SmartPark

IoT based Parking Solution!!

PROBLEM STATEMENT

Parking has always been an issue in the metropolitan cities.

- It wastes time, fuel, and causes stress.
- Circling for spots increases traffic congestion, fuel wastage, and pollution.

To overcome this, we are designing:

- A low-cost system that will use Raspberry Pi and ESP32 CAM Module for detecting and updating the user real-time parking spots through a mobile app or webpage.
- This system is compact, cost-effective and can be installed at any parking location.

SYSTEM COMPONENTS

1. Raspberry Pi

- Acts as a decentralized server and handles image processing, computation and network management.

2. ESP32 CAM

- For taking images and sending it to Raspberry Pi.

3. 5V Power Supply

INTERACTION

- ESP 32 Camera will capture the image and send it to Raspberry Pi via wireless connection or cable.
- The Raspberry Pi will process the images using the ML Modeland determine parking slot availability.
- This data will be published on a webpage using AWS server and the user will know if parking slot is available or not.

TRADEOFFS

Local vs. Cloud Processing: We process data directly on the device for low-latency responses, although this limits us compared to the more powerful cloud processing.

Database System Decision: We use a straightforward database to save on costs and setup time, accepting that we might need to upgrade later as our data grows.

Computational Constraints: Our Raspberry Pi has limited CPU and RAM, so we use less complex algorithms to ensure the system runs smoothly.

TECH STACK

Hardware:

Raspberry Pi

ESP32 CAM Module

Software:

TensorFlow Lite/PyTorch: For on-device machine learning and image processing.

HTML/CSS/JavaScript: Builds and styles the web application’s user interface.

WebSocket/MQTT: Ensures real-time updates between server and client.

Arduino IDE: Programs Arduino-based hardware. GitHub: Manages version control and collaborative development.

Cloud Services & Databases:

AWS (Amazon Web Services): Hosts, scales, and manages the application and database.

CHALLENGES

Accurate Vehicle Detection:

- Striving for high accuracy with simpler algorithms on the Raspberry Pi.

Cloud Integration:

- Maintaining a stable, real-time connection with the cloud, balancing data processing demands and system stability.

Lighting and Environment:

- Adapting image processing to diverse lighting and weather conditions to preserve detection accuracy.

Connectivity and Latency:

- Ensuring consistent data flow between the ESP32-CAM, Raspberry Pi, and cloud, minimizing delays and data loss.

TIMELINE

Week 1 – Hardware Setup

Week 2 – Object Detection

Week 3 – Integration and Cloud setup

Week 4 – Testing, Refinement and Documentation

QUESTIONS AND SUGGESTIONS?