

Robotics

Digital input / Serial communication

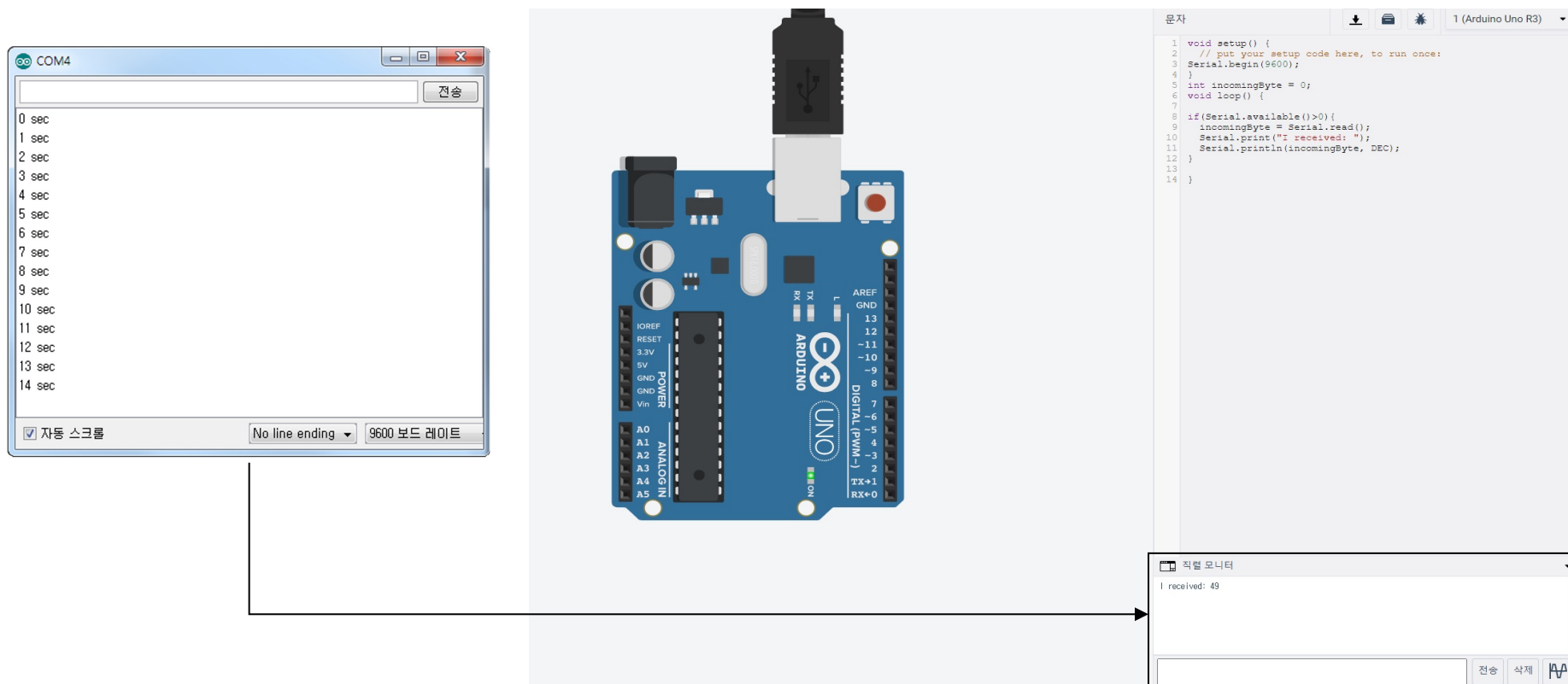
School of Computing, Gachon University

Youngmin Oh



Basics

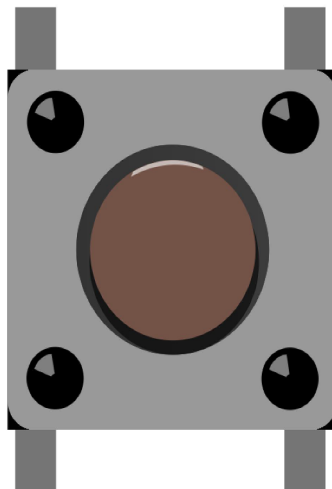
- Arduino environment is embedded with the built-in serial monitor to communicate with an Arduino board.


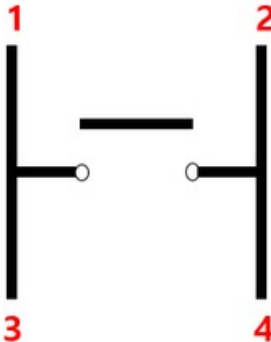
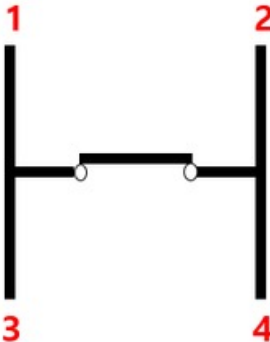


Basics

●Push button (Switch)

- The push button is a component that connects two points in a circuit when you press it.



		
푸시버튼 구성요소	푸시버튼을 누르지 않을 때	푸시버튼을 눌렀을 때

Functions

● Serial.begin()

- Sets the data rate in bits per second for serial data transmission
- Syntax
 - Serial.begin(speed)
 - Speed : 300, 600, 1200, 2400, 4800, **9600**, 14400, 19200, 28800, 38400, 57600, or 115200
- Example

```
void setup() {  
    Serial.begin(9600); // opens serial port, sets data rate to 9600 bps  
}
```

Functions [cont.]

● **Serial.print()**

- Prints data to the serial port as human-readable ASCII text
- Syntax
 - `Serial.print(val)`
 - `Serial.print(val, format)`
 - `val`: the value to print - any data type
 - `format`: specifies the number base (for integral data types) or number of decimal places (for floating point types)

Functions [cont.]

- Example

`Serial.print(78)` gives "78"

`Serial.print(1.23456)` gives "1.23"

`Serial.print('N')` gives "N"

`Serial.print("Hello world.")` gives "Hello world."

`Serial.print(78, BIN)` gives "1001110"

`Serial.print(78, OCT)` gives "116"

`Serial.print(78, DEC)` gives "78"

`Serial.print(78, HEX)` gives "4E"

`Serial.println(1.23456, 0)` gives "1"

`Serial.println(1.23456, 2)` gives "1.23"

`Serial.println(1.23456, 4)` gives "1.2346"

ASCII Code

Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char
0	0	000	NUL (null)	32	20	040	Space	64	40	100	@	96	60	140	`	128	Ç	161	í
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	a	129	ù	162	ó
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	b	130	é	163	û
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	c	131	á	164	ü
4	4	004	EOT (end of transmission)	36	24	044	\$	68	44	104	D	100	64	144	d	132	ä	165	ñ
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e	133	å	166	°
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f	134	æ	167	²
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g	135	ç	168	³
8	8	010	BS (backspace)	40	28	050	(72	48	110	H	104	68	150	h	136	è	169	´
9	9	011	TAB (horizontal tab)	41	29	051)	73	49	111	I	105	69	151	i	137	é	170	µ
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	j	138	ê	171	½
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k	139	ë	172	¾
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	l	140	ì	173	¿
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m	141	í	174	«
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	n	142	Ä	175	»
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	o	143	Å	176	•
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	p	144	Ê	177	◊
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	q	145	æ	178	■
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	r	146	Æ	179	▀
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	s	147	ø	180	▁
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	t	148	ö	181	▂
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	u	149	ù	182	▃
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v	150	û	183	▄
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w	151	ü	184	▅
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x	152	ÿ	185	▆
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y	153	ÿ	186	▇
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z	154	Ü	187	█
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[123	7B	173	{	155	ß	188	▉
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174		157	¥	189	▊
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135]	125	7D	175	}	158	₣	190	▋
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	~	159	₧	191	▌
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	DEL	160	à	192	▍

Functions [cont.]

● Serial.write()

- Writes binary data to the serial port. This data is sent as a byte or series of bytes
- Syntax
 - Serial.write(val)
 - val: a value to send as a single byte
- Example

```
void loop() {  
    Serial.write(49);    // send a byte "1" with the value 49  
}
```


Functions [cont.]

● Serial.available()

- Get the number of bytes (characters) available for reading from the serial port
- Syntax
 - Serial.available()
- Example

```
void loop() {  
    if (Serial.available() > 0) {        // send data only when you receive data  
        Serial.print("I received");  
    }  
}
```

Functions [cont.]

● Serial.read()

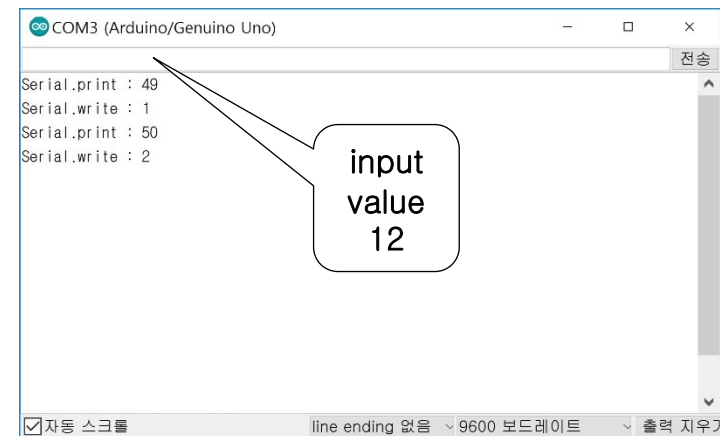
- Reads incoming serial data Syntax
- Syntax
 - Serial.read()
- Example

```
void loop() {
    if (Serial.available() > 0) {        // send data only when you receive data

        int incomingByte=Serial.read(); // read the incoming byte

        Serial.print("Serial.print : "); //
        Serial.println(incomingByte);
        Serial.print("Serial.write : ");
        Serial.write(incomingByte);

    }
}
```



Functions [cont.]

● Serial.parseInt()

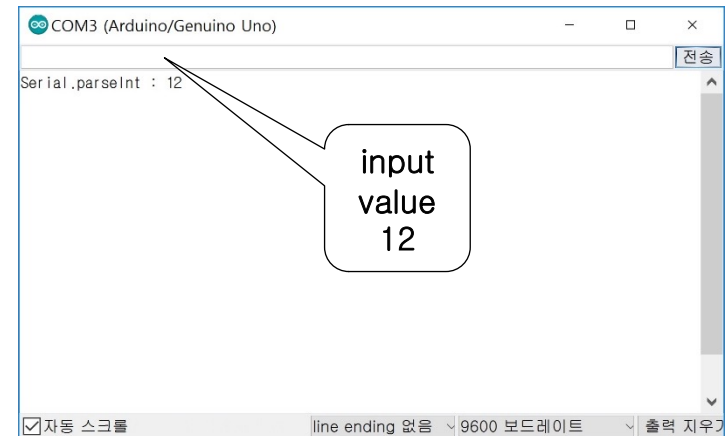
- This function returns the first valid (long) integer number from the serial buffer. Characters that are not integers (or the minus sign) are skipped.

- Syntax

- Serial.parseInt()

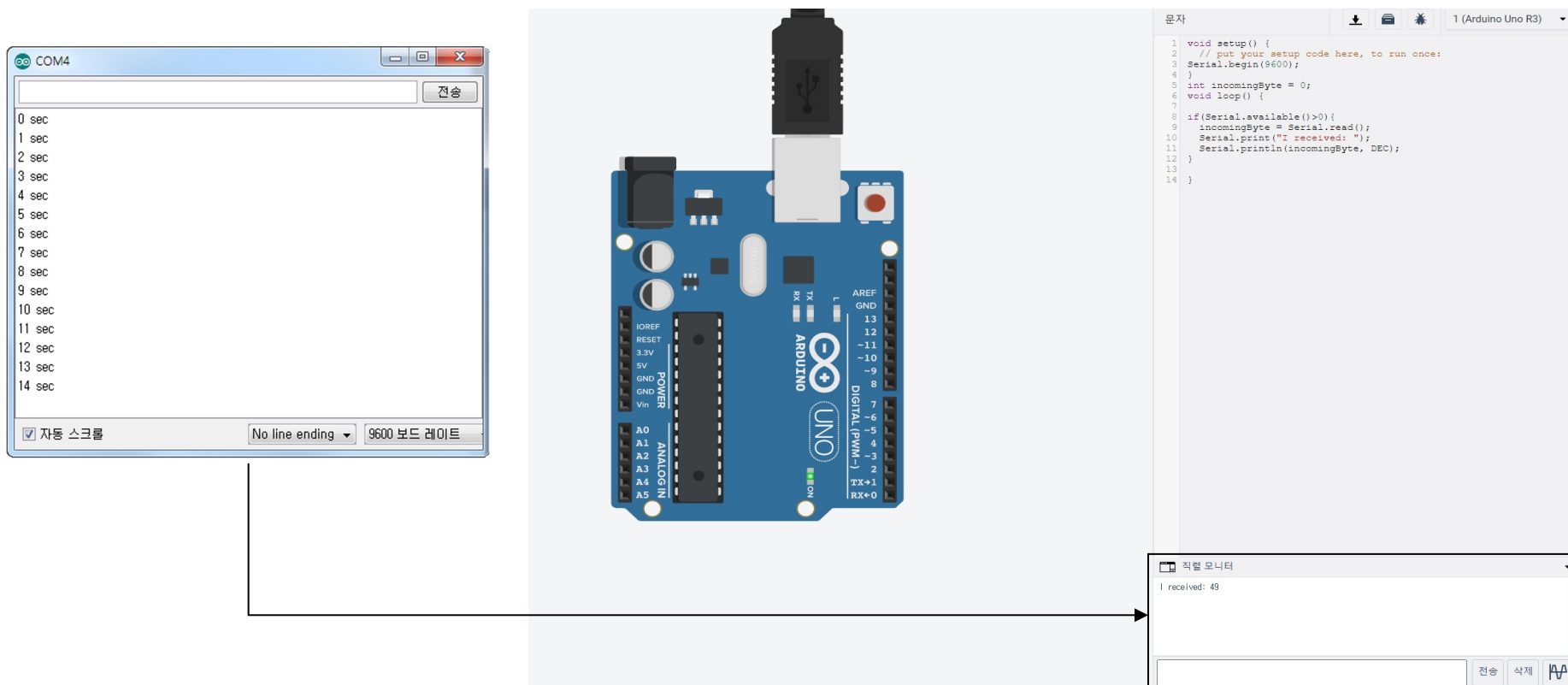
- Example

```
void loop() {
    if (Serial.available() > 0) {      // send data only when you receive data
        int par=Serial.parseInt();
        Serial.print("Serial.parseInt : ");
        Serial.println(par);
    }
}
```



Lab. 1 - Counter

- Use the built-in serial monitor to communicate with an Arduino board.
 - Click the serial monitor button in the toolbar (or use “직렬 모니터” in Tinkercad)



Lab. 1 - Counter [cont.]

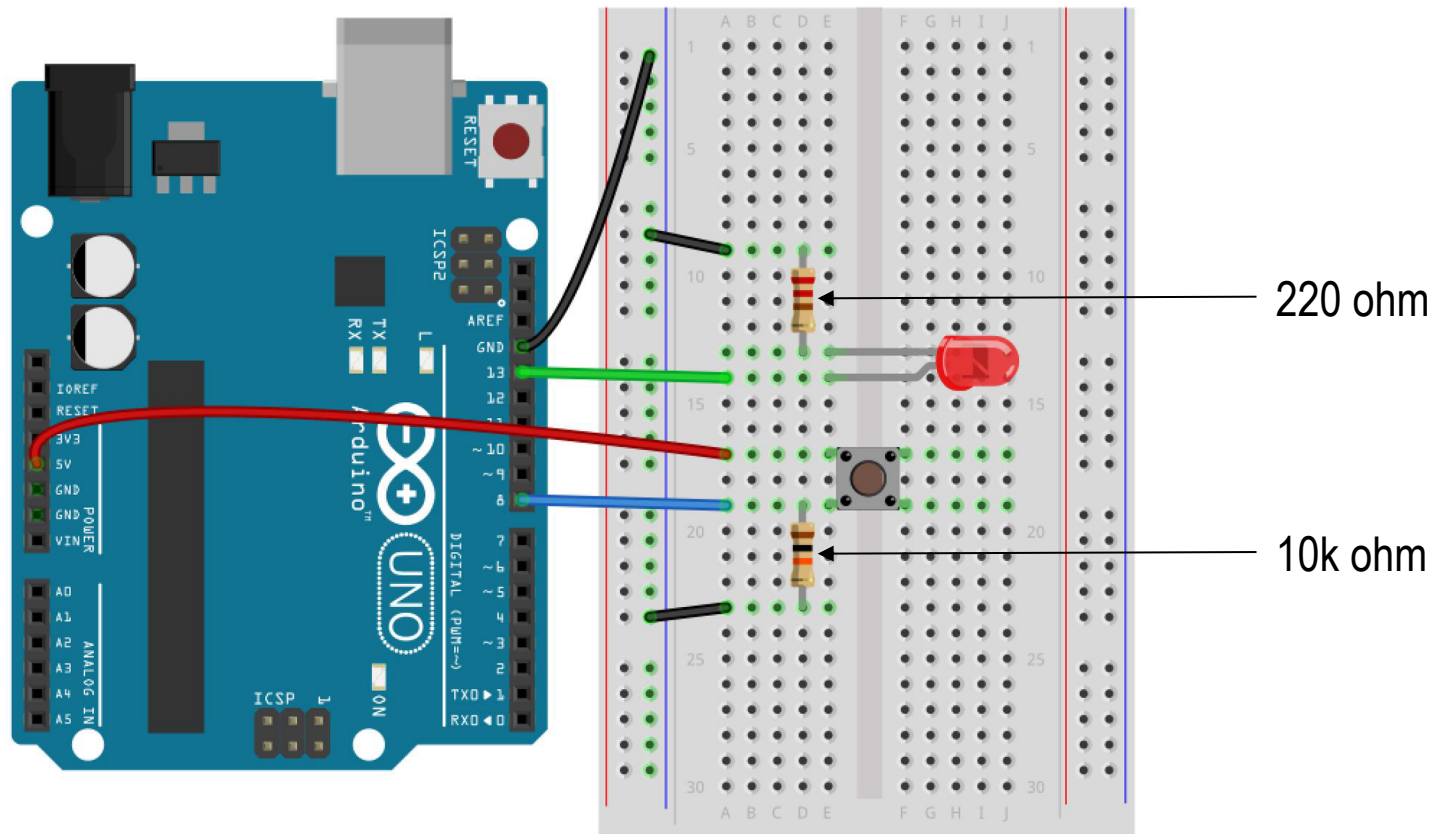
- Write a program to send increasing integer value to the board
 - Initialize serial port with baud rate 9600
 - `Serial.begin(9600);` (in setup function)
 - Send the counter value to the board for every 1sec
 - `Serial.println(count);`
 - `delay(1000);`
 - How to know the received number?
 - Click “Serial monitor” (magnifier icon)
 - or use “직렬 모니터” in Tinkercad

Lab. 2 - PushButton 1

- Write a sketch to turn on a LED when push button is pressed
 - Setup I/O pins
 - e.g. `pinMode(13, OUTPUT); pinMode(8, INPUT);`
 - Read the status of the push button
 - `buttonState = digitalRead(8);` // return HIGH or LOW
 - Set DPin13 as HIGH when the push button is pressed

Lab. 2 - PushButton 1 [cont.]

● Circuit diagram:



Lab. 3 - Counter+PushButton 1

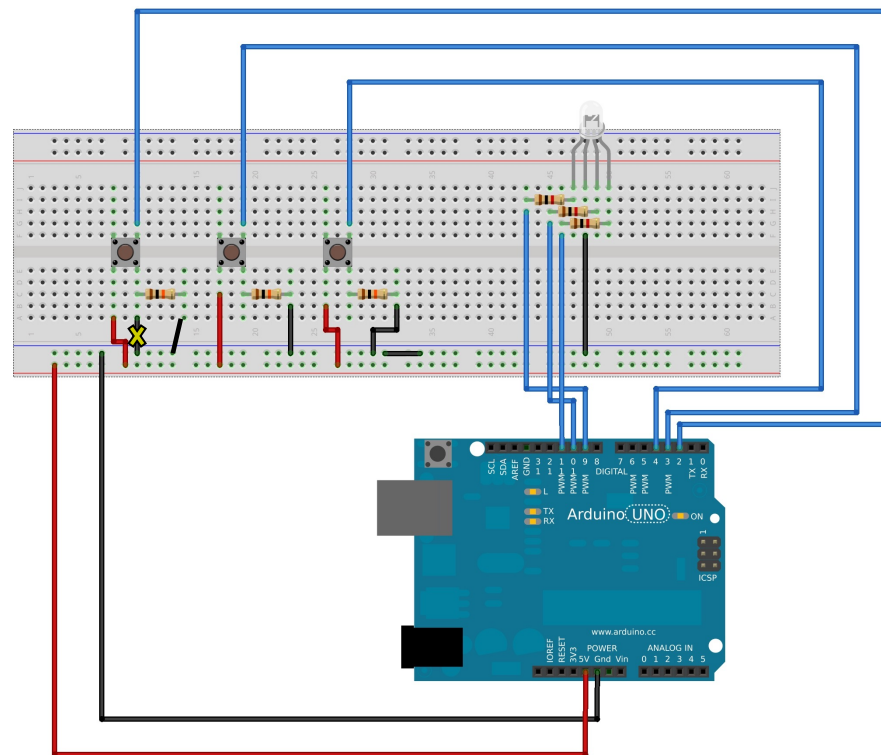
- Write a program to send increasing integer value to the board
 - When the push buttons are pressed, initialize the counter to zero

Lab. 4 - PushButton 2

● Change the color of RGB LED using buttons

- Allocated the red, green and blue to each button
- Change the color when the push buttons are pressed

● Circuit diagram:



Lab. 5 - Data Transmission

- **Blink The LED as many times as transmitted by serial monitor**
 - Transmit the numbers 0 to 9 to Arduino by a serial monitor (or by “직렬 모니터” in Tinkercad)