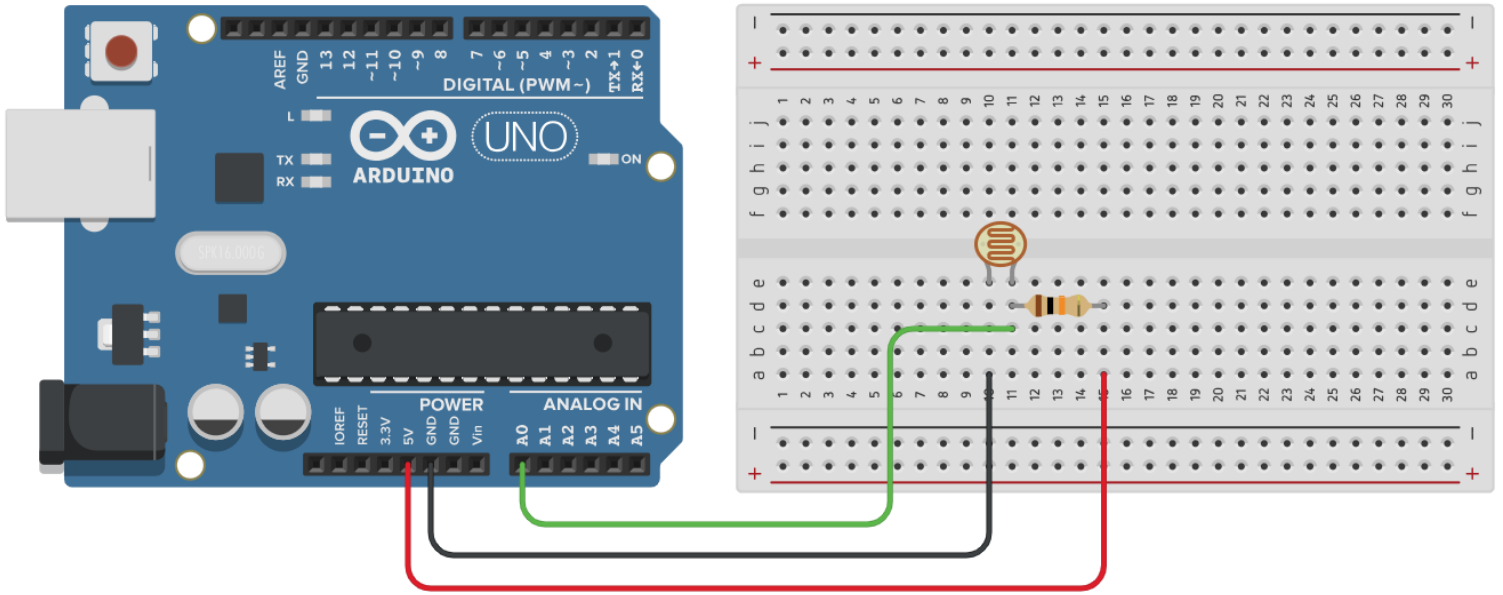
아두이노의 아날로그 입력

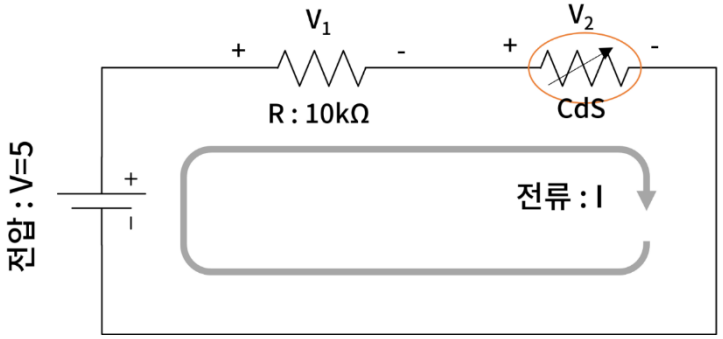
- 아날로그 센서 읽기 (예: A0)
  - 초기화 : `setup()`함수에서 `pinMode(A0, INPUT);`
  - 아날로그 값 읽기 : `int value = analogRead(A0);`
  - 입력 : 0 ~ 5V 전압 측정
  - 값의 범위 : 0 ~ 1023 (아두이노에서는 10 bit ADC를 사용)
    0000000000_2 (0_{10}) ~ 1111111111_2 (1023_{10})
  - 입력전압의 계산 : 입력전압 $= \frac{\text{아날로그입력값}}{1024} \times 5\ [\text{V}]$

5.3 조도 센서

Review

- CdS (황화카드뮴) : 포토레지스터, 조도센서 → 주위의 밝기에 반응하여 소자의 저항값이 수십Ω~수백kΩ으로 변하는 소자





- KVL : $V_1 + V_2 = 5$, CdS의 저항 : $R_C \rightarrow V_1 : V_2 = R : R_C$
- $V_2 = \frac{R_C}{R} V_1 = \frac{R_C}{R}(5 - V_2) \Rightarrow V_2 = \frac{R_C}{R+R_C} \times 5$ [V]
- A0의 입력값 : $A0 = \frac{R_C}{R+R_C} \times 1024$

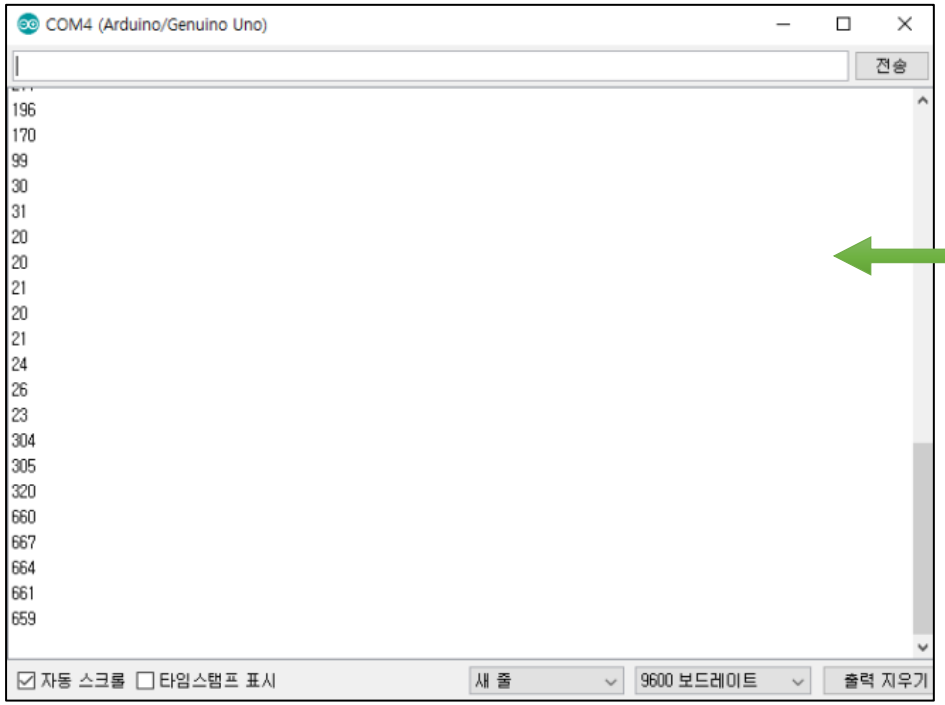
- 스케치

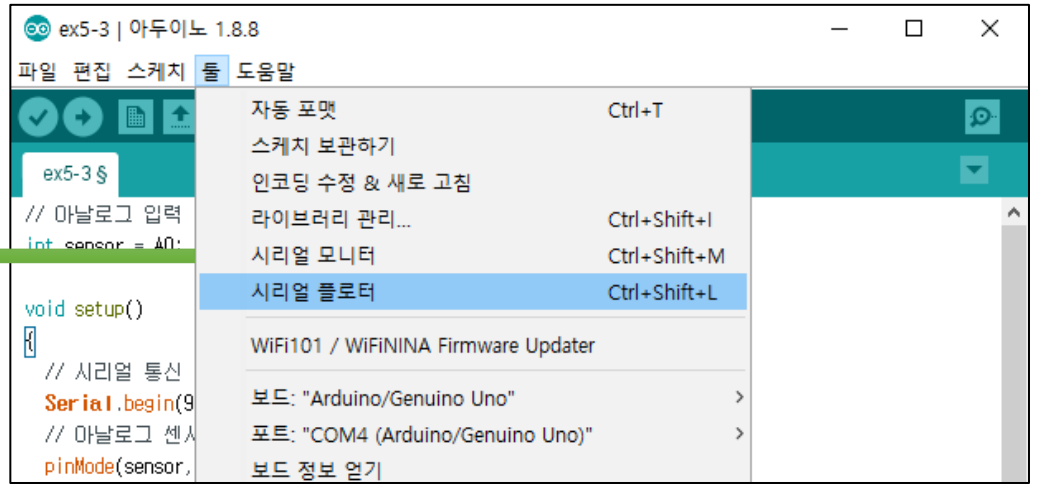
예제 5-3. 포텐쇼미터를 이용한 아날로그 입력 (예제 5-2와 동일)

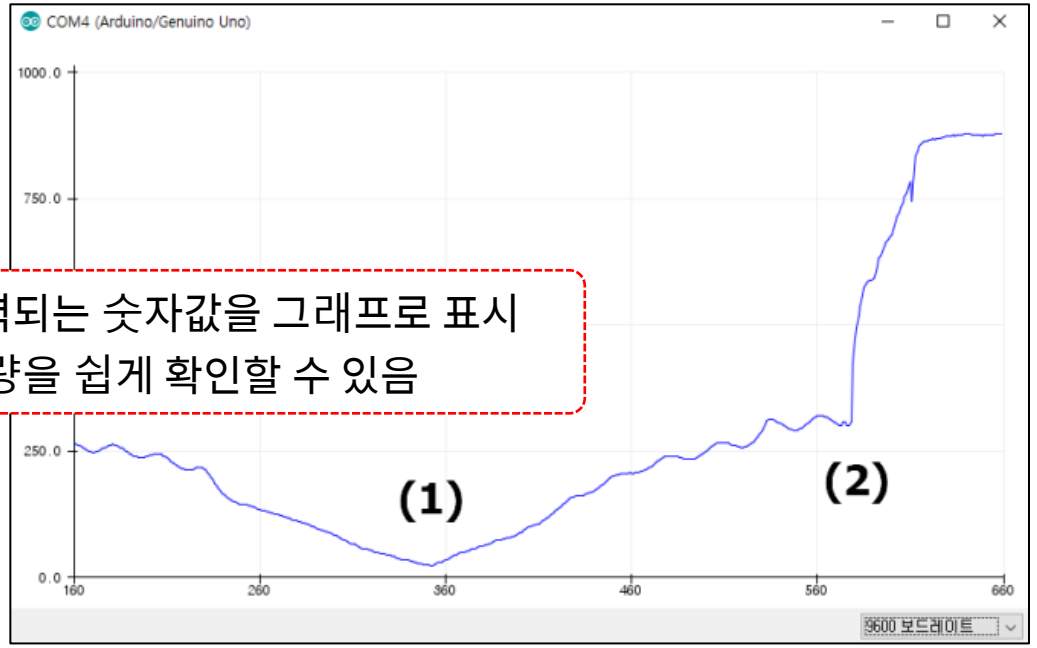
```
// 아날로그 입력 핀(A0~A5) 중에서 A0를 입력으로 사용
int sensor = A0;

void setup()
{
  // 시리얼 통신 초기화
  Serial.begin(9600);
  // 아날로그 센서 포트를 입력으로 사용
  pinMode(sensor, INPUT);
}

void loop()
{
  // 아날로그 센서를 통해서 값을 읽어들임
  int value = analogRead(sensor);
  // 시리얼 통신을 이용하여 PC로 전송
  Serial.println(value);
  // 100ms 대기
  delay(100);
}
```

시리얼 모니터 & 시리얼 플로터

Review







시리얼 플로터 : 시리얼 통신으로 입력되는 숫자값을 그래프로 표시

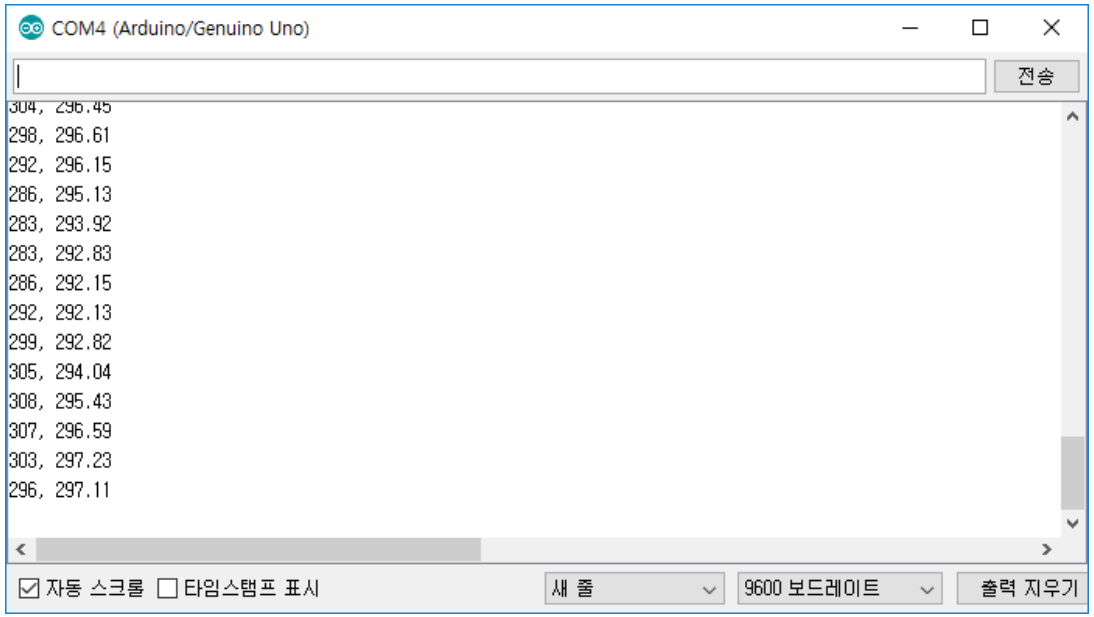
⇒ 시간에 따른 센서값의 변화량을 쉽게 확인할 수 있음

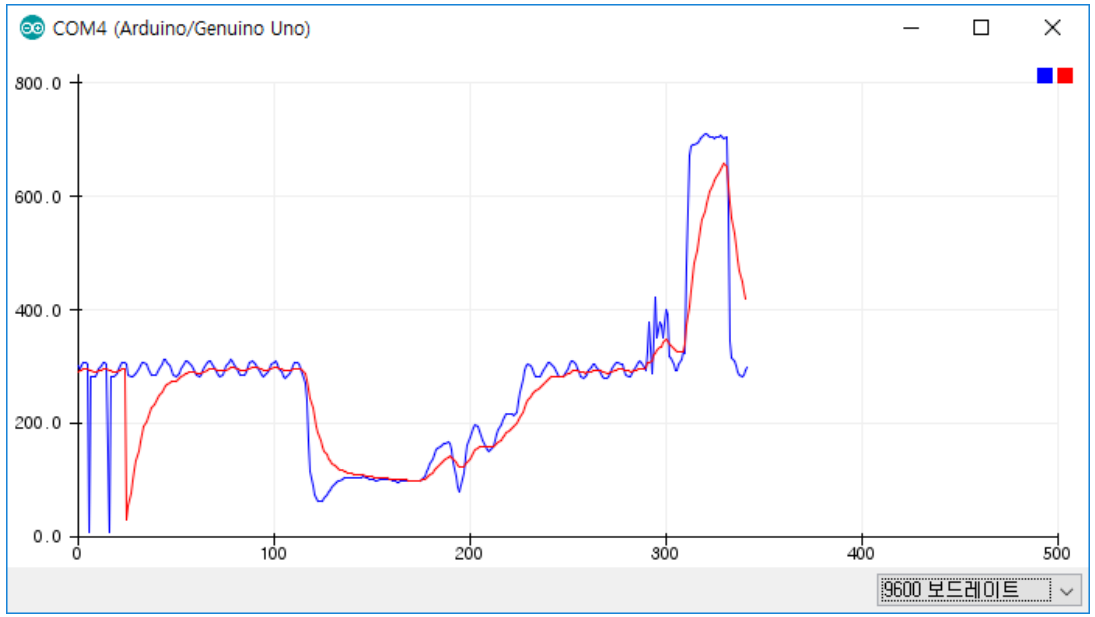
조도 센서의 응용

- 지수평균(exponential average) 구하기
  - 센서의 값이 외부 잡음 등의 영향으로 불안정할 때의 보상
  - 과거의 센서값과 현재의 센서값의 평균값 계산
  - 가중평균 : 각 값들의 가중치를 주어 평균 계산
    - 예] `value_ave = value_ave * (1.0 - 0.5) + value * 0.5;`
    - 현재값 가중치 : `0.5`

예제 5-4. 조도 센서값의 지수평균 구하기

```
int sensor = A0;       // 아날로그 입력 핀(A0~A5) 중에서 A0를 입력으로 사용
double value_ave = 0; // 센서값의 지수평균을 저장하는 변수

void setup()
{
  Serial.begin(9600); // 시리얼 통신 초기화
  pinMode(sensor, INPUT); // 아날로그 센서 포트를 입력으로 사용
}

void loop()
{
  int value = analogRead(sensor); // 아날로그 센서를 통해서 값을 읽어들임
  value_ave = value_ave * 0.9 + value * 0.1;   // 지수평균 구하기, 계수 : 0.1
  // 시리얼 통신을 이용하여 PC로 전송
  Serial.print(value); // 현재 측정값 전송, 줄바꿈 안함
  Serial.print(", ");  // 쉼표 및 공백
  Serial.println(value_ave); // 평균값 전송, 줄바꿈 하기
  // 100ms 대기
  delay(100);
}
```



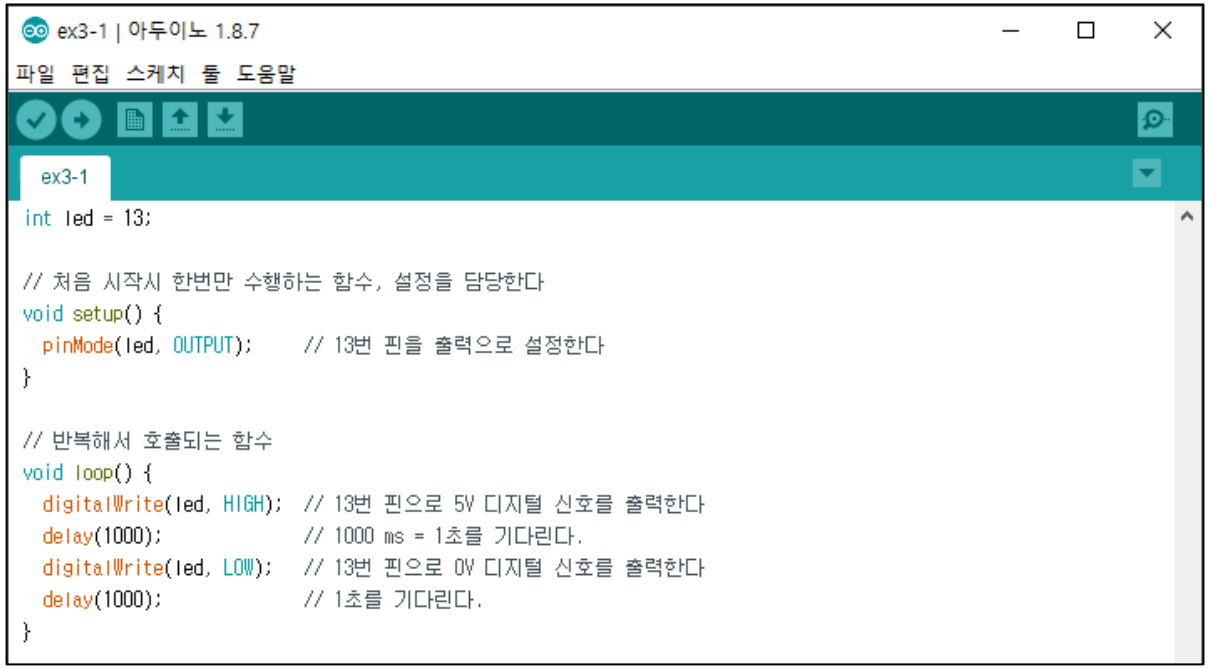


시리얼통신으로 한 줄에 2개의 값을 전송하면
시리얼 플로터에서는 두가지 색으로 나타남

- 파란색 선 : 센서로부터 입력받은 값
- 빨간색 선 : 지수평균값

4.4 부저의 사용

※ 교재에 포함되어 있지 않지만 디지털 출력에 해당하므로 4절에 추가합니다.



4.4 부저 사용법

- 부저 (buzzer)
  - 압전 물질을 사용하는 피에조(Piezo) 부저를 사용한 소리 출력
    - ➢ 압전 효과 (piezoelectric effect) : 결정체에 압력 → 전압 발생, 전압 인가 → 압력(늘어나거나 줄어듬) 발생
  - 압전 물질에 전압 인가 ⇒ 진동 (소리) 발생

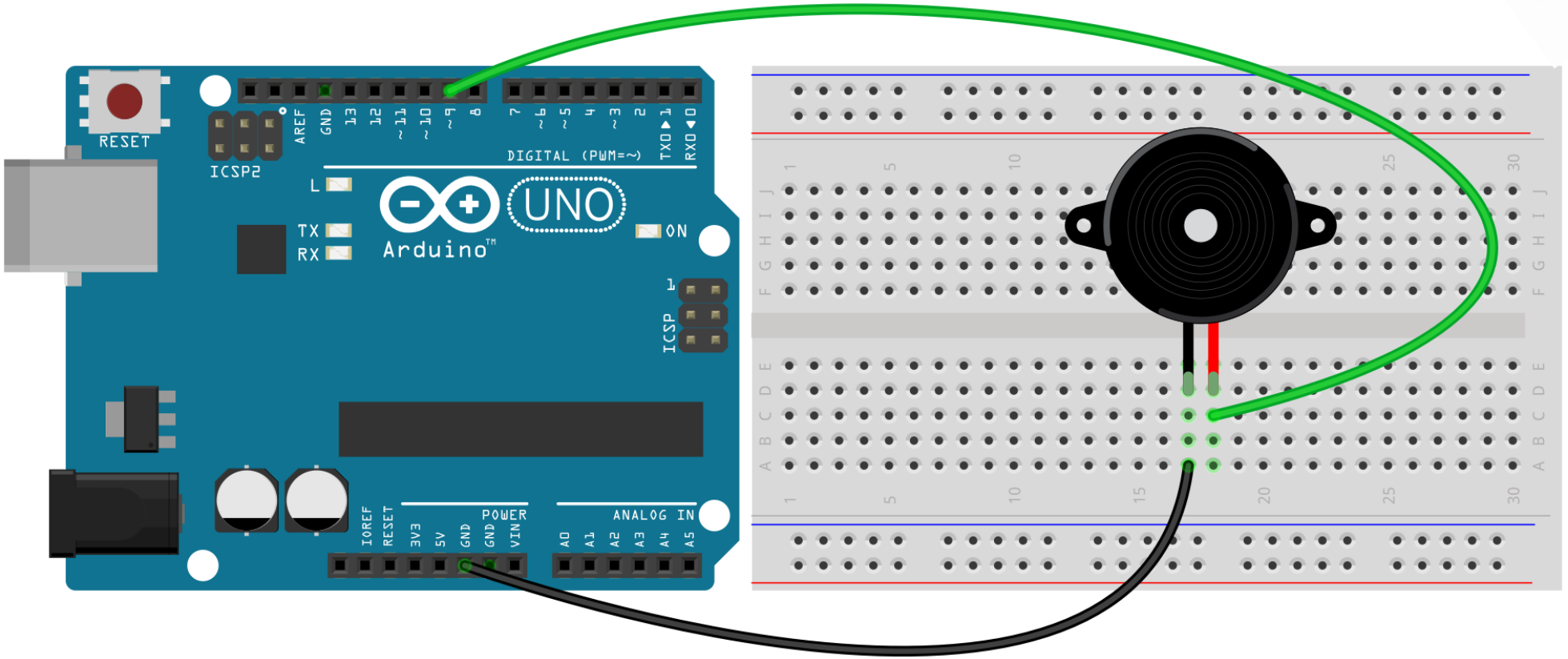
- 종류
  - 능동 부저 : 전류만 인가하면 정해진 소리를 발생시키는 소자

    - ➢ LED 제어와 동일한 방식
  - 수동 부저 : 압전 물질의 진동을 일으키는 전압을 직접 인가하여 소리 발생
    - ➢ 주기적인 펄스를 줌으로써 압전 물질이 인장/응축을 반복하여 공기의 진동을 일으킴 ⇒ 소리

[그림 출처 : www.mechasolution.com]

능동 부저(Active Buzzer)

- 능동 부저의 사용
  - LED 제어와 동일한 방법으로 제어
  - digitalWrite(핀번호, HIGH or LOW); ⇒ 부저를 켜고 끄기



능동 부저의 사용

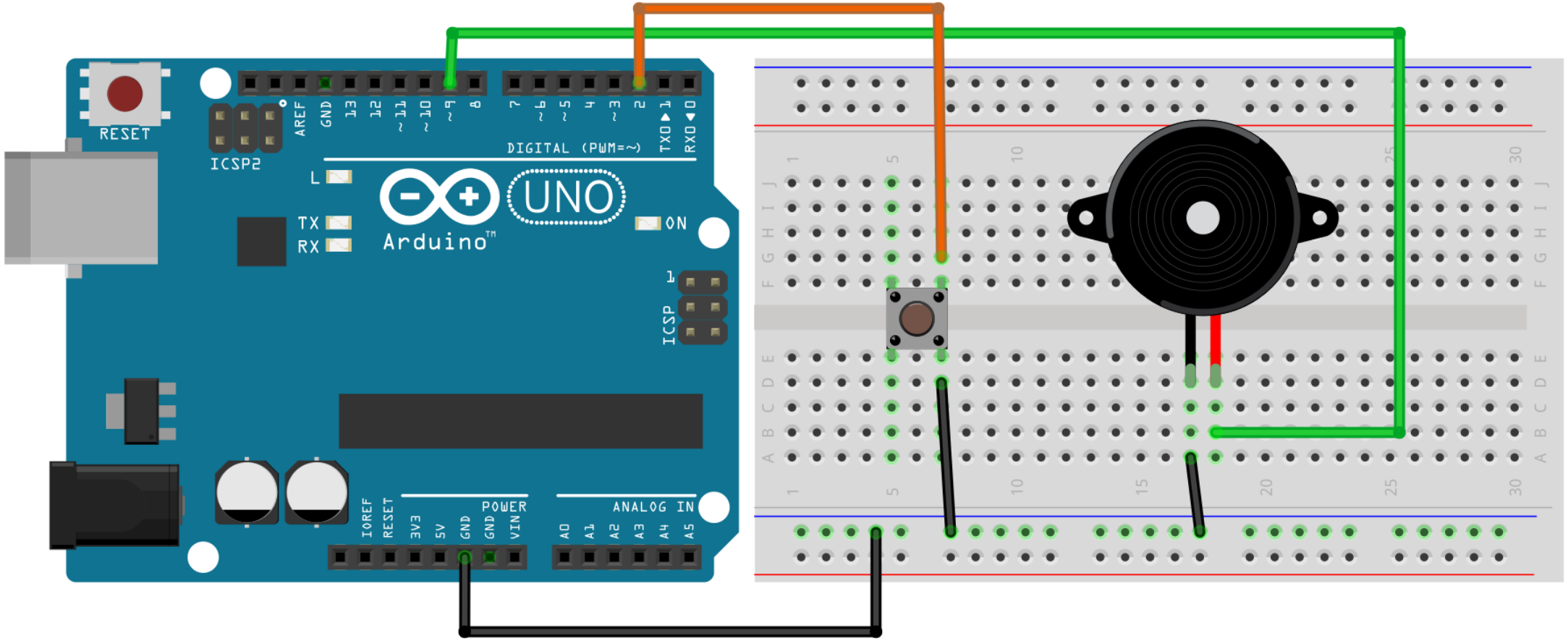
- LED 제어와 동일한 스케치

예제 4-4. 능동 부저의 사용

```
#define PIN 9

void setup() {
  pinMode(PIN, OUPUT); // 부저 연결 핀을 출력용으로
}

void loop() {
  digitalWrite(PIN, HIGH); // 1초동안 5V : ON
  delay(1000);
  digitalWrite(PIN, HIGH); // 1초동안 0V : OFF
  delay(1000);
}
```

예제 4-5. 부저 응용 예

- 스위치 입력 + 부저 출력
  - 아래의 회로를 이용하여 스위치가 눌려지면 부저 출력하기



능동 부저의 활용

- 스위치를 이용한 능동 부저의 제어

예제 4-5. 스위치를 이용한 능동 부저의 제어

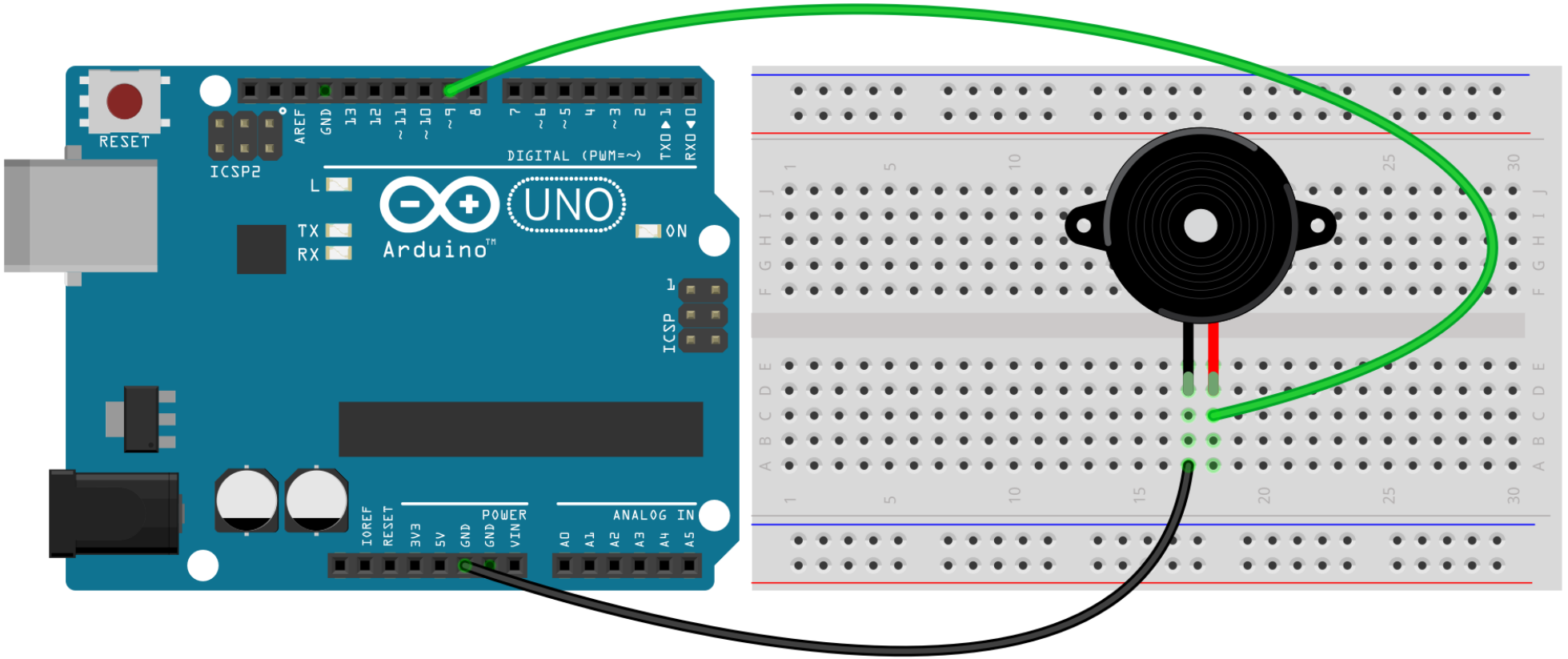
```
#define BUTTON 2  // 스위치 연결 핀
#define BUZZER 9  // 부저 연결 핀

void setup() {
  pinMode(BUTTON, INPUT_PULLUP); // 내부 풀업 저항 사용
  pinMode(BUZZER, OUTPUT); // 부저 연결핀을 출력용으로 설정
}

void loop() {
  int button = digitalRead(BUTTON); // 버튼 상태 읽기
  if(button == LOW) // 버튼이 눌려졌으면
    digitalWrite(BUZZER, HIGH);
  else
    digitalWrite(BUZZER, LOW);
  delay(10); // 스위치 안정화용 대기
}
```

수동 부저

- 수동 부저의 사용
  - 연결된 핀을 통해서 진동 전압을 직접 인가해야 함
    ⇒ 5V, 0V 를 계속 변환시키면서 인가
  - PWM 방식의 출력을 주파수를 변화시킬 수 있도록 제어
  - tone() 함수 사용
    - tone(핀번호, 주파수 [Hz], 시간 [ms]);
    - 예] `tone(PIN, 440, 1000);` ⇒ 1초 동안 라(A) 음을 출력
    - 시간을 생략하는 경우에는 계속 출력 ⇒ noTone() 함수로 종료
    - 소리를 출력하고 다음 문장이 계속 수행되므로 노래 연주시에는 delay() 등으로 시간 지연 후 다음 음계 출력
  - noTone(시간) : 소리를 멈추고 싶을 때 (쉼표)

수동 부저의 활용

- 능동부저와 동일한 방법으로 회로 작성
  - 부저만 수동 부저로 변경하며, 나머지는 동일



수동 부저의 활용

- 기본적인 사용 예

예제 4-6. 수동 부저의 사용

```
// 수동 부저를 사용 예제
#define PIN 9 // 부저 연결 핀

void setup() {
  // 부저 연결핀의 출력모드 설정
  pinMode(PIN, OUTPUT);
}

void loop() {
  // 600Hz로 1초동안 출력
  tone(PIN, 600, 1000);
  // 2초동안 기다림 => 1초 출력 후 1초 대기
  delay(2000);
}
```

수동 부저의 활용

예제. 수동 부저의 활용 : 사이렌 소리

```
// 수동 부저를 사용 예제
#define PIN 9 // 부저 연결 핀

void setup() {
  // 부저 연결핀의 출력모드 설정
  pinMode(PIN, OUTPUT);
}

void loop() {
  for(int i=200; i<1800; i+=10) {
    tone(PIN, i);
    delay(10);
  }
  for(int i=1800; i>=200; i-=10) {
    tone(PIN, i);
    delay(10);
  }
}
```

수동 부저와 능동 부저의 차이

- 수동 부저와 능동 부저의 차이
- 수동 부저와 능동 부저를 혼용해서 사용한다면?

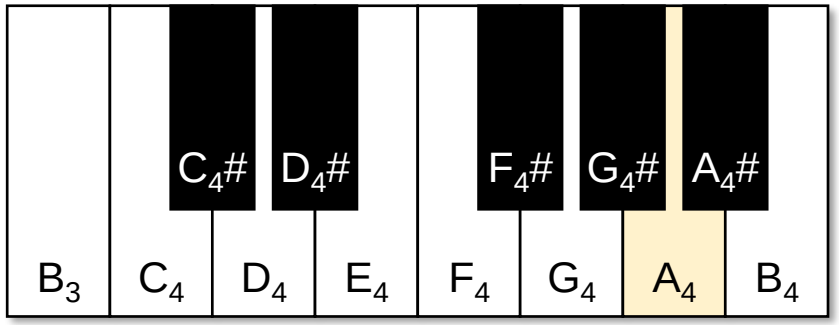
수동 부저 활용 #2

- 멜로디 연주
  - 음표 → 음계의 주파수 + 지연시간 연주
  - 음계의 주파수
    - ➢ 기준 음계를 이용하여 모든 음계의 주파수 계산
    - ➢ 기준 음계 : 4옥타브 '라'음
  - 지연시간
    - ➢ 음표와 악보의 템포에 따라 지정
    - ➢ 중간 Moderato : 템포(M) ♩ = 90~100
      → 4분 음표를 1분에 90~100회 진행
      → 온음 : 1분에 90 / 4 ~ 100 / 4회
    - ➢ x분음표 시간지연
      → $\frac{\text{총시간}}{\text{반복회수}} = \frac{60[s]\times 1000[ms]}{x\times M/4}$

명칭 (빠르기말)	독음 (발음)	템포 (M.M. ♩)
Prestissimo	프레티시모	200 이상
Presto	프레스토	180~200
Vivacissimo	비바티시모	170~180
Vivace	비바체	150~170
Allegrissimo	알레그리시모	140~150
Allegro	알레그로	120~140
Allegro moderato	알레그로 모데라토	100~120
Moderato	모데라토	90~100
Andantino	안단티노	70~100
Andante Moderato	안단테 모데라토	80~100
Andante	안단테	70~80
Adagietto	아다지에토	60~80
Adagio	아다지오	60~70
Larghetto	라르게토	50~60
Lento	렌토	40~50
Largo	라르고	40~50
Grave	그라베	40~50
Larghissimo	라르기시모	20~40
Largamente	라르가멘테	10~20

음계 주파수 계산

- 옥타브별 음계의 주파수 계산
  - 옥타브 : 주파수(진동수)가 2배 또는 ½ 배
  - 하나의 옥타브
    - ➢ 12개의 음계 (C, C#, D, D#, E, F, F#, G, G#, A, A#, B)
  - 기준 음계 주파수 : 4옥타브의 A(라) 음 = 440Hz
  - 연속음의 주파수 비율 : $\alpha = \sqrt[12]{2}$ (12번 곱하면 다음 옥타브의 비율 = 2)



	옥타브 3	옥타브 4	옥타브 5
G		440 / α / α	
G#	440 / 2 / α	440 / α	
A	440 / 2	440Hz	440 x 2
A#	440 / 2 x α	440 x α	
B		440 x α x α	

음계 주파수 계산

- 옥타브별 음계의 주파수 값

	1	2	3	4	5	6	7	8
C	32.7032	65.4064	130.8128	261.6256	523.2511	1046.5023	2093.0045	4186.0090
C#	34.6478	69.2957	138.5913	277.1826	554.3653	1108.7305	2217.4610	4434.9221
D	36.7081	73.4162	146.8324	293.6648	587.3295	1174.6591	2349.3181	4698.6363
D#	38.8909	77.7817	155.5635	311.1270	622.2540	1244.5079	2489.0159	4978.0317
E	41.2034	82.4069	164.8138	329.6276	659.2551	1318.5102	2637.0205	5274.0409
F	43.6535	87.3071	174.6141	349.2282	698.4565	1396.9129	2793.8259	5587.6517
F#	46.2493	92.4986	184.9972	369.9944	739.9888	1479.9777	2959.9554	5919.9108
G	48.9994	97.9989	195.9977	391.9954	783.9909	1567.9817	3135.9635	6271.9270
G#	51.9131	103.8262	207.6523	415.3047	830.6094	1661.2188	3322.4376	6644.8752
A	55.0000	110.0000	220.0000	440.0000	880.0000	1760.0000	3520.0000	7040.0000
A#	58.2705	116.5409	233.0819	466.1638	932.3275	1864.6550	3729.3101	7458.6202
B	61.7354	123.4708	246.9417	493.8833	987.7666	1975.5332	3951.0664	7902.1328

참고 : 음계 주파수 계산 프로그램 (Matlab)

```
% 음계 이름
note = [' C'; 'CS'; ' D'; 'DS'; ' E'; ' F'; 'FS'; ' G'; 'GS'; ' A'; 'AS'; ' B'];

A4 = 440; % 기준 음계 : 440Hz (A:'라', octave 4)
pitchidx = (-9-12*3):(2+12*4); % 음계의 종류 (옥타브당 12음)

rate = 2^(1/12); % 연속된 음의 주파수 변화율
freqs = A4 * rate.^pitchidx; % 각 음의 주파수 계산
frtable = reshape(freqs, 12, 8); % 벡터 -> 테이블로 변환

% 아두이노에서 사용할 헤더파일 생성

fid = fopen('pitches.h', 'w');
for k=1:8
    for n=1:12
        f = frtable(n, k);
        fprintf(fid, '#define %s%d %.0f\n', note(n,:), k, f);
    end
    fprintf(fid, '\n');
end
fclose(fid);
```

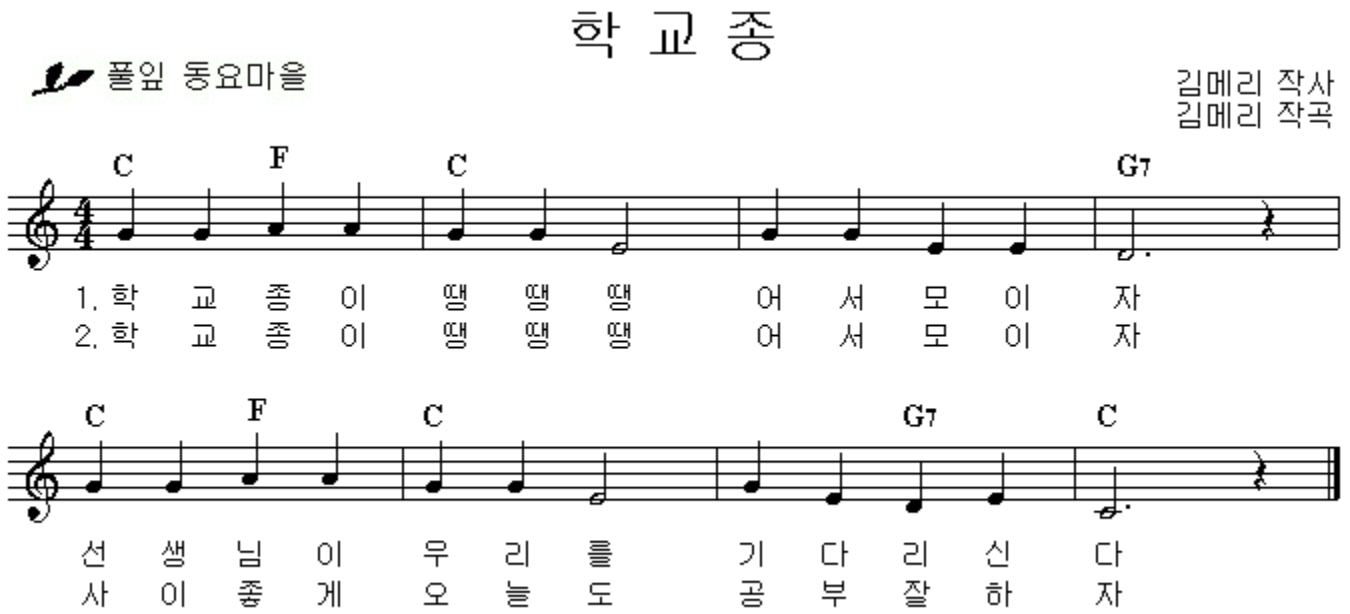
생성된 음계에 해당하는 주파수 데이터 (pitches.h)

```
#define  C1 33
#define CS1 35
#define  D1 37
#define DS1 39
#define  E1 41
#define  F1 44
...
#define  C4 262
#define CS4 277
#define  D4 294
#define DS4 311
#define  E4 330
#define  F4 349
#define FS4 370
#define  G4 392
#define GS4 415
#define  A4 440
#define AS4 466
#define  B4 494
...
#define AS8 7459
#define  B8 7902
```

	1	4	8
C	32.7032	261.6256	4186.0090
C#	34.6478	277.1826	4434.9221
D	36.7081	293.6648	4698.6363
D#	38.8909	311.1270	4978.0317
E	41.2034	329.6276	5274.0409
F	43.6535	349.2282	5587.6517
F#	46.2493	369.9944	5919.9108
G	48.9994	391.9954	6271.9270
G#	51.9131	415.3047	6644.8752
A	55.0000	440.0000	7040.0000
A#	58.2705	466.1638	7458.6202
B	61.7354	493.8833	7902.1328

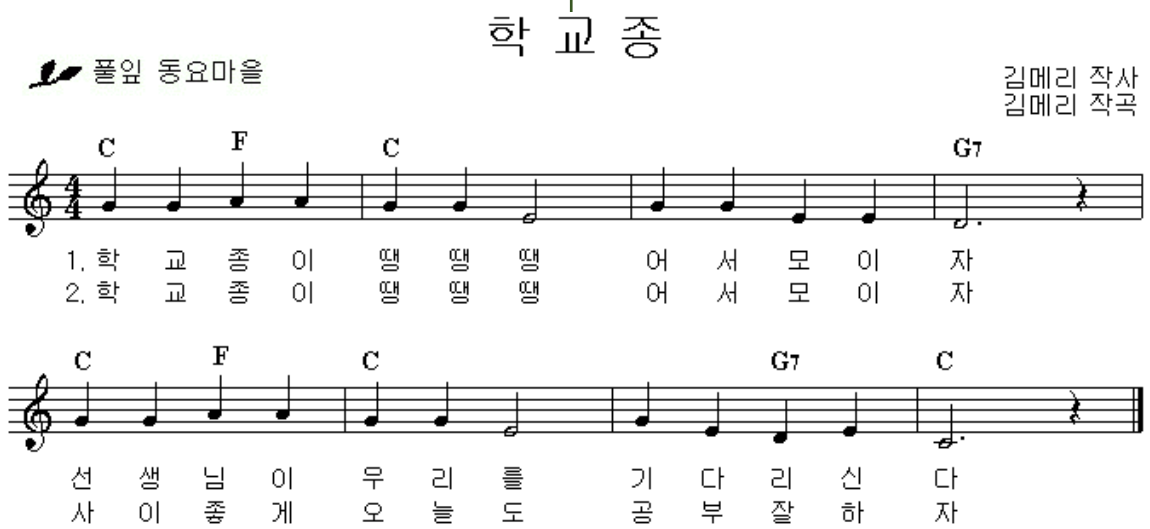
수동 부저를 이용한 멜로디 연주

- 악보 ⇒ 각 음표당 주파수 + 음계 지정 : 음표의 배열로 저장
  - 음표의 배열 : int notes[][2]; 하나의 음표당 주파수 및 음표 길이 지정
  - 학교 : 솔(G) + 4분음표 ⇒ { G4, 4 } (note, dur)
    - ➢ 주파수 : 392, 지연 : 60 * 1000 / (dur * 100.0 / 4) = 600[ms]
  - 땡 : 미(E) + 2분음표 ⇒ { E4, 2 }



예제 4-7. 멜로디 예제

```
#include "pitches.h" // 음계 주파수를 저장한 헤더파일
#define PIN 9 // 부저 연결 핀

int notes[][2] = {
  G4, 4, G4, 4, A4, 4, A4, 4, G4, 4, G4, 4, E4, 2,
  G4, 4, G4, 4, E4, 4, E4, 4, D4, 1,
  G4, 4, G4, 4, A4, 4, A4, 4, G4, 4, G4, 4, E4, 2,
  G4, 4, E4, 4, D4, 4, E4, 4, C4, 1,
};

void setup() {
  pinMode(PIN, OUTPUT);
  int count = sizeof(notes) / sizeof(int) / 2;
  for(int i=0; i<count; i++) {
    int note = notes[i][0]; // 음계
    int dur = notes[i][1];  // 길이
    int del = 4.0*60.0*1000.0/(dur*100.0); // moderato : *4/100
    tone(PIN, note, del);
    delay(del);
  }
} void loop() {} // loop() 함수는 없음 → 최초 1회만 수행하고 아무것도 안함
```



과제

- 앞에서 수행한 실습 과제를 동일하게 수행하고 스케치 파일, 결과 파일을 제출하라.

- 예제 4-7을 응용하여 8마디 이상의 음악을 연주하는 스케치를 완성하라.
  - 악보, 스케치 파일, 작동 동영상 (소리가 들려야 함, 화질은 저화질)을 제출하시오.