🏭 Industry Use Case: Smart Product Labeling and Traceability System for Quality Control Automation

Contents

---

1. Objective:

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

- **Device ID**
- **Batch ID**
- **Manufacturing Date**
- **RoHS Compliance**
- **Serial Number (QR or barcode)**

The system will leverage:

- **Mechatronic systems** for sensor-based inspection, actuation, and real-time control.
- **AI models** for dynamic verification and validation of label data and product compliance.
- **Automated labeling or data printing mechanisms** (simulated or real).
- **Data logging for traceability** and audit purposes.

---

2. Background:

In modern manufacturing, especially in **electronics, medical devices, and automotive components**, every product must be **individually labeled** with traceable information to comply with **regulatory**, **logistical**, and **quality** standards. This metadata allows tracking through production, warehousing, and customer delivery.

Currently, many factories use semi-automated labeling systems that are error-prone or disconnected from quality checks. Your project simulates the design of a **smart labeling system** that not only labels the product but verifies critical parameters and ensures compliance using **sensor data and intelligent logic**.

---

3. Problem Description:

You are building a **Smart Product Traceability Station** in a manufacturing line for small electronic devices.

System Requirements:

- Each unit arrives on a **conveyor** for inspection and labeling.
- The system must perform the following steps:
    1. **Identify the product** using a camera or sensor (or simulated input).
    2. **Verify compliance**:
        - Check **RoHS** status using part metadata or sensor flags.
        - Confirm **Batch ID** from production input.
        - Match **Device ID** and **Manufacturing Date** via database or simulation logic.
    3. **Apply or validate a label** (with QR code, barcode, or digital record).
    4. **Use actuators** to:
        - Mark or print the label (real or simulated).
        - Reject units with missing or mismatched data.
- AI Integration:
    - Use **OCR (optical character recognition)** or **image recognition** to validate label print quality or content.
    - Use **machine learning** to classify valid/invalid products or spot faulty labels.
- Store all data into a simulated or real-time **traceability log/database**.

---

4. Deliverables:

1. **System Architecture Design**:
   - Block diagram, flowchart, and hardware/software mapping.
2. **Label Inspection Simulation or Hardware**:
   - Simulated camera or sensor readings for product verification.
   - Logic for generating and applying labels.
3. **AI Module**:
   - OCR for label verification, or ML model for visual defect detection.
   - Accuracy and performance report.
4. **Control System Implementation**:
   - Embedded logic for sensor-actuator interaction and data validation.
   - Label printing/rejecting mechanism.
5. **Traceability Database or File Logging**:
   - Store details like DeviceID, BatchID, timestamps, inspection result.
6. **Final Demonstration and Report**:
   - Functional demo (video or live).
   - Documentation covering design, implementation, dataset used, AI training, and results.

---

5. Free Tools & Simulators for Students:

Tool	**Purpose**	**Link**
Tinkercad Circuits	Simulate microcontroller and sensor logic	Tinkercad
Google Colab + OpenCV + EasyOCR	Label inspection using AI/OCR in Python	Colab
LabelImg + YOLOv5 in Colab	Train object detection model for label check	YOLOv5
Python + SQLite/CSV	Simulate traceability database logging	Python Docs
Proteus or Fritzing	Simulate control logic and label activation	Proteus

Tool	Purpose	Link
MATLAB + Simulink	Advanced modeling of label print systems	MATLAB Student

6. Evaluation Criteria:

Criteria	Description	Score
1. Mechatronics System Design	Application of sensors, actuators, mechanical setup, and control logic for automated labeling and inspection. System layout, design clarity, and real-time execution quality.	20
2. Electronics & Embedded Systems	Circuit design, microcontroller programming, interfacing (sensors/actuators), communication protocols (I2C/SPI/UART), and power management.	15
3. AI and Machine Learning Integration	Implementation and training of an OCR/ML model for label verification or defect detection. Evaluation metrics, model integration, and performance.	20
4. Data Logging & Traceability Logic	Accuracy and completeness of logging product metadata (Device ID, Batch ID, etc.), use of databases or structured files, and traceability relevance.	10
5. System Functionality & Testing	Overall operation of the system, including labeling accuracy, real-time rejection, product tracking, and system stability during testing/demo.	10
6. Innovation & Problem-Solving	Creativity in design, optimization techniques, edge computing, use of cloud or dashboard integration, or other features extending the baseline.	10
7. Documentation & Technical Report	Quality of written report (clarity, structure, diagrams), inclusion of results, challenges, and learning outcomes.	5
8. Presentation & Demonstration	Clarity, professionalism, teamwork, and effectiveness of communication during the final presentation/demo.	5
9. Project Management & Teamwork	Effective collaboration, distribution of tasks, timeline adherence, and group coordination.	5

7. Dataset Recommendation for Defect Images

For defect detection, especially in a real-world industrial scenario like product labeling or quality control, using a relevant dataset will help students train AI models effectively. Here are some datasets that can be used for defect detection or quality inspection:

- **The PCB Defect Detection Dataset on Kaggle**:
  - This dataset contains images of PCBs (Printed Circuit Boards) with defects. It is perfect for students to apply computer vision techniques for defect detection.
  - Students can use **OpenCV**, **TensorFlow**, or **PyTorch** to implement machine learning models for detecting defects in the products.
- **Fruits and Vegetables Defect Dataset**:
  - This dataset includes images of fruits and vegetables with various defects such as bruises, rot, etc. It's a good fit for students to apply defect detection models in food industry quality control systems.
- **Steel Defect Dataset**:
  - This dataset contains images of steel sheets with different types of surface defects (e.g., pits, cracks, etc.). It can be used for defect detection in metal manufacturing processes.
- **MVTec Anomaly Detection Dataset**:
  - MVTec provides a comprehensive dataset for industrial object recognition and anomaly detection, with defects such as scratches, dents, and missing parts on industrial objects. This dataset is especially useful for students working with industrial automation in visual inspection tasks.