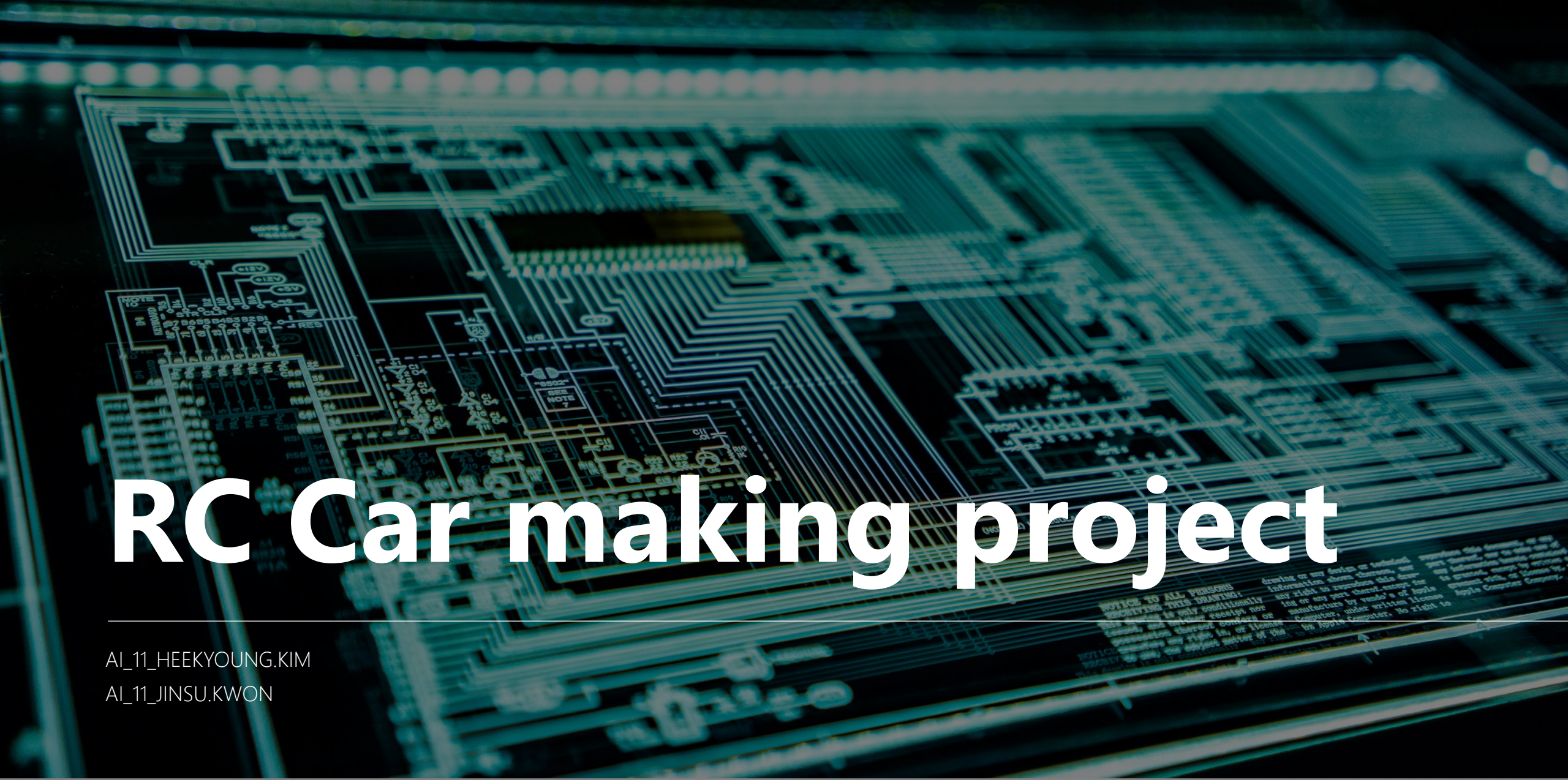


RC Car making project

AI_11_HEEKYOUNG.KIM
AI_11_JINSU.KWON

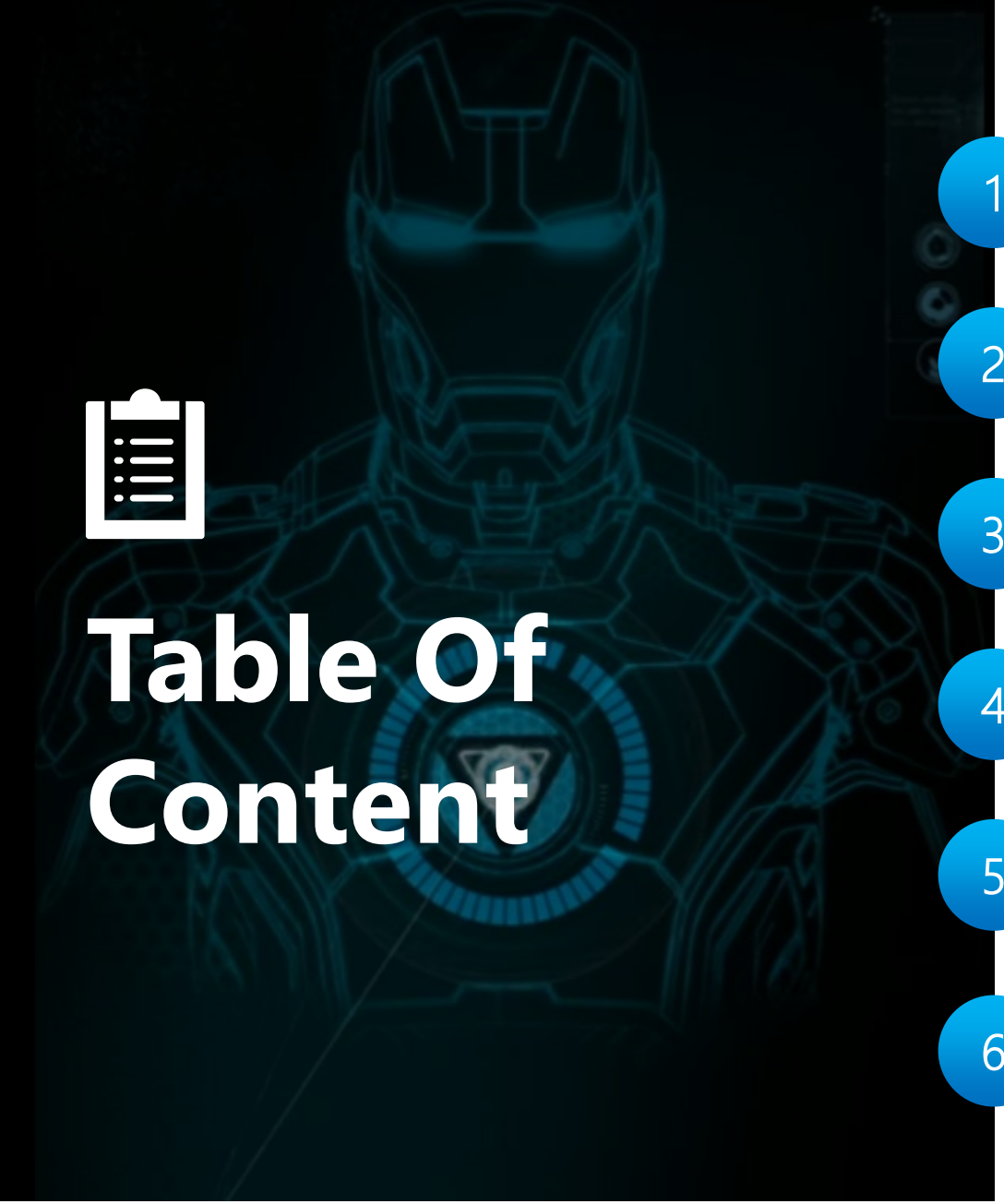
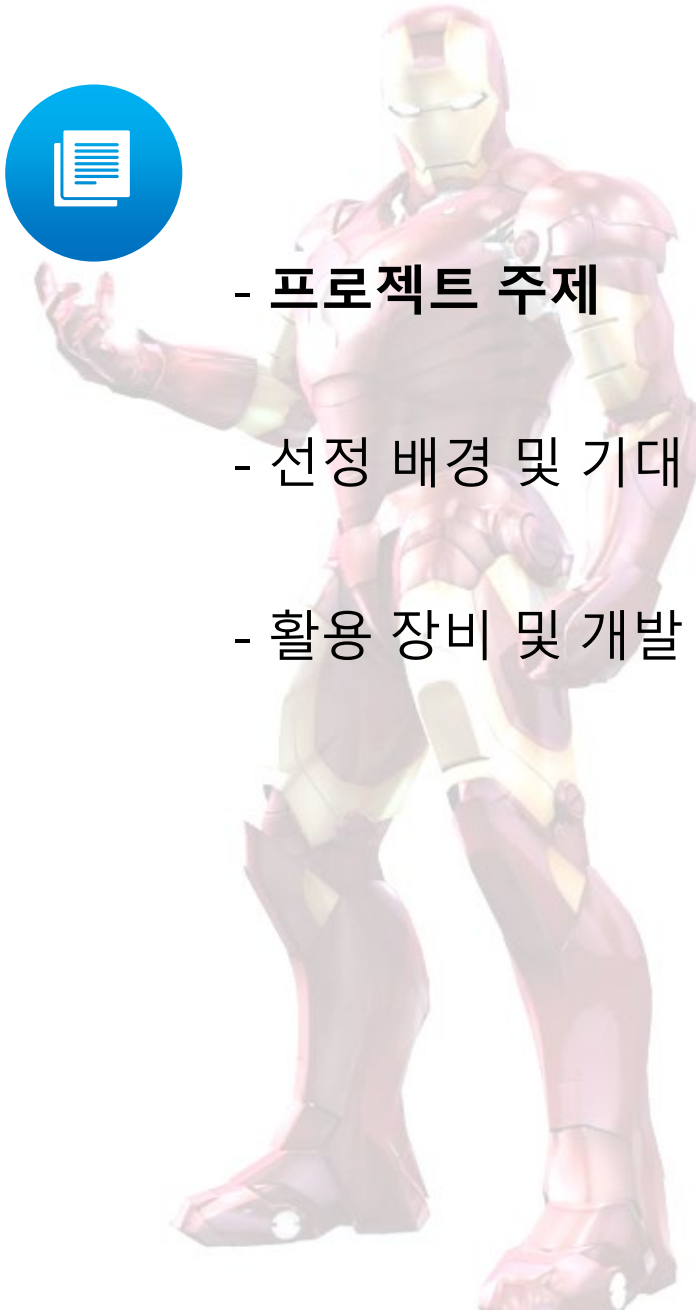


Table Of Content

1. 프로젝트 개요



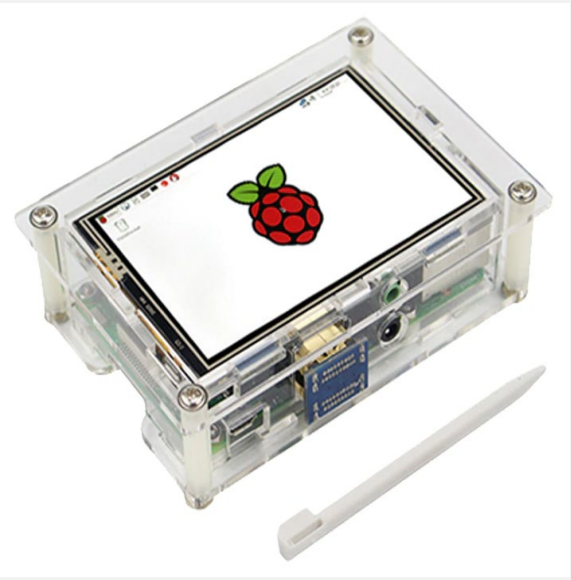
- **프로젝트 주제**

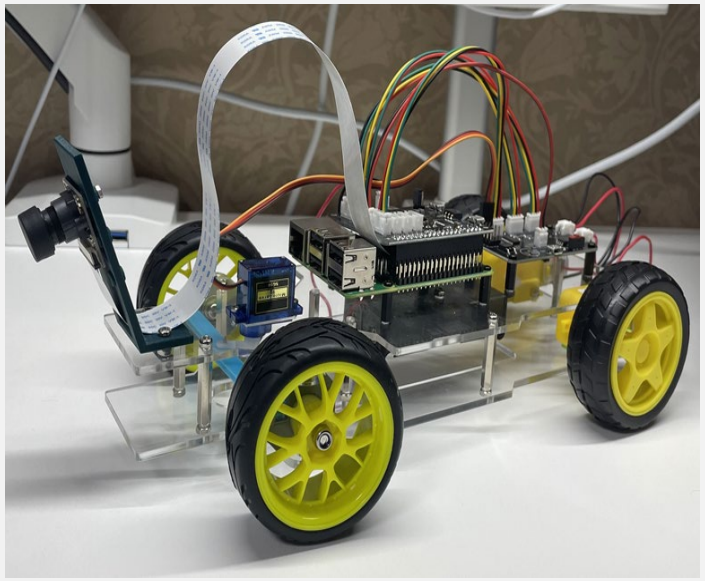
- 선정 배경 및 기대 효과

- 활용 장비 및 개발 환경

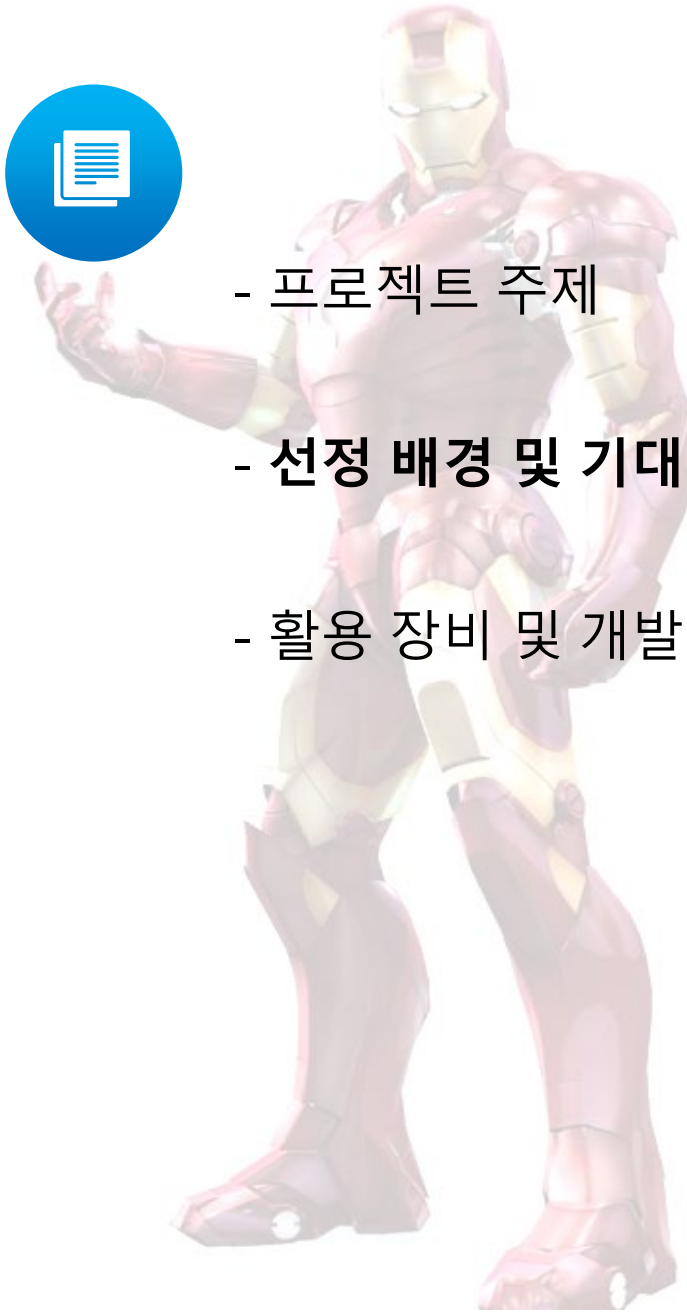
프로젝트 주제

라즈베리파이를 이용한 자율주행 RC CAR





1. 프로젝트 개요



- 프로젝트 주제
- **선정 배경 및 기대 효과**
- 활용 장비 및 개발 환경

선정 배경 및 기대 효과

선정 배경

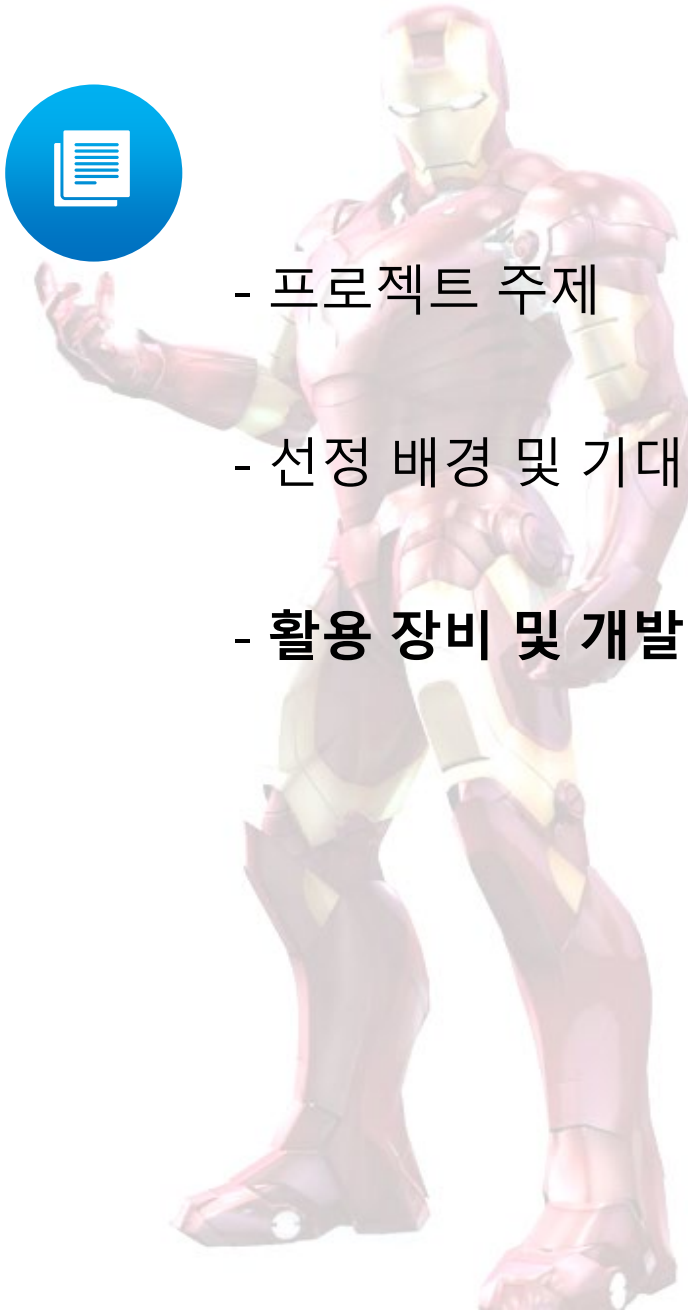
- 교통사고분석시스템(TAAS)통계에 따르면 최근 안전 운전 불이행(전방 미 주시, 과속, 신호위반)으로 인한 사고 발생률이 55% 이상 차지
- 운전자는 위험을 인지하고 있으나, 안전운전을 지키지 않는 경우, 위험을 인지하지 못하였을 때 갑작스럽게 발생하는 경우

기대 효과

- 최근 인공지능 기술이 발전함에 따라 자율주행,

  첨단 운전자 지원 시스템과 같은 기술들이 개발
- 이러한 기술들은 교통사고를 예방하여 사망률을 감소, 운전자의 편의성 향상

1. 프로젝트 개요



- 프로젝트 주제

- 선정 배경 및 기대 효과

- **활용 장비 및 개발 환경**

활용 장비 및 개발 환경

활용 장비

- RaspberryPi 3B+
- RaspberryPi 전용 RC카 키트
- 소프트웨어 설치 및 셋업 용 SD카드
- 원격 환경이 설정된 컴퓨터

개발 환경

- RaspberryPi 전용 OS(Raspbian) 64bit
- Python3 (v3.9.2)
- Opencv(v4.2)
- Tensorflow(v2.8.0)
- Keras(v2.8.0)
- Adafruit_servokit
- Rpi.gpio

2. 프로젝트 팀 구성 및 역할

팀장 : 김희경

- 라즈베리파이 셋업
- 개발 환경 구축
- 하드웨어 셋업

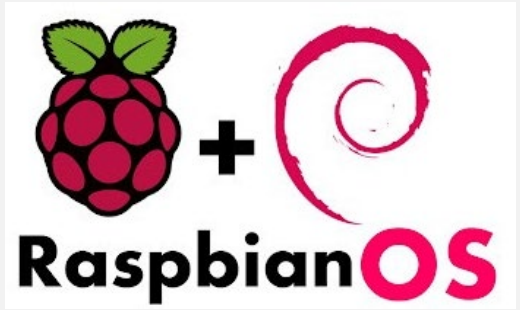
팀원 : 권진수

- 라즈베리파이 셋업
- 개발 환경 구축
- 하드웨어 셋업

3. 프로젝트 수행 절차 및 방법

구분	기간	주간 기획	수행 및 완료	비고
사전 기획	2022.07.11 ~ 2022.07.13	프로젝트시작 전 팀 결성 및 간단한 기획서 작성	프로젝트 방향에 대한 토론	
계획 수립	2022.07.14 ~ 2022.07.18	객체 탐지 모델 공부 및 추가 필요부분 찾아보기	조사했던 자료 공유 및 필요 장비 구매 목록 작성	라즈베리파이, RC카 키트 구입
환경 구축	2022.07.19 ~ 2022.07.22	필요한 소프트웨어 설치	Raspbian,Opencv, Tensorflow lite, cocomobile net 등 필요 라이브러리 설치	2022.07.22 RC카 키트 도착 후 조립완료
하드웨어 & 소프트웨어 셋업	2022.07.25 ~ 2022.07.28	하드웨어 구동, Opencv를 이용한 차선인식	RC카 구동 테스트완료, 차선 검출 테스트	
딥러닝 트레이닝	2022.07.29 ~ 2022.07.31	CNN을 이용한 차선 딥러닝	CNN을 이용한 차선 딥러닝 시도하였으나 실행 오류	딥러닝 학습이 안되는 이유 발견
환경 재구축	2022.07.31 ~ 2022.08.03	CNN을 이용한 차선 딥러닝	CNN학습을 위한 환경 재구축(Tensorflow, Keras) 후 딥러닝 완료	
발표자료 & 영상촬영	2022.08.04~2022.08.05		PPT 작성, 영상촬영 완료	

4. 프로젝트 구조

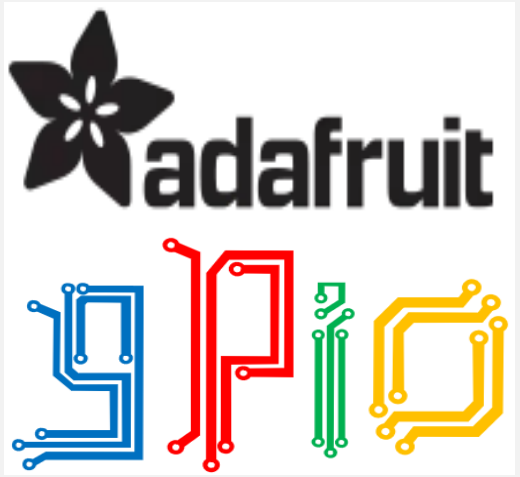


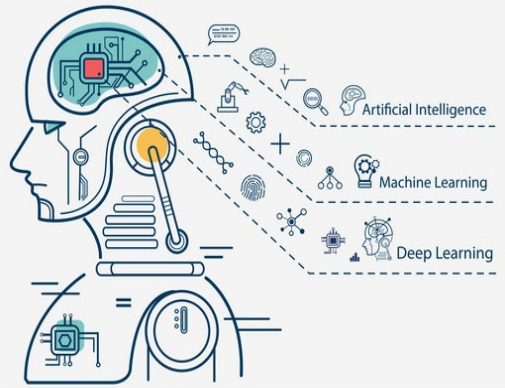
Raspberry_Pi에
Raspbian os 설치
기본 환경 셋팅

Python3, OpenCV,
Kesra,Tensorflow
환경 구축

하드웨어 조립

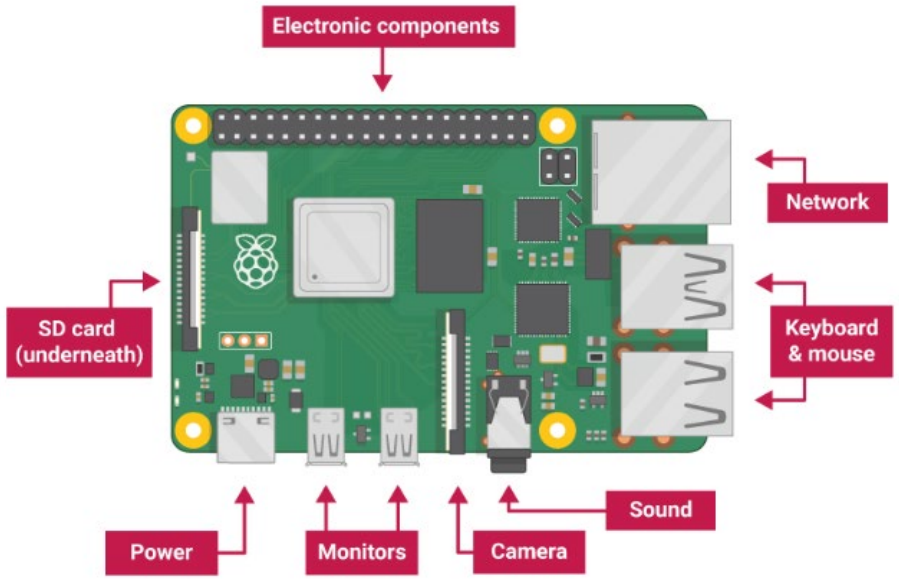
앞바퀴 서보모터 제어
뒷바퀴 모터 제어





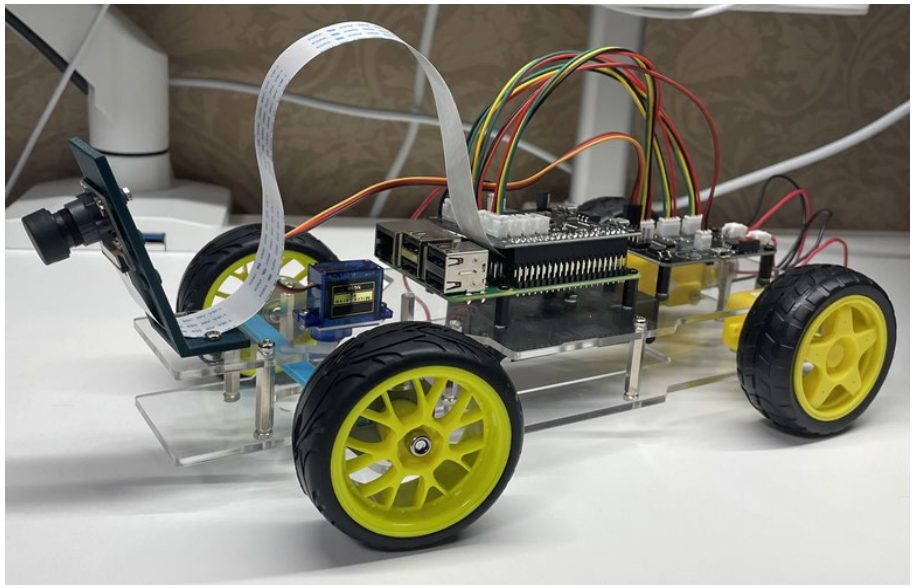
딥러닝 차선 인식 주행

5. 프로젝트 수행 결과



Raspberry Pi

- Raspbian OS(32bit -> 64bit) 설치
- VNC를 이용한 원격 환경 조성
- Python & Python3(v3.7.12 -> v3.9.2)
- Opencv(v3.4.6 -> v4.6.0)
- Tensorflow Lite에는 keras설치 X -> Keras(v2.8.0)
- Tensorflow(Lite -> v2.8.0)



하드웨어 조립(ssingssing car)

- 뒷바퀴 구동

  GPIO 라이브러리를 이용하여 각각의 모터 속도를 제어

- 앞바퀴 각도 제어

  Adafruit라이브러리를 사용해 조향용 서보모터 제어

- Pi 카메라 인식

5. 프로젝트 수행 결과

차선인식 주행

- 카메라를 통해 Opencv라이브러리를 이용하여 기초적인 차선인식 주행을 할 수 있다.
- RGB인 이미지가 사용하는 색공간을 HSV색공간으로 변환한다.

```
def _detect_edges(frame):
    # filter for blue lane lines
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    _show_image("hsv", hsv)
```

- 변환된 이미지를 Canny edge detection으로 이미지의 가장자리를 감지한다.

```
edges = cv2.Canny(mask, 200, 400)
_show_image("blue edge", edges)
```

- 라인 세그먼트를 1 또는 2개의 차선 라인으로 결합해 라인 기울기에 따라 좌우 차선 감지

```
def _average_slope_intercept(frame, line_segments):
```

- 차선 좌표를 기반해 조향 각도 찾기

```
def _compute_steering_angle(frame, lane_lines):
```

- 감지한 차선을 따라 주행을 하며 데이터 가공을 위한 영상데이터를 수집한다

Lane detection Algorithms

- Grayscale

  인식률을 높이고, 연산량이 감소하도록 이미지를 흑백화

- Gaussian

  필터를 이용하여 잡음 제거

- Canny Edge

  이미지의 엣지를 추출

- Region Of interest

  추출된 이미지에서 차선으로 인식 할 부분을 관심 영역으로 설정

- Hough Transform

  설정한 관심 영역을 통해 직선을 검출

- Average line detection

  그 후 각 차선 기울기의 평균을 구하여 최적 평균 직선을 검출

5. 프로젝트 수행 결과

```
To enable them in other operations, rebuild TensorFlow with the ap
Model: "Nvidia_Model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
dropout (Dropout)            (None, 3, 20, 64)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 1, 18, 64)         36928
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 100)               115300
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_2 (Dense)              (None, 10)                510
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
=================================================================
Total params: 252,219
```

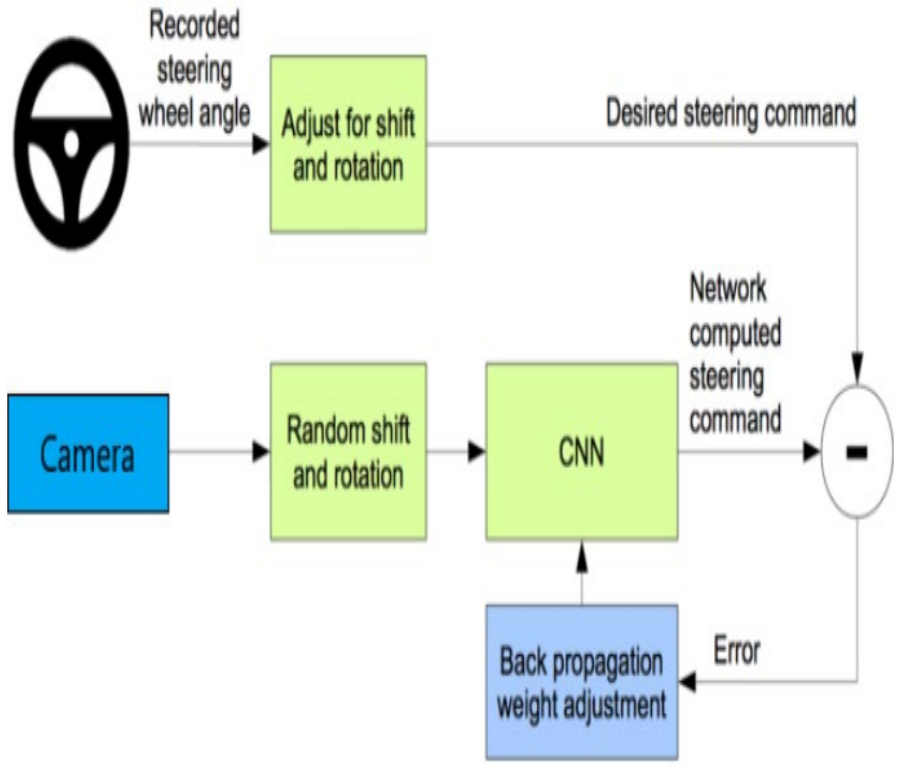
```
WARNING:tensorflow:From cobit_deep_learning.py:198: Model.fit_generator (from tensorflow.python.keras.engine
s deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
300/300 [==============================] - ETA: 0s - loss: 393.3931
Epoch 00001: val_loss improved from inf to 195.76166, saving model to output₩lane_navigation_check.h5
300/300 [==============================] - 180s 600ms/step - loss: 393.3931 - val_loss: 195.7617
Epoch 2/10
300/300 [==============================] - ETA: 0s - loss: 213.9230
Epoch 00002: val_loss improved from 195.76166 to 73.22735, saving model to output₩lane_navigation_check.h5
300/300 [==============================] - 179s 595ms/step - loss: 213.9230 - val_loss: 73.2273
Epoch 3/10
300/300 [==============================] - ETA: 0s - loss: 45.3683
Epoch 00003: val_loss improved from 73.22735 to 9.05148, saving model to output₩lane_navigation_check.h5
300/300 [==============================] - 178s 593ms/step - loss: 45.3683 - val_loss: 9.0515
Epoch 4/10
300/300 [==============================] - ETA: 0s - loss: 15.9749
Epoch 00004: val_loss improved from 9.05148 to 8.47410, saving model to output₩lane_navigation_check.h5
300/300 [==============================] - 179s 596ms/step - loss: 15.9749 - val_loss: 8.4741
Epoch 5/10
300/300 [==============================] - ETA: 0s - loss: 13.8560
Epoch 00005: val_loss improved from 8.47410 to 5.75985, saving model to output₩lane_navigation_check.h5
300/300 [==============================] - 179s 596ms/step - loss: 13.8560 - val_loss: 5.7598
Epoch 6/10
300/300 [==============================] - ETA: 0s - loss: 11.5239
Epoch 00006: val_loss improved from 5.75985 to 5.10758, saving model to output₩lane_navigation_check.h5
300/300 [==============================] - 178s 593ms/step - loss: 11.5239 - val_loss: 5.1076
Epoch 7/10
300/300 [==============================] - ETA: 0s - loss: 10.7540
Epoch 00007: val_loss improved from 5.10758 to 3.28538, saving model to output₩lane_navigation_check.h5
300/300 [==============================] - 181s 603ms/step - loss: 10.7540 - val_loss: 3.2854
Epoch 8/10
300/300 [==============================] - ETA: 0s - loss: 9.8109
Epoch 00008: val_loss did not improve from 3.28538
300/300 [==============================] - 176s 588ms/step - loss: 9.8109 - val_loss: 3.9919
Epoch 9/10
300/300 [==============================] - ETA: 0s - loss: 9.1177
Epoch 00009: val_loss improved from 3.28538 to 2.75921, saving model to output₩lane_navigation_check.h5
300/300 [==============================] - 166s 554ms/step - loss: 9.1177 - val_loss: 2.7592
Epoch 10/10
300/300 [==============================] - ETA: 0s - loss: 8.5395
Epoch 00010: val_loss did not improve from 2.75921
300/300 [==============================] - 174s 579ms/step - loss: 8.5395 - val_loss: 2.9378
Deep learinig training finished!
```

딥러닝 차선 인식 주행

- PNG 이미지와 이미지에 기록된 차선 각도를 이용하여 데이터 라벨링한다.
- 총 3000번의 학습을 진행
- 이미지를 사용하여 정보를 정확히 파악하고 자동차의 조향 각도를 예측하기 위해 Nvidia모델 사용

```
def nvidia_model(self):
    model = Sequential(name='Nvidia_Model')
```

5. 프로젝트 수행 결과

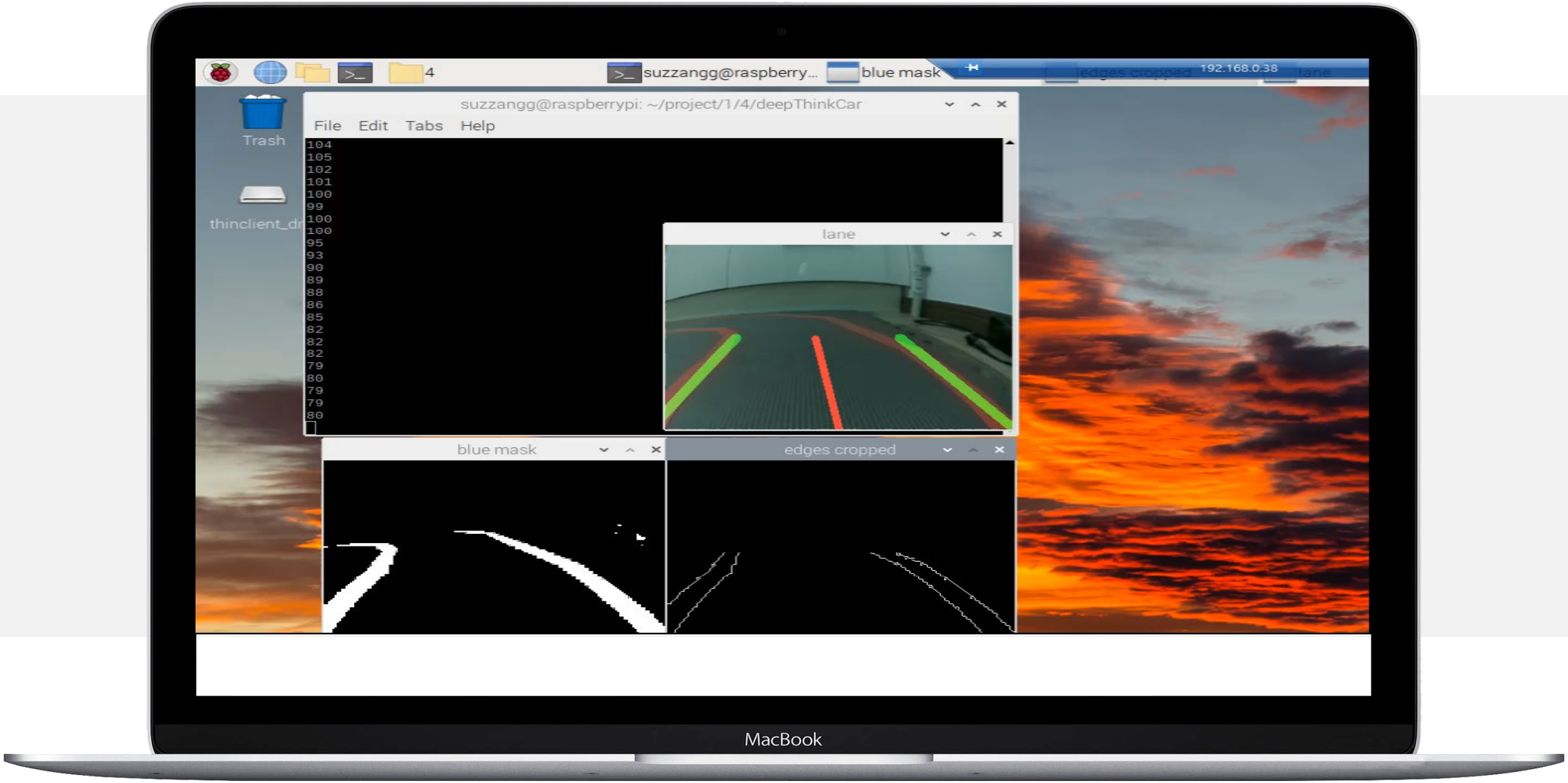


Nvidia Model

- Nvidia는 카메라로 입력받아 차량의 조향각도를 예측하여 출력하는 회귀모델이다.
- Nvidia모델에 사용된 CNN레이어는 총 30개층으로 구성되며 먼저 선과 가장자리를 추출한 뒤 마지막 신경 레이어를 통과하여 조향 각도를 예측한다.
- 예측 각도는 주어진 이미지에서 원하는 조향 각도와 비교되고 오류는 역전파를 통해 CNN훈련 프로세스로 피드백 된다.

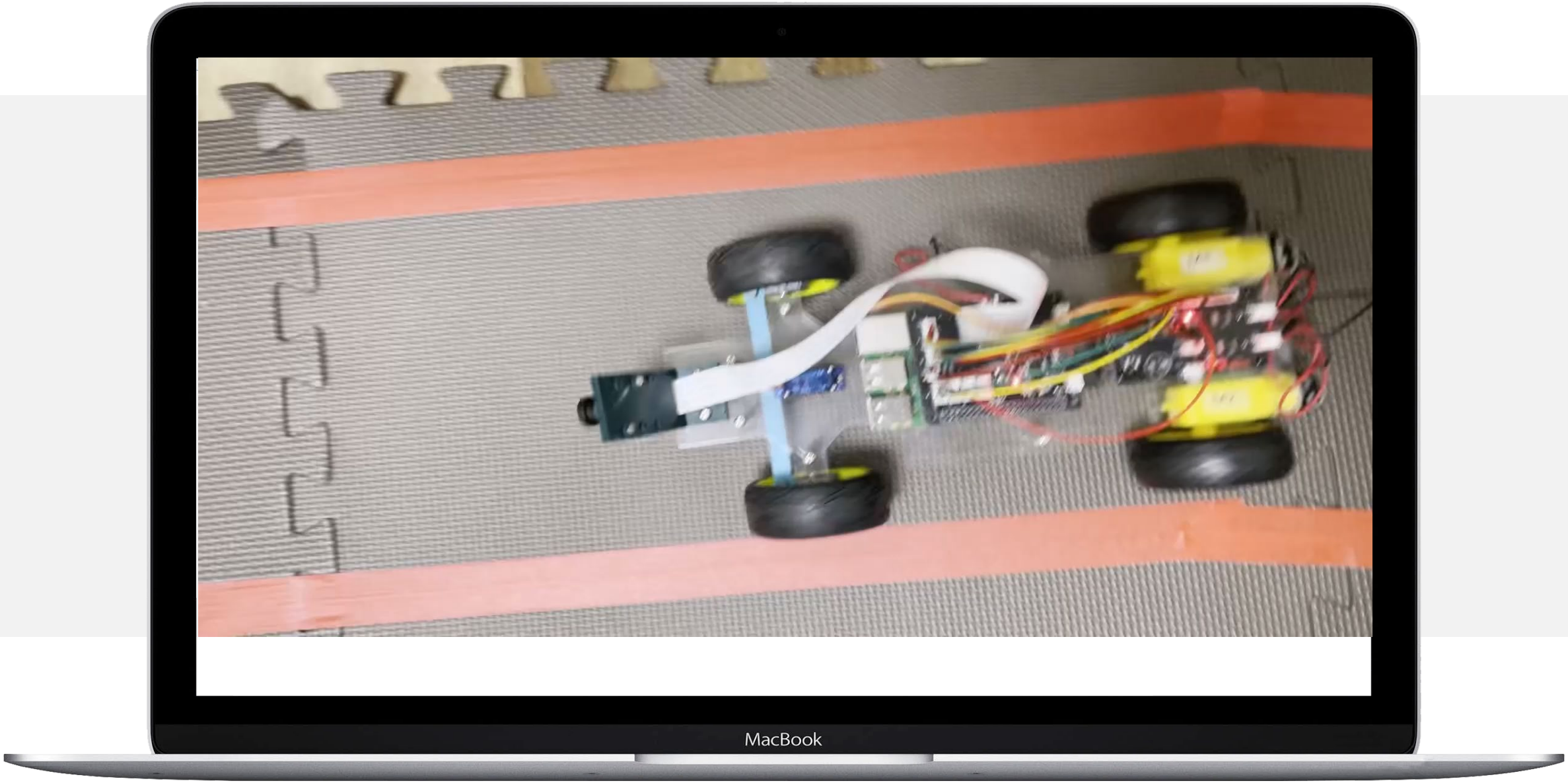
5. 프로젝트 수행 결과

시연 영상



5. 프로젝트 수행 결과

시연 영상



느낀점