International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

Remote Fire Detection using MQTT Protocol

Priyesh Shivkar

Department of Electronics & Telecommunication Engineering, Vidyalankar Insitute of Technology, University of Mumbai.

***Abstract* –** IoT platform integrates the collected data from various sensors and further analytics is performed on the data and valuable information is extracted as per requirement. Finally the result is shared with other devices for better user experience, automation and improving efficiencies. We have smart appliances, smart cars, smart homes, smart cities, where IoT is redefining our lifestyle and transforming the way we interact with technologies. The future of IoT industry is huge. Now in this generation our lifestyle has become smarter with the smart devices we use in our day-to-day life. So to avoid the severe accidents and losses from fire outbreak, the system has been designed in which there is a continuous monitoring of the temperature and fire. So whenever there is a detection of fire, the flame sensors will detect the fire and simultaneously the temperature sensor will sense the rising temperature and then this information will be monitored and accordingly the necessary action will be taken to extinguish the fire in that particular area. Here the MQTT (Message Queuing Telemetry Transport) protocol, which is TCP based SUBSCRIBE-PUBLISH messaging, which is designed for light weight M2M communication, will be used for the monitoring purpose. Originally developed by IBM, and based on HUB and SPOKE model.

Keywords: IoT, IoT Protocols, MQTT, Temperature sensing, Fire Detection.

1. INTRODUCTION

IoT is influencing our lifestyles from the way we react to the way we behave. From the Air Conditioner's you control from your smart phones to the smart cars providing the shortest route or your smart watch which is tracking your daily activities. IoT is a giant network with connected devices. These devices gather and share data about how they are used and the environment in which they are operated. It's all done using sensors, which are embedded in every physical device. It can be your mobile phone, electrical appliances, barcode sensors, and almost everything you come across in day-to-day life. These sensors continuously emit data about the working state of the devices, but the important question is how do they share this huge amount of data and how do we put this data to our benefit. IoT provides a common platform for all these devices to dump their data and a common language for all the devices to communicate with each other. Data is emitted from all the various sensors and sent to IoT platform security.

Now in this world, there are so many accidents recorded which causes a very great loss. One of them is the fire outbreak which can take place in any environment. So a MQTT network has been designed where in the nodes are been placed in the particular environment where there are more chances of fire outbreak, which continuously senses the temperature. In case of fire outbreak in that particular environment, the temperature rises and the flame sensors detects the fire and the duration of the fire. Now this data from the sensors is been published to the MQTT broker via Node. So basically the concerned person who has an access to the broker will subscribe to this information of the sensors and will monitor this data, and take necessary action in case of fire detection.

1.1 IoT Protocols and Comparison:

- **HTTP** and **Web sockets** are common Internet standards. They can be used to deliver information encoded in XML or JavaScript Object Notation.
- **HTTP** is the foundation of the client-server model used for the Web. Device should include only an HTTP client, not a server. An embedded device should never be set up to receive outside connections to prevent outsider's access to the local network.
- **Web Socket** is a protocol that provides a full-duplex link between client and server. It's part of the HTML 5 specification. Web Socket simplifies much of the complexity around bi-directional Web communication. The downside of these Web protocols is that they're often too data heavy for IoT applications.
- By contrast, **CoAP** was designed especially for use in devices operating on battery or energy harvesting. It works a lot like HTTP.
- **MQTT** is a publish-and-subscribe transport that is extremely lightweight. It helps minimize the resource requirements for IoT device, and can handle unreliable networks. MQTT runs on large networks of small devices that need to be monitored from a back-end server.

1.2 Why MQTT?

- MQTT collects data from various electronic devices and supports remote device monitoring. It is a subscribe/publish protocol that runs over Transmission Control Protocol (TCP), which means it supports event-driven message exchange through wireless networks.
- MQTT is lightweight and has the least header size of 2-byte per message. Where HTTP requires highest power and resource than any other protocols, where MQTT requires lower power and resource and MQTT is designed for low bandwidth devices.
- MQTT offers the highest level of quality of services with least interoperability. MQTT defines three QoS levels: 0- at most once (only TCP guarantee),1- at least once (MQTT guarantee with confirmation), 2- exactly once (MQTT guarantee with handshake).
- MQTT is an established M2M protocol and has been used and supported by the large number of organizations such as IBM, Facebook, Eurotech, Cisco, Red Hat, M2Mi, Amazon Web Services (AWS), InduSoft and Fiorano.[1]

2. DESIGN AND IMPLEMENTATION

In this project, the main aim is to build a network which consumes less power and is also cost effective. Now when it comes to the designing part, the DHT11 module which senses humidity and temperature, and the Flame sensor module which detects fire, are been connected to the ESP8266, which is suitable for the IoT applications. This is considered as a node. Now the MQTT broker which is available free on the internet is been used where the data of the node gets published.

ESP8266: ESP8266 is a low-cost Wi-Fi microchip, with a full TCP/IP stack and microcontroller capability, produced by Espressif Systems in Shanghai, China. This has been designed for mobile, wearable electronics and Internet of Things applications with the aim of achieving the lowest power consumption with a combination of several proprietary techniques. The power saving architecture operates mainly in 3 modes: active mode, sleep mode and deep sleep mode.

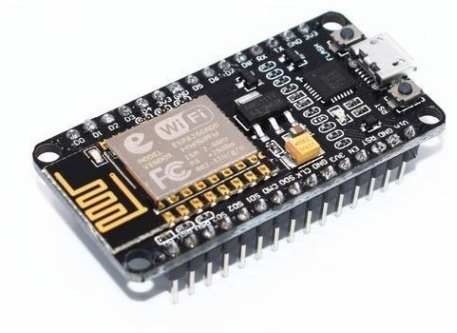

Figure 1: ESP8266 WIFI SERIAL MODULE

DHT11: The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin.

Figure 2: DHT11 Temperature and Humidity Sensor

Flame Sensor: Flame sensor module has photodiode to detect the light and op-amp to control the sensitivity. It is used to detect fire and provide HIGH signal upon the detection. ESP8266 reads the signal and provides alert by turning on LED.

Figure 3: Flame sensor

2.1 Working of the designed system:

PUBLISH PART:

- The Node can publish something and get subscribe from anywhere as long as it is able to connect. For Example, the YouTube channel, it publishes information and we subscribe to that channel so that we get that published information. Now the ESP8266 is connected programmed and sensors are been connected to the ESP8266 module.
- Now this ESP8266 is been connected with another platform in the internet which understands MQTT, where it publishes its address. So it is easy for ESP8266 to send the message as its address is

known to us and this platform could respond easily it knows the path back to our ESP8266.

- Now we have connected DHT11 and flame sensors to the ESP8266 and we monitor the temperature and fire detection in that particular area where the ESP8266 has been placed and this data is getting published to that internet platform. Now this platform has all the data from the sensors. This was the publish part of the protocol.

SUBSCRIBE PART:

- If we want to monitor the temperature of a particular area where the node has been placed, we need to subscribe to that information; basically these information channels are called TOPICS in MQTT protocol, and the internet platform which deals with the MQTT messages is called as MQTT BROKER.
- There are some free MQTT brokers available on the internet and in this case we have used the broker hosted by ADAFRUIT.IO. Now the temperature sensor will publish the data to the BROKER and as well as incase of fire, the flame sensor will detect and publish this information to the BROKER via ESP8266.
- For ease of use, MQTT library has been used on ESP8266. In this case, POP-SHOP client library is preferred, because it is readily available in AURDINO IDE and can be installed in ESP8266.

CODING:

- The sketch of the code is quite straightforward. First stage, we connect to the WiFi as usual. Next we have define the MQTT broker with an address and the port number, (this information is available on the homepage of the BROKER i.e. ADAFRUIT.IO) and then we have to connect to the BROKER, with the username and password of the account of the BROKER page. The password sometimes is also called the KEY, which is available there on the BROKER homepage. Once you create an account on the BROKER, all this information is available there.
- Then this program part is put in loop because if the connection to the broker is lost it automatically reconnects.
- Next, the subscribe command for the particular topic as soon as the connection established between the ESP8266 and BROKER, the temperature and flame sensor data is been published. These are the topics we need to publish to broker, so the topic names are areatemp1 and flame sensor topic names are area1north, area1east, area1wwest and area1south.
- Now the BROKER side, on ADAFRUIT, you have add topics in this format (username/feeds/topics) respectively and we can monitor this data if we subscribe to these topics. This coding is done in Aurdino software using C++ language and then programmed into ESP8266.

We can add dashboard as per the topics to show our feeds visually, as shown in the figure below.

Figure 4: MQTT Broker Dashboard

In the above figure 4, we can see that when there is a fire outbreak in a particular area where the node is placed, there is a temperature variation and the duration of the fire in that particular area is been captured by the flame sensors and this data is been displayed on the dashboard and thus needed action will be taken to extinguish the fire by the concerned person who is monitoring this information.

3. CONCLUSION

In this project, we have successfully designed and implemented the MQTT network for low-cost IoT application of fire detection by using the MQTT protocol as it is power efficient.

Now this designed system can be deployed in the suitable and required environment as it is cost effective as well as power efficient.

REFERENCES

[1] "Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP" Nitin Naik Defence School of Communications and Information Systems Ministry of Defence, United Kingdom.

[2] "Remote Sensing and Controlling of Greenhouse Agriculture Parameters based on IoT" Pallavi S., Jayashree D. Mallapur, Kirankumar Y. Bendigeri Department of Electronics and Communications Engineering.

[3] "Smart Healthcare Monitoring System Using MQTT Protocol" Borade Samar Sarjerao and Amara Prakasarao Department of Electronics and Communication Engineering.

[4] "Communication Protocol Stack for Constrained IoT Systems" Cheena Sharma Dr. Naveen Kumar Gondhi.

[5] "Comparison with HTTP and MQTT on Required Network Resources for IoT" Tetsuya Yokotani, Department of Electronics, Information and Communication Engineering, College of Engineering,Kanazawa Institute of Technology,Nonoichi, Ishikawa, Japan.

[6] "Internet of Things (IoT): Architecture and Design" Ali A. Abed,Computer Engineering Department,University of Basra.

[7] "Networking Protocols and Standards for Internet of Things", Tara Salman.

[8] "Energy efficiency of Machine-to-Machine protocols" Marko Pavelic, Vatroslav Bajt, Mario Kusek, University of Zagreb,Faculty of Electrical Engineering and Computing,Department of Telecommunications, Zagreb, Croatia.