International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

Establishing a Wireless-Local-Area-Network (WLAN) Connectivity between Multiple Nodes using ESP-Mesh Network Topology for IoT Applications

Monika Kumari[1], Dr. Vivek Kumar[2], Mr. Dayal Chandra Sati[3]

[1]M.Tech. Scholar, Dept. of Electronics & Communication Engineering, B.R.C.M.C.E.T., Bahal, Haryana, India
[2]Head of Dept. of Electrical & Electronics Engineering, B.R.C.M.C.E.T., Bahal, Haryana, India
[3]Assistant Prof., Dept. of Electronics & Communication Engineering, B.R.C.M.C.E.T., Bahal, Haryana, India

---

Abstract - *Here in this work efforts were made to design, develop and establish a Wireless-Local-Area-Network (WLAN) between different nodes connected in a Mesh network topology specifically for IoT applications. Here a hardware based demonstration was made for the implementation of mesh network topology to establish connectivity between multiple Wi-Fi enabled NodeMCU devices for the transfer of sensors data without using the internet and without using any router in between. Here as this system was connected in a mesh network topology, there was no central node or parent node in this system instead here each node communicates with every other node available. It was observed that the system was capable of establishing connectivity between each node automatically by making use of same ssid, password and port address for each node just like a wireless ad-hoc network.*

Key Words: WLAN, NodeMCU, Mesh Network, WiFi, IoT, etc.

1. INTRODUCTION

The development of the Internet of Things (IoT) requires an increasing number of nodes to connect to the internet. However, only limited number (usually fewer than 32) of nodes can directly connect to the same router. A mesh network is a local network topology in which the infrastructure nodes connect directly, dynamically and non-hierarchically to as many other nodes as possible and cooperate with one another to efficiently route data from/to clients. This lack of dependency on one node allows for every node to participate in the relay of information. In a traditional infrastructure Wi-Fi network there is a point-to-multipoint network where a single central node known as the access point (AP) is directly connected to all other nodes known as stations. The AP is responsible for arbitrating and forwarding transmissions between the stations. Some APs also relay transmissions to/from an external IP network via a router. But there are certain limitations associated with it like it has a limited coverage area due to the requirement that every station must be in range to directly connect with the AP. Also it is susceptible to overloading as the maximum number of stations permitted in the network is limited by the capacity of the AP. Therefore, to overcome these limitations there is another infrastructure known as ESP-MESH network where nodes are not required to connect to a central node. Instead, nodes are permitted to connect with neighboring nodes. Nodes are mutually responsible for relaying each other's transmissions. This network topology provides a much greater coverage area as nodes can still achieve interconnectivity without needing to be in range of the central node. Also it is less susceptible to overloading as the number of nodes permitted on the network is no longer limited by a single central node. Wireless Local Area Network (WLAN) includes protocols like Bluetooth, BLE, Thread, Wi-Fi, Zigbee, etc. ESP8266 chip is capable of Wi-Fi Mesh networking. In a mesh network, nodes can self organize and dynamically talk to each other. Any node in that network is able to transmit data packets to any other node, within range, which can then forward data packets through the network to the final destination. Each NodeMCU module that has an ESP8266 chip embedded can act as a node in the Mesh Network.

1.1 Introduction to IoT

The Internet of Things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IoT applications deployed.

2. AIM & OBJECTIVE

The objective of the study was to design, develop and establish a Wireless-Local-Area-Network (WLAN) between different nodes connected in a Mesh network topology specifically for IoT applications. The aim was to demonstrate

the mesh network topology implemented between some Wi-Fi enabled devices for the transfer of data without using internet and without using any router in between. It was expected that the system should be able to automatically establish connectivity between each node by using the same ssid, password and port address for each node just like a wireless ad hoc network. Elaborating the above statement a versatile system was required to establish communication between the different nodes using mesh network topology. Here each node was expected to enable the user to remotely access the status of other two nodes in real-time without using internet or a router. The system was expected to acquire both types of signals analog as well as digital as per the sensor output. WLAN Mesh protocol was to be utilized to establish communication between the hardware units. Hardware prototype should be able to validate the work.

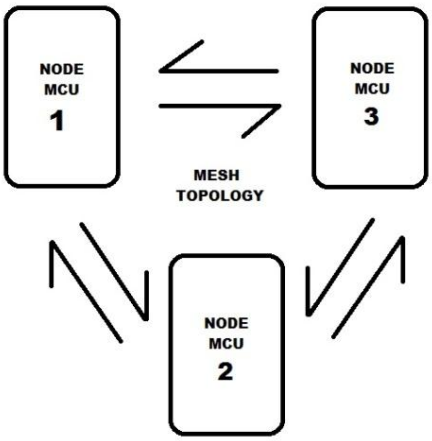

Fig.-1: A Simple Mesh Network Architecture

3. PROBLEM FORMULATION

1. A high performance, low cost microcontroller platform with Wi-Fi protocol integrated was required. After referring to the datasheets and application notes for some devices like ESP01, ESP-12, ESP-32, etc., finally ESP-12 based NodeMCU module was selected for this work
2. The detailed knowledge was gathered about the sensor specifications, their availability in market, operating principles, working procedures, driver circuits and also about their interfacing with the NodeMCU
3. Performed studies about establishing the wireless mesh network connections between multiple nodes deploying sensors, push-buttons, display and relays.
4. Prepared the Bill-of-Material and make procurement for the required components from local sources
5. Designed algorithm and written firmware for each individual node and tested it
6. Tested the system over a bread-board and finally implement it over a self-designed PCB in a CAD tool and finally perform multiple iterations to test it and calibrate it.

4. SYSTEM ARCHITECTURE

As it could be observed that there were three different devices/ nodes in this system. Therefore three separate hardware units centered on NodeMCU modules were designed and developed. As shown below in Fig.-2 a NodeMCU module was interfaced to two push buttons and an OLED display module. The push buttons acted as system inputs and OLED acted as system output. The NodeMCU along with the OLED and push buttons used 3.3V dc power supply for its operation. As OLED was an I2C device, only two pins SDA and SCL were required to interface it with the NodeMCU. Rest two pins were the power pins. Two push buttons were interfaced to the two digital I/O pins of the NodeMCU via two pull-up resistors of 10K in between. This was required to prevent any false triggering over these pins during high-impedance state.

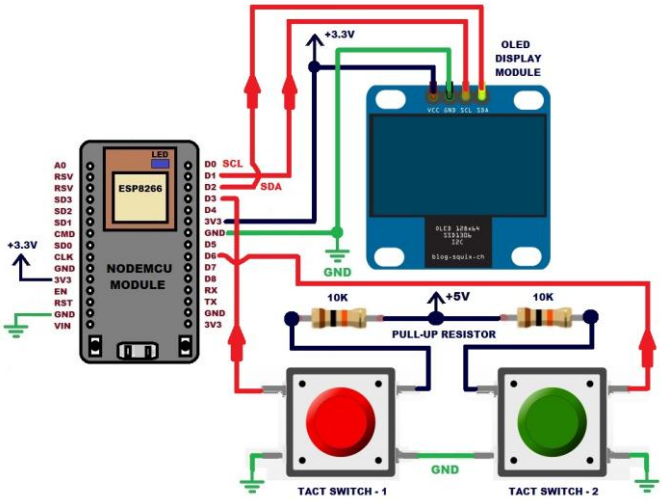

Fig.-2: Node-1 Architecture

Node-2 architecture as shown below in Fig.-3 was also implemented around a NodeMCU module. Here a commonly used temperature and humidity sensor module DHT-11 was interfaced to the NodeMCU. As DHT-11 module used here was a digital output type so instead of using an analog input channel it utilized a simple digital I/O channel. The sensor was made to operate on 3.3V dc and acted as an input for pin D7 of the NodeMCU. On the other side a SPDT relay was also interfaced to one of the digital I/O pins of the NodeMCU. The SPDT relay acted as an output device and was connected on pin D0 of the NodeMCU. The relay acted as an electromagnetic switch and was used here to switch 220V AC load connected, bulb in this case. Here as the relay power consumption was more so the system was fed with a higher power source 5V/ 1A.

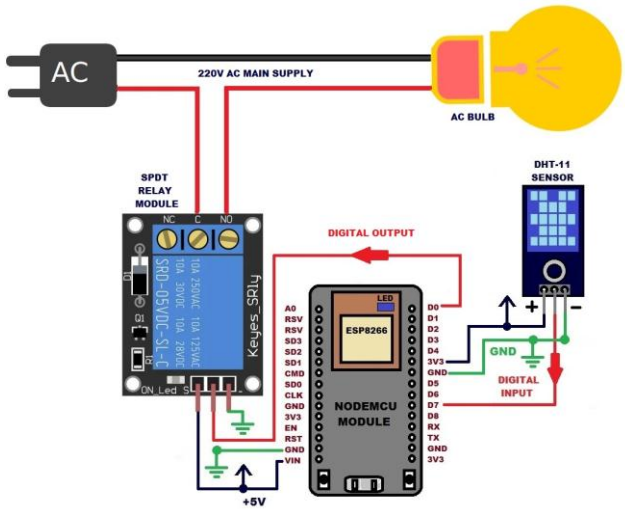

Fig -3: Node-2 Architecture

Node-3 architecture as shown below in Fig.-4 was implemented around another NodeMCU module. Here a commonly used gas sensor module MQ-135 was interfaced to the NodeMCU. As MQ-135 module used here was an analog output type so instead of using a digital input channel it utilized the only analog input channel A0 of NodeMCU. This channel fed internal ADC with the analog values and converted those values to their digital equivalents for further processing. The sensor was made to operate on 3.3V dc and acted as an input device here. On the other side another SPDT relay was interfaced to one of the digital I/O pins of the NodeMCU. The SPDT relay acted as an output device and was connected on pin D0 of the NodeMCU. The relay acted as an electromagnetic switch and was used here to switch 220V AC load connected, bulb in this case. Here as the relay power consumption was more so the system was fed with a higher power source 5V/ 1A.

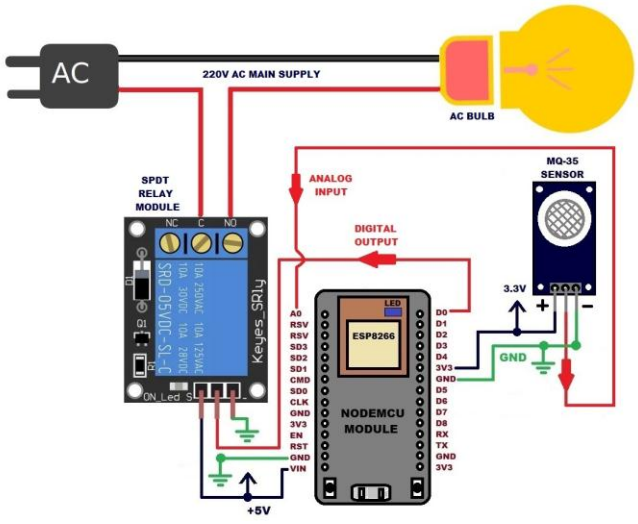

Fig -4: Node-3 Architecture

Table-1: Bill of Material

Sr. No.	Component/ Module	Specification	Quantity
1	NodeMCU Module	ESP-8266	3
3	DHT11 Sensor Module	3-PIN	1
3	MQ-5 Gas Sensor Module	4-PIN	1
4	SPDT Relay	+5V	2
5	IN4007 Diode	1A	4
6	Resistor 10K	Quarter Watt	6
7	Resistor 220E	Quarter Watt	6
8	Resistor 1K	Quarter Watt	4
9	Capacitor Electrolytic	2200uF/35V	3
10	Capacitor Electrolytic	10uF/35V	3
11	Voltage Regulator IC	LM7805	3
12	Copper Clad	130 x 85 mm^2	3
13	Transformer	12-0-12 / 200mA	3
14	Connecting Cable	220V AC	3
15	Berg Strip	Male Connector	6
16	Berg Strip	Female Connector	6
17	Connector	2-PIN	4
18	NPN Transistor	BC-547	2
19	Tact Switch	10mm	2

5. FIRMWARE

In this project we were implementing a wireless local area network between different nodes (NodeMCU devices) using mesh networking protocol through which we could remotely access the control of multiple appliances and also monitor multiple sensors data without internet or rather without having any router in between.
So, for the above mentioned purpose here we used three NodeMCU boards as given below:

1. Nodemcu-1 connected with OLED and two push buttons
2. Nodemcu-2 connected with one DHT-11 Module and one Relay module
3. Nodemcu-3 connected with one MQ-5 Gas sensor and one Relay module

5.1 Source Code for NodeMCU

First of all at the initial stage we downloaded and installed all the required libraries and also included those in the source code. These libraries were:

1. painlessMesh.h
2. ArduinoJson.h
3. SPI.h

4. Wire.h
5. Adafruit_GFX.h
6. Adafruit_SSD1306.h

Then we defined the Wi-Fi Credentials including the Mesh_SSID, Mesh_Password and Mesh_Port. These Wi-Fi credentials remained same for each and every board which we wanted to communicate between each other.

Sending Part of the Code:
First of all we read the button status and if the button status was low then we just toggled the button status variable. So, whenever the button was pressed the button status variable would change from one to zero or from zero to one according to the last variable saved in that variable. After saving the button status then was the time to send that and for sending that we used the JSON Format. JSON (Java Script Object Notation) is just an object notation or we can say a syntax in which a data is represented. It is very popular in sending and receiving data may be over the internet or may be between the hardware.

JSON Code Part:
First of all a JSON object was created in which there would be two values Relay1 and Relay2 and their keys were button1_status and button2_status respectively. So this was how JSON syntax looked like. Likewise we could create many values and assign them respective keys and this data was converted into string format by using the function in the code called as

serializeJson(doc,msg);

So then that JSON format data was saved in string called *msg.*

"Relay1":true,
"Relay2":false,
"Relay3":true,
"Relay4":true,
"Relay5":false

Where "Relayx" were known as 'Values' and true/false were called 'Keys'. Data transmitted in JSON Format appeared in the serial monitor window like this:

{"Relay1":false, "Relay2":false}
{"Relay1":true, "Relay2":false}

Then on the receiver side it was very easy to deserialize that string and just extract that particular data which we wanted out of that whole string.

Code Part at the Receiver Side:
On the receiver side if we wanted the data of Relay1 particularly so first of all we deserialized it

DeserializationError error = deserializeJson(doc, json);
and then

write relay1_status =doc["Relay1"];

which was the name of our value which data we wanted and we got the key of this value easily

digitalWrite(Relay1, relay1_status);

and we could easily turn on and off the relay by using this variable. So this was the simplest and practical explanation of how we transmitted and received data using JSON. Here we just serialized the push button data sending it after every second. Then that data was broadcasted to all the devices under same network under same port number. Then, talking about the receiving part of the code, after receiving the data we first deserialized the JSON and saved different data into different string.

String Temp = doc["TEMP"];
String Hum = doc["HUM"];
String Gas = doc["GAS"];

6. EXPERIMENTAL SETUP & RESULTS

The experimental set-up was developed for the implementation of proposed work as shown below. There were three sub-systems deployed to get connected through wireless local area network in a mesh topology. For the transfer of data between different NodeMCUs, each sub-system was powered up separately means there were three different power sources used here for three sub-systems. Three batteries could be used to power up these nodes. The system was operated on a 5V/ 1Ampere dc power source. The first node was connected with push buttons and OLED display module. It accepted inputs from the push buttons and sent the updated status remotely to other two nodes individually each time a dedicated push button got pressed. Similarly, this node displayed the outputs obtained remotely from other two nodes individually over an OLED display screen as shown.

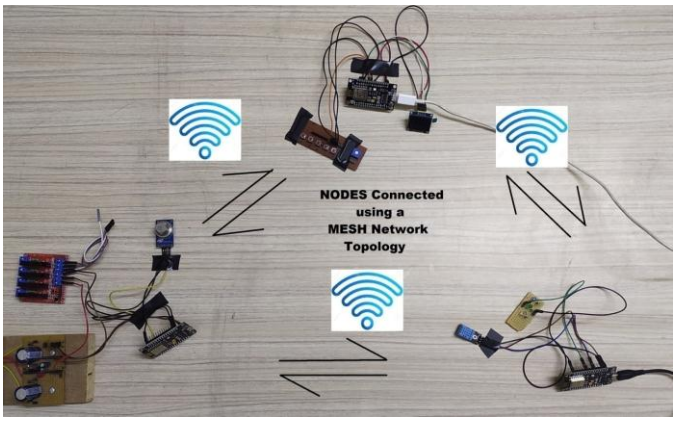

Fig -5: Experimental Setup

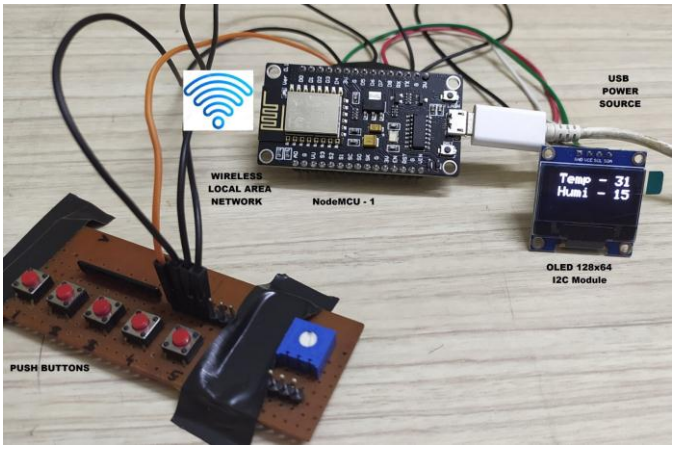

Fig -6: NodeMCU-1 Section

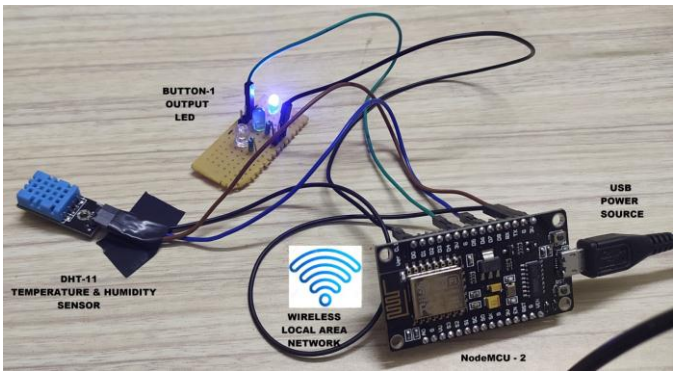

Fig -7: NodeMCU-2 Section

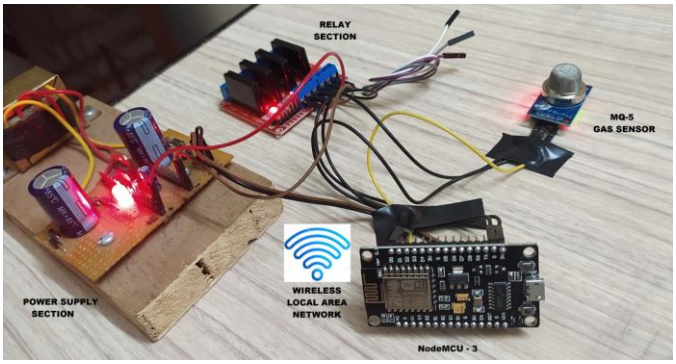

Fig -8: NodeMCU-3 Section

7. CONCLUSIONS

Here it could be easily observed and verified that to establish communication between multiple nodes in a wireless network could be through a router if it were a traditional Wi-Fi network which has its own set of limitations or it could be attained without using any router in between via a mesh network topology which has its own set of benefits. Here it was successfully demonstrated that a local wireless local area network could be established between multiple nodes without even using the internet. The number of nodes could increase to a significant level using this mesh network topology as here the nodes are mutually responsible for relaying each other's transmissions and these interconnected nodes resulted in a much larger coverage area.

REFERENCES

[1] Emanuele Di Pascale; Irene Macaluso; Avishek Nag; Mark Kelly; Linda Doyle, "The Network as a Computer: A Framework for Distributed Computing Over IoT Mesh Networks", IEEE Internet of Things Journal, 2018, Volume: 5, Issue: 3, Journal Article, Publisher: IEEE

[2] Khalid Haseeb; Ikram Ud Din; Ahmad Almogren; Naveed Islam; Ayman Altameem, "RTS: A Robust and Trusted Scheme for IoT-Based Mobile Wireless Mesh Networks", IEEE Access, 2020, Volume: 8, Journal Article, Publisher: IEEE

[3] Xiaofan Jiang; Heng Zhang; Edgardo Alberto Barsallo Yi; Nithin Raghunathan; Charilaos Mousoulis; Somali Chaterji; Dimitrios Peroulis; Ali Shakouri; Saurabh Bagchi, "Hybrid Low-Power Wide-Area Mesh Network for IoT Applications", IEEE Internet of Things Journal, 2020, Early Access Article, Publisher: IEEE

[4] Hamid Al-Hamadi; Mohammad Saoud; Ing-Ray Chen; Jin-Hee Cho, "Optimizing the Lifetime of IoT-Based Star and Mesh Networks", IEEE Access, 2020, Volume: 8, Journal Article, Publisher: IEEE

[5] Rajiv Kashyap; Mohamed Azma ; Jithu G. Panicker, "Ubiquitous Mesh: A Wireless Mesh Network for IoT Systems in Smart Homes and Smart Cities", IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2019, Publisher: IEEE

[6] Muhammad Usman Qadeer; Sheroze Liaquat; Muhammad Ali Shafique; Abdul Rahman Kashif, "Implementation of Wireless Mesh Network for IoT based Smart Homes", International Symposium on Recent Advances in Electrical Engineering (RAEE), 2019, Volume: 4, Publisher: IEEE

[7] Kanitkorn Khanchuea; Rawat Siripokarpirom, "A Multi-Protocol IoT Gateway and WiFi/BLE Sensor Nodes for Smart Home and Building Automation: Design and Implementation", 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES), 2019, Publisher: IEEE