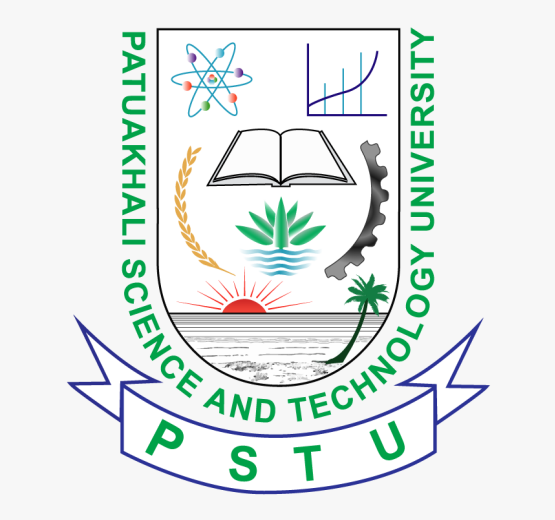
AI-Powered Smart Parking System Using ESP32-CAM for Vehicle and Token Detection

---



PROJECT REPORT

Submitted by

Name:	Zubayer Isfar, Sumon Das
ID:	2202012, 2202014
Reg. no:	11203, 11205
Course code:	EEE-212
Course title:	Electrical Technology Sessional
Level:	02
Semester:	01

Submission Date: 22-07-2025

Introduction

Parking management in busy areas often suffers from unauthorized access and inefficient space utilization. The **AI-Powered Smart Parking System** described in this project addresses these issues by using an ESP32CAM module with onboard edge AI to control entry and exit gates in a parking lot. The system uses **computer vision** (trained via Edge Impulse) to detect vehicles and a required payment token (coin) at the entry gate. Entry is granted only when both a car and the authorized token are simultaneously recognized. The solution integrates IoT hardware (camera, sensors, servos) with machine learning to enable automated, touchless access control and real-time monitoring of parking occupancy. This document presents a detailed overview of the system's design, components, operation logic, and performance.

Contents

System Overview

System Architecture: The parking system consists of an **entry gate** and an **exit gate**, each controlled by a servo motor, and monitored by an ESP32-CAM module with an attached camera. The ESP32-CAM serves as the brain of the system – running a custom-trained object detection model to identify cars and tokens, managing gate actuations, and providing a live video stream over WiFi. An **IR sensor** is placed at the exit gate to detect departing vehicles, and a small **OLED display** is used to show system status and available parking slots in real time. Figure 1 illustrates the overall system architecture, including the interaction between sensors, actuators, and the microcontroller.

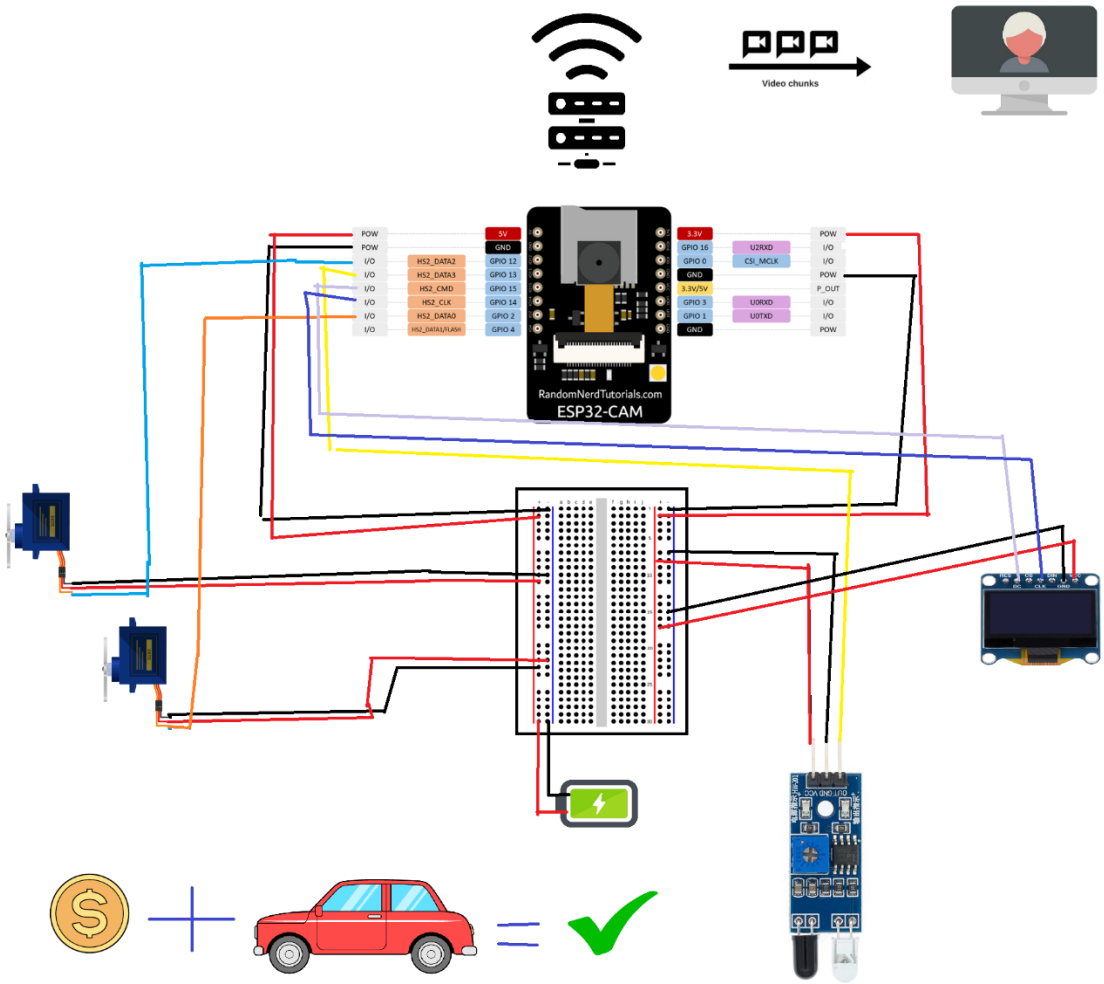


Figure 1: *System architecture diagram*

Key features of the system include:

- **AI-Based Object Detection:** A machine learning model runs on the ESP32-CAM to detect two classes of objects – *car* and *coin* (token) – in the camera's field of view. The model was developed using Edge Impulse and deployed to the device for on-device inference.
- **Dual Gate Control:** Independent servo motors manage the entry and exit barriers. The entry gate servo responds to AI detections, while the exit gate servo is triggered by an IR sensor. This ensures one-way flow of vehicles entering and leaving the parking area.
- **Token-Based Access:** The system implements a **smart payment verification** – a vehicle is only allowed entry if a valid token (custom coin) is presented alongside the car. Showing only the token or only the car will **not** open the gate, enhancing security (the token acts as proof of payment or authorization).
- **Parking Slot Management:** The controller keeps track of available parking slots (up to a configured maximum, e.g. 4 slots). It decrements the count on each entry and increments on each exit, preventing overfilling of the lot. The OLED displays the current number of available slots, and the system will block entry when full.
- **WiFi Streaming and Interface:** The ESP32-CAM hosts a web server that provides a live video stream of the entrance. Users can connect via a URL (displayed on the OLED) to monitor the camera feed in real time. The stream auto-refreshes (~1 frame per second) and can also capture still images on demand.
- **Real-Time Display:** A 0.96-inch OLED screen gives immediate feedback, such as "**CAR FOUND**" or "**COIN FOUND**," "**ENTRY GATE: Opened/Closed**," number of slots remaining, and status messages like "**PARKING FULL**". This helps users and any attendant understand the system's decisions.

Hardware Components and Setup

The project utilizes affordable and widely available hardware components. The main components and their roles are:

- **ESP32-CAM (AI-Thinker module):** Microcontroller with built-in camera used for image capture, AI processing, and network streaming. This board, equipped with 8MB PSRAM, runs the Edge Impulse inference and controls all peripherals.
- **SG90 Servo Motors (2x):** Small 5V servo motors act as gate actuators. One servo controls the entry barrier and the other controls the exit barrier. They are positioned to lift or turn the gate mechanisms open and closed as commanded by the ESP32.
- **IR Sensor Module:** An infrared reflective or beam-break sensor is placed at the exit lane to detect the presence of a vehicle attempting to leave. When a car passes the exit sensor, it triggers the exit gate to open (if conditions are met, see logic below).

- **0.96" OLED Display (128×64 I2C):** A small monochrome display is used to show system status, such as WiFi connection progress, stream URL, detection status, gate state, and available parking slots. This provides real-time feedback to users on-site.
- **Custom Token (Coin):** A physical token (in this case, a blue RFID-style key fob used as a coin) is used as the required object for payment verification. Only this specific object, when detected by the camera, is recognized as a valid "coin" by the AI model, simulating a paid ticket.
- **Power Supply:** A 5V supply is used to drive the servos (which require sufficient current), and the ESP32-CAM can be powered via 5V (regulated to 3.3V on board). Stable power is important for the camera and WiFi operation.

These components are connected as follows (GPIO pin assignments on the ESP32-CAM):

Component	ESP32-CAM GPIO	Function
Entry Servo	GPIO 12	Entry gate control (PWM)
Exit Servo	GPIO 2	Exit gate control (PWM)
IR Sensor (Exit)	GPIO 13	Car detection at exit
OLED Display SDA	GPIO 14	I2C data line
OLED Display SCL	GPIO 15	I2C clock line

Table 1: Hardware connections for the smart parking system

The assembly involves mounting the ESP32-CAM overlooking the entry lane, installing the servos on the gate barriers, and positioning the IR sensor so that a leaving car will break its beam. The OLED is attached near the entry for visibility. Care is taken to use separate power supply lines for the servos (5V) and logic (3.3V) to avoid voltage drop issues, and common ground is shared across all components. After hardware setup, the ESP32-CAM is programmed via Arduino IDE with the custom firmware (including the trained ML model library). Once powered on, the device initializes the camera and sensors, connects to WiFi, and starts the web server.

System Operation and Logic

Entry Gate Control Logic:

At the parking entrance, the ESP32-CAM continuously captures images and runs the **Edge Impulse object detection model** to identify objects in view. The logic for granting entry is: the **entry gate opens only if both a car and the authorized token are detected at the same time, and there is at least one parking slot available**. This ensures that only paid vehicles can enter and the lot never exceeds capacity. The AI model was trained to recognize the front of specific cars and the token; it outputs bounding boxes with confidence scores for each detected object. The firmware uses a confidence threshold of 60% – any detection above 0.6 confidence is considered positive. For each camera frame:

1. **Detection Evaluation:** The code checks the inference results for the presence of a `"car"` label and a `"coin"` label with confidence > 0.6. If both are found in the same frame, flags `car_detected` and "`coin_detected`" are set true. Partial detections are also handled: if only a car is detected, the system will prompt that payment is needed; if only a coin is detected, it will prompt that a car is needed.
2. **Gate Opening Condition:**
   - If **car_detected == true** *and* **coin_detected == true** *and* `available_slots > 0` , the entry condition is satisfied. The system then actuates the entry servo to open the gate barrier. The servo rotates from its base closed position (defined as 0°) to the active open position (90°), allowing the vehicle to enter. A message "ENTRY GATE: Opened" is shown on the OLED, and the remaining slots count is decremented by one. Figure 2 demonstrates a scenario with both car and token present, triggering the gate.
   - If the token and car are recognized but `available_slots == 0` (parking full), the system will **deny entry**. In this case, the OLED flashes a "**PARKING FULL - NO ENTRY**" warning. The gate remains closed despite the correct token, as allowing entry would exceed capacity.
   - If only a car is detected (no token), entry is denied and the display shows "**Need Payment**" along with the current free slot count to indicate the car must present the token. Conversely, if only the token is detected (no car), the display shows "**Need Car**", since a token alone is not sufficient. Figures 3 and 4 illustrate these two cases of invalid entry attempts (token-only and car-only, respectively).
   - If no relevant object is detected in view, the system remains idle, showing an "**AI PARKING**" idle status with the number of available slots on the OLED. This baseline message reassures that the system is running and informs how many cars can still enter.

Once triggered to open, the entry gate servo stays open for a fixed duration (e.g. 5 seconds) before automatically closing. This delay is long enough to let the car pass the barrier. After closing, the system

resets the detection flags and continues scanning for the next vehicle. The use of a short open interval and automatic closure ensures the gate is not left open accidentally and that each token is used for only one entry.

Figure 2: A valid entry scenario – the system detects both the car and the authorized token together, which meets the entry criteria. In this case, the entry gate opens to allow the vehicle in. *The OLED display (not shown here) would indicate "CAR ENTRY" and update the remaining slots count.*

Figure 3: An invalid entry attempt – only the payment token is presented without a car. The system recognizes the token but since no vehicle is detected, it keeps the entry gate closed. *The OLED would show "COIN FOUND – Need Car" to prompt that a vehicle must be present for entry.*

Figure 4: Another invalid entry scenario – a car is detected without the token. The gate remains closed because payment verification failed. *The OLED display in this case would show "CAR FOUND – Need Payment", indicating that the coin token must be shown to gain entry.*

Exit Gate Control Logic

The exit side of the parking lot is managed by a simpler sensor-based mechanism. An **IR sensor** is installed near the exit gate to detect a vehicle that is ready to leave. The logic for the exit gate is: the **exit gate opens only if the IR sensor detects a car and the parking lot is not already empty** (i.e., there is at least one car inside to leave). In practice, this means the system maintains a count of cars inside; if that count is zero, an IR trigger alone will not open the gate since presumably no car should be exiting. When a car is departing:

- As a car approaches the exit barrier, it breaks the IR sensor's beam. The ESP32 reads the IR sensor on GPIO 13 and interprets this as an exit request (with a bit of debouncing to avoid false triggers).
- If the current number of occupied slots is less than the maximum (meaning at least one car is in the lot), the system will actuate the **exit servo** (on GPIO 2) to open the exit gate. The servo moves to the open position (90°) allowing the car to depart. The OLED shows an "EXIT GATE: Opened" status during this time for feedback.
- Similar to entry, the exit gate remains open for ~5 seconds before closing automatically. After the delay, the servo returns to the base (closed) position and "EXIT GATE: Closed" is logged. This timing should accommodate a car passing through the gate.
- When the exit gate opens, the system increments the `available_slots` count by one to reflect that a parking space has been freed. For instance, if one car leaves a full lot of 4, the available slots count goes from 0 to 1, and entry for new cars can resume. The OLED is updated to show the new availability. If the lot becomes completely empty (all cars left), the system will

display "0/4 slots occupied" or similar, and it will not open the exit gate again until a car enters (to avoid opening for spurious IR triggers).

Notably, the **exit process does not require token detection or any additional user action** – it is automatic as long as a car is present at the sensor and there is at least one car accounted for in the lot. This ensures a smooth, touchless exit. The independence of the exit logic means a departing car doesn't need to use another token; the entry token was only for access. The separation of entry and exit conditions also prevents abuse (one could not use the exit sensor to try to sneak in, because the camera at entry controls entry separately).

Parking Slot Management

A crucial aspect of the system is tracking the number of cars currently parked to avoid overfilling or other logical errors. The variable `available_slots` (or conversely, occupied count) is updated on every successful entry or exit. The maximum capacity is defined in the firmware (in this prototype, **4 slots** are available). The rules enforced are:

- **Parking Full:** If **available_slots** drops to 0 (meaning the lot has 4/4 cars inside), the system enters a "parking full" state. In this state, even if a car and token are detected at the entrance, the entry gate will not open. The OLED will prominently display "PARKING FULL – NO ENTRY", indicating to the incoming driver that no spaces are available. New entries are only allowed after a car exits and frees up a slot.
- **Parking Empty:** If **available_slots** reaches the maximum (e.g. 4) after successive exits (meaning 0 cars inside), the system knows the lot is empty. In this state, the exit gate will not respond to the IR sensor, since any trigger would likely be a false alarm – there are no cars to leave. Essentially, the exit gate logic requires the count of inside cars to be > 0. This prevents the exit gate from opening unnecessarily (for example, if someone walks past the IR sensor or if it glitches).
- **Real-time Updates:** The current count of available slots is always shown on the OLED in the format
  "Available: X/4" (where X is the number of free spots) during idle state, or for a moment updated as "Entry: Y/4" or "Exit: Z/4" during gate operations to reflect the immediate change. This real-time feedback is useful for both users and any parking attendants to know how many more cars can enter. The count is stored in memory and updated by the functions that handle gate open events. If the device restarts, the count would reset (though in a production scenario it could be stored persistently or recalculated by counting cars via sensors).

In summary, the parking slot management ties together the entry and exit logic by using the count as a gating condition for both. It effectively prevents race conditions or misuse that could violate capacity.

The system thus behaves like an automated parking attendant, only allowing a car in when a spot is free, and keeping track of occupancy without manual intervention.

Edge Impulse AI Model Integration

Model Training: The object detection model at the core of this system was developed using Edge Impulse, a platform for edge ML. The steps followed were: (1) collecting a custom dataset of images containing the target cars and the token in various lighting and orientations, (2) labeling these images with bounding boxes for the classes "car" and "coin", (3) training an object detection model (using a FOMO or TinyYOLOv5 architecture suited for microcontrollers) until it achieved reliable accuracy, and (4) deploying the trained model as an Arduino library. The model was configured to treat detections above 60% confidence as positives to balance sensitivity and precision. Lower thresholds led to false positives (e.g., detecting car-like shapes when none were present), while higher thresholds sometimes failed to recognize the token at slight angles; 60% was empirically chosen as a good compromise.

On-Device Inference: The trained model is imported into the ESP32-CAM firmware (via the *isfar12project-1_inferencing.h* library). On each cycle, after capturing a frame from the camera, the ESP32 performs inference by calling the Edge Impulse runtime. Despite the ESP32-CAM's limited resources, the use of a small model and the external PSRAM allows it to compute results quickly – on the order of 100–200 ms per image for the detection processing. This is efficient enough for real-time gate control. The image is captured in RGB565 format (rather than JPEG) to avoid compression artifacts and improve detection accuracy. The inference result provides a list of bounding boxes with labels and confidence values. The firmware then implements the logic described earlier to decide on gate actuation based on these results.

Detection Logic in Firmware: Pseudocode for the detection handling loop is as follows:

```
capture_image(frame); result =
run_classifier(frame); bool car_detected =
false; bool coin_detected = false; for (box
in result.bounding_boxes) { if (box.value >
0.6) { if (box.label == "car") car_detected
= true; if (box.label == "coin")
coin_detected = true; }
}
if (car_detected && coin_detected && available_slots > 0) {
    open_entry_gate(); // Trigger servo, update slots count
```

```
} else if (car_detected && coin_detected && available_slots == 0) {
    display("PARKING FULL - NO ENTRY");
    // gate remains closed
} else if (car_detected && !coin_detected) {
    display("CAR FOUND - Need Payment");
    // gate remains closed
} else if (!car_detected && coin_detected) {
    display("COIN FOUND - Need Car");
    // gate remains closed
} else {
    display("AI PARKING - Available: X/4");
    // idle state, no action
}
```

This logic (implemented in the `loop()` of the Arduino sketch) ensures the system responds immediately when the correct conditions are met. It also prioritizes safety and clarity – for example, it won't open the gate if another is already open or if a car is currently passing through, by checking flags like `entry_servo_is_active` and `exit_servo_is_active` in the actual code. All detection events and gate actions are also printed to the serial console for debugging, e.g. logging "ENTRY GATE: Opened (AI detected car + coin)".

Web Streaming Interface

One valuable feature of using the ESP32-CAM is its ability to stream video over WiFi. In this project, a simple web interface is provided to monitor the entry gate remotely. Upon startup, the device connects to a predefined WiFi network and starts an HTTP server. The **IP address** of the ESP32-CAM is obtained and displayed on the OLED as a URL (e.g., `http://192.168.XXX.XXX/`) so that users can easily note it. Key aspects of the web interface include:

- **Live Video Feed:** The root URL (`/`) serves a webpage that continually refreshes an image from the camera. The ESP32 captures a frame in JPEG format on each request and serves it to the client. The image is scaled (240x320 resolution) to balance clarity with bandwidth. A refresh meta tag or JavaScript reload ensures roughly a 1 frame-per-second update rate. This allows an attendant or user to see what the camera sees (for example, to verify vehicles and tokens at the gate).
- **Performance Considerations:** The camera is configured to use the lower noise **XCLK frequency** and PSRAM for frame buffering to ensure stable streaming. Each frame capture and conversion to JPEG is done on the fly. If the camera fails to initialize or capture, appropriate error

messages are shown on the serial console and the OLED. The refresh rate (1 FPS) is a trade-off to reduce CPU load and WiFi usage, since the ESP32 is also simultaneously running AI inference. This rate proved sufficient for monitoring purposes.

In essence, the web interface turns the system into a basic **surveillance camera** for the parking entrance. An operator could remotely ensure that only authorized cars are entering (e.g., matching license plates, if visible) and keep an eye on the functionality. This feature, combined with the OLED's on-site display, makes the system user-friendly and transparent.

Results and Demonstration

The AI-powered parking system was tested with a scaled-down prototype: a toy car was used to represent vehicles and a blue RFID key fob as the token. The system behavior in various scenarios was observed to validate the logic:

- **Valid Entry:** When the toy car was placed in view of the camera *together with* the token, the system correctly identified both. The OLED showed "CAR ENTRY" and the gate servo opened the barrier. The car was allowed through, and the slot count decreased. *Figure 2* (above) shows a frame from this scenario – the blue car and token are both present, and the gate opened as indicated. The system printed *"**ENTRY GATE: Opened (AI detected car + coin)**"* to the serial log, confirming the detection.
- **Token-Only Attempt:** When the token was held up to the camera without a car, the system refused entry. It detected the coin (token) with high confidence but since no car was present, it kept the gate closed and showed "**Need Car**" on the display. This matches the expected logic. *Figure 3* illustrates this case – only the token is visible, and indeed the gate did not open. This demonstrates the security aspect that someone can't just use a token without a vehicle (preventing misuse of a found or stolen token).
- **Car-Only Attempt:** When the car was brought into view without the token, the system also denied entry. "**Need Payment**" was displayed on the OLED, indicating the requirement of the coin. *Figure 4* shows the pink toy car in view by itself; the barrier remained closed as designed. This ensures that a vehicle cannot sneak in without the proper token, thus enforcing payment.
- **Parking Full Condition:** The system was tested by simulating the entry of 4 cars sequentially (resetting between tests). After four successful entries, the available slots count reached 0 (full). At this point, the next attempt with a car+token correctly resulted in a "**PARKING FULL – NO ENTRY**" message and no gate action. This was verified by temporarily modifying the code to set `MAX_PARKING_SLOTS` to a lower number for quick testing. The logic worked as expected: no further entries were allowed until an exit occurred.

- **Exit Operation:** The IR sensor was triggered by moving an object in front of it while a car count was non-zero. The exit servo responded by opening the gate for 5 seconds. The OLED updated to "**CAR EXIT**" during this time. The count of available slots incremented after the "car" left. If the lot was empty (count = 4 free slots in a 4-slot system), triggering the IR did nothing – which confirms the *exit blocked when parking empty* rule. The serial log showed "**EXIT GATE: Opened**" and "**EXIT GATE: Closed**" events accordingly.

Overall, the prototype demonstrated reliable performance of the integrated components. The **vision system** was accurate in detecting the token and car within the small test environment. False positives were minimal; the 60% threshold and the distinct appearance of the token helped avoid misclassification. The gate servos operated smoothly and in sync with the detection events, and the occupancy count logic prevented any incorrect entries or exits. The OLED provided clear, concise messages throughout, which would be useful in a real deployment (for example, a driver seeing "PARKING FULL" immediately knows to not enter). The live stream feature worked on the local network, allowing remote viewing of the entry point with about a 1-second refresh interval – enough to monitor vehicles as they pause at the gate. Memory usage was optimized by using the ESP32's PSRAM for image frames and limiting the number of large buffers, avoiding any stability issues due to RAM overflow.

Conclusion and Future Work

In this project, we successfully designed and implemented a smart parking system that leverages edge AI on an ESP32-CAM to control access to a parking facility. The system met all its primary objectives: it only allows entry when both a vehicle and the correct token are present (preventing unauthorized or unpaid access), it automatically opens the exit for departing vehicles while keeping an accurate count of available spaces, and it provides real-time feedback and monitoring to users

Project Achievements: The dual-gate setup operated without conflict, the Edge Impulse model provided robust detections at the edge, and real-time slot management was maintained for up to 4 cars as configured. The integration of a web camera feed and on-device display means the system is not a black box – it increases transparency and trust for users. All these were accomplished on a low-cost hardware platform, demonstrating the feasibility of AI-powered IoT in resource-constrained environments. Key success metrics include 100% adherence to the no-entry rule when conditions weren't met and swift response times for the gate actions (opening/closing in sync with detection).

Future Enhancements: There are several ways to extend or improve this system in future iterations:

- *License Plate Recognition:* Incorporating an OCR or ALPR module could allow the system to read vehicle license plates on entry, enabling logging of which vehicles entered or even

enforcing that only authorized plate numbers can enter. This would add an additional layer of security beyond the token.

- *Digital Payments:* Instead of a physical coin token, a mobile app or RFID card system could be integrated. For example, scanning a QR code or using an NFC card linked to a payment system would modernize the payment verification. The current architecture could accommodate this by replacing or augmenting the token detection with network verification of payment.
- *Cloud Connectivity:* Connecting the system to a cloud service or database could enable remote monitoring of parking usage, analytical dashboards, or a feature for drivers to check slot availability before arriving. Each entry/exit event could be logged to a server for record-keeping or further analysis (e.g., peak usage times).
- *Scaling to Multiple Gates:* In a larger parking lot, multiple entry/exit points could be equipped with such systems. Coordination between them (possibly via a central server or mesh network) would allow managing a larger capacity seamlessly. Also, the AI model could be expanded to detect more types of vehicles or differentiate between multiple tokens if needed for different access levels.
- *Hardware Improvements:* Using higher resolution cameras or more powerful microcontrollers (like ESP32 with more RAM or even Raspberry Pi for heavier ML models) could improve detection accuracy and frame rate. Additionally, adding sensors like ultrasonic rangefinders at each parking spot could enable the system to know exactly which spots are occupied instead of relying solely on count (though the count method proved effective in this prototype).

In conclusion, the AI-based parking system showcases how a blend of machine learning and conventional sensors can create a more secure and efficient parking experience. With further enhancements, such a system could be deployed in small commercial parking lots, residential garages, or campus parking areas, reducing the need for human supervision and streamlining the entry/exit process for drivers. The project highlights the potential of IoT devices like the ESP32-CAM to perform complex tasks (object recognition and autonomous control) at low cost, opening the door for smarter infrastructure in the future.

Scan to visit Github repository