به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

دانشکده مهندسی مکانیک

رباتیک و مکاترونیک

مینی پروژه شماره ۳

نام و نام خانوادگی	محیا شهشهانی
شماره دانشجویی	۸۱۰۱۹۹۵۹۸
تاریخ ارسال گزارش	۱۸ خرداد ۱۴۰۳

Table of Contents

Part 1 – Introduction

The Robot Operating System (ROS) is a popular open-source framework for developing robotic applications, and the turtlesim simulator is a basic tool in ROS that provides a 2D environment for learning and experimenting. In this project, the goal is to utilize turtlesim to draw specified characters and then implement a rotating behavior for the turtles, allowing the student to gain hands-on experience with ROS concepts such as topics, messages, and launch files, while also applying programming skills in a robotics context.

Figure 1: Turtlesim environment

Part 2 – Drawing Characters

I was asked to draw two characters using turtles. For the letter 'M', I used eleven turtles, programming their positions to construct the desired shape. I also drew the Greek 'Omega' character, using eight turtles, applying a similar turtle graphics approach.

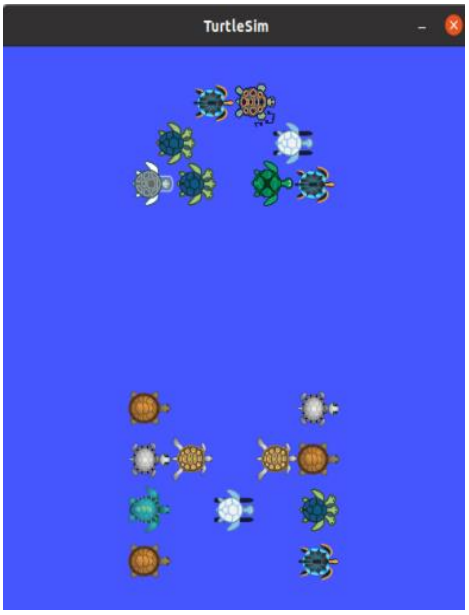

Figure2: writing letter M and Omega with turtles in turtlesim

The turtlesim_node, provides sevices like ‘spawn’ and ‘kill’. The ‘spawn’ service is used to create a new turtle at a specified position. It takes parameters such as the x and y coordinates of the turtle's initial position, its orientation (theta), and a name for the turtle. The ‘kill’ is used to delete an existing turtle from the simulation.

I wrote the ‘draw_m.py’ script which uses service proxies to interact with the ROS turtlesim environment.

With this command in the terminal, the outup will be like the figure shown above.

~$ roslaunch draw_m draw_m.launch

I’ll talk about challenges later in details.

Part 3 – Rotating the Turtles

The goal of this part of the project was to rotate all turtles in the turtlesim simulation simultaneously by sending right velocity commands. The rotation is shown in two frames down below!

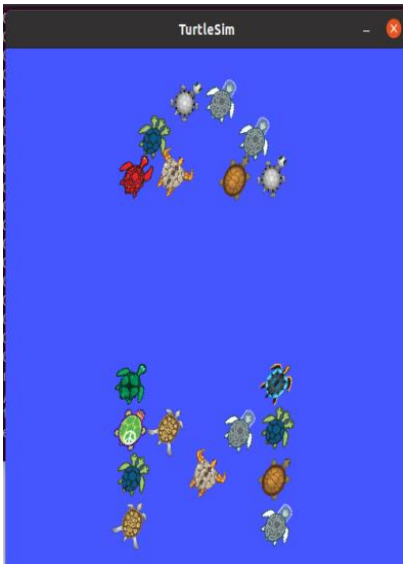

Figure 3: Rotating the Turtles, frame one.

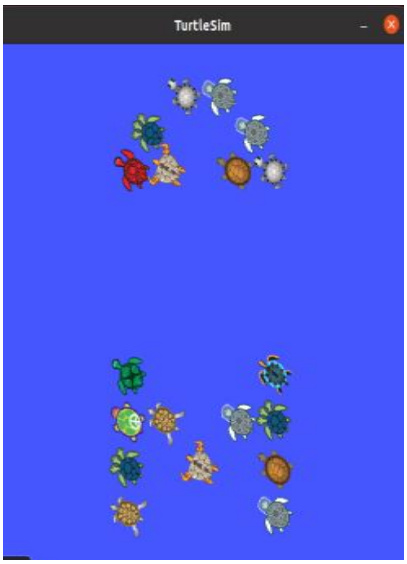

Figure 4: Rotating the Turtles, frame two.

Approach:
As we have learned from the class, I setup a node named "rotating_node" and then a subscription to the */turtle*/cmd_vel* topic to receive velocity commands for rotating each turtle. The velocity also was set to 1Hz for all 19 turtles. I defined a twist message to rotate these turtles CCW.

ROS Components Used:

- **Message Types**
  ***Twist**:* Used to control the movement of turtles by publishing velocity commands.
- **Services**
  ***Spawn**:* A service used to create new turtles at specified positions.
  ***Kill**:* A service used to delete existing turtles by name.
- **Topics**
  /turtle1/cmd_vel to */turtle19/cmd_vel*: Topics to which *Twist* messages are published to control the turtles' velocities.

A launch file is created to define the project execution. This file will specify all the required nodes for drawing the assigned letters. Running the launch file will execute the turtlesim_node and start the letter drawing process. The turtles will then rotate continuously until the process is stopped with Ctrl+C.

To run the launch file: ***~$ roslaunch draw_m draw_m.launch***

Part 4 – Moving turtles with arrow keys (Bonus)

As I was told, I made the first letter, which is M, controllable with arrow keys. I added a node named *'keyboad_control'* to the project. With the help of *'pynput'* I was able to control the movement of turtles. Based on the pressed arrow keys, the node determined the direction of movement (forward, backward, left, right) and published corresponding velocity commands to the turtles using the '*geometry_msgs/Twist'* message type.

I updated the launch file, so with the mentioned command before, this program works properly.

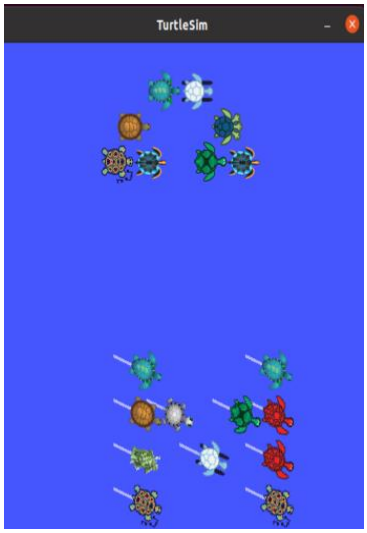

Figure 5: Moving with arrow keys, frame one.

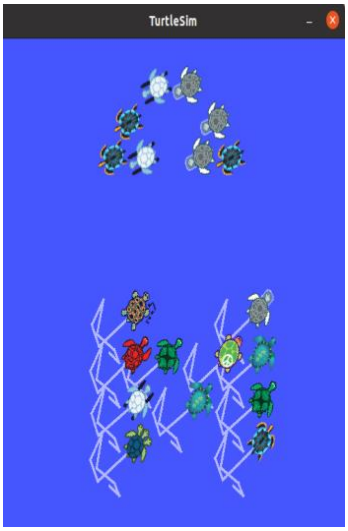

Figure 6: Moving with arrow keys, frame two.

Figure 7: Ctrl + C stops the process.

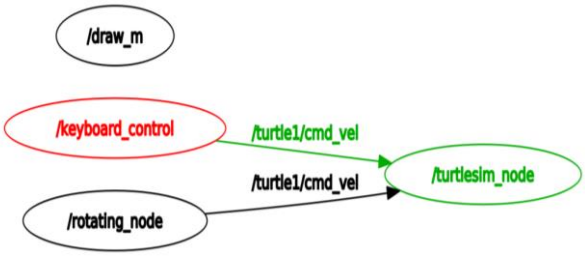

Figure 8: Nodes Graph

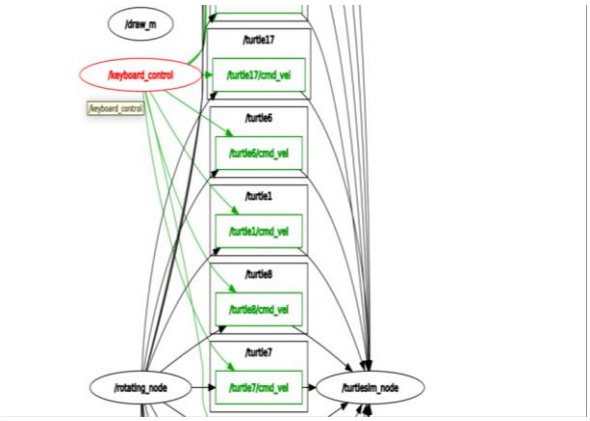

Figure 9: Nodes and (some of the) Topics Graph

Part 5 – Challenges and Solutions

While implementing this project I faced several challenges and errors which I am going to talk about them following.

- Launch File Error:
  Multiple files named [spawn_turtles.launch] in package [spawn_pckg]:
  It was caused by multiple files with the same name, I solved the problem with specifying the full path. To ensure the program runs correctly I changed the name of the file and created a new directory – draw_m.

- Package or Launch File Setup Error, mistake in package structure:
  [draw_m.launch] is neither a launch file in package [draw_m] nor is [draw_m] a launch file name
  The error caused because I forgot to make changes in launch file. I fixed that and the problem solved.
  The latest version of launch file contains all the nodes.

- Node Execution Error:
  ERROR: cannot launch node of type [draw_m/draw_m.py]: Cannot locate node of type [draw_m.py] in package [draw_m]. Make sure file exists in package path and permission is set to executable (chmod +x)
  I fixed this error while changing the program I was writing for the turtles. The error fixed because I forgot to access the permission of execution.

- Positioning/Moving all nineteen turtles' challenge:
  Turtle number one did not move. I was not able to position it, so I added a kill service to fix the problem.

- Using arrow keys challenge:
  I was not able to move turtles while they were rotating, and I couldn't run the move code in the terminal because it didn't work. So, I had to kill the 'rotating_node',

and then use the keys. I furthered fixed this with the help of ‘pynput’ now there is no need to killing nodes and the whole project works with launching the file.

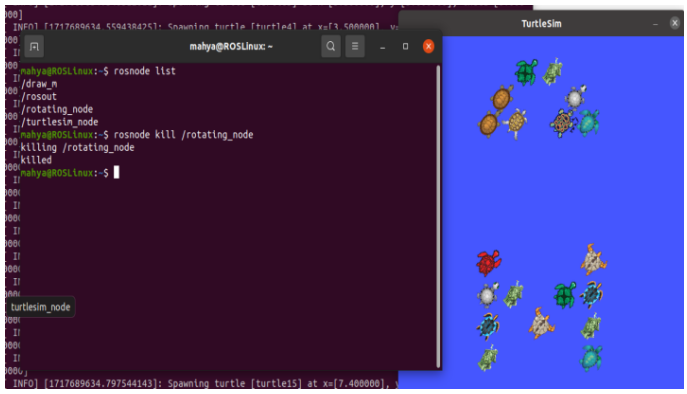

Figure 10: Killing node approach.

- #!/usr/bin/env python3:
  I learned my turtlesim only collaborates with *python 3*, not just *python*. I’m not sure why but I was able to fix the problem by adding the python version.

Node Graph (without bonus part)

- Nodes: Rotating node is responsible for publishing velocity commands to rotate the turtles.
- Topics:*/turtle1/cmd_vel* to */turtle19/cmd_vel*: Each topic corresponds to a turtle's velocity control.

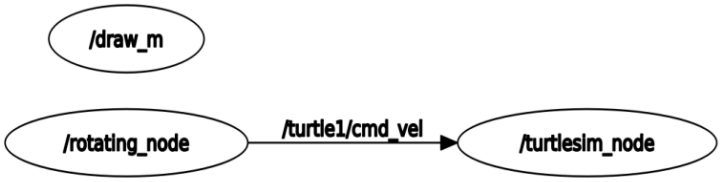

Figure 11: Nodes Graph

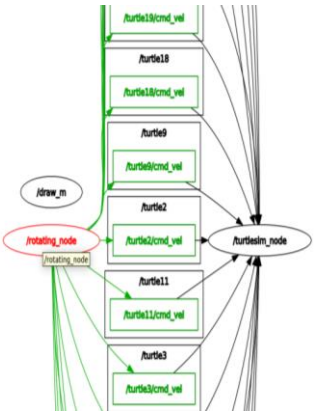

Figure 12: Nodes and (some of the) Topics Graph

Part 6 – Project file overview and description

1. draw_m.zip

- Contains the complete project without the bonus part.
- Executable using the command ***~$ roslaunch draw_m draw_m.launch***
- The turtles forming the letter 'M' are not controllable and only rotate, like the 'Omega' turtles.

2. draw_m_bonus.zip

- Includes the complete project with the bonus part.
- Also executable with ***~$ roslaunch draw_m draw_m.launch***
- Offers additional functionality compared to the regular version, allowing control over the 'M' turtles.

Both files contain ‘launch’ file, ‘scripts’, ‘CMakeLists’ and ‘package.xml’.

I’ve implemented this project in the ‘src’ directory of ‘catkin_ws’ in my linux ubunto virtual machine.