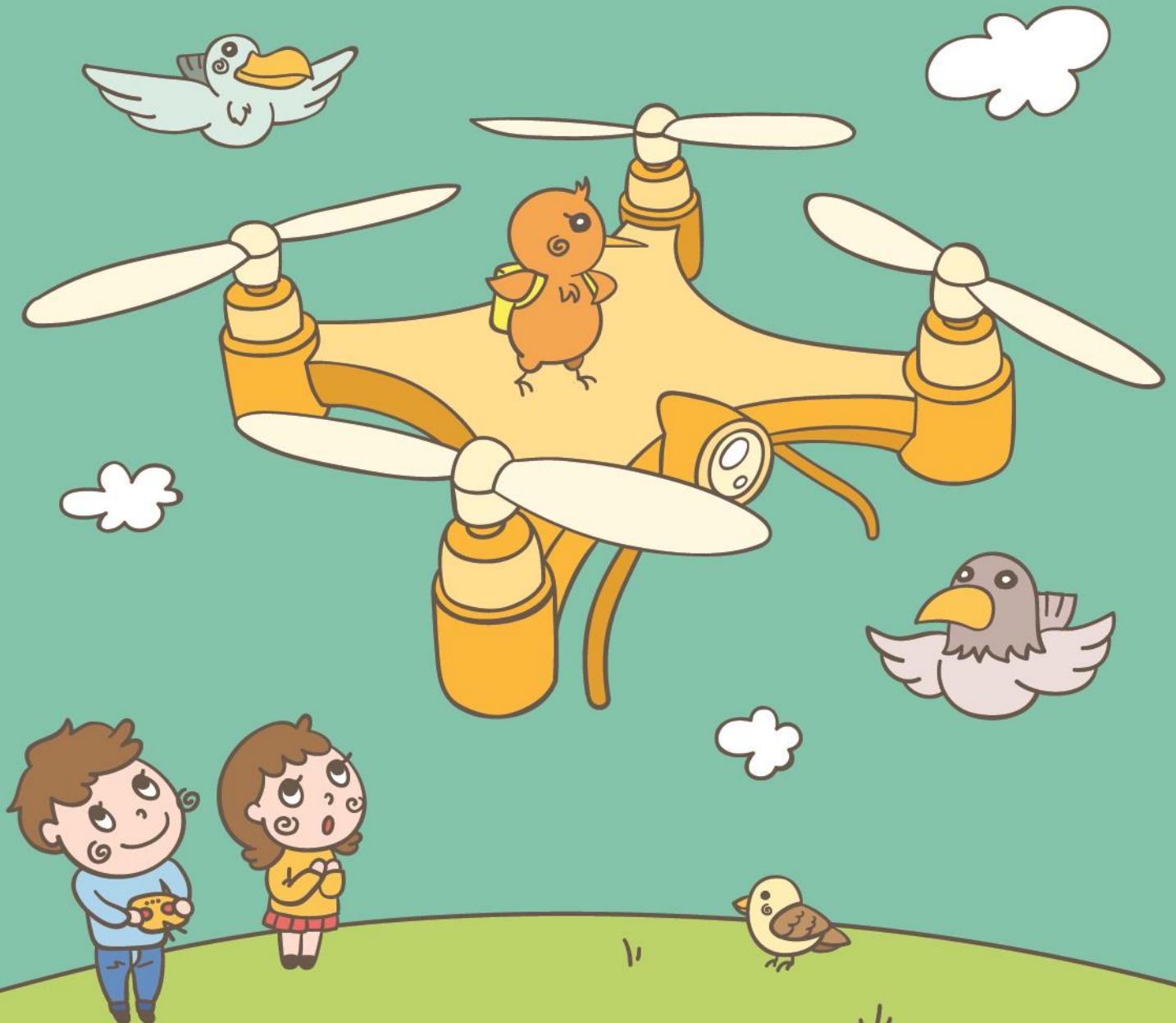




드론으로 배우는
프로그래밍 교실

캠프3h. 모터와 자이로



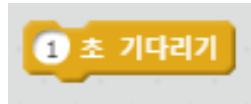
01 시간 함수



아두이노에는 시간에 대해 기본적으로 제공되는 함수들이 있습니다. 예를 들어 아두이노의 동작을 몇 초간 멈춘다거나, 실행된 시간을 알 수 있습니다. 이러한 기능은 아두이노를 동작시키는데 있어 꼭 필요한 기능들입니다. 만약, 시간과 관련된 기능이 없다면 아두이노는 장난감에 불과할 것입니다.

delay()

delay는 지연, 지체, 연기하다는 뜻을 가지고 있습니다.



```
delay(1000);
```

<그림1-1> 일정 시간만큼 정지

delay() 함수는 동작하고 있는 프로그램을 원하는 시간동안 멈추고자 할 때 사용합니다.
입력 단위는 Millisecond 단위로, 1 / 1000 초 입니다. 즉, 1초를 멈추고 싶다면 delay(1000); 을 해주어야 합니다.

millis()

millis() 함수는 아두이노가 실행된 이후 경과된 시간을 알고자 할 때 사용합니다.
대략 50일까지 셀 수 있습니다.



```
value = millis();
```

<그림1-2> 경과된 시간을 파악

millis()함수는 시간을 milliseconds 단위로 반환합니다.
위 명령에서 value에 들어있는 1000은 1초를 의미합니다.

**delay의
문제**

delay() 함수를 사용하게되면 아두이노의 모든 동작이 일시 정지됩니다. 이 때문에 delay()함수와 버튼 입력을 같이 사용하게 될 때에 문제가 생기게 됩니다.

버튼을 입력받아 어떤 동작을 하려하는데, 만약 아두이노가 delay()함수를 실행하고 있는 중이면 버튼이 입력되었는지 판별하는게 불가능해집니다.

이럴 때 millis()를 사용하면 문제를 해결할 수 있습니다.

```
if ( millis() - old_time > 2000 ) {  
    //2초마다 실행할 명령어 넣기  
    old_time = millis();  
}
```

<그림1-3> millis()로 시간 파악

위와 같은 코드를 작성하면 특정 명령어를 2초마다 실행할 수 있게 됩니다.

millis() 실습

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

ch4_4_1_millis

```

1 unsigned long past = 0;
2 const int MY_DELAY_TIME = 1000;
3
4 void setup() {
5     Serial.begin(9600);
6 }
7
8 void loop() {
9     unsigned long now = millis();
10    if (now - past >= MY_DELAY_TIME) {
11        past = now;
12        Serial.print("Time is gone ");
13        Serial.println(now/1000);
14    }
15 }
```

<그림1-4> millis() 실습 코드

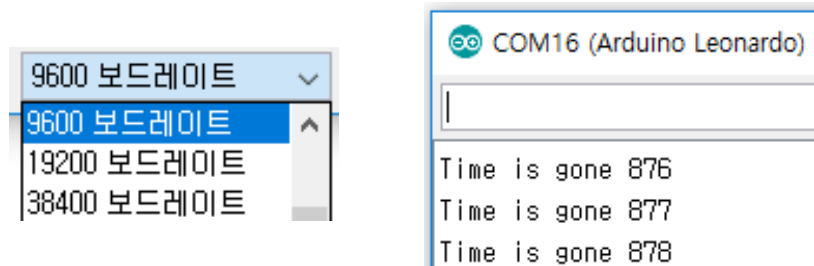
- 2  버튼을 눌러 시리얼 모니터를 켭니다.

꿀TIP

왜 int를 안 쓰지?

unsigned long 즉, 부호가 없는 long 자료형을 사용하게 되면 int를 사용할 때 보다 더 많은 숫자를 담을 수 있습니다.

- 3 보드레이트를 맞춘 후 시리얼 모니터에서 값이 변하는지 확인합니다.



<그림1-5> 값 확인

millis() 실습 해석

```
unsigned long past = 0; // 부호 없는 long타입으로 변수 선언
const int MY_DELAY_TIME = 1000; // 반복 시간 정하기
```

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

```
void loop() {
  unsigned long now = millis(); // 현재 시간 받아옴
  if (now - past >= MY_DELAY_TIME) { // 만약 시간이 지나면
    past = now; // past 재설정
    Serial.print("Time is gone "); // 문구 출력
    Serial.println(now/1000); // 초 출력
  }
}
```

*10번째 줄에서 now-past는 시간이 얼마나 지났는지를 체크하는데 사용됩니다. 만약, now-past가 1000 이상이라면 if문 내부 명령어가 실행됩니다.

* 11번째 줄의 past = now;는 past에 다시 현재 시간을 넣어주어서 1초 후에 실행될 수 있게 해줍니다. 만약 11번째 줄이 없다면 1초 후 계속해서 if문이 실행 될 것입니다.

millis() 실습 함수화하기

millis()
실습
함수화하기

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

ch4_4_1_millis_func

```

1 unsigned long past = 0;
2
3 void setup() {
4     Serial.begin(9600);
5 }
6
7 void loop() {
8     if (myTimer(1000)) {
9         Serial.println("1초마다 실행됩니다.");
10    }
11 }
12
13 boolean myTimer(int waitTime) {
14     unsigned long now = millis();
15     if (now - past >= waitTime) {
16         past = now;
17         Serial.println(now / 1000);
18         return true;
19     }
20     return false;
21 }

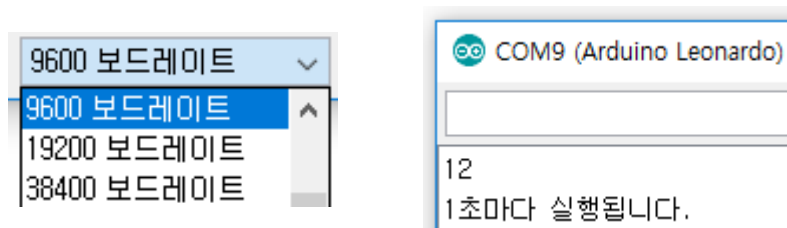
```

<그림1-6> millis() 실습 함수화 코드

- 2  버튼을 눌러 시리얼 모니터를 켭니다.

millis()
실습 함수화
해석

- 3 보드레이트를 맞춘 후 시리얼 모니터에서 값이 변하는지 확인합니다.



<그림1-7> 값 확인

```
unsigned long past = 0; // 시간 저장 변수
```

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

```
void loop() {  
  if (myTimer(1000)) { // 함수 입력을 1000으로 함수 실행  
    Serial.println("1초마다 실행됩니다.");  
  }  
}
```

```
boolean myTimer(int waitTime) { // 함수 정의  
  unsigned long now = millis(); // 현재 시간 저장  
  if (now - past >= waitTime) { // 시간 경과가 1초 이상임녀  
    past = now; // 현재 시간 저장  
    Serial.println(now / 1000); // 경과된 시간 표시  
    return true; // 참 반환  
  }  
  return false; // 거짓 반환  
}
```

*myTimer()라는 함수를 만들어서 loop문을 간단하게 만들어주었습니다. loop()에 많은 코드가 들어가게 될 경우 가독성이 떨어지므로, 특정 기능을 하는 함수를 따로 만들어 코드를 간결하게 만듭니다.

02 선풍기 만들어보기



집에는 꼭 한대씩 있는 선풍기! 선풍기는 어떻게 동작하는 걸까요?

선풍기는 220V를 통해 전력을 공급받고 버튼으로 제어되며 모터를 통해 바람을 만들어 냅니다.

아두이노를 통해서도 비슷한 선풍기를 만들 수 있습니다.

직접 선풍기를 만들어 보면서 선풍기의 원리에 대해 익혀봅시다.

선풍기는 어떻게 구성될까

선풍기 구조

집에 있는 선풍기를 떠올려 봅시다.



<그림2-1> 선풍기

우선 날개가 있고 날개를 돌릴 모터가 있습니다. 그리고 ON/OFF 버튼과 강, 중, 약의 세기를 조절하는 버튼, 회전과 예약 시간을 설정하는 버튼 등이 있습니다.

우리는 다음과 같이 선풍기를 만들어 봅시다.

- ON/OFF버튼과 표시 : 스위치와 LED
- 모터 속도 조절 : 가변저항
- 예약 시간 설정 : 시리얼 모니터



<그림2-2> 선풍기 버튼

조금 어려울 것도 같지만 차근차근 따라하다 보면 쉽게 선풍기를 만들 수 있을 겁니다.

전원 버튼 만들기

전원 버튼
구성

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

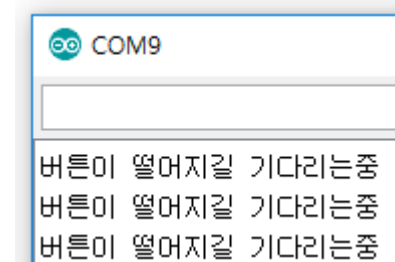
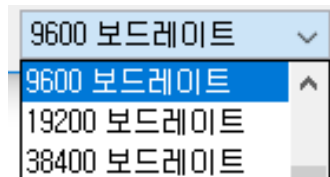
```
ch4_4_2_fan_ONOFF
1 boolean SW = false;
2 unsigned long past = 0;
3 unsigned long waitTime = 0;
4
5 void setup() {
6     Serial.begin(9600);
7     pinMode(2, INPUT);
8     pinMode(3, OUTPUT);
9     pinMode(5, OUTPUT);
10 }
11
12 void loop() {
13     changeSW();
14     if (SW) {
15         digitalWrite(3, HIGH);
16     } else {
17         digitalWrite(3, LOW);
18     }
19 }
20
21 void changeSW() {
22     if (digitalRead(2) == HIGH) {
23         SW = !SW;
24         while (digitalRead(2) == HIGH) {
25             Serial.println("버튼이 떨어지길 기다리는중");
26         }
27     }
28 }
```

<그림2-3> 전원 버튼 구성

- 2  버튼을 눌러 시리얼 모니터를 켭니다.

전원 버튼 구성 해석

- 3 보드레이트를 맞춘 후 스위치를 누름에 따라 시리얼 모니터에서 값이 변하는지, LED의 밝기변화가 있는지 확인합니다.



<그림2-4> 값 확인

```
boolean SW = false; // 전원ON/OFF 저장
unsigned long past = 0; // 추후 설명
unsigned long waitTime = 0; // 추후 설명
```

```
void setup() {
  Serial.begin(9600); // 시리얼 통신 시작
  pinMode(2, INPUT); // 스위치 입력 받는 2번 핀 설정
  pinMode(3, OUTPUT); // LED 출력 하는 3번 핀 설정
  pinMode(5, OUTPUT); // 모터 출력 하는 5번 핀 설정
}
```

```
void loop() {
  changeSW(); // 전원ON/OFF 함수 실행
  if (SW) { // 만약 전원이 켜진다면
    digitalWrite(3, HIGH); // LED 켜
  } else { // 전원이 꺼진다면
    digitalWrite(3, LOW); // LED 끄
  }
}
```

```
void changeSW() { // 전원 ON/OFF 함수 정의
  if (digitalRead(2) == HIGH) { // 버튼이 눌린다면
    SW = !SW; // 켜져있다면 끄고, 꺼져있다면 켜는 변수 저장
    while (digitalRead(2) == HIGH) { // 버튼이 떨어지길 기다림
      Serial.println("버튼이 떨어지길 기다리는중");
    }
  }
}
```

예약 시간 설정하기

예약 시간
설정하기
구성

1 아래 코드를 30번째줄부터 작성합니다.

```

30 boolean myTimer() {
31     unsigned long now = millis();
32
33     if (Serial.available()) {
34         char c = Serial.read();
35         past = now;
36         waitTime = c-'0';
37     }
38
39     int myTime = (now-past) / 1000;
40     Serial.println(myTime);
41
42     if (myTime >= waitTime && waitTime != 0) {
43         past = now;
44         waitTime = 0;
45         return true;
46     }
47     return false;
48 }

```

<그림2-5> 예약 시간 설정 함수 구성

2 좌측 코드를 우측과 같이 바꾼 후 업로드 합니다.

```

12 void loop() {
13     changeSW();
14     if (SW) {
15         digitalWrite(3, HIGH);
16     } else {
17         digitalWrite(3, LOW);
18     }
19 }

```



```

12 void loop() {
13     changeSW();
14     if (SW) {
15         if (myTimer()) {
16             SW = false;
17         } else {
18             digitalWrite(3, HIGH);
19         }
20     } else {
21         digitalWrite(3, LOW);
22     }
23 }

```

<그림2-6> 예약 시간 설정 loop 부분 변경

꿀TIP

줄 번호 표시

파일-환경설정의
줄 번호 표시에 체크하면
줄 번호가 보입니다.

☒ 줄 번호 표시

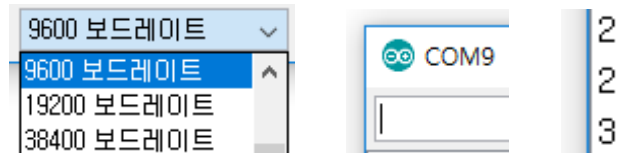
전원 버튼 구성 해석

꿀TIP

'0'을 빼는이유

Serial.read()로 들어온
값은 아스키 코드로
저장됩니다. 따라서 '0'을
빼주어야 원하는 숫자값을
얻을 수 있습니다.

- 3 보드레이트를 맞추고 전원을 켜면 숫자가 올라가는 것을 확인할 수 있습니다. 이후, 시리얼 모니터 창에 1~9 사이의 값을 입력하고 전송을 누르면 해당 시간 이후 전원이 꺼지는 것을 확인할 수 있습니다.



<그림2-7> 값 확인

```
void loop() {
  changeSW(); // 전원ON/OFF 함수 실행
  if (SW) { // 만약 전원이 켜진다면
    if (myTimer()) { // 만약 예약 시간이 다 됐다면
      SW = false; // 전원 끄
    } else { // 시간이 있다면
      digitalWrite(3, HIGH); // LED 켜
    }
  } else { // 전원이 꺼진다면
    digitalWrite(3, LOW); // LED 끄
  }
}

boolean myTimer() { // 예약 시간 설정 함수 정의
  unsigned long now = millis(); // 현재 시간 받아와 저장

  if (Serial.available()) { // 시리얼 모니터로 들어온 값이 있다면
    char c = Serial.read(); // 들어온 값을 char형으로 저장
    past = now; // 시간 저장
    waitTime = c - '0'; // 들어온 값을 숫자로 변형
  }

  int myTime = (now - past) / 1000; // 시간을 초 단위로 변환
  Serial.println(myTime); // 현재 시간 및 남은 시간 표시

  if (myTime >= waitTime && waitTime != 0) { // 예약 시간이 되면
    past = now; // 시간 저장
    waitTime = 0; // 시간 초기화
    return true; // 타이머 완료 실행
  }
  return false; // 시간 아직 안 됨
}
```

모터 세기 조절하기

모터 세기 조절하기 구성

- 1 좌측 코드를 우측과 같이 바꾼 후 업로드 합니다.

```

12 void loop() {
13   changeSW();
14   if (SW) {
15     if (myTimer()) {
16       SW = false;
17     } else {
18       digitalWrite(3, HIGH);
19     }
20   } else {
21     digitalWrite(3, LOW);
22   }
23 }

```

→

```

12 void loop() {
13   changeSW();
14   if (SW) {
15     if (myTimer()) {
16       SW = false;
17     } else {
18       digitalWrite(3, HIGH);
19       analogWrite(5, analogRead(A0) / 10);
20     }
21   } else {
22     digitalWrite(3, LOW);
23     analogWrite(5, 0);
24   }
25 }

```

<그림2-8> 모터 속도 제어 코드 추가

- 2 배터리와 모터를 연결하고 베이스보드의 스위치를 켭니다.(16페이지 참조)
- 3 버튼을 눌러 전원을 켜 후 모터가 도는지 확인합니다. 이후, 가변저항을 돌려 모터의 세기가 변하는지 확인합니다.
- 4 보드레이트를 맞추고 시리얼 모니터 창에 1~9 사이의 값을 입력하고 전송을 누르면 해당 시간 이후 전원이 꺼지는 것을 확인할 수 있습니다.



<그림2-9> 값 확인

모터 세기 조절하기 해석

```

digitalWrite(3, HIGH); // LED 켜
analogWrite(5, analogRead(A0) / 10); //모터 속도 조절
// 0~1023의 값을 0~102까지로 조절
digitalWrite(3, LOW); // LED 끄
analogWrite(5, 0); // 모터 끄

```

선풍기
전체 코드

ch4_4_2_fan_finish

```
1 boolean SW = false;
2 unsigned long past = 0;
3 unsigned long waitTime = 0;
4
5 void setup() {
6     Serial.begin(9600);
7     pinMode(2, INPUT);
8     pinMode(3, OUTPUT);
9     pinMode(5, OUTPUT);
10 }
11
12 void loop() {
13     changeSW();
14     if (SW) {
15         if (myTimer()) {
16             SW = false;
17         } else {
18             digitalWrite(3, HIGH);
19             analogWrite(5, analogRead(A0) / 10);
20         }
21     } else {
22         digitalWrite(3, LOW);
23         analogWrite(5, 0);
24     }
25 }
26
```

```

27 void changeSW() {
28     if (digitalRead(2) == HIGH) {
29         SW = !SW;
30         while (digitalRead(2) == HIGH) {
31             Serial.println("버튼이 떨어지길 기다리는중");
32         }
33     }
34 }
35
36 boolean myTimer() {
37     unsigned long now = millis();
38
39     if (Serial.available()) {
40         char c = Serial.read();
41         past = now;
42         waitTime = c - '0';
43     }
44
45     int myTime = (now - past) / 1000;
46     Serial.println(myTime);
47
48     if (myTime >= waitTime && waitTime != 0) {
49         past = now;
50         waitTime = 0;
51         return true;
52     }
53     return false;
54 }

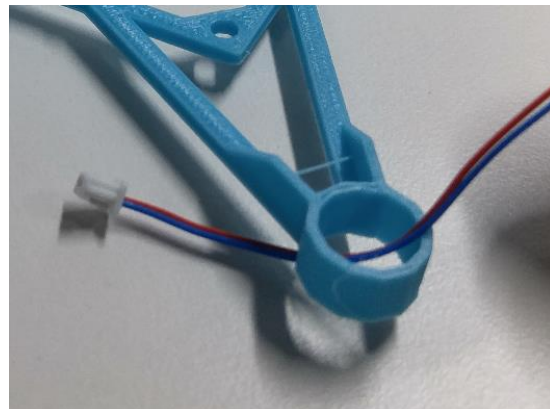
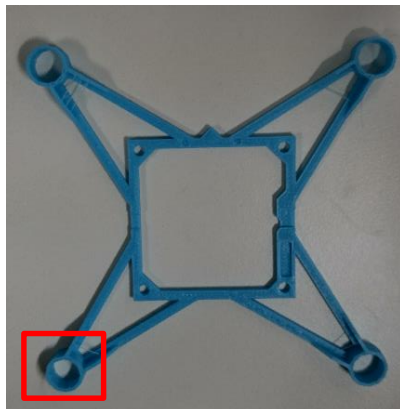
```

<그림2-11> 선풍기 만들어보기 전체 코드2

모터 연결하기

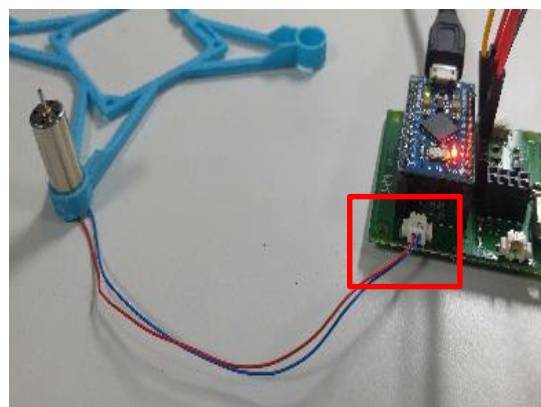
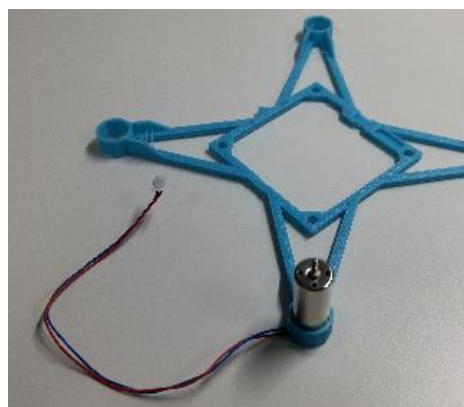
모터 연결 구성

- 1 드론 몸체의 좌측 하단에 빨파모터(선이 빨강, 파랑)를 꼬리부터 넣어서 절반정도 끼웁니다.(너무 꽉 끼우지 않습니다.)



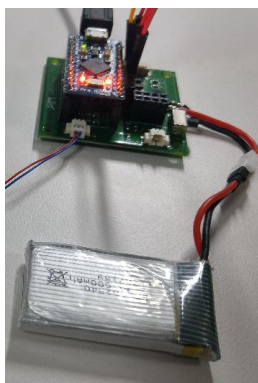
<그림2-12> 모터 연결하기

- 2 베이스 보드의 좌측 하단에 모터 꼬리를 연결합니다.



<그림2-13> 모터 연결하기

- 3 베이스 보드에 배터리를 연결하고, 모터에 R프로펠러를 끼웁니다. (L을 끼울 시 바람이 밑으로 나갑니다)

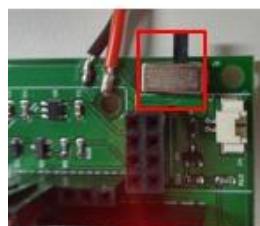
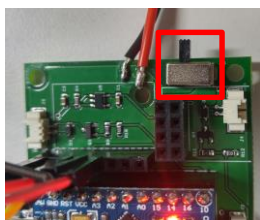


<그림2-14> 모터 연결하기

꿀TIP

베이스보드 스위치

베이스 보드의 스위치는 좌측이 꺼짐, 우측이 켜짐입니다.



03 자이로와 가속도 센서



자이로 센서는 회전값을, 가속도 센서는 가속값을 알아내는데 이용할 수 있습니다.

이러한 자이로 센서와 가속도 센서는 핸드폰에서도 쓰입니다.

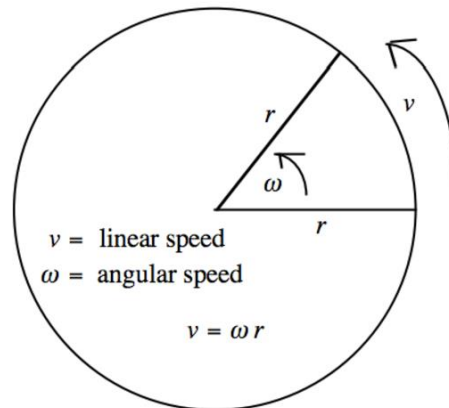
핸드폰을 세로에서 가로로 변할 때 화면이 회전하는 것을 본 적 있나요?

핸드폰 내 자이로 센서와 가속도 센서가 기울어짐을 감지해 핸드폰이 어떤 방향으로 위치하고 있는지 알 수 있습니다.

자이로 센서란?

자이로 센서

자이로 센서는 회전하는 물체의 각속도를 측정하는 센서입니다. 속도는 시간동안 움직인 거리를 나타내는 물리량인데, 각속도는 그럼 무엇일까요?



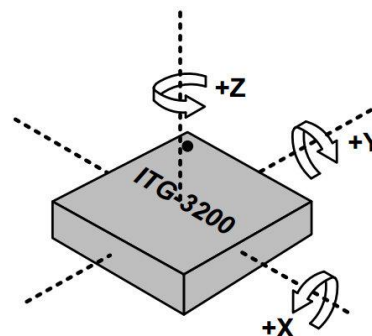
<그림1-1> 각속도

각속도는 시간동안 각도가 얼마나 변했는지를 나타내는 물리량입니다.

예를 들어 바퀴가 1초 동안 3바퀴 돌았다면

$$3 * 360\text{도} / 1 \text{ 초} = 1080\text{DPS(Degree Per Second)}$$

$$3 * 2\pi / 1\text{초} = 6\pi \text{ rad/sec}$$

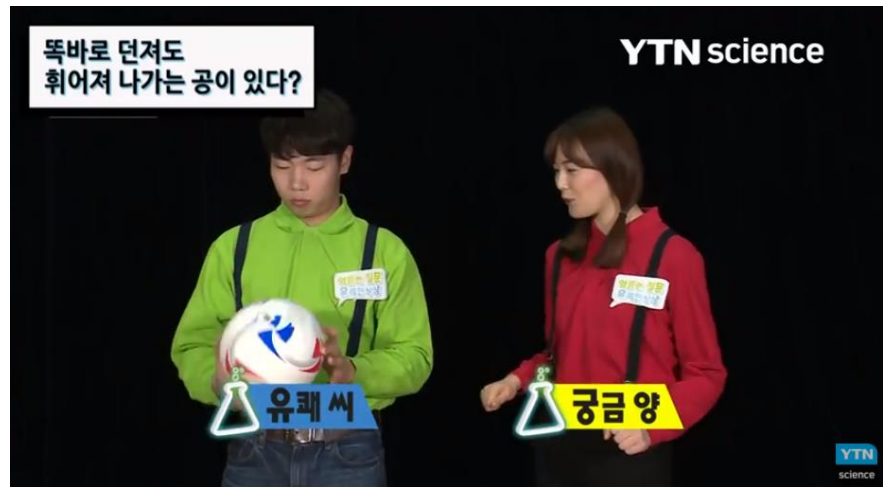


<그림1-2> 자이로 센서

위 그림처럼 자이로 센서는 물체가 x축, y축, z축으로 각각 회전함에 따라 발생하는 각속도를 읽어서 어느 방향으로 회전하고 있는지를 알아낼 수 있습니다.

전향력 (코리올리 힘)

자이로 센서는 어떤 원리로 작동하게 되는 걸까요?
회전하는 물체에는 코리올리힘이라는 가상의 힘이 작용하게 됩니다. 이를 이용하여 회전값을 측정할 수 있습니다.



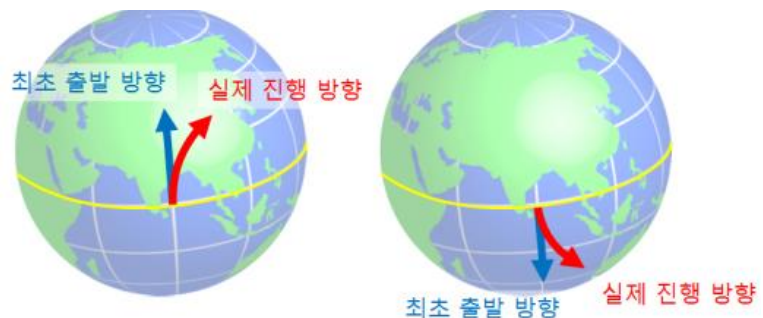
<그림1-3> 코리올리 효과

출처 : <https://www.youtube.com/watch?v=VEtUU1HJq-4>

영상에서처럼, 회전계 내부에서는 공이 휘어져서 나가게 되고, 회전계 외부에서는 직선으로 나가게 됩니다. 이렇게 휘어져서 나가게 되는 정도를 센서가 받아들이게 되고, 이 휘어짐 정도에 따라 회전이 얼마나 됐는지 구하게 됩니다.

자전과 코리올리

지구는 자전하기 때문에 최초 출발방향으로 미사일을 발사하여도, 북반구에서는 약간 오른쪽으로, 남반구에서는 약간 왼쪽으로 휘어지면서 이동하는 것처럼 보입니다.



<그림1-4> 자전과 코리올리

가속도 센서란?

가속도 센서

가속도 센서는 말 그대로 가속도를 측정하는 센서입니다. 우리 몸에 항상 작용하는 가속도 중 하나인 중력 가속도도 가속도 중 하나입니다.

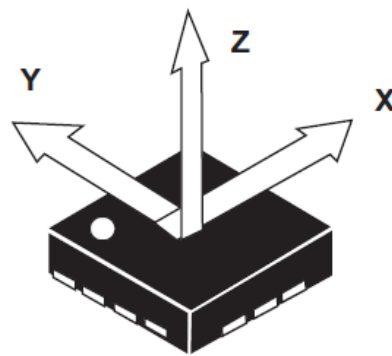
이러한 가속도는 뉴턴의 제 2법칙인 가속도의 법칙에 의해 구해집니다.

$$F=ma$$

<그림1-5> $F=ma$

몸무게가 무거운 사람은 밀어도 잘 밀리지 않습니다. 이렇게 일정한 힘이 주어졌을 때 질량에 따라 가속도는 달라지게 됩니다.

이렇게 가속도 센서는 얼마나 빠른 속도로 가속을 하게 되는지 측정할 수 있는 센서입니다.



<그림1-6> 가속도 센서

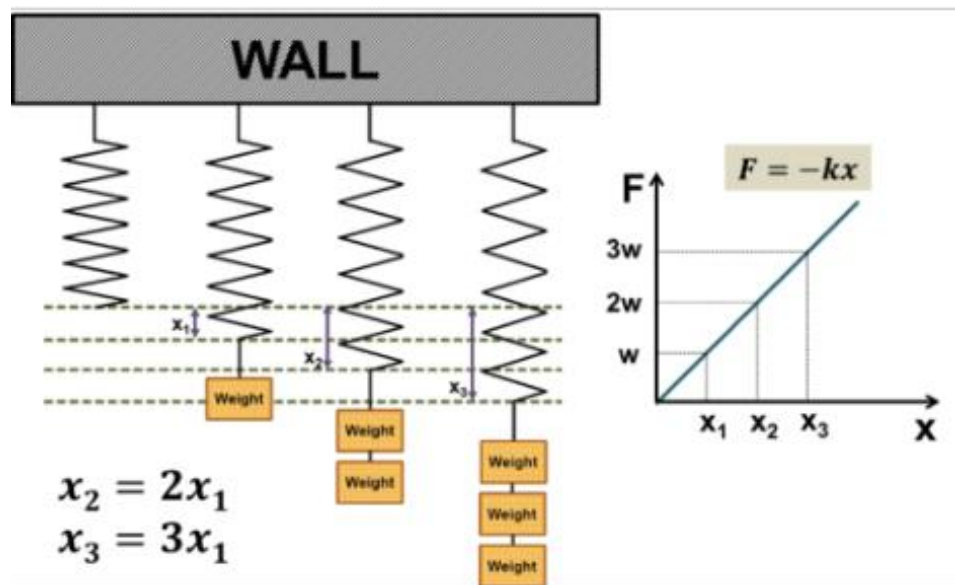
위 그림처럼 가속도 센서는 물체가 x축, y축, z축으로 각각 얼마 만큼의 가속도를 가지고 있는지 측정할 수 있으며 단위는 m/s^2 입니다.

훅(Hooke)의
법칙

이러한 가속도 센서는 어떻게 작동하는걸까요?
 훅의 법칙(Hooke's Law)은 용수철과 같이 탄성있는 물체가
 외력에 의해 늘어나거나 줄어드는 등 변형 될때, 원래
 모습으로 돌아오려고 저항하는 복원력의 크기와 변형의
 정도의 관계를 나타내는 물리법칙입니다.

$$F = kx$$

<그림1-7> 훅의 법칙



<그림1-8> 훅의 법칙 설명

$F = ma$ 랑 $F = kx$ 를 통해 가속도를 알아낼 수 있습니다.

- $ma = F = kx$
- $ma = kx$
- $a = kx/m$

이 때, k 는 상수이고, m 은 질량으로 일정합니다. 즉,
 용수철의 늘어난 길이 x 를 통해 가속도 a 를 구할 수
 있습니다.

MPU6050 센서

MPU6050

GY-521 MPU 6050 은 3축 자이로 센서와 3축 가속도 센서, 온도 센서가 한 칩 안에 구성되어 있는 센서입니다.



<그림1-9> MPU6050

GY-521 모듈은 아두이노와 I2C통신을 하게 됩니다. 이 때 VCC, GND, SCL, SDA의 4개의 핀을 이용합니다.

PIN	설명
VCC	전압
GND	접지
SCL	I2C Serial Clock
SDA	I2C Serial Data
XDA	마스터로 동작할 때 사용 (외부센서와 연결시)
XCL	마스터로 동작할 때 사용 (외부센서와 연결시)
AD0	I2C slave 주소
INT	인터럽트핀

<그림1-10> MPU6050 핀

04 I2C 통신



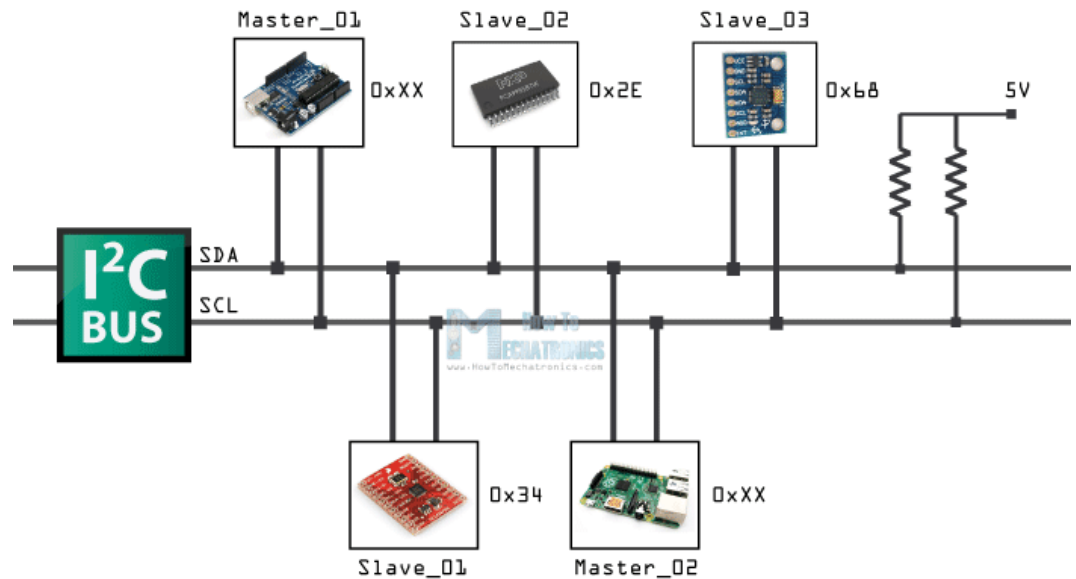
아두이노의 통신에는 시리얼 통신만 있는게 아닙니다.

통신 방식에는 I2C통신이나 SPI통신 등 다양한 통신 방법이 있는데, 앞서 배운 자이로, 가속도 센서를 사용할 땐 I2C통신을 사용하게 됩니다.

I2C통신이라고도 불리는 I2C통신은 단 두개의 선 연결만으로 최대 128개의 장치가 연결될 수 있습니다.

Inter
Integrated
Circuit

I2C통신은 2개의 신호선을 통해 여러 개의 디바이스들끼리 정보를 주고 받을 수 있는 통신 방식입니다.



<그림2-1> I2C통신

I2C통신
특징

이 때 사용되는 두 개의 핀은 각각 SCL, SDA로 각각 클락과 데이터를 담당합니다.

- SCL(Serial Clock) : 일정 주기의 클락신호를 내보낸다.
- SDA(Serial Data) : 클락신호에 따라 데이터를 보낸다.

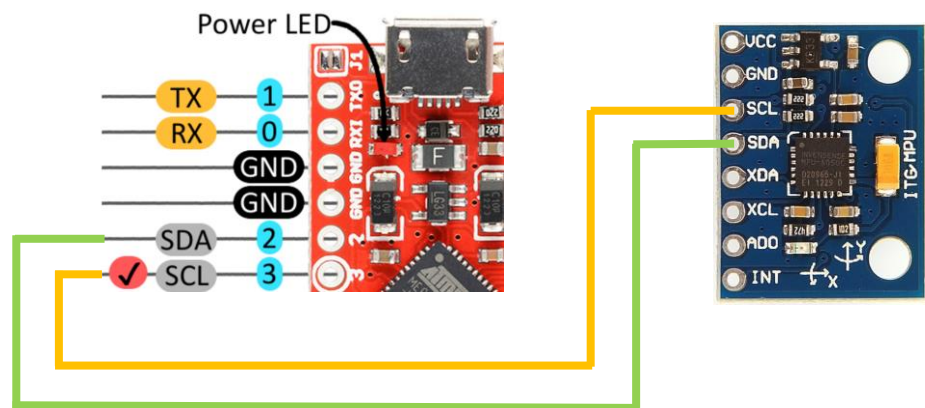
I2C통신에서 각각의 디바이스들은 고유의 주소를 가지고 있고, 이 주소를 통해서 서로를 구분할 수 있습니다.

각 디바이스는 마스터 또는 슬레이브로 동작할지가 정해지게 되는데, 대부분 아두이노가 마스터로, 센서가 슬레이브로 동작하게 됩니다.

이 때 마스터와 슬레이브는 둘 다 데이터의 입력과 출력이 가능합니다.

아두이노 I2C 연결

아두이노에서 I2C 통신을 할 때에는 다음과 같이 2번, 3번핀이 사용됩니다.(수업에서 사용하는 베이스보드를 통해 자체적으로 연결이 되어 있습니다)



<그림2-2> I2C 연결

Wire

Wire라이브러리는 아두이노에서 I2C통신을 쉽게 사용할 수 있게 해줍니다.

함수	설명
Wire.begin()	마스터 모드로 I2C통신을 시작합니다.
Wire.begin(addr)	슬레이브 모드(주소는 addr)로 I2C통신을 시작합니다.
Wire.beginTransmission(addr)	마스터에서 슬레이브로 전송을 시작하기 위한 슬레이브의 주소를 지정합니다.
Wire.write(value)	전송될 데이터를 임시 저장합니다.
Wire.endTransmission(true)	시작신호, 슬레이브 주소, 데이터, 정지신호 등을 전송합니다.
Wire.requestFrom(addr, quantity, true)	addr라는 주소의 슬레이브에게 quantity만큼의 데이터를 요청합니다.

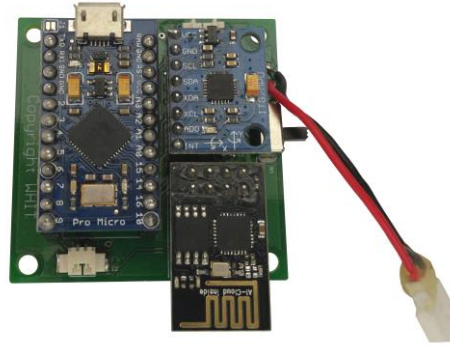
MPU6050
온도

- ① 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.


```
ch5_1_2_tmp
1 #include<Wire.h>
2
3 const int MPU_ADDRESS = 0x68;
4 int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
5
6 void setup() {
7   Wire.begin();
8   Wire.beginTransmission(MPU_ADDRESS);
9   Wire.write(0x6B);
10  Wire.write(0);
11  Wire.endTransmission(true);
12  Serial.begin(115200);
13  pinMode(5, OUTPUT);
14 }
15
16 void loop() {
17   Wire.beginTransmission(MPU_ADDRESS);
18   Wire.write(0x3B);
19   Wire.endTransmission(false);
20   Wire.requestFrom(MPU_ADDRESS, 14, true);
21
22   AcX = Wire.read() << 8 | Wire.read();
23   AcY = Wire.read() << 8 | Wire.read();
24   AcZ = Wire.read() << 8 | Wire.read();
25   Tmp = Wire.read() << 8 | Wire.read();
26   GyX = Wire.read() << 8 | Wire.read();
27   GyY = Wire.read() << 8 | Wire.read();
28   GyZ = Wire.read() << 8 | Wire.read();
29
30   // Serial.print("AcX ="); Serial.println(AcX);
31   // Serial.print("AcY ="); Serial.println(AcY);
32   // Serial.print("AcZ ="); Serial.println(AcZ);
33   Serial.print("Tmp ="); Serial.println(Tmp / 340.00 + 36.53);
34   // Serial.print("GyX ="); Serial.println(GyX);
35   // Serial.print("GyY ="); Serial.println(GyY);
36   // Serial.print("GyZ ="); Serial.println(GyZ);
37
38   // if(AcX < 0) AcX = -AcX;
39   // analogWrite(5, AcX / 200);
40 }
```

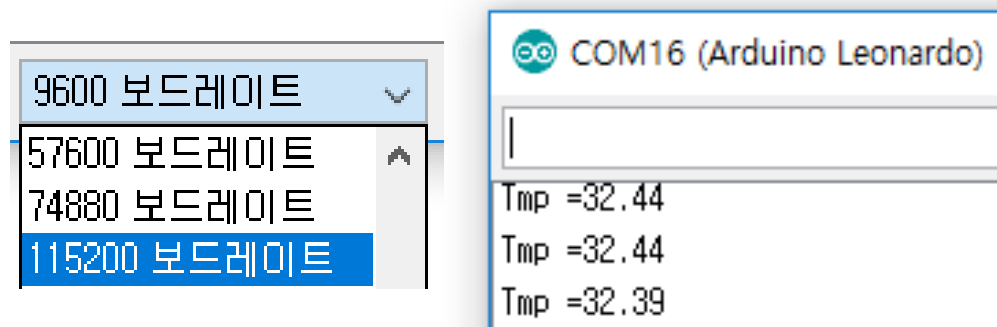
<그림2-3> I2C통신 예제

- 2 아두이노와 자이로 센서를 메인 보드에 끼워 넣습니다.



<그림2-4> 아두이노 자이로 연결

- 3 아두이노를 컴퓨터와 USB로 연결합니다.
- 4  버튼을 눌러 시리얼 모니터를 켭니다.
- 5 보드레이트를 맞춘 후, 자이로 센서에 손을 올려 온도가 변하는 것을 확인합니다.



<그림2-5> 시리얼 통신 확인

Tmp는 Temperature의 줄임말로, MPU6050센서에서 온도를 확인할 수 있습니다.
이렇게 자이로, 가속도 센서가 온도센서와 함께 존재하는 이유는 자이로, 가속도를 측정하는 센서가 온도의 영향을 받기 때문입니다.

꿀TIP

I2C 코드

<https://goo.gl/bXn6Fq>

MPU6050

온도 해석

```
#include<Wire.h>

const int MPU_ADDRESS = 0x68; //센서 주소값
int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ; //가속도, 자이로,
온도값을 받을 변수

void setup() {
  Wire.begin(); //I2C 실행
  Wire.beginTransmission(MPU_ADDRESS); //센서 주소 설정
  Wire.write(0x6B); //보낼 데이터 저장
  Wire.write(0); //보낼 데이터 저장
  Wire.endTransmission(true); //데이터 전송
  Serial.begin(115200);
  pinMode(5, OUTPUT);
}

void loop() {
  Wire.beginTransmission(MPU_ADDRESS); //센서 주소 설정
  Wire.write(0x3B); // 보낼 데이터 저장
  Wire.endTransmission(false); //데이터 전송
  Wire.requestFrom(MPU_ADDRESS, 14, true); //반환값 확인

  AcX = Wire.read() << 8 | Wire.read(); //데이터 저장
  AcY = Wire.read() << 8 | Wire.read();
  AcZ = Wire.read() << 8 | Wire.read();
  Tmp = Wire.read() << 8 | Wire.read();
  GyX = Wire.read() << 8 | Wire.read();
  GyY = Wire.read() << 8 | Wire.read();
  GyZ = Wire.read() << 8 | Wire.read();

  // Serial.print("AcX="); Serial.println(AcX); // 데이터 확인
  // Serial.print("AcY="); Serial.println(AcY);
  // Serial.print("AcZ="); Serial.println(AcZ);
  Serial.print("Tmp="); Serial.println(Tmp / 340.00 + 36.53);
  // Serial.print("GyX="); Serial.println(GyX);
  // Serial.print("GyY="); Serial.println(GyY);
  // Serial.print("GyZ="); Serial.println(GyZ);

  // if(AcX < 0) AcX = -AcX;
  // analogWrite(5, AcX / 200);
}
```


MPU6050 가속도

1 위에서 작성한 코드를 다음과 같이 변형하여 아두이노에 업로드합니다.

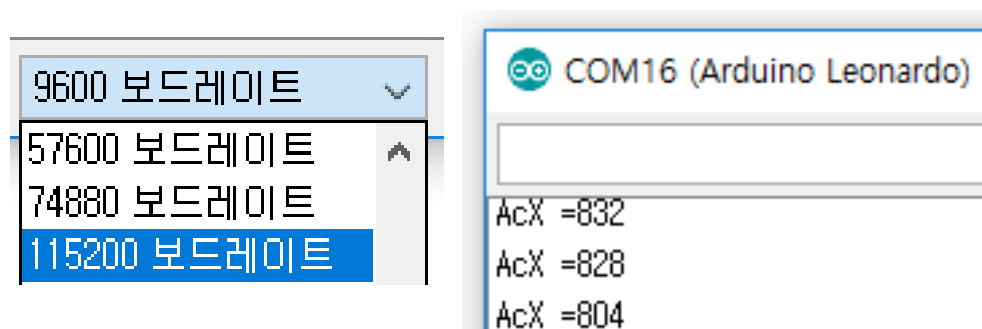
- 30번째 줄의 주석을 해제합니다.
- 33번째 줄에 주석을 넣습니다.
- 38, 39번째 줄의 주석을 해제합니다.

```
30 Serial.print("AcX ="); Serial.println(AcX);
31 // Serial.print("AcY ="); Serial.println(AcY);
32 // Serial.print("AcZ ="); Serial.println(AcZ);
33 // Serial.print("Tmp ="); Serial.println(Tmp / 340.00 + 36.53);
34 // Serial.print("GyX ="); Serial.println(GyX);
35 // Serial.print("GyY ="); Serial.println(GyY);
36 // Serial.print("GyZ ="); Serial.println(GyZ);
37
38 if(AcX < 0) AcX = -AcX;
39 analogWrite(5, AcX / 200);
40 }
```

<그림2-6> MPU6050 가속도 확인

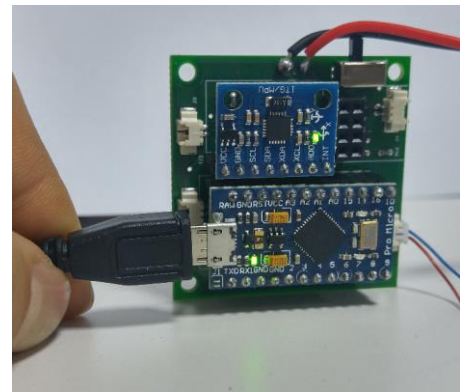
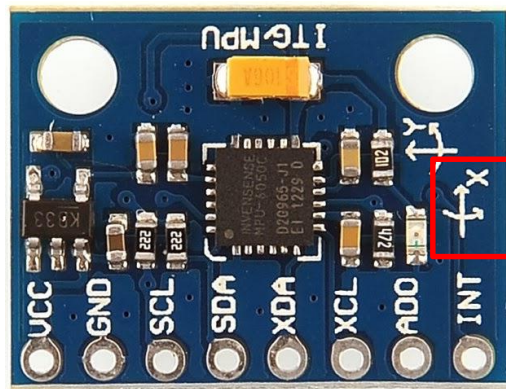
2  버튼을 눌러 시리얼 모니터를 켭니다.

3 보드레이트를 맞춘 후, AcX의 값을 확인합니다.



<그림2-7> 시리얼 통신 확인

- 4 배터리와 모터를 연결하고 스위치를 켭니다.(33페이지 참조)
- 5 자이로 센서의 x화살표가 위를 향하게 하면서 모터 속도의 변화를 확인합니다.



<그림2-8> MPU6050 방향 변화

MPU6050 가속도 해석

```
Serial.print("AcX="); Serial.println(AcX); // 데이터 확인
// Serial.print("AcY="); Serial.println(AcY);
// Serial.print("AcZ="); Serial.println(AcZ);
// Serial.print("Tmp="); Serial.println(Tmp / 340.00 + 36.53);
// Serial.print("GyX="); Serial.println(GyX);
// Serial.print("GyY="); Serial.println(GyY);
// Serial.print("GyZ="); Serial.println(GyZ);
```

```
if(AcX < 0) AcX = -AcX; //만약 AcX가 0보다 작으면 양수로 전환
analogWrite(5, AcX / 200); // 모터 출력
}
```

AcX는 Accelerometer의 줄임말로 X축으로의 가속도를 의미합니다. AcX의 값은 대략 16000 ~ -16000 사이의 값을 가집니다. x를 화살표 방향 위로 했을 때 최대값을 갖습니다. 이는 x축 방향으로 작용하는 중력가속도의 값을 의미합니다.


MPU6050 자이로

1 위에서 작성한 코드를 다음과 같이 변형하여 아두이노에 업로드합니다.

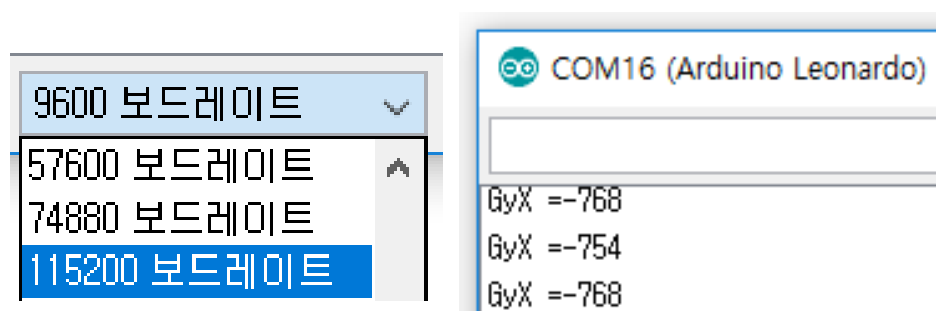
- 30번째줄에 주석을 넣습니다.
- 34번째줄에 주석을 해제합니다.
- 38, 39번째 줄의 AcX를 GyX로, 200을 400으로 바꿉니다.

```
30 // Serial.print("AcX ="); Serial.println(AcX);
31 // Serial.print("AcY ="); Serial.println(AcY);
32 // Serial.print("AcZ ="); Serial.println(AcZ);
33 // Serial.print("Tmp ="); Serial.println(Tmp / 340.00 + 36.53);
34 Serial.print("GyX ="); Serial.println(GyX);
35 // Serial.print("GyY ="); Serial.println(GyY);
36 // Serial.print("GyZ ="); Serial.println(GyZ);
37
38 if (GyX < 0) GyX = -GyX;
39 analogWrite(5, GyX / 400);
40 }
```

<그림2-9> MPU6050 자이로 확인

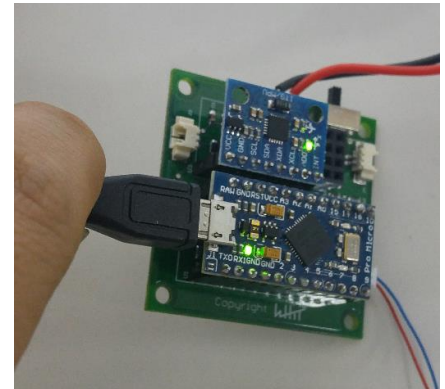
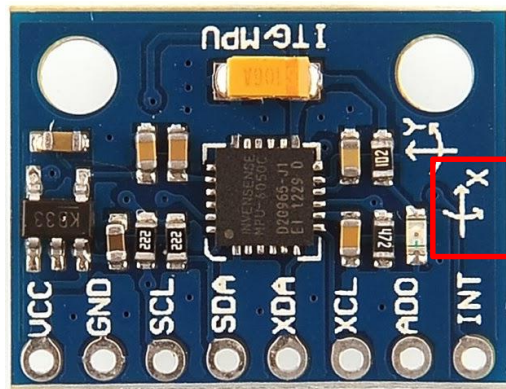
2  버튼을 눌러 시리얼 모니터를 켵니다.

3 보드레이트를 맞춘 후, GyX의 값을 확인합니다.



<그림2-10> 시리얼 통신 확인

- 4 배터리와 모터를 연결하고 스위치를 켭니다.(17페이지 참조)
- 5 자이로 센서의 x화살표를 넘어가는 화살표 방향으로 회전을 시킬 때 모터 속도의 변화를 확인합니다.



<그림2-11> MPU6050 방향 변화

MPU6050 자이로 해석

```
// Serial.print("AcX="); Serial.println(AcX); // 데이터 확인
// Serial.print("AcY="); Serial.println(AcY);
// Serial.print("AcZ="); Serial.println(AcZ);
// Serial.print("Tmp="); Serial.println(Tmp / 340.00 + 36.53);
Serial.print("GyX="); Serial.println(GyX);
// Serial.print("GyY="); Serial.println(GyY);
// Serial.print("GyZ="); Serial.println(GyZ);
```

```
if(GyX < 0) GyX = -GyX; //만약 GyX가 0보다 작으면 양수로 전환
analogWrite(5, GyX / 400); // 모터 출력
}
```

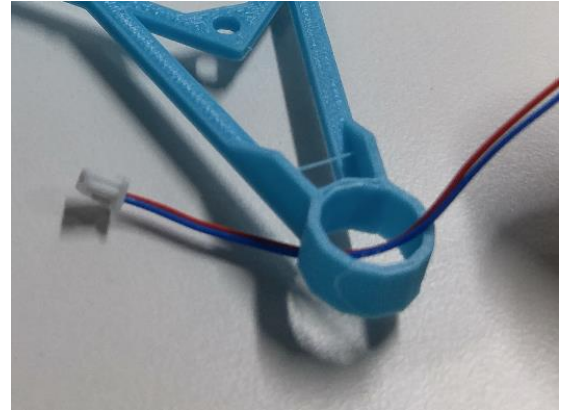
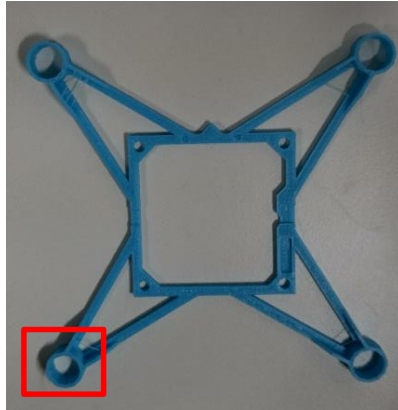
GyX는 Gyro Sensor의 줄임말로 x축으로의 회전을 의미합니다. GyX의 값은 대략 32000 ~ -32000 사이의 값을 가집니다. x화살표를 넘어가는 화살표 방향으로 회전을 했을 때 최대값을 가집니다.

이는 회전 방향으로 작용하는 각속도의 값을 의미합니다.

모터 연결하기

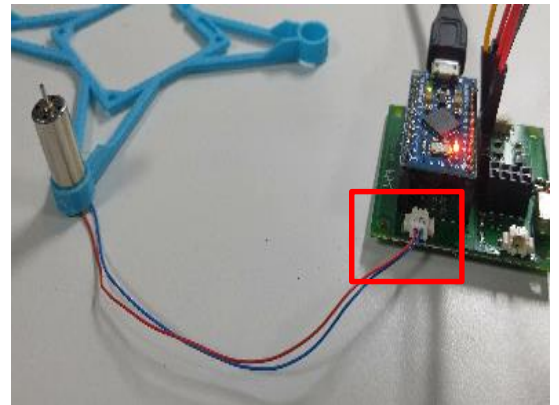
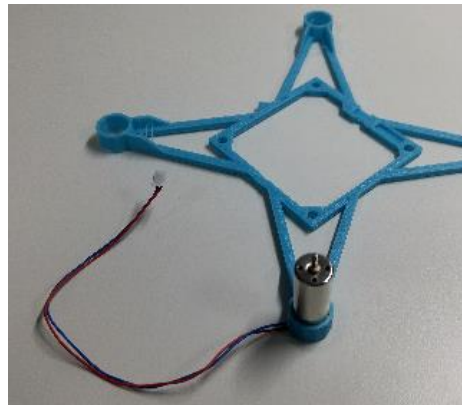
모터 연결 구성

- 1 드론 몸체의 좌측 하단에 빨파모터(선이 빨강, 파랑)를 꼬리부터 넣어서 절반정도 끼웁니다.(너무 꽉 끼우지 않습니다.)



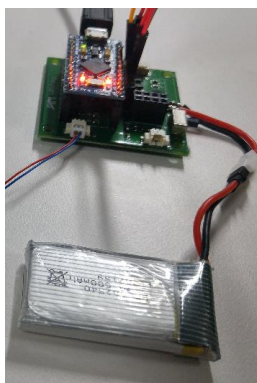
<그림2-12> 모터 연결하기

- 2 베이스 보드의 좌측 하단에 모터 꼬리를 연결합니다.



<그림2-13> 모터 연결하기

- 3 베이스 보드에 배터리를 연결하고, 모터에 R프로펠러를 끼웁니다. (L을 끼울 시 바람이 밑으로 나갑니다)

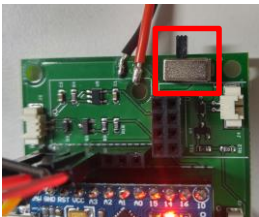


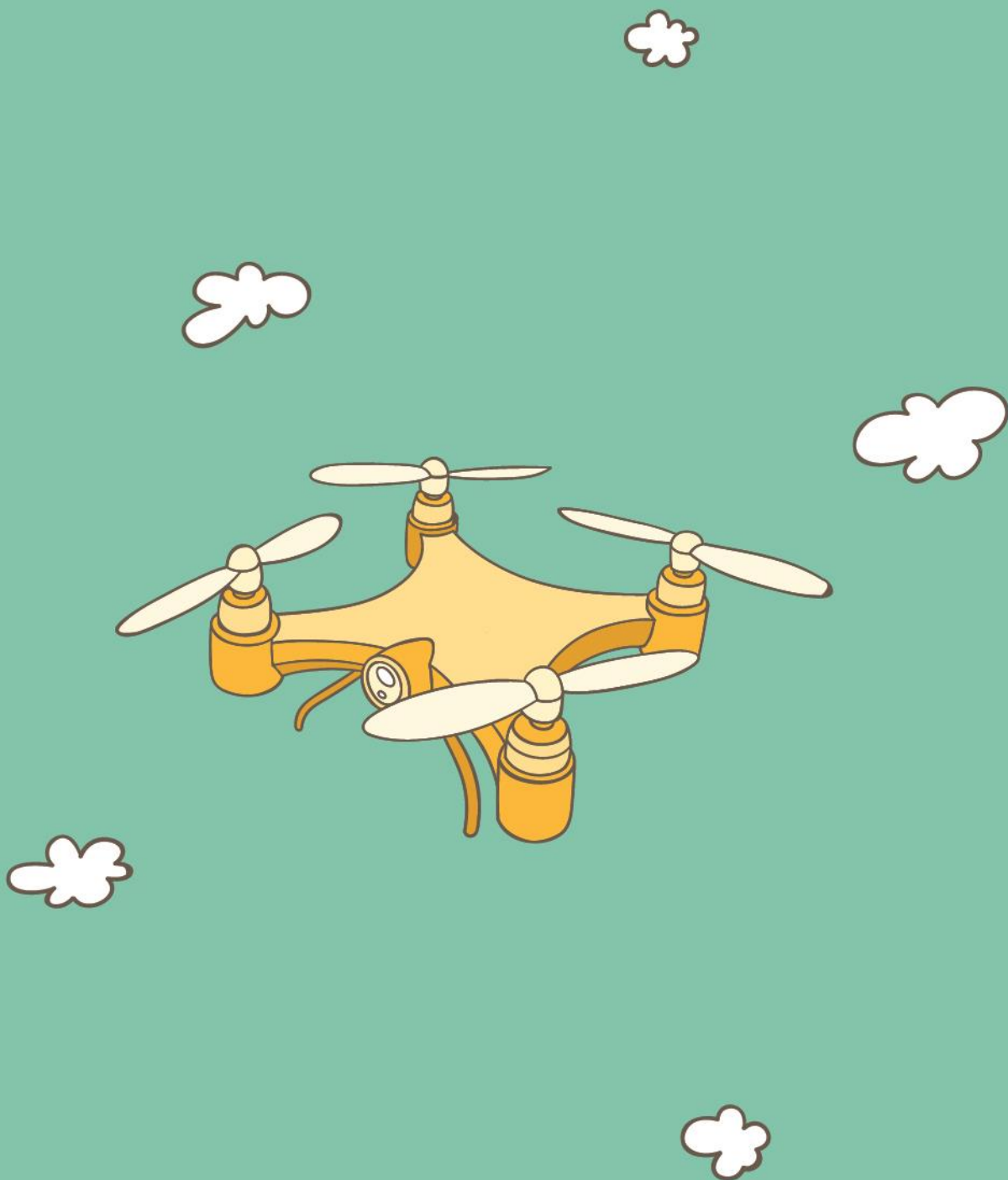
<그림2-14> 모터 연결하기

꿀TIP

베이스보드 스위치

베이스 보드의 스위치는 좌측이 꺼짐, 우측이 켜짐입니다.





WHIT