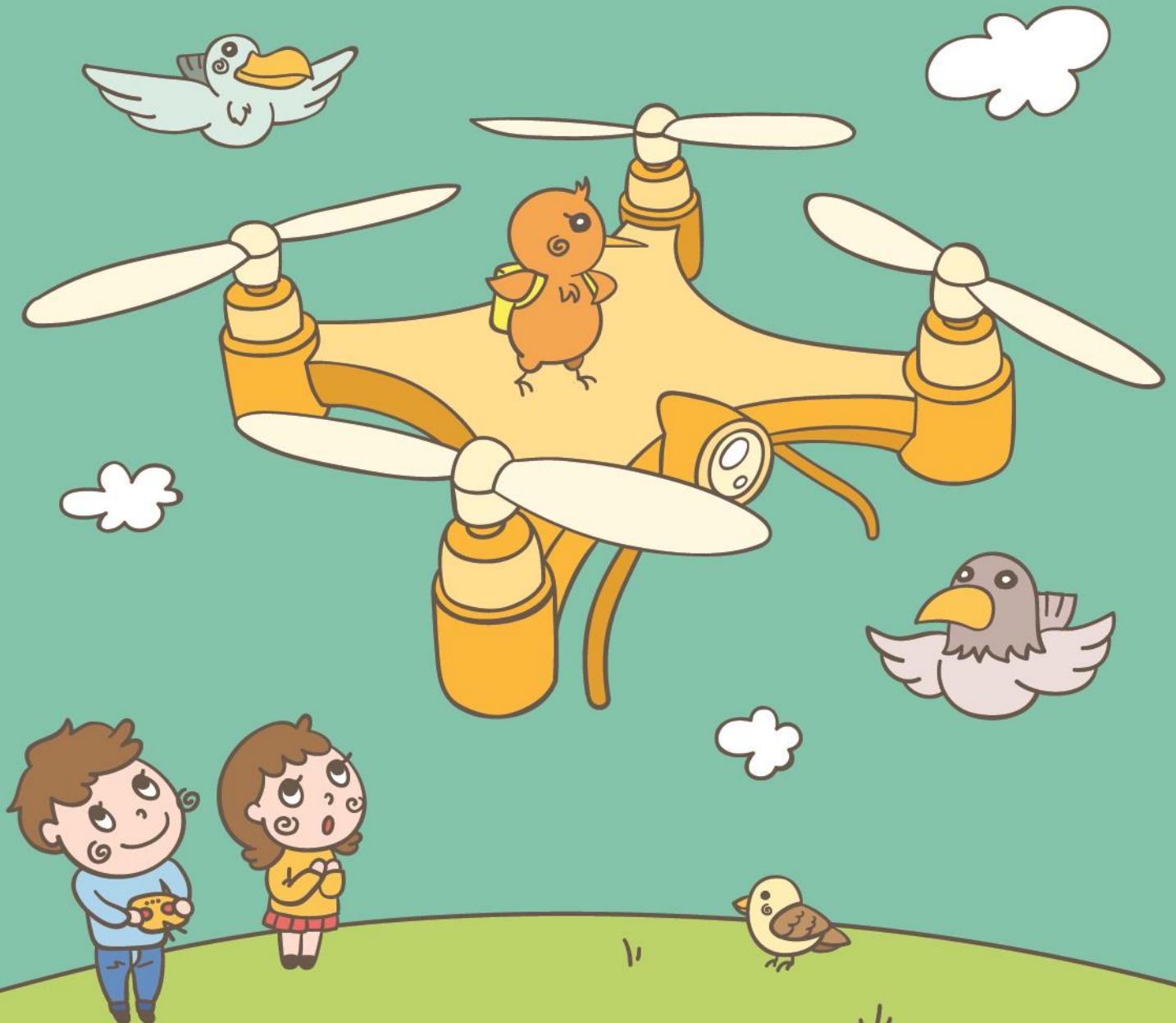




드론으로 배우는
프로그래밍 교실

캠프2h. 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.

4-1-3 상수에 대해서



상수는 변하지 않는 수입니다.

아두이노에는 기본적으로 존재하는 기본 상수가 있습니다.

이러한 기본 상수에 대해 잘 모르면, 아두이노를 프로그래밍 하는데 어려움을 겪을 수 밖에 없습니다.

아두이노의 기본 상수를 알아보며 상수를 익혀봅시다.

상수란

상수는 변하는 수인 변수와는 반대로 변하지 않는 수입니다. 아두이노에는 다음과 같이 미리 정해져 있는 상수들이 존재합니다.

상수는 프로그램을 읽기 쉽게 하기 위해 쓰입니다. 단순히 1 또는 0으로 표현하면, 그 의미를 알기 어렵기 때문입니다.

```
40 #define HIGH 0x1
41 #define LOW 0x0
```

<그림3-1> 미리 정의된 HIGH

이러한 상수들은 다음과 같이 아두이노가 설치되어 있는 폴더의 **Arduino.h** 파일 안에 정의되어 있습니다.

Arduino-1.8.4\hardware\Arduino\avr\cores\Arduino

참 / 거짓

참과 거짓은 논리값들을 정할 때의 Boolean값입니다. Boolean값은 true와 false로, true는 1, false는 0입니다. 거짓은 주로 0, 참은 주로 1로 나타냅니다. 하지만 보통 0이 아닌 수는 거의 참으로 간주됩니다. 따라서, Boolean에서는 -1, -2 그리고 -200 또한 참입니다.

HIGH /
LOW

Digital Pin 값들을 HIGH, LOW로 나누기 위함입니다. HIGH는 논리값1(ON 혹은 5V)로 LOW는 논리값0(OFF 혹은 0V)로 쓰입니다.

INPUT /
OUTPUT

상수값은 또한 pinMode함수에서 digitalWrite가 INPUT인지 OUTPUT인지 결정할 때 쓰입니다. INPUT은 0, OUTPUT은 1의 값을 가집니다.

#define과 const

#define const

상수를 만드는 방법에는 두 가지가 있습니다. 바로 `#define`과 `const`입니다.

#define

`define`의 의미는 '정의하다'입니다. 예를 들어 다음과 같이

```
1 | #define LED 3
```

`#define LED 3`이라는 명령어를 적게 되면 LED라는 글자는 숫자 3으로 쓰이게 됩니다.

이러한 `#define`은 단순히 글자가 바뀌는 것처럼 작용해서 메모리를 잡아먹지 않습니다.

const

`const`는 `constant`의 약자로 '일정한'이라는 의미와 '상수'라는 의미를 갖고 있습니다.

```
2 | const int BUTTON = 2;
```

`const int BUTTON = 2;`이라는 명령어를 적게 되면 앞으로 `BUTTON`이라는 변수는 읽기 전용이 되면서 상수로 사용되게 됩니다.

만약, `const`가 쓰인 변수에 다른 값을 집어넣으려고 하면 에러가 발생합니다.

#define vs const

보통 우리가 사용하는 컴퓨터와 달리 아두이노는 메모리를 많이 갖고있지 않습니다.

변수를 사용할 때는 컴퓨터와 마찬가지로 변수의 정보가 `ram`에 저장되게 됩니다.

요즘 컴퓨터는 보통 4G(약4,000,000,000)의 램을 사용하는 반면 아두이노는 2k(약2,000)의 램을 사용합니다.

따라서, 메모리를 아끼기 위해 `#define`을 사용하는게 좋겠죠?


상수 작성 해보기

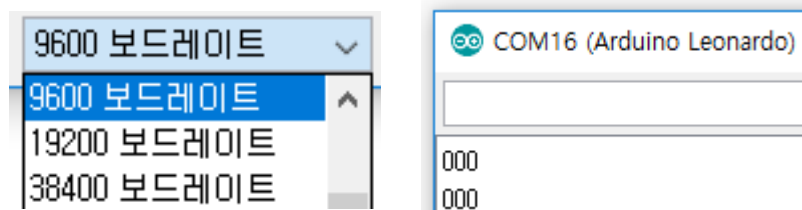
상수
작성

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

```
ch4_1_3_const
1 #define LED 3
2 const int BUTTON = 2;
3
4 void setup() {
5     Serial.begin(9600);
6     pinMode(LED, OUTPUT);
7     pinMode(BUTTON, INPUT);
8 }
9
10 void loop() {
11     if (digitalRead(BUTTON) == HIGH) {
12         Serial.print(true);
13         Serial.print(HIGH);
14         Serial.println(OUTPUT);
15         digitalWrite(LED, HIGH);
16     } else {
17         Serial.print(false);
18         Serial.print(LOW);
19         Serial.println(INPUT);
20         digitalWrite(LED, LOW);
21     }
22 }
```

<그림3-2> const 코드

- 2  버튼을 눌러 시리얼 모니터를 켭니다.
- 3 보드레이트를 맞춘 후 스위치를 눌러 시리얼모니터의 숫자가 변하는지, LED가 변하는지 확인합니다.



<그림3-3> 시리얼 통신 확인

시리얼 통신 해석

```
#define LED 3 // LED를 3으로 지정
const int BUTTON = 2; // BUTTON변수에 2를 저장 후 상수로

void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT); // 3번핀을 출력으로
  pinMode(BUTTON, INPUT); // 2번핀을 입력으로
}

void loop() {
  if (digitalRead(BUTTON) == HIGH) { // 만약 버튼이 눌리면
    Serial.print(true); // true값 출력
    Serial.print(HIGH); // HIGH값 출력
    Serial.println(OUTPUT); // OUTPUT값 출력 후 줄바꿈
    digitalWrite(LED, HIGH); // LED 켜
  } else {
    Serial.print(false); // false값 출력
    Serial.print(LOW); // LOW값 출력
    Serial.println(INPUT); // INPUT값 출력
    digitalWrite(LED, LOW); // LED 끄
  }
}
```

4-2-1 for문에 대해서



for문은 반복문 중 하나로 c언어에서 반복문을 표현할 때 주로 사용됩니다. 일정하게 숫자가 변할 때 각각의 숫자에 대해 다르게 적용시킬 명령이 있을 때 사용됩니다.

예를 들어 배열의 0번째부터 29번째까지 다르게 데이터를 넣어야 할 때 사용하면 편리하게 적용할 수 있습니다.

for문이란?

영어에서 **for**는 ~동안 이라는 뜻으로, 프로그래밍에서도 마찬가지로 정해진 기간동안 명령을 반복하는 반복문으로 사용됩니다.

변수를 선언한 후 그 변수가 정해진 범위 내에서 일정한 수만큼 증가함에 따라 명령을 다르게 할 때 주로 사용됩니다.

예를 들어 다음과 같이 변수 **i**가 0에서 3까지 1씩 증가하면서 명령어를 실행하면 0, 1, 2가 출력됩니다.



<그림1-1> for문 블록 코딩

for문은 아래 그림처럼 **초기값**, **종결값**, **변화조건**으로 구성됩니다. **i=0**처럼 반복의 시작이 어디서부터인지 정하는게 초기값입니다. 종결값은 **i<3** 인 부분입니다. 변화조건은 **i++** 인 부분으로 값을 변화시켜 반복을 제어하게 됩니다.

```
for( i=0 ; i<3 ; i++ )
{
} i=0으로 i가 3보다 i를 1씩
  초기화 작을때까지 증가
```

우선 **i**에 0을 대입하고 명령문을 실행합니다.
명령을 실행한 다음 변화조건을 적용합니다.
이후, 종결값과 **i**를 비교한 후 참이면 다시 명령을 실행하게 됩니다.

fading 예제 작성하기

fading
예제

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.


ch4_2_1_fading

```

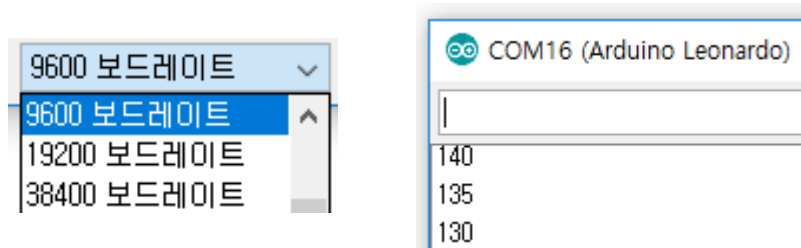
1 void setup() {
2   Serial.begin(9600);
3   pinMode(3, OUTPUT);
4 }
5
6 void loop() {
7   for (int i = 0; i <= 255 ; i += 5) {
8     Serial.println(i);
9     analogWrite(3, i);
10    delay(30);
11  }
12  for (int i = 255; i >= 0 ; i -= 5) {
13    Serial.println(i);
14    analogWrite(3, i);
15    delay(30);
16  }
17 }

```

<그림1-2> fading 예제

- 2  버튼을 눌러 시리얼 모니터를 켭니다.

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



<그림1-3> 값 확인

fading 예제 해석

```
void setup() {  
  Serial.begin(9600);  
  pinMode(3, OUTPUT);  
}  
  
void loop() {  
  for (int i = 0; i <= 255; i += 5) {  
    Serial.println(i); //i값 출력  
    analogWrite(3, i); // i값만큼 LED밝기 변화  
    delay(30);  
  }  
  for (int i = 255; i >= 0; i -= 5) {  
    Serial.println(i); //i값 출력  
    analogWrite(3, i); //i값만큼 LED밝기 변화  
    delay(30);  
  }  
}
```

4-2-2 while문에 대해서



while문은 조건이 맞으면 계속적으로 실행되는 반복문입니다.

보통 조건에 true를 집어넣고 계속적으로 반복시키며 사용합니다.

while문의 제어는 break나 continue 등을 통해 이루어집니다.

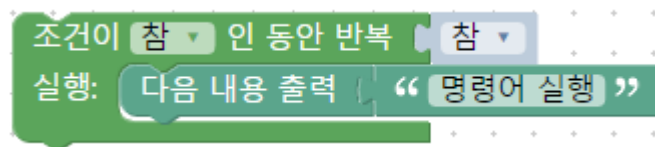
while문은 굉장히 간단한 구조로 광범위하게 사용되기에 꼭 사용법을 알아두어야 합니다.

**while문
이란?**

while문은 for문과 같은 반복문 중 하나로 간단한 구조를 가지고 있습니다.

for문과는 달리 조건에 대한 검사만 하며 조건이 참이면 명령을 반복합니다.

예를 들어 다음과 같이 괄호 안의 조건이 참인 경우만 계속 반복적으로 실행합니다.



<그림2-1> while문 블록 코딩

while문은 다음과 같은 구조를 가지고 있습니다.

```
while( i<3 )
{
    i가 3보다
    작으면 반복
}
```

while문은 조건을 빠져나갈 수 있는 **break;**와 조건 검사로 돌아가는 **continue;**를 통해 제어할 수 있습니다.

break

break를 쓰면 while문의 반복을 멈추고 중괄호를 빠져나갈 수 있습니다.

예를 들어 다음과 같은 코드에서는 **i++**이 실행되지 않습니다.

```
while( true )
{
    break;
    i++;
}
```

continue

continue를 쓰면 해당 지점에서 바로 다시 while문의 조건을 검사한 후 위에서부터 차례대로 명령을 실행합니다.

아래와 같은 코드에서는 i가 2와 같으면 continue에 의해 다시 조건 검사로 이동하기 때문에 i++이 적용되지 않아 반복문이 계속 실행됩니다.

```
int i = 0 ;
while( i < 3 )
{
    if( i == 2 )
    {
        continue;
    }
    i++;
}
```

do while문
이란?

do while문은 while문과 비슷하지만 약간 다릅니다.

while문에서는 조건을 먼저 검사한 후 참이면 명령이 실행되지만 do while문에서는 먼저 명령을 실행한 뒤 조건을 검사하여 반복할지 말지를 결정합니다.

```
do
{
    x++;
    Serial.print(x);
}
while ( x < 3 ) ;
```

while문과 스위치 카운트


while문과
스위치
카운트

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

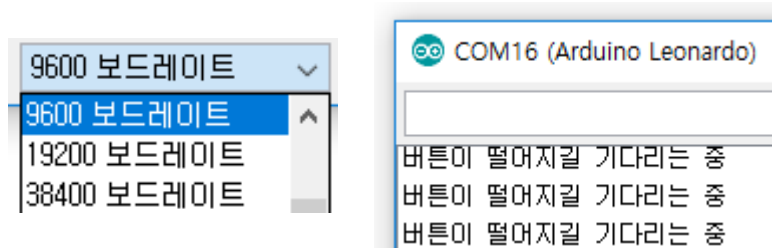
ch4_2_2_while_sw

```
1 int cnt = 0;
2
3 void setup() {
4     Serial.begin(9600);
5     pinMode(2, INPUT);
6 }
7
8 void loop() {
9     if (digitalRead(2) == HIGH) {
10         cnt++;
11         if (cnt > 3) cnt = 0;
12
13         while (digitalRead(2) == HIGH) {
14             Serial.println("버튼이 떨어지길 기다리는 중");
15         }
16
17         Serial.print("버튼 눌림 횟수 : ");
18         Serial.println(cnt);
19     }
20 }
```

<그림2-2> while문과 스위치 카운트

- 2  버튼을 눌러 시리얼 모니터를 켭니다.

- 3 보드레이트를 맞춘 후 스위치를 누르고 시리얼 모니터에서 값이 변하는지 확인합니다.



<그림2-3> 값 확인

while문과 스위치 카운트 해석

```
int cnt = 0; //스위치 눌림 횟수 카운트 변수
```

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

```
void loop() {  
  if (digitalRead(2) == HIGH) { //만약 스위치가 눌렀다면  
    cnt++; // cnt값 증가  
    if (cnt > 3) cnt = 0; //만약 cnt가 4이상이면 0으로  
    while (digitalRead(2) == HIGH) { //스위치가 눌린 상태면 반복  
      Serial.println("버튼이 떨어지길 기다리는 중"); //문구 출력  
    }  
    Serial.print("버튼 눌림 횟수 : ");  
    Serial.println(cnt); //스위치 눌림 횟수 출력  
  }  
}
```

아두이노는 빠른 속도로 동작하기 때문에, 스위치를 한 번만 누르려고 해도 순식간에 여러 번 눌리게 됩니다. **while**문을 사용하면, 스위치가 눌러 있는 동안 다음 명령어로 넘어가지 않기 때문에 스위치를 한 번만 누를 수 있게 됩니다.

4-2-3 배열



배열이란 변수들을 손쉽게 쓸 수 있는 도구입니다.

대량의 데이터를 다룰 때 배열이 없다면 변수를 일일이 선언해야 할 것입니다. 변수를 일일이 선언한다는 것은 프로그래밍의 취지인 단순 반복 작업의 해결과는 거리가 먼 행동입니다.

배열을 잘 활용하는 것은 초급 프로그래머를 탈출하기 위한 첫 관문입니다.

배열이란?

배열
정의

배열이란 간단히 말해 변수들을 한 묶음으로 묶어 놓은 것입니다. 만약 30명이 있는 1반 학생들의 영어 점수를 변수에 저장한다고 하면 다음과 같이 변수가 30개 필요하게 됩니다.

```
int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, ...
```

일일이 변수를 선언하는 작업은 매우 귀찮고 반복적인 작업을 필요로 합니다. 이 때 배열을 사용하면 간단히 문제를 해결할 수 있습니다.

```
int    a[30]    ;
```

int의 자료형을 사용하며 a라는 이름의 배열을 지정

변수의 선언부터 간단 해졌습니다. 배열을 만든 후 배열의 각 변수에 접근을 할 때는 다음과 같은 방법을 사용하면 됩니다.

```
a[0]    =    100    ;
```

a의 첫 번째 변수에 100이라는 값을 저장

배열에 들어있는 변수들은 0번부터 시작하여 29번까지 존재하게 됩니다. 배열에 들어있는 값은 다음과 같이 사용하면 됩니다.

```
int    b    =    a[0]    ;
```

int의 자료형을 사용하여 a[0]에 있는 값을 저장
b라는 이름을 지정하여

int형 배열

정수형 배열은 다음과 같이 선언하고 사용할 수 있습니다.

- `int prime[5] = {2, 3, 5, 7, 11};`
- `int three = prime[1];`

three라는 변수에는 prime이라는 배열의 2번째 값인 숫자3이 들어가게 됩니다.

double형 배열

실수형 배열은 다음과 같이 선언하고 사용할 수 있습니다.

- `double root[4] = {1.414, 1.732, 2.000, 2.236};`
- `double r5 = root[3];`

r5라는 변수에는 root라는 배열의 4번째 값인 2.236이 들어가게 됩니다.

char형 배열

문자형 배열은 다음과 같이 선언하고 사용할 수 있습니다.

- `char ch[5] = {'h', 'e', 'l', 'l', 'o'};`
- `char h = ch[0];`

h라는 변수에는 ch라는 배열의 1번째 값인 h가 들어가게 됩니다.

위 방식 말고도 아스키 코드값을 바로 넣을 수도 있습니다.

- `char asc[4] = {62, 95, 60, 98};`
- `char b = asc[3];`

위 코드를 넣고 실행시키면 숫자가 아니라 아스키코드표를 통한 문자가 나오게 됩니다.

또한, “”(쌍따옴표)를 통해서도 문자형 배열을 표현할 수 있습니다.

- `char str[6] = "world";`
- `char d = str[4];`

단, “”를 사용하여 문자형 배열을 만들 경우엔 배열의 길이가 문자의 길이보다 1만큼 더 길게 됩니다.

2차원 배열

2차원 배열을 사용하면 행렬을 만들 수 있습니다.

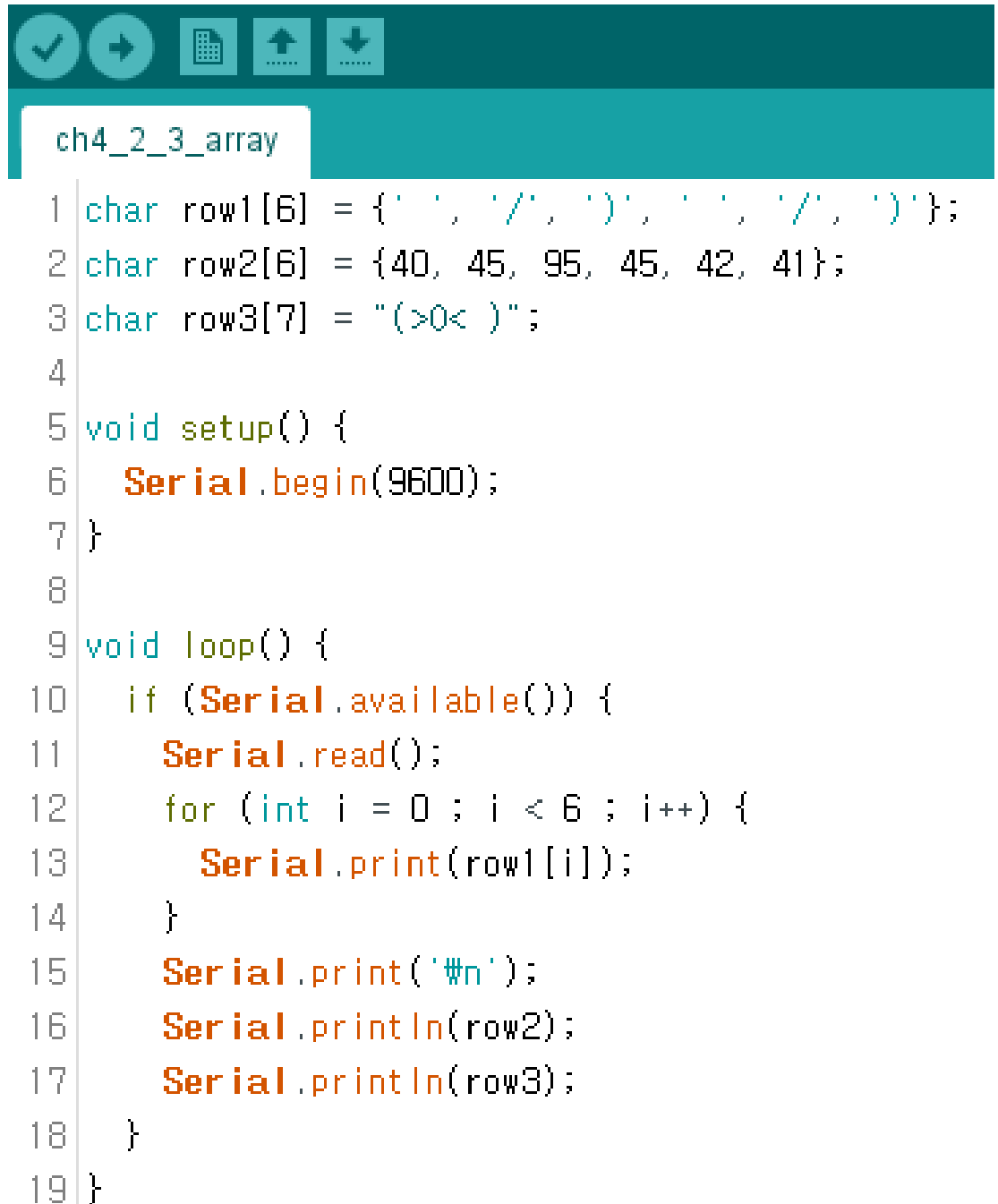
- `int rc[2][3] = {{1, 2, 3}, {4, 5, 6}};`

1	2	3
4	5	6

배열 작성 해보기

배열 작성
해보기

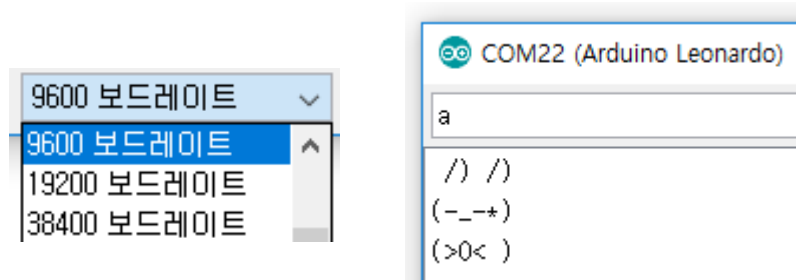
- ① 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.



<그림3-1> 배열 작성 코드

- ②  버튼을 눌러 시리얼 모니터를 켭니다.

- 3 보드레이트를 맞춘 후 시리얼 모니터에서 아무 값을 입력한 후 아두이노로부터 받은 값을 확인합니다.



<그림3-2> 값 확인

배열 작성 해석

```
char row1[6] = {';', '/', ')', ' ', ' ', ' '}; // 첫 번째 배열  
char row2[6] = {40, 45, 95, 45, 42, 41}; // 두 번째 배열  
char row3[7] = ">O<"; // 세 번째 배열로 문자열 작성
```

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

```
void loop() {  
  if (Serial.available()) {  
    Serial.read();  
    for (int i = 0; i < 6; i++) { // 배열의 길이만큼 반복  
      Serial.print(row1[i]); // 첫 번째 배열의 값들 출력  
    }  
    Serial.print("\n"); // 줄 바꿈  
    Serial.println(row2); // 두 번째 배열 출력  
    Serial.println(row3); // 세 번째 배열 출력  
  }  
}
```

4-3-1 모터에 대해서



드론에 들어가는 모터는 대략 두 종류가 있습니다.

DC모터는 보통 손바닥만한 드론이나 그 보다 작은 초소형 드론에 많이 사용됩니다.

BLDC모터는 그보다 큰 중형에서 대형 드론에 사용되며 DC모터에 비해 비싸지만 그만큼 내구성이 좋다는 특징이 있습니다.

짐벌이란?

드론 짐벌

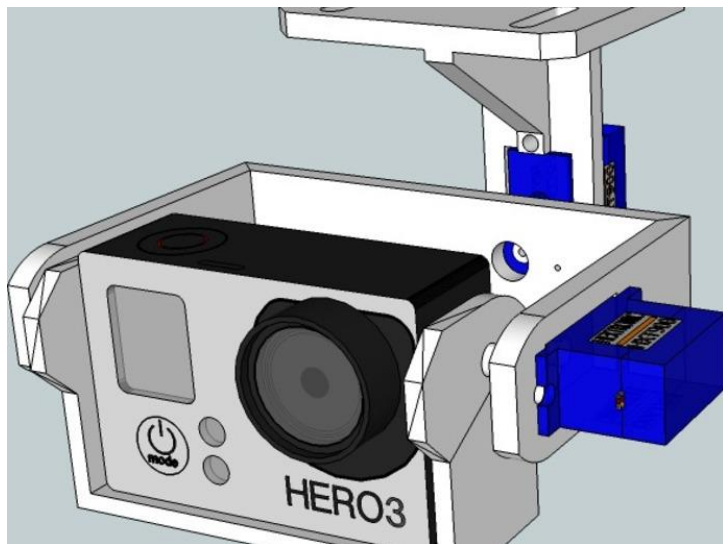
짐벌은 아래 그림과 같이 드론이 기울어져도 카메라는 기울어지지 않게 잡아주는 역할을 합니다.



<그림1-3> 짐벌 관련 영상

출처 : <https://www.youtube.com/watch?v=ajJ-c7ooOek>

서보모터는 드론의 짐벌을 만들 때 카메라의 수평을 유지하는 목적으로 사용될 수 있습니다.



<그림1-4> 짐벌 관련 이미지

이러한 서보모터의 제어는 PWM을 통해서 이루어집니다.

서보모터

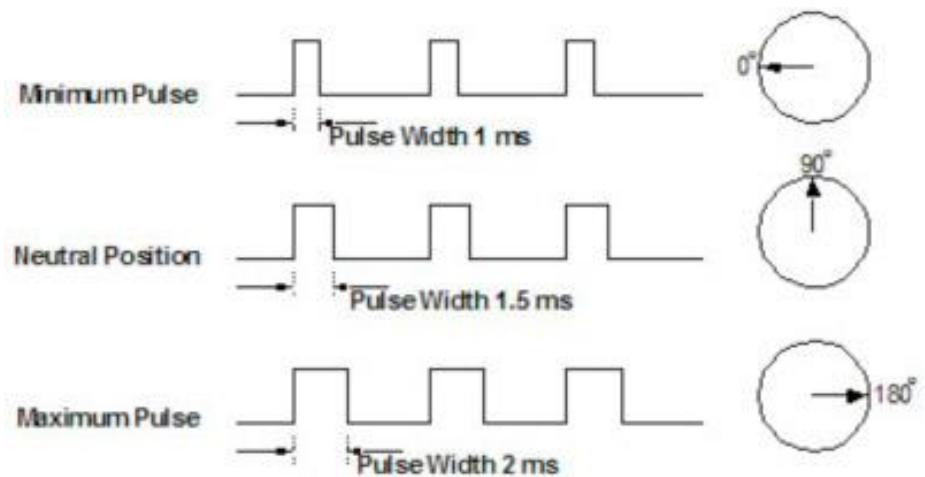
서보모터는 아래 그림처럼 상단부분의 모터에 의해 회전하는 부분과 하단 부분의 연결선 3개로 구성되어 있습니다.



<그림1-5> 서보모터

서보모터
제어

서보 모터는 펄스에 의해 제어 되는데, 펄스의 영어 의미는 맥박이고, 물리학적 의미는 매우 짧은 시간 동안만 흐르는 전류입니다.



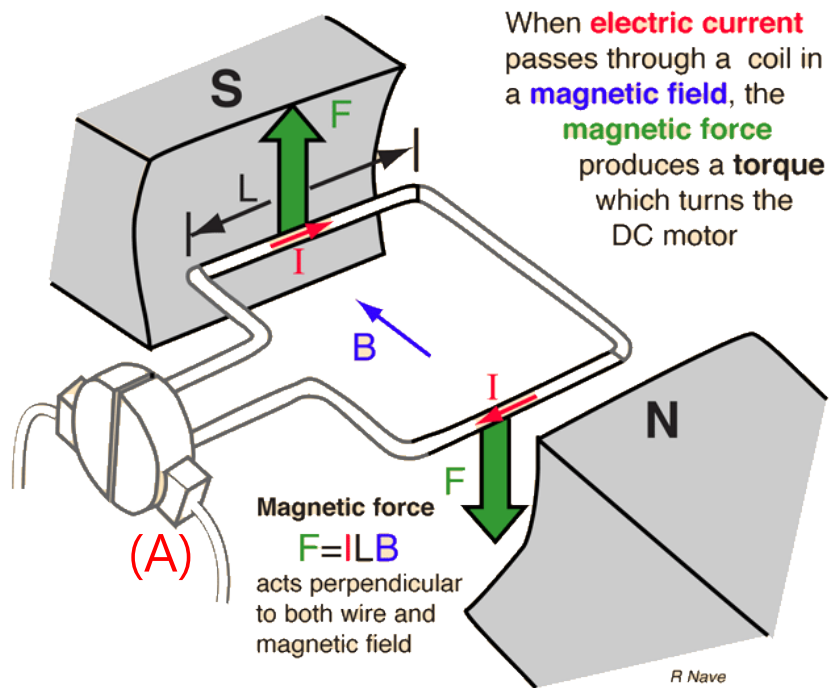
<그림1-6> 서보모터 제어 방식

서보 모터의 3개 핀 중 가운데 핀을 통해 펄스를 조정하여 서보모터를 제어하게 됩니다. 전체 20ms 중 몇 ms동안 높은 전압을 유지하냐에 따라서 서보모터의 각도가 달라지게 됩니다. 이 때 PWM이 사용되어 펄스의 폭을 얼마로 정할건지 결정하게 됩니다.

DC모터 내부 구성

위 그림과 같이 자석(자기장) 사이의 전선에 전류가 흐르게 되면 그 전선은 힘을 위쪽 방향으로 힘을 받게 됩니다.

이러한 플레밍의 왼손 법칙을 적용하여 아래 그림과 같이 모터가 구성됩니다.



<그림1-9> DC모터 내부 구성

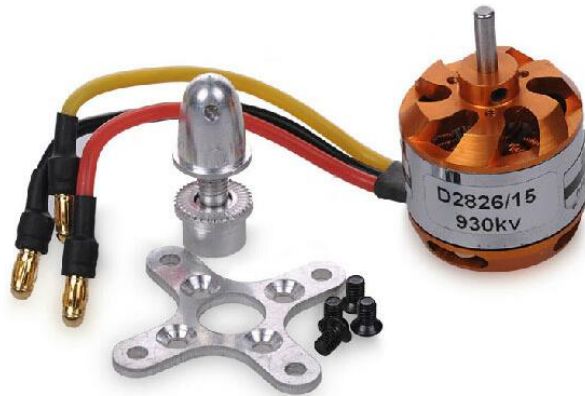
모터 내부에서는 왼쪽 전선은 위로, 오른쪽 전선은 밑으로 힘을 받기 때문에 모터가 회전할 수 있게 됩니다.

이 때, 받는 힘은 전류의 세기, 전선의 길이, 자기장의 세기에 비례하게 됩니다.

(A) 부분을 브러쉬라고 하는데, DC모터는 이 브러쉬 부분이 계속 닿는다는 단점을 가지고 있습니다. 모터가 계속 회전을 하게 되는데, 마찰이 일어나서 브러쉬가 닳게 됩니다.

BLDC모터

BLDC모터(Brushless DC Motor)는 DC모터의 단점인 브러쉬가 없이 동작하게 만든 모터입니다.



<그림1-10> BLDC 모터

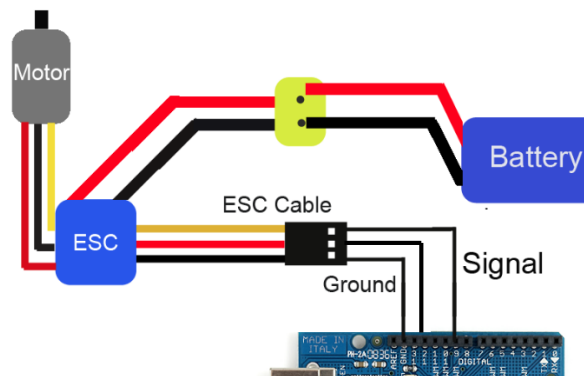
기존의 DC모터가 내부의 축(코일)을 회전시켰던 것과는 달리 BLDC모터는 바깥의 통(캔)을 회전시킵니다. 그렇기에 통돌이 모터라고도 불립니다.

이러한 BLDC모터는 브러쉬로 인한 마찰과 소음이 없는 장점으로 인해 전기차에서도 사용됩니다.

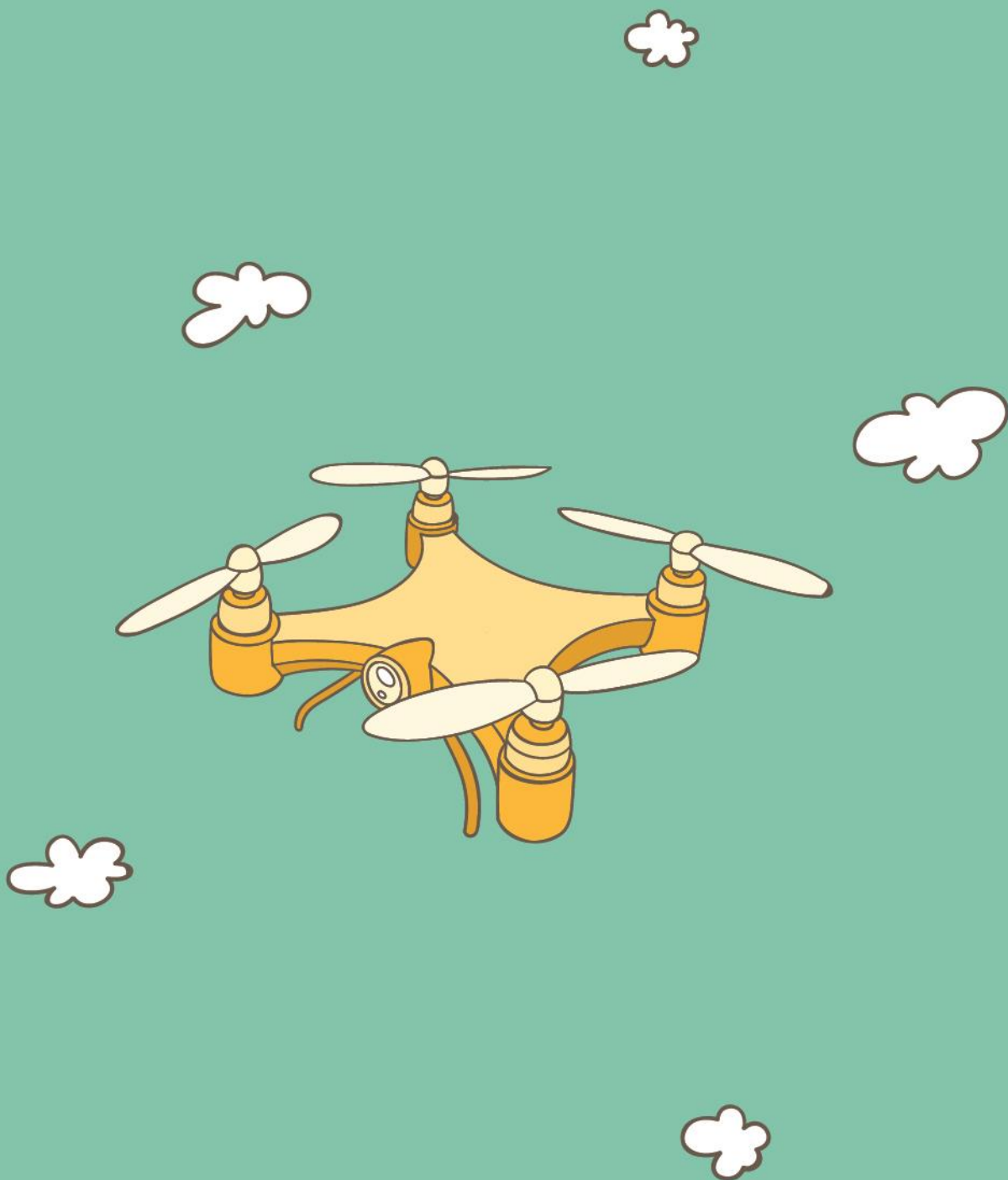
ESC

ESC(Electric Speed Controller)는 변속기로 모터의 속도를 제어할 때 사용됩니다.

다음과 같이 8개의 선 중 3개는 모터, 2개는 배터리, 3개는 신호선과 연결합니다.



<그림1-11> ESC



WHIT