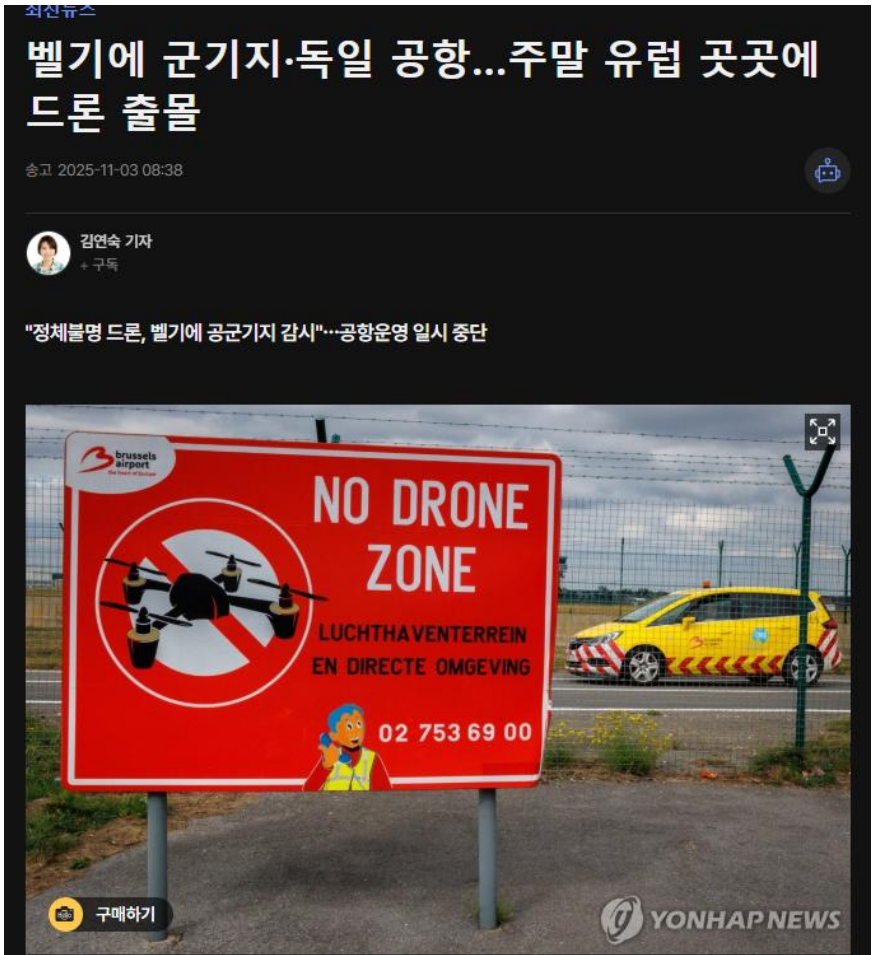
임베디드 프로세서2

실시간 3차원 전방 객체 탐지를 위한 3축 IMU 구형(Sphere) 짐벌(Gimbal) 제어

2021146036 최지헌

한국공학대학교
TECH UNIVERSITY OF KOREA

개발 이유



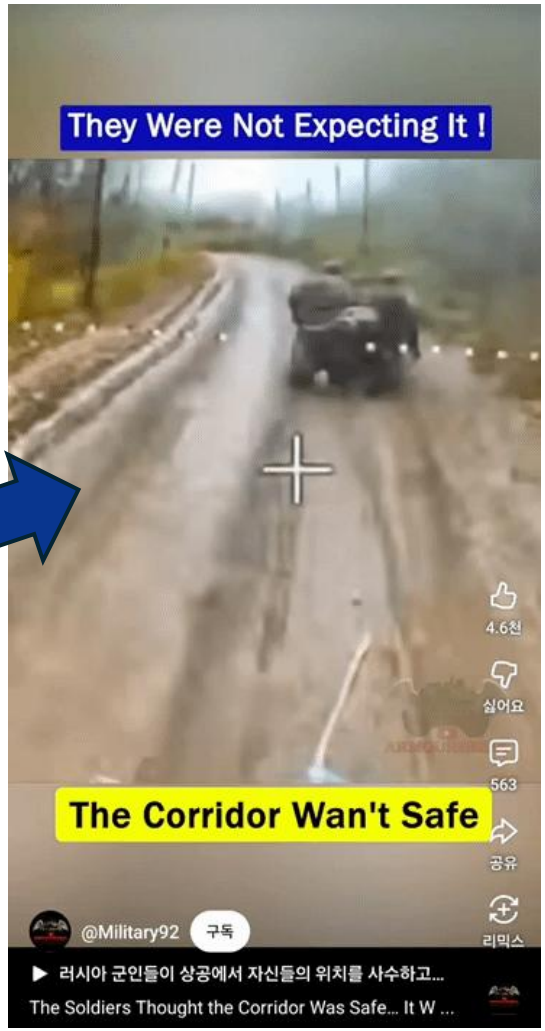
출처: 연합뉴스. 벨기에 군기지-독일 공항…주말 유럽 곳곳에 드론 출몰. 김연숙 기자 25.11.03.
https://www.yna.co.kr/view/AKR20251103021500009

출처: 서울신문. 멀리서도 고글 끼고 게임하듯 공격… "드론이 전쟁 양상 바꿨다". 윤창숙 기자 25.07.15.
https://www.seoul.co.kr/news/international/europe/2025/07/15/20250715014006

개발 이유

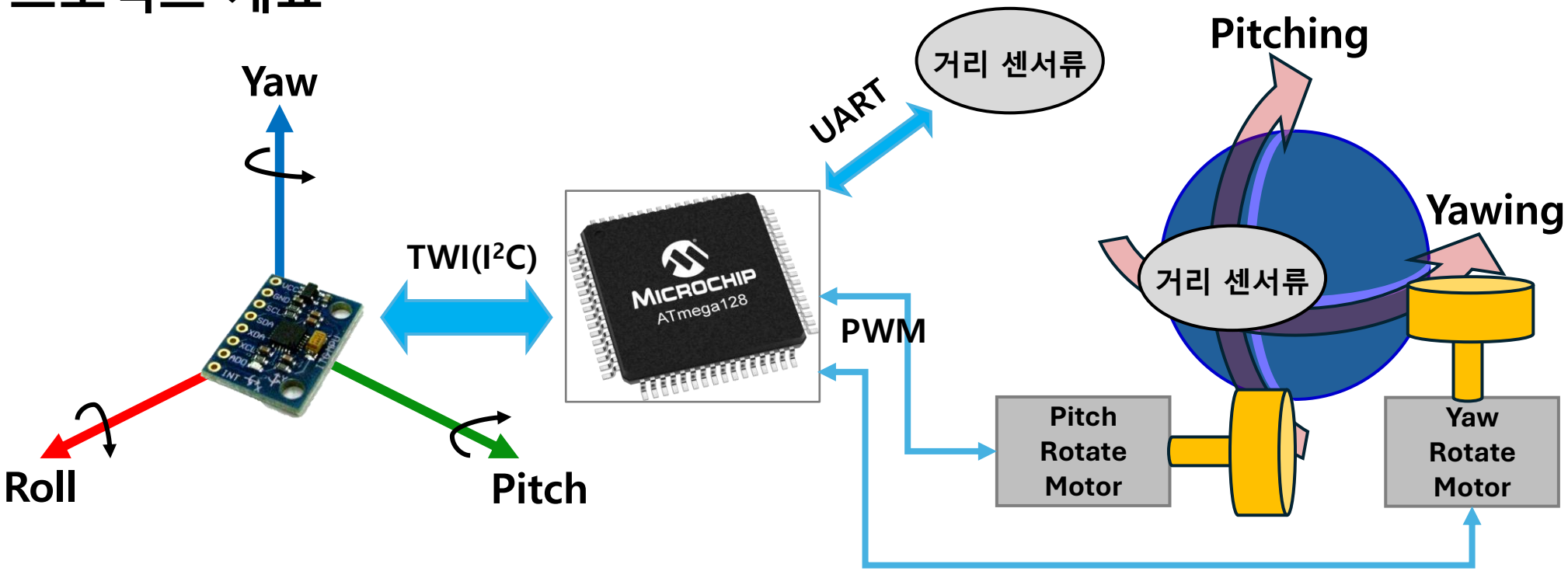
1. 드론전으로 전쟁 방식의 변화
2. 현대 Jammer는 사용자가 노출이 되어 있어서 매우 위험
3. 비행체 추적에 유리한 짐벌 제어가 필요함

안전하게 격추 할 수 있을까?



출처: youtube shorts. Military92

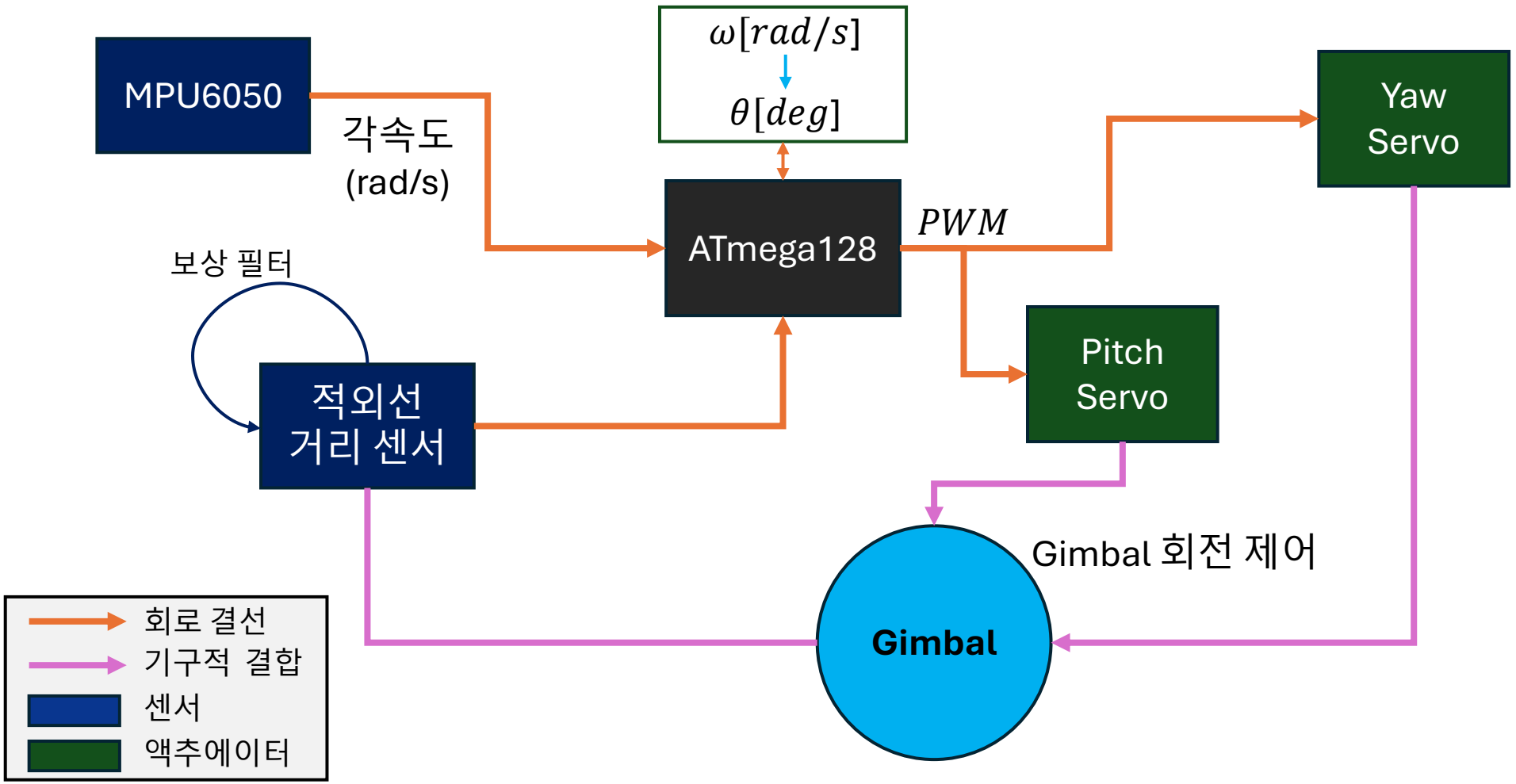
프로젝트 개요



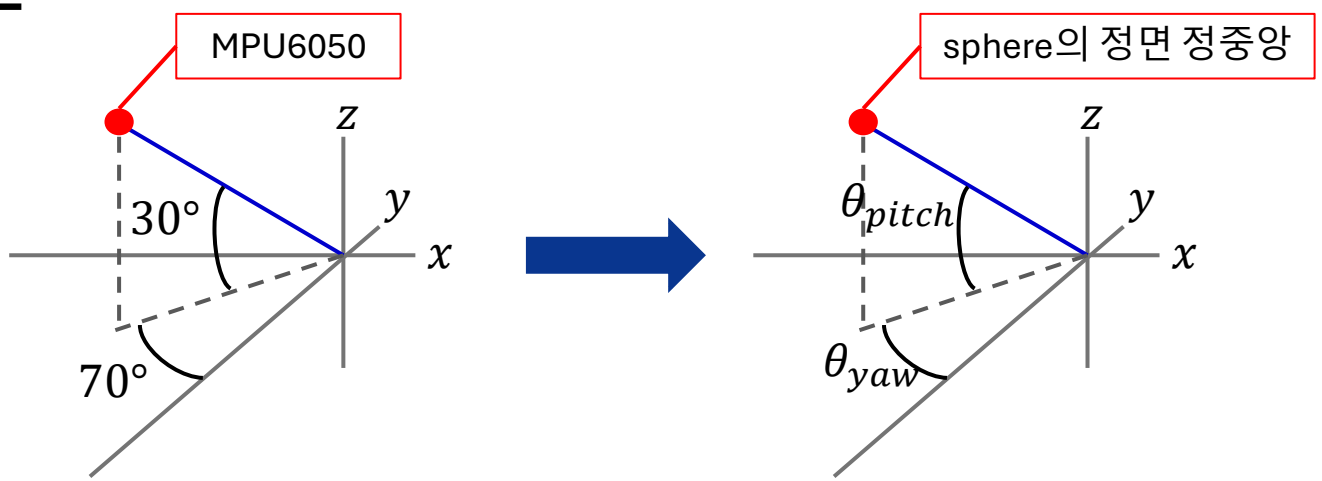
개발 목표

실시간으로 구(sphere) 정면에 위치한 거리 센서로 3차원 전방에 비행체를 탐지하도록 3축 IMU로 읽은 yaw, pitch로 구형 짐벌의 움직임을 제어하며 짐벌은 고각 0~80[deg], 방위각 -60~60[deg]의 커버리지를 갖는 시스템을 개발하고자 함

시스템 구성


MPU6050
각속도
(rad/s)
$\omega[rad/s]$
$\theta[deg]$
ATmega128
PWM
Yaw
Servo
보상 필터
적외선
거리 센서
Pitch
Servo
Gimbal 회전 제어
Gimbal
회로 결선
기구적 결합
센서
액추에이터

예시 시나리오



1. 컨트롤러(MPU6050)의 위치는 고정. 컨트롤러는 Yawing & Pitching 만 인식 됨
2. 컨트롤러의 움직임을 비례 이득 K_{yaw}, K_{pitch}를 곱하여 실제 컨트롤러 움직임에서 각 K배 만큼 Sphere를 움직임
3. K_{yaw}=0.75, K_{pitch}=1.3 일 때 컨트롤러가 Yaw 방향으로 70°, Pitch 방향으로 30° 움직였다면 Sphere의 정면 정중앙의 θ_{yaw}=52.5°, θ_{pitch}=39°이다.
4. sphere의 정면 정중앙에 부착된 적외선 거리 센서를 통해 전방에 물체가 있는지 없는지 판단

Scope Limitation

1. Sphere의 회전 각도 제한

→ IMU 360º 이상 회전해도 sphere는 360º 이상 회전하지 않음.

→ 중앙 정렬 기준 -60~60º(회전 각도 120º)로 움직임(Yaw, Pitch)

2. IMU와 Sphere의 회전비 설정

→ Sphere : IMU = 1 : 1.7

3. 물체 탐지 범위의 제한

→ 탐지 거리 30cm 이내에서의 장애물 판독

예상 문제

1. MPU6050의 누적 오차 발생

→ 정지해 있는 MPU6050에서 가속도가 있는 것처럼 센싱 되어 오차 값들이 읽힘

→ MPU6050이 움직이면 튀는 값이 발생함

→ MPU6050의 추정치와 실제 센싱되는 입력값을 칼만 필터의 칼만 이득으로 각 값들에 대한 신뢰도를 조절하여 정지/튀는 값 등에 대해서 잡아줌

→ 누적 오차로 인하여 재시작, calibration 필요

→ 일정 주기마다 calibration 되도록 함

예상 문제

1. MPU6050의 누적 오차 발생

→ 정지해 있는 MPU6050에서 가속도가 있는 것처럼 센싱 되어 오차 값들이 읽힘

→ MPU6050이 움직이면 튀는 값이 발생함

→ MPU6050의 추정치와 실제 센싱되는 입력값을 칼만 필터의 칼만 이득으로 각 값들에 대한 신뢰도를 조절하여 정지/튀는 값 등에 대해서 잡아줌

→ 누적 오차로 인하여 재시작, calibration 필요

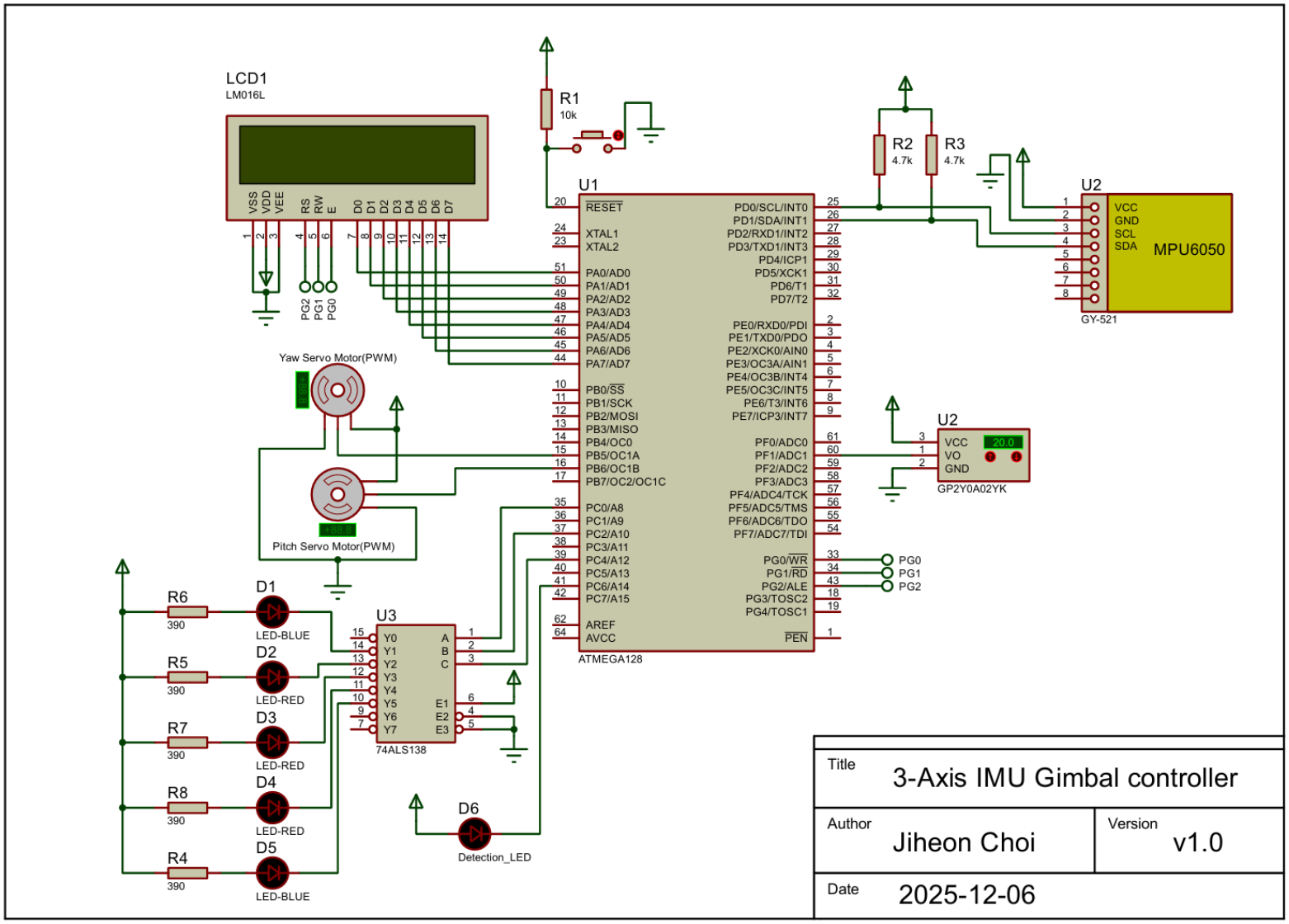
→ 일정 주기마다 calibration 되도록 함

예상 문제

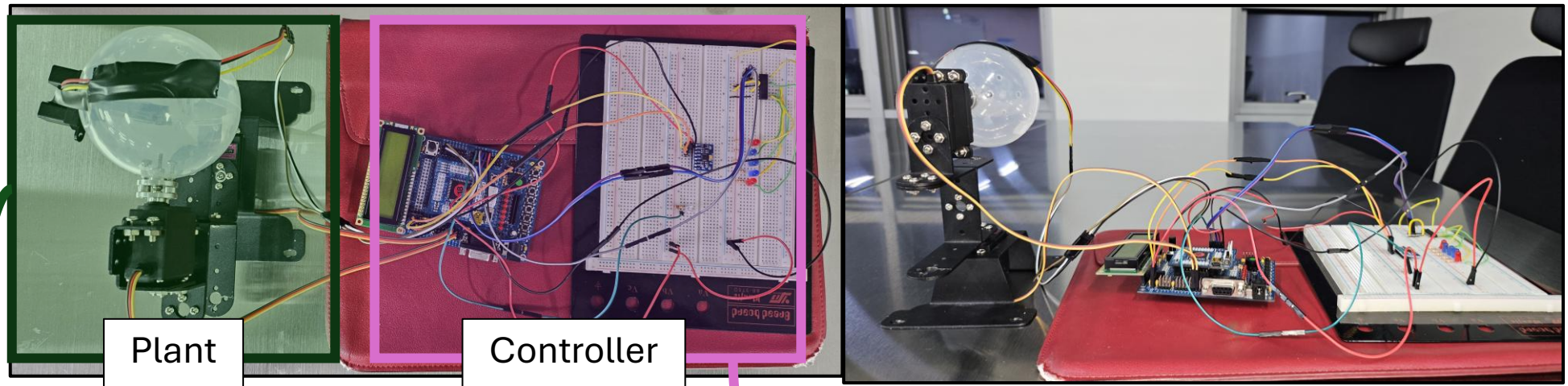
2. 모터의 각도 계산

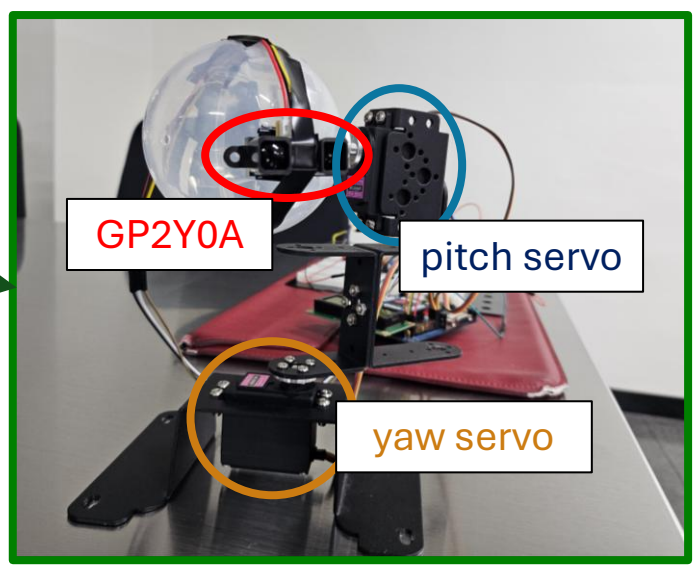
→ MPU6050의 움직임을 비례하여 움직인다 하더라도 서보모터가 MPU6050의 10°미만의 움직임을 제대로 추종할 수 없으므로 단위 각도를 생성하여 단위 각도 이상의 움직임 일 때 서보모터가 움직이도록 함

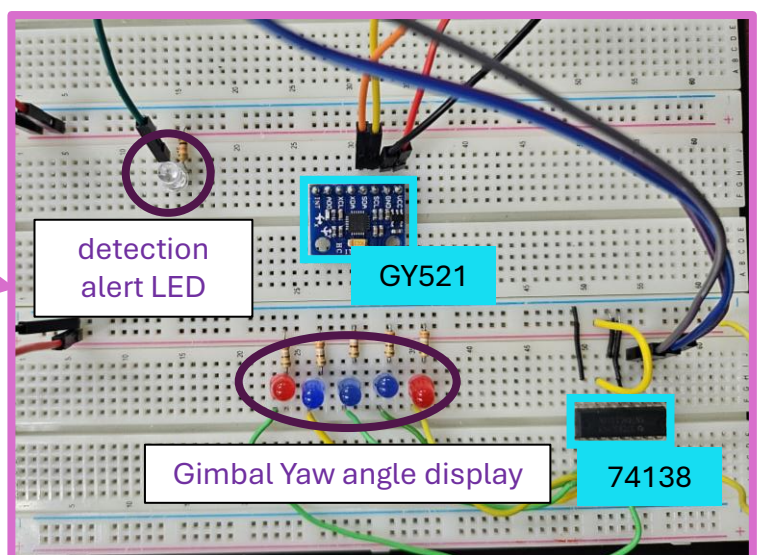
회로도


LCD1
LM016L
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7
PG2 PG1 PG0
R1
10k
R2
4.7k
R3
4.7k
U1
RESET
XTAL1
XTAL2
PA0/AD0
PA1/AD1
PA2/AD2
PA3/AD3
PA4/AD4
PA5/AD5
PA6/AD6
PA7/AD7
PB0/SS
PB1/SCK
PB2/MOSI
PB3/MISO
PB4/OC0
PB5/OC1A
PB6/OC1B
PB7/OC2/OC1C
PC0/A8
PC1/A9
PC2/A10
PC3/A11
PC4/A12
PC5/A13
PC6/A14
PC7/A15
AREF
AVCC
PD0/SCL/INT0
PD1/SDA/INT1
PD2/RXD1/INT2
PD3/TXD1/INT3
PD4/ICP1
PD5/XCK1
PD6/T1
PD7/T2
PE0/RXD0/PDI
PE1/TXD0/PDO
PE2/XCK0/AIN0
PE3/OC3A/AIN1
PE4/OC3B/INT4
PE5/OC3C/INT5
PE6/T3/INT6
PE7/ICP3/INT7
PF0/ADC0
PF1/ADC1
PF2/ADC2
PF3/ADC3
PF4/ADC4/TCK
PF5/ADC5/TMS
PF6/ADC6/TDO
PF7/ADC7/TDI
PG0/WR
PG1/RD
PG2/ALE
PG3/TOSC2
PG4/TOSC1
PEN
ATMEGA128
U2
VCC
GND
SCL
SDA
MPU6050
GY-521
Yaw Servo Motor(PWM)
Pitch Servo Motor(PWM)
U2
VCC
VO
GND
20.0
GP2Y0A02YK
R6
390
D1
LED-BLUE
R5
390
D2
LED-RED
R7
390
D3
LED-RED
R8
390
D4
LED-RED
R4
390
D5
LED-BLUE
U3
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
A B C
E1 E2 E3
74ALS138
D6
Detection_LED
Title
3-Axis IMU Gimbal controller
Author
Jiheon Choi
Version
v1.0
Date
2025-12-06

하드웨어 구성


Plant
Controller


GP2Y0A
pitch servo
yaw servo


detection
alert LED
GY521
Gimbal Yaw angle display
74138

수행 내용

계획 변경 사항

변경 전	변경 후
짐벌 고각(pitch), 방위각(yaw) -60~60º 제한	짐벌 고각 0~80º, 방위각 -60~60º
회전비 설정 Gimbal : IMU = 1 : 1.7	yaw. Gimbal : IMU = 0.5:1, pitch. Gimbal : IMU = 1:1
물체 탐지 거리 30cm 이내 제한	물체 탐지 거리 25~130cm
거리 센서 UART 통신	거리 센서 ADC
짐벌을 바퀴로 간접 제어	짐벌을 모터와 직접 연결해서 직접 제어

주요 핵심 기술

- 16x2 LCD제어
- PORTC로 3bit 이진수 & 단일 LED 제어
- Timer1 FastPWM Servo motor 제어
- I2C(TWI) 통신으로 Atmega128 <-> MPU6050
- Timer3 CTC 100Hz MPU6050 read
- MahonyAHRS PI제어 read mpu6050 알고리즘
- 각도 -> OCR 변환

$$OCR = OCR_{\min} + \frac{\deg}{180.0^{\circ}}(OCR_{\max} - OCR_{\min})$$

- ADC로 GP2Y0A의 output voltage -> 거리 L(cm) 변환

Algorithm: MahonyAHRS

input: gx, gy, gz, ax, ay, az, Current Quarternion:(q0, q1, q2, q3)
output: Qtn_filter

Initialize: $q0 \leftarrow 1.0, q1 \leftarrow 0.0, q2 \leftarrow 0.0, q3 \leftarrow 0.0,\ FBx = FBy = FBz \leftarrow 0$

$\mathbf{Gyro} \leftarrow \begin{bmatrix} gx \\ gy \\ gz \end{bmatrix}, \mathbf{Accel} \leftarrow \begin{bmatrix} ax \\ ay \\ az \end{bmatrix}, \mathbf{Qtn} \leftarrow \begin{bmatrix} q0 \\ q1 \\ q2 \\ q3 \end{bmatrix}, \mathbf{Ifb} \leftarrow \begin{bmatrix} FBx \\ FBy \\ FBz \end{bmatrix} \{integral\ feedback\}$

if ax≠0 and ay≠0 az≠0 **then**

recipNorm $\leftarrow 1/\sqrt{ax^2 + ay^2 + az^2}$

$\mathbf{Accel} \leftarrow \mathbf{Accel} \times$ recipNorm

$\mathbf{V_{half}} \leftarrow \begin{bmatrix} vx \\ vy \\ vz \end{bmatrix} = \begin{bmatrix} q1 * q3 - q0 * q2 \\ q0 * q1 + q2 * q3 \\ q0 * q0 - 0.5 + q3 * q3 \end{bmatrix}$

$\mathbf{Error_{half}} \leftarrow \begin{bmatrix} ex \\ ey \\ ez \end{bmatrix} = \begin{bmatrix} ay * vz - az * vy \\ az * vx - ax * vz \\ ax * vy - ay * vx \end{bmatrix}$

if $2K_i > 0.0$ **then**

$\mathbf{Ifb} \leftarrow \begin{bmatrix} FBx \\ FBy \\ FBz \end{bmatrix} = \begin{bmatrix} FBx \\ FBy \\ FBz \end{bmatrix} + \begin{bmatrix} 2K_i\Delta t & 0 & 0 \\ 0 & 2K_i\Delta t & 0 \\ 0 & 0 & 2K_i\Delta t \end{bmatrix} \mathbf{Error_{half}}$

$\mathbf{Gyro} \leftarrow \mathbf{Gyro} + \mathbf{Error_{half}}$

else

$\mathbf{Error_{half}} \leftarrow 0$

end

$\mathbf{Gyro} \leftarrow \mathbf{Gyro} + 2K_p \cdot \mathbf{V_{half}}$

end

$\mathbf{Gyro} \leftarrow 0.5\Delta\mathbf{t} \cdot \mathbf{Gyro}$

$\mathbf{Qtn} \leftarrow \mathbf{Qtn} + \begin{bmatrix} 0 & -gx & -gy & -gz \\ gx & 0 & gz & -gy \\ gy & -gz & 0 & gx \\ gz & gy & -gx & 0 \end{bmatrix} \mathbf{Qtn}$

recipNorm $\leftarrow 1/\sqrt{q0^2 + q1^2 + q2^2 + q3^2}$

$\mathbf{Qtn} \leftarrow$ recipNorm $\cdot \mathbf{Qtn}$

Qtn filter = **Qtn**

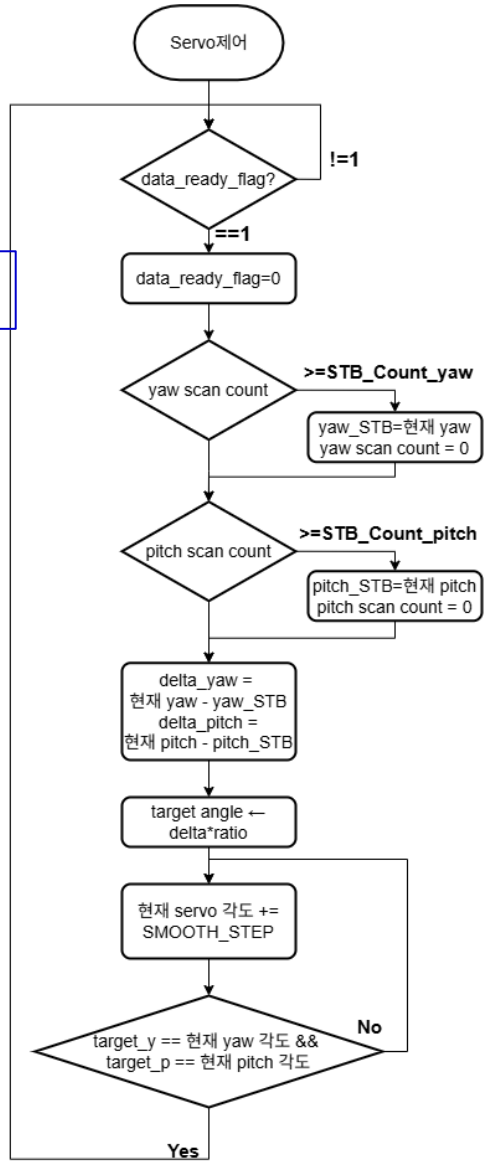
Servo motor

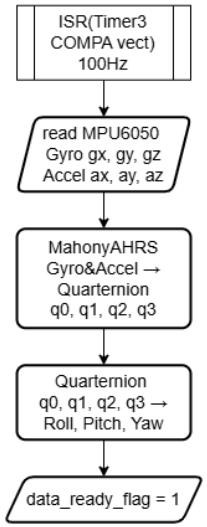
1. 0.01초 마다 timer3 비교일치 인터럽트 발생
2. 인터럽트 서비스 루틴에서 MPU6050의 gyro, accel 읽음

안정적인 Euler각 변환 method

3. MahonyAHRS 알고리즘으로 gyro, accel → Quarternion 변환 → Euler 각 변환
4. data 준비 플래그를 1로 세트하고 main 복귀
5. yaw, pitch 읽은 횟수가 STB_count 이상일 경우 Stabilize를 현재 각도로 업데이트
6. 현재 각도와 STB의 차이를 일정 비율을 곱해서 servo target angle을 생성
7. servo motor가 부드럽게 움직이는 step 각도를 계속 더해서 target 각도까지 움직임

예상 문제 MPU6050 누적오차(Drift)가 해결 됨!





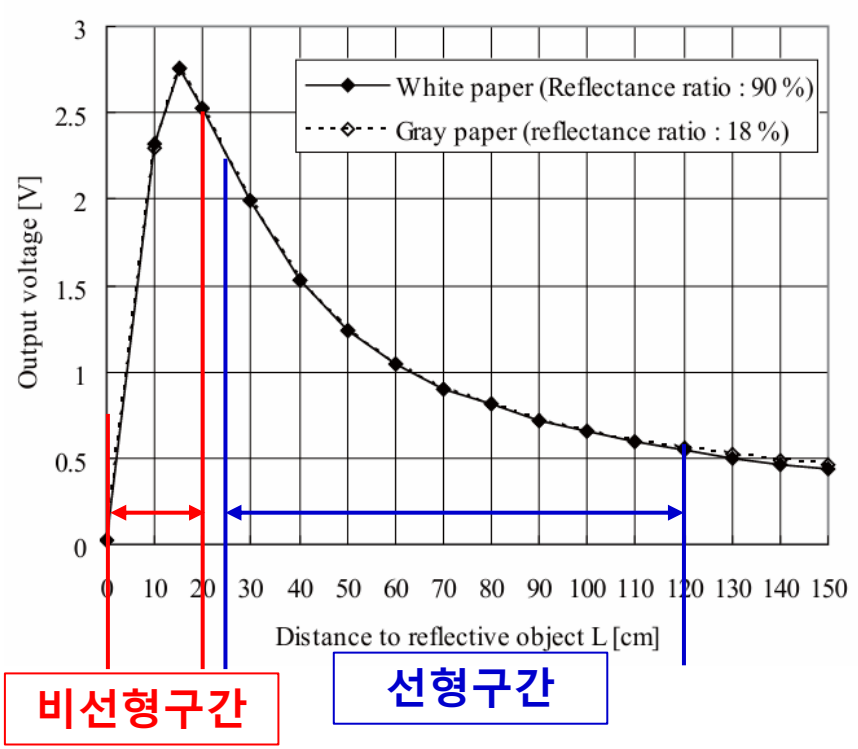
적외선 거리센서(GP2Y0A)

$$V_{out} = A * \frac{1}{L} + B$$

$$A = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{2.25 - 0.6}{\left(\frac{1}{25}\right) - \left(\frac{1}{100}\right)} = 55$$

$$B = y_1 - Ax_1 = 1.7$$

$$L = \frac{A}{V_{out} - B}[\text{cm}]$$

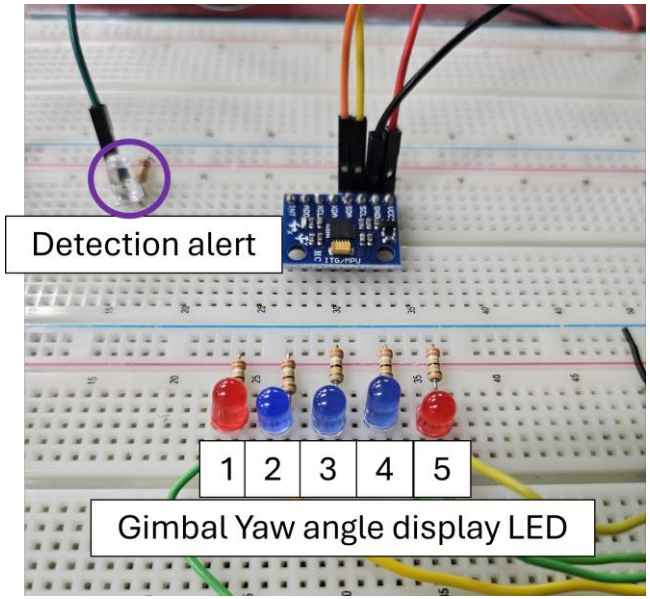


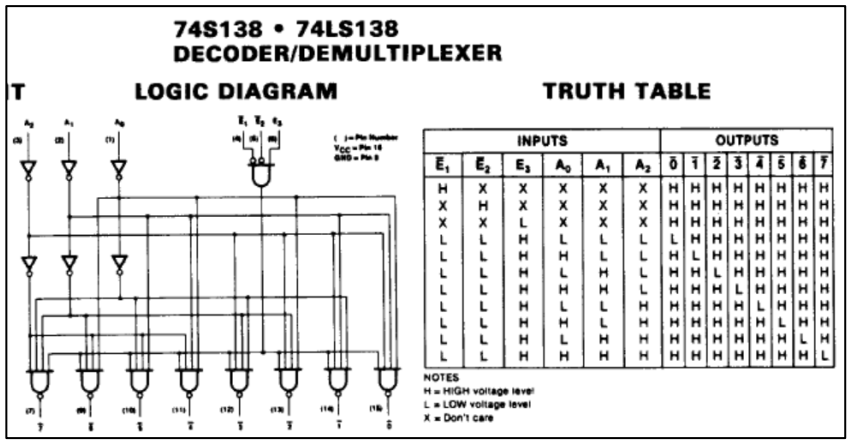
적외선거리센서에 비선형 구간이 포함되기 때문에 0~20cm까지 근접 거리 측정 불가

선형성이 있는 25~120cm까지만 사용

실제 시스템은 수백m~수km정도의 범위이므로 시연용 거리이다.

LED Interface




74S138 • 74LS138
DECODER/DEMULTIPLEXER

LOGIC DIAGRAM

TRUTH TABLE

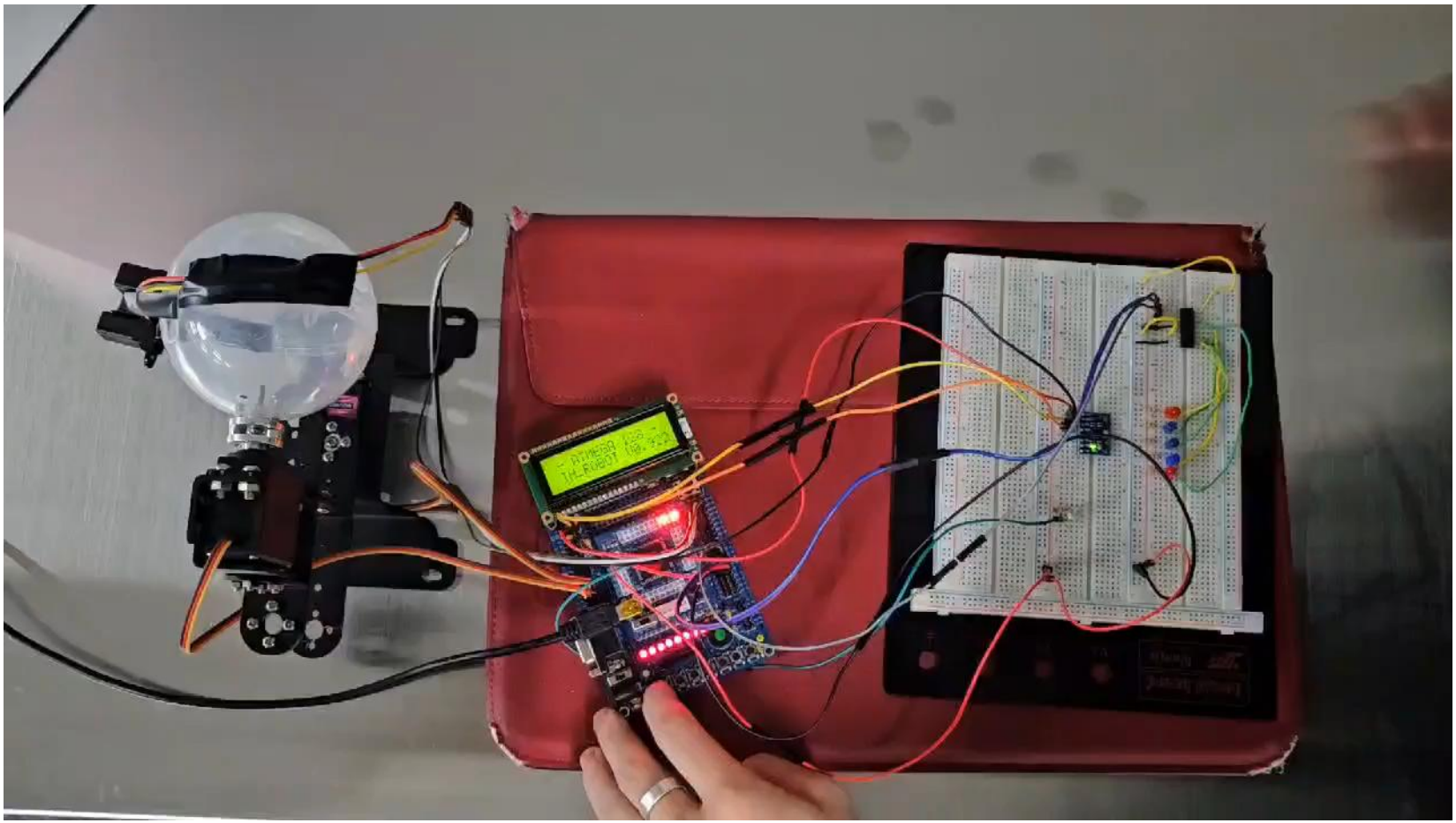
INPUTS						OUTPUTS						
E_1	$\overline{E}_2$	$\overline{E}_3$	A_0	A_1	A_2	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

NOTES
H = HIGH voltage level
L = LOW voltage level
X = Don't care

- detection alert LED는 적외선 거리센서의 거리 L[cm]가 50cm 미만일 때 LED를 점등함
- angle display LED는 gimbal의 Yaw 각도에 따라서 1~5가 각각 따로 점등한다.
- 더 적은 핀을 사용하기 위해 Decoder IC(74138)과 PORTC를 연결해서 사용

범위	C (PC4)	B (PC2)	A (PC0)	점등 LED
-60°~-40°	L	L	H	**1**
-39°~-20°	L	H	L	**2**
-19°~20°	L	H	H	**3**
21°~-40°	H	L	L	**4**
41°~60°	H	L	H	**5**

동작 영상



결론

- 수행 목록표에 나온대로 모든 목표를 달성했음
- 서보모터를 연속 시간으로 제어하려고 하다 보니 갑자기 변화하는 각도에 대해서 대응하기 위해 서보모터가 계단식으로 움직임, 멀티 프로세스나 멀티 스레드를 추가했다면 IMU를 읽는 스레드 따로 Servo 움직이는 스레드 따로 나눠서 목표 각도로 움직이는 중에도 다른 목표 각도로 움직이도록 변경할 수 있음
- 짐벌은 다양한 분야에서 사용됨 확장성이 있음
- 실시간 제어는 많은 연산량을 요구함 디테일하게 하고 싶으면 성능이 좋은 프로세서를 사용해야 함

Q&A