Methodology

The Smart Parking System is implemented through an integration of **IoT hardware**, **computer vision (OCR)**, and a **cloud-connected web platform**. The methodology is divided into **hardware setup**, **software implementation**, **system communication**, **system operation**, and **data management**.

---

1. Hardware Setup

- **Raspberry Pi 5**:
  Controls the **entry and exit lanes**, barrier servos, IR sensors, and the PiCamera for license plate recognition.
- **ESP32 Controller**:
  Handles **parking slot monitoring**, reservation management, and communication with Firebase.
- **IR Sensors**:
  Detect presence of cars at entry, exit, and inside slots.
- **Servo Motors**:
  Operate the entry and exit barriers.
- **RFID/I2C Card Reader**:
  Identifies vehicles and links them to billing.
- **LED Indicators**:
  Show slot status: Available (Green), Occupied (Red), Reserved (Yellow).
- **Firebase Cloud Database**:
  Stores slot states, reservations, billing records, and synchronization logs.
- **Web Dashboard (HTML/JS)**:
  Provides a real-time monitoring and control interface for users.

---

2. Software Implementation

- **Raspberry Pi (Python)**
  - `RPi.GPIO`: For IR sensors and servo motors.
  - `Picamera2 + OpenCV`: For number plate detection and preprocessing.
  - `pytesseract`: For OCR-based license plate recognition.
  - `smbus2`: For I2C communication with ESP32 (card IDs).
- **ESP32 (Arduino C++)**
  - `Firebase_ESP_Client`: For Firebase communication.
  - `WiFi.h`: Internet connectivity.
  - `Wire.h`: I2C data exchange with Raspberry Pi.
  - `WebServer.h`: Hosts a local web interface.

- **Frontend (Web Dashboard)**
  - HTML, CSS, JavaScript with Firebase SDK.
  - Real-time listeners update slot status and billing information instantly.
  - Provides reservation form and displays invoices.

---

3. System Communication

- **I2C Communication (Raspberry Pi ↔ ESP32)**
  - Card IDs are transmitted for entry and exit validation.
- **Wi-Fi + Firebase (ESP32 ↔ Cloud)**
  - ESP32 continuously updates slot status and billing info to Firebase.
  - Receives reservation requests from Firebase.
- **Local Web Server (ESP32)**
  - Hosts a dashboard for monitoring and sending commands when offline.

---

4. System Operation

Entry Process

1. Vehicle detected by **IR sensor at Lane 1**.
2. Barrier closes → Raspberry Pi starts **camera capture**.
3. Number plate is detected with **OpenCV + OCR**.
4. Raspberry Pi requests **card ID** via I2C.
5. If new vehicle → Added to `inside_cars` set → Barrier opens → Slot marked Reserved/Occupied in Firebase.
6. If duplicate card ID → Entry rejected.

Exit Process

1. Vehicle detected by **IR sensor at Lane 2**.
2. Barrier closes → Card ID requested.
3. If ID exists in `inside_cars` → Removed → Exit barrier opens → Billing triggered.
4. If ID not found → Exit denied.

Slot Monitoring

- Each slot has a sensor connected to ESP32.
- ESP32 updates Firebase with **Available/Reserved/Occupied** status.
- LED indicators and web UI update instantly.

Reservation Handling

- User sends reservation via **Web Dashboard form**.
- Request stored in Firebase (`/reservationRequest`).
- ESP32 validates and responds with “Accepted” or “Rejected”.
- Reserved slot turns Yellow until the car arrives.

Billing

1. Entry and exit times recorded using **NTP time sync**.
2. Bill calculation:
3. `Total = Reservation Fee + (Parking Duration × Rate per Hour)`
4. Bill pushed to Firebase under `/bills/last` and `/bills/history`.
5. Web dashboard displays popup invoice for the user.

---

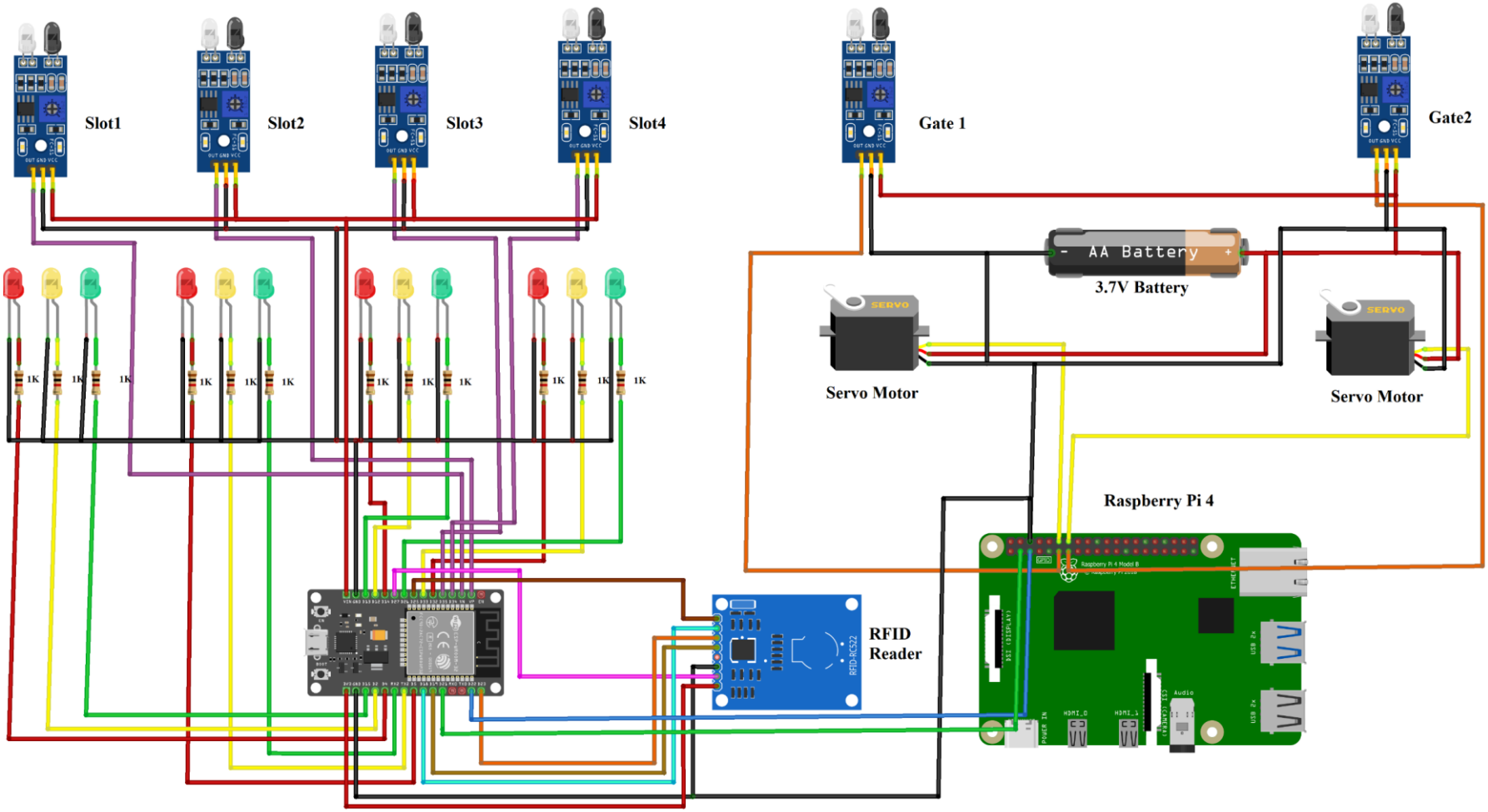
5. Data Management

- **Local Storage (Raspberry Pi)**: Saves detected plate images in `/plates/`.
- **Firebase Database**:
  - `/slots` → slot availability.
  - `/carNumbers` → car ID mapping.
  - `/reservationRequest` → reservation requests.
  - `/bills` → billing history.

---

6. Workflow Summary

1. Vehicle approaches entry → IR detects → Barrier closes → Camera OCR reads plate.
2. Card ID verified → If valid → Barrier opens → Slot updated → Billing starts.
3. Vehicle exits → IR detects → Card ID verified → Barrier opens → Billing ends.
4. Firebase syncs all status updates and bills.
5. User dashboard shows **real-time slot map, reservations, and invoices**.


Slot1
Slot2
Slot3
Slot4
Gate 1
Gate2
AA Battery
3.7V Battery
Servo Motor
Servo Motor
SERVO
1K
RFID
Reader
Raspberry Pi 4