부저

부저

❖ 부저

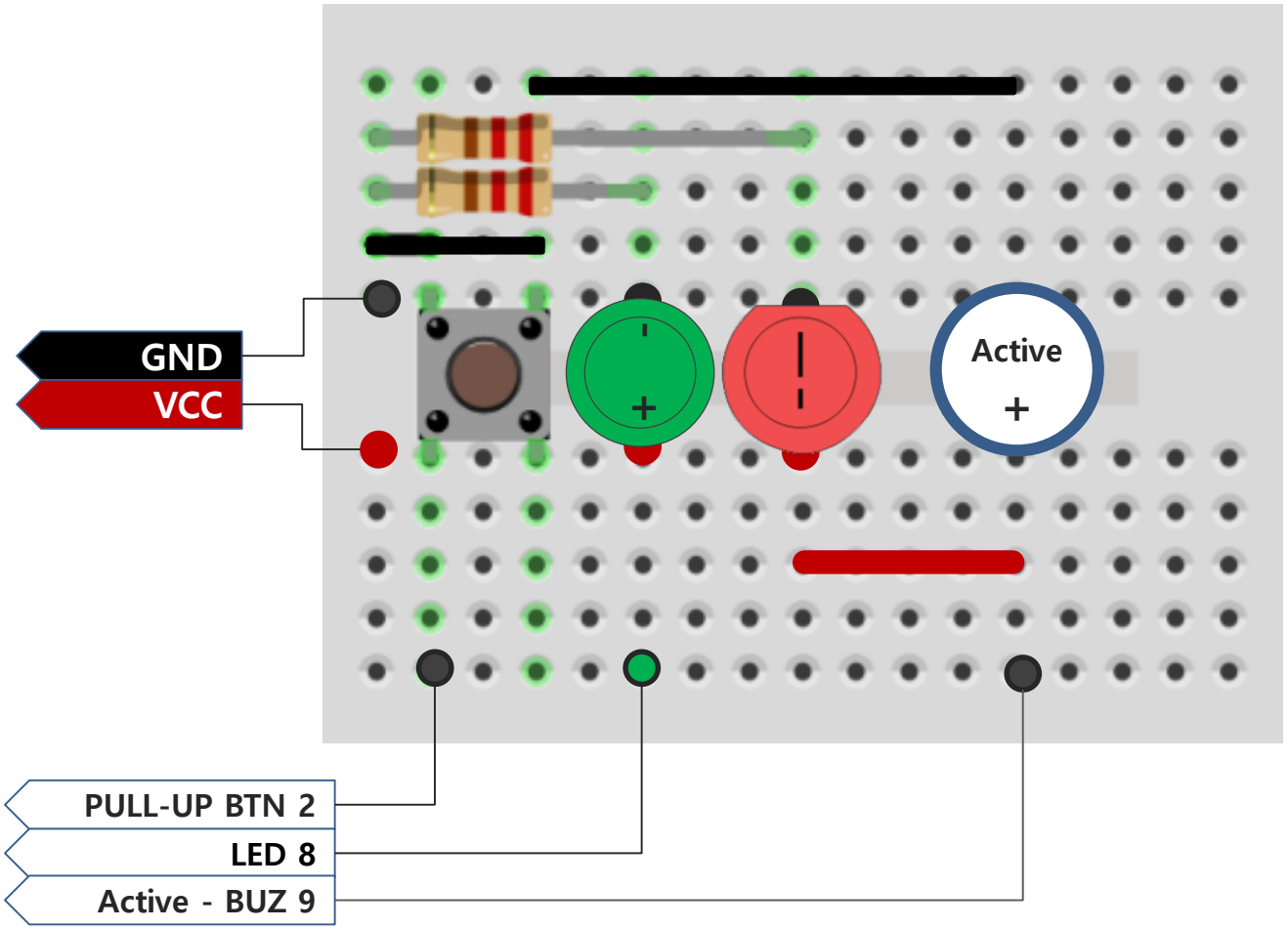
- ㅇ 능동부저(Active Buzzer)
  - ▪ 미리 설계된 회로를 가짐 → 음이 정해져 있음(단일 음)
    - 약 2KHz 대역의 소리를 출력
    - 경고 음으로 주로 활용
  - ▪ ON/OFF 전원 인가로 소리 제어
- ㅇ 수동 부저(Passive Buzzer)
  - ▪ 자체 회로를 가지지 않음
  - ▪ 출력 소리에 해당하는 주파수 신호를 인가하면 해당 소리 출력
  - ▪ 음계 표현 가능(멜로디 연주 가능)

능동부저
(Active Buzzer)

수동부저
(Passive Buzzer)

실습1: 능동 부저

❖ 회로도



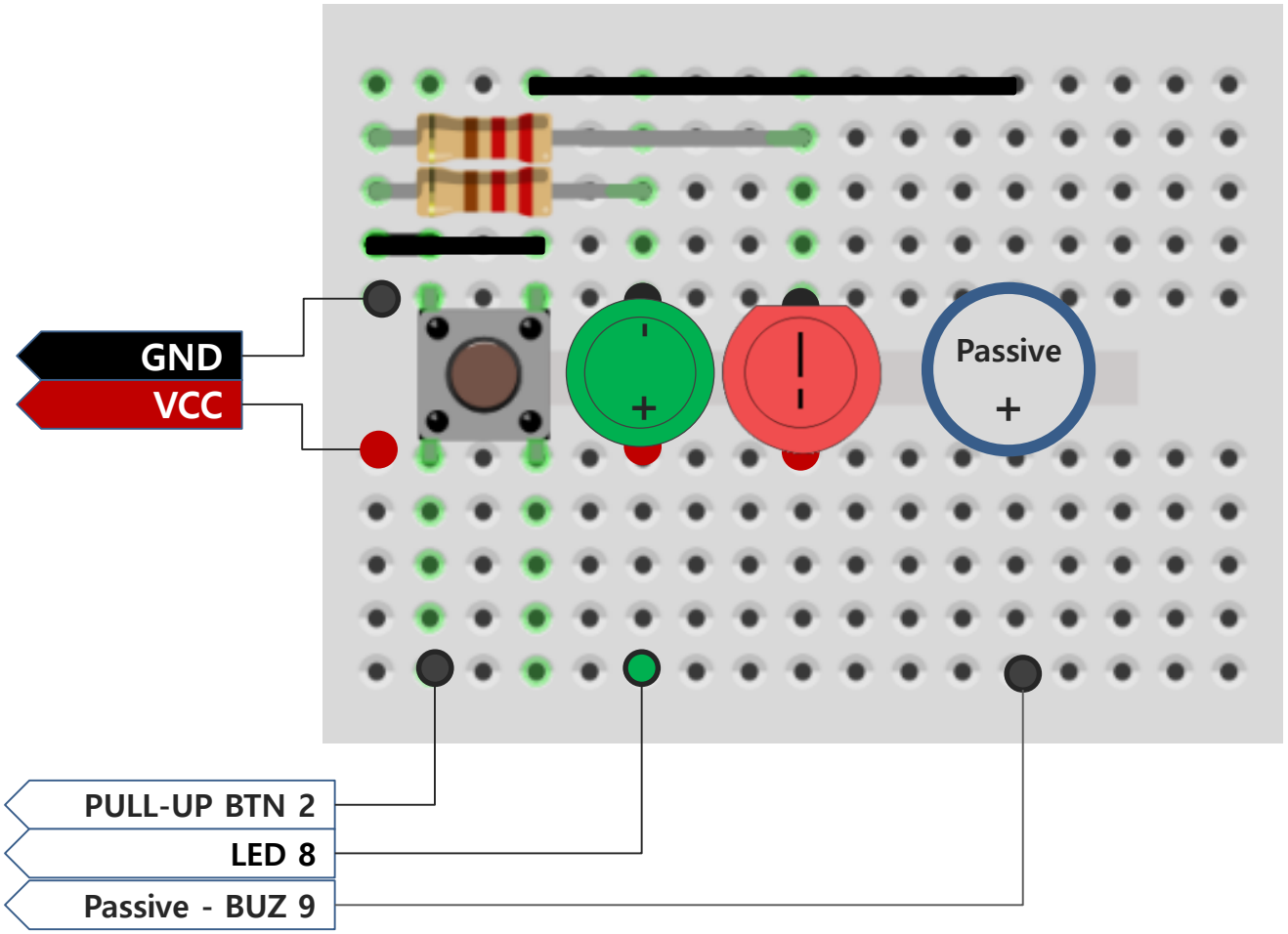
실습1: 능동 부저

❖ buzzer1.ino

```
// 부저 울리기(능동 부저)
const int buzzer_pin = 9;   // 부저 연결핀

void setup()
{
   pinMode(buzzer_pin, OUTPUT);    // 부저 연결핀 출력 설정
}

void loop()
{
   digitalWrite(buzzer_pin, HIGH);
   delay(2000);

   digitalWrite(buzzer_pin, LOW);
   delay(2000);
}
```

실습2: 수동 부저

❖ 회로도



실습2: 수동 부저로 멜로디 연주하기

❖ 파형 출력 함수

void tone(pinNum, frequency) void tone(pinNum, frequency, duration)		
기능	지정된 핀에 지정된 주파수의 구형파(듀티비 50%) 발생	
매개변수	pinNum	구형파가 출력되는 핀 번호
	frequency	출력되는 구형파 주파수(Hz) : 31~65635(unsigned int)
	duration	ms 단위의 파형 출력 지속 시간 : (unsigned long)
리턴 값	없음	

void noTone(pinNum)		
기능	지정된 핀에 출력되는 구형파 출력 정지	
매개변수	pinNum	구형파가 출력되는 핀 번호
리턴 값	없음	

실습2: 수동 부저로 멜로디 연주하기

❖ ex02/app.ino

```
#define NOTE_C4   262  // 4옥타브 도
#define NOTE_D4   294  // 4옥타브 레
#define NOTE_E4   330  // 4옥타브 미
#define NOTE_G4   392  // 4옥타브 솔
#define NOTE_A4   440  // 4옥타브 라
#define NOTE_C5   523  // 5옥타브 레

#define  NUM  49

int melody[NUM] = {     // 곰 세마리
   NOTE_C4, NOTE_C4, NOTE_C4, NOTE_C4, NOTE_C4,           // 도도도도도
   NOTE_E4, NOTE_G4, NOTE_G4, NOTE_E4, NOTE_C4,           // 미솔솔미도
   NOTE_G4, NOTE_G4, NOTE_E4, NOTE_G4, NOTE_G4, NOTE_E4,  // 솔솔미솔솔미
   NOTE_C4, NOTE_C4, NOTE_C4,                             // 도도도

   NOTE_G4, NOTE_G4, NOTE_E4, NOTE_C4,    // 솔솔미도
   NOTE_G4, NOTE_G4, NOTE_G4,             // 솔솔솔
   NOTE_G4, NOTE_G4, NOTE_E4, NOTE_C4,    // 솔솔미도
   NOTE_G4, NOTE_G4, NOTE_G4,             // 솔솔솔
```

실습2: 수동 부저로 멜로디 연주하기

❖ ex02/app.ino

```
   NOTE_G4, NOTE_G4, NOTE_E4, NOTE_C4,     // 솔솔미도
   NOTE_G4, NOTE_G4, NOTE_G4, NOTE_A4, NOTE_G4,  // 솔솔솔라솔
   NOTE_C5, NOTE_G4, NOTE_C5, NOTE_G4,     // 도솔도솔
   NOTE_E4, NOTE_D4, NOTE_C4};     // 미레도

// 음표 길이
int noteDuration[NUM] ={
  4, 8, 8, 4, 4,  4, 8, 8, 4, 4,  8, 8, 4, 8, 8, 4, 4, 4, 2,
  4, 4, 4, 4,  4, 4, 2, 4, 4, 4, 4,  4, 4, 2,
  4, 4, 4, 4,  8, 8, 8, 8, 2,  4, 4, 4, 4, 4, 4, 2};

const int speaker_pin = 9;

void setup()
{
  pinMode(speaker_pin, OUTPUT);
}
```

실습2: 수동 부저로 멜로디 연주하기

❖ ex02/app.ino

```
void loop()
{
  int  m, d, dd;

  for(m = 0;m < NUM;m++){
    d = 1000 / noteDuration[m];
    dd = d* 1.3;  // 음 출력 시간(4분 음표 325ms)

    tone(speaker_pin, melody[m], d);
    delay(dd);
  }
  delay(1000);
}
```

Melody 클래스

❖ 전체 음계 주파수 정의 파일

- C:\Program Files (x86)\Arduino\examples\02.Digital\toneKeyboard
  - pitches.h 파일을 복사

❖ 음악 파일

```
#include <pitches.h>

int notes[] = {
  // pitches.h에 정의된 음계로 정의
};
int durations[] = {
  // 연주 길이 정의   4: 4분 음표, 8: 8분 음표
};
```

Melody 클래스

❖ ex03/Melody.h

```
#pragma once
#include <Arduino.h>

class Melody {
protected:
    int pin;             // 핀번호
    int *notes;          // 음계 배열
    int *durations;      // 박자 배열
    int length;          // 음의 개수
    int cur_ix;          // 현재 연주하는 음의 인덱스
    long old_time;       // 이전 시간
    int note_duration;   // 현재 연주하는 음의 길이
    boolean b_play;      // 연주 여부
public:
    Melody(int pin, int *notes, int *durations, int length);
    void play();
    void stop();
    int toggle(bool bpause= false);
    void replay(); // 정지된 곳에서 다시 시작
    int getNote();  // 현재 재생 음
    void run();
};
```

Melody 클래스

❖ ex03/Melody.cpp

```
#include "Melody.h"

Melody::Melody(int pin, int *notes, int *durations, int length):
    pin(pin), notes(notes), durations(durations), length(length) {
    pinMode(pin, OUTPUT);
    cur_ix = -1;
    note_duration = 0;
    b_play = false;
    old_time = millis();
}

void Melody::play() {
    b_play = true;
    cur_ix = -1;
    note_duration = 0;
    old_time = millis();
}

void Melody::stop() {
    b_play = false;
}
```

Melody 클래스

❖ ex03/Melody.cpp

```
int Melody::toggle(bool bpause) {
    if(b_play) {    // 연주 상태이면
        stop();
    } else {    // 정지 상태이면
        if(bpause) {
            replay();
        } else  {
            play();
        }
    }
    return b_play;
}

void Melody::replay() {
    b_play = true;
}

int Melody::getNote() {
    if(!b_play) return 0;

    return notes[cur_ix];
}
```

Melody 클래스

❖ ex03/Melody.cpp

```
void Melody::run() {
    if(!b_play) return;

    long current = millis();
    long diff = current - old_time;

    if(diff >= note_duration) {
        cur_ix = (cur_ix+1)%length;
        note_duration = (1000 / durations[cur_ix]);
        tone(pin, notes[cur_ix], note_duration);
        note_duration = note_duration * 1.3;
        old_time = current;
    }
}
```

Melody 클래스

❖ ex03/Melody.cpp

```
#include <Melody.h>
#include "pirates.h"   // 연주할 파일

int length = sizeof(notes) / sizeof(int);
Melody melody(9, notes, durations, length);

void setup() {
    melody.play();
}

void loop() {
    melody.run();
}
```

MiniCom 연동

❖ ex04/app.ino

```
#include <MiniCom.h>
#include <Button.h>
#include <Melody.h>
#include "pirates.h"

MiniCom com;
Button btn(2);

int length = sizeof(notes) / sizeof(int);
Melody melody(9, notes, durations, length);

void check() {
    bool bplay = melody.toggle(true);
    if(bplay) {
        com.print(0, "play");
    } else {
        com.print(0, "pause");
    }
}
```

MiniCom 연동

❖ ex04/app.ino

```
void setup() {
    com.init();
    btn.setCallback(check);
    melody.play();
    com.print(0, "play");
}

void loop() {
    com.run();
    melody.run();
    btn.check();
}
```