Abstract

This report details my six-month internship as a Software Development Intern (Fullstack) at Unbox Robotics, a leading automation machinery manufacturing technology company. The primary focus was developing a PCB Testing Tool to automate the testing process of Printed Circuit Boards (PCBs) for various robotic systems. The tool utilized React (Vite.js) for the frontend, FastAPI for the backend, and WebSocket communication to interface with C++ middleware for serial communication with PCBs. Additionally, I contributed to implementing UART communication in C++ for sensor integration. The internship enhanced my skills in fullstack development, embedded systems, and automation, significantly reducing manual testing time and improving efficiency.

Contents

List of Figures

Chapter 1

Introduction

PCB Testing Tool: Developing an Browser based PCB Testing Tool which connects with PCB middle-ware interactions to automate the testing process of the PCB of different versions and models.

Implementing UART in C++ : This Project aims to demonstrate the feasibility of implementing UART communication in intel Atom Processor. The primary motivation for this project is to integrate a new PDS sensor into robots. UART has been chosen as the communication channel because the Atom has unused UART pins, and this approach helps minimize the use of USB ports, of which only three are available on the Atom.

1.1 Overview

Unbox Robotics specializes in robotics-based fulfillment and distribution technology for e-commerce, retail, and logistics enterprises. Their mission is to empower customers with efficient order fulfillment using advanced robotic systems, aiming to be a global leader in logistics automation.

During my internship from January 15, 2025, to July 15, 2025, till now I have worked on two key projects: developing a PCB Testing Tool and implementing UART communication in C++ for sensor integration. The PCB Testing Tool aimed to automate testing processes for various PCB models, while the UART project focused on enabling efficient sensor data communication on an Intel Atom processor.

1.2 Roles and Responsibilities

As a Software Development Intern (Full Stack) at Unbox Robotics, I was actively involved in the end-to-end development of an internal test tool for the Production Team as well as working on software products of Unbox.

My key responsibilities included:

- Design user flows, information architecture, and follow core design principles to build intuitive and user-centric software solutions.
- Define system designs and technical specifications in alignment with functional requirements.
- Design scalable and modular software architecture through active collaboration with stakeholders.
- Develop robust software tools and internal applications to support the production and operations teams.
- Conduct end-to-end verification, testing, and deployment of developed solutions to ensure stability and performance.
- Understand and implement communication protocols such as UART on embedded systems and integrate them with higher-level applications.
- Configure reliable communication between processing units and external peripherals to support data exchange and control operations.
- Write and test scripts for serial communication across systems and peripherals.
- Debug and resolve software issues to maintain system integrity and efficiency.
- Document technical findings and propose improvements for better performance, maintainability, and scalability.

Chapter 2

Analysis of Existing Solutions

The existing PCB Testing Tool, built in C++, tested boards like AgvCb, ods, and AgvCob using defined test cases. It utilized shared memory for inter-process communication (IPC) and featured a desktop UI for setup and test environment logging, with CSV report generation. However, it lacked scalability, modern UI interactions, and efficient automation for diverse PCB versions.

For UART communication, traditional solutions included microcontroller-based systems (e.g., Arduino, ESP32) and Linux-based frameworks like PySerial and termios. These were limited by hardware constraints or inefficient data handling for high-frequency sensor data (824 bytes at 24 Hz).

Chapter 3

Proposed System

3.1 Problem Statement

PCB Testing Tool: The existing tool required manual intervention, taking 1-2 hours per test cycle, and lacked a scalable architecture for new PCB versions. The goal was to develop an automated, user-friendly tool with a modern UI, reducing testing time and improving scalability.
UART Implementation: Integrate a PDS sensor into robots using UART to minimize USB port usage, ensuring reliable data transfer (824 bytes every 43 ms) with minimal CPU utilization of less than (3%).

3.2 Project Methodology/Techniques

Literature Review: Studied FastAPI, React, WebSocket protocols, PostgreSQL, and UART principles.
Experimental Approach: Conducted Proof of Concepts (POCs) to select the technology stack (FastAPI, React, PostgreSQL) and validate WebSocket communication for PCB testing.
Software Development:
Frontend: React (Vite.js) for interactive UI.
Backend: FastAPI for API development and WebSocket integration.
Database: PostgreSQL for test data storage.
Middleware: C++ for serial communication with PCBs, interfaced via WebSockets.
Hardware Configuration: Configured UART on Intel Atom Up Squared 2 for sensor communication.

3.3 Design and Architecture

PCB Testing Tool:

Frontend: React (Vite.js) provided a dynamic dashboard for test setup and result visualization.

Backend: FastAPI handled API endpoints for test automation and WebSocket communication.

Middleware: C++ managed serial communication with PCBs, relaying data to FastAPI via WebSockets.

Database: PostgreSQL stored test logs and reports.

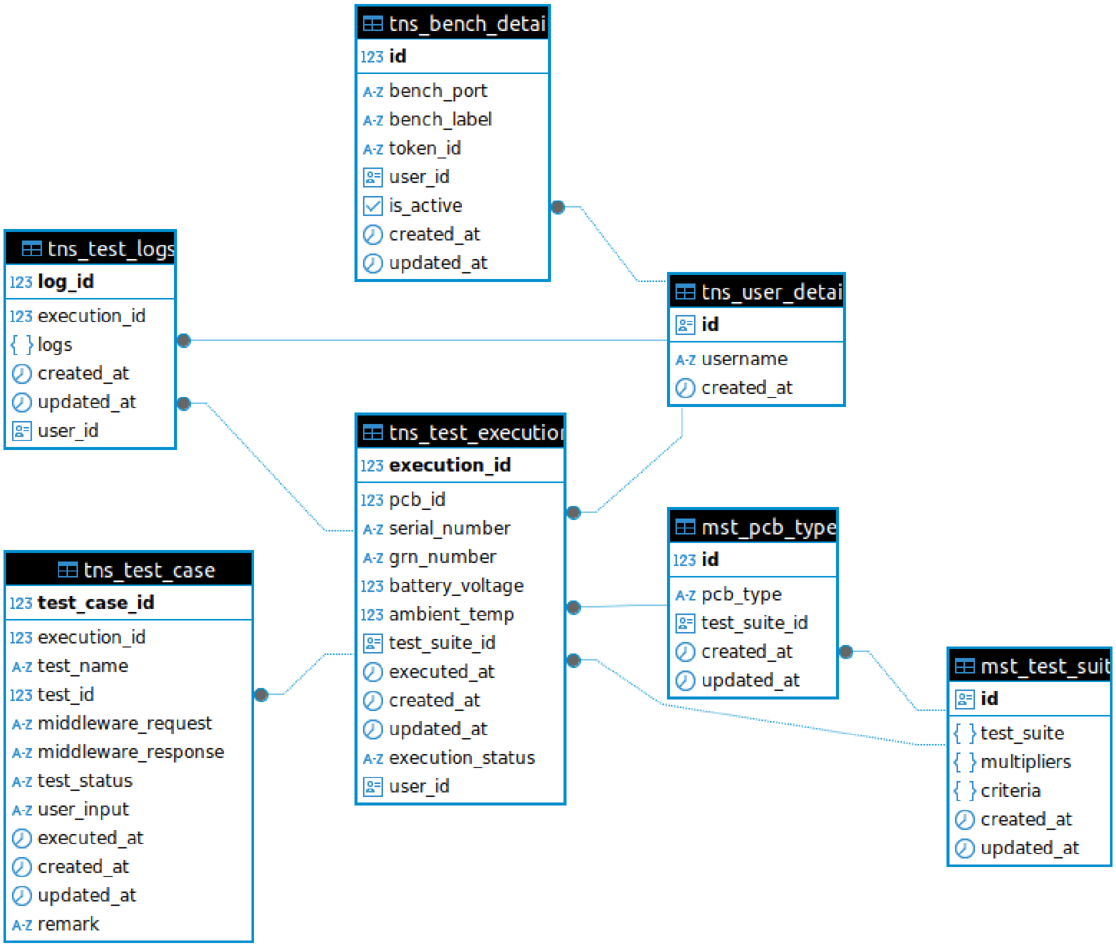
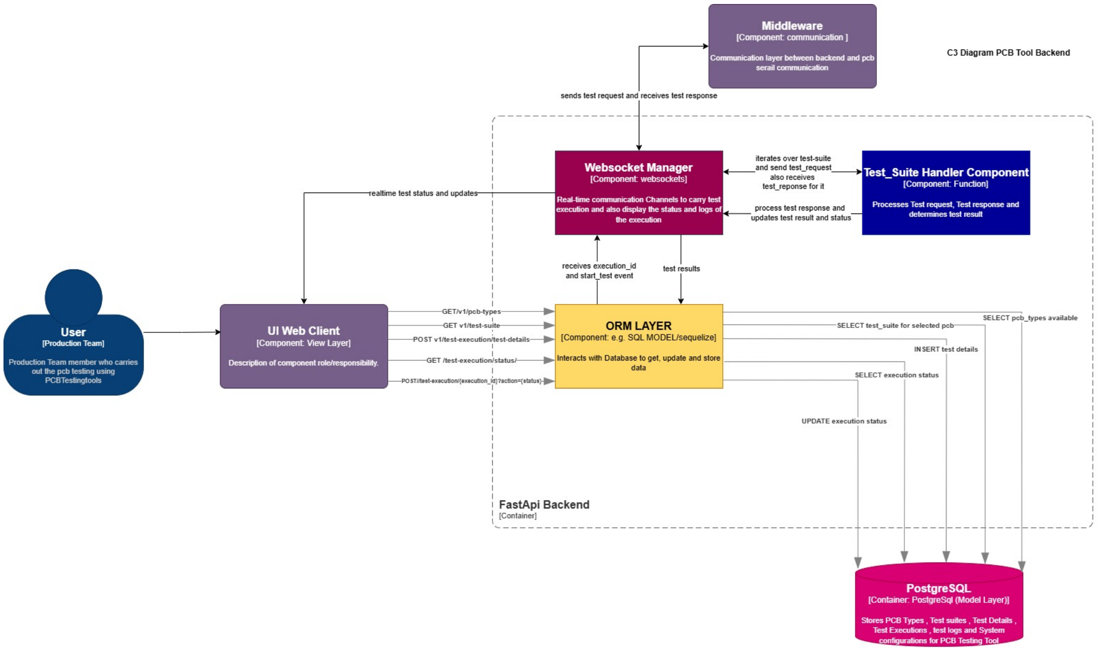


Figure 3.1: Database Architecture

Architecture: MVC pattern for backend, ensuring modularity and scalability.



Middleware
[Component: communication]
Communication layer between backend and pcb serail communication
C3 Diagram PCB Tool Backend
sends test request and receives test response
Websocket Manager
[Component: websockets]
Real-time communication Channels to carry test execution and also display the status and logs of the execution
iterates over test-suite and send test_request also receives test_reponse for it
process test response and updates test result and status
Test_Suite Handler Component
[Component: Function]
Processes Test request, Test response and determines test result
realtime test status and updates
receives execution_id and start_test event
test results
User
[Production Team]
Production Team member who carries out the pcb testing using PCBTestingtools
UI Web Client
[Component: View Layer]
Description of component role/responsibility.
GET/v1/pcb-types
GET v1/test-suite
POST v1/test-execution/test-details
GET /test-execution/status/
POST//test-execution/{execution_id}?action={status}
ORM LAYER
[Component: e.g. SQL MODEL/sequelize]
Interacts with Database to get, update and store data
SELECT pcb_types available
SELECT test_suite for selected pcb
INSERT test details
SELECT execution status
UPDATE execution status
FastApi Backend
[Container]
PostgreSQL
[Container: PostgreSql (Model Layer)]
Stores PCB Types , Test suites , Test Details , Test Executions , test logs and System configurations for PCB Testing Tool

Figure 3.2: C3 Diagram PCB Testing Tool

UART Implementation: Configured serial ports with specific baud rates, parity, and stop bits. Developed C++ scripts for data reading and error handling, integrated with CSV logging.

3.4 Skills Development

Software Development: Mastered fullstack development with React, FastAPI, and PostgreSQL; implemented WebSocket communication.

Embedded Systems: Gained expertise in UART configuration, C++ middleware, and PCB testing automation.

Testing and Automation: Developed automated test workflows and report generation systems.

Collaboration: Enhanced stakeholder communication and technical documentation skills.

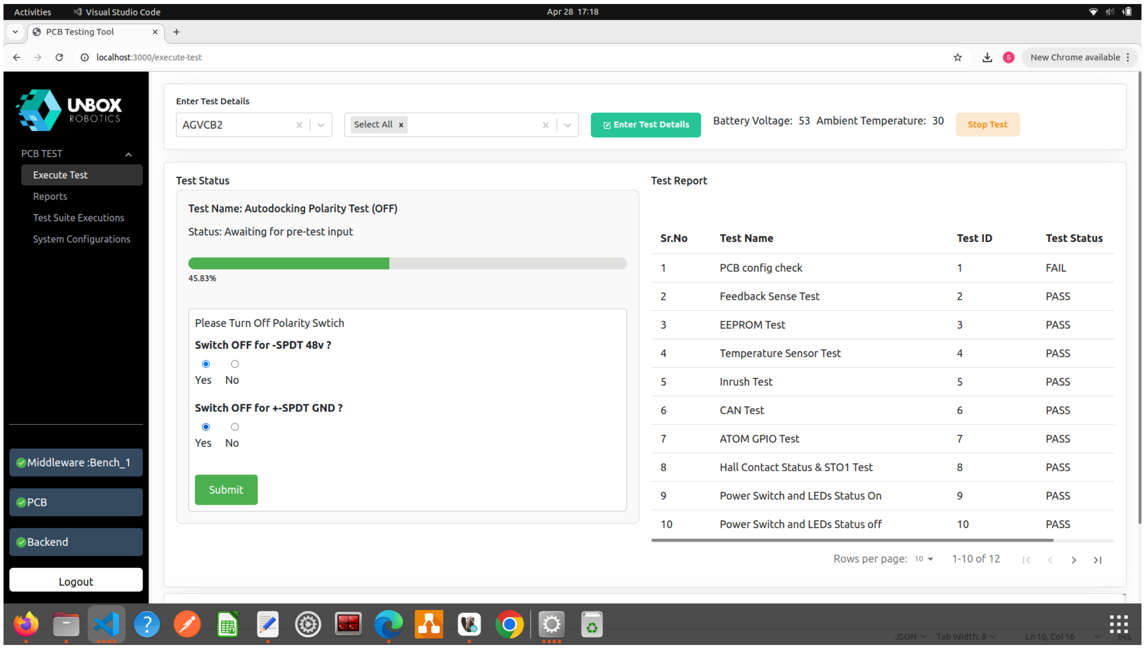
Chapter 4

Results and Discussion

4.1 Implementation Details

PCB Testing Tool:

Developed a responsive UI with React (Vite.js) for test configuration and result display. Built FastAPI backend with APIs for test execution and WebSocket endpoints for real-time middleware communication. Integrated C++ middleware for serial PCB interaction, ensuring seamless data flow. Implemented PostgreSQL schemas for storing test logs and generating CSV reports.



Enter Test Details

AGVCB2 | Select All

Battery Voltage: 53 Ambient Temperature: 30

Test Status

Test Name: Autodocking Polarity Test (OFF)

Status: Awaiting for pre-test input

45.83%

Please Turn Off Polarity Swtich

Switch OFF for -SPDT 48v ?

● Yes ○ No

Switch OFF for +-SPDT GND ?

● Yes ○ No

Test Report

Sr.No	Test Name	Test ID	Test Status
1	PCB config check	1	FAIL
2	Feedback Sense Test	2	PASS
3	EEPROM Test	3	PASS
4	Temperature Sensor Test	4	PASS
5	Inrush Test	5	PASS
6	CAN Test	6	PASS
7	ATOM GPIO Test	7	PASS
8	Hall Contact Status & STO1 Test	8	PASS
9	Power Switch and LEDs Status On	9	PASS
10	Power Switch and LEDs Status off	10	PASS

Rows per page: 10 1-10 of 12

Figure 4.1: Dashboard PCB Testing Tool

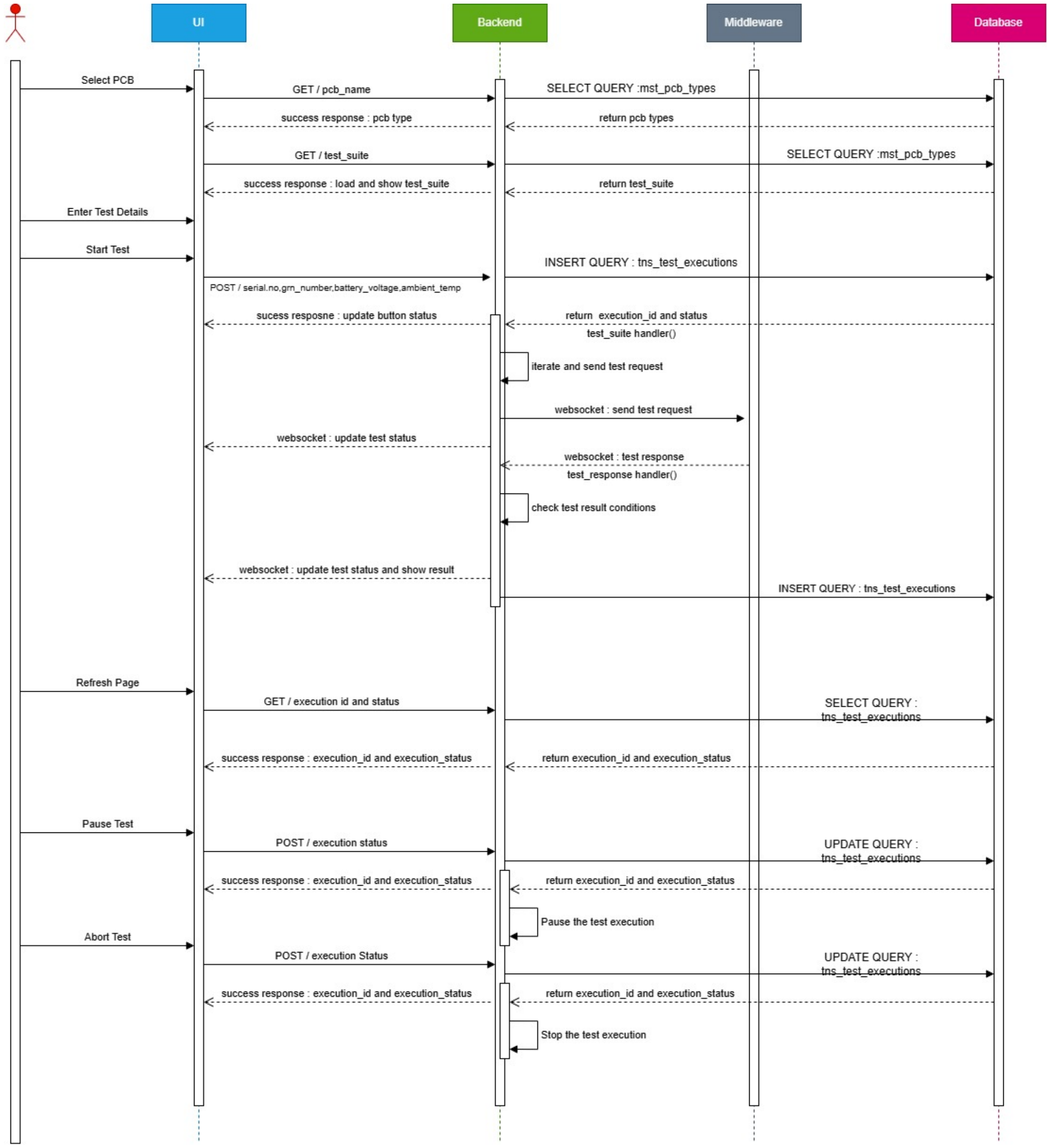

UI
Backend
Middleware
Database
Select PCB
GET / pcb_name
SELECT QUERY :mst_pcb_types
success response : pcb type
return pcb types
GET / test_suite
SELECT QUERY :mst_pcb_types
success response : load and show test_suite
return test_suite
Enter Test Details
Start Test
INSERT QUERY : tns_test_executions
POST / serial.no,grn_number,battery_voltage,ambient_temp
sucess resposne : update button status
return execution_id and status
test_suite handler()
iterate and send test request
websocket : send test request
websocket : update test status
websocket : test response
test_response handler()
check test result conditions
websocket : update test status and show result
INSERT QUERY : tns_test_executions
Refresh Page
GET / execution id and status
SELECT QUERY :
tns_test_executions
success response : execution_id and execution_status
return execution_id and execution_status
Pause Test
POST / execution status
UPDATE QUERY :
tns_test_executions
success response : execution_id and execution_status
return execution_id and execution_status
Pause the test execution
Abort Test
POST / execution Status
UPDATE QUERY :
tns_test_executions
success response : execution_id and execution_status
return execution_id and execution_status
Stop the test execution

Figure 4.2: Sequence Flow Diagram of PCB Testing Tool

UART Implementation:

Configured UART on Intel Atom Up Squared 2 to read 824 bytes every 43 ms. Developed C++ scripts for data processing, error handling, and CSV logging. Optimized data reading to prevent packet loss and ensure synchronization.

4.2 Testing Strategies

Unit Testing: Validated FastAPI endpoints and React components.

Integration Testing: Ensured seamless WebSocket communication between frontend, backend, and middleware.

Hardware Testing: Executed PCB test scripts to validate outputs.

Performance Testing: Optimized test execution to reduce cycle time.

UART Implementation:

Tested various baud rates and configurations to ensure reliable data transfer. Used tools like minicom to debug connectivity issues. Analyzed CSV logs to identify packet drops or mismatches.

4.3 Result Analysis

PCB Testing Tool:

Reduced manual testing time from 2-3 hours to 5 minutes. Achieved consistent UI design with existing Unbox Robotics products. Developed an initial MVP with frontend, backend, and database integration. Project remains in the development phase, with ongoing enhancements.

UART Implementation:

Achieved an average packet time difference of 42.82 ms, meeting the 43 ms requirement. Maintained CPU utilization below 1.5%, well under the 3% target. Identified and mitigated abnormalities (e.g., heartbeat packet drops) through error handling.

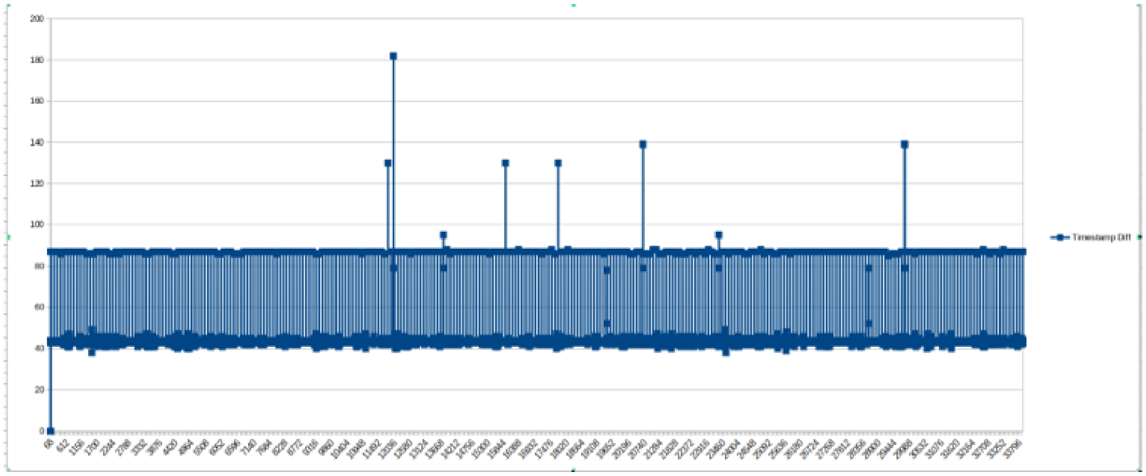


Figure 4.3: Output Data

Chapter 5

Summary of Work

During my internship at Unbox Robotics, I successfully contributed to developing a PCB Testing Tool, leveraging React (Vite.js), FastAPI, WebSocket communication, and C++ middleware to automate PCB testing, reducing testing time by over 90%. I also implemented UART communication in C++ for sensor integration, achieving reliable data transfer with minimal CPU usage. These projects enhanced my technical expertise in fullstack development, embedded systems, and automation, while fostering collaboration and documentation skills. The internship provided valuable industry experience, aligning with Unbox Robotics' mission to deliver efficient robotic solutions.

Appendices

Appendix A

Company Details

Unbox Robotics is a leading automation machinery manufacturing technology company, specializing in robotics-based fulfillment and distribution technology for small to large e-commerce, retail and logistics enterprises. Unbox Robotics' cutting edge technology solutions accelerates the parcel sortition and order fulfillment to facilitate efficient express logistics operations delivering seamless and customer experience.

Mission : -

Our mission is to empower our customers to deliver the most efficient order fulfillment experience using future-ready robotic systems

Vision : -

We envision to be the world's go to robotics technology & software company with efficient & revolutionary logistics automation solutions which deliver impact and enrich lives of our customers, partners and end consumers

Appendix B

Industry Mentor Details

Field	Details
Name	Unbox Robotics Private Limited
CIN No.	U72900PN2019PTC232507
Industry	Automation Machinery Manufacturing
Name of Mentor	Mr.Ujjval Pamnani
Designation	Associate Director (Software)
Email ID	ujjval.pamnani@unboxrobotics.com
Internship Duration	15 January 2025 to 15 July 2025 (6 months)