

# 제4회 국민대학교 자율주행 경진대회

## 예선 과제 안내

---

---

제 4회 자율주행 경진대회

---

---

일정 : 2021년 월 일

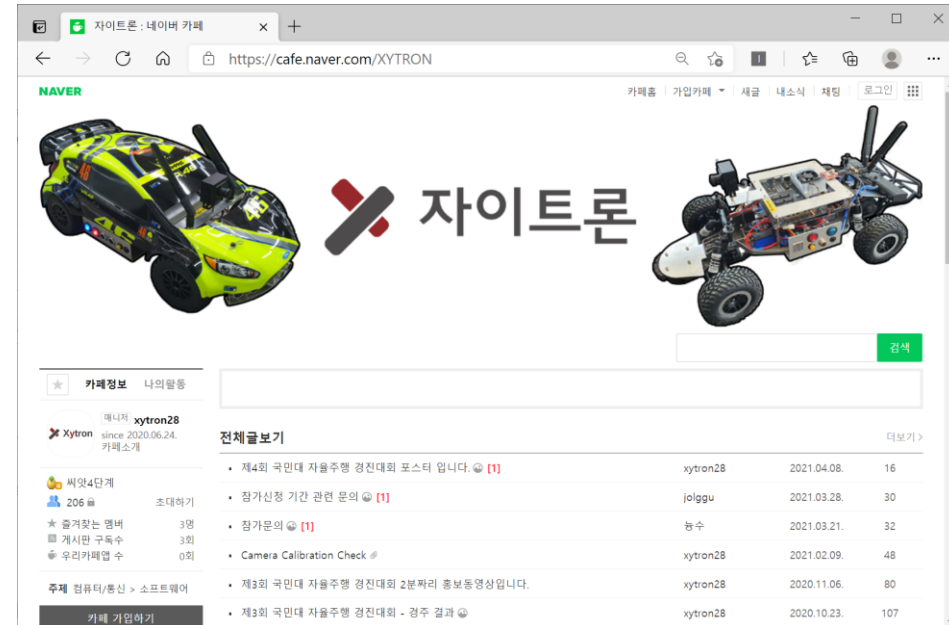
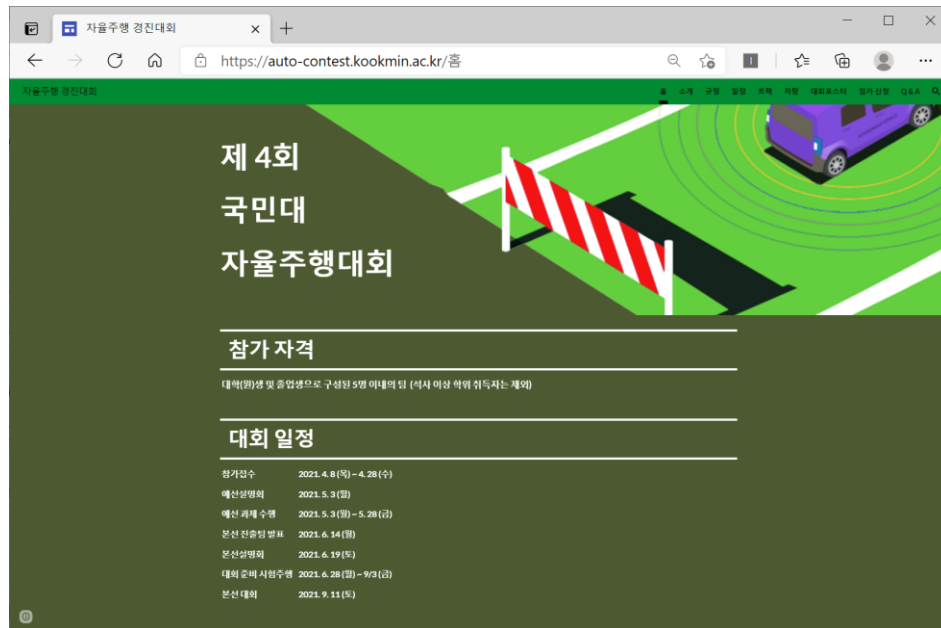
---

# 예선 안내

---

- 지원 자격
  - 서류 심사를 통과한 팀에게 예선전 관련 자료를 배포함(시뮬레이터, ROS 패키지, 설치 매뉴얼)
  - 예선 과제를 구현 제출하여 심사를 통과한 팀에게 본선 진출 자격을 부여함
  
- 과제
  - (1) 시뮬레이터 환경에서 초음파 센서 데이터를 이용한 주행 구현
  - (2) 도로 영상에서 차선을 검출하고 스티어링 휠을 제어
  - (3) 시뮬레이터 환경에서 AR태그 기반의 자율 주차 구현
  - (4) 논술 과제
  
- 평가
  - (1)번 과제 : 벽과 충돌하지 않고 ㄱ자 코스를 완주
  - (2)번 과제 : 차선을 검출 결과와 이에 따른 스티어링 휠 제어
  - (3)번 과제 : 주차 구역 선을 밟지 않고 AR태그 앞에 정면 주차
  - (4)번 과제 : 보고서의 내용을 검토하여 평가함

- 문의처
  - 국민대 자율 주행 대회 홈페이지 게시판
  - 자이트론 네이버 카페 - 제4회 국민대 대회 게시판 (<https://cafe.naver.com/XYTRON>)
  - 리눅스 와 ROS설치 방법 문의는 받지 않습니다



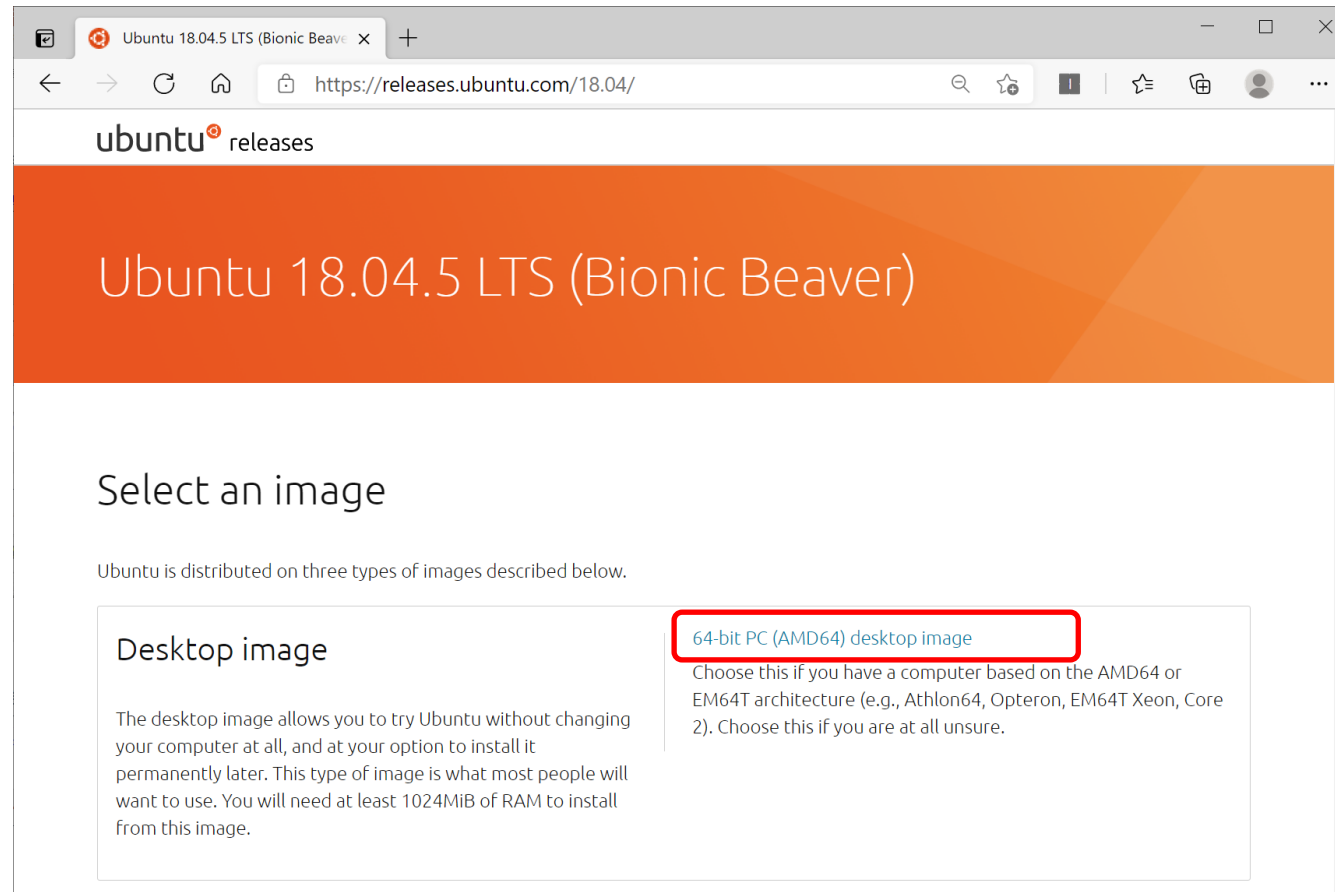
# 개발 환경 설정

---

UBUNTU 18.04  
ROS Melodic 설치

# Ubuntu 18.04 설치

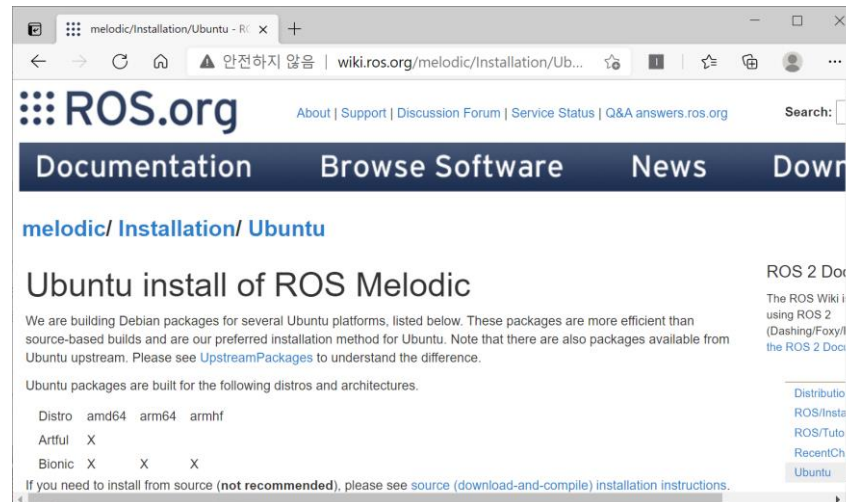
- 노트북이나 PC에 Ubuntu 18.04를 설치 합니다.
  - 64bit PC 버전을 다운받아서 설치함



# ROS Melodic 설치

- 설치 방법은 하기 사이트 참조

▶ <http://wiki.ros.org/melodic/Installation/Ubuntu>



- 설치 완료후 ROS 워크 스페이스 생성 바랍니다

▶ \$ cd      홈 디렉토리로 이동

▶ \$ mkdir -p ~/catkin\_ws/src    서브폴더 생성

▶ \$ cd catkin\_ws    catkin\_ws 폴더 아래로 이동

▶ \$ catkin\_make      ROS 코딩환경 셋업과 정리(빌드)

# ROS 작업환경 설정

- ROS 작업에 필요한 환경변수 설정
  - 홈 디렉토리에 있는 .bashrc 파일을 수정
    - ▶ \$ cd (홈디렉토리로 이동)
    - ▶ \$ sudo gedit ~/.bashrc (아래 내용을 마지막에 추가)
    - ▶ \$ source .bashrc (수정한 내용을 시스템에 반영)

## .bashrc 파일의 내용

```
...  
alias cm='cd ~/catkin_ws && catkin_make'  
source /opt/ros/melodic/setup.bash  
source ~/catkin_ws/devel/setup.bash  
export ROS_MASTER_URI=http://localhost:11311  
export ROS_HOSTNAME=localhost
```

# 시뮬레이터 의존 패키지 설치

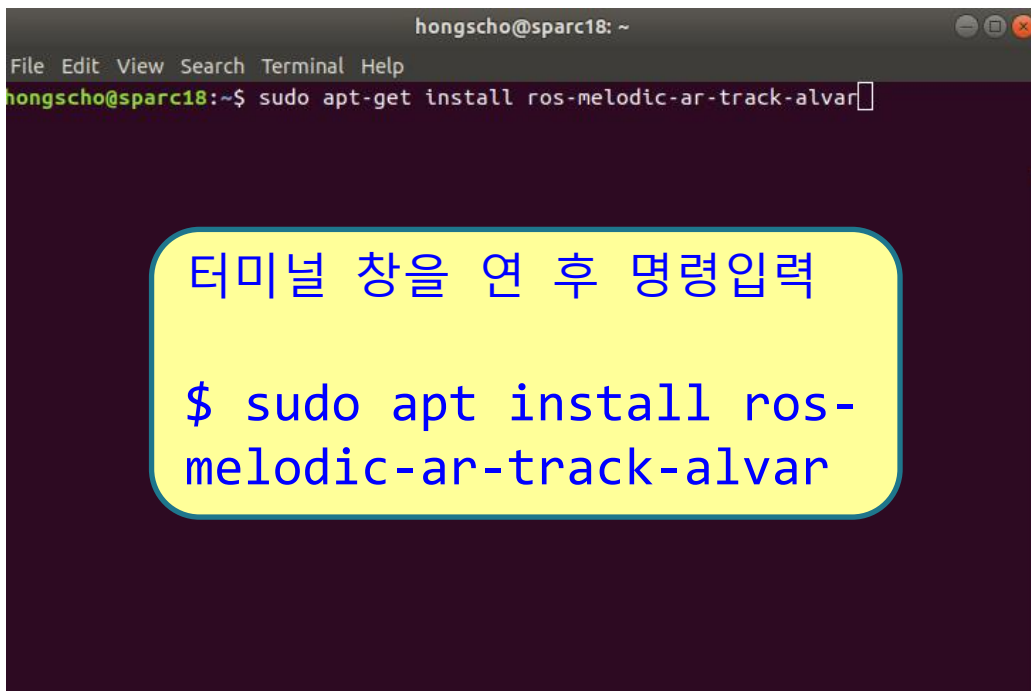
---

- 과제수행에 필요한 2D 시뮬레이터를 사용하려면 먼저 아래 의존 패키지들의 설치가 필요
- 설치할 의존 패키지들 (여기 순서대로 설치한다)
  - ar-track-alvar v0.7.1 - Alvar AR Tag 전용 ROS 패키지
  - pygame v1.9.6 - 파이썬 게임 제작 라이브러리
  - pillow v6.2.2 - 파이썬 이미지 처리 라이브러리



# ar-track-alvar 설치

- ar-track-alvar for ROS Melodic
  - \$ sudo apt update
  - \$ sudo apt-get install ros-melodic-ar-track-alvar



A terminal window titled 'hongscho@sparc18: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command 'hongscho@sparc18:~\$ sudo apt-get install ros-melodic-ar-track-alvar' is entered. A yellow callout box with a blue border contains the text: '터미널 창을 연 후 명령입력' and '\$ sudo apt install ros-melodic-ar-track-alvar'.



ROS.org  
ar\_track\_alvar

이 패키지는 오픈 소스 AR 태그 추적 라이브러리인 [Alvar](#)을 ROS 래퍼입니다.  
ar\_track\_alvar에는 4 가지 주요 기능이 있습니다.

1. 다양한 크기, 해상도 및 데이터 / ID 인코딩의 AR 태그 생성
2. 개별 AR 태그의 포즈를 식별하고 추적하여 선택적으로 더 나은 포즈 추정을 위해 키넥트 깊이 데이터를 통합합니다 (키넥트가 사용 가능한 경우).
3. 여러 태그로 구성된 "번들"의 자세를 식별하고 추적합니다. 이를 통해보다 안정적인 포즈 추정, 폐쇄 견고성 및 다면체 추적이 가능합니다.
4. 카메라 이미지를 사용하여 번들의 태그 간 공간 관계를 자동으로 계산하므로 사용자는 번들 기능을 사용하기 위해 XML 파일에서 태그 위치를 수동으로 측정하고 입력 할 필요가 없습니다 (\*\* 현재 작동하지 않음-아래 참조).

Alvar는 다른 ROS AR 태그 패키지의 기반이 된 ARToolkit보다 훨씬 새롭고 고급입니다. Alvar는 다양한 조명 조건을 처리 할 수있는 적응형 임계 값보다 안정적인 포즈 추정을위한 광학 흐름 기반 추적 및 태그 수가 증가해도 크게 느려지지 않는 향상된 태그 식별 방법을 제공합니다.

# Python pip 설치

- pygame과 pillow를 설치하려면 pip(python package index)가 필요
- pip는 파이썬으로 작성된 패키지 소프트웨어를 설치 · 관리하는 패키지 관리 시스템
- pip는 ubuntu 18.04의 기본 패키지가 아니므로 직접 설치한다.

(첫번째 방법)

```
$ sudo apt update
```

```
$ sudo apt-get install python-pip
```

(두번째 방법)

```
$ cd ~
```

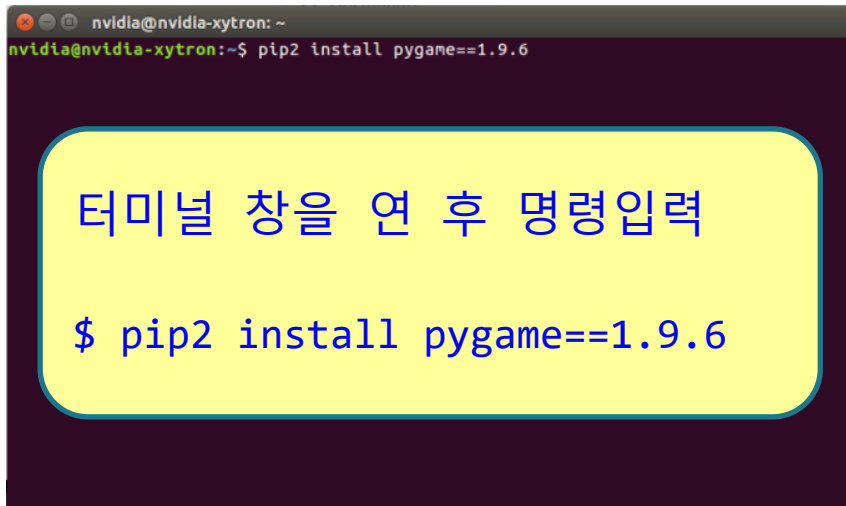
```
$ sudo apt install curl
```

```
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

```
$ python ~/get-pip.py
```

# Pygame 설치

- pygame 1.9.6
  - \$ pip2 install pygame==1.9.6



# Pillow 설치

- pillow 6.2.2
  - \$ pip2 install pillow==6.2.2



```
nvidia@nvidia-xytron: ~  
nvidia@nvidia-xytron:~$ pip2 install pillow==6.2.2
```

터미널 창을 연 후 명령입력

```
$ pip2 install pillow==6.2.2
```

## Python Imaging Library

소프트웨어



Python Imaging Library은 파이썬 인터프리터에 다양한 이미지 파일 형식을 지원하고 강력한 이미지 처리와 그래픽 기능을 제공하는 자유-오픈 소스 소프트웨어 라이브러리이다. 줄여서 PIL이라고 부른다. 윈도우와 맥 오에스 엑스, 리눅스를 지원한다. [위키백과](#)

작성 언어: [파이썬](#), [C](#)

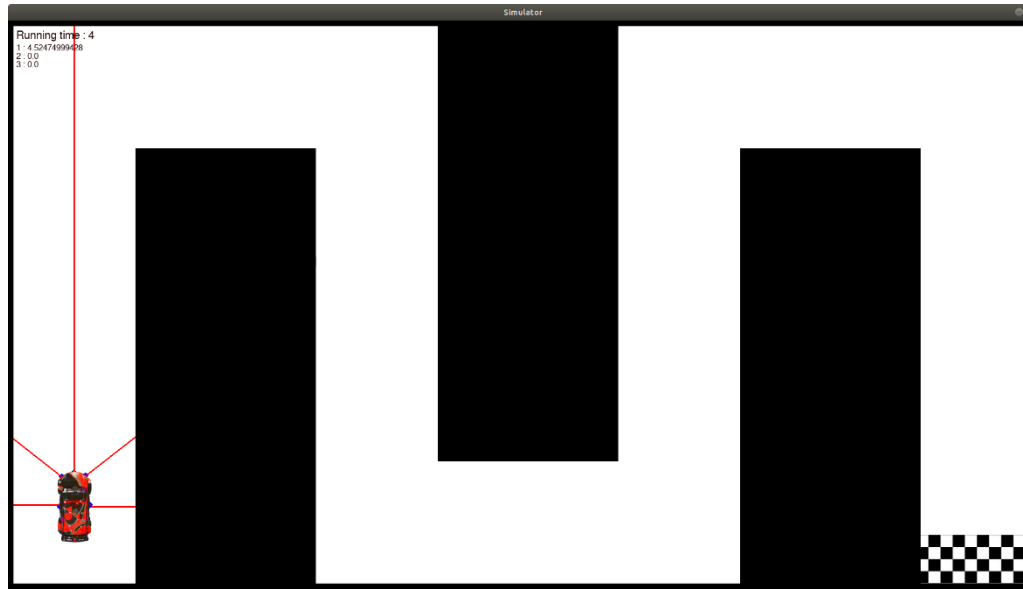
# 과제 #1


---

시뮬레이터에서 초음파 거리 센서를  
이용한 주행 구현

## 2D 시뮬레이터

- 시뮬레이터상에서 xycar는 5개의 초음파 거리 센서 데이터를 수신하여 벽을 인지하고 회피 주행하는 것을 목표로 함



- 모터제어 토픽을 보내 시뮬레이터 내부의 차를 조종 할 수 있다.
- 시뮬레이터 내부의 xycar가 벽에 부딪히지 않고 「」 자 코스를 운행하도록 프로그래밍한다.
- 시뮬레이터 내부의 xycar가 벽에 닿았을 때 실패 화면과 함께 재시작 된다. 3번 실패하면 처음부터 다시 시작된다.

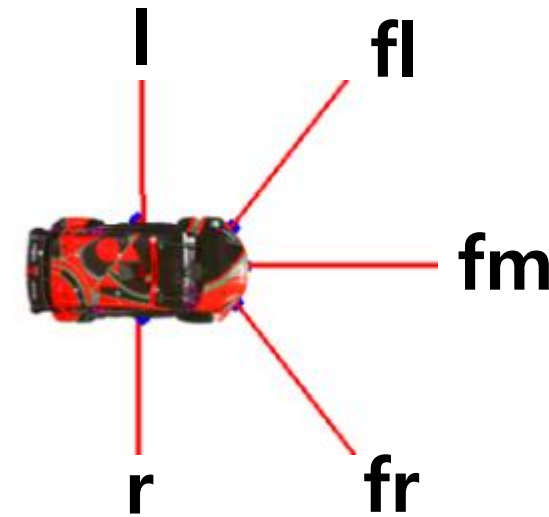
# 과제 목표 - 초음파 데이터 처리하기

- 시뮬레이터가 측정하는 초음파 센서의 값이 “ultrasonic” 토픽으로 전송

```

hongscho@sparc18: ~/xycar_ws/src/xycar_sim_drive/launch
File Edit View Search Terminal Help
fl: 110, fm: 769, fr: 114, bl: 0, bm: 0, br: 0, r: 137, l: 138
(110, 769, 114, 0, 0, 0, 137, 138)
(110, 768, 114, 0, 0, 0, 137, 138)
(110, 768, 114, 0, 0, 0, 137, 138)
(110, 768, 114, 0, 0, 0, 137, 138)
(110, 768, 114, 0, 0, 0, 137, 138)
(110, 768, 114, 0, 0, 0, 137, 138)
(110, 768, 114, 0, 0, 0, 137, 138)
(110, 767, 114, 0, 0, 0, 137, 138)
(110, 767, 114, 0, 0, 0, 137, 138)
(110, 767, 114, 0, 0, 0, 137, 138)
(110, 767, 114, 0, 0, 0, 137, 138)
(110, 767, 114, 0, 0, 0, 137, 138)
fl: 110, fm: 766, fr: 114, bl: 0, bm: 0, br: 0, r: 137, l: 138
(110, 766, 114, 0, 0, 0, 137, 138)
(110, 766, 114, 0, 0, 0, 137, 138)
(110, 766, 114, 0, 0, 0, 137, 138)
(110, 766, 114, 0, 0, 0, 137, 138)

```

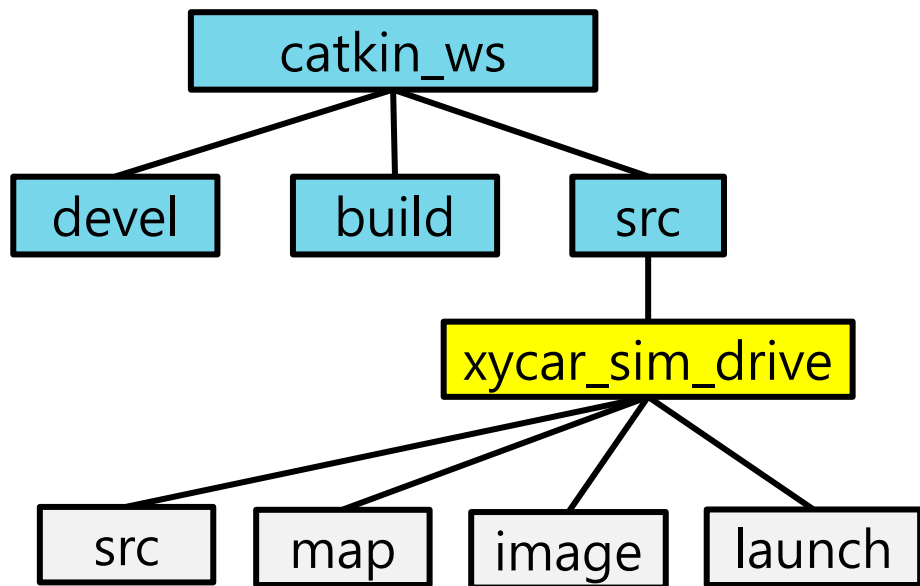


- 메시지는 실제 Xycar B2의 초음파 메시지 형식과 같은 구조로 저장되어 있다.
  - Int32MultiArray :
    - ▶ Data 저장 순서 : [정면좌측, 정면중앙, 정면우측, 후면좌측, 후면중앙, 후면우측, 우측, 좌측]  
(fl, fm, fr, 0, 0, 0, r, l)
    - ▶ 해당 시뮬레이터에는 후면 센서가 존재하지 않으므로 0 값이 들어간다.

# 제공된 파일

- xycar\_sim\_drive.tgz 파일을  
~/catkin\_ws/src/에 압축을 해제하고 cm 명령어를 사용하여 빌드한다

\$ cm



- 빌드가 완료되면 하기 명령으로 실행함
  - \$ roslaunch xycar\_sim\_drive xycar\_sim\_drive.launch

```

nvidia@tegra-ubuntu: ~/xycar
cmnvidia@tegra-ubuntu:~/xycar/src$ cd ~/xycar/ && catkin_make
Base path: /home/nvidia/xycar
Source space: /home/nvidia/xycar/src
Build space: /home/nvidia/xycar/build
Devel space: /home/nvidia/xycar/devel
Install space: /home/nvidia/xycar/install
####
#### Running command: "cmake /home/nvidia/xycar/src -DCATKIN_DEVEL_PREFIX=/home/
nvidia/xycar/devel -DCMAKE_INSTALL_PREFIX=/home/nvidia/xycar/install -G Unix Mak
efiles" in "/home/nvidia/xycar/build"
####
-- Using CATKIN_DEVEL_PREFIX: /home/nvidia/xycar/devel
-- Using CMAKE_PREFIX_PATH: /home/nvidia/xycar/devel;/opt/ros/kinetic
-- This workspace overlays: /home/nvidia/xycar/devel;/opt/ros/kinetic
-- Found PythonInterp: /usr/bin/python2 (found suitable version "2.7.12", minimu
m required is "2")
-- Using PYTHON_EXECUTABLE: /usr/bin/python2

```

```

-- Found Threads: TRUE
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.20
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-----
-- traversing 1 packages in topological order:
--   - xycar_sim_drive
--
-- +++ processing catkin package: 'xycar_sim_drive'
-- ==> add_subdirectory(xycar_sim_drive)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/nvidia/xytron_ws/build
####
#### Running command: "make -j4 -l4" in "/home/nvidia/xytron_ws/build"
####
nvidia@nvidia-xytron:~/xytron_ws$

```



# 구현 예시

- 제공된 기본 예제 코드 `ultra_driver.py`는 출발점에서 직진해서 벽과 충돌함
  - 예제 코드의 주행 부분을 직접 수정하여 벽을 피하고 길을 따라 이동하는 코드 구현

```
1 #!/usr/bin/python
2
3 import rospy, math
4 from std_msgs.msg import Int32MultiArray
5
6 def callback(msg):
7     print(msg.data)
8
9 rospy.init_node('guide')
10 motor_pub = rospy.Publisher('xycar_motor_msg', Int32MultiArray, queue_size=1)
11 ultra_sub = rospy.Subscriber('ultrasonic', Int32MultiArray, callback)
12
13 xycar_msg = Int32MultiArray()
14
15 while not rospy.is_shutdown():
16     angle = 0
17     xycar_msg.data = [angle, 10]
18     motor_pub.publish(xycar_msg)
```

주행코드 (조향각 0도, 속도 10 )

## 예상 결과

- 시뮬레이터의 차량이 벽과 충돌하지 않고 종료 구간까지 갔다가 다시 돌아온다.
- 시뮬레이터는 총 3회 주행성공하면 자동 종료가 됨. 그림1 -> 그림2 -> 그림 1 -> 종료 순서임.

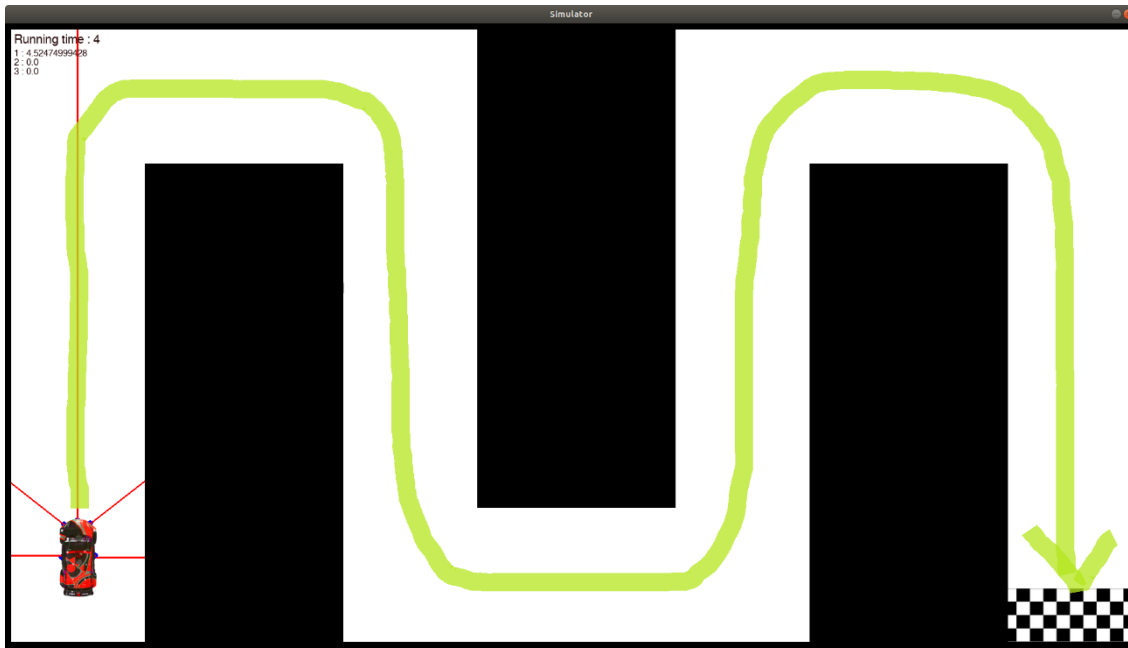


그림 1

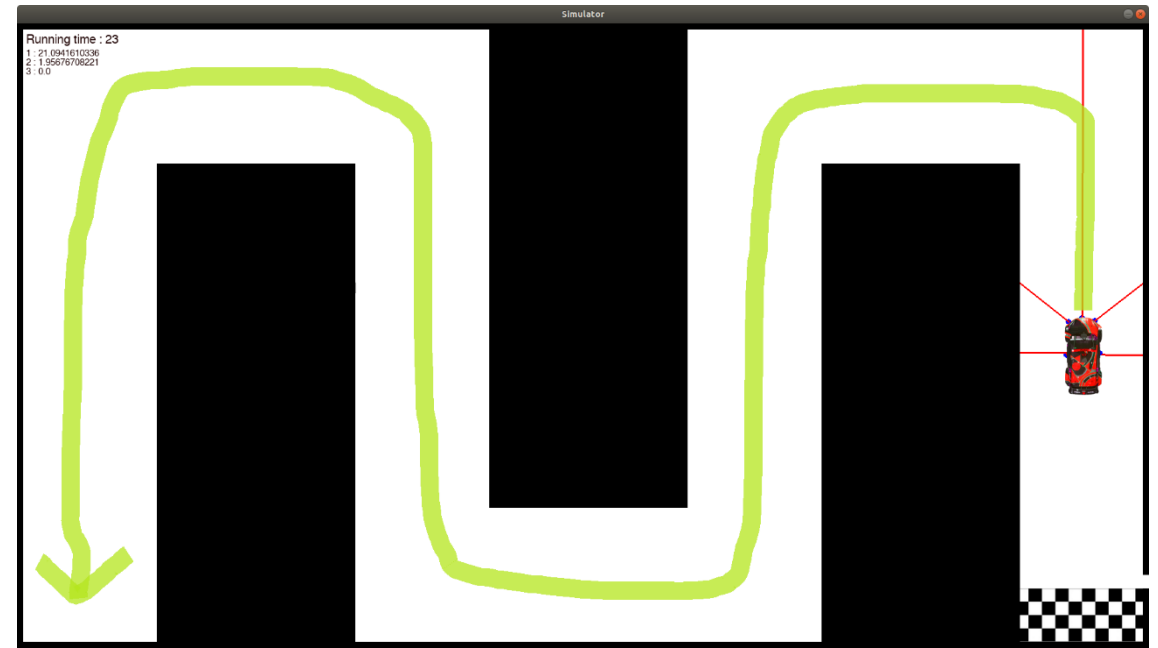
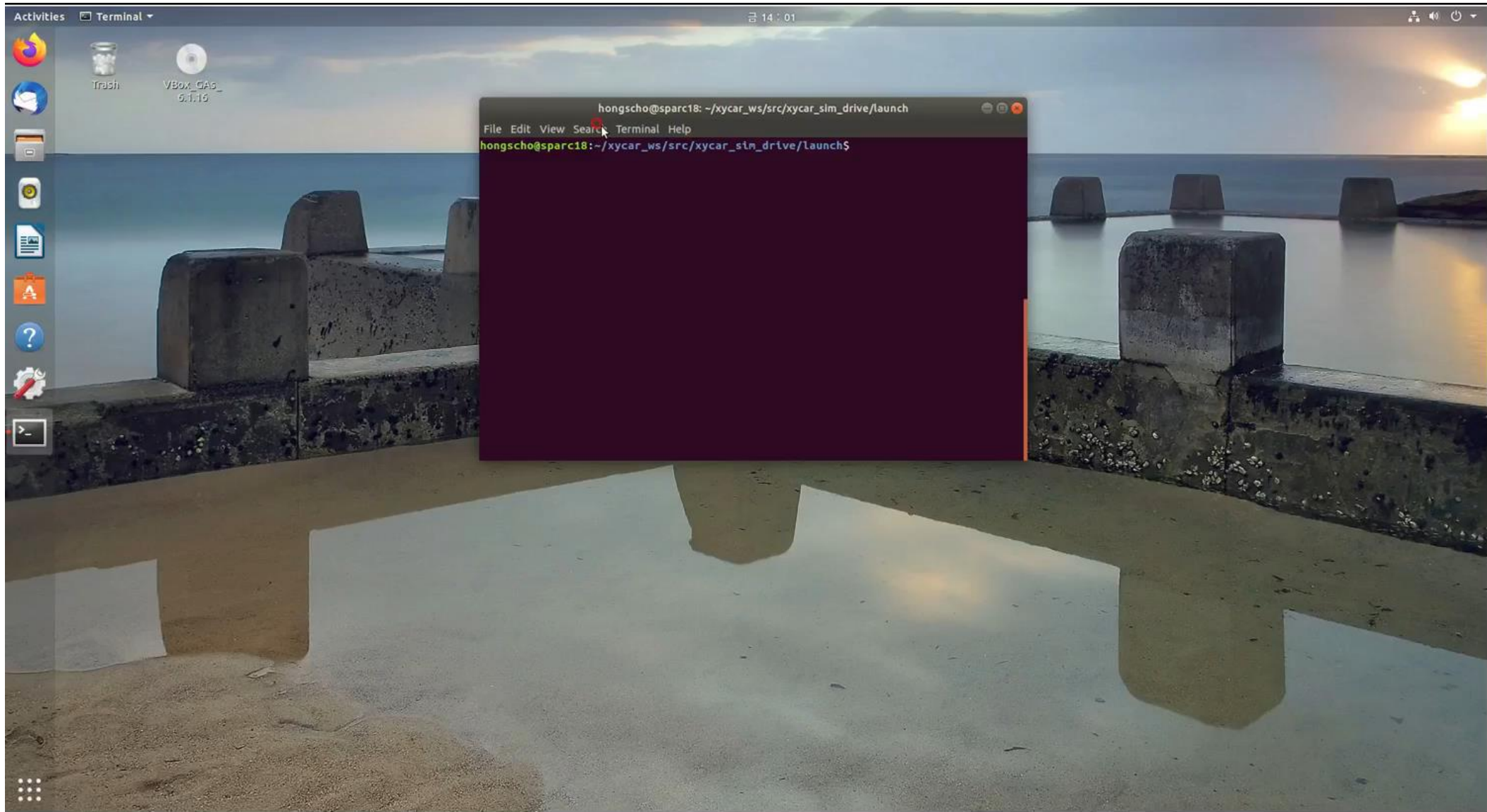


그림 2



# 결과물 제출

---

- 동영상 제출
  - 시뮬레이터에서 자동차가 움직이는 것을 휴대폰이나 동영상 캡처툴로 촬영해서 제출
- 파일 제출
  - ultra\_driver.py 파일 제출
- 문서 제출 (아래아 한글)
  - ultra\_driver.py 파일 소스코드를  
상세히 설명하는 문서를 작성하여 제출

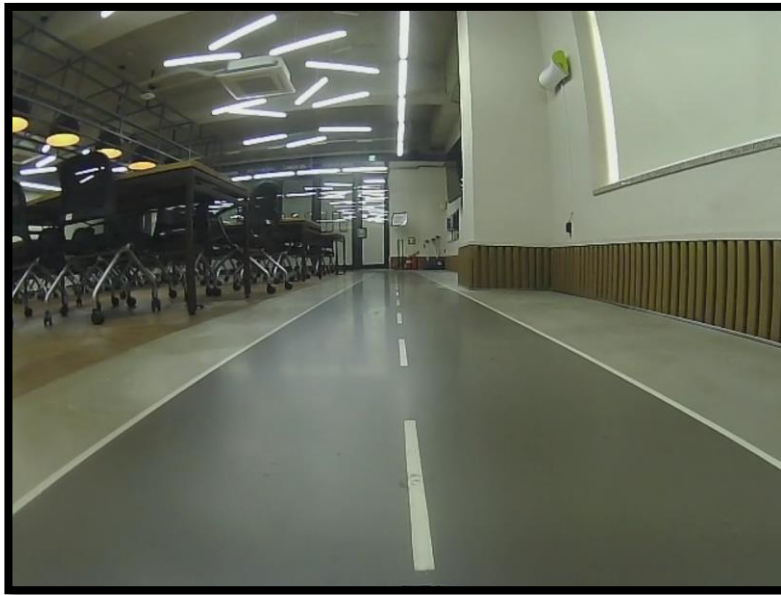
# 과제 #2

---

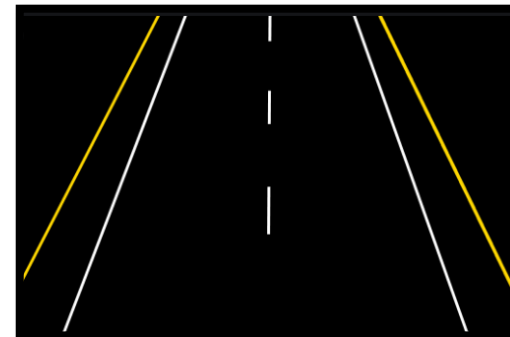
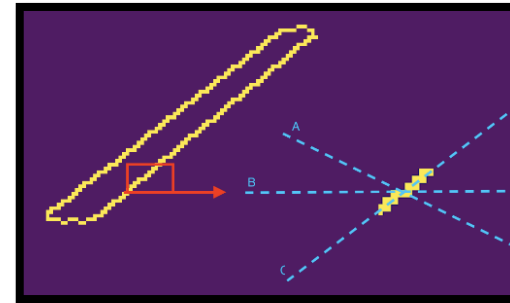
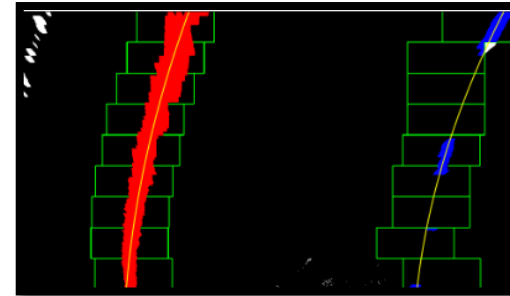
카메라 영상 데이터를 처리하여  
차선 인식 주행 구현

# 과제 목표 - 영상 처리

- 차선 인식
  - 제공된 트랙 주행 동영상에서 차선 인식하는 알고리즘을 OpenCV등의 영상 처리 라이브러리를 활용하여 구현



트랙 주행 영상 kmu\_track.mkv



- 알고리즘은 선택 (명도차, 허프, 슬라이딩 윈도우 등)

# 패키지 생성

- line\_drive 패키지 만들기 - ROS workspace의 src 폴더에서

- \$ catkin\_create\_pkg line\_drive std\_msgs rospy

- src 폴더 아래에

- 파이썬 파일 만들기

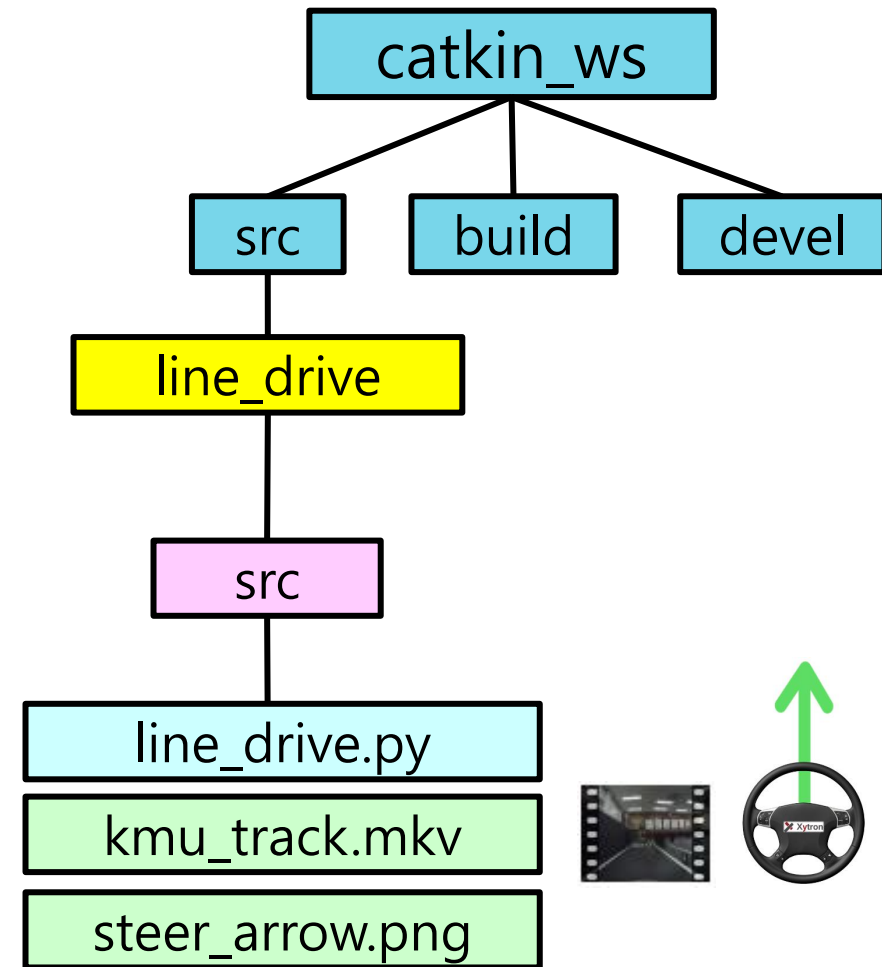
- ▶ line\_drive.py

- ▶ .mkv 동영상에서 차선을 찾는 코드

- 트랙 동영상 파일과 핸들 그림 파일 넣기

- ▶ kmu\_track.mkv : 트랙을 촬영한 동영상 파일

- ▶ steer\_arrow.png : 핸들 그림

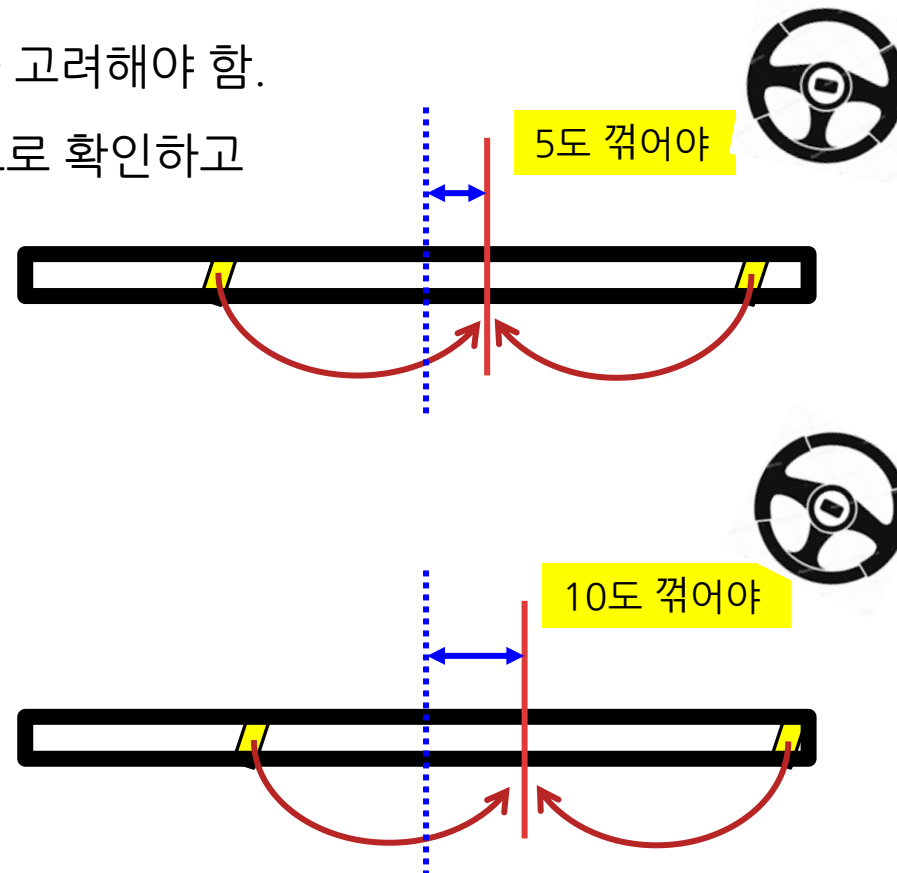
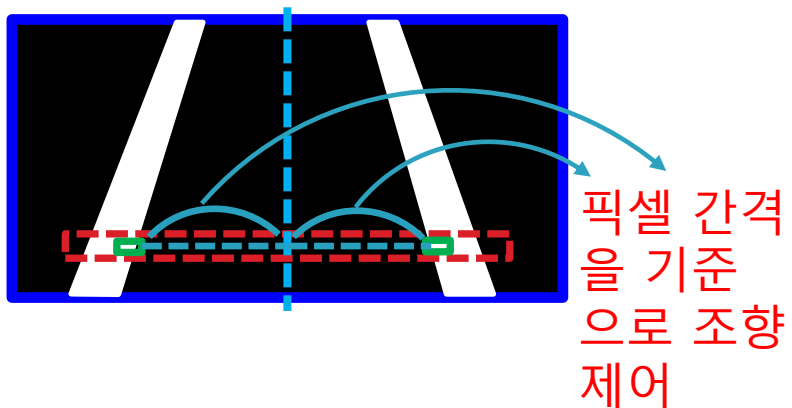


# 조향각 제어

- 차량 조향각 제어
  - 영상처리를 통해 인식된 차선의 위치를 카메라 중앙점을 기준으로 이동된 간격을 계산하여 조향각을 제어함.

차량의 속도와 조향각 결정까지의 처리 지연시간을 고려해야 함.

이동된 픽셀 간격과 조향각의 회전 비율은 시각적으로 확인하고 적절한 값을 찾는다

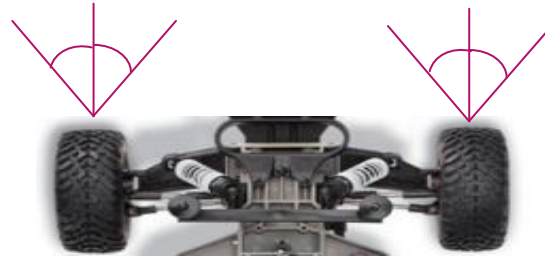




# 스티어링 휠 회전

- 조향각의 좌/우 범위를 확인하고 steer\_angle 을 제어한다

Angle : left MAX(-50) <- 초기 중앙(0) -> right MAX (50)



조향 범위는  
좌/우 20도까지

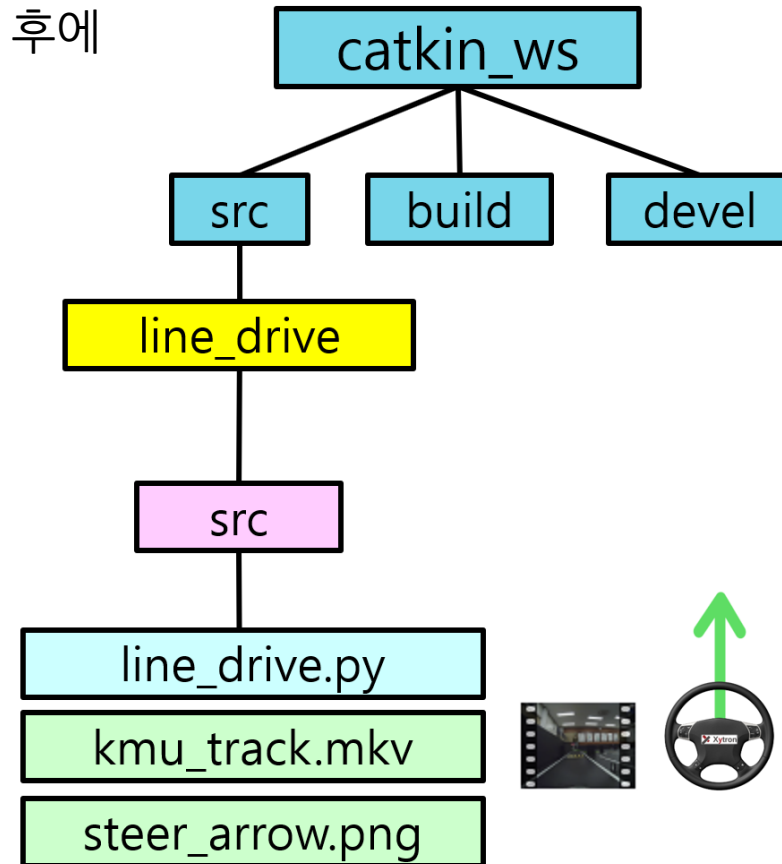
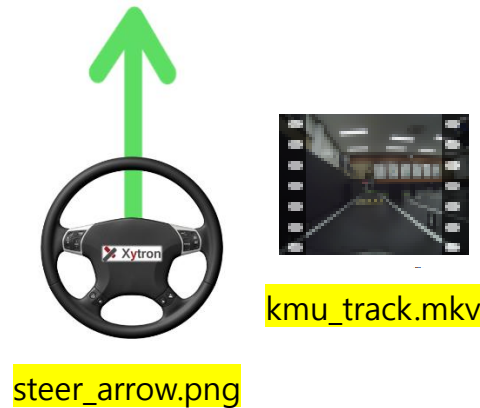


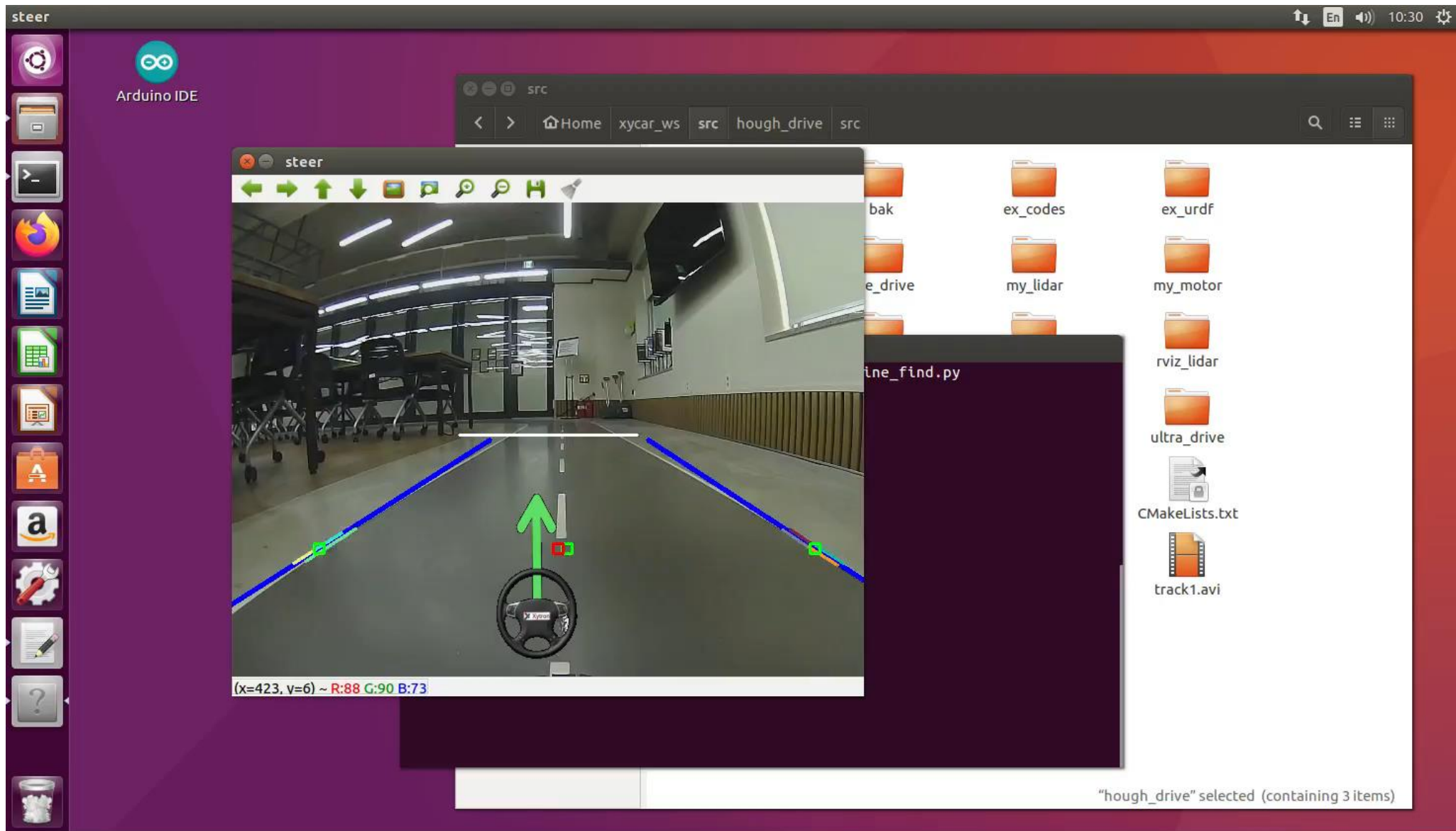
[예제] 왼쪽과 오른쪽 차선에 녹색 사각형 +  
그 둘의 중간 위치에 녹색 사각형  
+ 화면의 중앙빨간색 사각형  
+ 아래 중앙에 핸들 모양과 화살표

[중요] 스티어링 휠을 제외한 빨간색, 녹색 사각형 또는 직선 등을 나타내는 UI의 형태를 (더 멋있게) 변경해도 무방함. 양쪽 차선을 나타내는 녹색의 사각형은 반드시 넣어주기 바람.

# Line\_drive.py 실행

- ~/catkin\_ws/src/line\_drive/src 폴더 아래에
  - kmu\_track.mkv 파일과 steer\_arrow.png 파일을 준비한 후에
- 다음 명령으로 실행
  - \$ chmod +x line\_drive.py
  - \$ python line\_drive.py





# 결과물 제출

---

- 동영상 제출
  - 실행 결과를 휴대폰이나 동영상 캡처툴로 촬영해서 제출
- 파일 제출
  - line\_drive.py를 제출
- 문서 제출 (아래아 한글)
  - 소스코드를 상세히 설명하는 문서를 작성하여 제출

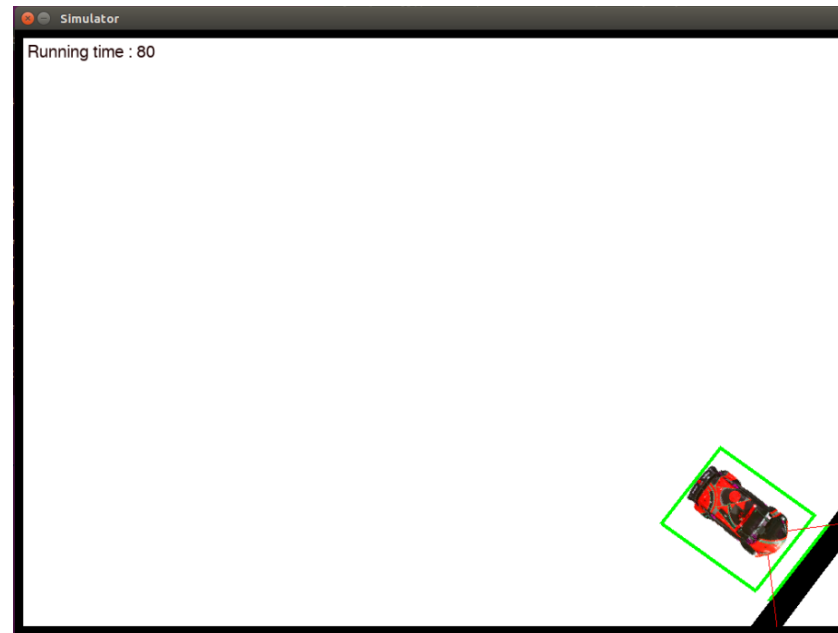
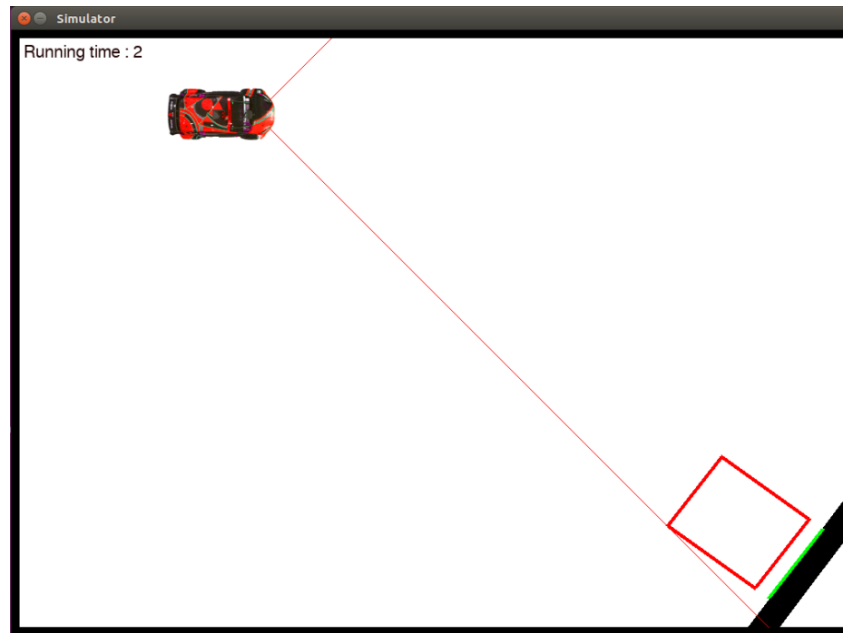
# 과제 #3

---

AR 태그 기반의 자율주차 프로그램 작성

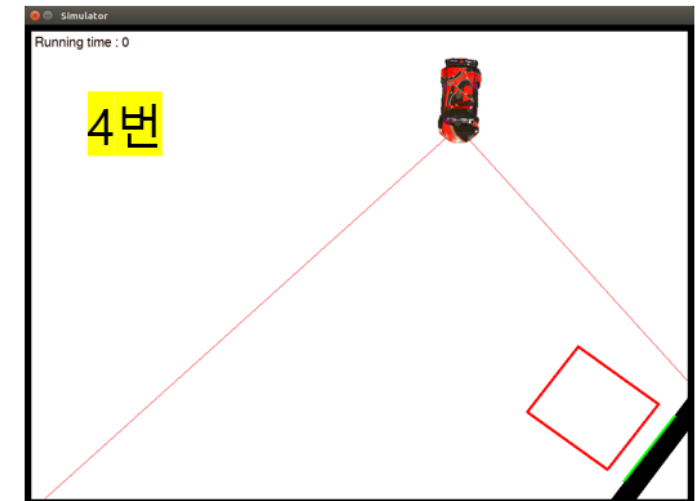
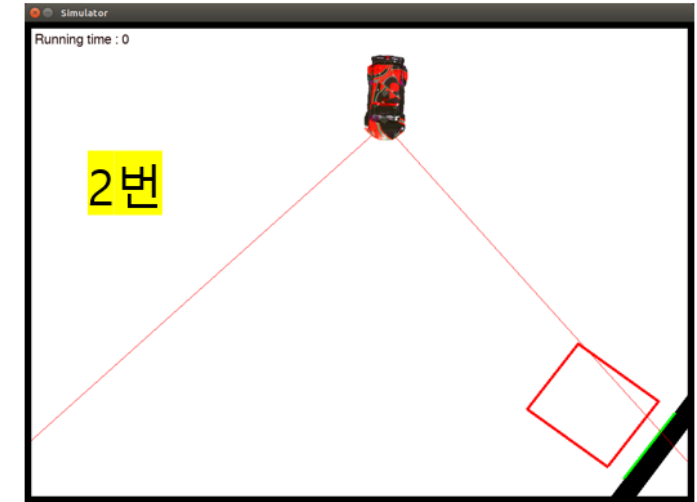
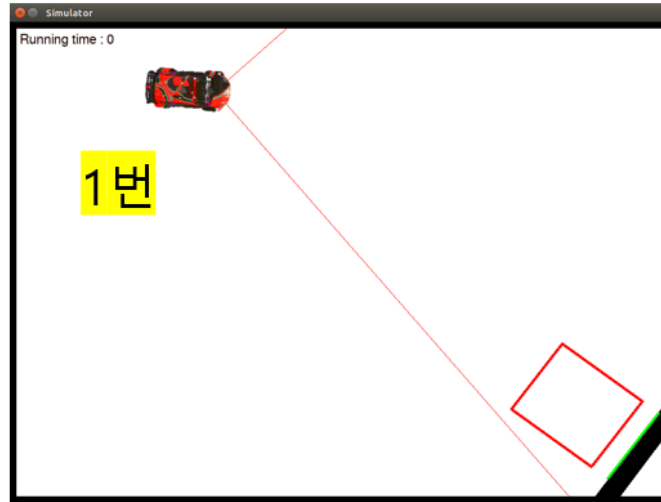
# 과제 설명

- 2D 시뮬레이터가 제공되고 자동차는 가상의 AR태그로부터 정보를 획득하여 차량을 움직여서 주차구역(사각형) 안에 정확하게 주차하는 과제임.
- 시뮬레이터는 주차공간 앞에 있는 녹색의 가상 AR태그가 차량의 카메라에 어떤 각도로 잡히는지 계산하여 자동차에게 AR태그의 위치와 자세 정보를 보내주며, 이 데이터를 처리하여 주차구역에 자동차가 정확하게 주차하도록 프로그래밍
- 차가 주차구역 안에 완전히 들어갔을 때 주차구역 선이 녹색으로 바뀐.



# 주차 출발 위치

- 주차 출발 위치는 총 4가지가 있으며, 시뮬레이터 창을 활성화한 상태에서 키보드 1~4를 누르면 출발 위치가 바뀌도록 시뮬레이터가 설계되어 있음. 위의 4가지 위치에서 출발하여 주차구역 안으로 안착하여 주차구역 선이 녹색으로 바뀌면 주차 성공
- 주차 도중에 키보드 1~4를 누르면 해당 출발위치로 강제 이동됨. 디버깅시 편리함.



# 시뮬레이터가 보내는 토픽 - /ar\_pose\_marker

- 현실에서 ar\_track\_alvar 패키지가 보내는 토픽과 동일
  - 토픽 이름 = /ar\_pose\_marker
  - 카메라 화면에서 AR 태그가 어떤 위치에 있는지, 어떤 자세를 하고 있는지에 대한 정보를 /ar\_pose\_marker 토픽에 담아 전송한다.
  - 실제와는 달리 시뮬레이터가 보내는 정보의 거리 단위는 m(미터)가 아니고 pixel(픽셀) 이다.
  - 자세 정보는 쿼터니언으로 표시되므로, 오일러 방식으로 바꾸는 작업이 필요.

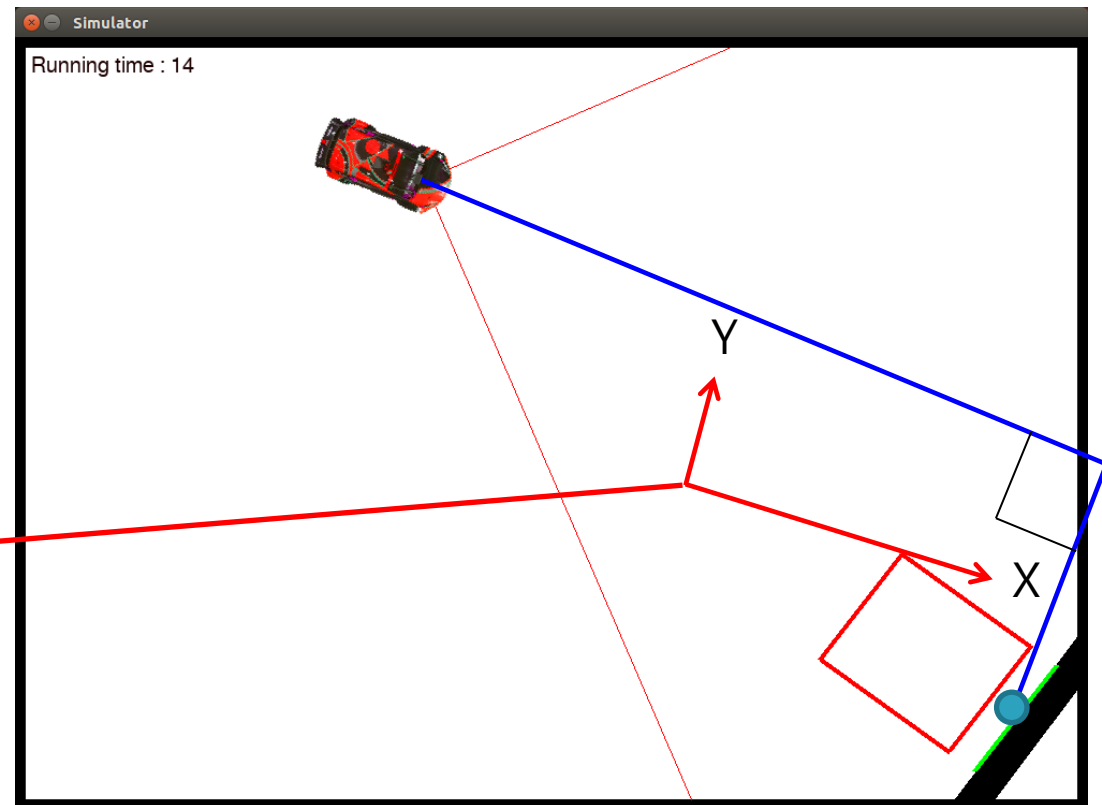
```
hscho@sparc: ~  
---  
header:  
  seq: 5773  
  stamp:  
    secs: 0  
    nsecs: 0  
  frame_id: ''  
markers:  
  -  
    header:  
      seq: 0  
      stamp:  
        secs: 1618466004  
        nsecs: 358340024  
      frame_id: "usb_cam"  
    id: 1  
    confidence: 0  
    pose:  
      header:  
        seq: 0  
        stamp:  
          secs: 0  
          nsecs: 0  
        frame_id: ''  
      pose:  
        position:  
          x: 335.838067754  
          y: 692.156433284  
          z: 0.0  
        orientation:  
          x: 0.0  
          y: 0.0  
          z: -0.0422303717216  
          w: 0.999107899931
```



# 시뮬레이터가 보내는 토픽 - /ar\_pose\_marker

- 3차원 공간에서의 좌표값 (x, y, z)
  - pose.pose.position (x, y, z)
  - 픽셀 단위의 거리 정보

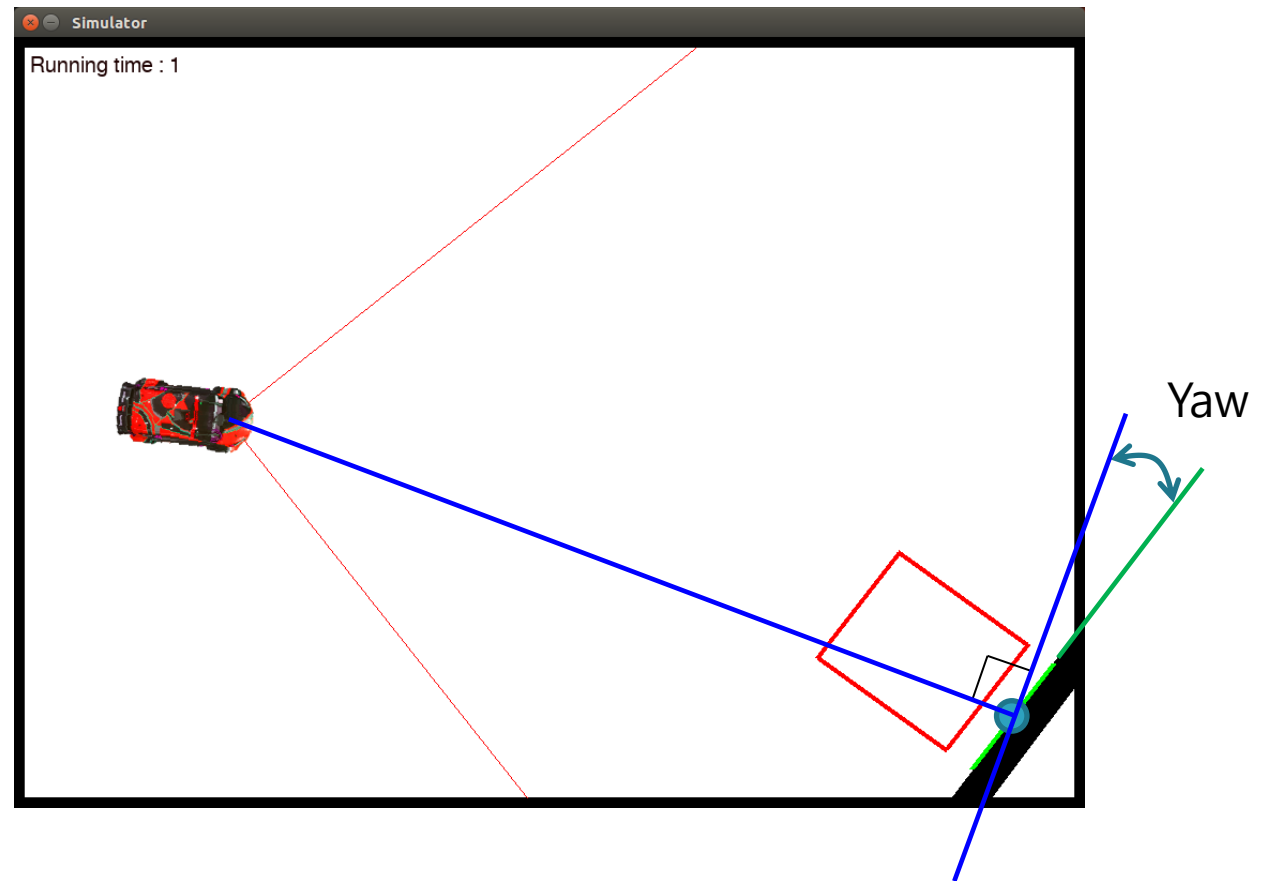
```
sungmin@machine: ~/
=====
roll  : 0.0
pitch : 0.0
yaw   : -4.6
x     : 368.0
y     : 688.0
z     : 0.0
('callback:', -50)
=====
roll  : 0.0
pitch : 0.0
yaw   : -5.0
x     : 310.0
y     : 694.0
z     : 0.0
```



# 시뮬레이터가 보내는 토픽 - /ar\_pose\_marker

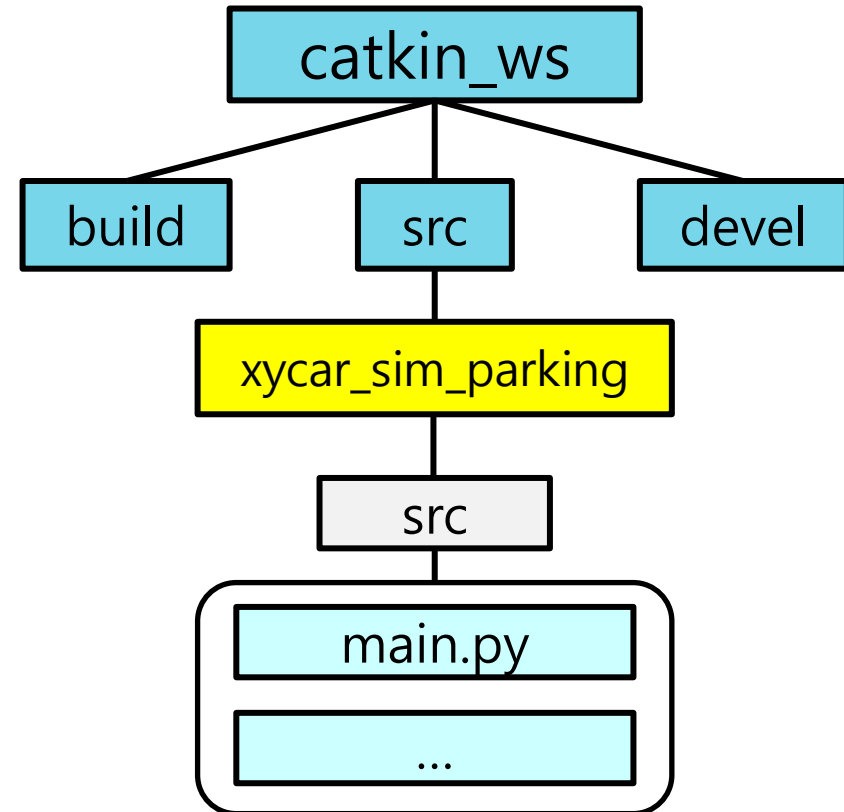
- 자세 정보값 (Roll, Pitch, Yaw)
  - pose.pose.orientation (x, y, z, w)
  - 하늘에서 보는 자동차의 진행 각도

```
sungmin@machine: ~/
=====
roll  : 0.0
pitch : -0.0
yaw   : 16.3
x : 203.0
y : 761.0
z : 0.0
('callback:', -50)
=====
roll  : 0.0
pitch : -0.0
yaw   : 16.2
x : 140.0
y : 754.0
z : 0.0
```



# 시뮬레이터 (xycar\_sim\_parking) 설치

- ROS 작업 디렉토리 아래의 src 디렉토리에서 작업
- xycar\_sim\_parking.tgz 파일을 복사한 다음에 압축 해제
  - `$ tar -zxvf xycar_sim_parking.tgz`
- xycar\_sim\_parking 패키지 빌드
  - `$ cm`



# 작업 폴더

- ar\_viewer 패키지 만들기 - ROS workspace의 src 폴더에서

- `$ catkin_create_pkg ar_viewer rospy std_msgs`
- `$ cm`

- 서브 폴더 만들기

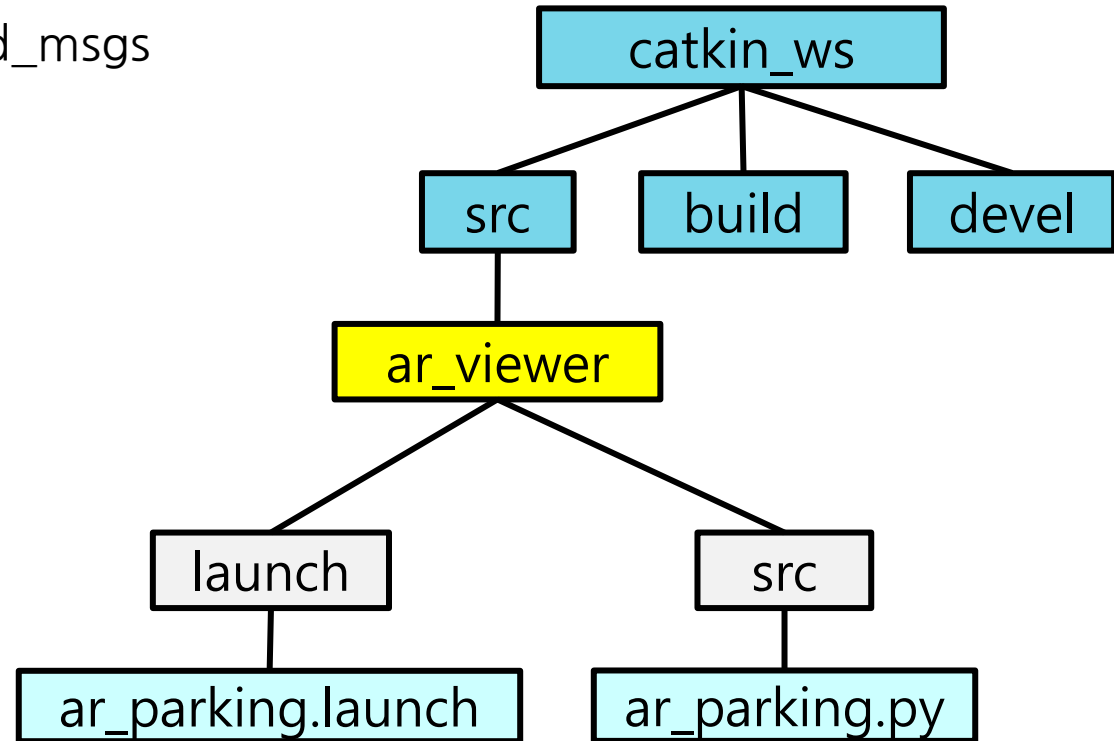
- `/launch` 폴더 만들기

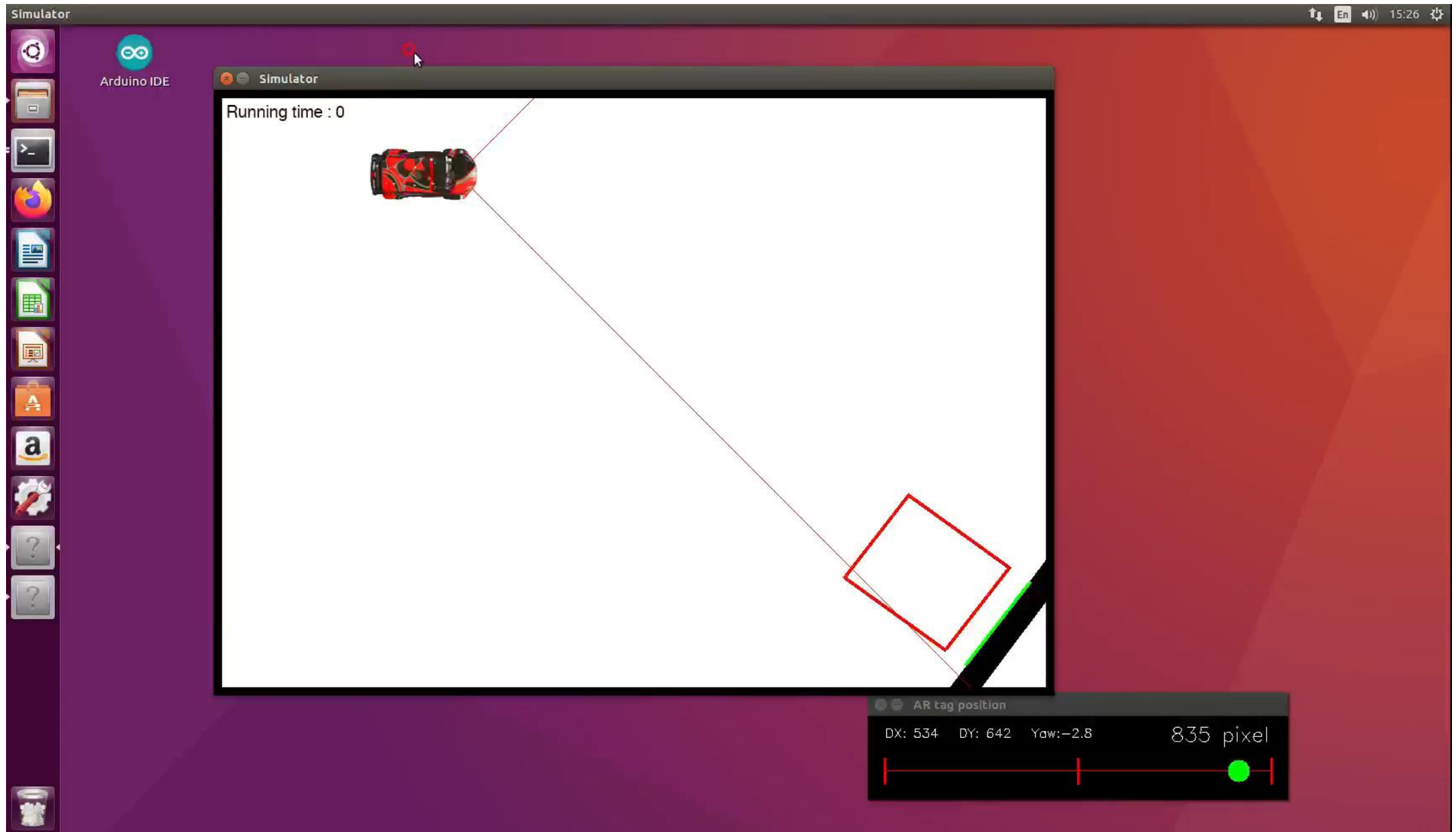
- 런치파일은 바꾸지 말것

- `ar_parking.launch` 사용

- 파이썬 코드 작성하기

- 기존 `ar_drive.py`에 코드 추가하여 제작
- `$ cp ar_drive.py ar_parking.py`
- `$ gedit ar_parking.py`





# 결과물 제출

---

- 동영상 제출
  - 실행 결과를 휴대폰이나 동영상 캡처툴로 촬영해서 제출
- 파일 제출
  - ar\_parking.py를 제출
- 문서 제출 (아래아한글)
  - 소스코드를 상세히 설명하는 문서를 작성하여 제출

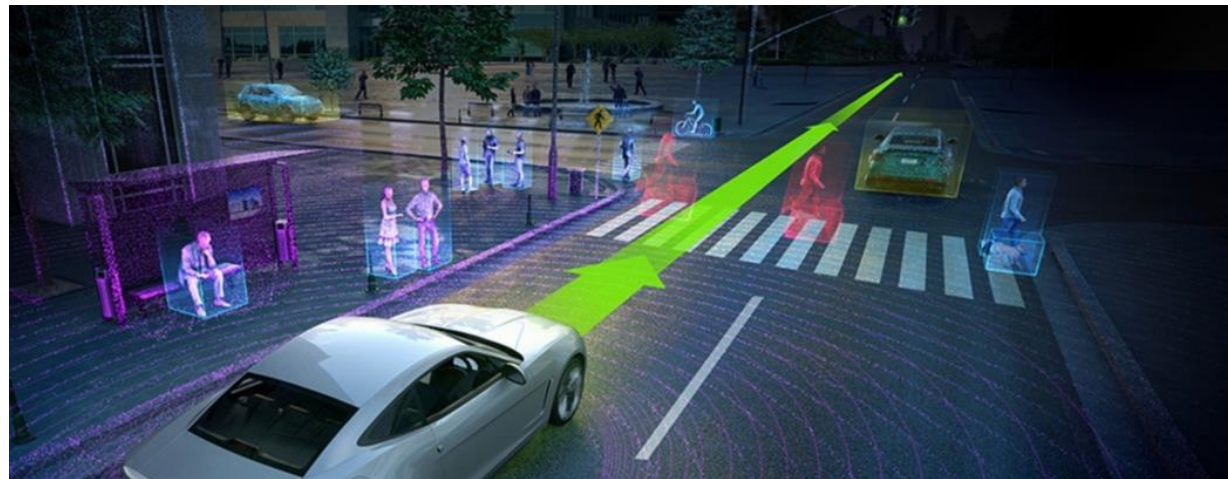
# 과제 #4

---

## 논술 과제

# 과제 설명

1. 코로나로 인하여 국민대 트랙에서 실차로 주행연습을 많이 할 수 없는 상황에서, 트랙을 사용하지 못하는 기간에는 어떻게 주행테스트/검증/개발작업을 진행할 계획인지에 대해 기술하세요.
2. 오프라인 모임을 갖기 어려운 상황인데 팀원들간의 협업은 어떻게 진행할 계획인지 이에 대해 기술하세요.
3. 각 팀원에 대한 장단점을 기술하고 본인들이 드림팀인 이유(경쟁력과 차별점)는 무엇인지 기술하세요.





# 결과물 제출

- 문서 제출 (아래아 한글)
  - 1, 2, 3번 문항에 대한 답변을 상세히 설명하는 문서를 작성하여 제출



# 과제 제출 방법

---

과제 제출 방법

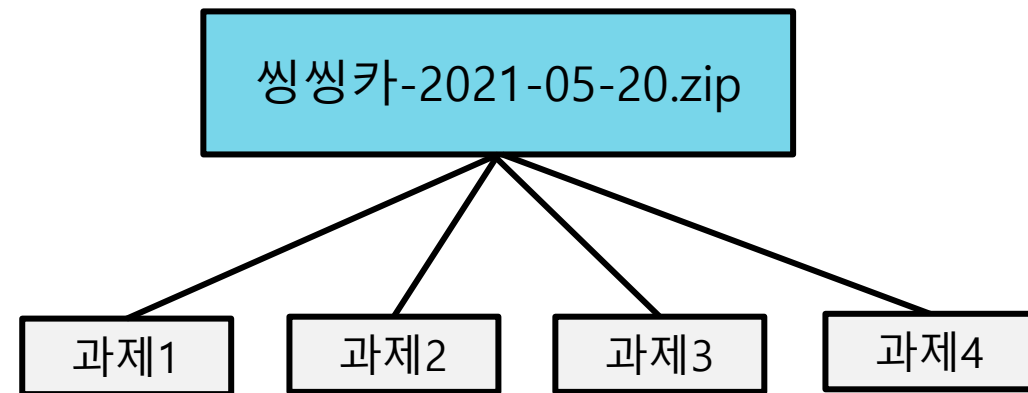
# 제출 방법

---

- 제출 기한 : 5월 28일 18시 정각까지 (기한 내에 제출하지 못한 팀은 실격)
- 제출 방법
  - 이메일 : [hscho@xytron.co.kr](mailto:hscho@xytron.co.kr)
- 첨부파일 형식
  - 팀명-날짜.zip 으로 압축하여 제출처에 기한 엄수하여 발송  
(예) 씹씹카-2021-05-20.zip

# 제출 형태

- 4개의 과제 폴더를 우측 형식에 맞게 압축



- 제출 파일

	과제1	과제2	과제3	과제4
동영상	ultra_driver.mp4	line_find.mp4	ar_parking.mp4	-
소스코드	ultra_driver.py	line_find.py	ar_parking.py	-
문서	ultra_driver.hwp	line_find.hwp	ar_parking.hwp	OO팀_소개_및_플랜.hwp

- 문서는 아래아한글(hwp), 동영상은 mp4, avi, mkv, mov 형태로 제출
- 첨부파일이 큰 경우, 네이버 또는 다음 계정 메일에서 지원하는 대용량 첨부 이용



# Q&A

---