미니프로젝트 결과 보고서

자취생을 위한 스마트 홈 IOT 보안시스템

• 분반	모바일 & 스마트시스템[7]
• 학번	2271387
• 제출일	2023. 12. 09 (토)
• 소속	웹공학트랙
• 이름	서준영

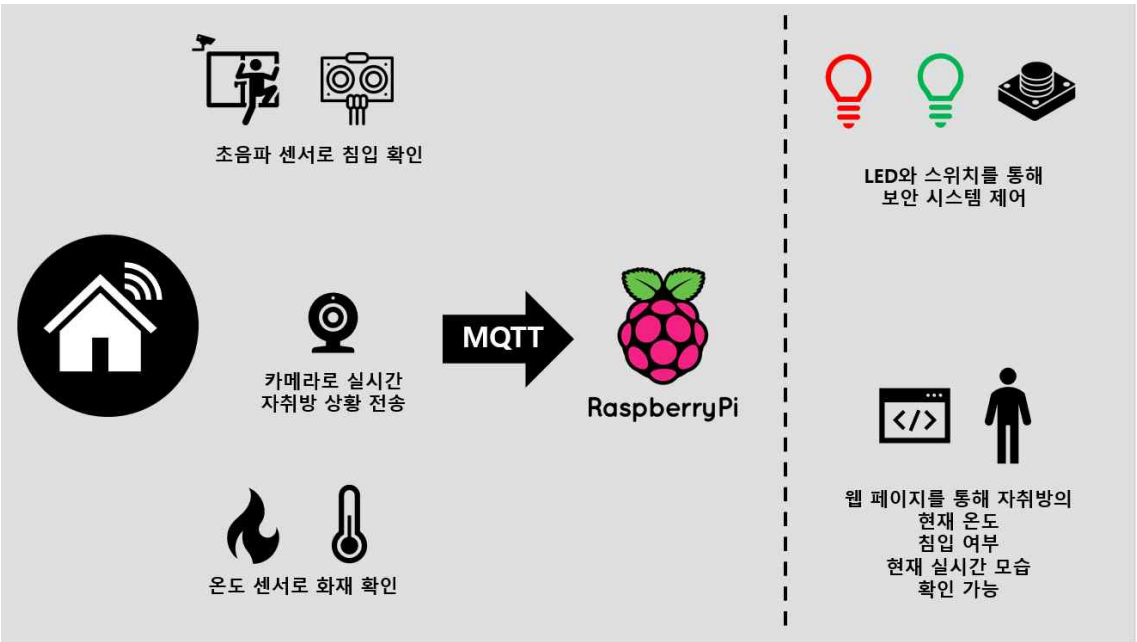
1. 시스템 개요 및 설명

자취를 하는 사람들은 대부분 도둑, 화재에 대한 걱정이 많다. 혼자 살기 때문에 집을 비운 동안 누가 들어오지는 않을지, 불이 나지는 않을지 항상 불안해하곤 한다. 이런 불안을 해소하고자 '자취생을 위한 스마트 홈 IOT 보안 시스템'을 제작했다.

이 시스템은, 초음파 센서를 통해 외부인의 침입을 감지하고, 온도 센서를 통해 화재를 감지해 웹 페이지에 알림을 보낸다. 사용자는 웹 페이지의 알림을 확인하고, 카메라를 통해 실시간 내 자취방의 상황을 확인할 수 있다. 또, 사용의 편리함을 위해 LED와 스위치를 통해 시스템을 제어할 수 있게 제작되었다.

사용자는 웹 페이지에 접속해서 Connect 버튼을 누르면 자동으로 연결되고, 시스템은 상황에 따라 사용자에게 알림을 보낸다. 집에 도둑이 들었다고 판단하면 도둑 그림이, 화재가 났다고 판단되면 불 그림이 웹 페이지에 나타난다. 특히 도둑이 들었을 경우에는 추후 경찰에 신고해야 하기 때문에 자동으로 카메라가 사진을 촬영한다. 도둑이 든 시간을 알기 위해 파일의 이름을 현재 날짜와 시간으로 설정한다. 알림을 받은 사용자는 "실시간 영상 보기" 버튼을 통해 실시간 자취방의 모습을 확인할 수 있다. 또, "온도 보기" 버튼을 통해 실시간 방의 온도를 확인할 수도 있다.

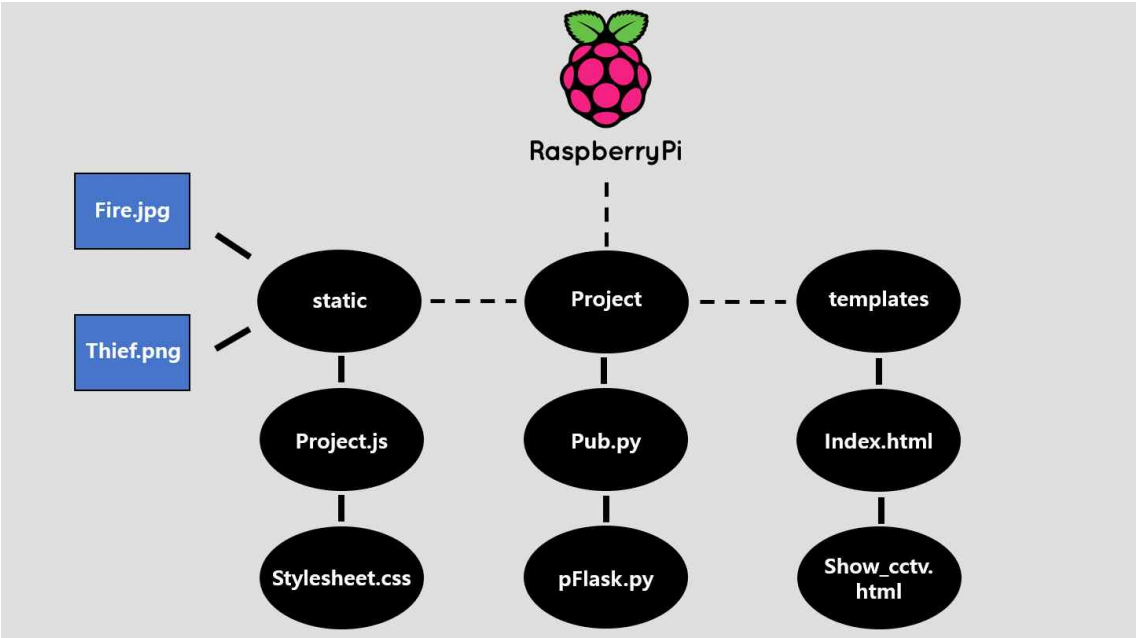
초록색 LED에 불이 들어오면 이 시스템이 정상적으로 작동함을 의미한다. 스위치를 누르면 초록색 LED의 불은 꺼지고, 빨간색 LED가 점등된다. 이 때는 시스템이 작동을 멈추고 감지하지 않는다. 사용자가 집에 들어와서 더 이상 자취방을 감시할 필요가 없을 때를 가정하고 제작됐다.



[그림 1]

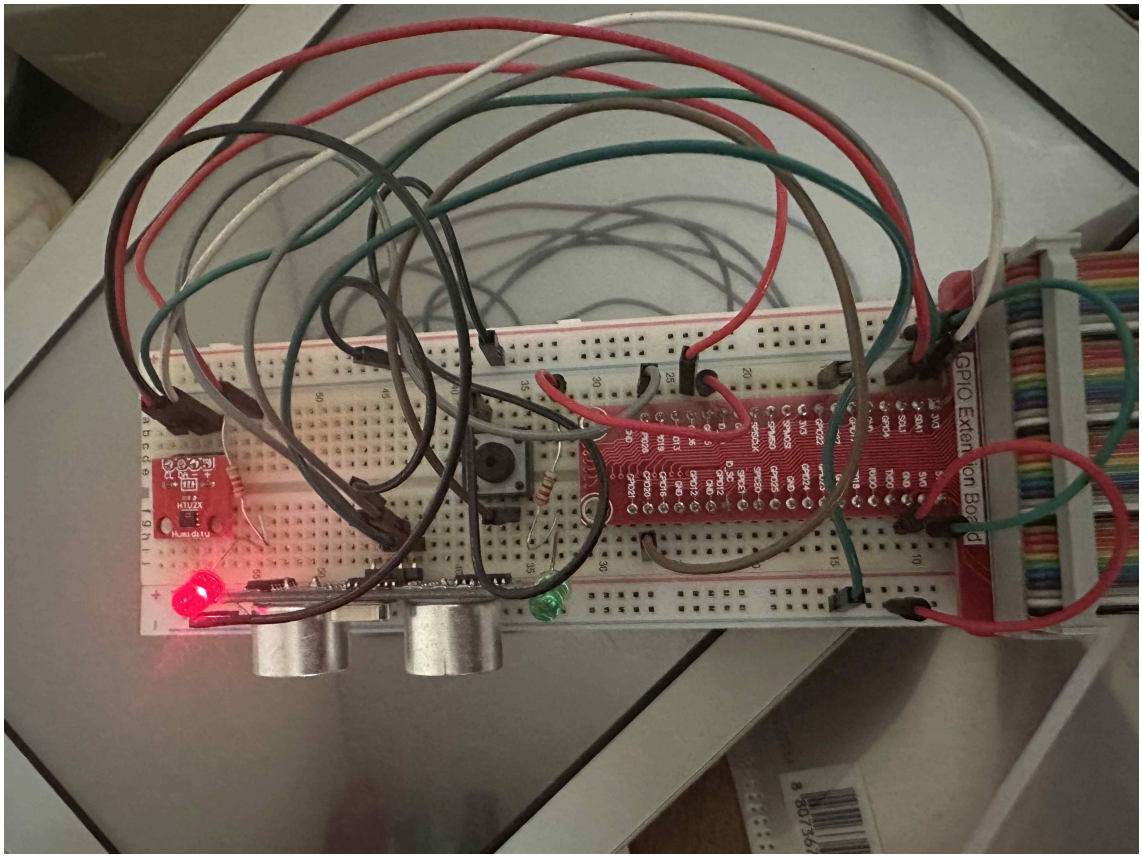
그림 1을 통해 이 시스템의 대략적인 개요를 파악할 수 있다.

2. 시스템 구조



[그림 2]

이 시스템은 그림 2와 같은 구조로 제작됐다. 메인이 되는 project 디렉토리에는 전체 센서를 제어하고 그 값을 받아 publishing하는 Pub.py 파일과 웹 페이지를 구성하는 pFlask.py 파일이 있다. pFlask.py를 통해 웹 페이지를 구성하는데 필요한 js파일과 css 파일은 static 디렉토리에 있다. 또, 웹 페이지에서 사용자에게 알림을 보내는 역할을 하는 Fire.jpg와 Thief.png 파일도 있다. templates 디렉토리에는 메인 웹 페이지의 역할을 하는 index.html, 실시간 CCTV를 보여주는 역할을 하는 Show_cctv.html 파일이 있다.



[그림 3]

그림 3은 이 시스템의 실제 회로 구성 모습이다. 초음파 센서, 온습도 센서가 각각 도둑 침입, 화재 감지를 위해 사용됐으며 제어를 위한 스위치와 상태 확인을 위한 LED 2개도 확인할 수 있다.

시스템 제작을 위한 브레드보드 구조는 다음과 같다.

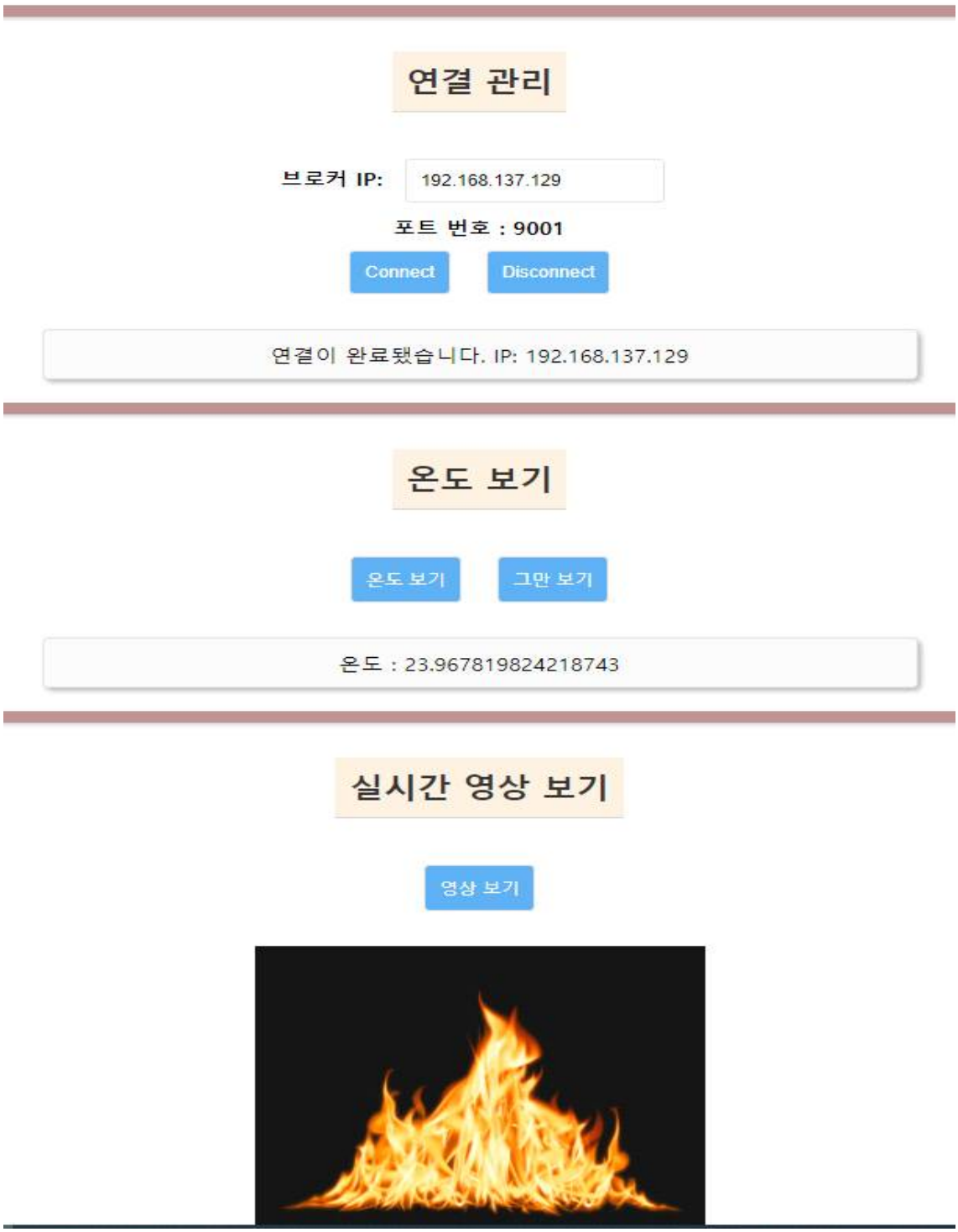
LED	GPIO 5, GPIO 6
초음파 센서	trig=26, echo=27
조도 센서	SPI_CE0(GPIO 8), SPI_MISO(GPIO 9), SPI_MOSI(GPIO 10), SPI_CLK(GPIO 11),
온습도 센서	SDA1(GPIO 2), SCL1(GPIO 3)
카메라	USB 연결
스위치	GPIO 19, GPIO 20

3. 작품 작동 모습

↳ 스마트 홈 IOT 보안 시스템 웹 페이지의 초기 화면 브로커 IP와 포트 번호는 자동으로 입력된다.

↳ “Connect” 버튼을 눌렀을 때의 모습.

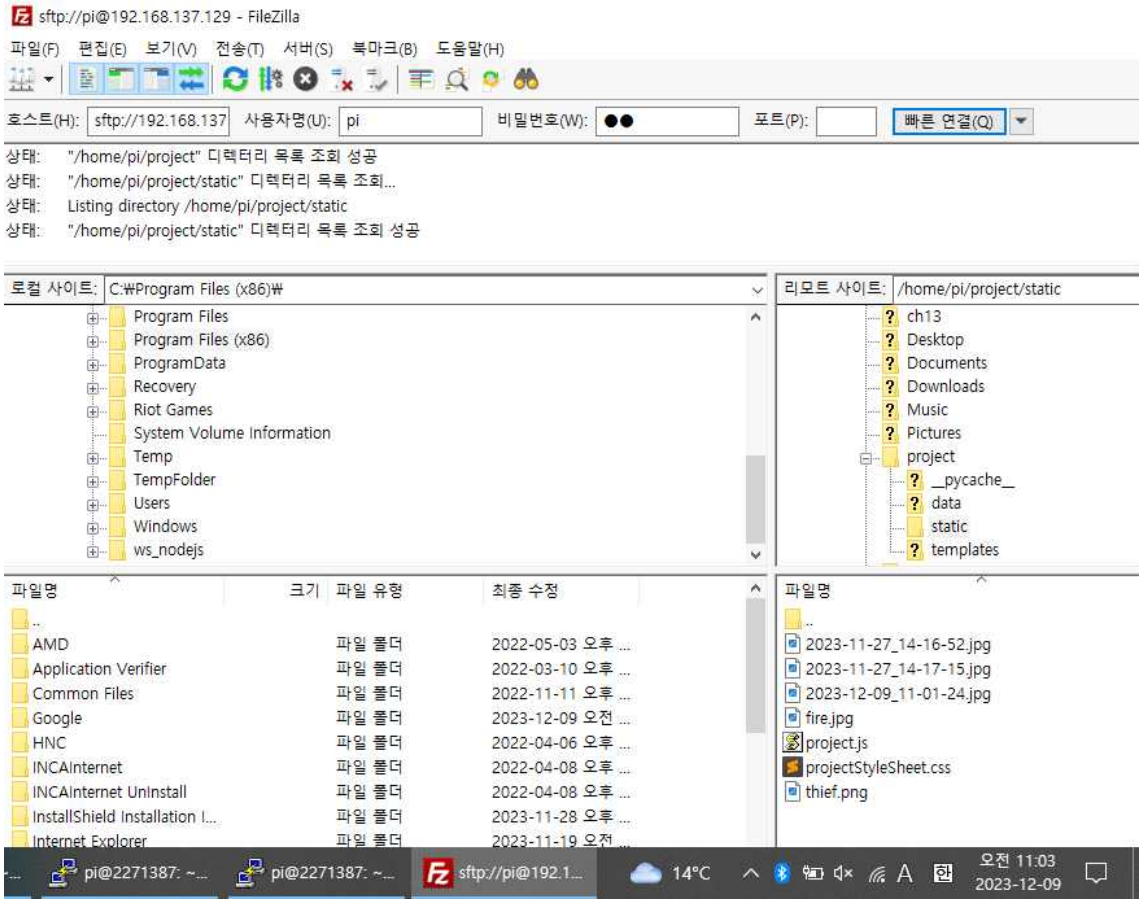
온도 센서는 일정 온도 이상이 측정되는 경우 불이 났다고 판단한다. 보고서 작성을 위해 온도를 섭씨 20도로 설정해 둬서 불이 났다고 판단했다. 실제 사용에는 더 높은 온도를 설정하면 된다.



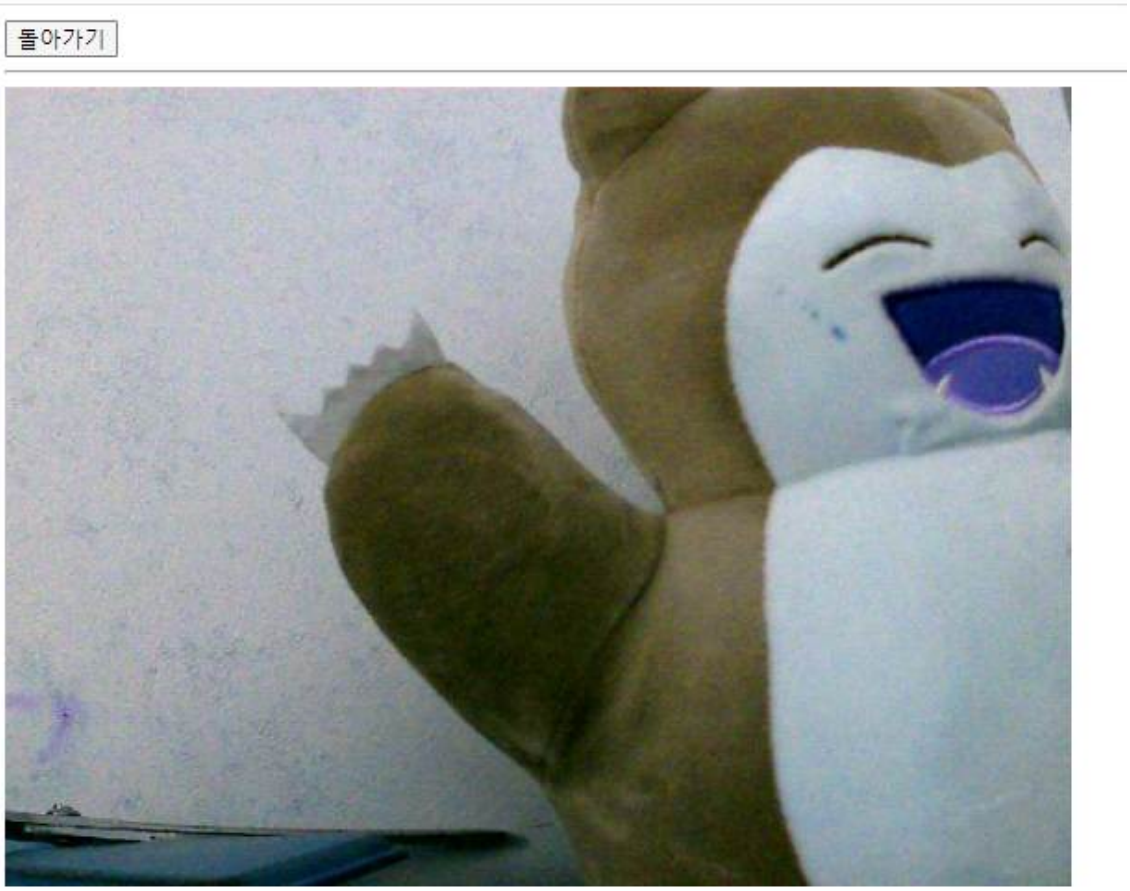
↳ 온도가 20도를 넘어가자, 웹 페이지에 불 그림이 나타난 모습.

초음파 센서는 계속해서 값을 측정하다 측정 값이 10cm 이상 달라지면 문이 열렸다고 판단한다. 측정을 위해 초음파 센서는 문이나 창문 앞에 위치해야 한다. 이 경우 웹 페이지에 도둑 그림이 나타난다.

↳ 온도가 20도를 넘는 상황에서 초음파 센서의 측정값이 변해 도둑이 들었다고 판단되어 웹 페이지에 도둑 그림이 나타난 모습.



↳ 특히 도둑이 들었다고 판단된 경우엔, 그 순간에 자동으로 사진을 찍는다. 경찰에 신고하기 위함이다. FileZilla 클라이언트를 통해 project-static 폴더에 현재 시간을 기준으로 한 사진 파일이 저장되었음을 확인할 수 있다.



↳사용자가 “실시간 영상 보기” 버튼을 누른 모습. 현재 카메라가 비추고 있는 모습을 웹 페이지에서 확인할 수 있다. 인형이 도둑이라고 가정한다면, 도둑이 들었다고 판단해 바로 경찰에 신고할 수 있다.

3. 시스템 제작에 사용한 코드

<pub.py>

이 코드로 각 센서를 제어하고, 값을 publishing 한다.

```
###라즈베리파이 Publisher 코드###
import RPi.GPIO as GPIO
import cv2
import time
import io
from PIL import Image, ImageFilter
import paho.mqtt.client as mqtt
import sys
from adafruit_htu21d import HTU21D
import busio
from datetime import datetime
trig =26  #GPIO 26핀 트리거
echo =27  #GPIO 27핀 에코
sda=2 #온습도센서 I2C
scl=3 #온습도센서 I2C
button=20 #버튼
controlFlag =True #버튼 플래그로 시스템을 제어함
class Led:
        def __init__(self):
                self.red=5
                self.green=6
                GPIO.setup(self.red,GPIO.OUT)
                GPIO.setup(self.green,GPIO.OUT)
        def led_on(self,pin):
                GPIO.output(pin,GPIO.HIGH)
        def led_off(self,pin):
                GPIO.output(pin,GPIO.LOW)
class Button:
        def __init__(self):
                self.led = Led()
                self.status=0 #버튼상태. 0이면 안눌러짐, 1이면 눌러짐.
                GPIO.setup(button, GPIO.IN, GPIO.PUD_DOWN)
                GPIO.add_event_detect(button, GPIO.RISING, callback=self.pressButton,
bouncetime=200)
        def pressButton(self,pin):
                global controlFlag#전역변수로
                if controlFlag:
                        self.led.led_on(self.led.green)
                        self.led.led_off(self.led.red)
                else:
                        self.led.led_off(self.led.green)
                        self.led.led_on(self.led.red)
                controlFlag =not controlFlag  # controlFlag 값을 토글
#초음파센서 제어 클래스
class Sonic:
        def __init__(self):#생성자
```

```
                GPIO.setup(trig, GPIO.OUT) #트리거는 출력용으로 설정 26
                GPIO.setup(echo, GPIO.IN)  #에코는 입력용으로 설정 27
        def measureDistance(self,trig, echo):
                GPIO.output(trig, GPIO.HIGH) # trig = 26 , echo = 27
                time.sleep(0.1)#0.1초 단위로 거리를 측정. 이 코드가 없으면 발사와 동
시에 수신해서 측정이 잘 안됨
                GPIO.output(trig, GPIO.LOW) # trig 핀 신호 High->Low. 초음사 발사
                while(GPIO.input(echo) ==0): # echo 핀 값이 1로 바뀔때까지 루프
                        pass
                # echo 핀 값이 1이면 초음파 발사
                pulseStart = time.time() # 초음파 발사 시간 기록
                while(GPIO.input(echo) ==1): # echo 핀 값이 0이 될때까지 루프
                        pass
                # echo 핀 값이 0이 되면 초음파 수신
                pulseEnd = time.time() # 초음파 돌아 온 시간 기록
                pulseDuration = pulseEnd - pulseStart # 경과 시간 계산
                distance = pulseDuration*340*100/2#cm 단위로
                return distance # 거리 계산하여 리턴(단위 cm)
        def isWhoInvade(self):
                current =self.measureDistance(trig,echo)
                time.sleep(1)
                distance =self.measureDistance(trig,echo)
                if(current-distance >10) or (distance-current >10):
                        #isInvade에 사용될 boolean값
                        #True면 도둑 침입
                        return True
                else:
                        return False
class Camera:
        def __init__(self):
                self.camera = cv2.VideoCapture(0, cv2.CAP_V4L)
                self.camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
                self.camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
                #640x480 사이즈 사진 촬영
                self.camera.set(cv2.CAP_PROP_BUFFERSIZE, 1)
                 # 버퍼 크기를 1로 설정
        def initCamera(self):
                if not self.camera.isOpened():
                        self.camera = cv2.VideoCapture(0, cv2.CAP_V4L)
                        self.camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
                        self.camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
                        self.camera.set(cv2.CAP_PROP_BUFFERSIZE, 1)
        def releaseCamera(self):
                self.camera.release()
        def take_picture(self):
                ret, frame=self.camera.read()
                if ret:
                        pilim = Image.fromarray(frame)  # 프레임 데이터를 이미지 형
태로 변환
                        stream = io.BytesIO()  # 이미지를 저장할 스트림 버퍼 생성
                        pilim.save(stream, 'jpeg')
                        imBytes = stream.getvalue()
                        return imBytes #Jpeg 데이터 반환
                else :
```

```
                return None
        def writeFile(self):
                imBytes =self.take_picture()
                if imBytes:
                        # 현재 시간을 파일 이름으로 사용
                        filename                                                =
datetime.now().strftime("%Y-%m-%d_%H-%M-%S.jpg")
                        with open(f'static/{filename}', 'wb') as file:
                                file.write(imBytes)
                        return filename
                else:
                        print("file write err")
class Temp:
    def __init__(self):
        self.i2c = busio.I2C(scl=scl, sda=sda)
        self.sensor = HTU21D(self.i2c)
    def getTemperature(self):
        temperature = float(self.sensor.temperature)
        return temperature
class Mqtt:
        def __init__(self,camera):#생성자
                self.camera=camera;
                broker_ip ="localhost"#publisher가 라즈베리파이므로 localhost
                self.client = mqtt.Client() #mqtt 클라이언트 객체 생성
                self.client.connect(broker_ip, 1883)  # 1883 포트로 mosquitto에 접속
                #카메라 자원 공유 문제 해결을 위한 구독
                self.client.subscribe("cameraControl")
                self.client.on_message =self.on_message
                self.client.loop_start()  # 메시지 루프를 실행하는 스레드 생성
        def __del__(self):#소멸자
                self.client.loop_stop()#클라이언트 객체 루프 종료
                self.client.disconnect()#클라이언트 객체 연결 종료
        def on_message(self,client, userdata, message):
                if  message.topic  =="cameraControl"and  message.payload.decode()
=='release':
                        self.camera.releaseCamera()  # 카메라 자원 해제
                elif  message.topic  =="cameraControl"and  message.payload.decode()
=='activate':
                        self.camera.initCamera() # 카메라 세팅
        def publishTemp(self,temp):
                self.client.publish("temp",temp,qos=0)
        def publishAlert(self,isInvade,temp):
                if(isInvade):
                        alert ="집에 침입자가 있을 수 있습니다."
                        self.client.publish("alert",alert,qos=0)
                if(temp >20 ):
                        alert ="집에 화재가 발생했을 수 있습니다."
                        self.client.publish("alert",alert,qos =0)
class Pub: #publisher 클래스
        def run(self):
                sonic = Sonic()
                temp=Temp()
                camera=Camera()
                mqtt=Mqtt(camera)
```

```
                button = Button()
                while True:
                        #사용자가 스위치를 눌러 끄지 않은 상태
                        if controlFlag:
                                temperature = temp.getTemperature()
                                mqtt.publishTemp(temperature)
                                isInvade = sonic.isWhoInvade()
                                mqtt.publishAlert(isInvade,temperature)
                                if isInvade:
                                        #도둑이 들 경우 사진 저장
                                        camera.writeFile()
                                time.sleep(1)  # 루프당 1초 대기
if __name__=="__main__":
        GPIO.setmode(GPIO.BCM)  # BCM 모드로 작동
        GPIO.setwarnings(False)  # 경고글이 출력되지 않게 설정
        pub = Pub()
        try:
                pub.run()
        except KeyboardInterrupt: #Ctrl C 입력시 예외처리
                print("Ctrl+C로 강제종료")
        finally:
                GPIO.cleanup()  # 어떤 식으로 종료되든 GPIO 정리
```

<pFlask.py>

이 코드로 웹 페이지를 렌더링한다.

```
import base64
import time
import cv2
from flask import Flask, render_template, url_for
from pub import Camera
app = Flask(__name__)  # 플라스크 객체 생성
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/show_cctv/')
def cctv():
    camera = Camera()
    imBytes = camera.take_picture()
    if imBytes is not None:
        # 바이트 배열을 Base64 문자열로 인코딩
        img_base64 = base64.b64encode(imBytes).decode('utf-8')
    else:
        img_base64 =None
        print("img is None")
    return render_template("show_cctv.html", img_data=img_base64)
if __name__ =='__main__':
    app.run(host='0.0.0.0', port=8080)  # debug=True 속성을 사용하면 오류 발생
```

<Project.js>

이 코드로 publishing 된 값을 받아서 웹 페이지에 출력하거나, 사용자가 요청하는 구독 신청을 처리한다.

```
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
let tempFlag=False;
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
        // id가 message인 DIV 객체에 브로커의 IP와 포트 번호 출력
        //document.getElementById("messages").innerHTML += '<span>접속 : ' +
broker + ' 포트 ' + port + '</span><br/>';
        //document.getElementById("messages").innerHTML += '<span>사용자 ID : ' +
        CLIENT_ID + '</span><br/>';
        // MQTT 메시지 전송 기능을 모두 가짐 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 onConnectLost()
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() {
        let broker =document.getElementById("broker").value; // 브로커 IP 가져오기
    let messageDiv =document.getElementById("connect-message");
    messageDiv.innerHTML ="연결이 완료됐습니다. IP: "+ broker; // 메시지 표시
        connectionFlag =true; // 연결 상태로 설정
        subscribe('alert');
}
function subscribe(topic) {
        if(connectionFlag !=true) { // 연결되지 않은 경우
                alert("연결되지 않았음");
                return false;
        }
        if (topic ==='show_cctv') {
        // 'show_cctv'를 구독하기 전에 'camera_control' 토픽에 메시지 발행
        publish('cameraControl', 'release'); // 카메라 자원 해제 요청
        // 실제 'show_cctv' 토픽 구독
        client.subscribe(topic);
        }
        // 구독 신청하였음을 <div> 영역에 출력
        //document.getElementById("messages").innerHTML += '<span>구독신청: 토픽 '
+ topic + '</span><br/>';
```

```
        client.subscribe(topic); // 브로커에 구독 신청

}
function publish(topic, msg) {
        if(connectionFlag !=true) { // 연결되지 않은 경우
                alert("연결되지 않았음");
                return false;
        }
        client.send(topic, msg, 0, false);
}
function switchToindex() {
    publish('cameraControl', 'release');  // 카메라 자원 해제 요청
        publish('cameraControl', 'activate'); //pub가 다시 카메라
    location.href ='./index/';  // index 페이지로 이동
}
function switchToCCTV() {
    publish('cameraControl', 'release');  // 카메라 자원 해제 요청
    location.href ='./show_cctv/';  // CCTV 페이지로 이동
}
function unsubscribe(topic) {
        if(connectionFlag !=true) return; // 연결되지 않은 경우
        if(topic ==="temp")
                document.getElementById("temp-messages").innerHTML ='다시 온도를
보려면 <온도 보기>버튼을 눌러 주세요.';
        // 구독 신청 취소를 <div> 영역에 출력
        //document.getElementById("messages").innerHTML  +=  '<span>구독신청취소:
토픽 ' + topic + '</span><br/>';
        client.unsubscribe(topic, null); // 브로커에 구독 신청 취소
}
// 접속이 끊어졌을 때 호출되는 함수
function onConnectionLost(responseObject) { // responseObject는 응답 패킷
        let messageDiv =document.getElementById("connect-message");
    messageDiv.innerHTML ="오류 : 연결이 종료되었습니다."; // 메시지 표시
        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 msg는 도착한 MQTT 메시지를 담고 있는
객체
        //console.log("onMessageArrived: " + msg.payloadString);
        if(msg.destinationName ==="temp"){
                document.getElementById("temp-messages").innerHTML  ='<span>온도
: '+ msg.payloadString +'</span><br/>';
        }
        let alertDiv =document.getElementById("alertImage");
        alertDiv.style.display ="block"; // alertImage DIV를 표시
    let fireImg =document.getElementById("fireImg");
    let thiefImg =document.getElementById("thiefImg");
    if(msg.destinationName ==="alert") {
        if(msg.payloadString.includes("화재")) {
            // 화재 이미지 표시
```

```
                fireImg.src ="/static/fire.jpg";
                        fireImg.style.display ="block"; // 이미지 표시
            } else if(msg.payloadString.includes("침입자")) {
                // 도둑 이미지 표시
                thiefImg.src ="/static/thief.png";
                        thiefImg.style.display ="block"; // 이미지 표시
            }
        }
}
// disconnection 버튼이 선택되었을 때 호출되는 함수
function disconnect() {
            if(connectionFlag ==false)
                    return; // 연결 되지 않은 상태이면 그냥 리턴
            client.disconnect(); // 브로커와 접속 해제
            let messageDiv =document.getElementById("connect-message");
        messageDiv.innerHTML ="연결이 종료되었습니다."; // 메시지 표시
            connectionFlag =false; // 연결 되지 않은 상태로 설정
}
```

<Index.html>

메인 웹 페이지를 구성하는 html 파일.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>스마트 홈 IOT 보안 시스템</title>
      <link rel="stylesheet"type="text/css"href="./static/projectStyleSheet.css">
      <script
src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.2/mqttws31.min.js"type="text/j
avascript"></script>
      <script src="./static/project.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>
   </head>
   <body>
         <h1>스마트 홈 IOT 보안 시스템</h1>
         <hr>
         <div id="centeralign">
         <h3 id="highlight-title">연결 관리</h3>
         </div>
         <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="connect()"value="Connect">
            <input type="button"onclick="disconnect()"value="Disconnect">
         </form>
            <div id="connect-message"></div>

        <hr>
        <div id="centeralign">
        <h3 id="highlight-title">온도 보기</h3>
        </div>
         <form id="invade-form">
            <input type="button"onclick="subscribe('temp')"value="온도 보기">
            <input type="button"onclick="unsubscribe('temp')"value="그만 보기">
         </form>
            <div id="temp-messages"></div>
         <hr>
         <div id="centeralign">
         <h3 id="highlight-title">실시간 영상 보기</h3>
         </div>
            <form id="streaming-form">
            <input type="button"onclick="switchToCCTV()"value="영상 보기">
```

```
            </form>
        <div id="imageContainer"style="text-align: center;"><!-- 중앙 정렬을 위한 컨테이너 추가 -->
            <div id="alertImage"style="display:none;">
                <img id="fireImg"src=""width="320"height="240"/>
                <img id="thiefImg"src=""width="320"height="240"/>
            </div>
        </div>
            <div id="messages"></div>
    </body>
</html>
```

<show_cctv.html>

사용자가 “실시간 영상 보기” 버튼을 눌렀을 때, 실시간 CCTV 영상이 출력될 웹 페이지를 구성한다.

```
<!DOCTYPE html>
<html>
<head>
    <title>CCTV</title>
    <script src="/static/project.js"type="text/javascript"></script>
    <meta http-equiv="refresh"content="0.5"><!-- 2초마다 페이지 새로고침 -->
</head>
<body>
    <input type="button"onclick="switchToindex()"value="돌아가기">
    <hr>
    {% if img_data %}
        <img src="{{ img_data }}">
    {% else %}
        <p>No image available.</p>
    {% endif %}
</body>
</html>
```

<ProjectStyleSheet.css>

사용감을 위해서 메인 웹 페이지 Index.html을 예쁘게 꾸며주는 코드.

```
body {
    background-color: #FDFDFD;
    color: #000;
    margin: 0;
  }
h1 {
    color: #333;
    text-align: center;
}
#temp-messages {
    text-align: center;/* 텍스트 가운데 정렬 */
    margin: 20px auto;/* 상하 마진 20px, 좌우 마진 자동 */
    padding: 10px;/* 패딩 10px */
    border: 1px solid #ddd;/* 경계선 */
    border-radius: 4px;/* 둥근 경계선 */
    box-shadow: 2px 2px 5px #aaa;/* 그림자 효과 */
    background-color: #F5F5F5;/* 배경색 */
    width: 80%;/* 너비 80% */
    max-width: 600px;/* 최대 너비 600px */
}
#connect-message {
    text-align: center;/* 텍스트 가운데 정렬 */
    margin: 20px auto;/* 상하 마진 20px, 좌우 마진 자동 */
    padding: 10px;/* 패딩 10px */
    border: 1px solid #ddd;/* 경계선 */
    border-radius: 4px;/* 둥근 경계선 */
    box-shadow: 2px 2px 5px #aaa;/* 그림자 효과 */
    background-color: #F5F5F5;/* 배경색 */
    width: 80%;/* 너비 80% */
    max-width: 600px;/* 최대 너비 600px */
}
h3 {
    color: #333;
    text-align: center;
}
#centeralign {
    text-align: center;
}
#highlight-title {
    background-color: #FCEBDB;/* 연한 배경색 또는 투명도 조절 */
    border-bottom: 1px solid #ccc;/* 아래쪽 테두리만 설정 */
    padding: 10px;/* 내부 여백 */
    margin: 10px auto;/* 상하 여백 10px, 좌우 자동 */
    display: inline-block;/* 박스를 내용에 맞춰 크기 조정 */
    font-size: 1.5em;/* 글꼴 크기 조정 */
    font-weight: bold;/* 글꼴 무게 */
    max-width: 80%;/* 최대 너비 설정 */
}
hr{
    border: none;
    height: 10px;
```

```
    background-color: #BC8F8F;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
    margin: 20px 0;
}
#connection-form, #invade-form, #streaming-form {
    margin: 20px auto;
    width: 80%;
    text-align: center;
}
input[type=text], input[type=button]{
    padding: 10px;
    margin: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
}
input[type=button]{
    background-color: #5CACEE;
    color: white;
    cursor: pointer;
}
input[type=button]:hover {
    background-color: #1E90FF;
}
#imageContainer {
    display: flex;
    justify-content: center;
    align-items: center;/* 만약 이미지가 수직으로도 중앙에 오기를 원한다면 이 속성을
유지합니다. */
    margin-top: 20px;/* 버튼과의 간격을 위해 상단 마진 추가 */
    margin-bottom: 20px;/* 다음 섹션과의 간격을 위해 하단 마진 추가 */
}
#alertImage {
    display: flex;
    flex-direction: row;/* 이미지를 가로로 배열 */
    justify-content: center;
}
#fireImg, #thiefImg {
    margin: 0 10px;/* 이미지 사이의 좌우 간격 */
}
```

4. 프로젝트 결론

인생에서 처음으로 제작해 본 프로젝트였다. 꼬박 한 학기 내에 배운 지식을 모두 활용해야 하는 정말 어려운 프로젝트였다.
돌이켜 보면, 아이디어를 선정하는 일이 가장 어려웠던 것 같다. 처음 강의를 수강할 때는 세상을 바꿀만한 대단한 무언가를 만들겠다는 마음이 가득했는데, 내가 생각해 낸 획기적인 아이디어의 무언가들은 이미 세상에 있었다. 그래서 내 생활에 필요한, 정말 사용할 수 있는 프로젝트를 하고자 했다. 그렇게 조금은 단순하고 진부할지라도 꼭 필요하다고 생각한 “자취생을 위한 스마트 홈 IOT 보안 시스템”을 제작하게 됐다.

단순히 교재의 코드를 베껴 쓴다고 가능한 일이 아니었다. 회로 구성부터 GPIO 핀 제어, 센서의 원리, 시스템 구조까지 모든 것을 이해해야 프로젝트를 제작할 수 있었다.

첫 시작부터 산 넘어 산이었다. 처음엔 초음파 센서가 말썽이었다. 초음파를 발사하고 수신하는 코드 사이에 sleep(0.1) 코드가 없었는데, 컴퓨터가 너무 빨리 작동하면서 문제를 일으켰다. 채 수신하기도 전에 다시 발사하면서 문제가 일어났던 것이다. 한참을 헤매다 결국 문제가 일어나는 근본적인 원인을 생각하게 됐고, 코드 한 줄로 문제를 해결할 수 있었다.

두 번째 어려움은 코딩이었다. 객체지향언어 1, 2 강의를 수강하면서 배운 객체지향적 프로그래밍을 적용하고 싶었다. 교재에 있는 코드를 이해하고, 객체지향적으로 코드를 작성하기 위해 많이 노력했다. 사실 완벽하게 캡슐화를 하지도, 클래스를 엄청 멋지게 활용하지도 못했다. 하지만 직접 코딩을 하면서 왜 객체지향 프로그래밍이 대세가 될 수 밖에 없었는지, 어떻게 구조를 짜야 좋은 코드인지 직접 느낄 수 있었다. 아직 실력이 부족하다는 점도 많이 느껴져 겨울 방학을 활용해 객체지향 프로그래밍을 좀 더 공부해 볼 생각이다.

마지막 어려움은 카메라 자원의 공유 문제였다. 내가 원하는 시스템은 도둑이 드는 순간의 사진도 촬영하고, 실시간 자취방 영상을 보여줄 수도 있어야 했다. 실시간 영상 보기를 먼저 구현했었다. 처음엔 단순히 책에 있는 코드를 베껴 쓴 수준이었다. 실시간 영상 보기 코드를 Flask 코드에 삽입했었는데, 이렇게 하면 하나의 카메라 자원을 pFlask.py 파일과 pub.py 파일이 공유하게 된다. 뒤에 실행하는 파일은 카메라를 인식할 수 없다는 오류를 냈다. 문제의 원인을 찾는 것 조차 쉽지 않았다. 처음에는 카메라나 USB 포트의 고장인 줄 알고 이리저리 바꿔가며 온갖 테스트를 다 했다. 초음파 센서 때와 마찬가지로 코드에 목 메는 것이 아닌 전체적인 구조를 파악하고 근본적인 원인을 찾는데 집중하자 결국 문제를 찾아낼 수 있었다.
내가 문제 해결을 위해 택한 방법은, 카메라 자원을 하나의 파일만 쓸 수 있도록 새로운 topic을 구독하는 것이었다. cameraControl 토픽에, release와 activate 메시지를 넣어 publishing하는 방식이다. 시스템이 동작하면 이 토픽을 시스템이 미리 구독한다. 도둑이 들면 카메라는 사진을 계속 찍다가, 사용자가 실시간 영상 보기를 누르는 순간, 카메라 자원을 해제하는 camerControl topic에 release 메시지를 넣어 보낸다. 그럼 pub.py 파일은 카메라 자원을 해제하고, pFlask.py 파일이 카메라 자원을 가져가게 된다. 반대로 실시간 영상 보기 페이지에서 돌아가기 버튼을 누르면 pFlask.py 파일이 카메라 자원을 해제한다.

이번 프로젝트를 통해 정말 많은 것을 배울 수 있었다. 내가 생각하는 프로그래머는 단순히 책상에 앉아 머리를 싸매고 코드를 작성하는 사람이었다. 하지만 이번 프로젝트를 통해, 진짜 실력 있는 프로그래머는 하드웨어와 소프트웨어를 막론하고 전체적인 이해가 필요하다는 것을 깨닫게 됐다. 만약 코드가 동작하는 방식, 센서가 값을 받아오는 방식, 회로가 구성되는 방식, MQTT 프로토콜, Flask 프레임워크 등 어떤 것 하나라도 제대로 이해하지 못했다면 내가 하고자 하는 프로젝트를 완성하지 못했을 것이다.

교수님께서 강조해 오시던 “결국은 이론 싸움이다.”라는 말을 뼈저리게 느낄 수 있었다. 프로젝트를 통해 단순히 코드를 짜는데에 급급하던 눈이 더 넓은 세상을 볼 수 있는 눈으로 트인 느낌이다. 앞으로도 학부생으로 도전할 수 있는 프로젝트가 몇 개 더 있을 텐데, 오늘의 교훈을 바탕으로 더 근사한 무언가를 완성해보고 싶다.