*electronics*

*Article*

# The Application of Explainable Artificial Intelligence to Low-Power Internet of Things Devices with Secure Communication Using Chaos-Based Cryptography

**Algirdas Dobrovolskis * and Egidijus Kazanavičius**

Faculty of Informatics, Kaunas University of Technology, 44249 Kaunas, Lithuania
* Correspondence: algdobr@ktu.lt

**Abstract:** This paper investigates the feasibility of employing expert knowledge-based Explainable Artificial Intelligence (XAI) for smart house control through low-power Internet of Things (IoT) devices that possess limited computational capabilities. By integrating Explainable AI, we seek to enhance the transparency of the model's decision-making process, thereby increasing its reliability for the end user. The Arduino Uno board was selected for IoT development because of its extensive popularity and affordability. A model of heating control has been developed using temperature sensors based on the presence of residents in the room. The operational prototype was evaluated by measuring the time taken between data input and decision-making, accompanied by an explanation, to identify any potential bottlenecks that may hinder the performance of the microcontroller. To enhance communication security, we developed a pseudo-random number generation function using chaos-based cryptography with hardware implementation, thus improving communication security without incurring additional computational costs. The method has demonstrated a time efficiency improvement of up to 67% for novice users, 58% for intermediate users, and 50% for expert users.

**Keywords:** explainable AI; expert knowledge; Internet of Things; embedded systems; chaos-based cryptography

check for updates

Academic Editors: Meng-Luen Wu, Feng-Cheng Chang and Chang-Yi Kao

Received: 27 February 2025
Revised: 19 March 2025
Accepted: 20 March 2025
Published: 22 March 2025

**Citation:** Dobrovolskis, A.; Kazanavičius, E. The Application of Explainable Artificial Intelligence to Low-Power Internet of Things Devices with Secure Communication Using Chaos-Based Cryptography. *Electronics* **2025**, *14*, 1255. https://doi.org/10.3390/electronics14071255

## 1. Introduction

Communication security plays a crucial role in IoT, because these systems frequently handle sensitive data. IoT devices transmit critical information, such as personal health records from wearables, financial data from smart payment solutions, and essential business information in industrial IoT contexts [1,2]. Ensuring secure communication channels is imperative to protect data from unauthorized access and interception.

Given these security challenges, the scientific community has shown increasing interest in adopting chaos-based cryptography for IoT applications. Chaotic systems provide advantages over conventional cryptographic techniques, including complexity, resistance to brute force attacks, and suitability for low-power applications. Many chaos-based security techniques in IoT leverage lightweight cryptosystems, data integrity protocols, image encryption techniques, and even field-programmable gate arrays (FPGAs). While these systems enhance security, they often come with computational overhead and high costs, particularly when FPGAs are involved [3–5].

To address security concerns, our paper proposes a power-efficient and cost-effective solution using PWM (pulse width modulation) and RC (resistor capacitor) circuits to generate pseudo-random numbers for encrypting communication protocol packets. Employing

RC circuits for random number generation is a well-established technique in hardware-based cryptographic systems [6], offering an effective balance between security, efficiency, and cost.

Power efficiency also plays a vital role [7] in the IoT landscape, primarily because of the extensive use of IoT devices that typically have a limited battery life. These devices are generally compact, powered by batteries, and often situated in hard-to-reach areas, complicating the process of battery replacement or recharging. Consequently, it is essential for these devices to be engineered for optimal power efficiency to enhance their operational lifetime and minimize expenses related to battery maintenance or replacement [8].

Furthermore, IoT devices generally utilize wireless communication over extended distances, necessitating considerable power for data transmission. An energy-efficient IoT device consumes less energy during data transmission, thereby extending battery life and reducing the environmental impact of IoT networks [9,10]. Beyond enhancing device longevity, energy efficiency is also crucial for the scalability of IoT networks [11,12]. As the number of IoT devices continues to grow, so does the total energy consumption of the network, leading to increased operational costs. Implementing energy-efficient IoT devices can mitigate these challenges and contribute to the environmental sustainability of IoT ecosystems [13,14]. Therefore, energy efficiency is a key consideration in IoT design, influencing durability, scalability, and ecological viability [15].

In parallel, achieving optimal performance in IoT systems integrated with deep learning algorithms necessitates specialized hardware and significant computational resources. Deep learning relies on intricate mathematical models, such as neural networks, which may contain millions of parameters requiring extensive processing power for training [7,16]. These computations demand substantial memory and processing capabilities, often beyond the reach of conventional hardware. To address this, specialized hardware such as GPUs (graphics processing units), NPUs (neural processing units), and TPUs (tensor processing units) are employed to enhance the computational efficiency [17,18].

Furthermore, effective deep learning requires vast datasets to train neural networks for pattern recognition and predictive analysis. The collection and storage of large datasets poses significant challenges, requiring substantial resources and time to accumulate sufficient data for effective algorithm training [19].

Several IoT applications rely on deep learning, including smart surveillance systems that use real-time video analytics, predictive maintenance in industrial IoT to forecast equipment failures, and healthcare monitoring devices that analyze patient vitals for early disease detection. While deep learning models excel in handling complex patterns and large datasets, they require extensive computation, making them impractical for resource-constrained IoT devices [17].

In contrast, fuzzy expert maps offer a more resource-efficient alternative for these applications. For example, instead of using deep learning for predictive maintenance in industrial IoT, fuzzy expert maps can apply expert-driven rules to assess equipment health based on temperature, vibration, and operational time, reducing the need for high-power computations [20,21]. Similarly, in healthcare monitoring, rather than training deep neural networks on vast patient datasets, fuzzy expert maps can use predefined rules from medical experts to evaluate abnormal conditions in patient vitals. This approach not only minimizes computational requirements but also provides transparent decision-making, which is critical in high-stakes applications like healthcare.

Fuzzy expert maps, in contrast, can operate on any hardware that supports C++ code and does not necessitate high processing capabilities. Additionally, it does not require data collection, as it relies on expert knowledge throughout the process [20,21]. The system's operation is articulated through rules formulated in if–then statements, which are significantly less resource-intensive than training deep learning algorithms with unprocessed data. This approach is suitable for the early stages of the operation of the IoT system when no data have been collected. Furthermore, this approach is readily explicable through the concept of computing with words. This makes fuzzy expert maps particularly suitable for the early stages of IoT system operation when no data have been collected. Moreover, they align with the concept of computing with words, enhancing explainability.

Explainability is significant in the realm of IoT, especially as IoT systems become more intricate and integrate with AI technologies. Incorporating explainable AI into the IoT not only promotes user confidence and promotes system transparency but also plays an important role in enhancing security and operational efficiency [22].

By integrating power-efficient IoT designs with secure and explainable AI approaches, we can create more sustainable, scalable, and trustworthy IoT ecosystems.

The novelty and contributions of the authors in this study are as follows:

- securing communication by integrating the chaos-based hardware number generation algorithm for encrypting packets sent by devices to the network;
- a method for defining the IoT system using expert knowledge rules and explainable code generation using AI;
- incorporation of explainability into code generation and control systems enabling users to comprehend the rationale behind their decisions, a factor that is essential for fostering user trust in AI-driven applications.

This manuscript is organized into sections as follows. Section 2 presents a comprehensive review of the relevant literature and essential works. The method for deploying secure IoT devices based on expert knowledge and explainability is thoroughly described in Section 3. A rigorous experimental analysis is conducted in Section 4. The paper concludes with Section 5, which discusses final insights and suggests directions for future inquiry.

## 2. Related Works

Security remains a critical aspect of IoT, particularly in communication. B. Ilyas et al. [2] underscored the importance of communication security due to the sensitive nature of data handled by IoT devices. Various studies [3] have highlighted the necessity of securing communication channels, as IoT devices often transmit personal health data, financial transactions, and crucial business information. Protecting these channels is essential to prevent unauthorized access and data interception.

To address IoT security concerns, J. R. Naif et al. [4] proposed an adapted version of the advanced encryption standard (AES) that incorporates chaotic systems to enhance security while maintaining a lightweight performance. This approach aims to defend against diverse cyber threats while ensuring data confidentiality and integrity in resource-constrained IoT environments.

Numerous surveys [7–9,11] have demonstrated the extensive utilization of IoT and highlighted the significance of power efficiency within the sector, primarily due to the extensive use of IoT devices that typically have a limited battery life. These devices are generally compact, powered by batteries, and often situated in hard-to-reach areas, complicating the process of battery replacement or recharging. Consequently, it is essential for these devices to be engineered for optimal power efficiency to enhance their operational lifetime and minimize expenses related to battery maintenance or replacement.

Building upon this need for efficiency, several reviews [10,12,13] delved into the evolution of standard sensors into smart sensor technologies, emphasizing the enhancement of data processing capabilities through AI. These advancements facilitate real-time decision-making and improve the overall efficiency of IoT systems. The reviews further explore innovations in sensor design, advanced data processing techniques, communication protocols, and the pivotal role of AI in enabling autonomous IoT applications. Additionally, they address existing challenges and outline future directions for AI-enhanced sensors in IoT applications.

As AI becomes increasingly integrated into IoT, the explainability of AI models gains importance. S. Ali et al. [8] and others [15–21,23–25] have discussed the significance of explainability in AI, while surveys [16,17] have emphasized the critical role of Explainable AI (XAI) in Industry 4.0 and smart cities. XAI is essential for bolstering trust, ensuring regulatory compliance, and improving efficiency in AI-driven industrial operations. Furthermore, these studies highlight the obstacles faced and propose research avenues for the deployment of Explainable AI solutions in industrial environments.

In parallel, energy-saving strategies in IoT-integrated smart homes have gained attention. Singh, P.P. et al. [26] provided a comprehensive examination of energy-efficient design aspects, data security models, and automation solutions in smart homes. They concluded that the deployment of intelligent scheduling algorithms for energy consumption may require sophisticated computational resources and resilient communication protocols to function effectively.

Beyond security, network resilience and efficiency are essential for the advancement of IoT applications. Y. He et al. [27] explored collaborative strategies, potential applications, and challenges in the implementation of integrated networks. By examining the interplay between aerial and terrestrial elements, the authors offered insights into developing more robust and efficient vehicular networks for future transportation systems.

Finally, addressing the challenge of optimizing sensor network coverage, Y. Yao et al. [28] proposed an adaptive Q-Learning algorithm (AQL) to determine the minimum exposure path (MEP) in directional sensor networks. They reinterpret the MEP problem as a path-planning challenge and introduce a weighted grid model to enhance action selection in Q-Learning. Their study also presents a strategy for optimizing security redundant node deployment to improve network coverage and intrusion detection capabilities. The simulation results indicate that their proposed method surpasses traditional techniques, achieving a 35.13% increase in network coverage and a 47.58% reduction in energy consumption associated with node movement.

Although there have been prior implementations of utilizing LLM for various source code generation and explanations [29,30], they have largely focused on AI, thereby missing the opportunity to leverage human expert knowledge effectively.

In summary, the extensive deployment of IoT devices necessitates advancements in power efficiency, AI-driven decision-making, security, and network resilience. The integration of Explainable AI, encryption techniques, and optimization algorithms plays a crucial role in shaping the future of IoT applications, ensuring efficiency, security, and sustainability. As a result, we proposed a method for deploying secured IoT devices based on expert knowledge to tackle explainability and security concerns.

## 3. A Method for Deploying Secured IoT Devices Based on Expert Knowledge and Explainability

Since the computational capability of low-energy IoT devices prevents them from executing deep learning algorithms, expert rule control using a large language model (LLM) to obtain the code from the rules is proposed, thus saving time and reducing the probability of human errors.

The image in Figure 1 depicts a high-level framework of an IoT system comprising two main components: the expert toolkit and the microcontroller.

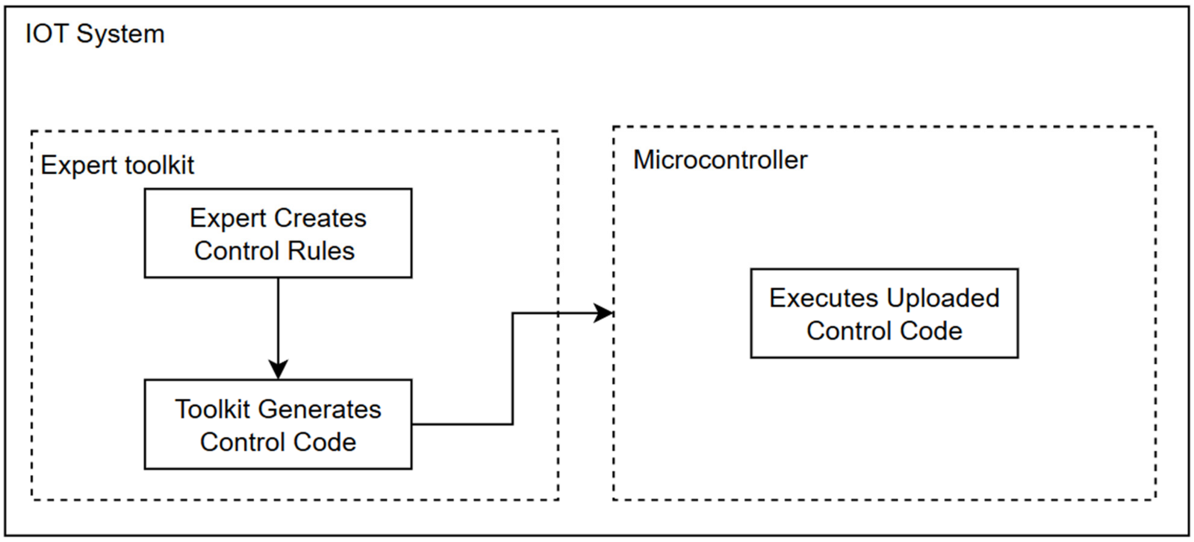

IOT System
Expert toolkit
Expert Creates Control Rules
Toolkit Generates Control Code
Microcontroller
Executes Uploaded Control Code

**Figure 1.** Proposed IoT framework concept.

Expert Creates Control Rules: An expert defines the control rules for the system using human language terms. These rules are based on the desired behavior and requirements of the IoT application.

Toolkit Generates Control Code: Once the control rules are created, the toolkit automatically generates the corresponding control code. This code is necessary for the microcontroller to execute the specified rules.

The microcontroller executes the uploaded control code: The control code generated by the expert toolkit is uploaded to the microcontroller. The microcontroller then executes this control code to perform the desired operations within the IoT system.

The flow chart of the method is shown in Figure 2 and has the following steps:

1. The expert uses the toolkit to create control rules.
2. The toolkit converts these rules into the control code.
3. The expert reviews the code and rectifies any problems identified.
4. The control code is then uploaded to the microcontroller.
5. The microcontroller executes the uploaded control code to manage the IoT system's functions.
6. The expert assesses the functionality of the microcontroller and rectifies any identified errors.

During the rule creation phase, the expert utilizes a specialized toolkit to formulate control rules. These rules delineate the expected behavior of the IoT system. Typically, the creation of these rules occurs through an intuitive user interface.

During the code generation phase, the toolkit autonomously transforms established rules into executable code. This automation minimizes the occurrence of manual coding errors. The code produced adheres to predetermined patterns and standards.

In the code review phase, the expert evaluates the code produced. They assess the score for logical inconsistencies and potential optimization enhancements. Any identified issues are recorded and rectified.

During the upload phase, the validated code is transferred to the microcontroller. This process generally requires the use of designated programming tools. The uploading procedure encompasses verification checks.

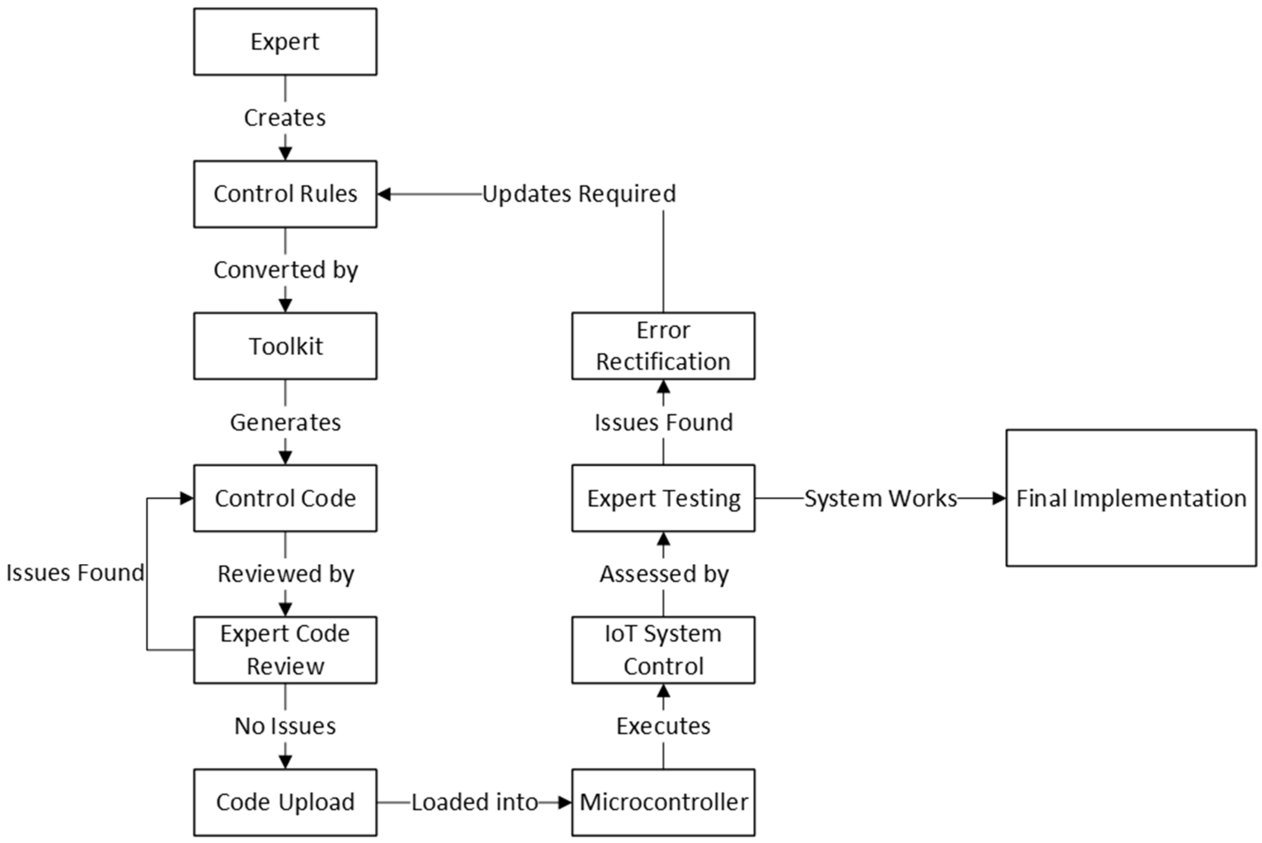


**Figure 2.** Flow chart illustrating the procedure of an expert rule system.

During the execution phase, the microcontroller executes the code uploaded. It supervises multiple functions of the IoT system, initiating a real-time assessment of the system's behavior.

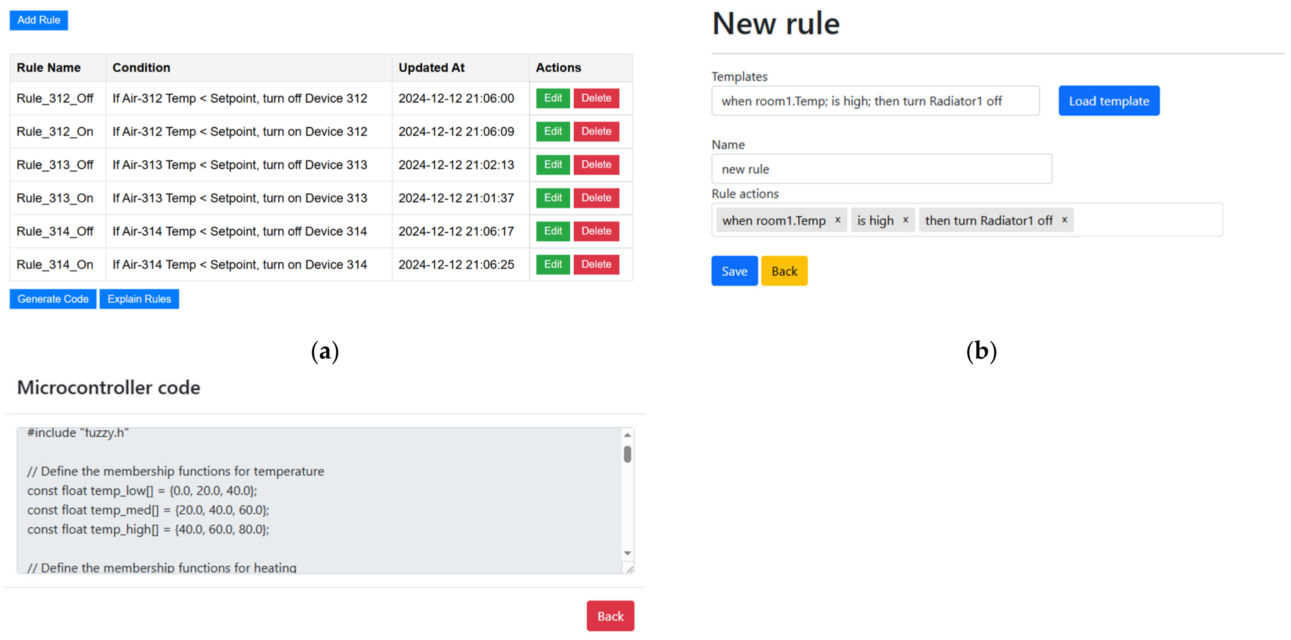
Throughout the testing and rectification phase, the expert observes the performance of the system. All implemented functions undergo testing, allowing for the identification and correction of any errors. If required, the process may revert to the initial step for modifications to the rules.

This approach guarantees the following benefits: minimized manual coding mistakes through automation, structured validation in various phases, the ability to continuously improve, and distinct differentiation between rule formulation and execution.

The used expert toolkit shown in Figure 3a was developed to simplify the creation of control rules for human experts. It has the ability to create and edit rules from scratch or use existing rule templates.

The page to add a new rule is shown in Figure 3b. The first step was to define the control model using IF<sensor>, THEN<actuator> syntax, from scratch, or by loading a suitable template from the template list to define the fuzzy model to use the eFLL library.

The rules that have been entered are subject to the expert's verification. If the complete set of rules for the current device is included in the rule list, a request for code generation can then be made to the framework. While expert knowledge is utilized to build the rules, the actual relationships between the membership functions and fuzzy rule sets for each input are generated using LLM, as seen in Figure 3c.



| Rule Name | Condition | Updated At | Actions |
|---|---|---|---|
| Rule_312_Off | If Air-312 Temp < Setpoint, turn off Device 312 | 2024-12-12 21:06:00 | Edit Delete |
| Rule_312_On | If Air-312 Temp < Setpoint, turn on Device 312 | 2024-12-12 21:06:09 | Edit Delete |
| Rule_313_Off | If Air-313 Temp < Setpoint, turn off Device 313 | 2024-12-12 21:02:13 | Edit Delete |
| Rule_313_On | If Air-313 Temp < Setpoint, turn on Device 313 | 2024-12-12 21:01:37 | Edit Delete |
| Rule_314_Off | If Air-314 Temp < Setpoint, turn off Device 314 | 2024-12-12 21:06:17 | Edit Delete |
| Rule_314_On | If Air-314 Temp < Setpoint, turn on Device 314 | 2024-12-12 21:06:25 | Edit Delete |

(**a**)

New rule

Templates: when room1.Temp; is high; then turn Radiator1 off

Name: new rule

Rule actions: when room1.Temp × is high × then turn Radiator1 off ×

(**b**)

Microcontroller code

```
#include "fuzzy.h"

// Define the membership functions for temperature
const float temp_low[] = {0.0, 20.0, 40.0};
const float temp_med[] = {20.0, 40.0, 60.0};
const float temp_high[] = {40.0, 60.0, 80.0};

// Define the membership functions for heating
```

(**c**)

**Figure 3.** The expert toolkit interface: (**a**) the rule list; (**b**) new rule page; (**c**) generated code page.

The prompt to generate LLM code is structured to provide three key items: IoT platform type and example code, as well as the rules provided by the expert. For example, "Generate {Arduino Uno} code, using example {code} with following {rules}, explain how rules are converted to code in comments". IoT devices frequently depend on particular libraries or APIs, such as Adafruit, Arduino, and ESP32. Additionally, IoT devices often possess platform-specific requirements, including GPIO pin configurations and real-time constraints. By incorporating a code example, one ensures that the language model utilizes the appropriate libraries and API calls. Code examples serve to illustrate the correct syntax, libraries, and overall structure necessary for the task at hand. This practice aids the language model in producing code that aligns with best practices and is more likely to function without issues. Should the initial code fail to operate as intended, one can enhance the prompt with further details or corrections. This iterative approach contributes to the refinement of the generated code's quality. This methodology diminishes the chances of errors and ensures that the resulting code is both functional and robust. The process also eliminates the need for manual labor in transferring control rules into the program code while enabling experts to validate the accuracy of the code generated by the AI. The intention of the method is to launch a campaign that advocates for cooperation in place of competition, with the goal of preventing exclusion and building trust within the community.

The code should also be tested using the embedded system to see how actuators would respond to various input values and to see whether it is functioning properly, and all the rules should function as intended.

For enhanced communication security, we implemented the chaos-based hardware number generation algorithm for encrypting packets sent by devices to the network. To produce a chaotic signal, PWM was set on digital pin D3 and connected to a RC circuit that feeds into the analog input A0. As illustrated in Figure 4, the circuit is relatively simple and can be integrated into a wide range of IoT devices.

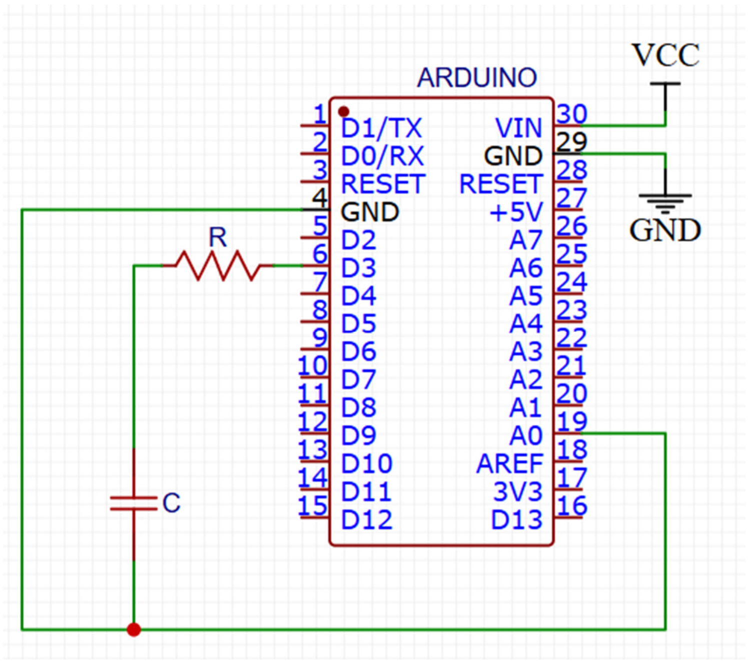


**Figure 4.** PWM to RC circuit for the Arduino microcontroller.

The output is pseudo-random numbers generated using three interchangeable parameters:

R—resistance of the resistor;
C—capacity of the capacitor;
F—PWM frequency.

Modifying any of the three parameters results in a complete change to the output function. Algorithm 1 outlines the code responsible for generating pseudo-random numbers and transmitting them to the serial port for reading.

The provided Algorithm 1 produces pseudo-random numbers utilizing a combination of analog input, PWM output, and a differential filter. The Initialization specifies the PWM pin number, the analog input pin number, the resistor value, the capacitor value, and the sampling rate as constants. The setup configures the PWM pin as an output and the analog pin as an input and establishes serial communication for debugging purposes. The main loop processes duty cycles in the following manner: It transmits the current duty cycle to the PWM pin, where the value is subjected to a differential filter. Subsequently, it acquires the analog value from the input pin and outputs the result to the serial port.

The security algorithm outlined is seamlessly embedded in the example code of the microcontroller, ensuring that all generated code will thereby safeguard the device's communication in the IoT network. During the TLS handshake, random values are exchanged to facilitate secure communication. In this instance, random numbers generated through our method were utilized to substitute the standard values, thereby enhancing security. Given that random numbers play a crucial role in cryptography, this approach could also be implemented in various other applications, including certificate generation and value hashing.

The generated code directly embodies the principles of the rules, enhancing the overall comprehensibility of the code. Each rule is designated as a separate code segment, which streamlines the adjustment process. Furthermore, the LLM provides explanations for the code, making code validation process for the expert.

**Algorithm 1** PWM Control with Moving Average Filtering

```
1:  Constants:
2:  $pwmPin \leftarrow 3$                                          ▷ PWM pin number
3:  $analogPin \leftarrow A0$                                      ▷ Analog input pin number
4:  $resistorValue \leftarrow 560$                                 ▷ Resistor value in ohms
5:  $capacitorValue \leftarrow 100$                                ▷ Capacitor value in microfarads
6:  $sampleRate \leftarrow 200$                                    ▷ Sampling rate in Hz
7:  procedure SETUP
8:  pinMode(pwmPin, OUTPUT)
9:  pinMode(analogPin, INPUT)
10:  Serial.begin(9600)
11: end procedure
12: procedure LOOP
13:  for $dutyCycle = 0$ to 255 do
14:    analogWrite(pwmPin, dutyCycle)
15:    $analogValue \leftarrow$ analogRead(analogPin)
                                                          ▷ Simple moving average filtering
16:    Static Variables:
17:    $filterBuffer \leftarrow$ array of size $sampleRate$
18:    $filterIndex \leftarrow 0$
19:    $filterBuffer[filterIndex] \leftarrow analogValue$
20:    $filterIndex \leftarrow (filterIndex + 1) \mod sampleRate$
21:    $filteredValue \leftarrow 0$
22:    for $i = 0$ to $sampleRate - 1$ do
23:      $filteredValue \leftarrow filteredValue + filterBuffer[i]$
24:    end for
25:    $filteredValue \leftarrow filteredValue/sampleRate$
26:    Serial.println(filteredValue)
27:    delay(1000/sampleRate)                    ▷ Delay for desired sampling rate
28:  end for
29: end procedure
```

Expert-defined control rules provide a fundamental structure for generating explanations within IoT systems, enabling users to gain a clearer understanding of the system's operations and the rationale behind specific actions. These control rules are expressed in an "if–then" format. For instance, "If the temperature exceeds 25 °C, then turn on the air conditioner", and the system is capable of verbalizing the rule in straightforward language. For example, "The air conditioner was turned on because the room temperature reached 27 °C, which is above your preferred maximum of 25 °C".

## 4. Results

Publicly accessible LLM services were tested to see if they could generate microcontroller code with rules provided by experts. The comparative results are displayed in Table 1.

The rules were generated using Gemini, GPT-4, and mistral language models. First, only the rules were provided for LLMs. Gemini and GPT4 were able to generate a working code for the microcontroller, while Mistral only provided a pseudocode. On the second attempt, a working code example was provided, along with expert rules. Mistral was also able to generate a working code. The third attempt was to add more complex rules and request a code for the microcontroller. Only GPT4 provided a completely working code, while Gemini and Mistral failed to implement a code that was compiled with additional rules and could be added to the code manually.

**Table 1.** Comparison of LLM performance for microcontroller code generation.

| | **Gemini 1.5** | **Mistral** | **GPT-4o** |
|---|---|---|---|
| **Only rules** | Yes | No | Yes |
| **With a code example** | Yes | Yes | Yes |
| **Added 2 more rules** | No | No | Yes |
| **Quality index [31]** | 71 | 60 | 71 |
| **Latency** | 0.35 | 0.45 | 0.41 |
| **CodeBLEU score** | 41.37 | 42.36 | 48.87 |

Since the code from the rules was created through several attempts with different prompts for the LLMs, the most effective result of each AI was evaluated. The generated code underwent a comparison with the reference code developed by an expert, using the CodeBLEU Python 3.6 library for assessment. The corresponding test scores can be found in Table 1. The GPT4 API was selected as the project code generation engine based on the test results and performance/latency ratings [31].

The generation of random numbers was tested with a resistor of 220 Ω, capacitor of 0, 1 mF, and a PWM frequency of 100 Hz, and the resulting output function for each generated value is provided in Figure 5.

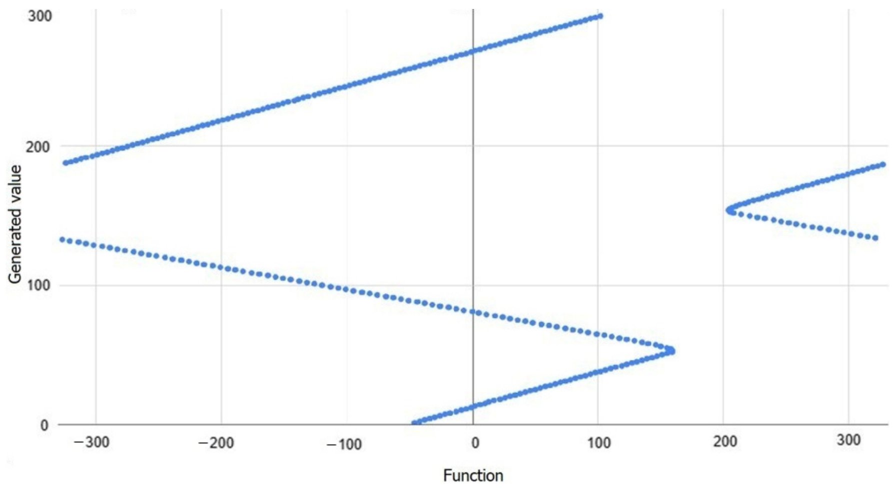


**Figure 5.** Pseudo-random output function.

Values fluctuate rapidly, making time an additional element of unpredictability in the generation of numbers. The statistical randomness test was implemented in MATLAB 2022b with the results of mean: 147.20, standard deviation: 99.47, and chi-squared statistic: 21.68, with a *p*-value: 0.833. The sequence appears to have a random mean. The number of runs suggests randomness. The chi-squared test suggests randomness. The *p*-value is not statistically significant. The frequency test in Figure 6 assesses the occurrences of random numbers to determine their uniform distribution.

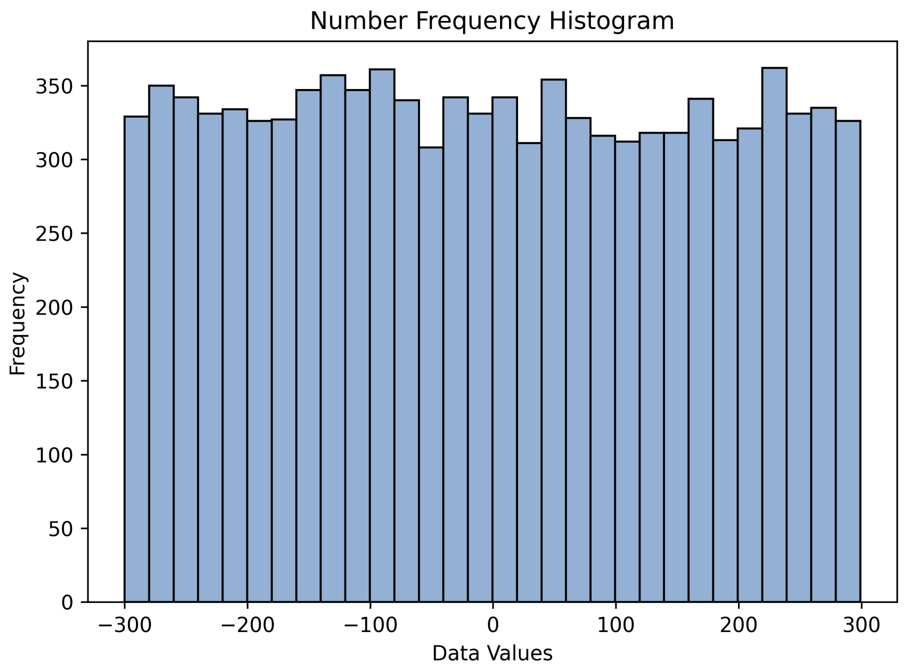


**Figure 6.** Number frequency test results.

The frequency test was conducted utilizing 10,000 data points to minimize the bias. The histogram provides a visual indication of the likely randomness and uniformity of the dataset.

Domain expert rules were used to define the behavior of the IoT system, which was smart heating control, in our case. A fuzzy library (eFLL) was incorporated to facilitate the implementation of fuzzy logic in the microcontroller. It is designed with ease of use in mind, providing a simple and accessible high-level interface for the development and operation of fuzzy models on Arduino-compatible microcontrollers.

The subsequent step involves the implementation of the fuzzy model within the Arduino environment, utilizing the eFLL library. This process requires the development of code that integrates both input and output, establishes the necessary rules, and executes the fuzzy inference mechanism. The eFLL library offers a range of functions to facilitate this process, enabling the developer to concentrate on the actual implementation of the model.

Upon implementation of the fuzzy model, it was executed on the Arduino microcontroller, enabling its output to facilitate system control. Specifically, in the context of heating regulation, the outputs derived from the fuzzy model were utilized to modulate the heating level in response to temperature and presence inputs. Numerous iterations were carried out to read the inputs, generate outputs, and provide verbal explanations, along with benchmarks measuring the duration required to complete a single control cycle iteration. The results are illustrated in Figure 7, which presents a graph detailing the time taken for each iteration.

Nine rules were assessed during the experiment, with each rule providing distinct explanations to facilitate user comprehension regarding the rationale behind the decisions made. The experiment documented twelve iterations, during which time was meticulously measured. The shortest iteration lasted 39 milliseconds, while the longest lasted 74 milliseconds to conclude. Given that the experiment was conducted on an Arduino Uno operating at a clock speed of 16 MHz, the results were deemed satisfactory, particularly as the reaction time remained below one second, ensuring an instantaneous response to the user.

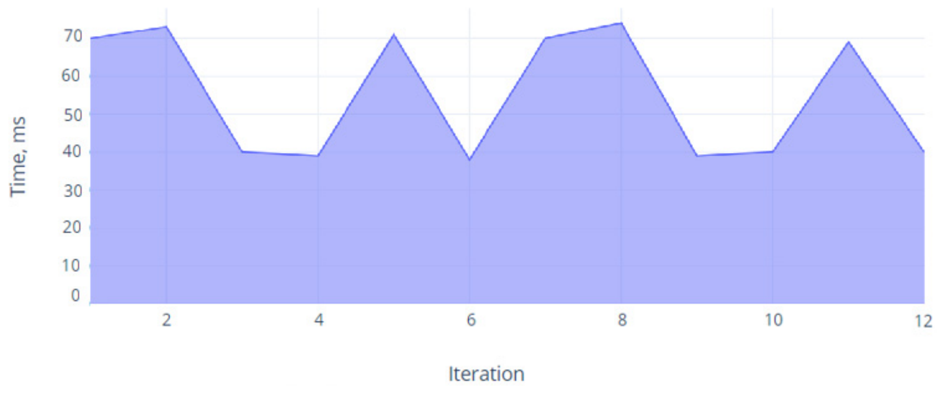


**Figure 7.** Time measurements of iterations for the control cycle.

## 5. Discussion

Communication security has been enhanced through the implementation of hardware-based chaos-based pseudo-random number generation. This improvement was achieved without incurring additional computational costs by using cheap hardware components—a PWM with RC circuit. The PWM with RC circuit was also effectively tested on ESP232 WROOM and Raspberry Pi Nano devices.

The expert toolkit enables experts to create and modify rules for IoT systems without coding while allowing them to review and validate the generated code. This human-in-the-loop approach enhances the interpretability and understanding of the system's decision-making process. The code that is generated accurately reflects the logic inherent in the rules, and the LLM is further instructed to deliver explanations, which improves explainability. Integrating explainability into smart heating control systems helps users understand the reasoning behind AI-powered decisions, promoting confidence in these applications.

Figure 8 demonstrates that the relative efficiency of code generation versus manual coding depends significantly on the user's level of proficiency. Code generation rules offer substantial advantages for novice users, providing up to a 67% improvement in time efficiency. However, as expertise grows, these benefits become less pronounced. Despite this, rule-based code generation continues to save time for intermediate and expert users by 58% and 50%, respectively, facilitating the rapid delivery of deployable code.

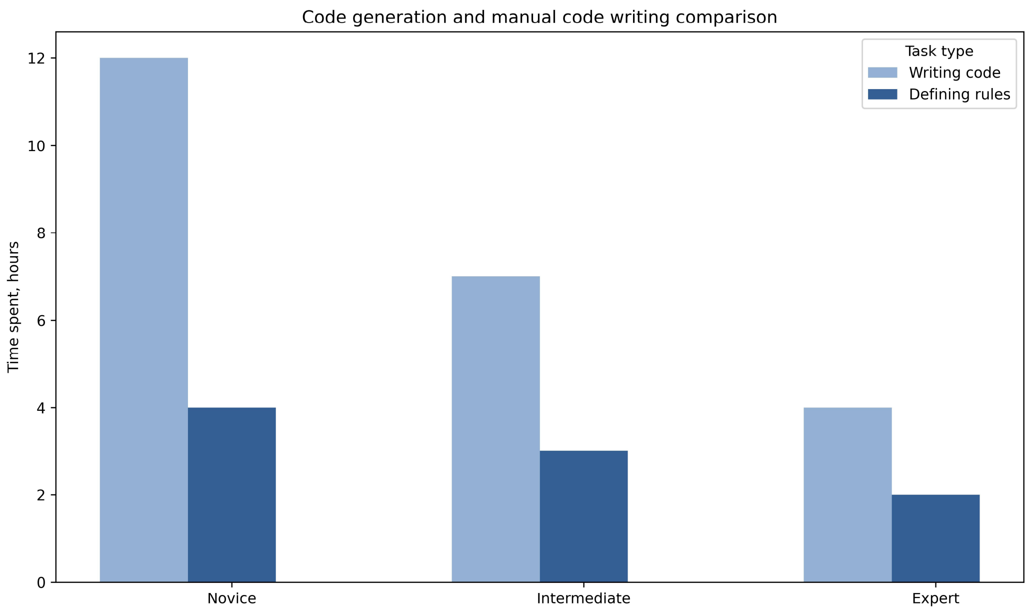


**Figure 8.** Code generation and manual code writing comparison.

One of the key challenges in adopting the expert toolkit is the learning curve associated with mastering its features. Effective use requires adequate training or prior familiarity with similar tools. Additionally, there may be resistance to changing established workflows or legacy systems, which can slow adoption. Ultimately, choosing the most suitable code development approach depends on the user's expertise. Novices may benefit the most from the efficiency of rule-based generation, while more experienced users may prefer the flexibility of manual coding.

The eFLL software (https://github.com/alvesoaj/eFLL) library and explanatory interface code are designed to be compatible with microcontrollers featuring architectures comparable to Arduino, such as ESP232 WROOM and Raspberry Pi Nano. This compatibility allows for more complex IoT scenarios beyond smart heating control, including ventilation and lighting control, where both automation and explainability are crucial.

Future research will be focused on the following:

- Creating a structured approach to utilize verbal explanations across different IoT platforms.
- Evaluating the effectiveness and safety of these systems.
- Comparison of the effectiveness of the proposed method with traditional methods of deploying the IoT system.

**Author Contributions:** Conceptualization, E.K.; Methodology, A.D.; Software, A.D.; Investigation, A.D.; Writing—original draft, A.D.; Writing—review and editing, E.K.; Supervision, E.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Economic Revitalization and Resilience Enhancement Plan "New Generation Lithuania" as part of the execution of the "Mission-driven Implementation of Science and Innovation Programmes" project (No. 02-002-P-0001).

**Data Availability Statement:** The data used to support the findings of this study are available from the corresponding author upon request.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Zolanvari, M.; Yang, Z.; Khan, K.; Jain, R.; Meskin, N. TRUST XAI: Model-agnostic explanations for AI with a case study on IoT security. *IEEE Internet Things J.* **2023**, *10*, 2967–2978.
2. Ilyas, B.; Raouf, S.M.; Abdelkader, S.; Camel, T.; Said, S.; Lei, H. An Efficient and Reliable Chaos-Based IoT Security Core for UDP/IP Wireless Communication. *IEEE Access* **2022**, *10*, 49625–49656. [CrossRef]
3. Clemente-Lopez, D.; de Jesus Rangel-Magdaleno, J.; Muñoz-Pacheco, J.M. A lightweight chaos-based encryption scheme for IoT healthcare systems. *Internet Things* **2024**, *25*, 101032. [CrossRef]
4. Naif, J.R.; Abdul-Majeed, G.H.; Farhan, A.K. Secure IOT System Based on Chaos-Modified Lightweight AES. In Proceedings of the 2019 International Conference on Advanced Science and Engineering (ICOASE), Zakho—Duhok, Iraq, 2–4 April 2019; pp. 1–6. [CrossRef]
5. Monani, R.; Rogers, B.; Rezaei, A.; Hedayatipour, A. Implementation of Chaotic Encryption Architecture on FPGA for On-Chip Secure Communication. In Proceedings of the 2022 IEEE Green Energy and Smart System Systems (IGESSC), Long Beach, CA, USA, 7–8 November 2022; pp. 1–6. [CrossRef]
6. Fei, Y.; Lixiang, L.; Qiang, T.; Shuo, C.; Yun, S.; Xu, Q. A Survey on True Random Number Generators Based on Chaos. *Discret. Dyn. Nat. Soc.* **2019**, *2019*, 2545123. [CrossRef]
7. Kök, I.; Okay, F.Y.; Muyanlı, Ö.; Özdemir, S. Explainable Artificial Intelligence (XAI) for Internet of Things: A Survey. *IEEE Internet Things J.* **2023**, *10*, 14764–14779. [CrossRef]
8. Sobin, C. A survey on architecture, protocols and challenges in IoT. *Wireless Pers. Commun.* **2020**, *112*, 1383–1429.
9. Shah, S.H.; Yaqoob, I. A survey: Internet of Things (IoT) technologies, applications and challenges. In Proceedings of the 2016 IEEE Smart Energy Grid Engineering (SEGE), Oshawa, ON, Canada, 21–24 August 2016; pp. 381–385.
10. Kishor, A.; Chakraborty, C. Artificial intelligence and Internet of Things based healthcare 4.0 monitoring system. *Wireless Pers. Commun.* **2021**, *127*, 1615–1631.

11. Ngu, A.H.; Gutierrez, M.; Metsis, V.; Nepal, S.; Sheng, Q.Z. IoT middleware: A survey on issues and enabling technologies. *IEEE Internet Things J.* **2017**, *4*, 1–20.
12. Ghosh, A.; Chakraborty, D.; Law, A. Artificial intelligence in Internet of Things. *CAAI Trans. Intell. Technol.* **2018**, *3*, 208–218.
13. Mukhopadhyay, S.C.; Tyagi, S.K.S.; Suryadevara, N.K.; Piuri, V.; Scotti, F.; Zeadally, S. Artificial intelligence-based sensors for next generation IoT applications: A review. *IEEE Sens. J.* **2021**, *21*, 24920–24932.
14. Ali, S.; Abuhmed, T.; El-Sappagh, S.; Muhammad, K.; Alonso-Moral, J.M.; Confalonieri, R.; Guidotti, R.; Del Ser, J.; Díaz-Rodríguez, N.; Herrera, F. Explainable artificial intelligence (XAI): What we know and what is left to attain trustworthy artificial intelligence. *Inf. Fusion* **2023**, *99*, 101805. [CrossRef]
15. Silva, P.V.B.C.; Taconet, C.; Chabridon, S.; Conan, D.; Cavalcante, E.; Batista, T. Energy awareness and energy efficiency in internet of things middleware: A systematic literature review. *Ann. Telecommun.* **2023**, *78*, 115–131. [CrossRef]
16. Rojat, T.; Puget, R.; Filliat, D.; Del Ser, J.; Gelin, R.; Díaz-Rodríguez, N. Explainable artificial intelligence (XAI) on time-series data: A survey. *arXiv* **2021**, arXiv:2104.00950.
17. Ahmed, I.; Jeon, G.; Piccialli, F. From artificial intelligence to eXplainable artificial intelligence in industry 4.0: A survey on what, how, and where. *IEEE Trans. Ind. Inform.* **2022**, *18*, 5031–5042. [CrossRef]
18. Javed, A.R.; Ahmed, W.; Pandya, S.; Maddikunta, P.K.R.; Alazab, M. A survey of explainable artificial intelligence for smart cities. *Electronics* **2023**, *12*, 1020. [CrossRef]
19. Hoang, N.X.; Hoang, N.V.; Du, N.H.; Huong, T.T.; Tran, K.P.; Hoang, N.V. Explainable anomaly detection for industrial control system cybersecurity. *IFAC-PapersOnLine* **2022**, *55*, 1183–1188.
20. Confalonieri, R.; Coba, L.; Wagner, B.; Besold, T.R. A historical perspective of explainable artificial intelligence. *Wiley Interdiscip. Rev. Data Min. Knowl. Disc.* **2021**, *11*, e1391. [CrossRef]
21. Ohana, J.J.; Ohana, S.; Benhamou, E.; Saltiel, D.; Guez, B. Explainable AI (XAI) models applied to the multi-agent environment of financial markets. In Proceedings of the International Workshop on Explainable, Transparent AI and Multi-Agent Systems, EXTRAAMAS 2021, Virtual Event, 3–7 May 2021; pp. 189–207.
22. Druce, J.; Harradon, M.; Tittle, J. Explainable artificial intelligence (XAI) for increasing user trust in deep reinforcement learning driven autonomous systems. *arXiv* **2021**, arXiv:2106.03775.
23. Arrieta, A.B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* **2020**, *58*, 82–115. [CrossRef]
24. Kamath, U.; Liu, J. *Explainable Artificial Intelligence: An Introduction to Interpretable Machine Learning*; Springer: Cham, Switzerland, 2021.
25. Ribeiro, O.; Gomes, L.; Vale, Z. IoT-based human fall detection system. *Electronics* **2022**, *11*, 592. [CrossRef]
26. Singh, P.P.; Khosla, P.K.; Mittal, M. Energy Conservation in IoT-Based Smart Home and Its Automation. In *Energy Conservation for IoT Devices*; Mittal, M., Tanwar, S., Agarwal, B., Goyal, L., Eds.; Studies in Systems; Decision and Control; Springer: Singapore, 2019; Volume 206. [CrossRef]
27. He, Y.; Wang, D.; Huang, F.; Zhang, R.; Min, L. Aerial-Ground Integrated Vehicular Networks: A UAV-Vehicle Collaboration Perspective. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 5154–5169. [CrossRef]
28. Yao, Y.; Tian, Y.; Li, X.; Yang, X.; Zhao, B.; Yang, Y. Q-Learning Based MEP Search Algorithm and Coverage Enhancement Strategy in IoT-Enabled Intrusion Detection. *IEEE Sens. J.* **2024**, *24*, 2180–2193. [CrossRef]
29. Yusuf, I.N.B.; Jamal, D.B.A.; Jiang, L.; Lo, D. RecipeGen++: An automated trigger action programs generator. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022); Association for Computing Machinery: New York, NY, USA, 2022; pp. 1672–1676. [CrossRef]
30. Breve, B.; Cimino, G.; Deufemia, V. Hybrid Prompt Learning for Generating Justifications of Security Risks in Automation Rules. *ACM Trans. Intell. Syst. Technol.* **2024**, *15*, 103. [CrossRef]
31. LLM Leaderboard—Comparison of GPT-4o, Llama 3, Mistral, Gemini and over 30 Models. Available online: https://artificialanalysis.ai/leaderboards/models (accessed on 26 February 2025).