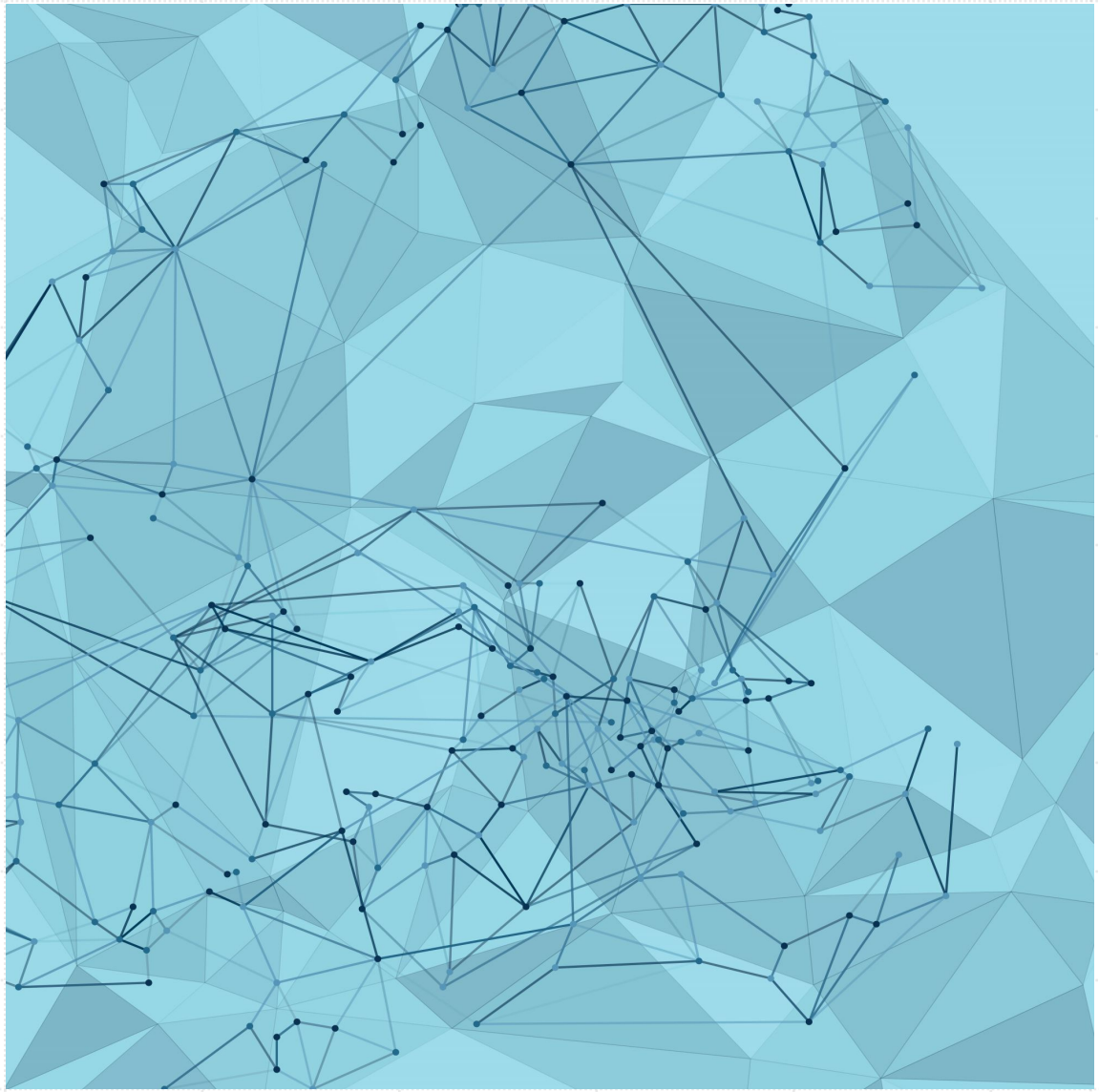
RaspberryPi를 활용한 광추적형 센싱 기술 구현하기



- 4조

202022311 곽희상

202126894 박승연

202126879 봉지수

(Project Goal)
프로젝트 목표

+ C programming







Light-Sensing System



(Analysis)
요구사항 분석

센서: 조도 센서

① SPI reference를 활용하여 조도 센서 값 읽기

② 8개의 센서 값 취합 및 xy-좌표평면 위의 8개 위치별로 값을 출력

연산: 광 추적 알고리즘

① 8군데 중 가장 빛이 많이 들어오는 지점을 찾는 데에 사용

② 해당 방향으로의 이동을 위한 정보를 표시

액추에이터: 서보 모터

① 모터 2개에 수직 이동 또는 수평 이동의 역할 부여

② 부여된 역할을 수행하도록 방향 및 상태를 조절하여 설치

소켓 통신

① 소켓 통신으로 센서와 연결된 2개의 라즈베리파이의 조도 센서 값 송수신

② 소켓 통신으로 3대의 라즈베리파이 연결 및 정보 주고받기

스레드

① 스레드로 엑츄에이터와 연결된 라즈베리파이가 값을 읽는 동시에 모터를 동작시키도록 구현

(Analysis)
요구사항 분석

- 센서: 조도 센서

  - 아날로그 값(빛의 세기)을 0~1023 사이의 디지털 숫자로 출력
    -→ ADC가 없는 라즈베리파이는 입력을 받은 후에 내부 프로그램을 통해 아날로그-디지털 변환 소자 연결 작업 필요

  - 빛의 양이 너무 많이 흐르게 되면 저항이 작아져 과전류가 흐르게 되므로 주의

  - 과제) 8개의 조도 센서로 8개의 디지털 input을 받아서 출력

- 엑츄에이터: 서보 모터(sg90)

  - 작동 전압: 4-7.2V (최소한 5V 이상의 전지를 사용하는 것을 권장)

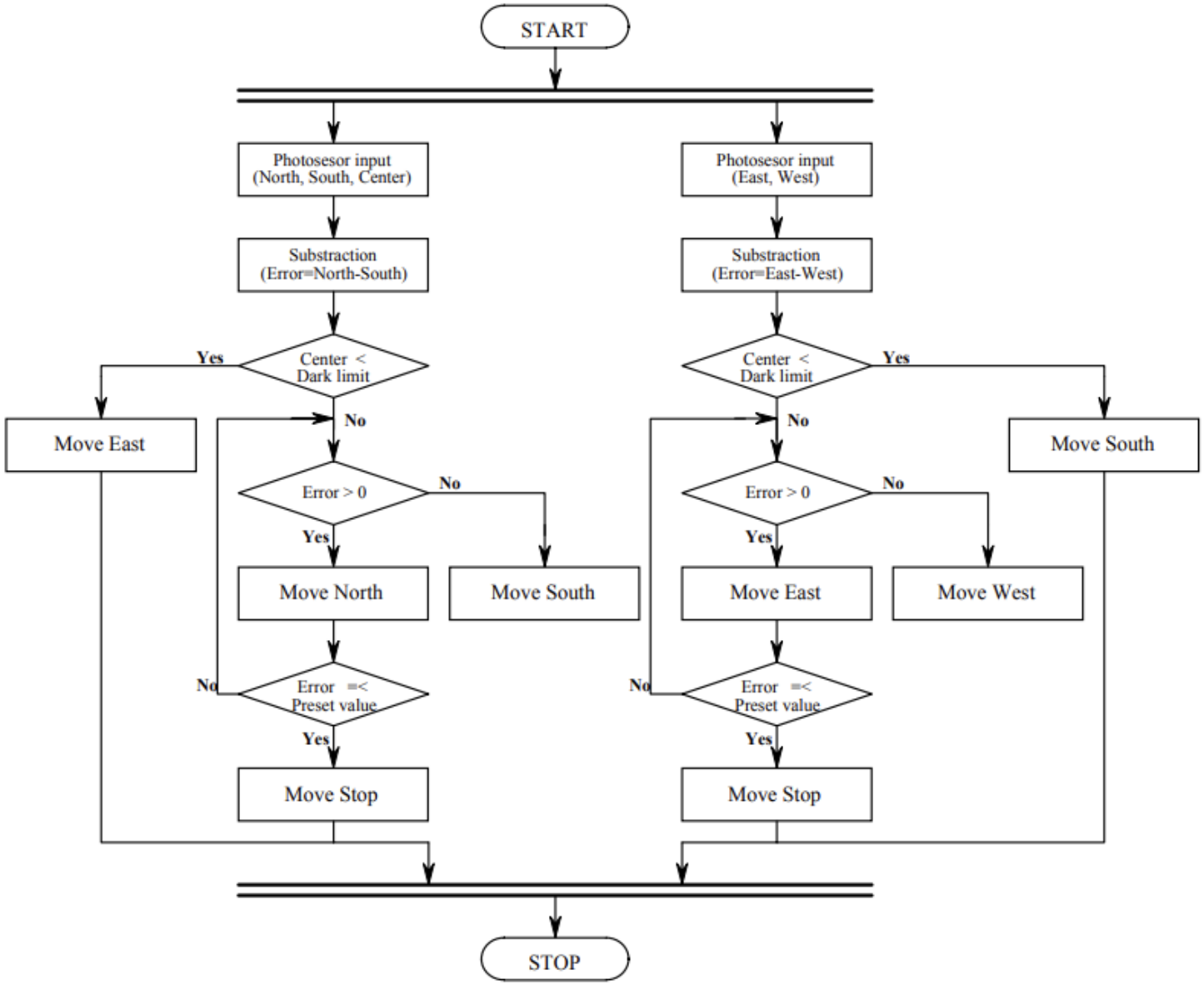
  - 190°도까지도 회전은 가능, 다만 고장에 유의 (최대 회전각 180°으로 설정)

  - 라즈베리파이는 5V와 3.3V 핀이 있는데 오랜 시간 모터 가동으로 인한 과열로 라즈베리파이가 손상될 수 있음

  - 과제) 2개의 서보 모터로 목표 (x,y) 방향으로 각도 조절

(Analysis)
요구사항 분석

- 연산: 광 추적 알고리즘



최정식·고재섭·정동화, “태양광 발전의 효율 향상을 위한 자동추적 제어 알고리즘 개발”, 전기학회논문지, 59권 10호 2010년 10월, 1827.

(Analysis)
요구사항 분석

- 소켓 통신

  - 로컬 네트워크를 공유하여 라즈베리파이 간에 socket을 주고 받는 통신 방법
  - 무선 또는 유선 방식으로 공유기에 라즈베리파이를 연결

  - UDP 통신이 간편하고 속도가 빠르기 때문에 고려하였으나 신뢰도가 높은 *TCP* 방식으로 최종 구현

  - 과제) 2대~3대의 라즈베리파이 간에 소켓 통신으로 정보 송수신
    ---> *3*대 모두 연결 *(TCP)*

- 스레드

  - 스레드는 프로세스 내에서 실행되는 흐름의 단위 (하나의 프로그램 – 하나의 스레드가 보편적)

  - 프로그램에 따라 둘 이상의 스레드를 동시에 실행할 수 있음 = multithread

  - 과제) 조도 센서 당 스레드 하나를 생성하여 multithread 방식으로 동시에 8개의 센서 값을 동시에 측정
    ---> 조도 센서 *4*개의 값을 하나의 *thread*로 읽어오기*(2*회*),* 이후 연산한 결과를 *thread*로 전달,
    이 값을 모터가 읽는 동시에 동작을 위한 *thread* 생성

기존의 (Limits & Obstacles) 도전적 이슈 및 제한사항

양방향 추적식 태양광 발전의 특징에 의한 제한 사항

태양 이외의 다른 종류의 빛을 인식

조도 센서의 인식 가능 범위의 한계

계절별 일조량 차이에 의한 임계치(tolerance) 수정 필요

조도 센서 사이에 가림막 설치 하여 문제 해결

변경된 (Limits & Obstacles) 도전적 이슈 및 제한사항

PWM 코드와 소켓 통신 코드 간 잦은 충돌로 인한 과부하

PWM으로 특정 조건일 때, 정해준 각도만큼 소폭 이동하는 기능 테스트 실패

라즈베리파이 3대 소켓 통신을 하기 위한 조건 (기존 방식은 2대 연결)

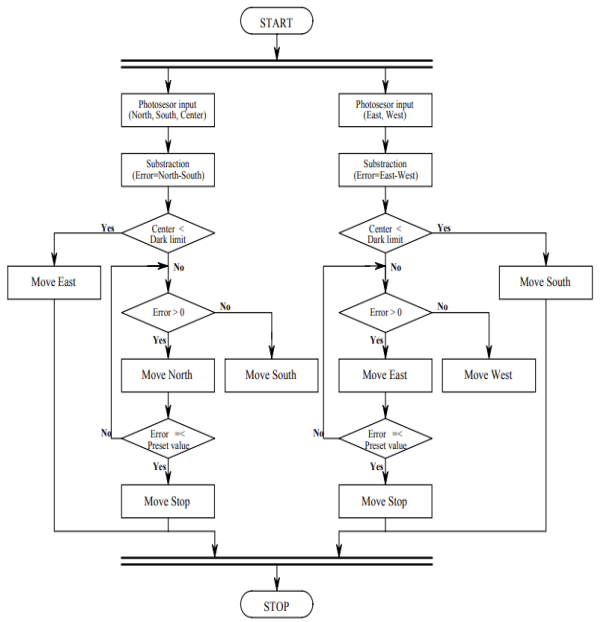
3대 소켓 통신과 동일한 이유로, 스레드와 소켓 통신 동시 실행 시 충돌 O

accept() 함수 관련 코드 수정으로 해결

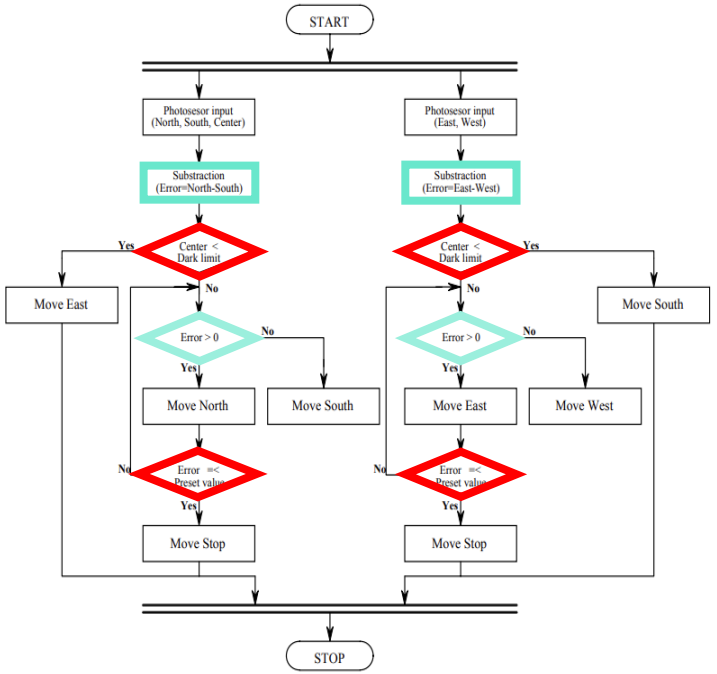
(System Flow)
응용 시스템 구조 설계

- 중간 발표 단계에서 작성한 분석 내용을 토대로 전체 구조 계획

a) 광센서 추적 알고리즘



a.1) 일부 연산 및 변수 수정



b) 조도 센서 x8 배치표 작성

Top-Left	Top-Middle	Top-Right
Middle-Left		Middle-Right
Bottom-Left	Bottom-Middle	Bottom-Right

< 조도 센서 배치표>

(System Flow)
응용 시스템 구조 설계

c) 구현 프로세스 3단계 분석 (변동 사항 O)

c.1) 조도 센서 설치



op-Left	Top-Middle	Top-Right
Middle-Left		Middle-Right
Bottom-Left	Bottom-Middle	Bottom-Right

1단계

- 조도 센서 8개를 태양광 판넬 위 8개 방향에 설치

- 광원의 세기 측정 (광 센서 추적 알고리즘)

2단계

c.2) 각도 연산

- X축 Horizontal 이동 각도 연산 후 서보 모터 동작
- Y축 Vertical 이동 각도 연산 후 서보 모터 동작

3단계

- 조도 센서로 측정한 빛의 세기 출력

- 온·습도 센서로 측정한 판넬의 온도 초과시 led on

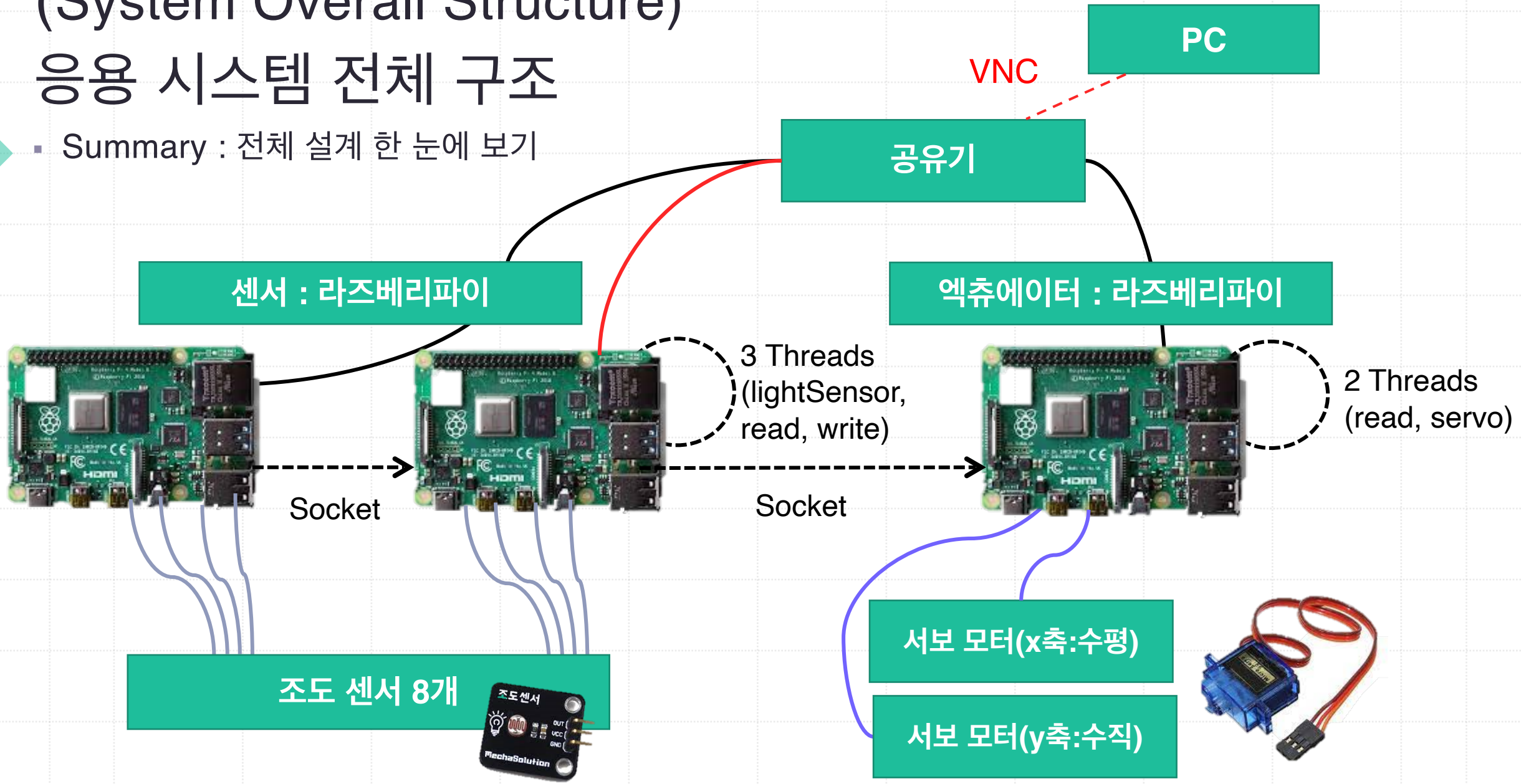
c.3) 빛의 세기 측정 후 바로 출력으로 변경

c.4) 온·습도 센서 & led 삭제

c.5) 센서 값 연산 결과를 기호로 출력하는 기능 추가

(System Overall Structure)
응용 시스템 전체 구조

- Summary : 전체 설계 한 눈에 보기



(System Algorithm)
응용 시스템 사용 알고리즘

- 전체

- socket()
- connect()

- lightSensor_thd()
- read_thd()
- servo_thd()

- pthread_create()
- pthread_join()

- read_thd()
- servo_thd()

RBP# 1



Socket

RBP# 2



3 Threads (lightSensor, read, write)

Socket

- socket()
- bind()
- listen() // from 1, 3
- accept()

RBP# 3



2 Threads (read, servo)



- prepare()
- readadc() // 0,2,4,6 channels



#include <WiringPi.h>
- softPwmCreate()
- softPwmWrite()

(System Algorithm)
응용 시스템 사용 알고리즘

- 세부

① computeAverage()

Top-Left	Top-Middle	Top-Right
Middle-Left		Middle-Right
Bottom-Left	Bottom-Middle	Bottom-Right

```
/*top_left,    top_middle,    top_right,
middle_left,                 middle_right,
bottom_left, bottom_middle, bottom_right*/

```

```
void computeAverage() {
    /* compute average */
    avg_top = (top_left + top_middle + top_right) / 3;
    avg_bottom = (bottom_left + bottom_middle + bottom_right) / 3;
    avg_left = (top_left + middle_left + bottom_left) / 3;
    avg_right = (top_right + middle_right + bottom_right) / 3;
    printf("avg_top : %d\n", avg_top);
    printf("avg_bottom : %d\n", avg_bottom);
    printf("avg_left : %d\n", avg_left);
    printf("avg_right : %d\n", avg_right);

    axis_h = avg_top - avg_bottom;
    axis_v = avg_right - avg_left;
    printf("axis_h : %d\n", axis_h);
    printf("axis_v : %d\n", axis_v);
    if (axis_h > 0) {
        printf("axis_h > 0\n");
        strcpy(h, "+");
    } else if (axis_h < 0) {
        printf("axis_h < 0\n");
        strcpy(h, "-");
    } else {
        printf("axis_h = 0\n");
        strcpy(h, "0");
    }

    if (axis_v > 0) {
        printf("axis_v > 0\n");
        strcpy(v, "+");
    } else if (axis_v < 0) {
        printf("axis_v < 0\n");
        strcpy(v, "-");
    } else {
        printf("axis_v = 0\n");
        strcpy(v, "0");
    }
}
```

(System Algorithm)
응용 시스템 사용 알고리즘

- 세부

② 실행문 inside servo_thd()

```
79  #define START 15
80  int gap = 2;
81
82  int v = START;
83  int h = START;
84  int dir=1;
85
86  void *servo_thd() {
87      while (1) {
88
89      // Test1
90      if((v<5||v>25)&&(h<5||h>25))
91      {
92          v=15;
93          h=15;
94          //printf("print in reverse way\n");
95      }
96
```

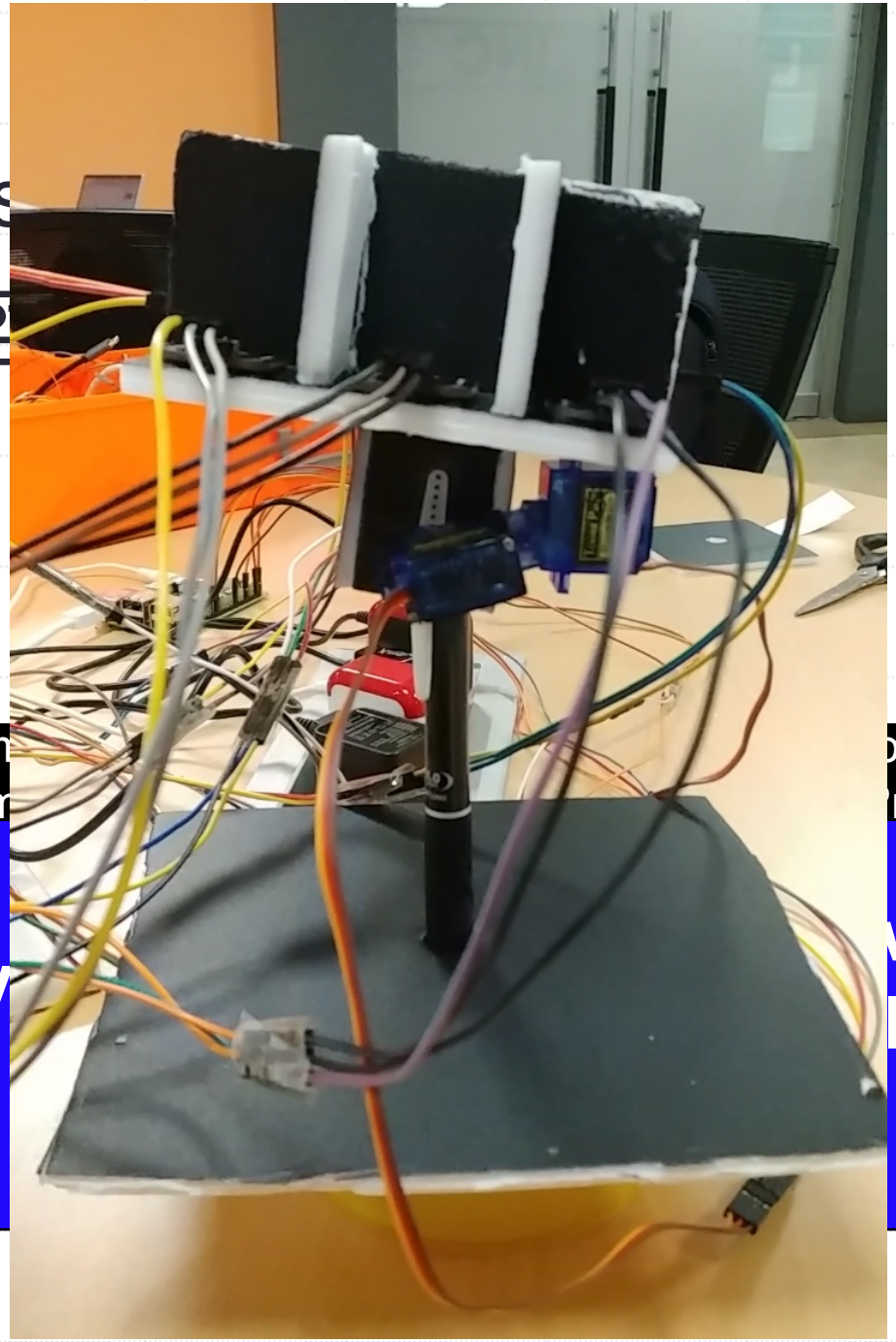
총 9가지 (3*3) 경우의 수
=> if 조건문 X9

```
if(!strcmp(msg1,"?")&&!strcmp(msg2,"?")) // (?,?)
{
    // if ? = "+", add gap to v or h -- value update
    v+=gap;
    h+=gap;
    // if ? = "-", minus gap to v or h -- value update
    v-=gap;
    h-=gap;
    // if ? ="0", no functional code -- pase updating

    // move servos with v or h degree (with updated v, h value)
    softPwmWrite(SERVO1, v);
    softPwmWrite(SERVO2, h);
    delay(500);
}
```

(Ass

시험



Comm
Prom

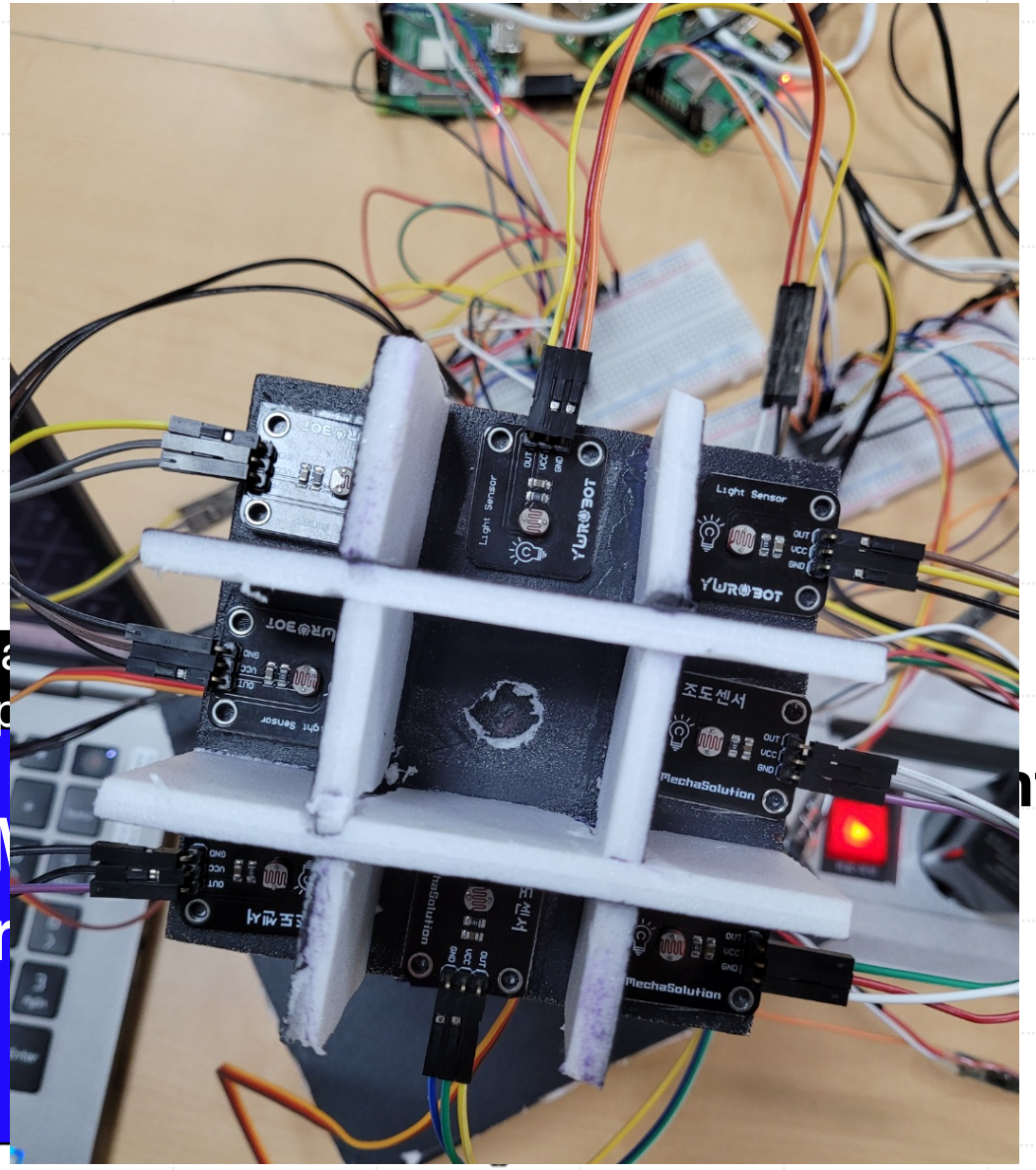
Serv

omma
romp

voM
lier

nted

order

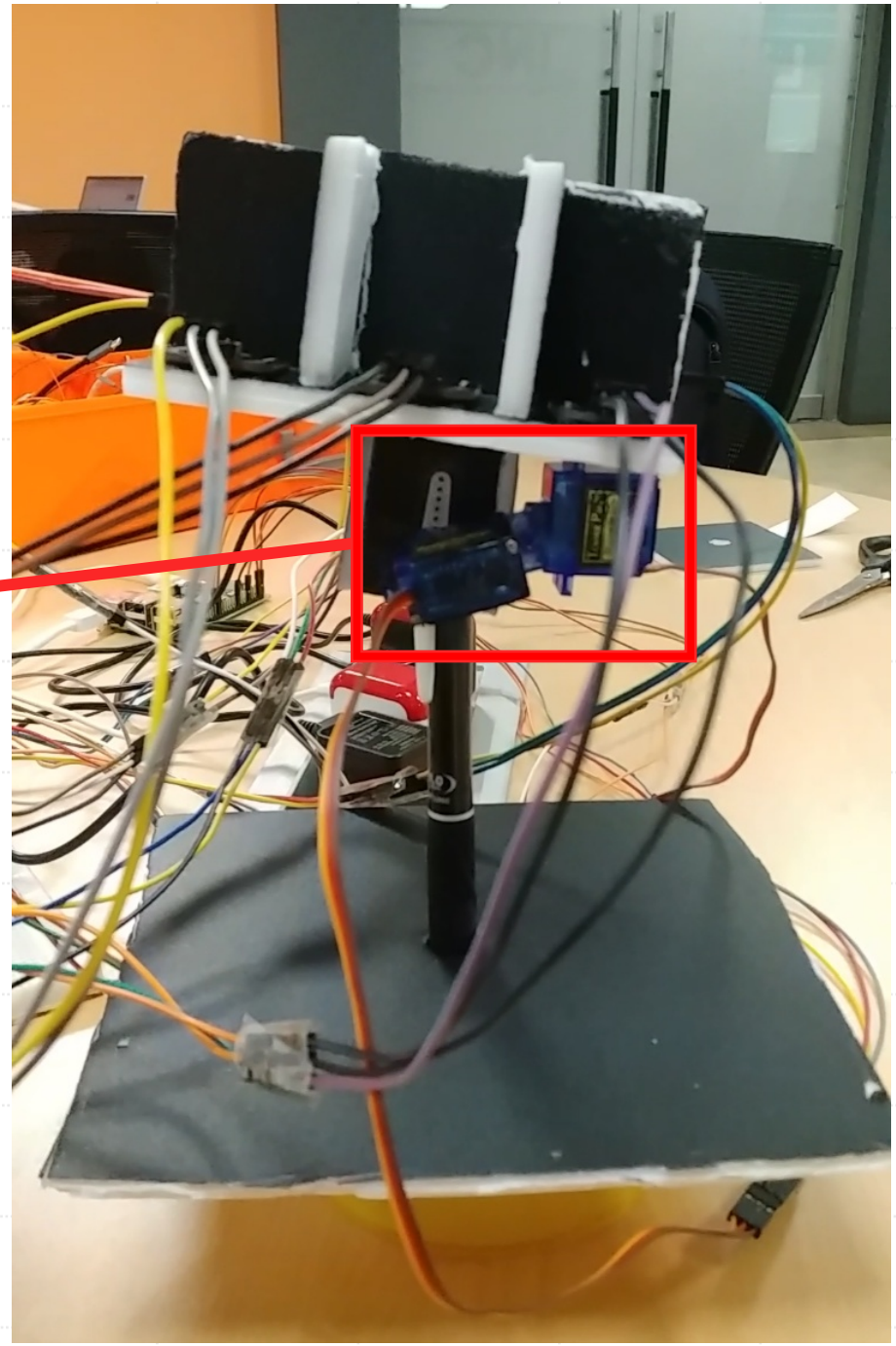


(Assessment Test)
시험 평가

➢ 문제점(Problem) 분석

P① 판의 무게
서보 모터가 조도 센서를 부착한 판의 무게를 견디지 못하여 회전이 원활하게 진행되지 않음

P② 라이브러리 사용
WiringPi 라이브러리 없이 PWM만으로 서보 모터를 작동시키는 데에는 성공했으나, 기타 구현사항(소켓 통신, 스레드)을 접목시키려 했을 때 과부하에 의해 프로그램이 중단되는 현상 발생 --> 해결X



감사합니다