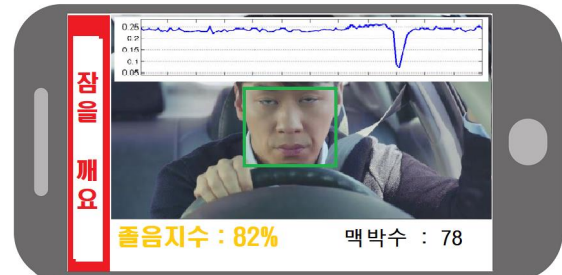
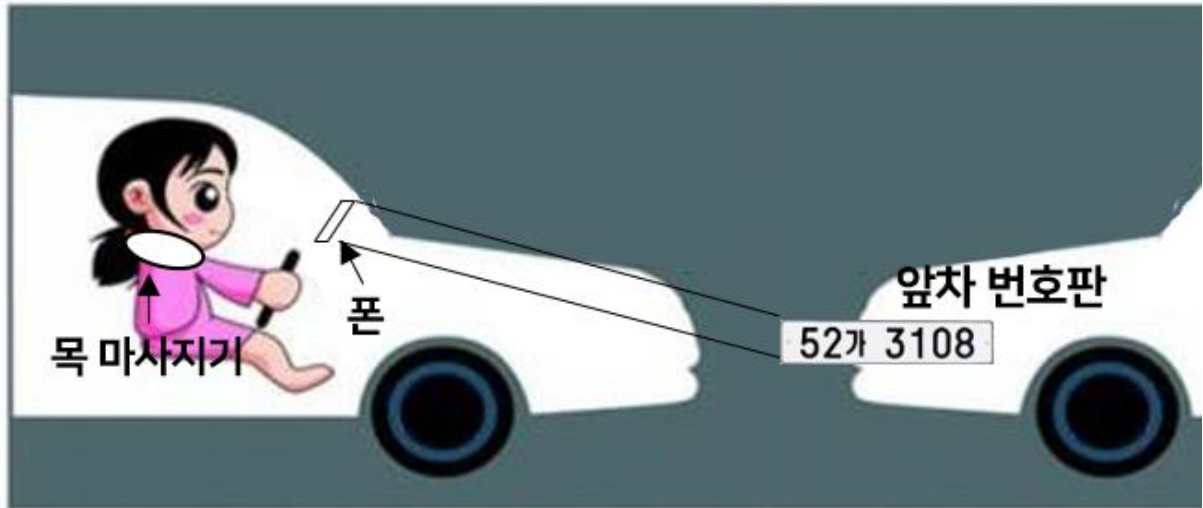


# 제18회 임베디드SW경진대회 개발일지


[임베디드SW 청소년 스타트업]

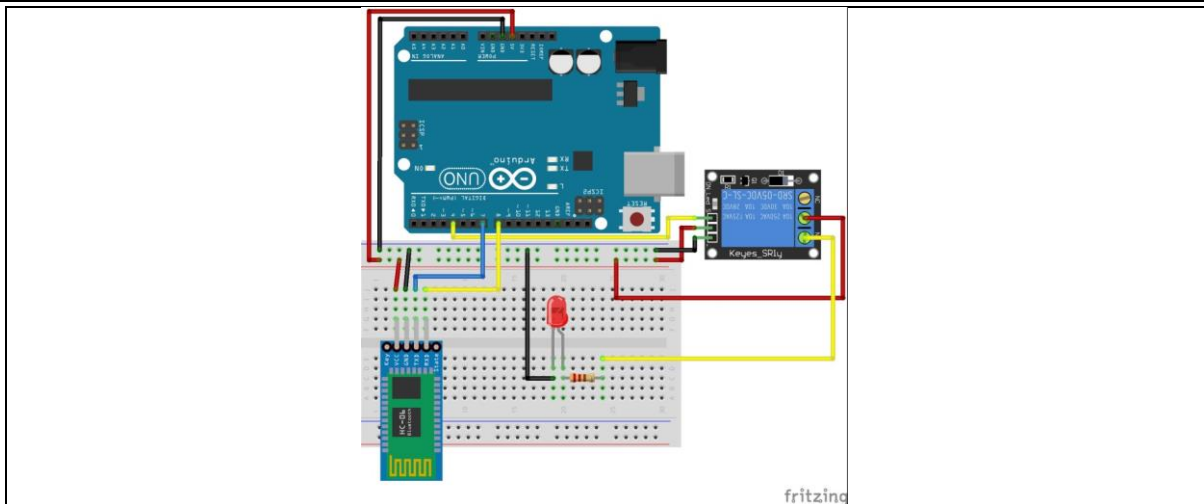


팀 명	타올주행
제품(서비스)명	운전상태 연동 마사지 장치
팀원	허 채원 팀장, 장 은우 팀원, 허 채영 팀원, 최준영 멘토

## 개발일지 목차

[illegible]

프로젝트1 (하드웨어 장치를 원격연결)	
프로젝트 기간 및 목표	
기간	6월11일 ~ 6월20일
목표	하드웨어 장치를 원격으로 연결하기
프로젝트 활동내용	
<p>운전자에게 무선으로 안마자극을 on/off 하기 위한 장치를 고민하였으며 HW의 원격제어, 무선제어로 검색하였는데 스마트 홈 장치로 검색이 찾아졌습니다.</p> <p>그래서 스마트 홈이나 스마트 팜 등의 핵심이 되는 장치를 조사해 보았더니 입력부 (센서) , 프로세스 ,출력부 (액츄에이터) 세가지 구성요소로 이루어진 시스템을 구성하고 있었습니다.</p> <p>스마트폰이나 아두이노를 통해서 판단하고 블루투스나 와이파이를 통한 원격신호 전달하면 릴레이를 이용한 출력이 이루어집니다. 그래서 우리도 시스템을 구성하기로 했습니다</p> <p>프로세싱은 스마트폰의 딥러닝을 이용하고 서는 스마트폰 카메라를 이용하였다</p> <p>출력은 블루투스와 연결된 릴레이로 안마장치의 전원을 on/off 하는 방식입니다.</p> <p>그림은 저주파 마사지기에 아두이노를 설치한 것입니다. 이 아두이노는 블루투스로 신호를 받아서 마사지 기기를 제어합니다.</p> <p>블루투스 신호에 따라서 저주파 마사지기의 강도나 종류를 바꿀 수 있습니다.</p>	
	
<p>다음 그림은 아두이노와 블루투스 그리고 릴레이를 사용한 회로도 입니다</p>	



딥러닝 학습에 의해 다양한 기능이 가능하며 우선 차량과 사람 그리고 신호등, 자전거 등을 구분하여 검색하는 기본기능을 처리하였습니다.

또한 이미지 프로세싱 기능을 적용하여 운전자의 심박수 변화를 모니터링 하면서 이상 징후가 나타나면 블루투스 신호를 발생 시킬 수 있습니다.

#### 이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

방송에서 많이 얘기되는 스마트 홈이 먼 이야기인 줄 알았습니다. 그러나 아두이노로 기본적인 구조를 만들어보니 스마트 홈이란 기본 모듈 몇가지를 다양하게 조합한 시스템이라는 것을 알았습니다. 이번 과제에는 빵판을 이용하였는데 다음에는 기판에 직접 납땜을 해서 만들고 싶습니다. 빵판 크기로 인해 제어 모듈의 크기가 큰 단점이 있었습니다

#### 지도교사 확인 및 의견

아이들이 이번 프로젝트를 수행하면서 스마트 홈이나 스마트 팜에 대해 많이 공부하였습니다. 기본이 되는 원격 신호전달 방식으로 아두이노에 연결시켜서, 진동모터를 릴레이로 조정하는 안마시스템을 구현하였습니다. 올해 자신감을 바탕으로 내년 대회에는 스마트 홈 아이템으로 도전해 보겠습니다. 우리 아이들이 이렇게 새로운 분야를 또 하나 익혀갑니다. 수고했습니다.

최준영

## 프로젝트2 (이미지방식으로 앞차와의 거리계산)

### 프로젝트 기간 및 목표

기간	6월21일 ~ 6월30일
목표	앞차와의 거리 측정 기능

### 프로젝트 활동내용

자율주행 차량은 라이다(Ridar) 라는 고가 장비를 사용하지만 우리가 만드는 제품은 별도의 장치없이 이미지만으로 거리 계산을 하고자 합니다.

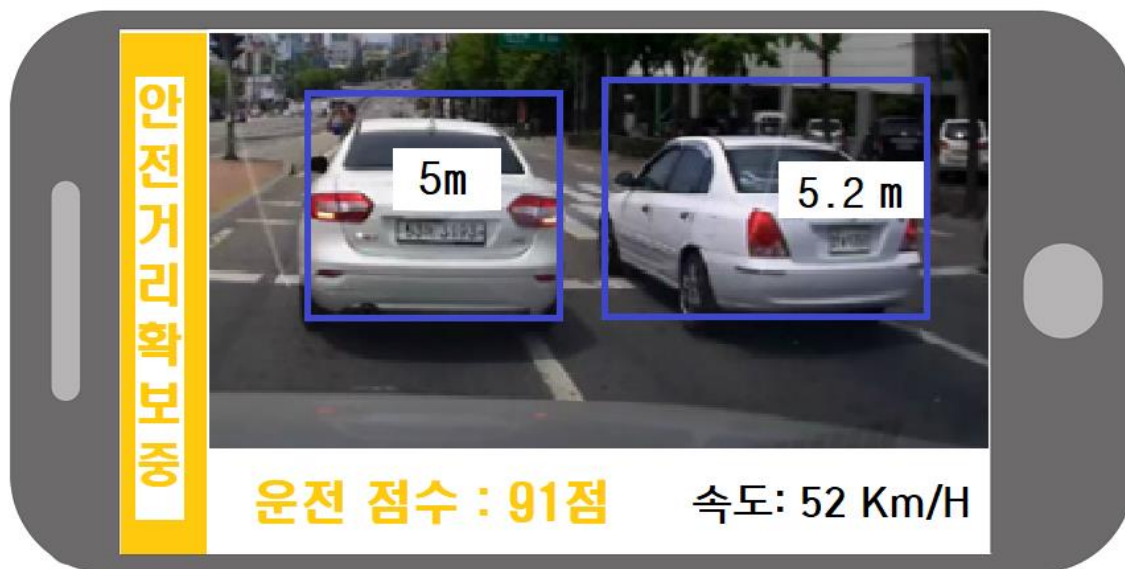
우리는 많은 고민을 통해 영상만을 사용하여 앞차와의 거리 계산을 해결하였습니다.

국내 자동차 번호판 크기는 몇 종류로 한정되며 통일되어 있다는 사실에 집중했습니다. 그래서 가까운 물체는 크게 보이고 멀리 있는 물체는 작게 보인다는 사실을 이용하여 번호판의 크기와 거리와의 관계를 레이저 거리계를 사용하여 테이블로 구성하였습니다.

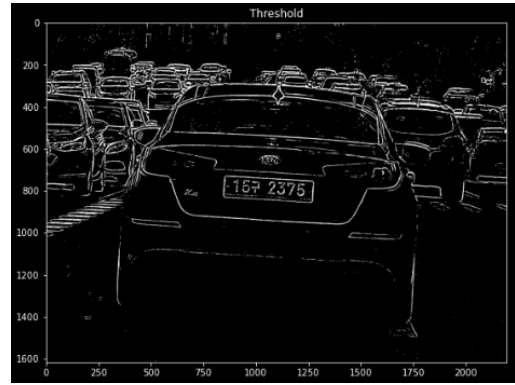
테이블 사이의 값은 보간 해서 사용하였는데 높은 정확도를 보여주었습니다.

기종이 다른 스마트폰 카메라에도 동일하게 적용하기 위해 스마트폰별로 보정 값을 준비하고 있습니다.

이 보정 값은 각 스마트폰 제조사만의 렌즈 특성이며 고유한 값입니다. 이 값을 곱해주면 모든 스마트폰에서 영상을 이용한 앞차와의 거리를 측정 가능합니다.



아래 그림은 찾아 낸 차량에서 번호판 영역을 찾는 과정입니다.



앞 차량의 번호판의 형상을 구분하기 힘들 정도로 멀리 있을 경우에는 차체 크기의 변화를 이용하여 상대적인 거리변화를 계산하였습니다.

### 이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

물체와의 거리계산을 센서가 아닌 이미지로 계산하는 방법은 찾기 힘들었습니다  
그러나 별도의 센서장치를 도입하는 것은 배보다 배꼽이 더 커질 것 같았습니다.  
번호판 크기 측정에 의한 거리계산은 상당히 정확했습니다  
그러나 차량이 너무 멀리 떨어진 경우에는 번호판이 너무 작아서 구분이 잘 안되는 단점이 있었습니다. 어느 이상 번호판이 작아지거나 보이지 않을 경우에는 차량 크기의 변화로 거리 변화를 계산하는 방법을 추가하였습니다.  
일반적인 경우에 잘되는 것이 전부 다 잘되는 것은 아니라는 것을 깨달았습니다.  
다양한 경우에 대한 Test가 필요한 것 같습니다.

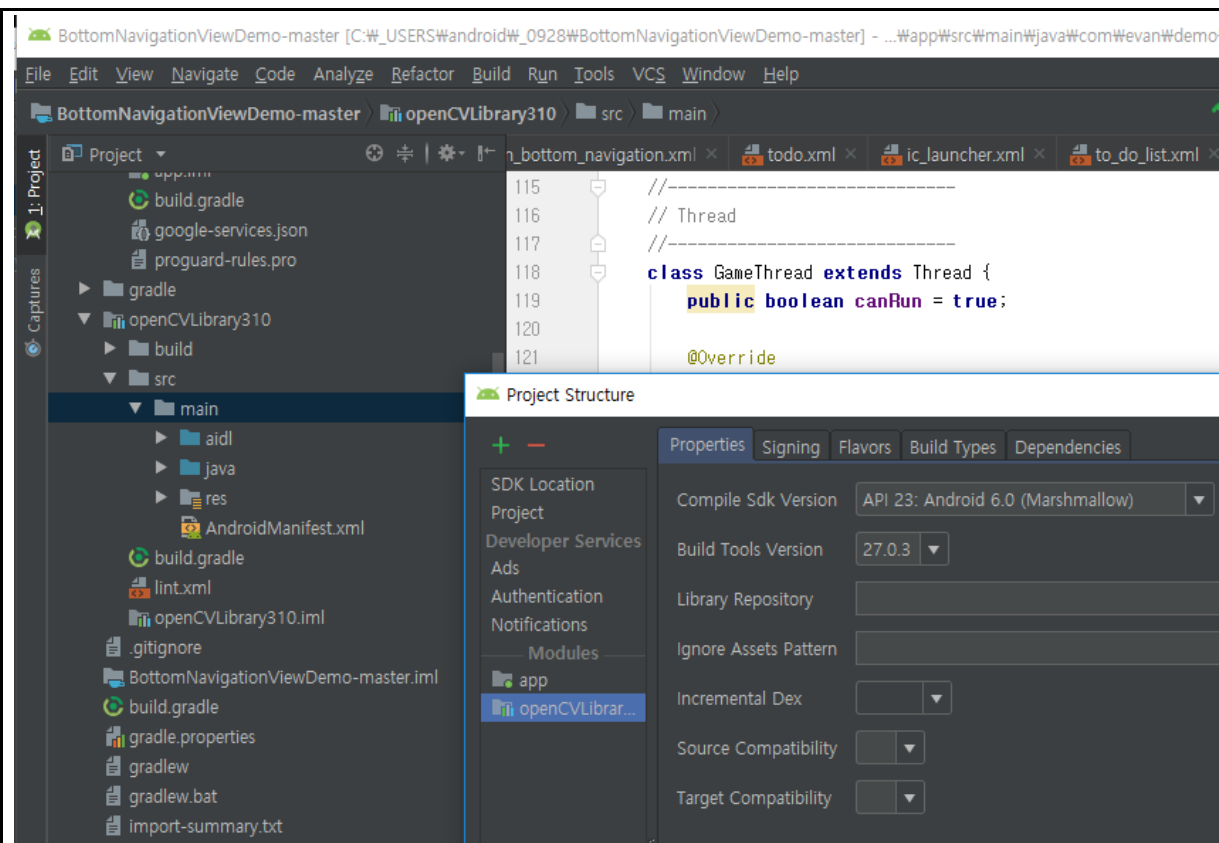
### 지도교사 확인 및 의견

본 제품에서 가장 아이디어가 돋보이는 부분입니다  
거리 계산에서 항상 기준 크기가 필요한데 규격화된 번호판을 기준 크기로 설정한 후 변화된 크기로 거리변화를 측정하는 방법이라 정확도가 보장이 되는 것 같습니다. 수고했어요

최준영

프로젝트3 ( 안드로이드에 OpenCV 연결하기 )	
프로젝트 기간 및 목표	
기간	7월1일 ~ 7월10일
목표	안드로이드에 OpenCV 연결하기
프로젝트 활동내용	
<p>스마트폰용 어플을 개발하는 경우에 카메라로 입력된 영상을 실시간으로 이미지처리 할 수 있는 방법을 네이버와 구글에서 검색하였습니다. Windows 프로그램의 영상처리에 많이 사용하는 OpenCV를 안드로이드에 JAVA로 적용할 수 있음을 찾았습니다</p> <p>첫번째 시행착오 -</p> <p>안드로이드에서 OpenCV를 사용하려면 무조건 NDK를 설치해야 하는 줄 알고 설치 후 C언어로 작성을 하였는데 너무 어려워서 할 수가 없었습니다</p> <p>다시 찾아보니 JAVA로도 OpenCV를 사용할 수 있음을 알게 되었습니다</p> <p>그래서 다시 처음부터 JAVA로 엔진을 작성하였습니다</p> <p>두번째 시행착오 -</p> <p>OpenCV는 버전이 여러 개 있습니다</p> <p>안드로이드에서 실행하기 위해서는 폰에도 라이브러리가 설치되어 있어야 합니다</p> <p>폰에 설치되는 라이브러리는 3.4버전인데 개발에 사용하는 버전과 일치하지 않으면 이상하게 작동됩니다, 이 이유로 3일쯤 고생하였습니다</p> <p>더 깊이 있게 OpenCV를 사용하려면 NDK를 사용하여 C로 작업을 하라고 하는데 너무 어렵습니다</p> <p>아래 그림은 OpenCV를 설정하는 모습입니다</p>	





- 작업 담당 (허채원 )

아래의 코드는 OpenCV를 연결하여 이미지 프로세싱 작업을 진행하는 코드입니다

```
try {
    Imgproc.cvtColor(mRgba2, mRgba2, Imgproc.COLOR_BGR2GRAY);
    Imgproc.warpAffine(mRgba2, mRgba2, rotateM, mRgba2.size() );
    Imgproc.resize( mRgba2, mRgba2, sz );
    //Size sz2 = new Size(3,3 );
    //Imgproc.blur( mRgba2, mRgba2,sz2);
    mRgba2.convertTo(mRgba2, 0); //converting the image to match with the type of the cameras image
    bitmap2 = Bitmap.createBitmap(mRgba2.cols(), mRgba2.rows(), Bitmap.Config.ARGB_8888);

    mRgba2.copyTo(mRgba22);
    bitmap22 = Bitmap.createBitmap(mRgba22.cols(), mRgba22.rows(), Bitmap.Config.ARGB_8888);

    //issue - this is going to create a Runnable for each frame
    Runnable runnable = new Runnable() {
        @Override
        public void run() {
            try {
                Utils.matToBitmap(mRgba2, bitmap2 );
                iview11.setImageBitmap( bitmap2 );
                iview11.invalidate();
                sleep(1);
            }
        }
    };
}
```



```

        // Compare2Image();
    } catch (Exception e) {
        //
    }
}

};
new Thread(runnable).start();
}
catch(Exception ex) {
    Log.i(TAG, "Exception onCameraFrame");
}

```

#### 이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

안드로이드도 능숙하게 사용 못하는데 OpenCV까지 해야 되어서 너무 힘들었습니다

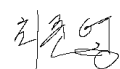
이미지 처리 계산이 너무 오래 걸리면 안드로이드 운영체제가 프로그램을 종료 시켜 버리는 상황도 발생하였습니다. 해결 방법을 많이 고민하였습니다.

찾아낸 방법은 이미지가 커서 시간이 많이 걸린 것 이었으며 이미지를 축소하여 처리하는 방식으로 시간을 줄였습니다.

다음에는 OpenCV를 더 공부하여 전문가처럼 되고 싶습니다

#### 지도교사 확인 및 의견

전체 개발과정에서 중요한 부분인데 자료가 부족하여 힘들었습니다.  
있는 자료도 예전 버전으로 실행되지 않는 경우도 많았습니다. 하지만  
최종적으로 구동되는 모습을 보고 만족하였습니다.



## 프로젝트4 (안마장치 형태결정)

## 프로젝트 기간 및 목표

**기간** 7월11일 ~ 7월20일

## 목표 | 안마장치 형태 설정

## 프로젝트 활동내용



다양한 안마기와 차량용 배게 및 쿠션을 구입 하였습니다.

초기 개발 계획에서는 목에 끼우는 안마기 형태로 결정하였습니다.

아래 형태의 제품이며 아두이노와 블루투스 그리고 진동모터를 결합 하였습니다.



그러나 차량에서 테스트 해본 결과 매번 들고 다녀야 하는 큰 단점이 보였습니다.

그래서 차량용 배게를 선택하였습니다. 그러나 차량용 배게는 운전자의 몸이 앞으로

기울어지면 배개와 목이 떨어져서 자극이 전달이 되지 않는 새로운 단점이 나타났습니다.

이러한 단점을 없애기 위해서 차량용 등받이 쿠션형태를 채택했습니다. 등받이 쿠션은 운전하는 동안 계속 운전자의 등과 접촉상태를 유지합니다. 그래서 차량에 탑승해서 스마트폰만 핸들 뒤에 거치 시키면 운전 연동 마사지 기능을 사용할 수 있습니다.

차량용 목 베개나 쿠션에 마사지 모듈을 내장시키는 방식입니다. 저희 같은 신생업체에서는 완성도를 사용 수준으로 높이기 어려우므로 판매중인 쿠션이나 베개에 저희 모듈을 집어넣는 방식을 채택하여 제품의 완성도를 높이하고자 합니다.

제작 소요기간은 3개월 정도로 예상됩니다. 차량 쿠션이나 제어 모듈은 알리 익스프레스에서 구입합니다. 포장 박스의 경우에는 디자인 및 생산 제조를 국내에서 진행합니다. 제품 디자인 박스 등은 정부에서 진행하는 지원사업으로 추진하고자 합니다.



아래 그림은 가장 간단한 형태의 제어장치입니다.

상자안에 들어있는 보드는 블루투스과 연산장치 그리고 릴레이가 결합된 작은 제품이며 이것만 있으면 대부분의 전자제품을 휴대폰으로 제어할 수 있습니다.



#### 이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

사용자가 사용하기 편한 안마장치를 찾기 위해 많은 지출이 있었음을 반성합니다. 조금 더 생각해보고 구매를 하였다면 이렇게까지 많은 제품을 사지는 않았을 텐데 과했다는 생각이 많이 듭니다.

대회가 끝나면 각 제품에 작은 기능을 넣어서 부모님과 선생님들께 선물하겠습니다. 다음 프로젝트에서는 구매에 있어서는 조금 더 생각해보고 결정하겠습니다.

#### 지도교사 확인 및 의견

사용자가 사용하기 편한 제품을 찾기 위해 아이들이 차에서 다양한 테스트를 진행하였습니다. 선생님이 시켜서 하는 피동적인 모습이 아니라 자신들의 목표를 위해 적극적으로 서로 상의하며 토론하는 모습이 좋았습니다. 저도 차량용 허리받침이 제일 마음에 들었습니다.

최준영



## 프로젝트5 (졸음운전 상태측정)

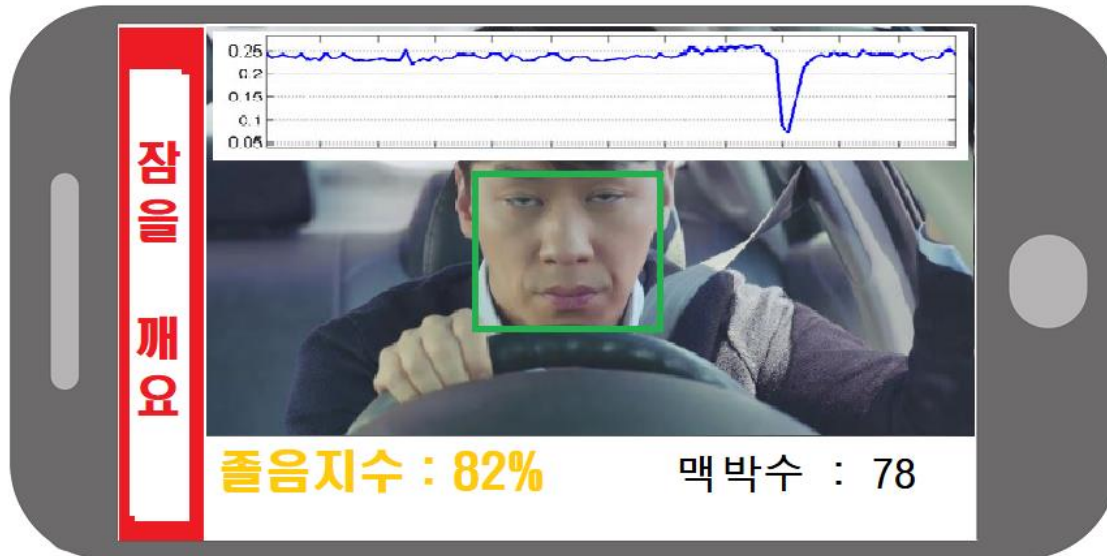
### 프로젝트 기간 및 목표

기간 7월21일 ~ 8월10일

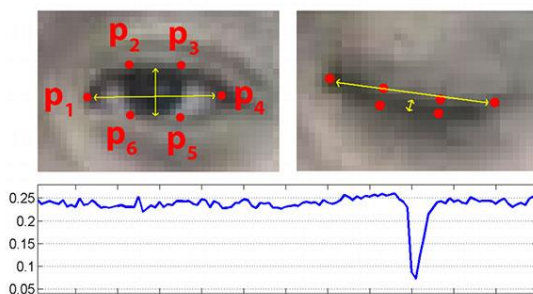
목표 졸음 운전 상태 측정

### 프로젝트 활동내용

운전자가 졸고 있을 때 이를 인식하는 방법은 얼굴과 눈을 검출하여 눈꺼풀이 닫힌 시간과 고개가 숙여진 상태 등을 고려하여 판단합니다.



영상에서 눈이 감겼는지 판단하는 근거는 Tereza Soukupova' 와 Jan Cech의 논문을 참조하였습니다. (21st Computer Vision Winter Workshop, Luka C̣ehovin, Rok Mandeljč, Vitomir Ṣtruc (eds.), Rimske Toplice, Slovenia, February 3–5, 2016, "Real-Time Eye Blink Detection using Facial Landmarks")



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

```
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

EYE_AR_THRESH = 0.2
#EYE_AR_THRESH = 0.175
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
ITER = 0
ALARM_ON = False

PREDICTOR_PATH = "68_face_landmarks.dat"

# 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"])
predictor = dlib.shape_predictor(PREDICTOR_PATH)

# 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"]

capture = cv2.VideoCapture(0)

while True:
    has_frame, frame = capture.read()
    if not has_frame:
        print('Can\'t get frame')
        break
    frame = imutils.resize(frame, width=300, height=200)
    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
# ear = leftEAR

# 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)

x.append(i)
y.append(ear)

ax.plot(x, y, color='b')
fig.canvas.draw()
ax.set_xlim(left=max(0, i-50), right=i+50)

#time.sleep(0.1)
i += 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
    ITER += 1

if ITER == 2 :
    COUNTER += 1
cv2.putText(frame, "COUNTER: {:d}".format(COUNTER), (10, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
cv2.putText(frame, "EAR: {:.2f}".format(ear).format(COUNTER), (10, 60),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
cv2.imshow('frame', frame)

capture.release()
cv2.destroyAllWindows()


```

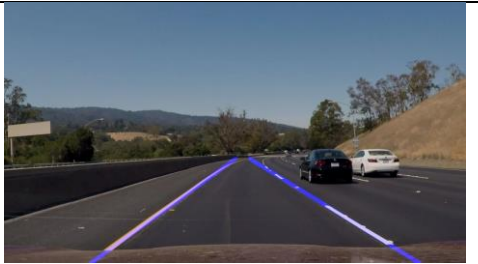



졸음 상태의 측정은 다양한 변수가 존재하므로 한가지로만 판단할 경우에는 예외상황으로 놓치는 경우가 많았습니다.

그래서 (1) EAR 값에 의한 눈 깜박임 측정 (2) 얼굴 움직임 측정 (3) 심박수의 변화 (4) 움직임 거동의 변화 이 값들을 조합하여 졸음운전을 판단하도록 발전시키고 있습니다.

그런데 예외상황의 처리가 귀찮고 어려움이 많습니다

지도교사 확인 및 의견	
눈의 위치를 정지시켜 놓고 눈 깜박임을 체크하면 거의 완벽하게 졸음상태를 찾아냅니다. 그러나 실제 차량에 부착해서 테스트해보니 얼굴의 움직임, 터널 진입, 야간, 고개 숙임 등 많은 방해요소를 맞이하였습니다. 학생들이 이런 경우에도 다 찾아내는 프로그램으로 만들겠다 합니다. 완성도에 신경 쓰는 모습은 대단히 칭찬할 만 합니다.	

프로젝트6 (도로영상에서 차선찾기)	
프로젝트 기간 및 목표	
기간	8월11일 ~ 8월20일
목표	도로영상에서 실시간으로 차선 찾아내기
프로젝트 활동내용	
<p>차선 인식 방법이 필요한 것은 저희들이 사용자가 차선을 바꿀 때마다 진동을 주어서 위험 운전임을 알리고 싶어서 입니다.</p> <p>차선인식은 도로 이미지를 입력 받아서 스무딩 작업으로 노이즈를 제거하고 이미지에 Hough Line 방법으로 Line을 추출합니다. 추출된 라인에서 쌍으로 된 라인을 차선으로 인식 시켰습니다. 아래 그림은 주요 코드 부분입니다. 이 코드로 찾아낸 차선을 파란색으로 표시하였습니다.</p>	
<pre>// 그레이스케일 영상으로 변환하여 에지 성분을 추출 cvtColor(img_filtered, img_gray, COLOR_BGR2GRAY); GaussianBlur(img_gray, img_gray, Size(3, 3), 0, 0); Canny(img_gray, img_edges, 50, 150); // 차선 검출할 영역을 제한함 //(진행방향 바닥에 존재하는 차선으로 한정) img_edges = region_of_interest(img_edges, points); UMat uImage_edges; img_edges.copyTo(uImage_edges); // 직선 성분을 추출vector&lt;Vec4i&gt; lines; HoughLinesP(uImage_edges, lines, rho, theta, hough_threshold, minLineLength, maxLineGap);</pre>	
이번 프로젝트에 대한 반성 / 다음 프로젝트 계획	
<p>hough line 방법이 어느정도 차선을 잡아냅니다. 곡선주행에서는 hough line이 차선을 놓친다. 곡선 주행시에 라인을 잡아 주기 위한 방법이 필요하며 차선을 바꿀 때 깜박이를 켤 때와 꺼지 않을 때를 구분하여 경고를 보내도록 발전시키겠습니다.</p>	
지도교사 확인 및 의견	
<p>차선 인식을 영상인식으로 경험해 보았습니다. 판매되는 차량에 탑재된 기술이 대단함을 느낍니다.</p> <p>스마트폰 영상으로 영상을 인식하고 차선을 찾아내는 것은 샘플영상에서는 잘 되었지만 실제 도로환경에서는 안되는 경우가 많이</p>	

있었습니다. 현실 세계의 데이터가 정제되지 않았다는 것을 배울 수  
있는 좋은 기회였습니다.

## 프로젝트7 (영상에서 심박수 확인)

### 프로젝트 기간 및 목표

기간	8월21일 ~ 9월10일
목표	영상에서 심박수 체크하기

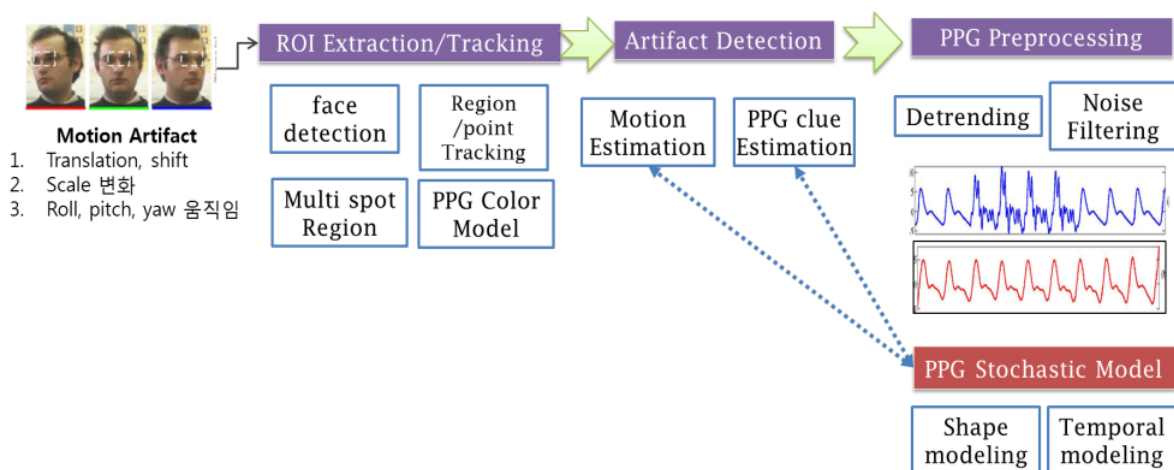
### 프로젝트 활동내용

얼굴 영상에서 심박수를 체크하는 방법은 이마와 코부분의 영상에서 녹색 성분만 추출하여 푸리에 방식으로 신호를 분석하면 근사적인 심박수를 측정할 수 있다는 논문과 자료가 많이 발표되었습니다.

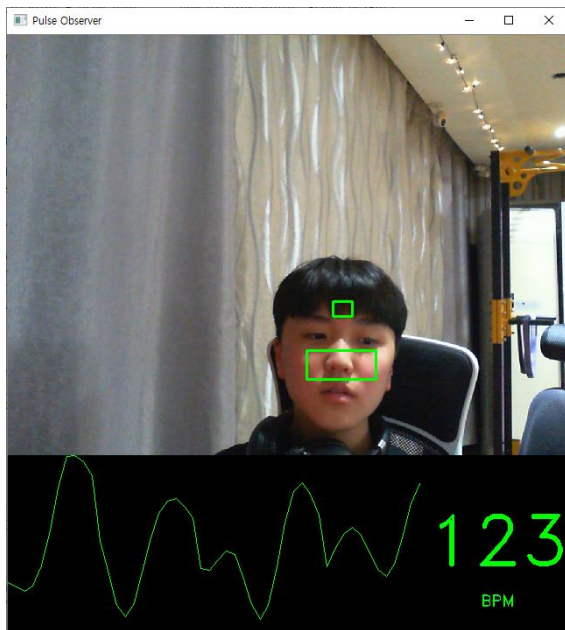
이 방법을 활용하여 운전자의 심박수 변화를 측정하고 이를 졸음운전 판단 자료에 사용하기로 했습니다.

인터넷상에 발표된 자료를 이용하여 코드를 작성하였으며 손가락에 끼우는 형태의 맥박계를 구하여 신호를 비교하였습니다. 검토 결과 절대적인 수치 값 보다 상대적인 심장박동의 경향이 유사하게 나오는 것을 확인하였습니다.

얼굴 피부에서 미세하게 혈류변화가 일어나는 현상을 카메라가 촬영하고 그 영상을 분석하여 실시간으로 혈류변화성분을 심장박동 신호로 변환하는 기술과 사람의 얼굴은 상시 움직임을 동반하므로 신호의 정확성을 위해 잡음을 제거하는 것이 핵심기술입니다



먼저 영상에서 얼굴을 찾아내고 찾아낸 얼굴에서 이마와 코부분의 ROI를 설정합니다. 노이즈를 제거하고 설정된 ROI 부분의 색상을 분석합니다. 분석된 색상 값으로 FFT를 처리하고 이 값을 이용하여 심박수를 유추해 냅니다.



```
# Calculate the pulse in beats per minute (BPM)
def compute_bpm(filtered_values, fps, buffer_size, last_bpm):
    # Compute FFT
    fft = np.abs(np.fft.rfft(filtered_values))

    # Generate list of frequencies that correspond to the FFT values
    freqs = fps / buffer_size * np.arange(buffer_size / 2 + 1)

    # Filter out any peaks in the FFT that are not within our range of [MIN_HZ, MAX_HZ]
    # because they correspond to impossible BPM values.
    while True:
        max_idx = fft.argmax()
        bps = freqs[max_idx]
        if bps < MIN_HZ or bps > MAX_HZ:
            fft[max_idx] = 0
        else:
            bpm = bps * 60.0
            break

    # It's impossible for the heart rate to change more than 10% between samples,
    # so use a weighted average to smooth the BPM with the last BPM.
    if last_bpm > 0:
        bpm = (last_bpm * 0.9) + (bpm * 0.1)

    return bpm

def filter_signal_data(values, fps):
    # Ensure that array doesn't have infinite or NaN values
    values = np.array(values)
    np.nan_to_num(values, copy=False)

    # Smooth the signal by detrending and demeaning
```

```

detrended = signal.detrend(values, type='linear')
demeaned = sliding_window_demean(detrended, 15)
# Filter signal with Butterworth bandpass filter
filtered = butterworth_filter(demeaned, MIN_HZ, MAX_HZ, fps, order=5)
return filtered

# Get the average value for the regions of interest. Will also draw a green rectangle around
# the regions of interest, if requested.
def get_roi_avg(frame, view, face_points, draw_rect=True):
    # Get the regions of interest.
    fh_left, fh_right, fh_top, fh_bottom = get_forehead_roi(face_points)
    nose_left, nose_right, nose_top, nose_bottom = get_nose_roi(face_points)

    # Draw green rectangles around our regions of interest (ROI)
    if draw_rect:
        cv2.rectangle(view, (fh_left, fh_top), (fh_right, fh_bottom), color=(0, 255, 0), thickness=2)
        cv2.rectangle(view, (nose_left, nose_top), (nose_right, nose_bottom), color=(0, 255, 0), thickness=2)

    # Slice out the regions of interest (ROI) and average them
    fh_roi = frame[fh_top:fh_bottom, fh_left:fh_right]
    nose_roi = frame[nose_top:nose_bottom, nose_left:nose_right]
    return get_avg(fh_roi, nose_roi)

# Main function.
def run_pulse_observer(detector, predictor, webcam, window):
    roi_avg_values = []
    graph_values = []
    times = []
    last_bpm = 0
    graph_height = 200
    graph_width = 0
    bpm_display_width = 0

    # cv2.getWindowProperty() returns -1 when window is closed by user.
    while cv2.getWindowProperty(window, 0) == 0:
        ret_val, frame = webcam.read()
        # The original frame will be used to compute heart rate.
        view = np.array(frame)

        # Detect face using dlib
        faces = detector(frame, 0)
        if len(faces) == 1:
            face_points = predictor(frame, faces[0])
            roi_avg = get_roi_avg(frame, view, face_points, draw_rect=True)
            roi_avg_values.append(roi_avg)
            times.append(time.time())

```

```

# Don't try to compute pulse until we have at least the min. number of
frames
if curr_buffer_size > MIN_FRAMES:
    # Compute relevant times
    time_elapsed = times[-1] - times[0]
    fps = curr_buffer_size / time_elapsed # frames per second
    # Clean up the signal data
    filtered = filter_signal_data(roi_avg_values, fps)

    graph_values.append(filtered[-1])
    if len(graph_values) > MAX_VALUES_TO_GRAPH:
        graph_values.pop(0)

    # Draw the pulse graph
    graph = draw_graph(graph_values, graph_width, graph_height)
    # Compute and display the BPM
    bpm = compute_bpm(filtered, fps, curr_buffer_size, last_bpm)
    bpm_display = draw_bpm(str(int(round(bpm))), bpm_display_width, graph_height)

    last_bpm = bpm
    graph = draw_graph_text('No face detected', (0, 0, 255), graph_width, graph_height)
    bpm_display = draw_bpm('--', bpm_display_width, graph_height)

    graph = np.hstack((graph, bpm_display))
    view = np.vstack((view, graph))
    cv2.imshow(window, view)

```

### 이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

얼굴로 맥박수를 측정하는 아이디어는 멋진 것 같아 자료 검색을 많이 하였으며 제시된 방법으로 구현하였습니다.

비교를 위해 손가락에 끼우는 맥박계를 구입하였으며 수치 값을 비교하였습니다

절대값에는 조금 차이가 있었습니다

그러나 맥박이 올라가거나 맥박이 내려오는 것 같은 상대적인 거동에 대해서는 유사하게 찾아내는 것을 확인했습니다

다음 프로젝트에서는 심박수의 정확도를 조금 더 높이도록 하겠습니다.

### 지도교사 확인 및 의견

상당히 어려운 분야였습니다. 다행히 얼굴인식 라이브러리가 있어서 활용이 가능했습니다. 아이들이 할 수 있을까 염려되





었으나 이것을 해내는 것을 보고 생각보다 아이들이 코딩능력 이 뛰어나다고 느꼈습니다	
---	--

## 프로젝트8 (딥러닝을 이용한 차량과 사람인식)

### 프로젝트 기간 및 목표

기간	9월11일 ~ 9월20일
목표	딥러닝을 이용하여 차량과 사람을 인식합니다

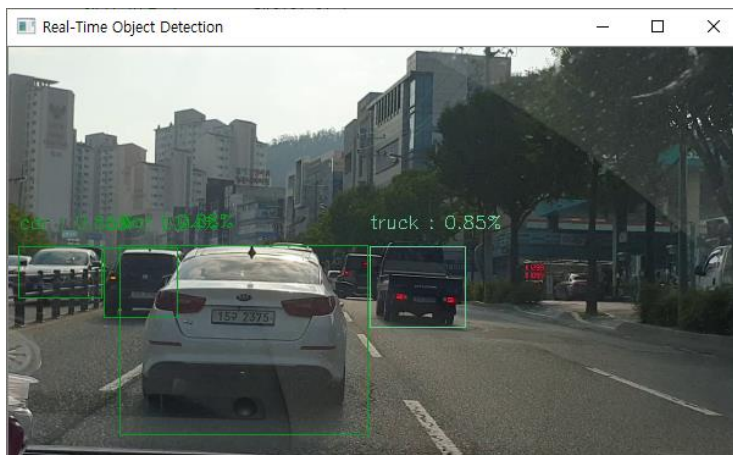
### 프로젝트 활동내용

타율 주행 프로그램은 스마트폰 카메라로 전방을 촬영하며 이 영상에서 차량과 물체를 인식합니다. 인식된 물체의 크기가 점차 커지거나 작아지는 비율을 이용하여 접근 중인지 멀어지는지를 알 수 있게 됩니다. 어느 이상 가까워지면 마사지 장치를 가동시켜 운전자에게 자극을 보냅니다.

영상에서 물체를 인식하는 방법은 딥러닝 방법인 YOLO V3를 이용하였습니다.

자율주행 분야에서도 많은 적용이 이루어지고 있는 모델입니다.

아래 그림은 해당 모델을 이미지에 적용하여 차량을 검색한 모습입니다 .



```
LABELS = open(labelsPath).read().strip().split("\n")
# 각 클래스에 대한 색깔 random 지정
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3), dtype="uint8")
weightsPath = "D://data//yolov3.weights" #os.path.sep.join(["yolo-coco", "yolov3.weights"]) # 가중치
configPath = "D://data//yolov3.cfg" #os.path.sep.join(["yolo-coco", "yolov3.cfg"])
# 모델 구성
# COCO 데이터 세트(80 개 클래스)에서 훈련된 YOLO 객체 감지기 load
print("[YOLO 객체 감지기 loading...]")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
# YOLO 에서 필요한 output 레이어 이름
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
vs = cv2.VideoCapture(0)
(W, H) = (None, None)
```

```

while True:
    # 프레임 읽기
    ret, frame = vs.read()
    # 프레임 크기 지정
    frame = imutils.resize(frame, width=600)
    # 프레임 크기
    if W is None or H is None:
        (H, W) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (256,256), swapRB=True, crop=False)

    # 객체 인식
    net.setInput(blob)
    layerOutputs = net.forward(ln)
    boxes = []
    confidences = []
    classIDs = []
    # layerOutputs 반복
    for output in layerOutputs:
        # 각 클래스 레이블마다 인식된 객체 수 만큼 반복
        for detection in output:
            # 인식된 객체의 클래스 ID 및 확률 추출
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]
            # 객체 확률이 최소 확률보다 큰 경우
            if confidence > args["confidence"]:
                box = detection[0:4] * np.array([W, H, W, H])
                (centerX, centerY, width, height) = box.astype("int") # (중심 좌표 X
, 중심 좌표 Y, 너비(가로), 높이(세로))
                # bounding box 왼쪽 위 좌표
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))
                # bounding box, 확률 및 클래스 ID 목록 추가
                boxes.append([x, y, int(width), int(height)])
                confidences.append(float(confidence))
                classIDs.append(classID)
    # bounding box 가 겹치는 것을 방지(임계값 적용)
    idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"], args["threshold
"])

    # 인식된 객체가 있는 경우
    if len(idxs) > 0:
        # 모든 인식된 객체 수 만큼 반복
        for i in idxs.flatten():
            # bounding box 좌표 추출
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])
            # random 으로 지정된 색깔
            color = [int(c) for c in COLORS[classIDs[i]]]
            # bounding box 출력
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 1)

```

```

# 클래스 ID 및 확률
text = "{} : {:.2f}%".format(LABELS[classIDs[i]], confidences[i])
# label text 잘림 방지
y = y - 15 if y - 15 > 15 else y + 15
# text 출력
cv2.putText(frame, text, (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color,
1)
# 프레임 출력
cv2.imshow("Real-Time Object Detection", frame)
key = cv2.waitKey(1) & 0xFF

```

#### 이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

YOLO V3 모델은 정확성은 무척 높았습니다. 그런데 계산량이 많아서인지 실시간 처리가 많이 어려웠습니다. 그래서 저희는 모든 영상에 YOLO V3를 적용하지 않고 인터벌을 길게 해서 한번 씩 YOLO V3로 물체를 인식하고 그 사이는 찾아낸 물체를 추적하는 방법을 사용하여 실간 계산이 가능하도록 고민하였습니다.

이걸 하면서 쉽게 되는 건 하나도 없다는 말이 저절로 나왔습니다.

#### 지도교사 확인 및 의견

YOLO V3를 해 본 것은 인공지능에 평균이상의 수준에 도달했다고 생각합니다. 코딩과 인공지능까지 우리 아이들의 도전은 어디까지 일지 앞으로 더 기대됩니다.



## 프로젝트9 (판매중인 블루투스 안마기에 접속)

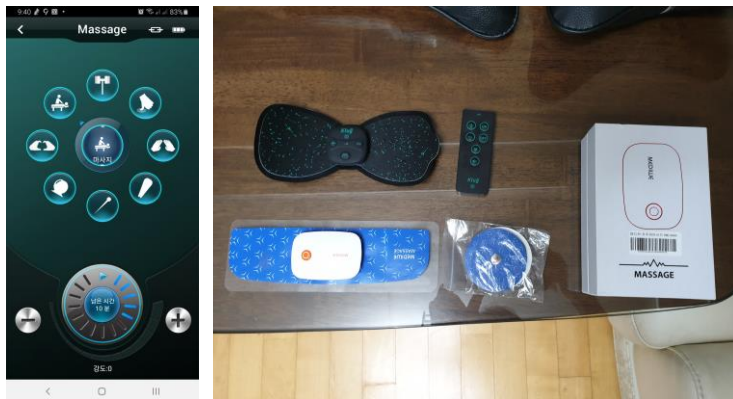
### 프로젝트 기간 및 목표

**기간** 9월21일 ~ 9월30일

**목표** 타사 블루투스 안마기를 블루투스 방식 접근하기

### 프로젝트 활동내용

아래 그림은 시판중인 블루투스 목 안마기 제품이며 안드로이드 어플로 마사지 종류와 강도를 조절할 수 있으며 3만원에 판매 중입니다.



이 제품은 중국 OEM 제품이며 국내 의료기기 업체가 판매 중입니다. 이 업체에 의뢰하여 작동에 관련된 블루투스 신호를 공유하고자 합니다.

해당 업체는 스마트 기능에 대한 홍보에 사용할 수 있는 권리를 부여합니다. 이럴 경우에 제품 개발시간은 팜플렛 제작 시간과 프로그램에서 블루투스 신호 등록 시간만 필요합니다. 블루투스 신호만 알면 해당 제품의 안마기를 원격제어가 가능해집니다. 제품 개발 시간은 한달정도로 예상합니다

현재 해당 국내업체 정보와 중국 OEM 업체 정보 그리고 중국 어플 제작자의 이메일 주소를 확보하였습니다.

### 이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

우리가 만드는 제품에 다른 회사제품을 이용할 수 있다는 생각은 꿈에도 해본적이 없는데, 선생님이 상업적인 완성도를 위해 특정 부분을 사서 넣을 수 있다는 말씀에 제품화 후에 판매가 안정되면 자체 생산품으로 바꾸는 방법이 망할 위험을 줄여주는 것 같아서 열심히 타사제품을 분석하였습니다.

판매를 위한 제품이 기능이 간단해도 완성도가 높다는 생각이 많이 들었습니다.  
우리도 판매까지 가기 위해 완성도를 더 높일 필요가 있습니다.

**지도교사 확인 및 의견**

창업을 위한 제품을, 만들어 파는 제품처럼 보이는 쪽으로 방향을 정했습니다. 한번 만들어보는 것이 아니라 한번 팔아보는 제품, 소비자에게 좋다는 별점을 받을 수 있는 제품이 되고자 했습니다. 우리 학생들이 고생이 많았습니다.

