졸음 운전 방지시스템

15181447 이현진

16100166 윤인재

Index

01 프로젝트 개요

문제 인식

고속도로에서 졸음으로 인한 교통사고는 실제 사고 사망자 중 31%로 매우 위험한 요소이다.

졸음이 오는 이유는 다양하나 가장 주된 원인은 잠 부족, 고속도로 최면효과 이다.

잠을 깨는 방법은 껌 씹기, 창문열기, 수다떨기 등이 있으나 실제로는 큰 효과가 없다.

최근 3년간 교통사고 사망자

해결 방안

실시간으로 운전자의 표정변화, 눈 깜빡거림, 하품을 감지하는 알람기를 제작한다.

카메라

+

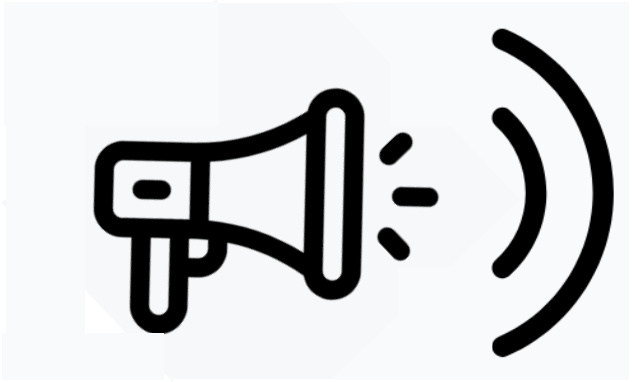
스피커

01 프로젝트 개요

작동시나리오

sleepy

detect



Warning sound

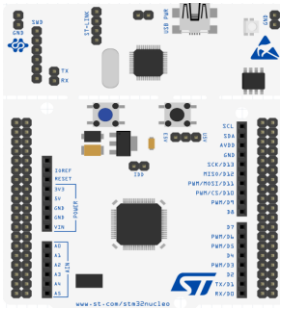
Awake!

01 프로젝트 개요

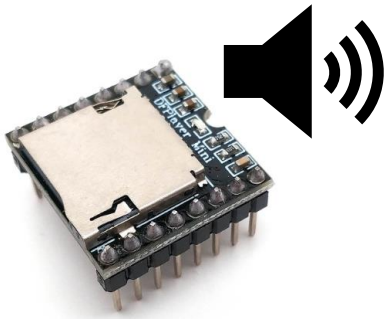
각 요소별 역할

Raw image

is sleepy



trigger



Camera

Raw image를 PC측으로 전달

PC

받은 Raw 이미지를 통해 운전자의 상태를 판단

MCU

운전자의 상태에 따라 스피커에 트리거신호를 전달

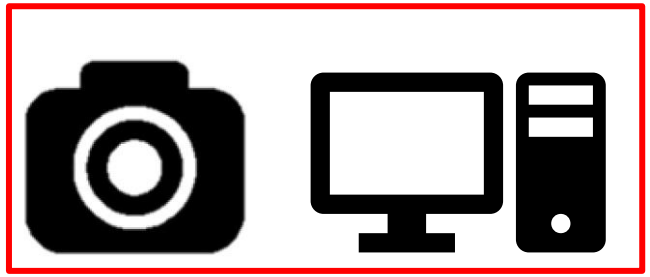
Speaker

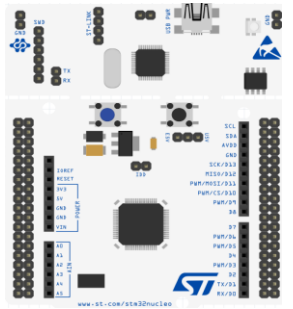
저장된 음성을 출력


상세 시스템

02 상세 시스템

PC 각 모듈의 역할





openCV

카메라로부터 이미지 받아옴

Dlib

ROI검출 및 얼굴의 특징점을 추출

PySerial

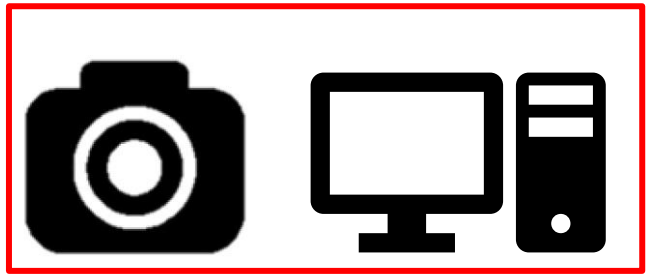
운전자의 상태에 따라 스피커에 트리거신호를 전달

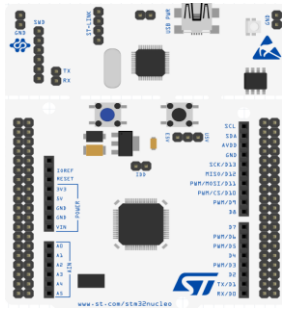
Python

기본 연산 및 시스템 구조 구성.

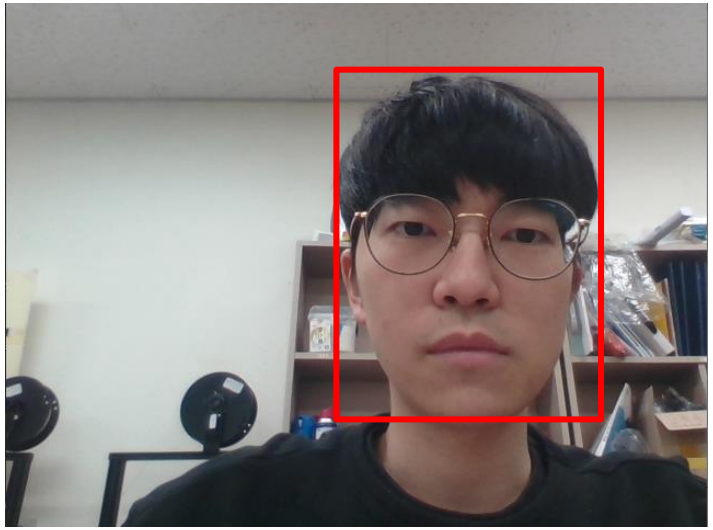
02 상세 시스템

특징점 추출 과정

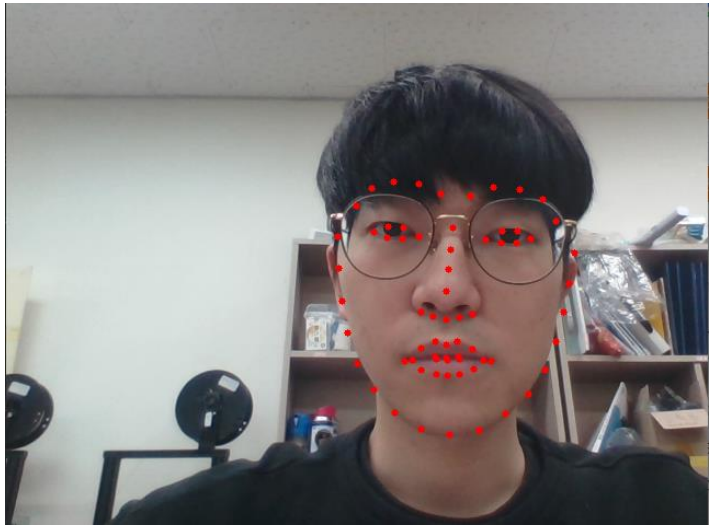




OpenCV의 VideoCapture를
이용해 카메라로 부터
raw image를 받아옴.



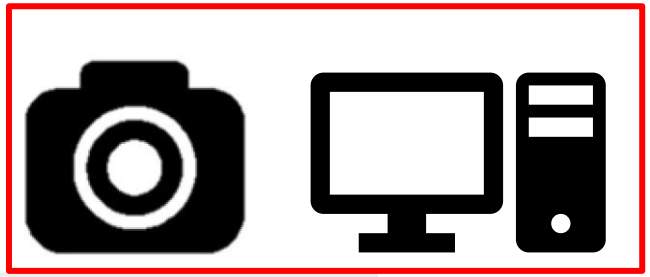
dlib의 get_frontal_face_detector를
통해 얼굴의 ROI를 추출

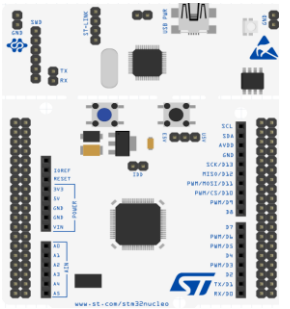


dlib의 shape_predictor와 미리
학습된 dat file을 이용하여
얼굴의 특징점을 추출

02 상세 시스템

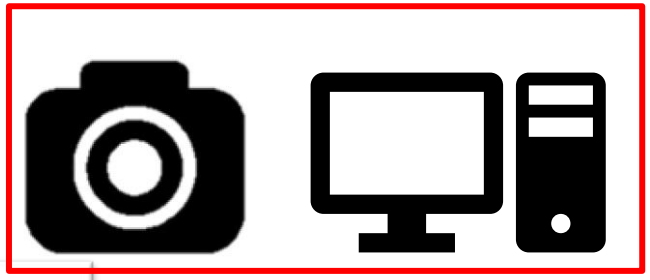
특징점 추출 실행영상

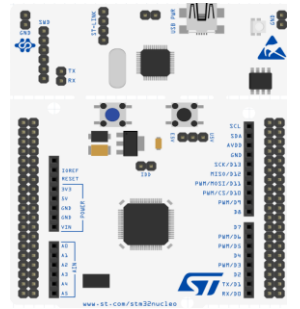


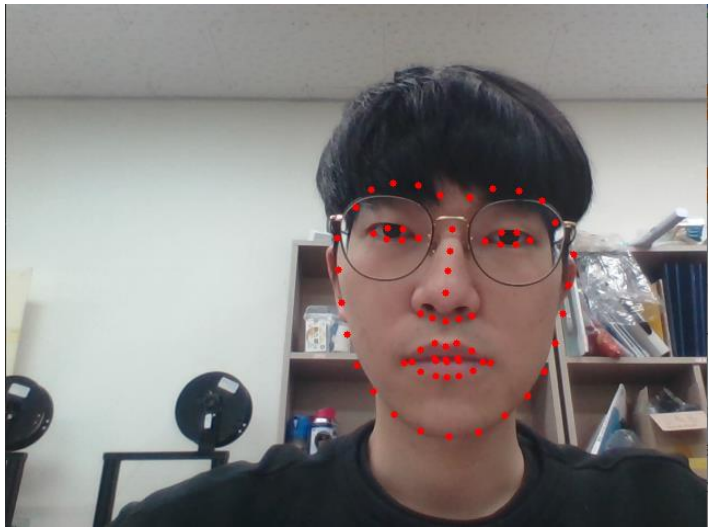


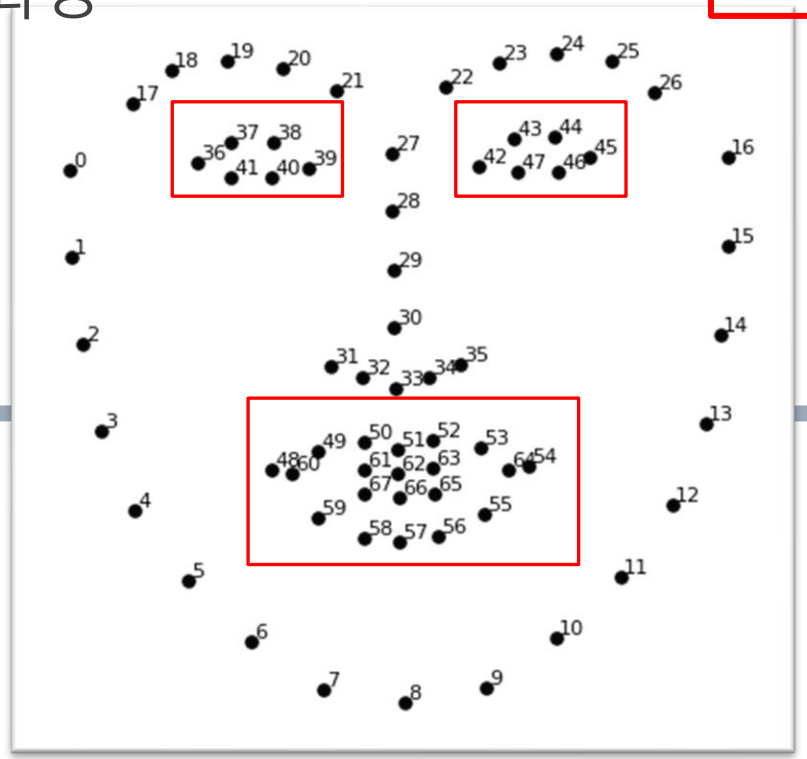
02 상세 시스템

입과 눈을 통한 피곤함 판단과정





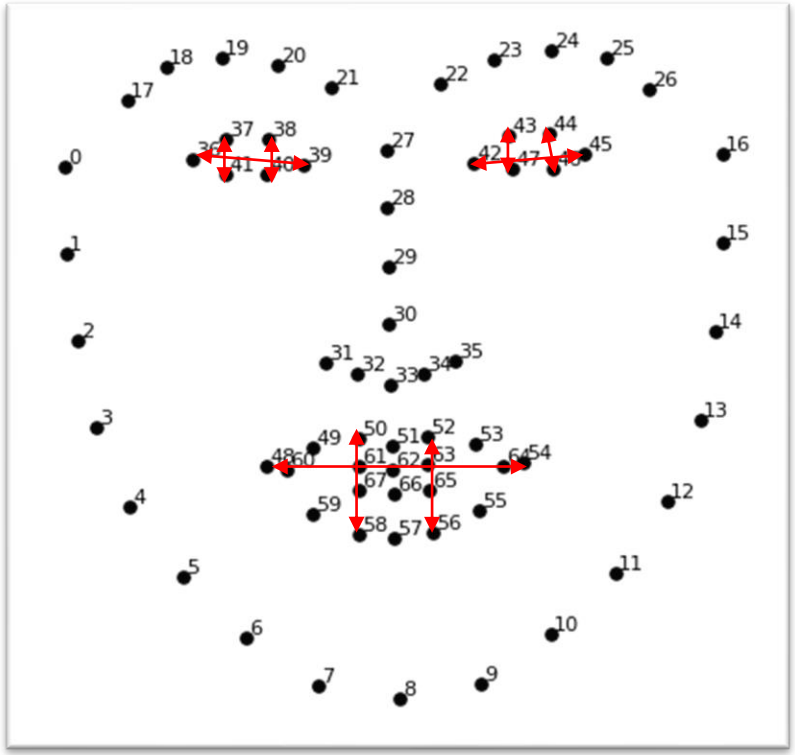


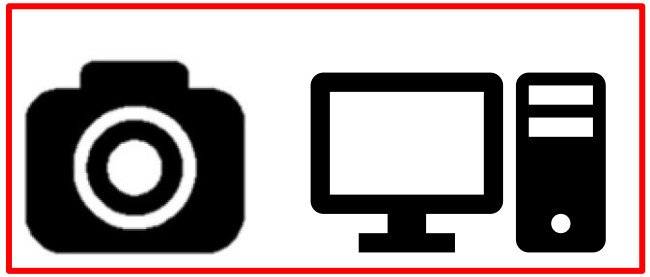


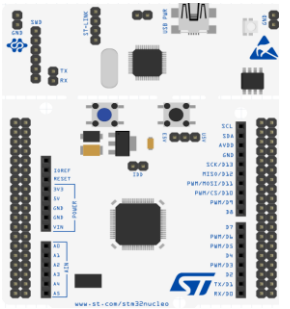
슬라이싱을 통해
눈과 입에 해당하는 특징점만 따로 추출.

02 상세 시스템

입과 눈을 통한 피곤함 판단과정







$$mouth\ open\ ratio = \frac{distance(p(50), p(58)) + distance(p(52), p(56))}{2distance(p(48), p(54))}$$

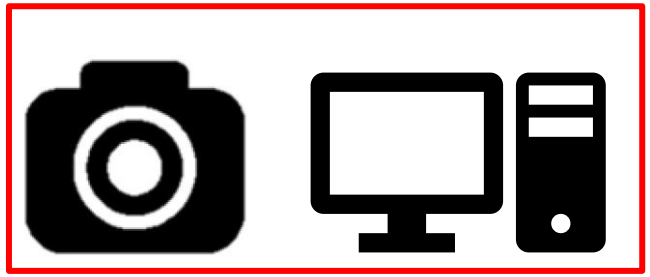
$$left\ eye\ open\ ratio = \frac{distance(p(37), p(41)) + distance(p(38), p(40))}{2distance(p(36), p(39))}$$

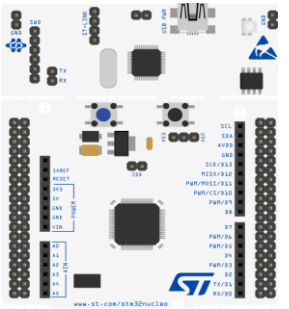
$$right\ eye\ open\ ratio = \frac{distance(p(43), p(47)) + distance(p(44), p(48))}{2distance(p(42), p(45))}$$

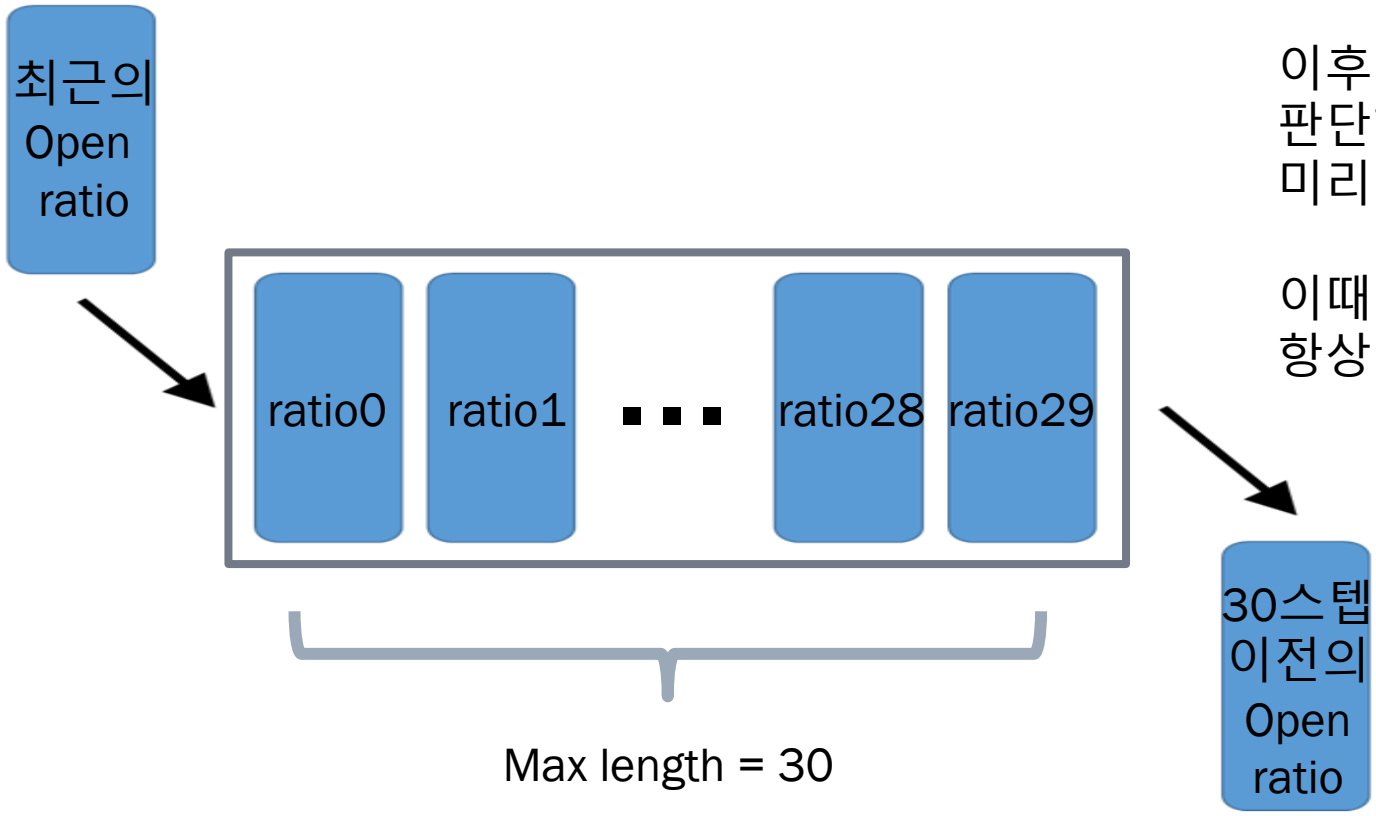
이후 하품과 게슴츠레 한 눈을 판단하기 위한
mouth open ratio와
eye open ratio를 위와같이 정의하고
실시간으로 이를 구함.

02 상세 시스템

입과 눈을 통한 피곤함 판단과정



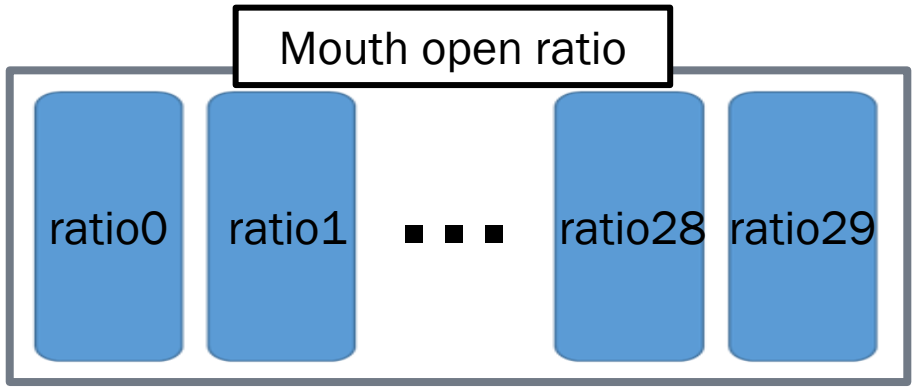


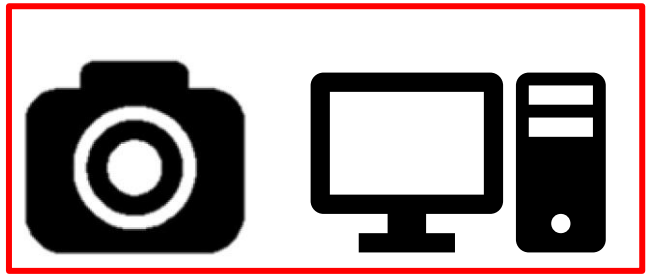


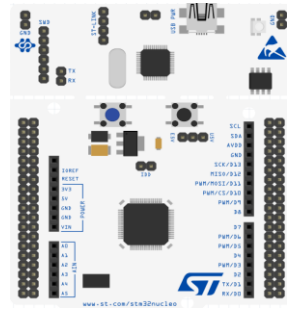
이후 하품과 게슴츠레한 눈을
판단하기 위해 최근의 open ratio들을
미리 선언된 deque(double-ended queue)에 집어넣음

이때 deque에 들어갈 수 있는 자료의 수를 따로 지정하여
항상 최신의 데이터를 유지하게 함.

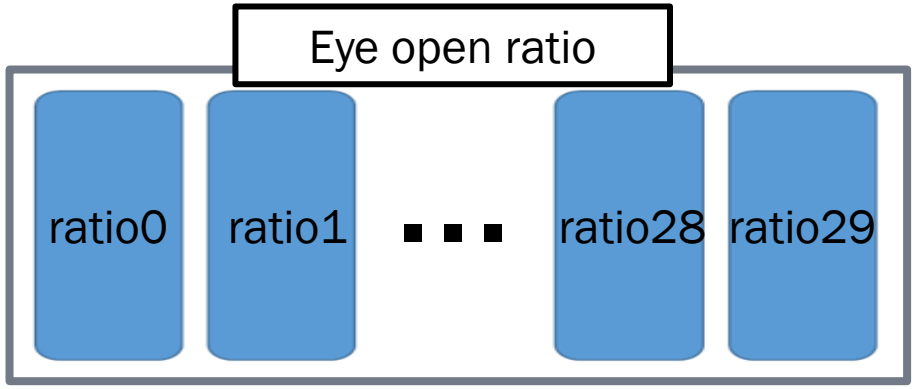
02 상세 시스템







$$yawn\ factor = E(mouth\ open\ ratio)$$
$$if\ yawn\ factor > thereshold:$$
$$num\ yawn++$$
$$if\ num\ yawn == 3:$$
$$sleepy = True$$

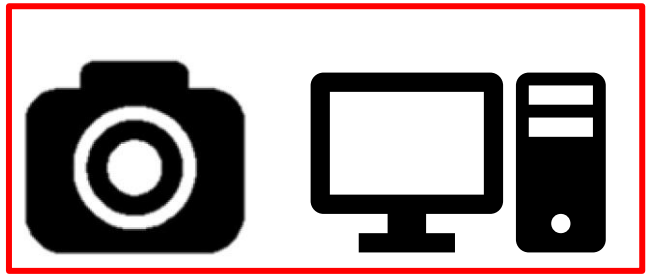


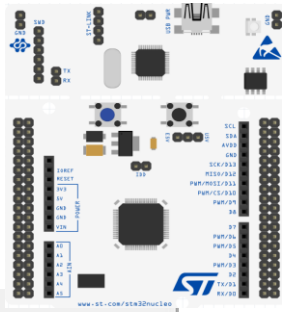
$$bleary\ factor = E(eye\ open\ ratio)$$
$$if\ \ bleary\ factor < threshold:$$
$$num\ bleary++$$
$$if\ num\ bleary == 100:$$
$$sleepy = True$$

두 queue 안에 들어있는 값들의 평균을 각각
하품을 판단하기 위한 지표와 눈을 게슴츠레 뜬 것을 판단하기 위한 지표로 사용하며.
각 지표와 threshold를 비교하여 하품의 횟수를 측정하거나, 눈을 게슴츠레하게 뜬 시간을 측정
하품의 횟수와 눈을 게슴츠레하게 뜬 시간을 측정하여 피곤함을 판단하고 시리얼통신으로 값을 전달함.

02 상세 시스템

하품 및 게슴츠레한 눈 판단 실행영상



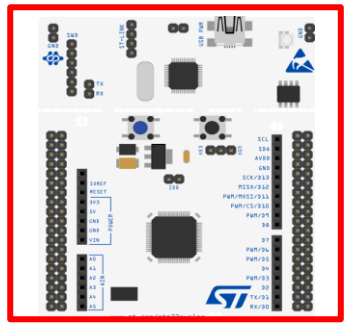


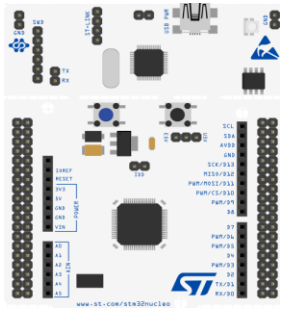
하품 인식

게슴츠레 하게 뜬 눈 인식

02 상세 시스템

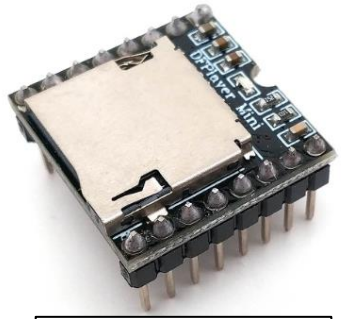
MCU – arm_cortex M4_F401re





MCU

Python과 serial 통신을 하며 trigger 신호를 받으면 경고음 재생



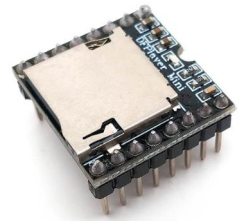
DF_Player

트리거 신호를 받으면 받은 parmeter에 맞는 경고음 재생

Speaker

사운드 크기에 맞는 경고음 발생

02 상세 시스템



DF_Player_Mini

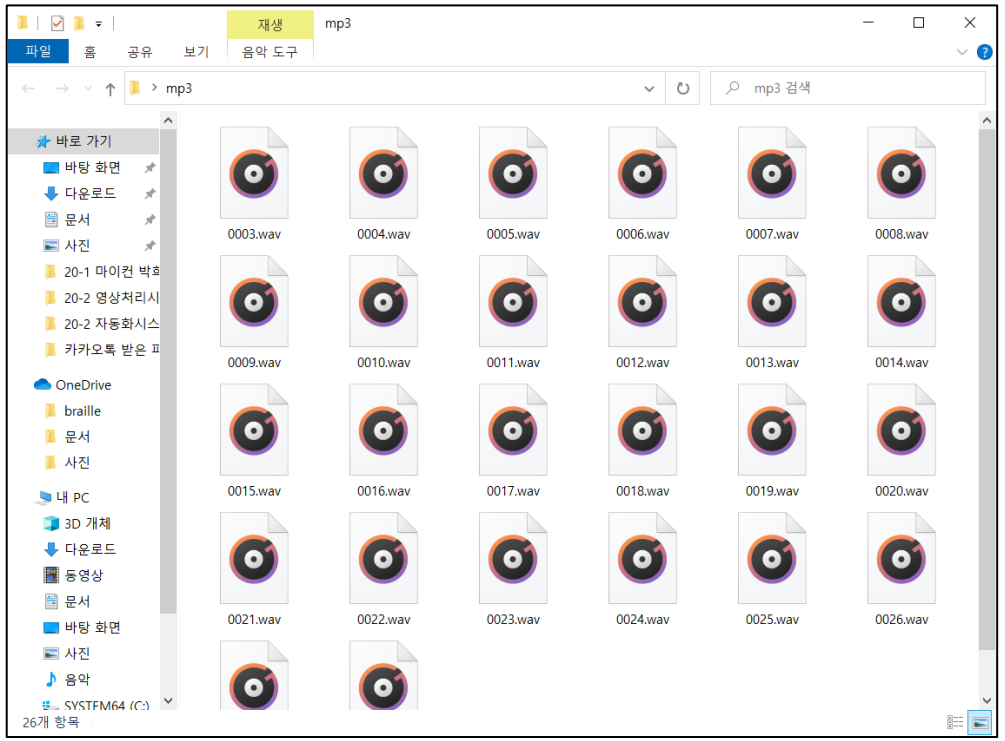
모듈 특징

간단한 serial command만 입력하면 녹음파일 재생이 가능하다.

녹음 파일 해석은 MP3, WMV을 지원한다.

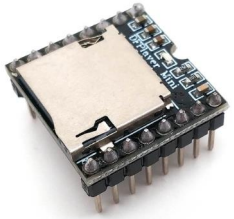
SD card(16, 32GB)로 음악을 최대 1000곡 까지 넣을 수 있다.

볼륨 크기는 10~30까지 조절 가능하다.

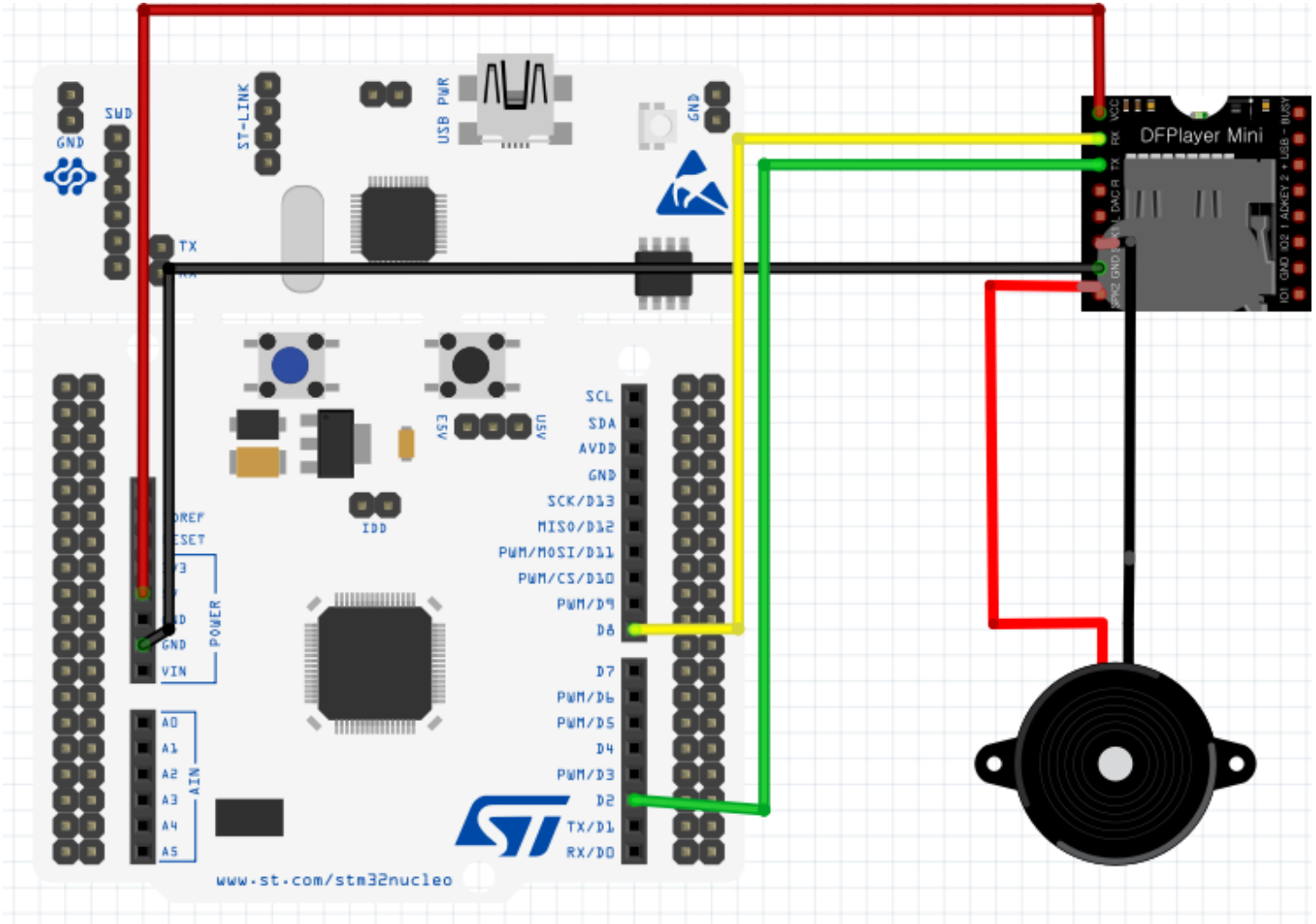


SD card를이용해 녹음본 저장

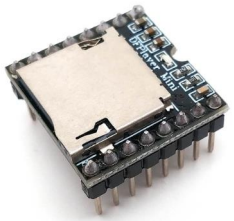
02 상세 시스템



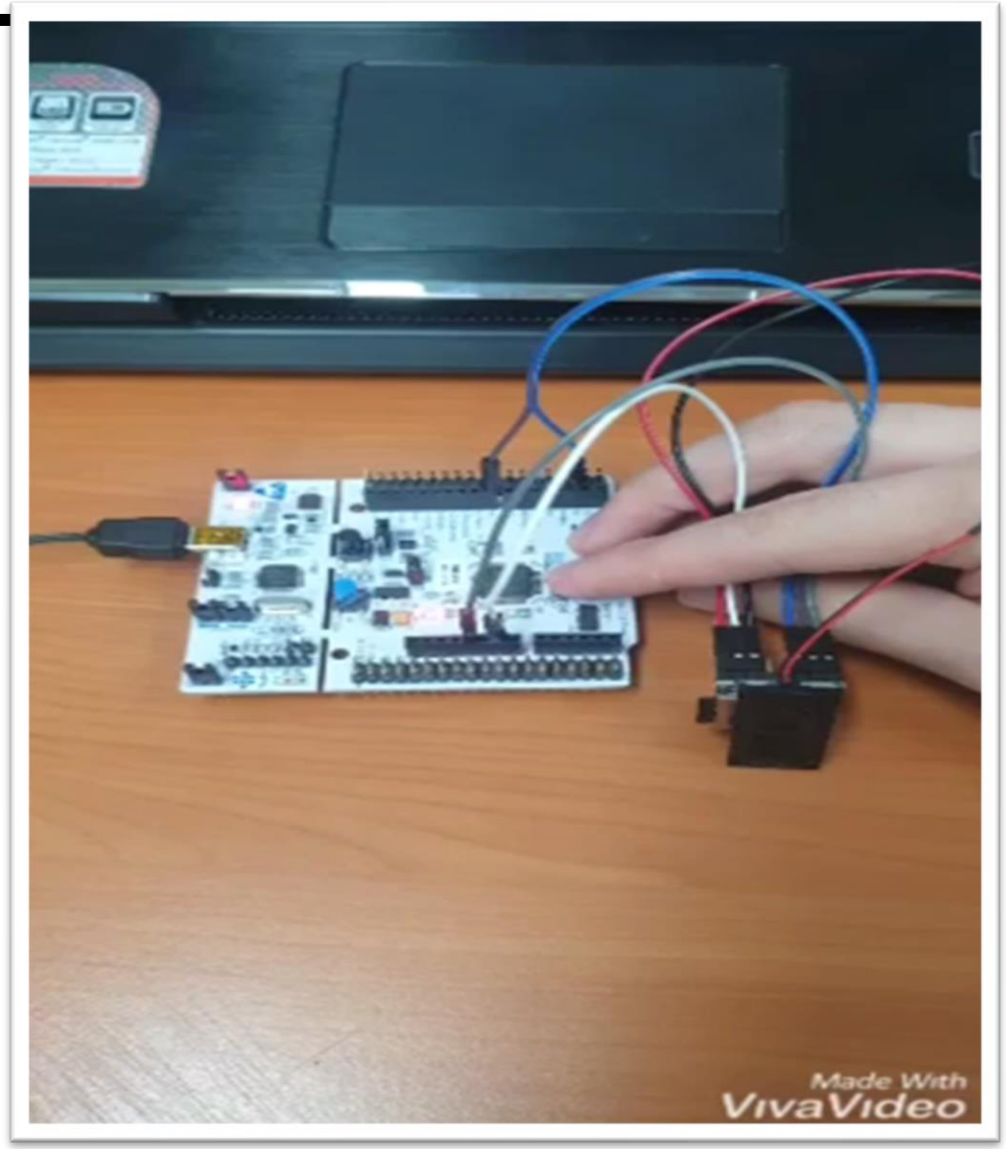
회로도


DFPlayer Mini
SWD
GND
ST-LINK
USB PWR
GND
TX
SCL
SDA
AVDD
GND
SCK/D13
MISO/D12
PWM/MOSI/D11
PWM/CS/D10
PWM/D9
D8
D7
PWM/D6
PWM/D5
D4
PWM/D3
D2
TX/D1
RX/D0
POWER
GND
VIN
AIN
A0
A1
A2
A3
A4
A5
www.st.com/stm32nucleo

02 상세 시스템



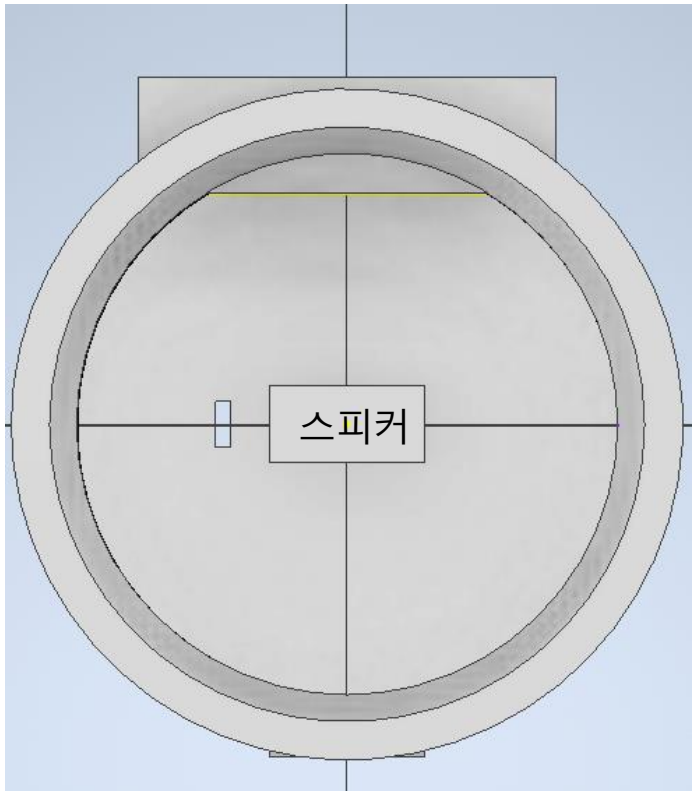
DF_Player Test



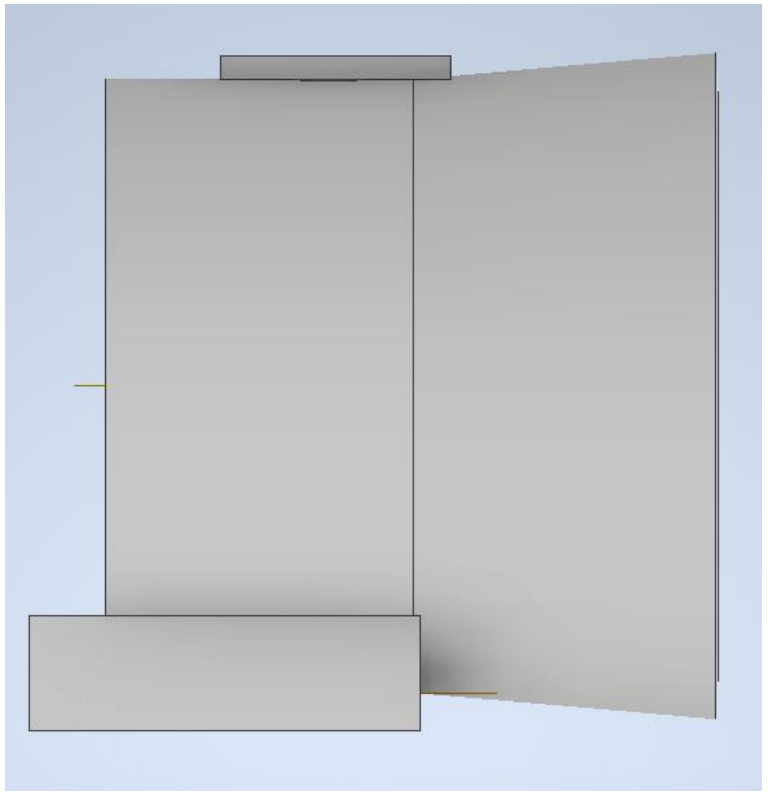
02 상세 시스템

카메라 거치대 & 스피커 케이스 제작

사시도



평면도



우측면도

03 결과 및 고찰

결과물 및 고찰

03 결과 및 고찰

결과물 Test

https://www.youtube.com/watch?v=E0NJ9TD3UEU

03 결과 및 고찰

장점

파이썬, mbed로 직접 구현하여 기능 추가가 용이하다.

다양한 경고음으로 사용자의 취향에 맞게 선택이 가능하다.

보완 요소

사람마다 입, 눈 크기가 다르므로 초기 값을 설정해주어야 한다.

졸린 눈, 하품 외에도 고개의 위아래 움직임, 시선처리 인식 추가를 하면 더욱 완성도 있을 것이다.

기대 효과

저전력에 강한 이점을 가지는 mbed는 IoT 기술에 적합하며 주행자동차 옵션으로 추가하기 좋다.

실시간으로 운전자 표정을 감지하여 졸음 운전 사고 예방에 기여할 수 있다.

녹음 파일을 SD card에 넣기만 하면 재생이 가능하므로 사용자의 취향에 맞게 조정할 수 있다.

미니 스피커를 사용하여 케이스를 카카오프렌즈, 캐릭터 미니어처로 커스터마이징이 가능하다.

Appendix

04 Appendix

역할 분담

	이현진	윤인재
자료조사	O	O
Image processing	Main	Sub
Firmware	Sub	Main
3D 케이스 설계 및 영상제작	Sub	Main

Appendix - Python code

```
from imutils import face_utils
from collections import deque

import numpy as np
import imutils
import time
import dlib
import cv2
import serial
import os

flag = False
yawn_cnt = 0
bleary_cnt = 0

# 입과 눈의 벌림을 파악하기 위한 특징점들의 인덱스를 설정.
ma = (3, 10); mb = (4, 8); mc = (0, 6)
ea = (1, 5); eb = (2, 4); ec = (0, 3)

# 시리얼 객체 선언.
if flag :
            ser = serial.Serial('COM3', 9600)

# 각 특징점의 거리연산을 위한 함수 선언.
def dist(A, B) :
            ret = sum((A - B) ** 2)**0.5
            return ret
```

```
# 눈 혹은 입에 해당하는 특징점을 np array로 받고 벌림 정도를 파악하기위해
# 필요한 특징점들의 인덱스를 받아온 뒤 벌림정도를 리턴하는 함수 작성
def get_open_ratio(points, pA, pB, pC):
            A = dist(points[pA[0]], points[pA[1]]) # 51, 59
            B = dist(points[pB[0]], points[pB[1]]) # 53, 57
            C = dist(points[pC[0]], points[pC[1]]) # 49, 55

            open_ratio = (A + B) / (2.0 * C)

            return open_ratio
# 어느정도 이상 혹은 이하일때 하품이나 눈을 게슴츠레 떴는지 판단하기위한 임계값 설정.
YAWN_THRESH = 1.1
BLEARY_THRESH = 0.2

# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("[INFO] loading facial landmark predictor...")

# cv2를 이용한 VideoCapture 객체를 선언.
cap =cv2.VideoCapture(0, cv2.CAP_DSHOW)
time.sleep(1.0)

# detector : dlib의 정면 얼굴에 대한 영역을 반환하는 get_frontal_face_detector
인스턴스를 선언
# predictor : 정면 얼굴에서 다시 각각의 특징점들을 찾아내는 shape_predictor를 미리 학습된
# shape_predictor_68_face_landmarks.dat 를 이용하여 선언.
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(os.path.dirname(os.path.realpath(__file__))
+ '/shape_predictor_68_face_landmarks.dat')
```

Appendix - Python code

```
# 입(m)에 해당하는 인덱스의 시작 및 끝점.
# 왼쪽눈(le)과 오른쪽눈(re)에 해당하는 인덱스의 시작 및 끝점을 미리 지정.
(mStart, mEnd) = (48, 67+1)
(leStart, leEnd) = (36, 41+1)
(reStart, reEnd) = (42, 47+1)

# 벌림 정도를 지속적으로 deque에 넣고 그 평균값을 이용하여 하품이나 눈을 게슴츠레 뜬 것을 파악할것.
# 이때 deque의 maxlen을 지정함으로써 시간이 좀 지난 데이터들은 평균을 취할때 제거할 수 있게됨.
# 하품의 경우 벌림의 정도가 클때에 하품이라고 인식하고 눈의 경우에는 벌림의 정도가 작을때에 게슴츠레 뜬것이라 판단했기
때문에 deque의 원소를 각각 0과 1로 초기화해둬서 초기상태를 안정적인상태로 만듦.
yawn_buff = deque([0]*30, maxlen = 30)
bleary_buff = deque([1]*30,maxlen = 30)
```

Appendix - Python code

```
while True:
        # VideoCapture객체에서 한 프레임을 받아온 뒤 Gray scale로 변환.
        _, frame = cap.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # dlib frontal_face_detector 인스턴스를 이용하여 얼굴에 대한 ROI들을 찾고, rects에 저장.
        rects = detector(gray, 0)
        # rects에 있는 ROI에 대해 얼굴의 특징점을 추출함.
        for rect in rects:
                # ROI에서 특징점을 추출 한 뒤, np array로 변환함.
                shape = predictor(gray, rect)
                shape = face_utils.shape_to_np(shape)

                # np array로 변환된 특징점들의 좌표 중 이후 각 부분(눈과 입)에 해당하는 좌표를 따로 분리.
                mouth = shape[mStart:mEnd]
                left_eye, right_eye = shape[leStart:leEnd], shape[reStart:reEnd]

                # 분리한 좌표들을 연산하여 벌림정도를 파악. 눈의 경우에는 두 눈의 평균값을 취하기로함.
                mouth_open_ratio = get_open_ratio(mouth, ma, mb, mc)
                left_open_ratio = get_open_ratio(left_eye, ea, eb, ec)
                right_open_ratio = get_open_ratio(right_eye, ea, eb, ec)

                eye_open_ratio = (left_open_ratio + right_open_ratio)/2

                # 벌림정도를 maxlen이 정해진 deque에 밀어넣음.
                bleary_buff.append(eye_open_ratio)
                yawn_buff.append(mouth_open_ratio)

                # 하품과 게슴츠레 눈을 뜬 정도를 평균값을 이용하여 판단.
                yawn_factor = sum(yawn_buff) / len(yawn_buff)
                bleary_factor = sum(bleary_buff) / len(bleary_buff)
                print(yawn_factor, bleary_factor)
```

Appendix - Python code

```
                        # 눈과 입에 해당하는 특징점을 frame에 표시
                        mouthHull = cv2.convexHull(mouth)
                        leHull = cv2.convexHull(left_eye)
                        reHull = cv2.convexHull(right_eye)
                        # for p in shape :
                        #           p = tuple(p)
                        #           cv2.circle(frame, p, 3, (0,0,255), -1)
                        cv2.drawContours(frame, [mouthHull], -1, (0, 255, 0), 1)
                        cv2.drawContours(frame, [leHull], -1, (0, 255, 0), 1)
                        cv2.drawContours(frame, [reHull], -1, (0, 255, 0), 1)

                        cv2.putText(frame, "YAWN : {0}".format(yawn_cnt), (30, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                        cv2.putText(frame, "BLEARY: {0}".format(bleary_cnt), (30, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

                        # 각 임계값에 따라서 mcu에 값을 전달시켜 mp3모듈을 작동한 뒤 버퍼를 다시 초기화
                        if yawn_factor > YAWN_THRESH:
                                    cv2.putText(frame, "yawning!",
(30,90),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)
                                    yawn_buff = deque([0]*30, maxlen = 30)
                                    yawn_cnt += 1
                                    if yawn_cnt == 3 and flag:
                                                ser.write(b'p')
                                                yawn_cnt = 0
                        if bleary_factor < BLEARY_THRESH :
                                    cv2.putText(frame, "blearing!", (30,120),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                                    bleary_cnt += 1
                                    if bleary_cnt == 100 and flag:
                                                ser.write(b's')
                                                bleary_cnt = 0
```

```
                        cv2.imshow("Frame", frame)
                        key = cv2.waitKey(1) & 0xFF
                        # q(quit)를 누를시에 프로그램을 종료하고 VideoCapture 객체를 release.
                        if key == ord("q"):
                                    break

cv2.destroyAllWindows()
cap.release()
```

Mbed Code

```
#include "mbed.h"
#include "DFPlayerMini.h" //slave
Serial pc(USBTX, USBRX);
DFPlayerMini dfplayer(D8,D2); //slave
char c;
uint8_t parameter;
int main(){

    uint8_t current_volume = 30;
    while(true){
        //pc.scanf("%s", command);
        c = pc.getc();
        pc.printf(" %c   ", c);

        if(c=='p'){
            dfplayer.mp3_play_physical (1);
        }
        else if(c=='s'){
            dfplayer.mp3_play_physical (2); }
```

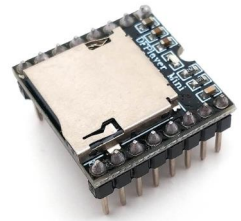
D2 : TX , D8 : RX

Max volume set

input from the python

```
        else if(c=='u'){
            pc.printf("     Volume up!  ");
            if(current_volume >= 30){
                current_volume = 30; //slave
            }
                                        else{
                current_volume += 1; //slave
                dfplayer.mp3_volumeplus(); //slave
            }
        pc.printf("      Current Volume = %d\n", current_volume); //slave
    }
    else if(c=='d'){
        pc.printf("     Volume down!  ");
        if(current_volume <= 0){
            current_volume = 0; //slave
        }
        else{
            current_volume -= 1; //slave
            dfplayer.mp3_volumeminus(); //slave
        }
        pc.printf("    Current Volume = %d\n", current_volume); //slave}}}}
```

Volume up



Volume down