

김민규 (Me-in-U)

All-round Developer

 Email contact@ios.kr  Phone 010-9705-6500  GitHub Me-in-U  Location Busan, Korea

 끊임없이 성장하는 개발자
 새로운 기술에 도전하고, 문제를 해결하는 과정을 즐깁니다

목차 (Table of Contents)

 Education & Activities •  Tech Stack •  Projects •  Alpacar •  Tako •  NAMUH •  Links

Education & Activities

학력

동아대학교

컴퓨터공학과

2019.03 ~ 2025.02

학사 졸업

교육

SSAFY

삼성 청년 SW AI 아카데미 13기

2025.01 ~

1학기 CA (부반장)

이수 중

활동

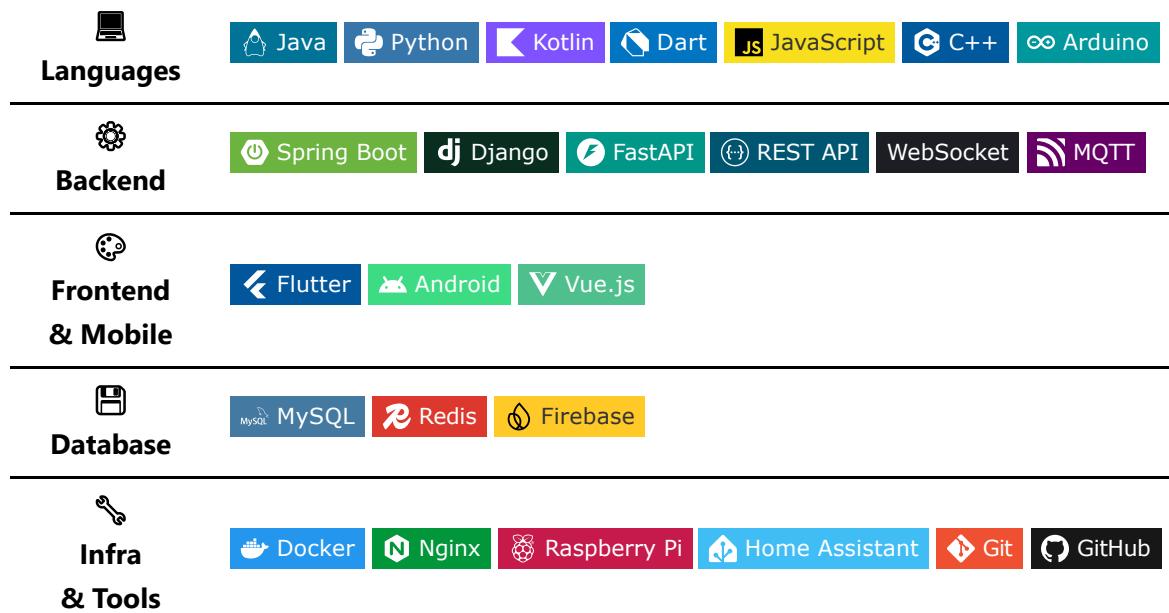
GDSC Dong-A University

1기 ~ 2기

2023.01 ~ 2024.09

★ Core Member

🛠 Tech Stack



🚀 Projects

프로젝트	한줄 요약	링크
01. 🐾 Alpacar	IoT & AI 기반 스마트 주차 관리 및 자율주행 보조 시스템	바로가기
02. 🐸 Tako	블록체인 & AI 기반 TCG 카드 P2P 경매 플랫폼	바로가기
03. 🐦 NAMUH (Tori)	Vision AI 기반 소아암 환아 케어 휴머노이드 로봇	바로가기

01. 🐾 Alpacar

IoT & AI 기반 스마트 주차 관리 및 자율주행 보조 시스템



📅 Period 2025.07.14 ~ 2025.08.19 (6 Weeks)

👥 Team 6 Members (BE 1, Emb 3, AI 1, FE 1)

👤 Role Backend & Embedded Developer

GitHub Repository

💡 핵심 역할

- 🛠 Django 기반 REST API 서버 설계 및 구축
- 💡 Arduino/Raspberry Pi ↔ 서버 간 통신 프로토콜 설계 및 구현
- 🅿 주차 관제 시스템 로직 개발 및 하드웨어 제어

⌚ 담당 역할 & 기여도

🛠 Backend

Django REST API 서버 전체 설계
WebSocket 기반 실시간 통신
주차 관제 로직 구현

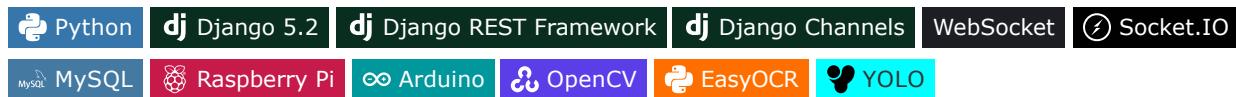
기여도: 100%

💡 Embedded

Raspberry Pi OCR 시스템
Arduino 하드웨어 제어
IoT 통신 프로토콜 설계

기여도: 100%

🛠 My Tech Stack



🌐 기술 선택 이유

Django Channels (WebSocket)

- HTTP Polling 대비 실시간성 향상 필요
- Raspberry Pi ↔ 서버 양방향 통신 구현
- Django 생태계와 원활한 통합

EasyOCR + OpenCV

- Raspberry Pi 제한된 리소스에서 동작
- 한글 번호판 인식 정확도 높음
- 전처리 파이프라인 커스터마이징 용이

◆ 주요 기능



지능형 주차 추천

차량 크기/주차 실력 분석 → 최적 주차 공간 자동 배정



차량 번호 인식

Raspberry Pi + AI(OCR) 활용 실시간 LPR 및 입출차 관리



자율 주행 보조

Jetson Nano 객체 인식/라인트레이싱 기반 자율 주차 보조



실시간 관제

웹 대시보드를 통한 주차장 현황 모니터링 및 제어

💡 성능 개선 & 문제 해결

▶ ⚡ 번호판 인식 성능 최적화 : 처리 속도 10배 향상

⚠ Problem

- 라즈베리파이 카메라 기반 번호판 객체 인식 및 OCR 처리 시 **8초 이상 소요**
- 실시간 입출차 관리 불가능한 수준의 성능 저하
- 동시 처리 요청 시 **메모리 부족** (2GB RAM 환경)
- 한글 번호판 인식 정확도 **78%** 수준

💡 Solution

1. 이미지 전처리 파이프라인 최적화

- 해상도 1920x1080 → **640x480으로 축소** (75% 화소 감소)
- **ROI(번호판 영역)** 동적 추출로 처리 영역 90% 감소
- Grayscale 변환 + 적응형 임계값 처리로 **노이즈 제거**

2. 경량화된 OCR 모델로 변경

- EasyOCR 최적화 설정 (GPU 방식 → **CPU 전용 모드**)
- Beam Search Width 5 → **3으로 축소**
- **한글 특화 모델** 사용

3. 비동기 처리 구조 도입로 병목 현상 해소

- asyncio 기반 **동시 요청 처리** 구현
- 감지 → OCR 파이프라인 **병렬화**

☑ Result

- 처리 속도 8.3초 → **0.8초 이하로 단축** (약 **10배 성능 향상**)
- **한글 인식 정확도 78%** → **94%** (전처리 파이프라인 개선)
- 메모리 사용량 **40% 감소** (2GB → 1.2GB)
- 실시간 차량 번호판 인식 및 입출차 자동 관리 실현

▶ 💡 IoT 통신 최적화 : WebSocket 도입

⚠ Problem

- HTTP Polling 방식의 높은 지연 시간으로 인한 실시간 제어 불가

💡 Solution

- **Django Channels (WebSocket)** 도입하여 양방향 실시간 통신 구현

Result

- 통신 지연 시간 단축 및 실시간 하드웨어 제어 성공

02. 🐙 Tako

블록체인 & AI 기반 TCG 카드 P2P 경매 플랫폼



Period | 2025.08.07 ~ 2025.09.29 (7 Weeks) | Team | 5 Members (BE 2, FE 2, AI 1) | Role | Backend Developer

GitHub Repository

💡 핵심 역할

- ⌚ 스케줄러 기반 경매 자동화 시스템 설계 및 구현
- 👉 Redis(Lua Script) 활용 실시간 입찰 동시성 제어 및 인기 랭킹 관리
- 🔔 N-gram 알고리즘 및 FCM/SSE 기반 실시간 알림 서비스 개발

⌚ 담당 역할 & 기여도

🛠 Backend Core

Spring Batch 경매 자동화
 Redis Lua Script 동시성 제어
 N-gram 검색 엔진 구현
 FCM/SSE 알림 시스템

기여도: 50%

🔗 Integration

AWS S3 이미지 관리
 Swagger API 문서화

기여도: 50%

🛠 My Tech Stack



🌐 기술 선택 이유

Redis Lua Script

- 입찰 검증/캡신 원자성(Atomicity) 보장
- Race Condition 완전 차단
- 분산 환경에서 데이터 정합성 100% 확보

Spring Batch

- 대량 경매 데이터 자동화 처리
- 스케줄링 기반 안정적 마감 처리
- Chunk 기반 효율적 리소스 관리

N-gram (Bi-gram)

- Full Table Scan 제거 (성능 8배 향상)
- 부분 검색 정확도 92% 달성
- 역색인 기반 빠른 검색 응답

Web3j (Blockchain)

- 스마트 컨트랙트로 위변조 방지
- 투명한 경매 기록 보장
- 에스크로로 안전한 결제 구현

◆ 주요 기능



투명한 경매

블록체인 스마트 컨트랙트를 통한 **위변조 불가능한 경매 기록** 및 에스크로 결제



AI 등급 판정

객체 탐지 모델(YOLO)을 활용한 **TCG 카드 상태 자동 등급 분류**



실시간 입찰

WebSocket 및 Redis를 활용한 **초저지연 실시간 입찰 시스템**



실시간 알림

경매 시작/종료, 입찰 경쟁, 낙찰 등 주요 이벤트 **즉각 알림**

❖ 성능 개선 & 문제 해결

▶ 🔥 동시성 이슈 해결 : Redis Lua Script 도입으로 Race Condition 100% 차단

⚠ Problem

- 인기 경매 마감 직전 다수의 동시 입찰 발생 시 **Race Condition** 발생
- 중복 낙찰 및 입찰가 불일치 등 데이터 정합성 오류 발생
- 분산 환경(3대 서버)에서 트랜잭션 간 충돌 발생
- 실제 테스트 결과: **동시 요청 100건 중 7건** 오류 발생

💡 Solution

1. Redis Lua Script 도입

- 입찰 검증/갱신 로직을 단일 **Lua 스크립트**로 통합
- 원자적 연산으로 **Race Condition** 완전 차단
- MULTI/EXEC 대비 네트워크 라운드trip 70% 감소

2. 분산 환경 동시성 제어

- Redis 분산 락 (Redlock) 패턴 적용
- 입찰 순서 보장 (timestamp 기반)

3. 로그 기반 디버깅 체계

- 모든 입찰 시도 로깅 (Elasticsearch)
- 정합성 검증 대시보드 구축

Result

- **Race Condition 100% 차단** (동시 요청 1000건 테스트 오류 0건)
- 입찰 처리 응답 속도 **120ms → 45ms로 단축** (62% 향상)
- 분산 환경에서 데이터 정합성 **100% 보장**
- 중복 낙찰 사고 **0건** (6주간 운영 기록)

Solution

- **Redis Lua Script**를 도입하여 입찰 검증 및 갱신 로직의 **원자성(Atomicity)** 보장
- 분산 락(Distributed Lock) 구현으로 동시성 제어 강화

Result

- 데이터 정합성 100% 확보
- 동시 입찰 처리 성능 및 응답 속도 대폭 향상

▶ 🔎 검색 성능 개선 : N-gram 알고리즘

Problem

- 카드명 부분 검색 시 **Like 쿼리**로 인한 Full Table Scan 발생
- 1만 건 이상의 카드 데이터에서 평균 검색 응답 시간 **2.5초** 소요

Solution

- **Bi-gram(2-gram)** 토큰ナイ저 적용하여 검색 인덱스 구축
- 역색인(Inverted Index) 기반 검색 최적화

Result

- 평균 검색 응답 시간 **2.5초 → 0.3초** (약 8배 성능 향상)
- 검색 정확도(Precision) **92%** 달성

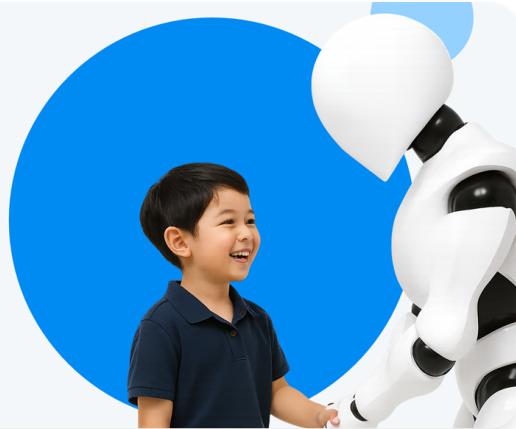
03. NAMUH (Tori)

Vision AI 기반 소아암 환아 케어 휴머노이드 로봇

우리 아이의 친구

NAMUH^S

일상의 작은 변화, 아이의 밝은 미소



Period 2025.10.14 ~ 2025.11.23 (6 Weeks)

Team 6 Members (BE 2, FE 2, Emb 2)

Role Embedded & Backend Developer

GitLab Repository

💡 핵심 역할

- ⌚ 6축 로봇팔(DOFRBOT) 제어 시스템 설계 및 구현
- 👉 11+ 감정 표현 모션 라이브러리 개발 (하트, 포옹, 인사 등)
- 📡 MQTT 기반 실시간 양방향 로봇 제어 프로토콜 설계
- ◎ OpenCV 기반 실시간 얼굴 추적(Face Tracking) 시스템 구현
- ⚙️ 멀티스레딩 동시성 제어 및 선점형 명령 처리 시스템 구현

⌚ 담당 역할 & 기여도

⌚ Embedded Core

DOFBOT 6축 제어
11+ 모션 라이브러리
PID 얼굴 추적
멀티스레딩 동시성 제어

기여도: 45%

📡 Integration

MQTT 프로토콜 설계
Raspberry Pi 5 설정

ESP32 통신 구현
ROS2 연동

기여도: 30%

🔧 Backend

로봇 제어 API
명령 큐 관리
이벤트 처리 시스템

기여도: 25%

🛠 My Tech Stack



🌐 기술 선택 이유

MQTT Protocol

- 경량 프로토콜로 라즈베리파이 적합
- Pub/Sub 구조로 비동기 양방향 통신
- QoS 1로 메시지 전달 보장
- HTTP 대비 60% 응답 지연 감소

PID Controller

- 오버슈트 없는 안정적 추적
- 적분항으로 정상 상태 오차 제거
- 미분항으로 빠른 응답성 확보
- 2축 독립 제어로 정밀한 시선 추적

Threading + Lock

- 서보 명령 충돌 100% 해결
- Lock 기반 I/O 직렬화
- Event 패턴으로 협력적 취소
- 선점형 명령 처리 시스템

ROS2 + Isaac Sim

- 실제 환경 시뮬레이션 가능
- Isaac Lab으로 강화학습 테스트
- MoveIt으로 경로 계획 자동화
- 표준 ROS 생태계 확장성

◆ 주요 기능



감정 표현

11+ 종류의 세밀한 **양팔 협응 모션**으로 풍부한 감정 전달



얼굴 추적

PID 제어 기반 실시간 얼굴 인식 및 시선 추적



스마일 포착

Pre-buffering 기술로 **웃음의 맥락까지 포착**하여 자동 저장



AI 체조

MediaPipe 기반 **양방향 체조 인터랙션** 및 동작 교정

🔧 성능 개선 & 문제 해결

▶ ⚡ 모션 부드러움 최적화 : Overlap 기반 동작 전환

⚠ Problem

- 연속 모션 전환 시 각 서보가 목표 각도 도달 후 대기하여 **부자연스러운 끊김 현상** 발생
- 11+ 동작 시퀀스에서 누적되어 전체 UX 저하

💡 Solution

- Overlap 스케줄링** 도입: 이전 동작 완료 전(80-90% 시점) 다음 동작 시작
- 모션 시퀀스 제어 메서드에 시간차 중첩 기능 구현하여 동작 전환 자동화

☑ Result

- 모션 전환 지연 **50%** 이상 감소
- 자연스러운 흐름으로 감정 표현 몰입도 향상

▶ ↪ 얼굴 추적 응답성 개선 : PID 제어 알고리즘 도입

⚠ Problem

- 단순 비례 제어로 인한 과도한 오버슈트 및 진동

- 빠른 얼굴 이동 시 추적 실패 및 부자연스러운 움직임

Solution

- **Positional PID Controller** 구현 ($K_p=0.3$, $K_i=0.1$, $K_d=0.05$)
- 적분항으로 정상 상태 오차 제거, 미분항으로 오버슈트 억제
- S1(수직), S4(수평) 2축 독립 제어로 추적 정확도 향상

Result

- 추적 안정성 및 응답성 **대폭 향상**
- 부드럽고 자연스러운 시선 추적 실현

▶ 동시성 제어 : 멀티스레딩 I/O 직렬화

Problem

- 얼굴 추적, 모션 제어, MQTT 통신이 동시 실행되어 **서보 명령 충돌** 발생
- 간헐적 각도 오작동 및 제어 불능 상태 진입

Solution

- **threading.Lock** 기반 Arm I/O 직렬화로 동시 접근 방지
- **threading.Event**를 활용한 협력적 취소(Cooperative Cancellation) 패턴 구현
- 선점형(Preemptive) 명령 처리 시스템으로 안전한 동작 전환 보장

Result

- 제어 충돌 **100% 해결**
- 실시간 명령 선점 및 안전한 동작 전환 보장

▶ 실시간 제어 최적화 : MQTT 프로토콜 설계

Problem

- HTTP Polling 방식의 높은 지연 시간으로 실시간 상호작용 불가
- 네트워크 트래픽 과다 및 배터리 소모 증가

Solution

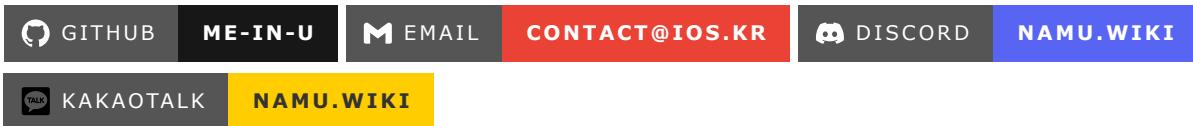
- **MQTT Pub/Sub** 아키텍처 도입 (분리된 토픽: command/event/joint)
- QoS 1 기반 메시지 전달 보장으로 신뢰성 확보
- 양방향 실시간 통신으로 즉각적인 명령-응답 구현

Result

- 명령 응답 지연 시간 **약 60% 감소** (평균 100ms 이하)

- 네트워크 트래픽 약 40% 절감
-

🔗 Links



☞ 함께 성장하는 개발자가 되고 싶습니다!