QUADRUPED CHALLENGE 2025

ROS2-Based Autonomous Navigation System

Team: DeepLearners

Developer: Aman Jaiswal

IIT Bhubaneswar Quadruped Challenge
October 2025

Challenge Overview

Problem Statement & Objectives

- Develop **ROS2-based control and navigation stack** for quadruped robots
- Execute predefined path patterns autonomously in simulated environment
- Implement precise trajectory control for **square and circular paths**
- Demonstrate robust navigation capabilities in Gazebo simulation platform

Primary Objective

Create a robust, autonomous navigation system capable of precise path following for quadruped robotic platforms

Solution Architecture

System Design & Implementation Strategy

- Built on **ROS2 Humble Hawksbill** framework for modern robotics development
- Integrated with **Gazebo Classic 11.10** for high-fidelity simulation
- Python 3.10 implementation for rapid development and maintainability
- Velocity-based control architecture using **/cmd_vel** topic interface

Core Strategy

Time-based open-loop control system with precisely calibrated velocity commands for deterministic path execution

Technology Stack

Components, Tools & Infrastructure

FRAMEWORK	ROS2 Humble
SIMULATION	Gazebo 11.10
LANGUAGE	Python 3.10
PLATFORM	Ubuntu 22.04
ENVIRONMENT	WSL2
ROBOT MODEL	TurtleBot3
CORE LIBRARIES	rclpy, geometry_msgs
DISPLAY SERVER	VcXsrv X11

Square Path Navigation

Algorithm Design & Implementation

```
START EXECUTION
  ↓
Initialize ROS2 Node & Publisher
  ↓
LOOP: FOR each of 4 sides
    → Execute Forward Motion (2m at 0.3 m/s)
    → Stabilization Pause (0.5s)
    → Execute 90° Turn (0.3 rad/s)
    → Stabilization Pause (0.5s)
  ↓
Path Complete → Stop Robot
  ↓
END EXECUTION
```

Linear Velocity: 0.3 m/s

Angular Velocity: 0.3 rad/s

Circular Path Navigation

Algorithm Design & Implementation

```
START EXECUTION
  ↓
Initialize ROS2 Node & Publisher
  ↓
Calculate Angular Velocity
ω = v / r = 0.3 / 1.5 = 0.2 rad/s
  ↓
Execute Synchronized Motion
(Linear + Angular simultaneous)
  ↓
Monitor Progress (Complete 360° rotation)
  ↓
Path Complete → Stop Robot
  ↓
END EXECUTION
```

Code Implementation

Square Path Controller Example

```
def execute_square(self):
    # Execute 4-sided square path
    for side in range(1, 5):
        # Move forward 2 meters
        self.move_forward(2.0, 0.3)
        time.sleep(0.5) # Stabilization

        # Turn 90 degrees
        self.turn(90, 0.3)
        time.sleep(0.5) # Stabilization
```

SQUARE_PATH.PY

76 lines

CIRCULAR_PATH.PY

59 lines

System Architecture

Component Interaction & Data Flow

```
┌────────────────────────────────────────┐
│      Gazebo Simulation Environment     │
│                                        │
│  ┌──────────────────────┐              │
│  │  TurtleBot3 Robot    │ ← /cmd_vel   │
│  │   Physics Engine     │              │
│  └──────────────────────┘              │
└────────────────────────────────────────┘
                         ↑
                   │ Twist Messages
                         │
┌────────────┴───────────────────────────┐
│    ROS2 Navigation Controller          │
│                                        │
│  ┌───────────────┐  ┌────────────────┐ │
│  │ Square Path   │  │ Circular Path  │ │
│  │    Node       │  │     Node       │ │
│  └───────────────┘  └────────────────┘ │
│                                        │
│  rclpy + geometry_msgs Libraries       │
└────────────────────────────────────────┘
```

Square Path Results

Performance Metrics & Execution Analysis

✓ Successfully Completed

DIMENSIONS	EXECUTION TIME	TOTAL TURNS	ACCURACY
2m × 2m	48 sec	4 × 90°	High ✓

Execution Pattern

Robot successfully executed 4 precise straight-line segments with accurate 90° turns at each corner, maintaining consistent velocity throughout the trajectory

Circular Path Results

Performance Metrics & Execution Analysis

✓ Successfully Completed

RADIUS	EXECUTION TIME	ROTATION	SMOOTHNESS
1.5 m	31 sec	360°	Excellent ✓

Execution Pattern

Robot achieved smooth, continuous circular motion through synchronized linear and angular velocity control, completing a perfect 360° trajectory

Technical Challenges

Problems Encountered & Solutions Implemented

WSL2 Gazebo Display Issues

Configured VcXsrv X server with proper DISPLAY environment variables and OpenGL settings for GUI rendering

Robot Spawn Timing

Implemented integrated launch file to ensure proper initialization sequence and node synchronization

Path Accuracy Calibration

Fine-tuned velocity parameters through iterative testing to achieve precise trajectory execution

Time-based Control Stability

Added stabilization wait periods between motion phases to prevent momentum-induced errors

Key Learning

Precise simulation parameter tuning and systematic testing are critical for achieving reliable autonomous navigation performance

THANK YOU

GitHub Repository

github.com/amanraj74/Quadruped-Challenge-IIT-Bhubaneswar

Team Information

Team: DeepLearners

Developer: Aman Jaiswal

Email: aerraj50@gmail.com

Questions?