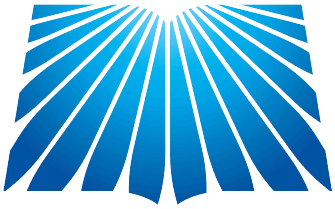
제출일: 2020년 12월 11일



스마트 공기청정기

조원

학번: 20184902, 이름: 진미강

학번: 20173740, 이름: 정택수

목차

I. 작품개요

기존 공기청정기가 가진 기능인 스위치를 이용해 공기청정기의 동작을 제어하고, 미세먼지 센서의 측정값을 LCD로 표시해주는 기능과, 이에 더해 애플리케이션을 통해 실시간 미세먼지 값을 모니터링하고, 최근 24시간, 7주일 단위로 미세먼지 변화를 알 수 있는 그래프를 확인할 수 있는 기능과 가정 또는 외부에서도 공기청정기의 동작을 제어할수 있는 IOT기능, 그리고 먼지센서 농도에 따라 팬 속도를 제어하는 자동모드 기능을 라즈베리파이를 이용해 파이썬으로 구현하였습니다. 애플리케이션은 파이썬 웹서버(flask)와 연동하여, HTML, CSS, JS를 이용해 웹과 서버간 통신 및 DB데이터 처리를 구현하였습니다

II. 업무분담

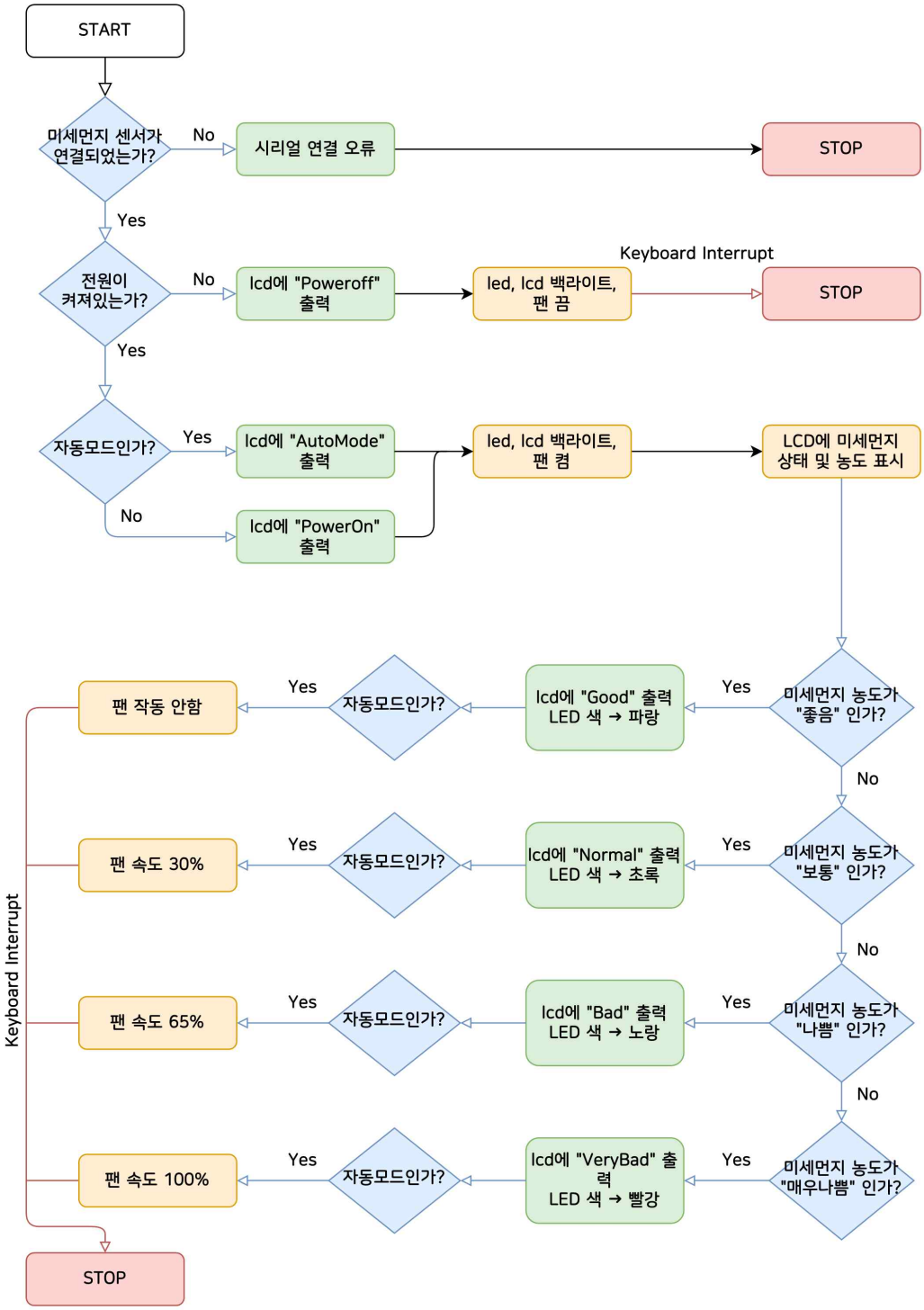
	참여 여부	
분류	정택수	진미강
아이디어 (정보수집)	○	○
부품 구매	○	○
SW	△	○
HW	○	△
작품 테스트	○	○
1차 발표	○	
2차 발표	○	△
3차 발표		○
최종 보고서 작성		○

III. 진행일정표

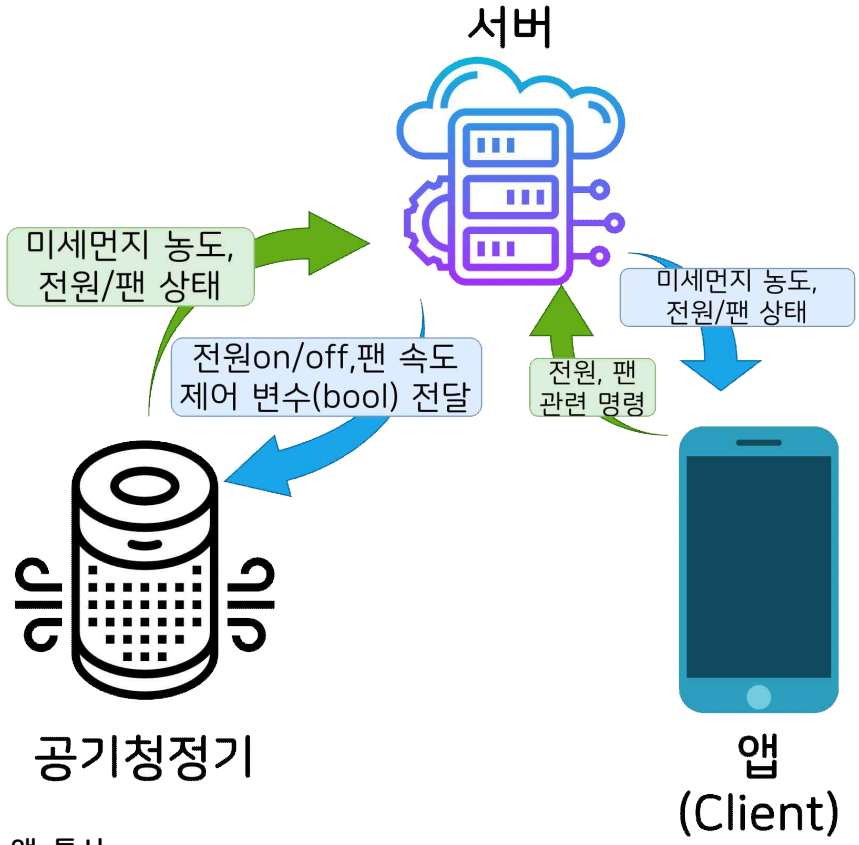
계획: 일정:

IV. 동작 설명

1. 프로그램 블록도



2. 상세 동작 설명



1) 서버 및 앱 통신

Python의 웹 서버 라이브러리 Flask를 사용하여, 앱 및 웹 클라이언트와 기기간의 통신을 구현하였습니다. 학교 및 가정의 내부 방화벽에 제한 없이 사용할수 있게 서버를 라즈베리 파이 외부에서 구동시켜, 라즈베리파이와 클라이언트가 각각 서버를 거쳐 http통신으로 데이터를 주고받을수 있도록 구현했습니다. 미세먼지 센서 데이터를 라즈베리 파이에서 측정하고 서버로 전송합니다. 그리고 앱에서는 데이터를 받아와 화면에 표시하고 전원 및 팬 상태 제어 명령을 서버를 거쳐 라즈베리파이로 전달합니다. 서버로 전송된 미세먼지 데이터는 데이터베이스에 저장되고, 저장된 데이터베이스로 앱에 최근 1주일, 최근 24시간동안의 미세먼지 변화 데이터를 그래프로 나타냅니다. 또한 서버를 도메인으로 등록하여, 기기에 부착된 QR코드를 통해 간편하게 연결할 수 있도록 하였습니다.

서버와 공기청정기간의 명령 전달을 위해 각 명령마다 bool타입의 변수를 만들어 명령이 요청되면 True로 공기청정기에 전달되고, 명령이 전달된 것을 확인하면 False로 바꾸는 방식으로 http 양방향 통신을 구현하였습니다. 그리고 공기청정기에서 서버로 미세먼지 데이터를 보낼 때는 dict 변수를 json 타입으로 변경해서 서버에 POST요청을 보내는 방식으로 통신한다.

서버와 앱 간의 통신은 javascript에서 jquary 라이브러리에서 ajax를 사용하여 새로고침 없이 서버로부터 데이터를 받아오고, 앱 화면에 실시간으로 디스플레이 됩니다. 그리고 전원 on/off 및 팬 속도 제어는 서버 하위 루트(ex;"/poweron","/poweroff)에 요청이 들어오면 위에서 언급한 명령 변수를 True로 변경한다.

2) 구글 어시스턴트

ifttt.com에서 제공하는 서비스를 이용하여, 구글 어시스턴스 명령어를 커스텀으로 생성했습니다. 웹서버를 기반으로 구현했으며, 구글 어시스턴트로 특정 명령을 실행하면, 지정된 웹 주소에 Web Request를 전송하는 방식으로, 서버에서 해당 주소에 Request가 들어오면 전원 및 팬 속도 제어를 할 수 있도록 구현하였습니다. 예를 들어 "OK Google, 공기청정기 켜줘" 라고 하면 ifttt.com을 거쳐 "http://aircleaner.software/poweron"으로 Web Request를 요청하여 공기청정기의 전원이 켜지게 됩니다.

3) 라즈베리 파이 GPIO제어 및 기본동작

초기 전원은 0으로, 무한 루프 안에 코드를 작성하여, 루프가 1번 순환할때마다 PMS-7003센서로부터 미세먼지 데이터를 읽고, 데이터베이스에 저장합니다. 그리고 서버의 발신 경로(http://aircleaner.software/senddata)에 GET Requset를 보내 클라이언트로부터 입력받은 명령을 수신받고 명령에 따라 프로그램의 상태 변수를 변경합니다. 현재 공기청정기의 상태(전원, 팬 속도, 미세먼지 데이터)를 json 타입으로 "../receivedata" 에 POST 요청을 보내 데이터를 전송합니다.

버튼 제어는 메인 루프와 별개로 실행되는 함수로 멀티쓰레딩을 이용했습니다. power_state 변수가 0일 때는 전원 꺼짐, 1일 때는 켜짐, 2일 때는 자동 모드로 지정하여 택트 스위치를 클릭할 때마다 전원 상태가 변경, 순환됩니다. 이는 메인루프가 실행중일 때 누르면 현재 루프가 끝나고 다음 루프가 시작할 때 전원 상태가 변경됩니다. 팬 버튼도 마찬가지로 스위치를 클릭할 때마다 모터 드라이버의 pwm 핀의 value 값이 0.3 → 0.65 → 1.0 으로 변경 및 순환합니다. 이 또한 메인루프와 별개로 누르는 즉시 팬 속도가 변경됩니다.

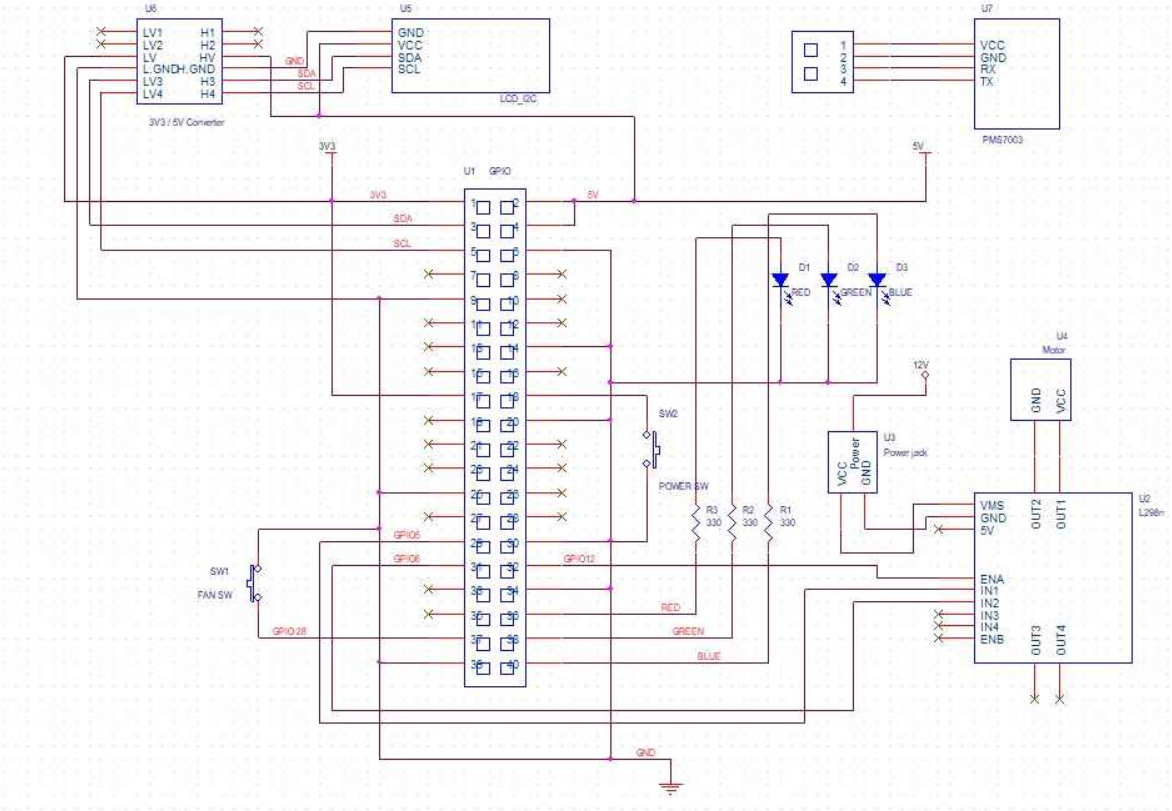
전원 상태가 0이면 LCD, LED, FAN을 끄고 LCD에 "Power Off" 문구를 표시합니다.

전원 상태가 1이면 LCD, LED, FAN을 켜고, LCD에 "Power On" 문구를 1초간 표시하고, 미세먼지 농도에 따라 LCD의 윗줄에는 좋음(GOOD), 보통(NORMAL), 나쁨(BAD), 매우나쁨(VERY BAD) 로 표시하고, 아랫줄에는 PM1.0, PM2.5, PM10 각각의 미세먼지 크기에 따른 농도를 1초간 표시합니다.

전원 상태가 2이면 미세먼지 상태가 "좋음" 이면 팬을 멈추고, "보통" 이면 팬 속도를 30%, "나쁨" 이면 팬 속도를 65%, "매우나쁨" 이면 100%로 동작하며, 팬 속도를 따로 제어할 수 없습니다. 이 외의 동작은 전원 상태 1과 같습니다.

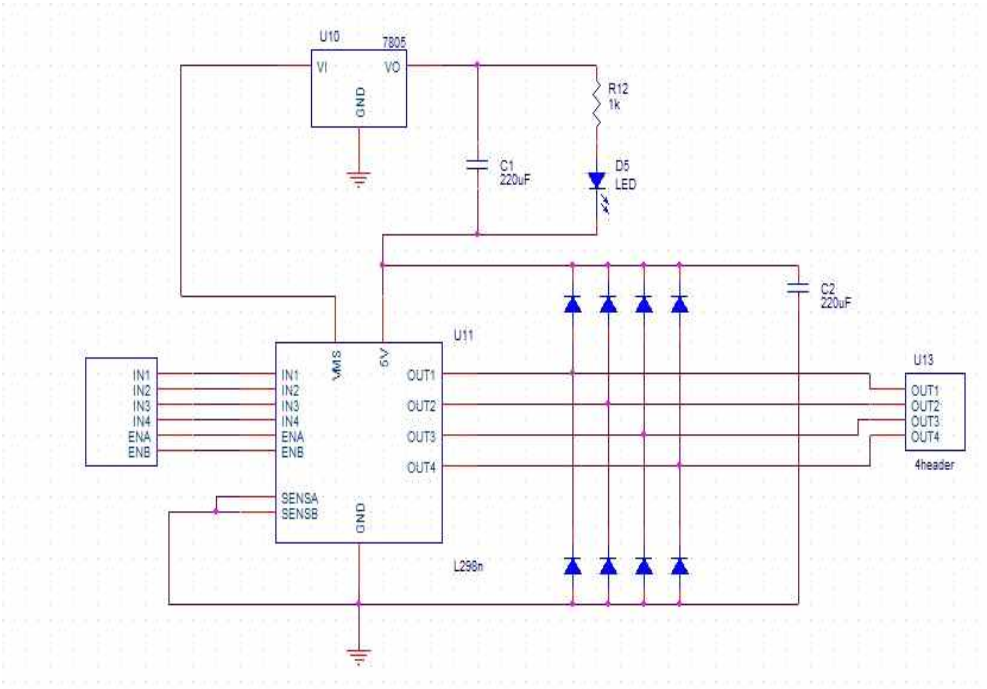
V. 회로도 및 회로 설명

1. 메인 회로



왼쪽 위부터 3.3V 5V 컨버터와 I2C LCD입니다. LCD는 5V를 LCD를 사용해서 라즈베리파이의 GPIO전압인 3.3V로 동작하게 하기 위해 컨버터를 이용해 LV(3.3V) → HV(5V)로 바꾸어주게 됩니다. 그리고 오른쪽에 있는 PMS-7003 먼지센서는 USB를 이용해서 rx, tx, gnd, vcc를 연결해 주었습니다. 그 아래에는 RGB LED가 있는데 이는 모듈로 사용하여 모듈 내부의 저항을 표기했습니다. 그 왼쪽 아래에는 Power SW와 FAN SW가 있습니다. 이는 라즈베리파이 내부의 풀다운 저항을 코딩을 이용해 사용했기에 따로 저항을 추가하지 않았습니다. 마지막으로 왼쪽 아래에는 L298n 모터드라이버가 있습니다. 모터 드라이버에 12V 외부전원을 연결해서 모터를 PWM 핀과 함께 모터의 회전방향 및 속도를 제어합니다.

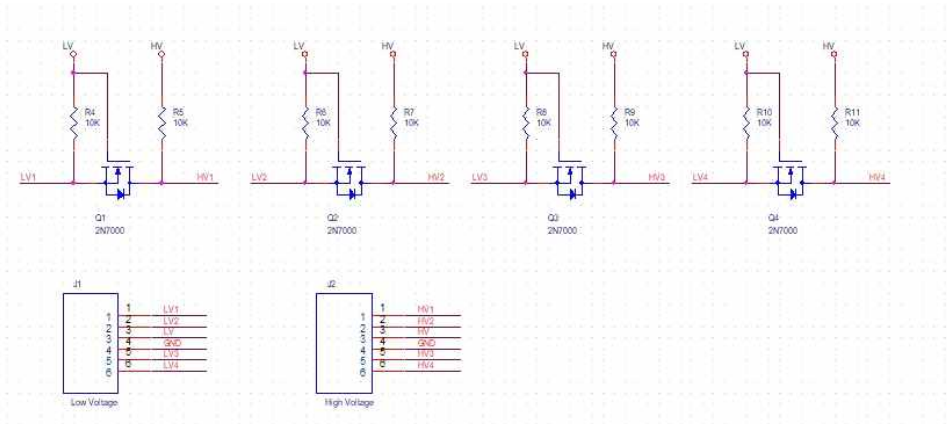
2. 모터 드라이버 회로



L298N 모터 드라이버 모듈 회로로, 외부전원을 입력받아 입력신호에 따라 OUT으로 출력을 내보냅니다. 입력 신호에 따른 출력은 아래 표와 같습니다.

ENA A(B) - PWM핀	INPUT 1(3)	INPUT 2(4)	모터 동작
HIGH	HIGH	LOW	순방향 회전
HIGH	LOW	HIGH	역방향 회전
HIGH	HIGH	HIGH	빠른 정지
HIGH	LOW	LOW	빠른 정지
LOW	X	X	자연 정지(출력없음)

3. 3.3V to 5V logic converter 회로



I2C LCD의 SCL과 SDA핀에 5V의 로직전압을 입력시키기 위해 사용한 로직 컨버터입니다. 트랜지스터와 저항을 이용해 3.3V 로직을 5V 로직으로 변환시켜줍니다.

VI. 프로그램 소스코드

Github 메인 코드: https://github.com/Luvdduk/Air_Cleaner-Main
Github 서버 코드: https://github.com/Luvdduk/Air_Cleaner-Server

1. main.py(공기청정기 메인동작)

```
from gpiozero import Button, RGBLED, DigitalOutputDevice, PWMOutputDevice #gpio 제어
from colorzero import Color # 컬러 설정
from signal import pause # pause 사용 - 코드 진행중 대기 위해서
import time # time.sleep 사용
import serial # 시리얼 라이브러리 - PMS-7003 연결 위해서
from PMS7003 import PMS7003 # 먼지센서 라이브러리
import lcd_i2c as lcd # lcd 라이브러리
from configparser import ConfigParser # 설정파일 저장 - 마지막 팬 속도 기억
import pymysql # mysql 데이터베이스 사용
import requests # 서버 통신을 위해 http 리퀘스트
import json # 서버 통신을 위해 json 타입 - dict 타입간 변환
import datetime # 데이터베이스에 현재 날짜 및 시간 삽입 위해
# 핀 설정
powersw = Button(24 ) # 전원버튼
fansw= Button(26 ) # 팬속버튼
fan_pwm = PWMOutputDevice(12 ) # 모터드라이버 pwm
fan_pin1 = DigitalOutputDevice(5 ) # 모터드라이버 IN1
fan_pin2 = DigitalOutputDevice(6 ) # 모터드라이버 IN2
led = RGBLED(16 , 20 , 21 ) # rgb led

# 초기 전원
power_state=0

#먼지센서 오브젝트
dustlib = PMS7003()

# 시리얼
Speed = 9600 # 보드레이트 9600
SERIAL_PORT = '/dev/ttyUSB0' # usb로 연결함

# DB 연결
dust_db = pymysql.connect(
    user ='luvdduk',
    passwd ='***',
    host ='aircleaner.software',
    db ='air-cleaner',
    charset ='utf8'
)
cursor = dust_db.cursor(pymysql.cursors.DictCursor) #dict형태로 데이터베이스에서 가져옴

# 팬 on/off 변수
ON = 1
OFF = 0
```

```
# 팬속 지정 변수
FULL = 1.0
MID = 0.65
SLOW = 0.3

# config.ini 가 있으면 로드 없으면 에러
try :
    conf = ConfigParser()# 설정파일 로드
    conf.read('config.ini') # 같은 경로에 있는 config.ini 읽어오기
    fan_state = conf['FAN']['fan_speed']# 팬속 로드
    print ("설정파일 팬속도: %s " %fan_state)
except :
    print ("설정파일 로드 오류")

# pwm변수로 지정
if  fan_state == "FULL":
    fan_pwm.value = 1.0 # 팬 속도 100%
elif  fan_state == "MID":
    fan_pwm.value = 0.65 # 팬 속도 65%
elif  fan_state == "SLOW":
    fan_pwm.value = 0.3 # 팬 속도 30%
else :
    print ("설정파일 로드 오류")

lcd.lcd_init()# lcd초기화

def  Button_Ctrl (): # 버튼 제어
    global  power_state
    global  fan_state
    powersw.when_pressed = powerctrl #파워버튼 누르면 실행
    fansw.when_pressed = fan_speedsw #팬속도 조정버튼 누르면 실행
    pause() # 누를때까지 대기

# 파워 꺼짐: 0, 켜짐: 1, 자동: 2
def  powerctrl (): # 전원버튼 누를때마다 실행
    global  power_state
    if  power_state == 0 : # 전원꺼짐 상태일때 전원켬
        power_state = 1 # 전원 상태 켜짐
        print ("전원켬")
        return
    if  power_state == 1 : # 전원켜짐 상태일 때 자동모드로 변경
        power_state = 2 # 전원 상태 자동모드
        print ("자동모드로 변경")
        return
    if  power_state == 2 : # 자동모드일때 전원꺼짐 상태로 변경
        power_state = 0 # 전원 상태 꺼짐
        print ("전원끔")
        return

def  fan_speedsw (): # 팬 스피드 조절, 팬속버튼 누를때마다 실행
    global  fan_state, power_state
    if  power_state == 1 : #일반모드, 전원켜짐 상태에서만 동작
        if  fan_state == "SLOW": # 팬속도가 1단계일때 2단계로 변경
            fan_pwm.value = 0.65 # 팬 속도 65%
            fan_state = "MID" # 팬 상태 2단계
```

```
            conf['FAN']['fan_speed'] = fan_state #팬 상태 저장
            with open ('config.ini', 'w') as configfile: # 파일 쓰기
                conf.write(configfile)
            print ("팬속도: %f "%fan_pwm.value)
            return
        if fan_state == "MID": # 팬속도가 2단계일때 3단계로 변경
            fan_pwm.value = 1   # 팬 속도 100%
            fan_state = "FULL"   # 팬 상태 3단계
            conf['FAN']['fan_speed'] = fan_state #팬 상태 저장
            with open ('config.ini', 'w') as configfile:# 파일 쓰기
                conf.write(configfile)
            print ("팬속도: %f "%fan_pwm.value)
            return
        if fan_state == "FULL": # 팬속도가 3단계일때 1단계로 변경
            fan_pwm.value = 0.3   # 팬 속도 30%
            fan_state = "SLOW"  # 팬 상태 1단계
            conf['FAN']['fan_speed'] = fan_state #팬 상태 저장
            with open ('config.ini', 'w') as configfile: # 파일 쓰기
                conf.write(configfile)
            print ("팬속도: %f "%fan_pwm.value)
            return
    else :
        print ("일반모드에서만 동작")

# 팬 on/off 제어
def fan_power (state ):
    if state: # 1이 입력되면 팬 전원 켬
        fan_pin1.on() # 순방향으로 동작
        fan_pin2.off()
    else : # 팬 전원 끔
        fan_pin1.off()
        fan_pin2.off()

# lcd에 먼지농도 표시, 매개변수로 pm1, pm25, pm10변수를 전달받음
def display_dust (duststate1 , duststate2 , duststate3 ):
    global power_state, fan_state
    # 좋음
    if duststate3 <= 30 and (duststate1 + duststate2) <= 15 :
        lcd.lcd_string("    GOOD    ", lcd.LCD_LINE_1) # lcd에 good표시
        led.color = Color("blue") # led 색 지정 파랑
        if power_state == 2 : #자동모드 팬 끄기
            fan_power(OFF)  # 팬 정지
    # 보통
    elif duststate3 <= 80 and (duststate1 + duststate2) <= 35 :
        lcd.lcd_string("    NORMAL    ", lcd.LCD_LINE_1)# lcd에 normal표시
        led.color = Color("green") # led 색 지정 초록
        if power_state == 2 : #자동모드 팬 속도 30%
            fan_power(ON)  # 팬 동작
            fan_pwm.value = 0.3  # 팬 속도 30%
            fan_state = "SLOW"  # 팬 상태 1단계
    # 나쁨
    elif duststate3 <= 150 and (duststate1 + duststate2) <= 75 :
        lcd.lcd_string("    BAD    ", lcd.LCD_LINE_1)# lcd에 bad표시
        led.color = Color("yellow") # led 색 지정 노랑
        if power_state == 2 :#자동모드 팬 속도 65%
```

```
        fan_power(ON)  # 팬 동작
        fan_pwm.value = 0.65  # 팬 속도 65%
        fan_state = "MID"  # 팬 상태 2단계
    # 매우나쁨
    elif  duststate3 > 150  or  (duststate1 + duststate2) > 75 :
      lcd.lcd_string("   VERY BAD  ", lcd.LCD_LINE_1)# lcd에 verybad표시
      led.color = Color("red") # led 색 지정 빨강
      if  power_state == 2 :#자동모드 팬 속도 100%
        fan_power(ON)  # 팬 동작
        fan_pwm.value = 1  # 팬 속도 100%
        fan_state = "FULL"  # 팬 상태 3단계
    else :
      print ("LCD표기 오류 or 먼지센서 데이터 오류")

    # pm1.0 표시
    lcd.lcd_string("PM1.0: %d  ug/m3 " %duststate1, lcd.LCD_LINE_2)
    powersw.wait_for_press(timeout =2 )
    if  powersw.is_pressed:
      return
    # pm2.5 표시
    lcd.lcd_string("PM2.5: %d  ug/m3 " %duststate2, lcd.LCD_LINE_2)
    powersw.wait_for_press(timeout =2 )
    if  powersw.is_pressed:
      return
    # pm10 표시
    lcd.lcd_string("PM10: %d  ug/m3  " %duststate3, lcd.LCD_LINE_2)
    return


# 메인루프
def  loop ():
    global  power_state, fan_state
    while  True :
      # 먼지센서 동작
      ser = serial.Serial(SERIAL_PORT, Speed, timeout  = 1 )
      buffer = ser.read(1024 )
      if (dustlib.protocol_chk(buffer)):
        data = dustlib.unpack_data(buffer)
        global  pm1, pm25, pm10
        # 변수에 데이터 저장
        pm1 = int (data[dustlib.DUST_PM1_0_ATM])
        pm25 = int (data[dustlib.DUST_PM2_5_ATM])
        pm10 = int (data[dustlib.DUST_PM10_0_ATM])
        now = datetime.datetime.now()
        # db에 데이터 저장
        cursor.execute("INSERT INTO status(timestamp, powerstate, PM1, PM25, PM10) VALU
ES ('%s ', '%d ', '%d ','%d ','%d ')"%(now, power_state, pm1, pm25, pm10))
        dust_db.commit()
        print ("====================")
        print  ("PMS 7003 dust data")
        print  ("PM 1.0 : %d  " % (pm1))
        print  ("PM 2.5 : %d  " % (pm25))
        print  ("PM 10.0 : %d  " % (pm10))
        print ("====================")
      else :
```

```
        print  ("먼지센서 데이터 read 오류")


    # 서버통신 - 데이터 수신
    try :
        r_get_order = requests.get('http://aircleaner.software/senddata') #데이터 요청
        order = r_get_order.json()# 데이터 json타입으로 저장
        print (order)
    except  :
        print ("##################")
        print ("리퀘스트 수신 에러")
        print ("##################")

    # 수신값 받아서 설정 변경
    if  order['power_on']: # 전원 켬
        power_state = 1
    if  order['power_off']: # 전원 끔
        power_state = 0
    if  order['auto_mode']: # 자동모드
        power_state = 2
    if  order['fan_slow']: # 팬 30%
        fan_state = "SLOW"
        fan_pwm.value = 0.3
    if  order['fan_mid']: # 팬 65%
        fan_state = "MID"
        fan_pwm.value = 0.65
    if  order['fan_full']: # 팬 100%
        fan_state = "FULL"
        fan_pwm.value = 1

    conf['FAN']['fan_speed'] = fan_state # 팬 속도 저장
    with  open ('config.ini', 'w') as  configfile:
        conf.write(configfile)
    send_data = { #보낼 데이터
        'power_state':power_state, # 전원변수
        'fan_state': fan_state, # 팬속변수
        'pm1' : pm1, # 먼지농도
        'pm25' : pm25, # 먼지농도
        'pm10' : pm10 # 먼지농도
    }
    # 서버통신 - 데이터 전송
    send_data = json.dumps(send_data) # json타입으로 변경
    print (send_data)
    try :
        r_send_data = requests.post('http://aircleaner.software/receivedata', data
= send_data ) #json을 post 방식으로 전송
    except :
        print ("##################")
        print ("리퀘스트 전송 에러")
        print ("##################")
    # 전원 상태에 따른 동작
    if  power_state == 0 :
        print ("전원꺼짐")
        fan_power(OFF) #팬 off
        led.off() #led off
```

```
        lcd.LCD_BACKLIGHT = 0x 00  #lcd 백라이트 off
        lcd.lcd_string("  Power Off  ", lcd.LCD_LINE_1) # lcd 윗줄에 poweroff 표시
        lcd.lcd_string("", lcd.LCD_LINE_2)
    if  power_state == 1 :
        print ("전원켜짐")
        fan_power(ON) #팬 on
        # led.on() #led on
        lcd.LCD_BACKLIGHT = 0x 08  #lcd 백라이트 on
        lcd.lcd_string("  Power On  ", lcd.LCD_LINE_1)# lcd 윗줄에 poweron 표시
        lcd.lcd_string("", lcd.LCD_LINE_2)
        time.sleep(1 )
        display_dust(pm1, pm25, pm10) #매개변수로 데이터 전달
    if  power_state == 2 :
        print ("자동모드")
        # led.on() #led on
        lcd.LCD_BACKLIGHT = 0x 08  #lcd 백라이트 on
        lcd.lcd_string("  Auto Mode  ", lcd.LCD_LINE_1)# lcd 윗줄에 automode 표시
        lcd.lcd_string("", lcd.LCD_LINE_2)
        time.sleep(1 )
        display_dust(pm1, pm25, pm10) #매개변수로 데이터 전달
    time.sleep(1 )
```

2. app.py(서버)

```
from  flask import  Flask, render_template, jsonify, request  # Flask  서버  라이브러리
import  time  # time.sleep  사용
import  database  # 데이터베이스  분리
import  pandas as  pd  # 데이터분석용  데이터프레임  객체  생성
import  json  # 서버 - 클라이언트 - 기기  통신을  위해  json  타입과  파이썬  객체간  변환

#Flask 객체 인스턴스 생성
app = Flask(__name__)

state = { # 명령 전달 변수(dict)
 "power_on" : False ,
 "power_off" : False ,
 "auto_mode" : False ,
 "fan_slow" : False ,
 "fan_mid" : False ,
 "fan_full" : False
}

# 공기청정기  상태 변수
power_state = 0
fan_state = 0
pm1 = 0
pm25 = 0
pm10 = 0

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

```
@app.route ('/graph') # graph.html 렌더링
def  graph ():
 return  render_template('graph.html')

@app.route ('/dustinfo') # dustinfo.html 렌더링
def  dustinfo ():
 return  render_template('dustinfo.html')

@app.route ('/fan_speed') # fan_speed.html 렌더링
def  fan_speed ():
 return  render_template('fan_speed.html')

@app.route ('/info') # info.html 렌더링
def  info ():
 return  render_template('info.html')

@app.route ('/poweron') # main 전원변수 0으로
def  poweron ():
 global  state # dict 타입 명령 변수 전역변수로 지정
 print  ("poweron")
 state["power_on"] = True  # 전송용 상태 변수 True로 변경
 return  ("nothing")

@app.route ('/poweroff') # main 전원변수 1로
def  poweroff ():
 global  state # dict 타입 명령 변수 전역변수로 지정
 print  ("poweroff")
 state['power_off'] = True  # 전송용 상태 변수 True로 변경
 return  ("nothing")

@app.route ('/modeauto') # main 전원변수 2로
def  modeauto ():
 global  state # dict 타입 명령 변수 전역변수로 지정
 print  ("automode")
 state['auto_mode'] = True  # 전송용 상태 변수 True로 변경
 return  ("nothing")

@app.route ('/fanslow') # main 팬 PWM Value값 0.3으로 변경
def  fanslow ():
 global  state # dict 타입 명령 변수 전역변수로 지정
 print  ("fanslow")
 state['fan_slow'] = True  # 전송용 상태 변수 True로 변경
 return  ("nothing")

@app.route ('/fanmid') # main 팬 PWM Value값 0.65으로 변경
def  fanmid ():
 global  state # dict 타입 명령 변수 전역변수로 지정
 print  ("fanmid")
 state['fan_mid'] = True  # 전송용 상태 변수 True로 변경
 return  ("nothing")
```

```
@app.route ('/fanfull') # main 팬 PWM Value값 1로 변경
def  fanfull ():
 global  state # dict 타입 명령 변수 전역변수로 지정
 print  ("fanfull")
 state['fan_full'] = True # 전송용 상태 변수 True로 변경
 return  ("nothing")


@app.route ('/hours_graph', methods  = ['GET']) # 1시간별 데이터 반환, GET요청만 허용
def  hours_graph ():
 #database.py의 list변수를 데이터프레임으로 변환
 df = pd.DataFrame(database.oneday_hours_mean())
 df = df.set_index("hours") # index를 Hours 변수로 지정 (1~24)
 return  df.to_json() # json타입으로 변환하여 리턴

@app.route ('/days_graph', methods  = ['GET']) # 1일별 데이터 반환, GET요청만 허용
def  days_graph ():
 #database.py의 list변수를 데이터프레임으로 변환
 df = pd.DataFrame(database.oneweek_days_mean())
 df = df.set_index("days_ago")# index를 Days_ago 변수로 지정 (1~7)
 return  df.to_json() # json타입으로 변환하여 리턴

# 라즈베리 파이로부터 상태 변수 수신
@app.route ('/receivedata', methods  = ['GET', 'POST'])
def  receivedata ():
 global  data, power_state, fan_state, pm1, pm25, pm10
 data = request.get_json(force  =True  ) # json 형태로 변경
 print  (data)

 # 받은 데이터 변수에 저장
 power_state = data["power_state"]
 fan_state = data["fan_state"]
 pm1 = data["pm1"]
 pm25 = data["pm25"]
 pm10 = data["pm10"]
 # 명령변수 명령 전달 확인시 False로 변경
 if  (state["power_on"] == True  ) & (power_state == 1  ):  # power_on명령이 전달된게 확인되면
  state["power_on"] = False  #  power_on 명령변수를 False로 변경
 if  (state["power_off"] == True  ) & (power_state == 0  ):  #power_off명령이 전달된게 확인되면
  state["power_off"] = False  #  power_off 명령변수를 False로 변경
 if  (state["auto_mode"] == True  ) & (power_state == 2  ):#auto_mode명령이 전달된게 확인되면
  state["auto_mode"] = False  #  auto_mode 명령변수를 False로 변경
 if  (state["fan_slow"] == True  ) & (fan_state == "SLOW"):  # fan_slow명령이 전달된게 확인되면
  state["fan_slow"] = False  #  fan_slow 명령변수를 False로 변경
 if  (state["fan_mid"] == True  ) & (fan_state == "MID"):  #fan_mid명령이 전달된게 확인되면
  state["fan_mid"] = False  #  fan_mid 명령변수를 False로 변경
 if  (state["fan_full"] == True  ) & (fan_state == "FULL"):  #fan_full명령이 전달된게 확인되면
  state["fan_full"] = False  #  fan_full 명령변수를 False로 변경
 return  ("data received")
```

```
@app.route ('/senddata', methods  = ['GET', 'POST']) # 라즈베리파이에 명령 전달
def  senddata ():
 global  state # dict 타입 명령 변수 전역변수로 지정
 return  jsonify(state) # json 타입으로 변환하여 리턴

@app.route ('/stuff', methods  = ['GET']) # 실시간 데이터 반환, GET요청만 허용
def  stuff ():
 try :
  return  jsonify(data) # json 타입으로 변환하여 리턴
 except :
  return  ("no data")

if  __name__=="__main__": # 이 파일을 직접 실행하면
 app.run(host ="0.0.0.0", debug =True ) # 현재 ip에서 서버 호스트
```

3. database.py(서버-데이터베이스 참조)

```
import  pymysql # mysql 데이터베이스
import  datetime # 날짜와 시간을 사용하기 위해
import  pandas as  pd # 데이터분석 라이브러리 데이터프레임 객체 생성


# DB 연결, DB참조 오류로 인해 각각 지정
dust_db = pymysql.connect(
  user ='luvdduk',
  passwd ='***',
  host ='aircleaner.software',
  db ='air-cleaner',
  charset ='utf8'
)
dust_db2 = pymysql.connect(
  user ='luvdduk',
  passwd ='***',
  host ='aircleaner.software',
  db ='air-cleaner',
  charset ='utf8'
)


def  oneday_hours_mean ():
  cursor1 = dust_db.cursor(pymysql.cursors.DictCursor) # DB커서 지정
  NowDate = datetime.datetime.now() # 현재시간
  left24h = NowDate - datetime.timedelta(days =1 ) # 24시간전
  cursor1.execute("SELECT * FROM status WHERE TIMESTAMP(timestamp) BETWEEN '%s
' AND '%s ';"%(left24h, NowDate)) # DB 데이터 선택
  df = cursor1.fetchall() # 변수에 데이터 저장
  df = pd.DataFrame(df) # 데이터프레임으로 변경
  hour = 1  # 1시간전부터 시작
  hours_mean = list () # 반환타입 리스트 변수로 지정
  while  hour <= 24 : # 24번 반복 24시간 전의 데이터까지 저장
    if  hour > 0 :  # 오류방지용 0일때는 실행안함
      start_time = datetime.datetime.now() - datetime.timedelta(hours  =hour) # 시작시간 부
터
      end_time = start_time + datetime.timedelta(hours =1 ) # 끝나는시간까지
      one_hour = (df['timestamp']>= start_time) & (df['timestamp']< end_time) # 데이터 추출
```

```
        df_1h = df.loc[one_hour] # 지정된 범위의 데이터(1시간)만 df_1h에 저장
            mean_1h = { "hours" : hour, "time": start_time.strftime("%Y-%m-%d %H:%M:%S"), "pm1": df_1h['PM1'].mean(), "pm25": df_1h['PM25'].mean(), "pm10": df_1h['PM10'].mean()} # 데이터별 평균을 구해서, dict형식으로 저장
        hours_mean.append(mean_1h) # dict변수 mean1_h를 list변수 hours_mean에 추가
        hour += 1
    return  hours_mean # list변수 리턴


def  oneweek_days_mean ():
    cursor = dust_db2.cursor(pymysql.cursors.DictCursor) # DB커서 지정
    NowDate = datetime.datetime.now() # 현재시간
    left7d = NowDate - datetime.timedelta(days =8 ) # 24시간전
    cursor.execute("SELECT * FROM status WHERE TIMESTAMP(timestamp) BETWEEN '%s' AND '%s ';"%(left7d, NowDate)) # DB 데이터 선택
    df = cursor.fetchall() # 변수에 데이터 저장
    df = pd.DataFrame(df) # 데이터프레임으로 변경
    days = 0 # 0일전, 오늘부터 시작
    days_mean = list () # 리스트 변수 선언
    while  days <= 7 : # 7번, 7일간의 데이터 저장
        day = datetime.date.today() - datetime.timedelta(days =days) # 평균 낼 날자 지정
        day = datetime.datetime.combine(day, datetime.time()) # datetime -> date 형식으로 변경
        one_day = (df['timestamp']>= day) & (df['timestamp']< (day + datetime.timedelta(hours =24 ))) # 1일동안으로 범위 지정
        df_1d = df.loc[one_day] # 지정된 범위의 데이터(1일)만 df_1d에 저장
            mean_1d = {"days_ago": days, "date":day.strftime("%Y-%m-%d"), "pm1": df_1d['PM1'].mean(), "pm25": df_1d['PM25'].mean(), "pm10": df_1d['PM10'].mean()} # 데이터별 평균을 구해서, dict형식으로 저장
        days_mean.append(mean_1d)# dict변수 mean_1d를 list변수 days_mean에 추가
        days += 1
    return  days_mean # list변수 리턴
```

4. index.html(메인 페이지)

```
<!DOCTYPE  html >
<html  lang ="ko">
<head>
 <meta  charset ="UTF-8">
 <meta  name ="viewport" content ="width=device-width, initial-scale=1.0">
 <title>공기청정기 정보 </title><!-- 제목 -->
 <link  rel ="stylesheet" href ="{{url_for('static',filename='index.css')}}"><!-- 전체 css -->
 <script  src ="https://kit.fontawesome.com/e08131f43f.js" crossorigin  ="anonymous"></script><!-- ccs 폰트 -->
</head>
<body  onload ="update_dust ();"><!-- body가 로드될 때 데이터 수신을 위해 함수 실행 -->
 <div  class ="content">
  <nav  role ="navigation"><!-- 상단 타이틀 및 좌측 네비게이션 바 -->
   <a  href  ="{{ url_for('index') }}" class  ="logo"><div  class  ="logo-box"><h1>공기청정기</h1></div></a>
   <div  id ="menuToggle"><!-- 네비 바 -->
    <input  type ="checkbox" />
     <span></span>
     <span></span>
     <span></span>
    <ul  id ="menu">
```

```
      <li><a  href ="{{ url_for('index') }}">정보 </a></li>
      <li><a  href ="{{ url_for('graph') }}">그래프 </a></li>
      <li><a  href ="/static/app/Air_Cleaner_app-스마트공기청정기.apk">앱 다운로드 </a></li>
      <li><a  href ="{{ url_for('info') }}">작품 소개 및 발표자료 </a></li>
     </ul>
    </div>
   </nav>
  </div>
  <main>
   <div  class ="sw">
    <div  id ="swtext">전원 </div><!-- 전원 스위치 -->
    <input  type ="checkbox" id ="switch1" name ="switch1" class ="input__on-off" onclick
="powerctrl ()">
    <label  for ="switch1" class ="label__on-off">
     <span  class ="marble"></span>
     <span  class ="on">on </span>
     <span  class ="off">off </span>
    </label>
   </div>

   <a  href ="{{ url_for('dustinfo') }}">
   <div  class ="dust">
    <div  id ="dustname"> 현재 미세먼지 농도 </div><!-- 상태별 먼지농도 이모티콘 표시 -->
    <i  class ="far fa-smile" id ="good"></i>
    <i  class ="far fa-meh" id ="normal"></i>
    <i  class ="far fa-frown-open" id ="bad"></i>
    <i  class ="far fa-tired" id ="verybad"></i>
        <!-- js에서 각 이모티콘들 숨김/표시 제어 -->
    <div  id ="dustvalue"> <span  id ="duststate">알수없음 </span></div>
   </div>
   </a>
   <div  class ="sw"><!-- 자동모드 스위치 -->
    <div  id ="swtext">자동모드 </div>
    <input  type ="checkbox" id ="switch2" name ="switch2" class ="input__on-off" onclick
="autoctrl ()">
    <label  for ="switch2" class ="label__on-off" id ="sw2lb">
     <span  class ="marble"></span>
     <span  class ="on">on </span>
     <span  class ="off">off </span>
    </label>
   </div>
   <div  class ="sw" onclick ="location .href ='/fan_speed'"><!-- 풍량조절 스위치 -->
    <div  id ="swtext">바람세기 </div>
    <div  style ="display: inline-block;" id ="speedsel"><span  id ="fan_speed">알수없음
</span><i  class ="fas fa-chevron-right" style
="font-size: 23pt; margin-left: 15px;"></i></div>
   </div>
  </main>
  <script  src ="{{url_for('static', filename = 'jquery.min.js')}}"></script>
  <script  type =text/javascript  src ="{{url_for('static', filename = 'index.js')}}"></script>
</body>
</html>
```

5. graph.html(그래프 표시 페이지)

```
<!DOCTYPE html >
<html lang ="ko">
<head>
 <meta charset ="UTF-8">
 <meta name ="viewport" content ="width=device-width, initial-scale=1.0">
 <title>공기청정기 그래프 </title><!-- ccs 폰트 -->
 <link rel ="stylesheet" href ="{{url_for('static',filename='index.css')}}"><!-- ccs 폰트 -->
</head>
<body>
 <div class ="content">
  <nav role ="navigation"> <!-- 상단 타이틀 및 좌측 네비게이션 바 -->
   <a href ="{{ url_for('index') }}" class ="logo"><div class ="logo-box"><h1>공기청정기
</h1></div></a>
   <div id ="menuToggle"><!-- 네비 바 -->
    <input type ="checkbox" />
     <span></span>
     <span></span>
     <span></span>
    <ul id ="menu">
     <li><a href ="{{ url_for('index') }}">정보 </a></li>
     <li><a href ="{{ url_for('graph') }}">그래프 </a></li>
     <li><a href ="/static/app/Air_Cleaner_app-스마트공기청정기.apk">앱 다운로드 </a></li>
     <li><a href ="{{ url_for('info') }}">작품 소개 및 발표자료 </a></li>
    </ul>
   </div>
  </nav>
 </div>
 <main class ="main-content"><!-- canvas태그로 그래프 위치 지정, div로 여백 지정 -->
  <div style ="margin: 15px 15px;"><canvas id ="hours_graph" width ="300" height
="300"></canvas></div>
  <div style ="margin: 15px 15px;"><canvas id ="days_graph" width ="300" height
="300"></canvas></div>
 </main>
 <script src ="{{url_for('static', filename = 'jquery.min.js')}}"></script>
 <script src ="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js" integrity
="sha512-d9xgZrVZpmmQlfonhQUvTR7lMPtO7NkZMkA0ABN3PHCbKA5nqylQ/yWlFAyY6hYg
dF1Qh6nYiuADWwKB4C2WSw==" crossorigin ="anonymous"></script>
 <script>
  // 24시간, 시간별 그래프
  $.ajax ({ //js로드시 실행
  method : "GET",
  url : '/hours_graph',
  dataType: "json",

  success : function (data ){ //ajax 통신이 되면 그래프 그리기
   new Chart (document .getElementById ("hours_graph"), {
    type: 'line',
    data: { labels : [1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20
,21 ,22 ,23 ,24 ],
          datasets : [{
           label: 'PM1.0',
           data: Object .values (data .pm1 ),
           fill : false ,
           borderColor : 'rgba(1, 197, 196, 0.5)',
```

```
            backgroundColor : 'rgba(1, 197, 196, 0.5)'
           },{
            label: 'PM2.5',
            data: Object .values (data .pm25 ),
            fill : false ,
            borderColor : 'rgba(184, 222, 111, 0.5)',
            backgroundColor : 'rgba(184, 222, 111, 0.5)'
           },{
            label: 'PM10',
            data: Object .values (data .pm10 ),
            fill : false ,
            borderColor : 'rgba(243, 146, 51, 0.5)',
            backgroundColor : 'rgba(243, 146, 51, 0.5)'
           }]
      },
      options : {
       title : { //차트 제목
        display : true ,
        text   : "최근 24시간",
        fontSize : 20  },
       scales: {
        xAxes: [{ //x축 제목
         scaleLabel: {
          display: true ,
          labelString: '시간전',
          fontSize : 15
         }
        }],
        yAxes: [{ //y축 제목
         scaleLabel: {
          display: true ,
          labelString: 'μg/m³',
          fontSize : 15
         }
        }]
      }
      } } ); },

   error : function (error_data ){
    alert ("공기청정기와의 연결이 원활하지 않습니다. 새로고침하거나 공기청정기의 전원플러그를 재연결
후 시도해주세요.")
    console .log ("ajax error")
    console .log (error_data ) }
 } )
  //일주일, 일별그래프
  $.ajax ({ //js로드시 실행
  method : "GET",
  url   : '/days_graph',
  dataType: "json",

  success : function (data ){ //ajax 통신이 되면 그래프 그리기
   new  Chart (document .getElementById ("days_graph"), {
    type: 'line',
    data: { labels  : ["오늘",1 ,2 ,3 ,4 ,5 ,6 ,7 ],
         datasets : [{
```

```
            label: 'PM1.0',
            data: Object .values (data .pm1 ),
            fill : false ,
            borderColor : 'rgba(1, 197, 196, 0.5)',
            backgroundColor : 'rgba(1, 197, 196, 0.5)'
           },{
            label: 'PM2.5',
            data: Object .values (data .pm25 ),
            fill : false ,
            borderColor : 'rgba(184, 222, 111, 0.5)',
            backgroundColor : 'rgba(184, 222, 111, 0.5)'
           },{
            label: 'PM10',
            data: Object .values (data .pm10 ),
            fill : false ,
            borderColor : 'rgba(243, 146, 51, 0.5)',
            backgroundColor : 'rgba(243, 146, 51, 0.5)'
           }]
      },
      options : {
       title : { //차트 제목
        display : true ,
        text   : "최근 일주일",
        fontSize : 20
       },
        scales: {
         xAxes: [{ //x축 제목
          scaleLabel: {
           display: true ,
           labelString: '일전',
           fontSize : 15
          }
         }],
         yAxes: [{ //y축 제목
          scaleLabel: {
           display: true ,
           labelString: 'μg/m3',
           fontSize : 15
          }
         }]
        }
      } } ); },

    error : function (error_data ){
     alert ("공기청정기와의 연결이 원활하지 않습니다. 새로고침하거나 공기청정기의 전원플러그를 재연결
후 시도해주세요.")
     console .log ("ajax error")
     console .log (error_data ) }
  } )
   </script>
</body>
</html>
```

6. fan_speed.html(팬 속도 조절 페이지)

```
<!DOCTYPE html >
<html lang ="ko">
<head>
  <meta charset ="UTF-8">
  <meta name ="viewport" content ="width=device-width, initial-scale=1.0">
  <title>팬 속도 제어 </title><!-- 제목 -->
  <link rel ="stylesheet" href ="{{url_for('static',filename='index.css')}}"><!-- 전체 ccs -->
  <script src ="https://kit.fontawesome.com/e08131f43f.js" crossorigin ="anonymous">
</script><!-- ccs 폰트 -->
</head>
<body>
 <div class ="content">
  <nav role ="navigation"><!-- 상단 타이틀 바 -->
   <i class ="fas fa-chevron-left" id ="left" onclick ="history .back ()"></i>
   <a href ="{{ url_for('index') }}" class ="logo"><div class ="logo-box" id ="logo2">공기청
정기 </div></a><!-- 로고-->
  </nav>
 </div>
 <main class ="main-content"><!-- 팬 속도 조정 리스트 -->
  <div class ="sw fanspeed" id ="slow" onclick ="fanslow ()">1단계(30%)</div>
  <div class ="sw fanspeed" id ="mid" onclick ="fanmid ()">2단계(65%)</div>
  <div class ="sw fanspeed" id ="full" onclick ="fanfull ()">3단계(100%)</div>
 </main>
 <script src ="{{url_for('static', filename = 'jquery.min.js')}}"></script>
 <script type =text/javascript src ="{{url_for('static', filename = 'index.js')}}"></script>
</body>
</html>
```

7. dustinfo.html(상세 먼지농도 표시 페이지)

```
<!DOCTYPE html >
<html lang ="ko">
<head>
  <meta charset ="UTF-8">
  <meta name ="viewport" content ="width=device-width, initial-scale=1.0">
  <title>미세먼지 상세정보 </title><!-- 제목 -->
  <link rel ="stylesheet" href ="{{url_for('static',filename='index.css')}}"><!-- 메인 css -->
  <script ipt src ="https://kit.fontawesome.com/e08131f43f.js" crossorigin ="anonymou
s"></script><!-- ccs 폰트 -->
</head>
<body>
 <div class ="content">
  <nav role ="navigation" id ="navid">
   <i class ="fas fa-chevron-left" id ="left" onclick ="history .back ()"></i>
   <a href ="{{ url_for('index') }}" class ="logo"><div class ="logo-box" id ="logo2">공기청
정기 </div></a><!-- 로고 -->
  </nav>
 </div>
 <main class ="main-content"><!-- 실시간으로 먼지 데이터 업데이트 -->
  <a href ="https://www.me.go.kr/mamo/web/index.do?menuId=16201">
   <div class ="sw fanspeed">PM1.0 농도: <span id ="dustvalue1"></span>㎍/m³</div>
   <div class ="sw fanspeed">PM2.5 농도: <span id ="dustvalue2"></span>㎍/m³</div>
   <div class ="sw fanspeed">PM10 농도: <span id ="dustvalue3"></span>㎍/m³</div>
  </a>
```

```
 </main>
 <script  src ="{{url_for('static', filename = 'jquery.min.js')}}"></script>
 <script  type =text/javascript  src ="{{url_for('static', filename = 'index.js')}}"></script>
</body>
</html>
```

8. info.html(작품 소개 페이지)

```
<!DOCTYPE  html >
<html  lang ="ko">
  <head>
    <meta  charset ="UTF-8">
    <meta  name ="viewport" content ="width=device-width, initial-scale=1.0">
    <title>졸업작품 소개 </title><!-- 제목 -->
    <link  rel ="stylesheet" href ="{{url_for('static',filename='index.css')}}"><!-- 전체 css -->
    <link  rel ="stylesheet" href ="{{url_for('static',filename='info.css')}}"><!-- 버튼 css -->
    <script  src ="https://kit.fontawesome.com/e08131f43f.js" crossorigin ="anonymous">
</script><!-- ccs 폰트 -->
  </head>
  <body>
    <div  class ="content">
      <nav  role ="navigation"><!-- 상단 타이틀 및 좌측 네비게이션 바 -->
      <a  href ="{{ url_for('index') }}" class ="logo"><div  class ="logo-box"><h1>공기청정
기 </h1></div></a><!-- 로고 -->
      <div  id ="menuToggle"><!-- 네비 바-->
        <input  type ="checkbox" />
        <span></span>
        <span></span>
        <span></span>
        <ul  id ="menu">
          <li><a  href ="{{ url_for('index') }}">정보 </a></li>
          <li><a  href ="{{ url_for('graph') }}">그래프 </a></li>
          <li><a  href ="/static/app/Air_Cleaner_app-스마트공기청정기.apk">앱 다운로드
</a></li>
          <li><a  href ="{{ url_for('info') }}">작품 소개 및 발표자료 </a></li>
        </ul>
      </div>
      </nav>
    </div>

    <main  class ="main-content" style ="margin-left: 20px; margin-right: 20px; padding-t
op: 10px;"><!-- 메인 콘텐츠 -->
      <div  style ="text-align: center;">
        <img  src ="/static/pic/동미대_마크.png" width ="100%" style ="margin-bottom: 10
px;" alt =""><!-- 학교로고 -->
      </div>
      <h2>조원 </h2>
      <div  style ="font-size: 20px;">이름: 진미강, 학번: 20184902 </div>
      <div  style ="font-size: 20px;">이름: 정택수, 학번: 20173740 </div>
      <h2>작품 소개 </h2>
      <div  style ="margin: 10px 0px; font-size: 12pt; text-align: justify;">  기존 공기청정기가
 가진 기능인 스위치를 이용해 공기청정기의 동작을 제어하고, 미세먼지 센서의 측정값을 LCD로 표시해주
는 기능과, 이에 더해 애플리케이션을 통해 실시간 미세먼지 값을 모니터링하고, 최근 24시간, 7주일 단위로
미세먼지 변화를 알 수 있는 그래프를 확인할 수 있는 기능과 가정 또는 외부에서도 공기청정기의 동작을 제
어할수 있는 IOT기능, 그리고 먼지센서 농도에 따라 팬 속도를 제어하는 자동모드 기능을 라즈베리파이를 이
```

```
용해 파이썬으로 구현하였습니다. 애플리케이션은 파이썬 웹서버(flask)와 연동하여, HTML, CSS, JS를 이용해 웹과 서버 간 통신 및 DB데이터 처리를 구현하였습니다 </div>
        <iframe width="100%" height="200px" src="https://www.youtube.com/embed/ceI64_MZTFY" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen ></iframe>
        <a href="/static/pdf,ppt/졸업작품 3차 발표-최종.pptx" style="text-align: center;"><button id="bt1"><i class="far fa-file-powerpoint" style="margin-right: 10px;"></i> 3차 발표자료 PPT </button></a><!-- ppt 다운로드 버튼 -->
        <a href="/static/pdf,ppt/졸업작품_최종_보고서.pdf" style="text-align: center;"><button id="bt2"><i class="far fa-file-pdf" style="margin-right: 10px;"></i> 최종 보고서 PDF </button></a><!-- pdf 다운로드 버튼 -->
      </main>
    </body>
</html>
```

9. index.js(메인, 먼지농도, 팬 속도조절 관련 자바스크립트)

```
let pm1 ;
let pm25 ;
let pm10 ;
let power_state ;
let fan_speed ;
//각각의 경로에 GET요청을 보내 경로에 지정된 함수를 실행
function poweron (){ // 전원 켜기
  $.getJSON ('/poweron',
        function (data ) {
      }); // /poweron 경로에 GET요청
      return false ;
}
function poweroff (){ //전원 끄기
  $.getJSON ('/poweroff',
        function (data ) {
      }); // /poweroff 경로에 GET요청
      return false ;
}
function modeauto (){ // 자동모드
  $.getJSON ('/modeauto',
        function (data ) {
      }); // /modeauto 경로에 GET요청
      return false ;
}
function fanslow (){ //팬 속도 65%
  $.getJSON ('/fanslow',
        function (data ) {
      }); // /fanmid 경로에 GET요청
   return false ;
}
function fanmid (){ //팬 속도 65%
  $.getJSON ('/fanmid',
        function (data ) {
      }); // /fanmid 경로에 GET요청
   return false ;
}
function fanfull (){ //팬 속도 100%
```

```
  $.getJSON ('/fanfull',
       function (data ) {
    }); // /fanfull 경로에 GET요청
  return  false ;
}


let  intervalID  = setInterval (update_dust , 1000 ); //1초마다 update_dust() 함수 반복 실행
function  update_dust () {
  $.getJSON ('/stuff', //ajax로 실시간 미세먼지값 수신
  function  (data ) {
  console .log (data );
  //변수에 실시간 데이터값 저장(int)
  pm1  = Number (data .pm1 );
  pm25  = Number (data .pm25 );
  pm10  = Number (data .pm10 );
  power_state  = Number (data .power_state );
  fan_state  = data .fan_state ;
  // 실시간 미세먼지 값을 해당 id 태그에 업데이트
  $('#dustvalue1').text (pm1 );
  $('#dustvalue2').text (pm25 );
  $('#dustvalue3').text (pm10 );

  if  (fan_state  == "SLOW"){//팬 속도가 1단계일때
    $('#fan_speed').text ("1단계"); //팬 스위치에 상태 표시
  }
  else  if (fan_state  == "MID"){//팬 속도가 2단계일때
    $('#fan_speed').text ("2단계"); //팬 스위치에 상태 표시
  }
  else  if (fan_state  == "FULL"){//팬 속도가 3단계일때
    $('#fan_speed').text ("3단계"); //팬 스위치에 상태 표시
  }
  else {
    $('#fan_speed').text ("Error");
    console .log ("풍량표시 오류")
  }

  //미세먼지 이모티콘 및 색 변화
  if ((pm10  <= 30 ) && ((pm25  + pm1 ) <= 15 )){ //좋음
    $("#duststate").css ("color", "#a3e7d6");
    $('#duststate').text ("좋음");
    $("#good").css ("display", "block"); //좋음 이모티콘만 표시
    $("#normal").css ("display", "none"); //나머지 숨김
    $("#bad").css ("display", "none"); //나머지 숨김
    $("#verybad").css ("display", "none"); //나머지 숨김
    console .log ("좋음")
  }else  if ((pm10  <= 80 ) && ((pm25  + pm1 ) <= 35 )){ //보통
    $("#duststate").css ("color", "#77dd77");
    $('#duststate').text ("보통");
    $("#good").css ("display", "none"); //나머지 숨김
    $("#normal").css ("display", "block"); //보통 이모티콘만 표시
    $("#bad").css ("display", "none"); //나머지 숨김
    $("#verybad").css ("display", "none"); //나머지 숨김
    console .log ("보통")
  }else  if ((pm10  <= 150 ) && ((pm25  + pm1 ) <= 75 )){ //나쁨
```

```
        $("#duststate").css ("color", "#fd9a18");
        $('#duststate').text ("나쁨");
        $("#good").css ("display", "none"); //나머지 숨김
        $("#normal").css ("display", "none"); //나머지 숨김
        $("#bad").css ("display", "block"); //나쁨 이모티콘만 표시
        $("#verybad").css ("display", "none"); //나머지 숨김
    }else if ((pm10 >150 ) || ((pm25 + pm1 ) > 75 )){ //매우나쁨
        $("#duststate").css ("color", "red");
        $('#duststate').text ("매우나쁨");
        $("#good").css ("display", "none"); //나머지 숨김
        $("#normal").css ("display", "none"); //나머지 숨김
        $("#bad").css ("display", "none"); //나머지 숨김
        $("#verybad").css ("display", "block"); //매우나쁨 이모티콘만 표시
    }else {
        console .log ("미세먼지 상태 표시 오류")
        $('#duststate').text ("Error!");
    }
    if (power_state == 0 ){ //전원이 꺼져있을때
        $("#switch1").prop ("checked", false ); //전원스위치 체크해제(Off)
        $('#switch2').attr ("disabled", true ); // 자동모드 스위치 비활성화
        $("#sw2lb").css ("background-color", "#bebebe"); // 자동모드 스위치 회색으로 설정
    }else { // 켜져있거나 자동모드일때
        $("#switch1").prop ("checked", true ); //전원스위치 체크(On)
        $('#switch2').removeAttr ("disabled"); // 비활성화 해제
    }
    if (power_state == 2 ){ // 자동모드일때
        $("#switch2").prop ("checked", true ); //자동모드 스위치 체크
        $("#sw2lb").css ("background-color", "#0bba82"); // 자동모드 스위치 초록색으로
    }else if (power_state ==1 ){
        $("#switch2").prop ("checked", false );//자동모드 스위치 체크해제
        $("#sw2lb").css ("background-color", "#ed4956");// 자동모드 스위치 빨간색으로
    }else { //자동모드가 아닐때
        $("#switch2").prop ("checked", false );
        $("#sw2lb").css ("background-color", "#bebebe");// 자동모드 스위치 회색으로
    }
    });

};
function powerctrl (){ // 파워스위치 누를시 실행, 파워모드 변경
    console .log ("파워조정");
    if (power_state == 0 ){ //전원이 꺼져있을때 누르면
        poweron (); //전원켬
    }
    else if (power_state == 1 ){ //전원이 켜져있을때 누르면
        poweroff (); //전원끔
    }
    else if (power_state == 2 ){ //자동모드 일때 누르면
        poweroff (); //전원끔
    }
    else { //변수 오류 방지
        console .log ("전원 변수 오류")
    }
};
function autoctrl (){ // 자동모드 스위치 누를시 실행, 자동모드 설정 및 해제
    console .log ("모드조정");
```

```
  if (power_state == 1 ){ //전원이 켜져있을때 누르면
    modeauto (); //자동모드로 설정
  }
  else if (power_state == 2 ){ //자동모드 일때 누르면
    poweron (); //자동모드 끔(전원켬 상태로 변경)
  }
  else {
    console .log ("전원 변수 오류");
  }
};
```

10. index.css (html 요소 배치 및 디자인)

```
a {
 text-decoration : none ;
}
body {
 background-color : #f9f9f9 ;
 margin : 0 ;
 padding : 0 ;
}
#menu a {
 text-decoration : none ;
 color : #1E1E23 ;
 opacity :1 ;
 font-family : 'work sans', sans serif ;
 font-size : 1.5em ;
 font-weight : 400 ;
 transition : 200ms ;
}
#menu a:hover {
 opacity :0.5 ;
}
ul {
 padding : 0 ;
 list-style-type : none ;
}
/*네비 바*/
nav {
 background-color : #95c6e6 ;
 height : 65px ;
 width : 100%;
 position : fixed ;
 margin-top : -65px ;
 z-index : 10 ;
}
#menuToggle {
 display : inline-block ;
 flex-direction : column ;
 position : relative ;
 top : 25px ;
 left : 25px ;
 z-index : 1 ;
 -webkit-user-select : none ;
 user-select : none ;
 width : auto ;
```

```
}
#menuToggle input
{
 display : flex ;
 width : 40px ;
 height : 32px ;
 position : absolute ;
 cursor : pointer ;
 opacity : 0 ;
 z-index : 2 ;
}
#menuToggle span
{
 display : flex ;
 width : 29px ;
 height : 2px ;
 margin-bottom : 5px ;
 position : relative ;
 background : #ffffff ;
 border-radius : 3px ;
 z-index : 1 ;
 transform-origin : 5px 0px ;
 transition : transform 0.5s cubic-bezier (0.77 ,0.2 ,0.05 ,1.0 ),
        background 0.5s cubic-bezier (0.77 ,0.2 ,0.05 ,1.0 ),
        opacity 0.55s ease ;
}
#menuToggle span:first-child
{
 transform-origin : 0% 0%;
}
#menuToggle span:nth-last-child (2 )
{
 transform-origin : 0% 100%;
}
#menuToggle input:checked ~ span
{
 opacity : 1 ;
 transform : rotate (45deg ) translate (-3px , -1px );
 background : #36383F ;
}
#menuToggle input:checked ~ span:nth-last-child (3 )
{
 opacity : 0 ;
 transform : rotate (0deg ) scale (0.2 , 0.2 );
}
#menuToggle input:checked ~ span:nth-last-child (2 )
{
 transform : rotate (-45deg ) translate (0 , -1px );
}
#menu
{
 position : absolute ;
 width : 180px ;
 height : 100vh ;
 box-shadow : 0 0 10px #85888C ;
```

```
 margin : -50px  0  0  -50px ;
 padding : 50px ;
 padding-top : 125px ;
 background-color : #F5F6FA ;
 -webkit-font-smoothing : antialiased ;
 transform-origin : 0% 0%;
 transform : translate (-100%, 0 );
 transition : transform 0.5s  cubic-bezier (0.77 ,0.2 ,0.05 ,1.0 );
}
#menu  li
{
 padding : 10px  0 ;
 transition-delay : 2s ;
}
#menuToggle  input:checked  ~ ul
{
 transform : none ;
}
/*네비 바 end*/
/*로고*/
.logo {
 color : white ;
 display : inline ;
}
.logo-box {
 position : absolute ;
 display : flex ;
 flex-direction : column ;
 left : 50%;
 margin : -5px  -60px  0 ;
 width : auto ;
 height : auto ;
}
/*로고 end*/
div.qw {
 width : 300px ;
 height : 300px ;
 background-color : aqua ;
 margin : 2px  0 ;
 border : 2px  solid  black ;
 border-radius : 5px ;
}
/*메인컨텐츠 여백*/
main {
 margin-top : 65px ;
}


.dust {
 width : 100%;
 height : 180px ;
 color : black ;
 text-align : center ;
 transition : color  1s ;
}
```

```
/*먼지 데이터*/
#dustvalue {
 height : auto ;
 display : inline-block ;
 font-size : 35pt ;
 -webkit-transition : all .3s ;
 -moz-transition : all .3s ;
 -ms-transition : all .3s ;
 -o-transition : all .3s ;
 transition : all .3s ;
}
#powerbutton {
 text-align : center ;
}
/*스위치*/
.sw {
 display : flex ;
 padding-top : 15px ;
 margin-left : 10vw ;
 margin-right : 10vw ;
 text-align : center ;
 border-bottom : 1px solid #85888c9c ;
 -webkit-justify-content : space-between ;
 justify-content : space-between ;
}
#swtext {
 font-size : 20pt ;
 float : left ;

}


#dustname {
 font-size : 15pt ;
 font-weight : 600 ;
}
/* 파워스위치 */
input [type ="checkbox"] {

 display : none ;
}
.label__on-off {
  overflow : hidden ;
  position : relative ;
  display : inline-block ;
  width : 70px ;
  height : 30px ;
  -webkit-border-radius : 13px ;
  -moz-border-radius : 13px ;
  border-radius : 30px ;
  background-color : #ed4956 ;
  color : #fff ;
  font-weight : bold ;
  cursor : pointer ;
  -webkit-transition : all .3s ;
```

```
    -moz-transition : all .3s ;
    -ms-transition : all .3s ;
    -o-transition : all .3s ;
    transition : all .3s ;
    /* left: 35%; */
    /* float: right; */
    text-align : justify ;
}
.label__on-off > * {
    vertical-align : sub ;
    -webkit-transition : all .3s ;
    -moz-transition : all .3s ;
    -ms-transition : all .3s ;
    -o-transition : all .3s ;
    transition : all .3s ;
    font-size : 14px ;
}
.label__on-off .marble {
    position : absolute ;
    top : 1px ;
    left : 1px ;
    display : block ;
    width : 28px ;
    height : 28px ;
    background-color : #fff ;
    -webkit-border-radius : 50%;
    -moz-border-radius : 50%;
    border-radius : 50%;
    -webkit-box-shadow : 0 0 10px rgba (0 , 0 , 0 , .3 );
    -moz-box-shadow : 0 0 10px rgba (0 , 0 , 0 , .3 );
    box-shadow : 0 0 10px rgba (0 , 0 , 0 , .3 );
}
.label__on-off .on {
    display : none ;
    padding-left : 15px ;

}
.label__on-off .off {
    padding-left : 40px ;
    line-height : 25px ;
}
.input__on-off:checked + .label__on-off {
    background-color : #0bba82 ;
}
.input__on-off:checked + .label__on-off .on {
    display : inline-block ;
}
.input__on-off:checked + .label__on-off .off {
    display : none ;
}
.input__on-off:checked + .label__on-off .marble {
    left : 40px ;
}
/*스위치 end*/
```

```
/*이모티콘*/
.fa-smile {
 display : none ;
 font-size : 30pt ;
 color : #a3e7d6 ;

}
.fa-meh {
 display : none ;
 font-size : 30pt ;
 color : #77dd77 ;

}
.fa-frown-open {
 display : none ;
 font-size : 30pt ;
 color : #fd9a18 ;
}
.fa-tired {
 display : none ;
 font-size : 30pt ;
 color : red ;
}
/*이모티콘end*/

.fanspeed {
 font-size : 20pt ;
}
#fan_speed {

 font-size : 20pt ;
}
#left {
 font-size : 27pt ;
 color : white ;
 padding : 12px  20px ;
 left : 10%;
}
h1 {
 display : inline ;
}
#navid {
 text-align : justify ;
}
#logo2 {
 font-size : 18pt ;
 font-weight : 600 ;
 display : inline ;
 margin : 10px  -60px  0 ;
}
```

VII. 작품 활용 방안

1. 여가 학습 활동 공간 및 집안 내부에서 사용하여 공기를 정화하여 더 나은 환경을 제공합니다.
2. 쉽게 접할 수 있는 미세먼지 수치에 대한 정보는 시·구 규모를 기준으로 한 수치가 보통이기 때문에 이 공기청정기를 사용하여 근접한, 주변환경에 대한 미세먼지 수치를 알 수 있으며, 앱으로 실시간 미세먼지 농도, 시간별·월별 미세먼지 변화 그래프를 확인할 수 있습니다.
3. 직접 동작시키기 어려운 상황이거나 거동이 불편하신분 혹은 수동 조작에 제한이 되시는 분들이 앱을 통해 제어하거나, 구글 어시스턴트를 이용하여 음성으로 공기청정기의 동작을 제어할 수 있습니다.

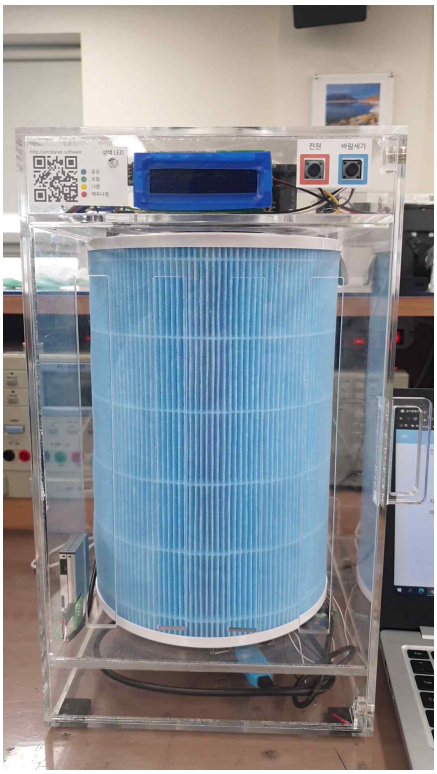
VIII. 졸업 작품 후 소감

졸업작품을 기회로 라즈베리파이로 웹서버를 만들며 많은 공부를 하게 되었습니다. 졸업작품을 진행하기 전부터 파이썬 언어를 접할 수 있는 기회가 있었기에 이를 응용하고, 새로운 것을 시도해보기 위해 Flask 웹서버, html, css, javascript를 날이 새 가며 찾아보고, 공부했습니다. 물론 항상 수월히 진행되지만은 않고 고작 코드 10줄 쓰는데 일주일이 걸린 적도 있었습니다. 하지만 그 과정에서 해결 방법을 찾아내려 노력하고, 해결해서 원하는 대로 동작했을 때의 기쁨과 그 성취감은 이루 말할 수 없이 좋았습니다. 또, 모르는 부분은 서로 물어보고, 도와주며 서로의 부족한 부분을 채워나갔고 서로 맡은 부분, 하드웨어와 소프트웨어에서 각자 최선을 다해 작품을 만들었습니다. 졸업작품을 마무리한 후엔 조금 더 성장한 자신을 볼 수 있어 뿌듯했습니다.

IX. 총 제작비

부품명	수량	단가	가격
Raspberry Pi 4B+	1	75,000	75,000
점퍼선 40개 묶음	3	3,000	9,000
아크릴 재단 및 시공비용	1	100,000	100,000
Raspberry Pi 4 Heat sink case with fan	1	11,500	11,500
EVERCOOL EC12038H12SA-3P (시스템 팬)	1	8,000	8,000
CSPCSZ-AL-M4-16 (M4 * 16mm 볼트)	10	204	2,040
HNT1-316L-M4 (M4 너트)	10	340	3,400
PCB 기판	1	1,200	1,200
RGB LED Module	1	780	780
I2C LCD 1602 Module	1	2,400	2,400
3.3V to 5V Logic Converter	1	500	500
PMS-7003 Dust Sensor	1	25,000	25,000
Tact Switch	2	100	200
Ugreen Type C 연장케이블 0.5m	1	10,330	10,330
합계	-	-	249,350

X. 졸업작품 사진



<정면>

<후면>

<좌측>

<우측>