ACIG
APPLIED CYBERSECURITY & INTERNET GOVERNANCE

NASK

Performance and Simulation-Based Comparison of IoT Access Control Systems Using ESP32, Arduino Uno, and Raspberry Pi Pico W

Svitlana Sotnik | Department of Computer-Integrated Technologies, Automation, Robotics, and Safety Engineering, Kharkiv National University of Radioelectronics, Ukraine | ORCID: 0000-0002-6035-2388

Abstract

This article presents the development and comparative analysis of automated Internet of Things (IoT)-based access control systems across various microcontroller platforms. The study aims to compare the performance, reliability, and energy efficiency of the ESP32, Arduino Uno, and Raspberry Pi Pico W microcontroller platforms for use in automated IoT-based access control systems. To achieve this goal, three alternative implementations of the access control system were developed, and a series of experiments were conducted using simulation modelling in the Wokwi environment, supplemented by analytical modelling. The results showed that the ESP32 achieved the best performance, with a response time of 287 milliseconds, a network delivery reliability of 98.7%, and an overall system reliability of 97.3%, which aligns with indicative thresholds for high-reliability IoT security applications under the simulation conditions used. The Raspberry Pi Pico W platform demonstrated the highest energy efficiency at 180 mW, while Arduino Uno, due to its 94.2% network delivery reliability, an overall system reliability of 88.6%, and lower speed, is suitable only for educational purposes. It is concluded that the ESP32 is suitable for high-performance

Received: 29.11.2025

Accepted: 24.04.2026

Published: 18.06.2026

Cite this article as:
S. Sotnik, "Performance and simulation-based comparison of IoT access control systems using ESP32, Arduino Uno, and Raspberry Pi Pico W," ACIG, vol. 5, no. 1, 2026, doi: 10.60097/ACIG/221089

Corresponding author:
Svitlana Sotnik, Department of Computer-Integrated Technologies, Automation, Robotics, and Safety Engineering, Kharkiv National University of Radioelectronics, Ukraine; E-mail: svetlana.sotnik@nure.ua
0000-0002-6035-2388

systems, and the Raspberry Pi Pico W for autonomous solutions. Promising directions for further research are identified, including the integration of biometric sensors and machine learning for detecting abnormal access attempts.

Keywords

access control, biometric authentication, IoT system, microcontroller platform

1. Introduction

Today, with the increasing security needs of critical infrastructure facilities, industrial enterprises, and commercial premises, access control systems are becoming increasingly relevant. They allow for effective restriction of unauthorised access and monitoring of personnel movements in protected areas, making them an integral part of modern security systems [1–4].

Modern Internet of Things (IoT) technologies open up new opportunities for creating flexible access control systems based on microcontroller platforms [5–9]. The combination of biometric sensors, radio-frequency identification (RFID) readers, and wireless data transmission modules allows for the implementation of autonomous systems capable of transmitting information about security events to a local network or remote servers. The key issue in designing such systems is the selection of the optimal hardware platform that will provide a necessary balance between speed, data transmission reliability, and energy efficiency.

There are a significant number of microcontroller solutions on the market, differing in architecture, performance, built-in communication modules, and cost. The lack of systematic comparative studies complicates the choice of a platform for specific application conditions. In particular, ESP32 offers integrated wireless fidelity (Wi-Fi) and Bluetooth modules, Raspberry Pi Pico W is notable for its energy efficiency, and Arduino Uno remains a popular solution due to its simplicity and affordability.

The purpose of this study is to conduct a comparative analysis of the performance, reliability, and energy efficiency of the ESP32, Arduino Uno, and Raspberry Pi Pico W platforms for use in automated IoT access control systems. To achieve this, simulation modelling was performed in the Wokwi environment under controlled identical conditions, supplemented by analytical assessment of

energy consumption and identification characteristics based on technical datasheets.

The following research questions guided this study:

1. Which platform demonstrates the lowest response time (t_o) in the authentication cycle?
2. Which platform achieves the highest reliability in terms of False Acceptance Rate/False Rejection Rate (FAR/FRR) metrics?
3. Which platform offers the best energy efficiency for autonomous deployment?

We hypothesize that ESP32, due to its integrated wireless modules and higher clock speed, will outperform the other platforms in response time and reliability, while Raspberry Pi Pico W will demonstrate superior energy efficiency.

2. Problem Statement

The increasing level of threats to infrastructure and business makes automation of access control one of the key priorities. Modern security requires intelligent systems capable of reliably, instantly, and cost-effectively recognising users and providing remote real-time monitoring of events.

The existing access control systems often face problems with identification reliability and request processing speed, especially under high load conditions or when working with a large number of users. These limitations significantly affect the efficiency and security of facilities, as delays in processing requests or identification of errors can lead to unauthorised access or, conversely, to blocking access for authorised users.

Integrating access control systems with modern IoT technologies and cloud platforms remains a challenging task. Many existing solutions lack effective means for real-time data transfer and centralised management of distributed access points, which limits their application in large facilities with multiple protected areas. This problem is particularly acute in the context of the need for rapid response to security incidents.

In light of these challenges, there is an urgent need to develop an integrated access control automation system that combines reliable identification with the ability to effectively transfer data via IoT platforms. Such a system must be energy-efficient, affordable, and

capable of operating in a variety of operating conditions typical for guarded facilities of various purposes.

The development of a simulation model of the access control automation system appears to be a promising solution that can meet the above requirements and allow the system to be tested before its physical implementation.

The novelty of this work lies not merely in the comparison of three platforms but in the development and application of a unified multi-criteria evaluation framework that simultaneously assesses response time, FAR/FRR identification reliability, network stability, and energy consumption under identical controlled conditions. This framework – combining simulation-based benchmarking with analytical modelling based on technical datasheets – provides a replicable methodology for hardware platform selection in IoT security systems before physical prototyping, which has not been previously applied in this specific context of microcontroller-level IoT access control evaluation.

The mathematical formulation includes a model of the user identification process:

$$t_{ide} = t_{red} + t_{pr} + t_{dm}, \quad (1)$$

where t_{red} – reading time;

t_{ide} – user identification time;
t_{pr} – data processing time; and t_{dm} – time of decision on granting/denying access.

Along with time characteristics, a key indicator of system quality is its accuracy. To quantitatively assess accuracy and measure the level of confidence in the system, an identification reliability model is used, which determines the probability of its error-free operation:

$$P_{rl} = 1 - (P_{FAR} + P_{FRR}) \quad (2)$$

where P_{FAR} – probability of false admission of an unauthorised user; and P_{FRR} – probability of false rejection of an authorised user.

Accurate identification means correctly recognising the user and making the right decision about granting or denying access.

There are two types of correct decisions: when the system has let an authorised user through (correctly recognised and allowed) and

when the system has not let an unauthorised user through (correctly detected the intruder and blocked). Accordingly, there are two types of errors: FAR (falsely allowed an «outsider») and FRR (falsely denied an «insider»).

For further analysis and calculation of error probabilities (FAR and FRR), it is necessary to mathematically describe their nature. It is assumed that the identification error model can be modelled as a random variable with a normal distribution:

$$\varepsilon \sim N(0,\sigma^2) \tag{3}$$

where N – normal (Gaussian) distribution of the values of the identification error itself; and σ – standard deviation (*SD*) of the identification error.

At the same time, $N(0,\sigma^2)$ the error, follows a normal, or Gaussian, distribution. A mathematical expectation equal to zero indicates the absence of systematic error. This means that, on average, the system does not have a consistent bias towards underestimating or overestimating results, viz. it is unbiased. Dispersion σ^2 is a measure of the spread or uncertainty of errors around this zero mean value.

The error variable ε in Eq. (3) represents the deviation of the biometric similarity score from its expected value. Decision thresholds τ_{FAR} and τ_{FRR} are applied asymmetrically to this distribution to reflect different costs of security errors:

$$\text{FAR} = P(\varepsilon > \tau_{\text{FAR}}) = 1 - \Phi\left(\frac{\tau_{\text{FAR}}}{\sigma}\right), \tag{4}$$

$$\text{FRR} = P(\varepsilon > -\tau_{\text{FRR}}) = \Phi\left(\frac{-\tau_{\text{FRR}}}{\sigma}\right), \tag{5}$$

where $\Phi(\cdot)$ is the standard normal Cumulative Distribution Function (CDF) and σ is as defined in Eq. (3). Since $\tau_{FAR} \neq \tau_{FRR}$ in security applications – false acceptance being more critical than false rejection – FAR and FRR take different values despite sharing the same underlying distribution.

The effective σ varies across platforms: ESP32 benefits from hardware-accelerated universal asynchronous receiver/transmitter (UART), while Pico W relies on the MicroPython software UART, introducing additional timing jitter, and Arduino Uno requires an

external module with additional protocol overhead – each resulting in a progressively larger effective σ. The specific FAR and FRR values in Table 5 are analytical estimates derived using this framework, with parameters consistent with published real-world performance ranges for optical fingerprint sensors [18, 19], and are intended for comparative platform assessment only. Empirical validation with physical hardware is planned as future work.

In addition to accuracy and reliability, the system's performance and efficiency in a network environment are critical for practical application. Additional models are introduced to analyse these aspects. Thus, the system's throughput is determined by the maximum number of requests processed per unit of time and is described by the model:

$$N_{\max} = \frac{1}{t_{\text{ide}}}, \tag{6}$$

where $N_{\max}$ – maximum number of access requests per unit of time.

For systems integrated into the IoT infrastructure, it is also necessary to consider data transmission characteristics, which are modelled as:

$$D_{\text{trf}} = \frac{L}{B \cdot (1 - P_{\text{los}})}, \tag{7}$$

where D_{trf} – data transfer time via the IoT platform, sec;

B – channel capacity, bits/sec;
L – data packet size, bits;
P_{los} – probability of data packet loss.

Along with internal characteristics such as accuracy and performance, external factors significantly affect the overall performance of the system, especially in a distributed IoT infrastructure. These critical factors include data transmission reliability and energy efficiency. The exchange of information between system components (readers, servers, and databases) via wireless communication channels requires an assessment of the probability of data loss. For autonomous battery-powered devices, energy consumption optimisation becomes a determining factor.

Physical data transmission reliability model:

$$P_{\text{err}} = 1 - (1 - p)^k, \tag{8}$$

where p – probability of a single bit transmission error; and k – number of bits in a message.

Thus, P_{err} there is a probability of losing information about an access event due to communication errors, which directly affects the integrity of the monitoring and logging system.

Therefore, in this study, there are two types of reliability: identification reliability P_{rl} (decision-making quality) and physical data transmission reliability (information flow integrity), P_{err}, which together determine the overall performance.

The analysis of data transmission reliability directly affects the energy efficiency requirements of the system, especially for autonomous devices. To ensure a low probability of error P_{err} in challenging communication conditions, it is necessary to increase the energy per bit transmitted (e.g. by increasing transmitter power or using more complex coding schemes that require more computation):

$$E_{con} = E_{idl} + N_{ev} \cdot E_{ev}, \tag{9}$$

where E_{con} – total energy consumption of the access control system;

E_{idl} – power consumption in standby mode;
N_{ev} – number of access events during the period under review; and
E_{ev} – energy required to process a single event.

The planned results include mathematical models of the access control system for assessing identification accuracy, data transmission reliability, and energy consumption as well as simulation models based on ESP32, Arduino Uno, and Raspberry Pi Pico W for experimental verification of theoretical calculations. Subsequently, a comparative analysis of the effectiveness of different hardware platforms will be carried out based on the criteria of speed, reliability, and energy efficiency.

This will ensure a comprehensive approach to the design, verification, and implementation of an automated access control system with a well-founded choice of the optimal hardware platform.

3. Review of the Literature

Access control systems have been extensively studied in the context of IoT and embedded systems, yet a critical gap remains

in systematic comparative evaluations of microcontroller platforms under identical conditions.

Early research established centralised architectures as the dominant model, but subsequent studies revealed their vulnerability to single points of failure [10]. This led to the emergence of hybrid architectures where critical decisions are made locally on the device while a central server handles logging and threat analytics [11]. Although this shift improved fault tolerance, the question of which hardware platform best supports such local decision-making remains underexplored.

Regarding identification methods, multi-factor authentication combining RFID, biometric verification, and Global System for Mobile (GSM) communication tokens has demonstrated strong reliability [12], while near-field communication (NFC)-based systems have been extended beyond access control to employee time-tracking applications [13]. However, these studies focus primarily on authentication protocols rather than the underlying hardware performance, leaving platform selection largely unaddressed.

Platform-specific studies exist but are narrow in scope. The Arduino Uno-based systems have proven viable for basic RFID access control, yet lack real-time notification mechanisms and support only single-factor authentication [14]. ESP32 has been applied in more complex scenarios, including facial recognition via MTCNN, though without systematic measurement of response time – a critical parameter for real-time systems [15]. Raspberry Pi Pico W has been evaluated primarily in the context of cryptographic security for wireless sensor networks, without addressing the full access control cycle performance [16].

Despite these valuable contributions, each platform has been assessed in isolation and for different purposes, making direct comparison impossible. Consequently, no existing work provides a unified experimental evaluation of ESP32, Arduino Uno, and Raspberry Pi Pico W across response time, reliability, and energy efficiency within the same access control scenario.

Furthermore, no prior study has combined simulation-based platform comparison with analytical modelling of FAR/FRR, energy consumption, and network delivery reliability within a single unified framework. This study addresses this gap by providing the first such integrated evaluation.

4. Materials and Methods

4.1. System Architecture

The design of the access control automation system will be based on the IoT architecture with an emphasis on reliability, security, scalability, and energy efficiency [13–15, 17]. A modular architecture was adopted, providing division into specialised interconnected subsystems (Figure 1).

The core of the system is an access control subsystem responsible for user authentication and physical access management. Its central element is the ESP32 microcontroller, which collects data from two authentication devices: an NFC scanner and a fingerprint scanner. User authentication is implemented via two-factor verification, combining contactless card identification with subsequent biometric confirmation.

System resistance to attacks is reinforced through the use of MIFARE DESFire EV1/EV2 contactless cards, which employ DES/3DES/

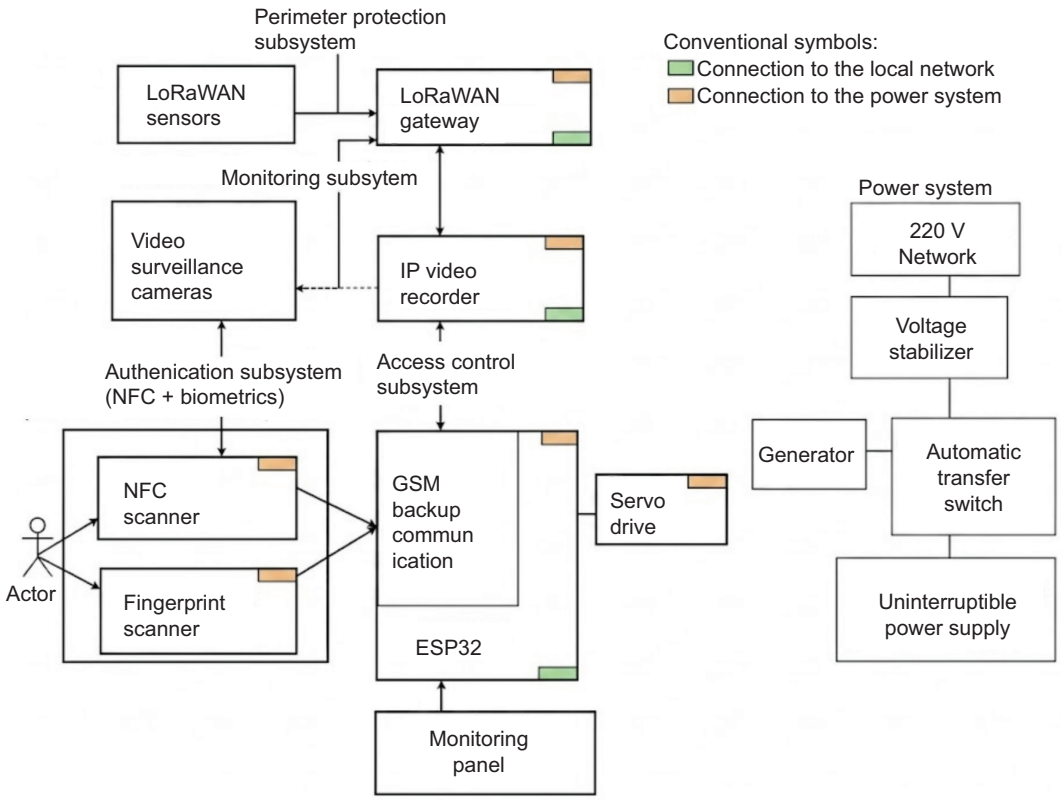

Figure 1. Architecture of an IoT system for a protected facility.

AES-128 cryptographic algorithms, dynamic authentication, protection against unauthorised recording, and differentiation of access rights at the application level. The NFC and fingerprint modules are integrated with the ESP32 via serial peripheral interface (SPI) and UART interfaces, respectively. To ensure stable operation under unreliable power conditions, both peripheral devices are equipped with individual stabilised 5 V power supplies connected via an uninterruptible power supply (UPS). A positive authentication result triggers pulse-width modulation (PWM) signal generation to control a servo drive that mechanically opens the access device, powered from a separate stabilised source to isolate the control link from interference.

The territory-monitoring subsystem provides continuous surveillance through a network of Power over Ethernet (PoE)-powered Internet Protocol (IP) cameras, with video streams recorded on a central IP recorder for structured archiving. An additional protection layer is provided by a perimeter control subsystem using wireless sensors with the LoRaWAN protocol, selected for its low power consumption, long communication range (up to 10–15 km in line of sight), and ability to operate in remote areas without wired infrastructure. This subsystem monitors intrusion attempts and object movement, transmitting alarm signals via a LoRaWAN gateway for processing in the local network.

The power supply architecture is based on redundancy principles to eliminate downtime. The primary source is a stabilised 220 V network, backed by a diesel generator connected automatically via an automatic transfer switch (ATS) unit. Key components are additionally equipped with UPS units to bridge the interval between mains failure and generator activation. Backup GSM communication is maintained through a dedicated module, enabling the system to send SMS alerts and automatic calls to the security service even when the primary network infrastructure is unavailable.

The proposed architecture thus integrates wireless communications, cryptographic data protection based on the DESFire standard, comprehensive territory and perimeter monitoring, and reliable backup power with emergency communication. This multi-level security model provides effective protection against both physical and cyber threats, making the system suitable for facilities with elevated security requirements.

The simulation model was designed to reflect the architecture of a physical system intended for real-world deployment. The target

hardware configuration includes: an R503 fingerprint sensor and PN532 NFC module for authentication; an ESP32-WROOM-32 DevKitC v4 board with SIM800L GSM/GPRS module as the access controller; an MG996R servo drive for lock actuation; a Raspberry Pi 4 Model B with a 7" display as the control unit; Dragino LSN50 v2 sensors and Hikvision DS-2CD2043G0-I cameras for perimeter monitoring; and a TP-Link TL-SG1210MP switch with redundant power supply, including a LogicPower LPT-W-500VA stabilizer, Schneider Electric ATSE 63A automatic transfer switch, Honda EU22 generator, and APC Smart-UPS 1500VA. However, as noted in Section 4.2, these components were not available for simulation in Wokwi and are intended for validation at the physical prototyping stage.

It should be noted that while the proposed architecture includes IP cameras, LoRaWAN sensors, and redundant power supply components, the simulation conducted in the Wokwi environment focused specifically on the core access control logic and microcontroller-level interactions. The monitoring, perimeter, and power supply subsystems are presented as part of the complete system design intended for physical implementation. Their detailed simulation and empirical testing are beyond the scope of the current study and are planned as directions for future work.

4.2. Simulation Environment and Limitations

The Wokwi online simulator was selected as the primary development and testing environment for this study. This platform enabled the design, implementation, and functional verification of three alternative access control system configurations under controlled and identical conditions, without the need for physical hardware prototypes at this stage of research.

Wokwi supports the simulation of core microcontroller components used in this study, including the ESP32 DevKit V1, Arduino Uno, and Raspberry Pi Pico W boards, as well as standard peripheral devices such as servo drives, passive infrared (PIR) motion sensors, light-emitting dioded (LEDs), liquid crystal displays (LCDs), potentiometers, and push buttons. This allowed the authentication logic, event logging, servo-based lock control, and Telegram Bot API integration to be implemented and tested within a unified virtual environment.

However, the simulator imposes a number of technological limitations that defined the boundaries of the experimental component of this study. Wokwi does not support full emulation of biometric fingerprint sensors or NFC/RFID modules. Consequently, user

authentication was simulated via push buttons acting as conditional inputs, replicating the logical behaviour of these components without reproducing their physical signal characteristics. Similarly, real power parameters, such as current draw and instantaneous power consumption, cannot be measured within the simulator. Therefore, energy consumption values reported in this study were estimated analytically based on the technical datasheets of the respective microcontrollers and their peripheral components under typical operating conditions.

External infrastructure components, including UPS units, voltage stabilizers, diesel generators, LoRaWAN gateways, and IP cameras, are not supported by the Wokwi environment and were therefore excluded from the simulation scope. These components form part of the complete system architecture described in Section 4.1 and are intended for implementation and testing at the physical prototyping stage.

Response time and data transfer latency were measured by recording software cycle execution timestamps within the simulator, rather than capturing physical signal propagation between real hardware modules. All reported values represent the mean of 10 repeated test cycles per platform, expressed as mean ± *SD*.

4.3. Experimental Platforms

Three microcontroller platforms were selected for comparative evaluation, each representing a distinct approach to building an automated IoT access control system.

ESP32 DevKit V1 was chosen as the primary platform due to its advanced capabilities for IoT applications. Its key advantages include built-in Wi-Fi and Bluetooth modules, dual-core architecture with FreeRTOS multithreading support, and a large number of digital and analog GPIO pins. These features enabled the implementation of the most functionally complete configuration, including two-factor authentication, real-time Telegram Bot API notifications, PIR-based motion detection, and battery-level monitoring via an analog potentiometer. Full compatibility with the Arduino IDE and the Wokwi simulator established this platform as the baseline configuration for this study.

The connection diagram of components to the ESP32 microcontroller is shown in Figure 2A. The pinout diagram of the ESP32 board with the designation of the functions of each pin is shown in Figure 2B.

The access control system built on ESP32 requires two-step identification from the user. First, the correct name must be entered via the serial interface. Then, the user must pass a biometric check by pressing the blue button (simulating NFC), followed by the green button (simulating fingerprint scanning).

If successful, the system activates the green indicator and smoothly opens the lock via the servo drive. Access is granted temporarily, and after a set interval, the lock closes automatically. Any error activates the red LED, signaling that the request has been rejected. The system is additionally equipped with a PIR sensor for motion detection and a potentiometer for battery-level simulation. The connection diagram is shown in Figure 2A; the board pinout is shown in Figure 2B.

Arduino Uno was selected to represent an autonomous offline configuration. The absence of built-in network modules makes this platform suitable for isolated environments without Internet connectivity. Its simplicity and low cost make it a widely used solution for educational purposes and introductory automation projects. The implemented configuration uses standard libraries – Servo.h for lock control and LiquidCrystal.h for LCD output – and performs all basic access control functions in offline mode. Internet connectivity would require an external module such as ESP8266/ESP-01, which increases system complexity and power consumption and was therefore excluded from this configuration.

The connection diagram of components to the Arduino Uno microcontroller is shown in Figure 3A. The pinout diagram of the Arduino Uno board with the designation of each pin's function is shown in Figure 3B.

The Arduino Uno configuration implements three key functions: multi-level identification, servo-based lock control, and PIR motion

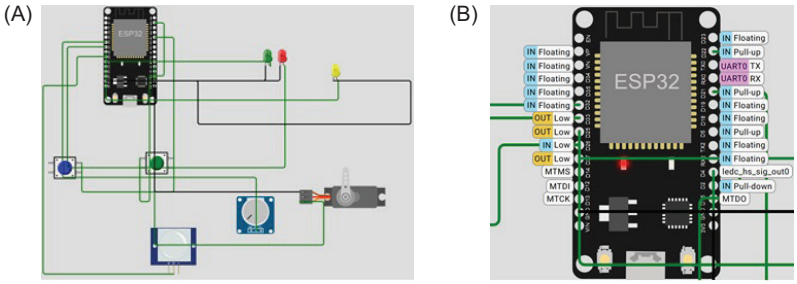

Figure 2. Project diagram on ESP32.

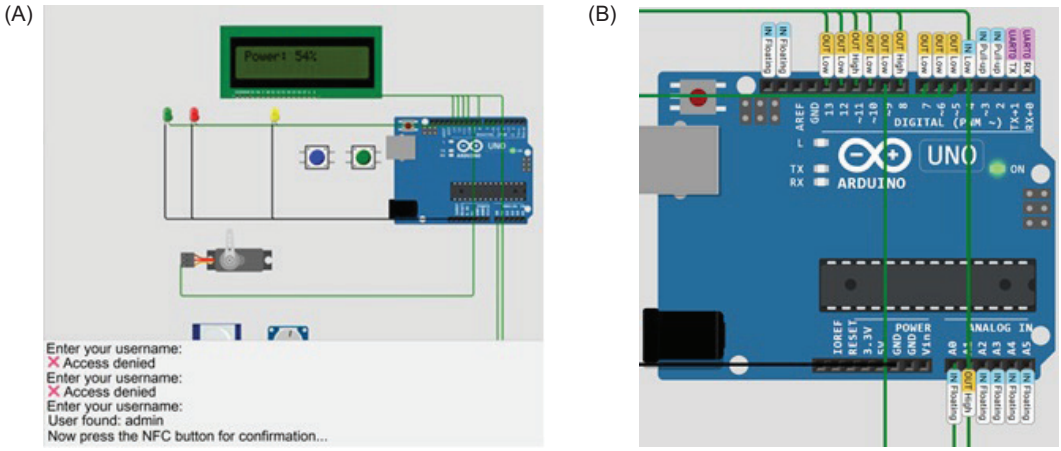

Figure 3. Project diagram on Arduino Uno.

monitoring. System initialisation is confirmed by the message «System is ready» on the LCD display. A background process continuously monitors the potentiometer position, displaying a percentage charge indicator, such as «Power: 54%». When motion is detected, the system displays «ALARM! Movement» on the LCD, activates the yellow LED, and logs the event to the serial monitor. The connection diagram is shown in Figure 3A; the board pinout is shown in Figure 3B.

Raspberry Pi Pico W was integrated as a platform demonstrating the advantages of MicroPython-based development. Its built-in Wi-Fi module provides network functionality comparable to ESP32, including Telegram Bot API integration. A distinctive feature of this configuration is the development of a full web application alongside the microcontroller firmware, providing a browser-based interface for event history review, security settings configuration, and real-time status monitoring. The platform's low power consumption and competitive cost make it particularly relevant for energy-constrained deployments.

The connection diagram of components to the Raspberry Pi Pico W microcontroller is shown in Figure 4A. The pinout diagram of the Raspberry Pi Pico W board with the designation of the functions of each pin is shown in Figure 4B.

The Raspberry Pi Pico W configuration simulates access verification using two buttons representing positive and negative authentication outcomes. A PIR sensor detects movement in the controlled area. System status is visualised by a three-colour LED indicator:

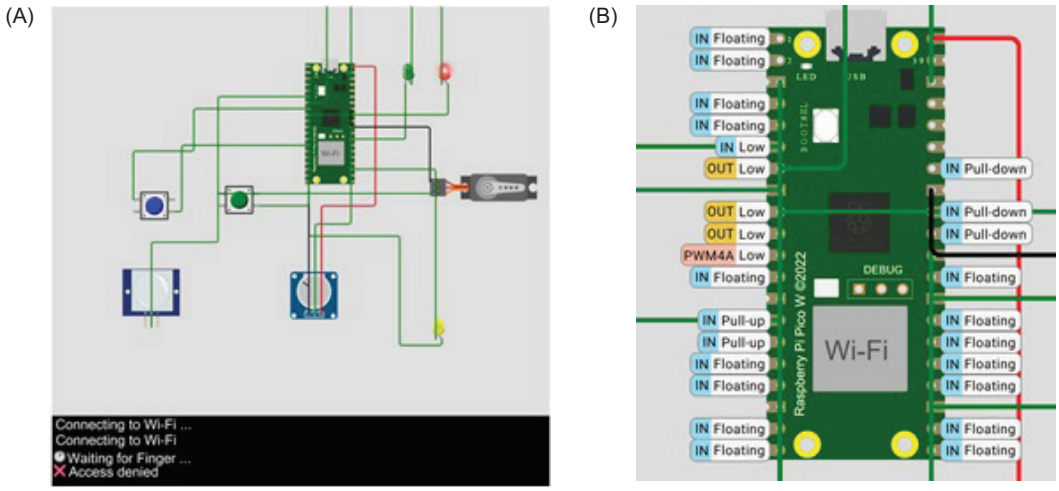

Figure 4. Project diagram on Raspberry Pi Pico W.

green confirms access, red signals rejection, and yellow warns of motion detection. The servo drive simulates mechanical lock operation upon successful authorisation. The connection diagram is shown in Figure 4A; the board pinout is shown in Figure 4B.

A structured comparison of the three platforms across key criteria is presented in Table 1.

4.4. Experimental Protocol

To ensure comparability of results across all three platforms, a unified experimental protocol was established and applied consistently throughout the study.

Each platform underwent 10 repeated authentication cycles under identical simulated conditions in the Wokwi environment. A single authentication cycle consisted of the following sequential steps: system initialisation, authentication trigger input (simulated via push button), two-factor verification logic execution, servo drive activation upon successful authentication, and transmission of the access decision to the server or Telegram Bot API, where supported. The authentication logic is illustrated in Figure 5.

The authentication algorithm begins with system initialisation, followed by NFC card validation and biometric verification. A maximum of three failed attempts are permitted before the alarm mode is activated. Upon successful two-factor verification,

Table 1. Comparison of platform capabilities.

Criterion	Arduino Uno	ESP32 DevKit V1	Raspberry Pi Pico W
Development complexity	Low (basic system without network)	Medium/high (working with network libraries)	Low/medium (working with MicroPython and the network)
Internet connectivity	External module only	Built-in Wi-Fi and Bluetooth	Built-in Wi-Fi
Programming language	C++ (Arduino IDE)	C++ (Arduino IDE)	MicroPython
Energy consumption	Lowest (in offline mode)	Medium/high (with Wi-Fi enabled)	Low
Telegram Bot API support	–	+ (via Wi-Fi, HTTPS requests)	+ (via Wi-Fi, MicroPython requests)
Multithreading support	No	Yes (FreeRTOS)	Yes (dual-core)
Advanced network scenarios	–	+	+

Note: Power consumption for Arduino Uno is reported for offline mode without an external Wi-Fi module. With an ESP8266/ESP-01 module, total consumption increases to approximately 195–295 mW depending on operating conditions.

the system unlocks the access device, logs the event, and returns to standby mode.

The following performance indicators were measured for each platform:

Average system response time – the time interval from authentication trigger to result display, measured by recording the software cycle execution timestamps within the simulator.

Average data transfer latency – the delay between controller-side decision and server-side registration, applicable to ESP32 and Raspberry Pi Pico W configurations with active Wi-Fi.

Message delivery reliability – the percentage of successfully delivered messages out of total transmission attempts, evaluated over 1000 packets per platform.

Average power consumption during the authentication cycle – estimated analytically based on technical datasheets of each microcontroller and its peripheral components under typical operating conditions, as direct power measurement is not supported in the Wokwi environment.

FAR and FRR identification metrics – calculated analytically based on known characteristics of equivalent biometric and NFC solutions for each platform, as full sensor emulation is not available in Wokwi.

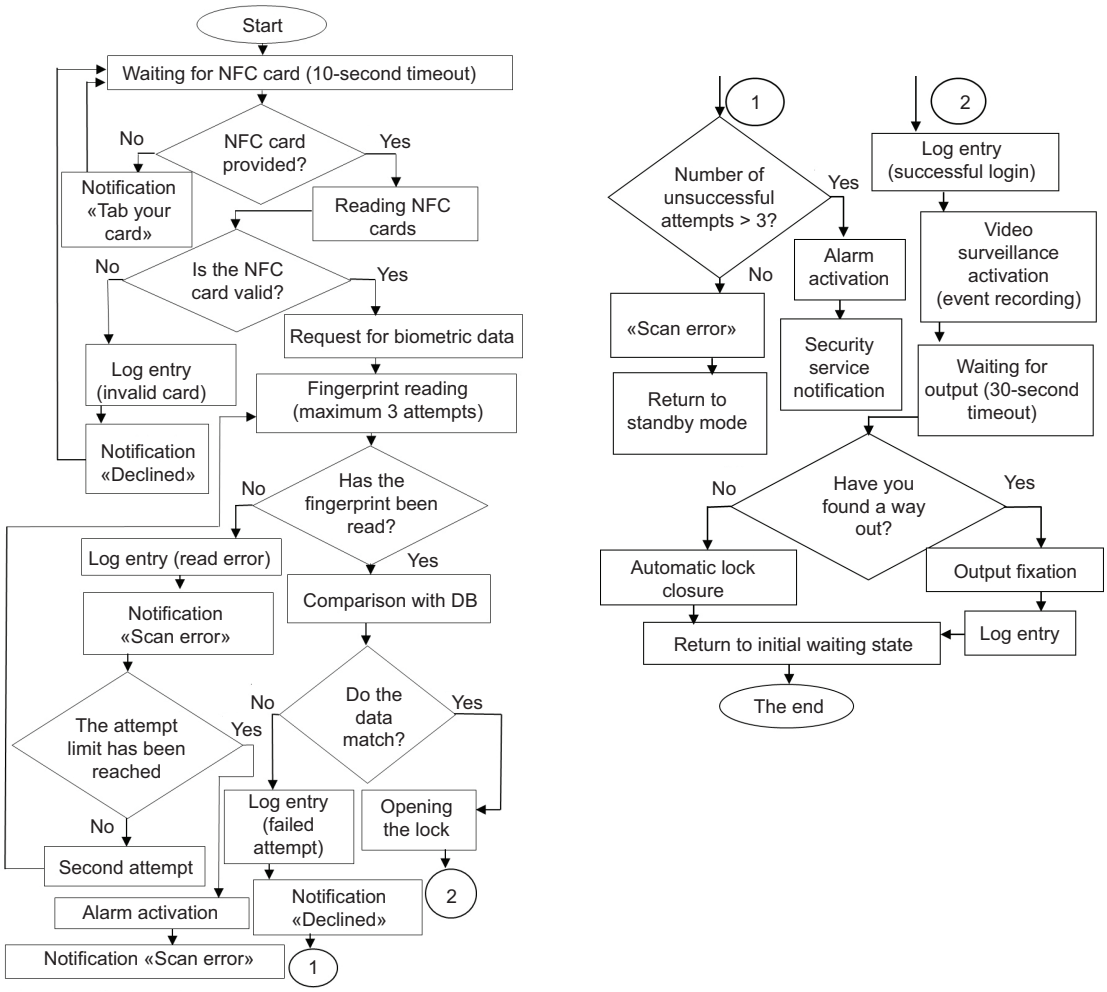

Figure 5. Algorithm of the authentication subsystem (NFC and biometrics).

The sample size of $n = 10$ per platform was constrained by the manual nature of the experimental protocol, in which each authentication cycle required sequential button inputs to simulate NFC and fingerprint events. Automating this input sequence within the simulation environment was beyond the scope of the current study. While this sample size is sufficient for detecting the large inter-platform differences observed (Cohen's $d > 3.5$ for all pairwise comparisons), it limits precision for within-platform variability characterisation. Larger sample sizes ($n \geq 30$) and automated test execution are planned for the physical prototyping stage.

All response time and latency values are reported as mean ± *SD* across 10 test cycles. FAR and FRR values are predictive in nature

and are intended for comparative platform assessment only. Final empirical validation of identification metrics is planned at the physical prototyping stage using real biometric sensors and RFID/NFC hardware.

Wi-Fi connection stability was evaluated separately by monitoring connection interruptions, packet loss, and recovery time over a dedicated test session of 1000 transmission attempts per platform. Arduino Uno, which does not support native Wi-Fi, was assessed under offline conditions only, with network-dependent metrics marked as not applicable.

5. Experiments

Experimental testing was conducted across all three platform implementations – ESP32, Raspberry Pi Pico W, and Arduino Uno – following the unified protocol described in Section 4.4. The experiments aimed to verify authentication logic, remote notification functionality, and system behaviour under both normal and error conditions.

Upon successful system initialisation, ESP32 automatically transmitted a startup confirmation to the designated Telegram chat, including the exact date and time of activation (Figure 6). This confirmed correct network initialisation and device readiness for operation.

Access denial scenarios were tested by entering incorrect credentials or performing button presses in wrong sequence. In all such cases, the system activated the red LED and transmitted an access denied notification via Telegram (Figure 7).

Energy monitoring functionality was verified through the analog potentiometer simulation. When the potentiometer reached its maximum position, the system generated a critical battery level alert and transmitted it to the Telegram chat with a timestamp (Figure 8).

Motion detection was tested on the Raspberry Pi Pico W configuration. Upon PIR sensor activation, the system turned on the

Figure 6. Notification via Telegram messenger.

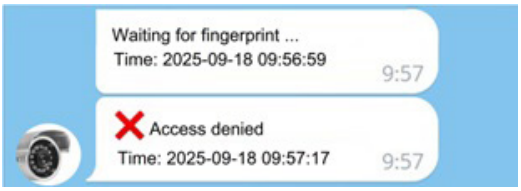

Figure 7. Access denied message.

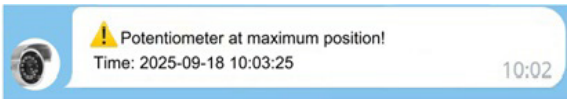

Figure 8. Telegram message about the potentiometer position.

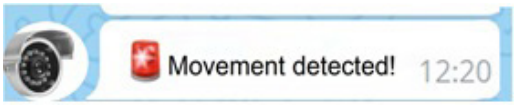

Figure 9. Movement detection notification.

yellow LED for local indication and simultaneously transmitted a "Movement detected!" warning to the administrator via Telegram (Figure 9).

6. Results

The results of the experiment are shown in Table 2.

All response time and latency values are reported as mean ± *SD*, with 95% confidence intervals (95% CI) calculated using the *t*-distribution for $n = 10$ measurements per platform ($t = 2.262$).

Arduino Uno requires an external Wi-Fi module (ESP8266/ESP-01), which increases complexity and delays.

All response time and latency values are reported as mean ± *SD*, with 95% CI calculated using the *t*-distribution for $n = 10$ measurements per platform ($t = 2.262$). The low coefficient of variation (CV) for response time confirms statistical stability: ESP32 (CV = 4.2%), Raspberry Pi Pico W (CV = 5.3%), and Arduino Uno (CV = 5.3%). Based on the assumption of normal distribution, percentile estimates for response time are: ESP32 (p50 = 287 ms, p90 = 302 ms, and p95 = 307 ms), Raspberry Pi Pico W (p50 = 342 ms, p90 = 365 ms, and p95 =

Table 2. Results of the experiment for three platforms.

Indicator/platform	ESP32	Raspberry Pi Pico W	Arduino Uno
Average system response time (t_o), milliseconds (ms)	287 ± 12 [278.5; 295.5]	342 ± 18 [329.3; 354.7]	856 ± 45 [824.2; 887.8]
Average data transfer delay (latency), ms	45 ± 8 [39.3; 50.7]	62 ± 11 [54.2; 69.8]	124 ± 23 [107.7; 140.3]
Network message delivery reliability, %	98.7	97.3	94.2
Average power consumption during cycle, mW	240	180	95
Wi-Fi connection stability	High	Average	Not supported*
Multithreading support	Yes (FreeRTOS)	Yes (dual-core)	No

Note: Arduino Uno's power consumption is shown without taking into account the consumption of external Wi-Fi module (ESP8266/ESP-01).

372 ms), and Arduino Uno (p50 = 856 ms, p90 = 914 ms, and p95 = 930 ms).

To confirm the statistical significance of the observed inter-platform differences, a one-way analysis of variance (ANOVA) was conducted for the average system response time. The results confirm a statistically significant effect of platform type on response time [$F(2, 27) = 1113.8$; $p < 0.001$]. Pairwise comparisons using Tukey's Honestly Significant Difference (HSD) test confirmed significant differences between all platform pairs ($p < 0.001$). To quantify the practical significance of the observed differences, effect sizes were calculated using Cohen's d (Table 3).

The comparative results are illustrated in Figure 10: (A) average system response time (t_o), ms; (B) average data transmission delay (latency), ms; (C) message transmission reliability, %; and (D) average power consumption during the cycle, mW.

Analysis of the experimental results showed that:

- ESP32 demonstrates the best average system response time (t_o; 287 ms), which is 19% faster than Raspberry Pi Pico W and three times faster than Arduino Uno;
- ESP32 provides the lowest latency (45 ms), thanks to its built-in Wi-Fi module, which is critical for real-time systems;
- ESP32 achieves a network delivery reliability of 98.7%, which falls within the range commonly considered acceptable for high-reliability IoT security applications under the simulation conditions used. Formal compliance with specific industrial security

Table 3. Statistical significance and effect size of inter-platform differences (system response time).

Comparison of platforms	Difference of means (ms)	Combined *SD* (ms)	Cohen's *d*	Interpretation	p-value (Tukey HSD)
ESP32 and Raspberry Pi Pico W	55	15.3	3.59	Big effect	<0.001
ESP32 and Arduino Uno	569	32.9	17.30	Big effect	<0.001
Raspberry Pi Pico W and Arduino Uno	514	34.3	15.01	Big effect	<0.001

Note: Cohen's *d* was calculated using the formula $d = (M_1 - M_2)/SD_{pooled}$ where M_1 and M_2 are the arithmetic means of the two samples, and $SD_{pooled} = \sqrt{\frac{SD_1^2 + SD_2^2}{2}}$ is the pooled *SD*. Interpretation thresholds for Cohen's *d* were as follows: $d = 0.2$ – small effect; $d = 0.5$ – medium; $d = 0.8$– large; $d > 1.2$ – very large effect. Retrospective power analysis confirms ($1 - \beta > 0.99$ at $\alpha = 0.0.05$) the representativeness of the sample ($n = 10$ for each scenario) for detecting the observed effect sizes.

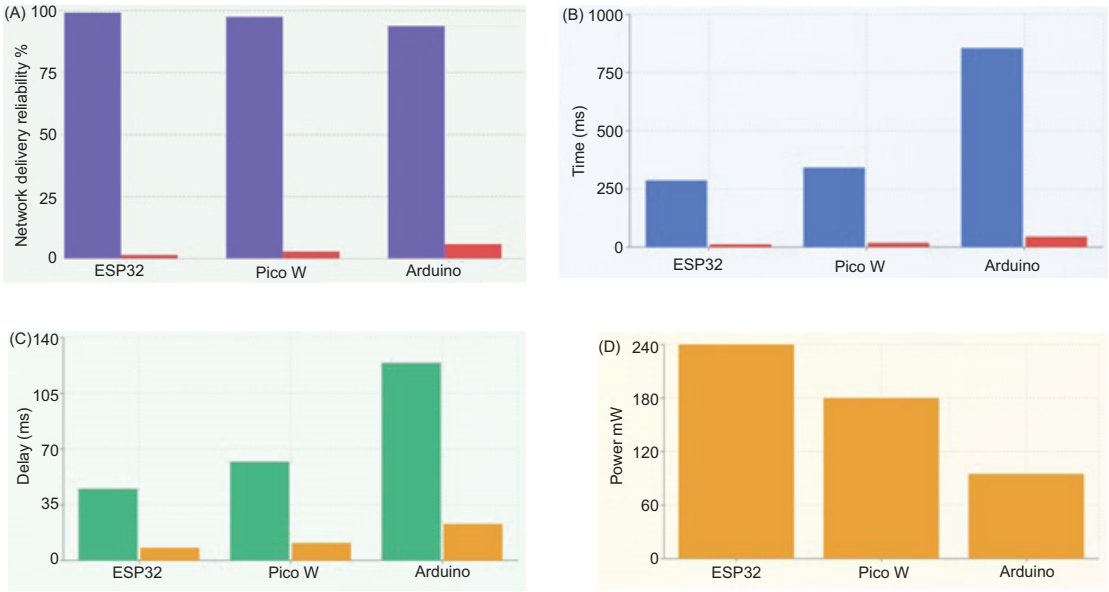

Figure 10. Diagrams showing the results of the experimental study of ESP32, Raspberry Pi Pico W, and Arduino Uno.

standards would require certified testing with physical hardware. Arduino Uno, with a network delivery reliability of 94.2%, exhibits limitations that may make it less suitable for critical applications requiring high communication reliability;

- Raspberry Pi Pico W is the most energy-efficient platform (180 mW), but for a system with UPS and a generator, the 60 mW difference between ESP32 and Raspberry Pi Pico W is not critical, compared to the advantages of ESP32 in terms of performance.

- ESP32 demonstrates relatively balanced performance across the evaluated criteria and, based on the results of this study, can be considered a promising option for industrial access control systems. The results of the Wi-Fi connection stability test are presented in Table 4.

Owing to Wokwi's inability to emulate biometric sensors and NFC modules, FAR and FRR were calculated analytically. The analytically estimated values are presented in Table 5.

The overall system reliability (p_{rl}) presented in Table 5 is calculated as the product of two independent components: the biometric subsystem reliability (p_{bio}) and the network message delivery reliability (p_{net}, corresponding to the 'Network message delivery reliability' row in Table 2), according to the formula $p_{rl} = p_{bio} \times p_{net}$. This approach accounts for two independent failure modes: sensor-level identification errors (quantified by FAR/FRR) and network-level transmission failures.

For the ESP32 and Raspberry Pi Pico W configurations, the biometric subsystem reliability is based on the same optical fingerprint sensor model (R503). For the ESP32, $P_{bio} = 1 - (0.006 + 0.008) = 0.986$, giving $P_{rl} = 0.986 \times 0.987 = 0.973$. For the Raspberry Pi Pico W, $P_{bio} = 1 - (0.009 + 0.019) = 0.972$, giving $P_{rl} = 0.972 \times 0.973 = 0.946$ (94.6%).

For Arduino Uno, the lack of native biometric support necessitates an external module connected via UART, which introduces higher

Table 4. Results of the experiment to test the stability of the Wi-Fi connection.

Platform	Connection breaks	Recovery time	Lost packages
ESP32	0	–	13 out of 1000
Raspberry Pi Pico W	2	3.2 s	27 out of 1000
Arduino Uno	8	12.5 s	58 out of 1000

Table 5. Analytically estimated identification reliability.

Platform	FAR (%)	FRR (%)	p_{rl} (%)
ESP32	0.6	0.8	97.3
Raspberry Pi Pico W	0.9	1.9	94.6
Arduino Uno	2.2	3.7	88.6

error rates (FAR 2.2% + FRR 3.7%, giving P_{bio} = 1 – (0.006 + 0.008) = 0.986. Consequently, its overall reliability is P_{rl} = 0.941 × 0.942 = 0.886 (88.6%). The values in Table 5, therefore, differ from the network-only reliability figures in Table 2, as they reflect the compounded effect of both subsystem failure modes.

The FAR and FRR estimates reflect typical error ranges for optical fingerprint sensors operating under realistic conditions (e.g. wet or dirty fingers, partial contact) rather than ideal laboratory specifications. While manufacturer datasheets for specific modules, such as the R503 optical fingerprint sensor report FAR ≤ 0.001% and FRR ≤ 1% under optimal conditions [18], independent studies of optical fingerprint sensors in real-world deployment suggest more conservative rates of FAR 0.5–2% and FRR 0.5–3% due to environmental factors and skin conditions [19]. NFC/RFID modules are less susceptible to environmental factors and were assigned FAR < 0.1% accordingly. These conservative estimates were deliberately chosen to reflect practical deployment conditions, rather than best-case sensor performance.

The data is predictive in nature and is used for comparative assessment of platform potential.

Final experimental verification of identification indicators is planned at the physical prototyping stage using real biometric sensors and RFID/NFC cards.

7. Discussion

The results demonstrate clear performance differentiation across the three platforms. Among the evaluated platforms, ESP32 exhibits the most favourable performance characteristics under simulated conditions, suggesting its potential suitability for access control applications where high reliability and low latency are critical. The discussion covers technical aspects, practical applicability, and economic feasibility of each platform.

ESP32, with an average response time of 287 ± 12 ms and a *network delivery reliability of 98.7%*, provides the best results, which can be explained by its advanced architecture. The dual-core Xtensa LX6 processor with a clock speed of 240 MHz allows biometric data to be processed in parallel while performing network communication without reducing performance. Built-in support for wireless protocols eliminates the need for additional communication via UART or SPI, significantly reducing the overall system latency.

The presence of a hardware cryptographic coprocessor further accelerates authentication processing. The relatively low *SD* (±12 ms) suggests greater measurement consistency, likely attributable to the deterministic execution of the FreeRTOS scheduler, compared to the single-threaded Arduino runtime. These findings align with PBV et al. [15], who demonstrated the ESP32-based home security with facial recognition, though their study lacked systematic timing analysis – a gap the present work addresses by providing quantitative response time metrics under controlled simulation conditions.

Raspberry Pi Pico W demonstrated the highest energy efficiency at 180 mW, making it particularly suitable for autonomous, battery-powered deployments. This finding aligns with the work of Gocan et al. [16], who demonstrated secure wireless sensor network implementation using the Raspberry Pi Pico W platform. In terms of response time, Pico W shows satisfactory performance at 342 ± 18 ms for systems where latency up to 350 ms is acceptable. However, its network delivery reliability of 97.3% represents a trade-off that must be considered. While these values are acceptable for many IoT monitoring applications, they fall short of the stringent requirements for real-time industrial access control, where low latency and high reliability are critical [1, 9]. The higher latency observed for Pico W (62 ms) compared to ESP32 (45 ms) can be attributed to architectural factors: the lack of hardware encryption and the use of an external CYW43439 Wi-Fi chip connected via SPI interface, which introduces additional communication overhead compared to the ESP32's fully integrated wireless architecture. This positions Pico W as an excellent choice for energy-sensitive applications, rather than time-critical industrial systems.

Arduino Uno, with a response time of 856 ± 45 ms and network delivery reliability of 94.2%, demonstrates significant limitations for real-time industrial access control applications. These results are consistent with the architectural constraints of the platform: the 8-bit ATmega328P processor running at 16 MHz proves inefficient for processing complex biometric data, while the limited 2-kB RAM leads to data fragmentation and increased processing cycles. The need for an external ESP8266/ESP-01 Wi-Fi module further compounds these issues by introducing an additional communication layer through a slow UART interface. Although the platform remains valuable for educational purposes [8], its network delivery reliability of 94.2% is insufficient for critical security systems. Based on the findings of this study, it can be inferred that network delivery reliability should approach 98–99% to meet the operational

stability requirements typical for industrial security systems – a level achieved by ESP32 (98.7%) but not by Arduino Uno (94.2%). These findings corroborate the earlier work done by Kordov and Bilyal [14], who characterised the Arduino-based access control as suitable only for basic single-factor authentication scenarios. Furthermore, while Arduino Uno shows the lowest nominal power consumption (95 mW) in offline mode, this figure is misleading as it excludes external Wi-Fi module, which increases total consumption to 195–295 mW – exceeding both ESP32 and Raspberry Pi Pico W.

The non-overlapping 95% CI across all three platforms – ESP32 [278.5; 295.5], Raspberry Pi Pico W [329.3; 354.7], and Arduino Uno [824.2; 887.8] – confirm that performance differences are statistically significant and reflect genuine architectural characteristics.

Although the sample size of n = 10 per platform is modest, the statistical analysis confirms its sufficiency for the observed effect magnitudes. One-way ANOVA yielded $F(2, 27) = 1113.8$ (which is significantly higher than the critical value $F_{crit} = 3.35$ at $\alpha = 0.05$), confirming statistically significant inter-platform differences ($p < 0.001$). Cohen's d values exceed 3.5 for all pairwise comparisons, indicating very large effect sizes. A *post hoc* power analysis confirms statistical power $1 - \beta > 0.99$, meaning the probability of a Type II error is negligible. These results validate the use of n = 10 as sufficient for detecting performance differences of this magnitude.

Data transmission latency metrics directly correlate with the architecture of the Wi-Fi subsystem of each platform. The best performance is demonstrated by the ESP32, with a latency of 45 ± 8 ms, due to the presence of an integrated TCP/IP stack and hardware support for IEEE 802.11 b/g/n protocols. Raspberry Pi Pico W, with a latency of 62 ± 11 ms, has higher latency due to the use of SPI interface for communication between the RP2040 processor and the CYW43439 radio module. The highest latency is observed with Arduino Uno (124 ± 23 ms), where data transmission via UART at 115,200 baud (Bd) between the ATmega328P controller and the external ESP-01 Wi-Fi module causes a significant increase in response time. For access control systems, latency below 50 ms is critically important, as this threshold ensures natural user interaction without noticeable delay. While the coefficient of variation for latency is higher (≈18% for all platforms) due to inherent wireless network variability, this does not affect the primary comparison of response time. Future work with larger sample sizes would enable more precise characterisation of latency distribution.

Network message delivery reliability of 98.7% demonstrated by ESP32 is ensured by a stable TCP/IP stack with automatic reconnection functionality and hardware QoS support. ESP32 demonstrated zero connection interruptions during the test period. Raspberry Pi Pico W, with 97.3%, approaches an acceptable level but remains vulnerable to packet loss under network load due to SPI interface bandwidth limitations. Arduino Uno, with a network delivery reliability of 94.2%, demonstrates lower performance under the tested conditions. Consequently, within the scope of this study, it may be considered less suitable for applications requiring high network delivery reliability (e.g. >98%), as commonly expected in industrial contexts.

It should be emphasised that the FAR and FRR values presented in Table 5 are analytical estimates rather than experimental measurements. While these estimates are grounded in published characteristics of comparable hardware components, they serve as comparative indicators of platform potential rather than empirically validated performance metrics. Final confirmation of identification accuracy requires physical prototyping with real biometric sensors.

Energy consumption assessment was carried out analytically based on datasheet specifications. Raspberry Pi Pico W demonstrated the highest energy efficiency at 180 mW. ESP32 consumes around 240 mW, which is 33% more; however, for systems with UPS, this difference is negligible. While Arduino Uno shows nominal consumption of 95 mW in offline mode, this excludes the external Wi-Fi module (80–170 mW), making its actual total consumption 195–295 mW – exceeding other two platforms.

To assess the impact on facility energy consumption, a calculation was made for a system with 10 controllers operating around the clock:

- ESP32: 10 × 0.24 W × 24 h × 365 days = 21.02 kilowatt-hour (kWh);
- Raspberry Pi Pico W: 10 × 0.18 W × 24 h × 365 days = 15.77 kWh; and Arduino + ESP-01: 10 × 0.23 W × 24 h × 365 days = 20.15 kWh. At a rate of 2.64 UAH/kWh (for Ukraine, October 2025):
- ESP32 – 55.49 UAH/year;
- Pico W – 41.63 UAH/year; and Arduino – 53.20 UAH/year.

The savings with Raspberry Pi Pico W are ≈13.9 UAH/year (~0.35 USD), negligible compared to hardware costs but indicating the architectural efficiency of RP2040, which becomes significant in large-scale deployments or battery-powered installations.

The statistical analysis presented is based on simulated measurements obtained in the Wokwi environment rather than physical hardware. Consequently, the reported values should be interpreted as indicative of relative platform differences, rather than absolute real-world performance metrics. The absence of real sensor noise, network interference, and hardware variability means that actual deployment conditions may yield different absolute values, though the relative ranking of platforms is expected to remain consistent.

To further clarify the evidential basis of the results, three categories of evidence are presented in this study. First, directly simulated metrics (response time, latency, and network delivery reliability) were obtained under controlled simulation conditions in the Wokwi environment. Second, analytically derived estimates (energy consumption) calculated from manufacturer datasheet specifications without experimental hardware measurement. Third, analytically calculated projections (FAR, FRR, and biometric subsystem reliability, P_{bio}), derived using the identification reliability model and grounded in published characteristics of comparable hardware components. The overall system reliability P_{rl} combines category 1 and category 3 estimates ($P_{rl} = P_{bio} \times P_{net}$). Conclusions drawn from each category carry different levels of empirical confidence and should be interpreted accordingly.

The claim of suitability for industrial deployment should be interpreted within the context of this study's methodological constraints. While the measured network reliability of 98.7% represents a high level of performance, formal compliance with specific industrial standards (e.g. IEC 62443, ISO/IEC 27001) was not assessed and would require certified testing with physical hardware in operational environments.

Furthermore, the reliance on simulated sensor inputs (push buttons instead of actual biometric and NFC modules) means that real-world factors, such as sensor reading variability, environmental interference, and user interaction patterns, were not captured. This may affect the generalisability of the findings to deployments with different sensor characteristics or operating conditions.

The sample size of 10 measurement cycles per platform was chosen as a practical trade-off between statistical reliability and the manual nature of test execution in the Wokwi simulation environment. While this sample size is sufficient to detect the large performance differences observed between platforms (e.g. 287 ms vs. 856 ms response time), it limits the statistical power for detecting subtle

variations within a single platform. The reported confidence intervals account for this sample size through the use of *t*-distribution. Future work with larger sample sizes ($n \geq 30$) would enable more precise characterisation of performance variability.

Broader applications. Beyond the specific access control scenario examined in this study, the comparative methodology and findings have implications for a wide range of IoT security applications. The demonstrated performance hierarchy – with ESP32 leading in real-time responsiveness, Raspberry Pi Pico W excelling in energy efficiency, and Arduino Uno serving educational roles – can inform platform selection for applications such as smart building automation, industrial monitoring, perimeter security, and remote environmental sensing. In large-scale deployments, the energy efficiency of Pico W becomes particularly significant, potentially reducing maintenance costs and extending system lifetime in battery-powered sensor networks. The modular architecture and Telegram integration can be adapted for other IoT domains requiring remote monitoring and alerting, including healthcare, agriculture, and critical infrastructure protection. The methodology itself – combining simulation-based benchmarking with analytical modelling – provides a template for evaluating emerging hardware platforms in other IoT application domains before physical deployment.

These results contribute to the broader discussion on IoT implementation challenges [5] and the development of modern access control technologies [6], demonstrating the importance of platform selection based on specific application requirements.

8. Conclusions

This study addressed the relevant problem of selecting an optimal microcontroller platform for automated IoT-based access control systems through a systematic comparative analysis of ESP32, Arduino Uno, and Raspberry Pi Pico W under identical simulated conditions.

To the best of the author's knowledge, this is the first systematic comparative evaluation of these three platforms within a unified access control simulation framework, assessing performance, reliability, network stability, and energy efficiency simultaneously. The novelty of this work lies not merely in the comparison itself but in the development and application of a unified multi-criteria evaluation framework that combines simulation-based benchmarking with analytical modelling based on technical datasheets.

This framework provides a replicable methodology for hardware platform selection in IoT security systems before physical prototyping – an integrated approach that, to the best of the author's knowledge, has not been applied previously in this specific IoT access control context.

The following key results were obtained:

1. A systematic comparative analysis of the three platforms was conducted based on unified simulation modelling in the Wokwi environment, ensuring objective assessment under identical operating conditions. Quantitative dependencies between architectural features and operational characteristics were established: ESP32 provides a 19% faster response time compared to Raspberry Pi Pico W, and threefold faster compared to Arduino Uno, which is critical for real-time systems.
2. Mathematical models were developed to evaluate identification reliability, system throughput, energy consumption, and data transmission quality, formalising the design process for access control systems and enabling the prediction of operational characteristics before physical prototyping. Based on the analytical framework presented in Section 2, the ESP32 configuration with the R503 sensor is analytically projected to yield the lowest combined error rate under optimal sensor operating conditions (FAR 0.6% and FRR 0.8%), yielding the highest calculated biometric subsystem reliability (p_{bio} = 0.986) among the evaluated platforms. These estimates are derived from manufacturer datasheet specifications [18] and reflect optimal operating conditions. However, real-world performance may be lower, consistent with performance degradation reported for optical sensors under non-ideal conditions [19], necessitating empirical validation at the physical prototyping stage.
3. A modular IoT system architecture was developed as a unified testing framework for multi-level security. This architecture integrates high-security standards (MIFARE DESFire) with diverse communication protocols (LoRaWAN and GSM), providing a scalable and interoperable foundation for comparative analysis and future industrial IoT deployments.
   Statistical analysis (one-way ANOVA, $F(2, 27) = 1113.8$, $p < 0.001$; Cohen's $d > 3.5$ for all pairs) confirms that these differences are not attributable to random variation, validating the conclusions despite the modest simulation sample size ($n = 10$ per platform).
4. Based on the combined evidence from simulated metrics and analytical estimates, ESP32 demonstrates the most balanced performance profile among the three platforms, with a

response time of 287 ms, a latency of 45 ms, a network delivery reliability of 98.7%, an overall system reliability of 97.3%, and zero Wi-Fi connection drops during testing. These results suggest its suitability for industrial access control applications. However, formal certification against industrial standards, such as IEC 62443, would require additional empirical validation with physical hardware. Raspberry Pi Pico W demonstrated the best energy efficiency at 180 mW (~$0.35 annual savings per controller compared to ESP32), making it attractive for autonomous battery-powered deployments, while Arduino Uno appears more suitable for educational or prototyping applications within the scope of this study, as its network delivery reliability of 94.2% may not fully satisfy typical industrial reliability expectations. These findings quantitatively confirm and extend previous observations: ESP32's superiority aligns with PBV [15], Arduino Uno's limitations corroborate Kordov and Bilyal [14], and Pico W's energy efficiency validates the findings of Gocan et al. [16]. These results confirm the initial hypothesis: ESP32 outperforms competing platforms in response time and network delivery reliability, while Raspberry Pi Pico W demonstrates superior energy efficiency. Unlike these isolated studies, the present work provides the first unified cross-platform comparison within a single access control scenario.

The practical value of the results lies in the development of a methodology for reasoned hardware platform selection for access control systems. Integration of the Telegram Bot API on ESP32 and Raspberry Pi Pico W provides practical value for remote monitoring and timely security event notifications. The proposed architecture is designed to address modern requirements for critical infrastructure security systems [1, 9], while the Telegram Bot API integration extends remote monitoring capabilities discussed in Eboka et al. [6]. Formal compliance with specific industrial standards would require certified testing with physical hardware in operational environments.

Prospects for further research include: (1) creating physical prototypes for empirical validation of FAR and FRR metrics using real biometric sensors and NFC modules; (2) developing machine learning algorithms to detect anomalous access patterns and predict potential threats; (3) exploring blockchain integration for immutable access event logging; (4) optimising LoRaWAN data transmission algorithms for perimeter control to increase range and reduce sensor network power consumption; and (5) investigating the integration of edge artificial intelligence capabilities for real-time threat

detection directly on microcontroller platforms, reducing reliance on cloud connectivity.

The algorithms of the power supply, perimeter protection, and monitoring subsystems will be presented in subsequent publications.

References

[1] S. Sotnik, "Integration of IoT into security systems: opportunities and risks," *International Journal of Academic Engineering Research (IJAER)*, vol. 8, no. 11, pp. 56–61, 2024.

[2] A. Anagnostopoulou, E. Kehrioti, I. Mavridis, D. Gritzalis, "Bolstering IIoT resilience: The synergy of blockchain and CapBAC," in *Proceedings of the 22nd International Conference on Security and Cryptography*, vol. 1, 2025, pp. 120–131, doi: 10.5220/0013513800003979.

[3] M.S. Ahsan, A.S.K. Pathan, "A comprehensive survey on the requirements, applications, and future challenges for access control models in IoT: The state of the art," *IoT*, vol. 6, no. 1, Art. no. 9, 2025, doi: 10.3390/iot6010009.

[4] B. He, T. Feng, C. Liu, C. Su, "Cd-bishac: Cross-domain scheme for blockchain-based industrial Internet of Things security hybrid access control," *IEEE Internet of Things Journal*, vol. 12, no. 6, pp. 7164–7179, 2024, doi: 10.1109/JIOT.2024.3492279.

[5] R. Fikri, A.D. Samala, D. Faiza, "Performance analysis of an ESP32-based smart home system: A laboratory scale study," *Jurnal Vocational Teknik Elektronika dan Informatika*, vol. 13, no. 4, pp. 223–231, 2025, doi: 10.24036/voteteknika.v13i4.136492.

[6] A.O. Eboka, F.O. Aghware, M.D. Okpor, C.C. Odiakaose, E.A. Okpako, A.A. Ojugo, "Pilot study on deploying a wireless sensor-based virtual-key access and lock system for home and industrial frontiers," *International Journal of Informatics and Communication Technology (IJ-ICT)*, vol. 14, no. 1, pp. 287–297, 2025, doi: 10.11591/ijict.v14i1.pp287-297.

[7] U.U. Nordin, N.A.M. Alduais, "Web-based IoT smart door lock system for secure building access management," *Applied Information Technology and Computer Science*, vol. 6, no. 2, pp. 416–432, 2025, doi: 10.30880/aitcs.2025.06.02.023.

[8] V. Shetye, A. Shetty, T. Shetty, K. Samdani, "Intelligent door entry: RFID-based authentication with pin and keystroke profiling," in *International Conference on Computer Science and Communication Engineering (ICCSCE 2025)*. Dordrecht: Atlantis Press, 2025, pp. 1555–1570, doi: 10.2991/978-94-6463-858-5_127.

[9] Y. Vasylchenko, S. Sotnik, "Development of security and fire alarm integrated automation system at enterprise," *WSEAS Transactions on Systems*, vol. 24, pp. 642–664, 2025, doi: 10.37394/23202.2025.24.56.

[10] C. Zhonghua, S.B. Goyal, A.S. Rajawat, "Smart contracts attribute-based access control model for security & privacy of IoT system using blockchain and edge computing," *Journal of Supercomputing*, vol. 80, no. 2, pp. 1396–1425, 2024, doi: 10.1007/s11227-023-05517-4.

[11] J. Aswini, K.S. Rekha, R.A.A. Rosaline, A. Sivaneshkuma, "Enhancing security in cloud computing systems using hybrid feature selection and ensemble-based machine learning for intrusion detection," *Evolving Systems*, vol. 16, no. 101, 2025, doi: 10.1007/s12530-025-09725-6.

[12] N.H. Valentine, E.I. Akaerue, M.G. Etido, C.S. Davies-Ekpo, "A 3-factor authentication access control system using RFID, fingerprint, token and code," *Multimedia Tools and Applications*, vol. 83, no. 15, pp. 43635–43647, 2024, doi: 10.1007/s11042-023-17325-2.

[13] L. Havaš, M. Kraš, J. Srpak, E. Tomičić, "Application of NFC technology in authentication," in *Proceedings EDULEARN24*, 2024, pp. 3910–3917, doi: 10.21125/edulearn.2024.0991.

[14] K.M. Kordov, I.V. Bilyal, "Access control system using Arduino microcontroller and RFID reader," *Annual of Konstantin Preslavsky University of Shumen*, Shumen, Bulgaria, vol. XXIV C, pp. 9–15, 2023, doi: 10.46687/QSFY7970.

[15] R.R. PBV, V.S. Mandapati, S.L. Pilli, P.L. Manojna, T.H. Chandana, V. Hemalatha, "Home security with IoT and ESP32 CAM-AI thinker module," in *2024 International Conference on Cognitive Robotics and Intelligent Systems (ICC-ROBINS)*, pp. 710–714, doi: 10.1109/ICC-ROBINS60238.2024.10533960.

[16] E. Gocan, G. Bucur, C. Popescu, "End-to-end secure wireless sensor network using PKI infrastructure and Raspberry Pi Pico W," *Romanian Journal of Petroleum & Gas Technology*, Ploiești, Romania, vol. 1, pp. 341–358, 2025, doi: 10.51865/JPGT.2025.01.23.

[17] A. Veľas, M. Boroš, R. Kuffa, F. Lenko, "Testing of permeability of RFID access control system for the needs of security management," *Applied Sciences*, vol. 14, no. 10, Art. no. 4227, 2024, doi: 10.3390/app14104227.

[18] Hangzhou Grow Technology Co. Ltd. *R503 fingerprint module user manual*. (2022). [Online]. Available: https://download.mikroe.com/documents/datasheets/R503_datasheet.pdf [Accessed: Mar. 2, 2026].

[19] P. Krishnasamy, S. Belongie, D. Kriegman, "Wet fingerprint recognition: Challenges and opportunities," in *Proceedings of IEEE International Joint Conference on Biometrics (IJCB)*, Osaka, Japan, 2011, pp. 1–7, doi: 10.1109/IJCB.2011.6117594.