

모바일&스마트시스템(B)

라즈베리파이를 이용한 홈 CCTV

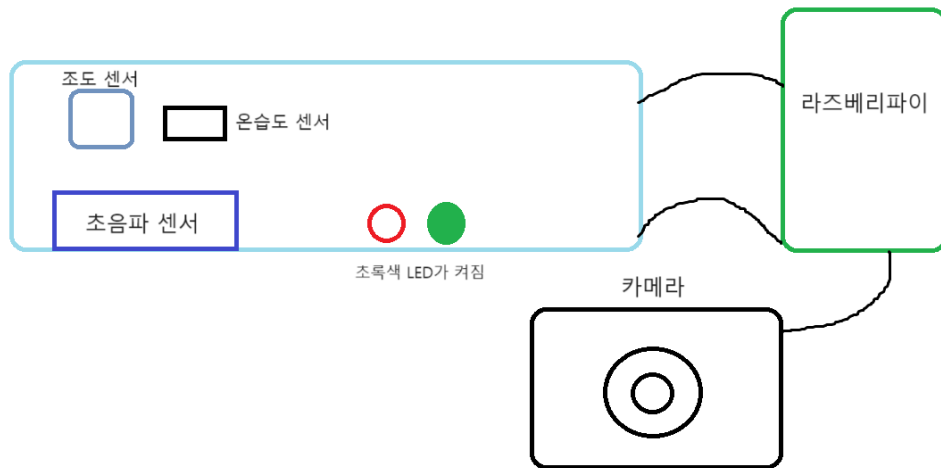
모바일소프트웨어트랙

2291031

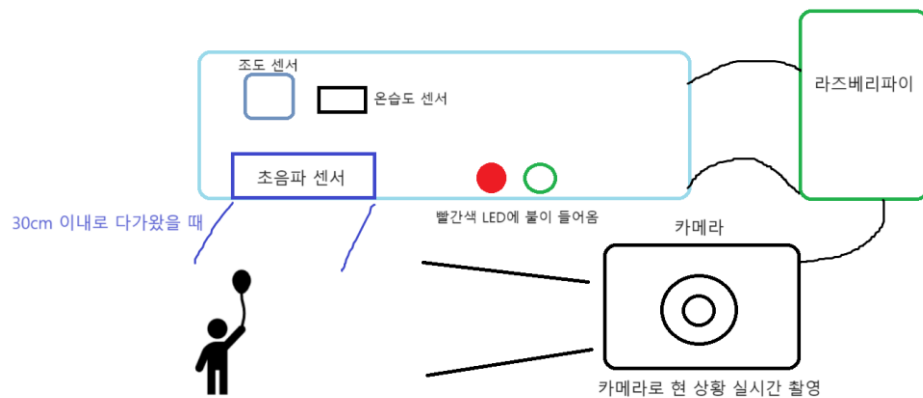
전아린

1. 작품 개요

현대 사회에서 맞벌이 가정이 증가함에 따라 아이가 혼자 집에 머무는 시간이 늘어나게 되었다. 이러한 상황에서 아이가 집에 혼자 있을 때 발생할 수 있는 위험 상황에 대응하기 위해 집의 현재 상황을 파악할 수 있는 홈 CCTV를 이번 프로젝트의 주제로 삼게 되었다. 먼저, LED와 초음파 센서를 이용해 평소에는 초록색 LED를 켜 안전하다는 것을 보여주고, 아이가 위험한 곳에서부터 30cm 이내에 있을 경우엔 빨간색 LED를 켜 아이가 위험을 느끼고 피할 수 있도록 한다. 동시에 아이 보호자에게 현재 아이가 위험한 상황에 처했다는 사실을 알리고 현재 상황을 카메라로 촬영해 보여준다. 아이가 위험한 상황에서 벗어난 경우 다시 초록색 LED를 키고 실시간으로 보여주던 카메라의 작동을 멈춘다. 또한, 조도 센서와 온습도 센서를 이용해 집이 너무 어둡거나 습하지는 않은 지 사용자가 언제든지 알아볼 수 있도록 한다.



<아무 일도 없을 때>



<30cm 이내로 아이가 다가왔을 때>

2. 구현 방법

2.1 하드웨어 부분

라즈베리파이에 카메라 1개, 빨간색 LED 1개, 초록색 LED 1개, 초음파 센서 1개, 조도 센서 1개, 온습도 센서 1개를 사용한다.

LED

빨간색 LED: GPIO6 초록색 LED: GPIO13

초음파 센서

trig: GPIO20 echo: GPIO16

온습도 센서

SCL: GPIO3 SDA: GPIO2

조도 센서

cs/shdn: GPIO8 clk: GPIO11 miso: GPIO9 mosi: GPIO10

카메라

라즈베리파이에 연결

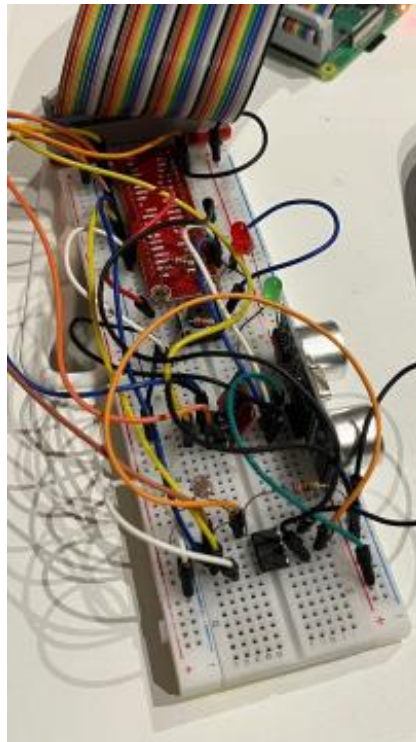
2.2 소프트웨어 부분

- 웹 브라우저로부터 접속과 요청을 받아 처리하는 플라스크
 - 초음파 센서, LED, 조도 센서, 온습도 센서 제어 코드
 - 카메라 제어 코드
 - MQTT 브로커에 연결한 뒤 장치와 센서를 제어하고 모니터링하는 코드
 - paho 라이브러리를 사용하여 MQTT 브로커에 접속하는 자바스크립트
1. 자바스크립트 함수를 통해 웹 페이지에서 MQTT 브로커에 접속
 2. 센서 제어 코드에서 조도와 온습도 값을 받아 웹 브라우저에서 출력함. 평상시엔 센서 제어 코드에 의해 초록색 LED의 불을 켜
 3. 센서 제어 코드에서 초음파 센서가 아이가 30cm 이내로 다가온 것을 감지한 경우 초록색 LED를 끄고 빨간색 LED의 불을 켜 뒤 카메라 제어 코드를 작동시킴. 카메라

코드는 아이의 얼굴을 인식한 뒤 읽은 값을 html 코드로 넘겨 브라우저에 현재 상황을 실시간으로 출력함.

3. 구현 내용

3.1 회로도



<설치 구조>

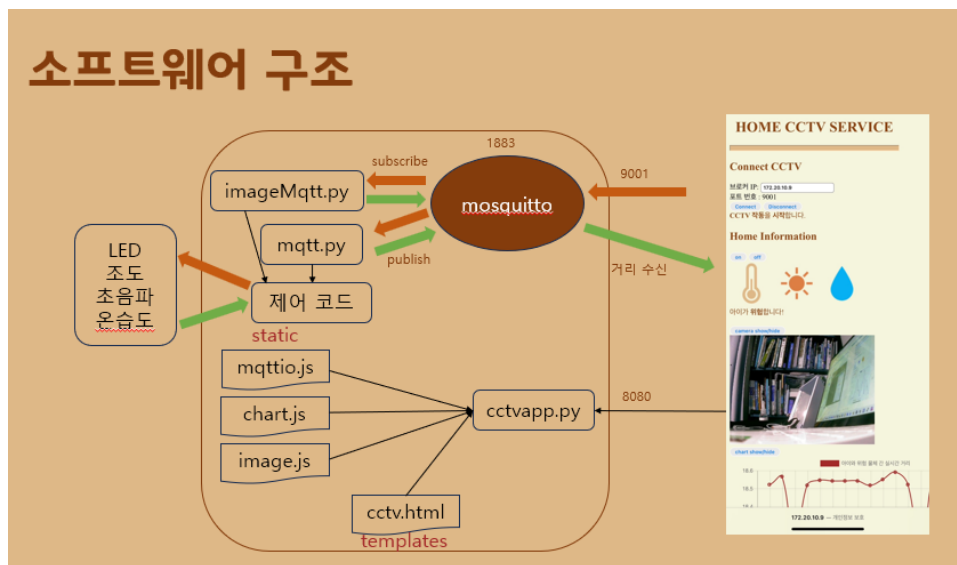
3.2 프로그램 코드 및 설명

```

pi@jun2291031:~/project2 $ tree
.
├── camera.py
├── cctvapp.py
├── classifier.py
├── control.py
├── haarModels
│   └── haarcascade_frontalface_default.xml
├── imageMqtt.py
├── led.py
├── lumi.py
├── mqtt.py
├── __pycache__
│   ├── camera.cpython-39.pyc
│   ├── classifier.cpython-39.pyc
│   ├── control.cpython-39.pyc
│   ├── led.cpython-39.pyc
│   ├── lumi.cpython-39.pyc
│   └── temp.cpython-39.pyc
├── static
│   ├── averagehumid.png
│   ├── averagetemp.png
│   ├── chart.js
│   ├── cold.png
│   ├── hot.png
│   ├── humid.png
│   ├── image.js
│   ├── lowhumid.png
│   ├── moon.png
│   ├── mqttio.js
│   ├── sun.png
│   └── templates
│       └── cctv.html
└── temp.py

```

<디렉토리 구조>



<소프트웨어 구조>



<static 디렉토리에 저장한 이미지 파일>

led.py (LED 제어)

```
import time
import RPi.GPIO as GPIO

# pin 에 연결된 LED 에 value(0/1) 값을 출력하여 LED 를 켜거나 끄는 함수
def ledOnOff(pin, value):
    GPIO.output(pin, value)

def controlledLED(on_off):
    GPIO.output(redLed, on_off)
    GPIO.output(greenLed, on_off)

GPIO.setmode(GPIO.BCM) # BCM 모드로 작동
GPIO.setwarnings(False) # 경고글이 출력되지 않게 설정

redLed = 6 # 빨간색 LED 를 GPIO6 에 연결
greenLed = 13 # 초록색 LED 를 GPIO13 에 연결
GPIO.setup(redLed, GPIO.OUT) # GPIO6 핀을 출력으로 지정
GPIO.setup(greenLed, GPIO.OUT) # GPIO13 핀을 출력으로 지정
```

lumi.py (조도 센서 제어)

```
import time
import RPi.GPIO as GPIO
import Adafruit_MCP3008

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

```
mcp = Adafruit_MCP3008.MCP3008(clk=11, cs=8, miso=9, mosi=10)
```

```
def getLuminance():  
    return mcp.read_adc(0)
```

temp.py (온습도 센서 제어)

```
import time  
import RPi.GPIO as GPIO  
from adafruit_htu21d import HTU21D  
import busio  
  
def getTemperature(sensor) : # 센서로부터 온도 값 수신 함수  
    return float(sensor.temperature) # HTU21D 장치로부터 온도 값 읽기  
  
def getHumidity(sensor) : # 센서로부터 습도 값 수신 함수  
    return float(sensor.relative_humidity) # HTU21D 장치로부터 습도 값 읽기  
  
GPIO.setmode(GPIO.BCM)  
GPIO.setwarnings(False)  
sda = 2 # GPIO2 핀. sda 이름이 붙여진 핀  
scl = 3 # GPIO3 핀. scl 이름이 붙여진 핀  
  
i2c = busio.I2C(scl, sda) # I2C 버스 통신을 실행하는 객체 생성  
sensor = HTU21D(i2c) # I2C 버스에서 HTU21D 장치를 제어하는 객체 리턴
```

control.py (초음파 센서 제어)

```
import time  
import RPi.GPIO as GPIO  
  
# 초음파 센서를 제어하여 물체와의 거리를 측정하여 거리 값 리턴하는 함수  
def measure_distance():  
    GPIO.output(trig, 1) # trig 핀에 1(High) 출력  
    GPIO.output(trig, 0) # trig 핀에 0(Low) 출력. High->Low. 초음파 발사 지시  
  
    while(GPIO.input(echo) == 0): # echo 핀 값이 0->1로 바뀔 때까지 루프  
        pass  
  
    # echo 핀 값이 1이면 초음파가 발사되었음  
    pulse_start = time.time() # 초음파 발사 시간 기록  
    while(GPIO.input(echo) == 1): # echo 핀 값이 1->0으로 바뀔 때까지 루프  
        pass  
  
    # echo 핀 값이 0이 되면 초음파 수신하였음  
    pulse_end = time.time() # 초음파가 되돌아 온 시간 기록  
    pulse_duration = pulse_end - pulse_start # 경과 시간 계산  
    return pulse_duration*340*100/2 # 거리 계산하여 리턴(단위 cm)
```

```
# 초음파 센서를 다루기 위한 전역 변수 선언 및 초기화
trig = 20 # GPIO20
echo = 16 # GPIO16
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(trig, GPIO.OUT) # GPIO20 핀을 출력으로 지정
GPIO.setup(echo, GPIO.IN) # GPIO16 핀을 입력으로 지정
```

camera.py (카메라 제어 코드)

```
import sys
import time
import cv2
camera = None
# 카메라 초기화
def init(camera_id=0, width=640, height=480, buffer_size=1):
    global camera
    camera = cv2.VideoCapture(camera_id, cv2.CAP_V4L)
    camera.set(cv2.CAP_PROP_FRAME_WIDTH, width)
    camera.set(cv2.CAP_PROP_FRAME_HEIGHT, height)
    camera.set(cv2.CAP_PROP_BUFFERSIZE, buffer_size)

def take_picture(most_recent=False):
    global camera
    # most_recent 가 True 이면 버퍼에 저장되어 있는 프레임을 전부 버리도록 한다.
    len = 0 if most_recent == False else camera.get(cv2.CAP_PROP_BUFFERSIZE)
    while(len > 0):
        camera.grab() # 버퍼에 저장되어 있는 프레임을 버린다.
        len -= 1
    success, image = camera.read()
    if not success:
        return None

    return image
# 카메라 해제
def final():
    if camera != None:
        camera.release()
    camera = None
```

classifier.py (사진에서 얼굴을 찾아 사각형을 그리고 리턴하는 코드)

```
import cv2

# 검출 객체 2 개 생성(얼굴 인식용 1 개, 눈 인식용 1 개)
```



```

faceClassifier =
cv2.CascadeClassifier('./haarModels/haarcascade_frontalface_default.xml')

def classify(image):

    # 이미지를 흑백으로 바꾸고 얼굴과 눈 탐지
    image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = faceClassifier.detectMultiScale(image_gray) # 얼굴 탐지

    # 노란색(0, 255, 255) 사각형으로 얼굴 표시
    numberOfPeople = 0
    for x, y, w, h in faces:
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 255), 4)

    return image

```

cctvapp.py (웹 페이지를 렌더링해 리턴하는 플라스크 앱)

```

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

# 자바스크립트 코드나 이미지 파일 등에 대해
# 브라우저에게 캐시에 저장한 파일을 사용하지 않도록 지시
app.config['SEND_FILE_MAX_AGE_DEFAULT']=0

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

if __name__ == "__main__":
    # 서버 실행
    app.run(host='0.0.0.0', port=8080, debug=True)

```

mqtt.py (센서로부터 1초 간격으로 값을 읽어 mosquitto로 전송하는 코드)

```

import io, time
import paho.mqtt.client as mqtt
import control
import base64
import lumi
import temp
import led

```

```

# CCTV 가 현재 작동 중인지 확인하는 변수
isStart = False

def on_connect(client, userdata, flag, rc):
    client.subscribe("led", qos = 0) # "led" 토픽으로 구독 신청

def on_message(client, userdata, msg) :
    global isStart
    # CCTV 작동 중일 경우 True, 작동 중이지 않을 경우 False
    if msg.payload.decode('utf-8') == 'start':
        isStart = True
    elif msg.payload.decode('utf-8') == 'stop':
        isStart = False
    elif int(msg.payload) == 1 or int(msg.payload) == 0:
        led.controlLED(int(msg.payload)) # LED 를 켜거나 끄

ip = "localhost" # 현재 브로커는 이 컴퓨터에 설치되어 있음

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

client.connect(ip, 1883) # 브로커에 연결
client.loop_start() # 메시지 루프를 실행하는 스레드 생성

# CCTV 가 실행 중일 경우
# 병렬적으로 1 초 단위로 센서로부터 값을 읽어 전송하는 무한 루프 실행
while True:
    if isStart == True:
        # 센서로부터 거리, 조도, 온도, 습도를 읽어
        # 각각의 토픽에 맞게 값을 전송
        distance = control.measure_distance()
        client.publish("ultrasonic", distance, qos=0)
        luminance = lumi.getLuminance()
        client.publish("luminance", luminance, qos=0)
        temperature = temp.getTemperature(temp.sensor)
        client.publish("temperature", temperature, qos=0)
        humidity = temp.getHumidity(temp.sensor)
        client.publish("humidity", humidity, qos=0)

        # 거리가 30 이하일 경우 빨간색 LED 를 켜
        if(distance <= 30):
            led.ledOnOff(led.redLed, 1)
            led.ledOnOff(led.greenLed, 0)
        # 거리가 30 초과일 경우 초록색 LED 를 켜
        else :
            led.ledOnOff(led.redLed, 0)

```

```

        led.ledOnOff(led.greenLed, 1)
        time.sleep(1) # 1 초 동안 잠자기
client.loop_stop() # 메시지 루프를 실행하는 스레드 종료
client.disconnect()

```

imageMqtt.py (카메라로 1초 간격으로 사진을 촬영해 mosquitto로 전송하는 코드)

```

#publisher

import io, time
import cv2
from PIL import Image, ImageFilter
import paho.mqtt.client as mqtt
import cv2
import camera
import base64
import classifier

isStart = False

def on_connect(client, userdata, msg, rc):
    client.subscribe("camera")

def on_message(client, userdata, msg):
    global isStart
    # 촬영을 시작하라는 메시지를 받을 경우
    if msg.payload.decode('utf-8') == 'start':
        isStart = True
    # 촬영을 멈추라는 메시지를 받을 경우
    else:
        isStart = False
    pass

broker_ip = "localhost"

client = mqtt.Client()
client.connect(broker_ip, 1883) # 1883 포트로 mosquitto 에 접속
client.on_connect = on_connect
client.on_message = on_message

client.loop_start() # 메시지 루프를 실행하는 스레드 생성

camera.init() # 카메라 초기화
stream = io.BytesIO()

```

```

while True:
    # isStart 참일 경우 실행
    if isStart == True:
        # 사진을 촬영하고 분류기에서 얼굴을 인식한 이미지를 받음
        frame = camera.take_picture(); frame = classifier.classify(frame)
        stream.seek(0) # stream 파일의 커서를 맨 앞으로 이동한다
        image = Image.fromarray(frame) # numpy array 데이터를 PILLOW의 image
        데이터 포맷으로 변환
        image.save(stream, format='JPEG') # 이미지 변환후 JPEG 형식으로 이미지를
        저장

        client.publish("image", stream.getvalue(), qos=0) # image 토픽으로
        이미지 데이터 전송
        stream.truncate() # stream 파일에 있는 모든 내용을 지운다
    else:
        time.sleep(1)

camera.release() # 카메라 사용 끝내기
client.loop_stop() # 메시지 루프를 실행하는 스레드 종료
client.disconnect()

```

mqttio.js (paho 라이브러리를 사용하여 MQTT 브로커에 접속하는 자바스크립트)

```

let client = null; // MQTT 클라이언트의 역할을 하는 Client 객체를 가리키는
전역변수
let connectionFlag = false; // 연결 상태이면 true
const CLIENT_ID = "client-
"+Math.floor((1+Math.random())*0x10000000000).toString(16) // 사용자 ID 랜덤
생성

function connect() { // 브로커에 접속하는 함수
    if(connectionFlag == true)
        return; // 현재 연결 상태이므로 다시 연결하지 않음
    // 사용자가 입력한 브로커의 IP 주소와 포트 번호 알아내기
    let broker = document.getElementById("broker").value; // 브로커의 IP 주소
    let port = 9001 // mosquitto를 웹소켓으로 접속할 포트 번호

    // MQTT 메시지 전송 기능을 모두 가진 Paho client 객체 생성
    client = new Paho.MQTT.Client(broker, Number(port), CLIENT_ID);

    // client 객체에 콜백 함수 등록 및 연결
    client.onConnectionLost = onConnectionLost; // 접속 끊김 시 onConnectionLost()
    실행
    client.onMessageArrived = onMessageArrived; // 메시지 도착 시
    onMessageArrived() 실행

```

```

// client 객체에게 브로커에 접속 지시
client.connect({
    onSuccess:onConnect, // 브로커로부터 접속 응답 시 onConnect() 실행
});
}

// 브로커로의 접속이 성공할 때 호출되는 함수
function onConnect() {
    document.getElementById("messages").innerHTML = '<span><b>CCTV 와  
연결</b>되었습니다.' + '</span><br/>';
    connectionFlag = true; // 연결 상태로 설정
}

function subscribe(topic) {
    if(connectionFlag != true) { // 연결되지 않은 경우
        alert("연결되지 않았음");
        return false;
    }

    // 구독 신청하였음을 <div> 영역에 출력
    document.getElementById("messages").innerHTML = '<span><b>CCTV 작동</b>을  
<b>시작</b>합니다.</span><br/>';
    // 브로커에 구독 신청
    client.subscribe(topic);
    client.subscribe("temperature");
    client.subscribe("luminance");
    client.subscribe("humidity");
    client.send("led", "start", 0, false);
}

function publish(topic, msg) {
    if(connectionFlag != true) { // 연결되지 않은 경우
        alert("연결되지 않았음");
        return false;
    }
    // 메시지 전송
    client.send(topic, msg, 0, false);
    return true;
}

function unsubscribe(topic) {
    if(connectionFlag != true) return; // 연결되지 않은 경우
    // CCTV 작동 중단 메시지 출력
    document.getElementById("messages").innerHTML = '<span><b>CCTV 작동</b>을  
<b>중단</b>합니다.</span><br>';
    document.getElementById("childInfo").innerHTML = "";
    // 브로커에 구독 신청 취소
    client.unsubscribe(topic);
}

```

```

client.unsubscribe("luminance");
client.unsubscribe("temperature");
client.unsubscribe("humidity");
client.send('led', '0', 0, false);
client.send('led', 'stop', 0, false);
client.send('camera', 'stop', 0, false);
clear(); clear();
}

// 접속이 끊어졌을 때 호출되는 함수
function onConnectionLost(responseObject) {
    if (responseObject.errorCode !== 0) {
        document.getElementById("messages").innerHTML = '<span>오류 : ' +
            responseObject.errorMessage + '</span><br/>';
    }
    connectionFlag = false; // 연결 되지 않은 상태로 설정
}

// 메시지가 도착할 때 호출되는 함수
function onMessageArrived(msg) { // 매개변수 msg 는 도착한 MQTT 메시지를 담고 있는
객체
    // 토픽이 "ultrasonic"일 경우
    if(msg.destinationName == "ultrasonic"){
        let distance = parseInt(msg.payloadString);
        // 거리가 30 이하일 경우 카메라를 실행시키고 위험하다는 정보 출력
        if(distance <= 30) {
            startCamera();
            document.getElementById("childInfo").innerHTML = "<span>아이가<b>
위험</b>합니다!</span>";
        }
        // 거리가 30 초과일 경우 안전하다는 정보 출력
        else {
            stopCamera();
            document.getElementById("childInfo").innerHTML = "<span>아이는
<b>안전</b>합니다.</span>";
        }
        // 차트에 추가
        addChartData(parseFloat(msg.payloadString));
    }
    // 토픽이 "temperature"일 경우
    else if(msg.destinationName == "temperature"){
        let temp = parseInt(msg.payloadString);
        let tempImg = document.getElementById('tempImg');
        // 현재 온도에 맞게 이미지 출력
        if(temp >= 28)
            tempImg.src = "static/hot.png";
        else if(temp <= 18)
            tempImg.src = "static/cold.png";
    }
}

```

```

        else
            tempImg.src = "static/averagetemp.png";
            tempImg.title = '온도: ' + temp;
        }
        // 토픽이 "luminance"일 경우
        else if(msg.destinationName == "luminance"){
            let lumi = parseInt(msg.payloadString);
            let lumiImg = document.getElementById('lumiImg');
            // 현재 조도에 맞게 이미지 출력
            if(lumi <= 60)
                lumiImg.src = "static/moon.png";
            else
                lumiImg.src = "static/sun.png";
            lumiImg.title = '밝기: ' + lumi;
        }
        // 토픽이 "humidity"일 경우
        else if(msg.destinationName == "humidity"){
            let humid = parseInt(msg.payloadString);
            let humidImg = document.getElementById('humidImg');
            // 현재 습도에 맞게 이미지 출력
            if(humid >= 60)
                humidImg.src = "static/humid.png";
            else if(humid <= 40)
                humidImg.src = "static/lowhumid.png";
            else
                humidImg.src = "static/averagehumid.png";
            humidImg.title = '습도: ' + humid;
        }
        // 토픽이 "image"일 경우
        else if(msg.destinationName == "image") {
            drawImage(msg.payloadBytes); // 메시지에 담긴 파일 이름으로 drawImage()
호출
        }
    }
}

// disconnection 버튼이 선택되었을 때 호출되는 함수
function disconnect() {
    if(connectionFlag == false)
        return; // 연결 되지 않은 상태이면 그냥 리턴
    document.getElementById("messages").innerHTML = '<b>CCTV 연결</b>을<br>종료</b>합니다.</span><br>';
    document.getElementById("childInfo").innerHTML = "";
    connectionFlag = false; // 연결 되지 않은 상태로 설정
    client.send('led', "0", 0, false); // led 를 끄도록 메시지 전송
    client.send('led', "stop", 0, false);
    clear();
    client.disconnect(); // 브로커와 접속 해제
    client.unsubscribe('led');
}

```

```
}
```

chart.js (웹 페이지에 차트를 그리는 코드)

```
let ctx = null;
let chart = null;
let config = {
  // type 은 차트 종류 지정
  type: 'line', // 라인그래프
  // data 는 차트에 출력될 전체 데이터 표현
  data: {
    // labels 는 배열로 데이터의 레이블들
    labels: [],
    // datasets 배열로 이 차트에 그려질 모든 데이터 셋 표현. 그래프 1 개만 있음
    datasets: [{
      label: '아이와 위험 물체 간 실시간 거리',
      borderColor: 'brown',
      borderWidth: 2,
      data: [], // 각 레이블에 해당하는 데이터
      fill : false, // 채우지 않고 그리기
    }]
  },
  // 차트의 속성 지정
  options: {
    responsive : false, // 크기 조절 금지
    scales: { // x 축과 y 축 정보
      xAxes: [{
        display: true,
        scaleLabel: { display: true, labelString: '시간(초)' },
      }],
      yAxes: [{
        display: true,
        scaleLabel: { display: true, labelString: '거리(cm)' }
      }]
    }
  }
};

let LABEL_SIZE = 20; // 차트에 그려지는 데이터의 개수
let tick = -1; // 도착한 데이터의 개수임, tick 의 범위는 0 에서 99 까지만

// 차트 생성
function drawChart() {
  ctx = document.getElementById('chartCanvas').getContext('2d');
  chart = new Chart(ctx, config);
  init();
}
```



```

}

// 차트 초기화
function init() { // chart.data.labels의 크기를 LABEL_SIZE로 만들고 0~19까지 레이블 붙이기
    for(let i=0; i<LABEL_SIZE; i++) {
        chart.data.labels[i] = i;
    }
    chart.update();
}

function addChartData(value) {
    tick++; // 도착한 데이터의 개수 증가
    tick %= 100; // tick의 범위는 0에서 99까지만. 100보다 크면 다시 0부터 시작
    let n = chart.data.datasets[0].data.length; // 현재 데이터의 개수
    if(n < LABEL_SIZE) // 현재 데이터 개수가 LABEL_SIZE보다 작은 경우
        chart.data.datasets[0].data.push(value);
    else { // 현재 데이터 개수가 LABEL_SIZE를 넘어서는 경우
        // 새 데이터 value 삽입
        chart.data.datasets[0].data.push(value); // value를 data[]의 맨 끝에
        추가
        chart.data.datasets[0].data.shift(); // data[]의 맨 앞에 있는 데이터
        제거

        // 레이블 삽입
        chart.data.labels.push(tick); // tick 값을 labels[]의 맨 끝에 추가
        chart.data.labels.shift(); // labels[]의 맨 앞에 있는 값 제거
    }
    chart.update();
}

// 차트 캔버스 보이기 숨기기
function chartHideshow() {
    let canvas = document.getElementById('chartCanvas'); // canvas DOM 객체
    알아내기
    if(canvas.style.display == "none") // canvas 객체가 보이지 않는다면
        canvas.style.display = "inline-block"; // canvas 객체를 보이게 배치
    else
        canvas.style.display = "none" ; // canvas 객체를 보이지 않게 배치
}

window.addEventListener("load", drawChart); // Load 이벤트가 발생하면
drawChart() 호출하도록 등록

```

image.js (웹 페이지에 이미지를 출력하는 코드)

```

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

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

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

// 바이트 코드를 base64 로 인코딩
function bytes2base64( bytes ) {
    var binary = '';
    var bytes = new Uint8Array( bytes );
    var len = bytes.byteLength;
    for (var i = 0; i < len; i++) {
        binary += String.fromCharCode( bytes[ i ] );
    }
    return window.btoa( binary );
}

// drawImage()는 "image" 토픽이 도착하였을 때 onMessageArrived()에 의해 호출된다.
function drawImage(bytes) { // imgUrl 은 이미지의 url
    img.src = "data:image/jpeg;base64," + bytes2base64(bytes);
}

var isImageSubscribed = false;
function startCamera() {
    // 토픽 'image'를 구독하지 않았을 경우 구독
    if(!isImageSubscribed) {
        subscribe('image');
        isImageSubscribed = true;
    }
    publish('camera', 'start'); // 토픽: camera, start 메시지 전송
}

function stopCamera() {
    clear();
    clear();
    // 토픽 'image'를 구독했을 경우 구독 해제
    if(isImageSubscribed) {

```

```

        unsubscribe('image');
        isImageSubscribed = false;
    }
    publish('camera', 'stop'); // 토픽: camera, stop 메시지 전송
}

// 카메라 캔버스 지우기
function clear() {
    context.clearRect(0, 0, canvas.width, canvas.height);
}

// 카메라 캔버스 보이기 숨기기
function cameraHideshow() {
    let canvas = document.getElementById('cameraCanvas'); // canvas DOM
    객체 알아내기
    if(canvas.style.display == "none") // canvas 객체가 보이지 않는다면
        canvas.style.display = "inline-block"; // canvas 객체를 보이게
        배치
    else
        canvas.style.display = "none" ; // canvas 객체를 보이지 않게
        배치
}

```

cctv.html (웹 페이지 코드)

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>홈 CCTV 시스템</title>
    <!-- 라이브러리 코드 및 자바스크립트 코드-->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-
mqtt/1.0.2/mqttws31.min.js" type="text/javascript"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js "
type="text/javascript"></script>
    <script src="./static/image.js" type="text/javascript"></script>
    <script src="./static/chart.js" type="text/javascript"></script>
    <script src="./static/mqttio.js" type="text/javascript"></script>
    <script>
        window.addEventListener("load", function () {
            // http://224..129:8080/에서 224...의 IP 만 끌어내는 코드
            let url = new String(document.location);
            ip = (url.split("//"))[1]; // ip = "224...:8080/"
            ip = (ip.split(":"))[0]; // ip = "224..."

```

```

        document.getElementById("broker").value = ip
    });
</script>
<!-- 스타일 코드 -->
<style>
    #body {
        background-color: beige;
    }
    #title {
        color: saddlebrown;
        text-align: center;
    }
    #label, #messages, #childInfo {
        color:saddlebrown;
    }
    #line {
        height: 10px;
        background-color: burlywood;
        border-color:burlywood;
    }
</style>
</head>
<body id="body">
    <h1 id="title">HOME CCTV SERVICE</h1>
    <hr id="line">
    <h2 id="label">Connect CCTV</h2>
    <!-- 브로커 ip 와 포트 번호를 입력해 연결하거나 연결 해제-->
    <form id="connection-form">
        브로커 IP:
        <input id="broker" type="text" name="broker" value=""><br>
        포트 번호 : 9001<br>
        <input type="button" onclick="connect()" value="Connect">
        <input type="button" onclick="disconnect()" value="Disconnect">
    </form> <div id="messages"></div>

    <h2 id="label">Home Information</h2>
    <!-- 버튼을 눌러 구독하거나 구독을 해제해 CCTV 연결을 제어-->
    <form id="subscribe-form">
        <input type="button" onclick="subscribe('ultrasonic')" value="on">
        <input type="button" onclick="unsubscribe('ultrasonic')"
value="off">
    </form>
    <!-- 이미지 출력-->
    <img src="" id="tempImg" alt="">
    <img src="" id="lumiImg" alt="">
    <img src="" id="humidImg" alt=""><br>
    <!-- 아이의 현재 정보 출력-->
    <div id="childInfo"></div><br>

```

```

<!-- 카메라 보기/숨기기 버튼과 카메라를 출력할 캔버스 -->
<button id="hideshow" onclick="cameraHideshow()">camera
show/hide</button><br>
<canvas id="cameraCanvas" width="320" height="240" style="display:
none;"></canvas><br>

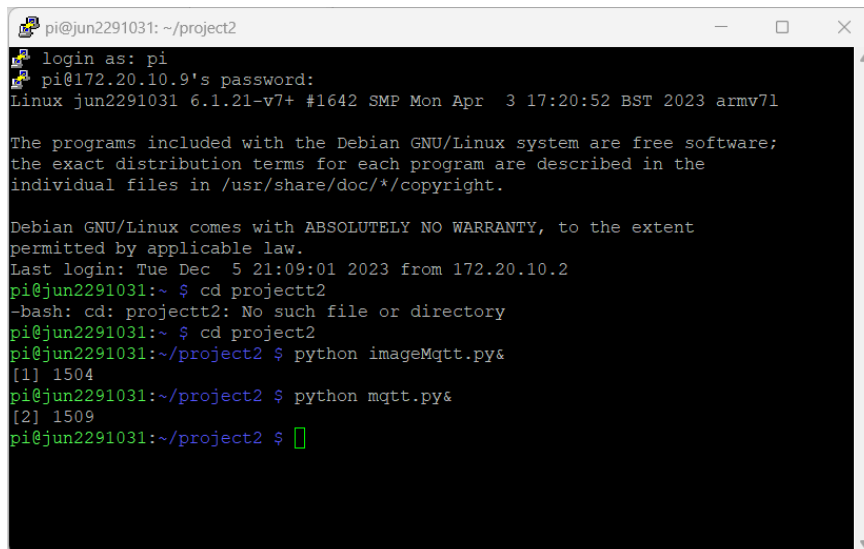
<!-- 차트 보기/숨기기 버튼과 차트를 출력할 캔버스 -->
<button id="hideshow" onclick="chartHideshow()">chart
show/hide</button><br>
<canvas id="chartCanvas" width="600" height="400" style="display:
none;"></canvas>

</body>
</html>

```

4. 실행 과정 및 결과

1) mqtt.py와 imageMqtt.py를 백그라운드에서 실행시키고 cctvapp.py 플라스크 앱을 실행시킨다.



```

pi@jun2291031: ~/project2
login as: pi
pi@172.20.10.9's password:
Linux jun2291031 6.1.21-v7+ #1642 SMP Mon Apr  3 17:20:52 BST 2023 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

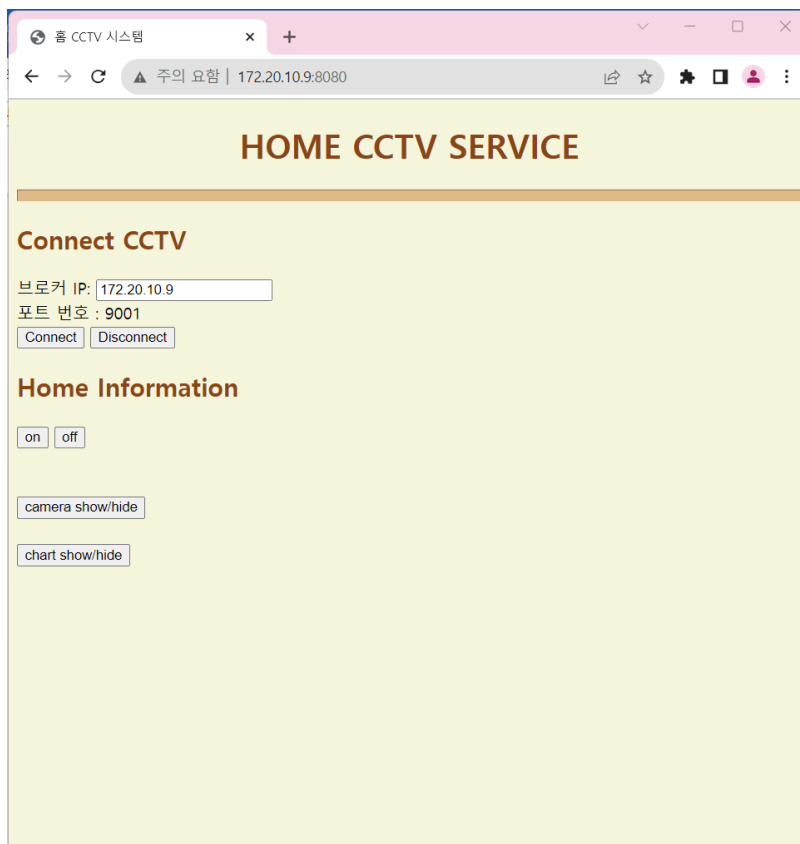
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Dec  5 21:09:01 2023 from 172.20.10.2
pi@jun2291031:~ $ cd projectt2
-bash: cd: projectt2: No such file or directory
pi@jun2291031:~ $ cd project2
pi@jun2291031:~/project2 $ python imageMqtt.py&
[1] 1504
pi@jun2291031:~/project2 $ python mqtt.py&
[2] 1509
pi@jun2291031:~/project2 $ 

```

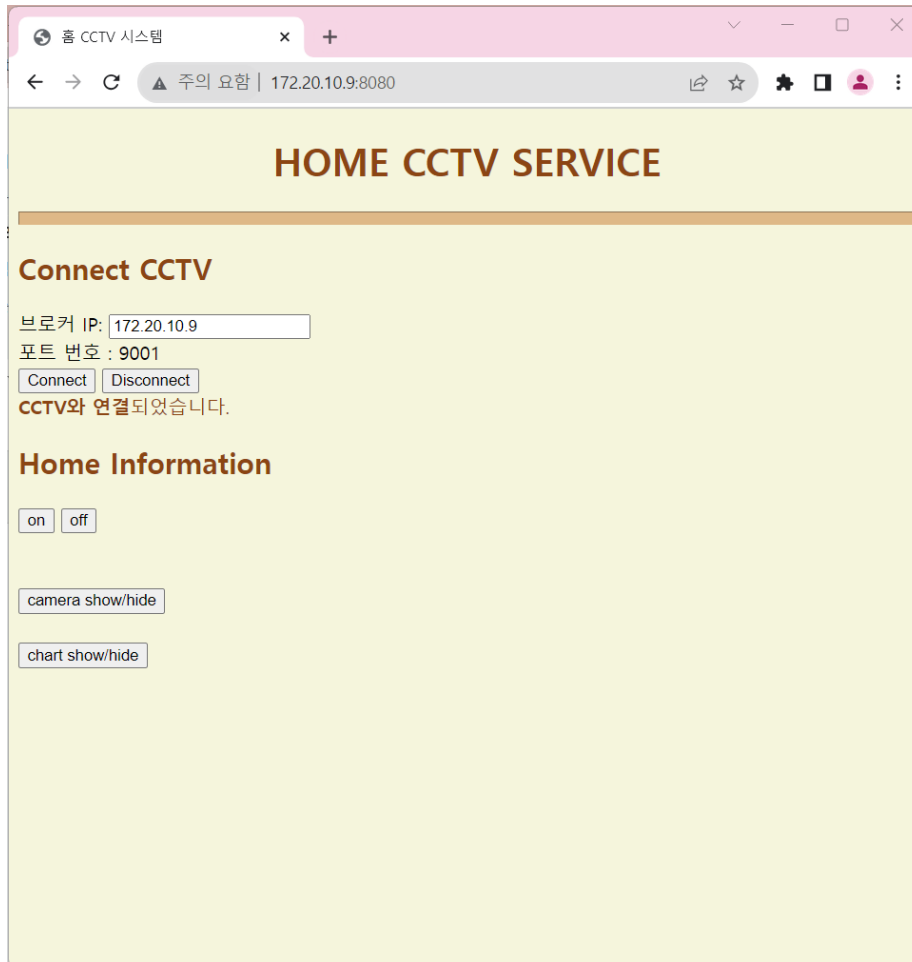
```
pi@jun2291031: ~/project2
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Nov 27 23:46:36 2023
pi@jun2291031:~ $ cd project2
pi@jun2291031:~/project2 $ python cctvapp.py
* Serving Flask app "cctvapp" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 225-893-293
172.20.10.2 - - [05/Dec/2023 21:18:00] "GET / HTTP/1.1" 200 -
172.20.10.2 - - [05/Dec/2023 21:18:00] "GET /static/image.js HTTP/1.1" 304 -
172.20.10.2 - - [05/Dec/2023 21:18:00] "GET /static/chart.js HTTP/1.1" 304 -
172.20.10.2 - - [05/Dec/2023 21:18:00] "GET /static/mqttio.js HTTP/1.1" 304 -
172.20.10.2 - - [05/Dec/2023 21:18:01] "GET /favicon.ico HTTP/1.1" 404 -
```

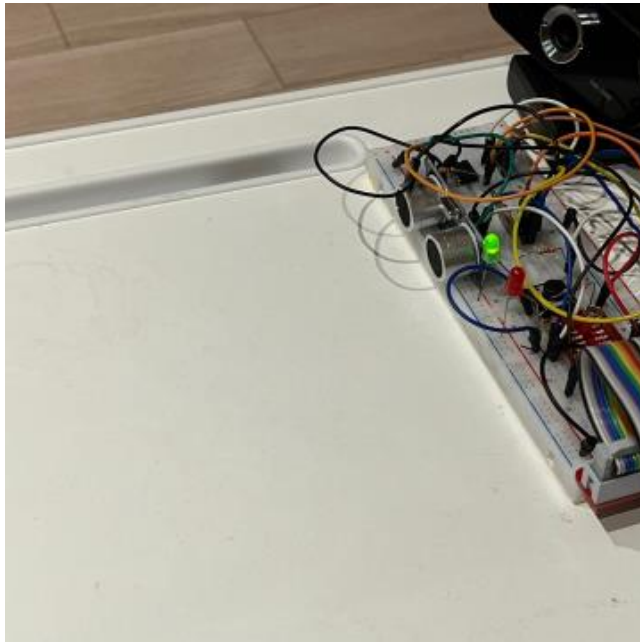
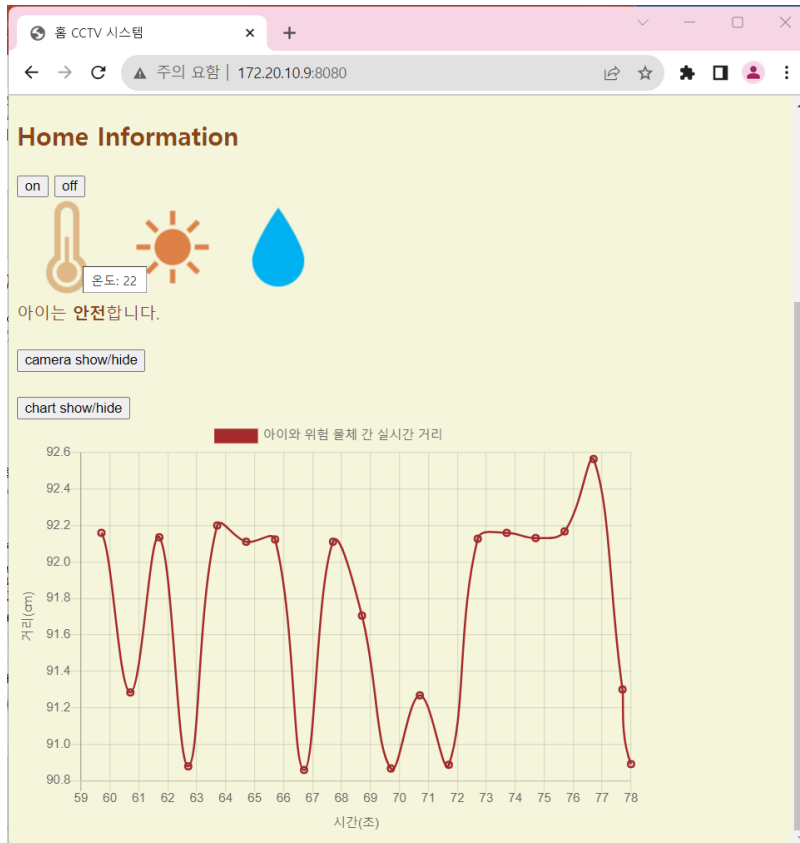
2) 웹 페이지에서 172.20.10.9:8080으로 접속한다.



3) connect 버튼을 눌러 mosquitto에 접속하면 CCTV와 연결되었다는 메시지가 출력된다.

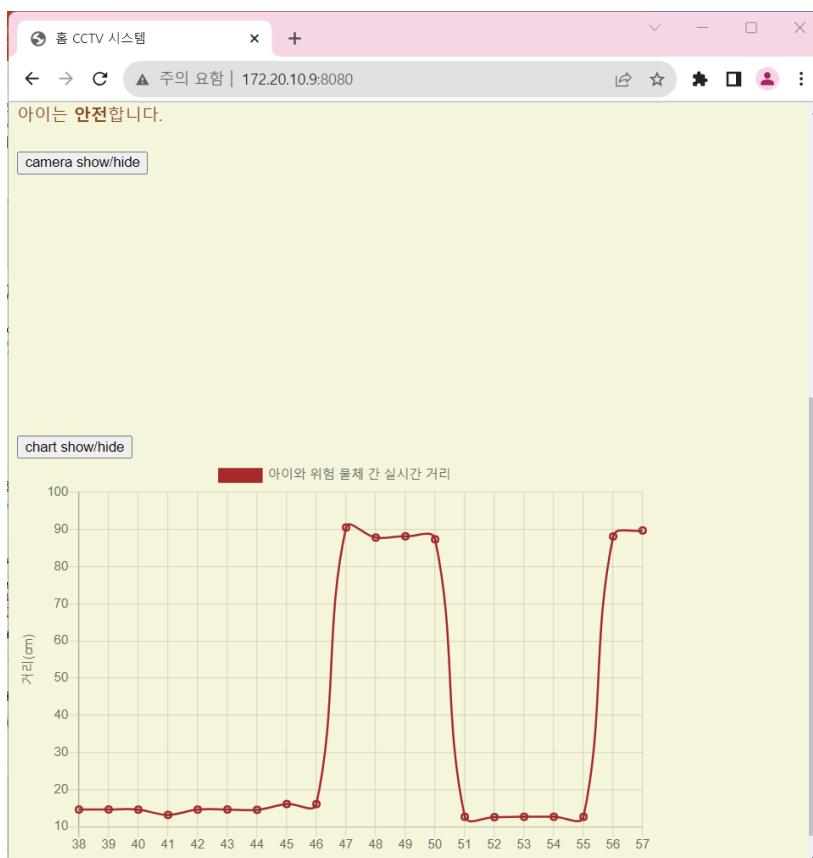
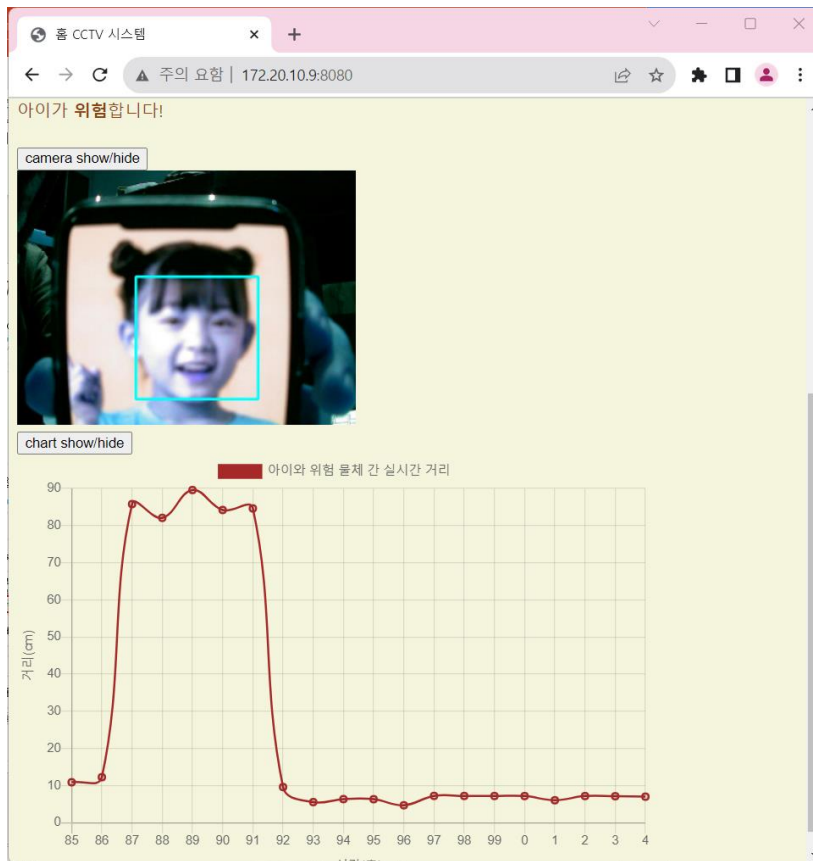


4) on 버튼을 누르면 CCTV 작동을 시작한다는 메시지를 출력하고 토픽 구독을 통해 센서들로부터 값을 읽어온다. 각 아이콘에 마우스를 올리면 현재 온도/조도/습도 값을 확인할 수 있다. chart show/hide 버튼을 누르면 그래프를 보거나 숨길 수 있다. 그래프에서는 현재 초음파 센서로부터 읽어오는 거리 값을 그래프로 확인할 수 있다. 현재 아이가 30cm보다 멀리 있을 경우 "아이는 안전합니다."라는 메시지를 출력하고 초록색 LED를 켤다.

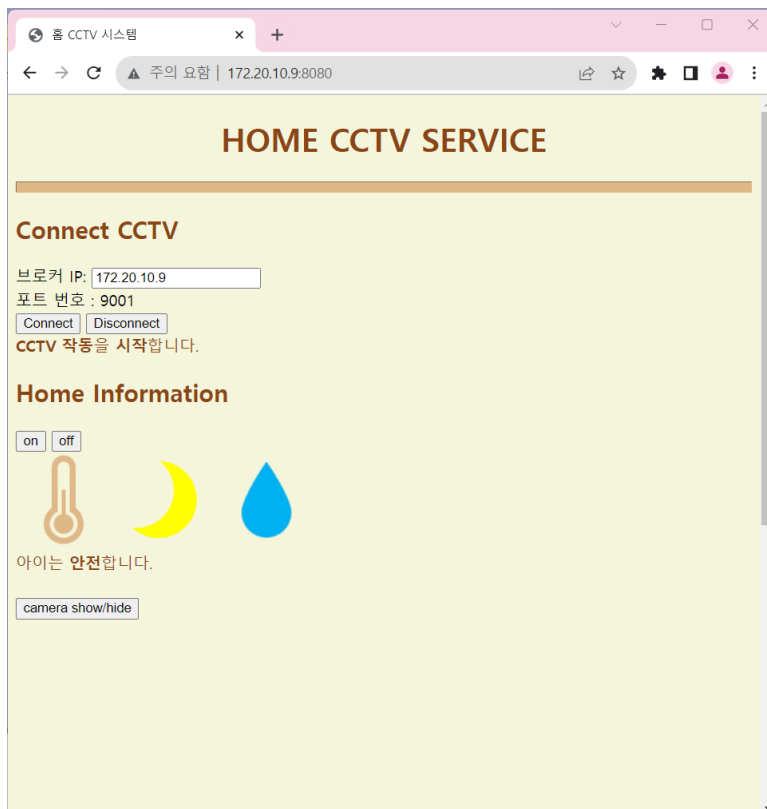
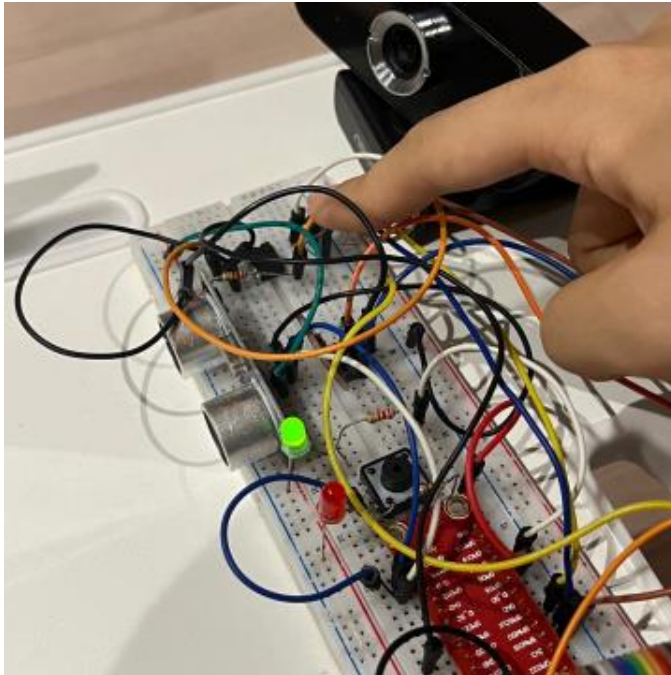


5) 아이가 30cm이내에 있을 경우에는 카메라로 촬영하고 얼굴 인식 모델을 활용해 아이 얼굴에 사각형을 그린 뒤 웹 페이지에 실시간으로 출력하고, 빨간색 LED를 켜다. 그리고 웹 페이지에 "아이가 위험합니다!" 메시지를 출력한다. camera show/hide 버튼을 누르면 카메라 화면을 보거나 숨길 수 있다. 아이가 30cm보다 멀리 떨어질 경우 카메라 작동을 멈추고 출력되었던 화면을 없앤

다.



6) 조도 센서를 손으로 가릴 경우 그에 맞게 웹 페이지의 조도 이미지를 바꾼다.



7) off 버튼을 누르면 CCTV 작동을 중단한다는 메시지가 출력된다. Disconnect 버튼을 누르면

CCTV 연결을 종료한다는 메시지를 출력한다. off버튼이나 disconnect 버튼을 누르면 토픽 구독을 해제하고 브레드보드에 켜져 있던 LED의 불을 끈다.



5. 결론

모바일&스마트시스템 강의를 통해 처음 라즈베리파이를 접하게 되어 초반 4주 정도는 수업에 잘 따라가지 못하고 많이 헤매었습니다. 그러나 집에서 코드를 복습하면서 다시 작성해보니 명확하게 이해할 수 있었고 이를 토대로 이후 수업 시간에도 내용을 잘 따라갈 수 있게 되었습니다.

프로젝트 계획서를 쓸 당시에 기능들을 잘 구현할 수 있을 지에 대한 걱정을 많이 했습니다. 그렇지만 수업을 열심히 듣고 강의에서 배운 코드를 활용하며 프로젝트를 진행하니 예상했던 것보다 수월하게 프로젝트를 제작할 수 있었습니다. 특히, 구현하고 싶은 내용을 그림으로 정리하는 것이 프로젝트 제작에 큰 도움이 되었다고 생각합니다. 프로젝트에 추가하고 싶었던 기능을 하나씩 구현하면서 뿌듯함과 자신감을 얻을 수 있었습니다. 또한, 모바일 트랙 전공필수 수업이지만 JavaScript, HTML, CSS 코드 작성 경험을 쌓으면서 2트랙인 웹공학에 대한 이해도도 높일 수 있어 일석이조의 수업이었다고 생각합니다. 이번 학년 가장 기억에 남는 수업이 될 것 같습니다.

한 학기동안 강의하시느라 수고 많으셨습니다. 감사합니다.