INTEL UNNATI INDUSTRIAL TRAINING PROGRAMME

(PROJECT REPORT)

Smart Product Labeling and Traceability System for Quality Control Automation

Submitted by:

Team Auto Veritas

George K John

Jeffin I Patrick

Abijith SS

Problem Statement

In PCB manufacturing, relying on manual processes for inspection and labeling can result in frequent human errors, low efficiency, and limited traceability. These shortcomings can compromise product quality and hinder adherence to regulatory standards such as RoHS. To address these limitations, there is a need for an intelligent PCB traceability system that automates identification, performs real-time compliance verification, and utilizes AI for label validation and defect detection. This system should seamlessly integrate with a conveyor-based workflow, using sensors and database lookups to confirm key production parameters such as batch ID, device ID, and manufacturing timestamp. Incorporating AI technologies like image recognition and OCR ensures accurate label reading and validation. Additionally, every inspection and action should be recorded in a centralized database to support compliance audits and traceability, offering a scalable, autonomous solution that enhances production reliability and regulatory alignment.

Solution

The objective is to build an intelligent Product Labelling and Traceability Station capable of automatically identifying each incoming PCB using integrated cameras, sensors, or virtual triggers. The system should validate compliance with environmental standards such as RoHS by utilizing either sensor inputs or embedded data (e.g., QR codes), and cross-check essential identifiers like batch ID, device ID, and manufacturing date with a centralized or simulated database.

To enhance inspection accuracy, machine learning algorithms will be employed to determine whether a PCB is acceptable or faulty, especially in detecting fine-grained defects in labels or prints that may go unnoticed using traditional inspection methods. All compliance results, inspection records, and labeling actions must be consistently logged into a dedicated or simulated traceability database, ensuring full visibility for audits, quality checks, and regulatory oversight.

WORKING

The system begins by using an infrared (IR) sensor to detect the presence of a PCB on the conveyor belt. Once detected, a camera captures an image of the PCB, which is processed using OpenCV. The image is analyzed by a YOLOv8-based deep learning model to identify common defects such as open circuits, shorts, or missing drill

holes. Simultaneously, the system scans the QR code on the PCB to extract key production details, including the device ID, batch number, and timestamp. This information is then verified against a RoHS compliance database. If the PCB is found to be non-compliant or defective, a servo motor is activated to physically remove it from the conveyor path. All inspection results and product information are logged into a database for traceability and future audit purposes.

Github repository link:

https://github.com/Smart-Product-Labelling-Traceability/Smart-Product-Labelling-and-Traceability

Demo Video Link:

https://drive.google.com/file/d/1ZvCEKYGpbcOFqddEUBZn2hIgykIMfiTf/view?usp=sharing

Code

```
import cv2

import time

import sqlite3

import numpy as np

from PIL import Image, ImageDraw, ImageFont
```

```
import RPi.GPIO as GPIO

from pyzbar.pyzbar import decode

from datetime import datetime

from ultralytics import YOLO


# GPIO Setup

GPIO.setmode(GPIO.BCM)


# IR Sensor

IR_PIN = 17

GPIO.setup(IR_PIN, GPIO.IN)


# LEDs

LED_PASS = 20   # example GPIO pin for PASS LED

LED_REJECT = 21 # example GPIO pin for REJECT LED

GPIO.setup(LED_PASS, GPIO.OUT)

GPIO.setup(LED_REJECT, GPIO.OUT)


# Motor Driver (HW-095 / L298N)

MOTOR_IN1 = 23
```

```
MOTOR_IN2 = 24

MOTOR_ENA = 18

GPIO.setup(MOTOR_IN1, GPIO.OUT)

GPIO.setup(MOTOR_IN2, GPIO.OUT)

GPIO.setup(MOTOR_ENA, GPIO.OUT)

motor_pwm = GPIO.PWM(MOTOR_ENA, 1000)

motor_pwm.start(0)


def start_conveyor(speed=100):

GPIO.output(MOTOR_IN1, GPIO.HIGH)

GPIO.output(MOTOR_IN2, GPIO.HIGH)

motor_pwm.ChangeDutyCycle(speed)

print(f"Conveyor running at {speed}% speed")


def stop_conveyor():

GPIO.output(MOTOR_IN1, GPIO.LOW)

GPIO.output(MOTOR_IN2, GPIO.LOW)

print("Conveyor stopped")


# Servo Motor Setup (GPIO 12 for example)
```

```
SERVO_PIN = 12
GPIO.setup(SERVO_PIN, GPIO.OUT)
servo_pwm = GPIO.PWM(SERVO_PIN, 50)  # 50Hz
servo_pwm.start(0)

class RealServo:
    def __init__(self, pwm):
self.pwm = pwm

    def move_to_angle(self, angle):
        duty = 2 + (angle / 18)
self.pwm.ChangeDutyCycle(duty)
print(f"Servo → {angle}°")
time.sleep(0.5)
self.pwm.ChangeDutyCycle(0)

    def mid(self):
self.move_to_angle(180)

    def min(self):
```

```
self.move_to_angle(0)


servo = RealServo(servo_pwm)


# Load YOLOv8 model
model = YOLO("best.pt")


# Traceability DB
conn = sqlite3.connect('pcb_traceability.db')
cursor = conn.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS pcbs (
device_id TEXT,
batch_id TEXT,
    timestamp TEXT,
    defect TEXT,
rohs_status TEXT,
    status TEXT
)''')


# RoHS DB
```

```
rohs_conn = sqlite3.connect('rohs_data.db')
rohs_cursor = rohs_conn.cursor()

def wait_for_pcb():
print("Waiting for PCB...")
    while GPIO.input(IR_PIN) == 1:
time.sleep(1.5)
print("📦 PCB Detected!")

    # ☑ Light up PASS LED briefly when PCB is detected
GPIO.output(LED_PASS, GPIO.HIGH)
time.sleep(0.5)
GPIO.output(LED_PASS, GPIO.LOW)

def capture_image(filename="pcb.jpg"):
    cap = cv2.VideoCapture(0)
time.sleep(2)
    for _ in range(10):
cap.read()
```

```
time.sleep(0.1)
    ret, frame = cap.read()
    if ret:
        cv2.imwrite(filename, frame)
cap.release()
    return ret


def decode_qr(image_path="pcb.jpg"):
    image = cv2.imread(image_path)
decoded_objs = decode(image)
    for obj in decoded_objs:
        data = obj.data.decode('utf-8')
print("QR Code Data:", data)
        return data
    return ""


def parse_qr_data(data):
device_id, batch_id = "UNKNOWN", "UNKNOWN"
    parts = data.split(';')
    for part in parts:
```

```
        if 'DeviceID' in part:
device_id = part.split(':')[1].strip()
        if 'BatchID' in part:
batch_id = part.split(':')[1].strip()
    return device_id, batch_id


def check_rohs(device_id, batch_id):
rohs_cursor.execute("SELECT rohs_status FROM rohs_compliance WHERE device_id= ? AND batch_id= ?", (device_id, batch_id))
    result = rohs_cursor.fetchone()
    return result[0] if result else "Unknown"


def detect_defect(image_path="pcb.jpg"):
    try:
        results = model.predict(image_path, conf=0.25, imgsz=640)
        labels = [model.model.names[int(cls)] for cls in results[0].boxes.cls]
print("Detected labels:", labels)
reject_labels = {'mouse_bite', 'spur', 'missing_hole', 'short', 'open_circuit', 'spurious_copper'}
```

```
        return any(label.lower() in reject_labels for label in labels)
    except Exception as e:
print("Defect detection error:", e)
        return False


def generate_label(device_id, batch_id, rohs_status, status):
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
label_text = f"Device ID: {device_id}\nBatch ID: {batch_id}\nTimestamp: {timestamp}\nRoHS: {rohs_status}\nStatus: {status}"
label_img = Image.new('RGB', (300, 150), color='white')
    draw = ImageDraw.Draw(label_img)
    font = ImageFont.load_default()
draw.text((10, 10), label_text, fill='black', font=font)
label_img_cv = cv2.cvtColor(np.array(label_img), cv2.COLOR_RGB2BGR)
    filename = f"label_{device_id}_{timestamp.replace(':', '-')}.jpg"
    cv2.imwrite(filename, label_img_cv)
print(f"Label saved: {filename}")
    cv2.imshow("Label", label_img_cv)
```

```
    cv2.waitKey(3000)
    cv2.destroyAllWindows()


def log_data(device_id, batch_id, rohs_status, defect, status):
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
cursor.execute("INSERT INTO pcbs VALUES (?, ?, ?, ?, ?, ?)",
                   (device_id, batch_id, timestamp, "Yes" if defect else "No", rohs_status, status))
conn.commit()


def reject_pcb():
    # ☑ Light up REJECT LED when rejecting
GPIO.output(LED_REJECT, GPIO.HIGH)


servo.mid()
time.sleep(0.5)
servo.min()
time.sleep(0.5)
servo.mid()
time.sleep(0.5)
```

```
GPIO.output(LED_REJECT, GPIO.LOW)


def main():
start_conveyor()
    while True:
wait_for_pcb()
time.sleep(1)
stop_conveyor()


        if not capture_image():
time.sleep(2)
start_conveyor()
            continue


qr_data = decode_qr()


        if not qr_data:
print("No QR code found. Skipping...")
start_conveyor()
```

```
            continue

device_id, batch_id = parse_qr_data(qr_data)
rohs_status = check_rohs(device_id, batch_id)

        if rohs_status != "Compliant":
            defect = False
            status = "Rejected: RoHS"
reject_pcb()
        else:
            defect = detect_defect()
            if defect:
                status = "Rejected: Defect"
reject_pcb()
            else:
                status = "Passed"

log_data(device_id, batch_id, rohs_status, defect, status)
generate_label(device_id, batch_id, rohs_status, status)
start_conveyor()
```

```
if __name__ == "__main__":
    try:
main()
    finally:
print("Cleaning up...")
motor_pwm.stop()
servo_pwm.stop()
conn.close()
rohs_conn.close()
GPIO.cleanup()
```

CHALLENGES FACED

1. Hardware Challenges

Sensor Sensitivity and Detection Reliability

- The IR sensor occasionally fails to detect smaller PCBs, especially those with reflective surfaces, leading to inconsistent triggering.
- External light sources or environmental noise can interfere with the sensor's accuracy, resulting in false detections.

Motor and Servo Control

- Servo motors may not consistently reach the intended angle due to minor voltage drops or timing mismatches on GPIO pins.
- Variations in conveyor motor speed can cause the PCB to move during image capture, reducing detection accuracy.

Camera Quality and Imaging

- Budget-friendly cameras may suffer from poor resolution or motion blur, which directly impacts the defect detection process.
- Improper focus and inconsistent lighting conditions degrade the performance of the YOLOv8 model during inspection.

---

2. Software Challenges

Model Training and Inference (YOLOv8)

- Creating a robust defect detection model requires a large, well-labeled, and diverse training dataset.
- Running real-time inference on Raspberry Pi is computationally limited due to the absence of dedicated GPU support.
- The system occasionally misclassifies PCBs due to false positives or negatives, affecting rejection accuracy.

QR Code Recognition

- Decoding errors can occur if the image is blurred or the QR code is poorly positioned.
- Some QR formats or damaged codes are difficult to interpret, which hinders accurate device identification.

System Stability

- Prolonged execution may lead to memory usage issues or unhandled exceptions, causing unexpected system crashes.
- If hardware functions (such as servo actuation or image capture) are delayed, it may block other threads or processes.

---

3. Integration & Timing Issues

Component Coordination

- The system demands precise synchronization between detection, camera activation, motor movement, and sorting actions.
- Even slight delays in one component can lead to incorrect classification or missed inspections.

Database Access and Logging

- Simultaneous access to the RoHS database for compliance checks and logging inspection data can cause locking or corruption issues in SQLite.
- Lack of transaction handling may risk data inconsistency during high-throughput operations.

Label Generation and Display

- Generating output labels dynamically must be optimized; otherwise, it slows down the conveyor process.
- Improper font handling or errors in image rendering libraries (e.g., PIL) can cause crashes or display glitches.

4. Accuracy and System Reliability

Detection Accuracy

- The model may confuse PCB design elements (e.g., traces vs. unwanted copper) without fine-tuned thresholds.
- Subtle defects may be missed entirely if the model isn't trained with enough examples.

System Reliability

- Hardware failures such as camera disconnection, GPIO misconfiguration, or database corruption can halt operations.
- Without a fallback or recovery mechanism, the system is vulnerable to downtime during real-time production.

---

5. Scalability and Maintenance

High-Speed Production Compatibility

- Scaling the system for faster conveyor speeds requires strict timing control and faster image processing.
- The Raspberry Pi may become a bottleneck without performance optimization.

Model Update and Expansion

- Updating the AI model involves time-consuming retraining and validation.
- Accommodating new defect types or different PCB formats may require significant manual adjustments and data collection.

FEATURES OFFERED

1. Real-Time PCB Detection

2. Automated Image Capture

3. AI-Powered Defect Detection (YOLOv8)

4. QR Code-Based Product Identification

5. RoHS Compliance Verification

6. Label Generation and Display

7. Intelligent Rejection Mechanism

8. Visual Indicators

9. Traceability Logging System

10. Conveyor Belt Control

11. Fully Automated Workflow

12. Scalable and Modular

PROCESS FLOW

PCB Detection:

The IR sensor detects the arrival of a PCB on the conveyor belt.

Conveyor Stops & Image Captured:

The conveyor pauses automatically, and a camera captures an image of the PCB.

QR Code Decoding:

The system reads the QR code on the PCB to extract the Device ID and Batch ID.

RoHS Compliance Check:

Using the extracted IDs, the system checks the RoHS compliance status from a local database.

Defect Detection (YOLOv8):

If the PCB is compliant, the captured image is analyzed using a YOLOv8 model to detect physical manufacturing defects.

Decision Making:

Based on the RoHS and defect status:

If non-compliant or defective, the PCB is rejected using a servo actuator.

If compliant and defect-free, the PCB is passed for labeling.

Label Generation:

A label is generated showing all inspection details (IDs, RoHS status, result, timestamp) and optionally displayed or printed.

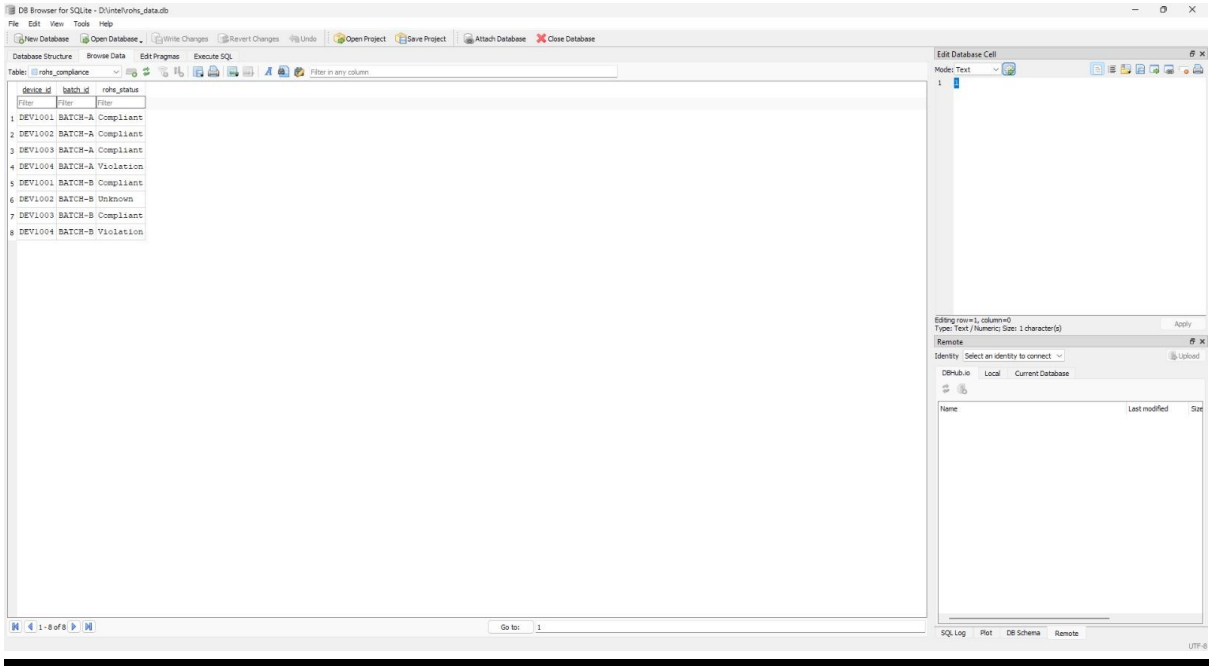
Logging:

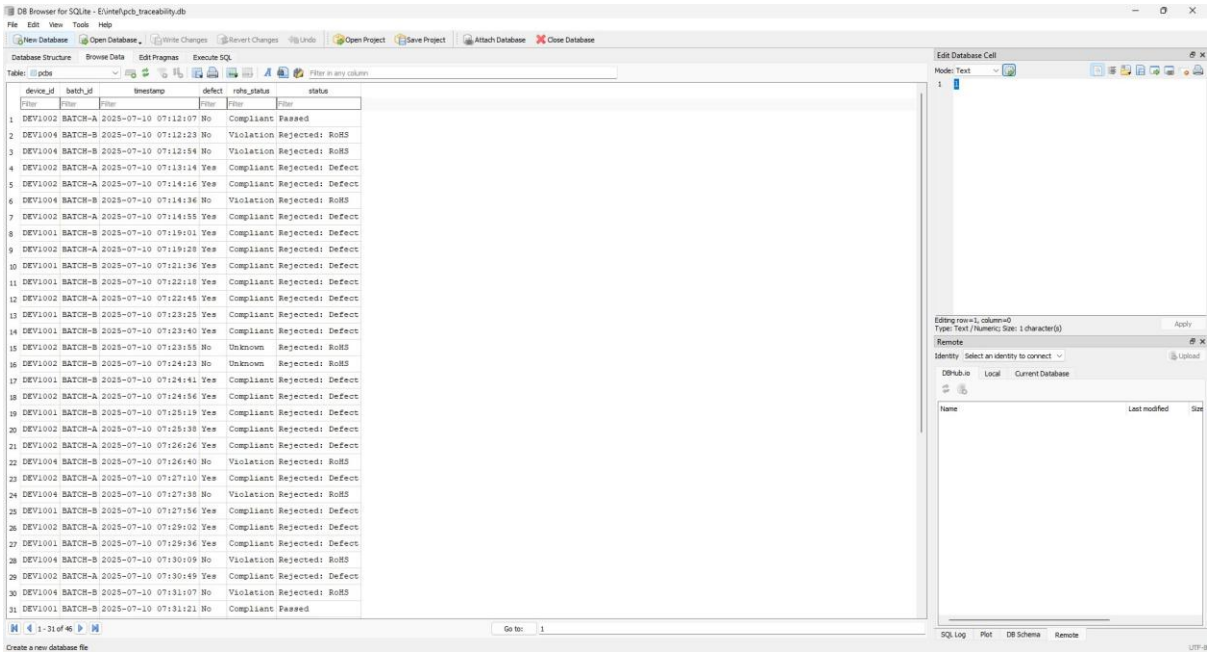
All results are stored in a traceability database for future audits and performance tracking.

Conveyor Resumes:

The system restarts the conveyor and waits for the next PCB.

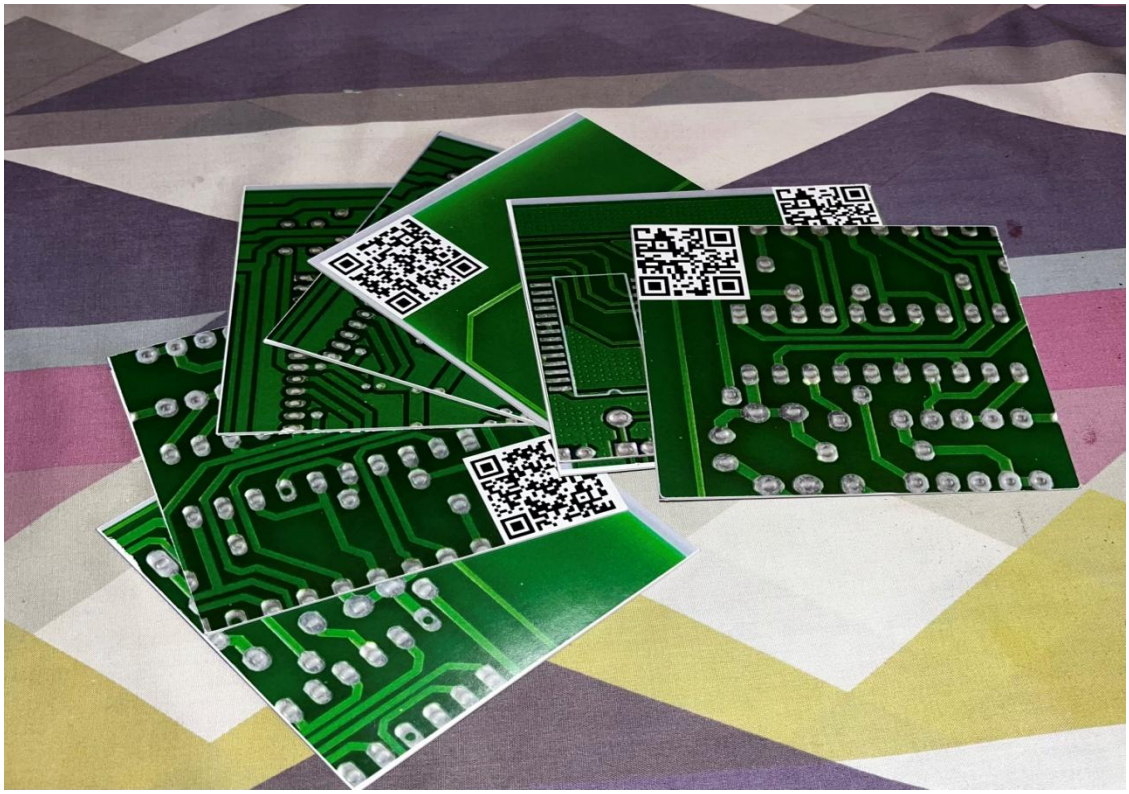
DATABASES


DB Browser for SQLite - D:\intel\rohs_data.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: rohs_compliance
Filter in any column
device_id batch_id rohs_status
1 DEV1001 BATCH-A Compliant
2 DEV1002 BATCH-A Compliant
3 DEV1003 BATCH-A Compliant
4 DEV1004 BATCH-A Violation
5 DEV1001 BATCH-B Compliant
6 DEV1002 BATCH-B Unknown
7 DEV1003 BATCH-B Compliant
8 DEV1004 BATCH-B Violation
1 - 8 of 8
Go to: 1
Edit Database Cell
Mode: Text
Editing row=1, column=0
Type: Text / Numeric; Size: 1 character(s)
Apply
Remote
Identity Select an identity to connect
Upload
DBHub.io Local Current Database
Name Last modified Size
SQL Log Plot DB Schema Remote
UTF-8


DB Browser for SQLite - E:\intel\pcb_traceability.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: pcbs
Filter in any column
device_id batch_id timestamp defect rohs_status status
1 DEV1002 BATCH-A 2025-07-10 07:12:07 No Compliant Passed
2 DEV1004 BATCH-B 2025-07-10 07:12:23 No Violation Rejected: RoHS
3 DEV1004 BATCH-A 2025-07-10 07:12:54 No Violation Rejected: RoHS
4 DEV1002 BATCH-A 2025-07-10 07:13:14 Yes Compliant Rejected: Defect
5 DEV1002 BATCH-A 2025-07-10 07:14:16 Yes Compliant Rejected: Defect
6 DEV1004 BATCH-B 2025-07-10 07:14:36 No Violation Rejected: RoHS
7 DEV1002 BATCH-A 2025-07-10 07:14:55 Yes Compliant Rejected: Defect
8 DEV1001 BATCH-B 2025-07-10 07:19:01 Yes Compliant Rejected: Defect
9 DEV1002 BATCH-A 2025-07-10 07:19:28 Yes Compliant Rejected: Defect
10 DEV1001 BATCH-B 2025-07-10 07:21:36 Yes Compliant Rejected: Defect
11 DEV1001 BATCH-B 2025-07-10 07:22:18 Yes Compliant Rejected: Defect
12 DEV1002 BATCH-A 2025-07-10 07:22:45 Yes Compliant Rejected: Defect
13 DEV1001 BATCH-B 2025-07-10 07:23:25 Yes Compliant Rejected: Defect
14 DEV1001 BATCH-B 2025-07-10 07:23:40 Yes Compliant Rejected: Defect
15 DEV1002 BATCH-B 2025-07-10 07:23:55 No Unknown Rejected: RoHS
16 DEV1002 BATCH-B 2025-07-10 07:24:23 No Unknown Rejected: RoHS
17 DEV1001 BATCH-B 2025-07-10 07:24:41 Yes Compliant Rejected: Defect
18 DEV1002 BATCH-A 2025-07-10 07:24:56 Yes Compliant Rejected: Defect
19 DEV1001 BATCH-B 2025-07-10 07:25:19 Yes Compliant Rejected: Defect
20 DEV1002 BATCH-A 2025-07-10 07:25:38 Yes Compliant Rejected: Defect
21 DEV1002 BATCH-A 2025-07-10 07:26:26 Yes Compliant Rejected: Defect
22 DEV1004 BATCH-B 2025-07-10 07:26:40 No Violation Rejected: RoHS
23 DEV1002 BATCH-A 2025-07-10 07:27:10 Yes Compliant Rejected: Defect
24 DEV1004 BATCH-B 2025-07-10 07:27:38 No Violation Rejected: RoHS
25 DEV1001 BATCH-B 2025-07-10 07:27:56 Yes Compliant Rejected: Defect
26 DEV1002 BATCH-A 2025-07-10 07:29:02 Yes Compliant Rejected: Defect
27 DEV1001 BATCH-B 2025-07-10 07:29:36 Yes Compliant Rejected: Defect
28 DEV1004 BATCH-B 2025-07-10 07:30:09 No Violation Rejected: RoHS
29 DEV1002 BATCH-A 2025-07-10 07:30:49 Yes Compliant Rejected: Defect
30 DEV1004 BATCH-B 2025-07-10 07:31:07 No Violation Rejected: RoHS
31 DEV1001 BATCH-B 2025-07-10 07:31:21 No Compliant Passed
1 - 31 of 46
Go to: 1
Edit Database Cell
Mode: Text
Editing row=1, column=0
Type: Text / Numeric; Size: 1 character(s)
Apply
Remote
Identity Select an identity to connect
Upload
DBHub.io Local Current Database
Name Last modified Size
SQL Log Plot DB Schema Remote
Create a new database file
UTF-8

PROTOTYPE

PCB



LABELS GENERATED

Device ID: DEV1002
Batch ID: BATCH-A
Timestamp: 2025-07-10 07:24:56
RoHS: Compliant
Status: Rejected: Defect

Device ID: DEV1004
Batch ID: BATCH-B
Timestamp: 2025-07-10 07:39:56
RoHS: Violation
Status: Rejected: RoHS

TECHNOLOGIES USED

YOLOv8: An advanced deep learning model for real-time object detection which enables identification of objects with high accuracy

OpenCV: A computer vision library used for video capture

DC Motor:To rotate the conveyor

Motor Driver: To control the DC motor

IR sensor: to detect PCB

Servo Motor: To remove unwanted PCBs

Sqlite3: To store the data

Team members&contribution

NAME	CONTRIBUTION
Abijith SS	Hardware
George K John	Hardware
Jeffin I Patrick	Software

CONCLUSION

The Smart PCB Traceability and Compliance Verification System represents a major step forward in enhancing automation within PCB manufacturing. By integrating computer vision, machine learning–based defect detection, and automated labeling with real-time database operations, the system enables accurate product tracking, efficient compliance checks, and reduced reliance on manual intervention.

This intelligent solution enhances both product quality and manufacturing efficiency, while ensuring readiness for regulatory audits. Leveraging technologies like OCR, AI, and automated sorting, the system successfully overcomes the common shortcomings of traditional inspection methods and aligns well with the goals of Industry 4.0.

Overall, this project contributes to building a smarter, more reliable, and fully traceable production environment—paving the way for future-ready, scalable electronics manufacturing systems.