2023. 소프트웨어대회
학생지도역량강화 직무연수
충북컴퓨터꿈나무축제 중등 SW제작 부문

강사 박정진

- 강의자료: https://arduino.datahub.pe.kr/2023
- 프로그램: https://arduino.datahub.pe.kr/sw
- 소스코드: https://arduino.datahub.pe.kr/2023/src

Directory Listing

https://arduino.datahub.pe.kr/2023/src

정보교과 수업전문성 강화 연수

Directory Listing 7 directories 2 files

/2023/src/

Up

1. Button/	11-Jul-2023 15:33	-
2. Mutitasking/	13-Jul-2023 14:57	-
3. Buzzer/	11-Jul-2023 15:26	-
4. FND/	13-Jul-2023 14:12	-
5. Solution/	14-Jul-2023 02:04	-
software/	14-Jul-2023 12:41	-
기출문제/	13-Jul-2023 11:33	-
2023 SW대회 학생지도역량강화 직무..>	12-Jul-2023 08:38	6M
아두이노 메이킹.pdf	13-Jul-2023 11:27	12M

아두이노 프로그래밍 강사 박정진(akapo@naver.com)

컴퓨터꿈나무축제 SW제작 부문

- 역대 기출문제 분석

년도	학교	구분	제목	재료 및 요구 기술
2022	고	지역	경품 추첨 도우미	버튼, LED, FND, 난수
		도	3·6·9 게임 연습기계 제작	버튼, LED, 4자리 FND, 부저, 가변저항, 사운드 센서, 시프트레지스터
	중	지역	7세그먼트 원리와 활용	버튼, LED, FND, 부저, 시리얼 입출력
		도	홈트레이닝	버튼, 부저, FND, 초음파센서
2021	고	지역	무인 입장 시스템 만들기	버튼, 3색 LED, 시리얼 출력
		도	공연 웨이팅 프로그램	버튼, FND, 시리얼 입출력
	중	지역	빛의 3원색	버튼, 3색 LED, 부저, 시리얼 출력
		도	최고의 기차여행	버튼, LED, 파싱, 알고리즘

컴퓨터꿈나무축제 SW제작 부문

- 역대 기출문제 분석

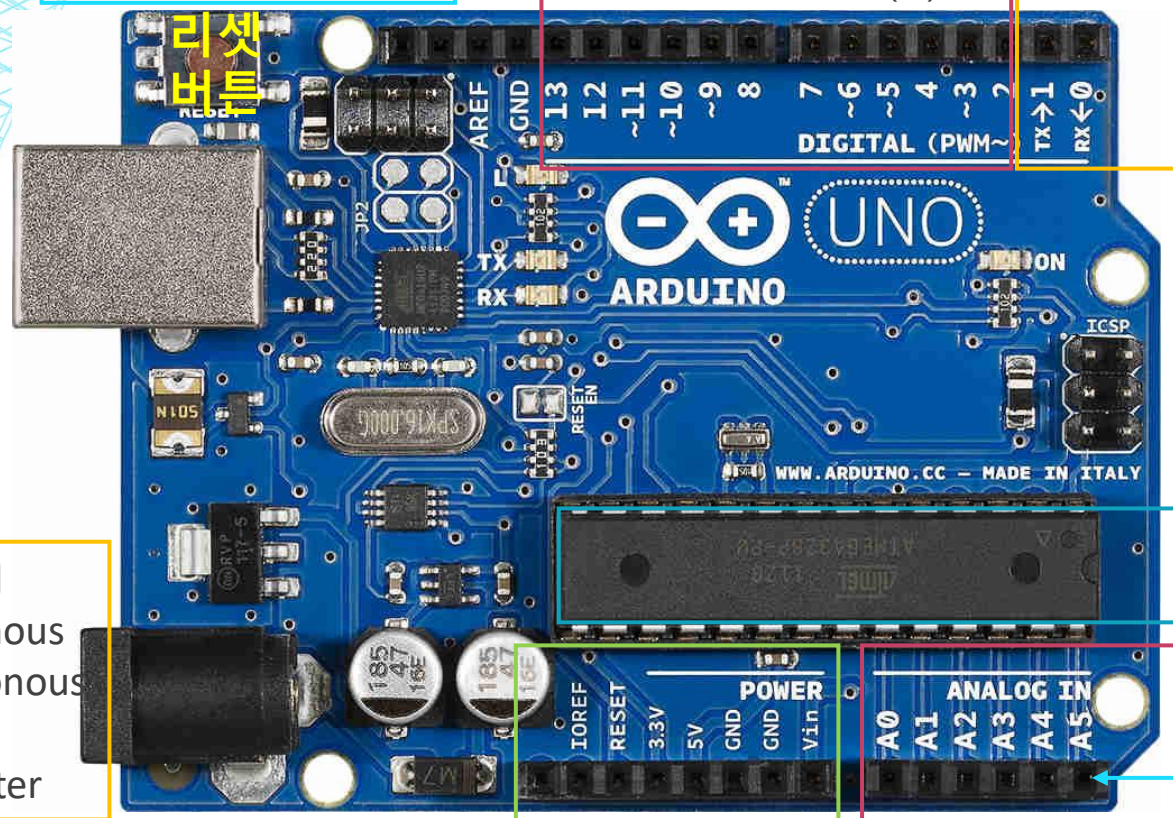
년도	학교	구분	제목	재료 및 요구 기술
2019	고	지역	달리기 왕	버튼, 적외선 센서, 시리얼 입출력
		도	엘리베이터 시뮬레이션	1602 LCD, 버튼, 알고리즘
	중	지역	순발력 게임	버튼 누른 순서 시간 감지
		도	추억의 뽑기 게임기	2자리 FND, 시리얼 입출력
2018	고	지역	서커스공연 입장 도우미	버튼, LED, 2자리 FND
		도	알람시계	버튼, 1602 LCD, (스탑&고)시계
	중	지역	기억력 게임 만들기	버튼, LED, 부저, 시간측정, 난수
		도	정렬과 탐색	시리얼 입출력, LED, FND, 부저, 알고리즘

아두이노 UNO (DIP) 핀 아웃

모든 GPIO 핀
디지털 입출력 가능

디지털 출력
아날로그 출력(~)

USART
통신



- 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.

Universal
Synchronous
Asynchronous
Receiver
Transmitter

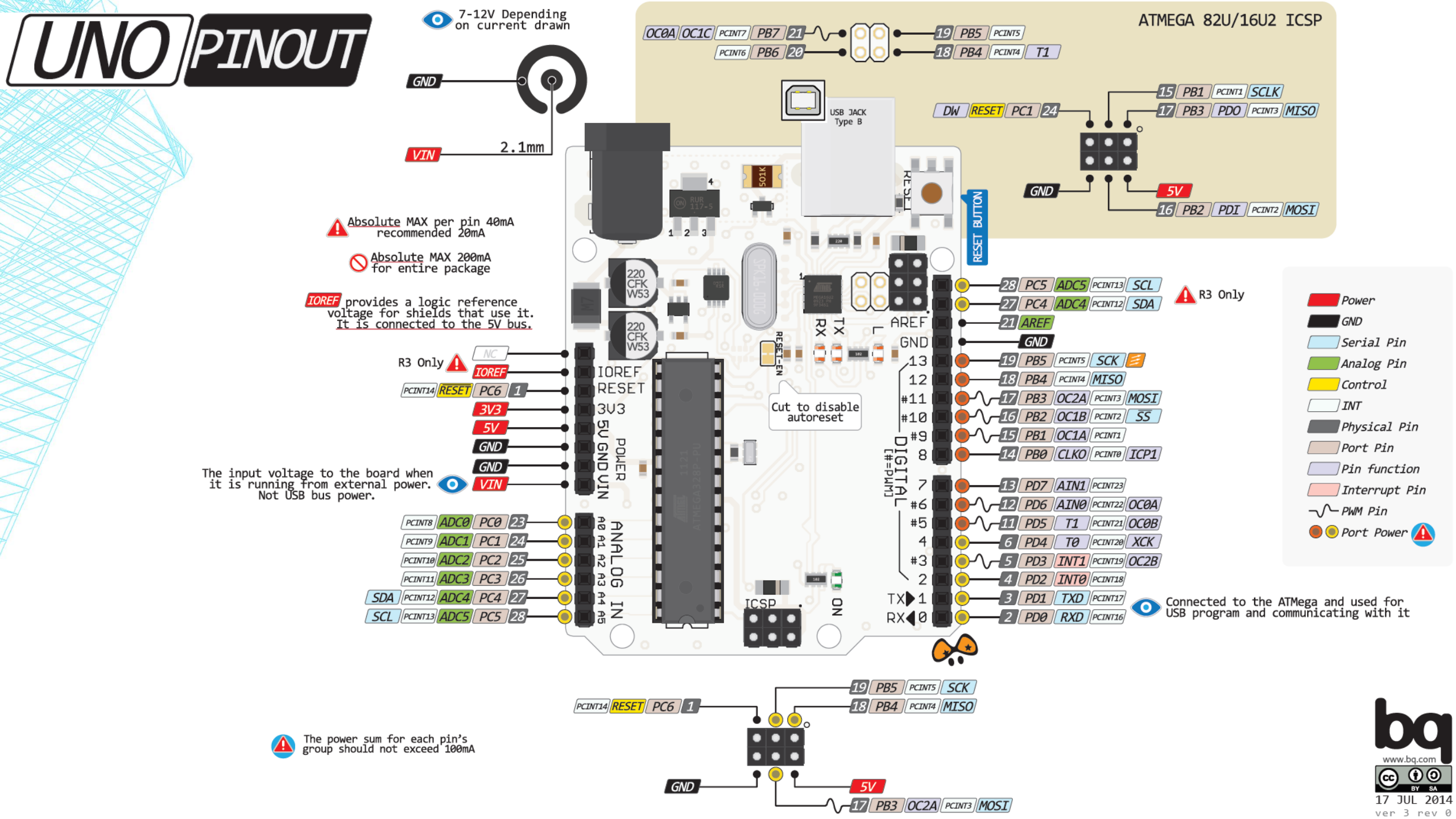
ATMEGA 328P
MCU 칩

전원

아날로그 입력

핀
or
포트

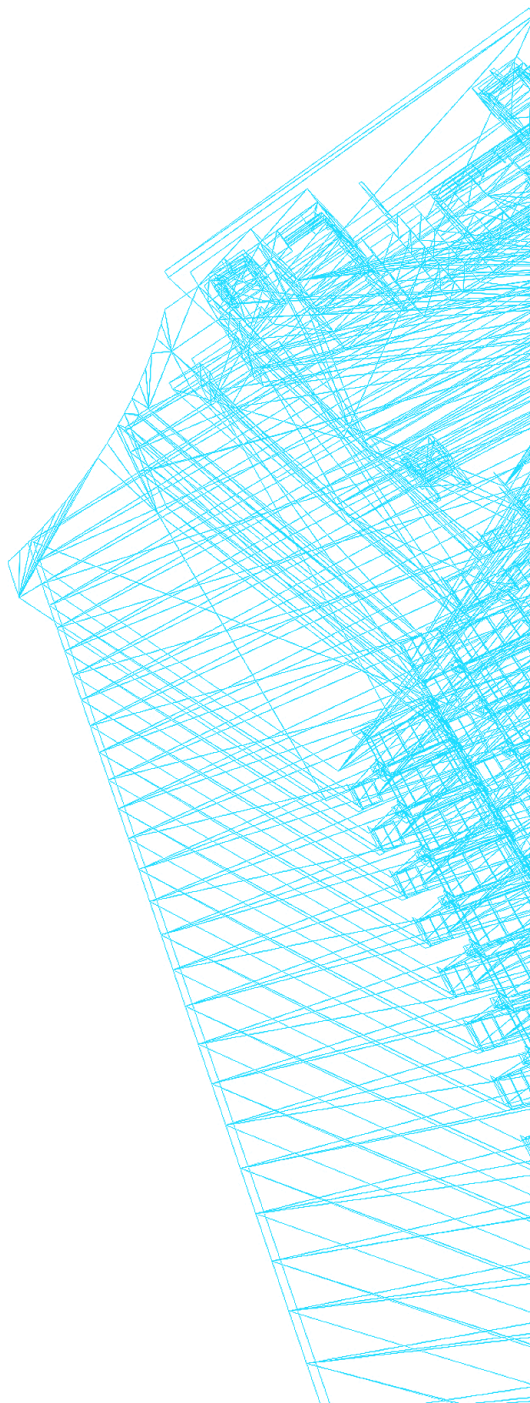
UNO PINOUT



17 JUL 2014
ver 3 rev 0

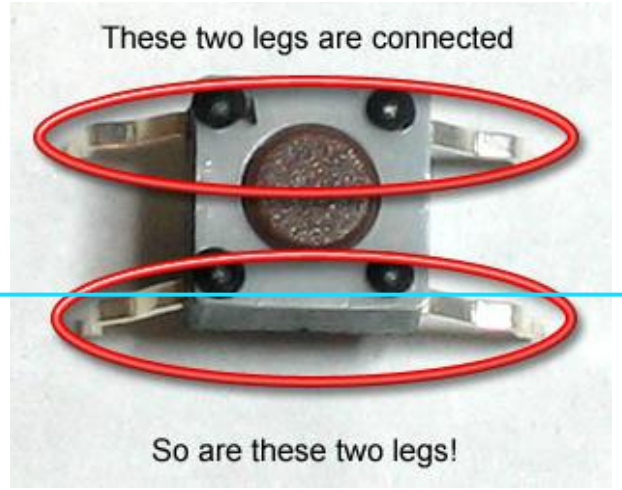
1. 버튼 다루기

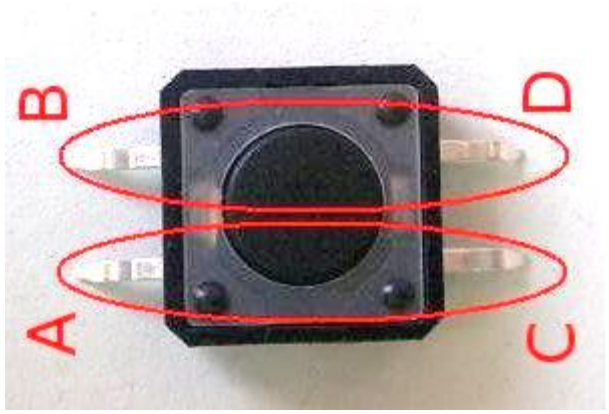
- 버튼 작동에 대한 오해
- 버튼 회로 설계 3가지 방법
- 채터링과 해결방법
- Button 클래스 설계

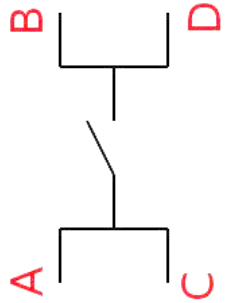


버튼

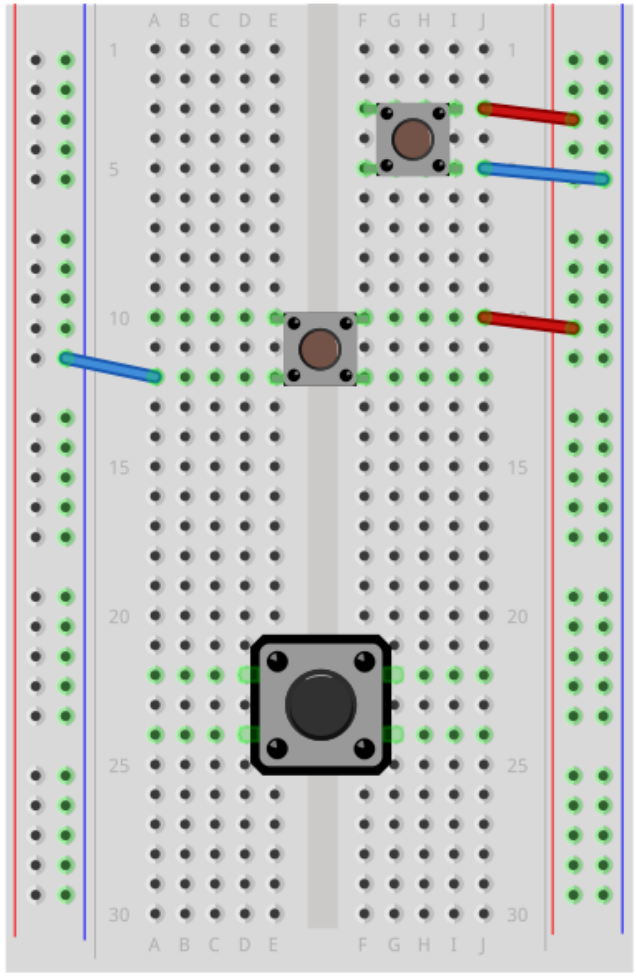
- 푸시 버튼







- 브레드보드에 배치 예시



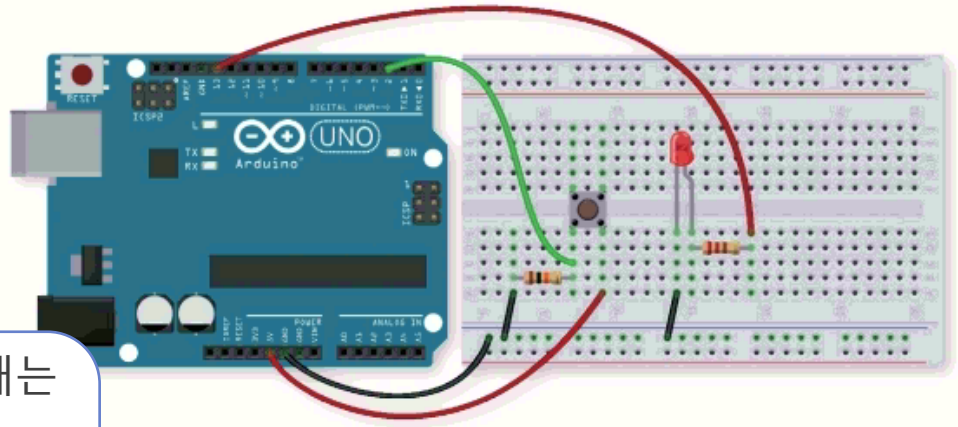
버튼 작동에 대한 오해

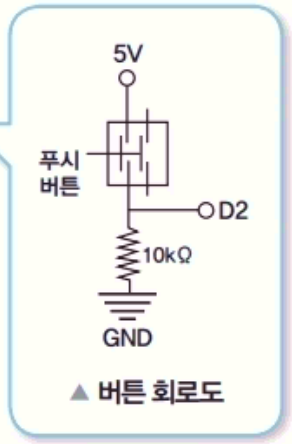
교과서 오류 보정

고등학교 정보(씨마스) p209

버튼은 어떤 원리로 동작할까?

버튼을 누른다는 것은 스위치가 연결된 것과 같으며, 스위치가 연결되면 전류가 흐른다. 버튼의 상태는 디지털 값으로 표현할 수 있고, [그림 Ⅳ-12]와 같이 풀다운 저항으로 연결했을 때 버튼을 누르면 디지털 값이 HIGH 상태, 누르지 않으면 디지털 값이 LOW 상태가 된다.





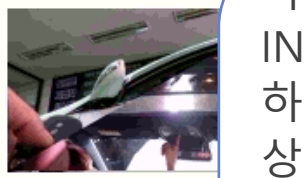
▲ 버튼 회로도

[그림 Ⅳ-12] 풀다운 저항으로 연결한 버튼 회로 구성

버튼을 누르면 스위치가 연결되고 전류가 흘러 D2의 값이 HIGH 상태가 되고, 버튼을 누르지 않으면 스위치는 끊어지고 D2의 상태는 LOW가 된다.

버튼을 활용한 예는 어디에서 볼 수 있을까?

승강기의 층 버튼, 사이드미러나 썬루프 열림 등에 활용된다.



푸시 버튼

푸시 버튼은 전류의 흐름
나 연결하는 스위치의 한

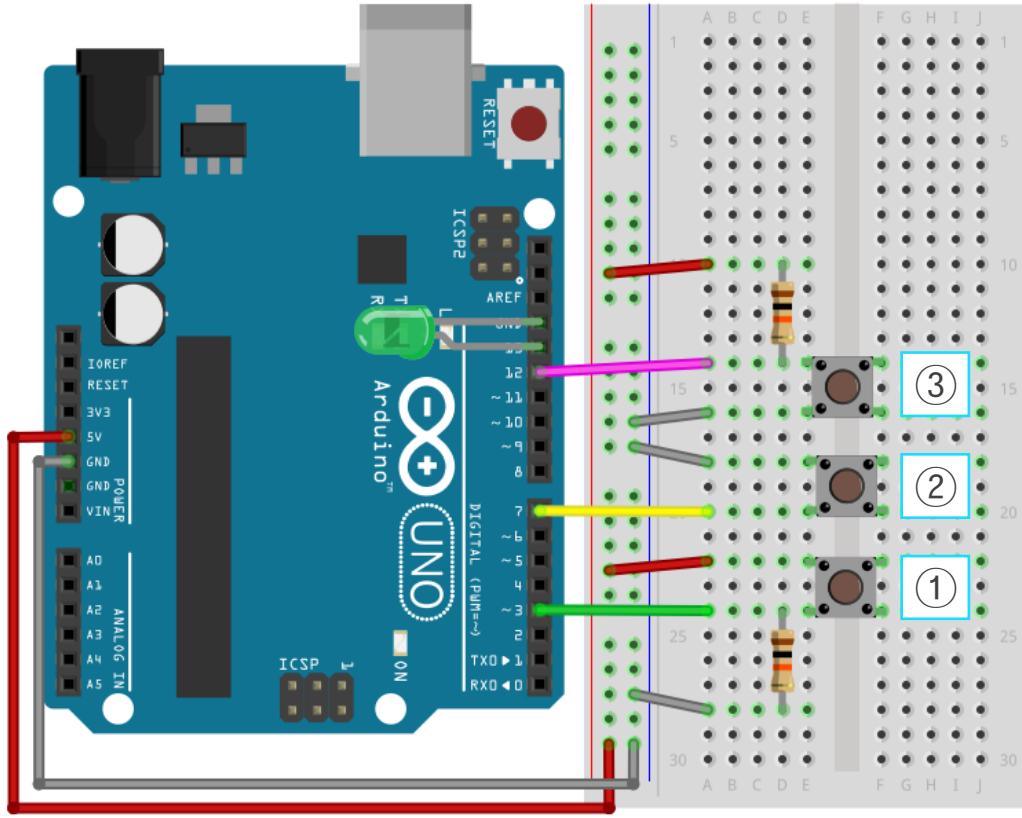
(HIGH, LOW 등) 상태는 전압으로 결정되고, 전류는 LED를 켜거나, 모터를 돌리는 등 작동에 관여한다.

아두이노의 핀은 INPUT 모드에서 하이임피던스 상태가 되어 전류가 거의 흐르지 못한다. 아두이노의 핀은 전류를 감지하는 것이 아니라 전압을 감지하는 것이다.

버튼 회로 설계

- 버튼 회로별 설계 방법 요약 (3종의 버튼 회로를 한번에 실습)

구분	①Active HIGH 버튼	②Active LOW 버튼	③Active LOW 버튼
눌렀을 때	HIGH	LOW	LOW
떼었을 때	LOW	HIGH	HIGH
버튼 연결	Vcc – IN	GND – IN	GND – IN
저항유무	필요함	불필요	필요함
저항위치	GND	-	Vcc
방법	풀다운 저항	MCU내부 저항	풀업 저항
pinMode	INPUT	INPUT_PULLUP	INPUT



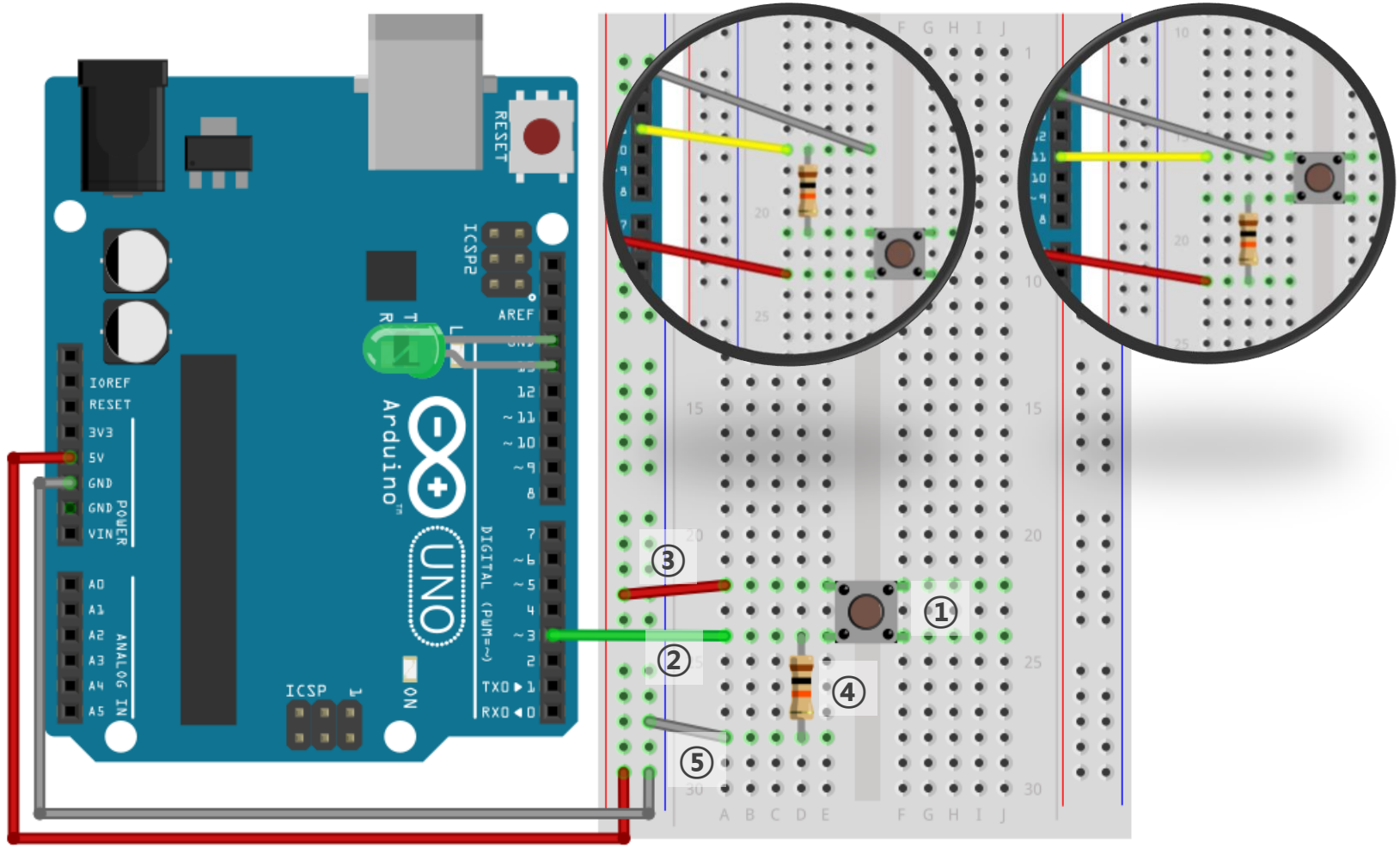
버튼 회로 설계 (①Active HIGH)

① Active HIGH 버튼 설계

구분	①Active HIGH 버튼
눌렀을 때	HIGH
떼었을 때	LOW
버튼 연결	Vcc – IN
저항유무	필요함
저항위치	GND
방법	풀다운 저항
pinMode	INPUT

• 잘못된 사례 1

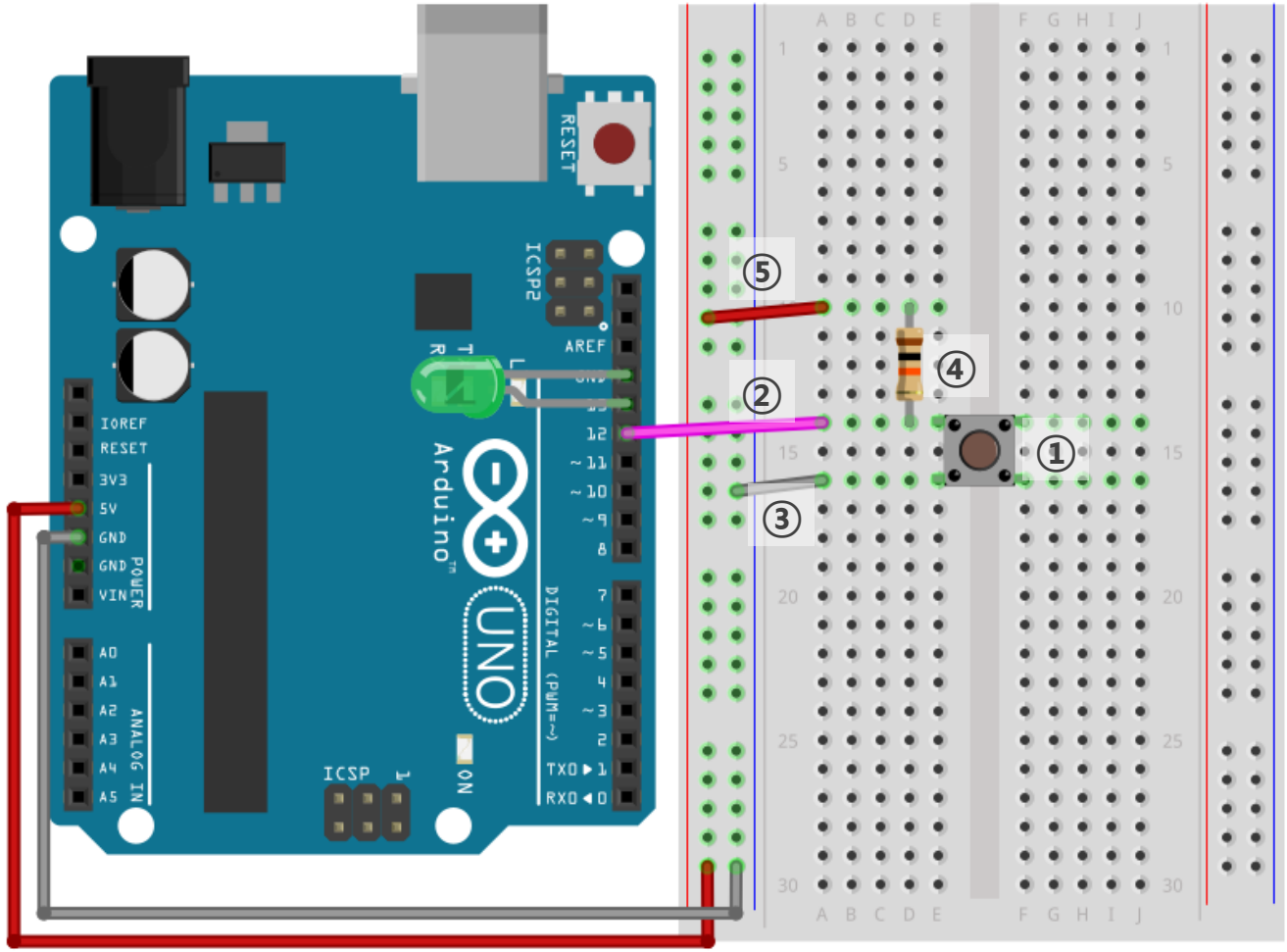
• 잘못된 사례 2



버튼 회로 설계 (③Active LOW)

② Active LOW 버튼 설계

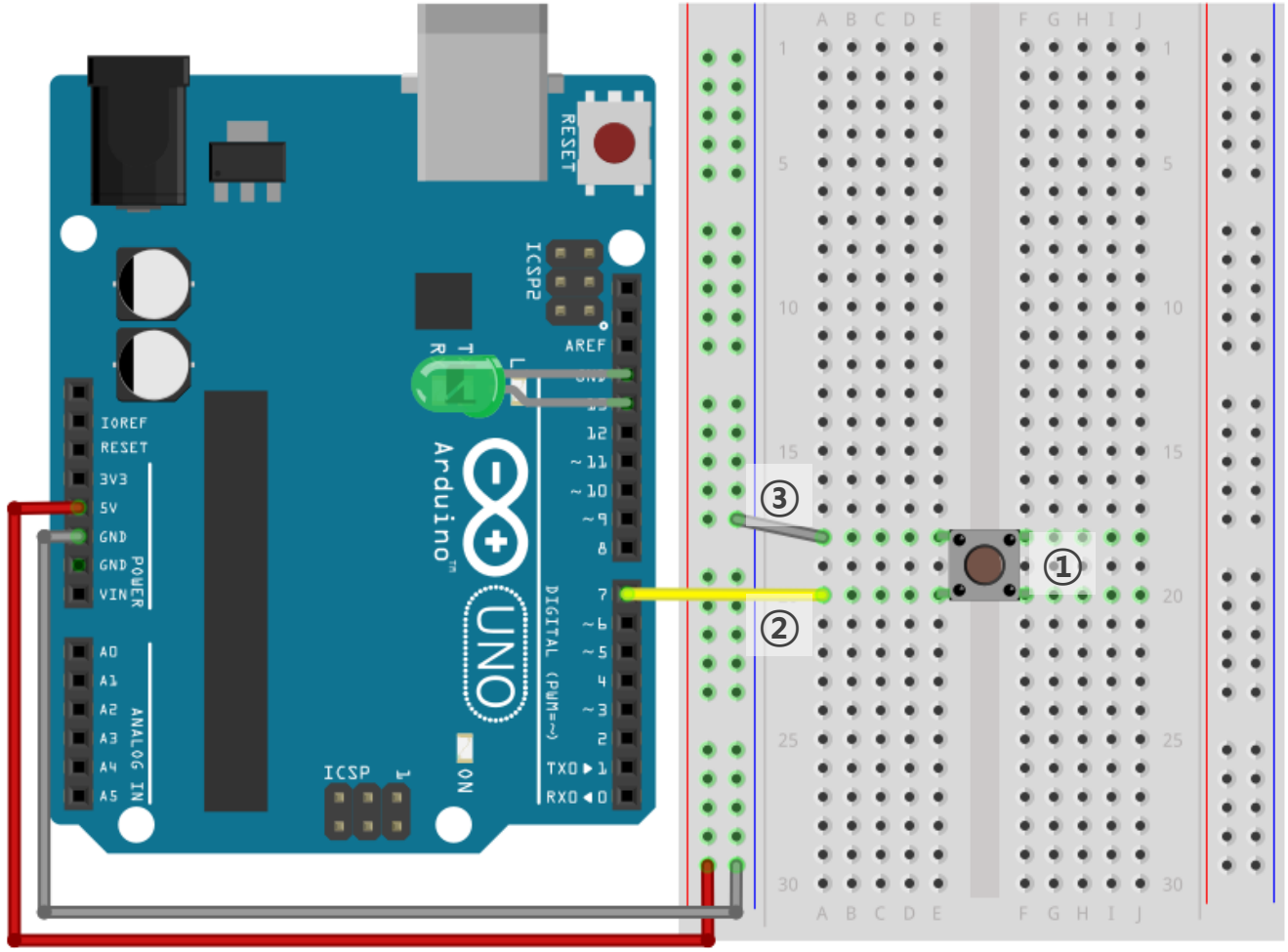
구분	②Active LOW 버튼
눌렀을 때	LOW
떼었을 때	HIGH
버튼 연결	GND – IN
저항유무	필요함
저항위치	Vcc
방법	풀업 저항
pinMode	INPUT



버튼 회로 설계 (②Active LOW)

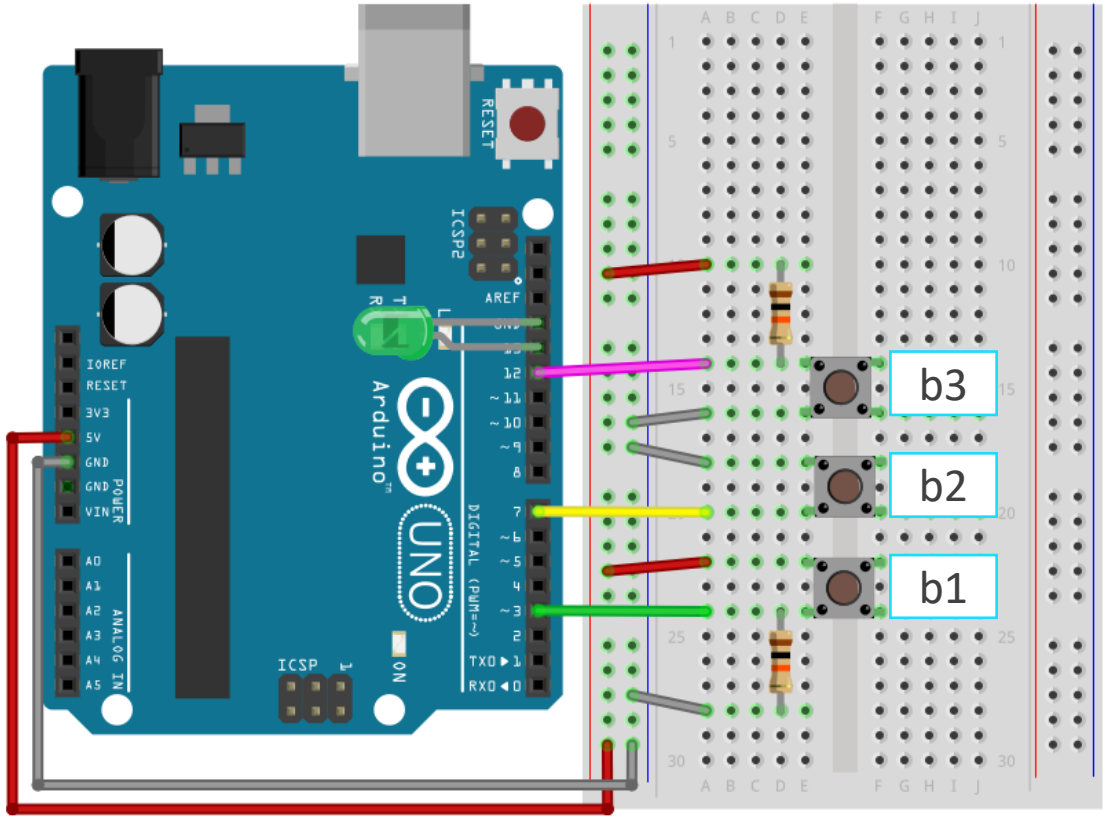
② Active LOW 버튼 설계

구분	②Active LOW 버튼
눌렀을 때	LOW
떼었을 때	HIGH
버튼 연결	GND – IN
저항유무	불필요
저항위치	-
방법	MCU내부 저항
pinMode	INPUT_PULLUP



버튼 회로 실습

- 회로



buttonAll.ino

```
#define BTN1  3 // ACTIVE HIGH
#define BTN2  7 // ACTIVE LOW(INPUT_PULLUP)
#define BTN3 12 // ACTIVE LOW

void setup() {
  Serial.begin(9600);
  pinMode(BTN1, INPUT);
  pinMode(BTN2, INPUT_PULLUP);
  pinMode(BTN3, INPUT);
}

void loop() {
  int b1 = digitalRead(BTN1);
  int b2 = digitalRead(BTN2);
  int b3 = digitalRead(BTN3);

  Serial.print(" b1: ");
  Serial.print(b1);
  Serial.print(" b2: ");
  Serial.print(b2);
  Serial.print(" b3: ");
  Serial.println(b3);
  delay(500);
}
```

토글 버튼의 구현

- 실습퀴즈

  아래와 같이 작동하는 버튼을 구현하시오.

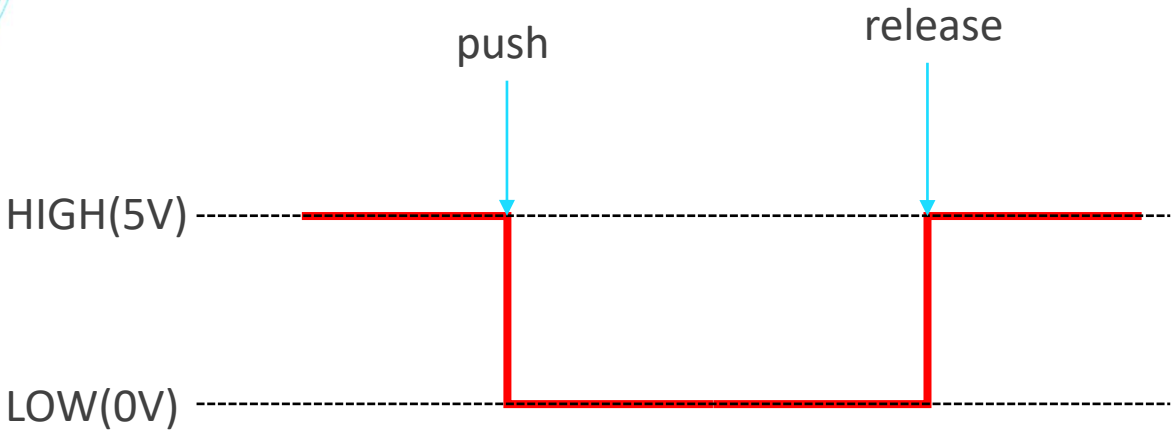
  - 버튼을 한 번씩 누를 때 마다
  - 13번 LED의 켜짐과 꺼짐이 반전

- 소스코드

```
#define BTN 12
#define LED 13

void setup() {



}

void loop() {




}
```

토글 버튼의 구현

- INPUT_PULLUP 버튼의 상태도



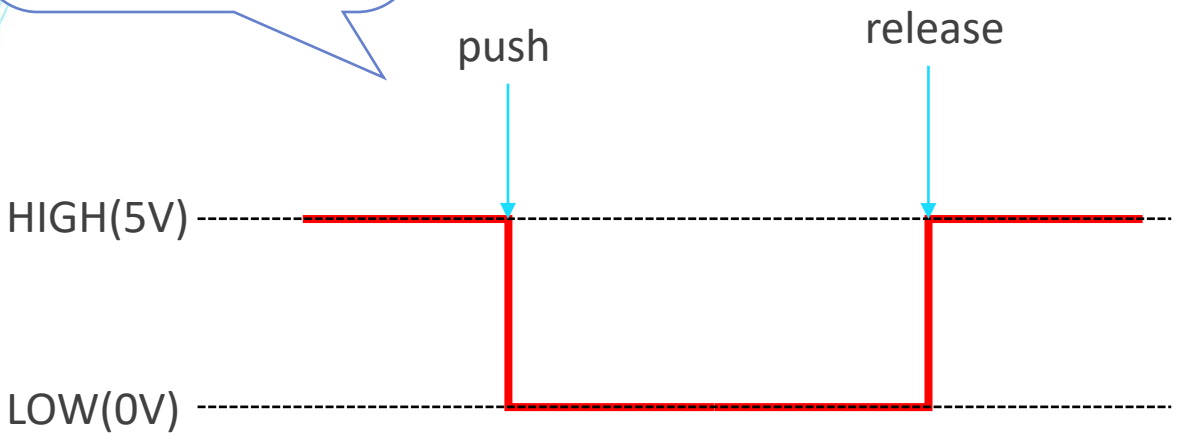
toggle1.ino

```
#define BTN 12
#define LED 13

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

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

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

토글 버튼의 구현

- INPUT_PULLUP 버튼의 상태도

이론적으로는 완벽하지만 실제 작동은 완벽하지 않음. 그 이유를 확인해 보자.

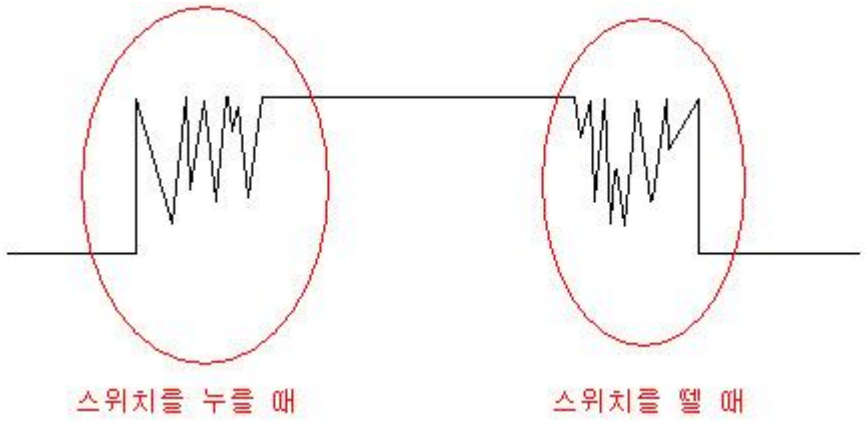


toggle2.ino

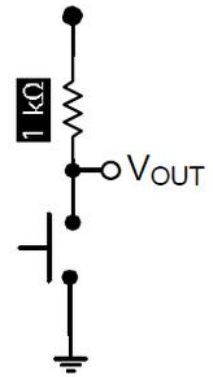
```
#define BTN 12
#define LED 13

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;
}
```

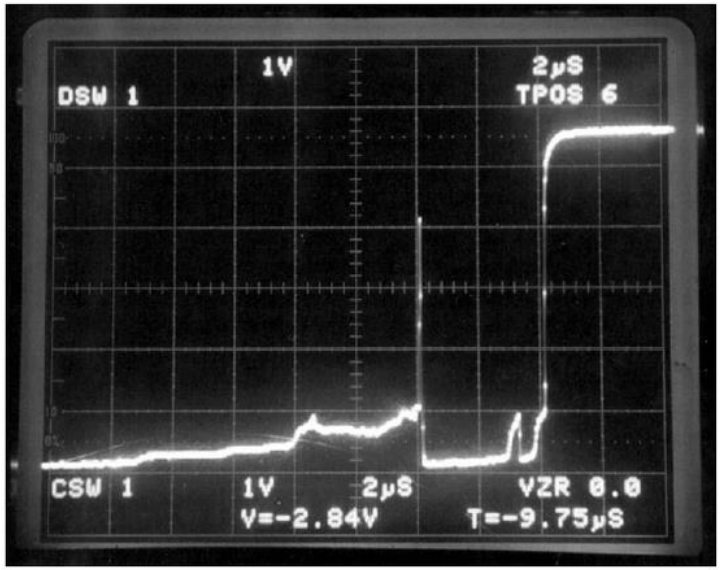
채터링(chattering)

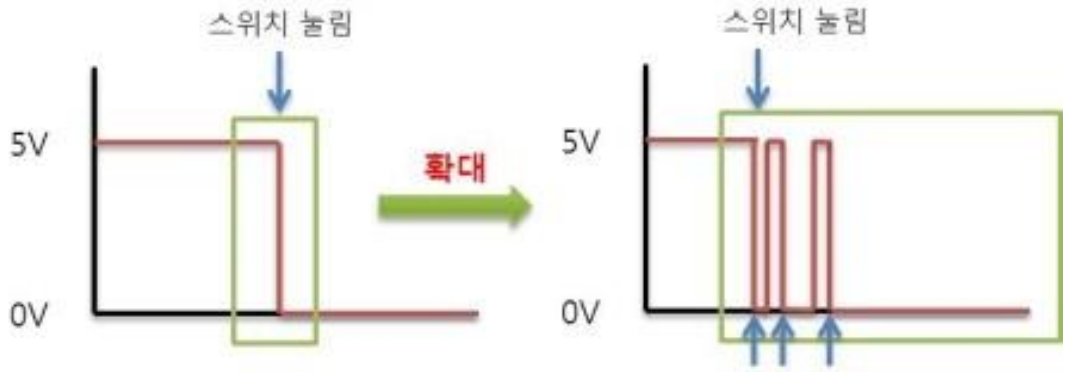
- Switch bounce 현상



- **Switch Bounce**



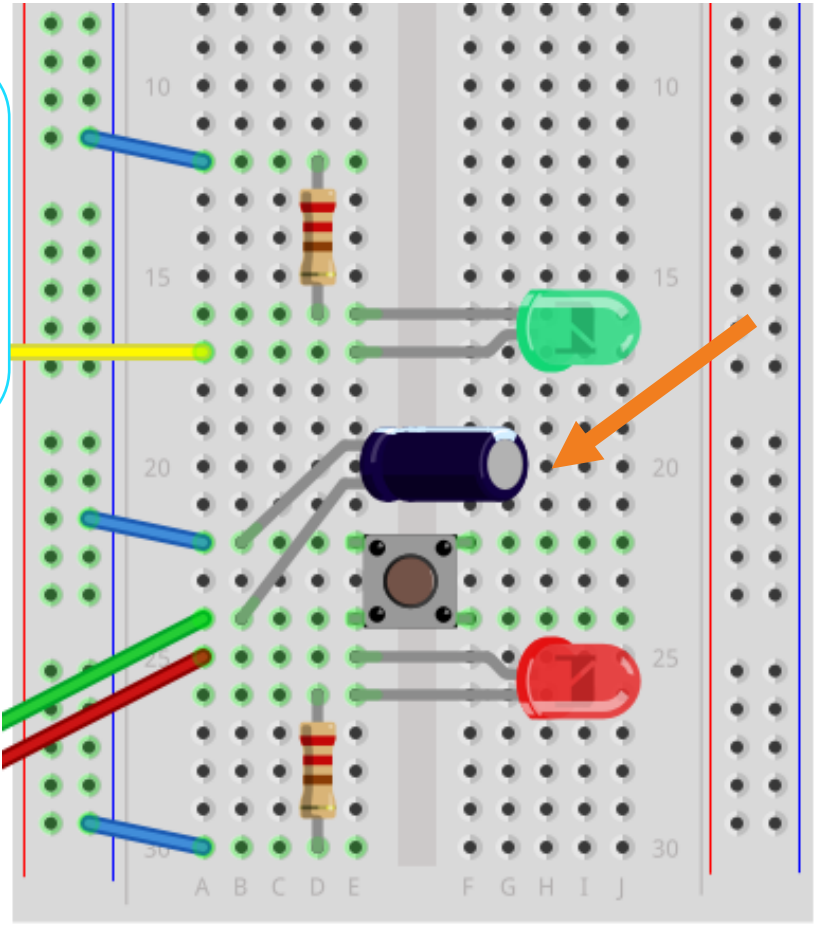




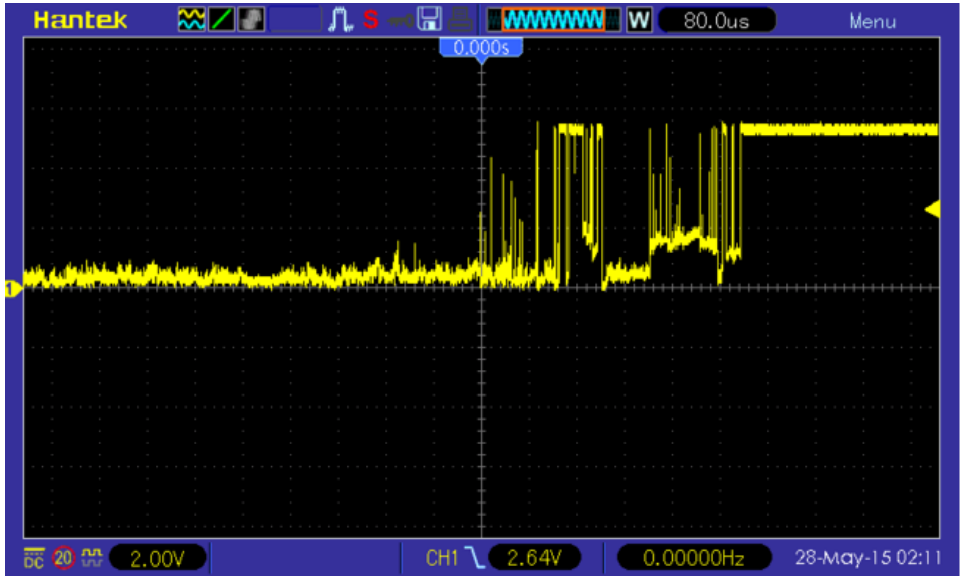
채터링 보정

- 하드웨어적인 해결책
  - 10uF 콘덴서 추가

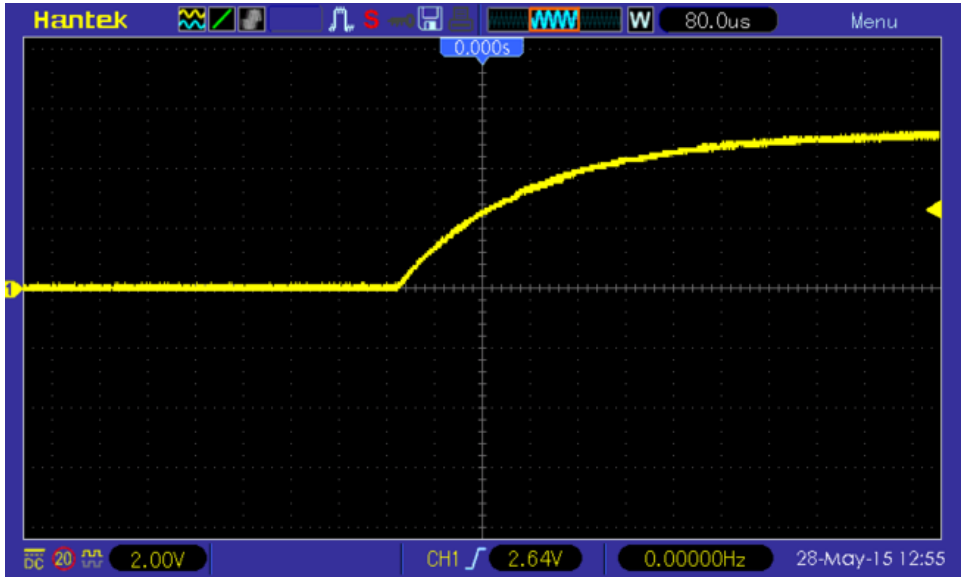
채터링 문제는
해결되었지만
버튼의 반응속도가
느려지는 부작용이
발생한다.



- 사용 전



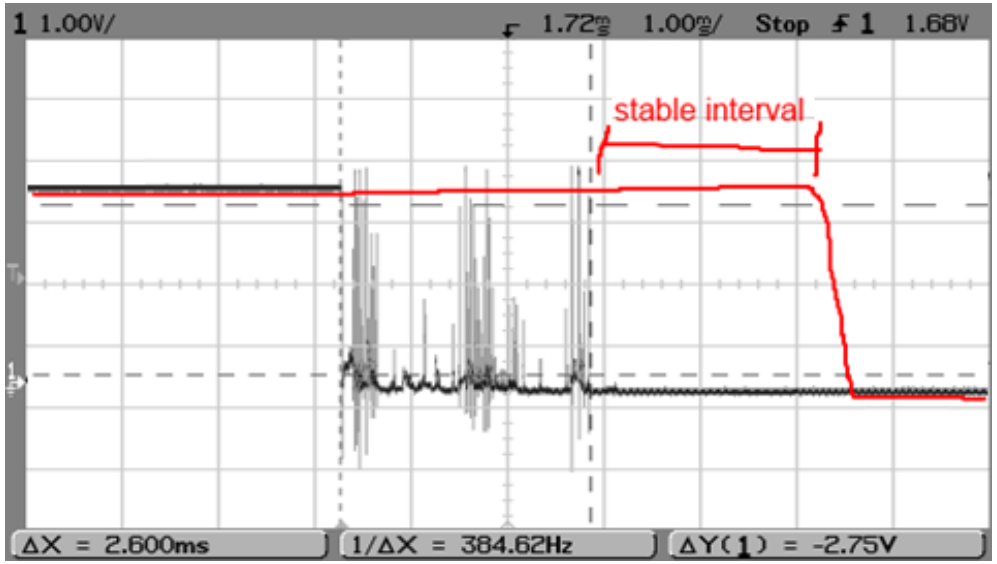
- 사용 후



채터링 현상은 사라졌지만 Rise time이 늘어나는 부작용 발생

채터링 보정(소프트웨어적인 방법)

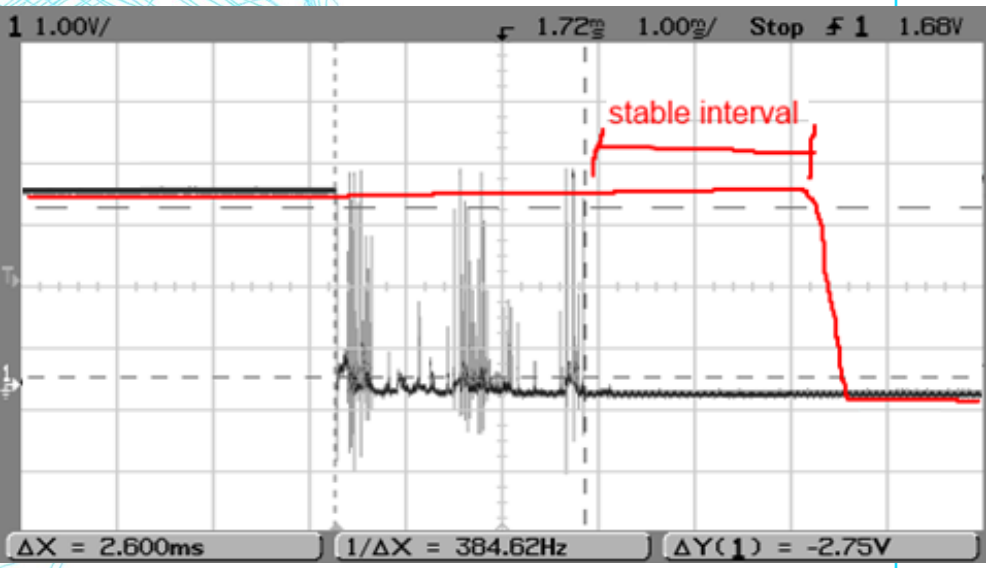
- Button 클래스 설계
  - 채터링 보정 기능
  - update() 메소드를 지속적으로 호출하면서 버튼 상태 감시



Button.h

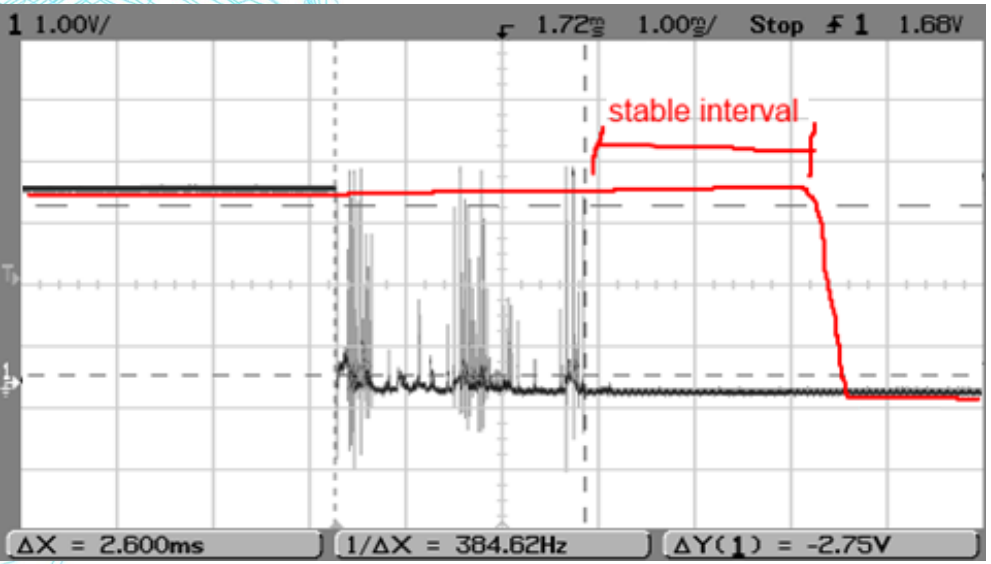
```
class Button {
  private:
    int  _pin;
    int  _interval;
    bool _stabled;      // 안정상태가 되었는가?
    bool _changed;      // 상태변화가 있었는가?
    bool _preState, _curState;
    bool _stableState;  // 현재 버튼 (안정)상태
    unsigned long _preTime, _curTime;

  public:
    Button() { }
    Button(int mode, int pin, int interval) {
      attach(mode, pin, interval);
    }

    void attach(int mode, int pin, int interval) {
      _pin = pin;
      pinMode(pin, mode);
      _interval = interval;
      _stableState = _preState = digitalRead(pin);
      _preTime  = millis();
      _stabled = false;
    }
```

Button




```
bool update() {   // 버튼 상태 업데이트(변화 있으면 true 반환)
  _curState = digitalRead(_pin);
  _curTime  = millis();

  if(_preState != _curState) {   // 이전과 지금 상태가 다르면,
    _stabled  = false;      // 변화가 일어나고 있는 중이므로 안정상태 아님
    _changed  = false;      // 아직 변한거 아님
    _preState = _curState;  // 버튼 상태 업데이트
    _preTime  = _curTime;   // 현재 시간 백업
    return false;
  }
  else {  // 이전과 지금 상태가 같은데...
    if(_curTime - _preTime >= _interval) {  // 인터벌 시간 이상 지속되고,
      if(_stableState != _curState)  { // 안정상태가 새롭게 바뀌었으면,
        _stableState = _curState; // 현재 상태를 안정상태로 기록
        _stabled = true;          // 안정상태 맞음
        _changed = true;          // 상태 변화가 일어남
        _preState = _curState;    // 버튼 상태 업데이트
        _preTime  = _curTime;     // 현재 시간 백업
        return true;
      }
    }
    // 아직 인터벌 시간 이상 지속되지 않았거나,
    // 안정상태가 바뀐 것도 아니면,
    _stabled = false;   // 안정상태 아님
    _changed  = false;  // 아직 변한거 아님
    return false;
  }
}
```

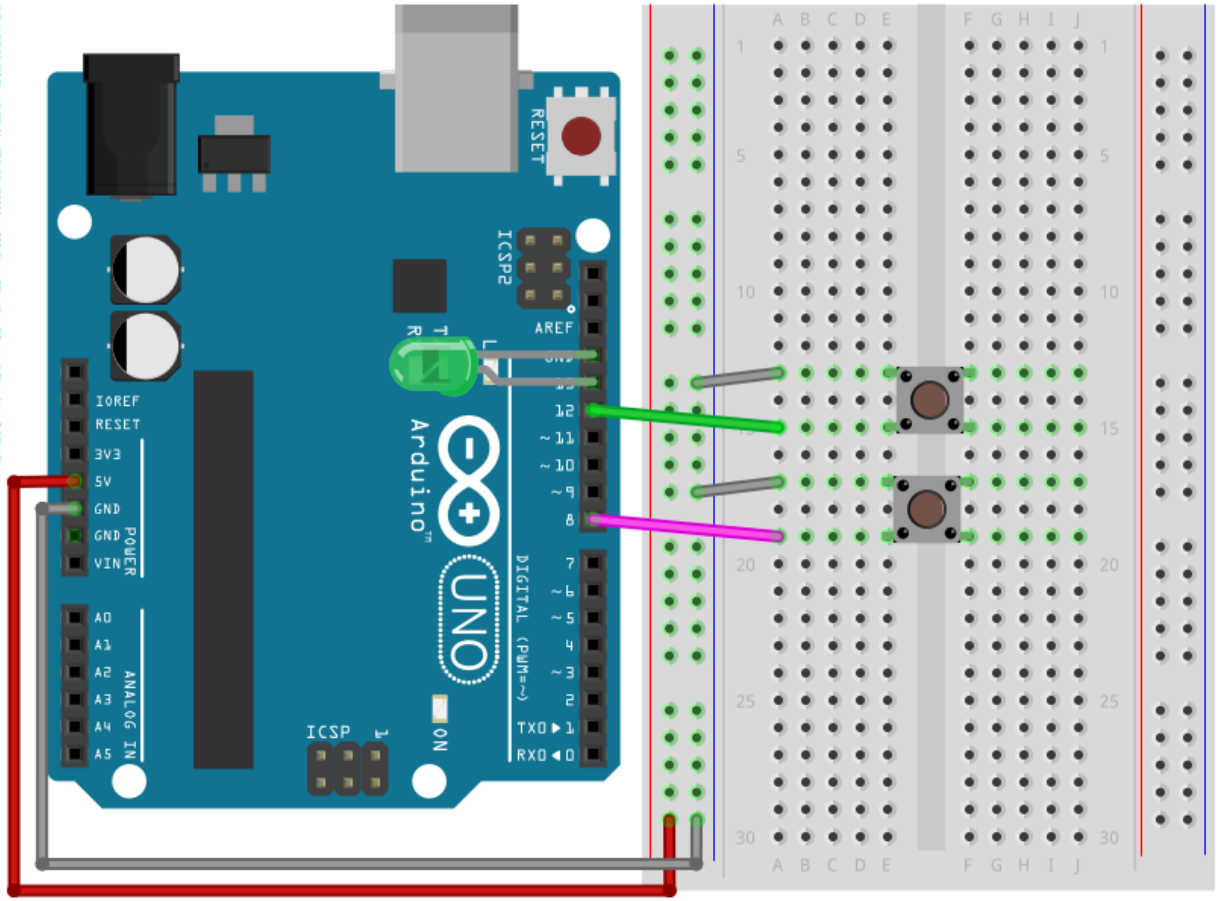
Button 클래스 설계




```
    bool read() {   // 안정상태 버튼 값 반환
      return _stableState;
    }

    bool rose() {   // 최근의 안정상태 변화가 상승에지 였는가?
      return (_changed && _stableState==HIGH)? true:false;
    }

    bool fell() {   // 최근의 안정상태 변화가 하강에지 였는가?
      return (_changed && _stableState==LOW)? true:false;
    }

    bool changed() {
      return _changed;
    }
};
```

Button 클래스 이용

- 회로



toggle3.ino

```
#include "Button.h"
#define BTN 12
#define LED 13
// 버튼을 12번핀에 INPUT_PULL모드로 부착
// 40ms 이상 상태가 지속되어야 안정상태로 인정
Button btnToggle(INPUT_PULLUP, BTN, 40);

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

void loop() {
  static int n=0;
  btnToggle.update();

  if(btnToggle.fell()) { // 누르는 순간 작동
    Serial.println(++n);
    digitalWrite(LED, !digitalRead(LED));
  }
}
```

상승에지·하강에지 감지하기

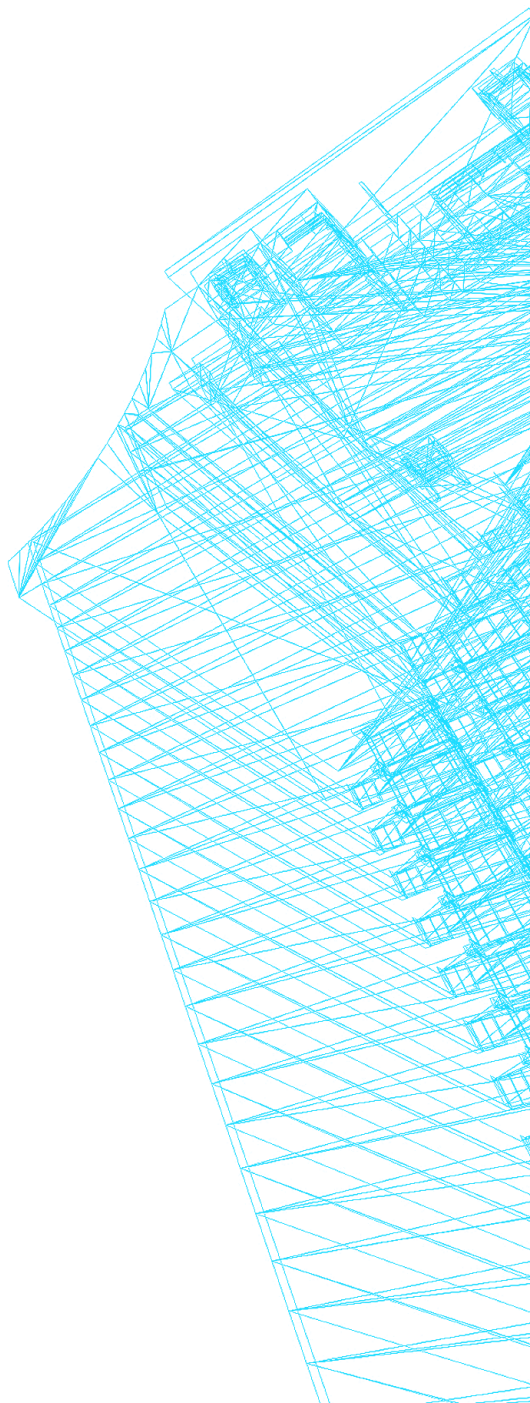
edgeDetect.ino

```
// 버튼 두 개일 때,
// 버튼 누를 때와 땔 때를 모두 감지하는 방법 예시

#include "Button.h"

#define BTN_TOGGLER  8
#define BTN_COUNTER 12
#define LED 13

Button btnToggler(INPUT_PULLUP, BTN_TOGGLER, 40);
Button btnCounter(INPUT_PULLUP, BTN_COUNTER, 40);
// INPUT_PULLUP 버튼이므로
// 누르면 falling edge, 떼면 rising edge가 된다.

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

```
void loop() {
  btnToggler.update();
  btnCounter.update();

  if(btnToggler.rose()) { // 눌렀다 뗄 때 작동
    digitalWrite(LED, !digitalRead(LED));
  }

  static int n=0;
  if(btnCounter.changed()) { // 변화 발생
    ++n;  // 변화 횟수: 눌러도+1, 떼어도+1
    if(btnCounter.fell())
      Serial.print("[fell] ");
    else if(btnCounter.rose())
      Serial.print("[rose] ");

    if(n%2 == 0)
      Serial.println(n/2);
  }
}
```

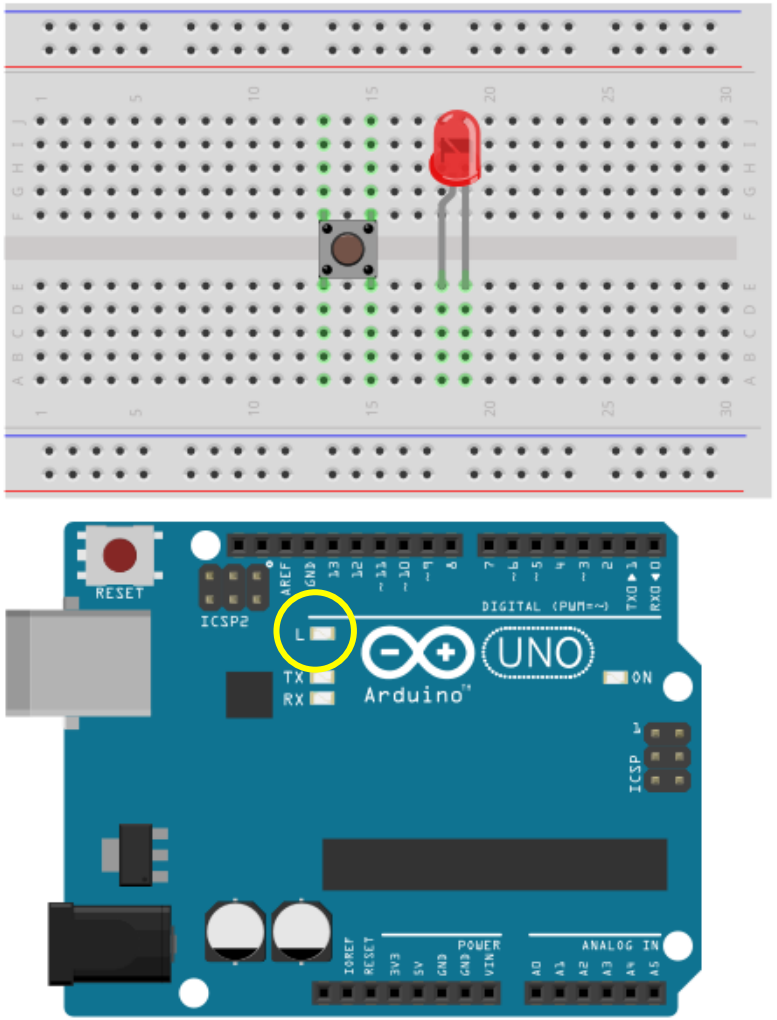
2. 멀티태스킹 구현하기

- delay() 함수의 덫
- millis() 함수 소개
- SimpleTimer 클래스 설계
- 멀티태스킹 실험
- 타이머



[실습퀴즈] delay() 함수의 덫

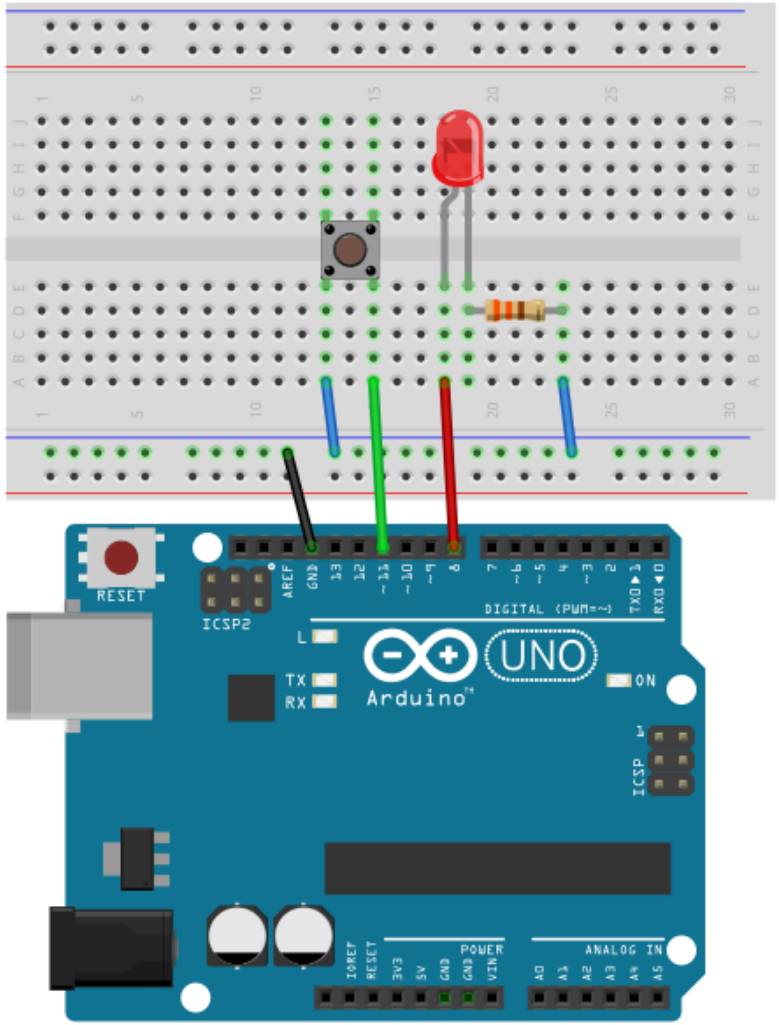
- 회로 구성



- 아래와 같은 일을 하는 회로를 구성하고, 제어 프로그램을 작성하시오.
  - 13번 핀에 연결된 내장 LED는 2초 간격으로 계속 깜박인다.
  - 11번 핀에 연결된 버튼을 누르면 8번 핀에 연결된 LED가 즉시 켜지고, 버튼에서 손을 때면 즉시 꺼진다.

[실습퀴즈] delay() 함수의 덫

- 회로 구성



- 버튼
  - 11번에 연결됨
  - ACTIVE LOW로 설계
  - 내부 풀업 저항 사용
  - INPUT_PULLUP 사용해야 함.
- RED LED
  - 8번에 연결
  - ACTIVE HIGH

[실습퀴즈] delay() 함수의 덫

- 소스코드

delay_side_effect.ino

```
#define RED  8
#define BTN 11
#define LED 13

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

```
void revertLed() {
  digitalWrite(LED, !digitalRead(LED));
  delay(2000);
  // 2초 동안 먹통이 됨.
}

void loop() {
  revertLed();
  digitalWrite(RED, !digitalRead(BTN));
}
```

millis() 함수

- 해설
  - 아두이노가 현재 프로그램의 실행을 시작한 이후 흐른 시간을 밀리초 단위로 반환한다. 이 숫자는 대략 50일 후 오버플로우(0으로 되돌아감) 될 것이다.
- 리턴
  - 프로그램이 시작된 이후 지나간 시간의 밀리초(unsigned long)
- 주의
  - millis() 함수의 리턴 값이 unsigned long 이라는 것에 주의
  - 보다 작은 타입(int 자료형 등)과 연산 시 주의 필요
- 예시

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

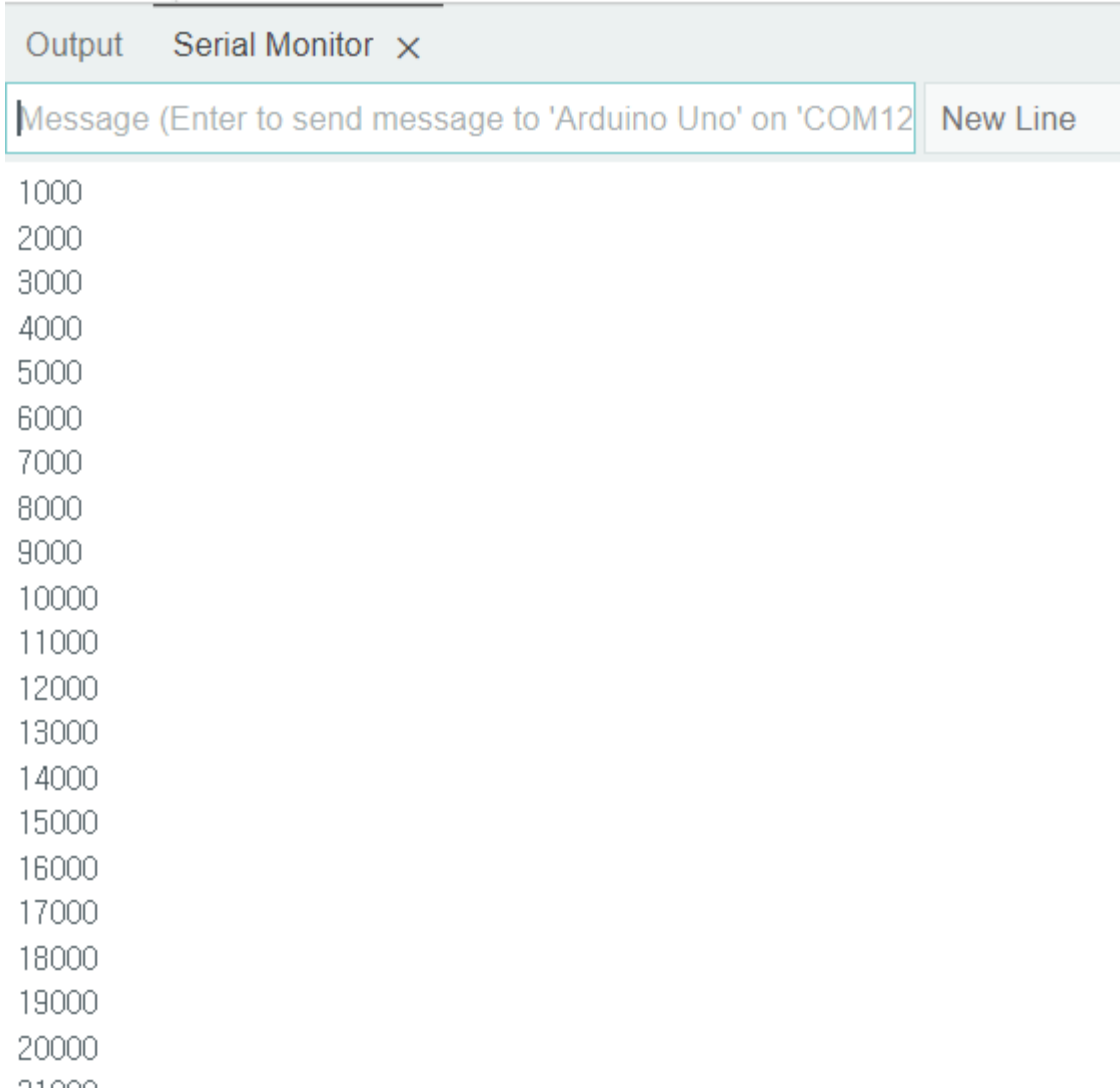
millis() 함수

- 소스코드

millis1.ino

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

int preSec = 0;

void loop() {
  unsigned long now = millis();
  int nowSec = now/1000; //초단위로 변경

  if(preSec != nowSec) {
    Serial.println(now);
    preSec = nowSec;
  }
}
```

- 실행결과



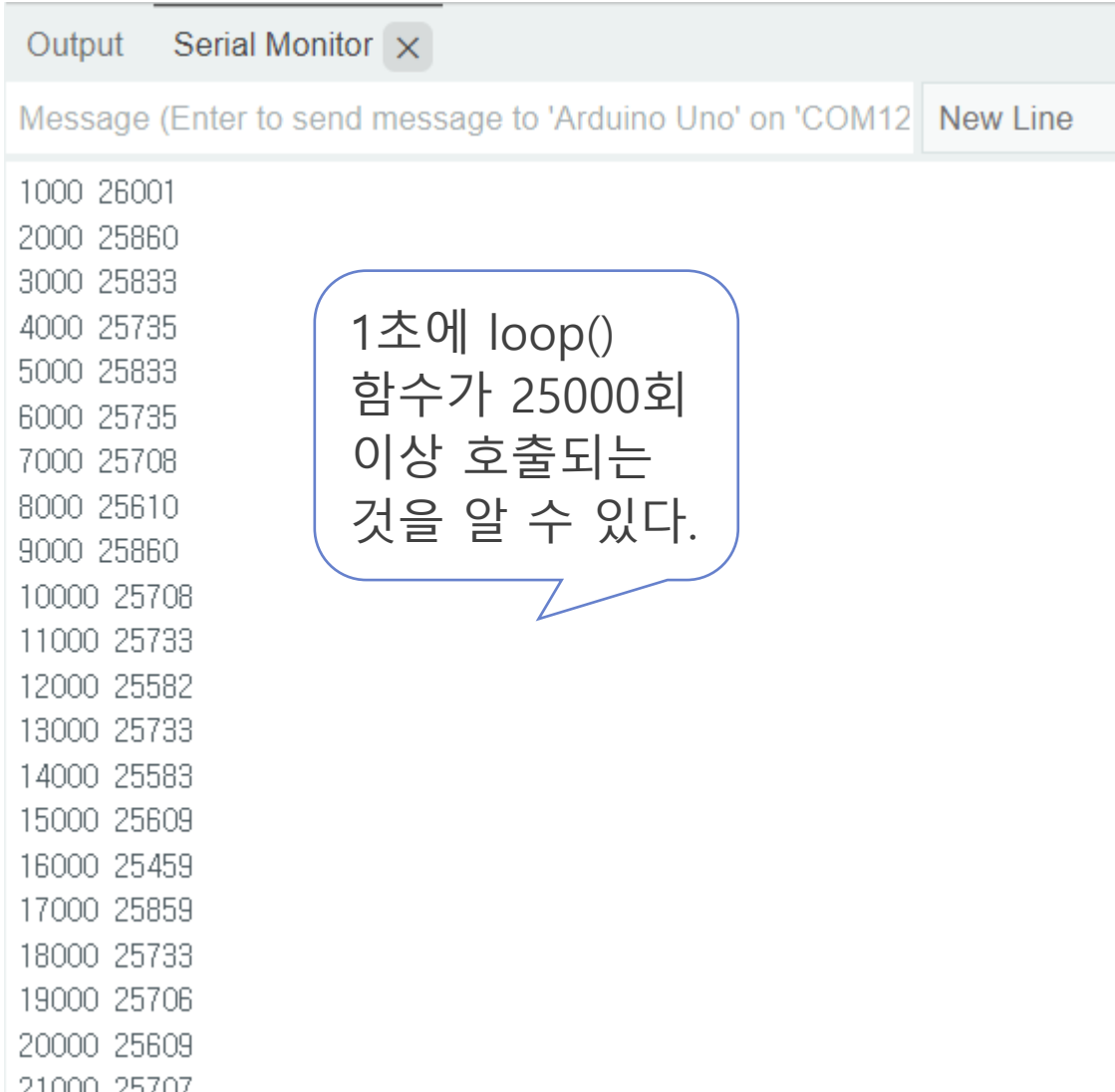
millis() 함수

- 소스코드

millis2.ino

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

int preSec = 0;
int loopCnt = 0;

void loop() {
  unsigned long now = millis();
  int nowSec = now/1000;

  if(preSec != nowSec) {
    Serial.print(now);
    preSec = nowSec;

    Serial.print(" ");
    Serial.println(loopCnt);
    loopCnt=0;
  }
  loopCnt++;
}
```

- 실행결과



[퀴즈정답] delay() 함수의 덫

- millis() 함수 이용 흐른 시간을 측정하여 반전

delay_x.ino

```
#define RED  8
#define LED 13
#define BTN 11

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

void loop() {
  revertLED();
  digitalWrite(RED, !digitalRead(BTN));
}
```

```
unsigned long preTime = 0;

void revertLED() {
  unsigned long curTime = millis();
  // curTime = 현재시간

  //(현재시각 - 이전시간 >= 2초)
  if(curTime - preTime >= 2000) {  // 2초 지났니?
    digitalWrite(LED, !digitalRead(LED));
    preTime = curTime;
  }
}
```

```
void revertLed() {
  digitalWrite(LED, !digitalRead(LED));
  delay(2000);  // 2초 동안 먹통이 됨.
}
```

millis()를 이용한 방법 고찰

- 문제점
  - delay() 함수를 회피하기 위해서 매번 시간 백업 변수를 만들고 현재 시간을 측정하여 원하는 시간이 흘렀는지 감시하는 루틴의 작성이 요구됨.
  - 시간에 의존하는 작업이 1개 늘어 날 때마다,
  - 시간 백업을 위한 전역 변수를 또 만들고 시간을 감시하는 루틴을 계속 작성해야 하므로 소스코드가 복잡해짐.
- 해결책
  - 정해진 시간마다 원하는 함수를 자동으로 호출해주는 시간관리 모듈이 필요함.
  - 호출해야 하는 함수와 시간간격을 한 번 등록해 두면 시간 감시와 함수 호출의 자동화가 이루어져야 함.
  - 이 모든 것을 단순하게 만들어주는 클래스를 구현 해보자!!

MillisTimer 클래스 설계

MillisTimer.h

```
#define MAX_TIMERS  10
typedef unsigned long ulong;
typedef void (*timer_callback)(void);

typedef struct {
  bool enabled;
  timer_callback func;
  ulong interval;
  ulong preTime;
} timer;

static inline ulong elapsed() {
  return millis();
}

class MillisTimer {
  private:
    // 이전에 run() 함수가 실행된 시간
    static ulong _pre_run_ms;
    static ulong _refresh_ms;
    static int   _num_timers; // 등록된 타이머 개수
    static timer _timers[MAX_TIMERS];
```

	_timers
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

```
  public:
    static void add(timer_callback f,
                    ulong interval) {
      _timers[_num_timers].enabled = true;
      _timers[_num_timers].interval = interval;
      _timers[_num_timers].func = f;
      _timers[_num_timers].preTime = elapsed();
      _num_timers++;
    }

    static void enable(int n) {
      _timers[n].enabled = true;
    }

    static void disable(int n) {
      _timers[n].enabled = false;
    }

    static void get_refresh_ms() {
      return _refresh_ms;
    }
```

MillisTimer 클래스 설계

```
    // this function must be called inside loop()
    static void run() {
      ulong now_ms = elapsed();
      _refresh_ms = now_ms - _pre_run_ms;
      _pre_run_ms = now_ms;

      for(int i=0; i<_num_timers; i++) {
        if(_timers[i].enabled) {
          ulong diff = now_ms - _timers[i].preTime;
          if(diff >= _timers[i].interval) {
            (*_timers[i].func)();
            _timers[i].preTime +=timers[i].interval;
          }
        }
      }
    }
};

static ulong MillisTimer::_pre_run_ms;
static ulong MillisTimer::_refresh_ms;
static int   MillisTimer::_num_timers;
static timer MillisTimer::_timers[MAX_TIMERS];
```

MillisTimer 사용 예시

revert_millis_timer.ino

```
#include "MillisTimer.h"

#define RED  8
#define BTN 11
#define LED 13

void setup() {
  pinMode(RED, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(BTN, INPUT_PULLUP);
  MillisTimer::add(revertLed, 2000);
}

void revertLed() {
  digitalWrite(LED, !digitalRead(LED));
}

void loop() {
  MillisTimer::run();
  digitalWrite(RED, !digitalRead(BTN));
}
```

3가지 작업 멀티태스킹 실험

test_multitasking.ino

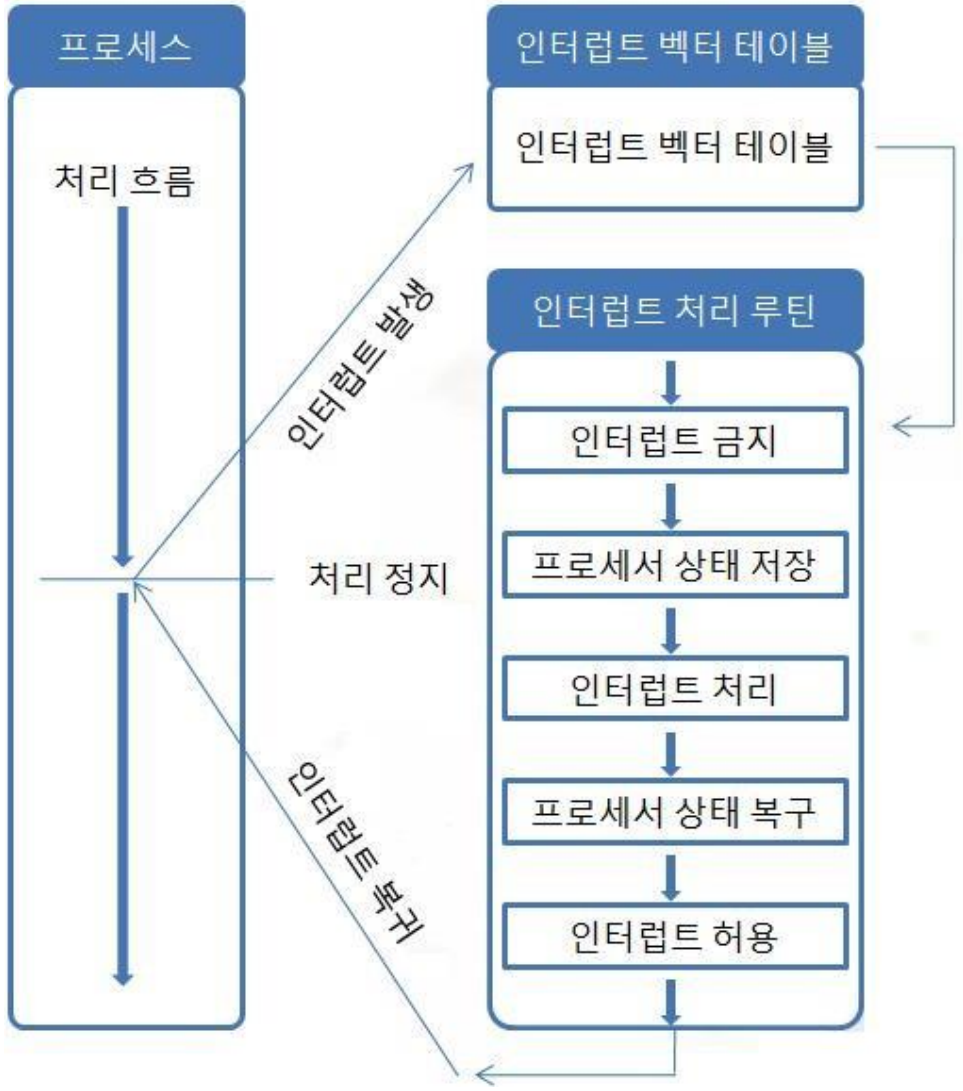
```ino
#include "MillisTimer.h"
#define RED  8
#define LED 13
#define BTN 11

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

  MillisTimer::add(revertLed, 500);
  MillisTimer::add(checkBtn, 1);
  MillisTimer::add(printDigitalClock, 1000);
}

void revertLed() {
  digitalWrite(LED, !digitalRead(LED));
}

void checkBtn() {
  digitalWrite(RED, !digitalRead(BTN));
}
```

```ino
void printDigitalClock() {
  int h, m, s;
  s = millis() / 1000;
  m = s / 60;
  h = s / 3600;
  s = s - m * 60;
  m = m - h * 60;
  Serial.print(h);
  printDigits(m);
  printDigits(s);
  Serial.println();
}

void printDigits(int digits) {
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}

void loop() {
  MillisTimer::run();
}
```

[MillisTimer 담당]
1) 0.5s마다 13번 LED 토글
2) 1ms마다 버튼 눌림 확인
3) 1.0s마다 디지털시계 출력

타이머 인터럽트

- 인터럽트란?
  - CPU가 특정 기능을 수행하는 도중에 급하게 다른 일을 처리하고자 할 때 사용할 수 있는 기능
  - CPU는 한 순간에는 하나의 일 밖에 처리할 수 없기 때문에 어떤 일을 처리하는 도중에 우선순위가 급한 일을 처리할 필요가 있을 때 대처할 수 있는 방안이 필요



타이머 인터럽트

- 아두이노 Uno와 Mega의 타이머 요약

구분	보드	해상도	Uno		Mega	
			관련 기능	PWM pin	관련 기능	PWM pin
timer0	Uno, Mega	8	millis() micros() delay()	5, 6	millis() micros() delay()	13, 4
timer1	Uno, Mega	16	I2C Servo	9, 10	I2C	11, 12
timer2	Uno, Mega	8	tone()	3, 11	tone()	10, 9
timer3	Mega	16				5, 2, 3
timer4	Mega	16				6, 7, 8
timer5	Mega	16			Servo	44, 45, 46

타이머 인터럽트

- 관련 레지스터
  - **TCCRx** - Timer/Counter Control Register. (타이머/카운터 제어)
    - The pre-scaler can be configured here.
  - **TCNTx** - Timer/Counter Register. (타이머/카운터 실제값)
    - The actual timer value is stored here.
  - **OCRx** – Output Compare Register (비교값)
  - ICRx - Input Capture Register (only for 16bit timer)
  - **TIMSKx** - Timer/Counter Interrupt Mask Register. (타이머 인터럽트 활성화)
    - To enable/disable timer interrupts.
  - TIFRx - Timer/Counter Interrupt Flag Register. (인터럽트 발생여부를 저장)
    - Indicates a pending timer interrupt.

타이머 인터럽트

- 아두이노 오실레이터 기본 값 = 16Mhz
- Prescaler 설정

Table 17-6. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

타이머 인터럽트

- TCCRn (Timer / Counter Contorl Register)

TCCR0 - Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR1A - Timer/Counter1 Control Register A

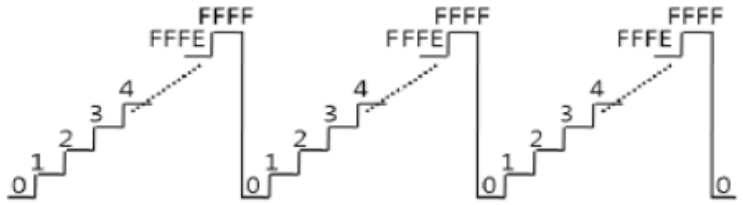
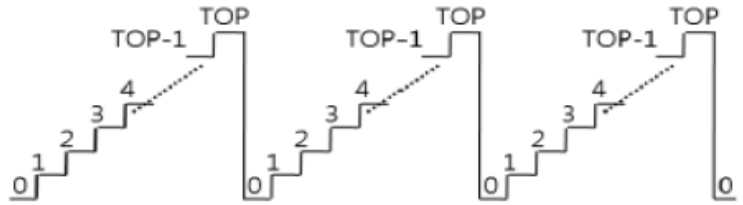
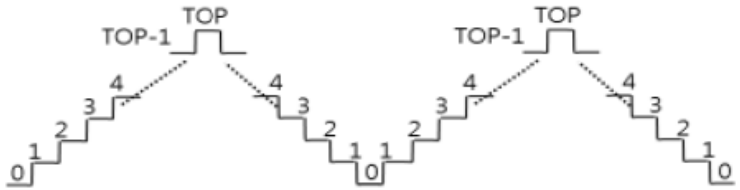
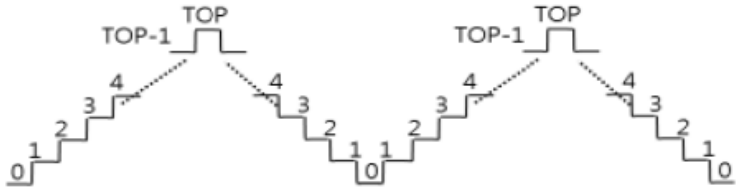
Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR1B - Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR1C - Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
	FOC1A	FOC1B	FOC1C	–	–	–	–	–	TCCR1C
Read/Write	W	W	W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

파형 발생모드	TCCRnB 레지스터		TCCRnA 레지스터		최댓값 (TOP)	TCNTn 카운터의 16진 계수 순서
	WGMn3	WGMn2	WGMn1	WGMn0		
일반 모드	0	0	0	0	FFFF	
고속 PWM 모드	0	1	0	1	FF	
	0	1	1	0	1FF	
	0	1	1	1	3FF	
	1	1	1	0	ICRn	
	1	1	1	1	OCRnA	
위상정정 PWM 모드	0	0	0	1	FF	
	0	0	1	0	1FF	
	0	0	1	1	3FF	
	1	0	1	0	ICRn	
	1	0	1	1	OCRnA	
위상–주파수 정정 PWM 모드	1	0	0	0	ICRn	
	1	0	0	1	OCRnA	
CTC 모드 (Clear Timer on Compare Mode)	0	1	0	0	OCRnA	
	1	1	0	0	ICRn	

- CTC Mode 라는 비교일치 인터럽트를 이용하여 주기적으로 특정 함수 실행 가능
- CTC Mode(Clear Timer on Compare Mode) 는 클럭 주기마다 TCNT(Timer Counter Register)의 값이 0부터 1씩 증가하고, 그 값이 OCR(Output Compare Register) 값 또는 ICR(Inturrupt Control Register)과 일치하면 0으로 초기화 된다.
- 그리고 다시 증가를 반복한다.

타이머 인터럽트 소스코드

timer1_interrupt.ino

```
void timer1_setup() {
  cli();  //stop interrupts
  // set timer count for 2Hz increments
  // f(주파수 Hz) = 1 / T(주기)
  // (2Hz = 0.5초마다 1회 호출되도록 만들려면 아래와 같이 세팅한다.)

  TCCR1A = 0;  // set entire TCCR1A register to 0
  TCCR1B = 0;  // set entire TCCR1B register to 0
  TCNT1  = 0;  //initialize counter value to 0;
  // turn on CTC(Clear Timer on Compare Match) mode
  TCCR1B |= (1 << WGM12);
  // Set CS12 bit for 256 prescaler
  TCCR1B |= (1 << CS12);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A); // TIMSKx  Timer/Counter Interrupt Mask Register.

  // 아래 숫자를 무엇으로 하느냐에 따라 호출 주기가 달라진다.
  // 계산방법: 16Mhz(CPU속도)/Prescaler/원하는 주기
  // 여기에서는 Prescaler가 256이므로...
  OCR1A = 31250;   // 16MHz/256/2Hz = 31250
  // 만약 1초에 10번 호출되게 하려면,  16Mhz/256/10Hz => 6250로 지정하면 됨.
  sei();  //allow interrupts
}
```

타이머 인터럽트 소스코드

timer1_interrupt.ino

```
// 아래 ISR 함수는 자동으로 1초에 2번 호출된다.
// 심지어 loop()함수의 delay()무시
// timer compare interrupt service routine
ISR(TIMER1_COMPA_vect) {
  // ISR은 빠른 속도로 재 호출되므로 빠르게 리턴될 수 있도록 가능한한 간결하게 작성하여야 한다.
  // ISR 내부에서 delay 함수 사용 금지
  digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  timer1_setup();
}

void loop() {
  // LED 깜빡임은 타이머 인터럽트에 의해 알아서 깜빡인다.
  // 여기서는 다른 작업에 신경쓰는 코드를 작성하면 됨.
  // 심지어 아래에서와 같이 delay()를 사용해도 타이머 인터럽트 서비스루틴[ISR(TIMER1_COMPA_vect)]은
  // 지정한 주기마다 제대로 호출된다.
  delay(1000);
}
```

IntTimer1 클래스 (MillisTimer의 인터럽트 버전)

```
#define MAX_TIMERS  10
typedef unsigned long ulong;
typedef void (*timer_callback)(void);
typedef struct {
  bool enabled;
  timer_callback func;
  ulong interval;
  ulong preTime;
} work;

class IntTimer1 {
  private:
    volatile static ulong _now_ms;
    volatile static ulong _refresh_ms;
    volatile static int   _num_timers;
    volatile static work _timers[MAX_TIMERS];
  public:
    static void add(timer_callback f, ulong interval) {
      _timers[_num_timers].enabled = true;
      _timers[_num_timers].interval = interval;
      _timers[_num_timers].func = f;
      _timers[_num_timers].preTime = _now_ms;
      _num_timers++;
    }
```

IntTimer1.h

```
    static void start() {
      cli();  //stop interrupts

      TCCR1A = 0;
      TCCR1B = 0;
      //initialize counter value to 0;
      TCNT1  = 0;
      TCCR1B |= (1 << WGM12);

      // Set prescaler to 64
      TCCR1B |= ((1<<CS11)|(1<<CS10));
      // 1초에 200번(0.05초 마다) 호출되도록
      setRefreshRate(200);

      // enable timer compare interrupt
      TIMSK1 |= (1 << OCIE1A);

      sei();  //allow interrupts
    }

    static void stop() {
      TIMSK1 &= ~(1 << OCIE1A);
    }
```

IntTimer1 클래스 (MillisTimer의 인터럽트 버전)

```
static void resume() {
  TIMSK1 |= (1 << OCIE1A);
}

static void setRefreshRate(int hz) {
  _refresh_ms = 1000/hz;
  OCR1A = 16000000UL/64/hz;
}

static void overflow() {
  _now_ms += _refresh_ms;
  for(int i=0; i<_num_timers; i++) {
    if(_timers[i].enabled) {
      if(_now_ms - _timers[i].preTime >= _timers[i].interval) {
        if(_timers[i].func != NULL) {
          (*_timers[i].func)();
          _timers[i].preTime += _timers[i].interval;
        }
      }
    }
  }
}
```

IntTimer1 클래스 (MillisTimer의 인터럽트 버전)

```
    static void enable(int n) {
      _timers[n].enabled = true;
    }
    static void disable(int n) {
      _timers[n].enabled = false;
    }
};

volatile static ulong IntTimer1::_now_ms;
volatile static ulong IntTimer1::_refresh_ms;
volatile static int   IntTimer1::_num_timers;
volatile static work  IntTimer1::_timers[MAX_TIMERS];

ISR(TIMER1_COMPA_vect) {
  // ISR은 빠른 속도로 재 호출되므로 빠르게 리턴될 수 있도록 가능한한 간결하게 작성하여야 한다.
  // ISR 내부에서 delay 함수 사용 금지
  IntTimer1::overflow();
}
```

4가지 작업 멀티태스킹 실험

```ino
#include "IntTimer1.h"
#define RED  8
#define LED 13
#define BTN 11

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

  IntTimer1::add(revertLed, 500);
  IntTimer1::add(checkBtn, 1);
  IntTimer1::add(printDigitalClock, 1000);
  IntTimer1::start();
}

void revertLed() {
  digitalWrite(LED, !digitalRead(LED));
}

void checkBtn() {
  digitalWrite(RED, !digitalRead(BTN));
}
```

test_multitasking.ino

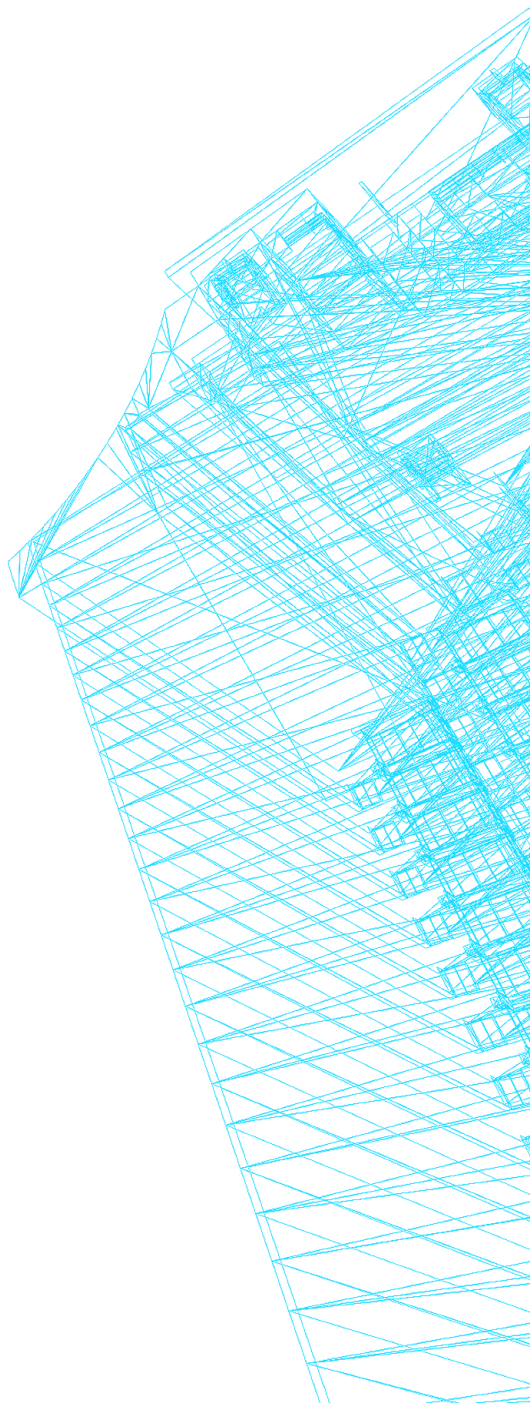
```ino
void printDigitalClock() {
  int h, m, s;
  s = millis() / 1000;
  m = s / 60;
  h = s / 3600;
  s = s - m * 60;
  m = m - h * 60;
  Serial.print(h);
  printDigits(m);
  printDigits(s);
  Serial.println();
}

void printDigits(int digits) {
  Serial.print(":");
  if(digits < 10) Serial.print('0');
  Serial.print(digits);
}

void loop() {
  //MillisTimer::run(); // 불필요
  static int count=0;
  Serial.println(count++);
  delay(200);  // loop에 delay()써도 타이머가 정상 작동됨
}
```

[IntTimer1 담당]
1) 0.5s마다 13번 LED 토글
2) 1ms마다 버튼 눌림 확인
3) 1.0s마다 디지털시계 출력

[loop() 담당]
4) 0.2초 마다 카운터 증가 출력

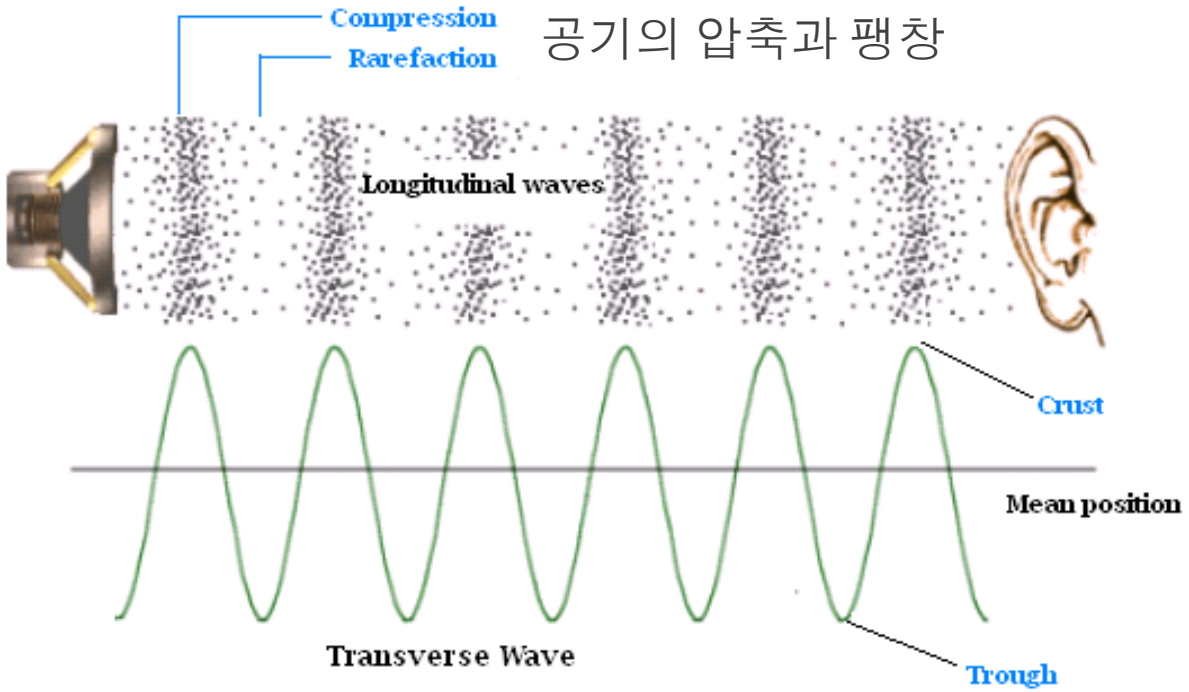
3. 부저로 연주하기

- Piezo buzzer
- 음과 주파수
- 소리 출력 - MelodyPlayer 클래스 설계
- 악보 입력 – CNote 클래스 소개
- 배경음악 연주 - BgmPlayer 클래스 설계



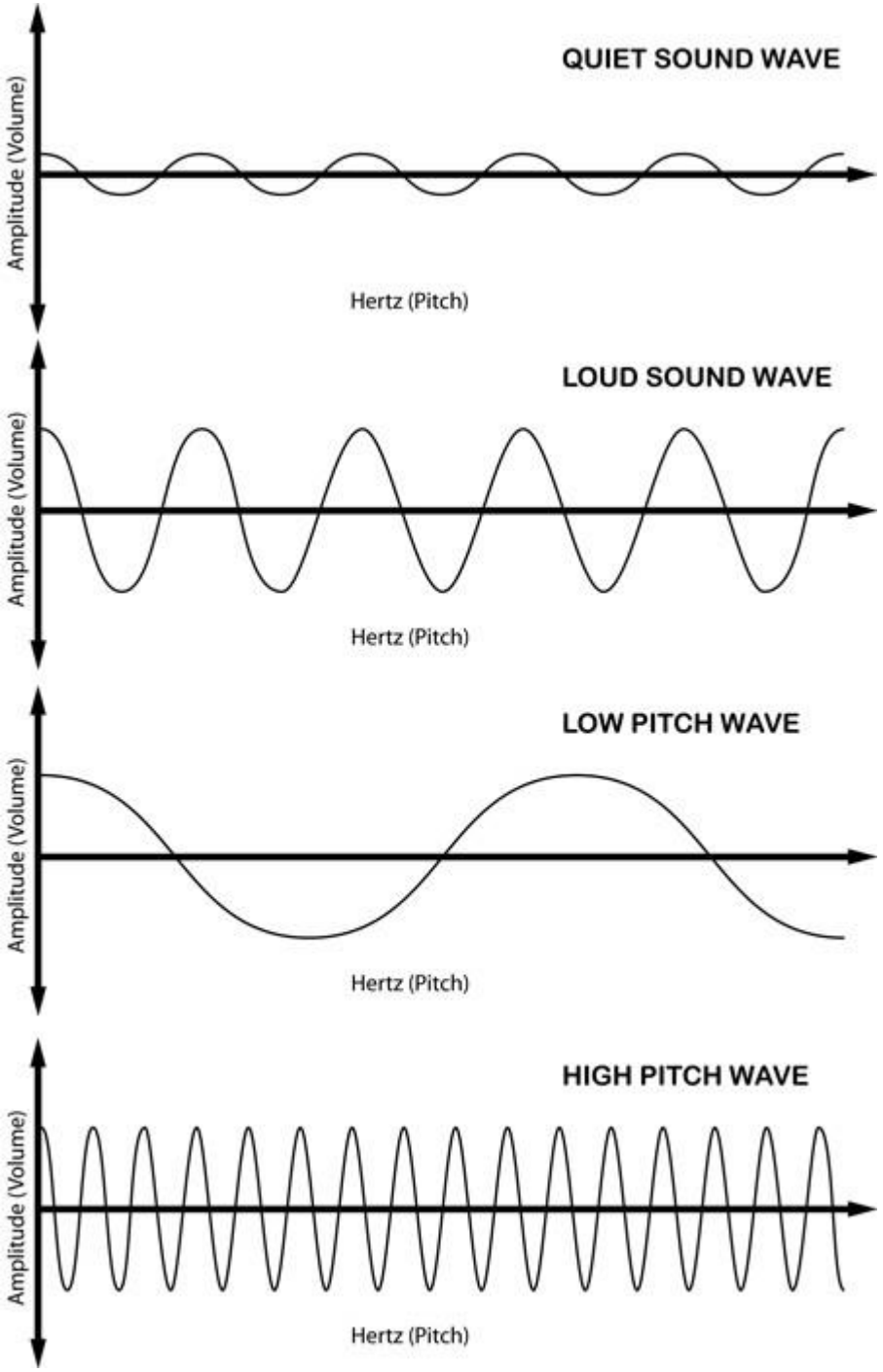
음과 주파수

- 음파



공기의 압축과 팽창

소리의 크고 작음은 진폭에 의해서,
소리의 높고 낮음은 주파수에 의해서 결정

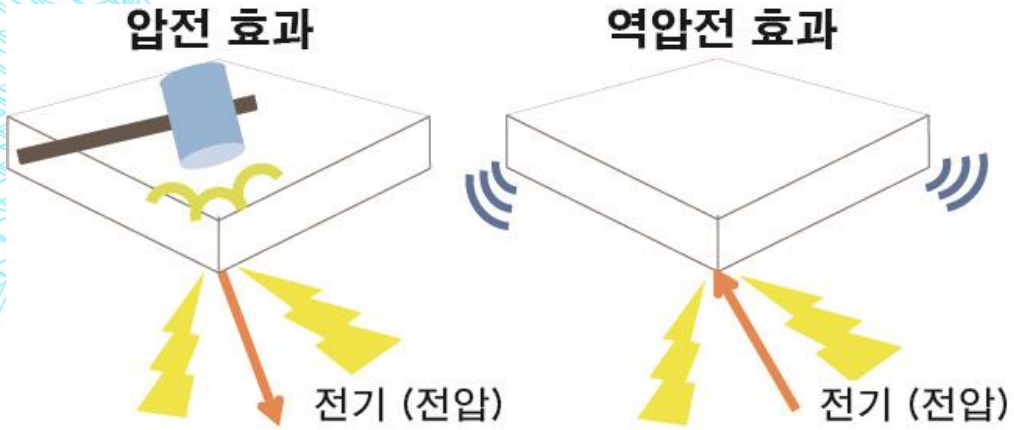


Buzzer

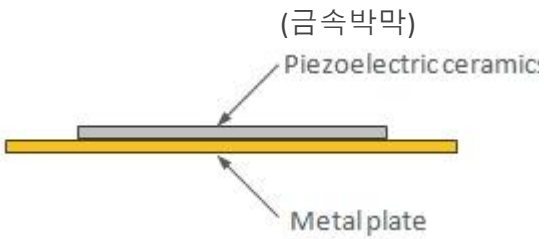
- An **active buzzer** will generate a tone using an internal oscillator, so all that is needed is a DC voltage.

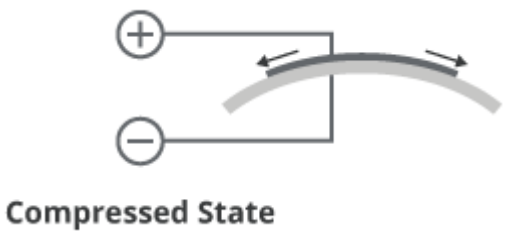
- The **passive buzzer** does not have an internal oscillating source, and it has to be driven with square wave and different frequency needed. It is like an electromagnetic speaker, where a changing input signal produces the sound.

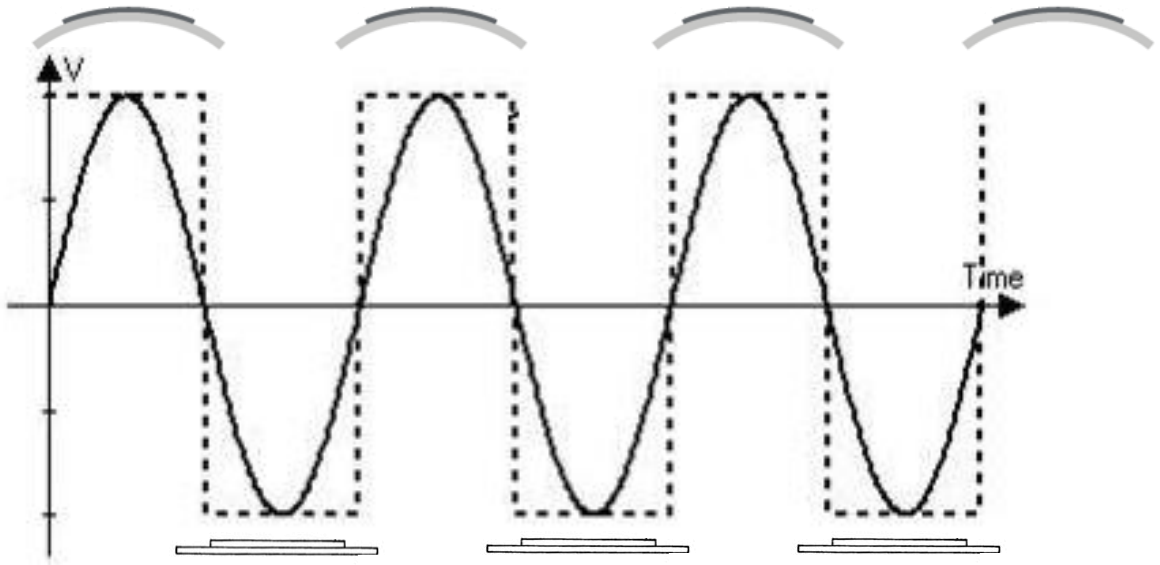
Piezo Buzzer



피에조는 압전 소자 (Piezoelectric Element, Piezoelectric Device)를 뜻한다.
압전 소자는 힘 (압력)을 가함으로써 전압을 발생 (압전 효과)시키거나, 전압을 가함으로써 변형 (역압전 효과)시키는 디바이스이다.





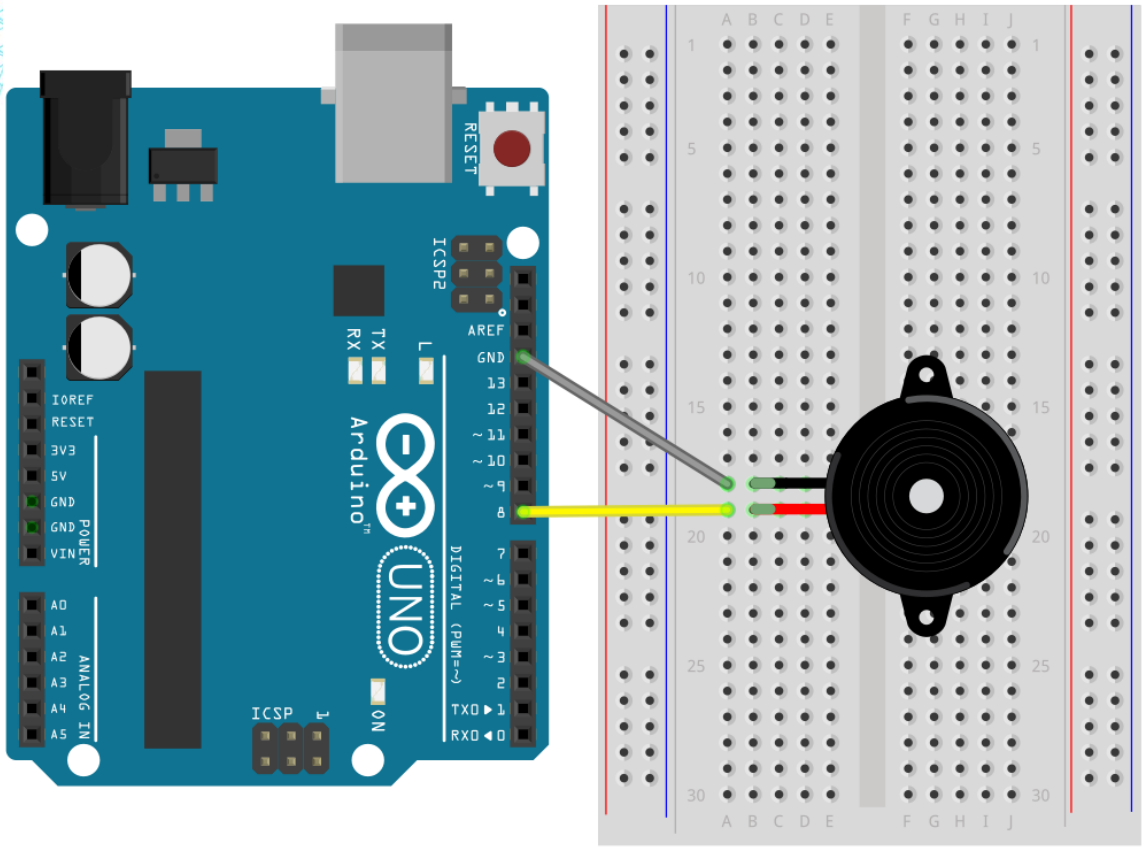


tone() 함수

- **tone(pin, freq [, duration])**
  - pin : 부저나 스피커가 연결된 디지털 핀 번호
  - freq : 주파수 (범위 : 31 ~ 65535), 31보다 작은 값을 넣어주면 잡음 출력
  - duration : (옵션) 음의 발생 시간 (밀리초)
- 핀에 지정된 주파수(50 % 듀티비)의 구형파를 발생시킨다. 지속 시간을 정할 수 있으며 따로 지정하지 않으면 noTone()을 호출 할 때까지 계속 된다.
- 핀은 피에조 부저 또는 스피커에 연결하여 톤을 재생할 수 있다.
- 한번에 하나의 톤 만 생성 할 수 있다. tone()이 이미 다른 핀에서 재생 중이면 tone() 호출은 아무 효과가 없다.
- Mega/ADK 이외의 보드에서 tone() 함수를 사용하면 3번과 11번 핀의 PWM 출력을 방해한다(즉, analogWrite 못쓴다는 뜻)
- tone() 함수는 기능을 수행하기 위해 하드웨어 PWM이 아닌 마이크로컨트롤러의 timer 사용한다. 따라서 디지털 출력을 할 수 있는 어떤 핀에든 할당 가능하다.

Passive buzzer

- 회로구성



- 소스코드

tone.ino

```
#define LED 13
#define BUZZER 8

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

void loop() {
  tone(BUZZER, 440); // 440Hz 소리내고
  delay(500);        // 0.5초
  noTone(BUZZER);    // 소리끄고
  delay(500);        // 0.5초

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

Buzzer (딩동댕동/땡)

dingdong1.ino

(index)	0	1	2	3
pitch	523	659	784	1046
length	250	250	250	500

```
#include "Button.h"

#define BUZZER  8
#define BTN_OK  7
#define BTN_NO 12

typedef struct {
  int pitch;
  int length;
} Note;
// 딩동댕동~ (도미솔도~)
Note noteOK[] = { {523,250}, {659,250},
     {784,250}, {1046,500} };
// 땡~ (미~~~~)
Note noteNO[] = { {659,1000} };

Button btnOK(INPUT_PULLUP, BTN_OK, 40);
Button btnNO(INPUT_PULLUP, BTN_NO, 40);

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

```
void loop() {
  btnOK.update();
  btnNO.update();

  if(btnOK.fell()) {
    int n=sizeof(noteOK)/sizeof(Note); // 12 / 4
    for(int i=0; i<n; i++) {
      tone(BUZZER, noteOK[i].pitch, noteOK[i].length);
      delay(noteOK[i].length);
    }
  }
  if(btnNO.fell()) {
    int n=sizeof(noteNO)/sizeof(Note); // 4 / 4
    for(int i=0; i<n; i++) {
      tone(BUZZER, noteNO[i].pitch, noteNO[i].length);
      delay(noteNO[i].length);
    }
  }
}
```

[문제점]
①악보정보를 읽기 위한 순회문 작성
②delay()함수 사용

MelodyPlayer 클래스 설계

MelodyPlayer.h

```
typedef struct {
  int pitch;
  int length;
} Note;

class MelodyPlayer {
  private:
    int   _buzzer;
    Note  *_notes;
    bool  _enabled;
    bool  _playing;
    bool  _autoReplay;
    int   _noteCount;
    int   _noteIndex;  // 연주해야 할 음표 idx
    int   _notePitch;  // 음표 높낮이
    unsigned long _noteLength;  // 음표 길이
    unsigned long _preTime;

  public:
    MelodyPlayer() { }
    MelodyPlayer(int pin) { init(pin); }
```

```
    void init(int pin) {
      pinMode(pin, OUTPUT);
      _buzzer = pin;
      _enabled = false;
      _playing = false;
      _noteLength = 0;
      _noteIndex = 0;
      _noteCount = 0;
      _autoReplay = false;
    }

    void setReplay(bool b) {
      _autoReplay = b;
    }

    void setNote(Note *notes, int len) {
      _notes = notes;
      _noteCount = len;
    }
```

```
  void play() {
    if(!_enabled) { _playing = false;  return; }

    unsigned long curTime = millis();
    // 아직 음표 길이만큼 연주 못했으면…
    if(curTime - _preTime < _noteLength) return;
    else noTone(_buzzer);

    if(_noteIndex < _noteCount) {
      _notePitch = _notes[_noteIndex].pitch;
      _noteLength = _notes[_noteIndex].length;

      if(_notePitch >= 31) {
        tone(_buzzer, _notePitch, _noteLength);
        _playing = true;
        _preTime = curTime;
      }

      _noteIndex++;
      if(_noteIndex >= _noteCount) {
        _noteIndex = 0;
        if(!_autoReplay) {
          _enabled = false;
        }
      }
    }
  }
```

```
    void startPlay() {
      _enabled = true;
      _noteIndex = 0;
      _preTime = 0;
      play();
    }

    void enable()    { _enabled = true;  }
    void disable()   { _enabled = false; }
    bool enabled()   { return _enabled;  }
    bool isPlaying() { return _playing;  }
};
```

Buzzer (MelodyPlayer 활용)

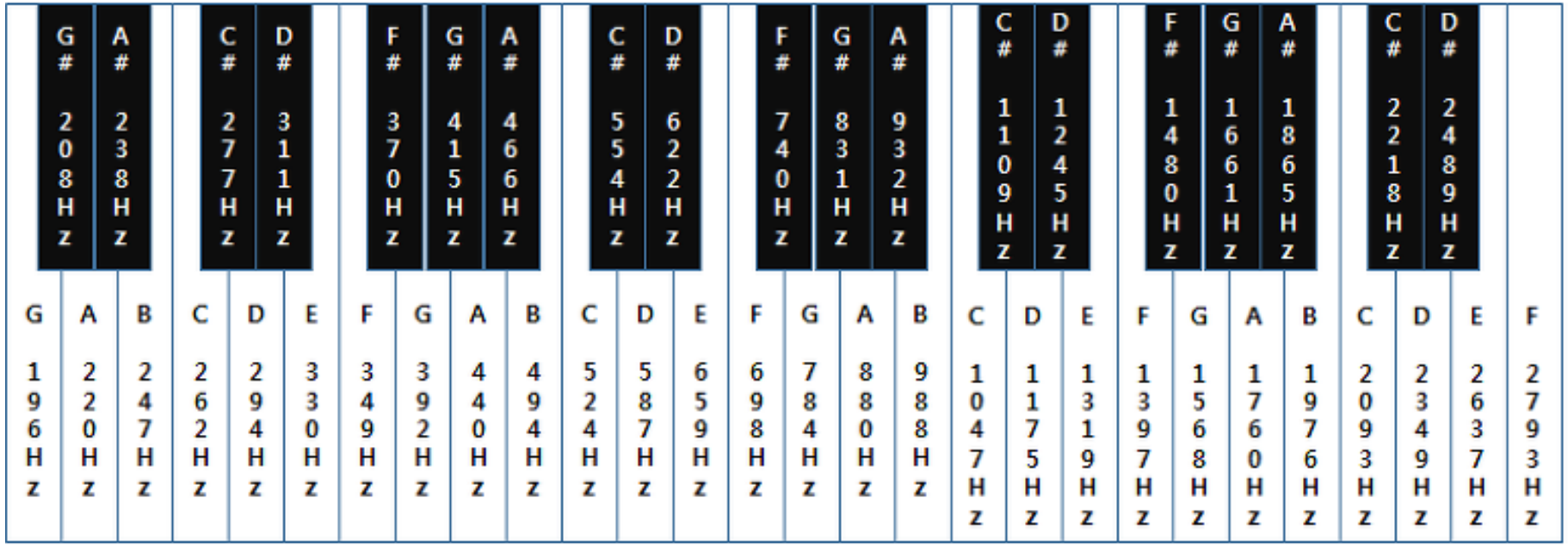
dingdong2.ino

```
#include "Button.h"
#include "MelodyPlayer.h"

#define BUZZER  8
#define BTN_OK  7
#define BTN_NO 12

// 딩동댕동~ (도미솔도~)
Note noteOK[] = { {523,250}, {659,250},
{784,250}, {1046,500} };
// 땡~ (미~~~~)
Note noteNO[] = { {659,1000} };

Button btnOK(INPUT_PULLUP, BTN_OK, 40);
Button btnNO(INPUT_PULLUP, BTN_NO, 40);

MelodyPlayer playerOK(BUZZER);
MelodyPlayer playerNO(BUZZER);
```

```
void setup() {
  Serial.begin(9600);
  playerOK.setNote(noteOK, 4);
  playerNO.setNote(noteNO, 1);
}

void loop() {
  btnOK.update();
  btnNO.update();

  if(btnOK.fell()) playerOK.start();
  if(btnNO.fell()) playerNO.start();

  playerOK.play();
  playerNO.play();
}
```

음계와 주파수

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



음계와 주파수

- 음계의 주파수 구성요소
  - 4옥타브 라 = 440Hz
  - 주파수가 2배인 음의 폭 = 옥타브
  - 총 12음계 존재 (라,라#,시,도,도#,레,레#,미,파,파#,솔,솔#)
  - 옥타브 구간의 주파수를 동일한 간격으로 12분할하여 12음계에 배당
  - b(플랫)은 반음 내림, #(샵)은 반음 올림으로 알았는데 실상은 그렇지 아니하고 모두 등비 간격임.

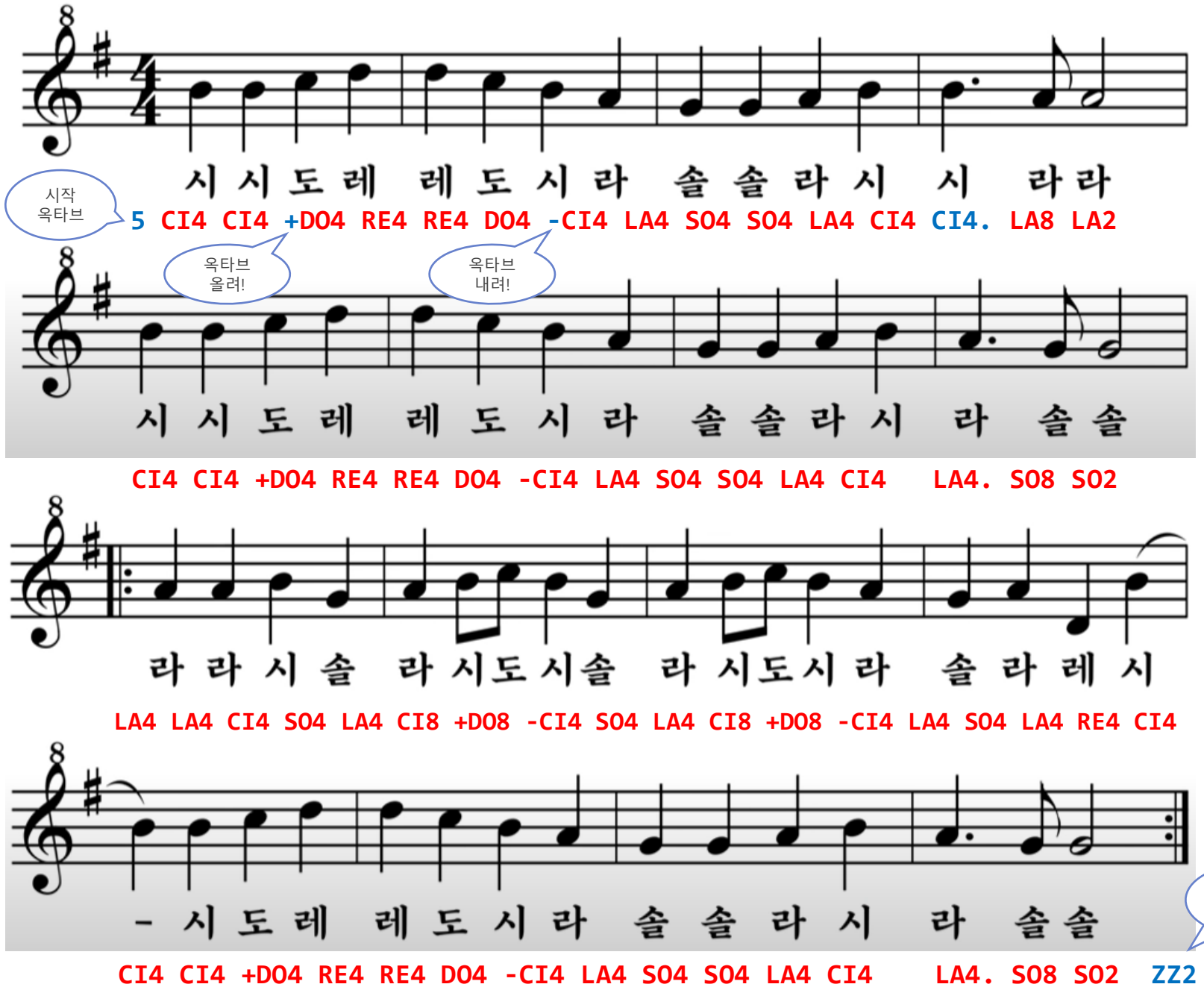
옥타브	음계	주파수	계산
4	라	440	$440 * 2^{0/12}$
4	라#	466	$440 * 2^{1/12}$
4	시	494	$440 * 2^{2/12}$
4	도	523	$440 * 2^{3/12}$
4	도#	554	$440 * 2^{4/12}$
4	레	587	$440 * 2^{5/12}$
4	레#	622	$440 * 2^{6/12}$
4	미	659	$440 * 2^{7/12}$
4	파	698	$440 * 2^{8/12}$
4	파#	740	$440 * 2^{9/12}$
4	솔	784	$440 * 2^{10/12}$
4	솔#	831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

악보를 문자열로 변환 [베토벤 환희의 송가]

[음계표]
도 DO
레 RE
미 MI
파 FA
솔 SO
라 LA
시CI
(쉼표) ZZ

도# DS
레# RS
파# FS
솔# SS
라# LS

레b RF
미b MF
솔b SF
라b LF
시b CF



[사용예]
도4분음표
DO4

레#2분음표쩜
RE#2.

4분음 쉼표
ZZ4

한 옥타브 올려서
미8분음표쩜
+MI8.

한 옥타브 내려서
시8분음표
-CI8

5옥타브라4분음표
5LA4
:
:

문자열로 기록된 악보를 해석하는 CNote 클래스

CNote.h

```h
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

class CNote {
  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;

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

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;
}
```

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

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;
}
```

```h
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; }

void CNote::replay(bool val) {
  _auto_replay = val;
}

// 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(솔플랫) 등

  표기법 예시
  3DO4 -> 3옥타브 도4분음표
*/
```

MusicPlayer 클래스 구현

```
#include "CNote.h"

class MusicPlayer {
  private:
    int   _buzzer;
    bool  _enabled;
    bool  _playing;
    bool  _verbose;
    CNote _cnote;
    unsigned long _noteLength;
    unsigned long _preTime;

  public:
    MusicPlayer() { }
    MusicPlayer(int pin) {
      init(pin);
    }
    MusicPlayer(int pin, char* score, int tempo) {
      init(pin);
      setScore(score);
      setTempo(tempo);
    }
    void init(int pin) {
      pinMode(pin, OUTPUT);
      _buzzer = pin;
      _noteLength = 0;
      _verbose = false;
    }

    void setScore(char* score) {
      _cnote.score(score);
    }

    void setTempo(int tempo) {
     _cnote.tempo(tempo);
    }

    void setReplay(bool b) {
      _cnote.replay(b);
    }

    void setVerbose(bool b) {
      _verbose = b;
    }
```

```
  void play() {
    if(!_enabled) { _playing = false;  return; }

    unsigned long curTime = millis();
    if(curTime - _preTime < _noteLength) return;
    else noTone(_buzzer);

    if(_cnote.next()) {
      int notePitch = _cnote.pitch();
      _noteLength = _cnote.length();

      if(notePitch >= 31) {
        tone(_buzzer, notePitch, _noteLength);
        _playing = true;
        _preTime = curTime;
      }

      if(_verbose && Serial) {
        Serial.print(_cnote.note());
        Serial.print(" ");
        Serial.print(_cnote.pitch(), DEC);
        Serial.print(" ");
        Serial.println(_cnote.length(), DEC);
      }
    }
  }
```

```
    void start() {
      _enabled = true;
      _preTime = 0;
      play();
    }

    void setEnable(bool b) {
      _enabled = b;
    }
    void enable()    { _enabled = false; }
    void disable()   { _enabled = true;  }
    bool enabled()   { return _enabled;  }
    bool isPlaying() { return _playing;  }
};
```

MusicPlayer 클래스를 이용한 연주

```
#include "Button.h"
#include "MusicPlayer.h"

#define BUZ  8
#define BTN 12

// 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";

const char *BUBBLE_BUBBLE =
  "5 CF8 LA8 SO8. FA16 LA8 SO8 FA8 MF8 SO8 FA8 MF16 RE16 ZZ8 \
   FA4. re16 do16 -cf8 +do8 re8 mf8 do8 re16 mf16 mf8 fa8 \
   fa8 so8 la16 so16 zz8 fa8 fa8 so8 la8 cf8 la8 so8. fa16 la8 so8 fa8 mf8 so8 \
   fa8 mf16 re16 zz8 fa4. re16 do16 -cf8 +do8 re8 mf8 do8 re16 mf16 mf8 fa8";

Button btnSwitch(INPUT_PULLUP, BTN, 40);
MusicPlayer bgmPlayer(BUZ);
```

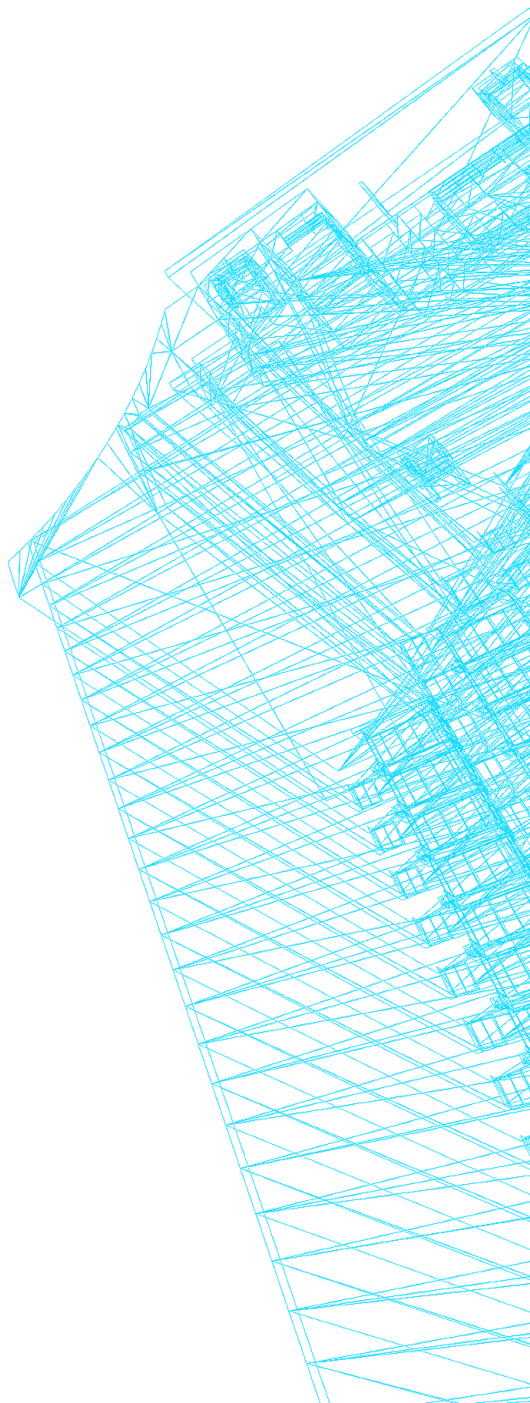
bgmPlay.ino

```
void setup() {
  Serial.begin(9600);
  bgmPlayer.setScore(BEETHOVEN_SCORES);
  bgmPlayer.setTempo(120);
  bgmPlayer.setVerbose(true);
  bgmPlayer.start();
}

void loop() {
  btnSwitch.update();

  if(btnSwitch.rose()) {
    bool e = bgmPlayer.enabled();
    bgmPlayer.setEnable(!e);
  }

  bgmPlayer.play();
}
```

(7 segmented led)

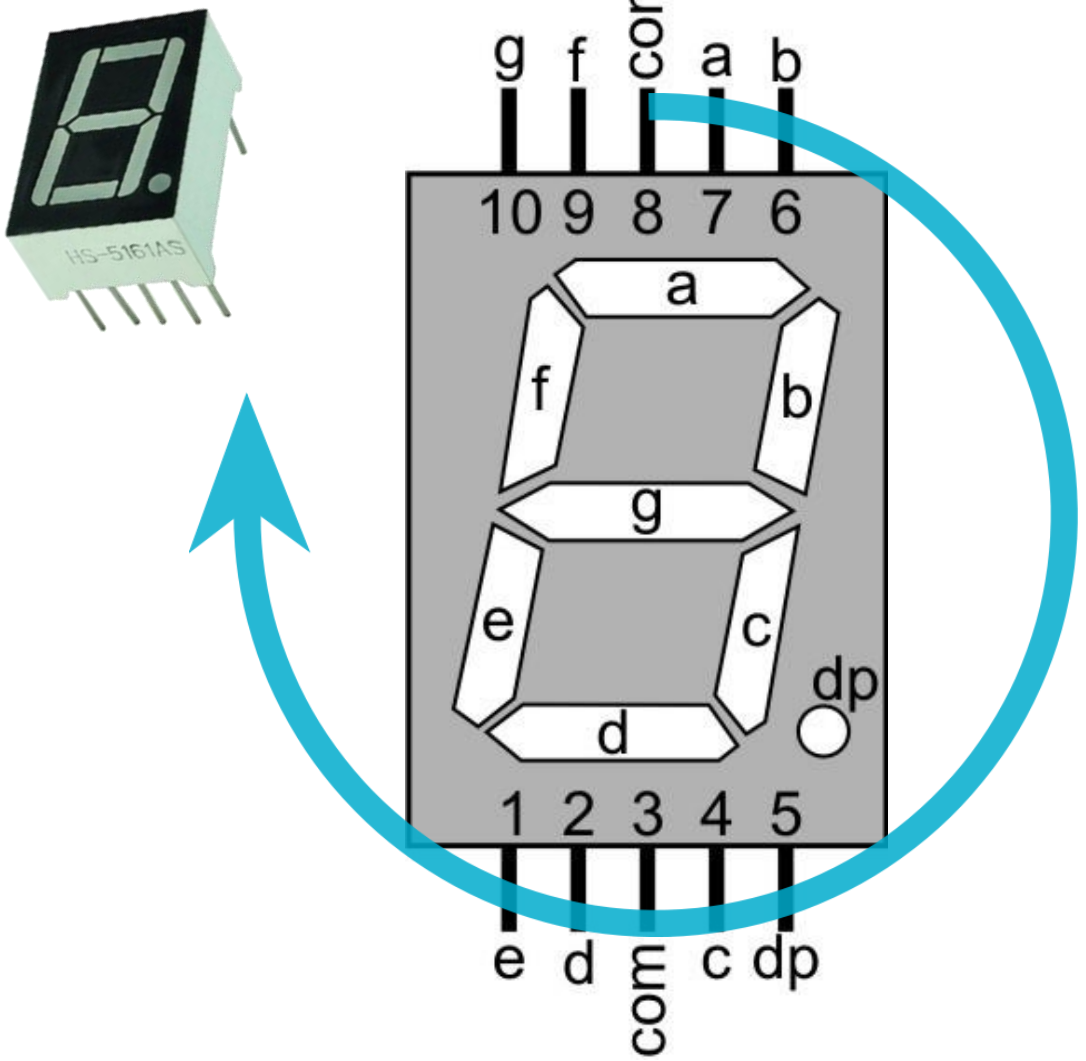
4. FND 제어하기

- FND 내부구조
- 숫자 & 문자 그리기
- 4-digits FND의 제어
- 74HC595 Shift Register 다루기
- FND 클래스 설계하기



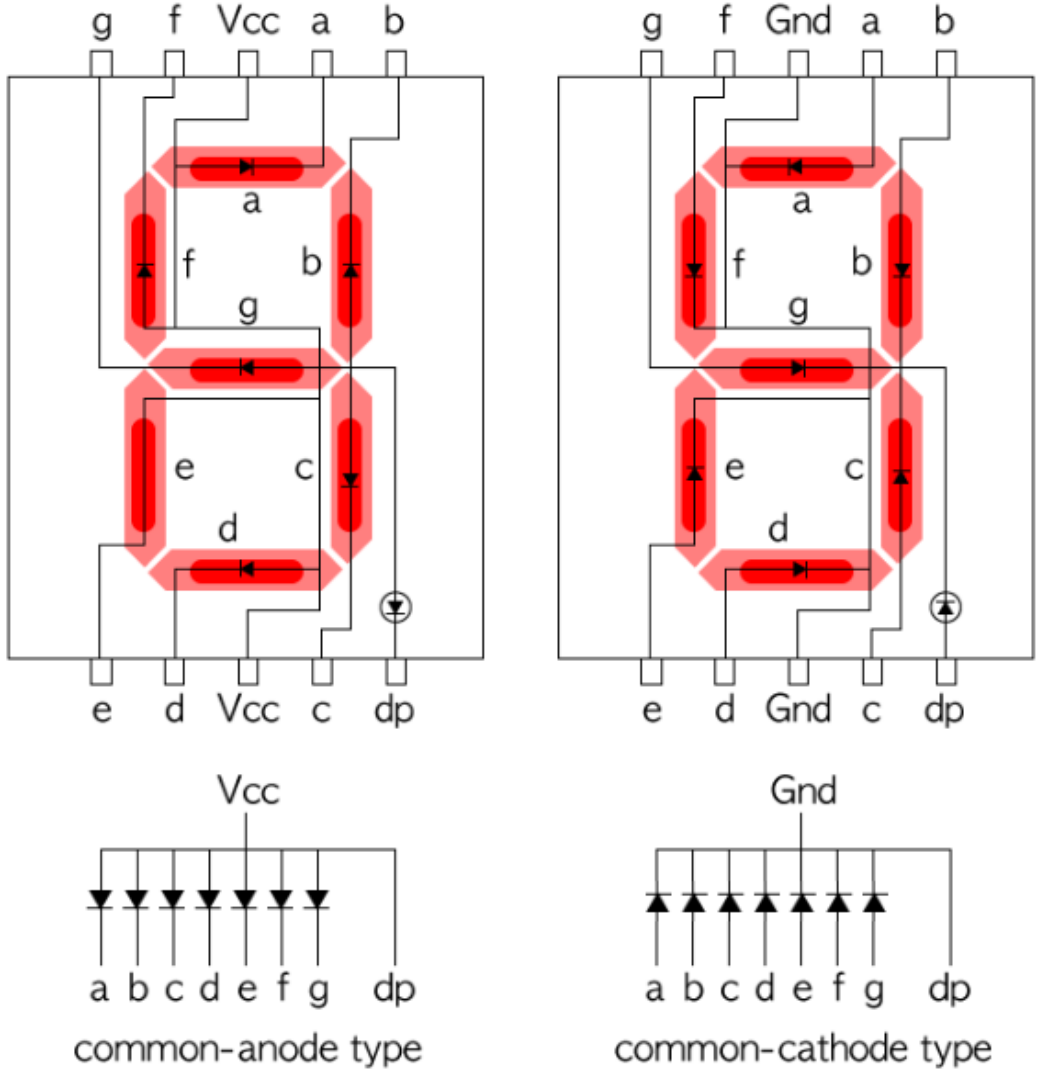
FND는…

- FND(Flexible Numeric Display)
  - Flexible Numeric Display(가변 숫자 표시기)의 약자로, 7-segment LED라고도 한다.
  - 주로 숫자를 표시하기 위해서 만들어진 LED의 조합
  - 수치 데이터를 알아보기 쉽게 전달 가능, 현황판 등에 많이 사용
  - a~g,dp에 해당하는 핀에 HIGH 또는 LOW를 인가하여 LED를 점등



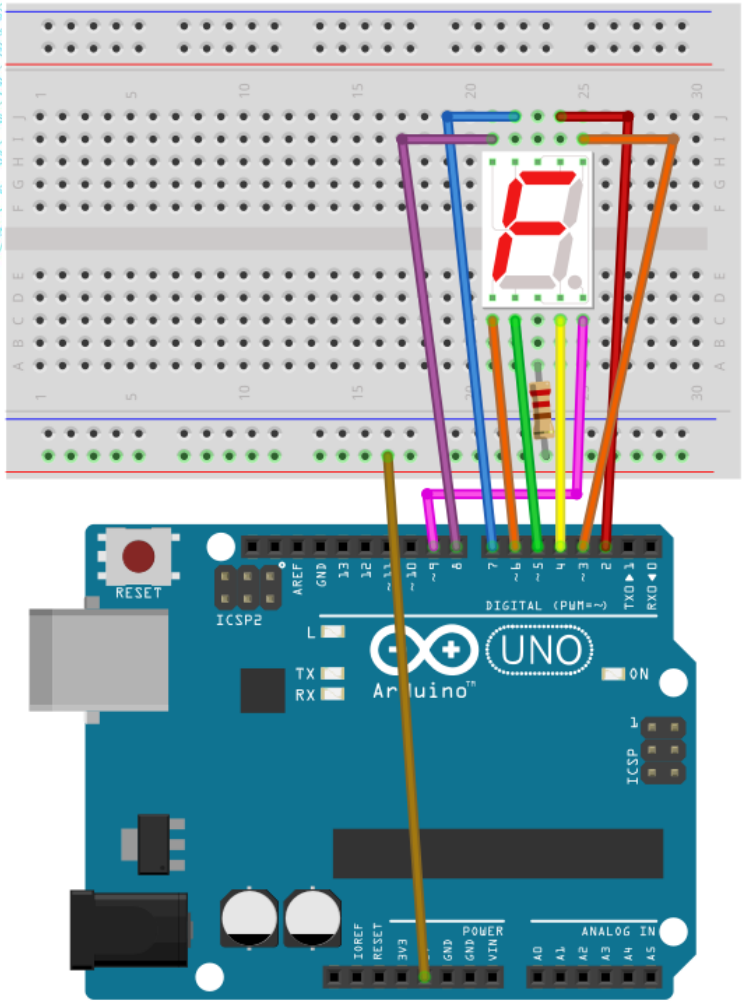
FND(7 Segment LED) 내부구조

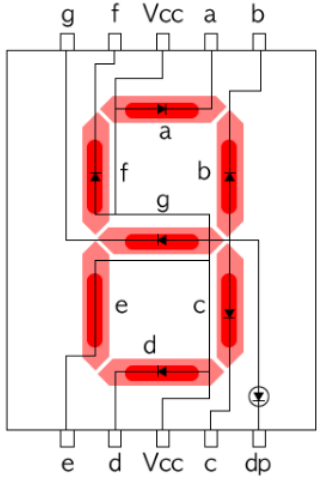
- 공통 양극(common-anode) type
  - 공통 Vcc
  - 전류가 Vcc에서 공급되어 MCU의 핀으로 흘러 들어가는 방식
  - MCU 핀이 LOW 일 때 켜짐
- 공통 음극(common-cathode) type
  - 공통 GND
  - 전류가 MCU의 핀에서 공급되어 GND 단자로 흘러 나가는 방식
  - MCU 핀이 HIGH 일 때 켜짐



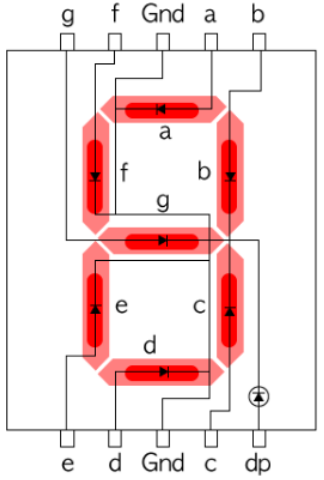
FND 회로연결

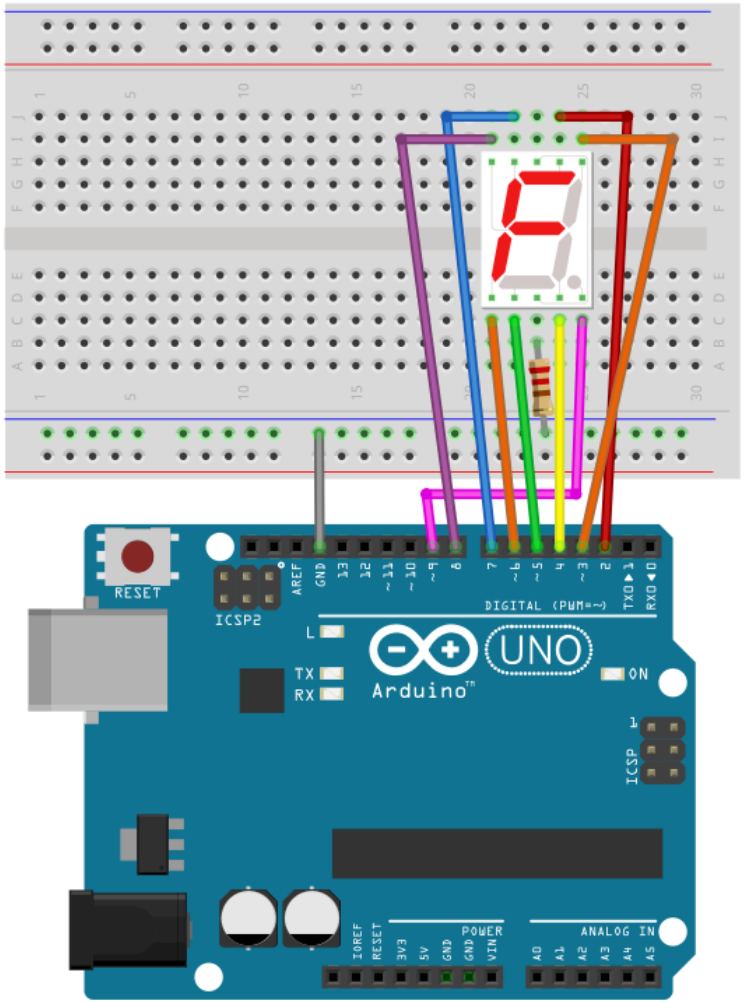
- 공통 양극 일 때,





- 공통 음극 일 때,





FND	아두이노	FND	아두이노
A	2	E	6
B	3	F	7
C	4	G	8
D	5	DP	9

둘 다 연결 핀 동일
Common 단자만 다름

FND에 숫자 출력하기

- 고찰
  - 조각난 LED를 조합하여 특정 글자를 만들려면 어떤 LED를 켜고 꺼야 하는가에 대한 정보가 필요
  - LED 세그먼트 개수는 (a, b, c, d, e, f, g, dp) 총 8개
  - 1 세그먼트마다 on 인지 off 인지 두 가지 상태만 저장하면 되므로 1비트로 충분
  - 8세그먼트 = 8bit = 1byte
  - 0 을 그리려면 a,b,c,d,e,f를 켜야 한다라는 정보를 단, 1바이트인 3F 로 표현

- FND_FONT

DISP	DP	g	f	e	d	c	b	a	HEX
	8	4	2	1	8	4	2	1	
	0	0	1	1	1	1	1	1	3F
	0	0	0	0	0	1	1	0	06
	0	1	0	1	1	0	1	1	5B
	0	1	0	0	1	1	1	1	4F
	0	1	1	0	0	1	1	0	66
	0	1	1	0	1	1	0	1	6D
	0	1	1	1	1	1	0	1	7D
	0	0	0	0	0	1	1	1	07
	0	1	1	1	1	1	1	1	7F
	0	1	1	0	0	1	1	1	67

FND에 숫자 출력하기

DISP	DP	g	f	e	d	c	b	a
	8	4	2	1	8	4	2	1
	0	0	1	1	1	1	1	1

fnd1_num.ino

```
// common cathode example
#define FND_A  2
#define FND_B  3
#define FND_C  4
#define FND_D  5
#define FND_E  6
#define FND_F  7
#define FND_G  8
#define FND_DP 9

byte FND_FONT[] = {
   0x3F, 0x06, 0x5B, 0x4F, 0x66,
   0x6D, 0x7D, 0x07, 0x7F, 0x67  };

bool dot = false;

void setup() {
    for(int i=FND_A; i<=FND_DP; i++)
        pinMode(i, OUTPUT);
}
```

```
void outputFND(int num, bool dot) {
    byte maskbit = B00000001;
    byte font = FND_FONT[num] | (dot<<7);

    for(int i=FND_A; i<=FND_DP; i++) {
        if((font & maskbit) != 0)
            digitalWrite(i, HIGH);
        else
            digitalWrite(i, LOW);

        maskbit = maskbit << 1;
    }
}

void loop() {
    for(int num=0; num<=9; num++) {
        outputFND(num, dot);
        delay(500);
    }
    dot = !dot;
}
```

(FONT)	0	0	1	1	1	1	1	1
(MASK) &	0	0	0	0	0	0	0	1
ON/OFF	0	0	0	0	0	0	0	1

(FONT)	0	0	1	1	1	1	1	1
(MASK) &	0	0	0	0	0	0	1	0
ON/OFF	0	0	0	0	0	0	1	0

(FONT)	0	0	1	1	1	1	1	1
(MASK) &	0	0	0	0	0	1	0	0
ON/OFF	0	0	0	0	0	1	0	0

(FONT)	0	0	1	1	1	1	1	1
(MASK) &	0	0	0	0	1	0	0	0
ON/OFF	0	0	0	0	1	0	0	0

(FONT)	0	0	1	1	1	1	1	1
(MASK) &	0	0	0	1	0	0	0	0
ON/OFF	0	0	0	1	0	0	0	0

(FONT)	0	0	1	1	1	1	1	1
(MASK) &	0	0	1	0	0	0	0	0
ON/OFF	0	0	1	0	0	0	0	0

(FONT)	0	0	1	1	1	1	1	1
(MASK) &	0	1	0	0	0	0	0	0
ON/OFF	0	0	0	0	0	0	0	0

(FONT)	0	0	1	1	1	1	1	1
(MASK) &	1	0	0	0	0	0	0	0
ON/OFF	0	0	0	0	0	0	0	0

common cathode 일 때 (active high)

DISP	DP	g	f	e	d	c	b	a	HEX
	8	4	2	1	8	4	2	1	
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	0	1	1	1	67
A	0	1	1	1	0	1	1	1	77
b	0	1	1	1	1	1	0	0	7C
C	0	0	1	1	1	0	0	1	39
d	0	1	0	1	1	1	1	0	5E
E	0	1	1	1	1	0	0	1	79
F	0	1	1	1	0	0	0	1	71

상호 변환은
bit not 연산자
~ 로 가능하다.

~(0x3F) = 0xC0

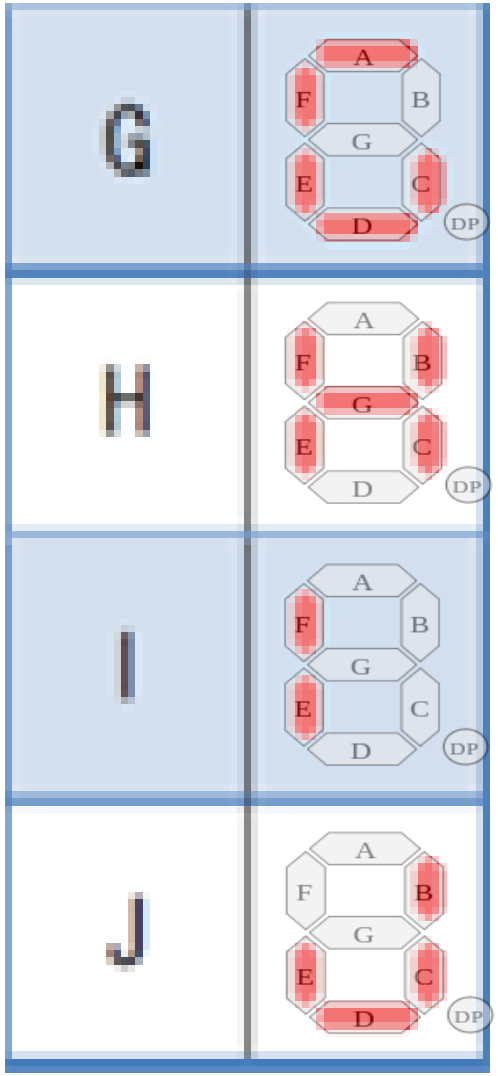
common anode 일 때 (active low)

DISP	DP	g	f	e	d	c	b	a	HEX
	8	4	2	1	8	4	2	1	
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	0	0	0	0	90
A	1	0	0	0	1	1	0	0	8C
b	1	0	1	0	1	1	1	1	BF
C	1	1	0	0	0	1	1	0	C6
d	1	0	1	0	0	0	0	1	A1
E	1	0	0	0	0	1	1	0	86
F	1	0	0	0	1	1	1	0	8E

FND_FONT 추가

- 2022 지역대회 (중) 7세그먼트의 원리와 활용

0	0	A	A	K	K	U	U
1	1	B	b	L	L	V	V
2	2	C	c	M	M	W	W
3	3	D	d	N	n	X	X
4	4	E	E	O	o	Y	Y
5	5	F	F	P	P	Z	Z
6	6	G	G	Q	q		
7	7	H	H	R	r		
8	8	I	I	S	S		
9	9	J	J	T	t		



DISP	DP	g	f	e	d	c	b	a	HEX
	8	4	2	1	8	4	2	1	
G	0	0	1	1	1	1	0	1	3D
H	0	1	1	1	0	1	1	0	76
I	0	0	1	1	0	0	0	0	30
J	0	0	0	1	1	1	1	0	1E
K	0	1	1	1	1	0	1	0	7A
L	0	0	1	1	1	0	0	0	38
M	0	1	0	1	0	1	0	1	55
N	0	1	0	1	0	1	0	0	54
O	0	1	0	1	1	1	0	0	5C
P	0	1	1	1	0	0	1	1	73
Q	0	1	1	0	0	1	1	1	67
R	0	1	0	1	0	0	0	0	50
S	0	1	1	0	1	1	0	1	6D
T	0	1	1	1	1	0	0	0	78
U	0	0	1	1	1	1	1	0	3E
V	0	1	1	1	1	1	1	0	7E
W	0	1	1	0	1	0	1	0	6A
X	0	0	1	1	0	1	1	0	36
Y	0	1	1	0	1	1	1	0	6E
Z	0	1	0	0	1	0	0	1	49

FND에 숫자&문자 출력하기

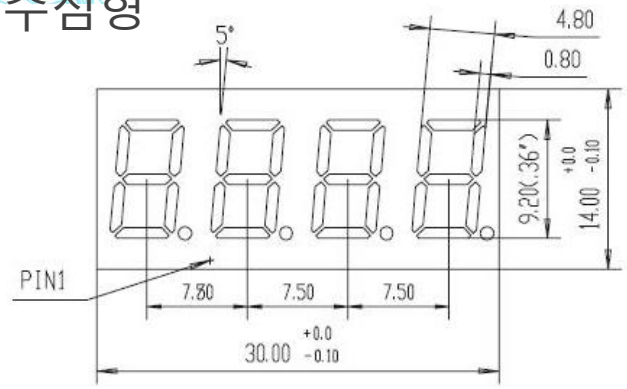
fnd1_num_char.ino

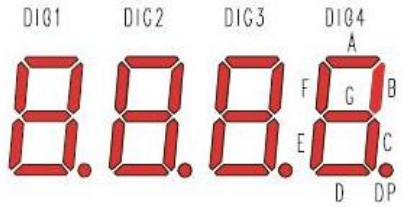
```
#define FND_A  2
#define FND_B  3
#define FND_C  4
#define FND_D  5
#define FND_E  6
#define FND_F  7
#define FND_G  8
#define FND_DP 9

unsigned char FND_FONT[] =
   // 0,    1,    2,    3,    4,    5,    6,    7,    8,    9,
  {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67,
   // A,    B,    C,    D,    E,    F,    G,    H,    I,    J,
   0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71, 0x3D, 0x76, 0x30, 0x1E,
   // K,    L,    M,    N,    O,    P,    Q,    R,    S,    T,
   0x7A, 0x38, 0x55, 0x54, 0x5C, 0x73, 0x67, 0x50, 0x6D, 0x78,
   // U,    V,    W,    X,    Y,    Z,   공백
   0x3E, 0x7E, 0x6A, 0x36, 0x6E, 0x49, 0x00 };

bool dot = false;

void setup() {
    for(int i=FND_A; i<=FND_DP; i++)
        pinMode(i, OUTPUT);
}
```

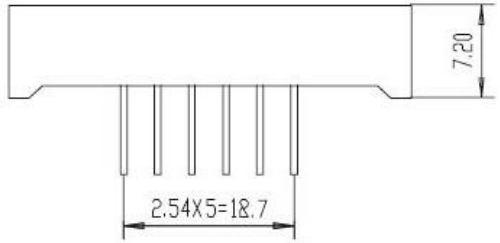
```
void outputFND(int num, bool dot) {
    byte maskbit = B00000001;
    byte font = FND_FONT[num] | (dot<<7);

    for(int i=FND_A; i<=FND_DP; i++) {
        if((font & maskbit) != 0)
            digitalWrite(i, HIGH);
        else
            digitalWrite(i, LOW);

        maskbit = maskbit << 1;
    }
}

void loop() {
    for(int num=0; num<=9; num++) {
        outputFND(num, dot);
        delay(500);
    }
    for(int ch='A'; ch<='Z'; ch++) {
        int idx = ch-'A'+10;
        outputFND(idx, dot);
        delay(500);
    }
    dot = !dot;
}
```

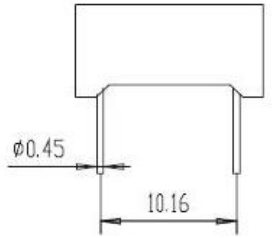
4-Digit FND

- 소수점형

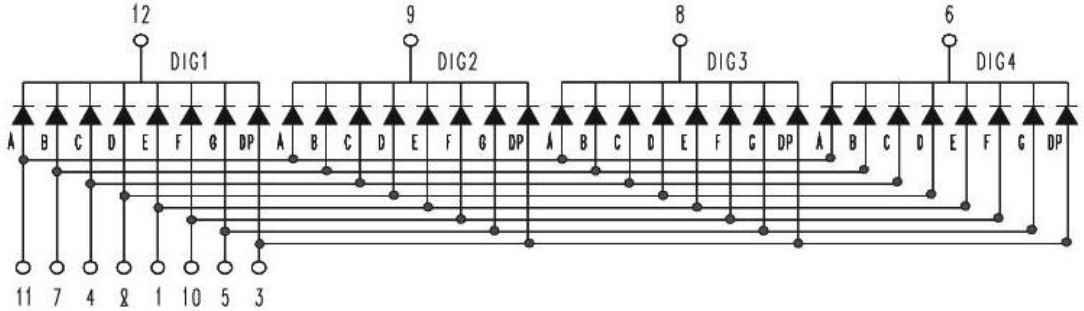




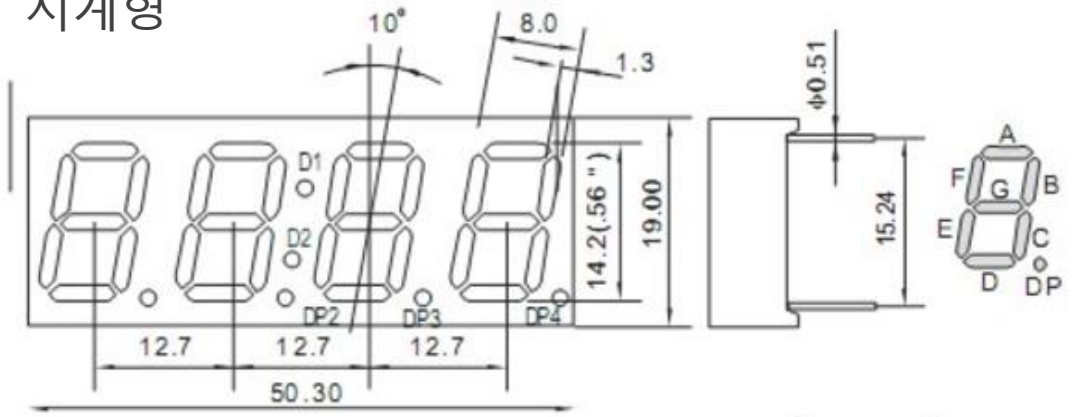


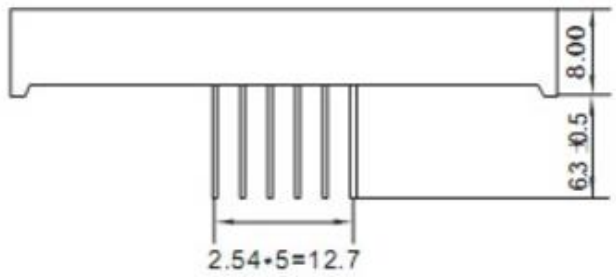


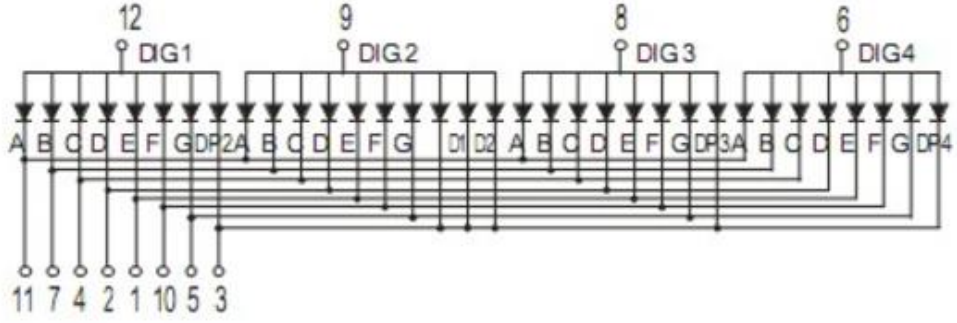
HS-3461A



- 시계형

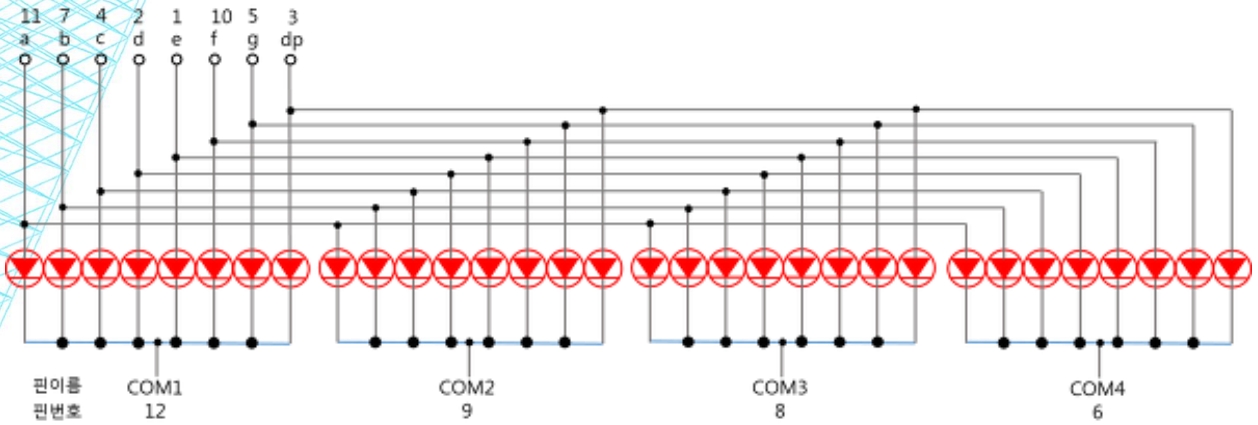




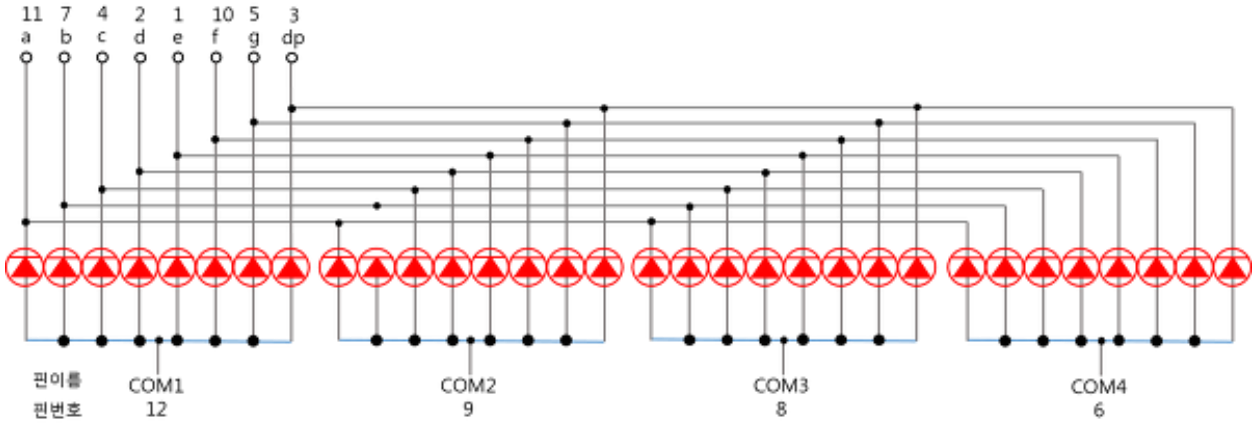


4-Digit FND

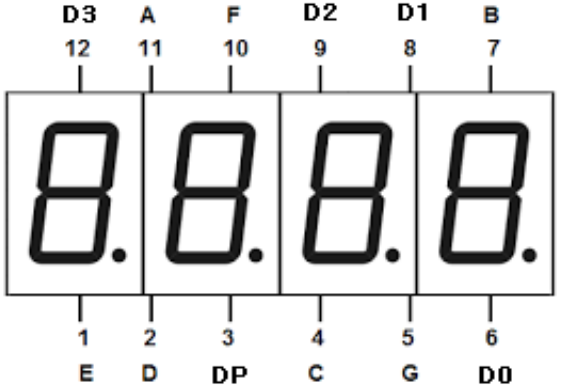
- 공통 음극(common cathode)

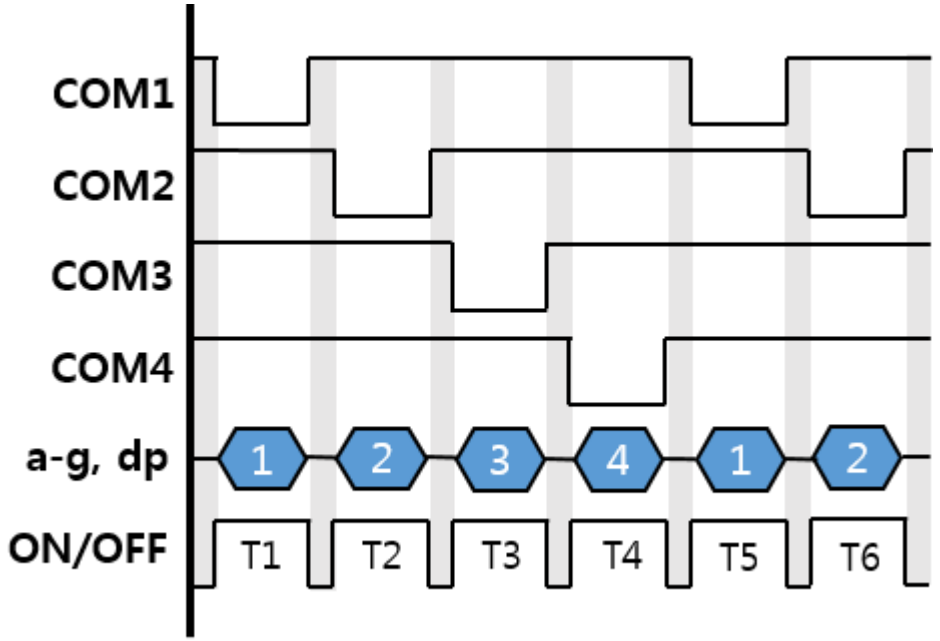


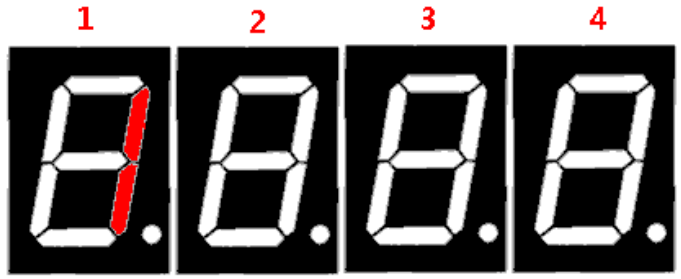
- 공통 양극(common anode)



- 제어 요령

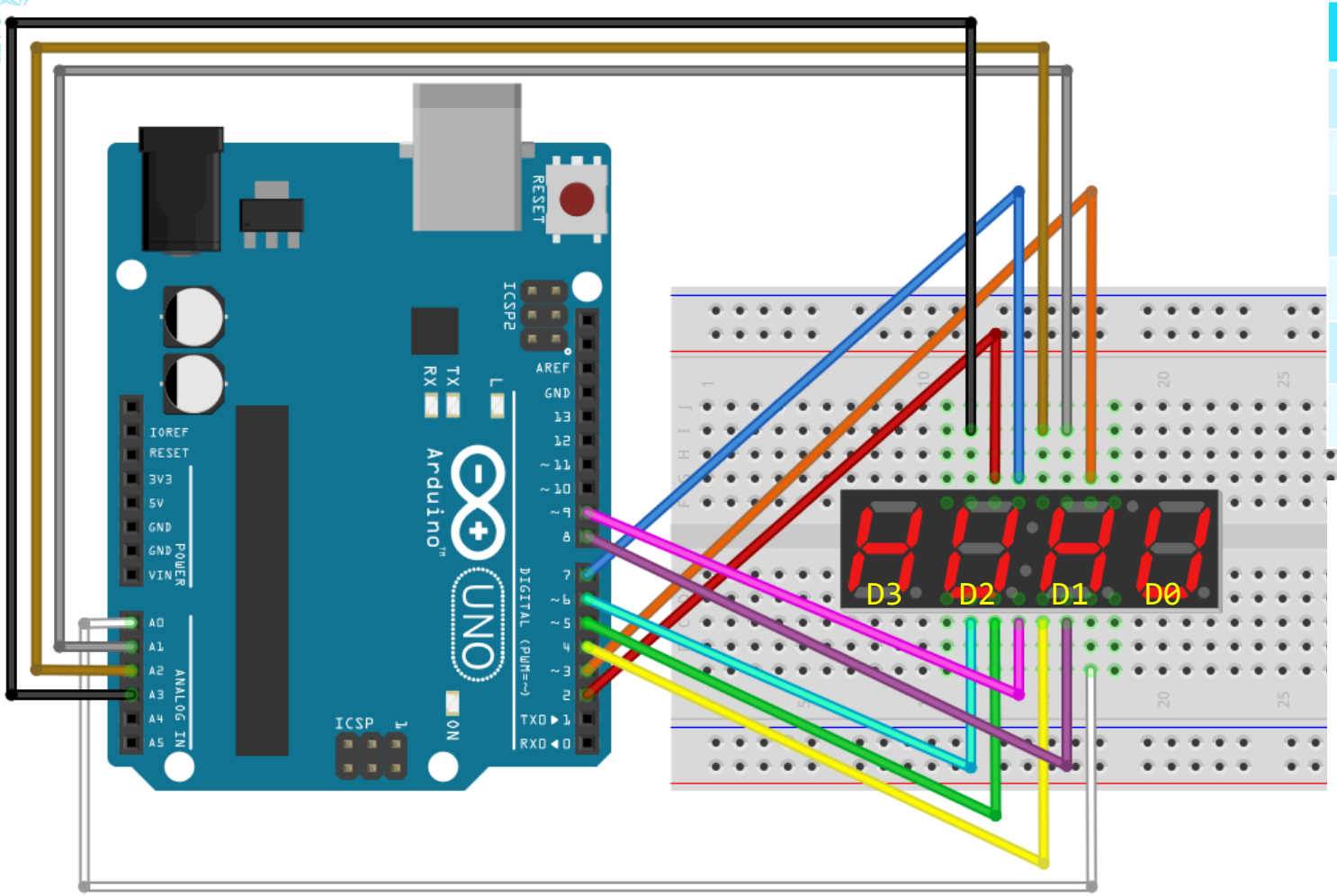






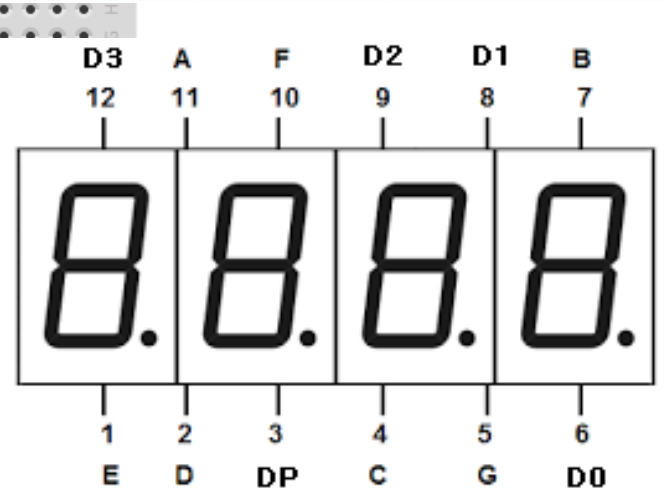
4-Digit FND

- 회로도



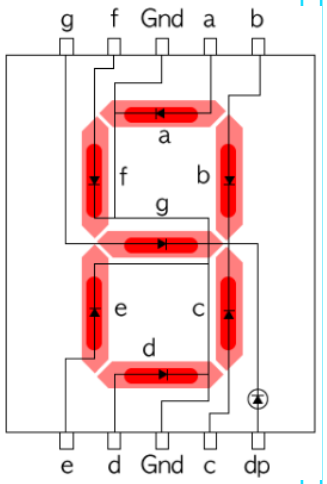
- 결선

FND	아두이노	FND	아두이노
A	2	G	8
B	3	DP	9
C	4	D0	A0
D	5	D1	A1
E	6	D2	A2
F	7	D3	A3

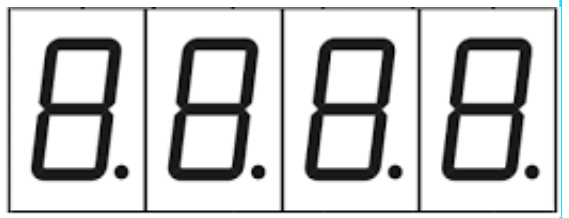


fnd4_output_cc.ino

```
// 4-digit FND for common cathode example
#define FND_D0 A0
#define FND_D1 A1
#define FND_D2 A2
#define FND_D3 A3

int FND_NUMBER;    // 출력할 숫자
int FND_PNTPOS=1; // 점을 찍을 위치 3210

void setup() {
    int i = 0;
    for(i=FND_A; i<=FND_DP; i++)
        pinMode(i, OUTPUT);
    for(i=FND_D0; i<=FND_D3; i++) {
        pinMode(i, OUTPUT);
        digitalWrite(i, HIGH);    // 모든 FND를 OFF
    }
}

unsigned long preTime = 0;
void loop() {
    unsigned long nowTime = millis()/10;
    static int count = 0;
    if(preTime != nowTime) {
      count++;
      preTime = nowTime;
    }
    outputNumber(count, FND_D2);
}
```

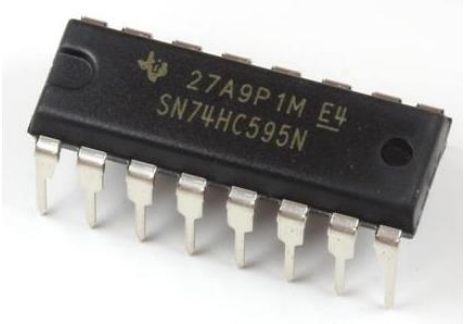


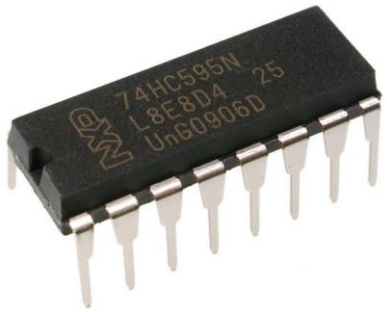

```
// outputFND()는 이전과 동일
void outputFND(int num, bool dot) {
    byte maskbit = B00000001;
    byte font = FND_FONT[num] | (dot<<7);

    for(int i=FND_A; i<=FND_DP; i++) {
        if((maskbit & font) != 0)
            digitalWrite(i, HIGH);
        else
            digitalWrite(i, LOW);

        maskbit = maskbit << 1;
    }
}

void outputNumber() {
    int number = FND_NUMBER;
    for(int p=FND_D0; p<=FND_D3; p++) {
        int digit = number % 10;
        outputFND(digit, p == FND_PNTPOS);
        digitalWrite(p, LOW);      // p번 FND ON
        delayMicroseconds(200);    // 0.2ms
        digitalWrite(p, HIGH);     // p번 FND OFF

        number = number /10;
        if(number <= 0)
           break;
    }
}
```

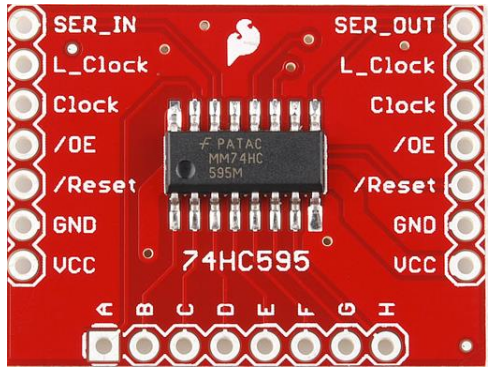


74HC595 Shift Register

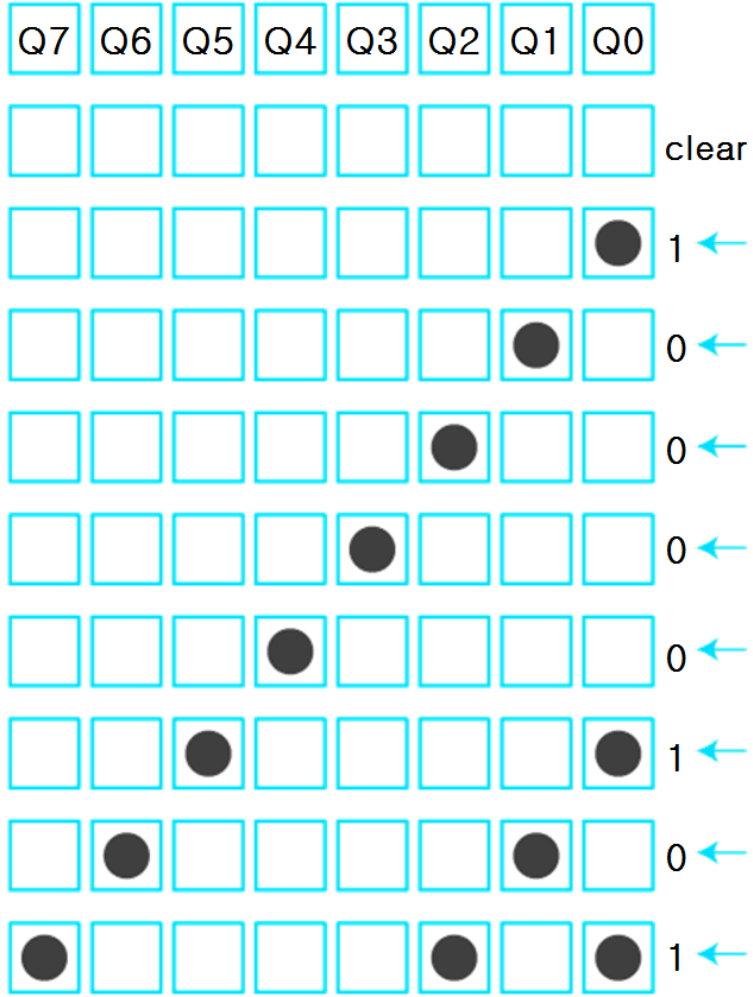
- 칩 외관





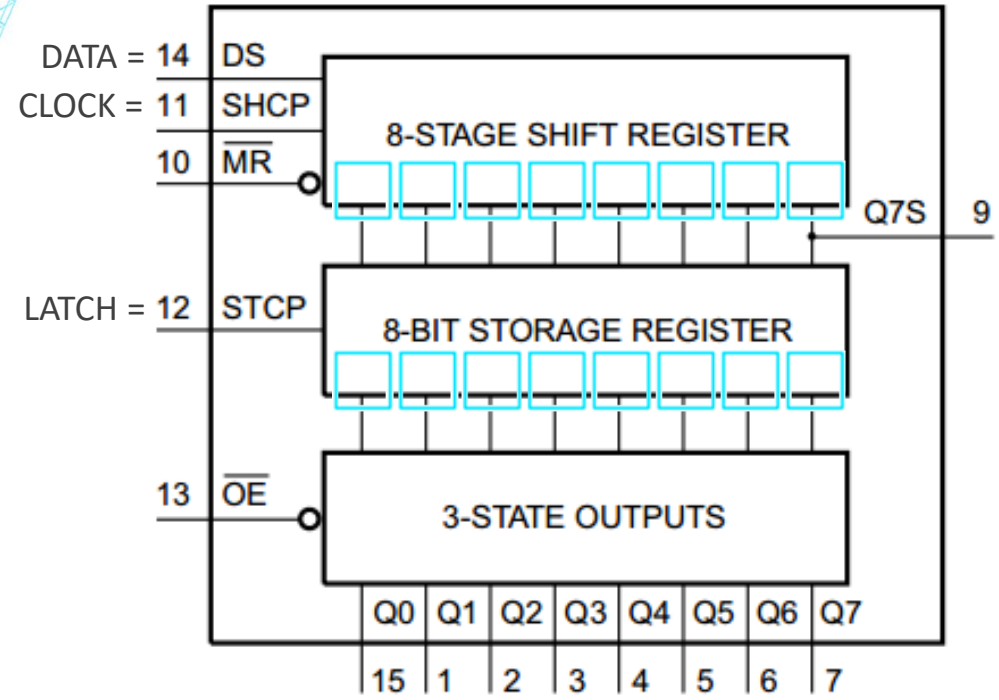


- Shift 동작의 이해



74HC595 Shift Register

Functional diagram

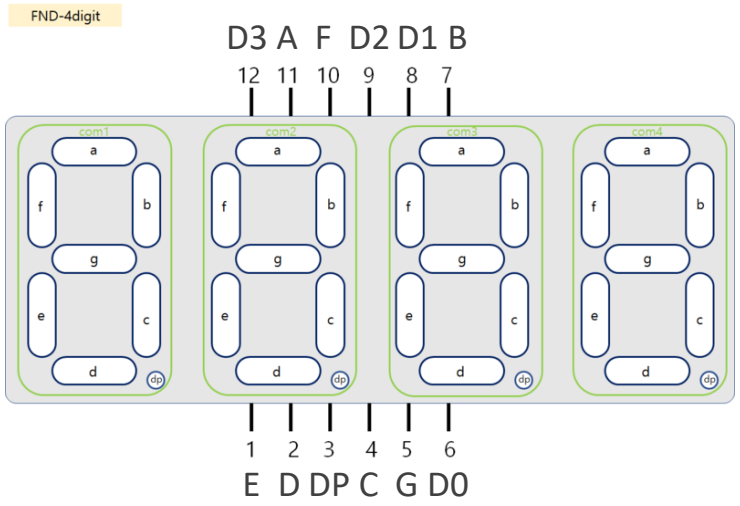


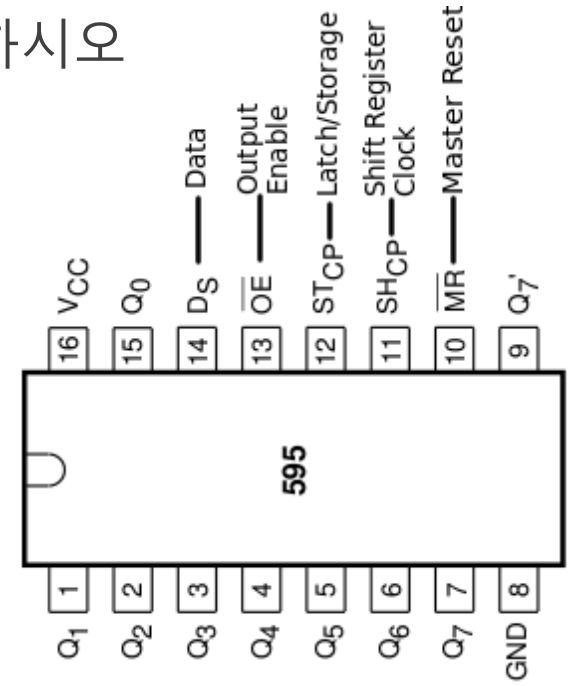
The 74HC595 is an 8-bit serial-in / parallel-out shift register with a storage register and 3-state outputs. Both the shift and storage register have separate clocks. The device features a serial input (DS) and a serial output (Q7S) to enable cascading and an asynchronous reset MR input. **A LOW on MR will reset the shift register. Data is shifted on the LOW-to-HIGH transitions of the SHCP(CLOCK) input. The data in the shift register is transferred to the storage register on a LOW-to-HIGH transition of the STCP input.** If both clocks are connected together, the shift register will always be one clock pulse ahead of the storage register. **Data in the storage register appears at the output whenever the output enable input (OE) is LOW.** A HIGH on OE causes the outputs to assume a high-impedance OFF-state. Operation of the OE input does not affect the state of the registers. Inputs include clamp diodes. This enables the use of current limiting resistors to interface inputs to voltages in excess of VCC.

data sheet: http://assets.nexperia.com/documents/data-sheet/74HC_HCT595.pdf

4-digit FND using 74HC595

- 실습과제
  - shift register를 이용하여 4-digit 7-segment led를 구동하는 회로를 제작하고,
  - 프로그램을 작성하시오





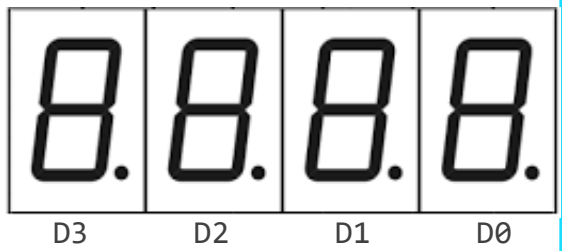
Pin	Symbol	연결
1	Q1	FND_B
2	Q2	FND_C
3	Q3	FND_D
4	Q4	FND_E
5	Q5	FND_F
6	Q6	FND_G
7	Q7	FND_DP
8	GND	GND
9	Q7`	다음 DS에 연결 (지금은 사용 안함)
10	$\overline{MR}$	LOW이면 리셋되어 버리므로 5V에 연결
11	SH_CP	CLOCK
12	ST_CP	LATCH
13	$\overline{OE}$	LOW 일 때 enable 되므로 GND에 연결
14	DS	DATA
15	Q0	FND_A
16	Vcc	5V

fnd4_sreg_cc.ino

fnd4_sreg_ca.ino

```
// 4digit FND using 74HC595 example
#define FND_D0 A0
#define FND_D1 A1
#define FND_D2 A2
#define FND_D3 A3
#define CLOCK  10
#define LATCH  11
#define DATA   12
//#define COMMON_CATHODE
#ifdef COMMON_CATHODE
const int D_ON = LOW, D_OFF = HIGH;
#else
const int D_ON = HIGH, D_OFF = LOW;
#endif

byte FND_FONT[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66,
   0x6D, 0x7D, 0x07, 0x7F, 0x67 };

unsigned long preTime = 0;

void setup() {
    for(int p=FND_D0; p<=FND_D3; p++) {
        pinMode(p, OUTPUT);
        digitalWrite(p, D_OFF); //@
    }
    pinMode(CLOCK, OUTPUT);
    pinMode(LATCH, OUTPUT);
    pinMode(DATA,  OUTPUT);
}
```

```
void outputFND(int num, bool dot) {
    byte font = (FND_FONT[num] | (dot<<7));
    if(D_ON==HIGH) font = ~font;
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, MSBFIRST, font);  // MSB부터 밀어넣기
    digitalWrite(LATCH, HIGH); // 출력에 적용
}

void outputNumber(int number, int dotpos) {
    for(int p=FND_D0; p<=FND_D3; p++) {
        int digit = number % 10;
        outputFND(digit, p == dotpos);
        digitalWrite(p, D_ON);     // p FND ON
        delayMicroseconds(200);    // 0.2ms
        digitalWrite(p, D_OFF);    // p FND OFF
        number = number /10;
        if(number <= 0)
           break;
    }
}
void loop() {
    unsigned long nowTime = millis()/10;
    static int count = 0;
    if(preTime != nowTime) {
      count++;
      preTime = nowTime;
    }
    outputNumber(count, FND_D2);
}
```



```
// Shift Register 미사용시

void outputFND(int num, bool dot) {
    int maskbit = 0x01;
    unsigned char font = FND_FONT[num] | (dot<<7);

    for(int i=FND_A; i<=FND_DP; i++) {
        if((maskbit & font) != 0)
            digitalWrite(i, HIGH);
        else
            digitalWrite(i, LOW);
        maskbit = maskbit << 1;
    }
}

void outputNumber(int number, int dotpos) {
    for(int p=FND_D0; p<=FND_D3; p++) {
        int digit = number % 10;
        outputFND(digit, p == dotpos);
        digitalWrite(p, LOW);      // p FND ON
        delayMicroseconds(1000);   // 1ms
        digitalWrite(p, HIGH);     // p FND OFF

        number = number /10;
        if(number <= 0)
           break;
    }
}
```

```
// Shift Register 사용시

void outputFND(int num, bool dot) {
    byte font = FND_FONT[num] | (dot<<7);
    digitalWrite(LATCH, LOW);
    // MSB부터 데이터 밀어넣기
    myShiftOut(DATA, CLOCK, MSBFIRST, font);
    digitalWrite(LATCH, HIGH); // 출력에 적용
}

void outputNumber(int number, int dotpos) {
    for(int p=FND_D0; p<=FND_D3; p++) {
        int digit = number % 10;
        outputFND(digit, p == dotpos);
        digitalWrite(p, LOW);      // p FND ON
        delayMicroseconds(1000);   // 1ms
        digitalWrite(p, HIGH);     // p FND OFF

        number = number /10;
        if(number <= 0)
           break;
    }
}
```

FND 클래스 설계

FND.h

```
#include <stdarg.h>
#include <ctype.h>

class FND {
  private:
    int _pinsLed[8];
    int _pinsDx[4];
    int _pinClock, _pinLatch, _pinData;
    int _outNumber;  // 출력할 숫자
    char _outBuffer[10];  // 출력할 내용
    int _posPoint;     // 소수점 위치
    int _numDigits;    // 자리수
    bool _enabled;
    bool _shiftRegMode;  // 시프트레지스터 사용여부
    bool ON;   // commonCathod이면 HIGH로 설정하시오.
    bool OFF;
    bool cmmON, cmmOFF;
    static unsigned char FND_FONT[];

  public:
    FND(bool levelON = HIGH) {
      _outNumber = 0;
      _posPoint = -1;  // -1: 소수점 숨김, 1: 소수1자리
      _numDigits = 1;  // 자리수
      _enabled = true;
      setLevelON(levelON);
    }
```

```
    void setLevelON(bool level) {
      ON = level;
      OFF = !ON;
      cmmON = OFF, cmmOFF = ON;
    }
    void allocLedPins(int a, int b, int c, int d,
                      int e, int f, int g, int dp) {
      pinsLed[0]=a, pinsLed[1]=b;
      pinsLed[2]=c, pinsLed[3]=d;
      pinsLed[4]=e, pinsLed[5]=f;
      pinsLed[6]=g, pinsLed[7]=dp;

      for(int i=0; i<8; i++) {
        pinMode(pinsLed[i], OUTPUT);
        digitalWrite(pinsLed[i], OFF);
      }
      shiftRegMode = false;
    }

    void allocRegisterPins(int clock, int latch, int data) {
      pinMode(clock, OUTPUT);
      pinMode(latch, OUTPUT);
      pinMode(data, OUTPUT);
      pinClock = clock;
      pinLatch = latch;
      pinData  = data;
      shiftRegMode = true;
    }
```

FND 클래스 설계 cont.

```
void allocDxPins(int cnt, ...) {
  va_list ap;
  va_start(ap, cnt);

  for(int i=0; i<cnt; i++) {
    pinsDx[i] = va_arg(ap, int);
    pinMode(pinsDx[i], OUTPUT);
    digitalWrite(pinsDx[i], cmmOFF); // <=
  }
  va_end(ap);
  numDigits = cnt;
}

void outputLedPins(int num, bool dot) {
  int maskbit = 1;
  byte font = FND_FONT[num] | (dot<<7);

  for(int i=0; i<8; i++) {
      if((maskbit & font) != 0)
        digitalWrite(pinsLed[i], ON);
      else
        digitalWrite(pinsLed[i], OFF);

      maskbit = maskbit << 1;
  }
}
```

```
void outputShiftRegPins(int num, bool dot=false) {
  byte font = FND_FONT[num] | (dot<<7);
  if(ON == LOW) font = ~font;

  digitalWrite(_pinLatch, LOW);
  shiftOut(_pinData, _pinClock, MSBFIRST, font);
  digitalWrite(_pinLatch, HIGH);
}

void outputFND(int num, bool dot=-1) {
  if(!_enabled) return;

  if(_shiftRegMode)
    outputShiftRegPins(num, dot);
  else
    outputLedPins(num, dot);
}

// pinDx[d] 핀에 전류가 흐르도록 설정함.
void onDx(int d) {
  digitalWrite(pinsDx[d], cmmON);
}

// pinDx[d] 핀에 전류가 흐르지 않도록 설정함.
void offDx(int d) {
  digitalWrite(pinsDx[d], cmmOFF);
}
```

FND 클래스 설계 cont.

```
int getCharFontIndex(char c) {
  int pos;
  if(isdigit(c))
    pos = c - '0';
  else if('A'<=c && c<='Z' || 'a'<=c && c<='z')
    pos = toupper(c) - 'A' + 10;
  else if(c == '-')
    pos = 36; // 마이너스
  else //출력할 내용이 숫자도 문자도 아니면,
    pos = 37; // 마지막 공백 문자
  return pos;
}

void outputFND(char ch, bool dot=false) {
  if(!_enabled) return;

  int font_idx = getCharFontIndex(ch);
  if(_shiftRegMode)
    outputShiftRegPins(font_idx, dot);
  else
    outputLedPins(font_idx, dot);
}

void outputNumber(int number, int pos=-1) {
  _outNumber = number;
  itoa(number, _outBuffer, 10);
  outputString(pos);
}
```

```
void outputCharAt(int pos, char font_idx, bool dot=false) {
  for(int p=0; p<_numDigits; p++)
    digitalWrite(_pinsDx[p], cmmOFF);    // p FND OFF

  if(_shiftRegMode)
    outputShiftRegPins(font_idx, dot);
  else
    outputLedPins(font_idx, dot);

  digitalWrite(_pinsDx[pos], cmmON);     // d FND ON
}

//문자열의 마지막 4글자를 가장 오른쪽부터 4개 출력
void outputString(char* strOutput, int dotpos=-1) {
  if(!_enabled) return;

  int idx = strlen(strOutput)-1;  //문자열의 마지막
  for(int d=0; d<min(4,_numDigits); d++, idx--) {
    if(idx < 0) return;
    outputFND(strOutput[idx], d==dotpos);
    digitalWrite(_pinsDx[d], cmmON);   // d FND ON
    delayMicroseconds(200);            // 0.2ms
    digitalWrite(_pinsDx[d], cmmOFF);  // d FND OFF
  }
}
```

FND 클래스 설계 cont.

```
    void outputString(int dotpos=-1) {
      outputString(_outBuffer, dotpos);
    }

    void enable()    { _enabled = true;  }
    void disable()   {
      _enabled = false;
      for(int d=0; d<min(4,_numDigits); d++)
        digitalWrite(_pinsDx[d], cmmOFF);
    }
    bool enabled()   { return _enabled;  }
};

// Active HIGH 일 때 폰트임.
// FND가 common annode 인 경우 아래 데이터를 비트 NOT하여 사용할 것.
unsigned char FND::FND_FONT[] =
   // 0,    1,    2,    3,    4,    5,    6,    7,    8,    9,
  {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67,
   // A,    B,    C,    D,    E,    F,    G,    H,    I,    J,
   0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71, 0x3D, 0x76, 0x30, 0x1E,
   // K,    L,    M,    N,    O,    P,    Q,    R,    S,    T,
   0x7A, 0x38, 0x55, 0x54, 0x5C, 0x73, 0x67, 0x50, 0x6D, 0x78,
   // U,    V,    W,    X,    Y,    Z,   - ,  공백
   0x3E, 0x7E, 0x6A, 0x36, 0x6E, 0x49, 0x40, 0x00 };
```

FND 클래스 활용

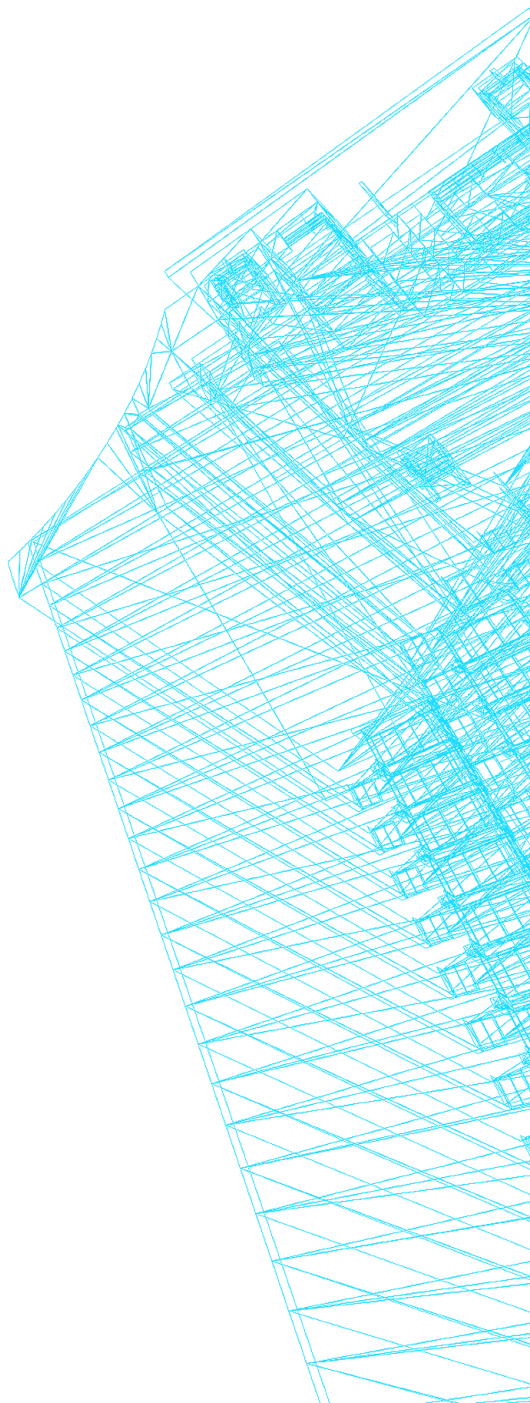
lib_fnd4_fnd1_together.ino

```
#include "FND.h"

// 1자리 FND는 2,3,4,5,6,7,8,9번 핀 사용
FND fnd1;
// 4자리 FND는 Shift Register와 함께..
// 10(CLOCK), 11(LATCH), 12(DATA),
// 공통단자 A0,A1,A2,A3핀에 연결
FND fnd4;

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

  fnd1.allocLedPins(2,3,4,5,6,7,8,9);
  fnd1.setLevelON(HIGH);  // common cathod

  fnd4.allocRegisterPins(10,11,12);
  fnd4.allocDxPins(4,A0,A1,A2,A3);
  fnd4.setLevelON(LOW);  // common anode
}
```

```
int preSec;
char str[] = "ABCD";
void loop() {
  int nowSec = millis()/1000;
  // FND1에는 숫자출력
  fnd1.outputNumber(nowSec%10, -1);
  // FND4에는 문자열출력
  fnd4.outputString(str);

  if(preSec != nowSec) {  // 초가 바뀔 때마다
    for(int i=0; i<4; i++)
      str[i]++;

    preSec = nowSec;
    if(str[0]=='X')
      strcpy(str, "ABCD");
  }
}
```

5. 기출문제 풀이

- 2018 도대회 (고) 알람시계
- 2019 도대회 (중) 추억의 뽑기 게임
- 2022 도대회 (고) 3·6·9게임 연습기계



2018 도대회 (고) 알람시계

철수는 시계의 기능을 아두이노로 구현하려고 한다. 철수가 한 자리에 앉아서 얼마나 집중했는지 시간을 측정하려고 한다. 이를 위해 앉았을 때 버튼을 누르면, 시간이 흐르기 시작하고, 자리에서 일어나면서 버튼을 누르면 시간이 멈추고 저장된다. 이런 과정을 최대 10회까지 저장하려고 한다.

◈ 필수해결 요소

① 버튼을 설치하고, 버튼을 누르면 시간이 0에서부터 시작된다.
② LCD에는 버튼을 누른 시점부터 시간의 변화를 초단위로 출력한다.
③ 다시 버튼을 눌렀을 때, 시간이 멈추고 현재 공부한 시간이 출력되며, 현재까지의 공부한 시간 중 몇 번째로 많은지 출력하게 한다.
④ 최대 10개의 시간이 저장되게 만든다.
⑤ 종료 버튼을 추가하고 종료를 한 경우에는 현재까지의 공부한 시간 중 최저 시간과 최고 시간이 출력되며, 평균의 시간을 출력한다.
⑥ 다시 한번 종료 버튼을 누르면, 현재까지 공부한 시간 중 긴 시간부터 차례대로 출력하도록 만든다.

2018 도대회 (고) 알람시계

2018-DO-HS%20StopWatch.ino

```
#include "IntTimer1.h"
#include "Button.h"
#include "FND.h"

#define  STUDY_NUM_MAX  10
#define  BUZ  8

FND fnd4(HIGH);  // common cathode
//FND fnd4(LOW);  // common anode

// start/stop toggle button
Button btnToggle(INPUT_PULLUP, 2, 40);
Button btnStore(INPUT_PULLUP, 3, 40);

unsigned long TIME;
int tmrIdTimeTick;
unsigned int studyTime[STUDY_NUM_MAX];
unsigned int studyNum = 0;  // 공부시간 저장횟수
unsigned int studyTimeRank[STUDY_NUM_MAX];
```

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

  fnd4.allocDxPins(4,A0,A1,A2,A3);
  fnd4.allocRegisterPins(10,11,12);
  tmrIdTimeTick = IntTimer1::add(timeTick, 100);
  IntTimer1::add(checkButton, 1);
  IntTimer1::start();
}

void timeTick() { // 0.1초 마다 실행됨
  TIME++;          // 시간변수 증가
  Serial.println(TIME/10.0); // _._ 형태로 출력
}

void checkButton() {
  btnToggle.update();
  btnStore.update();
}
```

2018 도대회 (고) 알람시계

```
void loop() {
  fnd4.outputNumber(TIME, 1);

  if(btnToggle.fell()) {  // 버튼이 눌리는 순간
    if(IntTimer1::enabled(tmrIdTimeTick))
      IntTimer1::disable(tmrIdTimeTick);
    else
      IntTimer1::enable(tmrIdTimeTick);

    tone(BUZ, 880, 100);
    delay(100);
    tone(BUZ, 440, 100);
  }

  if(btnStore.fell()) {
    // 시간이 멈춰 있을 때만 작동하도록...
    if(!IntTimer1::enabled(tmrIdTimeTick)) {
      studyTime[studyNum] = TIME;
      outputRank();
      studyNum++;
      if(studyNum >= STUDY_NUM_MAX)
        studyNum = 0;

      tone(BUZ, 440, 100);
      delay(100);
      tone(BUZ, 880, 100);
      TIME = 0;
    }
  }
}

void filloutRank() {
  for(int i=0; i<STUDY_NUM_MAX; i++)
    studyTimeRank[i]=1; // 일단 1등이라고 가정

  for(int i=0; i<STUDY_NUM_MAX; i++) { // i의 순위 결정
    for(int j=0; j<STUDY_NUM_MAX; j++) // 모든 j와 비교하여
      if(studyTime[i]<studyTime[j]) // j가 더 공부시간이 많으면,
        studyTimeRank[i]++; // i의 순위 뒤로 밀림
  }
}

void outputRank() {  // 순위 출력
  filloutRank();
  for(int i=0; i<STUDY_NUM_MAX; i++) {
    Serial.print(i+1);
    Serial.print(": ");
    Serial.println(studyTime[i]/10.0);
  }
  Serial.print(studyTimeRank[studyNum]);
  Serial.println("번째로 공부를 오래 하였습니다.");
}
```

2019 도대회 (중) 추억의 뽑기 게임

◈ 필수해결 요소

① 가위 바위 보 대결 시스템. ex: 시리얼 모니터에 가위, 바위, 보(1,2,3//a,b,c등으로 대체 가능)를 입력하면 승 또는 패를 알려주기(LED, 소리 등을 활용)

② 가위 바위 보 대결에서 이겼을 경우 랜덤하게 1~9까지의 숫자가 나오도록 하기(시리얼 모니터 활용), 졌을 경우 게임 종료

③ 게임이 종료되면 메달 획득 개수를 기준으로 랭킹 나오기(시리얼 모니터 활용)

④ 랭킹이 나오고 나면 다음 사람도 게임에 참여할 수 있도록 처음 상태로 되돌린다.(메달 획득 랭킹은 유지)

◈ 창의적 문제 해결

① 가위, 바위, 보를 입력하는 시스템을 버튼으로 직접 구현한다.

② 1~9까지의 숫자가 FND를 통해서 나오도록 표현한다.

③ 메달의 보상이 9까지가 아니라 20까지 표현되도록 FND를 구성한다.

④ 적절한 상황에 재미를 더할 수 있는 음향 효과를 추가한다.

2019 도대회 (중) 추억의 뽑기 게임

2019-DO-HS%20PickGame.ino

```
#include "MillisTimer.h"
#include "Button.h"
#include "FND.h"
#include "MelodyPlayer.h"

#define BTN_C 2
#define BTN_R 3
#define BTN_P 7
#define BUZ   8
enum SRP { NOTYET=0, SCI=1, ROC=2, PAP=3 };
char *strSRP[]={" ", "가위","바위","보"};

Note noteWin[] = { {523,250}, {659,250},
{784,250} };  // 딩동댕~ (도미솔~)
Note noteLose[] = { {659,750} };  // 땡~ (미~~~~)
Note noteTick[] = { {440, 80} };  // 띠(아주짧음)
MelodyPlayer mPlayer(BUZ);

FND fnd4(LOW);  // common anode mode
Button btnC(INPUT_PULLUP, BTN_C, 40); //가위
Button btnR(INPUT_PULLUP, BTN_R, 40); //바위
Button btnP(INPUT_PULLUP, BTN_P, 40); //보
```

```
float TIME = 0;
void tick() {
  if((int)(TIME*10) % 10==0) // 매 1초마다 시간 출력
    Serial.println((int)TIME);

  TIME += 0.1;
}

void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));

  fnd4.allocDxPins(4,A0,A1,A2,A3);
  fnd4.allocRegisterPins(10,11,12);
  MillisTimer::add(tick, 100);
}

char getCRPcode(SRP num) {
  if(num == SCI) return 'C';
  else if(num == ROC) return 'R';
  else if(num == PAP) return 'P';
  else return ' ';
}
```

2019 도대회 (중) 추억의 뽑기 게임

```
void loop() {
  TIME = 0.1;
  int  computerPick = NOTYET;
  int  gamerPick = NOTYET;
  bool isWin = false;  // 승리했는가?
  int  numMedal;

  // 카운트 다운 3.2.1.
  while(TIME <= 3) {
    MillisTimer::run();
    int outNum = (int)(4-TIME)*1111;
    fnd4.outputNumber(outNum);
  }

  // 버튼 눌러 가위바위보 내기
  fnd4.disable();
  computerPick = random(3)+1;
  while(TIME<=4 && gamerPick==NOTYET) {
    MillisTimer::run();
    btnC.update();
    btnR.update();
    btnP.update();

    if(btnC.fell()) gamerPick = SCI;
    if(btnR.fell()) gamerPick = ROC;
    if(btnP.fell()) gamerPick = PAP;
  }
```

```
  // 컴퓨터:사람 승부 출력
  char str[]="    ";
  str[0]=getCRPcode(computerPick);
  str[3]=getCRPcode(gamerPick);
  Serial.print("컴| ");
  Serial.print(strSRP[computerPick]);
  Serial.print(" : 나| ");
  Serial.println(strSRP[gamerPick]);
  fnd4.enable();

  if(gamerPick==SCI && computerPick==PAP ||
     gamerPick==ROC && computerPick==SCI ||
     gamerPick==PAP && computerPick==ROC)
    isWin = true;

  if(isWin) {
      mPlayer.setNote(noteWin, 3);
      mPlayer.start();
  }
  else {
      mPlayer.setNote(noteLose, 1);
      mPlayer.start();
  }
  while(TIME <= 5) {
    MillisTimer::run();
    fnd4.outputString(str, 2);
    mPlayer.play();
  }
```

2019 도대회 (중) 추억의 뽑기 게임

```
  // 메달 보너스 출력
  if(isWin) numMedal = random(20)+1;
  else return; // loop 탈출

  unsigned long startTime = millis();
  int preOutNum=-1;
  while(TIME <= 8) {
    MillisTimer::run();

    // 메달 출력 루틴에 진입한 이후 흐른 시간(0.1초단위)
    int outNum = (millis() - startTime) / 100;
    if(preOutNum!=outNum && outNum<numMedal) {
      mPlayer.setNote(noteTick, 1);
      mPlayer.start();
      preOutNum = outNum;
    }
    fnd4.outputNumber(min(outNum, numMedal));
    mPlayer.play();
  }
}
```

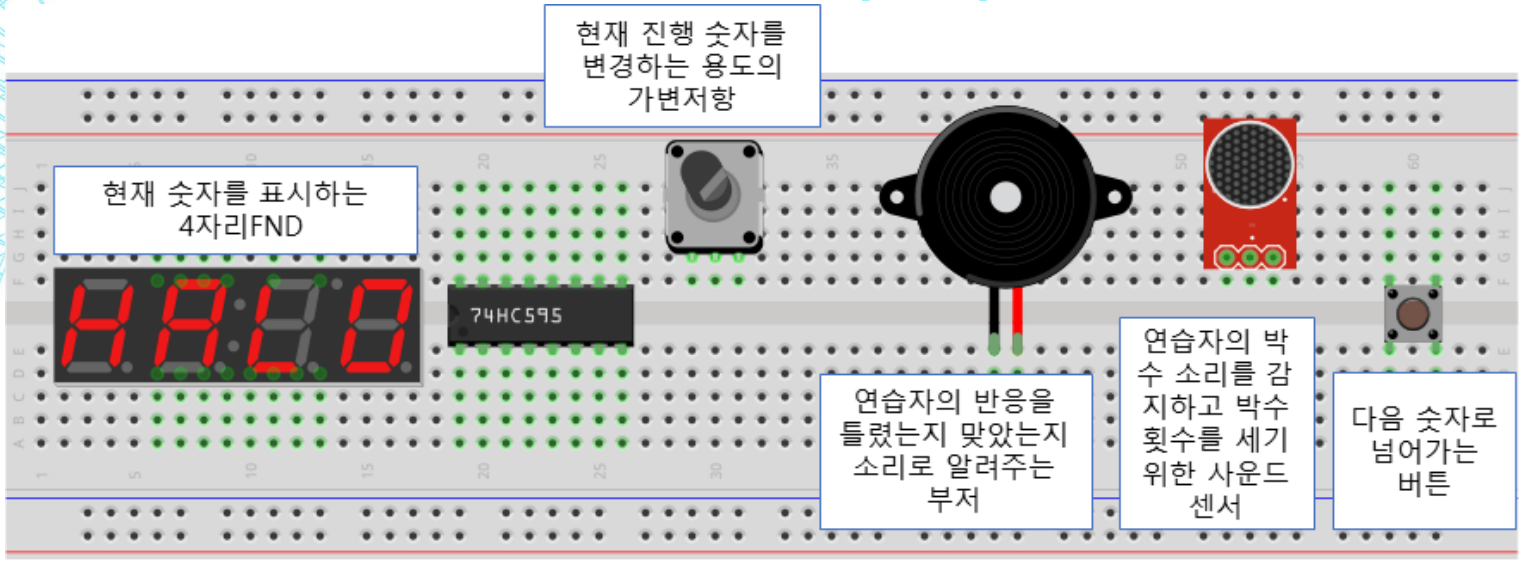
2022 도대회 (고) 3·6·9게임 연습기계

네 자리 FND에 현재 진행 중인 숫자를 표시한다. 그런데 매번 연습할 때마다 1부터 시작하면 재미가 없으니 가변저항 이용하여 진행 중인 숫자를 바꿀 수 있도록 한다. 가변저항을 왼쪽 끝으로 돌리면 1이 되고, 오른쪽 끝으로 돌리면 1000이 되는 식이다. 네 자리 FND에 현재 진행 중인 숫자가 표시되면(또는 가변저항을 돌려서 현재 진행 중인 숫자가 결정되면) 연습자는 1.5초 내에 올바른 반응을 해야 한다.

만약 현재 숫자가 3,6,9가 포함되는 숫자가 아니면 다음 버튼을 눌러야 하고 현재 숫자가 3,6,9가 포함되는 숫자이면 박수로 반응을 해야 하는데 현재 숫자에 포함된 3,6,9 갯수 만큼 박수를 쳐야 한다. (예를 들어 13이면 박수 1회, 36이면 박수 2회)

연습자의 반응에 기계가 응답을 해야 하는데 만약 올바르게 반응하였다면 '딩동~'하는 소리를 출력하고, 틀리게 반응하였다면'땡~'하는 소리를 출력한다. 만약 FND의 숫자가 바뀐지 1.5초가 지나도록 아무런 반응을 하지 않았다면 틀린 것으로 간주한다. 그리고 연습자의 반응에 따른 응답을 출력하였으면 즉시 다음 숫자로 넘어가야 한다.

2022 도대회 (고) 3·6·9게임 연습기계



① 현재 진행 중인 숫자가 네 자리 FND에 깜빡임 없이 자연스럽게 표시되어야 한다.
② 연습자의 박수 소리와 횟수를 올바르게 인식하여야 한다. (박수 횟수를 인식하기 어려우면 박수 소리만 감지하는 방식으로 간략화 가능. 단, 감점 있음)
③ 연습자가 반응하도록 주어지는 시간 타임아웃 1.5초가 올바르게 작동하여야 한다.
④ 연습자의 응답을 판단한 결과를 적절한 소리로 출력하여야 한다.
⑤ 가변저항으로 현재 진행 중인 숫자를 변경하는 기능이 자연스럽게 작동하여야 한다.
⑥ 시리얼 모니터에 게임 현황이 올바르게 출력되어야 한다.

해답: https://arduino.datahub.pe.kr/2023/src/5.%20Solution/2022-DO-HS%20369Game.ino