**Papers Summary**

G.S. Ruparani

1. **ROBOT SURVEILLANCE FOR CONNECTED HOME INTEGRATION**

**Aim**

The project, *"Robot Surveillance for Connected Home Integration,"* focuses on creating an autonomous surveillance robot using TurtleBot3 and ROS2 (Robot Operating System 2) to enhance home security. The main objective is to replace static CCTV systems with a mobile, intelligent surveillance robot that can navigate autonomously, respond to sensor inputs, and provide real-time updates to homeowners.

The bot is designed to:

* Autonomously navigate using sensor inputs.
* Capture and transmit images of suspicious activity.
* Integrate with smart home systems for real-time alerts via Telegram.
* Avoid obstacles and optimize navigation with SLAM (Simultaneous Localization and Mapping).

#### **How the Problem is Solved**

1. **Sensor-Triggered Surveillance:** When a motion sensor or environmental sensor detects an anomaly (e.g., movement, sudden temperature change), the TurtleBot3 robot is activated.
2. **Autonomous Navigation:** The bot moves to the detected location, guided by Navigation2 and SLAM, avoiding obstacles along the way.
3. **Real-Time Communication:** Upon reaching the location, the bot captures an image using its Pi Camera and sends it to the homeowner via Telegram messaging app.
4. **User Interaction:** Homeowners can monitor the bot through a Graphical User Interface (GUI) and control it remotely if needed.

#### **Technologies Used**

* Hardware:
  + TurtleBot3 (with motors and sensors)
  + Raspberry Pi 3B+ (computational unit)
  + Pi Camera Module (image capturing)
  + DHT22 sensor (temperature & humidity)
  + MPU6050 (motion tracking)
  + LDR Module (light intensity detection)
* Software & Frameworks:
  + ROS2 (Robot Operating System 2) – Main framework for robotics communication
  + Navigation2 – Path planning & obstacle avoidance
  + SLAM (Simultaneous Localization and Mapping) – Mapping indoor environments
  + PyQt5 – For developing a user-friendly GUI
  + Telegram API – Real-time alerts & communication

#### **How it Works**

1. **Setup & Integration:** Sensors are strategically placed in different areas of the house and connected to Raspberry Pi units. The TurtleBot3 robot is configured with navigation, mapping, and real-time monitoring capabilities.
2. **Trigger & Movement:** When a sensor detects movement, the bot calculates the best route and moves toward the triggered area while avoiding obstacles.
3. **Image Capture & Alerts:** The bot captures an image of the scene and sends it to the homeowner via Telegram notifications.
4. **User Interaction & Control:** Users can view live updates, images, and sensor data on a GUI or Telegram app and manually override the bot if necessary.
5. **Performance Evaluation:** The system is tested for navigation efficiency, response time, and image accuracy in real-world scenarios.

### **Conclusion**

This project provides a cost-effective, smart, and autonomous alternative to traditional home surveillance. By integrating robotics, real-time sensors, and smart home connectivity, the system enhances security and offers homeowners greater control over their home monitoring. It serves as a foundation for future advancements in smart home security, potentially expanding to commercial applications.

1. **INTEGRATING REAL-TIME OBJECT DETECTION WITH LIDAR DATA FOR ENHANCED ROBOTIC AUTONOMOUS NAVIGATION**

#### **Aim**

This thesis focuses on improving autonomous mobile robot navigation in outdoor environments by integrating YOLO-based object detection with LiDAR data. The research aims to enhance the perception and obstacle avoidance capabilities of robots operating in dynamic, unstructured environments such as urban roads, factories, and outdoor delivery routes.

The main problem addressed is that while LiDAR sensors are excellent for detecting static obstacles, they struggle with identifying and tracking dynamic obstacles such as pedestrians, vehicles, and cyclists. This research integrates AI-driven object recognition (YOLO) with LiDAR sensor data to improve obstacle classification and navigation efficiency in real-world settings.

#### **How the Problem is Solved**

1. **Sensor Fusion Approach:** The robot combines LiDAR readings with YOLO-based computer vision for enhanced obstacle recognition.
2. **AI-Powered Object Detection:** The YOLO algorithm identifies and classifies objects using real-time camera feeds.
3. **Dynamic Costmap Enhancement:** Object detection results are integrated into the ROS2 Nav2 costmap, allowing the robot to make better navigation decisions.
4. **Path Planning & Obstacle Avoidance:** The system continuously updates its navigation path to avoid moving objects based on combined sensor inputs.
5. **Testing in Simulated & Real Environments:** The approach is validated using Webots simulations before deploying on an actual AgileX Scout 2.0 mobile robot.

#### **Technologies Used**

* Hardware:
  + AgileX Scout 2.0 (Outdoor mobile robot)
  + TurtleBot3 (Indoor simulation platform)
  + Intel NUC (Computing unit)
  + RPLiDAR A2 (2D laser scanner)
  + Camera Module (For object detection)
* Software & Frameworks:
  + ROS2 (Robot Operating System 2) – Main robotics middleware
  + Navigation2 (Nav2) – Autonomous navigation stack
  + YOLO (You Only Look Once) – Deep learning-based object detection
  + OpenCV – Image processing
  + Webots – 3D robotics simulation
  + RViz2 – Visualization tool for sensor data

#### **How it Works**

1. **Sensor Data Collection**:
   * LiDAR scans the surroundings and detects obstacles.
   * The camera captures real-time images for object classification.
2. **Object Recognition & Sensor Fusion**:
   * The YOLO model identifies objects (e.g., cars, pedestrians, traffic signs).
   * The detected object’s position is aligned with LiDAR data to improve accuracy.
3. **Navigation & Path Planning**:
   * The Nav2 costmap is updated dynamically using sensor fusion results.
   * The robot adjusts its path based on detected obstacles.
4. **Real-Time Decision Making**:
   * If a pedestrian or moving vehicle is detected, the robot modifies its trajectory to avoid collisions.
   * The system can predict and track the movement of dynamic obstacles.
5. **Testing & Validation**:
   * The system is first tested in Webots simulations.
   * It is then deployed on the AgileX Scout 2.0 for real-world outdoor tests.

### **Conclusion**

This research successfully integrates AI-powered object detection with LiDAR-based navigation to enhance autonomous robots’ perception and decision-making capabilities. By improving obstacle classification and avoidance, this system paves the way for more reliable and safer autonomous mobile robots in urban and industrial environments. Future work could focus on real-time adaptation to fast-moving objects and improving computational efficiency for real-world deployments.

1. **TOWARDS A ROBOTIC INTRUSION PREVENTION SYSTEM: COMBINING SECURITY AND SAFETY IN COGNITIVE SOCIAL ROBOTS**

#### **Aim**

This research focuses on enhancing sAdaptive: afety and cybersecurity in cognitive social robots. The main goal is to develop a Robotic Intrusion Prevention System (RIPS) for ROS2-based robots that can detect and mitigate cyber-physical threats in real time. Unlike traditional Intrusion Prevention Systems (IPS), RIPS incorporates System Modes that dynamically adjust the robot’s functionality based on the level of security threats detected.

The system ensures that cognitive social robots are:

* **Safe & Reliable:** Capable of operating in human-shared spaces without posing risks.
* **Secure Against Cyber Attacks:** Prevent unauthorized access and manipulation of robotic systems.
* **Adaptive:** Adjust robot behaviors dynamically based on intrusion levels.

#### **How the Problem is Solved**

1. **Intrusion Detection & Threat Assessment**
   * The RIPS monitors robotic communications using predefined security rules.
   * It detects anomalies through rule-based and machine-learning approaches.
   * Alert levels are assigned to the system (e.g., DEFAULT → ALERT → COMPROMISED → HALT).
2. **Adaptive System Modes for Mitigation**
   * When a threat is detected, the robot’s functionality is restricted based on the alert level.
   * Example: In ALERT mode, the robot’s camera may be turned off to prevent data leaks.
   * If a severe threat is found (COMPROMISED mode), the robot may halt movement to prevent unsafe behavior.
3. **Experimental Validation in a Real Robot**
   * The system was tested on a Tiago2 robot in a simulated industrial setting.
   * When an intrusion was detected, the robot dynamically recalculated its navigation path to avoid restricted areas.
   * The camera feed was deactivated to prevent potential data theft.

#### **Technologies Used**

* Hardware:
  + Tiago2 Robot (cognitive social robot)
  + Ubuntu 22.04 with ROS2 Humble (for RIPS implementation)
* Software & Frameworks:
  + ROS2 (Robot Operating System 2) – Middleware for robotic control.
  + System Modes Framework – Enables adaptive functionality based on security levels.
  + Snort IDS – Detects network-level threats.
  + YARA – Used for malware detection in robot communications.

#### **How it Works**

1. **Real-Time Monitoring:**
   * The RIPS node continuously analyzes messages and connections within the ROS2 system.
   * It detects unauthorized access, data tampering, or abnormal robotic behavior.
2. **Alert Levels & Response Actions:**
   * DEFAULT: Normal operation.
   * ALERT: The robot avoids high-security areas and disables sensitive functions like camera feeds.
   * COMPROMISED: Critical functions are shut down, and human intervention is required.
   * HALT: The robot is completely disabled for safety.
3. **Experimental Testing:**
   * The Tiago2 robot navigated through a predefined path in a test environment.
   * Upon detecting an intrusion, it restricted its movement and deactivated the camera.
   * The system successfully reacted in real time, ensuring both security and safety.

### **Conclusion**

This research successfully integrates cybersecurity with robotic safety by introducing RIPS for cognitive social robots. The system dynamically adjusts the robot’s behavior in response to cyber threats, making social robots safer and more resilient to attacks. Future developments will focus on enhancing adaptability, real-world deployment, and expanding security policies for broader applications.

1. **INTRUSION COUNTERMEASURE SYSTEM**

#### **Aim**

The Intrusion Countermeasure System is a security solution designed to prevent unauthorized access in restricted areas. It aims to enhance defense mechanisms, particularly for border security, by using automation, machine learning, and real-time surveillance. The project is motivated by the high casualty rates among soldiers in extreme environments like the Siachen Glacier, emphasizing the need for advanced, autonomous security systems.

#### **How It Is Solved**

The system integrates machine learning-based human detection, motion tracking, and automated navigation to monitor and counter intrusions. It operates in the following steps:

1. **Detection:** A camera sensor continuously monitors the environment, using OpenCV and machine learning models to detect intruders.
2. **Alert System:** Upon detecting an intruder, the system sends real-time alerts via Telegram bots and transmits the intruder’s coordinates to security personnel.
3. **Tracking & Aiming:** A servo motor-driven aiming system aligns with the intruder’s position for counteraction.
4. **Navigation:** A ROS2-based robotic bot autonomously patrols and maps the secured area, ensuring comprehensive surveillance.
5. **Backup Communication:** A Bluetooth-based navigation system serves as a redundancy mechanism in case of network failures.

#### **Technologies Used**

* Hardware:
  + Arduino Uno (for motor and sensor control)
  + Camera sensors (for face detection and tracking)
  + Servo motors (for automated aiming and movement)
  + DC motors with IBT2 drivers (for navigation)
* Software:
  + ROS2 (Robot Operating System 2): Enables autonomous navigation with SLAM (Simultaneous Localization and Mapping).
  + Machine Learning & Computer Vision:
    - OpenCV: Face detection and motion tracking
    - Caffe DNN model: Face recognition (trained using the WIDER Face dataset)
    - TensorFlow: Neural network-based data processing
  + Python & NumPy: Data processing and integration
  + Telegram API: Real-time alert system

#### **How It Works**

1. **Face Detection**:
   * The camera sensor captures live video feeds.
   * Machine learning models identify human presence using pretrained facial recognition models.
   * If a face is detected, an alert is sent via Telegram with image and location data.
2. **Autonomous Navigation & Surveillance**:
   * The ROS2-based bot continuously patrols using SLAM.
   * The navigation system ensures movement across different terrains.
   * If needed, the bot can switch to Bluetooth-based backup navigation.
3. **Threat Response**:
   * The system calculates intruder coordinates and adjusts servo motors for aiming.
   * Security personnel receive alerts and can take action accordingly.

#### **Performance & Limitations**

* **Performance**:  
  1. **94% accuracy** in face detection.
  2. **30 FPS processing speed**, ensuring real-time monitoring.
  3. **Robustness** tested across lighting conditions, occlusions, and different demographic features.
* **Limitations**:  
  1. Cannot distinguish between friend and foe autonomously.
  2. Accuracy decreases when detecting targets beyond camera range.
  3. Vulnerable to destruction by explosives, necessitating multiple units for redundancy.

### **Conclusion**

The Intrusion Countermeasure System provides a cutting-edge security solution, enhancing border defense and surveillance automation. Its machine learning-based detection, real-time alerts, and autonomous navigation make it a valuable tool for military and critical infrastructure security. While challenges remain, its potential applications extend to industrial security, nuclear facilities, and other high-risk zones.

1. **A DEVELOPMENT OF MOBILE ROBOT BASED ON ROS2 FOR NAVIGATION APPLICATION**

#### **Aim**

The paper presents the development of an autonomous navigation mobile robot using Robot Operating System 2 (ROS2) with low-cost embedded hardware. It aims to improve reliability, safety, and real-time performance over ROS1 by leveraging Data Distribution Service (DDS) for communication. The project uses Cartographer for SLAM (Simultaneous Localization and Mapping) and Navigation2 for path planning and dynamic obstacle avoidance.

The objective is to create an affordable, ROS2-based differential-drive robot that can perform:

1. **Mapping** using 2D LIDAR
2. **Localization** with AMCL (Adaptive Monte Carlo Localization)
3. **Navigation** using A\* path planning and DWB local planner
4. **Real-time microcontroller integration** using **Micro-ROS**

#### **How the Problem is Solved**

The robot was built using a layered mechanical design with three sections:

1. Top Plate – Supports the LIDAR sensor
2. Mid Plate – Houses the Raspberry Pi 4 (Main processor)
3. Bottom Plate – Contains the motor driver board and microcontroller (Raspberry Pi Pico)

The hardware integration ensures smooth operation:

* Raspberry Pi 4 (4GB RAM): Runs ROS2 and handles SLAM, localization, and navigation.
* Raspberry Pi Pico (RP2040): Controls motor speed and processes encoder feedback.
* YD Lidar X4: Provides 360-degree laser scanning for mapping and navigation.
* Motor with quadrature encoder: Enables precise odometry calculations.
* 3S LiPo battery: Powers the robot.

To improve communication reliability, Micro-ROS replaces traditional ROS serial communication by utilizing micro-XRCE-DDS (a DDS middleware for embedded systems).

#### **Technologies Used**

1. ROS2 (Foxy Fitzroy) – Robotics framework running on Ubuntu 20.04
2. Micro-ROS – Allows microcontrollers to use ROS2
3. Cartographer – For real-time SLAM (mapping and localization)
4. Navigation2 – Provides path planning, obstacle avoidance, and goal tracking
5. *A Algorithm*\* – Used for global path planning
6. DWB (Dynamic Window Approach) Planner – Used for local path planning
7. LIDAR (YD Lidar X4) – Captures 2D environment data for navigation
8. Raspberry Pi 4 – Main computing unit
9. Raspberry Pi Pico (RP2040) – Controls motor speed and handles odometry

#### **How It Works**

1. **Mapping Phase**:
   * The robot is controlled via teleoperation from a PC.
   * It explores the environment, collecting LIDAR and odometry data.
   * Cartographer processes this data to create a 2D occupancy grid map.
2. **Navigation Phase**:
   * The global planner (A\*) calculates a path to the goal using the generated map.
   * The local planner (DWB) dynamically adjusts movement to avoid obstacles.
   * The robot moves towards its goal while continuously replanning in case of obstacles.
   * The system was tested using Rviz2 (3D visualization tool for ROS2).
3. **Performance Evaluation**:
   * The robot successfully reached its goal without collision in a static map.
   * In dynamic obstacle environments, the local planner adjusted paths efficiently.
   * The average position error at the goal was 6.96 cm in static environments and 4.39 cm in dynamic environments.
   * Time to reach the goal increased when obstacles were present, but the robot successfully avoided them.

### **Conclusion**

The study successfully developed a low-cost autonomous navigation robot using ROS2 and Micro-ROS. The robot can perform SLAM, localization, and autonomous navigation while handling dynamic obstacles. Future work will focus on improving localization accuracy and testing in larger environments.

1. **LIDAR-BASED DYNAMIC PATH PLANNING OF A MOBILE ROBOT ADOPTING A COSTMAP LAYER APPROACH IN ROS2**

#### **Aim**

This thesis explores LiDAR-based dynamic path planning using a costmap layer approach in ROS2. The goal is to improve indoor autonomous navigation by enabling robots to handle dynamic obstacles more effectively. The research is motivated by challenges in motion planning, particularly in autonomous mail delivery robots at the LINKS Foundation. The project aims to enhance the ROS2 Navigation Stack (Nav2) by integrating a dynamic obstacle handling mechanism.

#### **How It Is Solved**

1. **Research & Development**:
   * Investigates the state of the art in autonomous navigation and ROS2 functionalities.
   * Develops a new costmap layer plugin to improve obstacle detection and navigation.
2. **Implementation**:
   * Uses LiDAR-based object detection for real-time obstacle tracking.
   * Applies Kalman filters and Hungarian algorithms for tracking and velocity estimation.
   * Modifies Nav2’s costmap structure to integrate dynamic obstacles.
   * Conducts simulations in Webots and Rviz to validate the approach.
3. **Performance Evaluation**:
   * Compares results with existing Nav2 solutions.
   * Identifies critical areas for further optimization.

#### **Technologies Used**

* **Hardware**:  
  + TurtleBot3 Burger (used for testing)
  + LiDAR Sensors:
    - LDS-01 (default sensor)
    - RPLIDAR A3 (enhanced sensor for better accuracy)
* **Software**:  
  + ROS2 Navigation Stack (Nav2):
    - Behavior Tree Navigator (controls navigation flow)
    - Planner, Controller, and Recovery Servers (modular navigation components)
    - Costmap2D (environmental representation)
  + Machine Learning & Algorithms:
    - Kalman Filters (velocity estimation)
    - Hungarian Algorithm (object tracking)
    - Dynamic Window Approach (DWA) (motion planning)
    - Timed Elastic Band (TEB) Controller (trajectory optimization)
  + Simulation Tools:
    - Webots (simulation environment)
    - Rviz (visualization tool for ROS)

#### **How It Works**

1. **Obstacle Detection**:
   * Uses LiDAR data to detect and separate static and dynamic obstacles.
   * Applies image processing algorithms and running average filters for object detection.
2. **Obstacle Tracking**:
   * Tracks detected obstacles over time.
   * Uses a Hungarian algorithm to assign objects to tracked entities.
   * Estimates velocity using a Kalman filter.
3. **Navigation & Cost Assignment**:
   * Updates ROS2 costmap layers dynamically based on obstacle movement.
   * Implements a Gaussian-based cost assignment around moving obstacles.
   * Optimizes motion planning algorithms for efficient path adjustments.
4. **Simulation & Testing**:
   * Conducts simulation experiments in Webots and Rviz.
   * Evaluates performance improvements in handling dynamic obstacles.
   * Benchmarks existing Nav2 controllers (DWB & TEB) against the new approach.

### **Conclusion**

This thesis enhances ROS2’s navigation capabilities by introducing a dynamic obstacle-aware path planning approach using LiDAR and costmap layers. The system improves autonomous navigation in dynamic indoor environments, such as office mail delivery robots. While challenges remain, it provides a scalable and modular solution for future robotics applications.

1. **AN OPEN-SOURCE MULTI-ROBOT FRAMEWORK SYSTEM FOR COLLABORATIVE ENVIRONMENTS BASED ON ROS2**

#### **Aim**

The paper presents a framework for coordinating multiple robots in a collaborative environment using ROS2. The goal is to improve interoperability, autonomous navigation, computer vision, and task allocation for robots working together in industrial and service applications. The framework enables effective communication between robots and sensors in environments like warehouses, restaurants, and industrial settings.

#### **Problem & Solution**

The challenge in multi-robot systems (MRS) lies in efficient communication, task allocation, localization, and coordination. The proposed solution is a centralized architecture that integrates:

1. **Autonomous Navigation Module** – Ensures safe and efficient movement.
2. **Computer Vision Module** – Uses **ArUco markers** for localization.
3. **Task Controller Module** – Assigns missions dynamically based on robot proximity and availability.

#### **Technologies Used**

* ROS2 (Robot Operating System 2) – Core framework.
* Gazebo – Simulation tool for testing.
* ArUco Markers & OpenCV – For vision-based localization.
* LiDAR & SLAM (Simultaneous Localization and Mapping) – For real-time mapping and navigation.
* Nav2 Package – Handles navigation via behavior trees.
* Intel Realsense Cameras – Used for visual detection.

#### **How It Works**

1. **Multi-Robot Communication:** A central node manages robots and sensors, making high-level decisions.
2. **Localization via Computer Vision:** ArUco markers help in determining robot positions. A filtering system selects the most accurate camera data.
3. **Autonomous Navigation:** Robots use LiDAR, maps, and path planners to reach target destinations while avoiding obstacles.
4. **Task Assignment:** A task manager assigns tasks to the nearest available robot. Tasks include object delivery, waypoint navigation, and workspace interactions.
5. **Simulation & Testing:** The system was tested in industrial warehouses and restaurant simulations, handling 6 to 18 robots with high efficiency.

#### **Conclusion**

The ROS2-based framework provides a scalable, flexible, and robust solution for multi-robot collaboration. It successfully integrates vision-based localization, intelligent task allocation, and autonomous movement. The framework enhances robotic automation across various industries. Future improvements could involve more advanced AI integration and extended real-world testing.

1. **STABILIZATION AND MOTION CONTROL OF TWO-WHEELED SELF-BALANCING MOBILE ROBOT USING ROS2 AND GAZEBO SIMULATOR**

#### **Aim**

This research focuses on stabilizing and controlling a Two-Wheeled Self-Balancing Mobile Robot (TWSBMR) using ROS2 and the Gazebo Simulator. The TWSBMR is a nonlinear, underactuated system commonly used in robotics research, Segway transporters, and humanoid robots. The study aims to:

1. Implement and compare two control strategies: PID (Proportional-Integral-Derivative) controller and LQR (Linear Quadratic Regulator).
2. Validate real-time balancing and motion control using the Gazebo Simulator.
3. Explore the advantages of ROS2 over ROS1 in terms of real-time capabilities, security, and scalability.

#### **How It Is Solved**

The project follows these steps:

1. **Mathematical Modeling**:
   * Develops a **state-space model** of the TWSBMR.
   * Defines **dynamic equations** based on Newton-Euler mechanics.
2. **Implementation of Controllers**:
   * Designs **PID and LQR controllers** using MATLAB.
   * Implements these controllers in **ROS2 as separate nodes**.
   * Uses **RQT tools** to visualize control responses.
3. **Simulation & Testing**:
   * Builds the **TWSBMR model in the Gazebo Simulator**.
   * Uses **ROS2 nodes to handle sensors, actuators, and control logic**.
   * **Compares PID and LQR responses** using **real-time plots**.

#### **Technologies Used**

* **Hardware**:
  + Two-wheeled self-balancing robot.
  + Sensors: **IMU (Inertial Measurement Unit)** for balance detection.
  + Motor encoders for odometry.
* **Software**:
  + **ROS2**:
    - **Nodes**: IMU sensor node, motor control node, PID/LQR controller node.
    - **Communication**: Uses **publish-subscribe model**.
  + **Gazebo Simulator**:
    - Simulates real-world physics and sensor data.
    - Uses **SDF (Simulation Description Format)** for robot modeling.
  + **MATLAB**:
    - Designs **state-space models** and **controller tuning**.
    - Computes **LQR gain matrix**.
  + **RQT Tools**:
    - **RQT Graph**: Visualizes **ROS2 nodes & communication**.
    - **RQT Plot**: **Compares PID & LQR responses** in real-time.
    - **RQT Dynamic Reconfigure**: Adjusts **controller parameters** without restarting nodes.

#### **How It Works**

1. **Robot Model & Simulation**:
   * TWSBMR is modeled in Gazebo using SDF format.
   * The IMU sensor detects pitch angle, and motor encoders track movement.
2. **Control System**:
   * **PID Controller**:
     + Uses Ziegler-Nichols method for tuning.
     + Optimized with KP = 1.62, KI = 7, KD = 0.0938.
   * **LQR Controller**:
     + Optimizes a cost function to minimize energy and deviation.
     + Gains are computed using MATLAB.
3. **ROS2 Implementation**:
   * IMU node publishes sensor data.
   * PID/LQR nodes compute control signals.
   * Motor driver node applies commands to wheels.
   * Gazebo visualizes real-time behavior.
4. **Comparison of PID vs LQR**:
   * RQT Plot shows response curves for:
     + Pitch angle stability.
     + Angular velocity.
     + Linear acceleration.
   * LQR outperforms PID in stability and response time.

#### **Performance & Limitations**

* **Performance**:
  1. Successfully **stabilizes TWSBMR in Gazebo**.
  2. **LQR provides smoother control** compared to PID.
  3. **ROS2 tools simplify debugging and parameter tuning**.
* **Limitations**:
  1. **Gazebo simulation differs from real-world physics**.
  2. **LQR requires precise system modeling**, making it less adaptable.
  3. **IMU sensor noise affects accuracy**.

### **Conclusion**

This study demonstrates the effectiveness of ROS2 and Gazebo for self-balancing robot control. By comparing PID and LQR controllers, it highlights LQR’s superior stability. The ROS2 framework, combined with real-time visualization tools, makes robotic control development more scalable and efficient. Future work includes testing on a physical robot and improving sensor noise handling.

1. **DIFFERENTIAL-DRIVE MOBILE ROBOT CONTROLLER WITH ROS2 SUPPORT**

#### **Aim**

The paper presents the development of a differential-drive mobile robot controller with ROS 2 (Robot Operating System 2) support, using micro-ROS for embedded systems. The goal is to improve the hardware and software capabilities of autonomous mobile robots (AMRs), which are used for internal logistics in industries. The new controller is an evolution of a previous version used for over 10 years at the CIII (UTN-FRC) research center. It aims to enhance flexibility, energy efficiency, and compatibility with modern robotic applications while integrating ROS 2 for autonomous navigation.

#### **How the Problem is Solved**

The study focuses on hardware development, selecting a microcontroller with native support for micro-ROS. The controller includes:

* A power-efficient design using a switching power supply
* USB communication for PC interfacing
* Battery voltage sensing for power monitoring
* A debugging port (JTAG) for development and testing
* PCB (Printed Circuit Board) design optimized for compatibility with existing systems

The software validation involved creating an HTTP-based test application to verify correct component operation. Additionally, a micro-ROS environment was tested, ensuring the controller could communicate with a ROS 2-based system.

#### **Technologies Used**

1. Robot Operating System (ROS 2) – A middleware for robotic applications
2. Micro-ROS – A lightweight ROS 2 implementation for embedded systems
3. ESP32-WROOM-32E – A microcontroller with WiFi and Bluetooth capabilities
4. PID Controllers – For motor speed regulation
5. SLAM (Simultaneous Localization and Mapping) – For autonomous navigation
6. HTTP Server – For debugging and testing via a web interface
7. JTAG Debugging – Using OpenOCD for real-time debugging
8. KiCAD – For designing the PCB layout

#### **How It Works**

* The robot controller receives velocity commands from an onboard computer.
* It reads sensor data from optical encoders and battery voltage monitors.
* Using PID control, it adjusts motor speeds for smooth motion.
* The micro-ROS framework ensures seamless integration with ROS 2-based robotic applications.
* A WiFi-based web interface enables remote debugging and testing.
* The controller’s JTAG port allows firmware debugging and execution monitoring.

### **Conclusion**

The newly developed ROS 2-compatible controller improves on previous designs by incorporating modern embedded systems, efficient power management, and seamless integration with ROS 2. Future work will focus on further software development, optimizing motion control algorithms, and testing the controller in real-world robotic applications.

1. **DYNAMIC PATH PLANNING OF A MOBILE ROBOT ADOPTING A COSTMAP LAYER APPROACH IN ROS2**

#### **Aim**

This paper presents a Dynamic Obstacle Layer (DOL) approach to enhance ROS2 Navigation (Nav2) for mobile robots in dynamic environments. The aim is to improve autonomous navigation and obstacle avoidance by integrating dynamic obstacle information into the costmap layer using LiDAR sensors. The study evaluates the DOL approach through simulations in Webots with a TurtleBot3 robot equipped with RPLIDAR A3.

#### **Problem & Solution**

The challenge in ROS2’s default navigation system (Nav2) is its inability to handle dynamic obstacles efficiently. Current methods create overly cautious routes, reducing efficiency. The proposed DOL approach:

1. **Detects and tracks dynamic obstacles** using LiDAR-based costmaps.
2. **Predicts obstacle motion** using **Kalman filters**.
3. **Assigns Gaussian-based cost values** to dynamically adjust paths.
4. **Integrates seamlessly into Nav2**, improving navigation safety and efficiency.

#### **Technologies Used**

* ROS2 (Foxy) – Core framework.
* Nav2 (Navigation Stack) – Provides path planning and obstacle avoidance.
* Webots – Simulation environment.
* LiDAR (RPLIDAR A3) – Senses obstacles.
* Kalman Filters – Tracks moving objects.
* Hungarian Algorithm – Assigns detected objects to tracks.
* OpenCV – Processes LiDAR costmaps.

#### **How It Works**

1. **Object Detection:** Costmaps are processed to identify moving obstacles using image processing and averaging filters.
2. **Object Tracking:** Detected obstacles are tracked and their velocities estimated via a Kalman filter.
3. **Cost Assignment:** The Gaussian function inflates the costmap values around moving obstacles, assigning higher costs based on speed and direction to guide safer navigation.
4. **ROS2 Integration:** The DOL integrates into Nav2’s local costmap, enhancing real-time path planning without modifying the core navigation stack.

#### **Simulation & Testing**

* The framework was **tested in Webots** with a **TurtleBot3**.
* **Dynamic obstacles** were simulated as moving boxes.
* The **DOL approach reduced collisions** and **improved navigation efficiency** compared to standard **DWB Controller** (ROS2’s default local planner).
* **Two speed tests (0.6 m/s and 0.8 m/s) were conducted**:
  + **At 0.6 m/s:** DOL **reduced collisions from 18% to 4%**.
  + **At 0.8 m/s:** DOL **reduced collisions from 56.7% to 13.3%**, showing **improved obstacle avoidance** even at higher speeds.

#### **Conclusion**

The DOL approach enhances ROS2 navigation by integrating real-time dynamic obstacle tracking, making path planning safer and more efficient. The modular plug-and-play design allows easy adoption in industrial automation, surveillance, and logistics applications. Future improvements include real-world validation, multi-sensor fusion (e.g., cameras), and optimized computational performance.

1. **xx**