WhoRU

Face Recognition Verification System for Sharing Car

System Design Document

March 29, 2020

Version 1.0

Table of Contents

1. 프로젝트 소개

1.1. 작품 제목

WhoRU

1.2. 프로젝트 기간

2020.02.28 ~ 2020.03.28

1.3. 팀원

국민대학교 자동차 IT 융합학과 홍희정
서울시립대학교 기계정보공학과 박진석
서울과학기술대학교 기계자동차공학과 윤승한

2. 작품 개요

2.1. 개발 배경

다가오는 모빌리티 시장에는 퍼스널 모빌리티 뿐만 아니라 차량 공유 서비스(카쉐어링 서비스)도 진출했다. 차량 공유 서비스는 필요할 때에 원하는 시간만큼만 차량을 쓰는 서비스이다. 최근 스마트폰 어플을 통하여 계정 등록만 하면 간편하게 차량을 예약하여 사용할 수 있는 차량 공유 서비스 시장이 커지면서, 미성년자 등 무면허 운전이 급증하였다. 이는 기존의 렌터카와는 달리 차량을 인수하고 반납하는 과정을 모두 무인으로 운영하여 계정의 회원 정보와 실제 사용자가 일치하는지 알 수 없는 취약점 때문이다. 그래서 우리는 차량 공유 서비스에서의 무면허 및 명의 도용 방지를 막기위해 '얼굴인식 본인인증 절차'를 추가하고, 차량마다 이 인증 장치를 통하여 start button(시동 버튼)을 제어하는 'WhoRU'를 제시한다.

2.2. 기대효과

추후에 WhoRU 제품을 기존의 차량에 쉽게 장착할 수 있고 별도의 전원 없이 제어할 수 있게 발전시킬 수 있다. 차량의 OBD 단자와 연결하고 차량의 시동을 제어할 수도 있다. 이러한 방식으로 기존에 있는 공유 차량에 적용하며 WhoRU 를 상업화한다면 얼굴인식을 통해 무면허/미성년자들의 무면허 운전을 방지할 수 있다.

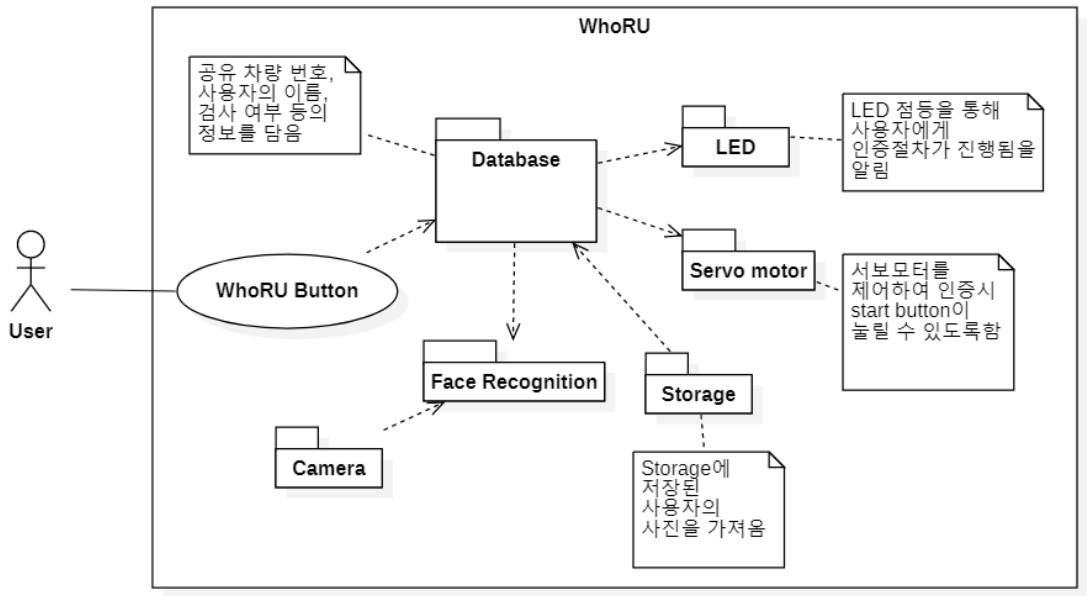
WhoRU 를 이용한다면 카 셰어링 업체 측에서는 어플을 통해 차량을 예약한 사람의 정보와 실제 운전자가 일치하는지 항상 확인할 수 있다. 이로써 업체는 기존의 어플을 통해 편리한 무인 차량 대여 서비스를 유지하고 위험부담은 줄일 수 있다. 즉, 공유 차량 서비스의 최고의 장점인 편리함을 유지할 수 있게 된다.

마지막으로 카 셰어링 업체에서는 무면허 운전자뿐만 아니라 현재 이슈 되고 있는 무보험 운전자들로 인한 차량 사고 방지 대책으로 'WhoRU 설치'를 내세울 수 있다. 이렇게 위험 부담이 줄어들게 된다면 WhoRU 를 설치한다는 것은 차량을 대여할 때의 보험료가 인하되는 요소로 작용할 것이라고 예상된다. 이로써 사용자 또한 '보험료 인하'라는 혜택을 누릴 수 있을 것이라고 기대된다.

3. 작품 설명

3.1. 주요 동작 및 특징

작동 Youtube: https://www.youtube.com/watch?v=tI2bRzswx5U&feature=youtu.be

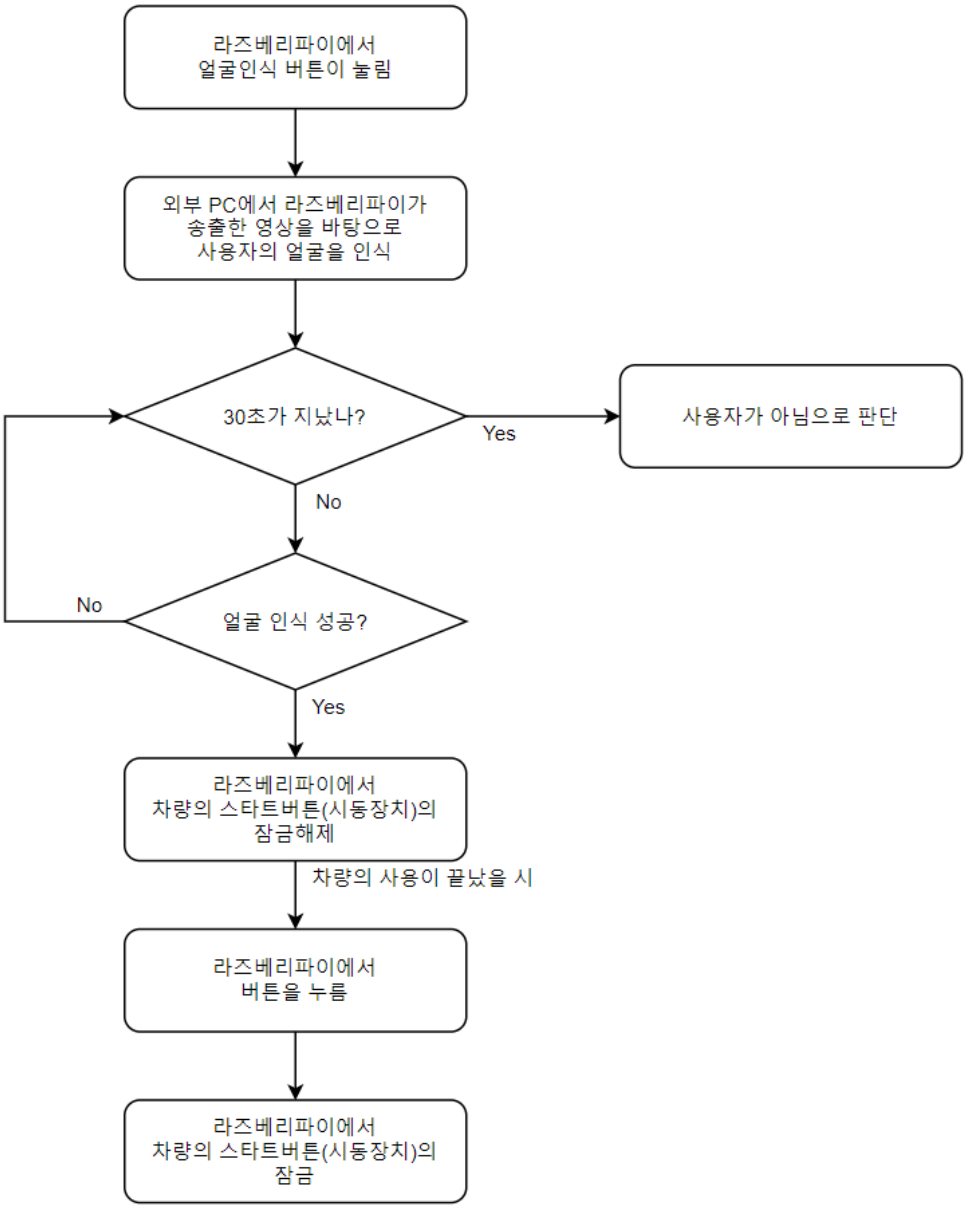


a. use-case diagram

공유 차량에 대해 WhoRU-Device 를 장착하고 Server 와 연동하여 start button(시동 버튼)을 제어하도록 설계하였다. 사용자가 공유차량을 예약할 때 면허증 사진을 등록한다. 이렇게 저장된 사진은 Storage 에 저장되고, 사용자가 공유차량을 이용할

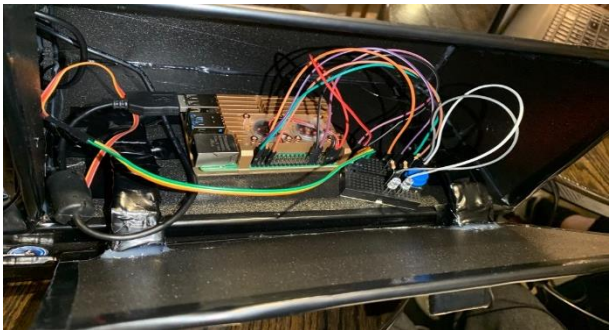
때 WhoRU-Device 의 버튼을 눌러주면, 카메라를 통해 사용자의 얼굴을 실시간으로 받아온 뒤 Storage 에 저장된 사진과 얼굴 이미지를 비교한다. WhoRU-Server 에서 비교를 마친 후 사용자 인증이 성공적으로 이루어지면 파란색 LED 가 켜지며 start button 을 막고 있던 잠금 장치가 열리게 된다. 사용자가 공유 차량 이용을 마친 후 버튼을 다시 눌러주면 LED 가 꺼지며 잠금 장치는 다시 닫히게 된다.

3.2. 전체 시스템 구성



b. flow chart

c. 차량의 가상 계기판 모델



d. 라즈베리파이와 각종 구성 장치들

3.3. 개발 환경(개발 언어, Tool, 사용 시스템 등)

항목		내용
서버	Tool	Firebase
	OS	Ubuntu 18.04 (Linux)
	개발 언어	python
H/W	Devices	Raspberry pi 4, Camera (savitmicro VIJE Q-800), Servo motor, LED, button, 가상 계기판
	Tool	Pycharm, soildworks
	개발 언어	python

4. 단계별 제작 과정

4.1. Overview

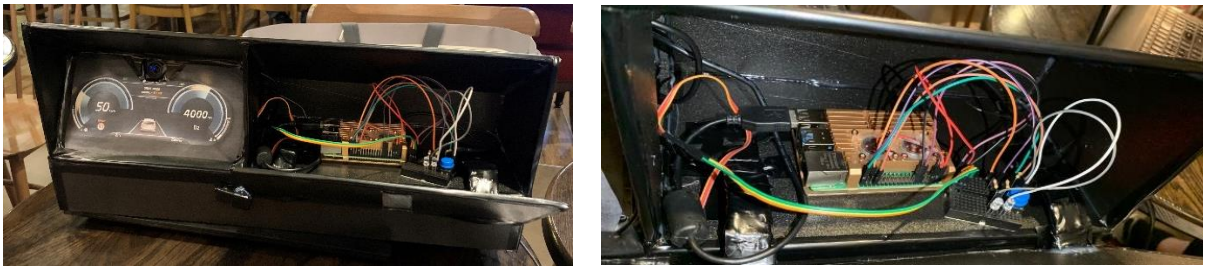
제작과정은 크게 WhoRU-Device 와 WhoRU-Server 로 나눠진다. 차량에 장착되는 WhoRU-Device 는 Raspberry pi 를 메인 보드로 사용하고 camera, servo-motor, led 등을 연결하여 차량 고유의 원격 장치를 구성하였다. WhoRU-Server 는 face recognition 작업을 진행하는 부분으로 우리는 GPU 를 사용하는 pc 를 server 로 하고 프로젝트를 진행했다.

4.2. 외관 Hardware 구성

WhoRU 작동 방향을 보여주기 위해 가상 계기판을 제작하였다. 아래의 그림은 솔리드웍스 툴을 이용하여 가상 계기판을 설계한 그림이다. 좌측에 계기판이 있고 우측 공간 안에는 라즈베리파이가 들어간다. 상단에는 led 가 연결되고, 하단에는 서보모터를 이용한 start button(시동 버튼) 잠금 장치가 연결된다. WhoRU 시작 버튼도 상자 밖에서 누를 수 있도록 구성하였다.

e. 가상 계기판 설계도

f. 초기 가상 계기판 바디



g. 가상 계기판 완성 사진

위와 같이 우드락으로 계기판 모델을 제작하여 카메라의 적합한 위치 파악과 완성된 시스템의 프로토타입을 만들었다.

4.3. Device (Raspberry pi)

4.3.1. Firebase

- Setting

라즈베리파이에서 `firebase`를 사용하기 위해 관련 패키지들을 설치해준다.
라즈베리파이에서 `python`으로 작업하기 때문에 `firebase-admin`을 추가로 설치해 주어야 한다.

```
pip install firebase
```

```
pip install firebase-admin
```

- Thread

라즈베리파이는 지속적으로 Firebase의 데이터베이스에서 확인을 해야 한다.
데이터베이스에서 데이터를 읽는데 걸리는 시간이 1초정도로 느리다. 하나의 프로세스에서 작업할 경우, 원활한 진행에 문제가 발생하였기 때문에 데이터베이스 읽은 과정은 따로 스레드를 만들어서 처리를 해주었다.

- wifi

라즈베리파이를 무선네트워크 연결을 하는 데 연결이 자꾸 끊기는 현상이 발생했다.
라즈베리파이를 부팅한 후 다음 명령어를 입력하면 이 현상을 방지할 수 있다.

```
sudo iw reg set US
```

4.3.2. **Camera**

- Motion

Motion 패키지는 카메라 모듈과 라즈베리파이를 통하여 IP 카메라 기반의 CCTV 를 만들어 주는 패키지이다. Motion 패키지의 일부 기능 중 하나인 카메라에서 받은 실시간 영상을 HTML 을 통하여 영상을 전송하는 기능을 사용하였다. 그리고 Motion 패키지는 라즈베리파이의 전원이 켜지는 즉시 백그라운드에서 자동으로 실행이 된다. 여럿 카메라 패키지 중 Motion 을 쓴 이유는 사용자가 차량에 탑승하는 즉시 차량에서 자동으로 카메라를 활성화를 시켜야 하므로 이 패키지를 쓰게 되었다.

- Setting

```
sudo apt-get install motion
```

위의 명령어로 Motion 패키지를 설치하여 준다.

```
# set to 'yes' to enable the motion daemon
start_motion_daemon=yes
```

/etc/default 디렉토리 안에 motion 이라는 파일을 위의 내용대로 수정하여 준다.

```
# Start in daemon (background) mode and release terminal.
daemon on
```

/etc/motion 디렉토리 안에 motion.conf 파일을 위와 같이 수정하여 준다. 위의 작업은 라즈베리파이가 켜지면 자동으로 백그라운드에 motion 이 실행되게 해주는 기능이다.

4.3.3. **Devices**

- Servo motor

h. lock start button **i. unlock start button**

얼굴인식을 통해 사용자 인증에 성공하면 start button(시동 버튼)의 잠금 장치를 서보 모터를 통해 열어준다. 즉, database 의 'approved' 값이 1 로 바뀌면 작동하게 되며, 한번 더 눌러주면 시스템이 초기화되며 잠금 장치가 닫히게 된다.

- Led

UI/UX 관점으로 사용자에게 얼굴 인증이 진행되고 있음을 알려준다. 사용자가 WhoRU-Device 버튼을 눌러주면 노란색 불이 들어오며 얼굴인식이 진행된다. 이 후 인증이 완료되면 파란 불을 켜준다. 공유 차량 이용 후 버튼을 다시 눌러주면 모든 불이 꺼지게 된다.

LED 제어는 firebase 의 flag 값 기반으로 제어한다.

a. 얼굴인식이 진행 중인 모습 **b. 얼굴인식이 완료된 모습**

4.4. Server

4.4.1. Firebase

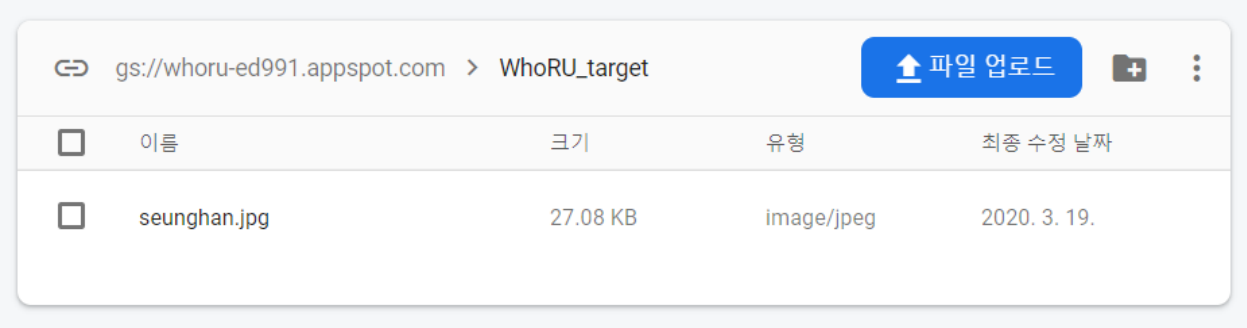
- Database

Firebase 의 database tree 는 다음과 같다. Carlist 아래에 공유 차량의 번호를 적어 두고, 각 차량 번호 아래에는 'approved', 'request', 'username'이 있다. 사용자가 WhoRU-Device 의 버튼을 누르면, 해당 디바이스와 연결된 차량번호 하위 항목의 'request'값이 1 로 바뀐다. WhoRU-Server 에서 얼굴인식이 성공적으로 이루어지면 'approved'값이 1 로 바뀐다. 공유 차량 이용 후 WhoRU-Device 의 버튼을 다시 눌러주면 'approved'와 'request' 값이 0 으로 바뀐다.

j. database tree

- Storage

Firebase 의 Storage 에는 사용자의 운전면허 사진이 등록된다. 저장된 사진은 WhoRU-Server 가 작동되면서 해당 사용자의 이름과 일치하는 이미지를 불러와 실시간 촬영되는 사용자의 이미지와 비교하여 얼굴을 인식한다.



k. storage file

Storage 에 저장된 사진은 url 형태로 저장되며 이를 Server 에서 jpg 파일로 변환시켜 불러온다. 이를 위해 아래의 패키지를 설치해준다.

```
pip install image
```

4.4.2. Face Recognition

- face_recognition

face_recognition 패키지는 얼굴 인식에 KNN (k-nearest-neighbors) 알고리즘을 사용한다. 저장된 인물 사진을 기반으로 deep metric learning 을 활용하여 판별해야 하는 사람의 얼굴을 인식하고 그 사람의 이름을 알려준다. 우리는 Adam Geitgey 가 작성한 face_recognition 패키지에서 ip-camera 를 통해 받은 실시간 화면과의 얼굴 인식을 하는 기능을 활용하였다.

- Deep metric learning

Deep metric learning 은 학습을 할 때 '분류'를 하는 것이 아니라, 같다/다르다를 나타내는 distance 를 계산하는 metric 을 사용하는 방식으로 문제를 바꾼다. 이 학습법은 단순히 '맞다/틀리다'가 아닌 입력 값이 다수인 얼굴 인식에 많이 쓰인다.

- Setting

Face recognition 을 실행하기 위하여 Ubuntu 환경에 scikit-learn, opencv, numpy 패키지를 설치해준다.

```
pip3 install scikit-learn
```

```
pip3 install numpy
pip3 install opencv-contrib-python
```

또 라즈베리파이에서와 마찬가지로 firebase 와 연동하기 위해 관련 패키지를 설치해준다.

```
pip install google-cloud-storage
pip install firebase
pip install firebase-admin
```

- 얼굴인식 민감도 조절

tolerance 값은 민감도에 영향을 주는 변수이다. 이 값이 낮을수록 얼굴인식이 민감해지고 커질수록 둔감해진다. 단, 값이 너무 낮아지면 조도, 얼굴의 각도 등 여러 변수들이 얼굴인식에 영향을 미쳐 얼굴인식이 잘 안될 수 있다. 기본 값은 0.45 이다.

우리는 여러 인물들의 사진을 기반으로 테스트 해본 결과 0.4 정도의 값이 적절하다고 판단되어 0.4 로 설정하였다.

```
217    def compare_faces(known_face_encodings, face_encoding_to_check, tolerance=0.4):
218        """
```

/face_recognition/api.py 라는 파이썬 파일에 위와 같이 tolerance 값을 변경할 수 있다.

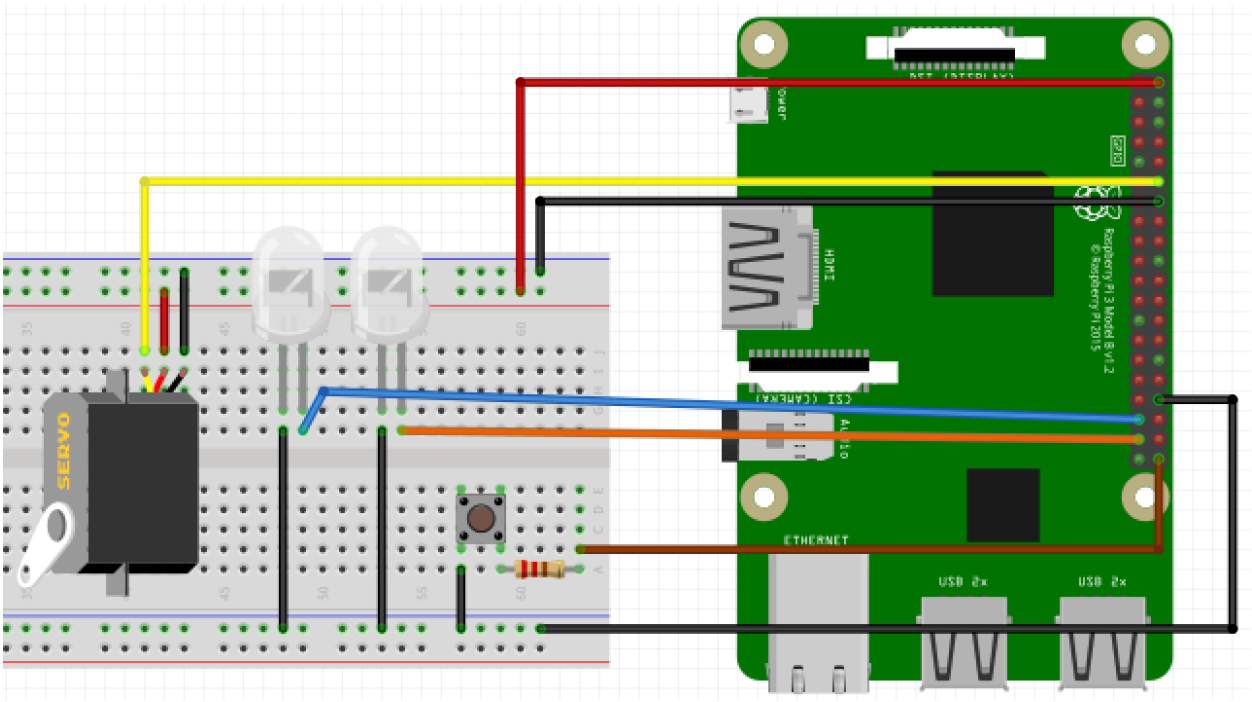
```
42    def test_image(image_to_check, known_names, known_face_encodings, tolerance=0.4, show_distance=False):
43        unknown_image = face_recognition.load_image_file(image_to_check)
```

```
99    @click.option('--tolerance', default=0.4, help='Tolerance for face comparisons. Default is 0.6. Lower this if you get multiple matches for
100   @click.option('--show-distance', default=False, type=bool, help='Output face distance. Useful for tweaking tolerance setting.')
```

그와 동시에 '/face_recognition/face_recognition_cli.py' 에도 tolerance 값을 같이 변경해주어야 한다.

5. 기타

5.1. 회로도



n.회로도

5.2. 소스코드

Github: https://github.com/ParkJinSuk/WhoRU

5.2.1. WhoRU_Device.py (Raspberry pi)

```
'''
2020.03.21
버튼이 눌리면 firebase database 에 있는 'request' 값을 1 로 바꿈
2020.03.22.
'approved'값이 1 이 되면 servo motor 제어
'''
import threading, requests
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
import RPi.GPIO as GPIO
import time

pin_servo_motor = 18 # GPIO.BCM
pin_switch = 21 # GPIO.BCM
pin_led_yellow = 19 # GPIO.BCM
pin_led_blue = 26 # GPIO.BCM
```

```
# red_led = port 12
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin_led_yellow, GPIO.OUT)
GPIO.setup(pin_led_blue, GPIO.OUT)
GPIO.setup(pin_switch, GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.setup(pin_servo_motor, GPIO.OUT)

db_url = 'https://whoru-ed991.firebaseio.com/'
cred = credentials.Certificate("myKey.json")
db_app = firebase_admin.initialize_app(cred, {'databaseURL': db_url})
ref = db.reference()

#pin_servo_motor = 12 # GPIO.BOARD

#p = GPIO.PWM(pin_servo_motor, 50)
p = GPIO.PWM(pin_servo_motor, 50)

p.start(0)

cnt = 0
pwm = 0
switch = 0
flag = 0
request = 0
input_state = 0
input_state_pre = 0
isrequest = False
GPIO.output(pin_led_yellow, GPIO.LOW)
GPIO.output(pin_led_blue, GPIO.LOW)

class get_database (threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)

    def run(self):
        print("[Thread] get database start")
        global flag
        global request
        while True:
            flag = ref.child("carlist/06 수 8850/approved").get()
            request = ref.child("carlist/06 수 8850/request").get()
            print("[Thread] flag\t: {}".format(flag))
```

```
            print("[Thread] request\t: {}".format(request))



if __name__ == "__main__":
    try:
        _time = 0
        thread_database = get_database()
        thread_database.start()
        p.ChangeDutyCycle(1)

        while True:
            input_state_pre = input_state
            input_state = GPIO.input(pin_switch)

            # APPRROVED
            if flag == '1' and request == '1':
                print("flag 1")
                # led off
                GPIO.output(pin_led_yellow, GPIO.LOW)
                GPIO.output(pin_led_blue, GPIO.HIGH)
                # motor on
                p.ChangeDutyCycle(10)
                print("angle : {}".format(pwm))


            if (input_state != input_state_pre): # switch edge detecting
                print("edge!")
                if input_state == 0:
                    switch += 1
                else:
                    pass

            if (switch % 2 == 1) and (isrequest == False):
                print("request 1")
                _time =time.time()
                ref.child("carlist/06수 8850").update({'request': '1'})
                GPIO.output(pin_led_yellow, GPIO.HIGH)
                GPIO.output(pin_led_blue, GPIO.LOW)
                isrequest = True
            if (switch % 2 == 0) and (isrequest == True):
                print("request 0")
                _time = 0
```

```
            GPIO.output(pin_led_yellow, GPIO.LOW)
            GPIO.output(pin_led_blue, GPIO.LOW)
            p.ChangeDutyCycle(1)

            ref.child("carlist/06수8850").update({'request': '0'})
            ref.child("carlist/06수8850").update({'approved': '0'})
            isrequest = False
            time.sleep(1)

        '''
        if time.time() - _time > 15: # time out
            switch += 1
        '''


except KeyboardInterrupt:
    p.stop()
    GPIO.cleanup()
```

5.2.2. **WhoRU_Server.py (face recognition)**

```
# -*- coding: utf-8 -*-


import cv2
import math
from sklearn import neighbors
import os
import os.path
import pickle
from PIL import Image, ImageDraw
import face_recognition
from face_recognition.face_recognition_cli import image_files_in_folder
import numpy as np


# FOR STORAGE
try:
    from google.cloud import storage
except ImportError:
```

```
def show_prediction_labels_on_image(frame, predictions):
    global cnt_face
    pil_image = Image.fromarray(frame)
    draw = ImageDraw.Draw(pil_image)


    for name, (top, right, bottom, left) in predictions:
        top *= 2
        right *= 2
        bottom *= 2
        left *= 2


        draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 255))
        name = name.encode("UTF-8")
        names = name.decode()
############## name compare #################
        if cnt_face < 40:
            # LED = Orange
```

```
        raise ImportError('Failed to import the Cloud Storage library for Python. Make sure "to install the "google-cloud-storage" module.')
from firebase_admin import storage
import datetime
import urllib.request
# FOR DATABASE
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db


import time


####### Global Variable ######
cnt_face = 0


CARNUMBER = None
USERNAME = None
##############################


########## firebase import name ###############
db_url = 'https://whoru-ed991.firebaseio.com/'
cred = credentials.Certificate("myKey.json")
db_app = firebase_admin.initialize_app(cred, {'databaseURL': db_url})
alldata = db.reference()


sr_buck = 'whoru-ed991.appspot.com'
sr_app = firebase_admin.initialize_app(cred, {'storageBucket': sr_buck, }, name='storage')
##########################################


ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'JPG'}



def train(train_dir, model_save_path=None, n_neighbors=None, knn_algo='ball_tree', verbose=False):
```

```
                if names == USERNAME:
                    print("true")
                    cnt_face += 1
                else:
                    print("false")
                    cnt_face = 0


                if cnt_face == 40:
db.reference('carlist').child("{}".format(CARNUMBER)).update({'approved': '1'})
                    print("Recognition successfully!")
                    # LED = Green
                elif cnt_face > 40:
                    cap1.release()
                    cv2.destroyAllWindows()
                    # LED = OFF
                    exit(0)
                now = time.time()
                if now > start_time + 15:
                    db.reference('carlist').child("{}".format(CARNUMBER)).update({'request': '0'})
                    print("Disapproved. Try Again.")
##########################################


            text_width, text_height = draw.textsize(name)
            draw.rectangle(((left, bottom - text_height - 10), (right, bottom)), fill=(0, 0, 255), outline=(0, 0, 255))
            draw.text((left + 6, bottom - text_height - 5), name, fill=(255, 255, 255, 255))


        del draw


        opencvimage = np.array(pil_image)
        return opencvimage
```

```
    X = []
    y = []
    for class_dir in os.listdir(train_dir):
        if not os.path.isdir(os.path.join(train_dir, class_dir)):
            continue

        for img_path in image_files_in_folder(os.path.join(train_dir, class_dir)):
            image = face_recognition.load_image_file(img_path)
            face_bounding_boxes = face_recognition.face_locations(image)

            if len(face_bounding_boxes) != 1:
                if verbose:
                    print("Image {} not suitable for training: {}".format(img_path, "Didn't find a face" if len(face_bounding_boxes) < 1 else "Found more than one face"))
            else:
                X.append(face_recognition.face_encodings(image, known_face_locations=face_bounding_boxes)[0])
                y.append(class_dir)

    if n_neighbors is None:
        n_neighbors = int(round(math.sqrt(len(X))))
        if verbose:
            print("Chose n_neighbors automatically:", n_neighbors)

    knn_clf = neighbors.KNeighborsClassifier(n_neighbors=n_neighbors, algorithm=knn_algo, weights='distance')
    knn_clf.fit(X, y)

    if model_save_path is not None:
        with open(model_save_path, 'wb') as f:
            pickle.dump(knn_clf, f)
```

```
if __name__ == "__main__":
    try:
        while True:
############### Get Car Number ###############
            print("####\tGet Car Number.....\t####")
            carlist = db.reference('carlist').get()
            # print(carlist.items())

            for carNumber, val in carlist.items():
                Request = db.reference('carlist').child('{}/request'.format(carNumber)).get()
                if Request == '1':
                    CARNUMBER = carNumber
                    print("Request : {} \t carNumber : {}".format(Request, carNumber))
                    print("####\tGet Car Number Complete!!\t####")

################ Call Image ################
                    print("####\tGet User Image.....\t####")
                    USERNAME = db.reference('carlist').child('{}/username'.format(CARNUMBER)).get()
                    print("username : {}".format(USERNAME))

                    bucket = storage.bucket(app=sr_app)
                    blob = bucket.blob("WhoRU_target/{}.jpg".format(USERNAME))
                    user_path = "./knn_examples/train/{}".format(USERNAME)
                    if not os.path.isdir(user_path):
                        os.mkdir(user_path)
                    img_url = blob.generate_signed_url(datetime.timedelta(seconds=300), method='GET')
```

```
    return knn_clf


def predict(X_frame, knn_clf=None, model_path=None, distance_threshold=0.5):
    if knn_clf is None and model_path is None:
        raise Exception("Must supply knn classifier either thourgh knn_clf or model_path")

    if knn_clf is None:
        with open(model_path, 'rb') as f:
            knn_clf = pickle.load(f)

    X_face_locations = face_recognition.face_locations(X_frame)

    if len(X_face_locations) == 0:
        return []
    faces_encodings = face_recognition.face_encodings(X_frame, known_face_locations=X_face_locations)

    closest_distances = knn_clf.kneighbors(faces_encodings, n_neighbors=1)
    are_matches = [closest_distances[0][i][0] <= distance_threshold for i in range(len(X_face_locations))]

    return [(pred, loc) if rec else ("unknown", loc) for pred, loc, rec in
            zip(knn_clf.predict(faces_encodings), X_face_locations, are_matches)]
```

```
urllib.request.urlretrieve(img_url, '{0}/{1}.jpg'.format(user_path, USERNAME))
                print("####\tGet User Image Comoplete!!\t####")

##########################################
                print("####\tTraining KNN classifier...\t####")
                classifier = train("knn_examples/train", model_save_path="trained_knn_model.clf", n_neighbors=2)
                print("####\tTraining complete!\t####")
                # process one frame in every 30 frames for speed
                process_this_frame = 29
                print('####\tSetting cameras up...\t####')
                # multiple cameras can be used with the format url = 'http://username:password@camera_ip:port'
                url = 'http://192.168.43.78:8081/'
                cap = cv2.VideoCapture(url)
                # 시작 시간 기록
                start_time = time.time()
                while 1 > 0:
                    ret, frame = cap.read()
                    if ret:
                        img = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)
                        process_this_frame = process_this_frame + 1
                        if process_this_frame % 30 == 0:
                            predictions = predict(img, model_path="trained_knn_model.clf")
                        frame = show_prediction_labels_on_image(frame, predictions)

                        cv2.imshow('camera', frame)
```

	if ord('q') == cv2.waitKey(10):
	cap1.release()
	cv2.destroyAllWindows()
	exit(0)
	
	else:
	pass
	time.sleep(1)
	except KeyboardInterrupt:
	print("KeyboardInterrupt!!")

5.3. 참고문헌

Firebase setting 참조

https://medium.com/@97preveenraj/image-upload-to-firebase-storage-with-python-ebf18c615f34

사진 불러오기참조

https://firebase.google.com/docs/reference/admin/python/firebase_admin.db

Firebase Storage

https://cloud.google.com/storage/docs/downloading-objects?hl=ko#storage-download-object-python

https://cloud.google.com/storage/docs/reference/libraries?hl=ko

Firebase-admin github

https://github.com/firebase/firebase-admin-python/blob/master/firebase_admin/storage.py

얼굴인식 관련 참고

https://github.com/ageitgey/face_recognition

motion 관련 참고

https://motion-project.github.io/