Electronics and Information Technologies, 2026, 35, 141–160
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC: 004.056:004.738.5:681.518

AUTOMATED ACCESS CONTROL SYSTEM FOR A SMART HOUSE USING RASPBERRY PI AND AUTOMATIC NUMBER-PLATE RECOGNITION

***Marko Tabachyshyn*[1], *Danylo Tabachyshyn*[2], *Nataliia Kunanets*[2]*, *Andrii Spitzer*[1]**
Lviv Polytechnic National University,
[1] Department of Specialized Computer Systems,
28A Stepana Bandery Street, Lviv, 79013, UKRAINE
[2] Department of Information Systems and Networks,
5 Metropolitan Andrey Street, Lviv, 79016, UKRAINE

Tabachyshyn M., Tabachyshyn D., Kunanets N., & Spitzer A. (2026). Automated Access Control System for a Smart House Using Raspberry Pi and Automatic Number-Plate Recognition. *Electronics and Information Technologies*, 35, 141–160. https://doi.org/10.30970/eli.35.11

ABSTRACT

Introduction. The article is devoted to the development of an access automation system for a smart home based on a Raspberry Pi 4 single-board computer, ESP32 microcontrollers, a Raspberry Pi Camera Module V2, the YOLOv8 computer vision model (for license plate recognition), and Radio-frequency identification (RFID). The relevance of the study is determined by the need to enhance the security level of smart home systems through the implementation of autonomous access control mechanisms. An analysis of existing solutions and approaches in the field of automated access control and smart home systems was conducted.

Materials and Methods. An architecture of a software and hardware access control system is proposed, which integrates license plate recognition based on the YOLOv8 model and the Tesseract OCR system, RFID authentication, and PIN-based access. MQTT protocol is used for data exchange between system components, while monitoring and control are implemented via a Python microframework Flask-based web interface. Algorithms for automatic gate and door opening, event logging, and management of authorized users and vehicle lists were developed. Experimental evaluation of the system was conducted under static and dynamic conditions.

Results. As a result of the study, an access control system for a smart home supporting Automatic number-plate recognition (ANPR), RFID, and PIN authentication technologies was implemented. Experimental evaluation showed a license plate recognition accuracy of 94% in the intended static operating mode and 50% in the supplementary dynamic stress test. The results obtained confirm the good functionality of the proposed architecture and its applicability for automated access control in a smart home.

Conclusions. It has been established that the use of local data processing based on Raspberry Pi ensures system autonomy and reduces dependence on external services. The practical contribution of the work lies in the development of a low-cost, open, and locally operated access control system integrating ANPR, RFID, and PIN-based authentication for private residential use.

Keywords: smart home, access control system, Raspberry Pi 4, YOLOv8, MQTT, ANPR.

© 2026 Marko Tabachyshyn et al. Published by the Ivan Franko National University of Lviv on behalf of Електроніка та інформаційні технології / Electronics and Information Technologies. This is an Open Access article distributed under the terms of the Creative Commons Attribution 4.0 License which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

INTRODUCTION

The rapid development of the Internet of Things (IoT), computer vision, and wireless communication technologies has accelerated the adoption of smart home systems for residential and commercial applications. Modern solutions provide lighting, climate, energy, and appliance control while improving user comfort and resource efficiency. As automation increases, secure and reliable access control becomes a critical requirement.

Most existing smart home systems rely on single-factor authentication, including RFID cards, mobile applications, or PIN-based locks. In addition, many commercial solutions depend on cloud services, introducing risks related to Internet connectivity, personal data transmission, and cybersecurity. Vehicle access automation presents another challenge, as most commercial Automatic Number Plate Recognition (ANPR) systems target corporate or parking applications, making them costly and impractical for private households.

Recent advances in computer vision, particularly YOLO-based models, enable real-time object detection on edge devices, creating new opportunities for autonomous access control. However, integrating ANPR, RFID authentication, and PIN-based access into a unified, locally operating smart home system remains insufficiently explored.

Therefore, this work proposes an autonomous smart home access control system combining computer vision, RFID authentication, and web-based management on cost-effective Raspberry Pi and ESP32 platforms. To justify the proposed approach, existing industrial solutions and recent research on smart home security and automated access control were analyzed.

Ronghua Jiang [1] proposed a Raspberry Pi 4B-based smart home control system integrating motion detection, environmental monitoring, and remote device control through a Flask web interface. Although it uses the same hardware platform as the proposed system, security aspects are only partially addressed.

Study [2] presents a remote keyless access system based on Physical Unclonable Functions (PUFs), providing stronger resistance to key cloning and forgery than RFID or password-based authentication at the cost of increased hardware complexity.

An IoT-based smart home using NodeMCU (ESP8266) and Arduino is described in [3]. Compared with the proposed Raspberry Pi 4 and ESP32 architecture, it offers lower computational performance and focuses primarily on comfort functions rather than security. Similarly, the system presented in [4] supports fire, gas, and motion detection but does not implement access control or vehicle recognition.

Study [5] proposes a two-factor garage access system combining password authentication and license plate recognition. However, the recognition algorithm is not described, while password-based authentication reduces automation and may be impractical when drivers cannot legally use mobile devices.

Commercial smart home platforms, including Google Home [6] and Xiaomi Mi Home [7], provide surveillance cameras, smart locks, and security sensors but rely on cloud services; Google Home additionally requires a paid subscription for video storage.

Industrial ANPR systems automate vehicle access by comparing recognized license plates with predefined allowlists or blocklists. Hikvision solutions integrate cameras, illumination, controllers, and barrier control while supporting event logging and integration through Wiegand and HTTP API [8]. Milesight applies edge AI processing without a dedicated server, supports local allowlists/blocklists of up to 100,000 entries, and integrates through Wiegand, TCP/HTTP, MQTT, and ONVIF, achieving approximately 98% recognition accuracy according to the manufacturer [9].

Ajax Systems can complement third-party ANPR cameras by providing automation, notifications, and scenario execution, enabling coordinated interaction with other security subsystems in residential environments [10]. A detailed comparison of the reviewed solutions is presented in **Table 1**.

Table 1. **Comparative Analysis of Modern Access Control Systems**

Solution	License Plate Recognition	Local Processing	Cloud Dependency	Integration
Hikvision	Yes	Partially	Low	Wiegand, HTTP
Milesight	Yes	Yes	Low	MQTT, ONVIF
Ajax	No	Yes	Low	Relay, scenarios
Proposed system	Yes	Yes	None	MQTT, Web

Industrial solutions represent the current baseline for automated vehicle access control, whereas the proposed system emphasizes autonomous local operation on low-cost hardware.

The analysis of scientific publications and industrial solutions shows that modern smart home systems widely employ IoT, computer vision, and wireless communication for access control. However, most existing solutions either prioritize user comfort or implement isolated security mechanisms, while autonomous systems combining license plate recognition, RFID authentication, and local decision-making on low-cost hardware remain insufficiently explored. This highlights the need for a Raspberry Pi- and ESP32-based solution that operates independently of cloud services while improving security and reducing implementation costs [11].

The aim of this study is to develop and evaluate an autonomous smart home access control system based on Raspberry Pi 4, ESP32 microcontrollers, and computer vision, providing user and vehicle identification, automatic control of actuating mechanisms, and minimal dependence on external cloud services.

To achieve this aim, the following objectives were defined:

- analyze existing commercial and scientific smart home access control solutions;
- justify the selected hardware and software platforms;
- design the Raspberry Pi 4- and ESP32-based system architecture;
- implement license plate recognition using YOLOv8 and OCR;
- implement multi-factor authentication based on RFID, PIN codes, and vehicle recognition;
- develop a web interface for system administration and monitoring;
- experimentally evaluate recognition accuracy and system response time.

The object of the study is the processes of automated access control to smart home facilities.

The subject of the study is the methods and software-hardware means for implementing access control systems based on IoT technologies, computer vision, and multi-factor authentication.

The practical contribution of this work lies in the development and experimental evaluation of a low-cost, open, and locally operated smart home access control system that integrates automatic number plate recognition, RFID, and PIN-based authentication on Raspberry Pi 4 and ESP32 hardware. Rather than introducing a new recognition model or authentication algorithm, the proposed system demonstrates practical architecture for combining existing computer vision and IoT technologies into a unified access control solution suitable for private residential environments. The use of local processing, open-source software, and commercially available low-cost hardware reduces dependence on cloud services and expensive proprietary access control platforms, while providing flexibility for further customization and integration with other smart home subsystems.

MATERIALS AND METHODS

A more detailed consideration of different smart home operating modes related to access control and security is provided. In this work, two main operational modes are defined:

Vehicle entry to the territory: A camera continuously monitors incoming vehicles. The captured images are transmitted to a central server, where a license plate recognition algorithm (YOLOv8) is used to make a decision regarding gate opening. If the license plate is identified as authorized, a control signal is sent to the gate actuator to open the barrier.

Building access: Entry to the building is controlled using a keypad lock and an RFID reader. After entering a PIN code or scanning an RFID card, the information is verified by an API server. If the password or card ID is authorized, the lock is opened.

After analyzing commercial smart home and security systems as well as scientific publications, it was necessary to define a set of microcontrollers, sensors, cameras, and actuators that would distinguish the proposed system in terms of quality, accessibility, computational power, energy efficiency, and interoperability.

For descriptive purposes, the access control process can be represented using a simplified binary classification model that structures the main decision-making stages of the implemented system. Let X denote the input data (license plate image or RFID identifier / access code). The system makes a decision according to the rule:

$$f(X) = \begin{cases} 1, & \text{if } X \in D_allow \\ 0, & \text{if } X \notin D_allow' \end{cases}$$

where D_allow is the set of authorized identifiers (license plates, RFID cards, passwords).

In the case of computer vision usage, the function $f(X)$ is implemented as a composition:

$$f(X) = g[h(X)],$$

where: $h(X)$ is the detection and recognition model (YOLOv8 + OCR); $g[\cdot]$ is the function of comparison with the database.

Among the most commonly used microcontrollers in smart home systems are ESP8266/ESP32, Arduino, Raspberry Pi, and STM32. ESP8266 and ESP32 are among the most popular solutions in the smart home domain due to their low cost and built-in Wi-Fi modules. Their low cost and ability to integrate with various sensors make them a suitable choice for many projects [11].

Raspberry Pi is a single-board computer that provides high performance and the ability to run full operating systems. Raspberry Pi can function as a server for managing the entire smart home system, support multimedia processing, or enable video surveillance. Its high performance and wide range of connectivity options make it indispensable for complex system development [12].

Considering that the proposed system must perform license plate recognition and data analysis, this series of single-board computers was selected as the main processing unit for the smart home access automation system.

The Raspberry Pi Camera Module V2 is an official camera module for Raspberry Pi computers that enables high-resolution video recording and image capture. The module is compatible with all Raspberry Pi models and connects via the CSI (Camera Serial Interface), ensuring fast data transmission. It can be used for image recognition, video and photo processing, and machine learning-based applications [13].

The ESP32 microcontroller is widely used in IoT projects and energy-efficient portable systems. It features low power consumption, high functionality, and built-in Wi-Fi and

Bluetooth support. In addition, ESP32 provides a sufficient number of GPIO pins for connecting sensors and actuators [11].

Motion sensors are commonly used for detecting human presence indoors or outdoors. The HC-SR501 is one of the most widely used motion sensors based on infrared technology (PIR, Passive Infrared). This sensor is passive, meaning it does not emit energy itself but only detects changes in the surrounding infrared radiation [14].

The 4x4 Matrix Keypad [15] is a 16-button matrix keypad. Due to its compact size and ease of use, it is an ideal choice for microcontroller-based projects.

The RFID-RC522 module [16] is widely used in access control systems. It supports a simple communication protocol and can operate with most microcontrollers. The module ensures stable data exchange and low power consumption. Its small size allows integration into space-constrained systems.

The 28BYJ-48 [17] is a popular stepper motor commonly used for mechanisms requiring precise rotation. It provides smooth and accurate movement of shutters or gates. Its advantages include low cost and simple control via the ULN2003 driver [18], as well as reliability for everyday automation tasks.

The SG90 is a widely used miniature servo motor often applied in small mechanical systems. In this project, it is used for locking or unlocking the door mechanism. It offers simple PWM-based control, reliable operation for low-load mechanisms (such as locks), low cost, and low power consumption.

Another important element of the security system is door and window opening sensors. These sensors detect the state of doors and windows, enabling automatic access blocking and sending notifications about unauthorized entry. For this task, the CoVi Security MC-21 Brown [19] sensor was selected due to its reliability, ease of installation, and effectiveness in detecting unauthorized access.

The 2x16 LCD display [20] is a liquid crystal display capable of showing text in two rows of 16 characters each. In this case, it is used together with the 4x4 Matrix Keypad. The display visualizes system status information, Wi-Fi connection status, or MQTT connectivity. Thanks to its compact size and low power consumption, the 2x16 LCD is well suited for smart home integration. When using an I2C adapter, the display requires only two ESP32 GPIO pins, leaving more resources available for other system components. Thus, the selection of hardware components is based on low cost, reliability, ease of integration, low power consumption, and high functionality of each component to ensure efficient operation of the smart home system (**Table 2**).

In the vehicle entry zone, a Raspberry Pi Camera Module V2 is installed, which is responsible for continuous monitoring of vehicles approaching the gate. The camera

Table 2. **Main Components of the Proposed System**

Component	Purpose
Raspberry Pi 4	Central server
Raspberry Pi Camera V2	Image acquisition
ESP32	Local control
RFID RC522	User identification
4×4 Keypad	PIN code input
SG90	Lock control
28BYJ-48	Gate control
HC-SR501	Motion detection
MC-21	Door monitoring / control

transmits images to a central Raspberry Pi 4 server, where vehicle detection is performed using the pretrained YOLOv8 Nano model (`yolov8n.pt`), followed by license plate localization using a separate pretrained YOLO-based detector (`license_plate_detector.pt`). Both models were used without additional training or fine-tuning within this study. The Raspberry Pi 4 then utilizes Tesseract OCR [21] for character recognition on the detected license plate region. If the vehicle is authorized for entry (i.e., its license plate is present in the database), the server sends a command to the ESP32 microcontroller, which controls the gate actuator and opens the gate. This integration enables convenient access control without requiring additional actions from the owner. Additionally, a manual push button is installed on the premises to open and close the gate, enabling vehicle exit or allowing quick access for a vehicle whose license plate is not registered in the database.

Fig. 1 illustrates the structural diagram of the smart home access automation and control system.

Access to the building is controlled using an electronic lock with a keypad input or an RFID reader, both connected to a single ESP32 microcontroller. The user can enter a PIN code or use an RFID card, after which the ESP32 transmits the data to the Raspberry Pi, where verification is performed by comparing the entered code or scanned card against a database of authorized users. If the data matches, the Raspberry Pi sends an unlock signal to the ESP32. A door opening sensor is also installed to monitor whether the entrance door is properly closed. This system provides a high level of security, as data is verified in real time, and unauthorized access attempts can be automatically logged and reported to the owner via the web interface.

The user interacts with the smart home access automation and control system through a convenient web interface accessible from any device. Through this interface, the user can add, modify, or delete license plates for automatic gate opening, as well as update access codes and RFID cards for building entry. The interface also allows viewing the event log, including access attempts, sensor activity, and alarm signals.

The diagram shown in **Fig. 2** illustrates the sequence of interaction between the user and the system via the web interface, an API server (on Raspberry Pi 4), ESP32, and actuators. It demonstrates the main automation scenarios.

Fig. 3 presents a functional diagram of the operation and connections between the Raspberry Pi 4 and the Raspberry Pi Camera Module V2 for the license plate recognition system.

Functional Diagram of the Gate Opening System. The gate opening system consists of an ESP32 microcontroller, an ULN2003 driver, a 28BYJ-48 stepper motor, one push button, and two LEDs (red and blue). The ESP32 receives an open/close command from the Raspberry Pi 4 via Wi-Fi and generates control signals for the stepper motor driver, which rotates the motor by 90 degrees. The LEDs are used to indicate the gate status. **Fig. 4** presents the functional diagram of the operation and interconnections of the gate opening system.

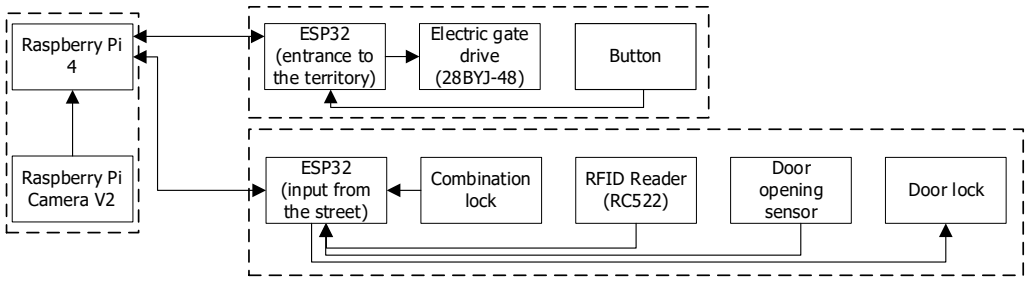


Fig.1. Structural Diagram of the Smart Home Access Automation and Control System

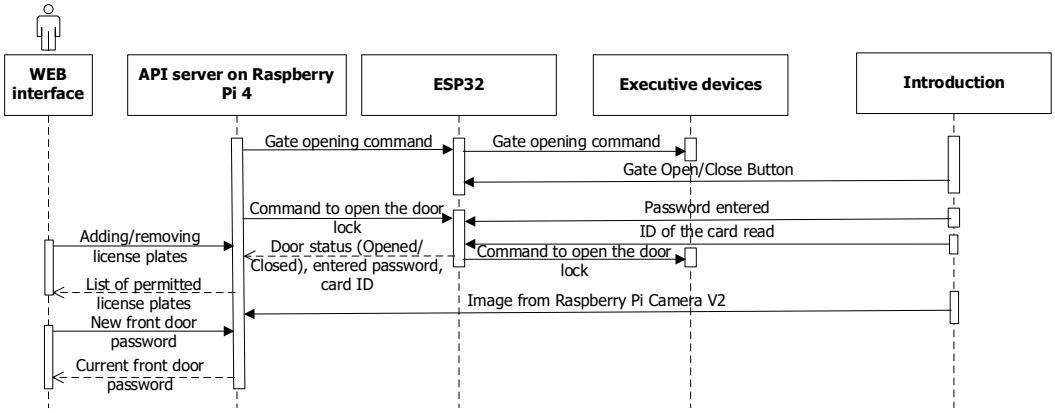


Fig.2. UML sequence diagram of user interaction with the smart home access automation and control system

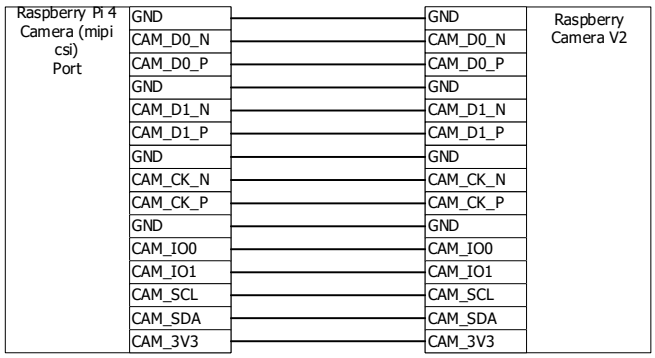


Fig.3. Functional diagram of the connection of the License Plate Recognition System

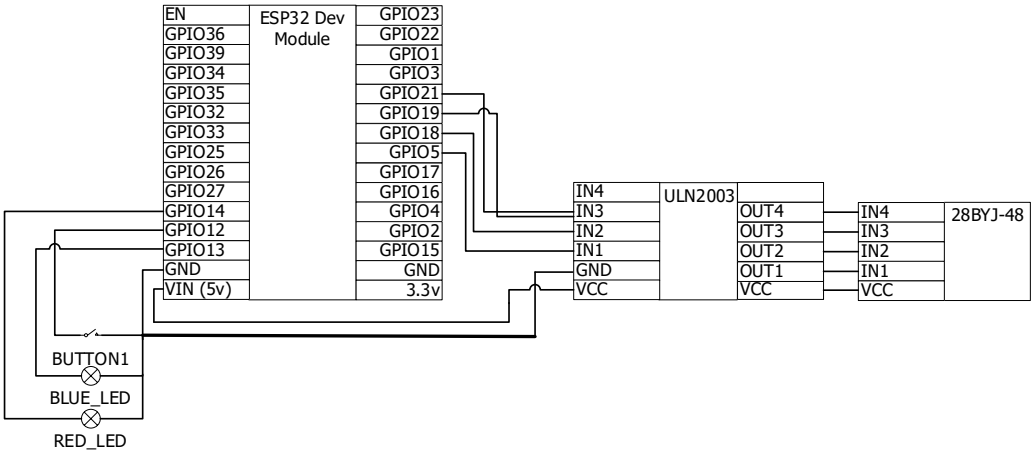


Fig.4. Functional diagram of the connection of the Gate Opening System

Functional Diagram of the Door Opening System. The door opening system consists of an ESP32 microcontroller, a 4×4 matrix keypad, an RFID-RC522 module, a 2×16 LCD display, an SG90 servo motor [22], and a CoVi Security MC-21 sensor. The user enters

a password via the keypad or scans an RFID card; the ESP32 transmits this data to the Raspberry Pi 4, where server-side verification is performed by checking whether the password or card ID exists in the database of authorized users. If the credentials are valid, the Raspberry Pi 4 sends an unlock command back to the ESP32 via Wi-Fi. The ESP32 activates the servo motor, which rotates by 180 degrees to unlock the door. **Fig. 5** presents the functional diagram of the operation and interconnections of the door opening system.

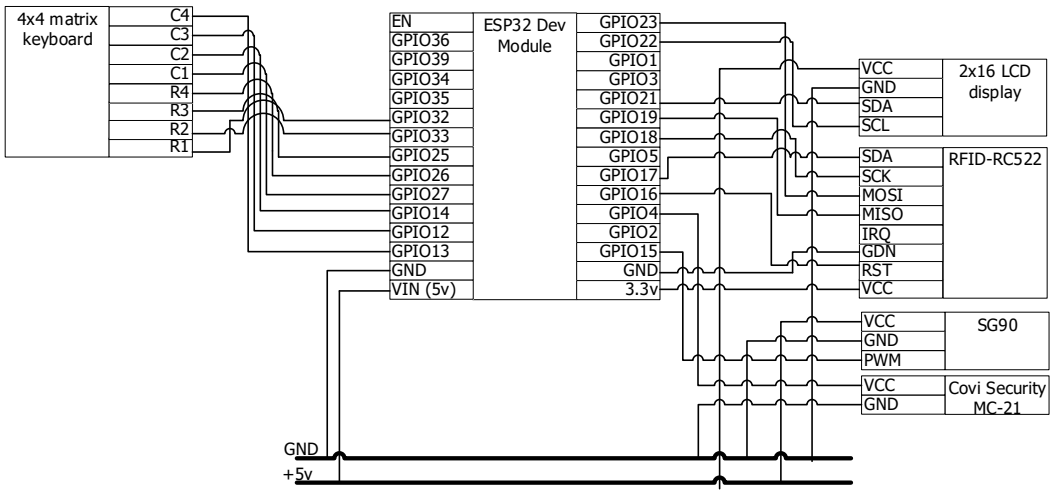


Fig.5. Functional diagram of the connection of the Door Opening System

Let us consider in more detail the algorithms developed to ensure secure access to the smart home, namely the license plate recognition algorithm and the door/gate control system. The license plate recognition program performs image processing, identifies vehicles and their license plates, and verifies whether the detected license plate exists in the database. **Fig. 6** illustrates the algorithm flowchart. The main stages of the program are listed below.

- Image acquisition is performed by capturing a snapshot of the vehicle using the libcamera-still command. The system then checks whether the operation was successful and whether the corresponding image file exists. If the capture is successful, the program proceeds to the next stage; otherwise, the process loop is restarted.
- Vehicle detection is performed using the standard pretrained YOLOv8 Nano model (`yolov8n.pt`), which automatically detects and localizes regions containing vehicles. The model was used with its pretrained weights without additional training or fine-tuning within the scope of this study. If no vehicles are detected, the execution loop is restarted.
- License plate detection is then performed for each detected vehicle using a separate pretrained YOLO-based model (`license_plate_detector.pt`). This model was also used without additional training or fine-tuning on a custom regional license plate dataset within this study. The detected license plate regions are saved as separate images for further processing. The original source and training dataset of the pretrained `license_plate_detector.pt` weights were not documented during prototype development; therefore, no specific claims regarding its training-set composition are made in this work.

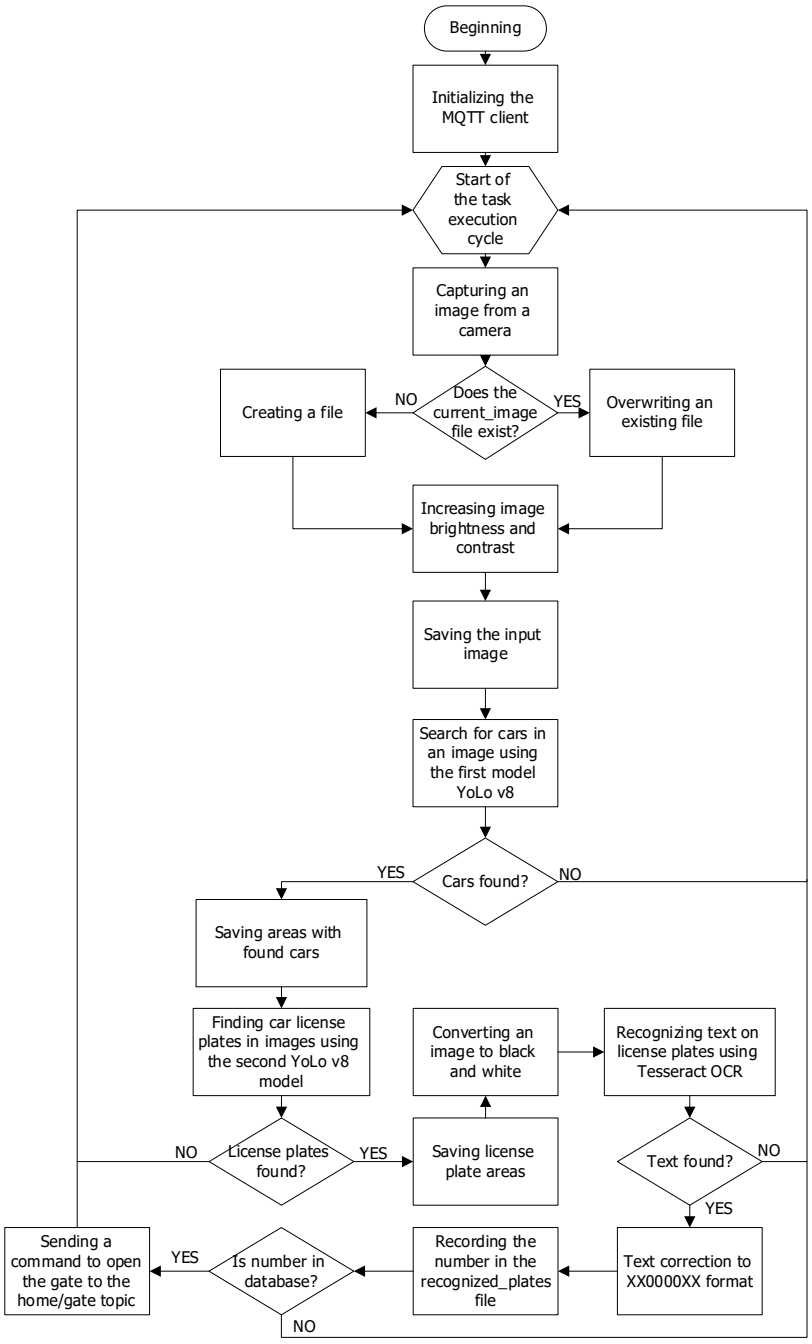


Fig.6. Flowchart of the License Plate Recognition System Algorithm

- License plate recognition is carried out by processing the localized regions using the Tesseract OCR engine, which converts the textual image of the license plate into a character-based representation, enabling extraction of the plate number for verification.
- Text filtering and correction are applied to the recognized text string to remove recognition errors and normalize it into a standardized license plate format. Excess characters are removed, and the resulting text is validated against predefined patterns.

- Database verification is performed by comparing the recognized license plate with a local SQLite database containing a list of authorized vehicles. If the plate is approved, a command to open the gate is transmitted via MQTT.
- Logging is implemented by saving all recognized license plates into a text file together with corresponding timestamps for event logging and further analysis.
- Cyclic execution ensures continuous operation of the system, where image acquisition, vehicle detection, license plate recognition, and verification processes are repeated every 3 seconds.

The program provides automatic gate control using MQTT commands or a physical button, as well as automatic closing after a predefined time interval. It operates in a continuous loop in which incoming commands are processed, and the gate state and its movement are monitored.

One of the key characteristics of the access control system is the response time to the appearance of a vehicle near the gate. The total access time is determined by the cumulative time required for image acquisition, data processing using computer vision techniques, license plate character recognition, database verification, and execution of the control command. To evaluate system performance, the total access time can be expressed as:

$$T_{\{access\}} = T_{\{Capture\}} + T_{\{detect\}} + T_{\{ocr\}} + T_{\{verify\}} + T_{\{actuator\}} \quad (1)$$

where:

$T_{\{Capture\}}$ is image acquisition time from the camera;

$T_{\{detect\}}$ is vehicle and license plate detection time using the YOLOv8 model;

$T_{\{ocr\}}$ is license plate text recognition time using OCR methods;

$T_{\{verify\}}$ is time required to validate the recognized plate in the local database;

$T_{\{actuator\}}$ is time required to transmit the control command and activate the gate mechanism.

This simplified representation is used as an explanatory tool to structure the main components of the system response time and identify the stages that contribute most to the overall delay.

During experimental studies, it was found that the average value of T_{access} is 5–8 seconds. The largest portion of the total processing time is consumed by the license plate detection stage and OCR recognition. The CPU load on the Raspberry Pi 4 during these operations ranged from 60% to 85%.

Fig. 7 illustrates the flowchart of the gate control program.

- Connection to Wi-Fi and the MQTT broker is performed during the initialization stage of the program, before processing any control commands. After establishing the connection, the system subscribes to an MQTT topic where commands for opening or closing the gate are received and processed.
- Command processing is performed for messages received via MQTT or as a result of a button press, after which the current state of the gate (open or closed) is checked to determine the next action.
- Gate opening is executed by calling the openGate() function, which, if the gate is in a closed state, moves it to the open position with a rotation angle of 90°. At the same time, the “open” indicator (blue LED) is activated, and an automatic closing timer is started.
- Gate closing is performed by the closeGate() function, which, after the defined time interval has elapsed, returns the gate from the open state to the initial closed position (0°). The “closed” indicator (red LED) is activated, and the current state is stored in the device memory (EEPROM).

- Idle mode keeps the system in a ready state for receiving control commands via MQTT or button input. If the gate remains open for more than one minute (default setting), it is automatically closed.

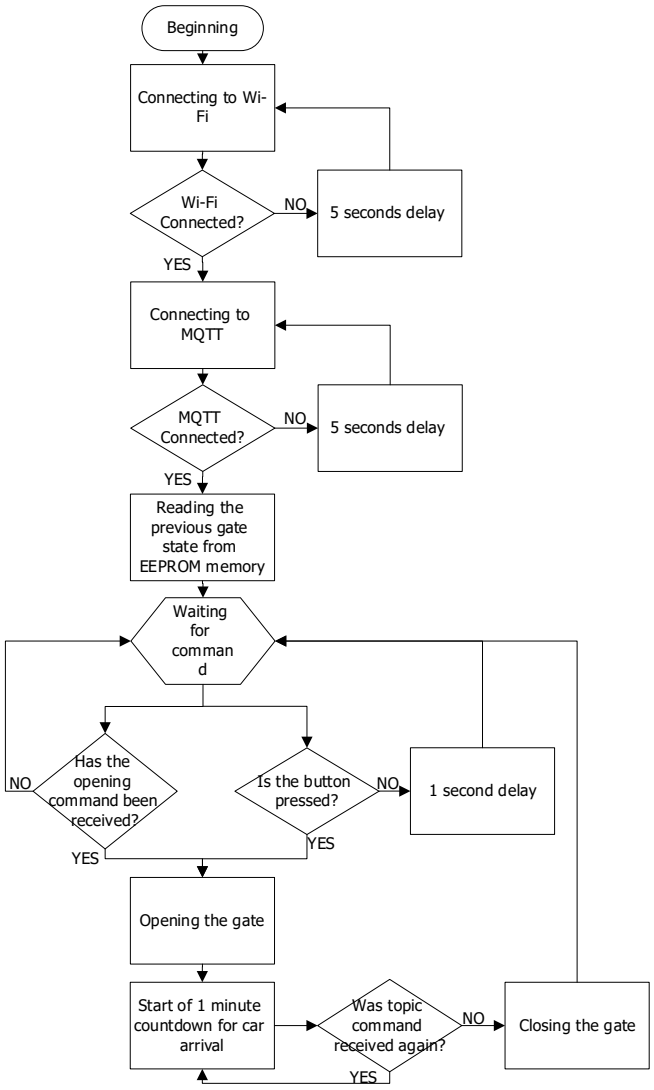


Fig.7. Flowchart of the Gate Opening System Algorithm

Another program provides automatic control of the door lock using a keypad, RFID cards, and MQTT. It also includes status indication on a 2×16 display and supports automatic door locking after a predefined time interval. **Fig. 8** illustrates the flowchart of the door control program algorithm.

- System initialization establishes the Wi-Fi and MQTT connections, tests the servo motor, and displays “System: Ready” on the 2×16 LCD to indicate successful startup.
- Input handling waits for keypad input or RFID card detection and initiates user authentication.
- RFID authentication reads the card identifier, sends it to the server via MQTT, and grants or denies access based on the response.

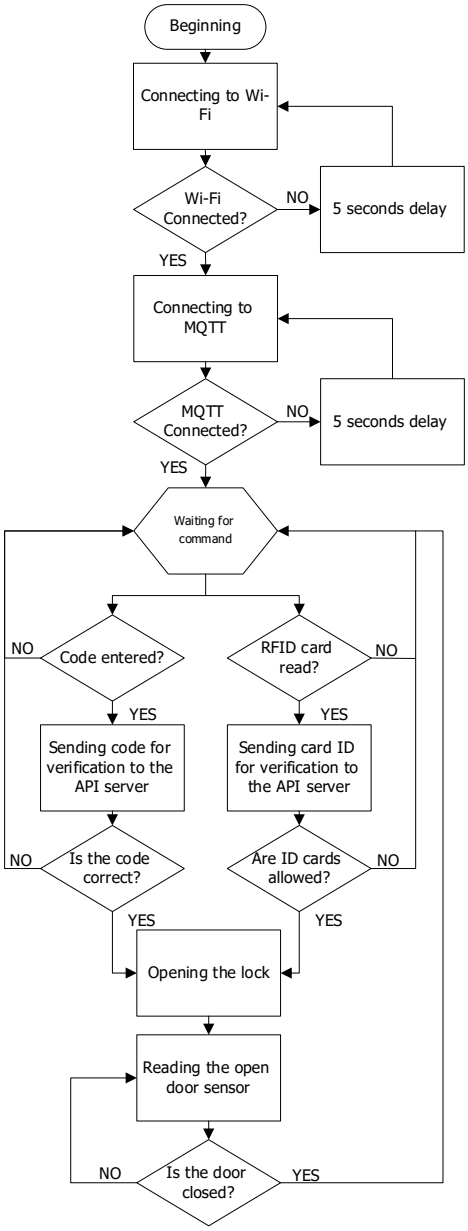


Fig.8. Flowchart of the Door Opening System Algorithm

- PIN authentication sends the entered access code to the server for verification and unlocks the door if valid.
- Door unlocking rotates the servo motor by 180°, displays “Access: Granted”, stores the open state, and starts the automatic closing timer.
- Door locking returns the servo motor to 0° after the preset interval (10 s) or when triggered by the door sensor.
- Status display provides feedback on authentication and access decisions, including “Access: Denied” for invalid PINs or unauthorized RFID cards.
- The main loop continuously processes user input and timer events, ensuring autonomous system operation.

Raspberry Pi OS x64 was selected for its hardware optimization, 64-bit support, and compatibility with Python libraries. Thonny IDE was used to develop Python applications for license plate recognition, OpenCV image processing, the Flask web interface, and MQTT communication. Arduino IDE was used to develop software for the ESP32 distributed control nodes. The license plate recognition subsystem comprises vehicle detection, license plate localization, OCR, and database verification modules.

The software implementation consists of functions for vehicle and license plate detection, OCR-based character recognition, database verification, and RFID/PIN authentication. The `detect_cars_and_plates` function performs vehicle detection using YOLOv8 and invokes license plate localization and OCR processing. The `recognize_plate_text` function preprocesses the detected plate region and applies Tesseract OCR, while the RFID handling routine reads the card identifier and transmits authentication or registration requests to the server via MQTT.

After license plate recognition, the is_plate_allowed function verifies the detected plate against the database. If authorized, an “Open” command is published to the MQTT topic “home/gate”, where the ESP32 controller executes the openGate function. The gate remains open while the vehicle is detected and closes automatically via closeGate.

The ESP32 entrance controller reads the PIN code and RFID card ID using the handleKeypadInput and handleRFID functions and sends them to the API server. The server verifies the data and returns “VALID” or “INVALID”. If authentication succeeds, unlockDoor is executed, while the door sensor invokes lockDoor after the user enters.

The Flask-based API server coordinates communication between system devices and the database via MQTT and HTTP, synchronizing system states and processing incoming messages.

The web interface is a convenient tool for managing smart home systems, providing users with access to system settings and monitoring through a browser. It is implemented using the Flask framework and utilizes HTML templates to create a simple and user-friendly graphical interface. **Fig. 9** shows the appearance of the web interface.



Fig.9. Appearance of the Web Interface

The web interface provides the following functions:

1. Door password update allows users to change the access code, which is stored in the database and applied automatically.
2. Vehicle management enables adding and removing vehicles by specifying the license plate and owner information. The database is updated and synchronized with the access control module.
3. Notifications inform users about completed operations, including successful password updates, vehicle additions, and system errors.

RESULTS AND DISCUSSION

The license plate recognition system was evaluated primarily under static operating conditions, which corresponds to the intended deployment scenario of the proposed residential access control system. In normal operation, a vehicle is expected to stop in front of the gate before license plate recognition and access authorization are performed. An additional dynamic test was conducted to evaluate the behavior of the prototype under more challenging conditions involving a moving vehicle; this scenario was considered a supplementary stress test rather than the intended operating mode of the system.

The experimental dataset consisted of 100 test cases divided equally between static and dynamic conditions, with 50 cases in each group. The static tests were performed with stationary vehicles, relatively stable illumination, and frontal license plate positioning. Images of different vehicles were used, and the recognition results were recorded in the system logs. The dynamic tests involved moving vehicles under variable lighting conditions (glare from oncoming headlights and streetlights, glare from the sun, rain, etc.), at distances of approximately 2–10 m and viewing angles of up to 30°. All 50 license plates used in the dynamic experiment were unique.

Under the intended static operating conditions, 47 of the 50 test cases were recognized correctly, corresponding to a recognition accuracy of 94%. The approximate 95% Wilson confidence interval for this result is 83.8–97.9%. In the supplementary dynamic test, 25 of the 50 unique license plates were recognized correctly, corresponding to an accuracy of 50% with an approximate 95% Wilson confidence interval of 36.6–63.4%. Across all 100 test cases, 72 license plates were recognized correctly, resulting in an overall accuracy of 72% with an approximate 95% Wilson confidence interval of 62.5–79.9%.

The lower accuracy observed in the dynamic experiment was mainly associated with vehicle motion, variable viewing angles, insufficient nighttime illumination, and degraded license plate visibility. Since the proposed residential access control system is intended to perform recognition after the vehicle has stopped in front of the gate, the static result is more representative of its intended practical operating conditions. The dynamic experiment was included as an additional stress test to characterize the limitations of the prototype outside its primary operating scenario.

The proposed system was evaluated using a Raspberry Pi 4, ESP32, Raspberry Pi Camera Module V2, RC522 RFID reader, SG90 servo motor, and 28BYJ-48 stepper motor. YOLOv8 was used for vehicle and license plate detection, while Tesseract OCR performed character recognition. The evaluation covered recognition accuracy, response time, and automatic gate and door control.

System response time comprised image acquisition, data processing, database verification, and control action execution. The response-time evaluation was performed using the same 100 test cases employed for the license plate recognition accuracy assessment, including 50 static and 50 dynamic cases. For each test case, the response time was measured from image acquisition to execution of the corresponding access control action. The static subset represents the intended operating mode of the proposed system, whereas the dynamic subset was included as a supplementary stress test. Experimental results are presented in **Table 3**.

Table 3. **The results of license plate recognition**

Operating mode	Number of tests	Successful recognitions	Accuracy
Static mode	50	47	94%
Dynamic mode	50	25	50%
Total	100	72	72%

The results in **Table 3** should be interpreted with respect to the intended operating scenario of the system. The proposed access control architecture is designed for static license plate recognition, where a vehicle stops at the gate before authorization. Therefore, the 94% accuracy obtained in the static tests is the primary performance indicator for the intended application. The combined accuracy of 72% includes the supplementary dynamic stress test and should not be interpreted as the expected recognition accuracy during normal system operation.

The obtained results demonstrate that the system achieves high recognition accuracy under static conditions when the vehicle is stationary. The reduction in accuracy in the dynamic mode is explained by image blur, changes in the viewing angle, and limited computational resources of the Raspberry Pi during real-time video processing.

The next step was the evaluation of the system's timing characteristics. The measured execution times of individual processing stages are presented in **Table 4**.

The analysis of the timing characteristics presented in **Table 4** indicates that the main factor affecting system performance is image processing using computer vision methods. The combined time of license plate detection and OCR recognition accounts for more than 70% of the total access time. In contrast, the stages of database verification and actuator control introduce only minor delays. This confirms the effectiveness of the chosen system architecture and highlights directions for further optimization.

The results also show that the detection and OCR stages dominate the overall processing time, with an average measured access time of approximately 6 s.

It should be noted that the average access time of approximately 6 s represents the processing time from image acquisition to gate actuation. Since image acquisition is performed cyclically every 3 s, an additional waiting time of up to 3 s may occur. Therefore, the total response delay may reach approximately 9 s in the worst case.

CPU utilization, RAM consumption, and processor temperature were monitored during normal operation of the ANPR subsystem while image acquisition, YOLOv8 detection, OCR processing, database verification, and control functions were active. The original prototype tests did not use a predefined long-term monitoring duration or a standardized stress-testing protocol. Therefore, the values presented in **Table 5** characterize the computational load observed during the performed functional tests rather than long-term worst-case resource consumption.

Table 4. **Components of the system response time**

Processing stage	Average time, s
Frame capture	0.5
License plate detection (YOLOv8)	2.8
OCR recognition	1.6
Database verification	0.2
Gate opening	0.9
Total access time	6

Table 5. **Raspberry Pi 4 resource utilization**

Metrics	Value
Average CPU load	63%
Maximum CPU load	85%
RAM usage	1.2 GB
Average CPU temperature	58 °C
Number of active services	6

As shown in **Fig. 10**, the average CPU load was 63%, with a maximum observed value of 85%. RAM usage was 1.2 GB, and the average processor temperature was 58 °C. These measurements characterize resource utilization during the performed functional tests and do not represent long-term worst-case operating conditions.

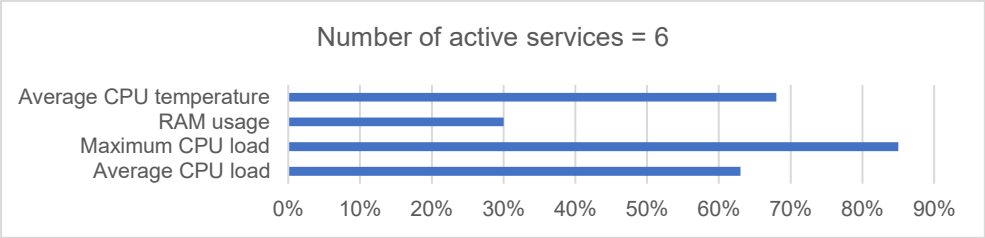


Fig.10. Characteristics of CPU and memory usage of Raspberry Pi 4

RFID identification and PIN-based authentication were also functionally tested during system operation. In the performed test attempts, the system produced the expected authorization and rejection responses without observed false acceptances or false rejections. However, the number of individual RFID and PIN authentication attempts was not recorded separately during the original testing. Therefore, these results should be interpreted as functional verification of the implemented authentication logic rather than as a statistically validated 100% authentication accuracy.

A comparative analysis of the proposed solution and industrial systems is presented in **Table 6**. It should be noted that the recognition accuracy reported by Milesight is based on information provided by the manufacturer [9], whereas the accuracy of the proposed system was obtained experimentally in this study. Therefore, these values should be interpreted with caution because they were obtained under different test conditions and evaluation protocols.

Table 6. **Comparative analysis of the proposed system and industrial analogues**

System	Local processing	ANPR	RFID	Cloud dependency	ANPR recognition accuracy
Hikvision ANPR	Partial	Yes	No	Low	Not specified
Milesight ANPR	Yes	Yes	No	Low	≈98%*
Ajax + third-party ANPR	Partial	Yes	Yes	Medium	Depends on third-party ANPR system
Proposed system	Yes	Yes	Yes	None	94% in intended static mode; 50% in dynamic stress test

Compared with industrial solutions, the proposed system provides local processing, multi-factor authentication, and independence from cloud services using low-cost, commercially available hardware. However, its ANPR performance should also be considered when evaluating the system. Under the intended static operating conditions, the proposed system achieved 94% recognition accuracy, while the supplementary dynamic stress test resulted in 50%. Milesight reports approximately 98% recognition accuracy according to the manufacturer [9]. These accuracy values are not directly comparable because they were obtained using different datasets, operating conditions, and evaluation protocols. Therefore, the main advantage of the proposed system is not superior recognition performance, but its low-cost, open, customizable, and locally operated architecture intended primarily for private residential applications.

The experimental results confirm the functionality of the proposed access control system, which provides autonomous operation and supports multiple authentication methods. The recognition accuracy achieved under the intended static operating conditions demonstrates the applicability of the prototype to residential access control, while the supplementary dynamic tests reveal its limitations under more challenging operating conditions. Overall, the combination of local processing, low-cost hardware, and independence from cloud services makes the proposed architecture suitable for private households and small-scale infrastructure.

The implemented prototype has several security limitations that should be considered when interpreting its suitability for real-world deployment. The RFID subsystem is based on the MFRC522 reader, and card identification is performed using the RFID card UID. This approach provides a simple and low-cost identification mechanism but should not be regarded as cryptographically strong authentication, particularly when MIFARE Classic cards are used. Such cards rely on the legacy Crypto1 mechanism and may be vulnerable to cloning. Therefore, for security-critical deployments, the RFID subsystem should be replaced or complemented with more secure card technologies and stronger authentication mechanisms.

MQTT communication in the implemented system uses broker-level username and password authentication on both the ESP32 nodes and the Raspberry Pi server. However, the current implementation uses an unencrypted MQTT connection over the standard TCP port 1883 and does not employ TLS. Consequently, MQTT credentials and control messages may be exposed to interception or modification by an attacker with access to the local network. A production version should therefore use TLS-protected MQTT communication, certificate-based broker verification, stronger credentials, and topic-level access control.

The RFID registration procedure also represents a potential security weakness. During initial configuration, if no master card has yet been assigned, the first detected RFID card is automatically registered as the master card, and its identifier is transmitted to the server. The server verifies that no master card already exists but does not require an additional administrator credential for the initial registration operation. Therefore, the registration procedure should be protected by an additional authorization step, such as an administrator PIN, an authenticated web interface, or a dedicated physical setup mode.

The gate controller includes partial resilience to power interruptions. Its open and movement states are stored in EEPROM, allowing the controller to restore the previous state and complete an interrupted movement after power is restored. A local control button and automatic gate-closing logic are also implemented. However, the current architecture does not provide a complete fail-safe/fail-secure policy for all components. In particular, the ESP32 nodes use a blocking MQTT reconnection procedure, which may temporarily prevent normal local processing while the broker or network is unavailable. Future versions should implement non-blocking reconnection logic, explicitly defined fail-safe or fail-secure behavior for each actuator, backup power for critical components, and local fallback control that remains available independently of network connectivity.

CONCLUSION

The developed system is an autonomous smart home access control solution based on a Raspberry Pi 4, ESP32 microcontrollers, Raspberry Pi Camera Module V2, and YOLOv8. It integrates automatic license plate recognition, RFID authentication, and PIN-based access control into a unified hardware–software platform, enabling local decision-making without continuous dependence on cloud services.

The proposed architecture combines vehicle recognition, gate and door control, a web administration interface, MQTT communication, and a local database of authorized users and vehicles. Simplified formal representations were used to structure the access control logic and response-time analysis.

Experimental evaluation demonstrated 94% license plate recognition accuracy in static conditions and approximately 50% in dynamic conditions, with an average access time of about 6 s. Hardware resource measurements during the performed functional tests indicated that the prototype operated within the observed CPU and memory utilization limits of the Raspberry Pi 4.

The results confirm the feasibility of the proposed system for residential smart home access control. Its main advantages are autonomous operation, multi-factor authentication, independence from cloud infrastructure, and the use of low-cost, commercially available hardware.

Future work will focus on improving recognition accuracy in dynamic conditions, integrating advanced deep learning models for real-time video processing, implementing biometric authentication, and extending integration with other smart home subsystems.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The authors received no financial support for the research, writing, and/or publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that the research was conducted in the absence of any conflict of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, [M.T.]; methodology, [M.T.]; validation, [M.T., D.T., A.S.]; formal analysis, [M.T.]; investigation, [M.T., D.T.]; resources, [M.T.]; data curation, [A.S.]; writing – original draft preparation, [M.T., D.T.]; writing – review and editing, [D.T., N.K.]; visualization, [N.K.]; supervision, [A.S., N.K.]; project administration, [N.K., A.S.].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Jiang, R. (2025). The smart home control system based on Raspberry Pi. *Journal of Research in Science and Engineering, 7*(9), 21–25. https://doi.org/10.53469/jrse.2025.07(09).04

[2] Ramadhan, M., & Yendri, D. (2025). A door lock security system with remote keyless feature using the PUF approach method. *CHIPSET, 6*, 103–116. https://doi.org/10.25077/chipset.6.02.103-116.2025

[3] Sayeduzzaman, M., Hasan, T., Nasser, A., & Negi, A. (2023). An Internet of Things-integrated home automation with smart security system. https://doi.org/10.1002/9781394213948.ch13

[4] Garba, A., As, I., & Baballe, M. (2024). Design of an ESP32-based IoT smart home automation management system. *Zenodo*. https://doi.org/10.5281/zenodo.13334218

[5] Aramice, G. A. (2017). Smart House Two Level Security System. Journal of Al-Turath University College, 23, 15–25. https://www.researchgate.net/publication/319182193_Smart_House_Two_Level_Security_System
[6] Google. (n.d.). *Google Nest smart home devices*. https://store.google.com/category/connected_home
[7] Xiaomi. (n.d.). *Mi Home IoT privacy white paper*. https://trust.mi.com/docs/iot-privacy-white-paper-global/3/6
[8] Hikvision. (2024). *Vehicle access control management*. https://www.hikvision.com
[9] Milesight. (2026). *Edge AI-powered ANPR access control: Faster, smoother, and more precise vehicle access*. https://www.milesight.com
[10] Ajax Systems. (n.d.). *Products and integrations ecosystem*. https://ajax.systems
[11] Espressif Systems. (n.d.). *ESP32 series datasheet*. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
[12] Raspberry Pi Foundation. (n.d.). *Raspberry Pi 4 Model B*. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
[13] Raspberry Pi Foundation. (n.d.). *Raspberry Pi Camera Module V2*. https://www.raspberrypi.org/products/camera-module-v2/
[14] Components101. (n.d.). *HC-SR501 PIR sensor datasheet*. https://components101.com/sites/default/files/component_datasheet/HC%20SR501%20PIR%20Sensor%20Datasheet.pdf
[15] SparkFun Electronics. (n.d.). *4x4 matrix keypad datasheet*. https://cdn.sparkfun.com/assets/f/f/a/5/0/DS-16038.pdf
[16] Handson Technology. (n.d.). *RFID-RC522 datasheet*. https://www.handsontec.com/dataspecs/RC522.pdf
[17] Mouser Electronics. (n.d.). *28BYJ-48 stepper motor datasheet*. https://www.mouser.com/datasheet/2/758/stepd-01-data-sheet-1143075.pdf
[18] STMicroelectronics. (n.d.). *ULN2001A/ULN2003A high-voltage, high-current Darlington transistor arrays*. https://www.st.com/resource/en/datasheet/uln2001.pdf
[19] Forter. (n.d.). *CoVi Security MC-21 Brown door sensor*. https://www.forter.com.ua/datchik-vdkrittya-dvereyvkon-covi-security-mc-21-brown/
[20] Vishay Intertechnology. (n.d.). *LCD 16x2 character display (LCD016N002BCFHET) datasheet*. https://www.vishay.com/docs/37484/lcd016n002bcfhet.pdf
[21] Tesseract OCR. (n.d.). *Tesseract OCR engine*. GitHub. https://github.com/tesseract-ocr/tesseract
[22] Tower Pro. (n.d.). *SG90 servo motor datasheet*. http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf
[23] Ultralytics. (2024). *YOLOv8 documentation*. https://docs.ultralytics.com/models/yolov8/

АВТОМАТИЗОВАНА СИСТЕМА КОНТРОЛЮ ДОСТУПУ ДЛЯ РОЗУМНОГО БУДИНКУ З ВИКОРИСТАННЯМ RASPBERRY PI ТА АВТОМАТИЧНОГО РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ

Марко Табачишин[1]*, Данило Табачишин[2], Наталія Кунанець[2], Андрій Шпіцер[1]
Національний університет «Львівська політехніка»,
[1] Кафедра спеціалізованих комп'ютерних систем
вул. Степана Бандери 28а, Львів, 79013, УКРАЇНА
[2] Кафедра інформаційних систем та мереж
вул. Митрополита Андрея 5, Львів, 79016, УКРАЇНА

АНОТАЦІЯ

Вступ. Статтю присвячено розробленню системи автоматизації доступу до розумного будинку на основі одноплатного комп'ютера Raspberry Pi 4, мікроконтролерів ESP32, камери Raspberry Pi Camera Module V2, моделі комп'ютерного зору YOLOv8 (для розпізнавання номерних знаків) та технології безконтактної радіочастотної ідентифікації (РЧІ, англ. Radio-frequency identification, RFID). Актуальність дослідження зумовлена потребою підвищення рівня безпеки систем розумного будинку шляхом упровадження автономного контролю доступу. Проаналізовано сучасні рішення й підходи у сфері автоматизованого контролю доступу.

Матеріали та методи. Запропоновано архітектуру програмно-апаратної системи контролю доступу, що поєднує розпізнавання номерних знаків на основі моделі YOLOv8 і системи розпізнавання тексту Tesseract, РЧІ-автентифікацію та доступ за PIN-кодом. Для обміну даними використано протокол MQTT, моніторинг і керування реалізовано через вебінтерфейс на основі Python-мікрофреймворку Flask. Розроблено алгоритми автоматичного відкривання воріт і дверей, запису подій, керування списками авторизованих користувачів і транспортних засобів. Проведено експериментальне оцінювання системи у статичних і динамічних умовах.

Результати. Реалізовано систему контролю доступу до розумного будинку з підтримкою автоматичного розпізнавання номерних знаків (АРНЗ, англ. Automatic Number Plate Recognition, ANPR), РЧІ та PIN-автентифікації. Точність розпізнавання номерних знаків становила 94 % у передбаченому статичному режимі роботи та 50 % у додатковому динамічному стрес-тесті. Система призначена для використання лише у статичних умовах, оскільки камера встановлюється навпроти воріт. Отримані результати підтверджують працездатність запропонованої архітектури та її придатність для автоматизованого контролю доступу.

Висновки. Використання локального опрацювання даних на базі Raspberry Pi забезпечує автономність системи й зменшує залежність від зовнішніх сервісів. Практичний внесок роботи полягає у розробленні недорогої, відкритої та локально керованої системи контролю доступу, що поєднує АРНЗ, РЧІ та PIN-автентифікацію для приватного житлового використання.

Ключові слова: розумний будинок, контроль доступу, Raspberry Pi 4, YOLOv8, MQTT, автоматичне розпізнавання номерних знаків.

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
05 July, 2026	07 September, 2026	10 September, 2026	28 September, 2026