International Journal of Scientific Research in Science, Engineering and Technology

Print ISSN - 2395-1990
Online ISSN : 2394-4099

OPEN ACCESS

Available Online at : www.ijsrset.com
doi : https://doi.org/10.32628/IJSRSET2411114

Empowering IoT Healthcare Solutions with Enhanced Flexibility and Efficiency

Dr. Levina T*[1], Obed Junias[2], Savitha S[3]

*[1]Associate professor, Department of Computer Science & Engineering/K N S I T Bangalore, Karnataka, India
[2]Member Of Technical Staff, ORACLE, Bangalore, Karnataka, India
[3] Associate professor, Department of Computer Science & Engineering/K N S I T Bangalore, Karnataka, India

ARTICLEINFO

Article History :

Accepted: 15 Jan 2024
Published: 31 Jan 2024

Publication Issue :
Volume 11, Issue 1
January-February-2024
Page Number :
132-140

ABSTRACT

This research proposes a architecture called HealthDash for developing Internet of Things (IoT) solutions in healthcare. The main Aim of this research is to over come the complexity problems in creating and keeping up with computational health solutions with increased flexibility, where scenarios are often complex and dynamic. HealthDash aims to unify technologies at both the fog and cloud layers, providing dynamism in creating IoT applications and reducing data transmission between these layers. To specify which layer (fog or cloud) data flows will be executed, the suggested approach makes use of the data flow-oriented programming paradigm, empowers IoT healthcare solutions with greater flexibility, efficiency, and privacy.The architecture aims to simplify the creation and maintenance of computational solutions in the complex and dynamic healthcare domain. HealthDash is designed to address challenges such as low latency, data transmission efficiency, and privacy concerns in remote patient monitoring scenarios. Enabling dynamic creation of IoT applications in both fog and cloud layers. Reducing data transmission between fog and cloud layers. Utilizing a unified data flow-oriented programming paradigm.

Keywords : Internet of Things(IoT), Cloud Computing(CC), Fog Computing, HealthDash.

I. INTRODUCTION

Healthcare systems worldwide strive to improve the quality of life for patients with chronic diseases and reduce healthcare costs. Remote patient monitoring using IoT technology has emerged as a promising solution for achieving these goals. However, traditional cloud-based approaches may not be suitable for all IoT applications, especially those requiring low latency and tighter coupling between events and actions. Fog computing bridges the gap between cloud and IoT devices by distributing computational resources closer to the edge of the network, offering benefits like reduced response time, improved data privacy, and bandwidth efficiency.

The idea of the Internet of Things (IoT) has recently sparked the creation of technologies to allow the

remote monitoring of these patients [2]. One of these approaches suggests virtualizing "things" and utilizing cloud services, providing a clear abstraction of gadgets and a standard programming language to simplify the development process. However, some IoT applications require processing closer to the device and a tighter coupling between events and actions, and these applications do not work well with this technique.[3].

In the modern age of computers applied to detect health issue,especially in monitoring patients remotely for the treatment of diseases at home, the use of the cloud raises some of the difficulties such as patient reaction time, the amount of data transmitted over the network, as well as issues with privacy and security of the data. Response time is essential when monitoring patients for conditions like heart disease, and if communication with the cloud is not possible in this case, the patient's health could be significantly compromised. Utilizing technology that propagates from the cloud layer to the fog layer in this way contributes to the re-usability and extensibility of components, which helps to simplify the development process[1].

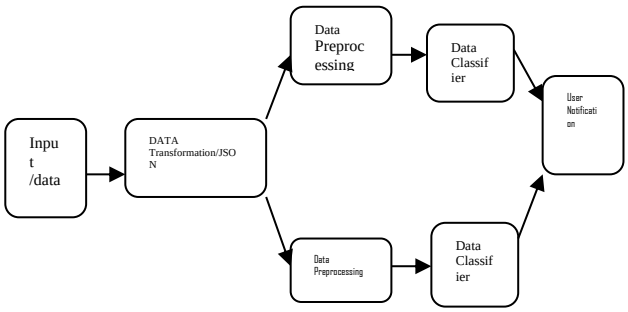

Figure 1. Data Flow for Monitoring a Patient's Health Parameters.

IoT solutions is using in healthcare can be significantly impacted by some of the restrictions mentioned earlier. Implementing a "fog" layer made up of devices placed between the cloud and the end user is one way to circumvent these difficulties. These devices, located closer to the data source, take on the responsibility of processing service requests at the network's edge. Fog computing acts as a bridge between devices and the cloud, enabling the processing of network requests in closer proximity to the source [4].In scenarios where health services require rapid response times and are highly sensitive to latency, direct communication with the cloud becomes impractical. This is where fog computing becomes pivotal, as it facilitates data processing at the network's edge. For instance, consider the prompt detection of cardiac arrhythmias that necessitates immediate notification. Relying on cloud-based processing could jeopardize the patient's well-being due to potential communication delays.

The fog layer also serves to preprocess raw data, transmitting only pertinent information to the cloud. This approach enhances the quality of transmitted data and reduces the strain on bandwidth usage. For example, a service focused on identifying patient falls through inertial sensors like gyroscopes and accelerometers could process data within the fog layer. Only relevant details would then be forwarded to the cloud, such as a notification of a fall event.

Additionally, the fog approach offers an advantage in ensuring data privacy. Unlike hosting patient data in the cloud, which exposes the information to cloud administrators at some point, the fog approach allows data processing within layers controlled by users.

Given the diversity of data generated by disparate devices, interoperability concerns emerge. The fog layer must support a range of equipment from diverse manufacturers, models, operating systems, and communication protocols in order to address this. This necessitates the establishment of a unified data structure, enabling information to be labeled and validated. Such an approach enhances the services provided by the cloud platform.

II. RELATED WORKS

The work described in reference [5] introduces a framework designed for sharing and processing health-related information that originates from various data sources. To enhance control over privacy, security, and

information sharing concerns, the authors incorporate a fog layer between the user and the cloud, thus preventing unnecessary information flow. They also develop a mobile application that captures data at specified time intervals for the purpose of evaluating their proposal. Once collected, the data undergoes pre-processing before being transmitted to the cloud. However, the framework lacks flexibility in allocating specific operations to different layers, and all functionalities are tailored to a particular platform.

A distinct approach known as Mobile Fog is presented by researchers in reference [6]. This programming model enables developers to specify the execution of applications across diverse fog computing devices, supporting the dynamic allocation of computational resources. Despite employing a uniform language for operations across all layers, this model demands programming expertise to create the necessary algorithms. Additionally, it is confined to a specific application and lacks the generality needed for extension to other contexts.

In the domain of fog computing architecture, a notable contribution is the development of an intelligent gateway named UT-GATE, as outlined in reference [7]. UT-GATE operates within a fog computing architecture and provides various functionalities like local storage, real-time data processing, and data mining. However, these functionalities are implemented independently within each layer (fog and cloud). Notably, there is no provision to determine the appropriate layer for executing a particular function. The case study presented showcases the direct implementation of noise reduction functions for ECG signals within the gateway, utilizing the Python programming language. This approach results in fixed data processing locations predetermined by the initial programming.

Lastly, the architecture proposed in reference [8], known as iFogSim, is structured with multiple layers and builds upon the principles of fog computation.

III. Data Flow Oriented Programming

Programming paradigm called "Data Flow Oriented Programming" (DFOP) that is applied to the domain of health data analysis, particularly in the context of an application called HealthDash. DFOP seems to provide a way to manage the complexity of developing programming code for health data analysis by abstracting concerns and encapsulating logic, making it easier to create applications without requiring extensive programming knowledge.

1. Data Flow Oriented Programming (DFOP): This is a programming paradigm that focuses on the flow of data between different processing components. Instead of explicitly writing code that specifies the control flow, you design your program as a series of data transformations and the dependencies between them. This can help simplify the development process and make the program more intuitive to understand.

2.Health Data Analysis: Health data analysis involves processing and extracting insights from medical and health-related data. This data can come from various sources such as patient records, medical tests, wearable devices, and more. Analyzing this data can help healthcare professionals make informed decisions and improve patient care.

3.Abstraction of Concerns: Abstraction in programming refers to the practice of hiding complex implementation details behind a simpler interface. It seems that DFOP in the context of health data analysis allows developers to abstract away some of the complexities, allowing them to focus on higher-level logic rather than getting bogged down in technical details.

4.Encapsulation of Logic: Encapsulation involves packaging the data and methods that operate on the data into a single unit, or object. In this context, encapsulating logic could mean grouping together the data processing steps and functions required for health data analysis into manageable units.

5. Flow-Based Approach: The flow-based approach involves representing the program's logic as a series of connected processing nodes. Each node performs a specific data transformation or action, and the connections between nodes represent the flow of data between them. This can make the overall program structure more visual and easier to comprehend.

6.HealthDash Application: HealthDash is mentioned as an example of an application that uses the DFOP approach for health data analysis. It appears to break down the problem solution into a logical sequence of steps, which can help developers focus on higher-level aspects while still allowing for specific function implementations when needed.

7. User Accessibility: One notable advantage of this approach is that it makes it easier for a wider range of users, including non-specialists, to extend applications more quickly and efficiently. This is particularly valuable in the field of health data analysis, as it can enable healthcare professionals who may not have extensive programming knowledge to contribute to the development and customization of tools like HealthDash.

8. Dynamic Tools for Monitoring Patients: The flow-based approach used in HealthDash seems to facilitate the creation of dynamic tools for monitoring patients with varying clinical conditions. This likely means that the application can adapt its analysis and presentation based on the specific health conditions and data inputs, enhancing its utility in healthcare scenarios.

Figure 1 in the paper represents a portion of the software used to analyze health data[1]. The data flow starts with an input node that receives data. The data is then transformed from text to JSON format, classified according to the type of reading, and analyzed for abnormal readings. If an abnormal reading is detected, a notification is sent to the user[6].

The diagram visually shows the sequence of data processing steps for health data analysis using the Data Flow Oriented Programming approach.

This data flow diagram is used to monitor a patient's health parameters and is represented as a flowchart. The diagram consists of several interconnected nodes and processes that handle the data flow.

- Input Node: The data flow begins with an input node, which receives data related to the patient's health parameters, specifically blood pressure and blood glucose information. This data is received via the MQTT (Message Queuing Telemetry Transport) protocol.
- Data Transformation: The received data in text format is transformed into JSON format for easier processing and analysis.
- Data Classification: The transformed data is then classified based on its type (blood pressure or blood glucose).
- Anomaly Detection: The classified data is analyzed to detect any anomalies or abnormal readings. If an abnormal reading is detected, the system triggers a notification process.
- User Notification: In case of abnormal readings, a notification is sent to the user (possibly a healthcare professional or the patient) to take appropriate action.

9. Please note that the actual visual representation of this data flow diagram would provide a more detailed and intuitive view of the process

IV. Health Dash Architecture Overview

HealthDash comprises three layers: the "things" layer with IoT devices, the fog computing layer with intelligent gateways, and the cloud layer with servers for data management. The architecture provides component reuse and extension by integrating development tools from the cloud to the fog layer. The administrative interface enables management of data flows and schemes as well as the creation of patient-specific care plans.

Figure 2 presents the HealthDash architecture, which consists of three layers: the "things" layer, the fog computation layer, and the cloud layer. The "things" layer includes IoT devices, while the fog layer comprises intelligent gateways communicating with these devices. The cloud layer consists of servers responsible for managing the system. The architecture allows developers to choose the execution layer for data flows.

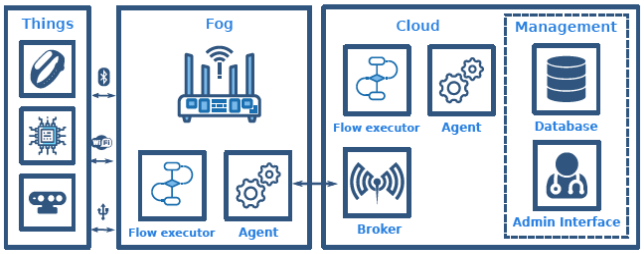

Figure 2 presents the HealthDash architecture

The cloud layer is composed of the following components:

1. Admin Interface: The solution as a whole is controlled by this element.
2. Data flow execution by a flow executor.
3. Database: utilized to keep data on hand.
4. Broker: communicate with the fog layer to get data.
5. Agent: prevent data stream modification.

The HealthDash architecture composed by three main layers: the "things" layer, the fog computing layer, and the cloud layer. Here is the description of the architecture:

- Things Layer: This layer includes IoT devices, such as sensors for monitoring health parameters, connected to a gateway.

- The things layer's IoT devices connect with the fog layer's intelligent gateways via the fog computing layer. The gateways are responsible for local data processing, pre-processing, and some level of analysis. They act as a bridge between the IoT devices and the cloud.

- Cloud Layer: Servers for collecting, storing, and controlling the data gathered from the IoT devices are part of the cloud layer.

It also hosts the HealthDash application, including the Admin Interface and Flow Executor components.

- Admin Interface: The Admin Interface allows the system administrator and health coaches to manage and configure the HealthDash system. It includes functionalities for managing data schemes, data flows, patient records, and care plans.

- Flow Executor: The Flow Executor is responsible for controlling the execution of data flows in both the fog and cloud layers. It communicates with the Agent component in the fog layer to modify the data flows dynamically.

- Data Schema: This component allows modeling and versioning of data schemas for different types of health data collected from the patients.

- Data Flow: This component represents the data flow of the application, specifying the sequence of steps for processing and analyzing the health data.

- Agent: The Agent is responsible for controlling data flow execution in the fog layer and acts as a communication interface between the fog and cloud layers.

The Agent enters a listening state on the $CONFIG/TOKEN after successfully verified for the authentication of the Broker, where the TOKEN acts as a communication token for authentication. A call to the Flow Executor API is made when a new message is

received, after the Agent has verified that it is properly structured. The Flow Executor API, which is implemented as an instance of Node-RED, is responsible for managing the execution of the data flow lifecycle in the fog layer.The Flow Executor interacts with the Agent and modifies the data flow as necessary using this API. Each flow [1].

DataSchema, DataFlow, and Record are just a few of the classes in the data package that are concerned with the organization and storage of data. The "Coaching" package, which also includes the classes Patient and CarePlan, contains classes that are connected to coaching. The "Gateway" package contains classes related to fog devices.
The many data schemas that HealthDash may accept can be modelled using the DataSchema class. Each schema has a versioning code and a URI (Universal Resource Identifier) that uniquely identify it.This allows HealthDash to handle multiple versions of data schemas and adapt to updates in the schema for a given model.

The DataFlow class is used to represent the DataFlows used in care plans. Each data stream has a name, a unique identity, and a representation that is saved as a string that has been JSON encoded. Additionally, the DataFlow class has a field that specifies whether it should run in the cloud or fog layer.

V. RESULT ANALYSIS

In the evaluation of HealthDash, the focus was on monitoring chronic patients at home, with a specific emphasis on collecting data related to blood pressure and blood glucose levels. This evaluation aimed to address two crucial aspects: response time and data transmission. The comparison revolved around utilizing programming with data flow within both the fog and cloud layers.

A. Monitoring Of Chronic Diseases

This will helps monitoring of individuals chronic illnesses within their home environment calls for a comprehensive strategy that integrates various information technologies. This strategy involves transmitting the patient's health data from their home to the appropriate medical facility without the need for a physical presence of healthcare professionals. Instead, patients or their family members can take the responsibility of collecting / transmitting the monitored data.

Research highlights the favorable outcomes of this remote monitoring approach, benefiting both the patient's health and the overall healthcare process. Technologically, two primary methods are commonly used: local processing through fog computing facilitated by a gateway, and direct data processing within the cloud. The choice between these methods depends on factors like the nature of the collected data. Implementing these technologies, however, poses challenges related to response time, security, privacy, and data transmission.

In the evaluation of the proposed architecture, the same data flow was tested using two distinct approaches: one involving anomaly detection in the cloud layer, and the other employing anomaly detection in the fog layer, as illustrated in Figure 5. One significant advantage of this architecture is its utilization of the same technology (data flow programming) to define the application logic in either layer.

Figure 3. Administrative interface

The experimental process encompasses steps such as (1) receiving data, (2) packaging the data, and (3) transmitting the data to the cloud. Subsequent steps involve (4) unpacking the data within the cloud, (5) anomaly detection, and (6) issuing notifications. The specific layer and sequence of executing each step depend on the chosen approach.

To conduct the experiments, two different strategies for processing sensor data in a health monitoring system were evaluated, focusing specifically on blood glucose and blood pressure measurements. The data flow diagram (Figure 2) illustrates the steps involved in both the fog and cloud layers of the system. These steps encompass the reception, packaging, and transmission of sensor data to the cloud. In the cloud-based processing approach, sensor data is encapsulated in the fog layer and then forwarded to the cloud for anomaly detection. In contrast, the fog-based approach processes sensor data locally, and only if anomalies are detected, the data is sent to the cloud for further action and notification.

B. Experimental Setup

The experimental setup involved selecting three different cloud providers—Azure, Digital Ocean, and Google Cloud—each configured with similar virtual machine settings. Communication with the cloud was facilitated by an open-source distributed message broker MQTT called EMQ X, known for supporting scalability and high availability in IoT applications. A simulator was developed to generate blood glucose and blood pressure data, with adjustable parameters such as the number of concurrently connected users, the frequency of daily readings, and the proportion of abnormal readings.

The simulator was tested with varying parameters, including 100, 1000, and 2000 simultaneous users, generating 3 blood glucose and 3 blood pressure readings each, with 20% of them considered abnormal. This resulted in simulated data ranging from 600 to 12000 records in different test scenarios. The evaluation primarily focused on observing the average response time, total data transmitted, and privacy considerations for both fog-based and cloud-based implementations, each with anomaly detection in their respective layers.

$$Lat_{fog|cloud} = T_{preproc} + T_{trans} + T_{proc} \quad (1)$$

It seems like you're discussing the calculation of latency in health monitoring systems. Latency refers to the time delay that occurs when transmitting data or performing operations in a system. In this case, latency is calculated using Equation 1, which takes into account various time components involved in the process. Let me break down the components and equation for you:

Equation 1:

Latency = $T_{preproc} + T_{trans} + T_{proc}$

Where: **Tpreproc:** Average data pre-processing time in the local fog layer, measured in milliseconds (ms).

- **Ttrans:** Average data transmission time to the cloud layer in milliseconds (ms).

- **Tproc:** Depending on the selected technique, the average time in milliseconds (ms) for data analysis and decision-making in the cloud or fog layer.

Breaking it down further:

- **Tpreproc**: This is the time taken to preprocess the data before transmission. It includes tasks like data cleaning, filtering, and formatting to make the data ready for analysis.

- **Ttrans:** This is the time it takes to transmit the preprocessed data from the local fog layer to the cloud layer. It accounts for network latency and bandwidth limitations.

- **Tproc:** This is the time required for the actual data analysis and decision-making process. It involves

performing computations, running algorithms, and making decisions based on the received data.

By these three components, you can estimate the total latency involved in the health monitoring system. Keep in mind that minimizing latency is crucial in real-time applications like health monitoring to ensure timely and accurate responses.

Overall, the study aims to compare the performance of the two approaches and identify any differences in terms of latency, data transmission, and privacy to determine which implementation is more suitable for the health monitoring system. Results, summarized in Table 1, demonstrate that the transmission time is really what determines the delay.

In the HealthDash architecture, it appears that a test was run to compare the volume of data sent from the gateway to the cloud layer. The results are presented in Table 1, and it shows that the fog-based approach significantly reduces the bandwidth utilization compared to the cloud-based approach.

According to the data obtained, there is an approximate 78% reduction in data traffic when using the fog-based approach for a 1-day monitoring period. This means that the fog architecture efficiently processes and filters the data closer to the source (gateway) before sending it to the cloud, which reduces the overall data transfer volume.

The degree of bandwidth reduction might change depending on the kind of communications being examined. When employing the fog-based technique as opposed to the cloud-based approach, some cases with greater sampling rates may incur even more drastic bandwidth reductions. This is so that the least amount of data is transferred to the cloud possible. The fog architecture may analyse and aggregate the data locally.

Table 1. Latencies for clouds and fog with data traffic on average

Users	Lat_{cloud}(ms)				Lat_{fog}(ms)				KBtrafficked	
	$T_{preproc}$	T_{trans}	T_{proc}	T_{total}	$T_{preproc}$	T_{trans}	T_{proc}	T_{total}	*Cloud*	*Fog*
100	0.10	17.05	0.19	17.34	0.17	7.12	0.14	7.43	139.31	29.66
1000	0.08	172.87	0.14	173.09	0.18	169.22	0.12	169.52	1398.91	297.70
2000	0.10	279.91	0.19	280.2	0.18	159.82	0.11	160.11	3770.69	596.01

VI. Conclusion

HealthDash presents a novel IoT architecture for healthcare solutions, offering dynamism and efficiency in developing applications while reducing data transmission between fog and cloud layers. The data flow-oriented programming approach simplifies development and customization, making HealthDash a promising tool for remote patient monitoring and other healthcare applications.

In conclusion, the work that has been given introduces the HealthDash architecture, which was created for creating IoT solutions for the healthcare industry. Data flow-oriented programming is used in the design to integrate technologies from both the cloud layer and the fog layer. Its primary goals are to simplify and increase adaptability while developing and sustaining computational solutions for complicated and dynamic healthcare contexts.

The key contributions of HealthDash are:

1. Dynamism in IoT application creation: HealthDash facilitates the dynamic creation of IoT applications, offering a unified approach in both the fog and cloud layers, streamlining the development process.
2. Data volume reduction: HealthDash facilitates data analysis and filtering at the network's edge, thereby reducing the amount of data that needs to be transmitted between the fog and cloud layers. This results in significant bandwidth conservation.

The suggested design has the potential to improve user data security and privacy. HealthDash enables localized processing of sensitive data before transferring only relevant events to the cloud by allowing users to select at what level of the application each type of information will be treated.

Future actions will involve simulations with more users to demonstrate scalability and real-world chronic patient monitoring at home. Additionally, new privacy controls will be introduced to let HealthDash customers choose which information may be sent to the cloud and which processes can be completed.

Overall, HealthDash demonstrates potential in revolutionizing IoT solutions in the healthcare domain, offering a more streamlined, flexible, and secure approach to data processing and analysis.

VII. REFERENCES

[1]. Levina, P. Kaliyamoorthi, Vinjamuri S.N.CH. Dattu, G Dineshnath, S. Jaiganesh, Dinesh Chandra Dobhal, C. R. Rene Robin. "A Novel Architecture for Developing IoT Solutions Applied to Healthcare", 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), 2023 Publication

[2]. Aazam, M. and Huh, E. N. (2015). E-HAMC: Leveraging Fog computing for emergency alert service. In 2015 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2015, pages 518–523. IEEE.

[3]. Ahmad, M., Amin, M. B., Hussain, S., Kang, B. H., Cheong, T., and Lee, S. (2016). Health Fog: a novel framework for health and wellness applications. The Journal of Supercomputing, 72(10):3677–3695.

[4]. Hong, K., Lillethun, D., Ramachandran, U., Ottenwa¨lder, B., and Koldehofe, B. (2013). Mobile fog: a programming model for large-scale applications on the internet of things. In Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computingMCC '13, page 15, New York, New York, USA. ACM Press.

[5]. Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., and Lilje- berg, P. (2018). Exploiting smart e-Health gateways at the edge of healthcare Internet- of-Things: A fog computing approach. Future Generation Computer Systems, 78:641– 658.

[6]. Gupta, H., VahidDastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Software: Practice and Experience, 47(9):1275–1296.

[7]. Sousa, T. B. (2012). Dataflow Programming Concept, Languages and Applications. Doc- toral Symposium on Informatics Engineering, 7(November):13.

[8]. Liu, L., Stroulia, E., Nikolaidis, I., Miguel-Cruz, A., and Rios Rincon, A. (2016). Smart homes and home health monitoring technologies for older adults: A systematic review. International Journal of Medical Informatics, 91:44–59.

[9]. Farahani, B., Firouzi, F., Chang, V., Badaroglu, M., Constant, N., and Mankodiya, K. (2018). Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare. Future Generation Computer Systems, 78:659–676.

[10]. Giang, N. K., Blackstock, M., Lea, R., and Leung, V. C. (2015). Developing IoTapplicati- ons in the Fog: A Distributed Dataflow approach. Proceedings - 2015 5th International Conference on the Internet of Things, IoT 2015, (January 2016):155–162.

Cite this article as :

Dr. Levina T, Obed Junias, Savitha S, "Empowering IoT Healthcare Solutions with Enhanced Flexibility and Efficiency", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 11 Issue 1, pp. 132-140, January-February 2024. Available at doi : https://doi.org/10.32628/IJSRSET2411114

Journal URL : https://ijsrset.com/IJSRSET2411114