



드론으로 배우는
프로그래밍 교실

캠프1h. 별첨자료



별첨 자료

코딩버드는 총 34시간의 수업으로 구성되어 있습니다.
밑에 나오는 내용들은 이번 시간에 다루지 못한 더 자세한 내용들입니다.
코딩에 대해 더 알고 싶은 학생은 다음 내용들을 살펴보고,
어떻게 프로그램이 동작하는지 배워가길 바랍니다.



드론으로 배우는
프로그래밍 교실

초판발행 2016년 9월 23일
지은이 최정애 | 펴낸이 최정애
펴낸곳 WHIT | 주소 안산시 한양대학교55 창업보육센터 B01
전화 010-5125-2139

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

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

3-1-1 아두이노와 친해지기



아두이노는 하드웨어를 다루기 쉽게 해주는 관련 도구 및 개발 환경입니다.
아두이노 이전에는 하드웨어에 대해 쉽게 접근하기 어려웠는데, 아두이노의 등장으로 손 쉽게 하드웨어 제품을 만들 수 있게 되었습니다.
납땀할 필요 없이 핀을 이용하여 쉽게 하드웨어를 제작하고, 레지스터 설정이 필요 없이 쉽게 프로그래밍이 가능합니다.
아두이노와 함께 DIY세계로 빠져 봅시다.

마이크로 컨트롤러란?

마이크로 컨트롤러

마이크로 컨트롤러는 보통 검은색의 작고 네모난 칩으로, 컴퓨터와 같은 역할을 합니다. 그림과 같이 생겼으며, 초소형 컴퓨터 역할을 하여 보통 작은 전자제품에 들어가 정해진 작업을 반복합니다.



<그림1-2> 마이크로 컨트롤러(atmega32u4)

활용분야

마이크로 컨트롤러는 생활 및 산업 전 영역에서 사용되고 있으며, 무궁무진한 응용이 가능합니다. 여러분의 집에 있는 세탁기, 라디오, 선풍기 등 다양한 전자제품에 사용됩니다.



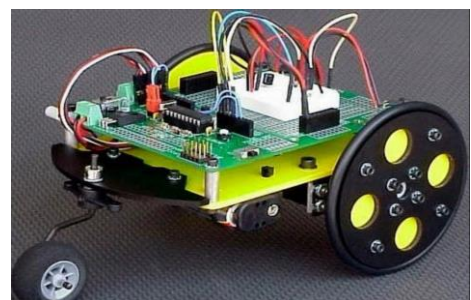
<그림1-3> 의료용 기구'



<그림1-4> 선풍기



<그림1-5> 리모콘



<그림1-6> 라인 트레이서

아두이노 활용

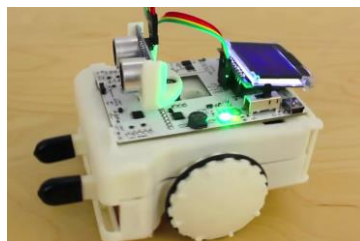
아두이노 스마트 전등



<그림1-11> 아두이노 스마트 전등

출처 : <https://www.youtube.com/watch?v=PHPU-xoFYgA>

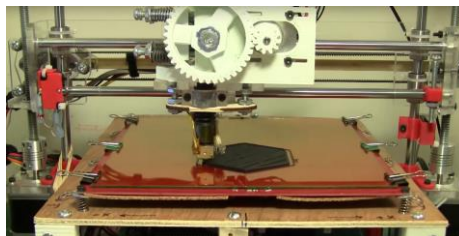
아두이노 로봇



<그림1-12> 아두이노 로봇

출처 : https://www.youtube.com/watch?time_continue=42&v=2igPl-MTfTQ

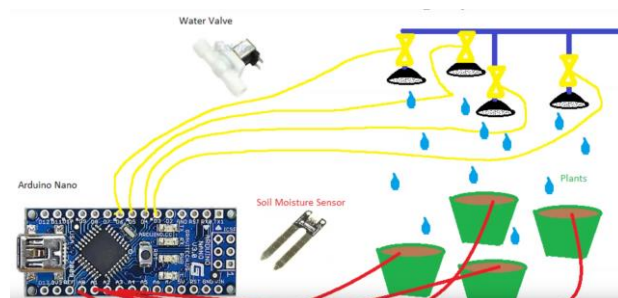
아두이노 3D 프린터



<그림1-13> 아두이노 3D 프린터

출처 : https://www.youtube.com/watch?time_continue=127&v=pyFZKc356eQ

아두이노 농업 자동화



<그림1-14> 아두이노 농업 자동화

출처 : <https://www.youtube.com/watch?v=z0WRY56i8RA>

3-1-2 아두이노 스케치



아두이노 스케치는 아두이노에 들어갈 소프트웨어를 구성할 수 있는 IDE입니다. IDE란 Integrated Development Environment의 약자로 통합 개발 환경라는 뜻입니다. 통합 개발 환경이란 개발을 하는데 필요한 것들을 지원해주는 것입니다.

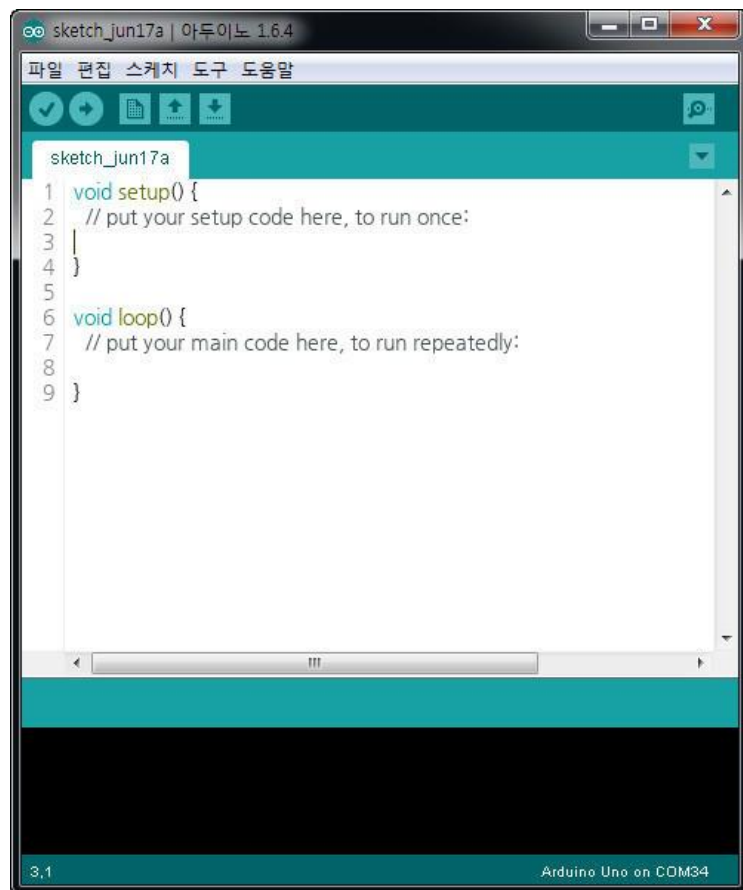
아두이노 스케치를 통해서 여러분들은 프로그래밍을 하게 되니까, 스케치를 잘 다룰 줄 알아야 합니다.

아두이노 스케치란?

아두이노
스케치

아두이노 스케치는 아두이노에 프로그래밍을 하기 위한 도구입니다. 메모장이 메모를 하기 위한 도구인 것처럼 말입니다.

메모장에 메모를 하듯 스케치에 프로그래밍을 하게 되는데, 이 때 이 스케치를 IDE(Integrated Development Environment: 통합 개발 환경)이라고 합니다.



<그림2-1> 아두이노 스케치

프로그래밍
업로드

앞으로 여러분은 아두이노 스케치를 이용해 프로그래밍을 하게 될 것입니다. 여기서 만들어진 프로그램이 실제 아두이노 프로 마이크로에 들어가게 될 것입니다. 이렇게 프로그램을 아두이노 보드에 집어 넣는 것을 업로드라고 합니다.

메뉴 바

메뉴바에는 File, Edit, Sketch, Tools, Help가 있습니다.

- File : 새 파일 작성, 열기, 저장, 닫기 및 텍스트 되돌리기 등의 기능이 있습니다. 중요 기능은 예제 불러오기인데, 각종 예제 파일들을 가져올 수 있어서 유용합니다.
- Edit : 텍스트 에디터 창에서 글을 잘못 썼을 경우 되돌리기, 주석 등의 편집 관련 기능을 제공합니다.
- Sketch : 컴파일 및 업로드 등의 기능을 제공합니다.
- Tools : 보드 선택, 포트 선택 등의 기능을 제공합니다.
- Help : 도움을 얻을 수 있습니다.

툴 바

툴 바에서는 컴파일, 업로드, 새 파일, 열기, 저장 등의 기능을 아이콘으로 제공하여 편리하게 기능을 이용할 수 있게 되어 있습니다. 보통 컴파일이나 업로드의 경우 메뉴바가 아닌 툴 바를 이용합니다.

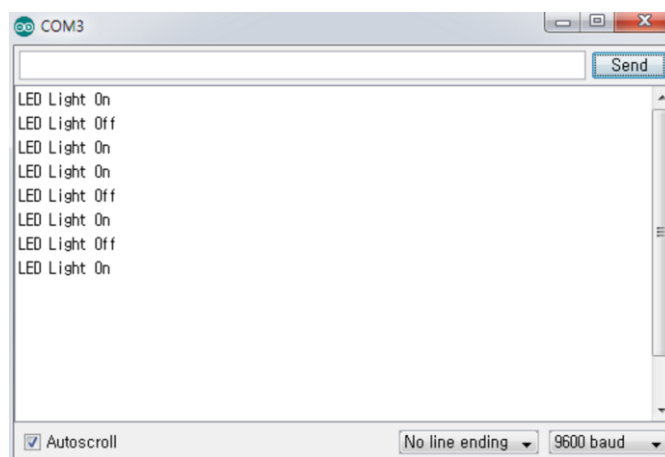


<그림2-3> 툴 바

시리얼 모니터



시리얼 모니터는 아두이노와 컴퓨터 간 통신을 하면서 서로 주고 받는 데이터를 눈으로 확인할 수 있는 대화의 창입니다. 보통 아두이노에 있는 데이터가 정확한 값이 맞는지 체크할 때 쓰입니다.



<그림2-4> 시리얼 모니터

꿀TIP

컴파일이란?

컴파일은 작성한 프로그램의 정상 동작을 위해 에러가 있는지 확인하는 과정입니다.

텍스트 에디터 창



<그림2-5> 텍스트 에디터 창

꿀TIP

줄 번호 표시

파일 - 환경설정에서
줄 번호 표시 네모박스에
V체크를 합니다.

☒ 줄 번호 표시

텍스트 에디터 창에서는 소스코드를 작성하고 수정할 수 있습니다.

좌측에는 줄 번호가 매겨져 있는데, 코드의 위치를 파악하기 쉽게 해 줍니다.

들여쓰기 등의 기능을 지원 해 줍니다.

콘솔 창



<그림2-6> 콘솔 창

콘솔 창에서는 스케치를 컴파일한 결과나 에러 등을 볼 수 있습니다.

컴파일이나 업로드 등에서 발생하는 문제에 대해 알려줌으로 해결 방안을 찾을 수 있습니다.

3-1-3 아두이노 프로그래밍 기초



아두이노의 기본 함수는 프로그램의 본격적인 시작 전 설정을 해 주는 `setup()`과 반복적인 일을 수행하는 `loop()`로 구성되어 있습니다.

아두이노의 기초 함수에 대해 알아보고, 함수가 무엇인지, 함수는 어떻게 구성되는지 배워 봅시다.

프로그래밍의 기초적인 상식으로 중괄호, 세미콜론, 블록 주석, 문장 주석에 대해 알아 봅시다.

함수 작성 해보기

뱀셈 함수
작성

다음 뱀셈 함수를 작성 해 봅시다.

ch3_1_3_sub

```

1 void setup() {
2     // put your setup code here, to run once:
3     sub(15, 9);
4 }
5
6 void loop() {
7     // put your main code here, to run repeatedly:
8 }
9
10 int sub(int a, int b){
11     return a-b;
12 }

```

<그림3-6> 뱀셈 함수



작성이 완료 되면 좌측 상단의 컴파일 버튼을 눌러 틀린 부분이 있는지 검사합니다.

이상이 없으면 컴파일 완료가 컴파일창에 나타납니다.

컴파일 완료.

스케치는 프로그램 저장 공간 221995 바이트(51%)를 사용, 최대 434160 바이트, 전역 변수는 동적 메모리 31568바이트(38%)를 사용, 50352바이트의 지역변수가 남음

ESP8266 Module 80 MHz 40MHz DIO 115200 512K (64K SPIFFS) ok Disabled None on COM3

<그림3-7> 컴파일 완료

곱셈 함수
작성

Q 세개의 숫자를 받아서 곱하는 함수는 어떻게 만들까요?

ch3_1_3_sub_mul

```

1 void setup() {
2     // put your setup code here, to run once:
3     sub(15, 9);
4     mul(4, 6, 5);
5 }
6
7 void loop() {
8     // put your main code here, to run repeatedly:
9 }
10
11 int sub(int a, int b){
12     return a-b;
13 }
14
15
16
17

```

꿀TIP

sub, mul

sub는 subtraction
mul은 multiplication
의 줄임말입니다.
코딩에선 줄임말을 많이
사용합니다.

<그림3-8> 곱셈 함수 작성 해보기

곱셈 함수

A 세개의 숫자를 받아서 곱하는 함수 정답

ch3_1_3_sub_mul

```
1 void setup() {  
2   // put your setup code here, to run once:  
3   sub(15, 9);  
4   mul(4, 6, 5);  
5 }  
6  
7 void loop() {  
8   // put your main code here, to run repeatedly:  
9 }  
10  
11 int sub(int a, int b){  
12   return a-b;  
13 }  
14  
15 int mul(int a, int b, int c){  
16   return a*b*c;  
17 }
```

<그림3-9> 곱셈 함수

Q 나눗셈 함수도 생각해 봅시다!

3-2-1 아두이노와 컴퓨터 연결



아두이노에 프로그램을 어떻게 전달할까요?

아두이노와 컴퓨터를 연결해주는 것은 하드웨어적으로는 USB케이블이 있고, 소프트웨어적으로는 드라이버가 있습니다.

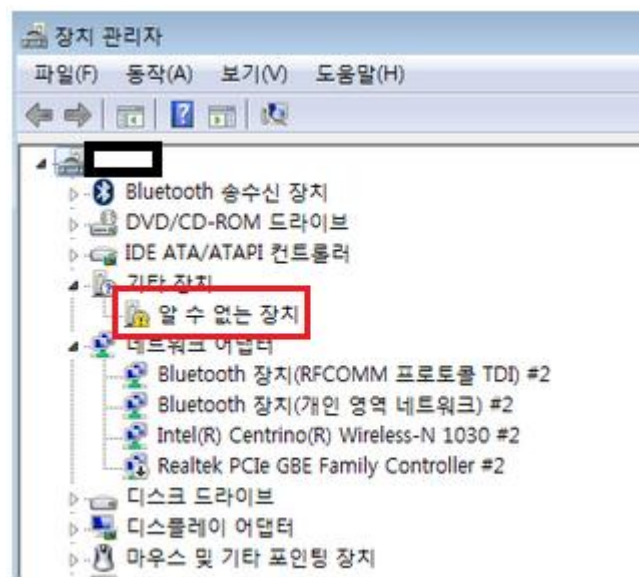
아두이노라는 장치를 컴퓨터에서 인식하기 위해서는 드라이버가 깔려 있어야 합니다.

아두이노 드라이버 업데이트

드라이버 업데이트

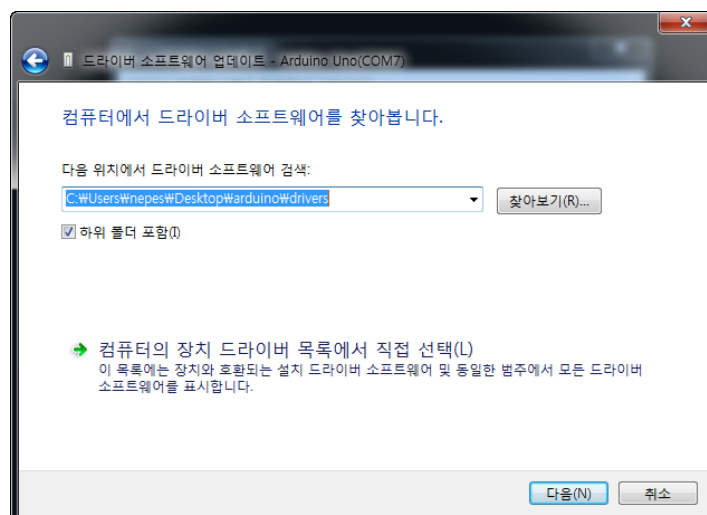
아두이노를 처음 연결할 경우 알 수 없는 장치가 연결되었다고 뜨는 경우가 종종 있습니다. 이럴 경우 장치 드라이버를 업데이트 시켜 줘야 합니다.

- 1 시작 – 장치관리자 를 통해 장치관리자를 엽니다.



<그림1-4> 장치 관리자

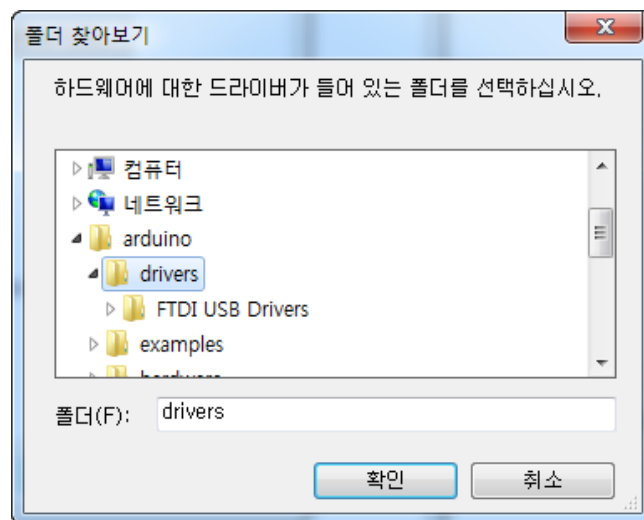
- 2 알 수 없는 장치를 우 클릭 합니다.
- 3 드라이버 소프트웨어 업데이트를 클릭합니다.



<그림1-5> 드라이버 소프트웨어 업데이트

컴파일 및 업로드

- 4 다운로드 받은 폴더 안의 drivers 까지의 경로를 입력합니다. Ex) C:\Users\m\desktop\arduino\drivers



<그림1-6> 드라이버 소프트웨어 업데이트

- 5 확인을 누르고 설치가 완료되길 기다립니다.

컴파일 업로드

컴파일은 작성된 코드가 아두이노로 들어갈 때 이상이 없는지 에러를 체크하는 과정입니다. 만약, 컴파일이 없다면 에러가 있는 프로그램이 들어가 아두이노를 망가뜨릴 수 있습니다.

업로드는 코드를 실제 아두이노에 넣는 것입니다. 아래 그림의 화살표 버튼을 누르면 컴파일과 업로드가 같이 진행됩니다.



<그림1-7> 컴파일 및 업로드 버튼

3-2-2 LED 깜빡이기 프로그램



아두이노를 처음 다뤄본다면 꼭 한번 접하게 되는 프로그램이 LED 깜빡이기입니다.

아두이노의 LED를 1초 간격으로 깜빡이는 예제를 통해서 아두이노라는 하드웨어를 어떻게 소프트웨어로 제어하는지에 대한 감을 잡을 수 있을 겁니다.

Setup()과 loop()

아두이노 기초 함수

아두이노 프로그램은 쉽게 이해할 수 있는 단순한 구조로 구성되어 있으며, 크게 2개의 파트로 나뉘어져 있습니다.

```
void setup()
{
    기본 세팅;
}
void loop()
{
    실행 함수;
}
```

setup()

setup()은 프로그램이 시작할 때, 사용됩니다.
pin mode를 초기화하거나 serial를 초기화합니다.
내용이 없어도 무조건 있어야 프로그램이 실행됩니다.

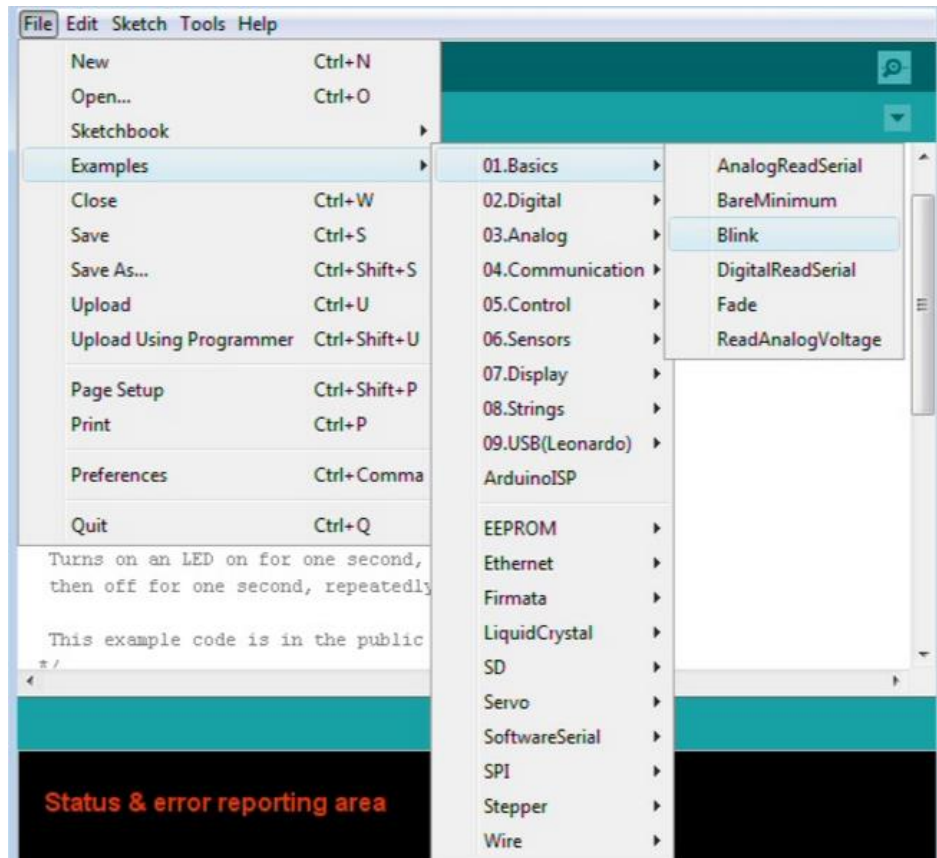
loop()

setup()실행 후, loop()함수는 반복적으로 정해진 명령어를 실행합니다.

<그림2-1> 아두이노 기초 함수

예제
가져오기

메뉴 바 - 파일 - 예제 - 01.Basics - Blink 를 통해서 LED 깜빡이기 예제를 가져 옵니다.



<그림2-3> LED Blink 예제 가져오기

아두이노에는 미리 작성되어 있는 다양한 예제가 있습니다.

프로그래밍을 처음 배울 때에는 무엇부터 해야 할지 모르는 경우가 많고, 어떻게 프로그래밍을 시작해야 할지 막막한 경우가 많습니다.
이럴 때 예제를 참고하면 도움이 많이 됩니다.

아두이노에서 기본적으로 제공 해주는 예제에는 Serial통신, LED Fade, analog 등 많은 예제들이 있습니다.

3-2-3 아두이노 하드웨어 구성



아두이노의 하드웨어는 어떻게 구성될까요?

아두이노 스케치를 통해서 소프트웨어는 제작하였는데, 실제로 LED를 깜빡이기 위해서는 하드웨어적인 구성이 필요합니다.

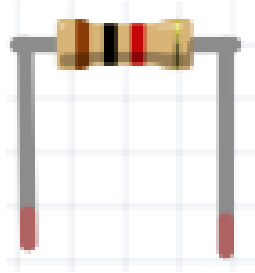
아두이노가 LED를 제어할 때는 핀을 사용합니다.

핀을 사용하여 입력과 출력을 받아 외부 센서를 제어하게 됩니다.

저항 값 읽기

저항값 읽기

Q 이 저항의 저항 값은?



_____ Ω

| 색 상 COLOR | 저 항 환 산 표 | | | |
|--------------|-----------|------|-------------|-------|
| | 첫째 수 | 둘째 수 | 셋째 수(곱하는 수) | 오차표시 |
| 검 정(흑색) | 0 | 0 | 1 | |
| 밤 색(갈색) | 1 | 1 | 10 | |
| 빨 강(적색) | 2 | 2 | 100 | |
| 주황색(동색) | 3 | 3 | 1000 | |
| 노 랑(황색) | 4 | 4 | 10000 | |
| 초록색(녹색) | 5 | 5 | 100000 | |
| 파랑색(청색) | 6 | 6 | 1000000 | |
| 보라색(자색) | 7 | 7 | 10000000 | |
| 회 색(회색) | 8 | 8 | 100000000 | |
| 흰 색(백색) | 9 | 9 | 1000000000 | |
| 금 색 | | | 0.1 | ± 5% |
| 은 색 | | | 0.01 | ± 10% |
| 무 색 | | | | ± 20% |

<그림3-14> 저항 값 계산 표

A 첫 번째 띠가 밤색(갈색)이므로 첫 번째 수는 1
 두 번째 띠가 검은색(흑색)이므로 두 번째 수는 0
 세 번째 띠가 빨강색(적색)이므로 곱하는 수는 100
 $10 * 100 = 1000\text{옴} = 1\text{K옴}$

꿀TIP

띠가 5개면

세 번째 띠까지 숫자를
 세고, 네 번째 띠를
 곱하는 수로 합니다.

Ex) 주주검검금
 $330 \times 1 = 330\text{옴}$

LED 속도 변화 해보기

LED 속도
변화

지금 아두이노는 1초간 켜지고 1초간 꺼지는 프로그램대로 동작하고 있습니다.

delay()의 괄호 안에 적절한 값을 적은 뒤 업로드하여 LED의 속도를 변화시켜 봅시다.



0.5초마다 깜빡이게 만들어보세요.

ch3_2_3_blink

```
1 void setup() {  
2     pinMode(3, OUTPUT);  
3 }  
4  
5 void loop() {  
6     digitalWrite(3, HIGH);  
7     delay(    );  
8     digitalWrite(3, LOW);  
9     delay(    );  
10 }
```

<그림3-15> LED속도 변화 해보기

LED 속도 변화 해보기 정답

A delay()의 괄호안에 1000 대신 500을 넣으면 0.5초마다 깜빡이게 됩니다.

delay()함수는 시간을 ms단위로 받기 때문에, 1000을 넣으면 1000ms = 1s로 1초가 되게 됩니다.

```
ch3_2_3_blink
1 void setup() {
2   pinMode(3, OUTPUT);
3 }
4
5 void loop() {
6   digitalWrite(3, HIGH);
7   delay(500);
8   digitalWrite(3, LOW);
9   delay(500);
10 }
```

<그림3-16> LED속도 변화 해보기 정답

3-3-2 조건문



조건문은 해당 조건이 참인지 거짓인지 판별하여 그에 따라 각각 다른 명령어를 실행시키고자 할 때 사용됩니다.

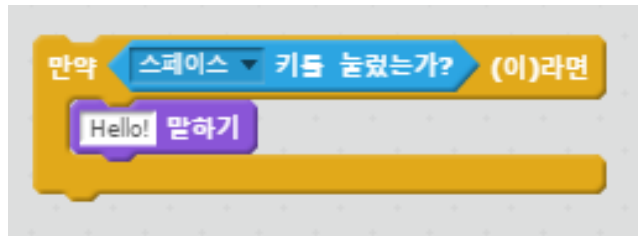
조건문은 프로그래밍의 기본인 순차, 조건, 반복의 셋 중 하나로 가장 빈번하게 사용됩니다.

조건문을 잘 쓰면 상황에 맞는 명령을 내릴 수 있습니다.

if문이란

if문은 "만약 ~라면 ~한다"입니다. 특정 조건에 해당되면 미리 정해 놓은 명령을 실행할 수 있습니다. 이러한 if문을 조건문이라고 합니다.

예를 들어 다음과 같이 "만약 스페이스키가 눌렸는가?" 같은 조건문이 있고, 스페이스바를 누르면 해당 명령문인 "Hello! 말하기"가 실행 됩니다.



<그림2-1> 조건문 스크래치 예시

위와 같은 기능을 아두이노에서는 다음과 같이 표현할 수 있습니다.

```

8 | if(digitalRead(2) == HIGH){
9 |     Serial.write("Hello!");
10| }
    
```

<그림2-2> 조건문 아두이노 예시

조건문 구성

아두이노에서의 조건문은 다음과 같이 구성됩니다.

| | | |
|-------|---------------------------------|------------|
| | | 조건이 참이면 |
| if 만약 | if (digitalRead(2) == HIGH) { | |
| | Serial.write("Hello!"); | 명령문 실행 |
| | } | |

<그림2-3> 조건문 구성

==기호의 의미

이 때 사용되는 == 기호는 좌변과 우변이 같은지를 판단하는 기호로써 =기호와 다른 기호입니다.

만약 ==기호의 좌변과 우변이 같다면 참을 반환하게 되고 if문의 중괄호 안에 있는 명령어가 실행되게 됩니다.

else if

else if를 사용하면 조건을 여러 번 검사할 수 있습니다.

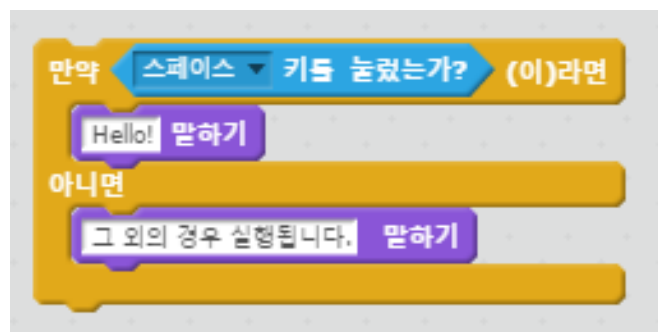
```
9 | if(digitalRead(2) == HIGH){  
0 |   Serial.write("Hello!");  
1 | } else if( value > 5){  
2 |   Serial.write("value가 5보다 큼니다.");  
3 | }
```

<그림2-4> else if 구성

위 코드에선 if문의 조건이 참인지 거짓인지 판별하고, 조건이 거짓일 경우엔 두 번째 조건을 검사하게 됩니다.

else

else는 if문의 조건이 해당되지 않는 경우 명령문을 실행합니다.



```
9 | if(digitalRead(2) == HIGH){  
10 |   Serial.write("Hello!");  
11 | } else{  
12 |   Serial.write("그 외의 경우 실행됩니다.");  
13 | }  
14 | }
```

<그림2-5> else

if
else if
else

if와 else if, else를 사용하면 다양한 경우의 조건문을 만들어낼 수 있습니다.

```
9 | if (digitalRead(2) == HIGH) {  
10 |     Serial.write("Hello!");  
11 | } else if (value > 5) {  
12 |     Serial.write("value가 5보다 큽니다.");  
13 | } else {  
14 |     Serial.write("그 외의 경우 실행됩니다.");  
15 | }
```

<그림2-6> if else if else

단, 이 때 주의할 사항으로는 다음과 같습니다.

- 조건문의 시작은 if문이어야 합니다.
- else if문은 여러 번 들어갈 수 있습니다.
- else는 가장 마지막에 나와야 합니다.

논리연산자

if문의 조건은 다양한 경우의 수가 나올 수 있습니다.

- 두 가지의 조건이 모두 참일 때 명령어 실행
- 두 가지의 조건 중 하나만 참이어도 명령어 실행
- 조건이 참이 아닐 때 명령어 실행

이럴 때 사용할 수 있는게 논리 연산자입니다.

1 AND논리

If(x > 0 && x < 5)

// x가 0보다 크고 또한 5보다 작을 때만 참

2 OR논리

If(x > 0 || y < 5)

// x가 0보다 크거나 또는 y가 5보다 작으면 참

3 !논리

If(!x > 0) // 조건이 거짓일때만 참

꿀TIP

|| 기호

|기호는 바라고 불리며,
보통 키보드의 엔터키
위에 있습니다.

if문 작성 해보기

if문 작성
해보기

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.


ch3_3_2_if_serial

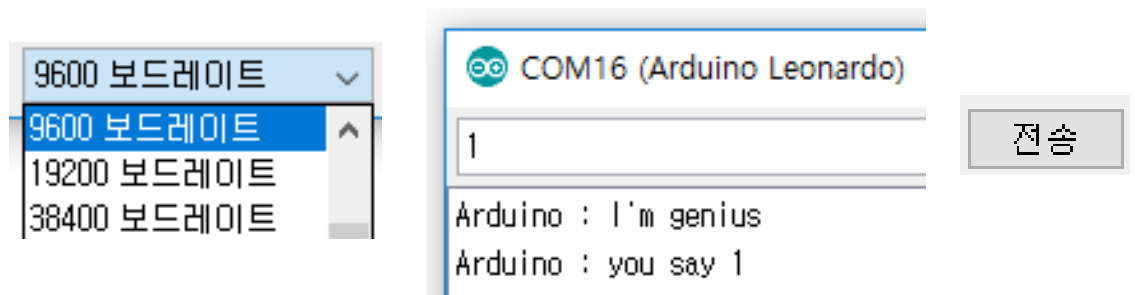
```

1 void setup() {
2   Serial.begin(9600);
3 }
4
5 void loop() {
6   if (Serial.available()) {
7     Serial.write("Arduino : I'm genius #n");
8     int val = Serial.read();
9
10    if (val == '0') {
11      Serial.write("Arduino : you say ");
12      Serial.write(val);
13    } else if (val == '1') {
14      Serial.write("Arduino : you say ");
15      Serial.write(val);
16    } else {
17      Serial.write("Arduino : what? ");
18    }
19
20    Serial.write("#n");
21  }
22 }
```

<그림2-7> if문 예제

if문 해석

- 2  버튼을 눌러 시리얼 모니터를 켵니다.
- 3 보드레이트를 맞춘 후 원하는 숫자를 적은 뒤 전송 버튼을 클릭합니다.



<그림2-8> 시리얼 통신 확인

- 4 1또는 0을 전송했을 때와 그 외의 값을 전송했을 때 차이를 비교해 봅니다.

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

void loop() {
  if (Serial.available()) { //만약 사용자의 입력 있다면
    Serial.write("Arduino : I'm genius\n"); //문구 출력
    int val = Serial.read(); //사용자의 입력을 변수에 저장

    if (val == '0') { //만약 입력값이 0과 같다면
      Serial.write("Arduino : you say ");
      Serial.write(val);
    } else if (val == '1') { //만약 입력값이 1과 같다면
      Serial.write("Arduino : you say ");
      Serial.write(val);
    } else { //조건이 전부 맞지 않다면
      Serial.write("Arduino : what? ");
    }

    Serial.write("\n"); //줄 바꿈
  }
}
```

3-3-3 변수 알아보기



변수는 변하는 수입니다. 암산으로 풀 수 없는 수학문제를 풀 때 공책에 풀이 과정과 숫자를 적게 됩니다. 컴퓨터에서도 계산을 할 때 숫자를 임시로 적어 놓는데, 이 적어 놓은 숫자가 어떤 형태인지, 얼마큼의 크기를 가지는지 파악하기 위해 변수를 사용합니다. 공책은 메모리에 해당하게 됩니다. 변수를 사용하는 법을 익혀서 프로그래밍을 해 봅시다.

변수 정의
특성

변수란 변하는 수로, 항상 같은 수인 상수와 대비되는 개념입니다.

컴퓨터 소스코드에서 일반적으로 데이터 저장위치와 그 안의 내용과 관련되어 있는 것들입니다.

변수를 선언하여 데이터를 저장함과 동시에 컴퓨터에게 이 만큼의 메모리 공간을 빌려 달라고 말하는 것과 같습니다.



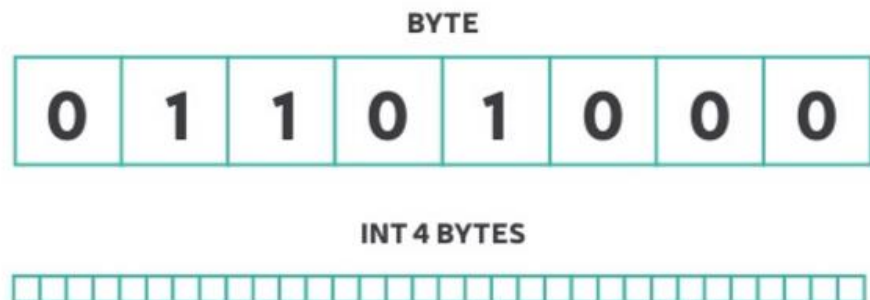
<그림3-1> 메모리에 데이터 저장

위 그림에서는 **car**라는 이름의 변수(상자)에 **Toyota**라는 값을 저장 해 두었습니다. 그 변수(상자)에 들어있는 값을 꺼낼 때는 **car**라는 변수명을 이용하면 됩니다.

정수형
변수 **int**

컴퓨터 메모리에는 0 또는 1이 들어갈 수 있는 방식이 무수히 많이 있습니다.

정수형 데이터는 4byte(32bit)를 차지하여 **int x;** 를 실행할 경우 총 32칸의 방식이 만들어집니다.



<그림3-2> int 자료형의 4bytes

변수의 종류

지역변수

전역변수

변수의 종류로는 지역변수와 전역변수가 존재합니다.

지역변수는 함수 내(중괄호 안)에서만 존재하며, 함수 밖에서는 사용할 수 없습니다.

전역변수는 함수 밖(중괄호 밖)에서 선언되어 존재하며, 함수 내부, 외부 둘 다에서 쓰일 수 있습니다.

```
int global =100;

void setup()
{
}

void loop()
{
}

void myFunction()
{
    int local =100;
}
```

global은 전역 변수로
함수의 외부에 존재하며
어떤 함수에서도 사용
가능합니다.

local은 지역 변수로
myFunction()이라는 함수
안에서 존재하며 밖에서는
사용할 수 없습니다.

<그림3-3> 전역변수, 지역변수

전역변수의 경우 사용이 편리하다는 장점이 있지만, 여러 함수에서 사용할 경우 값이 나도 모르게 바뀔 수 있다는 단점이 있습니다.

지역변수의 경우 함수 내에서만 사용이 가능하지만, 함수가 종료됨과 동시에 메모리에서 할당이 해제됩니다.

변수 선언 및 초기화

변수 선언

변수의 선언은 다음과 같이 자료형, 변수명, 세미콜론으로 구성 됩니다.

```
int    a    ;
```

int의 자료형을 사용하며 **a**라는 이름을 지정

변수 초기화

변수 선언 후에는 보통 변수에 초기값을 넣어줍니다. 이를 변수의 초기화라고 합니다.

```
a    =    100    ;
```

a라는 이름의 변수에 **100**이라는 값을 저장

변수 선언 및 초기화

아래처럼 변수의 선언과 초기화를 한번에 할 수도 있습니다.

```
int    a    =    100    ;
```

int의 자료형을 사용하여 **100**의 값을 **a**에 저장
a라는 이름을 지정하여

변수 선언 규약

- 변수 이름 선언 시 반드시 알파벳으로 시작해야 합니다.
- 알파벳 뒤에 숫자는 첨가하여 표현할 수 있고 공백은 불가능합니다.
- 변수의 이름은 중복되어 사용해서 안됩니다.
- 알파벳 대소문자 사용여부는 상관없으나 둘은 구분이 됩니다. 예를 들어 **A**와 **a**는 서로 다른 변수입니다.

변수명에 의미를 담아 보자

변수의 의미

사람의 이름에는 의미가 있습니다. 변수도 마찬가지로, 변수의 이름을 만들 땐 변수가 어떤 데이터를 가지고 있는지 파악하기 쉽게 만듭니다.

a, b 처럼 아무 의미 없는 변수명을 지으면 프로그램을 만들면서도 헛갈리고 나중에 보면 이해가 잘 안될 수 있습니다.

변수명 짓기



다음의 변수를 직접 작성 해 보면서 변수 이름을 어떻게 짓는지 익혀 봅시다.

ch3_3_3_variable

```
1 int sensorValue;
2 int pushButton = 2;
3 int buttonState;
4 int ledPin = 3;
5 int brightness = 0;
6 int fadeAmount = 5;
```

<그림3-4> 변수 작성 예제

카멜 표기법

변수의 이름을 지을 때 사용하는 변수 표기법에는 헝가리안, 파스칼, 언더바, 카멜 등의 대표적인 표기법이 있습니다.

그 중 카멜 표기법은 낙타의 혹처럼 중간에 대문자가 들어가는 형식입니다.

변수명의 맨 처음은 소문자로 시작하여 복합어의 경우에는 두 번째 단어가 시작할 때 대문자를 사용하여 표기합니다.

3-4-2 연산과 친해지기

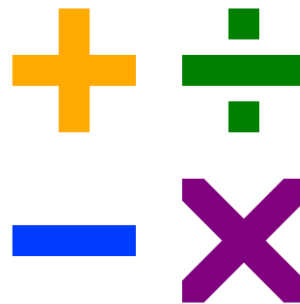


프로그래밍에서의 연산자는 수학에서의 연산자와는 약간 차이가 있습니다.
수학에서의 연산자와 프로그래밍에서의 연산자의 차이에 대해 알아보고,
프로그래밍에서의 연산자의 특성을 알아 봅시다.
나아가 프로그래밍 연산자를 직접 써보며 어떻게 사용되는지 감을
잡아봅시다.

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

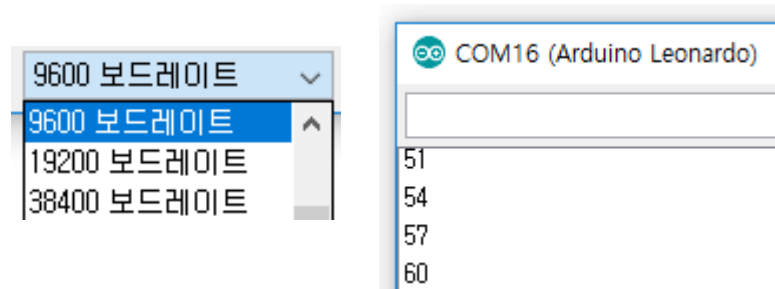
```
1 int pushValue = 0;
2 int cnt = 3;
3
4 void setup() {
5     Serial.begin(9600);
6     pinMode(3, OUTPUT);
7     pinMode(2, INPUT);
8 }
9
10 void loop() {
11     if (digitalRead(2) == HIGH && cnt < 5) {
12         pushValue += cnt;
13         if (pushValue > 255) pushValue = 0;
14         Serial.println(pushValue);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

3-4-3 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.

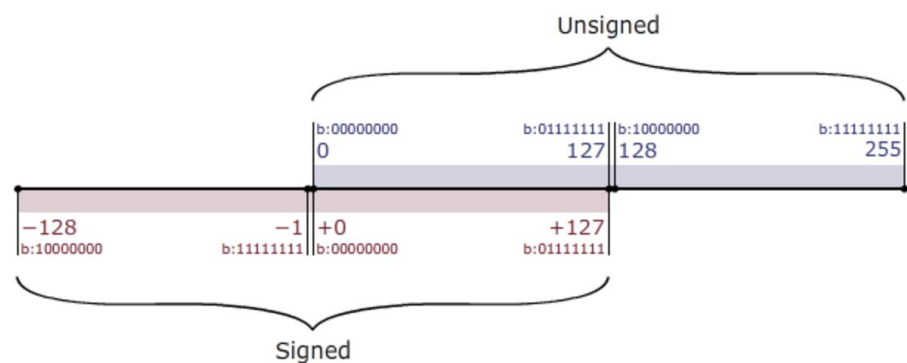
또한, 자료형을 명시해 놓으면 어떤 형태의 데이터가 들어와야 하는지 미리 알 수 있습니다.

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.

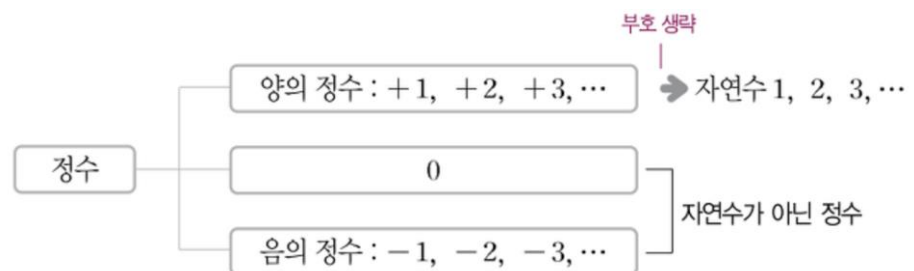


<그림3-2> signed와 unsigned

int형

Int형은 integer의 약자로 정수를 의미합니다.

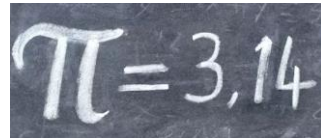
정수는 소수점이 없는 숫자로 -2, -23, 0, 3, 177 등의 숫자가 정수입니다. 앞에서 나온 short형과 long형 또한 정수를 표현할 때 사용하는데, 보통 int형을 사용합니다.



<그림3-3> 정수

float
double

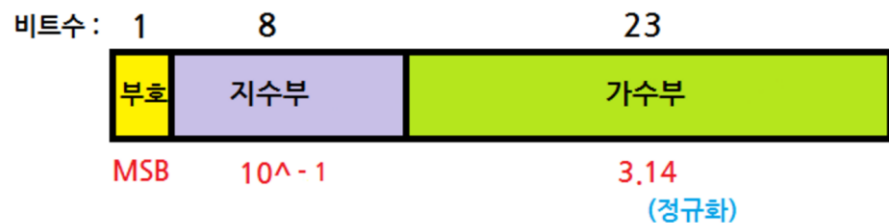
float형과 double형은 실수를 나타낼 때 사용됩니다. 정수 뒤 점 이후에 나오는 숫자들을 표현할 수 있습니다.



<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.

```

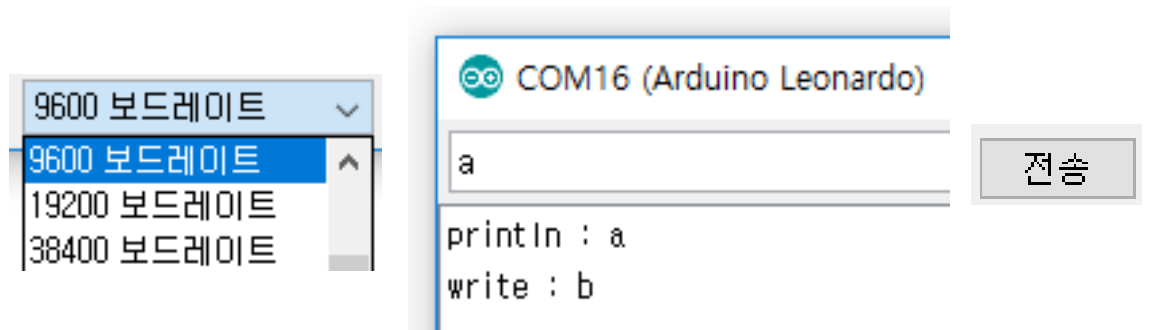
ch3_4_3_sw_led_char_serial
1 void setup() {
2     Serial.begin(9600);
3     pinMode(3, OUTPUT);
4 }
5
6 void loop() {
7     if (Serial.available()) {
8         char data = Serial.read();
9
10        if(data == 'a') digitalWrite(3, HIGH);
11        else digitalWrite(3, LOW);
12
13        Serial.print("println : ");
14        Serial.println(data++);
15        Serial.print("write : ");
16        Serial.write(data);
17        Serial.write('\n');
18    }
19 }

```

<그림3-7> 자료형 실습

- ②  버튼을 눌러 시리얼 모니터를 켭니다.

- 3 보드레이트를 맞춘 후 원하는 알파벳을 적고 전송 버튼을 클릭합니다.



<그림3-8> 시리얼 통신 확인

- 4 되돌아온 문구를 확인합니다.

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

void loop() {
    if (Serial.available()) { // 사용자의 입력이 있다면
        char data = Serial.read(); //입력을 읽어 옴

        if(data == 'a') //만약 data가 'a'와 같다면
            digitalWrite(3, HIGH); //LED 켜
        else digitalWrite(3, LOW); //그 외의 경우 LED 끄

        Serial.print("println : "); //문구 출력
        Serial.println(data++); //data값 출력 후 data값 1증가
        Serial.print("write : "); //문구 출력
        Serial.write(data); //data값 출력
        Serial.write("\n"); //줄 바꿈
    }
}
```

data에 들어오는 값은 'a'로 아스키 코드값으론 97입니다. 즉, char data = 97;과 같습니다. 아스키 코드97을 컴퓨터로 보내면, 컴퓨터는 이를 a로 해석합니다. 이후, 97을 98로 증가시킨 후 컴퓨터로 보내면 컴퓨터는 이를 b로 해석하게 됩니다.

자료형 실습2

- 1 다음과 같이 코드를 변형하여 아두이노에 업로드합니다.

ch3_4_3_sw_led_char_serial2

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

<그림3-8> 시리얼 통신 확인

data에 들어오는 값은 '1'로 아스키코드값으론 49입니다. 즉, int data = 49;과 같습니다. char에서와 다른 점은 println시 int형에서는 49를 둘로 나눠 4와 9의 아스키코드값인 52, 57을 보낸다는 것입니다. 컴퓨터는 이를 4와 9로 해석하여 49를 보여줍니다. write에서는 50을 보내고 이를 2로 해석합니다.



WHIT