

IoT Smart Parking: Real-Time Vehicle Detection Using Dual-Model Computer Vision

Luke Redmore (2219318) | Supervisor: Simon P Thomas | BSc Software Engineering | University of Wales Trinity Saint David

1

Summary

This project built a smart parking system that uses cameras and sensors to automatically detect when cars enter or leave a multi-storey car park, updating a live dashboard in real time. The system was tested in a real parking environment and correctly identified every vehicle with 93% precision and under one second response time, running entirely on low-cost hardware without any permanent modifications to the building. The system addresses the strain that ever-increasing vehicle numbers in urban areas place on existing parking infrastructure [1].

2

Context

- Multi-storey car parks present a unique challenge: drivers have no visibility of space availability across floors until physically navigating each level [5].
- Existing solutions target single-level environments and assume controlled lighting - conditions that fail in enclosed multi-storey facilities [2][3].

3

Conclusion

Affordable edge hardware combined with dual-model YOLO detection [3] delivers reliable real-time parking monitoring without proprietary infrastructure. The system achieved 100% recall and 93.3% precision during live deployment with sub-second latency and autonomous fault recovery. Future work: physical drone validation for bay-level reconciliation, expanded environmental testing.

4

Aims and Objectives

Aim: Develop an intelligent parking management system providing real-time occupancy tracking through computer vision.

- Critically analyse literature on automated parking, computer vision, and IoT monitoring.
- Design a prototype: event-driven server, dual-model YOLO pipeline, proposed drone reconciliation.
- Develop with real-time SSE streaming, alerting, and drift correction.
- Evaluate using detection confidence, count accuracy, and response latency.

5

System Architecture

- Arduino MKR WiFi 1010 (ultrasonic + servo) → MQTT → Raspberry Pi 5 (broker + camera) → Node.js server (dual YOLO worker) → Dashboard [4]

6

System Deployment Topology

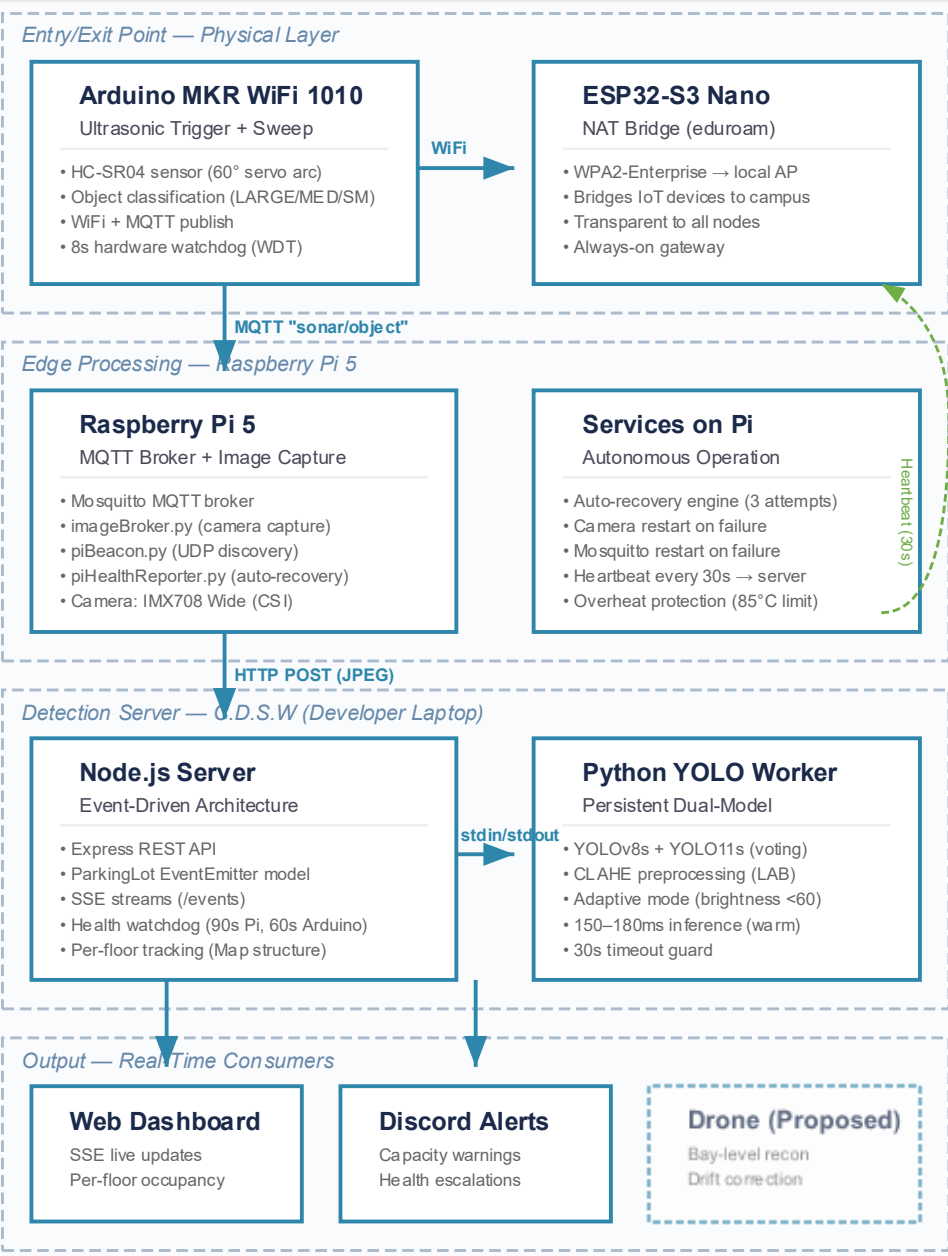


Figure 3 - Diagrammatic overview of the full system deployment outlining each module's area of responsibility and status in the hierarchy

7

Experiment Results

- Detection (controlled): 100%
 - Detection (low light + CLAHE): 100%
 - Live deployment recall: 100%
 - Live deployment precision: 93.3%
 - End-to-end latency: <1 second
 - Inference (warm): 150–180ms
 - False positives: 2 (van → car)
 - Autonomous recovery: Validated
- 32 inferences processed during live deployment

8

Topology – IoT Pipeline

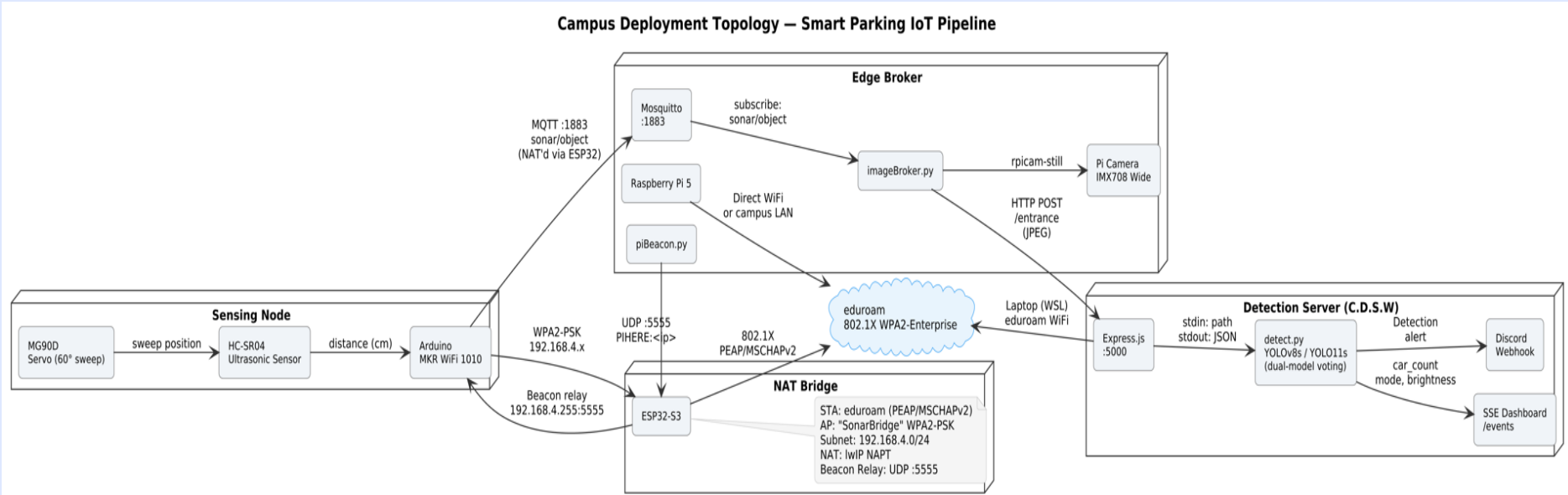


Figure 1 - Logical topology of the full system as it was deployed for the live experiment, demonstrating how the physical layer interconnects and hosts software that enables effective interoperability

9

Detection

- Ultrasonic sensor detects vehicle → triggers camera
- Raspberry Pi captures image → transmits to server
- Dual-model YOLO voting (YOLOv8s + YOLO11s) [3]
- CLAHE preprocessing adapts to low light
- Results streamed to dashboard + Discord alerts



Figure 2 - Full stack deployed. The red box outlines the MKR 1010 board with HC-SR04 sensor; the blue box is the eduroam NAT gateway running on an ESP32 Nano S3; and the yellow/orange box shows the developer laptop running the C.D.S.W detection suite

10

Discord Alerts & Heartbeat

- Each node monitors the one below it and heals itself before asking for help.
- The Arduino reboots itself if it hangs.
- The Pi restarts crashed services automatically.
- The server watches everything and only alerts the operator when self-repair has failed meaning the system runs unattended without human intervention.

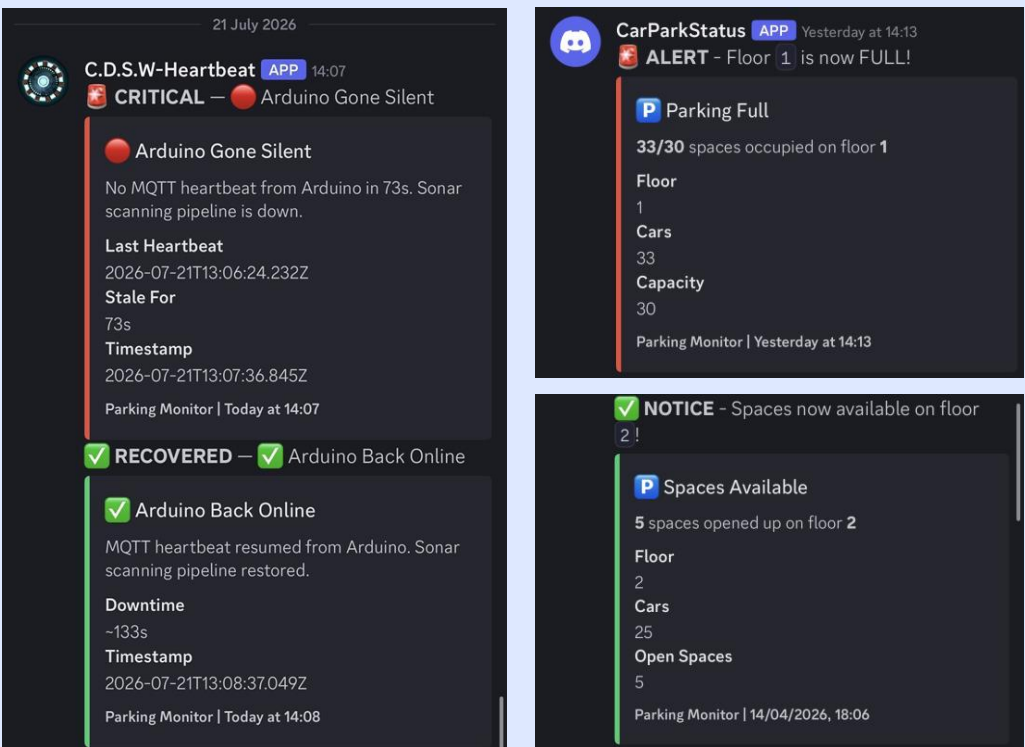


Figure 4 – Screen captures of the C.D.S.W heartbeat (system updates) in action alerting to a connection issue during the experiment and also alerting the user to issue being resolved (self-recovery). To the right is a C.D.S.W status update to alert users when all available spaces are full and when five or more are free