객체지향모델링 결과 보고서

제목: 로봇 프로그래밍을 통한 객체지향 설계 모델링

제출일	2019년 12월 03일
1분반 2조	팀장: 신휘정 (20173116) 팀원: 송민광 (20153308) 팀원: 김현직 (20153240) 팀원: 안지영 (20163331)

이 결과 보고서 양식은 ISO/IEC/IEEE 29148:2011 요구공학 국제표준과 스크럼 애자일 프로세스를 적용하여 진행된 객체지향설계 교과목의 설계 프로젝트에 맞게 테일러링한 것입니다.

목 차

1. 프로젝트 개요

1.1 비전

본 프로젝트에서는 주어진 미로에서 출구를 찾아갈 수 있고, 여러 장애물이 있는 주행시험장 환경에서 도로를 주행할 수 있는 자율 주행 로봇 프로젝트를 제작한다. 각 시험 환경에는 주어진 조건들이 있고 이 조건을 충족하면서 시험 환경을 완주할 수 있어야 한다. 이를 통해 주어진 문제를 해결함으로 객체지향적 분석 및 설계를 할 수 있다.

제작할 로봇 프로그램에서는 상황에 맞게 이동과 방향전환을 할 수 있으며, 카메라를 통해 입력된 영상을 이용하여 벽과 선, 장애물을 구분하고 진행할 수 있는 길을 판별하여 충돌하지 않고 목표지점에 도달할 수 있다.

우리는 이 프로젝트를 통해 로봇 프로그래밍을 학습하고 설계함으로써 객체지향 설계 모델링 기법의 전반적인 지식을 쌓는다. 발생하는 문제들에 대해서는 디자인 패턴을 적용하고 Agile development process의 하나인 스크럼을 적용하여 팀원 모두가 같은 최종 결과물을 생각하며 프로젝트를 진행해보고자 한다.

1.2 문제 기술

요구사항과 제한조건이 많은 프로젝트를 진행할 경우에, 설계 과정을 거치지 못 한 프로젝트는 완성 이후에 발생하게 되는 문제가 발생할 수가 있다. 이들을 최소화 하기 위해 본 프로젝트에서는 소프트웨어 설계 과정을 거침으로써 재사용성을 높이고, 최소 비용으로 유지 보수를 쉽게 할 수 있어야하며, 리소스의 낭비를 최소화함을 원칙으로 한다.

본 프로젝트는 사용자의 시작 입력 이외에는 아무 조작없이 스스로 주행해야하는 자율주행 로봇을 동작시킴으로써, 자율주행 로봇이 겪게 되는 여러 문제점들을 파악하여 계획, 분석, 설계 과정을 거쳐 객체지향 적으로 구현한다.

1.3 요구사항 목록

연번	우선순위	설 명
F-01	1	카메라에서 이미지 정보를 받아온다.
F-02	4	레이저 거리 센서로 거리 정보 및 벽과의 거리를 받아온다.
F-03	21	시간을 잴 수 있다.
F-04	2	움직이거나 멈출 수 있다.
F-05	12	주행시험장에서 코너 길을 차선을 넘지 않고 회전주행 할 수 있다.

F-06	6	1차선,2차선 코스대로 갈 수 있다.
F-07	3	차선을 검출할 수 있다.
F-08	9	로봇이 장애물을 인식하여 장애물과 부딪히지 않고 주행할 수 있다.
F-09	14	정지선에서 3초간 대기한다.
F-10	5	특정 색을 검출할 수 있다.
F-11	7	로봇이 벽을 인식하여 벽에 부딪히지 않는다.
F-12	13	차단바를 인식하여 차단바에 따라 출발여부를 결정할 수 있다.
F-13	14	STOP sign을 인식할 수 있고 인식하여 3초간 정지할 수 있다.
F-14	6	자율 주행이 가능하다.
F-15	22	T자 주차를 할 수 있다.
F-16	23	평행 주차를 할 수 있다.
F-17	18	가장 작은 회전각으로 회전할 수 있다.
F-18	19	현재 좌표에서 목표좌표까지 각도를 알 수 있다.
F-19	16	저장해놓은 좌표간의 이동이 가능하다.
F-20	15	현재 로봇이 위치한 좌표를 알 수 있다.
F-21	10	미로탈출이 가능하다.
F-22	8	차선과 일정 거리 유지한다.
F-23	20	탈출 후 최단거리로 다시 돌아올 수 있다.
F-24	11	양옆의 통로 유무를 알 수 있다.
F-25	17	로봇이 진행했던 좌표를 알 수 있다.
F-26	24	분기점에서 탐색하지 않은 곳의 진입점의 정보를 저장할 수 있다.

1.4 관련 기술

1.4.1 기능적 요구사항

연번	특징	관련 기술
F-01	카메라에서 이미지 정보를 받아온다.	카메라를 통해 받아온 이미지를 이용하여 로봇의 선, 벽, 색 등을 인식할 수 있도록 활용한다.
F-02	레이저 센서로 물체와의 거리 정보 및 벽과의 거리를 받아온다.	레이저 센서를 이용하여 로봇과 벽, 장애물 사이의 거리를 계산하여 주행에 활용한다.
F-03	시간을 잴 수 있다.	로봇이 일정시간 멈췄다가 주행해야 할 경우 활용하고, 미로 진행 시간을 확인한다.
F-04	움직이거나 멈출 수 있다.	N/A
F-05	주행시험장에서 코너 길을 차선을 넘지 않고 회전주행 할 수 있다.	로봇이 모서리나 커브길을 주행하는 경우 활용한다.
F-06	1차선,2차서 코스대로 갈 수 있다.	로봇의 시험장 주행 시 코스별로 판단하여 주행한다.
F-07	차선을 검출할 수 있다.	로봇의 선 인식 후 차선을 검출하여 주행한다.
F-08	로봇이 장애물을 인식하여 장애물과 부딪히지 않고 주행할 수 있다.	로봇 자율주행 시 장애물을 마주쳤을 때 이를 장애물이라 인식하고 부딪히지 않고 주행한다.
F-09	정지선에서 3초간 대기한다.	N/A
F-10	특정 색을 검출할 수 있다.	로봇이 특정 색을 검출하여 이를 활용하여 표지판, 차선, 차단바 등을 구분한다.
F-11	로봇이 벽을 인식하여 벽에 부딪히지 않는다.	로봇이 벽을 판단, 구분하여 미로 길찾기 시에 주행이 가능하다.
F-12	차단바를 인식하여 차단바에 따라 출발여부를 결정할 수 있다.	차단바 여부에 따라 로봇이 출발/멈춤을 구분하여 주행한다.
F-13	STOP sign을 인식할 수 있고 인식하여 3초간 정지할 수 있다.	STOP sign을 인식하여
F-14	자율 주행이 가능하다.	N/A
F-15	T자 주차를 할 수 있다.	
F-16	평행 주차를 할 수 있다.	
F-17	가장 작은 회전각으로 회전할 수 있다.	
F-18	현재 좌표에서 목표좌표까지 각도를 알 수 있다.	
F-19	저장해놓은 좌표간의 이동이 가능하다.	
F-20	현재 로봇이 위치한 좌표를 알 수 있다.	

F-21	미로탈출이 가능하다.	로봇이 미로를 주행하면서 출구를 찾아 나갈 수 있다.
F-22	차선과 일정 거리 유지한다.	
F-23	탈출 후 최단거리로 다시 돌아올 수 있다.	
F-24	양옆의 통로 유무를 알 수 있다.	
F-25	로봇이 진행했던 좌표를 알 수 있다.	
F-26	분기점에서 탐색하지 않은 곳의 진입점의 정보를 저장할 수 있다.	

1.5 이해 당사자 요구사항

연번	이해 당사자	요구사항
St-01	프로그램 기획자	- 자율주행 로봇은 어떠한 미로맵을 주어도 탈출할 수 있어야 하고, 돌아갈 때는 최단경로로 갈 수 있어야 한다. - 주행시험장 주행 시 차단바, 멈춤 표지판에서 멈출 수 있어야 하고, 도로 이외에 탈선을 해서는 안 된다. - 주차공간을 만났을 시에는 주차를 할 수 있어야 한다.
St-02	프로그램 사용자	- 자율주행 로봇이지만, 로봇의 경로 이탈이 생길 경우에 키보드로 로봇을 이동시킬 수 있어야 한다.
St-03	프로그램 개발자	- 자율주행 로봇이 오류,이탈 없이 스스로 주행할 수 있도록 설계한다.

2. 반복에 의한 구현

2.1 운전 주행장 전체 유스케이스

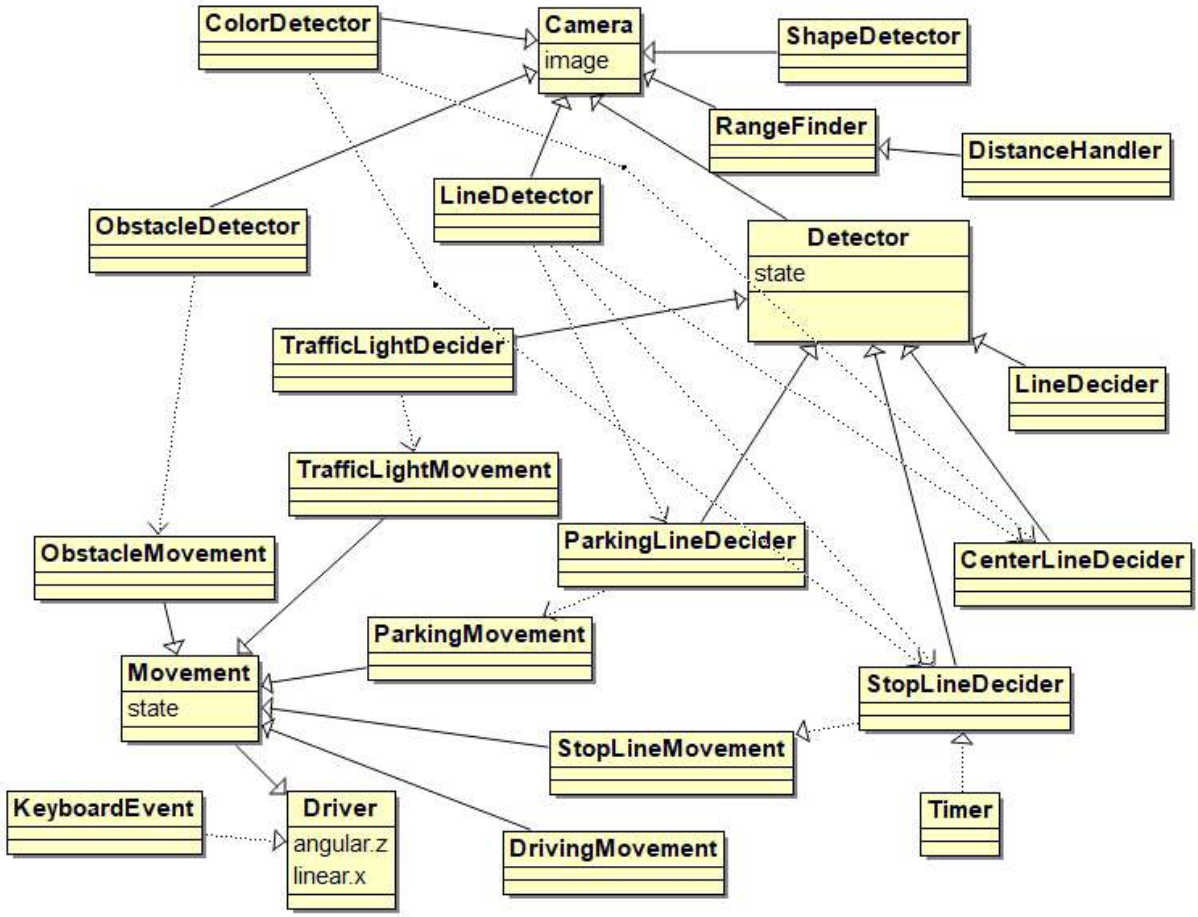
2.1.1 분석

2.1.1.1 확장 유스케이스

유스 케이스	운전 주행장 전체 유스 케이스
액터	우측 카메라, 좌측 카메라, 정면 카메라, 터틀봇, 레이저 센서
목적	주어진 운전 주행장의 조건에 따라 자율 주행을 하는 터틀봇을 구현한다.
개요	운전 주행장에서 주어진 조건에 따라 주행하는 터틀봇의 기능을 구현한다.
유형	기본, 핵심
패턴	② State 패턴

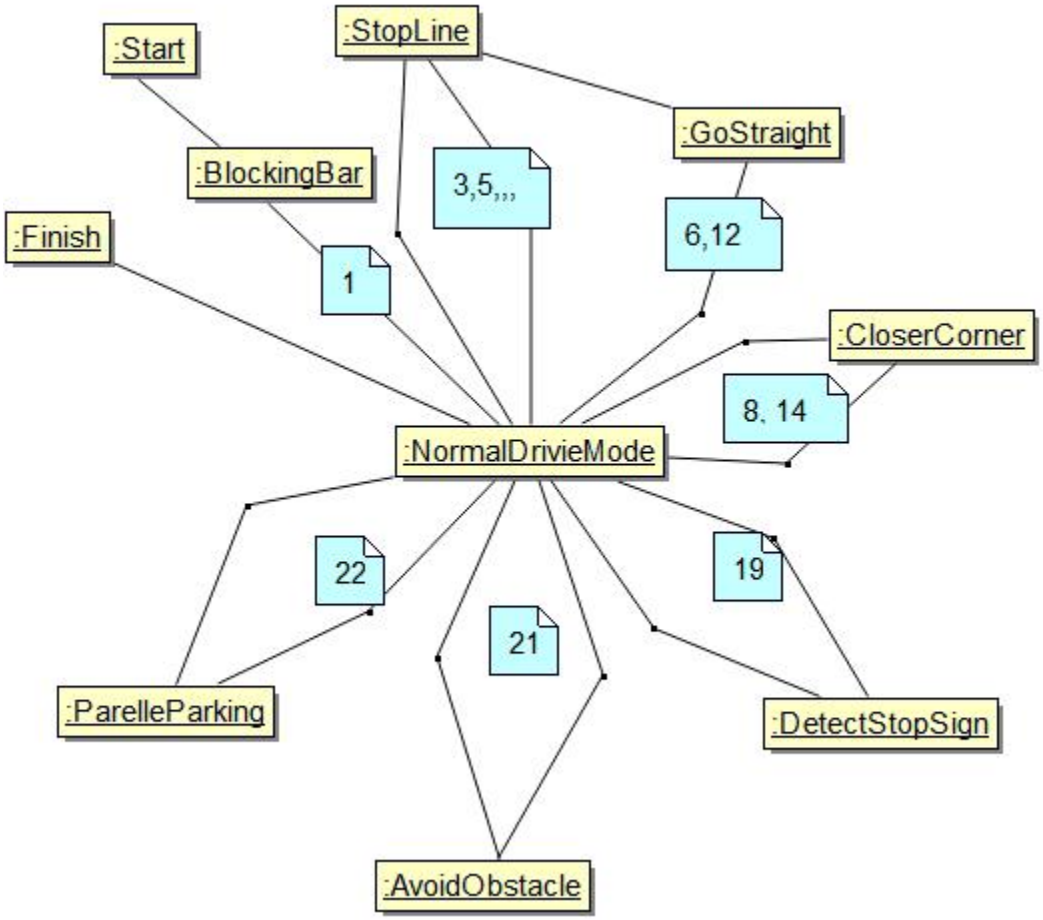
참조	Programming Robots with ROS	
이벤트 흐름 (flow of events) 또는 주흐름 (main flow, basic flow)	**액터**	**Robot Operating System**
	① 터틀봇이 출발선에 위치함으로써 유스케이스가 시작된다.	
	② 정면 카메라에서 영상을 받아온다.	③ 차단바를 인식한다.
	④ 우측 카메라와 좌측 카메라에서 영상을 받아온다.	⑤ 좌측화면과 우측화면에서 차선을 인식한다.
		⑥ 차단바가 없다고 인식한다.
	⑦ 차선을 따라 주행을 한다.	⑧ 정지선을 인식한다.
	⑨ 3초간 정지한다.	
	⑩ 차선을 따라 주행한다.	⑪ 정지선(교차로)을 인식한다.
	⑫ 3초간 정지한다.	
	⑬ 2초간 직진한다.	⑭ 코너 코스로 들어가기 위해 정지선을 인식한다.
		⑮ 코너 코스를 탈출 후 우회전한다.
	⑯ 교차로의 정지선에서 정지한다.	
	⑰ 정지선에서 2초간 직진한다.	⑱ T자 주차코스를 들어가기 위해 정지선을 인식한다.
	⑲ T자 주차를 하고 난 후 후진하여 나온다.	
		⑳ 교차로의 정지선을 인식한다.
	㉑ 교차로에서 좌회전을 한다.	
		㉒ 정지 표지판을 인식한다.
	㉓ 3초간 정지한다.	
	㉔ 차선을 따라 주행한다.	㉕ 장애물을 인식한다.
		㉖ 장애물이 멀어지는 순간에 맞추어 출발한다.
	㉗ 차선을 따라 주행한다.	
		㉘ 평행 주차 코스를 인식한다.
		㉙ 평행 주차를 한다.
		㉚ 평행주차 코스를 빠져나와 안쪽 차선으로 이동한다.
	㉛ 차선을 따라 주행한다.	㉜ 정지선을 인식한다.
	㉝ 정지한다.	
대체 이벤트	⑦⑧⑨ 가는 길에 정지선이 나오면 3초간 정지하는 행동을 반복한다.	

2.1.1.2 도메인 모델


ColorDetector
Camera
image
ShapeDetector
RangeFinder
DistanceHandler
LineDetector
ObstacleDetector
Detector
state
TrafficLightDecider
LineDecider
TrafficLightMovement
ObstacleMovement
ParkingLineDecider
CenterLineDecider
ParkingMovement
Movement
state
StopLineDecider
StopLineMovement
KeyboardEvent
Driver
angular.z
linear.x
DrivingMovement
Timer

2.1.2 설계

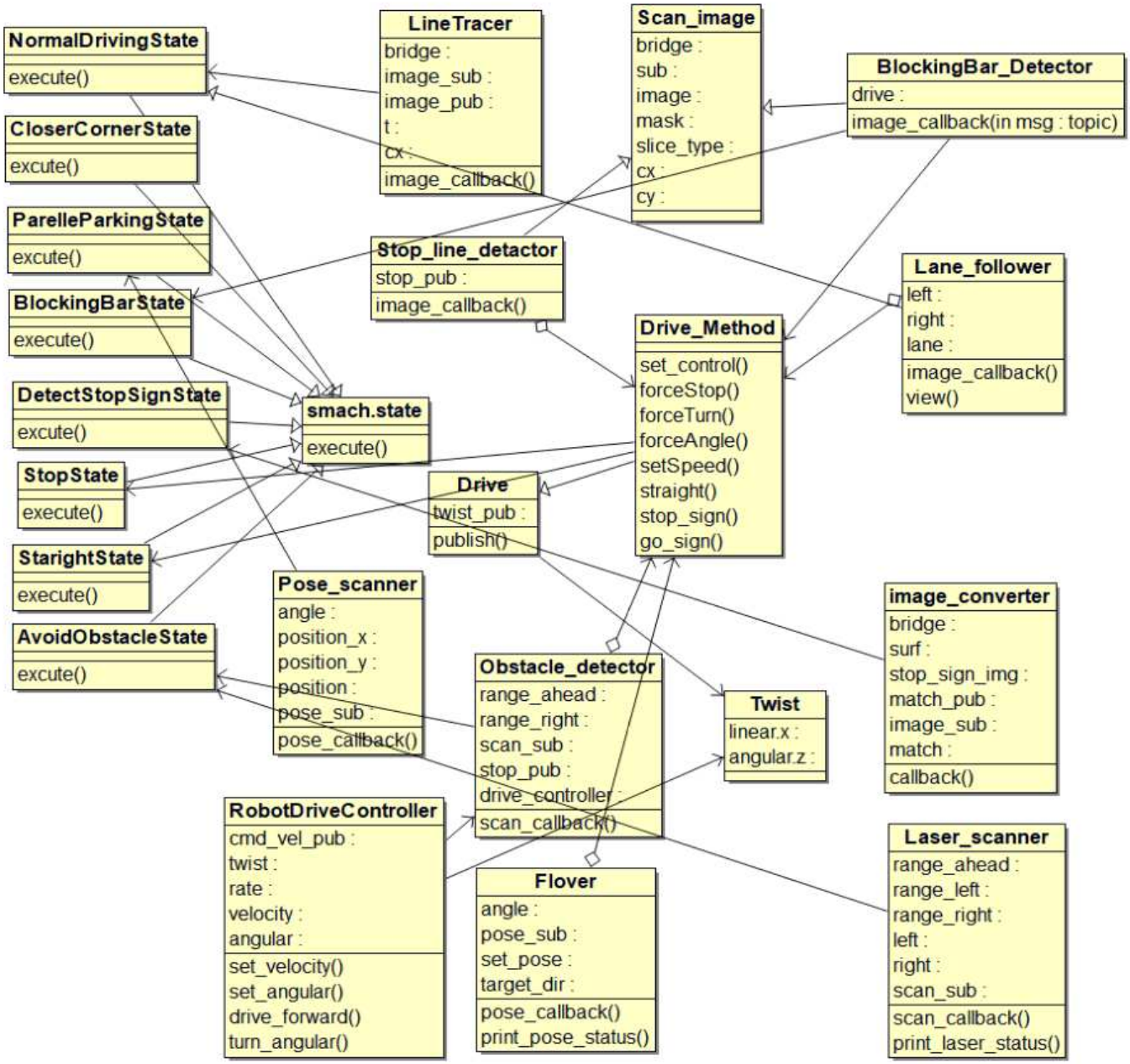
2.1.2.1 통신 다이어그램



StateMachine 상태 순서 :

시작:	BlockingBar	NormalDrivingMode
정지선 인식	StopLine	NormalDrivingMode
첫 교차로 정지선에서 직진	StopLine	GoStraight
코너코스로 들어가기 위함.	NormalDrivingMode	CloserCorner
곡선 코스진행 후 교차로 정지선	NormalDrivingMode	StopLine
T자코스 향하는 코스: 직진필요	NormalDrivingMode	GoStraight
T자코스 진입	NormalDrivingMode	CloserCorner
장애물,표지판 코스로 향하는 곳.	NormalDrivingMode	StopLine
정지 표지판 인식	NormalDrivingMode	DetectStopSign
장애물 피하는 코스	NormalDrivingMode	AvoidObstacle
평행 주차 코스	NormalDrivingMode	ParelleParking
정지선 인식 후 종료.	NormalDrivingMode	StopLine

2.1.2.2 설계 클래스 다이어그램



state의 상태 함수안에 객체를 생성하여 StateMachine을 사용하려 하였습니다. 위와 같은 설계를 바탕으로 구현을 시도 하였지만, 파이썬의 **Call by Object Reference**의 문턱을 넘지 못하고 싱글턴을 이용하려 하거나 flag값으로 변화를 주려하는 등 노력을 하였으나 StateMachine은 구현하지 못하였고 설계까지만 하였습니다. 하지만 StateMachine에 필요한 기능 중 평행주차만 제외한다면 모두 구현에 성공하였습니다.

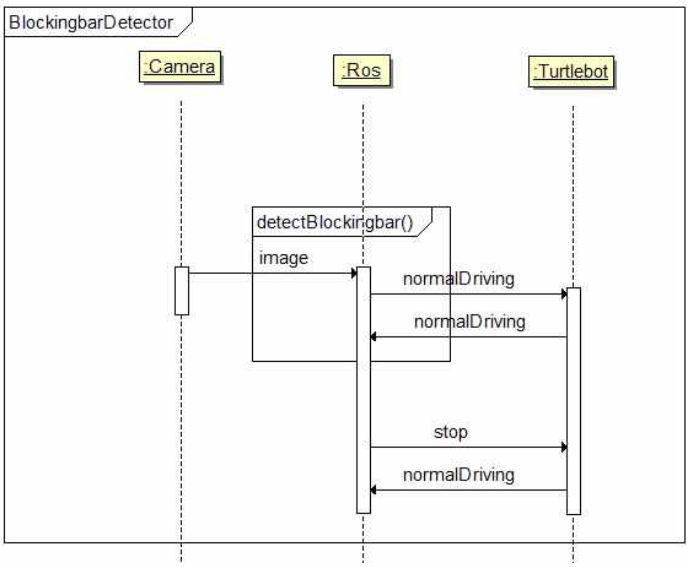
2.2 차단바 인식

2.2.1 분석

2.2.1.1 확장유스케이스

유스 케이스	차단바 인식	
액터	정면 카메라, 터틀봇	
목적	차단바의 존재 여부에 따라 출발 및 정지 상태를 결정한다.	
개요	차단바가 있으면 정지하고 없으면 출발하는 터틀봇의 기능을 구현한다.	
유형	기본, 핵심	
패턴	② State 패턴	
참조	Programming Robots with ROS	
이벤트 흐름 (flow of events) 또는 주흐름 (main flow, basic flow)	액터	Robot Operating System
	① 터틀봇이 출발선에 위치함으로써 유스케이스가 시작된다.	
	② 정면 카메라에서 영상을 받아온다.	
	④ 정지 상태를 유지한다.	③ 차단바가 있다고 인식한다.
	④ 출발 상태가 된다.	③ 차단바가 없다고 인식한다.

2.2.1.2 시스템 순차 다이어그램



2.3 차선 주행 유스케이스

2.3.1 분석

2.3.1.1. 확장 유스케이스

유스 케이스	차선 주행 유스케이스	
액터	우측 카메라, 좌측 카메라, 터틀봇	
목적	차선을 넘지 않고 차선을 따라 이동하게 하기 위함이다.	
개요	차선의 곡률에 상관없이 차선과 일정거리를 유지하며 주행하는 터틀봇의 기능을 구현한다.	
유형	기본, 핵심	
패턴		
참조	Programming Robots with ROS	
이벤트 흐름 (flow of events) 또는 주흐름 (main flow, basic flow)	액터	Robot Operating System
	① 터틀봇이 출발선에 위치함으로써 유스케이스가 시작된다.	
	② 우측 카메라와 좌측 카메라에서 영상을 받는다.	③ 차선을 인식한다.
		④ 좌측과 우측 화면에서 무게중심을 구한다.
		⑤ 양쪽 무게 중심의 평균값에 적절한 연산처리를 한 후 그 값을 publish한다.
	⑥ publish한 값에 따라 로봇이 주행한다.	
		⑦ 정지선을 인식한다.
	⑧ 로봇이 3초간 정지한다.	
		⑨ 3초 후 ④번으로 되돌아 간다.
	⑩ 마지막 정지선때 까지 유스케이스를 반복한다.	

2.3.2 설계

2.3.2.1 시스템 오퍼레이션 정의

시스템 오퍼레이션 정의	
이름	Lane_follower(Scan_image)
책임	스캔한 이미지에서 차선을 인식한다. 차선을 인식하면 무게중심을 구한 뒤 주행한다.
유형	소프트웨어 클래스
참조	유스케이스: 차선 주행
주석	이미지를 받을 때마다 실행된다.
예외조건	3개가 인식될 경우 가까이에 있는 차선으로 인식한다.
출력	해당사항 없음
사전조건	차선을 찾을 카메라가 있어야 한다.
사후조건	차선이 곡선구간일 경우 err값을 변경하여 가운데로 이동하도록 한다(애트리뷰트 변경)

2.4 T자 주차

2.4.1 분석

2.4.1.1. 확장 유스케이스

유스케이스	T자 주차 유스케이스	
액터	우측 카메라, 좌측 카메라, 터틀봇	
목적	벽에 부딪히지 않고 주차를 할 수 있도록 하기 위함이다.	
개요	터틀봇이 T자 주차코스로 들어가고 주차한 후, 코스를 나갈수 있도록 한다.	
유형	보조	
패턴		
참조	Programming Robots with ROS	
이벤트 흐름 (flow of events) 또는 주흐름 (main flow, basic flow)	액터	Robot Operating System
	① 터틀봇이 T자 주차 코스의 정지선을 인식하면서 유스케이스가 시작된다.	
		② 차선을 인식한다.
	③ 차선을 따라 이동한다. 이때 터틀봇은 T자 주차 안까지 이동한다.	④ T자 주차선의 안쪽 선을 인식한다.
	⑤ 정지한다.	
	⑥ 180도 회전을 한다.	
	⑦차선을 따라 이동한다.	

2.4.2 설계

2.4.2.1 시스템 오퍼레이션 정의

시스템 오퍼레이션 정의	
이름	T_Parking
책임	T형 주차구간에서 정지후 다시 되돌아 나오도록 한다.
유형	시스템
참조	유스케이스: T자 주차
주석	3면이 막힐 경우 사용한다.
예외조건	해당사항 없음
출력	해당사항 없음
사전조건	T_Parking이 다른 상황에서 발생하지 않아야 한다.
사후조건	angular.z를 변경하여 터틀봇이 180도 회전하도록 한다.(애트리뷰트 변경)

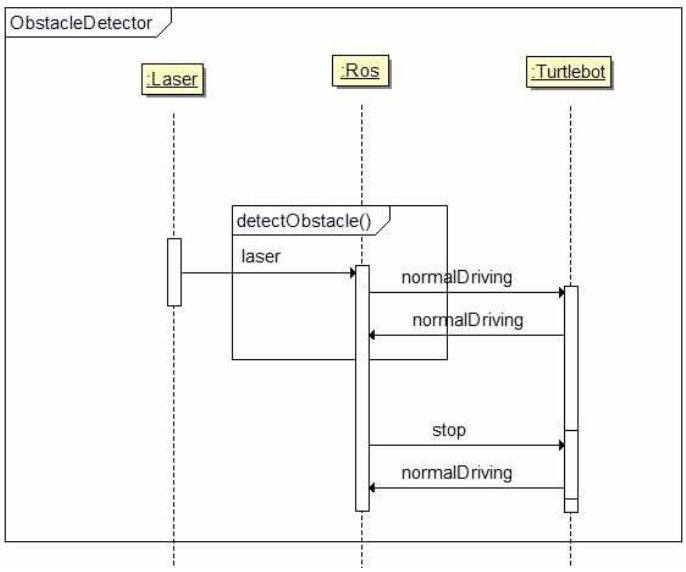
2.5 장애물 인식

2.5.1 분석

2.5.1.1. 확장 유스케이스

유스 케이스	장애물 인식 유스케이스	
액터	레이저 스캔, 터틀봇	
목적	터틀봇이 장애물과 추돌 사고가 생기지 않게하기 위함이다.	
개요	터틀봇이 장애물에 부딪히지 않고 주행을 할 수 있도록 한다.	
유형	기본, 핵심	
패턴		
참조	Programming Robots with ROS	
이벤트 흐름 (flow of events) 또는 주흐름 (main flow, basic flow)	액터	Robot Operating System
		① 레이저 센서에 터틀봇이 인식되면 유스케이스가 시작된다.
		② 터틀봇의 앞에 장애물이 있다면 장애물과의 거리를 계산한다.
		③ 장애물이 터틀봇의 앞에서 없어질 때까지 기다린다.
		④ 장애물이 터틀봇의 앞에 없다고 판단한다.
	⑤ 차선을 따라 이동한다.	

2.5.1.2 시스템 순차 다이어그램



2.5.2 설계

2.5.2.1 시스템 오퍼레이션 정의

시스템 오퍼레이션 정의	
이름	BlockingBar_Detector(Scan_image)
책임	스캔한 이미지에서 차단바를 인식한다. 차단바를 인식하면 정지를 하며 인식하지 못하면 직진한다.
유형	시스템
참조	유스케이스: 차단바 인식
주석	이미지를 받을 때마다 실행된다.
예외조건	해당사항 없음
출력	해당사항 없음
사전조건	차단바를 인식할 조건을 설정해 두어야 한다.
사후조건	차단바를 인식하면 self.drive.stop_sign() 인식하지 못하면 self.drive.go_sign()을 실행한다.

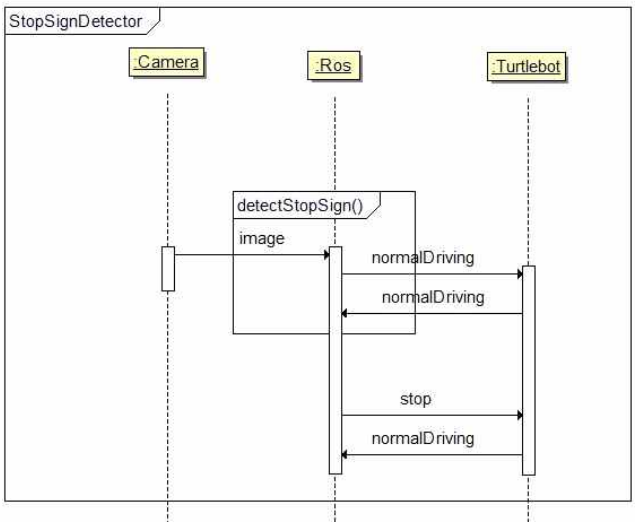
2.6 표지판 인식

2.6.1 분석

2.6.1.1. 확장 유스케이스

유스케이스	표지판 인식 유스케이스	
액터	정면 카메라, 터틀봇	
목적	정지 표지판을 인식한다.	
개요	정지판 이미지 정보를 가지고 정지 표지판인지 여부를 판단한다.	
유형	기본, 핵심	
패턴		
참조	Programming Robots with ROS	
이벤트 흐름 (flow of events) 또는 주흐름 (main flow, basic flow)	액터	Robot Operating System
		① 정면 카메라에 표지판이 인식되면 유스케이스가 시작된다.
		② 인식되는 객체와 저장되어 있는 표지판의 이미지를 비교한다.
	③ 표지판이라고 판단되면 3초간 정지한다.	
대체 이벤트	③ 표지판이 아니라고 판단되면 주행한다.	

2.6.1.2 시스템 순차 다이어그램



2.6.2 설계

2.6.2.1. 시스템 오퍼레이션 정의

시스템 오퍼레이션 정의	
이름	Stop_line_detactor(Scan_image)
책임	정지 표지판을 인식하면 터틀봇이 멈추도록 작동해야 한다.
유형	소프트웨어 클래스
참조	유스케이스: 표지판 인식
주석	이미지를 받을 때마다 실행된다.
예외조건	해당사항 없음
출력	해당사항 없음
사전조건	정지표지판을 비교 할 수 있도록 표지판 이미지를 미리 저장해야 한다.
사후조건	stop_sign이미지와 비교를 한다.

2.7 미로 탈출 전체 유스케이스

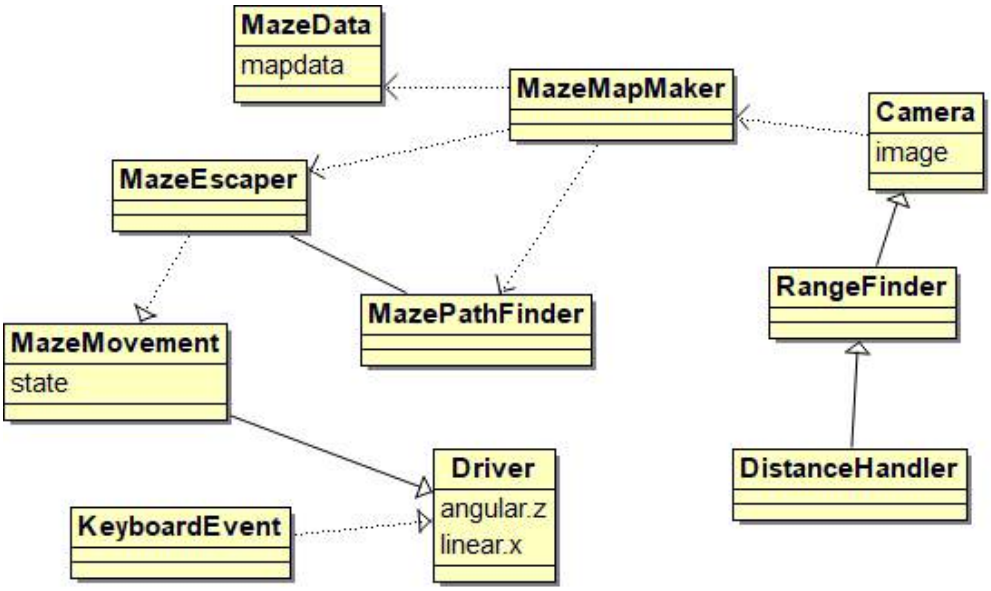
2.7.1 분석

2.7.1.1. 확장 유스케이스

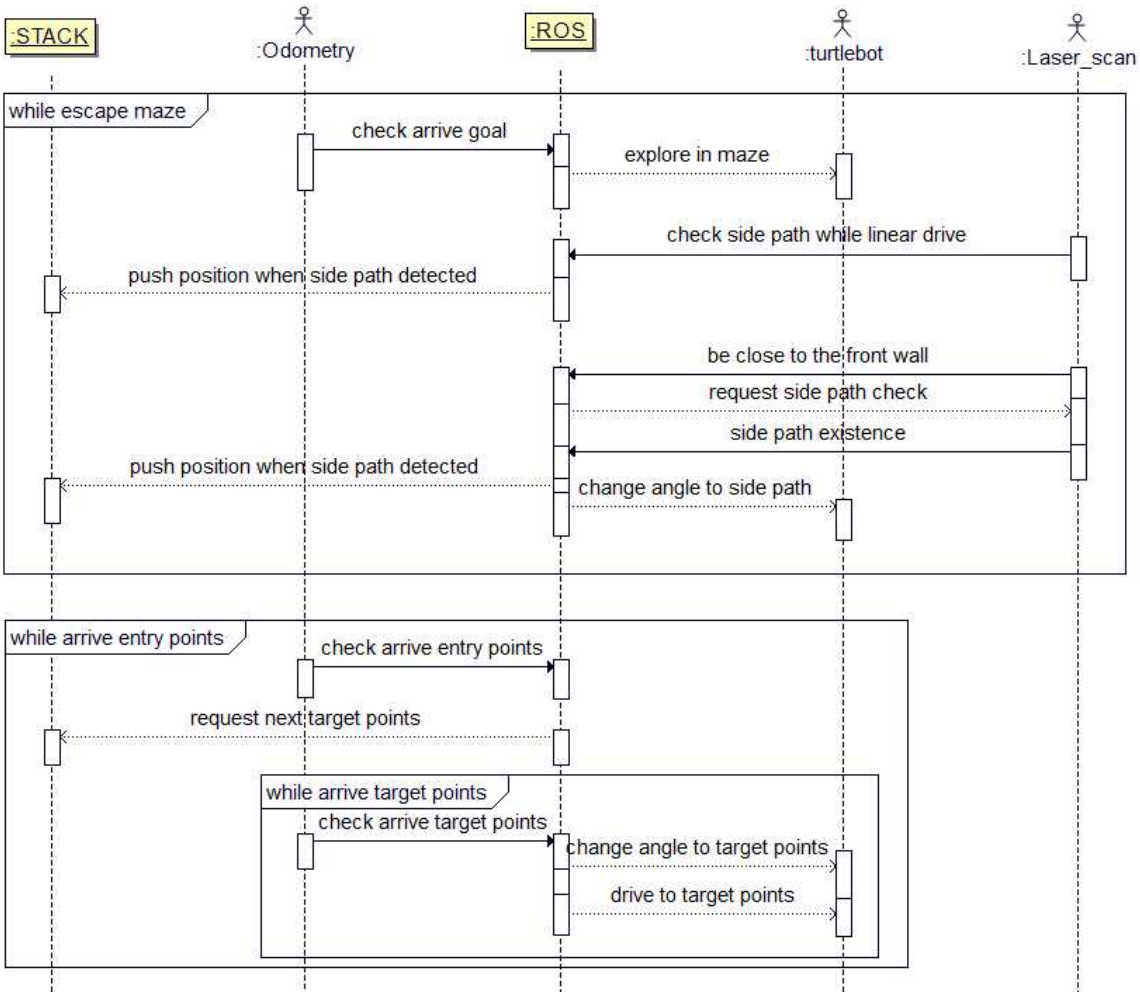
유스케이스	미로 탈출 전체 유스케이스
액터	레이저 스캔, 터틀봇
목적	미로의 벽에 부딪히지 않고 미로를 탈출한 뒤 최단거리로 진입지점으로 되돌아가는 터틀봇을 구현한다.
개요	벽에 부딪히지 않고 미로를 탈출 할 수 있으며 진행 경로를 저장할 수 있는 터틀봇을 구현한다.
유형	기본, 핵심

패턴		
참조	Programming Robots with ROS	
이벤트 흐름 (flow of events) 또는 주흐름 (main flow, basic flow)	액터	Robot Operating System
	① 터틀봇이 직선으로 주행한다.	
	② 레이저 센서 입력값을 확인한다.	② 주행 중 측면 벽과 가까워지면 벽과 반대방향으로 회전하며 진행한다.
		② 주행 중 측면에 통로가 있는 것을 확인하고 해당 위치 정보를 저장한다.
		② 주행 중 정면에 벽이 있으면 위치정보 저장 및 정지한 후 양 측면에 통로 유무를 확인하여 진행 가능한 통로로 진입한다.
		③ 목표좌표와 현재좌표를 비교하여 미로 탈출 여부를 확인한다.
		④ 미로탈출 성공 시 미로에 다시 진입한다.
		⑤ 저장해둔 위치 정보를 이용해 진입지점으로 돌아간다.
대체 이벤트	② 양쪽 통로가 모두 비어있을 경우 왼쪽 통로에 먼저 진입한다. ② 진행 가능한 통로가 없을 경우 양쪽 통로가 모두 진행 가능했던 지점까지 되돌아간다. ② 저장해둔 위치정보들 중 양쪽 통로가 모두 진행 가능했던 지점이 없을 경우 직선 주행 중 미방문했던 통로에 이동해 진입한다. ③ 목표지점에 도착하지 않았을 경우 ①, ②를 반복한다.	

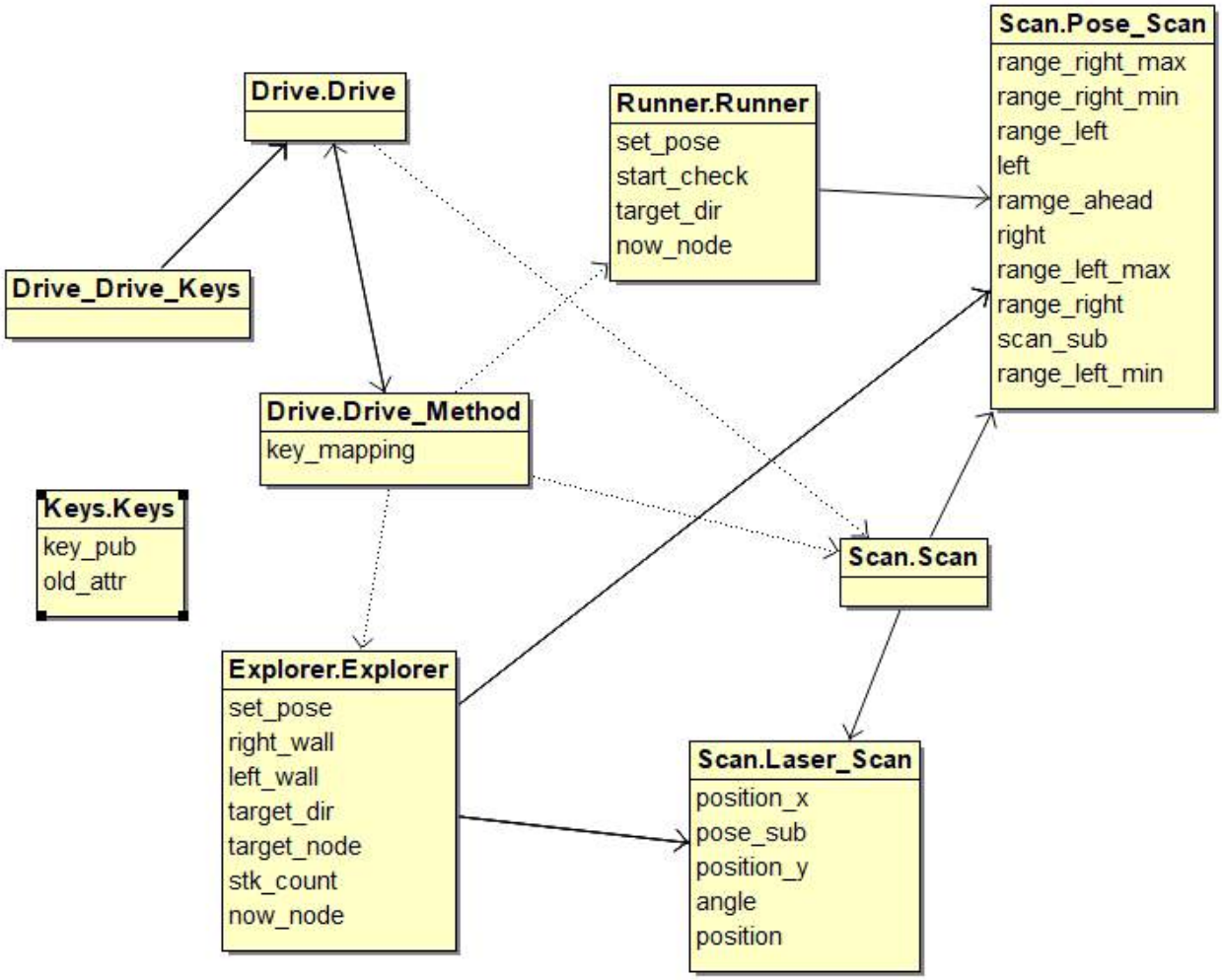
2.7.1.2 본문 분석



2.7.1.3 시스템 순차 다이어그램

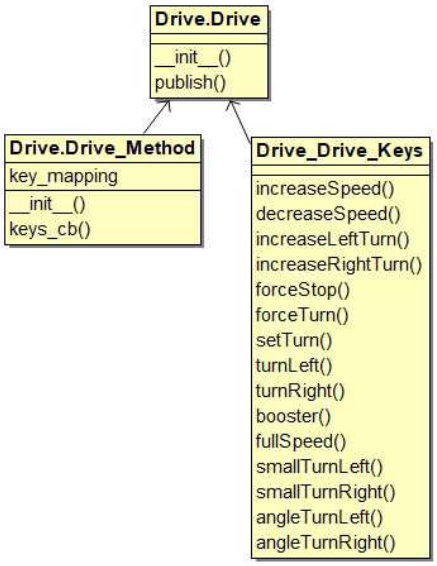


2.7.1.4 도메인 모델

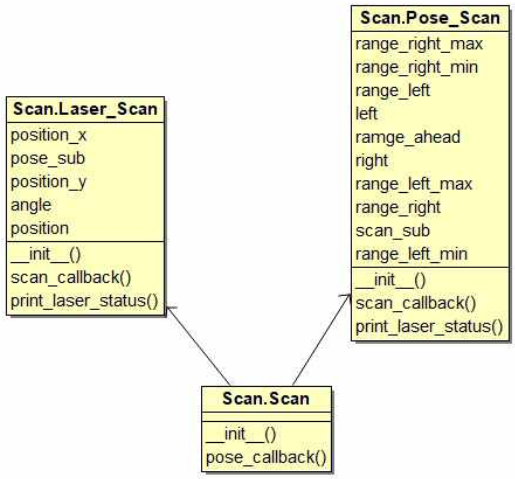

Drive.Drive
Runner.Runner
set_pose
start_check
target_dir
now_node
Scan.Pose_Scan
range_right_max
range_right_min
range_left
left
ramge_ahead
right
range_left_max
range_right
scan_sub
range_left_min
Drive_Drive_Keys
Drive.Drive_Method
key_mapping
Keys.Keys
key_pub
old_attr
Scan.Scan
Explorer.Explorer
set_pose
right_wall
left_wall
target_dir
target_node
stk_count
now_node
Scan.Laser_Scan
position_x
pose_sub
position_y
angle
position

2.7.2 설계

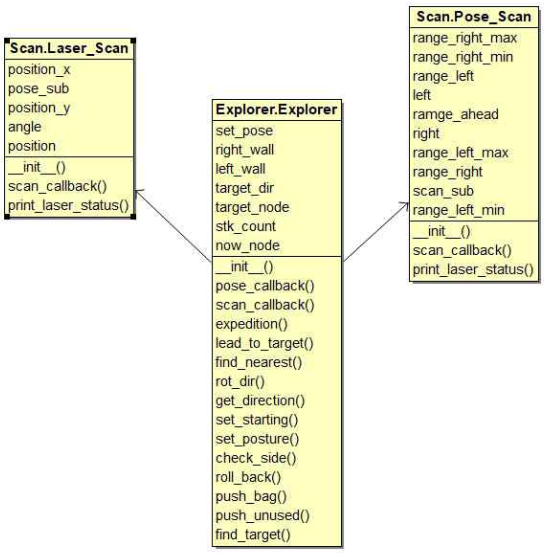
2.7.2.1 설계 클래스 다이어그램



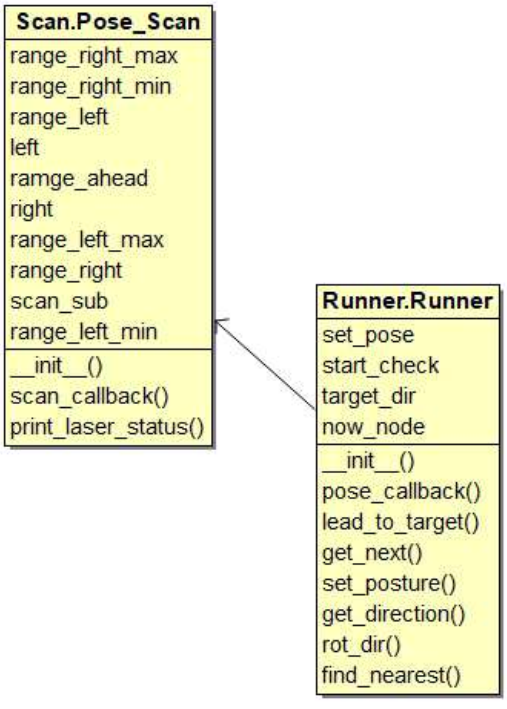
미로 주행 클래스



미로의 좌표 및 레이저 스캔 클래스



미로 탈출 클래스



미로 최단 거리 탈출 클래스

3. 프로젝트 결과

3.1 프로젝트 완성도

프로젝트 기능	완성도
카메라에서 이미지 정보를 받아온다.	100%
레이저 센서로 물체와의 거리 정보 및 벽과의 거리를 받아온다.	100%
시간을 잴 수 있다.	80%
움직이거나 멈출 수 있다.	100%
주행시험장에서 코너 길을 차선을 넘지 않고 회전주행 할 수 있다.	98%
1차선,2차선 코스별로 갈 수 있다.	60%
차선을 검출할 수 있다.	100%
로봇이 장애물을 인식하여 장애물과 부딪히지 않고 주행할 수 있다.	90%
정지선에서 3초간 대기한다.	90%
특정 색을 검출할 수 있다.	100%
로봇이 벽을 인식하여 벽에 부딪히지 않는다.	98%
차단바를 인식하여 차단바에 따라 출발여부를 결정할 수 있다.	100%
STOP sign을 인식할 수 있고 인식하여 3초간 정지할 수 있다.	100%
자율 주행이 가능하다.	70%
T자 주차를 할 수 있다.	90%
평행 주차를 할 수 있다.	40%

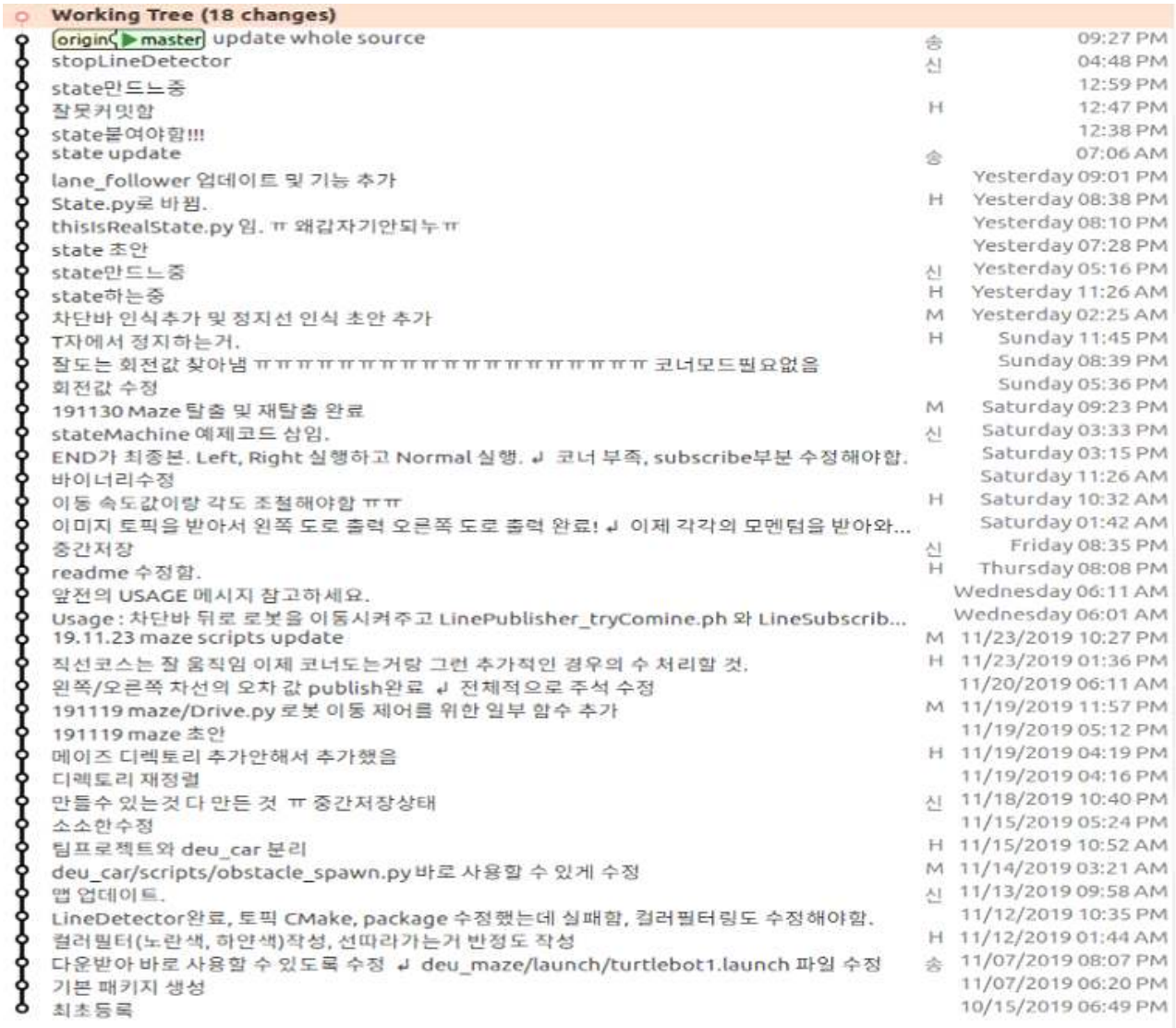
가장 작은 회전각으로 회전할 수 있다.	100%
현재 좌표에서 목표좌표까지 각도를 알 수 있다.	100%
저장해놓은 좌표간의 이동이 가능하다.	100%
현재 로봇이 위치한 좌표를 알 수 있다.	100%
미로탈출이 가능하다.	98%
차선과 일정 거리 유지한다.	100%
탈출 후 최단거리로 다시 돌아올 수 있다.	98%
양옆의 통로 유무를 알 수 있다.	100%
로봇이 진행했던 좌표를 알 수 있다.	100%
분기점에서 탐색하지 않은 곳의 진입점의 정보를 저장할 수 있다.	100%

3.2 일정 계획 평가

3.2.1 WBS

Activity No.	이름	소요기간 (주)	의존성 (선행 작업)
A	시작	0	-
B	요구사항 탐구	0	A
C	계획서 작성	1	B
D	일정 계획	1	C
E	개발 환경 준비	1	A
F	프로젝트 구조 분석	1	B, C

G	프로젝트 기능 상세분석	1	F
H	클래스 구조 설계	1	G
I	클래스 다이어그램 작성	1	H
J	이동 및 멈춤 구현	1	I
K	코너 회전 주행 기능 구현	1	J
L	카메라를 통한 이미지 정보 기능 구현	1	I
M	레이저 센서 (벽, 거리 인식) 기능 구현	1	I, L
N	영상 처리 기능 구현	1	I, L
O	선 인식 구현	1	L, N
P	(장애물 및 차단바) 물체인식 기능 구현	1	L, M, N
Q	타이머 기능 구현	1	I, J
R	좌표 파악 및 응용 기능 구현	1	I, J, M
S	미로 탈출 및 알고리즘 구현	1	I, J, K, R
T	주차 기능 구현	1	I, J, M, O
U	기능 별 단위 테스트	0.5	S, T
V	프로젝트 통합 테스트	0.5	U
W	최종 검토 및 수정	0.5	V
X	보고서, 발표자료 작성	0.5	W
Y	프로젝트 종료	0	-



일정 계획 평가 내용

- 최초 계획서에 작성하였던 프로젝트 계획일정에서 변동이 발생하였다.
- 분석, 설계 이후 구현 과정에서 계획 대비 구현 기간을 더 길게 잡아 프로젝트를 진행하였다.
- O, P, T 단계에서 프로젝트 진행 중 수정사항의 발생으로, 예상 소요시간보다 실제 작업 시간이 조금 더 소요되었다.
- 일정 계획 대비 달성도는 약 85%이다.

3.3 역할 수행 평가

이름	내 용	점수
김현직	클래스 다이어그램, ppt 제작	50
송민광	maze &car 코딩	100
신휘정	car 코딩 및 보고서 작성	100
안지영	ppt 및 보고서 작성	100

3.4 설계 구성요소

항목	내 용
분석	▣ **Deu_maze** ● 벽과의 거리 인식 기능 : 레이저 센서로 로봇과 벽과의 거리를 인식하는 기능을 수행하기 위한 이벤트 흐름을 기술함. ● 로봇 좌표 파악 기능 : 현재 로봇이 위치한 좌표를 파악하는 기능을 수행하기 위한 이벤트 흐름을 기술함. ● 저장해놓은 좌표간의 이동 기능 : 미로 주행 시 위치정보를 미리 저장하여 원하는 좌표로 이동하는 기능을 수행하기 위한 이벤트 흐름을 기술함. ● .미로 탈출 기능 : 로봇의 자율주행으로 미로탈출이 가능한 기능을 수행하기 위한 이벤트 흐름을 기술함. ● 최단거리 주행 기능 : 로봇의 미로탈출 후 최단거리로 돌아가는 기능을 수행하기 위한 이벤트 흐름을 기술함. ● 통로 확인 기능 : 미로주행 시 로봇의 양옆의 통로 유무 여부를 확인하는 기능을 수행하기 위한 이벤트 흐름을 기술함. ▣ **Deu_car** ● 카메라 응용 기능 : 카메라를 통해 영상을 받아오는 기능을 수행하기 위한 이벤트 흐름을 기술함. ● 차단바 인식 기능 : 주행 시 차단바를 인식하여 차단바의 여부에 따라 주행 또는 멈추는 기능을 수행하기 위한 이벤트 흐름을 기술함. ● 거리 정보 파악 기능 : 레이저 센서로 로봇과 차선과의 거리를 인식하여 응용하는 기능을 수행하기 위한 이벤트 흐름을 기술함. ● 차선 인식 및 검출 기능 : 차선의 곡률에 상관없이 차선과 일정 거리를 인식 및 유지기능을 수행하기 위한 이벤트 흐름을 기술함. ● 주차 기능 : T자 주차코스 진입 시 코스대로 주차하는 기능을 수행하기 위한 이벤트 흐름을 기술함. ● 장애물 인식 기능 : 로봇이 장애물을 인식하여 부딪히지 않고 주행하는 기능을 수행하기 위한 이벤트 흐름을 기술함.

제작	● 강의 PPT 활용 – 설명 및 예제를 통해 프로젝트에 응용할 수 있었음. ● 교재 활용 – Programming Robots with ROS 서적을 통해 프로젝트의 전반적인 이해와, OpenCV에 대해 응용 및 구현할 수 있었음.
시험	**▣ Deu_maze** ● 벽과의 거리 인식 기능 : 기능을 시험하기 위하여 명세기반 테스트 기법인 경계값 분석 기법을 사용하였다. ● 로봇 좌표 파악 기능 : 기능을 시험하기 위하여 명세기반 테스트 기법인 동등 분할 기법을 사용하였다. ● 저장해놓은 좌표간의 이동 기능 : 기능을 시험하기 위하여 명세기반 테스트 기법인 동등 분할 기법을 사용하였다. ● 미로 탈출 기능 : 기능을 시험하기 위하여 구조기반 테스트 기법인 조건 테스팅 기법을 사용하였다. ● 최단거리 주행 기능 : 기능을 시험하기 위하여 명세기반 테스트 기법인 경계값 분석 기법을 사용하였다. ● 통로 확인 기능 : 기능을 시험하기 위하여 명세기반 테스트 기법인 경계값 분석 기법을 사용하였다. **▣ Deu_car** ● 카메라 응용 기능 : 기능을 시험하기 위하여 구조기반 테스트 기법인 결정 테스팅 기법을 사용하였다. ● 차단바 인식 기능 : 기능을 시험하기 위하여 구조기반 테스트 기법인 조건 테스팅 기법을 사용하였다. ● 거리 정보 파악 기능 : 기능을 시험하기 위하여 구조기반 테스트 기법인 조건 테스팅 기법을 사용하였다. ● 차선 인식 및 검출 기능 : 기능을 시험하기 위하여 구조기반 테스트 기법인 조건테스팅 기법을 사용하였다. ● 주차 기능 : 기능을 시험하기 위하여 구조기반 테스트 기법인 조건테스팅 기법을 사용하였다. ● 장애물 인식 기능 : 기능을 시험하기 위하여 구조기반 테스트 기법인 조건테스팅 기법을 사용하였다.

3.5 현실적 제한조건

항목	내 용
생산성	● Gazebo를 통하여 시뮬레이션 상의 로봇을 작동 시킨다. ● Rviz를 통하여 로봇의 시점에서 보여지는 이미지를 응용한다. ● SmartGit 형상관리 도구를 활용하여 프로젝트 팀원사이에서 소스 코드 공유가 원활하게 진행되었고 commit message를 이용하여 추가와 변경을 확인할 수 있다.

산업표준	● 개발 단계에서 테스팅시 ISO/IEC/IEEE 29119 SW 테스팅 국제 표준에서 정의하는 명세기반 테스트 설계 기법(동등분할, 경계값 분석), 구조기반 테스트 기법(결정 테스팅, 조건 테스팅)을 사용하였다. ● SmartGit을 사용하여 형상 관리 기법을 적용하였다.

4. 소감

김현직	난이도 높은 프로젝트에서 포기하지않고 끝까지 노력해준 팀원들에게 감사를 표합니다. 새로운 것을 배우는 학업의 길은 쉽지만은 않은 도전인 것 같습니다.
송민광	프로젝트를 진행하면서 처음 시작은 파이썬 문법이 익숙하지 않아 많이 해매고 ROS와 터틀봇간의 작동 방식을 제대로 이해하지 못하여 많은 어려움이 있었습니다. 이번 프로젝트에서는 내가 만든 알고리즘과 코드로 로봇을 구동할 수 있다는 것이 정말 흥미로웠고 opencv를 활용하여 이미지를 처리하는 여러가지 방법들도 흥미로운 것들이 많았습니다. 프로젝트 초반에 미로 탈출에 빠져 운전 면허 시험장 프로젝트를 시간을 많이 투자하지 못 한 것과 스테이트 머신을 제대로 구현하지 못한 것이 너무 아쉬웠고 프로젝트를 깔끔하게 마무리 하지 못하여 너무 아쉽습니다.
신휘정	아쉬움이 많이 남습니다. 나름 미리 시작해서 이것 저것 예제도 돌려보고 해보았지만 새로운 언어에 대한 레퍼런스를 찾는 것은 어려웠고 어디서 어떻게 시작 해야 할지 많이 어려웠던 프로젝트였습니다. 마지막에 많은 사람들과 힘을 합쳐서 이것 저것 해결하면서 사람들의 능력에 감탄도 하고 많은 걸 배우면서도 한편으로는 혼자 노력을 했어도 많이 부족하구나를 느끼게 해주었던 프로젝트입니다. 마지막에 거의 모든 기능을 구현 다 하였지만 StateMachine에서 파이썬의 어려움을 느끼며 마지막 이어붙이는 제일 중요한 작업을 못하고 마무리한 것도 안타깝습니다.

안지영	프로젝트를 진행하면서 여러 어려움을 많이 겪어 문제점이 꽤나 발생하고 부족함을 많이 느꼈지만, 훌륭한 팀원들 덕분에 많이 배워가는 경험이 되었습니다. 기존에 진행하던 프로젝트와는 전혀 다르게 ROS를 통한 자율주행 로봇 프로그래밍을 접하게 되어 흥미롭기도 하고 또 제가 계획했던 일들을 진행하지 못 해 상당히 어려웠지만, 열과 성을 다해 진행해준 팀원들에게 고마움을 많이 느꼈습니다. 더더욱 노력하고 학습해야겠다는 생각이 들었습니다.