

# Comparative Analysis of TLS 1.3 and Light Weight Protocol Over IoT Network



Author

BASIT ALI

Regn Number

0000329816

Supervisor

Asst. Prof. LT. CDR. DR. GUL SHAHZAD


DEPARTMENT OF CYBER SECURITY  
PAKISTAN NAVY ENGINEERING COLLEGE  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY  
KARACHI  
FEBRUARY, 2023


# National University of Sciences and Technology


## MASTER'S THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by: (Student Name & Regn No.) BASIT ALI (00000329816) Titled: Comparative Analysis of TLS 1.3 and Light weight protocol over IoT Network be accepted in partial fulfillment of the requirements for the award of Master's degree.

### EXAMINATION COMMITTEE MEMBERS

1. Name: Asst. Prof. Dr. Muhammad Usama Signature: 

2. Name: Asst. Prof. Dr. M. Ayaz Shirazi Signature: 

Supervisor's name: Asst. Prof. Lt. Cdr. Dr. Gul Shahzad PN Signature:   
Date: 22/12/23

  
AALIYA ALI  
Head of Department Pakistan Navy  
HOD CySD

22-12-2023  
Date

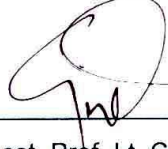
### COUNTERSIGNED

Date: 22-12-2023

DR NADEEM KURESHI  
Commodore  
DEAN MIS  
PNS JALPA  
  
Dean / Principal


## THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS thesis written by BASIT ALI Regn No. 00000329816 of NUST- PNEC (College) has been vetted by undersigned, found complete in all respects as per NUST Status/Regulations, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have been incorporated in the said thesis.

Signature:  \_\_\_\_\_

Name of Supervisor: Asst. Prof. Lt. Cdr. Dr. Gul Shahzad PN

Dated: 22-12-2023 \_\_\_\_\_

Signature: (HoD):  \_\_\_\_\_

Dated: 22-12-2023 \_\_\_\_\_  
AALIYA ALI  
Lt Cdr Pakistan Navy  
HOD CySD

Signature: (Dean/Principal):  \_\_\_\_\_

Dated: 22-12-2023 \_\_\_\_\_  
DR MADEEM KHURESH  
DEAN  
PNS JAMSHED

APPROVAL

It is certified that the contents and form of the thesis entitled "Comparative analysis of TLS 1.3 and Lightweight Protocol over IoT Network" submitted by "Basit Ali" have been found satisfactory for the requirement of the degree.

Advisor:  Lt Col Dr. Gul Shahnaz

Signature:  [Signature]

Date:  22-12-2023

Committee Member :  Dr. Muhammad Usama

Signature:  [Signature]

Date:  16 Feb 2024

Committee Member :  Dr. M Ayaz Shirazi

Signature:  [Signature]

Date:  22-02-2024

## CERTIFICATE FOR PLAGIARISM

1. It is certified that PhD / M.Phil / **MS** Thesis Titled "**Comparative Analysis of TLS 1.3 and Lightweight Protocol Over IoT Network** " by **BASIT ALI** (**2020NUST-MS Cyber Security (CyS Fall 20)**) has been examined by us. We undertake the follows:

- a. Thesis has significant new work / knowledge as compared already published or is under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.
- b. The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented as Author own work.
- c. There is no fabrication of data or results which have been compiled / analyzed.
- d. There is no falsification by manipulating research materials, equipment, or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.
- e. The thesis has been checked using TURNITIN (copy of originality report attached) and found within limits as per HEC Plagiarism Policy and instructions issued from time to time.

  
**DR GUL SHAHZAD**  
**Name & Signature of Supervisor**  
Head of Postgraduate Program  
Electronics & Power Engg  
NUST - PNEC

## Certificate of Originality

I certify that this research work titled "*Comparative analysis of TLS 1.3 and light weight protocol over IoT network*" is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.



Signature of Student

BASIT ALI

2020-NUST-MS-CYS-0000329816

## **Copyright Statement**

- Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST Pakistan Navy Engineering College (PNEC). Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the author.
- The ownership of any intellectual property rights which may be described in this thesis is vested in NUST Pakistan Navy Engineering College, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the PNEC, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST Pakistan Navy Engineering College, Karachi.

## Acknowledgements

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You setup in my mind to improve it. Indeed, I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout in every department of my life.

I would also like to express special thanks to my supervisor **Lt. Cdr. Dr. Gul Shahzad** for his help throughout my thesis. I can safely say that I haven't learned any other engineering subject in such depth than the ones which he has taught.

I would also like to pay special thanks to **Dr. Arshad Aziz** for his tremendous support and cooperation. Each time I got stuck in something, he came up with the solution. Without his help I wouldn't have been able to complete my thesis. I appreciate his patience and guidance throughout the whole thesis.

I would also like to thank **Dr. M. Ayaz Shirazi**, and **Dr. Muhammad Usama** for being on my thesis guidance and evaluation committee and express my special thanks to Hamza Qureshi for his help. I am also thankful to Muhammad Zubair Khan for their support and cooperation.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.



*This thesis is dedicated to my teachers, my affectionate mother, my father, family and friends...*

## Abstract

The usage of IoT devices worldwide is predicted to dramatically expand by 2024, resulting in a massive number of internet-connected gadgets. While this development is exciting and has the potential to provide real-time data to prevent critical situations that also poses significant security risk. It is therefore essential to prioritize the security of IoT networks and devices to prevent hackers from taking control of them on a large scale. One crucial aspect of IoT security is data encryption and device authentication. Our goal is to compare TLS 1.3, a standard internet communication protocol, with a lightweight protocol for Internet of Things devices in this research project. Using intelligent power profiling to optimize energy usage, the work aims to assess the protocols' performance, security, and energy consumption. This will allow for the proposal of a dependable, scalable, and secure method for IoT networks that addresses security concerns while effectively utilizing device power performance. Thus, the research is expected to contribute towards the development of more secure, energy efficient and reliable IoT systems.

**Key Words:** *MQTT, IoT Network, Power Profiling, Energy Consumption, TLS 1.3, Security Protocol, ESP32, Cyber Security, Arduino IDE, Deep-sleep mode, Monitoring*

# Table of Contents

|  |           |
|--|-----------|
| Certificate of Originality .....   | i         |
| Copyright Statement .....  | vi        |
| Acknowledgements .....   | vii       |
| Abstract .....   | ix        |
| Table of Contents.....   | x         |
| List of Figures .....  | xiii      |
| List of Tables.....  | xiv       |
| <b>CHAPTER 1: INTRODUCTION.....</b>  | <b>1</b>  |
| 1.1    Background, Scope and Motivation .....  | 1         |
| 1.1.1    Perception Layer and Security Attacks .....   | 2         |
| 1.1.2    Potential Attacks and Threats to IoT Network.....   | 3         |
| 1.2    Chapter Structure .....   | 4         |
| <b>CHAPTER 2: UNDERSTANDING IOT AND ITS APPLICATIONS, NETWORKS AND APPLICATION PROTOCOLS .....</b> | <b>6</b>  |
| 2.1    Internet of Things .....  | 6         |
| 2.2    Trending Applications of IoT.....   | 7         |
| 2.2.1    Smart Cities and Urban Management .....   | 7         |
| 2.2.2    Precision Agriculture .....   | 8         |
| 2.2.3    Health Care Monitoring and Remote Care .....  | 8         |
| 2.2.4    Industrial IIOT for Manufacturing.....  | 8         |
| 2.2.5    Energy Management and Smart Grids .....   | 9         |
| 2.2.6    Environmental Monitoring and Conservation.....  | 9         |
| 2.2.7    Retail and Customer Experience Enhancement .....  | 10        |
| 2.2.8    Connected Transportation and Logistics.....   | 10        |
| 2.2.9    Smart Home Automation .....   | 11        |
| 2.2.10    Water Management and Conservation .....  | 11        |
| 2.3    IoT Networks .....  | 12        |
| 2.4    IoT Messaging Protocols.....  | 13        |
| 2.4.1    HTTP .....  | 13        |
| 2.4.2    AMQP.....   | 14        |
| 2.4.3    CoAP .....  | 15        |
| 2.4.4    MQTT.....   | 16        |
| <b>CHAPTER 3: CHALLENGES OF SECURING IOT AND ROLE OF SECURITY PROTOCOLS .....</b>                  | <b>19</b> |
| 3.1    Challenges in Securing IoT .....  | 19        |
| 3.2    Role of Security Protocols .....  | 19        |
| 3.2.1    Confidentiality .....   | 20        |

|                   |  |           |
|-------------------|--|-----------|
| 3.2.2             | Integrity .....  | 20        |
| 3.2.3             | Availability .....                                       | 21        |
| 3.3               | Standard Security Protocols and OSI Model Layer .....    | 21        |
| 3.3.1             | IPsec – Layer 3 .....                                    | 22        |
| 3.3.2             | Transport Layer Security – Layer 5 .....                 | 22        |
| 3.3.3             | Datagram transport layer security – Layer 5 .....        | 24        |
| 3.3.4             | Wireless transport layer security – Layer 5 .....        | 24        |
| 3.3.5             | Simple Network Management -Layer 7 .....                 | 25        |
| 3.3.6             | Kerberos Protocol-Layer 7 .....                          | 25        |
| <b>CHAPTER 4:</b> | <b>LITERATURE REVIEW .....</b>                           | <b>26</b> |
| 4.1               | Analysis of IoT Applications .....                       | 26        |
| 4.1.1             | Previous Versions of TLS Protocol .....                  | 30        |
| 4.1.2             | Improvements Through TLS 1.3 .....                       | 31        |
| 4.2               | Related Research .....                                   | 32        |
| 4.2.1             | Related Work .....                                       | 32        |
| 4.2.2             | Significance of Research Work .....                      | 33        |
| 4.2.3             | Research Papers Summary .....                            | 34        |
| 4.3               | Problem Statement .....                                  | 35        |
| 4.4               | Objective .....  | 35        |
| <b>CHAPTER 5:</b> | <b>PROPOSED WORK .....</b>                               | <b>36</b> |
| 5.1               | Architecture of The IoT Network .....                    | 36        |
| 5.2               | Comparison Parameters .....                              | 37        |
| 5.3               | Testing Methodology .....                                | 38        |
| 5.4               | Implementation Platform .....                            | 39        |
| 5.5               | Experimental Setup .....                                 | 39        |
| 5.5.1             | Sensor Node Setup .....                                  | 40        |
| 5.5.2             | MQTT Server Deployment .....                             | 41        |
| 5.5.3             | Monitoring Through MQTT Client .....                     | 42        |
| 5.5.4             | Private Certificate Authority to Implement TLS 1.3 ..... | 42        |
| 5.5.5             | Our Network Analysis .....                               | 48        |
| 5.5.6             | Current Measurement .....                                | 48        |
| <b>CHAPTER: 6</b> | <b>RESULT &amp; DISCUSSION .....</b>                     | <b>50</b> |
| 6.1               | Comparison Table for Network Architectures .....         | 50        |
| 6.2               | Experiment Results .....                                 | 50        |
| 6.2.1             | MQTT Implement Libraries .....                           | 50        |
| 6.2.2             | Power Profiling with Power Modes Results .....           | 51        |
| 6.3               | Reading Results Analysis .....                           | 52        |
| 6.3.1             | Formula to Calculate the Battery Life .....              | 52        |
| 6.3.2             | Uses Cases for Our Battery life Calculations .....       | 53        |

|  |           |
|--|-----------|
| <b>CHAPTER 7: CONCLUSION &amp; FUTURE WORK</b> ..... | <b>56</b> |
| 7.1 Conclusion .....                                 | 56        |
| 7.2 Future Work.....                                 | 57        |
| <b>REFERENCES</b> .....                              | <b>58</b> |

## List of Figures

|  |    |
|--|----|
| <b>Figure 1.1:</b> IoT Security Architecture.....                      | 3  |
| <b>Figure 2.1:</b> IoT Ecosystem.....                                  | 7  |
| <b>Figure 2.2:</b> IoT Network.....                                    | 12 |
| <b>Figure 2.3:</b> IoT Messaging Protocol Stack.....                   | 13 |
| <b>Figure 2.4:</b> HTTP Architecture.....                              | 14 |
| <b>Figure 2.5:</b> AMQP Architecture.....                              | 15 |
| <b>Figure 2.6:</b> CoAP Architecture.....                              | 16 |
| <b>Figure 2.7:</b> Trends of IoT Messaging Protocols.....              | 17 |
| <b>Figure 2.8:</b> MQTT Architecture.....                              | 18 |
| <b>Figure 3.1:</b> CIA Triad.....                                      | 20 |
| <b>Figure 3.2:</b> TLS Protocol in OSI Model.....                      | 23 |
| <b>Figure 4.1:</b> Power Consumption Vs Resources Requirement.....     | 28 |
| <b>Figure 4.2:</b> Reliability Vs Interoperability.....                | 29 |
| <b>Figure 5.1:</b> Overview of IoT Network for Monitoring.....         | 36 |
| <b>Figure 5.2:</b> Proposed IoT Network Topology.....                  | 37 |
| <b>Figure 5.3:</b> Parameters Consideration.....                       | 38 |
| <b>Figure 5.4:</b> Testing Methodology.....                            | 38 |
| <b>Figure 5.5:</b> Implementation Platform.....                        | 39 |
| <b>Figure 5.6:</b> Experimental Setup.....                             | 39 |
| <b>Figure 5.7:</b> Arduino IDE and Serial Monitor.....                 | 41 |
| <b>Figure 5.8:</b> MQTTX Client Interface.....                         | 42 |
| <b>Figure 5.9:</b> Network Analysis With Wireshark.....                | 48 |
| <b>Figure 5.10:</b> Current Measurement Using Multi-meter.....         | 49 |
| <b>Figure 6.1:</b> General Formula to Calculate Battery Life [64]..... | 52 |
| <b>Figure 6.2:</b> Derived Formula to Calculate Battery Life.....      | 53 |
| <b>Figure 6.3:</b> Battery Life Graph for Use Cases.....               | 55 |

## List of Tables

|  |    |
|--|----|
| <b>Table 4-1:</b> Messaging Protocol Comparative analysis for IoT Systems[45].....                   | 26 |
| <b>Table 4-2:</b> Summary Table of Research Papers.....  | 34 |
| <b>Table 6-1:</b> Comparison Table for Architectures .....   | 50 |
| <b>Table 6-2:</b> Reading Results for AsyncMQTT and Pub-Sub Client Library Current Consumption ..... | 51 |
| <b>Table 6-3:</b> Reading Results for Power modes with Pub-Sub Library Current Consumption.....      | 51 |

# CHAPTER 1: INTRODUCTION

There are two components to the research effort that is described in this dissertation. The first section deals with the implementation of an Internet of Things network using an IoT-friendly message protocol that secures communication in contrast to the application layer use of the standard HTTP protocol. In order to handle the energy consumption of the network's remote sensing nodes and compare energy consumption across situations, the second section suggests an energy-efficient method utilizing power profiling.

## 1.1 Background, Scope and Motivation

The risk of cyberattacks and illegal access has increased in tandem with the growth of IoT devices, making communication protocols in IoT networks a key issue in recent years [1]. Numerous studies have examined ways to lower end node energy consumption while preserving the necessary network security in order to address these security concerns. A thorough review paper on the subject addressed the shortcomings of current protocols and emphasized the significance of secure communication in Internet of Things networks [1]. The Transport Layer Security protocol (TLS) in version 1.3 is one of the primary protocols assessed in the comparative study. A popular protocol that offers safe internet communication is TLS 1.3.

One such alternative is the use of lightweight protocols that are specifically designed for resource-constrained IoT devices. These lightweight protocols aim to provide secure communication while minimizing the resource consumption of IoT devices. Which of course comes at some cost and in our case, it's the security which is reduced in terms of strength of encryption algorithm, key exchange method or trimming cipher suite.

Because the underlying communication architecture of IoT networks is often TCP/IP, it is desirable to adopt established security protocols in terms of implementation efficiency and dependability [2]. In this situation, the most viable option for protecting IoT connectivity is Transport Layer Security [2]. Nonetheless, a noteworthy constraint of TLS is that its widely used encryption suites were not initially intended for IoT devices with limited resources [2]. UDP-based alternatives, including Datagram Transport Layer Security, have been introduced as lightweight



methods to get around this restriction [2]. However, these UDP-based methods lose some of their advantages over TLS when security is applied. For secure communication in IoT networks, it is therefore frequently more desirable to employ TLS implementations that are widely available and optimized.

The comparative analysis of TLS 1.3 and lightweight protocols over IoT networks aims to evaluate the suitability of each protocol for secure communication in the context of resource constrained IoT devices and large-scale fog computing architectures. This analysis will consider factors such as resource consumption, security features, implementation efficiency, and scalability.

Researchers can learn more about the advantages and disadvantages of each protocol in the context of the Internet of Things by contrasting Security Protocols with lightweight application protocols.

To sum up, evaluating the effectiveness and applicability of each protocol for securing IoT communication requires comparing TLS 1.3 with lightweight protocols over IoT networks. This investigation focuses on assessing the usefulness and efficiency of lightweight protocols and TLS 1.3 in Internet of Things communication. by being aware of their network's requirements.

### **1.1.1 Perception Layer and Security Attacks**

Information collection and transmission in the real world is a crucial function of the Internet of Things. As a result, several data gathering, processing, and transmission devices, including Bluetooth, Zigbee, pressure and temperature sensors, are included in the perception layer.

The perception layer consists of two components: the perception network, which interfaces with the higher layer of the Internet of Things architecture, and the perception node, which can be a sensor, controller, etc. Perception nodes, such as sensors and actuators, collect and control information.

Perception layer network communication is secured through the use of key management techniques and cryptographic encryption algorithms. Without the need for intricate key management methods, device authentication can guarantee system security by utilizing a private key algorithm that is more scalable [1]. Information Processing Security Hierarchy provides information security and secrecy; this is mostly relevant to middleware security, privacy

protection, and other related issues. It aligns with application-decentralized security in the context of the Internet of Things.

Three hierarchies comprise the network structure of the Internet of Things. As seen in Figure 1, the lower hierarchy is the information-gathering sensor device; the middle hierarchy is the data transmission network; and the upper hierarchy is intended for middleware and applications [2].

|                   |  |                         |
|-------------------|--|-------------------------|
| APPLICATION LAYER | Application service data Security                              | Safety guarantee system |
| NETWORK LAYER     | Access and core network and information security               |                         |
| PERCEPTION LAYER  | Perception layer network transmission and information security |                         |
|                   | Perception layer local Security                                |                         |

**Figure 1.1:** IoT Security Architecture

### 1.1.2 Potential Attacks and Threats to IoT Network

The cloud-IoT architecture is susceptible to numerous dangers. Attack points in this kind of architecture might be IoT sensors, local user devices, and cloud gateways that provide cloud services like virtual machines, databases, and administration. Highlighted Such systems are vulnerable to spoofing, cloud, MITM, and packet sniffing attacks. Side-channel attacks are another type of cryptographic algorithm implementation that targets users. In order to comprehend what should be avoided, the attacks will be further described in the section that follows [3].

**Packet Sniffing** is the process of data interception via a communication channel. If the data is delivered in clear text or encrypted with a weak encryption method that is readable by an attacker, packet sniffing presents a serious risk.

**Man in the middle attack** (MITM) attack happens when someone snoops on and modifies data being transferred between an end user and an IoT sensor. The attacker can snoop on and change the data using this kind of assault. For instance, if cloud gateways are used by IoT sensors to transmit data to the cloud, an attacker may pose as one of these gateways and alter the data that is

transmitted from the sensors to the end user. An additional scenario is that certain staff members with access to the cloud service might read and alter the data before it is received by the customer. The data's security and integrity may be jeopardized by an MITM attack.

**Spoofing attack** happens when someone is impersonated by a user or program. nevertheless, by manipulating data. Here, by altering their MAC or IP address, an attacker can pretend to be a different user or Internet of Things device in order to retrieve or transmit messages. Additionally, if a message is occasionally reused, an attacker can seem to be a genuine user by replaying it.

**Side Channel Attacks** take advantage of data gleaned from the system. Concrete implementations are the focus of side-channel attacks, not abstract algorithms. Physical information can be leaked by attackers and used for their own purposes, such as time, energy usage, or electromagnetic radiation. Such assaults can be carried out simply by watching how the devices behave; they do not involve manipulating sensors. Invasive side-channel attacks are also possible. Cryptographic keys can be extracted from a sensor by an attacker if they manage to obtain a physically inexpensive sensor. IoT sensors frequently lack physical security. Although exploiting keys is a difficult task, it is possible.

## 1.2 Chapter Structure

The paper begins with giving a basic understanding of IoT, exploring various challenges that come with implementation of secure IoT Network. Afterwards; it proposes the scalable and reliable IoT architecture for huge number of applications in real life scenarios. Then it dives deep into security and impact of the implementation of TLS v1.3 over the MQTT and focuses to work out the better solution to gain long battery life for remote areas or wireless sensing nodes. Finally, the proposed work is evaluated by comparing it with relevant work using the key parameters before summarizing it and to identify the future research directions. The chapter-by-chapter breakdown is appended below:

- **Chapter 2** explains IoT and applications, IoT Networks, Application Protocols for IoT and its applications.
- **Chapter 3** Discusses available security protocols and schemes which are applicable to secure any network and then explores various challenges associated with acquiring the optimal energy consumption to maximize the battery operated micro-controller lifetime as well as securing the MQTT based IoT network.

- **Chapter 4** reviews previous literature on IoT security, Architecture and reducing the energy consumption of the micro controller and suitable IoT architecture.
- **Chapter 5** Presents the proposed suitable IoT architecture and comparison with traditional IoT networks.
- **Chapter 6** discusses the flow of our work in comparative analysis for energy consumption and implementation of test bed.
- **Chapter 7** talks about results of our Comparative analysis of secure and unsecured connection over MQTT protocol network.
- **Chapter 8** concludes our research work.

## CHAPTER 2: UNDERSTANDING IOT AND ITS APPLICATIONS, NETWORKS AND APPLICATION PROTOCOLS

### 2.1 Internet of Things

The network of physical "things" that have sensors, software, and other technologies integrated in them with the aim of communicating and sharing data with other devices and systems via the internet is known as the Internet of Things (IoT). These gadgets vary from commonplace home items to highly advanced industrial instruments. IoT has emerged as one of the key technologies of the twenty-first century in the last few years. Smooth communication between people, processes, and things is now feasible because to the ability to connect commonplace items like baby monitors, cars, thermostats, and kitchen appliances to the internet via embedded devices. Physical objects can communicate and collect data with minimal human intervention thanks to low-cost computers, the cloud, big data, analytics, and mobile technologies. Digital systems have the capacity to capture, track, and modify every interaction between connected objects in our highly networked environment. When the real and digital worlds collide, they work together. Although the concept of the Internet of Things has been around for a while, it has just been feasible due to a number of technological advancements. Low-cost, low-power sensor technology: More firms are able to utilize IoT technology thanks to the availability of reasonably priced, dependable sensors. Connectivity: A variety of internet-based network protocols have made it simple to link sensors to other "things" and the cloud for effective data transfer.



## **Figure 2.1: IoT Ecosystem**

Platforms for cloud computing: As cloud platforms become more widely available, consumers and companies can get the infrastructure they require to scale up without having to handle it all. Machine learning and analytics: Businesses can now quickly and readily obtain insights thanks to advancements in these fields as well as access to a wide variety of massive amounts of data stored on the cloud. The development of these complementary technologies keeps pushing the limits of IoT, and these technologies also rely on the data generated by IoT. Artificial intelligence (AI) for dialogue. Natural-language processing, or NLP, is now available for Internet of Things (IoT) devices (like Alexa, Cortana, and Siri), making them enticing, practical, and economical for usage at home thanks to advancements in neural networks.

## **2.2 Trending Applications of IoT**

The Internet of Things (IoT) is an exemplary instance of innovation in a rapidly evolving technological world, bringing in a new era of connectivity and change. As we look towards 2023 and beyond, the potential for IoT to reshape industries and revolutionize everyday life continues to grow. This section explores into the some futuristic IoT applications that are reshaping industries and pushing conventional boundaries. From improving efficiency and connectivity to driving sustainable transformation, these applications showcase the remarkable capabilities of IoT.

### **2.2.1 Smart Cities and Urban Management**

The Internet of Things, or IoT, is a major factor in the development of smart cities, which maximize available resources and enhance the standard of living for citizens. IoT allows physical items to connect to other devices and systems over the internet and exchange data by integrating sensors, software, and other technologies into the object itself. Thanks to this technology, cities can now adapt more dynamically to changing conditions, use less energy, and operate more efficiently. Examples of these systems include trash management programs, smart lighting, and real-time traffic monitoring systems. For example, smart streetlights can save up to 65% on energy costs since they can be remotely controlled and continuously monitored [3]. Similarly, real-time garbage level monitoring and improved collection routes may be achieved using IoT sensors in waste management systems.

### **2.2.2 Precision Agriculture**

The Internet of Things (IoT) is being used by the agriculture sector more and more to improve resource efficiency and crop yields. IoT makes it possible for physical items to communicate and share data with other systems and devices over the internet by integrating sensors, software, and other technologies into the object. The development of networked sensors that track crop health, weather patterns, and soil conditions in real time has been made possible by technology. Farmers are able to make informed decisions about pest management, fertilization, and irrigation because to the data gathered by these sensors. IoT-enabled irrigation systems, for example, may be set up to only water crops when necessary, saving up to 60% of water [4]. In a similar vein, IoT-enabled pest management systems can employ sensors to identify pests in real time and react by applying a precise amount of pesticide [5]. Physical objects may exchange and gather data with minimum human interaction by utilizing low-cost computing, cloud computing platforms, big data analytics, machine learning, and mobile technologies. Digital systems are able to record, monitor, and modify every interaction between linked objects in this hyper connected environment.

### **2.2.3 Health Care Monitoring and Remote Care**

Healthcare is being revolutionized by IoT-enabled wearables and medical sensors that make remote patient monitoring possible. IoT makes it possible for physical items to communicate and share data with other systems and devices over the internet by integrating sensors, software, and other technologies into the object. Healthcare practitioners may now give patients with individualized treatment plans and prompt interventions thanks to the advancement of wearable technology and medical sensors that can monitor patients in real-time. For example, wearables with Internet of Things capabilities may track vital indications in real-time, including blood pressure, blood sugar, and heart rate [6]. Likewise, Internet of Things-enabled medical sensors have the ability to track patients' health in real time and notify medical professionals as needed. [7].

### **2.2.4 Industrial IIOT for Manufacturing**

By gathering and evaluating data from machinery and equipment, the Industrial Internet of Things (IIoT) transforms production processes in the industrial sector [8]. By providing real-time data monitoring, predictive maintenance, process optimization, and data-driven decision-making, this

technology has completely changed the industry. Predictive maintenance is among the IIoT's most important benefits. IIoT can anticipate when repair is necessary by gathering data from machines and other equipment, which decreases downtime and boosts output. A further advantage is instantaneous quality control. IIoT can provide real-time production process monitoring, guaranteeing that goods fulfill quality requirements. Furthermore, by offering real-time data on inventory levels, shipment schedules, and other crucial information, IIoT facilitates effective supply chain management [9]. Many industrial sectors find the IIoT concept very appealing because of its improved production process operational efficiency capabilities, embedded technologies that enable smart item recognition, intelligent automation capabilities, and 24/7 monitoring capabilities. The Internet of Things (IoT) technologies that are currently ensuring efficient task execution in various areas, including industrial, commercial, and social, provide the foundation for the growth of the IIoT phenomena. [8].

### **2.2.5 Energy Management and Smart Grids**

The development of smart grids made possible by the Internet of Things (IoT) is revolutionizing the energy management industry. In order to promote a more sustainable and effective energy ecology, these grids are made to track energy usage, control demand, and make it easier to integrate renewable energy sources [10]. When it comes to integrating increasing amounts of variable renewable energy sources, like solar and wind power, and new loads, like energy storage and electric car charging, smart grids are very helpful in preserving the stability and efficiency of the system. Smart grids enable utilities to optimize operations and cut costs by supplying real-time data on energy consumption and demand [11]. A framework for the Internet of Things that links citizens and governments to smart city solutions includes the smart grid technology. Building a safe, clean future depends on it. It is imperative to view the smart grid as a mission-critical asset, capable of remotely monitoring and controlling various infrastructure components such as street lighting, transmission lines, substations, cogeneration, outage sensors, and early detection of power interruptions caused by extreme weather and earthquakes [10].

### **2.2.6 Environmental Monitoring and Conservation**

Monitoring environmental parameters including water and air quality, tracking wildlife, and patterns of deforestation are monitored by linked sensors and equipment. The data collected by



these sensors can help with conservation efforts and early environmental danger detection thanks to the Internet of Things (IoT) technology [12]. These sensors can now gather data, which may be utilized to support conservation efforts and early environmental danger identification thanks to Internet of Things (IoT) technology [13]. Our air, soil, and water may be cleaned and protected by governments and businesses with the help of IoT-based environmental monitoring systems, which can identify toxic materials, chemical spills, dangerous pollutants, and more. With the use of IoT environmental monitoring sensors and connection, we can effectively and efficiently monitor and maintain a healthy environment. We can also lower our carbon footprint by reducing energy consumption and utilizing tools for analysis and early identification of pollutants.

Rapid detection, reporting, data insights, and remediation are made possible by the deployment of sensors, IoT devices, remote connectivity, and edge computing. These monitoring systems can be configured to recognize anomalies or particular circumstances, after which automatic procedures and email or text warnings are sent out. These might be anything from opening support requests to turning off systems to prevent a catastrophe [12].

### **2.2.7 Retail and Customer Experience Enhancement**

The retail business has experienced a change because to the Internet of Things (IoT) technology, which has made it possible to create smart stores that provide real-time customer insights, inventory management, and tailored marketing [14]. Smart stores are equipped with a range of IoT devices such as beacon technology, RFID tags, and smart shelves that streamline operations and provide customers with seamless shopping experiences [15]. Personalized marketing is one of the main benefits of IoT in retail. By collecting data from IoT devices, retailers can offer personalized promotions and advertisements to customers based on their preferences and shopping history. Another benefit is inventory management. IoT devices can monitor inventory levels in real-time, ensuring that products are always in stock. Additionally, IoT enables real-time customer insights by providing data on customer behavior and preferences, allowing retailers to optimize their operations and improve customer satisfaction [14].

### **2.2.8 Connected Transportation and Logistics**

Real-time fleet monitoring is made possible by IoT-based tracking systems, which give precise data on the position, speed, and routes driven by each vehicle in the fleet. Utilizing this data can

improve supply chain efficiency, cut fuel costs, and optimize routes [16]. Fleet management is one of the main benefits of IoT in logistics and transportation. Fleet management systems (FMSs) powered by the Internet of Things (IoT) give businesses access to real-time vehicle data, empowering them to make better decisions [17]. FMSs provide the ability to track the condition of vehicles and maintenance plans, which helps save costly emergency repairs and downtime. Furthermore, by tracking vehicle speed and engine performance, FMSs may spot inefficiencies that result in higher fuel usage. By using this data, vehicle maintenance or driver education programs can increase fuel economy [16]. In the logistics and transportation sector, IoT technology has also made smart inventory management possible. Multiple warehouses and distribution centers can employ IoT sensors to track the amount of assets in each location and keep an eye on the present state of affairs. This lowers the possibility of human error and gives quick knowledge of how many items are at each facility [17].

### **2.2.9 Smart Home Automation**

The development of smart houses, which provide more convenience and energy efficiency, has revolutionized the residential market thanks to Internet of Things (IoT) technology. A seamless and networked living environment is created by the assortment of Internet of Things (IoT) gadgets that smart homes are outfitted with, including voice-activated assistants, security cameras, and smart thermostats. Energy saving is among the Internet of Things' most important benefits for smart homes. With the ability to recognize customer preferences, smart thermostats may modify the temperature in response, saving energy and money on utility bills. Convenience is also another advantage. Lights, thermostats, and entertainment systems are just a few of the home electronics that may be managed by voice-activated assistants like Google Home or Alexa from Amazon. [18]. Additionally, IoT-enabled security cameras provide real-time monitoring of the home, enhancing safety and security [19].

### **2.2.10 Water Management and Conservation**

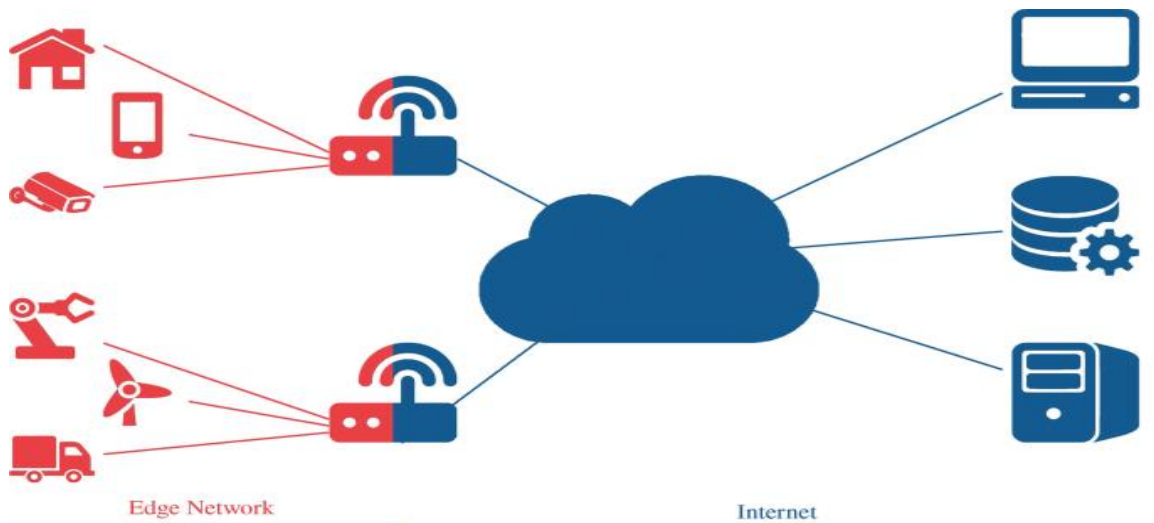
The development of smart water systems that monitor water usage, identify leaks, and control water distribution systems has been made possible by the Internet of Things (IoT) technology, which has completely changed the water management industry. These applications are essential for preserving water supplies and guaranteeing effective management of the water supply. Water

conservation is one of the main benefits of IoT in water management. IoT-based solutions can assist in minimizing water waste and guaranteeing that water is used effectively by tracking water usage and identifying leaks. Another benefit is efficient water supply management. IoT-based systems can monitor water distribution systems in real-time, ensuring that water is distributed efficiently and effectively [20].

The convergence of IoT technologies with various industries showcases its potential to reshape how we live, work, and interact with our environment. Creating innovative applications will lead to a more interconnected and data-driven world as IoT evolves.

### 2.3 IoT Networks

IoT networks are made up of the communication technologies that IoT devices utilize to exchange data with one another. Cellular networks such as LTE-M and NB-IoT, Wi-Fi, Bluetooth Low Energy, Sigfox, LoraWAN, Zigbee, RFID, and Ethernet are a few examples of IoT networks. Usually, wireless communication technologies are used to connect Internet of Things devices. Rarely, you might come across IoT devices that are Ethernet-connected. On the other hand, Internet of Things protocols like as MQTT, CoAP, AMQP, and so forth function at the application layer.



**Figure 2.2:** IoT Network

The network layer of the internet architecture is covered by these communication technologies. The first decision to be made when creating an IoT application is selecting a network. The use case

often determines which IoT network is best. IoT networks can be divided into the following four major classes: cellular networks, such as NB-IoT, LTE-M, etc. LAN/PAN, similar to WiFi, Bluetooth, etc. LPWAN, such as Sigfox, LoRaWAN, etc. Mesh protocols include Z-wave, RFID, and ZigBee. IoT network classification helps you narrow down your options for a certain use case. The necessary coverage area, cost, device environment, density of IoT devices, power consumption, kind of machine-to-machine communication, needed network bandwidth, security, etc., may all play a role in which IoT network is chosen. Our research will concentrate on networks using the MQTT Protocol over 802.11 (Wi-Fi).

### 2.4 IoT Messaging Protocols

Depending on the needs and protocol appropriateness, a large range of communications protocols can be utilized while dealing with Internet of Things networks. An example of the IoT network protocol stack may be found below.

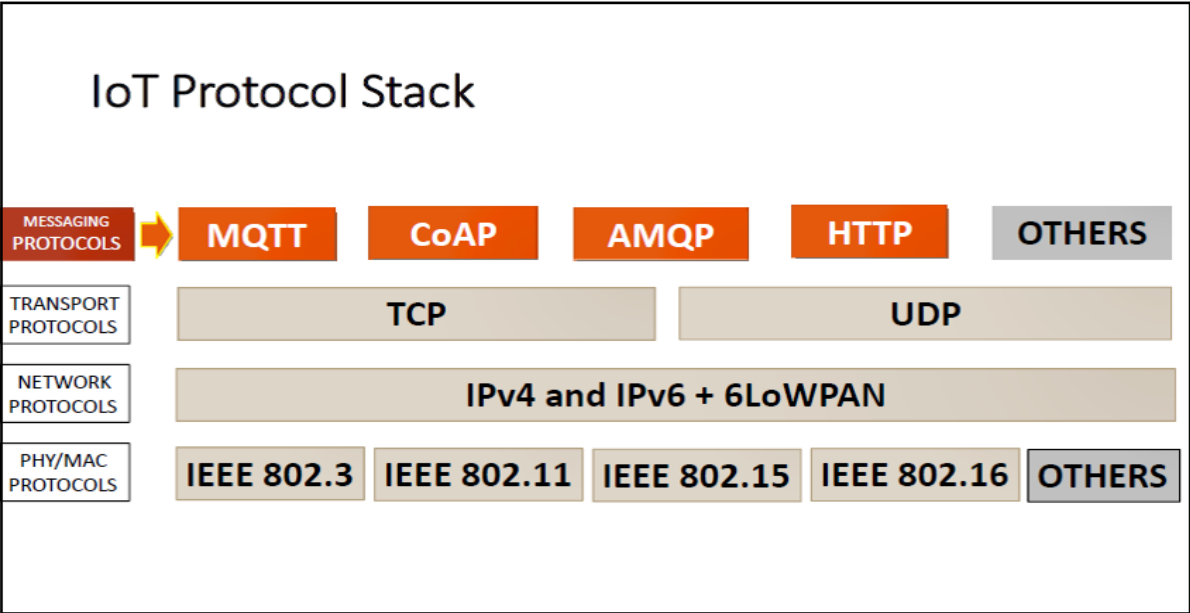


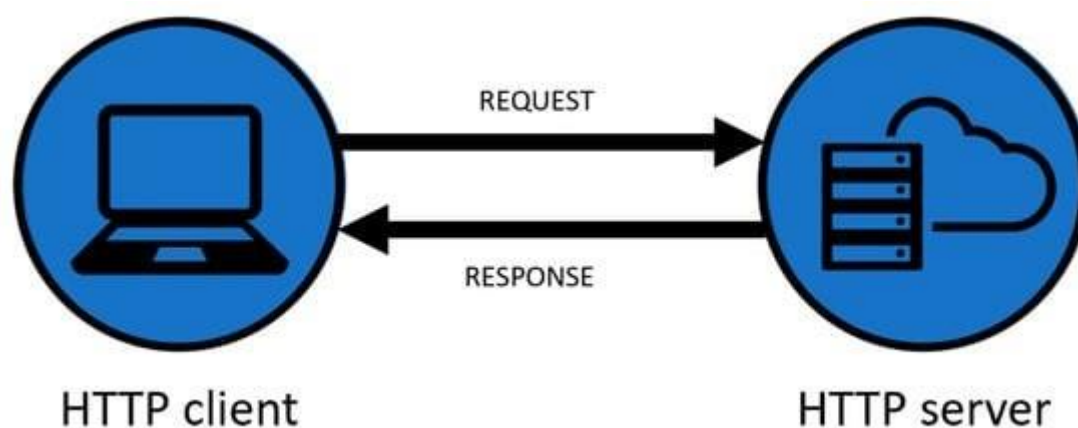
Figure 2.3: IoT Messaging Protocol Stack

#### 2.4.1 HTTP

The web message protocol known as "HTTP" was first created by Tim Berners-Lee and subsequently co-developed by the IETF and W3C. It was initially released in 1997 as a standard protocol and is currently widely used. Similar to CoAP, HTTP employs Universal Resource

Identifiers (URI) in place of topic to facilitate the RESTful web's "request/response architecture." Data is sent by the server using URIs, and data is received by the client using unique URIs. The size of the message payload and header in a text-based protocol like HTTP varies depending on the web server or computer language used. Through the use of TLS/SSL for security and TCP as the default transport protocol, HTTP makes connection-oriented client-server communication possible. While HTTP does not define QoS directly, it does offer a number of capabilities that need further support, like request pipelines, chunked transfer encoding, and persistent connections. [21].

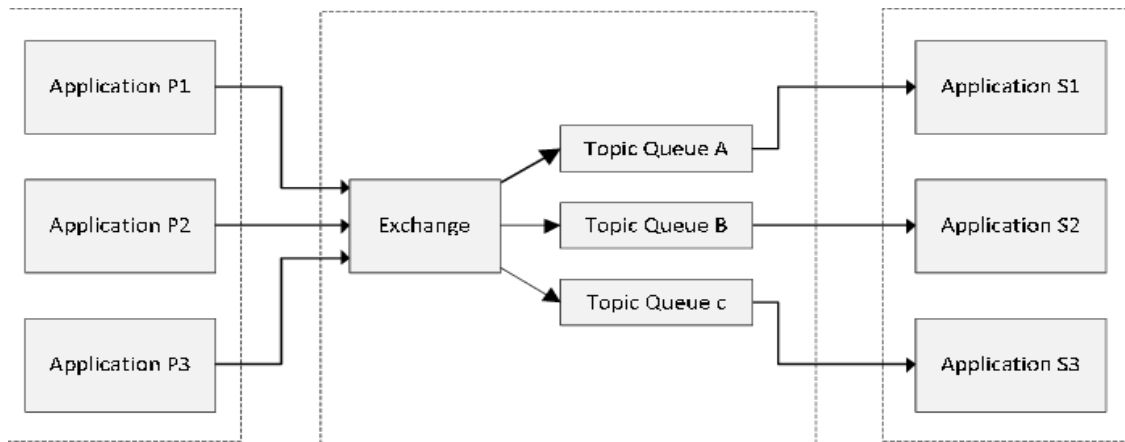
The request response architecture of HTTP is illustrated below.



**Figure 2.4:** HTTP Architecture

## 2.4.2 AMQP

At JPMorgan Chase in London, UK, in 2003, 'John O'Hara' created the Advanced Message Queuing Protocol, or "AMQP." A corporate messaging protocol known as AMQP provides a host of features, including transactions, topic-based publish-and-subscribe messaging, dependable queuing, and flexible routing. It is compatible with publish/subscribe and request/response architectures. Either the publisher or the consumer initiates an exchange with a particular name in the AMQP communication system, after which the name is broadcast. AMQP can exchange messages in a number of methods, including topic-based, fan-out, direct, and based on headers. TCP is the default transport protocol for AMQP, a binary protocol that uses TLS/SSL and SASL for security. One of AMQP's primary characteristics is reliability. It provides two initial QoS levels for message delivery: Unsettled Format (not reliable) and Settled Format (reliable) [23]. The architecture of AMQP is illustrated below [29].



**Figure 2.5: AMQP Architecture**

### 2.4.3 CoAP

A new web application transfer protocol called CoAP aims to operate on a very small amount of resources. CoAP offers a more cost-effective design while yet offering the same core set of functions as "compressed HTTP" [24]. The main method by which CoAP reduces complexity is by using UDP in place of TCP and defining a very basic message layer for the retransmission of dropped packets. In "CoAP" for UDP packets, a set of options (each with a one-byte header that can be stretched to two bytes for larger option values) are placed after a 4-byte binary header. A typical request can have a total header size of 10 to 20 bytes thanks to its condensed yet simple-to-parse encoding. Differential encoding of optional types spares straightforward implementations while enabling the requisite expansion in the future [24]. The action requested by the client is identified by the method, the URI, and in some cases metadata about the request.

There are four different methods defined in the CoAP standard:

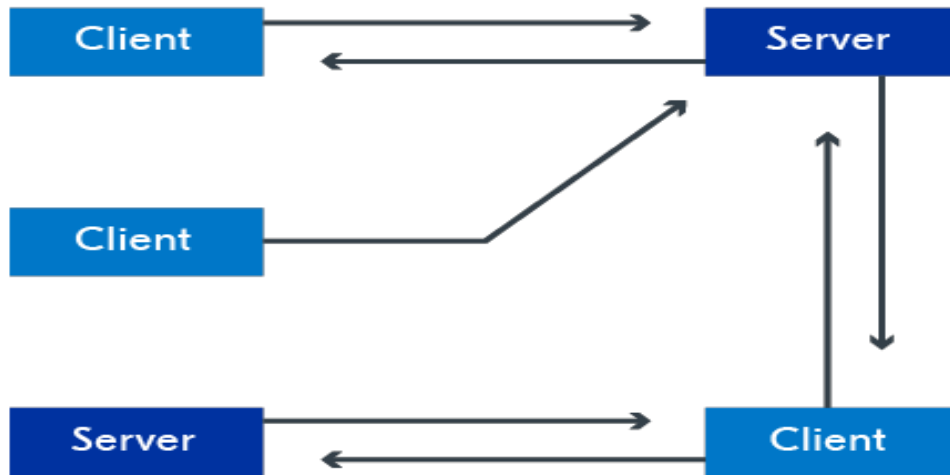
GET - to retrieve resource representation.

POST - to transfer information representation.

PUT - to update resources on the server.

DELETE - to delete resource [25].

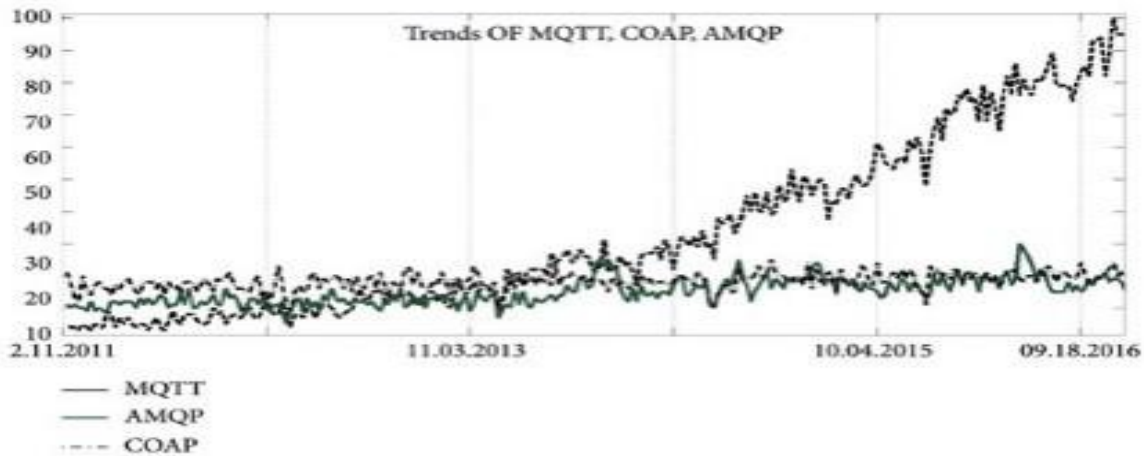
Below is the illustration describing the overall architecture of CoAP from [30].



**Figure 2.6:** CoAP Architecture

#### 2.4.4 MQTT

A lightweight messaging protocol called MQTT (MQ Telemetry Transport) satisfies the more sophisticated communication requirements of today. The protocol is essential to the Internet of Things and is used for machine-to-machine communication. MQTT is able to distribute telemetry data via a publish/subscribe communication pattern in situations where the network bandwidth is low, the network latency is significant, and the devices have limited processing and memory capacity. It allows Internet of Things (IoT) devices to publish or send data on a topic head to a server (such as a MQTT broker), which then distributes the data to clients who have already subscribed to that topic [26]. Many different industries today employ MQTT, including manufacturing, oil and gas, telecommunications, automotive, and agriculture. created in 1999 by "Arlen Nipper" and "Andy Stanford-Clark (IBM)" to connect oil pipeline telemetry systems via satellite. Despite having begun as a proprietary protocol, it was made available without a royalty in 2010 and eventually became an OASIS standard in 2014. MQTT being one of the primary protocols for IOT (internet of things) installations is quickly emerging. The graph research below, created by [28], demonstrates unequivocally how the industry has adopted MQTT and how quickly it is growing.



**Figure 2.7:** Trends of IoT Messaging Protocols

MQTT comes in two flavors and multiple iterations.

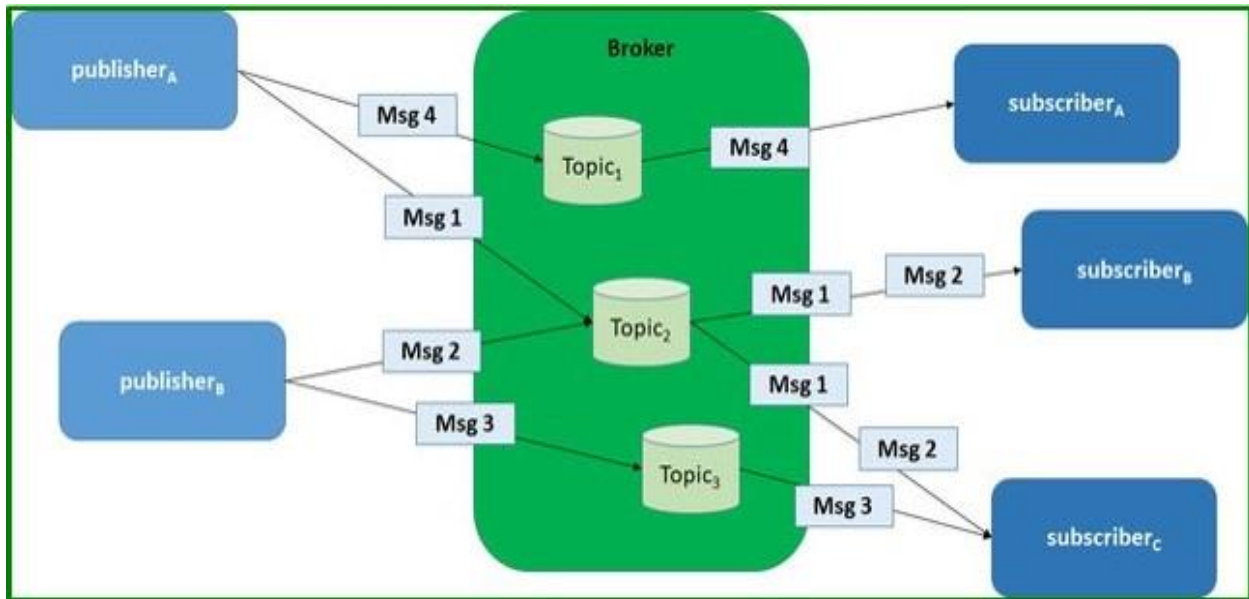
MQTT versions 3.1.0, 3.1.1, v5, and MQTT-SN

The original MQTT was created for TCP/IP networks, and it has been in use for many years. It was designed in 1999. The most often used version of MQTT is 3.1.1. Two primary types can be distinguished between IoT messaging protocols based on the interaction paradigm: Protocols for publish-subscribe Response-oriented protocols The research study graph below illustrates the significant trend toward MQTT over the period, which is increasing daily. [27].

### **MQTT Architecture**

MQTT connects IoT devices with a publish/subscribe architecture. In contrast to the request/response paradigm used by HTTP, MQTT functions event-driven, enabling the pushing of messages to clients. This architectural approach breaks dependencies between data producers and consumers, enabling extremely scalable applications. MQTT Clients and MQTT Broker are two essential parts of setting up a MQTT connection for publishing and subscribing to messages. [27].





**Figure 2.8:** MQTT Architecture

**MQTT Broker:**

By acting as a middleman between senders and recipients, a MQTT broker delivers messages to the right people. The MQTT broker employs a subscriber list that it keeps up to date to send messages to the appropriate clients.

**MQTT Clients:**

Other clients subscribe to particular topics to receive messages published by MQTT clients to the broker. Every MQTT message has a topic, and subscribers can subscribe to topics that interest them. [27].

## **CHAPTER 3: CHALLENGES OF SECURING IOT AND ROLE OF SECURITY PROTOCOLS**

### **3.1 Challenges in Securing IoT**

Concern over IoT security issues is rising in the linked world of today. Securing these devices is becoming a major concern as the number of IoT devices keeps growing. For Internet of Things devices to function properly, hardware, software, and connectivity security are essential [31]. However, IoT expansion may be hindered by security concerns. Outdated hardware and software can pose security risks, and using weak or default credentials can leave devices vulnerable to attacks. The prevalence of malware and ransomware attacks adds to security challenges. Additionally, it can be difficult to detect whether a device is affected, leading to data protection and security issues. Ensuring the security of Enterprise IoT involves anticipating and averting IoT attacks. Companies need to be proactive in spotting possible weak points and reducing risks. Businesses must take proactive steps to identify potential vulnerabilities and mitigate risks. By understanding the techniques used by hackers and staying ahead of emerging threats, businesses can take preventative measures to protect their IoT networks [31].

Weak authentication mechanisms, inadequate encryption, and vulnerable firmware attacks are just some of the security vulnerabilities that can affect the integrity of IoT devices [32]. Compromising IoT devices can lead to serious consequences, including unauthorized access to sensitive data, violate privacy and even manipulate critical systems [32]. In short, securing IoT devices is a challenge that businesses must address. By implementing strong security measures, businesses can mitigate these challenges and protect their IoT networks.

### **3.2 Role of Security Protocols**

To protect IoT devices from attacks and weaknesses, security protocols are used. Weak firmware, insufficient encryption, and weak authentication methods are a few of the flaws that security protocols address [33]. Security measures that guard against breaches of privacy, illegal access to private information, and compromise of vital systems [34]. Three principles serve as the foundation for the CIA triad, which is used to secure information security systems or any type of system: availability, confidentiality, and integrity. It is crucial to remember, too, that the security

standards directly address the first two—confidentiality and integrity—and indirectly address the third—availability, of the CIA trinity [37].



**Figure 3.1: CIA Triad**

### **3.2.1 Confidentiality**

The term confidentiality describes measures used to keep information private or secret. Controlling information access is necessary to avoid unintended or intentional illegal data sharing in order to achieve this. Ensuring that assets vital to your business are inaccessible to anyone lacking the necessary authorization is a crucial aspect of security maintenance. Encrypting data as it is being transmitted is another way to guarantee data security.

On the other hand, an efficient system also makes sure that people who require access have the appropriate rights. Security procedures make ensuring that data can only be accessed or modified by authorized people and processes. Security methods shield sensitive data from unauthorized access by encrypting it while it's in transit. [38].

### **3.2.2 Integrity**

Integrity is making sure that your data is reliable and hasn't been altered; data integrity is only preserved when it is reliable, accurate, and legitimate. Integrity violations are frequently committed on purpose. An attacker could change file configurations to permit unauthorized access, get around a "IDS" intrusion detection system, or alter system logs to hide the attack. Inadvertent violations of integrity are also possible. It's possible for someone to carelessly enter the incorrect code or do something else. Furthermore, if a business has insufficient security policies, measures,

and processes in place, integrity may be compromised without any employees being held responsible. Security protocols ensure that data is maintained in a correct state and nobody can improperly modify it, either accidentally or maliciously. By using digital signatures and message authentication codes (MACs), security protocols ensure that data is not tampered with during transmission [38].

### **3.2.3 Availability**

Data can often be futile if it cannot be accessed by authorized individuals, even if it is kept private and its integrity is maintained. This implies that applications, networks, and systems must function as intended and when required. Furthermore, getting access to the data won't take too long, and anyone who need it will be able to do it whenever they need it. Security protocols ensure that authorized users can access data whenever they need to do so. By preventing denial-of-service (DoS) attacks and other disruptions, security protocols ensure that IoT devices are available when needed [38].

## **3.3 Standard Security Protocols and OSI Model Layer**

To secure networks and communication with different goals and setups, a number of common security protocols are employed. Every one of them has advantages and disadvantages. The various OSI model levels are where these protocols function. It is essential to comprehend the OSI model and its layers before we can talk about these protocols.

An industry standard for how apps interact with one another via networks is called Open Systems Interconnection, or OSI. It demonstrates how every communication layer—from the physical wire to the apps that try to connect with other devices over the network—is layered on top of the others. The OSI model aids security teams in comprehending which network levels they must protect, potential attack points, and ways to stop and lessen certain security risks [39].

The following levels are part of the OSI Model:

**Layer 1**—Physical Layer— The actual wire or wireless connection between network nodes.

**Layer 2** —Data Link Layer— establishes and breaks connections; it divides packets into frames and sends them from one place to another.

**Layer 3** — Network layer - separates segments into packets for networking, reassembles them after receiving them, and uses the best paths to send packets over the actual network.

**Layer 4** — The Transport Layer — is in charge of reassembling the segments and transforming them into data that the session layer can use.

**Layer 5** — Session layer - establishes sessions, or channels of communication, between devices while data is being transferred, keep sessions open and close them when you're done.

**Layer 6** — Presentation layer - ensure that the data is received correctly, prepares it for the application layer by figuring out how to compress, encode, and transmit it across two devices.

**Layer 7** — Application layer – Email programs and web browsers are examples of end-user software that utilize this layer. Use protocols like HTTP, FTP, and DNS to send and receive useful information to end users [39].

### **3.3.1 IPsec – Layer 3**

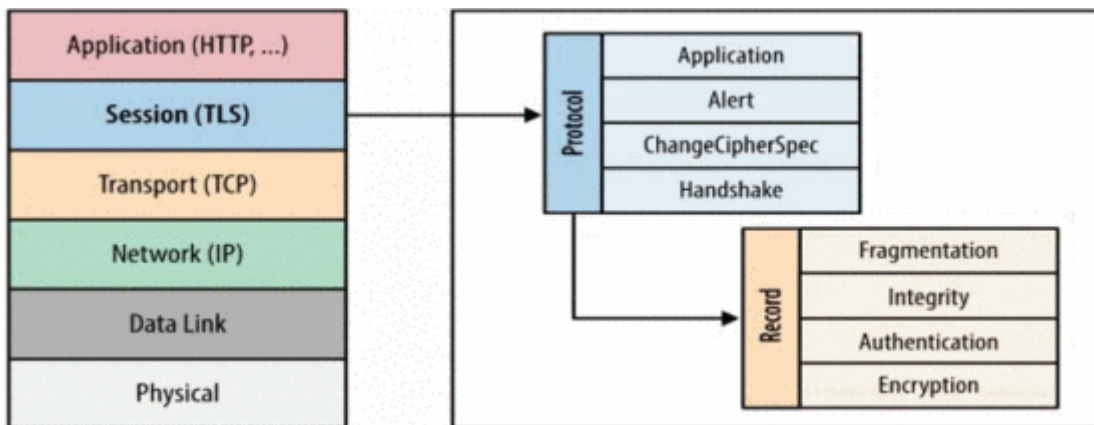
A collection of methods and protocols known as IPsec are used to protect data transfer over open networks like the Internet. In the 1990s, the Internet Engineering Task Force (IETF) invented IPsec protocols. For IP layer security, they authenticate and encrypt network packets. IPsec was once comprised of the AH and ESP protocols. While the Authentication Header (AH) safeguards data integrity and offers replay protection, the Encapsulated Security Payload (ESP) encrypts data and performs authentication. The Internet Key Exchange (IKE) protocol, which offers shared keys to create secure associations (SAs), has since been added to the suite. They enable encryption and decryption via routers or firewalls. IPsec offers tunnels to secure data communications, protecting VPNs and sensitive data. It can facilitate authentication without encryption and encrypt data at the application layer. [39].

### **3.3.2 Transport Layer Security – Layer 5**

Since 1999, the TLS protocol has been in operation. Its implementation has changed a few times since then. Released in April 2017, TLS 1.3 was the final version of these. It is crucial to the current Internet protocol architecture [43].

Originally introduced as the Secure Sockets Layer (SSL) protocol, which guarantees message integrity, authenticates the provenance of data, and encrypts data. For client and server authentication, it makes use of X509 certificates. By exchanging secure session parameters,

creating session keys, and shaking hands, SSL authenticates the server. Then, by verifying the data's provenance, it can send it securely. The encryption algorithm used during the handshake between the client and server is the same as the one used during the SSL session. The server is capable of supporting encryption with Triple DES and AES algorithms. In order for the client to authenticate the server through X509, the server certificate for SSL is necessary. X509 client certificates can also be used by SSL for authentication. A reliable certificate authority from the server's keyring must sign these certificates. The IETF established Transport Layer Security (TLS), an SSL-based protocol (SSL is not) [39]. The SSL protocol was developed by Netscape to be used in e-commerce and to secure financial transactions. To protect customer credit card information and personal data, this is a must. The SSL protocol was put into place at the OSI model's application layer in order to fulfil this. Later, when SSL was regarded as standard, the standards organization IETF dubbed it TLS, or transport layer security. The OSI model's placement of TLS and its variations, such as WTLS and DTLS, is depicted in the image below. [43].



**Figure 3.2:** TLS Protocol in OSI Model

TLS is made up of two primary parts.: -

**Handshake Protocol;** Establish shared cryptographic hardware, agree on cryptographic modes and settings, and authenticate the parties involved in communication. Because of the tamper-resistant nature of the handshake protocol, peers cannot be forced to negotiate parameters in a different way than they would if the connection was not under assault.

**The Record Protocol;** employs the handshake protocol's settings to safeguard communication between peers. Traffic is separated into a number of records by the record protocol, each of which is safeguarded separately [44].

In the later chapter, the comparison of TLS and its variants is covered in detail with regard to the protocols.

### **3.3.3 Datagram transport layer security – Layer 5**

Based on TLS, DTLS is a datagram communication security mechanism. This does not ensure that communications arrive in a timely manner or that they are delivered. The benefits of the datagram protocol are also present in DTLS, such as lower latency and cheaper costs [39]. DTLS was built with the fundamental design tenet of "TLS over datagram transport." Datagram transit does not necessitate or offer dependable or systematic data delivery. This feature is preserved for application data by the DTLS protocol. Because the data transported by these applications is latency-sensitive, packet transmission is used for communication in applications like online gaming, Internet telephony, and multimedia streaming. When DTLS is used to secure connections, these applications behave in the same way since DTLS does not make up for lost or rearranged data transmission. Observe that phones utilize DTLS to establish keys and Transfer Protocols, whereas streaming and low-latency gaming use it to safeguard data (like the WebRTC data channel). SRTP, or secure real-time downloading, protects data [39].

### **3.3.4 Wireless transport layer security – Layer 5**

A security layer for the Wireless Application Protocol (WAP) that is unique to WAP-using applications is called Wireless Transport Layer Security (WTLS). Its foundation is Transport Layer Security (TLS) v1.0, the Internet's security layer that replaces Secure Sockets Layer (SSL) 3.1. [41]. Initially designed for mobile devices, it is optimized for low-bandwidth mobile devices, uses compressed certificate and data formats, and supports packet-based transport protocols. WTLS was depreciated due to some defects and threats. Some of the reasons were compatibility issues, performance trade-offs, and design flaws. Additionally, the Protocol is susceptible to man-in-the-middle, replay, and truncation attacks. [40].

### **3.3.5 Simple Network Management -Layer 7**

An application-layer network device management and monitoring protocol is called SNMP. On both local and wide area networks, it can safeguard devices. Through the network management system, SNMP offers a common language for servers and routers to communicate with one another. SNMP is a built-in component of the Internet protocol suite that the IETF has established. The management information bases (MIBs), agents, and managers comprise the SNMP architecture. The MIB is the database, the manager is the client, and the agent is the server. Using the MIB, the SNMP agent replies to the manager's request. SNMP is widely available, but in order to deploy the protocol, administrators need to change the default settings so that agents and network management systems can communicate. Three crucial security features were added to the SNMP protocol in 2004 with the release of SNMPv3: packet encryption to stop eavesdropping, integrity checking to make sure packets aren't altered during transmission, and authentication to confirm that the communication is coming from a reputable source. [39].

### **3.3.6 Kerberos Protocol-Layer 7**

A service request authentication mechanism called Kerberos is used on untrusted networks, including the public Internet. Supporting Linux, Mac, and Windows operating systems, it provides integrated authentication for requests between trusted servers. Windows employs the Kerberos authentication protocol by default, and it is an essential part of services like Active Directory (AD). It is used by broadband service providers to verify cable modems and set-top boxes that are gaining access to their networks. When utilizing Kerberos, users, services, and systems should only have faith in KDC. To enable nodes to authenticate one another, KDC offers ticketing and authentication. Packets are protected during transmission and authenticated by Kerberos using shared secret cryptography. [39].



## CHAPTER 4: LITERATURE REVIEW

We had to collect and conclude various aspects of the IoT architecture, Security and energy consumption of the proposed scheme. Therefore, we divided our literature into several parts: We started with intension to find the best suitable application protocol for the IoT network considering our parameters i.e. energy consumption, reliability, scalability security. Then, we moved forward on to make sure we use the best standard security protocol or its variant in the business in our architecture. Later on, we focused on the reduction of the energy consumption through different approaches. the summary of the literature review of the research papers is discussed under:

### 4.1 Analysis of IoT Applications

The author of article [45] provided a much-needed comparison of the popular and developing messaging protocols for Internet of Things systems, namely "MQTT," "CoAP," "AMQP," and "HTTP," based on a number of indicators to show their characteristics in a comparative way. Important findings from a comparison analysis of these are shown in table 4-1 below.

**Table 4-1:** Messaging Protocol Comparative analysis for IoT Systems[45]

| Criteria           | MQTT  | CoAP  | AMQP                                  | HTTP  |
|--------------------|---|---|---------------------------------------|---|
| Year               | 1999  | 2010  | 2003                                  | 1997  |
| Architecture       | Client/Broker                                   | Client/Server or Client/Broker                                    | Client/Broker or Client/Server        | Client/Server   |
| Abstraction        | Publish/Subscribe                               | Request/Response or Publish/Subscribe                             | Publish/Subscribe or Request/Response | Request/Response  |
| Header Size        | 2 Byte  | 4 Byte  | 8 Byte                                | Undefined   |
| Message Size       | Small and Undefined (up to 256 MB maximum size) | Small and Undefined (normally small to fit in single IP datagram) | Negotiable and Undefined              | Large and Undefined (depends on the web server or the programming technology) |
| Transport Protocol | TCP (MQTT-SN can use UDP)                       | UDP, SCTP   | TCP, SCTP                             | TCP   |
| Security           | TLS/SSL   | “DTLS, IPsec  | TLS/SSL, IPsec, SASL                  | TLS/SSL   |
| Licensing Model    | Open Source                                     | Open Source   | Open Source                           | Free  |
| Encoding Format    | Binary  | Binary  | Binary                                | Text  |

|                        |   |  |  |                              |
|------------------------|---|--|--|------------------------------|
| Organizational Support | IBM, Facebook, Eurotech, Cisco, Red Hat, Software AG, Tibco, ITSO, M2Mi, Amazon Web Services, InduSoft, Fiorano | Large Web Community Support, Cisco, Contiki, Erika, IoTivity | Microsoft , JP Morgan, Bank of America, Barclays, Goldman Sachs, Credit Suisse | Global Web Protocol Standard |
| Standards              | OASIS, Eclipse Foundations  | IETF, Eclipse Foundation                                     | OASIS, ISO/IEC   | IETF and W3C                 |

One can decide which protocol can be best suited for the use case with the above comparison table. However, there are some more parameters which should be kept in mind to take well aware decision.

The author categorized further analysis in the segments where each protocol has been analyzed in the respective category which is discussed below:

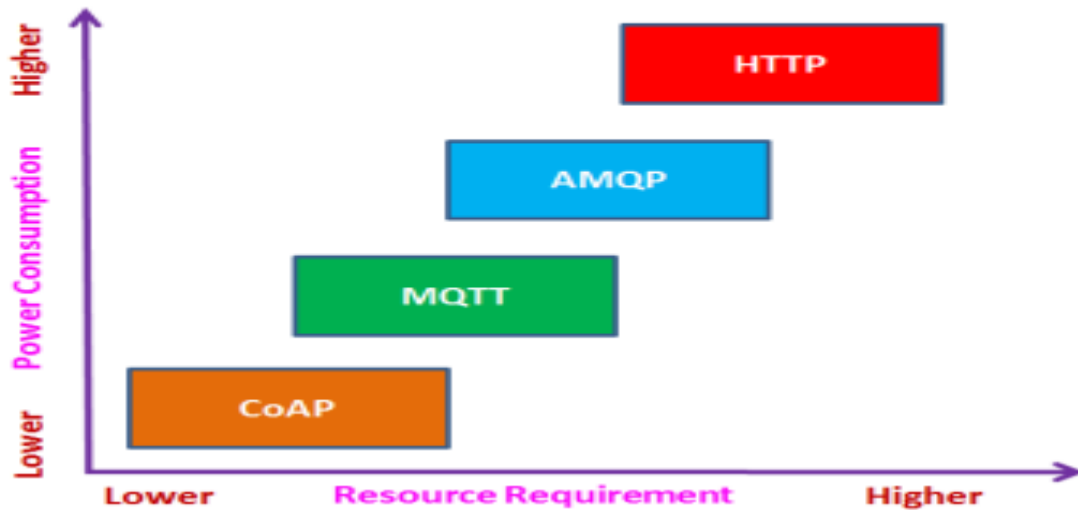
**Message Size vs. Message Overhead**

The biggest message size and overhead among these protocols is found in HTTP, which is followed by the other protocols, with CoAP having the shortest overhead and message size. Since MQTT, AMQP, and HTTP utilize TCP, they cause all overheads associated with the establishment and termination of TCP connections. In contrast, "MQTT" has the shortest header size—two bytes per message. However, the need for a TCP connection increases gross cost and, consequently, message size. Because CoAP employs UDP, which functions on a fire-and-forget basis, there are no connection overheads. As a result, message size and total overhead are greatly reduced. Another lightweight binary protocol is "AMQP." However, fault tolerance, security, and dependability are encouraged, which enhances interoperability and provisioning.

**Power Consumption vs. Resource Requirement**

Graph 4.1 presents a comparison of messaging protocols with respect to their typical power consumption and resource needs. According to the graph, CoAP has the lowest resource and power requirements while HTTP has the most. Both CoAP and MQTT can be implemented on an 8-bit controller with hundreds of bytes of memory, and they are both intended for devices with limited

resources and low bandwidth. In comparable conditions, such as unreliable scenarios (MQTT QoS 0 vs. CoAP NON) and reliable scenarios (MQTT QoS 1 or 2 vs. CoAP CON), provided no packet losses occur, many experimental tests have indicated that CoAP uses slightly less power and resources. Because AMQP needs to do certain activities for provisioning and reliability, it needs greater power and resources. Lastly, for the same task, HTTP demands more processing power and resources than any other protocol. [45].



**Figure 4.1:** Power Consumption Vs Resources Requirement

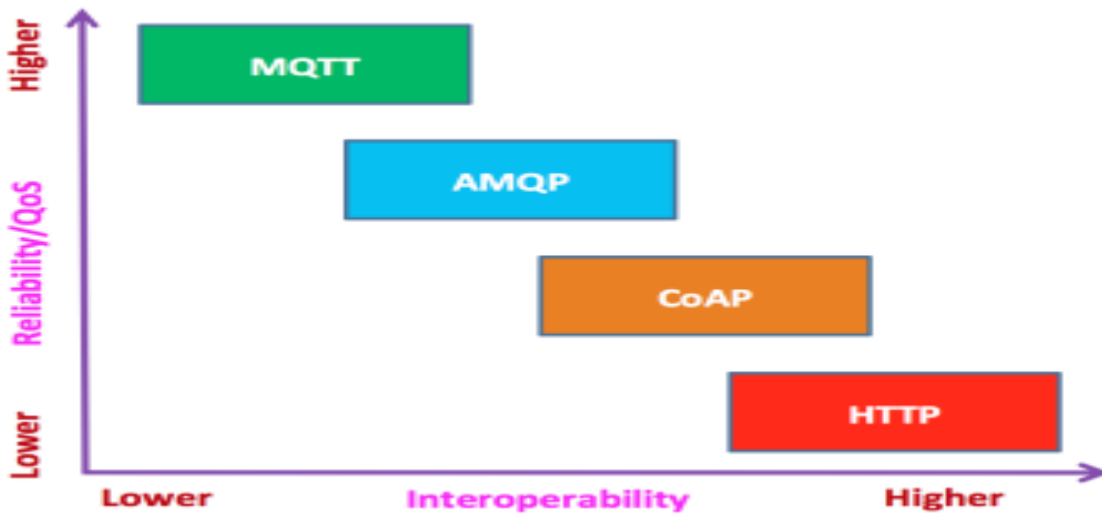
### **Bandwidth vs. Latency**

According to the study, CoAP has the lowest latency and bandwidth of all the protocols, while HTTP has the highest. In MQTT, AMQP, and HTTP, TCP is utilized to calculate the latency and bandwidth needed. However, because TCP starts slowly to prevent network congestion, it does not reduce latency and does not use all of the available network capacity during the first few roundtrips of a connection. On the other hand, CoAP lowers the network load response times by requiring only two UDP datagrams for a UDP transaction—one in each direction. For the same payload transferred under the same network conditions (MQTT QoS 1 or 2 vs. CoAP CON), several experimental investigations have demonstrated that MQTT uses more bandwidth than CoAP. Furthermore, because QoS 2 uses a four-way handshake, MQTT's bandwidth use is roughly twice that of CoAP when comparing "QoS 2" with "CoAP CON." The additional services offered

by AMQP require somewhat higher latency and bandwidth. HTTP requires a lot more bandwidth and delay. [45].

### Reliability/QoS vs. Interoperability

The QoS support of the MQTT, AMQP, and CoAP protocols varies, as seen by the graph in figure 4.2. Three QoS levels are defined by MQTT: 0 for at most once, 1 for at least once, and 2 for exactly once. There are two QoS levels defined by AMQP: Unsettle Format (which is akin to MQTT QoS 1) and Settle Format (which is akin to MQTT QoS 0). By specifying a retransmission mechanism and offering a resource discovery mechanism with resource description, CoAP makes up for the unreliability of the UDP protocol. CoAP allows the use of non-confirmable messages (NON) and confirmable messages (CON), which is very similar to MQTT QoS 0 and QoS 1, even though it does not explicitly support QoS. Because all that's required to facilitate message exchanges is an HTTP stack, HTTP-based RESTful clients and servers are the most interoperable.



**Figure 4.2:** Reliability Vs Interoperability

In terms of latency and stability, the MQTT protocol has been compared with SeedLink in another research publication [46]. Studies have shown that the MQTT protocol offers benefits in terms of stability and latency. The suggested approach might detect occurrences around 1.3 seconds earlier, leading to a total warning time of about 1.6 seconds less, according to the results of P-phase sampling and conventional EEWs. This guarantees that MQTT will be the best option in situations where energy usage, latency, and reliability are important considerations.

## **Security vs. Provisioning**

While "MQTT" is primarily a message protocol and offers the lowest level of security and other services, "AMQP" offers the highest level of security and additional services. CoAP employs IPsec and DTLS for encryption, integrity, and authentication. The two authentication methods that "HTTP" offers are "HTTP Digest" and "HTTP Basic." The strongest security is offered by "AMQP," which uses several TLS negotiation strategies. There are no further services provided by "MQTT," such as message labeling. Several extensions for increased services are available in "CoAP," based on the needs of the IoT system. A number of services are provided by the "HTTP" full web standard, including header compression, server push, stream dependencies and priority, and "multiplexing" and concurrency. "AMQP" is the go-to option for companies due to its extensive array of messaging-related services, including dependable Improvements in Transport Layer Security.

Earlier, we have discussed the overview of transport layer security protocol, which covered the basic architecture of the protocol. We will move forward to discuss the versions of the protocol with vulnerabilities and issues in those protocols and finally discuss how latest version of the protocol addresses them.

### **4.1.1 Previous Versions of TLS Protocol**

SSL 3.0 and TLS 1.0, which was first published as RFC 2246 and launched in 1999, are quite comparable. But TLS 1.0 has a way for a TLS implementation to force the connection down to SSL 3.0, which reduces security.

The second version of TLS was published in 2006 and is known as TLS 1.1. The following are the main distinctions between TLS 1.1 and TLS 1.0:

To defend against CBC assaults, an explicit Initialization Vector (IV) is used in place of the implicit Initialization Vector (IV).

To prevent CBC attacks, padding error handling is adjusted to utilize the "bad\_record\_mac" alert instead of the "decryption\_failed" signal, and the protocol settings are defined in the Internet Assigned Numbers Authority (IANA) registry.

Now, a session can continue even after it has ended prematurely. [36].

Compared to TLS 1.1, TLS 1.2 is the most popular version of TLS and has made various security enhancements. TLS 1.2 differs significantly from its predecessor in the following ways:

The pseudorandom function (PRF) now uses cipher-suite-specified PRFs in place of the MD5/SHA-1 combination.

The digitally signed element now uses a single hash instead of the MD5/SHA-1 combination. A field that clearly states the hash algorithm that was utilized is now present in signed elements.

This allows for more efficient specification of the hash and signature algorithms that the client and server will accept.

Additional data modes with support for authorized encryption have been added.

The cipher suites AES and HMAC-SHA256 have been added. [36].

#### **4.1.2 Improvements Through TLS 1.3**

The authors of paper [47] thoroughly analyzed the latest version of TLS 1.3 with previous version and concluded results were in favor of TLS 1.3. The authors have divided their analysis in two sections which are summarized below:

##### **Security Improvements**

TLS 1.2 is susceptible to man-in-the-middle and downgrade attacks. For instance, POODLE exploits the CBC-mode padding vulnerability when falling back to SSL 3.0. To address this issue, TLS 1.3 introduces a downgrade protection mechanism. When clients negotiate with an older TLS version (or SSL 3.0) server, the TLS 1.3 server must include one of two predefined values (DOWNGRD01 or DOWNGRD00) in server random as a downgrade signal. This mechanism is similar to the TLS\_FALLBACK\_SCSV that aims to protect a session from being downgraded due to the web browser's TLS fallback mechanism. TLS 1.3 also introduces certificate extension fields in the Certificate message to efficiently process certificate-related TLS extensions. Currently, RFC8446 describes signed certificate timestamps (SCTs) and OCSP stapling for the extensions, but is not limited to only them. Please note that TLS implementations need to be updated to process the new Certificate message, even if the TLS implementations have functions related to SCTs and OCSP stapling.

##### **Performance Improvements**

TLS 1.3 reduces the two round-trip times (RTT) for a handshake down to only one RTT. In TLS 1.2, the ClientHello and ServerHello messages are combined with the key exchange messages in

the second roundtrip. However, in TLS 1.3, the ClientHello, ServerHello, and key exchange messages are combined in the first roundtrip itself. Additionally, TLS 1.3 introduces the early\_data extension, which allows clients to send application data along with the first handshake message, without requiring a handshake procedure for resumed sessions. This feature is also known as 0-RTT. In contrast, TLS 1.2 requires one RTT before sending application data.

## **4.2 Related Research**

### **4.2.1 Related Work**

In paper [48], The Authors discussed the alternatives for TLS as TLS is considered resource hungry and impacts the current consumption of the microcontroller. author suggested lightweight versions of TLS; “DTLS” (Datagram Transport Layer Security) and “WTLS” (Wireless Transport Layer Security), used in UDP-based applications and mobile communications respectively, to assure integrity and privacy of data. However, WTLS uses DES or 3DES which is vulnerable now and DTLS only works in case of UDP network.

In paper [49], Authors proposed a light weight algorithm which is symmetric key block cipher. It constitutes 64-bit key. Typical cryptographic algorithm usually consists of on average 10 to 20 rounds The proposed algorithm restricted to only five rounds. The algorithm utilizes the Feistel network. they were able to reduce the current consumption of the micro-controller but their solution is compromising the security to fit in with. Authors of the Paper [51], discussed some approaches to reduce energy consumption of micro-controller over the time and discussed utilization of sleep modes and DVFS technique. authors also demonstrated that entering and waking up from power-saving modes will introduce energy consumption overhead, which must be compensated when configuring sleep cycles of micro-controller. on the other hand, author discussed that unlike the cost of DVFS (Dynamic voltage and frequency scaling), the cost of entering and waking up from energy-saving mode is much smaller, and it is even negligible. Authors also suggested that sleep time of end node should be at least 6 seconds to compensate sleep-wake up overhead. the authors did not demonstrate the power profiling of the microcontrollers.

Mehmat erkan Yüksel in his paper [52] practically demonstrated that Radio transmissions (i.e., transmitting and receiving data) in IoT applications consume much energy compared to sensing and data processing activities. Author also suggested reducing the power consumption of a Wi-Fi

IoT device by leveraging the different sleep modes supported by the device can be the right solution. Using sleep modes is a valuable strategy for dramatically extending the battery life and operational lifetime of the Wi-Fi IoT device that does not need to be active all the time. However, author only provided research over comparison of different transmission channels but the study did not include the practical implementation and analysis of the sleep modes.

In research paper [53], Authors demonstrated the IoT infrastructure with the application of agriculture and applied the TLS v1.2 over the IoT Network to secure the network. The authors state that they were able to secure the network at cost of slightly high consumption as compare to normal consumption. The model uses Http protocol with TLS 1.2 over the network. TLS 1.2 uses RSA which is resource hungry and Vulnerable to factorization attack. Authors did not use updated version of TLS protocol and typical light weight IoT protocol. moreover, energy consumption reduction is not focused. In our perspective, there are some changes required which should be performed to enhance the security and scalability of the network and to reduce the energy consumption of the end nodes.

#### **4.2.2 Significance of Research Work**

It is worth noting that all studies considered for the literature review were dated from 2018 onwards. All articles, papers and research materials are drawn from recently completed work and most of them are from known journals. Not ten years ago or more. This is because the researcher wants to preserve the originality, authenticity, and practicality of the current study. The reason why the most recent studies carried out in this field are considered in the following research works is to actively cooperate to find suitable results.



### 4.2.3 Research Papers Summary

**Table 4-2:** Summary Table of Research Papers

| Journal Paper   | Paper Summary   | Research Limitation   |
|---|---|---|
| [1]<br>Mrabet, H,<br>Belguith, S,<br>Alhomoud, A and<br>Jemai, A                                | <ul style="list-style-type: none"> <li>• TLS alternatives discussed (lightweight protocols)</li> <li>• DTLS(Datagram Transport Layer Security) and WTLS(Wireless Transport Layer Security) , used in UDP-based applications and mobile communications respectively.</li> </ul>    | WTLS uses DES or 3DES which is vulnerable now.<br>DTLS works in case of UDP network.  |
| [2]<br>Sohel Rana1<br>Saddam<br>Hossain2, Hasan<br>Imam Shoun3,<br>Dr. Mohammad<br>Abul Kashem4 | <ul style="list-style-type: none"> <li>• Proposed algorithm is a 64-bit key symmetric key block cipher.</li> <li>• Typical cryptographic algorithm consists 10 to 20 rounds</li> <li>• The proposed algorithm restricted to only five rounds</li> </ul>                           | Their solution is compromising the security to fit in.  |
| [3]<br>Wu, H.; Chen,<br>C.; Weng, K   | <ul style="list-style-type: none"> <li>• DVFS has significant overhead.</li> <li>• Least impact over energy consumption.</li> <li>• Waking up from power-saving modes introduces overhead.</li> <li>• Sleep duration must be at least 6 seconds to cover the overhead.</li> </ul> | No implementation and analysis of sleep modes.  |
| [4]<br>Mehmet Erkan<br>Yüksel”  | <ul style="list-style-type: none"> <li>• Radio transmissions in IoT applications consume much energy.</li> <li>• Reducing the power consumption of device by using sleep modes.</li> </ul>  | Only focused on the sleep modes energy consumptions and their overheads were discussed.   |
| [5]<br>Radomir<br>Prodanović,Dejan<br>Rančić,Ivan<br>Vulić ,Nenad<br>Zorić                      | <ul style="list-style-type: none"> <li>• The model uses Http protocol</li> <li>• The model implements TLS 1.2</li> <li>• Uses RSA which is resource hungry and vulnerable to factorization attack &amp; chosen plain attack.</li> </ul>   | No updated version of TLS security and light weight IoT network MQTT is used. moreover energy consumption reduction is not focused. |

### 4.3 Problem Statement

As per literature survey we can conclude that:

- There is a lack of proper approach to secure the end nodes communication with the broker. researchers have been using different lightweight security protocols to encrypt the communication channel.
- Security should not be compromised and it should also be reliable enough to be used in the highly confidential networks.
- IoT networks itself should base on the IoT friendly messaging protocol and should be able to cater huge network with large number of end nodes.
- Current security schemes have higher CPU utilization, execution time, bandwidth usage, and power consumption with respect to required battery life for node even if we use light weight security protocol.
- Therefore, a solution which offers which offers non vulnerable security with longer battery life is so much needed.

### 4.4 Objective

The major objective of the research is to

- propose well secured IoT network with lightweight messaging protocol MQTT which will be reliable enough to cater huge number of connected devices;
- Implement TLS 1.3 on IoT network and compare with the network without TLS 1.3 to determine the energy consumption of end nodes.
- Furthermore, through this work, we will Perform power profiling using sleep modes for remote area applications to reduce battery consumption for test cases.

## CHAPTER 5: PROPOSED WORK

In this chapter we will discuss the proposed architecture followed by methodology to apply power profiling technique and implementation platform with experimental setup to perform testing.

### 5.1 Architecture of The IoT Network

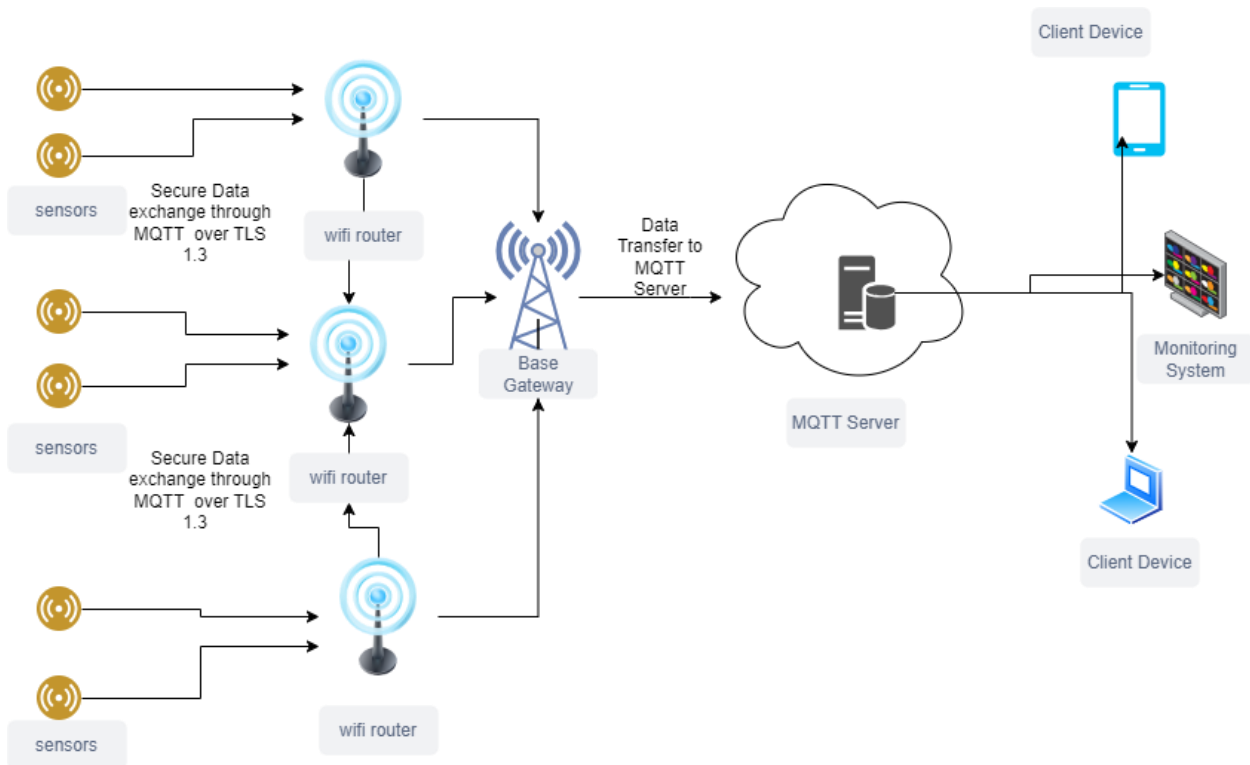
The model attempts to satisfy sensor network security criteria. Its core tenets include data transmission and actor identification in a secure manner, non-repudiation of message receipt, and dependable data provision for decision-making. The TLS 1.3 protocol is used by the model to secure data exchanges, where keys are generated through ECC. Additionally, it implements the MQTT protocol for the IoT sensor network.

The proposed architecture uses a modular network topology that depends on the following Wi-Fi network components: Wi-Fi-enabled sensor nodes, Wi-Fi bridge routers, Wi-Fi gateway, and MQTT server. The figure below illustrates the overview of the model implementation in the field:



**Figure 5.1:** Overview of IoT Network for Monitoring

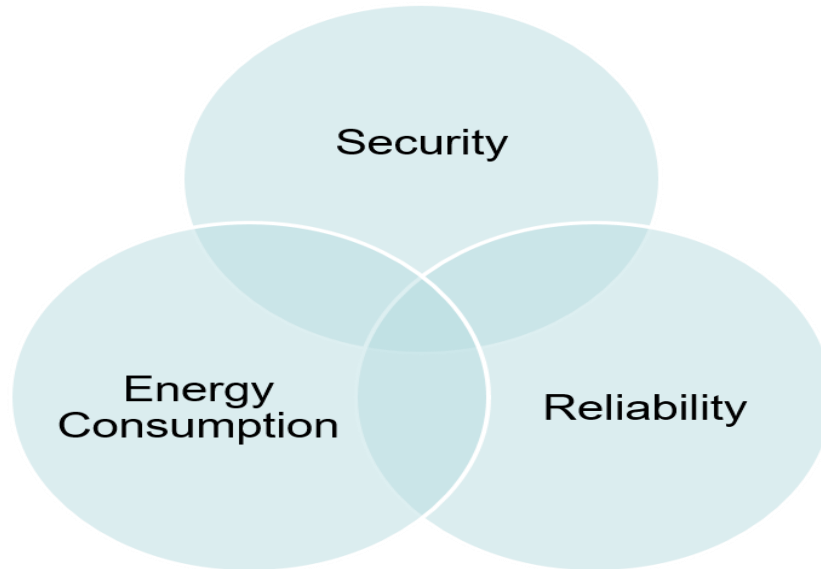
The topology illustrates a more insight view of working of the proposed network. Starting from the sensor node, the data is transferred in MQTT packets to the routers through Wi-Fi which are connected to the base antenna. The communication is secured through recent version of transport layer security. The packets are then delivered to the MQTT server where the received data is accessible to the authorized clients and monitoring system. As we discussed earlier, MQTT has significantly decreased the latency of the network and has been proved more reliable then the IoT network over HTTP. The network topology diagram is given below:



**Figure 5.2:** Proposed IoT Network Topology

## 5.2 Comparison Parameters

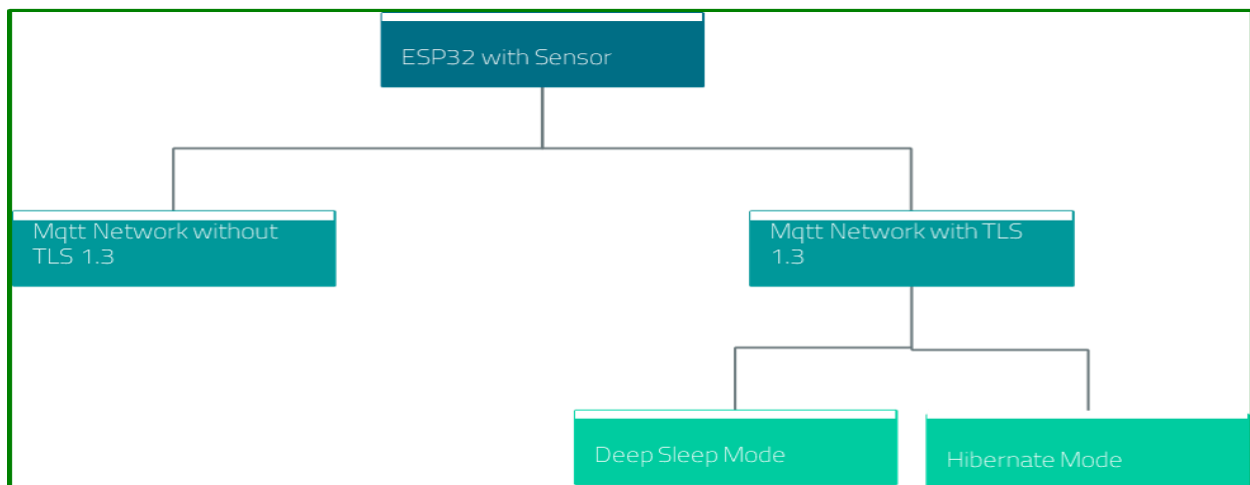
Our proposed approach is evaluated based on three parameters that are crucial for any network, especially when it comes to high-profile communication. Our study considered these parameters to ensure security requirements performance requirements are met while also extending the life cycle of end nodes over a reliable network. These parameters are demonstrated below:



**Figure 5.3:** Parameters Consideration

### 5.3 Testing Methodology

We developed a testing methodology to implement the power profiling technique and compare the results. The first phase of testing involves the environment setup of our proposed architecture without secure connection to fetch the energy consumption of the microcontroller over different stages. The second phase includes the implementation of TLS protocol over the communication to find out the difference. The final phase implements power profiling of microcontroller to minimize the power consumption of the microcontrollers to find out the effectiveness of the technique. Below is illustration of the proposed testing methodology.



**Figure 5.4:** Testing Methodology

## 5.4 Implementation Platform

Our implementation platform can be divided into three parts which are publishers (end nodes), broker (middle-ware) and subscribers (user end devices). whereas we have used some additional soft wares and tools for testing purposes. we intend to describe the implementation platform and then move forward towards the detailed analysis of the experimental setup. The below is the high level illustration of the implementation platform.

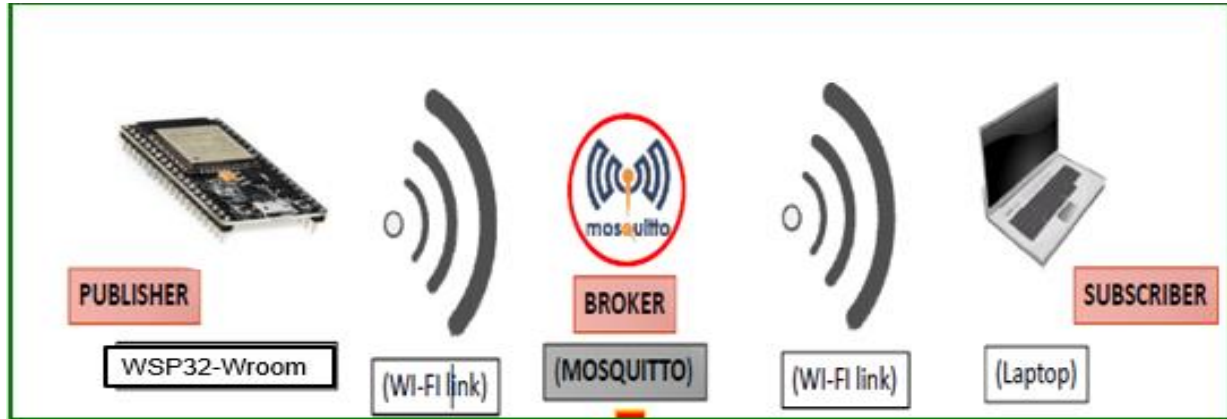


Figure 5.5: Implementation Platform

## 5.5 Experimental Setup

We have a combination of hardware and software tools and devices in our experimental setup. We have segmented the experiment setup to provide a better understanding of it. The general experimental setup for the research investigation is shown below:

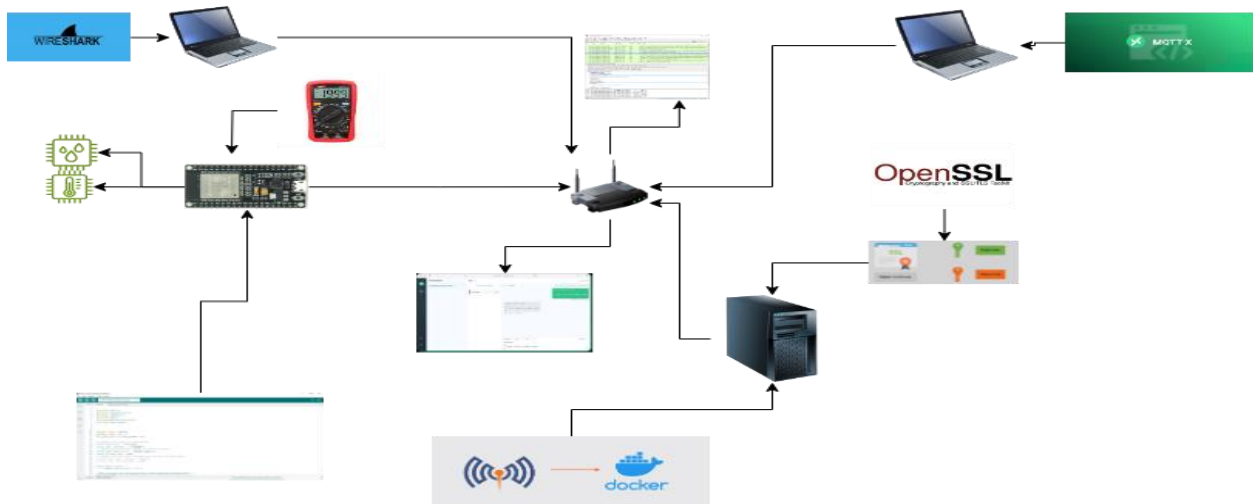


Figure 5.6: Experimental Setup

### 5.5.1 Sensor Node Setup

We used a ‘ESP32’ microcontroller with a ‘DHT 11’ temperature sensor, which is a widely used temperature and humidity sensor that includes an 8-bit microcontroller to output the temperature and humidity values as serial data [56] and a ‘Capacitive Soil Moisture Sensor’, which uses capacitive sensing to measure soil moisture levels by varying capacitance based on the amount of water in the soil. Voltage level is essentially converted from capacitance between 1.2V and 3.0V maximum. For our end node to monitor the agricultural land, the advantage of the Capacitive Soil Moisture Sensor is that it is built of a corrosion-resistant substance giving it a long service life [57]. Powerful System on Chip (SoC) microcontroller ESP32 includes dual-mode Bluetooth 4.2, integrated Wi-Fi 802.11 b/g/n, and a number of peripherals. Its two cores, each clocked at a distinct speed of up to 240 MHz, make it a more sophisticated chip than the 8266 [54]. Because this microcontroller has a large number of sleep modes. Advanced power-management technologies enable the ESP32 to transition between several power modes. These power settings include hibernation, deep sleep, light sleep, and modem sleep. We leveraged the deep sleep power mode and hibernate mode for our use case as these consume the least power [55]. The sample code to leverage Deep Sleep Mode is below which can also be used to enable hibernate mode by little tweaking.

```
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);  
esp_deep_sleep_start();
```

Moving forward, we used ‘Arduino IDE’ (Integrated Development Environment) to program the microcontroller. It is an official program that was unveiled by Arduino.cc and is mostly used for uploading, modifying, and compiling code into Arduino devices. This program is freely accessible for download and installation, and it can be used to compile code on the go for almost any Arduino module. The Arduino (IDE) also provides the platform to communicate with Arduino boards to upload programs and analyze the communication between broker and the micro-controller [58]. Below is the screenshot of test code upload and serial monitor window of Arduino software which is used to determine the status of the communication.

```
118 //mqttClient.onUnsubscribe(onMqttUnsubscribe);
119 mqttClient.onPublish(onMqttPublish);
120 mqttClient.setServer(MQTT_HOST, MQTT_PORT);
121 // If your broker requires authentication (username and password), see
122 //mqttClient.setCredentials("REPLACE_WITH_YOUR_USER", "REPLACE_WITH_Y
123 connectToWifi());
```

Serial Monitor x Output

Not connected. Select a board and a port to connect automatically.

```
Publishing on topic esp32/dht/temperature at QoS 1, packetId: 0Message: 31.60
Publishing on topic esp32/dht/humidity at QoS 1, packetId 0: Message: 48.00
Connecting to MQTT...
Disconnected from MQTT.
Publishing on topic esp32/dht/temperature at QoS 1, packetId: 0Message: 31.60
Publishing on topic esp32/dht/humidity at QoS 1, packetId 0: Message: 48.00
Publishing on topic esp32/dht/temperature at QoS 1, packetId: 0Message: 31.60
Publishing on topic esp32/dht/humidity at QoS 1, packetId 0: Message: 48.00
Connecting to MQTT...
Disconnected from MQTT.
Publishing on topic esp32/dht/temperature at QoS 1, packetId: 0Message: 31.60
Publishing on topic esp32/dht/humidity at QoS 1, packetId 0: Message: 48.00
Publishing on topic esp32/dht/temperature at QoS 1, packetId: 0Message: 31.60
Publishing on topic esp32/dht/humidity at QoS 1, packetId 0: Message: 48.00
Connecting to MQTT...
```

**Figure 5.7:** Arduino IDE and Serial Monitor

## 5.5.2 MQTT Server Deployment

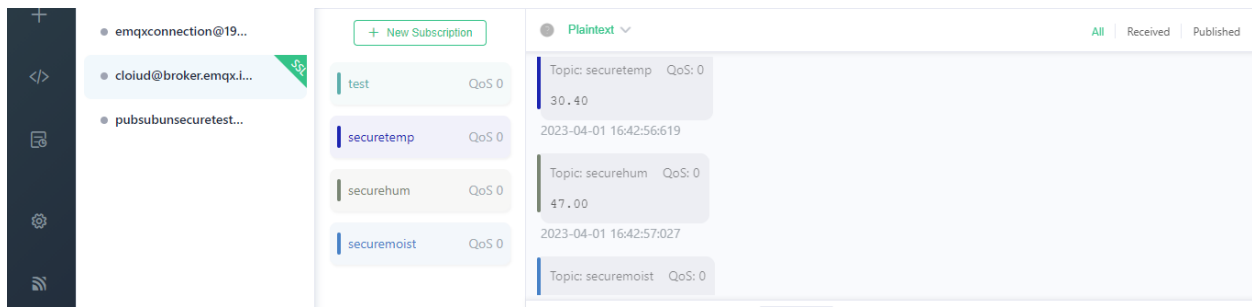
For Deployment of MQTT broker 'Mosquito' is a dependable and open-source (EPL/EDL licensed) message broker that supports MQTT protocol versions 5.0, 3.1.1, and 3.1. We used a machine as a server with conventional specs of 16 GB RAM, core i7 2.6 GHz processor, and Windows Operating machine. We installed the 'Docker' software over the system and pulled the image of mosquito broker from Docker repository. Docker is an open source platform for running applications and streamlining the development and deployment processes. Docker-built applications are packaged in a standard format known as a container, which includes all supported dependencies. These containers run independently on top of the operating system kernel. Additional levels of abstraction can impact performance. Docker provides a way to automate the deployment of applications into Containers. Container environments, where applications are running virtualized, add an additional layer of delivery engine. It is intended to give a quick and easy environment to run code effectively. It is designed to provide a fast and simple environment to run your code efficiently. In addition, it provides additional opportunities for specialized work processes to retrieve code from computers for testing before production, and most importantly, significantly reduces the cost of rebuilding cloud development platforms [59]. we configured the mosquito broker with default ports for MQTT communication port 1883 and 8883 for unsecured



and connection over TLS 1.3 respectively. We also configured the TLS version to 1.3 in the config file of the broker to make sure the broker uses the desired version of TLS. It also provided us the authentication functionality to secure the network from MITM attacks and unauthorized access.

### 5.5.3 Monitoring Through MQTT Client

After configuring the server, we installed the MQTT client software ‘MQTTX’ over the laptop which we used as a subscriber to test the connection and fetch the readings provided by the deployed publisher as MQTT protocol routes messages based on topic. MQTTX provides user friendly interface to connect with broker and subscribe the desired topic. below is the screen view of the MQTTX which illustrates the testing on the client software.



**Figure 5.8:** MQTTX Client Interface

### 5.5.4 Private Certificate Authority to Implement TLS 1.3

We deployed private certificate authority to implement TLS 1.3 over IoT network. ‘Open SSL’ tool was leveraged to create the private certificate authority and generate certificates and keys. The process starts with the generation of private keys using EC algorithm, which is best alternative to RSA as EC is more successful in some metrics, such as "memory usage, key size, energy consumption, signature generation time, key generation and execution time, and decryption time". ECC outperforms RSA in terms of security and operating efficiency on small devices with limited resources. [60].

The following is the command to generate the private key using OpenSSL:

```
openssl ecparam -name prime256v1 -genkey -noout -out private-key.pem
```

This resulted in a PEM file holding our EC private key, which looks like this:

```
-----BEGIN EC PRIVATE KEY-----  
  
MHcCAQEIEuBpBiHkZQYlORbCy8gGTz8tzrWsjBJA6GfFCrQ98coAoGCCqGSM49  
  
AwEHoUQDQgAEOr6rMmRRNKuZuwws/hWwFTM6ECEeAJGGARCJU04UfoUR18b4JThG  
  
t8VDFKeR2i+ZxE+xh/wTBaJ/zvtSqZiNnQ==  
  
-----END EC PRIVATE KEY-----
```

Now that we had our private key, we utilized it to create another PEM, including our public key, using the following command:

```
openssl ec -in private-key.pem -pubout -out public-key.pem
```

This generated another PEM file, which contained the public key:

```
-----BEGIN PUBLIC KEY-----  
  
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEOr6rMmRRNKuZuwws/hWwFTM6ECEe  
  
AJGGARCJU04UfoUR18b4JThGt8VDFKeR2i+ZxE+xh/wTBaJ/zvtSqZiNnQ==  
  
-----END PUBLIC KEY-----
```

Now, root certificate will be generated:

```
openssl req -x509 -new -nodes -key myCA.key -sha256 -days 1825 -out myCA.pem
```

This prompted for the passphrase of the private key we just chose and a bunch of questions. The answers to those questions aren't that important. They show up when looking at the certificate. We made the Common Name something that we'll recognize as our root certificate in a list of other certificates. That's really the only thing that matters. The CA prompt look like below:

```
Enter pass phrase for myCA.key:
```

```
You are about to be asked to enter information that will be incorporated  
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
```

```
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
```

```
-----
```

```
Country Name (2 letter code) [PK]:PK
```

```
State or Province Name (full name) [Some-State]: Sindh
```

```
Locality Name (eg, city) []: Karachi
```

```
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ABC network
```

```
Organizational Unit Name (eg, section) []: subsection
```

```
Common Name (e.g. server FQDN or YOUR name) []: Basit Server
```

```
Email Address []: basit@ABC.com
```

Now that we have a private key and a root certificate, we will use the private key to generate a self-signed certificate:

```
openssl req -new -x509 -key private-key.pem -out cert.pem -days 360
```

This generated one more PEM file, which contains the certificate created by our private key:

```
-----BEGIN CERTIFICATE-----
```

```
MIIB4DCCAYWgAwIBAgIUH53ssiPt4JEGx+VJyntCpHL+TdAwCgYIKoZIzj0EAwIw
```

```
RTElMAkGA1UEBhMCVUxEzARBgNVBAgMC1NvbWUtU3RhdGUxITAfBgNVBAoMGE1u
```

```
dGVybmlV0IFdpZGdpdHMgUHR5IEEx0ZDAeFw0yMDA3MTgxMTE4NDNaFw0yMTA3MTgx
```

```
MTE4NDNaMEUxCzAJBgNVBAYTAkFVMRMwEQYDVOQQIDApTb211LVN0YXR1MSEwHwYD
```

```
VQkDBhJbnR1cm5ldCBXaWRnaXRzIFB0eSBMdGQwWTATBgqhkJ0PQIBBgqhkJ0
```

```
PQMBBwNCAAQ6vqsyZFE0q5m7DCz+FbAVMzoQIQRokYYBEI1Q7hR+hRGXxvg10Ea3
```

```
xUMUp5HaL5nET7GH/BMFon/O+1KpmI2do1MwJTAdBgNVHQ4EFgQU9yjFBqAZOMv+
```

```
cD6a3KHTWuYrcFEwHwYDVR0jBBgwFoAU9yjFBqAZOMv+cD6a3KHTWuYrcFEwDwYD
```

```
VR0TAQH/BAUwAwEB/zAKBggqhkjOPQDAQgNJADBGAiEAWCpA5Nx083qqUqU6LUd0
```

```
vzZLK4etuInxNvXohXH5LiACIQDSI63J4DiN3dq2sPPLw5iQi9MMefcV1iAySbKT
```

```
B9BaAw==
```

```
-----END CERTIFICATE-----
```

We generated two more certificates to complete the process which are server certificate and client certificate which are signed by self-signed certificate. These are required to add extra layer of authentication. The server certificate is generated through commands below:

```
openssl req -new -newkey ec:ec_private_key.pem -keyout ec_server_key.pem -out  
ec_server_csr.pem
```

```
openssl x509 -req -in ec_server_csr.pem -CA ec_self_signed_cert.pem -CAkey  
ec_private_key.pem -CAcreateserial -out ec_server_cert.pem -days 365
```

and commands used to generate client certificate similar to server certificate generation are below:

```
openssl req -new -newkey ec:ec_private_key.pem -keyout ec_client_key.pem -out  
ec_client_csr.pem
```

```
openssl x509 -req -in ec_client_csr.pem -CA ec_self_signed_cert.pem -CAkey  
ec_private_key.pem -CAcreateserial -out ec_client_cert.pem -days 365
```

These set of commands executed for server certificate and client certificate generated two certificates which looked like below:

```
-----BEGIN CERTIFICATE-----  
  
MIIB4DCCAYWgAwIBAgIUH53ssiPt4JEGx+VJyntCpHL+TdAwCgYIKoZIzj0EAwIw  
  
RTELMAkGA1UEBhMCQVUxEzARBgNVBAgMC1NvbWUtU3RhdGUxITAfBgNVBAoMGE1u  
  
dGVybmV0IFdpZGpdHMgUHR5IEEx0ZDAeFw0yMDA3MTgxMTE4NDNaFw0yMTA3MTMx  
  
MTE4NDNaMEUxCzAJBgNVBAYTAkFVMRMwEQYDVQQIDApTb211LVN0YXR1MSEwHwYD  
  
VQQKBHJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQwWTATBgqhkJOPQIBBggqhkJ0  
  
PQMBBwNCAAQ6vqsyZFE0q5m7DCz+FbAVMzoQIQRokYYBEI1Q7hR+hRGXxvg1OEa3  
  
xUMUp5HaL5nET7GH/BMFon/O+1KpmI2do1MwUTAdBgNVHQ4EFgQU9yjFBqAZOMv+  
  
cD6a3KHTWuYrcFEwHwYDVR0jBBgwFoAU9yjFBqAZOMv+cD6a3KHTWuYrcFEwDwYD  
  
VR0TAQH/BAUwAwEB/zAKBggqhkJOPQDAgNJA DBGAiEAwCpA5Nx083qqUqU6LUd0  
  
vzZLK4etuInxNvXohXH5LiACIQDSI63J4DiN3dq2sPPLw5iQi9MMefcV1iAySbKT  
  
B9BaAw==  
  
-----END CERTIFICATE-----
```

After obtaining keys and certificates for our private certificate authority, we uploaded the CA certificate, private key and client certificate over the ESP32 and configured the Certificate

Authority folder with the broker. lastly we uploaded the same certificates and private key over the MQTTX client software. This completed the process to implement the TLS 1.3 over our network.

### 5.5.5 Our Network Analysis

After setting up our experiment platform, we installed the network analysis software ‘Wireshark’ to analyze the network during communication and ensure the required protocols are used when required. Wireshark is a network protocol analyzer that captures packets sent across a network connection, such as from a computer to a home office or the internet. A packet refers to a distinct unit of data in a standard Ethernet network. Wireshark, like any other packet sniffer, has three primary functions: packet capture, filtering, and visualization. Packet Capture entails listening to a network connection in real time and capturing full streams of communication. Filters allow you to slice and dice all of this random live data. Visualization allows you to go straight into the middle of a network packet and see full conversations and network streams [61]. we used the tool to analyze the packets of MQTT over our network and make sure that data is encrypted over the network and using the TLS 1.3 when cipher suites come in the picture. The screenshot below is the analysis window of the tool which shows the packets are being transmitted as per desired.

| No. | Time      | Source            | Destination   | Protocol | Length | Info   |
|-----|-----------|-------------------|---------------|----------|--------|--|
| 82  | 46.592115 | Fiberhom_f6:e0:1e | Broadcast     | ARP      | 42     | Who has 192.168.10.9? Tell 192.168.10.1              |
| 83  | 47.208363 | 3.228.54.173      | 192.168.10.10 | TLSv1.3  | 94     | Application Data                                     |
| 84  | 47.209108 | 3.228.54.173      | 192.168.10.10 | TLSv1.3  | 93     | Application Data                                     |
| 85  | 47.209108 | 3.228.54.173      | 192.168.10.10 | TLSv1.3  | 96     | Application Data                                     |
| 86  | 47.209248 | 192.168.10.10     | 3.228.54.173  | TCP      | 54     | 61929 → 8883 [ACK] Seq=546 Ack=5025 Win=131328 Len=0 |
| 87  | 47.616223 | Fiberhom_f6:e0:1e | Broadcast     | ARP      | 42     | Who has 192.168.10.9? Tell 192.168.10.1              |
| 88  | 48.232243 | 3.228.54.173      | 192.168.10.10 | TLSv1.3  | 94     | Application Data                                     |
| 89  | 48.232892 | 3.228.54.173      | 192.168.10.10 | TLSv1.3  | 113    | Application Data                                     |
| 90  | 48.232971 | 192.168.10.10     | 3.228.54.173  | TCP      | 54     | 61929 → 8883 [ACK] Seq=546 Ack=5124 Win=131328 Len=0 |
| 91  | 48.606320 | Fiberhom_f6:e0:1e | Broadcast     | ARP      | 42     | Who has 192.168.10.9? Tell 192.168.10.1              |

```

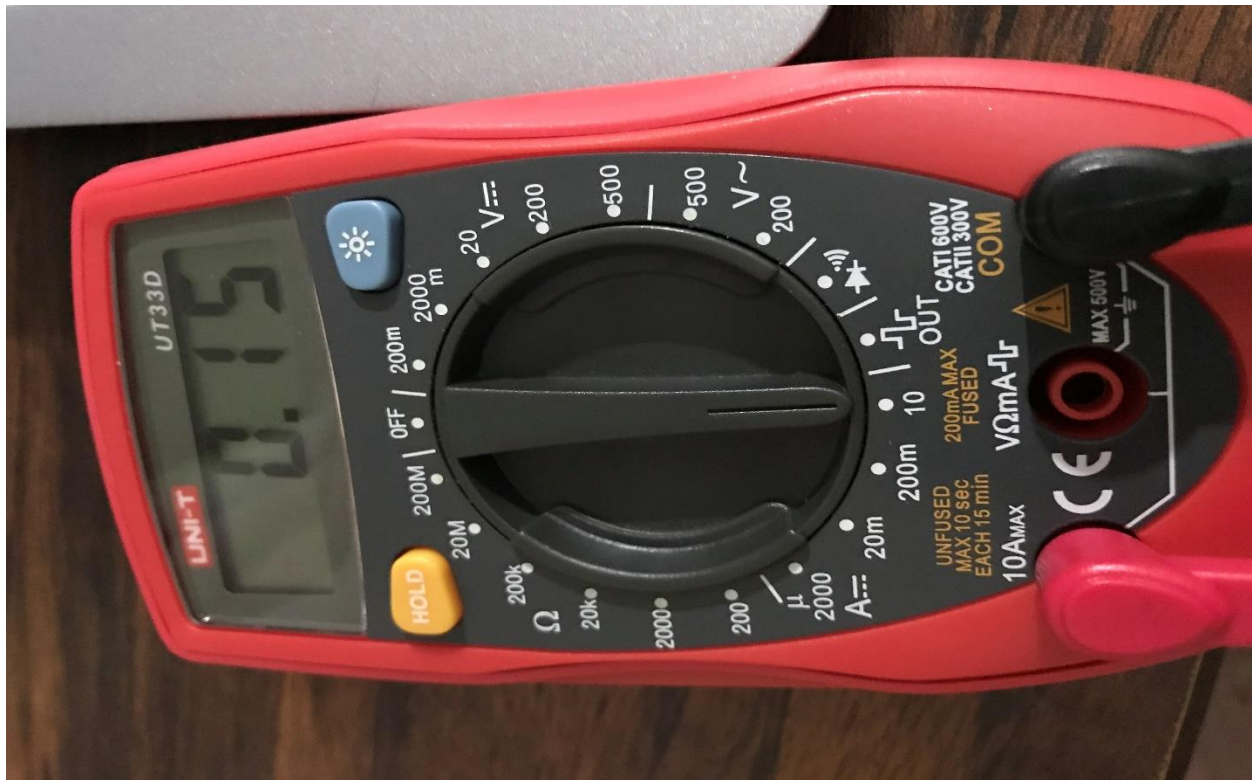
> Frame 85: 96 bytes on wire (768 bits), 96 bytes captured (768 bits) on interface \Device\NPF_{A2174904-3DA1-4C63-B3B6-F7EA75E4EA9A}, id 0
> Ethernet II, Src: Fiberhom_f6:e0:1e (ec:8a:c7:f6:e0:1e), Dst: IntelCor_79:f9:f5 (34:f3:9a:79:f9:f5)
> Internet Protocol Version 4, Src: 3.228.54.173, Dst: 192.168.10.10
> Transmission Control Protocol, Src Port: 8883, Dst Port: 61929, Seq: 4983, Ack: 546, Len: 42
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Application Data Protocol: mqtt
    Opaque Type: Application Data (23)
  
```

**Figure 5.9:** Network Analysis With Wireshark

### 5.5.6 Current Measurement

This is where our methodology comes into picture as we have developed and tested our experimental setup, we began our testing with a digital Multimeter, which is an instrument for measuring two or more electrical variables, namely voltage (volts), current (amps), and resistance (ohms). [62]. As per our methodology, we started filling up our readings table with the current

consumption of the ESP32 when it sends data over the MQTT without any protocol followed by the readings with TLS 1.3 implemented. Next wards, we implemented power profiling technique; In our case of agricultural monitoring application of IoT, we need to wake up our end node 6 six times per 24 hours on average according to R. Evans and D. Cassel [63] and recorded readings of two variants ‘Deep-Sleep’ and ‘Hibernate’ of sleep modes of the ESP32 microcontroller. We also recorded readings for two variants of MQTT functionality libraries and the results were impressive. Below is the glimpse of the measurement of the current consumption.



**Figure 5.10:** Current Measurement Using Multi-meter



## CHAPTER: 6 RESULT & DISCUSSION

In this part, we will exhibit comparison table for the architectures and go through our results followed by a detailed discussion.

### 6.1 Comparison Table for Network Architectures

The comparison table of proposed architecture with the architecture presented by authors of [53] is put down below to illustrate the difference of the two architectures:

**Table 6-1:** Comparison Table for Architectures

| Architecture                           | Application Protocol | Security Protocol | Security Key Generation Algorithm | Reliable for Real-time Applications | Latency | Energy Efficient |
|--|----------------------|-------------------|-----------------------------------|-------------------------------------|---------|------------------|
| <b>Prodanović R, Rančić D and team</b> | HTTP                 | TLS 1.2           | RSA                               | Yes                                 | High    | No               |
| <b>Proposed</b>                        | MQTT                 | TLS 1.3           | EC                                | Yes                                 | Low     | High             |

The comparison table clearly describe that proposed architecture has been tweaked to make the architecture communication more secure, Scalable and light weight with energy efficient as well. The changes in architecture are focused to provide more IoT friendly solution.

### 6.2 Experiment Results

#### 6.2.1 MQTT Implement Libraries

During the experiment, we notice significant difference between the energy consumptions cause by the implementation libraries of the MQTT protocol. When we leveraged the AsyncMQTT library in the code and took the readings it was far more high then the readings taken with PubSub Library. Even the TLS 1,3 implemented with ‘PubSubClient’ library consumed less current then the AsyncMQTT unsecured connection while operating. The primary reason for the difference is

because ASyncMQTT is asynchronous and PubSub Client has synchronous structure. The table Below shows the difference in the energy consumption due to these libraries

**Table 6-2:** Reading Results for AsyncMQTT and Pub-Sub Client Library Current Consumption

| Connection Mode             | startup current (1-sec) | When sending data | Operating Current |
|-----------------------------|-------------------------|-------------------|-------------------|
| <b>Unsecure (AsyncMQTT)</b> | 150mA                   | 110mA             | 80mA              |
| <b>Unsecure (PubSub)</b>    | 110mA                   | 70mA              | 60mA              |
| <b>TLS 1.3 (PubSub)</b>     | 120mA                   | 90mA              | 60mA              |

### 6.2.2 Power Profiling with Power Modes Results

Since Pub-Sub Client Library proved to be more energy efficient than the ‘AsyncMQTT’, we used it the implementation and analysis for the current consumption of the power modes with power profiling. The result shows that no difference was noted in the deep sleep mode and hibernate mode because there are certain variants of the ‘ESP32’ with different energy consumption over the same power mode. However, they have found effective in the extending the battery life significantly. The reading results for the Sleep modes (Deep-Sleep and Hibernate) are as under.

**Table 6-3:** Reading Results for Power modes with Pub-Sub Library Current Consumption

| Sleep Mode                | Startup Current (1-sec) | When Sending Data | Normal Mode | Sleep Mode Current |
|---------------------------|-------------------------|-------------------|-------------|--------------------|
| <b>TLS 1.3 Deep Sleep</b> | 120mA                   | 90mA              | 60mA        | 20mA               |
| <b>TLS 1.3 hibernate</b>  | 120mA                   | 90mA              | 60mA        | 20mA               |

Moving forward, we will discuss the reading result and use these readings to calculate the expected battery life of the End nodes.

## 6.3 Reading Results Analysis

Since there was no difference noted in two variant of sleep modes, we will be using Deep-Sleep mode in our scenario, Power prowlng in our case requires us to put the esp32 end node on sleep mode for the rest of time and fetch the reading six times a day as per suggested by R. Evans and D. Cassel [63].

### 6.3.1 Formula to Calculate the Battery Life

To work out how much battery life, we will achieve over each use case. below is the general formula for the battery life which is the base of our calculation for energy consumption. we need to consider the following to meet the requirements of end node battery life:

#### BATTERY LIFE FORMULA

$$\text{Battery Life} = \frac{\text{Battery Capacity (mAh)}}{\text{Load Current (mA)}}$$

**Figure 6.1:** General Formula to Calculate Battery Life [64]

The average current draw over the period of an hour The peak current draw the device will draw The length of time you need the device to run for between battery changes / recharges. as per our use case scenarios, we will need to calculate factors which combined derive the load current. we need to calculate the sleep mode duration and current consumption, data transmission current consumption and data processing time and current consumption. for this purpose, following formula will be used which is derived originally from the general formula for battery life calculation.

To calculate battery life, divide battery capacity by average device current consumption over time. You can then divide it into four categories.:

- The current consumed on average when collecting data, scaled by the ratio of time that the device is collecting data.
- The current consumed when transmitting data, scaled by the ratio of time that the device is transmitting data.
- The current consumed when processing data, scaled by the ratio of time that the device is processing data.
- The current consumed when in Sleep Mode, scaled by the ratio of time that the device is in Sleep Mode.

$$B_l = \frac{B_c}{\frac{I_s t_s}{t_s + t_t + t_c + t_p} + \frac{I_t t_t}{t_s + t_t + t_c + t_p} + \frac{I_c t_c}{t_s + t_t + t_c + t_p} + \frac{I_p t_p}{t_s + t_t + t_c + t_p}}$$

**Figure 6.2:** Derived Formula to Calculate Battery Life

Where:

- $B_l$  = Battery lifetime in hours
- $B_c$  = Battery capacity in mAh
- $I_s$  = Device current consumption when in Sleep Mode in mA
- $I_t$  = Device current consumption when transmitting data in mA
- $I_c$  = Device current consumption when collecting data in mA
- $I_p$  = Device current consumption when processing data in mA
- $t_s$  = Time spent in Sleep Mode in seconds (per cycle)
- $t_t$  = Time spent transmitting data in seconds (per cycle)
- $t_c$  = Time spent collecting data in seconds (per cycle)
- $t_p$  = Time spent processing data in seconds (per cycle)

### 6.3.2 Uses Cases for Our Battery life Calculations

In this subsection we will take the readings of each testing scenario and calculate the battery life of micro-controller in each case followed by combined results graph.

### **Esp32 readings with Async MQTT library:**

In case 1, we used Async MQTT library for MQTT functionality in our code. we did not implement TLS 1.3 over the network. the traffic was insecure and not encrypted. ESP consumes 80mAh in active mode with a peak of 110mAh when sending data and doesn't go into sleep mode, with this average current consumption a 2400mAh battery would last

$$(2400\text{mAh} / 90.022 \text{ mAh}) = 26.66 \text{ Hours.}$$

### **Esp32 readings with Pub-Sub library:**

In case 2, we used Pub-Sub library for MQTT functionality in our code. we did not implement TLS 1.3 over the network. the traffic was insecure and not encrypted. ESP consumes 60mA with a fluctuation of 70mA when sending data and doesn't go into sleep mode, with this average current consumption a 2400mAh battery would last for

$$(2400\text{mAh} / 63.34\text{mA}) = 37.89 \text{ hours before running flat.}$$

### **Esp32 readings with Pub-Sub library over TLS 1.3:**

In case 3, we used Pub-Sub library for MQTT functionality in our code. now, we implement TLS 1.3 over the network. the traffic was secure and encrypted. ESP consumes 60mA when active with a fluctuation of 100mA when sending data and doesn't go into sleep mode, with this average current consumption a 2400mAh battery would last for approximately

$$(2400\text{mAh} / 73.34\text{mA}) = 32.72 \text{ hours.}$$

### **Esp32 ideal readings with Deep-Sleep mode:**

In case 3, we used Pub-Sub library for MQTT functionality in our code. now, we implement TLS 1.3 over the network. the traffic was secure and encrypted. moreover, we applied power profiling over esp32 to keep it awake six times per 24 hours for 6 seconds and keep it in sleep mode for the rest of the time. ESP32 consumes 60mA when active, 100mA when sending data and 20mA when in deep sleep mode, and it wakes up 6 times per 24 hours for 6 seconds each time to take readings. with this average current consumption, a 2400mAh battery would last for approximately

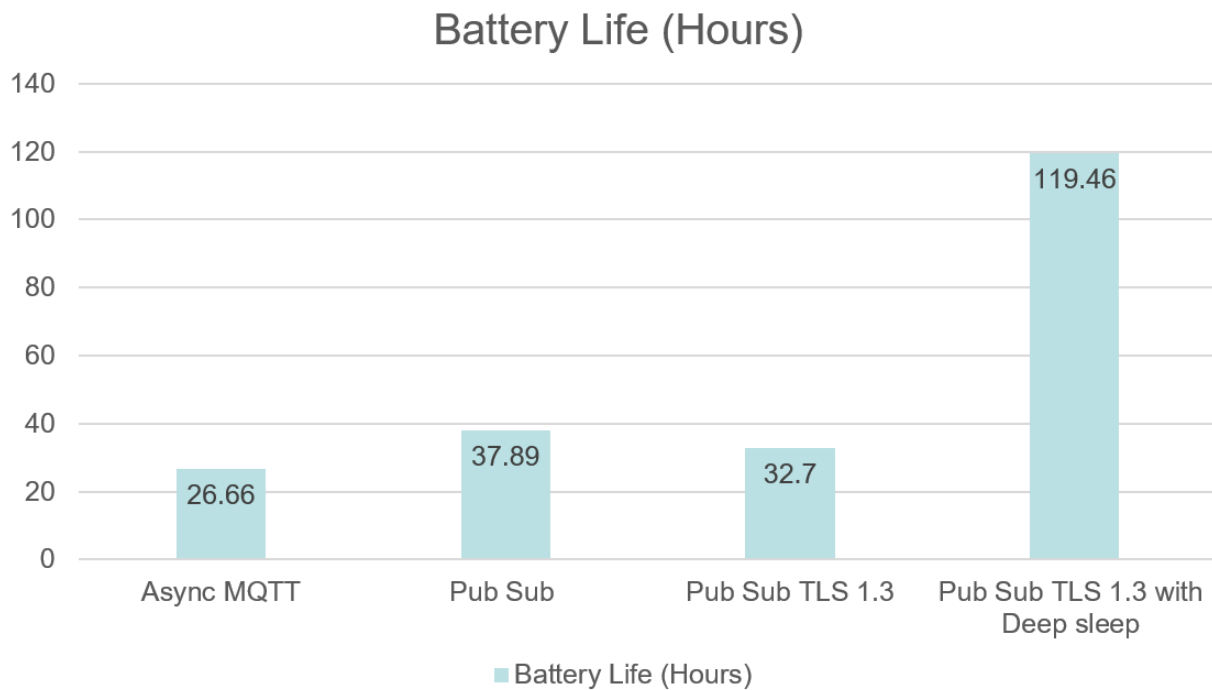
$$(2400\text{mAh} / 20.090\text{mA}) = 119.46 \text{ hours or about 5 days.}$$

### Esp32 Original Readings with deep sleep mode with 10000 mAh battery:

Now, we have increased the battery size to determine the impact of high capacity of battery over the life cycle of the end node. ESP32 consumes 60mA when active, 100mA when sending data and 20mA when in deep sleep mode, and it wakes up 6 times per 24 hours for 6 seconds each time to take readings. With this average current consumption, a 10000mAh battery would last for approximately

$$(10000\text{mAh} / 20.090\text{mA}) = 497.76 \text{ hours or about } 20.7 \text{ days}$$

The following results covers the four scenario and excludes the ideal scenario since battery capacity is changed.



**Figure 6.3:** Battery Life Graph for Use Cases

## CHAPTER 7: CONCLUSION & FUTURE WORK

### 7.1 Conclusion

In our research, we demonstrated the lightweight and IoT-friendly reliable network where we replaced the typical Application protocol with lightweight and vulnerable security protocol version with efficient and secure one where EC algorithm was used to generate keys instead of RSA to minimize the energy consumption and increase the performance. additionally, the library to implement the application protocol was swapped to reduce the energy consumption which greatly impacted results of our research. The results showed that optimized programming of the microcontroller has unneglectable impact over the energy consumption. we presented the comparison table for the proposed architecture where we outlined the basic differences.

Implementation of Transport Layer Security has a noticeable overhead which causes the reduced battery life. However, unsecure connections are not so battery life friendly too. The MQTT protocol with secure connection over TLS 1.3 is far more capable to tackle the huge IoT networks comparatively. Power profiling increases the battery life significantly which fulfills the requirements of wireless nodes and remote areas nodes. Through our research we concluded following with baseline of A sync MQTT library without TLS:

- Pub-sub Library without TLS 1.3 has **42.11%** more battery life.
- Pub-sub Library with TLS 1.3 has **22.54%** more battery life.
- Pub-sub Library with TLS 1.3 after power profiling has **347.98%** more Battery life.

Power profiling can be used in the several applications and it can increase the battery time from months to years depending upon the requirement of the use case Scenario and micro-controller used. This power profiling approach can be used with more advanced hardware which can inherently enhanced the power consumption.

## 7.2 Future Work

We believe that it is worthwhile to select a broader range of microcontrollers of different types and designs, as well as in diverse external contexts, to take a deeper look. Sleep modes not only support time based wakeup but there are two other modes where it uses pin high low state and pin touch to wake up. These methods can also be used in the implementations of various applications other than monitoring.



## REFERENCES

- [1] Lightweight cryptography in IoT networks: A survey Muhammad Rana, Quazi Mamun, Rafiqul Islam School of Computing, Mathematics and Engineering, Charles Sturt University, Australia
- [2] Perception layer security in Internet of Things Hasan Ali Khattak a, Munam Ali Shah a, Sangeen Khan a, Ihsan Ali b, Muhammad Imran.
- [3] Smart streetlights in Smart City: a case study of Sheffield Easley Dizon1· Bernardi Pranggono Journal of Ambient Intelligence and Humanized Computing (2022) 13:2045–2060 <https://doi.org/10.1007/s12652-021-02970-y>
- [4] <https://www.forbes.com/sites/forbestechcouncil/2021/09/22/what-to-know-about-smart-farming-using-iot/>
- [5] IoT in Agriculture: 9 Technology Use Cases for Smart Farming (and Challenges to Consider) (easternpeak.com)
- [6] How Wearable Devices and IoT in Healthcare are Monitoring Patient Health, Improving Outcomes, and Transforming the Healthcare Industry - Healthtech News | The Financial Express
- [7] IoT Remote Monitoring is Transforming Remote Patient Monitoring (drkumo.com)
- [8] Madakam, S., & Uchiya, T. (2019). Industrial Internet of Things (IIoT): Principles, Processes and Protocols. In Computer Communications and Networks (pp. 19-36). Springer International Publishing
- [9] Ten benefits/advantages of IIoT and analytics for manufacturers. (2020). IIoT World. <https://www.iiot-world.com/industrial-iiot/connected-industry/ten-advantages-of-iiot-and-analytics-for-manufacturers/>
- [10] IEA. (n.d.). Smart grids. IEA Energy System. <https://www.iea.org/energy-system/electricity/smart-grids>.
- [11] Telit. (2023). IoT Energy Management Systems: 7 Benefits of Smart Grids. Telit. <https://www.telit.com/blog/iiot-smart-grid-benefits/>
- [12] Digi International. (2022). IoT-Based Environmental Monitoring: Types and Use Cases. <https://www.digi.com/blog/post/iiot-based-environmental-monitoring>

- [13] Environment.co. (n.d.). IoT-Based Environmental Monitoring | Environment.co. <https://environment.co/iot-based-environmental-monitoring-sustainability/>
- [14] Kshetri, N., & Voas, J. (2020). The Internet of Things in Retail: A Review. *IEEE Internet Computing*, 24(2), 26-35.
- [15] Berman, B. (2018). The Internet of Things (IoT) in retail. *Research-Technology Management*, 61(2), 44-50
- [16] Real Time Tracking - IoT for Transportation and Logistics. (n.d.). IoT For All. <https://www.iotforall.com/real-time-tracking>
- [17] Madakam, S., & Uchiya, T. (2019). Industrial Internet of Things (IIoT): Principles, Processes and Protocols. In *Computer Communications and Networks* (pp. 19-36). Springer International Publishing.
- [18] IoT Home Automation - GeeksforGeeks. (n.d.). <https://www.geeksforgeeks.org/iot-home-automation/>
- [19] Smart homes and IoT: How technology is revolutionizing architecture? (2023). Parametric Architecture. <https://parametric-architecture.com/smart-homes-and-iot-how-technology-is-revolutionizing-architecture/>
- [20] Digi International. (2022). IoT-Based Environmental Monitoring: Types and Use Cases. <https://www.digi.com/blog/post/iot-based-environmental-monitoring>
- [21] Nitin Naik, Defence School of Communications and Information Systems "Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP"
- [22] Mrabet, H., Belguith, S., Alhomoud, A., & Jemai, A. (2020). A survey of IOT security based on a layered architecture of sensing and data analysis. *Sensors*, 20(13), 3625. <https://doi.org/10.3390/s20133625>
- [23] <https://www.amqp.org/about/what>
- [24] Bormann, C., Castellani, A., & Shelby, Z. (2012). COAP: an application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, 16(2), 62–67. <https://doi.org/10.1109/mic.2012.29>
- [25] D. Ugrenovic and G. Gardasevic, "CoAP protocol for Web-based monitoring in IoT healthcare applications," 2015 23rd Telecommunications Forum Telfor (TELFOR), Belgrade, Serbia, 2015, pp. 79-82, doi: 10.1109/TELFOR.2015.7377418

- [26] Mishra, B. (2018). Performance Evaluation of MQTT Broker Servers. In *Lecture Notes in Computer Science* (pp. 599–609). [https://doi.org/10.1007/978-3-319-95171-3\\_47](https://doi.org/10.1007/978-3-319-95171-3_47)
- [27] <https://mqtt.org/getting-started>
- [28] Fu, C., Liu, P., Zhu, J., Gao, S., Zhang, Y., Duan, M., Wang, Y., & Hwang, K. (2020). Improving Topic-Based Data Exchanges among IoT Devices. *Security and Communication Networks*, 2020, 1–14. <https://doi.org/10.1155/2020/8884924>
- [29] Koksal, Omer & Tekinerdogan, Bedir. (2017). Feature-Driven Domain Analysis of Session Layer Protocols of Internet of Things. 10.1109/IEEE.ICIoT.2017.19.
- [30] <https://www.wallarm.com/what/coap-protocol-definition>
- [31] (issaa.org) The Future of Enterprise IoT: Security Challenges and Opportunities
- [32] Robots.net | What Are The Main Challenges Of IoT
- [33] <https://www.rapid7.com/fundamentals/vulnerabilities-exploits-threats/>
- [34] <https://informationsecurity.wustl.edu/vulnerabilities-threats-and-risks-explained/>
- [35] <https://www.cloudflare.com/learning/ssl/why-use-tls-1.3/>
- [36] O. Ivanov, V. Ruzhentsev and R. Oliynykov, "Comparison of Modern Network Attacks on TLS Protocol," 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 2018
- [37] Fortinet | What is the CIA Triad and Why is it important?
- [38] | Cato Networks | 6 Network Security Protocols You Should Know
- [39] <https://www.rfc-editor.org/rfc/rfc9147.html>
- [40] G. Radhamani and K. Ramasamy, "Security issues in WAP WTLS protocol," IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions, Chengdu, China, 2002, pp. 483-487 vol.1, doi: 10.1109/ICCCAS.2002.1180664.
- [41] [https://www.techtarget.com/searchmobilecomputing/definition/Wireless-Transport-Layer-Security\\_WAP-261-WTLS-20010406-a.PDF](https://www.techtarget.com/searchmobilecomputing/definition/Wireless-Transport-Layer-Security_WAP-261-WTLS-20010406-a.PDF) (openmobilealliance.org)
- [42] L. Alqaydi, C. Y. Yeun and E. Damiani, "Security enhancements to TLS for improved national control," 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), Cambridge, UK, 2017, pp. 274-279, doi: 10.23919/ICITST.2017.8356398.
- [43] <https://datatracker.ietf.org/doc/html/rfc8446>

- [44] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," 2017 IEEE International Systems Engineering Symposium (ISSE), Vienna, Austria, 2017, pp. 1-7, doi: 10.1109/SysEng.2017.8088251.
- [45] P. Pierleoni, R. Concetti, S. Marzorati, A. Belli and L. Palma, "Internet of Things for Earthquake Early Warning Systems: A Performance Comparison Between Communication Protocols," in IEEE Access, vol. 11, pp. 43183-43194, 2023, doi: 10.1109/ACCESS.2023.3271773.
- [46] O. Ivanov, V. Ruzhentsev and R. Oliynykov, "Comparison of Modern Network Attacks on TLS Protocol," 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 2018, pp. 565-570, doi: 10.1109/INFOCOMMST.2018.8632026.
- [47] Hyunwoo Lee, Doowon Kim, and Yonghwi Kwon. 2021. TLS 1.3 in Practice:How TLS 1.3 Contributes to the Internet. In Proceedings of the Web Conference 2021 (WWW '21). Association for Computing Machinery, New York, NY, USA, 70–79. <https://doi.org/10.1145/3442381.3450057>
- [48] Mrabet H, Belguith S, Alhomoud A, Jemai A. A Survey of IoT Security Based on a Layered Architecture of Sensing and Data Analysis. *Sensors*. 2020; 20(13):3625. <https://doi.org/10.3390/s20133625>
- [49] Sohel Rana, Saddam Hossain, Hasan Imam Shoun and Dr. Mohammad Abul Kashem, "An Effective Lightweight Cryptographic Algorithm to Secure Resource-Constrained Devices" International Journal of Advanced Computer Science and Applications(IJACSA), 9(11), 2018. <http://dx.doi.org/10.14569/IJACSA.2018.091137>
- [50] Fu Chen, Peng Liu, Jianming Zhu, Sheng Gao, Yanmei Zhang, Meijiao Duan, Youwei Wang, Kai Hwang, "Improving Topic-Based Data Exchanges among IoT Devices", Security and Communication Networks, vol. 2020, Article ID 8884924, 14 pages, 2020. <https://doi.org/10.1155/2020/8884924>
- [51] Wu H, Chen C, Weng K. An Energy-Efficient Strategy for Microcontrollers. *Applied Sciences*. 2021; 11(6):2581. <https://doi.org/10.3390/app11062581>
- [52] Yüksel ME. A Power Consumption Analysis for Wi-Fi IoT Devices. *Electrica*, 2020; 20(1): 62-71.

- [53] Prodanović R, Rančić D, Vulić I, Zorić N, Bogićević D, Ostojić G, Sarang S, Stankovski S. Wireless Sensor Network in Agriculture: Model of Cyber Security. *Sensors*. 2020; 20(23):6747. <https://doi.org/10.3390/s20236747>
- [54] M. Babiuch, P. Foltýnek and P. Smutný, "Using the ESP32 Microcontroller for Data Processing," 2019 20th International Carpathian Control Conference (ICCC), Krakow-Wieliczka, Poland, 2019, pp. 1-6, doi: 10.1109/CarpathianCC.2019.8765944.
- [55] [https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf)
- [56] <https://datasheetspdf.com/pdf/785590/D-Robotics/DHT11/1>
- [57] Placidi P, Gasperini L, Grassi A, Cecconi M, Scorzoni A. Characterization of Low-Cost Capacitive Soil Moisture Sensors for IoT Networks. *Sensors*. 2020; 20(12):3585. <https://doi.org/10.3390/s20123585>
- [58] <https://docs.arduino.cc/learn/starting-guide/the-arduino-software-ide>
- [59] Bashari Rad, Babak & Bhatti, Harrison & Ahmadi, Mohammad. (2017). An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security*. 173. 8.
- [60] Vahdati, Zeinab & Ghasempour, Ali & Salehi, Mohammad & Md Yasin, Sharifah. (2019). COMPARISON OF ECC AND RSA ALGORITHMS IN IOT DEVICES. *Journal of Theoretical and Applied Information Technology*. 97. 4293.
- [61] <https://www.wireshark.org/docs/relnotes/wireshark-4.0.4.html>
- [62] <https://meters.uni-trend.com/products/digital-multimeters/>
- [63] Evans, R. O., Sneed, R. E., & Cassel, D. K. (1991). Irrigation scheduling to improve water- and energy-use efficiencies. NC Cooperative Extension Service.
- [64] Scherz, P. (2006). *Practical Electronics for Inventors 2/E*. McGraw Hill Professional.