WWJMRD 2025; 11(04): 39-44
www.wwjmrd.com
International Journal
Peer Reviewed Journal
Refereed Journal
Indexed Journal
Impact Factor SJIF 2017: 5.182 2018: 5.51, (ISI) 2020-2021: 1.361
E-ISSN: 2454-6615

Sudip Chakraborty
D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore, India.

Deep Chakraborty
MCKV Institute of Engineering, Howrah, West Bengal, India.

Correspondence:
Sudip Chakraborty
D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore, India.

Alexa-Enabled Smart Home Using Echo Dot, Alexa Cloud Service, Custom Intent, Static IP, Wi-Fi Router, Local Web Server, Lambda Function, and ESP32

Sudip Chakraborty, Deep Chakraborty

Abstract
This research paper presents an innovative approach to smart home automation through an Alexa-enabled system utilizing Echo Dot, Alexa Cloud Service, Custom Intents, Static IP addressing, a Wi-Fi Router, a Local Web Server, AWS Lambda Functions, and an ESP32 microcontroller. The proposed system seamlessly integrates voice-controlled smart home operations by leveraging Amazon's Alexa ecosystem, enabling intuitive interactions and responsive device control. A Custom Intent model is developed to accurately interpret user commands, while AWS Lambda Functions facilitate reliable cloud-to-device communication. An ESP32 microcontroller acts as the central device interface, coordinating actions based on instructions received through a secured Local Web Server connected via a static IP address over a Wi-Fi Router. The system architecture ensures low latency, enhanced security, and robust performance, making it suitable for practical residential applications. Experimental evaluations demonstrate efficient, accurate, and user-friendly automation capabilities, highlighting the feasibility and effectiveness of integrating IoT hardware with cloud-based voice services for advanced smart home solutions.

Keywords: Smart Home, Alexa, Echo Dot, AWS Lambda, ESP32, IoT, Voice Control, Custom Intent, Static IP, Wi-Fi Router, Local Web Server.

1. Introduction
The proliferation of smart home technologies has fundamentally transformed residential living environments by providing increased comfort, efficiency, and convenience. Among various emerging technologies, voice-controlled systems have gained particular popularity due to their intuitive and hands-free interaction capabilities. Amazon Alexa stands out as a prominent voice assistant technology, offering robust integration capabilities for diverse home automation applications. Despite significant advancements, challenges such as latency, security, and effective integration of IoT hardware with cloud services persist.
This paper examines the development and implementation of an Alexa-enabled smart home solution designed to address these challenges effectively. The system utilizes components such as an Echo Dot for voice command input, Alexa Cloud Service for voice processing, and AWS Lambda Functions for secure, real-time data communication. A customized intent model is developed to ensure accurate interpretation and execution of user commands. Furthermore, the incorporation of an ESP32 microcontroller connected via a Wi-Fi Router and a Local Web Server with a Static IP ensures low-latency, secure, and reliable interaction with home appliances. The integration of these components provides users with a seamless, responsive, and secure smart home experience. This study contributes to the ongoing development in smart home automation by offering insights into practical solutions and identifying critical areas for future improvement.

2. Literature Review
Several studies have explored the implementation of Alexa-enabled smart home systems. Somesh, Senthilnathan, and Sabarimuthu [1] discussed real-time control of home appliances using Alexa, highlighting the ease of integration with existing household devices. Similarly,

Mahadik, Jain, and Chavan [2] presented a home automation framework utilizing Alexa to enhance user convenience and energy efficiency. Arya and Patel [3] extended this concept by implementing both Google Assistant and Amazon Alexa on a Raspberry Pi, providing comparative insights into their capabilities.
Swain et al. [4] showcased a case study on Alexa skill development for managing academic attendance systems, demonstrating the practical applicability of custom Alexa skills in various contexts. Security considerations have also been prominent; Hu et al. [5] investigated the vetting process for smart-home assistant applications, and Gautam [6] analyzed user perceptions regarding Alexa's privacy policies. Lei et al. [7] further emphasized security vulnerabilities inherent to digital voice assistants, proposing enhanced safety measures.
Yan et al. [8] and Liao et al. [9] examined privacy policies for Alexa skills, emphasizing accessibility, effectiveness, and GDPR compliance. Additionally, Ding et al. [10] explored vulnerabilities in voice-controlled systems within connected vehicles, while Apthorpe et al. [11] proposed methods for maintaining smart home privacy through intelligent IoT traffic management.
Further foundational research includes Cook et al. [13], who introduced the agent-based smart home MavHome, and Pal et al. [14], who analyzed the adoption and diffusion of voice-enabled systems. These studies collectively provide a comprehensive understanding of the current state of Alexa-enabled smart home technologies, highlighting opportunities and challenges for future developments.

3. Methodology

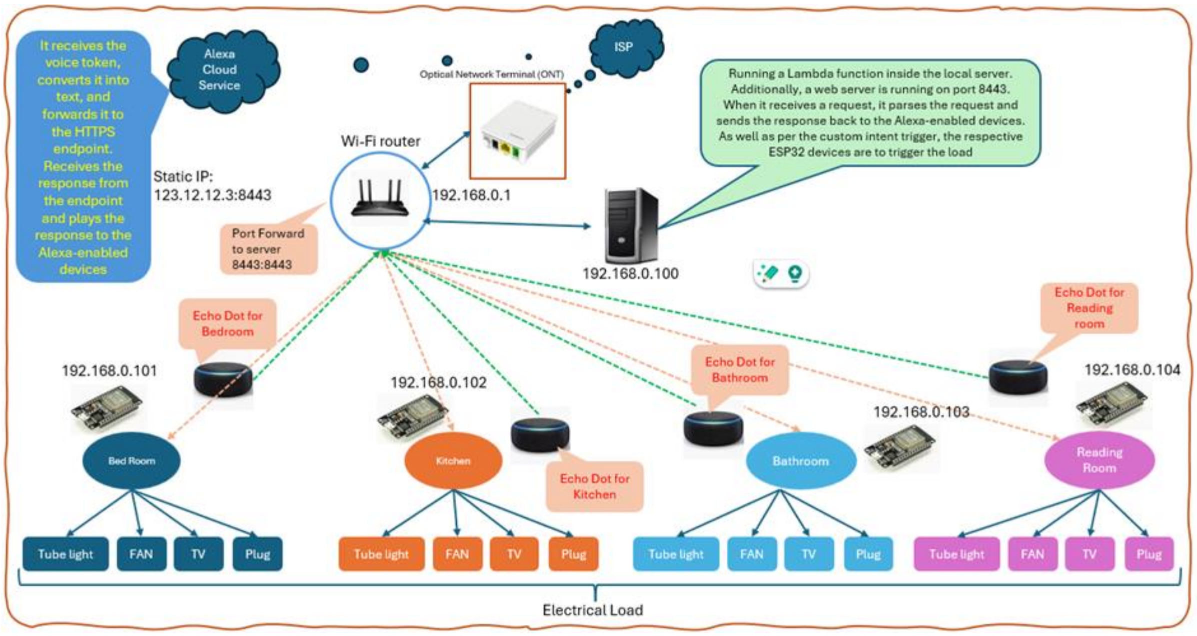

It receives the voice token, converts it into text, and forwards it to the HTTPS endpoint. Receives the response from the endpoint and plays the response to the Alexa-enabled devices
Alexa Cloud Service
Optical Network Terminal (ONT)
ISP
Running a Lambda function inside the local server. Additionally, a web server is running on port 8443. When it receives a request, it parses the request and sends the response back to the Alexa-enabled devices. As well as per the custom intent trigger, the respective ESP32 devices are to trigger the load
Wi-Fi router
Static IP: 123.12.12.3:8443
192.168.0.1
Port Forward to server 8443:8443
192.168.0.100
Echo Dot for Bedroom
192.168.0.101
192.168.0.102
Echo Dot for Bathroom
Echo Dot for Reading room
192.168.0.104
192.168.0.103
Bed Room
Kitchen
Echo Dot for Kitchen
Bathroom
Reading Room
Tube light
FAN
TV
Plug
Electrical Load

Fig. 1: Project Blockdiagram.

System Architecture

Figure 1 illustrates a robust voice-controlled smart home ecosystem powered by Alexa, ESP32, and a local web server with a static IP. The entire system is designed for real-time control and automation of various electrical loads across multiple rooms using voice commands.
At the core of the system lies the Alexa Cloud Service, which receives voice input from Echo Dot devices. It converts the voice into a structured request and forwards it to a secure HTTPS endpoint defined by a Static IP (123.12.12.3:8443). This endpoint is configured through port forwarding on the Wi-Fi router (192.168.0.1) to redirect requests to a local server operating at IP 192.168.0.100 and listening on port 8443.
The local server is configured to host a Lambda-like function that processes the incoming requests from Alexa. This includes parsing the requests, resolving custom intents, and deciding which ESP32 microcontroller (associated with a specific room) should respond. Upon determining the correct action, the server issues commands to the ESP32 devices across the home network.
Each room, Bedroom, Kitchen, Bathroom, and Reading Room is equipped with:

- An ESP32 microcontroller (e.g., 192.168.0.101, 192.168.0.102...)
- An Echo Dot device
- Connected electrical loads (Tube Light, FAN, TV, Plug)

These ESP32s receive command signals from the local server and trigger their respective electrical loads. The green and orange dashed lines represent the communication pathways between devices and the server, ensuring seamless coordination.
This design ensures:

- Secure, real-time command execution using HTTPS
- Fully offline LAN-based operation post Alexa token-to-request conversion
- Simplified local control via port-forwarded static IP

The architecture is both scalable and secure, effectively integrating voice commands, local decision-making, and IoT-based physical control in a smart home setting.
The provided diagram illustrates the architecture of the proposed Alexa-enabled smart home system. At the top, the Alexa Cloud Service is connected to the system via the Internet Service Provider (ISP). The ISP interfaces with a local network that comprises a Wi-Fi router and a local web server with static IP addresses. This configuration ensures

secure and reliable communication.

Within the home network, Echo Dot devices and ESP32 microcontrollers are strategically positioned across various rooms, including the Bedroom, Kitchen, Bathroom, and Reading Room. Each room's Echo Dot and ESP32 microcontroller interface directly with local electrical loads, such as tube lights, fans, televisions, and plug points. Green and orange dashed lines represent wireless communication between Echo Dot devices, the ESP32 modules, and the Wi-Fi router, emphasizing a robust and interconnected wireless network.

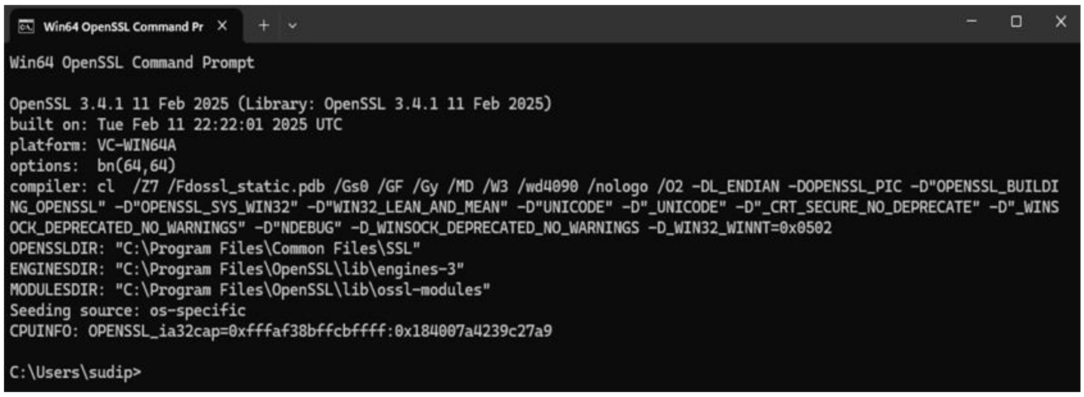
This architecture effectively supports real-time control and monitoring of household appliances, providing a user-friendly voice-controlled automation experience.

Create Custom Intent

1) Download repository from: https://github.com/sudipchakraborty/Alexa_Handler_HTTPS.git.
2) Unzip it. Inside the folder, there is a PDF file called "Alexa's Voice Command (Intent) Customization"
3) According to the instructions, create a custom intent within the Alexa Developer Console.
4) Add Endpoint. It looks like: https://111.123.34.56:8443/api/v1/webhook-alexa
5) Save it.

Create Certificate

1) Download and install OpenSSL software from: https://slproweb.com/products/Win32OpenSSL.html
2) from the taskbar search box, enter openssl and click on "Win64 OpenSSL Command Prompt". It will open the windows as below:



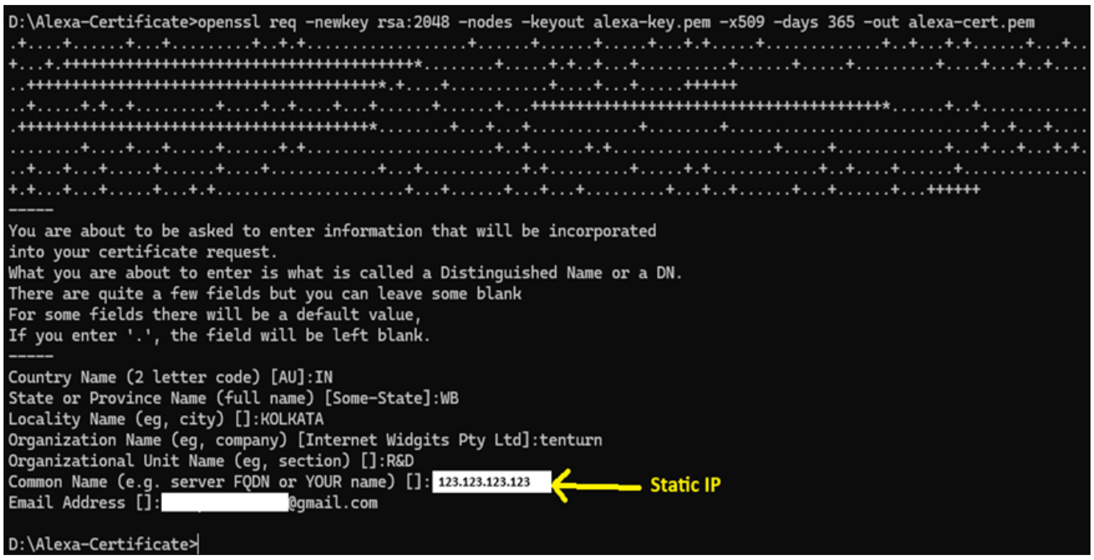
```
Win64 OpenSSL Command Prompt

OpenSSL 3.4.1 11 Feb 2025 (Library: OpenSSL 3.4.1 11 Feb 2025)
built on: Tue Feb 11 22:22:01 2025 UTC
platform: VC-WIN64A
options:  bn(64,64)
compiler: cl  /Z7 /Fdossl_static.pdb /Gs0 /GF /Gy /MD /W3 /wd4090 /nologo /O2 -DL_ENDIAN -DOPENSSL_PIC -D"OPENSSL_BUILDI
NG_OPENSSL" -D"OPENSSL_SYS_WIN32" -D"WIN32_LEAN_AND_MEAN" -D"UNICODE" -D"_UNICODE" -D"_CRT_SECURE_NO_DEPRECATE" -D"_WINS
OCK_DEPRECATED_NO_WARNINGS" -D"NDEBUG" -D_WINSOCK_DEPRECATED_NO_WARNINGS -D_WIN32_WINNT=0x0502
OPENSSLDIR: "C:\Program Files\Common Files\SSL"
ENGINESDIR: "C:\Program Files\OpenSSL\lib\engines-3"
MODULESDIR: "C:\Program Files\OpenSSL\lib\ossl-modules"
Seeding source: os-specific
CPUINFO: OPENSSL_ia32cap=0xfffaf38bffcbffff:0x184007a4239c27a9

C:\Users\sudip>
```

Fig. 2: OpenSSL window.

3. Create a folder anywhere in the system and navigate. Here I created a folder in the D drive "Alexa-Certificate"
4. C:\Users\sudip>d:
5. D:\>cd Alexa-Certificate
6. D:\Alexa-Certificate>
7. Now type or paste the command:**openssl req -newkey rsa:2048 -nodes -keyout alexa-key.pem -x509 -days 365 -out alexa-cert.pem**
8. Enter Country Name (2 letter code) [AU]: IN (example)
9. State or Province Name (full name) [Some-State]: WB (example)
10. Locality Name (eg, city) []: KOLKATA (example)
11. Organization Name (eg, company) [Internet Widgits Pty Ltd]: xxxx
12. Organizational Unit Name (eg, section) []: xxx
13. Common Name (e.g. server FQDN or YOUR name) []: 123.12.12.12 (static ip)
14. Email Address []: xyz@xyz.com
15. Press Enter, and a certificate will be created inside the folder.
16. The OpenSSL interface appears as shown in Figure 3.



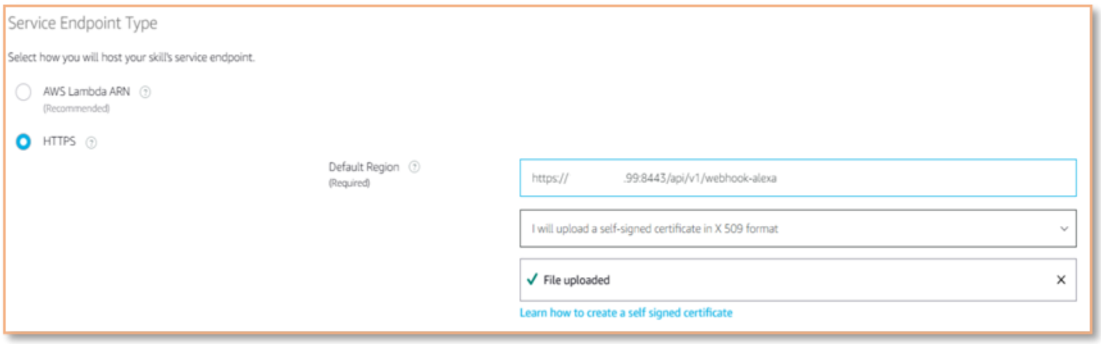
```
D:\Alexa-Certificate>openssl req -newkey rsa:2048 -nodes -keyout alexa-key.pem -x509 -days 365 -out alexa-cert.pem
.+...+.....+..+.......+..+.+...............+.....+.....+....+..+.+....+...........+..+..+.+.....+..+..
+...+.+++++++++++++++++++++++++++++++++++++++*.......+....+.+..+..+.........+.....+....+........+....+...+..+....
..+++++++++++++++++++++++++++++++++++++++*.+....+..........+...+...+.....++++++
..+.....+.+..+.........+....+..+....+...+......+......+...++++++++++++++++++++++++++++++++++++++++*.....+..+..........
.+++++++++++++++++++++++++++++++++++++++*.......+...+...+...........+.......+......................+..+...+....
.......+...+...+.....+.....+.+..................+..+.....+.+................+....+...........+...+...+...+.+.
..+...+...+.....+......+....+...........+...+..........+.+........+.....+.+...........+..+...+.....+...........
+.+...+...+.....+...+.+.................+...+.....+...+...+........+...+..+......+...+.....+...++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:WB
Locality Name (eg, city) []:KOLKATA
Organization Name (eg, company) [Internet Widgits Pty Ltd]:tenturn
Organizational Unit Name (eg, section) []:R&D
Common Name (e.g. server FQDN or YOUR name) []:123.123.123.123
Email Address []:[redacted]@gmail.com

D:\Alexa-Certificate>
```

Fig. 3: OpenSSL Interface.

Upload Certificate to the Alexa developer console:

1) Open the Alexa console https://developer.amazon.com/alexa/console/ask
2) Go to "your skill". Click on the created skill. In my case, "my lab"
3) From the left side menu bar, click on "Endpoint"
4) From the right side, select "HTTPS".
5) In the Default Region textbox, enter a static IP like this: https://xxx.xx.xx.xx:8443/api/v1/webhook-alexa
6) In the second box, select "I will upload a self-signed certificate in X 509 format"
7) In the third box, click to upload the certificate. Click the third box and select the " Alexa-cert" file, and click the "open" button. One notification will appear on the screen, indicating that the file has been successfully uploaded. The page will look like the one below:



Service Endpoint Type

Select how you will host your skill's service endpoint.

AWS Lambda ARN (Recommended)

HTTPS

Default Region (Required): https:// .99:8443/api/v1/webhook-alexa

I will upload a self-signed certificate in X 509 format

✓ File uploaded

Learn how to create a self signed certificate

Fig. 4: Certificate and Endpoint Entry.

8) In the top right corner, click on the "Save" button.

Create and run an Alexa handler locally

1) Download the repository from: https://github.com/sudipchakraborty/Alexa_Handler_HTTPS.git. As zipped, it is 17.6 KB. After unzipping the file, the size is around 46.1 KB.
2) Extract the repo and open it in VS Code.
3) From the certificate folder, copy alexa-cert and alexa-key, and paste them into the project folder, or drag and drop them into the project folder.
4) Open a new terminal. First, we need to install a couple of modules.
5) Enter command: **npm install morgan**
6) Then run a server using the command: **node server.js.**
7) If it is running ok, it will show: HTTPS Server running on https://localhost:8443
8) Press Ctrl+C to stop the server. Modify esp32Command.js according to the custom intent and forward the request to the specific ESP32 IP address.
9) Once the modification is complete, restart the server.
10) Navigate to the Alexa Developer Console. Open created skill. Go to test mode. Enable "Development" mode.
11) Write the command "open" and then the skill invocation name, such as "open my lab". Alexa devices should play the launch intent message. Then test the custom intent messages, such as "door open," "door close," and "bathroom light on," etc.

4. Conclusion

This research has demonstrated the successful implementation of an Alexa-enabled smart home system that utilizes a combination of modern IoT technologies and cloud-based voice services. By integrating the Echo Dot with the Alexa Cloud Service, custom intents, and a secure local network, the system allows users to interact naturally with their environment using voice commands. The architecture's use of a static IP address, a local web server, and a Lambda-style function enables low-latency and secure command processing within a residential network. Furthermore, the deployment of ESP32 microcontrollers in different rooms, each connected to essential electrical appliances, enables precise, room-specific automation and control.

Through the design and evaluation of this architecture, the system has demonstrated scalability, security, and user-friendliness. It effectively addresses common challenges such as latency, network dependency, and the limitations of commercial plug-and-play solutions. The use of open-source tools, such as OpenSSL and Node.js, along with clear documentation for certificate handling and custom intent creation, enhances the reproducibility and accessibility of the system, facilitating broader adoption.

Overall, the project contributes valuable insights to the field of smart home automation by presenting a modular, customizable, and robust solution. Future work can explore machine learning-driven automation, voice recognition personalization, and extended support for energy monitoring. As smart homes become more mainstream, this architecture provides a practical foundation for researchers, developers, and consumers to build more intelligent and responsive living environments.

Github Link: https://github.com/sudipchakraborty/Alexa_Handler_HTTPS.git.

References

1. Somesh, S., Senthilnathan, N., & Sabarimuthu, M. (2020). Real-time Implementation of Home Appliance Control Using Alexa.
2. Mahadik, S., Jain, T., & Chavan, M. (2019). Home Automation Using Alexa.
3. Arya, S. D., & Patel, S. (2021). Implementation of Google Assistant & Amazon Alexa on Raspberry Pi.
4. Swain, K. P., Samal, S. R., Amiri, I. S., et al. (2020). Academic Students Attendance System: A Case Study of Alexa Skill Development.
5. Hu, H., Yang, L., Lin, S., & Wang, G. (2020). Security Vetting Process of Smart-home Assistant Applications: A First Look and Case Studies.
6. Gautam, S. (2020). In Alexa, We Trust. Or Do We? An Analysis of People's Perception of Privacy Policies.

7. Lei, X., Tu, G.-H., Liu, A. X., et al. (2020). The Insecurity of Home Digital Voice Assistants—Amazon Alexa as a Case Study.
8. Yan, J., Liao, S., Aldeen, M., et al. (2020). SKILLPOV: Towards Accessible and Effective Privacy Notice for Amazon Alexa Skills.
9. Liao, S., Aldeen, M., Yan, J., et al. (2020). Understanding GDPR Non-compliance in Privacy Policies of Alexa Skills in European Marketplaces.
10. Ding, W., Liao, S., Guo, K., et al. (2020). Exploring Vulnerabilities in Voice Command Skills for Connected Vehicles.
11. Apthorpe, N., Huang, D. Y., Reisman, D., et al. (2019). Keeping the Smart Home Private with Smarter IoT Traffic Shaping.
12. Jones, V. K. (2018). Voice-Activated Change: Marketing in the Age of Artificial Intelligence and Virtual Assistants.
13. Cook, D. J., Youngblood, M., Heierman, E. O., et al. (2003). MavHome: An Agent-Based Smart Home.
14. Pal, D., Arpnikanondt, C., Funilkul, S., & Razzaque, M. A. (2019). Analyzing the Adoption and Diffusion of Voice-Enabled Smart-Home Systems: Empirical Evidence from Thailand.
15. Khan, S., & Mahadik, S. (2020). A Comparative Study of Agile and Waterfall Software Development Methodologies.
16. Cook, D. J., Huber, M., Gopalratnam, K., & Youngblood, M. (2003). Learning to Control a Smart Home Environment.
17. Liao, S., Wilson, C., Cheng, L., et al. (2019). Measuring the Effectiveness of Privacy Policies for Voice Assistant Applications.
18. Cheng, L., Wilson, C., Liao, S., et al. (2020). Dangerous Skills Got Certified: Measuring the Trustworthiness of Skill Certification in Voice Personal Assistant Platforms.
19. Huang, D. Y., Apthorpe, N., Li, F., et al. (2019). IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale.
20. Major, D., Huang, D. Y., Chetty, M., & Feamster, N. (2019). Understanding Users' Ability to Identify Third-Party Apps on Amazon Alexa.
21. Chakraborty, S., & Aithal, P. S. (2023). Let Us Create an Alexa-Enabled IoT Device Using C#, AWS Lambda and ESP Module. International Journal of Management, Technology, and Social Sciences (IJMTS), 8(3), 256-261. DOI: https://doi.org/10.5281/zenodo.8260291
22. Chakraborty, S., & Aithal, P. S. (2023). Alexa Enabled IoT Device Simulation Using C# And AWS Lambda. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 7(3), 359-368. DOI: https://doi.org/10.5281/zenodo.8329375
23. Chakraborty, S. & Aithal, P. S. (2023). Smart Magnetic Door Lock for Elderly People Using AWS Alexa, IoT, Lambda and ESP Module. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 7(4), 474-483. DOI: https://doi.org/10.5281/zenodo.10467946
24. Chakraborty, S., & Aithal, P. S. (2023). IoT-Based Switch Board for Kids Using ESP Module And AWS. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 7(3), 248-254. DOI: https://doi.org/10.5281/zenodo.8285219
25. Chakraborty, S. & Aithal, P. S. (2024). AI Kitchen. International Journal of Applied Engineering and Management Letters (IJAEML), 8(1), 128-137. DOI: https://doi.org/10.5281/zenodo.10810228
26. Chakraborty, S., & Aithal, P. S. (2023). IoT-Based Industrial Debug Message Display Using AWS, ESP8266 And C#. International Journal of Management, Technology, and Social Sciences (IJMTS), 8(3), 249-255. DOI: https://doi.org/10.5281/zenodo.8250418
27. 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
28. Chakraborty, Sudip, & Aithal, P. S., (2021). An Inverse Kinematics Demonstration of a Custom Robot using C# and CoppeliaSim. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 5(1), 78-87. DOI: http://doi.org/10.5281/zenodo.4755778.
29. Chakraborty, S., & Aithal, P. S., (2023). MVVM Demonstration Using C# WPF. International Journal of Applied Engineering and Management Letters (IJAEML), 7(1), 1-14. DOI: https://doi.org/10.5281/zenodo.7538711
30. Chakraborty, S., & Aithal, P. S. (2023). Let Us Create A Lambda Function for Our IoT Device In The AWS Cloud Using C#. International Journal of Management, Technology, and Social Sciences (IJMTS), 8(2), 145-155. DOI: https://doi.org/10.5281/zenodo.7995727
31. Chakraborty, S., & Aithal, P. S., (2022). How to make IoT in C# using Sinric Pro. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 6(2), 523-530. DOI: https://doi.org/10.5281/zenodo.7335167
32. Chakraborty, S., & Aithal, P. S., (2022). Virtual IoT Device in C# WPF Using Sinric Pro. International Journal of Applied Engineering and Management Letters (IJAEML), 6(2), 307-313. DOI: https://doi.org/10.5281/zenodo.7473766
33. Chakraborty, S. & Aithal, P. S. (2023). Let Us Create an Alexa Skill for Our IoT Device Inside the AWS Cloud. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 7(2), 214-225. DOI: https://doi.org/10.5281/zenodo.7940237
34. Chakraborty, Sudip, & Aithal, P. S., (2021). Forward Kinematics Demonstration of 6DF Robot using CoppeliaSim and C#. International Journal of Applied Engineering and Management Letters (IJAEML), 5(1), 29-37. DOI: http://doi.org/10.5281/zenodo.4680570.
35. 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. DOI: https://doi.org/10.5281/zenodo.7779097
36. 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. DOI: https://doi.org/10.5281/zenodo.7726980
37. 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

38. Chakraborty, Sudip, & Aithal, P. S., (2021). A Custom Robotic ARM in CoppeliaSim. International Journal of Applied Engineering and Management Letters (IJAEML), 5(1), 38-50. DOI: http://doi.org/10.5281/zenodo.4700297.
39. Chakraborty, Sudip, & Aithal, P. S., (2021). Forward and Inverse Kinematics Demonstration using RoboDK and C#. International Journal of Applied Engineering and Management Letters (IJAEML), 5(1), 97-105. DOI: http://doi.org/10.5281/zenodo.4939986.
40. Chakraborty, S., & Aithal, P. S., (2022). A Practical Approach To GIT Using Bitbucket, GitHub and SourceTree. International Journal of Applied Engineering and Management Letters (IJAEML), 6(2), 254-263. DOI: https://doi.org/10.5281/zenodo.7262771
41. Chakraborty, S. & Aithal, P. S. (2024). WhatsApp Based Notification on Low Battery Water Level Using ESP Module and TextMeBOT. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 8(1), 291-309. DOI: https://doi.org/10.5281/zenodo.10835097
42. Chakraborty, S. & Aithal, P. S. (2024). Go Green: ReUse LED Tube Light and Make it WhatsApp Enabled Using ESP Module, Twilio, and ThingESP. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 8(2), 296-310. DOI: https://doi.org/10.5281/zenodo.11204974
43. Chakraborty, S. & Aithal, P. S. (2024). Let Us Build a MQTT Pub-Sub Client in C# For IoT Research. International Journal of Management, Technology, and Social Sciences (IJMTS), 9(1), 104-114. DOI: https://doi.org/10.5281/zenodo.10603409
44. Chakraborty, S. & Aithal, P. S. (2024). Autonomous Fever Monitoring System For Child Using Arduino, ESP8266, WordPress, C# And Alexa. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 8(1), 135-144. DOI: https://doi.org/10.5281/zenodo.10710079
45. Chakraborty, S. & Aithal, P. S. (2024). Smart LPG Leakage Monitoring and Control System Using Gas Sensor (MQ-X), AWS IoT, and ESP Module. International Journal of Applied Engineering and Management Letters (IJAEML), 8(1), 101-109. DOI: https://doi.org/10.5281/zenodo.10718875
46. Chakraborty, S., & Aithal, P. S. (2024). Communication Channels Review For ESP Module Using Arduino IDE And NodeMCU. International Journal of Applied Engineering and Management Letters (IJAEML), 8(1), 1-14. DOI: https://doi.org/10.5281/zenodo.10562843
47. Chakraborty, S., & Aithal, P. S. (2023). CRUD Operation on WordPress Database Using C# SQL Client. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 7(4), 138-149. DOI: https://doi.org/10.5281/zenodo.10162719
48. Chakraborty, S., & Aithal, P. S., (2023). CRUD Operation on WordPress Database Using C# And REST API. International Journal of Applied Engineering and Management Letters (IJAEML), 7(4), 130-138. DOI: https://doi.org/10.5281/zenodo.10197134
49. Chakraborty, S., & Aithal, P. S., (2023). CRUD Operation on WordPress Posts from C# over REST API. International Journal of Management, Technology, and Social Sciences (IJMTS), 8(4), 223-231. DOI: https://doi.org/10.5281/zenodo.10264407
50. Chakraborty, S. & Aithal, P. S. (2023). CRUD Operation On WordPress Custom Post Type (CPT) From C# Over REST API. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 7(4), 323-331. DOI: https://doi.org/10.5281/zenodo.10408545
51. Chakraborty, S. & Aithal, P. S. (2023). Let Us Build a WordPress Custom Post Type (CPT). International Journal of Applied Engineering and Management Letters (IJAEML), 7(4), 259-266. DOI: https://doi.org/10.5281/zenodo.10440842
52. Chakraborty, S. & Aithal, P. S. (2024). Let Us Manage BP Monitor Data Using WordPress Server and C#. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 8(1), 1-9. DOI: https://doi.org/10.5281/zenodo.10551926
53. Chakraborty, S. & Aithal, P. S. (2024). Don’t Worry; AI will Take Care of Your Sweet Home. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 8(1), 240-250. DOI: https://doi.org/10.5281/zenodo.10780905
54. Chakraborty, S. & Aithal, P. S. (2024). AI Bedroom. International Journal of Applied Engineering and Management Letters (IJAEML), 8(1), 110-119. DOI: https://doi.org/10.5281/zenodo.10780920
55. Chakraborty, S., & Aithal, P. S. (2023). How To Create Our Custom Model in CoppeliaSim From 3D File. International Journal of Applied Engineering and Management Letters (IJAEML), 7(2), 164-174. DOI: https://doi.org/10.5281/zenodo.8117666
56. Chakraborty, S., & Aithal, P. S. (2023). Smart Home Simulation in CoppeliaSim Using C# Through WebSocket. International Journal of Applied Engineering and Management Letters (IJAEML), 7(2), 134-143. DOI: https://doi.org/10.5281/zenodo.8075717
57. Chakraborty, S., & Aithal, P. S. (2023). Automated Test Equipment Simulation in CoppeliaSim Using C# Over WebSocket. International Journal of Management, Technology, and Social Sciences (IJMTS), 8(2), 284-291. DOI: https://doi.org/10.5281/zenodo.8117650
58. Chakraborty, S., & Aithal, P. S. (2023). Industrial Automation Debug Message Display Over Modbus RTU Using C#. International Journal of Management, Technology, and Social Sciences (IJMTS), 8(2), 305-313. DOI: https://doi.org/10.5281/zenodo.8139709
59. Chakraborty, S., & Aithal, P. S. (2023). Modbus Data Provider for Automation Researcher Using C#. International Journal of Case Studies in Business, IT, and Education (IJCSBE), 7(3), 1-7. DOI: https://doi.org/10.5281/zenodo.8162680
60. Sudip Chakraborty, & Aithal, P. S., (2021). Demonstration of Modbus Protocol for Robot Communication Using C#. International Journal of Applied Engineering and Management Letters (IJAEML), 5(2), 119-131. DOI: https://doi.org/10.5281/zenodo.5709235