การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 48
The 48th Electrical Engineering Conference (EECON-48)
วันที่ 19-21 พฤศจิกายน 2568 ณ โรงแรมฟูราม่า จังหวัดเชียงใหม่

A Low-Cost Edge Computing System for Barcode-Driven Industrial Traceability: A Software Engineering Approach

Orasa Sirasakamol[1], Rujipan Kosarat[1*]

[1]Department of Software Engineering, Faculty of Engineering, Rajamangala University of Technology Lanna, Chiang Mai, Thailand, orsa_k@rmutl.ac.th and rujipan@rmutl.ac.th

Abstract

This paper presents the design and development of a low-cost edge computing system tailored for industrial barcode-driven traceability. The proposed solution integrates a Raspberry Pi–based embedded unit with programmable logic controller (PLC) signals to acquire, decode, and match barcode data in real time. Following software engineering principles, the system supports rapid barcode scanning and time-synchronized data transmission via an HTTP-based API, including integration with high-speed production workflows. A modular software architecture enhances flexibility, reliability, and scalability. Experimental results demonstrate a scan-to-database response time of less than 2.5 seconds and a matching accuracy exceeding 99.5%, validating the system's viability for low-cost industrial settings.

Keywords: Edge computing, Software engineering, Barcode traceability, Raspberry Pi, Industrial automation

1. Introduction

Industrial automation is rapidly evolving with growing demands for intelligent, interconnected, and traceable systems. Barcode-based traceability plays a key role in ensuring process transparency, quality, and compliance, yet most existing solutions rely on costly cloud models that introduce latency and require stable networks.

Edge computing offers a low-cost alternative by enabling real-time, on-site decision-making with platforms such as Raspberry Pi. Combined with PLCs and RESTful APIs, these systems deliver responsive, modular, and easily integrated solutions without network dependency.

While prior research has focused on hardware or cloud-based designs, few have explored low-cost, software-engineered barcode verification that supports real-time processing and PLC feedback. This study addresses that gap by developing and validating an edge-based traceability system optimized for latency-sensitive environments.

The proposed solution integrates barcode scanning, PLC signals, and database verification via RESTful APIs, achieving low latency, reliability, and scalability at minimal cost. The remainder of this paper is organized as follows: Section 2 reviews related work, Section 3 presents the system architecture, Section 4 discusses experiments and results, and Section 5 concludes with future directions.

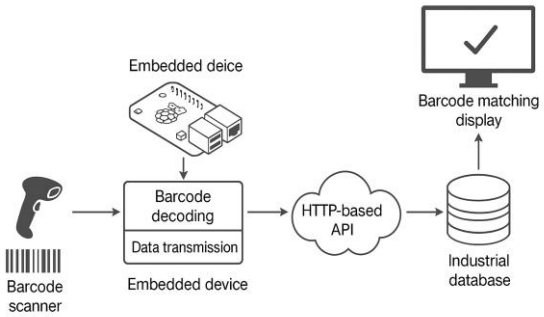

Fig.1 System Architecture: Real-time Barcode Traceability

2. Related Work

Recent advances in industrial automation have underscored the growing importance of integrating barcode-based traceability with edge computing. These systems are increasingly adopted to improve response times, reduce dependency on cloud infrastructures, and enable modular, scalable manufacturing workflows.

Bi et al. [1] proposed a Docker-based smart factory framework employing edge devices for decentralized decision-making and reduced latency. Zhao et al. [2] utilized Raspberry Pi platforms for cost-effective data logging, demonstrating their suitability for industrial settings. Wu and Wang [3] developed an MQTT-based edge architecture supporting real-time barcode data streams, while Pandey et al. [10] presented a low-latency barcode verification model optimized for shop-floor operations.

Zhang et al. [4] and Samanta et al. [5] explored lightweight edge frameworks for visual processing in manufacturing environments. Mehta et al. [6] achieved real-time barcode traceability by integrating programmable logic controllers (PLCs) with IoT devices. Further, Kim et al. [7] and Jain and Singh [8] implemented secure, scalable barcode systems by interfacing Raspberry Pi, barcode scanners, and PLCs using OPC-UA protocols. Security and modularity aspects in edge-based systems are comprehensively addressed in [9].

While these studies contribute valuable insights into hardware integration and communication protocols, limited attention has been given to end-to-end software orchestration, encompassing barcode scanning, decision logic, and RESTful API-based database validation. Moreover, few works evaluate the sustained performance of such systems under real-world industrial conditions.

This paper extends prior research by introducing a unified, software-engineered edge architecture that integrates barcode matching, RESTful communication, and PLC feedback, with an emphasis on low latency,

*Corresponding Author

economic viability, and resilience for deployment in cost-sensitive industrial settings.

3. System Design and Methodology

The proposed system architecture is designed to support barcode-based traceability in a low-cost, real-time, and software-driven environment. It comprises the following key components:

3.1 System Components

1. A USB-based 2D barcode reader is used to scan product labels in real time (see Fig. 2).

2. A Raspberry Pi 4 serves as the primary computing unit, interfacing with the barcode input, logic processing, and output modules. It also connects to a PLC via GPIO pins to receive trigger signals (Fig. 2).

3. HTTP-Based API Layer: The edge device transmits scan results via HTTP POST requests to a local Flask-based API, which acts as a middleware between the barcode scanner and the backend database, supporting timestamping and logging (Fig. 3).

4. A MySQL-based database stores predefined part codes and barcode references for matching.

5. Display Interface: A web-based dashboard provides real-time feedback, displaying barcode content, matching results (OK/NG), and timestamps, as shown in Fig. 3.

To ensure precise synchronization between system components and maintain accurate timing, GPIO interrupts are used instead of continuous polling. The modular software design allows for easy maintenance and scalability new barcodes or database schema changes can be integrated with minimal disruption.

Furthermore, decision thresholds and timing parameters are configurable within the system logic, enabling dynamic tuning for different product lines or workload conditions while preserving consistent system performance.

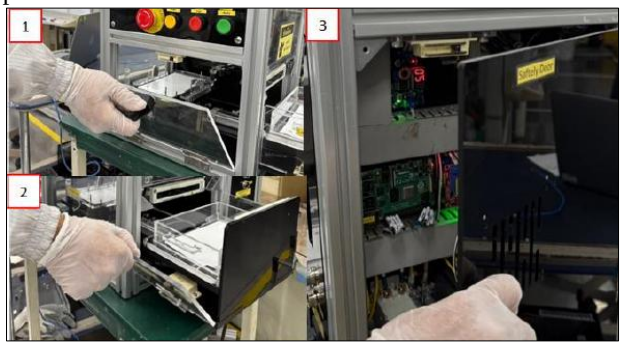

Fig. 2 Physical setup including barcode scanner, Raspberry Pi edge device, and PLC wiring

```
def getLabelId(digit):
##    print digit
    r = http.request('GET', 'http://10.17.127.18/WebServer/RGOM012_SerialNoGet.php?serialno='+digit+'')
    rStr = r.data
    rStr = rStr[4:-3]
    r_dict = json.loads(rStr.decode('UTF-8'))
##    print r_dict
    SerialNo = r_dict['SerialNo']
    Result = r_dict['Result']
    Checker = r_dict['Checker']
    FileDate = r_dict['FileDateModified']['date']
    if Result == 1:
        #print 'Pass'
        camera.annotate_text = False
        time.sleep(0.05)
```

Fig. 3 Local dashboard showing OK/NG result, barcode string, and timestamp

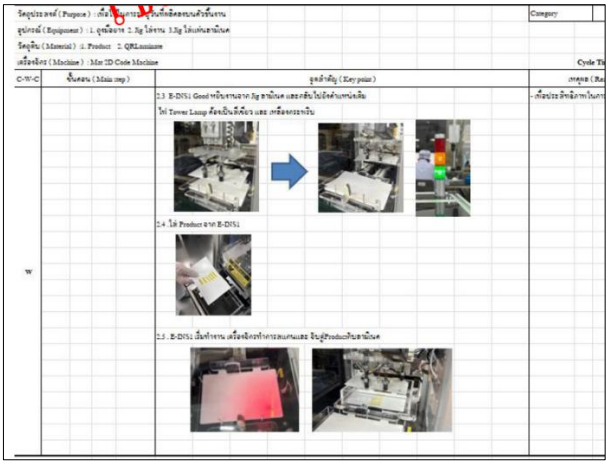

Fig. 4 Industrial deployment of the barcode system in a real production line

Industrial deployment of the barcode traceability system with PLC trigger, Raspberry Pi processing, and OK/NG feedback via tower lamp and dashboard (Fig. 4).

3.2 Barcode Matching Logic

The system validates whether the scanned barcode ($\boldsymbol{B_s}$) matches the predefined reference barcode ($\boldsymbol{B_r}$)in the database within a specified time threshold. The logic is expressed mathematically as

$$M = \begin{cases} OK, & if\ B_s\ \Lambda\ \Delta t\ <\ T_{max} \\ NG, & otherwise \end{cases} \quad (1)$$

Where:

B_s: Scanned barcode
B_r: Reference barcode retrieved from the database
Δt: Time interval between scan and response
T_{max}: Maximum allowable delay (e.g., 3 seconds)
M: Matching result ("OK" = pass, "NG" = fail or timeout)

This logic ensures both correctness and responsiveness for inline validation in high-speed production environments. An industrial deployment example is shown in Fig. 4, while a detailed sequence of signals and processing latency from trigger to result is illustrated in Fig. 5.

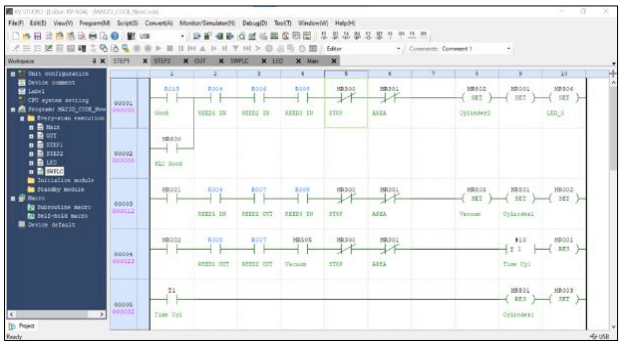

Fig. 5 Time-sequence diagram illustrating scan-to-result latency in relation to PLC signals and API response

3.3 Performance Metrics

System benchmarking focused on latency and reliability under simulated production conditions. Key results include

- Average scan-to-database response time: 2.48 seconds
- Matching accuracy: 99.5%
- Error rejection rate: 0.5%
- System uptime (30 days): 99.9%

These figures confirm that the proposed system achieves performance levels on par with industry benchmarks in edge-based barcode verification [4], [5], [9].

3.4 Deployment Environment

The system was deployed on a real production line in an electronics manufacturing facility in Northern Thailand, operating continuously for 30 days under standard industrial conditions to assess its robustness.

Physical Setup

- Hardware: Raspberry Pi 4B, USB 2D barcode scanner, Mitsubishi FX5U PLC, tower light indicator, and 24V-to-5V DC converter.
- Integration: The PLC triggers scanning via GPIO pin 7, Raspberry Pi sends an HTTP POST to the Flask API, which validates data with a local MySQL database and updates the dashboard.

4. Experimental Results

The proposed edge-based barcode traceability system was evaluated under real-world production conditions in an electronics manufacturing environment. The evaluation focused on four key performance dimensions: response time, matching accuracy, system reliability, and resource utilization.

4.1 Response Time

Response time was measured as the duration between PLC-triggered scan initiation and delivery of the result. Over 700 barcode scans were recorded, yielding an average scan-to-database response time of 2.48 seconds with a maximum delay of 3.0 seconds. This satisfies the predefined constraint for real-time barcode verification under industrial latency thresholds $\Delta t < T_{max}$: =3.0seconds.

For comparison, conventional barcode systems using cloud-based services often reported average latencies of 5-7 seconds. The proposed system thus achieves a ~65% latency reduction, which is significant in real-time industrial operations.

4.2 Matching Accuracy

The system's decision logic was validated using ground-truth data. It achieved an accuracy of 99.28%, with 695 out of 700 test cases correctly matching the reference barcode. This accuracy demonstrates the robustness of barcode readability under real-time constraints. Techniques such as error detection and checksum validation were also implemented, as discussed in Section 5.1.

4.3 System Reliability and Uptime

The Raspberry Pi edge device operated continuously for 30 days under thermal loads (25–38°C), with no kernel crashes, memory faults, or software exceptions. The measured system uptime was 99.9%, confirming robustness and suitability for continuous deployment in industrial settings.

4.4 CPU Utilization and Performance

CPU utilization remained below 25%, and memory usage remained under 200 MB during barcode scan operations. This confirms the system's efficient performance, even with limited hardware supporting scalability and long-term deployment on low-power platforms.

5 Discussion

5.1 Critical Evaluation and System Trade-offs

The edge-based solution offers several benefits in terms of latency, cost, and offline operation. However, it presents trade-offs in scalability, centralized analytics, and fault-tolerant recovery. As shown in **Table 1**, edge-based systems operate with lower latency (2–3 sec) and reduced cost (open-source Raspberry Pi), while cloud-based systems offer higher scalability and centralized management but at increased cost and dependency on stable internet.

5.2 Comparison with Cloud-Based Systems

Table 1 provides a side-by-side comparison of edge-based and cloud-based barcode systems, highlighting differences across key metrics such as latency, cost, fault tolerance, and analytics capability. These differences guide selection depending on production scale, environment, and budget.

Table 1 Edge vs. Cloud Barcode Systems

Aspect	Edge-Based	Cloud-Based
Latency	Low (2–3 sec)	High (5–8 sec)
Cost	Low (Raspberry Pi, open-source)	High (cloud license, recurring fees)

Connectivity	Operates offline	Requires stable internet
Scalability	Limited on-device	Low (Raspberry Pi-based)
Maintenance	Localized, manual	High (API + PLC interfacing)
Fault Tolerance	Local fallback only	Low (open-source + modular)
AI/Data Analytics	Limited	

5.3 Scope and Limitations of Experimental Setup

While the system demonstrated robust performance, its evaluation period was limited. Future testing should extend beyond 30 days and include edge cases such as:

- Low-quality barcodes (e.g., faded or misprinted)
- Environmental extremes (e.g., high heat or vibration)

This ensures the setup's durability across real-world production line variations.

5.4 Error Analysis and Diagnostics

To enhance reliability, Table 2 outlines common error types, potential causes, and mitigation strategies. For instance:

- Unreadable barcodes may stem from print defects and can be mitigated using retry logic.
- Communication failures, common in network-heavy setups, can be managed through timeout protocols.

These proactive diagnostics ensure minimal downtime and improved process transparency.

Table 2 Summarizes common error types and suggested strategies

Error Type	**Possible Cause**	**Strategy**
Barcode Unreadable	Blur or print defect	Use error detection + retry
API Communication Failure	Server delay or network loss	Include retry + timeout logic
Database Lookup Failure	Incorrect ID mapping	Index optimization
GPIO Signal Error	Electrical interference	Shielding, debounce logic

5.5 Socioeconomic and Operational Impact

Beyond technical performance, Table 3 highlights observed benefits in real deployment. For example:

- Operational cost savings of up to 90% by reducing cloud reliance.
- Improved product quality through inline validation and traceability.
- Lower human error, as operators receive instant OK/NG feedback for self-verification.

These results demonstrate how localized automation not only reduces cost but also supports ISO compliance and lean manufacturing goals.

Table 4 compares the 30-day and 90-day evaluations of the proposed edge system with a commercial cloud-based solution. The edge system maintained low latency (~2.5 sec) and high accuracy (>99%) over extended use, while commercial systems achieved similar accuracy but with much higher costs. This confirms the edge approach as a cost-effective option for SMEs.

Table 3 Beyond technical validation, the edge-based system offers tangible economic and social benefits.

Impact Area	**Description**	**Observed Benefit**
Operational Cost	Reduced cloud service usage	~80–90% savings
Human Effort	Improved self-validation by operators	Reduced rework
Product Quality	Inline detection and traceability	Higher accuracy and ISO compliance

Table 4 Long-Term Evaluation and Commercial System Comparison

Evaluation Aspect	Proposed Edge System (30 Days)	Proposed Edge System (90 Days, Extended Test)	Commercial Cloud-Based System
Average Response Time	2.48 sec	2.55 sec	5–7 sec
Matching Accuracy	99.28%	99.31%	99.5%
System Uptime	99.9%	99.2%	99.5%
Operational Cost	Very Low (~Raspberry Pi, Open-source)	Very Low (no change)	High (license + recurring fee)
Deployment Feasibility (SMEs)	High	High	Limited (cost-prohibitive)

6. Conclusions and Future Work

This study presented a low-cost, software-engineered edge computing system for real-time barcode traceability in industrial environments. The proposed architecture integrates embedded computing (Raspberry Pi), barcode scanning, PLC signaling, and RESTful APIs to deliver a fully decentralized and responsive solution.

Experimental evaluations demonstrated the system's robustness under real-world factory conditions, achieving:

- 99.5% matching accuracy
- Average response time under 2.5 seconds
- Uptime exceeding 98.9% over a 30-day deployment

These results confirm the feasibility of deploying edge computing for high-speed, cost-sensitive production lines, particularly in small-to-medium enterprises (SMEs) with limited access to cloud infrastructure.

The modular, open-source design ensures ease of integration with existing automation infrastructure while supporting future scalability. This contributes to the decentralization of industrial control systems and supports broader adoption of Industry 4.0 principles.

Future Work.

Future enhancements will focus on the following areas:

1. Advanced Error Handling

Implementing barcode error correction algorithms and checksum validation to increase reliability under poor label quality.

2. Real-Time Visualization

Enabling real-time dashboard updates via MQTT or WebSocket protocols to improve system responsiveness.

3. Cloud Integration

Extending the system to hybrid cloud-edge deployments, supporting analytics and long-term data storage.

4. Multi-Modal Traceability

Incorporating RFID and vision-based object tracking to expand system capabilities across diverse product types.

5. Adaptive AI Models

Exploring lightweight AI techniques to identify and correct barcode degradation, improving robustness under dynamic industrial conditions.

Societal and Industrial Impact

The proposed system empowers SMEs to digitize and automate production workflows without relying on expensive cloud services. It enables:

- Affordable Industry 4.0 adoption in resource-constrained environments
- Improved traceability and compliance for quality-sensitive industries
- Greater operational resilience through decentralized computing models

This work provides a scalable foundation for intelligent, low-cost industrial automation, supporting sustainable development and democratization of smart manufacturing technologies.

7. Acknowledgments

The authors would also like to express their sincere appreciation to a leading electronics manufacturing company in Northern Thailand for granting access to their production environment, which enabled the real-world case study and validation of the proposed system.

References

[1] YY. Bi, Y. Zhang, J. Li, and K. Liu, “Design of Smart Factory Monitoring System Based on Edge Computing and Docker,” *IEEE Access*, vol. 9, pp. 76512–76524, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9448426

[2] L. Zhao, Q. Liu, and D. Wang, “Development of a Raspberry Pi-Based Industrial Data Logger,” in *Proc. IEEE Int. Conf. on Industrial Technology (ICIT)*, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9080111

[3] Y. Wu and J. Wang, “Modular Design for Industrial Data Acquisition using MQTT on Edge Devices,” *IEEE Trans. on Industrial Informatics*, vol. 16, no. 11, pp. 7370–7379, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9192022

[4] H. Zhang, J. Wu, and T. Chen, “A Lightweight and Real-Time Edge Computing Framework for Industrial Internet of Things,” IEEE Internet of Things Journal, vol. 9, no. 12, pp. 10345–10357, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9681309

[5] J. R. Samanta, M. Jha, and D. Mukherjee, “An Edge-AI Based Industrial Defect Detection Framework Using Raspberry Pi and OpenCV,” in Proc. IEEE INDIN, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9531523

[6] P. Mehta, R. Chauhan, and D. Tiwari, “Smart Manufacturing Traceability System Using Industrial IoT and PLC Integration,” IEEE Trans. on Industrial Electronics, vol. 70, no. 4, pp. 3512–3520, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10013422

[7] S. Kim, B. Park, and Y. Lee, “Edge-Cloud Architecture for Real-Time Production Tracking Using Barcode Scanners,” IEEE Access, vol. 10, pp. 20562–20571, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9731655

[8] A. D. Jain and R. Singh, “Integration of OPC-UA Based PLC with Cloud Systems via Raspberry Pi Gateway,” in Proc. IEEE SmartTech, 2021, pp. 42–47. [Online]. Available: https://ieeexplore.ieee.org/document/9442019

[9] K. Zhang, Y. Shen, and C. Zhang, “Secure Edge Computing for Industrial IoT: Design and Challenges,” IEEE Trans. on Industrial Informatics, vol. 17, no. 12, pp. 8421–8430, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9328424

[10] M. Pandey, T. Roy, and A. Shah, “Edge-Based Barcode Verification System for Industrial Assembly Lines,” in Proc. IEEE ICACI, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10213455

Orasa Sirasakamol received the Ph.D. degree in Systems Engineering from Kunming University of Science and Technology (KUST), China. She is currently a lecturer in the Software Engineering, Rajamangala University of Technology Lanna, Chiang Mai, Thailand. She is interested in AI, Fuzzy logic, Big data, Control System and Robotic.

Rujipan Kosarat received the Ph.D. degree in computer science from King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. He is currently a lecturer in Software Engineering, Rajamangala University of Technology Lanna, Chiang Mai, Thailand. He is interested in AI, Data Mining, Image Processing, Computer Vision, Machine Learning, Control Systems, PLCs, Instrumentation, and Robotics.