A 5G-ENABLED ROVER CONTROL SYSTEM USING EVALUATION BOARD FOR REMOTE TELEOPERATION AND LIDAR-BASED MAPPING

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

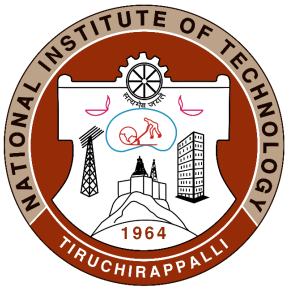
in

COMMUNICATION SYSTEMS

By

VIVEK MUKUNDAN

(208124033)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPPALLI – 620015

DECEMBER 2025

BONAFIDE CERTIFICATE

This is to certify that the thesis entitled **"A 5G-ENABLED ROVER CONTROL SYSTEM USING EVALUATION BOARD FOR REMOTE TELEOPERATION AND LIDAR-BASED MAPPING"** is a bonafide record of the work carried out by

VIVEK MUKUNDAN
(208124033)

in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Communication Systems** of the **National Institute of Technology Tiruchirappalli**, during the academic year 2025.

Dr. E. S. GOPI	**Dr. R. Pandeeswari**
Professor	Professor
ECE Department	ECE Department
(Project Guide)	*(Head of the Department)*

Project Viva-voce held on _____

Internal Examiner **External Examiner**

Contents

ABSTRACT

The rapid evolution of fifth-generation (5G) wireless networks has enabled a new class of ultra-reliable and low-latency applications, particularly in the domains of remote sensing, teleoperation, and autonomous systems. One such application is the remote control of mobile robotic platforms for tasks such as surveillance, disaster response, and crime scene investigation, where real-time responsiveness and situational awareness are critical. This thesis presents the design, implementation, and evaluation of a 5G-enabled rover control system using an Evaluation Board as a mobile gateway, integrating real-time teleoperation, video streaming, and LiDAR-based environmental mapping.

The private 5G network serves as the primary communication medium for bidirectional control commands and high-throughput sensor data transmission between the rover and the remote operator. The proposed system employs a private 5G Standalone (SA) network architecture to ensure predictable latency and high reliability. A distributed software framework is developed wherein control commands from a remote operator are transmitted over the 5G network to the rover, while sensor data, including live video and LiDAR measurements, are streamed back to the operator through an edge-assisted relay mechanism. To address computational constraints on the rover, intensive LiDAR processing tasks such as packet reconstruction, scan generation, and Simultaneous Localization and Mapping (SLAM) are offloaded to the 5G MEC Server via the Evaluation Board (node).

Multiple communication pipelines are implemented and validated, including a low-latency command relay for rover and camera control, a video proxy for real-time visual feedback, and a serial-over-TCP mechanism for high-rate LiDAR data transmission. The system successfully reconstructs 360-degree LiDAR scans under network fragmentation conditions and generates occupancy grid maps and point cloud representations of indoor environments. Experimental results demonstrate stable bidirectional communication with acceptable end-to-end latency for teleoperated exploration tasks.

The outcomes of this work indicate that private 5G networks are well suited for real time robotic teleoperation and sensing applications.

Keywords: 5G Standalone, Evaluation Board, Teleoperation, Simultaneous Localization and Mapping, Mobile Edge Computing, TCP, LiDAR, Rover Control

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my project guide, **Dr. E. S. Gopi**, Professor, Department of Electronics and Communication Engineering, National Institute of Technology Tiruchirappalli, for his continuous guidance, encouragement, and technical insights throughout the course of this work. His support was instrumental in shaping both the research direction and the presentation of this thesis.

I also thank the faculty members of the Department of Electronics and Communication Engineering for their valuable feedback during the various review stages of this project. Their suggestions helped improve the technical depth and clarity of the work.

I am grateful to my friends and colleagues for their constructive discussions, motivation, and support during the course of this research.

Finally, I express my heartfelt thanks to my family for their unwavering encouragement, patience, and support throughout my academic journey.

VIVEK MUKUNDAN

208124033

Chapter 1

INTRODUCTION

1.1 Background and Motivation

The increasing demand for real-time remote operations in applications such as disaster response, industrial inspection, and crime scene investigation has driven significant interest in mobile robotic platforms capable of transmitting high-fidelity sensory data while responding instantaneously to operator commands. Traditional wireless technologies, including Wi-Fi and fourth-generation (4G) cellular networks, often fail to meet the stringent latency, reliability, and bandwidth requirements necessary for such mission-critical operations [2].

The emergence of fifth-generation (5G) wireless networks represents a paradigm shift in wireless communication, offering ultra-reliable low-latency communication (URLLC) and enhanced mobile broadband (eMBB). These capabilities make 5G an ideal candidate for supporting real-time teleoperation of robotic systems, where delayed or lost packets can severely degrade operator perception and control accuracy [1].

In this context, the integration of 5G connectivity with mobile robotic platforms enables a new generation of telepresence systems that combine live video, sensor fusion, and responsive control. This project is motivated by the need to experimentally validate such an end-to-end system using a private 5G Standalone network and commercially available evaluation hardware.

1.2 Practical Applications and Use Cases

The proposed LiDAR-based mapping and teleoperation system is designed to address scenarios where direct human intervention is difficult, unsafe, or impractical. One key application is in high-altitude construction environments, such as bridge construction sites or mountainous regions, where terrain instability and safety risks limit manual surveying. A remotely operated rover equipped with LiDAR can traverse such environments and generate accurate spatial maps without exposing human operators to danger.

Another important use case is in disaster-prone areas, including post-earthquake or landslide zones, where debris, unstable structures, and poor visibility make human access hazardous. LiDAR-based mapping allows rapid assessment of debris distribution, obstacle locations, and accessible paths, enabling rescue planning and structural evaluation. Since LiDAR operates independently of ambient lighting, it is

particularly effective in smoke-filled or low-visibility conditions.

The system can also be employed in industrial inspection scenarios such as warehouses, tunnels, and underground facilities, where continuous mapping is required for navigation, inventory planning, or safety audits. By leveraging a private 5G network, the rover can transmit high-throughput sensor data and receive control commands reliably over extended distances.

Overall, the proposed architecture demonstrates how remote LiDAR-based mapping, combined with low-latency communication, can be used as a practical tool for environment monitoring, infrastructure assessment, and robotic exploration in environments that are inaccessible or unsafe for humans.

1.3 Problem Statement

Effective remote operation of mobile robotic systems requires a tightly coupled bidirectional communication framework that can support high-bandwidth sensory data transmission and ultra-low-latency control signaling. In applications such as crime scene investigation, hazardous environment monitoring, and disaster response, the operator must rely entirely on remote sensory feedback to make navigation and operational decisions. Any significant delay or loss of information can compromise mission effectiveness and operator safety.

Conventional wireless technologies such as Wi-Fi and 4G LTE exhibit variable latency, jitter, and limited reliability under network congestion. These characteristics are unsuitable for immersive teleoperation systems that demand deterministic performance. Although fifth-generation (5G) wireless networks promise ultra-reliable low-latency communication, there is a need for practical, end-to-end validation of their performance in real-world robotic teleoperation scenarios.

Furthermore, mobile robotic platforms are constrained by onboard computational resources. Tasks such as real-time video streaming, LiDAR packet decoding, scan reconstruction, and mapping can overload embedded processors if executed locally. This necessitates a distributed architecture that leverages edge computing while preserving low-latency control.

1.4 Objectives of the Thesis

The primary objective of this thesis is to design, implement, and evaluate a 5G-enabled rover control system capable of real-time teleoperation and environmental sensing. The specific objectives are as follows:

- To design a bidirectional communication architecture over a private 5G Stan-

dalone network for rover control and sensor data transmission.

- To implement a low-latency command relay mechanism enabling responsive rover and camera control.
- To develop a real-time video streaming and proxy framework suitable for teleoperation.
- To integrate LiDAR sensing and implement reliable serial-over-TCP data forwarding under network fragmentation.
- To offload computationally intensive LiDAR processing and mapping tasks to an edge computing node.
- To evaluate the performance of the system in terms of latency, reliability, and functional correctness.

1.5 Scope of the Work

This work focuses on the design and experimental validation of a prototype system operating within a controlled private 5G laboratory environment. The rover platform is teleoperated by a human operator using keyboard and gesture-based inputs, and autonomous navigation is beyond the scope of this thesis. While virtual reality (VR) interfaces are considered in the system design, the implementation primarily focuses on the communication, control, and sensing pipeline.

1.6 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 presents a detailed review of related work in 5G-enabled IoT systems, teleoperation, and LiDAR-based mapping. Chapter 3 describes the overall system architecture and hardware configuration. Chapter 4 details the software design and implementation of the communication and control framework. Chapter 5 presents experimental results and performance analysis. Finally, Chapter 6 summarizes the contributions of this work and outlines directions for future research.

Chapter 2

REVIEW OF LITERATURE

2.1 IoT and Wireless Communication in Industry 4.0

The Fourth Industrial Revolution, commonly referred to as Industry 4.0, is characterized by the convergence of cyber-physical systems, automation, and data-driven decision-making. At the core of this transformation lies the Internet of Things (IoT), which enables large-scale deployment of interconnected sensors, actuators, and intelligent devices [14]. These systems generate continuous streams of data that must be transmitted, processed, and analyzed in real time to optimize industrial operations, enhance safety, and enable predictive maintenance.

Traditional wireless technologies such as Wi-Fi and Zigbee have been widely used in early IoT deployments. While suitable for localized and low-mobility environments, these technologies face challenges in scalability, coverage, and deterministic performance. Cellular technologies such as 4G LTE extended coverage and mobility support but were primarily optimized for mobile broadband applications rather than mission-critical control systems.

2.2 5G Network Capabilities for Time-Sensitive Applications

Fifth-generation (5G) wireless networks represent a fundamental shift in cellular network design. The International Telecommunication Union (ITU) defines three primary service categories for 5G: enhanced Mobile Broadband (eMBB), massive Machine-Type Communications (mMTC), and Ultra-Reliable Low-Latency Communication (URLLC) [1].

URLLC is particularly relevant to robotic teleoperation and remote sensing applications. It targets end-to-end latencies as low as 1 ms with reliability exceeding 99.999%, enabling real-time closed-loop control. These capabilities make 5G a strong candidate for applications such as autonomous vehicles, remote surgery, and industrial automation, where delayed or lost packets can have severe consequences.

2.3 Private 5G Standalone Networks and Edge Computing

While public 5G networks provide wide-area coverage, their performance can be affected by congestion and shared resources. Private 5G networks offer dedicated spectrum, localized deployment, enhanced security, and predictable performance, making them well-suited for industrial and research environments [3].

A 5G Standalone (SA) architecture employs a native 5G core network independent of legacy 4G infrastructure. This architecture enables full support for URLLC features and seamless integration with edge computing platforms. Mobile Edge Computing (MEC) brings computational resources closer to the data source, reducing end-to-end latency and enabling real-time processing for time-sensitive applications [4].

Several studies have demonstrated that edge-assisted architectures significantly improve responsiveness and scalability in real-time robotic systems by distributing computational load across the network.

2.4 Robotic Teleoperation and Sensor Integration

Robotic teleoperation systems traditionally rely on visual feedback provided by cameras mounted on the robot. However, camera-only systems suffer from limited depth perception and reduced situational awareness. To address these limitations, additional sensors such as LiDAR are increasingly integrated into mobile robots [11].

LiDAR sensors provide accurate range measurements and enable 360-degree environment perception. However, LiDAR data streams are high-rate and require substantial processing for packet decoding, scan reconstruction, and mapping. On-board processors such as Raspberry Pi platforms may struggle to handle these tasks alongside real-time control and video streaming.

2.5 Research Gap

A review of existing literature reveals that while individual components of 5G-enabled robotic systems have been explored, there is limited work on end-to-end experimental validation of a complete teleoperation and sensing pipeline operating entirely within a private 5G Standalone network. In particular, the integration of low-latency control, real-time video streaming, LiDAR data forwarding, and edge-based mapping remains insufficiently addressed.

This thesis aims to bridge this gap by presenting a comprehensive system design and experimental evaluation of a 5G-enabled rover platform that integrates

teleoperation, video feedback, and LiDAR-based environmental mapping using an edge-assisted architecture.

Chapter 3

SYSTEM ARCHITECTURE AND HARDWARE DESIGN

3.1 Overview of the Proposed System

The proposed system is designed as a distributed, edge-assisted robotic teleoperation platform operating over a private 5G Standalone (SA) network. The architecture separates control, sensing, communication, and computation tasks across multiple nodes to achieve low-latency responsiveness while accommodating the computational constraints of the mobile rover platform.

At a high level, the system consists of four principal entities:

- A remote operator station used for control input and visualization.
- A mobile rover equipped with actuators, camera, and LiDAR sensor.
- A 5G evaluation board acting as a mobile gateway.
- A private 5G SA network infrastructure providing ultra-reliable connectivity and acting as edge computing node.

The system supports bidirectional data flow, wherein control commands are transmitted from the operator to the rover, while sensory data including live video and LiDAR measurements are streamed back to the operator via the evaluation board. Figure 3.1 illustrates the overall system architecture of the proposed solution. The rover platform hosts a Raspberry Pi that interfaces directly with the YDLIDAR X2L sensor via a USB serial connection. The Raspberry Pi is responsible for acquiring raw LiDAR byte streams, executing motor control logic, and forwarding sensor data over the network without performing computationally intensive processing.

The evaluation board functions as a communication gateway between the rover and the private 5G network. Its primary role is to relay raw LiDAR data received from the rover to the Mobile Edge Computing (MEC) server via the 5G uplink, and to forward control commands from the operator back to the rover. The evaluation board does not perform LiDAR parsing, SLAM, or mapping operations.

All higher-level processing is carried out on the MEC server. This includes LiDAR frame reconstruction, scan visualization, SLAM-based pose estimation, occupancy map generation, and post-processing tasks. By centralizing computation at the MEC layer, the system leverages the low-latency and high-throughput characteristics of the private 5G network while minimizing onboard processing requirements.

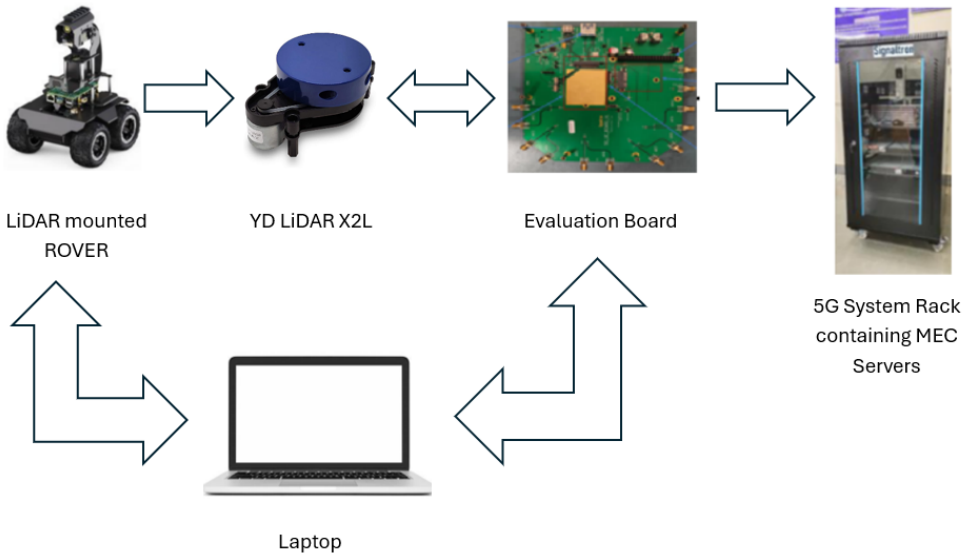


Figure 3.1: Overall system architecture of the proposed 5G-enabled rover control system

The rover platform is built around a Raspberry Pi, which serves as the primary onboard controller. The Raspberry Pi interfaces directly with the YDLIDAR X2L sensor through its USB serial interface, where the LiDAR device is exposed as a character device (e.g., `/dev/ttyUSB0`). The Raspberry Pi is responsible for acquiring raw LiDAR byte streams from the sensor, executing motor control commands, and forwarding sensor data over the network without local processing.

The evaluation board functions as a communication relay and gateway node between the rover and the private 5G system. It does not perform LiDAR parsing or mapping operations. Instead, it receives raw LiDAR data forwarded from the rover and transmits this data to the Mobile Edge Computing (MEC) server through the private 5G network. Similarly, control commands originating from the operator are relayed by the evaluation board to the rover, ensuring reliable bidirectional communication over the 5G link.

The Mobile Edge Computing (MEC) server serves as the primary processing unit in the proposed architecture. All computationally intensive tasks, including LiDAR frame reconstruction, scan visualization, SLAM processing, occupancy map generation, and post-processing operations, are executed on the MEC server. By offloading these tasks to the MEC layer, the system reduces onboard computation requirements and leverages the low-latency, high-throughput characteristics of the private 5G network.

Communication between system components is established using fixed IP addresses within the private 5G network. Raw LiDAR data is transmitted as a continuous byte stream from the rover to the MEC server via the evaluation board,

using TCP-based communication. On the MEC server, the incoming data stream is mapped to a virtual serial interface and processed using the YDLIDAR SDK and custom parsing scripts.

LiDAR data is logged on the MEC server in CSV format, with each entry containing timestamps, scan indices, angular positions, and distance measurements. This data format supports both real-time visualization and offline analysis, including dataset generation for machine learning and schematic extraction.

Overall, the modular separation of sensing, communication, and processing allows each block of the system to perform a well-defined role, improving scalability, debugging, and future extensibility.

3.2 Functional Decomposition

To ensure modularity and scalability, the overall system is decomposed into the following functional layers:

1. **Control Layer**: Responsible for capturing operator input and generating rover motion and camera control commands.

2. **Communication Layer**: Handles reliable and low-latency data transmission over TCP/IP sockets across the 5G network.

3. **Edge Processing Layer**: Performs computationally intensive tasks such as LiDAR packet reconstruction, scan generation, SLAM, and mapping on the Mobile Edge Computing (MEC) server within the private 5G network.

4. **Actuation and Sensing Layer**: Interfaces with the rover's motors, camera, and LiDAR sensor.

This separation of concerns simplifies debugging, improves system robustness, and enables independent evolution of each layer.

3.3 Hardware Components

3.3.1 Mobile Rover Platform

The rover serves as the physical exploration unit and is responsible for motion execution and raw sensor data acquisition. It is equipped with differential drive motors for locomotion, a camera module for visual feedback, and a two-dimensional LiDAR sensor for range sensing.

A Raspberry Pi single-board computer is mounted on the rover to manage motor control, camera interfacing, and network communication. Due to limited processing resources, the rover performs only lightweight tasks and forwards high-rate sensor data without interpretation.

3.3.2 LiDAR Sensor

A rotating two-dimensional LiDAR sensor is mounted on the rover to provide 360-degree range measurements of the surrounding environment. The LiDAR outputs distance measurements as a continuous serial byte stream with fixed packet headers. These raw measurements are essential for environment mapping and spatial understanding but require careful handling due to packet fragmentation when transmitted over a network.

3.3.3 5G Evaluation Board

The 5G evaluation board functions strictly as a communication relay and gateway between the rover and the private 5G network. It forwards raw LiDAR data and video streams from the rover to the MEC server over the 5G uplink and relays control commands from the operator back to the rover. The evaluation board does not perform LiDAR parsing, SLAM, or mapping operations.

3.3.4 Remote Operator Station

The operator station consists of a standard laptop or workstation connected to the private 5G network. It provides the user interface for teleoperation, including keyboard or gesture-based control input and visualization of live video and mapping outputs. The operator station runs client-side software to transmit control commands and receive streamed sensory data.

3.4 Network Architecture

The system operates within a laboratory-scale private 5G Standalone network. The network comprises a 5G core, gNodeB (gNB), and associated switching and routing infrastructure. Unlike public networks, the private deployment offers dedicated bandwidth, controlled latency, and enhanced security.

All long-haul communication between the operator station and the evaluation board traverses the 5G network. Local communication between the evaluation board and the rover is established via a Wi-Fi or Ethernet link, forming a hybrid communication topology optimized for performance and reliability.

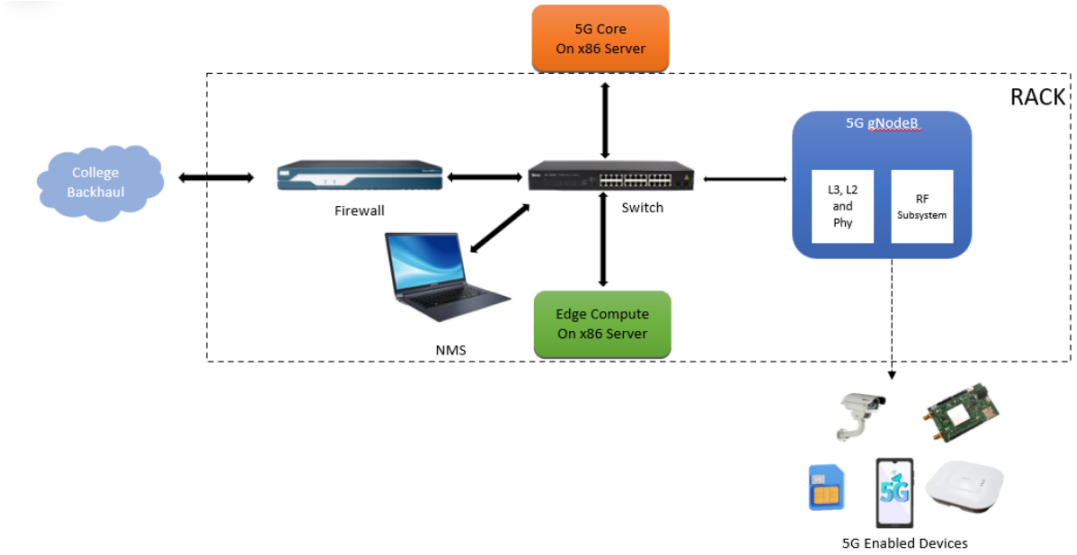


Figure 3.2: Private 5G Standalone network architecture used in the experiments

Figure 3.2 depicts the private 5G Standalone network architecture employed in this work. The network comprises a 5G core, a gNodeB, and a 5G user equipment integrated within the evaluation board. The use of a private network ensures predictable latency, dedicated bandwidth, and enhanced security for real-time robotic teleoperation.

In the proposed system, the private 5G Standalone network is utilized as the primary communication backbone between the rover platform and the operator station. Control commands issued by the operator are transmitted to the rover via the 5G uplink and downlink, while sensor data including live video streams and LiDAR measurements are sent in the reverse direction. Compared to conventional wireless technologies such as Wi-Fi, the private 5G setup provides improved reliability, predictable latency, and dedicated network resources, which are critical for real-time robotic teleoperation.

3.5 Data Flow and Communication Model

The communication model is designed around persistent TCP connections to ensure reliable data delivery. Three primary data flows exist in the system:

1. **Control Command Flow**: Operator-generated commands are transmitted to the evaluation board and relayed to the rover with minimal processing delay.

2. **Video Streaming Flow**: The rover streams camera frames to the evaluation board, which acts as a proxy and forwards the stream to the operator.

3. **LiDAR Data Flow**: Raw serial LiDAR bytes are forwarded from the rover to the evaluation board using a serial-over-TCP mechanism.

By isolating these flows into independent sockets and threads, the system ensures that high-bandwidth sensor data does not interfere with low-latency control commands. Figure 3.3 illustrates the bidirectional flow of control and sensor data

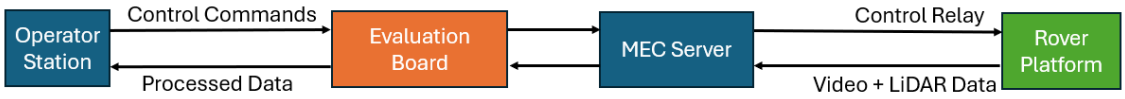


Figure 3.3: Bidirectional data flow between operator, rover, evaluation board, and MEC server

within the system. Control commands originate at the operator station, traverse the private 5G network, and are relayed via the evaluation board to the rover. Sensor data, including video streams and raw LiDAR measurements, follow the reverse path, being forwarded from the rover to the evaluation board and subsequently to the MEC server for processing before visualization at the operator station.

3.6 Design Rationale

The architectural choices made in this system are driven by three primary considerations: latency minimization, computational efficiency, and robustness [14]. Offloading heavy processing to the MEC Server reduces the computational burden on the rover and allows more sophisticated algorithms to be employed. The use of a private 5G SA network ensures predictable communication performance, while modular software components enhance maintainability and extensibility.

This architecture provides a flexible foundation for future enhancements, including immersive virtual reality interfaces, autonomous navigation algorithms, and multi-robot coordination.

Chapter 4

SOFTWARE ARCHITECTURE AND IMPLEMENTATION

4.1 Overview

The software framework of the proposed system is designed to support real-time bidirectional communication, concurrent sensor data processing, and reliable rover control under strict latency constraints. A modular, multi-threaded architecture is adopted to decouple control logic, video streaming, and LiDAR data handling, thereby preventing resource contention and ensuring system responsiveness.

All major software components are implemented in Python due to its extensive networking libraries, rapid prototyping capabilities, and compatibility with embedded Linux platforms. The software stack is distributed across the rover, the 5G evaluation board, and the remote operator station. All communication between the operator, evaluation board, and rover is implemented using persistent TCP sockets to ensure reliable and ordered data delivery [5, 6].

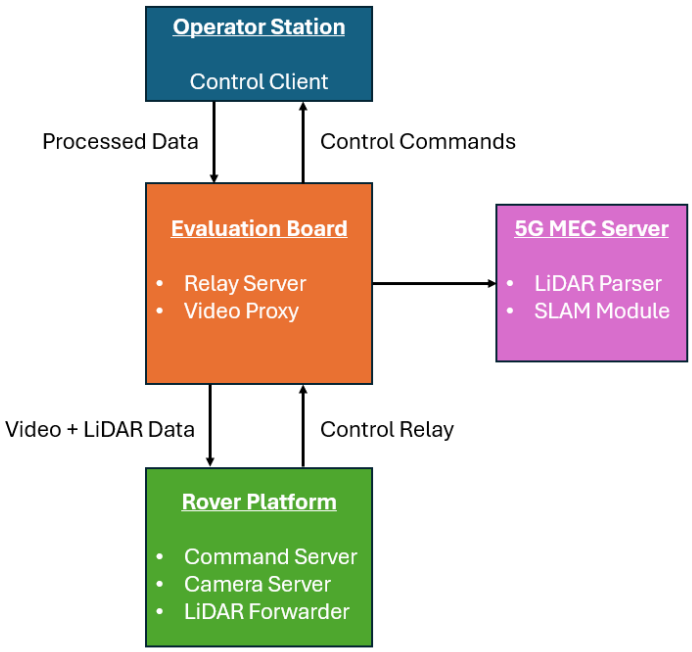


Figure 4.1: Software architecture of the proposed 5G-enabled rover system

Figure 4.1 presents the software architecture of the proposed system. The operator station hosts the control client, while The evaluation board runs relay and proxy services that forward control commands and sensor data between the rover and the MEC server. All LiDAR parsing, SLAM, and mapping modules are executed on the MEC server. The rover platform executes command handling, camera streaming,

and LiDAR data forwarding. This layered architecture enables efficient edge-assisted processing and low-latency teleoperation.

4.2 End-to-End Execution Flow of the System

This section describes the complete step-by-step execution flow of the proposed LiDAR-based mapping and teleoperation system, starting from hardware initialization and concluding with validation and iterative refinement. The description closely follows the actual experimental procedure adopted during implementation and testing.

Step 1: Hardware Power-Up. The execution begins by powering up both the rover platform and the evaluation board. The YDLIDAR X2L sensor is securely mounted on the rover, and its USB cable is connected to the rover's USB serial interface. Upon connection, the LiDAR device is exposed as a serial port on the rover, typically enumerated as `/dev/ttyUSB0` or `/dev/ttyACM0`. Proper device enumeration is verified before proceeding.

Step 2: Network Setup. The rover's onboard network interface is initialized by enabling its Wi-Fi hotspot or local network connection. Network connectivity between the rover and the evaluation board is established over the available communication medium, which may be either private 5G or Wi-Fi depending on the deployment scenario. Additionally, the operator's laptop is verified to have reliable connectivity with the evaluation board to support teleoperation command transmission.

Step 3: Serial Forwarder Initialization on the Rover. On the rover, the script `rover_serial_forwarder_client.py` is launched. This program opens the LiDAR serial device and establishes a TCP connection to a designated port on the evaluation board. The script continuously reads raw bytes from the LiDAR sensor and forwards them over the network using `sock.sendall()`, without performing any local parsing or processing.

Step 4: Virtual Serial Device Creation on the Evaluation Board. On the evaluation board, the `socat` utility is started to listen on the specified TCP port (for example, port 25001). Incoming TCP connections are mapped to a pseudo-terminal (PTY) device such as `/dev/ttyV0`. This mechanism allows raw LiDAR bytes received over the network to be forwarded unchanged to the Mobile Edge Computing (MEC) server for further processing.

Step 5: LiDAR Software Environment Setup on the MEC Server. A Python virtual environment (for example, `lidar_env`) is activated on the MEC server. The YDLIDAR SDK and custom driver scripts, including `ydlidar_x2.py`, are verified to be available within this environment. Required environment variables, such as `TMPDIR`, are configured to prevent temporary directory or permission-related errors during execution.

Step 6: Validation of the Serial-to-TCP Link. To validate correct transmission of LiDAR data, the virtual serial device exposed on the MEC server is opened using a lightweight reader or standard utilities such as `cat` or `xxd`. The presence of valid LiDAR frame headers (`0xAA 0x55`) in the byte stream confirms that raw LiDAR data forwarded from the rover is reaching the evaluation board intact and without corruption.

Step 7: LiDAR Parser and Driver Execution. The LiDAR parser is started by running `ydlidar_x2.py` or `lidar_realtime_to_csv.py`, configured to read from the virtual serial device `/dev/ttyV0`. The parser employs a rolling buffer mechanism to detect the `0xAA 0x55` frame header, reassemble complete LiDAR frames despite TCP fragmentation, and decode per-angle distance measurements.

Step 8: Scan Assembly Verification. Reconstructed 360-degree LiDAR scans are observed using visualization tools such as a Tkinter-based preview window or a simple plotting utility. The scan rate (typically in the range of 7–8 Hz) and the completeness of angular indices (0–359 degrees) are verified. A test CSV file is generated to confirm correct operation of the CSV logging mechanism.

Step 9: Teleoperation Loop Initialization. The teleoperation loop is initiated by starting the keyboard-based control client on the operator's laptop and the motor control server on the rover. Motion commands are issued via the keyboard, and the rover applies motion smoothing techniques to avoid abrupt movements that could negatively affect LiDAR scan quality.

Step 10: Real-Time Mapping and SLAM Execution. The SLAM pipeline is launched on the MEC Server by executing `ydlidar_SLAM.py`. For each incoming LiDAR scan, polar coordinates are converted to Cartesian coordinates, optional downsampling is applied, and ICP-based scan matching is performed against the previous keyframe using an initial pose estimate. Pose graph nodes and edges are appended incrementally, and periodic optimization is performed to reduce drift and improve global consistency.

Step 11: Data Logging and Dataset Creation. During real-time operation, timestamped LiDAR scans are simultaneously logged to CSV files using `lidar_realtime_to_csv.py`. Each record includes fields such as timestamp, scan index, angular position in degrees, and distance in millimeters. These datasets are used for offline analysis and machine learning dataset generation.

Step 12: Post-Processing and Schematic Extraction. After collecting sufficient LiDAR data, a post-processing pipeline is executed. This includes contrast normalization, thresholding, Canny edge detection, and Hough transform-based line extraction. The output of this pipeline is a schematic representation of the environment, generated in formats such as PNG or SVG.

Step 13: Validation and Iterative Refinement. Finally, the generated

occupancy map and schematic are compared against ground truth images of the environment. Metrics such as centroid shifts and ICP RMSE are inspected, and parameters including voxel size, ICP thresholds, and keyframe spacing are tuned iteratively. The experiment is repeated as necessary to improve mapping accuracy and robustness.

4.3 Software Component Distribution

The software modules are distributed as follows:

- **Operator Station**: Control client, visualization interfaces.
- **Evaluation Board**: Network relay server and video proxy services.
- **MEC Server**: LiDAR parsing, SLAM processing, mapping, and data logging services.
- **Rover**: Command server, motor controller, camera interface, LiDAR byte forwarder.

This distribution minimizes processing overhead on the rover while leveraging the MEC server for edge-assisted computation, with the evaluation board acting solely as a communication gateway.

4.4 Command and Control Communication Framework

4.4.1 Command Message Format

Control commands are transmitted using a lightweight JSON-based protocol over TCP sockets. Each message encapsulates the operator's intent in a structured format containing fields such as timestamp, operation mode, and control parameters.

Typical command fields include:

- `mode`: Specifies rover or camera control mode.
- `status`: Encodes discrete rover motion states.
- `x_angle, y_angle`: Continuous camera gimbal control values.
- `timestamp`: Synchronization and debugging reference.

This structured approach ensures extensibility while maintaining low parsing overhead.

4.4.2 Network Relay Service

A dedicated relay service runs on the evaluation board and acts as a transparent intermediary between the operator station and the rover. The relay listens for incoming TCP connections from the operator and forwards all packets to the rover's command server without interpreting their contents.

To support simultaneous bidirectional communication, the relay service employs a multi-threaded design. For each client connection, separate threads are spawned to handle upstream (operator to rover) and downstream (rover to operator) data flows. This prevents blocking behavior and ensures uninterrupted control signaling.

4.5 Rover Command Server Implementation

The rover hosts a command server responsible for interpreting incoming control messages and actuating the hardware accordingly. Upon accepting a TCP connection, the server continuously reads incoming JSON frames and routes them to the appropriate control logic based on the specified mode.

4.5.1 Rover Motion Control Logic

In rover control mode, discrete command states such as forward motion, turning, and stopping are mapped to predefined motor control parameters. These parameters are passed to a base controller interface that abstracts low-level motor actuation.

To prevent abrupt movements and reduce mechanical stress, a motion smoothing mechanism is implemented. Rather than applying instantaneous changes to motor pulse-width modulation (PWM) values, the controller gradually ramps the current speed toward the target speed using bounded step increments.

4.5.2 Camera Gimbal Control

In camera control mode, the rover server interprets continuous angular values transmitted by the operator. These values are forwarded directly to the camera gimbal controller, enabling smooth pan and tilt motion. A high-priority reset command is also supported, allowing the camera to return to a neutral orientation instantly.

4.6 Video Streaming and Proxy Mechanism

4.6.1 Onboard Video Capture

The rover captures live video using an onboard camera module interfaced through a dedicated camera library. Frames are acquired at a fixed resolution and encoded as JPEG images to balance visual quality and bandwidth usage.

4.6.2 MJPEG Streaming Server

A lightweight web server runs on the rover to serve the video stream in MJPEG format. The stream is exposed via an HTTP endpoint, enabling easy access by downstream clients[7].

4.6.3 Video Proxy on Evaluation Board

To decouple the operator from the rover's local network, the evaluation board hosts a video proxy service. This service retrieves the rover's MJPEG stream and re-serves it to the operator. Automatic reconnection logic is implemented to recover gracefully from temporary network disruptions or rover restarts[8].

4.7 Concurrency and Thread Management

Both the relay server and rover command server are designed around concurrent execution. Separate threads are allocated for network I/O, control processing, and streaming tasks. This ensures that high-bandwidth video transmission does not interfere with time-critical control commands.

Thread synchronization and resource cleanup are carefully managed to avoid socket leaks and race conditions, contributing to the overall robustness of the system.

4.8 Design Considerations

The software architecture prioritizes responsiveness, fault tolerance, and modularity. By isolating communication pipelines and distributing processing tasks across network nodes, the system achieves reliable real-time operation under varying network conditions. This software framework forms the backbone for advanced sensing and mapping capabilities discussed in subsequent chapters.

4.9 LiDAR Data Acquisition and Forwarding

4.9.1 Motivation for Edge-Assisted LiDAR Processing

LiDAR sensors generate high-frequency range measurements that provide precise spatial information about the rover's surroundings. However, decoding LiDAR packets, reconstructing full scans, and performing mapping operations require significant computational resources. Executing these tasks entirely on the rover would overload the onboard processor and adversely affect real-time control and video streaming.

To address this limitation, the proposed system adopts an edge-assisted approach in which the rover performs only raw data acquisition and forwarding, while all computationally intensive LiDAR processing is offloaded to the 5G MEC Server via the Evaluation Board.

4.9.2 Serial-to-TCP Forwarding Mechanism

The LiDAR sensor outputs distance measurements as a continuous serial byte stream over a USB interface. A lightweight forwarding client runs on the rover to bridge this serial data to the evaluation board using a TCP connection. The forwarder reads available bytes from the serial port and transmits them immediately over the network without interpretation or buffering logic.

This design choice minimizes latency and preserves packet ordering. Since the LiDAR packets may be fragmented or concatenated during TCP transmission, the forwarder deliberately avoids attempting packet reconstruction on the rover.

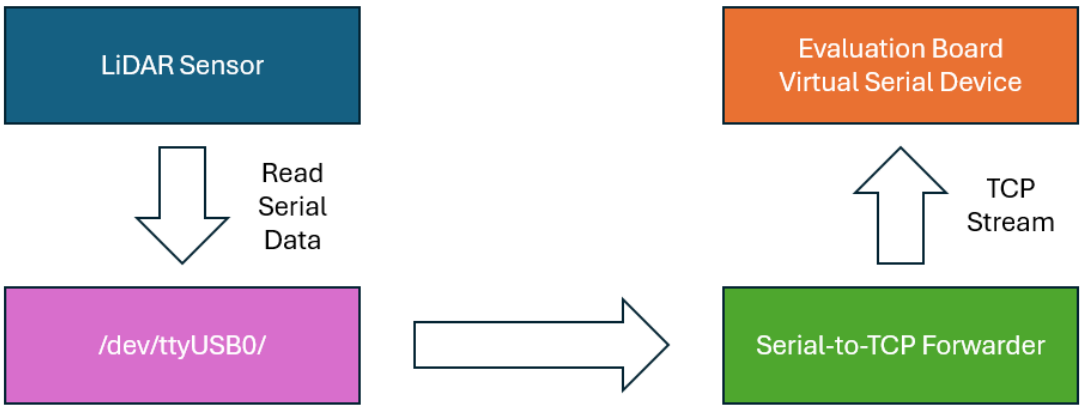


Figure 4.2: Serial-over-TCP forwarding of LiDAR data from rover to evaluation board

```
1 import socket
2 import serial
```

```
3
4  SERIAL_PORT = "/dev/ttyUSB0"
5  BAUD_RATE = 128000
6  TCP_IP = "192.168.50.206"
7  TCP_PORT = 9000
8
9  ser = serial.Serial(SERIAL_PORT, BAUD_RATE)
10 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11 sock.connect((TCP_IP, TCP_PORT))
12
13 while True:
14     data = ser.read(ser.in_waiting or 1)
15     sock.sendall(data)
```

Listing 4.1: Serial-to-TCP forwarding of raw LiDAR data

Listing 4.1 illustrates the lightweight serial-to-TCP forwarding mechanism used in *rover_serial_forwarder_client.py* to transmit raw LiDAR bytes from the rover to the evaluation board without local processing.

4.10 LiDAR Hardware Interface and Data Characteristics

The rover is equipped with a two-dimensional rotating LiDAR sensor that provides 360-degree range measurements of the surrounding environment. From a software perspective, the LiDAR appears as a serial device connected via a USB interface. The sensor continuously transmits distance measurements encoded as raw byte streams at a fixed baud rate, independent of the rover's motion state.

Each measurement corresponds to a specific angular position and represents the distance between the LiDAR and the nearest obstacle along that direction. The high scan rate and dense angular resolution make LiDAR an effective sensor for real-time environment perception. However, the continuous nature of the data stream and the absence of explicit packet boundaries at the transport layer introduce challenges when transmitting the data over packet-switched networks [12].

4.11 LiDAR Packet Structure and Parsing Methodology

The LiDAR sensor transmits data packets framed by fixed synchronization bytes followed by payload length fields. Each packet contains a sequence of range measurements encoded as little-endian values. When transmitted over a TCP connection,

these packets may be fragmented or concatenated due to the stream-oriented nature of TCP.

Figure 4.3: LiDAR packet parsing and scan reconstruction pipeline

Figure 4.3 shows the LiDAR packet parsing and scan reconstruction process. Incoming TCP byte streams are buffered and scanned for valid packet headers. Once detected, packets are decoded and distance measurements are inserted into a scan buffer indexed by angle. A complete scan is generated only after all angular measurements are received.

```
buffer = bytearray()

def parse_stream(data):
    global buffer
    buffer.extend(data)

    while len(buffer) > 5:
        if buffer[0] != 0xAA or buffer[1] != 0x55:
            buffer.pop(0)
            continue

        packet_length = buffer[2]
        if len(buffer) < packet_length:
            break

        packet = buffer[:packet_length]
        buffer = buffer[packet_length:]
        decode_packet(packet)
```

Listing 4.2: Buffered LiDAR packet parsing logic

Listing 4.2 This buffered parsing approach from *lidar_realtime_to_csv.py* ensures reliable packet reconstruction under TCP fragmentation.

To reliably reconstruct valid packets, a buffered parsing approach is implemented on the MEC Server. Incoming bytes are accumulated in a rolling buffer, which is continuously scanned for valid packet headers. Upon detecting a header, the payload length is extracted, and the parser waits until the complete packet is available before decoding its contents. Any extraneous bytes preceding the header are discarded.

Decoded distance values are sequentially inserted into a preallocated scan buffer indexed by angular position. A complete 360-degree scan is declared once all angular indices are populated. This approach ensures robustness against network fragmentation and guarantees scan integrity before further processing [12].

On the evaluation board, a virtual serial interface is created to emulate a local LiDAR device. This is achieved using a socket-to-PTY bridging utility that listens on a designated TCP port and maps incoming connections to a pseudo-terminal device.

As a result, LiDAR parsing software on the MEC Server can access the incoming byte stream as if it were reading directly from a physical serial port. This abstraction enables reuse of existing serial-based LiDAR drivers and simplifies integration.

4.12 LiDAR Packet Reconstruction and Scan Generation

To reliably reconstruct valid packets, a buffered reader mechanism is implemented. The buffered parsing mechanism described in Section 4.10 is used to reconstruct complete LiDAR packets and generate full 360-degree scans.

Each decoded packet contains a sequence of distance measurements corresponding to incremental angular positions. These measurements are inserted into a preallocated scan buffer indexed by angle. A complete scan is formed when measurements for all angular positions are received.

Once a full 360-degree scan is assembled, it is timestamped and passed to downstream processing modules. This modular approach allows scans to be visualized, logged, or processed for mapping independently.

LiDAR measurements are initially represented in polar coordinates. For mapping and visualization, each measurement is converted into Cartesian coordinates using standard trigonometric transformations. These points represent the local spatial layout of obstacles relative to the rover.

To reconstruct a global map of the environment, consecutive LiDAR scans are aligned using a scan-matching approach. The Iterative Closest Point (ICP) algorithm is employed to estimate the relative transformation between successive scans.

Each new scan is aligned with a reference scan or keyframe to estimate the rover's motion. The resulting transformations are accumulated to generate a global point cloud and optimized.

4.13 Fundamentals of SLAM

Simultaneous Localization and Mapping (SLAM) refers to the problem of constructing a map of an unknown environment while simultaneously estimating the pose of the robot within that environment. SLAM is a fundamental capability for mobile robotic systems operating in unstructured or previously unexplored spaces [11].

In the proposed system, SLAM is performed using LiDAR scan data processed at the Mobile Edge Computing (MEC) server. Each LiDAR scan provides a local snapshot of the environment, which must be aligned with previous scans to estimate the rover's motion and build a global representation. Given the teleoperated nature of the rover and the absence of wheel odometry feedback, scan matching forms the primary source of motion estimation.

4.14 ICP-Based Scan Matching Approach

Since the principles of the ICP algorithm were discussed earlier, the implementation here focuses on its practical integration within the SLAM pipeline rather than on reiterating its theoretical formulation. [10].

In each iteration, the algorithm establishes correspondences between points in the current scan and points in a reference scan. A rigid-body transformation consisting of translation and rotation is then estimated to minimize the mean squared error between these correspondences. This transformation represents the relative motion of the rover between scans.

The estimated transformations are accumulated over time to generate a global point cloud representation of the environment. While ICP provides accurate local alignment, it is susceptible to drift over long trajectories in the absence of loop closure detection. Nevertheless, it is well-suited for indoor mapping in controlled environments, as demonstrated in the experimental results presented in Chapter 5 [11]. The effectiveness of the ICP-based scan matching approach is demonstrated through the global point cloud reconstruction results presented in Figure 5.6.

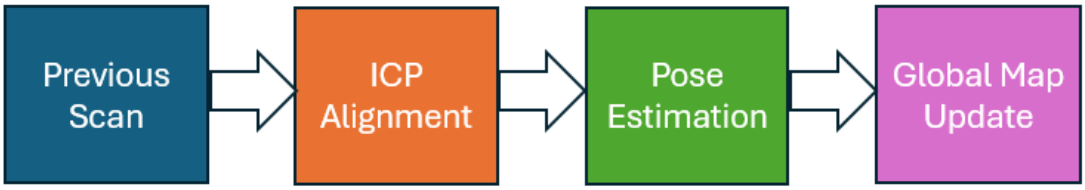


Figure 4.4: ICP-based SLAM pipeline for environment mapping

```python
import open3d as o3d

def align_scans(source, target):
    threshold = 0.2
    result = o3d.pipelines.registration.registration_icp(
        source, target, threshold,
        o3d.pipelines.registration.
    TransformationEstimationPointToPoint()
    )
    return result.transformation
```

Listing 4.3: ICP-based scan alignment loop

Listing 4.3 from *ydlidar_SLAM.py* shows the high-level implementation of ICP-based scan alignment used for point cloud reconstruction.

In addition to point cloud reconstruction, an occupancy grid representation of the environment is generated. Cartesian LiDAR points are rasterized into a two-dimensional grid with a fixed spatial resolution. Each cell in the grid represents the probability of occupancy based on observed measurements.

Occupancy grids provide a compact and intuitive representation of the environment and are well-suited for visualization and navigation tasks.

4.15 Data Logging and Dataset Generation

All reconstructed LiDAR scans and derived mapping outputs are logged with timestamps for offline analysis. Scans are stored in structured formats such as CSV files, enabling their use in machine learning pipelines for tasks such as localization, obstacle detection, and scene classification.

The ability to generate labeled spatial datasets directly from a teleoperated rover represents a valuable byproduct of the proposed system.

To ensure robustness under real-world conditions, fault-tolerance mechanisms are integrated throughout the software stack. Network reconnection logic allows services to recover automatically from temporary link failures. Buffer management and exception handling prevent malformed packets from disrupting the system.

Chapter 5

EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

5.1 Introduction

This chapter presents the experimental validation and performance evaluation of the proposed 5G-enabled rover control system. The experiments were conducted in a controlled private 5G Standalone laboratory environment to assess the system's ability to support real-time teleoperation, reliable sensor data transmission, and edge-assisted LiDAR-based environment mapping.

The results discussed in this chapter consolidate and extend the experimental work carried out during earlier project phases. Specifically, the communication and teleoperation performance evaluated during Project Review 2 (PR2) and the LiDAR sensing and mapping experiments conducted during Project Review 3 (PR3) are unified and analyzed within a single end-to-end system framework.

5.2 Experimental Setup

5.2.1 Network Environment

All experiments were performed using a private 5G Standalone network deployed in a laboratory setting. The network consisted of a 5G core, a gNodeB (gNB), and a 5G user equipment (UE) module integrated within the evaluation board. The use of a private network ensured predictable latency, controlled bandwidth allocation, and isolation from external traffic.

The evaluation board functioned as a 5G user equipment and communication relay, while all edge computing tasks were executed on the MEC server within the private 5G network.

5.2.2 Hardware Configuration

The rover platform was equipped with:

- A Raspberry Pi for motor control, camera interfacing, and network communication
- A differential drive motor base
- A camera module for live video streaming

- A two-dimensional LiDAR sensor for range sensing

The evaluation board hosted relay and proxy services, while all LiDAR processing and mapping software was executed on the MEC server.

5.2.3 Software Configuration

Persistent TCP connections were established for control commands, video streaming, and LiDAR data forwarding. All experiments were conducted using the multi-threaded software framework described in Chapter 4.

5.3 Teleoperation and Control Performance

To evaluate control responsiveness, operator-issued commands were transmitted from the operator station to the rover via the evaluation board. The rover's response was observed both physically and through the live video feed.

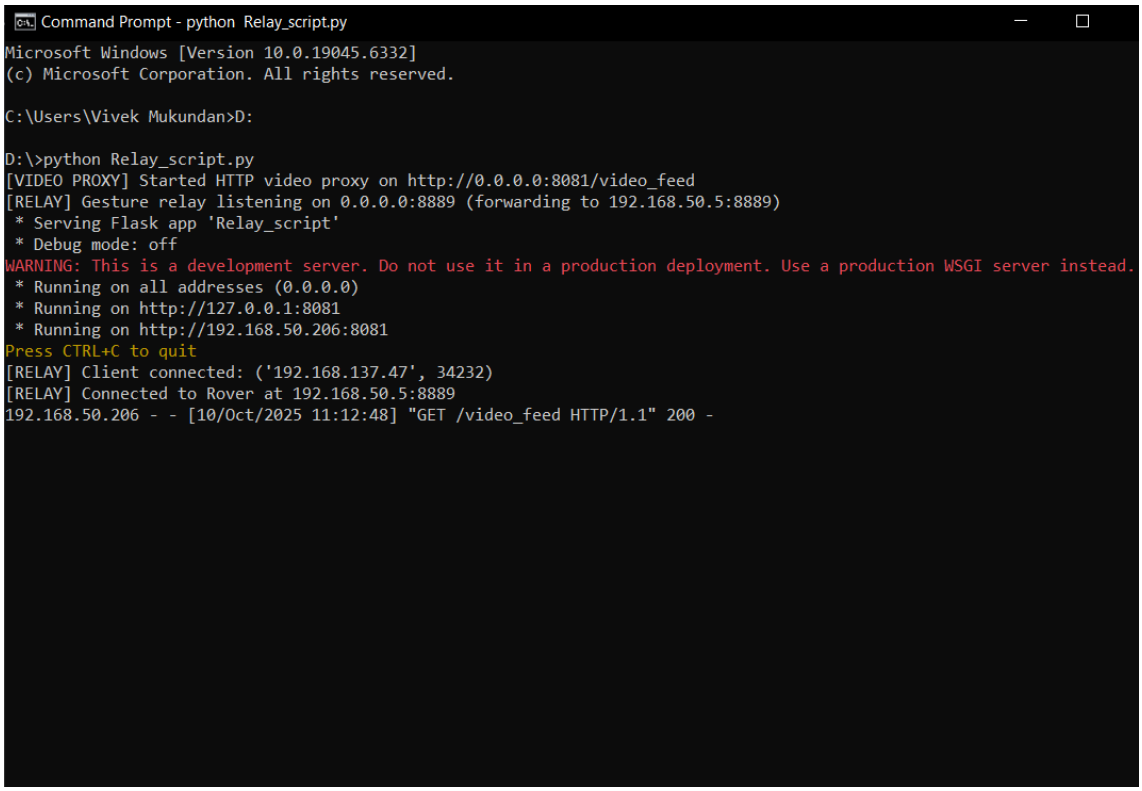


Figure 5.1: Validation of network relay and rover command communication

Figure 5.1 shows the terminal output of the relay server during system operation. Successful connections from both the operator station and the rover platform are observed, indicating reliable establishment of bidirectional communication channels required for real-time teleoperation.

The system demonstrated stable and consistent command execution. Discrete rover control commands such as forward motion, turning, and stopping were reliably executed without packet loss or misinterpretation. The motion smoothing logic implemented on the rover ensured gradual acceleration and deceleration, resulting in stable movement suitable for precise teleoperation.

5.3.1 End-to-End Control Latency

End-to-end latency was measured as the time interval between issuing a control command at the operator station and observing the corresponding response in the video feed. Across multiple trials, the measured latency ranged between approximately 350 ms and 400 ms.

While this latency exceeds the theoretical lower bounds of URLLC, it remained acceptable for the intended application of careful remote exploration and crime scene investigation. The observed delay was primarily attributed to video encoding and MJPEG streaming overhead rather than command transmission itself.

The experimental results demonstrate that the private 5G network is capable of supporting simultaneous bidirectional control traffic and high-throughput sensor data transmission without noticeable degradation in performance. The stable video streaming and consistent LiDAR data reception observed during experiments indicate that the 5G communication link provides sufficient bandwidth and reliability for real-time robotic applications.

5.4 Video Streaming Performance

The MJPEG-based video streaming mechanism provided a continuous and stable visual feed throughout the experiments. The evaluation board's video proxy successfully decoupled the operator from the rover's local network, allowing seamless access to the stream over the 5G link.

Automatic reconnection logic ensured that temporary network disruptions or rover restarts did not permanently interrupt the video feed.

Despite the high bandwidth requirements of video streaming, the multi-threaded design prevented interference with control commands. Control responsiveness remained stable even during sustained video transmission, validating the effectiveness of the concurrent software architecture. Figure 5.2 presents the live video feed received at the operator station through the MJPEG streaming pipeline. The continuous and stable video output demonstrates the effectiveness of the video proxy mechanism implemented on the evaluation board.

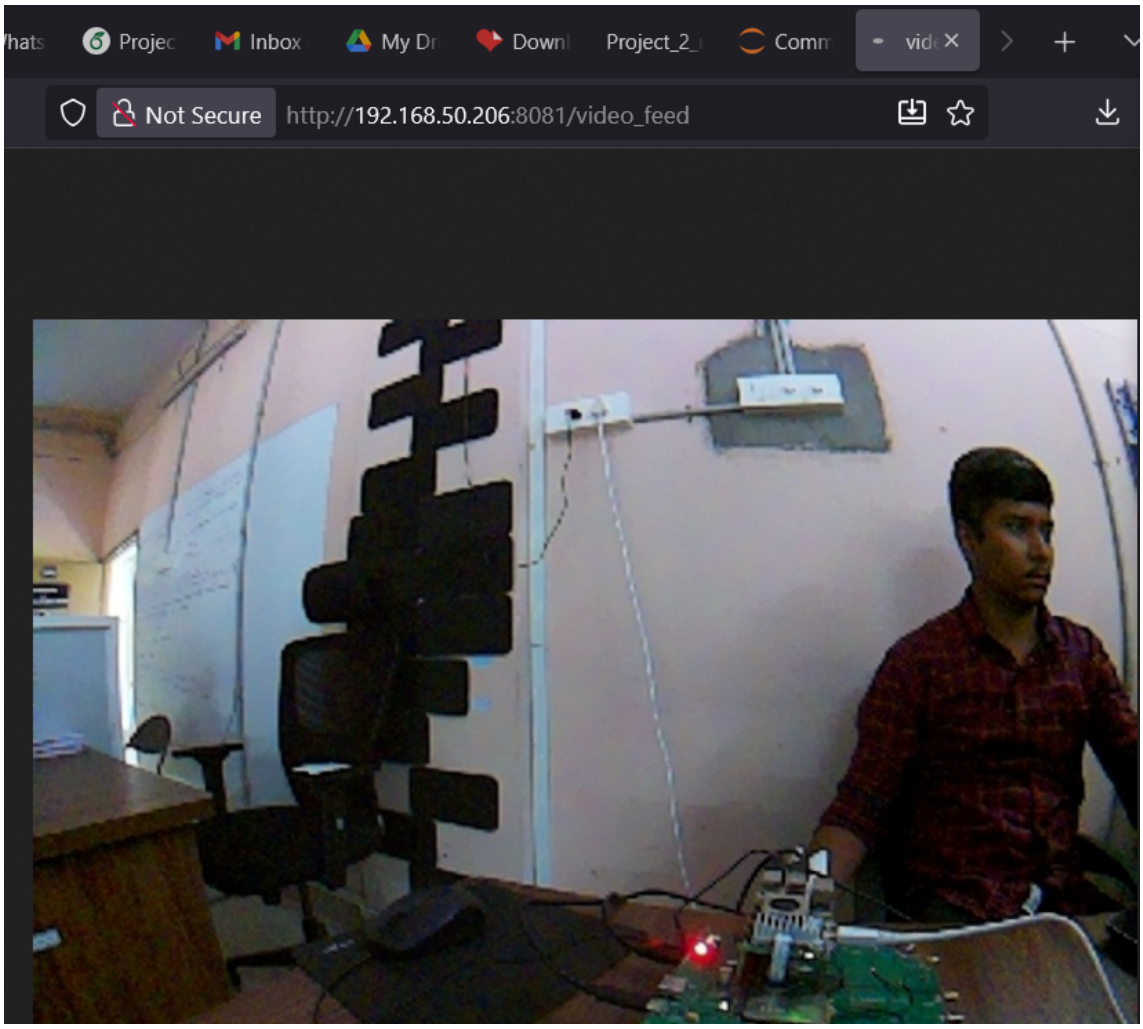


Figure 5.2: Live video stream received at the operator station

5.5 LiDAR Data Transmission Validation

The serial-to-TCP forwarding mechanism reliably transported raw LiDAR byte streams from the rover to the evaluation board. Packet headers were preserved despite TCP fragmentation, and no data corruption was observed during extended operation.

Validation of the forwarded stream confirmed the presence of correct packet headers and consistent scan rates, indicating successful low-level data transmission.

The buffered packet reconstruction logic on the MEC Server successfully assembled complete 360-degree LiDAR scans. Each scan contained distance measurements for all angular positions, confirming that packet fragmentation did not disrupt scan integrity.

Reconstructed scans were visualized in real time and logged for offline analysis.

Figure 5.3 shows a reconstructed LiDAR scan obtained after packet parsing on the MEC Server. The presence of continuous and evenly distributed distance

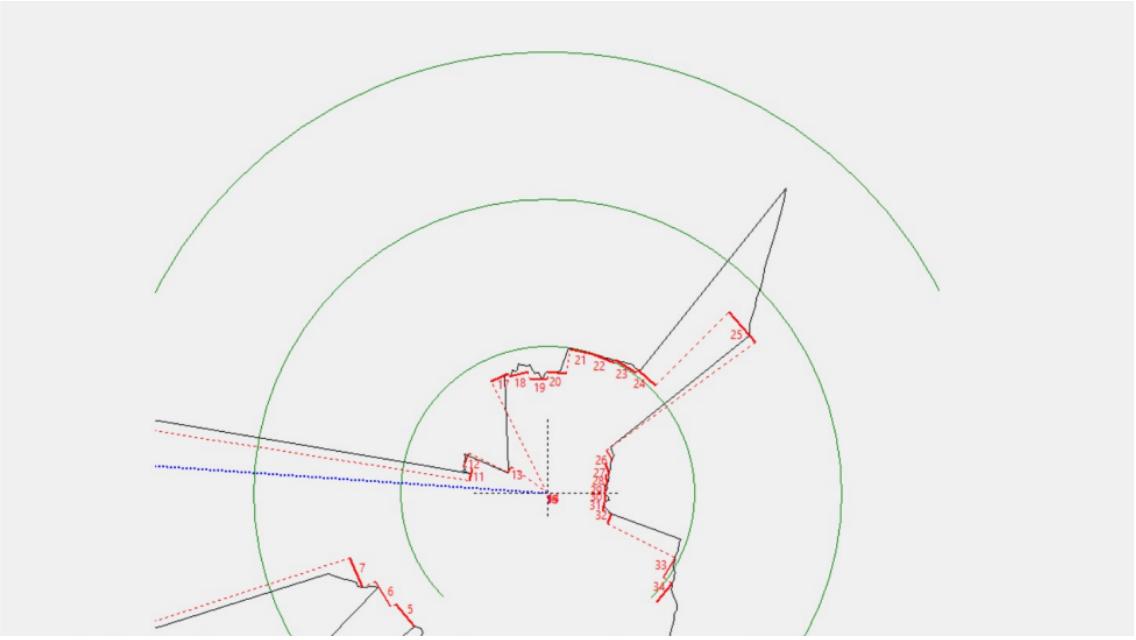


Figure 5.3: Visualization of a reconstructed 360-degree LiDAR scan

measurements across all angles confirms the integrity of LiDAR data transmission and scan reconstruction.

5.6 Dataset Generation and Logging

LiDAR scans were logged with timestamps and angular indices, creating structured datasets suitable for machine learning applications. These datasets can be leveraged for tasks such as localization, obstacle detection, and environment classification, extending the utility of the system beyond real-time teleoperation.

In Figure 5.4, *timestamp* shows precise system time at moment of capture, *scan_index* tracks the 0–359° completion cycle, *angle_deg* refers to direction of the LiDAR beam and *distance_mm* shows measured range in millimeters.

5.7 Environment Mapping Results

5.7.1 Occupancy Grid Mapping

Occupancy based grid maps generated from LiDAR scans accurately represented the spatial layout of indoor environments. Walls, corners, and obstacles were clearly identifiable in the reconstructed maps. The grid maps provided a compact and intuitive visualization suitable for navigation and analysis.

Figure 5.5 shows the occupancy grid map of the indoor environment constructed using accumulated LiDAR scans. Occupied regions corresponding to walls and

obstacles are clearly visible, while free space is represented by the unoccupied areas. This visualization demonstrates the system's ability to infer spatial occupancy and environment structure from LiDAR data.

	timestamp_iso	scan_index	angle_deg	distance_mm
2	22:55.3	0	0	32768
3	22:55.3	0	1	32768
4	22:55.3	0	2	32768
5	22:55.3	0	3	32768
6	22:55.3	0	4	32768
7	22:55.3	0	5	32768
8	22:55.3	0	6	3862
9	22:55.3	0	7	3840
10	22:55.3	0	8	3820
11	22:55.3	0	9	3773
12	22:55.3	0	10	3692
13	22:55.3	0	11	32768
14	22:55.3	0	12	32768
15	22:55.3	0	13	32768
16	22:55.3	0	14	21930
17	22:55.3	0	15	7680
18	22:55.3	0	16	41521
19	22:55.3	0	17	44469
20	22:55.3	0	18	17787

Figure 5.4: Sample of collected LiDAR dataset containing timestamp, scan_index, angle_deg, and distance_mm.

The occupancy-based environment representation provides an intermediate spatial understanding between raw LiDAR scans and full SLAM-based mapping. By accumulating multiple LiDAR measurements over time, regions corresponding to walls, obstacles, and free space begin to emerge clearly. Unlike a single scan, this representation captures spatial consistency across multiple rover positions, allowing persistent environmental features to be distinguished from noise and transient measurements. Such representations are particularly useful for validating the correctness of scan accumulation and pose estimation prior to performing global map reconstruction.

5.7.2 SLAM-Based Point Cloud Reconstruction

ICP-based scan matching enabled the generation of global point clouds by aligning consecutive scans. The resulting maps demonstrated reasonable spatial consistency, with minor drift observed during extended runs. Periodic alignment and pose correction improved map quality.

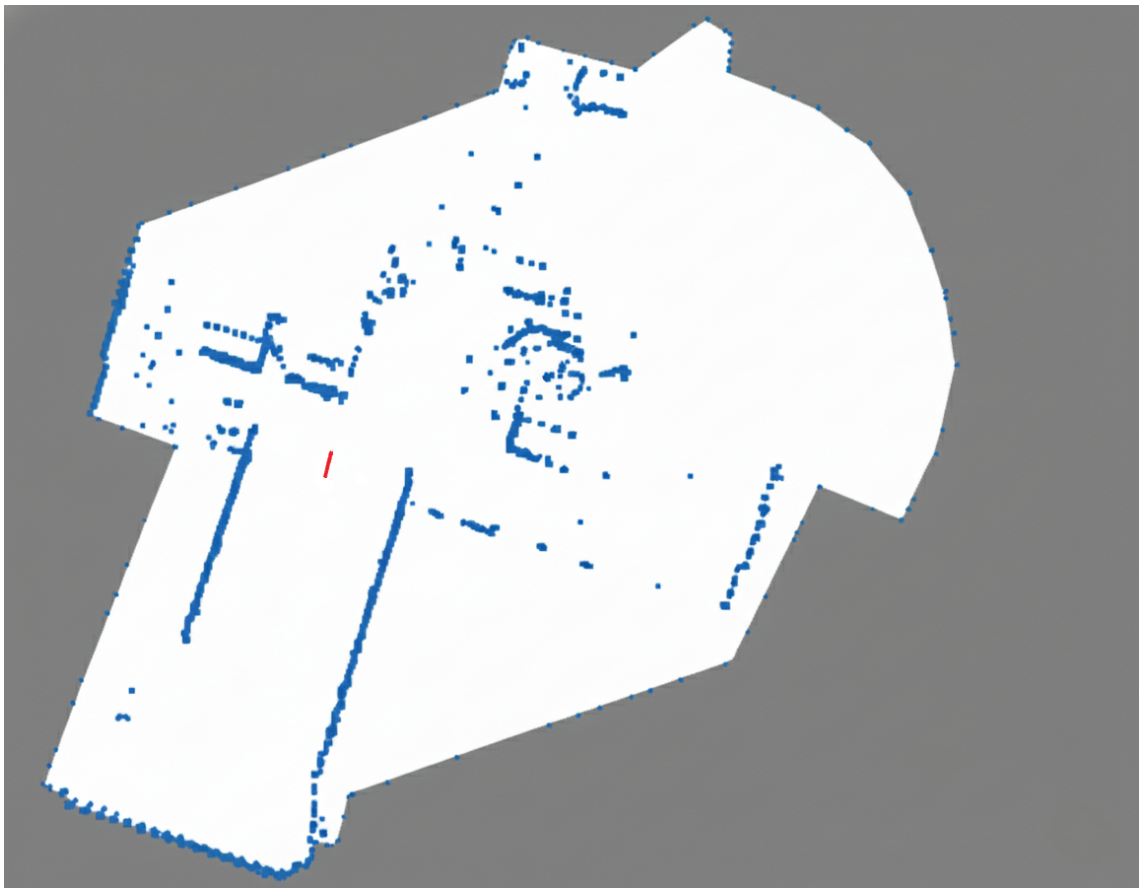


Figure 5.5: Occupancy-based environment representation generated from accumulated LiDAR scans

Figure 5.6 presents the global point cloud map generated by aligning consecutive LiDAR scans using the ICP algorithm. The map captures the spatial structure of the environment and validates the effectiveness of SLAM processing executed on the MEC server via the private 5G network. The red locus of points represents the path that is taken by the rover for tracing of the path. The arrow represents the direction in which the rover is facing while tracing. Same directional arrow is represented in the actual room picture.

While occupancy-based representations provide local spatial consistency, global environment reconstruction requires accurate alignment of successive scans as the rover moves through the environment. To achieve this, ICP-based SLAM is employed to estimate relative motion between consecutive LiDAR scans and integrate them into a single global reference frame. This process reduces cumulative drift and enables the construction of a coherent point cloud map that reflects the overall geometry of the environment. The resulting global map serves as a comprehensive validation of both the LiDAR data pipeline and the SLAM algorithm.

Figure 5.7 shows the actual indoor environment in which the LiDAR-based mapping experiments were conducted. This figure provides visual context for the

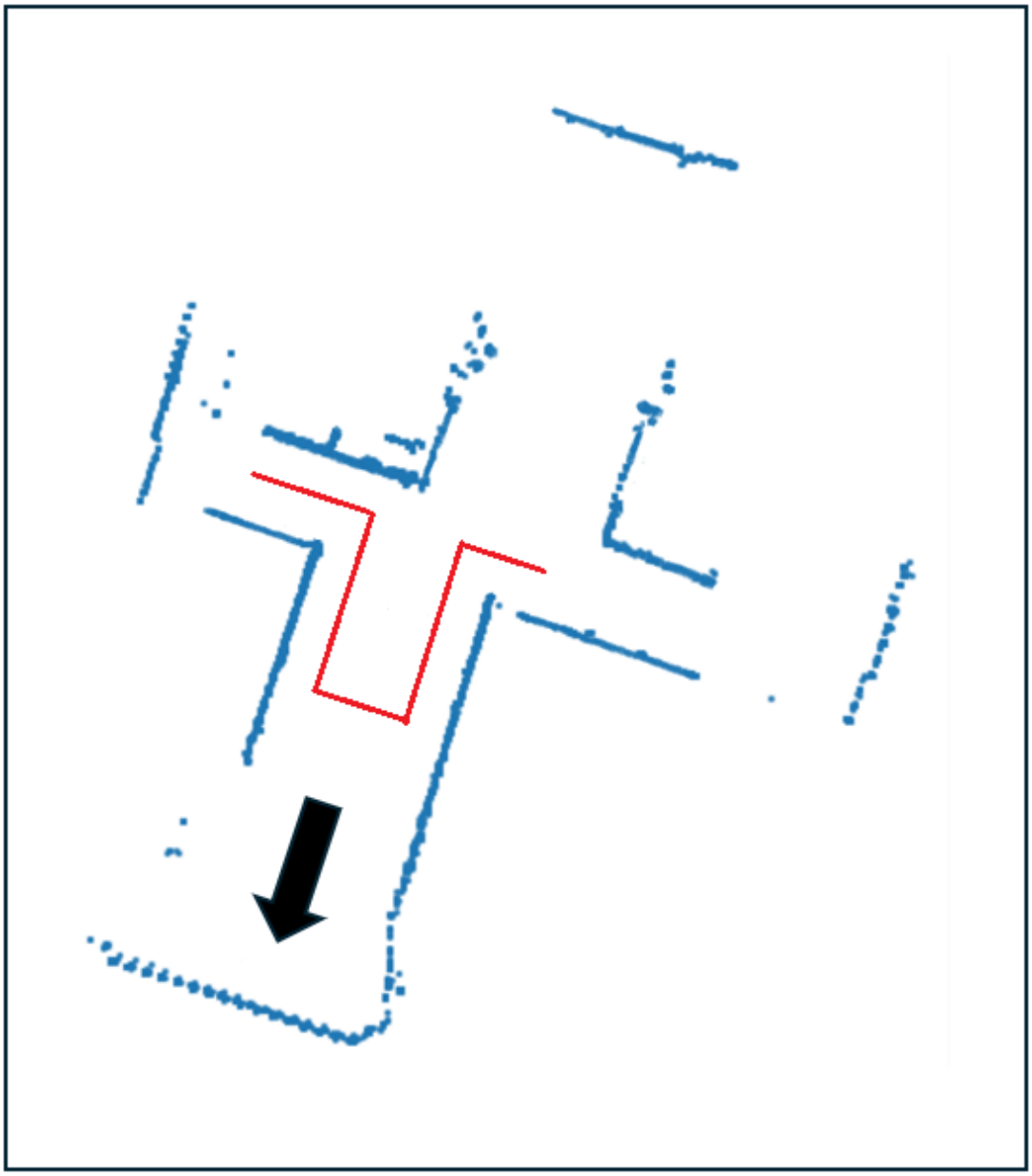


Figure 5.6: Global point cloud map generated using ICP-based SLAM

reconstructed maps and helps correlate physical structures such as walls and corners with the features observed in the LiDAR-based SLAM results.

5.8 Limitations

Although the system performed reliably, several limitations were identified:

- MJPEG video streaming introduced significant latency and bandwidth overhead.
- Latency increased slightly under high network load or when multiple clients accessed the video stream.

Figure 5.7: Actual indoor environment used for LiDAR mapping experiments

- ICP-based mapping exhibited drift in the absence of loop closure detection in larger environments.
- Occasional packet delays were observed during prolonged experiments, requiring manual restart of certain services.

These limitations highlight opportunities for further optimization and enhancement.

Chapter 6

SUMMARY, CONCLUSIONS, AND FUTURE SCOPE

6.1 Summary of the Work

This thesis presented the design, implementation, and experimental validation of a 5G-enabled rover control system capable of real-time teleoperation and environmental sensing using a private 5G Standalone network. The work was carried out in multiple phases and progressively enhanced to address limitations identified in earlier stages.

The initial phase of the work focused on establishing the motivation and feasibility of using private 5G networks for time-sensitive applications. A comprehensive study of Industry 4.0, Internet of Things (IoT) systems, and fifth-generation wireless communication technologies was conducted. The foundational communication architecture was designed to support bidirectional data flow between a remote operator and a mobile platform, validating the suitability of 5G Standalone networks for low-latency and high-reliability applications.

In the subsequent phase, a complete teleoperation framework was implemented. This phase introduced a network relay architecture using an evaluation board as an external system to bridge the operator and rover networks. A multi-threaded command relay service and rover-side command server enabled responsive control of rover motion and camera orientation. Simultaneously, a video streaming and proxy mechanism was developed to provide continuous visual feedback to the operator. Experimental validation demonstrated stable teleoperation with acceptable end-to-end latency for careful remote exploration tasks.

The final phase extended the system to include LiDAR-based environmental sensing and mapping. To overcome the computational constraints of the rover, an edge-assisted architecture was adopted in which raw LiDAR data were forwarded from the rover to the evaluation board using a serial-over-TCP mechanism. Robust packet reconstruction logic was implemented to handle network fragmentation, enabling reliable scan generation. Advanced processing tasks, including scan alignment, occupancy grid mapping, and point cloud reconstruction, were executed on the MEC Server via the 5G Evaluation Board. The system successfully generated spatial representations of indoor environments while maintaining real-time teleoperation capability.

Together, these phases resulted in a unified end-to-end system that integrates low-latency control, real-time video streaming, and LiDAR-based environment mapping over a private 5G network.

6.2 Conclusions

Based on the design, implementation, and experimental evaluation presented in this thesis, the following conclusions are drawn:

1. Private 5G Standalone networks provide a reliable and predictable communication backbone for real-time robotic teleoperation, offering significant advantages over conventional wireless technologies in terms of latency and stability.

2. The use of an evaluation board as a mobile gateway and relay node, with edge computing performed on the MEC server enables effective offloading of computationally intensive tasks, thereby reducing the processing burden on resource-constrained rover platforms.

3. A modular, multi-threaded software architecture is essential for maintaining responsiveness in systems that simultaneously handle control commands, high-bandwidth video streams, and high-rate sensor data.

4. The implemented network relay and video proxy mechanisms successfully decouple the operator from the rover's local network, improving system robustness and scalability.

5. Serial-over-TCP forwarding combined with buffered packet reconstruction provides a reliable solution for transmitting raw LiDAR data over packet-switched networks without loss of scan integrity.

6. LiDAR-based occupancy grid mapping and scan alignment techniques significantly enhance situational awareness compared to camera-only teleoperation, particularly in enclosed or visually ambiguous environments.

The experimental results validate the feasibility of integrating teleoperation, video streaming, and LiDAR-based mapping into a single coherent system operating over a private 5G network. Overall, the work demonstrates that 5G-enabled edge-assisted robotic systems can meet the performance requirements of real-time remote exploration and monitoring applications.

6.3 Limitations of the Present Work

Despite the successful implementation and validation of the proposed system, certain limitations were identified:

- The MJPEG-based video streaming approach introduces non-negligible latency and bandwidth overhead, limiting responsiveness for highly dynamic tasks.

- While ICP-based scan alignment produces usable maps, accumulated drift can occur in larger environments due to the absence of robust loop closure mechanisms.
- The system relies on manual teleoperation and does not incorporate autonomous navigation or obstacle avoidance.
- Experimental validation was conducted in a controlled laboratory environment, and performance in outdoor or highly congested radio environments was not evaluated.

These limitations highlight areas for improvement and guide future research directions.

6.4 Future Scope

The proposed system provides a strong foundation for several extensions and enhancements, including:

1. **Low-Latency Video Streaming**: Replacing MJPEG with more efficient video codecs such as H.264 or H.265, combined with hardware acceleration, can significantly reduce end-to-end latency.
2. **Virtual Reality Integration**: Integrating immersive VR headsets with head-tracking and stereo video can enhance operator perception and control accuracy.
3. **Autonomous Navigation**: Incorporating autonomous navigation, obstacle avoidance, and path planning algorithms can reduce operator workload and improve safety.
4. **Advanced SLAM Techniques**: Implementing loop closure detection and graph optimization can improve mapping accuracy and scalability.
5. **Multi-Robot Coordination**: Extending the architecture to support multiple rovers operating simultaneously over the same 5G network can enable collaborative exploration.
6. **Field Deployment**: Evaluating the system in real-world environments such as disaster sites or industrial facilities can provide further insight into practical performance and robustness.

The modular and scalable nature of the proposed architecture makes it well-suited for these future enhancements.

6.5 Closing Remarks

This thesis demonstrates the practical viability of combining private 5G networks, edge computing, and mobile robotics to achieve real-time teleoperation and environment sensing. The work bridges theoretical advances in 5G communication with practical robotic system design and provides a valuable reference for future research and development in 5G-enabled cyber-physical systems.

Bibliography

[1] ITU-R, "IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond," Recommendation ITU-R M.2083-0, International Telecommunication Union, Sep. 2015.

[2] J. G. Andrews, S. Buzzi, W. Choi, et al., "What Will 5G Be?," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[3] P. Rost, A. Banchs, I. Berberana, et al., "Mobile Network Architecture Evolution toward 5G," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 84–91, May 2016.

[4] ETS I, "Mobile Edge Computing (MEC); Framework and Reference Architecture," ETSI GS MEC 003, 2019.

[5] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.

[6] Python Software Foundation, "socket — Low-level networking interface," Python Documentation. [Online]. Available: https://docs.python.org/3/library/socket.html

[7] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, O'Reilly Media, 2018.

[8] G. Wallace, "The JPEG Still Picture Compression Standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.

[9] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005.

[10] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[11] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, June 2006.

[12] YDLIDAR Inc., "YDLIDAR X2/X2L Development Manual," 2021. [Online]. Available: https://www.ydlidar.com

[13] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *arXiv preprint arXiv:1801.09847*, 2018.

[14] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[15] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, 1994.

[16] Y. Cao et al., "Real-Time LiDAR Point Cloud Compression and Transmission for Resource-Constrained Robots," *arXiv preprint arXiv:2502.06123*, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2502.06123

[17] M. Kizilkaya et al., "5G-Based Low-Latency Teleoperation: Two-Way Timeout Approach," in *Proc. 2023 Int. Conf. on Advances in Vehicular Systems and Technologies (AVST)*, 2023. [Online]. Available: https://doi.org/10.1007/978-3-031-51015-8-6

[18] Y. Wang et al., "Occupancy-SLAM: an Efficient and Robust Algorithm for SLAM," *arXiv preprint arXiv:2502.06292*, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2502.06292

Chapter 7

Appendix

The codes used for this project can be found at:
https://github.com/VivekMukundan/5G-enabled-Rover-Control-using-Evaluation-Board-for-remote-teleoperation-and-LiDAR-based-Mapping