Tampere University

Mohammad Aqib Hassan

INVESTIGATE COMMUNICATION PROTOCOLS FOR IOT APPLICATION
Sensor Based Review

Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
May 2023

ABSTRACT

Mohammad Aqib Hassan: Investigate Communication Protocols for IoT Application: Sensor Based Review
M. Sc. Thesis
Tampere University
Master's Degree Programme in Software, Web & Cloud
April 2023

Internet of Things (IoT) is the hot topic in the field of technology nowadays. We want everything to be automated in our daily life. A huge revolution has been brought by IoT in the history of Internet. IoT enables smart devices to collect specific data and exchange data among them. Exchanging data between smart devices usually occurs through some IoT communication protocol. Efficiency of a protocol depends in so many things such as Data Packet Rate, Security, Bandwidth, Communication overhead, Support of Quality of Service (QoS) level. In this research, some IoT protocols will be compared based on their efficiency and the protocols can be MQTT (Message Queue Telemetry Transport), CoAP (Constrained Application Protocol), AMQP (Advanced Message Queuing Protocol). RuuviTag sensor (a wireless BLE device) will be connected to a mobile app though BLE (Bluetooth Low Energy). Though these protocols data from RuuviTag sensor will be transmitted among clients. These protocols will be compared based on their implementation complexity, data accuracy, application scalability, security. These comparison metrices among protocols will reveal the efficient protocol. The findings of this research will facilitate to a better understanding of the efficiency elements that are involved with IoT communication protocol by illuminating their performance traits and potential drawbacks. This research aims to identify the most suitable protocol based on the specific requirements of the endeavor. By carefully evaluating and analyzing the needs of the investigation, this study will contribute to the selection of an appropriate communication protocol. This research analyses the gaps and barriers of selecting an efficient protocol, and in order to improve the process this research will propose potential directions for future research.

Keywords: BLE, MQTT, CoAP, AMQP, IoT, IoT Communication Protocols

CONTENTS

List of Figures

1. INTRODUCTION

In the modern era of wireless telecommunication, Internet of Things (IoT) is the rapid growing paradigm and gaining the ground quickly in this technology. IoT is a concept that allows high level interaction among distributed things. IoT enabled devices became part of smart device systems since IoT incorporates machine-to-machine (M2M) communication. IoT includes embedded system that can work with different kinds of data via sensor [1].

1.1 Motivation

IoT has a goal of ubiquitous computing that helps to interconnect physical devices though a network which can be internet. IoT technology consists of a large number of smart devices. Smart devices are collaborative electronic gadgets that support daily activities and comprehend simple human commands. There are some key features that makes a device smart. These key features can be categorized into context-awareness, device connectivity and autonomy [2]. Along with these three key features smart also comes with features like learning, remote accessibility and so on. Most of these devices comes with different issues and those issues can be inefficient data processing ability, short in power life, small range, less compatibility, lack of security and many more. Thus, an implementation way is required that can carry out these conditions more efficiently.

These smart IoT devices produce large volume of data. The data they produce are from the surroundings of the devices. Usually, smart devices are dedicated to produce specific kind of data. After producing the data smart devices need to communicate with each other to transmit those data. For communication among them a communication protocol is necessary. With communication protocol these data can be shared efficiently and flexibly. Focus area for this study will be communication protocol in application layer. Along with some standard measurement of communication protocols performance issue, latency, power consumption and security will also be in review [3]. This comparative analysis on IoT communication protocol will help to find best communication protocol for smart devices.

1.2 Purpose

IoT network is dependent on networking, communication and sensory technology and is reliant on the number of linked devices. The advancement of the Internet of Things has been significantly aided by developments in the field of wireless sensor networks. Development of a variety of different devices and technologies like RFID (Radio Frequency Identification), NFC (Near Field Communication), BLE (Bluetooth Low Energy), ZigBee, smart phones, cloud computing, Wi-Fi and so on are employed to support IoT as shown in the Figure 1.

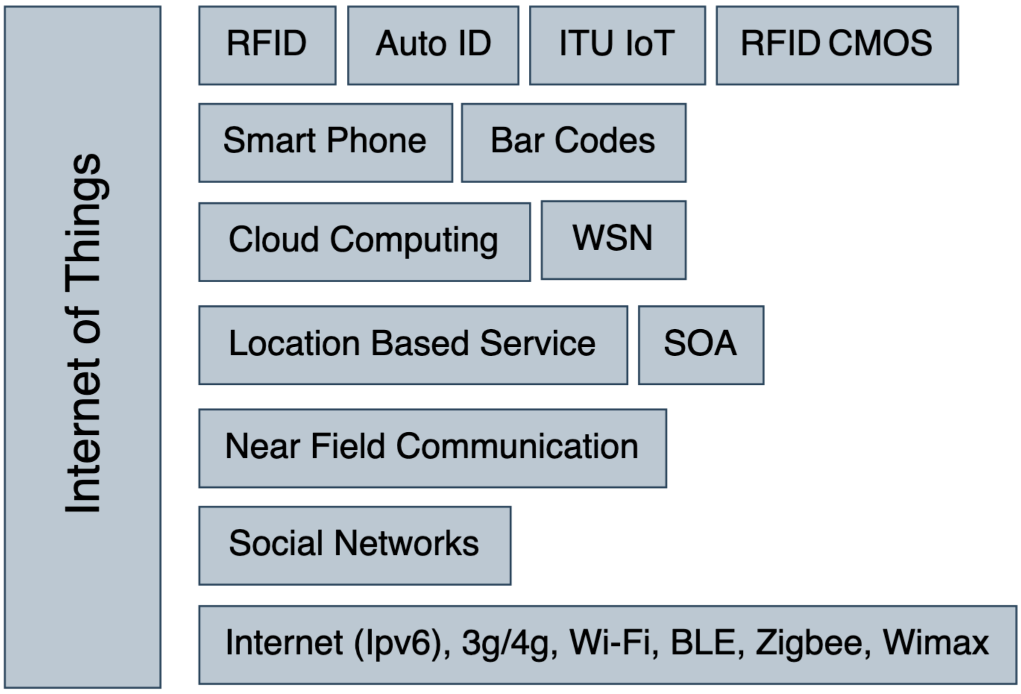


Figure 1. *Technologies connected to IoT [4].*

IoT is, therefore, becoming more popular across the enterprise, for instance in pharmaceutics, manufacturing, retail, and logistics. Due to the rapid growth of sensor networks, smartphones and wireless communication, a growing number of smart devices and networked things are now included on the Internet of Things. IoT-related technologies are having an enormous impact on enterprise system technologies and Information and Communication Technology (ICT) because of these advancements.

Due to these developments, there are new physical objects referred as sensors. These sensors are developed with advanced technologies. RuuviTag sensor is a wireless BLE (Bluetooth Low Energy) device, developed by a Finnish start-up company that measures temperature, air humidity, air pressure and movement. RuuviTag sensor track data using on ESP32 Bluetooth Low Energy Tracker Hub. RuuviTag sensor will be connected to a cross platform mobile app. A gateway named Ruuvi Gateway which is also included in this research and can be connected to the mobile app. Ruuvi Gateway will receive data from RuuviTag sensor and send the data to the connected mobile app. Various communication protocols such as MQTT, CoAP and AMQP will be assessed to accomplish the best protocol for these cases. Based on different metrices and requirements these protocols will be judged to be the fittest one.

1.3 Research Design

The Internet of Things (IoT) is a network that refers to billions of physical devices that collect and share data via internet such as different sensors, actuators and many more. The advancement has led to several new innovations that have made it more efficient, comfortable, and dependable. The “Thing” in IoT can be any object having a sensor implanted in it that has the capacity to gather data and communicate it across the network without operator intervention [5]. A great transformation IoT brings to our life by making revolution in the field of automation. One good example of IoT is that it has the concept of Smart Home Systems (SHS) and appliances that includes smart devices. SHS is an automation system that can check and monitor both indoors and outdoors data of home [6]. Smart devices can take decision according to the instruction what to do with those data such as filtering, modifying and so on. The common task of smart devices is to send data back to the central application.

The smart device used for this research work is RuuviTag sensor. RuuviTag is a Bluetooth beacon that can send small sized messages continuously. For that, it can be assumed that continuous message sending may affect its battery life but in real it does not affect battery life as it is a Bluetooth Low Energy (BLE) device which requires very small amount of power consumption. These small sized messages will be sent from RuuviTag sensor to a dedicated mobile app on which the sensor will be connected. Mobile app will control the connection of RuuviTag sensor, and more than one sensor will be possible to connect. Messages from different sensors will be handled separately. Another scenario of this research will be, Ruuvi Gateway will be connect to the mobile app and will send the data to the mobile app after acquiring it from RuuviTag sensor.

The connection between RuuviTag sensor or Ruuvi Gateway and mobile app will be established with Bluetooth. Then data will be transmitted across the IoT environment through IoT communication protocol from application layer. Few known protocols such as MQTT, CoAP and AMQP will be applied. Based on some standard criteria these protocols will be judged and most efficient protocol can be found for this research. Choosing criteria stand is also important as each protocol have their own advantage and disadvantage. The difference between criteria may result in varying efficiencies among protocols. Some basic criteria that can be considered for this research are QoS, performance, interoperability, security, fault tolerance and so on [7]. For handling large number of users with real time data the best protocol also can be judged by application scalability, flexibility, protocol implementation complexity, support of QoS level, protocol suitability for these specific scenarios and so on.

In this research, the mobile application will be able to handle multiple connections with RuuviTag sensors. From that point of view, mobile application and RuuviTag sensors will go through simultaneous connectivity. Scalability from both end of connectivity is one of the critical concerns here. If one of the ends lose scalability, communication protocols will not produce expected results as it should be. An existing mobile application with previous feature will be used for this research. After integrating the sensors, mobile application should be same flexible as it was before. High quality and robustness are essential during data exchange between sensors and application. Secure transmission will increase the reliability of the communication protocol.

As a Software Designer at a company, I have started working on this research. The company already possesses its own IoT solution, which includes a React Native mobile application. This application enables users to view telemetry data, encompassing information such as location, battery status, magnetometer, gyroscope, and accelerometer readings. The user can select specific information, which can then be sent to the cloud in a structured format, determined by the user-configurable sampling rate. Furthermore, the mobile application can generate reports based on the telemetry data for the user.

The remainder of the thesis is structured as follows. Chapter 2 provides an overview of the Internet of Things (IoT), followed by a brief discussion of the IoT architecture, which encompasses different layers. Among the various application domains within the realm of IoT, one specific application domain Smart Homes, is explored later in this chapter. Chapter 3 delves into three protocols, namely MQTT, CoAP, and AMQP, discussing their quality of service and various attributes. Chapter 4 serves as a methodological discussion, focusing on the RuuviTag sensor and React Native. The chapter outlines how data will be transferred between the RuuviTag sensor and the mobile application, followed by a discussion on the quality attributes that the mobile application should maintain. Chapter 5 presents the implementation details of all three protocols (MQTT, CoAP, and AMQP), highlighting the processing of RuuviTag sensor data throughout the implementation. The protocols are compared based on complexity, accuracy, scalability, and security. This chapter also addresses the limitations of this research. Finally, Chapter 6 offers a comprehensive conclusion of the research.

2. INTERNET OF THINGS

Internet of Things (IoT) has emerged in recent years as one of the most prominent 21st-century technologies. In this chapter a brief overview of IoT, IoT Architecture with different layers, IoT Application Domains and Smart Homes will be discussed.

2.1 Overview

The very first introduction of IoT came from a member of Radio Frequency Identification (RFID) development community in 1999 [8]. From the name it can be easily assumed that IoT is a combination of two words named “Internet” and “Things”. One of the most accurate ways to define IoT would be:

“An open and comprehensive network of intelligent objects that have the capacity to auto-organize, share information, data and resources, reacting and acting in face of situations and changes in the environment” [9]

In more concrete fashion, IoT is the network which connect computing devices, digital or mechanical machines, objects those are given unique identifiers (UIDs) and IoT has the capacity to transfer data without the necessity of human-to-human or human-to-computer interaction. The IoT notion is fairly broad and has many distinct types of meanings and it is associated with "Ubiquitous Computing".

Efficient real time data tracking making IoT more popular among large industries. Real-time data analytics helping these industries to detect malfunctions easily. More data is playing vital role to make more accurate decisions and stable connection. Better decisions for the products will improve customer experience.

From the Figure 2, a physical device or computer program known as an IoT gateway that connects the cloud to controllers, sensors, and other intelligent devices. A straightforward IoT gateway performs similar responsibilities to a Wi-Fi router. The gateway in a IoT system receives a Wi-Fi connection, and then routes the data from the IoT device to desired location such as user interface, backend system or cloud for data analysis, storage or to process. Devices with network connections and data transmission capabilities are called IoT enabled devices or smart devices. The ecosystem of IoT is made up of web-enabled smart devices that use embedded systems, such as processors, sensors, and communication hardware, to accumulate, send, and react on the data they get through their surroundings. Any IoT enabled device can share the data to an IoT gateway that they are connected with or any other edge devices such as IoT server or IoT router which will send the data to the cloud or analyze locally. These devices communicate with other devices such as smart switches, smart devices and so on, these are related to the process as an act on the data they received. Most of the work in this process is done by the devices without any human involvement. Although, human can only interact by setting them up, giving instruction or access data.

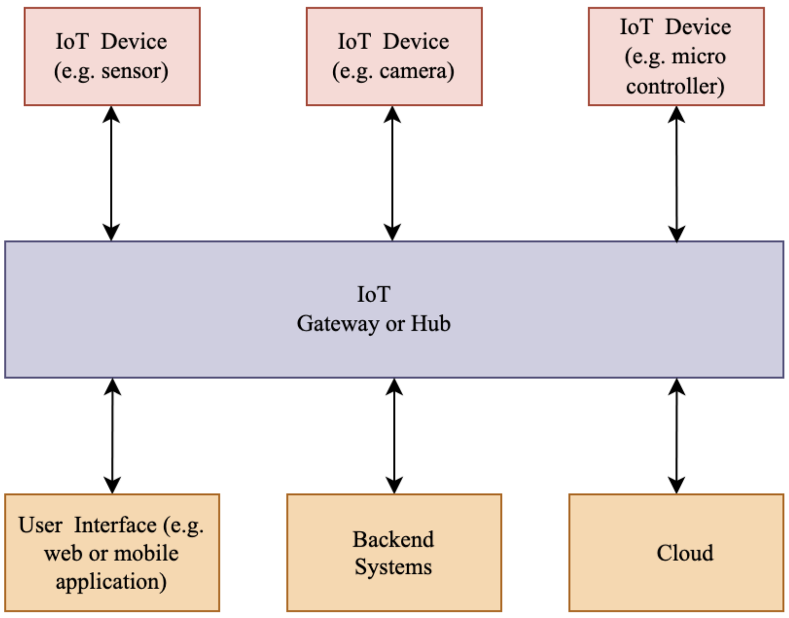


Figure 2. *Sample IoT System [10] [11].*

IoT is helping people to work and live more intelligently and slowly IoT is gaining maximum control over their daily life. IoT allowing people to automate almost everything, and it lightens the workload. A useful example can be smart watches. This device can automatically track user behaviours and make range of recommendations to the user based on user’s activity and location.

2.2 IoT Architecture

Architecture as a context can be simply defined as a structure of a system. This structure encompasses all elements, interactions between them, the environment in which they function, and the principles that used in designing. The most basic IoT architecture consists of 3-layer model. The layers in basic IoT architecture are Perception, Network and Application layer. In 3-layer architecture model, the perception layer has sensors that used to sense information from the surroundings. The network layer is responsible for connecting to other smart and network devices, and server. Its capabilities are also employed for processing and transferring sensor data. Delivering application-specific services to the user is the responsibility of the application layer.

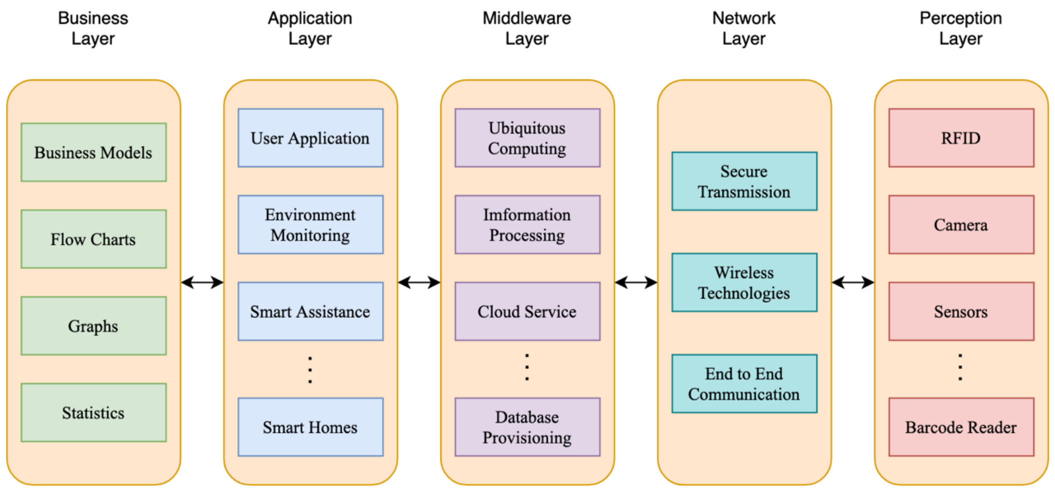


Figure 3. *IoT Architecture.*

When IoT project entails a variety of cutting-edge technology with a wide range of applications, 5-layer architecture assumed to be the ideal fit. The 5-layer model is just an extension to the basic model with two additional layers. As illustrated in the Figure 3, these two additional layers are Business layer and Middleware layer [12] [13] [14]. The business layer is in charge of supervising the IoT system's overall business logic and processes. The middleware layer within the IoT architecture serves as a crucial link connecting the application and network levels.

There are also other IoT architectural model such as 4-layer and 6-layer model. Along with basic layers an extra layer named Support layer comes on 4-layer architecture model [15]. For data processing or storage, the support layer is typically used. Based on project structure Support layer can be replaced by Data Processing layer. In 6-layer architecture, Observer or Monitoring layer comes with all other 5 layers from 5-layer architecture. Data authentications are done in the observer layer. All kind of software agents resides in this layer, and they are responsible for decision making and reasoning [16]. Following is a quick overview of each layer based on a 5-layer architecture.

2.2.1 Business Layer

The business layer illustrates the business model that helps an organization achieve its corporate goals. It engages with stakeholders to support the business decision-making process and assist them to gain insights from the information generated on the application layer and support the business decision making process. It entails insights such as creating graphs, flowcharts, outcomes analysis, and how the device can be updated, among other things. The business layer is the manager of IoT architecture because it manages the application, services, build business models and many more. Any device's success depends not just on the technologies it uses, but also on how it is distributed to its users. For the device, the business layer does these duties. IoT business layer allows research not only for business models but also or profit models. As business model plays a vital role, based on IoT services and activities business models are selected and it serves specific IoT maturity and adoption. Business model can create many uncertainties such as true economic uncertainty, competition uncertainty, true uncertainty, predictable future uncertainty, range of future uncertainties and many more. These uncertainties are considered as IoT challenges [17]. Uncertainties are always important for any business. For example, during COVID-19 every organization in the world had challenging times. Uncertainties helps a business to deal with this kind of situation. Sometimes there is no way to avoid a situation from being completely uncertain, but if managers are aware of a product's uncertainties, it will assist to mitigate the crisis somewhat and maintain consumers trust in the product. Along with business models with the help on insights business layer also provides the accurate usage and sales report. From these reports manager can manage the product more efficiently. This layer also takes good care of the user privacy which is very important in IoT. Finally, IoT business layer measure the IoT application with other relevant application that are in demand in the market currently.

2.2.2 Application Layer

Application layer mainly develop application based on the data that are processed from middleware layer and this application attain the need of the business. Application layer serves an interface between application users and IoT. This layer is mostly responsible for customer service, high-quality smart assistance, satisfy all kind of customer requirements. There are range of protocols in application layer which can establish the communication between the internet, IoT gateways and user application. These protocols carry command from the user application to the edge devices and acknowledge servers about the latest update of the device. Some most common application layer protocols are Constrained Application Protocol (CoAP), Message Queue Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP) and so on [18]. There is no standard protocol for every IoT application. Factors like data latency, reliability, scalability, bandwidth, transport helps to find the right protocol for a specific IoT application. Any user application manages to accomplish data storage, mining and intelligence in decision making with the help of application layer. Application layers helps IoT to push for a larger scale of development in the field of technology.

2.2.3 Middleware Layer

The first and foremost task of middleware layer is hiding the hardware details from the developers and encourage them to provide more concentration on the development process of the application. Middleware layer comes up with many advanced features such as ubiquitous computing, service management, information processing, data storage and many more. Other than the features this layer make sure scalability, abstraction, interoperability and provide service for the customers [19]. After receiving data from the hardware, it can make decision based on the data and can perform filtering and aggregation on the received data. If needed it can send the required actions over the network according to the commands that came through those data. The service and the requester of that service are bind together in this layer considering their name and address. Thus, this enables the developers to work without thinking about a specific platform and heterogeneous objects. Vendor provided web and mobile app can work here as well as the cloud service is integrated with this layer. The raw data from IoT devices are extracted here as a part of information processing and converted into human-readable or machine-readable format. This method distinguishes useful abstractions from the raw data.

2.2.4 Network Layer

The Internet of Things is a vast network that connects billions of different networks in addition to billions of physical objects. Communication between various networks and entities is therefore crucially important. The ultimate goal in IoT is to transmit information across the network. Therefore, network layer considered to be the heart of the IoT system. This layer deals with risks such as unauthorized access, confidentiality, any middle man attacks, precision of the data, service attacks and so on. Basic communication framework can be implemented by the network layer. Network layer also called transport layer because of its responsibility of establishing communication between middleware layer and perception layer by using 3G, 4G, Bluetooth, Wi-Fi, NFC and so on [20]. For routing, a variety of protocols can be used in this layer and these protocols falls into two categories: standard and non-standard. Routing Protocol for Low-Power and Lossy Networks (RPL), Collection Tree Protocol (CTP) and many more are standard protocols. Common Address Redundancy Protocol (CARP) is one of the non-standard protocols that are used in network layer for routing [21]. There are two sublayers called routing layer and encapsulation layer. The routing layer manages the packet transit between the source and destination. The packets are formed by the encapsulation layer [22].

2.2.5 Perception Layer

The primary responsibility of the perception layer is to detect physical characteristics of objects such as temperature, humidity, air pressure, location and so on by using a variety of sensors like RFID, wireless sensors, camera sensor, barcodes and many more. Then in this layer these characteristics are converted into digital signals that are quick and easy to transmit over a network. Because of this, the perception layer is also known as a physical layer [23]. For designing microchips in wireless communications perception layer plays a vital role.

Wireless sensors are one of most popular kind of way to transmit information in IoT. Wireless sensors consume very little power and therefore, wireless sensors never perform heavy data processing locally. As sensors transmit very light weight data, they can be easily maintained on slow networks. Wireless sensors can be differentiated into two types. Passive sensors that are self-powered devices and collect data from the environment. These sensors do not constantly observe for data thus they do not require any external source of energy. Another type is called Active sensors. They depend on external power as these sensors continuously monitor the environment.

The perception layer also comes up with few challenges. These challenges are mostly related to the IoT security. The most common IoT attacks happen in this layer such as jamming, tempering, collision, permanently disabling the tags and many more [24]. Therefore, designing an IoT product perception layer needs some extra bit of attention as security is the first priority to the user themselves.

2.3 IoT Application Domains

IoT aims to enhance most common objects in daily life with computing power and an internet connection to give them the ability to perceive, calculate, communicate, and control their environment. With this goal IoT application brings immense benefit to our lives. IoT is not only making life easy for individuals but also helping business to grow faster. Because of IoT customers can have more transparency and flexibility in a specific product and business owners can have real time data access which helps them to profit more. Considering these aspects consumer of IoT can be categorized into [25]:

- Individuals: those who want to raise their standard of living generally.
- Society: a community of people seeking solution to problems that affect everyone in the same way.
- Industry: an economic or industrial sector seeking to meet the needs and requirements of customers.

Applications those satisfy at least one of the consumer sections comes into IoT application domain. Few simple examples can be given to understand the IoT application domains a lot better. A traffic system that can sense the cars and pedestrian through some sensors and functions accordingly. A parking system that detects a specific parking is available or not and finally produces a result of how many parking spots are available. Along with these two systems many more other systems that is related to IoT and makes everyday city life more flexible, falls under smart city and it is an IoT application domain. Systems like smart television, temperature measurement device, smart switch and any other devices that makes a home more automated comes under smart homes. Smart homes are one of the popular IoT application domain.

From the Figure 4, here is some popular IoT application domains. There is no specific count of IoT application domains. Counts of application domains are increasing every day. Thus, we want our soundings to be automated then there should not be any boundaries for IoT application domain. Smart environment is something where users can engage and interact with their ambient environment without effort. In the home setting, smart environments are gaining a lot of attraction. By redefining the realm of devices and human interaction in healthcare solutions, IoT is unquestionably changing the healthcare sector. IoT has uses in healthcare that are beneficial to patients, doctors, hospitals, and financial institutions. Cities need to become smarter if they are to address climate change and make cities more liveable globally. A municipality is referred to as a "smart city" if it employs IoT to boost administrative effectiveness by using range of IoT devices and these devices collect and analyze data for betterment of human life in that city.

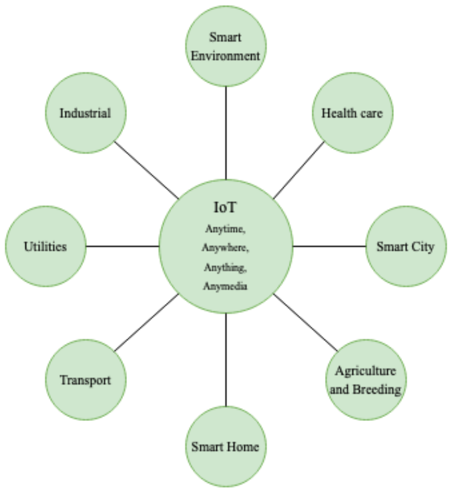


Figure 4. *IoT Application Domain [26].*

IoT smart agricultural solutions are made to assist in crop field monitoring using sensors and irrigation system automation. As a matter of fact, farmers and related brands may effortlessly and conveniently check on the state of the fields from any location. IoT in transportation is already a large industry. IoT devices are used in a variety of applications within the transportation industry, including tools and ticketing, informatics systems in vehicles, traffic congestion systems, and security and surveillance, to mention a few. The utilities industry has benefited greatly from the introduction of the Internet of Things. IoT helps to manage utilities, provide alternatives for remote control, improve safety, save prices, and address the issue of resource depletion. IoT in industry has a name Industrial Internet of Things (IIoT) and it is growing vastly than any other IoT domains. The deployment of smart sensors and actuators to improve manufacturing and industrial processes is known as the IIoT.

2.4 Smart Homes

Smart home is a simple configuration that allows apps or equipment to be autonomously controlled remotely or by sensors from anywhere using a network device that is connected to the internet. It is an intelligent environment that facilitates a range of activities at home. These activities include controlling home electronic appliances, housework, entertainment appliances and many more. Concepts like home intelligence, home automation, ambient intelligence, and ubiquitous computing were well-known in the early days of smart homes [27] [28]. Three essential elements are needed to call a home “Smart Home” and they are wireless, cable, wire; intelligent controlling- a gateway; home automation- accessible goods that can be controlled from both home and outside of home over some services and systems. The key achievements of smart home are automation, security, and sustainability. Smart homes have made tremendous advancements in automation, which allows homeowners to control many aspects of their homes remotely and via voice commands. Smart home offers advance security features that can detect possible security breaches and respond to emergencies quickly and effectively. Through sustainability smart homes achieved the ability to reduce their footprint and offer various way to conserve water, energy and other resources. These achievements emerge the popularity, reliability, and demand of smart home in the field of technology.

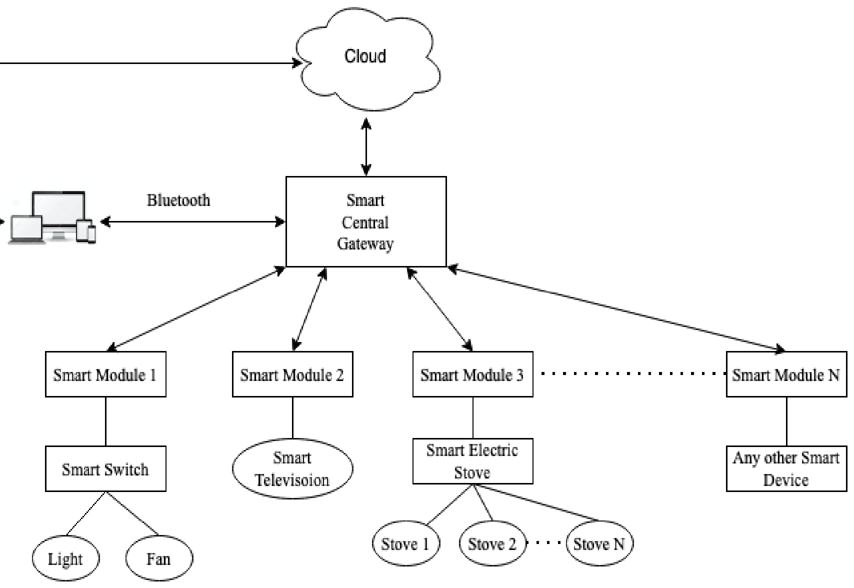


Figure 5. *Concept of a Smart Home using IoT [29].*

From the Figure 5, it can be assumed that in smart home a central gateway is the principal component of the system. This central gateway is connected to the internet and able to receive a set of instructions from user application. The central gateway sends those instruction accordingly to the smart module. Different smart module carries out different operations. A specific smart module is dedicated for specific task. The central gateway accepts any number of smart modules that is acceptable by its firmware. Central gateway is able to accept instructions over Bluetooth other than internet if the user is at home. Any changes that the central gateway receives from those smart modules, it has the capability to send those changes to the cloud over internet or via Bluetooth if the connection is established between user application and the gateway.

The diversity of technologies in a smart home is impressive. Since smart home is a complex network, the development of this network has been always challenging. The first issue that appears over all other challenges is choosing the appropriate wireless technology [30]. The primary wireless communication protocols that can be used to connect home include Bluetooth, Wi-Fi, Zigbee and so on. Based on the kind of data to be transmitted wireless communication protocol gets selected. Among all other protocols Zigbee, Bluetooth and Wi-Fi are on the top choice [31]. BLE connects through user device internet to the cloud using the interface of the user application while Wi-Fi enables the devices to connect to the internet through a router that resides at home. At smart home, the interoperability among smart devices must need to ensure for smoother data transmission between them. Any smart device that has limited technical coverage may lead to a bad user experience. Therefore, the smart devices should have the capability to keep the pace with rapidly changing technology.

3. IOT APPLICATION LAYER PROTOCOLS

In IoT, the numerous IoT communication protocols are the means of communication that can create the highest level of security for the data being transferred between linked IoT devices [32]. Any communication protocol is essential because protocol ensures data integrity, prevent deadlock, manage the flow control, manage congestion, check platform error and so on.

The IoT technology stack includes IoT protocols as an essential component. Hardware would be labelled useless without IoT protocols and standards. Most of the IoT devices are low power devices and protocols are being used to connect and transfer data between them. These protocols support point-to-point networking for the physical hardware that are at the user end. For real-time data collection, analysis and high-level IoT communication in a large-scale network, protocols should have the necessary qualities, such as low packet response time, high packet generation time and minimal packet loss [33].

All IoT devices needs to communicate with each other. Some IoT devices only send data whereas some devices both send and receive data. Thus, communication between IoT devices can be both unidirectional which only sends data to another device or bidirectional which do both of the tasks to send and receive data. For a remote communication a gateway is needed to transfer data. Keeping this communication fashion in mind there are many IoT application layer protocols are available. Some of them are Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), Advanced Message Queuing Protocol (AMQP), Data Distribution Service (DDS), Extensible Messaging and Presence Protocol (XMPP) and so on. Choosing right protocol is always essential for smoother user experience. Every protocol has its own advantages and disadvantages based on the task they have to perform. For example, if data transfer requires Quality of Service (QoS) then XMPP is not the right one to choose. Again, if the transmission of data should be based on User Datagram Protocol (UDP) then CoAP is the perfect choice for the task.

All protocols are focused to transmit light weight data. IoT devices operate in less power and memory thus without any internet connection protocols help those data to reach desired destination. Whenever sending data to cloud is a requirement, a gateway or server is needed protocols are not dedicated for this task. Data are sent into small packets though the protocols. Header size of those packets vary from protocol to protocol. Usually, header size can be 2 Bytes. Different protocols have different encryption and decryption method to keep the end-to-end data transmission secure and safe. Data transmission speed is very important, and it mostly depends on how much lightweight data the protocol is responsible to send.

3.1 MQTT

Message Queuing Telemetry Transport (MQTT) is a lightweight messaging protocol which follows client-broker architecture with communication model of publish subscribe. For Machine to Machine (M2M) communication on IoT, MQTT is the best suited protocol because of lower bandwidth, small code footprint and lower power consumption. Any mobile application battery power and bandwidth has always been a concern. Thus, MQTT became famous protocol in mobile application development [34].

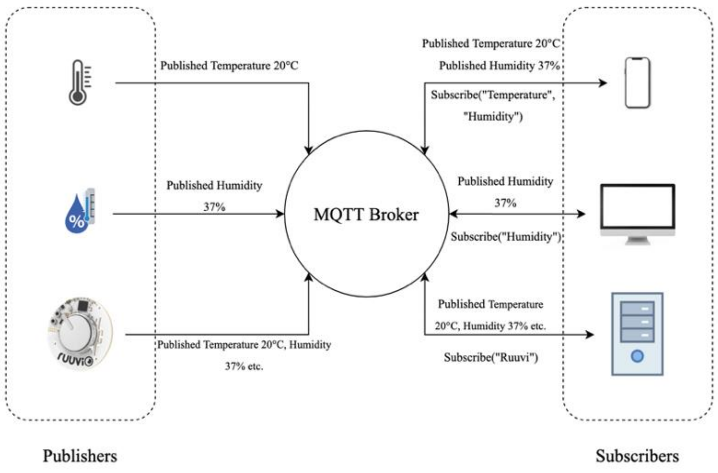

Published Temperature 20°C
Published Humidity 37%
Published Temperature 20°C, Humidity 37% etc.
MQTT Broker
Published Temperature 20°C
Published Humidity 37%
Subscribe("Temperature", "Humidity")
Published Humidity 37%
Subscribe("Humidity")
Published Temperature 20°C, Humidity 37% etc.
Subscribe("Ruuvi")
ruuvi
Publishers
Subscribers

Figure 6. *Basic MQTT Architecture.*

In the Figure 6, a basic MQTT workflow has been illustrated. MQTT broker is the engine of this workflow. Sensors those acts as publishers, published their respective data to the broker. A topic to a broker can be subscribed to by a subscriber. One subscriber is allowed to subscribe more than one topic at the same time. When the sensor senses some data from the surroundings and publish it to the broker. Then broker checks which devices are the subscribers for those data and publish those data only to those devices. For example, a temperature sensor published a data of 20°C to the broker. A mobile application and desktop application subscribe to the topic named “Temperature” but a server did not subscribe to that topic. The broker will only send the data to the mobile and desktop application and server will not even have any acknowledgment that some data has been published.

Every protocol has its own key highlights that helps them to define their work structure and efficiency. The publish subscribe architecture helps MQTT to gain its popularity in such a short time. MQTT is lightweight and the minimum header size that it accepts is 2 Bytes. MQTT mainly operates over TCP/IP protocol. This protocol is payload agnostic which means the content that is being served by the service is not aware to it. To lessen network traffic, a relatively small transport overhead and minimal protocol exchanges are used. For security, MQTT uses SSL/TLS and because of this an encryption is used to protect the transmission of the data. MQTT has the mechanism that alerts interested parties when an unexpected disconnection occurred.

3.2 CoAP

Constrained Application Protocol (CoAP) is a document transfer protocol with client-server architecture. In CoAP, a set of “resources” are published by CoAP server and the “Representational State Transfer (REST)” methodology is used by clients. To keep the overall structure lightweight, CoAP employs UDP (no TCP overhead) and uses HTTP commands such as GET, POST, PUT, and DELETE to provide resource-oriented communication. Clients can also "subscribe" to a resource to get ongoing updates whenever it is updated.

From Figure 7, In IoT when CoAP is in operation, it creates its own environment. When sensors and actuators are transferring surrounding data, they are connected to a CoAP server. Sensors and actuators act as clients in CoAP environment. All clients in CoAP environment can only communicate with CoAP server. When a client is able to receive data from CoAP server, it can subscribe to some specific resource to get notify if there is any new update or not. On the other side CoAP server is connected to a gateway which is REST-CoAP proxy. The gateway can establish a connection to the internet which can be cloud. The communication between the gateway and cloud can be HTTP

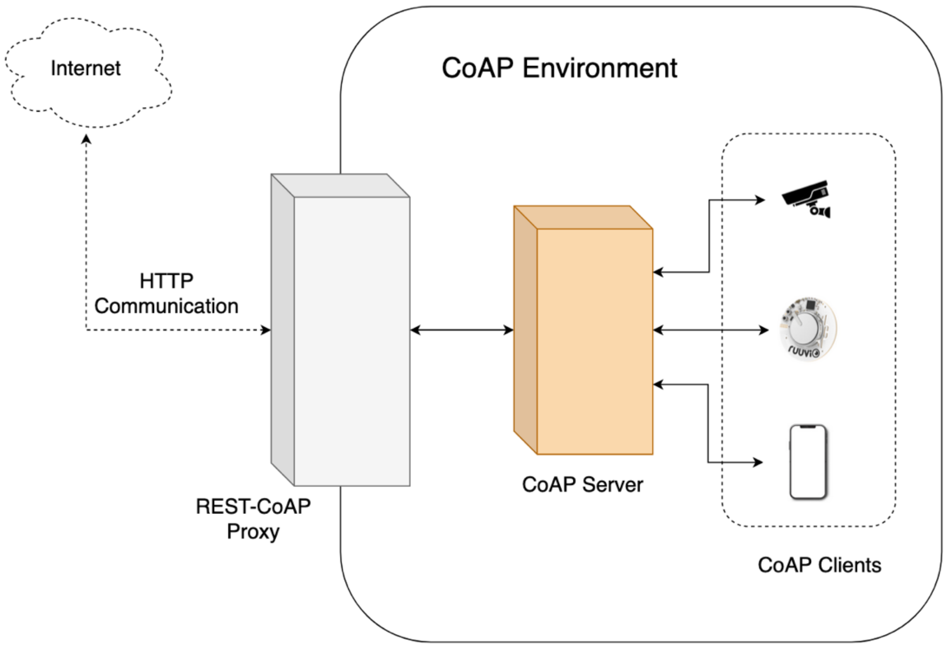


Figure 7. *CoAP Standard Implementation [35].*

CoAP operation is based on UDP/IP. Therefore, CoAP does not require any prior communication for setting up data paths or communication channels. This protocol is usually Unicast, but it can be Multicast as well. CoAP has reliability in security and for security it uses Datagram Transport Layer Security (DTLS). DTLS helps CoAP with data tempering, message forgery or eavesdropping. It is a decentralised protocol so there cannot be a single point of failure. CoAP nodes can be easily turned into servers, and it supports resource discovery. CoAP is widely used in smart homes and smart energy grids.

3.3 AMQP

Advanced Message Queueing Protocol (AMQP) is an open standard protocol. AMQP was developed for the systems for messaging interoperability between them. AMQP mostly used for business message transmission across applications. At small bandwidths, AMQP has a poor success rate, but as the bandwidth grows, the success rate rises. However, as compared to REST, AMQP may transmit more messages per second.

From Figure 8, when it comes to functionality, an AMQP broker consists of three components such as Exchange, Queue and Binding. These three components can be implemented to queueing and routing. Exchange receives messages from publishers. Within the message constructor will always be included to specify the message type, for example, URLs in UTF-8. Binding connects proper queue with exchange and then message is being sent to the appropriate queue for storage. Queue finds the subscriber for the message and forwards it to them. AMQP has two layers at the architectural level. Channel multiplexing, data representation, heartbeat, framing, error handling and content encoding are all handled by the transport layer, which is the lower layer. The function layer, which offers messaging capabilities, is placed above it.

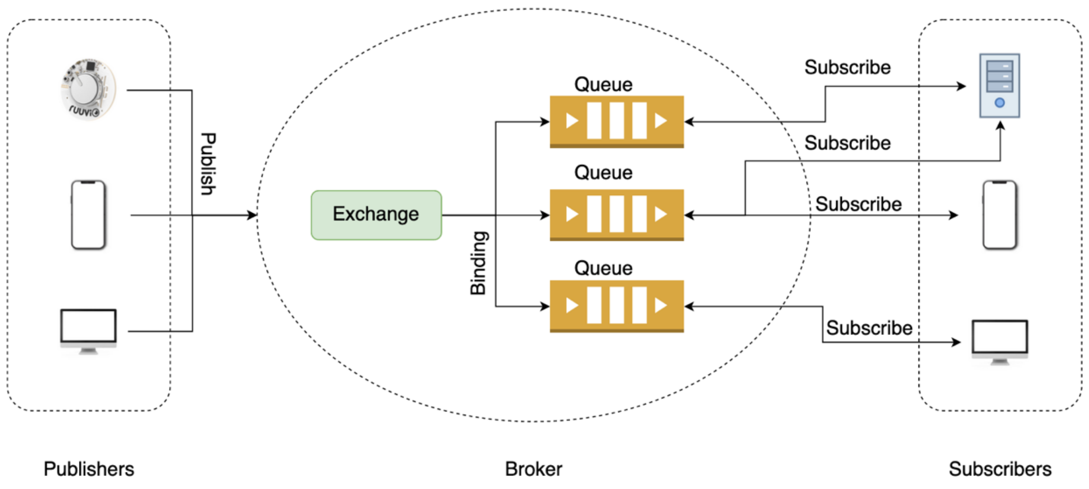


Figure 8. *AMQP Architecture [36].*

AMQP exchange messages from one point to another which is also called server to server exchange. AMQP ensures flexible message exchange thus the messaging pattern is relaxed in it. Through the use of broker services and TCP/IP connections, AMQP aims to facilitate message passing. Being a binary protocol as it sends binary data only AMQP is regarded as a compact protocol. The AMQP channels have a feature called QOS. The way AMQP consumers read messages from queues depends on the value of this QoS property. AMQP uses the same security as MQTT which is SSL/TLS.

3.4 Quality of Service

The ability of a network to offer a more suited or proper service to the application is referred to as Quality of Service (QoS). IoT implementation success relies on the volume of data sent or received over networks, the maintenance of QoS, and the strategies used to overcome the energy limitations of battery-operated devices. The services can be designed to provide effective support by taking into account a variety of QoS factors. QoS factors can influence IoT system reliability, scalability, security, latency, bandwidth, availability and so on. IoT QoS factors are essential for guaranteeing the dependability, effectiveness, and security of IoT systems. End-to-end latency, packet delivery ratio, throughput, and jitter are the QoS factors for networks [37].

In IoT, Quality of Service (QoS) is crucial because it makes sure that devices can interact with one another effectively and efficiently. The context of smart cities can be used to understand the significance of QoS in IoT. IoT devices are used by smart cities to collect information concerning everything from traffic patterns to air quality. Decisions about everything from urban planning to emergency response are made using these details. The choices made based on the data from these devices, however, could be incorrect or ineffective if the data is delayed or lost due to network congestion.

The reliability of message delivery between a publisher and subscriber is defined by the three tiers of Quality of Service (QoS) offered by the MQTT (Message Queuing Telemetry Transport) protocol. These tiers are:

1. QoS 0 (At most once): In this level of QoS, the publisher sends the message only once, and the broker does not acknowledge it. There is no assurance that the message will arrive because it could be misdirected or duplicated in the network. For non-critical applications, where the occasional message loss is acceptable and there is no requirement to resend the message, this QoS level is appropriate.
2. QoS 1 (At least once): The communication is acknowledged by the broker after it is received from the publisher at this level of QoS. The broker asks the publisher to transmit the message again if it is not received by it. If the publisher does not receive the acknowledgement, this level of QoS could lead to duplicate messages even though it ensures that the message will be delivered at least once. Applications that occasionally permit duplicate messages, but message loss is not acceptable can use this level of QoS.
3. QoS 2 (Exactly once): In this level of QoS, the broker acknowledges the message after receiving it from the publisher, and the broker acknowledges the publisher after the message has been successfully delivered. The message will be delivered precisely once, without loss or duplication, under this level of QoS. But it

necessitates more network overhead and could cause greater latency. Applications where message loss or duplication is unacceptable, and the message must be delivered precisely once should use this level of QoS.

Confirmable (CON) and Non-confirmable (NON) are two QoS categories supported by the Constrained Application Protocol (CoAP). By necessitating an acknowledgement for each message, the Confirmable (CON) message delivery in CoAP offers a reliability guarantee. This means that before the sender transmits the next message, the recipient must confirm that they received the message. This guarantees the message's delivery and offers a trustworthy method for resend in the event of packet loss. Non-confirmable (NON) message delivery, on the other hand, offers a lower degree of reliability since no acknowledgement is necessary. This makes it appropriate for use in situations where the message's delivery is not essential, like when frequent contact is necessary or when there are constraints on the network resources. The particular requirements of the application and the network conditions determine which QoS level should be used.

Support for Quality of Service is one of AMQP's main features. Settle format without acknowledgments (QoS level 0) and Unsettle format with acknowledgments (QoS level 1) are the two levels of QoS that AMQP offers. Messages are sent at QoS level 0 without awaiting a response from the recipient. This method of transmission is appropriate for uses where data loss is tolerable, such as with easily replaceable sensor data. Prior to sending another message at QoS level 1, the recipient must acknowledge it. The sender will resend the message if there is no acknowledgment. This method of delivery is appropriate for applications where a small amount of data loss is allowed but not at the expense of duplicate data.

3.5 Attributes for Protocol

IoT application layers protocols have some similarities and dissimilarities among them. To find a better protocol there needs to have some specific requirements. Based on those requirements better protocol can be found for a specific application. Requirements are not independent to the application to application. According to the application workflow, design and other facts create a list of attributes for protocol [38].

Basis of	MQTT	CoAP	AMQP
Architecture	Client-Broker	Client-Server	Both Client-Broker and Client-Server
Messaging Mode	Only Asynchronous	Both Asynchronous and Synchronous	Both Asynchronous and Synchronous
Transport Layer Protocol	Transmission Control Protocol	User Datagram Protocol	Transmission Control Protocol
Communication Model	Many to Many	One to One	Mainly One to Many
Resource Discovery	Low	Higher than MQTT	High
Transmission Speed	High	High	Slower than other protocols
Connection Status	Available	Unavailable	Unavailable
Complexity	Less complex	Less complex	Relatively complex
Broadcasting Messages	Possible	Not Possible	Possible
Security	Can secure through client identifier, client certificate along with username and password.	CoAP is highly dependent on UDP's security features.	Channel Authentication Rules (CHLAUTH), Connection Authentication (CONNAUTH), Java Authentication and Authorization Service (JAAS) are available.

***Table 1**: Difference between IoT communication protocols.*

4. METHODOLOGY

In this chapter process of data collection and analysis, different methods of this research, device that has been used and other tools that are needed to carry out the research will be discussed. RuuviTag sensor will sense surroundings and produce data accordingly. Data will be transferred to the ultimate destination that is mobile application developed on React Native platform through BLE. From mobile application data will be shared among client though the protocol.

4.1 RuuviTag Sensor

RuuviTag is a wireless Bluetooth sensor. Through this sensor node it can measure temperature, air humidity, air pressure and movement. All the data this sensor can measure are live data. RuuviTag is an open source BLE sensor and using the ESP32 Bluetooth Low Energy Tracker Hub, the Ruuvitag sensor platform monitors the output of RuuviTag Bluetooth Low Energy devices. Every time the sensor sends out a BLE broadcast, this component will track the temperature, humidity, acceleration, and battery voltage of the RuuviTag device using the RAWv1 protocol. The RAWv2 protocol is also supported. Further tracking includes tx power (indicates device’s worst case transmit power), movement count, and measurement sequence number.

There is a gateway that is called Ruuvi gateway that allows to monitor sensors remotely. Through a Wi-Fi or Ethernet connection, Ruuvi gateway connects to user’s personal Internet connection. This gateway can establish connection with RuuviTag sensors with bluetooth in such a short range. Ruuvi gateway can send the data to the cloud and from the cloud user can see the real time updates from the sensors. Being an open-source platform, it is not difficult to configure the gateway to any cloud service. In an advanced feature, Ruuvi Gateway can be connected to the internet via a covert Wi-Fi network.

4.2 React Native

React Native (RN) is a framework or tool for cross-platform mobile development that has been introduced by Facebook and released in 2015. JavaScript and ECMAScript 2015 are used to build and develop React Native applications (ES2015 or ES6). React Native supports both iOS and Android operating system. As an open-source platform React Native has a large and strong community around the world.

The base code for the iOS and Android platforms on the native side is written in Objective-C for iOS and Java for Android within the React Native framework. As a developer, no need to write native codes while developing the mobile application. React.js, more commonly referred to as React, is the foundation upon which React Native is constructed. Architecturally React Native have three parts such as Native Modules, React Native Bridge and JavaScript virtual machine. For running the code React Native uses JavaScriptCore to run the code and it is a JavaScript engine for WebKit which is open-source platform [39].

Due to flexible environment cross platform apps are getting attention all around the world. From both development and business perspective React Native has lots of advantages but from business perspective it has greater impact. From development perspective React Native is easy to work with. It is platform independent which means it allows to build app for iOS, Android and Windows with a single code base [40]. Being open source, a large developer community helping each other. Any plugins from third party are compatible with React Native. From business point of view React Native is cost effective and the speed of the development is faster. Maintenance cost is not high as it has only single code base to deal with. No change of user experience which is also a positive point from business end. For complex or gaming apps, React Native is not a good choice. React Native is still below of native apps in terms of performance. Keep pace with the latest React Native versions are difficult. Updating process of React Native version is a complex process.

4.3 Data Transfer

IoT-Ticket is a IoT tool suite and platform which allows to create web, mobile, cloud, and reporting applications using big data analytics and simple tools within a minute. This IoT-Ticket platform has a mobile app named IoT-Tracker app that tracks mobile phone's sensor data such as location, accelerometer, gyroscope, magnetometer, battery and so on. In this IoT-Tracker app RuuviTag sensor will be integrated. This mobile app is developed with React Native therefore it is available in both iOS and Android platform.

In this study, a communication between a mobile application and RuuviTag sensor need to be established to transmit data. For this transmission, a protocol is needed that should have some certain standard. The protocol should transmit data with minimum of latency, protocol should not affect the flexibility and scalability of the application, secure transmission is always expected from the protocol, very simple workflow and so on.

Data can be transferred into two scenarios. First one can be, IoT-Tracker app will search for nearby RuuviTag sensors through Bluetooth and establish a connection between them. Whenever user wants to get data from RuuviTag, only then the app will start receiving data. Then the app will send the data to the dedicated cloud with the help of application protocol. Other mobile app with the same user credentials will get the data as well even if the user of the is accessing remotely. Second scenario can be IoT-Tracker app will search for RuuviTag sensors and Ruuvi Gateway nearby and connect those through Bluetooth. IoT-Tracker app will depend on the gateway. RuuviTag sensor will send the data to the Ruuvi Gateway. The gateway will send the data to the mobile app through IoT application protocols. Also, gateway will send the data to the dedicated cloud.

4.4 Quality Attributes

The ability of an application system to cost-effectively provide increased throughput, decreased response time, and/or support more users when hardware resources are added is known as scalability [41]. In this research scalability of the mobile app is so important. Data will be transferred through IoT application protocols, and any large number of users will be using the app simultaneously. If the application is being used in any industry where it needs to be available to at least 200-300 user at the same time then scalability should be the main concern. If the app is not enough scalable then the user experience will be worst, and the app will lead to a failure project. In this case, scalability depends on mostly IoT application protocols because through protocol data will be sent to all of users and all the data's will be real time data.

The more a protocol can handle and transmit the data faster to its destination, it will achieve more scalability. Thus, less latency on data transmission of a protocol increases the scalability of the app. More latency will block the user usage as it will take more time to load the view for user which will lead to less scalable app. QoS is another important factor for protocol to make an app more scalable. QoS levels also provides reliability to the app. The protocol with more QoS levels comes up with more scalability and reliability for the app [42]. For instance, MQTT has more QoS strength because of 3 level of QoS. With 3 level of QoS, MQTT provides message storing facility to avoid duplications, sending message more than once and so on. These QoS facilities are absent from CoAP and AMQP.

Application security also plays a vital role to make application scalable from the protocol perspective. For make app secure compromising scalability is not acceptable and this is also true for the opposite scenario. Security with MQTT can be possible through some advanced authentication mechanism such as client identifier, client certificate along with username and password. Cryptography can be another key factor to enhance the security in applications that use MQTT as a protocol [43]. In terms of security, CoAP is highly dependent on UDP's security features. UDP relies on Datagram Transport Layer Security (DTLS) which provides privacy in the communication. DTLS offers autonomous key management, data integrity, confidentiality, and authentication. It also supports a variety of alternative cryptographic methods, making it a possible contender for a security protocol. In order to provide the necessary security services, CoAP defines four security modes in its draft. NoSec, PreSharedKey, RawPublicKey, and Certificate are these modes [44]. A range of security mechanism can be used to secure connections from AMQP clients and these mechanisms ensures data protection over the network. Channel authentication rules (CHLAUTH) to restrict the TCP connections to a queue manager. Connection authentication (CONNAUTH) to authentication connections to a queue manager. Optionally it is possible to configure AMQP channels by Java Authentication and Authorization Services (JAAS) which will check the username and password that is provided by AMQP client.

5. RESULT AND DISCUSSION

This research covers evaluation methods and conducts an experimental analysis of the results for three protocols MQTT, CoAP and AMQP. Work around of these protocols for the specific mobile app and RuuviTag sensor will be discussed. A decision will be made among these protocols about their compatibility, accuracy, scalability and so on.

From previous discussion, there are two scenarios in this research. First one is RuuviTag sensor communicating with mobile app. In the second scenario, RuuviTag gateway comes along with the mobile app and the RuuviTag sensor. There is a general implementation before protocols come into action. RuuviTag Sensor transfer data to the mobile app and gateway trough Bluetooth. Then IoT application protocols take the responsibility to transmit data over whole IoT system.

5.1 Implementation with MQTT

To establish the MQTT client, MQTT broker is very important that will work as a central hub for all MQTT messages. The next step is to configure the MQTT client on the client device that will communicate with the MQTT broker. For authentication, a fixed username and password is mandatory for all clients that will ensure a secure transmission of data. Based on Figure 9 and Figure 10, two scenario implementations with MQTT will be described respectively.

In the first scenario, when the mobile app launched if there is no Ruuvitag sensor paired it does not take any action. On the other hand, if there is a paired one then the mobile app will try to reach out to the RuuviTag sensor through Bluetooth. If the app can reach out to the Ruuvitag sensor, then it will connect to the web broker with appropriate client id and other attributes. The mobile app will also subscribe to the specific topic so that when another client publishes data the app also gets those data as one of the subscribers. When the mobile app is getting data from RuuviTag sensor, it will start publishing data to the specific topic that is dedicated only for that sensor. If the app is paired to more than one sensor, then publishing topic will be different from each other. After publishing data, clients those are subscribe to that topic will get the data. Backend will also be a client and will also get the data.

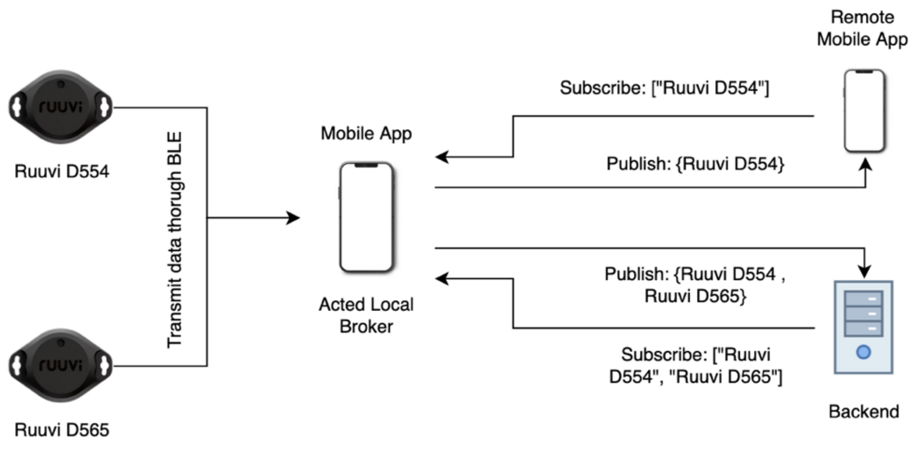

Remote Mobile App
Subscribe: ["Ruuvi D554"]
Publish: {Ruuvi D554}
Mobile App
Acted Local Broker
Ruuvi D554
Ruuvi D565
Transmit data thorugh BLE
Publish: {Ruuvi D554 , Ruuvi D565}
Subscribe: ["Ruuvi D554", "Ruuvi D565"]
Backend

Figure 9. *Scenario 1 implementation with MQTT.*

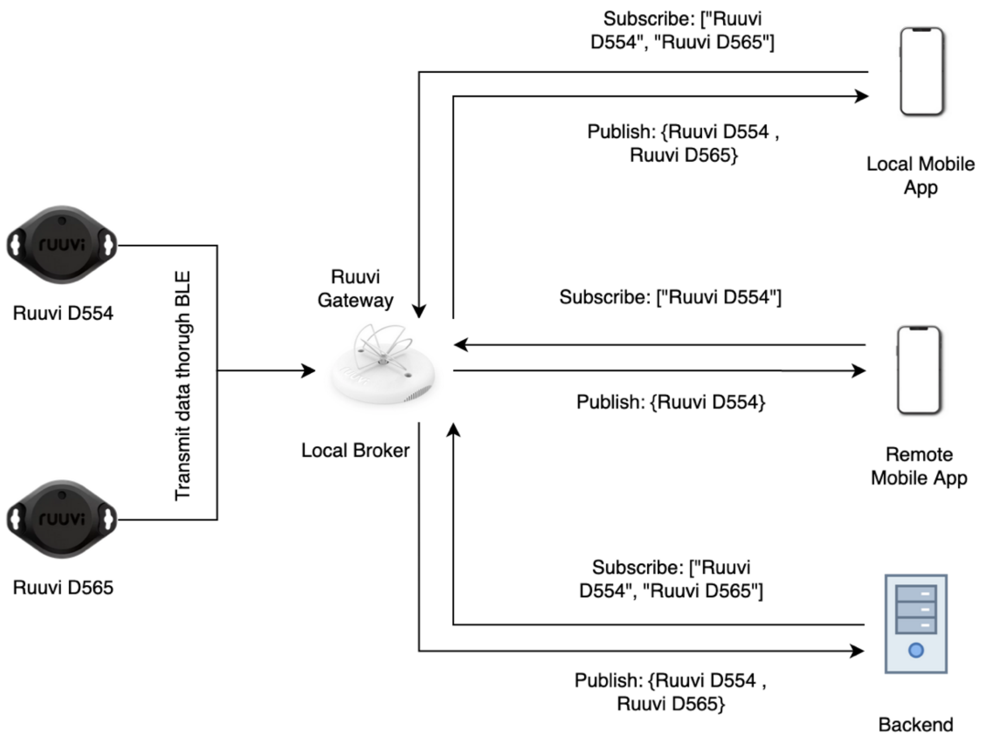

Subscribe: ["Ruuvi D554", "Ruuvi D565"]
Publish: {Ruuvi D554 , Ruuvi D565}
Local Mobile App
Ruuvi Gateway
Local Broker
Ruuvi D554
Ruuvi D565
Transmit data thorugh BLE
Subscribe: ["Ruuvi D554"]
Publish: {Ruuvi D554}
Remote Mobile App
Subscribe: ["Ruuvi D554", "Ruuvi D565"]
Publish: {Ruuvi D554 , Ruuvi D565}
Backend

Figure 10. *Scenario 2 Implementation with MQTT.*

Second scenario includes Ruuvi Gateway. With the help of Ruuvi Gateway RuuviTag sensors can be read all around the world. Ruuvi Gateway can be connected to the internet via Wi-Fi (WLAN) and send data to a dedicated backend. The mobile app will already be connected to the to RuuviTag sensor through Bluetooth. Ruuvi Gateway also has the capability to connect to the MQTT server which makes this implementation easier as it will publish real time data always and client all around the world can get the data though MQTT as a subscriber to the specific topic.

In both of the scenario, QoS level will be 2 as it guarantees the messages arrival only once and it is the safest and slowest QoS level.

5.2 Implementation with CoAP

A modified web transfer protocol called Constrained Application Protocol (CoAP) is used on the Internet of Things with constrained nodes and networks.

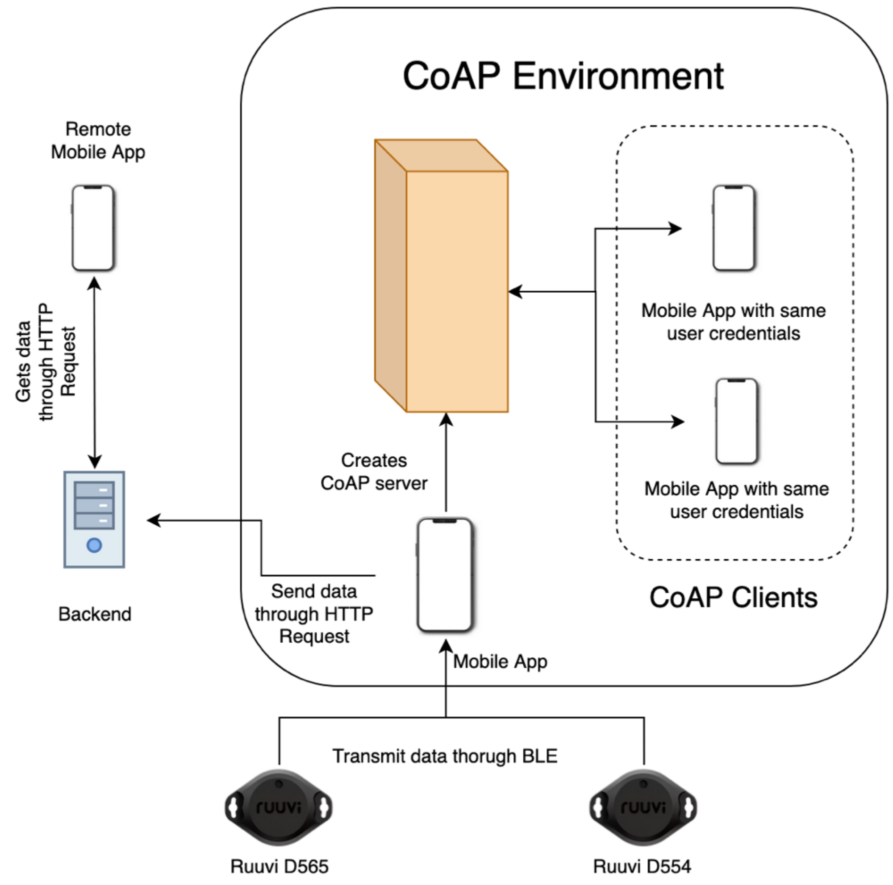

CoAP Environment
Remote Mobile App
Gets data through HTTP Request
Mobile App with same user credentials
Mobile App with same user credentials
Creates CoAP server
Backend
Send data through HTTP Request
CoAP Clients
Mobile App
Transmit data thorugh BLE
Ruuvi D565
Ruuvi D554

Figure 11. *Scenario 1 implementation with CoAP.*

From Figure 11, in the case of RuuviTag sensor and the mobile app, after getting data from the sensor through Bluetooth the mobile app start listening to a specific CoAP server. Any other client as mobile app wants to have the same sensor data then it tries to listen the server, but the server is already running thus it will get error. Then that client app will request for the data and will get data as a response. For sending data from client app to backend no proxy server is needed in this case. Mobile app will directly send data to the backend using a REST API.

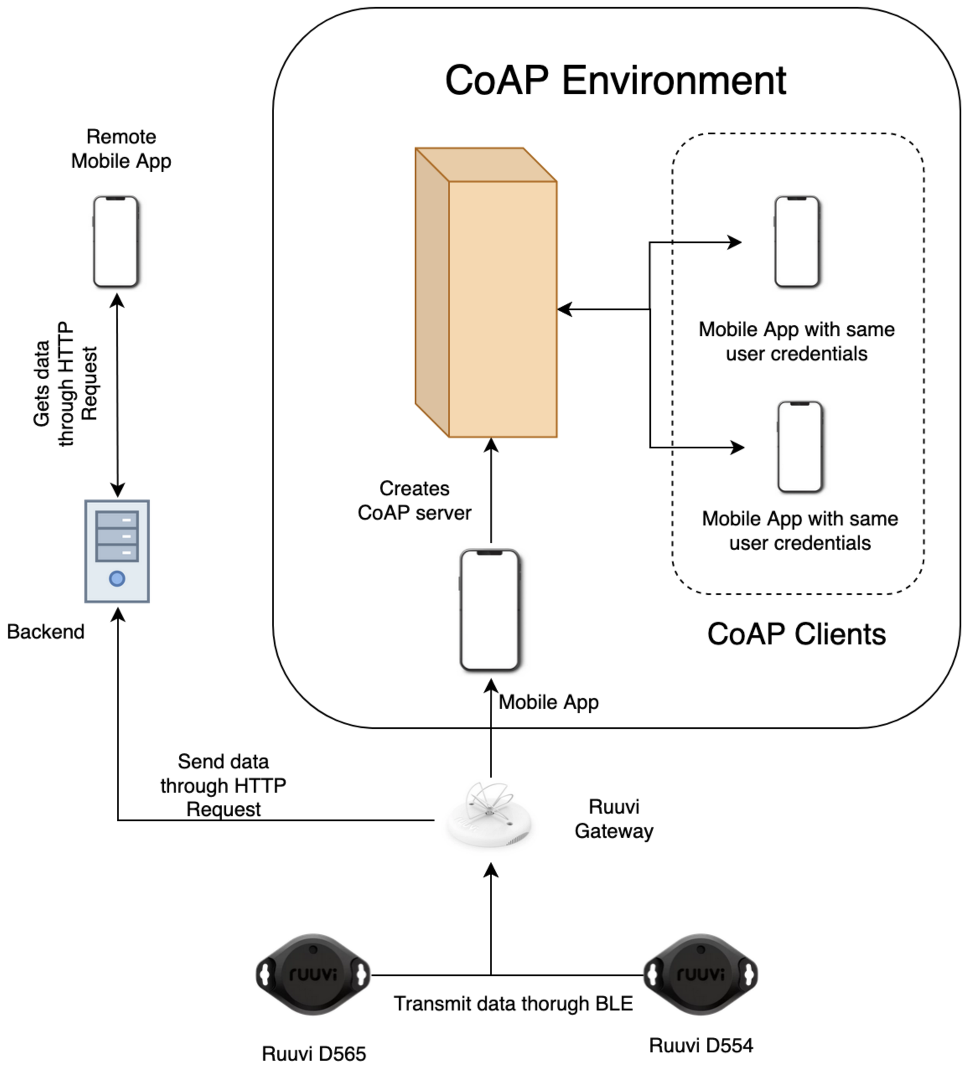

CoAP Environment
Remote Mobile App
Gets data through HTTP Request
Backend
Creates CoAP server
Mobile App with same user credentials
Mobile App with same user credentials
CoAP Clients
Mobile App
Send data through HTTP Request
Ruuvi Gateway
Transmit data thorugh BLE
Ruuvi D565
Ruuvi D554

Figure 12. *Scenario 2 implementation with CoAP.*

From Figure 12, in the case of Ruuvi Gateway along with RuuviTag sensor, the implementation gets a little bit complicated. As previous scenario mobile app will create the CoAP server. Ruuvi Gateway will collect data from nearby RuuviTag sensors. Mobile app will get data from Ruuvi Gateway through Bluetooth. Another usage of Ruuvi Gateway can be sending data to the backend by setting up the backend as a dedicated server of the gateway.

5.3 Implementation with AMQP

AMQP is one of the standard messaging protocols that is mostly used in open-source application development.

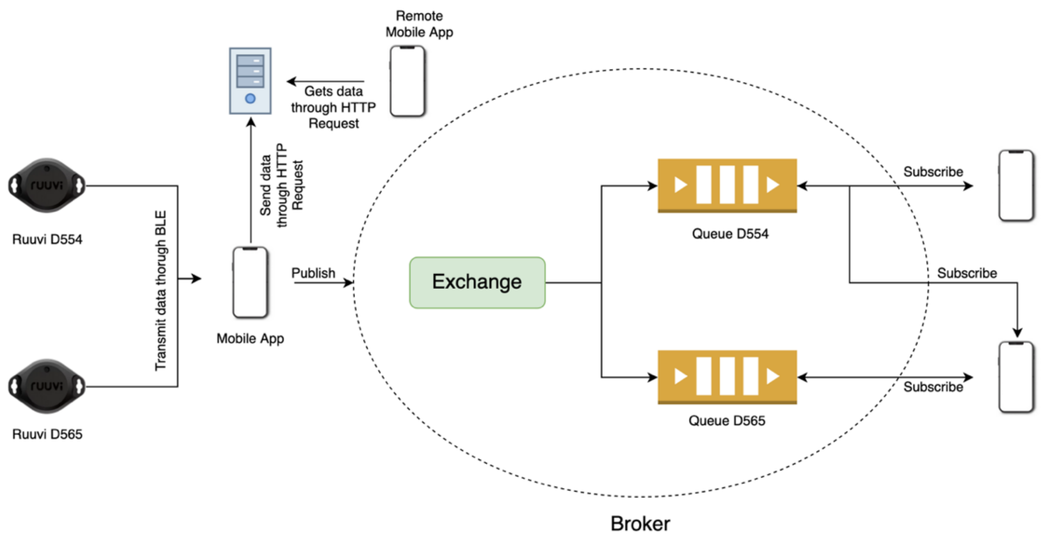


Figure 13. *Scenario 1 implementation with AMQP.*

In this research, AMQP is used as both the RuuviTag sensor and gateway, which are open-source platforms. In usage of AMQP from Figure 13, an exchange and a single queue for each of the RuuviTag sensor is used. The mobile app will establish a connection with specific configuration of AMQP. The connection will create an exchange and queue for the sensor that is connected. The name of exchange will be fixed all around the AMQP network. The name of the queue will be based on the RuuviTag sensor name. Then queue will be bind together with the exchange. After establishing all requirements of AMQP when the mobile app will get data from RuuviTag sensor it will publish data through exchange using a specific routing key and other clients that are subscribed to the queue will get the data as a message. Mobile app will directly send data to the backend using a REST API.

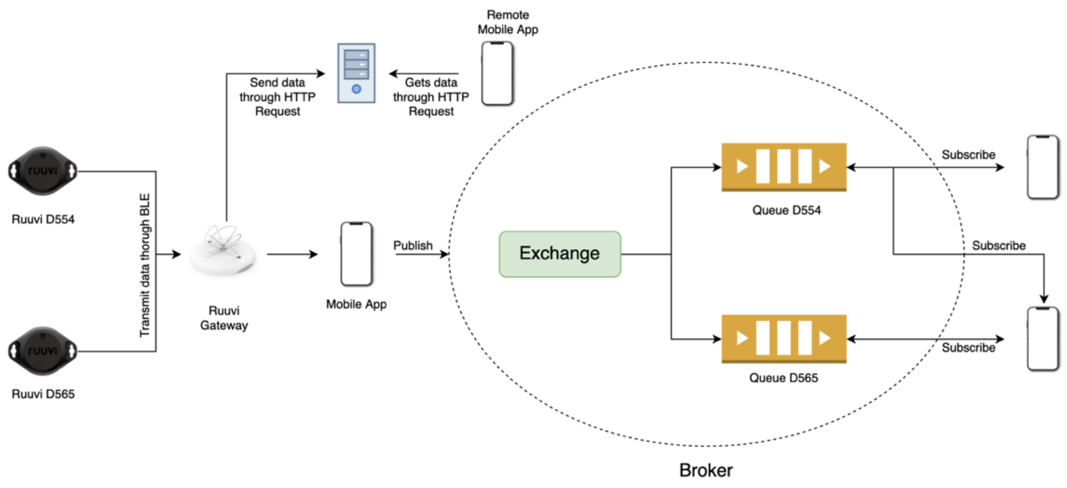


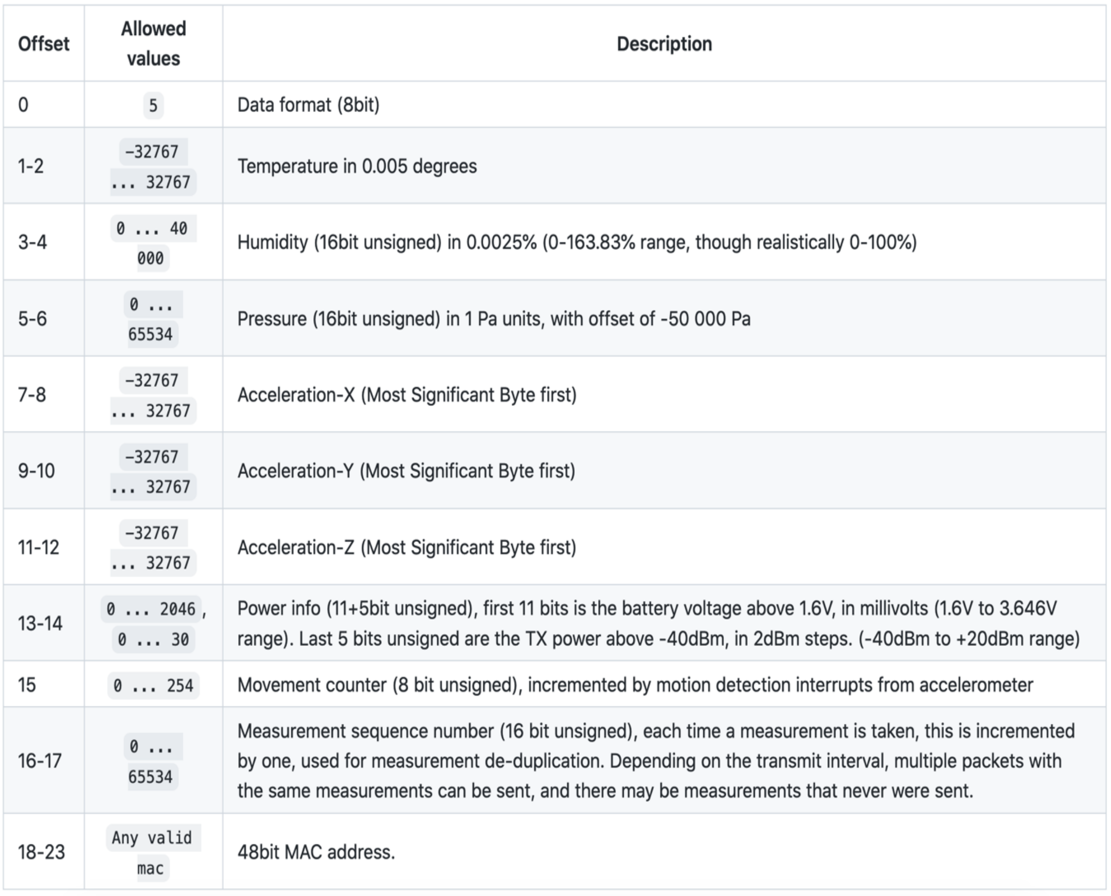
Figure 14. *Scenario 2 implementation with AMQP.*

From Figure 14, the Ruuvi Gateway does not have much impact in case of AMQP protocol. Process is the almost same as the first scenario. Only impact can be found that Ruuvi Gateway can send data to the backend.

5.4 RuuviTag Sensor Data Process

RuuviTag sensor sense data from surroundings and send those data through Bluetooth. These data that RuuviTag sensor sends have a specific format. RuuviTag sensor sends data into raw binary data and the data format is 5 protocol specification (RAWv2). Data sent in total of 24 offsets. Different offset or pair of offsets or a number of offsets represents different data.

Following this Figure 15 retrieving data such as temperature, humidity, air pressure and so on can easily be extracted from the raw data. For temperature, converting the binary data in decimal data and multiplying this with 0.005 we can get actual temperature that RuuviTag sensors wants to send. All other data can be achievable regarding this table in the Figure 15. There is an exception in movement counter. Movement counter returns value from 0 to 254 only. When the movement number is 255 RuuviTag starts counter from 1 again. If other values that is not mentioned in the table are considered to be invalid. Any invalid value means sensor cannot sense its surroundings for that specific asset.



Offset	Allowed values	Description
0	`5`	Data format (8bit)
1-2	`-32767 ... 32767`	Temperature in 0.005 degrees
3-4	`0 ... 40 000`	Humidity (16bit unsigned) in 0.0025% (0-163.83% range, though realistically 0-100%)
5-6	`0 ... 65534`	Pressure (16bit unsigned) in 1 Pa units, with offset of -50 000 Pa
7-8	`-32767 ... 32767`	Acceleration-X (Most Significant Byte first)
9-10	`-32767 ... 32767`	Acceleration-Y (Most Significant Byte first)
11-12	`-32767 ... 32767`	Acceleration-Z (Most Significant Byte first)
13-14	`0 ... 2046`, `0 ... 30`	Power info (11+5bit unsigned), first 11 bits is the battery voltage above 1.6V, in millivolts (1.6V to 3.646V range). Last 5 bits unsigned are the TX power above -40dBm, in 2dBm steps. (-40dBm to +20dBm range)
15	`0 ... 254`	Movement counter (8 bit unsigned), incremented by motion detection interrupts from accelerometer
16-17	`0 ... 65534`	Measurement sequence number (16 bit unsigned), each time a measurement is taken, this is incremented by one, used for measurement de-duplication. Depending on the transmit interval, multiple packets with the same measurements can be sent, and there may be measurements that never were sent.
18-23	`Any valid mac`	48bit MAC address.

Figure 15. *Data Format 5 Protocol Specification (RAWv2) of RuuviTag Sensor*

5.5 Protocol Comparison

From the implementation with three protocols, it is understandable that different protocol has different workflow. Accuracy, flexibility, and scalability of protocols also differs from each other. Comparing protocols will lead to a result to choose a protocol that is the best fit for this research. Each protocol will have its own drawback, but the protocol will less drawback in this research will be the ultimate result.

5.5.1 Complexity

Implementation of three protocols comes up with different complexity. With MQTT in the scenario of mobile app and RuuviTag sensor, at least one device with the mobile app will have to publish sensor data so that other client app can have the sensor data remotely. This problem can be solved when Ruuvi Gateway is in action because gateway has the MQTT capability, and it will always publish data for the remote client.

CoAP protocol needs to have a specific CoAP server. The complexity starts when there is two or more users wants to be in the same CoAP environment. The mobile app by default will have the capability to create the CoAP server. When another user in the same CoAP environment starts the mobile app, it will try to create same CoAP server and return an error. Thus, this error needs to be handled. While using CoAP as protocol in this research, the two scenarios does not come up with that much different implementation as it is with MQTT. The Ruuvi Gateway can only send data to the mobile app through Bluetooth and also can send data to the dedicated backend.

AMQP protocol implementation on React Native app is never developer friendly. AMQP is not backward version compatible. If current version become deprecated, then upgrading to a new version will not support deprecated items from old version. Data transfer with AMQP protocol can have more than one exchange and queue. In this research only one exchange is enough to implement the whole case. So, it is assumed that AMQP should be used in larger cases. AMQP with Ruuvi Gateway has the same issue as CoAP. Thus, AMQP also cannot have much impact on both of the scenarios as it is in MQTT.

5.5.2 Accuracy and Scalability

The ability to transfer data more precisely is considered as the accuracy of a protocol. QoS is one of the main factors in terms of accuracy of data transmission. In terms of QoS, MQTT is the strongest protocol among three because MQTT supports 3 QoS. In this research, QoS 2 has been used. QoS 2 has the guarantee that the receiver will receive the message only once. Although, QoS 1 has more latency than QoS 0 and QoS 2 has the highest average latency other two kind of QoS [45]. In this research accuracy of the data transmission is more important than latency and QoS 2 fits perfectly. In term of QoS, CoAP provides only two kinds of QoS which is Confirmable and Non-Confirmable. In Confirmable QoS an acknowledgement is received in the sender side which is missing in terms of Non-Confirmable. Confirmable QoS has been used in this research while implementing CoAP. AMQP also has two QoS and they are Settle format that is close to QoS 0 of MQTT which means at most once and Unsettle format that is kind of similar to QoS 1 of MQTT which means at least once. Unsettle format with an acknowledgement has been used in this research. From above comparison of accuracy among protocols MQTT is ahead of other two in this research as because it provides QoS 2 which suits most.

Scalability of a mobile application refers to the capacity of the app to handle its growth, more particularly in managing any number of users and evolving new features according to the needs. Scalability with CoAP is the lowest among three. The main problem of scalability of CoAP comes with creating CoAP server. More users create error while creating CoAP server which reduces scalability of the app. If the user with CoAP server suspends the mobile app then the environment of CoAP will break down. This case does not make the app scalable at all. Getting live data remotely demands a user with app always sending data to the backend. As previously discussed, AMQP can have as much queue as the case demand. In this research while implementing AMQP only one queue is enough to handle the case. Thus, in this research is a small aspect for AMQP implementation which does not make the mobile app scalable. MQTT solves above scalable problems as it can handle any number of users and any user that suspends the mobile app does not affect another user of the mobile app. The architecture and workflow of MQTT is the best fit for the case of this research. MQTT capability Ruuvvi Gateway adds extra benefit to get live data remotely makes MQTT stay ahead of other protocols.

5.5.3 Security

IoT protocols must prioritize security because IoT devices are linked to the internet and could be targets of cyberattacks. The risks connected with insecure devices increase in importance as IoT devices become more integrated into our daily lives. IoT protocols must have highly secured features like encryption, authentication, and access control in order to reduce these risks [46]. The security of IoT devices should also be taken into consideration during design, with safe default settings, frequent firmware updates, and other security features.

In this research, protocol compatibility was main issue rather than security. In terms of security, one main concern was that any third-party attack can change the data of RuuviTag sensor. Thus, some wrong data will be shared across the system and the decisions that would be taken based on those data will eventually be wrong. In this study, clients must be authenticated before they are permitted to connect to the broker in order to have a secure data transmission over MQTT. The username and password for this authentication are fixed in stone. For security in CoAP a token-based authentication is implemented. When the client does not have an authentication token it will ask for one authentication token with specific username and password. The server will response with a random authentication token and for each request this token will be sent. Without the token server will not response with the data from RuuviTag sensor. Message encryption is used in AMQP as a data transmission protection. A third-party library named “crypto-js” is used to encrypt and decrypt data in this research. It can be presumed from the above discussion of security that MQTT offers more robust security than other protocols, and that the security is also straightforward to implement.

5.6 Limitations

Every research study comes up with some limitations and this research study is not different from others. This research has some limitations that are more static such as fixed technologies, specific case scenario, predefined data format and so on. The technologies that used in this research such as React Native, RuuviTag sensor and Ruuvi Gateway are fixed and there is no way to try this research into other technologies. Therefore, it shortens the area of research. The case scenario of this research is very much specific. Only RuuviTag sensor and Ruuvi Gateway creates two scenarios and research has been conducted based on these scenarios only. The data that sent from the RuuviTag sensor are predefined from the sensor. For assumption if there are more sensor then there will be more format of data comes into action and these data transmission with protocols may open other scope of comparison between protocols.

6. CONCLUSION

The field of Internet of Things (IoT) is rapidly growing and has a wide range of potential applications that are already starting to change a number of sectors. The increased productivity and efficiency the Internet of Things can bring to different industries is one of its most important benefits. IoT also has the potential to make our homes and cities smarter, safer, and more sustainable. Although there are valid concerns about the security and privacy of IoT devices, there are also numerous potential benefits to be gained from this technology.

The MQTT, CoAP, and AMQP protocols—key elements of the Internet of Things (IoT) ecosystem—have been thoroughly reviewed and analyzed in this thesis work. The suitability of each protocol for IoT applications depends on several variables, including the particular use case, network topology, and device capabilities. Each protocol has its own distinctive features and capabilities. IoT applications that need real-time data exchange and low power usage are perfect candidates for the lightweight, effective protocol MQTT. On the other hand, CoAP is a RESTful protocol that offers an industry-standard method of getting and manipulating resources on constrained devices with limited bandwidth. AMQP is a strong and flexible messaging protocol that provides cutting-edge features like message queuing, routing, and transactional delivery. MQTT is appropriate for small-scale IoT projects that need real-time data exchange and minimal power consumption. While AMQP is suitable for large-scale IoT projects that need advanced messaging capabilities and guaranteed delivery, CoAP is suitable for applications that need minimal bandwidth and resource-constrained devices.

In summary of this research, implementation complexity of MQTT is less than other two protocols. With MQTT mobile applications can easily be connected to the environment with some specific attributes such as host, port, ClientID, username, password and so on. Because of less complex implementation MQTT ensures scalability to the mobile app. For CoAP environment, CoAP server error handling is mandatory as because one mobile app once create the server another one will get error while creating the same server. AMQP does not support backward version which a major drawback for this thesis work. MQTT comes out to have more accuracy than other two because of its QoS support. For secure transmission, MQTT client needs to authenticate through username and password. The compatibility of a protocol is the main focus of this thesis work, and MQTT is prioritized over the other two protocols.

REFERENCES

[1] Burak H. Çorak, Feyza Y. Okay, Metehan Güzel, Suat Ozdemir, “Comparative Analysis of IoT Communication Protocols”*,* 2018 International Symposium on Networks, Computers and Communications (ISNCC), *IEEE.*

[2] Manuel Silverio-Fernández, Suresh Renukappa, Subashini Suresh, “What is a smart device? – a conceptualisation within the paradigm of the internet of things”, *Vis. in Eng.* 6, 3, 2018, *Springer*.

[3] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, “A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration”, ACM Computing Surveys, Volume. 51, No. 6, Article 116, *Association for Computing Machinery*.

[4] Usman Ahmad, Junaid Chaudhary, Mudassar Ahmad, Amjad Ali Naz, “Survey on Internet of Things (IoT) for Different Industry Environments”, 2019, *Annals of Emerging Technologies in Computing* Volume: 3, No. 28-43.

[5] Radouan Ait Mouha, “Internet of Things”, 2021 *Journal of Data Analysis and Information Processing*.

[6] Sachin Kumar, Prayag Tiwari, Mikhail Zymbler, “Internet of Things is a revolutionary approach for future technology enhancement: a review”, 2019, *Journal of Big Data 6*, Article number: 111.

[7] Neerendra Kumar, Pragti Jamwal “Analysis of Modern Communication Protocols for IoT applications”, 2021 *Karbala International Journal of Modern Science* 7(4).

[8] Zainab H. Ali, Hesham A. Ali, Mahmoud M. Badawy “Internet of Things (IoT): Definitions, Challenges, and Recent Research Directions”, 2015, *International Journal of Computer Applications* 128(1):975-888, Accessed – 19/09/2022.

[9] Somayya Madakam, R. Ramaswamy, Siddharth Tripathi “Internet of Things (IoT): A Literature Review”, 2015, *Journal of Computer and Communications,* Volume: 3, No. 5.

[10] Fernando A. Aires Lins, Marco Vieira “Security Requirements and Solutions for IoT Gateways: A Comprehensive Study”, 2021, *IEEE Internet of Things Journal* Volume: 8, Issue 11, 01.

[11] Gunjan Beniwal, Anita Singhrova “A systematic literature review on IoT gateways”, 2022, *Journal of King Saud University - Computer and Information Sciences*, Volume: 34, Issue 10, Part B,

[12] Srinivasa A H, Dr. Siddaraju, “A Comprehensive Study Of Architecture, Protocols And Enabling Applications In Internet Of Things (Iot)”, 2019, *International Journal of Scientific & Technology Research,* Volume: 8, Issue: 11.

[13] Chang-Le Zhong, Zhen Zhu, Ren-Gen Huang, “Study on the IOT Architecture and Gateway Technology”, 2015, 14th *International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pp. 196-199.

[14] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, Huiying Du, “Research on the architecture of Internet of Things”, 2010, *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*. Volume: 5.

[15] F. Flammini, D. Dobrilović, A. Gaglione and D. Tokody, “LoRaWAN Technology Mapping to LayeredIoT Architecture”, 2020, Conference: *AIIT - International Conference on Applied Internet and Information Technologies* At: Zrenjanin, Serbia.

[16] Anas M Mzahm, Mohd Sharifuddin Ahmad, Alicia Y. C. Tang, “Agents of Things (AoT): An Intelligent Operational Concept of the Internet of Things (IoT)”, 2013, Conference: *13th International Conference on Intelligent Systems Design and Applications (ISDA 13)*, Malaysia At: Malaysia.

[17] Hubert C. Y. Chan, “Internet of Things Business Models”, 2015, *Journal of Service Science and Management*, pp.552-568.

[18] M. B. Yassein, M. Q. Shatnawi and D. Al-zoubi, "Application layer protocols for the Internet of Things: A survey", 2016, *International Conference on Engineering & MIS (ICEMIS)*, Agadir, Morocco, 2016, pp. 1-4.

[19] Zhexuan Song, Alvaro A. Cardenas, Ryusuke Masuoka, *“Semantic middleware for the Internet of Thing”*, 2010, *IEEE conference on Internet of Things (IOT)*, pp. 1-8.

[20] Pallavi Sethi, Smruti R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications", 2017, *Journal of Electrical and Computer Engineering*, Volume: 2017, Article ID: 9324035, 25 pages.

[21] Alhari Hanumat Prasad, T. Hema Bharat, “Network Routing Protocols in IoT”, 2017, *International Journal of Advances in Electronics and Computer Science*, ISSN: 2393-2835, Volume: 4, Issue: 4.

[22] Cheena Sharma, Dr. Naveen Kumar Gondhi, “Communication Protocol Stack for-Constrained IoT Systems”, 2018, *3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pp. 1 – 6.

[23] P.V. Vijaya Durga, T-Sujith Kumar, “A Literature Survey on Internet of Things”, 2020, *International Journal for Research in Engineering Application & Management (IJREAM)*, ISSN: 2454-9150, Volume: 06, Issue :09.

[24] K.Aarika a, M.Bouhlal , R.AitAbdelouahid , S.Elfilali, E.Benlahmar “Perception layer security in the internet of things”, 2020, *International workshop on Artificial Intelligence & Internet of Things (A2IoT)*, Leuven, Belgium.

[25] Adrian Pekar, Jozef Mocnej, Winston K.G. Seah, Iveta Zolotova, “Application Domain based Overview of IoT Network Traffic Characteristics”, 2020, *ACM Computing Surveys* 53(4):1-33.

[26] M. Gokilavani, Ashin Baiju, Unnikrishnan K N, Sunder.V, Ajay Basil Varghese, “A Study on IoT Architecture For IoT Application Domains”, 2019, *IJIRIS :: AM Publication*, India.

[27] Michael Friedewald, Olivier Da Costa, Yves Punie, Petteri Alahuhta, Sirkka Heinonen, “Perspectives of ambient intelligence in the home environment.”, 2005, *Telematics Informatics*, Volume: 22, Pages: 221–238.

[28] Muhammad Raisul Alam, Mamun Bin Ibne Reaz, Mohd Alauddin Mohd Ali, “A review of smart homes—Past, present, and future”, 2012, *IEEE Transaction on Systems*, Man, and Cybernetics, Part C, Volume: 42, Issue: 6.

[29] Mrs.Jyotsna P. Gabhane, Ms. Shradha Thakare, Ms. Monika Craig, “Smart Homes System Using Internet-of-Things: Issues, Solutions and Recent Research Directions”, 2017, *International Research Journal of Engineering and Technology (IRJET)* e-ISSN: 2395 -0056 Volume: 04 Issue: 05.

[30] Gaurav Tripathi, Dhananjay Singh, and Antonio J. Jara, “A survey of Internet-of-Things: Future Vision, Architecture, Challenges and Service”, 2014, *IEEE World Forum on Internet of Things (WF-IoT)*, pp. 287-292.

[31] Ming Wang, Guiqing Zhang, Chenghui Zhang, Jianbin Zhang, and Chengdong Li, “An IoT-based Appliance Control System for Smart Homes”, 2013, *Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 744-747.

[32] Jamuna M, A.M Vijaya Prakash, “A Study of Communication Protocols for Internet of Things (IoT) Devices: Review”, 2021, Atlantis Highlights in Computer Sciences, Volume: 4, *Proceedings of the 3rd International Conference on Integrated Intelligent Computing Communication & Security*.

[33] Poorana Senthikumar S, Bhavadharani Subramani,“Study on IoT Architecture, Application Protocol and Energy needs”, 2020, *IJRNSC*, Volume-8, Issue-5.

[34] M V Masdani, D Darlis, “A comprehensive study on MQTT as a low power protocol forinternet of things application”, 2018, IOP Conf. *Ser.: Mater. Sci. Eng*. 434 012274.

[35] Poonam G, Indra Om Prabha. M, “A Survey of Application Layer Protocols for Internet of Things”*,* 2021, *IEEE*, ICCICT.

[36] Pinchen Cui, “Comparison of IoT Application Layer Protocols”, 2017, thesis submitted to the Graduate Faculty of *Auburn University* in partial fulfilment of the requirements for the Degree of Master of Science, Auburn, Alabama.

[37] Anjum Sheikh, Asha Ambhaikar, Sunil Kumar, “Quality of Services Improvement for Secure Iot Networks”, 2019, *International Journal of Engineering and Advanced Technology (IJEAT)* ISSN: 2249-8958 (Online), Volume: 9 Issue: 2.

[38] Gaurav Tripathi, Dhananjay Singh, and Antonio J. Jara, “A survey of Internet-of-Things: Future Vision, Architecture, Challenges and Service”, 2014, *IEEE World Forum on Internet of Things (WF-IoT)*, pp. 287-292.

[39] Monika Bohara, “Comparison Between React Native and Native Application Development Platform”, 2020, Master of Science Thesis, *Tampere University*, master’s degree Programme in Information Technology.

[40] Oskar Svensson, Marcus Presa Käld, *“React Native and native application development: A comparative study based on features & functionality when measuring performance in hybrid and native applications”,* 2021, Independent thesis Basic level (degree of Bachelor), *Jönköping University*.

[41] Lloyd G. Williams, Connie U. Smith, "Web Application Scalability:A Model-Based Approach", 2004, Conference: *30th International Computer Measurement Group Conference*, Las Vegas, Nevada, USA, Proceedings.

[42] Pritam Manna, Rohit Kumar Das, “Scalability in Internet of Things: Techniques, Challenges and Solutions”, 2021, *International Journal for Research in Engineering Application & Management (IJREAM)* ISSN: 2454-9150 Volume: 07, Issue: 01.

[43] D. S. Ugalde, “Security Analysis for MQTT in Internet of Things”, 2018, *Degree project Computer Science and Engineering*, second cycle, 30 credits, Stockholm Sweden.

[44] Thamer A. Alghamdi, A. Lasebae, M. Aiash, “Security Analysis of the Constrained Application Protocol in the Internet of Things”, 2013, Conference: *Second International Conference on Future Generation Communication Technologies.*

[45] Cüneyt Bayılmış, M. Ali Ebleme, Ünal Çavuşoğlu, Kerem Küçük, Abdullah Sevin, "A survey on communication protocols and performance evaluations for Internet of Things", 2022, *Digital Communications and Networks*, Volume: 8, Issue: 6, Pages 1094-1104, ISSN 2352-8648.

[46] Nishat Tasnim Newaz, Mohammad Rafiqul Haque, Tajim Md. Niamat Ullah Akhund, Tania Khatun, Milon Biswas, Mohammad Abu Yousuf, “IoT Security Perspectives and Probable Solution”, 2021, *Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*.