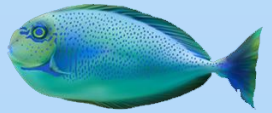
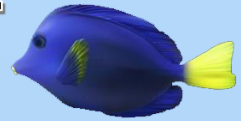
가정용 스마트 팜
아쿠아포닉스



meetPlatter
강원준, 허윤호, 김지은

1주차 결과 발표

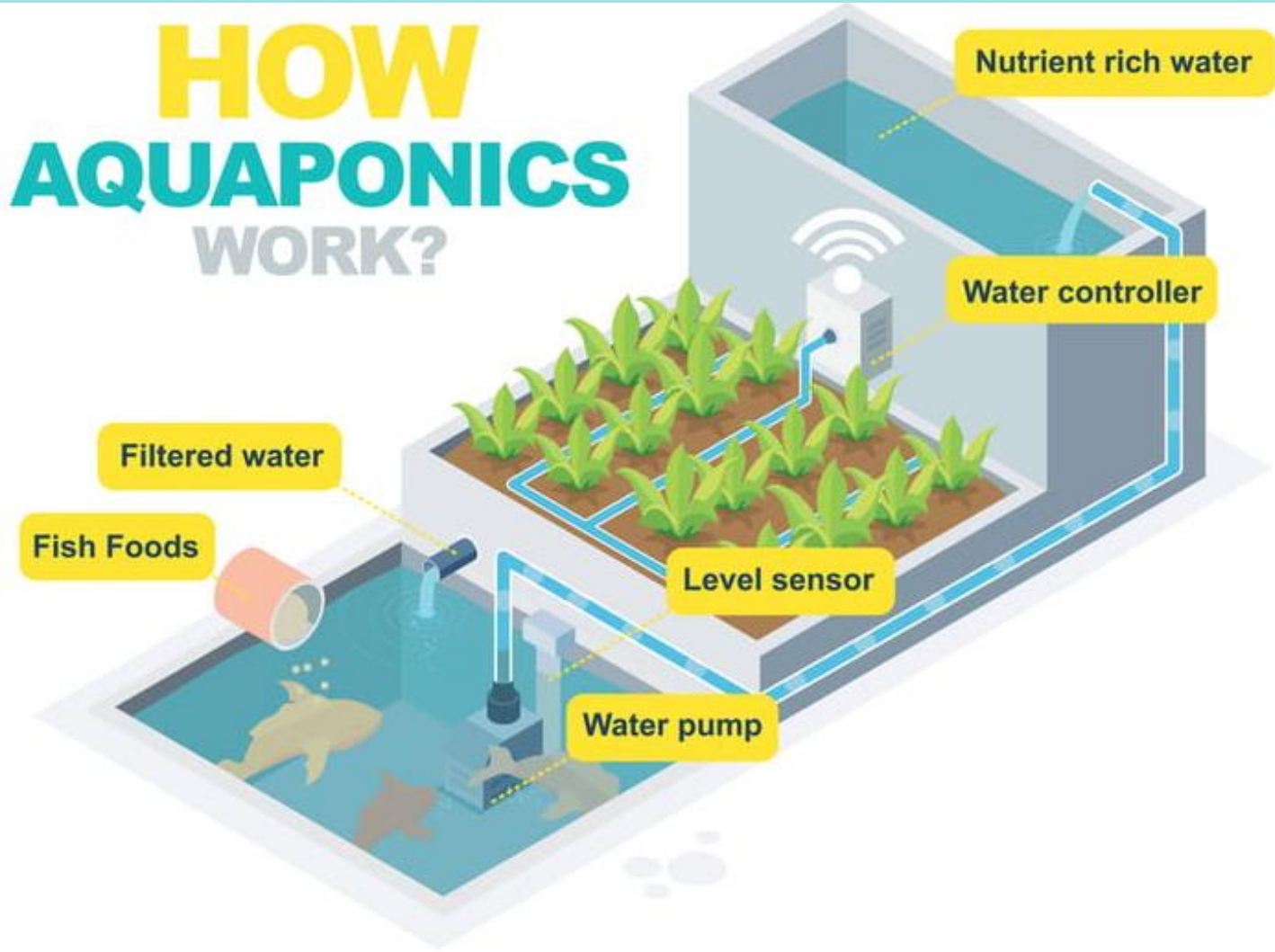
라이다 4기
배 진호 교수님

K-Digital

주제선정 이유

아쿠아포닉스란?

수경재배의 한 방법으로 물고기, 식물, 미생물을 조합한 친환경 순환재배 방법.



주제선정 이유

ISSUE

1인 가구 수↑
전염병 발생 확률↑
반려 시장↑

ISSUE

저택 내 거주시간↑
인테리어 시장↑
소품 시장↑

SOLUTION

반려 식물, 반려 물고기 동시 사육 가능
관리할 것이 적어 게으름이 많은 사람도 가능
자동 정수 시스템을 통해 물 사용량 감소
디자인에 따라 소품으로서 인테리어에 활용 가능
자급자족으로 지출 감소
자체 공급을 통한 수요량
감소로 물가 안정 도움

ISSUE

인간의 게으름
어항 환수

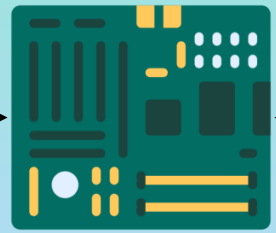
ISSUE

인플레이션
식자재 값↑

기술 조사

Hardware Architecture

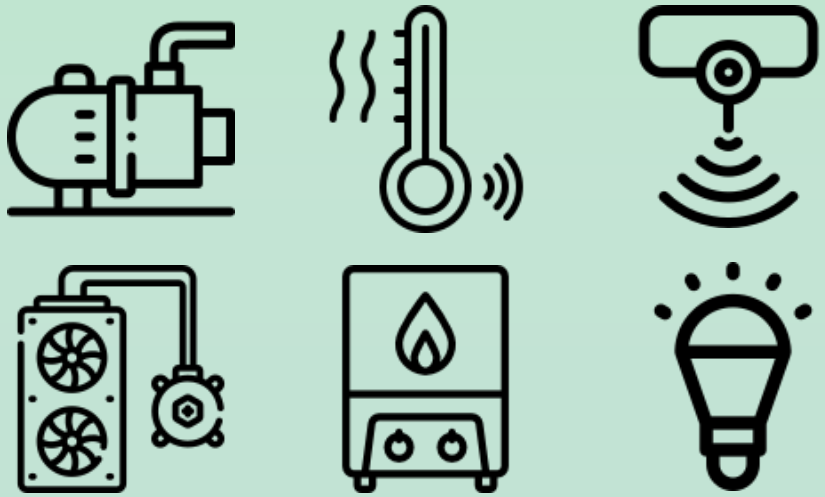
Arduino



••• Software

Control box

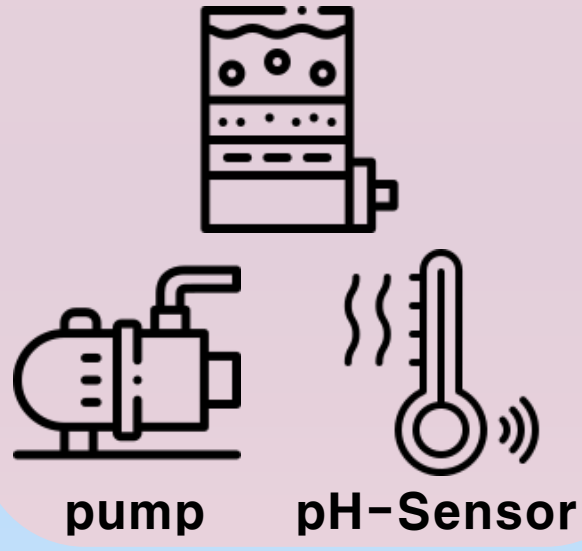
Pump Thermometer Sensor



Cooler Heater LED

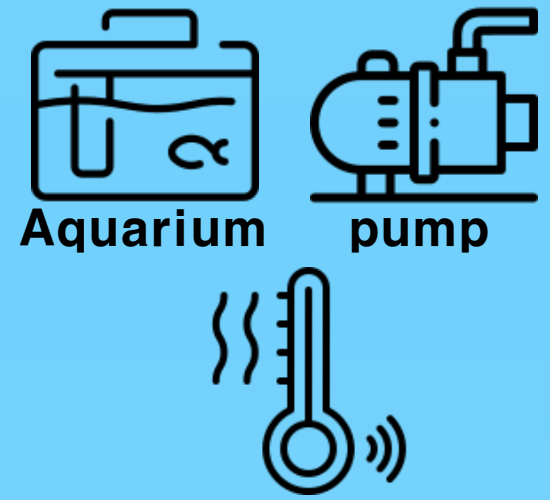
Bio Filter

Bio Filter Tank



pump pH-Sensor

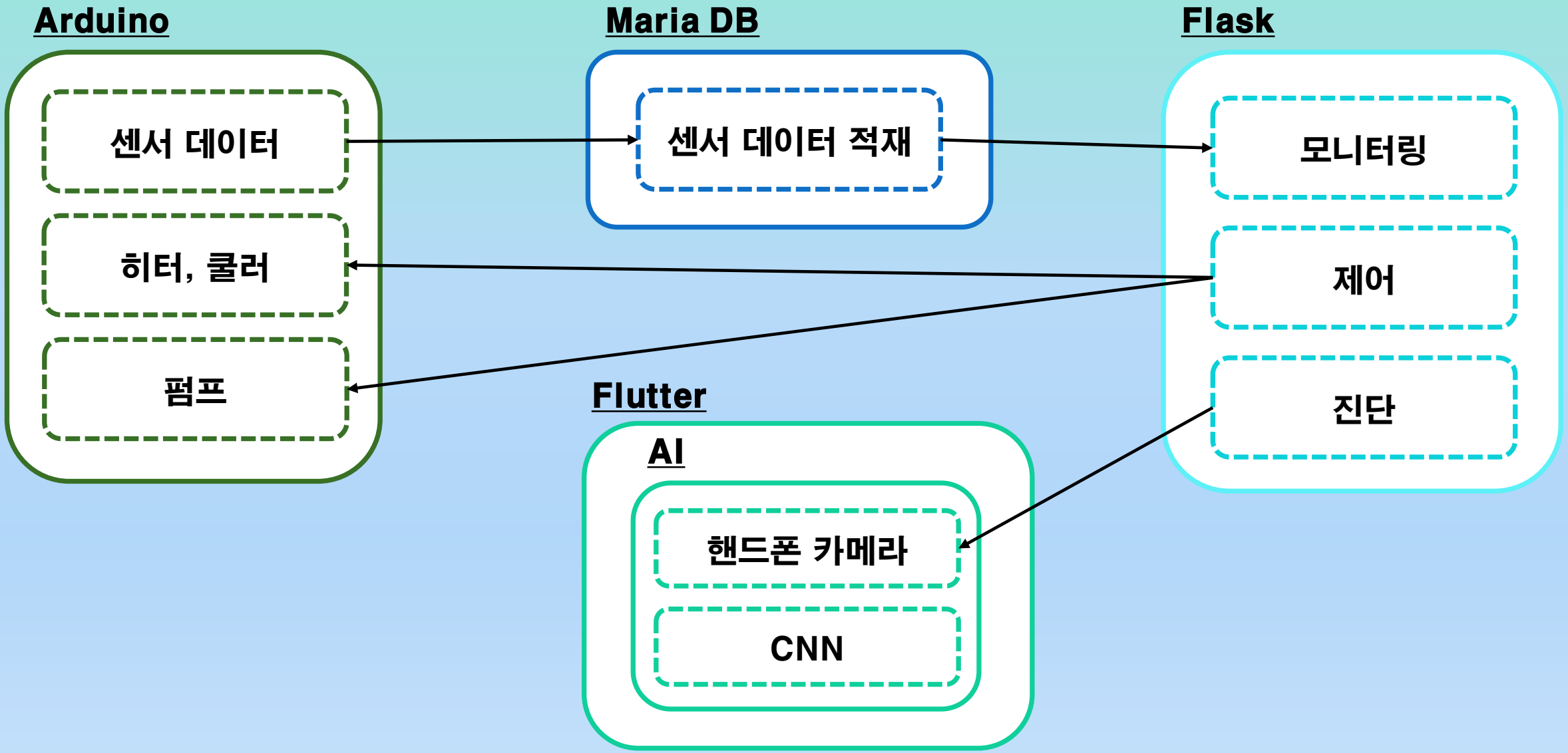
Aquarium



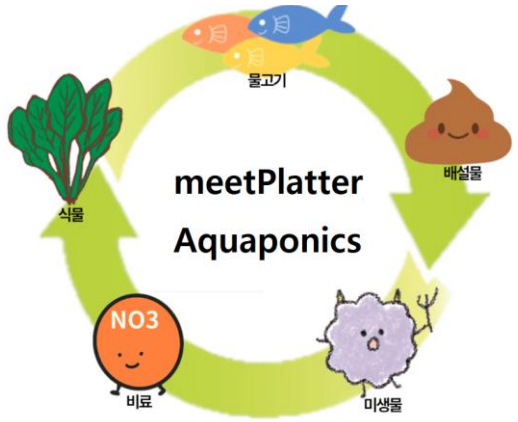
Aquarium pump

Thermometer

Software Architecture



Web GUI – 모니터링 및 제어



아쿠아포닉스 스마트팜의
센서정보를 모니터링하고
모터를 제어할 수 있는
웹사이트 입니다.

대기

CO2 농도		
322.2ppm	24℃	20,000lux

담액

이온농도 (pH)		
6.4	24℃	10cm

대기

CO2 농도		
322.2ppm	24℃	20,000lux

25 ℃

히팅

쿨링

이온농도 (pH)		
6.4	24℃	10cm

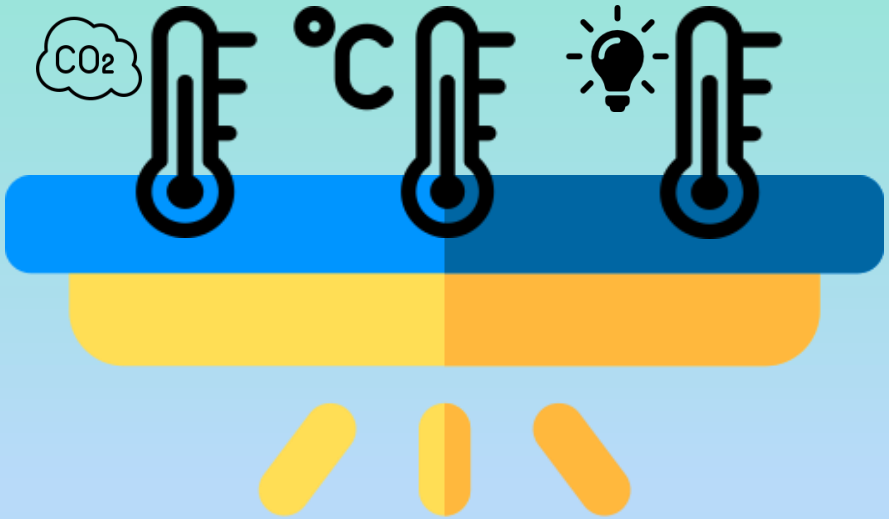
Mobile GUI – AI 진단

카메라

상태 : 신선함
생육단계 : 성목
병 유무 : 없음

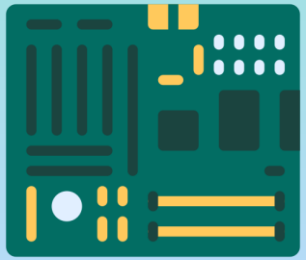
IOT 기술

식물환경 모니터링 및
제어센서(co2,온도,조도)



Output
(모바일 모니터링)

Input
(센서 데이터 수집)

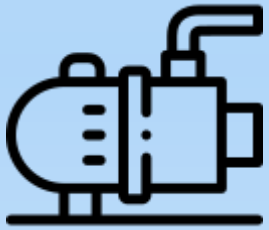


아두이노

스마트폰 모니터링

Output
펌프를 1시간 마다 15분씩 작동하여 물을 순환

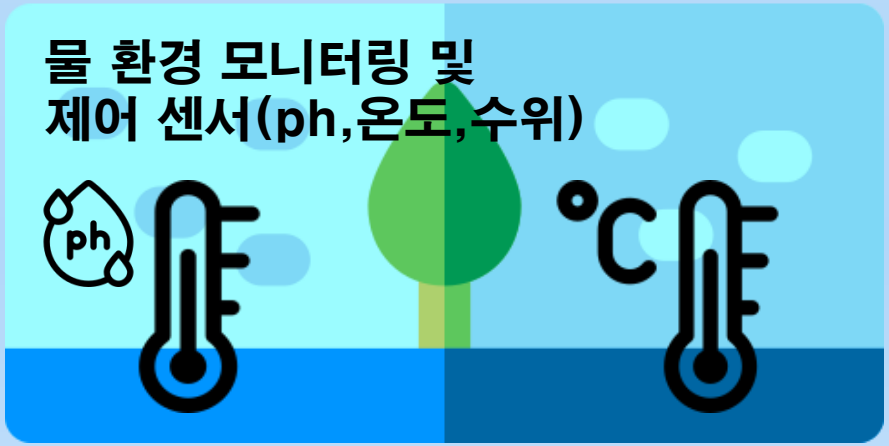
온도가 28℃ ↑ 쿨러를 작동하여 온도 ↓
온도가 24℃ ↓ 히터기 작동하여 온도 ↑



펌프

쿨러&히터

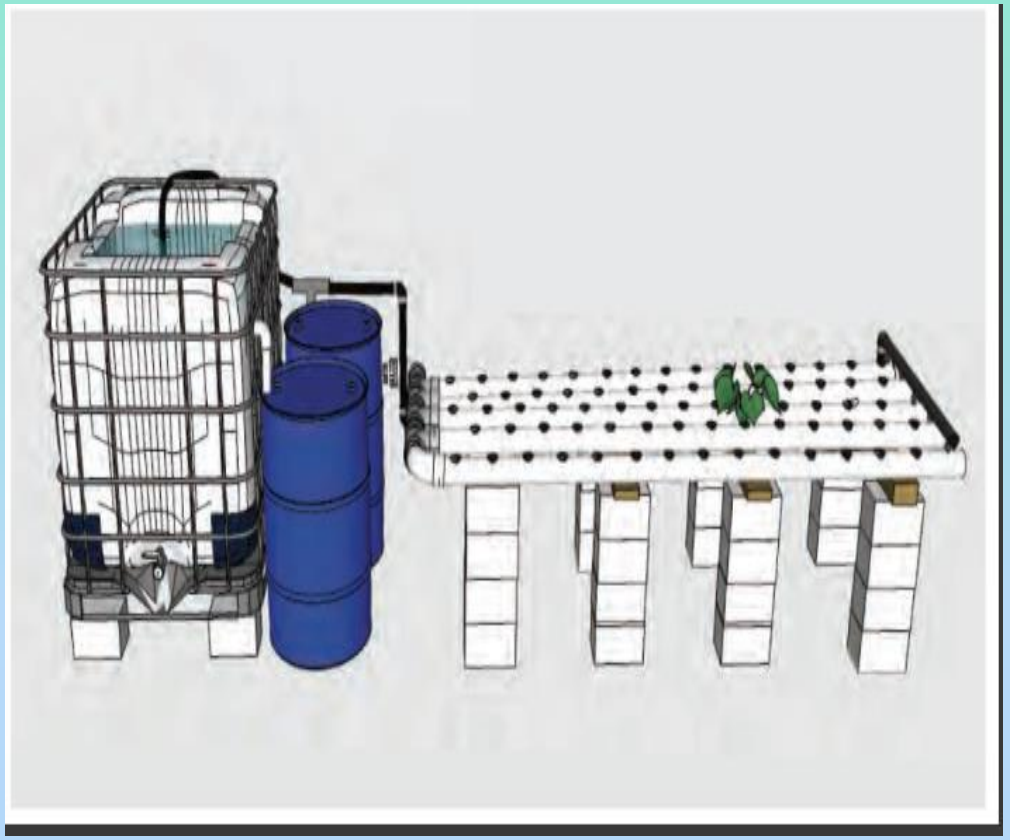
물 환경 모니터링 및
제어 센서(ph,온도,수위)



제어박스

IOT 기술

- **바이오 필터 박스**
  - pH 센서 : 제어 박스에 공급되는 ph농도 측정
  - 바이오 필터 : 식물의 영양분을 제공
  - 펌프 : 바이오 필터 박스에서 제어박스로 전달



- **어항**
  - 온도 센서 및 제어 : 열대어들이 자랄수있는 온도 제공
  - 공기 펌프 : 열대어들에게 산소를 제공하기위하여 설치
  - LED : 열대어들의 상태를 관찰할 수 있도록 설치 및 디자인적 요소
  - 펌프 : 어항에서 바이오 필터 박스로 전달

웹 개발 기술

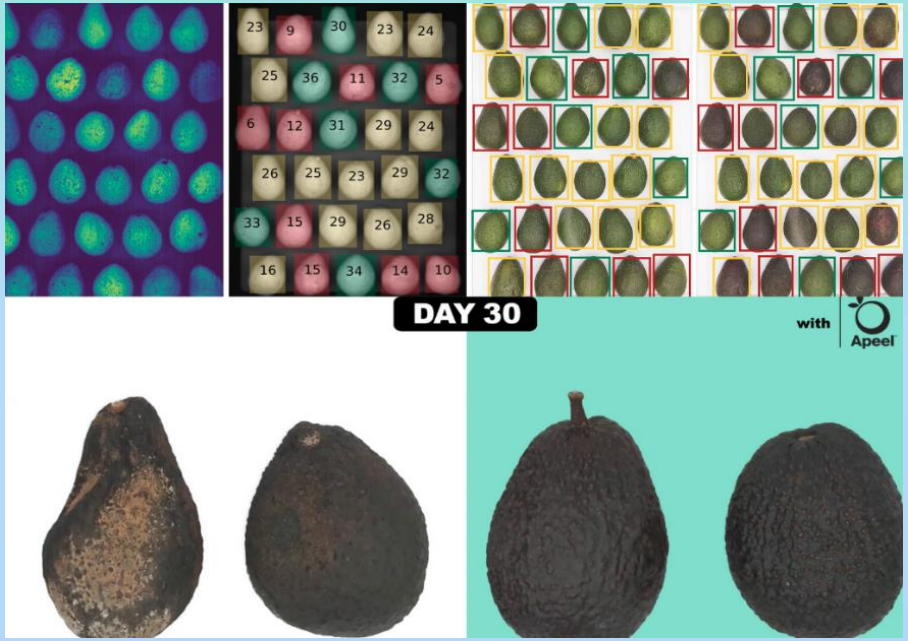
- 앱개발 + GUI 구성
  - Flask로 개발 예정
  - maria DB, arduino와 연동 가능
- 센서 모니터링
  - arduino로 부터 데이터를 받아 DB에 저장
  - Arduino – MySQL connector 이용하여 직접연결 (선택)
  - Arduino – PHP – MariaDB 이용하여 간접연결 -> 일의 양이 많아짐
- 모터 원격 제어
  - web의 GUI로 제어 가능함
  - 통신 방식 : wifi

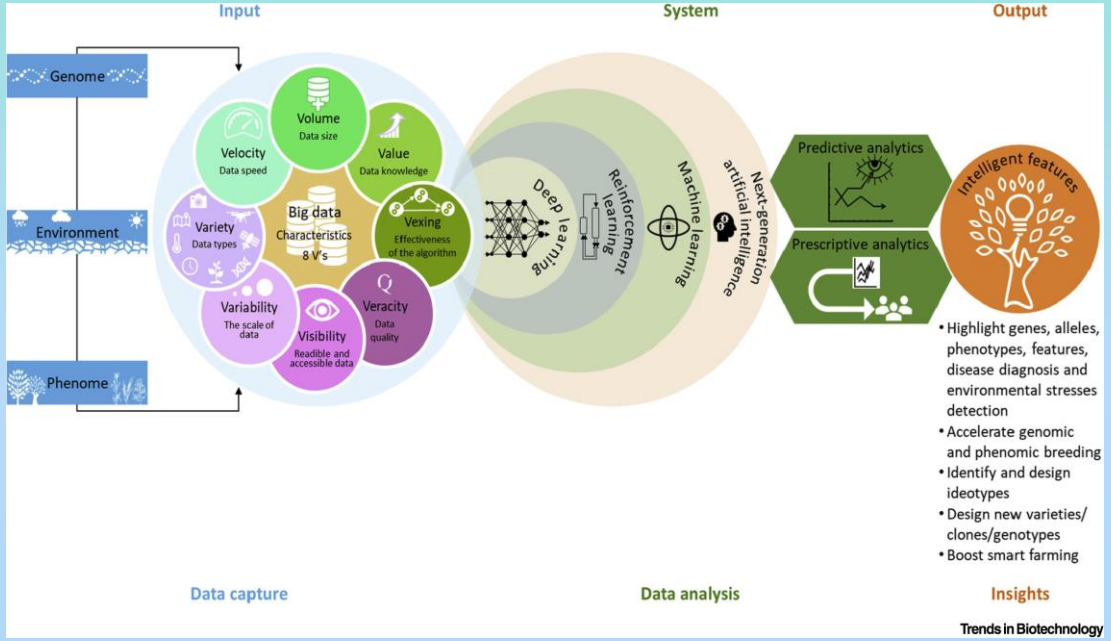
앱 개발 기술

- 앱개발 + GUI 구성
  - 프레임워크를 Flutter, 개발툴을 AndroidStudio로 개발 예정
  - maria DB와 연동 가능
  - 안드로이드 에뮬레이터 이용가능
  - 다른 os에 적용가능
- AI 진단
  - CNN으로 학습시킨 모델 이식
  - flask로 만든 웹과 연동가능
  - 사용자의 스마트폰으로 사진 입력

    -> 미리 학습된 모델을 이용하여 식물 또는 물고기의 상태 진단

현재 스마트 팜에 사용되는 AI 기술





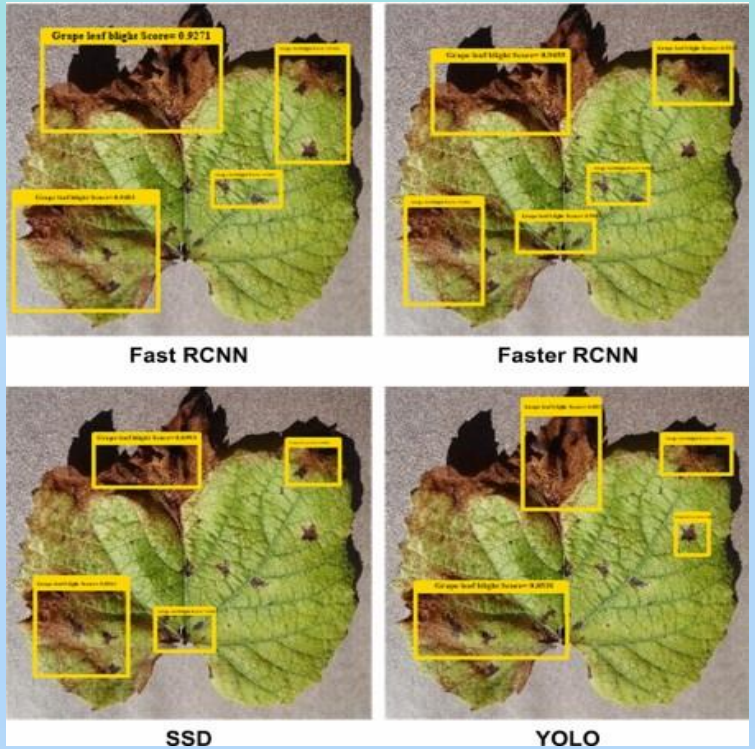
품질 관리

(Quality Control)

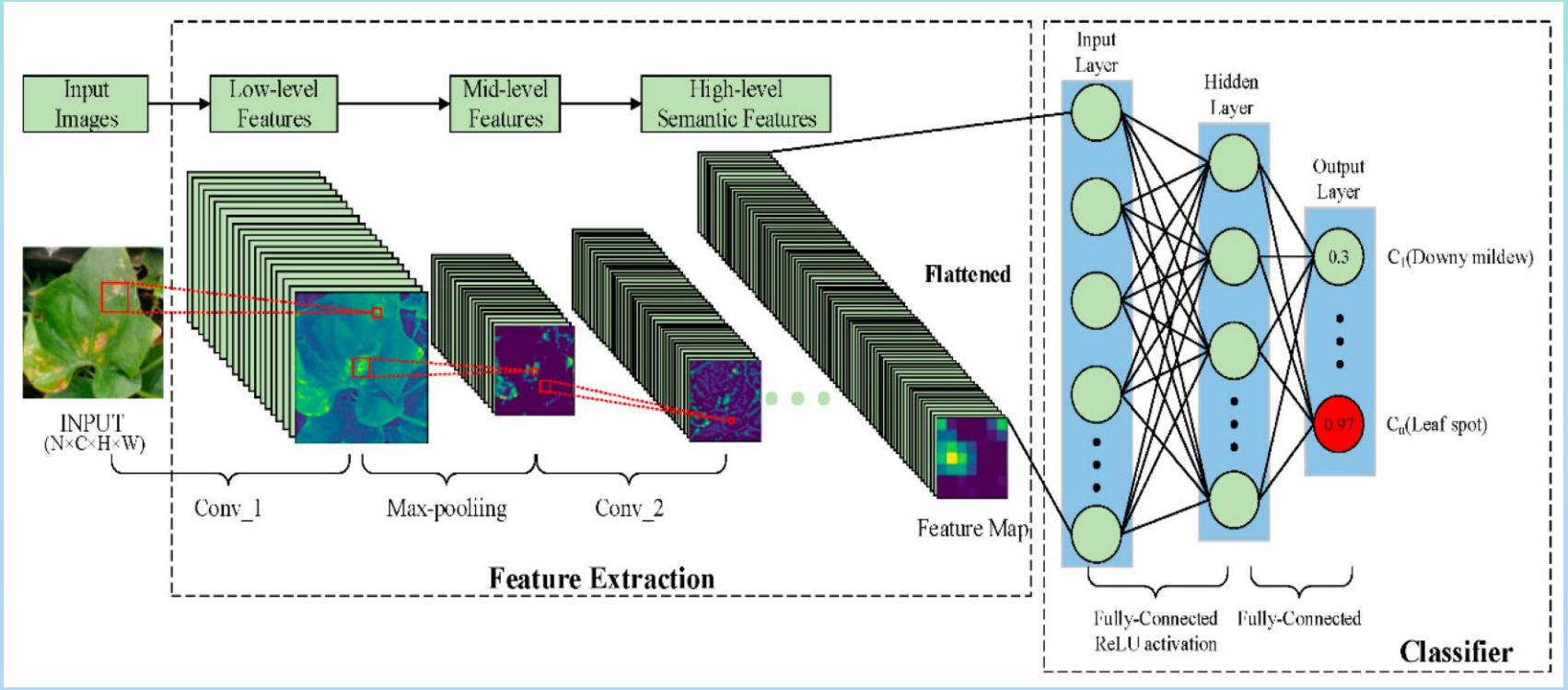
품종 개량

(Plant Breeding)

CNN



CNN 오픈 소스



CNN 구조

AI 진단을 통한 Quality Control

좋은 상추

시든 상추

정상 구피

백점병

병든 상추

상추 묘목

배마름병

임신 구피

강원준 일정

7월

task	14	15	18	19	20	21	22	25	26	27	28	29
주제선정 및 기본조사												
디자인 외주, 수정												
기존 스마트팜 AI 조사												
AI 작동 확인 및 프로젝트 상황에 맞게 변경												
디자인에 의거한 하드웨어 외주												

8월

task	1	2	3	4	5	8	9	10	11	12	16	17	18	19	22	23	24	25	26	29	30	31
휴가																						
디자인에 의거한 하드웨어 외주																						
AI 작동 확인 및 프로젝트 상황에 맞게 변경																						
AI를 통한 자동제어 구현																						
AI, IOT, Application 결합 및 테스트																						
미진한 부분 보완																						
서로 가르치기																						

9월

task	1	2	5	6	7	8
서로 가르치기						
발표준비						

2주차 일정(2022.07.27 ~ 2022.08.08) - 강원준

27일(수) ~ 01일(월) - 디자인 패턴 책 공부 완료.
(전략, 옵저버, 데코레이터, 팩토리, 싱글턴, 커맨드, 어댑터, 파사드, 템플릿 메소드, 반복자, 컴포짓, 상태, 프록시, 복합, 브릿지, 빌더, 책임 연쇄 등.)

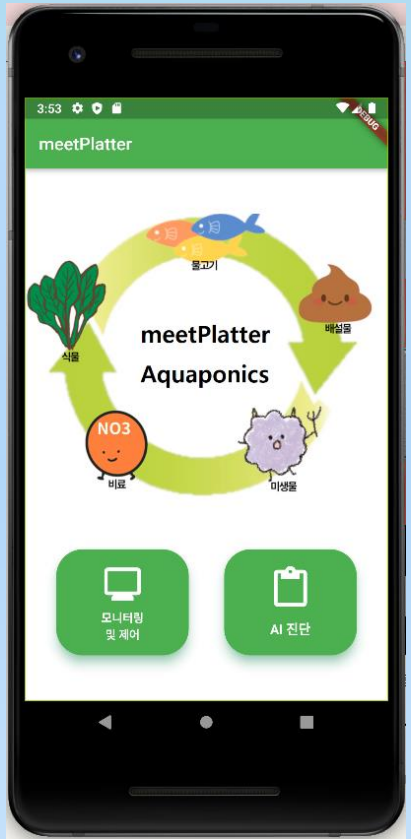
04일(목) - 배운 디자인 패턴을 기반 리팩토링 진행.
169줄 → 66줄 약 100줄 간소화.

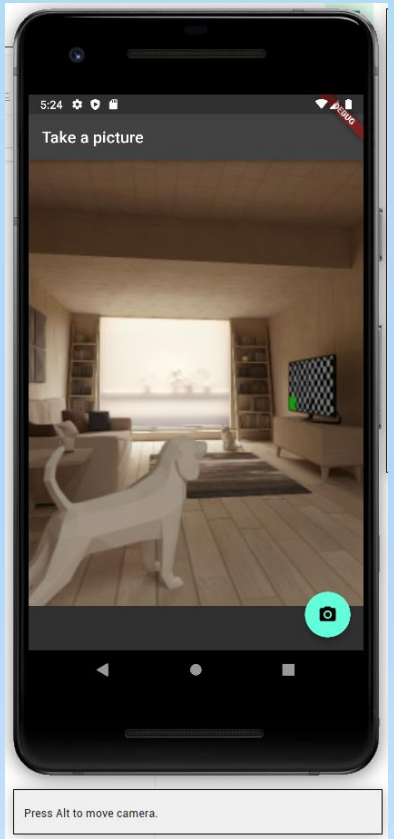
05일(금) ~ 07일(일) - Flutter 공부.

08일(월) - Flutter로 앱 제작 시작.

```python
if __name__ == '__main__':
    create_graph1()
    g1 = tf.get_default_graph()
    tf.reset_default_graph()
    create_graph2()
    g2 = tf.get_default_graph()
    tf.reset_default_graph()
    create_graph3()
    g3 = tf.get_default_graph()
    tf.reset_default_graph()
    create_graph4()
    g4 = tf.get_default_graph()

    with g1.as_default():
        run_inference_on_image1()
    with g2.as_default():
        run_inference_on_image2()
    with g3.as_default():
        run_inference_on_image3()
    with g4.as_default():
        run_inference_on_image4()
```

```python
if __name__ == '__main__':


    bpath = os.path.dirname("/home/won/project_AI/inception_4_project/result/output_graph1.pb")
    graph_path = [1, 2, 3, 4]
    label_path = [1, 2, 3, 4]
    graph_group = [1, 2, 3, 4]

    for i in range(4):
        graph_path[i] = bpath+"/output_graph"+str(i+1)+".pb"
        label_path[i] = bpath+"/output_labels"+str(i+1)+".txt"
        create_graph(i)
        graph_group[i] = tf.get_default_graph()
        with graph_group[i].as_default():
            run_inference_on_image(i)
        tf.reset_default_graph()
```





앞으로 일정

카메라 기능을 가진 view page 제작.
AI model과 연결 시키기.
촬영 → image → AI → App → view.
App 마켓 등록.
Firebase 연구.

허윤호 일정

7월

task	14	15	18	19	20	21	22	25	26	27	28	29
주제선정 및 기본조사												
IOT 아쿠아포닉스 기술조사												
기술수집종합 자재구입												
아두이노를 이용한 펌프제어 구현												

8월

task	1	2	3	4	5	8	9	10	11	12	16	17	18	19	22	23	24	25	26	29	30	31
휴가																						
아두이노를 이용한 모터제어(환풍및 온도조절)																						
아두이노를 이용한 이산화탄소 및 PH센서 제어																						
아두이노를 이용한 기술들 병합																						
스마트아쿠아포닉스 조립및 기술구현																						
데이터 종합및 팀원들과 공유																						

9월

task	1	2	5	6	7	8
서로 가르치기						
발표준비						

2주차 일정(2022.07.27 ~ 2022.08.08) - 허윤호

7월 27일 - 센서위치를 고려한 제어박스 모델링

7월 28일 - 바이오 필터 박스 최종모델링

7월 29일 - 모델링한 부분들 전부 실측하여 디자인 외주 맡길 도안 작성

8월 4일 - 앱을 활용하여 디자인외주 신청

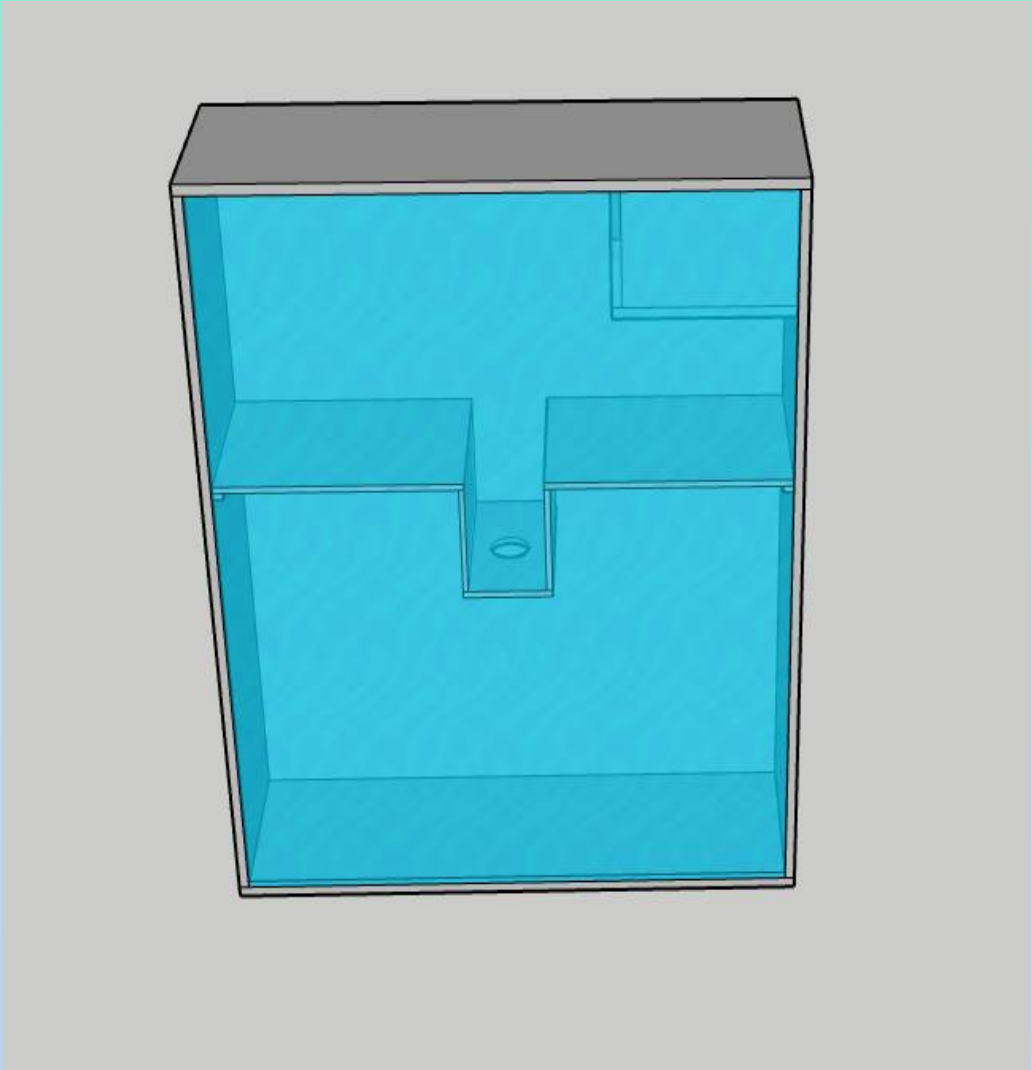
8월 8일 - 목재 선택, 목재 견적문의 온습도 센서와 co2센서 데이터 값을 받아 시리얼 모니터로 출력해주는 코드 작성 + 온도에 따른 모터제어 기능추가

온습도센서와 co2센서 데이터 값을 시리얼 모니터로 출력

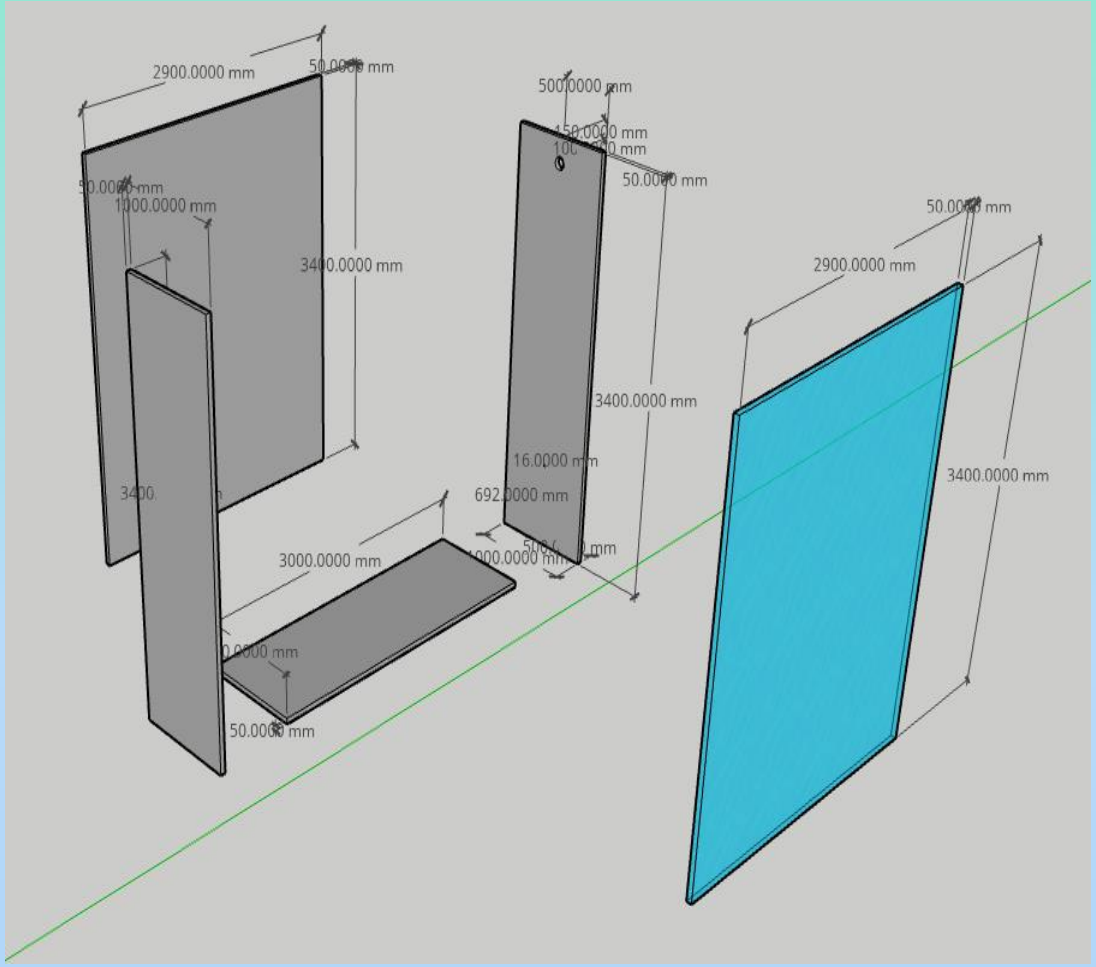
```
#include "DHT.h"
#include <SoftwareSerial.h>       /'SoftwareSerial.h' 포함
#include <MHZ19.h>          /'MHZ19.h' 포함

#define DHTPIN A3     // DHT22 센서의 데이터 핀번호를 정의해주었습니다

#define DHTTYPE DHT22   // DHT 22  타입으로 설정
SoftwareSerial ss(2,3);
DHT dht(DHTPIN, DHTTYPE);
MHZ19 mhz(&ss);
void setup() {
  Serial.begin(9600);
  Serial.println(F("START!"));  *시리얼 모니터 화면 출력
  pinMode(A4,OUTPUT);                *모터 드라이버 핀을 A4로 설정
  pinMode(A5,OUTPUT);                *모터 드라이버 핀을 A5로 설정
  dht.begin();
  ss.begin();
}

void loop() {
  delay(2000);

  MHZ19_RESULT response = mhz.retrieveData();
  if (response == MHZ19_RESULT_OK)
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
* 습도 온도 화씨 온도를 변수 선언
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
* 센서를 읽는데 실패하면 오류 메세지 출력
```

```
  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print(F("CO2: "));
  Serial.println(mhz.getCO2());        /시리얼 모니터에 Co2값 표시
  Serial.print(F("Min CO2: "));
  Serial.println(mhz.getMinCO2());        /시리얼 모니터에 최소 Co2값 표시
  Serial.print(F("Temperature: "));
  Serial.println(mhz.getTemperature());        /시리얼 모니터에 온도 값 표시
  Serial.print(F("Accuracy: "));    Serial.println(mhz.getAccuracy());        /시리얼 모니터에 정확도 !
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));

if(int(t)>=27) *온도가 27도 이상으로 올라가면
  {
    analogWrite(A4,255); *모터 켜짐
  }
  else
    analogWrite(A4,0);  *모터 꺼짐
}
```

온도 센서를 읽어 27도이상으로 올라가면 모터가 작동하도록 제어



바이오 필터 최종도안



바이오 필터 실측

김지은 일정

7월

task	14	15	18	19	20	21	22	25	26	27	28	29
주제선정 및 기본조사												
Application 기술조사(웹, MariaDB)												
아두이노 test 데이터로 DB 쌓기												
DB 바탕으로 웹 구현												

8월

task	1	2	3	4	5	8	9	10	11	12	16	17	18	19	22	23	24	25	26	29	30	31
휴가																						
아두이노 test 데이터로 DB 쌓기																						
DB 바탕으로 웹 구현																						
모바일 app 구현																						
AI, IOT, Application 결합 및 테스트																						
미진한 부분 보완																						
서로 가르치기																						

9월

task	1	2	5	6	7	8
서로 가르치기						
발표준비						

2주차 일정(2022.07.27 ~ 2022.08.08) - 김지은

27일(수) – xampp으로 db와 php 사용을 간편하게 함 -> 하지만 이를 사용하지 않아도 구현 가능함

28일(목) – mariadb에 실시간 데이터 전송 확인

29일(금) – 웹페이지로 온습도 데이터 모니터링 가능

4일(목) – 자동으로 온습도 업데이트 기능 추가(웹)

5일(금) – 아두이노 -> nodemcu로 센서의 데이터 전송 완료

8일(월) – flutter와 androidstudio를 이용하여 어플 gui 구성

앞으로 일정

- 어플과 db연동
- gui 다듬기

마리아 DB에 실시간 온습도 업데이트

```
MariaDB [test]> select * from DHT11;
+-------------+----------+---------------------+
| Temperature | Humidity | Time                |
+-------------+----------+---------------------+
|        24.5 |       47 | 2022-07-27 17:01:31 |
|        24.5 |       47 | 2022-07-27 17:01:37 |
|        24.5 |       47 | 2022-07-27 17:01:43 |
|        24.5 |       47 | 2022-07-27 17:01:50 |
|        24.5 |       47 | 2022-07-27 17:01:56 |
|        24.5 |       47 | 2022-07-27 17:02:02 |
|        24.5 |       46 | 2022-07-27 17:02:08 |
|        24.5 |       46 | 2022-07-27 17:02:14 |
|        24.5 |       46 | 2022-07-27 17:02:20 |
|        24.5 |       46 | 2022-07-27 17:02:26 |
|        24.5 |       46 | 2022-07-27 17:02:32 |
|        24.5 |       46 | 2022-07-27 17:02:38 |
|        24.5 |       47 | 2022-07-27 17:02:44 |
|        24.5 |       46 | 2022-07-27 17:02:51 |
|        24.5 |       47 | 2022-07-27 17:02:57 |
|        24.5 |       47 | 2022-07-27 17:03:03 |
|        24.5 |       46 | 2022-07-27 17:03:09 |
|        24.5 |       46 | 2022-07-27 17:03:15 |
|        24.5 |       46 | 2022-07-27 17:03:21 |
|        24.5 |       46 | 2022-07-27 17:03:27 |
```

```
11:07:20.998 -> Humidity: 46.00%  Temperature: 24.60°C Heat index: 24.32°C
11:07:21.031 -> 기준 시간 : 2022-07-27 17:07:20
11:07:21.031 ->
11:07:21.031 -> [HTTP] 서버 연결을 시도합니다...
11:07:21.031 -> [HTTP] 수집한 값의 POST 요청을 시도합니다...
11:07:21.064 -> [HTTP] 응답 Code : 200
11:07:21.064 -> 서버로부터 수신된 응답 : 데이터 입력이 완료되었습니다.
11:07:21.064 ->
```

웹페이지에 실시간 온습도 업데이트

```
13:11:27.942 -> getTemp received
13:11:27.975 -> data sent
13:11:37.985 -> GET /getTemp HTTP/1.1
13:11:37.985 ->
13:11:37.985 -> getTemp received
13:11:38.019 -> data sent
13:11:48.024 -> GET /getTemp HTTP/1.1
13:11:48.024 ->
13:11:48.024 -> getTemp received
13:11:48.024 -> data sent
13:11:58.035 -> GET /getTemp HTTP/1.1
13:11:58.035 ->
13:11:58.035 -> getTemp received
13:11:58.068 -> data sent
13:12:08.062 -> GET /getTemp HTTP/1.1
13:12:08.062 ->
13:12:08.062 -> getTemp received
13:12:08.095 -> data sent
```

플러터로 gui 구성


```dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

class MonitoringPage extends StatefulWidget {
  const MonitoringPage({Key? key}) : super(key: key);

  // final str brightness = 20;

  @override
  State<MonitoringPage> createState() => _MonitoringPageState();
}

class _MonitoringPageState extends State<MonitoringPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
appBar: AppBar(
      backgroundColor: Colors.green,
      title: Text('Monitoring & Control'),
      ),
      body: Container(
        child: Center(
          child: Column(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: [
                    Container(
                      child: Text('대기'),
                    ),
                    Container(
                      child: Row(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                          SizedBox(
                            width: 120,
                            height: 120,
```