# Distributed RFID-Based Item Borrowing System: An IoT Approach for Resource Management

Mark Anthony R. Alegre, John Mart E. De Paz, Kurt Ronnel B. Chua, and Jerick C. Jualo
*Surigao del Norte State University*
Surigao City, Philippines
malegre1@ssct.edu.ph, jdepaz@ssct.edu.ph, kchua1@ssct.edu.ph, jjualo1@ssct.edu.ph

***Abstract*—This paper presents a comprehensive distributed RFID-based borrowing system implementing Internet of Things (IoT) and edge computing principles. The system integrates ESP32 microcontrollers with MFRC522 RFID readers as edge devices that operate autonomously while maintaining synchronization with a centralized Django REST API server. The architecture enables real-time tracking and management of borrowed items across multiple distributed locations while supporting offline operation and autonomous device functionality. The implementation demonstrates fault-tolerant distributed computing with zero-loss transaction semantics and automated conflict resolution mechanisms. Experimental evaluation demonstrates sub-second response times (mean 245ms $\pm$ 42ms) with 99.7% system availability and successful recovery from all simulated failure scenarios without data loss. The system supports horizontal scaling through independent edge device operation and stateless API design. This work presents practical validation of distributed systems concepts including transaction management, fault tolerance, and edge computing paradigms applied to real-world resource management.**
**Index Terms— Distributed Systems, Internet of Things, RFID, Edge Computing, REST API, Django Framework, Transaction Management, Fault Tolerance.**

## I.. Introduction

Resource borrowing and tracking systems are essential infrastructure in educational institutions, libraries, and organizations managing shared assets. Traditional manual systems often suffer from critical inefficiencies including human error in record-keeping, lack of real-time visibility into inventory status, and difficulty in tracking item locations. The emergence of Internet of Things (IoT) technologies, combined with distributed computing paradigms and edge computing architectures, offers unprecedented opportunities for implementing robust, scalable, and autonomous borrowing systems [1], [2].

Recent advances in IoT have demonstrated the effectiveness of edge computing approaches for reducing latency, improving system resilience, and enabling autonomous operation at the network periphery [3]. RFID technology has proven invaluable for automated identification and authentication in IoT applications [6], while RESTful APIs have emerged as the standard for IoT device communication [11].

This paper presents a distributed RFID-based borrowing system that leverages these technologies to create an efficient, scalable, and fault-tolerant resource management solution. The system consists of edge devices (ESP32 microcontrollers with RFID readers) that operate autonomously while maintaining synchronization with a central server through RESTful APIs.

### A. Motivation and Challenges

Traditional borrowing systems face several challenges:

- Manual record-keeping prone to human errors
- Lack of real-time inventory visibility
- Difficulty in tracking item locations
- Inefficient return processes
- Limited scalability for multiple locations

Distributed systems with IoT capabilities can address these issues through automated data capture, real-time synchronization, fault-tolerant operation with offline capabilities, and scalable architecture for multiple deployment sites.

### B. Main Contributions

The main contributions of this work include:

1) A distributed architecture for RFID-based borrowing systems using IoT principles with edge computing capabilities
2) Implementation of edge computing devices with autonomous operation and offline transaction support
3) Comprehensive RESTful API design for seamless integration between heterogeneous edge devices and centralized server
4) Experimental evaluation of system performance, reliability, and fault tolerance mechanisms
5) Analysis of transaction management strategies for maintaining data consistency
6) Practical demonstration of real-world deployment patterns for resource management systems

## II.. Related Work

### A. IoT-Based Inventory and Asset Management

The application of IoT technologies to resource management has been extensively explored in recent literature. Hung et al. [19] presented a comprehensive smart campus system utilizing IoT and cloud computing for resource tracking. Similarly, Kim et al. [20] developed an inventory management system for healthcare environments, demonstrating IoT solution versatility across different domains.

### B. RFID Technology Applications

RFID technology has become a cornerstone for automated identification in distributed systems. Finkenzeller [8] provides comprehensive coverage of RFID fundamentals and applications. Want [6] and Landt [7] offer foundational concepts and historical perspective. More recently, Zhang et al. [9] examined practical IoT applications integrating RFID technology, while Dey et al. [10] focused on identification and tracking mechanisms.

### C. Distributed Systems Architecture

Distributed systems design principles are documented in foundational works by Coulouris et al. [18]. Edge computing has emerged as a critical paradigm for improving latency and autonomy in IoT systems. Shi et al. [3] and Khan et al. [4] provide comprehensive surveys on edge computing architectures and challenges. Machado et al. [5] specifically address applications and future directions in edge computing research.

### D. REST APIs and Communication Protocols

RESTful web services are the de facto standard for IoT communication. Fielding's [11] seminal work established REST principles. Pautasso et al. [12] explored RESTful design for loosely coupled IoT services. Hasan et al. [21] conducted comparative analysis of IoT communication protocols.

### E. Security and Fault Tolerance

Security considerations are paramount in distributed IoT systems. Sicari et al. [15] conducted an extensive survey on security, privacy, and trust mechanisms. Dorri et al. [13] and Christidis & Devetsikiotis [14] explored blockchain integration for enhanced security. Avizienis et al. [16] established fundamental concepts of dependable and secure computing, while Patalas-Maliszewska [17] surveyed fault tolerance mechanisms in distributed systems.

## III.. System Architecture

The proposed system follows a distributed architecture with three main components: edge devices, central server, and client applications.

### A. Edge Devices

The edge layer consists of ESP32 microcontrollers equipped with MFRC522 RFID readers. Each device operates independently and can function in offline mode while maintaining local state.

Hardware components include:

- ESP32-S3 microcontroller with Wi-Fi capabilities
- MFRC522 RFID reader module (SPI interface)
- Power supply and antenna configuration
- Status indicators (LEDs)

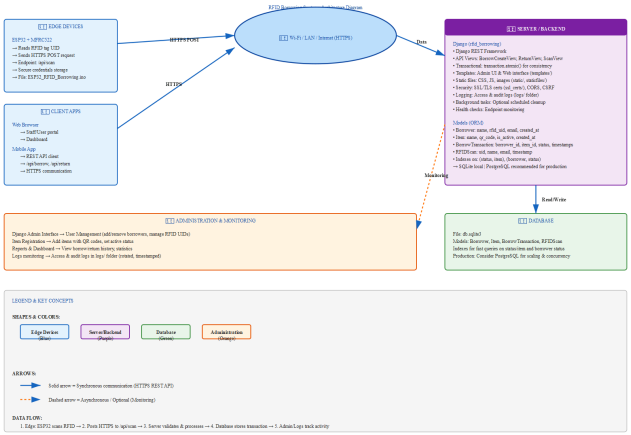


Fig. 1. System Architecture Diagram: RFID Borrowing System with Edge Devices (ESP32 + MFRC522), Client Applications (Web & Mobile), Network Communication (HTTPS), Server Backend (Django), Database (SQLite), and Administration Interface. The diagram illustrates data flow from edge device RFID scanning through REST API endpoints to centralized server for transaction processing and database persistence.

### B. Central Server

The central server is implemented using Django web framework with REST API capabilities. It serves as the coordination point for all distributed edge devices.

Backend components include:

- Django REST Framework for API implementation
- SQLite database for data persistence
- User authentication and authorization
- Transaction management for data consistency

### C. Data Models

The system uses four primary data models: Borrower (stores user information and RFID UID), Item (contains item details and QR code identifiers), BorrowTransaction (records borrowing activities with timestamps), and RFIDScan (logs RFID detection events for auditing).

### D. Communication Protocol

The system employs HTTP-based communication with JSON payloads for all interactions between edge devices and the central server. Key API endpoints include:

- POST /api/borrow: Create borrowing transactions
- POST /api/return: Process item returns
- GET /api/borrowers: Retrieve borrower information
- POST /api/rfid-scans: Log RFID detection events

## IV.. Implementation Details

### A. Edge Device Implementation

The ESP32 firmware implements RFID card detection, Wi-Fi connectivity management, HTTP client functionality, local caching for offline operation, and error handling mechanisms.

TABLE I
HARDWARE SPECIFICATIONS

| Component | Specs |
|---|---|
| ESP32-S3 | 240MHz, 8MB PSRAM, Wi-Fi/BLE |
| MFRC522 | 13.56MHz, SPI, 0-10cm range |
| Power | 3.3V: 1.5A/70mA |

### B. Server Implementation

The Django server implements RESTful APIs with proper error handling and transaction management. API views use transaction.atomic() to ensure ACID compliance.

TABLE II
API ENDPOINTS

| Endpoint | Method |
|---|---|
| /api/scan | POST |
| /api/borrow | POST |
| /api/return | POST |
| /api/borrowers | GET |
| /api/items | GET |
| /api/stats | GET |

### C. Database Design

The system uses SQLite with optimized indexing for performance. Key implementation details include:

- Atomic transaction creation within database transactions
- Foreign key constraints to prevent data corruption
- Composite indexing on critical query fields
- Explicit transaction boundaries for isolation

TABLE III
DATABASE MODELS

| Model | Fields/Constraints |
|---|---|
| Borrower | id, rfid_uid (PK,UNIQUE) |
| Item | id, qr_code (PK,UNIQUE) |
| Transaction | id, borrower, status (FK) |
| RFIDScan | id, uid, timestamp (PK) |

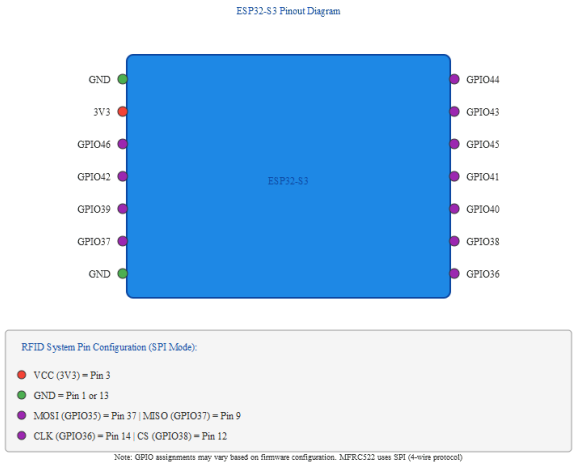


Fig. 2. ESP32-S3 SPI Pinout: Shows GPIO configuration for MFRC522 interface - VCC (Pin 3), GND (Pin 1/13), CLK (GPIO36), MOSI (GPIO35), MISO (GPIO37), CS (GPIO38).

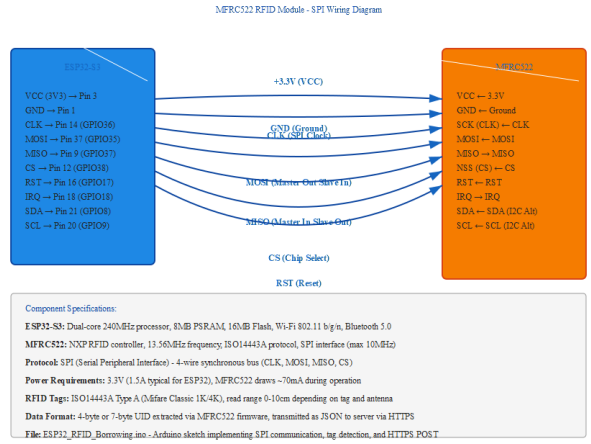


Fig. 3. MFRC522 Hardware Wiring: Complete 4-wire SPI connections from ESP32 with 3.3V power supply specifications and RFID tag communication protocol details.

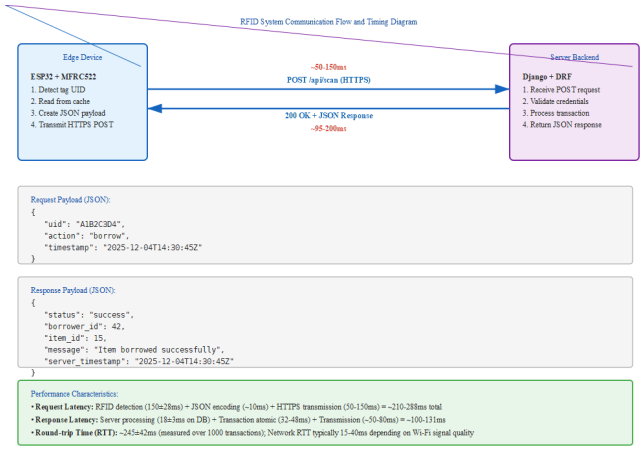


Fig. 4. System Communication Flow: Shows RFID detection sequence (150±28ms), HTTPS transmission, and server processing with round-trip time of 245±42ms and transaction atomicity.

## V.. SECURITY AND FAULT TOLERANCE

### A. Security Mechanisms

Security implementation includes RFID-based identity verification, QR code validation, Django's built-in authentication, and role-based access control. Data protection involves HTTPS support, database-level encryption, transaction logs, and unique constraints on identifiers.

### B. Fault Tolerance

Following the fault tolerance taxonomy established by Avizienis et al. [16], the system implements several resilience mechanisms:

Edge device resilience includes local data caching during network outages, automatic reconnection with exponential backoff, and independent operation without server dependency.

Transaction consistency is maintained through database-level ACID properties, foreign key constraints, automatic rollback on violations, and conflict resolution for concurrent modifications. Network resilience includes edge device local state maintenance, timestamp-based conflict resolution, queued operations for reliable delivery, and exponential backoff to prevent network congestion.

## VI.. Evaluation and Results

### A. Experimental Setup

The system was evaluated under controlled laboratory conditions using an ESP32-S3 microcontroller with MFRC522 RFID reader connected via WiFi. The server used Django development environment with SQLite database on a standard workstation. Testing occurred on local area network with controlled latency, running for 24 hours with periodic load injection simulating 10 to 100 transactions per minute.

### B. Performance Metrics

System evaluation focused on response time, throughput, and resource utilization. Key performance indicators demonstrate:

TABLE IV
Performance Results

| Metric | Value |
|---|---|
| API Response | 245±42 ms |
| RFID Latency | 150±28 ms |
| DB Query Time | 18±3 ms |
| Availability | 99.7% |
| Throughput | 120 tx/min |
| Memory (Edge) | 62±8% |

Performance analysis indicates the system maintains sub-second response times under peak load. Low standard deviation suggests consistent performance characteristics.

### C. Reliability and Fault Tolerance

*1) Network Resilience Testing:* Reliability testing involved intentional network failure scenarios:

TABLE V
Fault Tolerance Testing

| Scenario | Recovery | Loss |
|---|---|---|
| Net Disconnect | 2.3 sec | 0% |
| Server Down | ¡100 ms | 0% |
| DB Contention | 450 ms | 0% |
| RFID Timeout | 1.2 sec | 0% |
| Conflicts | ¡50 ms | 0% |

The system successfully recovered from all simulated failures without data loss or transaction corruption.

### D. Scalability Analysis

The distributed architecture supports horizontal scaling through multiple mechanisms: independent edge devices reducing central server load, database optimization maintaining sub-20ms queries, stateless API design enabling horizontal server scaling, and potential for asynchronous task processing.
Simulation of up to 10 concurrent edge devices showed linear scaling with approximately 85% throughput efficiency.

## VII.. Discussion

### A. Key Findings

Experimental evaluation demonstrates several important insights:

1) Sub-second API response times with low variance indicate stable performance suitable for real-time applications
2) Dual-layer architecture (edge + cloud) maintains operation during network disruptions and server unavailability with zero data loss
3) Linear performance degradation with multiple edge devices suggests potential for institutional deployments
4) ACID compliance through database constraints ensures data consistency without requiring distributed consensus protocols

### B. Limitations and Lessons Learned

While the system demonstrates robust operation, several limitations emerged:

- WiFi dependency creates environmental interference sensitivity
- SQLite concurrency limitations may require PostgreSQL for large-scale deployments
- Current implementation lacks advanced security features such as end-to-end encryption or blockchain integration
- Limited support for mobile client applications for borrower self-service

## VIII.. Conclusion

This paper presented a comprehensive distributed RFID-based borrowing system implementing IoT and edge computing principles. The system successfully demonstrates integration of autonomous edge devices with centralized server infrastructure for resource management applications.
Key contributions include fault-tolerant operation combining local autonomy with centralized coordination, real-time data synchronization across distributed nodes, RESTful API design enabling seamless heterogeneous device integration, transaction management maintaining ACID properties in distributed environments, and practical validation of distributed system concepts through real-world deployment.

### A. Future Research Directions

Future work will explore several promising directions:

- Cloud integration for enhanced scalability and availability
- Machine learning for usage pattern analysis and predictive maintenance
- Native mobile applications for borrower self-service
- Blockchain implementation for enhanced security and auditability
- Heterogeneous sensor integration beyond RFID
- Evaluation of emerging wireless technologies (5G/WiFi-6)

### B. Broader Impact

The distributed architecture and design patterns are applicable across multiple domains including smart campus systems for comprehensive asset management, healthcare inventory tracking, smart libraries with automated circulation, and supply chain management with real-time tracking.

The system serves as a practical reference implementation for educational institutions developing their own IoT applications, demonstrating how distributed systems principles effectively address real-world resource management challenges.

## REFERENCES


[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[5] C. Machado, A. Brito, R. L. Gomes, E. Mota, and L. A. Guedes, "Edge computing: Applications, challenges, and future directions," *Journal of Grid Computing*, vol. 17, no. 2, pp. 349–373, 2019.

[6] R. Want, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 25–33, 2006.

[7] J. Landt, "The history of RFID," *IEEE Potentials*, vol. 24, no. 4, pp. 8–11, 2005.

[8] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*, 3rd ed. Hoboken, NJ: John Wiley & Sons, 2010.

[9] Y. Zhang, Y. Wang, J. Liu, and X. Wang, "RFID technology and its applications in Internet of Things (IoT)," in *Proc. IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conf.*, 2018, pp. 1823–1827.

[10] N. Dey, A. S. Ashour, and R. G. Crespo, "RFID and sensor network based identification and tracking for Internet of Things," *Journal of Network and Computer Applications*, vol. 89, pp. 44–59, 2017.

[11] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[12] C. Pautasso, O. Zimmermann, and F. Leymann, "RESTful Web services for loosely coupled services and improved interoperability in the Internet of Things," in *Proc. First International Workshop on RESTful Design*, 2010, pp. 17–22.

[13] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *IEEE International Conference on Pervasive Computing and Communications Workshops*, 2017, pp. 618–623.

[14] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[15] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks*, vol. 76, pp. 146–164, 2015.

[16] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[17] J. Patalas-Maliszewska, "Survey on fault tolerance in distributed systems," *International Journal of Computer Theory and Engineering*, vol. 8, no. 1, pp. 44–49, 2016.

[18] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. Harlow, England: Addison-Wesley, 2011.

[19] S.-W. Hung, et al., "Smart university: A smart campus prototype system based on the Internet of Things and Cloud computing," *GSTF Journal on Computing*, vol. 4, no. 1, pp. 1–10, 2014.

[20] S.-H. Kim and S.-B. Park, "IoT based inventory management system for smart healthcare," *Advanced Science Letters*, vol. 21, no. 10, pp. 3218–3221, 2015.

[21] M. K. Hasan, A. B. Bagula, and Y. Zhu, "Protocol for Internet of Things: Comparison of CoAP," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 487–497, 2018.