FOSSEE Winter Internship Report

On

Industrial Engineering Research Projects

Submitted by Sanjeev Krishna S

3rd Year, Department of Computer Science and Engineering

Albertian Institute of Science and Technology, Ernakulam, Kerala

Under the Guidance of

Prof. Jayendran Venkateswaran

Department of IEOR

Indian Institute of Technology Bombay

February 2, 2026

Acknowledgments

I would like to express my sincere gratitude to everyone who supported and guided me throughout the course of my internship and the successful completion of this project. Their encouragement and guidance were instrumental in shaping my learning experience.

I am thankful to a friend and college peer who informed me about this internship opportunity and motivated me to approach the screening tasks with dedication and attention to detail, which helped me secure this internship.

I would like to acknowledge the FOSSEE Team for organizing a technically enriching and well-structured internship experience. I express my sincere thanks to **Prof. Jayendran Venkateswaran, IEOR Department, IIT Bombay**, for overseeing the technical aspects of the work and for his continuous guidance and encouragement. I am also grateful to the mentors from the department, including the M.Tech students, for their valuable technical support and constructive feedback throughout the project.

I extend my gratitude to **Prof. Prabhu Ramachandran**, Principal Investigator of the **FOSSEE project**, IIT Bombay for his leadership and support. I also express my sincere thanks to **Prof. Kannan M. Moudgalya**, former Principal Investigator, for his visionary guidance in establishing this platform and enabling this internship opportunity. I am especially thankful to my **FOSSEE guides, Prof. Sumanto Kar and Mr. Varad Patil** for coordinating all official and administrative aspects of the internship and for ensuring smooth communication during its course. I also thank the **FOSSEE Managers Ms. Usha Viswanathan and Ms. Vineeta Parmar** and their entire team for their consistent support.

I gratefully acknowledge the support of the **National Mission on Education through Information and Communication Technology (ICT)**, **Ministry of Education (MoE), Government of India**, for facilitating this project.

I would also like to thank my fellow interns and colleagues for their cooperation and shared learning experiences. Finally, I express my sincere gratitude to my **college, the Department of Computer Science, Head of the Department, and Principal** for their continued support throughout my studies.

Contents

Chapter 1

Introduction

1.1 National Mission on Education through ICT

The National Mission on Education through ICT (NMEICT) is a scheme under the Department of Higher Education, Ministry of Education, Government of India. It aims to leverage the potential of ICT to enhance teaching and learning in Higher Education Institutions in an anytime-anywhere mode.

The mission aligns with the three cardinal principles of the Education Policy—**access, equity, and quality**—by:

- Providing connectivity and affordable access devices for learners and institutions.
- Generating high-quality e-content free of cost.

NMEICT seeks to bridge the digital divide by empowering learners and teachers in urban and rural areas, fostering inclusivity in the knowledge economy. Key focus areas include:

- Development of e-learning pedagogies and virtual laboratories.
- Online testing, certification, and mentorship through accessible platforms like EduSAT and DTH.
- Training and empowering teachers to adopt ICT-based teaching methods.

For further details, visit the official website: `www.nmeict.ac.in.`

1.1.1 ICT Initiatives of MoE

The Ministry of Education (MoE) has launched several ICT initiatives aimed at students, researchers, and institutions. The table below summarizes the key details:

Table 1.1: Summary of ICT Initiatives by the Ministry of Education

No.	Resource	For Students/Researchers	For Institutions
Audio-Video e-content			

No.	Resource	For Students/Researchers	For Institutions
1	SWAYAM	Earn credit via online courses	Develop and host courses; accept credits
2	SWAYAMPRABHA	Access 24x7 TV programs	Enable SWAYAMPRABHA viewing facilities
Digital Content Access			
3	National Digital Library	Access e-content in multiple disciplines	List e-content; form NDL Clubs
4	e-PG Pathshala	Access free books and e-content	Host e-books
5	Shodhganga	Access Indian research theses	List institutional theses
6	e-ShodhSindhu	Access full-text e-resources	Access e-resources for institutions
Hands-on Learning			
7	e-Yantra	Hands-on embedded systems training	Create e-Yantra labs with IIT Bombay
8	FOSSEE	Volunteer for open-source software	Run labs with open-source software
9	Spoken Tutorial	Learn IT skills via tutorials	Provide self-learning IT content
10	Virtual Labs	Perform online experiments	Develop curriculum-based experiments
E-Governance			
11	SAMARTH ERP	Manage student lifecycle digitally	Enable institutional e-governance
Tracking and Research Tools			
12	VIDWAN	Register and access experts	Monitor faculty research outcomes
13	Shodh Shuddhi	Ensure plagiarism-free work	Improve research quality and reputation
14	Academic Bank of Credits	Store and transfer credits	Facilitate credit redemption

1.2 FOSSEE Project

The FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools in academia and research. It is part of the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education (MoE), Government of India.

1.2.1 Projects and Activities

The FOSSEE Project supports the use of various FLOSS tools to enhance education and research. Key activities include:

- **Textbook Companion**: Porting solved examples from textbooks using FLOSS.
- **Lab Migration**: Facilitating the migration of proprietary labs to FLOSS alternatives.
- **Niche Software Activities**: Specialized activities to promote niche software tools.
- **Forums**: Providing a collaborative space for users.
- **Workshops and Conferences**: Organizing events to train and inform users.

Figure 1.1: FOSSEE Projects and Activities

1.2.2 Fellowships

FOSSEE offers various internship and fellowship opportunities for students:

- Winter Internship

- Summer Fellowship
- Semester-Long Internship

Students from any degree and academic stage can apply for these internships. Selection is based on the completion of screening tasks involving programming, scientific computing, or data collection that benefit the FLOSS community. These tasks are designed to be completed within a week.

For more details, visit the official FOSSEE website.

1.3 Osdag Software

Osdag (Open steel design and graphics) is a cross-platform, free/libre and open-source software designed for the detailing and design of steel structures based on the Indian Standard IS 800:2007. It allows users to design steel connections, members, and systems through an interactive graphical user interface (GUI) and provides 3D visualizations of designed components. The software enables easy export of CAD models to drafting tools for construction/fabrication drawings, with optimized designs following industry best practices. Built on Python and several Python-based FLOSS tools (e.g., PyQt and PythonOCC), Osdag is licensed under the GNU Lesser General Public License (LGPL) Version 3.

1.3.1 Osdag GUI

The Osdag GUI is designed to be user-friendly and interactive. It consists of:

- **Input Dock**: Collects and validates user inputs.
- **Output Dock**: Displays design results after validation.
- **CAD Window**: Displays the 3D CAD model, where users can pan, zoom, and rotate the design.
- **Message Log**: Shows errors, warnings, and suggestions based on design checks.

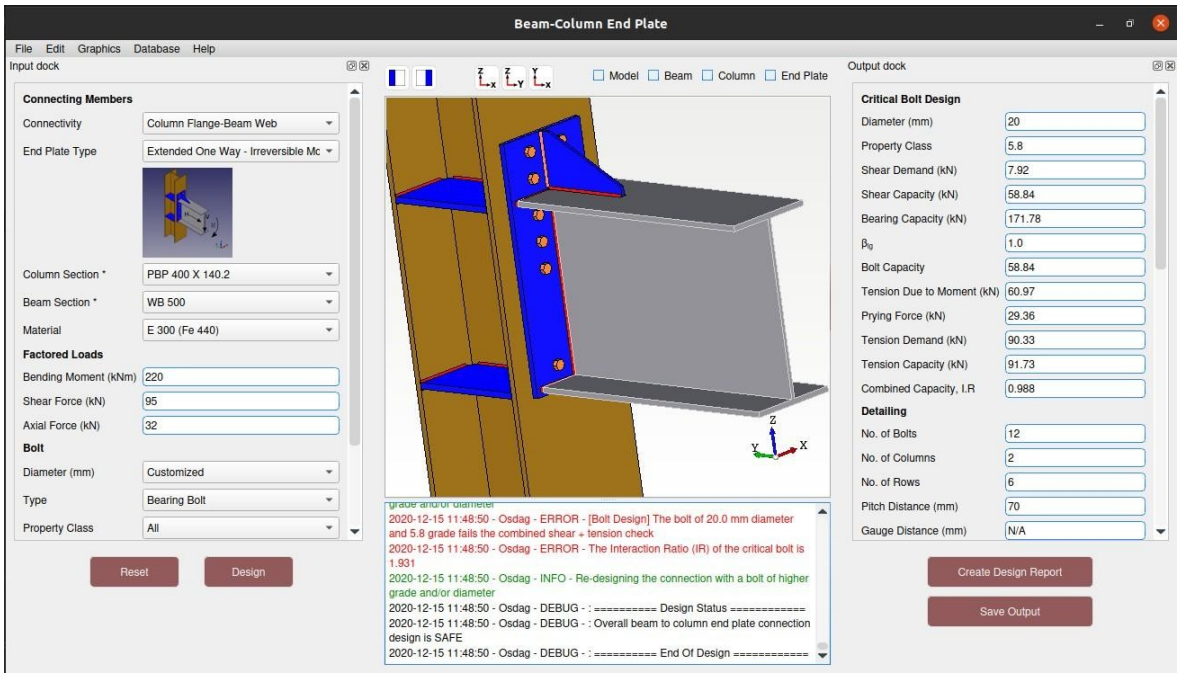

Figure 1.2: Osdag GUI

1.3.2 Features

- **CAD Model**: The 3D CAD model is color-coded and can be saved in multiple formats such as IGS, STL, and STEP.
- **Design Preferences**: Customizes the design process, with advanced users able to set preferences for bolts, welds, and detailing.
- **Design Report**: Creates a detailed report in PDF format, summarizing all checks, calculations, and design details, including any discrepancies.

For more details, visit the official Osdag website.

Chapter 2

Internship Task 1: Pick-by-Light System Demonstration with Colour Detection

2.1 Problem Statement

Pick-by-Light systems are used in industrial environments to guide operators during component picking by displaying required information for a selected product variant. Implementing a full industrial Pick-by-Light system involves complex infrastructure, which is not feasible within the scope of an internship project.

This project presents a **demonstration model** that showcases the basic working principle of a Pick-by-Light system. Variant selection is simulated using a colour sensor, and the corresponding component information is displayed using digital displays.

2.2 Tasks Done

2.2.1 Study and Requirement Understanding

- Studied the working principle of Pick-by-Light systems used in industrial environments.
- Identified the core elements required for demonstrating variant selection and component display.

2.2.2 System Design and Planning

- Designed a demonstration-oriented embedded system architecture.
- Selected appropriate hardware components such as the color sensor, shift register, and displays.

2.2.3 Sensor Interfacing and Input Handling

- Interfaced the TCS34725 color sensor with the Arduino using I^2C communication.

- Acquired raw RGB and clear light data from the sensor.
- Implemented normalization techniques to reduce the effect of ambient lighting.

2.2.4 Variant Simulation Logic

- Developed logic to interpret sensor input as a simulated variant selection.
- Implemented ratio-based color comparison for reliable input differentiation.
- Applied hysteresis logic to maintain stable detection during minor variations.

2.2.5 Display Interfacing and Output Control

- Interfaced two 7-segment displays using a 74HC595 shift register.
- Implemented daisy chaining to control multiple displays using minimal microcontroller pins.
- Displayed predefined component values corresponding to the selected variant.

2.2.6 Testing and Validation

- Tested the system under different input conditions.
- Verified sensor readings and display output using serial monitoring.
- Ensured consistent and stable operation of the demonstration setup.

2.3 Code Explanation

The Arduino code interfaces with the TCS34725 color sensor to read raw RGB values from an object placed in front of it. These values are normalized to compensate for ambient lighting, and ratio-based logic is applied to identify the dominant color, simulating a product variant. Hysteresis is used to maintain stable detection and prevent rapid changes. Once a color (variant) is detected, the corresponding component values are displayed on two 7-segment displays using a 74HC595 shift register, with daisy chaining employed to control multiple displays efficiently while minimizing microcontroller pin usage. Serial output is also provided for monitoring and debugging.

2.4 Learning Outcomes

- Gained understanding of the Pick-by-Light concept through a practical demonstration.
- Learned to interface and process data from a color sensor in an embedded system.
- Developed decision logic for input interpretation and output control.
- Understood the use of shift registers for expanding output capability.

- Learned the concept and practical implementation of daisy chaining.
- Improved skills in embedded system design, testing, and debugging.

Chapter 3

Internship Task 2: Industry Ordering System – ConfigFlow

3.1 Introduction

The Industry Ordering System – ConfigFlow is a Flutter and Firebase–based application developed to manage industrial product configurations and production workflows. The system replaces informal and error-prone ordering practices with a structured, verifiable pipeline that ensures data integrity from product selection to final production release.

The design philosophy follows an established industrial principle: products are configured, drafts are refined, and only validated specifications are committed to production. This approach minimizes accidental production triggers while maintaining a complete digital audit trail.

3.2 High-Level System Overview

The application enforces a linear workflow consisting of three distinct operational stages:

1. **Product Catalog**: Allows users to browse available products and initiate configurations.
2. **Configuration Hub (Drafts)**: A staging area where configurations are created, modified, and reviewed prior to confirmation.
3. **Production Floor**: Displays confirmed orders released into the manufacturing queue.

Firebase Realtime Database enables instantaneous synchronization across all clients, ensuring that draft states and production orders remain consistent in real time.

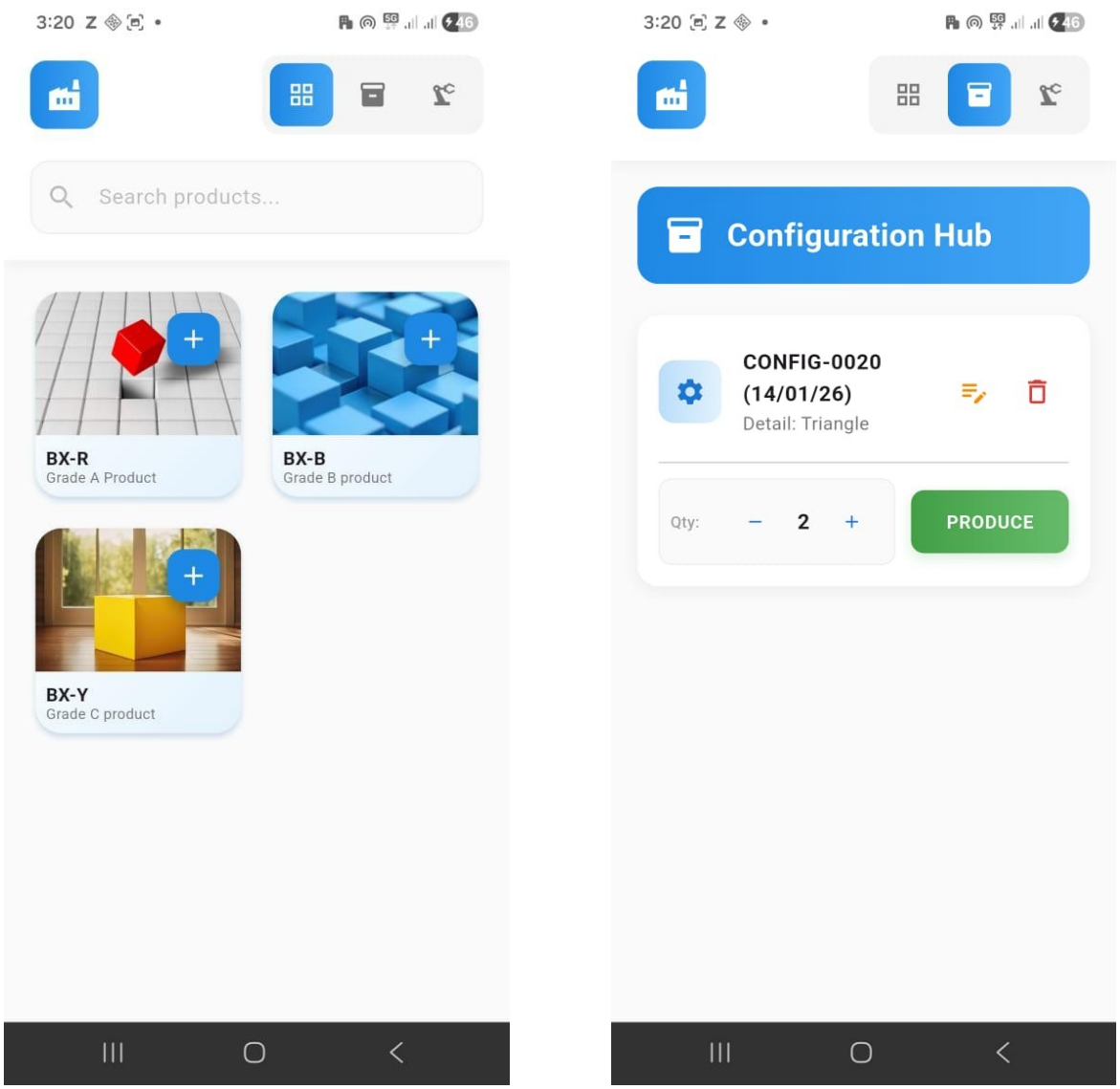

Figure 3.1: Product Catalog and Configuration Hub

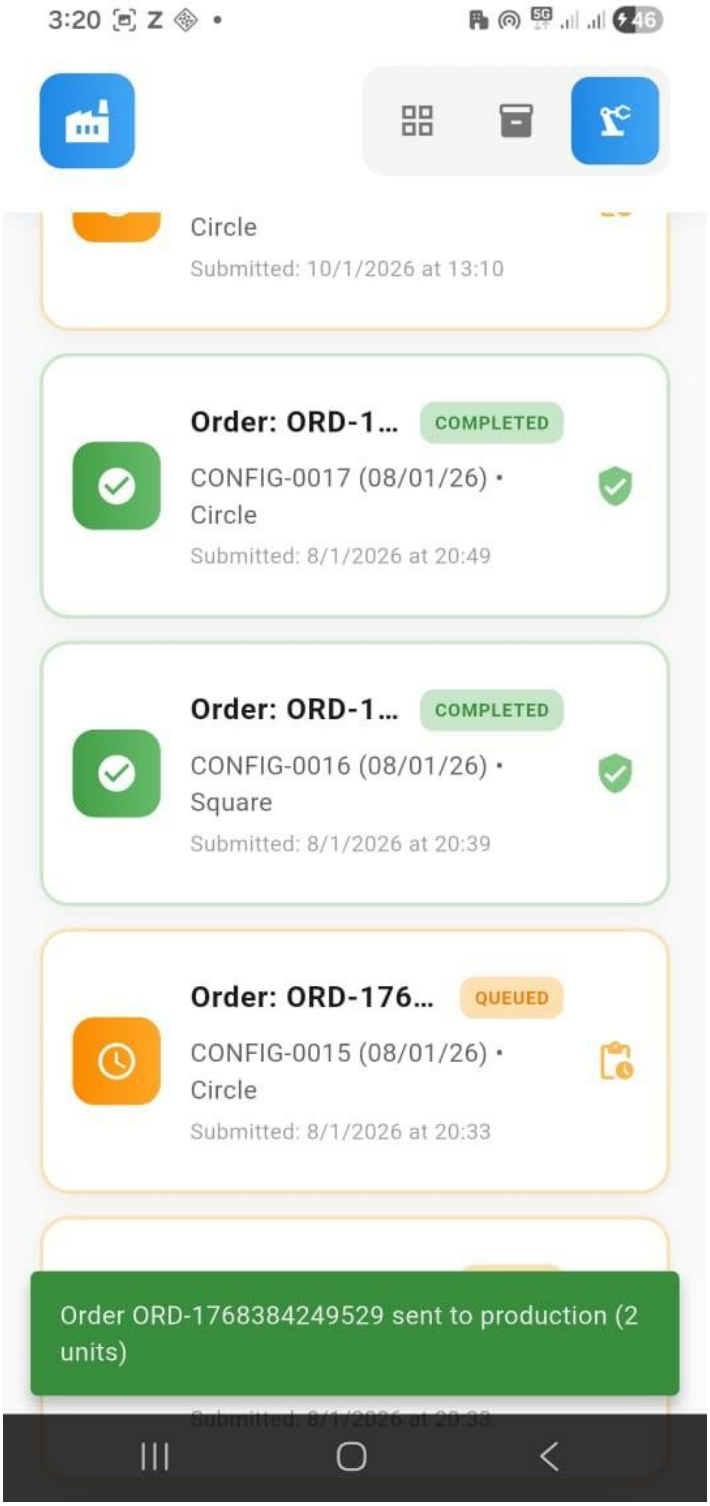

Figure 3.2: Production Floor: Confirmed Orders

3.3 Application Architecture

The application is built using a modular Flutter architecture that separates initialization, workflow control, navigation, and data modeling to improve maintainability and scalability.

3.4 Flutter Project Structure

The `lib/` directory contains all core application logic and is organized to clearly distinguish between entry points, workflow controllers, and data models.

3.4.1 Application Initialization and Core Setup

- **main.dart**: Serves as the primary entry point of the application. It initializes Flutter bindings, configures Firebase, and launches the root widget.
- **app.dart**: Defines the root application widget and configures `MaterialApp.router`, global theming, and navigation behavior.
- **firebase_options.dart**: Auto-generated using the FlutterFire CLI, this file contains platform-specific Firebase configuration details required for initializing backend services.

3.4.2 Core Workflow Controller

`lib/screens/MainScreen.dart`: Acts as the central controller of the application workflow. It manages:

- Navigation between the Product Catalog, Configuration Hub, and Production Floor.
- Active Firebase listeners and real-time data streams.
- Persistent navigation elements.
- State preservation using an IndexedStack to prevent unnecessary widget rebuilds.

3.4.3 Data Models

The `models/` directory defines structured representations of all entities stored in Firebase Realtime Database, ensuring consistent data handling across the application.

- **product.dart**: Defines product attributes used in the catalog, including identifiers, configuration parameters, and metadata.
- **configurations.dart**: Represents draft configurations created by users, capturing customized specifications, quantities, and workflow state flags.
- **production_orders.dart**: Defines immutable production orders generated from confirmed configurations. Each order stores a snapshot of configuration data along with server-generated timestamps.
- **users.dart**: Represents application users and is designed to support role-based access control in future enhancements.
- **customers.dart**: Stores customer-related data linked to configurations and production orders, enabling traceability.

3.5 Data Flow and Persistence

The system follows a strict hierarchical data transformation model:

$$Product \rightarrow Configuration(Draft) \rightarrow ProductionOrder$$

- Product Data Path: `/products`
- Configuration Data Path: `/configurations`
- Production Order Path: `/production_orders`

Server-generated timestamps (`ServerValue.timestamp`) are used throughout the system to ensure consistency across devices.

3.6 Functional Workflows

3.6.1 Configuration Logic

Product configuration is handled through a contextual overlay, allowing users to define custom parameters without navigating away from the catalog. This design minimizes workflow disruption and improves efficiency.

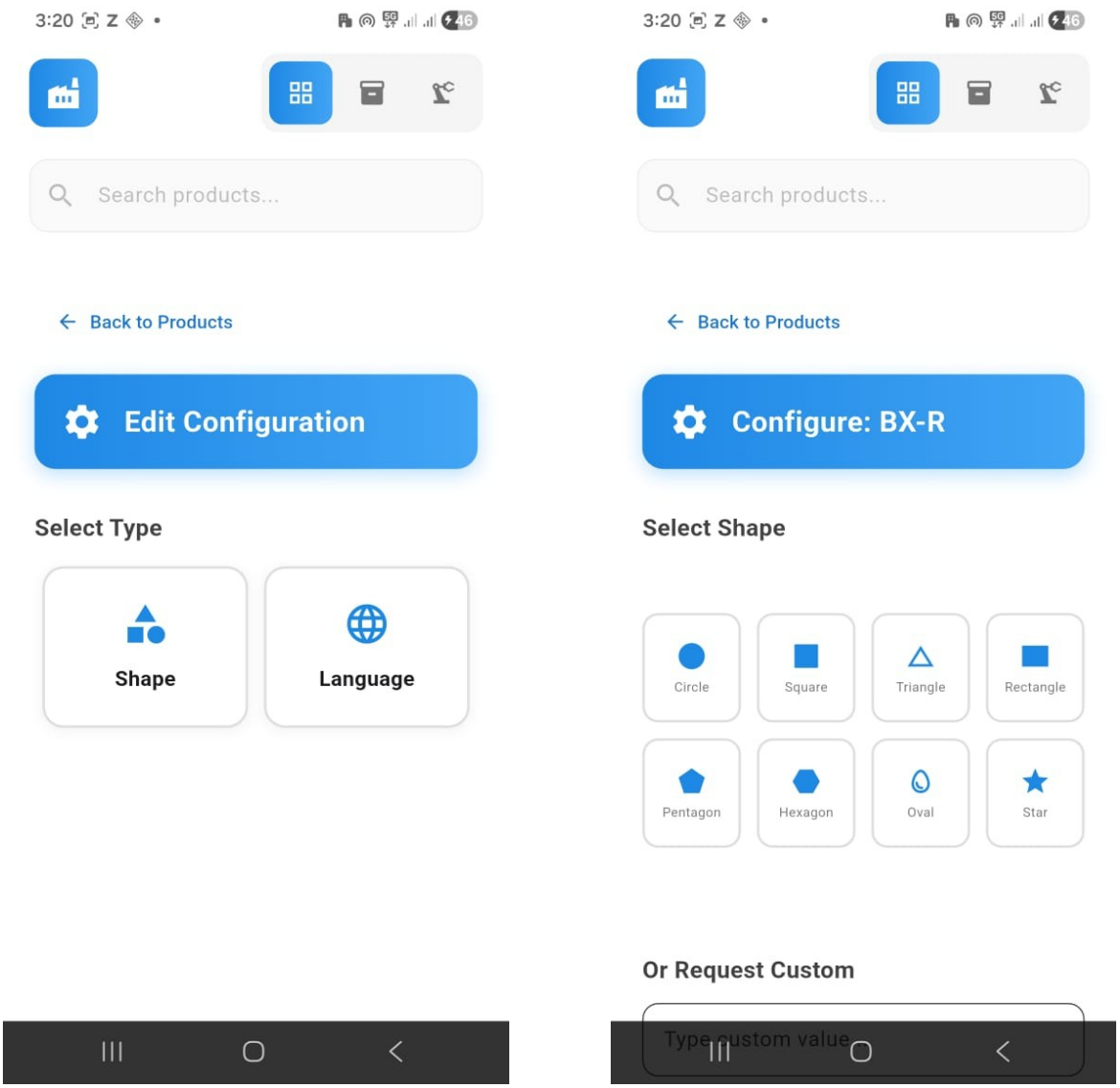

Figure 3.3: Configuration Workflow

3.6.2 Persistent Command Header

A fixed command header provides:

- Workflow navigation.
- Product search and filtering.
- Visual indication of the active workflow stage.

Figure 3.4: Persistent Command Header

3.7 Backend Infrastructure

- **Backend Service**: Firebase Realtime Database.
- **Synchronization**: Real-time data streams for low-latency updates.
- **Safety Measures**: Explicit state transitions and confirmation dialogs to prevent accidental production release.

3.8 Future Scope

The modular design allows for future expansion, including:

- Role-based access control using Firebase Authentication.
- Extended production states such as *Quality Check* and *Shipped*.
- Analytics dashboards.
- Migration to Firestore or SQL-based backends if required.

3.9 System Integration: Dobot Robotic Arms

The application extends its functionality to the physical manufacturing layer by connecting to **Dobot robotic arms**, which serve as the primary actuators for the Industry 4.0 manufacturing setup.

Integration is achieved via a dedicated **Python-based server** that interfaces with the Dobot API. The workflow operates as follows:

- The Flutter app pushes order specifications to the **Firebase Realtime Database**.
- The Python server, acting as a listener, retrieves new orders in real-time.
- The Dobot server processes the data, manages the execution queue, and commands the robotic arm to perform the manufacturing tasks.
- **Status Synchronization**: As the robotic arm completes various stages of the task, the Python server **updates the status field in Firebase as completed**.

- **Live Queue Updates**: These database changes are reflected instantly on the **Production Floor** screen, providing operators with a real-time view of the manufacturing progress.

3.10 Conclusion

IndustryOS ConfigFlow provides a structured and reliable solution for managing industrial product configurations and production workflows. By separating configuration drafts from finalized production orders and enforcing confirmation-based state transitions, the system minimizes errors and ensures data integrity.

With real-time synchronization through Firebase and immutable production records, the platform offers traceability and operational clarity. Its modular Flutter-based architecture makes the system scalable and suitable for real-world industrial deployment.

Chapter 4

Internship Task 3: Laser-Based Pick Guidance (Abandoned Prototype)

4.1 Overview

This project explored a laser-guided pick-by-location system using an ESP32 microcontroller to dynamically point a laser at target storage bins during order picking. The goal was to evaluate whether a servo-mounted laser pointer, controlled wirelessly by a mobile app, could replace traditional pick-by-light systems.

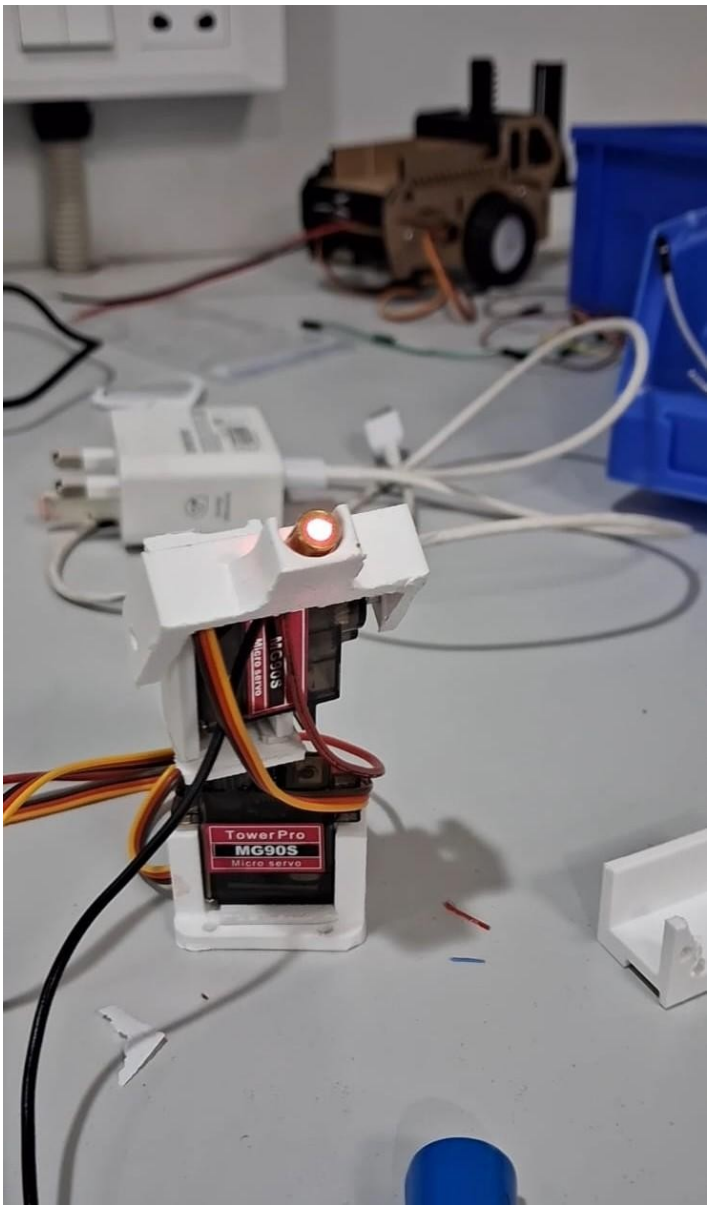

Figure 4.1: Laser-Based Pick Guidance Prototype

4.2 Intended Architecture

- **Frontend**: Flutter mobile app (variant selection & sequencing).
- **Controller**: ESP32.
- **Actuation**: Servo motors with laser diodes.
- **Guidance Method**: Laser physically points to target bin.
- **Communication**: Local Wi-Fi (HTTP commands).

4.3 What Was Implemented / Planned

- ESP32-based Wi-Fi command handling.

- Servo motor control to rotate laser to specific bin positions.
- Laser on/off control via GPIO.
- App-driven pick sequencing logic.
- Initial calibration mapping between servo angles and bin positions.
- **Planned**: Camera-based angle calibration to automate servo positioning and reduce human error.

4.4 Why the Approach Was Abandoned

The system failed at the **physical interaction layer**, not the software layer:

1. **Servo Precision Limits**:
   - Hobby servos could not reliably point the laser to exact bin positions.
   - Small drift and backlash caused frequent misses.
2. **Calibration Fragility**:
   - Even with a camera module, environmental variations (lighting, reflections) complicated reliable calibration.
   - Manual adjustments were still required.
3. **Latency and Jitter**:
   - Wi-Fi command latency affected smooth laser movement.
   - Timing inconsistencies reduced operator trust.
4. **Poor Scalability**:
   - One laser could not efficiently service many bins.
   - Maintenance effort exceeded potential benefit.

4.5 Key Engineering Insight

Even with advanced sensors (ESP32 Camera Module) and a capable controller, **mechanical pointing systems are inherently fragile** in industrial workflows. Reliable picking systems favor:

- Static indicators (LEDs).
- Sensor-based confirmation.
- Minimal moving parts.

This insight directly motivated the pivot toward **LED- and sensor-driven pick confirmation systems**.

4.6 Skills Demonstrated

- ESP32 Wi-Fi & GPIO control.
- Servo motor control.
- Hardware/software co-design.
- Camera-based feedback planning.
- Real-world calibration modeling.
- Human-safety and industrial compliance considerations.
- Recognizing and abandoning non-scalable designs early.

4.7 Project Status

Status: Abandoned after feasibility validation. Maintained as a documented learning and evaluation project.

4.8 References

This prototype was inspired by existing industrial **pick-by-light** and **pick-by-vision** systems used in warehouse and manufacturing environments.

Chapter 5

Internship Task 4: PulseLine – Sensor-Verified Pick-to-Light System

5.1 Introduction

Modern industrial assembly and sorting workflows demand high accuracy, repeatability, and minimal human error. Traditional manual confirmation systems rely heavily on operator input, which introduces the possibility of silent failures and incorrect picks.

This project implements a **Pick-to-Light system** that tightly integrates a **Flutter-based control application** with **physical hardware (Arduino R4 WiFi / ESP32)** to ensure that every pick action is **physically verified** using sensors before the workflow advances. The system replaces subjective confirmation with objective, hardware-validated truth.

5.2 System Overview

The Pick-to-Light system is designed around a **closed-loop verification philosophy**, where software instructions are enforced and validated by physical sensors. The system consists of two tightly coupled layers:

1. **Flutter Application (Control Layer)**
2. **Arduino R4 WiFi Hardware Controller (Execution Layer)**

Communication between the two layers is achieved using a **low-latency HTTP protocol over a local Wi-Fi network**.

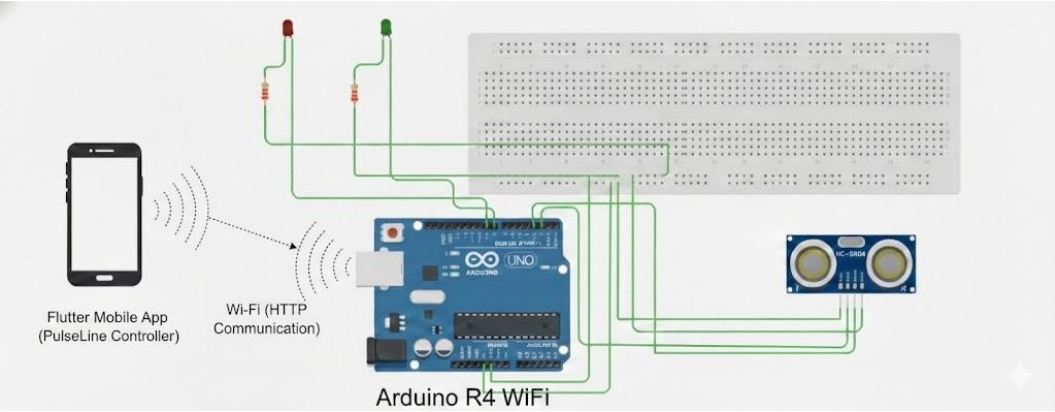

Figure 5.1: PulseLine System Architecture and Wiring

5.3 Software Architecture (Flutter Application)

5.3.1 Role of the Flutter Application

The Flutter application acts as the **central orchestration unit**, responsible for:

- Managing the sequence of pick steps.
- Handling product variants and quantities.
- Issuing commands to activate physical bins.
- Polling hardware status to verify task completion.
- Automatically advancing the workflow upon confirmation.

The application ensures that operators are guided step-by-step, while final confirmation is delegated to hardware sensors.

5.3.2 Workflow

- **Variant Selection**: The operator selects a specific product variant before initiating the pick process. This determines which bins are involved.
- **Dynamic Step Sequencing**: Based on the selected variant, the application computes a dynamic pick path (e.g., Box 1 → Box 3 → Box 2).
- **Quantity Enforcement**: If multiple items are required from the same bin, the application waits for multiple verified sensor triggers before proceeding.
- **Automatic Advancement**: Upon receiving confirmation from the hardware controller, the UI automatically advances to the next step without manual input.

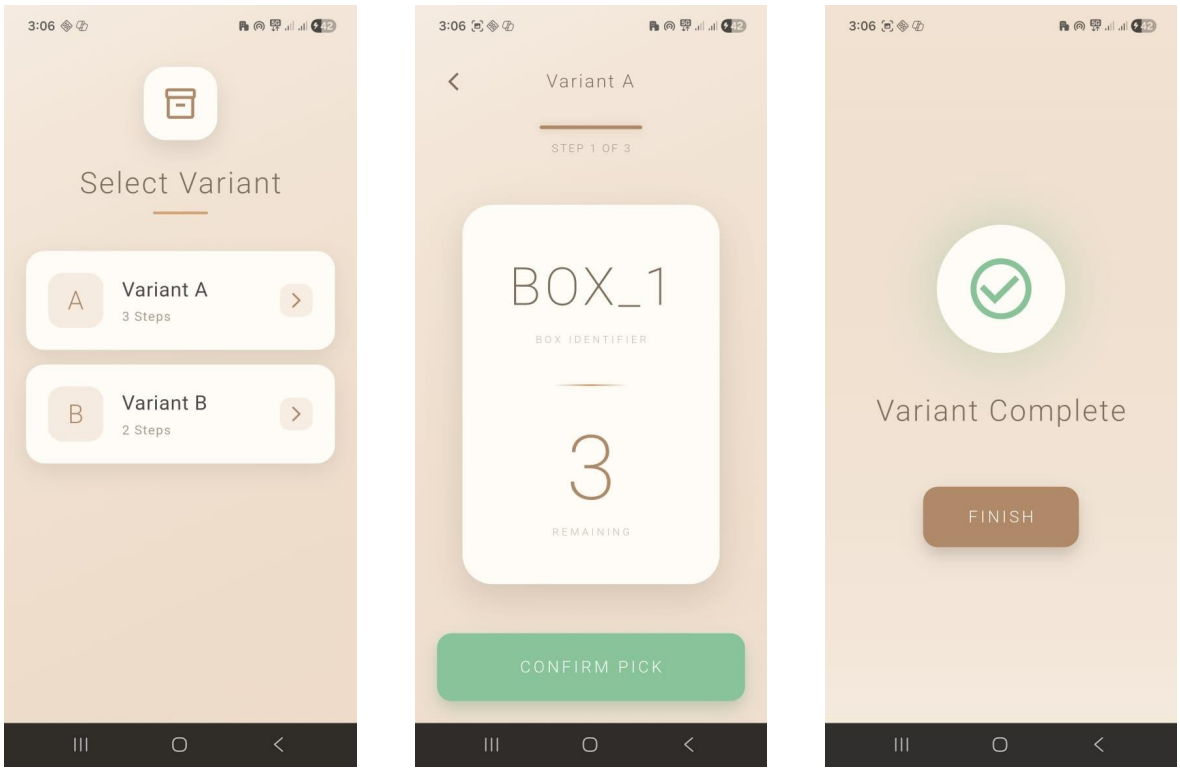

Figure 5.2: PulseLine App Interface: Variant Selection and Picking Steps

5.4 Hardware Architecture (Arduino R4 WiFi)

5.4.1 Hardware Controller Responsibilities

The Arduino R4 WiFi serves as the **physical enforcer of system rules**, ensuring that actions occur only in authorized zones. Its responsibilities include:

- Driving LED indicators for each bin.
- Monitoring ultrasonic sensors for hand presence.
- Detecting unauthorized access to inactive bins.
- Triggering audible alarms for violations.
- Hosting an HTTP server for communication with Flutter.

5.4.2 Physical Components

The system uses the following hardware:

- **Microcontroller**: Arduino R4 WiFi
- **Sensors**: HC-SR04 Ultrasonic Sensors
- **Indicators**: Dual LEDs (Red / Green) per box
- **Alert System**: Buzzer
- **Communication**: WiFi (HTTP Server on Port 80)

5.4.3 Pin Configuration

Component	Box 1	Box 2	Box 3
TRIG Pin	2	4	6
ECHO Pin	3	5	7
Red LED	9	12	13
Green LED	8	11	A0
Distance Threshold	10 cm	6 cm	6 cm

Table 5.1: PulseLine Hardware Pin Configuration

Buzzer Pin: 10

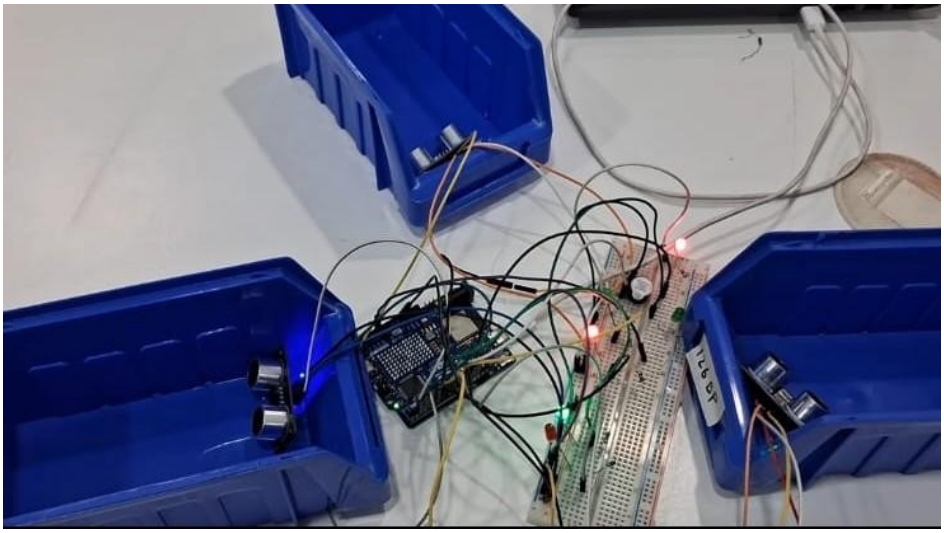

Figure 5.3: Physical Hardware Setup of PulseLine System

5.5 Sensor-Based Verification Logic

5.5.1 Active Box Concept

At any given time, only **one box is marked active** by the system. The active box is indicated using a **Green LED**, while all inactive boxes remain **Red**. The variable `activeBox` in the controller firmware determines the currently authorized bin.

5.5.2 Police Logic (Unauthorized Access Detection)

To prevent incorrect picks, the system implements a **Police Logic** mechanism:

- If a hand enters a **non-active box**, the buzzer is triggered.
- This ensures that mistakes are immediately noticeable.
- The alarm is rate-limited to prevent continuous noise.

This design prevents silent errors and enforces strict pick discipline.

5.6 Manual Override vs Sensor Automation

5.6.1 Manual Confirmation Button (Fail-Safe)

The Flutter application includes a *Confirm Pick* button, intended strictly as a fail-safe. It is used only under special conditions such as:

- Sensor obstruction or failure.
- Hardware maintenance.
- Training or testing scenarios.

5.6.2 Sensor-First Automation Philosophy

Under normal operation, the system prioritizes sensor-based confirmation:

- Eliminates unnecessary UI interaction.
- Reduces pick cycle time.
- Ensures physical action actually occurred.
- Removes reliance on subjective operator input.

This results in a zero-touch confirmation workflow.

5.7 Communication Protocol (HTTP API)

The Flutter application communicates with the hardware controller using HTTP GET requests.

5.7.1 Box Activation

Request: `GET http://<controller-ip>/?box_id=BOX_1`
Effect: Activates the specified box (Green LED ON).

5.7.2 System Deactivation

Request: `GET http://<controller-ip>/?box_id=OFF`
Effect: Deactivates all boxes (All LEDs Red).

5.7.3 Status Polling

Endpoint: `GET http://<controller-ip>/status`
Responses:

- `IDLE`: No pick detected.
- `TRIGGERED`: Sensor threshold breached in active box.

This endpoint is continuously polled by the Flutter application to detect physical confirmation.

5.8 Deployment and Calibration

5.8.1 Network Requirements

- Flutter device and controller must be on the same WLAN.
- Controller IP should be static or DHCP-reserved.

5.8.2 Sensor Calibration

Distance thresholds must be calibrated based on:

- Bin depth.
- Environmental reflections.
- Avoidance of false triggers from container walls.

5.9 Conclusion

This Pick-to-Light system replaces trust-based confirmation with hardware-enforced verification. By combining visual cues, ultrasonic sensing, and automated workflow progression, the system eliminates silent failures and ensures operational accuracy. The final result is a production-ready industrial interface where software does not assume correctness—it waits for physical proof.

Chapter 6

Learning Outcomes

- **Designed and implemented Flutter applications with structured UI flow, state management, and API-based communication.**
- **Gained in-depth experience with embedded systems, including microcontroller programming, sensor interfacing, LED control, and buzzer-based alert mechanisms.**
- **Developed systems engineering thinking by integrating software logic with real-time hardware feedback in a Pick-to-Light environment.**
- **Learned practical aspects of sensor calibration, fault handling, and trial-and-error debugging, addressing real-world constraints such as false triggers and environmental interference.**
- **Understood the importance of hardware-enforced verification and fail-safe design in reducing human error within industrial workflows.**
- **Gained insight into industrial engineering concepts, including shop-floor workflow design, human–machine interaction, and the role of automation in reducing manual errors.**

Chapter 7

Conclusion

The FOSSEE Winter Internship offered a rigorous, hands-on exposure to industrial engineering concepts through the integration of software systems and embedded hardware. The internship progressed from foundational demonstrations, such as the Pick-by-Light system using color detection, to full-stack industrial applications like the Industry Ordering System – ConfigFlow. These projects emphasized structured workflow design, data integrity, and disciplined state transitions, reflecting real-world industrial practices. The exploration and subsequent abandonment of a laser-based pick guidance system further reinforced the importance of feasibility, scalability, and mechanical reliability in industrial system design.

The internship culminated in the development of the PulseLine sensor-verified Pick-to-Light system, which replaced trust-based confirmation with hardware-enforced validation. By tightly coupling a Flutter-based control application with sensor-driven hardware logic, the system ensured that workflow progression depended on physical proof rather than manual input, significantly reducing the possibility of silent errors. Overall, the internship strengthened technical skills in embedded systems, mobile application development, and software–hardware co-design, while instilling a strong appreciation for pragmatic engineering decisions, fail-safe design, and industrial realism.

Bibliography

1. **Ministry of Education, Government of India**, *National Mission on Education through Information and Communication Technology (NMEICT)*. Official Website: https://www.nmeict.ac.in

2. **FOSSEE Project, IIT Bombay**, *Free/Libre and Open Source Software for Education (FOSSEE)*. https://fossee.in

3. **Kannan M. Moudgalya**, *FOSSEE – Promoting Open Source Software for Education.* Indian Institute of Technology Bombay.

4. **Osdag – Open Steel Design and Graphics**, *User Documentation and Software Overview.* https://osdag.fossee.in

5. **Bureau of Indian Standards**, *IS 800:2007 – General Construction in Steel – Code of Practice.* New Delhi, India.

6. **Arduino**, *Arduino Uno R4 WiFi – Hardware Documentation.* https://docs.arduino.cc

7. **Adafruit Industries**, *TCS34725 RGB Color Sensor – Datasheet and Application Notes.* https://www.adafruit.com

8. **Texas Instruments**, *74HC595 8-Bit Shift Register – Datasheet.*

9. **Flutter Development Team**, *Flutter Documentation.* https://docs.flutter.dev

10. **Google Firebase**, *Firebase Realtime Database Documentation.* https://firebase.google.com/docs

11. **Espressif Systems**, *ESP32 Technical Reference Manual.* https://www.espressif.com

12. **HC-SR04 Ultrasonic Sensor**, *Technical Datasheet and Application Guide.*