졸음 운전 및 음주 운전으로 인한 교통사고 예방을 위한 시스템 개발

부경대학교 컴퓨터공학전공 곽아은

부경대학교 컴퓨터공학전공 이승혜

1. 작품 제목

본 작품의 제목은 ‘졸음 운전 및 음주 운전으로 인한 교통사고 예방 시스템’이다.

2. 작품 개요

2017년부터 2021년까지 발생한 음주운전 교통사고는 총 86,747건으로, 1,573명이 사망하고 143,993명이 다쳤다. 전체 교통사고에서 음주운전 교통사고가 차지하는 비율은 8.1%로, 전국에서 하루 평균 약 48건이 발생했다. [1] 졸음운전은 이러한 음주운전보다 더 높은 치사율을 보인다. 경찰청 통계에 따르면 졸음 운전으로 인한 교통사고 사망률은 4.51%로, 음주운전으로 인한 교통사고 사망률인 2.58%의 약 1,7배를 기록하였다.[2]

이러한 상황을 고려하여 본 작품은, 음주운전 및 졸음운전을 예방하고자 하는 목적을 지닌다. 운전자의 중추신경계가 억제됨으로써 쉽게 흥분하고, 반응 속도와 판단력이 늦어진다. 음주운전자는 스스로 안전한 운전이 가능한 상태인지를 판별할 수 있는 능력이 현저히 떨어지므로 해당 상황에서는 시동을 켜고 페달에 발을 대는 것 자체가 문제가 될 수 있다. 본 작품은 해당 문제 해결을 위해 운전자가 시동을 걸고 페달을 밟기 전 음주측정 센서를 이용하여 검사를 하고, 음주 상태인 경우 시동을 끌 수 있도록 하였다.

졸음 운전자는 눈을 감고 졸 수 있기 때문에 전방 주시에 어려움이 생긴다. 그들이 위험상태를 인지하는 것은 사고가 나기 직전이나 난 후이다. 이러한 상태의 운전자를 보호하기 위해서는 운전자가 조는 경우 그들의 상태를 인지할 수 있게 하는 예방 시스템이 필요하다. 본 작품에서는 open cv를 이용한 카메라 시스템과 부저로 졸음을 감지하고 위험을 알릴 수 있도록 하였다

3. 작품 설명

본 작품에 대해 주요 동작 및 특징, 전체 시스템 구성, 개발 환경을 중심으로 설명한다. 각 섹션은 졸음운전 시스템과 음주운전 시스템을 구분하여 설명한다.

3-1. 주요 동작 및 특징

* 졸음 운전

졸음 측정 시스템을 구성하는 요소는 다음과 같다.

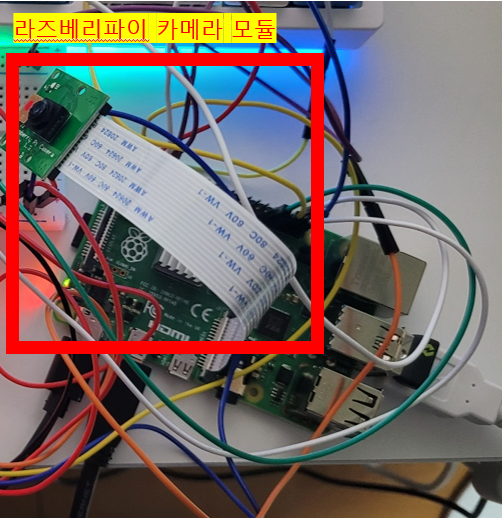


그림 1-1 졸음 측정 시스템 구성 회로

해당 시스템은 얼굴의 랜드마크를 사용하여 스트림에서 눈 깜빡임을 감지하는 방법을 사용한다. 랜드마크는 OpenCV에서 제공하는 Haar Cascade 알고리즘과 dlib의 shape\_predictor을 사용하여 추출된다. Haar Cascade는 머신러닝 기반의 오브젝트 검출 알고리즘으로 얼굴 감지에 사용된다. 먼저 이동하는 직사각형 내부 두 영역의 픽셀 합의 차이를 계산한다. 해당 직사각형의 두 영역이 각각 눈과 뺨에 위치했다면 눈이 뺨보다 더 어둡기 때문에 픽셀 합의 차이는 다른 영역보다 크다. 이러한 점을 이용하여, 어떤 영역에서 픽셀 합의 차이가 임계값을 넘으면 얼굴에 대한 Haar-Like-Feature가 된다. [3]

Haar Cascade 알고리즘을 사용하여 얼굴을 감지한 후에는 dlib의 shape\_predictor을 사용하여 랜드마크를 추출한다. 이는 이미지 영역에서 오브젝트의 두드러진 영역을 지역화하고 나타내는데 사용된다. 다음은 얼굴 감지 후 shape\_predictor을 사용하여 랜드마크 p1~p6를 추출한 결과이다.

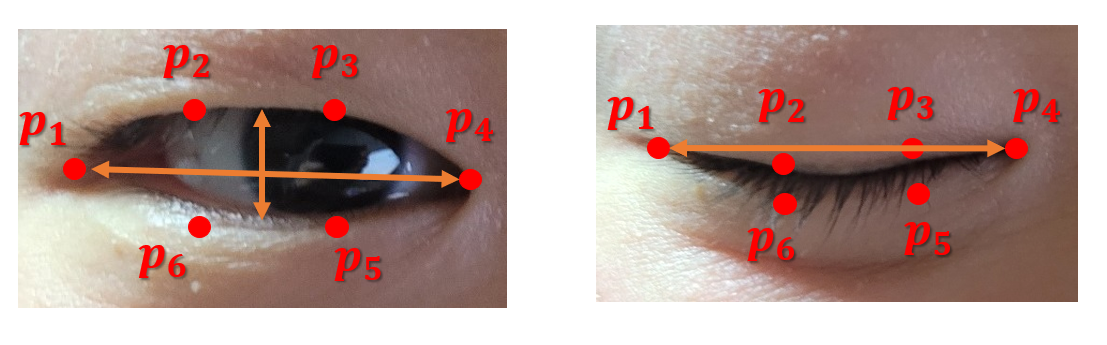


그림 1-2 눈의 랜드마크를 표시

p1~p6은 눈의 랜드마크를 의미한다. 이때 얼굴의 랜드마크를 기반으로 눈 종횡비(EAR)를 다음과 같은 방정식으로 나타낼 수 있다. [5]

수식 1-1. 눈 종횡비(EAR) 방정식

이 방정식의 분자는 눈 랜드마크의 수직 거리를 계산한다. 따라서 눈이 감기면 종횡비 EAR은 급격히 0으로 떨어진다. 하단의 그래프는 시간의 경과에 따른 눈 종횡비 값을 나타낸다. 사용자가 눈을 감았을 때 종횡비 값이 0으로 급격히 떨어지는 것을 확인할 수 있다.

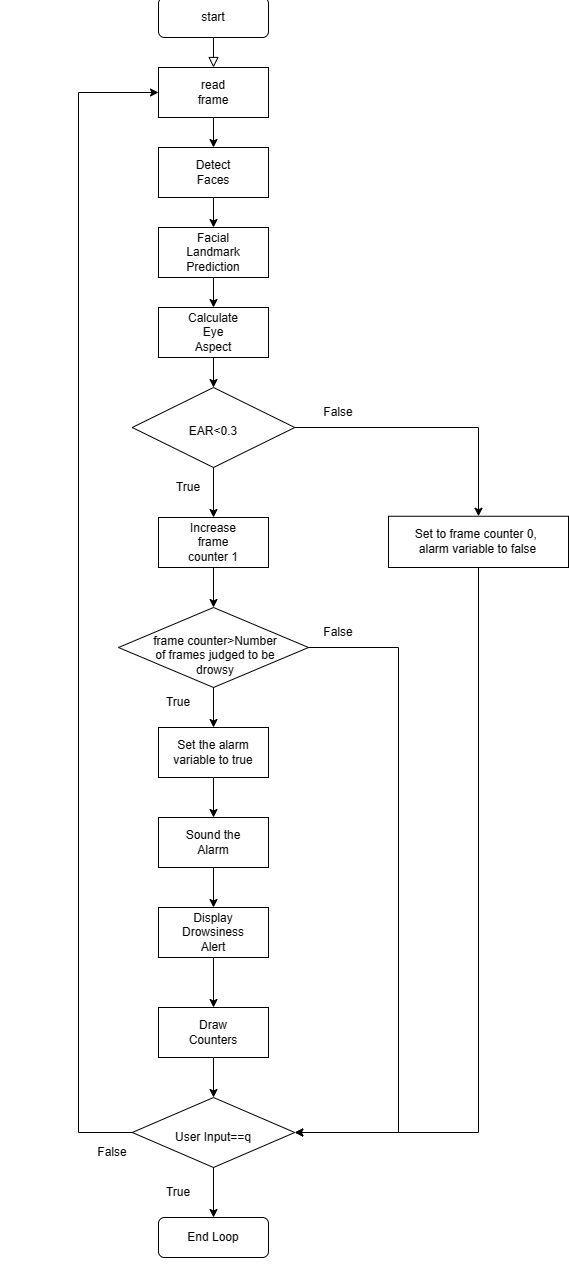


그림 1-3 졸음 운전 시스템 플로우 차트

이를 이용하면 눈의 깜빡임을 감지할 수 있다. 위 그림은 본 작품의 졸음운전 시스템의 플로우 차트이다. 본 작품에서는 임계값을 0.3으로 정하여 종횡비 EAR값이 임계값 아래로 떨어지면 눈이 감김으로 감지하도록 하였다. 눈 감김이 일정 시간 이상 지속되면 졸음 운전으로 판단하고 부저를 울린다. 이 작업은 종횡비 EAR 값이 임계값 미만인 프레임 수를 카운트하여 이루어진다. 본 작품에서는 종횡비 EAR 값이 48프레임 이상 동안 임계값 미만이면 졸음으로 판단하도록 하였다.

* 음주 운전

음주 측정 시스템을 구성하는 회로는 다음과 같다.

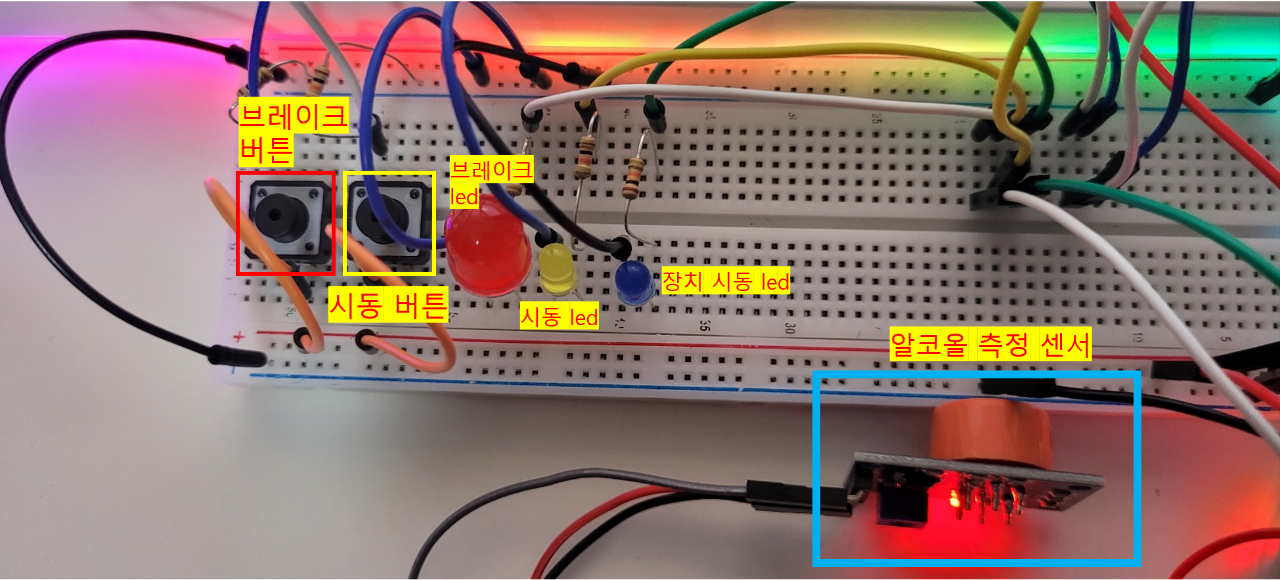


그림 1-4 음주 측정 시스템 구성 회로

사용자의 입력은 두 개의 버튼과 알코올 측정 센서로 이루어진다. 사용자가 브레이크 버튼과 시동 버튼을 누르면 최종적으로 장치 시동이 켜진다. 이때 각 상태를 표시하는 led가 존재한다. 브레이크 버튼의 클릭 여부는 붉은색 led가, 시동 버튼의 클릭 여부는 노란색 led가 표시하며, 최종적으로 장치의 시동이 켜졌는지 여부는 파란색 led가 표시한다. 즉, 각 기능이 활성화 되었는지는 각 led를 통해 확인할 수 있다.



그림 1-5 음주 운전 시스템 플로우 차트

자동차의 시동은 브레이크 페달을 밟은 후 시동 버튼을 눌러야 최종적으로 켜진다. 본 작품에서도 이러한 점을 반영하였다. 먼저 브레이크 버튼이 눌러졌는지 확인한다. 사용자가 브레이크 버튼을 누르면 브레이크 led(붉은색)가 켜진다. 그 후 시동 버튼을 누르면 브레이크가 활성화되었는지 확인한 후 시동 led(노란색)가 켜진다. 이때 브레이크 기능이 활성화는 브레이크 led의 온오프를 통해 확인할 수 있다. 마찬가지로 최종 장치 시동을 켤때 또한 또한 브레이크 led와 시동 led의 온오프 여부를 확인한다.

장치의 시동이 켜져있으면 알코올 센서가 측정을 시작한다. 사용자는 센서에 바람을 불어 알코올 수치를 측정할 수 있다. 측정한 값은 pcf8591t AD/DA 컨버터를 통해 메인보드, 즉 라즈베리 파이가 읽을 수 있는 값으로 변환된다. 이렇게 읽은 알코올 센서의 측정값이 일정 수치를 넘으면 모든 led가 꺼지며 이는 시동이 꺼짐을 의미한다.

3-2. 전체 시스템 구성

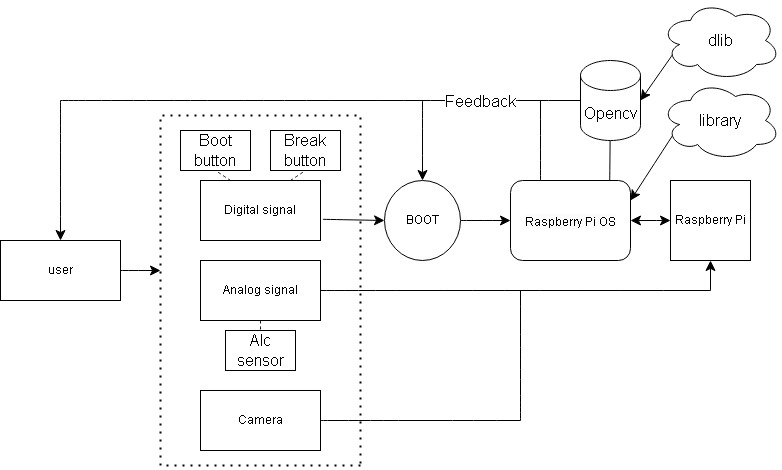


그림 2-1 전체 시스템 구성

위 구성도와 같이 사용자에게서 받아온 값을 라즈베리파이에서 처리한 후 알맞은 피드백 값을 사용자에게 출력한다. 이 과정에서 OS에 설치한 다양한 라이브러리를 사용하였다.

3-3. 개발 환경

* Raspberry Pi OS

시스템을 개발하기 위해 Raspberry PI 4 Model B를 사용하였다. Raspberry Pi OS는 라즈베리파이 하드웨어에 최적화된 Debian 기반의 무료 운영 체제이다. Debian의 APT(Advanced Packaging Tool)를 사용하여 소프트웨어의 설치, 업그레이드 및 제거를 관리했다. 그 외에도 File Manager, LX Terminal 등을 포함하여 다양하게 활용할 수 있다.

* 라즈베리파이와 아두이노의 차이점

라즈베리파이는 프로세서, 아두이노는 컨트롤러에 더 가깝다. 센서 등의 외부 기기를 제어하는 데에는 아두이노를 사용하는 것이 적합하고, 카메라 혹은 데이터 처리와 같은 복잡한 수치 계산은 라즈베리파이가 더 적합하다고 볼 수 있다. 3-1 주요 동작 및 특징에서 알 수 있듯이 ‘졸음운전 및 음주운전 예방 시스템’은 사용자의 얼굴을 인식하여 졸음 운전을 감지하기 위해 카메라를 사용해 데이터를 처리하므로 라즈베리파이를 통해 개발하였다.

* Python

컴파일 과정 없이 인터프리트가 소스 코드를 한 줄씩 읽어 곧바로 실행하는 스크립트 언어이다. 라즈베리파이에서 많이 사용하는 언어로 참고 자료가 많아 사용하게 되었다.

1. 단계별 제작 과정

본 작품의 제작과정을 단계별로 설명한다.

4-1. 졸음 운전

* 1. 카메라 작동 확인

**raspicam** 라이브러리 설치 후 다음명령어를 입력하여 카메라 작동을 확인한다.



그림 3-1 raspicam 테스트 명령어 입력

해당 명령어는 라즈베리 파이에 연결된 카메라를 통해 사진을 찍는다. 저장된 사진 파일의 이름은 test.jpg로 설정하였다.

다음과 같이 정상적으로 작동하는 것을 확인하였다.



그림 3-2 raspicam 테스트 정상 작동 확인

* 1. opencv 라이브러리 설치

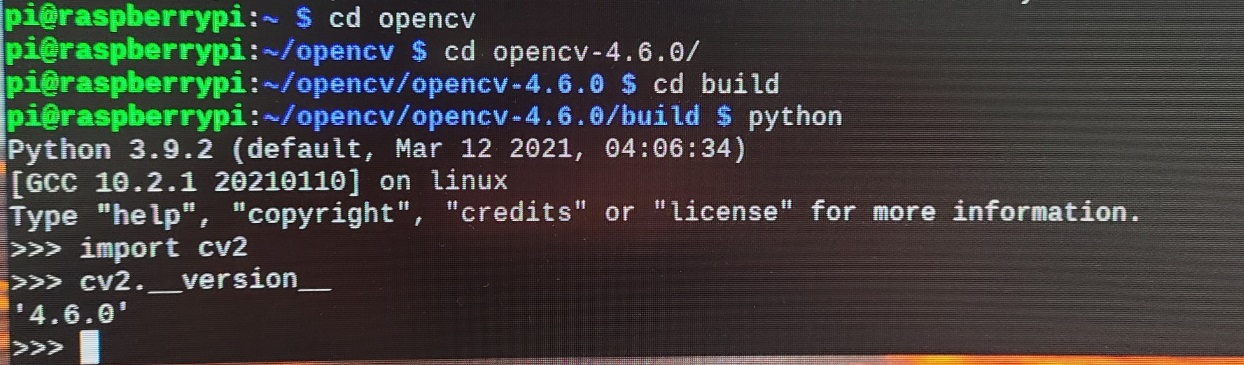


그림 3-3 opencv 설치

* 1. 버전의 opencv를 설치하였다.
  2. 눈 종횡비 수치 확인

눈을 떴을 때와 감았을 때 종횡비의 수치를 확인한다.

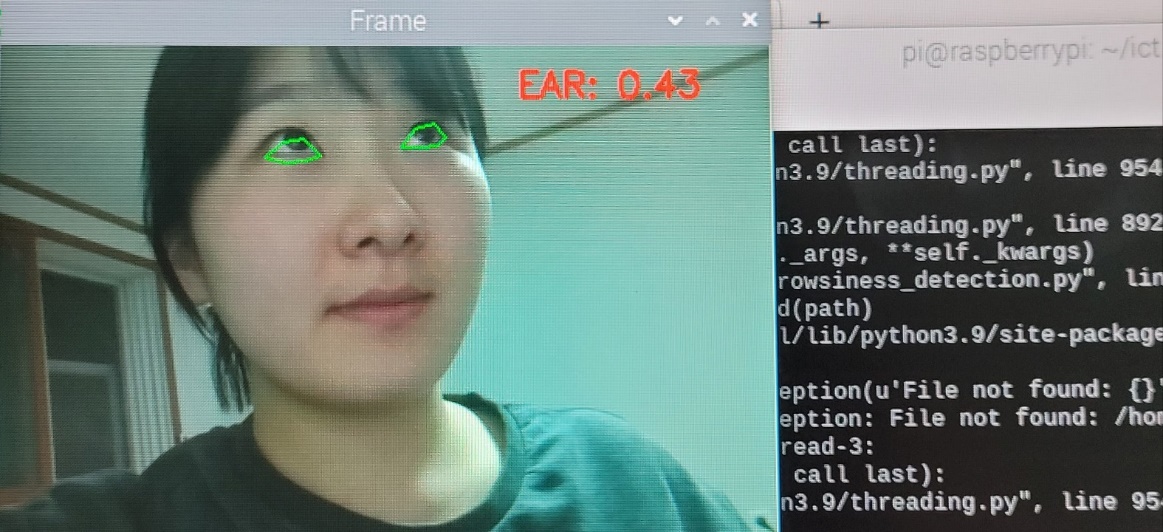


그림 3-4 눈 떴을 때 종횡비 수치

눈 떴을 때 종횡비가 0.43인 것을 확인할 수 있다.

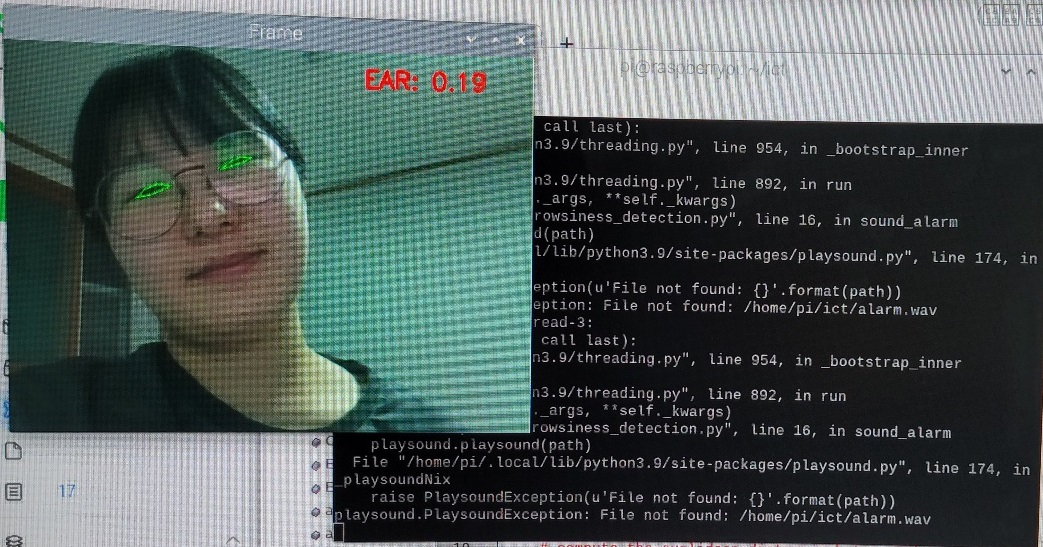


그림 3-5 눈 감았을 때 종횡비 수치

눈이 감겼을 때 수치가 0에 가까이 떨어졌다.

* 1. 부저 추가

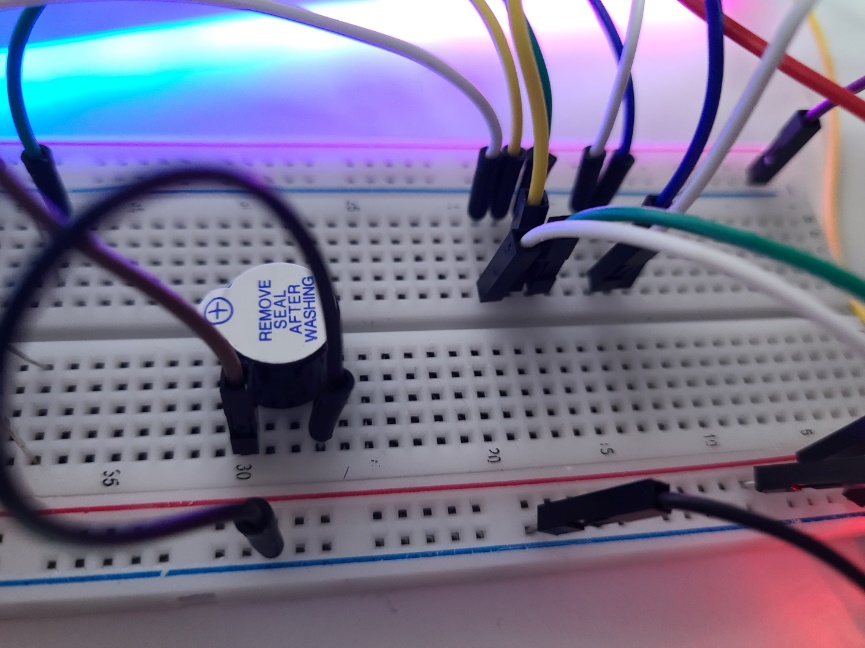


그림 4-1 회로에 부저 추가

졸음이 감지되었을 때 알림음을 울리는 부저를 추가한다.

* 1. 음주 운전
  2. 센서 작동 확인

소주병을 가까이 했을 때 알코올이 감지된다.

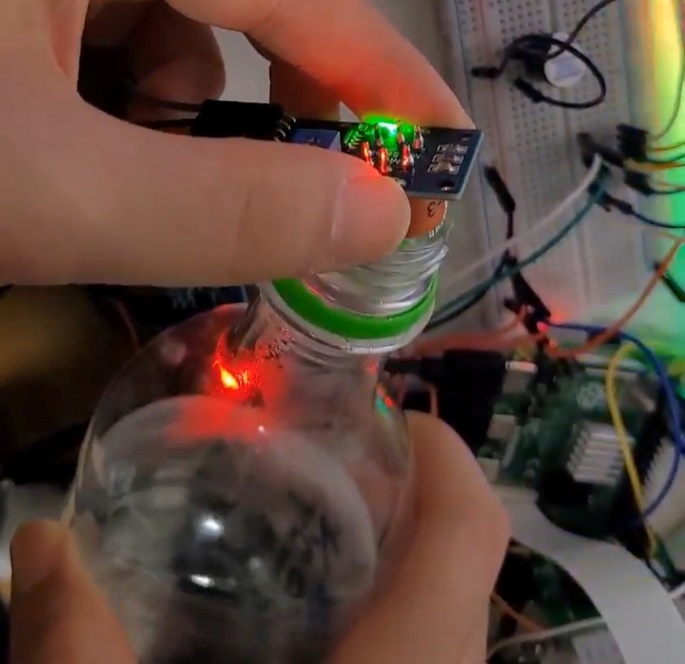


그림 4-2 알코올 센서 작동 테스트

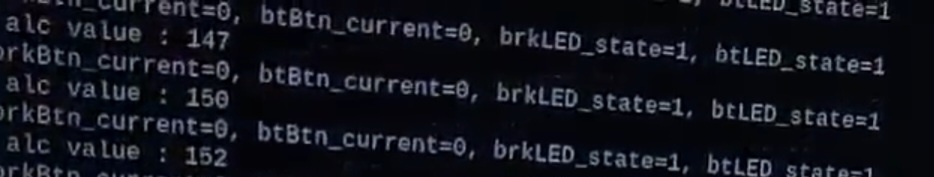


그림 4-3 알코올이 가까이 있을 때 측정 된 센서 값

알코올 수치가 150 내외로 측정되는 것을 확인할 수 있다.

* 1. 회로 구성

다음과 같이 회로를 구성한다.

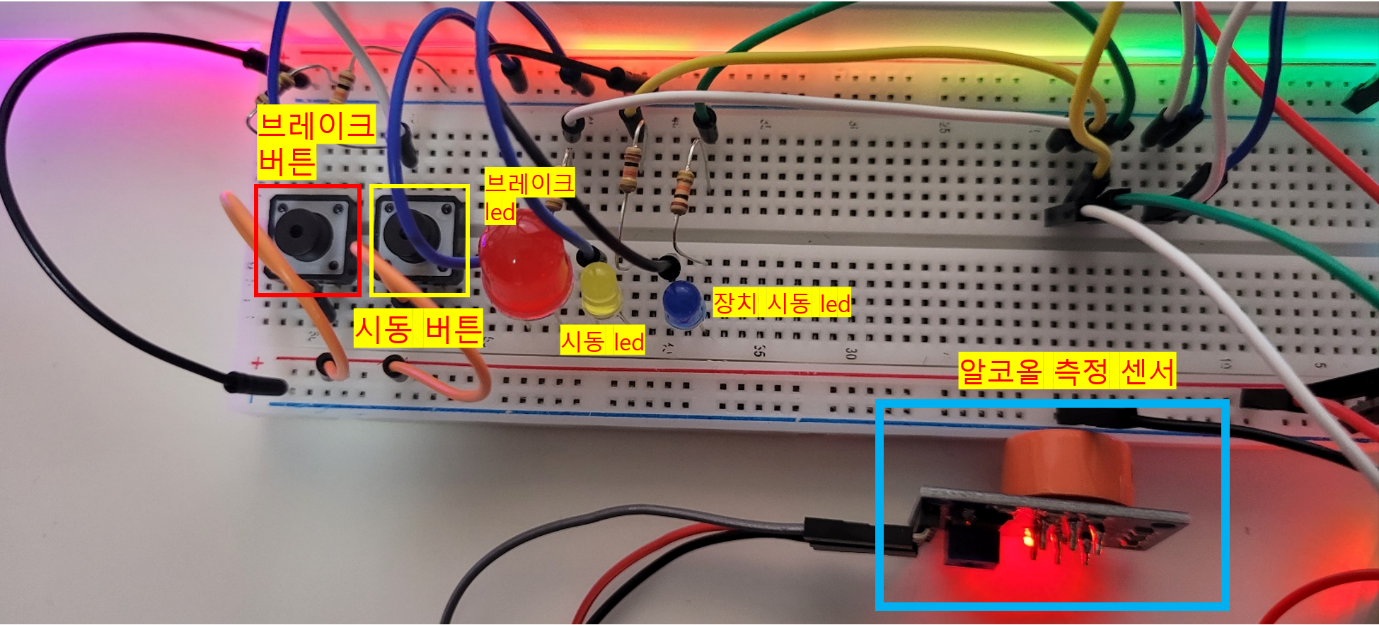


그림 4-4 음주 운전 예방 회로

* 코드 작성

각 모듈의 기능과 상호작용을 구현한다. 사용한 언어는 파이썬이다.

* 작동 확인

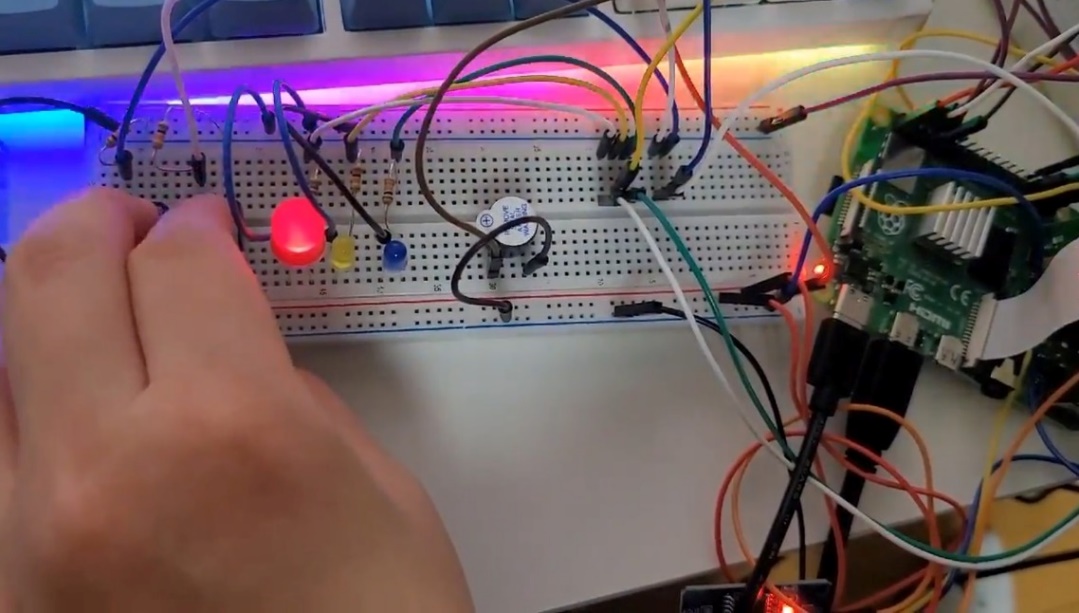


그림 4-5 음주 운전 예방 회로 브레이크 작동

브레이크 버튼을 누르고 브레이크 등이 켜지는지 확인한다.

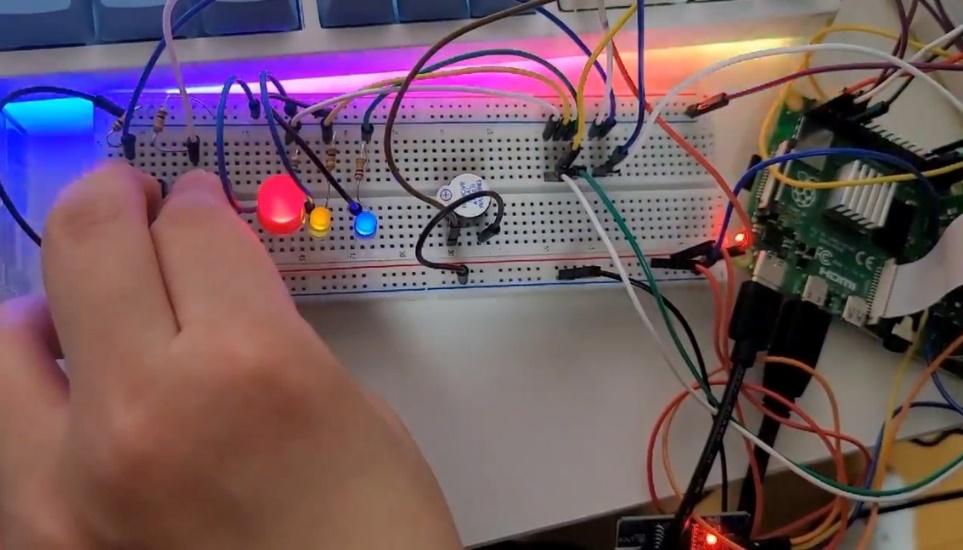


그림 4-5 음주 운전 예방 회로 시동 작동

브레이크 등이 활성화 된 상태에서 시동 버튼을 눌러 시동 등을 켠다. 브레이크 등과 시동 등이 모두 켜졌으므로 장치 시동 변수가 참이 되어 장치 시동 led를 켠다. 이때 음주 측정 기능이 활성화 된다.

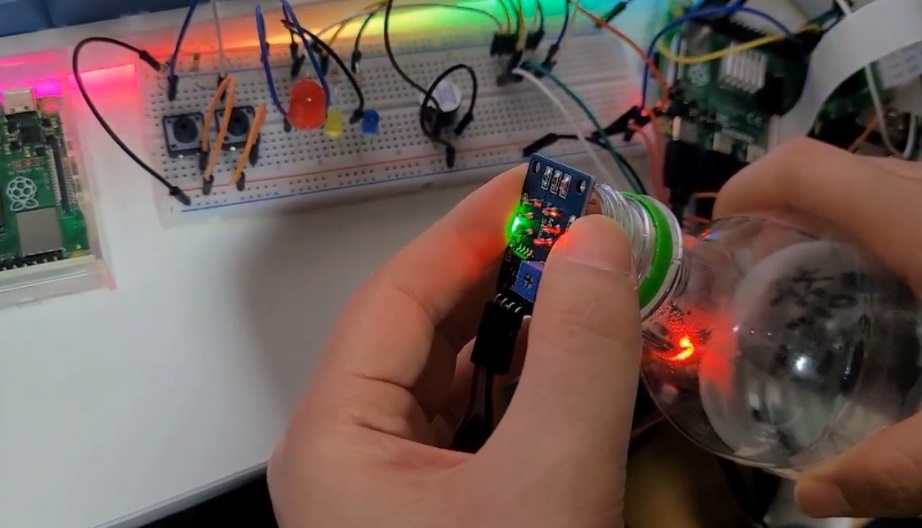


그림 4-6 음주 운전 예방 회로 음주 운전 시 시동 작동

1. 사용한 제품 리스트 (제품명, 제품 링크 포함)
   1. MQ-3(<https://www.devicemart.co.kr/goods/view?no=1327429>)



그림 5-1 MQ-3

알코올 센서 모듈이다. 음주 운전 확인을 위해 사용하였다.

* 1. pcf8591(<https://www.devicemart.co.kr/goods/view?no=12537448>)

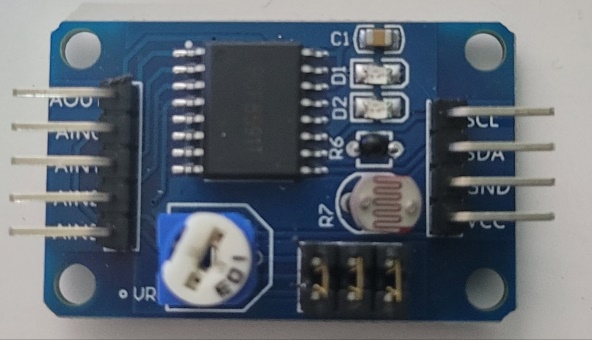


그림 5-2 pcf8591

아날로그 신호인 MQ-3의 값을 라즈베리파이에서 읽을 수 없으므로 컨버터 기능을 하는 pcf8591을 사용하였다.

* 1. LED(빨간색, 노란색, 파란색)

빨간색 (<https://www.devicemart.co.kr/goods/view?no=2851>)

노란색 (<https://www.devicemart.co.kr/goods/view?no=2852>)

파란색(<https://www.devicemart.co.kr/goods/view?no=1320882>)



그림 5-3 LED

자동차의 시동 장치를 LED를 통해 구현하였다.

* 1. 버튼(<https://www.devicemart.co.kr/goods/view?no=1361702>)

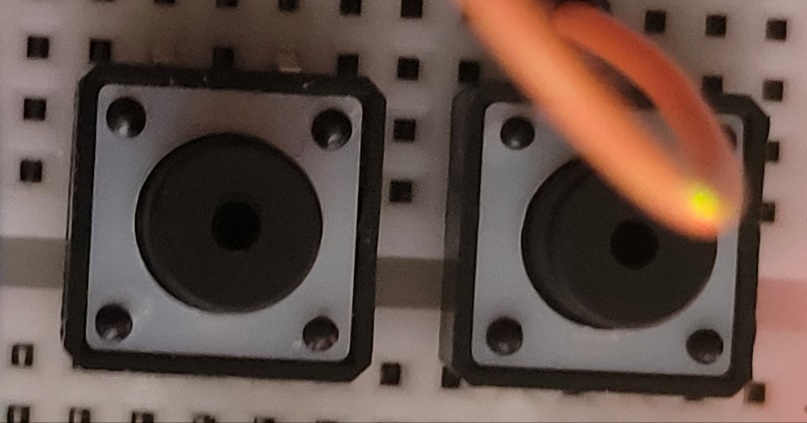


그림 5-4 버튼

시동을 걸 때에 시동 버튼과 브레이크를 구현하기 위해 사용하였다.

* 1. 부저 (<https://www.devicemart.co.kr/goods/view?no=2736>)

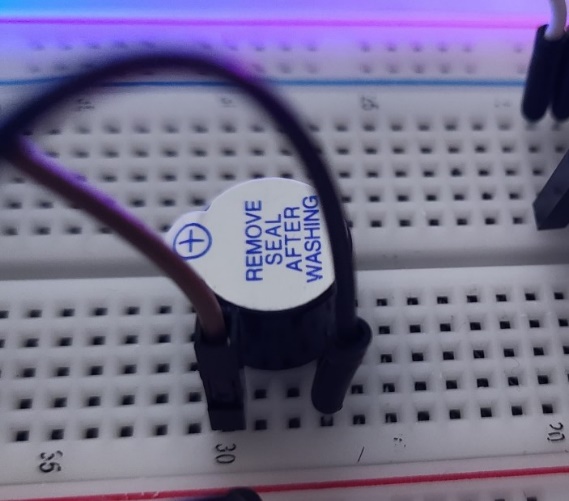


그림 5-5 부저

사용자의 졸음을 인식했을 때 부저를 울려 사용자를 깨운다.

* 1. 카메라([아두이노 라즈베리파이 카메라 모듈 (5MP, 8MP) : 알파마이크로 (naver.com)](https://smartstore.naver.com/misoparts/products/5533715543?site_preference=device&NaPm=ct%3Dlfp2u7tv%7Cci%3Dshopn%7Ctr%3Dsls_myr%7Chk%3D7c9d00b72aacb4353e5e3af8bfcf9d3635777aaf%7Ctrx%3D))

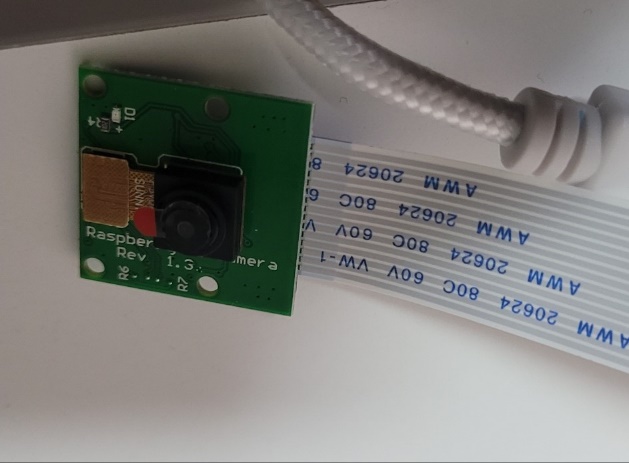


그림 5-6 카메라

졸음 운전의 여부를 판단하기 위해서 사용자의 눈을 인식하는 데에 사용하였다.

* 1. 라즈베리파이(<https://www.devicemart.co.kr/goods/view?no=12234534>)



그림 5-7 Raspberry Pi Model B 4GB

완전한 데스크탑 환경을 구축할 수 있으며 저전력으로도 잘 작동 된다. 큰 전력을 끌어 올 수 없는 차량 내부에 사용하기 적합하다.

6. 기타

6-1. 전체 회로도

텍스트, 전자제품이(가) 표시된 사진

자동 생성된 설명

6-1. 실제 장치 이미지

전자제품, 케이블, 커넥터, 어댑터이(가) 표시된 사진

자동 생성된 설명

6-3. 소스코드

|  |
| --- |
| import time  import wiringpi as wp  # import the necessary packages  from scipy.spatial import distance as dist  from imutils.video import VideoStream  from imutils import face\_utils  from threading import Thread  import numpy as np  import playsound  import argparse  import imutils  import time  import dlib  import cv2  import RPi.GPIO as GPIO  # Pin Definitions  boot\_button = 28  break\_button = 29  real\_boot\_led = 22  # green  break\_led = 25  # red  boot\_led = 24  # yellow  PCF = 120  # Setup Pins  wp.wiringPiSetup()  wp.pcf8591Setup(PCF, 0x48)  wp.pinMode(boot\_button, wp.INPUT)  wp.pinMode(break\_button, wp.INPUT)  wp.pinMode(real\_boot\_led, wp.OUTPUT)  wp.pinMode(break\_led, wp.OUTPUT)  wp.pinMode(boot\_led, wp.OUTPUT)  wp.digitalWrite(real\_boot\_led, wp.LOW)  wp.digitalWrite(break\_led, wp.LOW)  wp.digitalWrite(boot\_led, wp.LOW)  # Initial Variables  brkBtn\_previous = wp.LOW  brkBtn\_current = wp.LOW  btBtn\_previous = wp.LOW  btBtn\_current = wp.LOW  brkLED\_state = wp.LOW  btLED\_state = wp.LOW  is\_boot = 0  BUZZER\_PIN = 18  GPIO.setmode(GPIO.BCM)  GPIO.setup(BUZZER\_PIN, GPIO.OUT)  def sound\_alarm(path):      # play an alarm sound      # playsound.playsound(path)      # activate the buzzer      GPIO.output(BUZZER\_PIN, GPIO.HIGH)  def eye\_aspect\_ratio(eye):      # compute the euclidean distances between the two sets of      # vertical eye landmarks (x, y)-coordinates      A = dist.euclidean(eye[1], eye[5])      B = dist.euclidean(eye[2], eye[4])      # compute the euclidean distance between the horizontal      # eye landmark (x, y)-coordinates      C = dist.euclidean(eye[0], eye[3])      # compute the eye aspect ratio      ear = (A + B) / (2.0 \* C)      # return the eye aspect ratio      return ear  # construct the argument parse and parse the arguments  ap = argparse.ArgumentParser()  ap.add\_argument("-p", "--shape-predictor", required=True,                  help="path to facial landmark predictor")  ap.add\_argument("-a", "--alarm", type=str, default="",                  help="path alarm .WAV file")  ap.add\_argument("-w", "--webcam", type=int, default=0,                  help="index of webcam on system")  args = vars(ap.parse\_args())  # define two constants, one for the eye aspect ratio to indicate  # blink and then a second constant for the number of consecutive  # frames the eye must be below the threshold for to set off the  # alarm  EYE\_AR\_THRESH = 0.3  EYE\_AR\_CONSEC\_FRAMES = 48  # initialize the frame counter as well as a boolean used to  # indicate if the alarm is going off  COUNTER = 0  ALARM\_ON = False  # initialize dlib's face detector (HOG-based) and then create  # the facial landmark predictor  print("[INFO] loading facial landmark predictor...")  detector = dlib.get\_frontal\_face\_detector()  predictor = dlib.shape\_predictor(args["shape\_predictor"])  # grab the indexes of the facial landmarks for the left and  # right eye, respectively  (lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["left\_eye"]  (rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["right\_eye"]  # start the video stream thread  print("[INFO] starting video stream thread...")  vs = VideoStream(src=args["webcam"]).start()  time.sleep(1.0)  # Main Loop  while True:      brkBtn\_current = wp.digitalRead(break\_button)      btBtn\_current = wp.digitalRead(boot\_button)      brkLED\_state = wp.digitalRead(break\_led)      btLED\_state = wp.digitalRead(boot\_led)      #print(      #    "brkBtn\_current={}, btBtn\_current={}, brkLED\_state={}, btLED\_state={}".format(      #        brkBtn\_current,      #        btBtn\_current,      #        brkLED\_state,      #        btLED\_state))      #time.sleep(0.1)      # break led control with break button      if brkBtn\_current:          if brkBtn\_previous == wp.LOW:              brkLED\_state = wp.digitalRead(break\_led)              wp.digitalWrite(break\_led, not brkLED\_state)              brkBtn\_previous = wp.HIGH      else:          brkBtn\_previous = wp.LOW      # boot led control with boot button      if btBtn\_current:          if btBtn\_previous == wp.LOW and brkLED\_state == wp.HIGH:              btLED\_state = wp.digitalRead(boot\_led)              wp.digitalWrite(boot\_led, not btLED\_state)              btBtn\_previous = wp.HIGH      else:          btBtn\_previous = wp.LOW      # if both LEDs are HIGH, then boot the car      is\_boot = (wp.digitalRead(boot\_led) ==                 wp.HIGH and wp.digitalRead(break\_led) == wp.HIGH)      wp.digitalWrite(real\_boot\_led, is\_boot)      if is\_boot:          alc = wp.analogRead(PCF + 0)          print(" alc value : {}".format(alc))          wp.analogWrite(PCF + 0, alc)          if alc > 150:              wp.digitalWrite(real\_boot\_led, wp.LOW)              wp.digitalWrite(boot\_led, wp.LOW)              wp.digitalWrite(break\_led, wp.LOW)          # grab the frame from the threaded video file stream, resize          # it, and convert it to grayscale          # channels)      frame = vs.read()      frame = imutils.resize(frame, width=450)      gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)      # detect faces in the grayscale frame      rects = detector(gray, 0)      # loop over the face detections      for rect in rects:          # determine the facial landmarks for the face region, then          # convert the facial landmark (x, y)-coordinates to a NumPy          # array          shape = predictor(gray, rect)          shape = face\_utils.shape\_to\_np(shape)          # extract the left and right eye coordinates, then use the          # coordinates to compute the eye aspect ratio for both eyes          leftEye = shape[lStart:lEnd]          rightEye = shape[rStart:rEnd]          leftEAR = eye\_aspect\_ratio(leftEye)          rightEAR = eye\_aspect\_ratio(rightEye)          # average the eye aspect ratio together for both eyes          ear = (leftEAR + rightEAR) / 2.0          # compute the convex hull for the left and right eye, then          # visualize each of the eyes          leftEyeHull = cv2.convexHull(leftEye)          rightEyeHull = cv2.convexHull(rightEye)          cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)          cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)          # check to see if the eye aspect ratio is below the blink          # threshold, and if so, increment the blink frame counter          if ear < EYE\_AR\_THRESH:              COUNTER += 1              # if the eyes were closed for a sufficient number of              # then sound the alarm              if COUNTER >= EYE\_AR\_CONSEC\_FRAMES:                  # if the alarm is not on, turn it on                  if not ALARM\_ON:                      ALARM\_ON = True                      # check to see if an alarm file was supplied,                      # and if so, start a thread to have the alarm                      # sound played in the background                      if args["alarm"] != "":                          t = Thread(target=sound\_alarm,                                     args=(args["alarm"],))                          t.deamon = True                          t.start()                  # draw an alarm on the frame                  cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),                              cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)          # otherwise, the eye aspect ratio is not below the blink          # threshold, so reset the counter and alarm          else:              COUNTER = 0              ALARM\_ON = False              GPIO.output(BUZZER\_PIN, GPIO.LOW)          # draw the computed eye aspect ratio on the frame to help          # with debugging and setting the correct eye aspect ratio          # thresholds and frame counters          cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),                      cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)      # show the frame      cv2.imshow("Frame", frame)      key = cv2.waitKey(1) & 0xFF      # if the `q` key was pressed, break from the loop      if key == ord("q"):          break  # do a bit of cleanup  cv2.destroyAllWindows()  vs.stop() |

6-4. 참고문헌

[1] 도로교통공단 <교통AI빅데이터융합센터>, 음주운전 교통사고 빅데이터 분석결과 발표, 2022.12.22, [문서뷰어 (koroad.or.kr)](https://www.koroad.or.kr/synap/skin/doc.html?fn=BD202212291239530820.pdf&rs=/upfilePath/koroadNew/synap/result/)

[2] 김충현, “추석명절 졸음운전 주의! 음주운전사고 치사율의 1.7배“, 2013.09.17,  [추석명절 졸음운전 주의! 음주운전사고 치사율의 1.7배 홈 > 소식·정보 > 알림마당 > 보도자료 (koroad.or.kr)](https://www.koroad.or.kr/main/board/6/8362/board_view.do)

[3] Face Detection using Haar Cascades [OpenCV: Face Detection using Haar Cascades](https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html)

[4] dib.shape\_predictor [Classes — dlib documentation](http://dlib.net/python/index.html?highlight=shape_predictor#dlib_pybind11.shape_predictor)

[5] Tereza Soukupova and Jan ´ Cech, Real-Time Eye Blink Detection using Facial Landmarks, 2016, [05.pdf (uni-lj.si)](https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf)

[6] Raspberry Pi Documentation, Raspberry Pi OS, [Raspberry Pi Documentation - Raspberry Pi OS](https://www.raspberrypi.com/documentation/computers/os.html#time-lapse-captures)

[7] Python 3.11.2 documentation, tutorial [파이썬 튜토리얼 — 파이썬 3.11.2 문서 (python.org)](https://docs.python.org/3/tutorial/index.html)