



국민대학교  
전자정보통신대학  
컴퓨터공학부

# 캡스톤 디자인 I

## 종합설계 프로젝트

프로젝트 명	OLAF
팀 명	ELSA (Elaborate Localization System Architects)
문서 제목	결과보고서


Version	1.2
Date	2020-JUN-09

팀원	김다훈 (조장)
	김선필
	김명수
	배한울
	윤찬우

### CONFIDENTIALITY/SECURITY WARNING


이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인 수강 학생 중 프로젝트 “OLAF”를 수행하는 팀 “ELSA”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “ELSA”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

## 문서 정보 / 수정 내역

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09


<b>Filename</b>	수행결과보고서-OLAF.doc
<b>원안작성자</b>	김다훈, 김명수, 김선필, 배한울, 윤찬우
<b>수정작업자</b>	김다훈, 김명수, 김선필, 배한울, 윤찬우

수정날짜	대표수정 자	Revisio n	추가/수정 항목	내 용
2020-06-07	팀 전체	1.0	최초 작성	
2020-06-08	팀 전체	1.1	내용 수정	개발 내용 및 결과물, 자기 평가 내용 수정
2020-06-09	팀 전체	1.2	내용 수정	부록 내용 수정


 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

## 목 차

개요	5
프로젝트 개요	5
추진 배경 및 필요성	6
개발 내용 및 결과물	7
목표	7
연구/개발 내용 및 결과물	7
연구/개발 내용	7
2.2.1.1 서버	7
2.2.1.2 JSON 서버	8
2.2.1.3 사용자와 로봇간 데이터 통신	9
2.2.1.4 Web UI	9
2.2.1.5 하드웨어 제작	10
2.2.1.6 OpenCR Arduino Firmware 수정을 통한 직진 보정	15
2.2.1.7 카메라를 통한 타일 Edge Detection 및 제어	16
2.2.1.8 Way-Point 알고리즘	20
2.2.1.9 Wall-following 알고리즘	20
2.2.1.10 Object Detection - YOLO	21
2.2.1.11 Number Recognition	22
시스템 기능 요구사항	27
시스템 비기능(품질) 요구사항	27
2.2.3.1 usability requirement - 달성	27
2.2.3.2 efficiency requirement - 달성	28
2.2.3.3 reliability requirement - 달성	28
시스템 구조 및 설계도	29
활용/개발된 기술	30
2.2.6.1 ROS	30
2.2.6.2 OpenCV (Open-source Computer Vision Library)	32
현실적 제한 요소 및 그 해결 방안	33
2.2.7.1 공간에 장애물 (사람)이 많은 상황	33
결과물 목록	33
기대효과 및 활용방안	33
2.3.1 기대효과	33
2.3.2 활용방안	33
자기평가	34
참고 문헌	34
부록	36
사용자 매뉴얼	36

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

운영자 매뉴얼	36
5.2.1 서버	36
5.2.1.1 OS	36
5.2.1.2 언어	36
5.2.1.3 install package	36
5.2.1.4 install global node package	36
5.2.1.5 execute server	36
테스트 케이스	37

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

# 1 개요

## 1.1 프로젝트 개요

매년 들어오는 신입생 및 외부인들은 학과 건물과 같은 대규모 실내 공간에서는 종종 길을 헤매기도 한다. 하지만 실내에서는 GPS가 통하지 않기 때문에 스마트폰도 별다른 도움이 되지 않는 경우가 많다. 마침 실내지도가 있더라도, 지금 내 위치가 어디인지부터가 문제이다. 특히 국민대학교 미래관(구 7호관)의 경우 4층을 기준으로 구관, 신관이 나누어져 있기때문에 처음 방문하는 경우 상당히 혼란스럽다.


OLAF 는 이러한 문제점을 해결하고, 교내 학생 및 교내를 방문하는 외부인에게 도움을 주기 위해서 고안된 프로젝트이다. 국민대학교 각 건물만의 특화된 안내 시스템을 구축하는 것이 이 프로젝트의 목표이다.

주변 환경을 인지하면서 주행하는 실내 이동로봇은 현재 다양한 분야에서 활용되고 있다. 병원 건물 내에서의 물류 이동이나 휠체어 환자 이동, 보안 지키미 로봇, SAVIOKE 호텔 서빙 로봇, Naver의 Around M1 서점 도우미 로봇 등이 활용 되고 있다. 위의 사례들을 보며 OLAF를 통해 안내 시스템이 구축이 된 후 여러가지 분야에 응용을 하여 시스템의 활용 방안을 넓혀 나갈 계획이 있다.

실내 이동로봇은 실내 측위와 주변 환경에 대한 인지를 해야하는데 실내 측위에 대한 여러가지 기술들이 있다. 이러한 측위 기술을 이용한 사례로서 StarGazer 로봇용 측위 실내 센서를 사용하여 천장의 landmarker로 사용해서 측위하거나, 실내 GPS나 Wifi, Beacon 등을 이용하여 삼각 측량으로 측위한다.

로봇 개발을 위해 제공되는 개발환경으로는 ROS(Robot Operating System)를 사용한다. ROS는 로봇을 구성하는 다양한 센서 탐지 및 구동부 제어를 위한 프로그램 패키지 등을 제공한다. 복잡한 로봇 동작들을 위해 다중 프로세스들을 관리하고 연동할 수 있도록 함으로써 여러 개의 로봇 간에 협업도 가능하다. TF 라이브러리를 이용해 여러 좌표 프레임 간 실시간 좌표변환 데이터를 제공함으로써 자세제어를 용이하게 한다.


본 프로젝트에서는 ROBOTIS사의 Turtlebot3 플랫폼을 사용하고 Nvidia사의 Jetson TX2 보드를 주 제어 컴퓨터로 하여, 여기에 LINUX Ubuntu 16.04 버전 운영체제 상에 ROS Kinetic 버전의 메타운영체제를 설치해서 국민대학교 7호관 내에서 사용자가 목적지를 선택하면 해당 목적지로 사용자를 안내해주는 네비게이션 로봇을 구현한다. 하드웨어 제작시 외부 프레임의 경우 일일히 구매를 할시 시간적 비용적인 측면과 OLAF 만의 커스터마이징을 하기에 제약 조건이 많이 생기기 때문에 직접 3D 모델링 및 프린팅을 통해 제작한다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

## 1.2 추진 배경 및 필요성

매년 수천명의 신입생이 입학하고 신입생 뿐만 아니라 학교 외부 관계자, 배달원, 학부모 등 학교에 처음 방문 하는 사람들이 있다. 학교에 방문을 하면 가고자 하는 건물 까지는 핸드폰, 학교내부 지도 등을 이용하여 쉽게 찾아 갈 수 있지만 건물 안에서는 가고자 하는 강의실, 장소까지 쉽게 찾아 갈 수 없다. 우리는 이러한 사람들을 위해 건물 내부에서 원하는 목적지까지 안내해주는 로봇을 만들어 보다 쉽고 빠르게 안내해 주는 서비스를 제공 하려고 한다.

로봇을 제작하여 실내 길 안내가 필요한 한 이유는 대부분의 사람들은 모르는 지역을 갈 때 길 안내 방법으로 스마트폰을 통해 길을 찾고 GPS로 실시간으로 자신의 위치와 자신이 바라보는 방향, 목적지까지 거리 등의 정보를 얻어 목적지까지 간다. 하지만 실내에서는 GPS가 제대로 작동하지 않고 건물 내부에 안내지도가 있다고 하더라도 안내 지도까지 또 찾아가야 한다. 로봇을 사용하지 않고 실내에서 지도를 띄어 자신의 위치를 표시하고 최적의 경로로 길을 안내해 주는 어플들이 몇개 존재한다. 이러한 어플들도 GPS를 사용하지 못하기 때문에 실내 곳곳에 에 있는 무선 인터넷 주소를 읽어 자신의 위치를 파악하고 길 안내를 해주는 방식으로 서비스를 제공한다. 이러한 실내 길 안내 어플들의 공통점은 규모가 매우 큰 건물에서만 길을 안내해 줄 수 있다는 점이다. 따라서 GPS와 인터넷 주소, 비콘 등을 사용하지 못하는 실내 환경, 실내 규모가 작은 환경에서도 길 안내 서비스를 제공하기 위해서는 길 안내 로봇이 필요하다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

## 2 개발 내용 및 결과물

### 2.1 목표

OLAF는 교내 학생 및 교내를 방문하는 외부인에게 도움을 주기 위해서 고안된 프로젝트로서 국민대학교 각 건물만의 특화된 안내 시스템을 구축하는 것이 이 프로젝트의 목적이다.

학교 건물내부에는 GPS가 터지지 않고 어플을 사용하여 길을 안내하기에는 제한된 환경이므로 로봇을 통해 길 안내를 하도록 한다.

따라서 OLAF 프로젝트는 국민대학교 학과 건물을 처음 방문하거나 건물 지리를 잘 모르는 외부인, 신입생을 위한 효과적이고 도움이 될 수 있는 실내 길 안내 로봇을 개발한다.

### 2.2 연구/개발 내용 및 결과물


#### 2.2.1 연구/개발 내용

##### 2.2.1.1 서버

OLAF 구동을 위한 웹 / 앱 애플리케이션 서버는 AWS EC2 인스턴스를 생성하여 서버를 구축하였다. 서버는 Ubuntu 16.04 LTS를 사용하여 구축했으며 내부적으로 Node, npm, Ionic 을 설치하여 서버를 구동한다.

AWS EC2 인스턴스 서버를 구축하여 웹 접근이 가능하도록 설정을 하고, 추가적으로 Ionic, Angular 를 이용한 웹 서버를 구현한다. 웹 서버는 모바일 기기 환경에서의 페이지 구현을 한다.

- 서버 URL : <http://15.164.164.49:8080>

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

### 2.2.1.2 JSON 서버

JSON 서버는 웹 (사용자), 로봇 간 데이터 통신에 있어 중간 다리 역할을 하는 웹서버이다. 데이터 변동에 대해 읽기 및 쓰기를 할 수 있는 웹서버로, 해당 서버는 json 파일을 읽어 해당 json 데이터를 읽고 쓸 수 있는 api 가 구현되어 있는 서버이다.


웹, 로봇 모두 json 서버로 매초 데이터 폴링을 시도하면서 알맞는 코드값에 따라 각각 동작을 실행한다.

json 데이터는 object 타입으로 해당 필드중 isRunning : number 이란 값에 따라 웹, 로봇의 동작을 제어했다. 웹 - 로봇 간 데이터 규약은 다음과 같이 정의 했다.

isRunning	status	web action	olaf action
0	운행 대기	사용자로부터 목적지 입력을 받는다.  입력을 받으면 json 서버로 목적지와 코드값을 1로 업데이트 한다. 그 후 코드값 1에 해당하는 화면으로 전환한다.	원위치에서 사용자로부터 동작 제어 (목적지 선택)를 대기한다.  json 서버로부터 데이터 폴링을 하여 코드값이 1로 바뀌면 목적지로 운행을 시작한다.
1	운행중	사용자 입력을 막고 현재 운행중임을 화면에 표시한다.  사용자 입력은 받지 않으며 json 서버로부터 데이터 폴링을 하여 코드값이 바뀌면 해당 코드값에 맞게 화면을 전환한다.	목적지까지 운행한다.  목적지에 도착 후 json 서버로 코드값을 2로 업데이트 한다.
2	원위치 이동	사용자 입력을 막고 현재 운행중임을 화면에 표시한다.  사용자 입력은 받지 않으며 json 서버로부터 데이터 폴링을 하여 코드값이 바뀌면 해당 코드값에 맞게 화면을 전환한다.	목적지로부터 원위치까지 이동한다.  원위치에 도착 후 json 서버로 코드값을 2로 업데이트 한다.

[표 1] : json file - running status field description

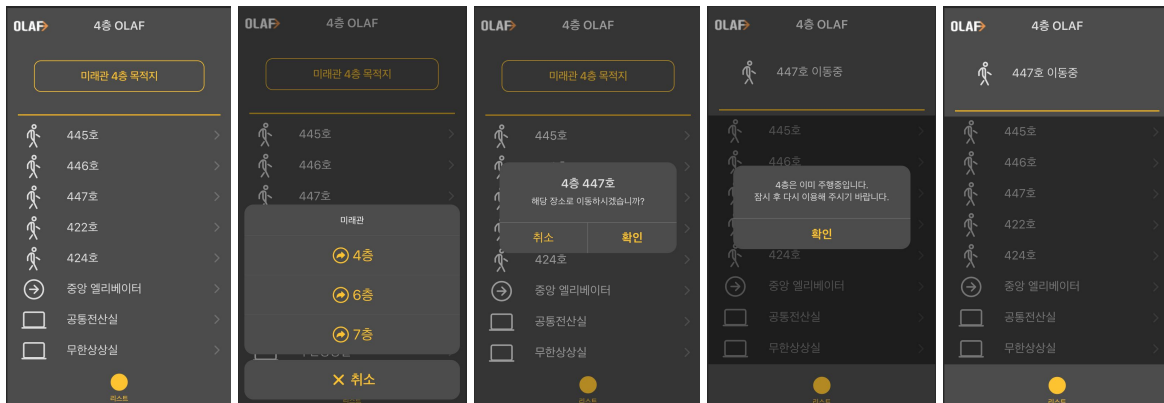


 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

### 2.2.1.3 사용자와 로봇간 데이터 통신

사용자는 로봇을 웹을 통해서 조작 한다. 사용자는 웹 페이지에서 목표 장소를 선택하면 웹서버에서 현재 상태값 및 목적 장소로 json 서버에 전송한다. 로봇에서는 매초 json 서버에서 해당 데이터를 받아오면서 운행에 필요한 데이터를 받게 되면 해당 목적 장소로 운행 후 서버에 운행대기 상태값을 json 데이터로 저장한다. 서버에서는 운행중일 때 매 초 해당 데이터를 확인하여 운행이 완료 됐다는 상태값을 받아오기 전까지 다른 목적지의 입력을 받지 않으며 알림창으로 표기를 해준다.

### 2.2.1.4 Web UI



[그림 1]


[그림 2]

[그림 3]

[그림 4]

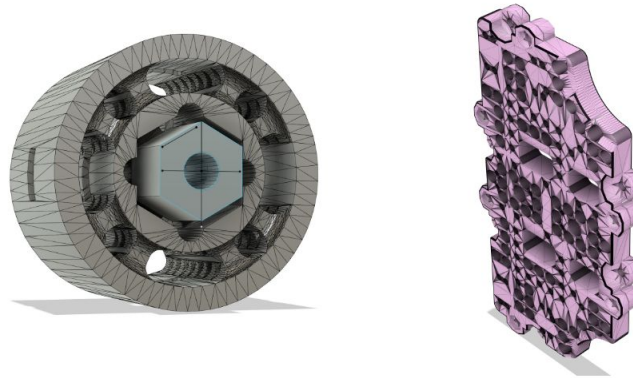
[그림 5]

- [그림 1] : 입력 대기 상태 화면
- [그림 2] : 목적지 층 변경 화면
- [그림 3] : 목적지 입력 화면
- [그림 4] : 목적지 입력시 이미 주행 중인 상태 화면
- [그림 5] : 이동중 화면

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09


### 2.2.1.5 하드웨어 제작

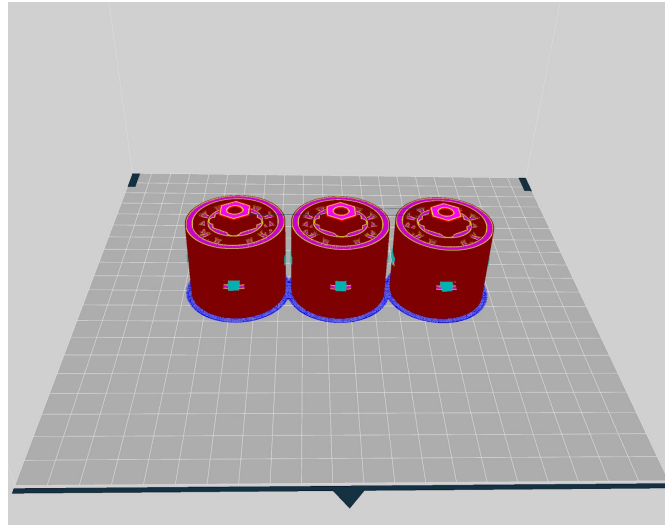
#### 1) 3D 모델링 및 프린팅



[그림 6] Motor-Wheel Joint & Turtlebot3 Plate 모델링

차량 플랫폼 설계 단계에서는 ROBOTIS 사의 TURTLEBOT3 모델을 참고했다. 기존의 TURTLEBOT3 BURGER 모델의 경우 본 프로젝트에서 필요한 다수의 센서 및 Nvidia Jetson TX2 보드를 부착하기에는 너무 작기 때문에 새롭게 설계를 하게 되었다. TURTLEBOT3의 강력한 장점중의 하나인 모듈형 구조를 이용하여 우리는 플랫폼의 크기를 키우기 위해 3D 모델링을 진행하여 외부 프레임을 제작 하였다. ROBOTIS사에서 제공하는 3D 모델링은 자사에서 사용하는 센서를 위해서 설계되었기 때문에 본 프로젝트 팀이 사용하는 센서를 부착하기 위해서는 자체적으로 모델링을 하여 사용해야했다. Autodesk사에서 제공하는 Fusion 360 3D CAD/CAM/CAE 툴을 사용하여 각 센서부를 지지해줄수 있는 부분과 새로 고안한 Plate 및 Monster 바퀴와 Dynamixel 을 연결해줄 Motor-Wheel Joint 부분을 직접 모델링을 진행하였다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09




[그림 7] 3D 파일 슬라이싱

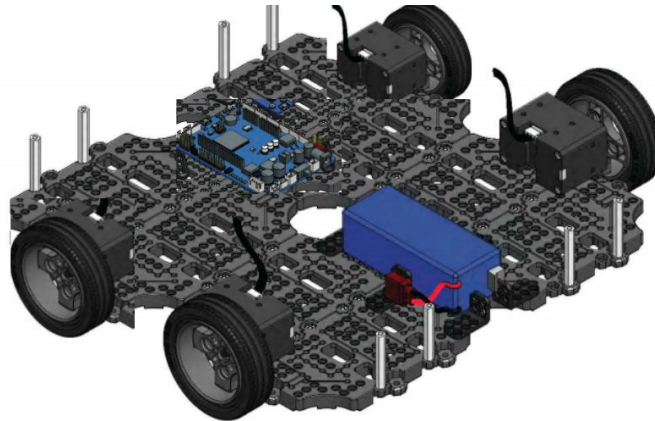
3D 모델링한 파일을 3D 프린터 출력을 진행하기 위해서는 몇가지 과정을 거쳐야 한다. 3D 출력에 필요한 세팅 및 stl 파일을 g code로 변환을 해주기 위해서는 슬라이싱 과정을 진행하였다. 슬라이싱 프로그램으로는 신도리코사의 3dWox 슬라이싱 프로그램을 사용하여 각 파트에 적합한 서포터, 밀도, 강도 등을 조절하였다.

슬라이싱된 파일을 국민대학교 7호관 P Lab에 위치한 신도리코 DP200 모델을 사용하여 출력을 진행하였으며 Plate 한개당 14시간이 소요됐다. 총 32개의 Plate를 제작하였으며, Wheel Joint 4개, opencr 서포터 1개, LiDAR 서포터 1개를 출력을 진행하였다.

#### 1) 하드웨어 설계

하드웨어 설계 부분은 참고할만한 설계도가 존재하지 않았기 때문에, 직접 설계 부터 재료 공수 및 조립을 진행하였다. 하드웨어 설계 및 조립기간은 완성까지 60일의 시간이 소요되었으며, 하드웨어를 설계하는데 다양한 이슈가 발생하여 많은 시간이 소요되었다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09




[그림 8] 1층 구조

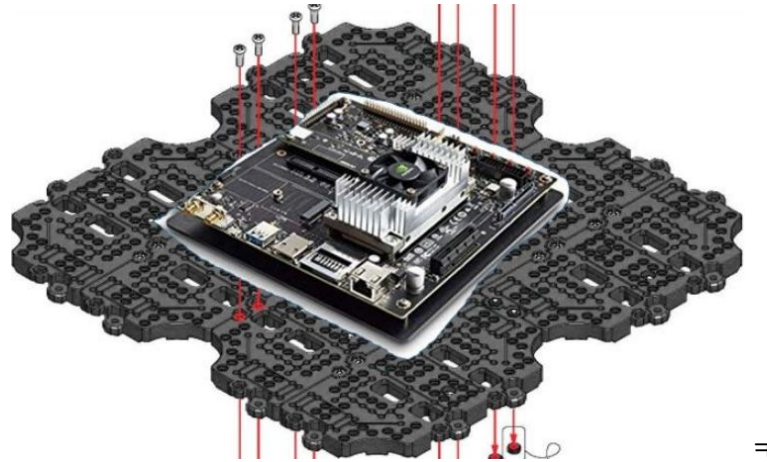
OLAF의 1층 구조는 OPENCNCR 보드 1개, Dynamixel 4개, OPENCNCR 보드용 배터리 1개를 사용하였다. 최하층 및 구동에 필요한 요소들이 많이 들어간 부분이라 안전하게 고정을 해줄 필요가 있었다.



[그림 9] 2층 구조

2층의 구조는 대용량 배터리와 USB 허브를 사용한다. 대용량 배터리는 Nvidia Jetson TX2 보드의 정격전압인 19V 출력을 하며 USB 허브에도 전원이 공급된다. Nvidia Jetson TX2 Board는 USB 허브를 통해 Camera, Lidar, OpenCR 및 IMU를 연결한다. Nvidia Jetson TX2 보드의 소리를 출력할 수 있는 스피커를 장착한다. 스피커는 TKDS 사의 TSOUND 제품을 사용 하였으며 스피커와 Nvidia Jetson TX2 보드간 블루투스 통신을 통해 Nvidia Jetson TX2 보드의 소리를 출력한다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09




[그림 10] 3층 구조

3층의 구조는 Nvidia Jetson TX2 보드가 들어간다. 층의 가로,세로 길이는 27cm x 27cm이며 보드의 가로, 세로 길이는 18cm x 18cm 로 다른 센서를 장착하기에는 공간적인 제약이 발생하여 Nvidia Jetson TX2 보드만 독립적으로 장착하였다.



[그림 11] 4층 구조

4층의 구조는 LiDAR를 위치시키기 위한 층으로 최상층에 위치한다. 층 높이는 42.5cm로

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09


사람의 무릎 높이에 위치하며 장애물 인식 및 Wall Detection 하는데 용이하게 로봇의 정중앙에 위치시키도록 한다. 또한 LiDAR의 영향을 안받기 위해 카메라를 4층 전면 하단부에 장착하였다.



[그림 12] OLAF 전체 구조

OLAF는 총 4층으로 구성된 구동체로 가로,세로,높이 40cm x 40cm x 104cm의 로봇으로 실내에서 큰 제약없이 주행이 가능하게 설계를 하였다. 주행적인 특이사항으로는 제자리에서 360도 회전이 가능하며, 최대 속도 시속 3km까지 나올 수 있다. 바퀴의 경우 기존에 터틀봇에서 사용하는 바퀴를 사용한 것이 아닌, Xycar-A2에 있는 몬스터 바퀴를 사용하였다. 각 층마다 주행시 진동을 줄이기 위해 쇠기둥을 앞, 뒤, 양옆으로 8개씩 기본으로 달았으며, 최하층의 중앙에도 2개를 추가함으로써 진동을 최대한 줄일 수 있게 설계하였다.



 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

### 2.2.1.6 OpenCR Arduino Firmware 수정을 통한 직진 보정


기존의 ROTIS에서 제공하는 Turtlebot3에서 자체적으로 커스텀을 하여 바퀴 변경, 차체 크기 증가, 보조 배터리 장착 등의 이유로 로봇이 무거워져 바퀴가 휘어지는 현상이 발생해 직진을 하지 못하는 문제가 발생했다. 따라서 이 문제를 해결하기 위하여 OpenCR 펌웨어를 직접 수정하는 방법으로 해결했다.

펌웨어를 수정하기 전에 엔코더로부터 양쪽 모터의 속도를 읽어 차이값을 계산하고 차이값을 스케일링하여 조향값으로 제어하는 방법으로 직진 보정을 하였었다. 이 방법은 OpenCR이 Dynamixel의 엔코더값을 읽고 엔코더 값을 master node로 publish 한다. 전달된 메시지를 Nvidia Jetson TX2 Board의 master node에서 subscribe하고 master node에서 조향값을 OpenCR Board에 시리얼 통신으로 전달한 뒤, OpenCR Board에서 subscribe한 조향값을 가지고 Dynamixel Motor에 명령을 내리는 방식이다. 이와 같은 보정방법은 엔코더로부터 양쪽 모터의 속도를 읽어 차이값을 계산하고 차이값을 스케일링하여 조향값으로 제어하는 방법이다.

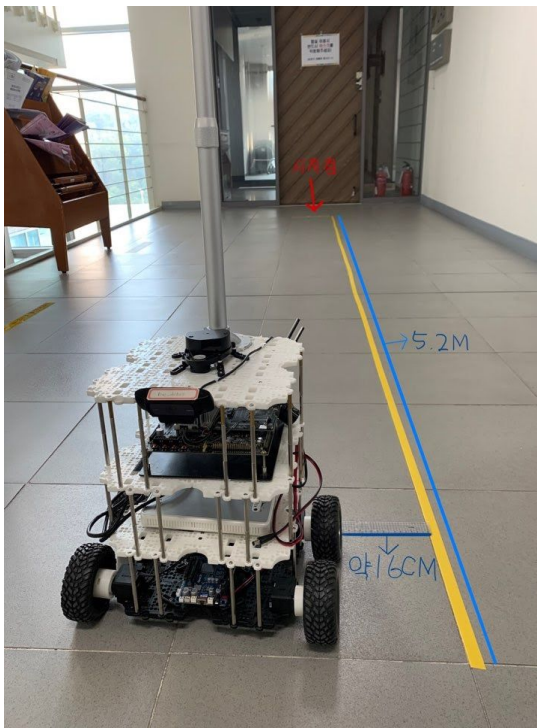
이 방법의 문제점은 조향값을 덮어 씌우는 방식이기 때문에 회전 조향값을 주어도 직진하는 문제가 발생하였다. 또한 모터를 개별 제어하지 못하기 때문에 보정 방식이 한정적이었다. 마지막으로 불필요하게 거쳐야 하는 소프트웨어 구조들이 있었기 때문에 피드백 시간이 긴 단점이 있다.

이를 해결하기 위하여 ROS 메시지 통신을 거치지 않고 OpenCR 펌웨어 상에서 직진 보정이 가능하도록 수정했다. 수정 방법으로는 시리얼 통신을 통해 OpenCR Board에게 Dynamixel Motor의 제어값이 들어오면 각 제어값에 맞는 Threshold를 설정하여 Dynamixel에서 읽은 엔코더값과 Threshold 값의 차이를 계산하여 계산된 차이만큼 새로 각각의 Dynamixel에게 제어값을 주는 방식으로 수정했다. 이러한 방법의 장점으로는 각각의 모터를 개별 보정이 가능해 졌고 설정한 Threshold에 가까워 지도록 P 제어를 하기 때문에 직진주행 뿐만 아니라 회전주행까지 보정이 가능해 졌다. 마지막으로 불필요한 소프트웨어 스택을 거치지 않게 되었다.

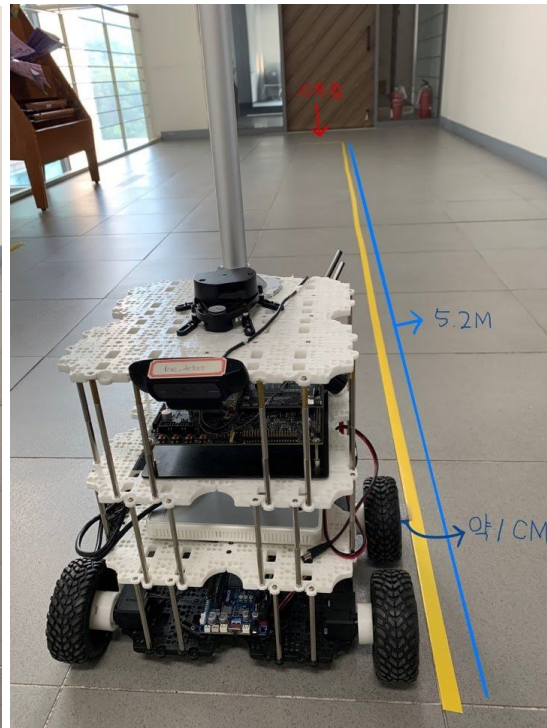
수정된 방법을 통해 직진주행의 성능을 테스트한 결과 기존의 방식은 5.2m 주행시 약

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

16cm의 오차가 발생하였지만 현재 방식으로 5.2m 주행시 약 1cm의 오차가 발생하는 것으로 확인했다. 테스트 중에는 엔코더 센서만을 사용하였다.



[그림 13] 기존의 보정방식




[그림 14] 새로운 보정방식

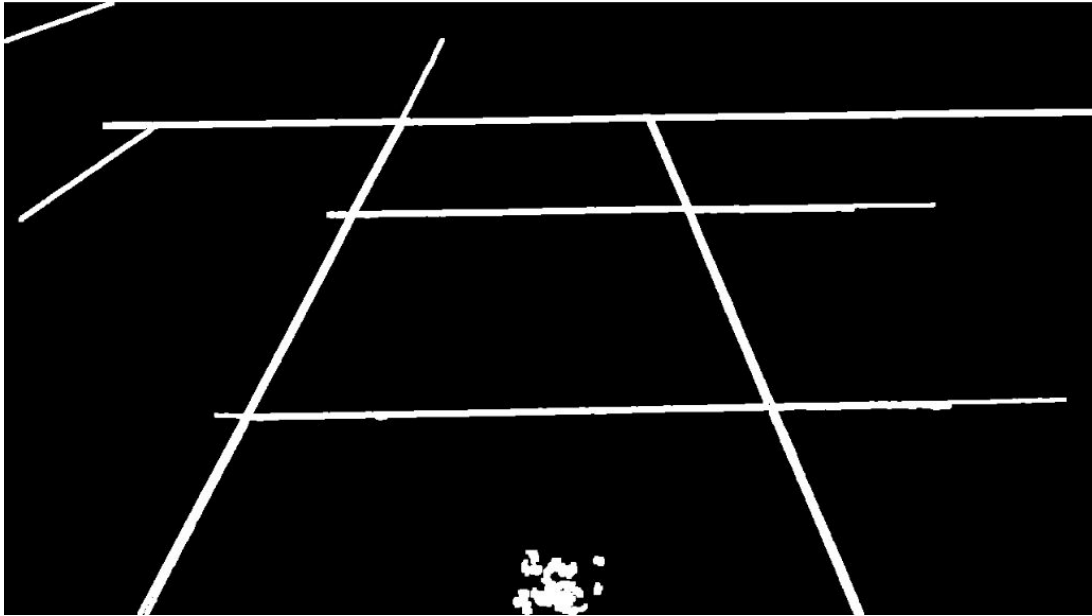
### 2.2.1.7 카메라를 통한 타일 Edge Detection 및 제어

(VER 1)

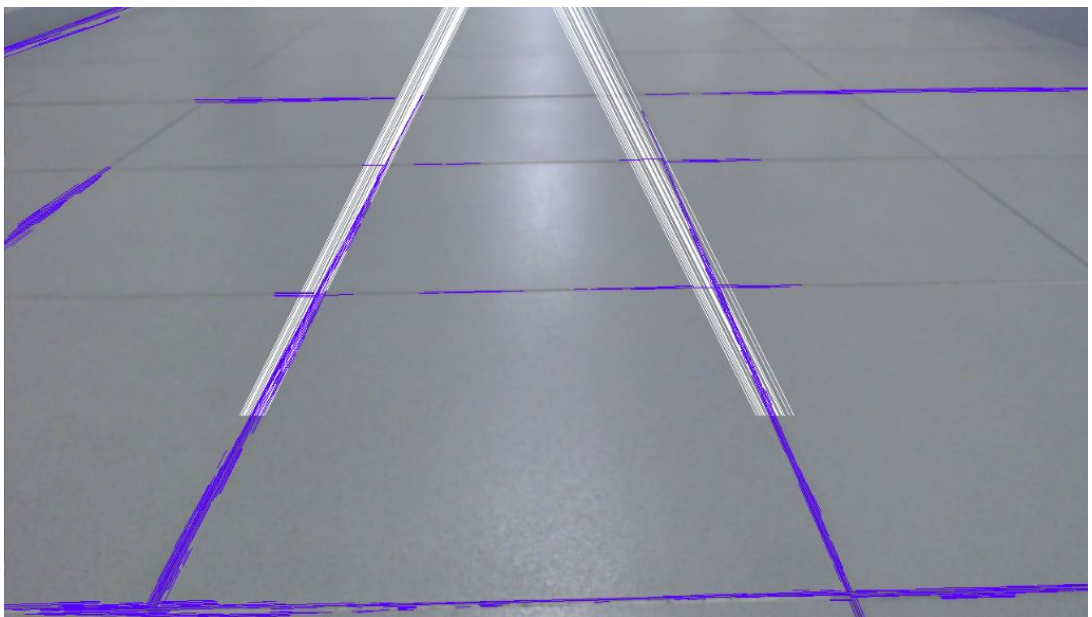
1차 중간발표의 알고리즘을 토대로 하여 더욱 안정적으로 Lane을 추출하는 알고리즘을 채택하였다. 하지만 여전히 빛의 영향에 따른 노이즈가 문제였다. 노이즈를 제거하기 위해 Gaussian Filter와 HSV색공간의 동적 인자와 Canny Edge함수의 동적인자를 사용하였고 추가적으로 히스토그램으로 후보가 되기 위한 임계값을 주어서 빛이 아닌 직선만 추출하였다. Hough Transform 알고리즘을 통해 직선들의 좌표와 기울기를 통해 수평에 가까운 기울기를 배제 하고 차선을 구해 가야할 방향의 직선을 구하였다.



 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09




[그림 15] 전처리에서 빛에 따른 노이즈 Edge(해상도 1280 X 720)



[그림 16] 후처리에서 빛에 따른 노이즈 제거 (해상도 1280 X 720)

위의 [그림 15]과 같이 빛에 의한 Edge를 [그림 16]와 같이 제거하였다. 연산량은 보다 많았지만, 더욱 안정적으로 차선을 추출하였다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09


## (VER 2)

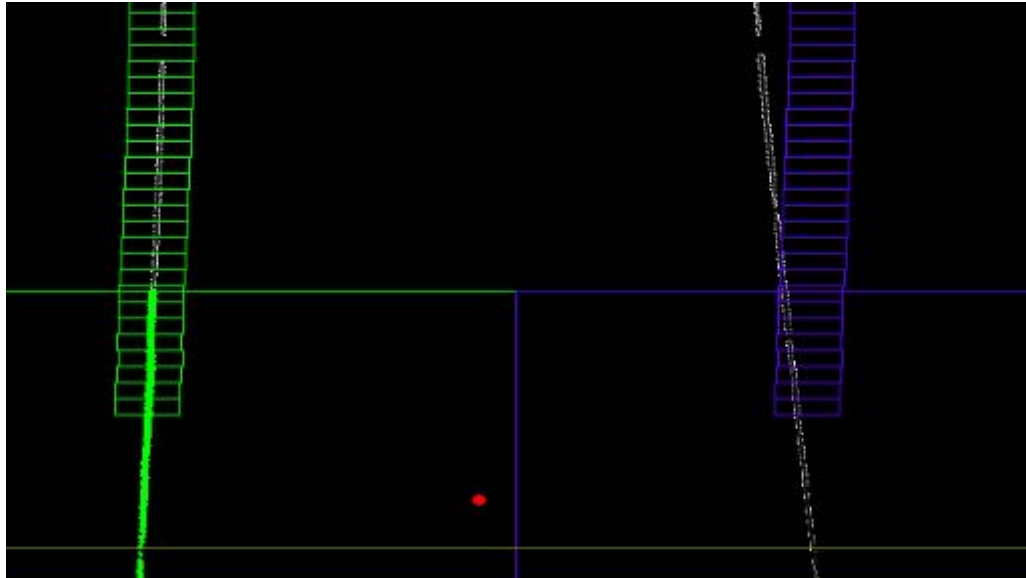
하지만 ROS의 다중 쓰레드 환경에서 많은 연산에 따른 Frame수 저하(8~10)로 인한 제약이 있음을 인지하였다. 이러한 문제는 실시간으로 차량이 주행하기에 중요한 이슈였고, 연산이 적으면서 빛의 영향에 덜 받을 수 있는 새로운 알고리즘을 도입할 수 있게 연구를 하였다.

새로운 알고리즘의 이미지 전처리 부분으로 연산량을 줄이기 위하여 RGB 이미지를 Grayscale화하여 3차원의 채널을 1차원으로 합친 후, Perspective Transform을 하여 연산하는 이미지의 크기를 1280X720에서 640X360 의 크기로 변환하였다. 타일 인식의 주 목표는 차량이 직진시 특정 타일상에서 주행하는 목표를 세웠기 때문에, 현재 차량 기준으로 수직인 Edge를 검출하기 위하여 Sobel Edge의 X filter를 이용하여 수직에 대한 Edge를 검출하였다.



[그림 17] Gaussian Blur와 Sobel Edge, Histogram 분석을 통한 유동적으로 노이즈 제거 및 Perspective Transform 을 이용한 연산량 감소 (해상도 640 X 360)


 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09



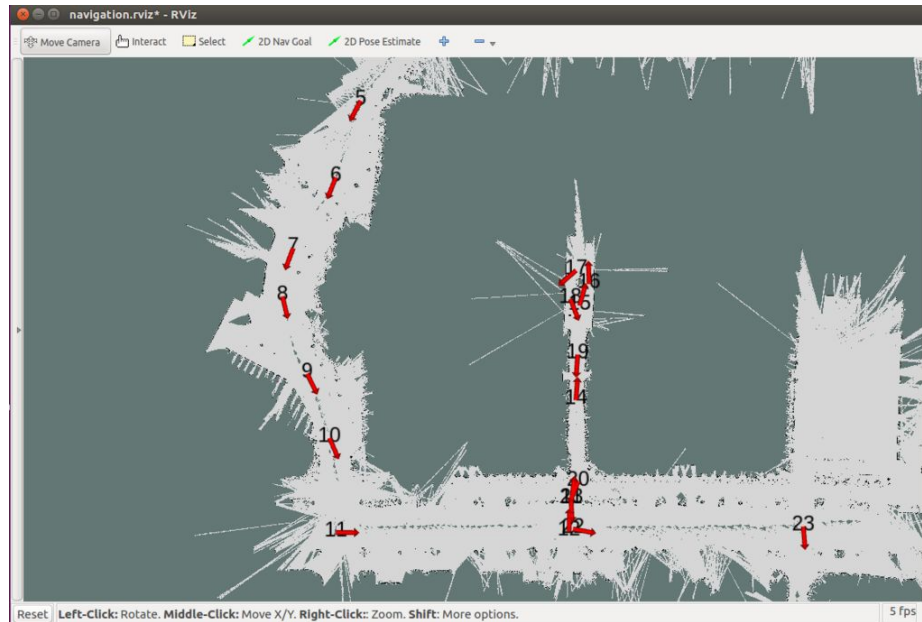
[그림 18] Sliding Window에 따른 Lane Detection 및 제어점 추출 (해상도 640 X 360)

[그림 17]에서 Gaussian Filtering를 통해 노이즈를 제거하고, Sobel Edge를 이용하여 수직인 직선을 추출하였다. 매 Frame마다 Histogram 분석을 통해 유동적인 후보군들은 통하여 통계적 이산점들을 제거하였다. [그림 18]에서 앞에서 구한 히스토그램 후보군을 직사각형의 ROI 영역에서 에지로 추정되는 좌표값을 얻어냈다. 위의 과정을 통해 얻어진 좌표값들을 Polyfitting 하여 직선의 기울기 및 Control Plane 상의 Control Point를 구하였다.

(Ver1) 알고리즘에 비해 영상에서 특정 부분을 분석하여 Control Point를 추출하기 때문에, 정확성 측면에서는 Control Point의 분포가 진동하는 현상이 종종 발견되었다. 이러한 문제를 해결하기 위해 Control Point에 대한 Filtering이 필요하다 판단되어, Low pass filter를 도입하였고, 보다 안정적인 Control Point를 제공할 수 있게 되었다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

### 2.2.1.8 Way-Point 알고리즘




[그림 19] way point 생성

먼저 맵에 강의실, 교차로, 특징점마다 포인트들을 생성한다. 각각의 포인트는 포인트끼리 구별할 수 있도록 인덱스를 가지고 포인트가 어느 지점을 나타내는지 이름을 가진다. 포인트들은 2차원으로 x,y좌표를 가지고 있고 해당 포인트가 교차로 부분인지 아닌지 판단할 수 있도록 하였다. 서버로부터 목적지좌표를 받으면 현재 위치에서 목적지까지 갈 수 있는 알고리즘을 구현했다. 시작지점 (0,0)을 기준으로 서버로부터 목적지 좌표를 받고 목적지 좌표와 시작 좌표를 사용하여 맵에 지정해 둔 포인트들 중 최단경로로 목적지까지 갈 수 있는 포인트들을 배열에 저장한다. 저장한 포인트들의 좌표를 순차적으로 로봇에게 전달하여 로봇이 직각주행만 할 수 있도록 한다. 시작좌표와 포인트좌표의 y좌표 차이를 이용하여 로봇이 90도회전, -90도회전을 판단하고 x좌표값의 차이를 이용하여 로봇이 직진 할 것인지 후진 할 것인지 판단한다. 로봇이 포인트들을 지나 최종 목표지점에 도달하면 길 안내를 종료하고 다시 시작지점으로 돌아 올 수 있도록 하였다.

### 2.2.1.9 Wall-following 알고리즘

Wall-following 알고리즘은 로봇이 Lane Detection을 할 수 없는 상황에 놓였을때,

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

LiDAR에서 나오는 Scanning Data을 사용하여 주행을 하고 벽에 부딪히지 않도록 하는 알고리즘이다. Lane Detection을 하지 못하였을 경우 로봇은 Odometry정보와 LiDAR센서의 데이터 값으로만 직진을 할 수 있어야 한다. 따라서 Lane Detection을 하지 못했을 경우 먼저 LiDAR값을 통해 로봇제어를 한다. LiDAR의 왼쪽 30도, 오른쪽의 30도에 존재하는 데이터값(거리, 강도)을 사용하여 각 방향에 따른 Threshold 값을 설정하고 Threshold에 맞지 않으면 알맞은 조향값을 주어 로봇이 직진을 할 수 있도록 제어한다. 이 방법의 경우 양 쪽 벽이 LiDAR에 인지가 되어야만 한다. 따라서 한 쪽벽만 인식 되거나 양 쪽 모두 인식되지 않을 경우에는 Odometry 정보에 의존하여 직진 보정을 한다.


Odometry 정보는 주행기록계로 엔코더 센서, IMU 센서, 바퀴의 지름, 로봇의 길이 등을 사용하여 로봇이 얼마나 움직였는가에 대해 추정한다. 엔코더 정보와 바퀴의 지름 등의 정보를 통해 로봇이 얼마나 전진했는지 계산하고 IMU 센서를 통해 로봇이 얼마나 회전하였는지 계산한다. 이 정보들은 오차가 누적되어 계산되기 때문에 실제 물리적인 로봇 작용과 차이가 발생하게 된다. 따라서 Odometry 정보는 신뢰성이 낮으므로 Lane\_detection을 하지 못하거나 LiDAR 센서를 이용하지 못하는 경우에 사용한다.

### 2.2.1.10 Object Detection - YOLO

실내와 같은 좁은 환경에서는 다양한 동적 장애물이 로봇에 주행에 영향을 끼칠 수 있다. LiDAR로 전방에 물체가 있는 것은 판단할 수 있지만, 그 장애물이 움직이는 물체인지 아닌지에 대한 판단할 근거는 명백히 떨어진다. 특히 2D LiDAR의 경우 특정 높이에서 한정된 데이터를 처리하기 때문에 단순히 LiDAR를 이용하여 장애물을 판단하고 기동하는 것은 많은 문제가 있을 것이라 판단하였다. 따라서 본 팀은 추가적으로 상단에 객체 인식용 카메라를 추가하고 YOLO 모델을 사용하여 객체 인식을 진행하였다.

YOLO 모델을 선택한 이유는 간단한 처리과정으로 문서상 Nvidia Jetson TX2 보드상에서 fps 23~25까지 출력되며, 이미지 전체를 한 번에 바라보는 방식으로 class에 대한 맥락적인 이해도가 높으며, 각 Object에 비해 좀 더 일반화된 특징을 학습함으로써 본 프로젝트에 적합하다고 판단되어 직접 ROS에 탑재하는 과정을 진행하였다.

성공적으로 YOLO 모델을 ROS에 탑재함으로써 다양한 실험을 진행하였다. Pretraining 된 가중치 파일을 이용하여 yolo\_v2, yolo\_v3를 테스트를 진행하였다. yolo\_v2의 경우

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

Nvidia Jetson TX2 보드에서 실험을 진행을 하였으며 ROS상 8개의 스레드가 도는 환경에서 fps 13 ~ 16, 정확도는 70%로 결과가 나왔으며, yolo\_v3의 경우 fps 3~5, 정확도는 90% 정도의 결과를 도출해냈다.




[그림20] YOLO 객체 검출 - 7호관 4층

상단에 부착된 웹캠을 이용하여 전방 180도를 인식하며, 차량이 진행하고 있는 방향에 존재하는 객체의 클래스와 개수, 위치등을 판단함으로써 주행시 다양한 환경에 대처에 필요한 정보를 얻어낸다. 본 프로젝트에서 주 검출 객체로는 사람으로 설정했으며, 사람이 인식되면 인식된 바운딩 박스 중심으로 로봇 기준 우측, 중앙, 좌측에 사람이 위치하는지 판단한다.

### 2.2.1.11 Number Recognition

차량이 주행을 시작하면 각 센서별로 노이즈가 발생하고, 그 노이즈로 인한 오차가 누적되어 원하는 목표 지점에 도달할 수 없게 된다. 특히 IMU, 엔코더의 경우 노이즈가 심한 센서로 알려져있으며, 이 두가지 정보를 신뢰하여 차량의 위치를 조정하는 것은 실제적인 환경에서 절대 불가하다. 따라서 학교라는 특정 공간에 맞춰 현재 차량이 주행하고 있는 Path의 정보를 줄 수 있는 부분은 각 강의실 입구 상단에 부착되어 있는




 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	결과보고서		
	프로젝트 명	OLAF	
	팀 명	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

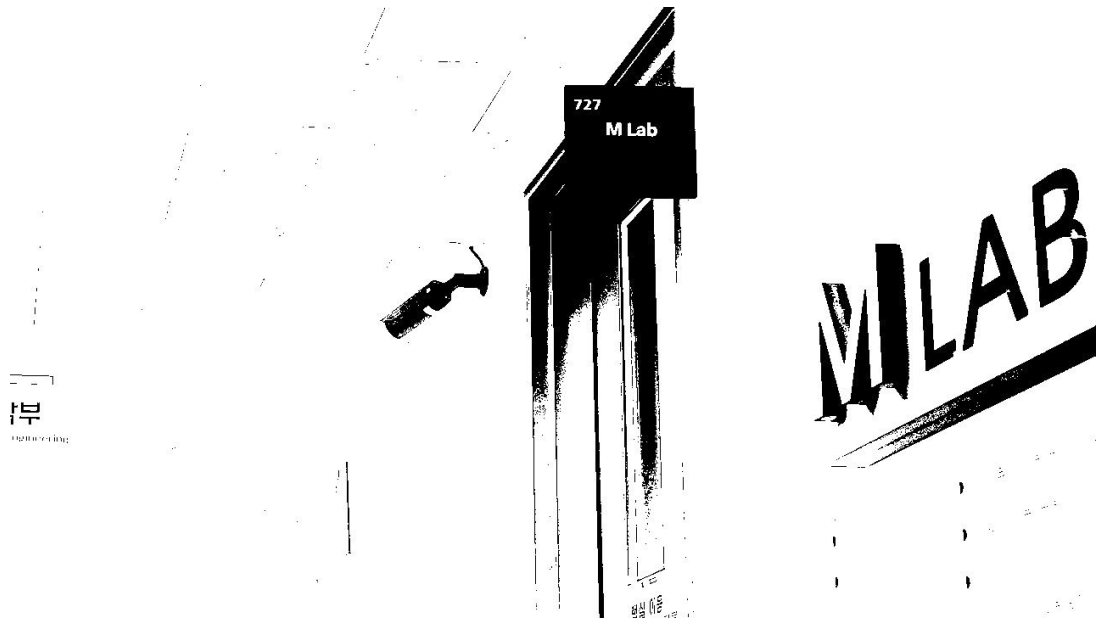
강의실 방 번호를 인식하는 것 뿐이다. 비슷하고 반복적인 구조로 구성되어있는 건물내에서는 LiDAR, IMU, Encoder 정보만으로 Localization을 하는 것은 신뢰성이 매우 떨어졌으며, SLAM을 이용하여 맵을 작성하고, 몬테카를로 Localization 등을 실험해보았지만 매우 불안정적인 결과를 얻어냈다. 따라서 본 팀은 현재 보유하고 있는 센서중 가장 신뢰할 수 있을 만한 카메라를 선택하여 현재 내 로봇의 대략적인 위치를 판단하기 위해 방 번호 인식 알고리즘을 구현하였다.



[그림 21] 원본 이미지

구동체의 상단 카메라가 달려있는 높이는 104cm로 평균 성인 남성의 허리 정도이다. 카메라 각도는 항상 정면을 주시하고 있기 때문에, 방 번호 영역이 존재하는 지역은 항상 규칙적으로 정해져 있으며, 현재 방번호의 뎁스 정보를 얻어낼 수 있는 센서를 보유하고 있지 않기 때문에, 원본 이미지에서 ROI(Region Of Interest)를 설정하였다. 또한 이미지 전체에 비해 방번호판 영역은 현저히 작기 때문에 ROI 를 Zoom in 하는 효과를 구현하여 보다 정확한 정보를 얻기 위해 알고리즘을 설계하였다. 또한 실시간적으로 현재 내 위치를 업데이트 해야하는 로봇의 특성상 이미지 연산량 자체를 크게 줄일 필요가 있어, 다양한 색공간을 사용하는 것이 아닌 RGB 채널을 흑백 영역으로 변환하고 오추 알고리즘을 이용하여 이진화를 진행하였다.


 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09



[그림 22] 전처리 결과 이미지

전처리된 이미지내에서 노이즈 제거를 위해 Gaussian Blur와 이미지내에서 객체의 윤곽선을 찾기 위하여 Canny Edge Detection을 진행하였다. 컴퓨터 비전에는 다양한 특징점 추출 알고리즘이 존재하였으나, 성능이 좋은 알고리즘의 경우 fps가 1~2 정도 나와 실시간성에 부합하지 않아 다른 알고리즘을 고안해내야만했다. 본 팀은 ROI에 추출된 영역에서 이미지 객체에 대한 규칙성을 찾아낸 후, 규칙성을 이용하여 번호 영역을 추출해낼 수 있었다.




 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09



[그림 23] 후보 객체 추출


위의 사진에서 볼 수 있듯이 ROI를 설정하였어도 다양한 후보 객체들이 인식되었다. 후보 객체를 위치 좌표  $x$ 를 기준으로 정렬을 한 뒤,  $x$  축을 기준으로 그레디언트 및 연속적으로 인식되는 객체를 판단하였다. 대개 번호판 영역은 3, 5개의 번호가 연속적으로 배열되어있는것을 분석한 본 팀은 이러한 규칙성을 이용하여 번호판 영역을 추출하였다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

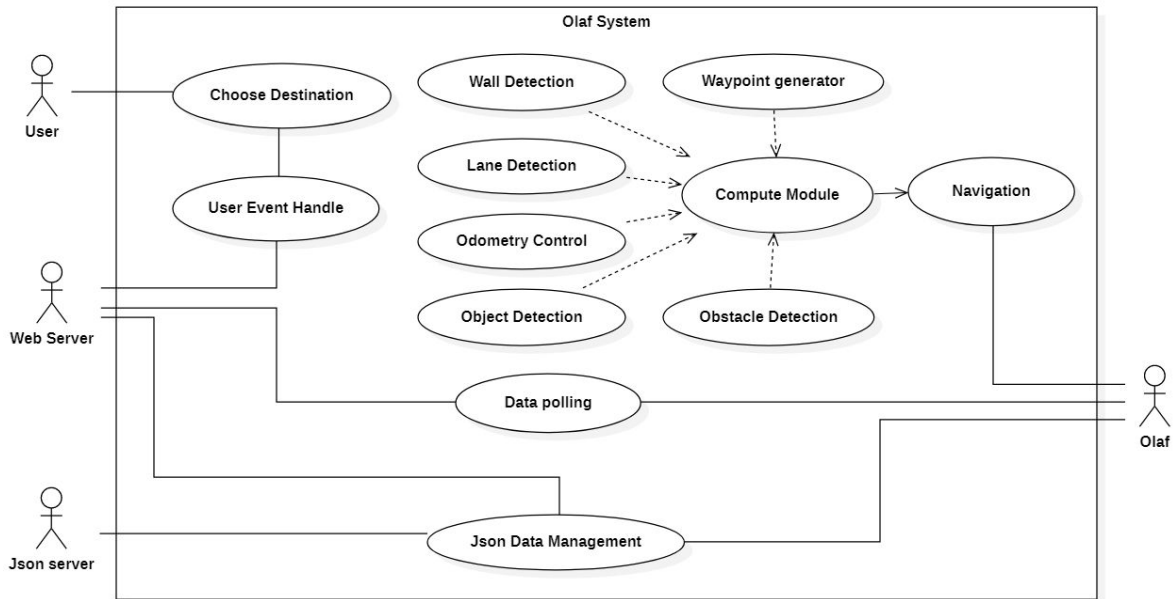


[그림 24] 추출된 번호 영역

추출된 번호 영역에서 숫자를 인식하는 부분은 광학 문자 인식 엔진인 Tesseract를 사용하여 번호 영역에서 번호를 인식하였다. 인식된 번호 영역을 기준으로 4개의 윈도우 영역을 나누어 인식을 진행한다. 우측 상단 부터 좌측 하단 순으로 영역을 구분하였으며 위 사진에서와 보이는 것과 같이 번호 영역 아래 바로 글자 영역이 존재하여 Tesseract가 오작동 하는 문제를 방지하기 위해서 사용하였다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

## 2.2.2 시스템 기능 요구사항



[그림 25] Use Case Diagram

[그림]은 현재 본 프로젝트의 시스템 기능 요구사항을 Use Case Diagram으로 나타낸 것이다. 먼저 User는 목적지를 결정하고 웹 페이지에서 해당목적지를 선택한다.

Web server 는 항상 Json server로 부터 Data를 polling을 실행 하고 있다. 사용자로 부터 목적지를 입력받으면 해당 목적지에 대한 정보를 Json server로 전송한다. Json server에서는 입력 받은 데이터로 Json data를 수정한다.


Olaf에는 Data polling을 진행하다가 Json data 변화를 감지하고 운행 상태로 바뀌면 Navigation을 시작한다.

Navigation은 Wall detection, Waypoint generator, Lane Detection, Odometry Control, Object Detection, Obstacle Detection을 수행 하며 각 모듈을 통합 하고 계산하면서 Navigation을 진행한다.

## 2.2.3 시스템 비기능(품질) 요구사항

### 2.2.3.1 usability requirement - 달성

사용자가 직관적이고 사용하기 편하게 웹 UI를 개발한다. 가장 먼저 해당 층에서 갈 수 있는 목적지를 리스트로 보여주고 다른 층을 가길 원하는 경우 버튼을 만들어 다른 층을 선택할 수 있게 한다. 목적지를 선택 하였을 경우 한번더 목적지를 확인하는 알림 창을 띄우고 로봇이 현재 어느 지점을 목표로 가고있는지 상단에 띄어 알려 준다.


 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

### 2.2.3.2 efficiency requirement - 달성

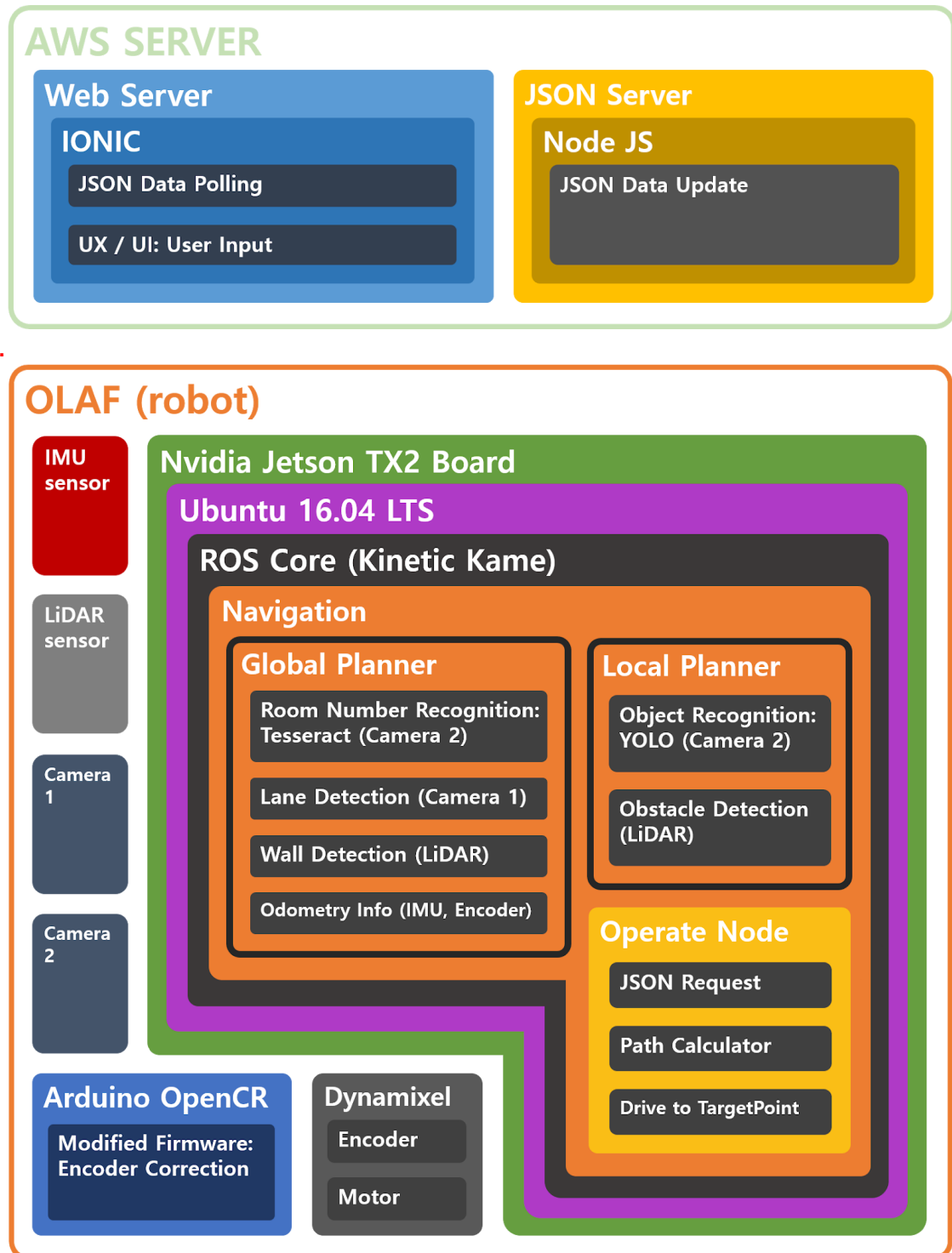
Line Detection, Object Detection, Wall Following 등과 같이 Navigation을 하는 도중에 실행하는 모듈들이 연산하고 연산한 결과값을 publish 하는 시간이 실시간성이 나오게 한다. 현재 대부분의 모듈들은 10프레임을 유지하지만 Object Detection 과 Obstacle Detection에서 yolo를 사용하는 모듈은 평균 3프레임이 나온다. 3프레임도 초당 3번을 publish 하므로 사용에는 문제가 되지 않는다.

### 2.2.3.3 reliability requirement - 달성


본 프로젝트의 핵심은 Navigation 기능이다. 현재 구현한 Navigation 기능은 목적지까지 사용자를 안내하고 안내를 마친 후 처음 시작위치로 돌아온다. 따라서 로봇이 얼마나 정확하게 목적지까지 안내를 하는지, 또 얼마나 정확하게 처음 위치로 돌아오는지에 대한 신뢰성이 필요하다. 국민대학교 7호관 기준으로 바닥이 타일로 이루어져 있는데 정확성의 목표치를 타일 한개, 즉 40cm 내로 목적지에 도착과 처음위치로 돌아오는 것으로 잡았다. 90%의 확률로 목표치 안으로 Navigation을 완료하고 10% 확률로 목표치보다 약 20cm 정도 떨어지게 Navigation을 한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	OLAF	
	팀 명	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

## 2.2.5 시스템 구조 및 설계도

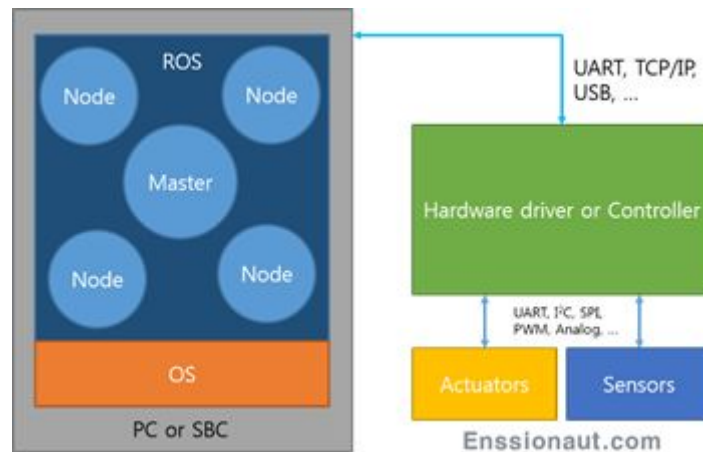


[그림 26] 시스템 구조

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	결과보고서		
	프로젝트 명	OLAF	
	팀 명	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

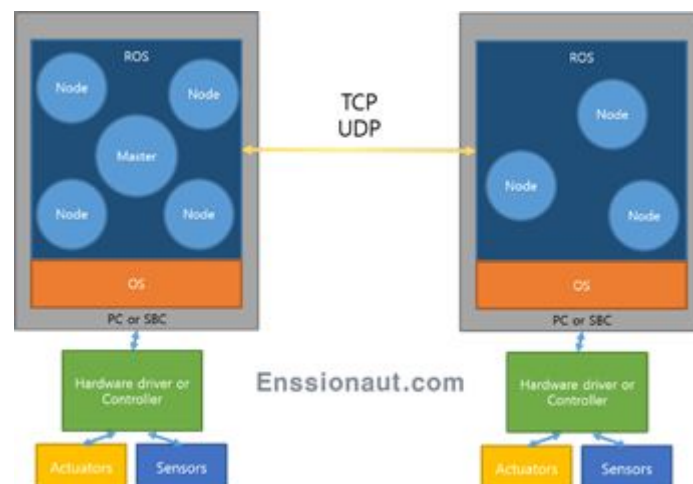
## 2.2.6 활용/개발된 기술

### 2.2.6.1 ROS




[그림 27] ROS 통신

ROS는 Robot Operating System의 약자이다. 이름과는 달리 OS의 기능을 일부 가지고 있으며 그 기능과 목적이 로봇의 구동과 운용, 제어에 초점이 맞춰져 있다. ROS는 Ubuntu와 같은 OS위에서 동작하기 때문에 이러한 OS가 구동 가능한 하드웨어가 준비되어야 한다. 개발용으로는 PC가 사용하기 편하며, 실제 로봇 내에서 ROS의 구동이 필요한 경우 SBC(Single Board Computer)를 사용하게 된다. SBC의 예로는 Raspberry Pi, ODROID, Inter Edison 등이 많이 사용한다. 현재 본 프로젝트에서는 SBC로 NVIDIA Jetson TX2 Board를 사용한다.



[그림 28] ROS의 TCP/UDP 통신

ROS는 ROS가 구동되는 시스템끼리 데이터를 주고받을 수 있다. 이 경우 TCP/IP 프로토콜을 통해 데이터가 전달되며, UDP 프로토콜도 지원되기는 하지만 거의 쓰이지 않는다. ROS가 데이터를

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

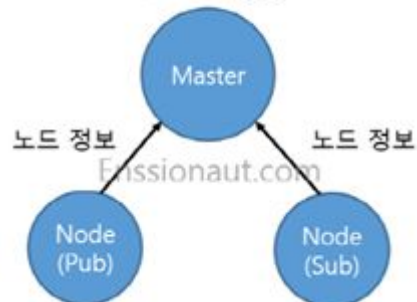
주고받는 형식은 ‘메시지(Message)’라는 단위를 기본으로 한다. 메시지는 토픽(topic)과 해당 토픽에 대한 데이터 값을 가지고 있다. 메시지를 주고받는 당사자들을 ‘node(노드)’라고 하며 노드는 ROS 위에서 구동되는 프로그램의 최소 단위이다. 마스터(master)는 각 노드들의 정보를 가지고 있으며, 어떤 노드가 어떤 토픽으로 메시지를 주거나 받으려 하는지 관리하는 일종의 서버이다.

ROS의 동작 원리는 다음과 같다.

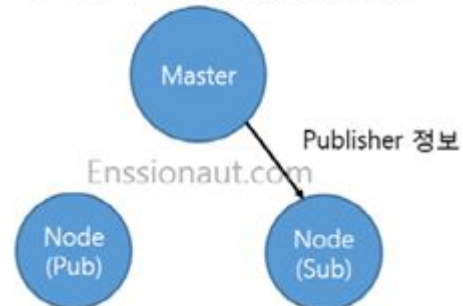
## 1. 마스터 실행



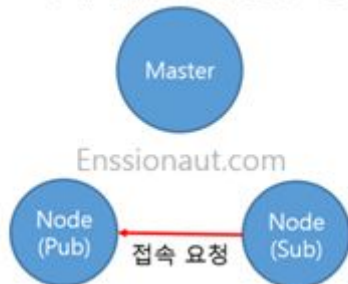
## 2. 노드 실행



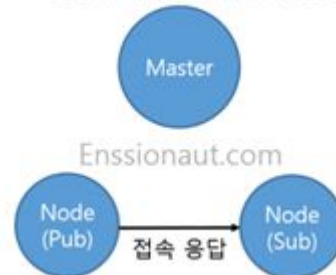
## 3. 마스터가 구독자에 발행자 정보 전달




## 4. 구독자가 발행자에 접속 요청

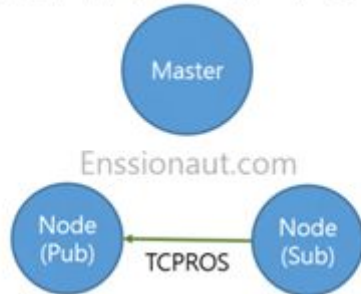


## 5. 발행자가 구독자에 접속 응답

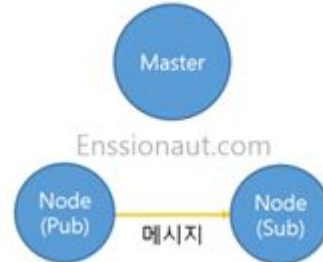


 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

#### 6. 발행자가 구독자 노드에 TCP/IP 방식 접속



#### 7. 발행자가 구독자에 메시지 전달



### 2.2.6.2 OpenCV (Open-source Computer Vision Library)

OpenCV는 인텔이 개발한 이미지 및 영상처리 관련 라이브러리이며, 윈도우(Windows), 리눅스(Linux), 안드로이드(Android), IOS에서 사용할 수 있습니다.


OpenCV는 오픈 소스 컴퓨터 비전 및 기계학습 소프트웨어 라이브러리입니다. 실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리로 실시간 이미지 프로세싱에 중점을 둔 라이브러리이다.

이 라이브러리는 고전 및 최첨단 컴퓨터 비전 및 기계 학습 알고리즘을 포괄적으로 포함하여 2,500 여 개가 넘는 알고리즘이 최적화되어 있습니다. 이 알고리즘은 얼굴을 감지하고 인식하며, 물체를 식별하고, 비디오에서 인간의 행동을 추적하고, 카메라의 움직임을 추적하고, 움직이는 물체를 추적하고, 물체의 3D 모델을 추출하고, 이미지를 결합하여 고해상도를 생성하는 데 사용할 수 있습니다. 이미지 데이터베이스에서 유사한 이미지를 찾고, 플래시를 사용하여 촬영 한 이미지에서 적목 현상을 제거하고, 눈동자를 따라 가며, 풍경을 인식합니다.

컴퓨터 비전 분야는 사람이 시각 정보를 입력 값으로 하여 행동하기 이전에 생각하고 판단하는 부분을 컴퓨터가 대신하도록 하는 인공지능 관련 학문이며 다만, 시각적인 입력 데이터, 즉 영상을 주로 다룬다는 것이 차이점이다.

OpenCV는 Computer Vision 관련 프로그래밍을 쉽게 할 수 있도록 도와주는 Open Library이다.



 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

## 2.2.7 현실적 제한 요소 및 그 해결 방안

### 2.2.7.1 공간에 장애물 (사람)이 많은 상황

현재 실내 길 안내 로봇을 국민대학교 7호관에서 길 안내를 진행한다. 보고서 작성시간인 2020.6.9 기준으로 코로나19 사태의 여파로 수업을 대부분 비대면 수업을 진행한다. 따라서 개발 테스트를 진행하는 경우 건물안에 사람이 별로 없고 로봇이 방해 받지 않는 상황에서 테스트를 진행하였다. 현실적으로 대면 수업을 할 경우 복도에 사람들이 많이 지나다니는 상황이 대부분이다. 이 문제를 해결하기 위해 로봇을 최대한 복도의 가운데로만 주행을 하게 하였다. 또한 Obstacle Detection에서 사람을 정확히 동적장애물로 판단하기 위해 LiDR 센서를 이용한 거리 정보 뿐만 아니라 yolo를 통해 사람인지 아닌지 판단하고 사람으로 판단 되었다 하더라도 한 번더 사람으로 인식한 부분의 가운데 픽셀 값과 너비를 가지고 로봇의 앞에 있는지 판단하게 하였다.

### 2.2.7.2 변칙적인 실내 환경

국민대학교 7호관에는 실내환경이 변칙적으로 구성되어 있다. 예를들어 4층은 신관과 구관이 연결되어 있어 자유롭게 이동이 가능하지만 그 윗층으로는 신관과 분리 되어 있어서 이동을 하지 못한다. 또한 계단과 난관이 많고 유리벽으로 되어 있는 부분이 많아서 LiDAR를 통한 SLAM을 활용하기에 적절하지 않은 환경이다. SLAM을 활용하지 못하면 정확한 Localization을 하기 힘들다. 따라서 Localization 문제를 해결하기 위해 Odometry 정보, 카메라 정보를 통해 Localization을 보완한다. 보완하는 방법은 '2.2.1.11 Number recognition'을 참조하면 된다.

## 2.2.8 결과물 목록

내용 중복을 피하기 위해 결과물 목록은 '2.2 연구/개발 내용 및 결과물' 로 대체한다.


## 2.3 기대효과 및 활용방안

### 2.3.1 기대효과

미래관 구관과 신관의 경계처럼 설명하기 어려운 경로를 OLAF에게 직접 안내 받아서 목적지에 쉽게 도달할 수 있다.

### 2.3.2 활용방안

실내 자율주행로봇의 관점에서 이동형 방범카메라 로봇, 심부름 로봇 등 여러가지 실내 자율주행이 필요한 로봇의 개발에 바탕이 될 수 있다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트명</b>	OLAF	
	<b>팀명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

### 3 자기평가


하드웨어 설계, 구성, 사용 센서 논의 부터 시작해서 하드웨어의 완성 이후 소프트웨어 설계, 구성, 각 센서에 맞는 패키지, 모듈, 알고리즘을 구현하였다. 목적지 도착까지를 주요 포인트로 두어서 정위치에 목적지 도착을 할 수 있게 계속해서 소프트웨어를 수정 및 구현해 나갔다. 현재 하드웨어 상태에서 목적지 도착에 대해선 오차율을 목표지점에 대해 40cm 내로 잡는데에 성공했다.

하지만 직접 하드웨어를 설계한 만큼 하드웨어적으로 외형이나 내구도에서 부족한 부분이 많다. 또한 Olaf에 맞게 소프트웨어를 개발하였기 때문에 다른 로봇에 이식하기 어렵다는 단점이 있다. 주변 환경의 영향을 많이 받아 사람이 밀집된 곳이나 장애요소가 많은 장소에서도 구동을 하기 힘들다는 단점이 있다. 이러한 이유로 이 프로젝트로 상용화는 힘들어 보이지만 다른 분야로 개발 해 나갈 수 있는 확장성은 가지고 있다 판단한다.


서버 구성은 원페이지 구성으로 최대한 단순하게 구성을 했다. 건물 내부의 특정 위치에 대한 json data 값만 바꾼다면 어느 건물에서든 사용이 가능하다.

### 4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	Web	Turtlebot3 e-Manual Git : <a href="https://github.com/ROBOTIS-GIT">https://github.com/ROBOTIS-GIT</a>	ROBOTIS		표윤석	
2	기사	정부, 안내로봇 KS규격 제정 검토...수요 형성 '기대반 우려반'	<a href="https://m.etnews.com/20200323000169">https://m.etnews.com/20200323000169</a>	2020.03.23	변상근	
3	기사	인천국제공항, 안내 로봇 '에어스타' 본격 운영	로봇신문사 <a href="http://www.irobotnews.com/news/articleView.html?idxno=14422">http://www.irobotnews.com/news/articleView.html?idxno=14422</a>	2018.07.11	정원영	
4	Web	OpenSlam Git : <a href="https://github.com/OpenSLAM">https://github.com/OpenSLAM</a>	OpenSlam			
5	기사	라이다의 원리와 장단점, 구현 방식에 따른 종류	TechWorld <a href="http://www.epnc.co.kr/news/articleView.html?idxno=82099">http://www.epnc.co.kr/news/articleView.html?idxno=82099</a>	2019.01.24	양대규	
6	논문	YOLO:Real-Time Object Detection	University of Washington	2016.05.09	Joseph Redmon	

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>				
	<b>프로젝트 명</b>	OLAF			
	<b>팀 명</b>	ELSA			
	Confidential Restricted	Version 1.2		2020-JUN-09	

			n			
7	논문	ROS를 활용한 서빙 이동로봇 구현	배재대학교	2019.02.10	김의선	
8	Web	딥러닝을 이용한 숫자 이미지 인식	조대협 의 블로그 <a href="https://bcho.tistory.com/1156">https://bcho.tistory.com/1156</a>	2017.01.09	조대협	
9	Web	ros-sensor-fusion Git: <a href="https://github.com/methylDragon/ros-sensor-fusion-tutorial">https://github.com/methylDragon/ros-sensor-fusion-tutorial</a>	methyDragon	2019.03.16	methyDragon	
10	Web	Fusion 360 tutorial for absolute beginners	autodesk	2019.09.06	Product Design Online	
11	세미나	오픈 로보틱스 세미나	ROBOTIS	2014.12.21	표윤석	
12	서적	칼만필터는 어렵지 않아	한빛아카데미	2019.05.20	김성필	
13	Web	Xycar-A2 pdf	Xytron <a href="http://auto-contest.kookmin.ac.kr/wp-content/uploads/2019/09/Xycar-A2-%FC%B0%A8%FB%9F%89%EC%84%A4%FB%AA%85%FC%84%9C.pdf">http://auto-contest.kookmin.ac.kr/wp-content/uploads/2019/09/Xycar-A2-%FC%B0%A8%FB%9F%89%EC%84%A4%FB%AA%85%FC%84%9C.pdf</a>		허성민	
14	Web	Naver Labs' On road Intelligence	Naver <a href="https://www.naverlabs.com/en/onroadintelligence">https://www.naverlabs.com/en/onroadintelligence</a>			
15	논문	실내 환경에서의 이동로봇의 위치추정을 위한 카메라 센서 네트워크 기반의 실내 위치 확인 시스템	University of Tokyo	2016.04.16	지용훈	

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

16	Web	Ionic WebSite	<a href="https://ionicframework.com/">https://ionicframework.com/</a>			
17	Web	json-server	npm - json-server <a href="https://www.npmjs.com/package/json-server">https://www.npmjs.com/package/json-server</a>			

## 5 부록

### 5.1 사용자 매뉴얼

사용자는 핸드폰을 카메라로 QR코드를 입력하고 OLAF를 사용할 수 있는 웹사이트로 이동한다.  
해당 웹사이트에서 목적지를 터치하여 OLAF를 사용하면 된다.

### 5.2 운영자 매뉴얼

#### 5.2.1 서버

##### 5.2.1.1 OS

ubuntu 18

##### 5.2.1.2 언어

Javascript  
Typescript

##### 5.2.1.3 install package


nodejs : 12.16.2  
npm : 6.14.4  
nvm: 0.34.0

##### 5.2.1.4 install global node package

ionic : 6.6.0  
json-server : 0.16.1

##### 5.2.1.5 execute server


1. github 파일중 src/web\_server/myApp 폴더를 다운로드 후 npm install 을 이용하여 local node package 를 설치한다.
2. src/web\_server/myApp/API 폴더에 들어간 후 json file 을 global node package 로 다운 받은 json-server 를 실행한다. 해당 json-server 의 json file 은 동일 포맷을 사용하면 데이터를 추가, 제거, 수정을 할 수 있다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09


3. src/web\_server/myApp/src/app/service/ApiService.ts 파일의 base\_ip 변수를 해당 서버의 url 로 수정한다.
4. src/web\_server/myApp 폴더에 들어간 후 ionic 실행으로 웹 서버를 실행한다.
- 5.

### 5.3 테스트 케이스

대분류	소분류	기능	테스트 방법	기대 결과	테스트 결과
로봇	로봇 실행	로봇의 현재 상태를 계산하고 출력하는 노드	<ol style="list-style-type: none"> <li>1. roscore</li> <li>2. roslaunch turtlebot3_bringup turtlebot3_robot.launch</li> <li>3. rostopic list</li> </ol> <ul style="list-style-type: none"> <li>- 로봇의 각종 센서 정보를 관장하는 노드로써 로봇에 tf, sensor state 등의 정보를 담고 있다.</li> <li>- 올바른 실행시 최상단 LiDAR가 작동하기 시작</li> <li>- 작동되지 않을시 실패한 것으로 간주하고 Error 로그를 통해 확인하여 수정한다.</li> </ul>	로봇에 부착되어 있는 센서에 대한 초기화의 진행과 TF, 배터리 사용량등의 정보를 한눈에 볼 수 있으며, 로봇의 현재 가지고 있는 문제점을 정확히 판단할 수 있다.	성공
로봇	Opencr Firmware 업로드	Dynamixel을 전체적으로 관리하는 보드에 Firmware를 업로드한다.	<ol style="list-style-type: none"> <li>1. cd opencr_update</li> <li>2. export OPENCNCR_PORT=/dev/ttyACM0</li> <li>3. export OPENCNCR_MODEL=olaf</li> <li>4. sudo ./update.sh \$OPENCNCR_PORT \$OPENCNCR_MODEL.opencr</li> </ol> <ul style="list-style-type: none"> <li>- 성공적으로 업로드 : OK 출력</li> <li>- 실패 : 업로드 중 NG 메시지가 출력된다. OPENCNCR의 배터리 및 포트 에러가 대부분이기에 이 부분을 확인하여 수정한다.</li> </ul>	로봇의 구동 모터 드라이버 업로드를 통한 현재 로봇과 OPENCNCR 보드의 연결을 진행	성공
로봇	Teleoperation node	키 입력을 통한 로봇 제어	<ol style="list-style-type: none"> <li>1. roscore</li> <li>2. roslaunch turtlebot3_bringup turtlebot3_robot.launch</li> </ol>	로봇의 구동 모터를 키 입력을 통해 제어함으로써	

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

			3. roslaunch olaf_teleop olaf_teleop_key  - 성공 : 키 입력으로 로봇의 바퀴가 구동 - 실패 : 어떠한 키 입력으로도 로봇의 바퀴 구동 불가	주행전 로봇의 상태를 파악할 수 있으며, 제어 로직들을 설계하여 미리 실험해볼 수 있다.	성공
로봇	웹캠 실행	두 개의 웹캠을 작동시켜 이미지 토픽을 발행한다.	1. roscore 2. roslaunch usb_cam test.launch  - 성공 : usb_cam/image 토픽 발행 - 실패 : 토픽 발행 실패	두 개의 카메라에서 받아오는 영상 데이터를 topic name을 부여하여 개별적으로 노드에 할당 가능	성공
로봇	Lane Detection	이미지 토픽을 구독하여 Lane Detection 알고리즘을 통과 시켜 Control Point를 Publish 한다.	1. roscore 2. roslaunch usb_cam test.launch 3. roslaunch vision_lane_detect.py  - 성공 : 실시간으로 현재 이미지를 기준으로 알고리즘이 적용된 이미지가 출력 - 실패 : 어떠한 이미지 출력 없음	하단 카메라에서의 입력을 받아 Lane Detection 알고리즘을 수행 후 Control Point 발행	성공
로봇	Wall Detection	LiDAR의 scan 토픽을 구독하여 전방, 양 측면 30도의 데이터 값을 필터링 처리 후 발행	1. roscore 2. roslaunch hlds_driver hlds_driver.launch 3. roslaunch turtlebot3_example olaf_obstacle  - 성공 : FRONT_DIST, LEFT_DIST, RIGHT_DIST 토픽 발행 - 실패 : 어떠한 토픽 발행 없음	LiDAR의 scanning 데이터를 전처리하여 목표로 하는 방향의 정보 발행	성공
로봇	sound_play	특정 mp3 데이터를 읽어들이어 ROS 상에서 sound를 플레이한다.	1. roscore 2. roslaunch sound_play soundplay_node.launch 3. roslaunch sound_play play.py  - 성공 : 연결된 스피커에서 mp3 데이터를 출력한다. - 실패 : 출력물 없음	사용자에게 출발, 목적지 도착 및 에러 정보 전달.	성공
로봇	SLAM	현재 olaf에 존재하는 데이터들을	1. roscore 2. roslaunch turtlebot3_bringup turtlebot3_robot.launch	Scanning Data, Odometry 정보들을	

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>결과보고서</b>		
	<b>프로젝트 명</b>	OLAF	
	<b>팀 명</b>	ELSA	
	Confidential Restricted	Version 1.2	2020-JUN-09

		토대로 SLAM을 실시한다.	3. roslaunch turtlebot3_slam turtlebot3_slam.launch slam_method:={gmapping, hector, cartographer} 4. rviz  - 성공 : rviz 상에 지도가 시각화 된다. - 실패 : 출력물 없음	이용하여 2D Costmap 을 획득	성공 (실험용 제작)
로봇	YOLO	상단 카메라를 이용하여 객체를 인식한다.	1. roscore 2. roslaunch usb_cam usb_cam2.launch 3. roslaunch yolo_ros yolo_test_rosbag.launch 4. roslaunch yolo_ros yolo_steering.py  - 성공 : 객체가 인식되면 객체의 y 위치좌표가 topic으로 발행 - 실패 : topic 발행 X	YOLO 를 이용하여 객체의 클래스를 파악하여 위치 정보와 함께 장애물을 인식하여 제어하는 로직을 개발한다.	성공