



Project Report

Robotics (MECH-M-2-ROB-ROB-ILV)

Go Turtlebot Go!

Mechatronics-Smart Technologies | Master

2nd semester

Lecturer: Daniel T. McGuiness, Ph.D

Group: MA-MECH-24-VZ

Author: Elias Karner

July 4, 2025

Contents

| | | |
|----------|---|------------|
| 1 | Introduction | 1 |
| 2 | Task Description | 1 |
| 3 | Implementation and Procedure | 1 |
| 3.1 | Python Node: turtle_controller.py | 1 |
| 3.2 | Automation Script: turtlesimAutomata.bash | 2 |
| 4 | Testing and Observations | 2 |
| | List of Figures | III |

1 Introduction

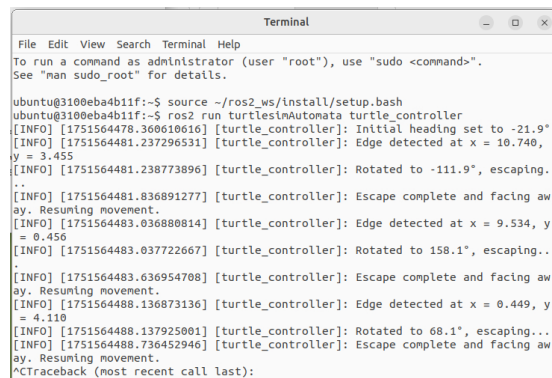
The objective of the assignment was to demonstrate proficiency in ROS 2 Humble and Python development by simulating a mobile robot using the `turtlesim` package.

The project involves spawning a turtle in the middle of a window with a random angle that explores the simulation window, detecting walls, and rotating to avoid collisions. The task integrates principles of robotic navigation and system integration in ROS 2.

2 Task Description

The goal was to implement the following behaviour:

- The turtle begins with random heading.
- Upon hitting the simulation boundary (edge) (see Figure 2.1):
 - It prints “edge detected!” to the terminal.
 - Rotates exactly 90° away from the wall.
 - Continues moving in the new direction.



```
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@3100eba4b11f:~$ source ~/ros2_ws/install/setup.bash
ubuntu@3100eba4b11f:~$ ros2 run turtlesimAutomata turtle_controller
[INFO] [1751564478.368610616] [turtle_controller]: Initial heading set to -21.9°
[INFO] [1751564481.237296531] [turtle_controller]: Edge detected at x = 10.740,
y = 3.455
[INFO] [1751564481.238773896] [turtle_controller]: Rotated to -111.9°, escaping..
[INFO] [1751564481.836891277] [turtle_controller]: Escape complete and facing aw
ay. Resuming movement.
[INFO] [1751564483.036880814] [turtle_controller]: Edge detected at x = 9.534, y
= 0.456
[INFO] [1751564483.037722667] [turtle_controller]: Rotated to 158.1°, escaping..
[INFO] [1751564483.636954708] [turtle_controller]: Escape complete and facing aw
ay. Resuming movement.
[INFO] [1751564488.136873136] [turtle_controller]: Edge detected at x = 0.449, y
= 4.110
[INFO] [1751564488.137925001] [turtle_controller]: Rotated to 68.1°, escaping...
[INFO] [1751564488.736452946] [turtle_controller]: Escape complete and facing aw
ay. Resuming movement.
^CTraceback (most recent call last):
```

Figure 2.1: Terminal output when detecting the wall

- This loop runs until it gets stopped.
- A bash script automates the full installation:
 - Creates the ROS 2 workspace and package structure.
 - Moves the `turtlesimAutomata` package from Downloads.
 - Updates files from personal repository on GitHub.
 - Builds and sources the workspace and gives further instructions in the first terminal.
 - Launches the `turtlesim` GUI.
 - Deletes the Assignment folder after installation (not .zip file).

3 Implementation and Procedure

3.1 PYTHON NODE: TURTLE_CONTROLLER.PY

The controller is designed as a state machine with three main states:

- `init`: Sets a random heading using `teleport_absolute`.
- `move`: Moves straight until a boundary is encountered.
- `escape`: Moves briefly forward after turning away.

Wall detection logic classifies the closest wall (left, right, top, bottom) and computes the most appropriate 90° turn. After an “escape” period, the controller checks if the turtle is moving away from the wall, and corrects orientation if necessary. This is necessary because in a case the turtle moves almost perpendicular to a wall. The turtle would rotate and then be almost parallel to the wall. After the escape time has elapsed, it would still be behind the margin and would rotate again. For this reason it is required to check if the turtle is already moving away from the wall so another rotation is not necessary.

3.2 AUTOMATION SCRIPT: `TURTLESIMAUTOMATA.BASH`

The bash script automates the full setup and includes:

- Workspace and source folder creation.
- Automatic removal of previous versions.
- Moving the `turtlesimAutomata` package from the Downloads directory.
- Overwriting critical files using GitHub `curl` calls.
- Setting execution permissions.
- Running `colcon build` and sourcing the environment.
- Launching the `turtlesim` GUI and the control of the turtle.
- Deleting the local Assignment folder to clean up (.zip file does not get deleted).

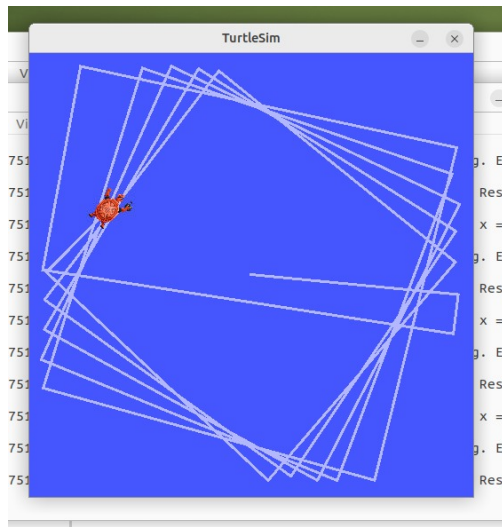
4 Testing and Observations

Working features:

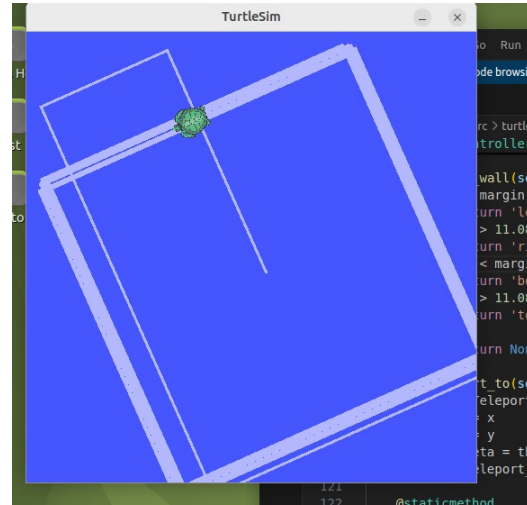
- Wall detection and 90° turn behaviour are consistent.
- Initial heading works reliably using teleportation.
- Redundant rotations are avoided after escaping a wall.

Challenges and fixes: The initial algorithm for controlling the turtles heading had another state which was *rotating*. The idea was to set the angular velocity of the turtle to turn the 90°. The problem with this method was the accuracy of the angle (see Figure 4.1a). With the method of directly changing the angle and teleport the turtle into the next heading helped improving the accuracy (see Figure 4.1b). Another problem can be seen in Figure 4.1b at the bottom and right edge. In the code the margin to the edge is for all four the same. But some turtles move behind the margin. I think this is because the center of the turtle is not aligning with the center of the image of the turtle. This problem does not occur for all turtles, just for some of them.

In Figure 4.2 the red circle shows a problem which I could not get rid of. When the turtle approaches the wall almost perpendicular its almost parallel after the rotation. It detects that its moving away from the wall but still its turning. So there must be a mistake in the logic. This error occurs not often but still it could be more robust.



(a) rotation with angular velocity



(b) rotation with directly changing the angle

Figure 4.1: Two rotations methods in comparison

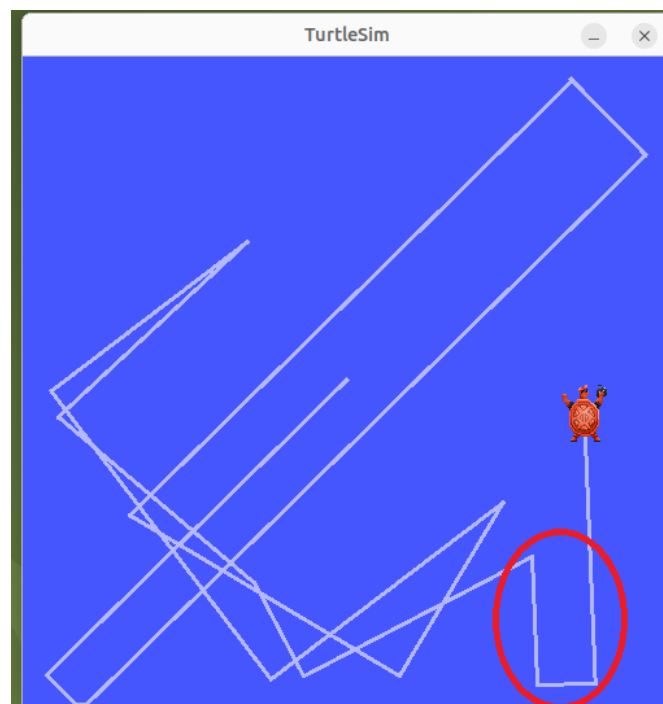


Figure 4.2: Problem of the current algorithm

List of Figures

| | | |
|-----|---|---|
| 2.1 | Terminal output when detecting the wall | 1 |
| 4.1 | Two rotations methods in comparison | 3 |
| 4.2 | Problem of the current algorithm | 3 |