CSE461 Software Lab 2 Report

Lab Number: Software Lab 2

Title: ROS Packages and Topics

Group Number: 4

Section: 4

Group Members:

1. Anika Ahmed, ID: 21101029
2. Nafis Chowdhury, ID: 21101034
3. Moinul Haque, ID: 21101186
4. Yeamin Hossain Shihab, ID: 21101174
5. Mirza Fahad Bin Kamal, ID: 20101399

1. Description:

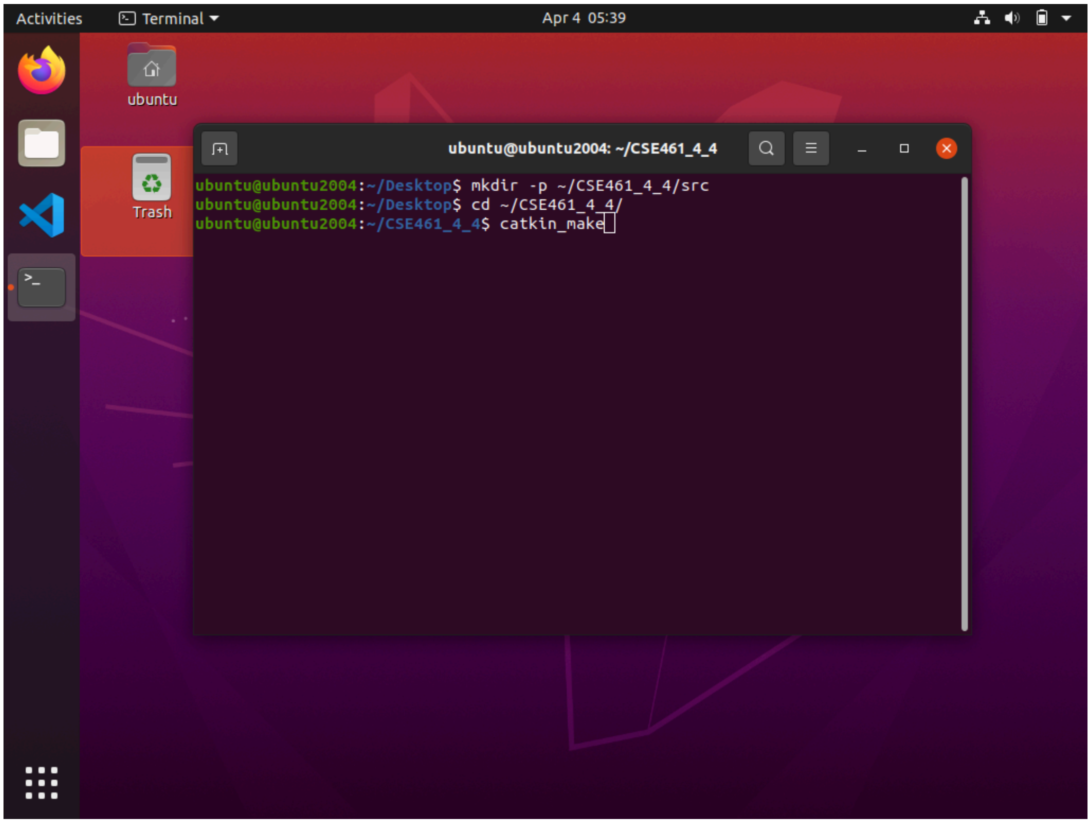
This lab report's focus is to investigate the concept of workspace, packages, and nodes. So in this lab, our prime focus was to understand the transmission of information from publisher to subscriber. And for the lab report, our task was to create a workspace and package for the husky driver. We would control the husky driver with the Python script in follow_path.py. We created the workspace and packages as taught to us in our labs. In order to control the husky driver easily we made some changes in the code. When we execute the script and run the code, we are prompted to enter speed, distance, isForward, and angle. We have to enter specific values of speed based on scenarios. Absolute values of speed is assigned to vel_msg.linear.x. So in this way, we will control the direction of the husky driver in the x-direction. We have to input either True for isForward to represent it will move forward or False to indicate it will move backward. It is done by assigning positive values of absolute speed in vel_msg.linear.x if we want to move forward and negative values of absolute speed in vel_msg.linear.x if we want to move backward. We can enter the turn we want by inputting the radian value when prompted for an angle. The value is assigned to vel_msg.angular.z if we want to turn the husky. However, if we do not want to turn the husky we will enter value 0 to ensure it. The distance variable will take the distance value we enter and convert it to float. A loop will run until the rospy is shut down, and inside the loop, we will take the present time using rospy.Time.now().to_sec() function and store it in a variable and also current_distance is also initialized to 0. Inside there is another loop that will execute until current_distance is less than the distance we entered. Inside the loop, we will publish the vel_msg to the husky driver so that it moves according to the way we want. Inside the loop, in another variable, we take the present time using rospy.Time.now().to_sec() and calculate the time interval and then multiply by speed to update current_distance. As the loop ends, vel_msg.linear.x is set to 0 to stop the robot and publish vel_msg to stop the husky driver. Therefore, in this lab, with the input we give in each scenario, we are able to control the movement of the robot and move its place from source to destination. We have added a text file containing the video link of controlling the husky driver and moving it to goal position.

2. Creating Workspace:

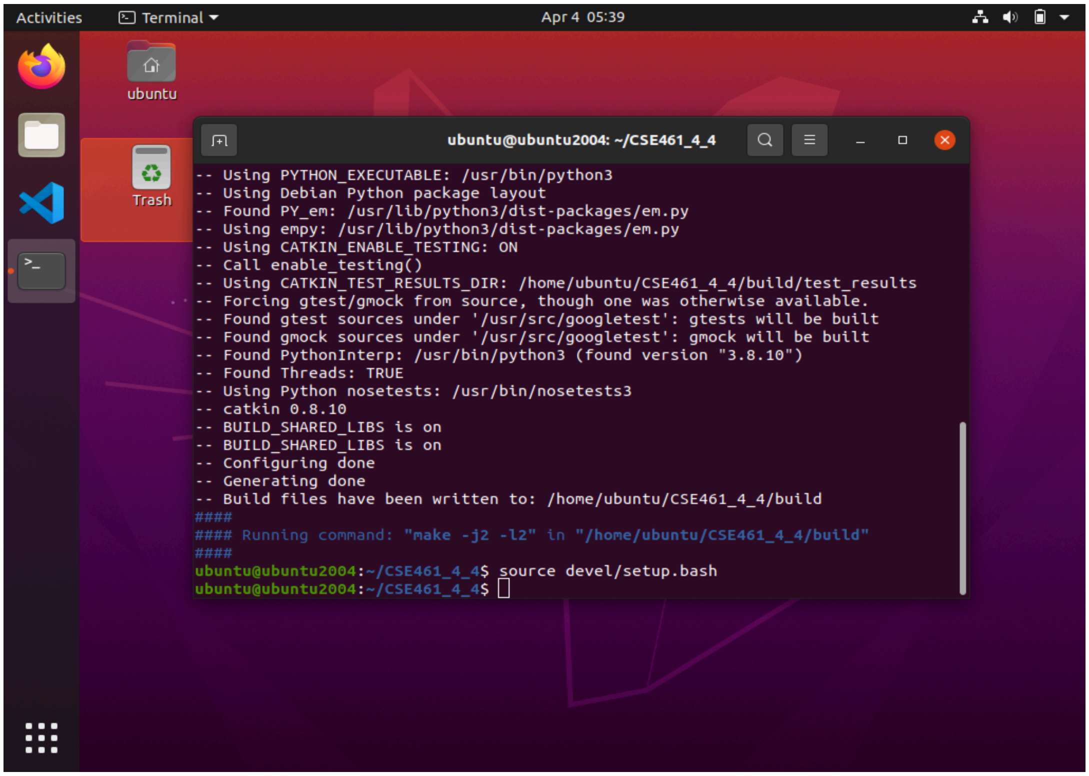
Commands to create workspace:

- mkdir -p ~/CSE461_4_4/src
- cd ~/CSE461_4_4/
- catkin_make
- source devel/setup.bash

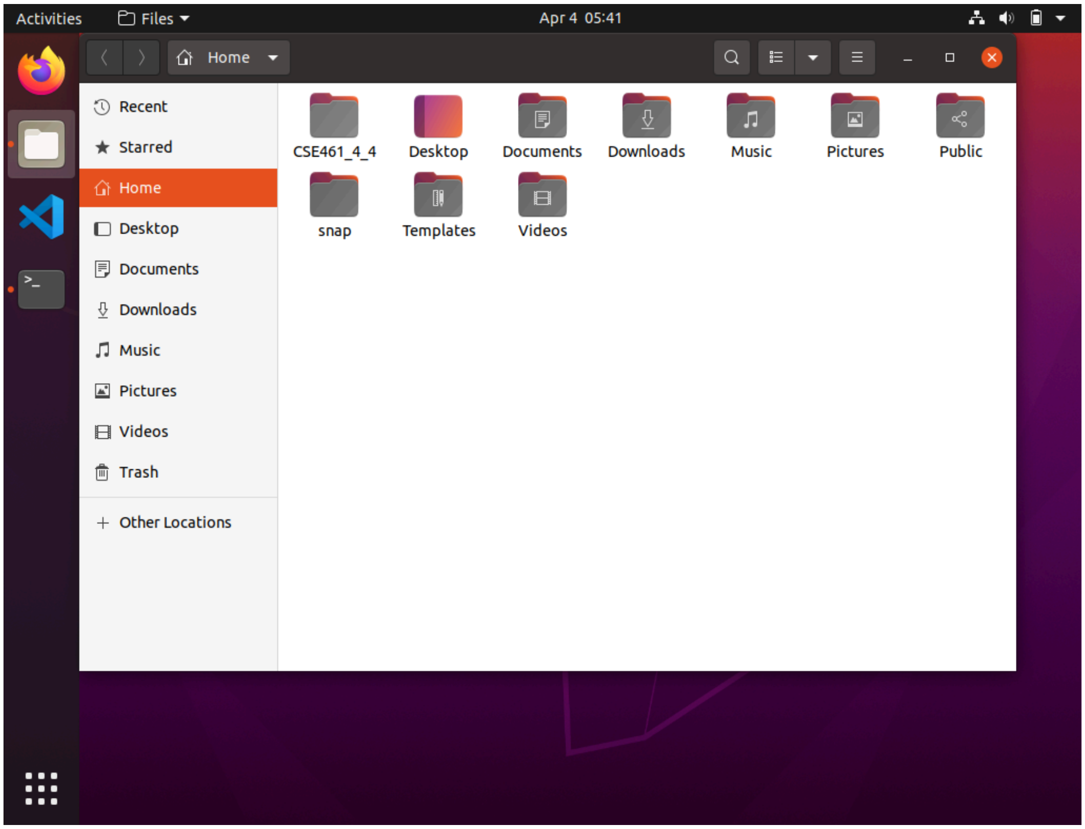
We kept our workspace in the user home directory.



```
ubuntu@ubuntu2004:~/Desktop$ mkdir -p ~/CSE461_4_4/src
ubuntu@ubuntu2004:~/Desktop$ cd ~/CSE461_4_4/
ubuntu@ubuntu2004:~/CSE461_4_4$ catkin_make
```



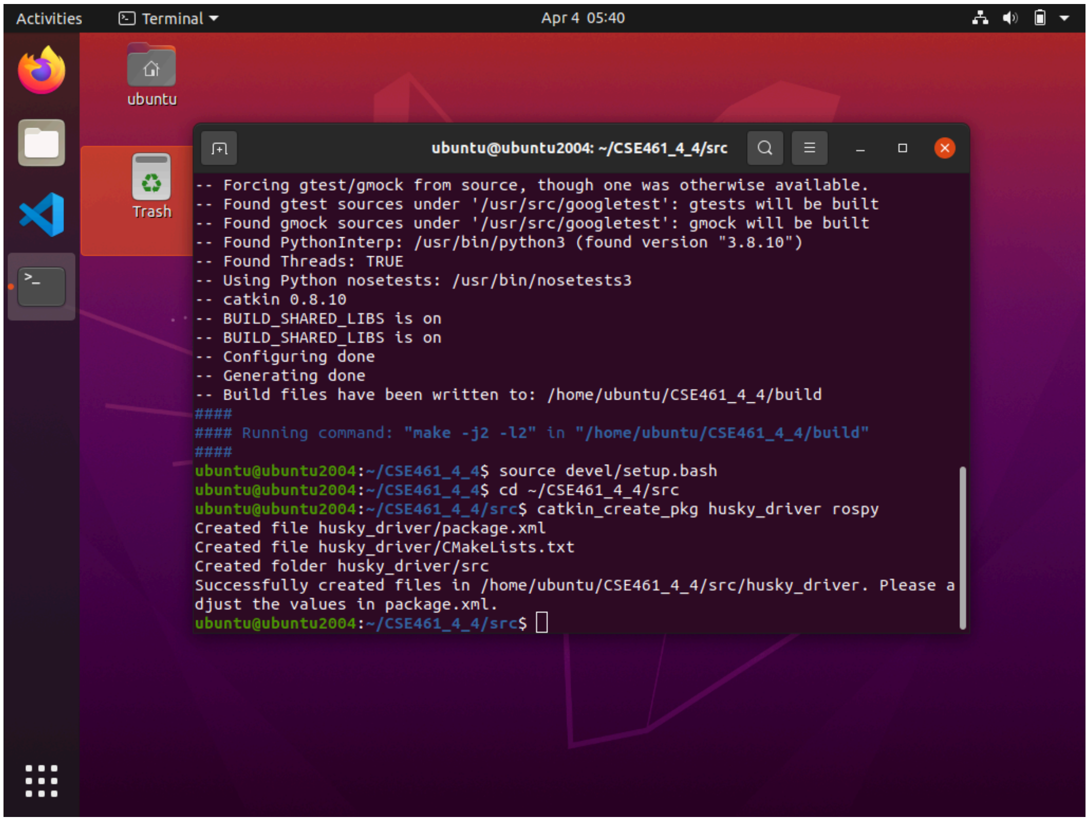
```
-- Using PYTHON_EXECUTABLE: /usr/bin/python3
-- Using Debian Python package layout
-- Found PY_em: /usr/lib/python3/dist-packages/em.py
-- Using empy: /usr/lib/python3/dist-packages/em.py
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/ubuntu/CSE461_4_4/build/test_results
-- Forcing gtest/gmock from source, though one was otherwise available.
-- Found gtest sources under '/usr/src/googletest': gtests will be built
-- Found gmock sources under '/usr/src/googletest': gmock will be built
-- Found PythonInterp: /usr/bin/python3 (found version "3.8.10")
-- Found Threads: TRUE
-- Using Python nosetests: /usr/bin/nosetests3
-- catkin 0.8.10
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- Configuring done
-- Generating done
-- Build files have been written to: /home/ubuntu/CSE461_4_4/build
####
#### Running command: "make -j2 -l2" in "/home/ubuntu/CSE461_4_4/build"
####
ubuntu@ubuntu2004:~/CSE461_4_4$ source devel/setup.bash
ubuntu@ubuntu2004:~/CSE461_4_4$ 
```



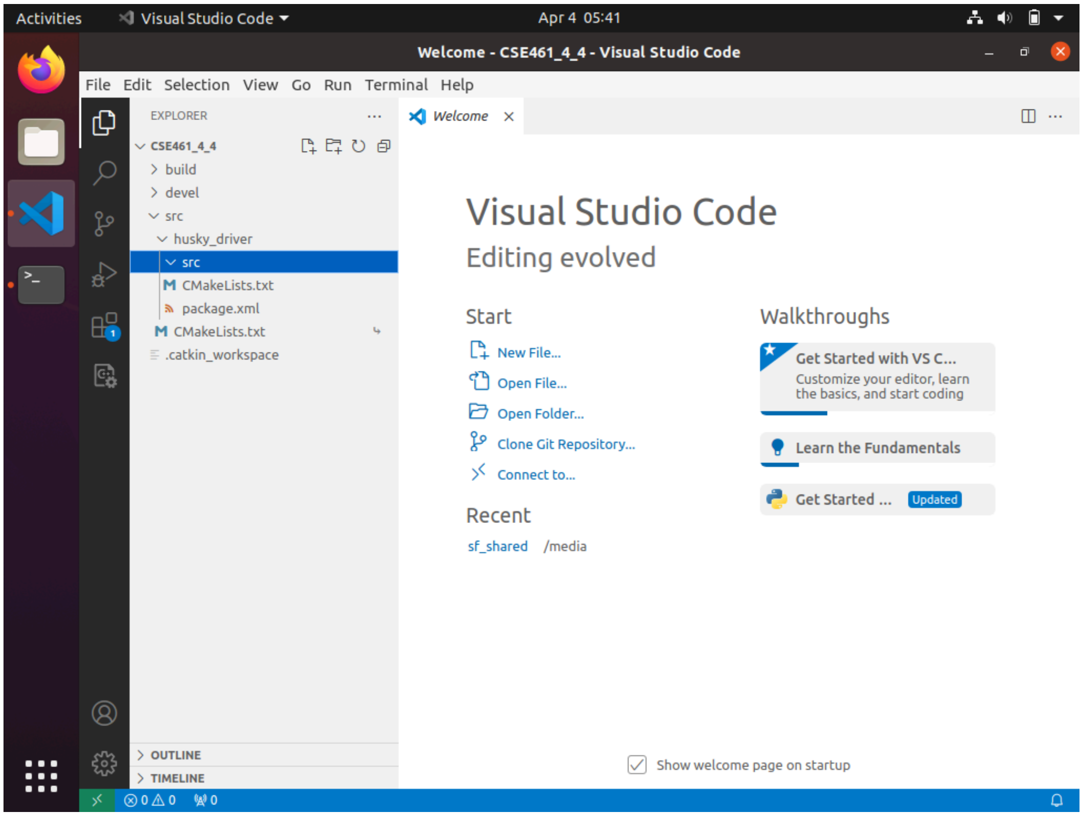
3. Creating package husky_driver

Command to create package:

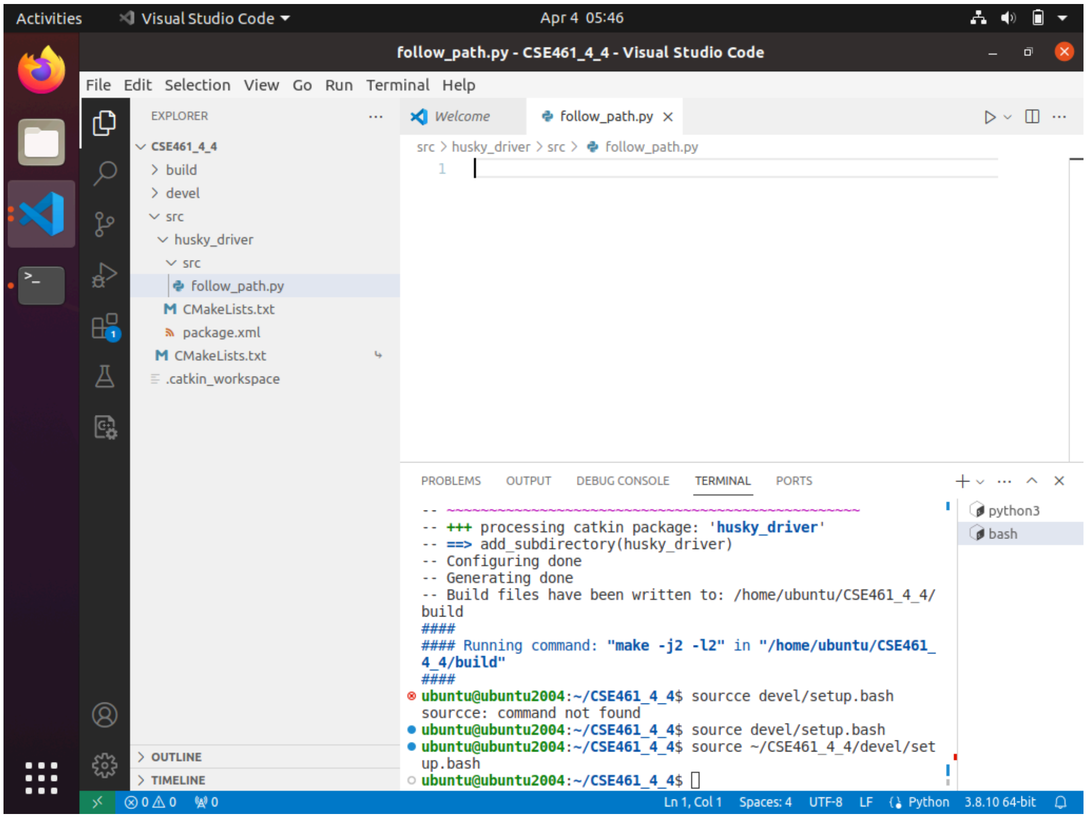
- catkin_create_pkg husky_driver rospy



```
-- Forcing gtest/gmock from source, though one was otherwise available.
-- Found gtest sources under '/usr/src/googletest': gtests will be built
-- Found gmock sources under '/usr/src/googletest': gmock will be built
-- Found PythonInterp: /usr/bin/python3 (found version "3.8.10")
-- Found Threads: TRUE
-- Using Python nosetests: /usr/bin/nosetests3
-- catkin 0.8.10
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- Configuring done
-- Generating done
-- Build files have been written to: /home/ubuntu/CSE461_4_4/build
####
#### Running command: "make -j2 -l2" in "/home/ubuntu/CSE461_4_4/build"
####
ubuntu@ubuntu2004:~/CSE461_4_4$ source devel/setup.bash
ubuntu@ubuntu2004:~/CSE461_4_4$ cd ~/CSE461_4_4/src
ubuntu@ubuntu2004:~/CSE461_4_4/src$ catkin_create_pkg husky_driver rospy
Created file husky_driver/package.xml
Created file husky_driver/CMakeLists.txt
Created folder husky_driver/src
Successfully created files in /home/ubuntu/CSE461_4_4/src/husky_driver. Please a
djust the values in package.xml.
ubuntu@ubuntu2004:~/CSE461_4_4/src$ 
```



4. Creating node follow_path.py



```
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- +++ processing catkin package: 'husky_driver'
-- ==> add_subdirectory(husky_driver)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/ubuntu/CSE461_4_4/
build
####
#### Running command: "make -j2 -l2" in "/home/ubuntu/CSE461_
4_4/build"
####
ubuntu@ubuntu2004:~/CSE461_4_4$ sourcce devel/setup.bash
sourcce: command not found
ubuntu@ubuntu2004:~/CSE461_4_4$ source devel/setup.bash
ubuntu@ubuntu2004:~/CSE461_4_4$ source ~/CSE461_4_4/devel/set
up.bash
ubuntu@ubuntu2004:~/CSE461_4_4$ 
```

5. Modified Code:

We modified the existing code a little. We introduced a variable named angle which will take input from the user on how much to rotate. The value of the angle is assigned to vel_msg.angular.z. So in this way, we can control how much to rotate instead of changing the code every time we want to take a turn

```
#!/usr/bin/python3
import rospy
from geometry_msgs.msg import Twist
def move():

   rospy.init_node('follow_path', anonymous=True)
   velocity_publisher = rospy.Publisher('/husky_velocity_controller/cmd_vel',


   Twist, queue_size=10)
```

```
vel_msg = Twist()

print("Let's move your robot")
speed = input("Input your speed:")
distance = input("Type your distance:")
isForward = input("Foward?: ")
angle = input("Type your angle:")
speed = float(speed)
distance = float(distance)
angle = float(angle)

if(isForward):
    vel_msg.linear.x = abs(speed)
else:
    vel_msg.linear.x = -abs(speed)

vel_msg.linear.y = 0
vel_msg.linear.z = 0
vel_msg.angular.x = 0
vel_msg.angular.y = 0
vel_msg.angular.z = angle
while not rospy.is_shutdown():

    t0 = rospy.Time.now().to_sec()
    current_distance = 0

    while(current_distance < distance):

        velocity_publisher.publish(vel_msg)

        t1 = rospy.Time.now().to_sec()

        current_distance = speed*(t1-t0)
```

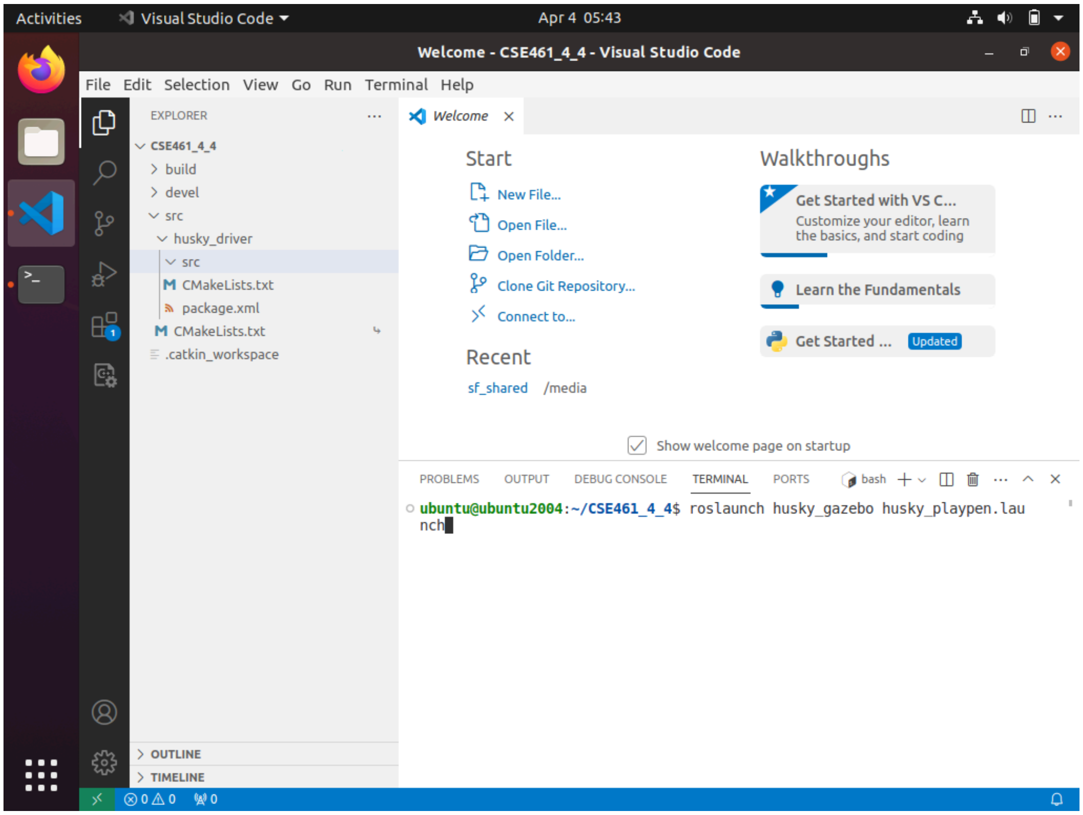
```
            vel_msg.linear.x = 0

            velocity_publisher.publish(vel_msg)
    if __name__ == '__main__':
        try: move()
        except rospy.ROSInterruptException: pass
```

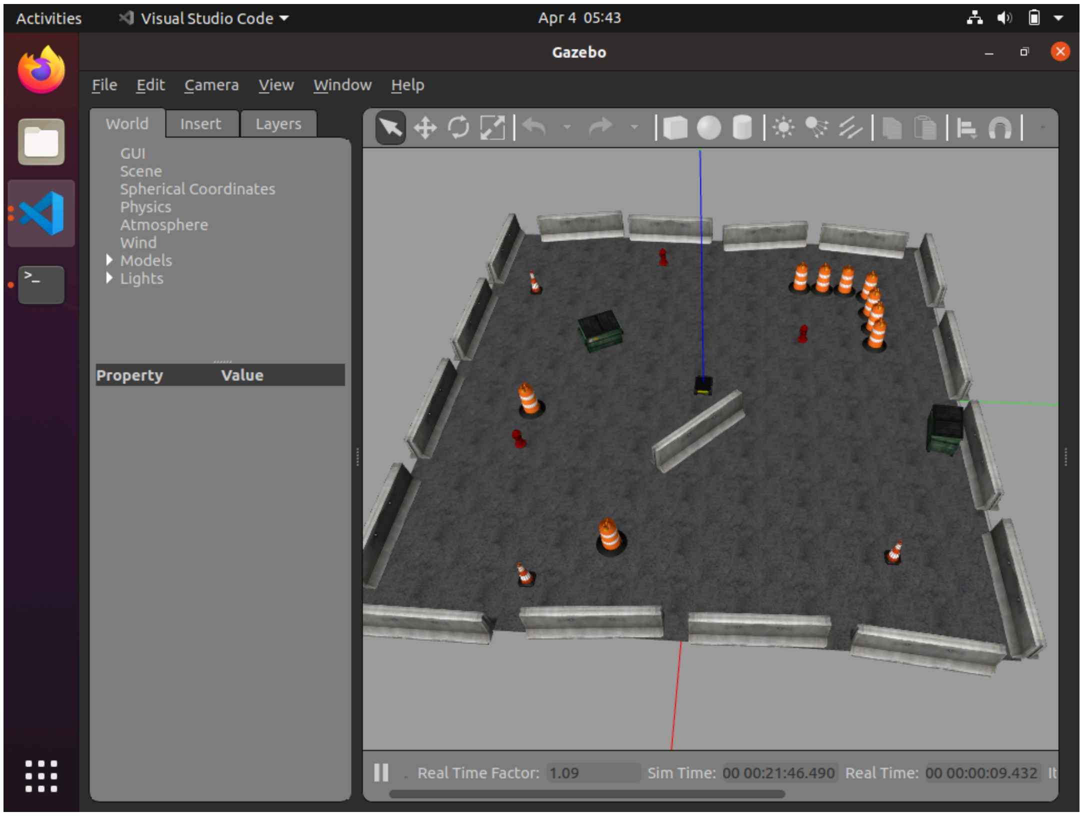
6. Launching Husky node:

Command:

- roslaunch husky_gazebo husky_playpen.launch



```
ubuntu@ubuntu2004:~/CSE461_4_4$ roslaunch husky_gazebo husky_playpen.lau
nch
```



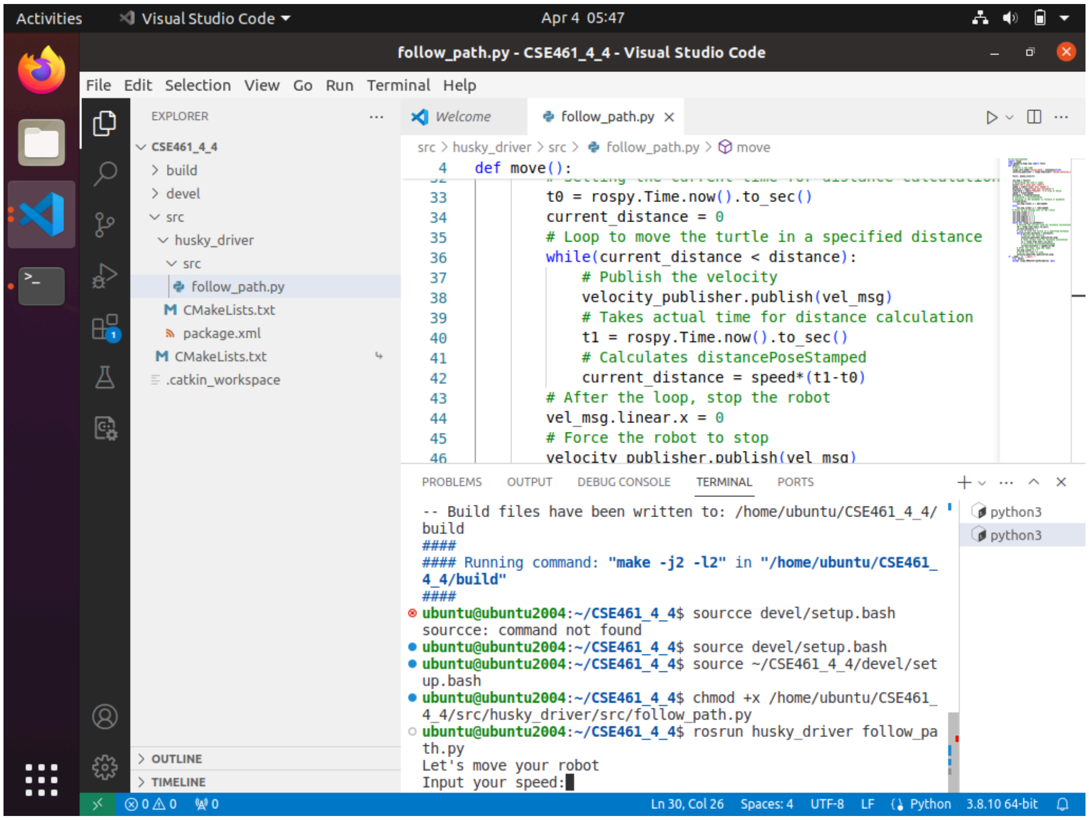
7. Making the python script executable:

After modifying the package, we will build workspace again

- cd ~/CSE461_4_4/
- catkin_make
- source devel/setup.bash
- source ~/CSE461_4_4/devel/setup.bash

Executing the python script

- chmod +x /home/ubuntu/CSE461_4_4/src/husky_driver/src/follow_path.py
- rosrun husky_driver follow_path.py



```python
def move():
        t0 = rospy.Time.now().to_sec()
        current_distance = 0
        # Loop to move the turtle in a specified distance
        while(current_distance < distance):
            # Publish the velocity
            velocity_publisher.publish(vel_msg)
            # Takes actual time for distance calculation
            t1 = rospy.Time.now().to_sec()
            # Calculates distancePoseStamped
            current_distance = speed*(t1-t0)
        # After the loop, stop the robot
        vel_msg.linear.x = 0
        # Force the robot to stop
        velocity_publisher.publish(vel_msg)
```

```
-- Build files have been written to: /home/ubuntu/CSE461_4_4/
build
####
#### Running command: "make -j2 -l2" in "/home/ubuntu/CSE461_
4_4/build"
####
ubuntu@ubuntu2004:~/CSE461_4_4$ sourcce devel/setup.bash
sourcce: command not found
ubuntu@ubuntu2004:~/CSE461_4_4$ source devel/setup.bash
ubuntu@ubuntu2004:~/CSE461_4_4$ source ~/CSE461_4_4/devel/set
up.bash
ubuntu@ubuntu2004:~/CSE461_4_4$ chmod +x /home/ubuntu/CSE461_
4_4/src/husky_driver/src/follow_path.py
ubuntu@ubuntu2004:~/CSE461_4_4$ rosrun husky_driver follow_pa
th.py
Let's move your robot
Input your speed:
```

We have added a text file containing the video link of controlling the husky driver and moving it to goal position.

The video can also be accessed via this link also: [video link]