

# 디지털 IO

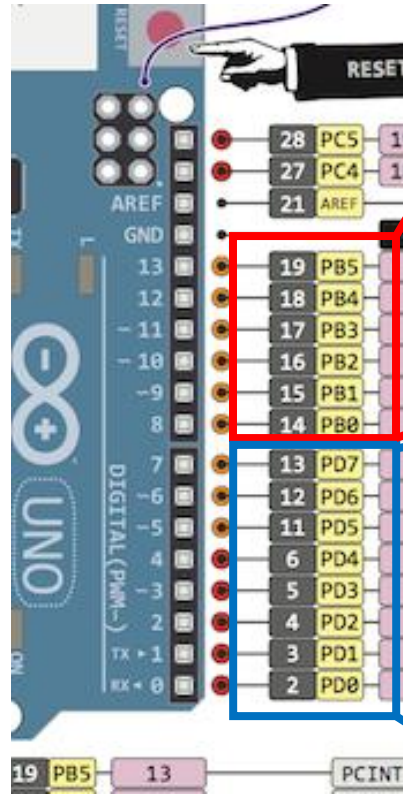
마이크로프로세서 종합 설계. 5주차.

# 목표

- 디지털 IO 실험을 통한 포트 입출력의 이해
- 디지털 IO 포트 관련 레지스터의 이해
- 아두이노 제어를 위한 C언어 '변수'의 이해

# 디지털 IO 포트 관련 레지스터

- Port



### 13.4.2 PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
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

### 13.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDBR
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	

#### 13.4.4 PINB – The Port B Input Pins Address

[illegible]

### 13.4.8 PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0
0xB (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
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

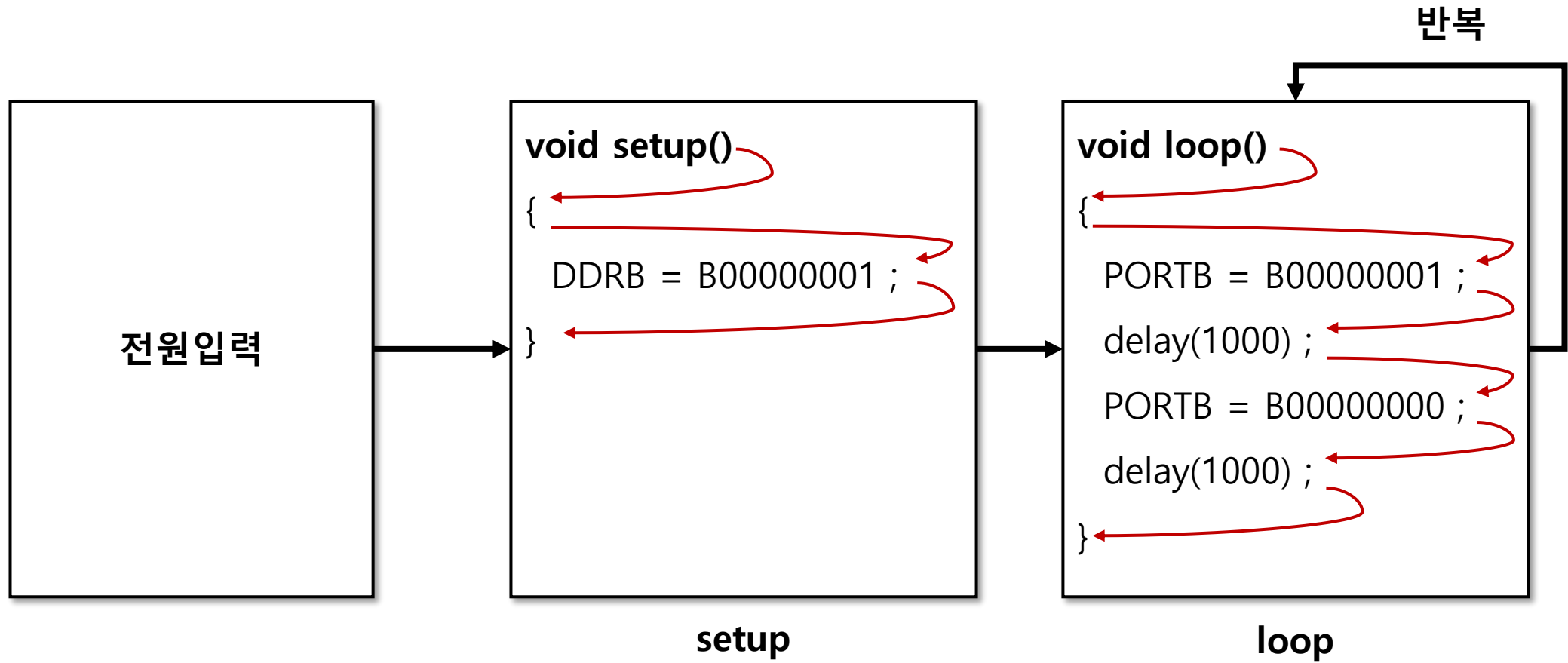
### 13.4.9 DDRD – The Port D Data Direction Register

[illegible]

### 13.4.10 PIND – The Port D Input Pins Address

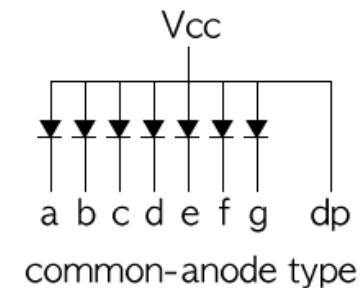
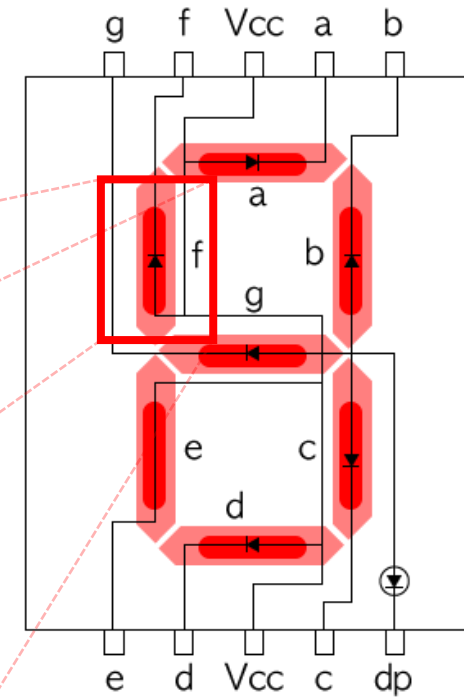
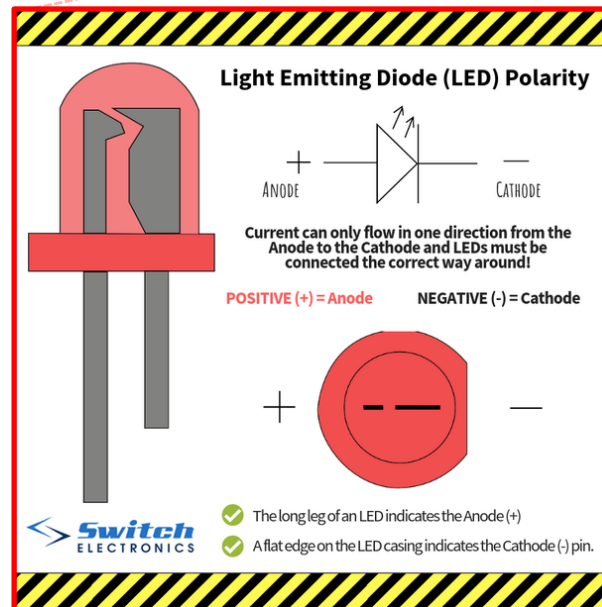
[illegible]

# IO 포트 출력 실험



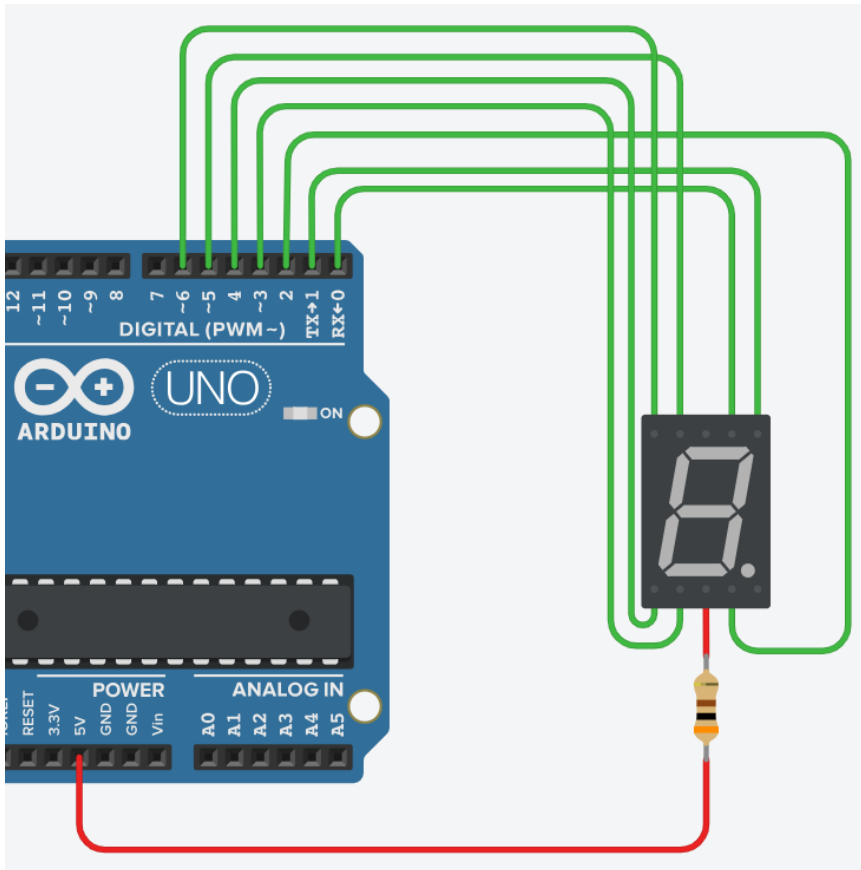
# LED를 이용한 포트 제어 실험

- 7-segment 실험



# LED를 이용한 포트 제어 실험

- 7-segment 실험



a	→	PD0
b	→	PD1
c	→	PD2
d	→	PD3
e	→	PD4
f	→	PD5
g	→	PD6
DP	→	PD7

# LED를 이용한 포트 제어 실험

## • 7-segment 실험

0 = a(0) b(0) c(0) d(0) e(0) f(0) g(1) DP(1)

1 = a(1) b(0) c(0) d(1) e(1) f(1) g(1) DP(1)

2 = a(0) b(0) c(1) d(0) e(0) f(1) g(0) DP(1)

3 = a(0) b(0) c(0) d(0) e(1) f(1) g(0) DP(1)

4 = a(1) b(0) c(0) d(1) e(1) f(0) g(0) DP(1)

5 = a(0) b(1) c(0) d(0) e(1) f(0) g(0) DP(1)

6 = a(0) b(1) c(0) d(0) e(0) f(0) g(0) DP(1)

7 = a(0) b(0) c(0) d(1) e(1) f(0) g(1) DP(1)

8 = a(0) b(0) c(0) d(0) e(0) f(0) g(0) DP(1)

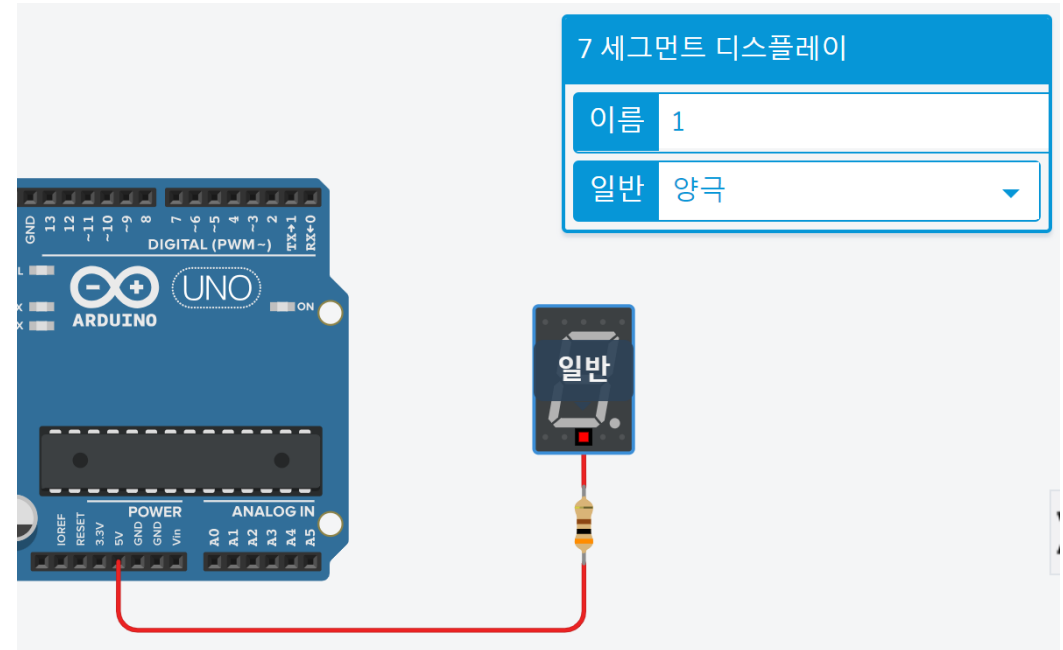
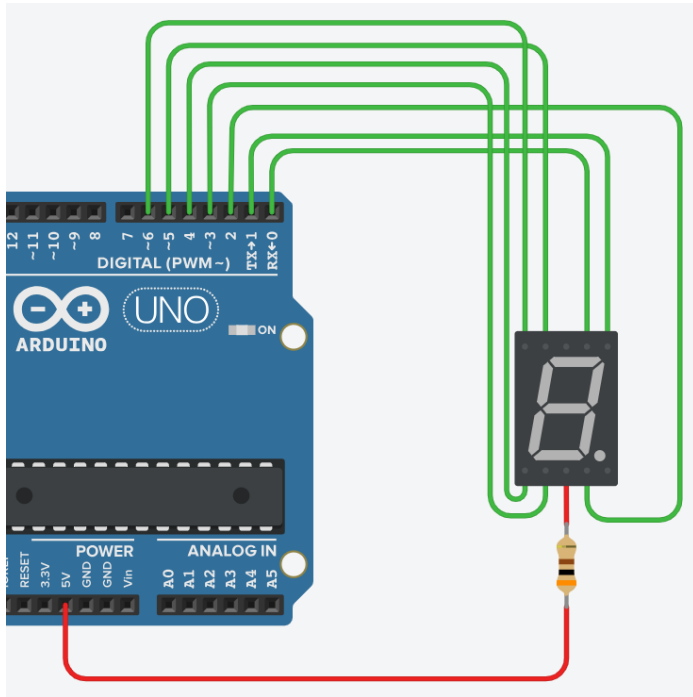
9 = a(0) b(0) c(0) d(0) e(1) f(0) g(0) DP(1)

. = DP(0)

		a	b	c	d	e	f	g	DP
		PD0	PD1	PD2	PD3	PD4	PD5	PD6	PD7
0	→	0	0	0	0	0	0	1	1
1	→	1	0	0	1	1	1	1	1
2	→	0	0	1	0	0	1	0	1
3	→	0	0	0	0	1	1	0	1
4	→	1	0	0	1	1	0	0	1
5	→	0	1	0	0	1	0	0	1
6	→	0	1	0	0	0	0	0	1
7	→	0	0	0	1	1	0	1	1
0	→	0	0	0	0	0	0	0	1
9	→	0	0	0	0	1	0	0	1

# LED를 이용한 포트 제어 실험

- 7-segment 실험





# LED를 이용한 포트 제어 실험

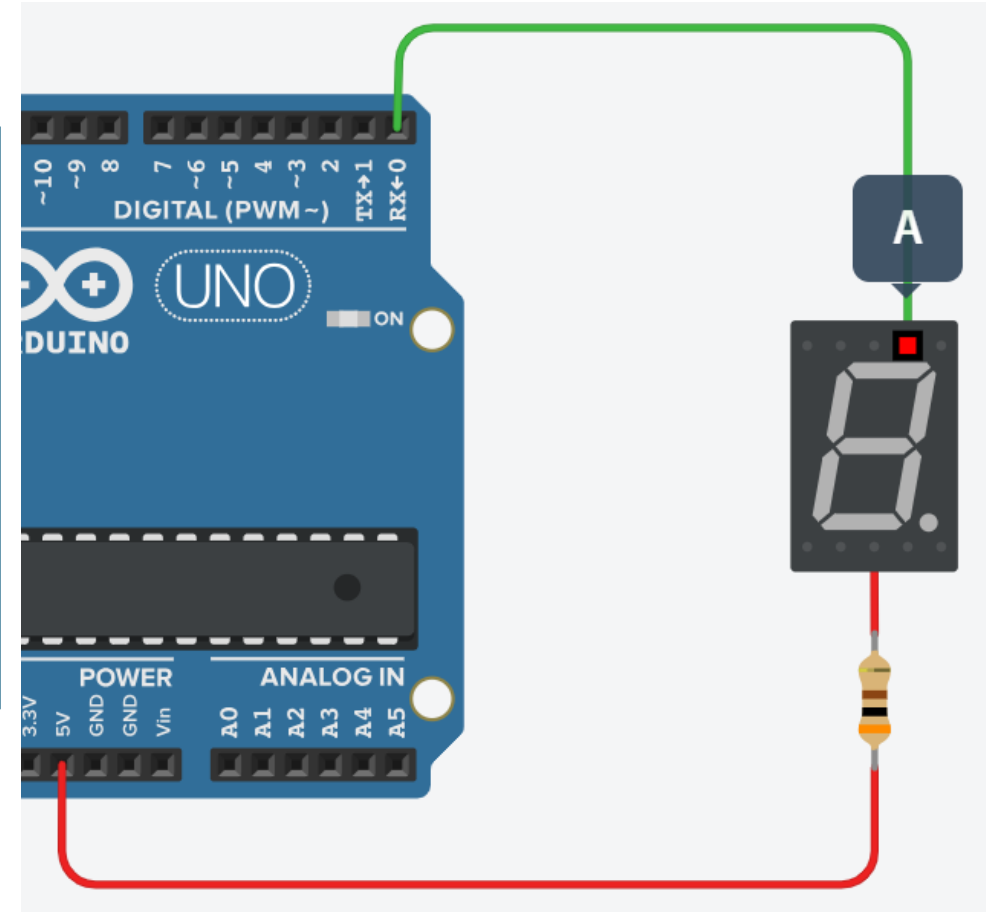
## • 7-segment 실험

```
void setup()
{
  DDRD = B00000001 ;
}

void loop()
{
  PORTD = B00000000 ;
}
```

```
void setup()
{
  DDRD = B00000001 ;
}

void loop()
{
  PORTD = B00000001 ;
}
```



# LED를 이용한 포트 제어 실험

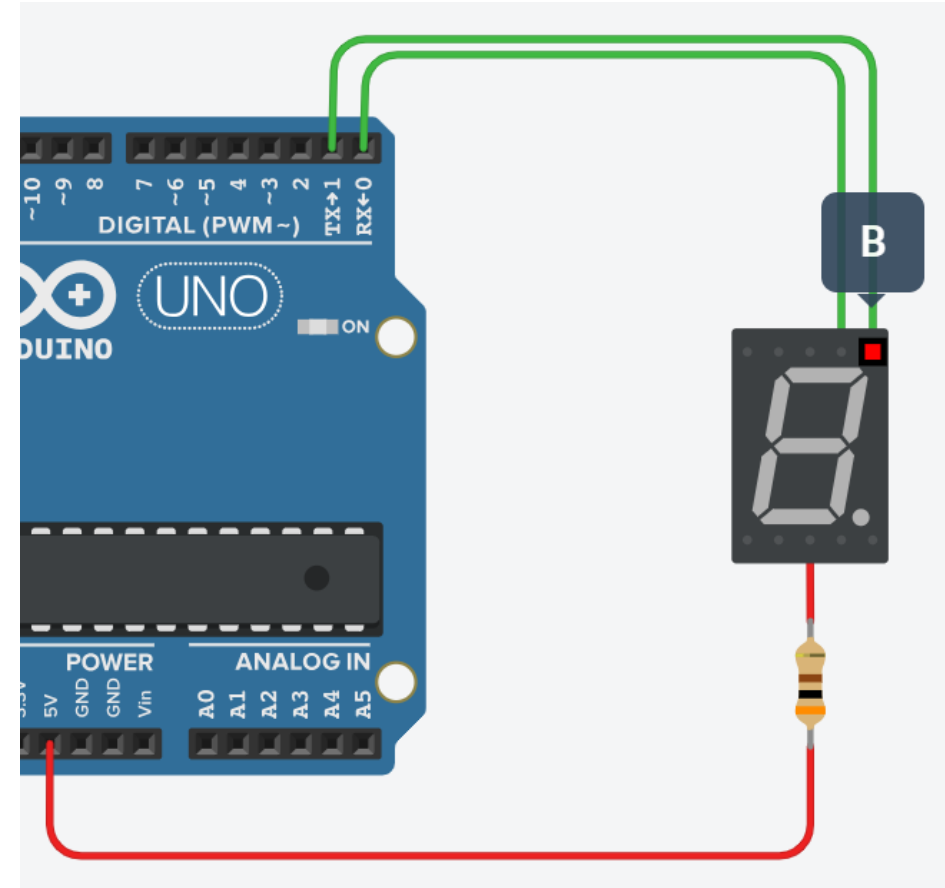
## • 7-segment 실험

```
void setup()
{
  DDRD = B00000011 ;
}

void loop()
{
  PORTD = B00000000 ;
}
```

```
void setup()
{
  DDRD = B00000011 ;
}

void loop()
{
  PORTD = B00000011 ;
}
```



# LED를 이용한 포트 제어 실험

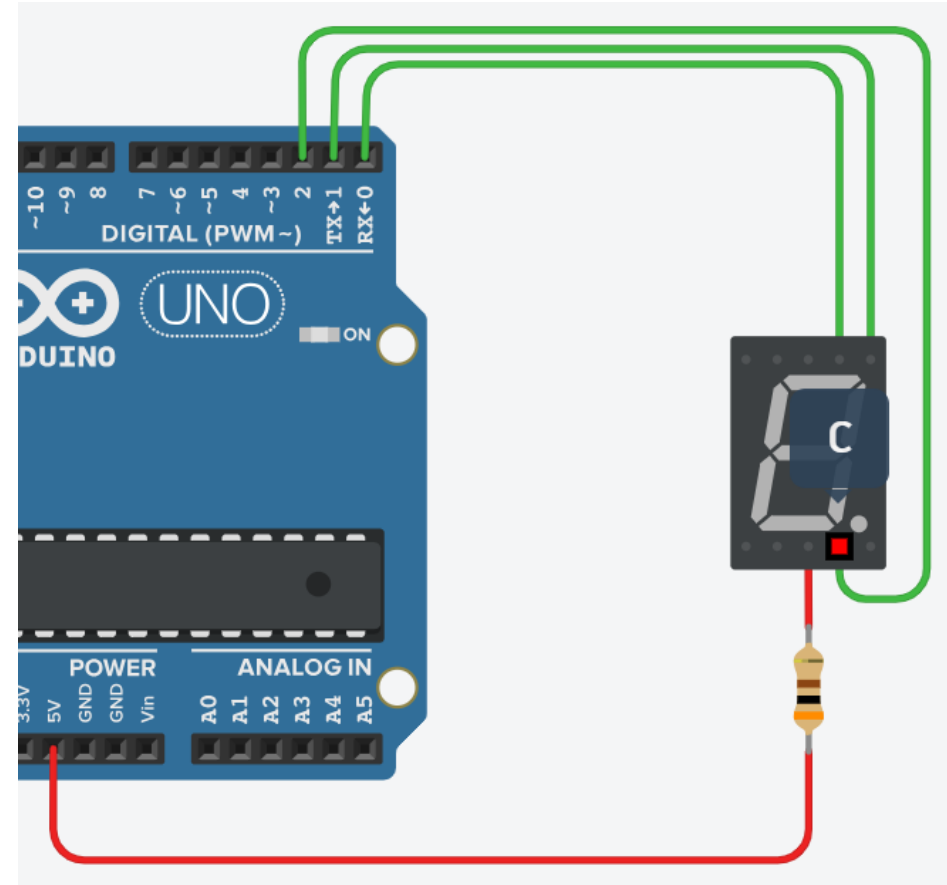
## • 7-segment 실험

```
void setup()
{
  DDRD = B00000111;
}

void loop()
{
  PORTD = B00000000;
}
```

```
void setup()
{
  DDRD = B00000111;
}

void loop()
{
  PORTD = B00000111;
}
```



# LED를 이용한 포트 제어 실험

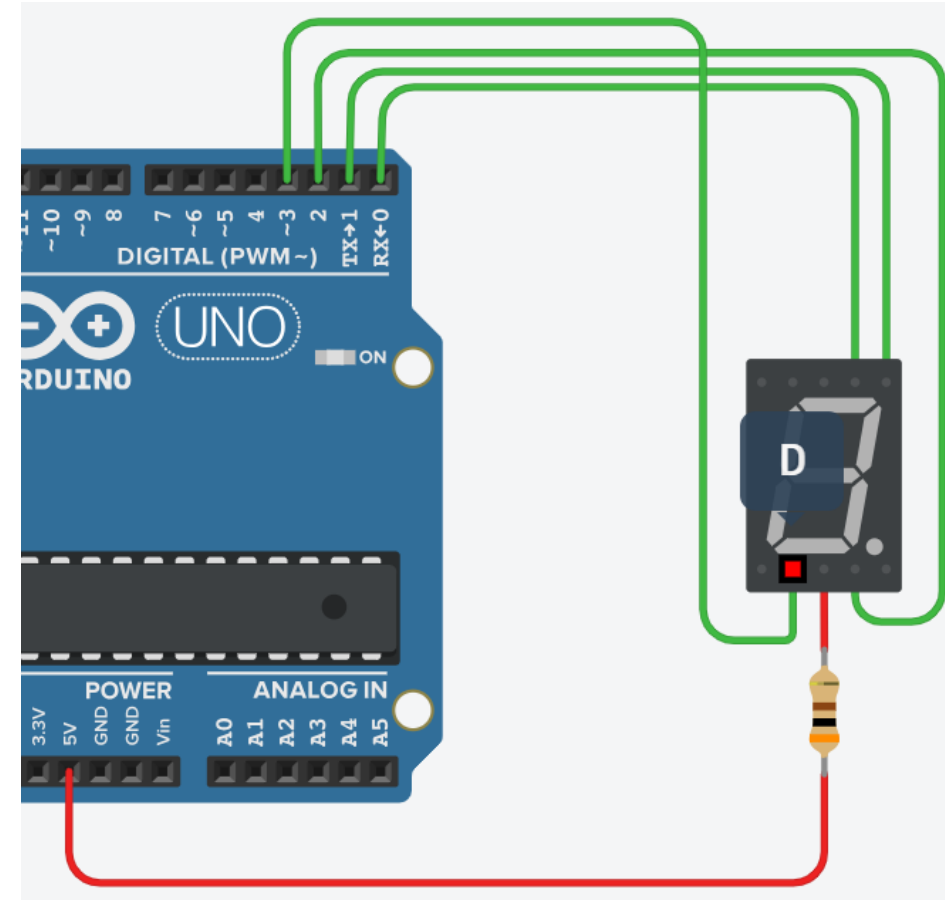
- 7-segment 실험

```
void setup()
{
  DDRD = B00001111 ;
}

void loop()
{
  PORTD = B00000000 ;
}
```

```
void setup()
{
  DDRD = B00001111 ;
}

void loop()
{
  PORTD = B00001111 ;
}
```



# LED를 이용한 포트 제어 실험

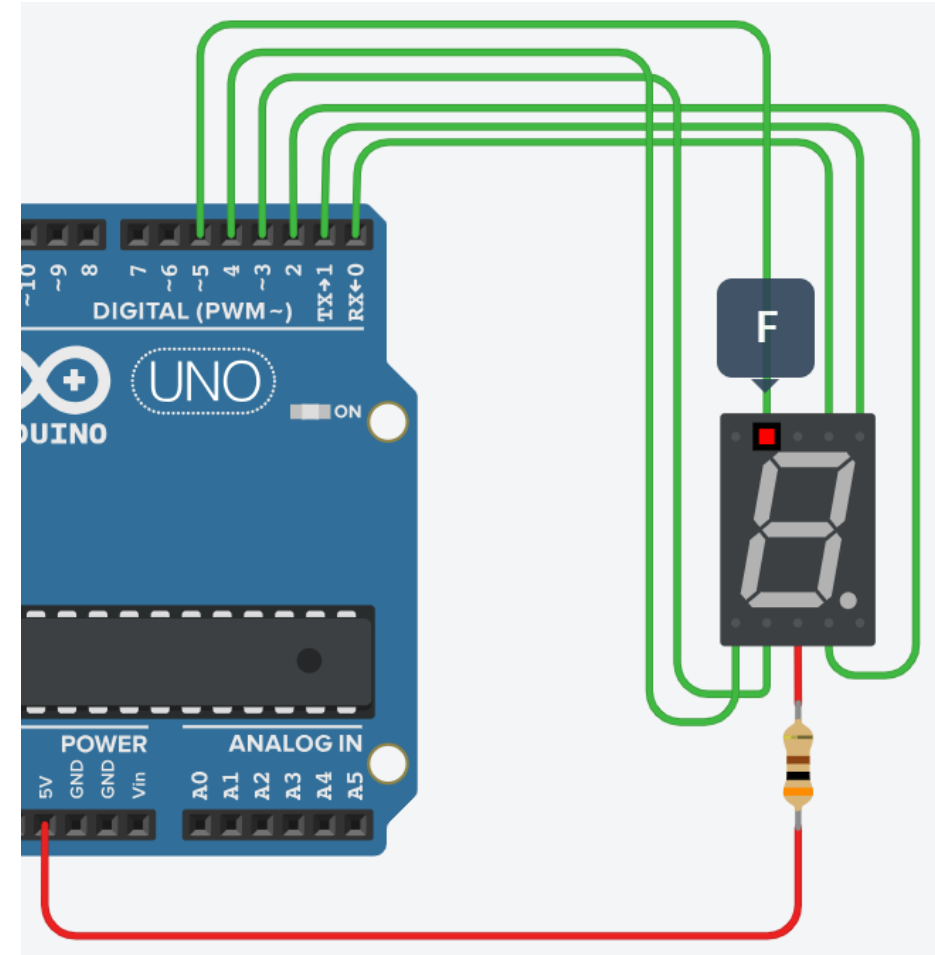
- 7-segment 실험

```
void setup()
{
  DDRD = B00111111;
}

void loop()
{
  PORTD = B0000000;
}
```

```
void setup()
{
  DDRD = B00111111;
}

void loop()
{
  PORTD = B00111111;
}
```



# LED를 이용한 포트 제어 실험

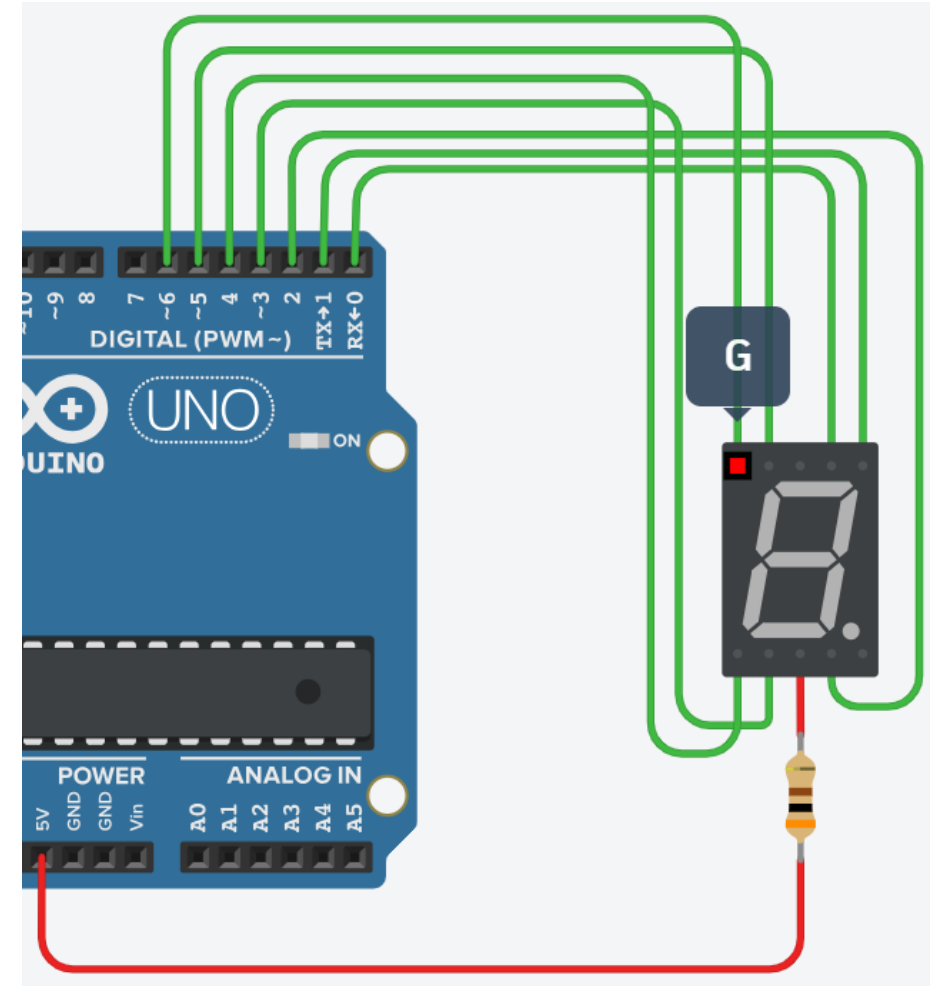
## • 7-segment 실험

```
void setup()
{
  DDRD = B01111111;
}

void loop()
{
  PORTD = B00000000;
}
```

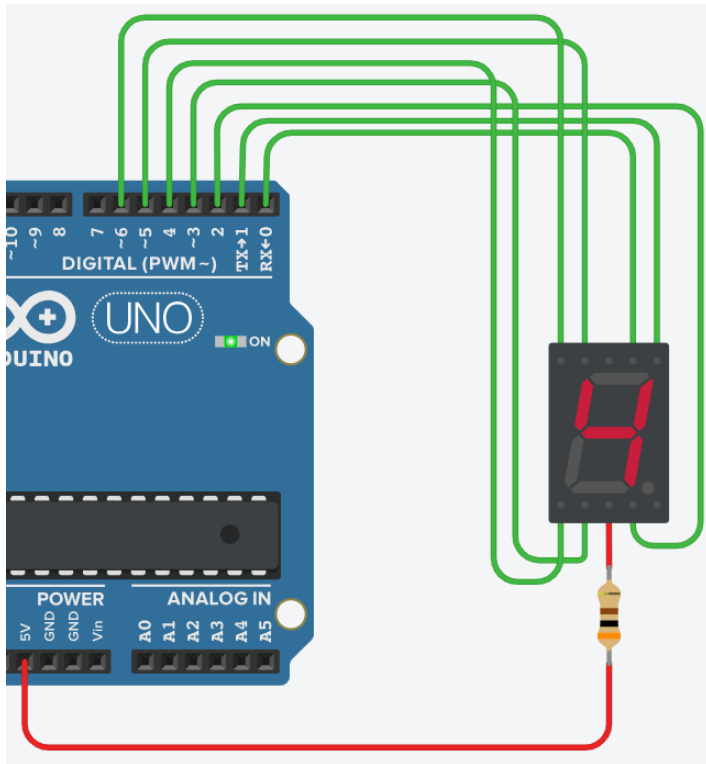
```
void setup()
{
  DDRD = B01111111;
}

void loop()
{
  PORTD = B01111111;
}
```



# LED를 이용한 포트 제어 실험

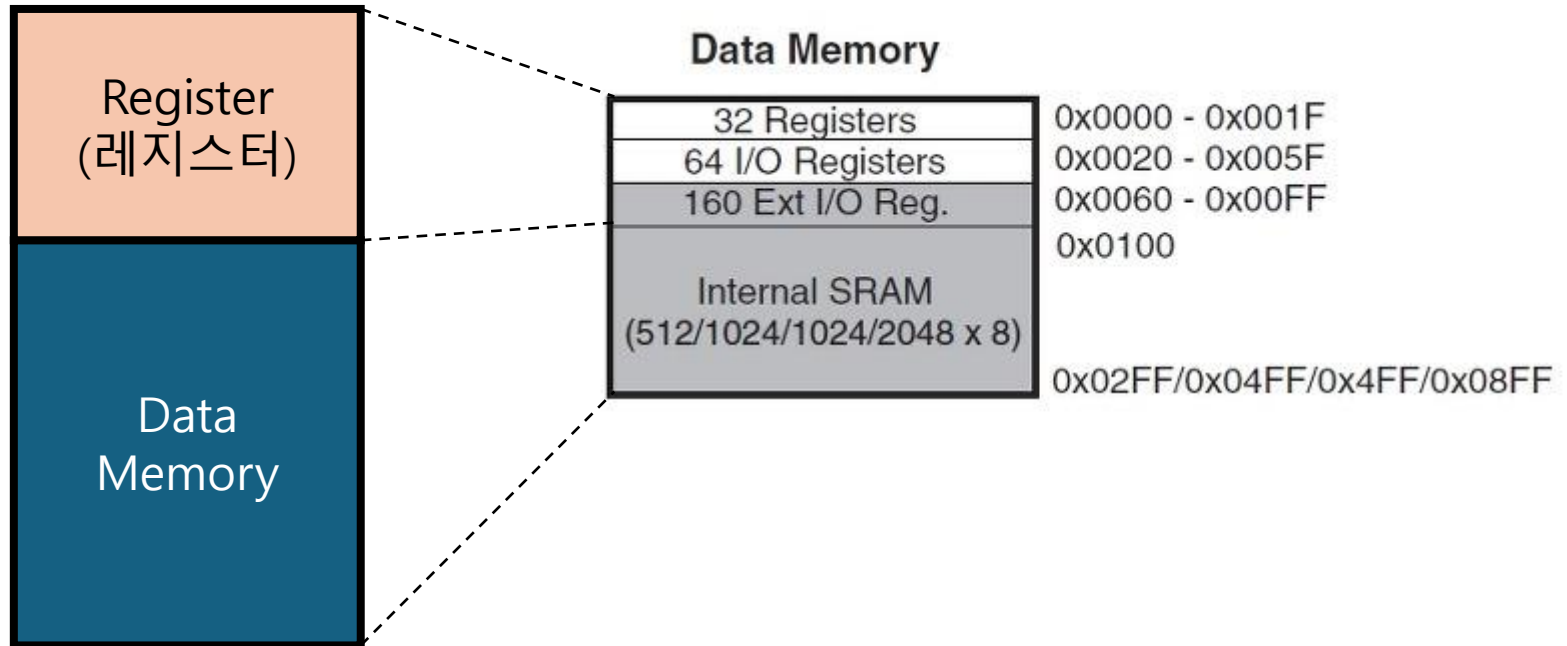
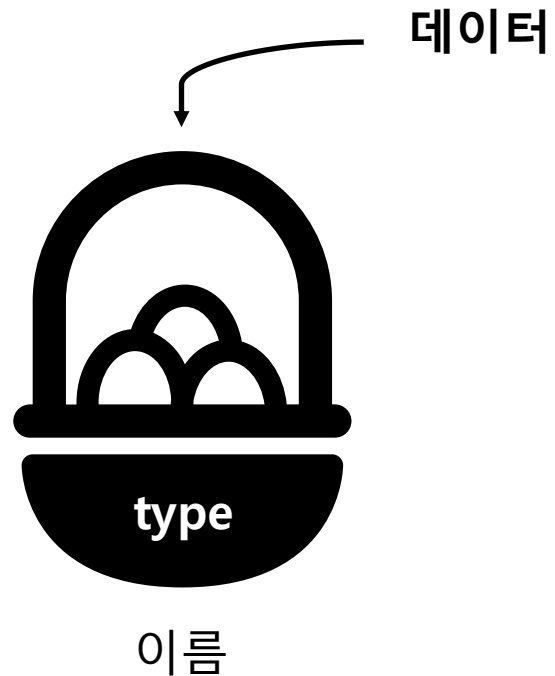
- 7-segment 실험 QUIZ – 숫자 **4**와 **2**를 1초 간격으로 표시 하는 코드를 완성 하시오.



```
void setup()
{
    DDRD = B[_____];
}

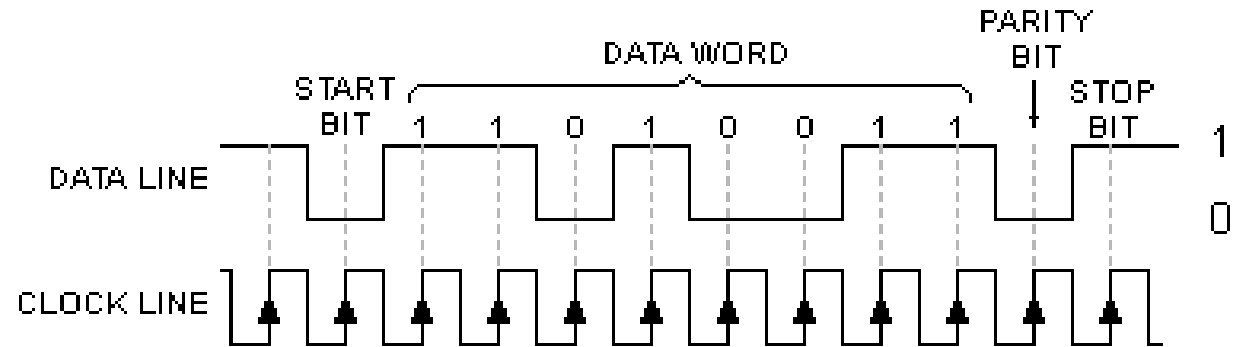
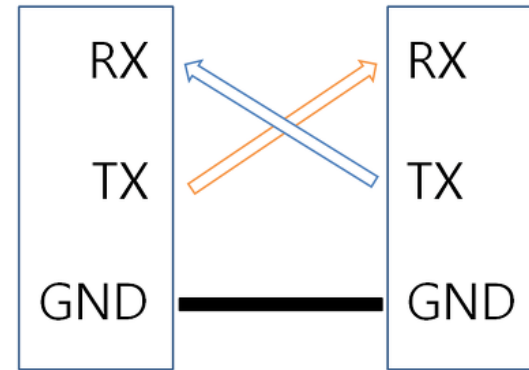
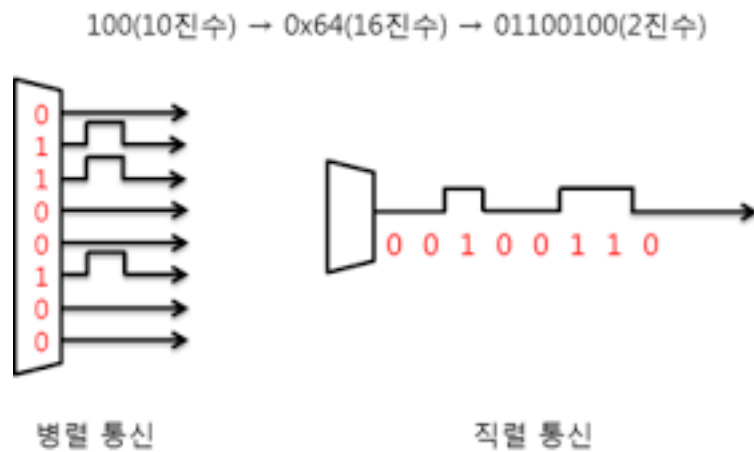
void loop()
{
    PORTD = B[_____];
    delay(1000);
    PORTD = B[_____];
    delay(1000);
}
```

# 마이크로프로세서와 C언어 - 변수





# 마이크로프로세서와 C언어 - 시리얼통신



동기/비동기 통신

# 마이크로프로세서와 C언어 - 시리얼통신

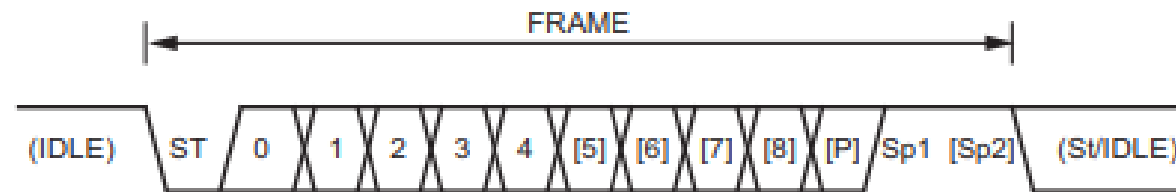
## • 비동기식 시리얼 통신

- UART 통신을 이용하기 위해서는 크게 다음의 두 가지 항목을 사전에 정의해줘야 한다.
  - **통신속도 : Baud rate**
    - Baud rate 의 단위는 bps(bits per second) : 1초당 전송하는 bit 수
    - 표준 bps: 1200, 2400, 4800, **9600**, 19200, 38400, 57600, 115200
  - **프레임사이즈 : Size of each frame field**
    - 일반적으로
      - Data bit는 1
      - Bytes site = 8 bits 사이즈로 설정
      - Stop bit는 1 bit
      - Parity bit는 0 bit로 설정
    - 통신을 사용하는 환경에 따라 미리 약속하여 사용

# 마이크로프로세서와 C언어 - 시리얼통신

- 시리얼통신 데이터 포맷

Figure 19-4. Frame Formats



**St** Start bit, always low.

**(n)** Data bits (0 to 8).

**P** Parity bit. Can be odd or even.

**Sp** Stop bit, always high.

**IDLE** No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

# 아두이노를 이용한 시리얼통신 실험

아두이노 우노로부터 시리얼통신으로 "Hello World"라는 데이터를 확인하는 코드를 작성해줘

아두이노 우노에서 "Hello World" 데이터를 시리얼 통신으로 전송하고, c 언어로 이를 수신하는 코드

## 1. 아두이노 코드 (Arduino IDE에서 업로드)

```
c
void setup() {
    Serial.begin(9600); // 시리얼 통신 시작 (9600bps)
}

void loop() {
    Serial.println("Hello World"); // "Hello World" 메시지 전송
    delay(1000); // 1초 대기 후 반복
}
```

### 설명

- `Serial.begin(9600);` → 9600bps 속도로 시리얼 통신을 시작합니다.
- `Serial.println("Hello World");` → "Hello World" 문자열을 전송합니다.
- `delay(1000);` → 1초 간격으로 데이터를 송신합니다.

```
void setup()
{
    Serial.begin(9600); // 시리얼 통신 시작 (9600bps)
}

void loop()
{
    // "Hello World" 메시지 전송
    Serial.println("Hello World");
    delay(1000); // 1초 대기 후 반복
}
```

# 마이크로프로세서와 C언어 - ASCII

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	ü	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	â	164	ü
4	4	004	EOT (end of transmission)	36	24	044	\$	68	44	104	D	100	64	144	d	132	ä	165	ÿ
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e	133	å	166	ª
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f	134	ä	167	º
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g	135	ç	168	¸
8	8	010	BS (backspace)	40	28	050	(	72	48	110	H	104	68	150	h	136	ê	169	—
9	9	011	TAB (horizontal tab)	41	29	051	)	73	49	111	I	105	69	151	i	137	ë	170	¬
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	j	138	è	171	½
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k	139	í	172	¾
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	l	140	î	173	ı
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m	141	ï	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	ò	182	⋯
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v	150	ù	183	⋯
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w	151	û	184	⋯
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x	152	ü	185	⋯
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y	153	Ö	186	⋯
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z	154	Û	187	⋯
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[	123	7B	173	{	156	£	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	⋯



# 마이크로프로세서와 C언어 - ASCII

[illegible]

0x48

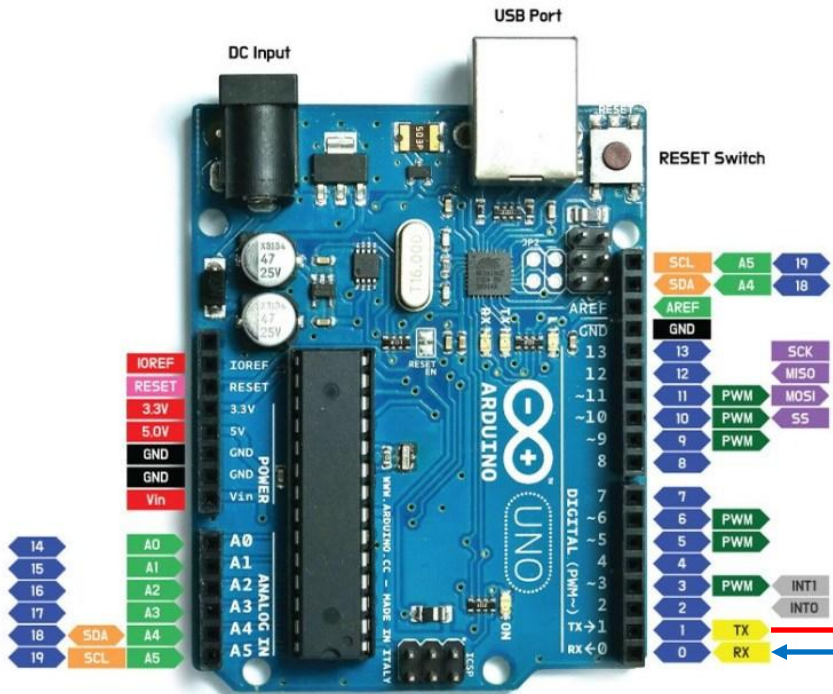
# 0x65

# 0x6C

# 0x6C

# 0x6F

# 아두이노를 이용한 시리얼통신 실험

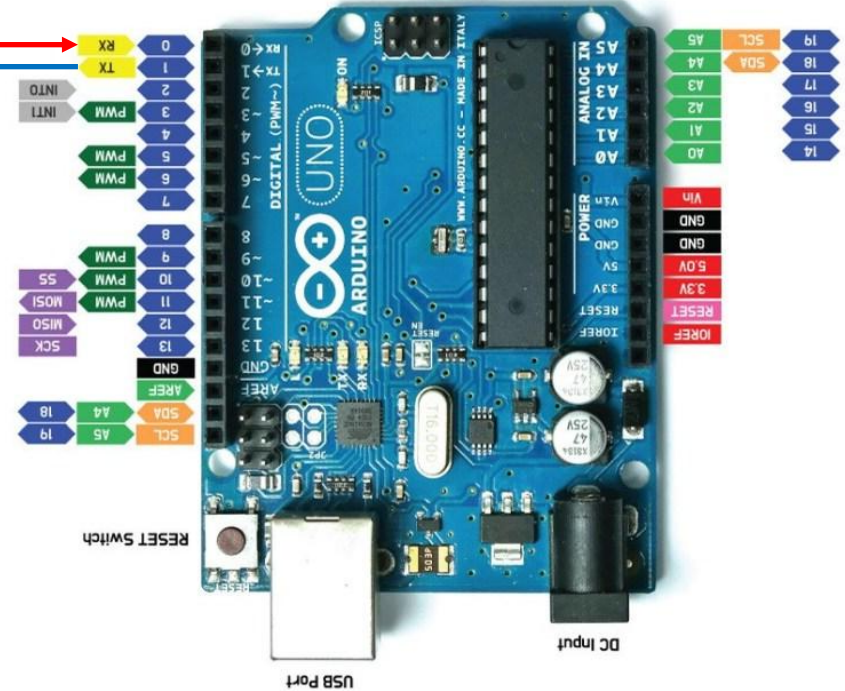


Digital Input/Output	I2C
Analog Output (PWM)	SPI
Analog Input	UART
External Interrupt	Power
Reset	Ground (GND)

iamamaker.kr

iamamaker.kr

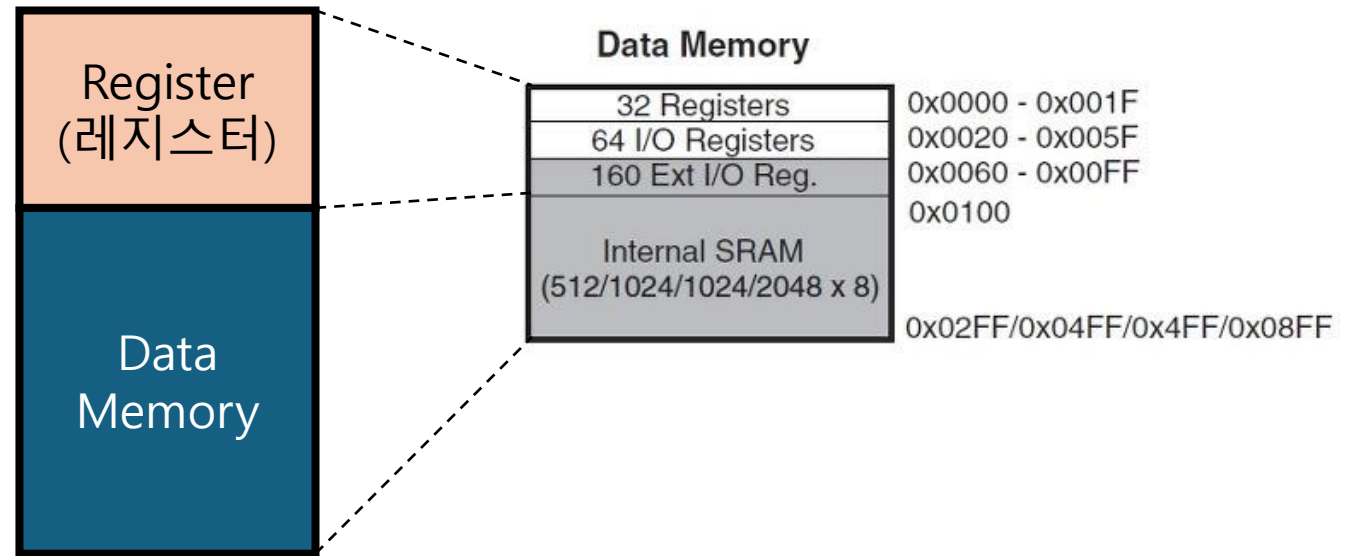
Digital Input/Output	Reset
Analog Output (PWM)	External Interrupt
Analog Input	Power
SPI	Ground (GND)
UART	



# 마이크로프로세서와 C언어 - 변수

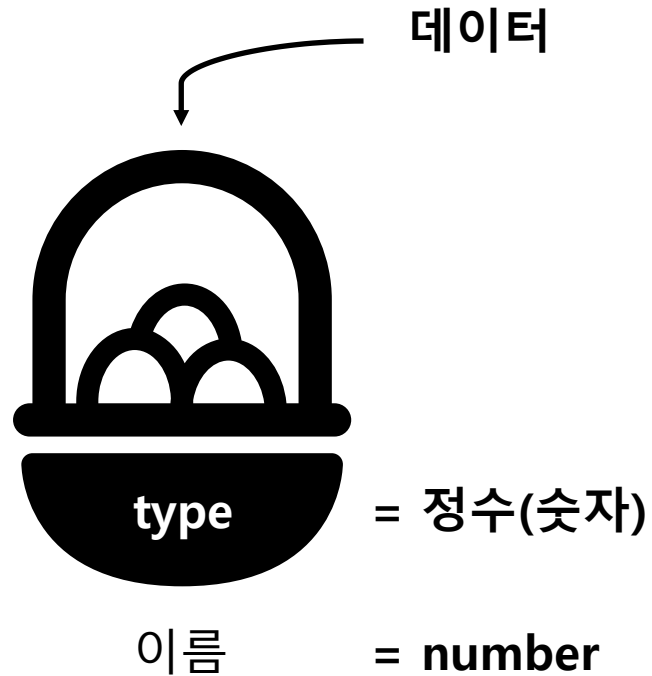
※ 메모리에 데이터를 기록하는 과정

1. 사용하지 않는 메모리 확인
2. 사용하고자 하는 사이즈 만큼의 메모리 확보
3. 데이터 저장
4. 메모리 주소를 이용하여 데이터 참조



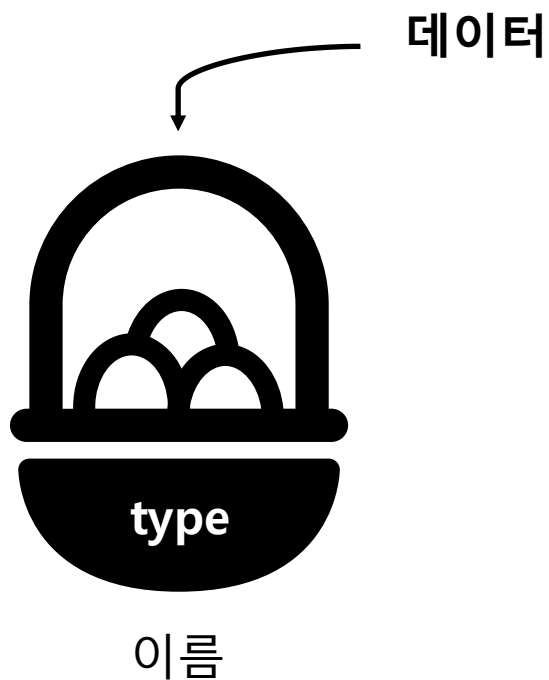


# 마이크로프로세서와 C언어 - 변수



- **int** number = 10 ;
- **unsigned int** number = 10 ;
- **unsigned int** number = -10 ;
- **long** number = -10 ;
- **unsigned long** number = -10 ;

# 마이크로프로세서와 C언어 – 변수의 종류



type	설명	사이즈(Byte)
char	단일 문자를 저장하거나 -128에서 127 사이의 작은 정수를 저장하는 데 사용	1
unsinged char	0에서 255 사이의 값을 저장할 수 있는 8비트 부호 없는 정수	1
byte	0에서 255 사이의 값을 저장할 수 있는 8비트 부호 없는 정수	
int	정수를 저장하는 데 사용, 아두이노에서는 보통 -32,768에서 32,767 사이의 값을 저장.	2
unsigned int	부호 없는 정수를 저장, 0에서 65,535 사이의 값을 저장	2
long	더 큰 정수를 저장할 수 있으며, -2,147,483,648에서 2,147,483,647 사이의 값을 저장	4
unsigned long	부호 없는 더 큰 정수를 저장하며, 0에서 4,294,967,295 사이의 값을 저장	4
short	-32,768에서 32,767 사이의 값을 저장할 수 있는 작은 정수	2
float	부동소수점 숫자를 저장하는 데 사용, 대략 -3.4E+38에서 +3.4E+38 사이의 값을 6~7자리 의 정밀도로 저장	4
double	아두이노에서는 float와 동일하게 처리되는 경우가 많으며, 일부 아두이노 보드(예: SAMD 기반의 보드)에서는 더 큰 범위와 정밀도를 제공	4
bool	true 또는 false 값을 저장하는 데 사용	1

# 정수형 변수에 데이터를 저장하고 시리얼 통신으로 데이터 확인



You

number라고 이름 붙인 정수형 변수에 데이터 10을 넣고 이를 시리얼통신으로 확인하는 아두이노 코드를 작성하세요.



ChatGPT

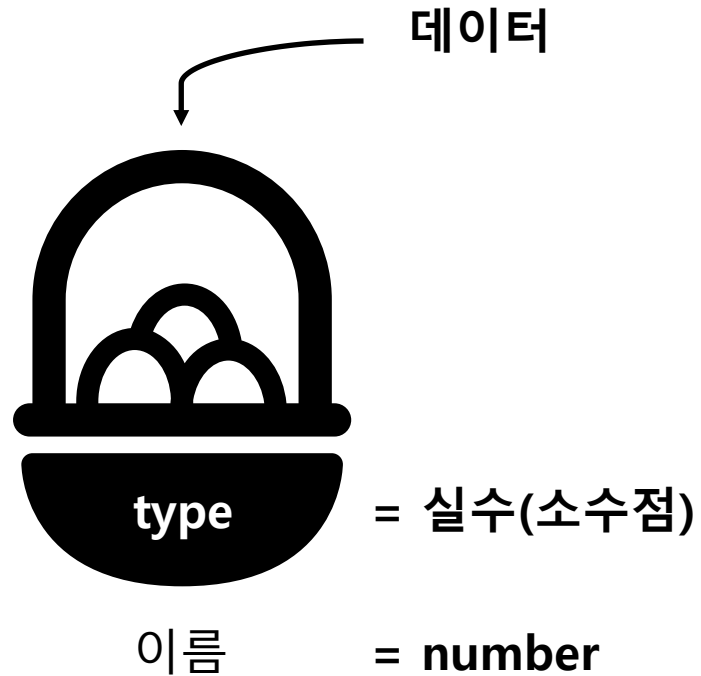
아두이노에서 특정한 데이터를 시리얼 통신을 통해 확인하기 위해서는 시리얼 통신을 초기화하고, 해당 데이터를 시리얼 모니터로 전송하는 코드를 작성해야 합니다. 아래의 코드 예제는 정수형 변수 `number`에 `10`을 할당하고, 이를 시리얼 통신을 통해 전송하여 시리얼 모니터에서 확인할 수 있도록 하는 아두이노 스케치입니다.

cpp

Copy code

```
void setup() {  
    // 시리얼 통신을 시작합니다. 보드레이트는 9600bps로 설정합니다.  
    Serial.begin(9600);  
}  
  
void loop() {  
    // 정수형 변수 number를 선언하고 10을 할당합니다.  
    int number = 10;  
  
    // number 변수의 값을 시리얼 모니터로 전송합니다.  
    Serial.println(number);  
  
    // 너무 빠르게 데이터가 전송되지 않도록 1초 간격으로 딜레이를 줍니다.  
    delay(1000);  
}
```

# 마이크로프로세서와 C언어 - 변수



- **float** number = 0.13 ;
- **double** number = -10.123 ;

# 실수형 변수에 데이터를 저장하고 시리얼 통신으로 데이터 확인



You

number라고 이름 붙인 실수형 변수에 데이터 10.12을 넣고 이를 시리얼통신으로 확인하는 아두이노 코드를 작성하세요.



ChatGPT

실수형 변수에 데이터를 할당하고, 이를 시리얼 통신을 통해 확인할 수 있는 아두이노 코드는 아래와 같습니다. 이 예제에서는 'number'라는 이름의 실수형(float) 변수에 '10.12'를 할당하고, 이 값을 시리얼 모니터를 통해 출력합니다.

cpp

Copy code

```
void setup() {  
    // 시리얼 통신을 시작합니다. 보드레이트는 9600bps로 설정합니다.  
    Serial.begin(9600);  
}  
  
void loop() {  
    // 실수형 변수 number를 선언하고 10.12를 할당합니다.  
    float number = 10.12;  
  
    // number 변수의 값을 시리얼 모니터로 전송합니다.  
    Serial.println(number);  
  
    // 너무 빠르게 데이터가 전송되지 않도록 1초 간격으로 딜레이를 줍니다.  
    delay(1000);  
}
```

# 실험 - 변수를 담는 메모리 사이즈를 알아봅시다.

## sizeof



You

변수형에 따른 메모리 사이즈를 시리얼통신으로 확인하는 코드를 만들어 줘



ChatGPT

아두이노에서 다양한 데이터 타입의 메모리 사이즈를 확인하고 이를 시리얼 통신을 통해 출력하는 코드를 작성할 수 있습니다. 이 코드는 `sizeof` 연산자를 사용하여 각 데이터 타입의 메모리 크기를 바이트 단위로 계산하고, 그 결과를 시리얼 모니터에 출력합니다.

```
cpp Copy code

void setup() {
    // 시리얼 통신을 시작합니다. 보드레이트는 9600bps로 설정합니다.
    Serial.begin(9600);

    // 데이터 타입별 메모리 사이즈를 시리얼 모니터로 출력합니다.
    Serial.print("char 크기: ");
    Serial.println(sizeof(char)); // char의 메모리 크기 출력

    Serial.print("int 크기: ");
    Serial.println(sizeof(int)); // int의 메모리 크기 출력

    Serial.print("unsigned int 크기: ");
    Serial.println(sizeof(unsigned int)); // unsigned int의 메모리 크기 출력

    Serial.print("long 크기: ");
    Serial.println(sizeof(long)); // long의 메모리 크기 출력

    Serial.print("unsigned long 크기: ");
    Serial.println(sizeof(unsigned long)); // unsigned long의 메모리 크기 출력

    Serial.print("float 크기: ");
    Serial.println(sizeof(float)); // float의 메모리 크기 출력


    Serial.print("double 크기: ");
    Serial.println(sizeof(double)); // double의 메모리 크기 출력 (아두이노에서 double은 flo

    Serial.print("bool 크기: ");
    Serial.println(sizeof(bool)); // bool의 메모리 크기 출력
}

void loop() {
    // 이 예제에서 loop() 함수는 사용하지 않습니다.
}
```

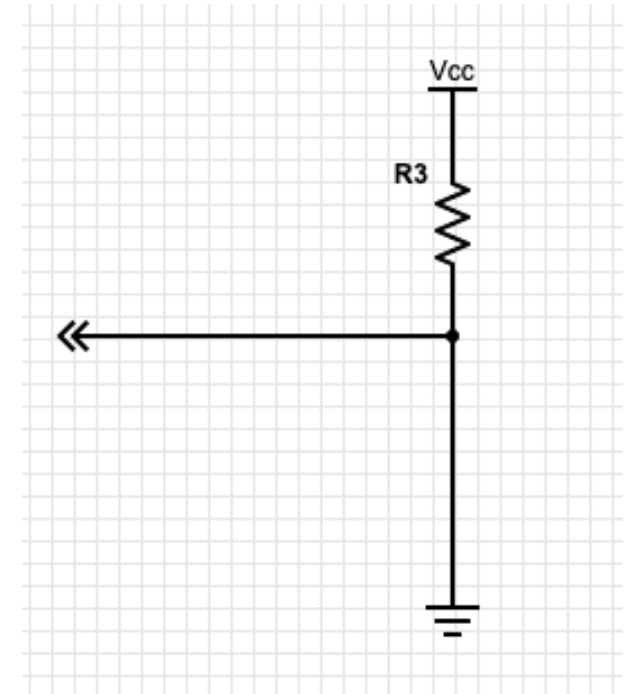
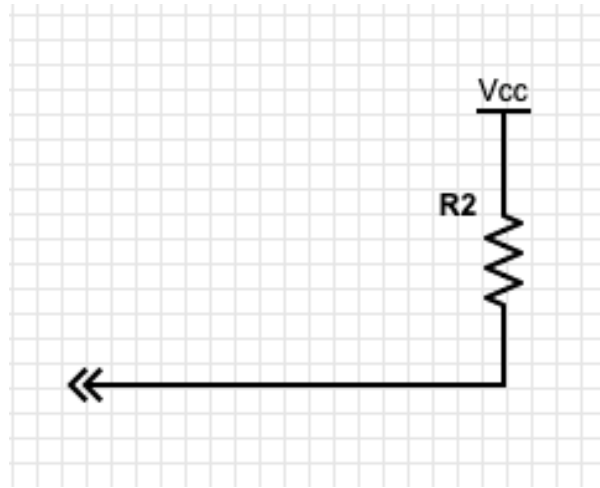
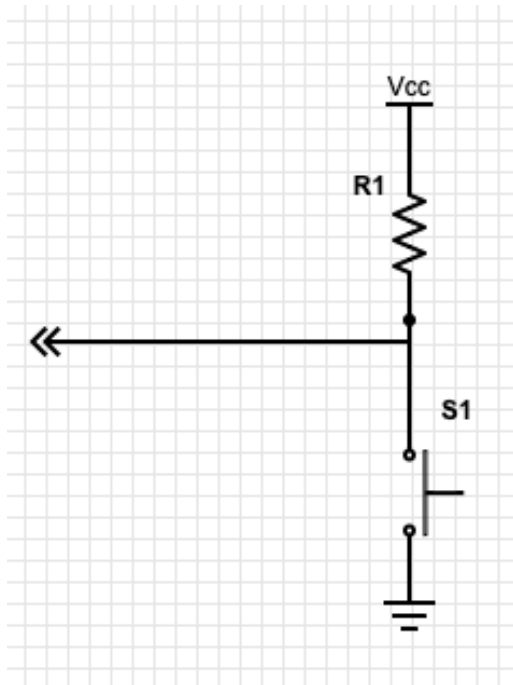
# 실험 – 1부터 10까지 곱한 결과는?

cpp

 Copy code

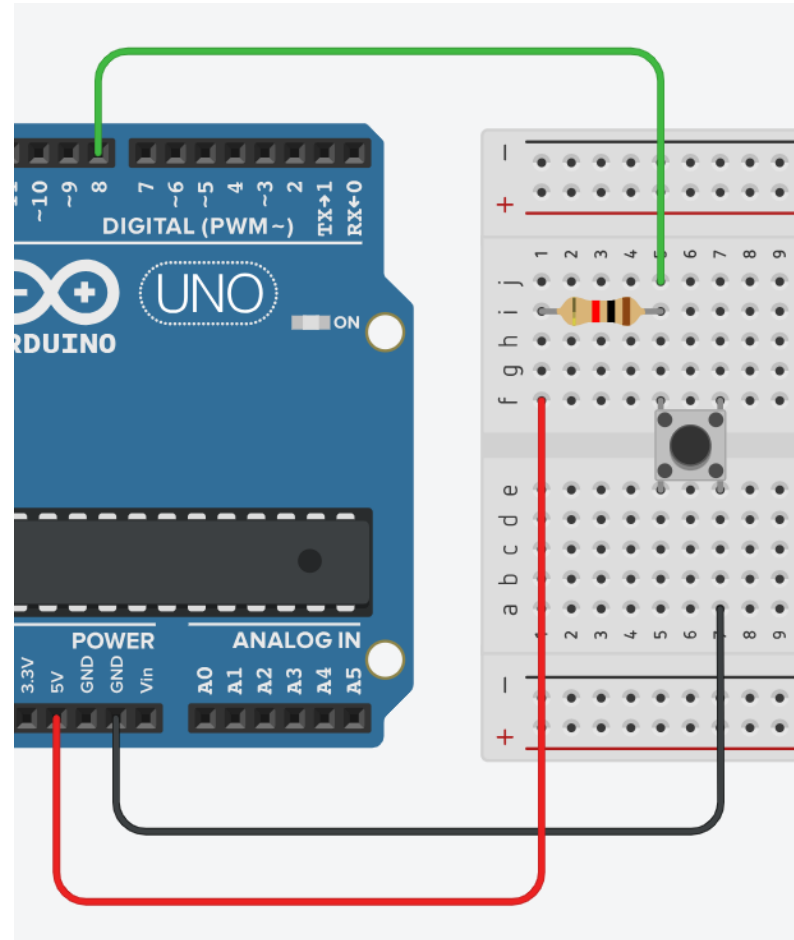
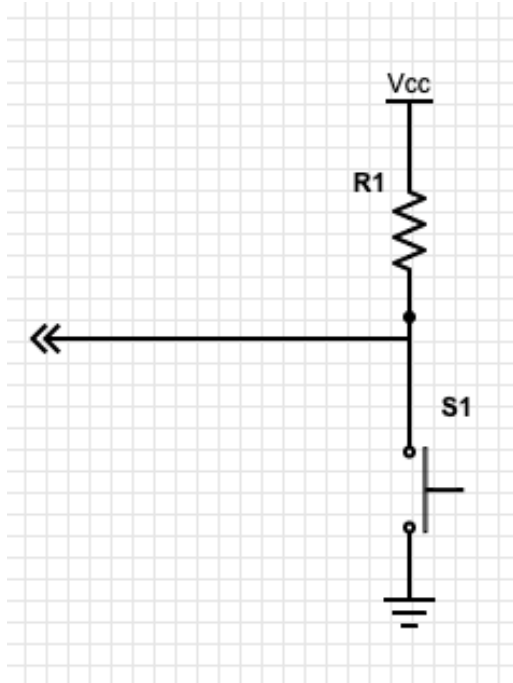
```
void setup() {  
    // 시리얼 통신을 시작합니다. 보드레이트는 9600bps로 설정합니다.  
    Serial.begin(9600);  
  
    // 1부터 10까지 곱한 결과를 직접 계산하고 변수에 할당합니다.  
    unsigned long result = 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10;  
  
    // 계산된 결과를 시리얼 모니터로 출력합니다.  
    Serial.print("1부터 10까지 곱한 결과: ");  
    Serial.println(result);  
}  
  
void loop() {  
    // 이 예제에서 loop() 함수는 사용하지 않습니다.  
}
```

# 버튼 입력 실험(Digital Input)

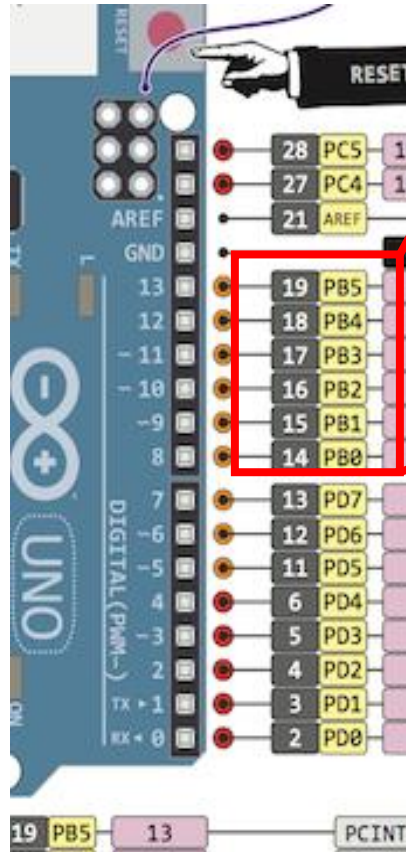
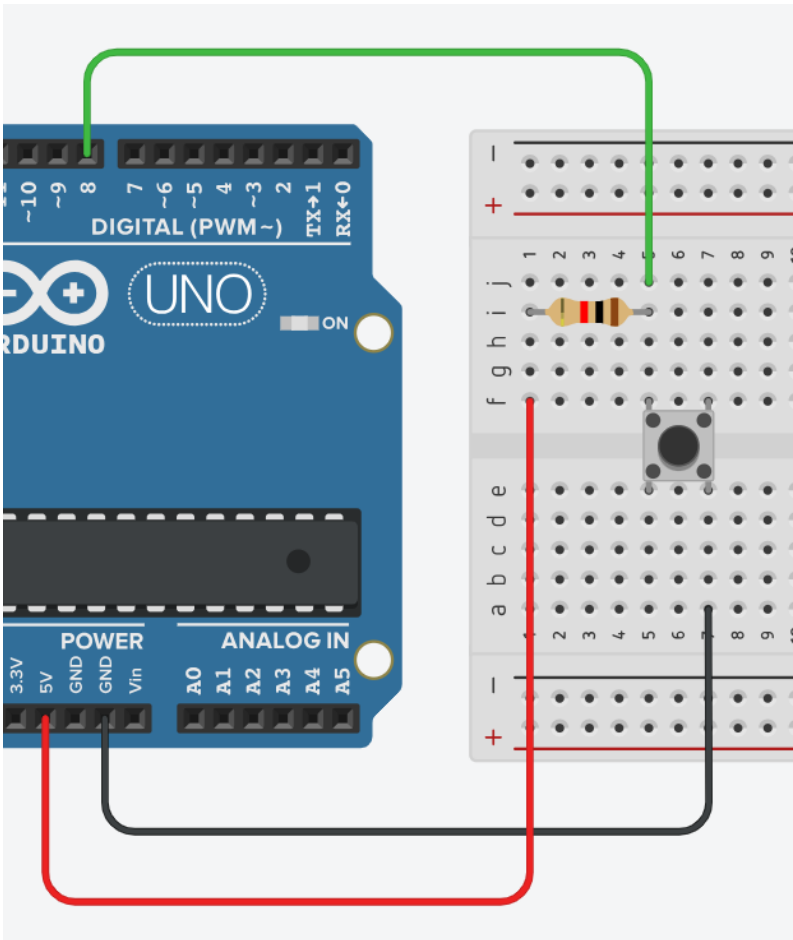




# 버튼 입력 실험(Digital Input)



# 버튼 입력 실험(Digital Input)



### 13.4.2 PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
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

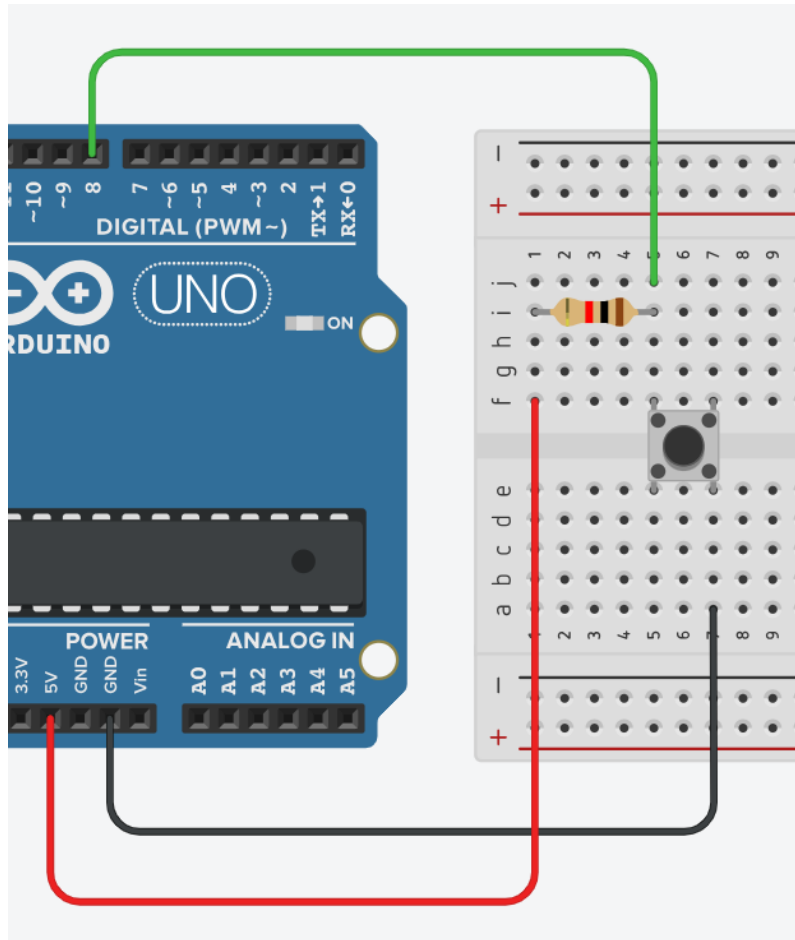
### 13.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
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

#### 13.4.4 PINB – The Port B Input Pins Address

[illegible]

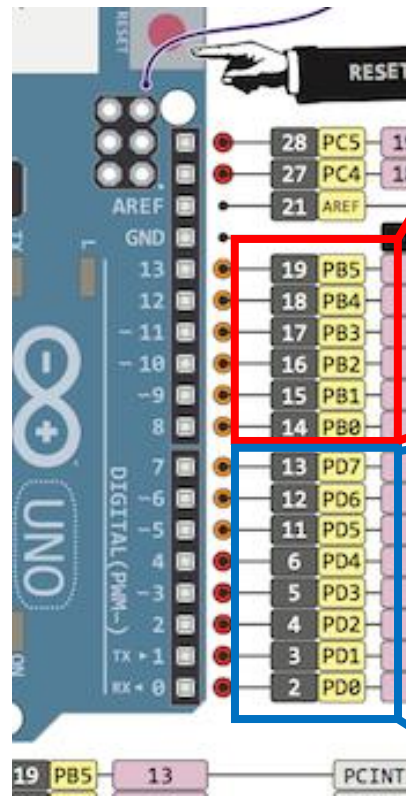
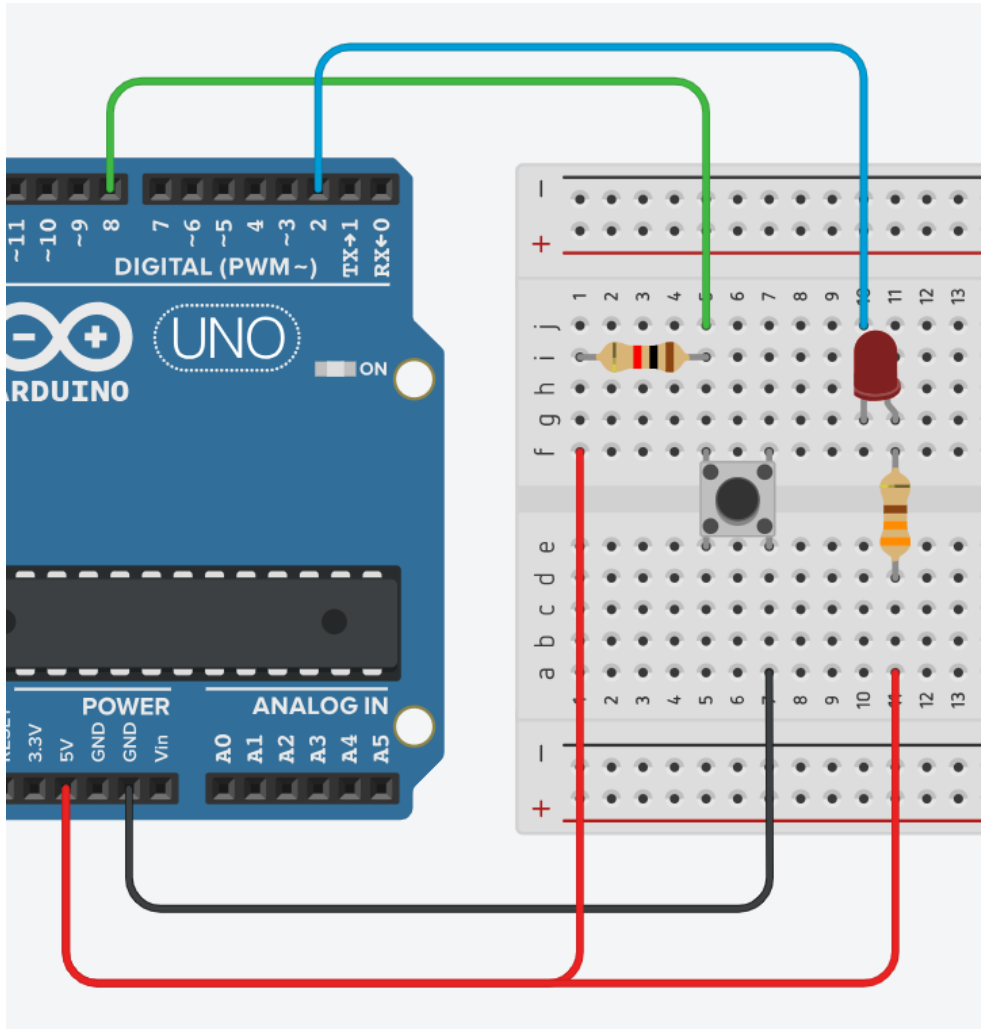
# 버튼 입력 실험(Digital Input)



```
void setup()
{
  DDRB = B00000000 ;
  Serial.begin(9600) ;
}

void loop()
{
  Serial.println(PINB) ;
}
```

# 버튼 입력 실험(Digital Input)



### 13.4.2 PORTB – The Port B Data Register

[illegible]

### 13.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
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

#### 13.4.4 PINB – The Port B Input Pins Address

[illegible]

### 13.4.8 PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
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

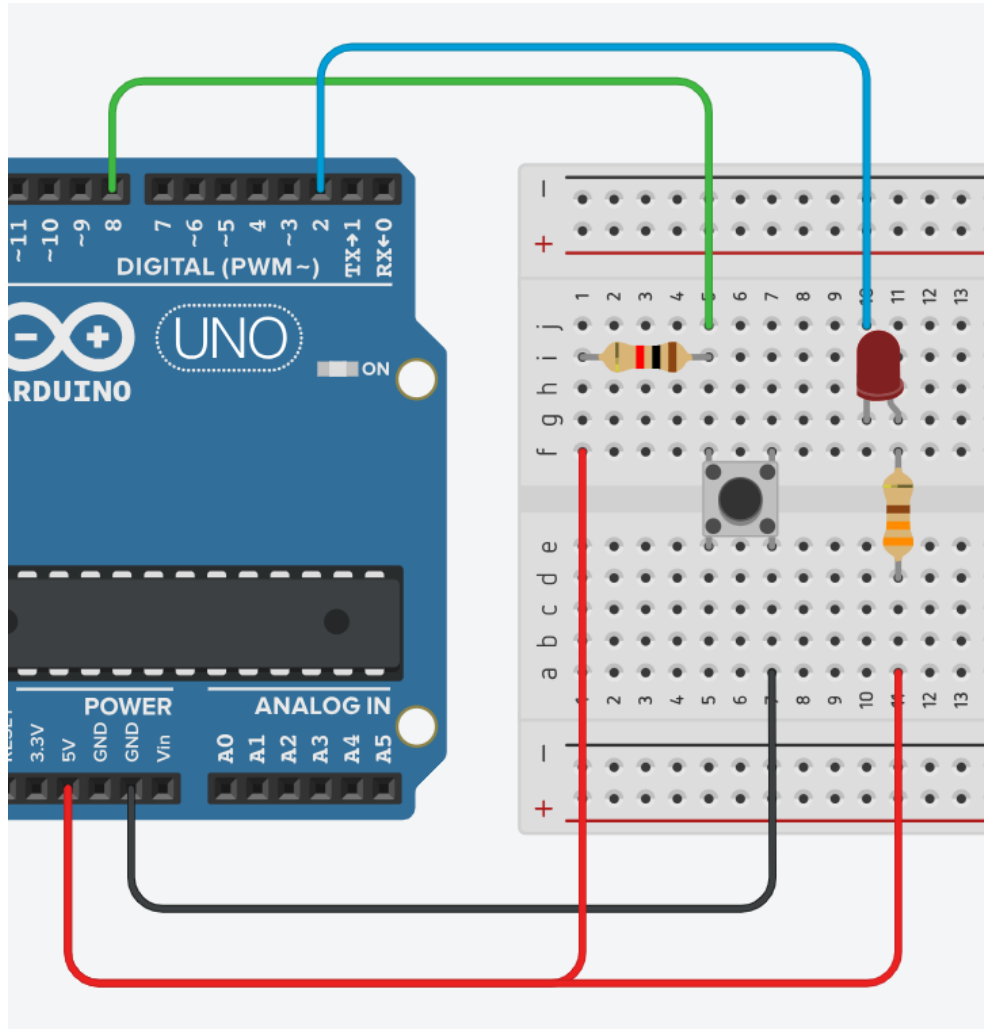
### 13.4.9 DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
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

### 13.4.10 PIND – The Port D Input Pins Address

[illegible]

# 버튼 입력 실험(Digital Input)



```
void setup()
{
  DDRB = B00000000 ;
  DDRD = B00000100 ;

  Serial.begin(9600) ;
}

void loop()
{
  Serial.println(PINB) ;

  if( PINB == B00000001 )
  {
    //버튼이 눌리지 않음
    PORTD = B0000100 ;           //LED 꺼짐
  }
  else
  {
    //버튼이 눌림
    PORTD = B00000000 ;         //LED 켜짐
  }
}
```