Let Us Create Multiple IoT Device Controller 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] Professor, Institute of Management & Commerce, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Subject Area: 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.7857660
Google Scholar Citation: IJAEML

How to Cite this Paper:
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. DOI: https://doi.org/10.5281/zenodo.7857660

International Journal of Applied Engineering and Management Letters (IJAEML)
A Refereed International Journal of Srinivas University, India.

Crossref DOI: https://doi.org/10.47992/IJAEML.2581.7000.0172

Received on: 04/03/2023
Published on: 24/04/2023

Let Us Create Multiple IoT Device Controller 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] Professor, Institute of Management & Commerce, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *The Internet of Things (IoT) has revolutionized how we interact with our environment by allowing various devices to connect and communicate. In this paper, we propose the development of a multiple IoT device controller using Amazon Web Services (AWS), ESP32, and C# programming language. The proposed system will allow users to control and monitor multiple IoT devices simultaneously through a centralized platform. The ESP32, a low-cost wifi module, will interface with the IoT devices and transmit data to the AWS IoT Core. The C# programming language will be used to develop the user interface and handle user requests. The proposed multiple IoT device controller using AWS, ESP32, and C# programming language is expected to provide a practical solution for managing and controlling multiple IoT devices, improving user experience, and advancing IoT technology. The code used in this paper is ready to download for continuing the research work.*

Design/Methodology/Approach: *The proposed system uses AWS IoT Core to manage the devices and their data. The ESP32 module connects to the IoT devices and sends/receives data to/from AWS IoT Core. The device controller application is developed using C# programming language to control the IoT devices. AWS IoT Core manages IoT devices and their data. The IoT devices are registered with AWS IoT Core, and their unique identifiers are stored in the AWS IoT Core registry. The ESP32 module is programmed to connect to the wifi network using the wifi module. This enables the module to access the internet and connect to AWS IoT Core.*

Findings/Result: *We developed a multiple-device controller using AWS IoT and ESP32 modules here. We created a single-channel IoT in AWS and broadcast it to all devices. Every message consists device id. All devices will receive the message but are responsible only for a specific device. For multicast messages, the response is by multiple devices. The c# application is the master; all ESP32 devices are clients.*

Originality/Value: *Most net documents are dedicated to the IoT device creation procedure. Here we demonstrate the complete example, i.e., how to create IoT Devices in the AWS server, the node devices, and control from the C# application. So the interested researcher can get complete information to integrate IoT into their project.*

Paper Type: *Experimental-based Research.*

Keywords: AWS IoT using ESP32, Smart Home using AWS, ESP Module, AWS IoT Shadow Devices, C# AWS IoT client.

1. INTRODUCTION :

The Internet of Things (IoT) has revolutionized how we interact with devices daily. The increasing number of connected devices has increased the demand for IoT device management systems. AWS has emerged as a leading provider of cloud-based IoT services, providing a scalable and secure platform for IoT device management. ESP32 is a popular low-cost Wi-Fi chip that can be easily integrated with IoT devices. C# is a popular programming language for desktop and mobile application development. This research paper proposes the development of a multiple IoT device controller using AWS, ESP32, and C#. The proposed system will allow users to monitor and control multiple IoT devices from a single platform. The system will utilize AWS IoT services for device management, ESP32 for device connectivity, and C# for application development. The primary objective of this research is to provide a comprehensive solution for managing multiple IoT devices from a single platform. The proposed system will be designed to be scalable, reliable, and secure. The system will allow users to add new devices, monitor device status, and control device operations remotely. The paper is structured as follows. Section 2 provides an overview of the related work. Section 3 presents the proposed system architecture. Section 4 describes the implementation details of the system. Section 5 discusses the results of the system evaluation. Finally, Section 6 concludes the paper and provides future directions for the proposed system. This research contributes to developing an efficient and effective solution for managing multiple IoT devices using AWS, ESP32, and C#. The proposed system can potentially improve the management and control of IoT devices in various industries, including home automation, industrial automation, and healthcare.

2. RELATED WORKS :

Gupta, Khera, and Turk (2021) proposed an MQTT protocol-based IoT home safety system using Attribute-Based Encryption (ABE) [1]. The study focused on ensuring data security and privacy for IoT devices in a home environment. Similarly, Froiz-Míguez et al. (2018) designed and evaluated an IoT home automation system for fog computing applications, employing MQTT and ZigBee-WiFi sensor nodes [2]. The study demonstrated the efficiency of the proposed system in terms of energy consumption, latency, and reliability. Uddin et al. (2021) conducted an empirical study on IoT topics discussed by developers on Stack Overflow [3]. Singh, Roy, and Singh (2022) proposed a serverless IoT architecture for intelligent waste management systems [4]. The study highlighted the advantages of employing IoT technology in waste management, including cost reduction, improved efficiency, and environmental sustainability. Pruna and Vasilescu (2020) introduced FitPi, a wearable IoT solution for daily innovative life applications [5]. FitPi aimed to enhance the user experience by seamlessly integrating various services, such as health monitoring, social networking, and personal productivity. Alam et al. (2020) presented a low-cost IoT-based weather station for real-time monitoring [6]. The study emphasized the importance of meteorological data for various applications, including agriculture, disaster management, and climate studies. Ahire et al. (2022) provided a comprehensive review of IoT-based real-time meteorological monitoring systems, discussing the challenges and opportunities in this field [7]. The proposed system aimed to optimize water usage in agriculture by employing sensors and actuators for real-time monitoring and control. Freitas (2021) proposed a low-cost IoT monitoring solution to increase student awareness on campus [8]. The study aimed to explore the potential of IoT technologies for enhancing campus life by providing real-time data on energy consumption, air quality, and other environmental factors.

3. OBJECTIVES :

This research paper aims to demonstrate the development of a multiple IoT device controller using AWS, ESP32, and C#. The proposed solution lets users control and monitor multiple IoT devices from a single platform. The AWS IoT platform provides a scalable, secure, and reliable cloud infrastructure to store and process data from IoT devices. Using the MQTT protocol, the ESP32 microcontroller connects the physical devices to the AWS cloud. The C# programming language is used to develop a user-friendly interface for controlling and monitoring IoT devices. The proposed solution can potentially enhance the efficiency of managing multiple IoT devices and reduce the complexity of their integration.

4. APPROACH AND METHODOLOGY :

Figure 1 depicts the project block diagram. Here we arrange four ESP modules for four different places around our house to convert the smart homes. Sometimes electric switchboard operating by the child is harmful, especially for a wet hand. Here we are making those IoT-enabled.
After installing the system, we must create an IoT device inside the AWS cloud. Download the project code from the git repository. Inside the code, there is a secret.h file. That needs to fill from AWS credentials. We add our Wifi id and password into the specific field. Finally, using a power supply, it will power up and run until the power is on.

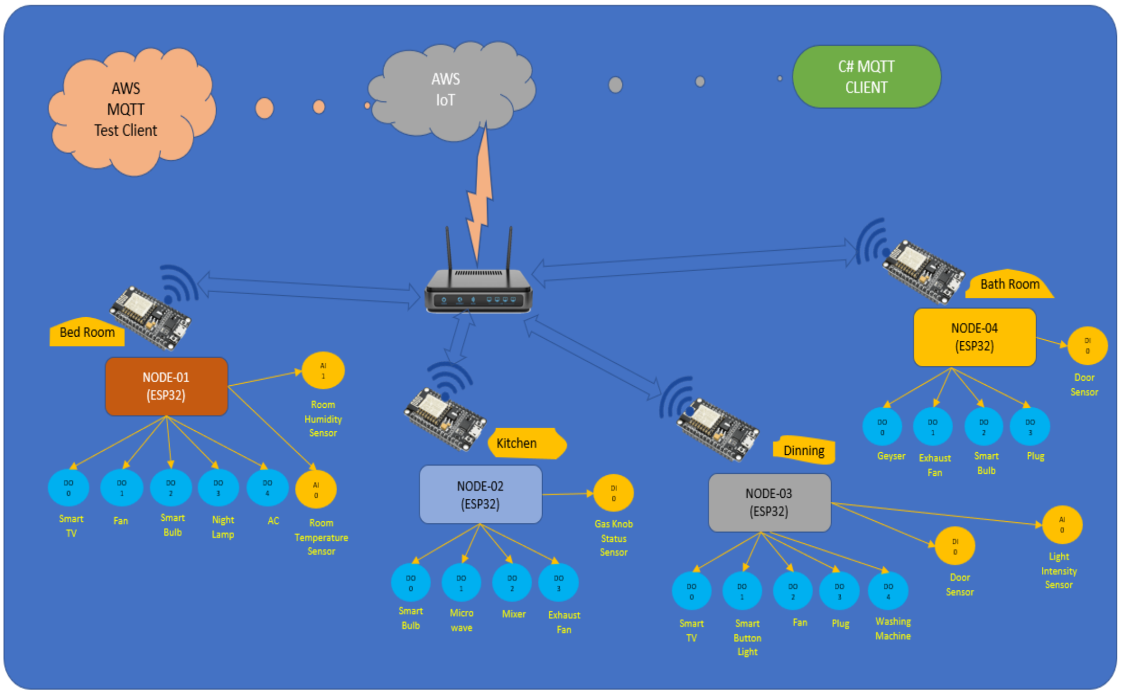
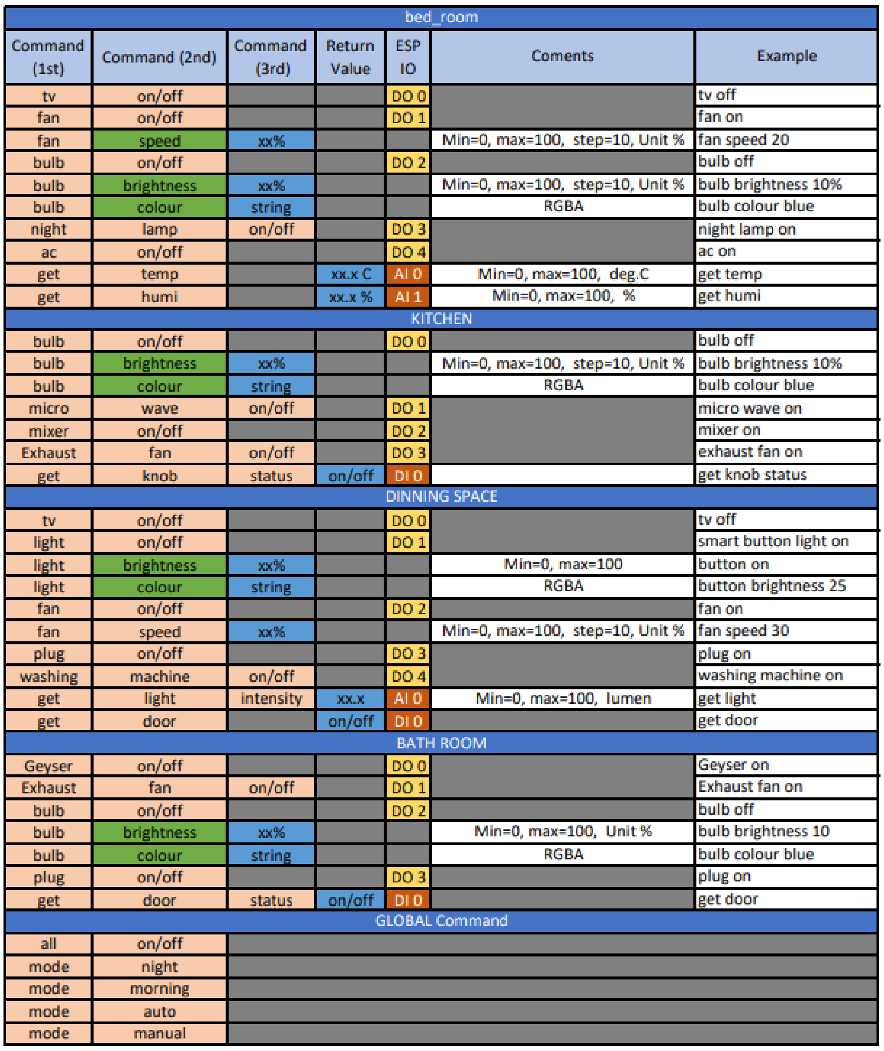


Fig. 1: Project's Block Diagram [Source: Authors]



Command (1st)	Command (2nd)	Command (3rd)	Return Value	ESP IO	Coments	Example
bed_room						
tv	on/off			DO 0		tv off
fan	on/off			DO 1		fan on
fan	speed	xx%			Min=0, max=100, step=10, Unit %	fan speed 20
bulb	on/off			DO 2		bulb off
bulb	brightness	xx%			Min=0, max=100, step=10, Unit %	bulb brightness 10%
bulb	colour	string			RGBA	bulb colour blue
night	lamp	on/off		DO 3		night lamp on
ac	on/off			DO 4		ac on
get	temp		xx.x C	AI 0	Min=0, max=100, deg.C	get temp
get	humi		xx.x %	AI 1	Min=0, max=100, %	get humi
KITCHEN						
bulb	on/off			DO 0		bulb off
bulb	brightness	xx%			Min=0, max=100, step=10, Unit %	bulb brightness 10%
bulb	colour	string			RGBA	bulb colour blue
micro	wave	on/off		DO 1		micro wave on
mixer	on/off			DO 2		mixer on
Exhaust	fan	on/off		DO 3		exhaust fan on
get	knob	status	on/off	DI 0		get knob status
DINNING SPACE						
tv	on/off			DO 0		tv off
light	on/off			DO 1		smart button light on
light	brightness	xx%			Min=0, max=100	button on
light	colour	string			RGBA	button brightness 25
fan	on/off			DO 2		fan on
fan	speed	xx%			Min=0, max=100, step=10, Unit %	fan speed 30
plug	on/off			DO 3		plug on
washing	machine	on/off		DO 4		washing machine on
get	light	intensity	xx.x	AI 0	Min=0, max=100, lumen	get light
get	door		on/off	DI 0		get door
BATH ROOM						
Geyser	on/off			DO 0		Geyser on
Exhaust	fan	on/off		DO 1		Exhaust fan on
bulb	on/off			DO 2		bulb off
bulb	brightness	xx%			Min=0, max=100, Unit %	bulb brightness 10
bulb	colour	string			RGBA	bulb colour blue
plug	on/off			DO 3		plug on
get	door	status	on/off	DI 0		get door
GLOBAL Command						
all	on/off					
mode	night					
mode	morning					
mode	auto					
mode	manual					

Fig. 2: The command details against module PIN details [Source: Authors]

Now we will see the behind the scene works in the process. When the ESP module power up, it connects with the internet via wifi. If the internet connection is successful, then using credentials, try to connect with the AWS IoT cloud server. If it passes all authentication, it is connected to the AWS cloud. The ESP module registers one publisher and one subscriber. Whatever data we send or receive is subscribed to four modules. The matching device will trigger the Load or read the parameter and send it to the primary node, i.e., the c# client. According to the ESP32 software, every module runs the same software. The device id is different. It is hardcoded. Figure 2 depicts how we can assign commands for multiple devices to trigger the loads.

4. EXPERIMENT :

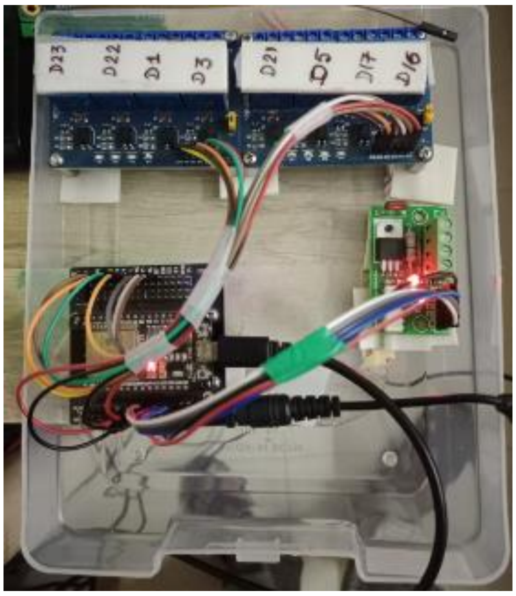


Fig. 3: Prototype [Source: Authors]

Fig. 4: Fan speed controller [Source: https://www.amazon.in/gp/product/B00SFLVFBE/ref=ox_sc_act_image_6?smid=A216LXYAA4QNPY&psc=1]

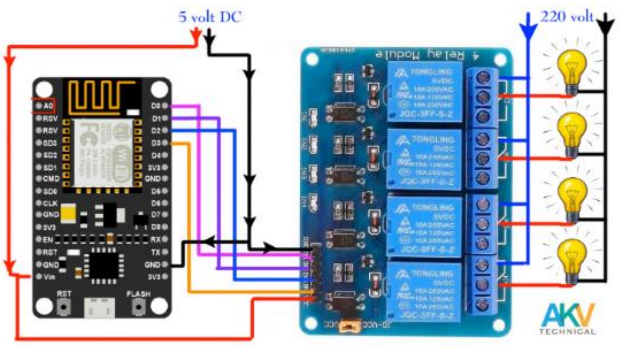
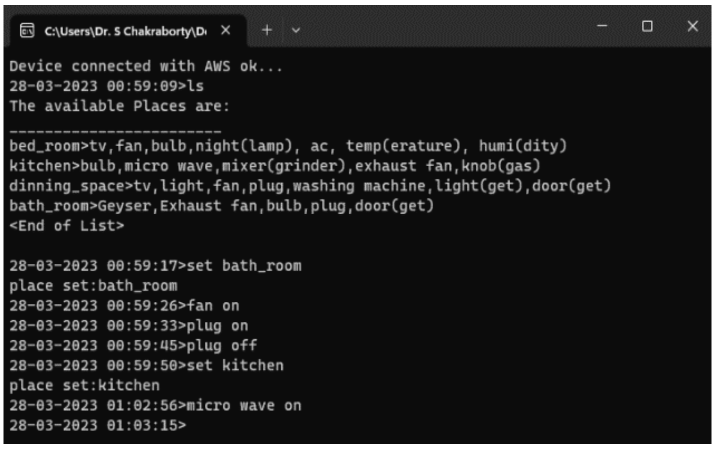


Fig. 5: Relay module connection [Source; https://akvtechnical.com/ESP32-wi-fi-home-automation-system-in-hindi/]

Now we can do some practical experiments. Figure 3 shows the prototype that we experiment with. Follow the below steps to do some hands-on.

1) Purchase ESP32 modules to convert a traditional switchboard into an intelligent switchboard. The ESP module is available from the local or online shop.

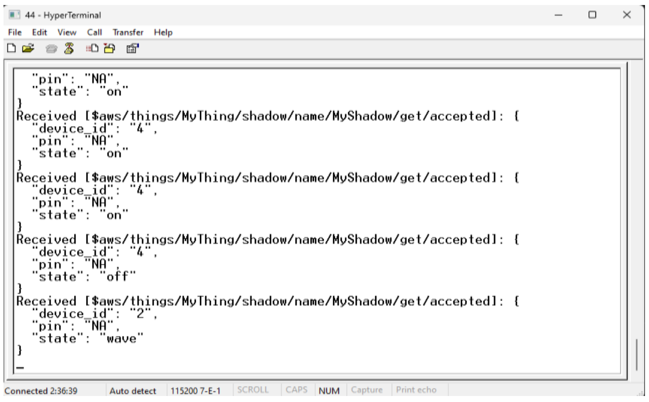
2) For Load on/off, we need a relay module. From One channel to 16 channels, modules are available—purchase according to our needs.
3) With less effort, we can use the module available in AWS to control Fan speed digitally. https://www.amazon.in/gp/product/B00SFLVFBE/ref=ewc_pr_img_1?smid=A216LXYAA4QNPY&psc=1 . figure 4 depicts the fan speed controller module.
4) After receiving all modules, connect the relay module with the ESP module. A sample connection diagram depicts in Figure 5.
5) Now We need to program the ESP32 module. Before programming, Install Visual Studio and Arduino IDE.
6) Download code from https://github.com/sudipchakraborty/Let-Us-Create-Multiple-IoT-Device-Controller-Using-AWS-ESP32-And-C-sharp.git
7) Create an AWS IoT account and IoT devices.
8) Open the ESP32 project code .ino. Add the IoT credential to the project file.
9) Build the project and upload the code to the ESP module.
10) Configure C# using C# AWS credential.
11) Run the C# application and send the command through the terminals.
12) The figure shows the command sent through the C# terminal, and Figure 7 shows the command received in the ESP32 module.



```
Device connected with AWS ok...
28-03-2023 00:59:09>ls
The available Places are:
------------------------
bed_room>tv,fan,bulb,night(lamp), ac, temp(erature), humi(dity)
kitchen>bulb,micro wave,mixer(grinder),exhaust fan,knob(gas)
dinning_space>tv,light,fan,plug,washing machine,light(get),door(get)
bath_room>Geyser,Exhaust fan,bulb,plug,door(get)
<End of List>

28-03-2023 00:59:17>set bath_room
place set:bath_room
28-03-2023 00:59:26>fan on
28-03-2023 00:59:33>plug on
28-03-2023 00:59:45>plug off
28-03-2023 00:59:50>set kitchen
place set:kitchen
28-03-2023 01:02:56>micro wave on
28-03-2023 01:03:15>
```

Fig. 6: Sending command through terminal [Source: Authors]



```
  "pin": "NA",
  "state": "on"
}
Received [$aws/things/MyThing/shadow/name/MyShadow/get/accepted]: {
  "device_id": "4",
  "pin": "NA",
  "state": "on"
}
Received [$aws/things/MyThing/shadow/name/MyShadow/get/accepted]: {
  "device_id": "4",
  "pin": "NA",
  "state": "on"
}
Received [$aws/things/MyThing/shadow/name/MyShadow/get/accepted]: {
  "device_id": "4",
  "pin": "NA",
  "state": "off"
}
Received [$aws/things/MyThing/shadow/name/MyShadow/get/accepted]: {
  "device_id": "2",
  "pin": "NA",
  "state": "wave"
}
_
```

Fig. 7: ESP32 received Command [Source: Authors]

6. RECOMMENDATIONS :

- Four-channel relay module purchase link: https://www.amazon.in/Robotbanao-Optocoupler-Channel-Control-Arduino/dp/B07DZR7SLH/ref=sr_1_3_sspa?crid=33N0S10JOUUP8&keywords=8+channel+relay+module+5v&qid=1679744398&sprefix=8+channel+relay+module+5v%2Caps%2C331&sr=8-3-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1
- https://www.amazon.in/s?k=ESP32+wifi+module&crid=3R1T3TYDE6QV0&sprefix=ESP32%2Caps%2C280&ref=nb_sb_ss_ts-doa-p_3_7
- Install Visual Studio from https://visualstudio.microsoft.com/downloads/
- Install Arduino IDE from https://www.arduino.cc/en/software
- The electrical connection should be under expert supervision, or else it gets an electrical shock. It can be harmful to life risk. Before any electrical work, you need to wire plastics or rubber shoes. It is strictly recommended to the researcher, especially for juniors or unfamiliar with handling electrics.
- Figure 4 adapted from https://akvtechnical.com/ESP32-wi-fi-home-automation-system-in-hindi/
- The c# application keeps it as simple as possible. The various way to keep configuration files like JSON. Etc. Understanding the complete project, the researcher can customize it according to their project need.
- This project is the foundation of Alexa's enabled IoT Device. We can quickly build an Alexa-driven Load of on-off equipment using this project.

7. CONCLUSION :

In conclusion, the project to create multiple IoT device controllers using AWS, ESP32, and C# has proven successful. The AWS cloud platform has played a significant role in the project, providing the infrastructure and services necessary to deploy and manage IoT devices. The ESP32 microcontroller, on the other hand, has provided a low-level hardware and software interface to connect and control IoT devices. C# programming language has been used to develop the graphical user interface (GUI) of the IoT device controller, enabling users to manage and control multiple devices simultaneously easily. This project has also provided a valuable opportunity to explore the capabilities of AWS, ESP32, and C# programming languages in IoT device development.

REFERENCES :

[1] Gupta, V., Khera, S., & Turk, N. (2021). MQTT protocol employing IOT-based home safety system with ABE encryption. *Multimedia Tools and Applications*, *80*(2), 2931-2949. Google Scholar↗

[2] Froiz-Míguez, I., Fernández-Caramés, T. M., Fraga-Lamas, P., & Castedo, L. (2018). Design, implement, and practically evaluate an IoT home automation system based on MQTT and

ZigBee-WiFi sensor nodes for fog computing applications. *Sensors*, *18*(8), 2660. 01-42. Google Scholar↗

[3] Uddin, G., Sabir, F., Guéhéneuc, Y. G., Alam, O., & Khomh, F. (2021). An empirical study of iot topics in iot developer discussions on stack overflow. *Empirical Software Engineering*, *26*, 1-45. Google Scholar↗

[4] Singh, R., Roy, B., & Singh, V. (2022). Serverless IoT Architecture for Smart Waste Management Systems. In *IoT-Based Smart Waste Management for Environmental Sustainability* (pp. 139-154). CRC Press. Google Scholar↗

[5] Pruna, S., & Vasilescu, A. (2020). FitPi: Wearable IoT solution for daily smart life. *International Journal of Advanced Statistics and IT&C for Economics and Life Sciences*, *10*(1), 67-79. Google Scholar↗

[6] Alam, M. J., Rafi, S. A., Badhan, A. A., Islam, M. N., Shuvo, S. I., & Saleque, A. M. (2020, December). Low Cost IoT Based Weather Station for Real-Time Monitoring. In *2020 IEEE 2nd International Conference on Circuits and Systems (ICCS)* (pp. 127-133). IEEE. Google Scholar↗

[7] 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↗

[8] Freitas, L. D. C. (2021). Low cost loT monitoring solution for increased student awareness on campus (Doctoral dissertation). Google Scholar↗

****