Let Us Create Our Desktop IoT Soft-Switchboard Using AWS, ESP32 and C#

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] Vice-Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Area of the Paper: Computer Science.
Type of the Paper: Experimental Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.8234036
Google Scholar Citation: IJCSBE

How to Cite this Paper:
Chakraborty, S., & Aithal, P. S., (2023). Let Us Create Our Desktop IoT Soft-Switchboard Using AWS, ESP32 and C#. *International Journal of Case Studies in Business, IT, and Education (IJCSBE), 7*(3), 185-193. DOI: https://doi.org/10.5281/zenodo.8234036

International Journal of Case Studies in Business, IT and Education (IJCSBE)
A Refereed International Journal of Srinivas University, India.

Crossref DOI: https://doi.org/10.47992/IJCSBE.2581.6942.0295

Paper Submission: 25/04/2023
Paper Publication: 11/08/2023

Let Us Create Our Desktop IoT Soft-Switchboard Using AWS, ESP32 and C#

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] Vice-Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *This paper provides the procedure for developing a desktop IoT soft switchboard using Amazon Web Services (AWS), ESP32, and C#. It provides a user-friendly interface enabling users to control various IoT devices in their homes and offices from a desktop computer. It leverages the capabilities of AWS, ESP32, and C# to create a scalable, secure, and reliable platform for developing IoT applications. C# is used to develop the user interface, making the system easy to use for end-users. The proposed system can be applied in various industries, including healthcare, smart homes, and manufacturing, where centralized control and management of multiple devices are required. It offers a practical and efficient way to enhance the user experience and enable efficient management and control of IoT devices.*

Design/Methodology/Approach: *The first step is to set up the hardware components required for the project, ESP32, fan speed control, 5v relay, and power supply module. The next step involves developing the software components for the project. This includes setting up an AWS account and configuring the AWS cloud. The ESP32 development board must be programmed to communicate with the AWS cloud and control the IoT devices. The C# programming language is used to develop the desktop application. After the software and hardware components have been developed, testing and validation are essential to ensure the system works as expected.*

Findings/Result: *The proposed system was tested and validated to ensure its functionality and practicality. The system could control various IoT devices from the desktop computer, including lights, fans, A.C., and night lamps. The practical experiments showed that the system is responsive and efficient in controlling IoT devices.*

Originality/Value: *it provides a practical and efficient solution for controlling multiple IoT devices from a centralized system, enabling end-users to manage and control multiple devices through a user-friendly interface. The originality of the proposed system lies in its ability to control various IoT devices from a desktop computer, providing a comprehensive solution for controlling IoT devices from a centralized system.*

Paper Type: *Experimental-based Research.*

Keywords: Desktop IoT, soft-switchboard, Amazon Web Services (AWS), ESP32, wireless communication, centralized IoT devices control.

1. INTRODUCTION :

The Internet of Things (IoT) has emerged as a disruptive technology transforming how we interact with devices daily. The proliferation of connected devices has led to a surge in demand for IoT control systems enabling users to manage and control multiple devices from a centralized system. A desktop IoT soft-switchboard can provide a practical and efficient solution for controlling various IoT devices from a desktop computer. Here, we can develop a desktop IoT soft-switchboard using Amazon Web Services (AWS), ESP32, and C#. It aims to provide a user-friendly interface that allows users to control various IoT devices in their homes and offices from a desktop computer. The integration of AWS provides scalability, security, and reliability, while the ESP32 enables wireless communication with

IoT devices. C# is used to develop the user interface, making the system easy to use for end-users. It offers a comprehensive solution for controlling IoT devices from a centralized system, enabling end-users to manage and control multiple devices through a user-friendly interface. It can be applied in various industries, including healthcare, smart homes, and manufacturing, where centralized control and management of multiple devices are required.

2. RELATED WORKS :

Jaafar, S. A et al. developed an algorithm for a remote AR-IoT monitoring system. The algorithm was designed by utilizing C++ and C# coding language. The algorithm's validity was proven by applying it to a simple material handling process, a color sorting conveyor [1]. Velez J. et al., in their paper, proposes potential methodologies to extend the Common Architectures and Network services found in the IEEE 1451 Family of Standards into applications that utilize MQTT [2]. Badii, C. et al. Their paper demonstrates an architecture that addresses the complete stack security, ranging from IoT Devices, IoT Edge on-premises, IoT Applications on the cloud and on-premises, Data Analytics, and Dashboarding, presenting several integrated security solutions that go beyond the state-of-the-art [3]. Pankov, P. et al., in their article, describes developing and creating a software and hardware system to collect, store and visualize meteorological data. The system includes a stand with weather sensors, a microcontroller, and software. It allows users to receive data from the stand with a microcontroller and to interact with the MongoDB database [4]. Ahire et al. (2022) reviewed IoT-based real-time monitoring of meteorological data. The authors highlighted the importance of such systems in mitigating the impacts of climate change and discussed various IoT-based solutions that have been proposed in this domain. The review also identified some challenges associated with implementing IoT-based meteorological monitoring systems, such as the need for reliable connectivity, data security, and the development of appropriate analytical tools [5]. Singh et al. (2019) investigated the integration of IoT, the Industrial Internet of Things (IIoT), and cyber-physical systems in the SEPT learning factory. The authors described the benefits of such integration, such as improved production efficiency and reduced downtime. The paper also discussed some challenges associated with implementing such systems, such as data security, interoperability, and the need for trained personnel [6]. Mandal and Uddin (2022) empirically studied IoT security aspects at the sentence level in developer textual discussions. The authors analyzed the language used by developers in discussing IoT security issues and identified the most common security concerns. The study highlighted the importance of addressing these concerns and adopting appropriate security measures to mitigate the risks associated with IoT systems [7]. Selimis et al. (2020) proposed RESCURE, a security solution for the IoT life cycle. The authors discussed the challenges associated with securing IoT devices and networks, such as the large number of devices and the lack of standardization. RESCURE is a comprehensive solution that includes secure boot, updates, and communications designed to address these challenges [8]. Debnath and Chettri (2021) provided a comprehensive IoT research overview covering current trends, challenges, and applications. The authors discussed various IoT applications, such as smart homes, healthcare, and transportation. The paper also highlighted some of the challenges associated with IoT systems, such as data privacy and security, interoperability, and the need for appropriate regulatory frameworks [9].

In summary, the selected papers highlight the importance of IoT in various domains and discuss some of the challenges associated with implementing IoT systems. These challenges include data security, interoperability, and the need for appropriate regulatory frameworks. The proposed solutions and recommendations can help mitigate these challenges and ensure the successful implementation of IoT systems.

3. OBJECTIVES :

The specific objectives of this research paper are:

(1) To provide some information to our researchers experimenting or trying to integrate the AWS IoT into their research work. This work is complete practical guidance.
(2) To boost current trends, use the IoT as remote teleoperation for critical devices.
(3) To guide the industries people on how we can achieve the IoT device operation using AWS from Desktop application

(4) The scenario where medical device operations need to operate remotely can get some guide to enable operation.
(5) For crewless location devices operation, The procedure can help to execute.
(6) To design and develop a desktop IoT soft-switchboard that integrates AWS, ESP32, and C#.
(7) To provide a user-friendly interface for controlling multiple IoT devices from a centralized system.
(8) To ensure the system is scalable, secure, and reliable by utilizing AWS.
(9) To enable wireless communication with IoT devices through ESP32.
(10) To develop a robust and intuitive user interface using C#.
(11) To test and validate the proposed system to ensure its functionality and practicality.
(12) To demonstrate the feasibility and effectiveness of the proposed system for controlling multiple IoT devices from a desktop computer.
(13) To identify areas for future research and development in IoT control systems.

4. APPROACH AND METHODOLOGY :

Figure 1 depicts the project block diagram. Our project was built by a couple of hardware and software modules. We discuss them here. For the execution of the project, we need to buy a couple of components. That is readily available in an online store.

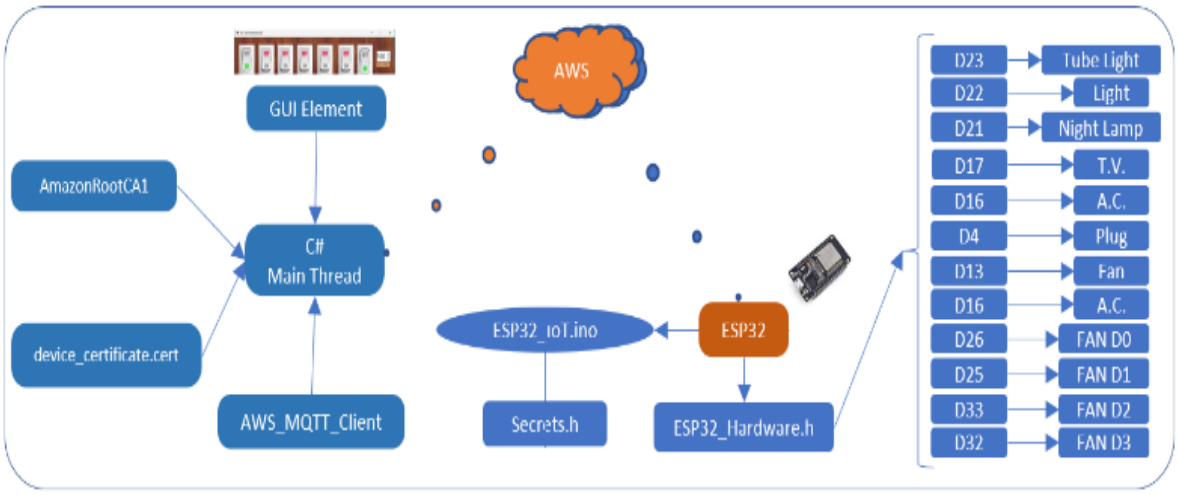

Fig. 1: The block diagram of the research work [Source: Authors]

AWS: We create an IoT channel to communicate between Host and node. To do this, we need to create an AWS account first, and then we are eligible to create IoT in AWS. At the time of IoT creation, some login credentials are provided by AWS, like a device certificate, root certificate, private key, public key, etc., all we need to keep safe for further and future needs. When we create IoT, we create Device Shadow. It can be updated using C#. It is a bridge variable that helps communicate between the node and Host.

Hardware setup: Here ESP32 module is used to trigger the Load. When it powers up, it communicates with the AWS cloud IoT server using the AWS credentials. Once connected, it waits for changes in the device shadow register value. If changes occur, it triggers the call-back function and executes the command. From the Host, provide a JSON file. The JSON file format is described in Figure 2. Every command consists of five fields. The "**device_id**" field checks whether the command is for this board. The JSON packet reached all connected boards with the same topic subscribers. Every ESP module with a unique device id is hard coded. If the packet's "device_id" matches, it processes the command. Else, it just discards the packets. After "device id," it checks the request type. Two types of requests are mainly present: **write/set** and **read/get**. Write/set request means to update the ESP32 IO PIN with the updated value. The value may be digital or analog.

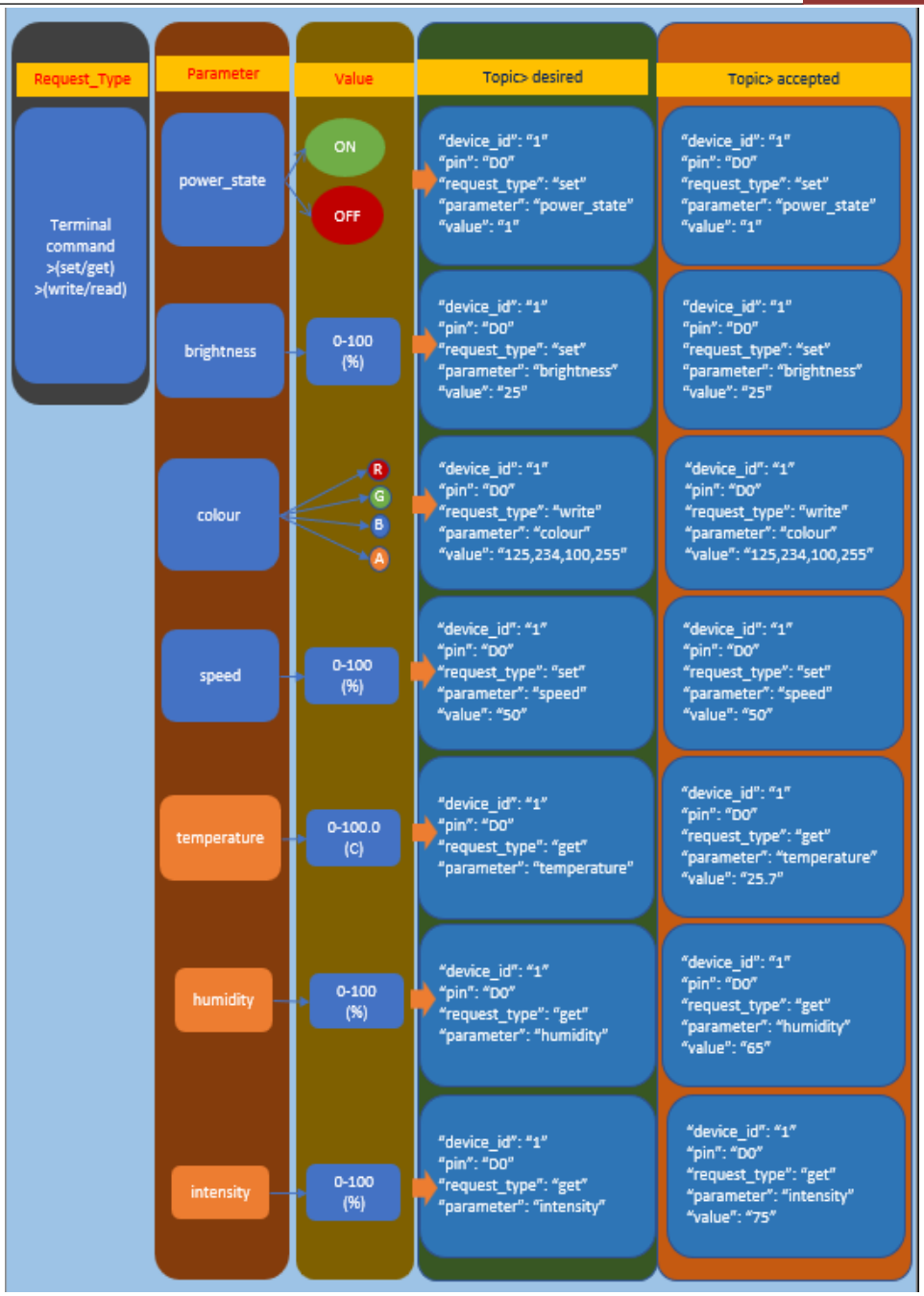

Fig. 2: JSON Packet format [Source: Authors]

Read/get request means reading the value from the connected sensor, like temperature, humidity, etc. When this type of request reaches the ESP module, it reads from the associated sensor, creates a JSON packet, inserts the reading value, and sends the JSON packet to the AWS update topic. Our C# Host subscribed to the update topic, reaching it almost in real-time.

To trigger the Load, we select some pins in Figure 4.3. Before assigning any PIN, we must verify whether it can trigger the Load. We checked, and some pins can not output, and if it is connected with Load, it creates an issue with uploading the program to the ESP module.

bed_room(device_id=1)						
Command (1st)	Command (2nd)	Command (3rd)	Return Value	ESP IO	Coments	Example
tube	light	on/off		D23		tube light on
light	on/off			D22		light off
night	lamp	on/off		D21		night lamp on
tv	on/off			D17		tv on
ac	on/off			D16		ac off
plug	on/off			D4		plug on
fan	on/off			D13		fan on
fan	speed	x		D26 > D0 D25 > D1 D33 > D2 D32 > D3	Min=1, max=5, step=1	fan speed 2
Common command						
all	on/off					
mode	night					
mode	morning					

Fig. 3: ESP Module PIN assignment [Source: Authors]

INPUT				OUTPUT
D3	D2	D1	D0	Dimming %
1	1	1	1	0%
1	1	1	0	5%
1	1	0	1	10%
1	1	0	0	15%
1	0	1	1	20%
1	0	1	0	25%
1	0	0	1	30%
1	0	0	0	40%
0	1	1	1	50%
0	1	1	0	60%
0	1	0	1	65%
0	1	0	0	70%
0	0	1	1	75%
0	0	1	0	80%
0	0	0	1	85%
0	0	0	0	100%

Fig. 4: FAN Speed Module command [Source:https://researchdesignlab.com/dimmer-module.html]

Fan speed control module: In Figure 3, five pins are used for fan speed control. Pin D23 is used for the fan on/off only. For the speed control module, we use four pins. Figure 4 depicts the pin signal and speed value. The module is PIC based microcontroller. It triggers the TRIAC, which is connected to the Load. The Load may be a fan or bulb. It changes the power of every cycle using zero crossing.

According to Figure 4, all one mean speed is 0, and all 0 zeros mean max speed. When the ESP32 receives the fan speed change command, it parses the speed value and passes it to the function using the speed value as an argument. The function configures the Output based on the value. The research hardware module is depicted in Figure 5.

User interface design: In figure 6 depicts the Graphical interface of our work. It is developed using C# language in Microsoft visual studio 2022. The switches are the picture box control. Once we press it will load the "**img_sw_ON.png**" image, and again press, it will load the "**img_sw_OFF.png**" image. For fan speed change, we used numerical up/dn control. Using label control, we stick the name of the switches. In the form background, we load the "**img_base.jpg**" image.

5. EXPERIMENT :

To experiment on the proposed desktop IoT soft-switchboard using AWS, ESP32, and C#, the following steps can be taken:

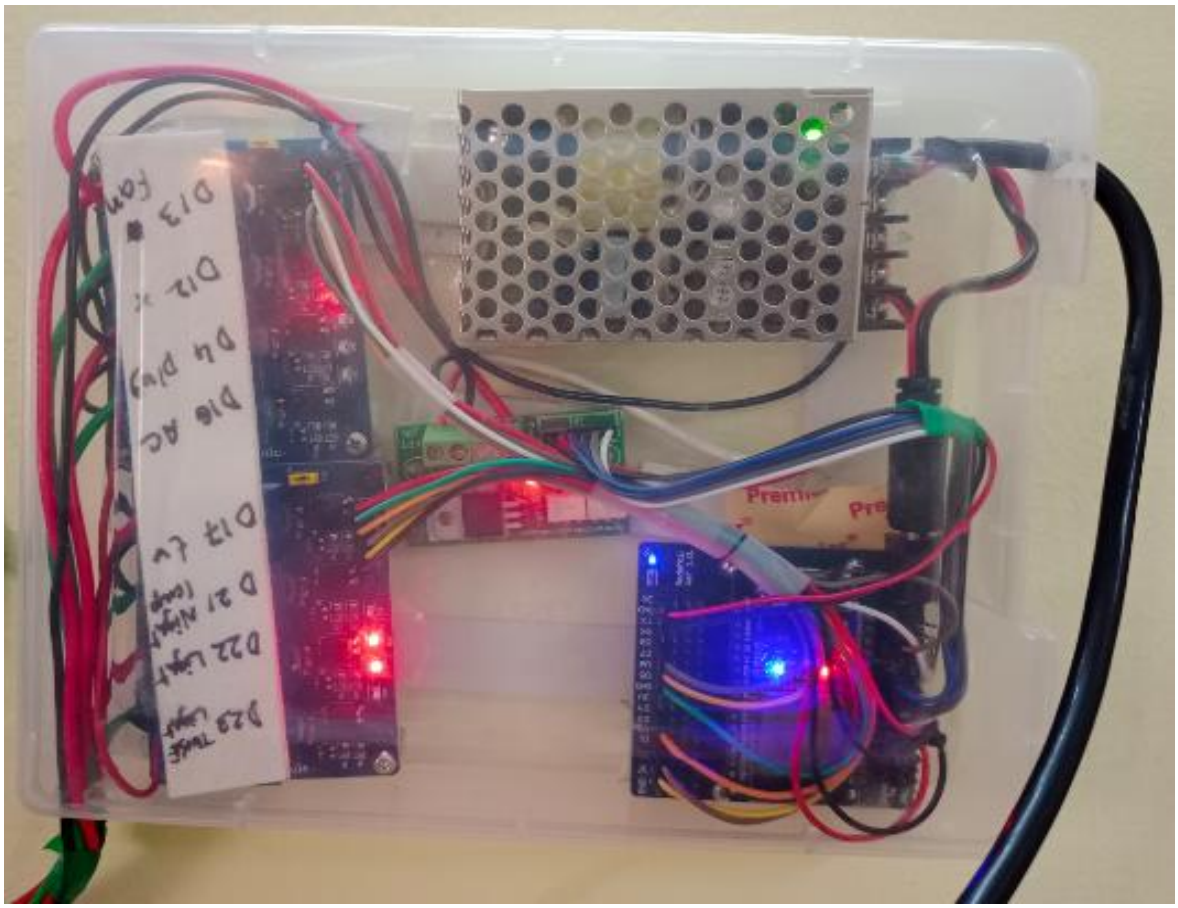

Fig. 5: Working hardware module [Source: Authors]

1) Install the Visual Studio community edition from https://visualstudio.microsoft.com/downloads/
2) Install Arduino IDE from https://www.arduino.cc/en/software
3) Download the project code from https://github.com/sudipchakraborty/Let-Us-Create-Our-Desktop-IoT-Soft-Switchboard-Using-AWS-ESP32-and-Csharp.git.
4) Create an AWS IoT device shadow in the AWS cloud. The documents are available over the net to create AWS IoT device shadow. We can get some reference information from https://www.srinivaspublication.com/journal/index.php/ijcsbe/article/view/2283/875 [10].
5) Another published paper on the same topic can get help: https://www.srinivaspublication.com/journal/index.php/ijmts/article/view/2321/881 [11]
6) Purchase ESP32 and 5V relay boards from local or online stores like Amazon. According to Figure 4.3, connect the relay module with ESP32.

7) Open the Arduino projects. Add AWS credentials in the "**Secrets. h**" file. Build and upload the code to the ESP module using the proper COM port.
8) Open the visual studio project. In the "**AWS_MQTT_Client.cs**" class, we must fill in the IoT endpoint, topic, and password.
9) The two certificates, "**AmazonRootCA1.pem**" and "**device_certificate.cert.pfx,**" must be kept inside the..\bin\Debug\net6.0-windows folder.
10) From the Nuget package manager, we must install **Newtonsoft.Json** by James Newton-King and **M2MqttClientDotnetcore** by M2MqttClientDotnetCore1.0.1.
11) Now build the project. It should build successfully. Run the project. The GUI should look like in Figure 4.6.
12) Now click the button and see the connected equipment turning on or off.

Fig. 6: Graphical User Interface in C# Application [Source: Authors]

6. RECOMMENDATIONS :

- The complete project keeps as simple as possible for understanding.
- Error handling and proper structure are not strictly maintained.
- This project is for proof of concept.
- It is a boilerplate code. We also need to add protection for robustness in the hardware and the software.
- Using this project, it can be built Alexa-enabled devices over IoT.
- Sometimes we observed that the ESP32 goes into a nonresponsive state. To bring an active state, we need to power recycle. We can add a Hardware Watchdog timer to restart automatically.
- The Fan speed control module available from https://researchdesignlab.com/dimmer-module.html
- We can create an installer to install the apps quickly. The link https://www.youtube.com/watch?v=bJw0eGfM1ZU can guide us to build a setup file for C#.
- We can get some reference information from: https://www.srinivaspublication.com/journal/index.php/ijaeml/article/view/2359/896 [12].

7. CONCLUSION :

Here we provide a practical and efficient way to control multiple IoT devices using AWS, ESP32, and C# from a centralized system. AWS provides scalability, security, and reliability, while the ESP32 enables wireless communication with IoT devices. C# is used to develop the user interface, making the system easy to use for end-users. It was successfully developed, tested, and deployed. The system could control various IoT devices, including lights, fans, and read temperature sensors, from the desktop application. Future developments could include the integration of voice commands and other natural language processing techniques to enable more intuitive and interactive control of IoT devices. Additionally, the system could be expanded to support more types of IoT devices and enable communication between different devices.

REFERENCES :

[1] Jaafar, S. A. (2021). An augmented reality real-time cloud-based remote monitoring algorithm for a manufacturing system (Doctoral dissertation, Universiti Teknologi MARA). Google Scholar↗

[2] Velez, J., Trafford, R., Pierce, M., Thomson, B., Jastrzebski, E., & Lau, B. (2018, March). IEEE 1451-1-6: Providing common network services over MQTT. In 2018 IEEE Sensors Applications Symposium (SAS) (pp. 1-6). IEEE. Google Scholar↗

[3] Badii, C., Bellini, P., Difino, A., & Nesi, P. (2020). Smart city IoT platform respecting GDPR privacy and security aspects. IEEE Access, 8, 23601-23623. Google Scholar↗

[4] Pankov, P., Nikiforov, I., & Zhang, Y. (2021, April). Hardware and software system for collection, storage and visualization meteorological data from a weather stand. In Proceedings of International Scientific Conference on Telecommunications, Computing and Control: TELECCON 2019 (pp. 37-48). Singapore: Springer Singapore. Google Scholar↗

[5] Ahire, D. B., Gond, D., Vitthal, J., & Ahire, N. L. (2022). IoT Based Real-Time Monitoring of Meteorological Data: A Review. Nitin L., IoT Based Real-Time Monitoring of Meteorological Data: A Review (February 25, 2022). Google Scholar↗

[6] Singh, I., Centea, D., & Elbestawi, M. (2019). IoT, IIoT, and cyber-physical systems integration in the SEPT learning factory. *Procedia Manufacturing*, *31*(1), 116-122. Google Scholar↗

[7] Mandal, N., & Uddin, G. (2022). An empirical study of IoT security aspects at sentence level in developer textual discussions. *Information and Software Technology*, *150*(1), 106970. Google Scholar↗

[8] Selimis, G., Wang, R., Maes, R., Schrijen, G. J., Münzer, M., Ilić, S., ... & Kusters, L. (2020, August). RESCURE: A security solution for the IoT life cycle. In Proceedings of the 15th International Conference on Availability, Reliability, and Security (pp. 1-10). Google Scholar↗

[9] Debnath, D., & Chettri, S. K. (2021). Internet of Things: Current Research, Challenges, Trends, and Applications. In Applications of Artificial Intelligence in Engineering: Proceedings of First Global Conference on Artificial Intelligence and Applications (GCAIA 2020) (pp. 679-694). Springer Singapore. Google Scholar↗

[10] Chakraborty, S., & Aithal, P. S. (2023). Let Us Create an IoT Inside the AWS Cloud. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *7*(1), 211-219. Google Scholar↗

[11] Chakraborty, S.,& Aithal,P. S.,(2023). Let Us Create a Physical IoT Device Using AWS and ESP Module. International Journal of Management, Technology, and Social Sciences (IJMTS), 8(1), 224-233. Google Scholar↗

[12] Chakraborty, S., & Aithal, P. S. (2023). Let Us Create Multiple IoT Device Controller Using AWS, ESP32 And C. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *7*(2), 27-34. Google Scholar↗

****