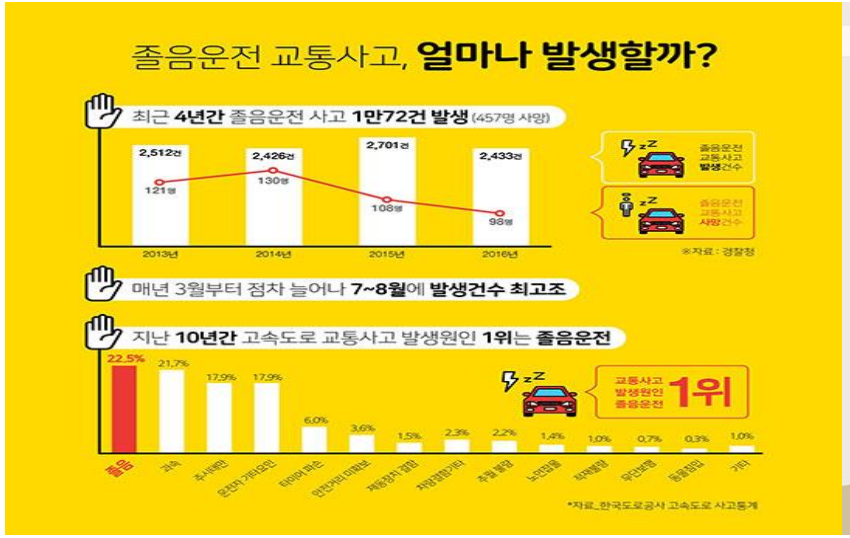
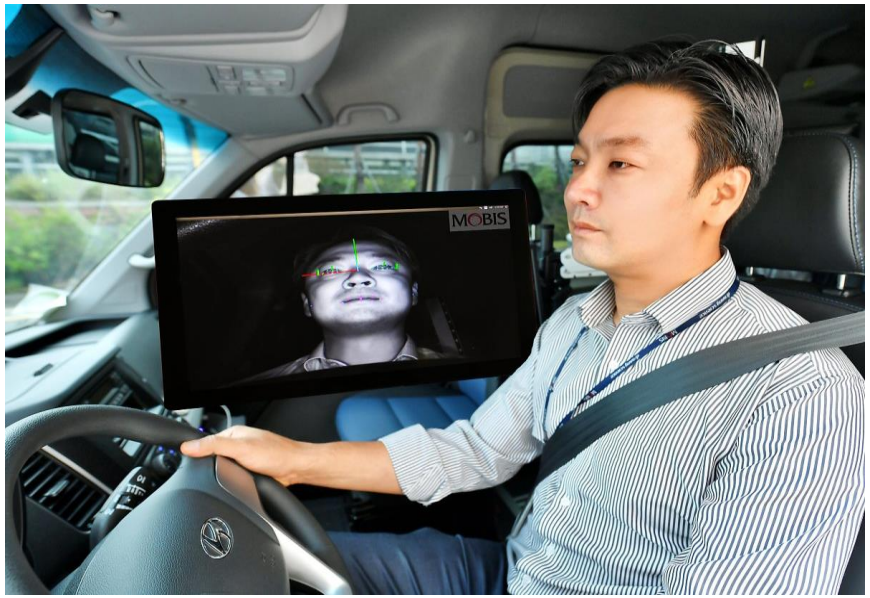
SW개발/HW제작 설계서

프로젝트 명 : 라즈베리파이를 활용한 졸음운전 경보기

[20_HF039] 한상현
[20_HF039] 김영한
[20_HF039] 유민국
[20_HF039] 한승빈



- 대부분의 운전자들이 졸음운전에 대한 위험성은 인지하지만, 이에 대해 적절한 대처를 하는 운전자의 비율은 그리 높지 않음.
- 한국도로공사 고속도로 사고 통계에 따르면 지난 10년간 고속도로 교통사고 빈도 1위는 졸음운전 그러나 그에 비해 해결방안이 다소 적음.
- 졸음쉼터와 같은 간접적인 방법보다는 운전자를 직접적으로 깨워줄 방법이 필요.



운전자의 부주의로 발생하는 교통사고를
방지할 수 있는 신기술이 있습니다.
바로 현대모비스가 개발한 첨단 운전자 부주의 경보시스템
DSW(Driver State Warning system)입니다.

- 현대 모비스 기업의 운전자의 얼굴을 알아보고 시선 추적까지 가능한 '운전자 부주의 경보 시스템' 인 DSW를 개발하였고, 2021년부터 국내 중요 중대형 사용차에 최초 적용될 예정.
- 이즈테크놀로지 사의 운전자 상태 모니터링(DGM) 기기는 운전자의 전방에 위치하여 카메라와 빅데이터를 통한 학습 결함을 기반하여 운전자에게 경보를 보내는 식의 제품을 개발.

요구사항ID	요구사항명	기능ID	기능명	세부사항	예외사항
A01	졸음운전 식별	A01_B01	졸음 식별	운전자의 눈커풀의 임계값 측정을 통해 졸음 식별.	
				차량의 차선 이탈 감지를 통한 부가 식별.	
				운전자의 심장박동을 측정함으로써 부가 식별.	
				차량 내부의 이산화탄소 측정을 통한 부가 식별.	
		A01_B02	졸음 자각	창문 개방을 통해 운전자의 졸음 퇴치.	
				경보음, 비상등을 통해 운전자 및 외부에 졸음 자각.	

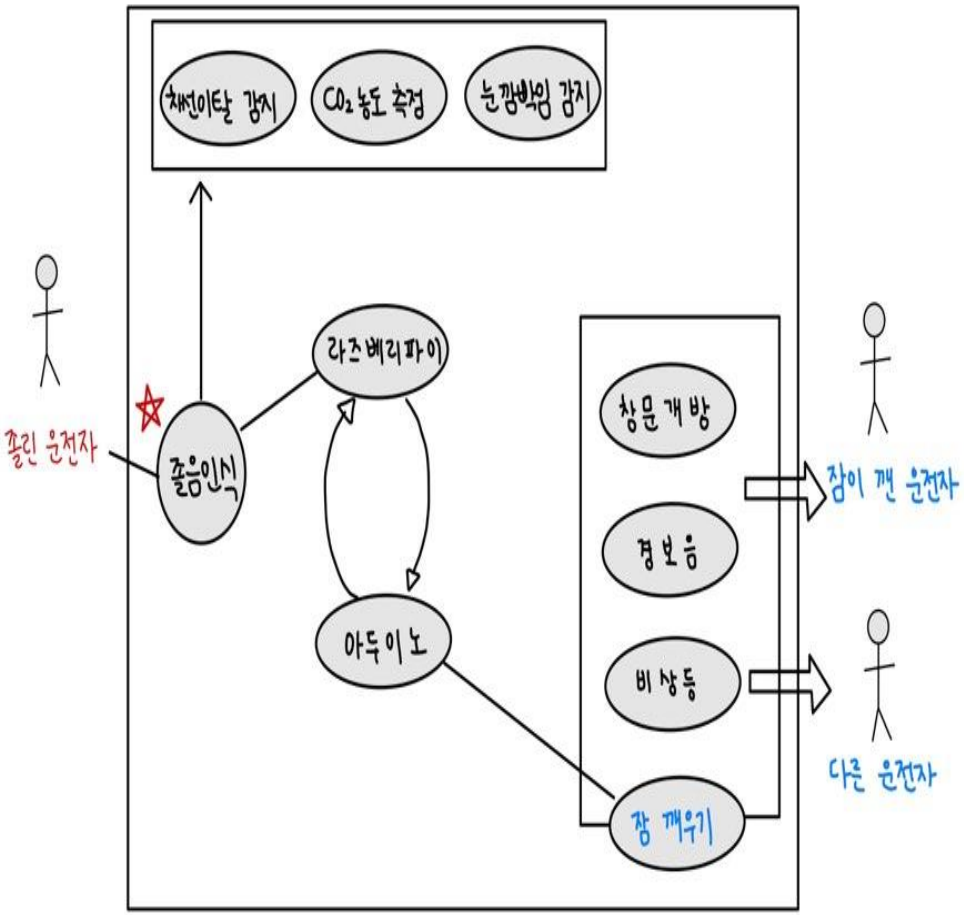
ㅇ S/W 주요 기능

기능	설명
눈 깜빡임 감지	openCV를 이용하여 운전자의 눈을 실시간으로 감지하고, 눈의 가로와 세로의 비율을 통해 눈이 감겨있는지 확인한다. 그 후 사전 정의된 조건에 따라 졸음이라고 판단.
차선 인식	영상처리 기법으로 차량이 비정상적으로 차선을 이탈했을 때를 감지. 예를 들어 깜빡이를 넣지 않고 차선 이탈을 할 경우 졸음운전으로 인한 차선 변경이라고 판단.
졸음 감지의 정확성을 높이기 위한 추가적 요소	각종 센서를 이용하여 눈 깜빡임이나 차선인식만으로는 충분하지 않을 수 있는 오작동 비율을 낮춘다. 눈 깜빡임, 차선 이탈, 각종 센서마다 임계값을 넘으면 경보음이 울리기 위한 수치를 설정해놓고, 이 수치의 총 합이 일정 값 이상을 넘을 시 졸음운전이라고 판단.

ㅇ H/W 주요 기능

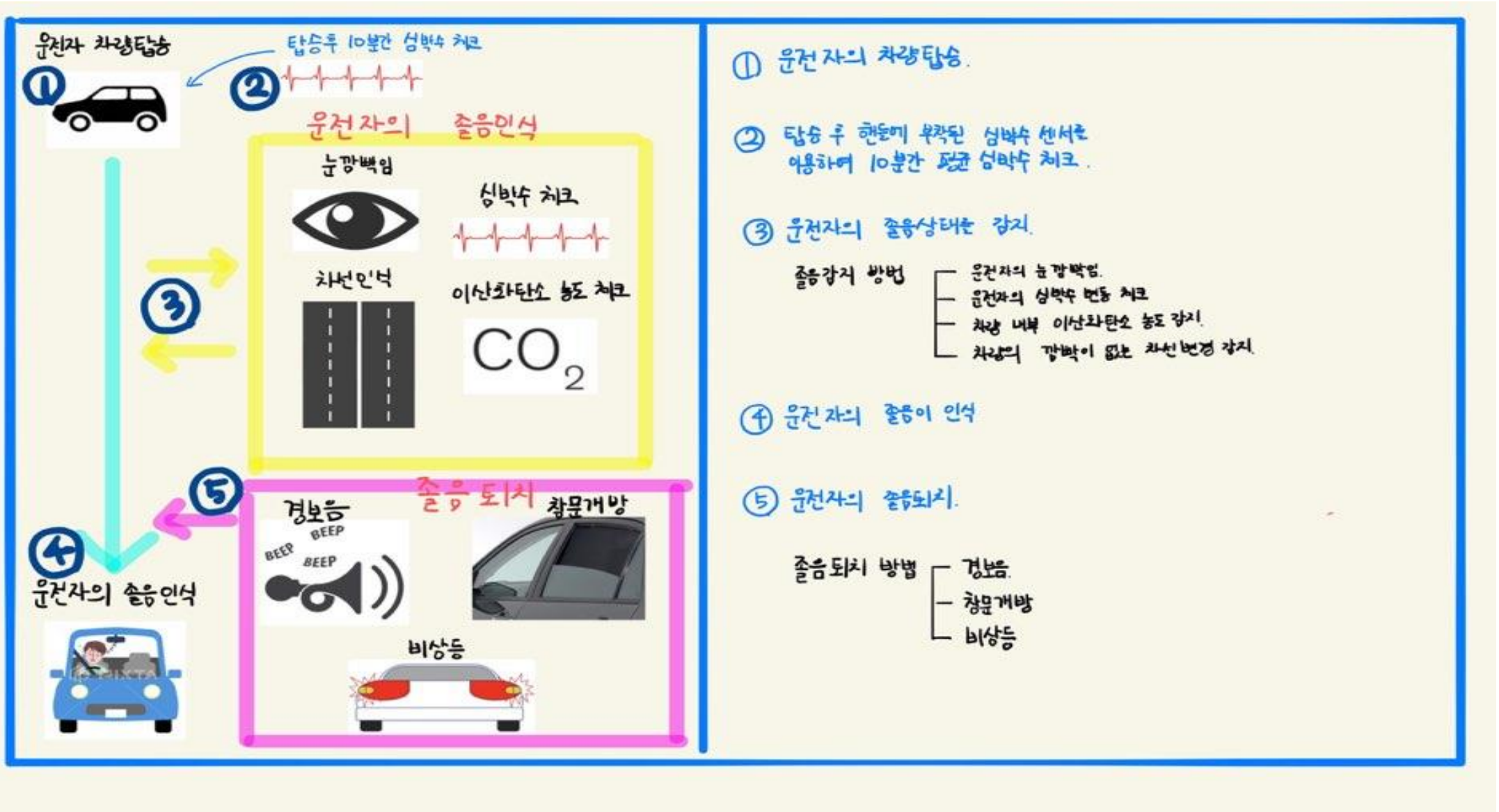
기능/부품	설명
졸음감지 / 카메라 2대	첫 번째 카메라는 운전자의 눈 깜빡임을 인식 두 번째 카메라는 차량 전방의 차선을 인식
졸음감지 / CO2센서 (창문 개방)	차량 내부의 CO2농도를 주기적으로 측정하고 그 측정값이 임계값을 넘으면 창문을 개방하여 졸음을 방지.
졸음감지 / 심장박동센서	차량 운전자가 탑승한 후 10분간 운전자의 심장박동수를 주기적으로 측정하고 그 측정값이 일정값이 넘으면 경보음이 울리기 위한 수치의 하한값을 증가시킴
졸음 퇴치 / 피에조 부저	각종 센서와 졸음 감지에 대한 조건을 충족하여 졸음이라고 감지되면 경보음을 발생시킴

| 유즈케이스 정의서

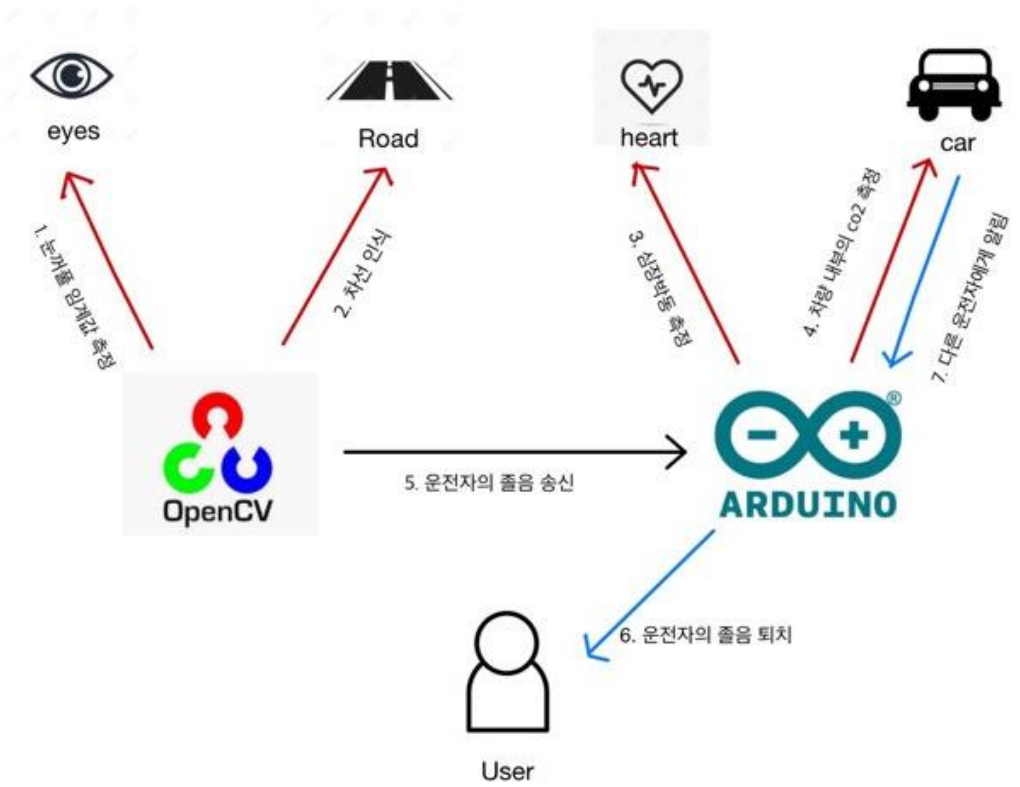


항목	설명		
개요	운전자가 운전중에 졸았을 경우 졸음 예방 및 인식.		
관련 액터	주 액터	라즈베리파이 opencv	
	보조 액터	Co2센서, 심장박동 센서, 차선인식 감지	
우선 순위	상	중요도	5
		난이도	운전자 눈동자의 임계값 및 차선의 이탈 감지 가능한 개발 능력.
선행 조건	라즈베리파이가 눈동자 크기를 잘 인식해줘야 함.		
후행 조건	운전자의 졸음을 깨워줘야 함.		
시나리오	기본 시나리오	1. 운전자는 운전을 시작한다. 2. 라즈베리파이 opencv 및 dlib을 통해 운전자 눈꺼풀의 임계값을 측정한다. 3. 아두이노는 차량 내부의 co2 농도, 운전자의 심장 박동을 측정하여 졸음 인식의 정확도 높인다. 4. 창문을 개방하여 운전자의 졸음을 깨워준다. 5. 계속하여 주요 및 부가기능을 통해 운전자의 졸음을 깨워준다.	
	대안 시나리오	**A1 : 눈동자의 크기를 잘 인식하지 못할 경우** 1. 차량 내부의 co2농도 및 외부의 차량 이탈 감지를 통해 졸음을 식별. 2. 심장박동센서를 통해 운전자의 졸음 식별. **A2 : 운전자가 졸음을 깨지 못할 경우.** 1. 비상등을 통해 외부의 다른 운전자에게 경고.	

| 서비스 구성도


운전자 차량탑승
탑승후 10분간 심박수 체크
①
②
운전자의 졸음인식
눈깜빡임
심박수 체크
차선인식
이산화탄소 농도 체크
CO2
③
⑤
졸음 퇴치
경보음
BEEP
BEEP
BEEP
창문개방
비상등
④
운전자의 졸음인식
① 운전자의 차량탑승.
② 탑승 후 핸들에 부착된 심박수 센서를 이용하여 10분간 평균 심박수 체크.
③ 운전자의 졸음상태를 감지.
졸음감지 방법
운전자의 눈 깜박임.
운전자의 심박수 변동 체크
차량 내부 이산화탄소 농도 감지.
차량의 깜박이 없는 차선변경 감지.
④ 운전자의 졸음이 인식
⑤ 운전자의 졸음퇴치.
졸음퇴치 방법
경보음.
창문개방
비상등

| 서비스 흐름도



- 운전자의 졸음 인식

1) Opencv에서 운전자의 눈꺼풀의 임계값을 인식한다.
2) 졸음의 인식 정확도를 더욱 높이기 위해 차량의 차선이탈을 감지한다.
3) 졸음의 인식 정확도를 더욱 높이기 위해 운전자의 심장박동이 낮아지는지 측정한다.
4) 차량 내부의 co2센서가 낮아지면 운전자의 졸음이 야기될 수 있으므로 co2 농도를 측정한다.

- 운전자의 졸음 퇴치

5) 라즈베리파이에서 측정한 값들을 아두이노에 보내 종합하여 졸음인지 식별.
6) 창문 개방 및 경보음을 통해 운전자의 졸음을 퇴치.
7) 외부의 다른 운전자들에게 위험을 인식.

| 하드웨어/센서 구성도

CO_2 센서

(1) 읽어 들이는 방법


아두이노 CO2 연결도안


아두이노 , CO_2 연결 실제 사진


CO_2센서 라이브러리 코드

(2) 파일에 쌓아두기
-> 이산화탄소 데이터를 이용하지 않고, 임계값이 넘어가는 것만 체크하면 됨으로 저장해둘 필요성이 없다.
디스플레이 활용

(3) 주기적으로 읽기
-> 1분에 한번씩 이산화탄소 센서에 값을 가져와서 2000ppm이상일때
자동차 외기버튼이 눌리거나, 환기 실시 이때 환기는 10~15분정도 실시해야한다.

(4) 정보
농도
~450 건강한 환기 관리 된 레벨
~700 장시간 있어도 건강에 문제가 없는 실내 레벨
~1000 건강 피해는 없지만 불쾌감을 느끼는 사람이 있는 레벨
~2000 졸림을 느끼는 등 컨디션 변화가 나오는 레벨
~3000 어깨 결림이나 두통을 느끼는 사람이 있는 등 건강 피해가 생기기 시작하는 레벨
3000~ 두통, 현기증 등의 증상이 나오고, 장시간으로는 건강을 해치는 레벨

차량 내부 CO_2 농도 2500PPM => 위험임계값
차량 내부 CO_2 농도 700PPM 이하 => 안전임계값
LED로도 표시가능

// 이산화탄소 농도가 높을때 충분한 환기 시간
-> 주행중 위험임계값 넘으면 안전임계값에 도달할때까지 창문개방

참 고

//SBS 졸음운전 연속기획
창문이 모두 닫힌채 운전하는 차량 내부 -> 승객도 운전자도 졸게된다.
-> 충분히 수면을 취해도 이산화탄소 농도가 높아지면 졸음이 온다

4명이 탄 승용차 기준
20여분에 5000 ppm 돌파

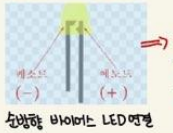
외기버튼 누르기 (창문 개방 대신)

경보음 and 비상등

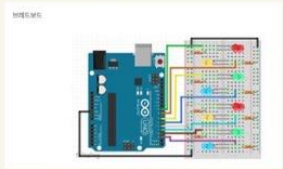
운전자가 자신의 졸음상태를 의식하지 못하는 경우에 사용
-> 눈 영상 처리를 통해 졸음을 인식하지 못하는 경우, 비상등과 경보음 발생
-> CO2 농도가 높다고 조는것은 아님으로 경보등이나 비상등 장치는 연결하지 않음

운전 차량에서 비상등이 켜지고, 운전자가 인식하고 비상등을 끄면 경보음도 같이 꺼지는 방식으로 구현.
경보음과 비상등은 Set.
경보음 -> 피에조 부저를 이용하여 표현
비상등 -> LED 깜빡거리는 방법으로 표현

비상등 구현 방법 (LED 사용)

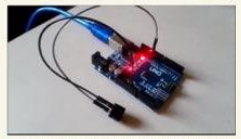


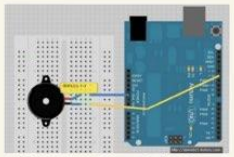
=> LED 전류제한용 저항 사용.
순방향 바이어스 LED연결


아두이노, LED 연결 도면


LED 구현 코드

경보음 구현 방법 (피에조 부저 사용)


아두이노, 피에조 부저 연결 사진


아두이노, 피에조 부저 연결 도면

```
##소스 1##
BEEP 음을 1초 마다 내는 소스, tone() 의 파라미터를 조절하면 다양한 음을 만들 수 있다.
int speakerpin = 12; //스피커가 연결된 디지털핀 설정

void setup() {
}

void loop() {
  tone(speakerpin,500,1000); //500: 음의 높낮이(주파수), 1000: 음의 지속시간(1초)
  delay(2000);
}
```
피에조 부저 구현 코드

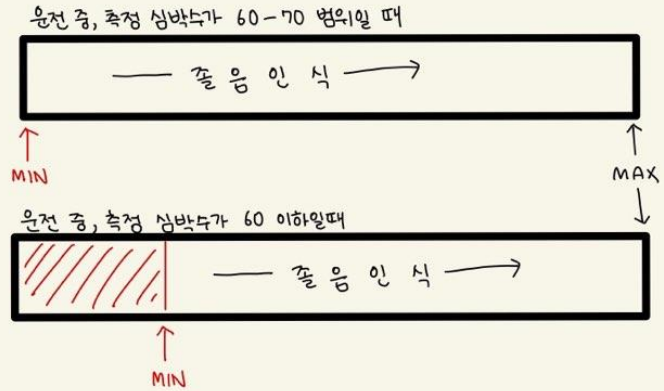
심 장 박 동 센 서

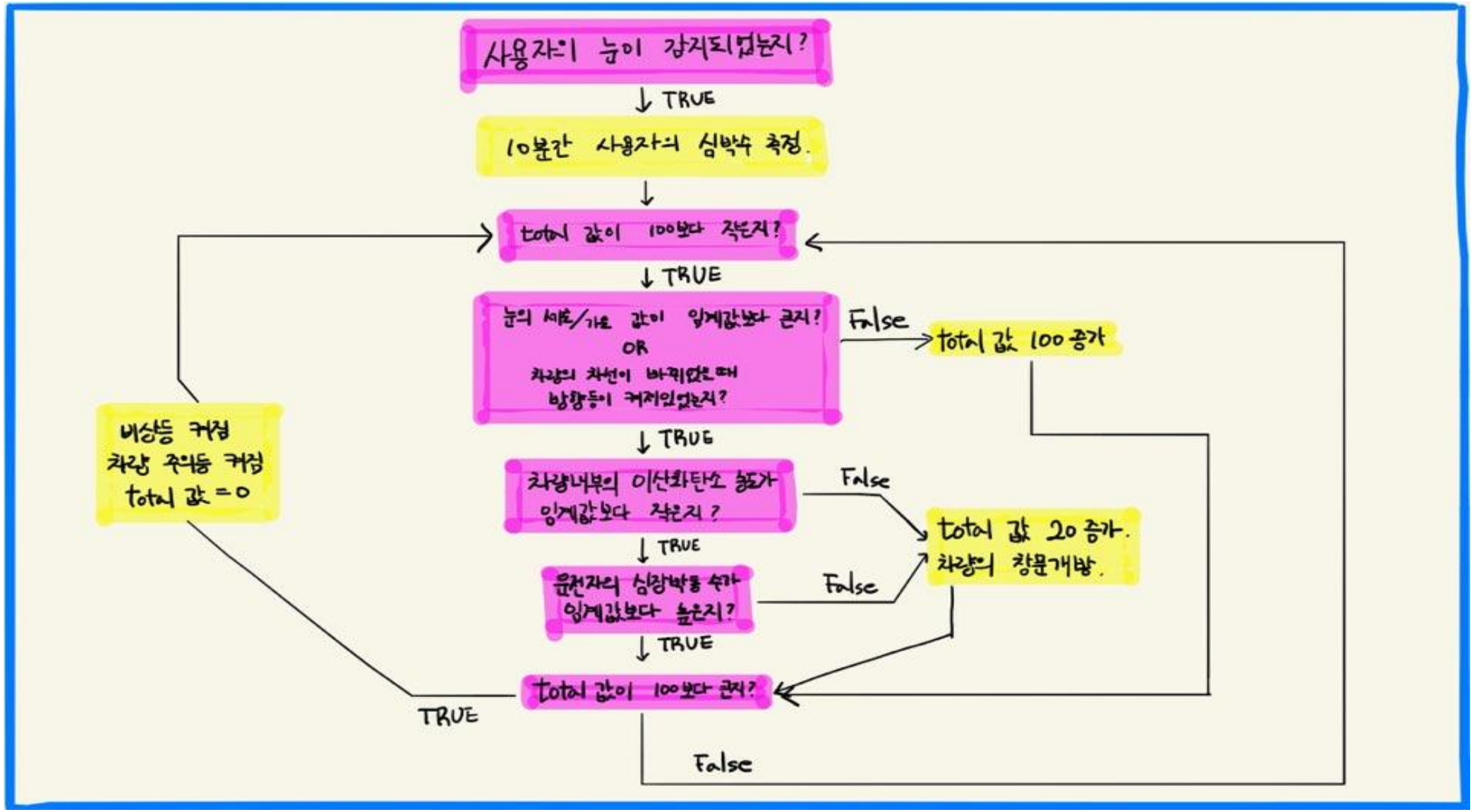
탑승 후 10분간 평균 심박수를 잰다.

그 평균 심박수를 기준으로 낮아지면 total 변수값의 min값을 올린다.

ex) total 변수의 범위가 0~100이라면 20~100 처럼 하한선을 올림
즉, 졸음운전 판단에 더 민감하게 반응

Ex) 10분간 측정한 평균 심박수 : 65




사용자의 눈이 감지되었는지?
TRUE
10분간 사용자의 심박수 측정.
total 값이 100보다 작은지?
TRUE
눈의 세로/가로 값이 임계값보다 큰지?
OR
차량의 차선이 바뀌었을때
방향등이 켜져있었는지?
False
total 값 100 증가
TRUE
차량내부의 이산화탄소 농도가
임계값보다 작은지?
False
total 값 20 증가.
차량의 창문개방.
TRUE
운전자의 심장박동 수가
임계값보다 높은지?
False
TRUE
비상등 켜짐
차량 주위등 켜짐
total 값 = 0
total 값이 100보다 큰지?
TRUE
False

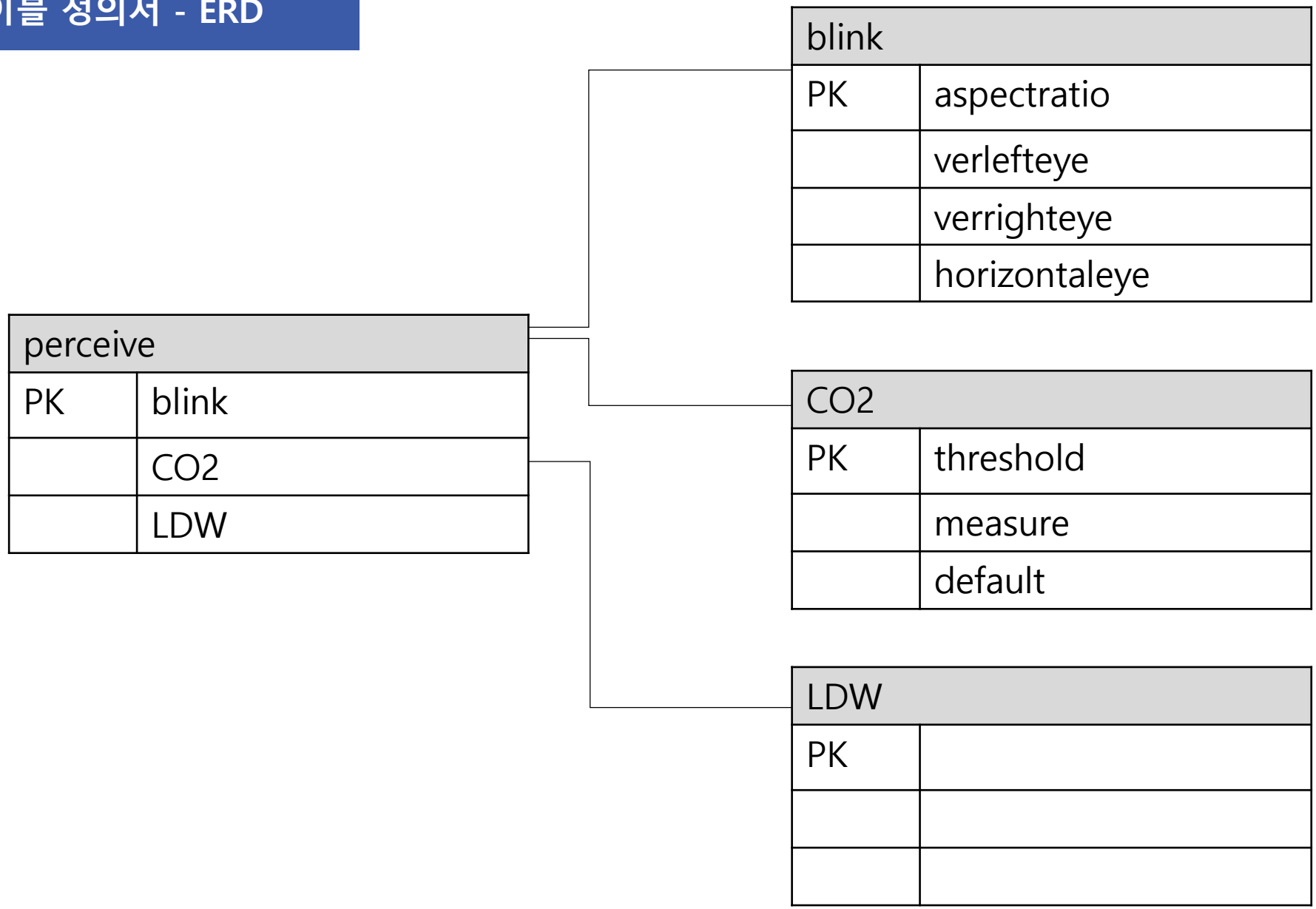
- EAR 알고리즘

Face landmarks를 사용해 Real-Time Eye Blink Detection의 작업을 기반으로 한 알고리즘이다. Face landmarks는 얼굴의 특정 부분마다 점을 찍어 나타내는 것이다. 대표적으로 68 points markup이 많이 사용되고 눈, 코, 입, 눈썹, 얼굴형 총 68개의 점으로 특정하여 얼굴을 인식하게 된다.
눈이 작아지면 세로 비율이 작아지고, EAR 비율 역시 작아진다. 사람이 졸리면 눈을 감게 되므로 EAR값을 이용해 주기적으로 낮아지면 알람을 울리게 하는 시스템을 만들 수 있다. 평소 눈의 임계값을 0.3정도로 해놓고 눈을 감아서 그 이하로 떨어지게 되면 알람이 울리게 된다.

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

| 프로그램 목록

기능 분류	기능번호	기능 명
PER	PER-01-01	눈 깜빡임(졸음) 감지
	PER-01-02	차선 인식
	PER-01-03	심장 박동수 인식
	PER-01-04	Co2 농도 측정
SOL	SOL-01	창문 개방
	SOL-02	경보음
	SOL-02-01	비상등
이하 생략		


blink
PK
aspectratio
verlefteye
verrighteye
horizontaleye
perceive
PK
blink
CO2
LDW
CO2
PK
threshold
measure
default
LDW
PK

<blink 테이블>

항목명	항목ID	필수/선택	값 목록	설명
왼쪽 수직 눈 랜드마크	verlefteye	필수		왼쪽 눈 수직 위치
오른쪽 수직 눈 랜드마크	verrighteye	필수		오른쪽 수직 위치
수평 눈 랜드마크	horizoneye	필수		눈 수평 위치
눈 종횡비	aspectratio	필수		눈 종횡비

<CO2 테이블>

항목명	항목ID	필수/선택	값 목록	설명
기본 설정값	default	필수		평상시 이산화탄소 농도
현재 측정값	measure	필수		현재 측정되고 있는 이산화탄소 농도
임계값	horizoneye	필수		임계값이 이상이면 졸음운전 위험

| 핵심소스코드(1)

```
        # 두 세트의 수직 눈 랜드 마크 (x, y) 좌표 간의 유클리드 거리 계산
        A = dist.euclidean(eye[1], eye[5])
        B = dist.euclidean(eye[2], eye[4])

        # 수평 눈 랜드 마크 (x, y) 좌표 간의 유클리드 거리 계산
        C = dist.euclidean(eye[0], eye[3])

        # 눈 종횡비 계산
        ear = (A + B) / (2.0 * C)

        # 눈 종횡비 반환
        return ear

# 파라메터 구문 분석
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True, help="path to facial landmark predictor"
ap.add_argument("-v", "--video", type=str, default="", help="path to input video file")
args = vars(ap.parse_args())

# 눈의 종횡비가 깜박임을 나타내는 상수와 임계값 보다 낮아야 하는 연속 프레임 수에 대한 상수 정의
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 3

# 프레임 카운터와 총 깜박임 수를 초기화
COUNTER = 0
TOTAL = 0

# dlib의 얼굴 탐지기 (HOG 기반)를 초기화 한 다음 얼굴 랜드 마크 예측 변수 생성
print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
```

```
# 왼쪽 눈과 오른쪽 눈에 대한 얼굴 랜드 마크의 인덱스 설정
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

# Video Stream 초기화
print("[INFO] starting video stream thread...")
#vs = FileVideoStream(args["video"]).start()
fileStream = False
vs = VideoStream(src=0).start()
# vs = VideoStream(usePiCamera=True).start()
# fileStream = False
time.sleep(1.0)

# Video Stream 반복
while True:
        # 파일 비디오 스트림인 경우 처리 할 버퍼에 프레임이 더 남아 있는지 확인
        if fileStream and not vs.more():
                break

        # 스레드 비디오 파일 스트림에서 프레임을 가져 와서 크기를 조정한 다음 Grayscale 채널로 변환
        frame = vs.read()
        frame = imutils.resize(frame, width=400)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Grayscale 프레임에서 얼굴 감지
        rects = detector(gray, 0)
```

| 핵심소스코드(2)

```
# 얼굴 감지 반복
for rect in rects:
	# 얼굴 영역의 얼굴 랜드 마크를 결정한 다음 얼굴 랜드 마크 (x, y) 좌표를 NumPy 배열로 변환
	shape = predictor(gray, rect)
	shape = face_utils.shape_to_np(shape)

	# 왼쪽 및 오른쪽 눈 좌표를 추출한 다음 좌표를 사용하여 두 눈의 눈 종횡비 계산
	leftEye = shape[lStart:lEnd]
	rightEye = shape[rStart:rEnd]
	leftEAR = eye_aspect_ratio(leftEye)
	rightEAR = eye_aspect_ratio(rightEye)

	# 두 눈의 평균 눈 종횡비
	ear = (leftEAR + rightEAR) / 2.0

	# 왼쪽 눈과 오른쪽 눈의 눈꺼플을 계산 한 다음 각 눈을 시각화
	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)

	# 눈의 종횡비가 깜박임 임계값 보다 낮은지 확인하고, 그렇다면 눈 깜박임 프레임 카운터를 늘림
	if ear < EYE_AR_THRESH:
		COUNTER += 1

	# 그렇지 않으면, 눈의 종횡비가 깜박임 임계값 보다 낮지 않음
	else:
		# 눈의 깜박임 수가 연속 깜박임 프레임 임계값 보다 큰 경우 총 깜박임 횟수 증가
		if COUNTER >= EYE_AR_CONSEC_FRAMES:
			TOTAL += 1

		# 눈 깜박임 프레임 카운터 재설정
		COUNTER = 0

	# 프레임의 계산 된 눈 종횡비와 함께 프레임의 총 깜박임 수 표시
	cv2.putText(frame, "Blinks: {}".format(TOTAL), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2
	cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2
```

< 핵심 소스코드(1) >

- 사용자의 얼굴의 두 눈을 인식하여 각각의 눈의 랜드마크에 대한 수직, 수평 유클리드 거리계산

- 임계값과 비교할 수 있도록 수직, 수평 유클리드 거리를 이용하여 눈의 종횡비를 구함.

- 눈의 종횡비가 깜빡임을 나타내는 상수와 임계값보다 낮아야하는 연속 프레임 수에 대한 상수를 정해줌

< 핵심 소스코드(2) >

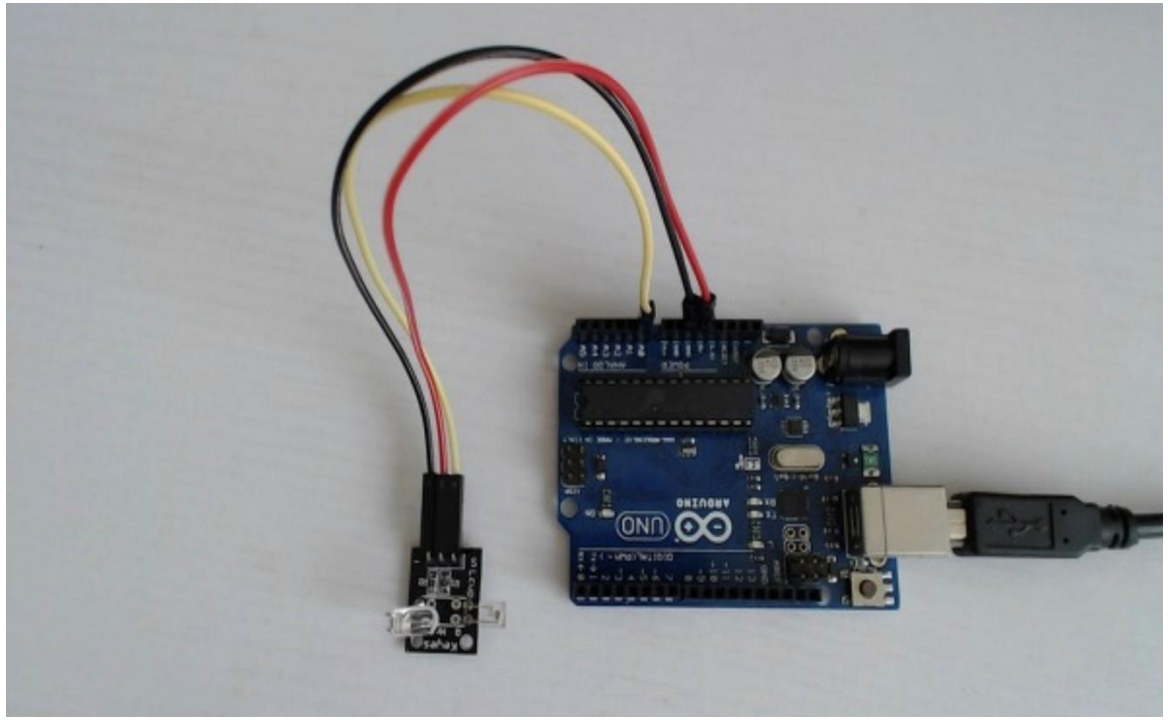
- 운전자의 얼굴 감지를 반복적으로 수행하며 매 순간마다 두 눈의 좌표를 추추라여 유클리드 거리 계산하고 종횡비를 계산
- 눈의 종횡비의 평균이 깜빡임 임계값보다 낮은지 확인하고, 낮으면 눈 깜빡임 프레임 카운터를 증가, 아닐경우 눈의 깜빡임수가 연속 깜빡임 프레임 임계값보다 큰 경우 총 깜빡임 횟수만 증가

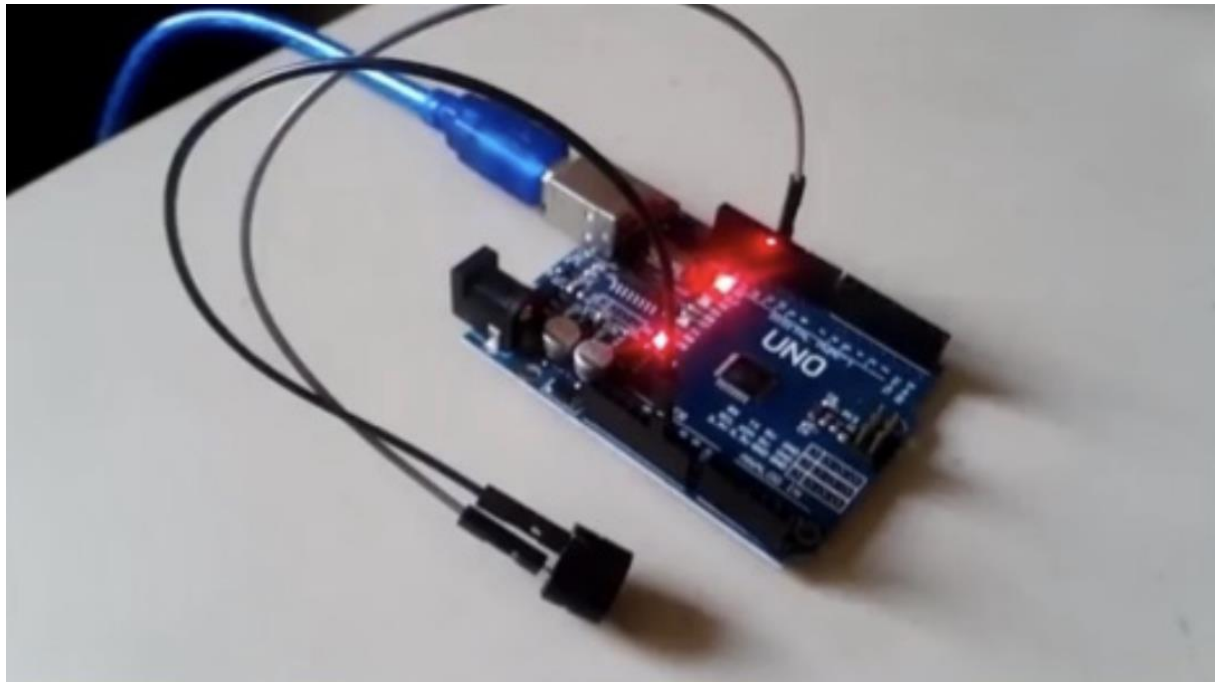
| 참조- 개발 환경 및 상세내용

구분		상세내용
S/W 개발환경	OS	Ubuntu
	개발환경(IDE)	PyCharm, terminal
	개발도구	PyCharm
	개발언어	Python
	기타사항	노트북을 이용
H/W 구성장비	디바이스	Raspberry, Arduino
	센서	Co2 센서, 피에조 부저, 심장박동 센서
	통신	와이파이
	기타사항	라즈베리파이와 아두이노를 연동해주는 모듈 존재
프로젝트 관리환경	형상관리	Git
	의사소통관리	애자일 방법론의 데일리 스크럼
	기타사항	멘토와 월 1회 이상 오프라인 미팅, 카카오톡을 통한 상시 멘토링


Hough

| 참조-H/W 기능 실사 사진




UNO