Real Time Monitoring of IOT devices using MQTT protocol and effective dashboard design with database optimization

Abhishek R Manas
Computer Science Department, RV College of Engineering

***Abstract*— With the increasing number of IoT devices being deployed in various applications, monitoring and managing these devices has become a critical task. This paper presents a real-time dashboard that leverages the MQTT protocol, React, Express, and PostgreSQL to monitor IoT devices. The dashboard provides users with a graphical user interface that allows for real-time monitoring of data from various IoT devices. The MQTT protocol is used to facilitate efficient communication between the IoT devices and the dashboard. React is utilised to provide a responsive and interactive user interface, while Express provides a scalable and flexible backend for data processing. PostgreSQL is used to store and manage data collected from the IoT devices, enabling efficient retrieval and analysis of historical data. The proposed solution is effective in monitoring and managing IoT devices in various applications. This real-time dashboard provides an effective solution for monitoring and managing IoT devices, enabling better decision-making, and enhancing operational efficiency in various applications. Database is optimised to handle real time data.**

***Keywords*— IoT, MQTT protocol, Realtime, Dashboard, React, Express, PostgreSQL, Database, Node**

1. INTRODUCTION

Real-time monitoring of IoT devices is critical for gaining valuable insights into device performance and facilitating better decision-making. To achieve this, a real-time dashboard leveraging the MQTT protocol, React, Express, and PostgreSQL has been proposed. The MQTT protocol is a lightweight communication protocol, which enables efficient communication between IoT devices and the dashboard. React, a popular JavaScript library, is used for building interactive user interfaces, while Express, a scalable backend framework, is used for data processing. PostgreSQL, a robust database management system, is used for storing and managing the data collected from IoT devices. The proposed real-time dashboard offers a graphical user interface for real-time monitoring of data from different IoT devices, which can be applied in smart homes, smart cities, and industrial automation. The use of MQTT, React, Express, and PostgreSQL enables efficient data transmission, responsive user interfaces, scalable data processing, and efficient data storage and retrieval. The paper presents the design and implementation of the dashboard and evaluates its performance through experiments, demonstrating its effectiveness in managing and monitoring IoT devices in various applications.

2. LITERATURE SURVEY

This section provides a comprehensive overview of the recent research conducted on dashboard design and MQTT protocol, which can help researchers and practitioners in the IoT domain to better understand the capabilities and limitations of this protocol. We have divided the research studies into six primary areas, which we will describe in detail. These categories include the benefits of using MQTT, a comparison between MQTT and other IoT protocols, the effective dashboard design, the benefits of using React js, the benefits of using Node and Express js and the benefits of using PostgreSQL.

2.1 MQTT PROTOCOL BENEFITS

Prada et al. [1] focused on the lightweight nature of the MQTT protocol when communicating with resource-constrained devices. They developed a module for an educational tool called EjsS (Easy Java/Javascript Simulations) to communicate with an Arduino-based device and evaluated the ability of MQTT to work with low-end devices. The module they developed enabled the educational tool to use MQTT protocol to communicate with the physical device powered by an Arduino microcontroller. Their experimental results demonstrate that the MQTT protocol can successfully facilitate communication between an interactive educational tool running on a web browser and a hardware platform with limited resources, without adding any additional time or complexity to the educators who use it.

Wagle [2] explored the potential of using MQTT protocol in wireless sensor networks that interface with the internet and implement machine learning algorithms over the cloud. They implemented an IoT application that involved ubiquitous sensing, machine-to-machine communication, cloud computing, and semantic data extraction. In their study, they evaluated the advantages, disadvantages, and suitability of MQTT for IoT applications. The authors' findings emphasised the importance of MQTT's message retention and other features for semantic data extraction and the easy integration of new devices. MQTT's availability of topics for subscription and publishing renders data routing procedures largely redundant, thereby eliminating the need for heavy mechanisms to direct data to specific buffers at the program level. Additionally, MQTT's quality of service (QoS) and Last Will and Testament features enhance the protocol's reliability, making it suitable for constraint-bound situations.

R A Atmoko et al. [5] suggested using MQTT as a communication protocol for IoT devices and utilising temperature and humidity sensors to collect real-time data. The data was monitored using a web-based and mobile interface. The study showed that using the MQTT protocol improved the quality and reliability of the collected data. The temperature and humidity parameters were selected as they are often used to monitor environmental conditions. Overall, this study highlights the benefits of using MQTT in data communication protocols for IoT devices.

2.2 COMPARISON BETWEEN MQTT PROTOCOL AND OTHER IOT PROTOCOLS

Yokotani and Sasaki [3] conducted a study to compare the effectiveness of MQTT, which is a protocol based on ICN architecture, with HTTP, which is a legacy protocol. They also suggested ways to improve MQTT for better performance. Their findings revealed that MQTT outperformed HTTP, and they concluded that protocols based on ICN architecture are more suitable for IoT systems.

Luzuriaga et al. [4] conducted an experiment to evaluate the performance of AMQP and MQTT protocols over unstable and mobile networks, focusing on factors such as message loss, latency, jitter, and saturation boundary values. They found that during message bursts, AMQP delivered messages in a LIFO order, while MQTT maintained packet delivery order. They also observed that AMQP prioritises security, while MQTT is more energy-efficient. The authors recommend using AMQP for reliable and scalable messaging platforms over ideal WLANs and using MQTT to connect edge nodes in constrained environments.

2.3 THE EFFECTIVE DASHBOARD DESIGN

Janes, Andrea & Sillitti, Alberto & Succi, Giancarlo [7] demonstrated how to use a GQM Strategies measurement model to create a dashboard that supports users in achieving their business goals. The study suggests that a dashboard's success depends on its perceived usefulness and ease of use, as per the Technology Acceptance Model. The dashboard must include data relevant to the business goals and requires input from management and experienced collaborators. Developing a dashboard is a continuous process, as organisations continually learn and update their business goals, assumptions, strategy, and measurement goals. It is important to continuously assess and update the dashboard to ensure it continues to support business goals.

2.4 BENEFITS OF USING REACT JS

Bhupati Venkat Sai [8] suggested react js is an ideal solution for projects that require component reusability, user interactions, or animations. It is a powerful UI library suitable for small, medium, and large-scale organisations, which is why many companies use it for their long-term business goals. Reactjs is non-risky, responsive, and advanced, with the ability to build large-scale applications with constantly changing data. It offers a virtual browser (DOM) that is faster and more user-friendly than the real one, making it easier to create interactive UIs. Additionally, Reactjs has JSX support, a component-based structure, and many other features that make it an attractive choice for both startups and enterprises.

2.5 BENEFITS OF USING NODE AND EXPRESS JS

X. Huang [9] suggested that Node.js is a server-side JavaScript interpreter that allows developers to create web applications with fast response times and easy scalability. It addresses the limitations of traditional development languages by using its own built-in attributes. One of the key features of Node.js is its event-driven, time loop mechanism, which allows it to perform functions that traditional JavaScript cannot. This includes file systems, modules, packages, operating system APIs, and network communications. Additionally, Node.js utilises a non-blocking I/O model, which allows it to handle multiple requests simultaneously, resulting in faster response times. JavaScript has historically been restricted to the browser environment, but Node.js is a successful attempt to extend its functionality to server-side applications. It has become the preferred choice for web developers due to its unique features and ease of use.

2.6 BENEFITS OF USING POSTGRESQL

Stonebraker, Michael and Lawrence A. Rowe [10] demonstrated that postgreSQL is a database management system that offers several advantages over traditional systems. It provides better support for complex objects, allowing users to store and manipulate data more efficiently. PostgreSQL also offers user extendibility for data types, operators, and access methods, which can be customised to meet specific business needs. Additionally, PostgreSQL provides facilities for active databases, including alerters and triggers, enabling real-time data processing and analysis. It also offers inferencing capabilities such as forward- and backward-chaining. PostgreSQL simplifies DBMS code for crash recovery and produces a design that can take advantage of various hardware configurations. Lastly, PostgreSQL aims to make as few changes as possible to the relational model, ensuring ease of transition and a familiar platform for managing data.

3. PROPOSED METHODOLOGY

This methodology provides a step-by-step guide for building a real-time dashboard for monitoring hardware and IoT devices using React, Express, PostgreSQL, and the MQTT protocol. In this methodology, we will discuss each step in detail to provide a comprehensive guide for building a real-time dashboard for monitoring hardware and IoT devices. By following this methodology, users can create an efficient and scalable solution that enables real-time monitoring and visualisation of data from various IoT devices.

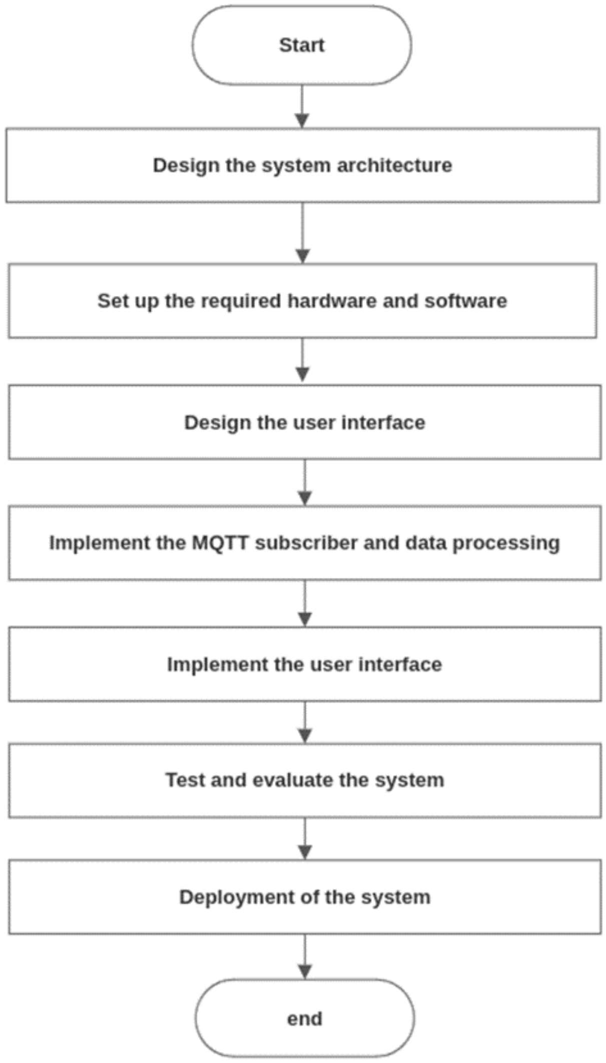

Start
Design the system architecture
Set up the required hardware and software
Design the user interface
Implement the MQTT subscriber and data processing
Implement the user interface
Test and evaluate the system
Deployment of the system
end

Fig.1 Block Diagram of the proposed architecture containing the systematic way to design a dashboard using MQTT protocol.

3.1 Designing the system architecture

The first step is to design the system architecture, which involves identifying the hardware and software components required for the system.

Hardware requirements for the system include at least 4 GB of RAM and 10 GB of available space on the hard disk. An Internet connection is also necessary for the system to function properly. Additionally, an Intel i3 or higher processor is recommended for optimal performance.

Software requirements for the system include compatibility with multiple operating systems such as Windows, Linux, and MacOS. The system also requires NodeJs, which is a virtual environment for running Javascript code, as well as the Node Package Manager.

The system also requires React version 16.0.1 or higher to be installed. For data transmission, the MQTT protocol has been selected. Additionally, PostgreSQL has been chosen as the database for the system.

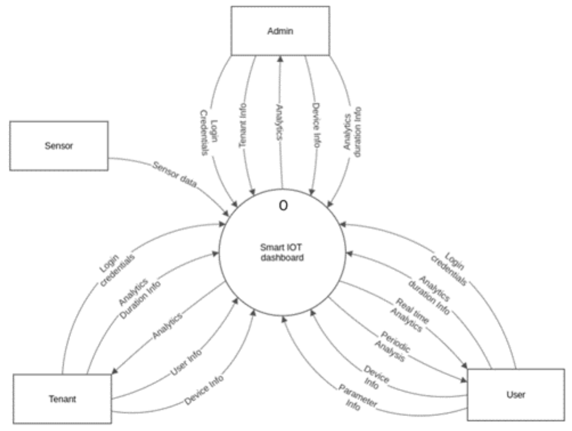


Fig.2 Represents the level-0 data flow diagram which is the visual representation of the information flow through a process or system.

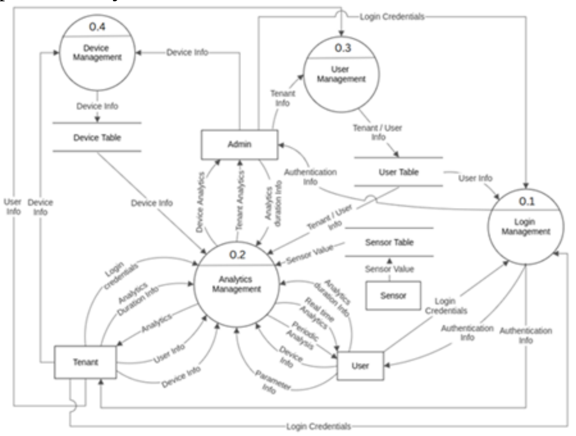


Fig.3 Represents the level-1 data flow diagram which gives more details regarding the sub-process involved in the system.

3.2 Setting up the required hardware and software.

Once the system architecture is designed, the next step is to set up the required hardware and software. This involves configuring the IoT devices, installing and configuring the MQTT broker, setting up the database, and deploying the server infrastructure. This step also includes configuring the security protocols to ensure data privacy and prevent unauthorised access to the system.

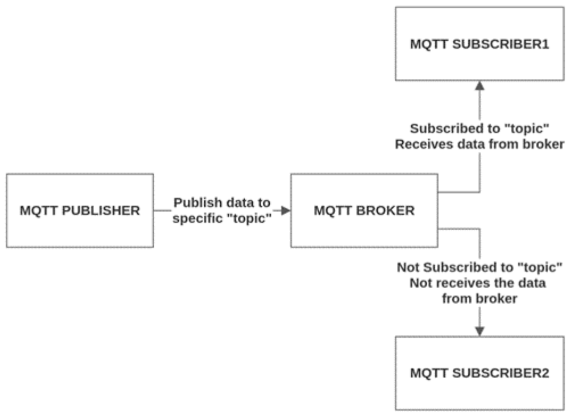

MQTT SUBSCRIBER1
Subscribed to "topic"
Receives data from broker
MQTT PUBLISHER
Publish data to specific "topic"
MQTT BROKER
Not Subscribed to "topic"
Not receives the data from broker
MQTT SUBSCRIBER2

Fig.4 Represents the MQTT protocol architecture which includes publisher, broker and subscriber.

3.3 Design the user interface

The third step is designing the user interface. This involves creating a visual representation of the data to be displayed on the dashboard. This step requires identifying the key performance indicators to be monitored, creating a layout for the dashboard, and designing the visual elements such as graphs, charts, and tables.

3.4 Implementing MQTT subscriber and data processing

The fourth step is implementing the MQTT subscriber and data processing. This involves developing a subscriber to receive data from the IoT devices, processing the data, and storing it in the database. This step also includes developing a mechanism for detecting and handling errors or data anomalies.

3.5 Implementing the user Interface

The fifth step is implementing the user interface. This involves integrating the data processing logic with the user interface to display the data on the dashboard. This step requires developing a front-end application using React to display the data on the dashboard and updating the dashboard in real-time as new data is received. The backend of the system is developed using Express, a popular Node.js web application framework. The backend includes the application logic, API development, and data processing.

3.6 Testing and evaluating the system

The sixth step is testing and evaluating the system. This involves testing the system functionality, performance, and security to ensure that it meets the desired requirements. This step requires developing test cases, conducting tests, and analysing the results to identify areas for improvement. Testing is done using Selenium.

3.6 Deployment

The final step is deploying the system to the production environment. This involves setting up the infrastructure necessary to deploy the system, ensuring that the system is scalable, and configuring the security protocols to ensure that the system is secure. Once the system is deployed, it can be used to monitor data from various IoT devices in real-time.

4. RESULTS AND DISCUSSION

This section provides the results obtained after the implementation of the dashboard.

4.1 Optimised database design to store real-time data

The database has been designed in such a way that it stores the real-time data and after every one hour the data will be flushed out and the average value is stored into another table. The below calculations are done for 1 day.

Table 1: Comparison of memory consumption between normal and optimised database approach

Number of sensors	Normal method (in kilobytes)	Optimised method (in kilobytes)
1	915840	1272
15	13737600	19080
30	27475200	38160

4.2 Dashboard Interface

The dashboard interface is the primary means of communication between the user and the system. It allows users to view real-time data visualisation, view analytical information from the system.

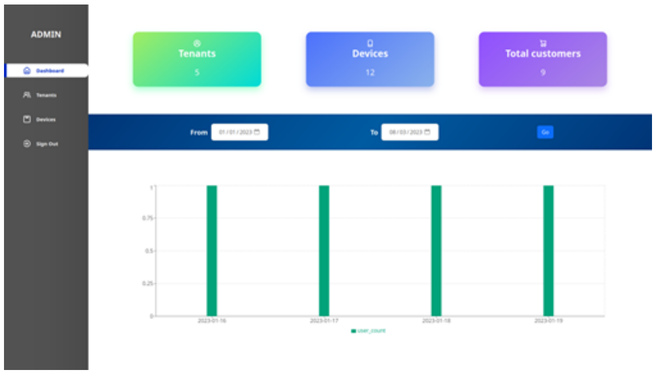
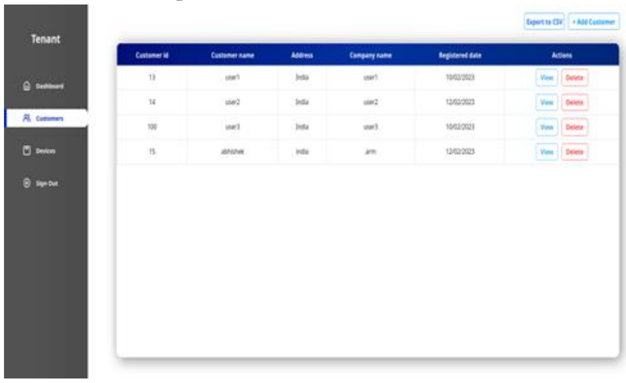


Fig.5 Represents the dashboard interface which includes real-time data visualisation and graphical analytics.

4.3 Analytics Interface

The analytics interface allows users to get analytics of the parameters in the specified duration. Users can also download the report in the form of CSV.



Customer id	Customer name	Address	Company name	Registered date	Actions
13	user1	India	user1	10/02/2023	View Delete
14	user2	India	user2	12/02/2023	View Delete
100	user3	India	user3	10/02/2023	View Delete
15	abhishek	India	arm	12/02/2023	View Delete

Fig.6 Represents the analytics interface which includes parameter analytics for the given duration.

4.4 Admin panel and Tenant section

It allows the admins to monitor the tenants, monitor the devices and to add new tenants to the platform.

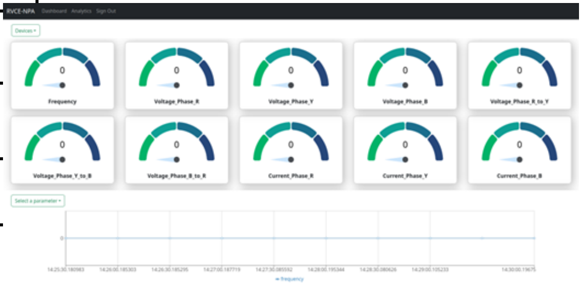


Fig.7 Represents the admin panel where admins can get the tenant analytics and general analytics.

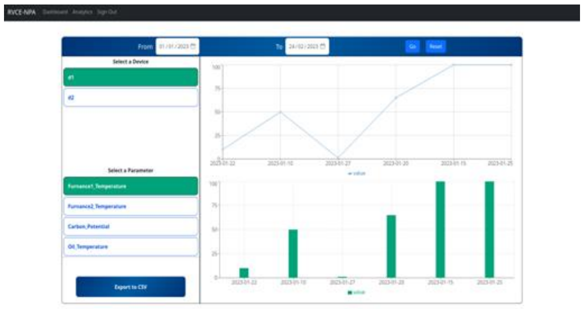
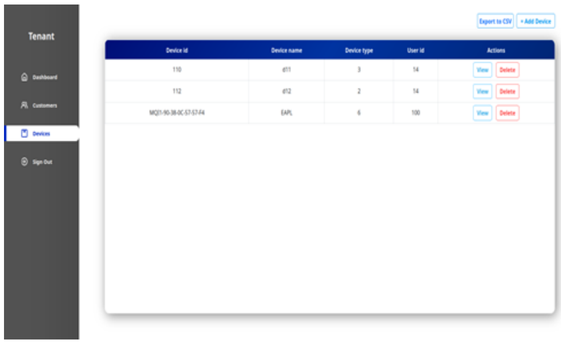


Fig.8 Represents the tenant section where admins can monitor the tenants and can add new tenants

4.5 Devices section

It allows admins and tenants to add and assign devices to the users. Also the details of the devices can be stored in the form of a CSV file.



Device id	Device name	Device type	User id	Actions
110	d11	3	14	View Delete
112	d12	2	14	View Delete
MQI1-90-38-0C-57-57-F4	EAPL	6	100	View Delete

Fig.9 Represents the devices section where admins can monitor the devices and can add new devices.

5. CONCLUSION

In this paper, an attempt has been made to implement a dashboard which can monitor real-time data from the IOT devices efficiently and in an optimised way. The real-time dashboard provides valuable insights into the performance of IoT devices and enables quick and informed decision-making based on real-time data analysis. The system is customizable, secure, and user-friendly, with a powerful dashboard that enables real-time visualisation and historical data analysis. The use of MQTT, a lightweight messaging protocol for IoT devices, provides efficient and reliable data transmission, while PostgreSQL, an open-source relational database management system, provides structured and efficient storage of IoT data. The use of React, a popular JavaScript library for building user interfaces, and Express, a popular Node.js web application framework, provides a powerful and flexible frontend and backend for the system. This dashboard can be improved iteratively as per the requirements, more visualisation tools can be added and efficiency can be increased.

6. REFERENCES

[1] M. A. Prada, P. Reguera, S. Alonso, A. Morán, J. J. Fuertes, and M. Domínguez, ‘‘Communication with resource-constrained devices through MQTT for control education,’’ IFAC-PapersOnLi

[2] S. Wagle, ‘‘Semantic data extraction over MQTT for IoTcentric wireless sensor networks,’’ in Proc. Int. Conf. Internet Things Appl. (IOTA), Jan. 2016, pp. 227–232, doi: 10.1109/iota.2016.7562727.

[3] T. Yokotani and Y. Sasaki, ‘‘Comparison with HTTP and MQTT on required network resources for IoT,’’ in Proc. Int. Conf. Control, Electron., Renew. Energy Commun. (ICCEREC), Sep. 2016, pp. 1–6, doi: 10. 1109/iccerec.2016.7814989.

[4] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni, ‘‘A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks,’’ in Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf. (CCNC), Jan. 2015, pp. 931–936, doi: 10.1109/ ccnc.2015.7158101

[5] R A Atmoko et al 2017 J. Phys.: Conf. Ser. 853 012003, DOI 10.1088/1742-6596/853/1/012003

[6] Atmoko R A 2013 Sistem Monitoring dan Pengendalian Suhu dan Kelembaban Ruang pada Rumah Walet Berbasis Android, Web, dan SMS Semantik 3 (1) pp. 283-290 ISSN 979-26- 02666

[7] Janes, Andrea & Sillitti, Alberto & Succi, Giancarlo. (2013). Effective dashboard design. Cutter IT Journal. 26. 17-24.

[8] Bhupati Venkat Sai Indla | Yogeshchandra Puranik "Review on React JS" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-5 | Issue-4, June 2021, pp.1137-1139, URL: www.ijtsrd.com/papers/ijtsrd42490.pdf

[9] X. Huang, "Research and Application of Node.js Core Technology," 2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), Sanya, China, 2020, pp. 1-4, doi: 10.1109/ICHCI51889.2020.00008.

[10] Stonebraker, Michael and Lawrence A. Rowe. “The design of POSTGRES.” ACM SIGMOD Conference (1986), DOI:10.1145/16894.16888 Corpus ID: 6740973

[11] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego and J. Alonso-Zarate, "A survey on application layer protocols for the Internet of Things", Trans. IoT Cloud Comput., vol. 3, no. 1, pp. 11-17, 2015.
[12] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision architectural elements and future directions", Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645-1660, 2013.
[13] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT CoAP AMQP and HTTP", Proc. IEEE Int. Syst. Eng. Symp. (ISSE), pp. 1-7, Oct. 2017.
[14] S. Bandyopadhyay and A. Bhattacharyya, "Lightweight Internet protocols for Web enablement of sensors using constrained gateway devices", Proc. Int. Conf. Comput. Netw. Commun. (ICNC), pp. 334-340, Jan. 2013.
[15] Y. Xu, V. Mahendran and S. Radhakrishnan, "Towards SDN-based fog computing: MQTT broker virtualization for effective and reliable delivery", Proc. 8th Int. Conf. Commun. Syst. Netw. (COMSNETS), pp. 1-6, Jan. 2016.