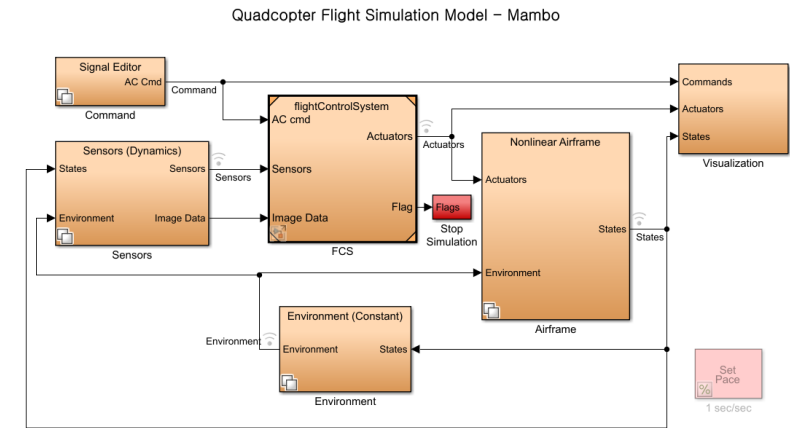
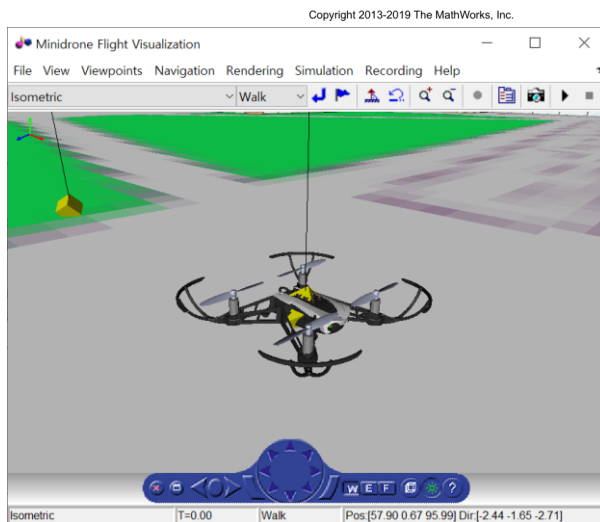


MATLAB, Simulink 드론 시뮬레이션 예제인 asbQuadcopter로 드론 제어
PARROT Mambo에 C코드 생성하여 실제로 비행도 가능



C Code
Generation



먼저 드론 모델 선정

논문 *Quadrotor control: modeling, nonlinear control design, and simulation*, FRANCESCO SABATINO

에 사용된 모델 사용

Linearization 하면 아래와 같다.

Disturbance는 d 매트릭스 처럼 알 수 없는 외부 힘 6개

이대로 사용한건 아니고 disturbance를 state로 추가하여 augmented matrix로 만듦

측정되는 신호는 roll, pitch, yaw, x, y, z로 하여 C matrix 구성

그리고 Discretization. 자세한건 LQR.m 참고

$$A = \frac{\partial f(x, u)}{\partial x} \bigg|_{\substack{x=\bar{x} \\ u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \frac{\partial f(x, u)}{\partial u} \bigg|_{\substack{x=\bar{x} \\ u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D = \frac{\partial f(x, u, d)}{\partial d} \bigg|_{\substack{x=\bar{x} \\ u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{I_z} \\ \frac{1}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$d = [f_{wx} \quad f_{wy} \quad f_{wz} \quad \tau_{wx} \quad \tau_{wy} \quad \tau_{wz}]^T$$

$$u = [f_t \quad \tau_x \quad \tau_y \quad \tau_z]^T$$

State space model 만들었으면 observer 설계
 Observer에 사용된 모델은 disturbance를 Augmented 시켜 state가 18개인 모델 사용
 Kalmanfilter 사용하여 state estimation 함

```
Ac = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1/Ixx 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1/Iyy 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1/Izz;
      0 -g 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1/mass 0 0 0 0;
      g 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1/mass 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1/mass 0 0 0;
      0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];

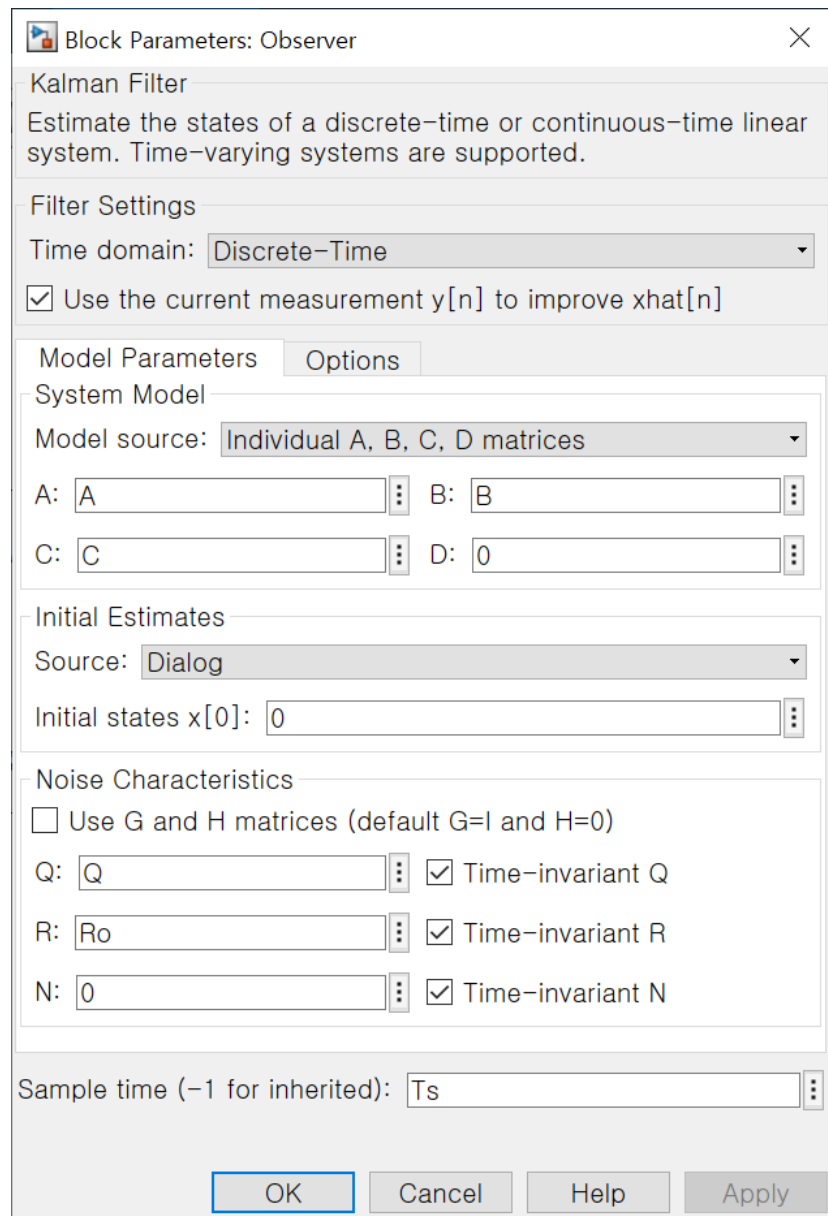
Bc = [0 0 0 0;
      0 0 0 0;
      0 0 0 0;
      0 1/Ixx 0 0;
      0 0 1/Iyy 0;
      0 0 0 1/Izz;
      0 0 0 0;
      0 0 0 0;
      -1/mass 0 0 0;
      0 0 0 0;
      0 0 0 0;
      0 0 0 0;
      0 0 0 0;
      0 0 0 0;
      0 0 0 0;
      0 0 0 0;
      0 0 0 0];

Cc = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0];
```

위의 Continuous model을 discretization 함

```
Dc = 0;
csys = ss(Ac, Bc, Cc, Dc);
dsys = c2d(csys, Ts);
```

Simulink Kalman filter block 사용



Block Parameters: Observer

Kalman Filter
Estimate the states of a discrete-time or continuous-time linear system. Time-varying systems are supported.

Filter Settings
Time domain: **Discrete-Time**
☒ Use the current measurement $y[n]$ to improve $\hat{x}[n]$

Model Parameters Options

System Model
Model source: **Individual A, B, C, D matrices**
A: B:
C: D:

Initial Estimates
Source: **Dialog**
Initial states $x[0]$:

Noise Characteristics
☐ Use G and H matrices (default $G=I$ and $H=0$)
Q: ☒ Time-invariant Q
R: ☒ Time-invariant R
N: ☒ Time-invariant N

Sample time (-1 for inherited):

OK Cancel Help Apply

시험하고, 신호 분석해서 튜닝
주로 Roll, Pitch torque disturbance를 빠르게 추종하도록 튜닝

```
Q(1,1) = 20;           Ro = eye(6);  
Q(2,2) = 20;  
Q(3,3) = 10;  
Q(7,7) = 10;  
Q(8,8) = 10;  
Q(9,9) = 10;  
Q(10,10) = 10;  
Q(11,11) = 10;  
Q(12,12) = 10;  
Q(13,13) = 200;  
Q(14,14) = 200;  
Q(15,15) = 1;  
Q(16,16) = 0.01;  
Q(17,17) = 0.01;  
Q(18,18) = 0.01;
```

제어

Non-zero set point

Feedforward control input을 구하고,
Feedback control input을 LQR로 구해보자
일단 제어모델을 만든다.

Observer모델의 disturbance가 아래와 같이 표현된다.

$$x_{k+1} = A\hat{x}_k + Bu_k + B_d\hat{d}_k$$

$$y_{k+1} = C\hat{x}_k$$

non-zero set point에서의 Steady-state일 때의 state X_s 를 구하고,
그때의 input U_s 를 구한다.

$$\begin{aligned} x_s &= A\hat{x}_s + Bu_s + B_d\hat{d}_k \\ r &= C\hat{x}_s \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} A - I & B \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} -B_d\hat{d}_k \\ r \end{bmatrix}$$

non-zero set point에서의 Steady-state일 때의 state X_s 를 구하고,
그때의 input U_s 를 구한다.

LQR로 feedback 게인 K를 구한다.
 제어에 사용된 모델은 disturbance를 뺀 모델이다.
 Disturbance는 Feedforward input인 U_s 에 반영된다.

$$A = \frac{\partial f(x, u)}{\partial x} \bigg|_{\substack{x=\bar{x} \\ u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \frac{\partial f(x, u)}{\partial u} \bigg|_{\substack{x=\bar{x} \\ u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

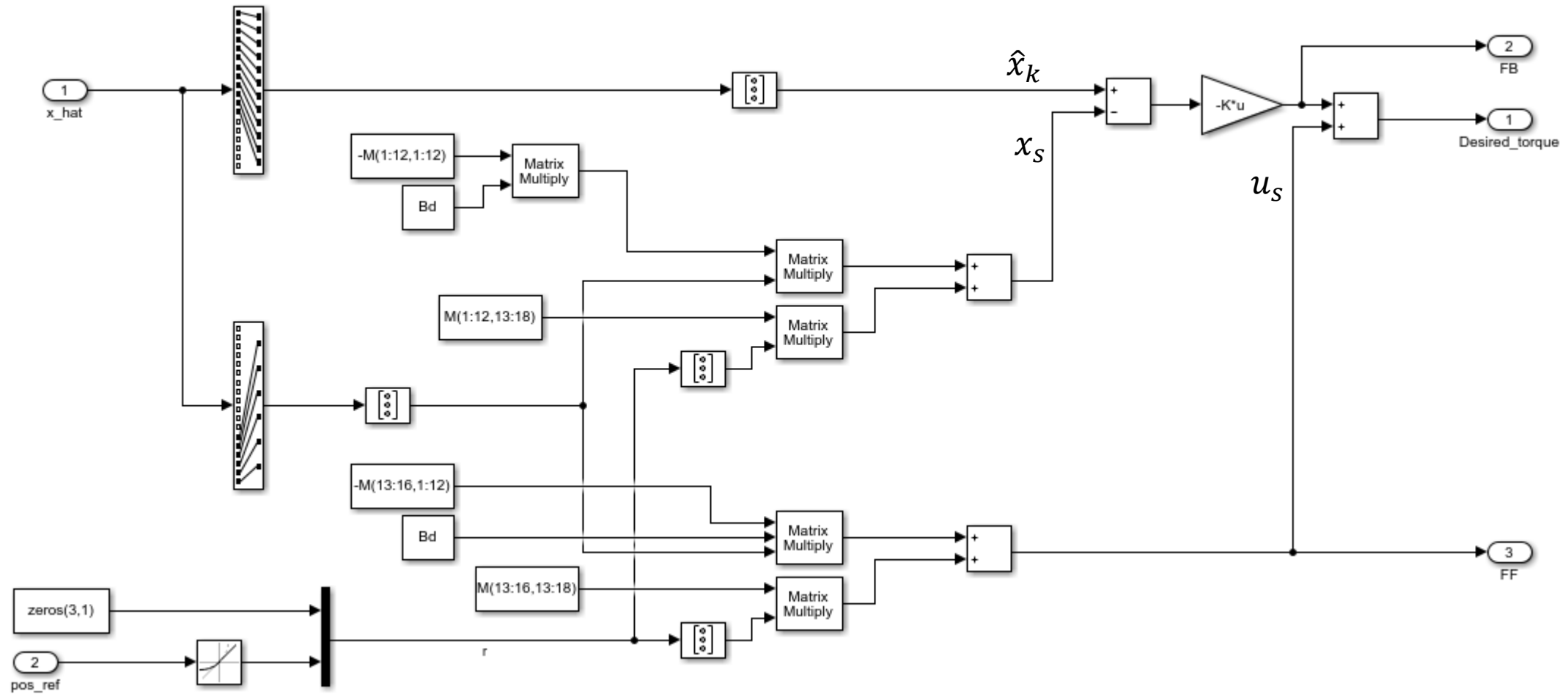
$$D = \frac{\partial f(x, u, d)}{\partial d} \bigg|_{\substack{x=\bar{x} \\ u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{I_z} \\ \frac{1}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$u = [f_t \quad \tau_x \quad \tau_y \quad \tau_z]^T$$

최종 제어 input은 아래와 같이 된다.

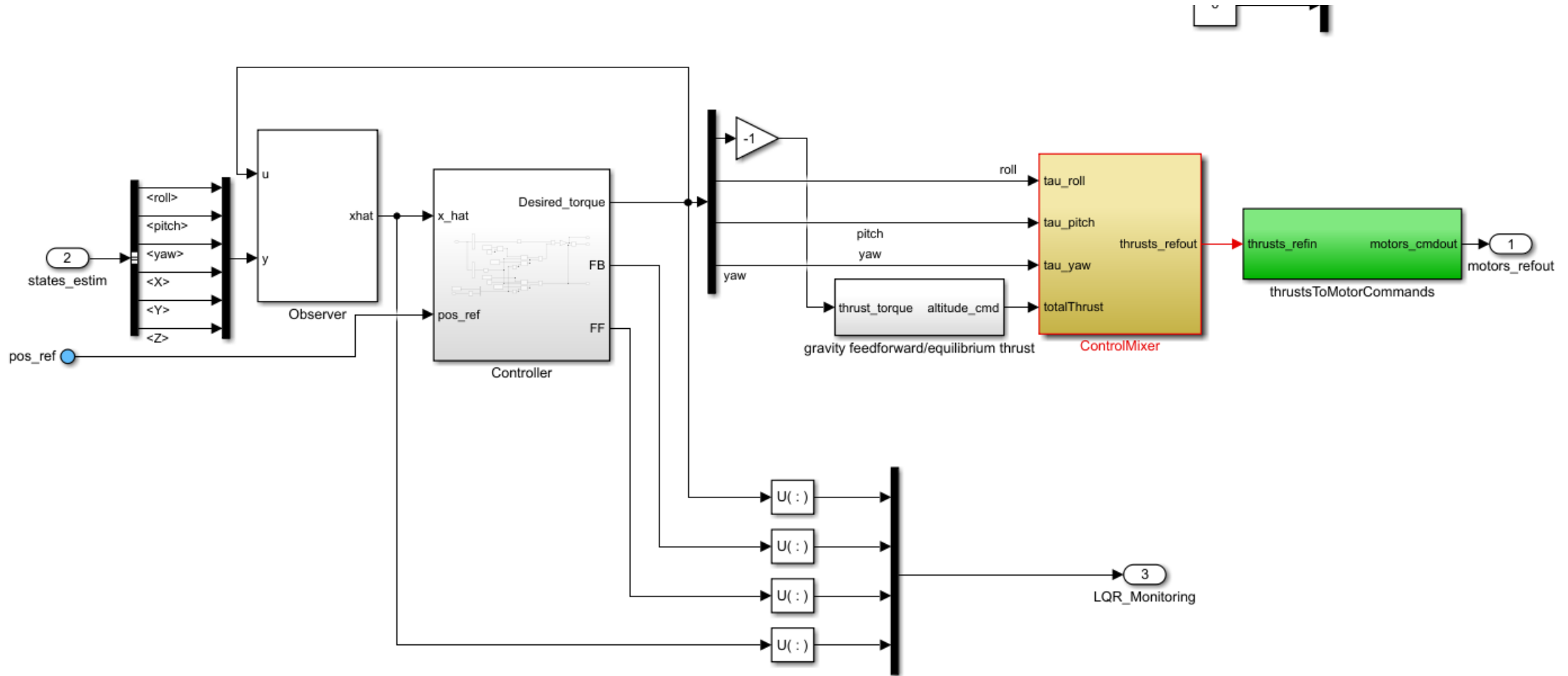
$$u_k = u_s - K(\hat{x}_k - x_s)$$

제어기를 Simulink 모델로 만들면~



전체 제어기는 이렇다.

아래의 LQR_Monitoring은 드론이 비행하는 동안 옱저버, 제어 신호를 저장한다.

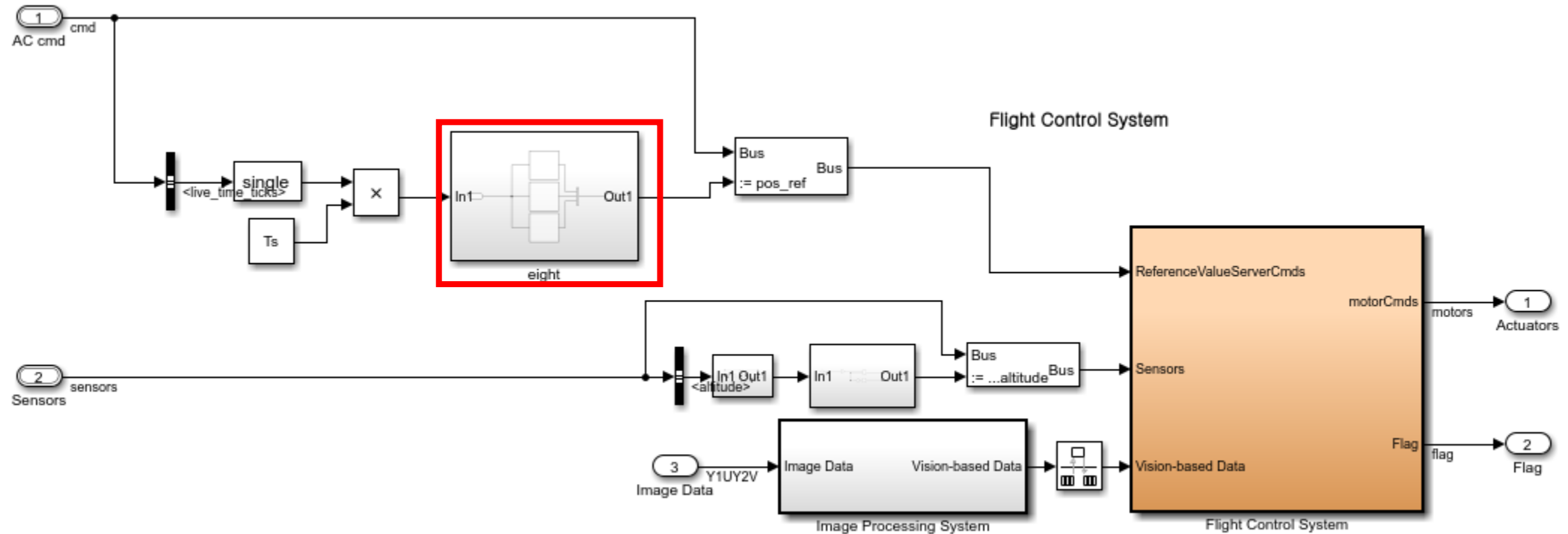


이제 다 되었지만...

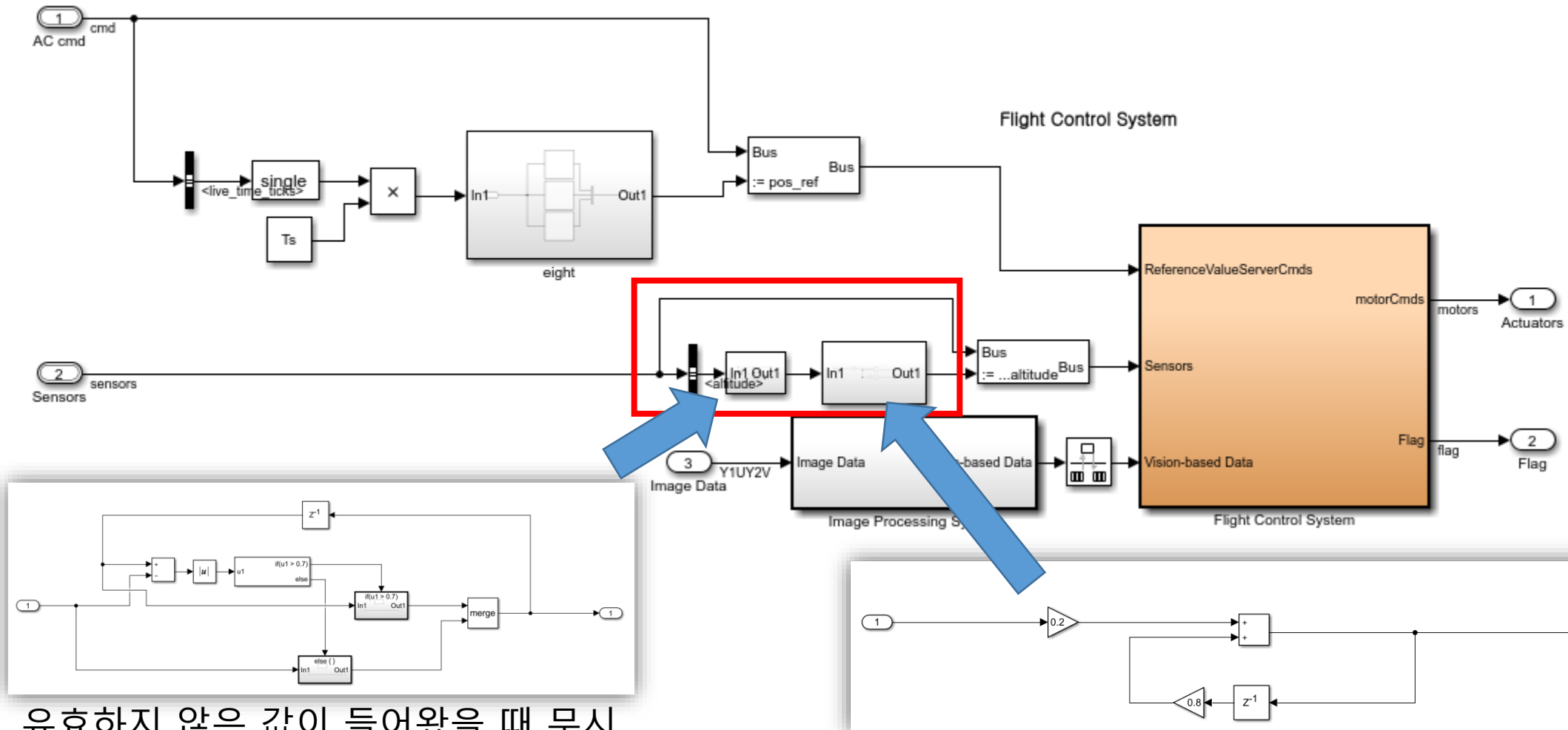
내가 원하는 건 드론이 날아서 hovering하다가 원하는 위치로 움직이는 것!

원하는 set point positio을 주기 위하여 아래 블록을 추가 하였다.

X, Y, Z 좌표를 맵으로 만든 것이다.



asbQuadcopter의 sensor fusion이 생각보다 성능이 좋지 않아
 간단하게 signal processing을 하였다.
 시간이 되면 sensor fusion도 직접 해볼 생각이다.

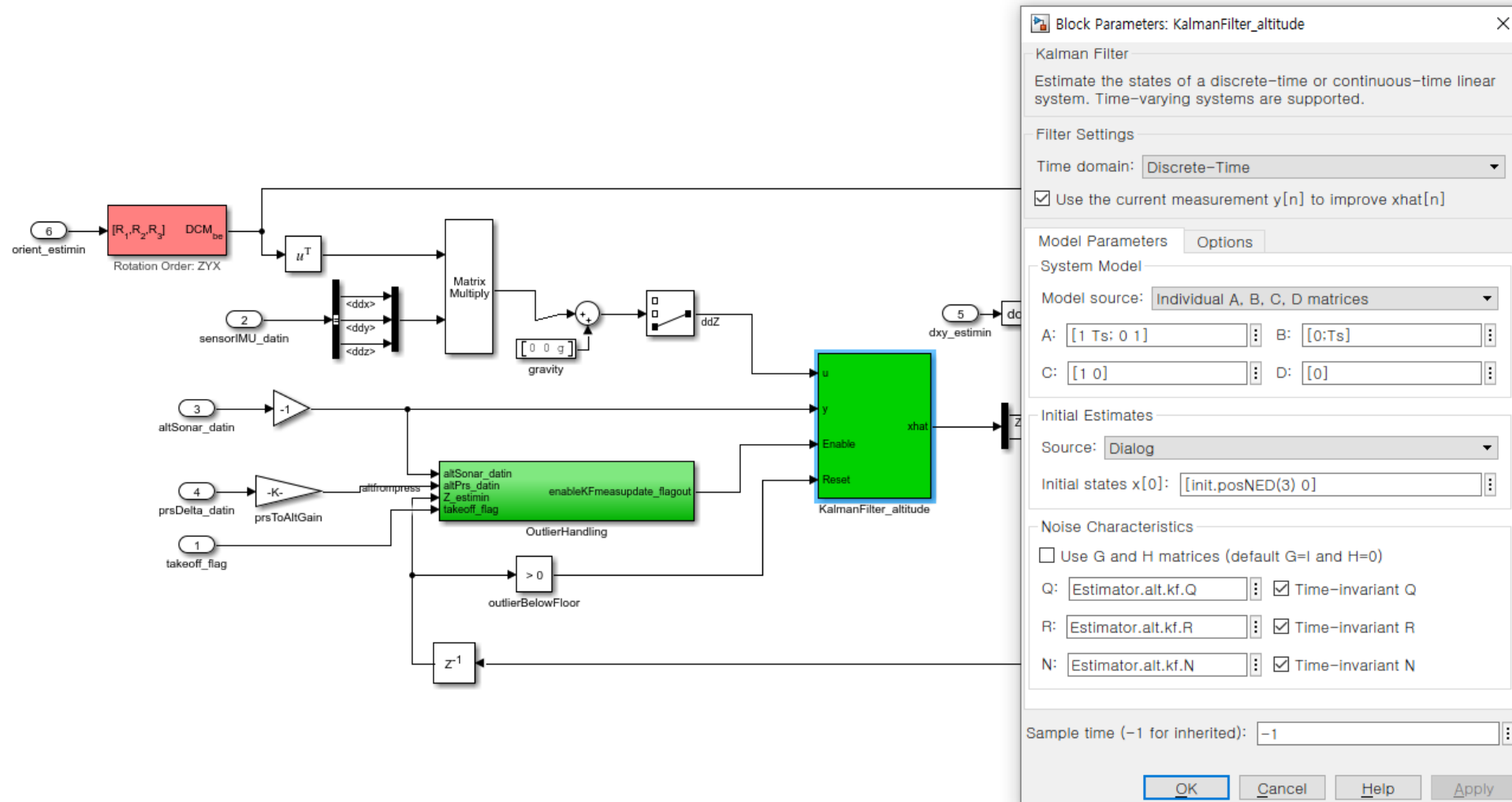


유효하지 않은 값이 들어왔을 때 무시
 좋은 방법은 아니다.

엄청 간단한 디지털 low pass filter

한가지 더 수정한건

asbQuadcopter/FCS (flightControlSystem)/Flight Control System/estimator (stateEstimator)/State Estimator/EstimatorAltitude에서
높이(z)를 적분하는 옵저버의 G, H matrix를 default값으로 하였다.



Github의 동영상을 보면 잘 된다.

튜닝이 너무 힘들었다.

궁금한게 있으면 hbkdk83@gmail.com으로