임베디드 컴퓨팅

스마트 택배함

무인택배관리 프로그램

제출일: 2019년 12월 12일

담당교수님: 김두현교수님

[15조] 소프트웨어학과

201511191 문병준

201611235 전희재

• 개요

1. 기말프로젝트 주제 설명

[문앞의 택배를 안전하게 보관할 방법이 없을까?]

경비실이 없는 개인 주택이나 빌라의 경우, 택배를 받을 수 없는 상황이 자주 발생한다. 그런경우, 다른 곳에 맡기긴 번거로우니 위험하더라도 대부분 택배 기사님께 집 앞에 택배를 두고 가주시길 부탁한다.
집 앞에 택배가 놓이면 도난의 위험이 있고, 언제 택배가 도착했는지도 알 수가 없다. 그 대안인 개인용 스마트 택배함이나 건물 공동으로 구입하자니 가격이 높아 부담된다. 스마트 택배함을 구비한다 해도 택배의 크기가 알 수 없는 경우도 많아, 택배를 넣을 수 없는 상황도 발생한다.

[집주인과 택배기사 모두 번거롭지않게 택배를 안전하게 보관할수있는 스마트 택배 보관함]

우리가 제안하는 스마트 택배 보관함은 실체가 없는 보관함으로, 택배의 크기에 제한되어있지 않다. 실시간으로 문 앞에 놓인 물체를 감지하고 택배로 취급되면 사용자에게 메시지를 전송한다. 사용자가 택배가 맞음을 확인하면 확인했음을 전송하고 그 후부터는 도난방지 모드로 전환해 택배를 지킨다. 도난감지시에는 적절한 경보를 울리고 사용자에게 도난당시의 영상을 찍어 전송한다.
이러한 시스템으로 택배기사님이 어떤크기의 택배라도 간단히 문앞에 놓고 가시더라도 안전하게 실시간으로 보관이 되므로 집주인은 간단한 장치로 도난의 걱정을 덜수있다. 또한 텔레그램봇을 이용한 메시지 전달로 모바일이나 PC 상관없이 여러명이 있는 채팅방또는 개인 메시지로 알람을 줄 수 있다. 이를 이용해 가족단위(가족 단톡방)에서도 관리가 가능하다.

2. 대표 시나리오

- 빌라나 경비실이 없는 아파트에서 문 앞에 라즈베리파이와, 브레드보드, 모듈들을 부착한다.

- 문 앞에 택배가 놓아지면 (일정시간 동안 적외선센서가 오래 감지되면)
  텔레그램봇으로 url이 첨부된 메시지를 보낸다. 고양이나 어떤 사람이 와서 센서가 감지되서 메시지가 보내질 수 도 있다. 카메라로 사진을 찍어 웹으로 보내기 때문에 사용자가 판단할 수 있다. 택배가 맞을경우 도난방지 모드가 작동된다.

- 도난방지 모드상태에서 택배가 사라지면 즉시 경보음이 울리고 메세지가 간다.
  집주인에게는 . 알람과 함께 도난을 감지한 당시의 동영상을 보여준다.

3. 계획서 대비 변경내용과 이유

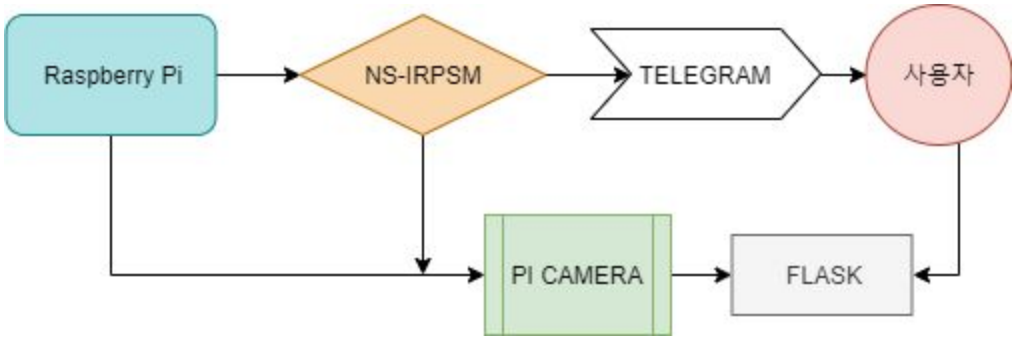
[도난감지에 대해 사용자의 판단 → 도난감지후, 사용자 판단없이 바로 경보, 동영상 촬영]
택배가 맞는지는 사용자의 판단이 필요하지만, 도난에 관해서는 사용자가 바로 관여하지 못하는 상황이 자주 생길수 있다는 팀 회의 결과가 나왔다. 만약 판단을 받기까지 알람이 계속 울리게 된다면 도난이 아닐경우 소음공해를 야기한다. 따라서 도난감지에 대한 판단은 묻지 않고 추가로 동영상을 녹화해 전달하기로 했다.

[웹 알람 → 텔레그램봇 메시지]
웹으로의 알람을 계획서에 기재를 했지만, 웹 알람의 경우 모바일로 확인이 불가능하니, 활용성이 떨어진다고 생각했고, 사용자가 모바일에서도 확인할 수 있는 메신저에서 알람을 받으면 좋겠다고 생각 했다. 라즈베리파이에서 메시지를 보낼수 있는 텔레그램모듈을 따로 공부해서 추가로 사용하기로 결정했다.

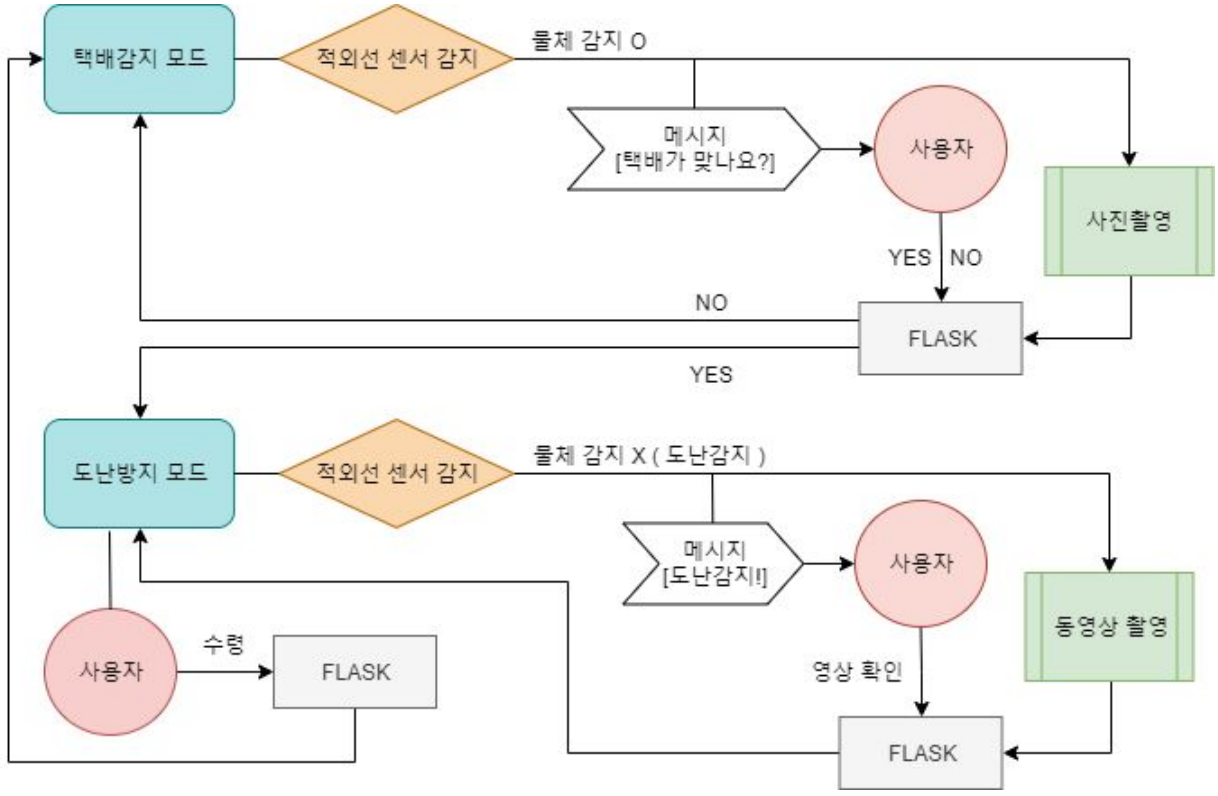
● 시스템 설계

1. 시스템 구성도

* 요약



* 세부내용



2. 구성도 설명

택배감지 모드) 택배가 오길 감지 함.

감지) 무언가가 적외선센서에 오래 머물러 있으면(10초정도) 텔레그램으로 메세지를 보냄.

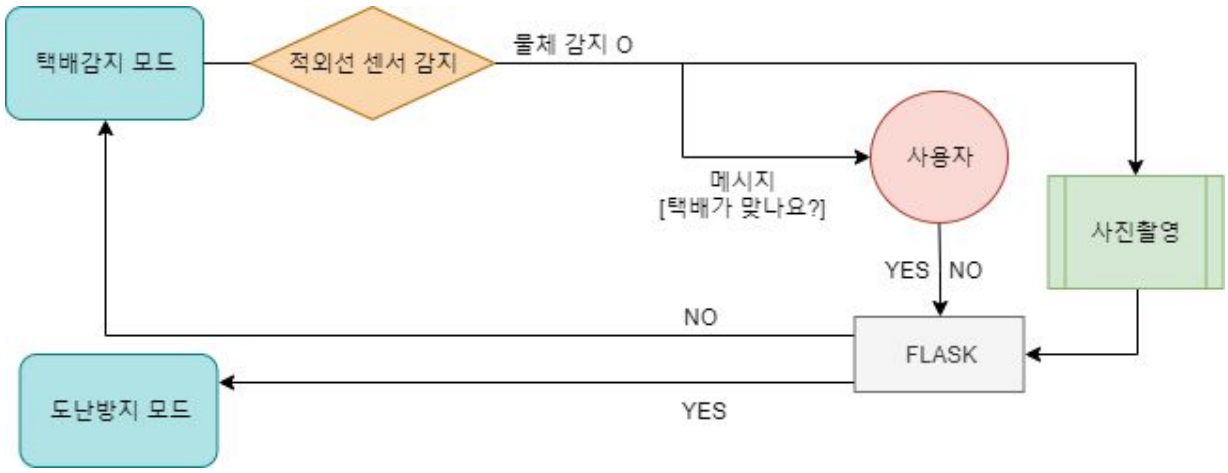
사용자 확인 상태) Flask웹 서버를 통해 실시간 카메라로 택배를 확인하고 택배상자가 맞을경우 사용자는 OK 승인을 함.
승인을 안할 경우 -> 감지 상태로 돌아감.
승인을 할 경우 -> 도난 방지 상태로 바뀐다.

도난 방지 모드) 적외선센서로 택배를 쭉 감지하고 있다가 센서에서 택배가 사라지면 스피커로 알람음 5회를 울리고 웹으로 도난당시의 동영상을 웹으로 전송과 동시에 사용자에게 텔레그램 메시지를 보낸다. 그리고 감지 상태로 돌아간다.

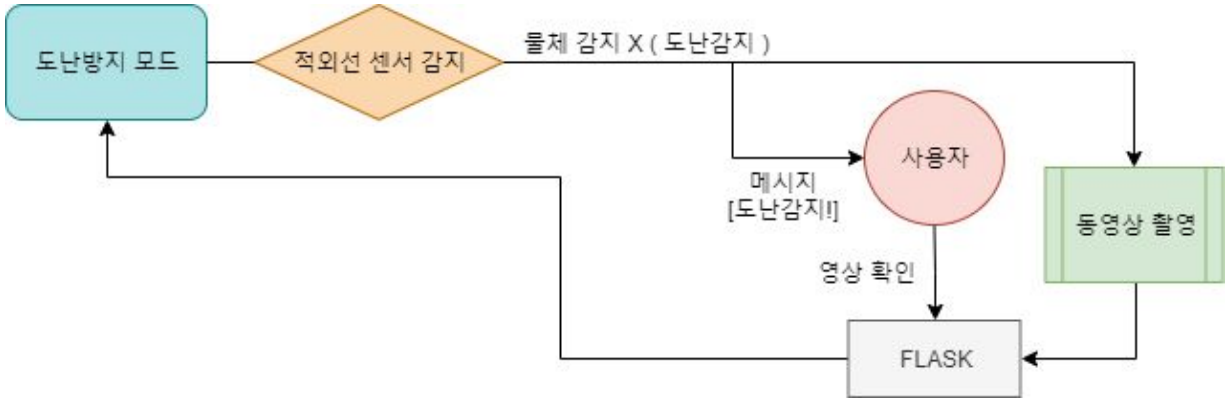
택배 수령) 내가 직접 택배를 수령할때 [수령]버튼을 누르면 도난 방지 모드에서 감지 모드로 다시 돌아간다.

3. 대표 시나리오 작동 플로우

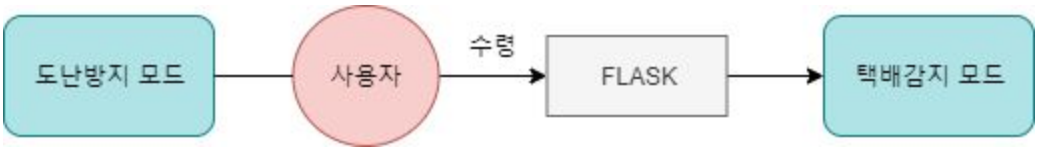
1) 문앞에 택배를 두고 가면



2) 누군가 택배를 가져가면



3) 사용자 수령



• 주요 코드

1. 주요 변수 설명

```
irpin = 21 # 적외선 센서
led = 20 # 표시LED
Time = 0 # 감지용 타임변수
Mode = 0 # [0: 감지상태] [1: 도난방지

FirstDetect = False # 처음감지됨 flag
x = 0 # 적외선 센서 감지결과
curName = '00' # 동영상을 저장할 이름
beep = pygame.mixer.Sound("beep3.wav")# 경보음
app = Flask(__name__) # 플라스크 객체
```

2. 적외선 센서를 통한 감지

```
try:
        threading.Timer(1,at.WebRun).start() # 웹 서버
        while True: # 계속 감지
            x = GPIO.input(irpin)# 적외선 센서
            time.sleep(0.1)
            if x != 1 : # 감지 O
                if FirstDetect == False and Mode == 0: # 처음감지
                    #print("물체가 감지되었습니다.")
                    at.Detector()
                    FirstDetect = True
                GPIO.output(led, GPIO.HIGH) # LED ON
            else: # 감지 X
                Time = 0
                if Mode == 1: # 도난방지모드일때 감지X == 도난
                    Mode = 2 # 도난
                    print("도난이 감지되었습니다!!!")
                    Send("!!!도난이 감지되었습니다!!!\nhttp://"+ ip +"/robbed")
                    threading.Timer(0,at.Record).start()
                    at.Alert()
                    Mode = 0 # 도난감지모드
                    FirstDetect = False
                GPIO.output(led, GPIO.LOW) # LED OFF
                FirstDetect = False
```

3. 각 thread 함수

```
def Detector(self): # 물체를 10초동안 감지후 메시지 전송
   global x, Mode, ip, t3
   if x != 1:
       global Time
       Time += 1
       print(Time)
       if Time == 10:
           threading.Timer(1,at.Shot).start()  # 타이머가 10초가 되면 사진 찍기
           print("물체가 감지되었습니다.")
           Send("* 물체가 감지되었습니다.\n* 택배가 맞으면 Yes, 아니면 No를
눌러주세요.\nhttp://"+ ip +"/detect") # 텔레그램 메세지 전송
           Time = 0
       else:
           threading.Timer(1,at.Detector).start() # 타이머가 10초가 안되면 함수를
다시불러서 시간이 증가하게 한다.
   else:
       Time =0
       Mode = 0
```

```
def Shot(self): # 사진저장
   camera = picamera.PiCamera()
   camera.resolution = (350, 400)
   camera.capture('static/ex1.jpg')
```

```
    time.sleep(0.1)
    camera.close()
```

```
def Record(self): # 동영상 저장
    global curName
    camera2 = picamera.PiCamera()
    camera2.resolution= (350, 400)
    now = datetime.now()
    curName = now.strftime('%Y%m%d%H%M%S') # 현재 시간으로 파일명 저장
    camera2.start_recording('static/'+str(curName)+'.h264')
    camera2.wait_recording(5)
    camera2.stop_recording()
    camera2.close()
    print(curName)
    call("MP4Box -add static/"+str(curName)+".h264
static/"+str(curName)+".mp4", shell=True) # 녹화가 끝나면 mp4로 인코딩
    #return render_template("DropBox.html")
```

```
def Alert(self): # 경보 울리기
    global respond, robbed, Time
    Time = 0;
    beep.play()# 외부 사운드 출력
    while True:
        Time += 1
        time.sleep(0.2)
        GPIO.output(led, GPIO.HIGH)
        time.sleep(0.2)
        GPIO.output(led, GPIO.LOW)
        time.sleep(0.2)
        if Time == 8:
            Time = 0
            break
```

```
def WebRun(self): # 웹서버 스레드
    global app
    app.run(host='0.0.0.0', port=80, debug=False, threaded=True)
```

4. Flask 웹서버 함수

```
@app.route("/detect") # 물체 감지확인 html
def detect():
    message = "물체가 감지되었습니다.\n택배가 맞나요?"
    templateData = {
        'message' : message,
    }
    return render_template('DetectMode.html', **templateData)
```

```
@app.route("/message/<action>") # 메시지 전달 html
def action(action):
    global x, Mode, FirstDetect
    if action == "receive" :
        message = "수령처리되었습니다."
```

```
        Send("택배 수령이 완료되었습니다.")
        Mode = 0
        FirstDetect = False
    else :
        if x != 1 :
            message = "택배가 아닌 것으로 처리되었습니다."
            Send("택배가 아닌 것으로 처리되었습니다.")
            Mode = 0
            FirstDetect = False
        else :
            message = "그 사이에 물체가 사라졌습니다."
    templateData = {
        'message' : message,}
    return render_template('Message.html', **templateData)
```

```
@app.route("/protect") # 택배보호 확인, 수령 html
def protect():
    global Mode, ip
    if x != 1 :
        Mode = 1 # 도난방지모드
        print("도난방지모드로 전환되었습니다.")
        Send("* 택배를 보호중입니다.\n* 택배를 수령하기전에 수령을
눌러주세요.\nhttp://"+ ip +"/protect")
        message = "택배를 보호중입니다."
        templateData = {
            'message' : message,
        }
        return render_template('ProtectMode.html', **templateData)
    else :
        message = "그 사이에 물체가 사라졌습니다."
        templateData = {
            'message' : message,
        }
        return render_template('Message.html', **templateData)
```

```
@app.route("/robbed") # 도난 감지확인 html
def robbed():
    global curName
    message = "도난이 감지되었습니다!!"
    templateData = {
        'curName' : curName,
        'message' : message,
    }
    return render_template('RobbedMode.html', **templateData)
```

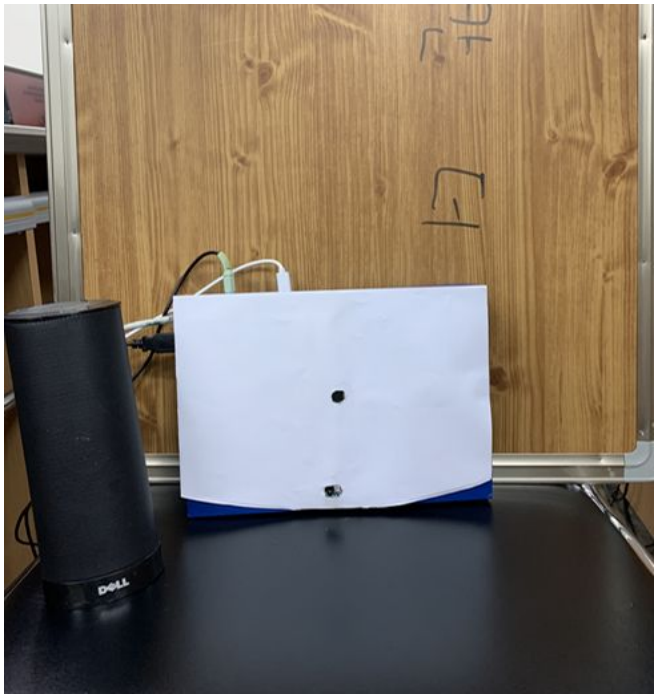
5. 텔레그램봇 메시지 보내기

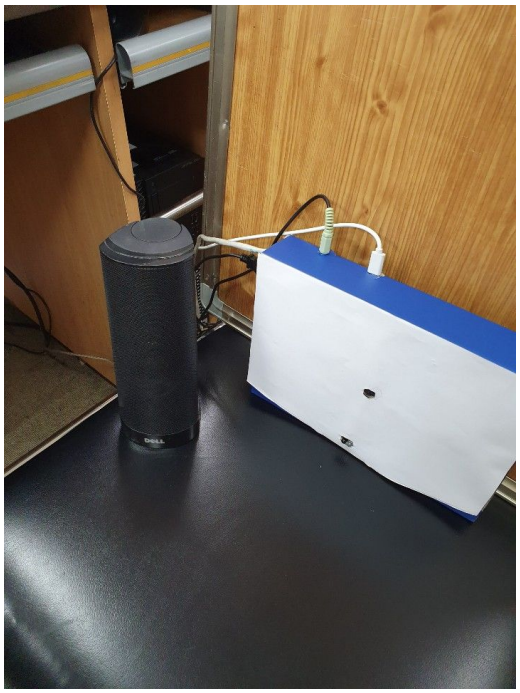
```
def Send(Message):
    telegram_id = '-328726901' #아이디 입력
    my_token = '1016022560:AAEaEHoJxgyqpqm_vXpDCnE_TrZhKM3sMy4' #token 입력
    bot = telepot.Bot(my_token) #봇을 생성해줍니다.
    bot.sendMessage(chat_id = telegram_id, text = Message) #메시지를 보내기
```

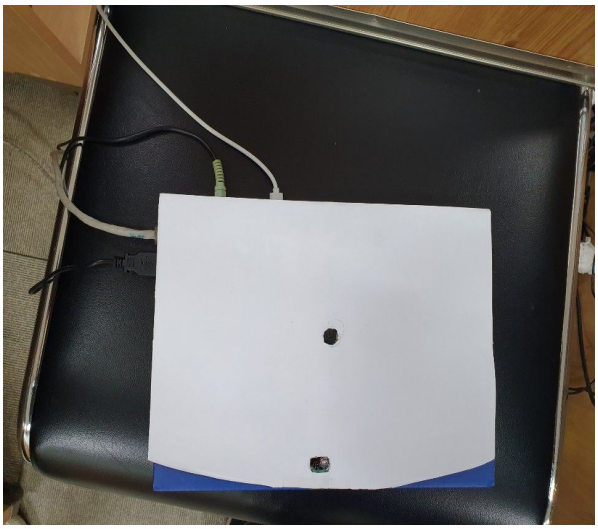
• 결과물

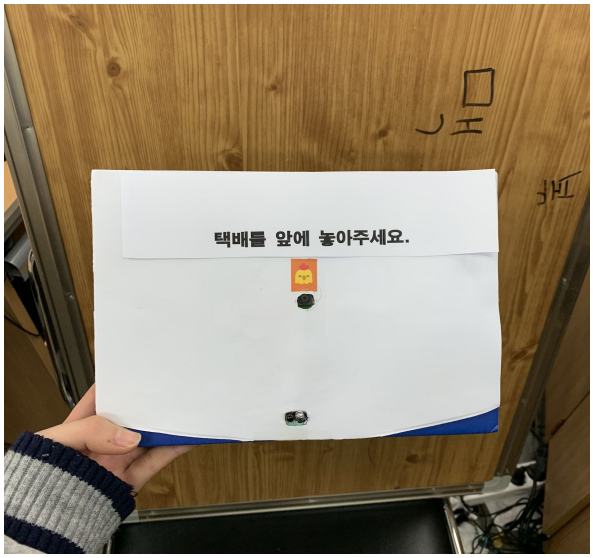
1. 수행 장면 & 분석

수행 환경 (판넬을 문으로 가정, 문앞설치)



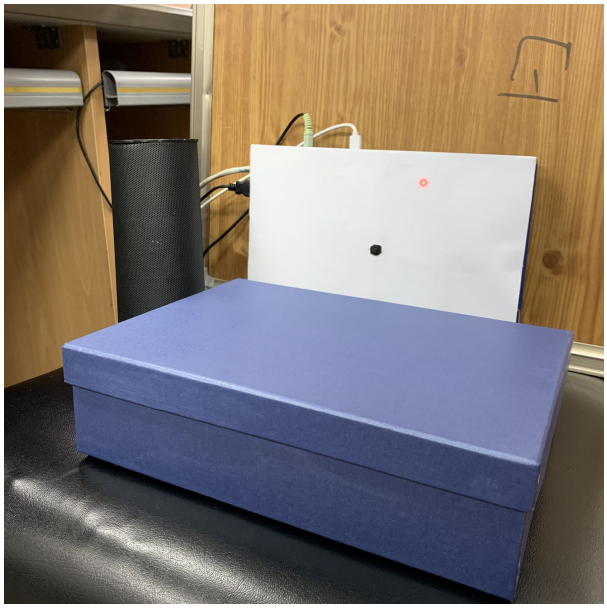




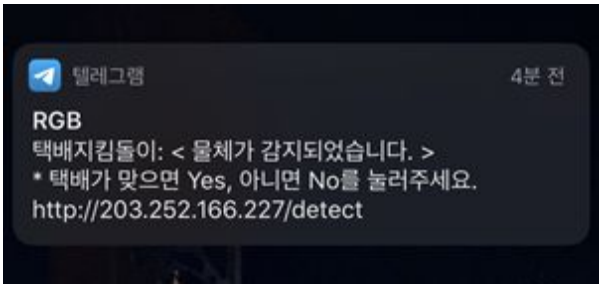


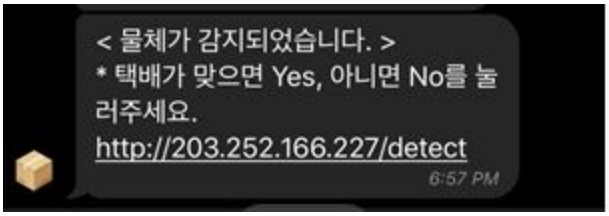
텔레그램 봇 추가 (개인 대화 & 그룹 대화 모두 가능)

물체감지 X → 물체감지 O(상단에 LED표시)

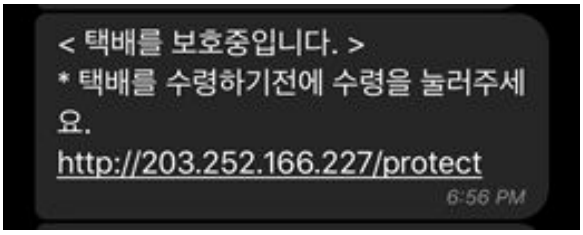


10초간 물체가 감지 될 경우 텔레그램 메시지 전송.



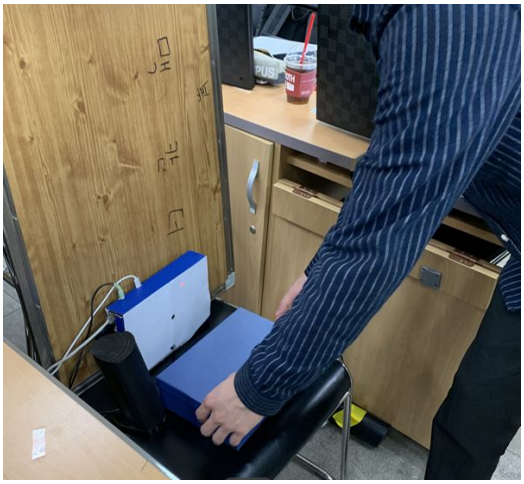


택배 확인 페이지URL → YES 선택 → **도난 방지 모드 실행 & 메세지 전송**



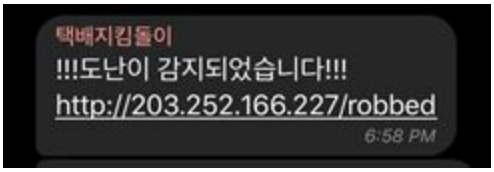
도난 감지 시 LED 점멸 및 경고음 발생 & 텔레그램 메세지 전송.

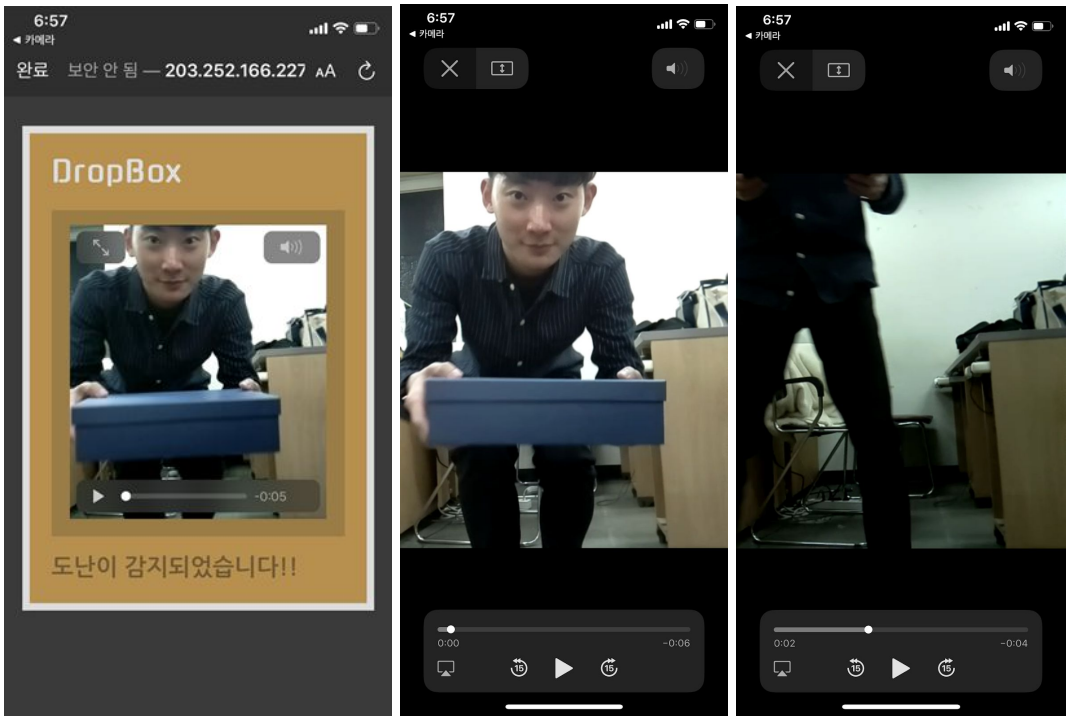
+ 외부 사운드 출력은 오디오 잭을 사용하는 스피커를 사용했지만 추후 USB스피커로 업그레이드가 가능하다.



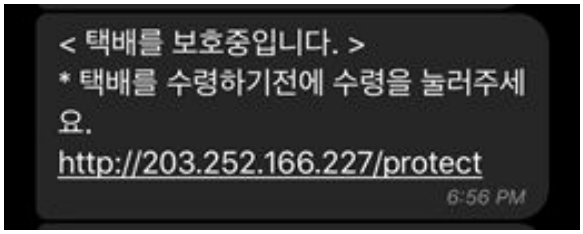
도난알람과 페이지 (도난 즉시 동영상을 촬영한다. 가져가는 사람의 얼굴도 찍힌다.)

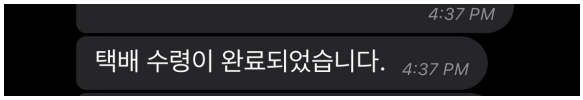
+ 센서 모듈의 인식 길이가 짧아서 큰 박스의 경우 제대로 구별이 아직은 힘들지만 추후 센서 업그레이드 시 더 좋은 결과를 기대해 볼 수 있을 것 같다.



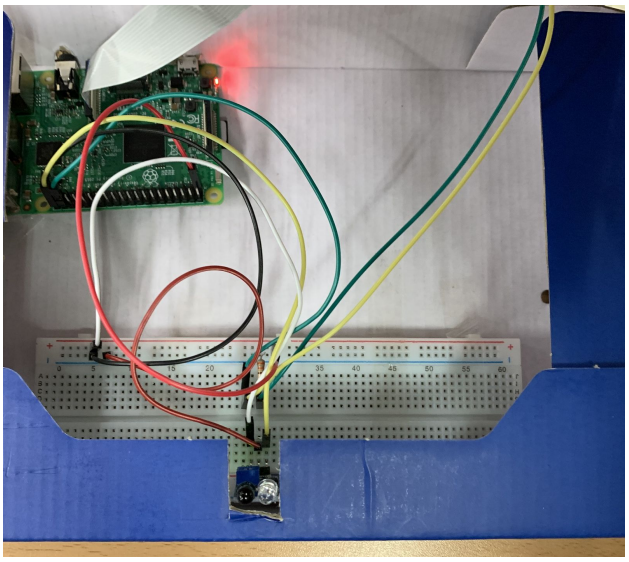


택배 수령하기 (주인은 도난 감지에 걸리면 안되므로 수령버튼을 누르고 택배를 회수한다.)



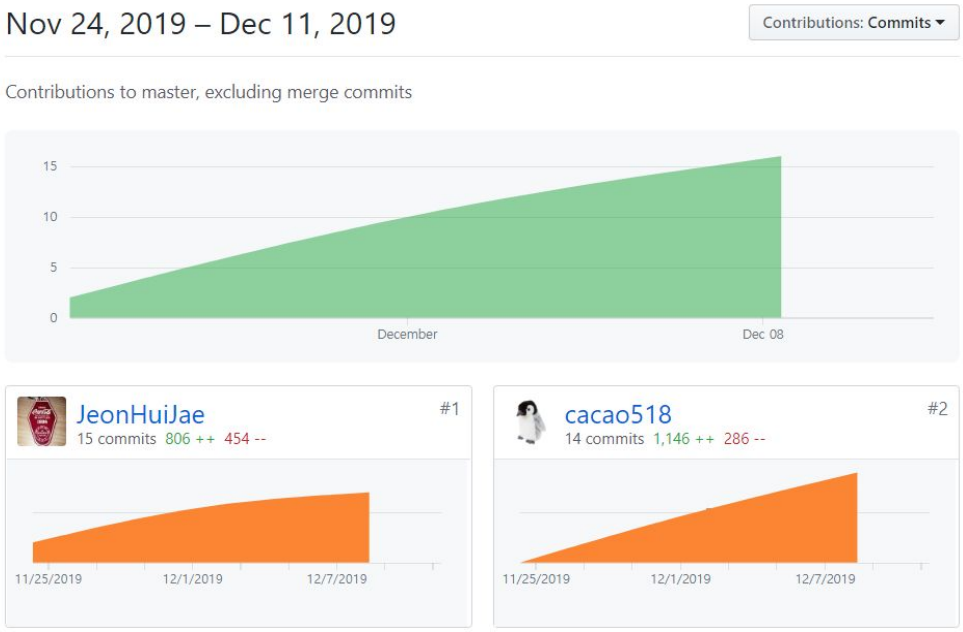


2. 브래드 보드 및 박스 내부



- 각자의 기여도

1. Github contribution



https://github.com/JeonHuiJae/Embedded_15

2. 각자의 역할

문병준: 기본구조 설계, 플라스크 함수와 스레드 함수 구현, 파이카메라 동영상 촬영과 웹 전송, 텔레그램 연동, 모바일용 웹 html

전희재: html 디자인, 적외선센서 인식과 LED 출력, 파이카메라 사진 촬영 연결, 브레드보드 설계, 외부 사운드 출력, 동영상 인코딩, GitHub

● 기말프로젝트를 통하여 배운 점

1. 기본적인 센서를 활용하여 조금이라도 더 실생활에 도움이 될만한 무언가를 만들 수 있다는걸 알게 되었다.

2. Github를 이용해 팀 프로젝트 진행을 하면서 종종 발생하는 커밋 오류에 대한 해결책을 찾았고, 그런 과정에서 활용하는 방법을 더 잘 익힐수 있게되었다.

3. Github를 이용하는것이 로컬에 저장해두는것보다 훨씬 안전하다는것을 몸소깨달았다. 팀프로젝트를 진행하다 SD카드의 문제가 생겼고, 그 결과로 그동안 진행했던 프로젝트내용이 모두 삭제됐다. 그 후로 Git에대한 소중함을 알게되었다.

4. pycamera로 촬영후 결과물은 무조건 static이라는 이름의 폴더에 저장 해야 된다는 것을 깨달았다.

5. flask 서버를 돌리면서 다른 작업을 하기 위해선 쓰레드를 이용하여 서버를 돌린뒤 다른 작업도 쓰레드를 통해 작업해야 한다는 것을 알았다.

6. pygame 이라는 모듈을 임포트해서 사운드를 간단하게 출력 할 수 있는 방법을 찾아 활용해볼수 있었다.

7. 텔레그램을 이용하면 간단하게 사용자에게 메시지를 송신 할 수 있는 것을 알았다. 카카오톡 챗봇보다 방법이 더 간단하고 다룰 수 있는 기능이 더 자유롭다.

8. 텔레그램에서는 그룹채팅방에도 챗봇을 추가할수 있었다. 이런 기능을 활용해 추후에 파이썬을 이용한 챗봇을 만들어보고자 한다.

9. h264파일을 웹에 전송하기 위해 파일 변환을 자동으로 해야했고, 파이썬 코드 내에서call함수를 이용해 Mp4Box를 이용하여 h264 파일을 mp4파일로 인코딩 할 수 있는 법을 알아냈다..

10. @app.route 을 이용하여 flask에서 주소를 바꿔주면 파이썬 코드에 명령을 내릴 수 있는 것을 알았다.

11. global 키워드가 정말 중요하다는 것을 알았다. 전역 변수를 잘 활용하면 스레드끼리 변수를 공유하여 유용하게 쓸 수 있었다.

12. datetime을 이용하여 현재 시간을 알아 낼 수 있었고 그것을 이용해 동영상 파일을 새로 저장하고 불러오게 할수 있었다.

13. html 에서 viewport를 이용하면 모바일 웹페이지에서 디바이스 크기에 맞는 화면을 제공 할 수 있는 것을 알았다.

14. app.after_request를 이용하여 html내 캐시를 지우면서 사진이나 동영상을 띄울때 갱신하는데 도움이 되는 것을 알았다.

15. flask 웹 서버를 연속으로 돌리면 프로세스가 꺼지지 않고 누적되어 남아 있는 것을 알았다. 그래서 종종 오류가 발생하였고, kill 명령어를 통해 남아있는 서버 프로세스를 종료한 뒤 다시 서버를 돌려야 하는 것을 알게 되었다.

16. 파이썬코드에서 전송하는 templateData는 html코드 내에서 {{ }} 안에 변수 이름을 집어넣으면 사용할 수 있는 것을 알았다.

17. 간단한 것이지만 datetime 을 불러오고 나서 str()을 사용해서 스트링 타입으로 바꾼뒤 call명령을 해야 하는 것을 알았다.

18. os.system을 사용하면 쉘에서 사용하는 명령을 파이썬코드에서 사용해 리턴 값을 반환 받을 수 있는 것을 알았다. 실제 코드에서는 쓰다가 지웠는데, call명령이 이와 비슷한 원리이다.