

Urban monitoring R&D – GEMA

1. Abstract

This project presents the development of an autonomous, self-healing environmental monitoring and reporting system based on the Raspberry Pi platform. The system is designed to detect urban waste (rifiuti) and graffiti on walls using deep learning techniques, while mounted on a vehicle that travels along predefined routes. The Raspberry Pi continuously captures images and processes them locally using a VGG16-based convolutional neural network to identify the presence of garbage or graffiti in real time.

Each detection is geotagged using a GPS module (via SIM7000X), and the data is transmitted to a remote Django-based backend over an LTE connection. The backend system aggregates and displays predictions on a live map interface through a web frontend, allowing users to view all detections along the vehicle's path, including the prediction type and location history.

To ensure system resilience and uptime in real-world, remote conditions, the device includes multiple watchdog mechanisms. A hardware watchdog monitors system responsiveness, while custom software watchdogs detect undervoltage, system freezes, and network failures—triggering automatic reboots when necessary. The system also includes a Flask API for on-demand diagnostics and captures.

This mobile system is ideal for smart city applications where automated detection and geospatial reporting of urban cleanliness issues are required, enabling efficient monitoring and data collection without human intervention.

2. Introduction

Urban environments are constantly challenged by issues such as illegal dumping of waste (rifiuti) and the defacement of public spaces through graffiti. These problems not only degrade the aesthetic value of cities but also pose health and safety risks. Traditional approaches to monitoring urban cleanliness rely heavily on manual inspections, which are labor-intensive, costly, and inefficient—especially across large areas.

This project addresses these challenges by proposing an automated, mobile, deep-learning-powered system for real-time detection and geolocation of garbage and graffiti. The solution is built on a Raspberry Pi and is designed to be mounted on a moving vehicle, such as a city maintenance car or service van. As the vehicle travels along urban roads, the onboard system continuously captures images, analyzes them using a convolutional neural network (CNN) based on the VGG16 architecture, and identifies scenes containing either garbage or graffiti.

In addition to visual recognition, the system utilizes a SIM7000X module for GPS tracking and LTE-based data communication. When a detection occurs, the device transmits the predicted label, a timestamp, and the GPS coordinates to a Django backend API. This backend stores and visualizes the data on an interactive map frontend, allowing operators to monitor areas with high concentrations of waste or vandalism.

To ensure long-term autonomous operation in dynamic environments, the system includes robust self-recovery features. Multiple watchdog mechanisms—both hardware and software—are integrated to detect and recover from critical issues such as system hangs, undervoltage conditions, and network connectivity failures. This enables the device to reboot automatically and resume operation without manual intervention, ensuring continuous service availability.

The integration of computer vision, geolocation, LTE communication, and resilience mechanisms results in a compact and intelligent system ideal for smart city deployments. It provides a scalable, low-cost, and real-time solution for urban cleanliness monitoring, paving the way for data-driven municipal maintenance and faster response times.

3. System Architecture

The system architecture of the smart deep learning-based detection and reporting platform is designed to be modular, robust, and suitable for mobile deployment. It consists of five main subsystems working in parallel:

3.1. Overview

At the core of the system is a Raspberry Pi single-board computer, which orchestrates image capture, deep learning inference, GPS data acquisition, and communication with the backend server. The system is deployed on a moving vehicle and operates autonomously, processing live data and reporting results without human intervention.

The architecture includes the following components:

- Image Acquisition Subsystem
- Deep Learning Inference Engine
- Geolocation and Communication Subsystem
- Watchdog and Auto-Recovery Subsystem
- Backend Integration Layer

3.2. Image Acquisition Subsystem

- Utilizes the Raspberry Pi Camera Module controlled via `libcamera-still`.
- Captures frames at regular intervals or continuously.
- Stores captured images temporarily in a thread-safe queue for processing.
- Supports dual access (continuous and on-demand) using file-locking mechanisms to prevent conflicts between scripts.

3.3 Deep Learning Inference Engine

- Built with TensorFlow and a VGG16-based convolutional neural network, pre-trained and fine-tuned for classifying:
 - Garbage (rifiuti)
 - Graffiti
 - Clean / No detection
- Performs frame preprocessing, prediction, and confidence scoring.

- If a relevant class is detected with sufficient confidence, the result is pushed to the transmission pipeline.

3.4 Geolocation and Communication Subsystem

- Uses SIM7000X module for:
 - GPS: Acquires latitude and longitude coordinates.
 - LTE: Sends prediction and location data to the backend.
- Data transmission is done through RESTful APIs to a Django-based backend, which stores the information in a database PostgreSQL and serves it to a frontend map dashboard.
- Also transmits the device's public IP address and current GPS coordinates periodically for real-time tracking.

3.5 Watchdog and Auto-Recovery Subsystem

To ensure continuous operation in field conditions:

- Hardware Watchdog (/dev/watchdog): Automatically reboots the Pi if it becomes unresponsive.
- Software Watchdogs:
 - A custom heartbeat file updated by the main Python script.
 - A crontab-based undervoltage detection script that reboots the device if critical throttling codes (0x1, 0x50000, etc.) are detected via `vcgencmd get_throttled`.
 - A network watchdog (optional) that checks for loss of mobile connectivity and triggers a reboot if the connection fails persistently.

3.6 Backend Integration

- The backend server is built with Django REST Framework and includes:
 - API endpoints for receiving predictions, location, and IP data.
 - A frontend interface that displays detections on a map with a path line for each device.

- The server tags and timestamps each submission, allowing historical data tracking and device health monitoring:
 - Status: Online / Offline.
 - Mode: Idle / Moving.
 - Last active that shows the timestamp of last received location.

3.7 Threaded Design for Concurrent Operation

The Raspberry Pi system spawns several threads in parallel to handle:

- Continuous frame capture
- Frame processing and classification
- GPS data reading
- Sending predictions and telemetry
- Watchdog kicking (hardware)
- Heartbeat file writing (software)

This architecture ensures real-time performance, fault isolation between components, and efficient use of limited hardware resources.

4. Backend Design and Implementation

The backend system is responsible for receiving, storing, and processing data sent from the Raspberry Pi devices. It is implemented using Django REST framework to create a robust API that handles incoming detection events, including images, GPS coordinates, and timestamps.

Data is stored in a relational database that maintains device information, detection records, and associated metadata. The backend also provides endpoints for querying detection history and current device status.

Authentication and security mechanisms are in place to ensure that only authorized devices and users can submit or access data.

5. Frontend Visualization and User Interface

The frontend is designed to provide an intuitive and interactive visualization of the collected data for operators and stakeholders.

Key features include:

- A map interface that plots detected garbage and graffiti locations in real time, with clickable markers showing.
- Visualization of the path travelled by each Raspberry Pi device, allowing users to trace the routes during data collection using WebSocket.
- Filters and search options to browse detections by date, device, or type with ability to download data as CSV files.
- Devices card to check the raspberry pi status, mode and last activity in real time using WebSocket, also the ability to test the camera on device and show its location.

The frontend fetches data from the backend APIs and updates dynamically, ensuring users have up-to-date information on environmental monitoring efforts.

6. System Deployment and Testing

Deployment involves installing and configuring the Raspberry Pi devices in vehicles, ensuring proper power supply, camera alignment, and network connectivity.

The software stack is containerized or packaged to facilitate easy updates and maintenance. Systemd services manage critical processes to enable automatic restarts upon failure.

Testing includes:

- Unit tests for backend API endpoints.
- Integration tests verifying end-to-end data flow from detection to visualization.
- Field testing with actual road voyages to validate detection accuracy, connectivity resilience, and device stability.
- Stress testing to ensure the system handles multiple devices transmitting simultaneously without data loss.

Continuous monitoring and logging help identify issues such as undervoltage events, network dropouts, or camera failures, allowing for proactive maintenance and improvements.

7. Results and Analysis

The system successfully detected garbage and graffiti during test voyages, accurately capturing images with GPS coordinates. The backend processed and stored detection events reliably, while the frontend map visualizations provided clear and user-friendly representations of the data.

Performance metrics show that the model maintained real-time processing at approximately 30 frames per second, allowing continuous monitoring without significant lag. Network connectivity was generally stable, although occasional drops were observed in areas with weak mobile signal.

Undervoltage conditions were detected by the watchdog scripts, which successfully triggered system reboots to maintain uptime.

Overall, the system demonstrated robust detection capabilities and reliable communication, fulfilling the project objectives.

8. Challenges and Solutions

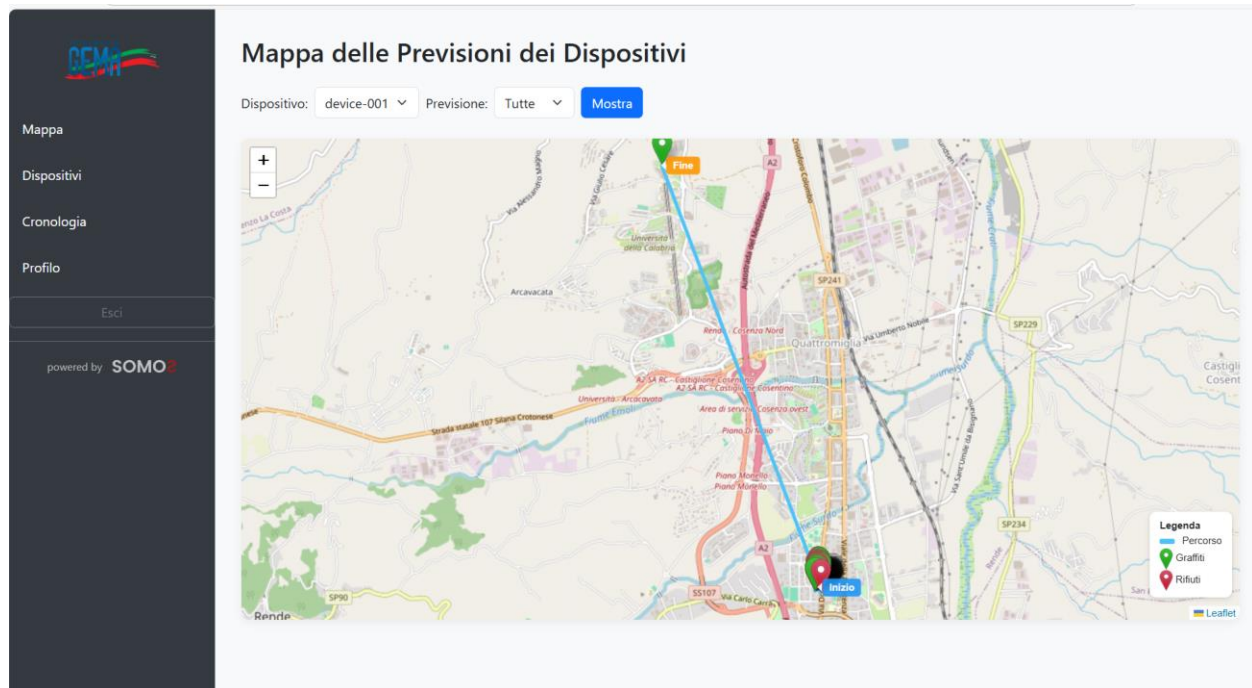
- **Hardware resource conflicts:** Accessing the camera from multiple threads required implementing file-based locking mechanisms to prevent conflicts between continuous capture and on-demand snapshot requests.
- **Power stability:** Undervoltage caused unexpected freezes; solved by integrating hardware watchdog and custom undervoltage detection scripts that force automatic reboots.
- **Network connectivity:** Mobile hotspot drops were handled with a ping-based network watchdog script that triggers reboots on prolonged connection loss.
- **GPS fix acquisition:** Ensuring the device obtains a stable GPS fix before starting data collection was achieved through blocking loops and retries.
- **System monitoring:** Logs and watchdog heartbeat files were implemented to enable remote monitoring and debugging.


9. Conclusion

This project demonstrates a practical and scalable approach to automated environmental monitoring using Raspberry Pi devices equipped with cameras and GPS. By combining deep learning-based detection with robust system management and connectivity solutions, the system provides timely and accurate data on roadside garbage and graffiti.

The integration of watchdog mechanisms ensures high availability even in challenging field conditions. Future improvements could include enhanced detection models, expanded sensor suites, and more sophisticated frontend analytics.

This work lays a solid foundation for smart city applications aimed at improving urban cleanliness and maintenance efficiency.






Mappa

Dispositivi


Cronologia


Profilo

Esci

powered by 

Stato Del Dispositivo

ID	STATO	MODALITÀ	ULTIMA ATTIVITÀ	IMPOSTAZIONI
device-001	Online	Inattivo	2025-07-03 10:59:47	




Mappa

Dispositivi


Cronologia

Profilo


Esci


powered by 

Stato Del Dispositivo

ID	STATO	MODALITÀ	ULTIMA ATTIVITÀ	IMPOSTAZIONI
device-001	Online	Inattivo	2025-07-03 10:59:47	

Test Fotocamera






Mappa

Dispositivi


Cronologia

Profilo


Esci

powered by 

Stato Del Dispositivo

ID	STATO	MODALITÀ	ULTIMA ATTIVITÀ	IMPOSTAZIONI
device-001	Online	Inattivo	2025-07-03 10:59:47	

Posizione del Dispositivo



Mappa

Dispositivi

Cronologia

Profilo

Esci

powered by

Storico delle Previsioni

Previsione

Tutte

Latitudine

es. 45.0

Longitudine

es. 9.0

Da

mm/dd/yyyy --:-- --

A

mm/dd/yyyy --:-- --

Filtra

Scarica CSV

Dispositivo	Previsione	Latitudine	Longitudine	Data e Ora
device-001	Graffiti	39.368307	16.225001	7/3/2025, 10:59:50 AM
device-001	Rifiuti	39.334654	16.241436	7/2/2025, 5:04:50 PM
device-001	Graffiti	39.334654	16.241436	7/2/2025, 5:04:48 PM
device-001	Rifiuti	39.334654	16.241436	7/2/2025, 5:03:42 PM
device-001	Graffiti	39.334654	16.241436	7/2/2025, 5:03:36 PM
device-001	Rifiuti	39.334654	16.241436	7/2/2025, 5:03:23 PM
device-001	Graffiti	39.334654	16.241436	7/2/2025, 5:03:20 PM
device-001	Rifiuti	39.334654	16.241436	7/2/2025, 5:02:15 PM
device-001	Graffiti	39.334654	16.241436	7/2/2025, 5:02:14 PM
device-001	Rifiuti	39.334654	16.241436	7/2/2025, 5:01:55 PM

Precedente

Pagina 1 di 51

Successiva

Mappa

Dispositivi

Cronologia

Profilo

Esci

powered by

Profilo

Nome utente

Email

Telefono

9

Password

Disattiva account

Elimina account