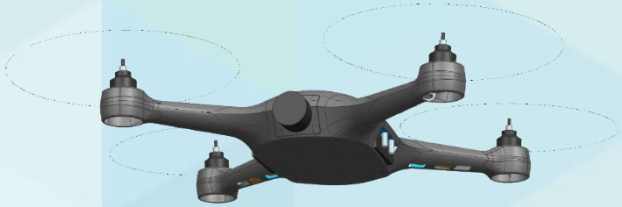


실내/외 자율 군집 비행 및 오픈소스 기반의 비행 제어 시스템 생태계 소개

한국항공우주연구원
융합기술연구본부 미래항공우주기술팀
문성태

안녕하세요.
저는 한국항공우주연구원 문성태 연구원입니다..

안녕하세요.
저는 한국항공우주연구원 문성태 연구원입니다..

안녕하세요.
저는 한국항공우주연구원 문성태 연구원입니다..

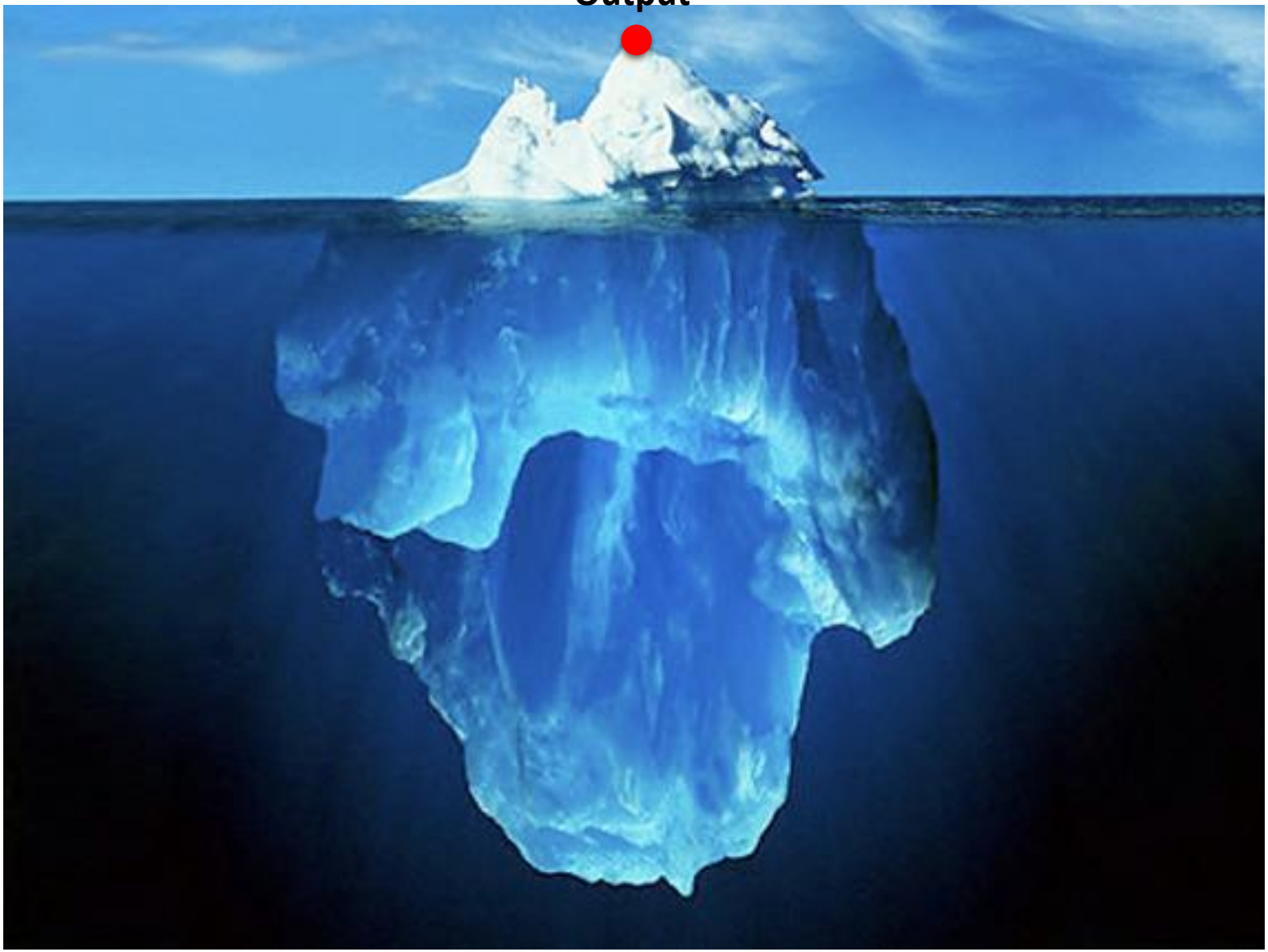
안녕하세요.
저는 한국항공우주연구원 문성태 연구원입니다..

~~Aerospace Engineering~~

Computer Science

Programming 

Open Source 


Output

그래서, 오늘은

오픈소스를 활용하여 개발한

좌충우돌 군집 비행 연구

에 대해 이야기 해볼까 합니다.

CONTENTS

실내/외 자율 군집 비행 및 오픈소스 기반의
비행 제어 시스템 생태계 소개

KOREA AEROSPACE RESEARCH INSTITUTE

I. 실내 군집 비행

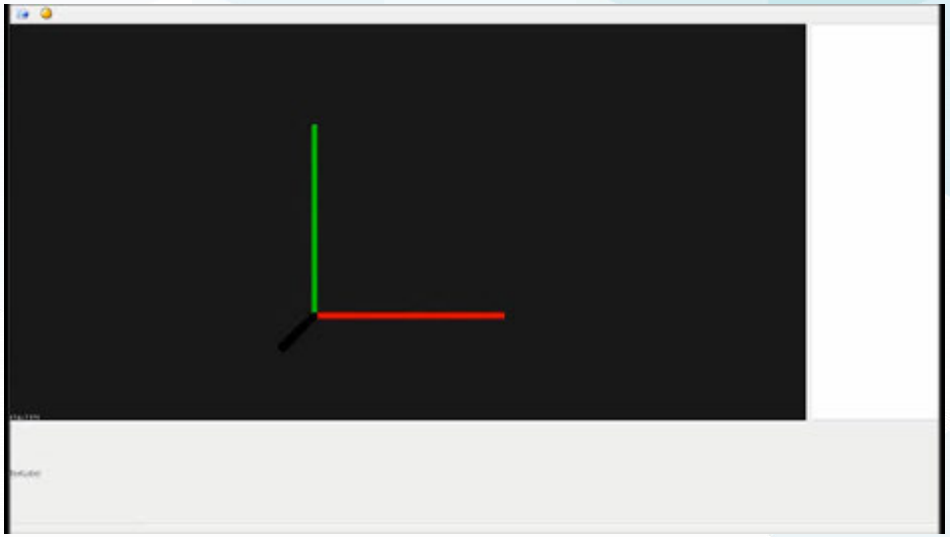
KOREA AEROSPACE RESEARCH INSTITUTE

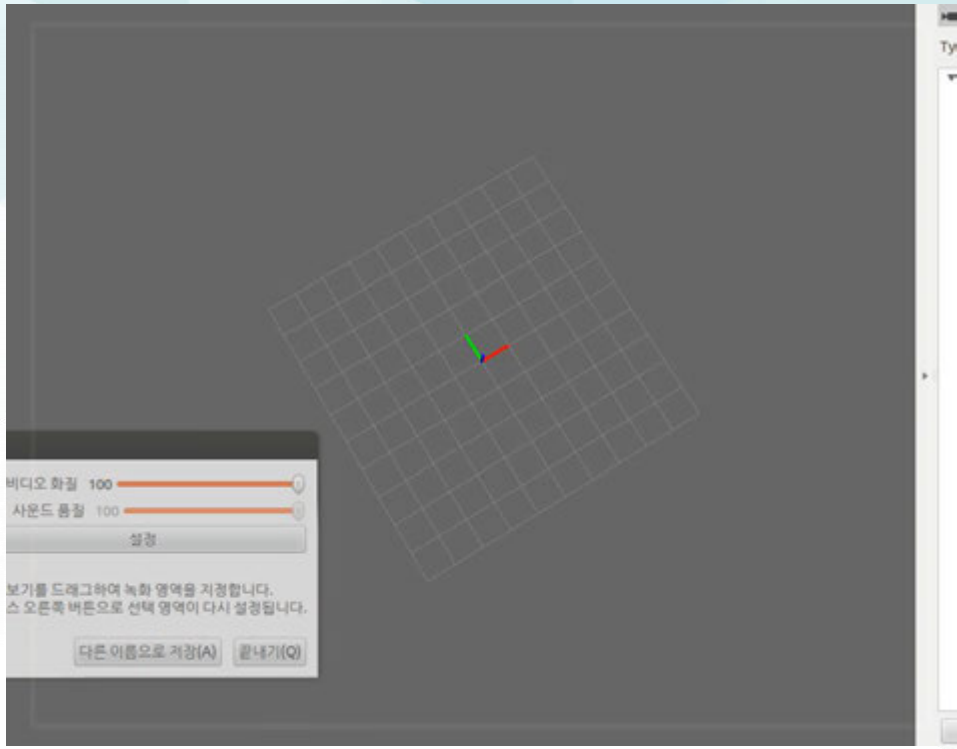
KARI 한국항공우주연구원 Korea Aerospace Research Institute

ADEX 2013

처음은 이게 아니었다

SLAM




비디오 화질 100
사운드 품질 100
설정
보기를 드래그하여 녹화 영역을 지정합니다.
스 오른쪽 버튼으로 선택 영역이 다시 설정됩니다.
다른 이름으로 저장(A)
끝내기(Q)

시작은 단순했다 …

시작은 단순했다

동시에 여러 비행체를 날릴 수 있을까?

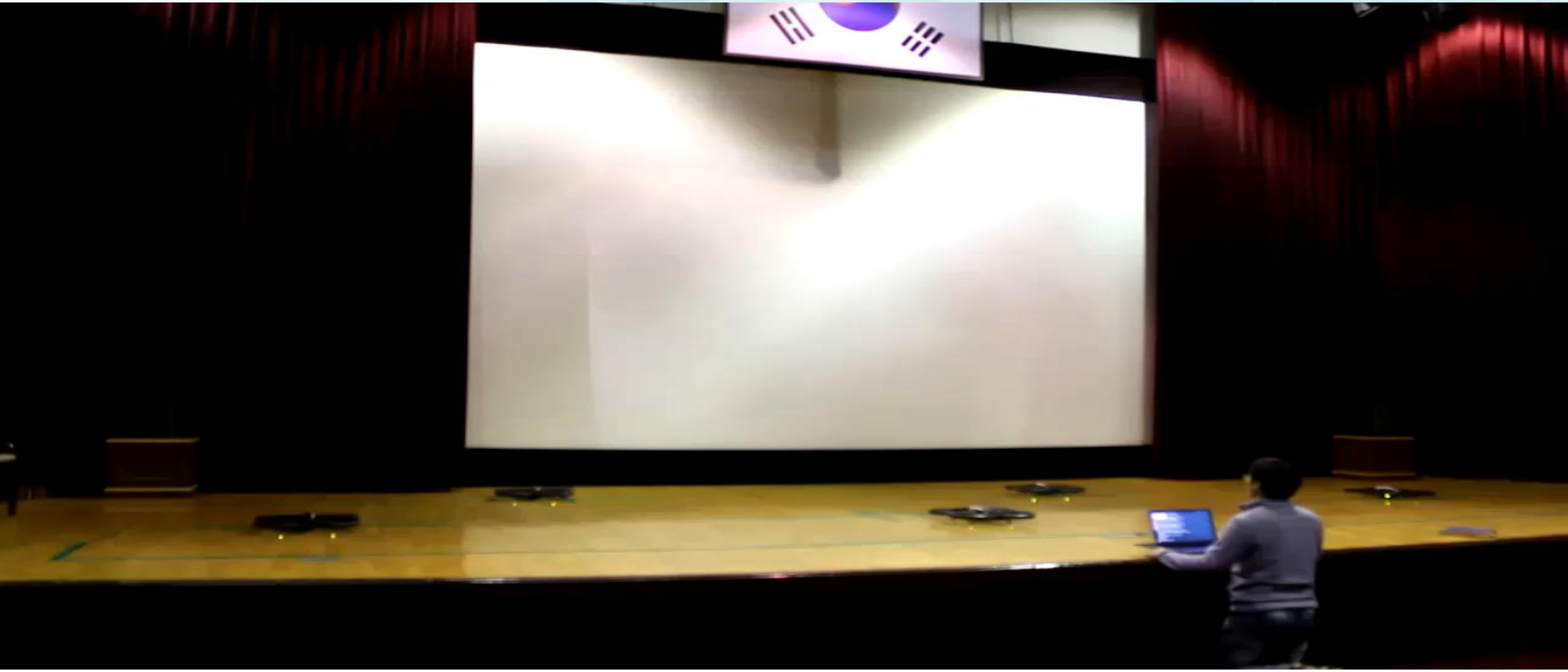
1. Linux 기반 시스템
2. WiFi 통신
3. SDK 제공
4. 착한 가격

시작은 단순했다

Server/Client 구조를 바꾸자

iwconfig ath0 mode managed essid ardrone_ap
ifconfig ath0 xx.xx.xx.xx netmask 255.255.255.0 up
route add default gw xx.xx.xx.1

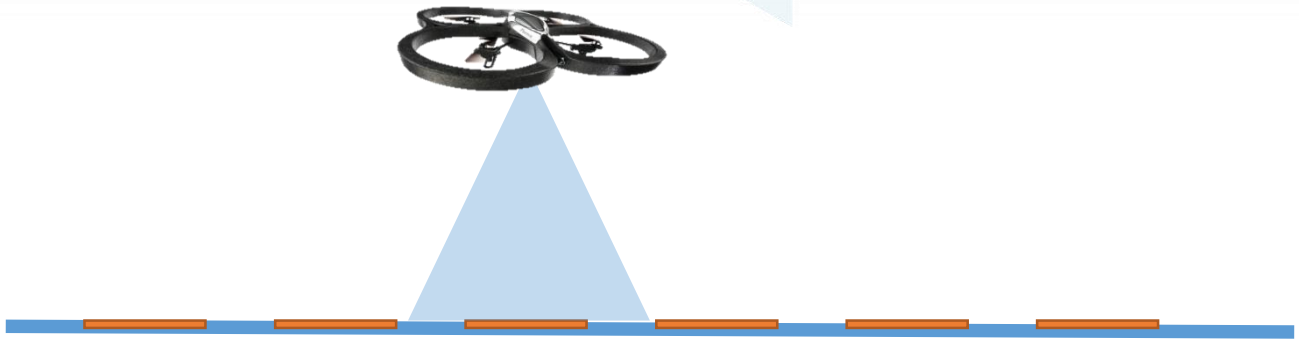
시작은 단순했다



자율 비행을 시키자

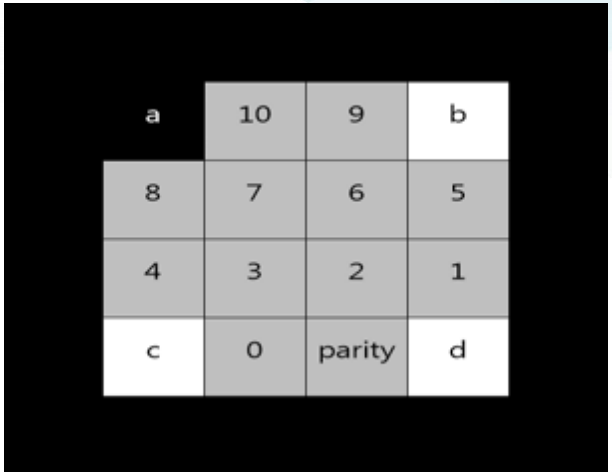
기본 아이디어

바닥에 패턴을 그리고 인식 시켜 위치를 확인하자



기본 아이디어

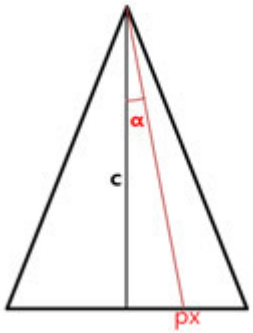
- 외곽선을 탐지하여 마커 인식 (using OpenCV)
- marker consists of
  - for direction, fixed sub blocks (a,b,c,d)
    - a block has to be always black
    - b, c, and d blocks are white
  - for marker ID, 0~10 blocks
    - can create 2048 markers
  - parity bit



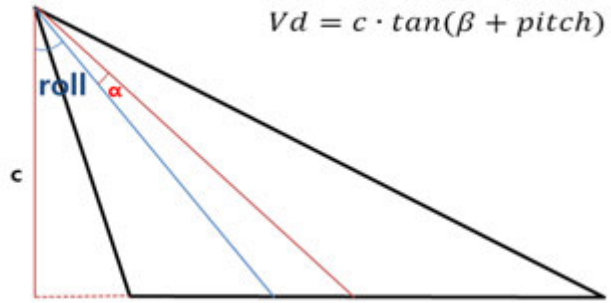
위치 인식 알고리즘

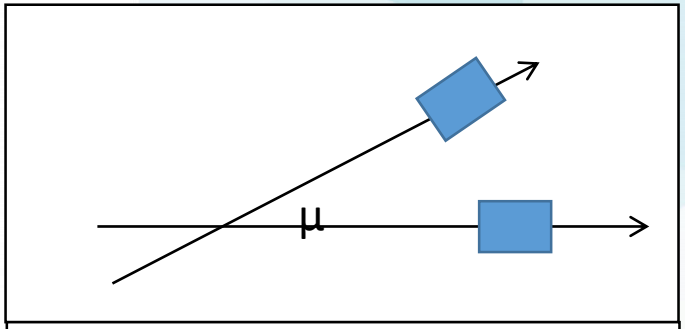
- we know
  - each marker position from real measurement
  - apertures of the camera（64°）from AR.Drone spec
  - altitude from SONAR sensor
  - roll/pitch/yaw from IMU
- We calculate position

  considering roll/pitch/yaw



$$Hd = c \cdot \tan(\alpha + roll)$$
$$Vd = c \cdot tan(\beta + pitch)$$





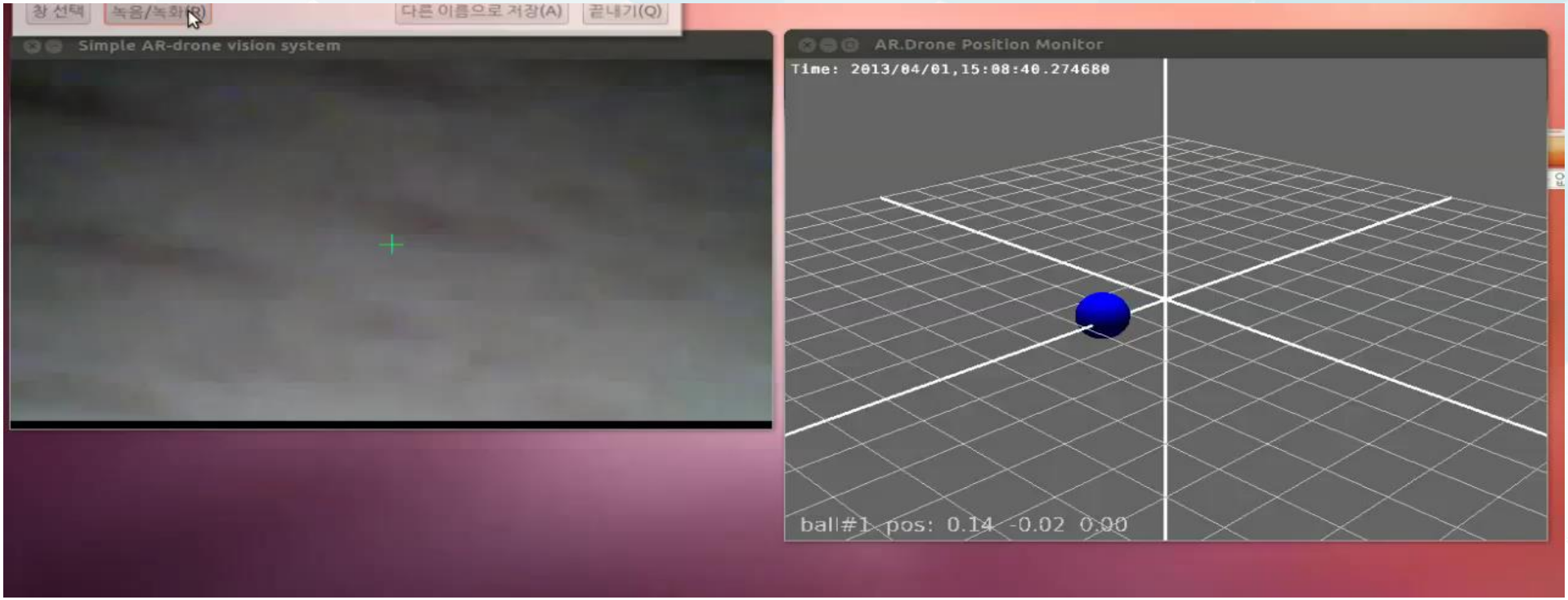
$$\theta = \arctan\left(\frac{Vd}{Hd}\right) + \mu$$

$$DeltaX = \sqrt{Hd^2 + Vd^2} \cdot \cos\theta$$

$$DeltaY = \sqrt{Hd^2 + Vd^2} \cdot \sin\theta$$

$$X = PatternX - DeltaX$$
$$Y = PatternY - DeltaY$$

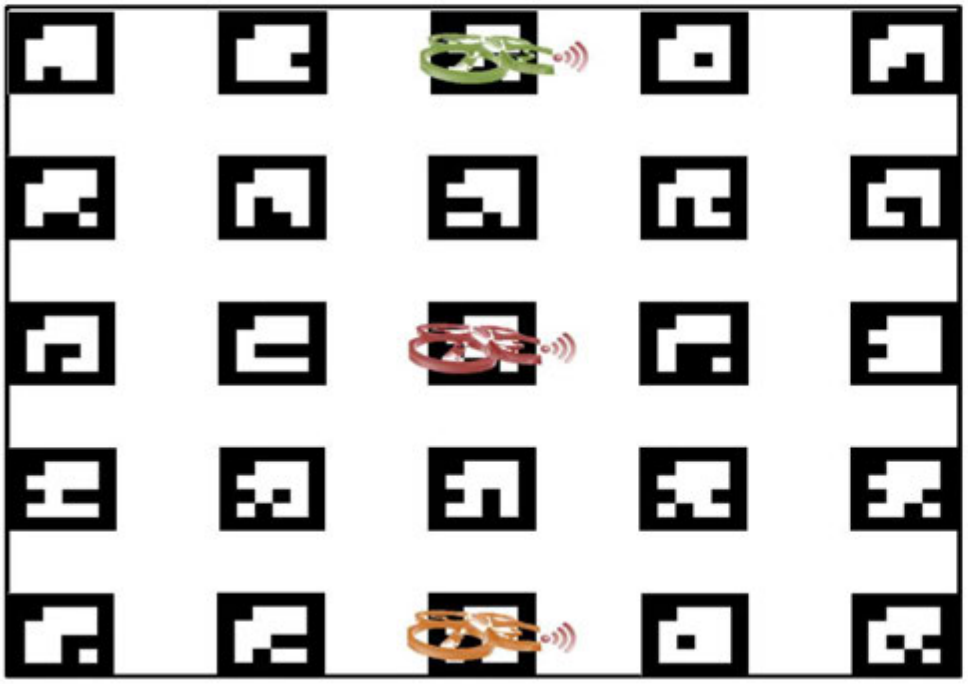
위치인식 알고리즘

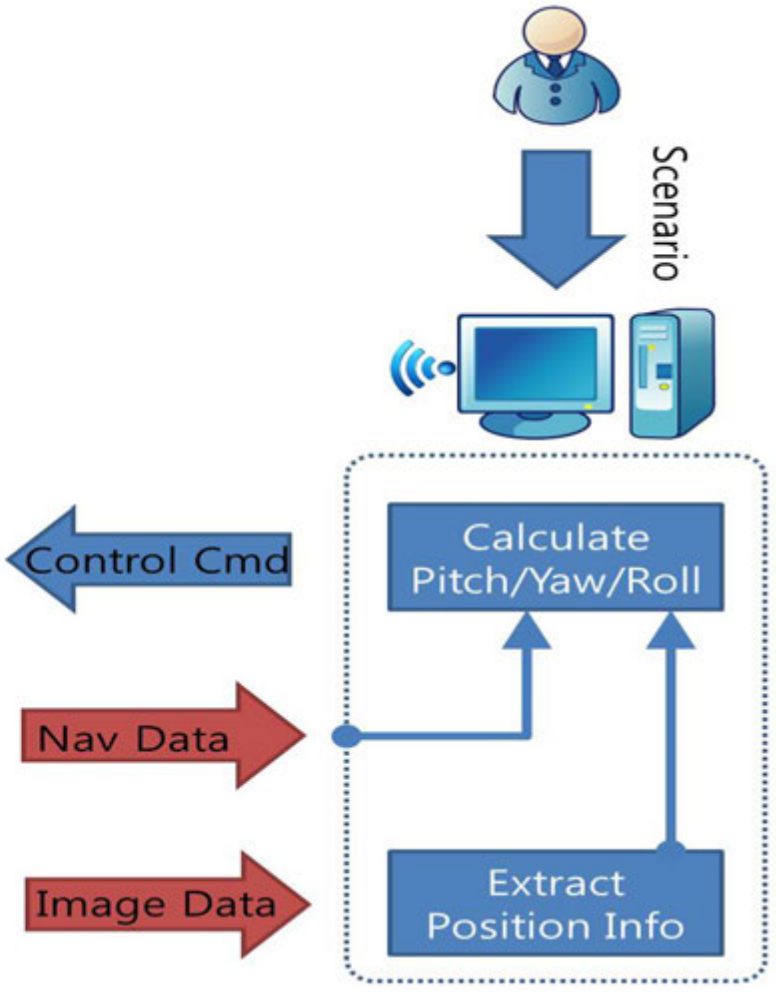

창 선택
녹음/녹화(R)
다른 이름으로 저장(A)
끝내기(Q)
Simple AR-drone vision system
AR.Drone Position Monitor
Time: 2013/04/01,15:08:40.274680
ball#1 pos: 0.14 -0.02 0.00

자율비행시험

다수 비행체를 동시에..

시스템 구조

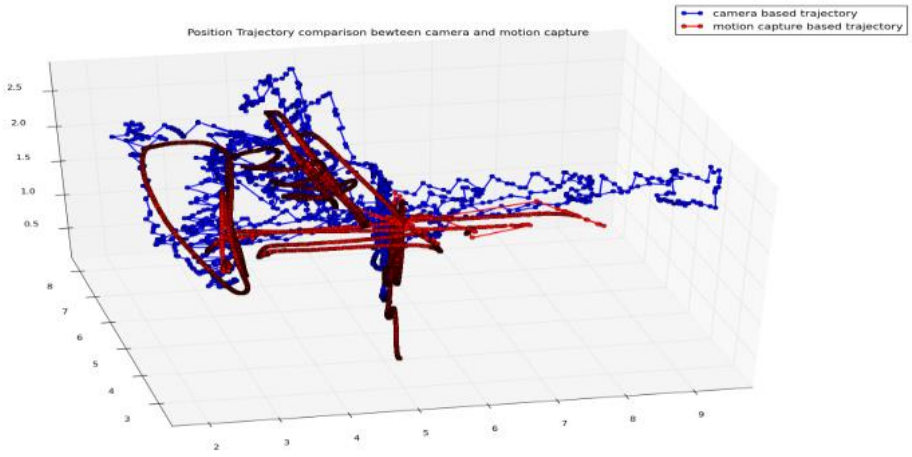


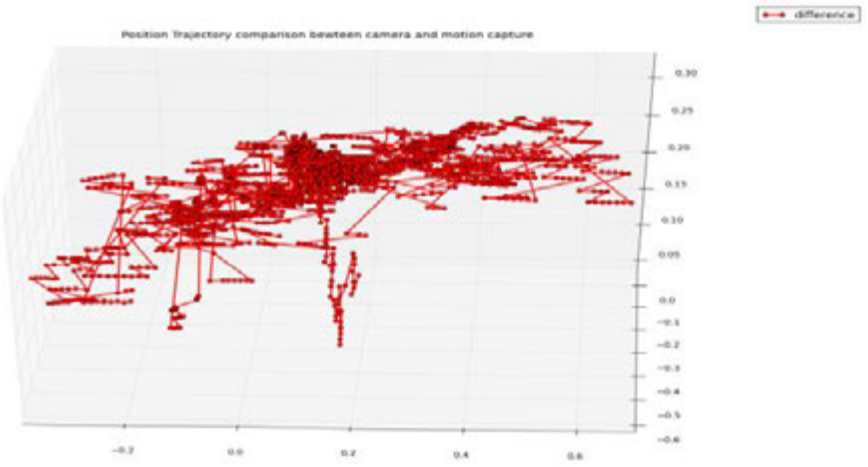

Scenario
Control Cmd
Nav Data
Image Data
Calculate
Pitch/Yaw/Roll
Extract
Position Info

군집 비행 시험

얼마나 정확할까…

- For accuracy test, use motion capture system
- maximum error is 0.6m when altitude is 1.8m



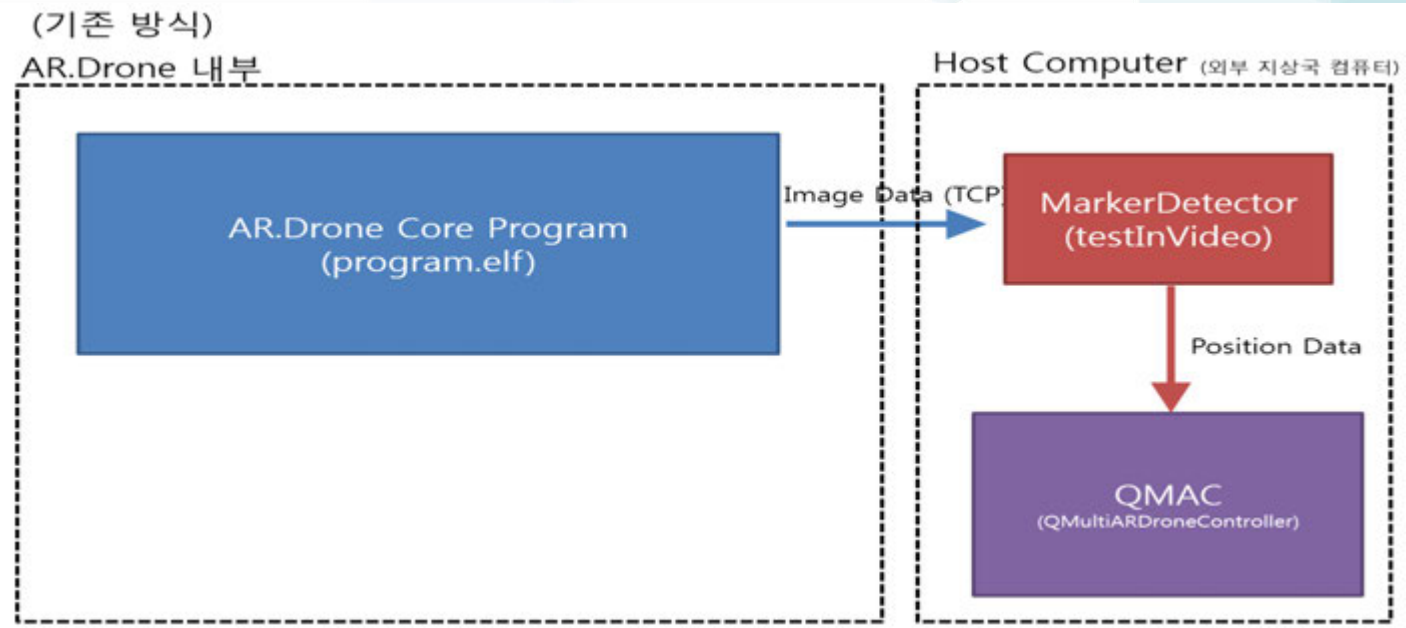


좀더 많이 해보자 그리고 한계...

영상 인식의 한계

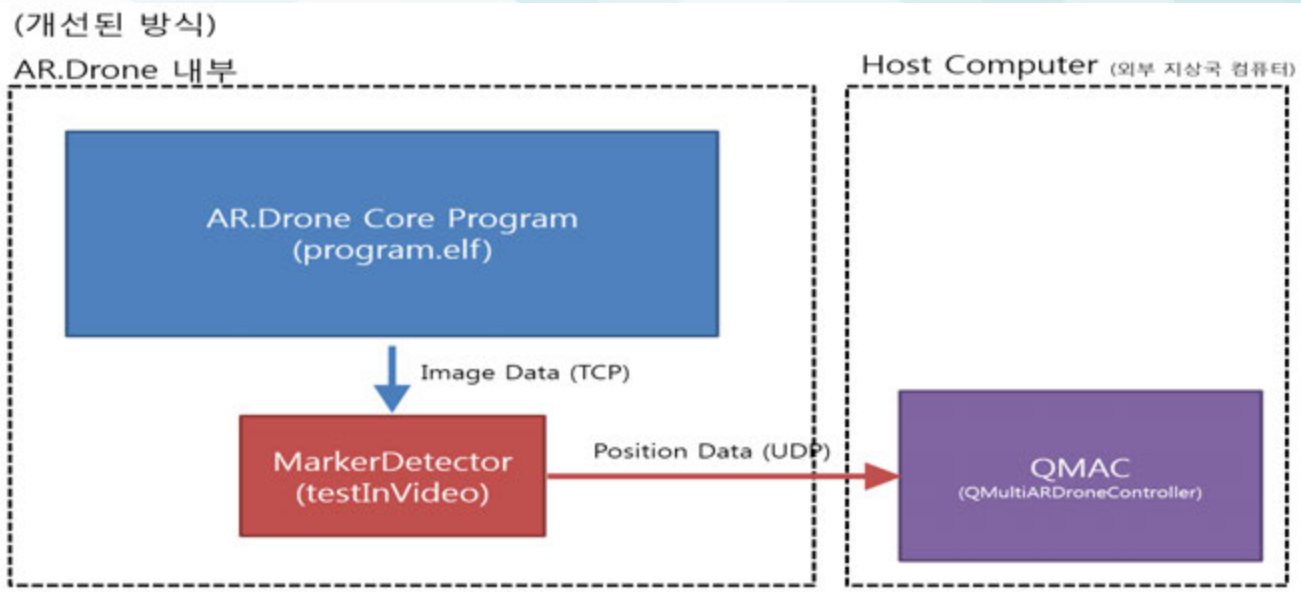
◆ 영상 처리의 부하

◆서버로 영상 받아서 처리하다 보니 8대 이상 처리 불가능



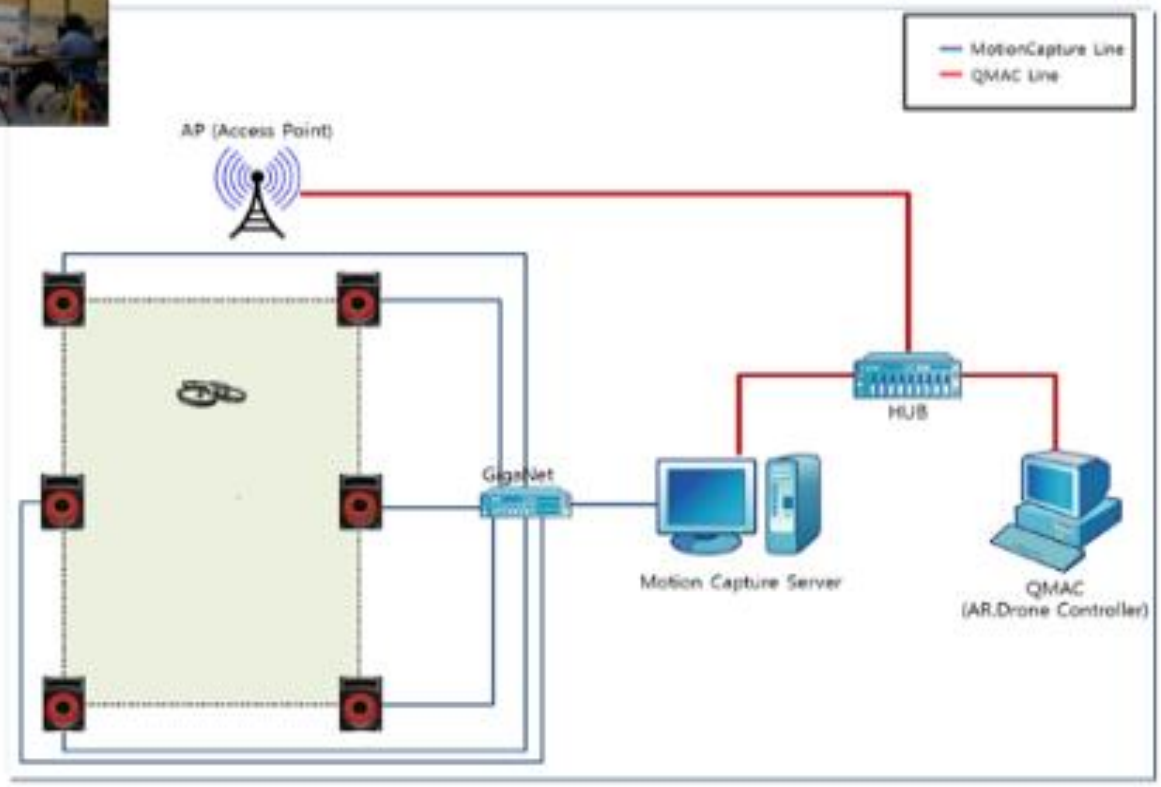
영상 인식의 한계

◆AR.Drone 내부에서 처리해보자



처리 방식	성능
Client-Server(기존 방식)	40Hz (25msec)
Distributed (기존 방식)	3.125 Hz (320msec)
Distributed (개선된 방식)	11.1Hz (90msec)

모션캡쳐를 사용해 보자

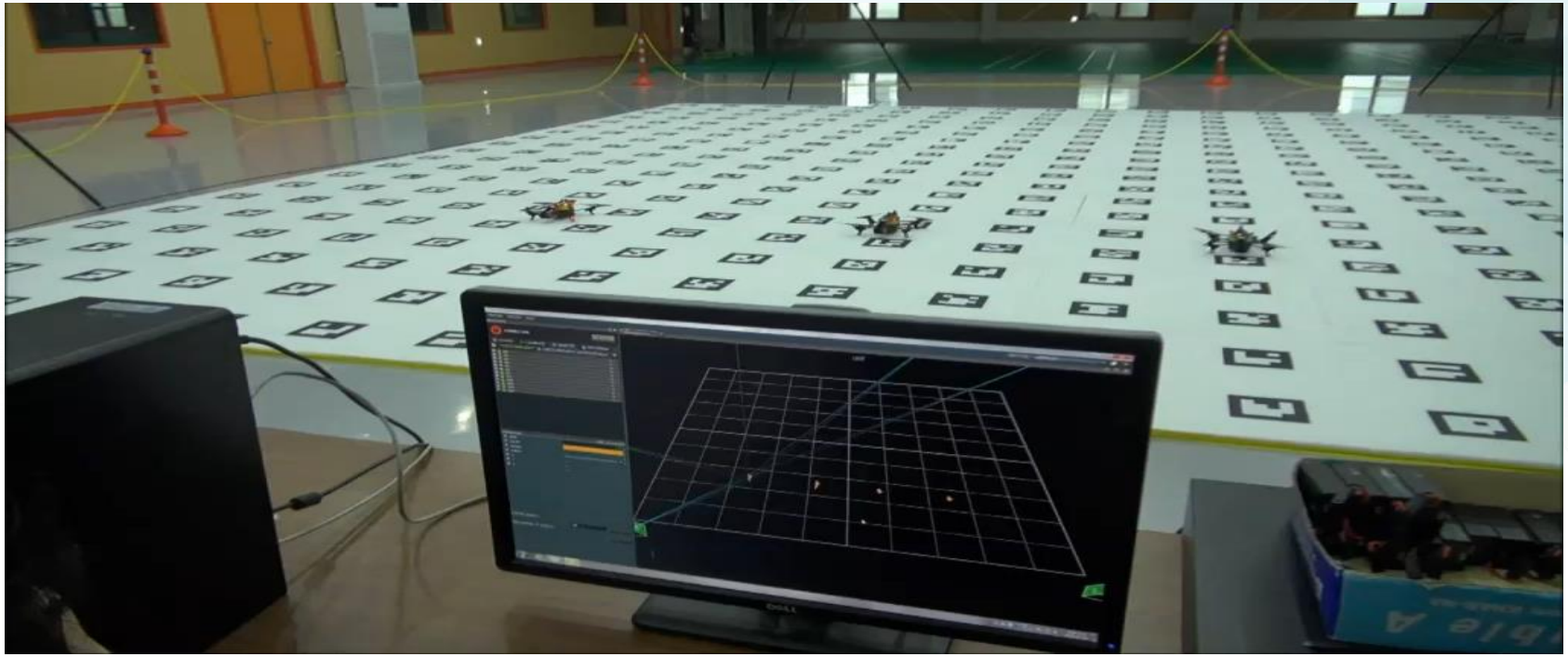

MotionCapture Line
QMAC Line
AP (Access Point)
HUB
GigaNet
Motion Capture Server
QMAC
(AR.Drone Controller)

모션캡쳐를 사용해보자

- 적외선 카메라를 이용하여 마커에 3차원 위치 인식
- 각 비행체에 마커의 패턴을 이용하여 비행체 인식
- 1mm 이하 오차로 측정 가능

모션캡쳐를 사용해보자

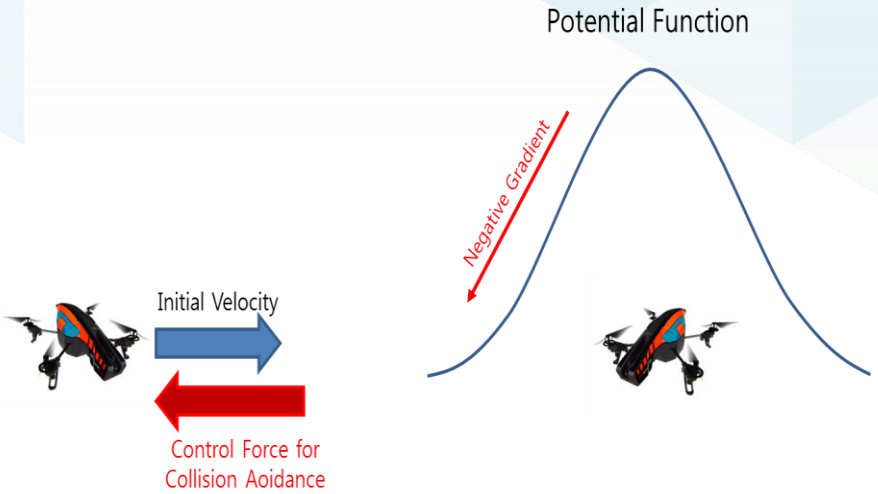
◆빠르고, 정확하다



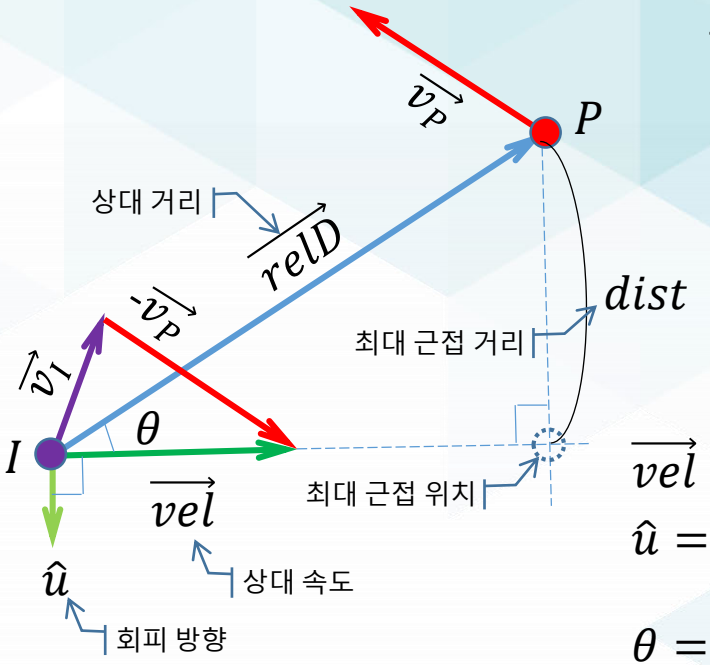
좀더 많이 해보자 그리고 한계...

충돌회피

- 드론이 많다보니, 서로간의 충돌 현상 발생
- IDEA: 어느 정도 가까이 있으면 서로 인식하고 피하자 (Potential Function)



충돌회피



'회피 방향'은 '상대 속도'의 직각 방향

<회피 조건 1>

$$\overrightarrow{relD} \cdot \overrightarrow{vel} > 0$$

'상대 거리'와 '상대 속도'가 같은 방향이어야 회피 함.

<회피 조건 2>

$$dist < 1$$

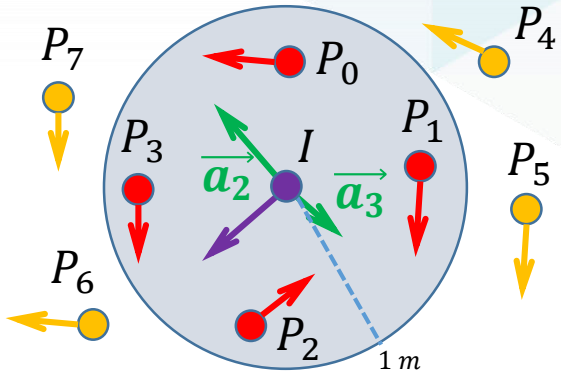
'최대 근접 거리'가 1 m 이내이어야 회피 함.

$$\overrightarrow{vel} = \overrightarrow{v_I} - \overrightarrow{v_P}$$

$$\hat{u} = \widehat{relD} \times \widehat{vel} \times \widehat{vel}$$

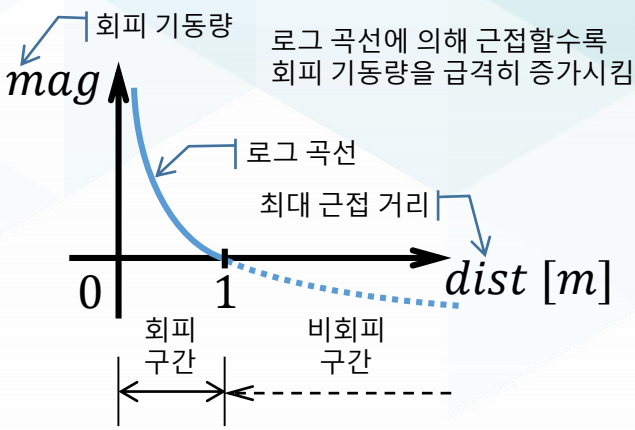
$$\theta = \cos^{-1} \frac{\overrightarrow{relD} \cdot \overrightarrow{vel}}{\left|\overrightarrow{relD}\right|\left|\overrightarrow{vel}\right|}$$

$$dist = \left|\overrightarrow{relD}\right| \sin\theta$$



P_0, P_1은 '회피 조건 1'을 P_4~P_7은 '회피 조건 2'를 만족하지 못하므로 회피하지 않음.

충돌회피



로그 곡선에 의해 근접할수록
회피 기동량을 급격히 증가시킴.

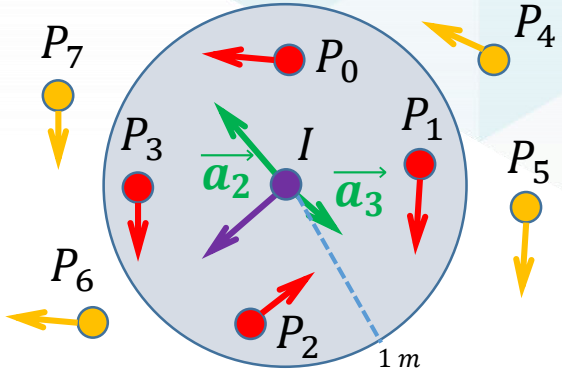
회피 기동 벡터

$$\vec{a} = mag \cdot \hat{u}$$

'회피 기동 벡터'는 '회피 기동량'과
'회피 방향'의 곱임.

$$\overrightarrow{sum} = \sum_{i}^{N} \overrightarrow{a_i}$$

'총 회피 기동 벡터'는 각 상대 별
'회피 기동 벡터'들의 합성 벡터임.



P_0, P_1은 '회피 조건 1'을 P_4~P_7은 '회피 조건 2'를
만족하지 못하므로 회피하지 않음.

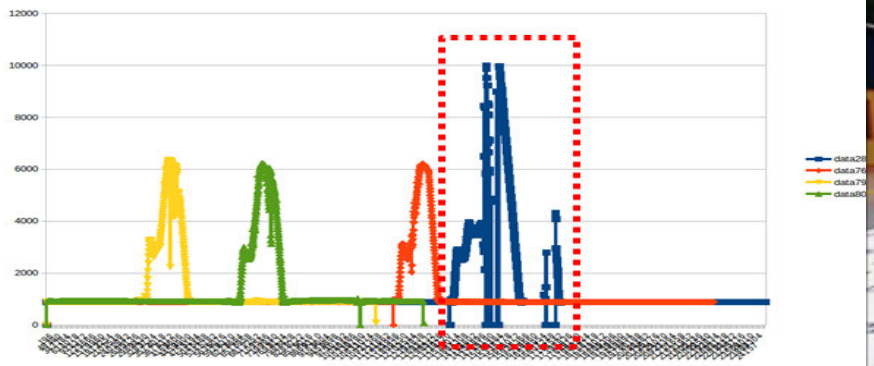
충돌회피

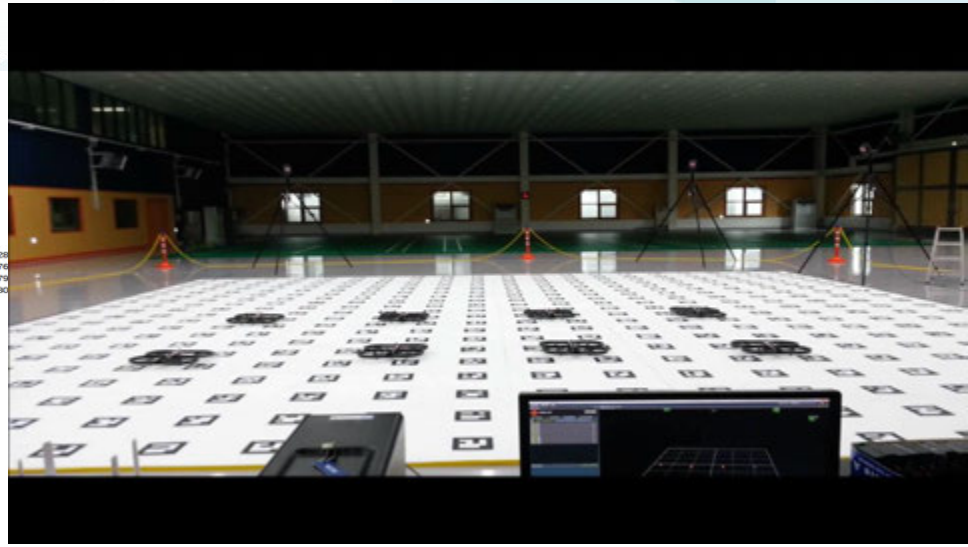
◆ 충돌 회피 시험

좀더 많이 해보자 그리고 한계...

고도 문제

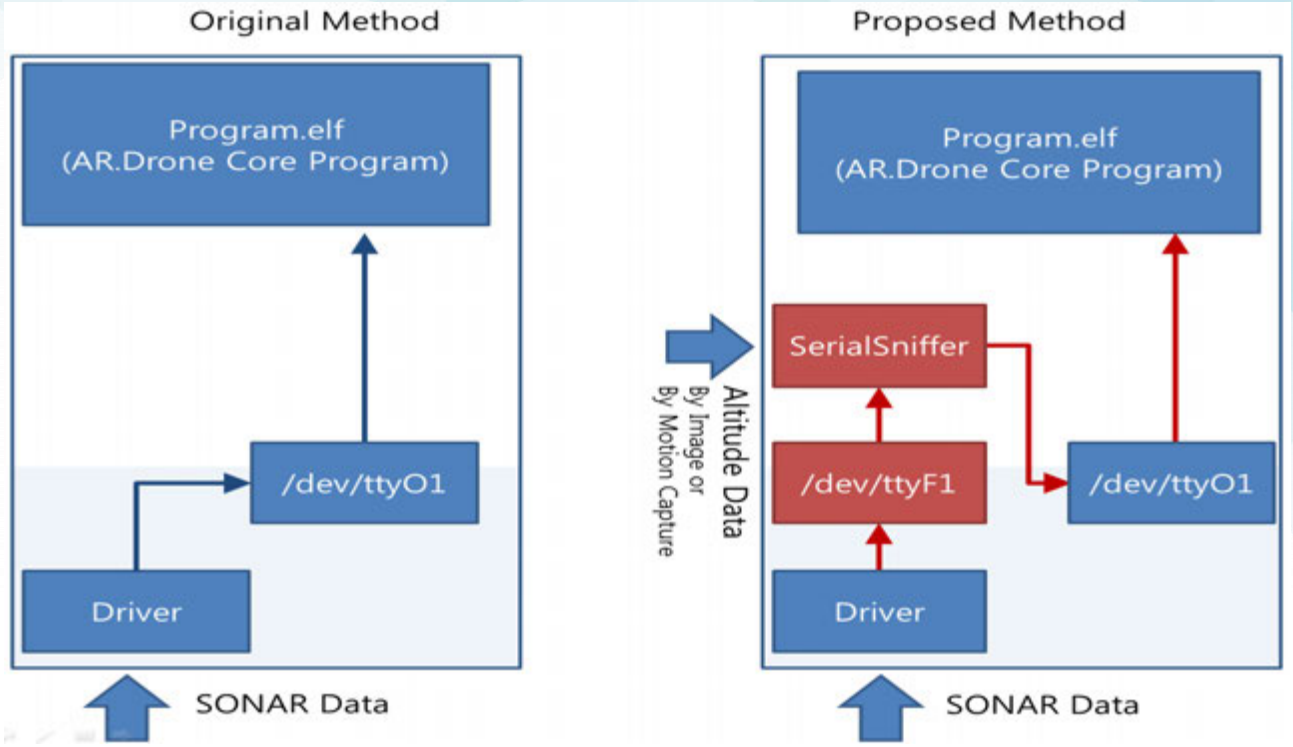
- The AR.Drone uses ultrasound data to measure its altitude
- However, when multiple quadrotors run in same area, the frequency interference happens because of collision of same frequency

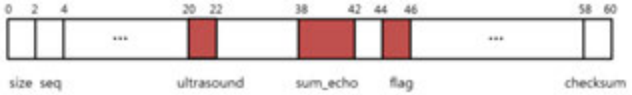




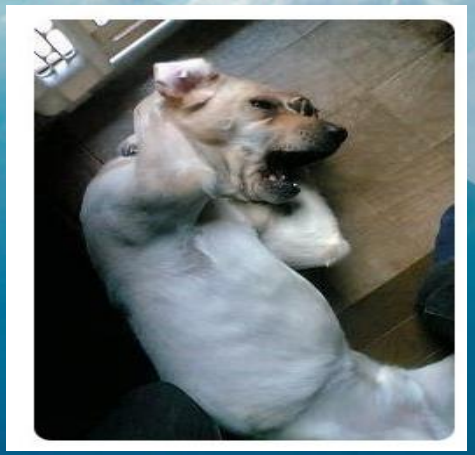
고도 문제

◆ 고도가 정확하지 않다면, 고도 정보를 바꿔 치기하자





좀더 많이 해보자 그리고 한계...

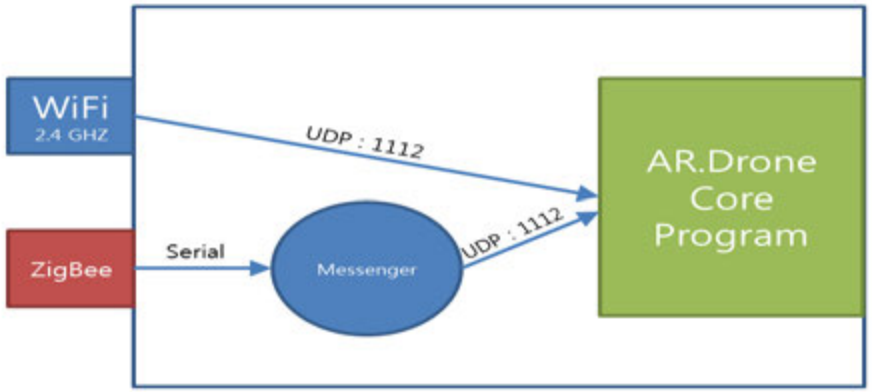


통신 문제

- 통신 불안정으로 인한 다수의 비행체 제어 불가능
- ISM 밴드를 사용하는 WiFi 문제 (2.4 GHz)

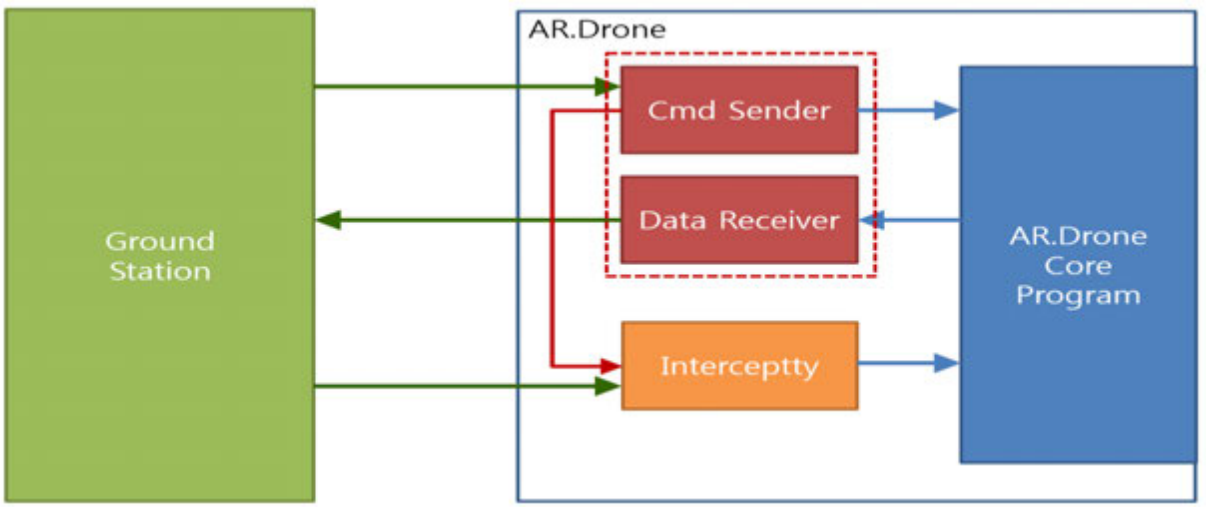
통신 문제

- IDEA 1
  - 두개의 다른 통신 모듈을 활용하여 수신 확률을 높이자
  - Zigbee와 WiFi 동시 연결



통신 문제

- IDEA 2
  - AR.Drone 내부를 수정하자
  - WiFi 특성 상 많은 데이터가 송수신 되면 문제가 발생하므로 송수신 데이터를 줄여보자

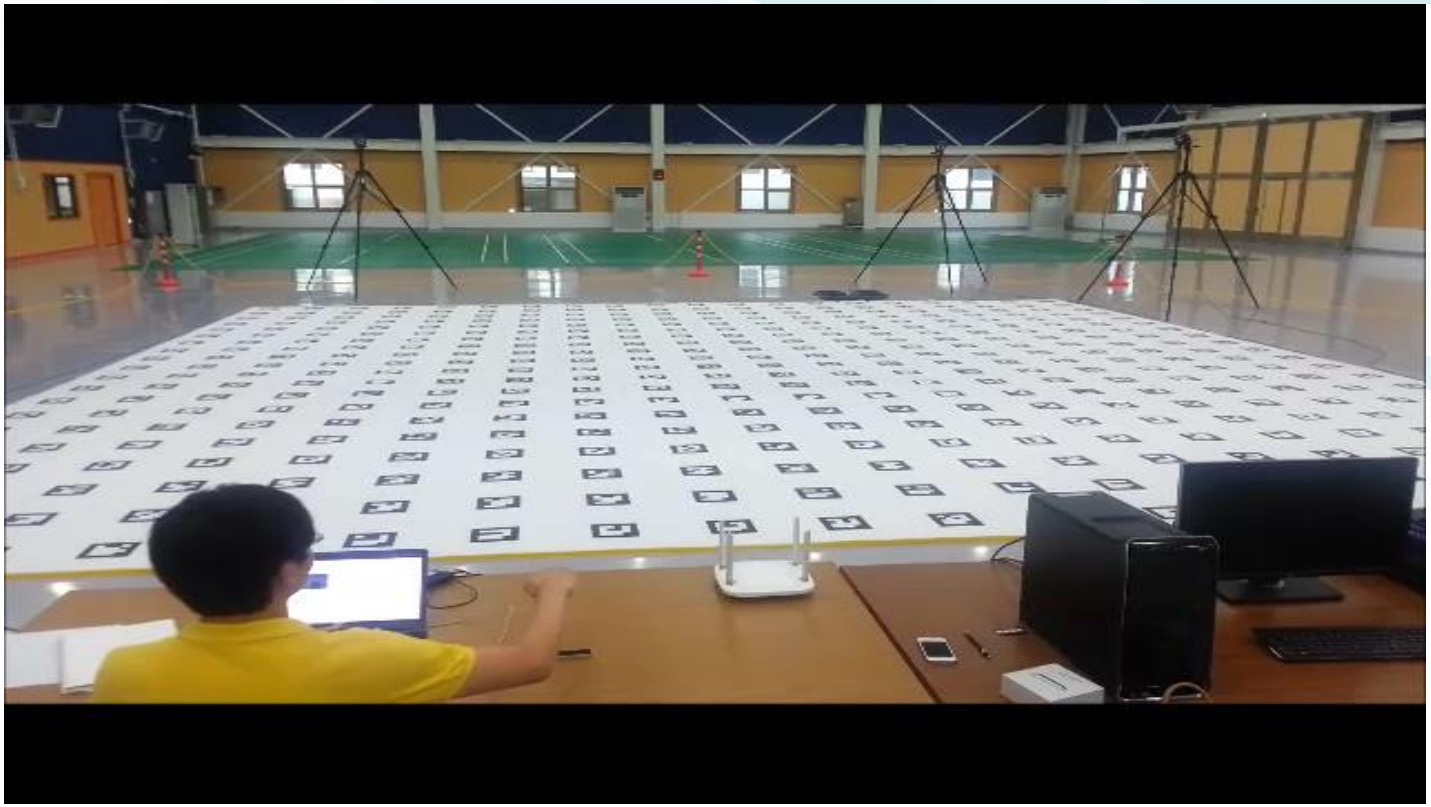


통신 문제

번외 I: 잡다한 생각들

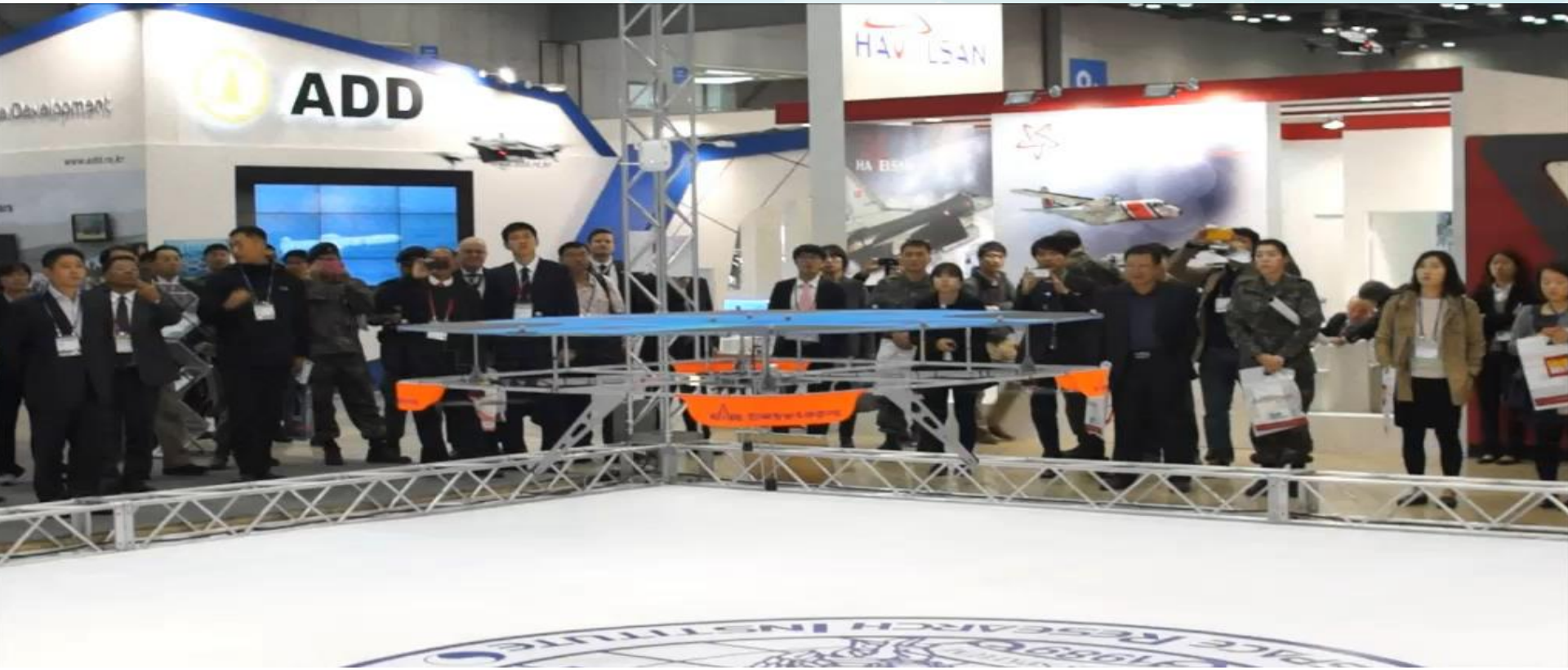
손동작 인식을 통한 제어

◆ Leap Motion (손동작 인식 장치)를 활용한 드론 제어



움직이는 물체에 착륙

◆ 캐리어와 드론간의 상대위치 파악을 통한 착륙



물체 추적 I

◆ 색상 인식을 통한 Yaw 제어

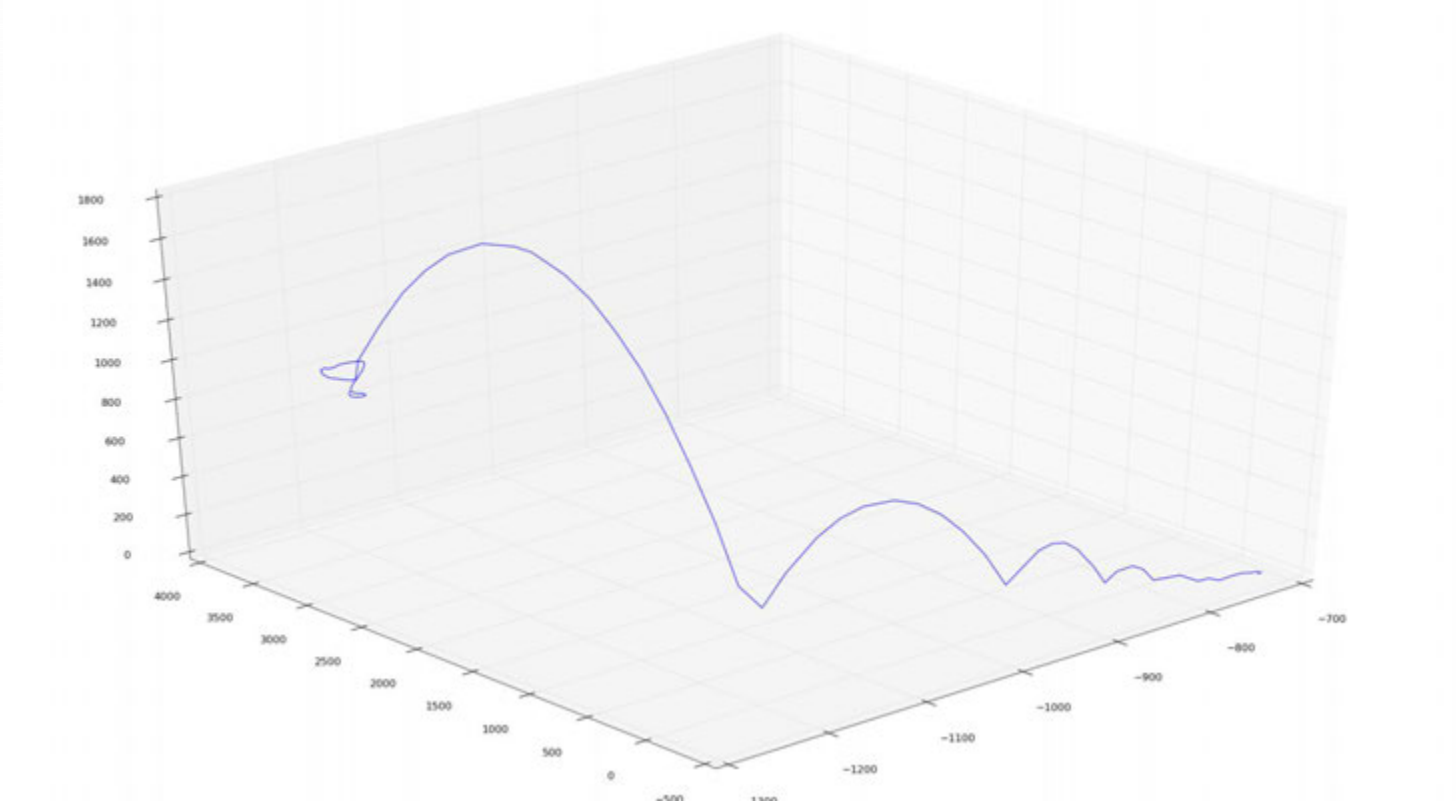
물체 추적 II

◆ TLD (Tracking, Learning, and Detection)를 활용한 물체 추적

번외 II: 볼잡는 드론

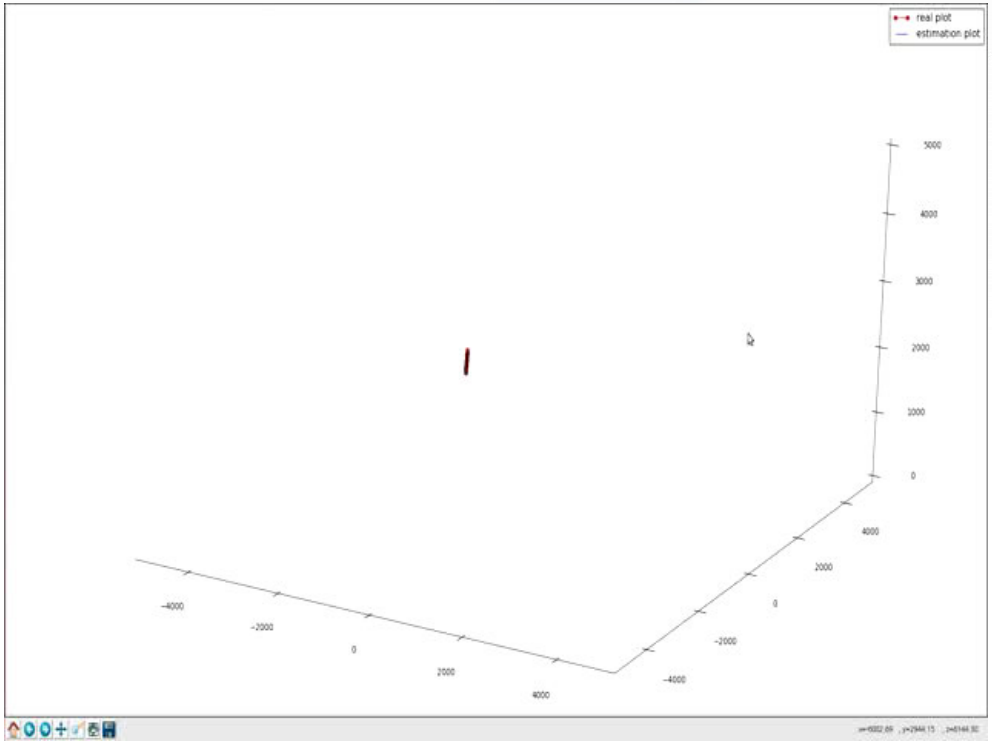
모션캡쳐를 통한 볼 추적

- 100, 200, 500 Hz로 볼 위치 추적 가능
- 잡음 제거 필요
- 볼 추적 데이터 구간 검출 필요



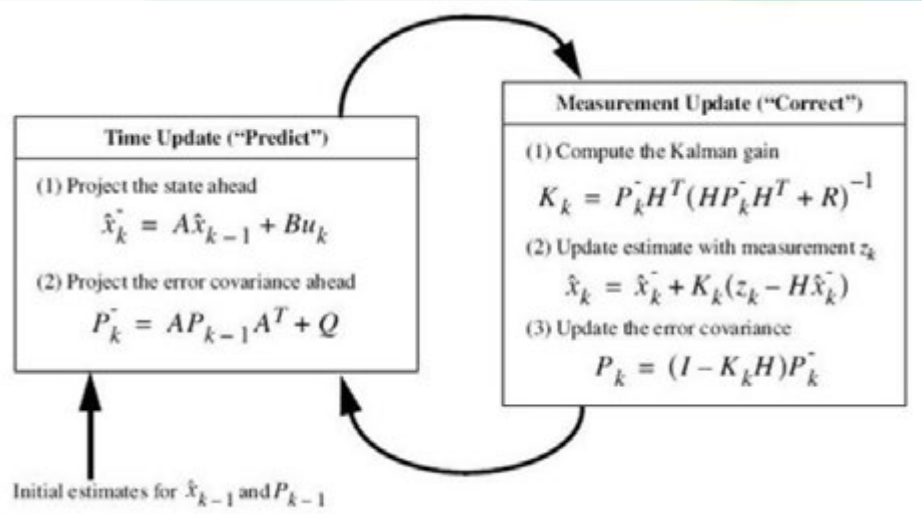
Least Square Method를 통한 볼 위치 추적

- 모션 캡쳐의 정확도 (<1mm)를 기반으로 Least Square Method 사용한 결과 근사한 위치 추적 가능
- 잡음으로 인한 정확도 감소 하는 경향 발생
- 데이터가 많아짐으로써 성능 저하 발생



Linear Kalman Filter를 통한 볼 위치 추적

◆ 볼의 위치 및 속도 모델



$$x = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\frac{g \cdot dt^2}{2} \\ 0 \\ 0 \\ -g \cdot dt \end{bmatrix}$$

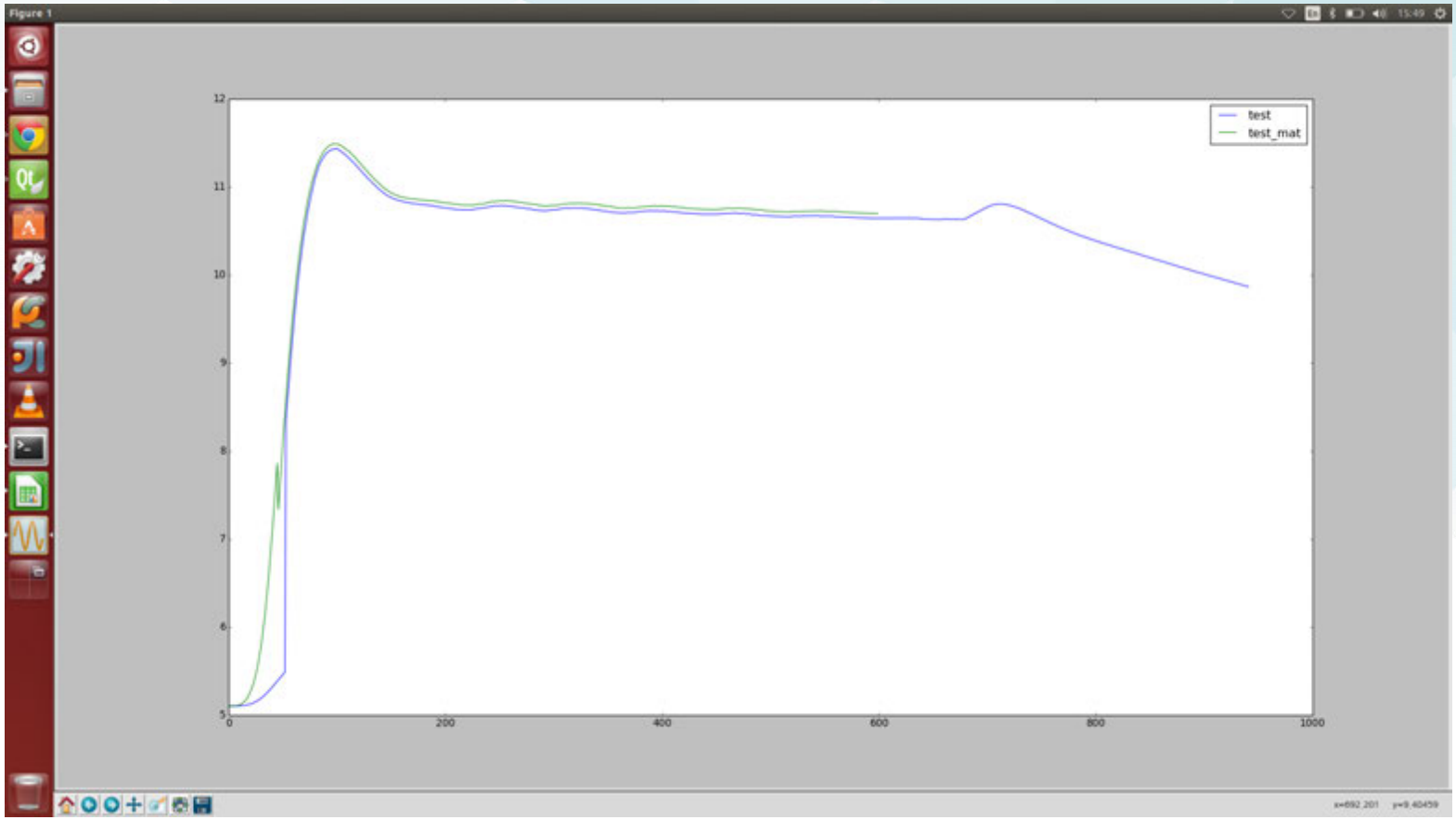
◆ 볼의 위치 추정 (Drag Coefficient 고려)

$$\overrightarrow{vel_{k+1}} = \overrightarrow{vel_k} - T_s \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - T_s \frac{\gamma D^2}{m} \|vel_k\| \overrightarrow{vel_k}$$

$$\overrightarrow{pos_{k+1}} = \overrightarrow{pos_k} + T_s \overrightarrow{vel_k} - \frac{T_s^2}{2} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

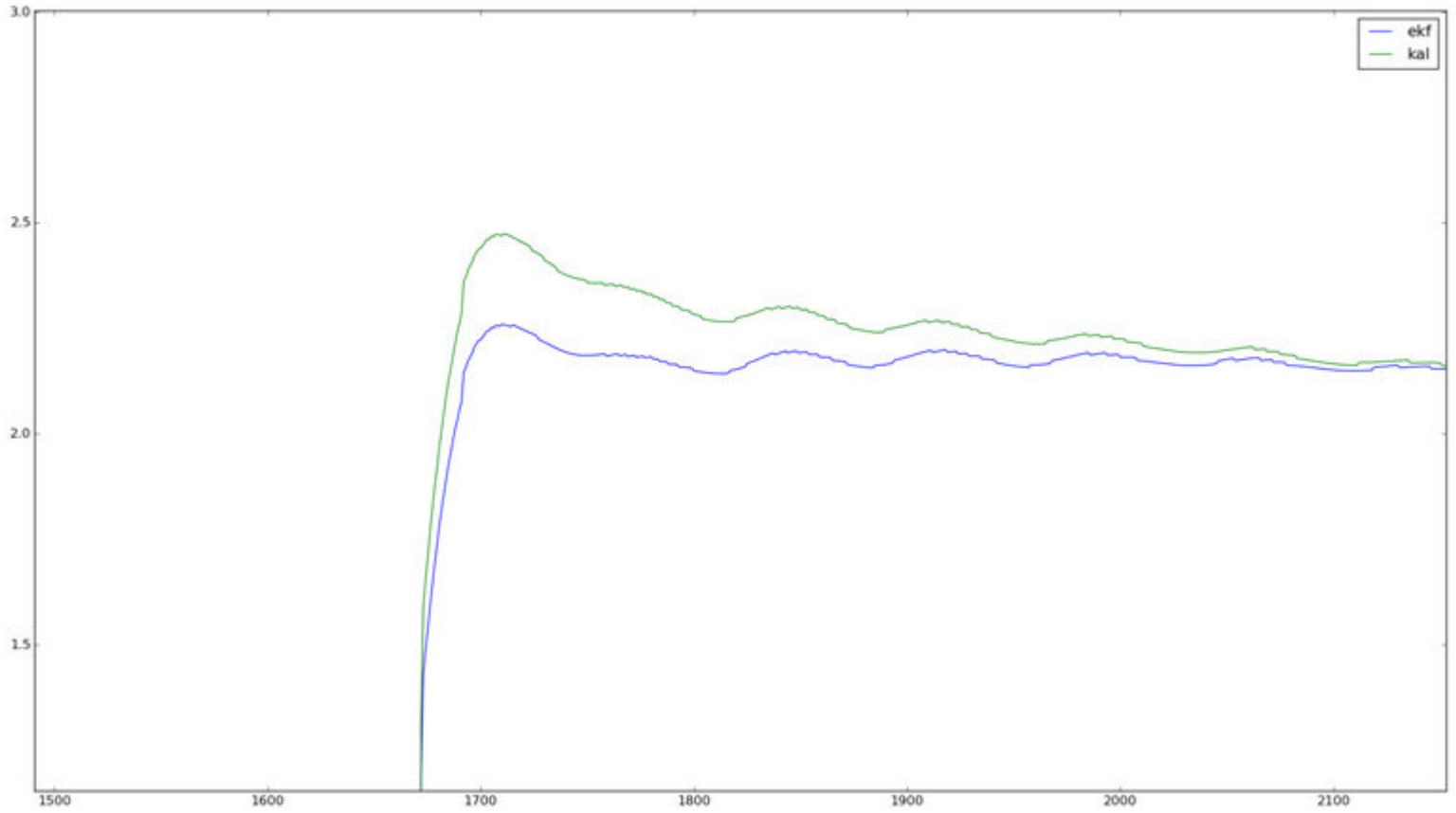
Linear Kalman Filter를 통한 볼 위치 추적

- 500 Hz 동작
- 공을 던지고, 200 ms 이내에 10cm 이하로 떨어지는 공 위치 확인 가능



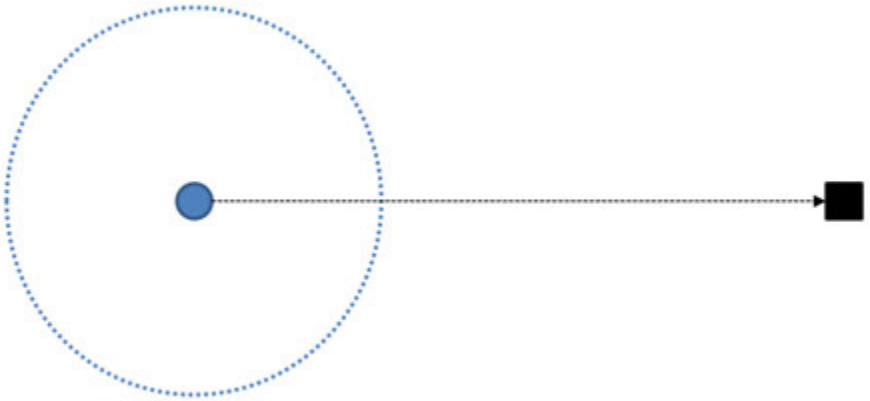
Extened Kalman Filter를 통한 볼 위치 추적

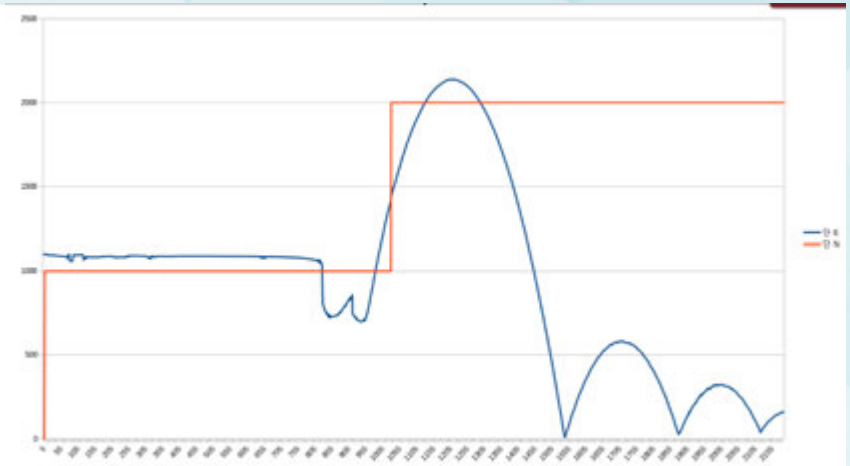
- 500 Hz 동작
- 공을 던지고, 200 ms 이내에 5cm 이하로 떨어지는 공 위치 확인 가능



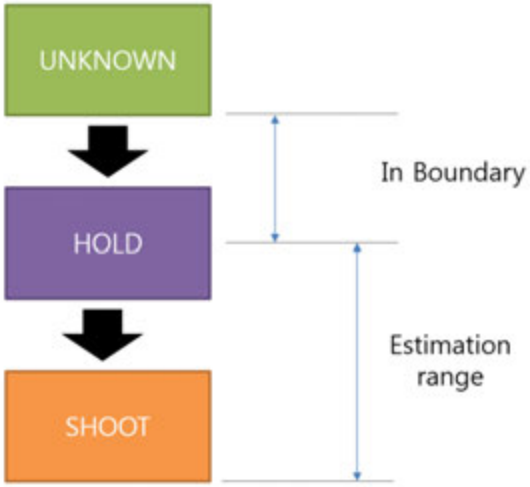
볼 위치 추적 시스템

◆ 모션 캡쳐의 노이즈 제거 및 볼 State를 구분하여 측정용 데이터 추출 구간 분석





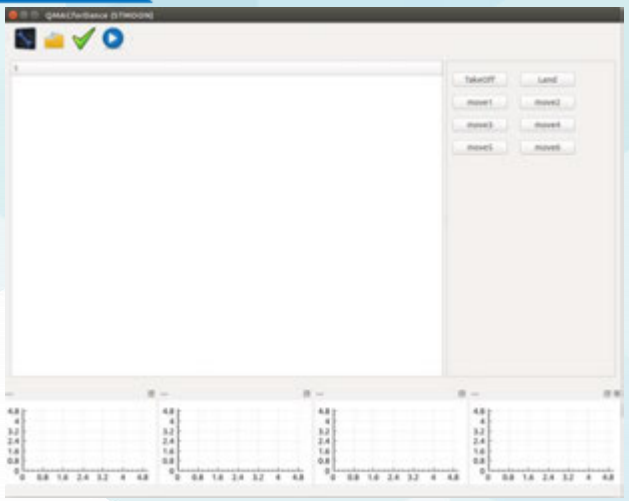
Remove Noise



볼 위치 추적 시스템

개발

- QT 4.0으로 개발
- 500 Hz로 볼 데이터 수신, 250Hz로 기체 제어



기체

- 시험용 기체로 허밍버드 사용
- Gain Scheduling PID 제어로 비행체 제어

시험 결과

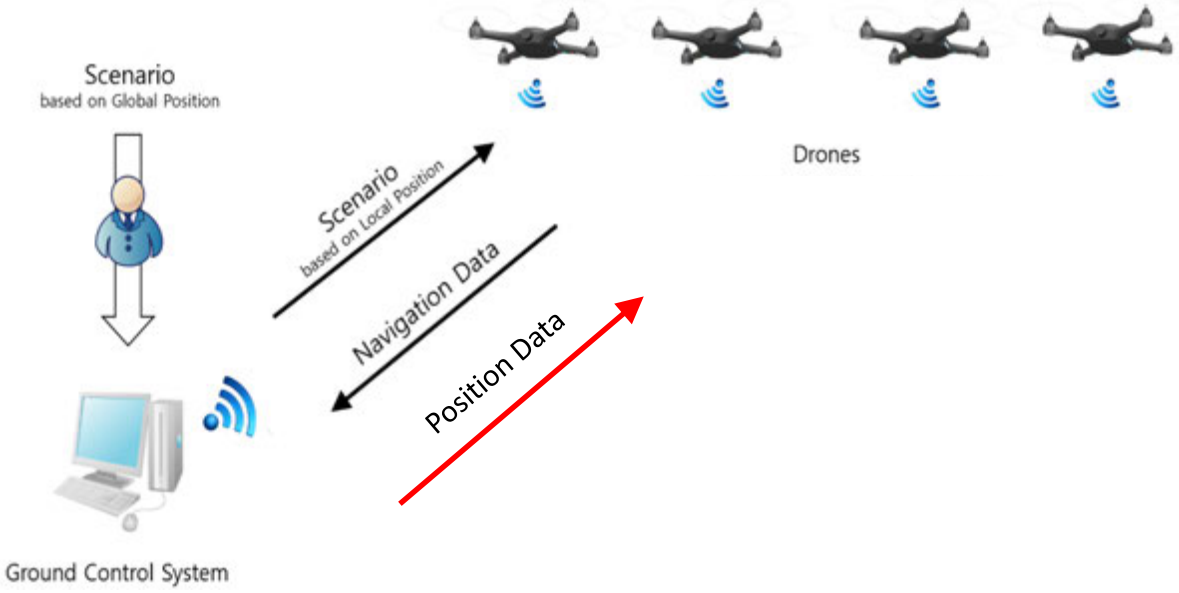
시험 결과

정리

군집비행 필요기술

실내 군집 비행 한계

- [위치 인식] 제한된 범위
  - 고가의 모션 캡처 시스템으로 인한 범위 제약
- [제어] 공개되지 않은 구조의 AR.Drone
- [통신] 제한된 드론 대수 (Centralized System)
  - 통신량 : n x (Scenario + Position) Data



#만약 실외에서 한다면?

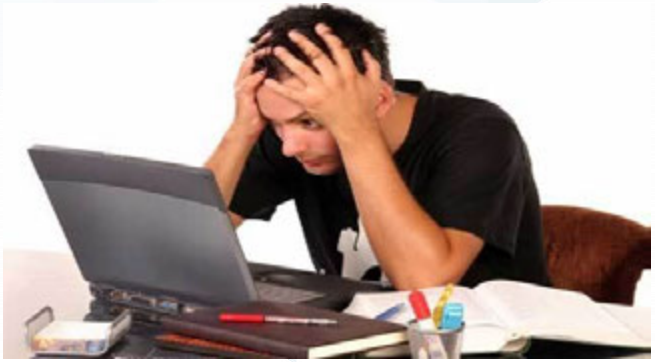
군집비행 필요기술

시스템

비행제어컴퓨터 위치인식 GPS

규모 통신

기체 환경(바람)

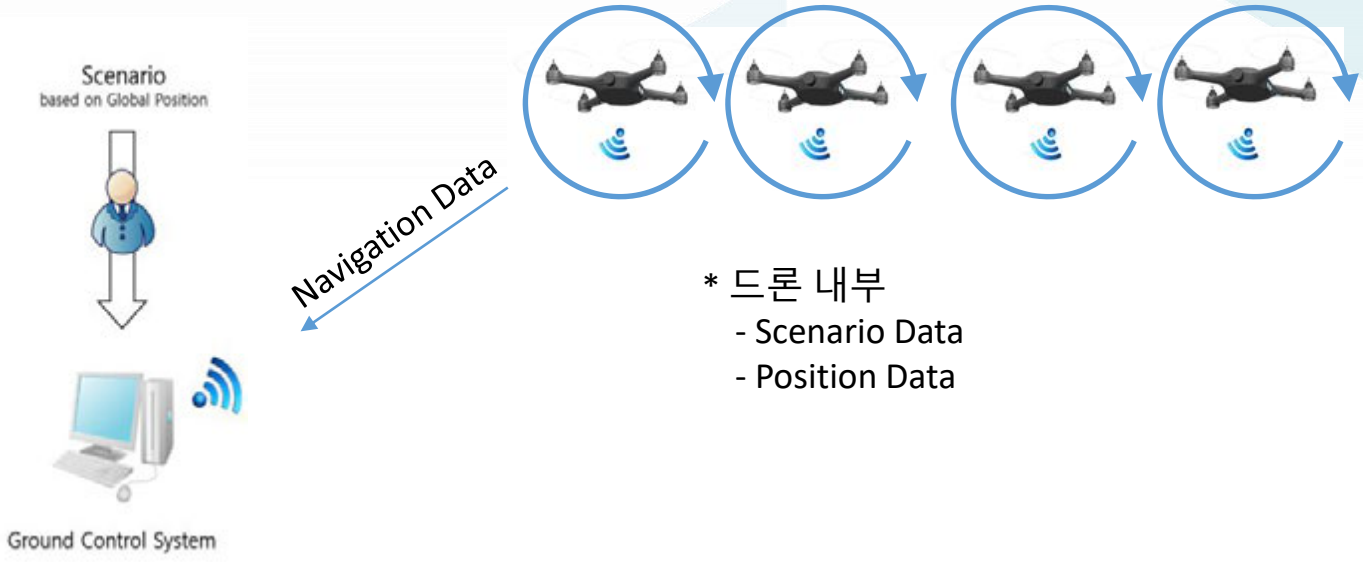


실내
군집 비행

실외
군집 비행

실외 군집 비행 계획

- [위치 인식] RTK-GPS 기반 센서 융합을 통한 위치 예측
  - 최대 10 Km 가능
- [제어] Open Source 기반의 비행제어 컴퓨터 사용
- [통신] 분산 방식 (Distributed System)
  - 통신량 감소 : ~~n x (Scenario + Position) Data~~



II. 비행조종컴퓨터 오픈 생태계

KOREA AEROSPACE RESEARCH INSTITUTE

KARI 한국항공우주연구원 Korea Aerospace Research Institute

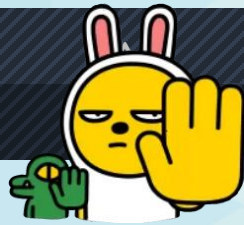
문제는 시스템…

◆ 기존의 시스템을 활용할 수 없을까

펌웨어 수정 불가능

기능 추가 불가능

여기서 잠깐!! (실외 드론 구성 요소)



임무 컴퓨터
(Mission Computer)
MC

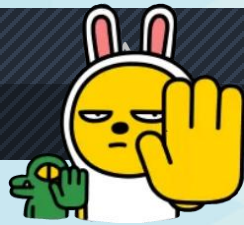
비행제어컴퓨터
(Flight Control Computer)
FCC

WiFi, Zigbee, LTE, ..
2.4 Ghz, 5 GHz, ...

지상국 시스템
(Ground Control System)

FHSS (Frequency Hopping Spread Spectrum)
2.4 GHz

여기서 잠깐!! (실외 드론 구성 요소)

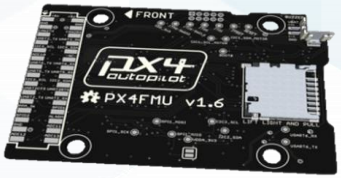


지상국 시스템

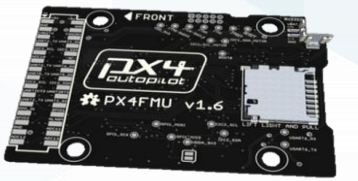
조종기

Pixhawk

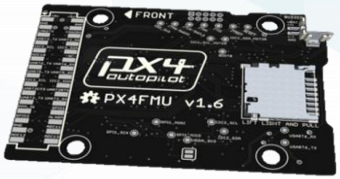
Pixhawk



Pixhawk



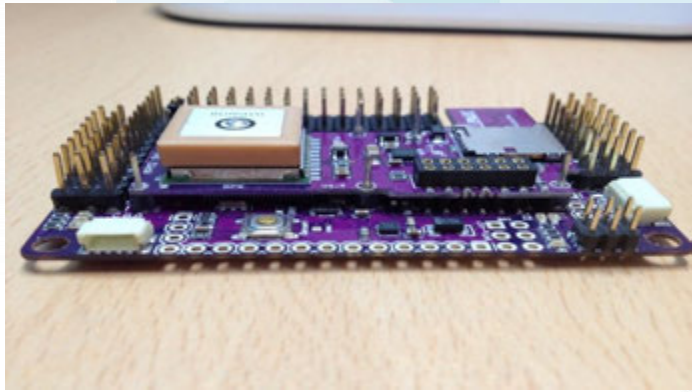
Pixhawk



#그런데 ArduPilot은?

ArduPilot – 구형 비행체 (2012)

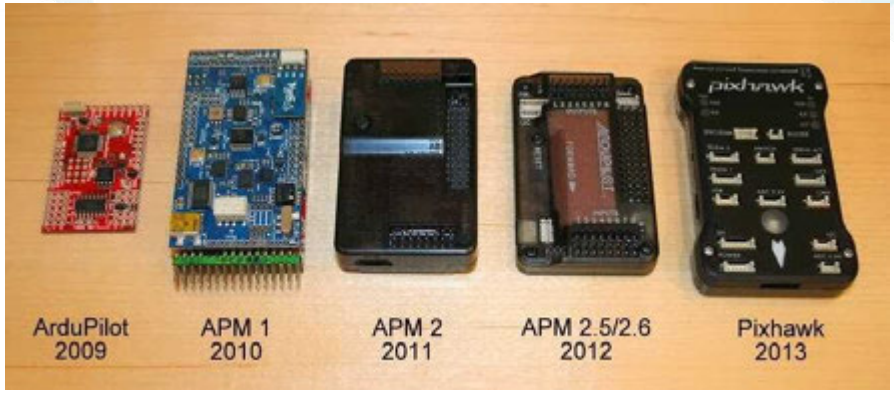
- APM
  - Arduino 기반의 비행제어컴퓨터
  - 간단한 시스템 구축 가능
  - 고성능이 필요한 시스템에 적합하지 않음. ➔ 확장성이 떨어짐



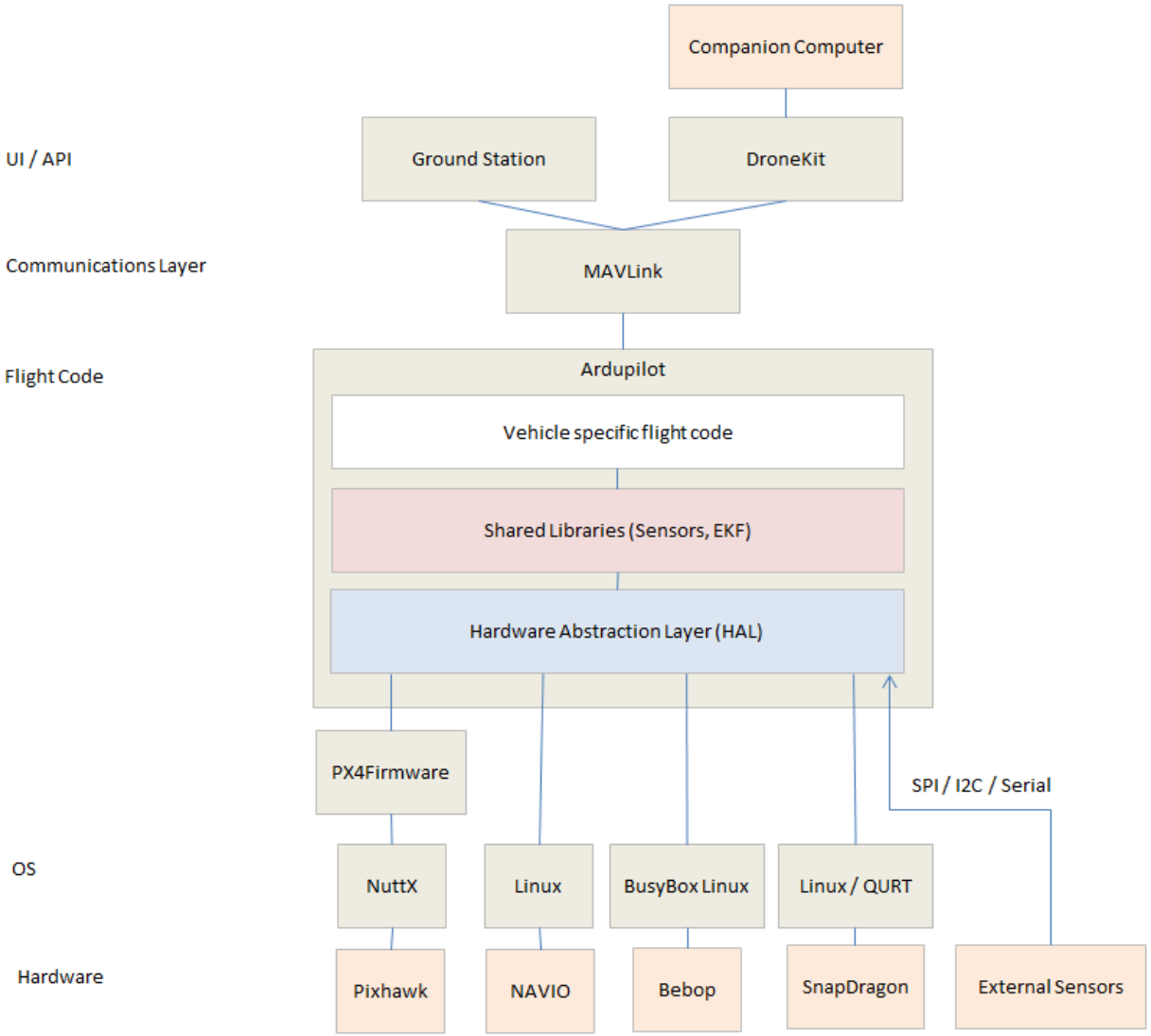
ArduPilot

Jordi Munoz (2009)

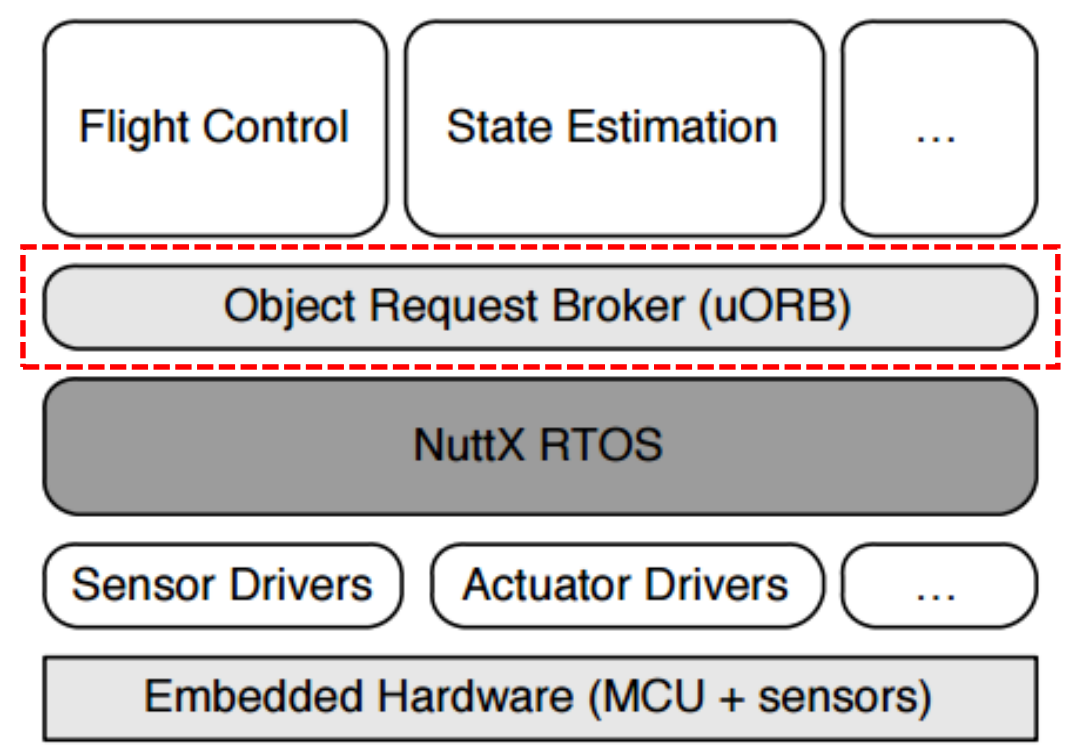
Andrew Tridgell (2011)

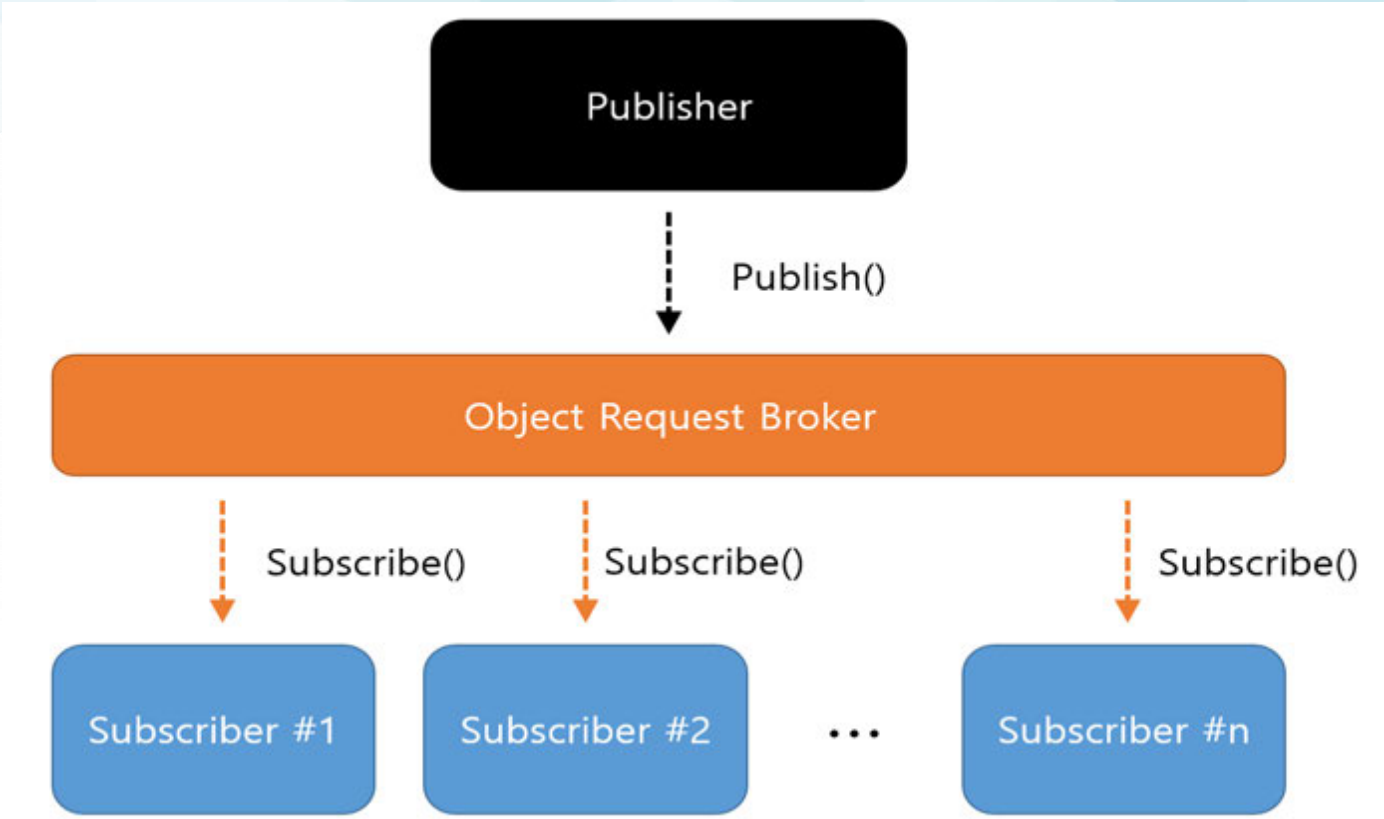


ArduPilot


Companion Computer
UI / API
Ground Station
DroneKit
Communications Layer
MAVLink
Flight Code
Ardupilot
Vehicle specific flight code
Shared Libraries (Sensors, EKF)
Hardware Abstraction Layer (HAL)
PX4Firmware
SPI / I2C / Serial
OS
NuttX
Linux
BusyBox Linux
Linux / QURT
Hardware
Pixhawk
NAVIO
Bebop
SnapDragon
External Sensors

Pixhawk 내부 구조




Publisher
Publish()
Object Request Broker
Subscribe()
Subscribe()
Subscribe()
Subscriber #1
Subscriber #2
...
Subscriber #n

Tutorial (hello world)

Firmware/src/module/px4_simple_app/module.mk

```
MODULE_COMMAND              = px4_simple_app
SRCS                        = px4_simple_app.c
```

Hello world program

```
#include <nuttx/config.h>
#include <stdio.h>
#include <errno.h>

__EXPORT int px4_simple_app_main(int argc, char *argv[]);

int px4_simple_app_main(int argc, char *argv[])
{
        printf("Hello Sky!\n");
        return OK;
}
```

Tutorial (hello world)

makefile

```
MODULES         += modules/px4_simple_app
```

compile

```
make clean
make px4fmu-v2_default
```

```
make upload px4fmu-v2_default
```

```
Found board 5,0 on /dev/tty.usbmodem1
erase...
program...
verify...
done, rebooting.
```

run

```
nsh> px4_simple_app
Hello Sky!
```

Tutorial (subscribe)

```
#include <poll.h>
#include <uORB/topics/sensor_combined.h>
..
int sensor_sub_fd = orb_subscribe(ORB_ID(sensor_combined));

/* one could wait for multiple topics with this technique, just using one here */
struct pollfd fds[] = {
        { .fd = sensor_sub_fd,   .events = POLLIN },
};

while (true) {
        /* wait for sensor update of 1 file descriptor for 1000 ms (1 second) */
        int poll_ret = poll(fds, 1, 1000);
..
        if (fds[0].revents & POLLIN) {
                /* obtained data for the first file descriptor */
                struct sensor_combined_s raw;
                /* copy sensors raw data into local buffer */
                orb_copy(ORB_ID(sensor_combined), sensor_sub_fd, &raw);
                printf("[px4_simple_app] Accelerometer:\t%8.4f\t%8.4f\t%8.4f\n",
                                        (double)raw.accelerometer_m_s2[0],
                                        (double)raw.accelerometer_m_s2[1],
                                        (double)raw.accelerometer_m_s2[2]);
}
```

```
[px4_simple_app] Accelerometer:    0.0483          0.0821          0.0332
[px4_simple_app] Accelerometer:    0.0486          0.0820          0.0336
[px4_simple_app] Accelerometer:    0.0487          0.0819          0.0327
[px4_simple_app] Accelerometer:    0.0482          0.0818          0.0323
[px4_simple_app] Accelerometer:    0.0482          0.0827          0.0331
[px4_simple_app] Accelerometer:    0.0489          0.0804          0.0328
```

Tutorial (publish)

```
#include <uORB/topics/vehicle_attitude.h>
..
/* advertise attitude topic */
struct vehicle_attitude_s att;
memset(&att, 0, sizeof(att));
int att_pub_fd = orb_advertise(ORB_ID(vehicle_attitude), &att);
```

```
orb_publish(ORB_ID(vehicle_attitude), att_pub_fd, &att);
```

멀티콥터형 드론 제어기 구조

History of Pixhawk Control Module

Loren Meier

Julian Oes

Architecture

Control &
Estimation

Paul Reiseborough
[EKF2]

Anton Babuskin
[position_estimation_inav]
[attitude_estimator_q]
[mc_att_control]
[mc_pos_control]
…

James Goppert
[LPE]

#그런데 왜 소스를 오픈하나?

Why do you open your code?

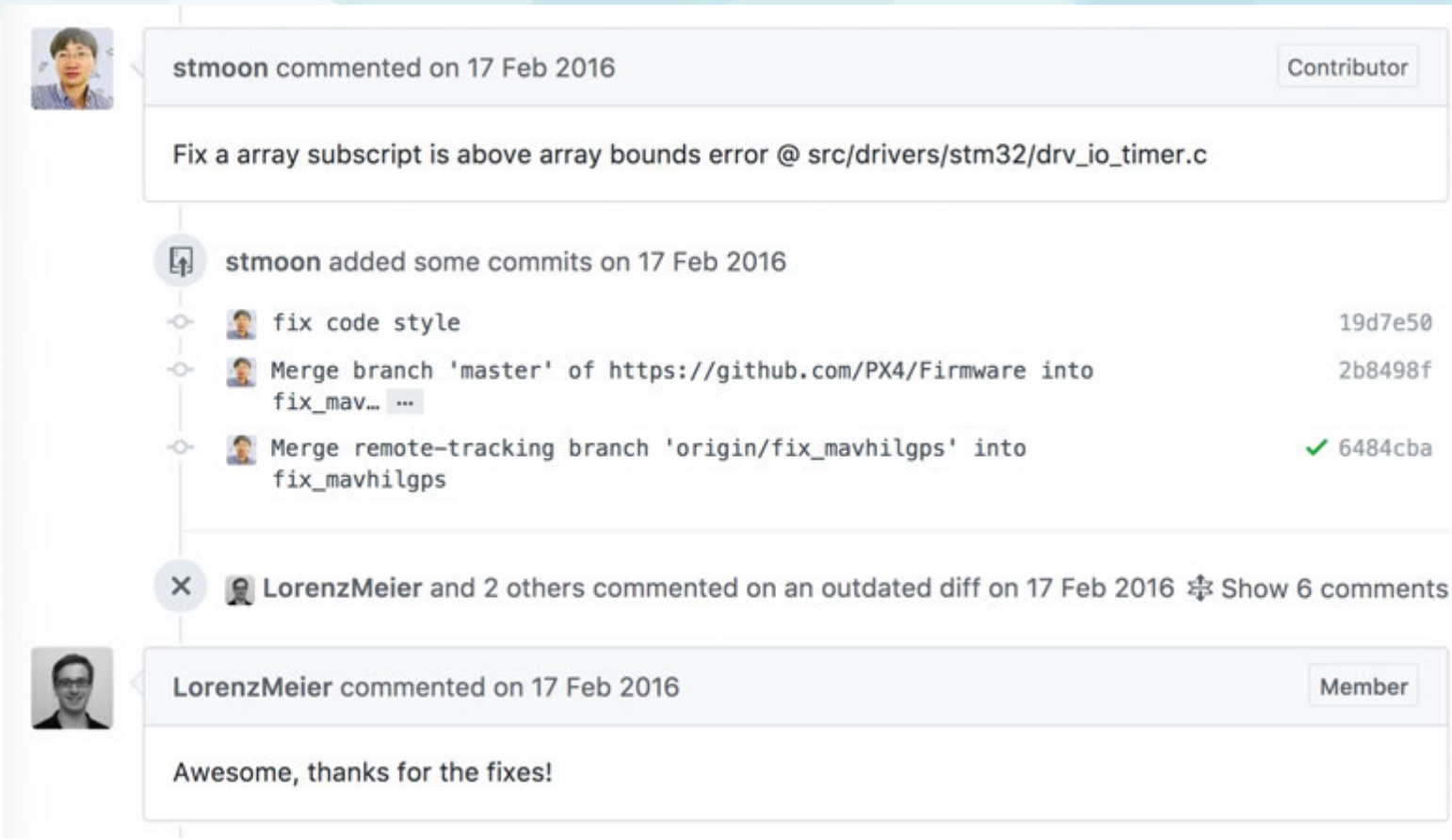
- Why *
  - 개인: 개인적인 호기심/재미, 개인 명성
    - ➔ **좋은 회사에 취직/연봉인상** 이 목적이 아닐까?
  - 회사: 기술력 과시, 좋은 개발자 채용
    - ➔ 하지만, **기술 공유** 보다는 **기술 종속**이 목적이 아닐까?

*) 김형준, "오픈소스 프로젝트 따라 잡기", SlideShare

#그런데 어떻게 공헌할까?

How to become contributor

How to become contributor



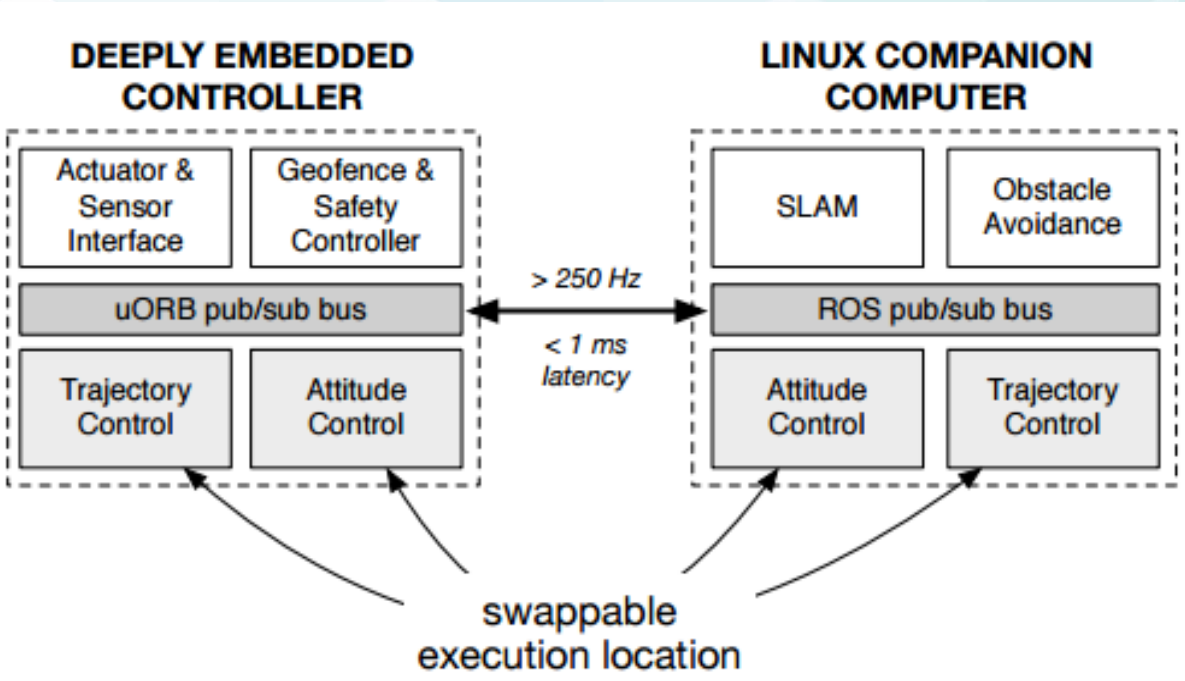
How to become contributor

GitHub, Inc. [US] https://github.com/PX4/Firmware/blob/master/src/modules/logger/params.c

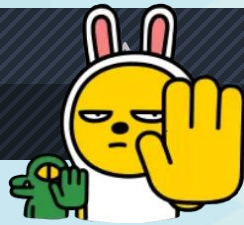
앱 Linear Algebra | Ma... TeamGantt DeepLearning COMS 파워포인트 Mad for Simplicity :... 영상처리 kalmanfilter

```
30   * POSSIBILITY OF SUCH DAMAGE.
31   *
32   ****************************************************************************/
33
34  /**
35   * UTC offset (unit: min)
36   *
37   * the difference in hours and minutes from Coordinated
38   * Universal Time (UTC) for a your place and date.
39   *
40   * for example, In case of South Korea(UTC+09:00),
41   * UTC offset is 540 min (9*60)
42   *
43   * refer to https://en.wikipedia.org/wik
44   *
45   * @unit min
46   * @min -1000
47   * @max  1000
48   * @group SD Logging
49   */
50  PARAM_DEFINE_INT32(SDLOG_UTC_OFFSET, 0);
```

Pixhawk 확장성


DEEPLY EMBEDDED CONTROLLER
LINUX COMPANION COMPUTER
Actuator & Sensor Interface
Geofence & Safety Controller
uORB pub/sub bus
Trajectory Control
Attitude Control
> 250 Hz
< 1 ms latency
SLAM
Obstacle Avoidance
ROS pub/sub bus
Attitude Control
Trajectory Control
swappable execution location

여기서 잠깐!! (드론 통신 프로토콜)



임무 컴퓨터
(Mission Computer)
MC

MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

WiFi, Zigbee, LTE, ..
2.4 Ghz, 5 GHz, ...

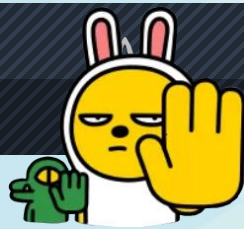
MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

비행제어컴퓨터
(Flight Control Computer)
FCC

FHSS (Frequency Hopping Spread Spectrum)
2.4 GHz

지상국 시스템
(Ground Control System)

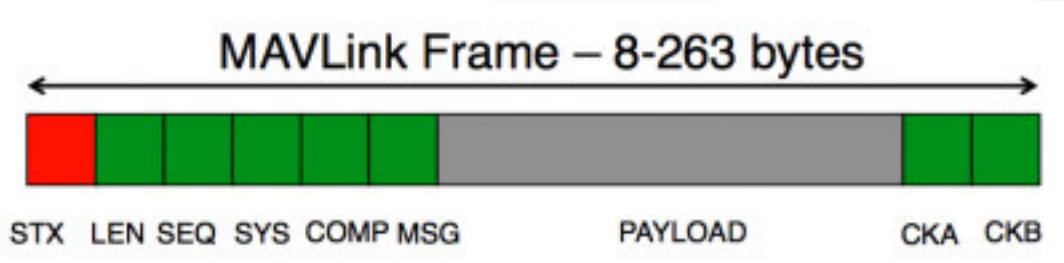
여기서 잠깐!! (드론 통신 프로토콜)



MAVLINK

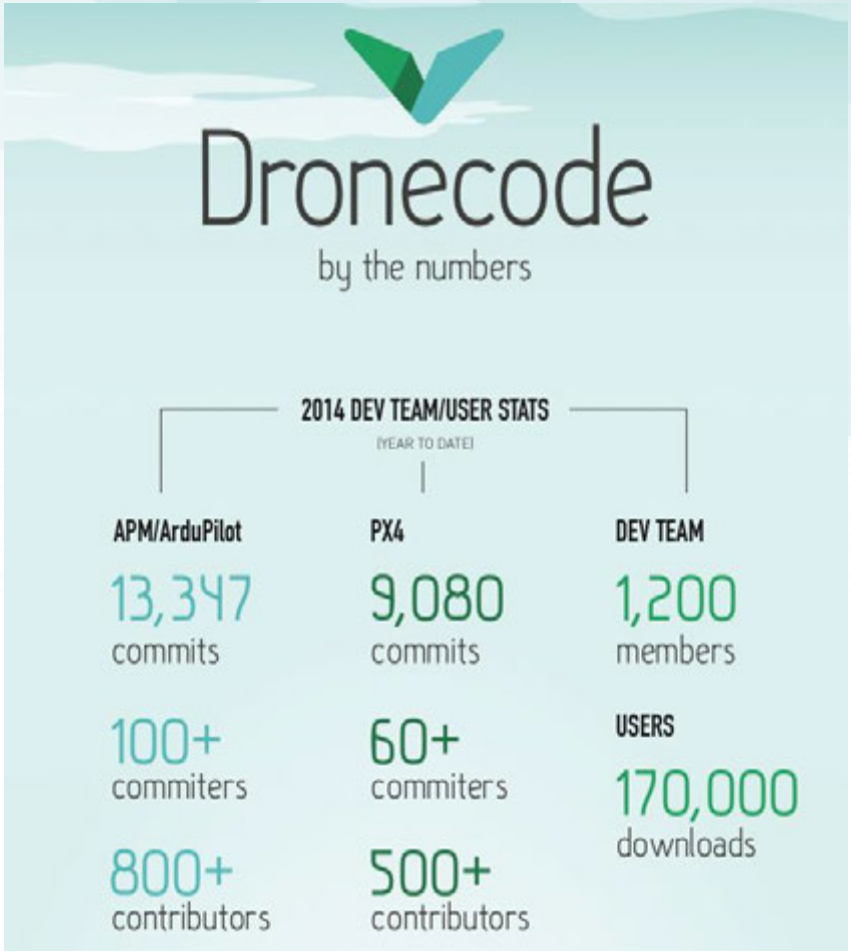
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

"MAVLink is a very lightweight, header-only message marshalling library for micro air vehicles."




```
<message id="11" name="SET_MODE">
  <description>THIS INTERFACE IS DEPRECATED. USE COMMAND_LONG with MAV_CMD_DO_SET_MODE I
  <field type="uint8_t" name="target_system">The system setting the mode</field>
  <field type="uint8_t" name="base_mode" enum="MAV_MODE">The new base mode</field>
  <field type="uint32_t" name="custom_mode">The new autopilot-specific mode. This field
</message>
```

Drone Open Source Eco-System



Drone Open Source Eco-System

III. 실외 군집 비행

KOREA AEROSPACE RESEARCH INSTITUTE

KARI 한국항공우주연구원 Korea Aerospace Research Institute

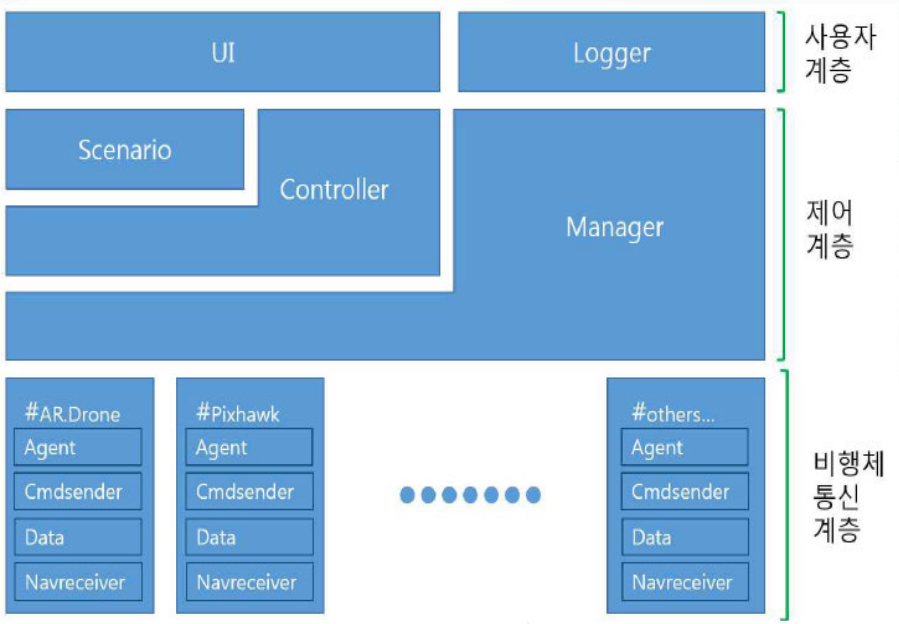
Control

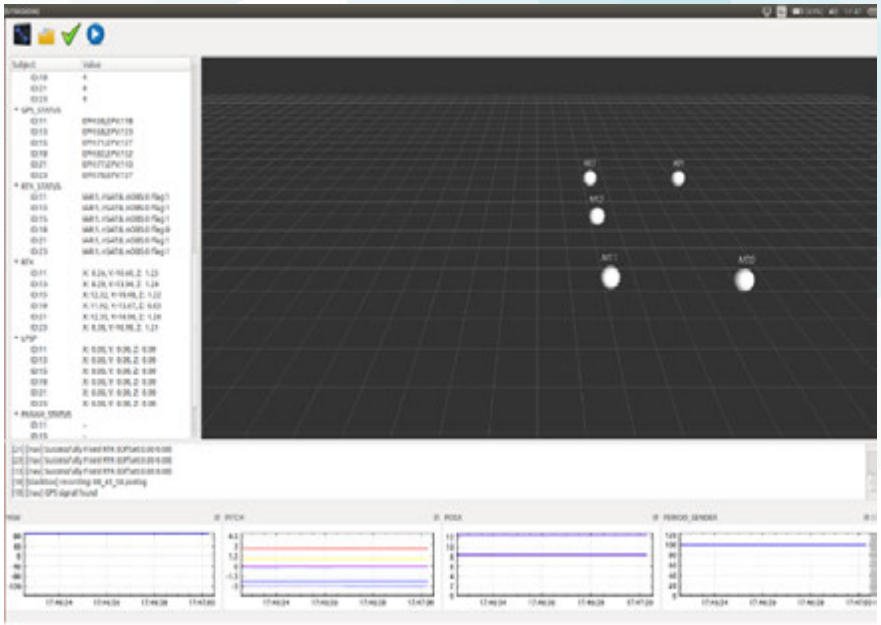
- 충격 감소를 위한 플라스틱 볼트 적용
- Yaw 축 제어 성능 향상 위한 모터 회전축 3° 기울임
- 단순한 구조로 프로펠러 후류를 이용한 냉각을 수행할 수 있도록 설계 보완

GCS

◆ 운영체제에 독립적인 지상국 시스템 개발 (Linux, MAC, Window, Android 등에서 활용 가능)

◆ 모듈화를 통한 이기종 기체에 적용 가능





위치 예측 방법

measure sensor information → Predict position → Correct position → correct accelerometer bias

◆ **Predict position by accelerometer**

$$\hat{p}^{-} = v\Delta t + \frac{1}{2}a\Delta t^2$$

$$\hat{v}^{-} = a\Delta t$$

◆ **Correct position**

$$\hat{p} = \hat{p}^{-} + e_p w_p \Delta t$$

$$\hat{v} = \hat{v}^{-} + e_p w_p^2 \Delta t + e_v w_v \Delta t$$

◆ **Correct Accel**

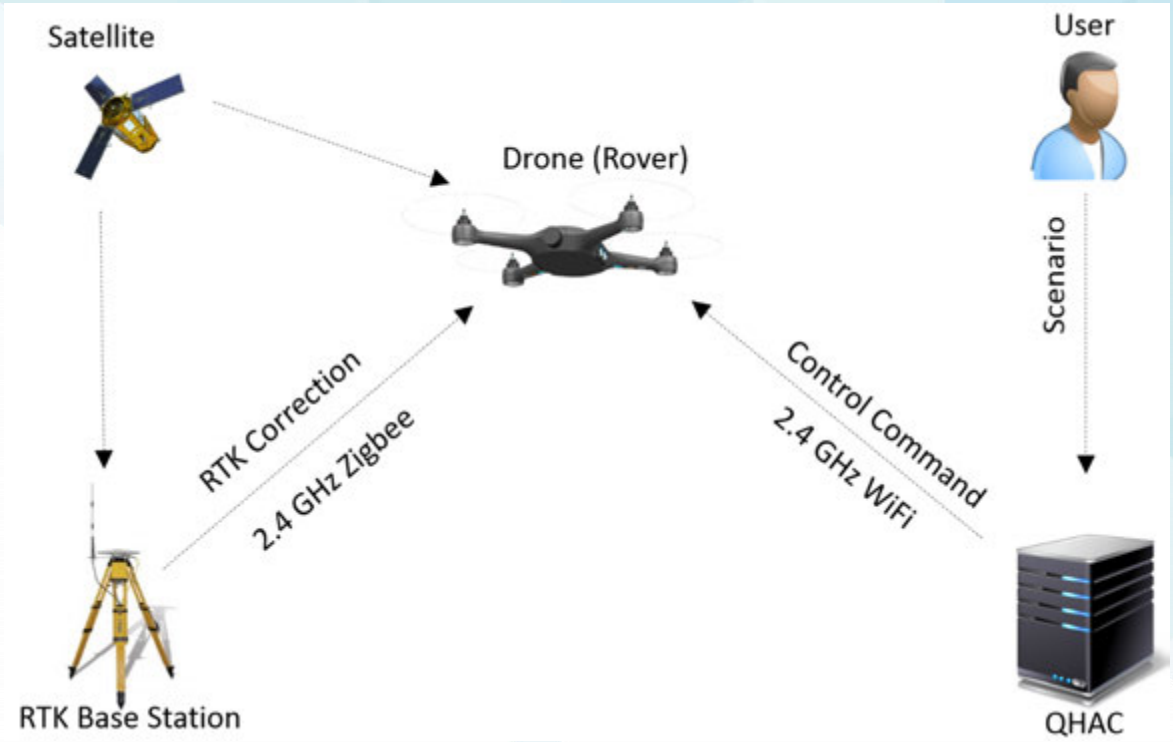
$$\hat{a}_{acc}^{-} = \hat{a}_{acc} - \delta_{gps} - \delta_{lidar}$$

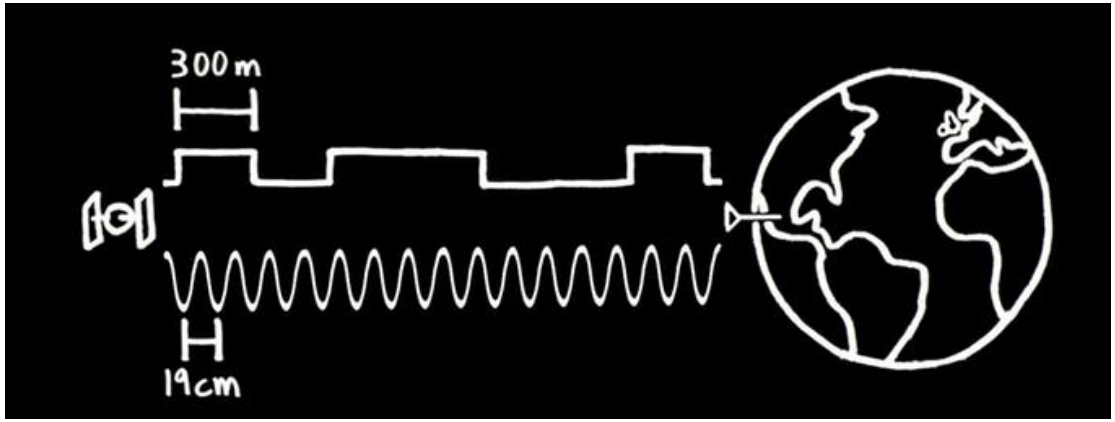
$$\delta_{gps} = e_p w_p^2 + e_v w_v$$

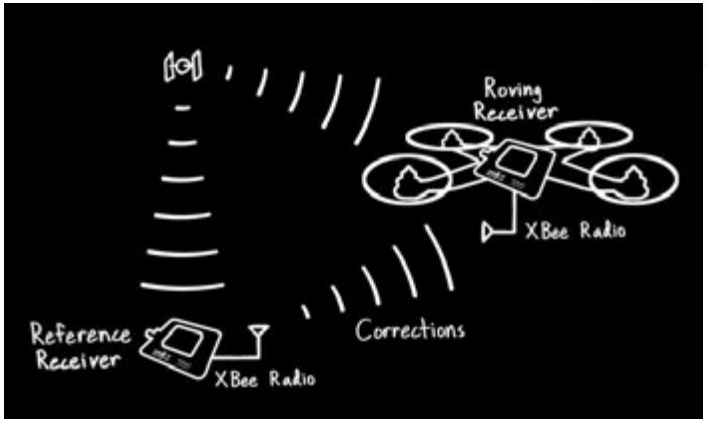
위치 인식 방법

- GPS-INS
  - 일반적인 시스템에서 활용
  - GPS 상태에 따라 다름 (최대 5m 범위)
- Vision
  - 영상 정보를 바탕으로 GPS가 없는 환경에서 활용 가능
  - 상황에 따라 매우 다양한 결과 생성
- UWB
  - 50 cm 이내 위치 정밀도 가능 (GPS 없는 환경에서 활용 가능)
  - 다수의 Base System 필요하고 한정적인 지역에서만 가능
- RTK-GPS
  - 10 cm 이내 위치 정밀도 가능
  - 고가 장비

RTK-GPS


Satellite
User
Drone (Rover)
Scenario
RTK Correction
2.4 GHz Zigbee
Control Command
2.4 GHz WiFi
RTK Base Station
QHAC


300 m
H
19cm


Roving Receiver
XBee Radio
Reference Receiver
XBee Radio
Corrections

RTK-GPS

Features

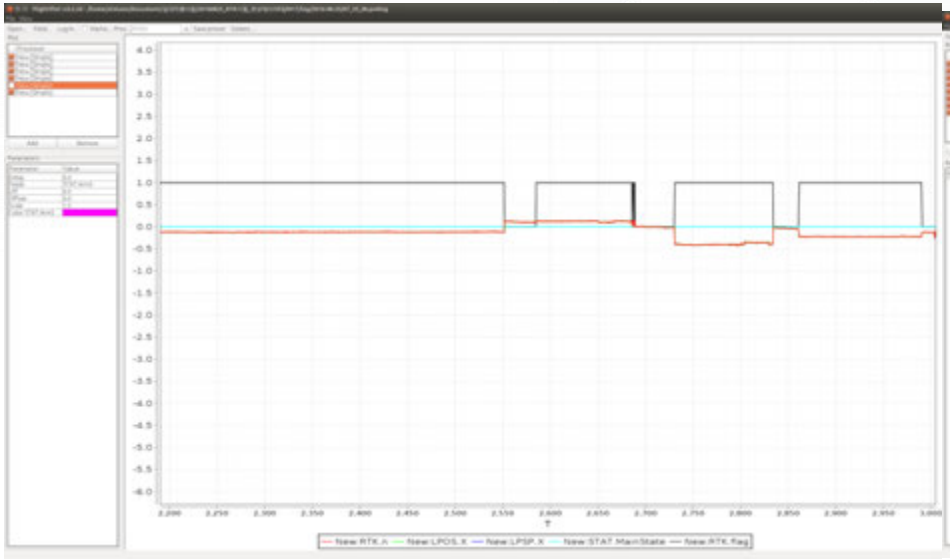
- Centimeter accurate relative positioning (Carrier phase RTK)
- 50 Hz position/velocity/time solutions
- Open-source software and board design
- Low power consumption - 500mW typical
- Small form factor - 53x53mm
- USB and dual UART connectivity
- Integrated patch antenna and external antenna input
- Full-rate raw sample pass-through over USB
- 3-bit, 16.368 MS/s L1 front-end supports GPS, GLONASS, Galileo and SBAS signals

Model	OEM V2	Piksi
Frequency	GPS/GLONASS (L1, L2)	GPS/GLONASS/ Galileo L1
Position Accuracy	1 cm	2~4 cm
Data Rate	50 Hz	10 Hz
Dimensions	60x100x13 mm	53x53x12.8 mm
Weight	56 g	32 g
Power	3.3 VDC, 1.2 W	3.5~5.5 VDC, 500 mW
Price	$ 10,000 이상	$ 495

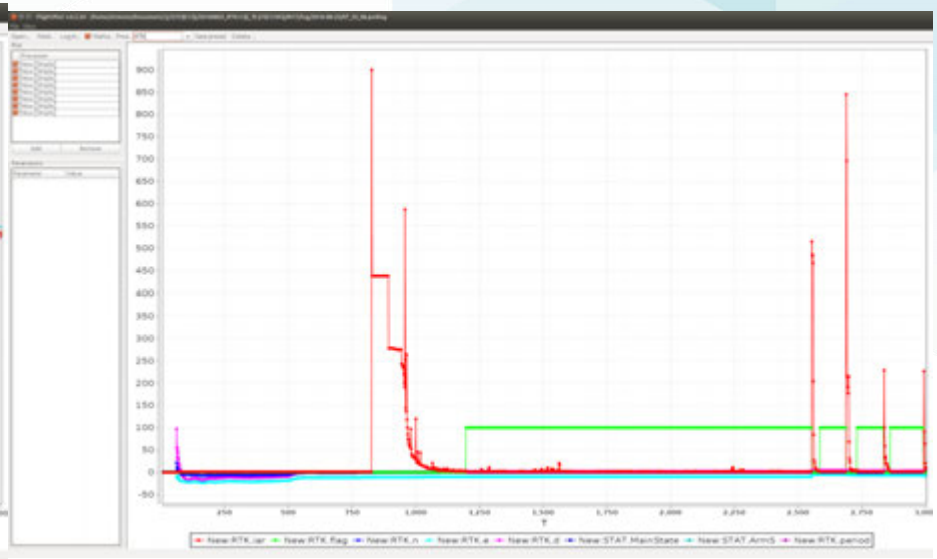
RTK-GPS

RTK-GPS Problem

- RTK 모듈
  - TTF (Time To Fixed) 1 시간 이상 발생
  - Float→Fixed→Float되는 경우 위치 변경 현상
  - Fixed 모드가 중간에 끊기는 현상



[문제 현상 #1]



[문제 현상 #2]

RTK-GPS → GPS-INS

실외군집비행

(야간 비행 3대)

미래항공우주기술팀

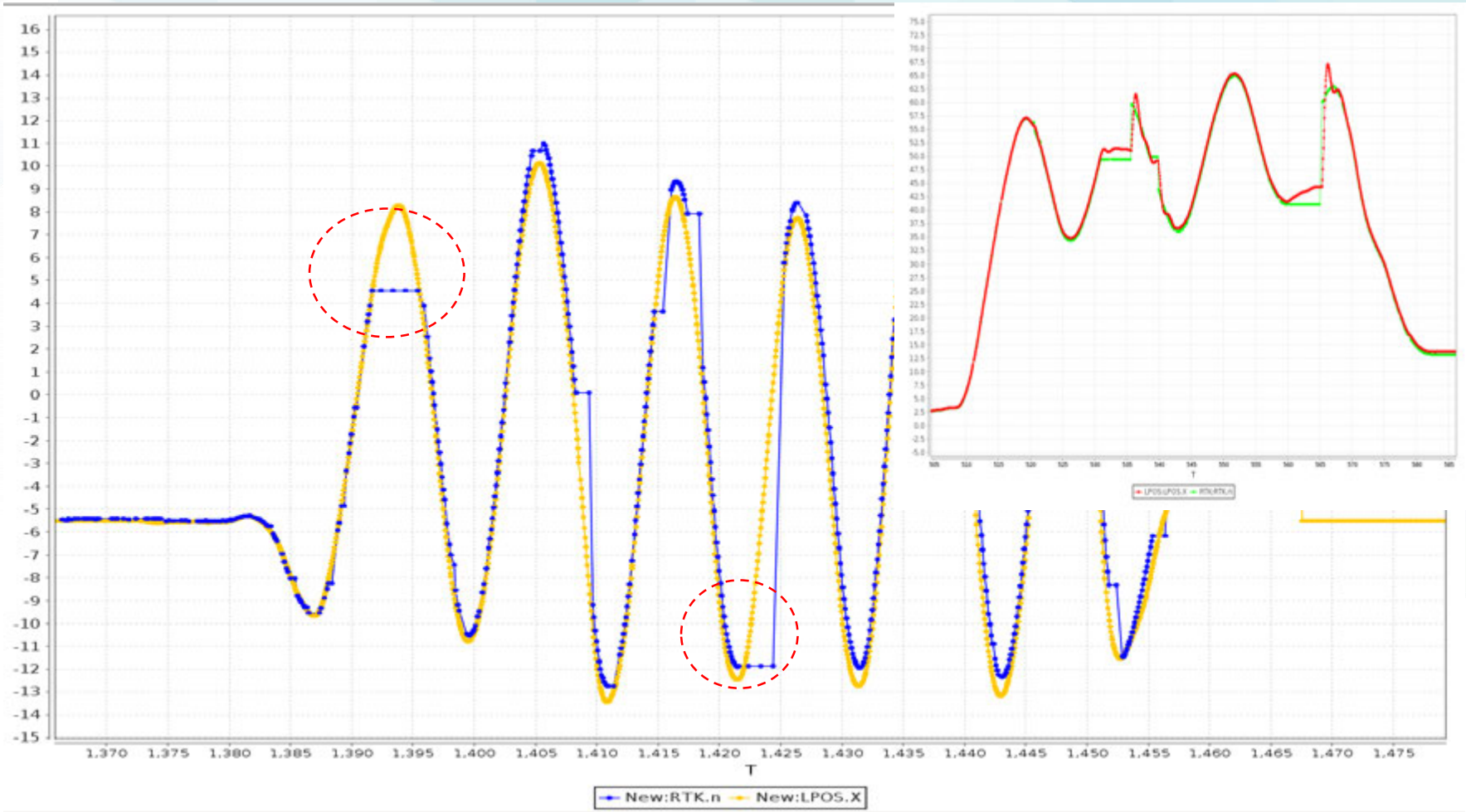
원 모양 군집 비행

RTK-GPS

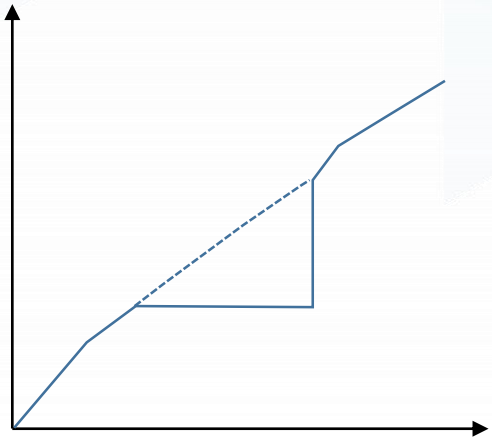
GPS-INS → RTK-GPS

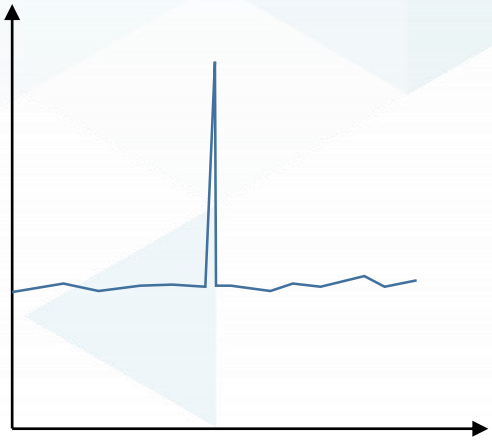
RTK-GPS

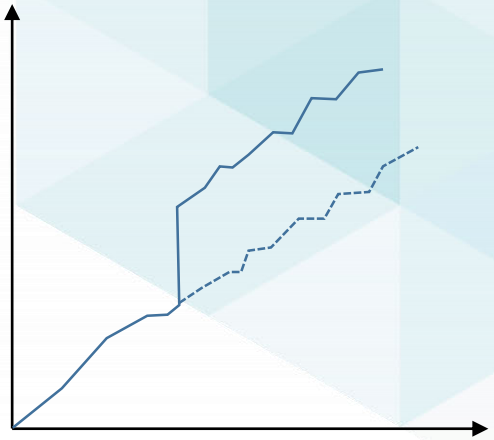
◆ Version Update



◆ Sensor Error (RTK-GPS)



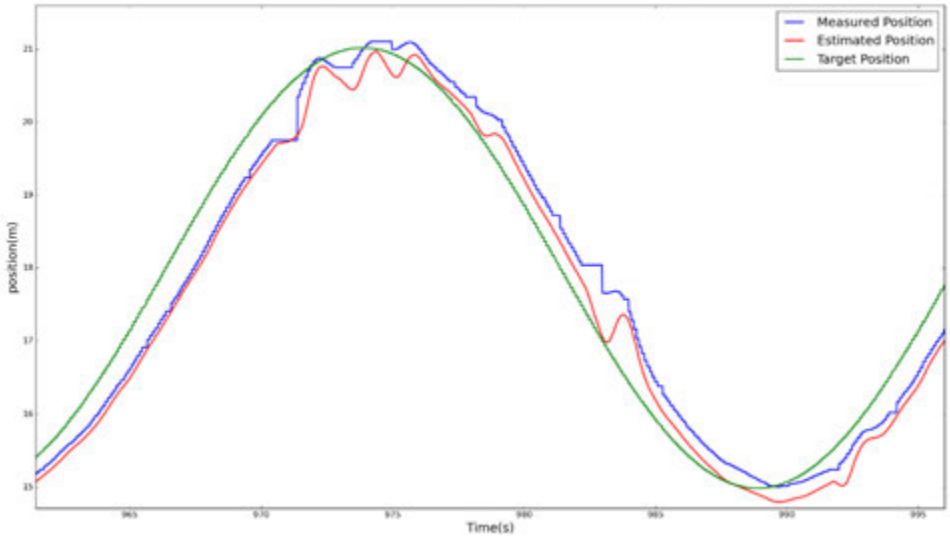




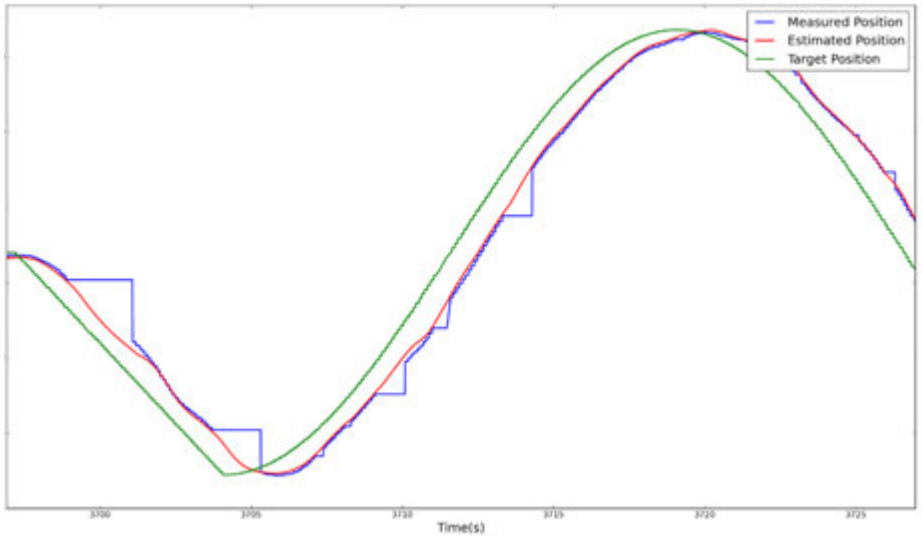
센서 모드 다중화

Communication	Position	Scenario
LTE	LPM	Embedded Scenario
	LiDAR	
Zigbee	RTK	
RC	GPS-INS	Barometer	Remote Scenario

Mode Switching Algorithm



[개선전]



[개선후]

실외 군집 비행 시험 결과

◆ 위치 정밀도 시험

실외 군집 비행 시험 결과

◆ 실외 군집 비행 시험 결과

[GPS-INS 위치 예측]

[Mode Switching 위치 예측]

실외 군집 비행 시험 결과

◆ 실외 군집 비행 시험 결과

실외 군집 비행 시험 결과

Outdoor

Swarming Flight

KARI

감사합니다

KARI 한국항공우주연구원
KOREA AEROSPACE RESEARCH INSTITUTE