Simplify the creation of remote controls and monitoring interfaces for microcontrollers and automation systems using IoT Cloud services

Filippos Kladouchas[1,†] and Nikitas N. Karanikolas[1,*,†]

[1] *University Of West Attica, Ag. Spyridonos street 12243 Egaleo, Greece*

Abstract

Usually, remote controls of automation systems or electronic devices is based on infrared radiation. Microcontroller based automation systems can be further controlled by laptops, tablets and smart phones with the help of the local Wi-Fi and HTTP Requests services running on the automation's microcontroller. For extending further the distance of the controlling device, out of the range of the local Wi-Fi of the automation system, some extra machine (usually computer) is needed for hosting some Web Server that communicates (on one hand) through the internet with the controlling device and communicates (on another hand) with the microcontroller with the local WiFi. Consequently, we have more complicated systems that demand more infrastructures and can cause troubles. In this study, we have investigated the possibility to decrease the complexity of controlling automation systems with the IoT Cloud. Our research shows that the existing technology together with the IoT Cloud can make possible the far-away control of Microcontroller based automation systems without the complexity of an extra machine (computer) and an extra Web Server. The relevant IoT Cloud services and the alternative solutions are discussed. Further, a prototype system is also described. The positive and negative conclusions are presented.

Keywords

IoT, Microcontrollers, Home Security, IoT Cloud

1. Introduction

A microcontroller is a compact, low-power integrated circuit designed to perform dedicated control tasks in embedded systems [1]. With built-in memory, processing, and I/O capabilities, microcontrollers such as Arduino [9], ESP8266 [6], and ESP32 [7] are widely used in automation. Their programmability and modularity allow developers to create flexible and customized solutions for home, industrial, and agricultural applications. Open-source ecosystems further support this adaptability.

Cloud computing, as described in [2], delivers scalable computing resources via the internet, eliminating the need for locally managed infrastructure. In automation, it enables seamless remote access, real-time data processing, and on-demand resource allocation.

IoT Cloud platforms—purpose-built for Internet of Things applications—combine cloud computing with embedded control [3, 4]. Services like Arduino IoT Cloud [8] and Blynk [17] simplify device integration, remote management, and dashboard creation by abstracting networking and server complexity.

Microcontroller connectivity is enhanced through communication modules such as Wi-Fi (e.g., ESP8266), GSM [11], and Bluetooth. These extend access to remote networks or local devices, making automation more scalable and responsive.

[1] *RTA-CSIT 2025: 6th International Conference on Recent Trends and Applications in Computer Science, May 22-24, 2025, Tirana, Albania.*

* Corresponding author.

† These authors contributed equally.

ice18390067@uniwa.gr (F. Kladouchas); nnk@uniwa.gr (N. N. Karanikolas)

0000-0003-1777-892X (N. N. Karanikolas)

This paper explores our conjecture that IoT Cloud services can simplify remote control and monitoring for microcontroller-based automation systems. By removing the need for an on-site server, such platforms reduce cost and complexity while improving system reliability. We compare traditional and cloud-based approaches and present a case study in home security to evaluate the proposed solution's effectiveness.

2. Automation systems

For each automation scenario presented, this chapter explores and compares three distinct implementation approaches: a from-scratch solution, a commercial off-the-shelf solution, and the proposed custom IoT Cloud-based approach, which is characterized by minimal coding effort and low hardware complexity.

2.1. Smart Lighting Automation

Smart lighting systems offer remote and automated control of lights using communication protocols such as Wi-Fi, Zigbee [21], Bluetooth, and cloud technologies. They allow users to adjust brightness, set schedules, and integrate lighting into broader home automation systems.

2.1.1. From Scratch – Local Server-Based – Approach for Lighting

In a from-scratch approach, a microcontroller like Arduino [9, 10, 15] connects to lighting components (e.g., relays, LEDs) and communicates with a local server (PC or Raspberry Pi) via Wi-Fi or Ethernet. The server, running software such as Apache, Flask, or Node.js, hosts a dashboard accessible through a browser or mobile app. When the user sends a command, the server relays it to the microcontroller using HTTP [22] or MQTT [13, 23, 24], which then controls the lighting devices.

This architecture offers full control and data privacy, without relying on third-party platforms. However, it requires a continuously running local server, increasing energy use and maintenance demands. Initial setup is complex and hardware-intensive, and remote access is restricted unless additional configurations like VPN [25] or port forwarding [26] are implemented.

To enable remote access, users typically configure port forwarding to direct incoming traffic to the server's local IP. Though easy to implement, it can introduce security vulnerabilities. Alternatively, VPNs provide secure, encrypted access by placing the user virtually within the local network. VPNs are generally considered safer for sensitive applications. Both methods are viable, but VPNs offer superior privacy and control.

2.1.2. Commercial IoT Cloud-Based Solutions for Lighting

Commercial smart lighting solutions typically rely on proprietary IoT cloud platforms for automation and remote access. Devices connect directly to the manufacturer's cloud using Wi-Fi or Zigbee, and users manage them via mobile apps or voice assistants like Alexa [27], Google Assistant [28], or Apple HomeKit [29]. The cloud handles command processing and sends instructions to the lighting devices.

This plug-and-play model requires no advanced network setup and provides seamless remote access. However, it operates within closed ecosystems, limiting customization. Additionally, functionality depends entirely on the provider's cloud infrastructure; service discontinuation could render devices unusable. These solutions are generally more expensive than DIY alternatives.

2.1.3. Custom IoT Cloud-Based – Proposed – Approach for Lighting

This approach uses platforms like Arduino IoT Cloud [8], Blynk [17], or Firebase [16] to implement a customizable and scalable smart lighting system. A microcontroller (e.g., ESP8266, ESP32, or Arduino) connects to lighting components and communicates directly with the cloud, which handles automation rules, real-time data exchange, and remote access. Users control the system via a mobile app or web dashboard linked to the cloud service.

The solution eliminates the need for a local server, lowering hardware and maintenance costs. It offers greater flexibility than commercial products, allowing full customization at a lower price. While some setup and coding are required, development is simplified through prebuilt dashboards and APIs. Free-tier cloud services may impose device or data limits, but the approach remains cost-effective and suitable for personalized automation systems.

2.2. Smart Irrigation Automation

A smart irrigation system is designed to automatically manage water usage in agricultural or garden environments, optimizing water consumption while ensuring plants receive adequate moisture. These systems often use sensors (such as soil moisture sensors) to collect real-time data, and utilize microcontrollers and communication technologies (Wi-Fi, Zigbee, Bluetooth) to control irrigation valves or pumps.

2.2.1. From Scratch – Local Server-Based – Approach for Irrigation

The microcontroller (e.g., Arduino, ESP32, or Raspberry Pi) collects real-time soil moisture data from soil moisture sensors and controls water pumps or solenoid valves. The local web server (running Flask, Node.js, or Apache) processes user commands and automation rules. Users access the system via a local interface to send HTTP requests for manual or scheduled irrigation. Operating entirely within the local network, this setup guarantees full data privacy, zero cloud reliance, and full customization. However, it requires a 24/7 running server, complex network setup, and port forwarding or VPN for remote access. Additionally, power consumption is higher due to continuous operation of the local host.

2.2.2. Commercial IoT Cloud-Based Solutions for Irrigation

Commercial irrigation controllers [20] connect to the manufacturer's cloud via Wi-Fi, Zigbee, or LoRaWAN [30], using data from soil sensors, weather APIs, and past trends to automate watering schedules. Users interact with the system through mobile apps or web interfaces, often enhanced with voice assistant integration (e.g., Alexa, Google Assistant, Apple HomeKit).

These solutions are easy to set up and provide seamless remote access. Some employ AI for smart scheduling to reduce water waste. However, they are dependent on the provider's cloud infrastructure, limit user customization, and tend to be more expensive, especially with subscription-based models.

2.2.3. Custom IoT Cloud-Based – Proposed – Approach for Irrigation

In this approach, a microcontroller (ESP8266, ESP32, or Arduino) connects to soil and temperature sensors and controls water pumps through an IoT cloud platform like Arduino IoT Cloud, Blynk, or Firebase. Connectivity is established via Wi-Fi or GSM, and users monitor and manage irrigation remotely through a mobile or web dashboard. The platform handles data processing and automation logic.

This solution is cost-effective, using low-cost hardware and free or low-tier cloud services, while offering high flexibility for configuring sensors, dashboards, and rules. It also avoids vendor lock-in by relying on open-source tools. Minor coding is needed, but far less than in server-based setups. Usage limits may apply to free cloud plans, but the overall approach offers an excellent trade-off between customization, scalability, and ease of deployment.

2.3. Smart Security System

Smart Security Systems are IoT-enabled systems with the purpose to monitor, detect, and respond to potential threats in both residential and commercial environments. These systems leverage sensors, cameras, and communication technologies to provide real-time alerts and remote access.

2.3.1. From Scratch – Local Server-Based – Approach for Security

In a local server-based setup, security components operate entirely within the home network, ensuring full data privacy and no reliance on third-party services. Microcontrollers (e.g., ESP32, Raspberry Pi, or Arduino) handle sensor inputs from PIR motion detectors, door/window contacts, and cameras. Security footage is stored locally using NAS [31, 32, 33] or SD cards. A Raspberry Pi or PC hosts a web server (Flask, Node.js, Apache) that manages user interaction and communication with wireless devices.

The microcontroller triggers alarms upon detecting unauthorized access, while more demanding tasks like video processing require an additional processor. Alerts are sent via SMS, email, or local apps. Remote access is possible only through VPN or port forwarding.

This setup offers full control, no recurring fees, and high customizability. However, it requires complex server configuration, dedicated hardware, and secure network management, increasing both cost and maintenance overhead.

2.3.2. Commercial IoT Cloud-Based Solutions for Security

Commercial security systems [18, 19] rely on cloud infrastructure for monitoring, video recording, and remote access. Cameras and sensors connect via Wi-Fi or proprietary protocols to the manufacturer's cloud, where motion or intrusion events trigger alerts and cloud-based recording. Users receive notifications and can view live feeds through mobile apps or web interfaces, with many systems supporting voice assistant integration (Alexa, Google Assistant, Apple HomeKit).

These solutions are easy to deploy and offer seamless remote access with advanced features like facial recognition and cloud analytics. However, they depend entirely on the provider's cloud, limit user customization, and often involve subscription fees for storage and premium capabilities.

2.3.3. Custom IoT Cloud-Based – Proposed – Approach for Security

This solution uses open-source microcontrollers (ESP32, NodeMCU [6]) connected via Wi-Fi to IoT cloud platforms like Arduino, Blynk, or Firebase. The microcontroller interfaces with PIR sensors, contact sensors, and optionally IP cameras. Through a cloud-linked dashboard, users monitor status, trigger alarms, and receive push or email alerts in real time. The cloud service automates security responses such as activating sirens.

It combines low cost with high flexibility, allowing full customization of automation rules and notifications, without vendor lock-in. While minimal programming is needed, advanced features like video handling may require more powerful or additional processors. Free cloud tiers may also have limits, but overall, this approach balances affordability, control, and scalability for effective security automation.

3. Methodology

Here we provide the general ideas behind each of the two non-commercial implementation approaches without focus on any specific automation scenario (Lighting, Irrigation, and Security). Based on these we draw our conclusion. We then apply our conclusion in a case study of a home security automation system utilizing IoT Cloud facilitates and not a Local Server. The results of the case study system verifies our conjecture.

3.1. From scratch Approach: Using a Local Server

A from scratch method of implementing home automation uses an Arduino microcontroller with a local server (e.g., PC or Raspberry Pi) that hosts a PHP/JavaScript dashboard. Through Wi-Fi, users send HTTP requests, which the Arduino processes to control devices and return sensor data.

Although this setup offers full control and local data handling, it has limitations. A computer must run continuously, increasing energy use and maintenance needs. External access requires VPN or port forwarding, and services like No-IP [34] provide DDNS for dynamic IP resolution. However, even with DDNS, port forwarding must be configured manually, and server upkeep—including updates and security—is required.

Communication via HTTP can be slow, adding latency. Improvements include adopting MQTT for more efficient messaging and using a secondary microcontroller (e.g., ESP8266/ESP32) for network communication, leaving the main controller focused on sensors. These challenges support the adoption of IoT Cloud platforms as a simpler, more scalable alternative.

3.2. IoT Cloud-Based Solution

The IoT Cloud-based approach marks a major improvement over traditional server-based systems. Instead of relying on a local server, platforms like Arduino IoT Cloud, Blynk, and Firebase allow microcontrollers such as Arduino, ESP8266, or ESP32 to connect directly to the internet via Wi-Fi or GSM, eliminating the need for intermediary infrastructure.

This simplifies remote access—users can control devices from any location through a cloud dashboard, without requiring VPNs, port forwarding, or firewall adjustments. Cloud-based systems reduce hardware requirements, leading to lower power consumption and easier setup.

Sensor data is processed in real time by the cloud, enabling immediate response for critical applications like security or irrigation. Cloud scalability ensures that expanding the system doesn't require architectural changes.

Development is accelerated using prebuilt APIs, dashboards, and mobile apps, while platform providers handle updates and backups, reducing maintenance. Security is also improved via encrypted protocols like HTTPS and MQTT over TLS. Centralized storage enhances data analytics, and regular system updates keep devices secure.

The cost is lower than local server setups, as free or affordable cloud tiers are often sufficient. The flexibility of open platforms also allows full customization, unlike proprietary commercial alternatives.

However, stable internet connectivity is essential, and reliance on a specific provider could lead to vendor lock-in or pricing issues. Cloud-based storage also raises privacy considerations, requiring trusted providers.

In summary, IoT Cloud solutions offer a scalable, low-cost, and secure foundation for remote automation. Their simplicity and flexibility make them ideal for modern systems across a wide range of applications.

3.3. Case Study: Home Security Automation Using NodeMCU and Arduino IoT Cloud

This project presents a home security automation system developed with cost-effectiveness and scalability in mind. The implementation leverages a NodeMCU board (ESP8266-based) as the primary microcontroller, combined with a range of hardware components and cloud services for remote monitoring and notification. The following paragraphs detail the system's hardware configuration, signal conditioning and isolation, user interface, cloud integration, and alternative notification solutions.

To achieve an efficient and economical design, the system utilizes a NodeMCU microcontroller operating at 3.3V. Given that many peripheral components—such as sensors, sirens, and batteries—commonly require a 12V supply, a dual-voltage approach was necessary. A commercially available 12V power supply was employed to power these components. However, to ensure the NodeMCU operates reliably, the 12V input had to be strictly reduced to the 3.2–3.3V range. This voltage regulation was achieved using an LM317 voltage regulator [5], which provided a stable 3.3V output for the NodeMCU.

To protect the sensitive logic circuitry of the NodeMCU and ensure proper interfacing between different voltage levels, electrical isolation was implemented. The outputs from the NodeMCU that drive the siren and buzzer were routed through optocouplers (PC817X). These devices, through their internal photodiode, offer galvanic isolation between the microcontroller and the high-power output circuitry. Additionally, transistors were connected on the 12V side to drive the external components (sensors and sirens) while providing further isolation and protection. This dual approach—using optocouplers combined with transistors—ensured that any voltage spikes or faults in the high-power circuitry did not affect the microcontroller.

For local, manual control of the system, a 3x4 matrix keypad was integrated. Although a typical keypad configuration would require seven digital I/O pins, this implementation leveraged a specially designed voltage divider to encode the keypad output into a single analog input. The NodeMCU's analog pin, which accepts voltages in the 0–3.3V range, reads the normalized values (mapped to a range of 0–1023) corresponding to different key presses. This configuration simplifies the hardware interface while ensuring reliable detection of user input for system activation and deactivation.

The schema in figure 1 represents the General Circuit Diagram of the designed home security automation. It respects/implements all the above (system's hardware configuration, power regulation, signal isolation, output control and local user input). The physical prototype, the implemented system, can be seen in figure 2.

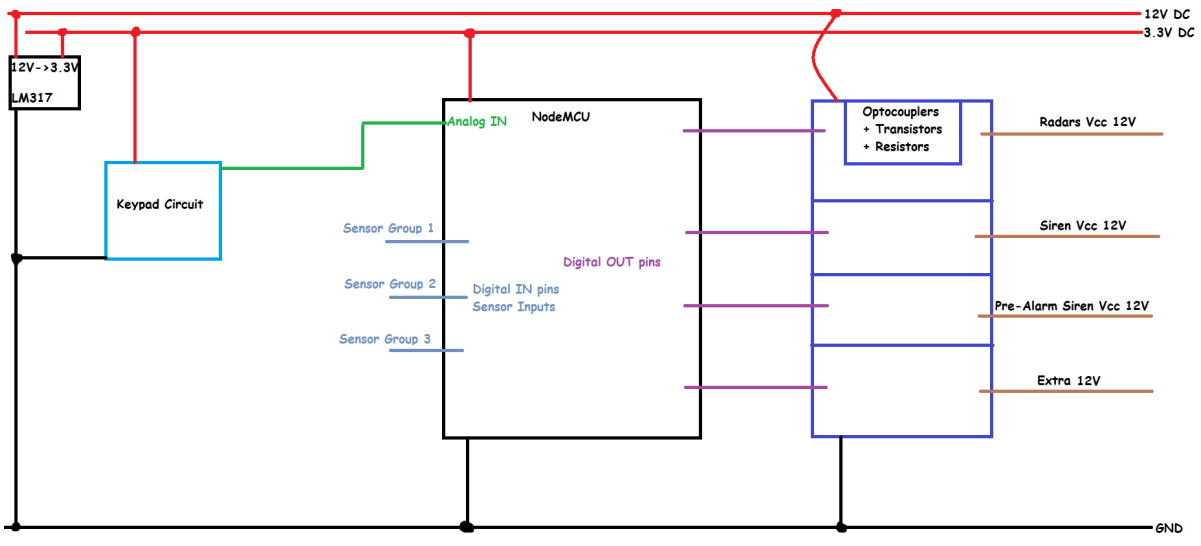

Figure 1: The General Circuit Diagram of the designed home security automation.

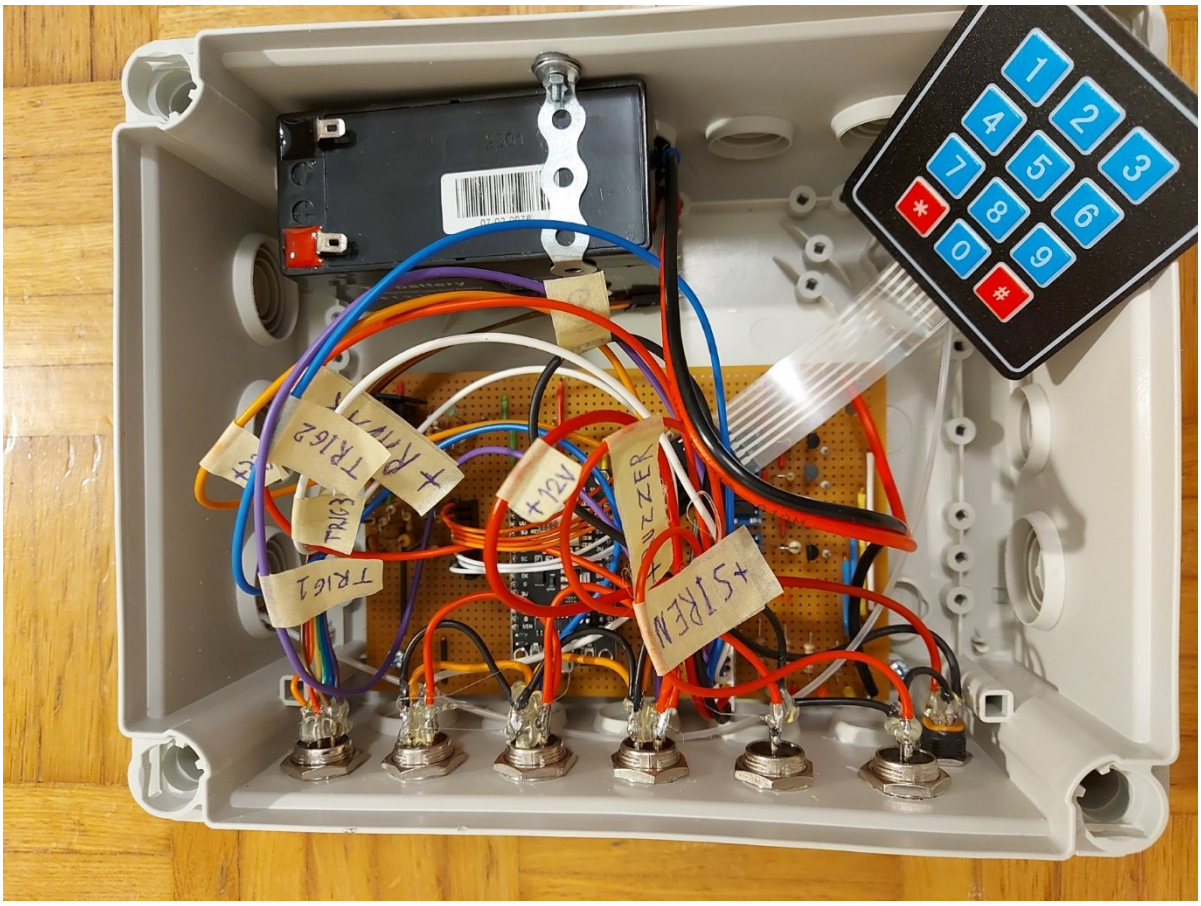

Figure 2: The physical / implemented prototype.

Remote access and control were achieved by integrating the system with the Arduino IoT Cloud. A dedicated cloud server was chosen due to its robust IoT services and strong security features. In the Arduino Cloud, a device object—named "JARVIS" (inspired by the artificial intelligence from Iron Man comics)—was created. This object holds various variables representing the states of sensors and the overall alarm system. A custom graphical dashboard was implemented in the cloud, allowing real-time reading and updating of these variables. Users interact with the system via the Arduino IoT Cloud Remote [12] mobile application, which provides an intuitive interface for monitoring sensor data and controlling the security system.

The notification system of the security automation operates in an autonomous, state-driven [14] manner. Each sensor is programmed with a series of states: IDLE, SENSOR_TRIGGERED, NOTIFY_SENT, ALARM_TRIGGERED, and ALARM_NOTIFY_SENT. These states determine the progression of the sensor's response to detected events. For example:
Detect Mode: Sensors in this mode can progress up to the NOTIFY_SENT state.
Detect & SMS Mode: These sensors can trigger both notifications and additional actions.
Armed Mode: Sensors configured in armed mode can trigger the alarm if an event occurs, even if the siren is already active, and they follow a specific countdown mechanism that eventually returns them to either IDLE or ALARM_TRIGGERED state.
This hierarchical state management ensures that the system can differentiate between minor events and serious security breaches, thereby optimizing the response actions such as sending notifications and activating alarms.

Although Arduino IoT Cloud supports email triggers, alternatives were explored to improve performance. These included HTTP-based email via Mailjet's API and direct SMTP email sending from the microcontroller. However, both methods added processing overhead to the NodeMCU, prompting the use of a secondary microcontroller. In this dual-processor setup, a secondary module (e.g., ESP-01) manages notifications independently, communicating with the primary NodeMCU over

serial or I2C. Offloading these tasks reduces latency and ensures that the primary microcontroller's performance is not compromised by intensive network operations.

The case study demonstrates a comprehensive approach to developing a home security automation system that is both cost-effective and scalable. By employing a NodeMCU-based architecture with careful voltage regulation, signal isolation, and intelligent sensor state management, the system achieves robust local performance. Integrating with the Arduino IoT Cloud facilitates secure remote access and user-friendly control via a mobile dashboard. Furthermore, exploring alternative notification methods and the incorporation of a secondary microcontroller addresses the processing limitations inherent in single-board designs, ensuring real-time performance and reliability. This multi-layered approach not only meets the immediate requirements for home security but also provides a flexible framework for future expansion and enhancements.

4. Discussions and Conclusions

The integration of IoT Cloud services into automation systems marks a significant advancement over traditional and commercial methods. Platforms like Arduino IoT Cloud and Blynk allow microcontrollers (e.g., ESP8266, NodeMCU, ESP32) to connect directly to cloud infrastructure, eliminating the need for a local server. This reduces hardware costs, setup complexity, and ongoing maintenance.

IoT Cloud solutions provide secure, built-in remote access via dashboards or mobile apps, avoiding the need for VPNs or port forwarding. This capability is essential in applications like home security, but also beneficial in lighting, irrigation, and broader automation use cases.

Their flexibility and scalability enable system expansion without reworking the architecture. The use of affordable hardware and free or low-cost cloud tiers makes them highly accessible. Unlike proprietary commercial solutions, which limit customization, IoT Cloud systems give developers full control over logic, interfaces, and data handling. As shown in the case study, sensor-based states can be tailored to different needs, while protocols like MQTT reduce latency and improve responsiveness.

By shifting processing and communication to the cloud, these systems become more energy-efficient and resilient to local server failures. Although commercial products offer simplicity, they often require subscriptions and restrict user control.

While concerns remain—such as reliance on internet connectivity or vendor-specific platforms—the benefits clearly outweigh the drawbacks. Hybrid approaches, combining local fallbacks with cloud services, may further improve reliability.

In conclusion, IoT Cloud platforms offer a cost-effective, customizable, and scalable foundation for smart automation, enabling the development of robust, user-centric systems that can evolve with future needs.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT, Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication’s content.

References

[1] Márquez-Vera, M.A., Martínez-Quezada, M., Calderón, R., Rodríguez, A., & Ortega-Mendoza, R. (2023). Microcontrollers programming for control and automation in undergraduate biotechnology engineering education. Digital Chemical Engineering, Volume 9, December 2023. doi:10.1016/j.dche.2023.100122

[2] Praveen Borra (2024). An overview of Cloud Computing and Leading Cloud Service Providers. International Journal of Computer Engineering and Technology, Volume 15, Issue 3, May-June 2024, pp. 122-133. doi:10.17605/OSF.IO/5HQ4M.
[3] Alessio Botta, Walter de Donato, Valerio Persico, Antonio Pescapé (2016). Integration of Cloud computing and Internet of Things: A survey. Future Generation Computer Systems, Volume 56, Pages 684-700. doi:10.1016/j.future.2015.09.021.
[4] H. F. Atlam, A. Alenezi, A. Alharthi, R. J. Walters and G. B. Wills (2027). Integration of Cloud Computing with Internet of Things: Challenges and Open Issues. IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, UK, 2017, pp. 670-675. doi:10.1109/iThings-GreenCom-CPSCom-SmartData.2017.105.
[5] Texas Instruments, LM317 Adjustable Voltage Regulator, URL: https://www.ti.com/product/LM317.
[6] Espressif Systems, NodeMCU (ESP8266), URL: https://www.espressif.com/en/products/hardware/esp8266ex/overview.
[7] Espressif Systems, ESP32 Overview, URL: https://www.espressif.com/en/products/socs/esp32.
[8] Arduino, Arduino Cloud, URL: https://www.arduino.cc/en/ArduinoCloud.
[9] Arduino, Arduino Uno Rev3, URL: https://store.arduino.cc/arduino-uno-rev3.
[10] Arduino, Arduino Mega 2560 Rev3, URL: https://store.arduino.cc/arduino-mega-2560-rev3.
[11] Arduino, GSM Shield 2.0, URL: https://store.arduino.cc/usa/gsm-shield-2.0.
[12] Arduino, Arduino IoT Remote, URL: https://www.arduino.cc/en/ArduinoIoTRemote.
[13] EMQ Technologies Co., MQTT vs HTTP: Why MQTT Is Better for IoT, URL: https://www.emqx.com/en/blog/mqtt-vs-http.
[14] J. E. Hopcroft, R. Motwani, and J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, 3rd ed. Boston, MA, USA: Pearson Education, 2006.
[15] (in Greek) Παπάζογλου and Σ. Π. Λιωνής, Ανάπτυξη Εφαρμογών με το Arduino, 3η έκδοση, Εκδόσεις Τζιόλα.
[16] Google, Firebase and Google Cloud, URL: https://firebase.google.com/firebase-and-gcp.
[17] Blynk, Blynk: A Low-Code IoT Software Platform, URL: https://blynk.io/.
[18] Ring, Business Security Systems - Cameras, Alarms, and More, URL: https://ring.com/business-security-systems.
[19] Arlo, Arlo Security Cameras System & Video Doorbells, URL: https://www.arlo.com/en-us/.
[20] RainMachine, RainMachine - Forecast Smart WiFi Irrigation Controllers, URL: https://www.rainmachine.com/.
[21] Zigbee Alliance, FAQ. (Archived on Wayback Machine.) URL: https://web.archive.org/web/20130627022836/http://www.zigbee.org/About/FAQ.aspx
[22] T. Berners-Lee, Basic HTTP as defined in 1992. World Wide Web Consortium, 1992. URL: https://www.w3.org/Protocols/HTTP/AsImplemented.html.
[23] OASIS, “MQTT Version 5.0, OASIS Standard, 07 March 2019”. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.
[24] S. Cope, “MQTT For Complete Beginners: Learn The Basics of the MQTT Protocol”, 2022, p. 71. ISBN: 9798779030762.
[25] NIST Computer Security Resource Center (CSRC) Glossary, “virtual private network (VPN)”. URL: https://csrc.nist.gov/glossary/term/virtual_private_network
[26] PC Magazine Encyclopedia, “port forwarding”. URL: https://www.pcmag.com/encyclopedia/term/port-forwarding.
[27] Wikipedia, Amazon Alexa. URL: https://en.wikipedia.org/wiki/Amazon_Alexa.
[28] Wikipedia, Google Assistant. URL: https://en.wikipedia.org/wiki/Google_Assistant.
[29] Apple Inc., HomeKit, URL: https://developer.apple.com/documentation/homekit/
[30] Wikipedia, LoRa. URL: https://en.wikipedia.org/wiki/LoRa.
[31] Seagate, NAS vs Desktop. URL: https://www.seagate.com/files/www-content/product-content/nas-fam/nas-hdd/en-gb/docs/nas-vs-desktop-marketing-bulletin-mb633-1-1304gb.pdf.

[32] Wikipedia, Network-attached storage. URL: https://en.wikipedia.org/wiki/Network-attached_storage.
[33] B. Callaghan, NFS Illustrated, Boston, MA, USA: Addison-Wesley, 2000. ISBN: 0-201-32570-5.
[34] P. Vixie (ed.), S. Thomson, Y. Rekhter, and J. Bound, Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136, Apr. 1997. URL: https://doi.org/10.17487/RFC2136.