Communication Channels Review For ESP Module Using Arduino IDE And NodeMCU

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

[1] D.Sc. Researcher, Institute of Computer Science and Information Science, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: drsudip.robotics@gmail.com

[2] Senior Professor, 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.10562843
Google Scholar Citation: IJAEML

How to Cite this Paper:
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

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.2581.7000.0209

Received on: 02/01/2024
Published on: 25/01/2024

Communication Channels Review For ESP Module Using Arduino IDE And NodeMCU

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: drsudip.robotics@gmail.com

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

ABSTRACT

Purpose: *Wireless communication is a common scenario in our everyday lives. There are several strong reasons why wired communication is becoming obsolete. Every day, in every field, devices are becoming wifi, the primary communication medium. The researcher is also integrating the Wifi into their project. In the Wi-Fi category, the ESP module from Espressif Systems is the most popular in the embedded world. Several board variations and modes of communication are also available. When we integrated the ESP module into our project, we faced several challenges due to the vast amount of information available over the net. Finding a workable code is a time-consuming task. Here, for the researcher, we provide a summary of the Esp module communication in various forms. All provided codes are tested in our labs and available on the Github repository for easy integration.*

Design/Methodology/Approach: *we created the software environment to test each communication channel. We use an ESP8266-based NodeMCU module. We use our online server to test the communication channels in several scenarios. A fiber optic backbone was used to get better performance. To program the nodemcu, we used Arduino IDE. It is a popular and rapid prototyping platform. The two modules were used for client-server communication.*

Findings/Result: *we tested various communication modes available for the NodeMCU module. To avoid communication latency, we integrate a high-bandwidth internet backbone. Among the available modes, we found that an MQTT performs better response. The other modes suffer data updation delays due to inherent protocol overhead. For real-time sensor applications, the lite weight MQTT protocol is the best way to integrate it into our research.*

Originality/Value/ Novelty: *The ESP module has been used in various research projects for decades. There are plenty of documents available around the globe. But the scenario is that when we start finding the effective code for our research project, most of the time, it consumes our valuable time. We need to provide the tested code to the researcher. So through this research work, the researcher can get esp module tested code for the available mode of communication.*

Type of Paper: *Experimental-based Research.*

Keywords: ESP module communication, ESP Module MQTT example. HTTP, HTTPS, and Websocket communication for ESP module.

1. INTRODUCTION :

Problem statement: IoT, the Internet of Things, is used everywhere. There are several modules available in the market. However, the ESP module occupies a place due to its cost and simplicity. Now, we are using this module everywhere. When our researchers wanted to integrate the wifi, they mostly decided to use this module because it is cheap. On the internet, a vast document is available. But which will work for us? We need to find out, and it is a time-consuming task. For most of the projects, we do not have time to provide time to find this kind of research work.

Indication of methodology: We conducted several experiments with the esp module to find the best-tested code. We use the Arduino IDE module, the popular platform to program the ESP module. The

wifi module is used primarily for IoT. Using this module, we can upload sensor data to the cloud server and consume the uploaded data on the other side using the MQTT protocol. Using HTTP protocol, we can communicate with the server and fetch the data. Using the HTTPS module, we can establish communication with a secure website.

Essential findings of others in this field: There are several studies already carried out, and day by day, they are increasingly using their projects using the esp module. Among those research works, we are describing a couple of projects here. Parihar Y. S. 2019 [1] illustrates the Internet of Things using nodemcu. The research work (of Sarah A. et al. (2020). [2]) describes the experiment of the home automation system. The researchers used an esp8266 module and programmed it inside the Arduino framework for this work. Oton et al. 2021 [3] developed a low-cost, open-source, internet-based SCADA system using ESP 32 modules and Arduino cloud. Chakraborty, S. et al. 2022-23 [4-13] have demonstrated some research using esp module and AWS IoT cloud infrastructure.

What study is done in this paper: we study a couple of subjects. First of all, we check which code is working in real-life projects. We compare the response time among different modules. We observed which mode of communication is effective and reliable for sensor applications.

Principal conclusion: for IoT applications, the popular used modules are ESP modules, which are available in various form factors. Most researchers find the proper code to integrate into their research project. Here, the different tested codes are open to the researcher so they can incorporate them into their project.

2. REVIEW OF LITERATURE/ CURRENT STATUS :

Before continuing the research, we studied the literature listed in Table 1, focusing on the subject area and technology used in their study.

Table 1: The projects with the subject and used technology.

S. No.	Focus/Subject	Technology/Algorithm/ Module/Components	Reference
1	Internet of Things and nodemcu	IoT, NodeMCU	Parihar, Y. S. (2019). [1]
2	Basic experiments of home automation using ESP8266, Arduino, and XBee. In the 2020 IEEE International Conference on Smart Internet of Things (SmartIoT)	ESP8266, arduino and XBee	Sarah et al. (2020).[2]
3	Low-cost open-source IoT-based SCADA system for a BTS site using ESP32 and Arduino IoT cloud	ESP32 and Arduino IoT cloud	Oton et al. (2021). [3]
4	A Practical Approach To GIT Using Bitbucket, GitHub, and SourceTree	Bitbucket, GitHub, and SourceTree	Chakraborty et al. (2022). [4]
5	How to make IoT in C# using Sinric Pro	C#, Sinric Pro	Chakraborty et al. (2022). [5]
6	Virtual IoT Device in C# WPF Using Sinric Pro	C# WPF, Sinric Pro	Chakraborty et al. (2022). [6]
7	Let Us Create An IoT Inside the AWS Cloud	AWS IoT Cloud.	Chakraborty et al. (2023). [7]
8	Let Us Create a Physical IoT Device Using AWS and ESP Module	AWS, ESP Module	Chakraborty et al. (2023). [8]
9	Let Us Create Multiple IoT Device controllers using AWS, ESP32, And C#	AWS, ESP32 And C#	Chakraborty et al. (2023). [9]

10	Let Us Create Our Desktop IoT Soft-Switchboard Using AWS, ESP32 and C#	AWS, ESP32 and C#	Chakraborty et al. (2023). [10]
11	Let Us Create A Lambda Function for Our IoT Device In The AWS Cloud Using C#	AWS Cloud, C#	Chakraborty et al. (2023). [11]
12	IoT-Based Industrial Debug Message Display Using AWS, ESP8266, And C#	AWS, ESP8266 And C#	Chakraborty et al. (2023). [12]
13	IoT-Based Switch Board for Kids Using ESP Module And AWS	ESP Module, AWS	Chakraborty et al. (2023). [13]
14	Internet of things-based photovoltaics parameter monitoring system using NodeMCU ESP8266	NodeMCU ESP8266	Sutikno et al. (2021). [14]
15	Design and implementation of a low-cost, open-source IoT-based SCADA system using ESP32 with OLED, ThingsBoard, and MQTT protocol	ESP32, OLED, ThingsBoard, MQTT protocol	Aghenta et al. (2019). [15]
16	Implementation of the Internet of Things (IoT) for remote light control using nodemcu esp8266 and thing speak via website-based internet	esp8266, things peak	Nasution et al. (2023). [16]
17	Design and Implementation of ESP32-Based IoT Devices	ESP32	Hercog et al. (2023). [17]
18	Smart Notice Board Using the Internet of Things–Based NODEMCU ESP8266	ESP8266	Turpati et al. (2023). [18]
19	Let Us Create an Alexa-Enabled IoT Device Using C#, AWS Lambda and ESP Module	C#, AWS Lambda and ESP Module	Chakraborty et al. (2023). [19]
20	Alexa Enabled IoT Device Simulation Using C# And AWS Lambda	C# And AWS Lambda	Chakraborty et al. (2023). [20]
21	Let Us Create an Alexa Skill for Our IoT Device Inside the AWS Cloud	Alexa Skill	Chakraborty et al. (2023). [21]
22	Smart Magnetic Door Lock for Elderly People Using AWS Alexa, IoT, Lambda, and ESP Module	AWS Alexa, IoT, Lambda and ESP Module	Chakraborty et al. (2023). [22]
23	Industrial Automation Debug Message Display Over Modbus RTU Using C#	Modbus RTU, C#	Chakraborty et al. (2023). [23]
24	Modbus Data Provider for Automation Researcher Using C#	C#	Chakraborty et al. (2023). [24]
25	MVVM Demonstration Using C# WPF	C# WPF	Chakraborty et al. (2023). [25]
26	Smart Home Simulation in CoppeliaSim Using C# Through WebSocket	C#, WebSocket	Chakraborty et al. (2023). [26]
27	Automated Test Equipment Simulation In CoppeliaSim Using C# Over WebSocket	CoppeliaSim, C#, WebSocket	Chakraborty et al. (2023). [27]

28	Using Arduino to Control a Wi-Fi-Based Home Automation System in the Cloud	Arduino, Wi-Fi	Rathod et al. (2023). [28]
29	Relay Driver Based on Arduino UNO to Bridge the Gap of The Digital Output Voltage of The Node MCU ESP32	Arduino UNO, Relay Driver, Node MCU ESP32	Yulianto et al. (2023). [29]
30	A Breakthrough in wireless sensor networks and internet of things	ESP8266	Mehta et al. (2015). [30]

3. OBJECTIVES OF THE PAPER :

Description for Objectives.

(1) To study the performance of different ESP module
(2) To review updated various communication modules for the research project.
(3) To analyze the performance of different communication methods
(4) To compare the different esp module performance.
(5) To evaluate the better and more reliable communication channel.
(6) To test the different ESP module performance
(7) To prove the esp module performance for different real-life projects.
(8) To design a robust communication channel for the sensor and real-time application.
(9) To develop some practical code to use in our project.
(10) To interpret the various communication methods
(11) To create a set of code that is irrespective of the project and can be used in our research work.

4. METHODOLOGY :

Now, we will see the method we applied to our research work. The free library is available to help us do the task. When we build the firmware with this type of configuration, we can get the features below from the module.

4.1 NodeMCU as HTTP/HTTPS Server and Client communication:

The HTTP, complete form, is a hypertext transfer protocol. The nodemcu can serve HTTP servers as well as clients. It has the following features.

1) It can act as an HTTP server functionality where it handles the incoming request from the client. It provides the requested information that the client requests.
2) It can be configured as an HTTP client. To fetch the data from the server, we use the client configuration. We can upload the sensor data to the cloud server using this configuration.
3) This module can serve as a web server. We store web pages inside the module and can fetch the server page from other modules or systems like a PC or laptop.
4) The available library has the feature to serve the get and post method. Using this method, it handles the client's request.
5) It can handle static and dynamic HTML page content. Displaying the sensor data is beneficial. For this, we need only the free browser in all the operating systems.
6) the module can act as a bridge between the nodes. It helps to exchange the sensor data between two devices.
7) For Some confidential applications or to challenge security issues, we can use HTTPS, which provides more security on data flow over communication channels. Figure 1 depicts the HTTP communication flow between the two modules.

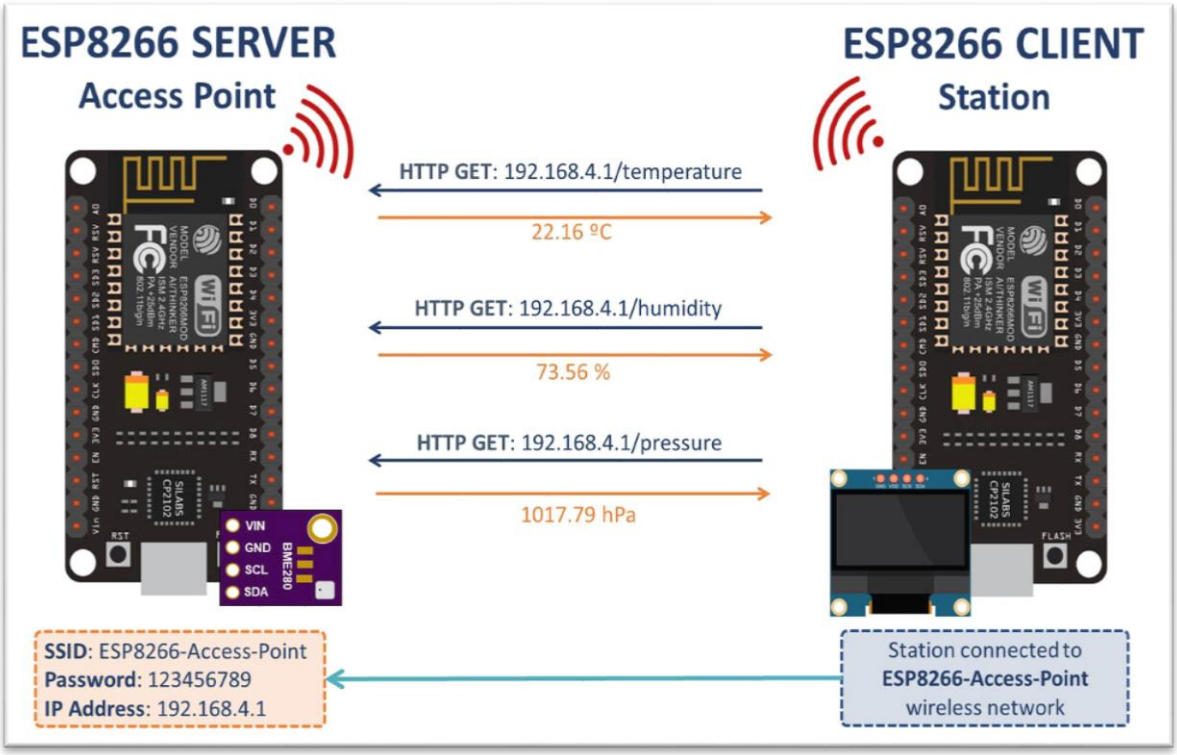

Fig. 1: HTTP communication between two ESP modules (Image Source: www.google.com)

4.2 NodeMCU as UDP Server/Client communication:

UDP is a lightweight protocol.

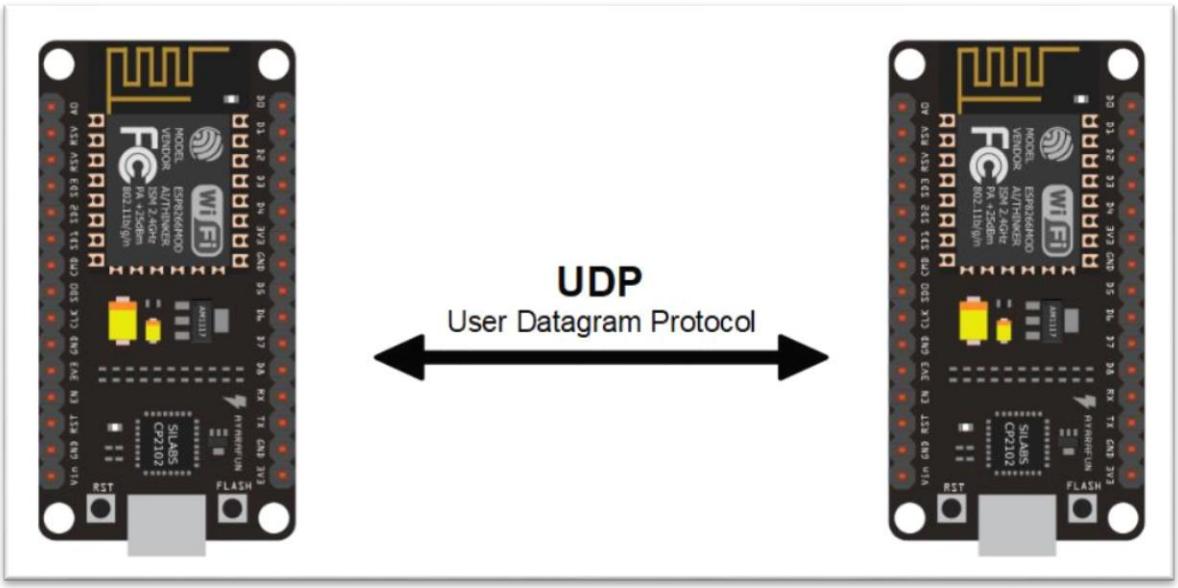

Fig. 2: The UDP communication between two ESP modules (Image Source: www.google.com)

Protocol overhead is lower than TCP. The nodemcu module can handle the UDP. The following features are available for the UDP protocol.

(1) Due to the connectionless properties, it is low latency in the communication channel. For this reason, it is suitable for real-time applications where data needs to flow in real-time.
(2) The UDP is a client and server model. The server provides the data on client requests. The node mcu can act as a UDP server and the client.
(3) The nodemcu in UDP communication can happen between the access point and the station.

(4) It is wifi enabled; there is no need for a physical connection. The module can transfer the data without physical wire over the wireless medium.
(5) The nodemcu can be programmed using Arduino IDE; it is an easy interface, and programming is also easy.
(6) The nodemcu can be set as an access point, I.e., run without a router and external network.
(7) Where we need quick data transfer, we can use nodemcu udp because it transfers data faster than other methods due to its less protocol overhead.

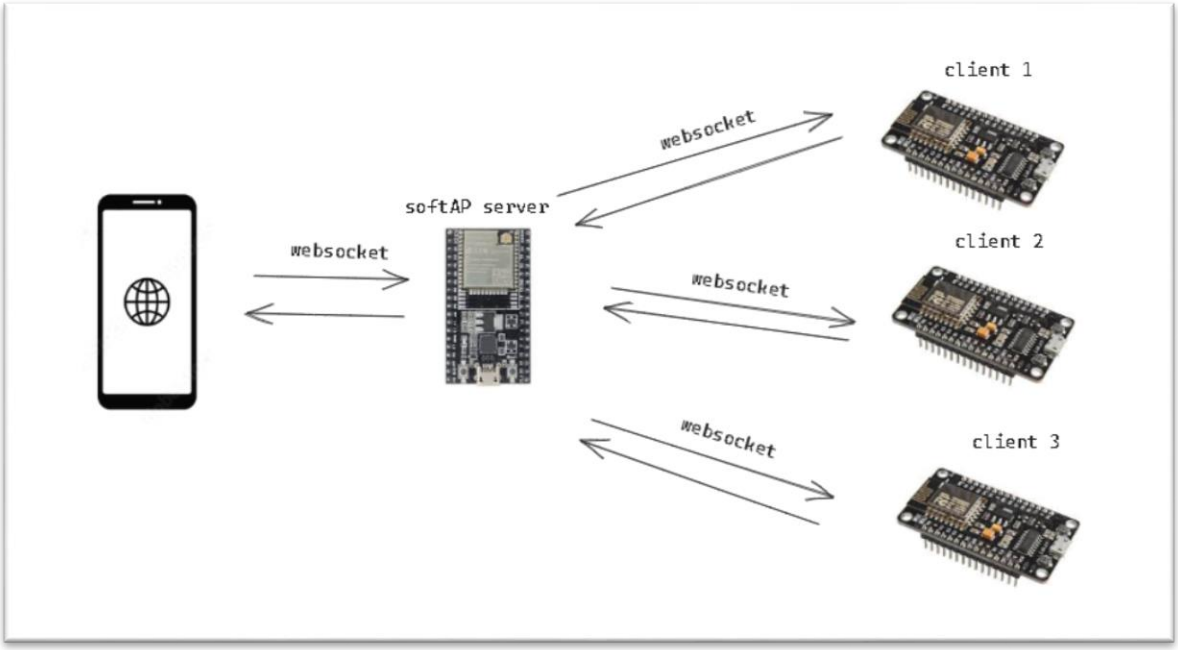

Fig. 3: The WebSocket communication between two ESP modules (Image Source: www.google.com)

4.3 NodeMCU as WebSocket Server/Client communication:

The WebSocket is bidirectional full duplex communication. It is becoming trendy. We can implement the WebSocket easily using the nodemcu module. Figure 3 depicts the data flow over WebSocket. The features are:

(1) All the esp modules support websocket communication. Various libraries are available to implement web sockets. It is a real-time bidirectional port, which is a perfect fit for real-time sensor applications.
(2) Using nodemcu and WebSocket, we can quickly implement a WebSocket server, which provides real-time data to the client devices.
(3) Using the WebSocket server, the system can remotely control the opening of webpages inside the browser.
(4) The nodemcu can act as a web socket client, which can fetch the data from the WebSocket server or update the server database.
(5) It can be used as data visualization through a webpage interface.
(6) This node mcu can be enabled using Arduino IDE for websocket implementation.
(7) The WebSocket is an inherently low latency protocol. It is a perfect fit for sensor data transfer applications.

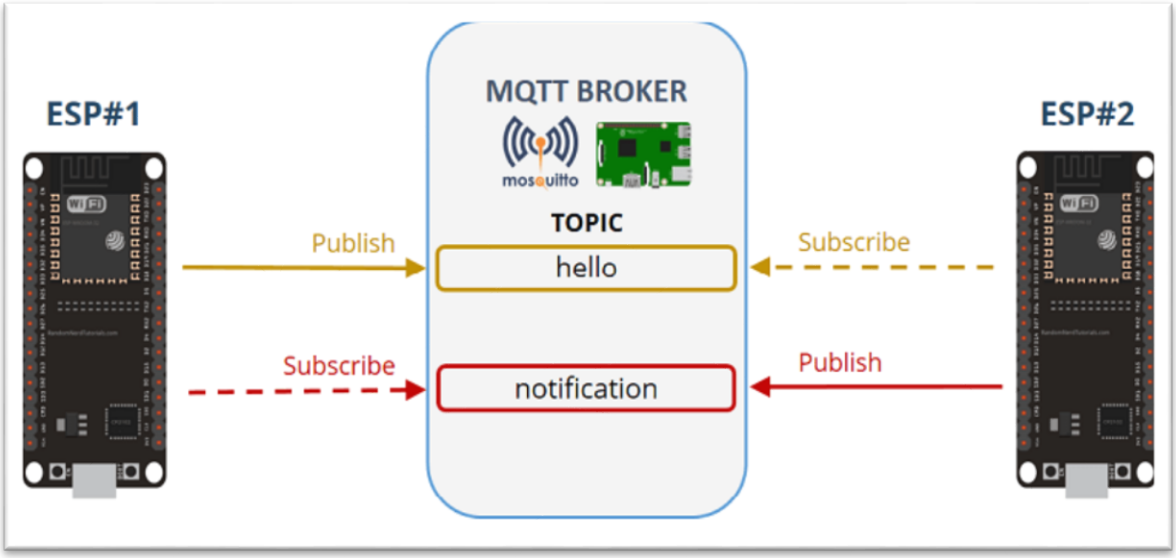

Fig. 4: The MQTT communication between two ESP modules (Image Source: www.google.com)

4.4 NodeMCU as MQTT communication:

The MQTT, which stands for message queuing telemetry transport, is now a widely used communication protocol depicted in Figure 4. Our nodemcu is capable of handling the MQTT protocol. It has the following features:-

1) Our node has enough hardware capable of handling the MQTT protocol.
2) The MQTT protocol follows a publish-subscribe model. It can perform both types of communication.
3) The mqtt segregates the message using topic. The nodemcu can publish or subscribe to the topic. The topic is used to exchange data among various devices.
4) When we use the nodemcu topic to send or receive the topic, care should take to avoid spaces and leading forward slashes. We should always keep a short and straightforward manner to understand at a glance. The nodemcu has limited processing power. The large and complex name takes longer to process, so the communication delay happens on the other device end, waiting for the response.
5) The integration of mqtt in nodemcu is pretty simple. Several modules are available to integrate the nodemcu as mqtt publish or subscribe nodes.

4.5 NodeMCU as ESP-NOW communication:

The nodemcu can be performed as espnow protocol. It has several advantages, which are listed below:

(1) It is a communication protocol developed by Espressif itself. It is used for low-power peer-to-peer communication.
(2) It is an inherently decentralized network. One major drawback in a centralized network is that the complete infrastructure is alive only when the central or coordinated node works. In the decentralized infrastructure, every independent system can send the data to another device without help from the other to find the communication route. In this scenario, no access point or router is required.
(3) The protocol is optimized for low latency and power consumption. It is a perfect fit for real-time sensor applications where the operating module has low-power budgets.
(4) Its configuration is simple. Using the Arduino IDE, we can program easily.
(5) It creates a mesh network, depicted in Figure 4. Every node can communicate with each other.
(6) It has several communication modes, like unicast, multicast, and broadcast.
(7) The protocol is best suited for IoT-based sensor networks. It saves the battery life of the node devices, keeping system performance optimal.
(8) It has significant advantages. It supports custom data packet format. It facilitates the developer using custom protocols for the project's needs.
(9) Using the Arduino IDE, we can program the nodemcu with this protocol.

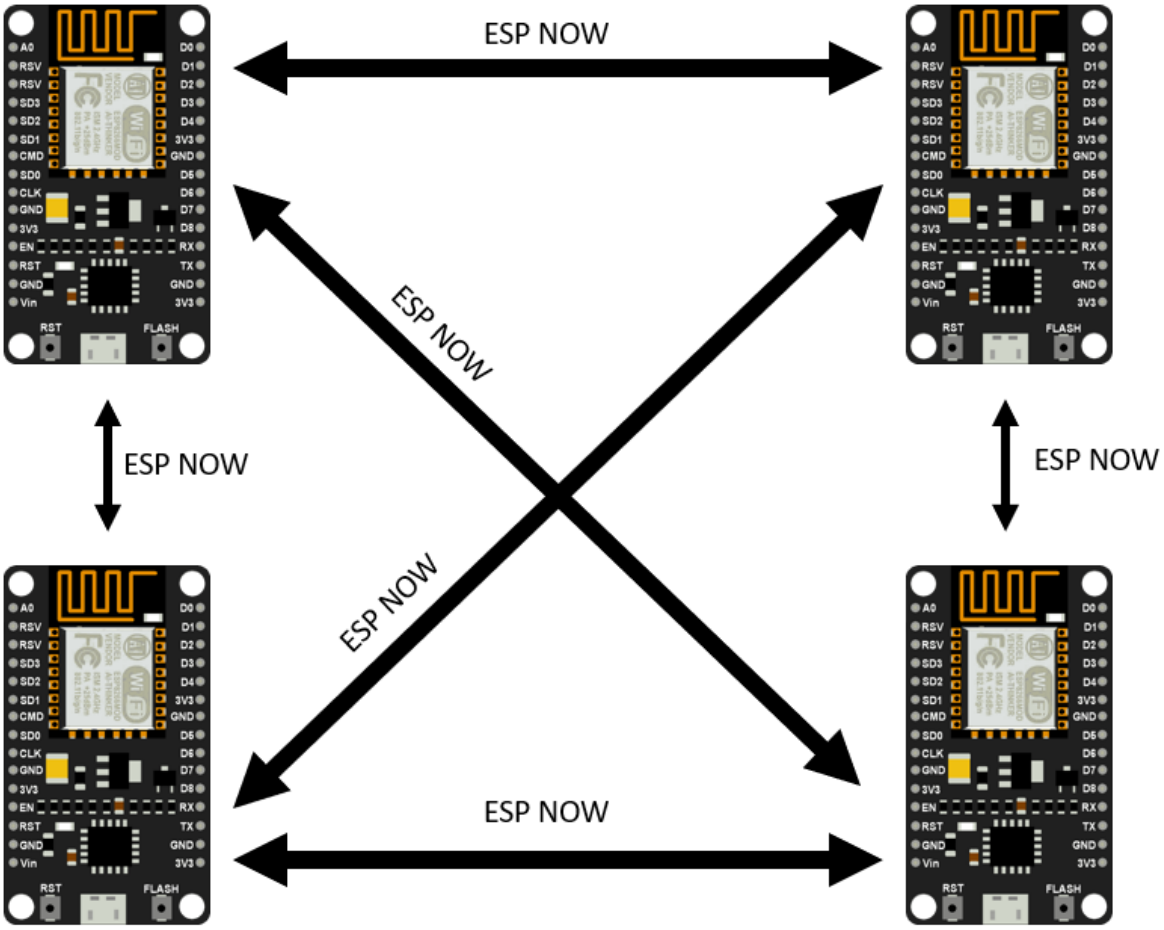

Fig. 4: The ESP-NOW communication Among ESP modules (Image Source: www.google.com)

5. EXPERIMENTS :

Now, we can do experiments with ESP modules to test the various modes of communication. All project codes are available from https://github.com/sudipchakraborty/Communication-Channels-Review-For-ESP-Module.git. Download the project. Extract the folder. There is a folder for each communication mode. Now, we are discussing how we will execute each mode of communication. Before the experiment, we must install Arduino IDE, freely available online from https://www.arduino.cc/en/software. Add package for esp8266 module.

5.1 NodeMCU as ESP-NOW:

We must follow the steps below for the ESP-NOW experiment.

1) For this experiment, we need at least two ESP modules. One is for the sender, and the other is for the receiver.
2) Under the project folder, one folder is available named "ESP-NOW." open "**ESP_NOW_Receiver. ino**" in a new project window.
3) Connect one nodemcu, build the project, and upload the code.
4) Open the terminal window and observe the message inside the terminal.
5) Connect another nodemcu with the system under the "SENDER" folder, and open "ESP_NOW_Sender. in" in another project window. Build the code and upload it to the esp module.
6) Open the terminal window and observe the data flow.

5.2 NodeMCU as HTTP Server/Client:

1) We need two NodeMCU8266(12E) to test http client and server communication.
2) Connect one module with a PC or laptop.
3) From the project folder, open the "..**HTTP\SERVER.ino**" file.
4) Add ssid and password.
5) Build and upload the code to the esp module.
6) Open a terminal and see the status information.
7) From the "..HTTP\CLIENT" folder, double-click on the "**CLIENT.ino**" file. It will open in a new Arduino project window.
8) Add wifi credentials and add the assigned server IP by DHCP.
9) Build and upload the code. Open the terminal window.
10) Observe the progress information.
11) This is the bare minimum code to see the workflow. Once the code works, we can customize it for our project. Most custom projects need a parser and custom packet formation code.

5.3 NodeMCU as HTTPS Client without certificate:

The HTTP client is almost identical to the HTTP protocol except for security features. We tested two types of available HTTPS codes. One is without certification, and the other is with a certificate code.

1) Under the "HTTPS\ Client_No_Certificate " open Client_No_Certificate.ino" .
2) Add the Wi-Fi credentials, SSID, and password.
3) Add the website address or IP to the "cmd" variable we want to connect.
4) Build the project and upload it to the esp module.
5) Open the terminal and observe the progress status.

5.4 NodeMCU as HTTPS Client with a certificate:

Now we see how an HTTPS connection can happen with a certificate.

1) Under the "HTTPS\ Client_Root_Certificate" open "**Client_Root_Certificate.ino**".
2) Add the Wi-Fi credentials, SSID, and password.
3) Add the website address or IP to the "cmd" variable we want to connect.
4) **Get Root Certificate**:
   a) Open the site where we want a certificate.
   b) Open Google Chrome. From the address bar, click "view sidebar information" on the left side of the site name.
   c) Click "connection is secure" from the dropdown list. Then click "Certificate is valid".
   d) click on the "Details" tab from the tabbed page. Select "ISRG Root X1".
   e) Click on the export button. Select save as type "Based64-encoded ASCII, single certificate."
   f) Give a name or keep it as default "ISRG Root X1".
   g) Click on save.
   h) Open the certificate file, which was downloaded just now.
   i) Paste in the code between "BEGIN CERTIFICATE" and "END CERTIFICATE."
5) Build the project and upload it to the esp module.
6) Open the terminal and observe the progress status.

5.5 NodeMCU is an MQTT client:

To test the MQTT client, follow the below steps:-

1) We need two ESP modules. One for publishing topic data and the other to subscribe to the topic data
2) Under the mqtt folder, open the "**Mqtt. ino**" file.
3) Add wifi credentials
4) Build the code and upload it to the esp module.
5) Observe the message that is publishing the topic within an interval.
6) Now, open the same file in another Arduino project window.
7) Add wifi credentials.
8) Instead of publishing, change to subscribe to the same topic.
9) built the project and uploaded it to the esp module.
10) Open the terminal and observe the message.

5.6 NodeMCU as WebSocket Client-Server:

To create a WebSocket server, we need to follow the below steps:

1) We need one NodeMCU. We can purchase from an online store.
2) From the "WEBSOKET" folder, Open the " **SERVER.ino**."
3) From the Arduino library manager, search "websocket," and from the list, install "**WebSockets by Markus Sattler.**"
4) Build the project and upload it to the NodeMCU.
5) From a terminal window, see the progress.
6) Open the browser and see the message inside the terminal window.
7) Open the "CLIENT.ino" inside a new Arduino project window.
8) Build and upload the code to the esp module. Open the terminal window. See status information.
9) This is the basic code. We can customize it according to the project.

6. RESULTS & DISCUSSIONS :

We experimented with most of the communication channels available for the esp module. The final result is encouraging, for every communication mode has pros and cons. The concluded subject is that we need to choose the ESP module and its mode of communication wisely. Otherwise, the project fails to comply with the requirements or might not fulfill the project goal.

7. ANALYSIS / Comparison OF RESULTS :

After observing the various modes of communication, due to the limited capacity of the esp module, it gets slow or responds when we run the more overhead protocol. The https takes more time due to certificate verification. The Mqtt UDP protocol is faster than another module. If most communication is under one local network, then TCP/IP is a good choice. ESP-NOW is the best choice if the application does not need any external network or internet. MQTT is the best choice if the application demands easy operation and real-time fast response over the network.

8. SUGGESTIONS / RECOMMENDATIONS :

- Most of the source code and images are adapted from https://randomnerdtutorials.com/. The best website to find various valuable projects for home automation on ESP modules.
- To know details about ESP-NOW: https://randomnerdtutorials.com/esp-now-esp8266-nodemcu-arduino-ide/
- How to create an MQTT application in C#: https://www.youtube.com/watch?v=R-JTYzT1yMk&t=38s
- For Windows Broker: https://www.youtube.com/watch?v=DH-VSAACtBk and https://www.youtube.com/watch?v=4ZEPPQLY5o4
- NodeMCU as MQTT Client https://www.youtube.com/watch?v=c1st5cVRRzo&t=619s
- NodeMCU as UDP communication: https://www.youtube.com/watch?v=pKsNk7RQevU
- NodeMCU as Tcp Server/client: https://www.youtube.com/watch?v=a7wEuzupfQc&t=59s
- WebSocket Server and client: https://www.youtube.com/watch?v=fREqfdCphRA&t=35s and https://gitlab.com/MrDIYca/code-samples/-/blob/master/mrdiy_websocket_project_example.ino

9. CONCLUSION :

The Wi-Fi module is suitable for IoT applications. It is cheap and available in various form factors, which is best fit for different IoT-based projects. Here, we discuss multiple modes of communication that the ESP module can handle. Using the Arduino IDE, a popular IDE, can program the ESP module very quickly. To describe the project, we discuss some research work like Parihar Y. S. 2019 [1], which is the best contribution in this field. Table 1 lists the contributor with focus or subject and technology used in their research work. Then, we listed out a couple of objectives of the research paper. After that, we demonstrate the research methodology using several figures. Doing practical experiments is the best to learn anything. We provide the steps to experiment with each mode of communication. We acknowledge our project contributor. Finally, we added the reference paper from where we got help and for the reader to further research work.

10. LIMITATIONS :

The esp module is used in various IoT-enabled devices. Through this research work, we found some limitations, which is not to discourage the use of the esp module, but we need to care about using the esp module:-

1) The esp 8266 module has a limited analog and GPIO pin. We cannot consider it for A big project
2) It has limited processing power. When it processes more tasks at a time, the response gets delayed.
3) It has limited RAM. It can handle a small amount of data.
4) It has a limited range. For long-range, this module does not fulfill the requirement.
5) The module performance depends on network bandwidth, protocol overhead, sampling rate, application, etc.
6) To deploy the esp module, choose the esp module and mode of communication based on the project requirement.
7) Before applying for the medical device, the system must undergo several test procedures and get approval from the medical instrument approval authority.
8) Sometimes, the module goes to a nonresponsive state. We need to implement the hardware reset feature so that when the system is in a test state, it can be brought to a working state.
9) The module starts to be delayed when it is affected by wifi congestion.

ACKNOWLEDGEMENT :

I sincerely thank my research supervisor, Dr. P. S. Aithal, for his guidance, supervision, encouragement, and support. I am grateful to Srinivas University for providing the resources, facilities, and assistance to facilitate this study. I sincerely appreciate my family's unwavering support, understanding, and motivation during this research journey. This work would not have been possible without the collective help and encouragement of these remarkable individuals and institutions.

REFERENCES :

[1] Parihar, Y. S. (2019). Internet of things and nodemcu. *Journal of emerging technologies and innovative research*, *6*(6), 1085-1088. Google Scholar↗

[2] Sarah, A., Ghozali, T., Giano, G., Mulyadi, M., Octaviani, S., & Hikmaturokhman, A. (2020, August). Learning IoT: Basic experiments of home automation using ESP8266, Arduino, and XBee. In 2020 IEEE International Conference on Smart Internet of Things (SmartIoT) (pp. 290-294). IEEE. Google Scholar↗

[3] Oton, C. N., & Iqbal, M. T. (2021, December). Low-cost open-source IoT-based SCADA system for a BTS site using ESP32 and Arduino IoT cloud. In 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) (pp. 0681-0685). IEEE. Google Scholar↗

[4] 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

[5] 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

[6] 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

[7] 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

[8] 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

[9] 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

[10] 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

[11] 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

[12] 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.

[13] 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

[14] Sutikno, T., Purnama, H. S., Pamungkas, A., Fadlil, A., Alsofyani, I. M., & Jopri, M. H. (2021). Internet of things-based photovoltaics parameter monitoring system using NodeMCU ESP8266. *International Journal of Electrical & Computer Engineering, 11*(6), 5578~5587. Google Scholar↗

[15] Aghenta, L. O., & Iqbal, T. (2019). Designed and implemented a low-cost, open-source IoT-based SCADA system using ESP32 with OLED, ThingsBoard, and MQTT protocol. *AIMS Electronics and Electrical Engineering, 4*(1), 57-86. Google Scholar↗

[16] Nasution, W. S. L., & Nusa, P. (2023). Implement the Internet of Things (IoT) for remote light control using nodemcu esp8266 and think of speaking via website-based internet. *Journal of Computer Science and Technology, 3*(1), 33-39. Google Scholar↗

[17] Hercog, D., Lerher, T., Truntič, M., & Težak, O. (2023). Design and Implementation of ESP32-Based IoT Devices. *Sensors, 23*(15), 6739. Google Scholar↗

[18] Turpati, S., Richi, S. R. P., Mohammed, T. S., Naveen, K. S., & Ranga, R. S. (2023). Smart Notice Board Using the Internet of Things–Based NODEMCU ESP8266. In *AI-Aided IoT Technologies and Applications for Smart Business and Production* (pp. 211-224). CRC Press. Google Scholar↗

[19] 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

[20] 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

[21] 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. Google Scholar↗ DOI: https://doi.org/10.5281/zenodo.7940237

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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

[27] 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

[28] Rathod, I., Mishra, P., Gupta, N., & Chouhan, D. (2023). Using Arduino to Control a Wi-Fi Based Home Automation System in the Cloud. In *Artificial Intelligence, Internet of Things, and Society 5.0* (pp. 453-463). Cham: Springer Nature Switzerland. Google Scholar↗

[29] Yulianto, Y. (2023). Relay Driver Based on Arduino UNO to Bridge the Gap of The Digital Output Voltage of The Node MCU ESP32. Engineering, Mathematics and Computer Science *Journal (EMACS), 5*(3), 129-135. Google Scholar↗

[30] Mehta, M. (2015). ESP8266: A Breakthrough in wireless sensor networks and the internet of things. *International Journal of Electronics and Communication Engineering & Technology*, *6*(8), 7-11. Google Scholar↗

****