An RFID-Based Smart Inventory Management System for Small-Scale Industries Using ESP32, Google Sheets, and App Sheet

Dr. J. Praveena [1], A. Sai Samhitha [2] and G. Keerthana Sri [3]

[1] Assistant Professor, Department of Mechanical Engineering, Andhra University College of Engineering for Women, Visakhapatnam

[2] Final year student, Department of Mechanical Engineering, Andhra University College of Engineering for Women, Visakhapatnam

[3] Final year student, Department of Mechanical Engineering, Andhra University College of Engineering for Women, Visakhapatnam

ABSTRACT: Accurate and timely stock control is one of the more persistent operational problems in small-scale manufacturing and distribution. Most small warehouses cannot justify the cost of enterprise inventory platforms, yet manual and barcode-based alternatives introduce systematic errors that distort purchasing and replenishment decisions. This work describes an RFID-based inventory management system built around an ESP32 microcontroller, an MFRC522 passive RFID reader, Google Apps Script as a cloud middleware layer, Google Sheets as a zero-cost database, and AppSheet for mobile dashboard access. A closed-loop tag reuse model is central to the design: rather than treating RFID tags as disposable, tags are returned with each delivery and reassigned in software, keeping ongoing hardware costs near zero. The system couples live RFID scan data with classical inventory methods—Economic Order Quantity, Reorder Level, Safety Stock, and ABC Analysis—so that replenishment decisions are driven by actual consumption rather than estimates. A three-month moving average provides a practical demand forecast for dynamic reorder-point computation. Testing confirmed end-to-end latency of 2–3 seconds from tag detection to dashboard update, 100% identification accuracy within the effective read range, and complete elimination of the duplicate-decrement problem through firmware-level debouncing. Total hardware cost is approximately USD 15–20, with no software licensing fees.

Keywords: RFID; ESP32; IoT; Inventory Management; Cloud Computing; Smart Warehouse; Small-Scale Industry

1. Introduction

1.1 Background

Inventory management is one of those problems that looks deceptively simple from the outside. Count what you have, order what you need—straightforward enough. In practice, especially in small-scale manufacturing and distribution, keeping stock records consistently accurate is genuinely difficult. A substantial share of small warehouses in developing economies still track inventory through handwritten registers, periodic physical counts, or spreadsheets updated after the fact [9]. These methods were workable when product ranges were narrow and throughput was low; as operations grow, the gap between what the records say and what is actually on the shelf tends to widen in ways that accumulate real cost.

The core failure of manual systems is temporal: they only update when someone decides to update them. In an active warehouse, goods move continuously, so the count is outdated almost as soon as it is taken. Barcode scanning helps somewhat—it is faster and less susceptible to arithmetic errors—but it still requires a worker to individually scan every item and still depends on readable labels. DeHoratius and Raman [10] measured this problem empirically across a set of retail stores and found record inaccuracies frequent enough and large enough to meaningfully distort replenishment orders.

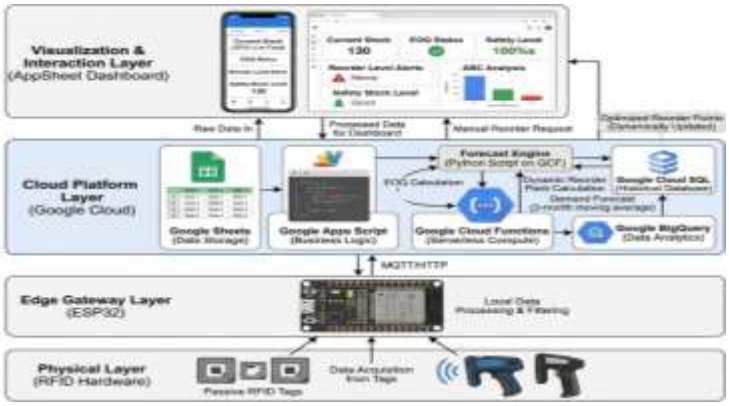
RFID addresses the timing problem directly. A passive reader placed at a warehouse gate can identify every tagged item passing through it without manual intervention and without needing a clear line of sight [3]. The inventory record updates the moment a physical movement happens rather than at the next scheduled count. When this kind of continuous automatic identification is available, classical inventory optimization techniques—EOQ, Reorder Level, Safety Stock—become considerably more useful, because the demand data feeding into them reflects what actually happened rather than what someone estimated.

The difficulty has historically been cost. Commercial RFID middleware, enterprise-grade readers, and dedicated server infrastructure place the technology out of reach for the average small warehouse. What has changed recently is the availability of capable, low-cost IoT hardware—the ESP32 system-on-chip in particular—and free cloud platforms such as Google Sheets that can stand in for a database server without any infrastructure investment. Together, these developments make it realistic to build a functioning RFID inventory system for USD 15–20 in hardware with no recurring software costs.

1.2 Research Gap and Contributions

The published work on RFID inventory systems has concentrated heavily on large-scale or sector-specific deployments: retail, pharmaceutical logistics, high-throughput distribution centers [2], [11]. Designs for resource-constrained, general-purpose small warehouses are underrepresented. Two specific gaps motivated the present work. First, while the cost benefits of reusing RFID tags have been noted in passing [13], no prior implementation provides an explicit operational model for how tags circulate between warehouse and distributor in practice. Second, real-time RFID data and classical inventory optimization methods have rarely been integrated within a single platform accessible to non-specialist operators.

This work contributes: (i) a complete prototype RFID inventory system using low-cost commercial components; (ii) a closed-loop tag reuse model with an explicit circulation workflow; (iii) integration of live RFID consumption data with EOQ, Reorder Level, Safety Stock, and ABC Analysis within a cloud spreadsheet; (iv) a three-month moving average demand forecast that updates reorder points dynamically; and (v) quantified performance data on latency, identification accuracy, and cost feasibility for small-scale deployment.



[Figure 1: Overall System Architecture – four-layer IoT stack from RFID hardware through ESP32, Google Cloud, and AppSheet dashboard]

2. Literature Review

2.1 Traditional Inventory Systems

Harris's 1913 EOQ formulation [14] is the reference point from which most inventory theory departs, establishing the core trade-off between ordering frequency and holding cost. Silver, Pyke, and Peterson [15] and Nahmias [16] built on this foundation with comprehensive treatments of lot-sizing, safety stock, and multi-item control—work that remains practically relevant even as the data-capture technologies around it have changed. The recurring theme across this literature is that the analytical models are only as good as the stock data feeding them. Manual record-keeping rarely meets that standard.

DeHoratius and Raman [10] made this point quantitatively: their field study found a non-trivial fraction of inventory records deviating from physical counts by amounts large enough to trigger wrong replenishment decisions. Inaccuracy is not just a nuisance—it translates into either excess stock or unnecessary stockouts, both of which carry real financial cost. That finding is essentially the empirical justification for automated data capture.

2.2 RFID in Supply Chain and Inventory Management

Academic interest in RFID for inventory accelerated after Walmart's 2003 supplier mandate brought the technology into mainstream supply chain discussion. Rekik [1] modelled the relationship between RFID-enabled inventory visibility and stockout/excess-stock reduction, finding that the benefit scales with how error-prone the system being replaced actually was—a useful sanity check for the present context, where manual tracking errors are common. Sarac, Absi, and Dauzère-Pérès [2] reviewed the RFID supply chain literature broadly and identified enhanced real-time visibility and improved order fulfillment accuracy as the technology's principal contributions.

Wang, Wang, and Wang [4] showed that gate-point RFID scanning—placing readers at inbound and outbound choke points rather than throughout the warehouse—provides an acceptable balance between coverage and cost. That architecture directly informs the deployment model used here. Ngai et al. [11] surveyed RFID research from 1995 to 2005 and found that implementation cost and integration complexity were the two factors most consistently cited as adoption barriers in smaller organizations. Finkenzeller [3] provides the technical foundations, covering antenna design, ISO/IEC 14443 protocol operation, and tag architectures relevant to the MFRC522 used in this work.

Cammarano, Marra, and Michelino [13] examined RFID uptake specifically in SMEs and confirmed that perceived cost and absence of in-house technical capacity were the dominant inhibitors—exactly the constraints the present system is designed to work within, by eliminating proprietary middleware and server infrastructure.

2.3 IoT-Enabled Inventory Monitoring

Extending RFID from isolated scanning events to continuous, networked inventory monitoring is the direction the IoT literature has taken. Ben-Daya, Hassini, and Bahroun [5] surveyed IoT applications across the supply chain and found that real-time connectivity between physical assets and cloud analytics improves responsiveness to demand changes—particularly in fast-moving distribution environments. Rejeb, Rejeb, and Simske [6] reviewed IoT inventory systems more narrowly and found that cloud integration, mobile access, and no-code development tools were the enabling factors for adoption where technical resources are limited.

Khan, Salahuddin, and Habib [7] demonstrated in an industrial deployment that continuous sensor-driven data collection, combined with automated analytical workflows, reduced inventory cycle time and excess stock simultaneously. Sahoo and Pati [17] built a smart inventory prototype combining RFID and IoT principles with similar design priorities—real-time monitoring, remote access, and low infrastructure overhead.

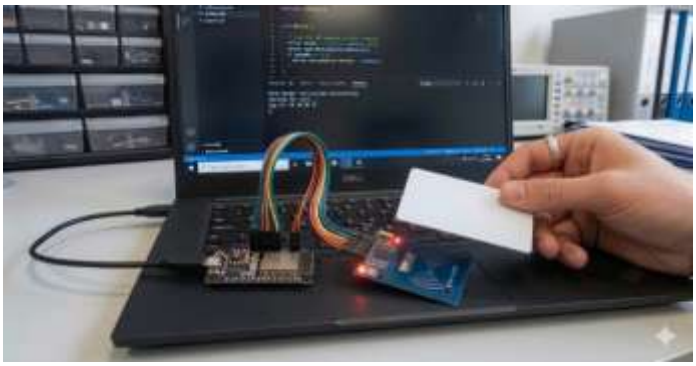
2.4 Industry 4.0 and Smart Warehousing

The industry 4.0 literature provides the broader conceptual frame for where RFID inventory systems fit in a longer trajectory of industrial digitization. Lee, Bagheri, and Kao [19] proposed a reference architecture for cyber-physical production systems in which sensor data flows continuously from the shopfloor to cloud-based analytics—exactly the pattern realized here at small scale. Winkelhaus and Grosse [8] reviewed Logistics 4.0 research and identified RFID-enabled automatic identification and cloud inventory visibility as defining capabilities of next-generation warehouse systems. Lu [20] surveyed Industry 4.0 industrial applications and noted that the convergence of IoT connectivity and no-code platforms is lowering the barrier to smart manufacturing for smaller enterprises. Kumar, Singh, and Kumar [9] extended this frame to include sustainability, finding that tighter inventory control also reduces material waste and improves resource utilization. Sharma and Gupta [18] demonstrated the practical feasibility of low-cost microcontroller-driven RFID systems for real-time warehouse stock management.

3. Proposed System Architecture

3.1 Overview

The system is organized into four layers: a hardware layer responsible for RFID sensing, a processing layer in the ESP32 microcontroller, a cloud layer implemented in Google Apps Script and Google Sheets, and an application layer delivered through AppSheet. As illustrated in Figure 1, data moves unidirectionally from physical tag detection through to dashboard visualization. Feedback—in the form of reorder alerts and restock actions—is initiated by the operator through the application layer. Under stable Wi-Fi conditions, the delay between a tag being scanned and the updated stock count appearing on the dashboard is consistently 2–3 seconds.



[Figure 2: Hardware Setup – ESP32 development board connected to MFRC522 RFID reader module with annotated pin connections (SPI bus: MOSI, MISO, SCK, SS, RST)]

3.2 Hardware Architecture

The ESP32 is a dual-core system-on-chip from Espress if Systems, running the Xtensa LX6 core at up to 240 MHz with integrated 802.11b/g/n Wi-Fi and Bluetooth 4.2. For this application, its main value is the combination of adequate processing speed, built-in wireless connectivity, and a price point well under USD 5. It is programmed using the Arduino IDE with the ESP32 Arduino core, which provides the MFRC522 RFID library, Wi-Fi client library, and a native HTTPS stack for secure communication with Google's servers. As shown in Figure 2, the MFRC522 reader connects to the ESP32 via the SPI bus using four data lines—MOSI, MISO, SCK, and SS—plus a hardware reset pin.

The MFRC522 operates at 13.56 MHz and conforms to ISO/IEC 14443A. Its effective read range for the MIFARE 1K tags used here is 2–5 cm. Each tag carries a factory-programmed 4-byte UID returned as an 8-character hexadecimal string. Importantly, no product-specific data is written to the tag memory. All associations between UIDs and product records live in the cloud database. This separation is what makes the closed-loop reuse model work: a tag carries no intrinsic meaning until mapped in software, so the same physical tag can represent different products over time simply by updating a spreadsheet cell.

3.3 Software and Cloud Architecture

The ESP32 firmware runs a continuous SPI polling loop at 50 ms intervals, checking for new tags. When a tag is detected, the UID is extracted and checked against a 2-second debounce window before anything is sent to the cloud. This debounce step is more consequential than it might seem—without it, holding a tag near the antenna for a few seconds can generate half a dozen scan events and decrement inventory by the same amount.

Validated UIDs go out as HTTPS GET requests to a deployed Google Apps Script web application. The ESP32's built-in TLS stack handles certificate verification, so no external cryptographic library is needed. On the server side, the Apps Script doGet(e) handler looks up the UID in the Inventory Master sheet, decrements the stock quantity, appends a record to the RFID Scan Log sheet, and returns a JSON status response. The ESP32 reads the response and lights an indicator LED to confirm the scan was processed. All of this runs inside the Google cloud without any dedicated hosting. Google Sheets stores three worksheets: the Inventory Master, the RFID Scan Log, and a Restock Log. AppSheet connects natively to Sheets and generates a mobile-responsive dashboard with role-based access control, without any front-end programming.

3.4 Data Flow

Figure 3 illustrates the complete data flow. A product with an attached RFID tag passes within range of the MFRC522 antenna. The reader energizes the passive tag through electromagnetic induction and reads its UID using the ISO 14443A anticollision protocol. The ESP32 firmware validates the UID, builds the HTTPS GET request, and sends it over Wi-Fi. The Apps Script processes the request—looking up the product, decrementing the stock count, and writing the log entry—then returns a confirmation response. Formula cells in the Inventory Master sheet recalculate immediately, updating stock status labels and reorder quantity figures. AppSheet polls the data source and refreshes the dashboard within the same 2–3 second window.

[Figure 3: RFID Tag Detection and Data Transmission Workflow – from tag presentation at reader antenna through ESP32 firmware, HTTPS request, Apps Script processing, and Google Sheets update]

3.5 Closed-Loop Tag Reuse Model

Each time a shipment goes out to the distributor, the RFID tags attached to the outgoing packages are removed and sent back to the warehouse on the next return vehicle. After a brief inspection, the tags re-enter the pool and are attached to the next batch of the same product type before its next dispatch. Because the UID-to-product mapping is maintained entirely in the Inventory Master sheet, reassigning a tag takes one cell edit. The tag itself requires no programming or modification.

For a small-scale warehouse, the tag pool needed to cover maximum concurrent in-transit inventory is typically a few dozen units—a one-time expenditure rather than a recurring consumable cost. This is one of the more practically significant aspects of the design for budget-constrained operators.

4. Inventory Management Model

4.1 Economic Order Quantity

Harris's EOQ model [14], further developed by Silver et al. [15], determines the replenishment quantity that minimizes the combined cost of ordering and holding inventory:

$$EOQ = \sqrt{(2DS / H)}$$

where D is annual demand in units, S is the fixed cost per order, and H is the annual holding cost per unit. EOQ calculations are only as reliable as the demand figure D. Manual systems estimate D from periodic counts subject to substantial measurement error. When D comes from the RFID scan log—where every dispatched unit is recorded with a timestamp—monthly consumption aggregates are considerably more accurate, and the resulting EOQ figures are correspondingly more useful.

4.2 Reorder Level

The Reorder Level marks the stock quantity at which a replenishment order needs to be placed for it to arrive before stock runs out. In this system it updates dynamically:

$$\mathbf{ROL = Forecast\ Demand + Safety\ Stock}$$

Because the forecast is recalculated each month from the RFID log, the ROL adjusts automatically as demand patterns shift. A product moving faster than it did three months ago will have a higher ROL and will trigger restocking earlier.

4.3 Safety Stock

Safety stock absorbs variability on both the demand side and the supply side—unexpected spikes in consumption or late deliveries from a supplier. The standard service-level formula [16] is:

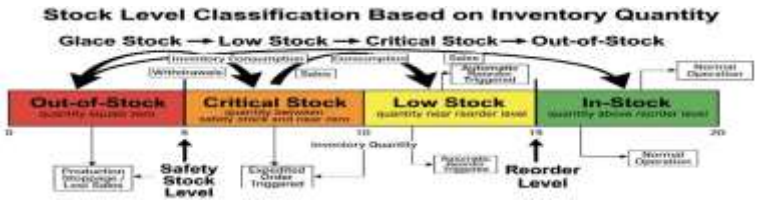
$$\mathbf{Safety\ Stock = Z \times \sigma\_LTD}$$

where Z is the standard normal deviate for the target service level (1.645 for 95%) and σ_LTD is the standard deviation of demand during the lead time. In the current implementation, safety stock values are set manually by the warehouse supervisor based on experience with specific suppliers. The RFID scan log accumulates the historical data that would eventually support automated computation of σ_LTD; for now, the manual approach reflects the limited historical dataset available.

4.4 ABC Analysis

ABC Analysis applies Pareto's observation—that a small fraction of items accounts for the majority of value—to inventory classification. Category A items (roughly 10–20% of SKUs but 70–80% of total inventory value) justify tight controls and accurate demand forecasting. Category B items (about 30% of SKUs, 15–20% of value) warrant moderate oversight. Category C items (50–60% of SKUs, 5–10% of value) can be managed with simpler replenishment rules [15].

Mapping this classification onto the RFID system enables differential monitoring: automated high-frequency tracking and low reorder thresholds for A-class items, standard monitoring for B-class, and periodic review sufficiency for C-class. The practical effect is that the system allocates attention where it has the largest financial impact.



[Figure 4: Inventory Status Classification Diagram – flow chart showing how current stock quantity maps to In-Stock / Low Stock / Critical / Out-of-Stock labels against Safety Stock, Reorder Level, and EOQ thresholds]

4.5 Demand Forecasting

A three-month simple moving average drives the demand forecast:

$$\mathbf{Forecast\ Demand = (D_1 + D_2 + D_3) / 3}$$

where D_1, D_2, and D_3 are actual dispatch quantities from the three most recent calendar months, aggregated from the RFID scan log. This approach was chosen deliberately over exponential smoothing or ARIMA models. Small warehouses often have limited, inconsistent historical data, and a moving average is easy to explain to a supervisor who needs to understand and trust the numbers [16]. The goal is not precise prediction but a reasonable working estimate that informs restock timing and order quantity.

The forecast feeds directly into the Reorder Quantity:

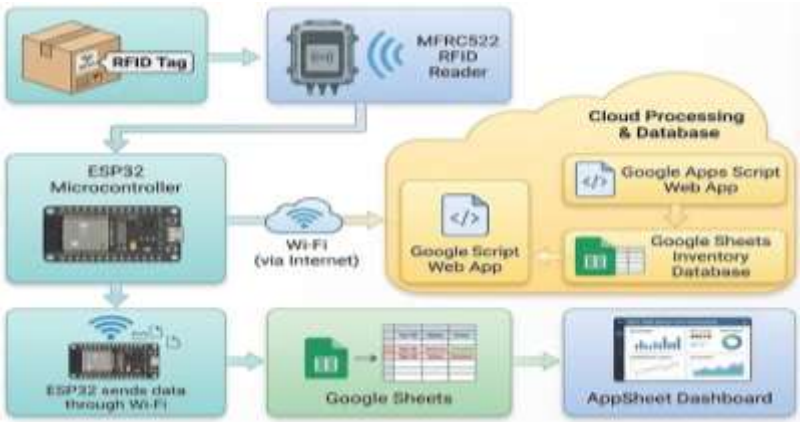
$$\mathbf{ROQ = Forecast\ Demand - Current\ Stock}$$

When a product drops below its Reorder Level, the dashboard displays the ROQ alongside the stock status label—giving the supervisor a concrete order quantity recommendation rather than just a flag that something needs attention.

5. System Implementation

5.1 RFID Scanning Workflow and ESP32 Firmware

On startup, the firmware initializes the SPI peripheral and the MFRC522 library, then enters a 50 ms polling loop. When the MFRC522 detects a tag, it runs the ISO 14443A anticollision sequence and returns the UID. The firmware converts this to an 8-character hex string and checks whether the same UID was processed within the last 2,000 ms. If it was, the detection is discarded. This debounces check proved essential during testing. Validated UIDs are placed into an HTTPS GET request with the UID as a query parameter.



[Figure 5: Cloud Data Flow – sequence diagram from ESP32 HTTPS GET request through Google Apps Script doGet() handler, Inventory Master sheet update, log append, and JSON response return]

5.2 Cloud Processing via Google Apps Script

The Apps Script doGet(e) function handles each valid scan in four steps. First, it extracts the UID query parameter. Second, it searches the RFID UID column of the Inventory Master sheet for a matching row. Third, if a match is found, it calls set Value() to decrement the stock quantity atomically. Fourth, it appends a new row to the RFID Scan Log containing the timestamp, scanned UID, matched Product ID, and post-decrement quantity. If no matching UID is found, the function returns an error response without modifying any data—a guard against unregistered or spare tags accidentally affecting inventory records.

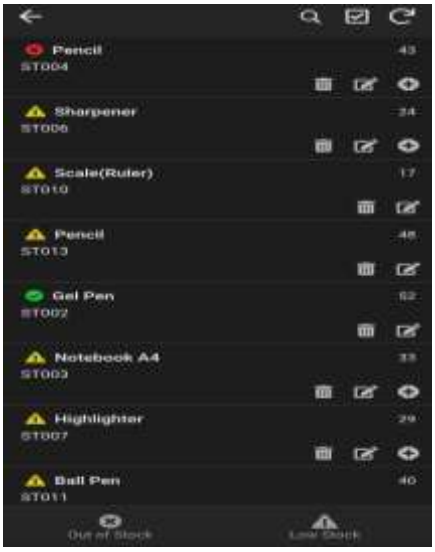
5.3 Google Sheets Database Structure

The Inventory Master sheet stores one row per product: Product ID, product name, current quantity, safety stock level, reorder point, forecast demand, three months of demand history, RFID UID assignment, stock status label, and Reorder Quantity. Formula cells compute the status label and ROQ automatically whenever the quantity column changes, so the monitoring state updates synchronously with each RFID scan. The Inventory Master also holds a demand history section drawing from a separate demand history worksheet updated monthly by the supervisor.

The RFID Scan Log is append-only by design: the Apps Script never modifies existing rows. This gives it the properties of a true audit trail—any question about when a stock level reached a particular value can be answered by scrolling back through the log. The Restock Log records every restocking action initiated through the AppSheet automation workflow, creating a parallel record of stock coming in to complement the scan log's record of stock going out.

5.4 AppSheet Dashboard

AppSheet connects to the Google Sheets data source and renders the inventory data through two primary views, illustrated in Figure 5. The main table view shows all products with current quantity, color-coded stock status (green for In-Stock, yellow for Low Stock, orange for Critical, red for Out-of-Stock), and ROQ. The colour coding is the most practically useful feature for operators: a single glance at the table reveals which products need attention without reading any numbers. Tapping a row opens the product detail view, which displays the full record alongside a Restock button. Pressing the button triggers an AppSheet automation that adds the current ROQ to the quantity field and simultaneously writes a record to the Restock Log—both operations in a single atomic workflow step. A secondary read-only scan log view shows recent scan events in reverse chronological order for post-session verification.



[Figure 6: AppSheet Inventory Dashboard – main table view showing product list with color-coded stock status labels and Reorder Quantity column; inset showing product detail view with Restock button]

6. Results and Discussion

6.1 System Performance

End-to-end latency testing measured the time from tag presentation at the MFRC522 antenna to the updated stock count appearing on the AppSheet dashboard. Under stable Wi-Fi conditions, this consistently fell in the 2–3 second range. RFID identification accuracy was 100% across all test scans for standard MIFARE 1K tags within the 2–5 cm effective read range. No false positive detections occurred; the Apps Script UID validation successfully rejected every scan of an unregistered tag.

The debounce mechanism worked as intended. Without filtering, holding a tag against the antenna for three seconds produced four to seven scan events and the same number of inventory decrements. With the 2-second debounce window active, the same action produced exactly one decrement in every test case. This result is not surprising in principle but confirms that debouncing is not merely a nice-to-have—it is a correctness requirement.

6.2 Improvements over Manual Systems

The operational difference most clearly demonstrated by the prototype is the immediacy and continuity of stock record updates. Manual systems record stock movements when someone decides to write them down, which in practice means the records are stale for hours at a time [10]. The RFID scan log updates within 3 seconds of a physical movement, keeping the Inventory Master sheet representative of current warehouse state throughout the working day.

That accuracy carries forward into the analytical models. When EOQ, ROL, and safety stock computations draw on consumption figures from the RFID log rather than periodic estimates, they reflect actual demand rather than someone's recollection of demand. The scan log also provides an audit trail that manual registers do not reliably maintain—particularly in busy periods when the discipline required to update registers consistently tends to break down. Rekik [1] and DeHoratius and Raman [10] both documented the financial consequences of record inaccuracy in prior work; this system directly addresses the root cause.

6.3 Accuracy and Latency

The 100% identification accuracy result should be interpreted in context: it holds within the 2–5 cm effective range of the MFRC522 module. Tags outside this range are not detected, and tags presented at an oblique angle can require repositioning. For a controlled scanning station—a counter or gate where operators pass items within a defined range—this is not a practical limitation. For area-wide monitoring of a large warehouse floor, it would be.

The 2–3 second latency is functionally real-time for inventory management purposes. No operational decision requires knowing that a unit left the warehouse within 3 seconds rather than 30; the meaningful improvement over manual systems is measured in hours, not fractions of a second.

6.4 Cost Effectiveness

Total hardware cost for the prototype was approximately USD 15–20: an ESP32 development board, an MFRC522 module, and a set of MIFARE 1K tags. Software costs are zero—the Arduino IDE, Google Apps Script, Google Sheets, and AppSheet at the free tier all require no licensing fees. The closed-loop tag reuse model converts what would otherwise be a recurring per-shipment label cost into a one-time pool investment. For a warehouse shipping

dozens of orders daily, even the per-unit cost of disposable RFID tags adds up quickly; the reuse model eliminates that accumulation entirely.

The no-code AppSheet dashboard removes the need for software developers or database administrators. A warehouse supervisor comfortable with spreadsheets can configure and maintain the system. This directly addresses the adoption barrier that Cammarano et al. [13] identified as dominant in SMEs: lack of in-house technical expertise.

7. Advantages and Limitations

7.1 Advantages

Several aspects of the design are worth summarizing as distinct strengths. Real-time stock visibility at sub-3-second latency means that replenishment computations are based on current data, not estimates. The hardware and software costs are genuinely low—this is not a low-cost system compared to enterprise alternatives; it is a low-cost system in absolute terms. The closed-loop tag reuse model eliminates recurring consumable costs without any tag reprogramming overhead. Coupling live RFID data with EOQ, ROL, and forecasting within the same spreadsheet environment connects data collection to decision support in a way that most small-business deployments do not achieve. AppSheet's mobile interface makes the dashboard accessible to operators on the floor without specialist training.

7.2 Limitations

Several limitations are real and worth stating plainly. The system's biggest vulnerability is its dependence on Wi-Fi. Network outages cause scan requests to fail silently—there is no local buffering, so movements during downtime are simply not logged. Implementing event buffering in ESP32 SPIFFS flash memory would fix this but adds complexity. The MFRC522's 2–5 cm read range is appropriate for a controlled scanning station but rules out area-wide monitoring. The system tracks aggregate quantities by product type, not individual unit serial numbers, making it unsuitable for applications requiring batch-level traceability. The three-month moving average cannot capture seasonality or long-run trends. Demand history is updated manually each month, introducing both effort and error risk. Finally, Google Sheets lacks transaction locking, so simultaneous scan requests from multiple ESP32 units would generate race conditions in the Apps Script handler—a constraint that limits scalability to single-reader deployments without architectural changes.

8. Conclusion and Future Work

The work described here demonstrates that RFID-based inventory management is achievable at a hardware cost below USD 20 and zero ongoing software cost, using commercially available components and free cloud platforms. The system addresses the two main problems with manual and barcode-based tracking in small warehouses: the frequency and consistency of stock record updates, and the absence of any analytical connection between raw count data and replenishment decisions. Testing confirmed the core performance targets—2–3 second end-to-end latency, 100% identification accuracy within range, and reliable debouncing—and the closed-loop tag reuse model keeps the economic case for adoption intact over the long run.

Several directions for future development are clear from the current limitations. Offline event buffering using ESP32 SPIFFS would make the system resilient to Wi-Fi outages—the most likely failure mode in a small industrial setting. A UHF reader operating at 860–960 MHz would extend effective read range to several meters and enable gate-level

bulk scanning without the operator needing to present items individually. The demand forecasting module would benefit from exponential smoothing with a trend component, particularly for products with non-stationary demand. Automated monthly aggregation of the RFID scan log would remove the current manual step in updating demand history. Longer-term, full ABC Analysis implementation with automated differential monitoring intensity, and supplier API integration for automated purchase order generation, represent the logical evolution toward a more complete inventory management platform.

References

[1] M. Rekik, "RFID in inventory management: A review," International Journal of Production Economics, vol. 112, no. 2, pp. 657–667, 2008.

[2] A. Sarac, N. Absi, and S. Dauzère-Pérès, "A literature review on the impact of RFID technologies on supply chain management," International Journal of Production Economics, vol. 128, no. 1, pp. 77–95, 2010.

[3] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication, 3rd ed. Chichester, UK: Wiley, 2010.

[4] L. Wang, G. Wang, and Y. Wang, "RFID-based inventory management system," IEEE Transactions on Automation Science and Engineering, vol. 6, no. 3, pp. 544–551, 2009.

[5] M. Ben-Daya, E. Hassini, and Z. Bahroun, "Internet of Things and supply chain management: A literature review," International Journal of Production Research, vol. 57, no. 15–16, pp. 4719–4742, 2019.

[6] A. Rejeb, K. Rejeb, and S. Simske, "IoT-based inventory management: A systematic review," IEEE Access, vol. 8, pp. 181345–181362, 2020.

[7] N. Khan, M. Salahuddin, and R. Habib, "Industrial IoT-based warehouse management," Journal of Industrial Information Integration, vol. 15, pp. 100–112, 2019.

[8] B. Winkelhaus and E. H. Grosse, "Logistics 4.0: A systematic review towards a new logistics system," International Journal of Production Research, vol. 58, no. 1, pp. 18–43, 2020.

[9] A. Kumar, R. Singh, and P. Kumar, "Industry 4.0 and sustainable supply chain management: A systematic literature review," Journal of Cleaner Production, vol. 245, article 118130, 2020.

[10] N. DeHoratius and A. Raman, "Inventory record inaccuracy: An empirical analysis," Management Science, vol. 54, no. 4, pp. 627–641, 2008.

[11] E. W. T. Ngai, K. K. L. Moon, F. J. Riggins, and C. Y. Yi, "RFID research: An academic literature review (1995–2005) and future research directions," International Journal of Production Economics, vol. 112, no. 1, pp. 510–520, 2008.

[12] J. H. Kembro, D. Näslund, and K. Olhager, "Information sharing across multiple supply chain tiers: A Delphi study on antecedents," International Journal of Production Economics, vol. 193, pp. 77–86, 2017.

[13] A. Cammarano, E. Marra, and M. Michelino, "Open innovation and RFID adoption in SMEs," Technovation, vol. 58–59, pp. 1–10, 2017.

[14] F. W. Harris, "How many parts to make at once," Factory: The Magazine of Management, vol. 10, no. 2, pp. 135–136, 1913.

[15] E. A. Silver, D. F. Pyke, and R. Peterson, Inventory Management and Production Planning and Scheduling, 3rd ed. New York, NY: Wiley, 1998.

[16] S. Nahmias, Production and Operations Analysis, 6th ed. New York, NY: McGraw-Hill/Irwin, 2009.

[17] S. K. Sahoo and B. B. Pati, "Smart inventory management using IoT," IEEE Access, vol. 9, pp. 102376–102385, 2021.

[18] A. Sharma and V. Gupta, "RFID-based warehouse automation system for real-time stock tracking," International Journal of Engineering Research and Technology, vol. 10, no. 3, pp. 45–52, 2021.

[19] J. Lee, B. Bagheri, and H. A. Kao, "A cyber-physical systems architecture for Industry 4.0-based manufacturing systems," Manufacturing Letters, vol. 3, pp. 18–23, 2015.

[20] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," Journal of Industrial Information Integration, vol. 6, pp. 1–10, 2017.