IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511148

Design And Implementation of An IOT-Based Smart Parking Slot Monitoring System With Cloud-Enabled Real-Time Accessibility

Pruthviraj Gaikwad[1], Akshay Patil[2], Sushant Bandgar[3], Rohit Keche[4]

Department of Computer Science and Engineering, S. K. N. Sinhgad College of Engineering, Korti, Pandharpur, Maharashtra, India[1-4]

Abstract: Parking-space search in colleges and other confined premises can cause unnecessary vehicle movement and congestion when drivers do not know which spaces are vacant. This paper presents Slotix, an Internet of Things (IoT)-based smart parking slot monitoring prototype that detects the occupancy of individual parking spaces and publishes the information through a cloud-connected dashboard. Four infrared (IR) sensors monitor four parking slots, identified as S01–S04. An ESP8266 NodeMCU reads the sensor outputs, calculates occupied and available slot counts, and sends the latest status to Firebase Realtime Database through Wi-Fi. A responsive dashboard retrieves the cloud data so that the same parking information can be viewed from multiple Internet-connected devices. The current implementation focuses on slot-level monitoring and cloud synchronization. Initial functional testing confirmed Wi-Fi connectivity and a successful Firebase write response (HTTP 200). Comprehensive measurements of detection accuracy, latency, and long-duration reliability remain to be completed. The design provides a low-cost and modular foundation for small parking areas.

Keywords: Internet of Things, Smart Parking, ESP8266, Infrared Sensor, Firebase Realtime Database, Cloud Dashboard, Real-Time Monitoring.

I. INTRODUCTION

Parking management is a common operational challenge in colleges, offices, hospitals and commercial premises. When drivers enter a parking area without knowing which spaces are vacant, they may spend additional time moving through the premises and searching for an available slot. This can contribute to congestion, unnecessary vehicle movement and inefficient use of available space.

Traditional parking areas often depend on visual inspection or manual supervision. Such approaches may be adequate for small facilities but do not continuously communicate slot-level information to users. A monitoring system that detects the status of each parking space and makes the information available through a shared dashboard can improve visibility and reduce uncertainty.

Slotix is designed as a compact IoT-based parking-slot monitoring prototype for an institutional parking environment. The current prototype monitors four slots, identified as S01, S02, S03 and S04. One IR sensor is assigned to each slot. The ESP8266 reads the sensor outputs, determines whether each slot is occupied or available, calculates aggregate counts and sends the information to Firebase Realtime Database over Wi-Fi.

The dashboard retrieves the cloud data and displays the status of all four slots. Since the data is stored in a cloud database rather than only on a local computer, authorized users can access the same dashboard from different phones or computers with an Internet connection. This paper focuses on slot occupancy monitoring, cloud data transfer and dashboard visibility.

II. RELATED WORK

IoT-based parking solutions generally combine sensing devices, embedded controllers, communication networks and software interfaces. Sensors installed at or near parking spaces identify vehicle presence. A controller processes the sensor readings and communicates the resulting status to a local server or cloud platform. The user interface then presents availability information to drivers or parking administrators.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511148

Different sensing approaches include ultrasonic ranging, infrared sensing, magnetic sensing and camera-based detection. Selection depends on cost, installation conditions, detection range, environmental effects and maintenance requirements. IR sensors provide a simple digital output that can be read by a low-cost microcontroller. However, their performance can be affected by sensor alignment, surface reflectivity, ambient light and installation distance.

Cloud platforms provide a shared data layer for connected devices and applications. Firebase Realtime Database is used in this project to store structured data and make updates available to connected clients. The ESP8266 provides Wi-Fi connectivity and embedded processing in a compact development board. Slotix combines these technologies in a small four-slot prototype.

A formal comparative review of published smart-parking implementations, with verified publication details and measured results, should be added before journal submission. This draft does not claim unverified research-paper citations.

III. RESEARCH GAP AND PROBLEM DEFINITION

A. Research Gap

A practical gap addressed by this prototype is the need for a simple and economical slot-level monitoring arrangement that connects low-cost sensors to a cloud database and exposes a common live view to multiple devices. Some small parking facilities continue to rely on manual observation or isolated local displays. A local-only implementation can limit access when users are away from the premises or when more than one device needs to view the same status.

Slotix addresses this focused requirement by combining four IR sensors, an ESP8266 Wi-Fi controller, Firebase Realtime Database and a responsive dashboard. The contribution is an integrated prototype and implementation workflow rather than a claim that smart parking or cloud-based monitoring is a new concept.

B. Problem Statement

To design and implement a low-cost IoT-based system that detects the occupancy status of individual parking slots, transfers the detected status through an ESP8266 controller to a cloud database, and displays the latest slot availability on a dashboard accessible from multiple Internet-connected devices.

C. Objectives

To develop a four-slot parking monitoring prototype using IR sensors.

To interface the sensors with an ESP8266 NodeMCU controller.

To determine and maintain the occupied or available status of each slot.

To transmit slot status and aggregate counts to Firebase Realtime Database through Wi-Fi.

To provide a responsive dashboard that reads and displays cloud data.

To enable the same parking information to be viewed from multiple connected devices.

To evaluate sensor detection, cloud update behaviour, synchronization and reliability.

IV. PROPOSED SYSTEM

The proposed Slotix system consists of four IR sensors, an ESP8266 NodeMCU controller, a Wi-Fi connection, Firebase Realtime Database and a web dashboard. Each sensor is associated with one parking slot. The controller periodically reads the digital output of each sensor. In the current firmware, a LOW reading is interpreted as occupied and a HIGH reading as available. This logic must be verified with the actual sensor module because some modules provide the opposite output polarity.

After reading all sensors, the controller calculates the number of occupied slots and derives the number of available slots from the total of four. It constructs a JSON object containing the status and sensor identifier for S01–S04, together with the occupied and available counts. The JSON object is sent to Firebase Realtime Database using an HTTPS PUT request to the parking data node.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511148

The dashboard reads the parking node from Firebase and renders the latest status for each slot. A green indicator can represent an available slot and a red indicator can represent an occupied slot. Aggregate counters summarize the number of available and occupied spaces. Cloud storage allows multiple dashboard clients to access a common data source rather than depending on the IP address of a local development computer.

V. SYSTEM ARCHITECTURE



Fig. 1 Architecture of the Slotix cloud-connected parking-slot monitoring system.

The architecture follows a data acquisition and cloud visualization flow. First, each IR sensor detects whether an object is present in its assigned parking space. Second, the ESP8266 reads the sensor outputs and forms the slot-status data. Third, the controller uses Wi-Fi to send the data to Firebase Realtime Database. Finally, the Slotix dashboard reads the stored data and displays the current status to connected users.

TABLE I HARDWARE AND SOFTWARE COMPONENTS

Component	Purpose in Slotix
IR sensors (4)	Detect vehicle presence at slots S01–S04.
ESP8266 NodeMCU	Reads sensor outputs, calculates slot counts and communicates over Wi-Fi.
Wi-Fi hotspot/router	Provides Internet connectivity to the controller.
Firebase Realtime Database	Stores latest slot status and aggregate counts in the cloud.
Slotix web dashboard	Displays slot status and availability on phones and computers.
Arduino IDE	Used to develop, compile and upload firmware to the ESP8266.

VI. HARDWARE AND SOFTWARE IMPLEMENTATION

A. IR Sensor Module

Four IR sensor modules are used, with one sensor assigned to each parking slot. The sensor output changes according to the reflected infrared signal and the detection characteristics of the module. Correct mounting position and calibration are important. The sensor should be aligned with the intended detection area and tested under the lighting and surface conditions of the installation site. The module's output polarity must be checked before operation.

B. ESP8266 NodeMCU

The ESP8266 acts as the main controller. It provides digital input pins for the four IR sensors and Wi-Fi connectivity for cloud communication. The firmware initializes the sensor pins, connects to the configured wireless network, reads each sensor, calculates occupied and available counts and prepares the JSON payload. The current pin assignment is S01 to D1, S02 to D2, S03 to D5 and S04 to D6.

C. Firebase Realtime Database

Firebase Realtime Database is used as the cloud data store. The project database is configured with project identifier slotix-a779f and endpoint https://slotix-a779f-default-rtdb.firebaseio.com. The firmware writes the parking object to the /parking node. The web dashboard uses the same database project to retrieve the latest values. Database security rules and appropriate authentication must be configured before public or production use.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511148

D. Dashboard

The Slotix dashboard provides a visual summary of the four monitored slots. Each slot is represented separately so users can identify which space is occupied or available. Aggregate counters summarize the number of available and occupied spaces. The dashboard is intended to be responsive for mobile phones, tablets and desktop computers.

VII. METHODOLOGY

1) Sensor installation: Mount one IR sensor at each of the four parking slots and connect the sensor output, supply and ground to the corresponding ESP8266 pins and common ground.

2) Controller configuration: Select the correct ESP8266 board and serial port in Arduino IDE. Configure the Wi-Fi credentials and Firebase Realtime Database endpoint in the firmware.

3) Sensor calibration: Test each sensor independently with and without a vehicle or suitable test object. Confirm whether LOW or HIGH represents detection for the installed module and adjust the code if required.

4) Data processing: Read all four digital inputs, assign the slot status, count occupied slots and calculate available slots as four minus the occupied count.

5) Cloud transmission: Serialize the slot states and summary counts into JSON and send the payload to the Firebase /parking.json endpoint using an HTTPS PUT request.

6) Dashboard synchronization: Configure the dashboard to read the parking data from Firebase and update the displayed slot cards and summary counters when cloud data changes.

7) Functional validation: Test each slot individually, combinations of occupied slots, all-empty and all-occupied conditions, Wi-Fi interruption and recovery, cloud write responses and access from multiple dashboard clients.

VIII. DATABASE STRUCTURE AND DATA FLOW

The current firmware stores the latest state of the four slots under the parking node. Each slot contains a status field and a sensor identifier. The same node also contains the available and occupied integer counts. A simplified representation is shown below.

```
parking/
 S01/ status: "available", sensor: "IR1"
 S02/ status: "occupied", sensor: "IR2"
 S03/ status: "available", sensor: "IR3"
 S04/ status: "available", sensor: "IR4"
 available: 3
 occupied: 1
```

The values above are an illustrative example of the data structure, not a claim about a measured parking event. The current firmware overwrites the /parking node with the latest snapshot. It therefore maintains current status but does not by itself create a historical event log. A separate history node and timestamped event-writing logic would be required for historical analysis.

IX. TESTING AND PERFORMANCE EVALUATION

The prototype should be evaluated using repeatable tests so that sensor behaviour and cloud communication can be reported quantitatively. Each sensor should be tested over multiple trials with the parking space empty and occupied. Cloud transmission should be tested by recording the HTTP response code and request-response duration. Dashboard synchronization should be checked by comparing the update observed on more than one connected device.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511148

TABLE II FUNCTIONAL TEST CASES

Test ID	Test condition	Expected result
TC-01	All four slots empty	Four available; zero occupied.
TC-02	Only S01 occupied	S01 occupied; three available.
TC-03	Only S02 occupied	S02 occupied; three available.
TC-04	Only S03 occupied	S03 occupied; three available.
TC-05	Only S04 occupied	S04 occupied; three available.
TC-06	Two slots occupied	Two occupied; two available.
TC-07	All four slots occupied	Four occupied; zero available.
TC-08	Successful Firebase write	HTTP 200 response from database endpoint.
TC-09	Wi-Fi interruption	Controller detects disconnection and attempts reconnection.
TC-10	Two dashboard clients	Both clients display the same latest cloud status.

During initial functional testing, the ESP8266 connected to the configured Wi-Fi network and a Firebase write returned HTTP 200, indicating that the database accepted the request. This confirms basic connectivity and write functionality for that test instance. It does not establish a general detection-accuracy percentage, average latency, long-term reliability or multi-device synchronization rate. Those values must be measured using the final assembled prototype.

TABLE III MEASUREMENTS TO BE RECORDED

Metric	Measurement method	Result
Occupancy detection accuracy	Correct sensor decisions / total trials × 100	To be measured
False detection rate	Incorrect sensor decisions / total trials × 100	To be measured
Firebase update latency	Elapsed time for each cloud write	To be measured
Cloud update success rate	Successful writes / total write attempts × 100	To be measured
Dashboard update delay	Sensor change to visible dashboard update	To be measured
Wi-Fi recovery time	Network restoration to successful reconnection	To be measured
Multi-device consistency	Compare status on simultaneously connected clients	To be measured

X. RESULTS AND DISCUSSION

The current implementation demonstrates the end-to-end data path from IR sensor inputs to the ESP8266 controller and onward to Firebase Realtime Database. The firmware is configured for four individually identified parking slots and calculates available and occupied totals from the sensor readings. A successful HTTP 200 response was observed during an initial Firebase write test, showing that the controller could communicate with the configured database endpoint.

The present evidence is sufficient to report basic firmware-to-cloud connectivity, but not to claim a validated system accuracy or performance benchmark. Sensor calibration, repeated occupancy trials, network-delay measurements, dashboard update measurements and simultaneous multi-device tests should be completed and documented before final publication. Results should include the test date, number of trials, test environment and any sensor-specific calibration changes.

IR-based detection is inexpensive and straightforward to implement, but it is sensitive to physical placement and environmental conditions. A stable installation should use secure sensor mounting, suitable shielding where necessary, consistent power supply and repeated calibration. Cloud access also depends on Wi-Fi and Internet connectivity. If the



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511148

connection is lost, the dashboard may continue to show the last successfully uploaded state until a new update is received.

XI. LIMITATIONS AND FUTURE SCOPE

The current prototype is limited to four parking slots and uses digital IR sensor outputs. Detection performance may vary with vehicle shape, sensor mounting, reflective surfaces, ambient light and installation geometry. The system depends on Wi-Fi and cloud service availability for remote dashboard updates. The current firmware writes the latest snapshot and does not maintain a timestamped event history. The prototype also requires appropriate Firebase security rules and authentication before it is exposed for general use.

Future work may include expanding the number of monitored slots, adding timestamped occupancy history, improving sensor filtering and fault detection, implementing secure device authentication, recording offline data for later synchronization and conducting longer field trials. The dashboard can also be enhanced with administrative controls and reports while retaining the core objective of accurate slot-level monitoring.

XII. CONCLUSION

This paper presented Slotix, an IoT-based smart parking slot monitoring prototype that combines four IR sensors, an ESP8266 NodeMCU controller, Wi-Fi connectivity, Firebase Realtime Database and a responsive dashboard. The system reads the status of individual parking slots, calculates occupied and available counts and uploads the latest state to a cloud database. The dashboard is designed to make the same parking information accessible from multiple Internet-connected devices.

Initial testing confirmed that the ESP8266 could connect to Wi-Fi and successfully write data to Firebase, as indicated by an HTTP 200 response. Further controlled testing is required to quantify detection accuracy, update latency, reliability and multi-device synchronization. The proposed architecture provides a practical starting point for low-cost parking monitoring in small institutional parking areas and can be extended through additional sensors and software capabilities.

ACKNOWLEDGMENT

The authors express their sincere gratitude to Prof. B. B. Jagdale, Project Guide, and the Department of Computer Science and Engineering, S. K. N. Sinhgad College of Engineering, Korti, Pandharpur, for their guidance and support during the development of the Slotix project.

REFERENCES

[1]. Espressif Systems, “ESP8266EX Datasheet,” Espressif Systems. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
[2]. Firebase, “Firebase Realtime Database,” Firebase Documentation. [Online]. Available: https://firebase.google.com/docs/database
[3]. Arduino, “Arduino IDE Documentation,” Arduino Documentation. [Online]. Available: https://docs.arduino.cc/software/ide/
[4]. ESP8266 Arduino Core, “ESP8266 Arduino Core Documentation.” [Online]. Available: https://arduino-esp8266.readthedocs.io/
[5]. Firebase, “Read and Write Data on the Web,” Firebase Documentation. [Online]. Available: https://firebase.google.com/docs/database/web/read-and-write
[6]. Firebase, “Realtime Database Security Rules,” Firebase Documentation. [Online]. Available: https://firebase.google.com/docs/database/security

 This work is licensed under a Creative Commons Attribution 4.0 International License