🚘 Smart License Plate Detection and Logging System

A Full-Stack IoT System with OCR, OpenCV, Flask, Tkinter, MQTT, SQLite, Logging, and ThingsBoard Integration

---

🎯 Objective

Design and develop a full-stack IoT system using Raspberry Pi that:

- Detects and reads vehicle license plates using a webcam and OCR.
- Logs detections locally using SQLite and structured logging.
- Provides:
  - A **Tkinter-based local dashboard** on the Laptop.
  - A **Flask-based remote admin panel** for system control and log access.
  - Real-time **MQTT streaming** of detection events.
  - A **centralized ThingsBoard cloud dashboard** to monitor and visualize plate logs.
- Supports **CI/CD** workflows via GitHub Actions.

Students will integrate image processing, IoT communication, database handling, UI/UX, cloud integration, and DevOps practices in a single deployable project.

🧰 Hardware Requirements

Component	Description
Raspberry Pi 4/5	Core processing unit
Pioneer600 HAT	Optional GPIO interface for alarms/LEDs
USB Webcam / Pi Camera	Video capture
Laptop	For admin access and testing
Internet Access	Required for MQTT and ThingsBoard

⚙️ Software Stack

🐍 Python Libraries

- `opencv-python` – Image capture and processing
- `easyocr` or `pytesseract` – OCR for license plates
- `tkinter` – Real-time local GUI
- `flask` – Admin web app
- `sqlite3` – Logging database
- `paho-mqtt` – MQTT client
- `logging` – Runtime logs
- `threading`, `datetime`, `os`, `re`, `json` – Utility support

🌐 Cloud Dashboard

- **ThingsBoard Cloud** (https://thingsboard.io/)
  - Device telemetry dashboard
  - Table, map, and graph widgets
  - Public or user-authenticated access

🛠️ DevOps & CI/CD

- GitHub + GitHub Actions (for testing and deployment)
- Optionally: Docker for Flask admin if needed

🧠 Functional Components

1. 🎥 Real-Time Detection (OpenCV + OCR)

- Detect license plates from webcam frames.
- Use OCR to extract alphanumeric characters.
- Pre-process images for accuracy (grayscale, contours, ROI).

2. 📝 SQLite + Structured Logging

- Log entries with:
  - Plate ID
  - Timestamp
  - OCR confidence
  - Snapshot path
- SQLite table: `plate_logs(id INTEGER, plate TEXT, time TEXT, image TEXT)`
- Log file: `detection.log` with rotating logging handler

3. 🖥️ Tkinter Dashboard (Local GUI)

- Show:
  - Live camera feed
  - Last 10 detections
  - Detection success/fail status
- Useful for monitoring on-the-spot without browser

4. 🌐 Flask Admin Panel (Remote Access)

- Routes:
  - `/logs`: View and filter logs
  - `/settings`: Configure thresholds, detection mode, MQTT
  - `/export`: Download `.csv` of logs
  - `/start`, `/stop`: Control detection engine
- Optional password login

5. 📡 MQTT Integration

- Publish messages on topic like `pi/license_plate`:

```
{
  "plate": "KA03MM1234",
  "timestamp": "2025-06-13T12:34:56",
  "confidence": 0.92
}
```

- Supports local Mosquitto broker or remote broker (HiveMQ, etc.)

6. 📊 ThingsBoard Dashboard

- Register Pi as a *device* on ThingsBoard
- Create and upload **custom telemetry**:
  - Plate ID
  - Timestamp
  - Count metrics (plates per hour/day)
- Build dashboard:
  - Plate table
  - Live counter
  - Chart of detections over time
  - Optional GPS map (with mock or static data)

7. 🔁 CI/CD via GitHub

- Push code to GitHub repo
- GitHub Actions:
  - Install dependencies
  - Run tests or lint
  - SSH deploy to Raspberry Pi
  - Restart services (systemctl, supervisor, or script)

📁 Suggested Project Structure

```
smart-license-plate/
│
├── main.py                  # Core detection + logging
├── ocr_utils.py             # OCR and cleaning
├── db_utils.py              # SQLite operations
├── log_utils.py             # Logging config
├── mqtt_client.py           # MQTT logic
├── camera_utils.py          # Frame capture
│
├── tkinter_dashboard.py     # Real-time local GUI
│
├── flask_admin/
│   ├── app.py               # Flask admin server
│   └── templates/
│       └── index.html
│
├── logs/
│   ├── detection.log
│   └── plates.db
│
├── thingsboard/
│   └── tb_client.py         # MQTT wrapper for ThingsBoard
│   └── device-config.json   # Optional template
│
├── .github/
│   └── workflows/
│       └── deploy.yml
│
├── requirements.txt
└── README.md
```

⏱️ Suggested Time Breakdown (16 hrs)

Time	Task
1 hr	Raspberry Pi + webcam setup
2 hrs	OpenCV-based plate detection
2 hrs	OCR integration and result parsing
1 hr	SQLite DB logging + structured logs
2 hrs	Tkinter dashboard development
2 hrs	Flask admin with settings and DB viewer
1 hr	MQTT setup and testing
2 hrs	ThingsBoard dashboard + data publishing
2 hrs	CI/CD with GitHub Actions
1 hr	Testing, polish, and documentation

📊 Evaluation Rubric (100 points)

Criteria	Points
Plate detection + OCR accuracy	15
SQLite logging + error logs	10
Local Tkinter dashboard	10
Flask admin panel (settings/logs)	15
MQTT communication	10
ThingsBoard cloud dashboard	15
CI/CD pipeline via GitHub Actions	10
Code quality, modularity, comments	5
Final demo and presentation	10

🌟 Optional Extensions

- GPS integration for map visualization in ThingsBoard
- Use Pioneer600 HAT for gate control (simulate barrier for known plates)
- Face detection of driver
- Email alert for blacklisted plates
- Use Google Sheets or Firebase as secondary data store

✅ Final Deliverables

1. Fully working project on Raspberry Pi
2. Flask admin accessible over LAN
3. Tkinter UI on Raspberry Pi display
4. Live MQTT feed and connected ThingsBoard dashboard
5. GitHub repo with:
    - Source code
    - Working CI/CD workflow
    - README with setup instructions