



드론으로 배우는
프로그래밍 교실

Ch3. 아두이노 기초1 - 3



❖ 목차 ❖

01 Serial 통신	01
시리얼 통신이란?	02
하드웨어 시리얼	04
Serial통신 실습하기	05
02 조건문	07
if문	08
if문 작성 해보기	11
03 상수에 대해서	13
상수란?	12
상수 작성 해보기	13



드론으로 배우는
프로그래밍 교실

초판발행 2016년 9월 23일
지은이 최정애 | 펴낸이 최정애
펴낸곳 WHIT | 주소 안산시 한양대로55 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.

01 Serial 통신



Serial 통신은 아두이노 통신의 기초입니다.

아두이노는 Serial통신 방식을 사용하여 컴퓨터와 의사소통을 하게 됩니다.

아두이노에 프로그램을 업로드할 때 뿐 아니라, 아두이노에 들어있는 데이터를 컴퓨터 상에서 확인할 때에도 Serial통신을 사용하게 됩니다.

Serial통신을 숙지하여 아두이노와 즐겁게 이야기 해 봅시다.

시리얼 통신이란?

Serial
통신

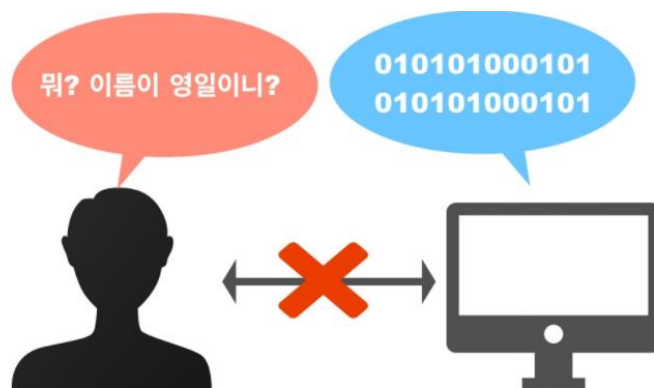
아두이노와 대화를 하려면 어떻게 해야 할까요?
컴퓨터와 의사소통을 하기 위해선 모니터, 키보드, 마우스 같은 도구가 필요합니다.

아두이노에서는 주로 시리얼통신이라는 방법을 통해 아두이노와 정보를 주고 받을 수 있습니다.

Serial
통신

Serial 통신은 영어 해석 그대로 직렬 통신으로, 한번에 한 비트씩 보내는 통신 방식입니다.

아두이노는 한국어를 알아듣지 못하기 때문에, 모든 정보를 0과 1의 비트 단위로 보내주어야 합니다.



<그림1-1> 사람과 아두이노의 언어 차이

Serial
통신

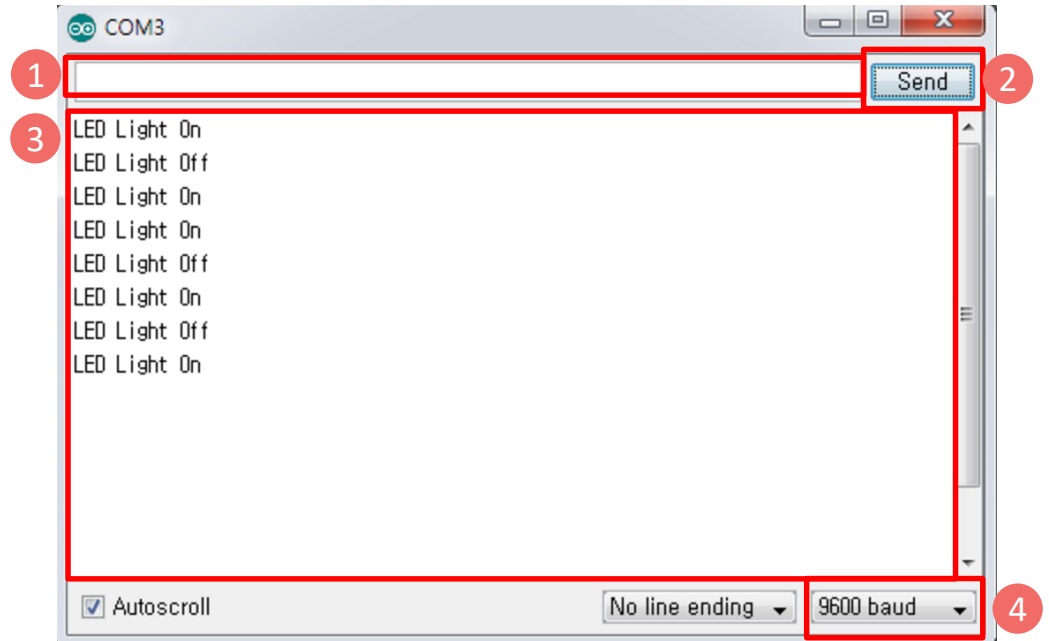
우리가 직접 모든 데이터를 0과 1로 바꿀 순 없습니다.
이런 작업을 편하게 해 주는게 시리얼 통신 함수입니다.

함수	설명
Serial.begin()	Baudrate를 정하여 통신을 알립니다.
Serial.available()	아두이노 버퍼에서 64byte까지 읽어오며 읽어올 byte가 없을 시 -1을 반환합니다.
Serial.read()	아두이노 버퍼에서 1byte만큼 데이터를 읽은 뒤 삭제하며 읽어올 byte가 없을 시 -1을 반환합니다.
Serial.write(val)	val 안 데이터를 TX핀을 통해 송신 후, 데이터의 크기를 반환합니다.

시리얼 모니터



시리얼 통신으로 주고 받는 데이터는 시리얼 모니터를 통해서 확인할 수 있습니다.

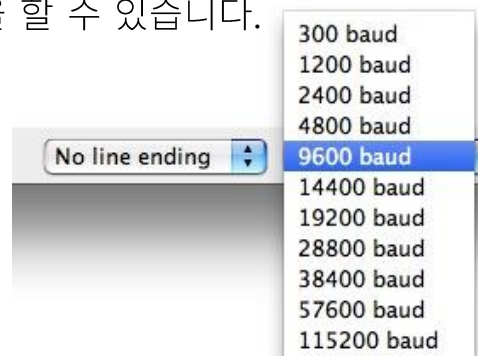


<그림1-2> 시리얼 모니터

- ① 텍스트 입력 창 : 아두이노로 보낼 데이터를 입력할 수 있습니다.
- ② 전송 : 아두이노로 데이터를 전송합니다.
- ③ 콘솔 창 : 아두이노로부터 받은 데이터를 확인할 수 있습니다.
- ④ 보드레이트 : 아두이노와의 통신 속도를 정할 수 있습니다.

보드레이트 (Baudrate)

보드레이트는 시리얼 통신을 할 때의 통신 속도를 나타냅니다. 보내는 쪽과 받는 쪽에서 속도가 같아야지만 서로 통신을 할 수 있습니다.



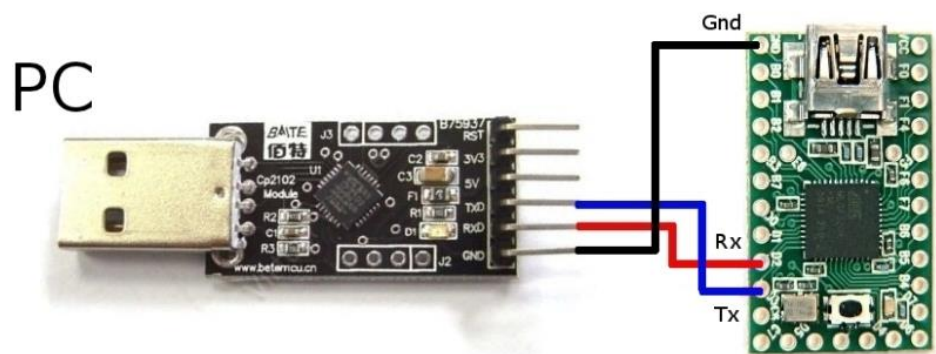
<그림1-3> 시리얼 모니터의 보드레이트 설정

HW Serial SW Serial

HW Serial 통신 특징

아두이노의 시리얼 통신에는 하드웨어 시리얼과 소프트웨어 시리얼이 있는데, 우리는 하드웨어 시리얼만 다룰 것입니다.

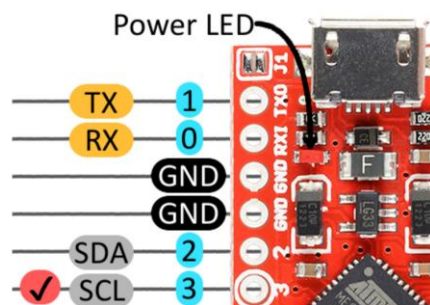
- 기본적으로 1:1 통신 방식입니다.
- 한 번에 한 비트씩 통신하는 직렬 통신입니다. (여러 비트를 동시에 보내는 병렬 통신과 상반된 개념)
- 시리얼 통신에는 USB 또는 RX, TX선이 사용됩니다.
- 시리얼 통신 시 디지털 0, 1번 핀은 사용 불가합니다.



<그림1-4> USB, RX, TX 연결

0, 1번 핀 사용불가

시리얼 통신을 할 때에는 아두이노의 0번 핀과 1번 핀을 사용할 수 없습니다. 그 이유는 바로 0번 핀과 1번 핀이 RX, TX와 같은 핀을 사용하기 때문입니다.



<그림1-5> 0, 1번 핀과 RX, TX 중첩


Serial통신 실습하기

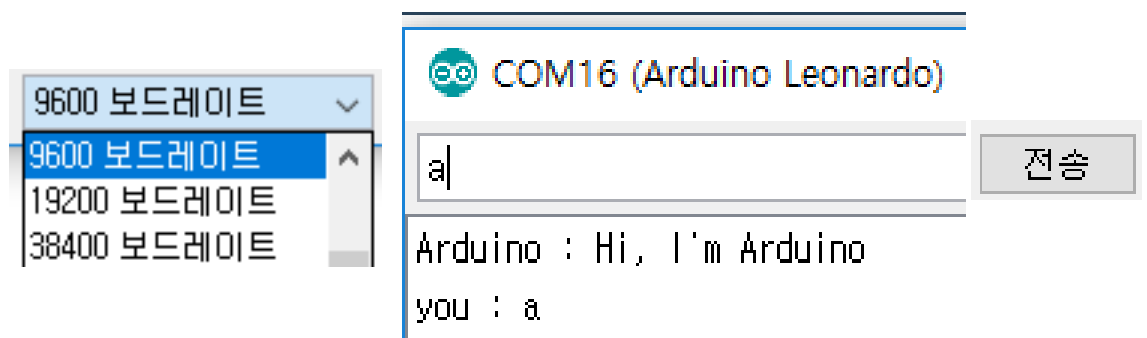
시리얼 통신 해보기

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

```
ch3_3_1_serial
1 void setup() {
2   Serial.begin(9600);
3 }
4
5 void loop() {
6   if (Serial.available()) {
7     Serial.write("Arduino : Hi, I'm Arduino");
8     Serial.write('\n');
9
10    Serial.write("you : ");
11    Serial.write(Serial.read());
12    Serial.write('\n');
13  }
14 }
```

<그림1-6> 시리얼 통신 코드

- 2  버튼을 눌러 시리얼 모니터를 켭니다.
- 3 보드레이트를 맞춘 후 원하는 알파벳을 적고 전송 버튼을 클릭합니다.



<그림1-7> 시리얼 통신 확인

- 4 되돌아온 문구를 확인합니다.

시리얼 통신 해석

```
void setup() {  
  Serial.begin(9600); //9600의 보드레이트로 시리얼 통신 실행  
}  
  
void loop() {  
  if (Serial.available()) { //만약 사용자의 입력이 있다면  
    Serial.write("Arduino : Hi, I'm Arduino"); //문구 출력  
    Serial.write('\n'); //줄 바꿈 표시  
  
    Serial.write("you : "); //문구 출력  
    Serial.write(Serial.read()); //사용자의 입력을 출력  
    Serial.write('\n'); //줄 바꿈  
  }  
}
```

시리얼 통신으로 아두이노와 대화를 해봤습니다. 감이 좀 잡히시나요??
시리얼 통신은 아두이노에 들어있는 데이터를 확인하거나
에러를 체크할 때 유용하게 쓰입니다.

아두이노야 말해봐

> 아두이노가 원하는 말을 할 수 있도록 코딩해보세요!!

```
ch3_3_1_serial2  
1 void setup() {  
2   Serial.begin(9600);  
3 }  
4  
5 void loop() {  
6   if (Serial.available()) {  
7     Serial.write("Arduino : 원하는 말을 적어보세요");  
8     Serial.write('\n');  
9   }  
10 }
```

<그림1-8> 아두이노야 말해봐

02 조건문



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

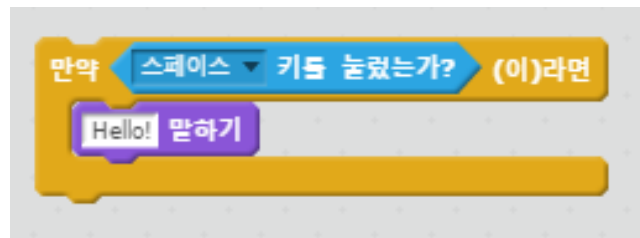
조건문은 프로그래밍의 기본인 순차, 조건, 반복의 셋 중 하나로 가장 빈번하게 사용됩니다.

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.

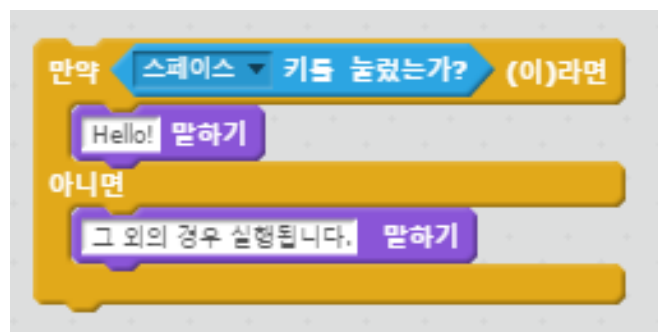
```
9 | if(digitalRead(2) == HIGH){  
0 |   Serial.write("Hello!");  
1 | } else if( value > 5){  
2 |   Serial.write("value가 5보다 큼니다.");  
3 | }
```

<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 큼니다.");  
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


```

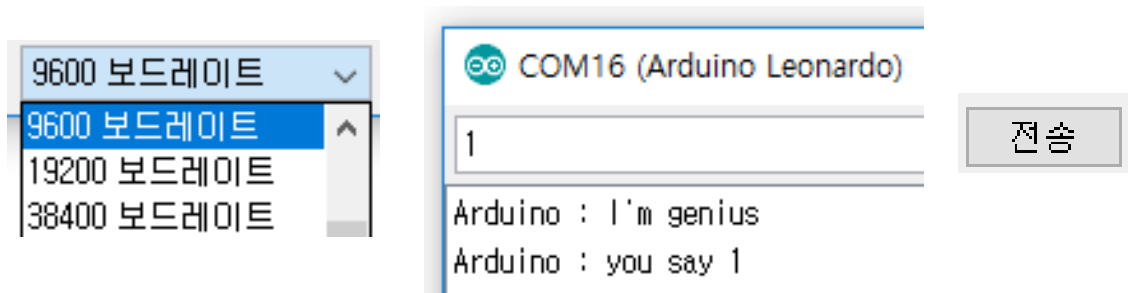
1 void setup() {
2     Serial.begin(9600);
3 }
4
5 void loop() {
6     if (Serial.available()) {
7         Serial.write("Arduino : I'm genius #n");
8         int val = Serial.read();
9
10        if (val == '0') {
11            Serial.write("Arduino : you say ");
12            Serial.write(val);
13        } 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 켭니다.
- 3 보드레이트를 맞추는 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
  }
}
```

03 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.



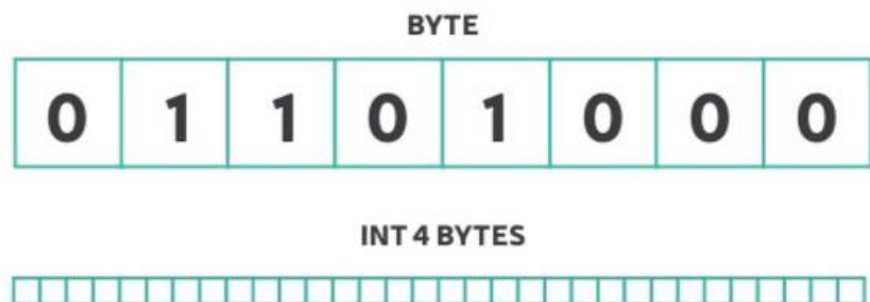
<그림3-1> 메모리에 데이터 저장

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 방이 만들어집니다.



<그림3-2> int 자료형의 4bytes

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.



WHIT