창업연계공학설계입문

AD Project 보고서

<주제: 퇴근하고 안전하게 집가기>

20191541 강현서

20191553 권순호

20191554 권은우

20191560 김서경

20165160 장진형

목차

1. 주제 및 주제선정이유

2. 주요기능
   2.1. 직선주행, 곡선주행
   2.2. 사람 인식
   2.3. 횡단보도 인식
   2.4. 신호등 인식
   2.5. 주차

3. 프로젝트결과

4. 수행후기

1. 주제선정이유

‘창업연계공학설계입문’ 과목의 목표는 임베디드시스템 프로그래밍 기초 센서, 액츄에이터 등을 제어하는 것, 그리고 자율주행 알고리즘을 습득하고 설계하는 것이다.
우리 조는 이 과목에서 배운 내용을 바탕으로 현실적인 제약이 따르는 상황에서의 시스템 설계에 익숙해지고, 도전적인 목표 달성을 위해 ‘퇴근길 안전하게 귀가하기’를 주제로 이 프로젝트를 이행하게 되었다.

‘퇴근길 안전하게 귀가하기’는 프로젝트는 퇴근에서 부터 집까지의 주행 상황을 가정하여 직선, 곡선도로를 주행한다. 또한 도로를 주행하며 발생하는 여러 요소(신호등 인식, 사람 인식, 횡단보도 인식)를 고려하여 차량이 자체적으로 속도를 제어한다.
이 프로젝트는 연출된 상황이지만 실생활에서 발생할 수 있는 요소들로 이루어져 있기 때문에, 조원들은 자율주행 소프트웨어의 기초적인 시스템설계 과정를 경험해볼 수 있다. 또한, 개발과정에서 센서들과 액츄에이터를 제어하며 임베디드시스템 프로그래밍분야에 흥미를 이끌어 낼 수 있을 것이다.

이에 따라 조원들이 각자 기능을 하나씩 맡아서 개발하는 것으로 이번 프로젝트를 진행하였다.

1. 프로젝트 개요

1.1 개요

‘창업연계공학설계입문’ 과목의 일환으로 진행된 AD Project의 목표는 임베디드시스템 프로그래밍 기초 센서, 액츄에이터 등을 제어하는 것, 그리고 자율주행 알고리즘을 습득하고 설계하는 것이다.

Xytron의 Xycar를 이용하여 자율주행차량을 만들었다. 실도로에서 필요하게 될 다양한 기능들은 구현함과 함께 충돌없이 모든 기능을 완수하기를 목표로 했다.

전체 프로젝트는 다음과 같은 프로세스로 진행하도록 연출했다. 직선과 곡선으로 구성된 도로위에서 Xycar가 주행하도록 하는 것을 기본으로 하며, 주행하는 도로 중간중간에 횡단보도 구간과 신호등 구간 주차 구간을 만들었다. 횡단보도 구간에서는 횡단보도와 횡단보도를 건너는 보행자를 인지하여 충돌을 회피하도록 했고, 신호등 구간에서는 신호등을 인지하여 멈춤과 진행을 하도록 했으며, 주차 구간에서는 주차공간에 평행주차했다.

1.2 프로젝트 추진 배경

프로젝트 주제는 이 과목에서 배운 내용을 바탕으로 시간적, HW적, 역량적인 제약들과 같은 현실적인 제약이 따르는 상황에서의 시스템 설계에 익숙해지고, 도전적인 목표 달성할 수 있어야 했다. 또한 실제 자율주행차량이 공도를 주행할 때 맞닥들이게 될 중요한 상황들을 가정해야 했다. 그래서 프로젝트 주제로 ‘퇴근길 안전하게 귀가하기’를 정했고 이행하게 되었다.

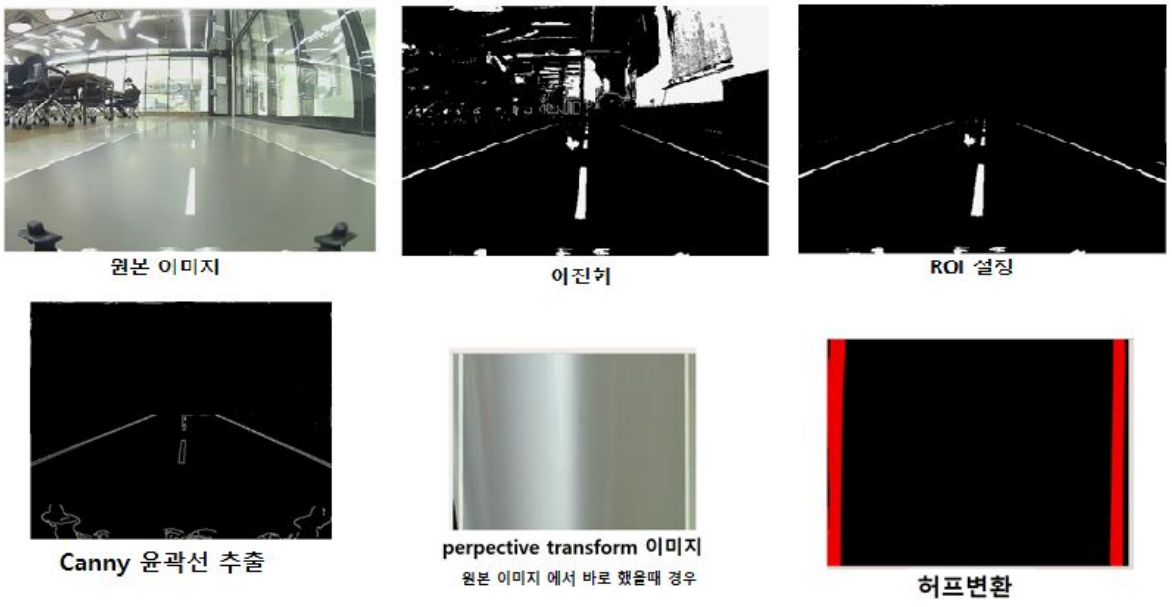
‘퇴근길 안전하게 귀가하기’는 퇴근후 집까지 자율 주행차량을 타고 갈 때 자율주행차량이 맞닥들이게 될 중요한 상황들을 가정했다. 먼저 차량이 차선을 인식하고 차선을 따라 가는 것이 가장 중요하다. 더불어서 신호등을 인지할 줄 알아야 도로교통법규를 준수하며 주행을 할 수 있다. 또한 보행자와의 접촉사고를 막기 위해 횡단보도 앞에서는 차량의 속도를 줄여야 해야하며 보행자가 지나갈 경우 멈춰서서 보행자가 지나갈 때까지 기다려야 할 것이다. 또한 주차공간에 자동으로 주차할 수 있다면 운전자의 수고를 많이 덜어줄 수 있을 것이다.

2. 주요기능

<기능 구현방법만 있고 설명없는 것 같아서 씀 맘에안들면 지워도댐>
'퇴근하고 안전하게 집가기' 프로젝트의 주요기능은 직선/곡선 주행, 사람인식, 횡단보도인식, 신호등 인식, 주차이다.
도로주행을 하면서 횡단보도를 감지했을 때 차량의 속도가 느려지고, 사람이 횡단보도에 위치할 때 차량을 멈춘다. 또한 신호등의 빨간불을 감지하면 차량을 멈춘다. 마찬가지로 초록불을 감지했을 때 차량은 다시 정상적으로 주행한다. 마지막으로 차량은 자동 평행주차를 하면서 주행 상황을 마무리 한다. 각각의 기능 구현 방법은 아래와 같다.

2.1. 직선주행, 곡선주행

기존 중간평가, 유레카 프로젝트때 사용했던 방법을 그대로 사용한다.



< 각 전처리 단계 >

허프 변환을하면 그에대한 theta값이 나오고 좌회전, 직진, 우회전 각각 각도값이 다르게 나오기 때문에 자회전 우회전 직진에대한 제어를 할수있다.

2.2. 사람 인식

사람 인식 과정

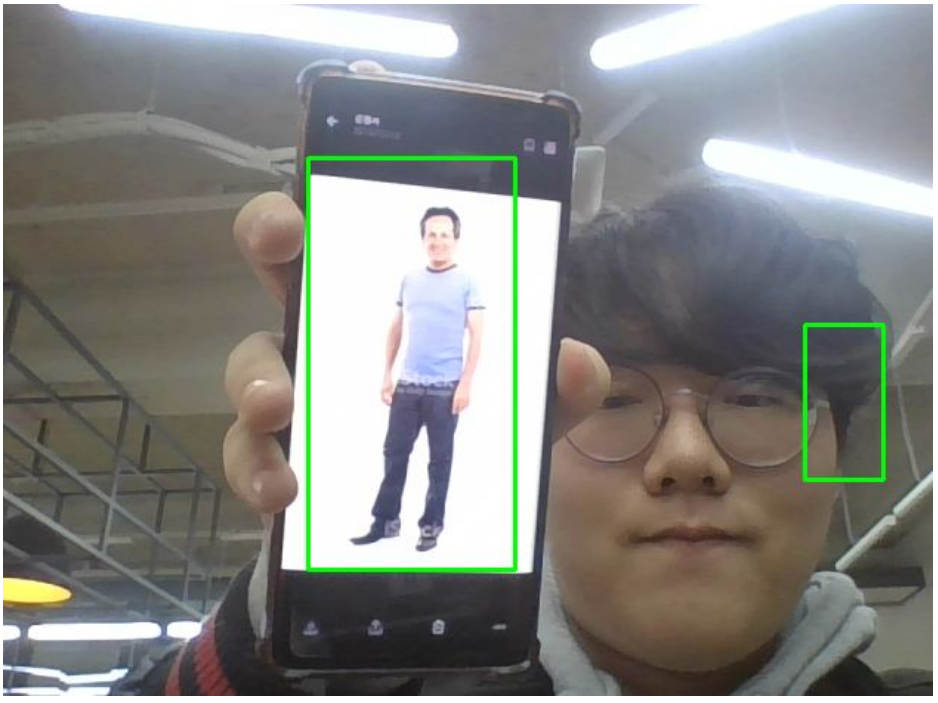
1. usbcam 에서 받아온 이미지를 자기고 class_classifier 를 통해 전신을 추출한다.
2. 추출한 전신을 전처리를 통해 신뢰되는 전심 사람값만 거른다.
3. 그뒤 그 사람 정보를 가지고 초음파 센서와 결합한다.
4. 그뒤 사람이 검출됫고 and 초음파 센서값이 일정 거리 미만이게 되면 사람을 인식한거라 판단
5. 차를 멈춘다.

```
face_cascade = cv2.CascadeClassifier('./haarcascade_fullbody.xml')
```

처음 전신 사람을 검출하기위해 opencv에서 제공하는 haarcascade 트레이닝 데이터를 읽고 객체를 생성 한다.

```
faces = face_cascade.detectMultiScale(frame, 1.03, 5)
if len(faces) > 0 and usonic_data[1] < 100:
    print('People!')
    isPeople = True
```

전신 사람 검출을 하기위해 생성 했던 CascadeClassifier 객체에다가 xycar 의 카메라 이미지 정보를 넣어 사람전신을 검출한다.
만약 검출이 됐고 초음파 센서로부터 받은 정보가 만약 1m 이하라면 True를 리턴해서 메인루프에서 차를 멈추는 로직을 실행할수 있도록 한다.



< 사람 인식을 제대로 하지만 그외 다른 이상한 것도 검출이 되는것을 볼수있다. >
< 그래서 초음파 센서를 추가해서 신뢰도 높은 데이터를 생성 >

2.3. 횡단보도 인식

횡단보도 인식과정은 5단계로 진행된다.

1. usbcam에서 받아온 이미지를 이미지의 원근감을 왜곡시켜 Top view로 변환시킨다.

2. 이미지에서 contour를 찾는다.
3. contour에서 다각형(polygon)을 검출한다.
4. 검출한 다각형들의 꼭지점 개수를 확인하여 사각형을 찾는다.
5. 현재 검출되는 사각형의 개수가 5개가 넘는다면 횡단보도로 판단한다.

우선, xycar usbcam에서 받아오는 이미지는 사다리꼴의 형태로 되어있기 때문에 도로위에 놓인 횡단보도 또한 뚜렷한 직사각형으로 인식할 수 없게 된다. 따라서 이미지를 Top view로 변환시켜서 횡단보도의 흰색 직사각형을 정확히 인식할 수 있게 해야한다.
이미지의 원근감을 왜곡시키는 과정은 아래 작성된 내용과 같다.

```
pts1 = np.float32([[210, 248], [400, 248], [20, 314], [600, 314]])
pts2 = np.float32([[[0, 0], [320, 0], [0, 240], [320, 240]]])
M = cv2.getPerspectiveTransform(pts1, pts2)
birdeye = cv2.warpPerspective(frame, M, (320, 240))
```

np.float32로 왜곡시킬 부분의 꼭지점좌표들을 pts1에 저장한다. 그리고 변환한 이미지의 꼭지점 좌표들을 pts2에 저장한다.
getPerspectiveTransform() 함수는 좌표 기반으로 픽셀을 늘리거나 줄이는 기능을 하는 함수이다. 픽셀 이동 매트릭스 M를 저장한다.
warpPerspective() 함수로 변환된 이미지를 변수에 저장한다. 함수에 들어가는 인자값은 순서대로 각각 변환할 이미지 객체, 픽셀이동 매트릭스 M, 변환될 이미지의 크기이다.

이렇게 얻어온 Top view형식 이미지에서 윤곽선을 추출한다.

```
_, contours, hierarchy = cv2.findContours(img_binary, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

findContours함수를 사용하여 contour들을 찾아냈다.

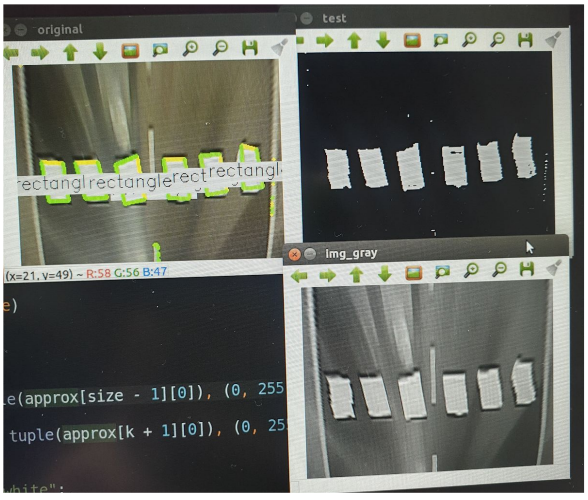
발견한 contours의 길이만큼 반복문을 실행하며 다각형을 찾아낸다. 그리고 다각형들 중 4개의 꼭짓점을 가지는 것, 윤곽선의 내부가 흰색으로 이루어져 있는 것 두가지의 조건을 만족할 때 사각형이라고 인식한다. 그리고 횡단보도, 즉 사각형의 개수가 6개 이상일 때 횡단보도 임을 인식한다.

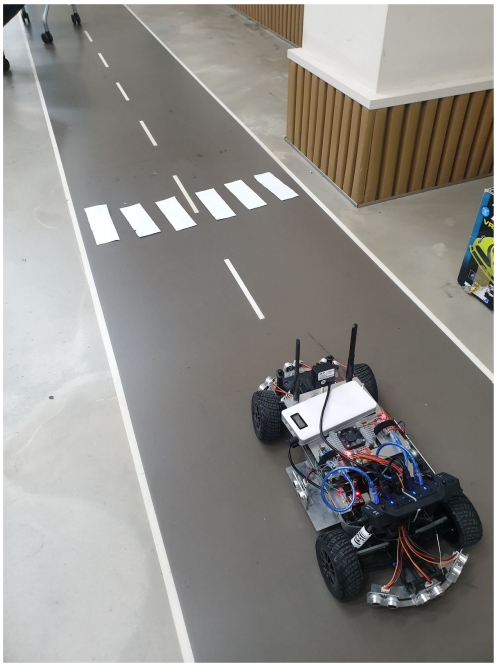
```
reccnt = 0
    for cnt in contours:
        epsilon = 0.04 * cv2.arcLength(cnt, True)
        approx = cv2.approxPolyDP(cnt, epsilon, True)

        size = len(approx)

        cv2.line(birdeye, tuple(approx[0][0]),
                          tuple(approx[size - 1][0]), (0, 255, 255), 3)
        for k in range(size - 1):
            cv2.line(birdeye, tuple(approx[k][0]),
                              tuple(approx[k + 1][0]), (0, 255, 100 + k * 10), 3)

        if cv2.isContourConvex(approx):
            if size == 4 and label(test, cnt) == "white":
                reccnt += 1
            else:
                pass

        if (reccnt >= 6):
            print("shape {} cnt    right".format(reccnt))
        else:
            print("{} none".format(reccnt))
```

approxPolyDP() 함수를 사용하여 주어진 cnt (contours가 가지고 있는 직선)을 epsilon값에 따라 꼭지점 수를 줄여 새로운 곡선을 생성하여 approx에 리턴한다. 여기서 epsilon값이 커지면 커질 수록 꼭지점이 줄어든다. epsilon값 조정을 통해 이미지 해상도에 따른 사각형 검출정도 조정도 할 수 있다.

실제 주행상황 연습에서는 5개의 기능을 모두 합쳐 실행해야 하기때문에 횡단보도 검출 함수를 호출하면 횡단보도를 인식했을 때 True를 리턴하고 인식하지 못했을 때 False를 리턴하여 리턴값에 따라 모터를 제어한다. True일 때 주행속도를 낮게하고, False일 때 주행속도를 원래의 값으로 유지한다.

아래의 사진은 횡단보도 기능만 테스트했을 때의 차량의 모습과 횡단보도 인식 화면이다. 흰색바탕의 사각형이 6개이상이므로 정상적으로 횡단보도라고 인식한다.





2.4. 신호등 인식

1 요구 사항

- 컴퓨터로 영상을 입력받는다.
- 영상에 나올 신호등의 빨강색, 초록색 범위, 그리고 신호등의 원의 크기를 설정한다.
- 원을 검출한다.
- 원 중에 빨강색, 초록색을 검출한다.
- 주행하는 중 빨강색 원을 인식하게될 시 정지, 초록색 원을 인식할 시 다시 주행하도록 한다.

2 구현

lower = {'red': (-10, 30, 30), 'green': (50, 30, 30)} upper = {'red': (10, 255, 255), 'green': (70, 255, 255)} colors = {'red': (0, 140, 255), 'green': (0, 255, 0)}	빨강색과 초록색의 범위를 조정한다. 빨강과 초록의 범위가 빛의 세기에 따라 많이 달라져서 lower과 upper이라는 최대, 최소 범위를 두어서 빨강과 초록의 범위를 자율주행스튜디오의 빛의 밝기에 맞추었다.

while not rospy.is_shutdown():
#cv2.imshow("camera", cv_image)
frame = cv_image
key_num = cv2.waitKey(1)
if key_num == 27:
break
elif key_num == 32:
while True:
if cv2.waitKey(1) & 0xFF == 32:
break
img_color = frame
--- 	
cv2.imshow('img_color', img_color)
	ros가 실행되는 동안 영상 정보를 입력받게 하고 img_color에 저장한다. 이 후원활하게 실행되는지 확인하는 창(img_color)을 만든다.

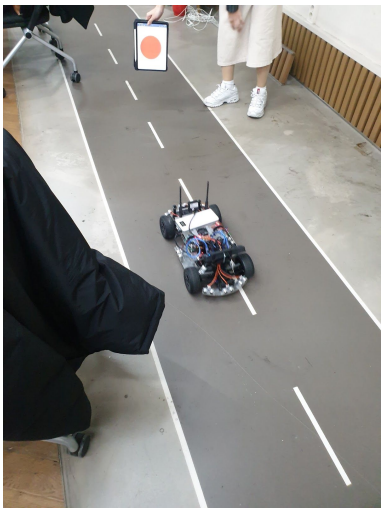
img_blurred = cv2.GaussianBlur(img_color, (7, 7), 0)	GaussianBlur를 통해 노이즈를 제거하면서 영상을 더 정확하고 부드럽게 받아 처리한다.
img_hsv = cv2.cvtColor(img_blurred, cv2.COLOR_BGR2HSV)	이미지가 정확히 검출되었는지 확인하도록 이진화를 행하기 위해 미리 블러처리한 영상을 hsv로 전환한다.
for key, value in upper.items():	hsv한 영상을 기반으로

kernel = np.ones((9, 9), np.uint8) mask = cv2.inRange(img_hsv, lower[key], upper[key]) mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel) mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel) ---------- cv2.imshow('img_mask', mask)	mask를 적용해 이진화하여 원이 검출된 부분을 화면에 하얗게 나오게 한다. 이 후 이것을 확인할 창(img_mask)을 만든다. 모폴로지 필터링(영상 분석과 처리를 위한 함수, 특정한 형태를 띠는 필터를 만들고 이 필터를 영상에 씌울때 쓰는 기법, 현재 픽셀을 기준하여 필터를 적용해 필터 영역안의 값을 확인 후 현재 픽셀의 값을 수정함)을 통해 영상의 노이즈를 조절한다.

if radius > 10:
cv2.circle(img_color, (int(x), int(y)), int(radius), colors[key], 2)
cv2.putText(img_color, key + " ball", (int(x - radius), int(y - radius)),
cv2.FONT_HERSHEY_SIMPLEX, 0.6,
colors[key], 2)
print(key + " ball")
if key == 'red':
speed = 90
elif key == 'green':
speed = 130
	만약 영상으로 입력받은 신호등의 원의 반지름이 10보다 클 때 (신호등의 원 크기) 그 신호등의 원 위에 원이 그려지고 ball이라는 문구가 표시될 수 있도록 한다.(원이 검출되는지 확인하기 위해)
	그리고 key값이 빨강으로 인식되면 red ball이라는 문구가 뜨는 동시에 정지하도록 하고 초록으로 인식되면 green ball이라는 문구와 함께 다시 주행하도록 한다.

그냥 단순히 빨강과 초록의 범위를 잡아 검출해내면 빛의 밝기, 주변 환경 등의 영향을 많이 받아 값이 달라지기 때문에 많은 필터링을 합쳐 코딩했다.

<신호등 설치>



<빨강 원 인식 및 정지>

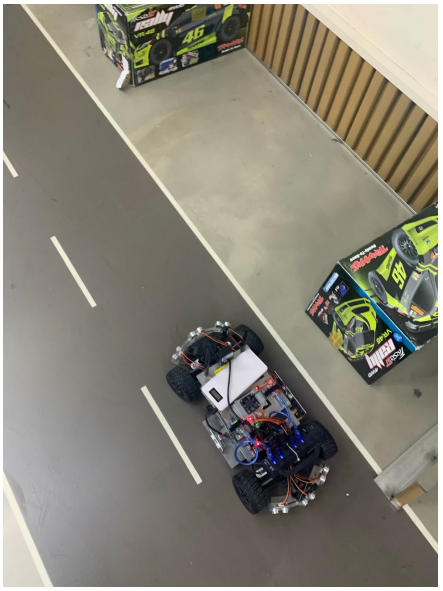
<초록 원 인식 및 직진>

2.5. 주차

다양한 주차방법을 시도하던 중 현 자율주행스튜디오의 트랙 위에서 구현하기 적합한 평행주차를 구현하기로 결정했다. 초음파 센서를 이용한 주차 과정은 다음과 같다.

1.주차 영역 감지때까지 전진
박스를 초음파센서로 감지한다. 우측 초음파센서가 이전보다 10 이상 가까워 지면 2.5초 동안 전진 한다. 6번 초음파센서가 이전보다 10이상 가까워지면 2.5초동안 [130,50]으로 꺾으면서 후진을 한다

```
def first_turn():
    global ultrasonic_data
    previous = 0
    temp = ultrasonic_data[6] - previous
    print(ultrasonic_data[6])
    while temp > -10:
        go_forward_slowly()
        temp = ultrasonic_data[6] - previous
        previous = ultrasonic_data[6]
    duration = 2.5
    start = time.time()
    end = start + duration
    max_angle, min_angle = [130, 50]
    while(time.time() < end):
        drive(35,115)
    stop_the_car()
```

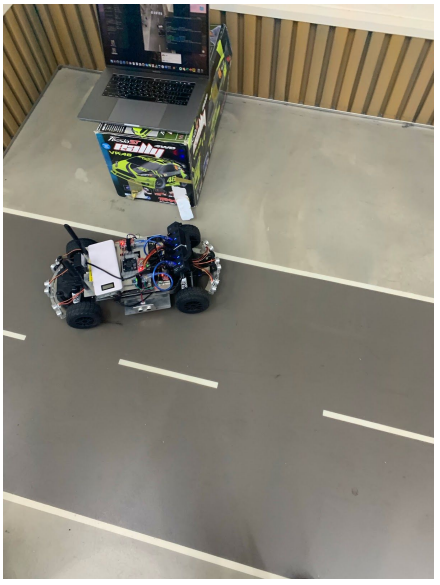


2. 단순 후진
후방 4번 초음파센서가 60보다 클 때 까지 [90,65]으로 후진을 한다.
60보다 같거나 작아지면 정지한다.

```
def change_the_gear():
    for stop_cnt in range(2):
        drive(90,90)
        time.sleep(0.1)
        drive(90,60)

def back_the_car_slowly(angle= 90):
    drive(angle, 65)

def stop_the_car():
    drive(90,90)

while distance(4) > 60:
        back_the_car_slowly()
```

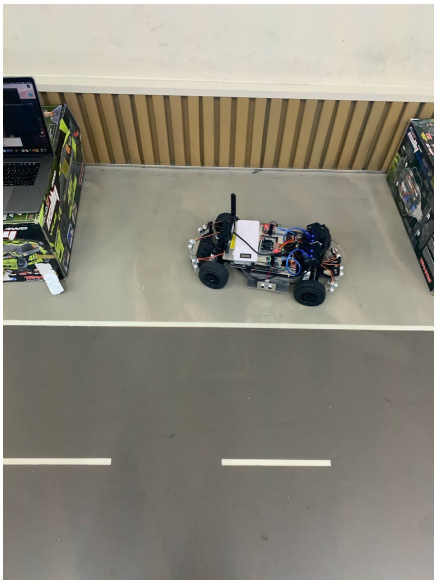
3. 꺾으면서 후진
90도에서 50도 까지 ([90,65] ~ [50,65]) 시간이 지남에 따라 부드럽게 5초동안 후진을 한다.
후방 4번 초음파 센서가 10보다 작으면 끝난다.

```
def soft_drive_in():
    duration = 5
    start = time.time()
    end = start + duration
    max_angle, min_angle = [130, 50]
    while(time.time() < end):
        x = (time.time() - start) / duration
        y = soft_steering(x)
        angle = 90 - (40*y)
        back_the_car_slowly(angle)
        if distance(4) < 10:
            break
```



4. 교정
2번 초음파 센서와 5번 초음파센서의 관측된 거리가 차이가 별로 없을 때(자동차가 거의 평행 상태)까지 자동차를 여러 방향으로 움직인다.

```
def calibrate():
    while (abs(distance(2) - distance(5)) > 3):
        if distance(1) < 15:
                change_the_gear()
            print("calibrate-go-back")
            while distance(4) < 10:
                drive(90,70)
        elif distance(2) > distance(5):
            drive(120,110)
        else:
            drive(60,110)
```



이 과정들을 통하여 평행주차를 초음파센서를 통하여 구현할 수 있었다.

3. 프로젝트 결과

5개의 기능을 모두 합쳐 연출된 상황에서 실행하였다. '퇴근하고 안전하게 집가기' 컨셉에 맞추어 퇴근한 직장인의 차량이 도로를 주행하다가 마주칠 수 있는 여러가지 상황을 재연하였다.

도로를 주행하기 시작하면 가장 먼저 횡단보도를 마주친다. 이 때 횡단보도의 약 1.5m앞에서 횡단보도를 인식하고 주행속도를 줄인다. 주행속도를 줄인 상태에서 횡단보도위에 사람이 있다는 것을 인식하면 차량이 멈추고, 사람이 지나가면 다시 주행을 시작한다.

그 다음, 주행 중 신호등의 빨간불을 인식했을 때 주행을 멈추고 초록불을 인식하면 원래의 속도로 주행한다. 마지막으로 드디어 차량이 퇴근한 직장인의 집에 도달했다고 가정하고, 주차장에 차량을 주차하면 주행이 끝나게 된다.

이렇게 '퇴근하고 안전하게 집가기' 프로젝트를 완성하였다.

이번 AD 프로젝트를 코딩하면서 조원들 모두 임베디드 시스템의 실제 데이터 수집, 가공 및 분석 등등을 이해하고 자율주행 소프트웨어 개발의 기초를 배울 수 있었다. 앞으로 만약 자율주행 소프트웨어를 개발할 기회를 또 갖게된다면, 이번 프로젝트 경험이 도움이 될 수 있을 것이다.

4. 수행 후기 (5줄 뽈마2 정도로 써주세요..?)

강현서	기존 주행평가나 유레카 프로젝트 같은경우 수행해야되는 것이 한 가지 밖에없어 역할분배가 힘들었는데 이번 창연공 AD프로젝트를 했을때에는 각 기능마다 역할분배를 할수있어 내가 맡은 사람인식,제어 역할에 더욱 몰두할수 있고 완성도 높은 결과를 얻을수 있어 더욱 공부가 되었던 것 같다. 하지만 이것역시 완전하게 정확하지 않고 일부 오차를 보이고 있어 더욱 공부가 필요할것 같다.
권순호	기존 수업 시간에서 자율주행스튜디오의 트랙을 한바퀴를 도는 자율주행에 다양한 기능을 추가할 수 있는 시간을 가지는 AD프로젝트였다. 단순 주행뿐만 아니라 다양한 일상생활에서의 자율주행 자동차에 대하여 고민해보는 시간을 가질 수 있었다. 또한, 차선인식 알고리즘과 다양한 주제에 맞춰 새로운 알고리즘과 기능들을 구현할 수 있어서 많은 도움이 되는 시간이었다. 하지만 기간이 많이 짧아서 많은 연구들을 할 수 없었던 점이 아쉬움으로 남았다.
권은우	과목 자체가 너무 어려웠고 힘들어서 포기하고 싶었지만 조원들이 다같이 열심히 하고 서로 도와주었기에 좋은 성과를 이뤘고 새로운 영역에 대한 지식을 많이 함양하게 된 것 같아 좋은 경험이었다. 신호등 기능을 맡아 구현하였는데 빨강색은 빛 때문에 주황색으로, 초록색은 흰색으로 인식하며, 원을 인식할 땐 멀리 있는 빛의 모양을 원으로 인식하는 등의 문제가 생겨 범위를 조정하고 필터링을 해내는데 많은 시도가 들어갔다. 여러 구글

	라이브러리를 계속 검색해서 필터링을 해내는 데 성공했을 때 성취감이 컸고 계속해서 이 과목에 적응하게 도와준 조원들이 너무 고마웠다.
김서경	이번 ad 프로젝트에서 횡단보도 기능을 구현하였다. 조원들의 도움을 받지않고 기능을 구현한 것에 성취감이 크다. 또한, 우리가 직접 정한 주제이기 때문에 이전의 과제들보다 더욱 흥미를 갖고 조사하며 개발할 수 있었다. 평소 프로그래밍을 할 때 opencv 라이브러리를 사용할 일이 별로없는데, 이번 프로젝트를 진행하며 opencv를 다양하게 사용해 보아서 좋은 경험이 되었다. 다음에 기회가 생긴다면 더 좋은 하드웨어를 가지고 개발해보고 싶다.
장진형	수업시간에 배우기만하는 것으로 끝나지 않고 스스로 어드밴스드해볼 수 있는 AD 프로젝트를 진행을 했더니 확실히 더 기억에 남는 것 같다. 아무래도 큰 프로젝트이다보니 팀원끼리 업무를 나눠 진행하고 마지막에 합치는 방식으로 진행했던 것이 인상깊었다. 불평없이 끝까지 함께해준 팀원들에게 감사하다.