글로벌시스템 융합과
2423016 배영민

캡스톤 디자인 (1)

프로젝트 상세 설명자료

영진전문대학교

0) 프로젝트 개요

목표 라즈베리파이 기반으로 RC카 기반 자율주행 시스템 구현

기간 **2024년 10월 6일** 시작하여, **24년 12월 15일** 까지

구현 특징

- **라즈베리파이4**를 활용한 모터 제어로 차량 주행
- 카메라 V2 모듈을 사용해 구간별 이미지 데이터 수집
- 차체 직접 설계하고 제작
- **CNN** 기반 모델을 적용하여 이미지에서 라인 패턴 학습

1) 하드웨어 - 라즈베리파이

컴퓨터는
라즈베리파이는

크게 범용 컴퓨터와 임베디드 컴퓨터로 나눌 수 있다.
특정 목적을 수행하기 위한 **Embedded Computer** 이다.

SBC, MCU, MPU란?

보드에 CPU, 메모리 등이 모두 포함된 **Single Board Com**
CPU와 메모리, I/O 기능이 통합된 단일 칩 **MicroController Unit**
CPU 역할만 수행, 입출력 장치가 별도로 구성 **MicroProcessor Unit**
시스템 구성이 복잡한 MPU가 처음 나오고, 현재 CPU의 대다수를 차지하는 MCU.
라즈베리파이는 **리눅스 기반의 초소형 범용 싱글 보드 컴퓨터**이다.

라즈베리파이와
젯슨 나노

처음 프로젝트를 시작할 때 **라즈베리파이**를 선택한 이유는 자율주행에 대한 여러 책을 참고하며 라즈베리파이가 **초보자에게 적합**하고, **GPIO 핀**을 활용해 하드웨어 제어도 비교적 간단하다고 판단했기 때문이다.
젯슨나노와 비교했을 때 **GPU 성능 부족** 등 **서보모터 제어시 튀는 현상** 등 여러 문제가 발생했지만, 라즈베리파이를 사용하면서 **하드웨어 제어와 시스템의 기초**를 배우며 많은 경험을 쌓을 수 있었다.

2) 라즈베리파이의 GPIO핀

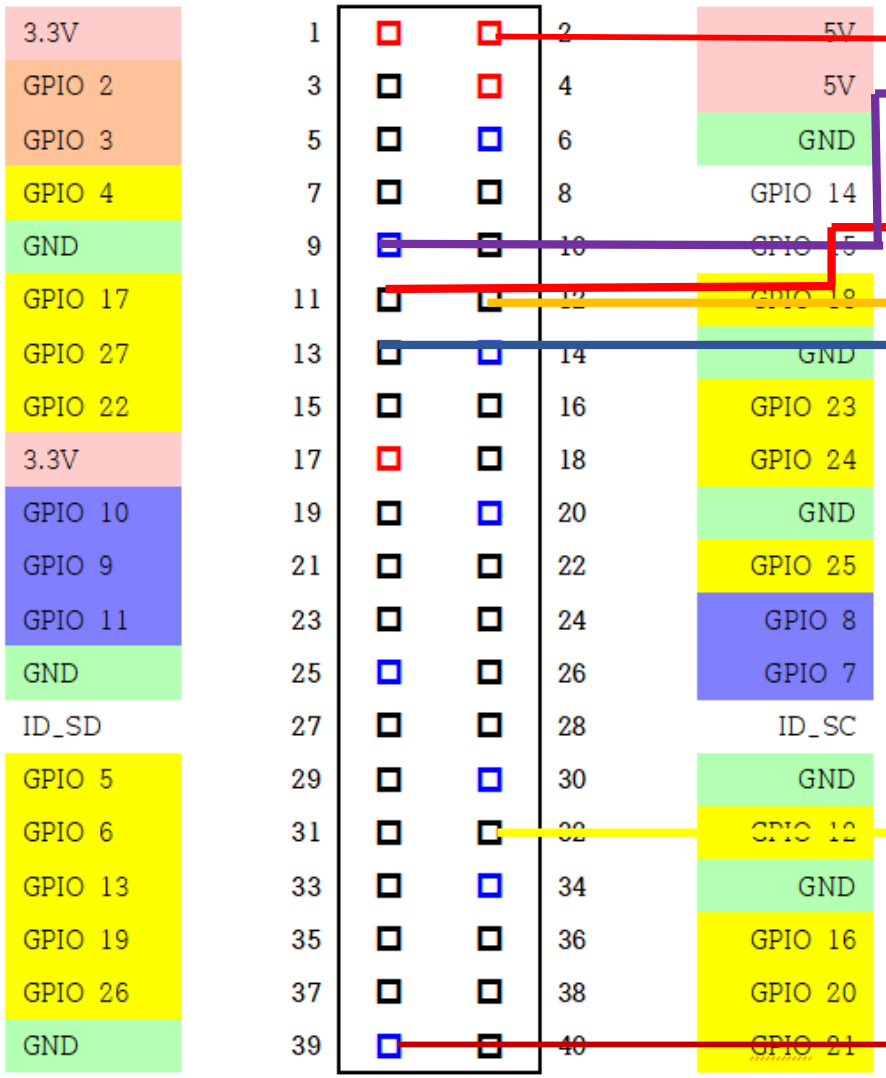
GPIO 핀이란 라즈베리파이와 전자적으로 통신하기위한 표준포트이다.

GPIO 핀 번호 체계 칩 번호 기반 핀 번호(BCM 모드)와 핀 배열 기준 (물리적)번호가 다르다. (코드 작성시 유의)

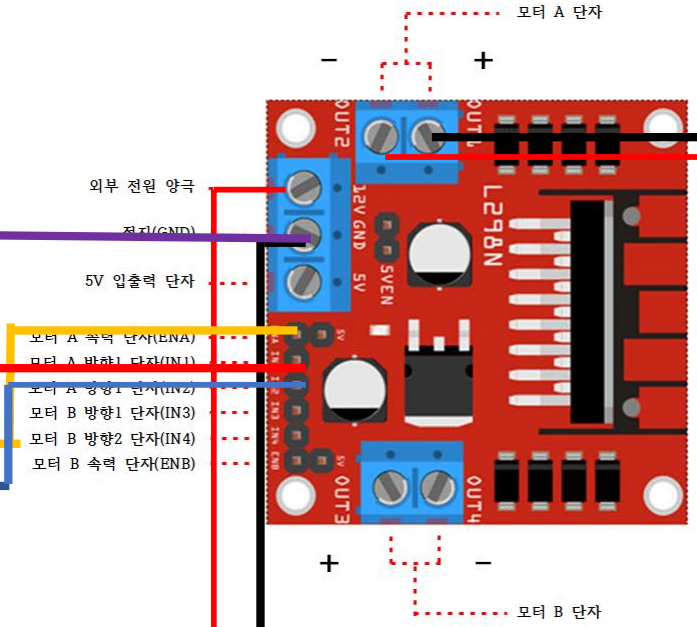
디지털 신호 GPIO가 사용하는 표준 전압은 3.3V이고, 입출력 되는 전압이 대략 1.7V 이상이면 1, 이하이면 0으로 인식하다. (이진 신호) GPIO핀은 이러한 신호를 통해 외부 장치와 통신한다.

아날로그 신호 라파는 기본적으로 아날로그 신호를 직접 출력하지 못한다. Pulse Width Modulation(PWM)을 사용하여 아날로그 신호를 흉내냄

3) 하드웨어 배선도



라즈베리파이 GPIO Pin Layout



L298N 구조

DC모터 12V

서보모터

3셀 리튬배터리

20000mAh 보조배터리
5V 3A
(라즈베리파이 전원)

3) GPIO핀을 통한 모터와 센서 제어

모터와 센서 제어

듀티 사이클(High 신호가 전체 주기에서 차지하는 비율)을 조정하여 PWM 신호의 세기를 변화시키고,
DC 모터의 속도(100Hz)와 서보모터의 각도(50Hz)를 조절한다.

L298N의 역할

1. H-Bridge 회로를 통해 DC모터의 회전방향을 변경한다.
2. L298N은 자체적으로 내장 전압 조정기가 있어서 GPIO에 적합한 전압으로 분배해준다. (점퍼 – 검정 커넥터 연결시 작동)

DC모터의 높은 전압

3셀 리튬배터리 (12~13V)가 L298N의 내장 전압 조정기가 활성화 되면 전압이 5V로 변환된다.

4) 소프트웨어 & 라이브러리

Python

라즈베리파이 환경과의 호환성이 뛰어나고, 친숙한 파이썬 언어를 사용함으로써 프로젝트에서 센서 제어, 이미지 처리, 데이터 훈련 등을 효과적으로 이해하고 구현할 수 있었다.

OpenCV

실시간 컴퓨터 비전을 처리하기 위해 만들어진 크로스 플랫폼(여러 운영체제에서 작동)

라이브러리로 리눅스에서 오픈소스로 사용하였다.

&

openCV에서 데이터 훈련 중 이미지 전처리 과정에서 가우시안 블러를 사용하여 선형 검출을 명확하게 하였다.

TensorFlow

텐서플로우는 구글에서 개발한 머신러닝 라이브러리로, 프로젝트에서는 데이터를 학습시켜 모델을 생성하고 예측하는 역할을 수행했다.

Libcamera

카메라 모듈을 제어하기 위해 사용된 라이브러리로, OpenCV 구동테스트, 데이터 수집, 데이터 훈련 과정에서 사용하며 프로젝트에서 핵심적인 영상 처리 도구로서 사용되었다.

5) 개발환경 구축

블로그에 원격접속하는 방법과 가상환경 설정까지 정리하였습니다.

프로젝트 개발의 기반 마련을 위해

개발환경 구축 단계로 OS설치 -> 원격 접속 -> 가상 환경 -> 라이브러리 설치로 나누었다.

리눅스 기반의 OS를 SD카드에 설치 후, 터미널에서 환경설정에 들어가 SSH접속을 켜주었다. PC에서 putty 프로그램을 원격으로 명령어를 입력하고 작업할 수 있게 하였고, VNC viewer 프로그램으로 GUI환경해서 실시간으로 화면을 보면서 작업을 할 수 있었다.

여기서

유의할 점으로 필요한 포트로 확인하는 것이다. 방화벽이 설정 되어있을 경우, 해당 포트를 허용을 해주어야 원활한 통신이 가능하고, SSH는 22번 VNC는 5900번 포트를 사용하였다.

가상환경 설정하여

독립적으로 라이브러리를 관리함으로써 의존성 충돌문제(여러 라이브러리가 서로 다른 버전을 요구)를 해결할 수 있었다. 가상환경 폴더를 생성 후 라이브러리를 설치하고 관리했다.

이후,

리눅스에서 apt라는 패키지 설치 방식으로 가상환경에서 의존성 문제를 해결하였고 효율적으로 라이브러리를 관리하였다.

이와 같이, 개발 환경 구축을 통해 프로젝트의 기반을 탄탄히 마련하였으며,
시스템 초기화나 보드 교체 시에도 원활하게 작업을 이어갈 수 있었다.

6) 하드웨어 제어 <서보모터>

서보모터의 각도는 데이터 수집하는데 있어서 가장 중요한 요소이다.

서보모터의 초기 각도는 90도로 설정하였고, 이를 기준으로 왼쪽 최대 각도는 0도, 오른쪽 최대 각도는 180도로 설정하였다.

그러나 **초기 모터 테스트** 과정에서 예상과 다르게 주행이 원활하게 이루어지지 않았다.

코드에서 아쉬운 점은 최대각과 최소값을 설정하지 않은점이다.

[서보모터의 동작 범위를 **10도 ≤ 각도 ≤ 170도**와 같이 제한하지 않았기 때문에 서보모터가 **비정상적인 각도**까지 회전하는 문제가 발생]

두번째로 갑작스러운 각도 변경 및 움직이지 않는 현상 해결을 위해 슬립시간을 설정하였는데,

서보모터가 각도를 부드럽게 조정하며, 조금이나마 급격한 동작을 방지할 수 있었다.

하지만 서보모터의 각도 제어만으로는 특정 구간에서 차량이 멈추거나 원활하게 주행하지 못하는 문제가 발생하였고, **코스 구간에서 속도를 높여 해결**할 수 있는 방법을 충분히 고려하지 못했다.

7) 데이터 수집

데이터 수집은

아래 코드(보기1)와 같이 단순하고 직관적으로 구현하였다.
차량의 조향 각도에 따라 이미지를 세 개의 폴더로 분류하여(보기2)
정면 주행의 정확도를 높이는 것을 목표로 하였다.

(보기1)

```
if keyValue == ord('q'):  # 'q'를 누르면 종료
    break
elif keyValue == 82:  # 위쪽 방향키: 전진
    print("go")
    carState = "go"
    motor_go(speedSet)
elif keyValue == 84:  # 아래쪽 방향키: 정지
    print("stop")
    carState = "stop"
    motor_stop()
elif keyValue == 81:  # 왼쪽 방향키: 좌회전
    print("left")
    carState = "left"
    motor_left(speedSet)
elif keyValue == 83:  # 오른쪽 방향키: 우회전
    print("right")
    carState = "right"
    motor_right(speedSet)
```

데이터 과적합

수집된 이미지의 비율을 확인한 결과, 왼쪽 회전 구간이 많은 코스
특성상 왼쪽 이미지가 과다하게 저장되었고, 데이터 과적합이 발생.

데이터 균형화

데이터 균형을 맞추기 위해 정면과 우측 구간에 대해 추가적인 데이터
수집을 진행하였고, **왼쪽 800**장, **정면 1000**장, **오른쪽 700장**으로 정제하였다.

결론

카메라 v2 모델은 기본적으로 초점이 과하게 확대되어 고정되어 있어서, 카메라 위치를
뒤로 조정하였으나, 여전히 **질적인 데이터 확보**에는 한계가 있었다.
장비의 제약으로 기존에 정제된 2500장의 이미지 데이터 셋을 기반으로 훈련을
마무리 하였다.

(보기2)

```
angle_folders = {
    "left": range(0, 85),   # 0~84도는 left 폴더
    "straight": range(85, 96),   # 85~95도는 straight 폴더
    "right": range(96, 181)   # 96~180도는 right 폴더
```

8) 데이터 전처리 과정

전처리 과정은

모델의 학습의 성능과 정확도를 높이기 위한 필수적인 단계이다.
내가 선택한 전처리 과정은 노이즈제거와 라인을 강조하기 위한 **Grayscale** (흑백 변환) 적용,
상단 30% 삭제 함으로써 불필요한 정보는 제거하고, 이미지 크기를 64*64 픽셀로 **리사이즈** 하였다.

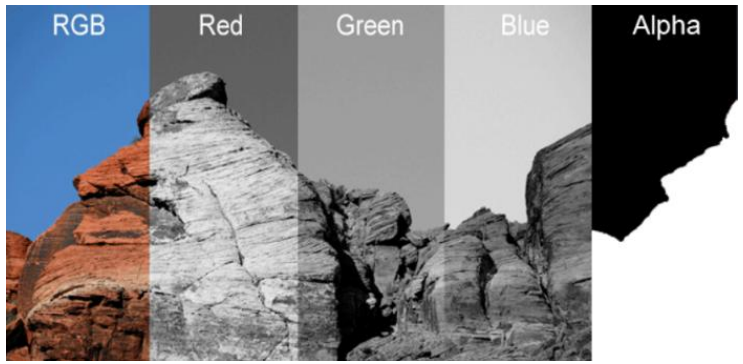
Grayscale

보기(1)과 같이 이미지는 기본적으로 **RGB 3채널**로 이루어져 있다. 각 픽셀은 이 세가지 값(0~255)을 통해 색상이 표현된다. 하지만, 그레이스케일로 변환을 하면 1채널로 연산 효율을 높이고 라인인식에 유리할거라는 판단 하에 진행하였고, 문제가 발생하였다.

노이즈 발생

조명의 영향으로 수 장의 이미지에서 불규칙한 명암 노이즈 보기(2) 가 발생하였다. 데이터 정제 과정 중 상당 수의 데이터를 삭제했지만, 그러나 **명암 대비에만 의존**하는 전처리 방식은 한계가 있었으며, 이를 통해 **전처리 과정의 다양성과 개선 필요성**에 대해 고민하게 되었다.

(보기1)



(보기2)

9) 데이터 모델 생성

CNN 모델구조

합성곱 신경망을 사용하여 이미지 데이터에서 라인 패턴을 학습하도록 하였다.
입력 데이터는 전처리한 **64*64 사이즈의 그레이스케일 이미지**이다.

단계별로 나누면

Conv2D 코드에서 이미지의 특징 (경계선, 라인 등)을 감지하고, **활성화 함수**로 **복잡한 패턴을 학습**.
MaxPooling2D 코드에서는 특징 맵의 크기를 줄여 연산량을 감소 및 **과적합을 방지.**
Dropout은 **일부 뉴런**을 무작위로 비활성화하여 과적합을 방지.
Flatten & Dense 코드에서는 위의 과정을 거친 데이터를 1차원 벡터로 변환 후(Flatten),
최종적으로 **조향 각도를 예측**(Dense)

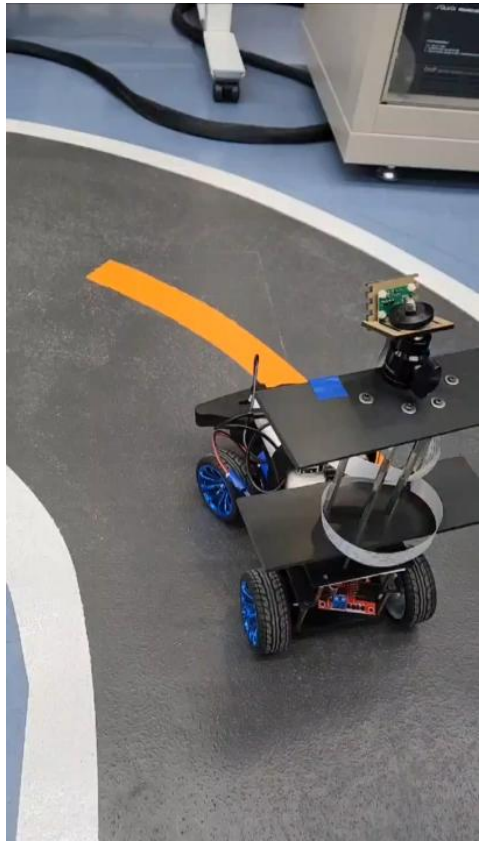
훈련과정

훈련데이터(70%)와 **검증 데이터**(30%)로 분리하여 학습과 평가를 진행하였다.
Epoch는 총 10회, Batch Size는 32로 설정하여 모델이 점진적으로 데이터를 학습하도록 하였다.

테스트 데이터

testdata는 모델의 최종 성능을 확인하는 중요한 데이터이다. 하지만 (5번의 시도 끝에 만든)
모델의 경우 자율 주행 성능이 충분히 배포할 만한 수준에 도달하지 못했다고 판단하여,
분류와 사용을 하지 않았다.

10) RC카 주행과 느낀점



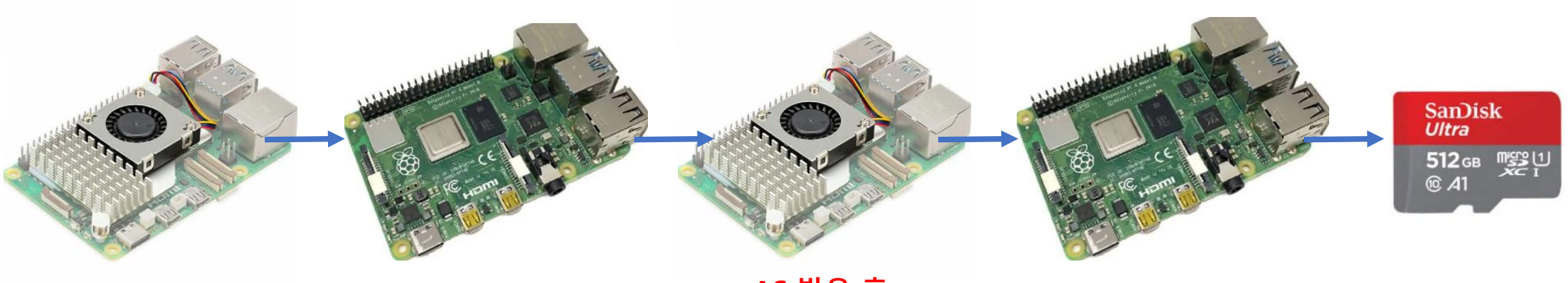
마지막으로 주행 영상을 통해 모델의 실제 성능을 확인하였다.
영상에서는 차량이 라인을 따라가려 하지만, **왼쪽으로 과도하게 꺾이는 현상**이 발생하였고 결국 **라인 검출에 실패**하는 모습이 나타났다.

카메라의 초점이 라인을 충분히 명확하게 표현하지 못했고, 데이터 수집 단계에서 발생한 **왼쪽 방향 데이터의 과적합**이 주요 원인이었다. 모델이 학습 과정에서 **왼쪽 패턴**에만 편향되었고, 정면과 오른쪽 데이터의 비율이 부족하여 제대로 반영되지 못했다.

이번 프로젝트를 통해 **철저한 사전조사를 통한 장비 선택의 중요함**과 데이터의 **불균형**과 **전처리 기법의 한계**가 모델 성능에 큰 영향을 미친다는 점을 확인할 수 있었다.
모델 성능을 최적화하지는 못했지만,

프로젝트를 마무리하며 **딥러닝 분야를 깊이 배우고 싶다**는 열망이 생겼다.

11) 문제해결 과정2)



초기 불량 → 4B로 교체 → AS 받은 후, 전원 쇼트 발생 → 4B로 교체 → 마감일 하루 전, SD카드 손상

두 번의 CPU 교체를 겪고 마지막 날에는 라즈베리파이 4B가 주행 중 **종종 꺼지는 현상**이 발생하였는데,

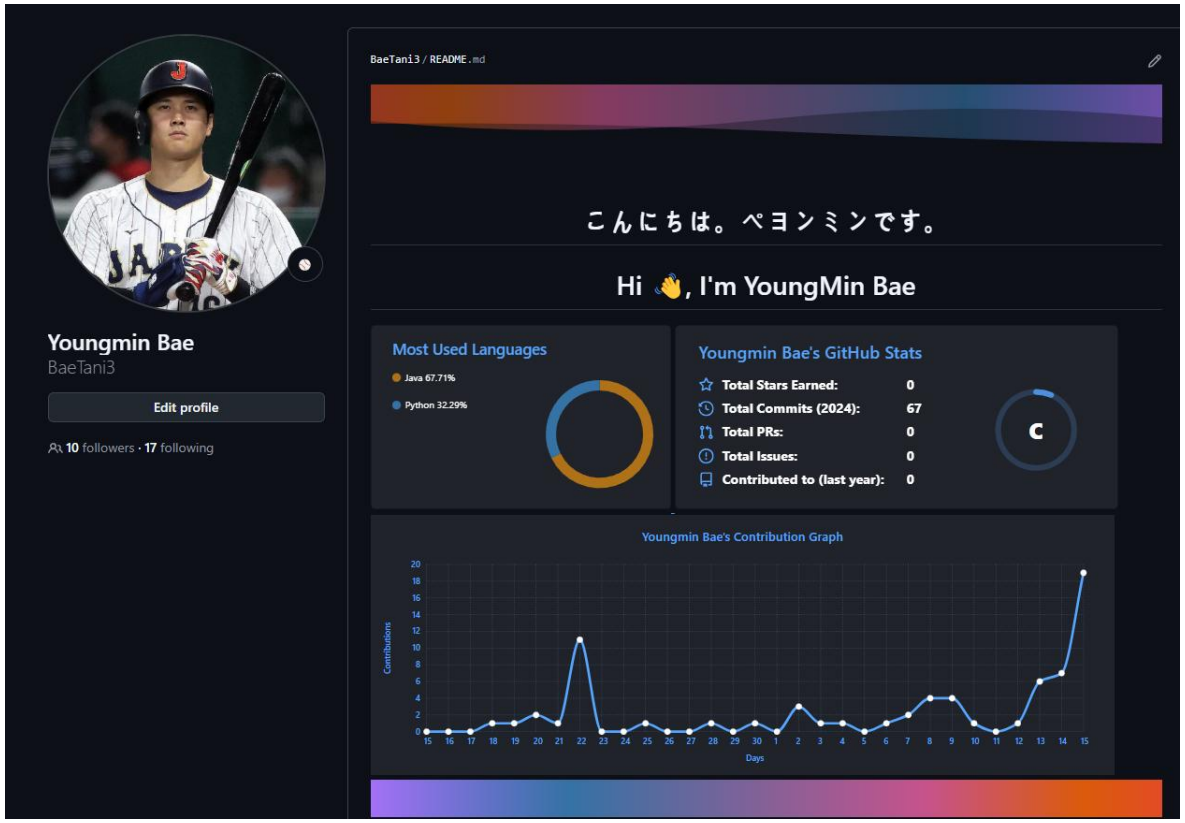
마감 하루 전에는 **SD카드 손상**까지 이어졌다. 시간 부족으로 팀원의 SD카드를 빌려 프로젝트를 진행하였다.

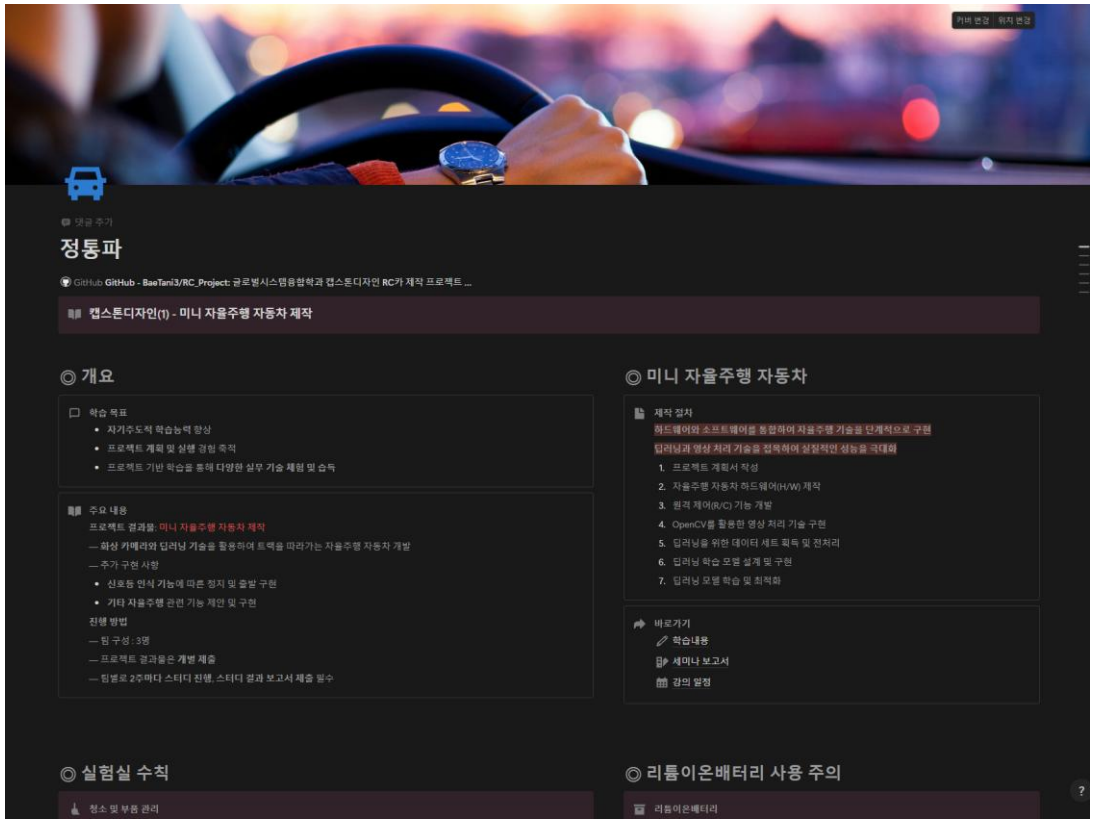
이 과정에서 장비를 소홀히 다룬 제 부주의도 있겠지만, 하나하나 문제를 해결하며 **멘탈을 단단하게 만들고**

엔지니어의 기본인 근성과 끈기를 키우는 소중한 훈련이 되었다.

기술적 한계를 극복하면서 **문제 해결 능력**을 키웠고, 팀원과의 **협력의 중요성** 또한 깨달을 수 있었다.

12) 깃허브 주소와 마무리





Ctrl + 클릭!

깃허브 RC_CAR

5조 <정통파> notion

감사합니다.