3. 변수

변수라는 것은 "변할 수 있는 수"로 어떤 값을 저장할 수 있는 공간을 의미합니다. 이를테면 빈 박스와 같습니다. 이 박스에는 사과도 담을 수 있고, 배도 담을 수 있습니다. 어떤 박스는 크기가 작아서 계란 하나만 담을 수 있습니다. 또 어떤 박스는 플라스틱 밀폐용기로 되어 있어서 물을 담을 수 있습니다.

변수는 컴퓨터 메모리에 차지하는 공간을 의미합니다. 거기에 작은 숫자만 저장할 수 있는 변수도 있고, 큰 숫자를 저장할 수 있는 변수도 있습니다.

변수를 쓰기 위해서는 먼저 변수를 쓰겠다고 알려야 합니다. 이걸 '선언' 이라고 하는데, 어려운 용어를 기억할 필요는 없습니다.

```
int value;
```

이렇게 value 라는 변수를 쓰겠다고 알리고 만드는 것이 사용하는 것보다 먼저 있어야 합니다.

```
int value;
value = 13;
```

이렇게 만들어진 변수 value 에 원하는 값 13 을 넣었습니다. 이제부터 value 라는 상자를 들여다보면 거기엔 13 이라는 값이 담겨있게 됩니다.

a. 변수 선언

변수 선언은 변수를 쓰기 전에 반드시 먼저 해야 합니다.

```
int value;
```

처럼 쓰기 전에 변수를 만들어야 합니다.

```
int value = 120;
```

처럼 변수를 만들면서 변수의 값을 넣어주어도 됩니다.

변수의 종류는 **byte, char, int, long, float** 외에도 몇 가지가 더 있습니다. char 은 문자를 다룰 때, int 는 정수형 숫자를 다룰 때, long 은 정수형 숫자로 큰 수를 다룰 때, byte 는 작은 숫자로 양수만 다룰 때, float 은 실수형 숫자를 다룰 때 사용합니다. 변수의 사용법은 이후에 다루도록 하겠습니다.

b. 전역 변수, 지역 변수

위에서 선언된 변수는 어디에 있느냐에 따라 전역변수 또는 지역변수로 다시 분류됩니다. 함수 안에서 만들어졌으며 지역변수라고 불리고 그 함수 안에서만 사용이 됩니다.

```
1 void func()
2 {
3     int i = 10;
4     Serial.println(i);
5 }
```

그리고, 함수가 끝나면 지역변수는 사라집니다. 위에서 func() 함수 밖에서 i 를 사용하면 에러가 나옵니다.

반면, 전역변수는 함수 밖에서 만들어집니다.

```
1 int i = 10;
2 void func()
3 {
4     Serial.println(i);
5 }
```

함수가 끝나도 함수 밖에서 만들어진 전역변수는 계속 남아있습니다.

지역변수는 함수 안에서만 사용되므로 같은 이름이 다른 함수 안에서 새롭게 만들어지고, 사용될 수 있습니다.

```
1  void func_i()
2  {
3     int i = 10;
4     Serial.println(i);
5  }
6
7  void func_j()
8  {
9     int i = 20;
10    Serial.println(i);
11 }
```

전역변수는 사용에 주의해야 합니다.

전역변수와 지역변수가 사용되는 곳을 알아봅시다.

```
1  int globalvalue;
2  void setup(){
3    Serial.begin(9600);
4  }
5
6  void loop(){
7     int b;
8     for (int a=0; a<10; a++) {
9           Serial.println(a);
10       }
11 }
```

전역변수인 **globalvalue** 는 어디서나 사용할 수 있습니다. 지역변수인 **a** 는 for 문 안에서만 사용할 수 있습니다. 역시 또 다른 지역변수인 **b** 는 loop() 안에서만 사용할 수 있습니다.

```
    void loop(){                                    // 1
        int b=10;                                   // 2
        for (int a=0; a<10; a++) {                  // 3
              Serial.print("a=");                   // 4
              Serial.println(a);                    // 5
          }                                         // 6
          Serial.print("a=");                       // 7
          Serial.print(a);                          // 8
          Serial.print("  b=");                     // 9
          Serial.print(b);                          // 10
    }                                               // 11
```

위의 예제를 실행하면 8번째 줄에서 에러가 납니다. 변수 a 는 3번째 줄, for 문에 포함되어 있습니다. for 문의 { } 문단 안에서만 변수 a 는 사용할 수 있습니다. 즉, 3번 줄에서 6번 줄을 벗어나서는 a를 쓸 수 없습니다. 억지로 사용하려고 하면 에러를 발생합니다.

여러 함수 안에서 변수의 값을 읽거나 쓰려면 전역변수를 사용하면 됩니다. 물론 그 방법 말고도 다른 좋은 방법이 있습니다만 여기서는 전역변수를 사용하는 방법을 알려드립니다.

```
int globalvalue;                                    // 1
void setup(){                                       // 2
    Serial.begin(9600);                             // 3
    globalvalue = 10;                               // 4
}                                                   // 5
void loop(){                                        // 6
    int b=10;                                       // 7
    globalvalue++;                                  // 8
        Serial.print("globalvalue=");                // 9
        Serial.print(globalvalue);                  // 10
}                                                   // 11
```

함수 외부에서 선언된 전역변수 globalvalue 는 어떤 함수 안에서도 사용이 가능합니다. globalvalue 를 setup() 함수에서 초기 값을 준 뒤, loop()함수에서 반복될 때마다 1씩 증가시켜보았습니다. 이처럼 전역변수는 여러 함수에서 읽고 쓸 수 있습니다.

c. 변수의 종류

byte

byte 변수는 0 부터 255 까지의 정수 숫자를 저장할 수 있습니다. byte 변수는 1개의 바이트를 사용합니다. 1 바이트는 8개의 비트의 모음입니다. 일반적으로 아두이노 같은 마이크로프로세서는 8비트, 즉 1바이트를 기본단위로 합니다. 1비트는 전기가 흐르거나 흐르지 않는 하나의 정보를 보관할 수 있는 장소입니다. 1 비트는 0 또는 1을 보관할 수 있는 공간입니다. 정보가 보관된 곳을 1로 표시하고 비어있는 것을 0으로 표시하면 다음과 같습니다.

이진수	십진수	계산
0000 0000	0	$= 0\times2^7+ 0\times2^6+ 0\times2^5+ 0\times2^4+ 0\times2^3+ 0\times2^2+ 0\times2^1+ 0\times2^0$
0000 0001	1	$= 0\times2^7+ 0\times2^6+ 0\times2^5+ 0\times2^4+ 0\times2^3+ 0\times2^2+ 0\times2^1+ \mathbf{1}\times2^0$
0000 0010	2	$= 0\times2^7+ 0\times2^6+ 0\times2^5+ 0\times2^4+ 0\times2^3+ 0\times2^2+ \mathbf{1}\times2^1+ 0\times2^0$
0000 0011	3	$= 0\times2^7+ 0\times2^6+ 0\times2^5+ 0\times2^4+ 0\times2^3+ 0\times2^2+ \mathbf{1}\times2^1+ \mathbf{1}\times2^0$
0000 0100	4	$= 0\times2^7+ 0\times2^6+ 0\times2^5+ 0\times2^4+ 0\times2^3+ \mathbf{1}\times2^2+ 0\times2^1+ 0\times2^0$
…	…	…
1111 1101	253	$= \mathbf{1}\times2^7+ \mathbf{1}\times2^6+ \mathbf{1}\times2^5+ \mathbf{1}\times2^4+ \mathbf{1}\times2^3+ \mathbf{1}\times2^2+ 0\times2^1+ \mathbf{1}\times2^0$
1111 1110	254	$= \mathbf{1}\times2^7+ \mathbf{1}\times2^6+ \mathbf{1}\times2^5+ \mathbf{1}\times2^4+ \mathbf{1}\times2^3+ \mathbf{1}\times2^2+ \mathbf{1}\times2^1+ 0\times2^0$
1111 1111	255	$= \mathbf{1}\times2^7+ \mathbf{1}\times2^6+ \mathbf{1}\times2^5+ \mathbf{1}\times2^4+ \mathbf{1}\times2^3+ \mathbf{1}\times2^2+ \mathbf{1}\times2^1+ \mathbf{1}\times2^0$

8개의 방, 비트를 가지고 셀 수 있는 수는 0부터 255까지 총 256 개가 됩니다.

byte 변수를 만들 때 다음과 같이 씁니다.

```
byte sample = 125;
```

byte 변수는 최대 255까지 담을 수 있습니다. 여기에 255 가 넘는 수가 들어오면 다시 0부터 시작됩니다.

```
byte sample = 255;          //   1111 1111
sample = sample + 1;        //   1111 1111 + 1

// 1 0000 0000 이 되고, 아래 8개의 비트만 변수에 저장
// sample 변수에 0000 0000 이 저장
```

```
byte sample = 0;
while(1) {
    sample++;
    Serial.println(sample);
}
```

위 내용을 실행시키면 0부터 255까지 화면에 나오고 다시 0부터 재 시작됩니다. sample 라는 변수가 byte 변수여서 8비트의 저장 공간을 가지고 있기 때문에 그 이상의 비트는 버려지게 됩니다.

char

char 변수는 키보드에 나와 있는 문자나 숫자 하나를 저장할 수 있습니다. 단, 한글을 저장할 수는 없습니다. 문자나 숫자를 저장할 때 ' ' 를 사용합니다. 8비트, 한 바이트짜리 저장공간을 사용합니다. 숫자 저장할 때는 127에서 -128까지 저장할 수 있습니다.

byte 와 char 는 동일한 저장 공간 크기인 1바이트를 가집니다. 단 byte 는 부호를 가지지 않아서 0부터 255까지 사용하지만 char 은 첫 번째 비트를 부호로 사용하기 때문에 7비트로 표시 가능한 0부터 127까지 사용합니다. 그리고 첫 비트가 1일 때는 음수를 표시합니다. -1부터 -128까지 가능합니다.

```
void setup(){
    Serial.begin(9600);
}
void loop(){
  char sample = 'A';
  Serial.println(sample);
}
```

char sample = 'A';

char sample = 65;

이진수	십진수	계산
XXXX XXXX	-128~127	X는 0 또는 1
= **X**×2^7+ **X**×2^6+ **X**×2^5+ **X**×2^4+ **X**×2^3+ **X**×2^2+ **X**×2^1+ **X**×2^0		

이진수	십진수	이진수	십진수
0000 0000	0	1000 0000	-128
0000 0001	1	1000 0001	-127
0000 0010	2	1000 0010	-126
0000 0011	3	1000 0011	-125
0000 0100	4	1000 0100	-124
…	…		
0111 1101	125	1111 1101	-3
0111 1110	126	1111 1110	-2
0111 1111	127	1111 1111	-1

TIP

[2진수와 음수를 나타내기 위한 2의 보수]

전기의 흐름, 즉 비트의 0과 1로만 숫자를 나타낼 수 있는 이진수로 디지털에서 숫자를 처리합니다. 2진수는 각 자리수마다 다음과 같은 방식으로 수를 셉니다. 8비트 이진수는 0000 0000부터 1111 1111 까지입니다. 이진수는 다음과 같이 10진수로 바꿀 수 있습니다.

이진수 : XXXX XXXX = $X\times2^7+ X\times2^6+ X\times2^5+ X\times2^4+ X\times2^3+ X\times2^2+ X\times2^1+ X\times2^0$
(X 는 0 또는 1)

일반적으로 음수는 숫자의 앞에 '-' 표시를 붙입니다. 하지만 디지털 이진수에서 음수를 표현하는 방법은 조금 특별합니다. 음수를 표시하기 위해 "2의 보수"라는 것을 사용합니다.
1 과 -1을 더하면 0이 됩니다. 이것을 이용해서 1과 더해서 0이 되는 수를 찾으면 됩니다. 1은 아두이노에서 이진수로 표현할 때 B 0000 0001 이 됩니다. 이진수 00000001에서 계속 증가하면 최종적으로 11111111까지 됩니다. 11111111에서 1이 증가하면 (1)000000000이 됩니다. 여기서 8비트를 넘어선 9비트째인 1은 버리고, 아래쪽 8개의 비트만 남게 되어 00000000 즉, 0이 됩니다. 다시 말해서 B11111111 에 1을 더하면 0이 됩니다. 그래서 1의 대한 음수표현인 -1은 디지털 이진수에서 11111111 이 됩니다.

0000 0001 과 1111 1111을 더하면 1 0000 0000 즉, 0이 됩니다. -1은 1111 1111입니다.
0000 0010 과 1111 1110을 더하면 1 0000 0000 즉, 0이 됩니다. -2는 1111 1110입니다.
0000 1111 과 1111 0001을 더하면 1 0000 0000 즉, 0이 됩니다. -15는 1111 0001입니다.

2의 보수를 이용한 음수표현은 이진수에서 1과 0을 서로 바꾼 다음 1을 더하면 됩니다.

-1을 표현해 보겠습니다. 1은 0000 0001입니다. 여기서 0과 1을 바꾸면 1111 1110 이 됩니다. 여기에 1을 더하면 최종적으로 1111 1111 이 됩니다.
-12를 표현해 보겠습니다. 12의 이진수는 0000 1100입니다. 0과 1을 바꾸면 1111 0011 이 됩니다. 여기에 1을 더해서 만들어진 1111 0100 이 -12입니다.

int

int 변수는 16비트, 2바이트를 사용합니다. int 변수가 표현할 수 있는 숫자는 -32,768 부터 32,765 까지의 정수 숫자입니다. 일반적으로 아두이노에서 정수 숫자를 사용할 때 가장 많이 사용하는 변수입니다.

```
int sample = 10000;
```

long

long 변수는 4바이트를 사용합니다. lng 변수로 표현할 수 있는 정수는 -2,147,483,648 부터 2,147,483,647 까지의 정수입니다. int 로 표현할 수 있는 숫자보다 더 큰 숫자를 사용할 때 long 변수를 사용합니다.

```
int sample = 214748364;
int sample = 21474 * 9876;
```

float

float 변수는 -3.4028235E+38 부터 3.4028235E+38 까지의 실수 숫자를 저장합니다. float 변수는 4바이트를 사용합니다.
단 아두이노에서 실수 연산은 속도가 많이 느려지게 됩니다.
그리고 실수연산은 정확도가 많이 떨어집니다. 가능하면 float 같은

실수 변수를 사용하지 않는 것이 좋습니다.

float 변수는 총 32비트 중 부호 1비트, 지수부 8비트, 가수부 23비트를 사용합니다. 그래서 십진수로 따지면 7자리숫자가 유효숫자가 됩니다. float 변수는 -3.4028235×10^{38} ~ 3.4028235×10^{38} 사이의 숫자를 표현할 수 있습니다.

```
float sample = 3.1415;
```

String

엄밀하게는 변수는 아니지만 아두이노에서 String 은 마치 문자열을 저장하고 읽는 일을 하는 변수처럼 사용됩니다.

```
void setup(){
    Serial.begin(9600);
}
void loop(){
    String sample = "Remember 0416";
    Serial.print(sample);
}
```

4. 연산

a. "=" 의 의미

수학에서 "="은 보통 좌변과 우변이 같다는 것을 의미합니다. 하지만 프로그램에서 "=" 는 오른쪽에 있는 값을 왼쪽으로 넣으라는 명령입니다.

```
x = x + 10;
```

x 에 10 을 더한 다음 그 값을 x 에 저장합니다. 그러면 x 라는 저장공간, 변수는 처음 값보다 10이 커진 값으로 바뀝니다.

이것을 더 간단하게 이렇게 쓸 수 있습니다.

```
x += 10;
```

자주 사용하는 1 을 더하는 것과 1 을 빼는 것은 ++, -- 를 사용합니다.

```
x = x + 1;
x++;
```

위의 두 줄은 같은 의미입니다.

b. 기본 수학연산

+, -, *, /, % 이렇게 다섯 가지 기본 수학연산이 가능합니다.

+ : 더하기

- : 빼기

* : 곱하기

/ : 나누기

% : 나머지

위의 = 과 결합하여 다음과 같이 사용할 수 있습니다.

`result = result + a;`	`result += a;`	변수 result 에 a 를 더해서 저장합니다
`result = result - a;`	`result -= a;`	변수 result 에 a 를 빼서 저장합니다
`result = result * a;`	`result *= a;`	변수 result 에 a 를 곱해서 저장합니다
`result = result / a;`	`result /= a;`	변수 result 를 a 로 나누고 저장합니다
`result = result % a;`	`result %= a;`	변수 result 를 a 로 나눈 나머지를 저장합니다

c. 비교

크거나 작거나 같거나를 가지고 참과 거짓을 판별한다.

```
x == y     // x 가 y 와 같다.
```

만약 x 와 y 가 같은 값이라면 이 결과는 "진실"이 되고, x 와 y 가 다른 값이라면 이 결과는 "거짓"이 된다. 이 외에 다음과 같은 비교가 가능하다.

```
x > y      // x 가 y 보다 크다.
x >= y     // x 가 y 보다 크거나 같다.
x < y      // x 가 y 보다 작다.
x <= y     // x 가 y 보다 작거나 같다.
x != y     // x 와 y 가 다르다.
```

d. 논리

비교와 함께 자주 사용된다.

```
if ( (x > 0) && (x < 10) )  // x 가 0 보다 크다 그리고 x 가 10 보다 작다
```

&& : 논리곱, 오른쪽과 왼쪽이 모두 참일 때 결과는 참이 된다.

|| : 논리합, 오른쪽 또는 왼쪽, 둘 중 하나라도 참이면 결과는 참이다.

! : 부정, 참이면 거짓이, 거짓이면 참이 된다.

```
( x>0 && x<5 )   // x 가 0와 5 사이일 때 참
( x>10 || x <5 )  // x 가 10보다 크거나 5보다 작을 때 참
( !x )            // x 가 참이면 거짓이, 거짓이면 참이 된다.
```

"비교"와 "논리"는 보통 조건문과 함께 사용된다. if 또는 for 문 등과 함께 사용된다.

5. 상수

변수와 달리 상수는 항상 그대로 있는 수를 말합니다. 보통 변해서는 안 돼는 수를 미리 지정해두는 식으로 사용됩니다.

```
#define  SIZE  1024
```

#define 문을 사용, 프로그램의 가장 앞부분에서 사용할 수 있습니다. 보통은 가장 많이 사용되는 방법입니다.

또는 const 변수를 사용할 수 있습니다. 기존의 변수 앞에 const를 추가하면 그 변수는 읽을 수는 있지만 쓸 수 없는 상수가 됩니다.

```
const int SIZE 1024;
```

이렇게 const 를 붙여서 만들 수도 있습니다. 보통 상수는 변수와 다르다는 것을 나타내기 위해 이름을 대문자로 사용합니다. SIZE 라는 변수는 상수 1024를 저장합니다. 그리고 SIZE 는 읽을 수만 있지 쓸 수는 없습니다. 즉, 값이 1024 는 변경되지 않는 값입니다.

변수 대신 상수를 이용해서 프로그램을 하는 이유는 실수로 상수를 다른 값으로 변경시키는 것을 방지하기 위해서입니다. 함께 프로그램을 하는 다른 사람일 수도 있고, 혹은 어떤 함수에서 변수를 이용하면서 값을 변경시킬 수도 있습니다. 이러한 실수를 미연에 방지하기 위해 변수 대신 상수를 사용합니다.

6. 참과 거짓

(a == 12) 는 a 의 값을 읽어 와서 그것이 12 인지를 비교해 봅니다. 12라면 참을 12가 아니라면 거짓을 보냅니다. 참은 1 이 되고, 거짓은 0 이 됩니다.

```
if ( a == 12 )
{
   할일 ;
}
```

a 가 12 일 때 (a==12) 는 참(1)이 되고, 그래서 "할일"을 하게 됩니다.

```
if ( a == 12 )
{
   Serial.println("Great! A=12");
}
```

프로그램 안에 위 코드를 넣어주면 a 값이 12가 될 때 화면에

Great! A=12

라는 문구를 보여줍니다.

7. HIGH / LOW

아두이노에서 HIGH 는 1 을 의미하고 이는 ON 또는 5V 나 3.3V 의 전원이 들어가는 것을 의미합니다. 반대로 LOW 는 0 을 의미하고, OFF 나 0V 즉, GND 에 연결되었다는 것을 뜻합니다. 즉, 1 대신 HIGH를 사용하고 0 대신 LOW를 사용할 수 있습니다.

```
digitalWrite(13, HIGH);  // 13번 핀에서 5V(또는 3.3V) 를 내보냅니다.
digitalWrite(13, LOW);   // 13번 핀이 GND(0V) 에 연결됩니다.

digitalWrite(13, 1);  // 13번 핀에서 5V(또는 3.3V) 를 내보냅니다.
digitalWrite(13, 0);   // 13번 핀이 GND(0V) 에 연결됩니다.
```

위의 예에서 보듯이 HIGH 와 1은 동일하고 LOW 와 0 도 동일합니다. 1 과 0 대신 HIGH 와 LOW를 써서 이유는 사람이 보기 쉽게 알아볼 수 있도록 합니다.