IOT 설계 및 프로그래밍

금오공과대학교
컴퓨터 공학과
20200123 김다은

개미의 신호등

2021년 12월 21일

작품의 목적 및 필요성

복잡한 도심을 걷다 보면 마주 오는 사람들과 어깨를 부딪히거나 넘어지는 경우가 허다하다. 사람들의 통행 방식에 뭔가 매끄럽지 못한 부분이 있기 때문이다. 신호등을 건널 때 시간이 부족한 경험 특히, 장애를 가진 사람의 경우 신호등을 건널 시간이 부족하다고 느낄 수 있다. 이 경우 버튼 클릭으로 신호등의 시간을 늘려주고 현재 교통 상황이 어떤지 그리고 만일 사고가 날 경우 그 상황을 알 수 있도록 웹을 통해 카메라로 현재 교차로의 상황을 확인할 수 있다.

이러한 신호등을 사용하여 보행자의 안전을 지키고 현재 교통상황을 쉽게 알 수 있다. 교통사고가 자주 발생하는 교차로에 이러한 신호등이 설치되어 있다면 큰 효과를 볼 수 있을 것이다.

작품의 최종 개발 사진

스티로폼을 사용하여 신호등에 부착하기 쉬운 형태로 제작하였다. 빨간색 LED 와 초록색 LED 는 스레드를 사용하여 교차로 반짝이도록 구현하였고 버튼을 클릭하면 초록불이 뜨고 있는 시간이 더 늘어난다. USB 카메라를 부착하여 현재 교통상황을 스트리밍하고 웹을 통해 확인할 수 있다.

상자 안에는 라즈베리 파이가 존재하고 전원을 꼽는 곳은 뒤에 구멍이 위치한다.

작품의 구조 및 설계 내용

■ Client 구조

CLIENT
- vue.js - axios - html
클라이언트에서는 flask 에서 받은

CLIENT
프레임 이미지를 계속 띄워주고 버튼이 클릭될 때 마다 웹에서 영상을 띄우던 thread 를 멈추도록 하여 영상을 멈출 수 있도록 구현하였다. 날이 어두울 경우에는 화면이 잘 보이지 않을 수도 있으므로 밝기를 조절할 수 있는 Input Box 를 만들어 두었다.

■ Server 구조

SERVER
- flask - grovepi
SERVER 에서는 led 를 빨간불 파란불로 바꾸어주고 opencv 를 이용하여 usb 카메라의 영상을 스트리밍할 수 있도록 해준다. 라우터 형식으로 접근하여 크롤링을 통해 현재날씨 정보를 알 수 있고 현재 빨간불인지 초록불인지 알 수 있다. 또한 submit 을 입력하면 submit 을 누른 시점의 frame 을 저장할 수 있다.

작품의 구현 내용

영상 스트리밍

■ Thread 를 사용하여 cam_get 호출

```
db = database()
th = Thread(target=cam_get)
th.daemon = True
th.start()
```

■ cam_get

videoCapture 객체를 생성하고 while 문을 사용해 초당 frame 을 생성하여 db 에 저장한다.

```
def cam_get():
    global db
    global img
    try:
        db.cap = cv2.VideoCapture(0)
        while True:
            lock.acquire()
            ret, img = db.cap.read()
            lock.release()
            time.sleep(0.03)
    except (KeyboardInterrupt, SystemExit):
        print('₩n! Received keyboard interrupt, quitting threads.₩n')
    finally:
        db.cap.release()
```

■ video_feed

db.img 을 encode jpeg 형식으로 변환한 뒤에 response 로 전달해준다.

client : axios 를 통해 video_feed에 접근하고 start 를 누르면 setInterval을 통해 이미지를 video_feed 에 접근하기 위한 getImg() 를 계속하여 호출한다.

```
@app.route('/video_feed')
def video_feed():
    global db
```

```
    lock.acquire()
    encoded = base64.b64encode(cv2.imencode('.jpeg', img)[1].tobytes())
    lock.release()

    response = make_response(encoded)
    response.headers.set('Content-Type', 'image/jpg')
    response.headers.set('Content-Disposition', 'attachment', filename='test.jpg')
    return response
```

버튼 클릭시 신호 전환 속도 조절

- traffic_light

while 문 한번이 돌 때 마다 time.sleep(1) 을 해준다. 따라서 db.lightcount 원래 시간에서 delay 만큼을 곱하면 초록불이 켜지는 신호가 늘어나게 된다. button_status 를 받아 버튼이 클릭되었다면 delay 를 증가 시켜 신호등이 켜지는 시간을 더 길게 한다.

```
button_status = digitalRead(button)
delay = 2
if(button_status):
    flag = True
if(count > db.lightcount):
    if(flag):
        delay = 4
    if(count > db.lightcount*delay):
        count = 0
    digitalWrite(greenled, 1)
    digitalWrite(redled, 0)
    db.light = not db.light
```

신호등 시뮬레이션

- traffic_light

flag 를 통해 빨간불 초록불을 전환 시킬 수 있도록함 만약에 flag 가 true 였다면 red led, false 였다면 green led 로 구현

```
def traffic_light():
    global db
    count = 0
    flag = False
    while True:
        count += 1
        button_status = digitalRead(button)
        delay = 2
        if(button_status):
            flag = True
        if(count > db.lightcount):
            if(flag):
                delay = 4
            if(count > db.lightcount*delay):
                count = 0
            digitalWrite(greenled, 1)
            digitalWrite(redled, 0)
            db.light = not db.light
        else:
            if(flag):
                flag = False
                delay = 2
            digitalWrite(redled, 1)
            digitalWrite(greenled, 0)
        time.sleep(1)
```

횡단보도의 신호 확인

127.00.:8000/get_settings/<값> 형태로 접근한다. db 에 저장되어있는 현재 상태를 받아 response 로 넘겨준다.

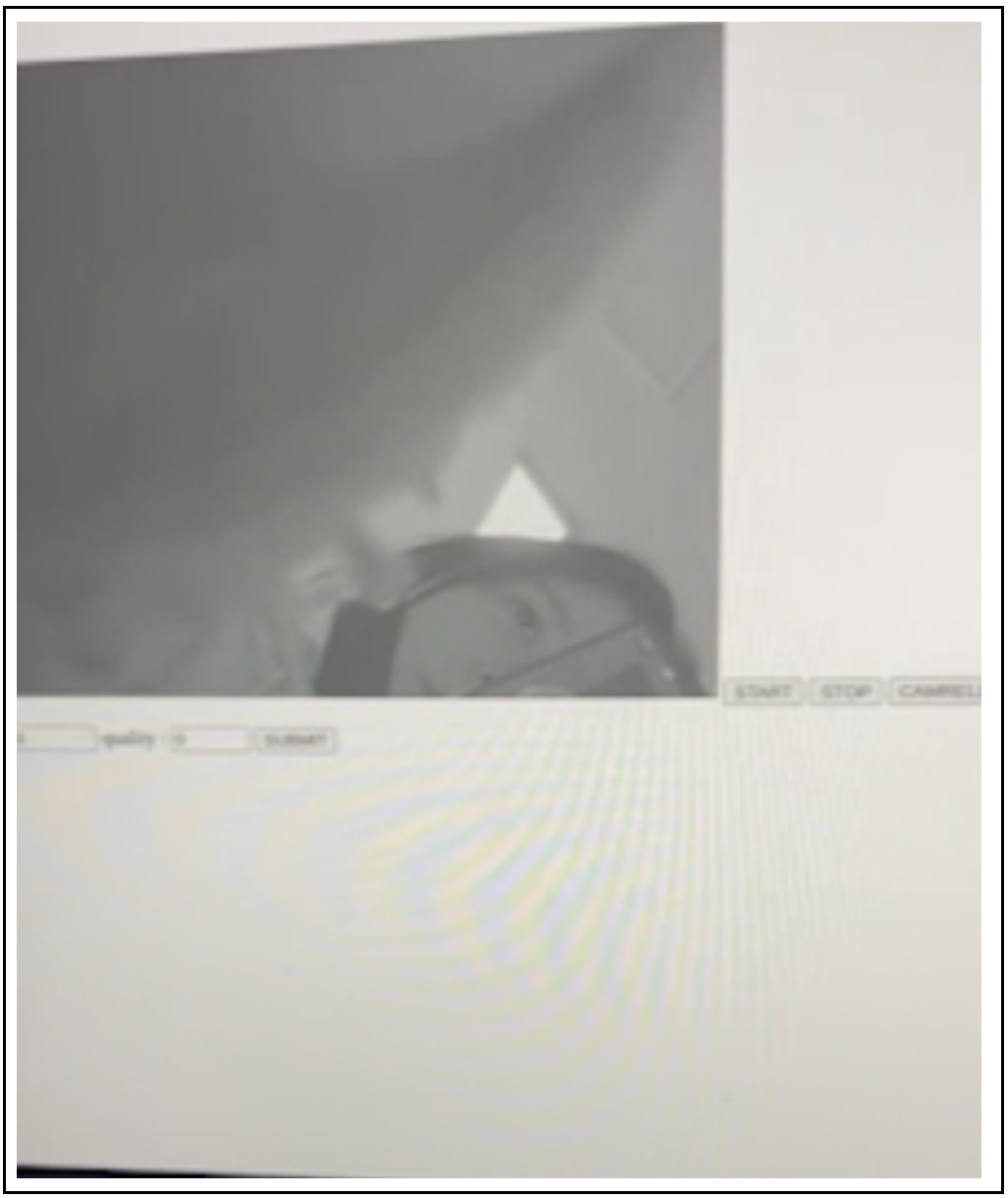
```
@app.route('/get_settings/<setting>')
    def get_settings(setting):
        global db
        if(setting == "brightness"):
            return str(db.brightness)
```

```
    if(setting == "quality"):
        return str(db.quality)
    if(setting == "weather"):
        return str(db.weather)
    if(setting == "light"):
        info = ""
        if(db.light):
            info = "RED"
        else:
            info = "GREEN"
        return str(info)
    response = make_response()
    return response
```

작품의 결과 및 완성도

영상 스트리밍

■ 스트리밍 화면



사고가 많이 발생하는 교차로의 신호등에 카메라를 부착하여 스트리밍 함을 통해 사고 예방 및 증거자료 수집에 유용하게 사용될 수 있다. SUBMIT 을 누르면 사진이 저장된다.

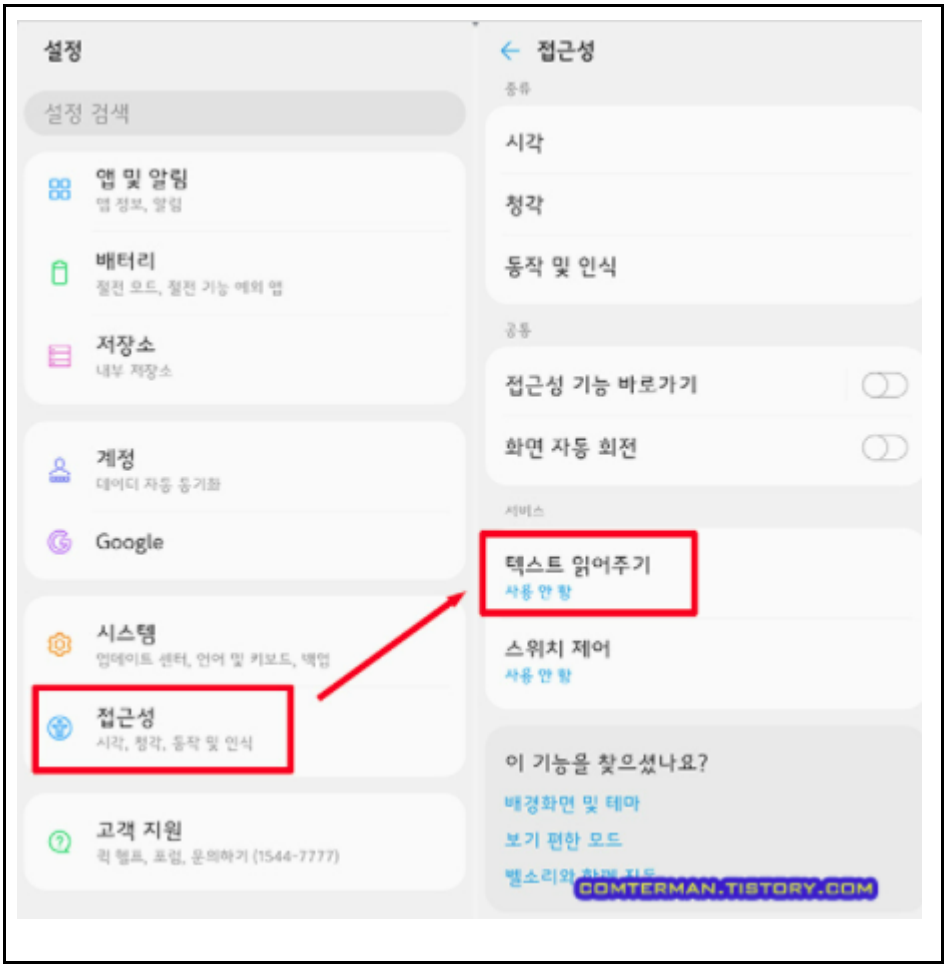
■ 저장된 화면 예시

버튼 클릭시 신호 전환 속도 조절

버튼 클릭시 초록색 신호를 해당 신호에만 길게 적용하여 거동이 불편하신 분이나 짐이 많은 사람 등의 보행을 안전하게 지켜준다.

신호등 시뮬레이션이 가능하도록 빨간색 파란색 led 를 부착하여 어떻게 작동하는지 더 잘 와닿을 수 있도록 한다.

횡단보도의 신호 확인



시각 장애인의 경우 현재 신호등의 상태를 알기 어렵다. 대다수의 신호등은 빨간불로 전환 되었을 때 이를 소리 정보로 알려주지 않는다. 하지만 스마트폰에는 웹사이트의 글을 읽어주는 기능이 존재한다. 이 기능을 활용할 수 있도록 get_settings/light을 구현하였고 현재 신호등이 빨간불인지 초록불인지 웹사이트를 통해 정보를 얻어 보행을 더 안전하게 할 수 있도록 돕는다.

참고 자료 및 구현 소감

■ 참고 자료

주제	링크
Flask 관련	https://www.tutorialspoint.com/flask/flask_redirect_and_errors.htm
Axios 관련	https://yamoo9.github.io/axios/guide/
raspberry pi 관련	https://projects.raspberrypi.org/en/projects/python-web-server-with-flask

■ 구현 소감

현실에서 발생할 수 있는 문제에 대해서 고민하고 설계하여 구현하는 것에 재미를 느꼈다. 아이오티를 통해 사람들의 생활을 편리하게 만들어줄 수 있는 기회가 무궁무진 할 것이라고 생각하였다. 또한 아이오티는 비용적인 측면에서 저렴해야 하는데 구현을 했을 때 이러한 비용적인 측면과 기대효과를 생각해보는 것이 재미가 있었다. 만약 gpu 가속이 가능했거나 소켓 프로그래밍을 배워 object_detection 을 따로할 수 있었다면 훨씬 더 좋은 기능을 구현할 수 있었을 것 같은데 아쉬움이 많이 남는다. object_detection 을 mobilenet 을 통해 구현해 보니 영상당 프레임의 계산 속도가 너무 느려 웹페이지에 바운딩 박스가 포함된 영상을 띄울 때 마다 라즈베리 파이 화면이 멈추는 오류가 발생하였다. 이를 해결하지 못한 것이 너무 아쉽다. 아이오티를 수강하면서 버전을 맞추는 법 다양한 오류를 해결하는 방법에 대해 알 수 있었다. 또한 설계를 할 때 비용적인 측면을 고려하고 어떻게 하면 제일 효율적일지 고민해볼 수 있는 기회가 되었다.