

# 2022년 한이음 ICT멘토링 프로젝트 결과보고서

프로젝트명

Unity ML-Agent를 이용한 이동체 로봇 주행 기술 개발

## 요 약 본

프로젝트 정보	
주제영역	<input type="checkbox"/> 생활 <input checked="" type="checkbox"/> 업무 <input checked="" type="checkbox"/> 공공/교통 <input type="checkbox"/> 금융/핀테크 <input type="checkbox"/> 의료 <input type="checkbox"/> 교육 <input type="checkbox"/> 유통/쇼핑 <input type="checkbox"/> 엔터테인먼트
기술분야	<input type="checkbox"/> IoT <input type="checkbox"/> 모바일 <input type="checkbox"/> 데스크톱 SW <input checked="" type="checkbox"/> 인공지능 <input type="checkbox"/> 보안 <input type="checkbox"/> 가상현실 <input type="checkbox"/> 빅데이터 <input type="checkbox"/> 자동제어기술 <input type="checkbox"/> 블록체인 <input type="checkbox"/> 영상처리
달성성과	<input checked="" type="checkbox"/> 논문게재 및 포스터발표 <input type="checkbox"/> 앱등록 <input type="checkbox"/> 프로그램등록 <input type="checkbox"/> 특허 <input type="checkbox"/> 기술이전 <input type="checkbox"/> 실용화 <input checked="" type="checkbox"/> 공모전(한이음 ICT 공모전) <input type="checkbox"/> 기타( )
프로젝트명	Unity ML-Agent를 이용한 이동체 로봇 주행 기술 개발
프로젝트 소개	<ul style="list-style-type: none"> <li>- 로봇 분야의 강화 학습을 위해 게임 엔진을 이용해 <b>가상 환경</b>을 만들고 가상 환경에서 로봇을 학습하여 로봇에 적용하는 방법인 <b>sim-to-real</b>을 적용하고자 함</li> <li>- 아직 실제 로봇에 적용하기에 완전한 기술은 아니지만, 선행 기술에 대해 학습해보면서 실제 로봇에 적용해보고자 함</li> <li>- <b>유니티 게임 엔진</b>을 이용하여 <b>가상 환경</b>을 만들고, <b>ML-Agent</b>를 이용하여 가상 환경에서 <b>강화 학습</b>을 하고 학습된 모델을 <u>ROS(Robot Operating System)</u> 기반 로봇에 적용하고자 함</li> </ul>
개발배경 및 필요성	<ul style="list-style-type: none"> <li>- <b>자율 주행</b>이 가능한 로봇에 관한 관심이 증가하면서 다양한 로봇 제품들이 개발되고 있고 이 기술은 서빙 로봇, 배달 로봇 무인 경비 로봇 등 다양한 분야의 핵심기술로서 서비스를 개발하기 위해서 <b>강화 학습</b>을 이용한 연구가 많이 진행되고 있음</li> <li>- 하지만 로봇 분야의 강화 학습은 데이터를 얻기 쉽지 않아 학습에 어려움이 있음</li> </ul>
프로젝트 주요기능	<ul style="list-style-type: none"> <li>- 자율 주행 : ROS를 이용한 로봇 기능 개발</li> <li>- 강화 학습 : ML-Agent를 이용한 학습 모델</li> <li>- 가상 환경 : Unity를 이용한 가상 환경 개발</li> </ul>
작품의 기대효과 및 활용분야	<ul style="list-style-type: none"> <li>- 로봇 티칭 : 가상 환경을 이용한 로봇 티칭에 활용</li> <li>- 서비스 로봇 : 서비스 로봇, 배달 로봇 등 다양한 분야에 활용</li> </ul>

# (본문) 프로젝트 결과보고서

## I. 프로젝트 개요

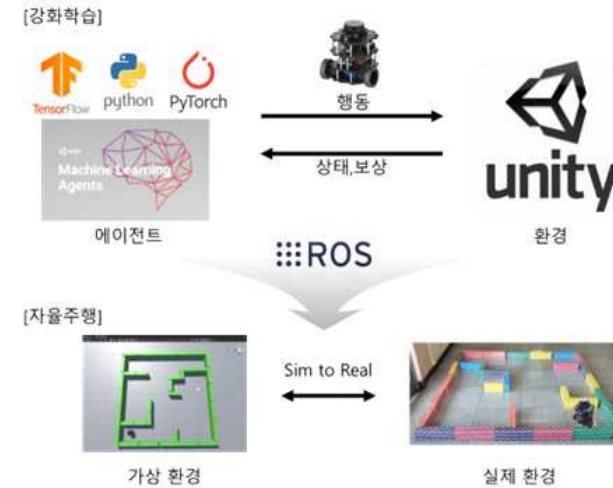
### 가. 프로젝트 소개

- 유니티 게임 엔진을 이용하여 **가상 환경**을 만들고, ML-Agent를 이용해 가상 환경에서 **강화 학습**을 하여 학습된 모델을 ROS(Robot Operating System) 기반 로봇에 적용하고자 한다. 이러한 가상 환경에서의 강화 학습된 모델을 통해 실제 로봇에 적용함으로써 **이동체 로봇 주행 기술**을 개발한다.
- **강화 학습**은 기계 학습의 한 영역으로, 어느 환경 안에서 학습하는 시스템을 **에이전트(Agent)**라고 부르며, 에이전트가 현재의 상태를 인식하여 선택 가능한 행동 중 **보상을 최대화**하는 행동 혹은 행동 순서를 학습하도록 하는 방법이다. 시간이 지나면서 가장 큰 보상을 얻기 위해 최상의 전략을 스스로 학습한다.

### 나. 개발배경 및 필요성

- 최근 **자율 주행**이 가능한 로봇에 관한 관심이 증가하면서 다양한 로봇 제품들이 개발되고 있다. 서빙 로봇, 배달 로봇, 무인 경비 로봇 등 적용 분야가 광범위한 핵심기술이다. 이런 로봇들의 서비스를 개발하기 위해서는 **강화 학습**을 이용한 연구가 많이 되고 있다. 하지만 **로봇 분야의 강화 학습은 데이터를 얻기가 쉽지 않아 학습에 어려움이 많다.**
- 이러한 문제를 보완하기 위해 **게임 엔진을 이용해 가상 환경**을 만들고 가상 환경에서 로봇을 학습하여 로봇에 적용하는 방법인 **sim-to-real**을 많이 연구하고 있다. 실제 환경과 달리 가상 환경에서는 대량의 데이터를 손쉽게 생성해 낼 수 있고 많은 시행착오를 겪어볼 수 있다는 이점이 있다. 아직 실 로봇에 적용하기에 완전한 기술은 아니지만, 선행 기술에 대해 학습해보면서 **실제 로봇에 적용해 보고자 한다.**

### 다. 작품 구성도



### 라. 작품의 특징 및 장점

- Unity는 다른 시뮬레이션과 달리 ROS 연동이 손쉽고 다양한 예셋이 존재하여 개발을 더 쉽고 빠르게 가능함
- 가상환경에서 다양한 환경을 구현 할 수 있기 때문에 새로운 환경에 대비해 학습해 둔다면 현실에서의 다양한 환경에서도 구동이 가능함
- 가상 뿐만 아니라 현실의 데이터도 ROS를 통해 받아와서 학습 데이터로 활용하여 강화학습의 정확도를 높힐 수 있음

## II. 프로젝트 수행결과

### 가. 주요기능

구분	기능	설명
S/W	Unity	가상 환경 구축을 위한 3D 개발환경을 제공하는 게임 엔진
	ML-Agent	지능형 에이전트를 훈련 시키는데 필요한 환경 역할을 할 수 있는 오픈소스 Unity 플러그인
H/W	Pytorch	파이썬을 활용한 딥러닝 네트워크를 구현하는 사용하는 프레임워크
	ROS	로봇 응용프로그램 개발에 필요한 오픈소스 기반 메타 OS
	Turtlebot3	학습된 모델을 적용할 ROS 기반 모바일 로봇
	LiDAR	로봇 주행 시 사물 간의 거리, 형태를 파악하는 센서
	Camera	주행에 필요한 영상처리 수행
	Block	로봇을 구동 시킬 가상 공간의 맵과 동일한 임의의 맵 구축

#### 나. 프로젝트 개발환경

구분	항목	적용내역
S/W 개발환경	OS	Window10, ubuntu 18.04, ubuntu 20.04 (ROS2)
	개발환경(IDE)	Unity 20.3.30, ML-Agent
	개발도구	PC, Raspberry Pi4
	개발언어	Python, C#
H/W 구성장비	디바이스	Turtlebot3, Raspberry Pi4, OpenCR
	센서	LDS-02
	통신	Tcp/Ip
	개발언어	Python, C#

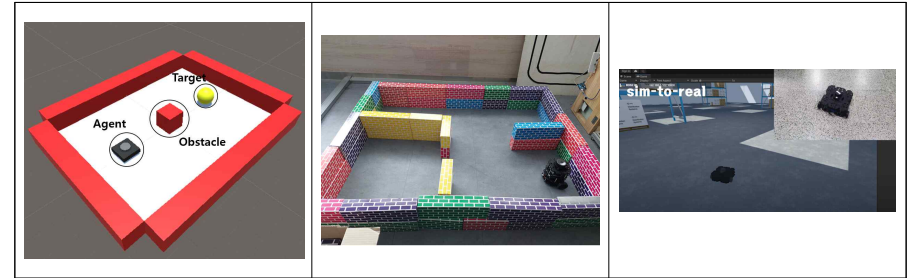
#### 다. 장비(기자재/재료) 활용

번호	품명	작품에서의 주요기능
1	Turtlebot3	- 학습된 모델을 적용할 ROS 기반 모바일 로봇
2	LiDAR	- 로봇 주행 시 사물 간의 거리, 형태를 파악하는 센서
3	Camera	- 주행에 필요한 영상처리 수행

#### 라. 프로그램 작동 동영상

- <https://youtu.be/mLbz10aVMZ8>

#### 마. 결과물 상세 이미지



#### 바. 달성성과

구분	게재(발표)자명	논문(포스터)명	게재(발표)처	게재(발표)일자
□ 논문게재 및 포스터발표	오세환, 박주원, 최홍용	ROS 기반 이동 로봇 강화학습을 위한 시뮬레이션 환경 구축	한국정보기술학회	2022. 12. 02.
□ 앱(APP) 등록	등록자명	앱(APP)명	등록처	등록일자
□ 프로그램 등록	등록자명	프로그램명	등록처	등록일자
□ 특허/실용신안 출원	출원자명	특허/실용신안명	출원번호	출원일자
□ 기술이전	기술이전기업명	기술명	금액	이전일자
□ 공모전	구분(교내/대외)	공모전명	수상여부(출품/수상)	상격
□ 실용화	#실용화한 내용에 대한 구체적 작품설명			
□ 기타				

### III. 프로젝트 수행방법

#### 가. 업무분장

번호	성명	역할	담당업무
1	최홍용	멘 토	- 프로젝트 총괄 및 지도, 업무 수행에 필요한 자료 제공
2	박주환	팀 장	- 학습 모델 로봇 적용 sim-to-real
3	오세현	팀 원2	- ML-agent를 활용한 강화 학습
4	김성훈	팀 원3	- ROS2 기반 모바일 로봇 구동
5	윤정훈	팀 원4	- ROS1 기반 모바일 로봇 구동
6	이진영	팀 원5	- 유니티 가상 환경 구축 및 강화 학습

#### 나. 프로젝트 수행일정

구분	추진내용	수행일정									
		3월	4월	5월	6월	7월	8월	9월	10월	11월	
계획	수행계획 수립 및 프로젝트 개설 신청										
	프로젝트 유사 사례 조사, 차별성 아이디어 구상										
	프로젝트 세부 미팅 계획 설정										
분석	Unity 및 인공지능 스터디										
	ML-agent 예제 분석 및 스터디										
설계	적용 로봇 모델 분석										
	유니티 가상 환경 구축										
개발	ML-agent를 활용한 강화학습										
	로봇에 학습 모델 적용 및 수정										
	알고리즘 수정 및 개선										
테스트	테스트 및 보완 사항 개선										
종료	최종 점검										
온라인 미팅	지도 및 업무 총괄										

#### 다. 문제점 및 해결방안

- 프로젝트 관리 측면
  - 프로젝트 진행에 있어서 다양한 분야를 같이 알아야 하는 작품이고 개인의

능력치가 달라 프로젝트 역할 배분에 있어 어려움이 있었음

- 기존에 경험이 있는 팀원이 새로운 분야를 접하는 인원에게 조언을 해주면서 공부 방향을 잡아 주었고 블렌디드 러닝을 함께 병행하여 프로젝트 수행 진도를 맞출 수 있었음

#### ○ 작품 개발 측면

- ROS나 Unity 등 프로젝트를 진행하면서 통신 시, 각각의 버전이 달라 하나를 맞추면 하나가 안되는 문제가 많이 발생하였음
- 같은 버전에 비슷한 기능을 하는 패키지가 있는지 찾고 설정 등을 비교 분석하여 수정해 나아갔음
- 정기 미팅을 통해 멘토님께 조언을 구하고 하나씩 차근차근 수정해 나가는 방향으로 진행하였음

### IV. 기대효과 및 활용분야

#### ○ 작품의 기대효과

- 가상 환경을 통해 실 환경에서 바로 적용하기 힘든 기술을 미리 적용 시켜볼 수 있음
- 로봇의 경우 실환경에서 필요한 대량의 데이터를 얻기 어렵지만 가상환경에서는 원하는 상황을 재연하여 최대한 많은 데이터를 얻을 수 있음
- 가변적인 실제 공간에서 데이터를 매번 얻기 어려우므로 가상 환경에서의 강화 학습된 모델을 통해 적용된 로봇은 활용도가 높음
- 이러한 인공지능 모델을 활용 시 별도의 추가적인 센서 없이 동작을 가능케 함

#### ○ 작품의 활용분야

- 본 작품은 이동체 로봇에 대한 연구이지만 가상환경을 활용한 강화학습은 4족 보행 로봇, 매뉴플레이터 등 다양한 로봇에 적용이 가능함
- 사용자가 많고 접근성이 좋은 유니티를 활용하여 구현하여 스마트 팩토리 등으로 환경을 변경하면 다양한 응용 분야에서 활용이 가능함

### V. 참고자료

#### 가. 참고 및 인용자료

- [도서] 파이토치와 유니티 ML-Agents로 배우는 강화학습
- [도서] 텐서플로와 유니티 ML-Agents로 배우는 강화학습
- [K-MOOC] ML-Agent와 강화학습

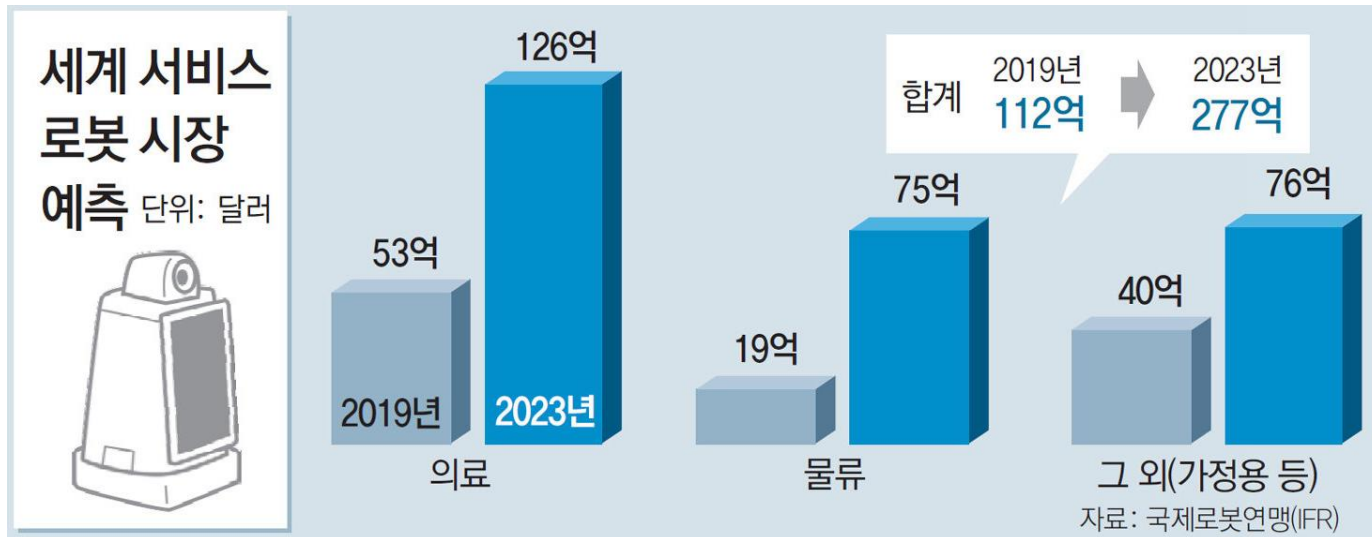


# SW개발/HW제작 설계서

프로젝트 명 : Unity ML-Agent를 이용한 이동체 로봇 주행 기술 개발

2022. 08. 22

## | 시장/기술 동향 분석



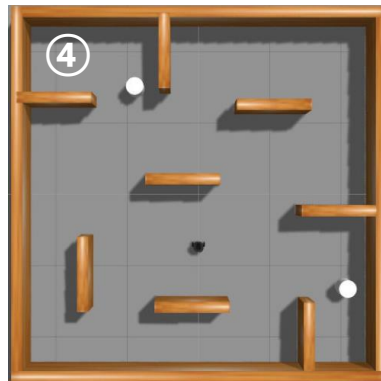
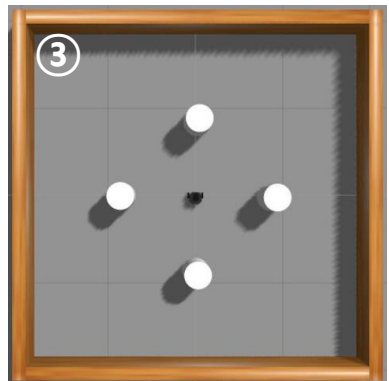
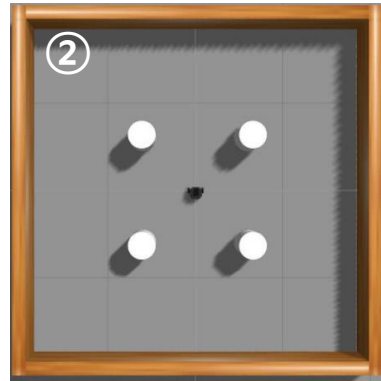
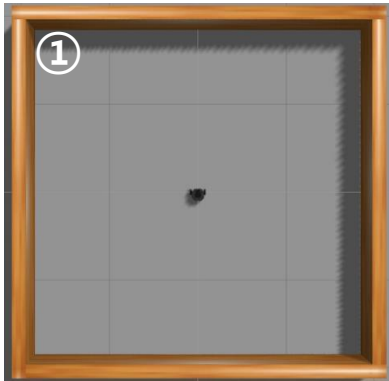
- 서비스 로봇 시장은 급속도로 커갈 것으로 전망된다.
- 지난달 IFR는 지난해 112억 달러(약 12조4992억 원)였던 세계 서비스 로봇 시장 규모가 2012년부터 2025년까지 연평균 25% 이상 성장할 것이라고 분석했다.
- 특히 서비스 로봇으로 분류되는 물류 자동화용 로봇이 연평균 42% 커지며 75억 달러에 이를 전망이다.

## | 요구사항 정의서

구분	기능	설명
S/W	Unity	3D 개발환경을 제공하는 게임 엔진을 활용한 가상 환경 구축
	ML-Agent	지능형 에이전트를 훈련 시키는데 필요한 환경 역할을 할 수 있는 오픈소스 Unity 플러그인
	Pytorch	하우스에 저장되어 있는 식물을 선택해 삭제할 수 있다.
	ROS	로봇 응용프로그램 개발에 필요한 오픈소스 기반 메타 OS

구분	기능	설명
H/W	Turtlebot3	학습된 모델을 적용할 ROS 기반 모바일 로봇
	LiDAR	로봇 주행 시 사물 간의 거리, 형태를 파악하는 센서
	Camera	주행에 필요한 영상처리 수행
	Block	가상 공간의 맵과 동일한 임의의 맵 구축

## | 서비스 구성도 - 서비스 시나리오

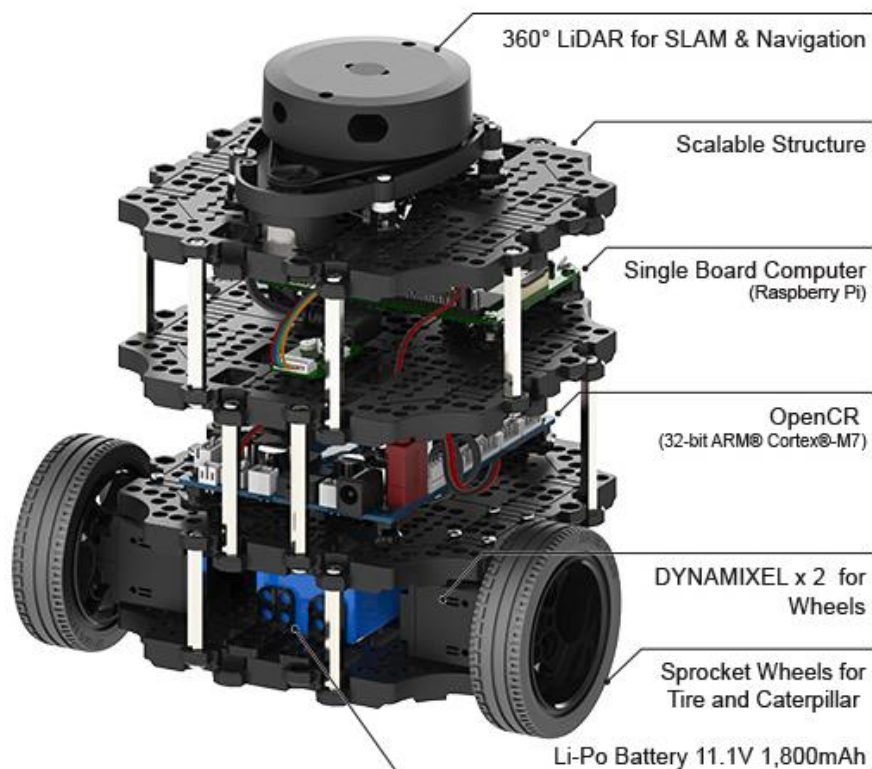


### 자율주행을 위한 강화학습

- ① 실제환경과 유사한 가상환경에서 학습 진행 후 자율주행 동작
- ② 가상환경에서 정적인 장애물을 추가하여 학습 진행 후 자율주행 동작
- ③ 가상환경에서 동적인 장애물을 추가하여 학습 진행 후 자율주행 동작
- ④ 복잡한 가상 환경에서 학습 진행 후 자율주행 동작

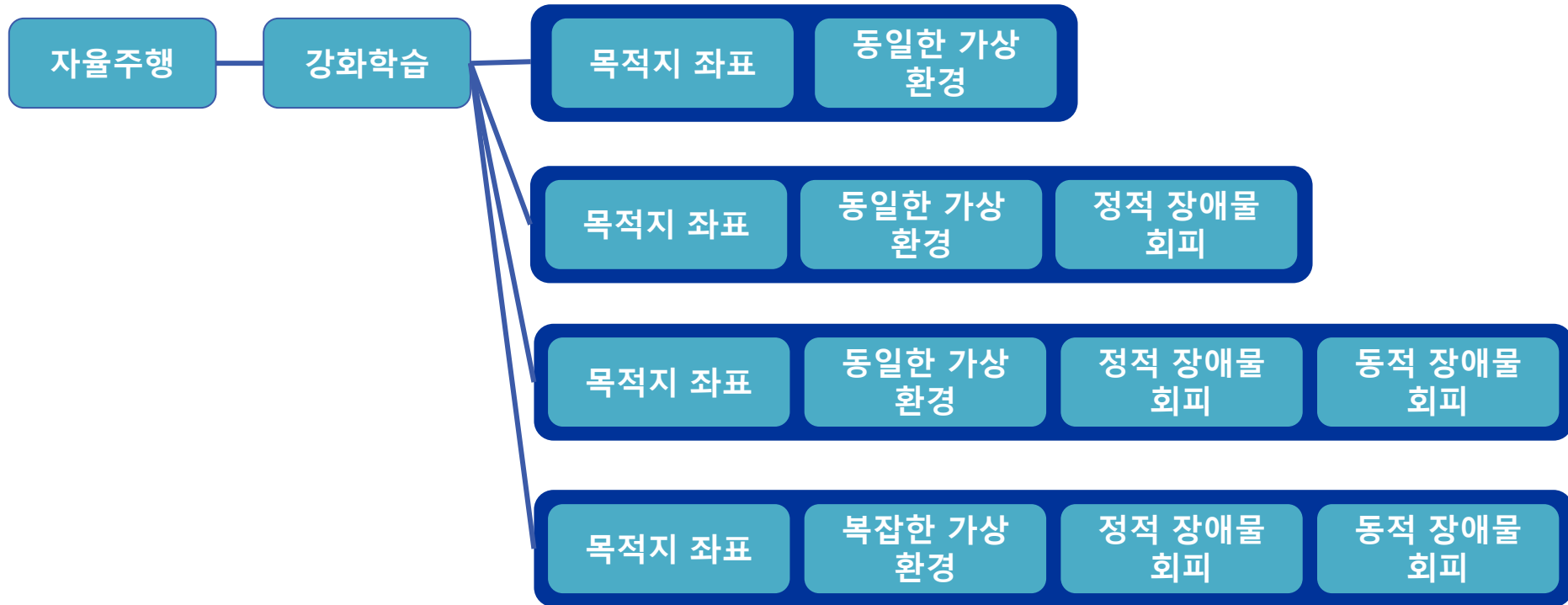


## | 하드웨어/센서 구성도



항목	버거
최대 이동 속도	0.22m/s
최대 회전 속도	2.84rad/s (162.72 deg/s)
최대 가반하중	15kgs
크기 (L x W x H)	138mm x 178mm x 192mm
무게 (+SBC+배터리+센서)	1kg
사용 시간	약 2시간 30분
충전 소요시간	약 2시간 30분
다이나믹셀	XL430-W250-T
SBC	라즈베리 파이
임베디드 컨트롤러	OpenCR (32-bit ARM® Cortex®-M7)
센서	360° LiDAR 3축 자이로 센서 3축 가속도 센서 3축 지자기 센서

## | 메뉴 구성도



## | 기능 처리도(기능 흐름도)



### [강화 학습]

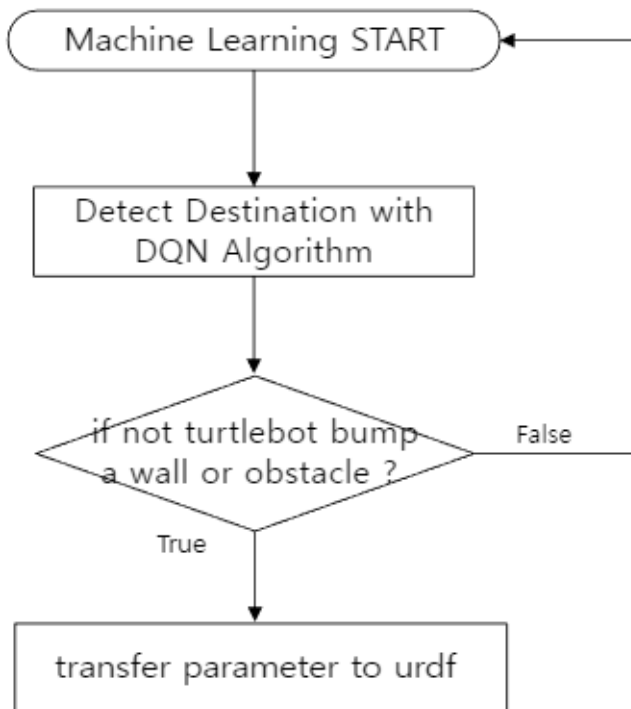
- ① unity에 실제와 유사한 가상 환경 구축
- ② 제작된 환경에 머신러닝 에이전트 적용 및 설정
- ③ 강화학습을 이용하여 에이전트 학습
- ④ 학습이 완료된 경우 에이전트 모델을 다시 유니티에 임베딩
- ⑤ 유니티 환경을 빌드하여 학습된 에이전트를 적용

### [자율 주행]

- ① Unity 와 ROS간의 통신환경 구축
- ② 유니티의 학습된 에이전트와 ROS로 구동되는 실제 로봇과 sim-to-real
- ③ 로봇이 목적지 까지 자율주행 수행

## | 알고리즘 명세서

✓ DQN 알고리즘을 이용한 Turtlebot3강화 학습



✓ 알고리즘 시나리오

- ① Unity ml-agent 환경 설정을 한다.
- ② DQN 코드를 위한 파이썬 라이브러리를 불러온다.
- ③ 파라미터 값을 결정
- ④ Model 클래스에서 CNN 구조 및 손실 함수 값의 계산 및 네트워크 학습을 위한 최적화 기법 결정
- ⑤ Agent 클래스에서 DQN을 위한 행동 선택, 경험 저장, 네트워크 학습 수행을 위한 함수 정의
- ⑥ Main 함수에서 Model, Agent Class를 이용해 행동 결정
- ⑦ 유니티 환경과 통신하여 학습을 수행
- ⑧ 학습이 완료 될 시 ROS로 학습 데이터 전송

## | 알고리즘 상세 설명서

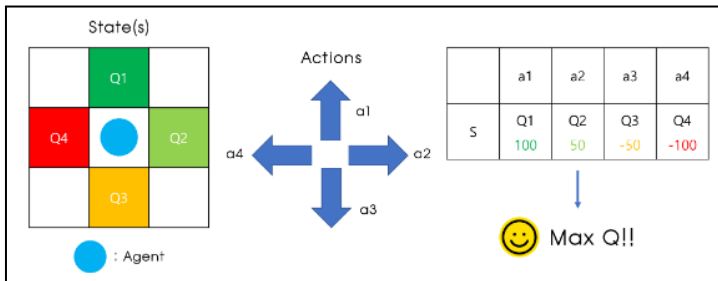
### ✓ 가치 기반 강화학습 – DQN 알고리즘

가치를 기반으로 의사 결정

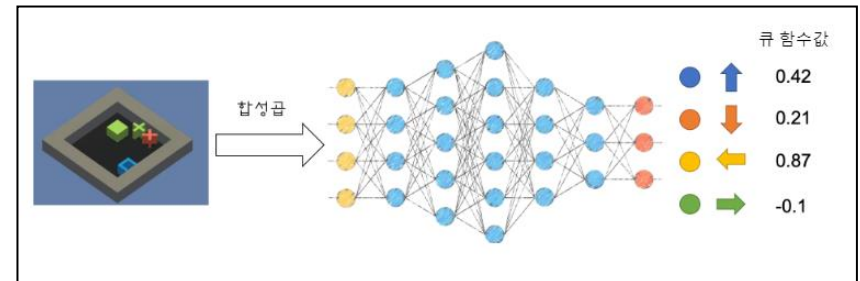
큐 함수를 학습하여 최적의 큐 함수를 얻고 이를 통해 의사결정 수행

CNN을 통해 각 상태와 행동에 대한 큐 함수들을 근사하여 모든 상태와 행동에 대한 큐 함수 값을 따로 저장하지 않고 큐 함수 값에 대한 추정을 수행한다. 그 결과 DQN은 많은 상태가 존재하는 환경에서도 학습이 가능하여 좋은 성능을 보인다.

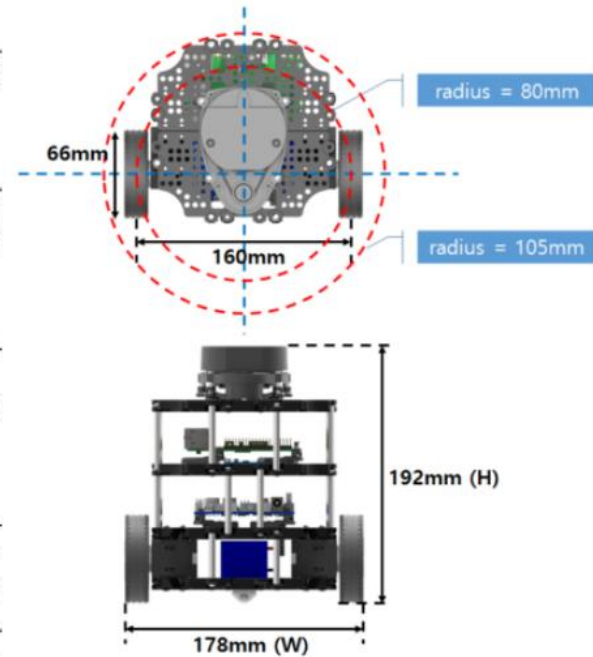
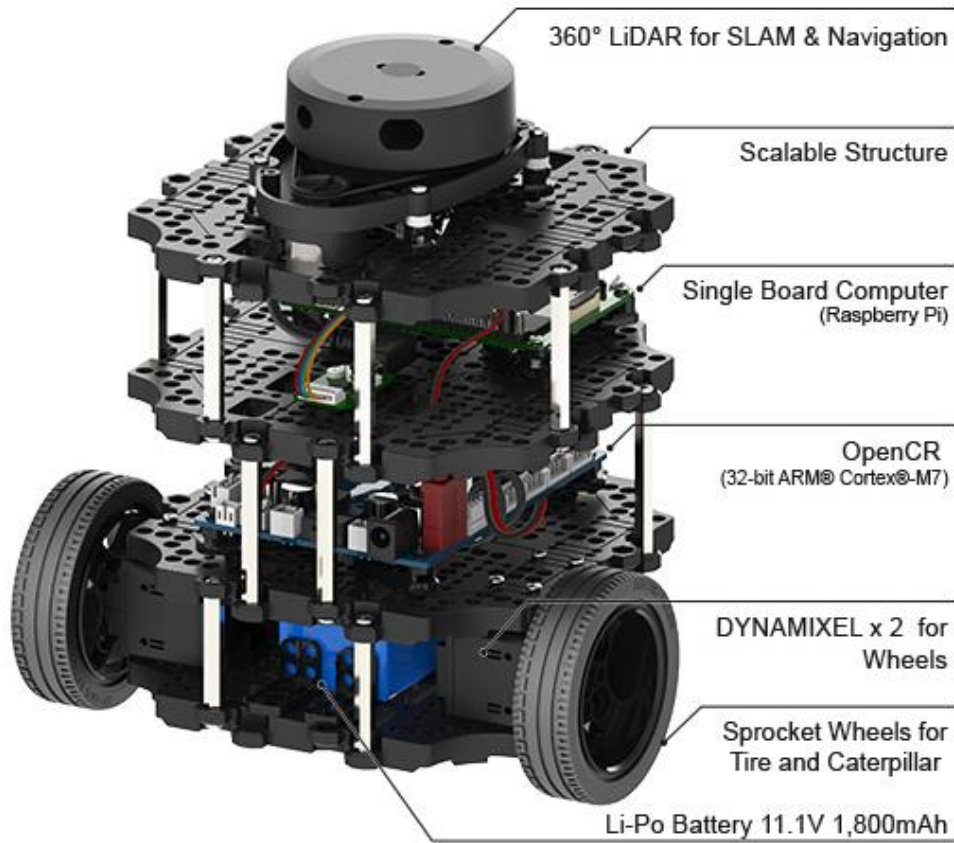
#### 큐함수를 기반으로 행동 선택 (Q-Learning)





#### CNN을 이용해 상태마다 각 행동의 큐함수들을 근사

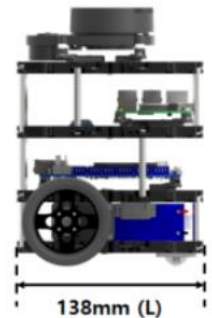


# | 하드웨어 설계도



 = 138 x 178 x 192 (L x W x H, mm)

 = 1 Kg



## | 프로그램 - 목록

기능 분류	기능	기능 명
S/W	Unity	가상 환경 생성
	ML-Agent	강화 학습
	Pytorch	딥러닝 프레임워크
	ROS	로봇 운영 체제
H/W	Turtlebot3	모바일 로봇
	LiDAR	물체 인식 센서
	Camera	영상 처리
	Block	실제 맵 구축

## | 핵심소스코드(1)

### ✓ Model.class 및 DQNAgent.class

TurtleBot3를 강화학습 시킴에 있어 가장 핵심이 되는 클래스들이다. Model 클래스는 DQN에서 사용하는 합성 신경망의 구조를 결정하고 손실함수 값의 계산 및 네트워크 학습을 위한 최적화 기법을 결정한다. Agent 클래스는 DQN을 위한 행동 선택, 경험 저장, 네트워크 학습 수행 등을 위한 다양한 함수를 정의한다.

```
# Model 클래스 -> 합성곱 신경망 정의 및 손실함수 설정, 네트워크 최적화 알고리즘 결정
class Model():
    def __init__(self, model_name):
        self.input = tf.placeholder(shape=[None, state_size[0], state_size[1], |
                                state_size[2]], dtype=tf.float32)
        # 입력을 -1 ~ 1까지 값을 가지도록 정규화
        self.input_normalize = (self.input - (255.0 / 2)) / (255.0 / 2)

        # CNN Network 구축 -> 3개의 Convolutional layer와 2개의 Fully connected layer
        with tf.variable_scope(name_or_scope=model_name):
            self.conv1 = tf.layers.conv2d(inputs=self.input_normalize, filters=32,
                                           activation=tf.nn.relu, kernel_size=[8,8],
                                           strides=[4,4], padding="SAME")
            self.conv2 = tf.layers.conv2d(inputs=self.conv1, filters=64,
                                           activation=tf.nn.relu, kernel_size=[4,4],
                                           strides=[2,2],padding="SAME")
            self.conv3 = tf.layers.conv2d(inputs=self.conv2, filters=64,
                                           activation=tf.nn.relu, kernel_size=[3,3],
                                           strides=[1,1],padding="SAME")

            self.flat = tf.layers.flatten(self.conv3)

            self.fc1 = tf.layers.dense(self.flat,512,activation=tf.nn.relu)
            self.Q_Out = tf.layers.dense(self.fc1, action_size, activation=None)
            self.predict = tf.argmax(self.Q_Out, 1)

            self.target_Q = tf.placeholder(shape=[None, action_size], dtype=tf.float32)
```

```
# DQNAgent 클래스 -> DQN 알고리즘을 위한 다양한 함수 정의
class DQNAgent():
    def __init__(self):

        # 클래스의 함수들을 위한 값 설정
        self.model = Model("Q")
        self.target_model = Model("target")

        self.memory = deque(maxlen=mem_maxlen)

        self.sess = tf.Session()
        self.init = tf.global_variables_initializer()
        self.sess.run(self.init)

        self.epsilon = epsilon_init

        self.Saver = tf.train.Saver()
        self.Summary, self.Merge = self.Make_Summary()

        self.update_target()

        if load_model == True:
            self.Saver.restore(self.sess, load_path)
```



## | 핵심소스코드(2)

### ✓ Main 함수

Model클래스에서 정의한 네트워크와 Agent 클래스에서 정의한 다양한 함수들을 이용해 행동을 결정하고 유니티 환경과 통신하며 학습을 수행한다.

```
# Main 함수
if __name__ == '__main__':
    # 유니티 환경 경로 설정 (file_name)
    env = UnityEnvironment(file_name=env_name)

    # 유니티 브레인 설정
    default_brain = env.brain_names[0]
    brain = env.brains[default_brain]

    # DQNAgent 클래스를 agent로 정의
    agent = DQNAgent()

    step = 0
    rewards = []
    losses = []

    # 환경 설정 (env_config)에 따라 유니티 환경 리셋 및 학습 모드 설정
    env_info = env.reset(train_mode=train_mode, config=env_config)[default_brain]

    # 게임 진행 반복문
    for episode in range(run_episode + test_episode):
        if episode > run_episode:
            train_mode = False
            env_info = env.reset(train_mode=train_mode)[default_brain]
        |
        # 상태, episode_rewards, done 초기화
        state = np.uint8(255 * np.array(env_info.visual_observations[0]))
        episode_rewards = 0
        done = False
```

```
# 한 에피소드를 진행하는 반복문
while not done:
    step += 1

    # 행동 결정 및 유니티 환경에 행동 적용
    action = agent.get_action(state)
    env_info = env.step(action)[default_brain]

    # 다음 상태, 보상, 게임 종료 정보 취득
    next_state = np.uint8(255 * np.array(env_info.visual_observations[0]))
    reward = env_info.rewards[0]
    episode_rewards += reward
    done = env_info.local_done[0]

    # 학습 모드인 경우 리플레이 메모리에 데이터 저장
    if train_mode:
        agent.append_sample(state, action, reward, next_state, done)
    else:
        time.sleep(0.01)
        agent.epsilon = 0.05

    # 상태 정보 업데이트 |
    state = next_state

    if episode > start_train_episode and train_mode:
        # 학습 수행
        loss = agent.train_model(done)
        losses.append(loss)
```

## | 핵심소스코드(3)

### ✓ Train


Train을 위한 config를 정의 후 anaconda 가상환경 내에서 mlagents-learn 명령어를 통해 실행함

INFO:mlagents.envs:Start training by pressing the Play button in Unirt Editor 명령어가 출력 시 Unity 내 환경을 실행함

trainer\_config - Windows 메모장  
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
default:
  trainer: ppo
  batch_size: 1024
  beta: 5.0e-3
  buffer_size: 10240
  epsilon: 0.2
  gamma: 0.99
  hidden_units: 128
  lambda: 0.95
  learning_rate: 3.0e-4
  max_steps: 5.0e4
  memory_size: 256
  normalize: false
  num_epoch: 3
  num_layers: 2
  time_horizon: 64
  sequence_length: 64
  summary_freq: 1000
  use_recurrent: false
  use_curiosity: false
  curiosity_strength: 0.01
  curiosity_enc_size: 128
```

```
Anaconda Prompt (anaconda3)
(base) C:\Users\lab419_os\Desktop>cd ml-agents-0.8.1
(base) C:\Users\lab419_os\Desktop\ml-agents-0.8.1>cd Unity_ML_Robot
(base) C:\Users\lab419_os\Desktop\ml-agents-0.8.1\Unity_ML_Robot>activate agent
(agent) C:\Users\lab419_os\Desktop\ml-agents-0.8.1\Unity_ML_Robot>mlagents-learn trainer_config.yaml --train
```



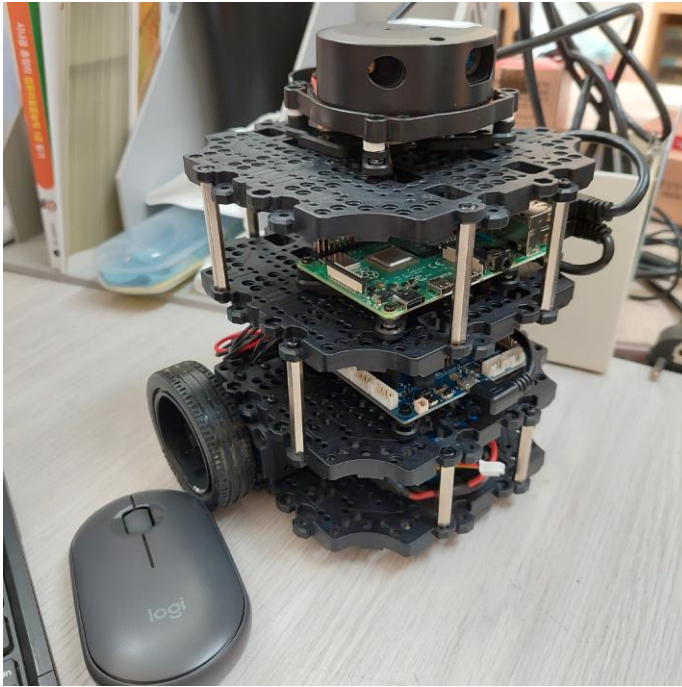
```
INFO:mlagents.trainers:{'--base-port': '5005',
  '--curriculum': 'None',
  '--debug': False,
  '--docker-target-name': 'None',
  '--env': 'None',
  '--help': False,
  '--keep-checkpoints': '5',
  '--lesson': '0',
  '--load': False,
  '--no-graphics': False,
  '--num-envs': '1',
  '--num-runs': '1',
  '--run-id': 'ppo',
  '--save-freq': '50000',
  '--seed': '1',
  '--slow': False,
  '--train': True,
  '<trainer-config-path>': 'trainer_config.yaml'}
INFO:mlagents.envs:Start training by pressing the Play button in the Unity Editor.
```

mlagents-learn trainer\_config.yaml --train

## | 참조-개발 환경 및 설명

구분		항목	적용내역
S/W 개발환경	Unity	Unity	가상 환경 구축을 위한 게임 엔진
		ML-Agent	Unity에서 제공하는 강화학습을 수행 하기 위한 플랫폼
	ROS	Ubuntu 18.04	로봇 구동을 위한 ROS 개발 환경
		Gazebo	ROS에서 제공하는 가상 시뮬레이션 툴로 강화 학습 예제 실습
H/W 구성장비	터틀봇3	라즈베리파이4	터틀봇3 구동을 위한 임베디드 보드
		LiDAR	주변의 사물을 인식하기 위한 센서로 로봇이 목적지 까지 장애물을 피해 이동할 수 있도록 해 줌

## | 참조-H/W 기능 실사 사진



터틀봇3(로봇)



실제 주행 맵

## | 참조-S/W 기능 실사 사진

```

[INFO] [1652874028.993023]: Setup subscriber on cmd_vel [go
[INFO] [1652874029.020375]: Setup subscriber on sound [turt
[INFO] [1652874029.046940]: Setup subscriber on motor_power
[INFO] [1652874029.074319]: Setup subscriber on reset [std
[INFO] [1652874031.239294]: Setup TF on Odometry [odom]
[INFO] [1652874031.256745]: Setup TF on IMU [imu_link]
[INFO] [1652874031.275650]: Setup TF on MagneticField [mag
[INFO] [1652874031.292996]: Setup TF on JointState [base_ll
[INFO] [1652874031.316629]: Connected to OpenCR board!
[INFO] [1652874031.337270]:
[INFO] [1652874031.373460]:
[INFO] [1652874031.380375]: Start Calibration of Gyro
[INFO] [1652874031.808284]: Calibration End

Control your TurtleBot3:
-----
moving around:
  a/s : increase/decrease linear velocity (Burger: ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
  d/a : increase/decrease angular velocity (Burger: ~ 2.84, Waffle and Waffle Pi : ~ 1.62)
  space key, q : force stop

CTRL-C to quit

currently: linear vel 0.0 angular vel 0.0
currently: linear vel 0.0 angular vel 0.0
currently: linear vel 0.01 angular vel 0.0
currently: linear vel -0.02 angular vel 0.0
currently: linear vel -0.03 angular vel 0.0
currently: linear vel 0.0 angular vel 0.0
  
```



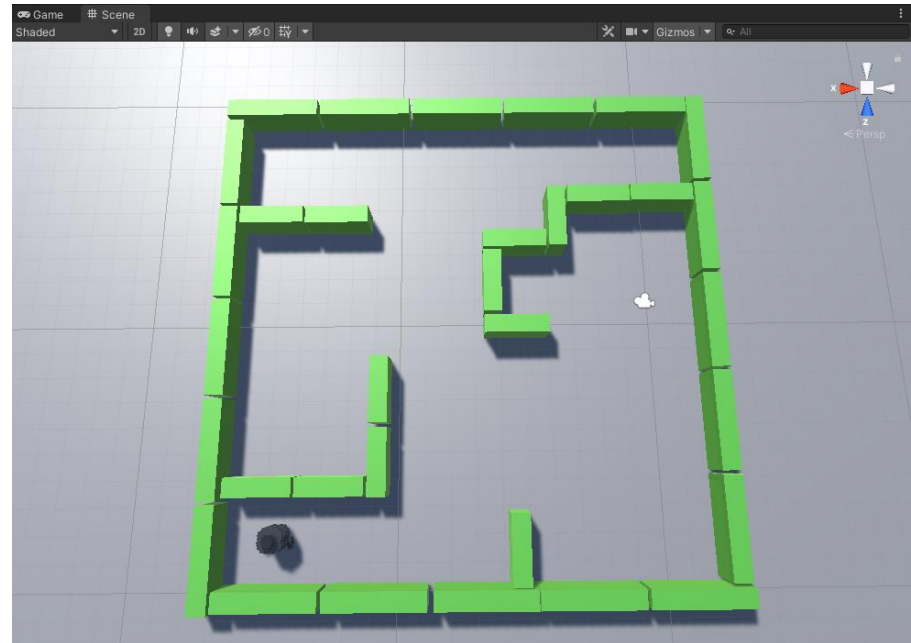
키보드 조작을 통한 터틀봇3 동작



## | 참조-S/W 기능 실사 사진

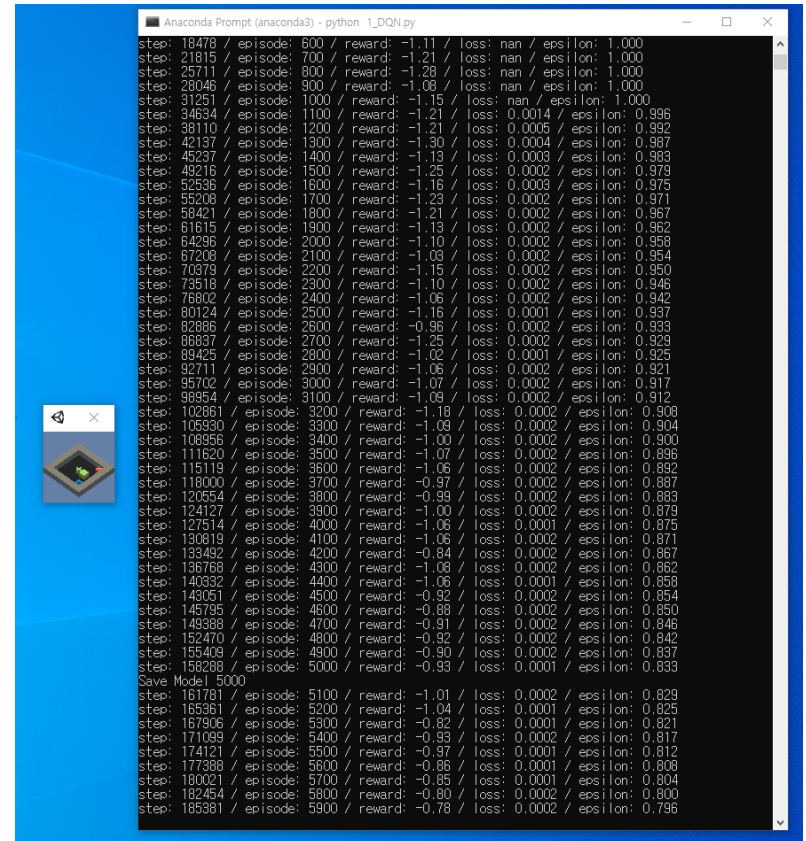
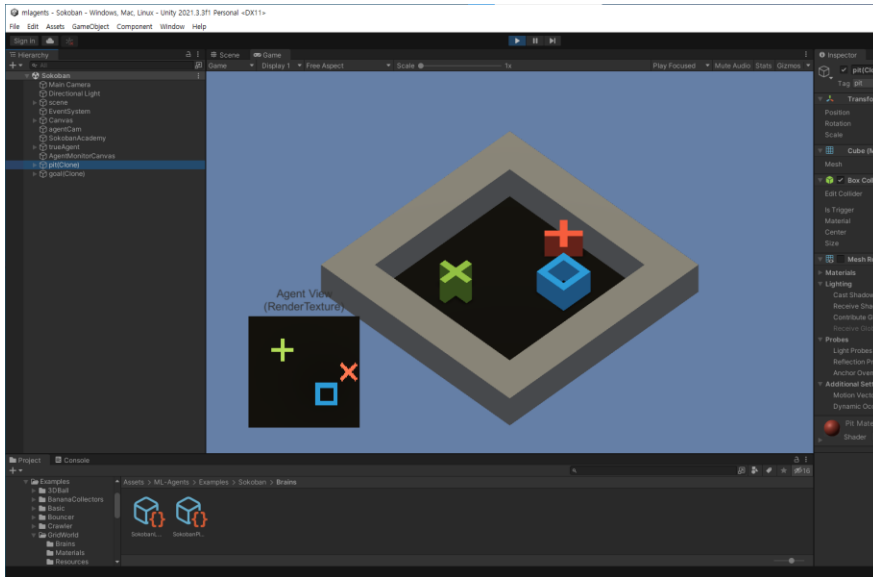


**Unity**



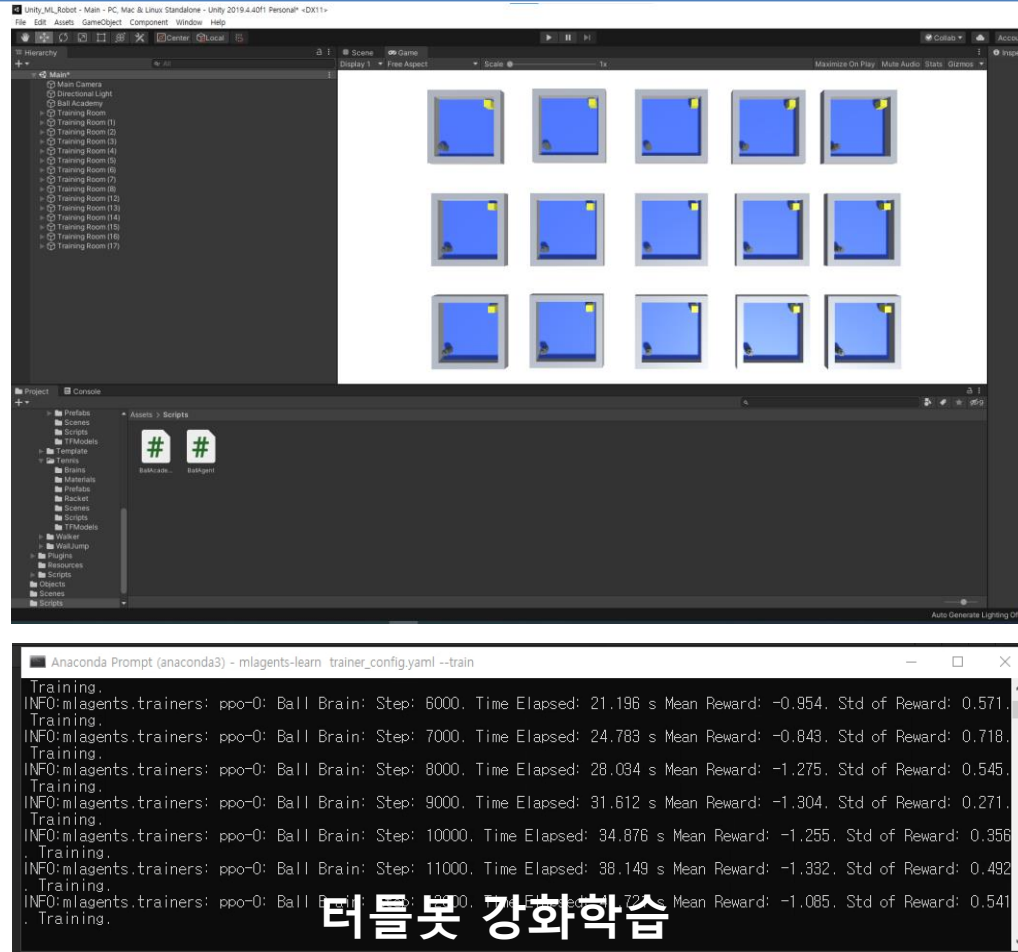
**Unity 가상 환경 구축**

## | 참조-S/W 기능 실사 사진



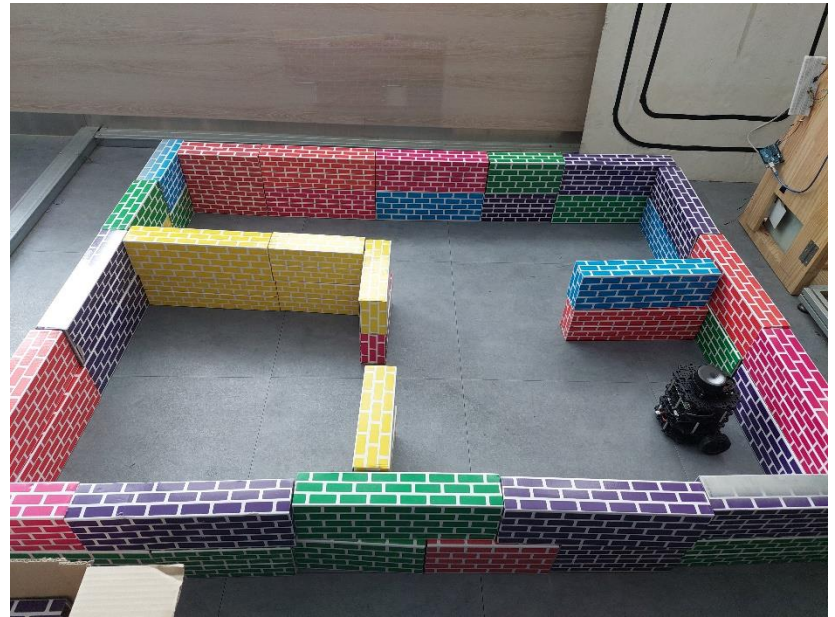
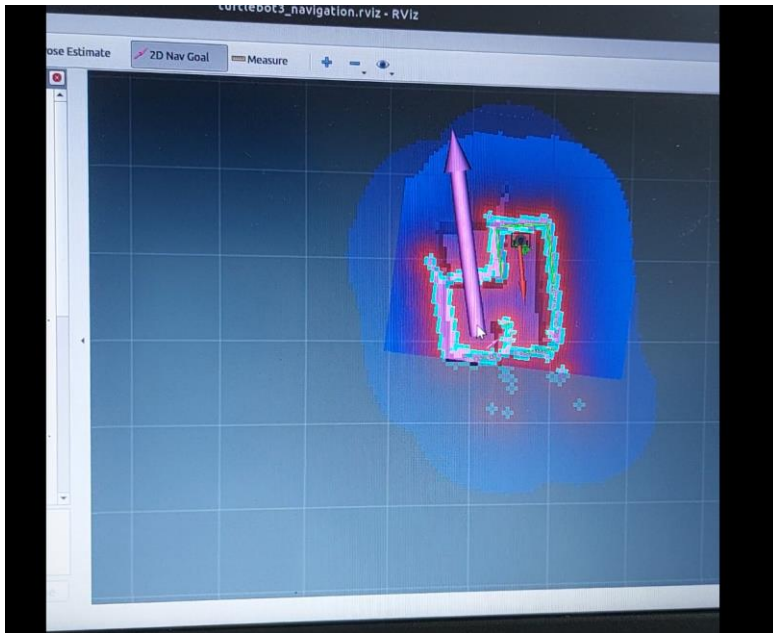
## ML-Agent 소코반 예제 실습

## | 참조-S/W 기능 실사 사진



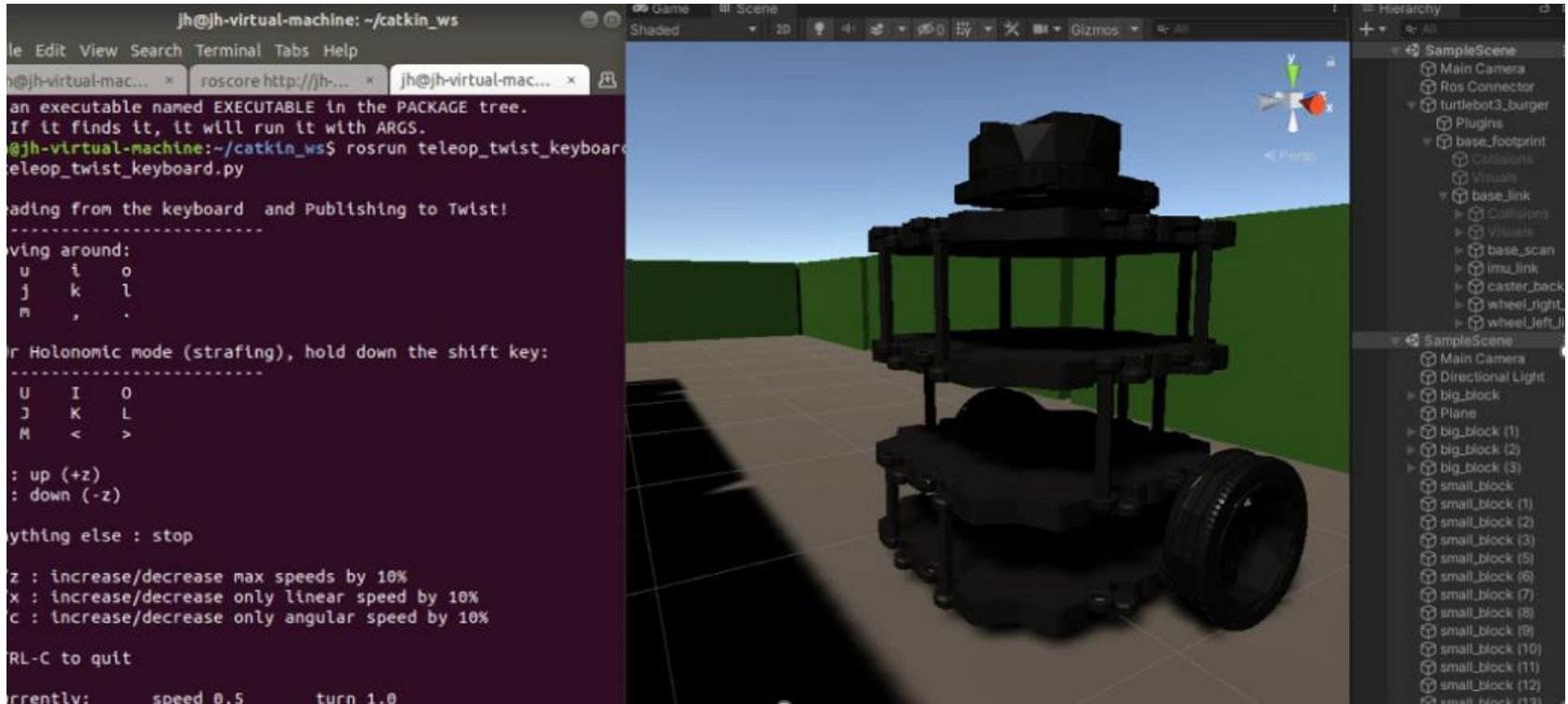


## | 참조-S/W 기능 실사 사진



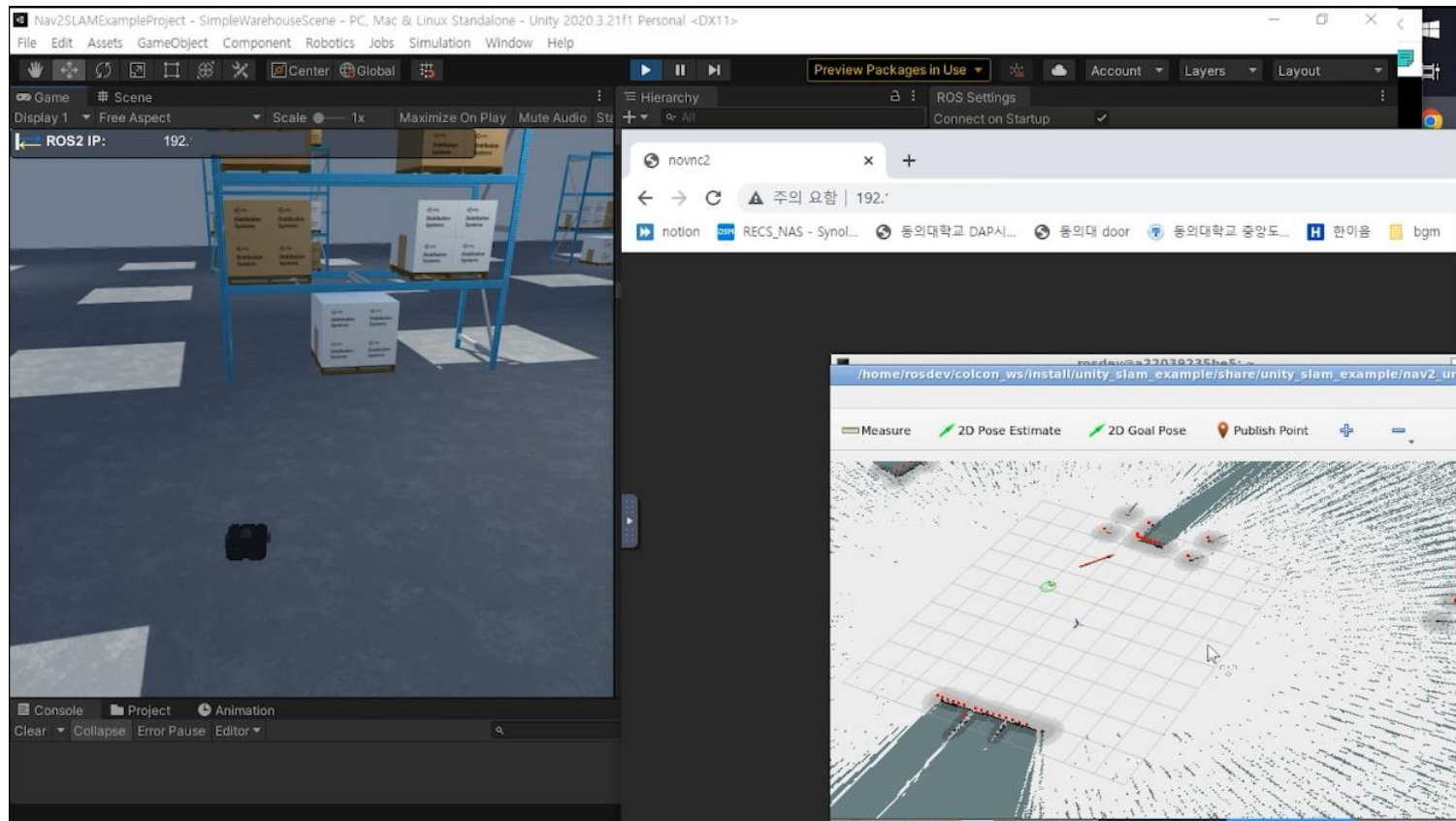
SLAM을 통한 자율주행

## | 참조-S/W 기능 실사 사진



Unity <-> ROS

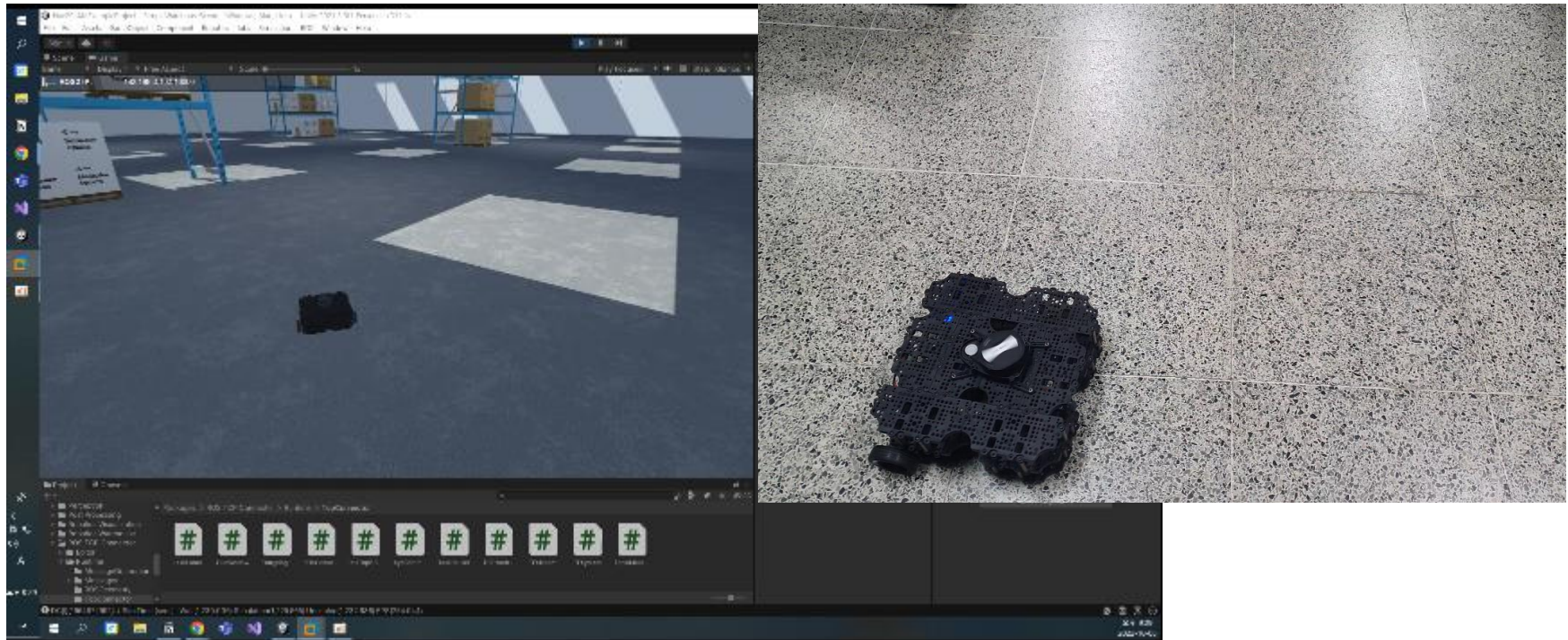
## | 참조-S/W 기능 실사 사진



Unity <-> ROS2



## | 참조-S/W 기능 실사 사진

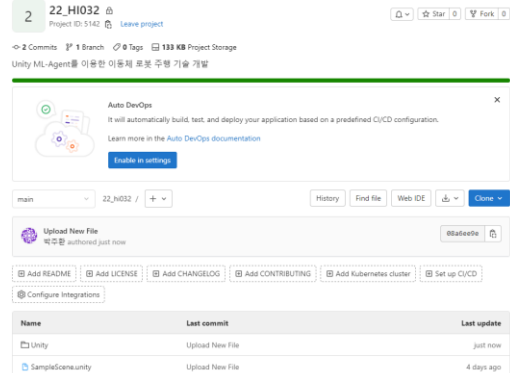
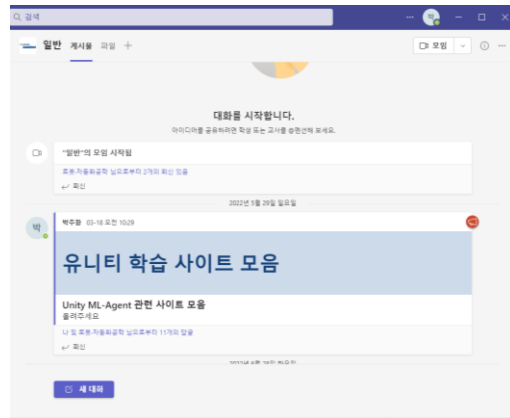


Sim-to-real

## | 참조-프로젝트 관리

### 관련 자료 관리

- Microsoft Teams, GitLab, kakaoTalk



### 형상 관리

- 팀블로그



구분	제목	작성자	작성일
수행 계획서	Unity ML-Agent를 이용한 이동체 로봇 주행 기술 개발	박주환	2022-03-21
온라인	회의록 (2022-05)(승인)	박주환	2022-05-29
오프라인	회의록 (2022-05-12)(신청)	박주환	2022-05-12
온라인	회의록 (2022-05)(신청)	박주환	2022-05-02
온라인	회의록 (2022-04)(승인)	박주환	2022-04-04

# Thank you

