
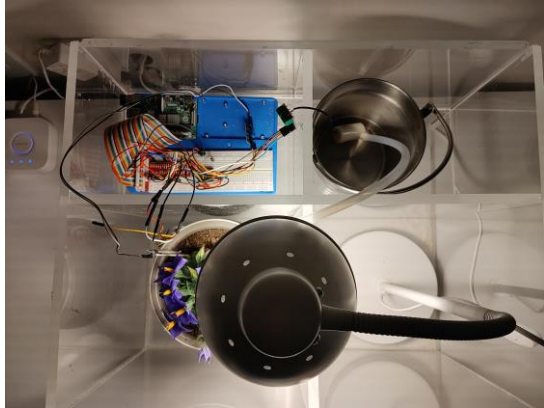
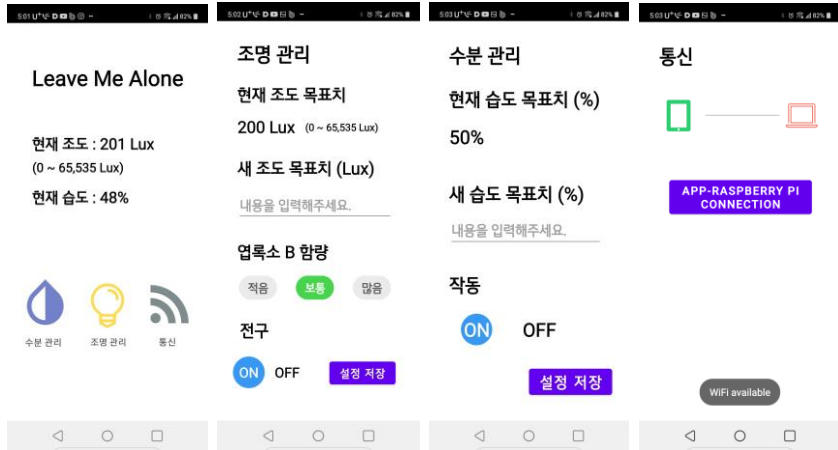


제19회 임베디드SW경진대회 개발완료보고서

[자유공모]

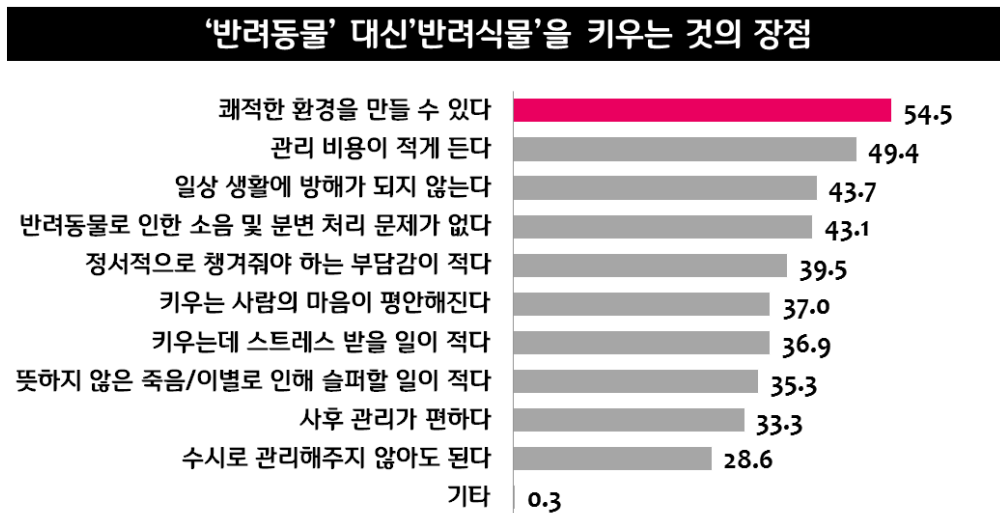
□ 개발 요약

팀 명	코드몽키
<div>   </div> <p><재배기 외형></p> <div>  </div> <p><안드로이드 앱 UI></p>	
작품명	Leave Me Alone
작품설명 (요약)	반려식물을 기르는 용도의 재배기. 사용자가 원하는 식물 상태를 정하면, 이를 유지한다.
소스코드	https://github.com/rkrmru/2021ESWContest_free_1071
시연동영상	https://www.youtube.com/watch?v=gTn71aVhop8

□ 개발 개요

○ 개발 작품 개요

- 2017년 엠브레인 트렌드 모니터에서 성인 1,000여명에게 조사한 바에 따르면, 응답자의 49.4%가 동물보다 식물을 길렀을 때의 장점으로 관리 비용이 적게 드는 걸 선택했다. 또한 43.7%의 응답자가 일상에 방해가 되지 않아 좋았다고 말했다.



(Base; 전체, N=1,000, 단위; 중복 %)

현재 실내 화초를 기르는 사람이 늘고 있다. 그리고 위의 설문처럼 화초를 기를 때, 요구되는 낮은 난이도는 화초를 키울 때 중요한 요소다. 때문에, 이를 위해 자동화된 실내 화분들이 나와있다. 하지만, 대부분의 화분 혹은 재배기는 일정 주기로 물과 조명을 제공하는 수준의 단순한 루틴으로 동작한다. 그래서 우리 팀은 개인용 식물재배기에 보다 세부적인 기능을 추가하고 싶었다.

○ 개발 목표

- 사용자가 원하는 화초 상태를 입력하면, 재배기가 이를 유지하도록 동작한다. 관련 동작은 센서를 통한 측정 값에 따라 달라진다.
- 과습 상태이거나 과한 빛을 받고 있다면, 추가적인 수분 공급이나 조명 제공을 중단한다.
- 식물에 따라 다른 파장대의 빛을 선호하는 걸 이용하여, 사용자가 원하는 파장대의 빛을 선택

해서 식물에게 제공할 수 있게 한다.

○ 개발 작품의 필요성

- 기존 식물 재배기는 물을 공급하는데 있어서, 식물 뿌리가 항상 습윤한 상태를 유지하도록 수경재배를 이용한다. 혹시 흙을 사용하더라도 토양의 하부가 항상 젖어 있도록 하는 저면 관수를 이용한다. 이는 비용을 줄이고 관리를 용이하게 하지만, 선인장과 같은 건조 기후의 식물은 기르지 못하게 한다.
- 기존의 조명 시스템은 식물에 따라 일정한 조명 시간을 매일 제공한다. 이를 위해 타이머를 사용한다. 이는 자연광이 충분한 상황에서 과조명을 유발하거나, 불필요한 조명 작동으로 에너지 소모를 늘린다.
- 또한, 식물은 개체마다 소모하는 빛의 파장이 다르다. 이는 식물이 가진 엽록소의 종류와 그 비율 때문이다.

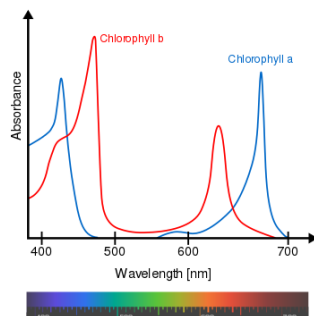


Figure 1. 엽록소 A(청색)과 엽록소 B(적색)의 가시광선 흡수 스펙트럼

식물은 주로 엽록소 A와 B를 사용해 광합성을 한다. 그리고 이 엽록소 A, B는 흡수하는 빛의 스펙트럼이 다르다. 또한, 식물 별로 이 엽록소 A, B의 구성비도 다르다. 그래서 많은 재배기에서 사용하는 고른 파장의 백색등을 쓰면, 에너지를 불필요하게 쓰게 된다.

- 때문에, 수경재배를 고려하지 않은 상태에서 일정 수분을 제공하면 더 다양한 식물을 기를 수 있다. 그리고 색상 조절이 가능한 전구를 이용하여, 사용자가 원하는 빛의 파장을 선택해서 제공하게 하면 보다 효율적인 식물 생육이 가능하다.


□ 개발 환경 설명

○ Hardware 구성

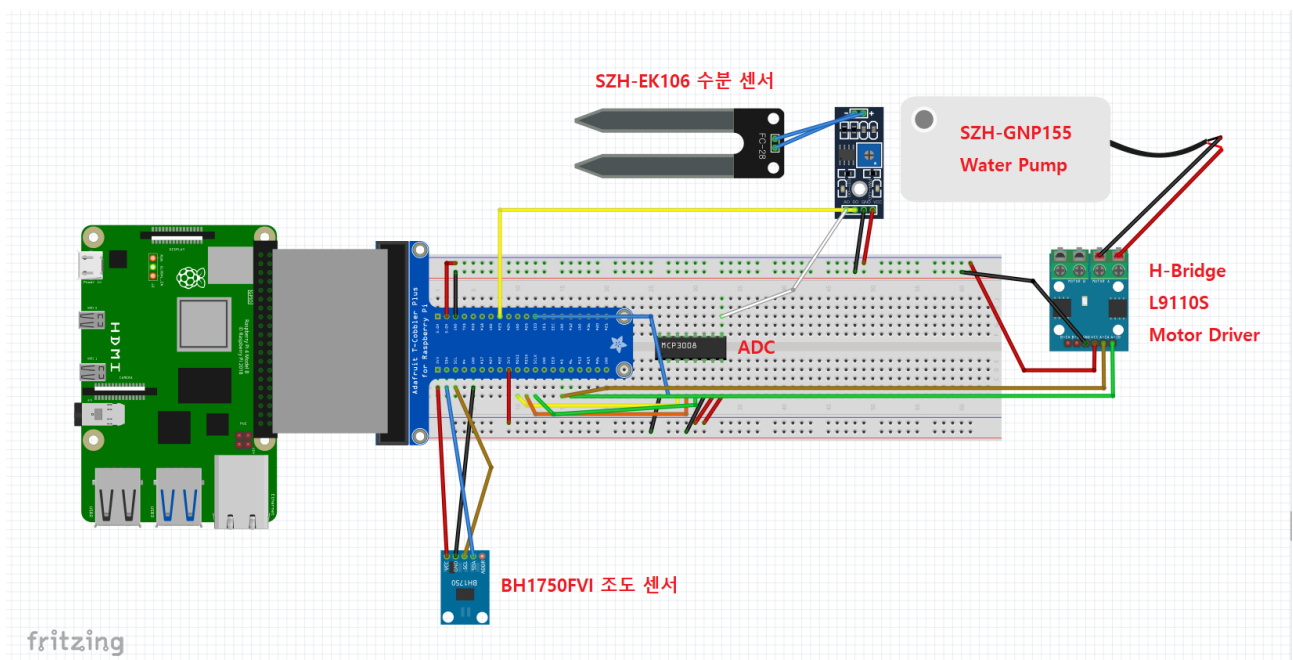
1) 부품 선정

파트	제품명 및 사진	선정 이유
Controller Unit	[Raspberry Pi 4B] 	WiFi 모듈을 내장하기 때문에 네트워크 접근성이 좋다. 또한, Ubuntu 설치가 가능하므로, 해당 OS 기반의 웹 서버 소프트웨어를 이용할 수 있다.
Sensor	[SZH-EK106] 	토양전기전도도(EC)를 측정하여 토양 습도를 유추할 수 있다. 저렴한 가격도 장점이다.
	[BH1750FVI] 	주변의 광도를 측정하기 위해서 선정했다. 1~65,535 Lux까지의 넓은 조도를 측정 가능한 게 장점이다.

ADC	[MCP3008-I/P] 	습도 센서에서 읽어온 아날로그 신호를 디지털 신호로 바꾸기 위해 골랐다. Raspberry Pi 4B에 ADC(Analog-Digital Converter) 기능이 없다.
Pump	[SZH-GNP155] 	수중에서도 취수가 가능한 펌프이다. 별도의 취수용 고무 호수를 마련하지 않아도 되어서 선택했다.
Motor Driver	[H-Bridge L9110S] 	펌프 작동 시 유량을 제어를 위해서 사용했다.
Lamp	[Philips Hue White and Color E26] 	색상 변화가 가능한 소켓 전구이다. 넓은 파장대의 빛을 제공하며, JSON 기반의 편리한 API를 이용할 수 있어서 선택했다.

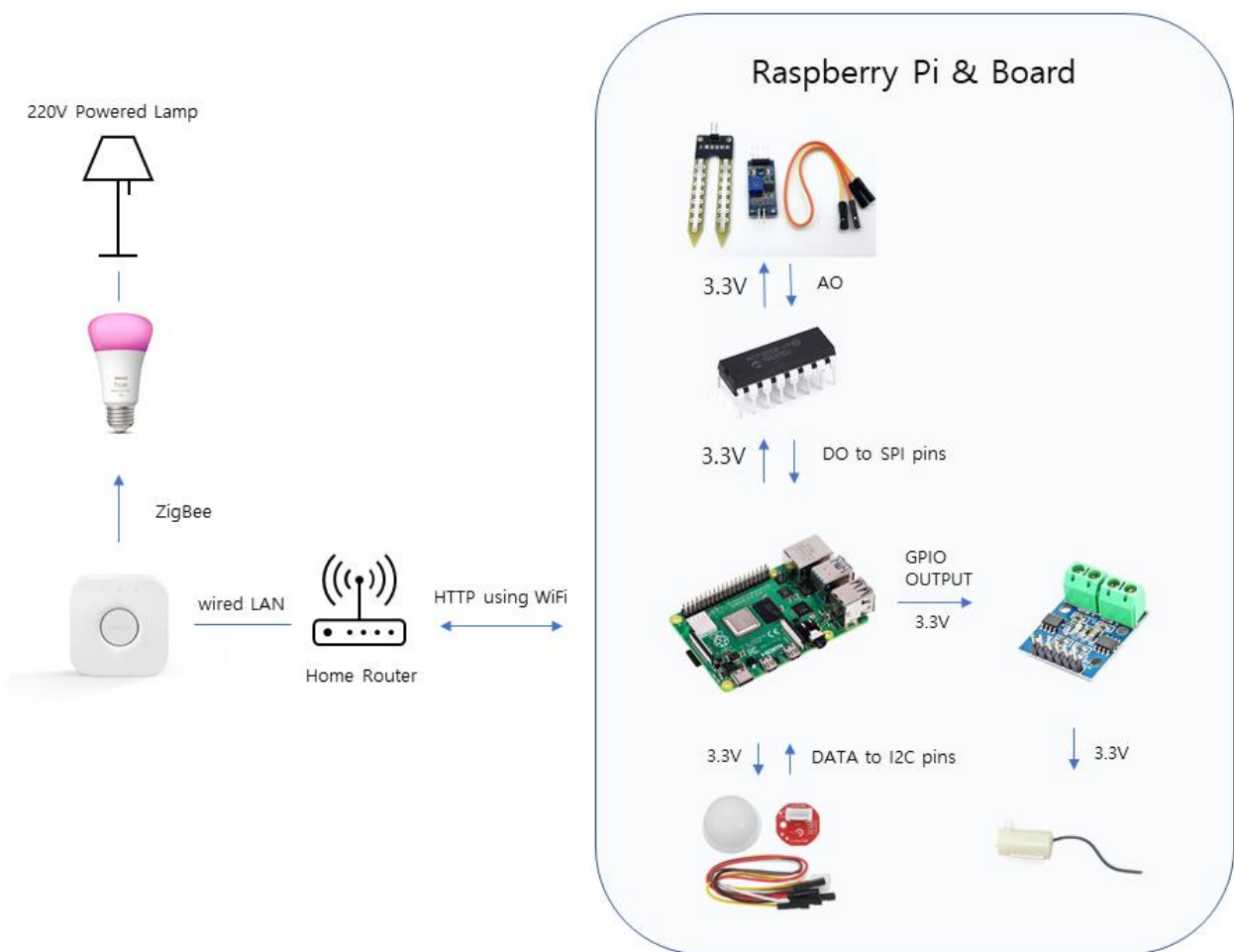
	<p>[Hue Bridge]</p> 	<p>Hue 전구를 제어하기 위해서 선택했다. Hue API에 맞는 HTTP 메시지가 input으로 들어오면, 이를 ZigBee 신호로 바꾸어 전구에게 발신한다. 이 Zigbee 신호로 Hue 전구가 작동한다.</p>
--	--	---

2) 회로 배선



센서와 대부분의 부품을 Raspberry Pi와 연결된 Bread Board에 배치했다. 다만, Hue Lamp와 Bridge는 Raspberry Pi 및 Board와 직접적인 회로 연결 없이 별개의 전원을 이용해 동작한다.

3) 하드웨어 시스템 구성



○ Hardware 기능 (제어 방법 등 서술)

1) INPUT

- (1) SZH-EK106 수분 센서 : 센서를 통해 측정된 EC(Electrical Conductivity, 전기전도도)에 대한 Analog Output이 ADC (회로 상의 MCP3008-I/P Chip)를 통해서 Digital Input으로 변환된다. 해당 신호는 Raspberry Pi의 SPI pin들을 (Pin 19 – SPI MOSI, Pin 21 - MISO, Pin 23 – SPI SCLK, Pin 24 – SPI CE0) 통해 입력 받는다.
- (2) BH1750 FVI 조도 센서 : 측정된 조도는 Raspberry Pi의 I2C pin들을 (Pin 3 – I2C SDA, Pin 5 – I2C SCL) 통해 입력 받는다.

2) OUTPUT

(1) Water Pump : Raspberry Pi의 GPIO 5번과 6번 pin은, Motor Driver의 A-1A와 A-1B pin을 통해 Pump와 연결되어 있다. Raspberry Pi의 GPIO OUTPUT을 이용하여 펌프를 끄거나 켜다.

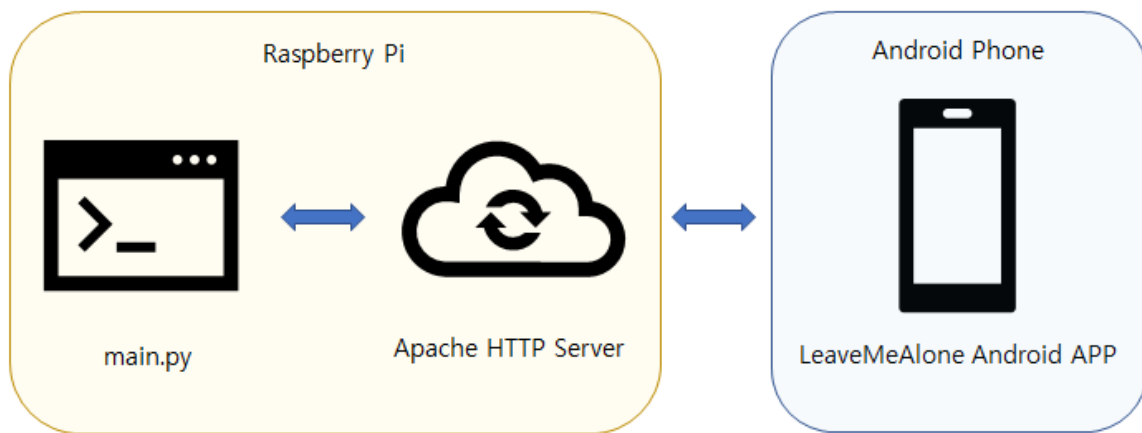
(2) Hue Lamp : Philip Hue White and Color E26, 이하 Hue Lamp는 Raspberry Pi와 회로로 연결이 되어있지 않다. 220V로 작동하는 별개의 스탠드형 램프에 꽂혀 있다. 다만, 스탠드에 전력이 공급된 상태라면, ZigBee 신호로 이 전구의 동작을 제어할 수 있다. 우리는 Hue Bridge, 이하 Bridge를 통해서 ZigBee 신호를 생성하고 Hue Lamp를 제어하였다. 그 과정은 다음과 같다.

1. Hue API에서 지원하는 JSON 형식의 HTTP 메시지가 Bridge의 INPUT으로 들어온다.
2. Bridge는 메시지가 가리키는 전구에게, 메시지에 적힌 관련 동작을 ZigBee 신호로 바꾸어 전달한다.

이러한 과정을 통해 우리는 Bridge를 통해서 Hue Lamp에게 동작을 요청할 수 있다. 재배기 구조상, Bridge는 가정 내 라우터와 유선 LAN으로 연결되어 있다. 그리고 Raspberry Pi는 IP 탐색을 통해 Bridge의 주소를 알아낸 뒤, 해당 주소로 메시지를 보내서 Hue Lamp를 작동시킨다.

○ Software 구성

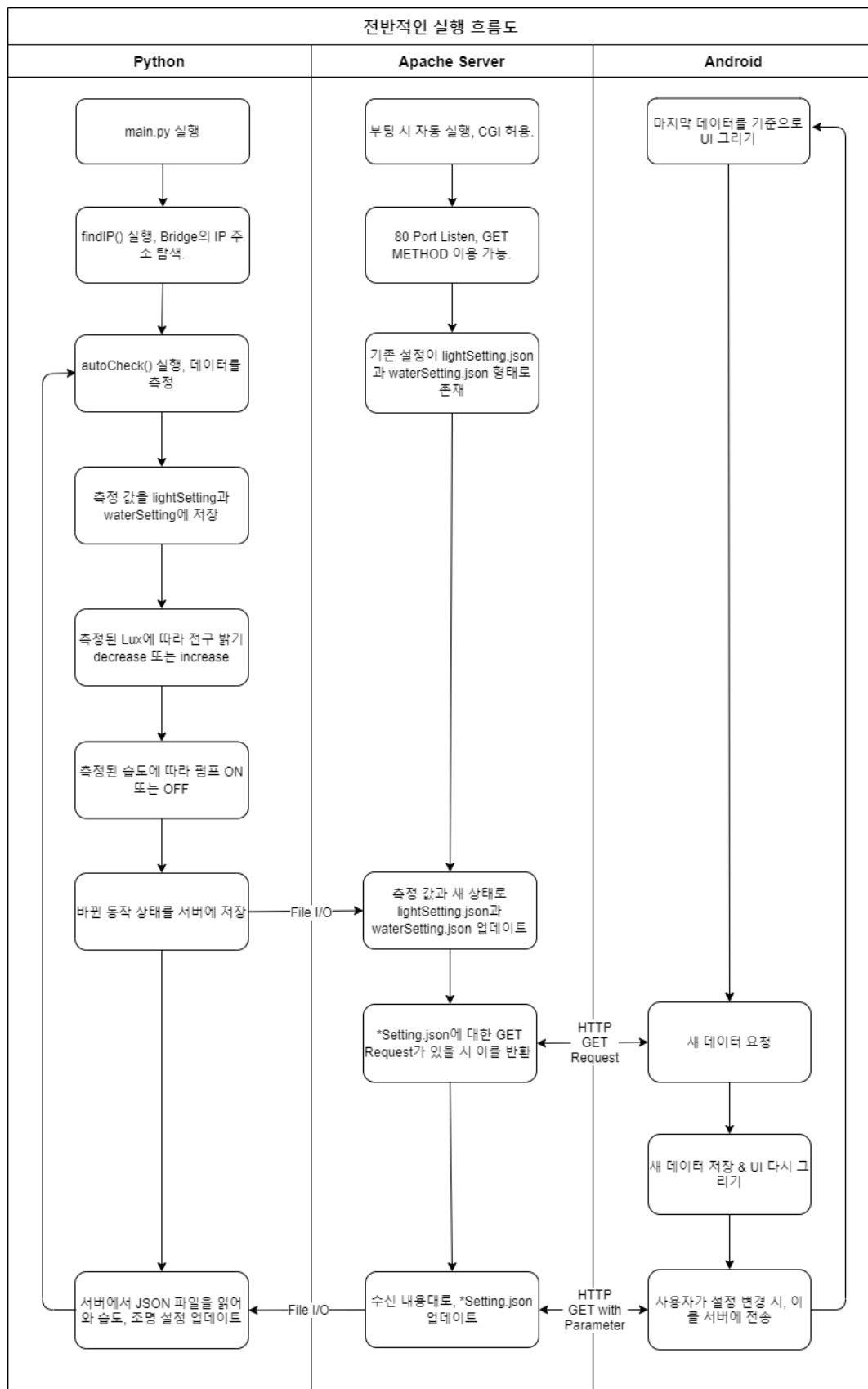
- 간략한 Software 구성은 다음과 같다.

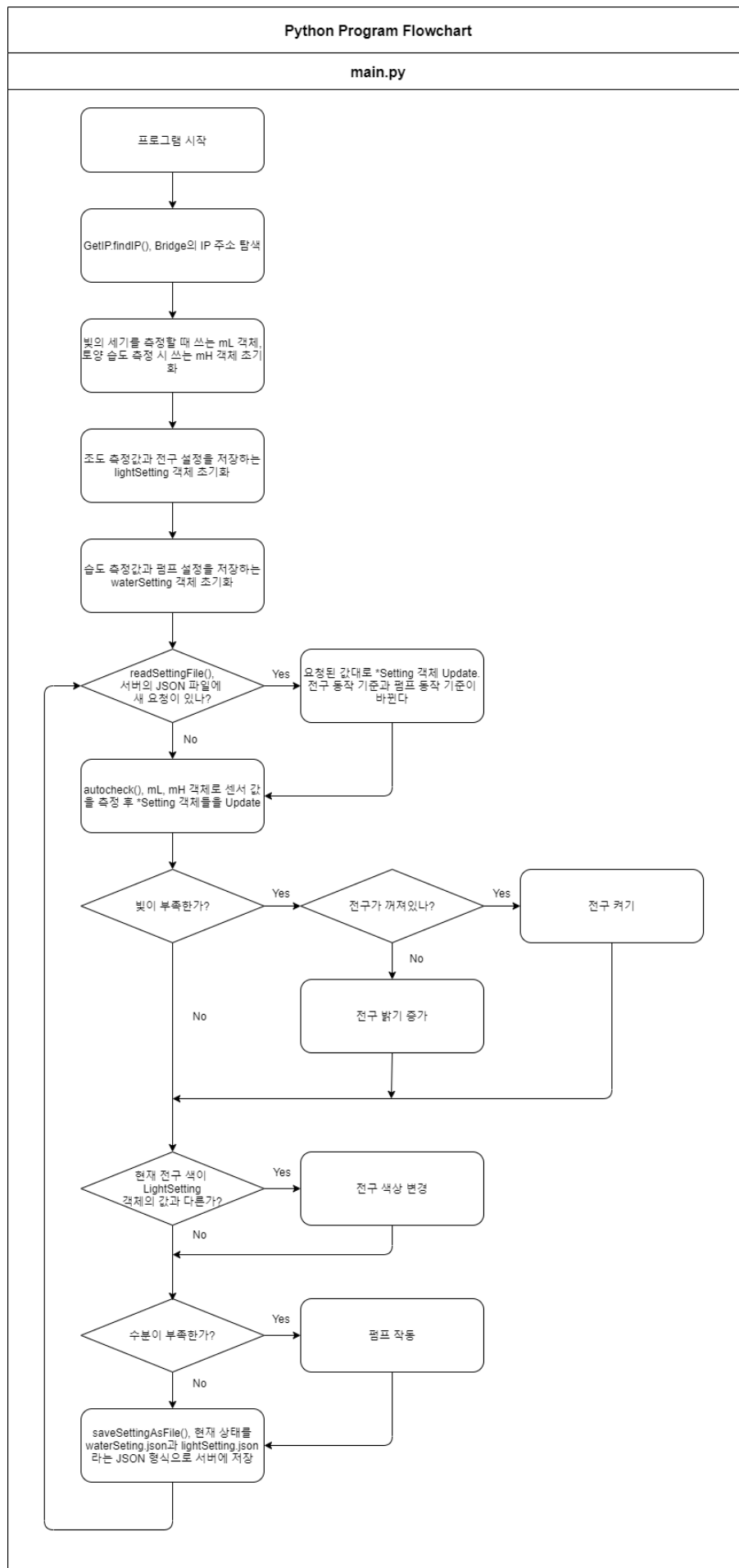


- Raspberry Pi에는 main.py를 실행한 Python 프로세스와 Apache HTTP 서버, 2개의 프로세스가 동작한다. Raspberry Pi 측에서 main.py를 실행시키면, 이는 필요한 패키지와 모듈을 이용해서 데이터를 측정하고 펌프 및 Lamp를 작동시킨다. 그리고 이러한 동작을 1 cycle 반복할 때 마다, 수행 결과를 Apache HTTP 서버에 저장한다. 서버 프로세스는 Raspberry Pi에서 구동된다. 때문에 Raspberry Pi는 제어기와 서버 역할을 겸한다. 그리고 이렇게서버에 저장된 정보를 이용하기 위해서, 사용자용 Android 앱이 존재한다. Android 앱에서 사용자는 원하는 대로 설정을 바꾸고 저장한다. 그러면, 이는 HTTP 메시지 형식으로 Apache 서버에게 전달된다. 전달된 값에 따라, 서버 정보가 수정되고 Raspberry Pi는 새 정보를 기준으로 관리 동작을 수행하게 된다.

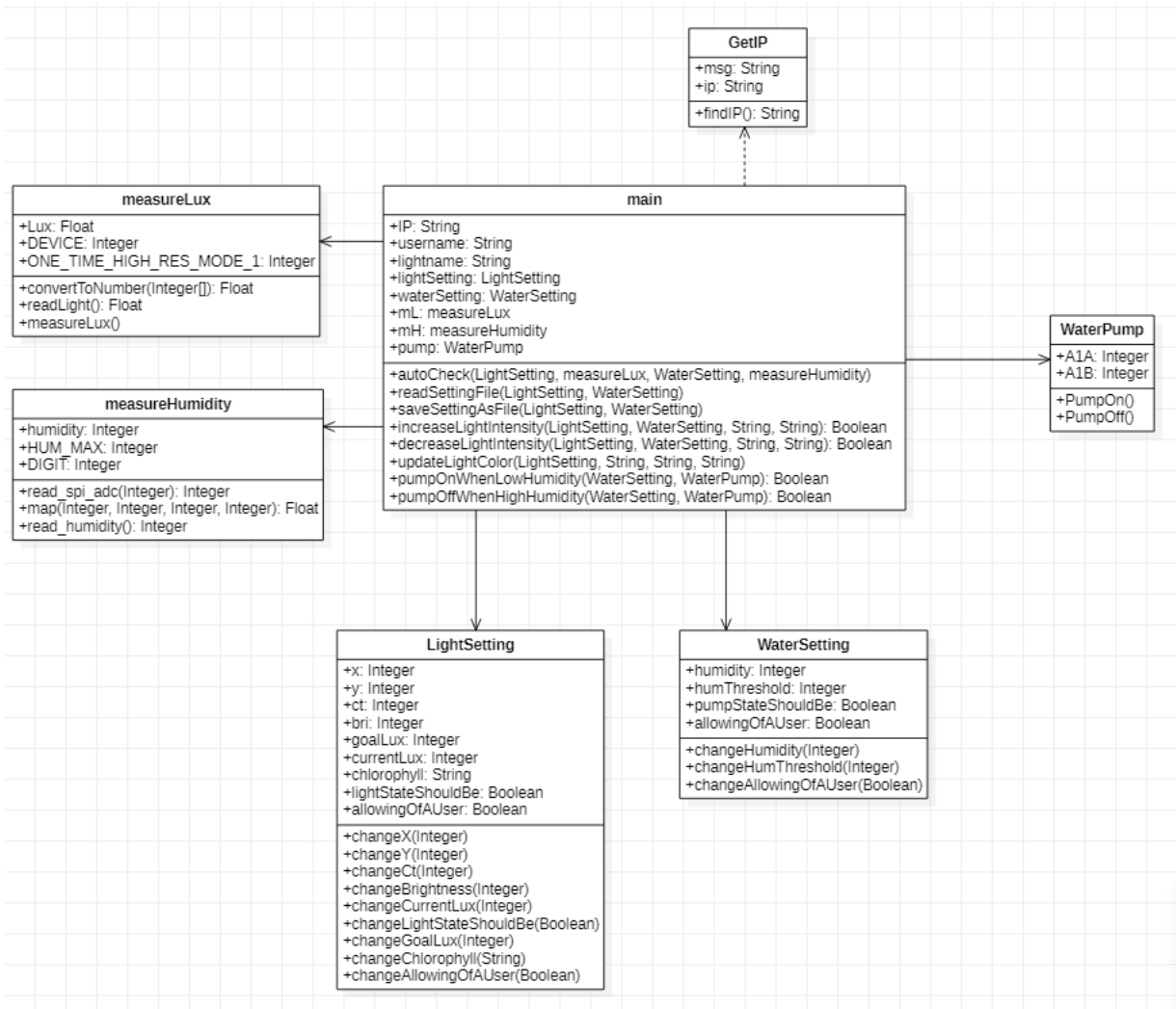
○ Software 설계도 (흐름도 및 클래스 다이어그램 등 / 개발언어에 따라 선택)

- *Setting은 lightSetting과 waterSetting 두 개를 동시에 가리키는 것이다.





- main.py 클래스 다이어그램



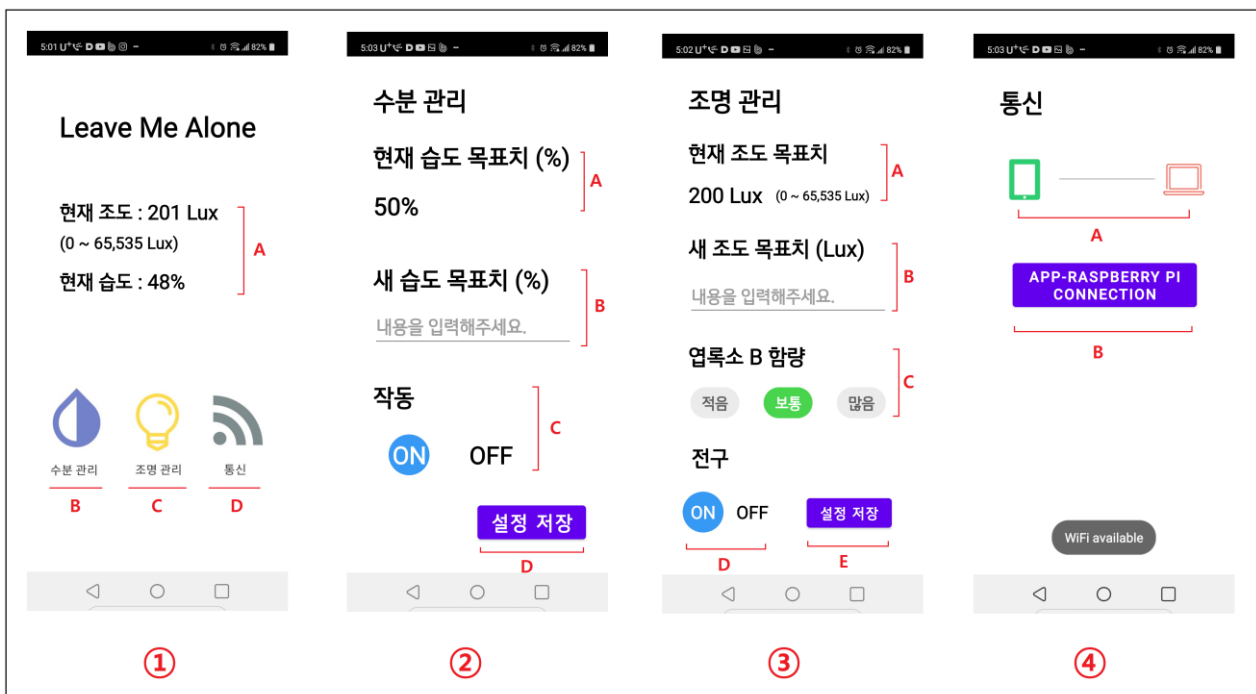
○ Software 기능 (필요 시 알고리즘 설명 포함)

- 습도 측정 : 습도를 측정하여 사용자에게 보여준다.
- 조도 측정 : 조도를 측정하여 사용자에게 보여준다.
- 목표 조도 설정 : 필요로 하는 최소한의 조도를 정한다.
- 목표 습도 설정 : 필요로 하는 최소한의 습도를 정한다.
- 전구 파장 변경 : '엽록소B 적음', '엽록소B 보통', '엽록소B 많음'의 3가지 선택지 중에서 하나를 골라서 원하는 파장의 조명을 제공한다.
- 습도 유지 : 정한 습도 보다 습도가 낮아지면 펌프가 자동으로 작동한다.

- 조도 유지 : 목표 조도 보다 측정되는 조도가 낮으면 전구가 켜진다. 이 때 전구의 밝기는 목표 조도를 유지하는 수준에서 조절된다. 주변이 조금 어두우면 전구가 약하게 켜지고, 많이 어두우면 밝게 켜진다. 주변 환경만으로 목표 조도를 충족하면 전구가 꺼진다.
- 전구 끄기 : 조도 유지 기능을 끈다. 전구가 어떠한 경우에도 작동하지 않는다.
- 펌프 끄기 : 습도 유지 기능을 끈다. 펌프가 어떠한 경우에도 작동하지 않는다.

○ 프로그램 사용법 (Interface)

- 다음은 재배기를 제어할 때 사용하는 안드로이드 앱의 인터페이스이다.



번호		설명
①	A	어플리케이션 실행 시 처음 보이는 화면이다. 측정된 조도와 습도 수치가 표시된다.
	B	② 그림 페이지로 이동하는 버튼이다.
	C	③ 그림 페이지로 이동하는 버튼이다.
	D	④ 그림 페이지로 이동하는 버튼이다.

②	A	현재 설정된 목표 습도를 보여준다.
	B	사용자는 이곳에 새 목표 습도를 입력할 수 있다.
	C	ON을 선택하면 습도 유지가 이루어지고, OFF를 선택하면 펌프가 어떠한 경우에도 작동하지 않는다.
	D	이 저장 버튼을 누르면 선택한 옵션들이 실제로 적용된다.
③	A	현재 설정된 목표 조도를 보여준다.
	B	사용자는 이곳에 새 목표 조도를 입력할 수 있다.
	C	엽록소 B 함량을 고르는 버튼이다. '적음'을 고르면, 조명이 4:1의 적색광과 청색광의 조합으로만 이루어진다. '많음'을 고르면, 스펙트럼에서 녹색광 비율이 증가한다. '보통'을 고르면, 조명이 '적음'과 '많음' 사이인 스펙트럼의 빛을 제공한다.
	D	ON을 선택하면 조도 유지가 이루어지고, OFF를 선택하면 조명이 어떠한 경우에도 작동하지 않는다.
	E	이 저장 버튼을 누르면 선택한 옵션들이 실제로 적용된다.
④	A	스마트폰과 Raspberry Pi 사이의 연결 상태를 보여주는 이미지이다. 연결 상태에 따라 표시되는 이미지가 달라진다.
	B	스마트폰의 Wi-Fi 관리 메뉴로 이동하는 버튼이다. 스마트폰이 Raspberry Pi와 미 연결 상태일 때 사용한다. 사용자는 Android의 Status bar를 이용하지 않고 도, 앱 내에서 스마트폰의 Wi-Fi 관리 메뉴로 진입하여 Raspberry Pi가 있는 네 트워크를 찾아 연결할 수 있다.

○ 개발환경 (언어, Tool, 사용시스템 등)

- S/W 개발환경

Ubuntu	Raspberry Pi 상에 Linux Ubuntu 20.04를 설치한 후, 개발 및 동작 환경을 구축하였다.
Python	재배기의 관리 동작을 수행하는 프로그램은 Python 3.8.6에서 동작하게 구성하였다.
IDLE	통합개발환경(IDE)은 Python 프로그램 작성 시, IDLE을 이용하였다.
Apache HTTP Server	Apache 2.4.46 버전을 HTTP 메시지 처리를 위한 서버로 사용했다.
Android	사용자 App을 Android 10(API level 29)에서 동작하게 만들었다.
Kotlin & Android Studio	Android App을 Kotlin 1.3.72 버전을 이용해 만들었으며, 해당 작업은 Android Studio에서 수행했다.
cgi	Apache 서버에서의 메시지 처리를 위해 Python의 cgi 라이브러리를 이용했다.
socket	Bride의 IP 탐색을 위해 UPnP를 사용하였으며, 이 과정에서 Python의 socket 라이브러리를 이용했다.
smbus2	I2C pin의 데이터를 읽어 오기 위해 Python의 smbus2 라이브러리를 사용했다.
spidev	SPI 통신을 이용하기 위해 Python의 spidev 라이브러리를 사용했다.

□ 개발 프로그램 설명

○ 파일 구성

- Raspberry Pi의 재배기 동작 관련 디렉토리

```
|—main.py – 재배기를 작동시킬 때 쓰는 파일이다. 이 파일에서 LightControl과
|           WaterControl 패키지, 그리고, FileIO.py, controlOfMain.py의 함수들을
|           이용해서 재배기를 작동한다.
|
|—FileIO.py – JSON 파일을 입출력 할 때 사용하는 파일이다.
|           main.py에서 서버 디렉토리에 재배기의 상태를 JSON 파일로 저장할 때 쓴다.
|           또한, Android 앱에서 사용자가 설정을 바꿨을 때, 새 설정이 전송되어 온다.
|           그러면 이 설정을 JSON으로 읽어야 하는데, 이 때도 이 FileIO.py 파일을
|           이용한다.
|
|—controlOfMain.py – 조도에 따라 전구를 조정하는 것과 관련 있는 파일이다.
|           어느 조건일 때, 전구를 켜지 끌지, 혹은 얼마나 밝기를 조절할지 등의
|           기준을 이 파일에서 정한다. 또한, 펌프 동작에 대한 기준도 여기서
|           정한다.
|
|—LightControl – 전구 동작과 관련된 python 패키지이다.
| |
| |—CustomException.py – 예외 처리를 위한 파일이다. Bridge를 이용할 때 발생하는
| |                       오류나 조도 측정 시 생기는 오류, 사용자가 잘못된 Input을
| |                       넣어서 발생하는 오류 등을 처리할 때 쓴다.
| |
| |—GetIP.py – Raspberry Pi에서 Bridge의 IP 주소를 탐색할 때 사용하는 파일이다.
| |
| |—GetUsername.py – Bridge를 사용하려면 최초에 username 생성이 필요하다.
| |                  처음 재배기를 쓸 때 username을 생성하는데 쓰는 파일이다.
| |
| |—LightControl.py – Bridge에 메시지를 보낼 때 사용하는 파일이다. 메시지 전송을
| |                  통해서 전구를 켜거나 끄거나, 밝기를 조절하거나 색을 바꾸는 등의
| |                  동작을 수행한다.
| |
| |—LightSetting.py – 측정된 조도, 점등 기준, 전구의 색상과 밝기 등 재배기가 저장할
| |                  데이터의 타입과 개수를 정하는 파일이다. 이 파일에서
```


└─ LightManagement.kt – 조명 관리창과 그 Activity를 위한 파일이다. 사용자가
 | 새로운 설정을 입력하면 이를 App 폴더에 저장한다.
 | 그리고, 수정된 정보를 Query string에 담아서 Raspberry Pi에
 | 전송한다. 이 과정에서 HttpURLConnection을 이용한다.
 |
 └─ MoistureManagement.kt – 수분 관리창과 그 Activity를 위한 파일이다.
 | 하는 일은 LightManagement.kt와 비슷하다.
 |
 └─ Communication.kt – 통신창과 그 Activity를 위한 파일이다. Android 기기의 WiFi 연결
 상태를 확인하는 기능을 해준다.

○ 함수별 기능

- 재배기 동작 관련 함수들

파일명	함수명	기능
main.py	autoCheck	빛과 습도를 측정하여 관련 객체에 저장한다.
FileIO.py	saveSettingAsFile	빛과 습도 데이터를 담고 있는 객체의 정보를 json 파일로 저장한다.
	readSettingFile	json 파일에서 정보를 읽어와서, 빛과 습도를 관리하는 객체의 정보를 변경한다.
controlOfMain	increaseLightIntensity	조도가 낮으면 전구를 켜고, 전구 세기를 증가시킨다.
	decreaseLightIntensity	조도가 높으면 전구 밝기를 감소시키거나, 전구를 끈다.
	updateLightColor	빛 객체의 데이터에 맞추어 전구 색을 바꾼다. 빛의 파장을 조절한다.
	pumpOnWhenLowHumidity	습도가 낮으면 펌프를 작동시킨다.
	pumpOffWhenHighHumidity	습도가 충분한데 지금 펌프가 작동 중이라면, 펌프를 끈다.

- 재배기 동작에 관한 것 중에 LightContol 패키지 안의 함수들

파일명	함수명	기능
GetIP.py	findIP	홈 네트워크 내에 존재하는 Bridge의 IP 주소를 탐색한다. SSDP(Simple Service Discovery Protocol)에 기반한 UPnP(Universal Plug and Play) 프로토콜을 사용한다. 소켓을 이용해 IP 주소 239.255.255.250로 탐색 메시지를 보낸다. 응답을 받으면, 이 텍스트를 parsing하여 Bridge의 IP 주소를 알아낸다. 혹시, UPnP 탐색에 실패하면 Bridge의 연결 정보를 저장하는 포털에서 IP 주소를 가져온다.
GetUsernamep.py	GetUsername2	재배기를 최초 사용 시 username을 얻기 위해 사용한다. Bridge의 IP 주소로 username을 요청하는 HTTP 메시지를 보낸다.
LightControl.py	getLightList	현재 재배기에 등록된 전구 목록을 확인한다.
	lightOff	Bridge에게 전구를 꺼 달라는 메시지를 보낸다.
	lightOn	Bridge에게 전구를 켜 달라는 메시지를 보낸다.
	changeXY	CIE color space에서 정의하는 x,y 값을 이용해 전구 색을 바꾼다.
	setColorToDefault	전구 색을 일반적인 전구 색인, 주광색으로 바꾼다.
	changeColorTypeA	엽록소 B가 가장 적은 식물에 맞도록 전구색을 바꾼다.
	changeColorTypeB	엽록소 B가 평범한 양으로 있는 식물에 맞도록 전구색을 바꾼다.
	changeColorTypeC	엽록소 B가 많은 식물에 맞도록 전구색을 바꾼다.
	changeBrightness	전구의 밝기를 변경한다.

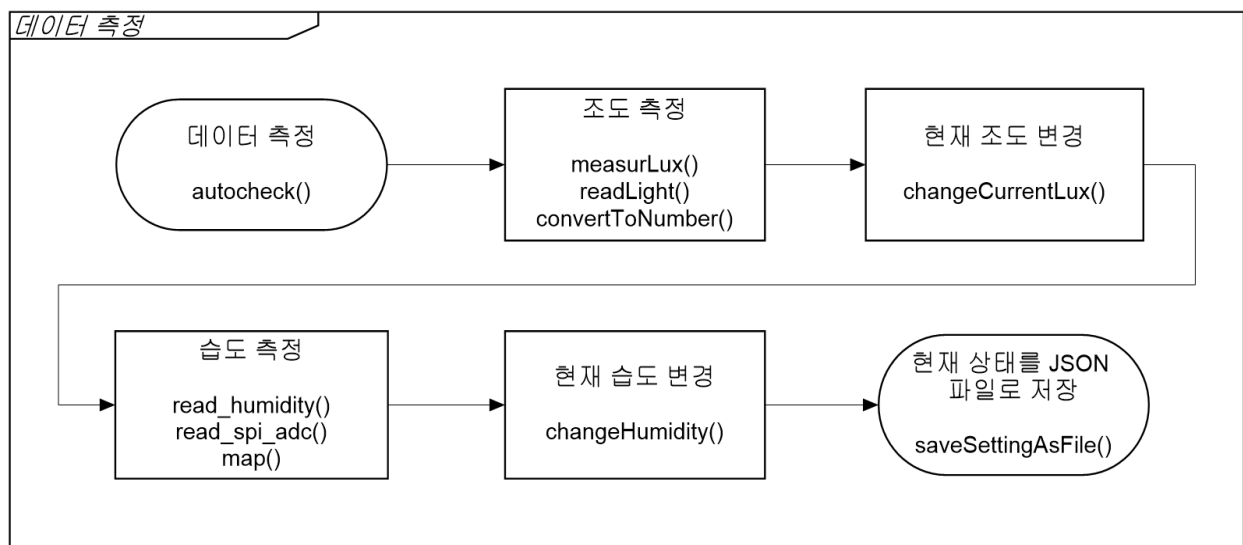
LightSetting.py	changeCurrentLux	새로 측정된 조도 데이터가 있으면, 해당 값으로 현재 조도를 교체한다.
	changeGoalLux	사용자가 조도 설정을 변경했다면, 재배기가 유지해야 하는 최소 조도량을 변경한다.
	changeChlorophyll	사용자가 식물의 엽록소 B에 대한 설정을 변경했다면, 조명이 제공해야 하는 전구색 설정을 바꾼다.
	changeAllowingOfAUser	사용자가 App에서 전구 Off 스위치를 눌렀다면, 현재 재배기 상태와 무관하게 전구를 꺼버린다. 혹은 전구 On 스위치를 눌렀다면, 현재 재배기 상태에 맞추어 전구를 다시 켜다.
measureLux.py	convertToNumber	2byte serial bits를 2byte의 실수로 바꾼다. 우리는 smbus에서 2byte로 된 serial data를 가져오므로, 이를 사용하기 위해서 unserialization 혹은 unmarshalling이 필요하다. 데이터의 상대 주소를 고려하여 2byte serial data를 실수화 한 뒤, 특정 값으로 이를 나누어 조도계의 오차를 보정한다.
	readLight	Smbus를 생성하고, 해당 bus를 이용해 조도계 주소에서 측정 값을 2byte로 읽어온다. 이를 convertToNumber 함수로 처리한 뒤 반환한다.
	measureLux	조도를 측정하는 함수다. 함수 내에서 readLight를 호출해 사용한다. 측정된 조도를 정수화 하여 반환한다.

- 재배기 동작에 관한 것 중에 WaterContol 패키지 안의 함수들

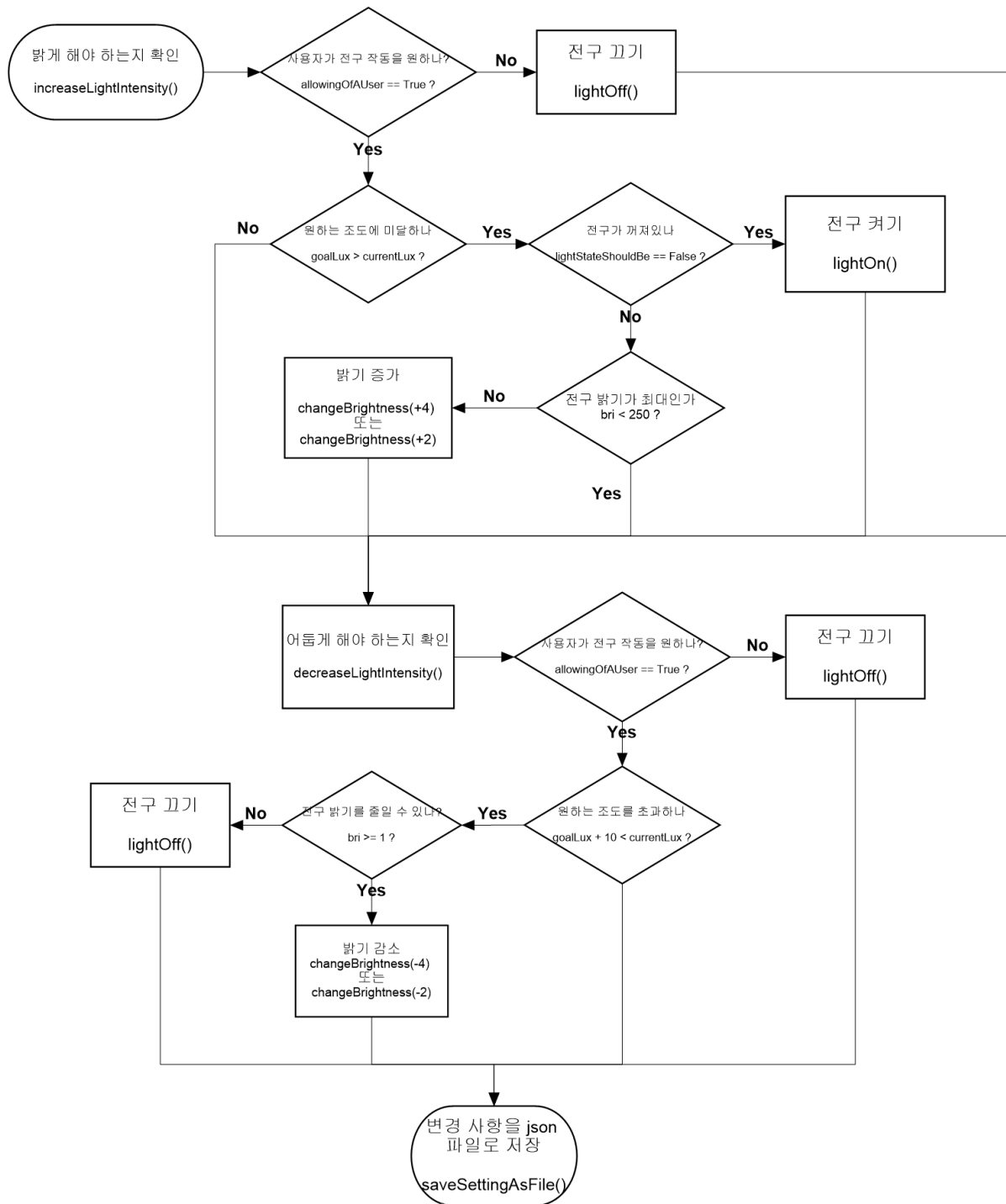
파일명	함수명	기능
WaterSetting.py	changeHumidity	새로 측정된 습도 데이터가 있으면, 해당 값으로 현재 습도를 변경한다.
	changeHumThreshold	워터펌프가 돌아가는 기준인 최소 습도를 변경한다.
	changeAllowingOfAUser	사용자가 App에서 워터펌프 Off 스위치

		를 눌렀다면, 현재 재배기 상태와 무관하게 펌프를 멈춘다. 혹은 워터펌프 On 스위치를 눌렀다면, 현재 재배기 상태에 맞추어 펌프가 작동하도록 한다.
WaterPump.py	PumpOn	워터펌프를 작동시킨다.
	PumpOff	워터펌프를 끈다.
measureHumidity.py	read_spi_adc	수분 센서에서 ADC(Analog - Digital Converter), 디지털화된 전류 세기를 읽어온다. 그리고 이를 0~1023의 값으로 정수화 한다.
	map	0~1023로 측정된 전류 세기를 0~100의 퍼센트화 된(percentage wise) 습도로 변환한다.
	read_humidity	read_spi_adc와 map 함수를 사용하여, 백분율로 표현되는 토양 습도 알아낸다.

○ 주요 함수의 흐름도



전구 조절 과정



○ 기술적 차별성

1) IoT화 가능

- 홈 네트워크 하에서 작동시키는 걸 가정하고 만들었으나, 재배기와 사용자 휴대폰 간 통신이 HTTP 기반으로 이루어진다. 때문에 Internet 접근이 가능한 원격지에서 사용자가 식물 상태를 확인하고, 환경을 조절하도록 개선할 수 있다.

□ 개발 중 발생한 장애요인과 해결방안

○ 조도 측정 오류

- BH1750FVI 센서를 이용해 조도를 측정해보았으나, 별도의 handheld 조도계로 측정한 수치와 다르게 나왔다. 예비 부품으로 교체하여 측정하거나, 사용한 resolution(측정 정밀도)를 변경해보았으나 결과가 같았다. 이후에 SMBus를 이용해 I2C block 데이터를 읽어오는 함수를 잘못 사용했는지 확인하였다. 그래서 사용한 package의 description을 참조하였으나 별다른 이상을 발견하지 못하였다.
- 추후에, 우리가 사용한 센서는 방수/방진을 위해 반투명 덮개가 있으므로 window influence (차양 효과)로 오차가 발생하는 것을 알아냈다.
- SMBus에서 읽어온 데이터를 숫자로 바꾸는 convertToNumber 함수에 값을 증폭하는 수식을 추가하여 오차를 보정하였다.

○ Bridge 성능 한계

- Bridge를 통한 전구 제어 command는 Philips에 따르면 초당 10회까지 가능하다. 그러나, 실 사용 시 느릴 때는 초당 5번 가량만 실행되었다. 반면에, 라즈베리파이를 이용한 센서 측정은 그 이상 빠르게 동작했다. 때문에 전구 제어와 데이터 측정 간 속도 차이가 존재했다. 이로 인해서 데이터 측정은 거의 실시간으로 되지만, 이에 대한 전구 제어가 뒤늦게 이뤄졌다. 그러므로, 전구가 이전 측정치에 대응하여 동작했다. 그래서 일정 밝기를 유지하지 않고, 어두워졌다 밝아지는 걸 반복하며 깜박거렸다.
- Bridge의 메시지 처리 속도를 반복적으로 측정하여 Worst case를 알아냈다. 그래서 이에 따라 센서 측정 시간을 초당 최대 4회 정도로 늦추었다. 그리고 이 측정 속도에 맞추어 전구도 초당 4회까지 상태 변화를 한다고 가정하고, 어두워지거나 밝아질 때의 밝기 변화량을 수정하였다.

□ 개발결과물의 차별성

○ 식물의 생육 속도 조절

- 광도를 세밀히 조절할 수 있기 때문에, 광보상점(식물 생존에 필요한 최소 광도)과 광포화점(광합성을 최대 효율로 행하는 광도) 사이에서 사용자는 조명을 조절하여 식물의 생육 속도를 조절할 수 있다.

○ 식물의 무늬 형성 조절

- 산세베리아, 드라세나 등의 반입식물(잎에 무늬가 있는 식물, variegated plant)들은 제공되는 광도와 빛의 파장에 따라, 무늬가 짙고 많아지거나, 옅고 적어진다. 사용자는 광보상점을 상회하는 수준에서 낮은 광도를 제공하고, 엽록소 B에 맞는 파장의 빛을 제공해서 무늬를 늘릴 수 있다. 반대로, 높은 광도를 제공하고, 엽록소 B가 적은 식물에 적합한 빛을 제공하여 무늬를 줄일 수도 있다.

□ 개발 일정

No	내용	2020年 / 2021年											
		11月			12月			1月			2月		
1	아이디어 선정 & 자료조사												
2	Usecase & Testcase 작성												
3	HW 설계 & 제작												
4	SW Planform 구축												
5	조명 조절 알고리즘 개발												
6	Simulation & Debug												
7	펌프 조절 알고리즘 개발												
8	Android App 스터디												
9	UI 설계 & App 개발												
10	Apache 서버 설계 & 구축												

□ 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	권민수	<p>전체 업무 총괄 & 개발</p> <ul style="list-style-type: none"> - Usecase / Testcase 작성 - SW Platform 구축 - 주요 Algorithm 설계 & 구현 - 회로 구성 & HW 조립 - Server 구축 및 스크립트 작성 - Android App 개발 - Bug Fix - 보고서 작성
2	팀원	김도균	<p>설계 보조 & 테스트</p> <ul style="list-style-type: none"> - Usecase / Testcase 작성 - Simulation 수행 - Bug report