



# 종합설계 프로젝트 수행 보고서

프로젝트명	스마트 화분 시스템 개발
팀번호	S4-7
문서제목	수행계획서( ) 2차발표 중간보고서( ) 3차발표 중간보고서( ) 최종결과보고서( O )

2020.06.27

팀원 : 문태웅 (팀장)  
이종원  
손지용

지도교수 : 노영주 

지도교수 : 나보균 

## 문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	
2019.12.20	문태웅(팀장)	1.0	수행계획서	최초작성
2020.01.20	문태웅(팀장)	2.0	1차수행 보고서	상세 설계추가
2020.01.28	문태웅(팀장)	2.1	피드백 반영	내용 추가
2020.02.06	손지용	2.2	피드백 반영	내용 추가
2020.03.02	문태웅	2.3	2차 발표ppt	내용 추가
2020.05.02	문태웅	3.0	3차 발표	내용추가
2020.06.25	문태웅	4.0	4차 발표	내용추가
2020.06.27	문태웅	4.1	피드백 반영	내용추가
2020.12.03	문태웅	5.0	최종발표 양식	내용추가

## 문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (2월)	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	I. 서론 (1~6)	I. 서론 (1~6)	I. 서론 (1~6)	I. 서론 (1~6)	I II III
	II. 본론 (1~3)	II. 본론 (1~4)	II. 본론 (1~5)	II. 본론 (1~7)	
	참고자료	참고자료	참고자료	참고자료	

이 문서는 한국산업기술대학교 컴퓨터공학부의  
 “종합설계” 교과목에서 프로젝트 “스마트 화분 시스템”을  
 수행하는 S4-7 문태웅, 이종원, 손지용이 작성한 것으로  
 사용하기 위해서는 팀원들의 허락이 필요합니다.

# 목 차

## I. 서론

1. 작품선정 배경 및 필요성 .....	4
2. 기존 연구/기술동향 분석 .....	4
3. 개발 목표 .....	5
4. 팀 역할 분담 .....	6
5. 개발 일정 .....	6
6. 개발 환경 .....	6

## II. 본론

1. 개발 내용 .....	7
2. 문제 및 해결방안 .....	7
3. 시험시나리오 .....	8
4. 상세 설계 .....	9
5. Prototype 구현 .....	13
6. 시험/ 테스트 결과 .....	14
7. Coding & DEMO .....	16

## III. 결론

1. 연구 결과 .....	22
2. 작품제작 소요재료 목록 .....	23

참고자료 .....	25
------------	----

# I . 서론

## 1. 작품선정 배경 및 필요성

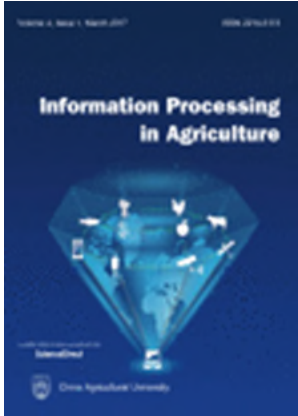
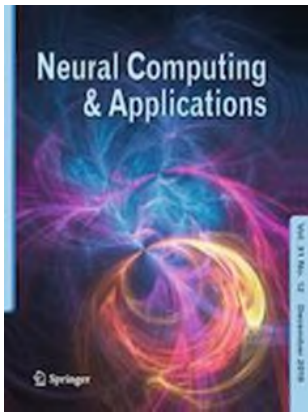
현대인들은 바쁜 시간을 보내기 때문에 식사와 수면을 통해 신체적으로 회복을 하  
나, 심적으로 회복을 하기는 어렵다. 그 때문에 많은 사람이 애완&반려의 목적으  
로 동물 또는 식물을 기르는데, 주로 시간과 정보가 부족해 곤란함을 겪는다. 또한  
바쁜 현대인들은 식물을 키운 경험이 없어 식물의 상태가 잘 파악할 수 없으며 식  
물을 키우기는 어렵다. 식물 초보자들이 급수와 조도를 조절지식이 없어 식물에  
부정적 영향이 있으며 식물 건강 상태를 즉각적으로 발견하지 못하여 식물이  
자주 죽기도 한다.

따라서 본 프로젝트에서 개발할 장치를 통해 자동으로 식물을 관리함과 동시에,  
상태 정보를 사용자에게 제공하여 편하게 대처할 수 있게 한다.

## 2. 기존 연구/기술 동향 분석

	
<p><b>스마트 화분 루아(loa)</b></p> <p>개발사: 둔 뮤디자인 출시: 2019 1월 가격: 약 20만원 주요기능: 5가지 센서를 통한 식물의 15가지 상태 파악</p>	<p><b>샤오미 로팟</b></p> <p>개발사: 샤오미 출시: 2019년 8월 가격: 약 5만원 주요기능: 센서를 통한 식물 상태 인식 및 애플리케이션을 통한 솔루션 제공</p>

2019년 이후로 여러 전자 관련 회사에서 스마트 화분, 식물 케어와 관련한 각종 기술을 개발하고 있다.

	
<p><b>Information Processing in Agriculture</b></p> <p>Detection of plant leaf diseases using image segmentation and soft computing techniques (이미지 분할 및 소프트 컴퓨팅 기술을 이용한 식물 잎 질병의 탐지)</p>	<p><b>Neural Computing and Applications</b></p> <p>Plant disease leaf image segmentation based on superpixel clustering and EM algorithm (식물 잎 질병의 이미지 분할을 기반으로 한 슈퍼 픽셀 클러스터링 및 EM알고리즘)</p>

식물 잎 질병 탐지 관련 알고리즘 ML 논문을 인용하여 ML 알고리즘 구축 예정  
해당 논문들은 식물 잎을 통하여 질병을 판별하는 ML 알고리즘에 관한 내용을 다루고 있어 앞으로 식물 상태를 파악할 수 있다.

### 3. 개발 목표

상기한 것과 같이 해당 프로젝트는 시간이 부족한 사람들의 시간을 대신하는 것을 목표로 한다.

따라서 본 프로젝트에서는 식물을 키우는 데에 필요한 사람의 수고를 덜기 위하여 식물 토양의 습도, 식물이 있는 공간의 조도를 센서로 파악하여 이를 모바일 환경에서 GUI로 정보를 제공하고 카메라 센서를 통해 식물의 상태를 파악하여 접합한 토양의 습도 및 조도를 찾고 자동으로 유지할 수 있도록 설계하였다.

#### 4. 역할 분담

[인베디드 시스템] / 손지용

자동으로 화분을 제어하기 위해 라즈베리파이와 아두이노 장치 사용

센서를 이용한 식물 상태 정보를 측정하여 식물 별로 온도, 습도, 광합성을 조절한다.

카메라 모듈을 이용한 타이밍을 설정하여 정기적으로 식물의 상태를 촬영하며 분석을 한다.

[웹 페이지&네트워크] / 이종원

JAVA와 JSP를 사용해서 웹 페이지로 GUI 제작

[머신러닝] / 문태웅

머신러닝과 네트워크를 연동하여 식물의 상태를 파악하며 자동으로 솔루션

카메라 센서를 이용한 식물 상태 촬영하여 업로드

기타 프로젝트와 관련한 키트, 문서 작업은 공동으로 작업

#### 5. 개발 일정

	1월	2월	3월	4월	5월	6월	7월	8월	9월
프로그램 계획 작성									
내용설계									
구조설계									
기능별 구현									
기능별 테스트									
종합 테스트									
피드백									

#### 6. 개발 환경

환경	상세
개발언어	JavaScript, CSS, Python
주요 라이브러리	Mysql-connector-java, TensorFlow2.0
서버 프로세서	Tomcat server v9.0 / DB : mysql

## II. 본론

### 1. 개발 내용

임베디드 프로그래밍

아두이노 (Arduino)	아두이노 보드 Arduino UNO R3와 Arduino IDE를 사용하여 식물 관리에 필요한 수분 센서, 조도 센서의 정보를 수집 및 통제하는 프로그램과 하드웨어 제작
라즈베리 파이 (Raspberry Pi)	아두이노와 유선으로 연결하여 아두이노 습도 및 조도 센서의 정보를 받아오고 라즈베리 파이의 카메라 센서를 사용하여 식물의 상태를 파악하는 프로그램 및 통신 프로그램 동작 및 구현 Picamera API를 사용하여 카메라센서 통제

웹 페이지 & 네트워크

자바스크립트 (Java Script)	자바스크립트(JSP)와 DM Mysql을 사용하여 모바일 환경 및 PC 환경에서 각각 대응하는 화분의 상태를 확인할 수 있는 동적 웹페이지 구축
네트워크 & 데이터베이스 (Network & DataBase)	센서를 통해 파악된 식물의 정보를 저장하고 관리할 데이터베이스 구축 및 네트워크를 통한 정보 조회기능 구현

웹 스크래퍼 & 머신러닝

웹 스크래퍼 (Web Scraper)	Python BeautifulSoup라이브러리 등을 사용한 웹 스크래퍼 개발 및 스크래퍼를 통한 학습자료 확보
머신러닝 (Machine Learning)	Python OpenCV 라이브러리를 이용하여 카메라센서의 이미지 파일에서 식물의 잎을 인식하고 잎의 상태를 파악하여 Python TensorFlow 라이브러리를 통한 식물의 상태학습 학습한 자료를 통해 실제 식물의 상태를 파악

### 2. 문제 및 해결방안

첫 프로젝트 기획당시 참고문헌인 Information Processing in Agriculture을 인용하여 ANN(인공 신경망) 기반의 강화학습을 사용하여 약 500개의 식물잎 데이터를 통하여 91% 정도의 인식률의 식물 질병 상태를 파악하는 알고리즘을 구축하려고 하였으나

프로젝트의 규모를 조절하지 못하여 1차 발표 이후 머신러닝 부분에서 모든 식물의 질병 상태를 파악하는 것은 시간/기술상 불가능할 것이라는 피드백을 받아서 한 개의 식물로 규모를 줄이고 식물의 영양 각 상태 부족들만 분류하고 바이러스 및 세균에 의한 질병은 너무 다양한 종류가 동반되어 질병 상태 한가지로만 분류하여 각 상황에 따른 잎 상태를 학습하고 분류하기로 변경하였다.

이렇게 분류의 범위를 줄일경우 ANN을 통한 강화 학습을 통하여 학습하는 방법을 채택할 수 도 있지만 강화학습이 숙련도 및 시간상 제약으로 제한될 경우 인식률을 확인한 뒤 CNN

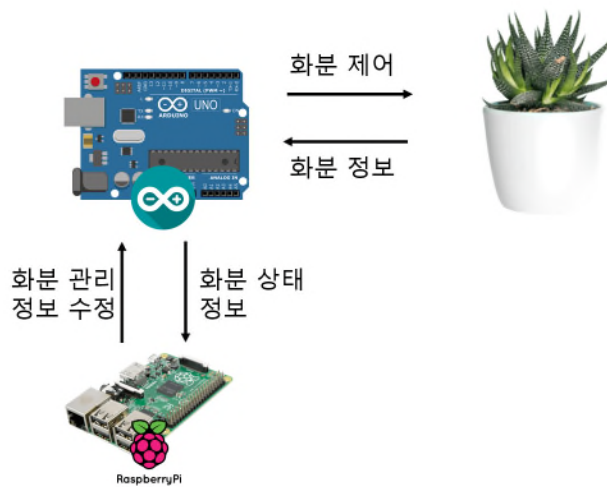
등의 딥러닝 알고리즘을 을 사용하기로 결정

또한, 위의 정보를 통하여 식물의 부족한 영양 정보를 피드백 하여 수분 및 조도를 변경 자동으로 해주고 질병 상태만 파악하는 시스템으로 변경하였다.

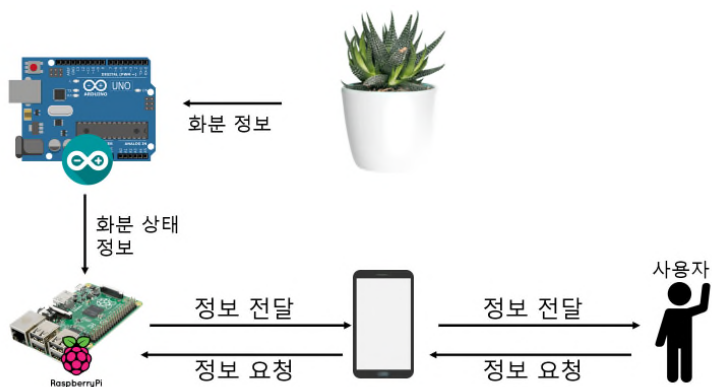
아래의 표는 프로젝트 진행 중 그 외의 부분에서 발생한 프로젝트의 문제 되는 내용 들을 기술하였다.

### 3. 시나리오

#### 1) 스마트 화분 내부(화분 제어 및 피드백)



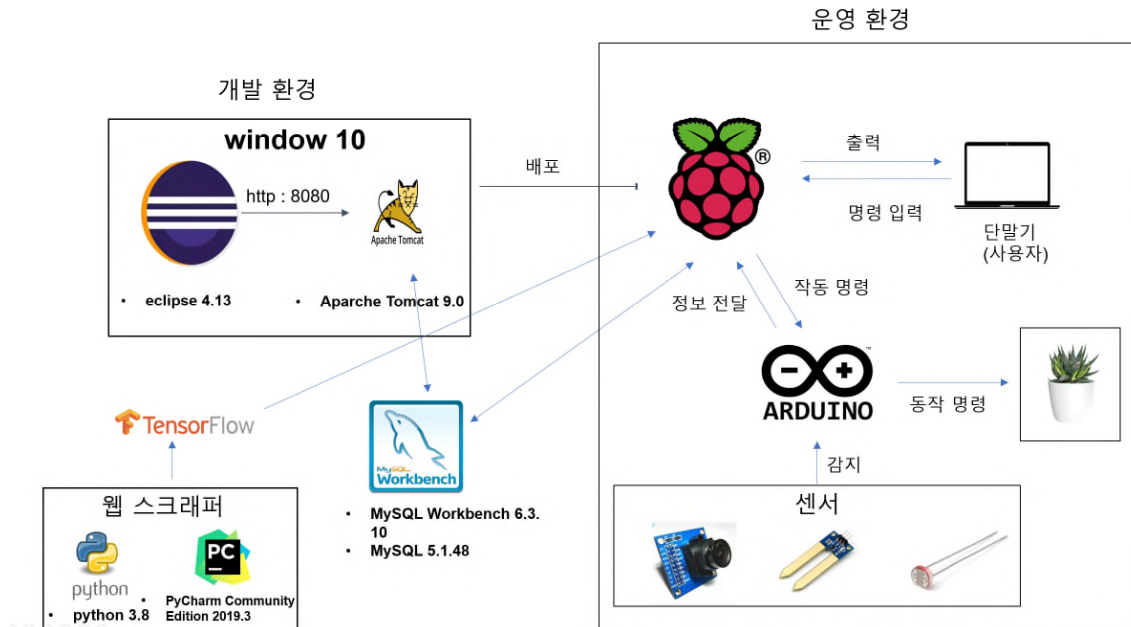
#### 2) 화분 상태 조회



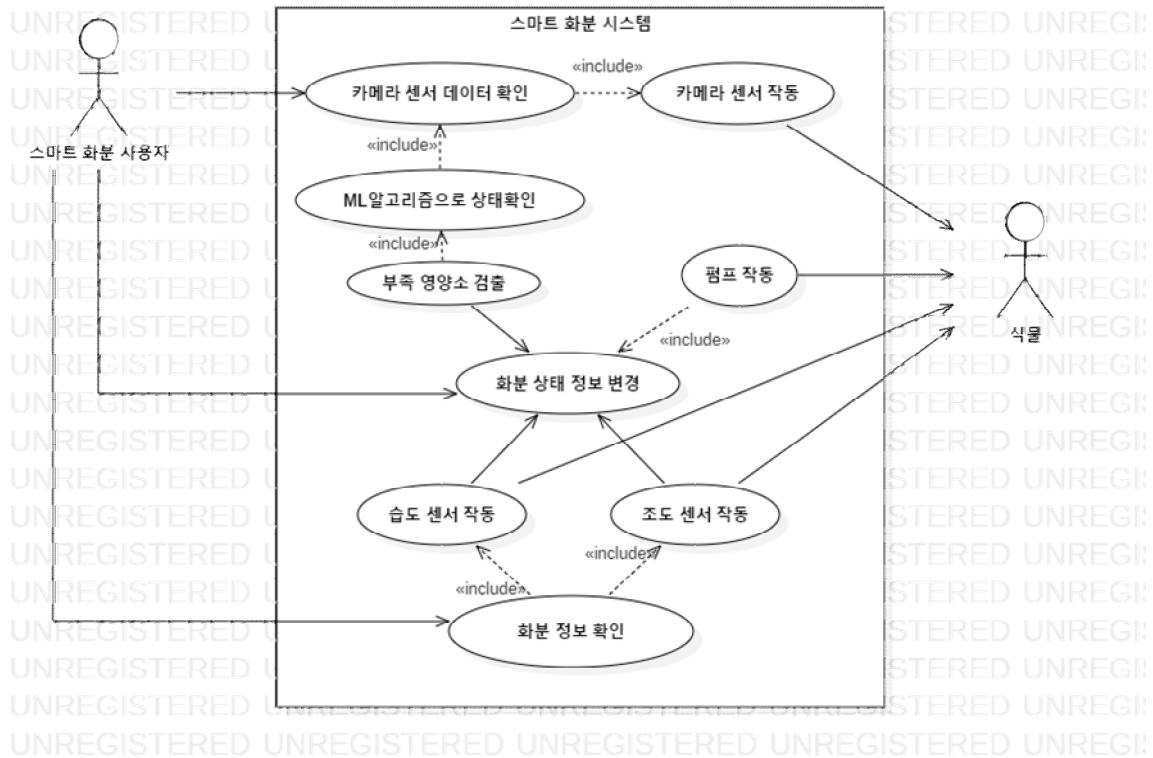


#### 4. 상세 설계

[개발 및 운영 환경 개요]



[유스케이스 다이어그램]

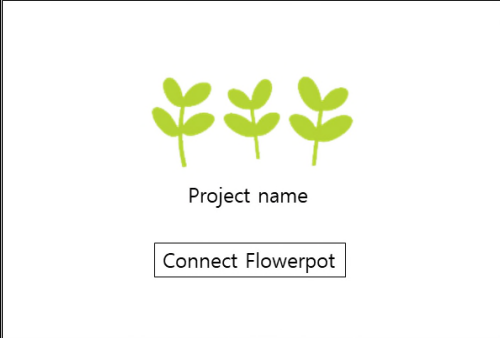
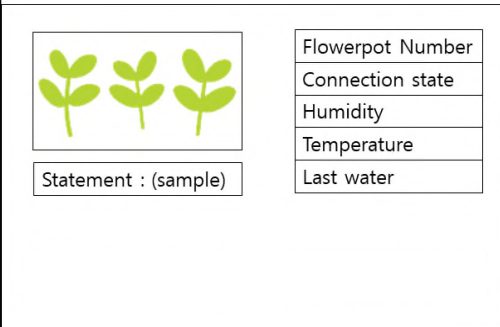


### 1-1) 운영 환경(하드웨어)

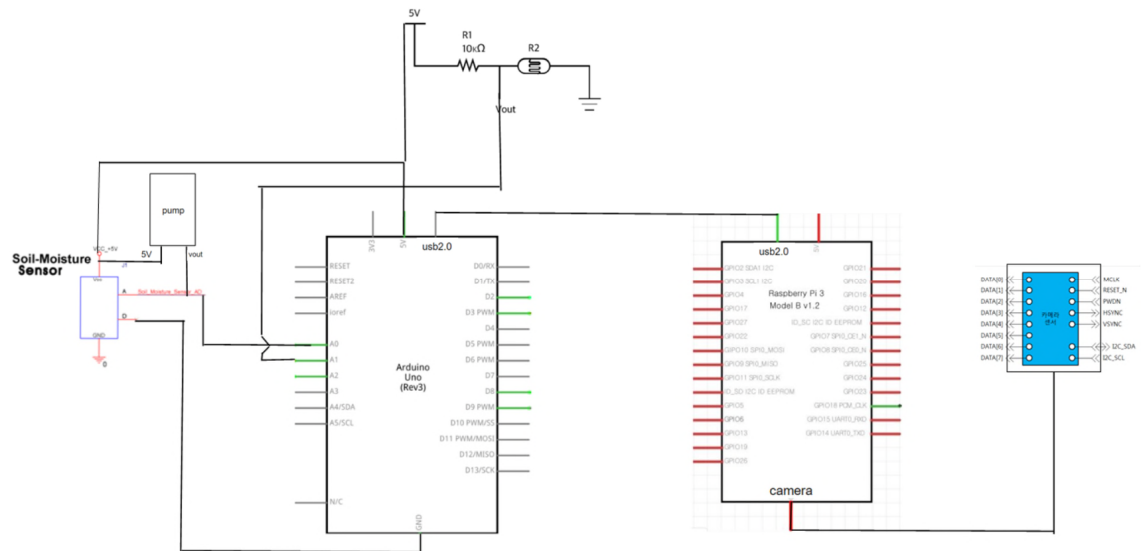
개요	디바이스	기능
조도 센서	CDS(GL10537)	조도 측정
카메라 모듈	OV2640	식물 상태 촬영
토양 습도 센서	SODIAL(R)	식물 수분 필요상태 측정
수위 센서	P5508	물 공급량 측정
급수 펌프	-	배수
아두이노 기판	Arduino UNO R3	센서 제어 및 데이터 수집
라즈베리 파이 기판	Raspberry Pi4	데이터 처리 및 서버 기능 구현
브레드보드, 케이블 등	-	각 디바이스 서로 연결 함

### 1-2) 운영 환경(소프트웨어)

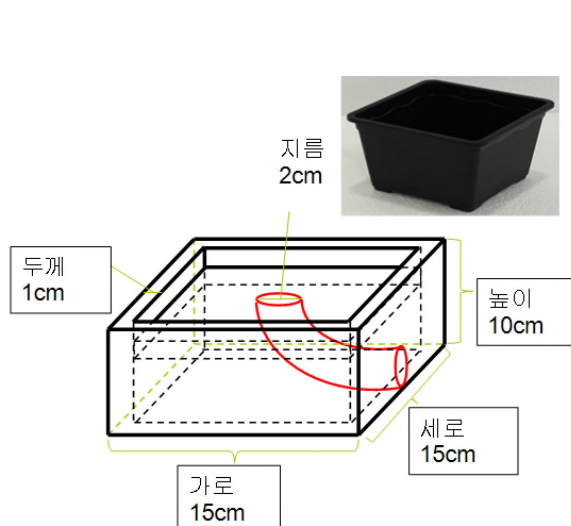
#### [웹 화면 구성]

화면	설명
	<p>메인 페이지</p> <ol style="list-style-type: none"> <li>1. 프로젝트 (홈페이지) 이름 표시</li> <li>2. 프로젝트 로고 이미지 표시</li> <li>3. 상태 페이지로 이동하는 기능</li> </ol>
	<p>화분 상태 페이지</p> <ol style="list-style-type: none"> <li>1. 가장 최근 촬영된 식물의 사진 표시</li> <li>2. 식물의 상태를 건조, 정상, 포화 3단계로 표시</li> <li>3. 화분의 습도량 표시</li> <li>4. 현재 조도량 표시</li> <li>5. 마지막 급수 시각 표시</li> <li>6. 다음 급수 예정 시각 표시</li> </ol>

[회로도]



[키트 예상안]



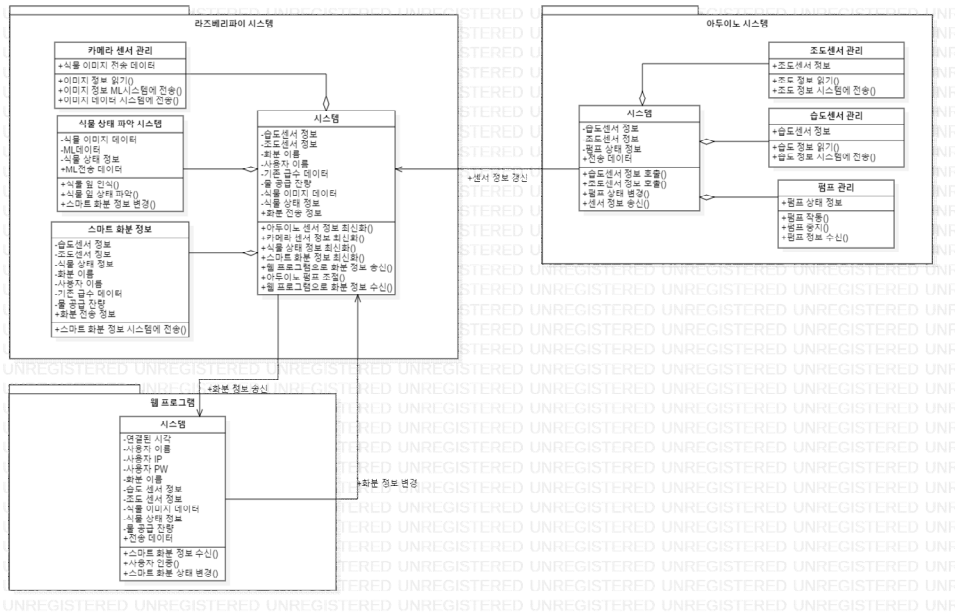
## ■ 센서 장착

높이 100cm  
(조절가능)

- 배수는 호스를 통해 자동 배출
- 상자 내부 시스템 설치
- 급수 탱크는 일반적 pt병 사용

## 2-1) 개발 환경(소프트웨어)

### [클래스 다이어그램 초안]



### [Web Data table]

정보	자료형	비고
사용자 정보[이름]	DATETIME	char형 배열로 10자 이하로 생성
사용자 정보[PW]	VARCHAR	VARCHAR형 배열로 15자 이하로 생성
화분 정보[이름]	CHAR	char형 배열로 10자 이하로 생성
센서 정보[습도]	FLOAT	
센서 정보[조도]	INT	
센서 정보[영상]	INT	
식물의 상태 정보[ML정보]	INT4	0정상상태,-1오류,그외 자연수로 상태 반환
물 공급 잔량[횟수]	INT	
기온 습도 데이터	INT	
ML데이터	-	

### [Arduino/Raspberry Pi Data table]

정보	자료형	비고
사용자 정보[이름]	DATETIME	char형 배열로 10자 이하로 생성
사용자 정보[PW]	VARCHAR	VARCHAR형 배열로 15자 이하로 생성
화분 정보[이름]	CHAR	char형 배열로 10자 이하로 생성
센서 정보[습도]	FLOAT	
센서 정보[조도]	INT	
센서 정보[영상]	INT	
식물의 상태 정보[ML정보]	INT4	0정상상태,-1오류,그외 자연수로 상태 반환
물 공급 잔량[횟수]	INT	
기존 급수 데이터	INT	
ML데이터	-	

## 5. Prototype 구현

### [임베디드 프로그래밍]

이번 데모 환경에서는 아두이노의 각 센서가 작동하는 모습과 각 센서가 정상 작동하는 모습 시연과 각 센서의 민감도를 변경하여 젖은 흙에서는 펌프가 작동하지 않고 수분 필요한 정도를 따라서 물량을 맞추게 공급한다.

라즈베리파이의 경우 아두이노에서 측정된 데이터를 serial를 이용해서 즉시 데이터베이스에서 저장 및 정기적으로 카메라를 작동하여 즉시 데이터베이스로 사진을 전송한다.

### [웹 페이지 & 네트워크]

이번 데모 환경에서는 데이터베이스 구현과 웹페이지에 적용되어 출력되는 모습을 로컬 서버를 통해 시연하고 추후 포트포워딩을 통해 외부에서 접속할 수 있게 할 예정

### [머신러닝]

식물의 잎을 인식할 수 있는 detection algorithm에 학습모델을 적용한다.  
전송받은 사진에서 위의 프로그램을 적용하여 사진에 식물잎 부분을 추출한 후 추출한 사진을 사용하여 정상적인 잎과 비정상적인 잎을 구분할 수 있는 classification 모델 구현  
현재 식물의 잎을 인식하는 것과 잎의 상태를 분류할 수 있는 모델 및 구현 완료  
이 과정을 노트북으로 보여드리고 결과가 웹으로 확인되는 것을 시연

### [프로토타입]



수정	수정사항	내용
초안	-	개인 사정으로 인한 데이터테이블 / 웹스토리보드 구성
1차 수정	웹페이지 구현	웹페이지 디자인 구현 및 이동 구현
2차 수정	웹페이지 구현	웹페이지 일부 기능 구현
3차 수정	데이터베이스 구현	일부 데이터베이스 구현
발표 이전	데이터베이스 완성 웹페이지 완성	데이터베이스 완성 및 웹페이지 완성

## [머신러닝]

수정	수정사항	내용
초안	-	3차 발표에 앞서 나뭇잎 데이터를 병든 잎과 정상상태 잎으로 분류할 수 있는 분류기 제작 DataSet: [Disease-500 / Healthy-500] Model: - Accuracy: 98% Test-case: 41%(41/100) 학습 데이터에서는 정상 작동하나 테스트데이터에서 정상적으로 결과가 나오지 않음
1차 수정	모델 수정	테스트데이터에서 좋은 결과값이 나오지 않아 학습모델이 문제라고 생각하여 학습모델을 가장 기본적인 AlexNet으로 수정후 트레이닝 진행 DataSet: [Disease-500 / Healthy-500] Model: AlexNet Accuracy: 100% Test-case: 38%(38/100) 정확도가 비정상적으로 올라가 불안하였고 예상대로 테스트데이터에서 결과가 좋지 않음
2차 수정	모델 수정	테스트데이터에서 좋은 결과값이 나오지 않아 학습모델이 문제라고 생각하여 학습모델을 가장 기본적인 VGG16모델로 수정후 트레이닝 진행 DataSet: [Disease-500 / Healthy-500] Model: VGG16(VGGNet) Accuracy: 100% Test-case: 39%(39/100) 여전히 테스트데이터의 결과가 좋지 않아 학습 모델의 문제가 아니라고 판단

3차 수정	학습 데이터 변경	<p>학습 데이터의 질적인 문제가 있다고 판단하여 훈련 데이터가 좀 더 명확하게 상태를 반영할 수 있도록 변경</p> <p>DataSet: [Disease-500 / Healthy-500]</p> <p>Model: VGG16(VGGNet)</p> <p>Accuracy: 100%</p> <p>Test-case: 38%(38/100)</p> <p>결과가 좋지 않아 다른 문제 탐색</p>
4차 수정	학습 데이터 추가	<p>학습 데이터의 양이 절대적으로 부족하다고 판단하여 학습 데이터 추가</p> <p>DataSet: [Disease-10000 / Healthy-10000]</p> <p>Model: VGG16(VGGNet)</p> <p>Accuracy: 99%</p> <p>Test-case: 95%(95/100)</p> <p>좋은 결과를 얻어 데이터가 부족했음을 알게됨</p>
5차 수정	올로 학습	<p>카메라 센서에서 들어오는 정보는 앞이 많은 사진이므로 Yolo를 통하여 앞들을 하나하나 인식할 수 있도록 학습하였으나 인식률이 나오지 않음</p>
6차 수정	네트워크 모델 연결	<p>라즈베리 파이에서 판별 프로그램을 돌리지 못할 것으로 판단하여 노트북에서 프로그램을 돌린 뒤 네트워크 프로그램으로 학습결과를 전송하여 결과를 저장하는 방식으로 변경</p>
발표 이전	올로 재학습 모델 수정	<p>Yolo를 통하여 나뭇잎 하나하나 인식할 수 있도록 학습 후 앞의 상태 별로 데이터를 분류하여 다양한 상태로 데이터 분류할 수 있도록 수정 및 인용한 논문의 모델로 수정(진행중)</p>

## 7. Coding & DEMO

### [임베디드 프로그래밍]

아두이노는 수분 센서를 통해서 수분량을 백분수로 출력하며 측정된 값을 따라 4가지 결과를 작동한다. 수분이 25%보다 적으면 펌프가 5초 동안 작동하며 25~50% 사이는 2초만 작동한다.

수분 70%보다 많으면 수분이 과다다는 메시지를 출력한다.



```

if(test >=Low_water && test<=High_water){
    Serial.print("Humidity is normal.\n");
}
else if(test <= Low_water){
    Serial.print("Humidity\n");
}
else{
    if(test >= No_water){
        Serial.print("Humidity is low\n");
        Serial.print("pump open\n");
        digitalWrite(pump,LOW);
        delay(5000);
        digitalWrite(pump,HIGH);
        Serial.print("pump close\n");
    }
    else{
        Serial.print("Humidity is low\n");
        Serial.print("pump open\n");
        digitalWrite(pump,LOW);
        delay(2000);
        digitalWrite(pump,HIGH);
        Serial.print("pump close\n");
    }
}
if(Intensity<=light){
    Serial.print("Intensity is Low\n");
}
    delay(1500);
}

```

라즈베리파이는 아두이노랑 serial를 이용해서 통신한다. 이를 통해서 라즈베리파이는 MYSQL과 아두이노의 serial를 연결하며 아두이노에서 반환하는 데이터를 즉시 라즈베리파이 에 있는 데이터베이스에 저장한다.

```

import pymysql
import serial
ser= serial.Serial('/dev/ttyACM0',9600,timeout=1)
try:
    while 1:
        response = str(ser.readline().decode())
        if response.startswith('ok'):
            print(response.strip('\n'))
            hum=int(response[2:4])
            tem=int(response[7:9])
            smoke=int(response[11:14])
            door=str(response[14:18])
            conn = pymysql.connect(
                host= '127.0.0.1',
                port=3306,
                user= 'root',
                passwd= 'shine2019',
                db= 'SafeHome',
            )
            cur=conn.cursor()
            cur.execute("insert into homedb(hum,tem,smoke,door) values('%d','%d','%d','%d','%s')"%(hum,tem,smoke,door))
            cur.close()
            conn.commit()
            conn.close()
except KeyboardInterrupt:
    ser.close()

```

PiCamera API를 이용하여 정기적으로 식물을 촬영하는 기능을 구현한다.

촬영된 사진은 년, 월, 일, 시, 분의 순서로 사진의 이름을 정해지며 데이터베이스에 저장한다. (저장 기능 아직 구현 못함.)

```
import os
from time import sleep
from datetime import datetime, timedelta
from picamera import PiCamera
from bpython import ByPy

def wait(delay_minute = 1):
    next_time = (datetime.now() + timedelta(minutes=delay_minute)).replace(second=0, microsecond=0)
    delay = (next_time - datetime.now()).seconds
    sleep(delay)

by=ByPy()
camera = PiCamera()
camera.start_preview()
wait()

for filename in camera.capture_continuous('img(timestamp:%Y-%m-%d-%H-%M).jpg'):
    print('capture %s' % filename)
    by.upload(filename)
    os.remove(filename)
    wait()
```

## [웹 페이지 & 네트워크]

### [데이터베이스]

아두이노 센서값들은 settings 테이블에 저장하고 라즈베리파이에서 찍은 사진과 사진을 통해 학습한 결과값을 photo\_ML 테이블에 저장한다. id를 참조 값으로 주어서 테이블 간 연결을 하고 auto\_increment를 주어 업데이트 횟수를 확인할 수 있게 하였다.

```
1 • create database Flowerpot CHARACTER SET utf8 COLLATE utf8_general_ci;
2   use Flowerpot;
3
4 • CREATE TABLE photo_ML(
5   ml_id varchar(10) NOT NULL primary key,
6   ml_state int NOT NULL,
7   photo varchar(255) NOT NULL
8 ) DEFAULT CHARSET=utf8;
9
10 • CREATE TABLE settings (
11   id int Not NULL auto_increment primary key,
12   state varchar(10) NOT NULL,
13   lux varchar(10) NOT NULL,
14   humidity varchar(10) NOT NULL,
15   set_date DATETIME DEFAULT CURRENT_TIMESTAMP
16 ) DEFAULT CHARSET=utf8;
17
18
19 • ALTER TABLE settings ADD FOREIGN KEY (state) REFERENCES photo_ML (ml_id);
```

## [서버]

라즈베리파이에서 간단한 테스트 서버를 구현하였고 기존에 작성하였던 JSP 파일을 옮겨서 웹페이지 서빙 할 계획임. 현재는 포트포워딩에 문제가 있어 로컬로만 접속이 가능함

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/flowerpot/<_name>')
def hello(_name):
    return render_template('page.html', name=_name)

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

실행과 연결됐다는 메시지

```
pi@raspberrypi:~/webapp $ ls
app.py static templates
pi@raspberrypi:~/webapp $ python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 517-354-658
192.168.0.2 - - [25/Jun/2020 13:55:34] "GET / HTTP/1.1" 200 -
```

## [머신러닝]

기존 이미지를 사용할 크기로 조정(Image resizing)

```
for img in files:
    print(str(j) + "/" + str(num), end="\r")
    trainImg.append(cv2.resize(cv2.imread(img), (ScaleTo, ScaleTo)))
    trainLabel.append(img.split('\\')[-2])
    j += 1

trainImg = np.asarray(trainImg)
trainLabel = pd.DataFrame(trainLabel)
```

```
for img in trainImg:
    blurImg = cv2.GaussianBlur(img, (5, 5), 0)

    hsvImg = cv2.cvtColor(blurImg, cv2.COLOR_BGR2HSV)
    # https://ko.wikipedia.org/wiki/HSV
    lower_green = (25, 40, 50)
    upper_green = (75, 255, 255)
    mask = cv2.inRange(hsvImg, lower_green, upper_green)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11, 11)) ##타원 모양의 11 * 11 커널 생성
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel) ##원본 이미지에를 깔끔하게
    # https://hoony-gunputer.tistory.com/entry/opencv-pythonErosion
    # https://swprog.tistory.com/entry/Mathematical-morphology-
    bMask = mask > 0 ## 배열을 불린언 형으로

    clear = np.zeros_like(img, np.uint8)
    clear[bMask] = img[bMask]
```

```
le = preprocessing.LabelEncoder()
# https://medium.com/@john\_analyst/%EB%8D%B0%EC%9D%B4%ED%84%B0-%EC%A0%84%EC
le.fit(trainLabel[0])
print("Classes: " + str(le.classes_))
encodeTrainLabels = le.transform(trainLabel[0])

clearTrainLabel = np_utils.to_categorical(encodeTrainLabels)
# https://nabzacko.tistory.com/7 ex) 1 => [1,0,0] 2=> [0,1,0] 3=>[0,0,1]
num_classes = clearTrainLabel.shape[1] # [1~19] [1~19] ~ [1~19] => 7057H
```

- 20 -

```
[ ] # VGG16 + GAP 모델
    np.random.seed(seed)

    input_tensor=Input(shape=(64,64,3))
    pre_trained_model = applications.VGG16(weights='imagenet', include_top=False, input_tensor=input_tensor)
    pre_trained_model.trainable=True
    pre_trained_model.summary()
```

VGG16모델을 기반으로 학습 진행

```
▶ learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss',
                                              patience=5,
                                              verbose=1,
                                              factor=0.4,
                                              min_lr=0.0001)

    filepath='/content/drive/My Drive/ML/'
    modelpath=filepath+'(epoch:02d)-(val_loss:.4f).hdf5'
    checkpoint=ModelCheckpoint(filepath=modelpath,monitor='val_loss',verbose=1,save_best_only=True)

    #hist=model.fit(x=trainX,y=trainY,epochs=200,validation_data=(testX,testY),batch_size=500,callbacks=callbacks_list)
    hist = model.fit_generator(datagen.flow(trainX, trainY, batch_size=20),
                              epochs=20, validation_data=(testX, testY),
                              steps_per_epoch=trainX.shape[0], callbacks=[checkpoint])
```

생성된 모델로 테스트데이터 확인

```
[ ] model.load_weights("/content/drive/My Drive/ML/19-0.0123.hdf5")

▶ path = '/content/drive/My Drive/ML/test_set/*/*'
  files = glob(path)

  clearTestImg = []
  testLabel = []
  j = 1
  num = len(files)

  for img in files:
      clearTestImg.append(cv2.resize(cv2.imread(img)[...,:-1],(ScaleTo,ScaleTo)))
      testLabel.append(img.split("/")[-2])

  testLabel = pd.DataFrame(testLabel)
  clearTestImg = np.asarray(clearTestImg)
```

### Ⅲ. 결론

#### 1. 연구 결과

##### 1-1) 구현 결과 및 문제점

분류기에 다양한 모델들을 프로그램에 적용해 보았으나, 3차 수정까지는 1,000여 장의 사진 데이터를 사용하여 학습하여 학습 데이터에서는 높은 일치율을 보였지만 테스트데이터에서는 낮은 일치율을 보여주었다.








낮은 일치율을 높이기 위하여 모델을 바꿔서 적용하였지만, 일치율이 오르지 않아 학습하는 자료가 부족하다고 판단하여 학습 데이터를 20,000여 장까지 늘렸더니 테스트데이터에서도 높은 일치율을 보였다. 탐색 알고리즘의 초반에 고전 머신러닝을 사용하였으나 학습 데이터들에서는 높은 일치율을 보였지만 일반적인 상황의 데이터에서도 학습률이 저조하게 나왔다. 이러한 문제를 해결하기 위하여 기존에 사용하였던 고전 학습모델을 포기하고 다른 모델을 사용하기로 하였다.

결과적으로 4차 수정에서는 Yolo v3를 적용하였고 간단한 테스트데이터에서는 높은 일치율을 보여주었으나, 많은 잎이 있는 실제 환경과 비슷한 데이터들의 경우는 다양한 각도의 잎들이 사진에 존재하기 이 때문에 일치비율이 나오지 않았다. 하지만 분류기와 달리 탐색 모델의 경우 학습 데이터를 늘리기가 쉽지 않기 때문에 탐색 모델에서는 학습 데이터 수를 제한하여 성능을 어느 정도 타협하였다.

##### 1-2) 결론 및 향후 연구과제

서론에서 소개하였듯이 식물을 키우는 다양한 제품들이 출시되고 있다. 그렇지만 자동으로 식물을 관리해주는 제품들은 많이 출시되고 있지 않다. 그러므로 현재 시스템에 자동으로 급수해주는 시스템뿐만 아니라 조도나 영양분 또한 자동으로 관리하여 줄 수 있는 시스템을 앞으로 추가하여 개발을 진행할 수도 있을 것이다. 이번 작품을 개발하면서 식물의 잎의 경우 다른 각도에서 다양한 모습으로 변화하여 학습을 진행하기 쉽지 않았다. 그렇기 때문에 식물을 관리할 때 잎이 아닌 다른 부분의 특징을 통하여 식물의 상태를 관리할 수 있는 시스템으로 변경하여 시도해볼 필요가 있다.

## 2. 작품제작 소요재료 목록

이미지	개요	디바이스	용도
	조도 센서	CDS(GL10537)	조도 측정
	카메라 모듈	OV2640	식물 상태 촬영
	토양 습도 센서	SODIAL(R)	식물 수분 필요상태 측정
	급수 펌프	-	배수
	아두이노 기판	Arduino UNO R3	센서 제어 및 데이터 수집
	라즈베리 파이 기판	Raspberry Pi4	데이터 처리 및 서버 기능 구현
	브레드보드	-	회로간 연결

	케이블 등	-	회로간 연결
	아크릴 판	-	케이스 제작



## 참고자료 / GitHub

[1] “ScienceDirect” “Information Processing in Agriculture” Detection of plant leaf diseases using image segmentation and soft computing techniques, 2017년 3월 수정  
<https://www.sciencedirect.com/science/article/pii/S2214317316300154#f0005>

[2] “SpringerLink” “Neural Computing and Applications” Plant disease leaf image segmentation based on superpixel clustering and EM algorithm, 2017년 6월 26일 수정  
<https://link.springer.com/article/10.1007/s00521-017-3067-8>

[3] “15가지 감정을 표현하는 스마트 화분 '루아'” lboon BIZION  
<https://lboon.kakao.com/bizion/5d363e07ec5db05127c910db>

[4] “샤오미 로팻 식물관리 블루투스 스마트화분” funshop  
<https://www.funshop.co.kr/goods/detail/58534>

[5] “나만의 웹 크롤러 만들기” GitBook  
<https://beomi.github.io/gb-crawling/?q=>

[6] “Plant Seedlings with CNN and Image Processing”  
<https://www.kaggle.com/nikkonst/plant-seedlings-with-cnn-and-image-processing/data>

[7] “Plant Leaf Disease Detection using Tensorflow & OpenCV in Python”  
<https://github.com/dwij28/Plant-Leaf-Disease-Detection>

[8] “딥러닝을 이용한 양파 발의 잡초 검출 연구”  
[http://kism.or.kr/file/memoir/7\\_3\\_2.pdf](http://kism.or.kr/file/memoir/7_3_2.pdf)

[-] GitHub  
[https://github.com/acidracon/2020\\_KPU\\_SmartFlowerpot](https://github.com/acidracon/2020_KPU_SmartFlowerpot)