

PortFolio

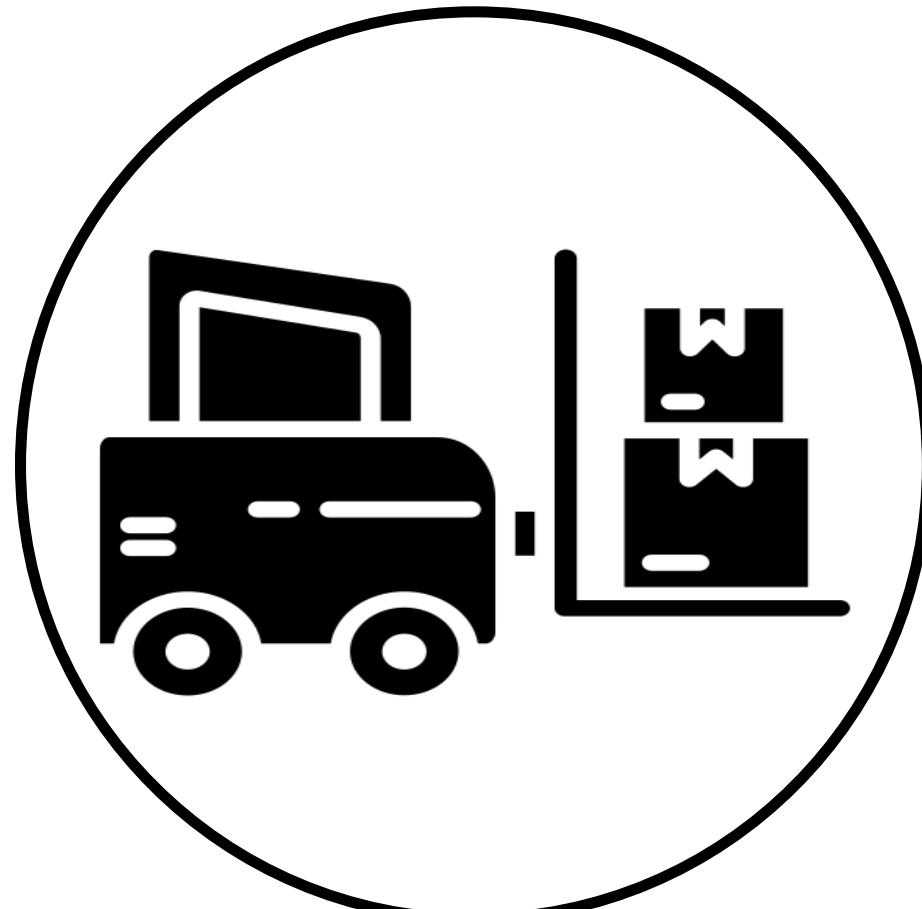
장명근

---

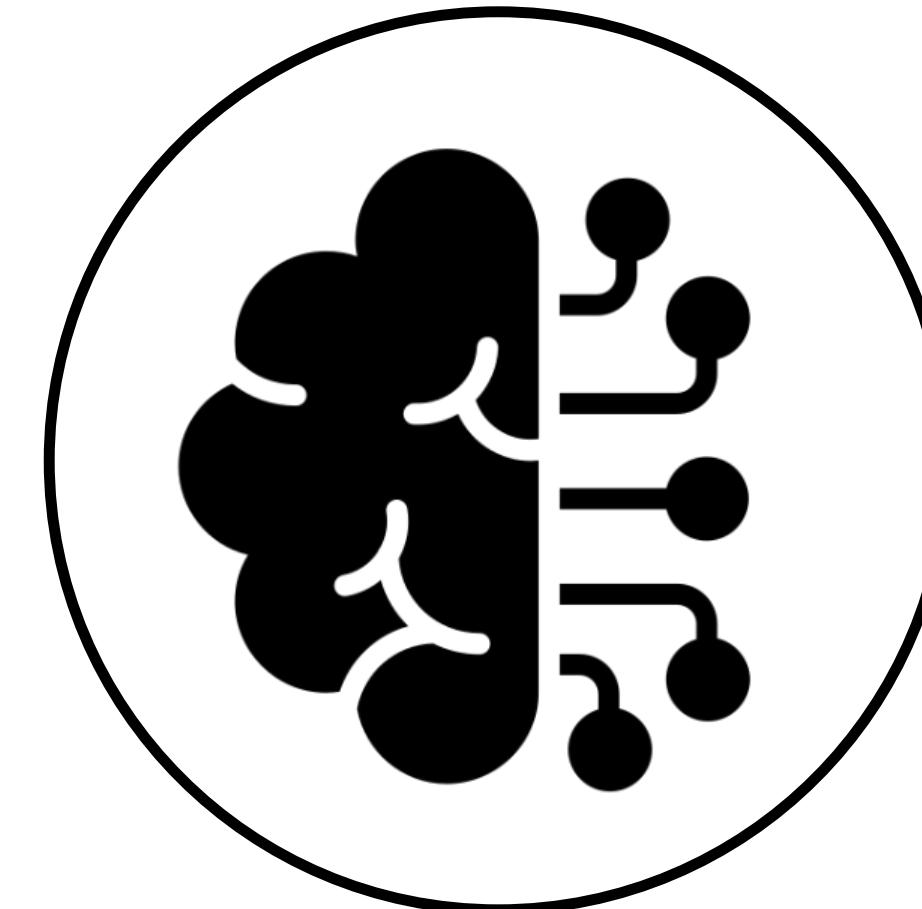
## 프로젝트 목차



자율주행 및  
홈 IoT 솔루션



스마트팩토리 솔루션



LLM 기반  
여행추천 APP

# 시뮬레이션 기반 자율 주행 SW개발

## 시뮬레이션 개요



- ROS 기반 자율주행 시뮬레이터
- 실제와 동일한 센서, 환경, 모델 제공
- 게임엔진 Unity를 사용하여 현실과 동일한 물리 환경적용
- 현재 현대그룹, 42dot등 다양한 차량 SW업체에서 사용 중인 SIM

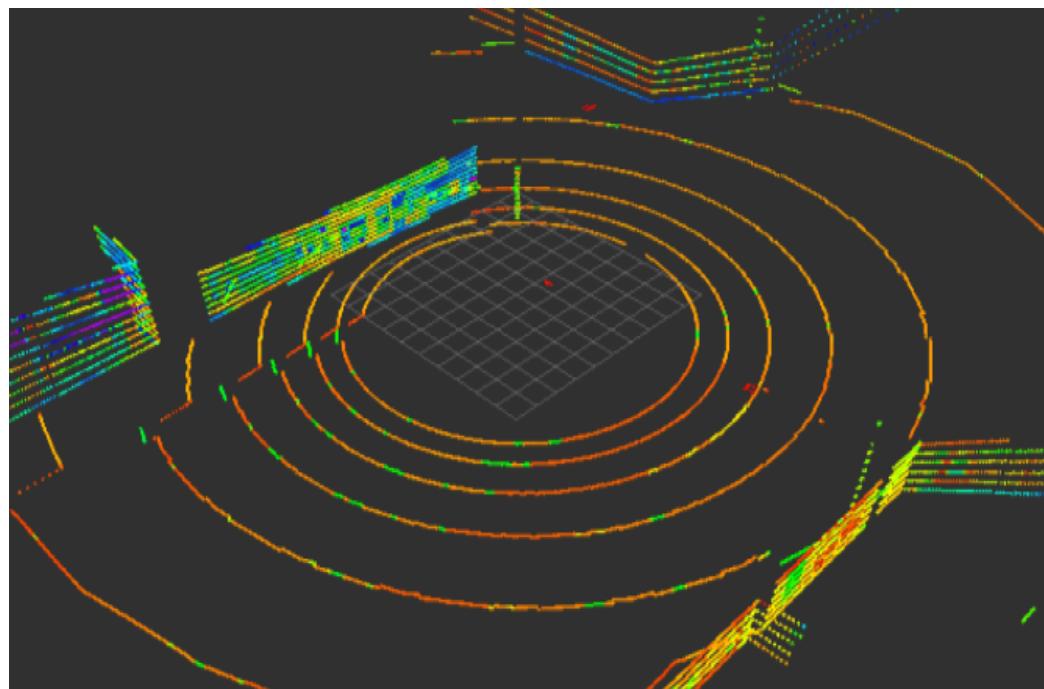
### 시뮬레이션 시스템 사양 요약-

OS: Window, Linux

CPU: Intel i7-10750H

RAM: 16GB, GPU: RTX 2070

## 시뮬레이션 SW 설계

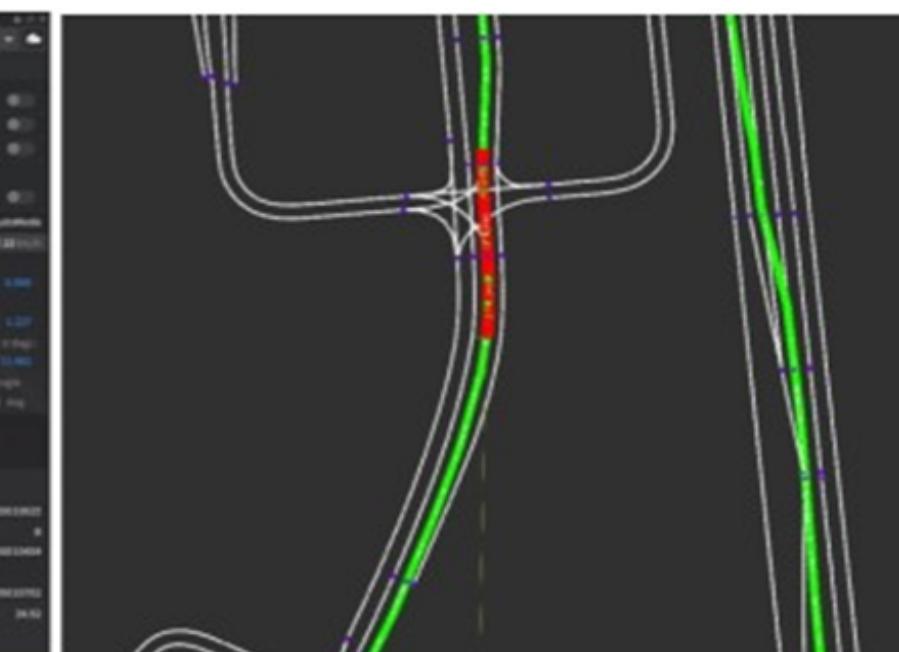
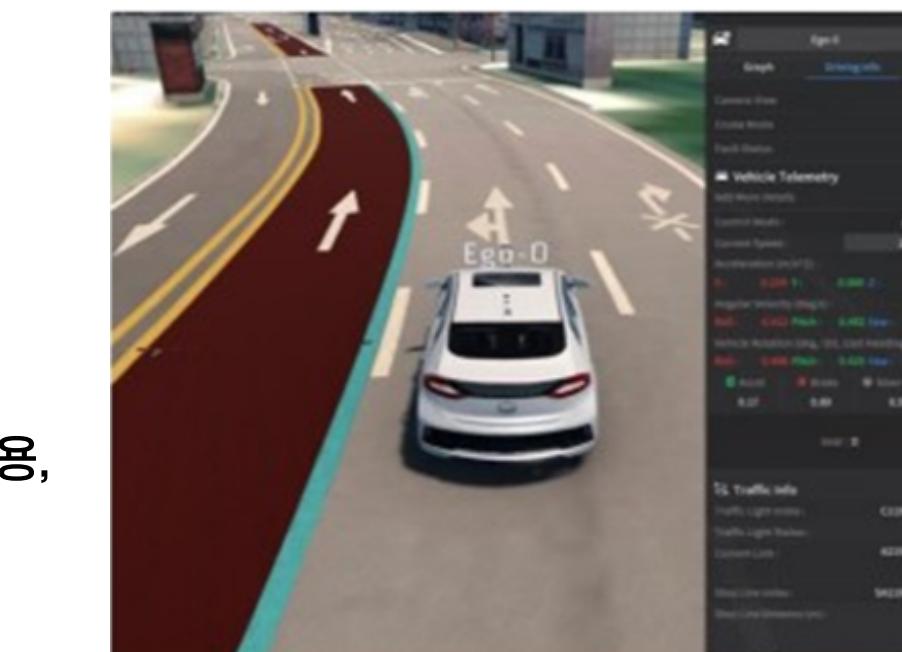


- 카메라, 라이다, 레이더 센서를 이용하여 주변 상황을 인지/판단
- V2X를 사용하여 신호등 및 도로 정보를 수집, 보다 안전한 자율 주행 구현
- 목적지를 입력 후 dijkstra 알고리즘을 활용, 최단 거리를 탐색

<LiDar 3D point Cloud 탐색 이미지>

## 프로젝트 명: 무인 자율주행 배송 차량 'LessGo'

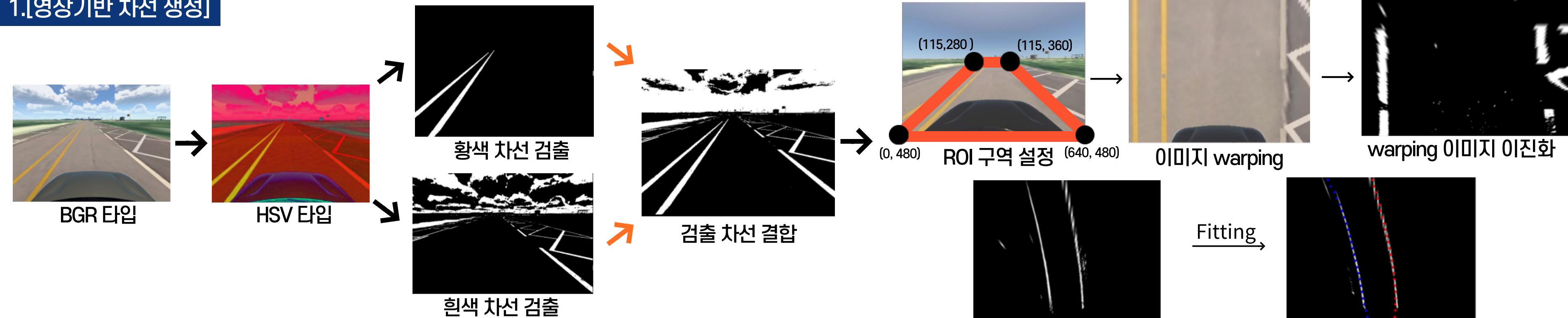
수행기간	2022.08.22~ 2022.10.07	참가인원	5명( 자율주행SW 4명, Web 1명 )
주관사	삼성전자 (SSAFY)		
담당업무 & 세부업무	<p>담당업무: 자율주행(영상인지, 차선유지)ADAS시스템 개발, 연동 모형 차량 제작(기여도 60 %)</p> <p>&gt; 기획: 배송 차량 선정, 서비스를 위한 기능 도출 및 구현</p> <p>&gt; 개발:</p> <ul style="list-style-type: none"><li>- OpenCv를 이용한 차선 검출</li><li>- PID제어, Pure persuit 주행 알고리즘 개발</li><li>- 시뮬레이터와 실제 연동되어 구동되는 Test차량 제작</li><li>- Ros 시각화 도구 Rviz를 통한 데이터 이미지화</li></ul>		
사용기술	<ul style="list-style-type: none"><li>- MORAI: ROS기반 자율주행 시뮬레이터. 카메라, 라이다, 레이더 등 ADAS시스템 센서 제공</li><li>- ROS: ROS통신 프로토콜을 이용하여 시뮬레이터와 우분투간 통신</li><li>- Ubuntu: 시뮬레이터에서 ROS를 통해 받아온 정보를 바탕으로 인지/판단/제어를 수행</li><li>- OpenCv: 차량의 카메라를 통해 들어온 정보 중 차선 추출시 사용</li><li>- Raspberry Pi: 시뮬레이터와 동일하게 작동하는 Embedded 차량 제작시 사용</li><li>- Mysql: 차량 주행 데이터를 수집하여 DB에 저장하기 위해 이용</li></ul>		



< 생성된 경로 및 자율주행 알고리즘에 따라 주행 중인 차량 이미지 >

# 자율 주행 - 영상처리 및 주행 Process

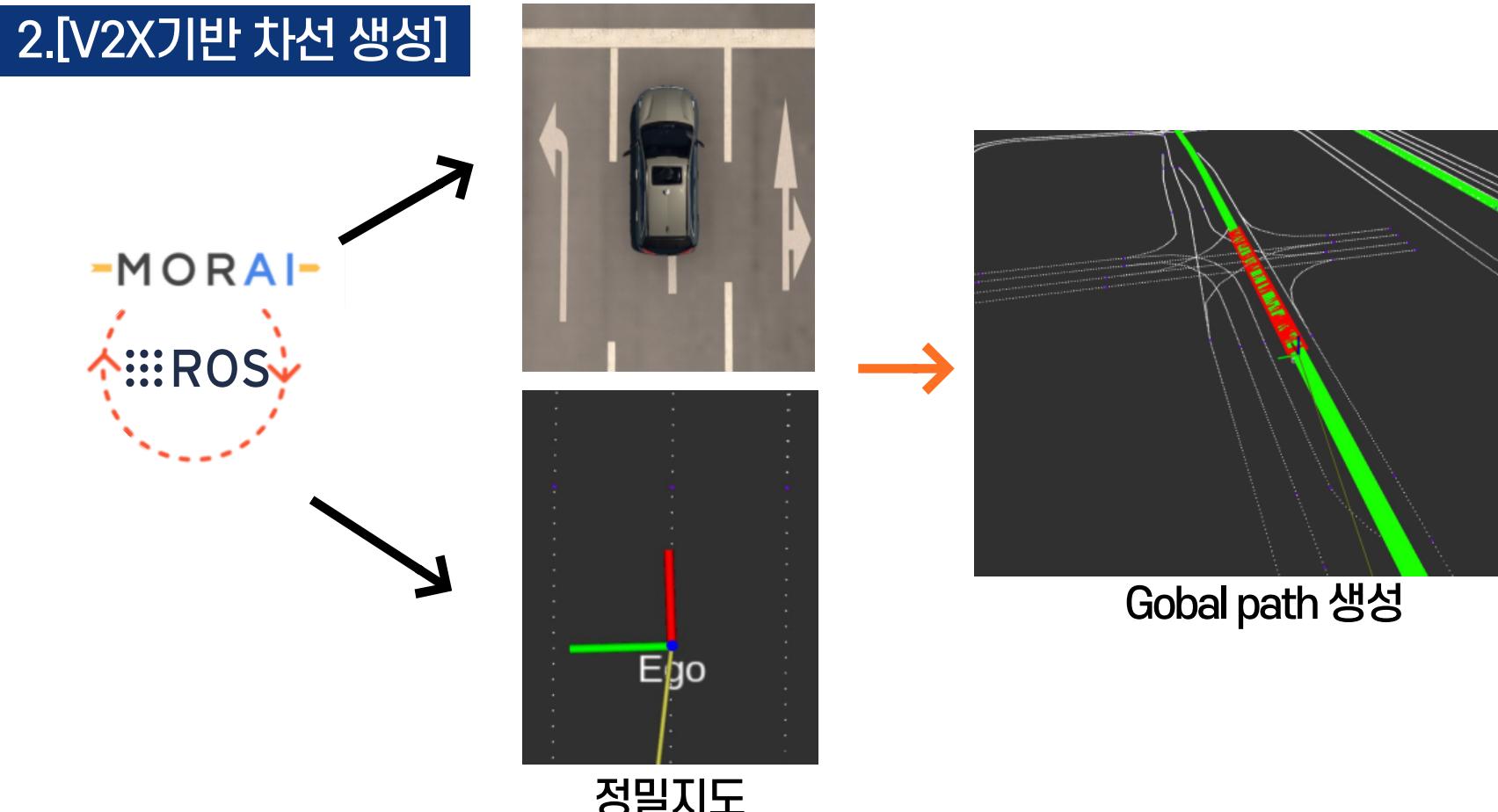
## 1.[영상기반 차선 생성]



- BGR 타입 데이터 이진화의 어려움 존재 -> HSV로 변환시 기준값을 설정하기 수월 = 이진화 정확도 상승
- 도로 차선의 핵심인 **황색**과 **흰색**을 0과 1로 이진화 후 검출하여 결합 -> 이미지에서 차선만을 추출

- ROI 영역을 설정하여 영역 외 부분은 모두 제거 후 영역 내 범위만 펴주는 작업(warping)을 합니다.
- warping 된 차선을 따라 **curve fitting** 을 하여 차선을 따라 node를 그립니다.

## 2.[V2X기반 차선 생성]



## 3.[차선 NODE 기반 주행]



# Embedded System Autonomous CAR

## HW System



IMU 차량제어



라이다 센서



NVidia Tx2  
메인 시스템



카메라 센서

- 시스템 사양 요약 -

CPU연산: 1.33 TFLOPS

RAM: 4GB , 클럭: 1.3 GHZ

## SW System



OpenCV  
영상처리



TensorFlow  
AI 머신러닝



Linux™  
리눅스 OS



ROS  
디바이스 제어

프로젝트 명: XyCAR 기반 차선인식 및 자율주행 프로젝트

수행기간

2022.10.31 ~ 2022.12.23

참가 인원

4명(자율주행SW 개발)

담당업무  
&  
세부업무

담당업무:

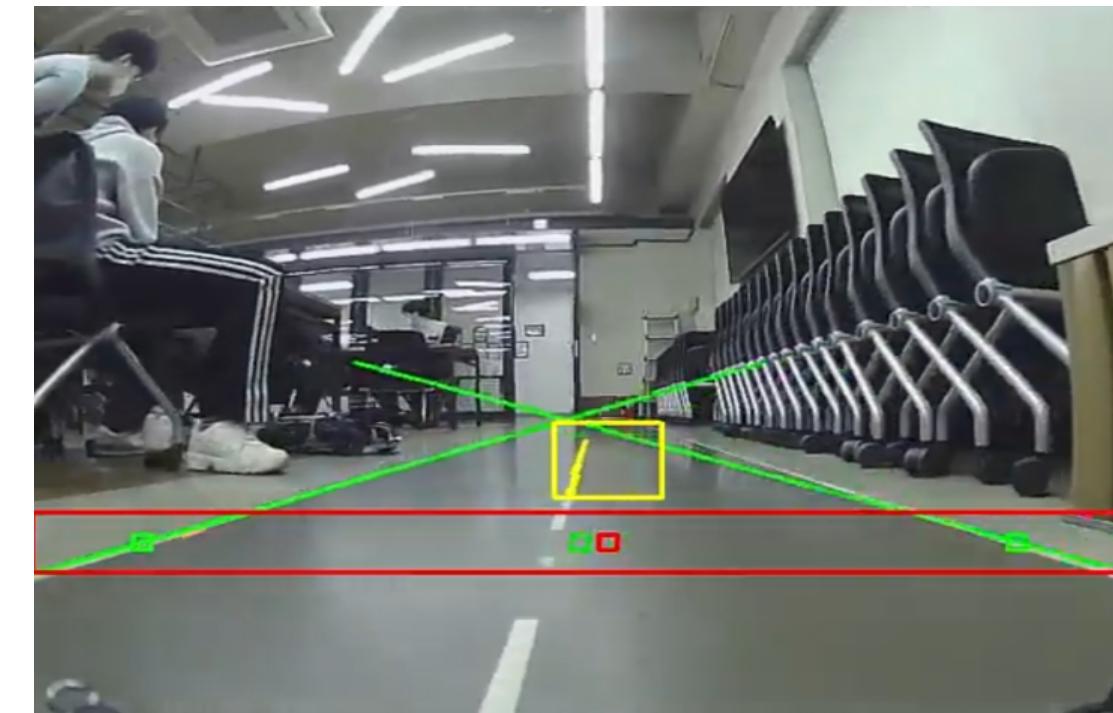
- > 개발:
  - 영상 오픈소스 라이브러리 OpenCv를 이용한 차선 검출
  - PID제어, Pure pursuit 알고리즘 실차 적용
  - 가상환경이 아닌 현실에서 차선을 인식할 경우 발생할 수 있는 문제점 해결
  - Nvidia Tx2 임베디드 시스템에 맞게 SW 알고리즘 최적화 진행

사용기술

- ROS: ROS통신 프로토콜을 이용하여 차량의 센서간 통신
- Ubuntu: 시뮬레이터에서 ROS통해 받아온 정보를 바탕으로 인지/판단/제어를 수행
- OpenCv: 차량의 카메라를 통해 들어온 정보 중 차선 추출시 사용
- Jetson TX2: 실습차량에 탑재된 중앙 임베디드 시스템
- Xycar: IMU, LIDAR, 초음파센서, 카메라 등이 탑재된 1/10 크기의 실습차량



< 자율 주행 Embedded System 차량 XyCar >



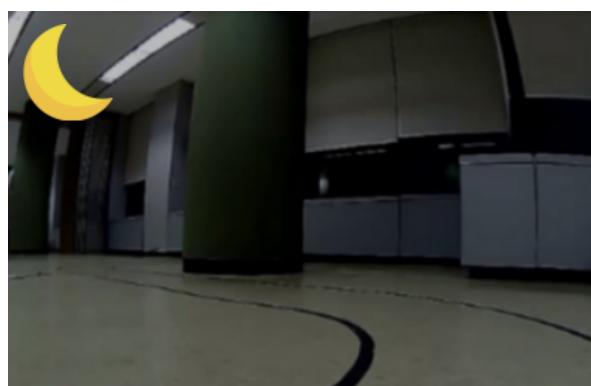
< 실제 모의 도로환경 차선 검출 이미지>

## 프로젝트 목표 및 Process

- 시뮬레이션에서 개발했던 ADAS 자율주행SW 시스템을 실제 임베디드 시스템 적용
- 가상환경에서 정상 작동하는 시스템이 실제 환경에서 발생하는 문제점 파악
- 이를 위해 다양한 TestCase를 생성하여 발생 문제점의 원인을 분석
- SW수정 및 최적화 후 다시 시스템 검증 Process를 정상 동작까지 반복 진행

# Nvidia Tx2 시스템 SW 적용 및 HW 검증

## 1. [시간별 조도 TestCase 분류]



실제 차량 운행 중 발생할 수 있는 환경의 가상 Case를 생성 및 구현하여 차선 검출 System01 정상 작동하는지 분석 및 검증을 진행

=> HSV의 범위를 기준에서 야간의 검출 정확성 향상을 위해 명도Value 의 범위를 조금 더 증가

변경 전: (20, 20, 100) ~ (36, 255, 150)

변경 후: (20, 20, 50) ~ (36, 255, 255)

## 2. [SW 알고리즘 최적화 진행]

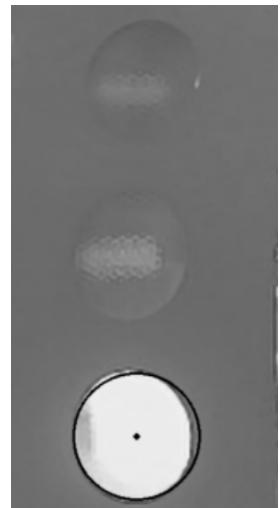
### 기존 문제점 분석

기존 고사양의 Window 환경에서 이용한 원 검출 알고리즘 Houghcircle가 Embedded 시스템에서 시스템 과부화를 발생. (약 5FPS) 안정성 보장 불가

```
void houghcircle_gary(Mat img_con)
{
    Mat img_copy;
    Mat img;
    img_con.copyTo(img_copy);
    img_con.copyTo(img);

    for (size_t i = 0; i < circles.size(); i++)
    {
        Vec3i c = circles[i];
        Point center(c[0], c[1]);
        int radius = c[2];
        rad_plus = radius * 1.2;
        cen_r = c[1];
        cen_c = c[0];

        circle(img_houghC, center, radius, Scalar(0, 255, 0), 2);
        circle(img_houghC, center, 2, Scalar(0, 0, 255), 3);
    }
}
```



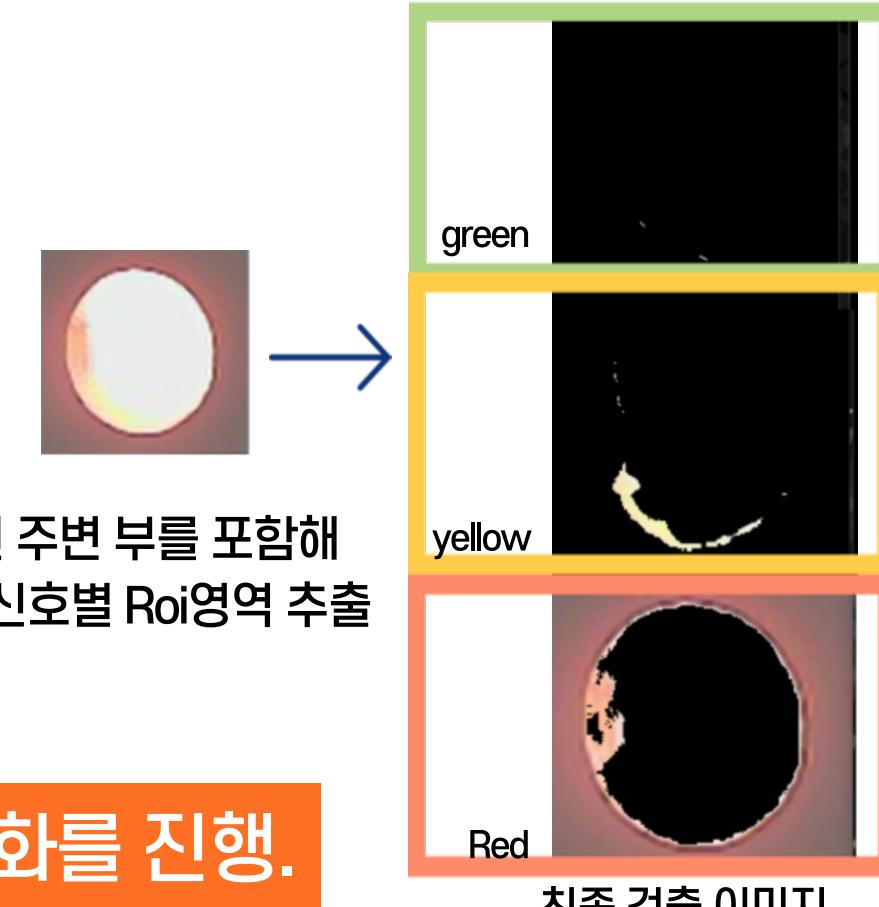
1. 이미지 이진화  
2. Houghcircle을  
통해 원 검출



```
int yellow_sum = 0, red_sum = 0;
// yellow 범위 추출
for (int row = 0; row < yellow_mask.rows; row++){
    for (int col = 0; col < yellow_mask.cols; col++){
        uchar b = yellow_mask.at<uchar>(row, col);
        yellow_sum += b;
    }
}
// red 범위 추출
for (int row = 0; row < red_mask.rows; row++){
    for (int col = 0; col < red_mask.cols; col++){
        uchar b = red_mask.at<uchar>(row, col);
        red_sum += b;
    }
}
waitKey(0);
```

### 해결 방안 모색

이를 해결하기 위해 신호등의 신호 판별시 원을 검출하는 것이 아닌 신호의 색상 분포도 검출을 통해 신호등의 신호를 판별하는 알고리즘을 개발. (약 30FPS 이상) 안정적인 신호 판별 가능



원 주변 부를 포함해  
각 신호별 Roi영역 추출

1.HW 시스템 사양을 분석 2. 기존 SW의 적용 및 검증 3. 문제 발견 4. 알고리즘 최적화를 진행.

최종 검출 이미지

# 4족 아이 케어 로봇 "보비" 개요



<아이케어 홈 IoT 시스템 로봇 "보비">

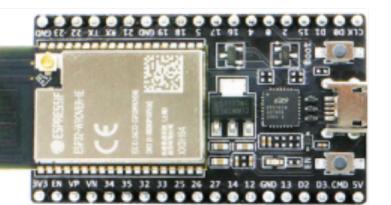
## - 시스템 사양 요약-

CPU연산: 0.015 TFLOPS  
RAM: 8GB , 클럭: 1.8 GHZ

## HW System



라즈베리파이4B  
메인시스템



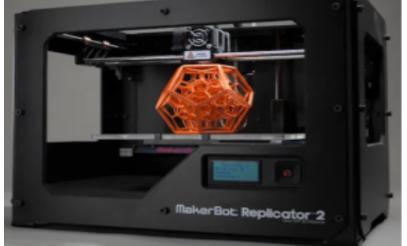
ESP32  
다리 모터 제어



감정 표현을 위한  
OLED Display



초음파 거리  
측정 센서



3D프린터  
외관 모델 출력



다리 구동을 위한  
Servo Motor \* 12

프로젝트 명: 아이 케어 로봇 '보비'			
수행기간	2022.07.05 ~ 2022.08.19	참가인원	6명( IoT 4명, front-end 1명, back-end 1명 )
주관사	삼성전자 (SSAFY)	성과	우수상 수상
담당업무 & 세부업무	담당업무: 팀장, 프로젝트 기획, IoT HW 및 SW개발(기여도:65%)  > 팀장: 팀원 선정, 프로젝트 역할 배분, Jira를 통한 프로젝트 일정관리 > 기획: 4족 보행 로봇 선정, 고객중심의 기능 도출 > 개발: - 아이를 스스로 인식하고 아이 이동시 자율이동을 통해 아이를 계속해서 팔로잉 - 상단에 탑재된 초음파 센서를 이용하여 아이와의 충돌을 방지 - 외관 모델링 직접 제작 후 3D프린터를 사용하여 제작 - DHT11을 이용한 온습도 측정, MQ2를 이용한 유해가스 등을 측정하여 보호자가 Web을 통해 실시간으로 아이 주변환경 확인		
사용기술	<ul style="list-style-type: none"><li>Raspberry Pi4: 영상처리 및 로봇의 연산 담당</li><li>ESP32: 12개의 모터 제어 및 RPI사이 UART통신</li><li>OpenCv, Yolo : 아이 탐색 및 추적을 위한 측정</li><li>AWS: DB 구축, 보비에서 측정된 데이터값을 Web으로 전송시 사용</li><li>MQTT: 로봇에서 측정된 센서값을 Mysql로 전송</li></ul>		

## SW System



데이터 저장을 위한  
Cloud 시스템



사용자 인터페이스 설계



딥러닝기반  
객체 탐색 알고리즘



IoT 통신 시스템



사용자 데이터  
저장 DB



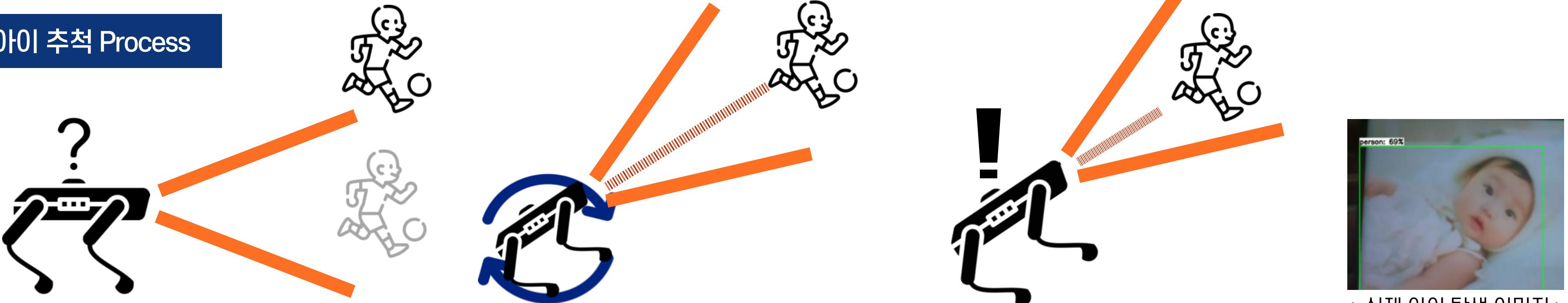
아이와 소통을 위한  
음성인식 시스템



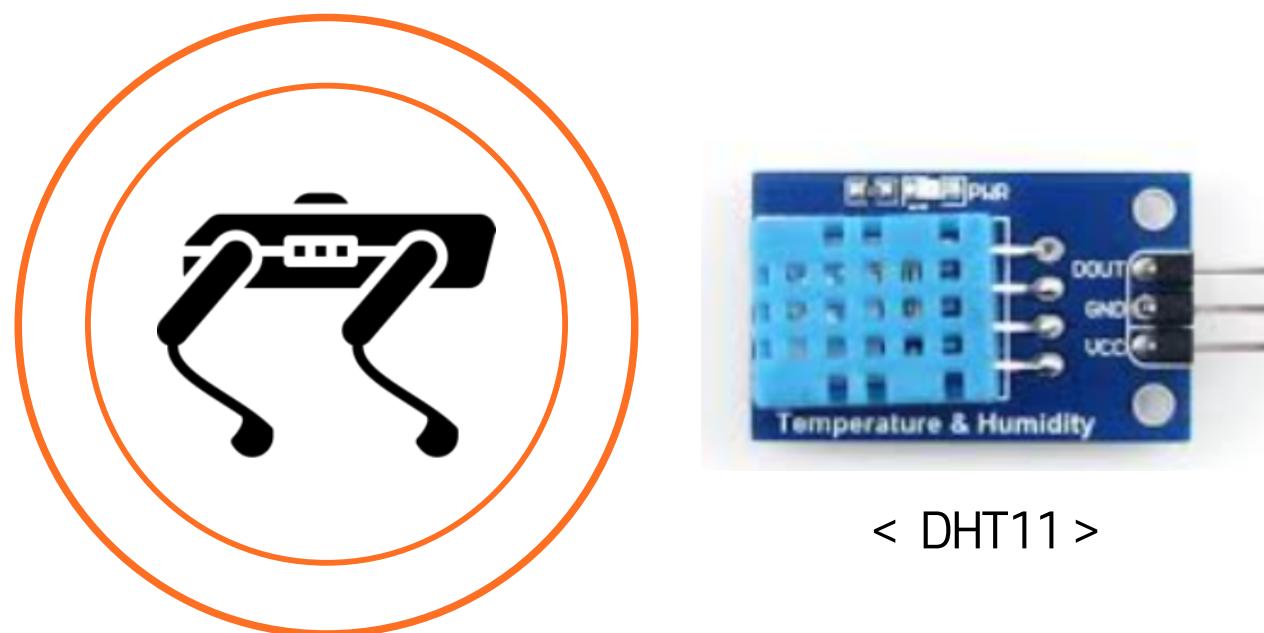
<팀 "BoBi" 팀원 사진>

# 4족 아이 케어 로봇 프로세스

## 1. 아이 추적 Process



## 1. 아이 주변 환경 분석



1. 아이의 환경을 DHT11 센서를 통해 실시간 데이터 수집

2. 수집된 주변 데이터를 AWS Cloud 서버DB Mysql에 기록

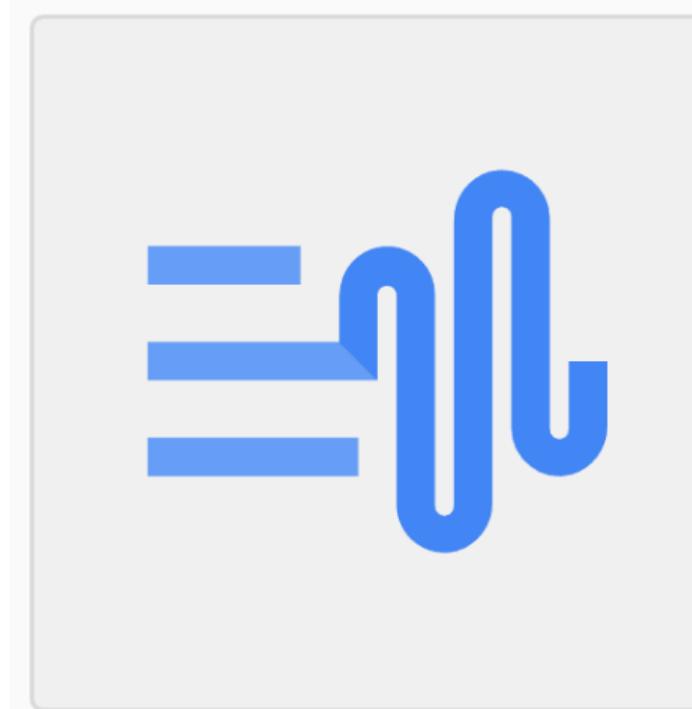
3. 수집된 데이터를 실시간으로 어플로 전송하여 모니터링 + 위험 감지시 알림 전송

# 4족 아이 케어 로봇 시스템 소개

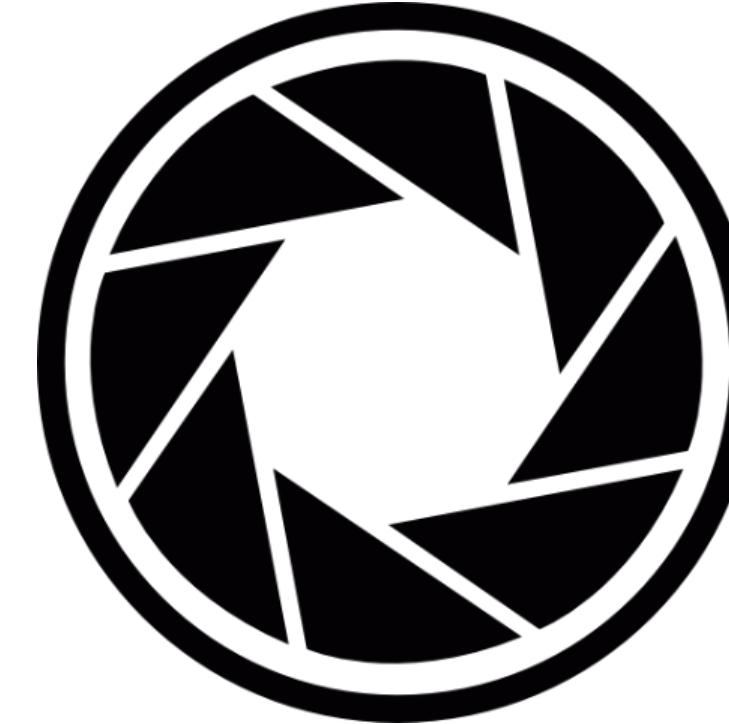


<WeaveGo>

- 로봇의 Base 골격 구조
- 기본적인 전후좌우 수동 조작 기능 탑재



- Google STT 기술을 활용
- 아이의 말을 Text로 변환
- 말을 알아 들고 로봇과 상호 작용



- 아이를 실시간으로 촬영
- OpenCv, haar cascade 이용, 아이를 추적 및 자율주행 통한 아이 팔로잉

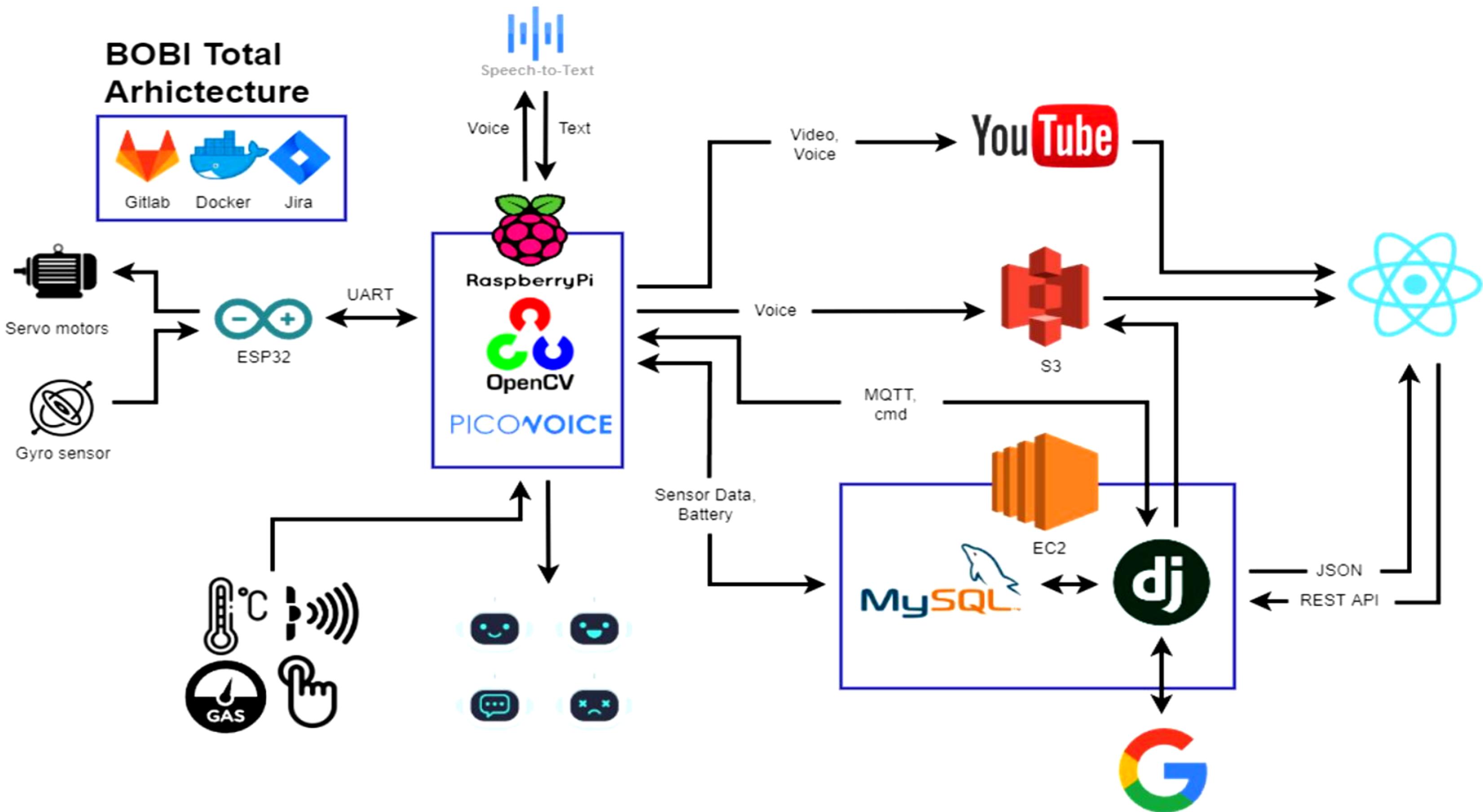


- WEB을 통해 보호자가 아이의 모습을 실시간으로 모니터링
- 그래프를 통해 아이의 주변 환경 상황을 확인



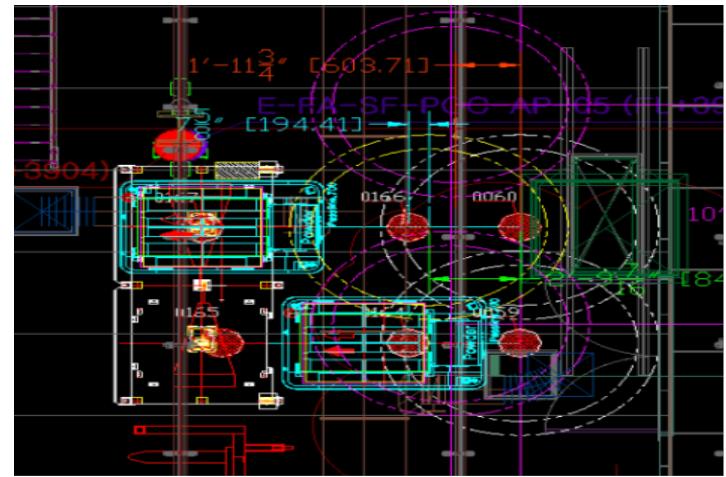
아이 케어 AI CCTV + 모니터링 어플리케이션 = 통합 솔루션 구축

# 4족 케어 로봇 - Total Architecture



# 스마트팩토리 시뮬레이션 검증

## AGV 시뮬레이션 및 실차 검증



< 공장 설계 예시 CAD Layout >



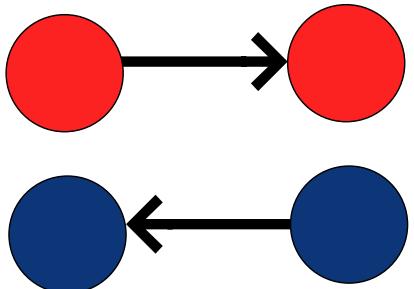
< 스마트팩토리 VD 공정 AGV 로봇 >



Node 및 Link  
자동 추출



각 공정별 AGV  
HW 스펙데이터 시트 분석



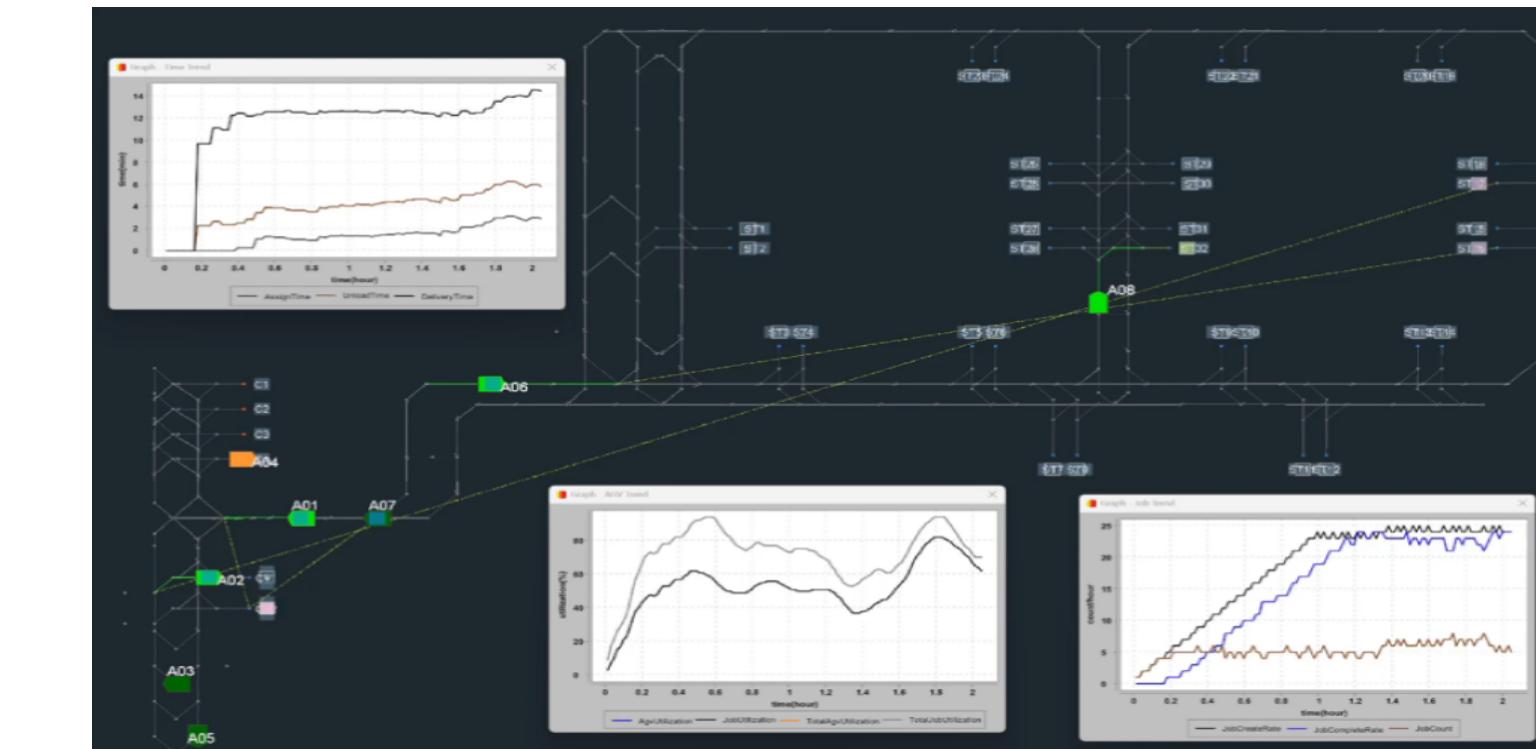
- 기본 속도, 코너 속도
- 물체 적재시 속도
- 이적 시간, 적재 시간
- 충전시간, 구동시간....



Test 시나리오 설계

1. AGV 시작 위치를 서로 다르게 하여 이상 발생 유무 검증
2. 한 동선에 많은 수의 AGV가 병목현상이 일어난 경우 검증  
→ 특히 양방향 2개의 동선 보다 단방향 1개의 동선인 경우 집중적으로 분석
3. 정상적으로 24시간 이상 구동시 평균 AGV들의 가동률 측정  
→ 고객사의 요청에 따라 AGV대수를 결정

프로젝트 명: ANT 프로젝트			
수행기간	2023.05.01 ~ 2023.11.15	참가인원	12명(SW개발 시뮬레이션 팀)
근무지	(주)티로보틱스	고객사	(주) SK ON
담당업무 & 세부업무	담당업무: 스마트팩토리 시뮬레이션 검증, AGV 제어기 개발, AGV 케일리브레이션 > 개발: - C++기반 AGV 제어기 솔루션 개발 - 스마트팩토리 AGV 동선설계 및 시나리오 TestCase 검증(가동률 70% 이하 설계) - CAD 공장 Layout 영상처리를 통해 시뮬레이션 소스 코드로 변환(node, Link) - SK on에 스마트팩토리에 최적화된 동선 설계 - 스마트팩토리 모니터링 프로그램 제작		
사용기술	- OpenCV: CAD 도면을 영상처리를 활용하여 코드로 변환 - C++: 제어기 시스템 언어 - React: 데이터 모니터링		



< 노드와 링크가 연결된 시뮬레이션 구동 예시 이미지 >

# LLM 기반 여행지 추천 어플리케이션



## 어플리케이션 동작 Process

2박 3일 친구들 끼리갈  
부산 여행지 추천해줘  
나는 특히 바다가 보고싶어

Prompt 입력



ChatGPT

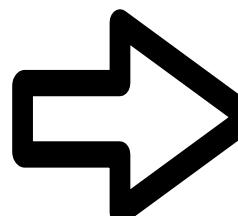
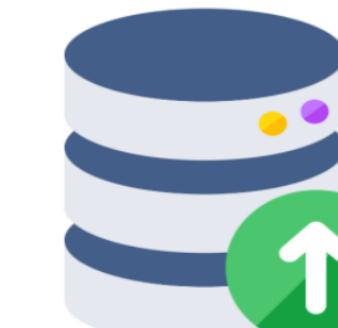


사용자 대화에서

Token(말뭉치) 추출

프로젝트 명: LLM 기반 여행지 추천 어플리케이션			
수행기간	2024.06.01 ~ 2024.08.30	참가인원	4명 (AI, Front-end, Back-end)
담당업무 & 세부업무	<p>담당업무: 프로젝트 기획, Back-end, AI (기여도 55%)</p> <p>&gt; 기획: 여행지 추천 방식 개발</p> <p>&gt; 개발:</p> <ul style="list-style-type: none"><li>- Spring MVC 패턴 CRUD 게시판 구축, Mysql DB 설계</li><li>- Chat GPT API를 사용하여 LLM으로 여행지 추천</li><li>- 각 기능과 LLM 성능에 맞게 Prompt Engineering 설계</li><li>- Image to Text (Blip)모델 여행지에 맞게 FineTuning을 통해 정확한 여행지 분석 가능</li></ul>		
사용기술	<ul style="list-style-type: none"><li>- Spring: Java 웹 프레임 워크</li><li>- Mysql: DB 설계</li><li>- AWS: EC2, S3를 사용하여 서버 및 데이터베이스 구축</li><li>- ChatGPT: 챗 API를 사용하여 사용자와 LLM으로 대화 구축</li><li>- BLIP: 사용자가 저장한 이미지를 분석하여 여행 내용 Text화</li><li>- llama3.1: 분석된 이미지와 여행지 설명데이터를 결합하여 여행 후기 자동 생성</li></ul>		

Vector DB



DB에 저장해둔 (관광지, 숙박, 시당)  
CSV 데이터를 기반으로 여행지 추천

네 여행지를 추천해 드릴게요

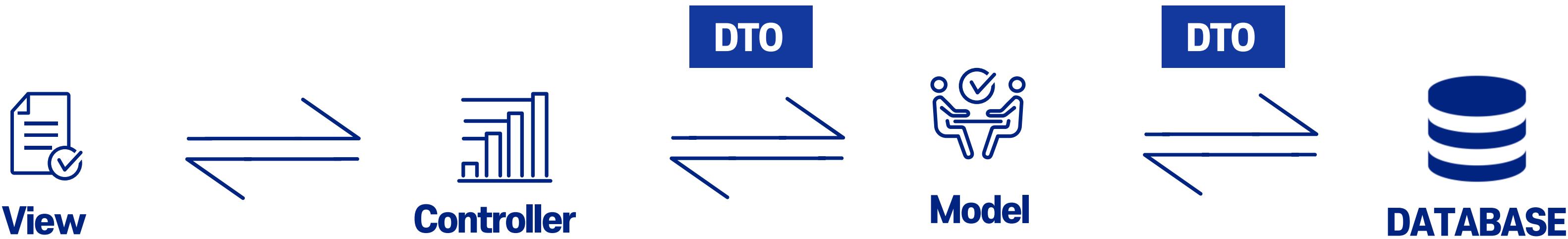
1일차:

- 아침: 해운대의 더베이 101에서 해산물 요리
- 관광: 광안대교 산책 및 사진 촬영
- 점심: 감천문화마을 근처의 전통 한정식 5
- 관광: 자갈치 시장 방문 및 부산타워 전망대
- 저녁: 자갈치 시장에서 회덮밥..

.....

# MVC 패턴 설계

- MVC 패턴이란 Model, View, Controller의 약자로 각각의 역할을 명확히 구별하여 역할을 효율적으로 할 수 있도록하는 하나의 디자인 패턴을 지칭



Controller에게 받은 Model의 데이터를 시각적으로 보여준다.

TripFlow

여행 일정 생성하기

사용자의 입력을 처리하고 url을 결정한다.

```
@PostMapping("/save")
public String save(BoardDTO boardDTO) {
    System.out.println(boardDTO);
    boardService.save(boardDTO);
    return "redirect:/list" ;
```

사용자가 편집하기 원하는 데이터를 가지고 있다.

```
@Getter
@Setter
public class Ddabong {
    private Long likeid;
    private Long postid;
    private Long memberid;
```

사용자의 정보 부터 서비스 이용시 필요한 모든 데이터를 가지고 있다.

content

```
1번 여행입니다. 너무 즐거웠습니다
2번 여행입니다. 바다 구경을 잘하고 왔습니다.
3번 여행입니다. 산 구경을 잘하고 왔습니다.
```

# 게시판 CRUD

- CRUD란 아래의 약자로 게시판의 기본 구조를 지칭

## C create

```
@Transactional  
@PostMapping("/savepost")  
public ResponseEntity<ResponseDTO_SavePost> savePost(  
    @RequestPart BoardDTO boardDTO,  
    @RequestPart List<HashDTO> hashDTOList,  
    @RequestPart (required = false)List<MultipartFile> files)
```



BoardDTO: 게시판 후기  
List<HashDTO>: 해시태그 목록  
MultipartFile: S3 업로드 이미지

## R read

```
@GetMapping("/list/{id}")  
public ResponseEntity<ResponseDTO_Listid> findDetail(  
    @PathVariable("id") Long id){  
    ResponseDTO_Listid responseDTO = new ResponseDTO_Listid(  
        message: "success", status: 200, boardDTO, hashDTO, commentDTO, findimageDTO);  
    return ResponseEntity.ok(responseDTO);}
```



BoardDTO: 게시판 조회  
List<HashDTO>: 해시태그 목록  
MultipartFile: S3 업로드 이미지

## U update

```
@PostMapping("/list/{id}/update") // 데이터 수정  
public ResponseEntity<ResponseDTO_SavePost> update(  
    @PathVariable("id") Long id,  
    @RequestPart BoardDTO boardDTO,  
    @RequestPart (required = false)List<MultipartFile> files,  
    @RequestPart List<HashDTO> hashDTOList) throws IOException {
```



BoardDTO: 게시판 후기  
List<HashDTO>: 해시태그 목록  
MultipartFile: S3 업로드 이미지

## D delete

```
@GetMapping("/list/{id}/delete") // post 게시물 삭제 // id = postid  
public ResponseEntity<ResponseDTO> delete(@PathVariable("id") Long id) {  
    boardService.deleteComment(deletePostDTO);  
    boardService.deletePostImage(deletePostDTO);  
    boardService.deleteHashtagJoin(deletePostDTO);  
    boardService.deletePost(deletePostDTO);
```



BoardDTO: 게시판 후기  
List<HashDTO>: 해시태그 목록  
MultipartFile: S3 업로드 이미지

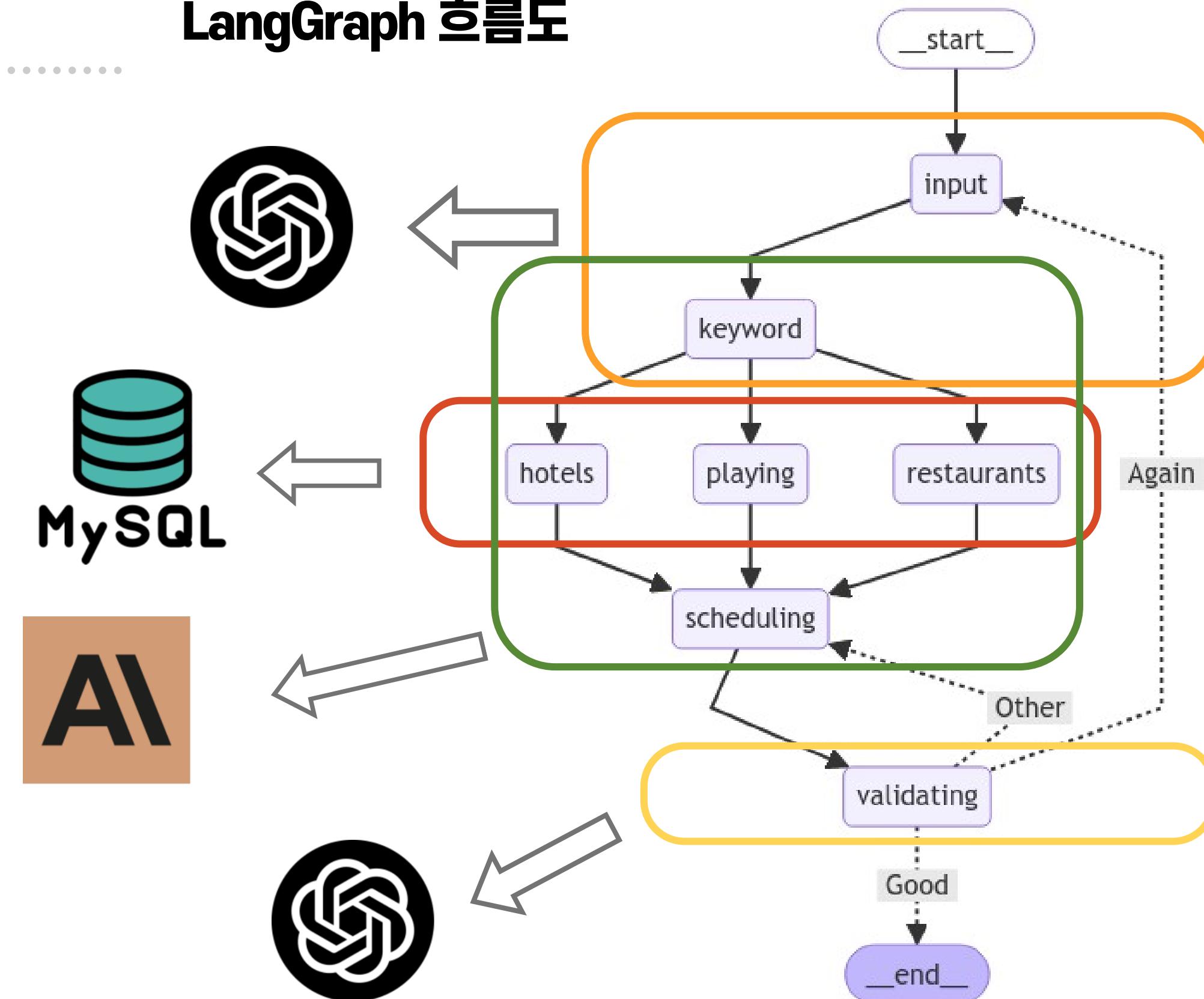
# AI - LangGraph

- LangGraph란 데이터의 흐름도를 나타낸것으로  
어떤 Process로 데이터가 분석되는지 확인 할 수 있다.

## LangGraph

```
class UserState:  
    def __init__(self):  
        # 초기 상태 설정  
        self.state = {  
            'question': None,  
            'keywords': {  
                'days': None,  
                'transport': None,  
                'companion': None,  
                'theme': None,  
                'food': None  
            },  
            'foods_context': [],  
            'playing_context': [],  
            'hotel_context': [],  
            'scheduler': "",  
            'explain': "",  
            'second_sentence': "",  
            'user_age': "",  
            'user_token': "",  
            'is_valid': 0,  
        }  
  
    def get_state(self):  
        return self.state
```

## LangGraph 흐름도



# Data Rag

- Rag를 사용하여 데이터의 수가 부족한 경우 FineTuning을 사용하지 않고 데이터의 신뢰성을 높여 할루시네이션을 방지

## Lodging

placename	latitude	longitude	teenager	twenties	thirties	fourties	fifties	sixties	parking	hoteltype
숙소1	35.1281967	128.8359738	104	325	302	498	235	31	Y	호텔
숙소2	35.21594619	128.991073123	408	489	481	462	428	282	Y	펜션
숙소3	35.01548385	128.83459472	592	348	411	396	176	171	N	모텔

# 숙박 유형 가중치 설정

```
accommodation_weight = {
    '가족': {'호텔': 0.3, '모텔': 0.1, '펜션': 0.6, '게스트하우스': 0},
    '부모': {'호텔': 0.48, '모텔': 0.125, '펜션': 0.395, '게스트하우스': 0},
    '친구': {'호텔': 0.265, '모텔': 0.205, '펜션': 0.26, '게스트하우스': 0.27},
    '연인': {'호텔': 0.27, '모텔': 0.37, '펜션': 0.27, '게스트하우스': 0.09},
    '혼자': {'호텔': 0.263, '모텔': 0.264, '펜션': 0, '게스트하우스': 0.473},
}
```

# 교통수단에 따른 주차 필터링

```
if condition['transport'] == '자차':
    filtered_data = filtered_data[filtered_data['parking'] == 'Y']
```

# 숙박 유형 점수 계산

```
filtered_data['typescore'] = filtered_data['hoteltype'].apply(
    lambda x: accommodation_weight[condition['companion']].get(x, 0))
```

# 연령대 점수 계산

```
filtered_data['age_like'] = filtered_data[user_age_group]
```

# 총점 계산

```
filtered_data['FinalScore'] = filtered_data['typescore'] * filtered_data['age_like']
```

# 총점 기준으로 상위 3k개 숙박지 추천

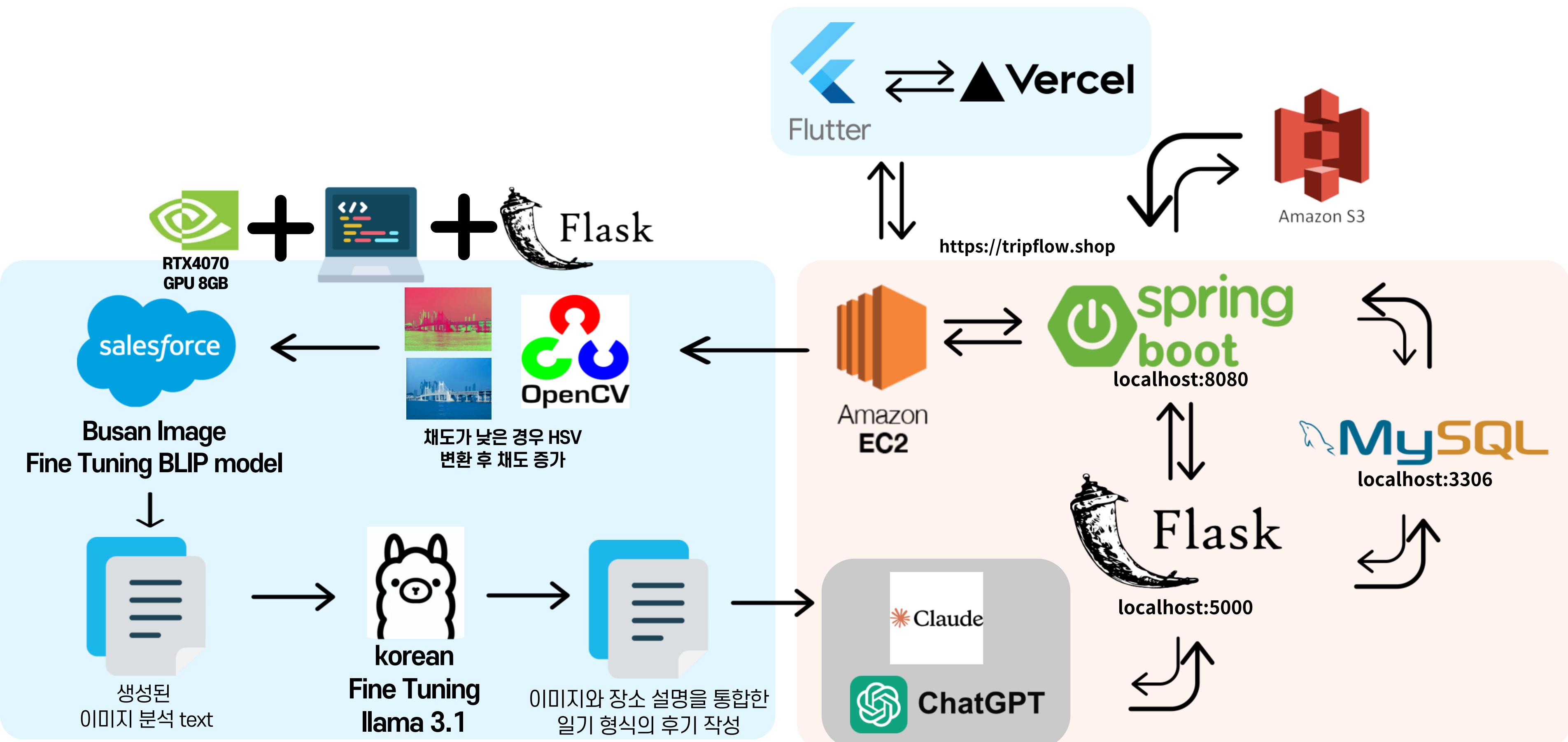
```
recommended = filtered_data.sort_values(by='FinalScore', ascending=False).head(k * 3)
```

# 이름, 위도, 경도 리스트로 반환

```
result = list(zip(recommended['placename'], recommended['latitude'], recommended['longitude']))
```

$Top - 3 * days(Weight * AgeLike)$

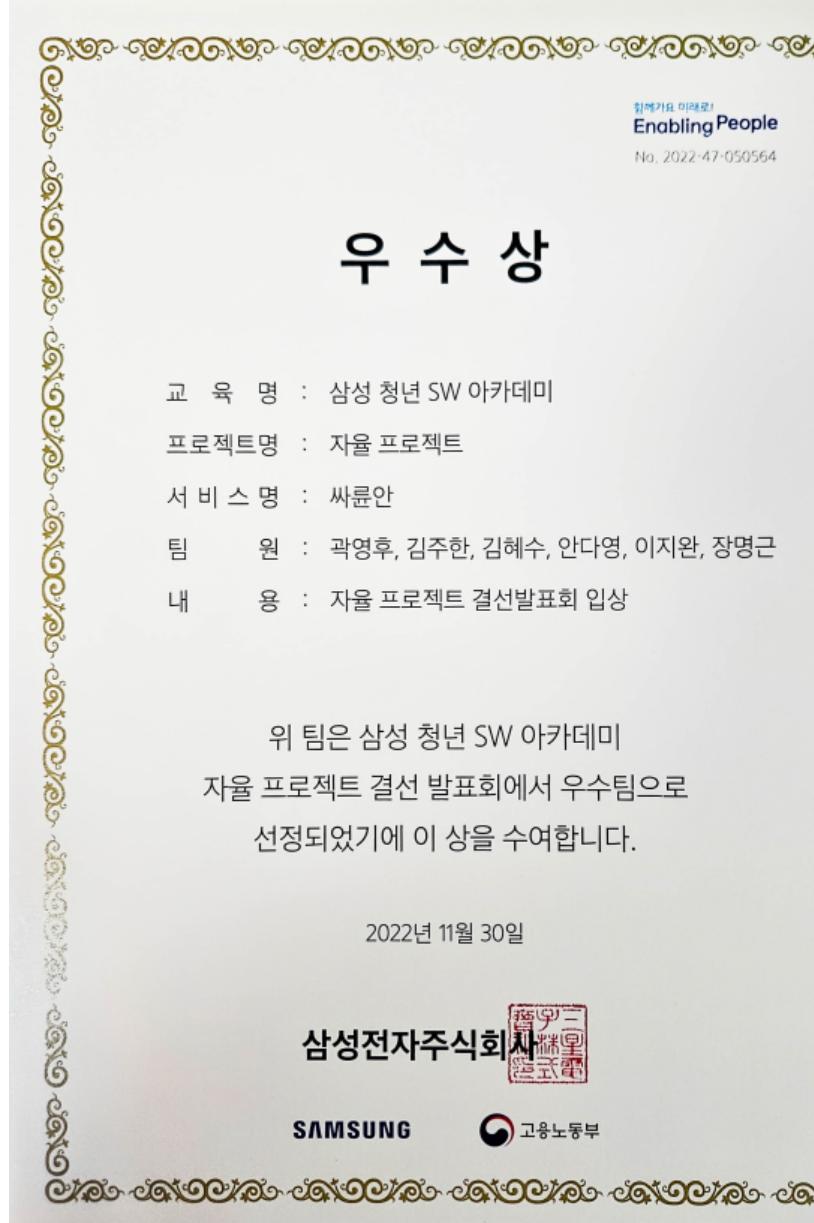
# TripFLow Total Architecture



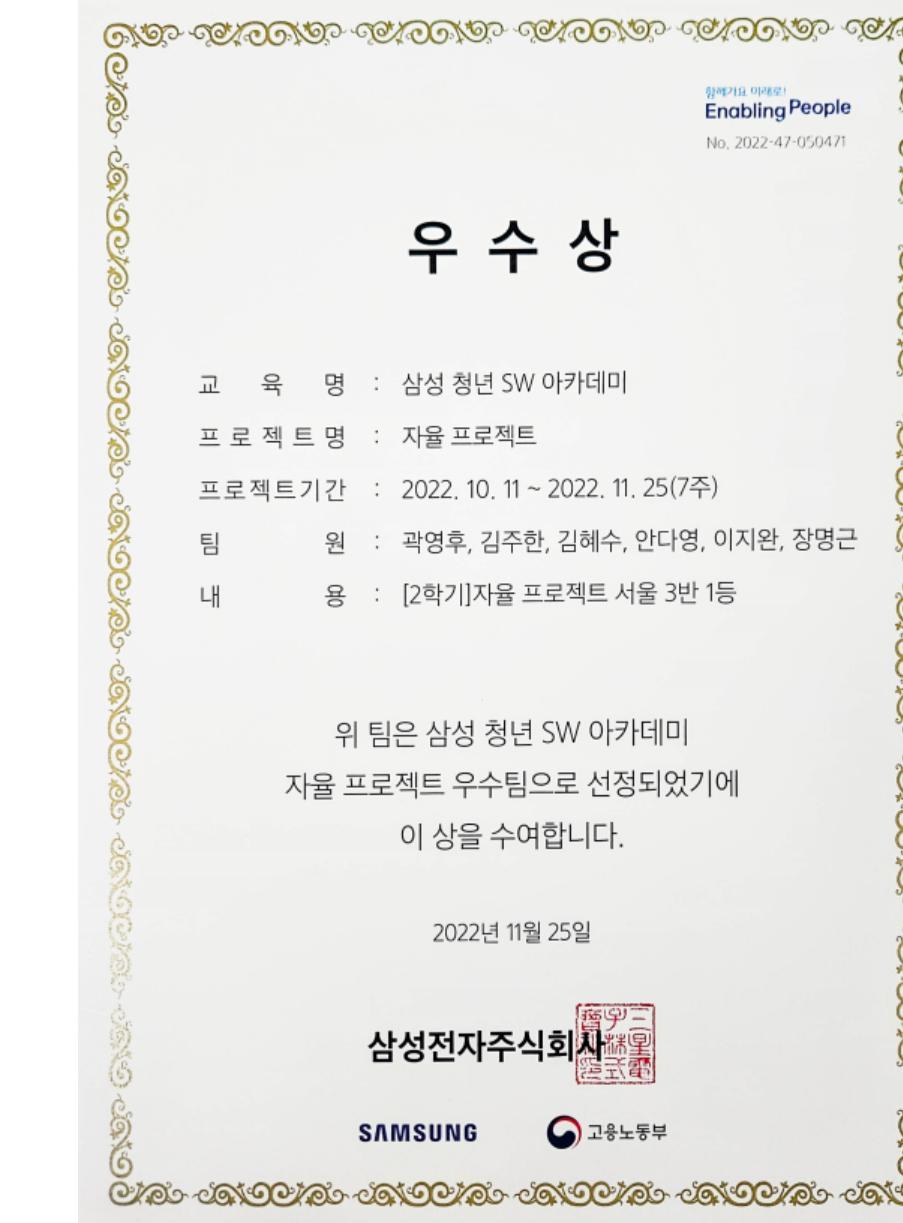
# 수상 및 증빙자료



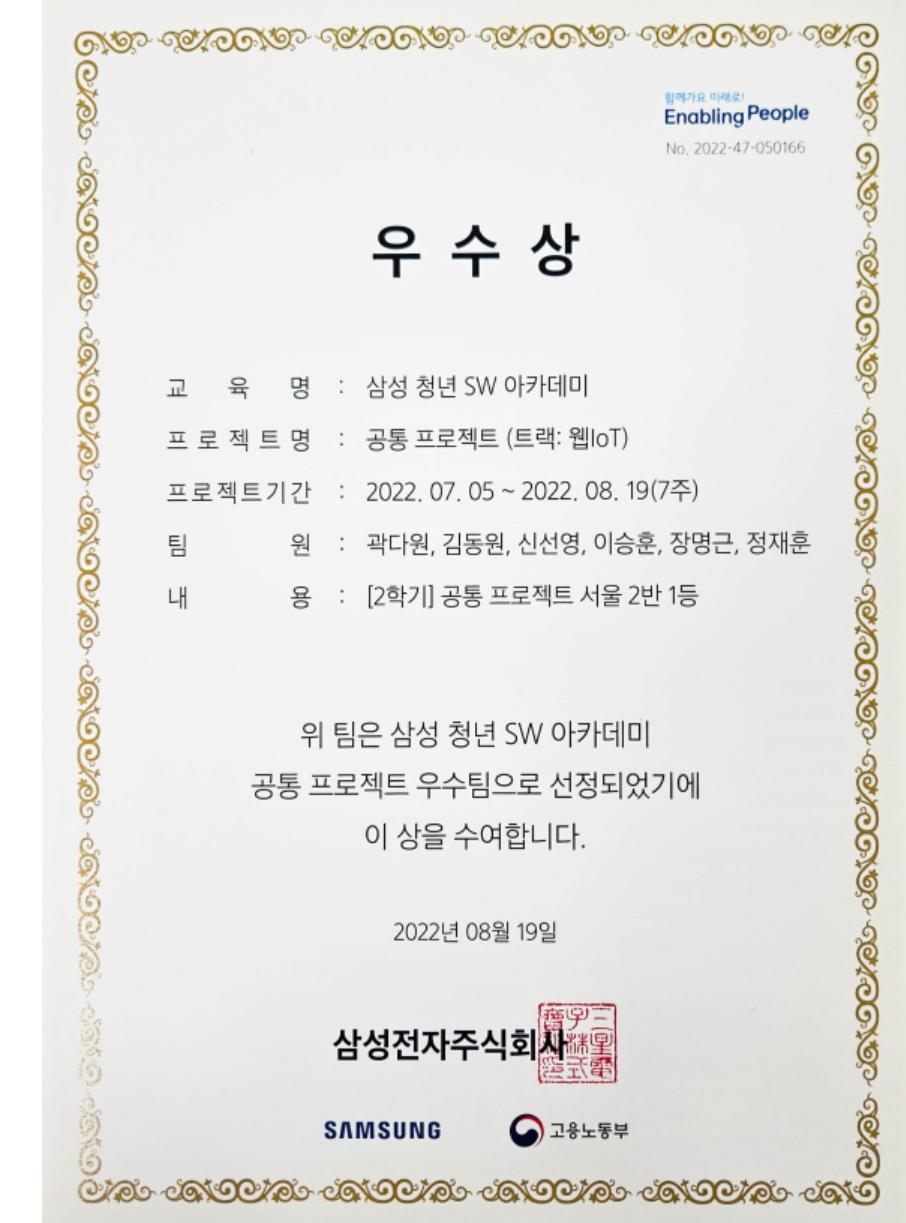
< 수료증 및 우수 등급 선발 >



< SSAFY 전국 발표회 입상 >



< SSAFY 자율프로젝트 수상 >



< SSAFY 공통프로젝트 수상 >

**감사합니다**

**THANK YOU**