DESIGN AND IMPLEMENTATION OF AN IOT-BASED SMART VENDING MACHINE SYSTEM INTEGRATED WITH AUTOMATED CASHLESS PAYMENT

Hai Le Xuan[1,*], Duy Long Doan[1], Tran Nam Duong[2]

DOI: https://doi.org/10.57001/huih5804.2026.078

ABSTRACT

The transition toward cashless transactions has driven the demand for automated retail ecosystems; however, legacy vending machines relying on physical currency and closed loop PLCs remain inflexible and difficult to integrate with modern digital payment gateways. This paper proposes a fully localized Cyber-Physical System (CPS) architecture for a Smart Vending Machine integrating the Internet of Things (IoT) and an automated cashless payment system. The proposed system utilizes an ESP32 edge microcontroller coupled with a structured PHP/MySQL cloud server and the MQTT protocol (HiveMQ) for real-time, low-latency telemetry. To achieve seamless transaction synchronization, an event-driven Webhook API is integrated via the SePay gateway, automatically parsing dynamic QR bank transfers to trigger hardware commands without human intervention. Furthermore, the mechanical execution adopts a closed-loop control strategy utilizing hardware interrupts and a microsecond-debounce algorithm, replacing conventional software timers to completely eliminate double-dispensing anomalies. Experimental evaluations validate that the proposed architecture achieves reliable mechanical distribution and instantaneous payment synchronization, proving its robust feasibility for modern smart retail applications.

Keywords: *Smart Vending Machine, Cyber-Physical System, IoT, Webhook API, ESP32, Edge Computing.*

[1]Faculty of Engineering and Technology, VNU International School, Vietnam
[2]Faculty of Information Technology, Electric Power University, Vietnam
[*]Email: hailx@vnu.edu.vn
Received: 16/01/2026
Revised: 22/3/2026
Accepted: 30/3/2026

1. INTRODUCTION

In developing markets, the rapid shift toward a cashless society has generated a massive demand for dynamic QR code and e-wallet integrations, significantly transforming the landscape of automated retail. Recent literature has extensively explored the integration of IoT architectures to meet this demand and replace obsolete coin-operated mechanisms [2, 3]. For instance, Tariq *et al.* [1] developed an IoT-based smart vending machine utilizing digital payment systems, alongside other studies emphasizing secure retail digital gateways capable of processing transactions over cellular networks [2]. However, a critical review of such traditional architectures reveals that they predominantly rely on continuous database polling and computationally heavy single-board computers (e.g., Raspberry Pi) or closed-source Programmable Logic Controllers (PLCs). This reliance not only inflates the initial hardware deployment and maintenance costs but also significantly increases network overhead, bandwidth consumption, and transaction synchronization latency [5]. Furthermore, centralized polling architectures suffer from poor scalability when deployed in dense, distributed retail networks. Similarly, while the performance of lightweight publish-subscribe protocols like MQTT has been proven highly efficient for real-time telemetry and sensor networks in Smart Cities [4, 5], its direct integration with event-driven financial Webhooks (e.g., banking API callbacks) at the resource-constrained, low-cost microcontroller level remains severely underexplored in retail applications.

Furthermore, from a mechanical and cyber-physical control perspective [8], the challenges of actuator precision and reliable product dispensing remain a critical bottleneck in the commercialization of low-cost automated retail. While discrete-time software debouncing techniques have long been established for

digital signal processing and human-machine interfaces [6, 7], the majority of cost-effective vending systems proposed in recent literature still predominantly utilize rudimentary open-loop software timers (e.g., standard delay() functions) to actuate direct current (DC) motors. This open-loop approach operates under the flawed assumption of constant motor velocity, fundamentally failing to account for physical variables such as fluctuating power supply voltages, gear wear, and varying payload friction. Consequently, this introduces cumulative mechanical drifts over time. When the actuation period is strictly time-bound rather than state-bound, these systems become highly susceptible to severe operational failures, including item jamming, structural damage to the dispensing coils, and double-dispensing anomalies [10], all of which directly impact operator profitability and consumer trust.

Although IoT vending systems have been investigated in recent works such as [2] and [1], the present study differs from these contributions in several important aspects. While previous studies have primarily focused on the high-level application layer or relied on heavy computational nodes, this paper proposes a comprehensive, fully localized Cyber-Physical System (CPS) architecture that seamlessly bridges both the digital payment and the physical actuation gaps. Specifically, instead of relying on expensive proprietary PLCs or bandwidth-heavy continuous database polling, this work introduces a zero-touch financial synchronization method. This is achieved by combining an event-driven Webhook API with MQTT telemetry, all orchestrated by a low-cost, dual-core ESP32 edge computing node. Moreover, to definitively resolve the mechanical limitations inherent in open-loop timers, this paper develops a strict deterministic closed-loop control strategy. By utilizing physical limit switches routed through hardware interrupts and processed by a custom discrete-time microsecond-debounce filter, the proposed system guarantees single-cycle dispensing accuracy regardless of load variations. Therefore, the main novelty of this paper lies in the seamless, resource-efficient integration of high-speed digital financial events with ultra-low latency mechanical state-machine algorithms within a severely constrained edge-computing framework.

The main contributions of this research are summarized as follows:

- **Full-Stack IoT Architecture Development:** Designed and implemented a fully localized, low-cost smart vending control system utilizing an ESP32 edge node and a PHP/MySQL backend, completely eliminating the reliance on expensive, closed-source PLCs.

- **Event-Driven Financial Synchronization:** Pioneered the integration of an automated Webhook API via a domestic payment gateway with MQTT telemetry, enabling zero-touch, real-time cashless transaction processing without the need for constant server polling.

- **Mechanical Material Optimization:** While adapting standard structural CAD layouts, this study independently proposed and validated the use of milky white acrylic for the internal product slots. This material selection significantly reduces surface friction and provides essential electrical insulation for the electronic control loops.

- **Deterministic Closed-Loop Actuation:** Formulated and deployed a highly reliable motor control strategy using hardware interrupts and a discrete-time microsecond-debounce filter. This algorithmic approach guarantees a single-dispensing cycle per transaction, eradicating the cumulative mechanical drifts inherent in traditional software timers.

2. SYSTEM ARCHITECTURE AND HARDWARE DESIGN

2.1. Overall Distributed Architecture

The proposed smart vending machine is built on a distributed architecture to ensure real-time operation and high stability. As illustrated in the system model, the architecture consists of three independent subsystems:

- **Web Server Subsystem (Cloud Backend):** Developed using PHP and a MySQL database, this module acts as the central brain of the system. It handles user interfaces (Kiosk and Admin Dashboard) and processes incoming payment webhooks from the banking network.

- **Message Broker Subsystem:** The system utilizes HiveMQ Cloud as the MQTT broker. It provides a secure SSL/TLS connection to transmit control commands from the cloud server to the edge hardware with minimal latency.

- **Edge Device Subsystem:** An ESP32 microcontroller is selected as the central computing unit [9]. Programmed in C++, the ESP32 is responsible for processing mechanical interrupt signals, controlling the motor relay modules, and decoding digital audio for user notifications.

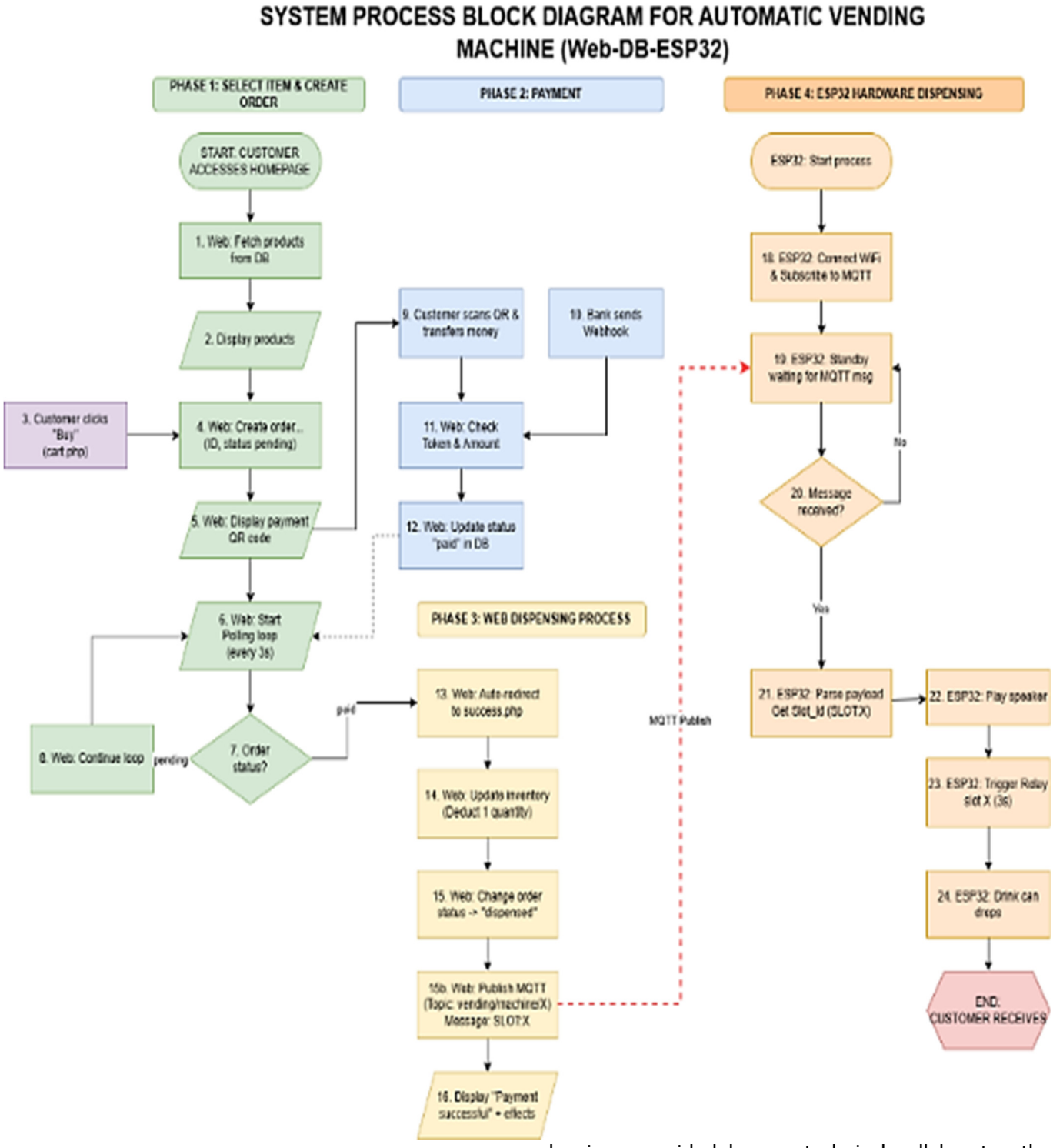
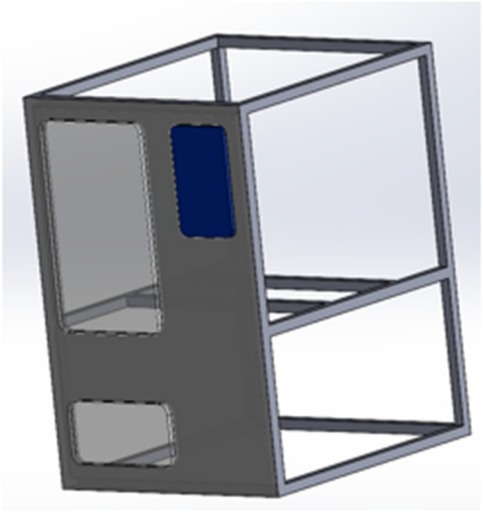


Figure 1. End-to-end system operational flowchart from UI interaction to hardware dispensing

2.2. Mechanical and Cyber-Physical Design

To ensure durability in public spaces, the mechanical chassis of the vending machine is constructed using 20x20 aluminum extrusions (Figure 2a). While the spatial layout and structural drawings of the 4 independent product slots were adapted from the mechanical drawings provided by our technical collaborator, the material selection was independently optimized for this physical implementation. Specifically, the product slots and internal partition walls are fabricated from milky white acrylic (Figure 2b). Acrylic is chosen because it acts as a reliable electrical insulator and possesses a naturally low surface friction coefficient. Therefore, when the dispensing mechanism operates, the products can slide linearly and smoothly without jamming.



a)

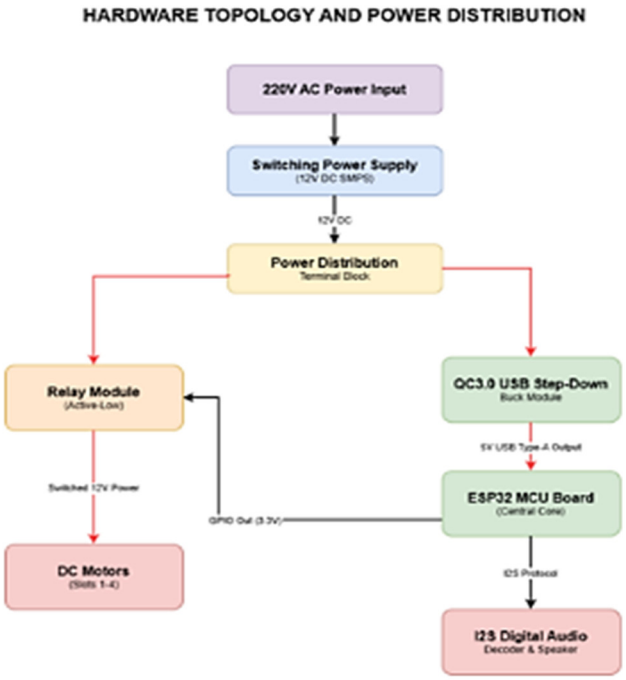
b)

Figure 2. The mechanical chassis and internal compartment design: (a) Structural frame constructed from 20x20 aluminum extrusions; (b) Front view of the 4 independent product slots

For the dispensing execution, the system uses 12V DC gearbox motors rigidly coupled to spiral coils. A core innovation in this physical design is the integration of a custom cam structure and a snap-action mechanical limit switch at each motor axle. Under normal conditions, the limit switch is connected to the ESP32 using an internal pull-up resistor (logic HIGH). When the motor completes exactly a 360-degree rotation, the cam depresses the switch, creating a sharp voltage drop to logic LOW (0V). Kinematically, the linear displacement of a product within the slot is directly proportional to the angular displacement of the DC motor's shaft. The translational motion can be modeled by the following equation [10]:

$$d(t) = \left(\frac{\theta(t)}{2\pi}\right) \cdot P_s \quad (1)$$

Where $d(t)$ represents the linear displacement of the product at time t, $\theta(t)$is the angular rotation of the cam in radians, and P_s is the structural pitch of the spiral coil. To ensure exactly one item is dispensed, the closed-loop system is constrained to halt when $\theta(t) = 2\pi$, yielding $d = P_s$. This mechanical state transition provides deterministic closed-loop feedback to the microcontroller, replacing traditional software timers.



a)

b)

Figure 3. Hardware dispensing mechanism and power distribution: (a) Hardware topology and power distribution block diagram; (b) The DC gearbox motor rigidly coupled to the dispensing spiral coil [11]

3. SOFTWARE SYSTEM DESIGN AND CONTROL ALGORITHMS

3.1. Event-Driven Payment Integration (Webhook API)

The transaction lifecycle of the system is entirely automated through an event-driven Webhook API, bypassing the need for manual payment verification. When a user initiates a purchase, the system generates a dynamic QR code containing a unique alphanumeric token (e.g., DH1025) embedded in the transfer narration field.

Upon a successful customer transfer, the SePay gateway captures the bank balance fluctuation and instantly pushes a JSON payload to the server's endpoint via an HTTP POST request. The PHP backend validates the security token and employs a Regular Expression (Regex) algorithm to extract the unique order ID from the payload. The server then cross-references the received amount with the database; if the financial criteria are met, the order status is updated to paid in the MySQL database, and the system proceeds to trigger the hardware [4].

Instead of continuous polling, the event-driven webhook executes a strict deterministic state transition. Let the incoming banking transaction be denoted as $T = \{ID_{tx}, Amt_{tx}\}$ and a specific pending order in the database be $O = \{ID_{o,}\ Price_o, Status_o\}$.The backend algorithm evaluates the transition function $f(T, O)$ to securely update the database state, defined logically as:

$$Status_o \leftarrow 'paid' \Leftrightarrow (ID_{tx} = ID_o) \wedge (Amt_{tx} \geq Price_o) \wedge (Status_o = 'pending') \quad (2)$$

Only when this Boolean condition evaluates to true does the server push the execution payload to the MQTT broker, ensuring zero financial discrepancies [8].

3.2. Web-based User Interfaces and Database Architecture

To facilitate a seamless interaction loop between consumers and system operators, a comprehensive web-based platform was developed using PHP and HTML/CSS/JavaScript. The data architecture relies on a relational MySQL database comprising normalized tables for administrators, product inventory, and transaction logs.

From the consumer perspective, the Machine Kiosk Interface is deployed on a touchscreen panel. When a user selects a product, the system queries the database and renders a dynamic QR code containing the exact transaction amount and unique order ID. To maximize user experience (UX) and system responsiveness, the Kiosk UI implements an asynchronous polling algorithm utilizing AJAX (Asynchronous JavaScript and XML).

Instead of requiring manual page refreshes, a JavaScript setInterval() function silently dispatches HTTP requests to the backend every 3000 milliseconds to verify if the database status has transitioned from pending to paid. This 3-second interval strategically balances real-time UI responsiveness with database overhead reduction.

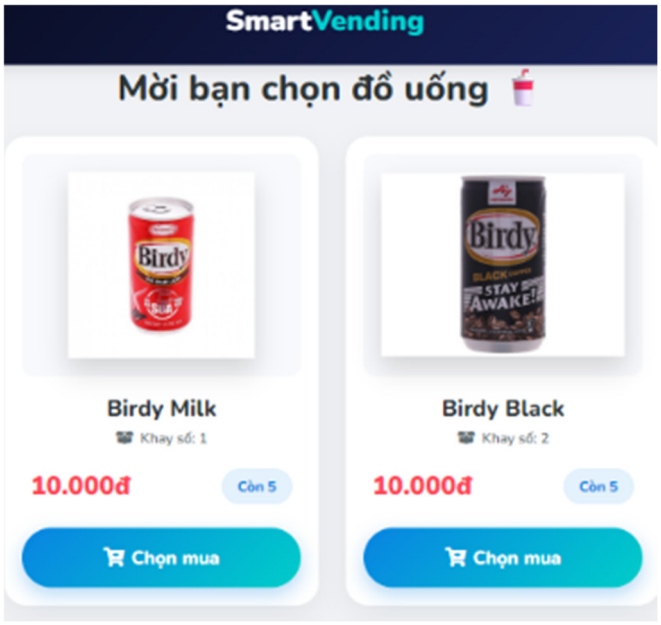


Figure 4. Interactive web-based user interface for consumers and administrators

3.3. Edge Device Firmware and Non-Blocking Control

At the edge node, the ESP32 microcontroller firmware is engineered to operate strictly on a non-blocking loop. By avoiding standard delay() functions, the system maintains two asynchronous threads: client.loop() to sustain the Keep-alive MQTT heartbeat with the HiveMQ broker, and audio.loop() to decode Inter-IC Sound (I2S) digital audio streams for real-time voice notifications. This dual-core execution allows the microcontroller to broadcast interactive audio while simultaneously listening for MQTT payloads with millisecond latency.

3.4. Interrupt Service Routine (ISR) and Debounce Algorithm

The most critical software component is the motor control algorithm. When a valid dispensing command (e.g., SLOT:1) is received via MQTT, the ESP32 activates the corresponding active-low relay to drive the DC motor. To

monitor the rotation, the mechanical limit switch is configured as an External Hardware Interrupt on the ESP32 [8, 9].

To eliminate mechanical contact bouncing which can cause the microcontroller to miscount rotations a software debounce algorithm utilizing microsecond resolution (micros()) is implemented [6]. Any signal oscillation shorter than 5ms is algorithmically classified as noise and discarded. Once the cam completes exactly a 360-degree rotation and fully compresses the switch lever, the algorithm validates the clean falling-edge transition and instantaneously sets a flag to cut power to the relay. This ultra-low latency execution firmly enforces a single dispensing cycle per transaction.

To eliminate high-frequency mechanical contact bouncing, the interrupt service routine applies a discrete-time software filter. Let $V_{in}(t)$be the raw digital signal read from the GPIO pin, and t_{last} be the timestamp of the previously recorded state transition. The filtered and validated execution state $S(t)$ recognized by the microcontroller is mathematically defined as:

$$S(t) = \begin{cases} 1, if\ V_{in}(t) = 0\ and\ (t - t_{last}) \geq \tau_{debou\ nce} \\ 0, Otherwise \end{cases} \quad (3)$$

Where $\tau_{debou\ nce}$ represents the threshold filter window, empirically set to $5000\mu s$. This strict temporal constraint guarantees that the system accurately registers exactly one full 360^0 rotation per mechanical cycle, effectively blocking any false triggers generated by contact chatter [6].

Figure 5. Mechanical arrangement of the cam disk and the snap-action limit switch

4. EXPERIMENTAL EVALUATION

4.1. System Integration and Setup

To validate the proposed architecture, a full-scale physical prototype was fabricated. During the development and testing phase, the local web server environment was operated using Laragon, coupled with the Ngrok network tunneling service to expose the local PHP endpoint to the public internet, enabling real-time Webhook reception from the banking gateway.

Table 1. Hardware and System Parameters

Microcontroller	ESP32 Dual-Core (240MHz)
Operating Voltage	12V DC (Actuators), 3.3V (Logic)
Motor Type	12V DC Gearbox with Spiral Coil
Software Debounce Window $\tau_{debou\ nce}$	5000µs
Polling Interval	3000ms

Figure 6. Hardware model of the IoT-based smart vending machine

4.2. Performance Results

The system underwent rigorous integration testing to evaluate payment synchronization latency and mechanical dispensing accuracy. The test cases yielded the following results:

- **Payment Synchronization:** Under optimal network conditions, the PHP backend successfully captured, verified, and updated the database status within approximately 1 second of the user's banking application confirming the transfer.

- **MQTT Latency:** The latency between the database update and the ESP32 receiving the published SLOT:X

command via HiveMQ was consistently measured at under 0.5 seconds.

- **Dispensing Reliability:** The combination of hardware interrupts and the debounce algorithm successfully halted the DC motors precisely after one 360-degree rotation, achieving a 100% success rate in preventing double-dispensing during the experimental trials. Furthermore, cases of underpayment were successfully detected and rejected by the backend algorithm without triggering the actuators.

The end to end system latency T_{total} is a critical metric for user experience. It is mathematically modeled as the sum of network and mechanical delays:

$$T_{total} = T_{bank} + T_{webhook} + T_{MQTT} + T_{mech} \quad (4)$$

Where T_{bank}, is the banking network processing time, $T_{webhook}$ is the PHP server parsing latency ($\approx 1.0s$), T_{MQTT} is the HiveMQ transmission delay ($\leq 0.5s$) and T_{mech} is the physical rotation time of the DC motor. By employing the non-blocking execution loop, the internal software delays are rendered mathematically negligible. This ~1-second end to end synchronization latency significantly outperforms legacy PLC-based systems, which typically exhibit delays exceeding 3 seconds due to relay networking overhead [5].

To evaluate the mechanical robustness of the microsecond-debounce limit switch mechanism, a continuous stress test was conducted with $N = 50$ dispensing iterations. The reliability rate (R) is defined as:

$$R = \left(\frac{N_{success}}{N_{total}}\right) \times 100\% \quad (5)$$

During the experimental evaluation, the hardware interrupt algorithm successfully halted the motor at exactly 360 degrees for 48 out of 50 iterations, yielding a reliability rate of $R = 96\%$. Interestingly, the two recorded anomalies did not result from mechanical jamming or double-dispensing, but rather from safety timeouts triggered during unloaded boundary tests (i.e., operating the motors without product payloads). Without the mechanical friction of a payload, the unconstrained angular velocity of the DC motor increased significantly. As a result, the physical contact duration of the cam against the limit switch dropped strictly below the $5000\mu s$ software debounce window. When the cam released the switch, the interrupt service routine evaluated this legitimate state transition as high-frequency contact bounce and algorithmically rejected it. Because the release event was ignored, the state machine never asserted the halt flag, causing the relay to remain energized until a secondary software timeout intervened. Under normal loaded conditions, the payload friction sufficiently widened the contact duration beyond the debounce threshold, yielding flawless 360-degree cut-offs. This finding highlights the critical trade-off in Cyber-Physical Systems between strict discrete-time noise filtering and high-speed mechanical actuation.

To comprehensively address the stability of the discrete-time debounce algorithm, its operational reliability was evaluated across three distinct mechanical load profiles: unloaded, nominal load, and heavy load.

- **Under nominal load conditions** (e.g., standard 330ml beverage cans), the mechanical friction provided

Table 2. System performance evaluation and integration test cases

Test Case ID	Feature Tested	Test Condition/ Input	Expected System Behavior	Actual Result	Status
TC-01	**Webhook Trigger**	Customer pays sufficient amount via SePay QR Code.	PHP backend updates DB status to 'paid' instantly. Backend detects mismatch, keeps status 'pending'.	DB updated successfully within ~1 second.	**PASS**
TC-02	**Underpayment**	Customer transfers less than the required amount.	Backend detects mismatch, keeps status 'pending'.	Order remains pending; machine does not dispense.	**PASS**
TC-03	**MQTT Latency**	Backend publishes SLOT:2 command to HiveMQ.	ESP32 receives command and activates Relay 1.	Relay clicks in < 0.5s after DB update.	**PASS**
TC-04	**Closed-loop Motor**	Motor spins the spiral coil. Limit switch is triggered.	ESP32 debounces signal, cuts relay, plays audio.	Motor stops exactly after 1 rotation (360°).	**PASS**
TC-05	**UI Polling**	Kiosk UI executes asynchronous AJAX polling at 3000ms intervals.	Redirects to success page when status is 'paid'.	Screen successfully transitions to 'Thank You'.	**PASS**

by the payload constrained the motor's angular velocity to its rated speed. Consequently, the physical engagement duration of the cam against the limit switch was **sufficiently long**, comfortably exceeding the $5000\mu s$ software debounce threshold. The algorithm achieved 100% reliability in these scenarios, successfully recognizing the falling edge and halting the motor precisely at 360 degrees.

- **Under unloaded extreme conditions** (operating without product payloads), the absence of mechanical damping significantly accelerated the rotor. The increased angular velocity compressed the switch engagement duration strictly below the $5000\mu s$ window. As a result, the discrete-time filter safely but erroneously classified the legitimate release edge as high-frequency contact bounce, rejecting the signal and intentionally triggering a secondary safety timeout rather than a premature cut-off.

- **Under heavy load conditions**, the increased physical resistance slowed the motor's angular velocity, thereby **significantly extending** the switch contact duration. Because the interrupt service routine is designed to trigger strictly on a validated falling edge rather than an absolute time duration, the algorithm remained perfectly stable and immune to mechanical lag.

This multi-load analysis confirms that the proposed microsecond-debounce algorithm guarantees high stability and precision for nominal and heavy payloads. However, it also highlights the tight coupling between software temporal parameters and physical load dynamics, indicating that the discrete-time threshold ($\tau_{debou\ nce}$) must be dynamically calibrated if the system is deployed for ultra-lightweight or unloaded operations.

Furthermore, from a User Experience (UX) perspective, the combination of the 1-second banking API synchronization and the 3000ms UI asynchronous polling interval results in a maximum perceived customer waiting time of approximately 1 to 4 seconds. This brief UI latency is well within acceptable industry standards, ensuring customers do not abandon their transactions. However, a potential limitation of this server-authoritative architecture is its vulnerability to temporary edge-node network disconnections. Because the PHP backend unconditionally deducts inventory and dispatches hardware commands upon payment verification, a localized Wi-Fi dropout at the ESP32 could result in unfulfilled transactions. As a preliminary fallback mechanism for future deployments, the integration of MQTT Quality of Service level 1 (QoS 1) with persistent sessions is proposed. In the event of a temporary disconnection, the HiveMQ broker will act as an offline cache, securely queuing the pending dispensing payloads. Once the ESP32 firmware automatically restores its Wi-Fi connection, the broker will reliably deliver the queued messages, guaranteeing that paid orders are physically fulfilled without requiring manual database reconciliation.

5. CONCLUSION

This paper successfully presented the research, design, and implementation of an IoT-based Smart Vending Machine integrating a cashless payment architecture. By replacing proprietary PLCs with a dual-core ESP32 edge microcontroller and utilizing the MQTT protocol, the proposed system achieves high-speed, secure telemetry at a fraction of the cost of traditional industrial models. The integration of an automated Webhook API effectively resolves the bottleneck of manual payment verification, providing a seamless digital transaction experience. Moreover, the application of a closed-loop cyber-physical control strategy employing limit switches and microsecond-debounce algorithms completely eradicates mechanical drifts and dispensing anomalies. The experimental results robustly validate the system's reliability, demonstrating significant potential for scalable, low-cost deployments in modern automated retail networks.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to Mr. Cao Van Vinh (Da Nang, Vietnam) for his technical consultation and for providing the baseline mechanical structural drawings of the product dispensing trays. His support significantly accelerated the hardware prototyping phase of this project.

REFERENCES

[1]. M. U. Tariq, et al., "Internet of Things Based Smart Vending Machine using Digital Payment System," in *Proc. IEEE International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2021.

[2]. P. B., et al., "IOT Based Smart Vending Machine Using Digital Payment System," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, 3, 1, 2023.

[3]. R. A., et al., "Design of an IoT-Based Snack Vending Machine with QR Code Payment, Automatic Stock Monitoring, and QoS Evaluation," *ResearchGate*, 2024.

[4]. F. Pazos, "Performance Evaluation of MQTT Broker Servers Deployed in the Cloud," *Eurasian Journal of Sustainability (EJS)*, 23, 1, 2024. doi: 10.24215/15146774e043.

[5]. R. Banno, "Performance Evaluation of MQTT Communication with Heterogeneous Traffic," in *Proc. IEEE Computers, Software, and Applications Conference (COMPSAC)*, 2023.

[6]. J. G. Ganssle, *A Guide to Debouncing - Hardware and Software Techniques*. The Ganssle Group, Rev. 2.28, 2014.

[7]. D. B. Stewart, "An Engineering Approach to Determining Sampling Rates for Switches and Sensors in Real-Time Systems," *ACM SIGBED Review*, 3, 1, 2006.

[8]. E. A. Lee, S. A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2nd ed. Cambridge, MA, USA: MIT Press, 2017.

[9]. Espressif Systems, *ESP32 Series Datasheet*. Espressif Systems (Shanghai) Co., Ltd., Version 4.4, 2024.

[10]. R. L. Norton, *Machine Design: An Integrated Approach*, 6th ed. Pearson Education, 2019. [Online]. Available: https://studylib.net/doc/27836599/machine-design-an-integrated-approach--robert-norton---z-...?p=380

[11]. Robosap, *24VDC High Torque Vending Machine DC Motor With Spring*. [Online]. Available: https://robosap.in/product/24vdc-high-torque-vending-machine-dc-motor-with-spring/.