



드론으로 배우는
프로그래밍 교실

Ch3-4. 스위치와 연산



01 스위치	01
스위치란?	02
스위치 연결	03
풀다운 방식	04
스위치 입력 받기 – 연결	05
스위치 입력 받기 – 코드	06
 02 연산과 친해지기	 07
연산이란?	08
다양한 연산자	09
연산 작성 해보기	12
 03 자료형	 14
자료형이란?	15
자료형의 종류	16
ASCII Code	18
자료형 실습하기	19



드론으로 배우는
프로그래밍 교실

초판발행 2016년 9월 23일
지은이 이상준 | 펴낸이 CodingBird
펴낸곳 WHIT | 주소 안산시 한양대학교로55 창업보육센터 B01

Published by WHIT. Printed in Korea
Copyright © 2016 CodingBird & WHIT

이 책의 저작권은 CodingBird와 WHIT에 있습니다.
저작권법에 의해 보호를 받는 저작물이므로
무단 복제 및 무단 전재를 금합니다.

01 스위치



스witch는 사용자가 눌렀을 때와 아닐 때를 구분하여 각각의 경우에 따라 다른 명령을 실행할 수 있게 해줍니다.

여러분의 컴퓨터 본체에도 스위치가 달려있고, 핸드폰에도 스위치가 달려 있습니다.

스위치의 작동 원리에 대해 살펴봅시다.

스위치란?

스위치
정의

스witch는 전기 회로를 이었다 끊었다 하는 장치로 보통 전등, 라디오, 텔레비전 따위의 전기 기구를 손으로 올리고 내리거나 누르거나 틀어서 작동하는 부분을 일컫는다.

출처: 네이버 백과사전



<그림1-1> 스위치 회로도

스witch는 쉽게 말해 연결해주는 장치입니다. 사용자의 선택에 따라 연결할 수도 끊을 수도 있습니다.

스위치
종류

스witch는 크게 탭트 스위치, 슬라이드 스위치, 푸쉬 스위치, 훅 스위치, 멀티웨이 스위치로 나뉩니다. 훅 스위치는 보통 전화기에 사용되어 무게에 따라 스위치가 동작하고, 멀티웨이 스위치는 보통 조이스틱에 사용됩니다.



<그림1-2> 탭트 스위치



<그림1-3> 슬라이드 스위치



<그림1-4> 푸쉬 스위치



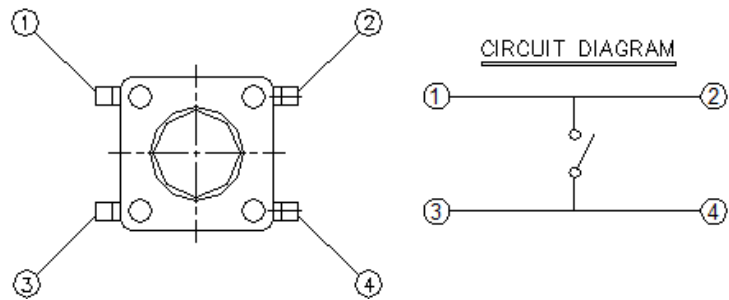
<그림1-5> 훅 스위치



<그림1-6> 멀티웨이 스위치

택트 스위치

택트 스위치는 가장 흔한 스위치로 버튼 혹은 key라고도 불립니다.



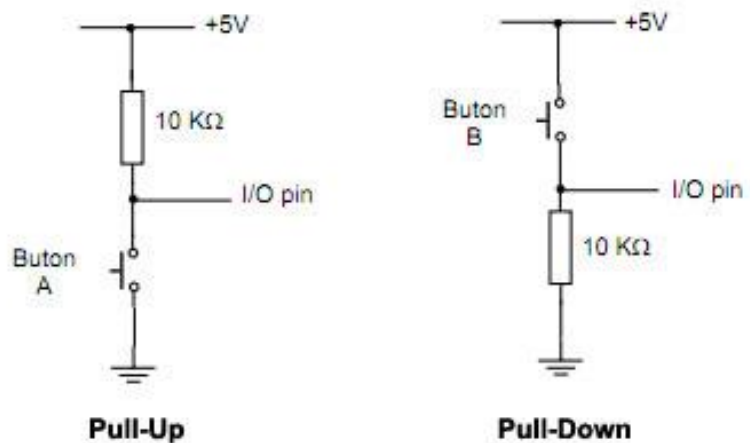
<그림1-7> 스위치 회로도

우리가 사용할 택트 스위치는 다리가 4개로, 1번과 2번이 기본적으로 연결되어 있고, 3번과 4번이 기본적으로 연결되어 있습니다.

이 때, 스위치를 누르게 되면 1번과 4번 혹은 1번과 3번을 연결할 수 있게 됩니다.

풀업 풀다운

아두이노와 스위치의 연결에는 풀업 방식과 풀다운 방식이 있습니다. 저항이 위에 있으면 풀 업, 아래 있으면 풀 다운입니다.



<그림1-8> 풀업 풀다운

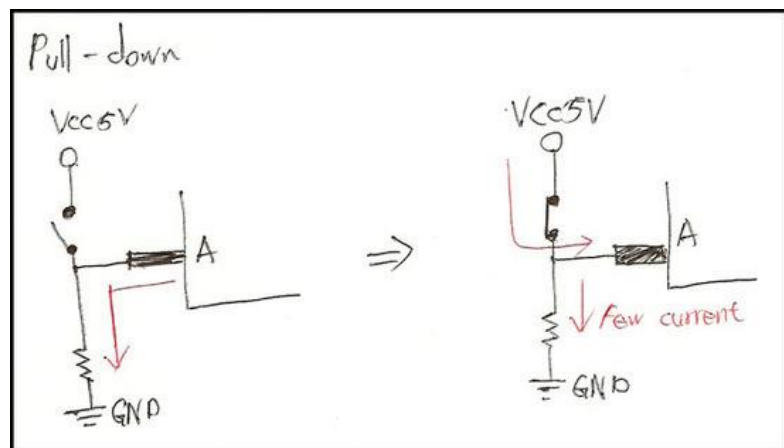
두 가지 방식 중 우리는 풀다운 방식을 사용할 것입니다.

풀다운 방식

스위치는 보통 두 선의 연결을 이어주거나 끊어줄 때 쓰이기도 하지만, 아두이노와 함께 쓰일 때는 아두이노 핀에 입력을 줄 때 쓰이기도 합니다.

아두이노의 핀은 디지털 핀과 아날로그 핀으로 나뉘고, 디지털핀은 받아들이는 전압이 높으면 1, 낮으면 0으로 입력을 받습니다.

풀다운방식을 사용하면 스위치를 누르지 않았을 경우 0, 스위치를 누를 경우 1이 입력으로 들어가게 됩니다.



<그림1-9> 풀다운 저항

위 그림과 같이 풀다운 방식으로 회로를 구성하여, 스위치가 눌리지 않았을 때는 입력핀이 GND와 연결되어 전압이 낮은 상태(LOW)로 유지됩니다.

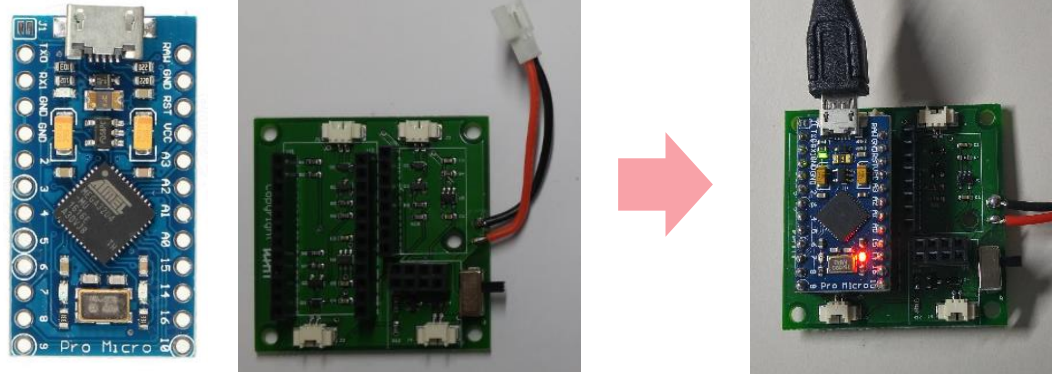
스위치가 눌릴 때는 VCC가 입력핀과 연결되어 입력핀은 VCC와 같이 전압이 높은 상태(HIGH)로 유지됩니다.

이를 통하여 `digitalRead(2)`를 했을 때 상태가 HIGH인지 LOW인지 구분하여 스위치의 눌림을 감지할 수 있게 됩니다.

스위치 입력 받기 - 연결

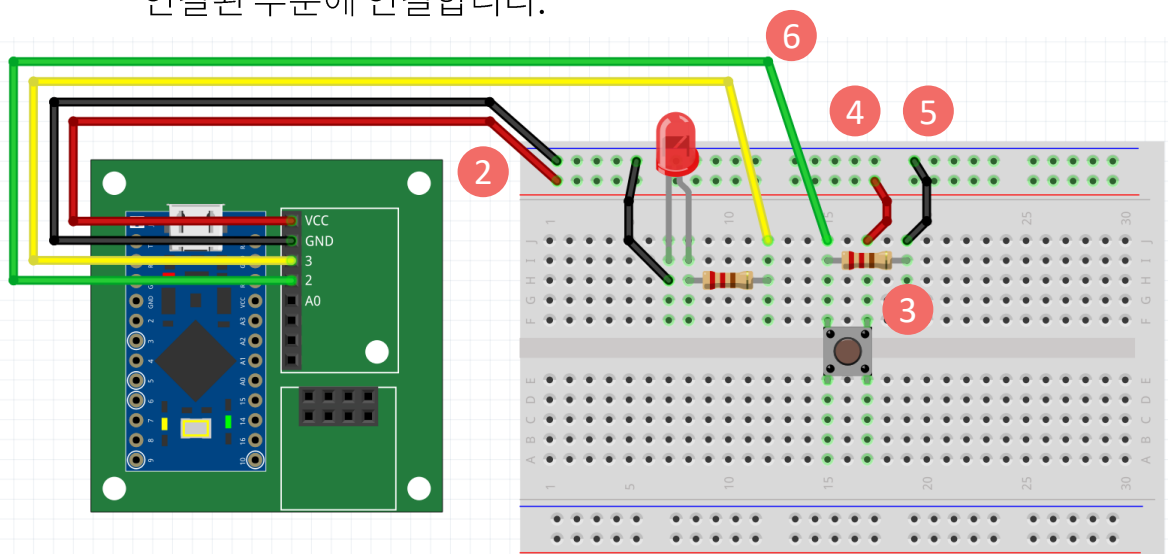
스위치 입력 받기 연결

- 1 UBS가 연결되어 있는 아두이노를 메인 보드에 끼웁니다.(방향에 유의합니다)



<그림1-10> 아두이노와 메인 보드 연결

- 2 메인 보드의 VCC핀(위에서1번째)과 빵판의 빨간줄을 점퍼선으로 연결합니다.
- 3 스위치와 저항을 빵판에 그림과 같이 꽂아 넣습니다.
(스위치의 왼쪽과 저항의 왼쪽이 세로로 같은 라인에 위치)
- 4 스위치의 우측상단 다리와 빵판의 빨간줄을 점퍼선으로 연결합니다.
- 5 저항의 우측을 빵판의 파란줄과 점퍼선으로 연결합니다.
- 6 메인 보드의 2번핀(위에서 4번째)을 스위치와 저항이 같이 연결된 부분에 연결합니다.



fritzing

<그림1-11> 메인 보드와 스위치 연결

스위치 입력 받기 - 코드

스위치 입력 받기 코드

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.


```
ch3_4_1_sw_serial
1 void setup() {
2   Serial.begin(9600);
3   pinMode(2, INPUT);
4 }
5
6 void loop() {
7   if(digitalRead(2) == HIGH){
8     Serial.println("원하는 문구 적기");
9   }
10 }
```

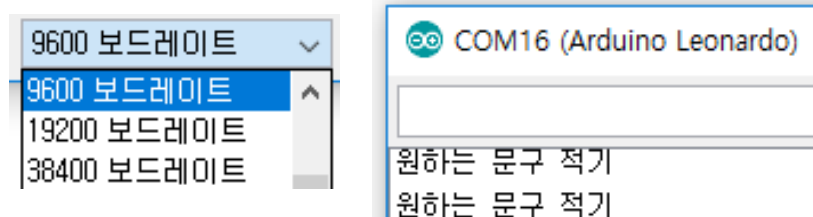
<그림1-12> 스위치 입력 받기

스위치 입력 받기 코드 해석

```
void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT); //2번 핀을 입력으로 설정
}
```

```
void loop() {
  if(digitalRead(2) == HIGH){//만약 2번핀이 HIGH이면 실행
    Serial.println("원하는 문구 적기"); //원하는 문구 출력
  }
}
```

- 2  버튼을 눌러 시리얼 모니터를 켭니다.
- 3 보드레이트를 맞추는 후 스위치를 누르고 시리얼 모니터에서 문구가 뜨는 것을 확인합니다.



<그림1-13> 문구 확인

02 연산과 친해지기

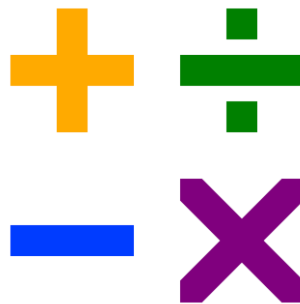


프로그래밍에서의 연산자는 수학에서의 연산자와는 약간 차이가 있습니다.
수학에서의 연산자와 프로그래밍에서의 연산자의 차이에 대해 알아보고,
프로그래밍에서의 연산자의 특성을 알아 봅시다.
나아가 프로그래밍 연산자를 직접 써보며 어떻게 사용되는지 감을
잡아봅시다.

계산과 연산
차이

계산이란 수를 세는 것입니다. 예를 들어 사과를 한 개, 두 개, 세 개 이렇게 셀 때 계산을 한다고 합니다.

연산은 어떤 식이 나타내는 규칙 또는 그것에 따라서 계산 하는 일입니다. 예를 들어 $3 + 4 = 7$ 과 같은 것이 연산입니다.



<그림2-1> 사칙연산

연산자

연산자는 연산을 할 때의 규칙으로 수학에서의 연산 규칙과 프로그래밍에서의 연산 규칙은 약간 다릅니다.

예를 들어 프로그래밍에서의 연산은 다음과 같이 구성됩니다.

- ① $x = x + 3$; // x값에 x+3값을 대입합니다.
- ② $x = x - 3$; // x값에 x-3값을 대입합니다.
- ③ $y = x * 6$; // y값에 x*6값을 대입합니다.
- ④ $y = x / 6$; // y값에 x/6값을 대입합니다.

1번과 같은 경우에 만약 x에 5가 들어있었다면 연산이 끝난 후 x에 들어 있는 값은 8이 됩니다.

수학과
프로그래밍
연산 차이

위와 같이 수학에서의 =기호는 '같다'라는 규칙을 가지고 있었는데 반해, 프로그래밍에서의 =기호는 '대입'이라는 규칙을 가지고 있습니다.

다양한 연산자

연산자 종류

연산자에는 다양한 종류가 있습니다.

분류	연산자
대입 연산자	=
산술 연산자	+, -, *, /, %
복합 대입 연산자	+=, -=, *=, /=, %=
증감 연산자	++, --
관계 연산자	>, <, ==, !=, >=, <=
논리 연산자	&&, , !
조건 연산자	?:
비트 논리 연산자	&, , ^, ~
비트 이동 연산자	>>, <<

<그림2-2> 연산자 종류

대입 연산자

대입 연산자는 말 그대로 대입을 해주는 연산자입니다. 예를 들어 `x = 5;` 를 실행할 경우 변수 `x`에는 5가 들어가게 됩니다.

산술 연산자

산술 연산자는 수학에서의 연산자와 같습니다. 덧셈, 뺄셈, 곱셈, 나눗셈, 나머지 등의 연산을 할 수 있게 해주는 연산자입니다.

복합 대입 연산자

복합 대입 연산자는 산술 연산과 대입 연산을 한번에 할 수 있게 해줍니다.

`x = x + 3;`의 연산은 `x += 3;`과 같은 연산입니다.

- ① $x+=y$ 는 $x=x+y$ 와 같습니다.
x에 들어있는 수를 y만큼 증가한 뒤 x에 대입합니다.
- ② $x-=y$ 는 $x=x-y$ 와 같습니다.
x에 들어있는 수를 y만큼 감소한 뒤 x에 대입합니다.
- ③ $x*=y$ 는 $x=x*y$ 와 같습니다.
x에 들어있는 수에 y를 곱한 뒤 x에 대입합니다.
- ④ $x/=y$ 는 $x=x/y$ 와 같습니다.
x에 들어있는 수에 y를 나눈 뒤 x에 대입합니다.
- ⑤ $x\%=y$ 는 $x=x\%y$ 와 같습니다.
x에 들어있는 수를 y로 나눈 나머지를 x에 대입합니다.

증감 연산자

증감 연산자는 1씩 증가 또는 감소시킬 수 있는 연산자입니다.

- ① $x++$ 는 $x=x+1$ 과 같습니다. 1씩 증가합니다.
- ② $x--$ 는 $x=x-1$ 과 같습니다. 1씩 감소합니다.

관계 연산자

관계 연산자는 주로 하나의 변수와 다른 변수를 비교하여 조건이 참인지 결정하는데 쓰입니다.

- ① $x == y$ // x와 y가 같은지 비교합니다.
- ② $x != y$ // x와 y가 다른지 비교합니다.
- ③ $x < y$ // x가 y보다 작은지 비교합니다.
- ④ $x > y$ // x가 y보다 큰지 비교합니다.
- ⑤ $x <= y$ // x가 y보다 작거나 같은지 비교합니다.
- ⑥ $x >= y$ // x가 y보다 크거나 같은지 비교합니다.

논리 연산자

논리 연산자는 보통 두 개 이상의 조건들을 비교하고 참, 거짓을 결정합니다. AND, OR와 NOT이 주로 쓰입니다.

1 AND논리

`if(x > 0 && x < 5)` //2개의 조건이 모두 참일 때만 참

2 OR논리

`if(x > 0 || x < 5)` //2개의 조건 중 하나라도 참이면 참

3 !논리

`if(! x > 0)` //조건이 거짓이면만 참

조건 연산자

조건 연산자는 `test ? run1 : run2` 처럼 `?:` 구조로 되어 있습니다. test라는 조건이 참일 경우 run1이 실행되고, 거짓일 경우 run2가 실행됩니다.

비트 연산자

비트 연산자는 비트 단위의 연산을 진행할 때 사용됩니다. 비트 단위의 연산은 AND, OR 연산 등이 있습니다.

꿀TIP

비트 연산자

비트 연산자는 조금 난이도가 있는 개념으로, 이 책에서는 깊게 다루지 않습니다.

AND연산은 비교하고자 하는 두 개의 비트가 둘 다 1일 때만 1이 나오는 연산입니다. 그 외는 다 0이 나옵니다.

OR연산은 비교하고자 하는 두 개의 비트 중 1개라도 1이면 1이 나오는 연산입니다. 둘 다 0일 경우 0이 나옵니다.

연산 작성
해보기

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

ch3_4_2_sw_led_arith

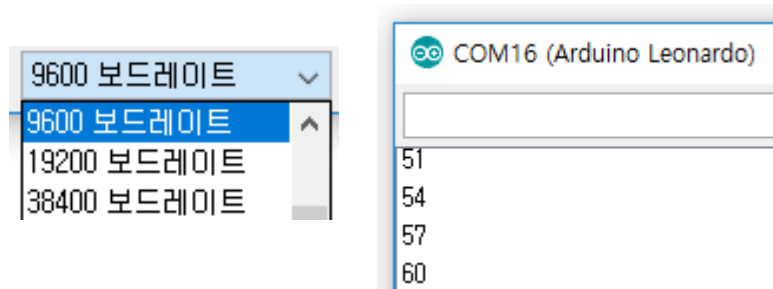
```
1 int pushValue = 0;
2 int cnt = 3;
3
4 void setup() {
5     Serial.begin(9600);
6     pinMode(3, OUTPUT);
7     pinMode(2, INPUT);
8 }
9
10 void loop() {
11     if (digitalRead(2) == HIGH && cnt < 5) {
12         pushValue += cnt;
13         if (pushValue > 255) pushValue = 0;
14         Serial.println(pushValue);
15     }
16     delay(10);
17     analogWrite(3, pushValue);
18 }
```

<그림2-3> 연산 작성 코드

- 2  버튼을 눌러 시리얼 모니터를 켭니다.

연산 작성 해석

- 3 보드레이트를 맞춘 후 스위치를 누르고 시리얼 모니터에서 값이 변하는지, LED의 밝기 변화가 있는지 확인합니다.



<그림2-4> 값 확인

```
int pushValue = 0; //버튼 눌림 횟수 저장 변수
int cnt = 3; //버튼을 눌렀을 때 증가하는 값

void setup() {
  Serial.begin(9600);
  pinMode(3, OUTPUT);
  pinMode(2, INPUT);
}

void loop() {
  //만약 버튼이 눌리고, cnt가 5보다 작으면 실행
  if (digitalRead(2) == HIGH && cnt < 5) {
    pushValue += cnt; //pushValue를 cnt만큼씩 증가
    if (pushValue > 255) pushValue = 0; //pushValue가 255를
    넘으면 0으로 초기화
    Serial.println(pushValue); //pushValue값 확인
  }
  delay(10); //증가 속도 변화
  analogWrite(3, pushValue); //LED 밝기 변화
}
```

03 자료형



자료형이란 변수가 어떤 종류인지 명시 해 놓는 형식입니다. 변수에 들어갈 숫자가 크지 않은데, 큰 메모리를 사용하게 되면 그만큼 손해입니다.

하지만, 자료형을 사용하게 될 경우 필요한 양만큼 필요한 공간을 받아서 사용하게 되어 메모리를 효율적으로 사용할 수 있게 됩니다.

자주 사용하게 되는 자료형으로는 정수를 표현하는 int형과 문자를 표현하는 char형, 실수를 표현하는 double형 등이 있습니다.

자료형이란?

자료형

자료형이란 영어로는 Data Type으로 자료가 어떤 종류인지 구분하기 위해 씁니다. 간단히 말해 박스의 크기와 종류를 나타낸다고 할 수 있습니다.

자료형에는 다음과 같은 대표적인 종류가 있습니다.

자료형	크기	표현 가능한 데이터 범위
char	1byte	-128 ~ +127
short	2byte	-32768 ~ +32767
int	4byte	-2147483648 ~ +2147483647
long	4byte	-2147483648 ~ +2147483647
float	4byte	$3.4 \times 10^{-37} \sim 3.4 \times 10^{38}$
double	8byte	$1.7 \times 10^{-307} \sim 1.7 \times 10^{308}$
unsigned char	1byte	0 ~ 127+128
unsigned short	2byte	0 ~ 32767+32768
unsigned int	4byte	0 ~ 2147483648 + 2147483648
unsigned long	4byte	0 ~ 2147483647 + 214783648

<그림3-1> 자료형 종류

컴퓨터는 모든 정보를 숫자로 보관합니다. 만약 자료형이 나뉘어져 있지 않다면, 컴퓨터는 사용자로부터 어떤 크기의 데이터를 입력 받을지 모르기에 저장 공간을 필요 이상으로 확보해 놓아야 합니다.

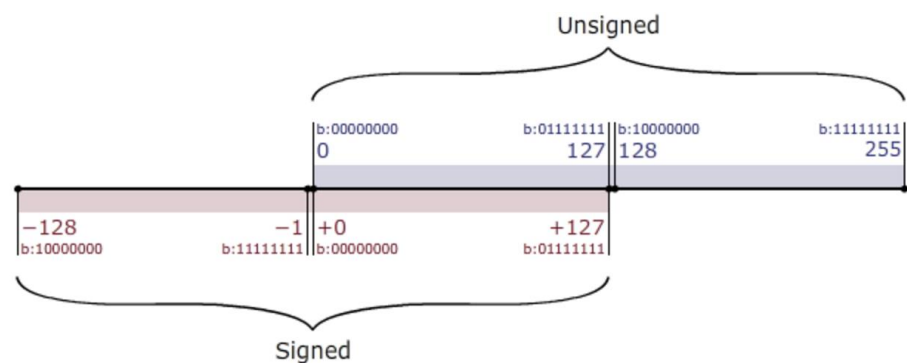
또한, 자료형을 명시해 놓으면 어떤 형태의 데이터가 들어와야 하는지 미리 알 수 있습니다.

부호

자료형에는 부호가 있는 것과 없는 것이 있습니다. 부호란 +와 -로 양수와 음수를 뜻합니다.

Signed가 부호가 있다는 뜻이고, unsigned가 부호가 없다는 뜻입니다. 이런 부호를 나타낼 때도 1비트가 사용됩니다.

그래서 만약 8비트로 숫자를 표현한다고 하면 부호가 있다고 한다면 -128에서 127까지의 숫자를 표현할 수 있고, 부호가 없다고 하면 0에서 255까지의 숫자를 표현할 수 있습니다.

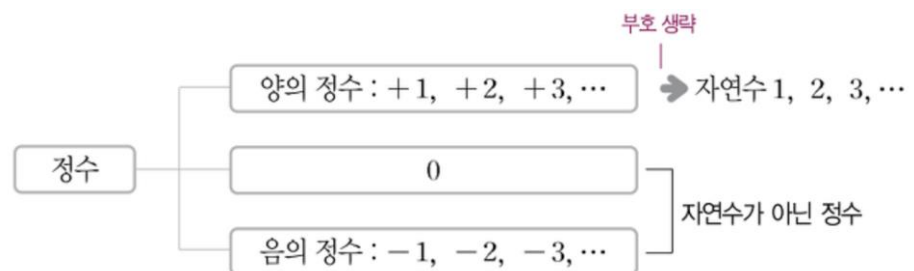


<그림3-2> signed와 unsigned

int형

Int형은 integer의 약자로 정수를 의미합니다.

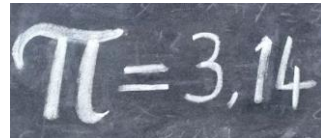
정수는 소수점이 없는 숫자로 -2, -23, 0, 3, 177 등의 숫자가 정수입니다. 앞에서 나온 short형과 long형 또한 정수를 표현할 때 사용하는데, 보통 int형을 사용합니다.



<그림3-3> 정수

float
double

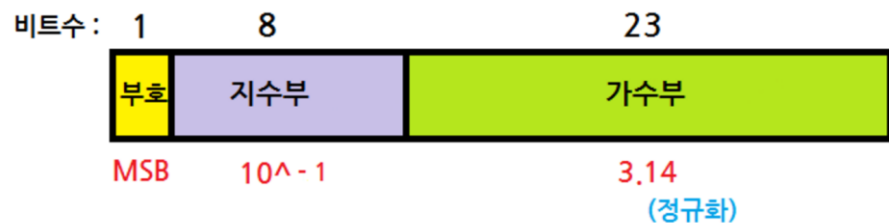
float형과 double형은 실수를 나타낼 때 사용됩니다. 정수 뒤 점 이후에 나오는 숫자들을 표현할 수 있습니다.



<그림3-4> 실수 예

실수형 변수는 정밀한 계산을 할 때 주로 사용하게 됩니다. 프로그래밍에서 소수를 표현할 때에는 부동소수점 오차 방식이 사용됩니다.

31.4를 부동소수점으로 표현할 때,



<그림3-5> 부동소수점 표현

부동소수점이란 소수점의 자리가 계속 움직이게 되어 적은 수로도 큰 숫자를 표현할 수 있는 방법입니다. 하지만, 표기법이 다양할 수 있기에, 한 가지 방법으로 표현하자는 약속인 정규화를 거치게 됩니다. 프로그래밍 시엔 정수 부분을 한 자리만 표현합니다.

char형

Char형은 character형의 약자로, 문자를 뜻합니다. Char형은 8비트로 char형의 숫자 처리 범위는 -128에서 127까지입니다.
($2^8 = 256$ 가지의 숫자를 나타낼 수 있습니다.)

보통 `char a = 'a';` 의 형태로 사용하게 되며, 문자를 저장할 때 쓰입니다.

앗, 그런데 Char형의 범위는 -128에서 127까지인데, 어떻게 문자를 저장할 수 있는 걸까요?

ASCII Code

ASCII 코드

아스키 코드는 숫자로 문자를 표현하기 위한 일종의 약속입니다. 컴퓨터는 숫자밖에 받아들이지 못하므로, 특정 숫자를 기호와 연결하여 표현하는 방법입니다.

제어 문자			공백 문자			구두점			숫자			알파벳		
10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`			
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a			
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b			
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c			
4	0x04	EOT	36	0x24	\$	68	0x44	D	100	0x64	d			
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e			
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f			
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g			
8	0x08	BS	40	0x28	(72	0x48	H	104	0x68	h			
9	0x09	HT	41	0x29)	73	0x49	I	105	0x69	i			
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j			
11	0x0B	VT	43	0x2B	+	75	0x4B	K	107	0x6B	k			
12	0x0C	FF	44	0x2C	,	76	0x4C	L	108	0x6C	l			
13	0x0D	CR	45	0x2D	-	77	0x4D	M	109	0x6D	m			
14	0x0E	SO	46	0x2E	.	78	0x4E	N	110	0x6E	n			
15	0x0F	SI	47	0x2F	/	79	0x4F	O	111	0x6F	o			
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p			
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q			
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r			
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s			
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t			
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u			
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v			
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w			
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x			
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y			
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z			
27	0x1B	ESC	59	0x3B	;	91	0x5B	[123	0x7B	{			
28	0x1C	FS	60	0x3C	<	92	0x5C	\	124	0x7C				
29	0x1D	GS	61	0x3D	=	93	0x5D]	125	0x7D	}			
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~			
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL			

<그림3-6> 부동소수점 표현

자료형 실습

- ① 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

ch3_4_3_char_serial

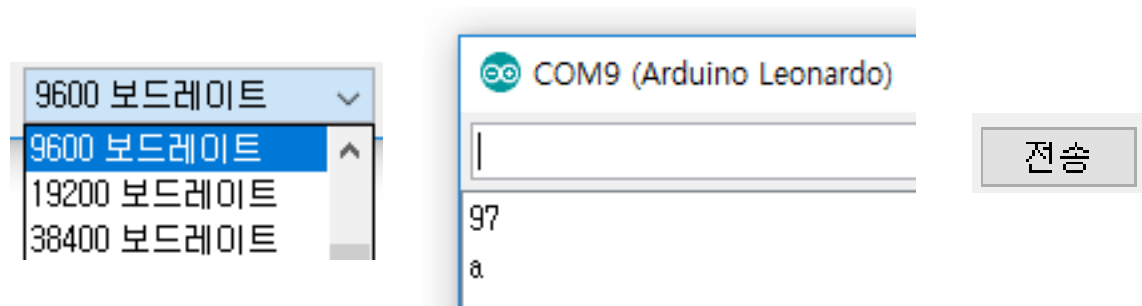
```

1 void setup() {
2     Serial.begin(9600);
3     pinMode(3, OUTPUT);
4 }
5
6 void loop() {
7     if (Serial.available()) {
8         char data = Serial.read();
9
10        if (data == 97) digitalWrite(3, HIGH);
11        else digitalWrite(3, LOW);
12
13        int data2 = data;
14
15        Serial.println(data2);
16        Serial.write(data2);
17        Serial.println();
18    }
19 }
```

<그림3-7> 자료형 실습

- ②  버튼을 눌러 시리얼 모니터를 켭니다.

- 3 보드레이트를 맞춘 후 어떤 알파벳을 전송했을 때 LED가 켜지는지 맞춥니다.



<그림3-8> 시리얼 통신 확인

- 4 되돌아온 문구를 확인합니다.

자료형 실습 해석

```
void setup() {  
  Serial.begin(9600); // 시리얼 통신 시작  
  pinMode(3, OUTPUT); // 3번 핀 출력으로 설정  
}  
  
void loop() {  
  if (Serial.available()) { // 만약 입력이 있으면  
    char data = Serial.read(); // data에 저장  
  
    if (data == 97) digitalWrite(3, HIGH); // data가 97과 같으면 켜  
    else digitalWrite(3, LOW); // 아니면 LED를 끄  
  
    int data2 = data; // data를 data2에 저장  
  
    Serial.println(data2); // data2 출력  
    Serial.write(data2); // data2 출력  
    Serial.println(); // 줄 바꿈  
  }  
}
```

```
8 | char data = Serial.read();
```

<그림3-9> data 저장

8번째 줄의 data에 들어오는 값은 'a'로 아스키 코드값으론 97입니다. 따라서 8번째줄은 char data = 'a';와 같고, 이것은 char data = 97;과 같습니다. 이로 인해 10번째 줄에서 LED에 불이 켜지게 됩니다.

```
13 | int data2 = data;
```

<그림3-10> data2 저장

이후 13번째 줄의 int data2 = data;를 통해 data2에는 97의 값이 들어가게 됩니다. 즉, int data2 = 97;과 같습니다.

```
15 | Serial.println(data2);
```

```
16 | Serial.write(data2);
```

<그림3-11> 시리얼 출력 2가지 방식

int형을 컴퓨터로 보낼 때 println을 사용하게 되면 97이 전송되고, write를 사용하게 되면 a가 전송되게 됩니다.

char vs int

char형에서는 Serial.println()을 사용하든, Serial.write()을 사용하든 값이 다 a로 보여집니다.

하지만, int형에서 Serial.println()를 사용하면 97를 둘로 나눠 9와 7의 아스키코드값인 57, 55를 보내고 컴퓨터는 이를 각각 9와 7로 해석하여 97을 보여줍니다.

int형에서 Serial.write()를 사용하면 97을 그대로 보내고 컴퓨터는 97을 아스키코드값으로 보고 해석하여 a를 보여줍니다.

꿀TIP

저장되는 데이터

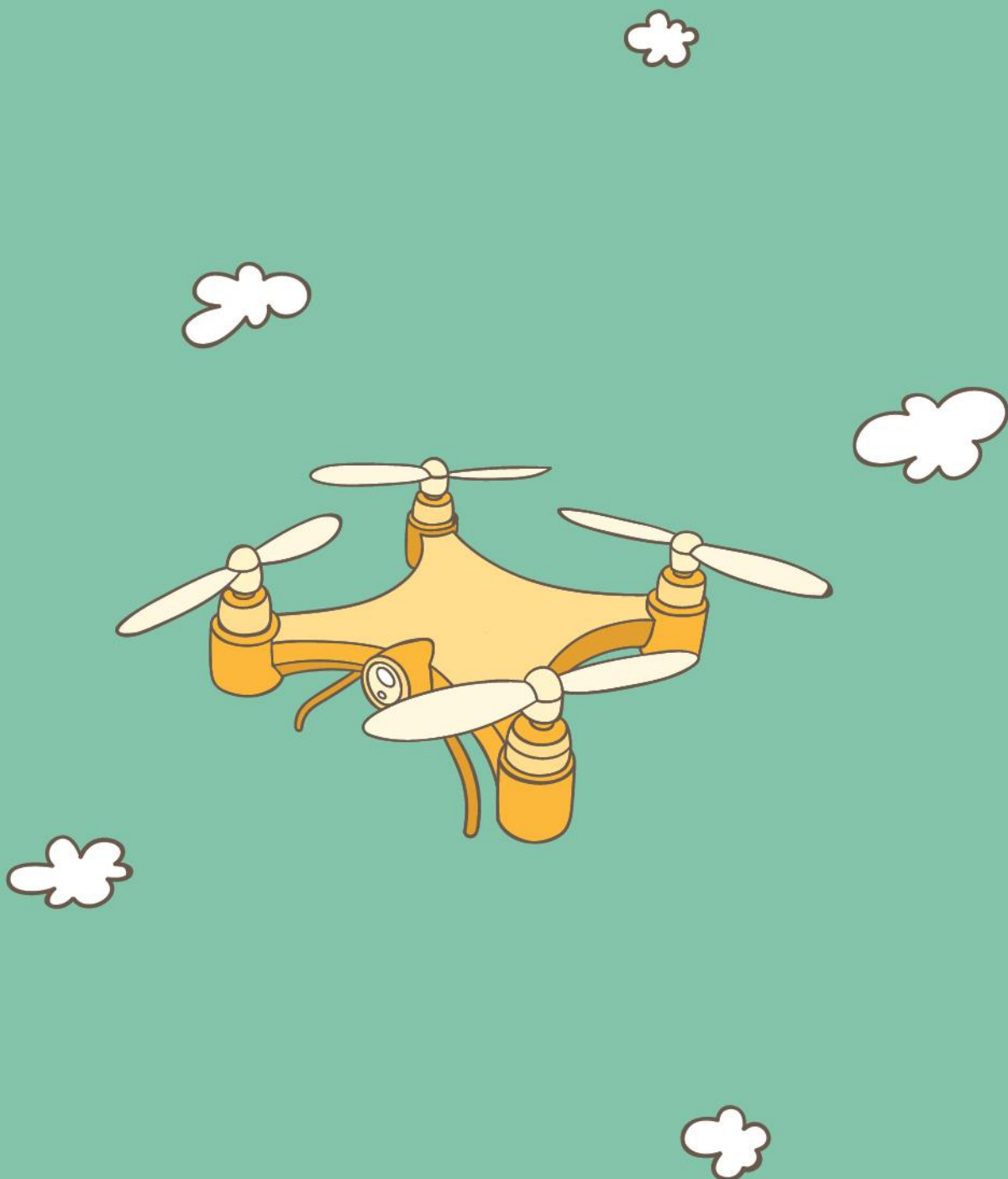
```
char data = 'a';
```

```
char data = 97;
```

두 경우 모두 data에는

97이라는 숫자가

저장되게 됩니다.



WHIT