String Class와 디지털 입력

부산대학교 공과대학 전기컴퓨터공학부
정보컴퓨터공학전공

String Class

❖ 문자열 처리를 위한 Class

- Reference : https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/
- C++의 문자열 처리를 위한 std::string Class 등과 유사
- Serial Class와 같이 별도의 Header file include 없이 사용할 수 있는 기본 Class

❖ String(val,base, decimalPlaces)

- Constructs an instance of the String class
- 문자, 문자열, 숫자 등으로부터 String 객체 생성 가능
- val : a variable to format as a String - Allowed data types: string, char, byte, int, long, unsigned int, unsigned long, float, double
- base (optional): the base in which to format an integral value
- decimalPlaces (optional): (only if val is float or double): the desired decimal places

❖ Example Code

```
String stringOne = "Hello String";
//using a constant String

String stringOne =  String('a');
//converting a constant char into a
String

String stringTwo =  String("This is a
string");
//converting a constant string into a
String object

String stringOne =  String(stringTwo + "
with more");
// concatenating two strings

String stringOne =  String(13);
// using a constant integer

String stringOne =  String(analogRead(0),
DEC);
// using an int and a base

String stringOne =  String(45, HEX);
// using an int and a base (hexadecimal)

String stringOne =  String(255, BIN);
// using an int and a base (binary)

String stringOne =  String(millis(),
DEC);
// using a long and a base

String stringOne =  String(5.698, 3);
// using a float and the decimal places
```

String Class : Operators

❖ Supported Operators

- [] (element access)
- + (concatenation)
- += (append)
- == (comparison)
- > (greater than)
- >= (greater than or equal to)
- < (less than)
- <= (less than or equal to)
- != (different from)

❖ + : concatenation

- String 객체와 다른 타입의 데이터 결합으로 새로운 String 객체 생성
- String("abc") + 123;
  // String 객체 + 정수
- cf) " " 표시되는 문자열은 String 클래스 객체가 아닌 문자 배열로 처리
- "abc" + 123;
  // 문자열 배열 + 정수 (×)

String Class : Member Functions

❖ Member Functions

- charAt()
- compareTo()
- concat()
- c_str()
- endsWith()
- equals()
- equalsIgnoreCase()
- getBytes()
- indexOf()
- lastIndexOf()
- length()
- remove()
- replace()
- reserve()
- setCharAt()
- startsWith()
- substring()
- toCharArray()
- toInt()
- toFloat()
- toLowerCase()
- toUpperCase()
- trim()

❖ int String::compareTo()

- Description
  - Compares two Strings, testing whether one comes before or after the other, or whether they're equal. The strings are compared character by character, using the ASCII values of the characters. That means, for example, that 'a' comes before 'b' but after 'A'. Numbers come before letters.
- Syntax : string.compareTo(string2)
- Returns
  - a negative number: if string comes before string2
  - 0: if string equals string2
  - a positive number: if string comes after string2

❖ bool String::equals()

❖ bool String::equalsIgnoreCase()

기초 정렬 알고리즘

- **Selection Sort**
  - http://www.cs.armstrong.edu/liang/animation/web/SelectionSort.html
- **Insertion Sort**
  - http://cs.armstrong.edu/liang/animation/web/InsertionSortNew.html
- **Bubble Sort**
  - http://cs.armstrong.edu/liang/animation/web/BubbleSortNew.html

Lab.4-1

String Class를 이용한 문자열 처리

❖ 실험 순서

① String 객체 생성 및 출력 (Sketch 4-4, Textbook pp. 73~74)

② String Class를 활용한 문자열 정렬 (Sketch 4-5, Textbook pp. 75)

③ Serial로부터 문자열 입력 받기

④ 실행 결과 확인

① String 객체 생성 및 출력

Sketch 4-4, Textbook pp. 73~74

```
void setup() {
  Serial.begin(9600);                    // 시리얼 포트 초기화
}

void loop() {
  String str1 = "One string", str2 = "Another string";
  int n = 1234;
  float f = 3.14;
  char c = 'A';

  Serial.println(str1);                  // String 출력
  Serial.println(str1 + " " + str2);     // 문자열 연결
  Serial.println(String(n));             // 정수로부터 10진 문자열 생성
  Serial.println(String(n, BIN));        // 정수로부터 2진 문자열 생성
  Serial.println(String(n, HEX));        // 정수로부터 16진 문자열 생성

// Serial.println(String(f));
  Serial.println(f);

  // 다른 타입의 데이터를 연결하여 새로운 String 객체를 생성한다.
  Serial.println("String + integer : " + String(n));
  String str3 = "String + character : ";
  str3 += n;
  Serial.println(str3);       //Serial.println("String + character : " + 1234) ?

  while (true);
}
```

오류가 발생하는 이유는?

② String Class를 활용한 문자열 정렬

Sketch 4-5, Textbook pp. 75~76

```
void setup() {
  Serial.begin(9600);                    // 시리얼 포트 초기화
}

void loop() {
  // 정렬할 문자열 배열
  String str[5] = {"abc", "ABC", "!@#", "라라라", "123"};

  // 문자열 정렬
  for (int i = 0; i < 4; i++) {
    for (int j = i + 1; j < 5; j++) {
      int compare = str[i].compareTo(str[j]);
      if (compare > 0) { // 오름차순으로 정렬
        String temp = str[i];
        str[i] = str[j];
        str[j] = temp;
      }
    }
  }

  // 정렬된 문자열 출력3
  for (int i = 0; i < 5; i++) {
    Serial.println(String(i) + " : " + str[i]);
  }

  while (true);
}
```

- 정렬할 문자열 배열이 {"red", "orange", "yellow", "green", "blue", "indigo", "violet"} 일 때 정렬 과정을 설명하면?
- 오름차순이 아닌 내림차순 정렬로 바꾸려면?

③ Serial로부터 문자열 입력 받기

❖ Arduino 문자열 입력 받기?

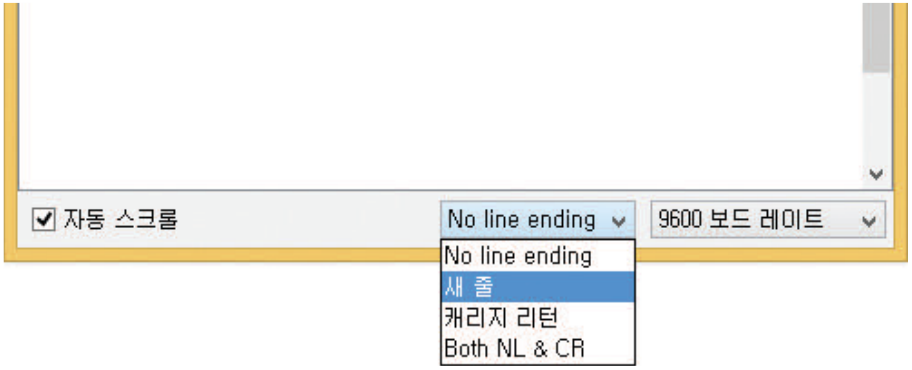
- Arduino Serial Class에서 제공하는 입력 멤버 함수들
  - read(), readBytes(), readBytesUntil(), parseInt(), parseFloat(), peek(), …
- Arduino에는 문자열 입력을 바로 지원하는 멤버 함수가 없음
  - 표준 C의 gets()와 같이 문자열을 입력 받는 간단한 함수가 없음
    - Arduino는 표준 C의 <stdio.h>의 함수들, 즉 printf(), scanf(), puts(), gets() 등을 기본으로 제공하지 않음
    - Arduino는 주로 장치 제어용으로 쓰임. 사람과의 Interface가 필요한 경우 거의 없음
- read(), readBytesUntil() 등의 함수를 활용하여 문자열 입력을 처리

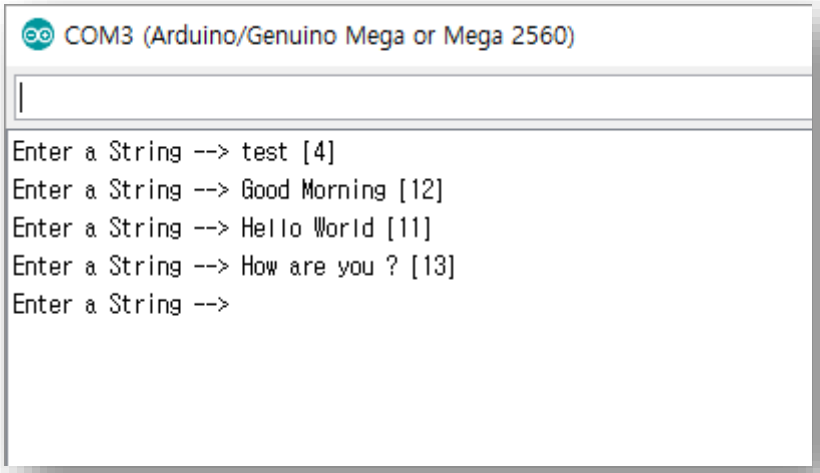
Sketch - Lab 4-1-3a

```
void setup() {
  Serial.begin(9600); // 시리얼 포트 초기화
}

void loop() {
  int state = 1, len = 0;
  char buffer[128];

  while (true) {
    if (state == 1) {
      Serial.print("Enter a String --> ");
      state = 2;
    }
    while (Serial.available()) {
      char data = Serial.read();
      if (data == '\n') {
      // 개행 문자 '\n'을 만날 때까지 읽음
        buffer[len] = '\0';
        String in_str = buffer;
        Serial.println(in_str + " [" +
          in_str.length()+"]");
        state = 1;
        len = 0;
        break;
      }
      buffer[len++] = data;
    }
  }
}
```

③ Serial로부터 문자열 입력 받기

실행을 위해서는 아래 그림과 같이 Serial Monitor에서 Enter 키 입력 시 새줄(New Line) 제어 문자를 전송하도록 설정하여야 함



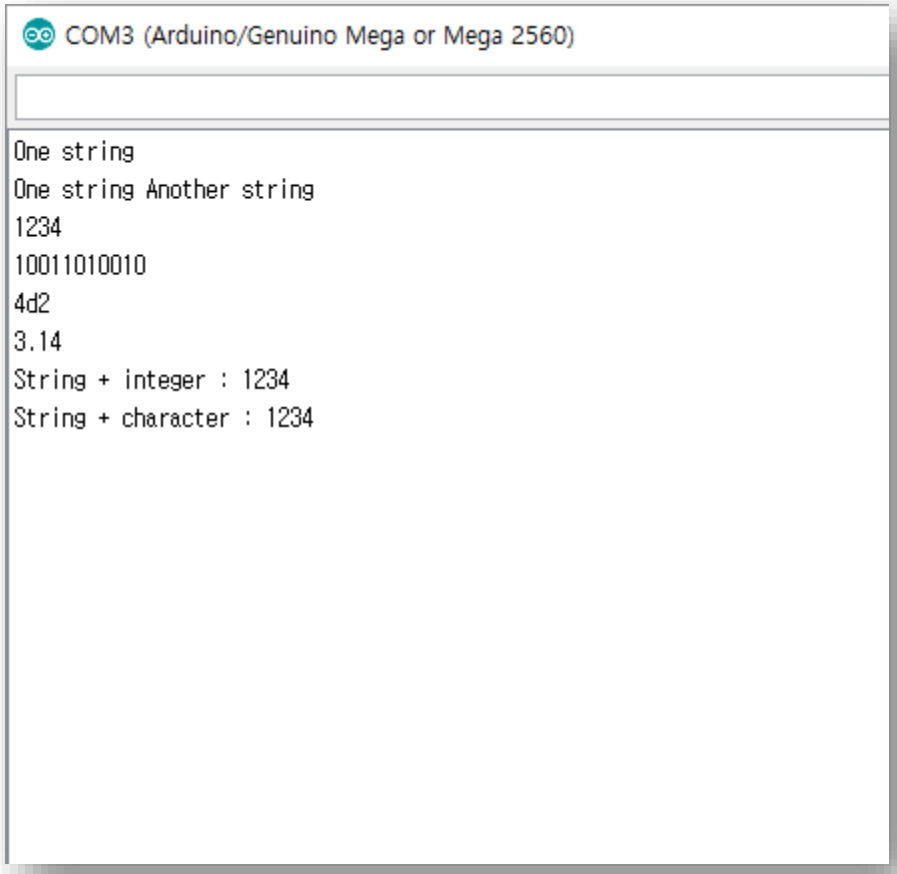


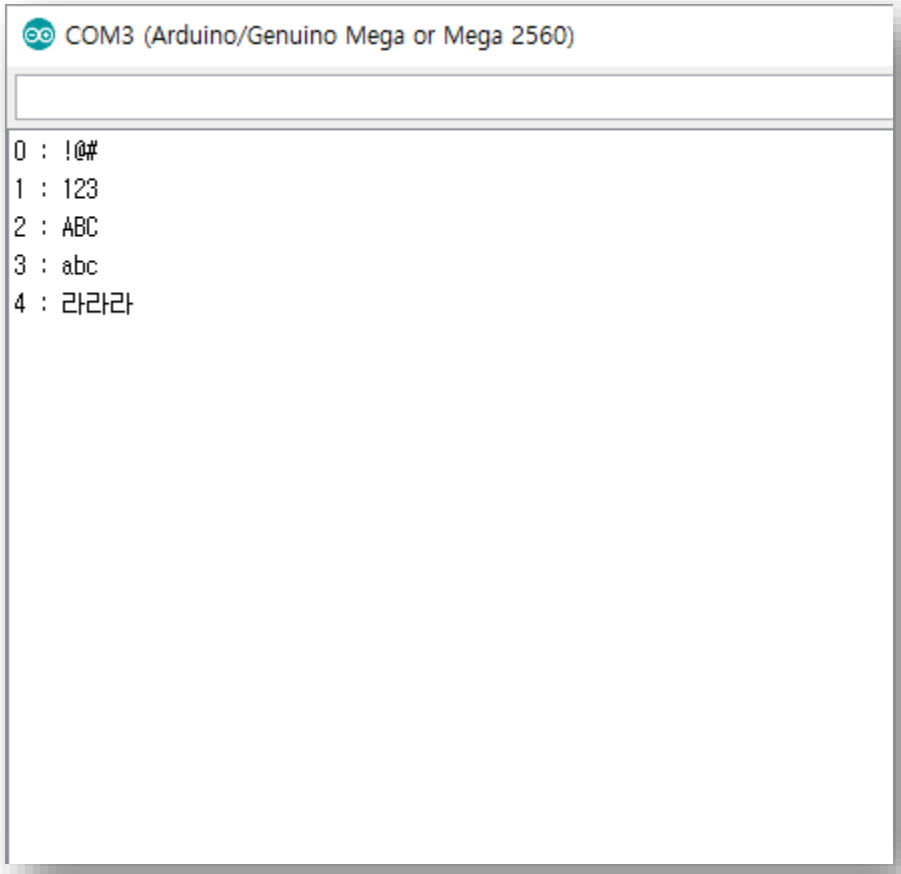
Sketch - Lab 4-1-3b

```
void setup() {
  Serial.begin(9600); // 시리얼 포트 초기화
}

void loop() {
  int state = 1;
  char buffer[128];

  while (true) {
    if (state == 1) {
      Serial.print("Enter a String --> ");
      state = 2;
    }
    while (Serial.available()) {
      int len = Serial.readBytesUntil('\n',
                buffer, 127);
      if (len > 0) {
        buffer[len] = '\0';
        String in_str = String(buffer);
        Serial.println(in_str + " [" +
          in_str.length()+"]");
        state = 1;
        break;
      }
    }
  }
}
```

④ 결과 확인

1. 작성한 4개의 Sketch 편집 창을 보여라
2. Lab 4-1-3b Sketch의 실행을 Serial Monitor를 통해 보여라
3. 조교의 물음에 답하고 채점 결과를 PLMS LAB 4-1에 직접 입력하라





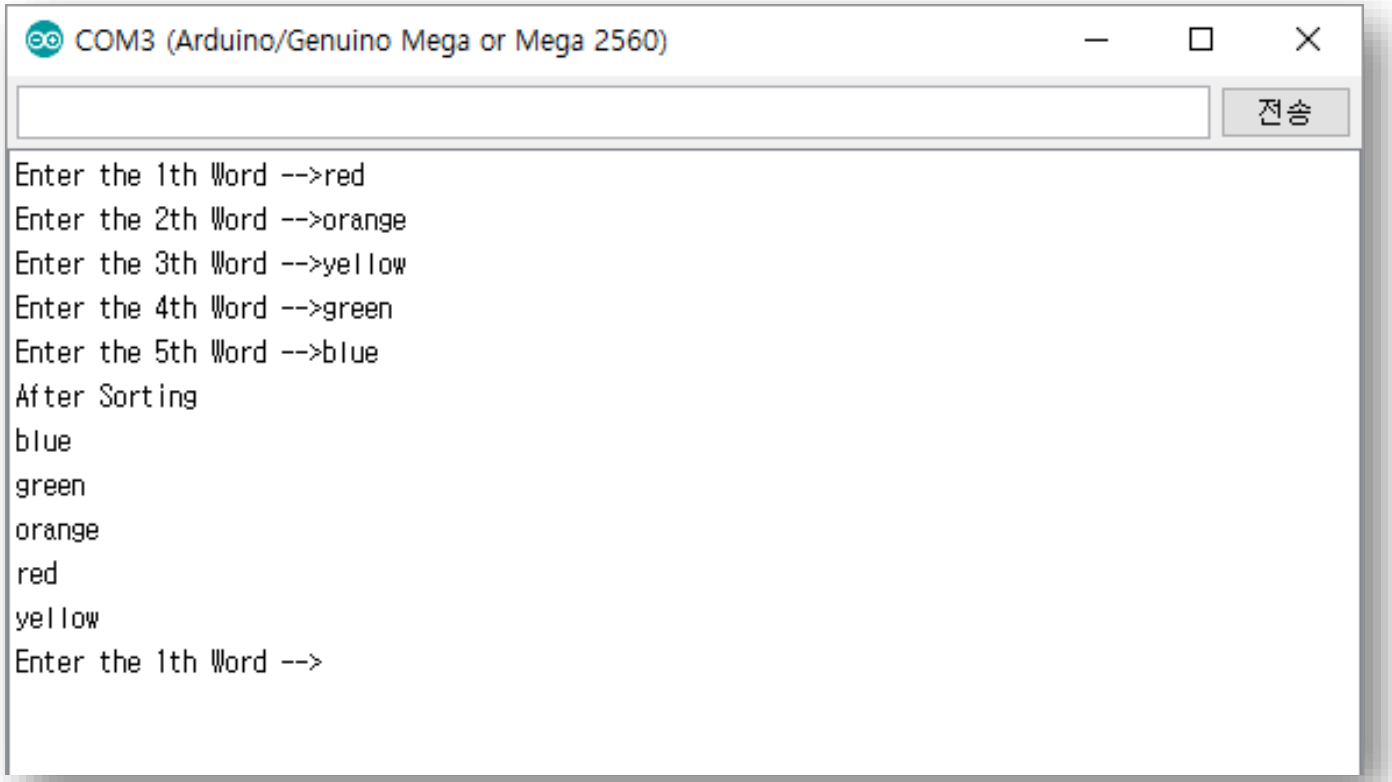
HW.4-1

실습 숙제

❖ 아래 그림의 예와 같이 Serial Monitor에서 5개의 단어를 입력 받고 오름차순으로 정렬하여 출력하는 Sketch를 작성하라

❖ 제출 방법

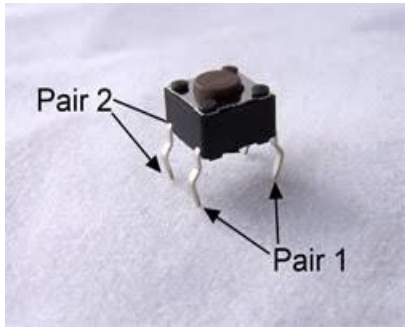
1. PLMS의 HW 4-1에 작성한 Sketch 코드를 입력하라
2. 또한 실행한 Serial Monitor를 캡쳐하여 이미지로 포함하라

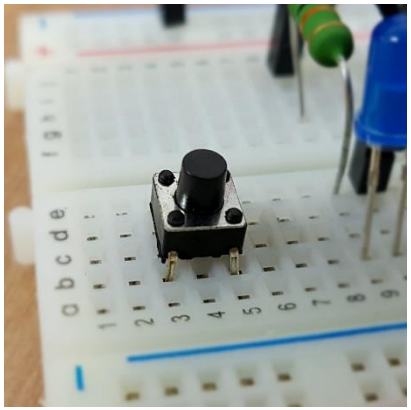


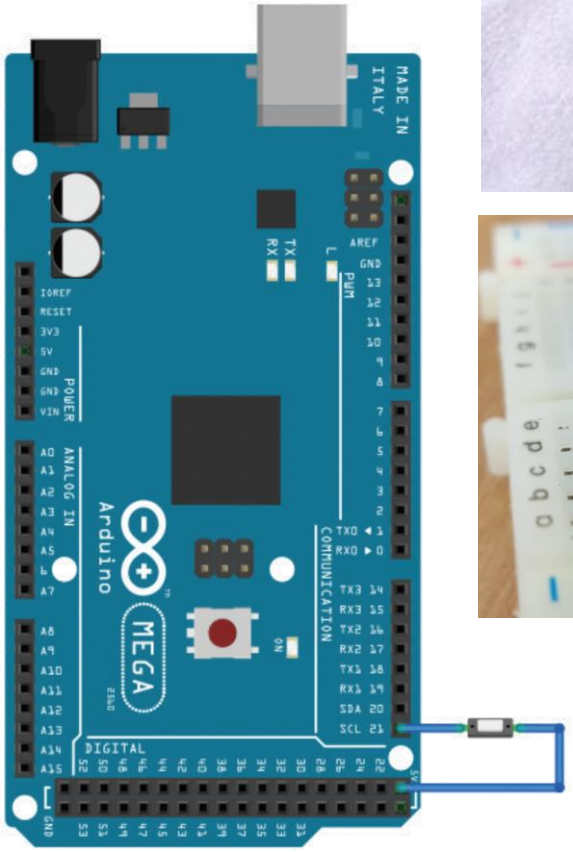
디지털 입력 - 버튼

버튼 연결 – 기본 회로

❖ 버튼 on(push) : 1 (Vcc)



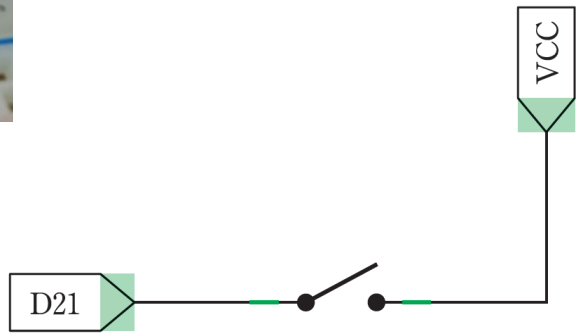




❖ 버튼 Off : 0 ?

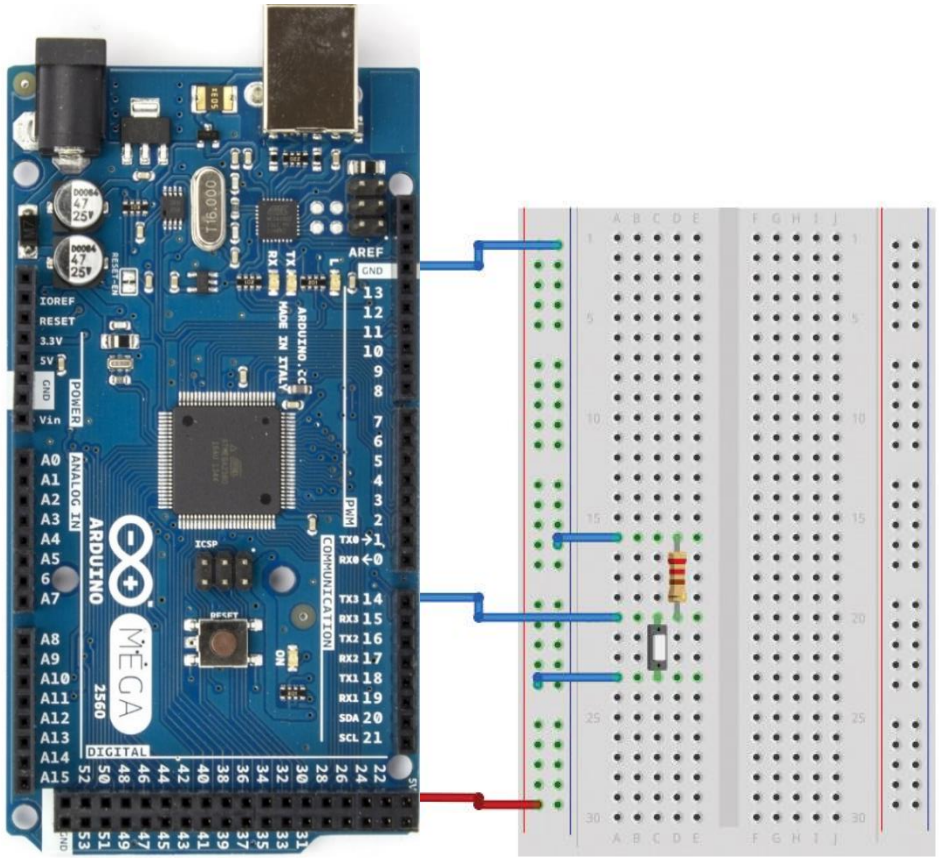
❖ Floating Input 문제 발생

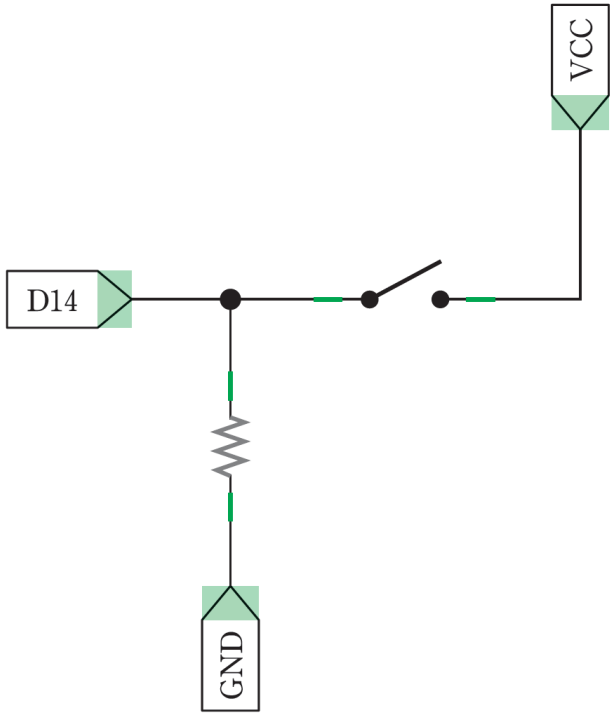
- Digital 입력 핀에 아무런 회로가 연결 되지 않은 상황
- 주변 핀의 상태나 정전기 등에 의해 임의의 값이 입력될 수 있음
- 버튼을 사용할 때는 회로가 오픈 되는 경우를 피해야 한다
- 풀다운 저항의 사용



버튼 연결 – 풀다운 저항

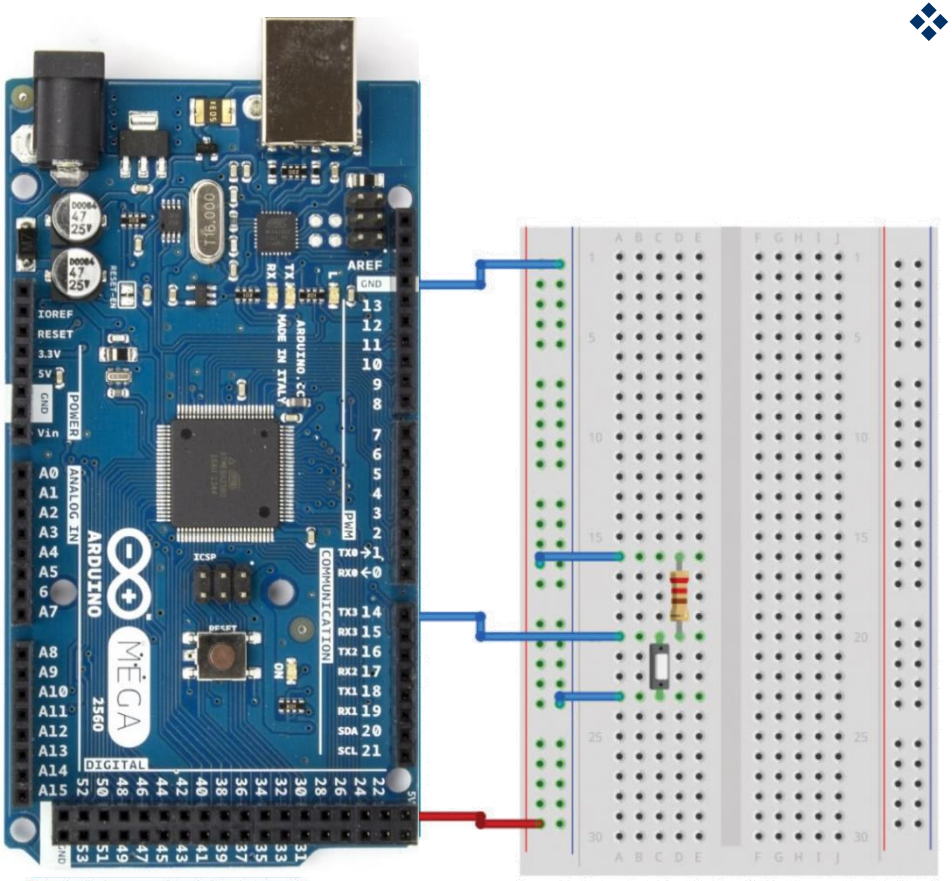
- ❖ Pull-down
- ❖ 버튼 on(push) : 1 (Vcc)
- ❖ 버튼 off : 0 (Gnd)

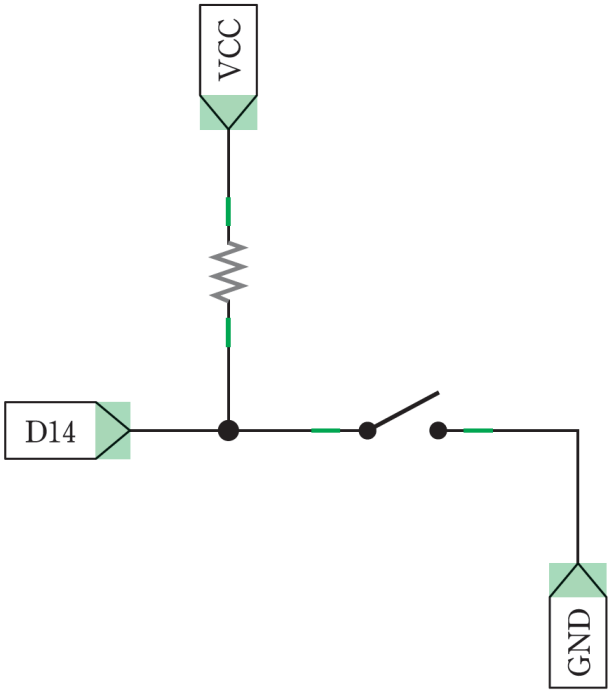




버튼 연결 – 풀업 저항

- ❖ Pull-up
- ❖ 버튼 on(push) : 0 (Gnd)
- ❖ 버튼 off : 1 (Vcc)





버튼 연결

풀업/풀다운 저항 사용	버튼 누르지 않음	버튼 누름
사용 안함	플로팅 (1이나 0이 아닌 미결정 상태)	1
풀다운 저항 사용	0	1
풀업 저항 사용	1	0

Lab.4-2

풀업 및 풀다운 버튼 및 이를 활용한 제어

❖ 실험 순서

① 풀업/풀다운 버튼 연결 회로 구성

② 버튼 상태 출력하기 (Sketch 6-1, Textbook pp. 103)

③ 내장 풀업 저항 사용 버튼 연결 회로 구성

④ 내장 풀업 저항이 연결된 버튼 상태 출력 (Sketch 6-2, Textbook pp.106)

⑤ LED 및 버튼 연결 회로 구성

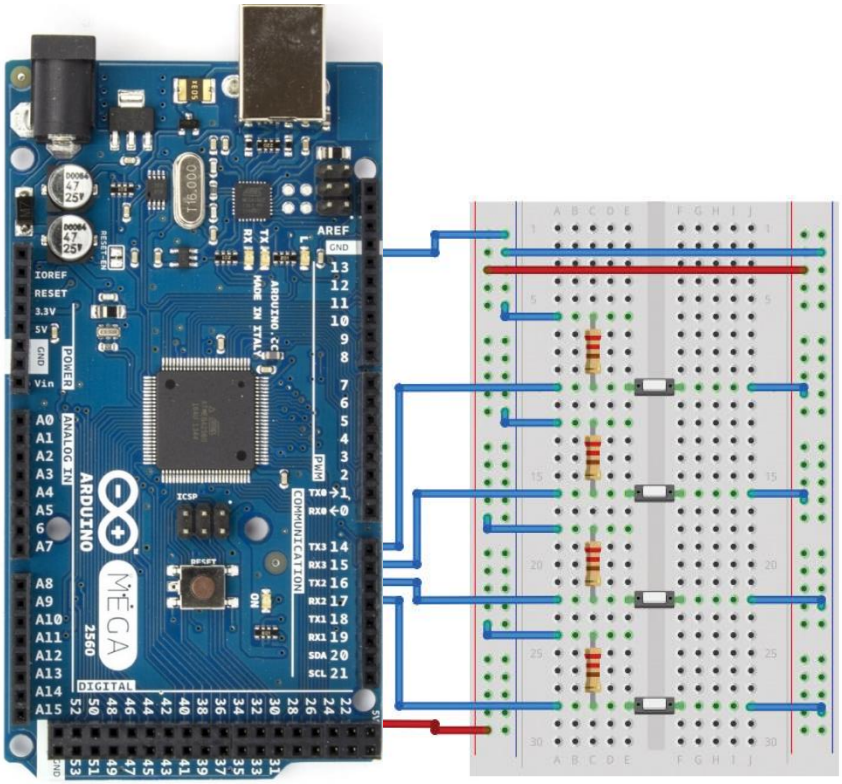
⑥ 버튼 입력을 LED로 표시하기 (Sketch 6-3, Textbook pp. 107~108)

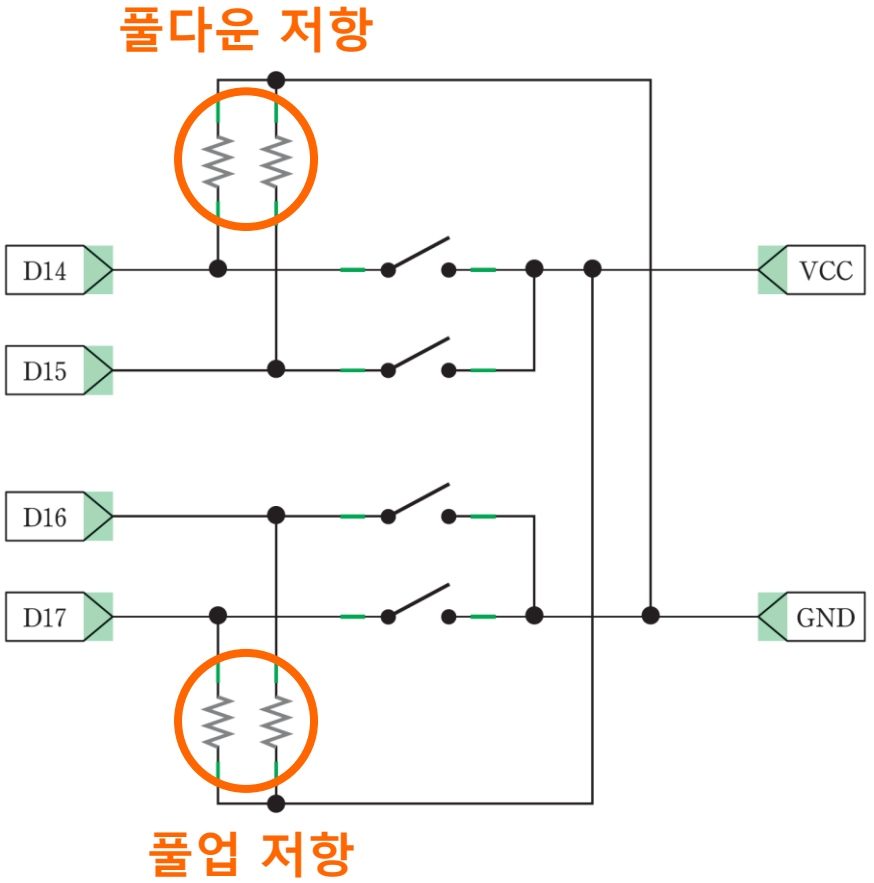
⑦ 버튼으로 LED 점멸하기 (Sketch 6-4, Textbook pp. 109)

⑧ 버튼 누른 횟수 세기 (Sketch 6-5, Textbook pp. 111~112)

⑨ 실행 결과 확인

① 풀업/풀다운 버튼 연결 회로 구성

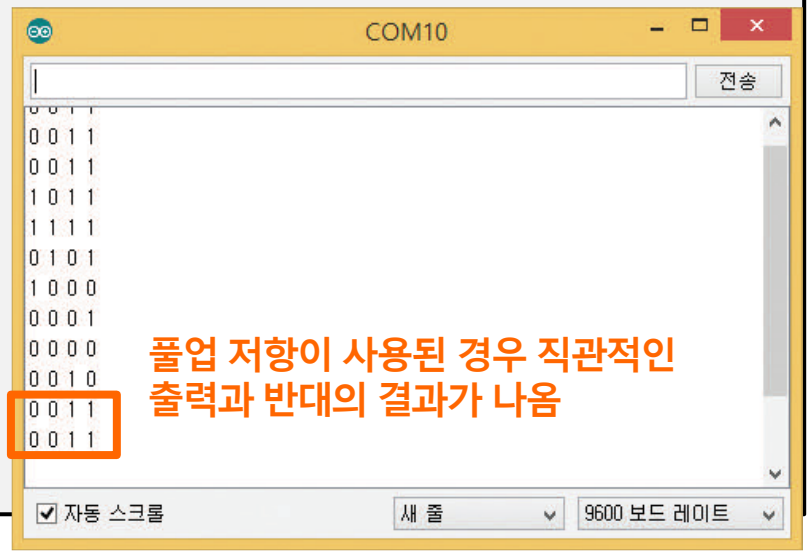




② 버튼 상태 출력하기

Sketch 6-1, Textbook pp. 103

```
int buttons[] = {14, 15, 16, 17}; // 버튼 연결 핀

void setup() {
  Serial.begin(9600); // 시리얼 통신 초기화
  for (int i = 0; i < 4; i++) { // 버튼 연결 핀을 입력으로 설정
    pinMode(buttons[i], INPUT);
  }
}
void loop() {
  for (int i = 0; i < 4; i++) {
    Serial.print(digitalRead(buttons[i])); // 버튼 상태 출력
    Serial.print(" ");
  }
  Serial.println(); // 줄 바꿈
  delay(1000);
}
```



풀업 저항이 사용된 경우 직관적인 출력과 반대의 결과가 나옴

데이터 입력을 위한 함수

void pinMode(uint8_t pin, uint8_t mode)

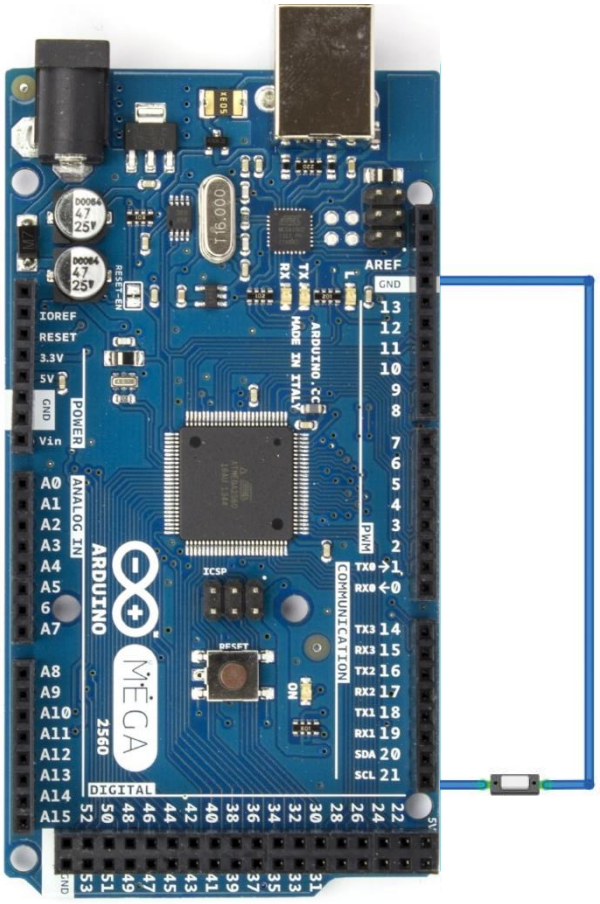
- **매개변수**
  - pin : 설정하고자 하는 핀 번호
  - mode : INPUT, OUTPUT, INPUT_PULLUP 중 하나
- **반환값** : 없음

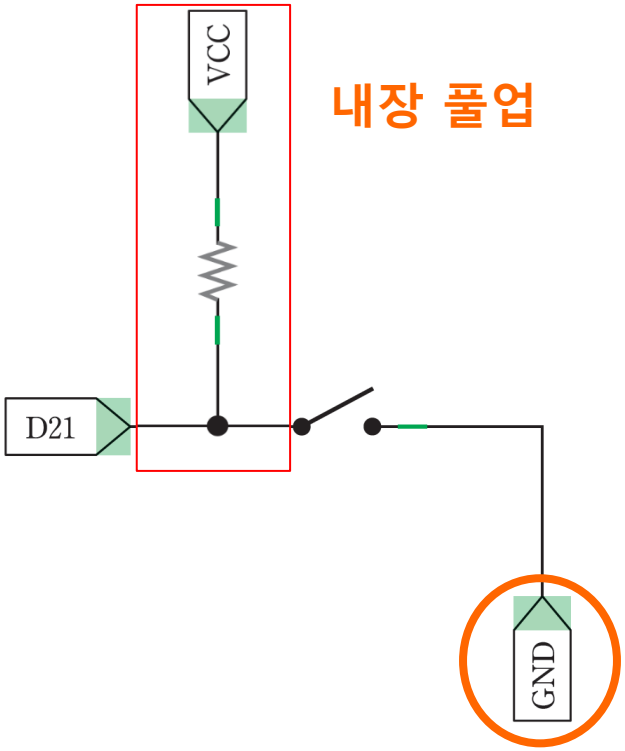
마이크로컨트롤러 내부에
내장 풀업 저항이 포함되어 있음

int digitalRead(uint8_t pin)

- **매개변수**
  - pin : 핀 번호
- **반환값** : HIGH(1) 또는 LOW(0)

③ 내장 풀업 저항 사용 버튼 연결 회로 구성





버튼 연결 기본 회로에서
는 VCC로 연결

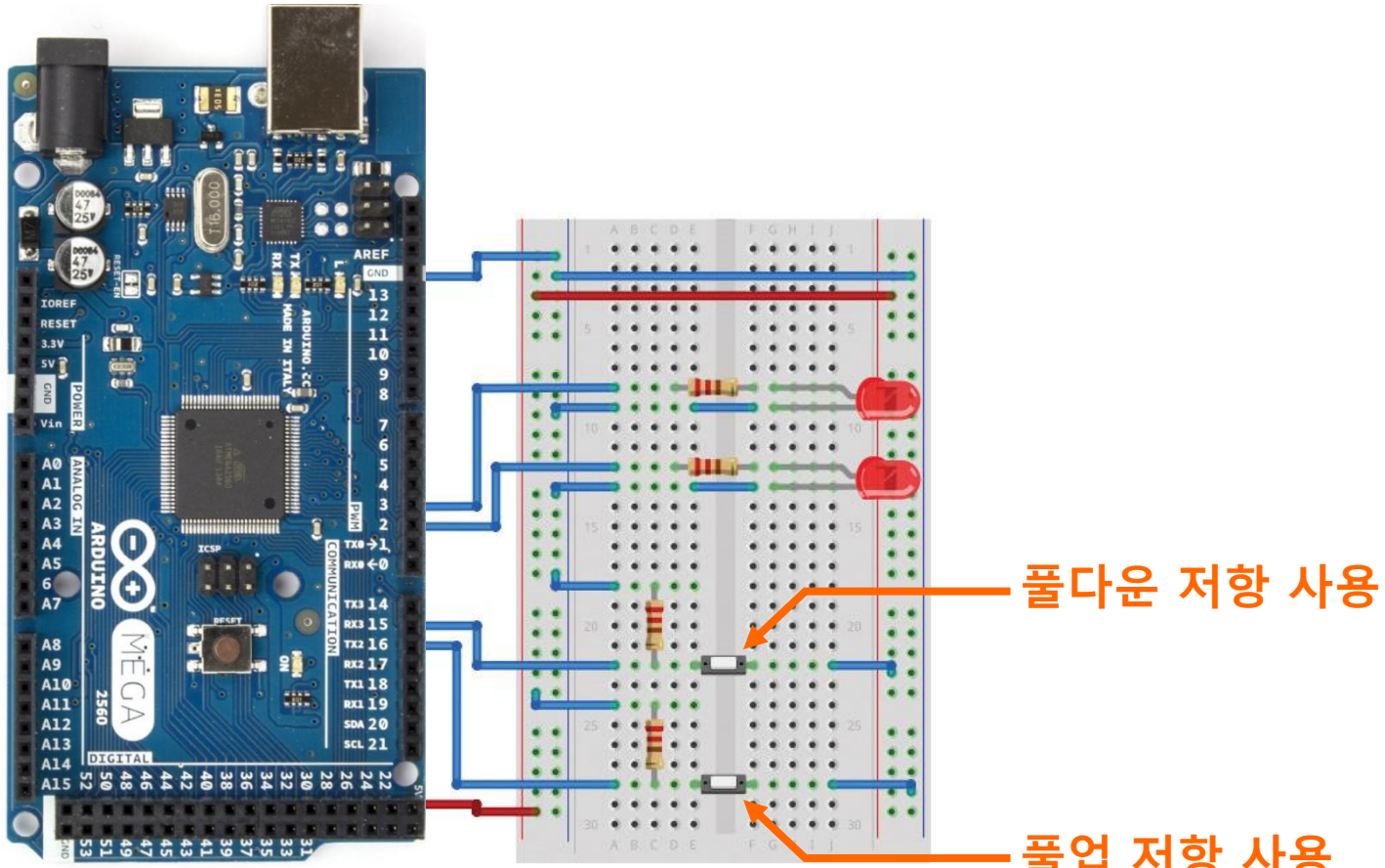
❖ Arduino의 각 핀에는 내장 풀업 저항이 달려있음

④ 내장 풀업 저항이 연결된 버튼 상태 출력

Sketch 6-2, Textbook pp. 106

```
int button = 21; // 버튼 연결 핀

void setup() {
  Serial.begin(9600); // 시리얼 통신 초기화
  pinMode(button, INPUT_PULLUP); // 버튼 연결 핀을 입력으로 설정
}

void loop() {
  Serial.println(digitalRead(button)); // 버튼 상태 출력
  delay(1000);
}
```

⑤ LED 및 버튼 연결 회로 구성



⑥ 버튼 입력을 LED로 표시하기

Sketch 6-3, Textbook pp. 107~108

```
int pins_LED[] = {2, 3}; // LED 연결 핀
// 버튼 연결 핀 (15 : 풀다운 저항, 16 : 풀업 저항)
int pins_button[] = {15, 16};

void setup() {
  Serial.begin(9600); // 시리얼 통신 초기화
  for (int i = 0; i < 2; i++) {
    pinMode(pins_button[i], INPUT); // 버튼 연결 핀을 입력으로 설정
    pinMode(pins_LED[i], OUTPUT); // LED 연결 핀을 출력으로 설정
  }
}

void loop() {
  for (int i = 0; i < 2; i++) {
    boolean state = digitalRead(pins_button[i]); // 버튼 상태 읽기
    digitalWrite(pins_LED[i], state); // LED 출력
    Serial.print(state);
    Serial.print(" ");
  }
  Serial.println();
  delay(1000);
}
```

⑦ 버튼으로 LED 점멸하기

❖ 풀업 저항이 사용된 경우 버튼이 눌러지지 않으면 HIGH가 입력되어 직관적인 동작과 반대임

❖ 버튼의 상태에 따라 실제 LED 패턴값을 배열로 작성

```
boolean on_off_pattern[2][2] = {
   {false, true},   // 풀다운 저항을 사용한 경우
   {true, false}    // 풀업 저항을 사용한 경우
};
```

❖ 버튼 상태에 따라 패턴값으로 LED 제어

```
boolean state = digitalRead(pins_button[i]);
digitalWrite(pins_LED[i], on_off_pattern[i][state]);
```

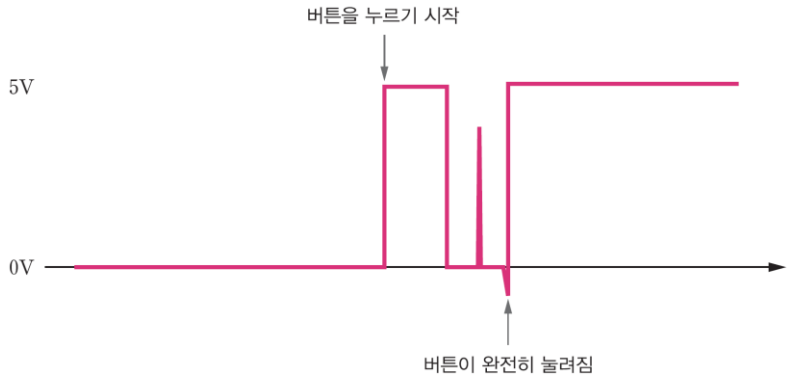
⑦ 버튼으로 LED 점멸하기

Sketch 6-4, Textbook pp. 109

```
int pins_LED[] = {2, 3}; // LED 연결 핀
// 버튼 연결 핀 (15 : 풀다운 저항, 16 : 풀업 저항)
int pins_button[] = {15, 16};

boolean on_off_pattern[2][2] = { // 점멸 패턴
  {false, true}, // 풀다운 저항을 사용한 경우
  {true, false} // 풀업 저항을 사용한 경우
};

void setup() {
  Serial.begin(9600); // 시리얼 통신 초기화
  for (int i = 0; i < 2; i++) {
    pinMode(pins_button[i], INPUT); // 버튼 연결 핀을 입력으로 설정
    pinMode(pins_LED[i], OUTPUT); // LED 연결 핀을 출력으로 설정
  }
}

void loop() {
  for (int i = 0; i < 2; i++) {
    boolean state = digitalRead(pins_button[i]); // 버튼 상태 읽기
    digitalWrite(pins_LED[i], on_off_pattern[i][state]); // LED 출력
    Serial.print(on_off_pattern[i][state]);
    Serial.print(" ");
  }
  Serial.println();
  delay(1000);
}
```

⑧ 버튼 누른 횟수 세기

- 버튼이 눌러진 경우 횟수를 증가시키면, 버튼을 누른 상태로 있으면 횟수가 계속 증가
- 버튼의 이전 상태(state_previous)와 현재 상태(state_current)를 비교하여, 이전 상태가 눌러지지 않은 상태이고 현재 상태가 눌러진 상태인 경우에만 횟수를 증가
- 버튼의 기계적인 진동에 의해 버튼을 누를 때 ON/OFF가 반복해서 나타날 수 있으며, 이를 바운싱(bouncing) 또는 채터링(chattering)이라 함
  - 채터링을 없애는 것을 디바운싱이라 함



⑧ 버튼 누른 횟수 세기

Sketch 6-5, Textbook pp. 111~112

```
int pin_button = 15; // 버튼 연결 핀
boolean state_previous = false; // 버튼의 이전 상태
boolean state_current; // 버튼의 현재 상태
int count = 0; // 버튼을 누른 횟수

void setup() {
  Serial.begin(9600); // 시리얼 통신 초기화
  pinMode(pin_button, INPUT); // 버튼 연결 핀을 입력으로 설정
}

void loop() {
  state_current = digitalRead(pin_button); // 버튼 상태 읽기
  if (state_current) { // 버튼을 누른 경우
    if (state_previous == false) { // 이전 상태와 비교
      count++; // 상태가 바뀐 경우에만 횟수 증가
      state_previous = true;
      Serial.println(count);
    }
    // delay(50); // 디바운싱
  }
  else {
    state_previous = false;
  }
}
```

⑨ 결과 확인

1. 작성한 5개의 Sketch 편집 창을 보여라
2. Lab 6-4 Sketch의 실행을 보여라
3. 조교의 물음에 답하고 채점 결과를 PLMS LAB 4-2에 직접 입력하라

HW.4-2

실습 과제

❖ 연습문제 6.3

- 디지털 2번 핀에서 5번 핀까지 4개의 LED를 연결하고 디지털 14번 핀에 풀다운 저항을 통해 버튼을 연결한다.
- 연결된 LED는 1->2->3->4->1 ... 순서로 0.5초 간격으로 반복 점멸되도록 한다.
- 단, 버튼이 눌러진 경우 턴의 진행 방향이 바뀌도록 스케치를 작성해보자.
- 즉, 버튼이 한번 눌려지면 점멸 방향이 바뀌어 4->3->2->1->4 ... 순서의 역방향으로 진행되도록 한다. 시작 시에는 순방향으로 패턴이 바뀌는 것으로 한다.

❖ 제출 방법

1. PLMS의 HW 4-2에 작성한 Sketch 코드를 입력하라
2. 또한 **실행 결과를 촬영한 동영상 링크를 포함**하여 제출하라.

패턴	디지털 핀			
	2번	3번	4번	5번
1	■			
2		■		
3			■	
4				■

맺는말

❖ 데이터 핀을 통한 디지털 데이터 입력

- digitalRead 함수로 비트 단위 입력이 가능하지만
- pinMode 함수로 입력으로 사용할 것임을 먼저 지정해야 함

❖ 버튼이 눌러지지 않은 경우 회로가 오픈 상태에 있어 핀으로의 입력이 결정되지 않는 플로팅 상태 발생

- 풀업 저항으로 버튼이 눌러지지 않은 경우 HIGH가 입력되도록 설정
- 풀다운 저항으로 버튼이 눌러지지 않은 경우 LOW가 입력되도록 설정
- ATmega2560 내부에는 풀업 저항이 포함되어 있으므로 별도의 저항 연결 없이 사용 가능