네 식물
내가
살려

내 식물의 무병장수

목차

1

프롤로그

기획 의도

WHY

반려식물, 상품
수요/매출 증가

WHO

반려식물을
잘 키우고 싶은
식집사들

WHAT

식물의 생장
병해충 판단
스마트 화분

기능

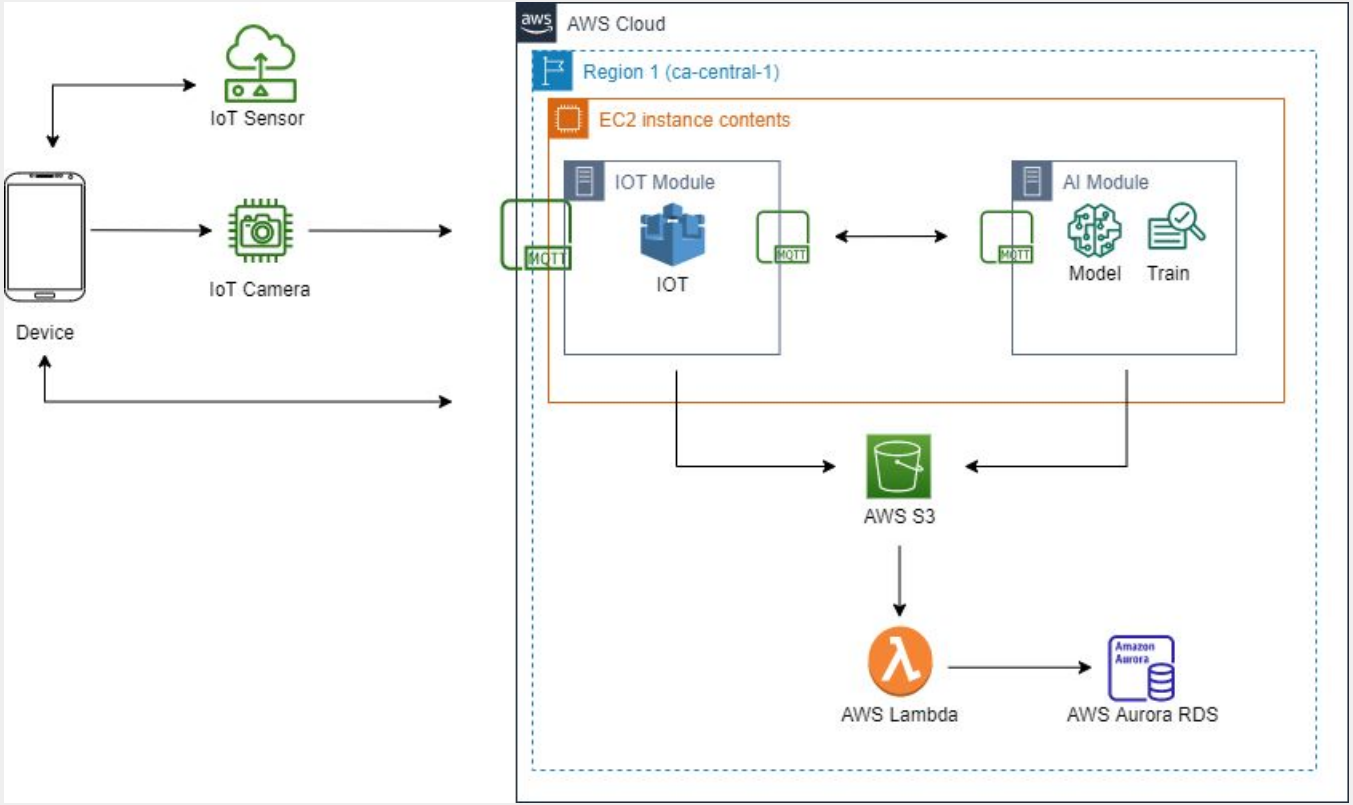
스마트
화분

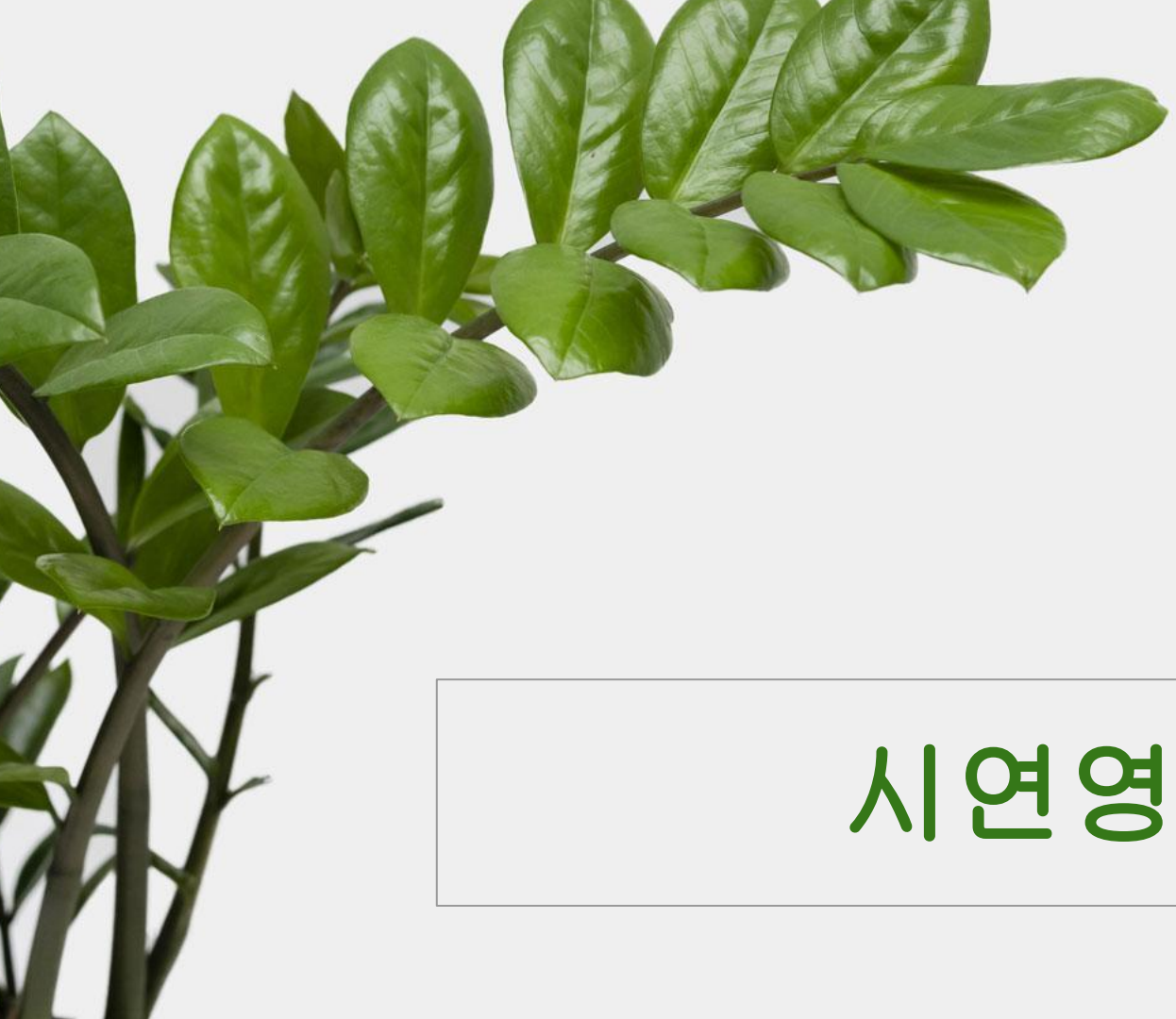
병충해
진단

기록
보관소

스마트 화분 프로세스

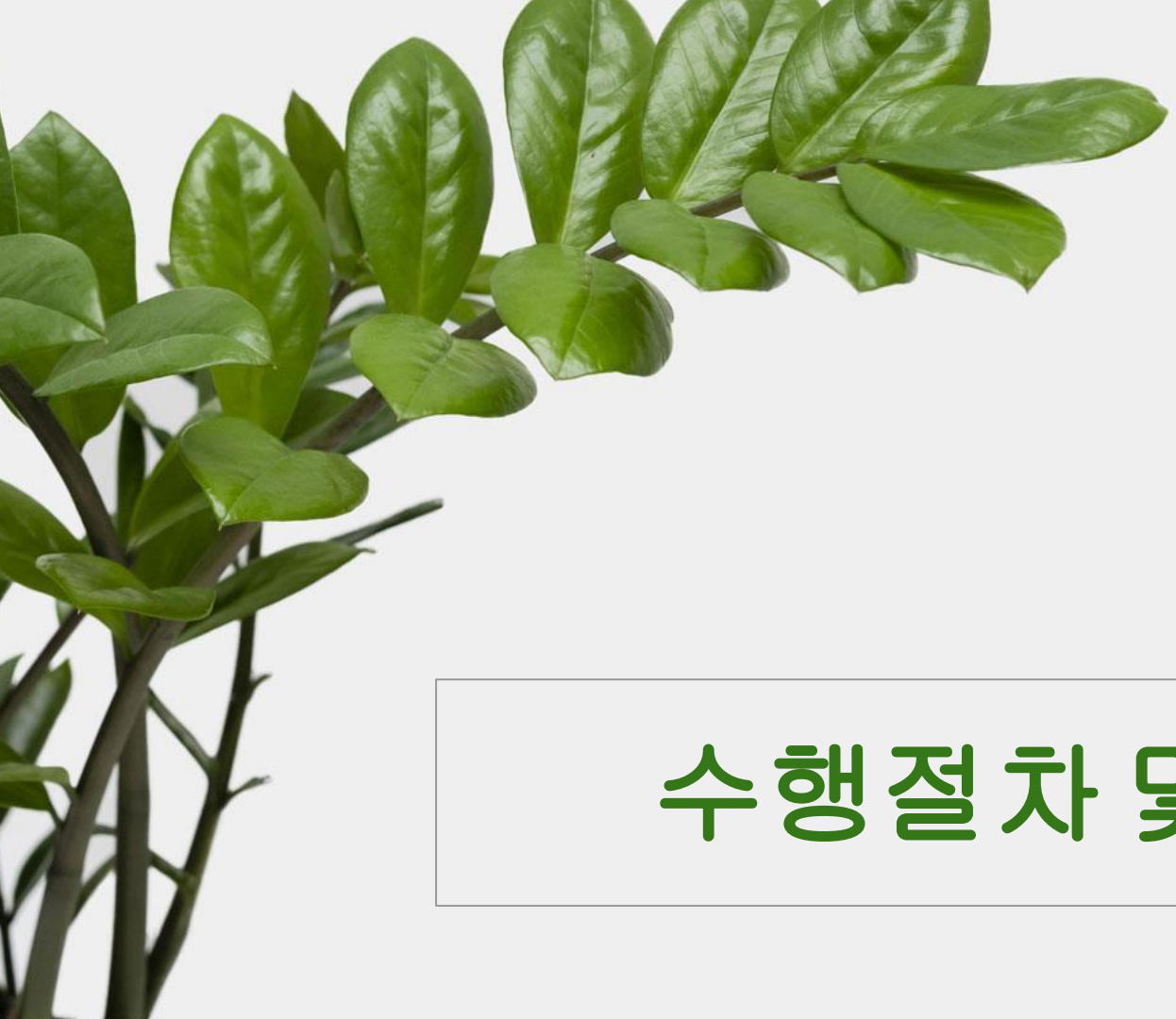
아키텍처


IoT Sensor
IoT Camera
Device
AWS Cloud
Region 1 (ca-central-1)
EC2 instance contents
IOT Module
IOT
MQTT
MQTT
AI Module
MQTT
Model
Train
AWS S3
AWS Lambda
Amazon Aurora
AWS Aurora RDS



2

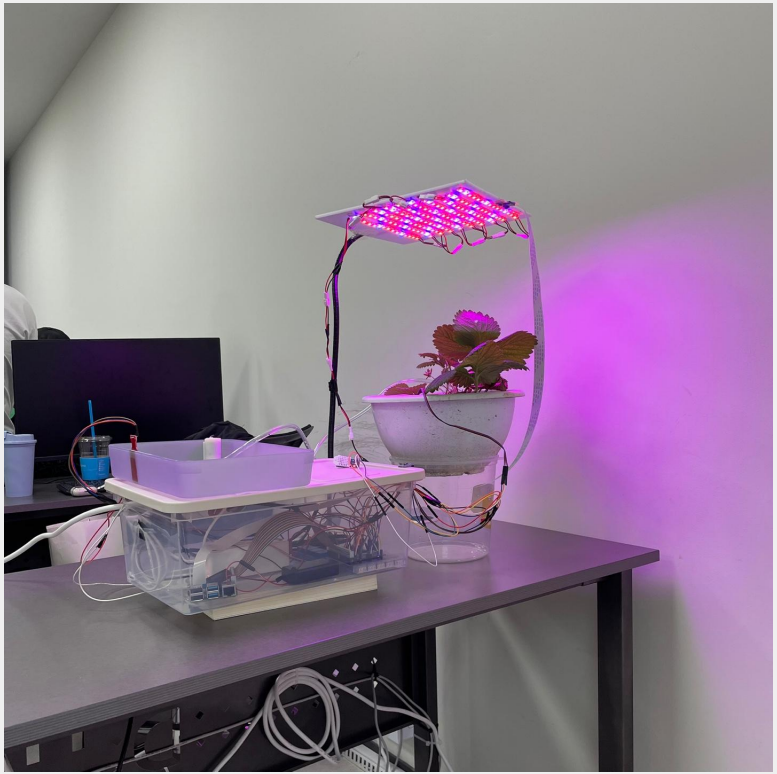
시연영상



3

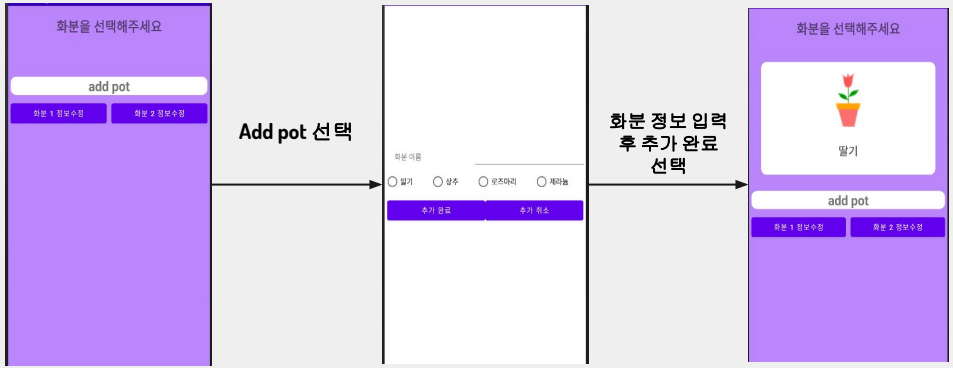
수행절차 및 방법

화분모습

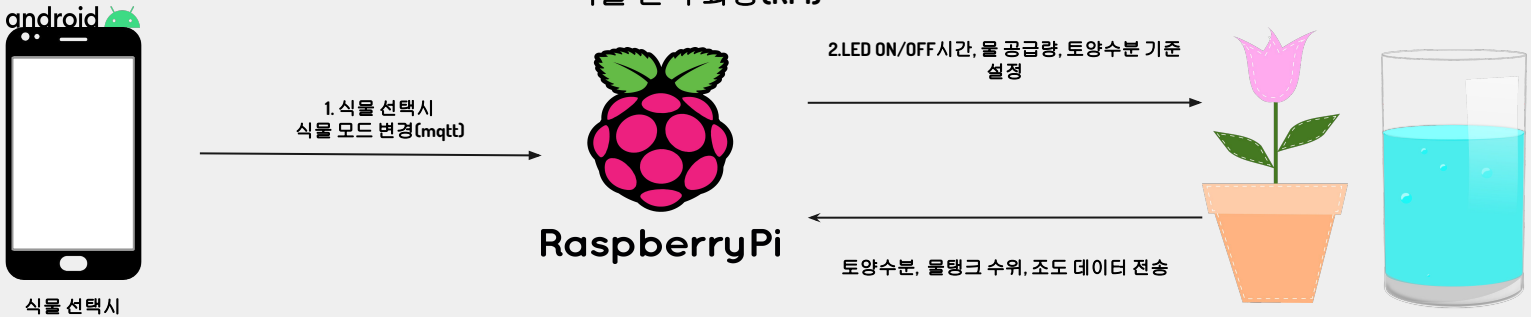


식물 선택

식물 선택 과정(앱)

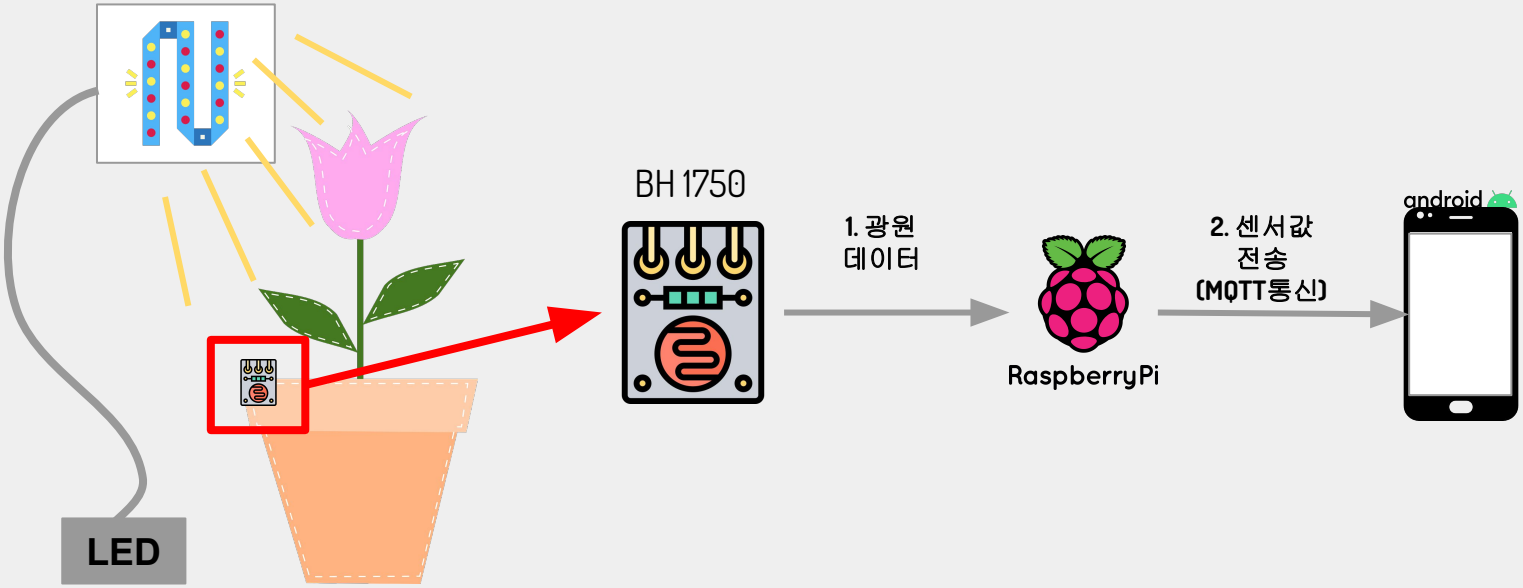


식물 선택 과정(RPI)

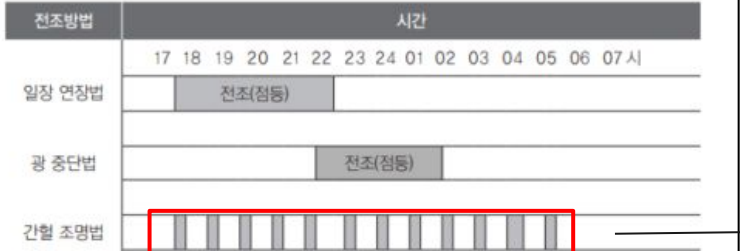


식물 선택시
온도, 조도, 토양습도 경고 기준 설정

조도 센서



조도 센서



(그림 33) 딸기재배 시 전조 방법별 점등시간

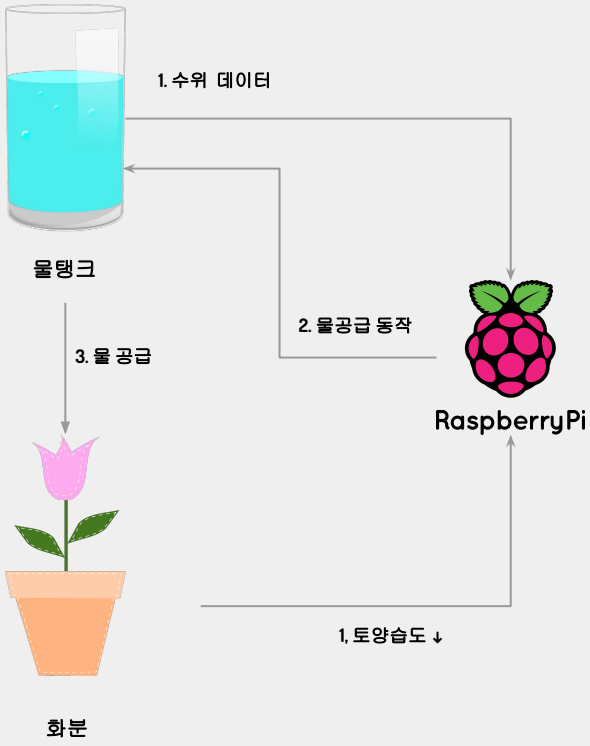
(표 38) 생육 단계별 전조 시간에 따른 상품수량

전조시간(회/일)	상품수량(g/주)				
	대왕	싼타	설향	레드펄 (육보)	아키히메 (장희)
강(12-12-12-8-4)	226.1	303.3	259.0	154.3	330.6
중(12-8-4-2-0)	238.7	297.3	300.6	170.4	298.6
약(12-6-2-0-0)	263.6	341.8	343.7	163.2	389.9
무	242.5	319.1	287.6	166.7	301.5

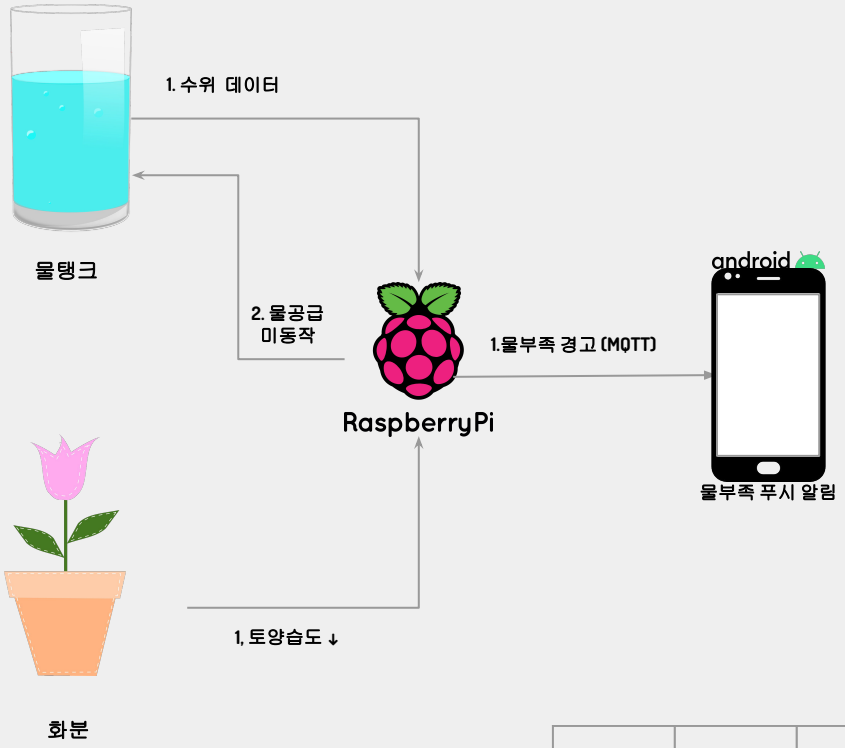
* 전조시간 : 출뢰기 – 수확 초기 – 수확 중기 – 수확 성기 – 수확 후기로 구분
전조광의 조도는 40~80lux— 30W 백열등 이용, 간헐 조명(10분/시간) 실시
고설식 수경재배 및 특성재배법 이용

단계별 추가 점등
품종 : 설향
전조방법 : 간헐 조명법
전조 시간 : 약

물주기(동작 흐름)



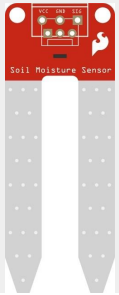
물이 충분한 경우



물이 부족한 경우

물주기(화분)

사용 장비

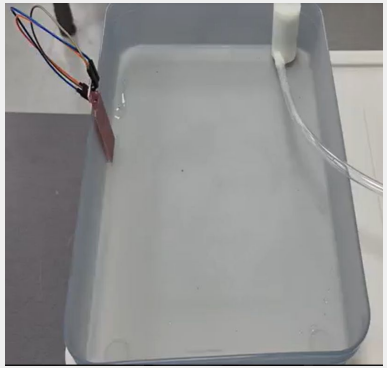


-토양습도센서
1. 화분의 수분량을 RPI로 전송
2. %로 판단하기 위해서 0~100% 로 변환

-워터펌프
1. 물탱크의 물을 화분으로 공급
2. 릴레이 사용하여 제어

-수위 센서
1. 물탱크 내 물의 양을 판단
2. 특정 값 이하면 안드로이드로 경고

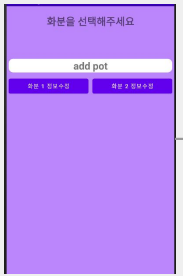
실제 동작 사진(화분)

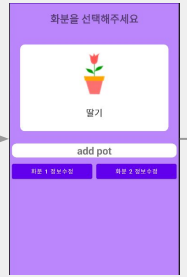


물주기(앱)

앱 동작

1. 화분 추가





화분 추가와
동시에 식물 설정

제한 토양수분,
물 공급량 설정

RaspberryPi

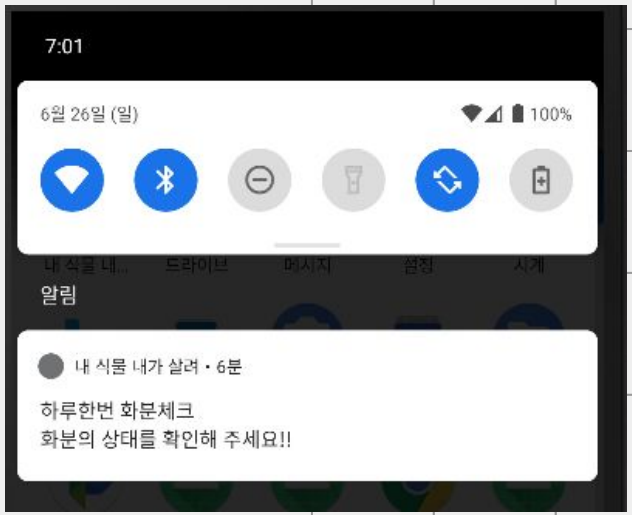
화분의 토양수분에 따라서
자동으로 물을 공급

2.물 부족시

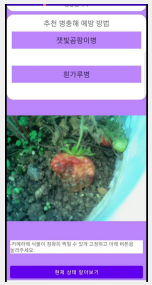
물공급시 남은 물이 공급량보다
부족하면 앱으로 신호전송

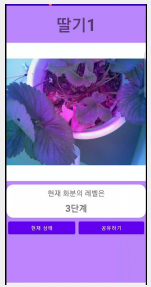
RaspberryPi

Firebase 클라우드 메시징 (FCM)



식물관찰 I





1. 생육단계, 질병판단 액티비티에서
해당하는 버튼 터치시

2.식물명+ 사진
촬영신호
Pub(MQTT)

8. 생육상태 + 병충해 데이터 Pub

RaspberryPi

3.사진 촬영
-해상도(2592x1944)

파이카메라

4. 사진+식물명 데이터
Pub

7. 생육상태 + 병충해 데이터 Sub

6. 생육상태 + 병충해 데이터 Pub

aws

mosquitto

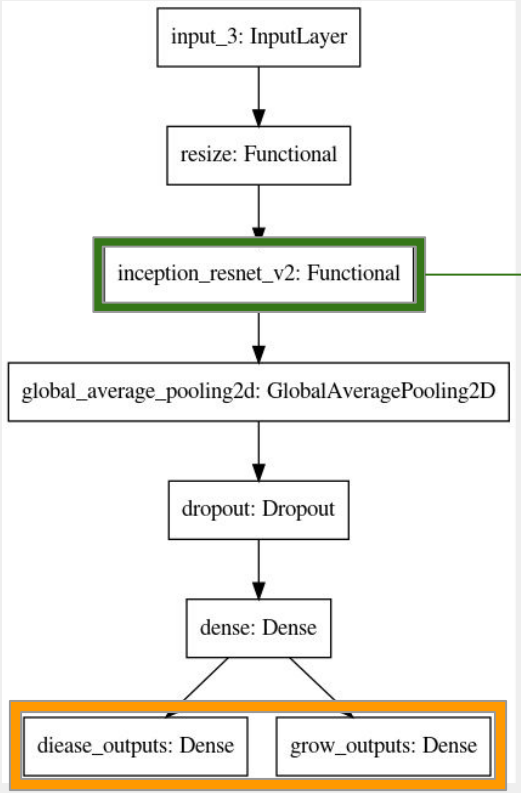
MQTT Broker(AWS)

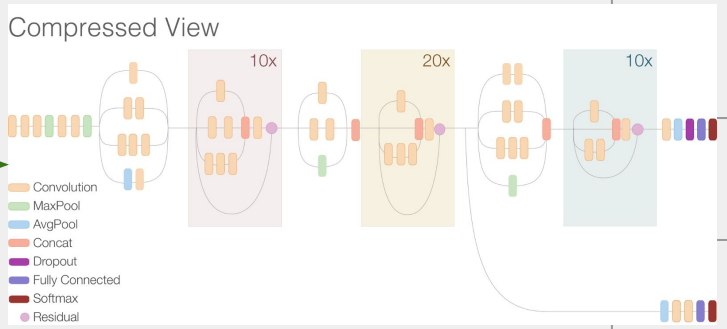
5. 사진+식물명 데이터 Sub

aws

AI Module(AWS)

식물 관찰 I





Multi-output model

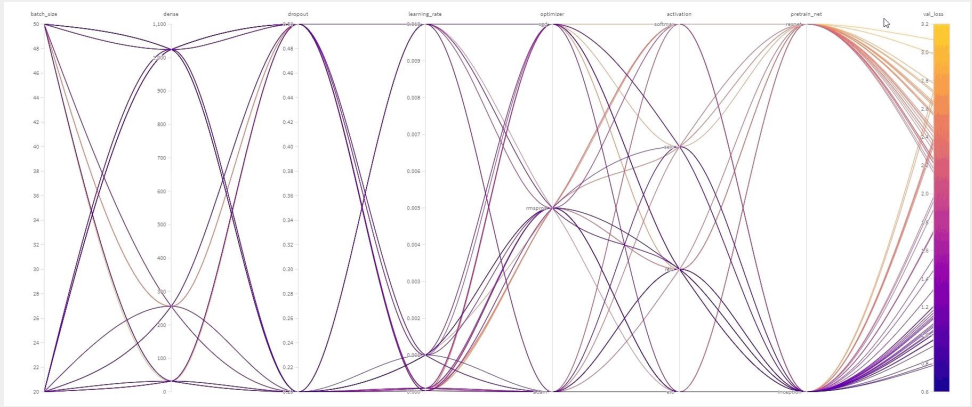
하나의 이미지로 두 가지 Task(질병, 생육단계 판단)를 수행하는 모델
Tensorflow Functional API로 구현

두 task가 공유하는 Backbone의 사전학습모델로
구글이 발표한 Inception Resnet V2를 최종 선택

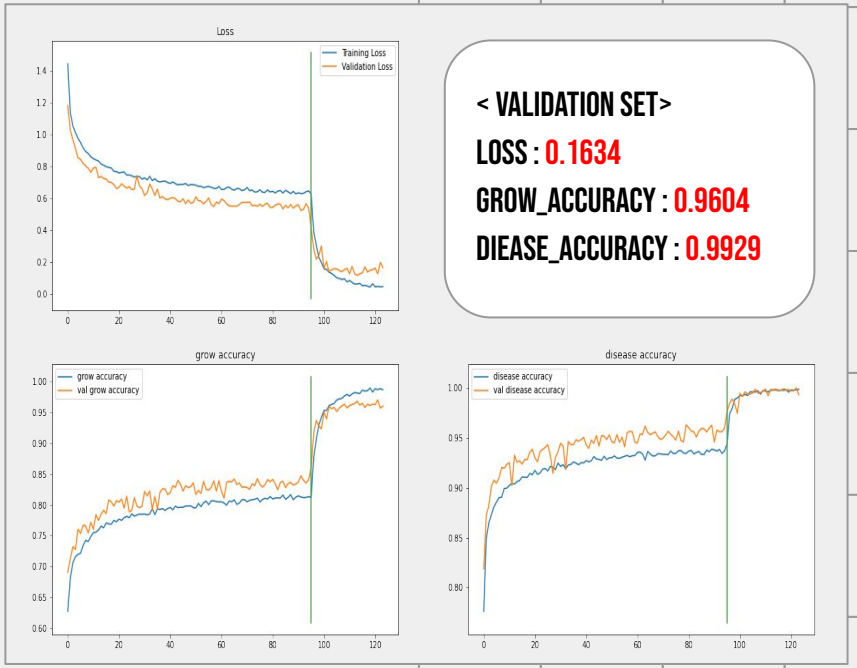
식물 관찰 I

Wandb 수행 결과 → 하이퍼 파라미터 종류

Pretrained_Net : inception, resnet
Activation : softmax, selu, relu, elu
Optimizer : adam, sgd, rmsprop
Batch_size : 20, 50
Dense : 32, 256, 1024
Dropout : 0.2, 0.5
Learning_rate : 1e-2, 1e-3, 1e-4, 3e-5



딸기 모델 Fine-tuning 전·후 비교



식물 관찰 I

Multi-output model

```
%%time
disease, grow = strawberry_model.predict(image)
disease = np.argmax(disease[0])
grow = np.argmax(grow[0])
result={'disease' : strawberry_dict['disease'][disease],
'grow' : strawberry_dict['grow'][grow]}
result
```

```
CPU times: user 1.25 s, sys: 691 ms, total: 1.94 s
Wall time: 653 ms
```

```
{'disease': '정상', 'grow': '4단계'}
```

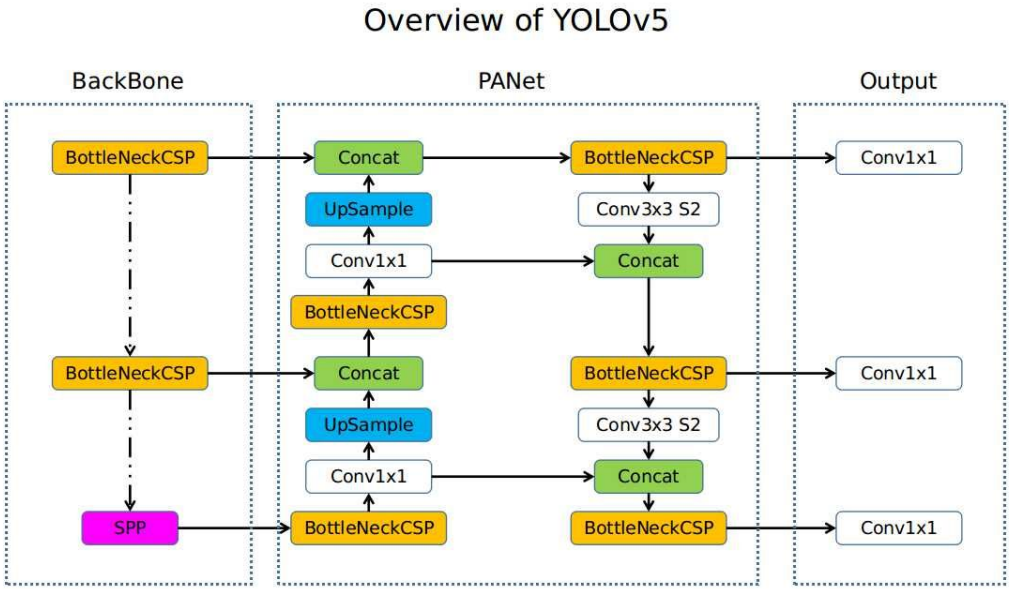
```
%%time
disease, grow = strawberry_model.predict(image)
disease = np.argmax(disease[0])
grow = np.argmax(grow[0])
result={'disease' : strawberry_dict['disease'][disease],
'grow' : strawberry_dict['grow'][grow]}
result
```

```
CPU times: user 1.22 s, sys: 779 ms, total: 2 s
Wall time: 650 ms
```

```
{'disease': '흰가루병', 'grow': '5단계'}
```

식물 관찰 Ⅱ

객체탐지 모델(YOLO v5)의 구조



크게 Backbone과 Output(Head) 부분으로 구성됨

Backbone은 CSP-Darknet를 사용하여 이미지로부터 Feature map을 추출함

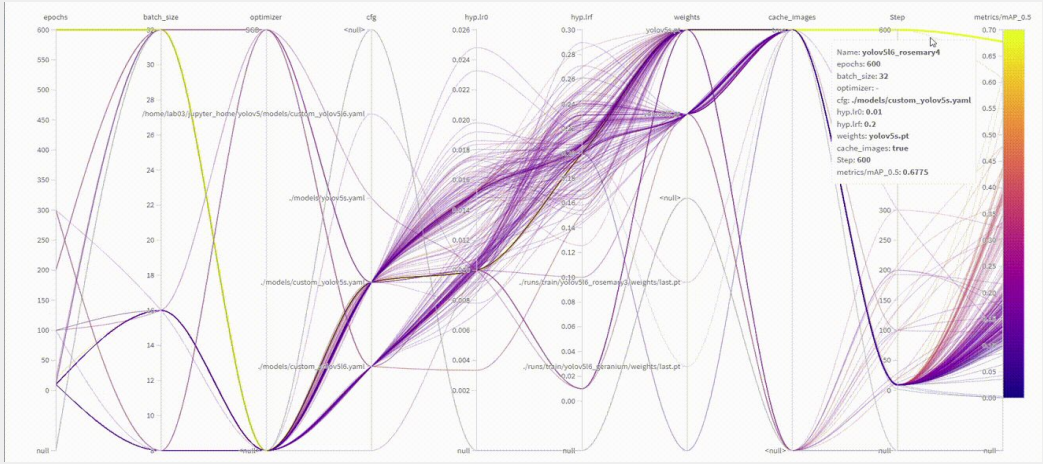
Output은 추출된 Feature map을 바탕으로 물체의 위치를 찾음

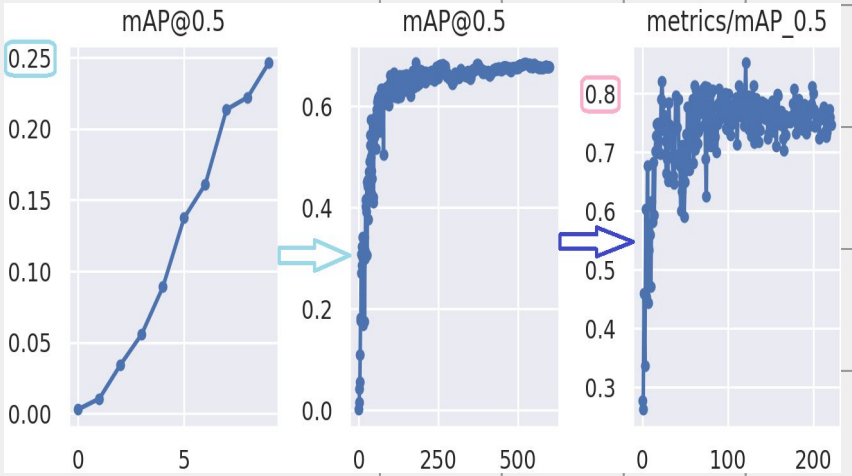
Anchor Box를 설정하고 이를 이용하며, 3가지의 scale에서 검출된 물체들(소/중/대)을 종합하여 최종적인 Bounding Box를 생성함

식물 관찰 Ⅱ

YOLO v5의 Wandb 수행 결과 → 하이퍼 파라미터 종류

Input Size : 224, 256, 416
Backbone : s, m, l, x
Pre-Trained_Weights : s, m, l, x
Pre-Hyperparameters : Objects365, scratch-low / med / high, VOC
Epochs : 10, 100, 300, 600
Batch_size : 8, 16, 32
Learning_rate : 1e-2, 2e-2, 3e-3, 7e-3, 8e-3



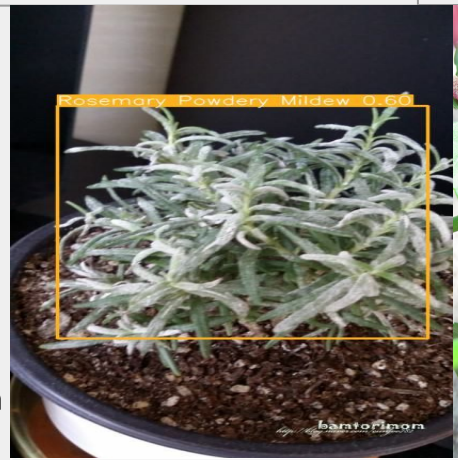


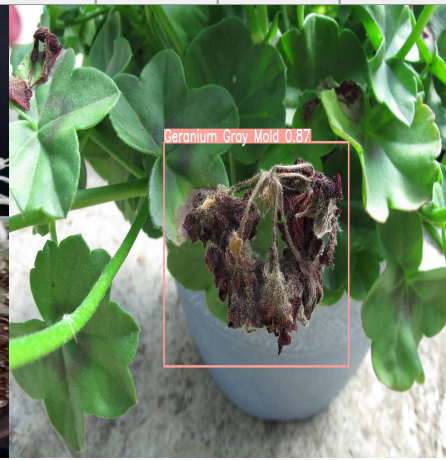
mAP? 모델이 얼마나 객체를 잘 검출하고 맞추는지!

식물 관찰 Ⅱ

로즈마리 흰가루병 | 제라늄 잿빛곰팡이병

Object Detection Model



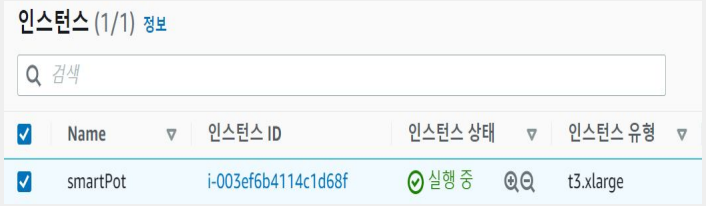



```
import torch

model = torch.hub.load('ultralytics/yolov5', 'custom', path='runs/train/best_geranium.pt', _verbose=False)
model.conf = 0.40
model.multi_label = False
model.max_det = 1

imgs = ['https://www.purduelandscapereport.org/wp-content/uploads/2020/06/Fig-3geranium-botrytis-blight.jpg']

results = model(imgs, size=256)
results.print()

b = results.pandas().xyxy[0]
print(str(b.values[-1][-1]))
```

```
Using cache found in /home/lab03/.cache/torch/hub/ultralytics_yolov5_master

image 1/1: 1365x2048 1 Geranium Gray Mold
Speed: 2233.7ms pre-process, 25.0ms inference, 1.2ms NMS per image at shape (1, 3, 192, 256)
Geranium Gray Mold
```

클라우드 서버

스마트화분 서버 (AI / IOT 모듈, MQTT 브로커 서버, S3 연동)



[스마트화분 EC2]

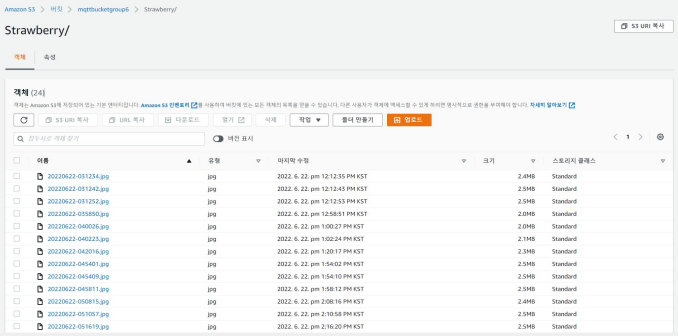
```
https://aws.amazon.com/amazon-linux-2/
(base) [ec2-user@ip-10-0-18-177 ~]$ cd image/
(base) [ec2-user@ip-10-0-18-177 image]$ ls
20220623-084402.jpg  20220624-074058.jpg  Geranium  Lettuce  Strawberry
(base) [ec2-user@ip-10-0-18-177 image]$ cd Strawberry/
(base) [ec2-user@ip-10-0-18-177 Strawberry]$ ls
20220622-031234.jpg  20220622-040223.jpg  20220622-050815.jpg  20220623-050108.jpg  20220623-050903.jpg
20220622-031242.jpg  20220622-042016.jpg  20220622-051057.jpg  20220623-050127.jpg  20220623-051107.jpg
20220622-031252.jpg  20220622-045401.jpg  20220622-051619.jpg  20220623-050138.jpg  20220623-051155.jpg
20220622-035850.jpg  20220622-045409.jpg  20220623-045058.jpg  20220623-050833.jpg  20220623-051415.jpg
20220622-040026.jpg  20220622-045811.jpg  20220623-045621.jpg  20220623-050842.jpg  20220623-064331.jpg
(base) [ec2-user@ip-10-0-18-177 Strawberry]$
```

[EC2에 마운트된 S3 저장소]

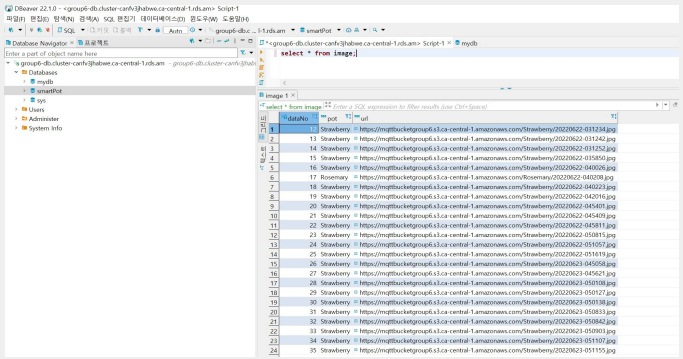
```
(base) [ec2-user@ip-10-0-18-177 Strawberry]$ systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker
   Loaded: loaded (/usr/lib/systemd/system/mosquitto.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2022-06-22 01:26:53 UTC; 3 days ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
 Main PID: 3514 (mosquitto)
   CGroup: /system.slice/mosquitto.service
           └─3514 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Jun 25 01:32:21 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656120741: New connection from 175.197.157.32 on port 1883.
Jun 25 01:32:22 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656120742: New client connected from 175.197.157.32 as paho201445758937460 (p2, c1, k60).
Jun 25 02:34:56 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656124496: Client paho100279654792004 has exceeded timeout, disconnecting.
Jun 25 02:35:01 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656124501: Client paho100280801649452 has exceeded timeout, disconnecting.
Jun 25 04:16:43 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656130603: Client paho201445758937460 disconnected.
Jun 25 04:16:45 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656130605: Client paho200931253826360 disconnected.
Jun 25 04:17:01 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656130621: New connection from 175.197.157.32 on port 1883.
Jun 25 04:17:01 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656130621: New client connected from 175.197.157.32 as paho211324790247460 (p2, c1, k60).
Jun 25 04:22:28 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656130948: New connection from 175.197.157.32 on port 1883.
Jun 25 04:22:28 ip-10-0-18-177.ca-central-1.compute.internal mosquitto[3514]: 1656130948: New client connected from 175.197.157.32 as paho211652753396460 (p2, c1, k60).
(base) [ec2-user@ip-10-0-18-177 Strawberry]$ cd ..
(base) [ec2-user@ip-10-0-18-177 image]$ cd ..
(base) [ec2-user@ip-10-0-18-177 ~]$ nohup python3 inference.py &
[1] 31750
(base) [ec2-user@ip-10-0-18-177 ~]$ nohup: ignoring input and appending output to 'nohup.out'
```

[모스키토 작동상태와 AI / IOT 모듈 실행]

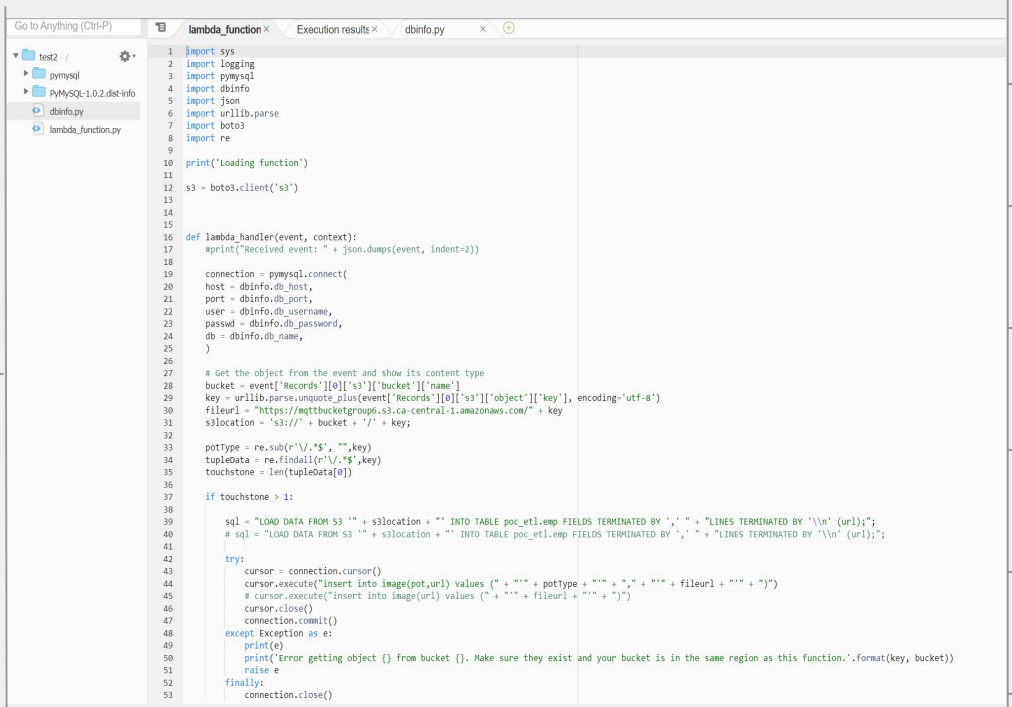
S3저장소&데이터베이스 연동



[AWS S3 저장소]



[AWS RDS]



[AWS Lambda]

- IOT 모듈에서 S3에 이미지를 저장하면 S3트리거를 통해 람다함수가 발동되며, RDS에 화분별로 이미지가 저장

최적 생육환경의 존재 여부 확인

사용자료

농촌진흥청
스마트팜 우수농가 공개데이터

조건

1.연속적 분포를 가정할 수 있어야 한다.
2.자료가 적어도 순위자료 이상의 자료여야함

가설

H_0 : A와 B가 같은 분포를 따른다.
H_1 : A와 B가 같은 분포를 따르지 않는다.

```
> ks.test(xxx,round(s21$inTp,number), exact = F)

        Two-sample Kolmogorov-Smirnov test

data:  xxx and round(s21$inTp, number)
D = 0.18453, p-value = 0.3373
alternative hypothesis: two-sided
```

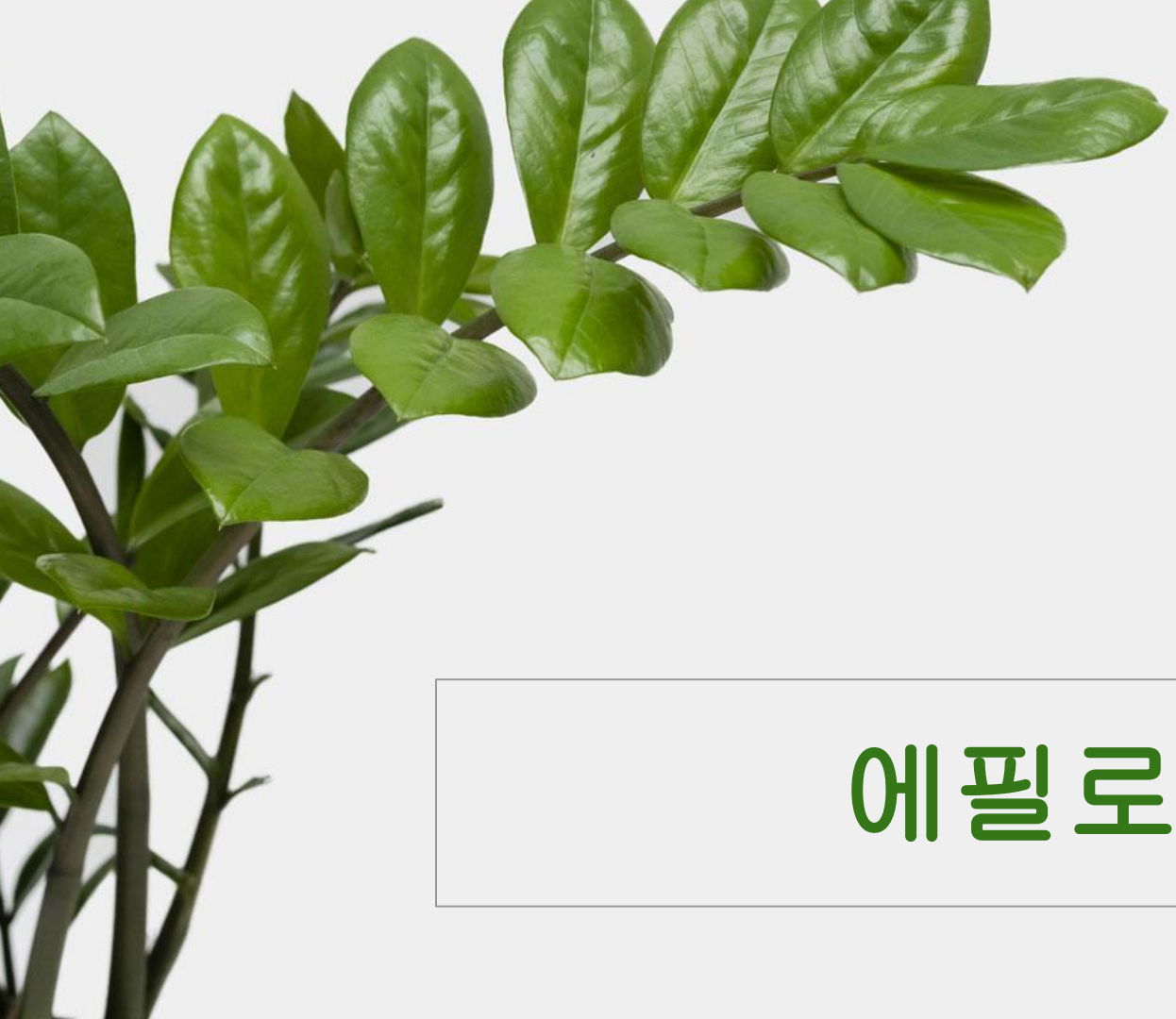
동질성 검정

	S21	S23	S26	S29	S31	S33	S37	S38	S41	S44	S45	S47	S67
S21	1.000	0.470	0.999	0.998	0.977	0.484	0.396	0.721	0.982	0.442	0.806	0.715	0.187
S23	0.470	1.000	0.618	0.270	0.069	0.960	0.000	0.834	0.074	1.000	0.792	0.524	0.156
S26	0.999	0.618	1.000	0.919	0.647	0.447	0.066	0.755	0.704	0.446	0.978	0.471	0.156
S29	0.998	0.270	0.919	1.000	0.998	0.216	0.372	0.312	0.840	0.200	0.372	0.125	0.053
S31	0.977	0.069	0.647	0.998	1.000	0.194	0.110	0.132	0.408	0.056	0.204	0.109	0.008
S33	0.484	0.960	0.447	0.216	0.194	1.000	0.128	0.668	0.330	0.923	0.795	0.554	0.504
S37	0.396	0.000	0.066	0.372	0.110	0.128	1.000	0.001	0.903	0.004	0.006	0.003	0.140
S38	0.721	0.834	0.755	0.312	0.132	0.668	0.001	1.000	0.114	0.951	0.910	0.994	0.241
S41	0.982	0.074	0.704	0.840	0.408	0.330	0.903	0.114	1.000	0.240	0.341	0.198	0.693
S44	0.442	1.000	0.446	0.200	0.056	0.923	0.004	0.951	0.240	1.000	0.984	0.980	0.502
S45	0.806	0.792	0.978	0.372	0.204	0.795	0.006	0.910	0.341	0.984	1.000	0.946	0.290
S47	0.715	0.524	0.471	0.125	0.109	0.554	0.003	0.994	0.198	0.980	0.946	1.000	0.419
S67	0.187	0.156	0.156	0.053	0.008	0.504	0.140	0.241	0.693	0.502	0.290	0.419	1.000

92.31%

	S21	S23	S26	S29	S31	S37	S38	S41	S44	S45	S47
S21	1.000	0.288	0.880	0.130	0.789	0.586	0.058	0.451	0.181	0.643	0.546
S23	0.288	1.000	0.185	0.986	0.109	0.039	0.953	0.070	0.666	0.111	0.763
S26	0.880	0.185	1.000	0.160	0.370	0.156	0.032	0.357	0.278	0.405	0.622
S29	0.130	0.986	0.160	1.000	0.020	0.009	0.470	0.019	0.711	0.057	0.617
S31	0.789	0.109	0.370	0.020	1.000	0.346	0.006	0.085	0.205	0.615	0.084
S37	0.586	0.039	0.156	0.009	0.346	1.000	0.002	0.096	0.265	0.801	0.044
S38	0.058	0.953	0.032	0.470	0.006	0.002	1.000	0.000	0.799	0.002	0.140
S41	0.451	0.070	0.357	0.019	0.085	0.096	0.000	1.000	0.146	0.192	0.202
S44	0.181	0.666	0.278	0.711	0.205	0.265	0.799	0.146	1.000	0.381	0.575
S45	0.643	0.111	0.405	0.057	0.615	0.801	0.002	0.192	0.381	1.000	0.232
S47	0.546	0.763	0.622	0.617	0.084	0.044	0.140	0.202	0.575	0.232	1.000

77.78%



4

에필로그

전망

B2C

다양한 홈가드닝
기기·서비스
증가

B2B

국내·외 기업의
플랜테리어 수요
증가

구현 예정 기능

커뮤니티

영양제
투여

기록보관소

팀별 역할

BIGDATA

프로젝트 당위성 확보
생육환경 동질성 분석
홈 가드닝 시장조사

CLOUD

스마트화분 실행 모듈 로컬이
아닌 클라우드 환경에서
작동되도록 구현

IOT

IoT센서 연동
MQTT통신 구현
안드로이드 어플

AI

식물의 생육 단계
병해충 판단 모델

팀 구성

BIG DATA

오병권
이준영

AI

박현상
이지연
이희경
조재성

IOT

이경훈
이채환

CLOUD

정민우

Rolling Paper

한 달 동안 행복했었습니다.
이제 모두 형,누나,오빠,동생
모두모두 고생하셨습니다💕
-준영

연락할게요~ 가끔 모여요들~
건강하게 지내세요~
화이팅~💪💪💪
-병권

좋은 팀원 만나서 즐겁게
작업할 수 있었습니다!
다들 꽃길만 걷길! 😉
- 재성

쉽지 않은 프로젝트였지만
좋은 분들과 함께해서 즐겁게
마칠 수 있었습니다 🙊
다들 취뽀 기원합니다!! 👏
- 희경

6조 너무 잘 만나서, 많이
배우고 즐겁게 작업했습니다!
같은 반 경훈님 특히 Thank U.
모두 취뽀 화이팅! ✌️- 채환

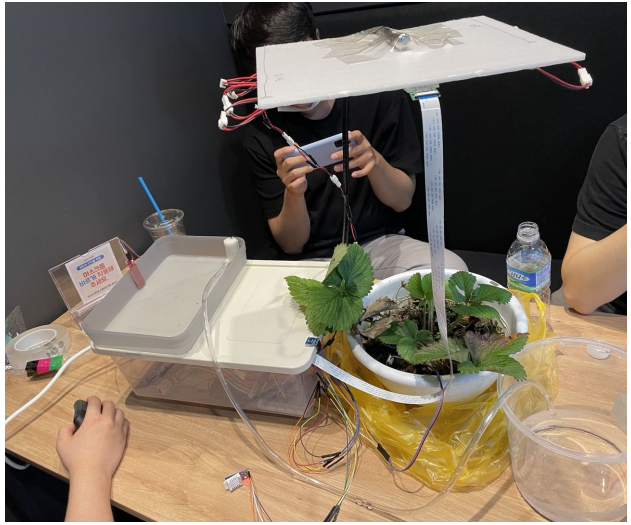
기획, 자료조사, 모델 튜닝이
생각보다 오래 걸리고
힘들었다! 😇 다들 고생
많았어요! 이제 성취뽀!
🐱💻
- 지연

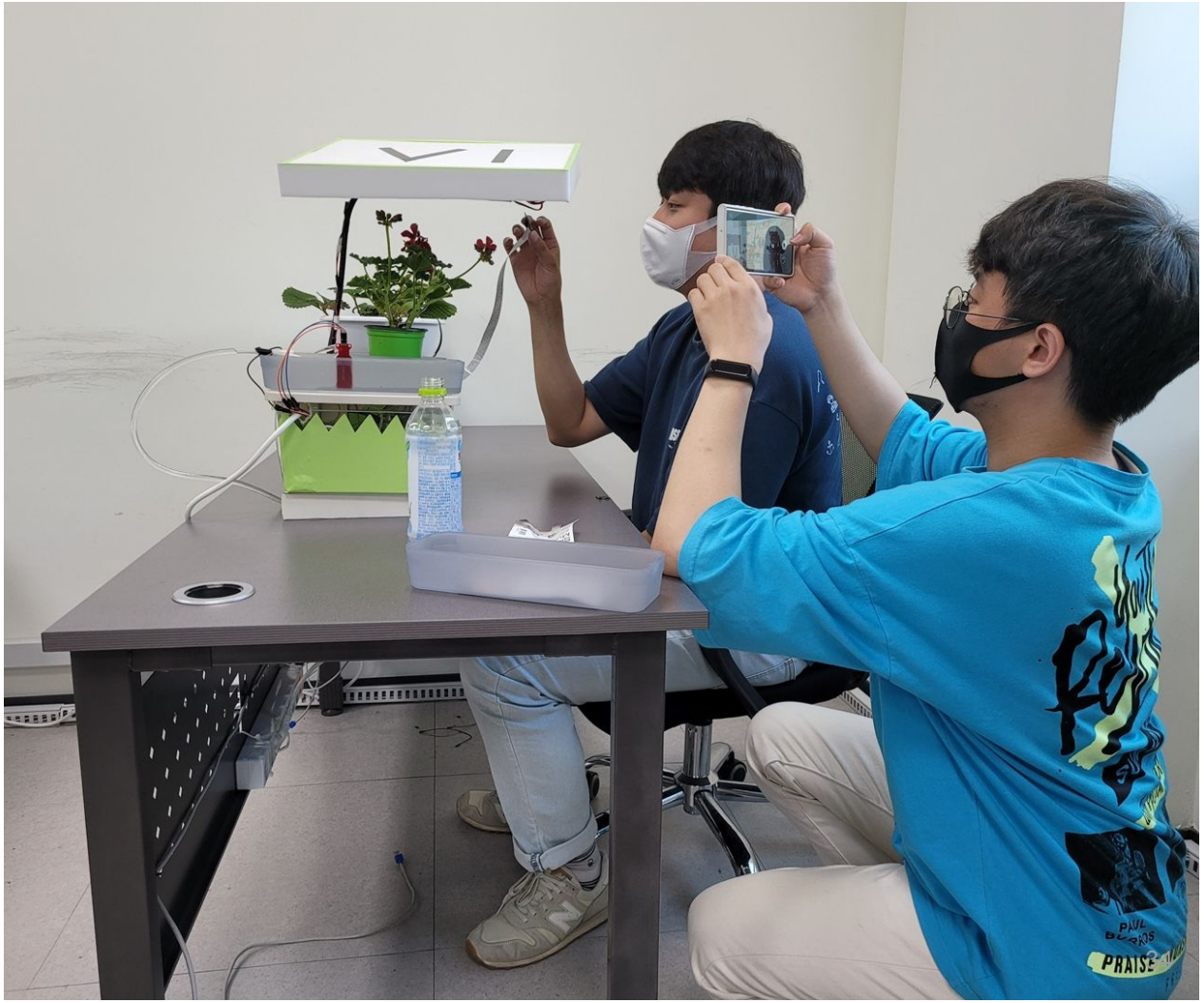
모두 열심히 해주셔서
감사하고.
좋은 결과가 나왔으면
좋겠네요!
-경훈

너무 좋은 조원분들
만났습니다! 그리고 많은 걸
배우는 프로젝트였습니다!😄
-현상

프로젝트 시작 때, 잘 해낼 수
있을까 걱정 많았는데, 우리
6조 팀원들 도움 덕분에 잘
진행 할 수 있었습니다~~!!
-민우 💪💪


MAGNIFICENT

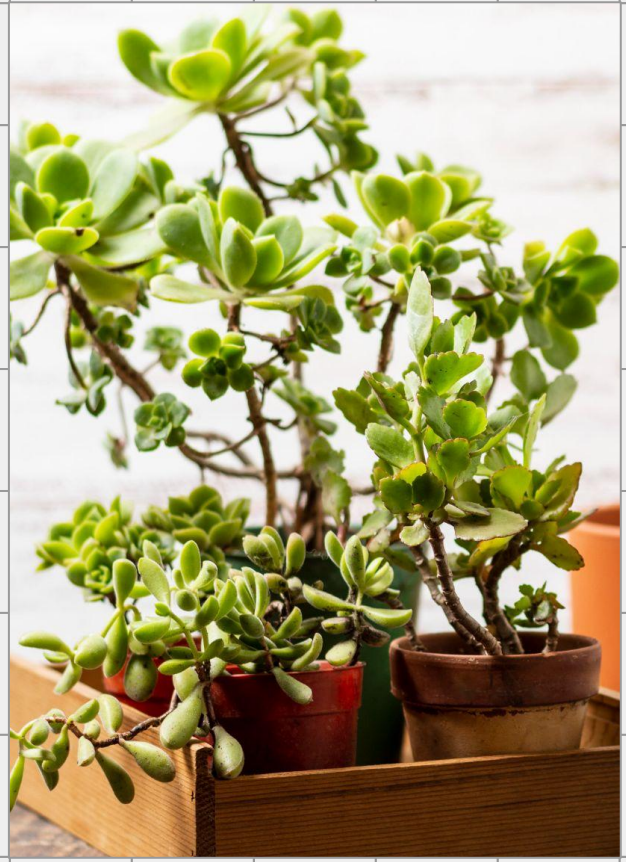


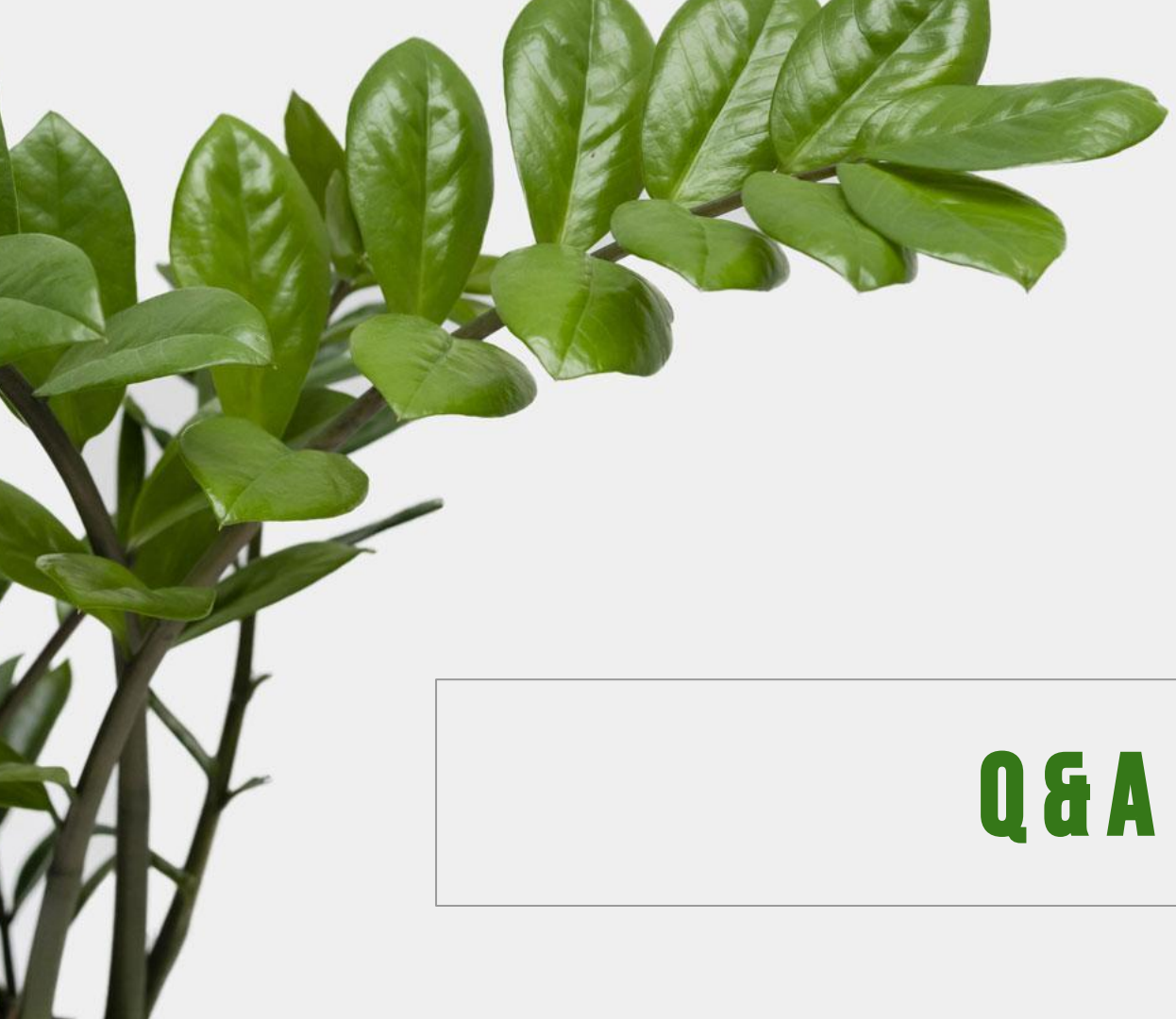

PRAISE


VI
MAGNIFICENT

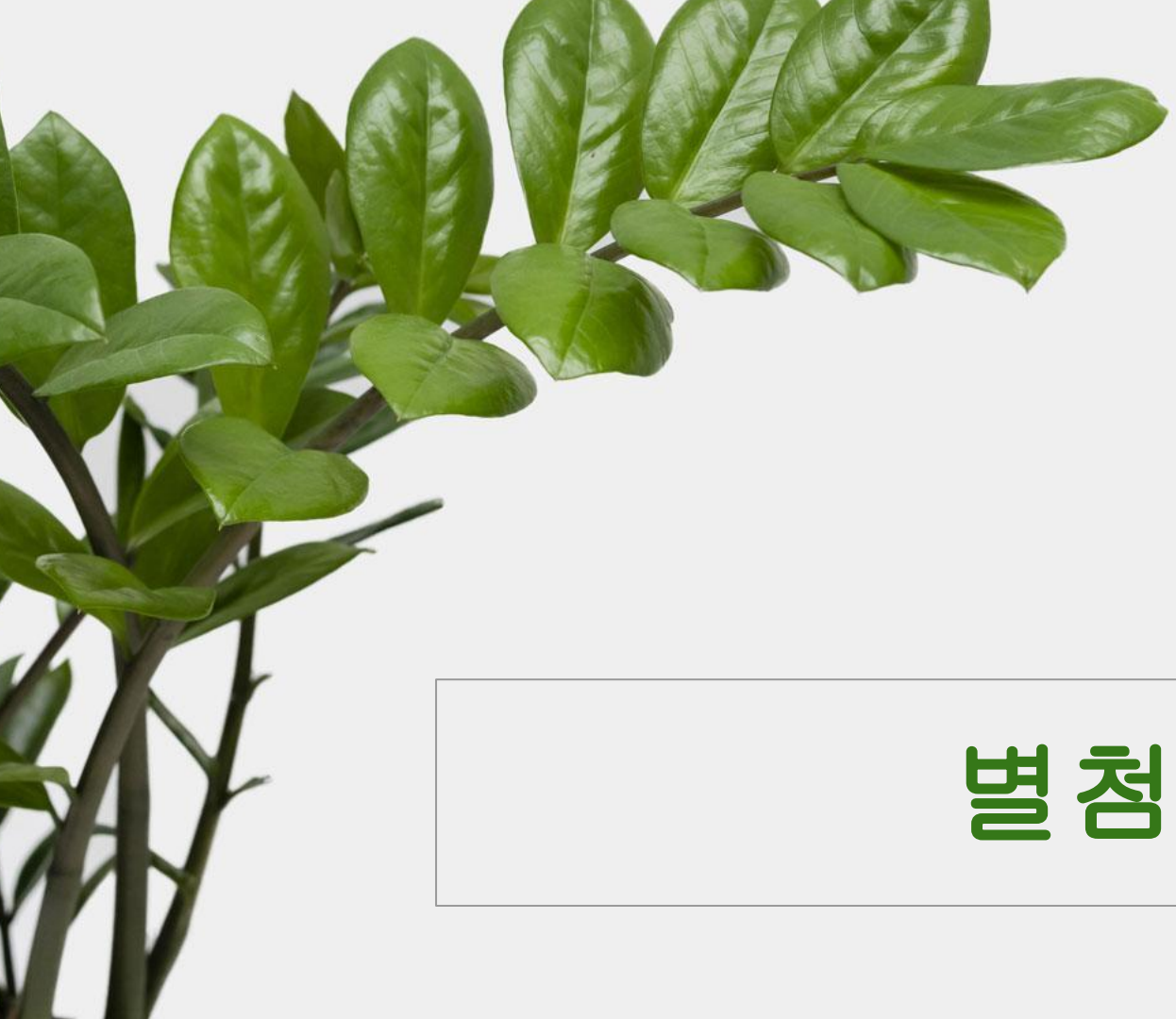

디지털 핵심 실무인재 양성훈련 (K-Digital Training)
"수료를 축하합니다"
multicampus
K-Digital 핵심
실무인재 양성사업
4차 산업혁명
선도인력양성
VI

감사합니다





Q&A



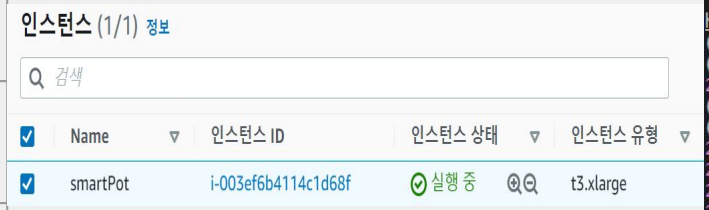
5

별첨

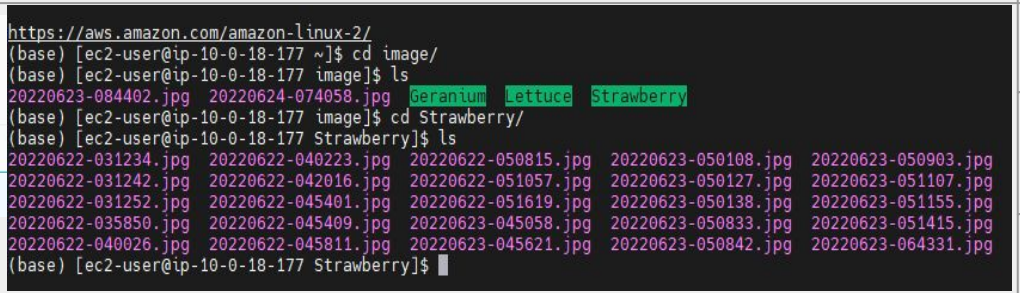
CLOUD

- 아키텍처
- AWS EC2
- AWS S3 TO RDS

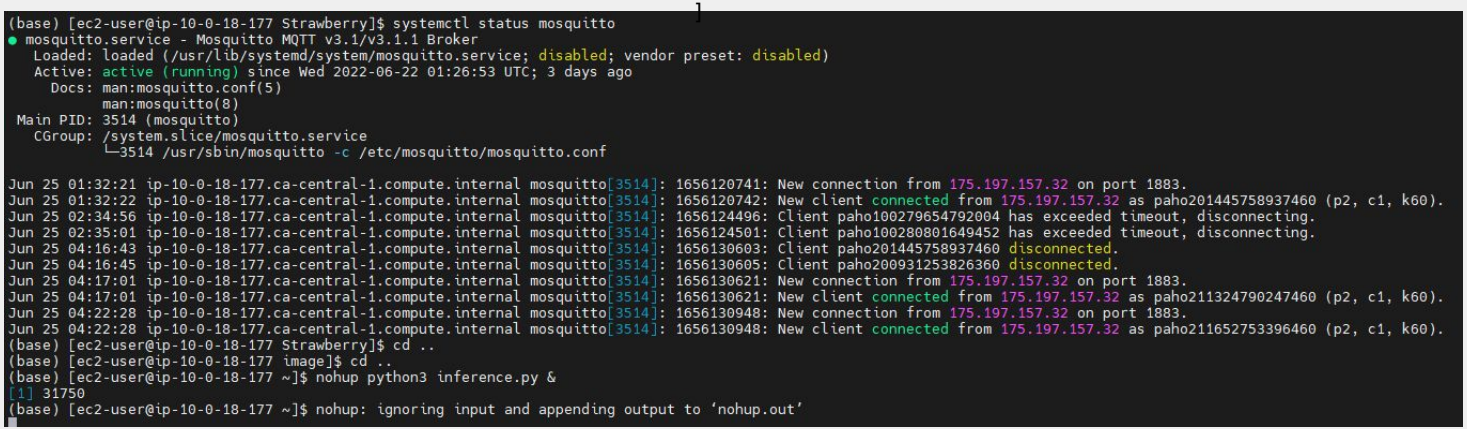
AWS EC2



[스마트화분 EC2]



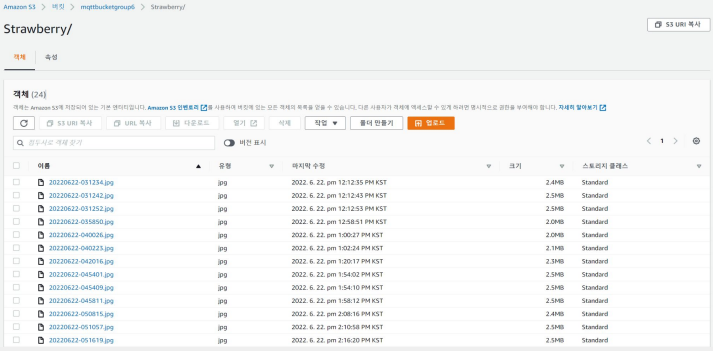
[EC2에 마운트된 S3 저장소]



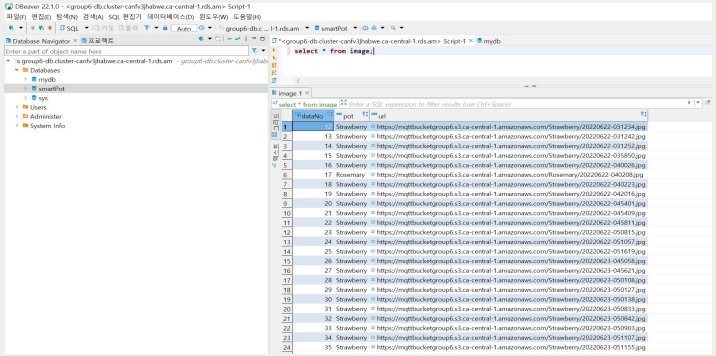
[모스키토 작동상태와 AI / IOT 모듈 실행]

- EC2에 이미지를 저장할 수 있는 S3를 마운트
- EC2에 MQTT 통신을 위한 모스키토의 작동과 AI / IOT 모듈을 실행하여, 모든 프로세스가 클라우드 상에서 처리

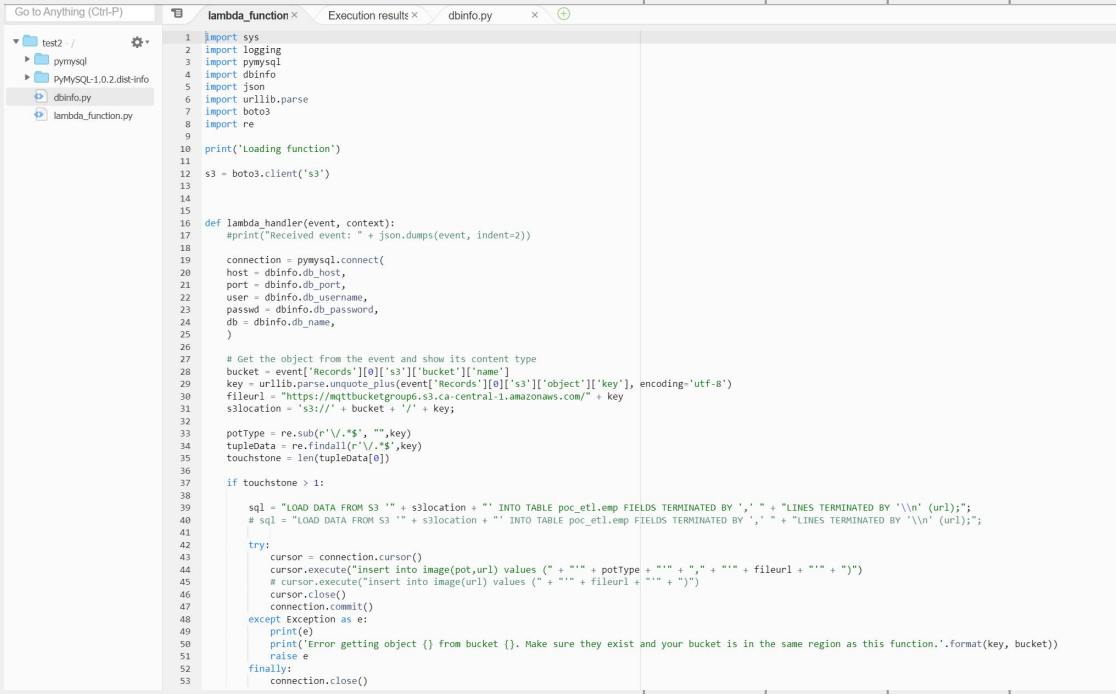
AWS S3 TO RDS



[AWS S3 저장소]



[AWS RDS]



[AWS Lambda]

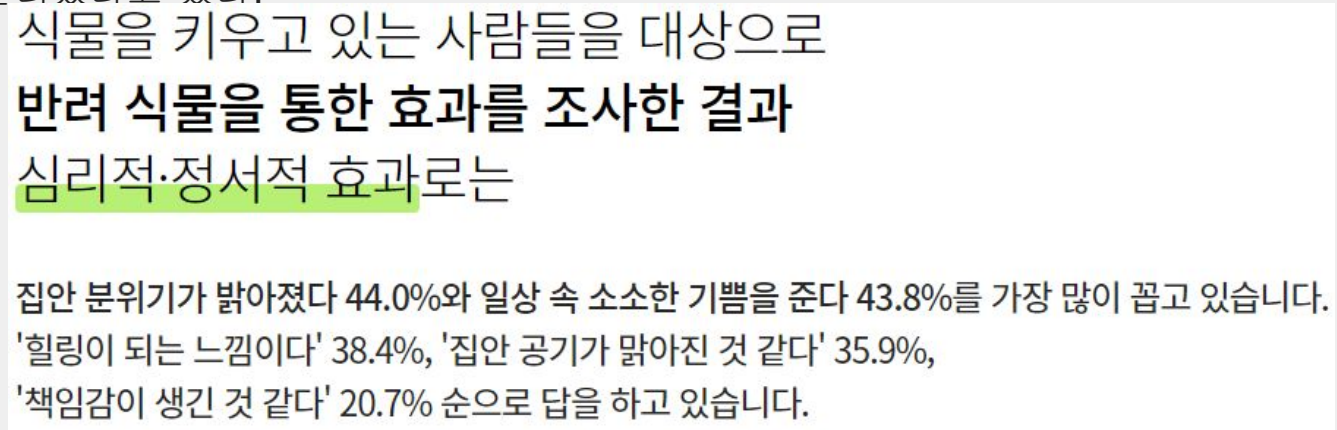
- IOT 모듈에서 S3에 이미지를 저장하면 S3트리거를 통해 람다함수가 발동되며, RDS에 화분별로 이미지가 저장

BIG DATA

반려식물의 심리적 효과와 시장성

심리적 효과

플랜테리어는 바이오필리아(Biophilia) 컨셉과 유사한데, 미국 생물학자 에드워드 윌슨은 1994년 자신의 저서에서 '생명'과 '사랑'의 합성어인 바이오필리아를 언급하며 '인간의 본질이 자연에서 비롯한 만큼 우리 유전자에는 살아있는 **생명체를 포용하고 곁에 두려는 욕망'**이 기저에 깔려있다고 했다.



출처 - 마이크로밀엠브레인, YTN사이언스

반려식물의 심리적 효과와 시장성

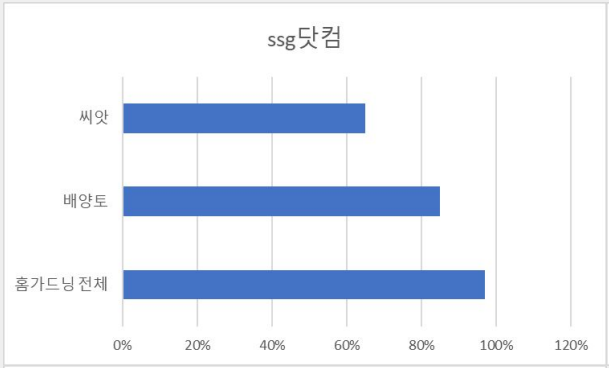
반려식물의 시장현황

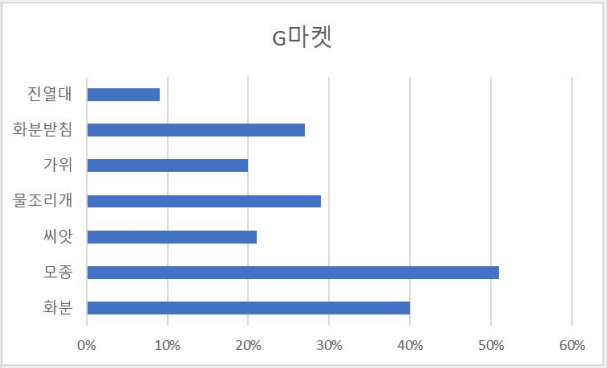
- ❏ 2019년 1분기만 해도 소셜네크워크서비스(SNS)상 반려식물 언급건은 150건 이하였으나 지난해 4분기 350건 이상으로 상승
- ❏ 식물재배 시장은 성장을 거듭하고 있다. 시장조사업체 마켓앤드마켓에 따르면 전 세계 실내 농업 시장은 연평균 9.4%씩 성장하며 지난해 145억달러(약 17조500억원)에서 2026년 248억달러(약 29조1600억원) 규모까지 커질 전망
- ❏ 롯데백화점에 따르면 지난해 '홈 가드닝' 관련 매출이 전년대비 2배 이상 신장한데 이어, 올해(1월1일~3월6일)도 전년 동기간 대비**70% 이상** 신장하며 지속적인 성장세를 유지하고 있다. -파이낸셜 뉴스
- ❏ 한국발명진흥회 지식재산평가센터가 발표한 자료에 따르면 국내 식물 재배기 시장 규모는 2019년 100억원에서 2020년 600억원으로 증가했다. 오는 2023년에는 5000억원까지 확대될 것으로 보인다.

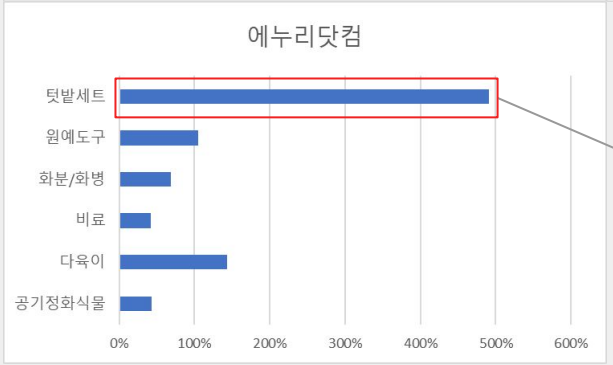
- 코로나19 이후 실내에서 식물키며 힐링하는 반려식물 문화 확산
- 유로모니터, 美가드닝시장 규모 2018년 402조 → 2023년 493조원
- 교원웰스, 가정용 식물재배기 코로나19속 판매급증 누적 5만대 돌파
- 반려식물 관리 앱 '그루우', 본엔젤스벤처에서 시드투자 유치 성공
- 식물 큐레이션 업체 '심다', 식물관리 서비스 '플리어리' 등 사업 활발

반려식물을 포함한 홈가드닝 용품 매출

e-commerce 대표 3사의 전년 대비 원예용품 매출 변화







데이터 크롤링을 통한 판매량 확인

-원예용품 등 전년 대비 최대 497% 이상 증가

반려식물 다양한 홈가드닝 기기·서비스

인테리어 가전으로 뜨는 식물재배기

LG ‘틔운 오브제컬렉션’
수분·통풍·온도 재배환경을 자동으로 관리, 수확 시기
가격 : 165만3900원

교원 ‘웰스팜’
무공해 채소가 자라는 모습을 관찰하고 직접 수확
가격 : 62만9000원

삼성물산 ‘포레어 슬림’
공기청정기가 결합된 식물재배기
가격 : 720만원

멀티캠퍼스 ‘네내’
최적 재배환경으로 식물 기르기를 돕고 질병과 생육단계까지 판단할 수 있는 **스마트 화분**

가격 : 15만원(예상)

반려식물위한 서비스

반려식물 호텔·병원

여행 등으로 집을 비울 때

반려동물호텔처럼 반려식물을 맡기면
물과 빛을 제공하며 돌봐주는
반려식물호텔 등장

반려식물을 돌봐주고 유기된 반려식물을
관리해 재분양.

식물 중고거래 진행, 분갈이 서비스
비용은 현재 무료지만 향후 **유료화**가
논의되고 있다.

'가든 어스' 식물호텔

'경기도 농업기술원'
식물병원

'화분 병원' 식물 병원

'허밍 그린' 식물 약국,
상담소

국내·외 기업의 플랜테리어 수요

세컨드 홈 인 할리우드
(Second Home in Hollywood)

이케아(IKEA)
오스트리아 빈 웨스트반호프역

더현대 서울
- 사운즈 포레스트 가든

국내·외 기업의 플랜테리어 수요

- 2019년에 오픈한 할리우드의 공유 오피스 '세컨드 홈 인 할리우드 (Second Home in Hollywood)'는 50여 년 된 건물을 레노베이션 함에 있어서 식물을 적극적으로 활용했다.

- 이케아(IKEA)는 현재 오스트리아 빈 웨스트반호프역에 수직정원을 콘셉트로 한 스토어를 만들고 있다. 현지 건축사무소 Querkraft Architekten이 디자인한 이 매장은 주차장은 없고 식물로 가득해 더욱 특별하다.

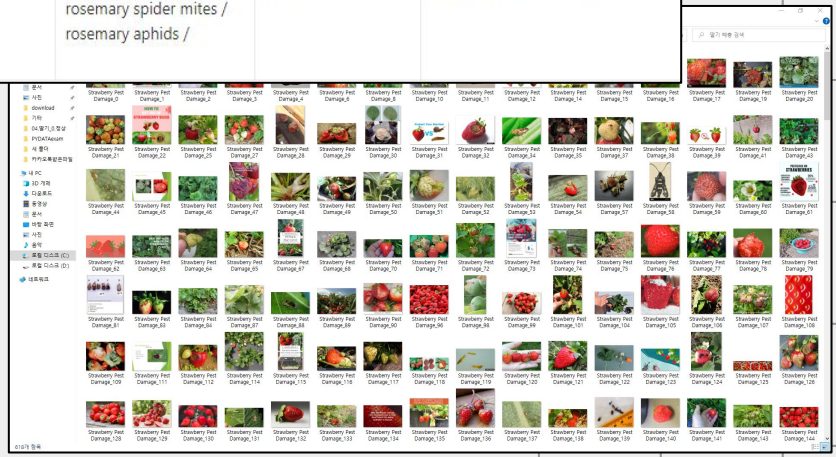
- 서울 최대 규모의 백화점 더현대서울이 '백화점엔 창문과 시계가 없다'는 낡은 공식을 깨고, 자연채광과 수풀이 우거진 공간을 선보이며 미래형 백화점의 청사진을 제시했다.

병충해 이미지 데이터 크롤링

```
from urllib.parse import quote_plus          # url 구문 분석을 위한 패키지
from bs4 import BeautifulSoup as bs

from selenium import webdriver
import time
from urllib.request import (urlopen, urlparse, urlunparse, urlretrieve)

chrome_path ='C:\Temp\chromedriver.exe'
base_url = "https://www.google.co.kr/imghp"

chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("lang=ko_KR") # 한국어
chrome_options.add_argument('window-size=1920x1080')



driver = webdriver.Chrome(chrome_path,chrome_options=chrome_options)
driver.get(base_url)
driver.implicitly_wait(3) # element가 로드될 때까지 지정한 시간만큼 대기할 수 있도록 설정
driver.get_screenshot_as_file('google_screen.png')

def selenium_scroll_option():
    SCROLL_PAUSE_SEC = 3
    last_height = driver.execute_script("return document.body.scrollHeight")
    while True:
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(SCROLL_PAUSE_SEC)
        new_height = driver.execute_script("return document.body.scrollHeight")
    if new_height == last_height:
        break
    last_height = new_height


from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
import urllib.request
import os
import pandas as pd


a=input("검색할 키워드 입력 : ")
image_name = input("저장할 이미지 이름 : ")
#b=int(input("몇 개 저장할래? : "))
driver = webdriver.Chrome(chrome_path)
driver.get('http://www.google.co.kr/imghp?hl=ko')
browser = driver.find_element_by_name("q")
browser.send_keys(a)
browser.send_keys(Keys.RETURN)

selenium_scroll_option()
driver.find_elements_by_xpath('//*[@id="islmp"]/div/div/div/div[1]/div[2]/div[2]/input')[0].click()
selenium_scroll_option()
```

Crawling Keywords

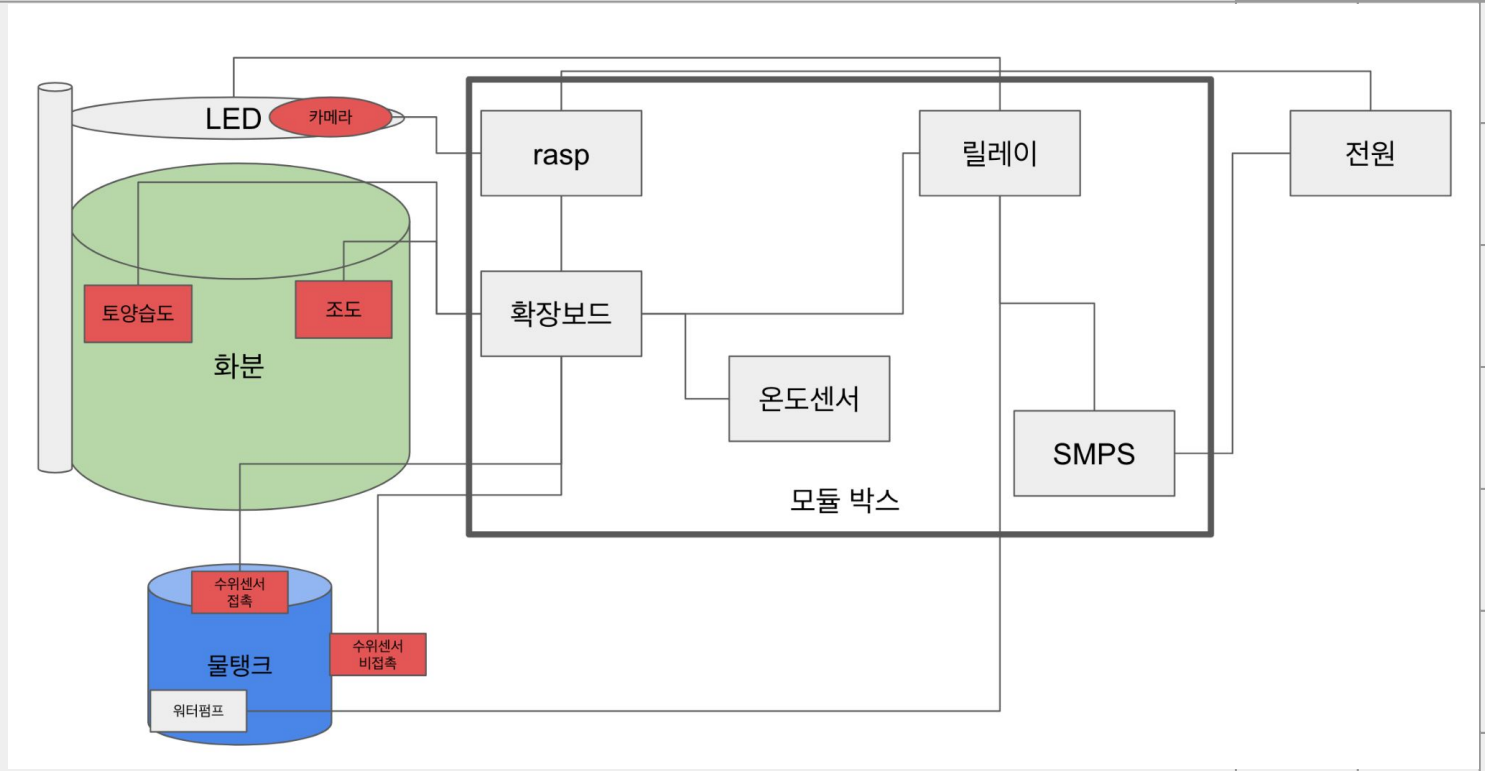
딸기 해충	Strawberry Pest Damage / Strawberry Pests	로즈마리 흰가루병	rosemary Powdery mildew
상추 해충	Romaine Lettuce Pests / Leaf Lettuce Pests / 상추 벌레 피해	로즈마리 점무늬병	Rosemary Entomosporium leaf spot (caused by Corynespora cassiicola)
제라늄 해충	Geranium Pests	제라늄 갈색무늬병	geranium leaf spot / leaf spot
로즈마리 해충	Rosemary Pests/ rosemary spider mites / rosemary aphids /	제라늄 잿빛곰팡이병	geranium botrytis cinerea



IOT

- 화분 구조도
- MQTT
- 장비 및 통신
- 동작흐름
- Android Application
- Raspberry PI
- Firebase

IOT 화분 구조도


LED
카메라
rasp
릴레이
전원
토양습도
조도
화분
확장보드
온도센서
SMPS
모듈 박스
수위센서
접촉
물탱크
수위센서
비접촉
워터펌프

IOT MQTT

-mqtt 클라이언트로 paho 라이브러리를 사용한다.

-AWS에 mosquitto를 설치하여 브로커로 사용함
-AWS에 사용하므로 어느 곳에서든 화분의 상태를 알 수 있다.

```
implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.2.0'
implementation 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
```

-안드로이드에서 사용한 라이브러리

```
import paho.mqtt.client as mqtt
import paho.mqtt.publish as publisher
```

-라즈베리파이에서 사용한 라이브러리

IOT 장비 및 통신

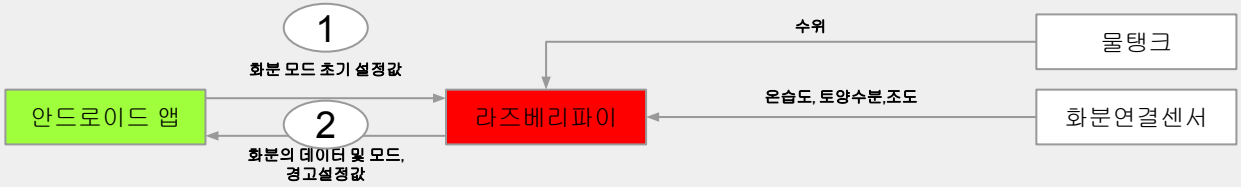
장비

- 토양수분센서, 조도센서, 온습도센서, 수위센서, 워터펌프, 릴레이, LED, SMPS

통신

- mqtt 브로커를 AWS 서버에 구축하여 사용
- mqtt를 사용하여 센서값, 이미지를 서버에 송신 후 생육상태, 병충해 결과를 라즈베리파이가 수신하여 처리

IOT_ 동작 흐름


1
화분 모드 초기 설정값
안드로이드 앱
2
화분의 데이터 및 모드,
경고설정값
라즈베리파이
수위
물탱크
온습도, 토양수분,조도
화분연결센서


화분을 선택해주세요
딸기
add pot
화분 1 정보수정
화분 2 정보수정

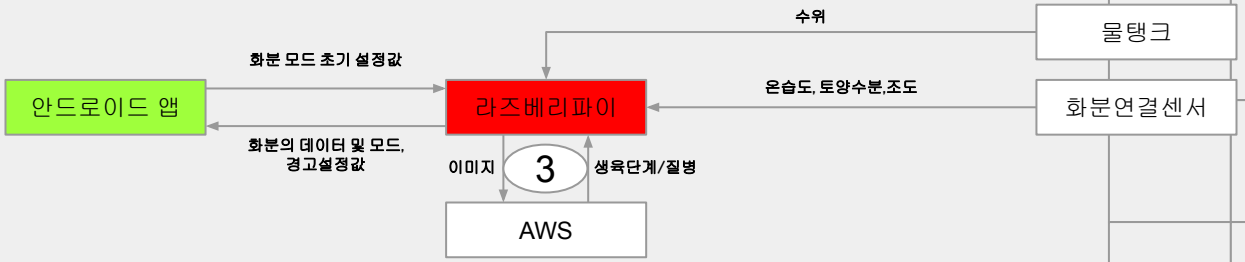

딸기
종류: Strawberry
현재 생육 상태
00%
레벨 확인
대기 온습도
20℃ 20%
조도
00lux
토양 습도
00%
병충해
정상
질병 판단
물탱크
00ml
구현예정

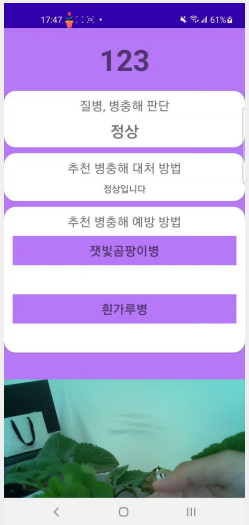

현재 조도 : 22lux
종류: Strawberry
현재 생육 상태
5단계
레벨 확인
대기 온습도
26.0℃ 59%
조도
22lux
토양 습도
95%
병충해
정상
질병 판단
물탱크
325ml
구현예정


물탱크 물 부족 물탱크의 물이 부족합니다
종류: Strawberry
현재 생육 상태
5단계
레벨 확인
대기 온습도
26.0℃ 59%
조도
512lux
토양 습도
95%
병충해
정상
질병 판단
물탱크
0ml
구현예정

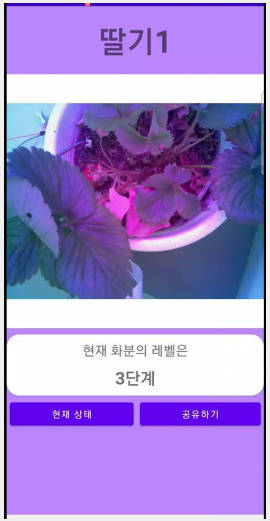

토양 수분 경고 2.71
종류: Strawberry
현재 생육 상태
5단계
레벨 확인
대기 온습도
25.7℃ 59%
조도
572lux
토양 습도
2.71%
병충해
정상
질병 판단
물탱크
325ml
구현예정

IOT_ 동작 흐름


수위
물탱크
화분 모드 초기 설정값
안드로이드 앱
라즈베리파이
온습도, 토양수분,조도
화분연결센서
화분의 데이터 및 모드,
경고설정값
이미지
3
생육단계/질병
AWS

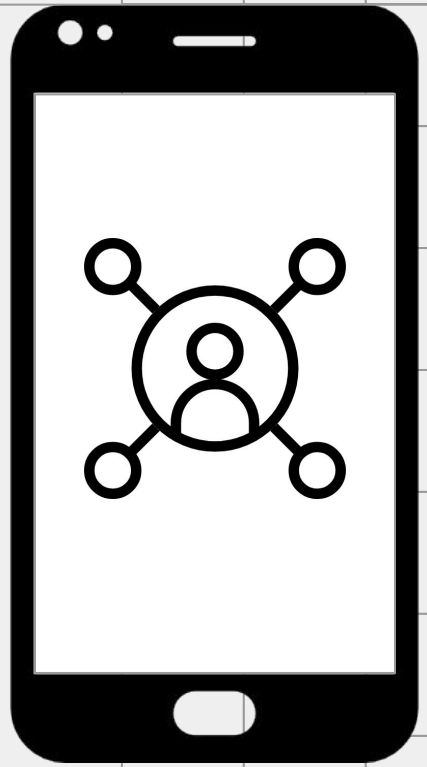

123
질병, 병충해 판단
정상
추천 병충해 대처 방법
정상입니다
추천 병충해 예방 방법
잿빛곰팡이병
흰가루병


추천 병충해 예방 방법
잿빛곰팡이병
흰가루병
현재 상태 알아보기


딸기1
현재 화분의 레벨은
3단계
현재 상태
공유하기

IOT_ANDROID APPLICATION

- 화분의 레벨 공유
- SharedPreferences
  - 화분 목록 초기화 방지
- Firebase를 통한 푸시 알림
- listView, recyclerView, intent



IOT_ANDROID APPLICATION

ANDROID

- 온습도, 조도, 토양습도, 물탱크 수위, 질병,병충해, 생육상태 확인 가능
- AWS의 AI에서 받는 결과에 따라서 경고 값, 공급 물 양 변화

IOT_ANDROID APPLICATION(병충해, 생육상태)

ANDROID

- 인텐트를 사용하여 화분 이름과 식물 이름을 각각의 액티비티로 전송

```
diseaseChk.setOnClickListener {  it: View!
    val disintent = Intent( packageContext: this, diseaseChkActivity::class.java).apply {
        putExtra( name: "potname", value: "${pot1name.text}")
        putExtra( name: "plantname", value: "${plantName.text}")
    }
    startActivity(disintent)
}
levelChk.setOnClickListener {  it: View!
    val levelIntent = Intent( packageContext: this,potLevelActivity::class.java).apply{
        putExtra( name: "potname",pot1name.text.toString())
        putExtra( name: "plantname",plantName.text.toString())
    }
    startActivity(levelIntent)
}
```

IOT_ANDROID APPLICATION(화분 추가시 데이터 유지)

ANDROID

```
when(potlist){ // 화분 갯수에 따라서 추가하는 화분의 키가 달라짐
    1 -> {
        result.data?.getStringExtra( name: "potname")?.let { myApp.prefs.setString( key: "pot1name", it) }
        result.data?.getStringExtra( name: "plantname")?.let { myApp.prefs.setString( key: "pot1plantname", it) }
        mymqtt?.publish( topic: "mode/pot1PlantName", result.data?.getStringExtra( name: "plantname")!!)
        myApp.prefs.setString( key: "potlist", str: "1")
    }
    2 -> {
        result.data?.getStringExtra( name: "potname")?.let { myApp.prefs.setString( key: "pot2name", it) }
        result.data?.getStringExtra( name: "plantname")?.let { myApp.prefs.setString( key: "pot2plantname", it) }
        myApp.prefs.setString( key: "potlist", str: "2")
    }
}
```

- 화분 추가시 SharedPreferences 사용하여 앱 종료해도 데이터 유지

IOT_ANDROID APPLICATION(안드로이드MQTT)

ANDROID

```
fun onReceived(topic:String, message: MqttMessage){
    when(topic){
        sub_topic -> {
            val msg = message.payload
            val image= BitmapFactory.decodeByteArray(msg, offset: 0, msg.size);
            image1 = image
            imageFlag = 1
            levelchkcamera.setImageBitmap(image)
            saveImageUpAndVerQ(image1)
        }
```

- 대부분 토픽별로 이미지 저장, 데이터 변경

IOT_ RASPBERRY PI(센서)

```
def run(self):
    try:
        while True:
            # 수위(접촉), 토양습도, 조도 센서에서 값을 얻을 예정
            luxBytes = self.i2c.read_i2c_block_data(self.BH1750_DEV_ADDR, self.CONT_H_RES_MODE, 2)

            # 바이트 배열을 int로 변환
            lux = int.from_bytes(luxBytes, byteorder='big')

            self.waterLevel = self.readadc(self.pot_channel1)
            self.soilWater = self.readadc(self.pot_channel2)
            # 토양 수분 값을 0~100으로 변환하는 코드
            new_soilWater_value = round((((self.soilWater - 0) / (700 - 0)) * (100 - 0) + 0), 2)
            new_waterLevel_value = round((((self.waterLevel - 0) / (650 - 0)) * (100 - 0) + 0), 2)

            # 물 주는 버튼을 꼭 사용해야 할까? -> 자동으로 주면 되는데
            publisher.single("sensor1/every",
                             str(new_waterLevel_value) + ":" + str(new_soilWater_value) + ":" + str(
                                 self.getdht11.gettemp11()) + ":" + str(lux) + ":" + str(
                                 self.getdht11.gethumid11()), hostname="3.96.178.99")
            print("----------------------------")
            print("waterLevel: %d" % self.waterLevel)
            print("soilWater: %d" % self.soilWater)
            print("humidity: %f" % self.getdht11.gethumid11())
            print("temperature: %f" % self.getdht11.gettemp11())
            print("lux: %d" % lux)
```

- MCP3008을 통해 아날로그 센서 값을 spi 통신으로 받음
- 수위, 토양수분, 온습도 값을 받아 앱으로 전송
- 대기 온습도는 10초마다 최신화

```
def run(self):
    try:
        while True:
            # 온습도 값 가져오기
            try:
                self.humidity_data = self.mydht22.humidity
                self.temperature_data = self.mydht22.temperature
            except RuntimeError as error:
                print(error.args[0])
            finally:
                pass
            time.sleep(10)
    except KeyboardInterrupt:
        pass
    finally:
        pass
```

IOT_ RASPBERRY PI(이미지 전송)

```
if message.topic == "iot/actLevelCamera":
    print("1111111111111111111111111111")
    self.camera.opCamera("levelDisease", self.mode)

# 질병 액티비티에서 카메라 동작시
elif message.topic == "iot/actDiseaseCamera":
    print("2")
    self.camera.opCamera("levelDisease", self.mode)
```

←레벨, 질병 토픽으로 이미지를 AWS로 보냄

```
def opCamera(self,mode,plantname):
    self.plantname = plantname
    self.camera.start_preview()
    sleep(5)
    self.camera.capture("/home/pi/mywork/basic/image_prac/test"+"10"+".jpg")
    f = open("/home/pi/mywork/basic/image_prac/test"+"10"+".jpg", 'rb')
    a = f.read()
    fimage = bytearray(a)
    f.close
    if mode == "periodic": # 주기적으로 보낼 때 동작하는 코드
        publisher.single("iot/awsperiodic/"+plantname,fimage,hostname="3.96.178.99")
    elif mode == "levelDisease": # 레벨이나 질병 페이지에서 신호를 보낼 때 동작하는 코드
        publisher.single("iot/awslevelDisease/"+plantname,fimage,hostname="3.96.178.99")
    self.camera.stop_preview()
```

←이미지를 AWS로 보낼 때 사용되는 메소드
토픽에 식물명을 붙여서 이미지와 같이 전송

IOT_ RASPBERRY PI(생육/병충해 수신)

```
elif message.topic == "iot/aiToIotValue":
    print("3")
    val = message.payload.decode("utf-8").split('/')
    print(val[0] + val[1] + ".")
    self.growmode = int(val[0])# 생육상태 + 질병상태 변수를 변경
    if self.growmode == 1:
        self.waterSpoutValue = 100
        self.ledOnTimeMode = 1
        self.soilWaterLevel = 80.0
    elif self.growmode == 2:
        self.waterSpoutValue = 50
        self.ledOnTimeMode = 1
        self.soilWaterLevel = 85.0
    elif self.growmode == 3:
        self.waterSpoutValue = 50
        self.ledOnTimeMode = 1
        self.soilWaterLevel = 85.0
    elif self.growmode == 4:
        self.waterSpoutValue = 100
        self.ledOnTimeMode = 1
        self.soilWaterLevel = 85.0
    elif self.growmode == 5:
        self.waterSpoutValue = 100
        self.ledOnTimeMode = 1
        self.soilWaterLevel = 85.0
```

```
    elif self.mode == "Lettuce" and self.growmode == 1:
        self.growmode = 8  # 상추 1단계 - 8 상추 2단계 - 9
        self.waterSpoutValue = 100
        self.ledOnTimeMode = 2
        self.soilWaterLevel = 85.0
    elif self.mode == "Lettuce" and self.growmode == 2:
        self.growmode = 9  # 상추 1단계 - 8 상추 2단계 - 9
        self.waterSpoutValue = 100
        self.ledOnTimeMode = 2
        self.soilWaterLevel = 85.0
    elif self.growmode == 7 and self.mode == "Rosemary":
        self.waterSpoutValue = 150
        self.ledOnTimeMode = 3
        self.soilWaterLevel = 85.0
    elif self.growmode == 7 and self.mode == "Geranium":
        self.waterSpoutValue = 200
        self.ledOnTimeMode = 4
        self.soilWaterLevel = 90.0
    self.disease = val[1]
    self.led.setLed(1,4) # led 설정변경
```

↑AWS 에서 AI를 통해 얻은 생육상태에 따라서 LED on/off 시간 및 물 공급량 변경

IOT_ RASPBERRY PI(물공급)

```
def waterPumpAct(self):
    try:
        self.count = 0
        while True:
            if (self.getSoilWaterData() < self.soilWaterLevel):  # 토양 수분 값이 정해진 제한 값 보다 작아지면
                currentWaterLevel = int(self.getWaterData() * 5)  # 맨 처음 물 양을 저장 500ML 기준
                if currentWaterLevel < self.waterSpoutValue:  # 물이 특정 값보다 적으면 동작하지 않음
                    print("물이 적음")
                    # 안드로이드로 물이 부족하다는 것을 전송하는 코드 작성
                    publisher.single("sensor1/waterWarn", "물부족", hostname="3.96.178.99")
                    time.sleep(1)
                else:  # 물이 특정 값보다 커지면 (정상상태)
                    while (currentWaterLevel - int(
                            self.getWaterData() * 5) < self.waterSpoutValue):  # 설정된 값보다 빠진 물이 작은동안만 무한루프
                        if self.count == 0:
                            self.waterPump.waterOpen()  # 펌프 오픈(1번만)
                            self.count = 1
                        time.sleep(0.5)
                    self.waterPump.waterOff()  # 무한루프에서 나오면 펌프를 닫음
                    self.count = 0
            time.sleep(1)
    except KeyboardInterrupt:
        pass
    finally:
        pass
```

←이전에 설명한 생육상태/식물종류에 따라서 물 공급량이 변화

IOT_ RASPBERRY PI(PI CAMERA)

```python
def opCamera(self, mode, plantname):
    self.camera.start_preview()
    sleep(5)
    self.camera.capture("/home/pi/mywork/basic/image_prac/test" + "10" + ".jpg")
    sleep(2)
    f = open("/home/pi/mywork/basic/image_prac/test" + "10" + ".jpg", 'rb')
    a = f.read()
    fimage = bytearray(a)
    f.close
    if mode == "periodic":  # 주기적으로 보낼 때 동작하는 코드
        publisher.single("iot/awsperiodic", fimage, hostname="35.182.237.235")
    elif mode == "levelDisease":  # 레벨이나 질병 페이지에서 신호를 보낼 때 동작하는 코드
        publisher.single("iot/awslevelDisease/" + plantname, fimage, hostname="35.182.237.235")
    sleep(2)
    self.camera.stop_preview()
```

↑촬영 후 AWS로 PUB (병충해, 질병과 주기적으로 보내는 코드 분리)

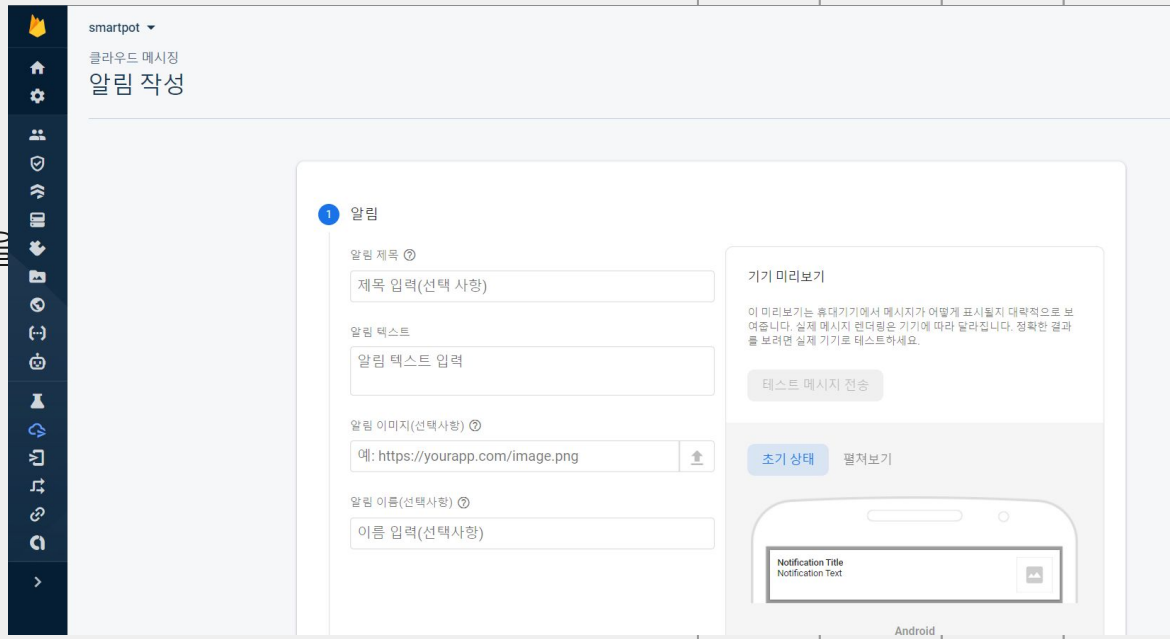
```python
def run(self):
    try:
        # self.opCamera("Aws")
        while True:
            self.currenttime = datetime.datetime.today().hour
            if self.currenttime == 12:  # 12시에 주기적으로 한번씩 보냄
                self.opCamera("periodic")
                print("awspubed")
            time.sleep(60)

    except KeyboardInterrupt:
        pass
    finally:
        self.camera.close()
```

↑ 주기적으로AWS에 카메라를 전송하는 코드

IOT_FIREBASE

FIREBASE

- Firebase의 클라우드 메시징 기능을 통해 백그라운드에서 푸시 알림
- 매일 1번씩 특정 시간에 앱으로 화분의 상태 체크할 수 있게 유도



↑푸시 알림을 보내는 화면

IOT_END

여러가지 시도

- Flask rest api 를 구현해서 AI 와 연결하려고 시도를 해 보았으나 구현하지 못함
- 앱에 커뮤니티 기능을 추가하려다가 리스트까지는 만들었으나 완벽하게 구현하지 못함
- AWS의 RDS와 기기와의 연결 방법에 대해 여러가지로 고민하다 시도해 보았음

AI

- 데이터 수집 및 전처리
- CUSTOM DATASET 구축
- Data Balancing
- Multi Output Model
- Yolo v5
- 예측

데이터 수집 및 전처리

딸기, 상추

AI 허브의 '시설 작물 질병 진단 이미지 소개' 데이터셋 활용 (Image, Annotation)

생육 단계별 기존 3가지 코드를 5단계로 세분화
(생장기 → 출뢰기 → 개화기 → 과실기 → 수확기)

AI Hub

AI 허브란?
AI 기술 · 제품 · 서비스 개발에 필요한 인프라
(데이터, API, 컴퓨팅 자원)를 지원하는 통합 플랫폼

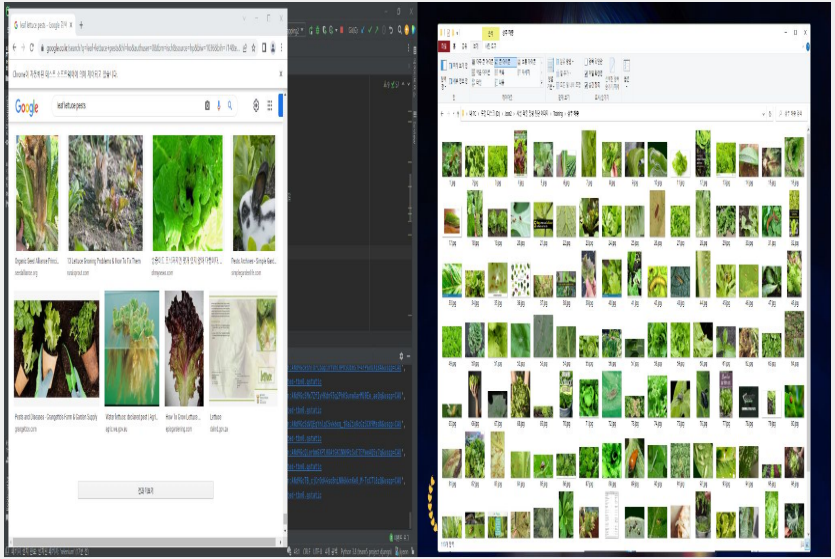
로즈마리, 제라늄

정상 / 질병 / 해충 이미지를 웹 스크래핑 후 라벨링 툴(Roboflow)로 라벨링 및 이미지 증식 작업

roboflow

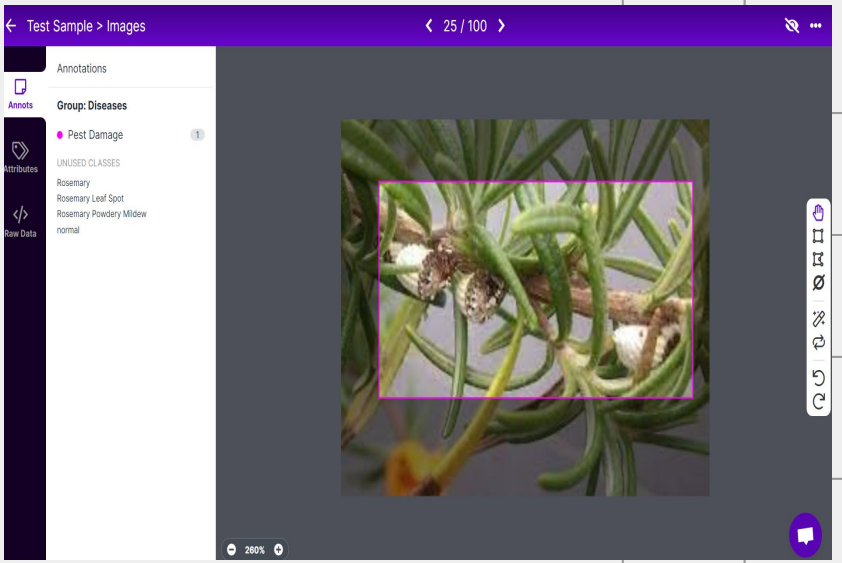
CUSTOM DATASET 구축

웹 스크래핑



urllib, bs4, selenium

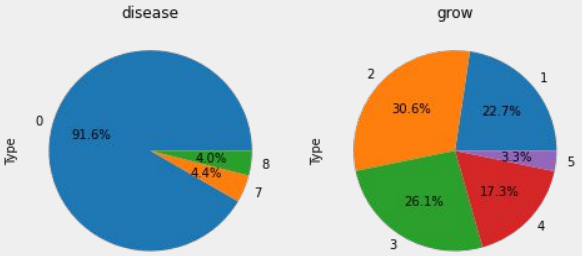
라벨링 툴



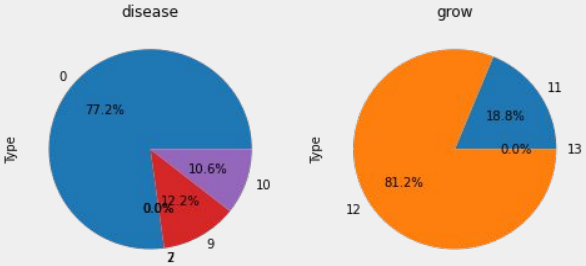
Roboflow

DATA BALANCING

strawberry



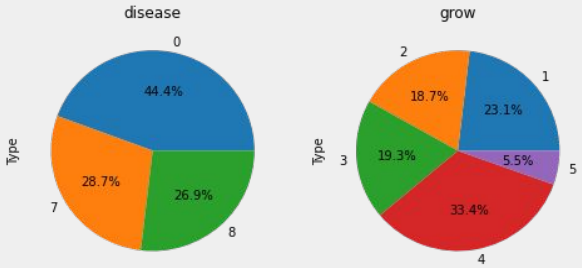
lettuce



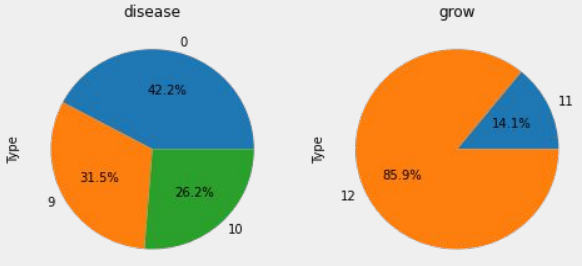
BEFORE

Augmentation,
UnderSampling,
Stratify...

strawberry



lettuce



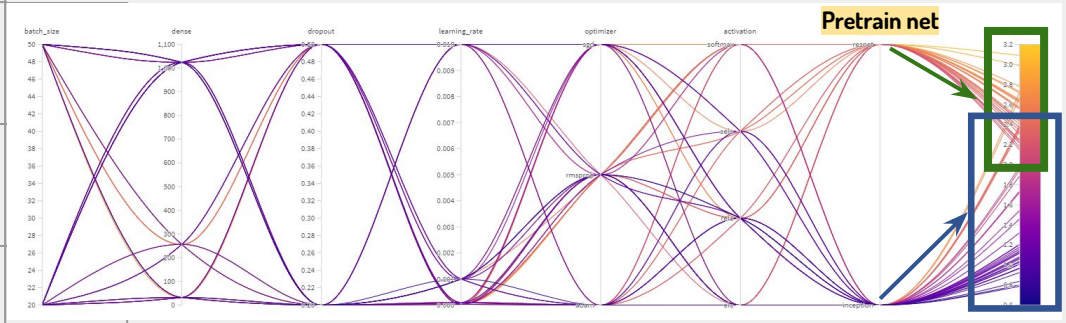
AFTER

※ 두 개이상의 특성(질병, 생육단계)을 고려하여 클래스별 데이터의 개수를 맞추어 모든 특성에 대하여 최적의 balancing이 어려웠음

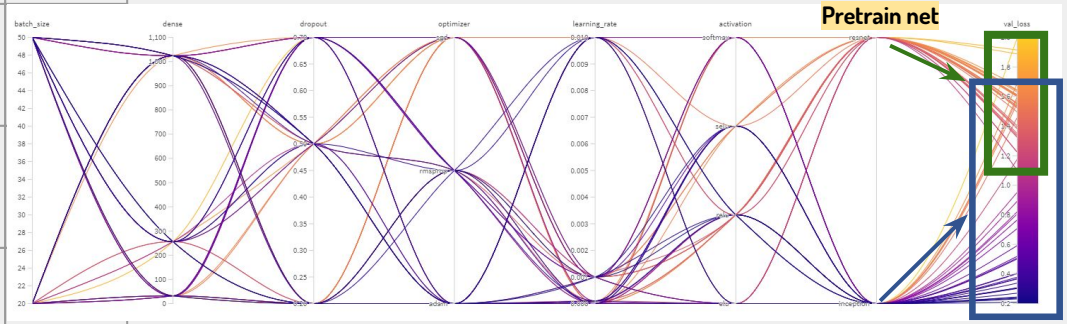
MULTI OUTPUT MODEL

HYPER PARAMETER TUNING (WANDB SWEEP)

trawberry



lettuce



튜닝의 사용한 하이퍼 파라미터의 후보*는
두 모델의 경우 같으며 성능이 가장 좋았던 하이퍼파라미터는

딸기의 경우

Batch size	Drop out	Dense	Activation	Optimizer	Learning rate
20	0.2	32	Elu	adam	1e-3

상추의 경우

Batch size	Drop out	Dense	Activation	Optimizer	Learning rate
50	0.2	32	relu	rmsprop	1e-3

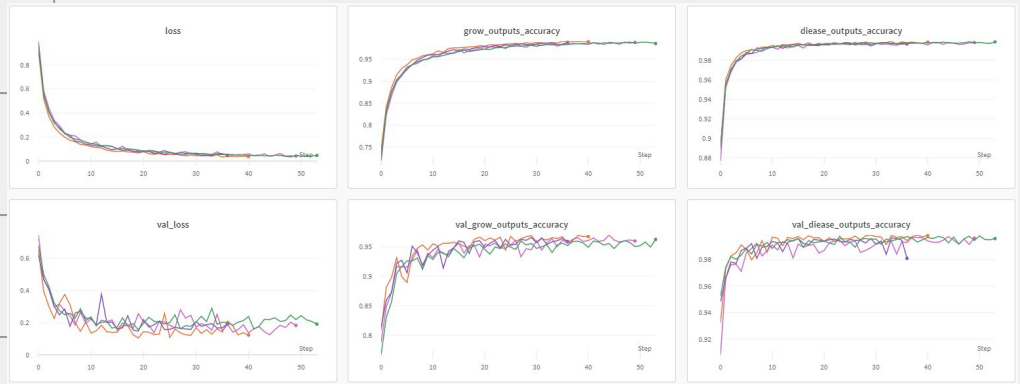
로 나타남

*18 page, wandb 수행 결과 -> 하이퍼파라미터 후보

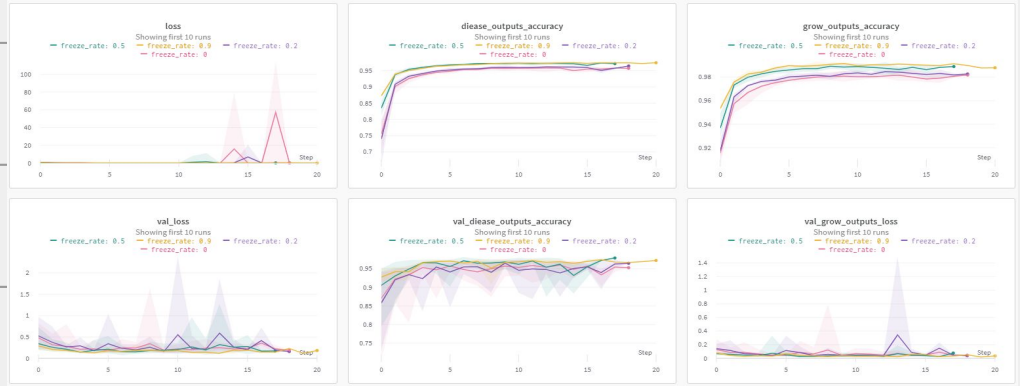
Resnet50과 Inception resnet v2 중
Inception resnet v2를 사용한 모델의 성능이 대체로 우수

MULTI OUTPUT MODEL

strawberry



lettuce



모델 학습

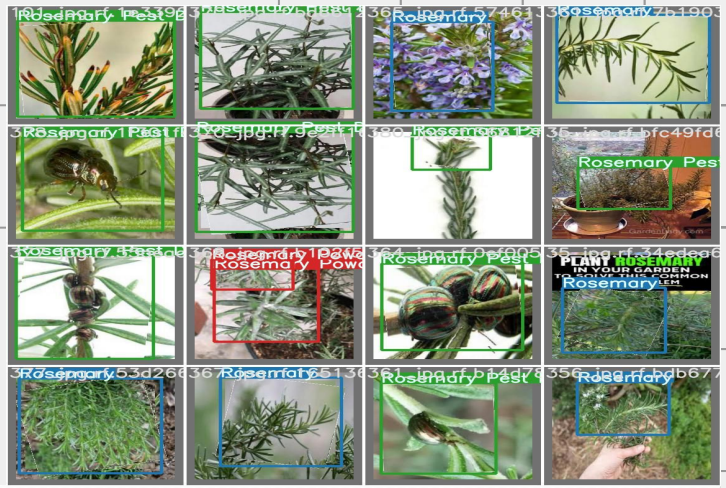
epoch은 매우 크게 설정하고 Early Stopping(patience=5)을 callbacks으로 지정하여 더 이상 학습이 진행되지 않는 지점에서 학습이 멈추도록 함.

Fine tuning

사전학습모델의 동결비율을 0, 0.2, 0.5, 0.9 네 가지 경우로 나누어 탐색,
탐색 후 가장 성능이 좋은 동결비율을 결정.
딸기와 상추 모두 최상의 동결비율은 0.5였으며 미세조정 후 성능이 급격히 상승

상추의 경우 optimizer로 rmsprop을 사용하여 Loss가 크게 진동하는 모습을 확인하였고,
동결비율이 작아질수록 크게 진동하였음

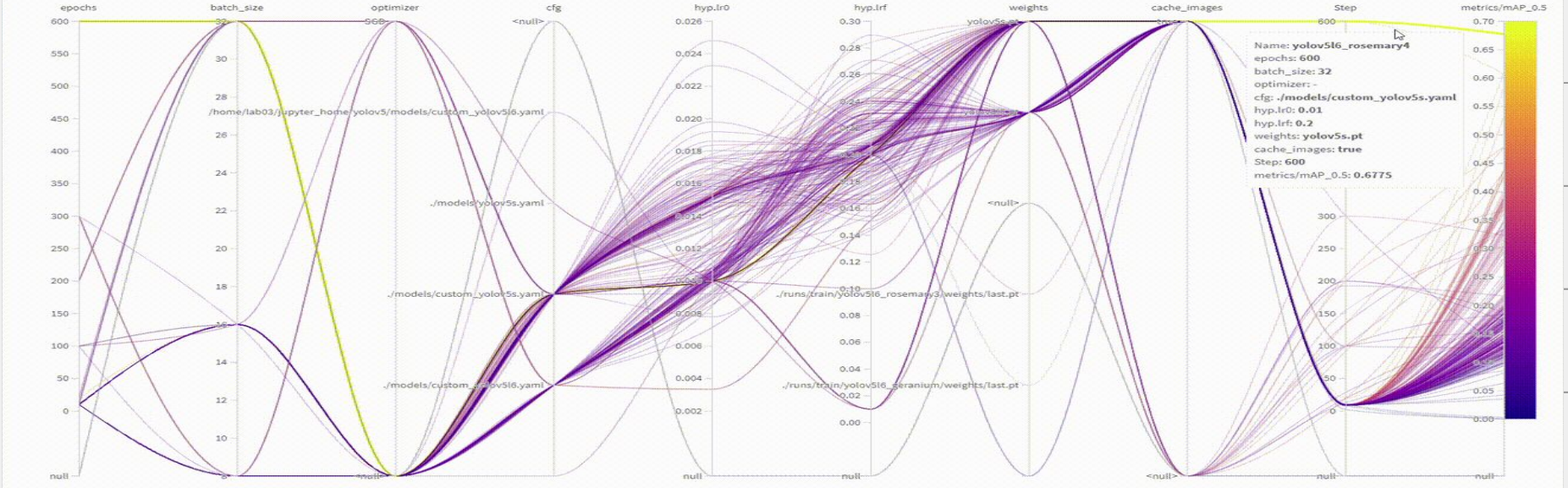
YOLO V5



OBJECT DETECTION

YOLO v5 → 로즈마리/제라늄 병충해 학습

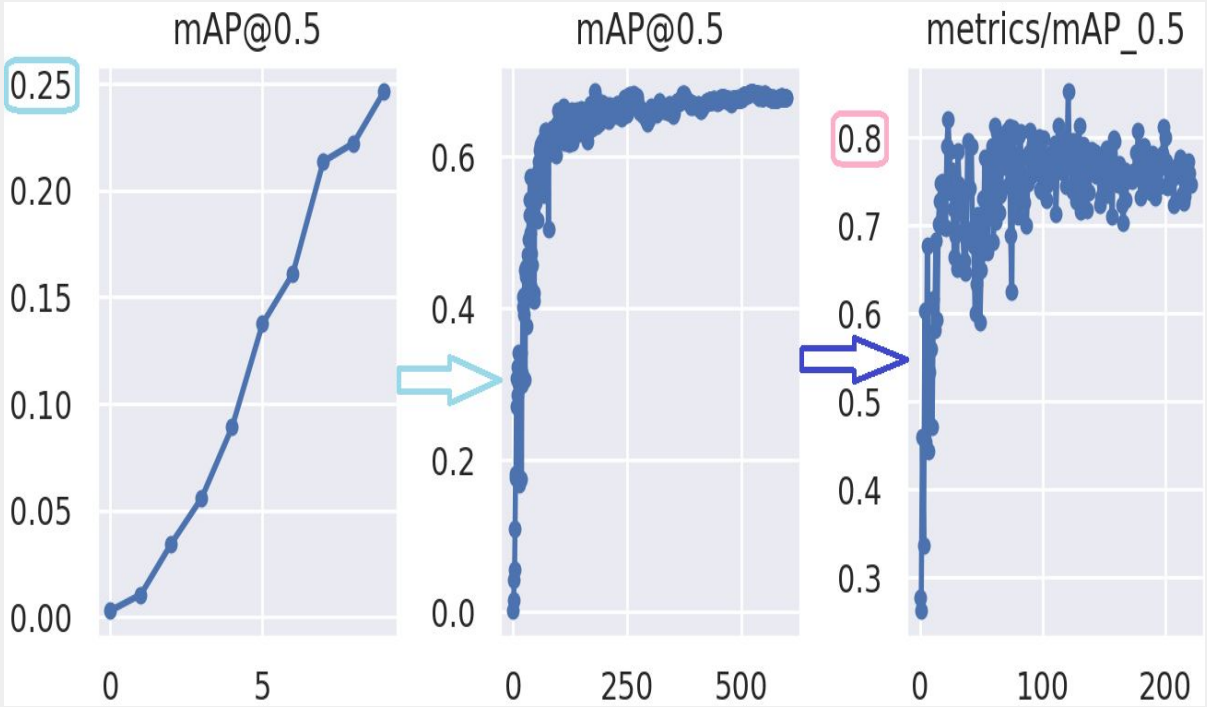
HYPER PARAMETER TUNING (YOLO V5 EVOLVE + WANDB)



YOLO V5

mAP_0.5란?
객체탐지 모델에
사용되는 대표적인
성능지표. 0-1 범위이며
높을수록 성능이 좋음

YOLO v5의 mAP 개선 과정 : 로즈마리 0.8526 | 제라늄 0.6853



- Custom Dataset의 불균형한 클래스 비중을 비슷하게 맞춤
- 학습 데이터의 1%는 탐지할 물체가 없는 Background Image
- Backbone 버전(s, m, l, x) 중 layer가 깊고 정확도가 높은 l6 사용
- Pre-Trained Weights / Hyper-Parameters 사용
- Hyper-Parameters 조정 : input size, epochs, batch size, optimizer, cfg, learning rate, weights, cache image
- YOLO v5의 Evolution 기법으로 하이퍼 파라미터 최적화

예측

질병 분류

딸기	상추	로즈마리	제라늄
잿빛곰팡이병	균핵병	점무늬병	갈색무늬병
흰가루병	노균병	흰가루병	잿빛곰팡이병
해충	해충	해충	해충

딸기 생육 단계

1	생장기
2	출뢰기 (꽃봉오리)
3	개화기
4	과실기
5	수확기

상추 생육 단계

1	육묘기
2	생장기

감사합니다