A Report On

Design and implement a smart, automated system for product labeling and traceability, capable of verifying product quality parameters and applying or validating labels

By

Kaushal Shankar Bobade

of Automation and Robotics

Under The Guidance of

Dr. Swapnil Vyavahare

And

Dr. Kavita Moholkar

JSPM'S RSCOE

CONTENTS

Smart Product Labeling and Traceability System (Simulated using Tinkercad & Python)

1. Abstract

In the era of Industry 4.0, where automation and real-time data tracking are pivotal, product labelling and traceability systems play a crucial role in maintaining production accuracy and compliance. This project presents a simulated smart labelling system that combines serial number assignment, QR code generation, label creation, and Optical Character Recognition (OCR)-based verification, all implemented in Python. A graphical user interface (GUI) allows users to simulate product detection and label printing events. The system ensures that every product is labelled accurately, any mismatch is flagged, and all transactions are logged with time-stamped entries for traceability and auditing. This low-cost simulation prototype demonstrates scalable concepts suitable for smart factories.

2.Introduction

Labelling is one of the final and critical steps in any manufacturing process. Labels contain essential information including product type, batch number, manufacturing date, and unique identifiers such as QR or barcodes. Errors in labelling can lead to product recalls, legal non-compliance, and customer dissatisfaction. Traditional manual labelling methods are time-consuming and prone to human error. To address this, the industry is adopting smart labelling systems that automate label generation and validation, thus enhancing operational efficiency and accuracy. The system developed in this project provides a simulated yet practical solution using open-source tools. It combines automatic data entry, QR generation, OCR-based validation, and GUI controls to simulate real-time industrial operations.

3. Problem Statement

Develop a **simulated smart product labeling and traceability system** that can:

- Automatically assign **serial numbers** to each product within a batch
- Generate **QR-embedded labels** containing product and batch information
- Use **OCR** to verify the correctness of printed labels

- Display **visual alerts** for accepted or rejected labels
- Maintain an **event log** for traceability and quality auditing

The solution should be **cost-effective**, **scalable**, and suitable for educational as well as small-scale industrial deployments.

4. System Architecture

Inputs:

- active_batch.txt stores the current batch ID in use
- Product detection is triggered through a GUI button press (simulated detection)

Processes:

- Generate a **unique device ID** combining the batch ID and a serial number
- Create a **label image** with text and a QR code
- Extract text from the label using **EasyOCR**
- Compare extracted data against the expected values to validate the label

Outputs:

- label_qr.png: generated label with text and QR code
- GUI displays **status lights**: green for PASS, red for REJECT
- traceability_log.csv: log of each event with ID, status, and timestamp
- batch_serials.json: JSON file tracking last serial number used for each batch

5. System Architecture Design

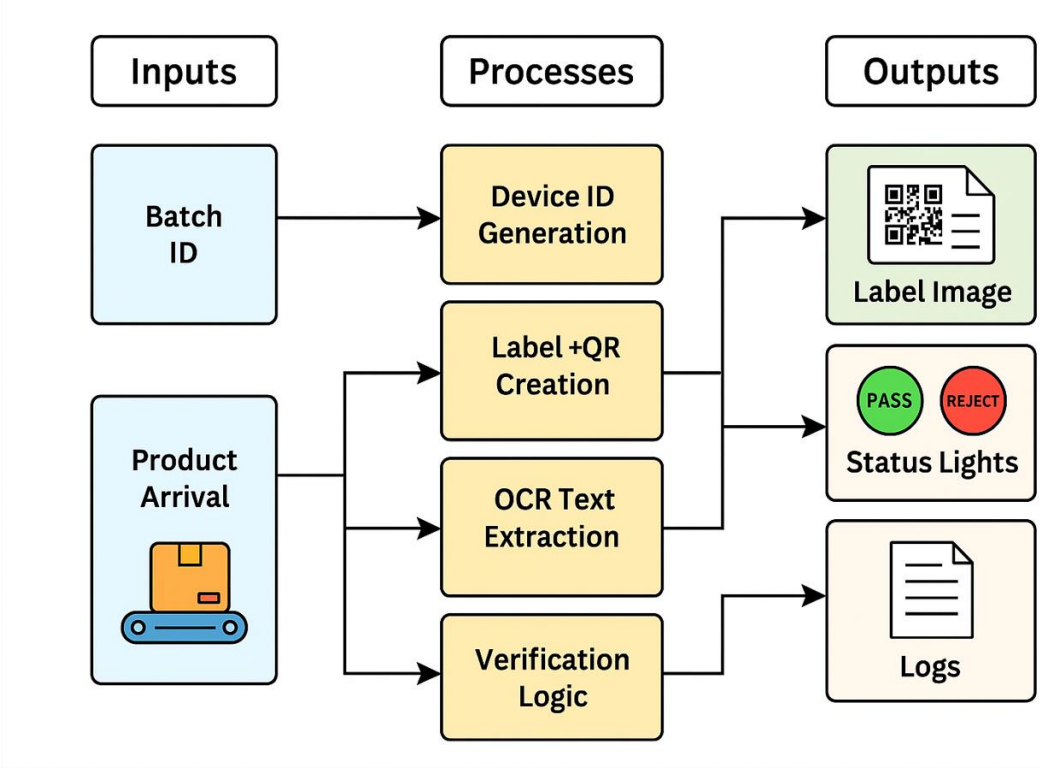

Figure 1: System Architecture

7.Technologies used

- Python 3.10+ : Core programming language used for label generation, decision-making, data logging, and OCR integration.
- Tkinter (GUI Development) : Provides a simple graphical user interface for manual testing and visualization of system actions.
- PIL (Python Imaging Library) : Used for dynamically creating and saving product label images including text and design elements.

- qrcode (QR Code Generation) : Allows automatic creation of QR codes for enhanced product traceability and digital tracking.
- EasyOCR (Optical Character Recognition) : Performs text extraction from label images for automated verification and compliance checks.
- CSV / JSON (Data Handling and Logging) : Stores inspection results, timestamps, and product information for traceability and future analysis.

8. Implementation

a) Tinkercad Simulation

• IR Sensor detects product

• LED turns green if passed, red if failed

• Servo used for rejection mechanism

b) Label Generation (Python)

• Dynamic label with Device ID, Batch ID, Date

c) OCR Module

• EasyOCR reads label image

• Text compared with expected values

d) Logging

• Data stored in CSV with timestamp, status

9. Feature

- Auto Batch-Based Device ID Generation : Each product label is generated with a unique **Device ID** and **Batch Number** to ensure traceability.
- QR Code Embedded with Full Label Details : Dynamically generated QR code contains all essential product information for quick digital scanning and tracking
- OCR-Based Validation of Printed Labels : EasyOCR automatically reads label text and verifies accuracy against expected values, reducing manual inspection errors.
- Graphical User Interface (GUI) with “Simulate Detection” Button : Simple **Tkinter-based GUI** allows users to simulate product detection and trigger label generation, making the system interactive.

- Visual Status Indication : **Label Printer PASS:** Green light indicates that the label is correct and the product is accepted. **Reject Bin FAIL:** Red light indicates failure in label verification, and the product is virtually rejected.
- Logging of Each Label with Status and Timestamp :
- Every label's data and inspection result (PASS/FAIL) is saved with a timestamp into a **CSV file**, enabling full traceability and analysis.

10. GUI Workflow

1. Click "Simulate Detection" : User clicks the simulation button in the **Tkinter-based GUI** to initiate the process.
2. System Generates Label : A new label image is created with **Device ID, Batch Number, Date, and QR Code**.
3. OCR Checks Correctness : The label is read using **EasyOCR**, and the extracted text is validated against the original data.
4. Lights Blink Based on Status : **PASS:** Green light for correct label (accept). **FAIL:** Red light for incorrect label (reject).
5. Record is Logged in traceability_log.csv : Each result (Accepted/Rejected) is automatically saved with a **timestamp** in a **CSV traceability log** for future reference.

11. Result & Analysis

- Successfully Generated and Verified Multiple Batches : Labels were created and validated for multiple simulated batches such as **B2026** and **B2027**, showcasing consistent system performance.
- OCR Correctly Handled Visual Distortions : The system demonstrated resilience by accurately interpreting distorted text (e.g., OCR output **TIOOX** was successfully identified as **T100X**)—highlighting the robustness of the AI verification.
- Realistic GUI Feedback for Demonstration : The system provided **visual light indicators (Green/Red)** within the GUI, closely mimicking physical industrial equipment for inspection feedback.
- Pass/Reject Statistics Logged for All Events : Every label event was recorded in the **traceability_log.csv**, including time, status (Pass/Reject), and label details—supporting future analysis, traceability, and reporting.

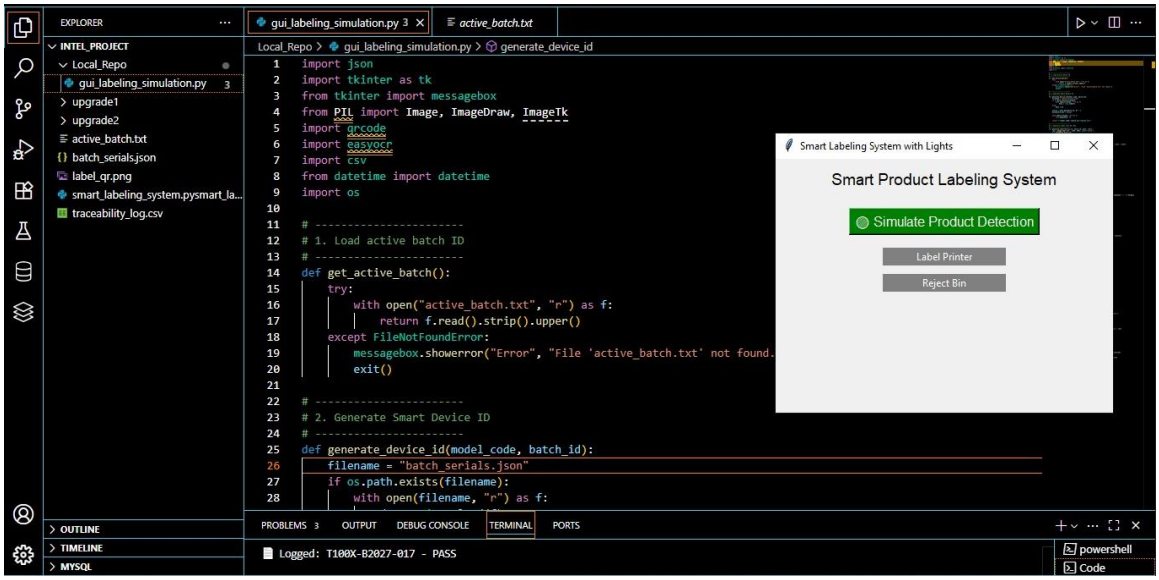

Figure 2 : Demonstration

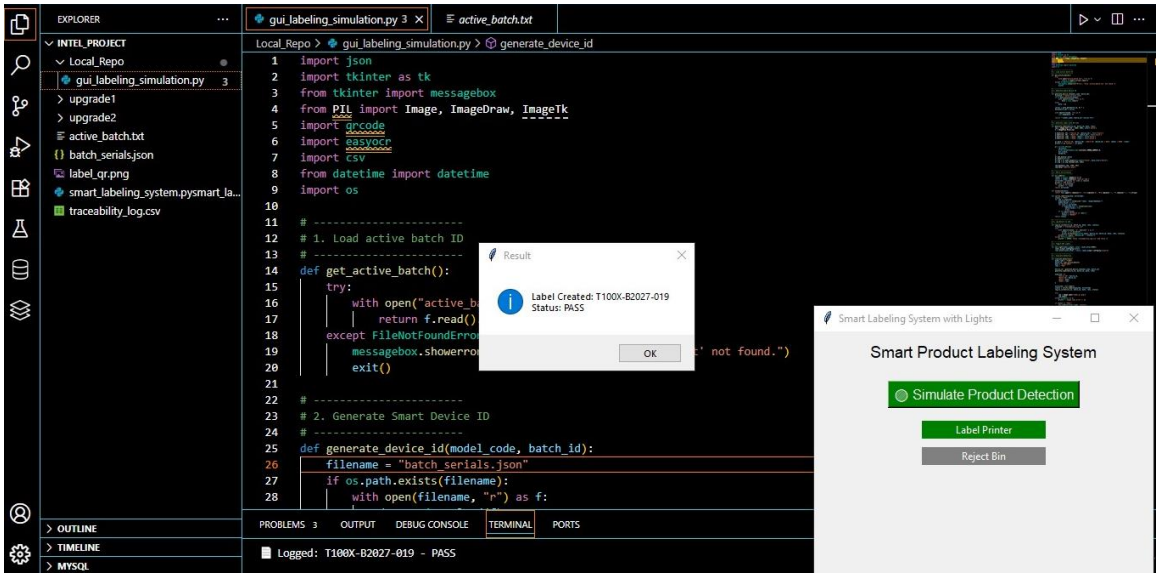

Figure 3 : Label Printing

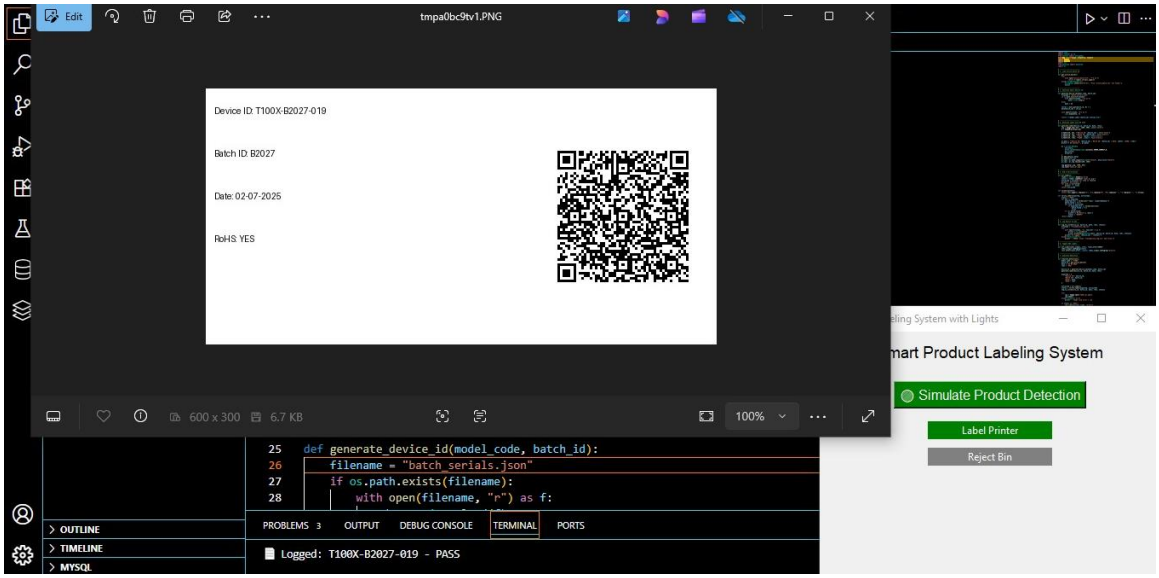

Figure 4 : Label Generation

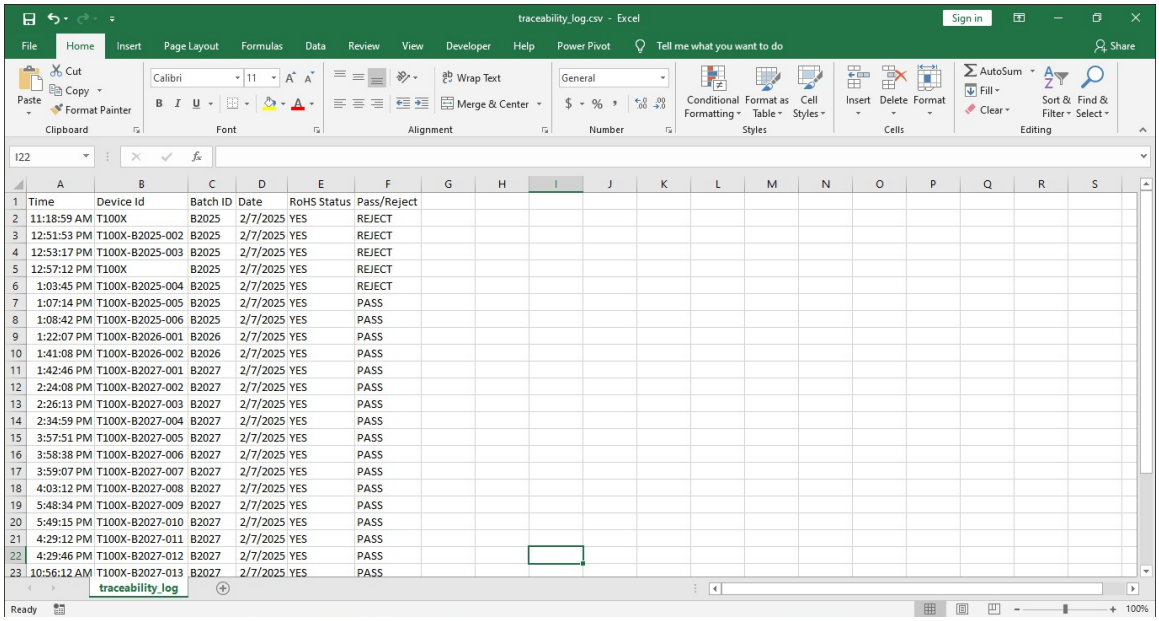

Figure 5 : Data Logging

12. Applications

- Automated Labeling in Smart Factories : The system can be adapted for real-time product labeling and verification on industrial production lines, enhancing efficiency and reducing errors.

- Barcode/QR Verification System : The solution can serve as the foundation for **barcode or QR code verification** in packaging, inventory management, and retail sectors.
- Track-and-Trace in Manufacturing or Logistics : Enables end-to-end **traceability** for components, assemblies, or final products, ensuring quality control, recall readiness, and supply chain visibility.
- Educational Tool for Learning Industry 4.0 Concepts : Serves as a hands-on learning platform for students and professionals to understand **automation, AI integration, traceability, and digital transformation** in modern industries.

13. Advantages

- System can be developed, tested, and demonstrated entirely in software.
- Great for prototyping or educational purposes without costly hardware.
- Automatically generates and verifies labels, minimizing human errors in production.
- Uses Optical Character Recognition to validate label content, ensuring quality control.
- Built entirely in Python with open-source tools—ideal for learning, prototyping, or low-budget deployment
- Maintains a detailed log of all labeled products with timestamps for audit and tracking.

14. Limitations

- OCR validation depends on clear fonts and good resolution.
- Simulated system — no integration with actual hardware like IR sensors or printers.
- No camera input or real conveyor belt integration.

15. Future Scope

- Integration with **Arduino/PLC sensors** for real-time detection.
- Output to **thermal printers** for physical labels.
- **Cloud Dashboard** with remote log access and batch analytics.
- Analytics Module to detect error trends and production bottlenecks.

16. Conclusion

This project demonstrates a **low-cost, scalable, and effective simulation** of a smart labeling and traceability system. Built entirely using Python and open-source libraries, it reflects real-world industrial practices such as QR-based labeling, OCR verification, and traceability logging. The GUI interface ensures usability while maintaining modularity for future hardware integration. With minimal additions, this system could evolve into a fully operational labeling station on a real production line. It serves as a foundational prototype for teaching, testing, and deploying Industry 4.0 concepts in both academic and professional environments.

17. References

- EasyOCR Documentation – Jaided AI.
- QR Code Generation in Python using qrcode Library – PyPI.
- K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*, Prentice-Hall, 1989.