Bahria University

Islamabad Campus

Robotics

Semester Project Report

Submitted by:

Muhammad Taqui

01-136221-021

Babar Ali

01-136221-051

Submitted to:

Sir Adil Khan

Project Report: Wall Following Robot

Introduction

The goal of this project is to create a wall-following robot controller using the Webots simulation environment. The robot should be able to navigate along a wall while avoiding obstacles.

Implementation Details

Robot Configuration

- The robot consists of two wheels: a left wheel and a right wheel.
- Eight proximity sensors (ps0 to ps7) are used to detect obstacles around the robot.

Controller Logic

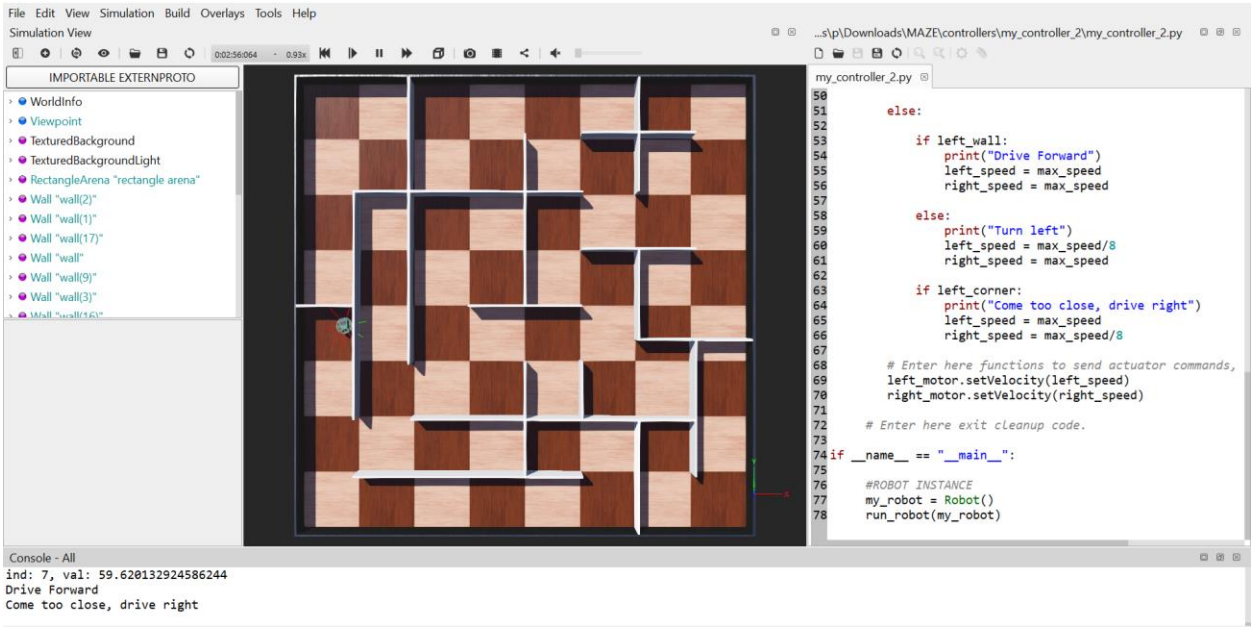
1. **Initialization:**
    - Set the time step for simulation.
    - Define the maximum speed for the robot.
2. **Motor Configuration:**
    - Enable the left and right motors.
    - Set the initial motor positions and velocities.
3. **Sensor Setup:**
    - Enable the proximity sensors.
    - Read sensor values during each simulation step.
4. **Main Loop:**
    - While the simulation is running:
        - Read sensor data from all eight proximity sensors.
        - Determine if there is a wall on the left, a corner on the left, or an obstacle in front.
        - Adjust the left and right motor speeds accordingly:
            - If there's a front wall, turn right in place.
            - If there's a left wall, drive forward.
            - If there's a left corner, steer right to avoid collision.
5. **Actuator Commands:**
    - Set the left and right motor velocities based on the calculated speeds.
6. **Exit Cleanup:**
    - Perform any necessary cleanup before exiting the simulation.

Code Implementation

```
# Import necessary modules
from controller import Robot

def run_robot(robot):
    # Set the time step
    timestep = int(robot.getBasicTimeStep())
    max_speed = 6.28

    # Enable motors
    left_motor = robot.getMotor('left wheel motor')
    right_motor = robot.getMotor('right wheel motor')
    left_motor.setPosition(float('inf'))
    left_motor.setVelocity(0.0)
    right_motor.setPosition(float('inf'))
    right_motor.setVelocity(0.0)

    # Enable proximity sensors
    prox_sensors = []
    for ind in range(8):
        sensor_name = 'ps' + str(ind)
        prox_sensors.append(robot.getDistanceSensor(sensor_name))
        prox_sensors[ind].enable(timestep)

    # Main loop
    while robot.step(timestep) != -1:
        # Read sensor data
        left_wall = prox_sensors[5].getValue() > 80
        left_corner = prox_sensors[6].getValue() > 80
        front_wall = prox_sensors[7].getValue() > 80

        # Calculate motor speeds
        left_speed = max_speed
        right_speed = max_speed

        if front_wall:
            print("Turn right in place")
            left_speed = max_speed
            right_speed = -max_speed
        else:
            if left_wall:
                print("Drive forward")
                left_speed = max_speed
                right_speed = max_speed
            else:
                print("Turn left")
                left_speed = max_speed / 8
                right_speed = max_speed
            if left_corner:
                print("Come too close, drive right")
                left_speed = max_speed
                right_speed = max_speed / 8

        # Set motor velocities
        left_motor.setVelocity(left_speed)
```

```
        right_motor.setVelocity(right_speed)

if __name__ == "__main__":
    # Create robot instance
    my_robot = Robot()
    run_robot(my_robot)
```

Sample Outputs



GitHub Repository

You can find the complete code and additional resources in the GitHub repository.

BSAI-Sem_Projects-BahriaUniversity/ROBOTICS-PROJECT-SEM05 at main · itaqiz/BSAI-Sem_Projects-BahriaUniversity (github.com)

Conclusion

This controller allows the robot to follow walls and avoid obstacles effectively. You can further enhance it by fine-tuning the parameters and adding additional features.