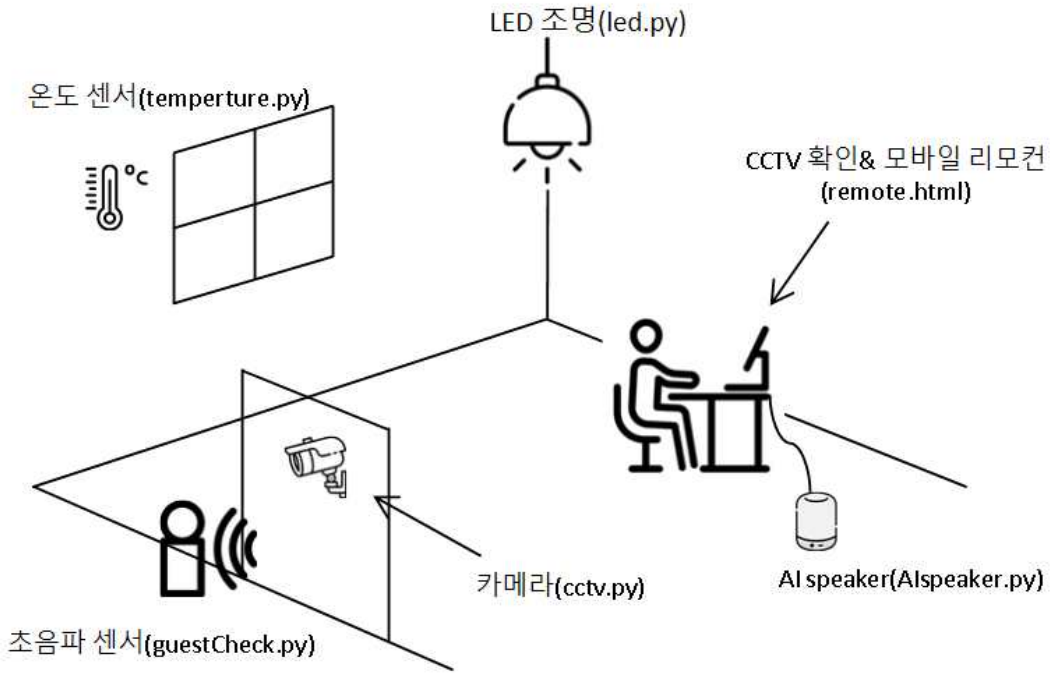
라즈베리파이를 이용한 스마트 홈 관리 시스템

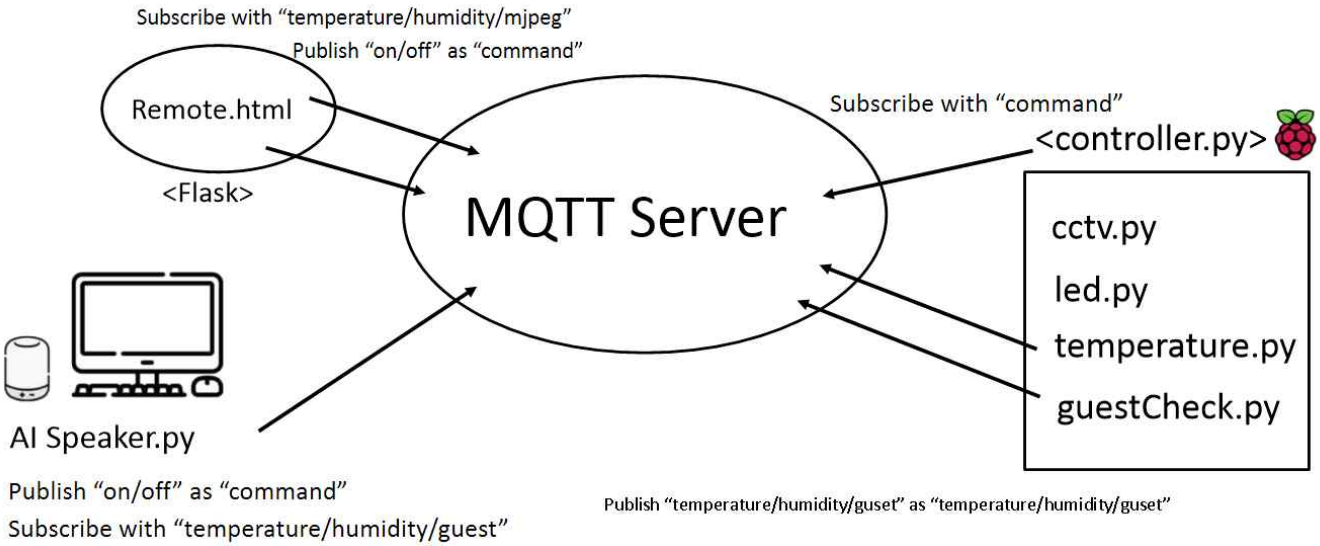
1. 개요

라즈베리파이를 활용하여 음성으로 지시를 내리면 집안의 다양한 시설을 이용할수 있도록 도와주는 스마트 홈 관리 시스템을 구현한다. 라즈베리 파이의 다양한 모듈과 mqtt 서버를 활용하여 인공지능 스피커와 서로 통신한다. 네이버의 클로바, 카카오의 카카오미니 등과 같은 AI Speaker의 열화판을 파이썬으로 구현하여 사용자가 특정 명령어를 음성으로 명령시 해당 명령에 맞춰 mqtt서버로 명령을 보내어 각각의 토픽으로 구독된 라즈베리파이 기기에서 해당 명령을 수행한다. 위 프로젝트의 경우 현재 사용가능한 라즈베리파이 기기가 한 개이므로, 하나의 라즈베리파이에 모든 모듈이 연결하여 사용하지만, 실제로는 다수의 라즈베리 파이가 각각의 기능을 갖고 있다고 가정하고, 추후 목적에 맞춰 사용할 수 있도록 코드는 분리하여 작성한다. 또한 같은 이유로 AI Speaker는 데스크탑에서 작동시킨다. 이 시스템을 사용하여 집안의 다양한 가구,등을 손을 대지 않고 이용할 수 있다. 예를 들어 사용자가 '불 꺼줘' 라고 명령을 내리면 AI Speaker에서 '불'이라는 키워드와 '꺼'라는 키워드를 인식하여 라즈베리파이에 연결된 led전등(led를 led전등이라고 가정) 꺼주는 식이다.

작성할 기능은 다음과 같다.
1. 불 키고 끄기(led모듈 이용)
2. 감시카메라 키고 끄기(정문에 카메라 모듈을 달은 라즈베리파이를 이용)
3. 온도 측정(조도 센서 이용)
4. 손님 확인(초음파 센서)

시스템 구조는 다음과 같다.





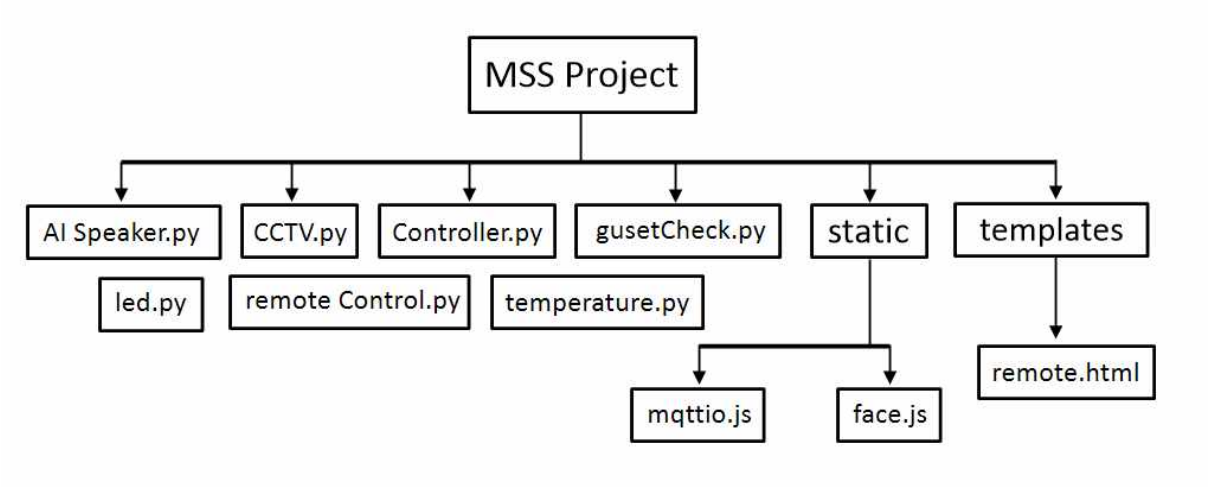
led전등은 on/off 명령에 따라 연결된 led의 조명을 키고 끄고, cctv는 on/off 명령에 맞춰 문 밖에 연결된 감시카메라를 작동시킨다. 온도센서는 창문 근처에 설치하여 사용자가 check 명령시, 해당 시점의 기온을 전송해준다. 초음파 센서는 문 밖에 연결되어 항시 작동하다 손님이 일정거리 이상 접근시 mqtt서버로 손님이 방문했음을 전송해주고, 스피커를 통해 사용자에게 알려준다.
또한 위의 모든 조작을 모바일 리모컨은 Remote.html에서 수동으로 할 수 있도록 한다.

2. 구현 방법

2.1 하드웨어 부분
- 다수의 라즈베리파이를 사용할 경우
: 라즈베리파이 마다 하나의 기능을 갖도록 하나의 모듈을 연결한다.
- 하나의 라즈베리파이를 사용할 경우(위 프로젝트의 경우)
: 라즈베리파이에 카메라 모듈과, led, 온도센서, 습도센서, 초음파 센서를 설치한다.
- AI Speaker
: 라즈베리 파이를 AI Speaker로 사용시, 라즈베리파이에 스피커와 마이크를 설치하여 하나의 기기로 만들 수 있지만, 해당 프로젝트에서는 컴퓨터에 마이크와 스피커를 연결하여 AI Speaker로 기능하도록 한다.

2.2 소프트웨어 부분
소프트웨어는 총 8개의 부분을 설계한다

<디렉토리 구조>



2.2.1 AI speaker.py

: python의 TTS와 STT 라이브러리와 구글 api를 이용하여 음성을 텍스트로 변환하여 조건에 맞춰, mqtt서버로 토픽을 보낸다. 또 mqtt로부터 수신받은 토픽을 음성으로 변환한다.

- 사용자의 음성 명령을 text로 변환하여 어떤 명령인지 파악하고, 해당 명령을 publish한다.
- 또한 적절한 대답을 음성 파일로 변환하여 TTS를 통해 출력한다.

```
# TTS와 STT를 활용한 인공지능 스피커 구현
import sys
import io
import time
import os  # 프로그램 종료 방지용
import threading
import speech_recognition as sr
from gtts import gTTS
from playsound import playsound
import paho.mqtt.client as mqtt

isCall = False  # 호출어가 명령된 상태인지 확인
wai = True  # mqtt 수신 명령을 말하는동안 스피커가 말하는 소리를 음성으로 인식하지 않도록
power = True  # 스피커의 전원에 해당
guest = True  # 손님 방문에 대한 정보를 알렸는지 여부를 저장
answer_text = '입력이 되지 않았습니다'
r = sr.Recognizer()
# = sr.Microphone()  # 마이크에 해당

broker_address = "192.168.216.87"  # input("브로커 IP>>")


# 스피커에서는 온습도정보, 손님방문정보를 구독함
def onConnect(client, userdata, flag, rc):
    if (rc == 0):
        print(" 연결되었습니다")
        client.subscribe("temperature", qos=0)  # 온도 정보
        client.subscribe("humidity", qos=0)  # 습도 정보
        client.subscribe("guest", qos=0)  # 손님 방문 유무
    pass
```

```
def onMessage(client, userdata, msg):
    global guest
    command = str(msg.payload.decode("utf-8")) # 문자열 변환
    if (msg.topic == "temperature"):
        speak("현재 온도는 " + command + "도 이며")
    if (msg.topic == "humidity"):
        speak("현재 습도는 " + command + "입니다.")
    if (msg.topic == "guest"):
        if guest:  # 방문정보를 아직 알린적 없다면
            speak("손님이 방문하였습니다. 모바일 리모컨에서 확인하세요")
            guest = False  # 알렸다고 표시 => 중복 알림 방지
            start_time = threading.Timer(60, reset_guest)  # 1분 후 알림정보 리셋
            start_time.start()
    pass


def listen(recognizer, audio):  # 음성 인식 (듣기 )
    global wait
    try:
        text = recognizer.recognize_google(audio, language='ko')
        print("[User] " + text)

        # mqtt 수신 명령에 대한 처리
        if '날씨' in text:
            client.publish("command", "temperature", qos=0)  # 온도 정보 요구
            client.publish("command", "humidity", qos=0)  # 습도 정보 요구
            wait = False  # 날씨 정보에 관해 스피커가 말하는것을 음성명령으로 인식하지 않도록
        else:  # mqtt 송수신 명령 이외의 명령의 경우 => 불 켜줘, 등~
            answer(text)
    except sr.UnknownValueError:
        print("인식 실패")  # 음성 인식이 실패한 경우
        speak("잘 듣지 못했어요")
    except sr.RequestError as e:  # 네트워크 등의 이유로 연결이 제대로 안됐을경우 API Key 오류, 네트워크 단절 등
        print("요청 실패 : {0}".format(e))  # 에러형식 출력
    pass


def answer(input_text):  # 어떤 대답을 할것인지 정의
    global isCall, power, wait
    global answer_text  # 컴퓨터가 대답할 말 key값이 들어갔다면 출력되도록

    if '안녕' in input_text:
        answer_text = "안녕하세요? 반갑습니다."
    elif '불' in input_text:
        if '켜' in input_text:
            answer_text = "불을 켭니다"
            client.publish("command", "ledOn")  # 조명도에 따라 다른 세기로 불 키기
        elif '꺼' in input_text:
            answer_text = "불을 끕니다"
            client.publish("command", "ledOff")
    elif '감시' in input_text:
        if '켜' in input_text:
            answer_text = "CCTV를 킵니다. 모바일 리모컨에서 확인하세요"
            client.publish("command", "cctvOn")
```

```
        elif '꺼' in input_text:
            answer_text = "CCTV를 끕니다."
            client.publish("command", "cctvOff")
    elif '고마워' in input_text:
        answer_text = "별 말씀을요."
    elif '종료' in input_text:
        answer_text = "다음에 또 만나요."
        isCall = False
        power = False
        wait = False # 프로그램이 완벽히 종료되도록
    elif '거북' in input_text:
        answer_text = "부르셨나요?"
    else: # 사용자가 원하는대로 새로운 명령을 추가 가능
        answer_text = "잘 이해하지 못했어요." #등록되어 있지 않은 명령을 할 경우
    speak(answer_text)


def speak(text):  # 소리내어 읽기 (TTS)
    global wait
    wait = False  # 스피커가 말하는 동안은 명령 수신x
    print('[인공지능] ' + text)  # 인공지능이 하는말 텍스트 출력
    if "손님" in text:
        file_name = 'voice2.mp3'  # 음성파일 중복 삭제로 인한 오류를 해결하기 위함
    else:
        file_name = 'voice.mp3'
    tts = gTTS(text=text, lang='ko')  # 한글로 저장
    tts.save(file_name)  # file_name으로 해당 mp3파일 저장
    playsound(file_name)  # 저장한 mp3파일을 읽어줌
    if os.path.exists(file_name):  # file_name 파일이 존재한다면
        os.remove(file_name)  # 실행 이후 mp3 파일 제거
        wait = True


# 음성 명령 권한을 회수하는 메소드
def reset_mode():
    global isCall
    isCall = False


# 손님 방문 알림에 대한 정보를 리셋
def reset_guest():
    global guest
    guest = True


# MQTT 제어
client = mqtt.Client()        # mqtt 클라이언트 객체 생성
client.on_connect = onConnect    # 연결요청시 Callback 함수
client.on_message = onMessage    # 이미지가 도착하였을때 Callback 함수
client.connect(broker_address, 1883)    # 브로커에 연결을 요청함
client.loop_start()  # 인공지능 스피커가 동작해야하므로 비동기식으로 별도의 스레드로 진행

# 스피커가 동작하는 공간
while power:
    with sr.Microphone() as source:  # 마이크에서 들리는 음성(source)을 listen을 통해 들음
        print('Listening')  # 잠깐의 대기 시간이 있으므로 확인용으로 텍스트 출력
```

```
        callName = r.listen(source)  # 마이크로부터 호출어 듣기
        text = ""  # 예외처리문 밖에서도 이용하기 위해
        # 호출어에 대한 예외처리
        try:
            # 구글 API 로 인식 (하루 50회만 허용)
            # 영어는 language = 'en-US
            text = r. recognize_google(callName, language='ko')  # ko = 한국어 음성을 인식
            print(text)  # 정상 작동 확인용
        except sr.UnknownValueError:
            print("인식 실패")  # 음성 인식이 실패한 경우
        except sr.RequestError as e:  # 네트워크 등의 이유로 연결이 제대로 안됐을경우 API Key 오류, 네트워크 단절 등
            print("요청 실패 : {0}".format(e))  # 에러형식 출력
            # 사용자에게 오류사실을 알림
            speak("네트워크 오류가 발생했습니다. 지속적인 오류가 발생할 경우 관리자에게 문의하세요")

        if "거북" in text:  # 음성이 키워드일때 명령을 할수있게 변경
            isCall = True
            wait = True
            speak("안녕하세요!")

            # 아무 명령을 하지 않더라도, 20초가 지나면 명령 권한 회수
            start_time = threading.Timer(50, reset_mode)  # 20초후 reset
            start_time.start()
        else:
            continue

    # 명령 권한이 있는 상태에서는 계속 명령을 받을 수 있음
    while isCall:
        while wait:
            with sr.Microphone() as source:
                command = r.listen(source)  # 마이크로부터 음성 듣기
                listen(r, command)
client.loop_stop()
#client.publish("command","ledOff") # 필요에 따라 추가 => 스피커를 끈다는 설정이므로 굳이 안넣음
#clientlient.publish("command","cctvOff") # 스피커가 꺼져도 명령은 유지되어야 하므로

client.disconnect()
```

2.2.2 CCTV.py

: mqtt에 camera command 라는 토픽으로 구독하여 on 명령이 오면 사진을 계속 찍어 전송하고 off 명령이 오면 전송을 중단한다. => 이미지의 경로를 받아와서 canvas에 그리는 방식으로 구현

```
# publisher/subscriber
import time
import paho.mqtt.client as mqtt
import mycamera  # 카메라 사진 보내기

flag = False

# mqtt 연결용
def on_connect(client, userdata, flag, rc):
```

```
    client.subscribe("command", qos=0)


def on_message(client, userdata, msg):
    global flag
    command = msg.payload.decode("utf-8")
    if command == "cctvOn":
        print("cctv On")
        flag = True
    elif command == "cctvOff":
        print("cctv Off")
        flag = False

# 별도의 라즈베리 파이에서 작동하는 경우
if __name__ == '__main__':
    broker_ip = "localhost"  # 현재 이 컴퓨터를 브로커로 설정
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message

    client.connect(broker_ip, 1883) # 새로운 라즈베리파이에서 동작시, mqtt에 별도로 연결함
    client.loop_start()
    while True:
        if flag:
            imageFileName = mycamera.takePicture()  # 카메라 사진 촬영
            print(imageFileName)
            client.publish("mjpeg", imageFileName, qos=0)
        time.sleep(0.5)
    client.loop_end()
    client.disconnect()
```

2.2.2-1

: cctv.py에 import되어 사용되며, 카메라로 사진 촬영 후 opencv를 이용하여 얼굴을 인식하여 얼굴에 사각형으로 그리고 jpg 파일로 저장. 그리고 jpg 파일 이름 리턴

```
import os; import io; import time
import picamera
import cv2; import numpy as np
fileName = ""
stream = io.BytesIO()
camera = picamera.PiCamera()
camera.start_preview()
camera.resolution = (320, 240)
time.sleep(1)

def takePicture() :
        global fileName, stream, camera

        if len(fileName) != 0:
                os.unlink(fileName)

        stream.seek(0)
        stream.truncate()
        camera.capture(stream, format='jpeg', use_video_port=True)
        data = np.frombuffer(stream.getvalue(), dtype=np.uint8)
```

```
        image = cv2.imdecode(data, 1)
        haar = cv2.CascadeClassifier('./haarCascades/haar-cascade-files-master/haarcascade_frontalface_default.xml')
        image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        faces = haar.detectMultiScale(image_gray,1.1,3)
        for x, y, w, h in faces:
                cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)

        takeTime = time.time()
        fileName = "./static/%d.jpg" % (takeTime * 10)
        cv2.imwrite(fileName, image)
        return fileName

if __name__ == '__main__' :
        while(True):
                name = takePicture()
                print("fname= %s" % name)
```

2.2.3 temperature.py

: mqtt에 temp command 라는 토픽으로 구독하여 check 명령이 오면 현재의 기온,습도, 등의 정보를 보낸다.

```
import RPi.GPIO as GPIO
import busio
import time
from adafruit_htu21d import HTU21D

sda = 2  # GPIO핀 번호
scl = 3  # GPIO핀 번호
i2c = busio.I2C(scl, sda)
sensor = HTU21D(i2c)  # HTU21D 장치를 제어하는 객체 리턴


# 온도 정보 전달용
def getTemperature():
    temperature = int(sensor.temperature)  # HTU21D 장치로부터 온도 값 읽기
    return temperature #소수점 제거해서 리턴


# 습도 정보 전달용
def getHumidity():
    humidity = int(sensor.relative_humidity)  # HTU21D 장치로부터 습도 값 읽기
    return humidity #수수점 제거해서 리턴

# 아래 코드는 독립적으로 코드가 실행되었을때를 가정하고 제작한다.
# controller에 import 될 때는 getTemperature()함수와 getHumidity()함수만 이용된다.

# 별도 작동용 => controller로 연결하지 않고 다수의 라즈베리파이를 사용할때

def onConnect(client, userdata, flag, rc):
    print("Connect with result code:" + str(rc))
    client.subscribe("command", qos=0)     # command라는 토픽 데이터 수신 요청
    pass

def onMessage(client, userdata, msg):
```

```
    command = str(msg.payload.decode("utf-8"))
    if command == 'temperature':
        print("온도 정보 전달 완료")
        print( getTemperature())
        client.publish(
            "temperature", getTemperature(), qos=0)
    elif command == 'humidity':
        print("습도 정보 전달 완료")
        print( getHumidity())
        client.publish("humidity", getHumidity(), qos=0)
    pass


# import 되지않고, 별도의 라즈베리파이에서 작동하는 상황을 가정 => 별도로 mqtt에 연결한다.
if __name__ == '__main__':
    import paho.mqtt.client as mqtt
    # mqtt 사용용
    broker_address = "localhost"

    client = mqtt.Client()
    client.on_connect = onConnect
    client.on_message = onMessage

    client.connect(broker_address, 1883)
    client.loop_start()

    while (True):
        time.sleep(0.1)  # 프로그램 종료 방지용
```

2.2.4 led.py

: mqtt에 light command 라는 토픽으로 구독하여 on/off 명령에 따라 led로 구성된 전등을 키고, 끈다.

```
import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

led = 6  # 핀 번호 GPIO6 의미, 빵판하단 16번 홀

GPIO.setup(led, GPIO.OUT)  # GPIO 6번 핀을 출력 선으로 지정.


# led 작동용 => controller.py에서 이용함
def ledOnOff(onOff):
    global led
    GPIO.output(led, onOff)


# 별도 작동용 => controller로 연결하지 않고 다수의 라즈베리파이를 사용할때
```

```
def onConnect(client, userdata, flag, rc):
    print("Connect with result code:" + str(rc))
    client.subscribe("command", qos=0)     # command라는 토픽 데이터 수신 요청
    pass


def onMessage(client, userdata, msg):
    command = str(msg.payload.decode("utf-8"))
    if (command == 'ledOn'):
        ledOnOff(True)        # LED ON
    elif (command == 'ledOff'):
        ledOnOff(False)    # LED OFF
    pass


# import 되지않고, 별도의 라즈베리파이에서 작동하는 상황을 가정
if __name__ == '__main__':
    import paho.mqtt.client as mqtt
    # mqtt 사용용

    broker_address = "localhost"

    client = mqtt.Client()
    client.on_connect = onConnect
    client.on_message = onMessage

    client.connect(broker_address, 1883)
    client.loop_start()

    while (True):
        time.sleep(0.1)  # 프로그램 종료 방지용
```

2.2.5 remote.html

: 모바일 리모컨으로서 기동하며, 위의 모든 조작을 수동으로 할 수 있는 UI를 제공해준다. 사용자는 이를 통해, 음성으로 명령을 내리지 않더라도 수동으로 직접 버튼을 눌러, 다양한 조작을 할 수 있다. 또한 cctv.py에서 전송된 이미지를 화면에 출력하여 cctvViewer로서의 역할을 수행한다. 웹 플라스크로 작동하기 때문에, 사용자는 언제 어디서나 cctv를 확인하고 끄고 킬 수 있다.

(templates 디렉토리 밑에 작성)

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>모바일 리모컨</title>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.2/mqttws31.min.js" type="text/javascript"></script>
        <script src="/static/mqttio.js" type="text/javascript"></script>
        <script src="/static/face.js" type="text/javascript"></script>
        <script>
            window.addEventListener("load", function () {
            // http://224..129:8080/에서 224...의 IP만 끊어내는 코드
            var url = new String(document.location);
            ip = (url.split("//"))[1]; // ip = "224...:8080/"
            ip = (ip.split(":"))[0]; // ip = "224..."
```

```
            document.getElementById("broker").value = ip
        });
        </script>
        <style>
            #myCanvas{
                position: relative;
                border: 3px solid black;
                background-color:white
            }
            body{
                background-color: gray;
            }
            #startBtn,#endBtn{
                position: relative;
                bottom: 10px;
            }
            #title{
                color:green; font-size:40px; text-shadow:2px 2px white
            }
        </style>
    </head>
    <body>
        <h3 id="title">모바일 리모컨</h3>
        <hr>
        <form id="connection-form">
            <b>브로커 IP:</b>
            <input id="broker" type="text" name="broker" value=""><br>
            <b>포트 번호 : 9001</b><br>
            <input type="button" onclick="startConnect()" value="Connect">
            <input type="button" onclick="startDisconnect()" value="Disconnect">
        </form>
        <hr>
        <h3>전등 조작 :  <label>on <input type="radio" name="led" value="1" onchange="publish('command', 'ledOn')"></label>
        <label>off <input type="radio" name="led" value="0" checked onchange="publish('command',
'ledOff')"><br><br></label></h3>
        <hr>
        <h3>실내 온도/습도 <input type="button" onclick="getTemperature()" value="확인하기"></h3>
        <br>
        <div id="temperature"></div>
        <div id="humidity"></div>
        <hr>
        <h3>CCTV(use OpenCV)</h3>
        <input id ="startBtn" type="button" onclick="recognizeStart()" value="시작">
        <input id ="endBtn" type="button" onclick="recognizeStop()" value="종료">
        <form>
            <canvas id="myCanvas" width="320" height="240"></canvas>
        </form>
        <div id="messages"></div>
    </body>
</html>
```

2.2.5-1 mattio.js

: remote.html에서 사용되는 JavaScript코드

mqtt연결과 remote.html에서 사용되는 함수들을 내포한다.

```
var port = 9001 // mosquitto의 디폴트 웹 포트
var client = null; // null이면 연결되지 않았음

function startConnect() { // 접속을 시도하는 함수
    clientID = "clientID-" + parseInt(Math.random() * 100); // 랜덤한 사용자 ID>
    // 사용자가 입력한 브로커의 IP 주소와 포트 번호 알아내기
    broker = document.getElementById("broker").value; // 브로커의 IP 주소
    // id가 message인 DIV 객체에 브로커의 IP와 포트 번호 출력
    // MQTT 메시지 전송 기능을 모두 가징 Paho client 객체 생성
    client = new Paho.MQTT.Client(broker, Number(port), clientID);
    // client 객체에 콜백 함수 등록
    client.onConnectionLost = onConnectionLost; // 접속이 끊어졌을 때 실행되는 >
    client.onMessageArrived = onMessageArrived; // 메시지가 도착하였을 때 실행[>
    // 브로커에 접속. 매개변수는 객체 {onSuccess : onConnect}로서, 객체의 프로[>
    // 접속에 성공하면 onConnect 함수를 실행하라는 지시
    client.connect({
        onSuccess: onConnect,
    });
}



var isConnected = false;

// 브로커로의 접속이 성공할 때 호출되는 함수
function onConnect() {
    isConnected = true;
    subscribe("temperature") #온도,습도,이미지를 받기위해 구독
    subscribe("humidity")
    subscribe("mjpeg")
}

var topicSave;
function subscribe(topic) {
    if(client == null) return;
    if(isConnected != true) {
        topicSave = topic;
        window.setTimeout("subscribe(topicSave)", 500);
        return
    }
// 토픽으로 subscribe 하고 있음을 id가 message인 DIV에 출력
    document.getElementById("messages").innerHTML += '<span>Subscribing to: ' + topic + '</span><br/>';

    client.subscribe(topic); // 브로커에 subscribe
}

function publish(topic, msg) {
    if(client == null) return; // 연결되지 않았음
    client.send(topic, msg, 0, false);
}

function unsubscribe(topic) {
    if(client == null || isConnected != true) return;

    // 토픽으로 subscribe 하고 있음을 id가 message인 DIV에 출력
    document.getElementById("messages").innerHTML += '<span>Unsubscribing to: ' + topic + '</span><br/>';
```

```
    client.unsubscribe(topic, null); // 브로커에 subscribe
}

// 접속이 끊어졌을 때 호출되는 함수
function onConnectionLost(responseObject) { // 매개변수인 responseObject는 응답 패킷의 정보를 담은 개체
    document.getElementById("messages").innerHTML += '<span>오류 : 접속 끊어짐</span><br/>';
    if (responseObject.errorCode !== 0) {
        document.getElementById("messages").innerHTML += '<span>오류 : ' + + responseObject.errorMessage + '</span><br/>';
    }
}

// 메시지가 도착할 때 호출되는 함수
function onMessageArrived(msg) { // 매개변수 msg는 도착한 MQTT 메시지를 담고 있는 객체
    console.log("onMessageArrived: " + msg.payloadString);

    // 도착한 메시지 출력
    if(msg.destinationName == "mjpeg") { # canvas에 이미지를 출력하기 위함
        drawImage(msg.payloadString); // 메시지에 담긴 파일 이름으로 drawImage() 호출. drawImage()는 웹 페이지에 있음
    }
    else if(msg.destinationName == 'temperature'){
        document.getElementById("temperature").innerHTML = "현재 온도 : " + msg.payloadString
    }
    else if(msg.destinationName == 'humidity'){
        document.getElementById("humidity").innerHTML = "현재 습도 : " + msg.payloadString
    }
    // 토픽 image가 도착하면 payload에 담긴 파일 이름의 이미지 그리기


    document.getElementById("messages").innerHTML = '<span>토픽 : ' + msg.destinationName + '  | ' + msg.payloadString + '</span><br/>';
}

// disconnection 버튼이 선택되었을 때 호출되는 함수
function startDisconnect() {
    client.disconnect(); // 브로커에 접속 해제
    document.getElementById("messages").innerHTML += '<span>Disconnected</span><br/>';
}

// 날씨 확인하기
function getTemperature(){
    client.send("command", "temperature", 0, false); //날씨 확인 요청
    client.send("command", "humidity",0,false);
}
```

2.2.5-2 face.js

: html에 이벤트 리스너를 등록하고, 이미지가 도착하면 canvas에 그린다.

cctv 작동과 중지에 대한 함수를 내포한다.

```
// 전역 변수 선언
var canvas;
var context;
var img;
```

```
// load 이벤트 리스너 등록. 웹페이지가 로딩된 후 실행
window.addEventListener("load", function() {
	canvas = document.getElementById("myCanvas");
	context = canvas.getContext("2d");

	img = new Image();
	img.onload = function () {
		context.drawImage(img, 0, 0); // (0,0) 위치에 img의 크기로 그리기
	}
});

// drawImage()는 "image' 토픽이 도착하였을 때 onMessageArrived()에 의해 호출된다.
function drawImage(imgUrl) { // imgUrl은 이미지의 url
	img.src = imgUrl; // img.onload에 등록된 코드에 의해 그려짐
}

var isImageSubscribed = false;
function recognizeStart() {
	if(!isImageSubscribed) {
		subscribe('mjpeg'); // 토픽 image 등록
		isImageSubscribed = true;
	}
	publish('command','cctvOn'); // 토픽: facerecognition, start 메시지 전송
}

function recognizeStop() {
    publish('command','cctvOff'); // 토픽: facerecognition, start 메시지 전송
}
```

2.2.6 guestCheck.py

: 초음파센서를 통해 사용자의 집에 일정거리 이상 사람이 가까워지면, mqtt서버로 손님이 방문했다는 알림을 보낸다.

```
# 손님이 왔는지 확인해서 mqtt로 전달함
import time
import RPi.GPIO as GPIO
import time
import paho.mqtt.client as mqtt

trig = 20  # 빵판으로는 상단19번
echo = 16  # 빵판으로는 상단18번
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(trig, GPIO.OUT)
GPIO.setup(echo, GPIO.IN)
GPIO.output(trig, False)

#  거리 측정용 함수
def measureDistance(trig, echo):
    time.sleep(0.5)
    GPIO.output(trig, True)  # 신호 1 발생
    time.sleep(0.00001)  # 짧은 시간을 나타내기 위함
    GPIO.output(trig, False)  # 신호 0 발생
```

```
    while(GPIO.input(echo) == 0):
        pulse_start = time.time()  # 신호 1을 받았던 시간
    while(GPIO.input(echo) == 1):
        pulse_end = time.time()  # 신호 0을 받았던 시간

    pulse_duration = pulse_end - pulse_start
    return 340*100/2*pulse_duration

# 별도의 라즈베리파이에서 독립적으로 작동되는 경우

if __name__ == '__main__':
    import paho.mqtt.client as mqtt  # 독립적으로 작동할때 mqtt 개별 연결
    ip = "localhost"

    client = mqtt.Client()
    client.connect(ip, 1883)

    client.loop_start()  # 비동기식으로 작동

    while True:
        distance = measureDistance()
        if distance < 10:  # 물체가 일정거리 이상 가까워질경우 => 손님방문
            print("손님 방문")
            client.publish("guest", distance, qos=0)
            client.publish("command","cctvOn",qos=0)
            time.sleep(10)  # 손님 방문사실을 알린 이후, 중복 메세지를 보내지 않도록 한다.
```

2.2.7 remoteControl.py

: Flask로 remote.html을 작동시켜, 사용자가 언제 어디서나 접속할수 있도록 한다.

```
from flask import Flask, render_template, request
app = Flask(__name__)

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

if __name__ == "__main__":
        app.run(host='0.0.0.0', port=8080)
```

2.2.8 controller.py

: mqtt서버와 연결하여 command에 따라 적절한 모듈과 연결하여 명령을 수행한다.
led.py, temperature.py, gusetCheck.py,등을 import하여 처리하는 중앙 처리장치의 역할을 한다.

```
import sys
import io
import time
import paho.mqtt.client as mqtt
# cctv 작동용
import mycamera  # 카메라 사진 보내기
import threading # timer함수를 이용하기 위해
import cv2

# 외부 모듈 import
```

```
import temperature
import led
import guestCheck

# 카메라 제어용
isStarted = False

# AI Speaker에서 mqtt로 보낸 command에 맞춰 명령을 처리하는 중앙 장치

# 손님 방문 사실 알림 여부를 저장
guest = True


# 손님 방문 알림에 대한 정보를 리셋 => 중복 알림 방지
def reset_guest():
    global guest
    guest = True


# mqtt 사용용
def onConnect(client, userdata, flag, rc):
    print("Connect with result code:" + str(rc))
    client.subscribe("command", qos=0)     # command라는 토픽 데이터 수신 요청
    pass


# command에 대응
def onMessage(client, userdata, msg):
     global isStarted
    command = str(msg.payload.decode("utf-8"))

    if command == 'temperature':
        print("온도 정보 전달 완료") #개발자 확인용
        print(temperature.getTemperature())
        client.publish(
            "temperature", temperature.getTemperature(), qos=0)
    elif command == "humidity":
        print("습도 정보 전달완료") #개발자 확인용
        print(temperature.getHumidity())
        client.publish("humidity", temperature.getHumidity(), qos=0)

    elif command == 'ledOn':
        led.ledOnOff(True)       # LED ON
    elif command == 'ledOff':
        led.ledOnOff(False)    # LED OFF

    elif command == 'cctvOn':
        print("cctv start!") #개발자 확인용
        isStarted = True
    elif command == 'cctvOff':
        print(("cctv stop!")) #개발자 확인용
        isStarted = False
    pass


broker_address = "localhost"
```

```
client = mqtt.Client()
client.on_connect = onConnect
client.on_message = onMessage

client.connect(broker_address, 1883)
client.loop_start()


while True:
    # cctv 사진 송출 시스템,
    # Controller를 사용하지 않고 개별로 동작시킬 경우 cctv.py와 gusetCheck.py에서 이 역할을 대신함
    if isStarted:
        imageFileName = mycamera.takePicture()  # 카메라 사진 촬영
        print(imageFileName)
        client.publish("mjpeg", imageFileName, qos=0)
    time.sleep(0.5)

    # 손님 방문 알림 시스템
    distance = guestCheck.measureDistance()
    if distance < 10:  # 물체가 일정거리 이상 가까워질경우 => 손님방문
        if guest == True:  # 손님 방문사실을 여러번 알리는것을 방지하기 위함
            print("손님 방문")
            isStarted = True # 손님 방문시 카메라 켜기
            client.publish("guest", distance, qos=0)
            guest = False  # 재알림 방지
            start_time = threading.Timer(60, reset_guest)  # 1분 후 알림정보 리셋
            start_time.start()

client.disconnect()
```

모든 코드들은 __name__을 이용하여, 독립적인 라즈베리파이에서 시행되었을 경우와, controller에서 제어되는 경우, 두 가지 상황을 모두 수행할 수 있도록 작성하였다. 예를 들어 집의 구조상 하나의 시스템에서 모든 것을 관리하기 어려운 경우 즉, controller.py에서 모든 것을 관리하기 어려운 경우를 대비하여 CCTV.py와 gusetCheck.py 등, 필요에 따라 새로운 라즈베리파이에 관련 모듈을 연결하여 사용할 수 있도록 하였다.

3. 실행 과정 및 결과

3.1 음성 명령을 통해 AI Speaker.py에서 인식하여 작동하는 모습

```
[인공지능] 불을 켭니다
[User] 불 꺼 줘
[인공지능] 불을 끕니다
[User] 고마워
[인공지능] 별 말씀을요.
[User] 감시 카메라 켜 줄래
[인공지능] CCTV를 킵니다. 모바일 리모컨에서 확인하세요
```

```
[User] 날씨 알려 줘
[인공지능] 현재 온도는 24도 이며
[인공지능] 현재 습도는 45입니다.
[User] 현재 습도는 손님이 방문하는
[인공지능] 잘 이해하지 못했어요.
[인공지능] 손님이 방문하였습니다. 모바일 리모컨에서 확인하세요
(base)
```

```
[User] 감시 카메라 꺼 줘
[인공지능] CCTV를 끕니다.
```

```
[User] 종료 할게
[인공지능] 다음에 또 만나요.
```

3.2 모바일 리모컨에서 동작하는 모습

(추가)
영상에는 CSS가 적용되어있지 않지만, 리모컨처럼 보이게끔 약간의 CSS를 추가하였다.

모바일 리모컨

브로커 IP: 127.0.0.1
포트 번호 : 9001
Connect Disconnect

전등 조작 : on ● off ○

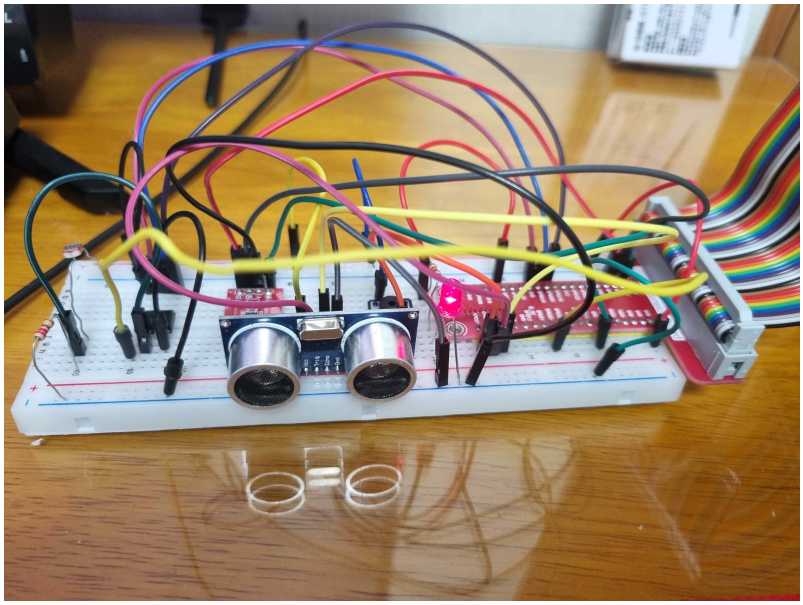
실내 온도/습도 확인하기

CCTV(use OpenCV)

시작 종료

3.3 명령에 맞춰 라즈베리파이가 동작하는 모습

ledOn명령



ledOff 명령