

2023 캡스톤 디자인 07분만 05조 최종 발표

영상처리를 이용한 차선 자동인식 알고리즘 개발

05조

1723001 김우성

1722427 이예찬

1930394 노이지

1923361 김지은

1922474 박아진

Contents

01

Motive

02

하드웨어구성

03

H/W Architecture

04

S/W Architecture

05

외형 모습

06

데모영상

07

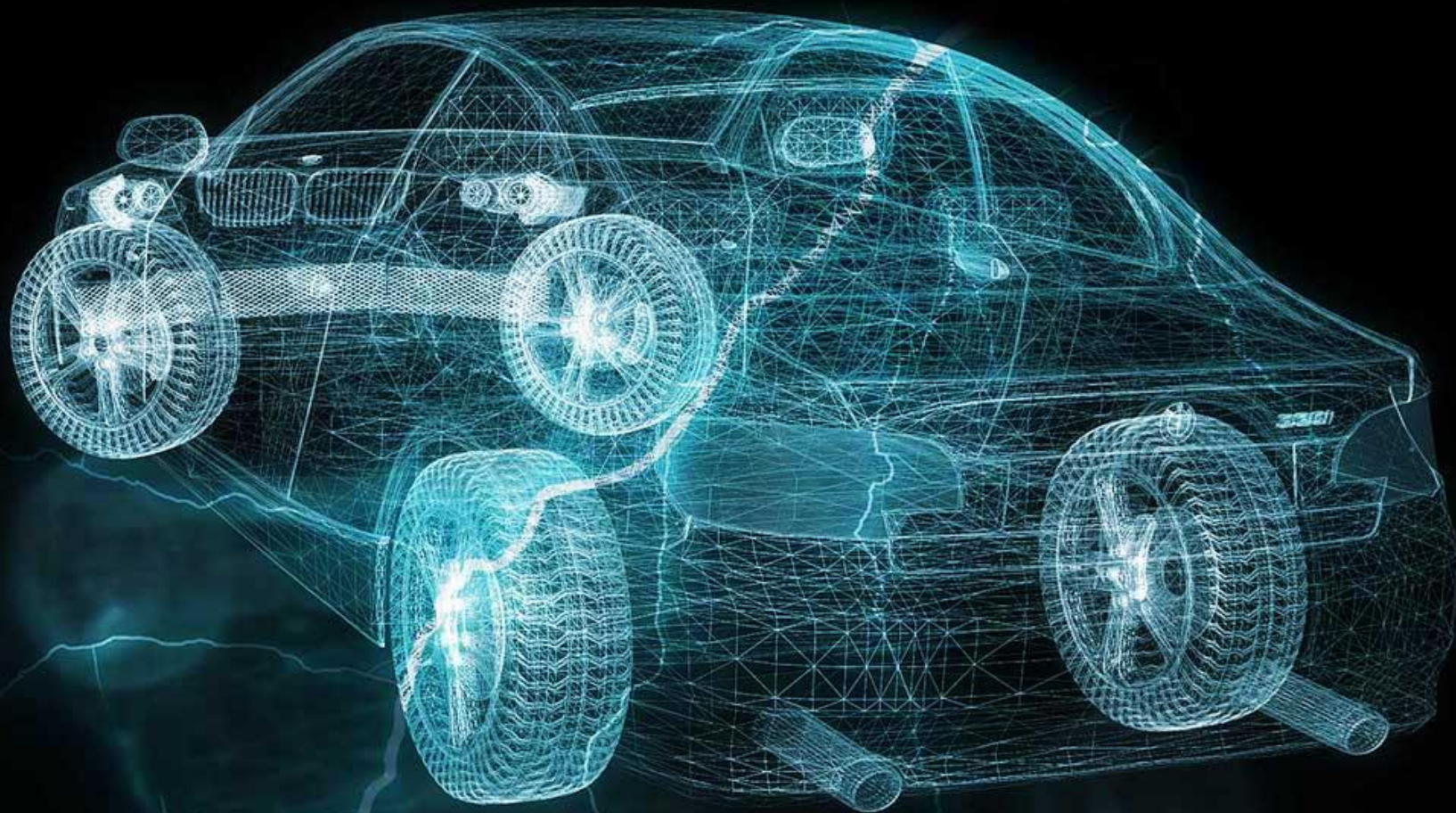
팀원역할

08

Q&A

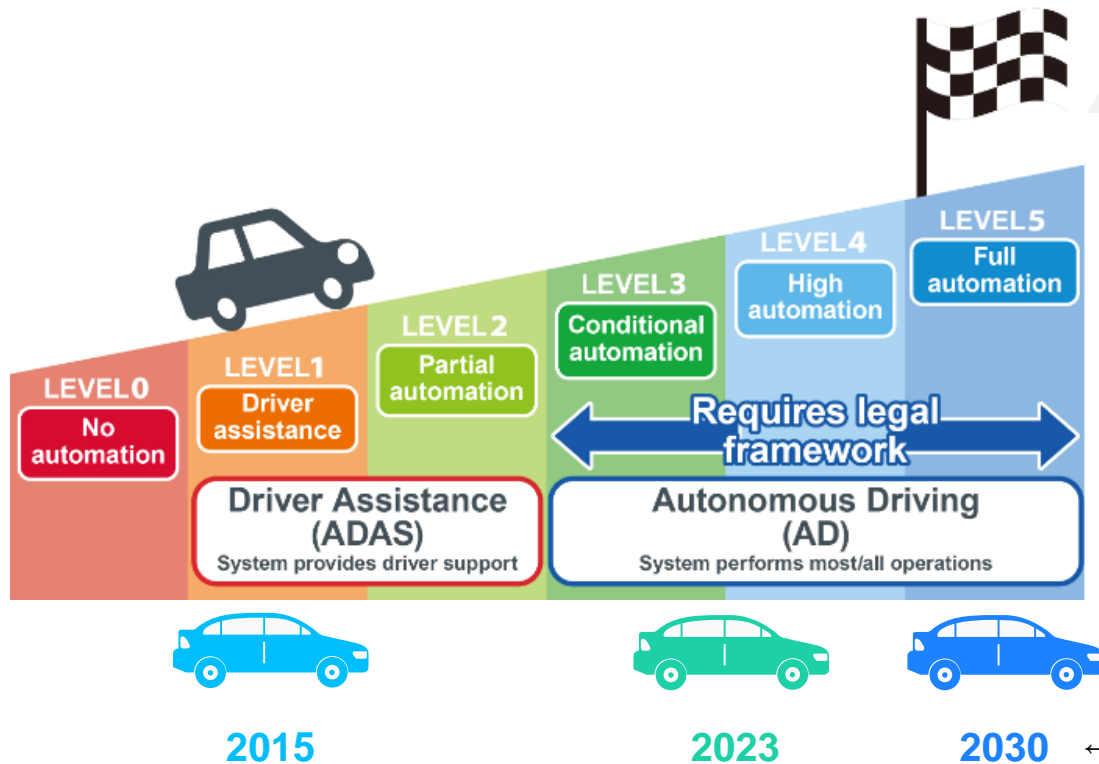


Motive



프로젝트 구상배경

“현재 급격히 발전하는 Automotive Industry, 최종 목표 : Autonomous driving System”

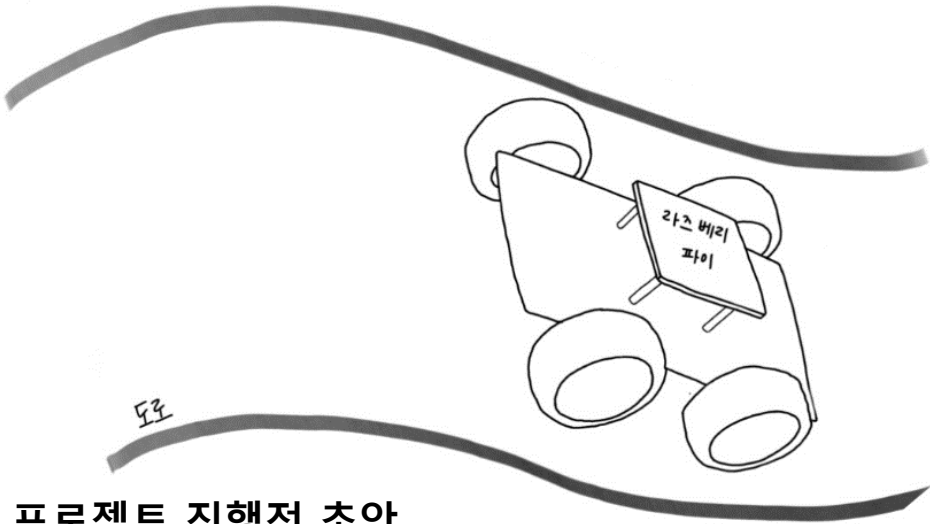


- Autonomous Driving의 제일 기초 기술
- 관련 S/W 기술 및 차량 반도체 수요 증가
- 사고율 감소
- 교통흐름 개선 기대효과

← 해당 연도는 상용화 기준임
이미 LEVEL4 자율주행 기술은 완성단계

프로젝트 구상배경

“차선을 인식하고 주행하며, 차선이탈시 운전자에게 알려주는 시스템 ”



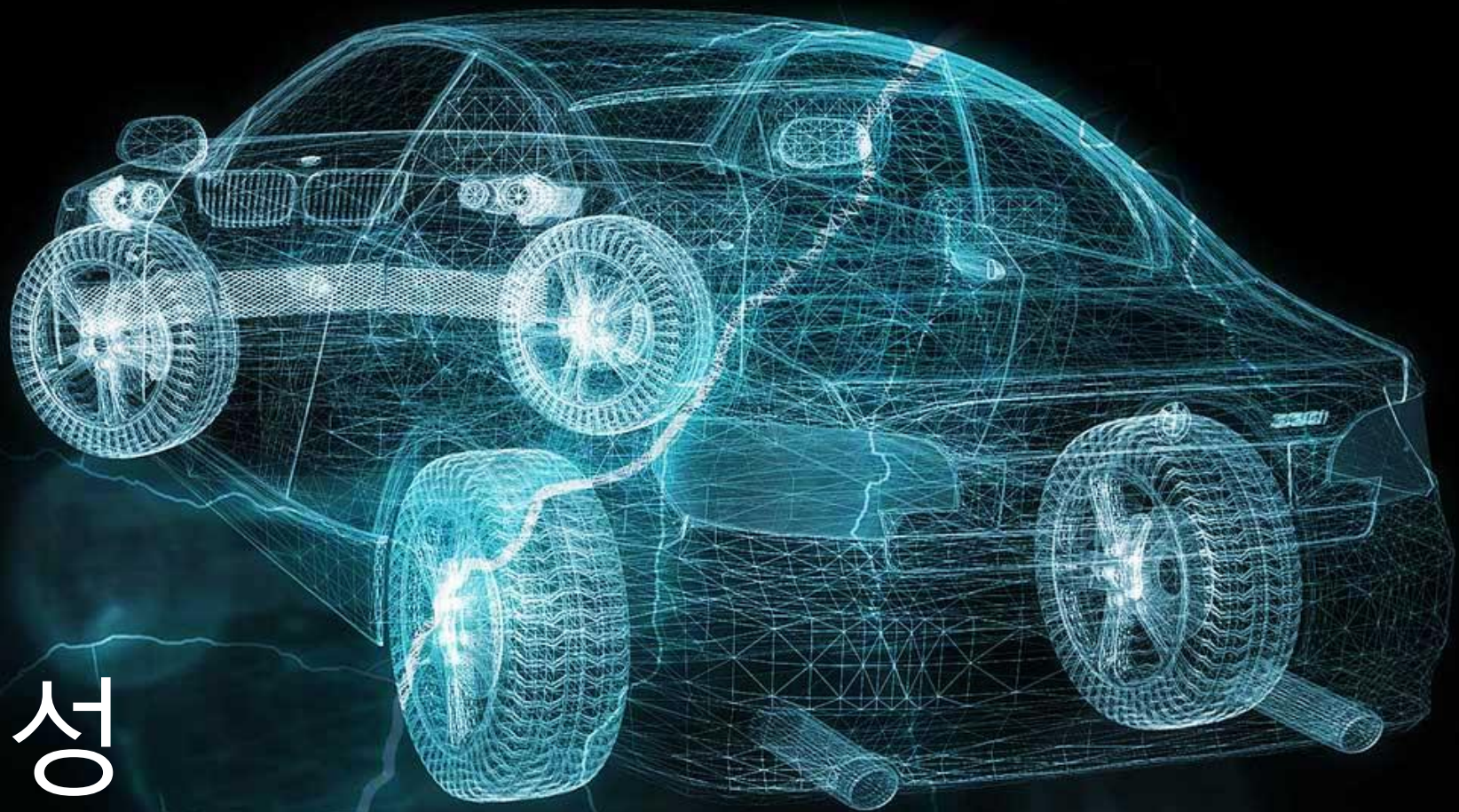
프로젝트 진행전 초안

- 직접 구성한 RC카에 라즈베리파이와 카메라 모듈을 탑재하여 차선인식을 진행하고 모터를 통해 RC카를 제어하고 차선이탈시 운전자에게 알려주는 시스템을 구현하고자함

- 직접 도로를 제작하여 시연을 진행하고 시연에 사용하는 차선 외에도 모든 차선에 동일한 동작이 가능도록 설계함

- 사용자가 직접 무선통신으로 RC카를 조종하고 차선이탈시 실시간으로 BUZZER와 LED등 알림이 오고 방향지시등을 켜고 차선 변경시 알림이 동작하지 않도록 구현

하드웨어구성



하드웨어구성 (차선인식)

라즈베리파이 3B+

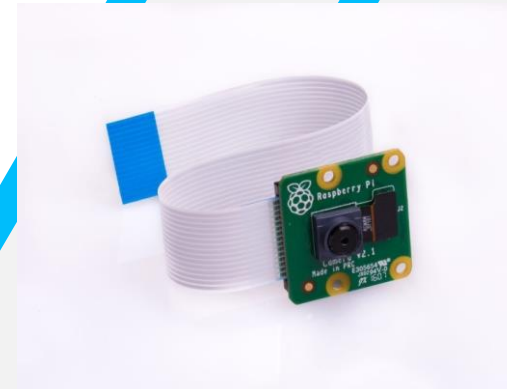
CPU	1.4 GHz	Main Memory	1GB LPDDR2 SDRAM
processor	64-bit quad-core ARMv8 BCM2837B0		
GPU	Broadcom Videocore-IV		
storage	Micro-SD		
Network	Gigabit Ethernet over USB 2.0, 2.4 GHz and 5 GHz 802.11b/g/n/ac Wi-Fi		
Bluetooth	Bluetooth 4.2, Bluetooth Low Energy (BLE)		
Other features	Dual-band 2.4 GHz and 5 GHz wireless LAN, PoE capability, four USB ports, extended 40-pin GPIO header		

The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range. It is also the most power hungry platform released to date and requires a 5.1V 2.5A power source.


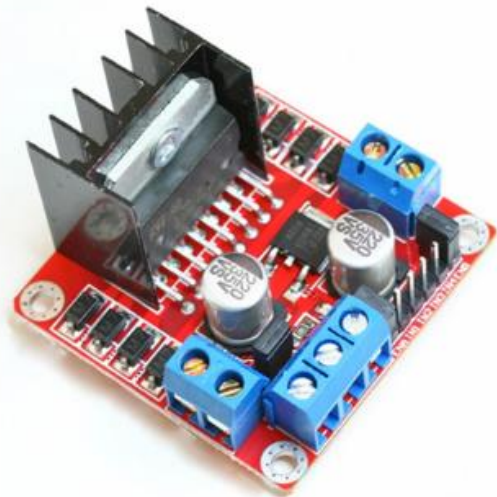



카메라모듈 V2

model	라즈베리파이 카메라모듈 V2, 8MP (RPI 8MP CAMERA BOARD)
Size	25 * 23 * 9 [mm]
Support	1080p30, 720p60, and 640*480p60/90 video record
Lens Size	1/4"
Connection to Raspberry Pi	15-pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2)

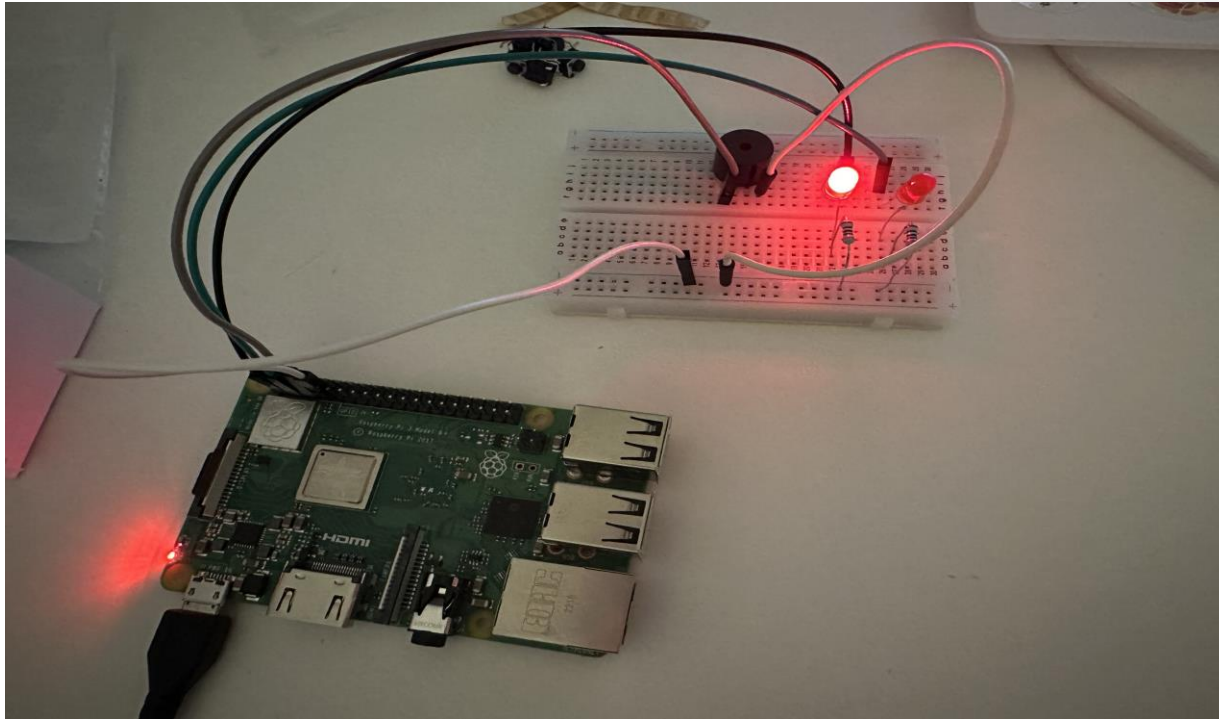


하드웨어구성 (모터제어)

DC 모터	모터드라이버 298N	POWER																						
<table><tr><td>model</td><td>NP01D-48</td></tr><tr><td>Operating Voltage</td><td>3 ~ 9 [V]</td></tr><tr><td>Gear Ratio</td><td>1:48</td></tr><tr><td>MAX SPEED</td><td>110 [rpm]</td></tr><tr><td>MAX torque</td><td>0.027 [N.M]</td></tr></table>	model	NP01D-48	Operating Voltage	3 ~ 9 [V]	Gear Ratio	1:48	MAX SPEED	110 [rpm]	MAX torque	0.027 [N.M]	<table><tr><td>model</td><td>SZH – EK001</td></tr><tr><td>Logical Voltage</td><td>5 [V]</td></tr><tr><td>Drive voltage</td><td>5 ~ 35 [V]</td></tr><tr><td>Logical Current</td><td>0 ~ 36 [mA]</td></tr><tr><td>Drive Current</td><td>2 [A] (MAX single bridge)</td></tr><tr><td>Max power</td><td>25 [W]</td></tr></table>	model	SZH – EK001	Logical Voltage	5 [V]	Drive voltage	5 ~ 35 [V]	Logical Current	0 ~ 36 [mA]	Drive Current	2 [A] (MAX single bridge)	Max power	25 [W]	AA Battery 1.5 * 4ea [V] 6[V]
model	NP01D-48																							
Operating Voltage	3 ~ 9 [V]																							
Gear Ratio	1:48																							
MAX SPEED	110 [rpm]																							
MAX torque	0.027 [N.M]																							
model	SZH – EK001																							
Logical Voltage	5 [V]																							
Drive voltage	5 ~ 35 [V]																							
Logical Current	0 ~ 36 [mA]																							
Drive Current	2 [A] (MAX single bridge)																							
Max power	25 [W]																							
																								

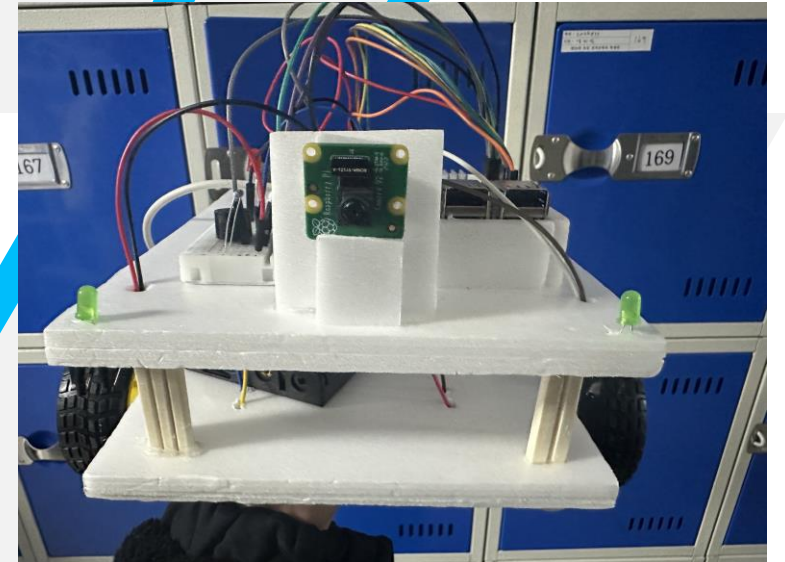
하드웨어구성 (LED, BUZZER알림)

LED



BUZZER

방향지시등 LED



하드웨어구성 (시연용 도로)

테이프
(White, Yellow)

Hard Board



하드웨어구성 (기타)

라즈베리파이 무선전원 보조배터리

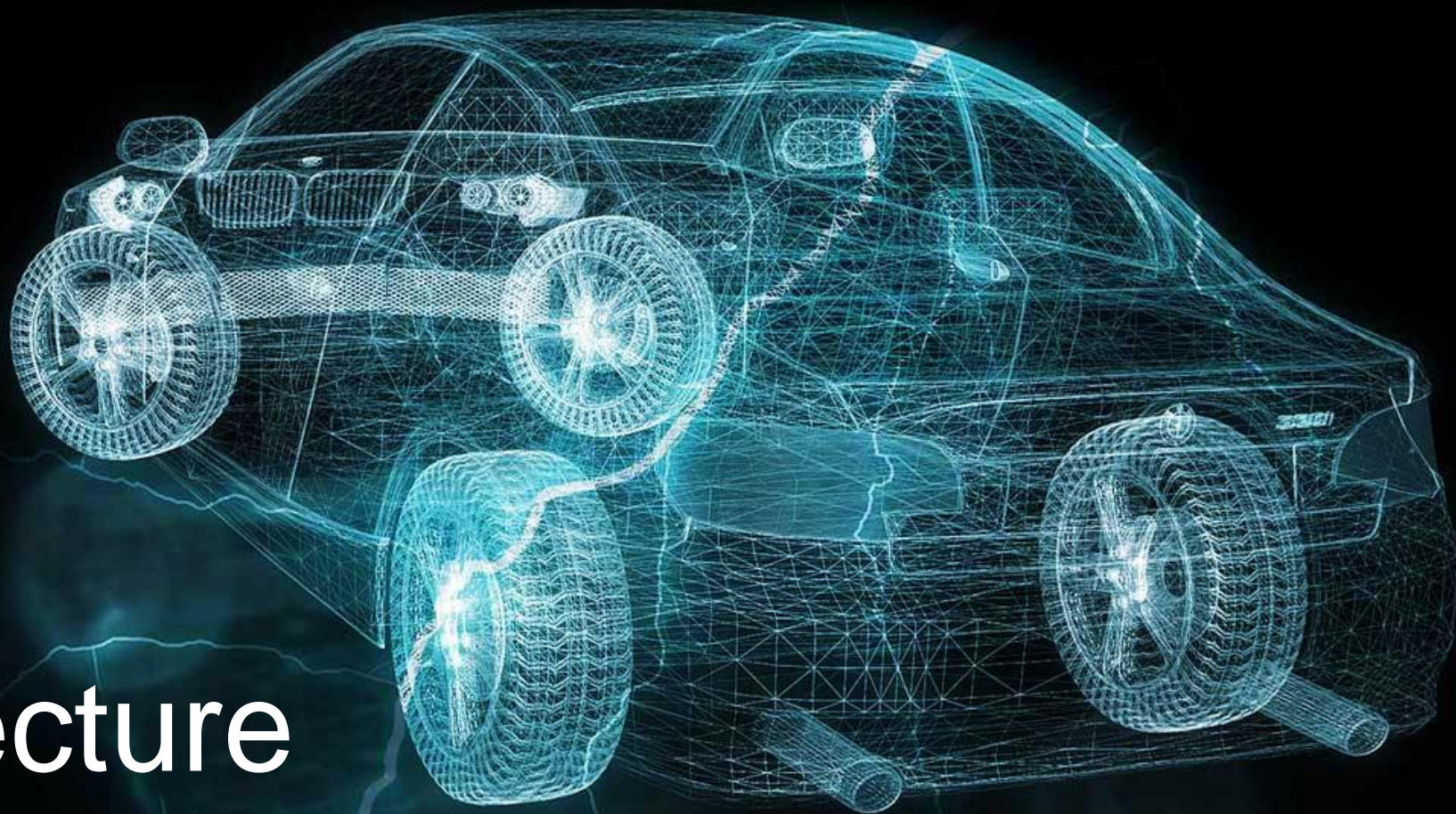
capacity↵	5000 [mAh]↵
INPUT PORT↵	C type↵
OUTPUT↵	USB 2.0[A] [1ea] , USB 3.0[A] [1ea] Total [2ea]↵



RC카 무선제어를 위한 노트북

CPU↵	Intel® Core™@ CPU i5-4560 3.4Ghz↵	Main Memory↵	16.00GB RAM↵
OS↵	Microsoft Windows 11↵		
개발 툴↵	Windows Application↵	Microsoft Visual Studio 2023 (python)↵	
	Real VNC↵	-↵	





H/W Architecture

카메라 모듈



- 카메라 모듈은 다음과 같이 차량의 앞쪽에 부착
- 블랙박스과 유사하게 동작해 차량 앞쪽 영상을 출력
- 출력된 영상에서 차선만을 인식해 프로그램에 활용

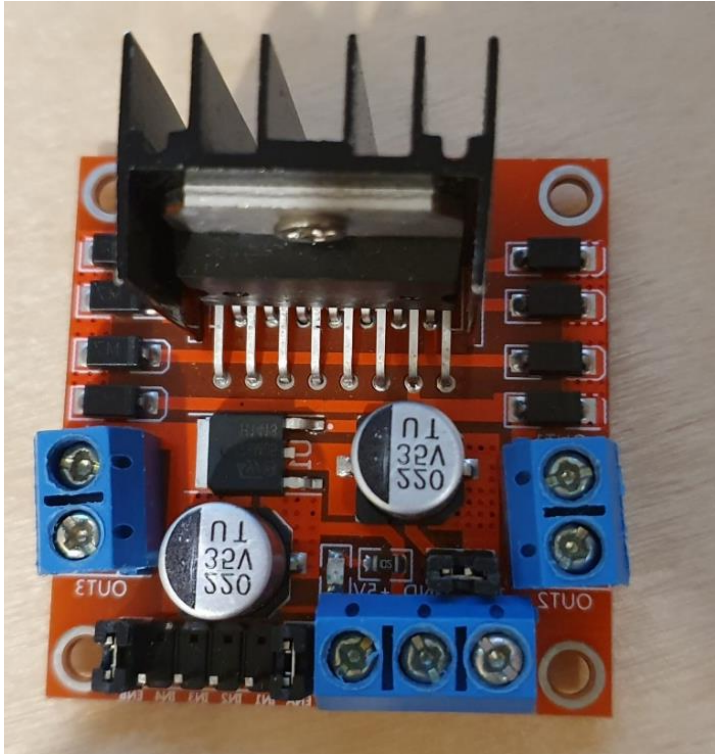
모터



- 전압 세기에 따른 rpm 변화

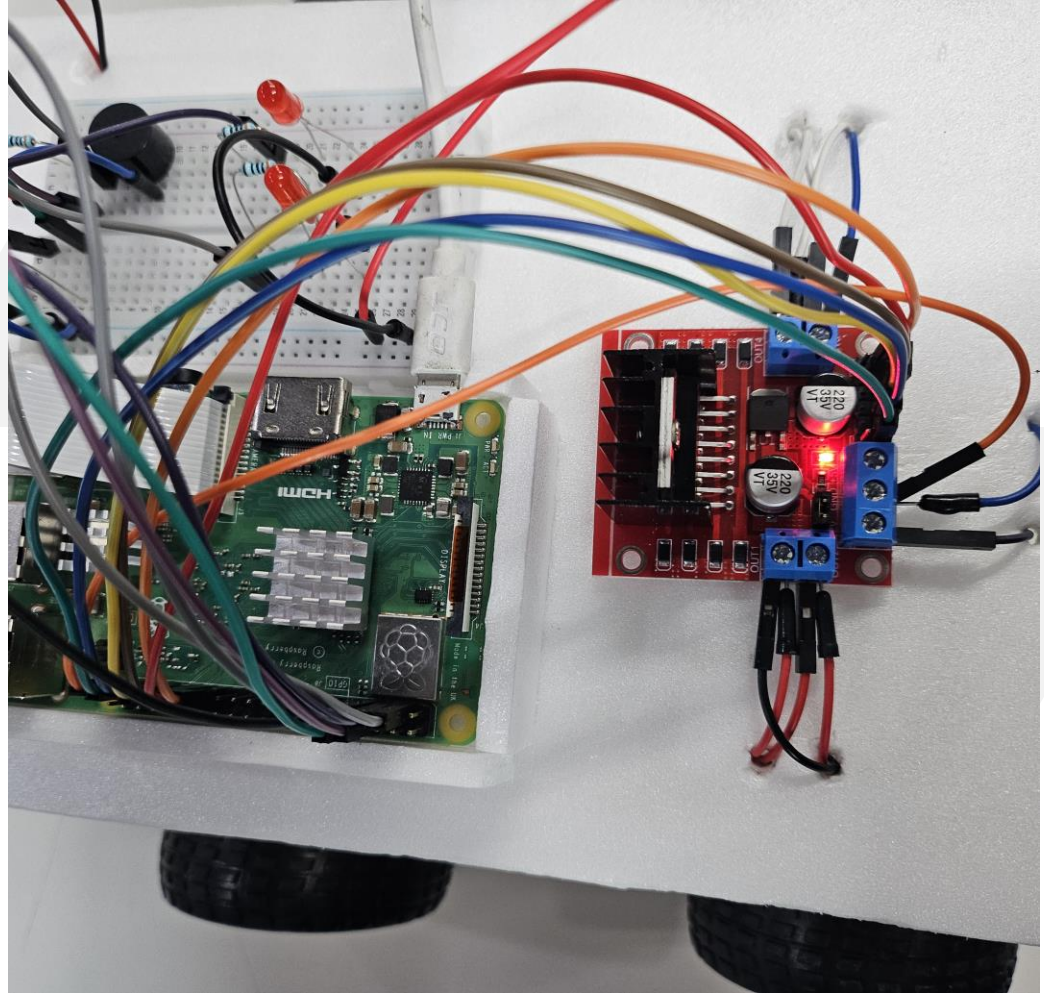
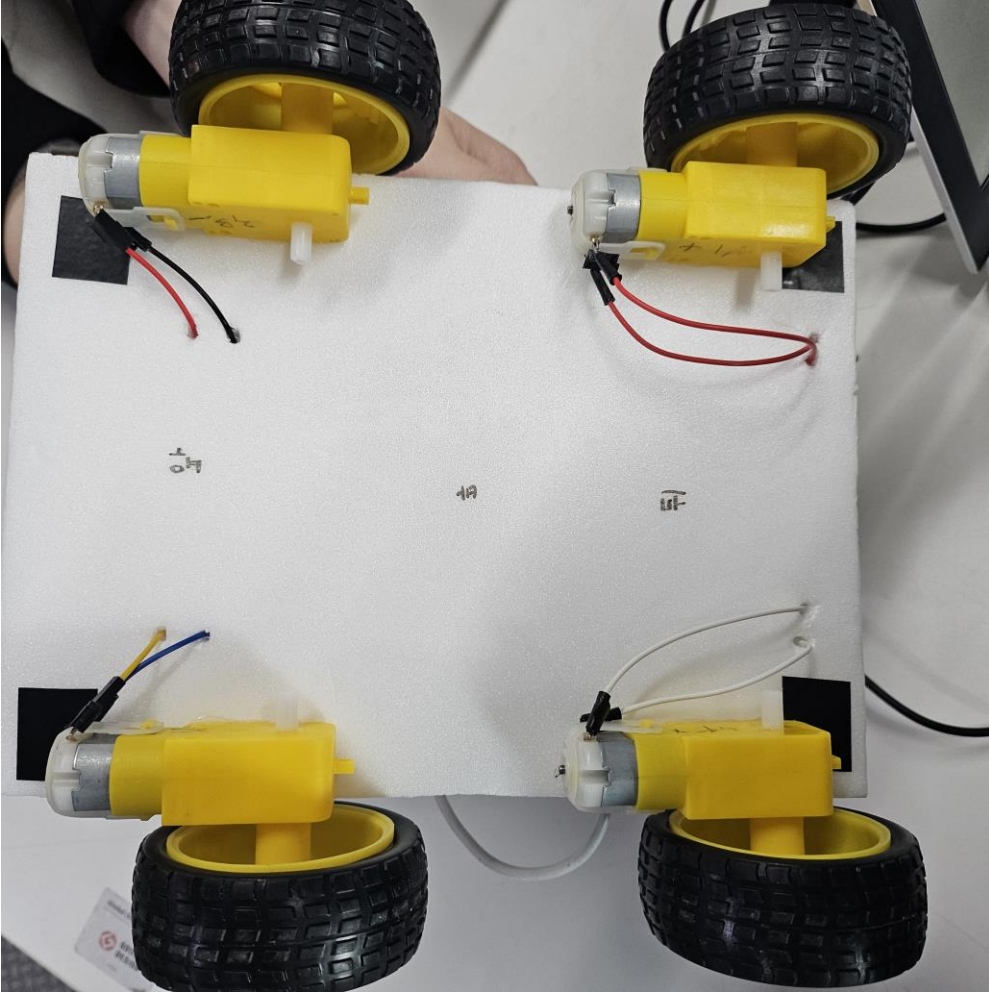
- DC모터 회전각 X 한방향 회전
(역전압 인가시 역방향 회전)

모터드라이브 298N

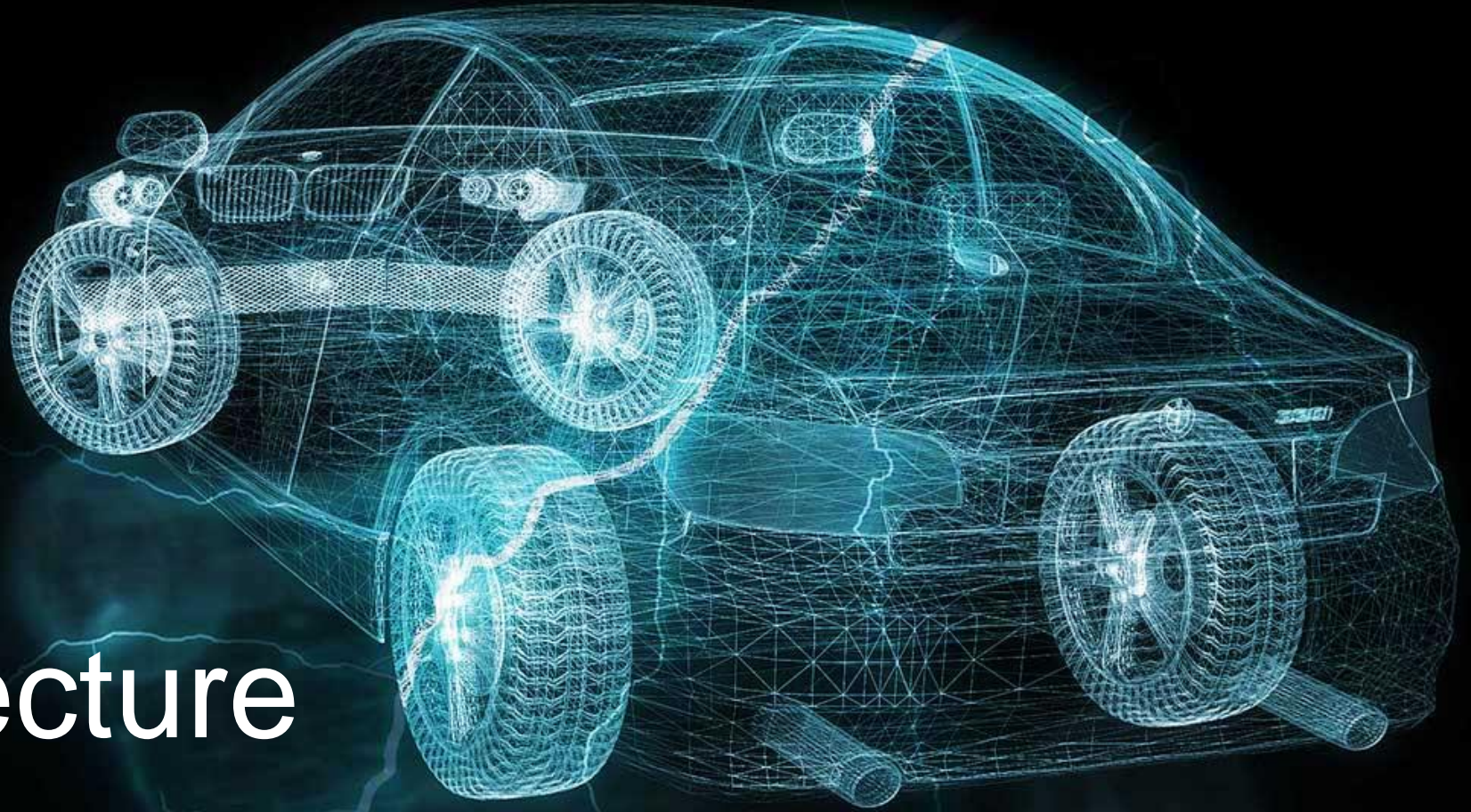


- 모터 제어
- 출력 채널 2채널로 구성
- 전압 분배

모터



S/W Architecture



차선인식 전체 코드

```
# -*- coding: utf-8 -*-
import cv2
import numpy as np
from collections import deque
from time import sleep
import time

def grayscale(img):
    return cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)

def gaussian_blur(img, kernel_size):
    return cv2.GaussianBlur(img, (kernel_size, kernel_size), 0)

def canny(img, low_threshold, high_threshold):
    return cv2.Canny(img, low_threshold, high_threshold)

def roi(img):
    mask = np.zeros_like(img)
    h, w = mask.shape
    vertices = np.array([(w/10, h), (w/10, h*2/5), (w*9/10, h*2/5), (w*9/10, h)], dtype=np.int32)
    cv2.fillPoly(mask, vertices, 255)
    roi_img = cv2.bitwise_and(img, mask)
    return roi_img

def restrict_deg(lines, min_slope, max_slope):
    if lines.ndim == 0:
        return lines, np.array([])
    lines = np.squeeze(lines)
    slope_deg = np.rad2deg(np.arctan2(lines[:, 1] - lines[:, 3], lines[:, 0] - lines[:, 2]))
    lines = lines[np.abs(slope_deg) < max_slope]
    slope_deg = slope_deg[np.abs(slope_deg) < max_slope]
    lines = lines[np.abs(slope_deg) > min_slope]
    slope_deg = slope_deg[np.abs(slope_deg) > min_slope]
    return lines, slope_deg

def separate_line(lines, slope_deg):
    l_lines = lines[slope_deg > 0, :]
    r_lines = lines[slope_deg < 0, :]

    l_lane = average_line(l_lines) if len(l_lines) > 0 else [0, 0, 0, 0]
    r_lane = average_line(r_lines) if len(r_lines) > 0 else [0, 0, 0, 0]

    return l_lane, r_lane

def average_line(lines):
    if len(lines) > 0:
        return [
            np.sum(lines[:, 0]) / len(lines),
            np.sum(lines[:, 1]) / len(lines),
            np.sum(lines[:, 2]) / len(lines),
            np.sum(lines[:, 3]) / len(lines)
        ]
    else:
        return [0, 0, 0, 0]
```

```
def hough(img, min_line_len, min_slope, max_slope):
    lines = cv2.HoughLinesP(img, rho=1, theta=np.pi/180, threshold=30, minLineLength=min_line_len, maxLineGap=30)
    lines = np.squeeze(lines)
    lanes, slopes = restrict_deg(lines, min_slope, max_slope)
    l_lane, r_lane = separate_line(lanes, slopes)
    return l_lane, r_lane

def lane_detection(img, min_line_len, min_slope, max_slope, low, high):
    gray_img = grayscale(img)
    blur_img = gaussian_blur(gray_img, 5)
    canny_img = canny(blur_img, low, high)
    roi_img = roi(canny_img)
    l_lane, r_lane = hough(roi_img, min_line_len, min_slope, max_slope)
    led3_state = GPIO.input(led3_pin)
    led4_state = GPIO.input(led4_pin)

    if all(l_lane):
        cv2.line(img, (int(l_lane[0]), int(l_lane[1])), (int(l_lane[2]), int(l_lane[3])), color=[0, 0, 255], thickness=15)
    else:
        if not led3_state and not led4_state:
            GPIO.output(led1_pin, GPIO.HIGH)
            time.sleep(0.02)
            GPIO.output(led1_pin, GPIO.LOW)
            GPIO.output(led2_pin, GPIO.HIGH)
            time.sleep(0.02)
            GPIO.output(led2_pin, GPIO.LOW)
        if all(r_lane):
            cv2.line(img, (int(r_lane[0]), int(r_lane[1])), (int(r_lane[2]), int(r_lane[3])), color=[255, 0, 0], thickness=15)
        else:
            if not led4_state and not led3_state:
                GPIO.output(led1_pin, GPIO.HIGH)
                time.sleep(0.02)
                GPIO.output(led1_pin, GPIO.LOW)
                GPIO.output(led2_pin, GPIO.HIGH)
                time.sleep(0.02)
                GPIO.output(led2_pin, GPIO.LOW)
    return img

def nothing(pgs):
    pass

# Lane detection and motor control loop
try:
    capture = cv2.VideoCapture(0)
    capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
    cv2.namedWindow(winname='Lane Detection')
    cv2.createTrackbar('houghMinLine', 'Lane Detection', 20, 200, nothing)
    cv2.createTrackbar('slopeMinDeg', 'Lane Detection', 100, 180, nothing)
    cv2.createTrackbar('slopeMaxDeg', 'Lane Detection', 140, 180, nothing)
    cv2.createTrackbar('threshold1', 'Lane Detection', 50, 1000, nothing)
    cv2.createTrackbar('threshold2', 'Lane Detection', 200, 1000, nothing)

    while cv2.waitKey(1) != ord('q'):
        _, frame = capture.read()
        min_line_len = cv2.getTrackbarPos(trackbarname='houghMinLine', winname='Lane Detection')
        min_slope = cv2.getTrackbarPos('slopeMinDeg', 'Lane Detection')
        max_slope = cv2.getTrackbarPos('slopeMaxDeg', 'Lane Detection')
        low = cv2.getTrackbarPos('threshold1', 'Lane Detection')
        high = cv2.getTrackbarPos('threshold2', 'Lane Detection')
        try:
            result_img = lane_detection(frame, min_line_len, min_slope, max_slope, low, high)
            cv2.imshow('Lane Detection', result_img)
        except Exception:
            cv2.imshow('Lane Detection', frame)

finally:
    capture.release()
    cv2.destroyAllWindows()
```

차선인식

```
def grayscale(img):  
    return cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)  
  
def gaussian_blur(img, kernel_size):  
    return cv2.GaussianBlur(img, (kernel_size, kernel_size), 0)  
  
def canny(img, low_threshold, high_threshold):  
    return cv2.Canny(img, low_threshold, high_threshold)  
  
def roi(img):  
    mask = np.zeros_like(img)  
    h, w = mask.shape  
    vertices = np.array([[w/10, h], (w/10, h*2/5), (w*9/10, h*2/5), (w*9/10, h)], dtype=np.int32)  
    cv2.fillPoly(mask, vertices, 255)  
    roi_img = cv2.bitwise_and(img, mask)  
    return roi_img
```

1. 그레이스케일 - 이미지 흑백 처리
2. 가우시안 블러 - 이미지 흐릿하게 처리
3. 캐니 외곽 검출법 - 이미지에서 외곽선 검출
4. ROI 설정 - 이미지에서 사용하고 싶은 영역만 활성화

차선인식

```
def restrict_deg(lines, min_slope, max_slope):
    if lines.ndim == 0:
        return lines, np.array([])
    lines = np.squeeze(lines)
    slope_deg = np.rad2deg(np.arctan2(lines[:, 1] - lines[:, 3], lines[:, 0] - lines[:, 2]))
    lines = lines[np.abs(slope_deg) < max_slope]
    slope_deg = slope_deg[np.abs(slope_deg) < max_slope]
    lines = lines[np.abs(slope_deg) > min_slope]
    slope_deg = slope_deg[np.abs(slope_deg) > min_slope]
    return lines, slope_deg

def separate_line(lines, slope_deg):
    l_lines = lines[(slope_deg > 0), :]
    r_lines = lines[(slope_deg < 0), :]

    l_lane = average_line(l_lines) if len(l_lines) > 0 else [0, 0, 0, 0]
    r_lane = average_line(r_lines) if len(r_lines) > 0 else [0, 0, 0, 0]

    return l_lane, r_lane

def average_line(lines):
    if len(lines) > 0:
        return [
            np.sum(lines[:, 0]) / len(lines),
            np.sum(lines[:, 1]) / len(lines),
            np.sum(lines[:, 2]) / len(lines),
            np.sum(lines[:, 3]) / len(lines)
        ]
    else:
        return [0, 0, 0, 0]

def hough(img, min_line_len, min_slope, max_slope):
    lines = cv2.HoughLinesP(img, rho=1, theta=np.pi/180, threshold=30, minLineLength=min_line_len, maxLineGap=30)
    lines = np.squeeze(lines)
    lanes, slopes = restrict_deg(lines, min_slope, max_slope)
    l_lane, r_lane = separate_line(lanes, slopes)
    return l_lane, r_lane
```

5. 각도 제한 - 필요한 이미지만 검출하기 위해 사용

6. 차선 구분 - 왼쪽과 오른쪽 차선 구분

7. 평균 합 계산 - 구분된 차선 이미지에 생긴 여러 선의 합의 평균을 구해 하나의 선으로 출력

8. 허프 변환 - 프로그램이 선을 인식할 수 있게 해주는 과정

차선인식

```
def lane_detection(img,min_line_len,min_slope,max_slope,low,high):  
    gray_img = grayscale(img)  
    blur_img = gaussian_blur(gray_img, 5)  
    canny_img = canny(blur_img, low, high)  
    roi_img = roi(canny_img)  
    l_lane,r_lane = hough(roi_img,min_line_len,min_slope,max_slope)  
  
    if all(l_lane):  
        cv2.line(img, (int(l_lane[0]), int(l_lane[1])), (int(l_lane[2]), int(l_lane[3])), color=[0, 0, 255], thickness=15)  
    if all(r_lane):  
        cv2.line(img, (int(r_lane[0]), int(r_lane[1])), (int(r_lane[2]), int(r_lane[3])), color=[255, 0, 0], thickness=15)  
  
    return img
```

9. 차선 인식 - 앞의 모든 과정에서 처리 된 이미지를 출력

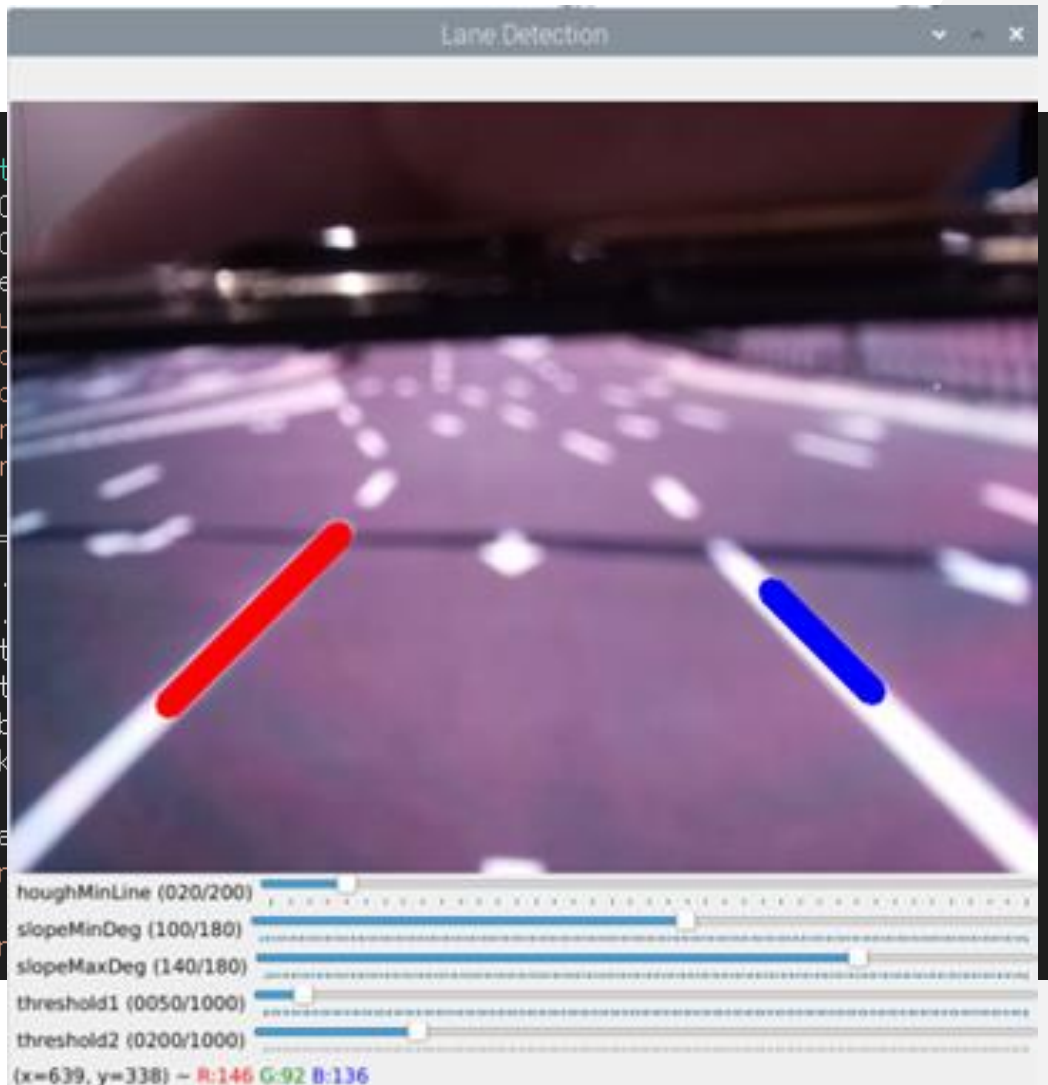
왼쪽 차선과 오른쪽 차선 따로 인식

차선 이탈의 핵심이 되는 부분

차선인식

```
try:
    capture = cv2.VideoCapture(0)
    capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
    cv2.namedWindow(winname)
    cv2.createTrackbar('houghMinLine', winname, 0, 200, 0)
    cv2.createTrackbar('slopeMinDeg', winname, 0, 180, 0)
    cv2.createTrackbar('slopeMaxDeg', winname, 0, 180, 0)
    cv2.createTrackbar('threshold1', winname, 0, 1000, 0)
    cv2.createTrackbar('threshold2', winname, 0, 1000, 0)

    while cv2.waitKey(1) != -1:
        _, frame = capture.read()
        min_line_len = cv2.getTrackbarPos('houghMinLine', winname)
        min_slope = cv2.getTrackbarPos('slopeMinDeg', winname)
        max_slope = cv2.getTrackbarPos('slopeMaxDeg', winname)
        low = cv2.getTrackbarPos('threshold1', winname)
        high = cv2.getTrackbarPos('threshold2', winname)
        try:
            result_img = lane_detect(frame, min_line_len, min_slope, max_slope, low, high)
            cv2.imshow('Lane Detection', result_img)
        except Exception:
            cv2.imshow('Lane Detection', frame)
```



10. 트랙바 생성

- 실시간으로 변수들 조절하기 위해 사용
- 허프 함수, 각도, 외곽선 검출 등을 트랙바로 사용
- 기본값 지정 가능

11. 영상 처리

- 차선이 인식되지 않더라도 영상 유지

모터제어 및 RC카 제어

```
# Motor Control Logic
def motor_thread():
    while True:
        user_input = input("Enter a command (e.g., 'go', 'back', 'left', 'right', 'stop'): ")
        execute_command(user_input)

STOP = 0
FORWARD = 1
BACKWARD = 2

CH1 = 0
CH2 = 1

OUTPUT = 1
INPUT = 0

HIGH = 1
LOW = 0

# PWM PIN
ENA = 26
ENB = 0

# GPIO PIN
IN1 = 19
IN2 = 13
IN3 = 6
IN4 = 5
```


모터제어 및 RC카 제어

```
def setPinConfig(EN, INA, INB):
    GPIO.setup(EN, GPIO.OUT)
    GPIO.setup(INA, GPIO.OUT)
    GPIO.setup(INB, GPIO.OUT)
    # 100khz 로 PWM 동작 시킴
    pwm = GPIO.PWM(EN, 100)
    # 우선 PWM 멈춤.
    pwm.start(0)
    return pwm

def setMotorControl(pwm, INA, INB, speed, stat):
    # 모터 속도 제어 PWM
    pwm.ChangeDutyCycle(speed)

    if stat == FORWARD:
        GPIO.output(INA, HIGH)
        GPIO.output(INB, LOW)
    elif stat == BACKWARD:
        GPIO.output(INA, LOW)
        GPIO.output(INB, HIGH)
    elif stat == STOP:
        GPIO.output(INA, LOW)
        GPIO.output(INB, LOW)

def setMotor(ch, speed, stat):
    if ch == CH1:
        setMotorControl(pwmA, IN1, IN2, speed, stat)
    else:
        setMotorControl(pwmB, IN3, IN4, speed, stat)

GPIO.setmode(GPIO.BCM)

pwmA = setPinConfig(ENA, IN1, IN2)
pwmB = setPinConfig(ENB, IN3, IN4)
```

모터제어 및 RC카 제어

```
def execute_command(command):  
    if command == 'g':  
        setMotor(CH1, 90, FORWARD)  
        setMotor(CH2, 90, FORWARD)  
    elif command == 'b':  
        setMotor(CH1, 100, BACKWARD)  
        setMotor(CH2, 100, BACKWARD)  
    elif command == 'r':  
        setMotor(CH1, 90, FORWARD)  
        setMotor(CH2, 40, FORWARD)  
    elif command == 'l':  
        setMotor(CH1, 40, FORWARD)  
        setMotor(CH2, 90, FORWARD)  
    elif command == 's':  
        setMotor(CH1, 0, STOP)  
        setMotor(CH2, 0, STOP)
```

LED, BUZZER 세팅

```
led1_pin = 2  
led2_pin = 3  
led4_pin = 14  
led3_pin = 15  
speaker_pin = 4
```

```
def led_blinking():  
    GPIO.setmode(GPIO.BCM)  
    GPIO.setup(led1_pin, GPIO.OUT)  
    GPIO.setup(led3_pin, GPIO.OUT)  
    GPIO.setup(led4_pin, GPIO.OUT)  
    GPIO.setup(led2_pin, GPIO.OUT)  
    GPIO.setup(speaker_pin, GPIO.OUT)  
    GPIO.output(led1_pin, GPIO.LOW)  
    GPIO.output(led2_pin, GPIO.LOW)  
    GPIO.output(led3_pin, GPIO.LOW)  
    GPIO.output(led4_pin, GPIO.LOW)
```

```
motor_thread = threading.Thread(target=motor_thread)  
led_thread = threading.Thread(target=led_blinking)  
motor_thread.daemon = True  
motor_thread.start()  
led_thread.start()  
led_thread.join()
```

- LED와 BUZZER 핀 세팅
- Rpi.GPIO로 LED와 BUZZER 세팅 후 초기 상태 LOW로 세팅
- Thread를 이용해 LED와 BUZZER가 포함된 Try 구문 개별 동작 세팅

차선 이탈시 LED, BUZZER 알림

```
def lane_detection(img, min_line_len, min_slope, max_slope, low, high):  
    gray_img = grayscale(img)  
    blur_img = gaussian_blur(gray_img, 5)  
    canny_img = canny(blur_img, low, high)  
    roi_img = roi(canny_img)  
    l_lane, r_lane = hough(roi_img, min_line_len, min_slope, max_slope)  
    led3_state = GPIO.input(led3_pin)  
    led4_state = GPIO.input(led4_pin)
```

- def lane_detection에서 차선 이탈 구문 사용
- LED 구분 필요 : led1과 led2는 빨간색 LED, 경고등으로 활용
led3과 led4는 초록색 LED, 깜빡이로 활용
- led_state를 선언해 깜빡이 역할을 하는 led3과 led4과 현재 켜져 있는지 꺼져 있는지 상태 확인

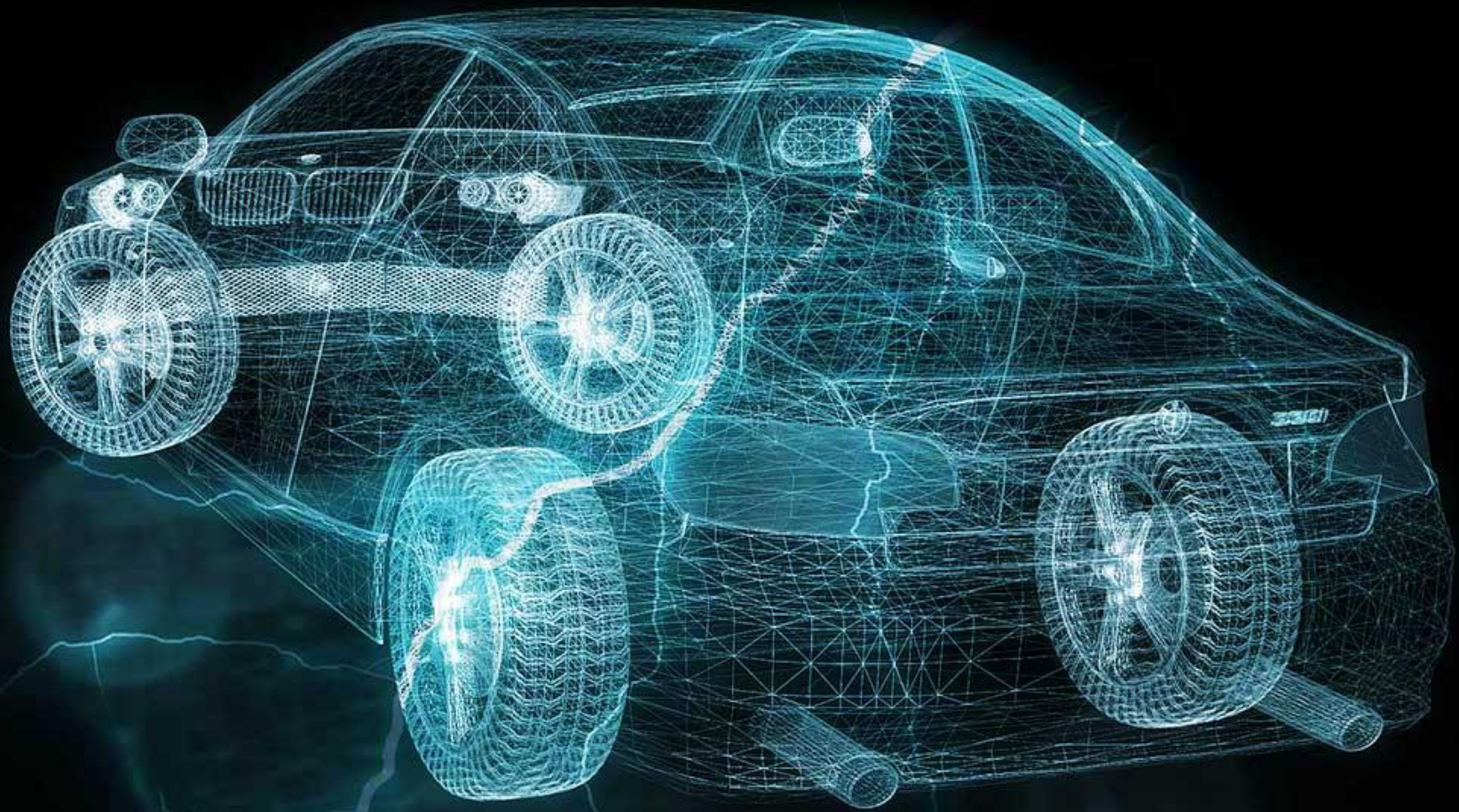
차선 이탈시 LED, BUZZER 알림

```
if all(l_lane):
    cv2.line(img, (int(l_lane[0]), int(l_lane[1])), (int(l_lane[2]), int(l_lane[3])), color=[0, 0, 255], thickness=15)
else:
    if not led3_state and not led4_state:
        GPIO.output(led1_pin, GPIO.HIGH)
        time.sleep(0.02)
        GPIO.output(led1_pin, GPIO.LOW)
        GPIO.output(led2_pin, GPIO.HIGH)
        time.sleep(0.02)
        GPIO.output(led2_pin, GPIO.LOW)
        time.sleep(0.02)
        GPIO.output(speaker_pin, GPIO.HIGH)
        time.sleep(0.02)
        GPIO.output(speaker_pin, GPIO.LOW)
    if all(r_lane):
        cv2.line(img, (int(r_lane[0]), int(r_lane[1])), (int(r_lane[2]), int(r_lane[3])), color=[255, 0, 0], thickness=15)
    else:
        if not led4_state and not led3_state:
            GPIO.output(led1_pin, GPIO.HIGH)
            time.sleep(0.02)
            GPIO.output(led1_pin, GPIO.LOW)
            GPIO.output(led2_pin, GPIO.HIGH)
            time.sleep(0.02)
            GPIO.output(led2_pin, GPIO.LOW)
            time.sleep(0.02)
            GPIO.output(speaker_pin, GPIO.HIGH)
            time.sleep(0.02)
            GPIO.output(speaker_pin, GPIO.LOW)
return img
```

◆ 차선 이탈의 핵심

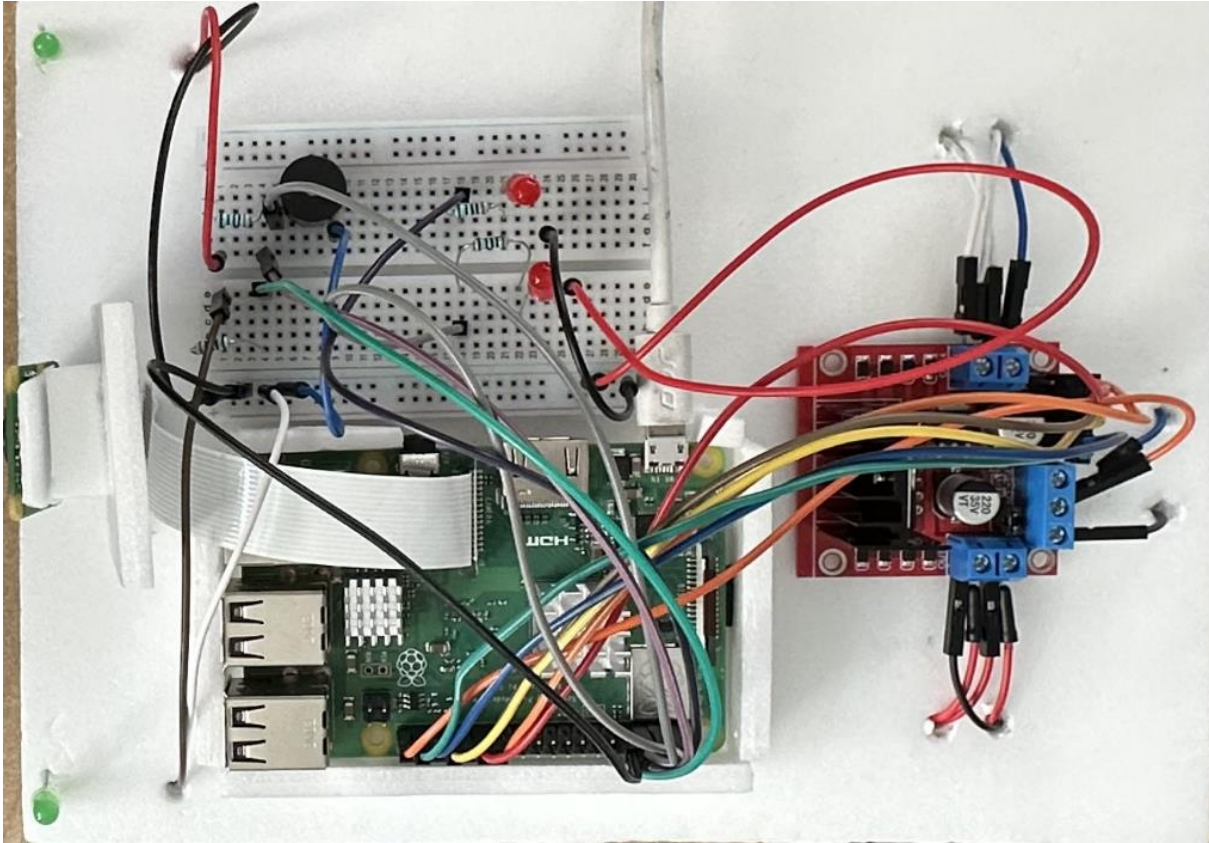
- 차선 인식 시 이미지 출력
- 차선 이탈 시 else문 실행
 - 만약 led3이나 led4가 꺼져 있다면, led1과 led2와 buzzer 실행
 - 만약 led3이나 led4가 켜져 있다면, led1과 led2와 buzzer 실행 X
- 왼쪽 차선과 오른쪽 차선 개별 동작
 - 오른쪽 차선이 인식되지 않을 경우 왼쪽으로 차선 이탈 간주
 - 왼쪽 차선이 인식되지 않을 경우 오른쪽으로 차선 이탈 간주

외형모습



RC카

자동차 상단부



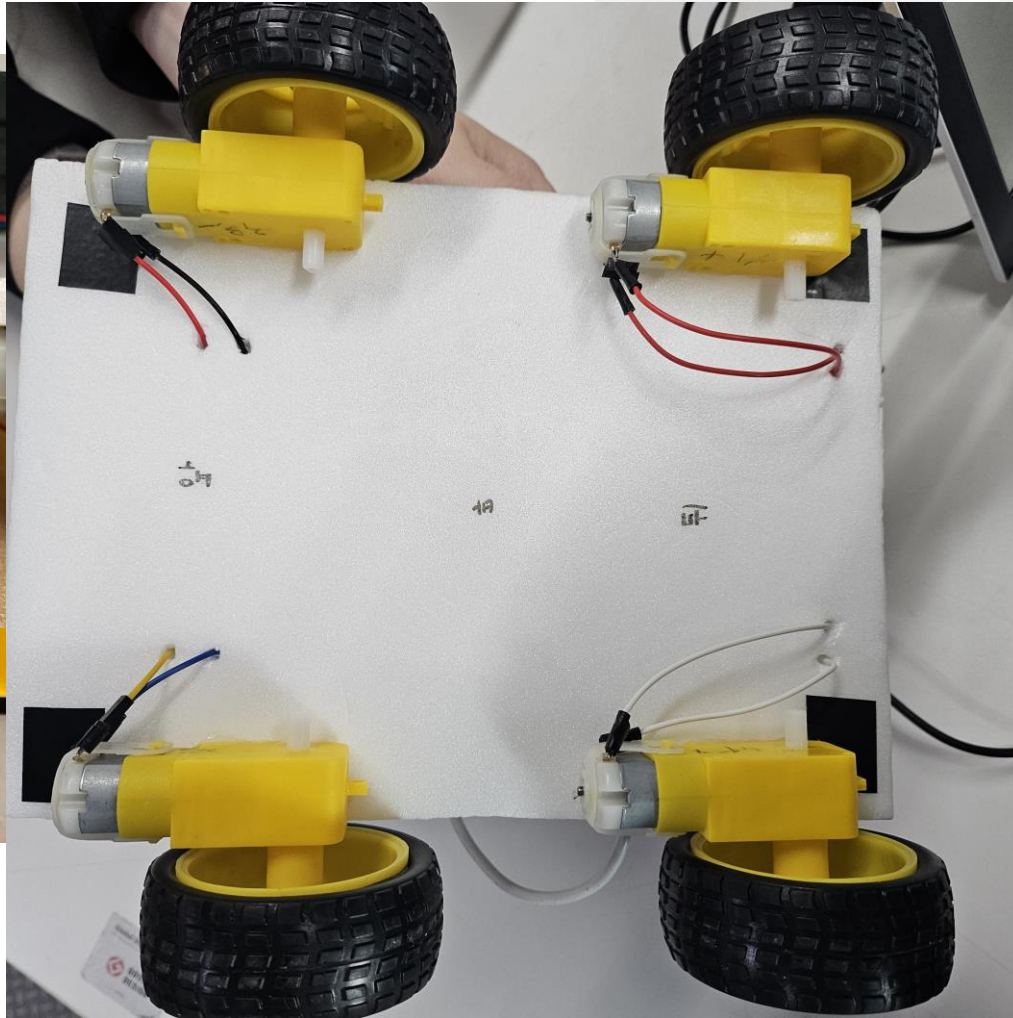
RC카

자동차 중앙부



RC카

자동차 하단부

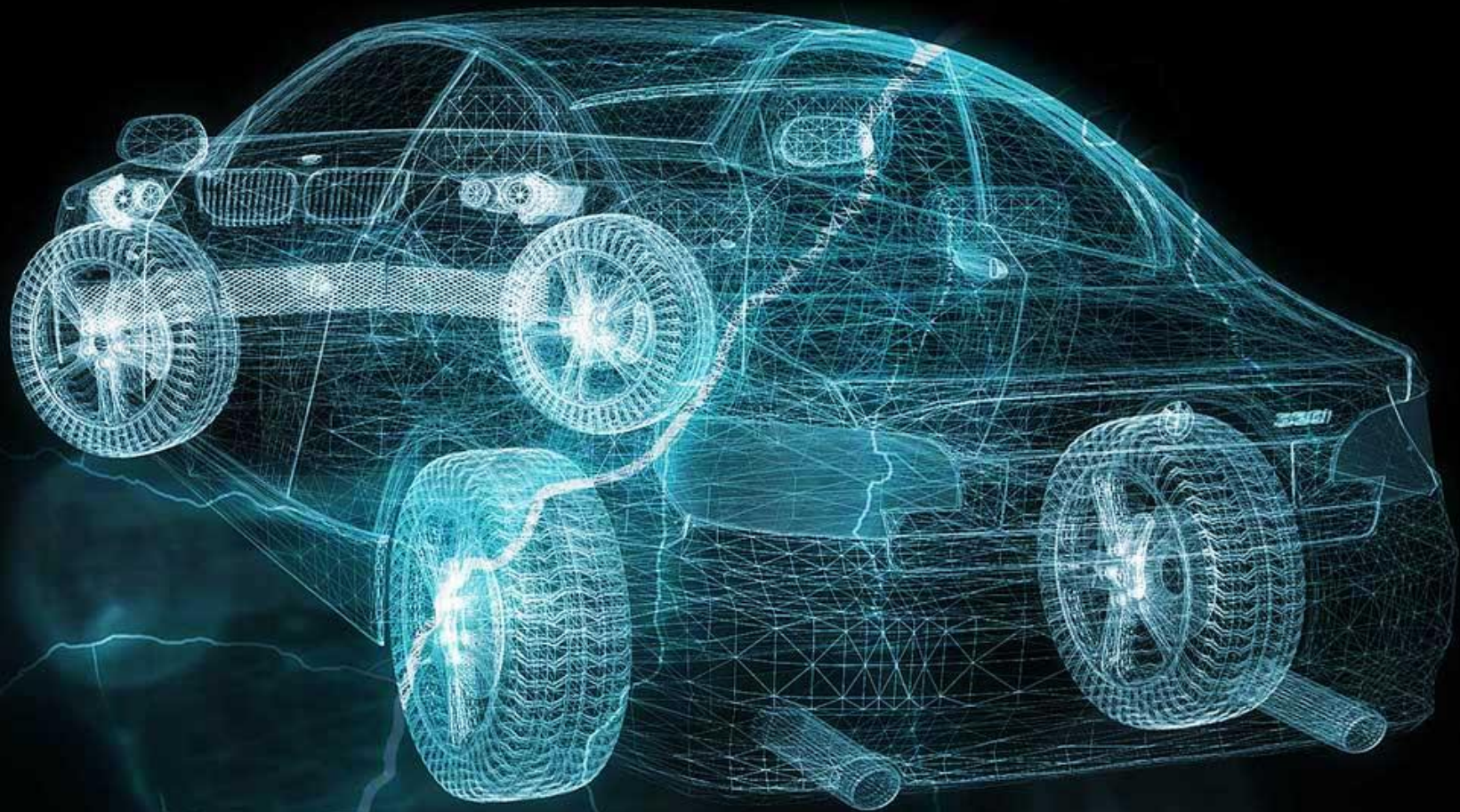


RC카

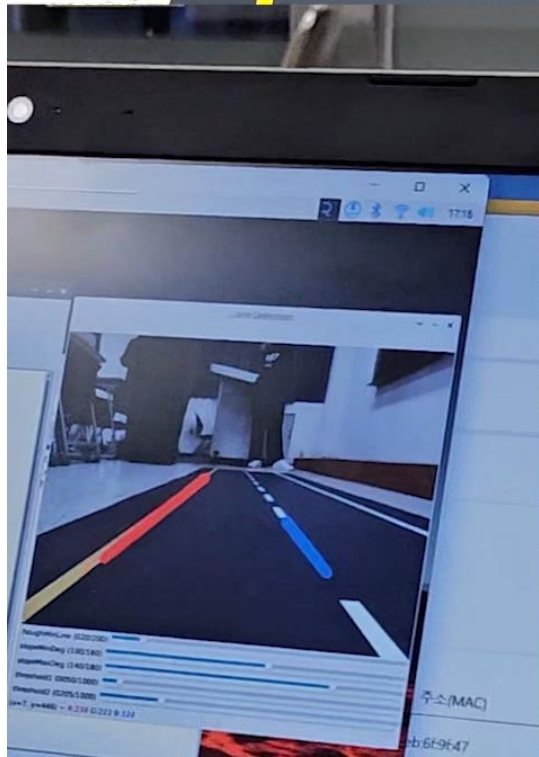
최종 외형모습



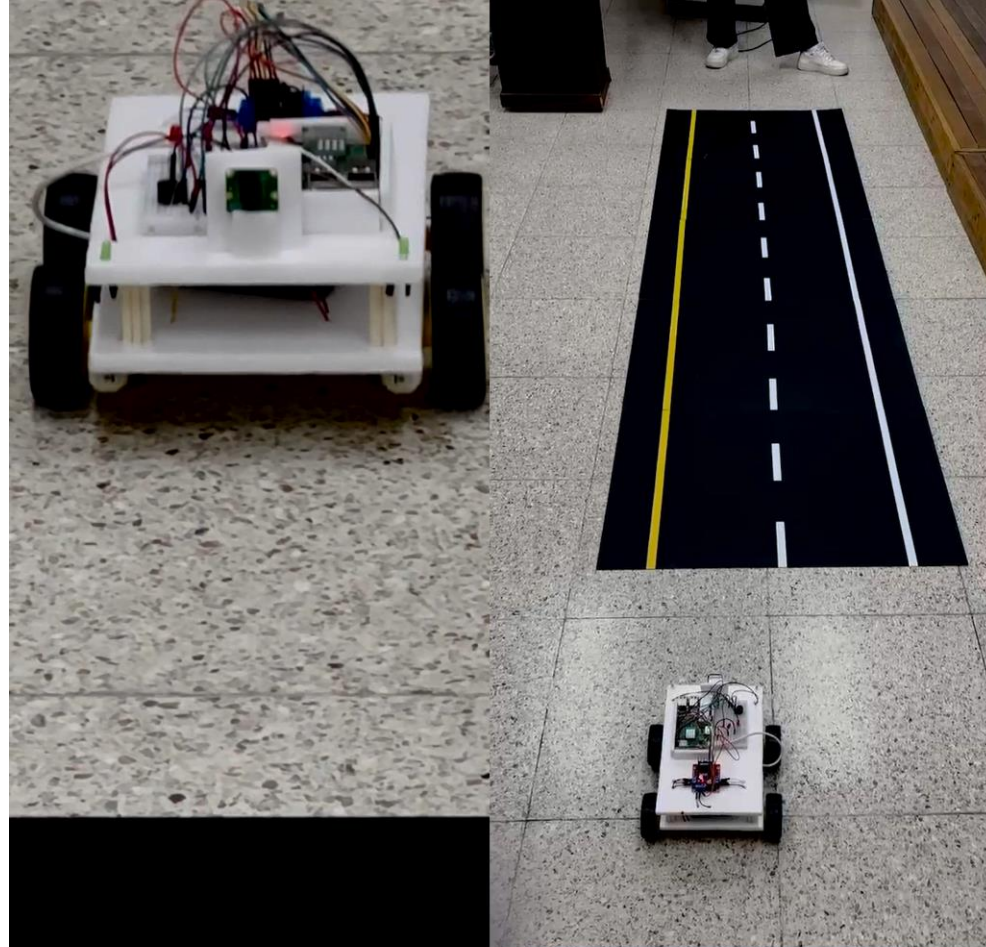
데모영상



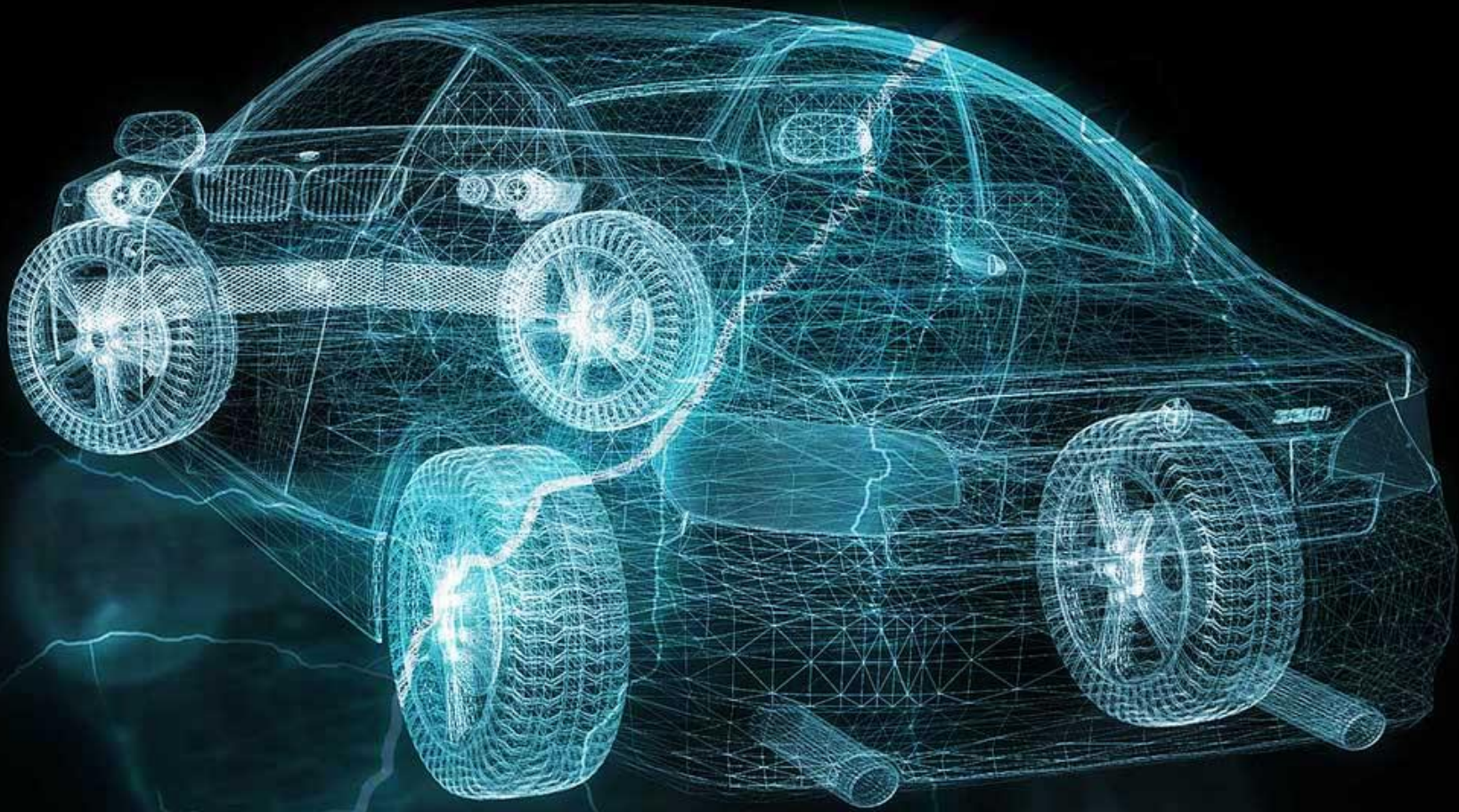
The photograph shows a person standing on a black mat with yellow, white dashed, and white solid lines, used for a driving simulation. A laptop in the foreground displays the corresponding simulation software interface, which includes a first-person view of the road and various performance metrics.



차선변경



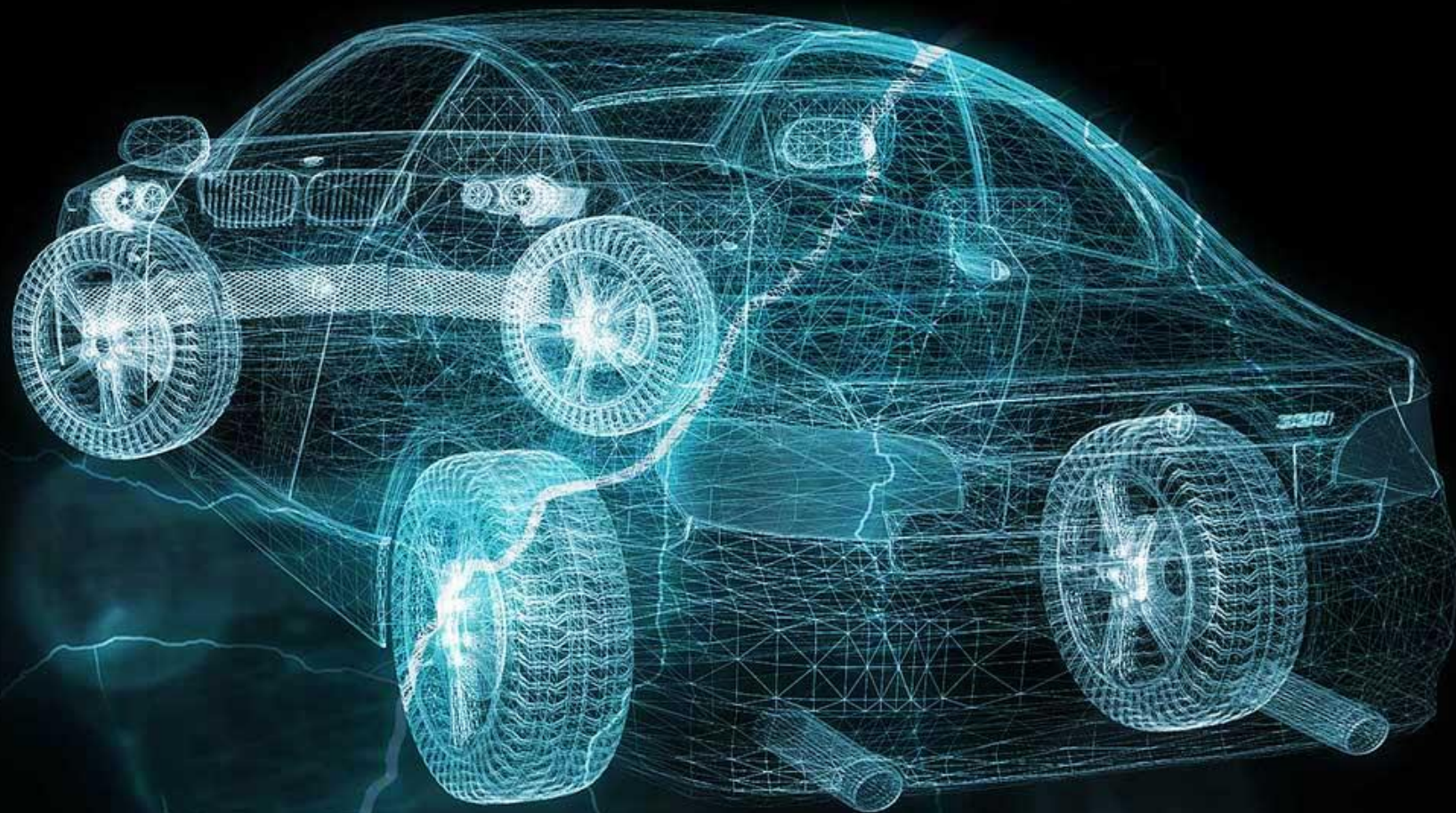
미래의
연료

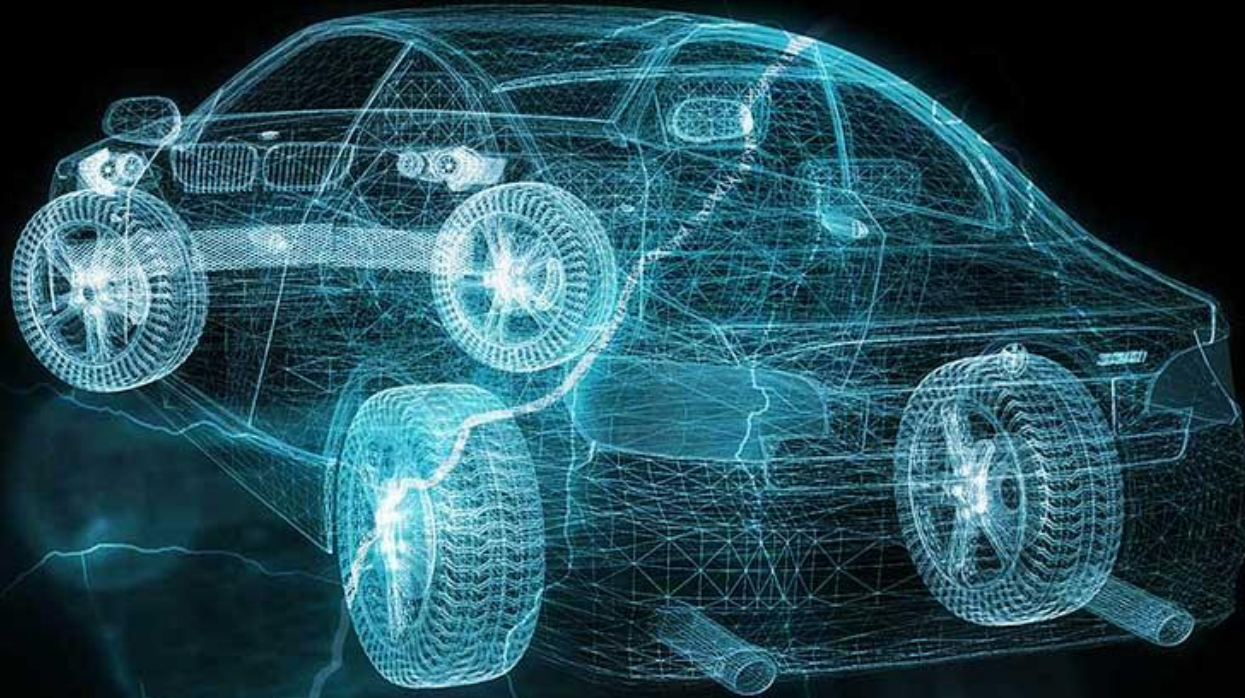


팀원 역할

개발 파트	학번/이름	내용
S/W	1723001	팀장, 모터제어 개발, 라즈베리파이 기본셋업, 차선이탈개발, 각 개발된 코드 통합
	김우성	
	1722427	차선인식 개발, 라즈베리파이 카메라셋업, 차선이탈개발, 각 개발된 코드 통합
	이예찬	
	1923361	차선이탈시 LED, BUZZER 작동 개발, 방향지시등 LED개발
	김지은	
H/W	1930394	RC카 제작, 시연용 도로 제작
	노이지	
	1922474	RC카 제작, 시연용 도로 제작
	박아진	

Q&A





THANK YOU

영상처리를 이용한 차선 자동인식 알고리즘 개발