String Class와 디지털 입력

부산대학교 정보·의생명공학대학
정보컴퓨터공학부

String Class

❖ Class for String Manipulation

- 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()

Basic Sorting Algorithms

- **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 Manipulation using String Class

❖ Experiment Sequence

① Creating and Printing a String Object (Sketch 4-4, Textbook pp. 73~74)

② String Sort using String Class (Sketch 4-5, Textbook pp. 75)

③ Receiving a String Input from Serial

④ Check the Result

① Creating and Printing a String Object

Sketch 4-4, Textbook pp. 73~74

```
void setup() {
  Serial.begin(9600);                    // Open the serial port at 9600 bps
}

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

  Serial.println(str1);                  // Prints out a string
  Serial.println(str1 + " " + str2);     // Concatenats strings
  Serial.println(String(n));             // Print as an ASCII-encoded decimal
  Serial.println(String(n, BIN));        // print as an ASCII-encoded binary
  Serial.println(String(n, HEX));        // print as an ASCII-encoded hexadecimal

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

  // Creates new String object concatenating different types of data
  Serial.println("String + integer : " + String(n));
  String str3 = "String + character : ";
  str3 += n;
  Serial.println(str3);       //Serial.println("String + character : " + 1234) ?

  while (true);
}
```

Why this code does not work?

② String Sort using String Class

Sketch 4-5, Textbook pp. 75~76

```
void setup() {
  Serial.begin(9600);                    // Open the serial port at 9600 bps
}

void loop() {
  // array of strings
  String str[5] = {"abc", "ABC", "!@#", "라라라", "123"};

  // string sort
  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) { // sorts an array in ascending order
        String temp = str[i];
        str[i] = str[j];
        str[j] = temp;
      }
    }
  }

  // prints an sorted array of strings
  for (int i = 0; i < 5; i++) {
    Serial.println(String(i) + " : " + str[i]);
  }

  while (true);
}
```

- Explain the sorting precedure of the following array, {"red", "orange", "yellow", "green", "blue", "indigo", "violet"}?
- How can you sort an array in decending order, not ascending?

③ Receiving a String Input from Serial

❖ Receiving a string input in Arduino?

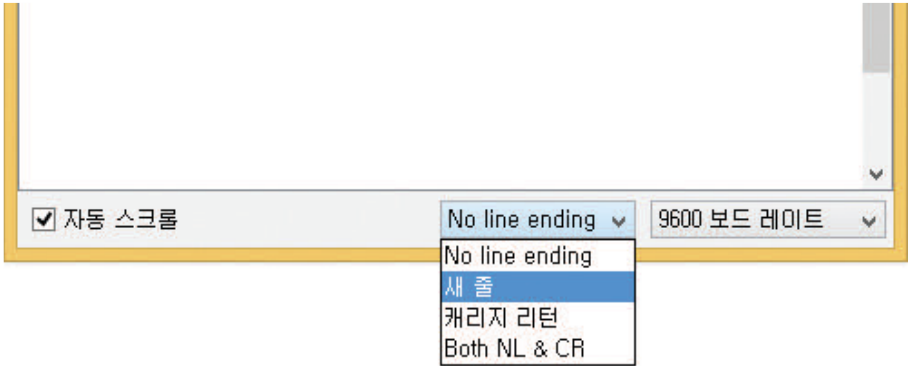
- Input functions of Arduino Serial Class
  - read(), readBytes(), readBytesUntil(), parseInt(), parseFloat(), peek(), …
- No function for string input in Arduino.
  - There is no function receiving a string input like gets() of standard C.
    - Arduino doesn't support <stdio.h> functions of standard C, such as printf(), scanf(), puts(), and gets()
    - Usually, Arduino is used for hardware control. So, human interaction is rare.
- Receiving a string input using read() or readBytesUntil()

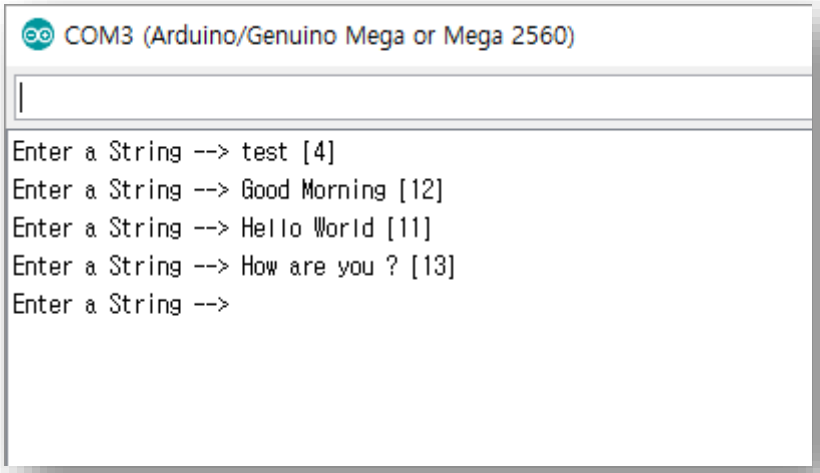
Sketch - Lab 4-1-3a

```
void setup() {
  Serial.begin(9600); // Open the serial port
}

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') {
      // reads input until newline '\n'
        buffer[len] = '\0';
        String in_str = buffer;
        Serial.println(in_str + " [" +
          in_str.length()+"]");
        state = 1;
        len = 0;
        break;
      }
      buffer[len++] = data;
    }
  }
}
```

③ Receiving a String Input from Serial

Select 'New Line'(새줄) to send new line character for every Enter in Serial Monitor as you can see the figure below.



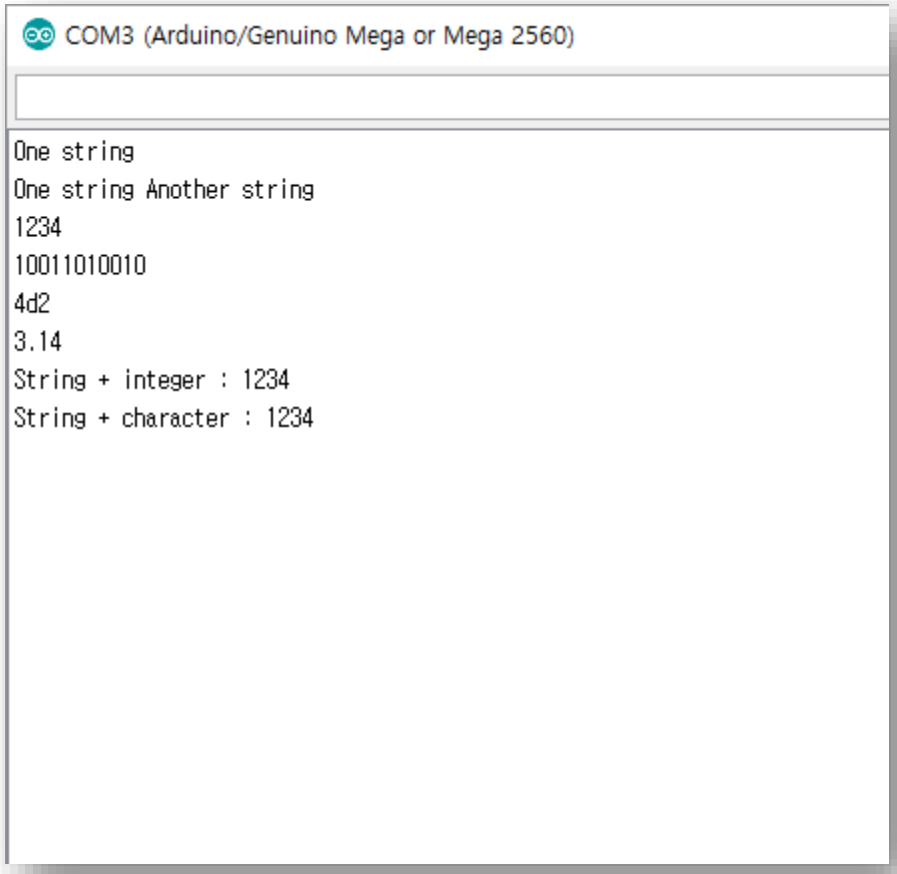


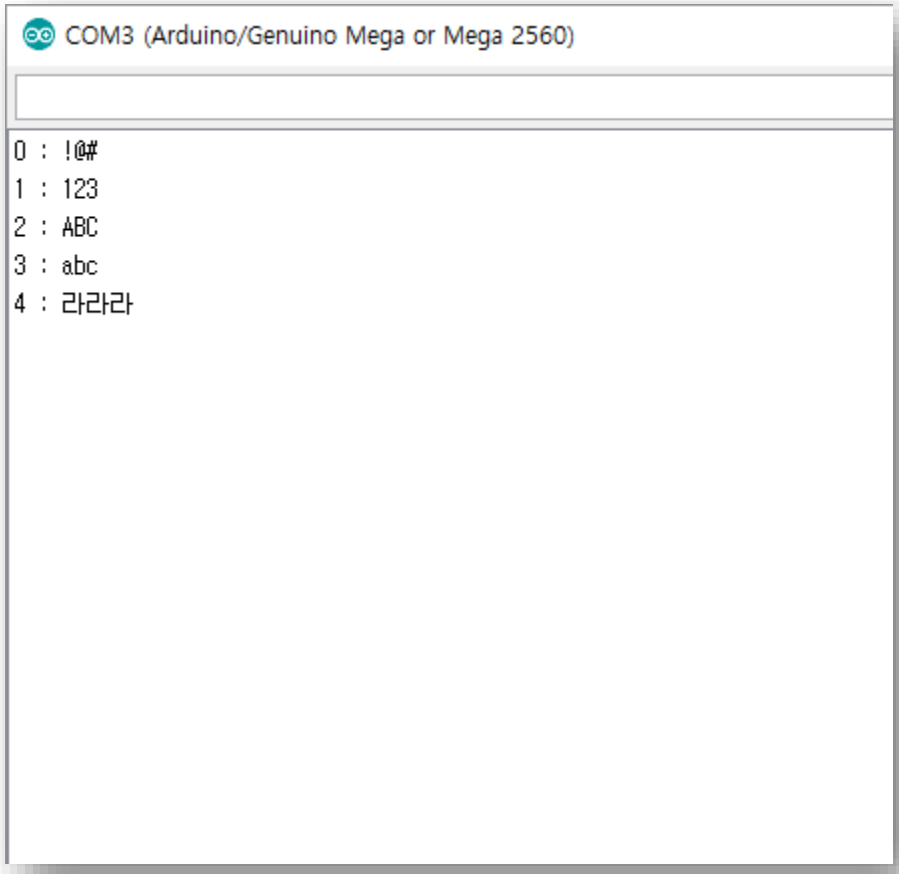
Sketch - Lab 4-1-3b

```
void setup() {
  Serial.begin(9600); // Open the serial port
}

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 Check the Result

1. Show your four Sketch code
2. Show the execution of Lab 4-1-3b Sketch on a Serial Monitor
3. Answer the TA's questions and insert the score result to PLMS LAB 4-1





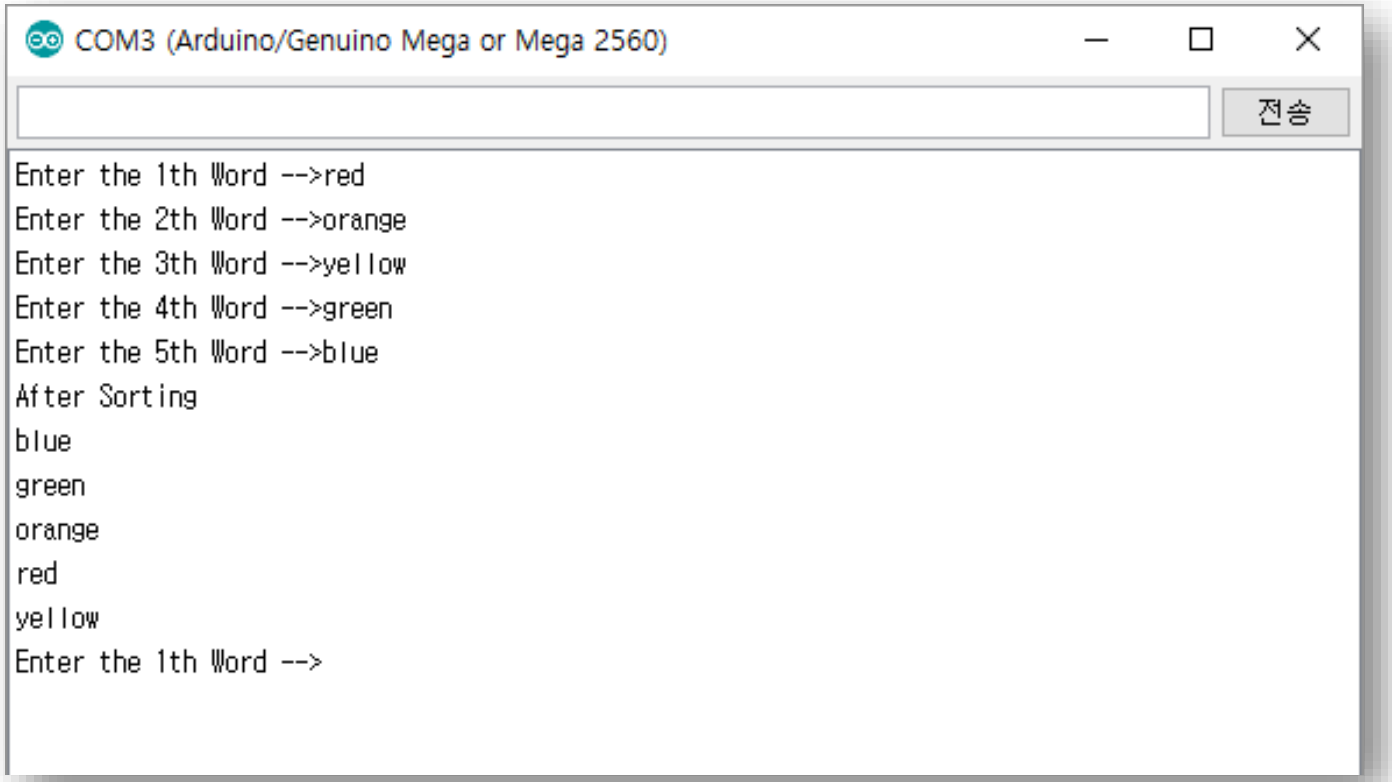
HW.4-1

Homework

❖ **Write a Sketch code that receives 5 words input, then print them out in ascending order as you can see in the following figure.**

❖ **Submission**

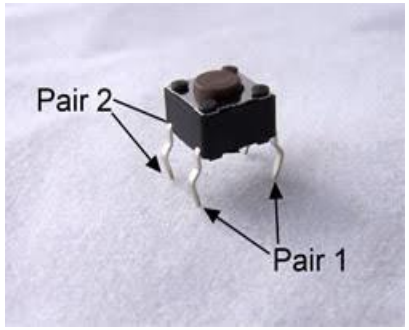
1. Insert your Sketch code to PLMS HW 4-1.
2. Capture a screenshot of the Serial Monitor and upload it to also PLMS.

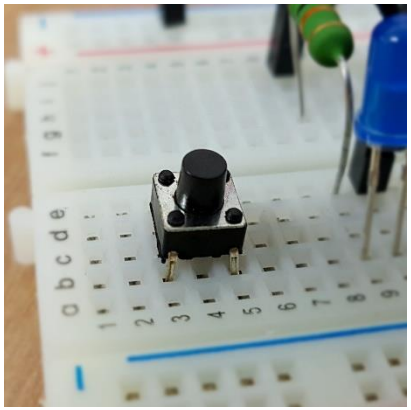


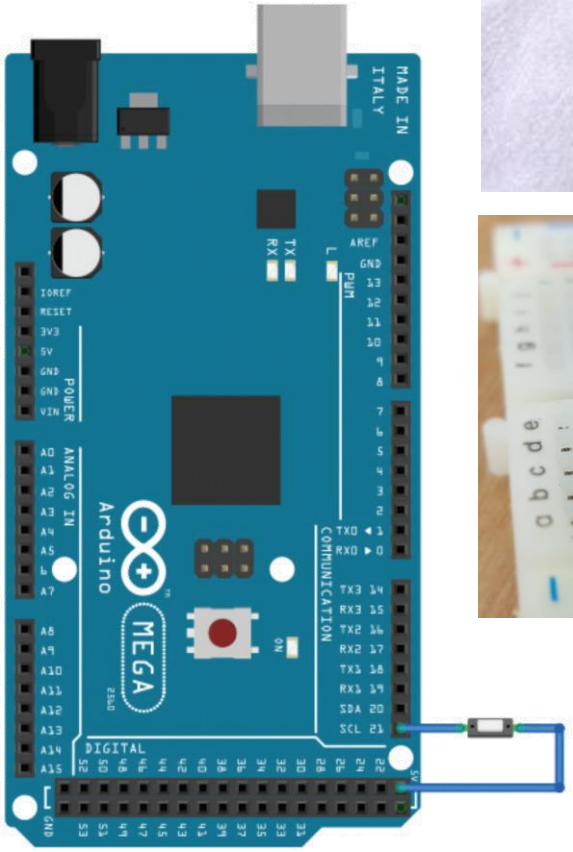
DIGITAL INPUT - BUTTON

Button Connecting – Basic Circuit

❖ 버튼 on(push) : 1 (Vcc)



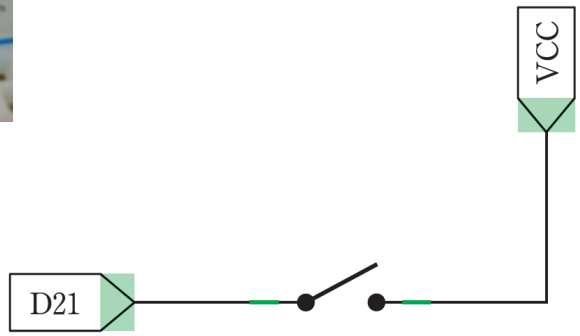




❖ 버튼 Off : 0 ?

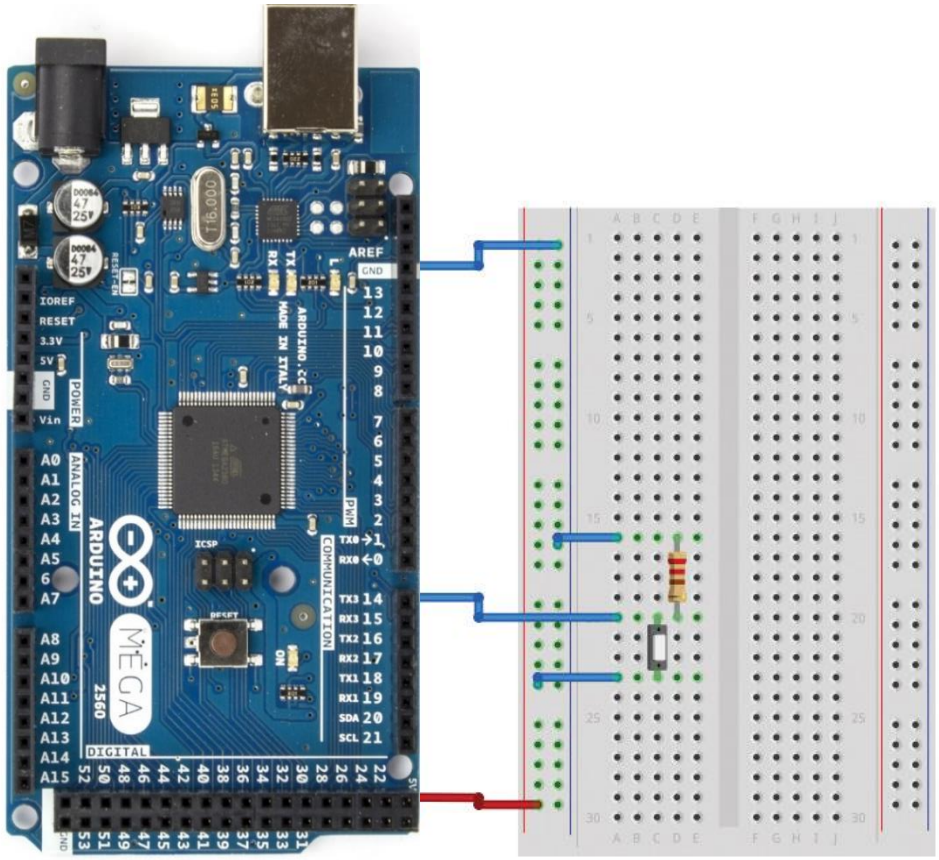
❖ Floating Input Problem

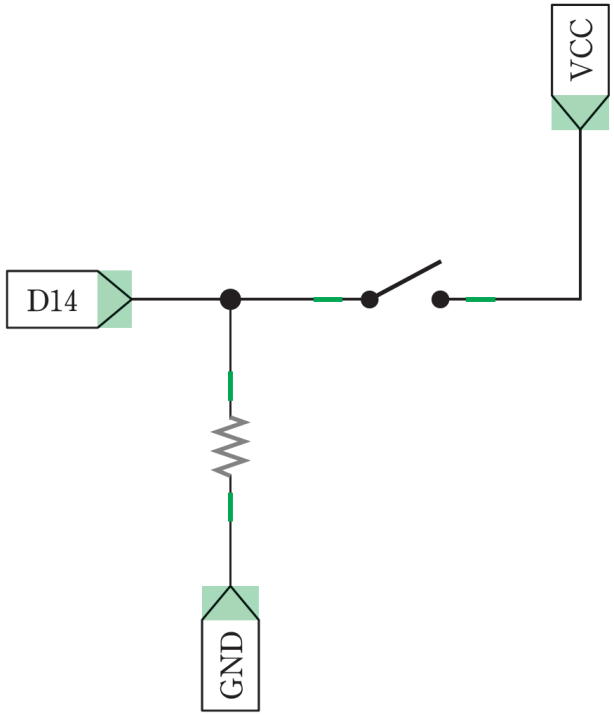
- Digital 입력 핀에 아무런 회로가 연결 되지 않은 상황
- 주변 핀의 상태나 정전기 등에 의해 임의의 값이 입력될 수 있음
- 버튼을 사용할 때는 회로가 오픈 되는 경우를 피해야 한다
- 풀다운 저항의 사용



Button Connecting – Pull-down Resistor

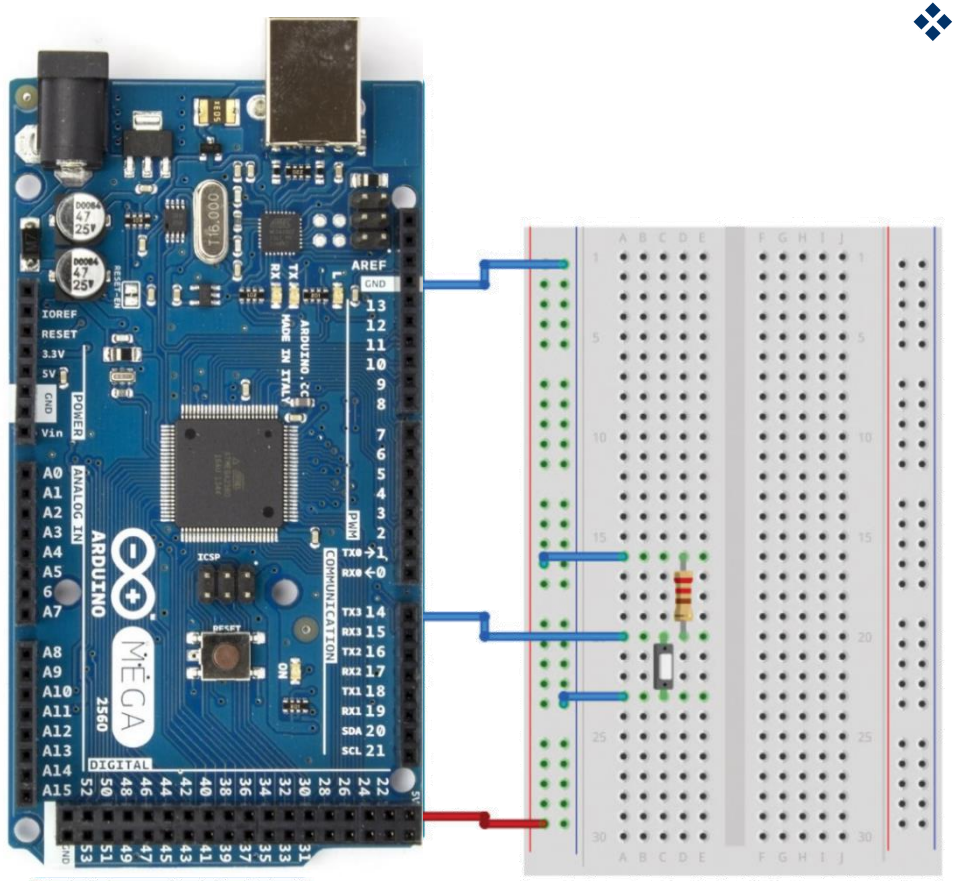
- Pull-down
- Button on(push) : 1 (Vcc)
- Button off : 0 (Gnd)



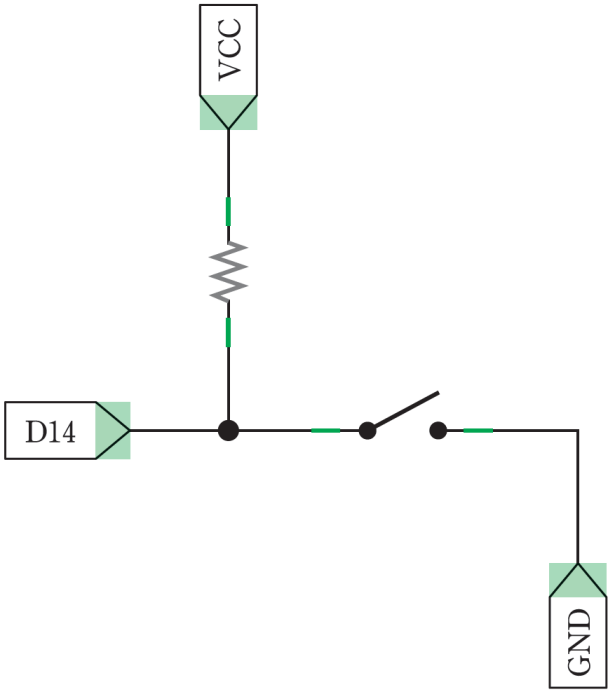


Button Connecting – Pull-up Resistor

- Pull-up



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



Button Connecting

PUll-up/Pull-down resistor	Unpressed	Pressed
Not used	Floating (neither 1 nor 0)	1
Pull-down resistor	0	1
Pull-up resistor	1	0

Pull-up / Pull-down Buttons

❖ Experiment Sequence

① Constructing a Circuit using Pull-up / Pull-down buttons

② Printing Out the Button State (Sketch 6-1, Textbook pp. 103)

③ Constructing a Circuit using the Button Connected to an Internal Pull-up Resistor

④ Printing Out the State of the Button connected to an Internal Pull-up resistor (Sketch 6-2, Textbook pp.106)

⑤ Constructing a Circuit using LEDs and Buttons

⑥ Indicating a Button State using LEDs (Sketch 6-3, Textbook pp. 107~108)

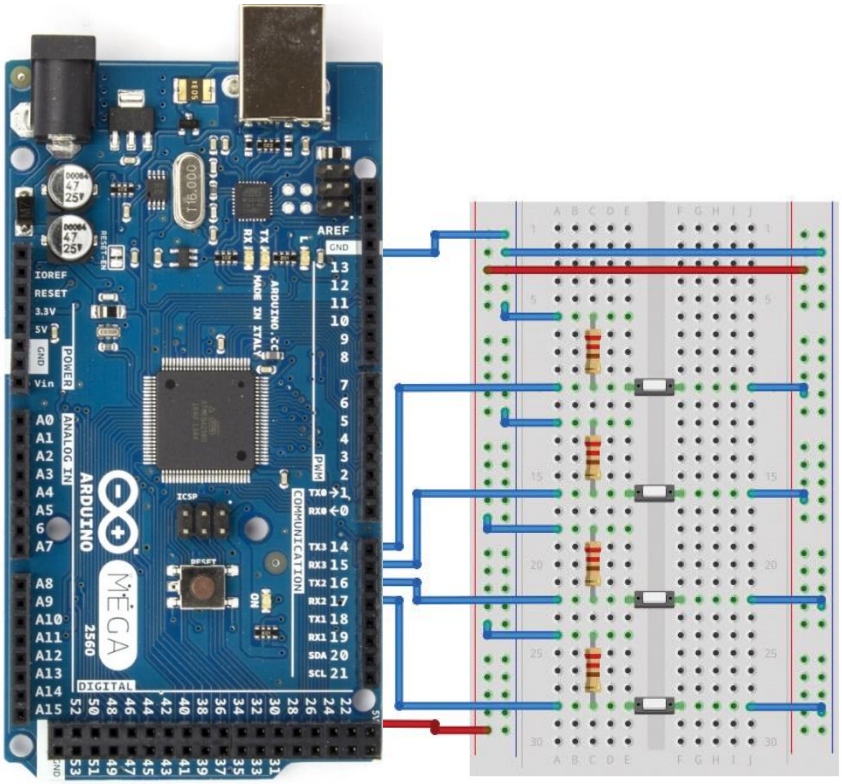
⑦ Turning on and off LED using Buttons (Sketch 6-4, Textbook pp. 109)

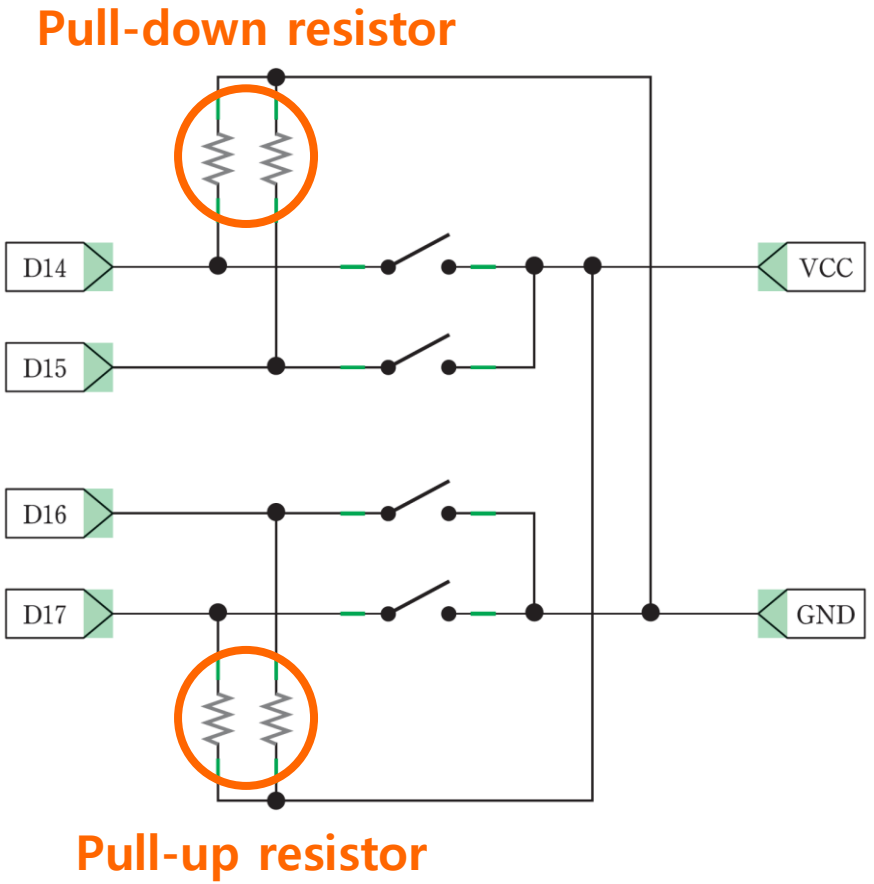
⑧ Counting the Number of Button Presses (Sketch 6-5, Textbook pp. 111~112)

⑨ Check the Result

Lab.4-2

① Constructing a Circuit using Pull-up / Pull-down Buttons

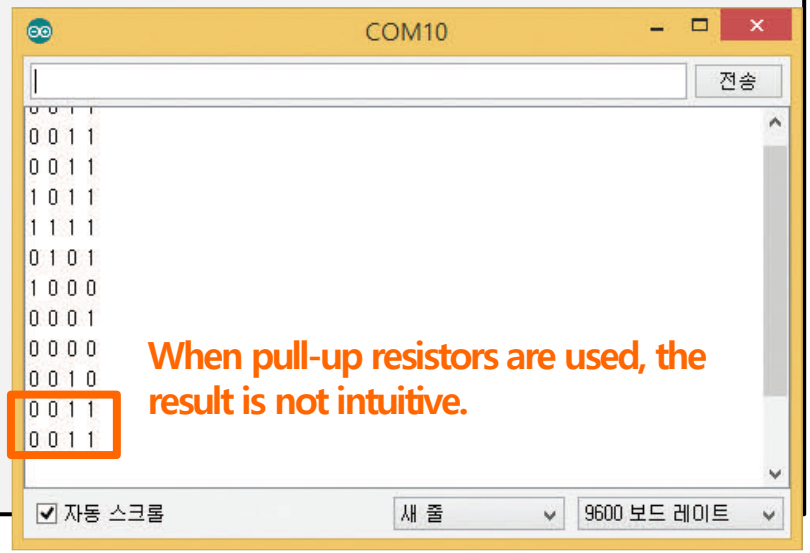




② Printing Out the Button State

Sketch 6-1, Textbook pp. 103

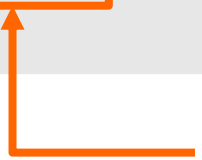
```
int buttons[] = {14, 15, 16, 17}; // Pins connected to buttons

void setup() {
  Serial.begin(9600); // Open the serial port at 9600 bps
  for (int i = 0; i < 4; i++) { // Sets the Digital Pins as input
    pinMode(buttons[i], INPUT);
  }
}
void loop() {
  for (int i = 0; i < 4; i++) {
    Serial.print(digitalRead(buttons[i])); // Prints out the button
    Serial.print(" ");                     // state
  }
  Serial.println(); // New line
  delay(1000);
}
```



Functions for Data Input

void pinMode(uint8_t pin, uint8_t mode)

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

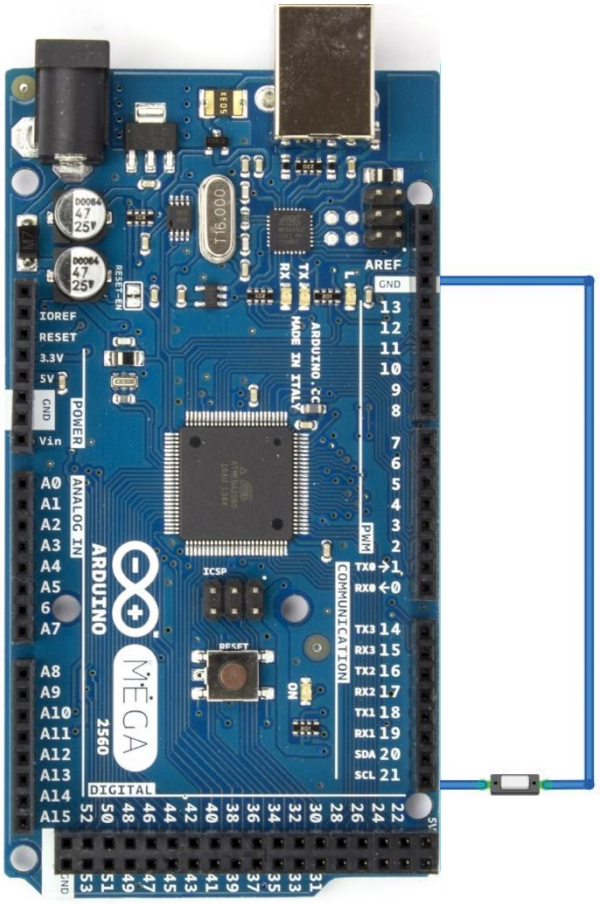


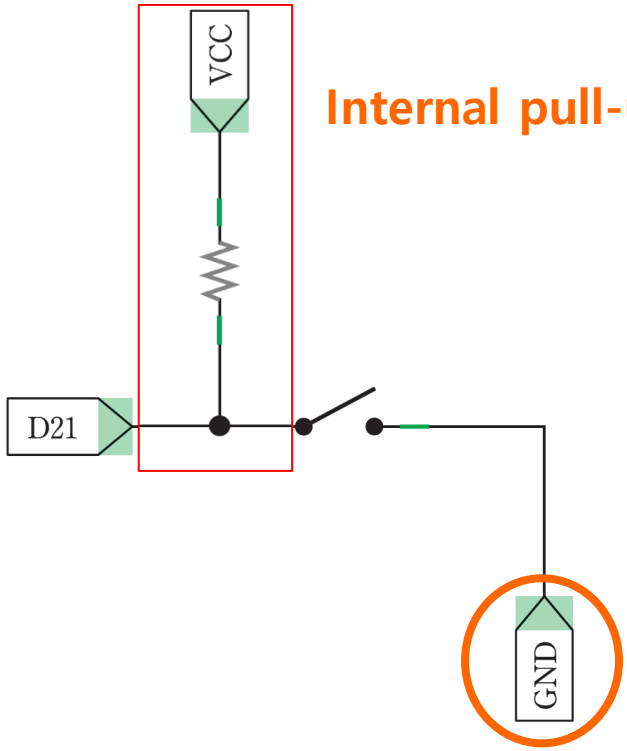
Arduino has internal pull-up resistors inside microcontroller

int digitalRead(uint8_t pin)

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

③ Constructing a Circuit using a Button Connected to an Internal Pull-up Resistors





Internal pull-up resistor

If an internal pull-up resistor is not used, a button should be connected to VCC

❖ In Arduino, each pin has an internal pull-up resistor

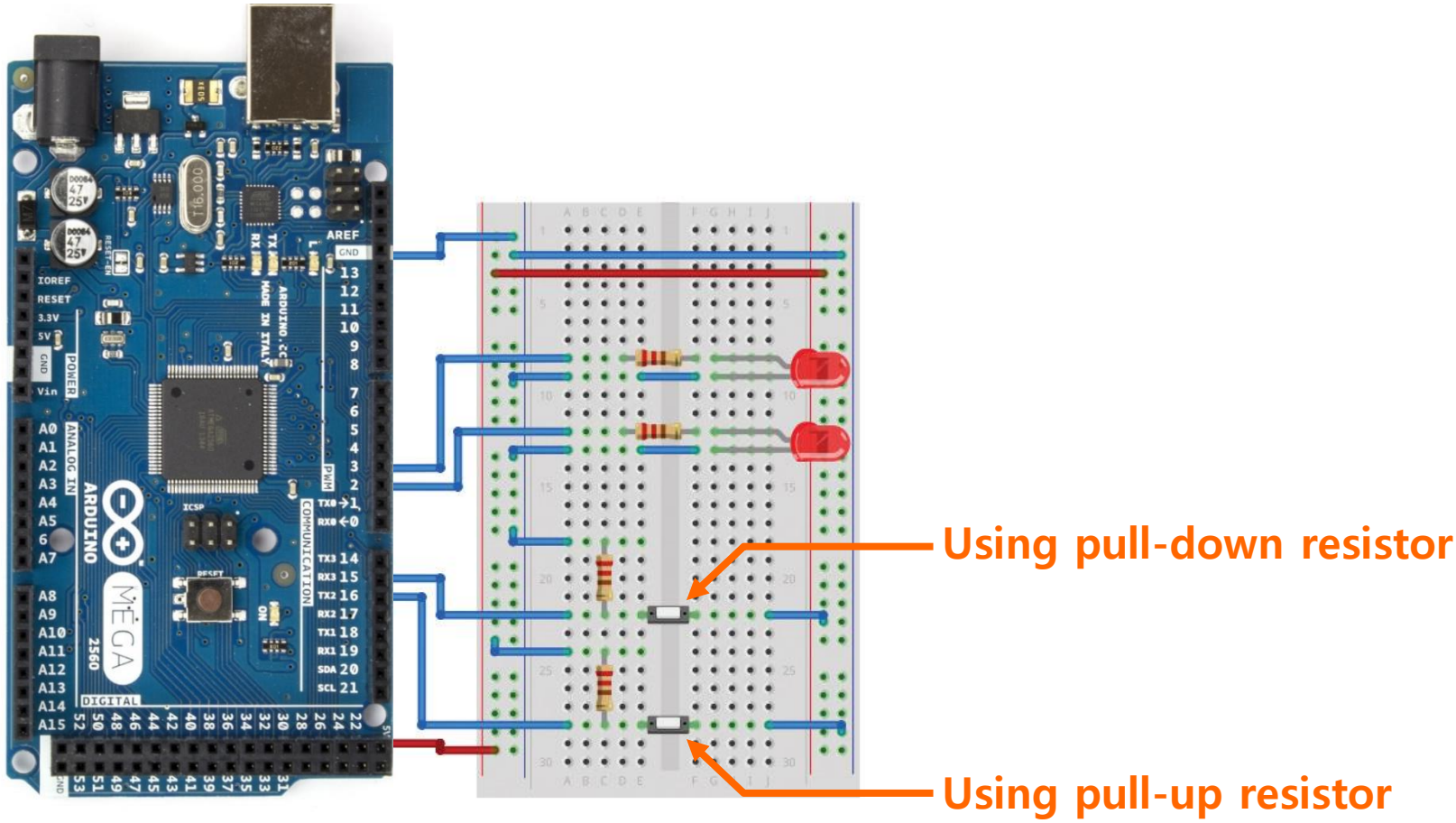
Lab.4-2

④ Printing Out the State of the Button Connected to an Internal Pull-up resistor

Sketch 6-2, Textbook pp. 106

```
int button = 21; // Pin connected to a button

void setup() {
  Serial.begin(9600); // Open the serial port at 9600 bps
pinMode(button, INPUT_PULLUP); // Sets the Digital Pins as input
}

void loop() {
  Serial.println(digitalRead(button)); // Prints out the button state
  delay(1000);
}
```

⑤ Constucting a Circuit using LEDs and Buttons



⑥ Indicating a Button State using LEDs

Sketch 6-3, Textbook pp. 107~108

```
int pins_LED[] = {2, 3}; // Pins connected to LEDs
// Pins connected to buttons (15 : pull-down resistor, 16 : pull-up resistor)
int pins_button[] = {15, 16};

void setup() {
  Serial.begin(9600); // Open the serial port at 9600 bps
  for (int i = 0; i < 2; i++) {
    pinMode(pins_button[i], INPUT); // Sets the Digital Pins as input
    pinMode(pins_LED[i], OUTPUT); // Sets the Digital Pins as output
  }
}

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

⑦ Turning on and off LED using Buttons

- If a pull-up resistor is used, the state of unpressed button is HIGH, not intuitive
- Creating an array of the LED patterns for button state

```
boolean on_off_pattern[2][2] = {
   {false, true},   // Using a pull-down resistor
   {true, false}    // Using a pull-up resistor
};
```

- Controlling a LED using pre-defined LED pattern values for button state

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

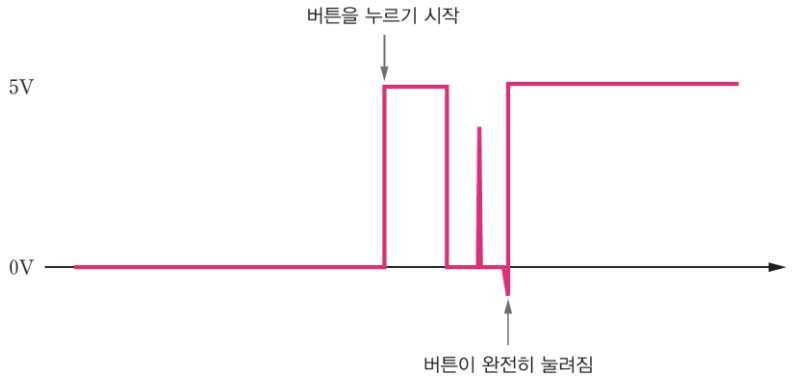
⑦ Turning on and off LED using Buttons

Sketch 6-4, Textbook pp. 109

```
int pins_LED[] = {2, 3}; // Pins connected to LED
// Pins connected to buttons (15 : Pull-down resistor, 16 : Pull-up resistor)
int pins_button[] = {15, 16};

boolean on_off_pattern[2][2] = { // LED lighting patterns
  {false, true}, // Using pull-down resistor
  {true, false} // Using pull-up resistor
};

void setup() {
  Serial.begin(9600); // Open the serial port at 9600 bps
  for (int i = 0; i < 2; i++) {
    pinMode(pins_button[i], INPUT); // Sets the Digital Pins as input
    pinMode(pins_LED[i], OUTPUT); // Sets the Digital Pins as output
  }
}

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

⑧ Counting the Number of Button Presses

- If a button is pushed, the button counter increases. If a button is pushed and held continuously, the button counter will keep increasing
- Increase the button counter only if previous state of the button (state_previous) is unpressed state and current state of the button (state_current)is pressed state
- Bouncing / Chattering: A button state changes repeatedly between ON and OFF due to mechanical vibration
  - Debouncing : removing a chattering



⑧ Counting the Number of Button Presses

Sketch 6-5, Textbook pp. 111~112

```
int pin_button = 15; // Pins connected to buttons
boolean state_previous = false; // Previous state of a button
boolean state_current; // Current state of a button
int count = 0; // The number of Button pressed

void setup() {
  Serial.begin(9600); // Open the serial port at 9600 bps
  pinMode(pin_button, INPUT); //} Sets the Digital Pins as input


void loop() {
  state_current = digitalRead(pin_button); // Reads the button state
  if (state_current) { // Button is pressed
    if (state_previous == false) { // Compares to previous state
      count++; // Increases the button counter
      state_previous = true;
      Serial.println(count);
    }
    // delay(50); // Debouncing
  }
  else {
    state_previous = false;
  }
}
```

⑨ Check the Result

1. Show 5 Sketch code you have written
2. Show an execution of the Lab 6-4 Sketch
3. Answer the TA's questions and insert the score result to PLMS LAB 4-2

HW.4-2

Homework

❖ Exercise 6.3

- Connect 4 LEDs to digital pin 2 ~5, and connect a button to digital pin 14 using pull-down resistor.
- LEDs should be blinked with 0.5 second interval in the forward direction, 1->2->3->4->1 ....
- If the button is pressed, LED bliking sequence should be inverted.
- In other words, if the button is pressed, LED bliking sequence is changed into backward direction, 4->3->2->1->4 ... Notice that in the beginning, the LED bliking sequence is the forward direction.

❖ Submission

1. Insert your Sketch code to PLMS HW 4-2
2. Submit a link to the video recording the result of your work.

Pattern	Digital Pin			
	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