Project Report: Wall Follower TurtleBot 3 Robot using ROS and PID Control

Introduction:

The goal of this project was to design and simulate a Wall Follower TurtleBot 3 robot using ROS (Robot Operating System) and PID control. The robot is programmed to follow a wall at a constant distance, demonstrating autonomous navigation capabilities.

Objective:

- **Design a Wall Follower TurtleBot 3 robot in a simulated environment.**
- **Implement PID control to maintain a constant distance from the wall.**
- **Evaluate the robot's performance in a simulated scenario.**

Methodology:

1. **Robot Design: The TurtleBot 3 robot was designed and simulated using ROS and Gazebo.**
2. **Sensor Integration: A lidar sensor was integrated into the robot to detect the distance from the wall.**
3. **PID Control: A PID controller was implemented to adjust the robot's velocity and steering angle to maintain a constant distance from the wall.**
4. **Simulation: The robot was simulated in a Gazebo environment with a predefined wall layout.**
5. **I have used the following PID gain values after manual tuning:**

   (Kp = 0.6, Ki = 0 ,Kd = 0.01)

6. **The robot maintains a distance of 1.3 and then takes angular turns to avoid crash.**

Steps:

1. **First we need to install ubuntu 20.04 (focal fossa) and ROS Noetic as the suggested framework to run this project**
2. **I assume there is already a catkin workspace in which we have to create our project**
3. **Within our project folder we create an executable script or node for wall follower turtlebot which will be executed when called(link for the script is available on my github account). Make sure the script is given permissions of both being executable and readable.**

4. **Following are the necessary installations before running the simulation:**

Before installing Turtlebot3, make sure to make the following two commands:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

The installation may fail if you do not upgrade.

Then, do the following (if you install for *noetic*, make `-b noetic-devel` to get the right branch)

```
$ cd ~/catkin_ws/src/
$ git clone
https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git -b
noetic-devel
$ git clone  https://github.com/ROBOTIS-GIT/turtlebot3.git
-b noetic-devel
$ cd ~/catkin_ws && catkin_make
```

If you install on *melodic*, change `-b noetic-devel` with `-b melodic-devel`

This will install the core packages of Turtlebot3.

Afterward, and after the correct compilation of the catkin_ws, you can download and installation the simulation packages

```
$ cd ~/catkin_ws/src/
```

```
$ git clone
https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
```

```
$ cd ~/catkin_ws && catkin_make
```

(As such, the Turtlebot3 simulator should be installed.)

(Then, I made the modification in the .bashrch file as follows):

```
$ cd gedit .bashrc
```

(Inside the bashrc file, put the following aliases to make it easier to access different executables in the alias section).

```
alias burger='export TURTLEBOT3_MODEL=burger'
alias waffle='export TURTLEBOT3_MODEL=waffle'
alias tb3fake='roslaunch turtlebot3_fake
turtlebot3_fake.launch'
alias tb3teleop='roslaunch turtlebot3_teleop
turtlebot3_teleop_key.launch'
alias tb3='roslaunch turtlebot3_gazebo
turtlebot3_empty_world.launch'
alias tb3maze='roslaunch turtlebot3_gazebo
turtlebot3_world.launch'
alias tb3house='roslaunch turtlebot3_gazebo
turtlebot3_house.launch'
```

(also, at the end of the file, write the following commands)

```
source /opt/ros/noetic/setup.bash
source /home/akoubaa/catkin_ws/devel/setup.bash
export TURTLEBOT3_MODEL=waffle
export SVGA_VGPU10=0
```

(The last command will let you open Gazebo on a Virtual Machine and avoid crashing its display).

5. **Now run the following commands in the terminal to make your own project directory:**

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg my_turtlebot_pkg rospy geometry_msgs
sensor_msgs
```

6. **Now move your python script to this directory:**

```
$ mv /path/to/wall_follower.py  ~/catkin_ws/src/my_turtlebot_pkg/src
```

```
$ chmod +x ~/catkin_ws/src/my_turtlebot_pkg/src/wall_follower.py
```

(make sure to rebuild your catkin workspace after this step)

7. **After all the necessary installations run the following commands in order to run the simulation perfectly:**

   ```
   $ export TURTLEBOT3_MODEL=waffle
   $ roslaunch turtlebot3_gazebo turtlebot3_stage_1.launch
   ```

8. **Then run the python script from visual studio code to make the node running in the backend.**

9. **(Optional step) if you want to reset the gazebo simulation to original state then stop running the python script,and pause the simulation from Gazebo and run the following command in the terminal:**

   ```
   $ rosservice call /gazebo/reset_simulation
   ```

   And then you may unpause and run everything again from scratch

Results:

- **The robot successfully followed the wall at a constant distance, demonstrating effective PID control.**
- **The robot adapted to changes in the wall layout and maintained a stable distance.**
- **Simulation results showed a consistent performance, indicating the robustness of the PID control algorithm.**