201221341 유진후
201221351 이태우

기계학습을 이용한 드론제어

DRONE HOVERING

2017-05-10 캡스톤

목차

개발 목표

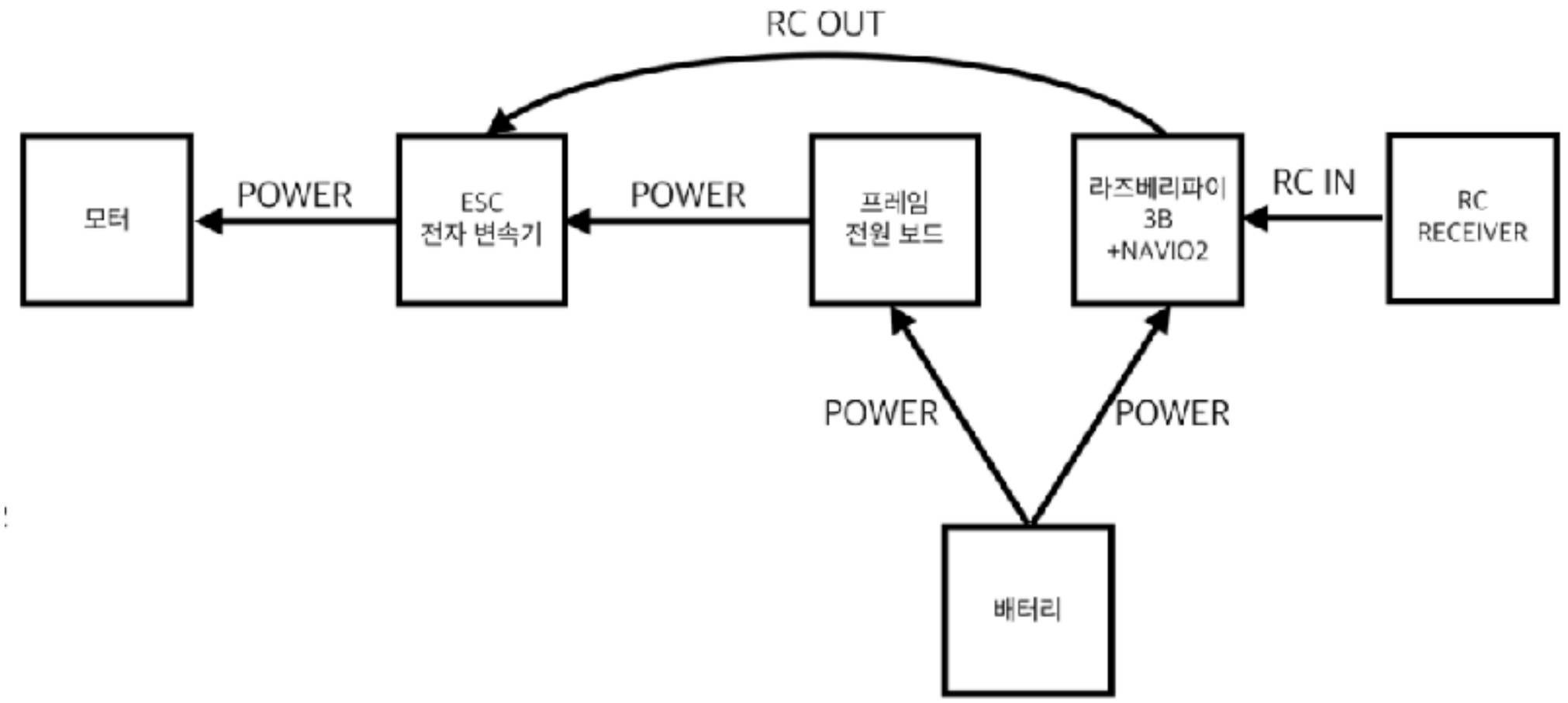
현재 차세대 운송 수단 및 국내 외 연구 동향을 살펴보면 다양한 실내 비행체의 연구 개발 및 시도가 이루어져 왔으나 새로운 실내 환경에 대한 학습이 용이한 무인 비행체 자율 비행 조종 기술에 대한 실용화 사례는 찾기 어렵다. 또한 실내 위치 측위를 위한 비콘 등 사전 설치 작업이 필요한 경우도 있어 새로운 실내 환경에 적용하기가 어렵다. 이를 극복하기 위해 기계학습을 이용하여 비행체를 실내 환경에서 자율 비행이 가능하도록 하는 연구를 진행하여 기술을 확보해야 할 필요성이 있다.

기존 실내 비행체의 자율 주행은 실내 측위 기술에 기반하여 비행을 제어하는데, 정확한 위치에 비콘이 설치되어 있지 않은 경우에는 정확성이 떨어진다. 이에 고성능으로 공개된 기계학습 API인 텐서플로우를 활용하여, 실용화가 용이한 실내 자율 비행 조종기법을 개발하려고 한다.

개발 범위

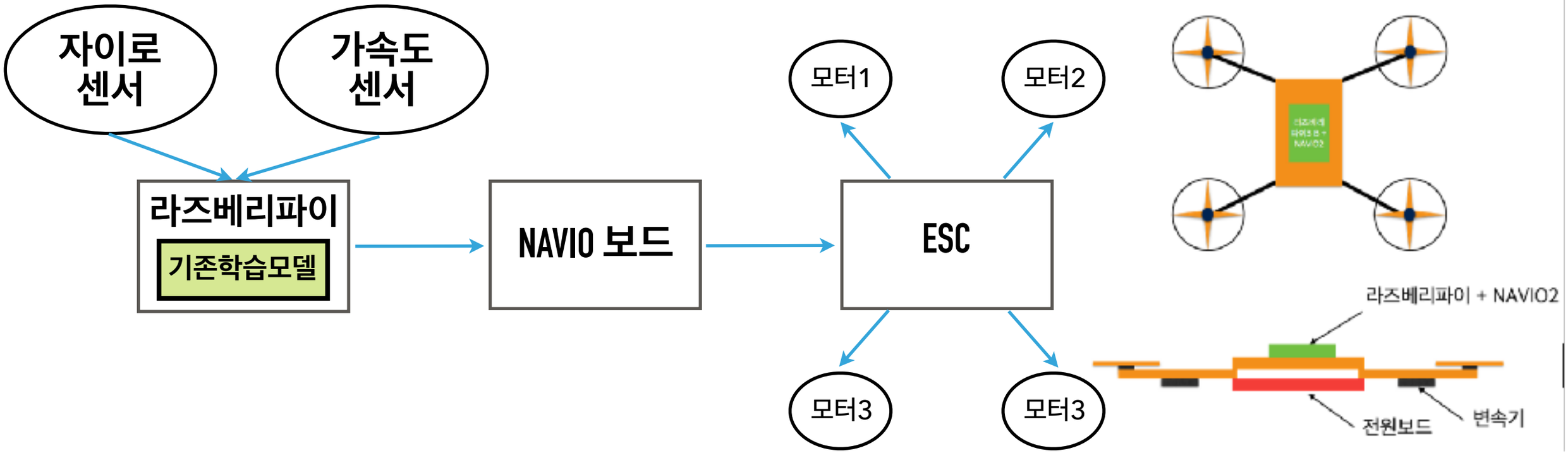
1. 자율주행을 하기위한 기반 기술로서 호버링을 우선적으로 구현한다.
2. 드론의 자세제어 및 주행기법에 기계학습을 접목하여 실내 자율 주행 드론 개발한다.
3. 드론의 자세 제어 학습을 위해 자이로 센서, 가속도 센서의 데이터를 수집한다.

드론 구성도



드론은 크게 모터 동작부와 제어부로 나뉜다. 제어부는 라즈베리 파이와 각종 센서가 부착이 되어 있는 NAVIO 보드로 구성되어 있고, RC에서 들어오는 제어 신호를 NAVIO보드에서 처리 한다. 이를 통해 ESC에 출력값을 전달하고 ESC는 모터의 출력을 제어 한다.

시스템 구성도



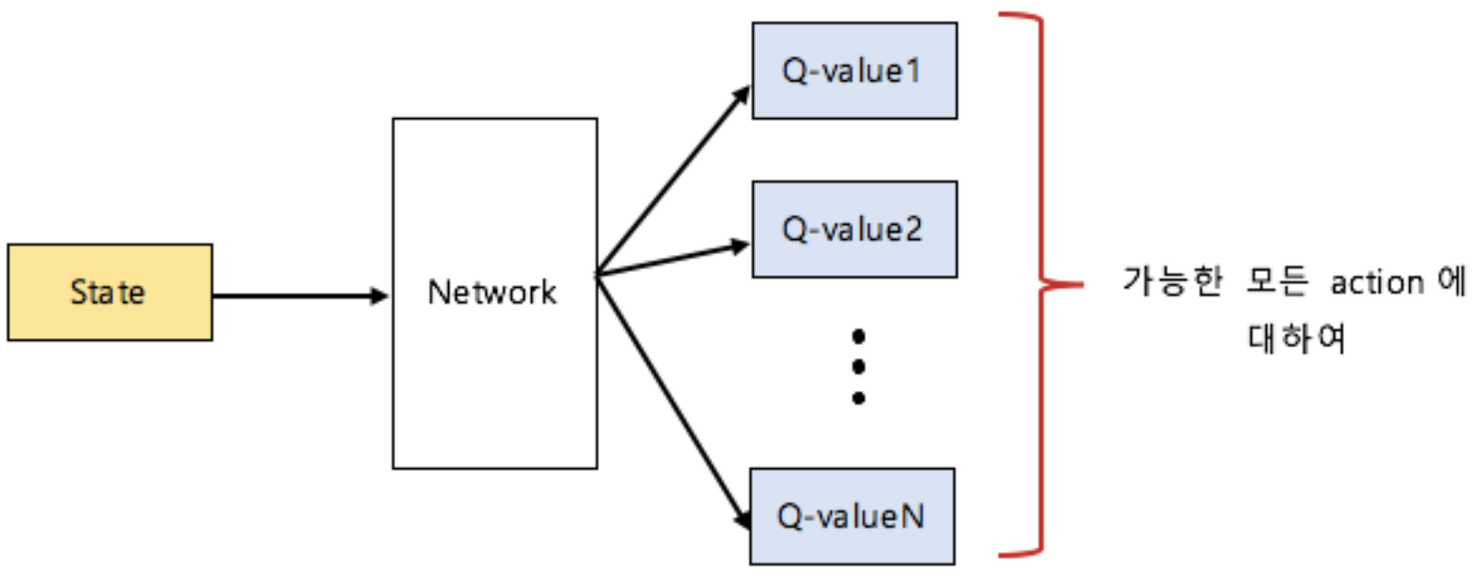
1. 자이로센서와 가속도 센서를 통해 X, Y, Z축 상의 드론의 기울기를 구한다.
2. 기울기 정보를 라즈베리파이로 전달하고, 전달받은 정보를 통해 라즈베리파이 내에있는 기존학습모델에 제공한다.
3. 기존에 미리 학습되어있는 모델을 바탕으로 드론의 날개를 움직이게하는 4개의 모터의 출력값을 조절한다.
4. 모터의 움직임을 통해 새롭게 얻어진 자이로센서와 가속도 센서를 통해 얻어진 기울기 데이터를 바탕으로 기존학습모델에 추가로 학습을 한다.

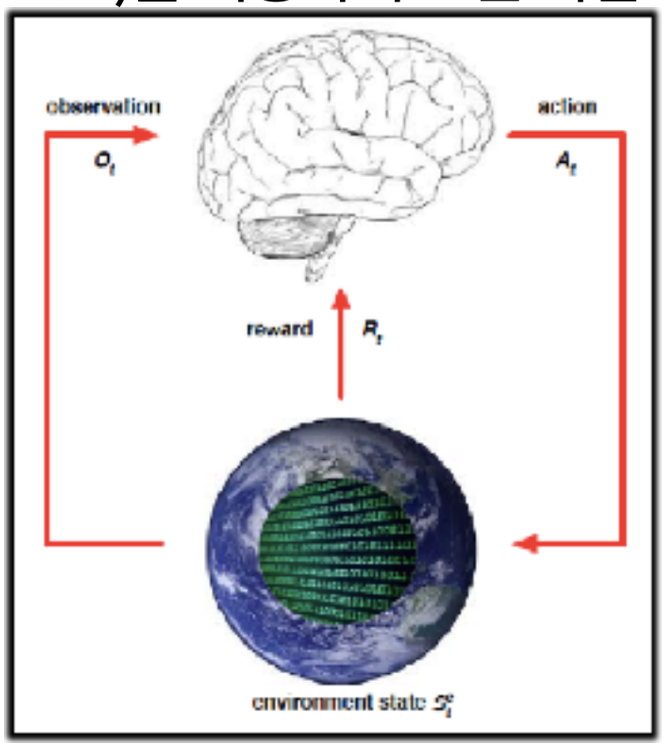
기존학습데이터와 추가로(리얼타임) 얻어지는 데이터를 바탕으로 학습된 모델을 이용하여 드론 모터들의 출력값들을 조절한다

구현 방법

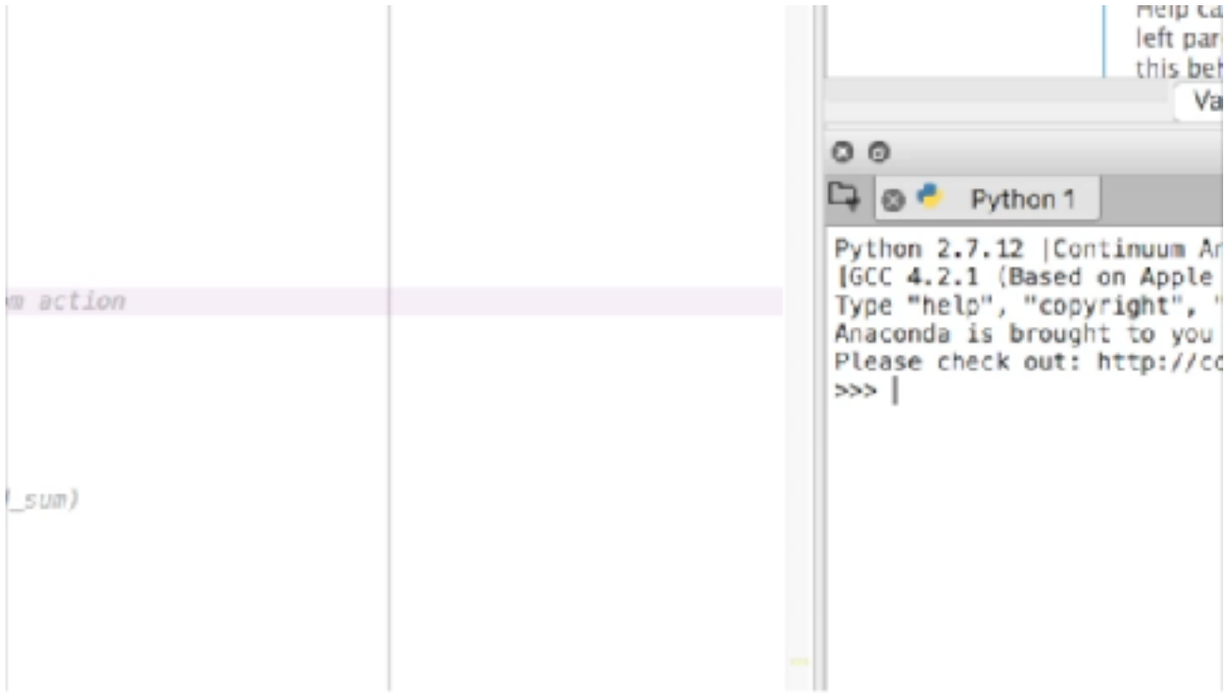
- 강화 학습을 통한 드론의 자세 제어 및 수평제어 (Hovering)
  - 강화학습의 한 종류인 Q-Learning을 이용하여 현재상태(state)에서의 올바른 행위(action)를 결정
  - Q-Learning이란?

    현재 상태에서의 행위를 미래 행위들로부터 얻게 되는 총 보답을 예측하는 행위값에 대응시키는 행위 함수를 학습하는 방법 ('행위함수'를 Q라고 지칭함)
  - 강화학습과 Neural Network를 합친 DQN(Deep Q-Netwrok)을 이용하여 Q를 학습시킴

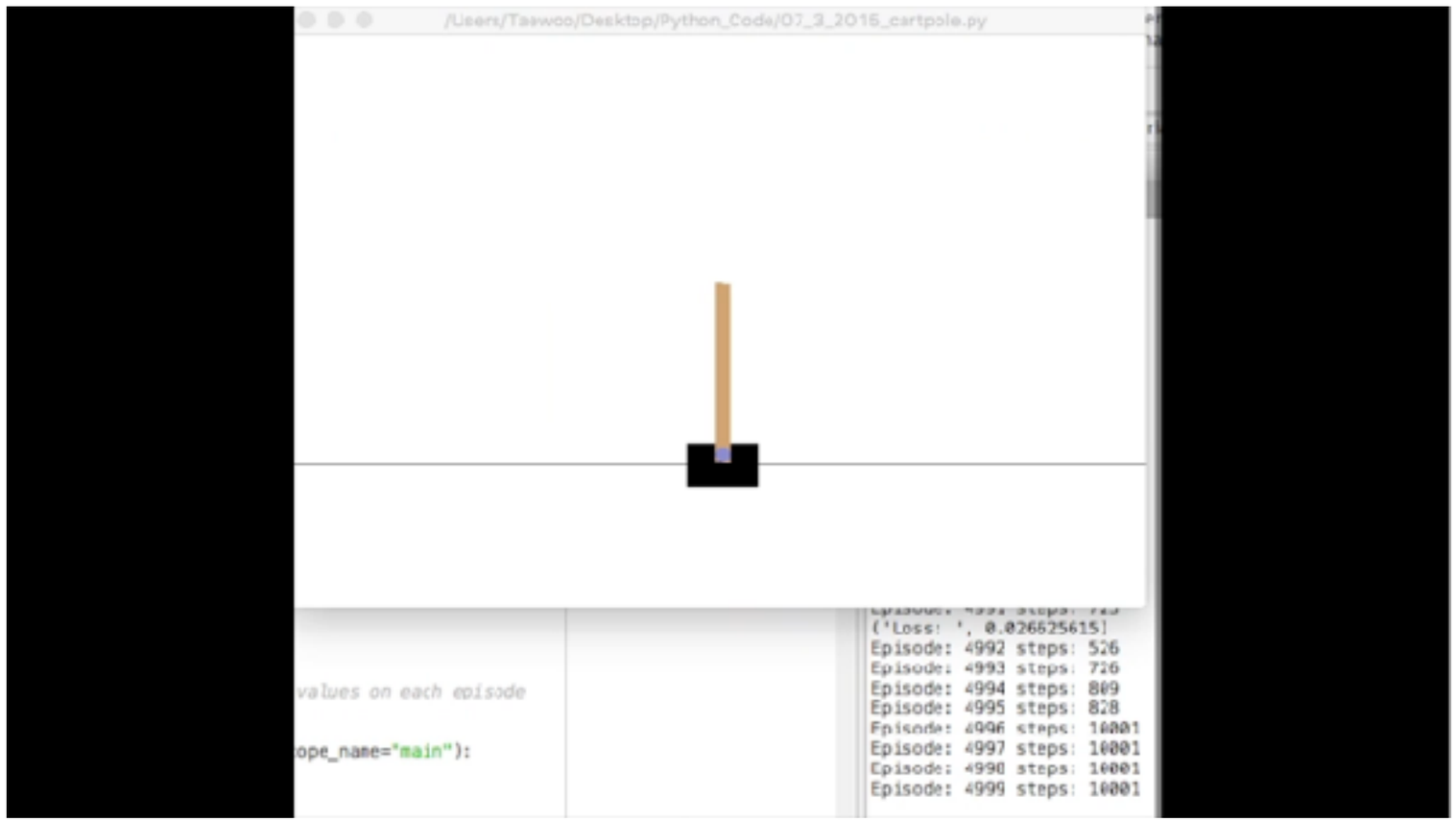




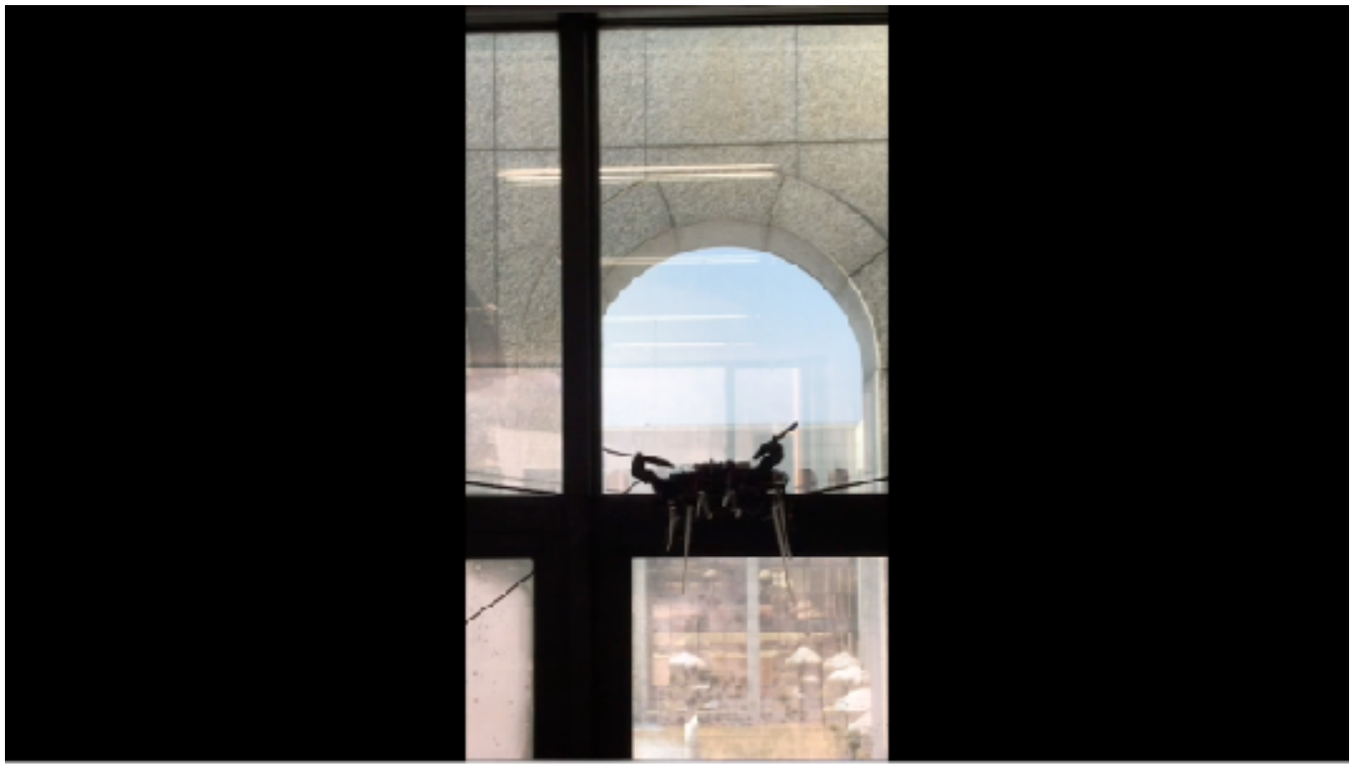
강화학습 구현 현황 (학습전)



강화학습 구현 현황 (학습후)



드론 진행사항 (수평 제어)

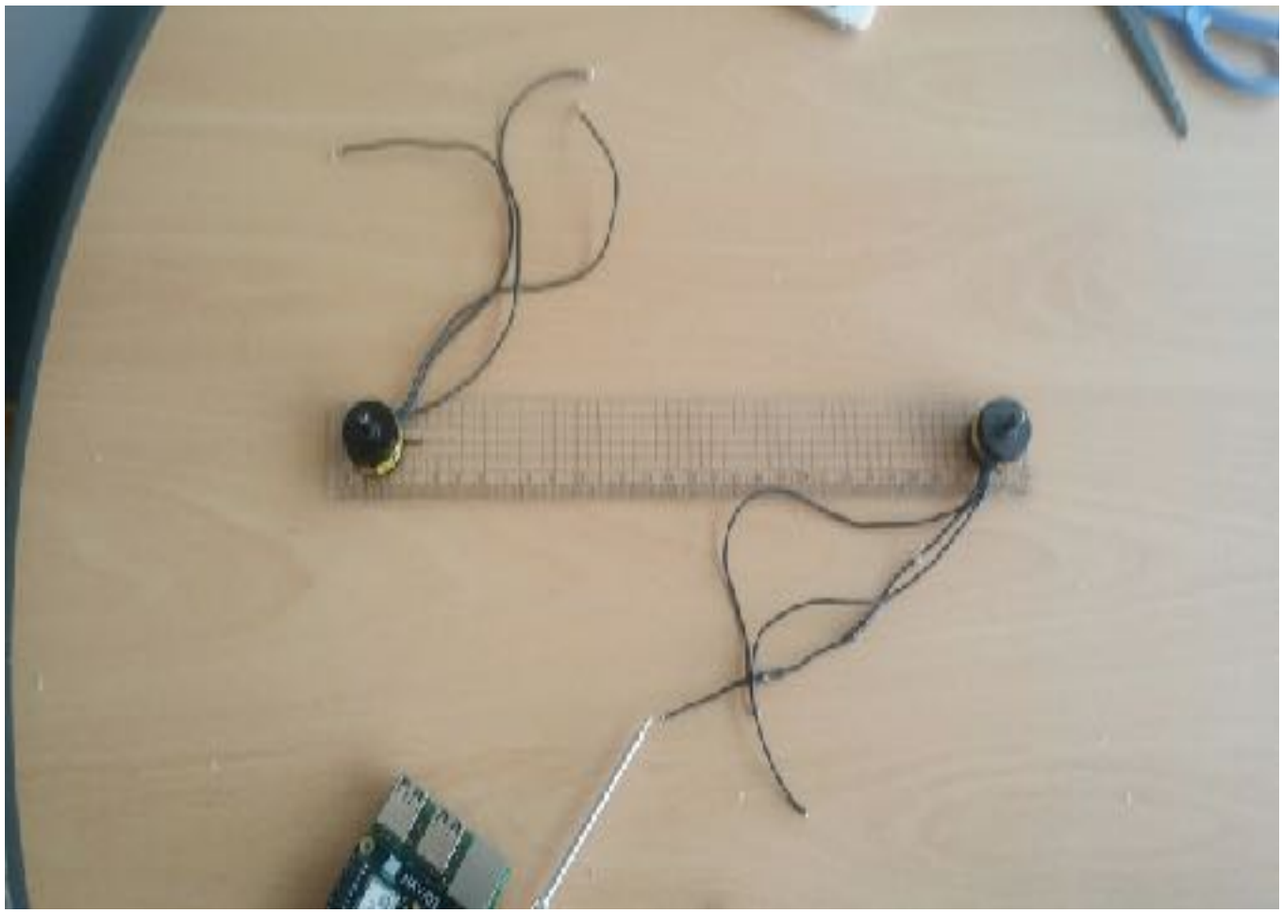


Q-LEARNING PROTOTYPE - 제작

강화학습을 이용하여 Hovering을 구현해보기 위한
수평제어 테스트 드론

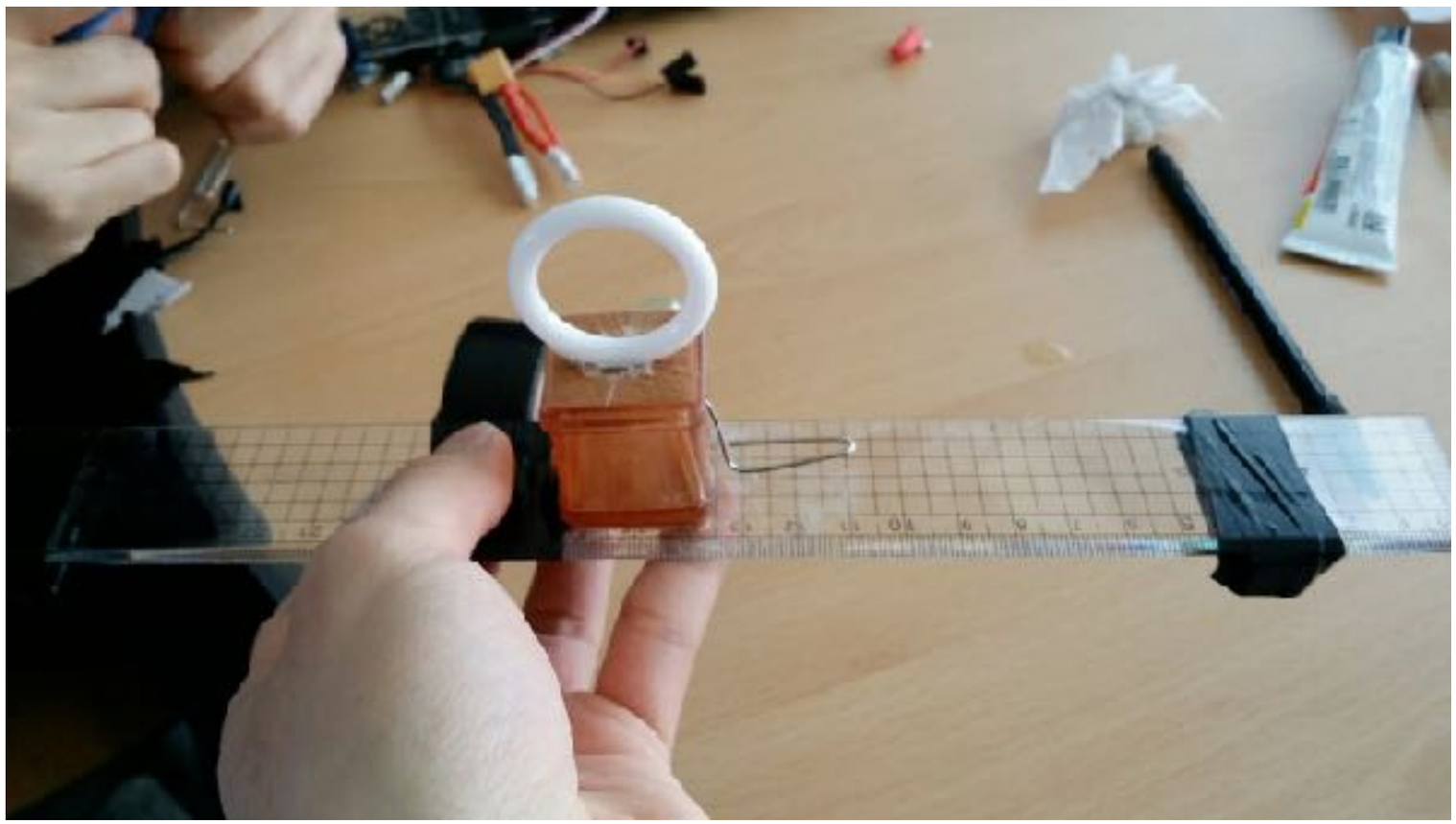
구성품 : Raspberry pi3, Navio FC shield, 브러시 리스 모터,
프로펠러, 3Cell 1300mAh Li-PO Bttery, 4 in 1 ESC
30cm자 2개, 투명 의자 발 캡 2개, 클립 1개, 커튼고리 1개

Q-LEARNING PROTOTYPE - 제작 1



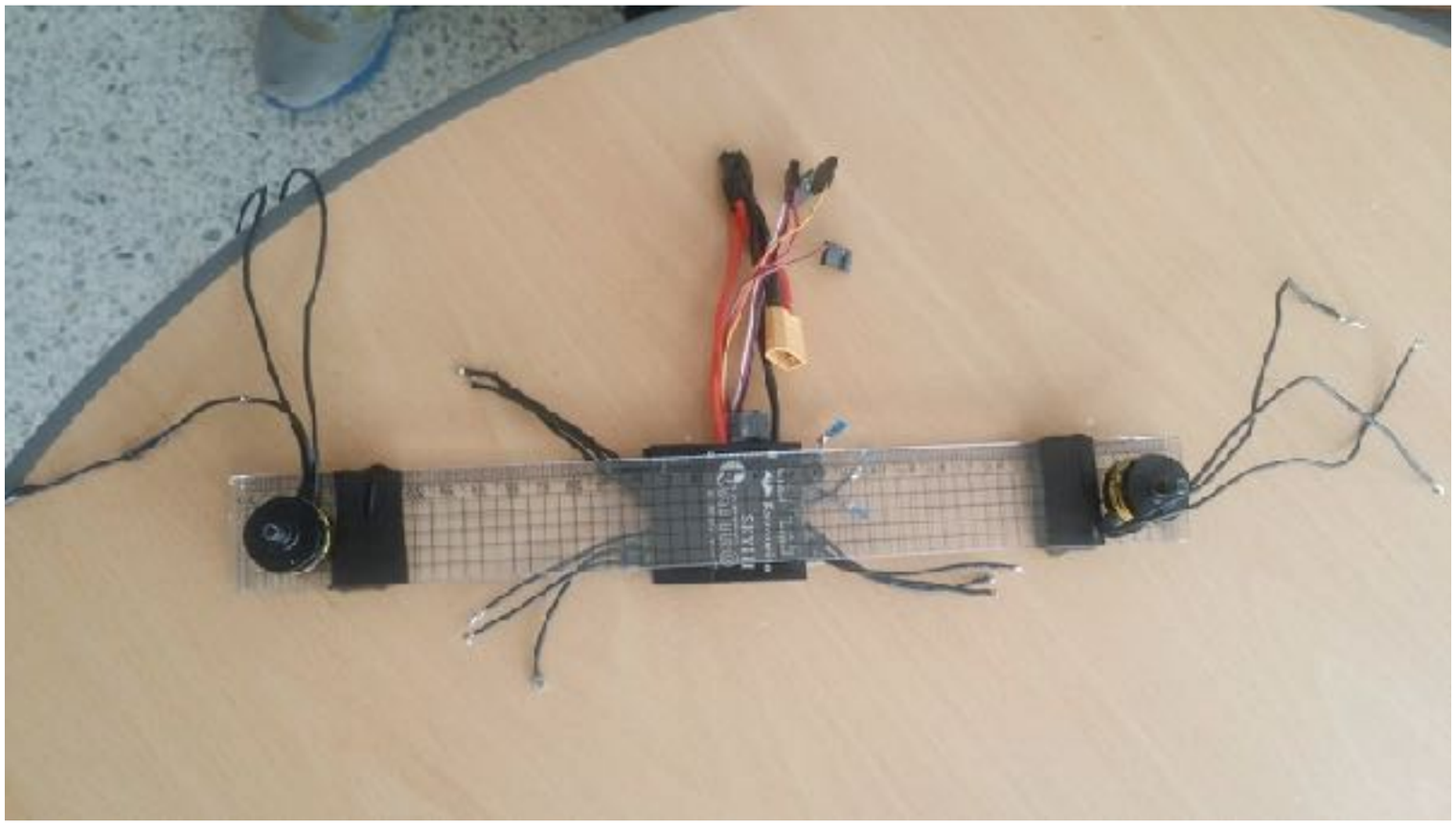
30cm 자에 나사로 구멍을 만들어 모터 결합

Q-LEARNING PROTOTYPE - 제작 2



지지대 역할과 드론이 시소운동을 할 수 있게 해주는 고리 부착,
자와 고리를 연결하기 위하여 의자 발 캡에 고리를 끼우고 캡을 자에 고정,
캡에 구멍을 뚫어 고리를 클립으로 고정한 뒤 자와 클립을 고정

Q-LEARNING PROTOTYPE - 제작 3

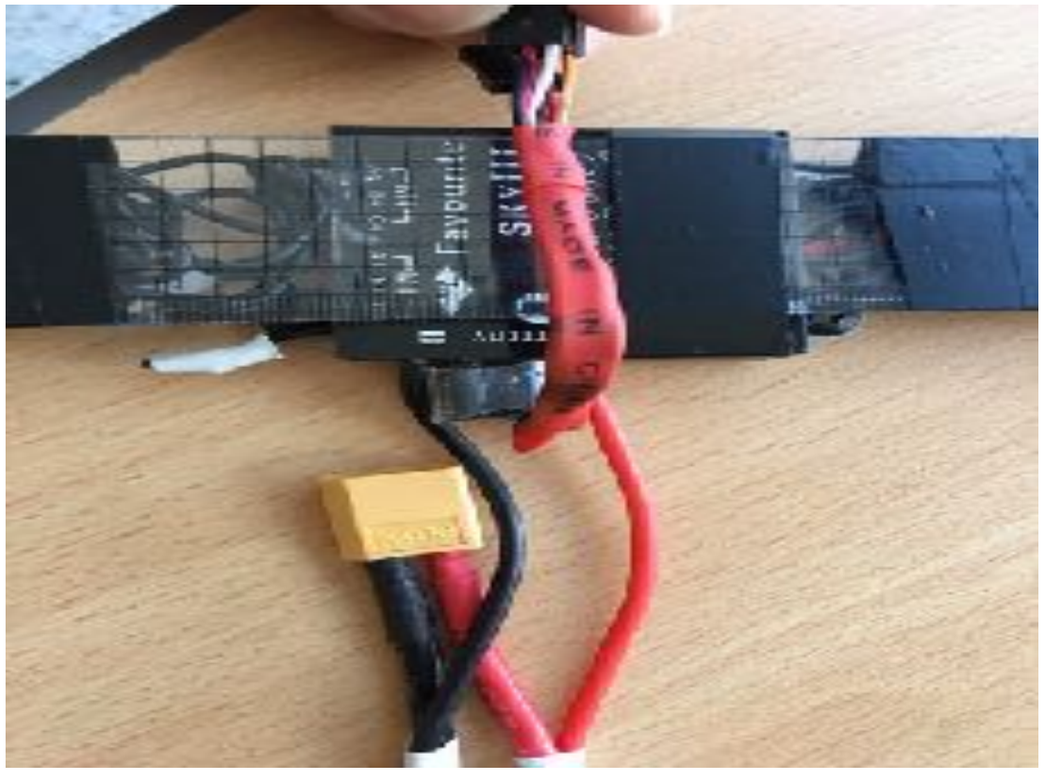


ESC전자 변속기에 XT60 커넥터 납땜 작업

**XT60 커넥터 : 배터리와 ESC연결하는 모듈

30cm자와 또 다른 30cm자 사이에 ESC 부착

Q-LEARNING PROTOTYPE - 제작 4



수축 튜브로 ESC와 모터간의 연결선 정리

**수축튜브:열을 가하면 크기가 줄어드는 튜브

배터리 커넥터를 수축 튜브로 마감 작업

Q-LEARNING PROTOTYPE - 제작 5

빨래 건조대를 뒤집어 두 다리 사이의 홈에 커튼 봉을 얹고,

고리가 달린 드론을 커튼봉에 끼움

HOVERING 이후의 계획

** Mission**

1. 특정 방향으로 이동하다가 정지해서 Hovering

2. 초음파센서를 이용하여 복도에서 벽에 부딪치지 않고 주행

3. 호버링 하는 상태에서 물건이 갑자기 나타났을 때 (공이 날아오는 등) 피하기

3월 29일 까지의 진행상황

1. 왼쪽 모터 제어, 오른쪽 모터 제어, 양쪽 모터 제어가 코드로 제대로 제어가 되지 않았던 이유?

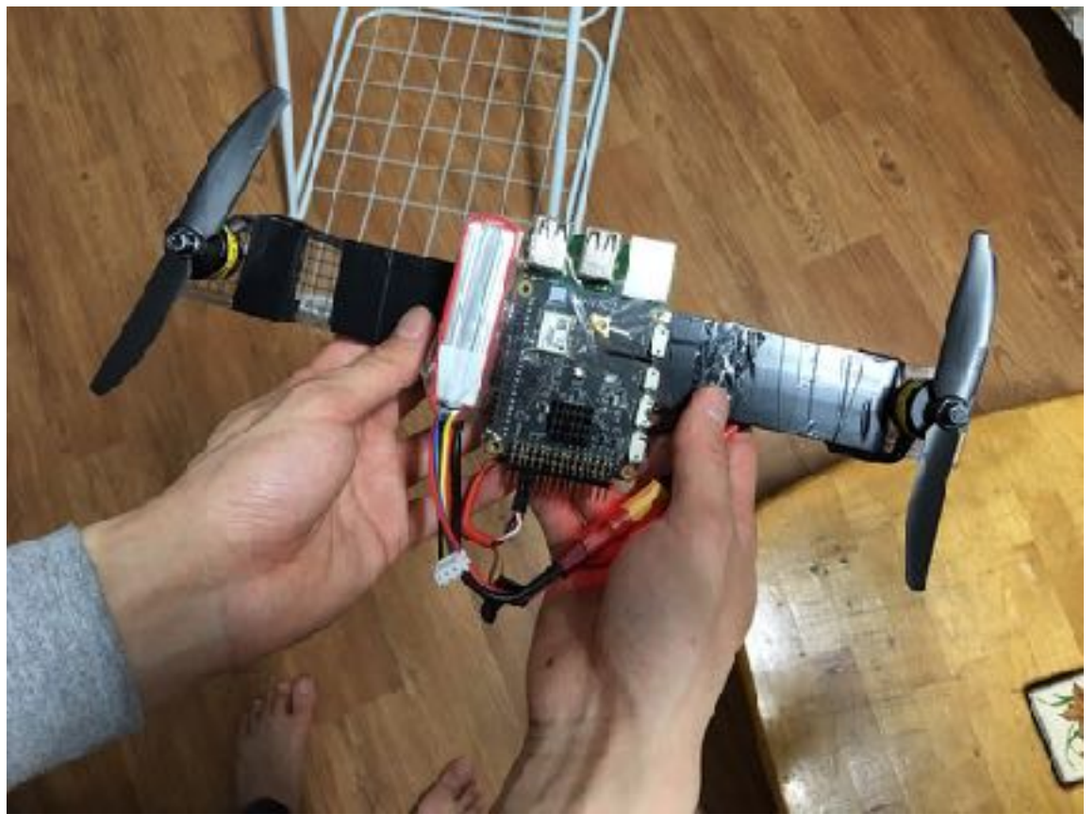
0, 1번 포트로 연결을 하고 Mission Planner를 사용해서 각각의 모터를 제어 했을 때는 문제가 발생하지 않았습니다.

그러나 0, 1번 포트와 연결해서 각 동작을 실행하는 코드를 실행해 보았을 때 계속 한 쪽 모터만 돌아가는 등 원하는 동작으로 실행이 되지 않아서 1, 2번 포트로 연결을 바꾸어 동일한 코드를 실행해 보았더니 잘 되었습니다.

0, 1번 포트 연결에서 Mission Planner로 제어시 문제가 없었으나 코드상 실행에서는 왜 문제가 발생했는지는 아직 의문입니다. (동일한 코드로 포트만 바꿔서 실행했을때 원하는 대로 잘 제어가 되는 것을 보면 코드 문제는 아닌듯)

2. 라즈베리파이 + Navio Board, 배터리의 배치는 어떤식으로?

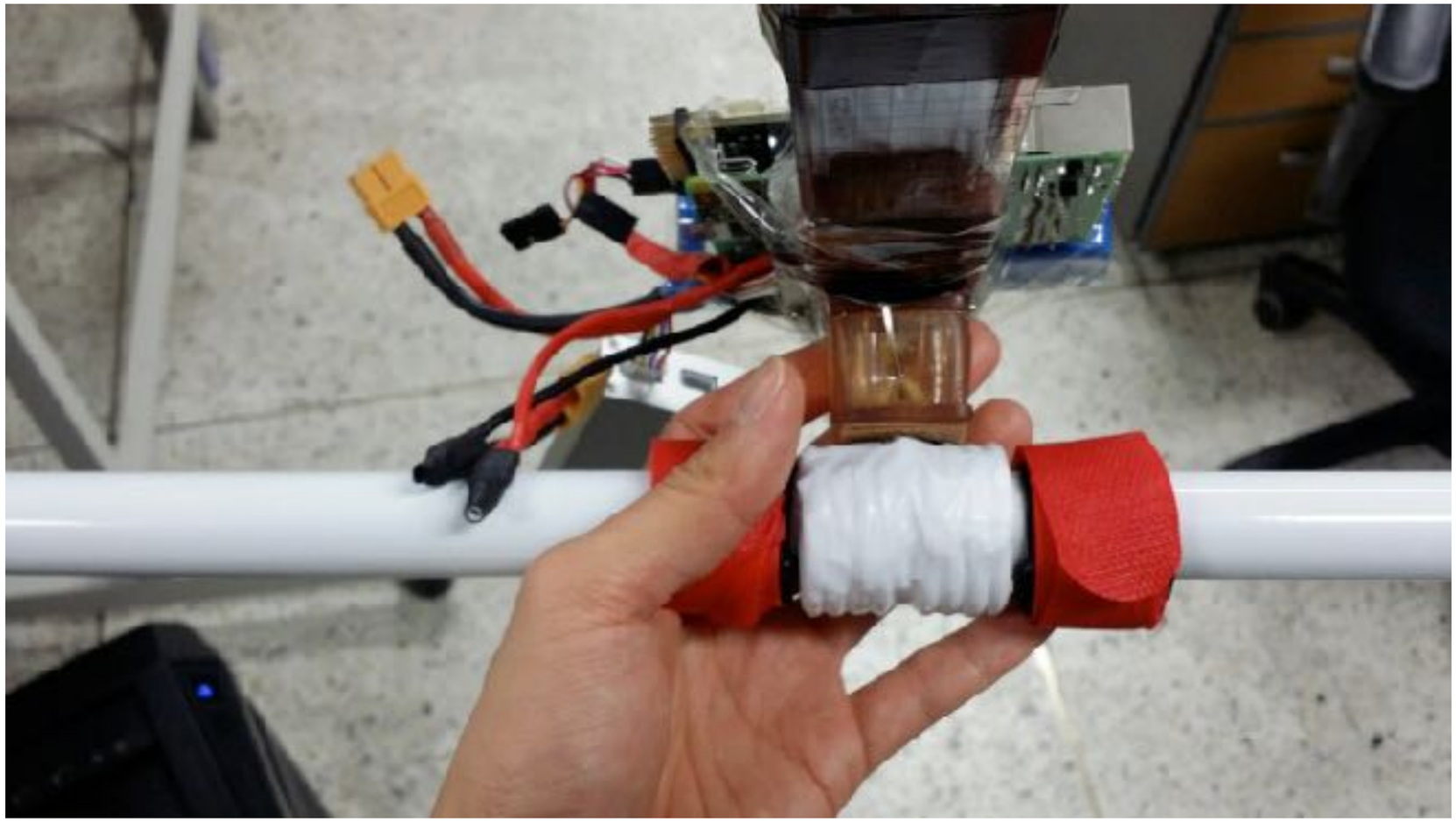
라즈베리파이를 세워 보려고 했지만, 세우게 되면 자이로 센서와, 가속도계의 보정 작업이 어려워 질것 같아서 라즈베리파이를 최대한 가운데에 배치. 배터리 또한 무게로 인한 영향을 최소화 하기 위해 세워서 최대한 가운데에 배치.



지난 시간의 FEEDBACK

- 실험 오류를 줄여보기
- 문제를 List up 하기
- 실험한 결과와 시행착오를 발표자료에 적기

실험 오류를 줄여보기



<달라진 점>

1. 커튼고리를 이어 붙여서 빨래봉과 닿는 면적을 넓혔다.

2. 드론 모형이 시소운동을 하는 것 이외에 좌우로 심하게 흔들리던 것이 줄어들었다.

3. 고리의 폭이 넓어져서 모터를 끄고 손으로도 수평을 잡은채로 세울 수 있게 되었다. *(영상 참고)*

문제를 LIST UP 해보기

단계적 목표 : 모터를 가동시킨 상태로 프로토타입 모형의 수평잡기

<현재의 문제>

1. 어떻게 정확한 기울기를 얻어낼 것인가?

- 가속도센서를 이용하여 Pitch 값을 얻어낼 것이다.

- 예견되는 문제점?

  - 가속도센서 : pitch와 roll값만 구할 수 있다.

    => 가속도센서가 회전 중심축에 연결되어야하고, 그 회전중심축은 제자리에 가만히 있을때를 제외하고는 가속도 센서만으로 정확한 측정이 불가능하다고 함.

  - 자이로센서 : pitch, roll, yaw값을 구할 수 있다.

    => 각속도 적분시 오차가 누적되면서 에러가 심해질 우려가 있다.

    (하지만, 움직이는 짧은 순간의 관점에서 보면 올바른 값을 나타냄)

    => 온도에 따라 그 값이 변하는 특성이 있다.

  - 지자기센서 (마그네틱센서) : yaw값을 구할 수 있다.

  - 칼만필터, 상보필터 등으로 진동으로 인해 튀는 값을 잡아주어야 할 것이다.

- 가속도센서 + 자이로센서 + 필터를 이용하여 정확한 기울기를 얻을 수 있다.

단계적 목표 : 모터를 가동시킨 상태로 프로토타입 모형의 수평잡기

<현재의 문제>

2. 얻은 기울기를 바탕으로 어떻게 조절할 것인가?

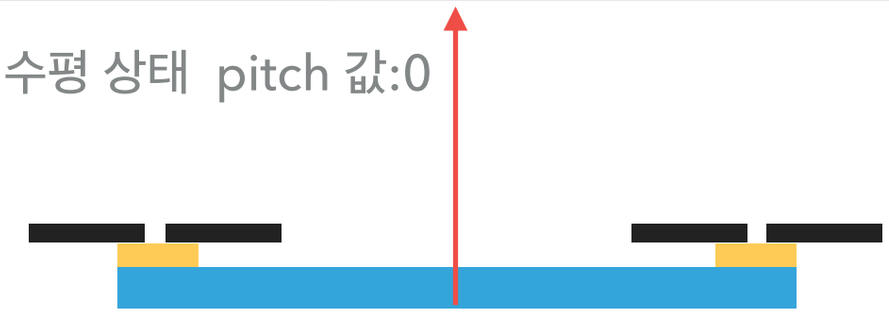
- 일단은 조건문을 통해 한쪽으로 기울면, 기운쪽 모터를 켜는 방식을 이용해보자.

```
# 왼쪽으로 기울었을 때
if (기울기 > 0 ) :
        왼쪽 모터 출력(1.1)
        오른쪽 모터 출력(1.0)          # 정지

# 오른쪽으로 기울었을 때
elif (기울기 < 0 ) :
        왼쪽 모터 출력(1.0)          # 정지
        오른쪽 모터 출력(1.22)

else :
        왼쪽 모터 출력(1.0)
        오른쪽 모터 출력(1.0)
```

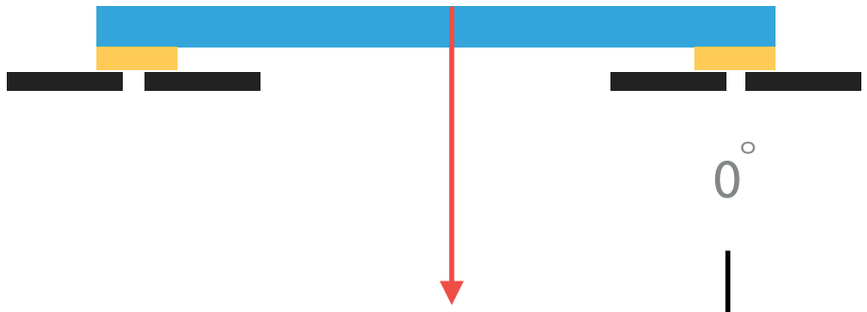
- 실험 결과영상

실험 결과 및 문제점

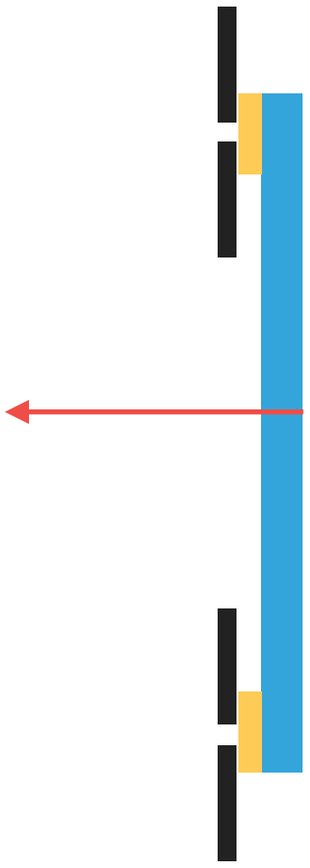
수평 상태 pitch 값:0



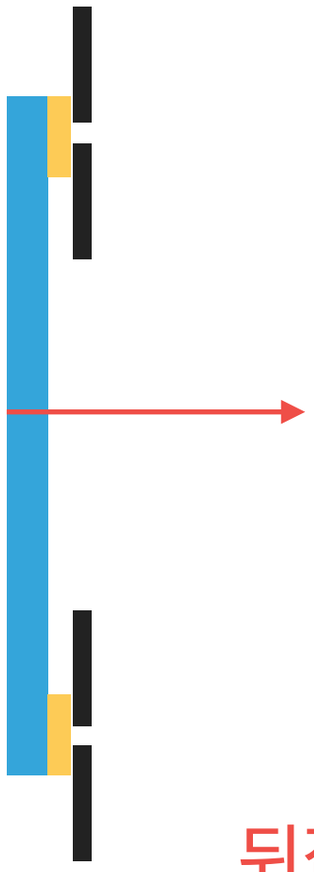
뒤집힌 수평 상태 pitch 값:0

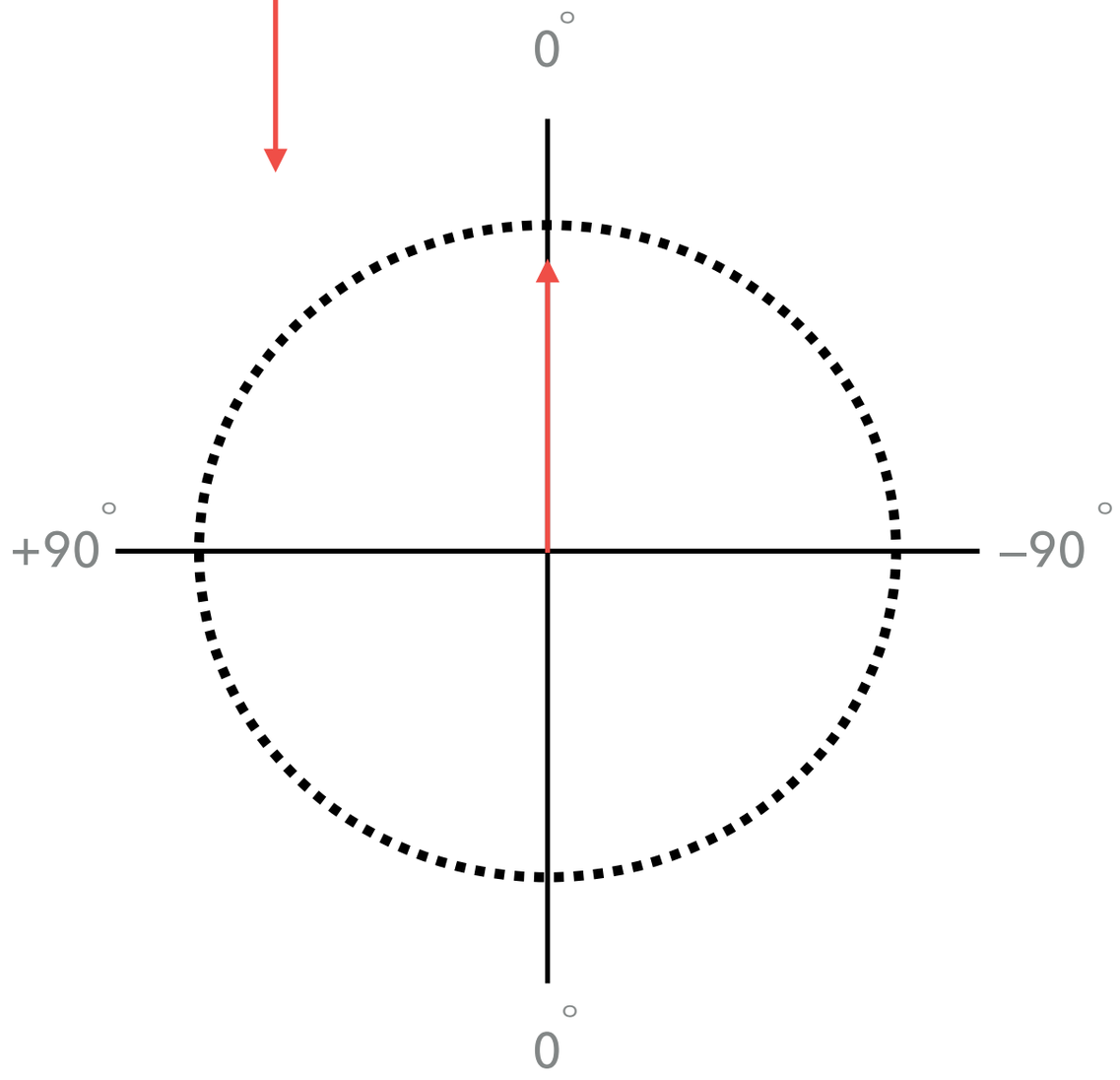


수직 상태 pitch 값:+90



수직 상태 pitch 값:-90





뒤집어진 기체와 올바르게 있는 기체의 구분이 되지 않는 문제 발생 !

실험 결과 및 문제점

<Drone 1 실험>

1. 실험오류를 줄이고 나서 모터를 가동한 상태로 수평을 유지하는 것이 가능해졌다. (손으로 중심잡기)
2. 뒤집어진 기체로 모터세기의 비율을 찾는건 실험결과로 봤을때 불가능하다는 판단을 함.
3. 모터에 같은 값을 주어도 다른 출력이 발생한다는 것을 확인함. (왼쪽 모터가 더 세다)
4. 대략 모터세기 L : 1.1 // R : 1.2215 ~ 1.2216 에서 수평값이 발생할 것으로 예상됨.

(문제점)3. 모터세기 L : 1.1 // R : 1.23일때 처음과 두번째가 다른 결과가 나옴.
(문제점)4. 아직 모터의 정확한 출력 비율을 찾지 못함.
(문제점)5. 리튬 폴리머 배터리가 죽어버려서 이후 Drone 2 실험에서는 기체의 모양, 무게등이 달라짐

<Drone 2 실험>

1. 바뀐 기체에서 대략 모터세기 L : 1.1 // R : 1.22 값으로 수평을 유지하는 것이 가능해짐. (손으로)

(문제점)2. 기체가 급격하게 움직이면 각도 값이 튀어서 짧은 순간이지만 엉뚱한 모터가 가동됨.
(문제점)3. 각도값이 (-)90 ~ (+)90 사이의 값만 존재해서 뒤집어진 기체와의 구별이 어려움.
(문제점)4. 기울기에 대한 반응을 하기 이전에 모터가 계속 움직이는 것으로 보임.
(문제점)5. 현재 계속 같은 세기로 모터를 움직이는 실험 2의 알고리즘으로는 수평제어가 불가능함.

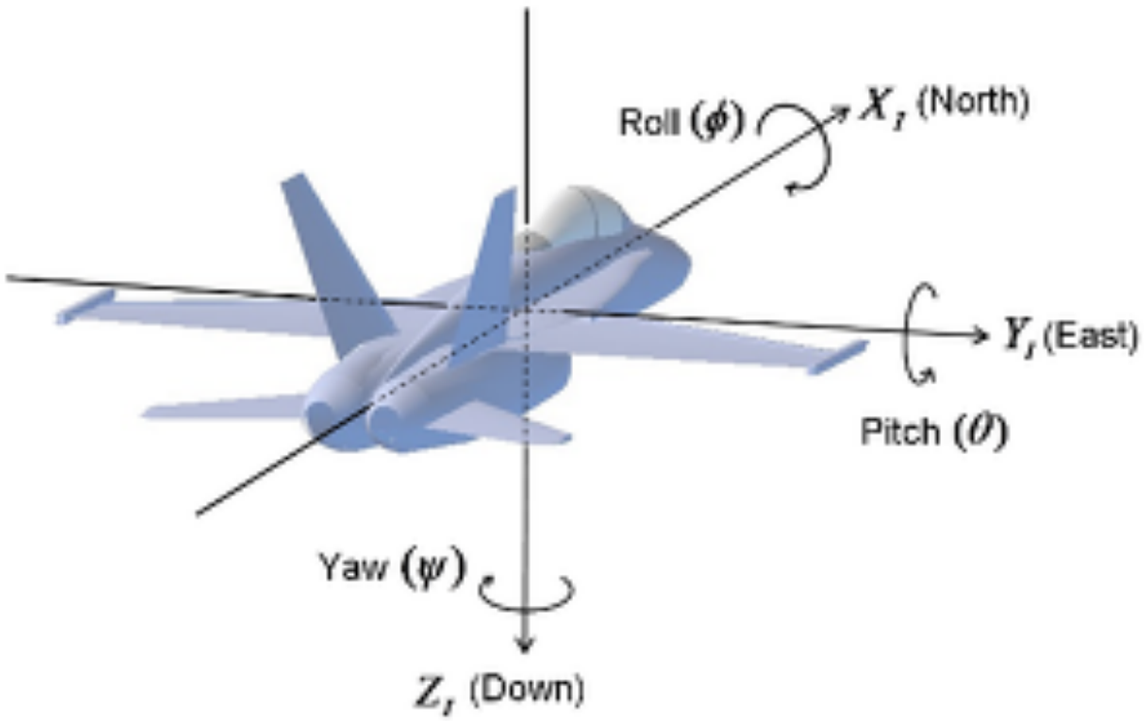
지난 시간의 FEEDBACK

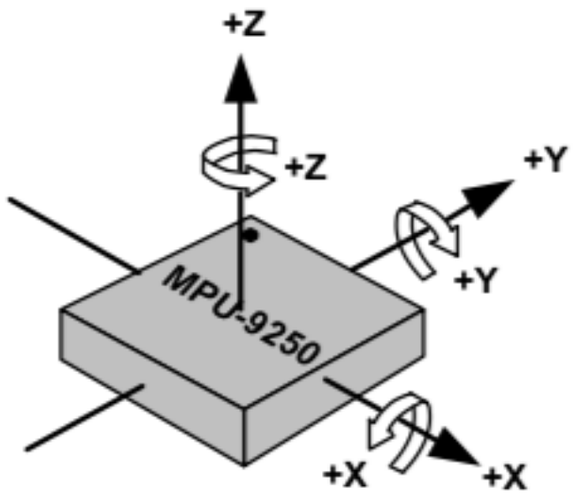
- 정말 가속도 센서만 가지고 각도를 판단할 수 있는가?

- 뒤집어서 원하는 각도를 맞춰보면 어떨까?

- 변화된 기울기를 감지하여 모터에 이전과 다른 값을 전달하고,
  모터가 변화된 값에 반응하여 기울기를 변화시키는데 걸리는 시간은 얼마나 걸릴까?

가속도 센서로 각도 구하기



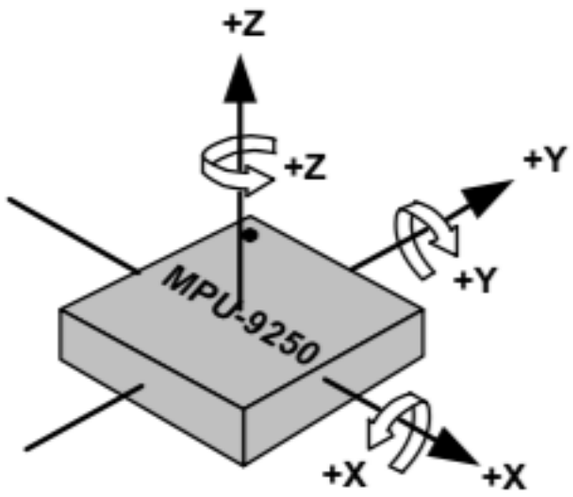


가속도/자이로 센서 방향

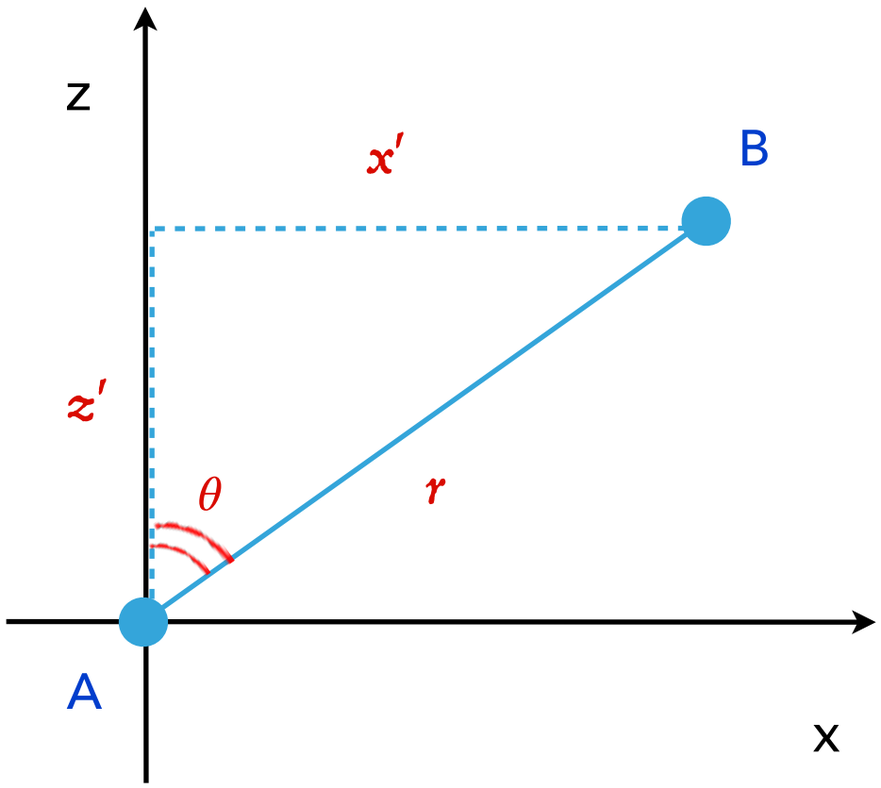
Pitch 값이란?

직선축 Y축을 기준으로 X, Z 평면이 얼마나 회전했는지 나타내는 값이다.

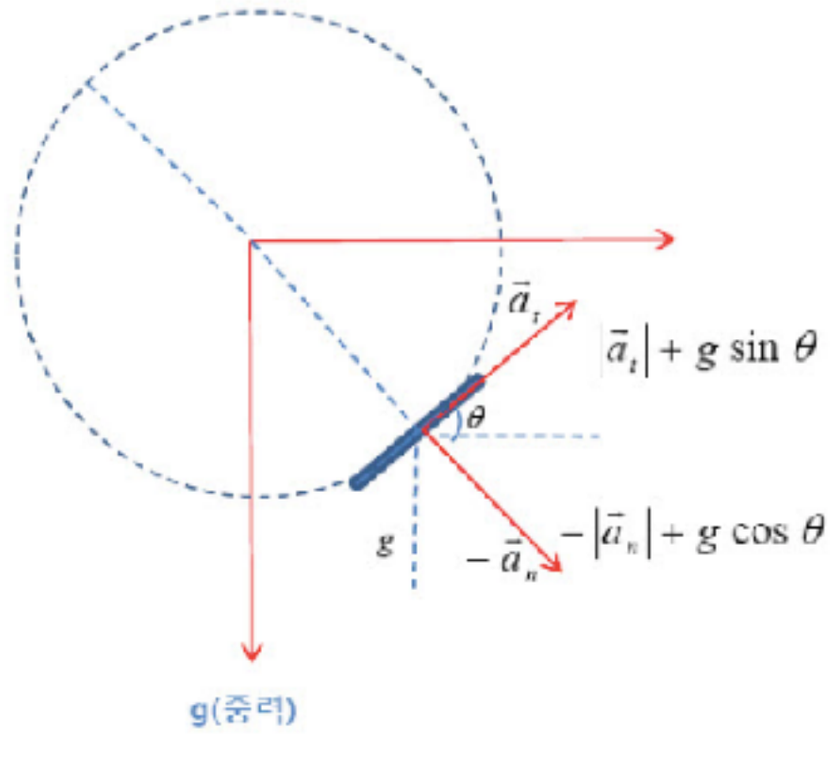
가속도 센서로 각도 구하기



가속도/자이로 센서 방향



참고)



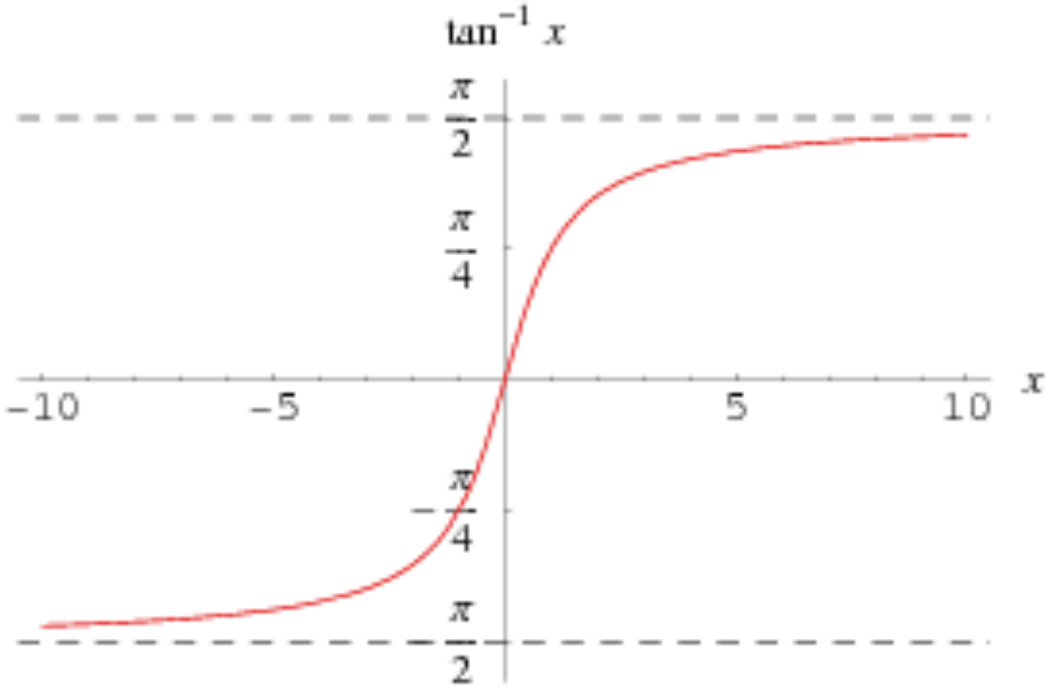
그렇다면 X, Z 평면상의 두 점 사이의 각도 θ 를 구해보자

(단, 한 점은 원점에 존재)

알고 있는 값 x' 와 z' 를 이용하면 각도를 구할 수 있다.

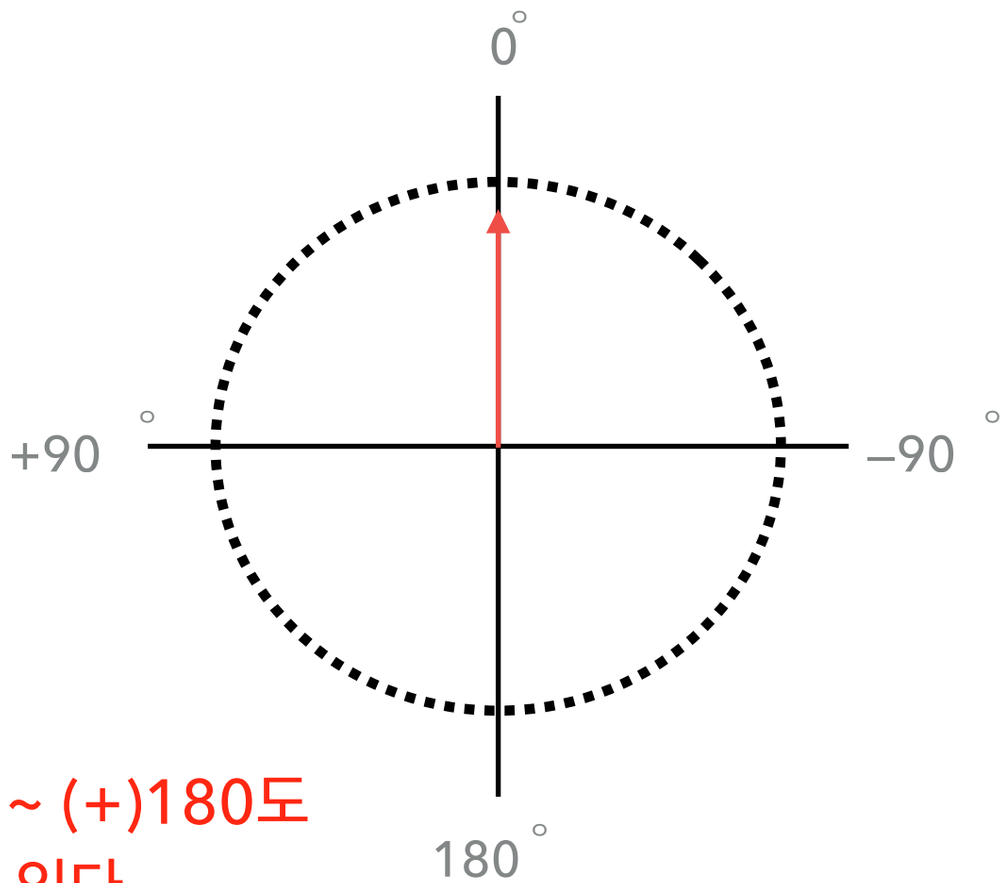
$$\tan\theta = \frac{x'}{z'} \xrightarrow{\text{tan의 역삼각 함수 이용}} \theta = \tan^{-1}\frac{x'}{z'}$$

가속도 센서로 각도 구하기



arctan를 이용하면 (-)90도 ~ (+)90도
사이의 값만 얻을 수 있다.

$$\text{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$



이를 보정한
atan2를 이용하면 (-)180도 ~ (+)180도
사이의 값들을 얻을 수 있다.

가속도 센서로 각도 구하기


```
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 7.2817290605
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 24.3623132812
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = -7.57904121591
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 17.1089013604
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 56.8317233372
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = -4.92100814561
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 25.2118358043
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = -4.00288295317
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 8.88619221168
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 26.5088724622
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 3.12201590755
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 34.6610368361
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = -3.56889093962
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 14.7381241145
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 18.8783410635
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = -2.589819667
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 33.5515792293
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = -0.379249185181
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 30.9466721629
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 12.4616861065
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 6.92884735039
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 23.367659112
pwm_v1 = 1.12 pwm_v2 = 1.22 degree = 4.2368459515
```

그러나 모터를 돌린 뒤 가속도 센서만으로

각도를 측정하면

의미없는 값들만을 얻을 수 있다.

모터로 인한 심한 진동으로 인해 손으로 잡고 드론의 모터를 돌렸을 때와
또 다른 결과 발생

가속도 센서 이외의 방법으로 각도 보정 필요

가속도 센서로 각도 구하기

AHRS.cpp ← 기존에 보드 제조사에서 제공한 각도 코드

↓ SWIG, BOOST로 변환 시도

코드의 복잡성으로 인해 실패

↓ PIPE 통신 시도

AHRS.cpp과 파이프 통신으로 각도값 구하기

가속도 센서로 각도 구하기

python 프로세스 관련 api 호출

```
import subprocess
```

python pipe 정의

```
proc = subprocess.Popen("/home/pi/ahrs/Navio2/C++/Examples/AHRS/./AHRS",
stdin=subprocess.PIPE, stdout=subprocess.PIPE)
```

python pipe 통신

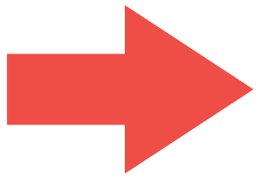
```
proc.stdin.write("get" + "\n")
pitch_v = proc.stdout.readline().rstrip("\n")
```

C++

```
cin >> python_message;
cout << pitch << endl;
```

뒤집어서 실험해보기

각도별로
다른 출력이 필요함



출력별 각도를
알아보려고 함

<실험 영상>

1. 동시에 초기값(L : 1.1 // R : 1.22)으로 모터를 켜고 5초에 한번씩 왼쪽 모터만 0.01씩 값을 증가시키며 각도 관찰
2. 동시에 초기값(L : 1.1 // R : 1.22)으로 모터를 켜고 5초에 한번씩 오른쪽 모터만 0.01씩 값을 증가시키며 각도 관찰
3. 오른쪽 모터에는 출력(1.0)으로 모터를 정지하고 왼쪽 모터는 초기값(1.1)으로 모터를 켜고 5초에 한번씩 왼쪽모터만 0.01 증가 시키며 각도 관찰
4. 왼쪽 모터에는 출력(1.0)으로 모터를 정지하고 오른쪽 모터는 초기값(1.22)으로 모터를 켜고 5초에 한번씩 오른쪽 모터만 0.01 증가 시키며 각도 관찰

<결과>

1. 실험1 최대 모터 값 (1.51), 실험2 최대 모터 값(1.57), 실험3 최대 모터 값(1.39), 실험4 최대 모터 값(1.47)

<문제점>

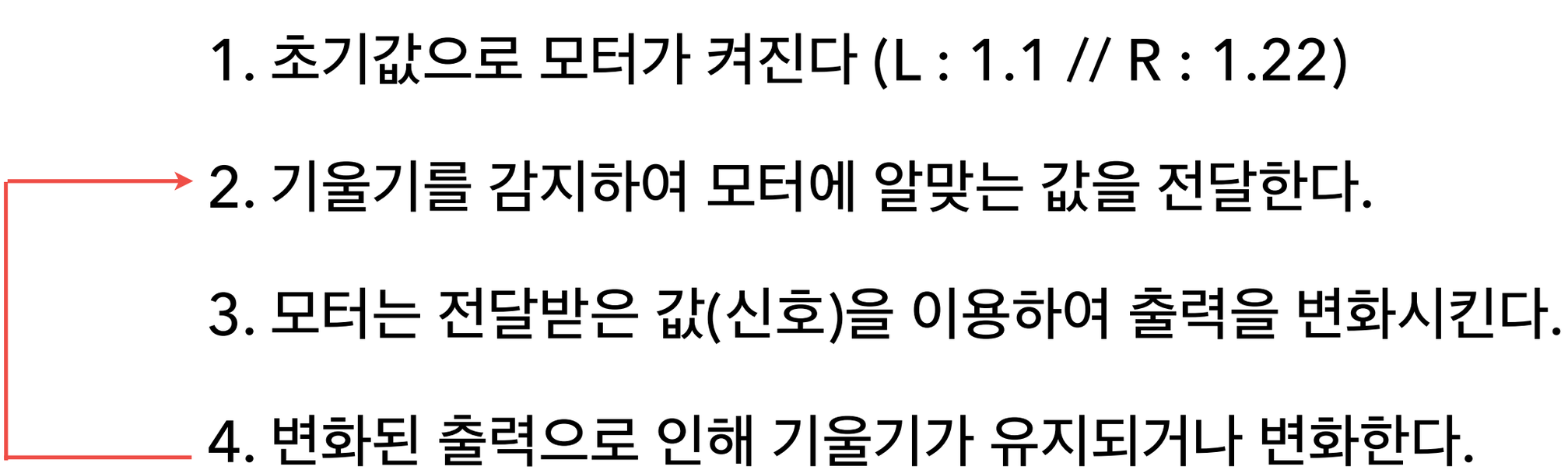
1. 올바른 자세에서 실험했을 때의 출력별 각도와 위 실험에서의 출력별 각도가 다르다.
2. Navio Board 제조사에서 제공하는 각도 역시 (–)90 ~ (+)90 사이의 값이다. (실제로는 (–)84 ~ (+)84범위)
3. 측정되는 Pitch값이 너무 많아서 출력별 각도를 딱 정의하기 힘들다.

제어이론

이전에 알고리즘을 적용한 실험에서 모터가 제대로 제어되지 않았던 이유는

제어시간과의 관련성 보다는 가속도센서만을 이용한 각도 측정의 문제로 확인하였다.

그렇지만 제어과정에 대하여 좀 더 생각해 보기로 했다.



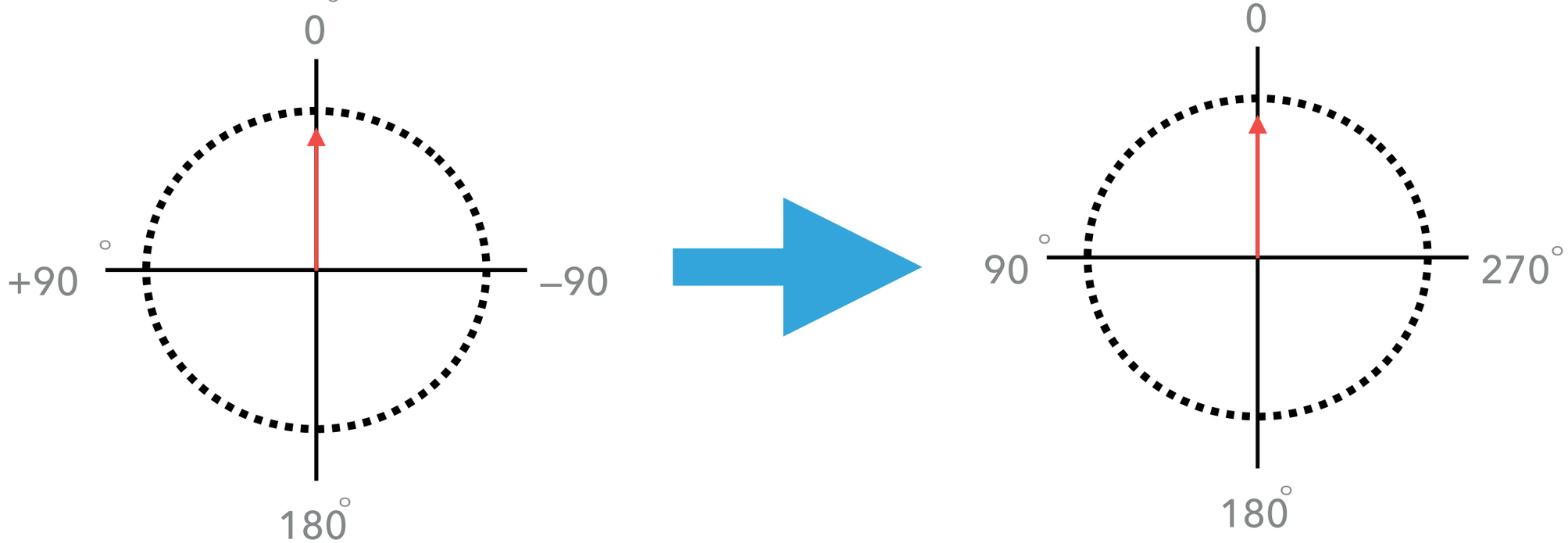
5월 10일 이후 진행 예정 사항

1. 가속도계의 측정 속도 체크 (1초에 몇 번 센서 값을 읽어들이는지)

   평균값 필터의 window 사이즈를 0.01초 이하의 delay가 발생하도록 조절

2. 센서를 통해 얻은 각도의 범위를 (-)180도 ~ (+)180도에서 0도 ~ (+)360도의 범위로 수정

   Degree = (X + 360) % 360 변환식 이용



5월 10일 이후 진행 예정 사항

3. 변화된 측정 각도를 통하여 기본적인 제어코드를 돌려보기

```
# 왼쪽으로 기울었을 때
        if (기울기 < 180 ) :
            왼쪽 모터 출력(1.1)
            오른쪽 모터 출력(1.0)          # 정지

        # 오른쪽으로 기울었을 때
        elif (기울기 > 180) :
            왼쪽 모터 출력(1.0)          # 정지
            오른쪽 모터 출력(1.22)

else :
        왼쪽 모터 출력(1.0)
        오른쪽 모터 출력(1.0)
```

5월 10일 이후 진행 예정 사항

4. 모터의 출력값을 각도에 비례하여 출력되게 수정하기

왼쪽 모터의 기본값 : 1.1
왼쪽 모터의 최대값(뒤집힘 값) : 1.51

➔ 값의 차 : 0.50
➔ 180 단계의 구간을 1.1에서 1.51까지의 값의 변화를 지수배로 증가시키기

$$x^{180} = 0.50$$

$$\sqrt[180]{x^{180}} = \sqrt[180]{0.50}$$

$$x = \sqrt[180]{0.50}$$

➔ output = 1.1 + pow(0.47, degree)

오른쪽 모터의 기본값 : 1.22
오른쪽 모터의 최대값(뒤집힘 값) : 1.57

➔ 값의 차 : 0.35
➔ 180 단계의 구간을 1.22에서 1.57까지의 값의 변화를 지수배로 증가시키기

$$x^{180} = 0.35$$

$$\sqrt[180]{x^{180}} = \sqrt[180]{0.35}$$

$$x = \sqrt[180]{0.35}$$

➔ output = 1.22 + pow(0.35, (degree-180))