REPUBLIC OF TURKEY
ISTANBUL GELISIM UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

Department of Electrical-Electronic Engineering

DESIGN AND IMPLEMENTATION OF AN INDUSTRIAL INTERNET OF THINGS BASED SCADA SYSTEM: CASE STUDY THE PETROLEUM PIPELINE SYSTEMS

Master Thesis

Omar Mohammed Mahdi AL MUBARAK

Supervisor

Asst. Prof. Dr. Musaria Karim Mahmood

Co-Supervisor

Asst. Prof. Dr. Ahmed Amin Ahmed Solyman

Istanbul – 2022

THESIS INTRODUCTION FORM

Name and Surname	**:**	Omar Mohammed Mahdi AL MUBARAK
Language of the Thesis	**:**	English
Name of the Thesis	**:**	Design and Implementation of an Industrial Internet of Things based SCADA system: case study the petroleum pipeline systems
Institute	**:**	Istanbul Gelişim University Graduate Education Institute
Department	**:**	Electrical-Electronic Engineering
Thesis Type	**:**	Post Graduate
Date of the Thesis	**:**	13/04/2022
Page Number	**:**	108
Thesis Supervisors	**:**	Asst. Prof. Dr. Musaria Karim Mahmood Co-Supervisor: Asst. Prof. Dr. Ahmed Amin Ahmed Solyman
Index Terms	**:**	Internet of Things (IoT), Supervisory Control and Data Acquisition (SCADA) Message Queuing Telemetry Transport (MQTT), Laboratory Virtual Instrument Engineering Workbench (LabVIEW), Node-RED, Arduino UNO
Turkish Abstract	**:**	Bu teklif, Midstream tatbikatlarında petrol boru hattı çerçevesinin uzaktan gözlemlenmesi için kontrol ve bilgi edinme için toplam bir çerçevenin ilerlemesini sunar. Bu çerçeve, güçlü kontrol ve kontrol için geleneksel çerçeveleri web yönetimleriyle koordine eden Nesnelerin İnterneti'nin (IoT tabanlı SCADA) SCADA mühendisliğine bağlıdır. Sıcaklık, basınç, akış hızı ve gaz ve yangın tanıma gibi bilgisayarlı sensörlerden oluşur.

Bir Arduino Uno, açık kaynak ile programlanmış bir veri toplama işlemcisi olarak çalışır; sırayla, bu bir uzak terminal birimini çalıştırır. Node-MCU, içerdiği Wi-Fi teknolojisi nedeniyle kablosuz iletişim kanalı olarak kullanılmaktadır. Ayrıca web sitesindeki (Ana Terminal Birimi) sensörlerden alınan veri okumalarının izlenmesi, kontrolü ve saklanması için bir Node-Red araçları programlama platformu da bulunmaktadır. Sensörlerden elde edilen verileri almak için LabVIEW grafik dili ortamı kullanılmaktadır. Boru hatlarını izlemek için önerilen SCADA sisteminde MQTT protokolü kullanılmaktadır.

MQTT protokolü, veri iletimleri ve salınımlar açısından HTTP protokolü ile karşılaştırılmıştır, burada sonuçlar MQTT protokolünün HTTP protokolünden daha az gecikme ve salınımlara sahip olduğunu göstermektedir. paket boyutu açısından HTTP, MQTT protokolünden daha iyi performansa sahiptir.

Distribution List **:**

1. To the Institute of Graduate Studies of Istanbul Gelisim University
2. To the National Thesis Center of YÖK (Higher Education Council)

Omar Mohammed Mahdi AL MUBARAK

REPUBLIC OF TURKEY
ISTANBUL GELISIM UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

Department of Electrical-Electronic Engineering

DESIGN AND IMPLEMENTATION OF AN INDUSTRIAL INTERNET OF THINGS BASED SCADA SYSTEM: CASE STUDY THE PETROLEUM PIPELINE SYSTEMS

Master Thesis

Omar Mohammed Mahdi AL MUBARAK

Supervisor

Asst. Prof. Dr. Musaria Karim Mahmood

Co-Supervisor

Asst. Prof. Dr. Ahmed Amin Ahmed Solyman

Istanbul – 2022

DECLARATION

I hereby declare that in the preparation of this thesis, scientific ethical rules have been followed, the works of other persons have been referenced in accordance with the scientific norms if used, there is no falsification in the used data, any part of the thesis has not been submitted to this university or any other university as another thesis.

Omar Mohammed Mahdi AL MUBARAK

…/…/2022

TO ISTANBUL GELISIM UNIVERSITY

THE DIRECTORATE OF INSTITUTE OF GRADUATE STUDIES

The thesis study of Omar Mohammed Mahdi AL MUBARAK titled as DESIGN AND IMPLEMENTATION OF AN INDUSTRIAL INTERNET OF THINGS BASED SCADA SYSTEM: CASE STUDY THE PETROLEUM PIPELINE has been accepted as MASTER THESIS in the department of ELECTRICAL-ELECTRONIC ENGINEERING by out jury.

Director *Asst. Prof. Dr. Musaria Karim MAHMOOD*
(Supervisor)

Member *Asst. Prof. Dr. Ahmed Amin Ahmed SOLYMAN*

Member *Asst. Prof. Dr. AFM Shahen SHAH*

APPROVAL

I approve that the signatures above signatures belong to the aforementioned faculty members.

... / ... / 2022

*Prof. Dr. İzzet G*ÜMÜŞ
Director of the Institute

SUMMARY

This proposal presents the advancement of a total framework for control and information obtaining for remote observing of the oil pipeline framework in Midstream exercises. This framework depends on the SCADA engineering of the Internet of Things (IoT-based SCADA) that coordinates customary frameworks with web administrations for strong checking and control. It comprises computerized sensors of extents like temperature, pressure, stream rate, and gas and fire recognition. An Arduino Uno operates as a data acquisition processor., which is programmed with open source; in turn, this operates a remote terminal unit. Node-MCU is used as a wireless communication channel due to contains Wi-Fi technology. There is also a Node-Red tools programming platform for the monitoring, control, and storage of data readings acquired from sensors in the website (Master Terminal Unit). The LabVIEW graphic language environment is used to receive the data obtained from the sensors. The MQTT protocol is used in the proposed SCADA system built to monitor pipelines., where the operator can monitor the data through a computer connected to the Internet or through a mobile application.

MQTT protocol is compared with HTTP protocol in terms of data transmissions and oscillations, where the results show that the MQTT protocol has less delay and oscillations than HTTP protocol. in terms of size of packets, HTTP has better performance than the MQTT protocol.

Keywords: Internet of Things (IoT), Supervisory Control and Data Acquisition (SCADA) Message Queuing Telemetry Transport (MQTT), Laboratory Virtual Instrument Engineering Workbench (LabVIEW), Node-red, Arduino UNO.

ÖZET

Bu teklif, Midstream tatbikatlarında petrol boru hattı çerçevesinin uzaktan gözlemlenmesi için kontrol ve bilgi edinme için toplam bir çerçevenin ilerlemesini sunar. Bu çerçeve, güçlü kontrol ve kontrol için geleneksel çerçeveleri web yönetimleriyle koordine eden Nesnelerin İnterneti'nin (IoT tabanlı SCADA) SCADA mühendisliğine bağlıdır. Sıcaklık, basınç, akış hızı ve gaz ve yangın tanıma gibi bilgisayarlı sensörlerden oluşur. Bir Arduino Uno, açık kaynak ile programlanmış bir veri toplama işlemcisi olarak çalışır; sırayla, bu bir uzak terminal birimini çalıştırır. Node-MCU, içerdiği Wi-Fi teknolojisi nedeniyle kablosuz iletişim kanalı olarak kullanılmaktadır. Ayrıca web sitesindeki (Ana Terminal Birimi) sensörlerden alınan veri okumalarının izlenmesi, kontrolü ve saklanması için bir Node-Red araçları programlama platformu da bulunmaktadır. Sensörlerden elde edilen verileri almak için LabVIEW grafik dili ortamı kullanılmaktadır. Boru hatlarını izlemek için önerilen SCADA sisteminde MQTT protokolü kullanılmaktadır.

MQTT protokolü, veri iletimleri ve salınımlar açısından HTTP protokolü ile karşılaştırılmıştır, burada sonuçlar MQTT protokolünün HTTP protokolünden daha az gecikme ve salınımlara sahip olduğunu göstermektedir. paket boyutu açısından HTTP, MQTT protokolünden daha iyi performansa sahiptir.Ayrıca, web sitesinde sensör okumalarını görüntülemek için kullanıcı arayüzü oluşturmak için Node-Red yazılım aracı kullanıldı.

Anahtar Kelimeler : Nesnelerin İnterneti (IoT), Denetleyici Kontrol ve Veri Toplama (SCADA) Mesaj Kuyruklama Telemetri Taşımacılığı (MQTT), Laboratuvar Sanal Enstrüman Mühendisliği Tezgahı (LabVIEW), Node-red, Arduino UNO.

TABLE OF CONTENTS

CHAPTER THREE

METHODOLOGY

CHAPTER FOUR

SIMULATION MODULE

CHAPTER FIVE

IMPLEMENTATION AND SIMULATION RESULTS

CHAPTER SIX

CONCLUSIONS AND FUTURE WORK

ABBREVIATIONS

SCADA	**:**	Supervisory Control and Data Acquisition
IoT	**:**	Internet of Things
IIoT	**:**	Industrial Internet of Things
PLC	**:**	Programmable Logic Control
RTU	**:**	Remote Terminal Unit
MTU	**:**	Master Terminal Unit
MQTT	**:**	Message Queue Telemetry Transport
HTTP	**:**	Hypertext Transfer Protocol
CoAP	**:**	Constrained Application Protocol
AMQP	**:**	Advanced Message Queuing Protocol
XMPP	**:**	Extensible Messaging and Presence Protocol
TCP/IP	**:**	Transmission Control Prot. /Internet Protocol
HMI	**:**	Human- Machine Interface
UDP	**:**	User Datagram Protocol
LAN	**:**	Local Area Networking
WAN	**:**	Wide Area Networking
SQL	**:**	Structured Query Language
NoSQL	**:**	Not Only Structured Query Language
QoS	**:**	Quality of Service
SSL	**:**	Secure Sockets Layer
M2M	**:**	Machine to Machine
API	**:**	Application Programming Interface

LIST OF TABLES

LIST OF FIGURES

LIST OF ANNEXES

PREFACE

During the preparation and writing process of this thesis, I would like to thank my esteemed professor Asst. Prof. Dr. Mussaria K. MAHMOOD. I also thank Asst. Prof. Dr. Ahmed Amin Ahmed Solyman for the support. I would like to thank the thesis's jury members for their help in managing this thesis and taking it forward with their valuable comments and suggestions throughout the process.

Finally, I want thanking my family for their supporting..

INTRODUCTION

Upstream, midstream, and downstream are the three sections of the petroleum industry. Raw petroleum is a fundamental feedstock for the creation of refined oil-based commodities. As indicated by one insights entry, produce more than four billion metric huge loads of oil worldwide consistently. The best three makers being Saudi Arabia, the United States, and Russia. The worldwide interest for oil and gas items keeps on rising. From 89 million barrels in 2012, the everyday worldwide oil utilization might ascend to 109 million barrels in (Nasir, Naidoo and Amoo, 2018). The expanding interest for oil implies that organizations in the oil business need to keep awake to-date on the most recent patterns and keep on working on their cycles, which will empower them to all the more likely oblige their purchasers, and increment income. While over-designed weighty steel has gotten to where are today, computerization is the following stage in the development of the oil and gas industry (Osabutey et al, 2013). No area of this industry is that clearer than in midstream. The midstream area of the oil and gas industry is characterized by the transportation and capacity of oil-based goods. It is essentially interfacing the different phases of industry and makes it workable for oil and gaseous petrol to get from the wellhead to its last objective and each point in the middle. In addition, the midstream market expects organizations to have clear hierarchical cycles that permit fast reaction to the fluctuating requests of the market (Osabutey et al, 2013). Midstream, which essentially incorporate the capacity and transportation of oil-based goods, happen after the exploration and production period of raw petroleum and flammable gas activities. The transportation of rough and refined items, just as oilfield side-effects, all throughout the planet, influences each part of the worldwide economy (Shi W., Yang Z. and Li K., 2013). Consistently, the midstream area assembles miles of new pipelines to oblige the expanding need to ship oil, gas and water from the upstream area. The developing utilization of robotization apparatuses and measures has midstream area more as compared to any instant in recent memory. Robotization gives different answers for oilfield pipelines and other transportation techniques. Computerization presents keen and associated hardware that support pipeline usefulness and benefit. Computerized advancement, then, at that point, gives the accompanying chances to pipeline organizations (Chen, A., and Dong, H., 2021):

1) Real-time remote observing
2) Complete functional control
3) Optimized energy and support costs
4) Better control room the executives
5) Easy admittance to knowledge
6) Reduced personal time

Overseers are right now more open to the IoT - Internet of Things, which is a game plan of figuring contraptions, electromechanical machines, and articles that engage data travel through an association without the necessity for human-to-human or human-to-machine correspondence (Chopra et al, 2019).

Generally speaking, a Supervisory Control and Data Acquisition (SCADA) System is a system that allows the user to collect data from remote locations in order to control equipment under a variety of conditions or to transmit limited monitoring instructions to the facility. The main function of SCADA is to obtain data from remote devices such as valves, batteries, pumps, sensors, transmitters, etc. It provides comprehensive monitoring and remote control, via the SCADA host platform (Lu X., 2014). The ability to oversee operations in a petroleum pipeline system may be required. It is critical to have coordinated control that is reliable, flexible, cost-effective, and sophisticated. In such a situation, the SCADA system is the best option (Trung, 1995). The system allows monitoring of remote sites without the necessity of staying in these sites or visiting them repeatedly when the process is operating normally. The SCADA system mainly has several basic functions which are data acquisition, data display, remote monitoring, supervisory control, and network data communication (Sayed and Gabbar, 2017). Field instrumentation, RTUs, MTUs, and a communication route connecting MTUs and RTUs are all essential components of the system (Xie Lu, 2014). SCADA software and hardware are generally classified into two broad categories; Open source and proprietary (Rakas, Stojanović and Marković-Petrović, 2020). An open-source system allows the user to match and mix components, and to choose the most suitable from among many suppliers. Thus, there is no single supplier with an open-source system responsible for the overall performance of the system. Whereas in a proprietary system, all components are from one manufacturer and standards are usually specific to this system. These components are only developed by

the manufacturer. Which puts the user vulnerable to a single supplier or manufacturer, because the supplier or factory may be slow to respond to the technological changes that occur in a subsystem of the SCADA system. Also, in the event that the manufacturer or supplier stops, the customer will also be exposed to a risk and the solution will be costly. In addition to another problem which is the problem of flexibility with the network and the existing devices. An open-source system is also a more efficient solution because the user is not beholden to only one manufacturer or supplier. In an Open-source system, the major components of the SCADA system adhere to certain standards that allow them to be replaced with other similar components manufactured by other factories with the same standards and specifications (Aghenta and Iqbal, 2019).

Therefore, an open-source SCADA system has been suggested in this thesis. The proposed structure contains Arduino-UNO microcontroller, Various sensors, and Espressif's ESP32 micro-controller as a communication channel between the system and the cloud. The acquired data is transferred from the sensors to the cloud, and the cloud, in turn, displays the acquired data remotely via the web in the monitor screens. By utilizing the Message Queuing Telemetry Transport (MQTT) library inside LabView, test systems viable with the convention are created and framework control is performed.

CHAPTER ONE

PURPOSE OF THE THESIS

1.1. Literature Survey

All over the world research communities have tried to solve the problems related to commercial SCADA systems (compatibility issues, high costs) by developing an open-source SCADA system, where each has different functions and costs.

(Mahmood, M. K. and Al-Naima, F., 2011) have presented An Internet-based Distributed Control System (DCS) enabling real-time data monitoring using a standard web browser. The system is applied for petroleum application (refineries), and has a structure similar to classical SCADA. Each DCS will be linked to the system web server, which will be located on the central server. The suggested system has a three-tier client-server architecture. C#.Net is used to create the application server. All of the DCSs' local database servers, as well as the system database server, run on SQL Database Server 2005. The suggested system includes real-time system alarm monitoring as well as all previous records.

(Mahmood, M., Al-Naima and Uzunoglu, N., 2012) explains how to construct and simulate a dependable data transmission network for a proposed SCADA system. A dependability study is conducted on real-time data communication networks with high load File Transfer Protocol (FTP), database access, E-mail, and Voice over IP (VoIP) with PCM quality application. In the simulation, numerous models are explored and tested against a variety of scenarios (high load, utilization factor of links, network topological change).

Vijeo Citect SCADA software is used as the Master Terminal Unit in the open- source operating system proposed by (Avhad, Golatkar, and Joshi, 2013). (MTU). In addition to the Modbus protocol, which offers a communication channel between the MTU and RTU, the AtMEGA 2560 microcontroller functions as a Remote Terminal Unit (RTU) to gather sensor data.

According to (Kodali and Soratkal, 2016), the lightweight MQTT protocol has been described. With MQTT running on the development board, they've created a prototype that works over Wi-Fi. For remote monitoring and control, sensors and

actuators are connected to ESP8266 and a Mosquitto-based MQTT broker is constructed. A cloud platform can be used to collect, analyze, and present data. It is possible to create a customized GUI for monitoring and controlling devices from a distance.

An open-source, low-cost SCADA system has been suggested by (Rajkumar, R.I., et al 2016). An Arduino microcontroller is used as a sensor gateway, and radio units (ZigBee) to transmit data from sensors such as temperature, level, flow sensors and control valves ... etc.

A weather monitoring system have been suggested by (Rao et al, 2016). According to their research, it is possible to monitor the weather conditions in a specific location from anywhere in the world using the Internet of Things. This system is only for surveillance, as there is no control part in it. A set of sensors are used, such as a flame sensor, a temperature sensor, a pressure sensor, etc. Also, Arduino microcontroller is used as part of the system. The SCADA system was first introduced to the electrical network distribution service for monitoring in load management and control and detection of malfunctions to give the appropriate solution or the appropriate option, and then this system was used on oil and gas pumping systems and pipelines.

(Rao et al, 2016) have Proposed a novel system to reduce the dependency on a specific SCADA structure. For example, for a power system consisting of applications of a combination of ocean waves-winds energy source, they propose an open source IoT-based SCADA system applicable to remote access for data inspection, analysis and control.

The upstream area has for some time been an ally of robotization in day-by-day boring reviews, investigations, climate observing frameworks, and pressing factor and stream estimation (Shoja S. and Jalali, 2017). The midstream area can likewise rely upon the flexibility of computerization to have a quick and enduring effect in everyday tasks. The utilization of mechanized instruments and cycles like the IoT and SCADA frameworks viably tackle a wide scope of ordinary issues across different industry portions.

An open-source SCADA system has also been suggested by (Mononen and Mattila, 2017). It is a low cost and consists of a microprocessor to process data and

transfer it to the cloud via the User Datagram Protocol (UDP), and Ethernet frame. As the cloud environment consists of virtual machines that operate user-defined algorithms to process data with great precision.

(Kao, Chieng, and Jeng, 2018) they have presented an Internet of Things (IoT) web application for an intelligent remote SCADA system for electrical power. It contains a wireless sensor network for capturing signal data and receiving information from various objects. – (sensor-nodes). Intelligent gadgets can be communicated using this technology. With a browser on their computer or smart device, users may access and monitor the system remotely. This technology is used for remote control and monitoring of inverters with basic features. In spite of the fact that it is still in its infancy, the system has a great deal of room for expansion. Autonomous automobiles, intelligent boats, and personal rapid transit all have the potential to benefit from this technology, which can be implemented in a variety of ways in the future. In order to ensure effective power management, the measurement data is kept in a database.

Using Internet of Things (IoT) technologies, (Medrano et al, 2018) have developed a low-cost SCADA system which uses TCP/IP to connect field devices to the RTU and relies on a NoSQL database for support. Using the traffic management process, the functionality of their developed system was evaluated. The main problem with such systems is the difficulty of use by the average end user.

An Open-source SCADA platform called OpenSCADA has been used by (Prokhorov et al, 2018) to monitor the flow rate and water level in a container utilizing an HMI (Human-Machine-Interface) to show data. Due to subscription fees, OpenSCADA isn't free for everyone, and the open system necessitates a significant amount of coding in order to limit access to more precise data within the OpenSCADA platform.

To better understand IoT security architecture, (Gupta and Quamara, 2020) compiled a taxonomy of main problems in the industry and essential technologies including Radio Frequency Identification (RFID) and Wireless Sensor Networks (WSN). In the development of the Internet of Things, RFID and WSN are key enablers. Additionally, the article addresses open-source tools and platforms that can be used to construct IoT infrastructure. Lastly, a quick summary of the significant open challenges, together with their possible answers and future research areas, is presented.

According to the proposal (Phuyal S. et al, 2020), is possible to design a system that serves as a supervisory control to obtain the acquired data, and this system is called SCADA. The standard used is IEEE C37.1. This system allows monitoring and control of various devices in the field of industry.

To locate the pipeline and assess data visualization, (Wu, Q. et al, 2020) have presented Asynchronous JavaScript and XML (AJAX) technology using the Baidu map Application Programming Interface (API). For oil pipeline monitoring, an IoT-based real-time visualization approach is presented. Using this strategy, we were able to quickly locate and monitor the pipeline in real-time, overcoming the challenge of low data readability.

(Chen, A., and Dong, H., 2021) have presented a new approach to increase pipeline efficiency by developing a real-time pipeline detection and control system based on the IoT platform. On the pipeline, detecting nodes and control valves are installed, and the IoT platform is utilized to collect and process data in order to remotely monitor pipeline network operations.

(Aba E. et al, 2021) demonstrated how to employ an IoT analytics platform service to simulate real-time pipeline monitoring and pinpoint the site of pipeline damage. In this study, pressure pulses based on the notion of pipe vibration are used for pipeline monitoring. In this work, the principle of the temporal delay between pulse arrivals at sensor sites is also used. A wireless communication device is created by combining an Arduino and a Wi-Fi module, programming it, and connecting it to the ThingSpeak internet of things (IoT) analytics platform. On the platform, five channels were constructed to collect data from the five sensors utilized in the experimental test of the vibrating gadget that utilized wireless communication devices.

(Puviarasi et al, 2021) have proposed a Global System for Mobile Communications (GSM900A) kit to notify supervisors of any installation problems. Raspberry Pi (RPi) takes tentative action in advance to prevent water loss due to leaks. To track each kit, many packages are deployed around the pipeline, including a Raspberry Pi, a pressure sensor, and a consistency sensor. One portable kit is designed to calculate the specific location of leakage between two observatory kits, allowing for the detection of a ground crack or leakage without the need to dig.

(Maheswari, et al 2021) have suggested using IoT to monitor temperature in several businesses. IoT is primarily supported by its communication protocols, which connect it to both the industry and the server. This component is made to be very important in all sectors. The numerous protocols used in IoT are compared in this article. Based on past research, it appears that different sectors use protocols based on their requirements. As a result, it is inappropriate to impose any approach, but they are free to do so. The study indicates that for IoT systems, it is now desirable to employ a Long-Range Wide Area Network. The Low Power Radio Module (LoRa) is the most widely utilized IoT communication network for data collection in remote areas with inadequate signal strength. In the future, cost-effective remote monitoring will be achieved by carefully picking the cloud access provided by various suppliers.

A highly portable, compact, and power-efficient integrated smart IoT module with a System on Chip design have been proposed by (Priyanka, Maheswari, and Thangavel, 2021). The introduction of the integrated IoT paradigm provides a better foundation for improving monitoring capabilities through the use of virtual and embedded-based field sensors. On the compact platform, multiple sensors are interfaced and communicated efficiently with the cloud server, resulting in reduced calculation time and higher throughput. Even in high-risk situations, it provides proactive smart monitoring and control in the oil pipeline lab-scale setting. As a result, the IoT application, using the built IoT smart module, makes a crucial decision by sending warning message alerts signposts to appropriate parameter labels in the web monitor to the supervision engineer to remedy before the situation deteriorates to its worst state.

(Priyanka, Thangavel, and Gao, 2021) have proposed leveraging Smart Grid (SG) to implement Cloud-based communication platforms. In the realm of SG monitoring and control systems, new communication technologies have the potential to make huge advances. Because traditional distribution networks are projected to become more observable in the near future, a substantial volume of measurement data will necessitate more efficient communication structures. With simulation analysis on the flex meter platform, a special analysis of chosen SG functions revealed two main communication topologies for power monitoring of oil pipeline substations with smart pigs. The proposed SCADA and Cloud-based system's theoretical concept.

(Chavala L., 2022) have suggested architecture for monitoring various metrics of oil pipeline management utilizing IoT and Lora WAN employing a LoRa sensor node and gateway customized boards. They simulated the Long-Range (LoRa WAN) using the OPNET simulator and examined characteristics such as Signal to Noise Ratio (SNR), Time on-Air (ToA), sensitivity, output power, Received Signal Strength Indicator (RSSI), and packet error rate. According to the findings of this investigation, the most severe correspondence range in this environment is roughly 4.8 km. With increasing payload size, Time on-Air (ToA) increases. When the distance is increased up to 3 km, the SNR falls, then shows a net reduction after 3 km.

1.2.Problem Statement

The midstream area utilizes pipelines to ship unrefined petroleum and flammable gas from drill locales to capacity and handling office. These pipelines ordinarily length a large number of miles over various sorts of brutal landscapes. Thus, midstream administrators might require a focal SCADA framework to screen and deal with all field instruments used across the whole oil or gas stream in the lines. Suction stations control the pressure factor passing through the oilfield pipelines. Also, midstream administrators check the entire cycle if occur any problem in the sector. Administrators can recognize, decide, and resolve issues like releases and harms in these lines by sending an extendable remote organization across siphon or blower stations, metering stations, and valve stations.

A SCADA framework gives successfully in monitoring and controls pipelines. SCADA system is prestigious for effectiveness in many fields, information correspondence across the organizations, precise information announcing, and smoothed out framework control capacities. Accordingly, they are incredible supporters of things to come of midstream mechanization.

As administrators are now more open to the Internet of Things that can fill as the basis for current mechanical activities. It offers approach to get information assortment and sharing it among various locations. Thusly, an IoT-based SCADA framework is especially advantageous in the oil and gas pipeline industry because of its capacity to advance estimation and information investigation, continuously, increment data transmission for business examination, screen measures, record occasions and updates in a log document, and cooperate with different midstream gadgets like sensors, valves, siphons, and engines.

1.3. Study objectives

The objectives of this thesis are:

1- To design a modern SCADA framework dependent on the Internet of Things to monitor pipelines in midstream sectors
2- To use open-source resources through an Arduino-based microcontroller with a set of commercially calibrated digital sensors through a graphic interface LabVIEW using an MQTT communication protocol.
3- The collected data will be displayed on a website using the Node-Red programming tool, also displayed through the phone application.

1.4. Thesis Organization

The thesis is composed of six chapters.

- **The first chapter** gives a straightforward prologue to the proposed framework and incorporates general and important data, an issue proclamation and the target of the theory.
- **The second chapter** includes the hypothetical foundation and the functioning strategies of the open-source SCADA IoT system and includes the working methods and characteristics of MQTT and HTTP Protocols.
- **The third chapter** includes the methodology, general structure of the proposed system, and the hardware setup and data acquisition of the proposed system with a MQTT protocol.
- **The fourth chapter** includes the simulation module and data acquisition of the proposed system with MQTT and HTTP Protocols.
- **The fifth chapter** includes the implementation and simulation results.
- **The sixth chapter** includes conclusions and future work concerning a continuation of this work.

CHAPTER TWO
THEORETICAL BACKGROUND

2.1. SCADA System

Control and data acquisition systems are referred to collectively as SCADA systems. Remote monitoring and control are made possible through the use of coded signals sent over communication channels. It can be used in combination with a data acquisition system to acquire information about the remote equipment's status, either for display or recording purposes. Control systems for industrial processes are a common application for this technology (Khera and Balguvhar, 2011). Automation systems for SCADA are typically used to supervise-control complex industrial processes where human intervention is impractical, such as systems with large control factors or move faster than humans can handle (Goel and Mishra, 2009). SCADA frameworks are adequately amazing to deal with the mind-boggling measures needed to work in the new oil world. These controlled frameworks give a stable and practical answer for the requirements of the oil and gas area. Organizations go to SCADA to manage activities, further develop proficiency and limit vacation. In addition, they depend on the framework for guidance ahead of time and effective closure instruments, assisting the executives with making purposeful and key activities to moderate harm (Stouffer and Scarfone, 2011). Affiliations responsible for working in all the levels of the creation cycle upstream, most of the way and downstream will find a usage for state-of-the-art data systems. A fast once-over of how SCADA systems support oil and gas industry/associations gives a blueprint of the benefits to these and various associations and shows how it is applied to each level out of action. Hardware architecture consist of SCADA structures which are made out of a couple of sub modules, the most critical of which are RTUs and PLCs. RTUs go about as information focused. Associated with sensors on location gear, they convert sensor indications to computerized information. They utilize telemetry equipment, which shares and sends this advanced information to the administrative framework. On the opposite end, RTUs get computerized orders from the administrative framework and send them to the end gadgets. PLCs screen tweaked limits, and show liquid levels, gas meters, voltage, ampere values, pressure, temperature, clamminess, thickness, volumetric stream rate, and mass. Sometimes, PLCs sub for RTUs as field contraptions since they are more reasonable and adaptable. It is becoming increasingly common for SCADA

systems to be replaced by Industrial Processes Control (IPC) that employ PLCs. IPCs are used for measure control, data getting, & assessment. Not in the slightest degree like PLCs, IPCs limits can be revived remotely through server-based control structures. This degree of organization considers brief change and climbs to the current control system. While PLCs are the industry standard for a long time, they aren't "splendid" technology like IPCs, which recommend advanced features and the ability to invigorate their programming through the server. Telemetry frameworks partner PLCs and RTUs to the rule control concentration and data stockrooms. They work via wired or remote system. SCADA systems use wired telemetry media to connect leased telephone lines and WAN. Their remote accomplices, meanwhile, join satellite, radio and microwaves. Human Machine Interface (HMI) is a dashboard or user interface that connects a human with a machine, system, or device. When used to describe a screen that allows a user to interact with an electronic device, the term HMI is more often associated industrial processes. Servers are dedicated to the collecting and management of data for a large number of cut-off points. Some handle alerts, while others collect data. Workers obtain information, for example, partner programming associations with RTUs, PLCs, and IPCs via telemetry. (Shahzad, Musa and Irfan, 2014).

The purpose of SCADA are:

1) Data Acquisition
2) Data presentation
3) Monitoring
4) Controlling
5) Alarming

SCADA systems have been developed in four different generation:

2.1.1. 1st Generation: “Monolithic”

In the early 1970s, "mainframe" computer systems dominated the computing landscape. A single computer system is common practice at the time because it handled all of the computing tasks. As depicted in figure 1, networks are non-existent,

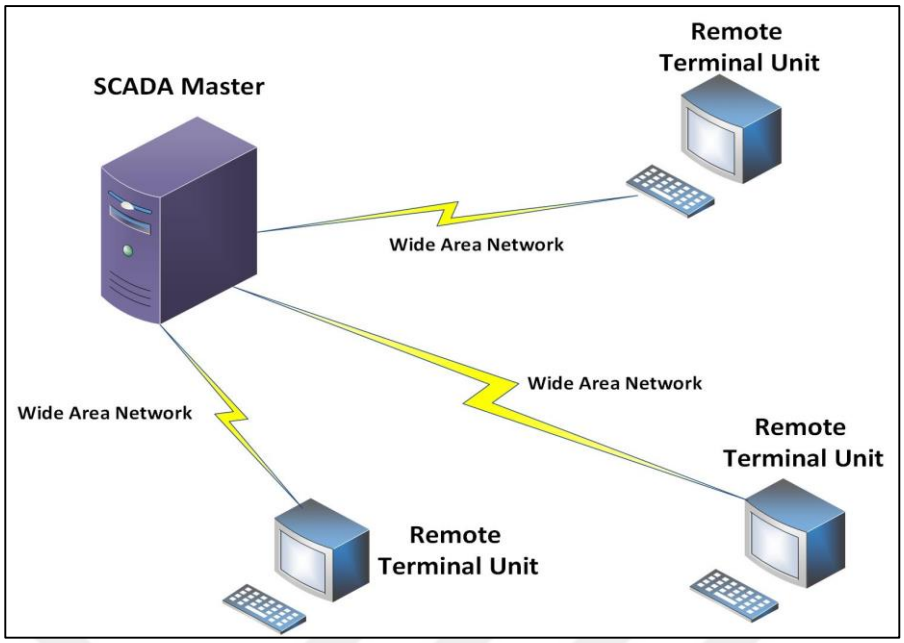

Figure 1. Monolithic SCADA Architecture

And the various centralized systems each operated on their own. As a result, SCADA systems are created to operate independently of other systems.

The WANs used to connect to remote RTUs are built with a single purpose in mind, which is connecting remote RTUs. In addition, the current WAN protocols are unknown at the time. In SCADA networks, the protocols used by RTU manufacturers to communicate with each other are often considered proprietary. This meant that no other vendors are allowed to develop equipment that communicated with RTUs using these protocols for certain RTU protocols. Furthermore, these protocols are typically "sparse" with little functionality beyond scanning for and controlling points on the remote device's interface. As a result, in most cases, other types of data traffic cannot be combined with RTU communications on the same network.

There are significant restrictions on access to the SCADA master station. To connect to itself when there is no network connectivity, a proprietary adapter or controller is plugged into the CPU backplane and used to make the connection. Again, only the system manufacturer's connectivity options are available.

System reliability, failover, and redundancy are achieved by connecting two identical mainframe systems via a bus in these generation systems. Typically, one system is designated as the primary system and the other as the standby system. It is the primary role of the standby system to monitor the primary system and assume control in the

event of a failure. In this type of standby operation, the standby system does not perform any processing. (McClanahan R. H., 2002).

2.1.2. 2nd Generation "Distributed"

SCADA systems of the next generation are able to distribute data more efficiently because of new developments in system miniaturization and LAN technology. LAN connects multiple computers, each with a specific function, to share real-time data, as shown in figure 2. There are workstations that use minicomputers, not mainframes, and are better and cheaper than predecessors from the first generation. For example, communications processors on some of these distributed stations are primarily used to communicate with field devices like RTUs. Several of these devices, which are used as operator interfaces, provided the HMI for system administrators.

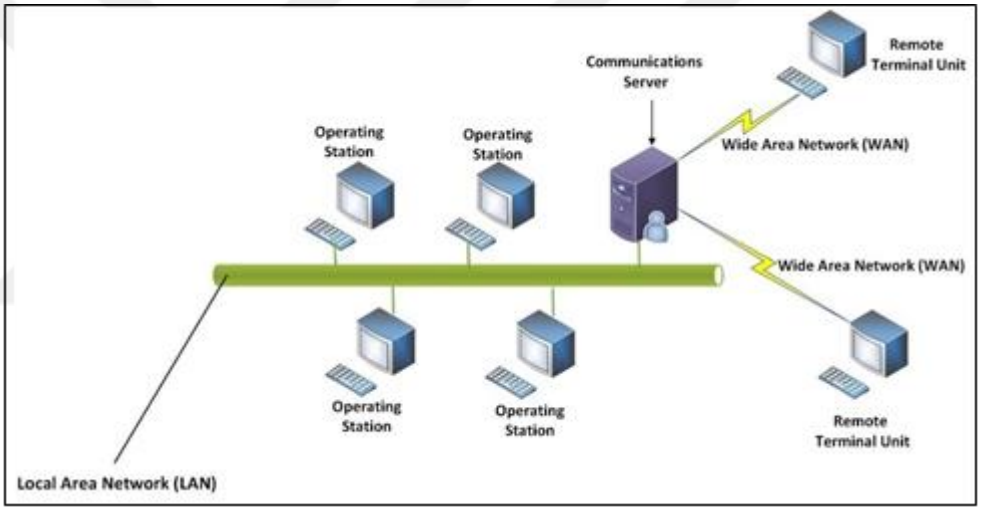

Figure 2. Distributed SCADA Architecture

Others served as database servers or computation processors for the rest of the team. Rather than relying on a single processor, the system's overall processing power is increased by distributing the SCADA system's functions across multiple systems. Networks that use LAN protocols to connect these systems can't communicate with anything outside of the local area they're connected to. This severely restricts what can be accomplished with a LAN. Real-time traffic can be optimized for the LAN protocol by a vendor. Network devices from other vendors can only be connected if they are restricted (or effectively eliminated) from doing so on the SCADA LAN. With the network-connected systems, the system's redundancy and reliability are increased. For SCADA systems of the second generation, vendor-supplied hardware and software,

as well as peripheral devices such as cameras, microphones and printers are required, the first generation of systems had this problem. (Yadav and Paul, 2021).

2.1.3. 3rd Generation "Networked SCADA "

An open system architecture rather than a vendor-controlled environment is the most significant difference between the third- and second-generation SCADA master station architectures.

Several other interconnected systems share the responsibilities of the central master station. Small numbers of remote terminals still use vendor-proprietary protocols. In the third generation, the most significant improvement is the ability to distribute SCADA functionality across a WAN rather than just a LAN, as shown in figure 3. Standards-based SCADA systems have overcome several shortcomings in previous generations of SCADA systems. Off-the-shelf systems make it easier for users to connect 3rd-party peripheral devices (such as CRTs, printers, disk drives, tape drives, and so on) In favour of "open" or "off-the-shelf" systems, SCADA vendors have largely given up on hardware development. These companies have relied on the operating system development expertise of companies like Compaq, Hewlett-Packard, and Sun Microsystems. The result of this is that SCADA software providers can concentrate their efforts on the SCADA master station software, where they can add the most value to their system. It is possible that open networks and established standards may not provide a solution to all of the problems associated with closed or proprietary systems. Even if all systems and equipment adhere to the same standards and are open, interoperability problems can still occur. WAN protocols (such as IP) are widely used in third-generation SCADA systems to communicate with communications equipment. This makes it possible to separate the "proper" communications of the master station with field devices (Yadav and Paul, 2021).

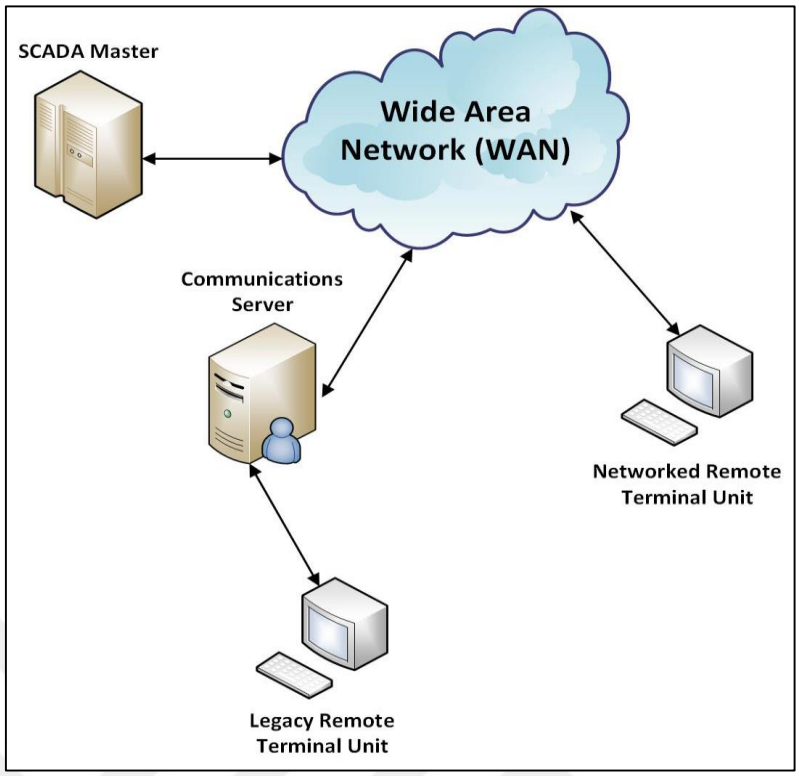

Figure 3. Networked SCADA Architecture

2.1.4. 4th Generation "IoT or (Industry 4.0)"

Businesses use the power of technology to build, monitor, and control. IoT innovation and affordable cloud computing with SCADA systems have greatly reduced the costs of its infrastructure and deployments. This is by using the Internet as communication backbone and system infrastructure. Furthermore, the integration and maintenance of the new generation are much easier than in the past due to the standardized protocols.

Industrial Internet of Things (IIoT) or Industry 4.0 refers to advancements in SCADA systems of the fourth generation. It is a network of devices with a primary focus on the transfer, control, and analysis of critical information using the Internet. As a result, a number of devices and protocols must be added to SCADA in order to implement IIoT. Figure 4 depicts a fourth-generation SCADA system for Industry 4.0. This includes distributed cognitive computing, Cyber-Physical Systems, IoT, and cloud computing. SCADA systems already include IoT features such as data access, manipulation, and visualization. Interoperability, scaling and big data analytics all differ in the IoT. All data is gathered and managed through the use of an open communication standard and protocols. Using cloud storage and data extraction, the

collected information can be mined for new large insights. Through the use of data-driven techniques, IIoT has been able to better identify anomalous behaviour and thereby increase its resilience. It's difficult to plan for critical infrastructures downtime. Predictive maintenance, on the other hand, allows for a reduction in downtime. Since the 1970s, these systems, which have evolved from a monolithic architecture to an IoT-based SCADA, have relied on effective information communication to collect, analyse, and display data from various heterogeneous devices over a wired or wireless network. A lack of homogeneity in terms of technologies, protocols, and proprietary architectures makes interoperability more difficult, more expensive, and more prone to failure (Feng et al 2019) (Yang et al, 2014).

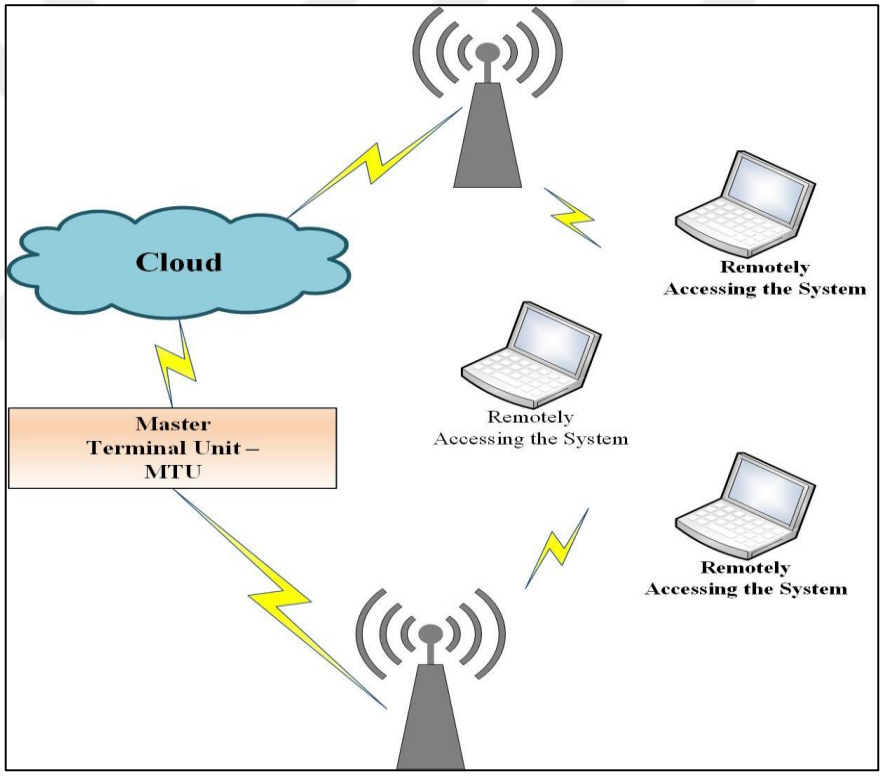

Figure 4. IoT-cloud based SCADA systems

2.2. The Internet of Things (IoT)

The IoT refers to devices and systems that are connected to each other via the Internet as communication networks, which are comprised of physical objects- sensor (or groups of such objects) (Mahmoud, R. et al, 2015). IoT is an innovation which empowers the internal inter connection of actual gadgets like structures, vehicles, and so forth with installed hardware, sensors, virtual products, and organization

availability to such an extent that the gadgets can gather and trade ongoing information among themselves and an administrator over a typical stage, the web or organization. As enter the twenty-first century, the IoT has emerged as a major technological advance. Using implanted gadgets, it is possible to connect everyday items, such as kitchen appliances, vehicles, indoor regulators, and child screens, to the internet so that they can communicate with each other and the world at large (Abomhara and Koien, 2014).

The risks associated with the development of Internet of Things technologies and products, particularly in the areas of privacy and security, have sparked widespread concern. As a result, the private sector and governments are working together to address these concerns, including the creation of international and local standards, guidelines, and regulatory frameworks (Statista, I. H. S., 2018).

2.2.1. The advances that made the IoT conceivable

Nowadays, IoT work is available on the market thanks to recent advances in a variety of fields. Recognition of low-cost and low-power sensor innovation. In order to make IoT innovation more accessible to more people, moderate and solid sensors are being developed. Advances in many fields enabled the IoT high speed development:

1) **Connectivity:** Sensors can now easily be connected to the cloud and other "things" via a variety of web-based organization conventions, allowing for the efficient transfer of data (Mumtaz, S. et al, 2017).
2) **Stages of cloud formation:** Customers and organizations alike benefit from the increased ease of access to cloud stages because they can now get to the foundation, they need to grow without having to oversee everything (Gubbi, J. et al, 2013).
3) **Inquiry into the use of machine learning:** With advancements in Artificial Intelligence (AI) and examination, as well as access to a wide range of data stored in the cloud, organizations are able to gather information more rapidly and more easily. As these partnered advancements continue to rise, so does IoT's ability to deliver the information necessary for these advancements (Hassan, Q. F., and Madani, S. A., 2017).
4) **Conversational computerized reasoning:** The advancements in neural networks have made it possible for IoT devices (such as the computerized personal

assistants Alexa, Cortana, and Siri) to handle normal language, making them more interesting, reasonable, and appropriate for use (Rani, P. J. et al, 2017).

2.2.2. Industrial Internet of Things (IIoT)

Current IIoT implies the applications of IoT development in many engineering fields such as mechanical, electrical, and chemical engineering, specifically with respect to control and instrumentation of sensors & devices which attract server propels. Lately, adventures have used Machine-to-machine (M2M) to achieve far off computerization and control (Butun, I., 2020). Regardless, with the improvement of server and added advances, ventures can achieve another automation layer and with it make new pay and strategies. IIoT is sometimes known as the 4th surge of the cutting edge upset, or Industry 4.0 (Singh, J., Tiwari, M., and Shrivastava, M., 2013).

Many applications can be accomplished by IIoT as shown in figure 5. The following are some common IIoT applications (Borhani et al, 2020):

- Smart producing
- Support that is both preventative and forewarned
- Energy/Power lattices
- Urban communities
- Coordination of the connection's
- Smart computerized supply chains

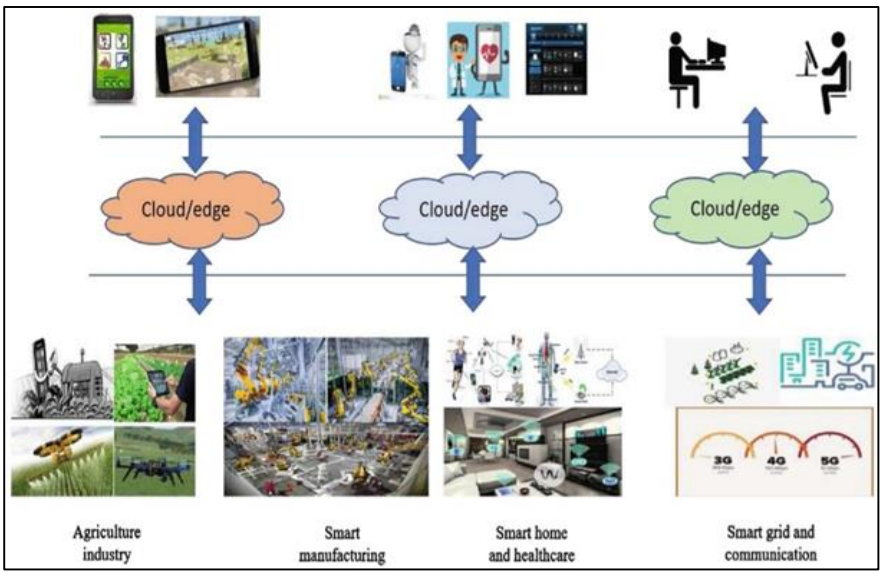

Figure 5. IIoT Applications

Whatever type of IoT design is used, a strong communication network is essential in an IoT-based system because devices are frequently dispersed across large geographical areas. The IoT headways make use of the four layers of the general TCP/IP model. The relationship between the TCP/IP model and the Internet of Things is depicted in figure 6. The application layer is the TCP/IP model's topmost layer. TCP/IP application protocols such as MQTT, HTTP, CoAP, and others are defined at the application layer, as well as how host programs interact with transport layer services to access the network. The transport layer is the third of four levels in the TCP/IP architecture, and it is responsible for data transmission. The protocol principal function in this layer is to enable communication between devices on the source and destination hosts. In the course of data transmission, the transport layer specifies the level of service and the current condition of the connection. TCP and UDP are the most often used protocols at the transport layer. The internet layer is a collection of Internet protocol suite internetworking methods, protocols, and standards that are used to transport network packets from the originating host across network borders and, if necessary, to a destination host designated by an IP address. The internet layer's name comes from the fact that it supports internetworking, which is the concept of connecting many networks via gateways. In the TCP/IP protocol architecture, the Network Access Layer is at the bottom. The protocols of this layer allow the system to send data to other devices on a directly connected network. It explains how to transmit an IP datagram across a network. Unlike higher-level protocols, Network Access Layer protocols must understand the underlying network (its packet structure, addressing, and so on) in order to properly format data and conform to network restrictions.

SCADA system proposed in this work is based on IoT-SCADA design, in which IoT features are merged into the traditional SCADA structure for more overwhelming data collection, remote checking, and authoritative control than is currently possible. This infers that IoT shows must be locked in with the IoT enabled SCADA system plan. Investigation work has shown that the most essential arrangement choices for making IoT enabled continuous applications like SCADA, text encoding plan, and the online or IoT stage. A highly efficient system for the Internet of Things can be constructed by picking the right protocols from each of the TCP/IP layers. IoT-based applications require a specific protocol to be chosen. As an example, deciding between HTTP, MQTT, CoAP, and the like at the Application Layer is difficult, as is selecting the

appropriate protocol for Network Access, Internet, Transport Layers. (Ismail, A. A., Hamza, H. S., and Kotb, A. M. , 2018). Using MQTT in the application layer is proven to be the best choice for the proposed system.

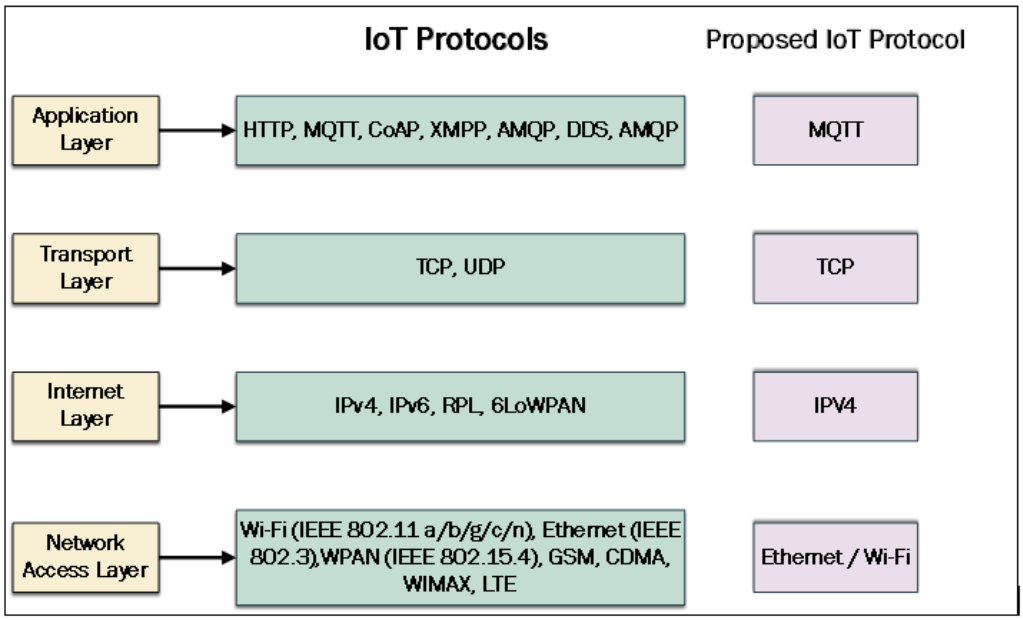

Figure 6. TCP/IP vs. IoT Protocols

2.3. PROTOCOLS

2.3.1. MQTT Protocol

One of the best M2M data communication tools is MQTT. This is a great fit for IoT applications because the MQTT, with a 2-byte fixed header, can be used with limited resources, such as a small amount of data transfer capacity and a small amount of computing power and storage capacity (Pirbhulal, S., et al, 2017). MQTT is based on the TCP/IP protocol suite and can be used on a variety of networks, including wired (Ethernet) and wireless LANs. Figure 7 illustrates the straightforward layout of the MQTT show (Sultana, T. and Wahid, K., 2019). Clients and other participants use MQTT, which is primarily a Publish-Subscribe advisory instrument. As a worker on a server or the web, the MQTT broker is in charge of preserving the circulated data based on various topics and sending the data to authentic subscriber who has joined up for the service (Bryce, R., Shaw, T., and Srivastava, G., 2018). The MQTT client is any piece of software, hardware, or a combination of both that works with the broker to exchange data. The communication is known as subscribing, and that client is known as a subscriber when a related contraption or client downloads data from the broker (laborer). When a connected client sends data to a broker, the client is referred to as

the publisher, and the communication is referred to as publishing. As a result, the term "publish-subscribe" instrument is used to describe the MQTT show. It's important to note that a broker can only serve a few subscribers and publishers at a time, depending on the capacity of the machine that runs the broker. A client can act as both a publisher and a subscriber in the same way. When a client circulates data, it assigns the data to a specific topic, with each topic bounded by a forward cut. There is a name and a level associated with each topic (Bryce, R., Shaw, T., and Srivastava, G., 2018). For devices with minimal resources, MQTT is developed as an application layer protocol. The publish-subscribe architecture strategy based on the topic name is the most important factor. The broker server receives data on a given topic from the source client. However, the destination customers must register as subscribers at the broker with the same previous subject's name in order to access the service. Each time the publisher sends data to the registered subject name, it is forwarded to the registered subscribers. End-to-end communication, a critical performance parameter in real-time and multimedia applications, is one of the primary challenges MQTT faces. Brokers act as an intermediary between publishers and subscribers. The subscriber can be overloaded by broker devices and packets are lost as a result, as shown in figure 8.

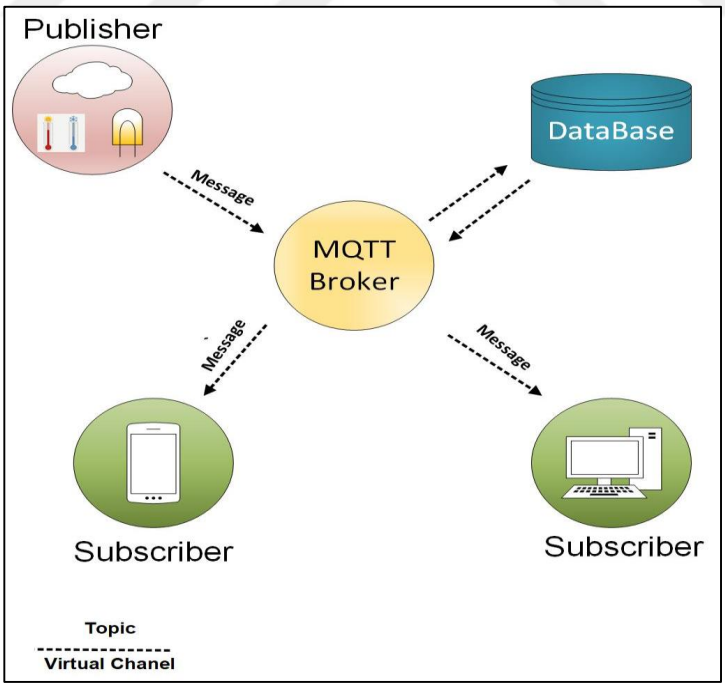

Figure 7. MQTT Architecture

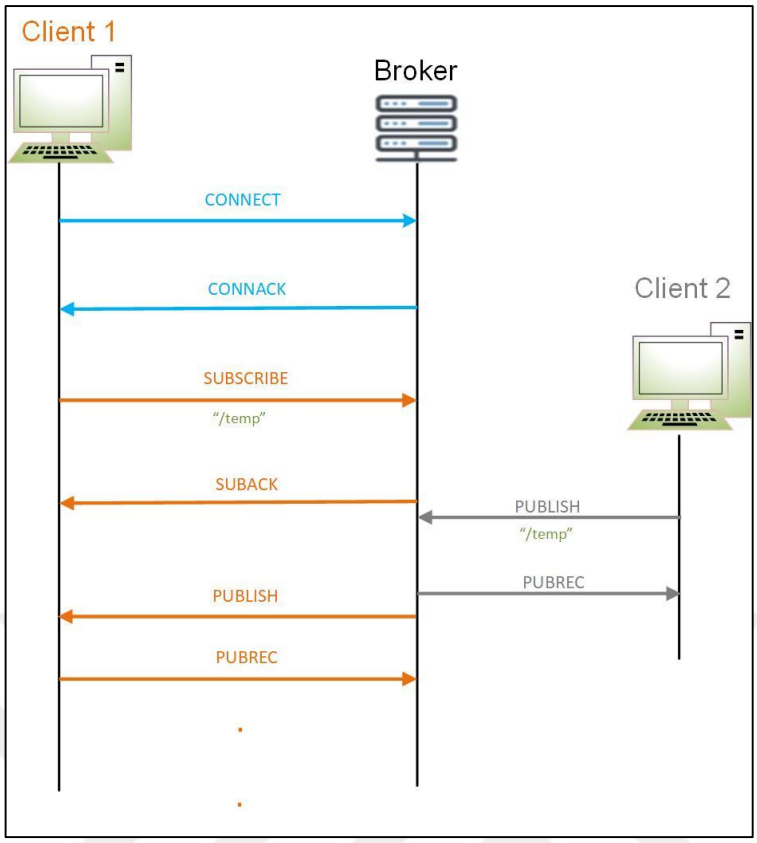

Figure 8. MQTT message control flow

For the transmission of messages, the MQTT protocol makes use of three levels of Quality of Service (QoS):

- **QoS 0 (at most once):** This service level assures that delivery is made with the utmost care. There is no guarantee that the order will be delivered. The recipient does not acknowledge the communication, and the sender does not store or re-transmit it as shown in Figure 9.
- **QoS 1 (at least once):** It ensures that a message is delivered at least once to the intended recipient. The message is saved until the sender receives a PUBACK packet acknowledging receipt from the receiver. A message can be sent or transmitted multiple times, Figure 10 illustrates this.
- **QoS 2 (exactly once):** It's the top tier of MQTT support. This level assures that each message is delivered only once to the designated recipient. When it comes to quality of service, QoS 2 is both the safest and slowest option. A four-part handshake between the sender and receiver is required to guarantee the exchange, Figure 11 illustrates this idea. The sender and recipient use the packet number from the original PUBLISH message to coordinate message delivery.

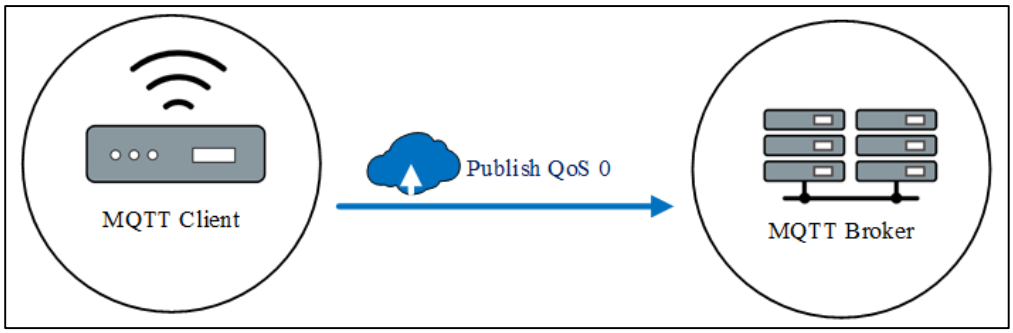

Figure 9. Level 0 of QoS

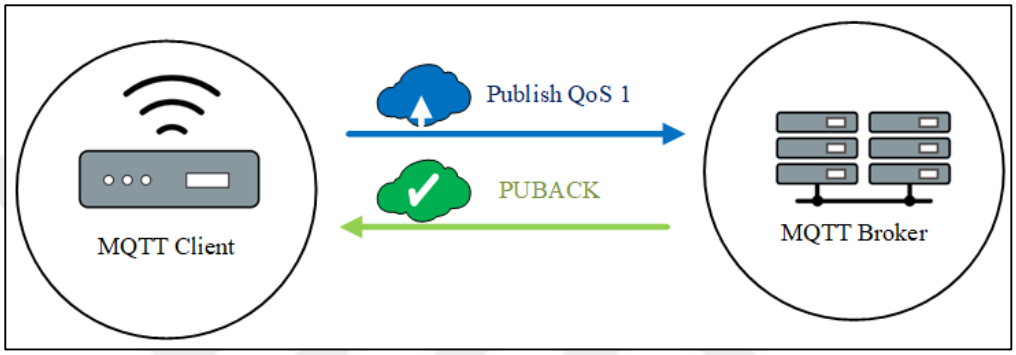

Figure 10. Level 1 of QoS

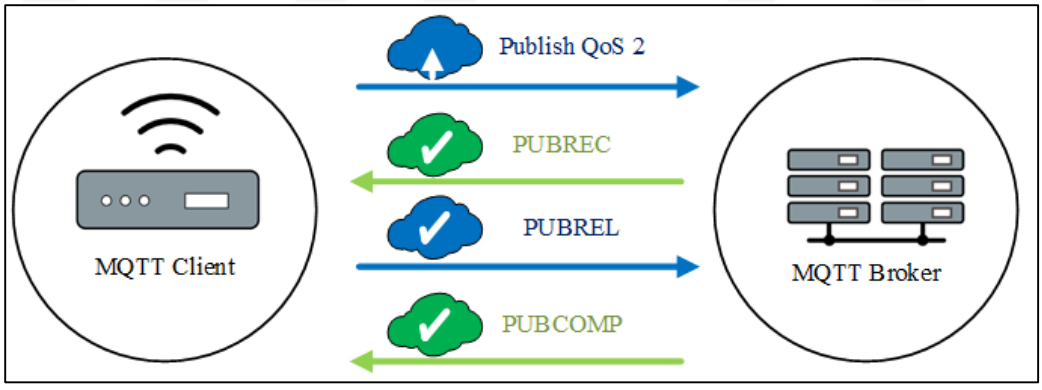

Figure 11. Level 2 of QoS

Along these lines, the protocol is dependable and appropriate for IoT applications where there are normally restricted assets like low data transmission, low memory, low force, and so for (Bryce, R., Shaw, T., and Srivastava, G., 2018) (Sahadevan, A. et al, 2017). This is the reason why MQTT is prefered in different information move protocols like HTTP, CoAP, REST API, and so on MQTT additionally runs on TCP/IP association dissimilar to CoAP which runs on UDP (Al-Fuqaha, A. et al, 2015). In light of these important elements of the MQTT convention,

various applications as of now use it. For instance, Facebook informing, savvy home observing, medical care, transport, energy metering, stopping, modern robots, producing, and shrewd checking frameworks have been carried out with MQTT protocol over different TCP/IP associations (Sarierao, B. and Prakasarao, 2018).

The MQTT protocol is made up of a sequence of MQTT control packets that are sent and received. which are: (Houimli, M., Kahloul, L., and Benaoun, S., 2017):

1. **CONNECT:** The packet that must be sent to the server after the network has been set up by asking a client to connect Servers have to disconnect clients if they send two CONNECT packets at the same time.
2. **CONNACK:** Acknowledgment connection requests are sent in response to a CONNECT packet from the client. The server transmits the first packet to the client. As soon as a CONNACK packet is not received in an acceptable amount of time, the client should close the network connection.
3. **PUBLISH:** is a packet that can be sent by both the server and the client in order to transport an application message between them.
4. **PUBACK:** This packet contains an acknowledgment of publication with a specified degree of QoS.
5. **PUBREC:** is the PUBLISH packet's QoS 2 response.
6. **SUBSCRIBE:** It is possible for one or more Subscriptions to be initiated by a customer who wants to express an interest in a certain topic(s). In addition, the SUBSCRIBE packet establishes the maximum QoS that the server can provide to the client when transmitting application messages (per Subscription).
7. **SUBACK:** confirmation packet that confirms receipt and processing of a SUBSCRIBE packet is called an acknowledgment packet.
8. **UNSUBACK:** the indication that the unsubscribe package Is received.
9. **DISCONNECT:** As a final control packet from the client to the server, it can signify a clean disconnect from the network.

2.3.1.1. MQTT Brokers

The primary responsibilities of an MQTT broker are to process communication between MQTT clients and to distribute messages between them based on their topics. At any given time, the broker can deal with thousands of connected devices. When the broker receives the message, it must search for all devices that have a subscription to

this topic (Grgić, K., Špeh, I., and Heđi, I., 2016). To choose a cloud-based broker, there are two options:

2.1.1.1.1. Private MQTT Brokers

There are three private brokers that are the most frequently employed among the diverse variety of available possibilities. Only devices that have been registered with the broker are allowed to operate. The first one is Azure, permits 100 connections accepted and 8000 messages/day in a manner free (Pierleoni, P. et al, 2019). The second broker is CloudMQTT permits 10 connections and a speed of 10Kbits/s and AWS permits 2.25 connection minutes 500,000 messages/month. The third private broker is AWS. The large distinction between them is the configurability that CloudMQTT presents contrasted with the other two, permitting the client to design the association points of the gadgets and the broker. Table 1 shows a table with the fundamental contrasts between these three private brokers.

Table 1. Private MQTT Brokers

Broker	AWS	Azure	CloudMQTT
TCP Port	N/A	N/A	Custom Port
SSL/TLS Port	8883	8883	Custom Port
WebSocket Port	443	443	Custom Port
Message Retention	N/A	N/A	Not Sure
Persistent Session	Limited	Limited	Yes
QoS Levels	0, 1	0, 1	0, 1 , 2

2.3.1.1.2. Public MQTT Brokers

In a public broker, any gadget can distribute and buy in to a topic that is in the broker; there's no Privacy. They ought to never be utilized underway. They are extremely helpful for figuring out how to utilize the interchanges convention. Table 2 shows the contrasts between the absolute generally known, additionally discover the connection to every one of the brokers, where its fundamental attributes and qualities

are determined in more detail. In the event that decide to work locally by introducing a broker on a PC or virtual machine, first should pick the worker need to work with. Most intermediaries have their own details and the Minimum necessities that the machine where the broker will be introduced should have for the right activity of the equivalent. Among the most popular are: HiveMQ, Mosquito, Eclipse Mosquito. In Table 2, some discrepancies between public brokers are shown.

Table 2. Public MQTT Broker

	Eclipes	**Mosquitto**	**HiveMQ**	**Flespi**	**Dioty**	**Fluux**	**EMQX**
TCP Port	1883	1883	1883	1883	1883	1883	1883
TLS port	N/A	8883	N/A	8883	8883	8883	8883
WebSocket Port	80, 443	80	8000	80, 443	8080 , 8880	N/A	8083
Message Retention	Yes	Yes	Yes	Yes	Yes	N/A	Yes

2.3.1.2. MQTT Clients

As referenced in past segments and sections, any gadget that can be associated with an organization and that can run a MQTT library is a possible customer for the convention. Know a portion of the important libraries so any gadget can interface with an organization and work with the convention. An application that is introduced can likewise be viewed as a customer on a gadget, for this situation we are discussing client test systems or customer applications.

2.3.2. HTTP protocol

In the early 1990s, Tim Berners-Lee developed HTTP, a web messaging protocol that is still in use today. Internet Engineering Task Force and World Wide Web Consortium later worked together to create the protocol, which is first published as a standard protocol in 1997. (Han, N. S., 2015). In order for RESTful Web architecture to be implemented, HTTP must have the ability to receive and respond to requests. The message control flow is depicted in figure 12. HTTP is similar to CoAP in that it uses Uniform Resource Identifiers (URIs) instead of topics. Using this unique

URI, both the server and the client can exchange information. It is the webserver or programming technology used to determine the size of the headers and message payloads in an HTTP text-based protocol. HTTP makes use of TCP as the default transport protocol and TLS/SSL as the security protocol (Grigorik, I., 2013). As a result, client-server communication is centered around the establishment of a connection. Without additional support, it doesn't define Quality of Service. HTTP is a widely used web messaging standard that provides a number of features, including persistent connections, request pipelining, and chunked transfer encoding, among many others (Naik, N., and Jenkins, P. 2016) HTTP is an easy-to-use protocol because of three key characteristics:

- **HTTP is connectionless:** The HTTP client, which is typically a browser, initiates an HTTP request and waits for the response after making a request. Upon receiving a response, the client disconnects from the server's network and disconnects from the Internet. When the current request and response cycle ends, the client and server are only aware of each other. This indicates that the client and server aren't familiar with each other because additional requests are made on new connections.
- **HTTP is media independent:** It means that HTTP can send any type of data as long as both the client and the server understand how to handle the data.
- **HTTP is stateless:** As previously stated, HTTP is a connectionless protocol, which is a direct result of HTTP's status as a stateless protocol. The server and client are only aware of each other during the course of a particular request. After that, neither of them remembers the other person anymore. Since the protocol is designed in this manner, neither the client nor the browser is able to retain information between successive requests across different web pages (Wood et al, 2009).

2.3.2.1. Advantages of HTTP:

1- REpresentational State Transfer (RESTful) architecture with URI addressing is supported.

2- Provides support for distributed tracing, which allows for the identification of failures across a distributed system architecture.

3- able to operate even when a firewall is closed.

4- makes use of a push approach, which involves maintaining a persistent connection between the client and the server.

2.3.2.2. Disadvantages:

1- Header size is large: This increases the burden on devices, particularly those in the IoT.

2- High network latency: Consequently, the protocol is unsuitable for real-time or mission-critical IoT systems.

3- employs text encoding rather than binary encoding.

4- The protocol does not support QoS.

5- The scalability of the protocol is difficult.

6- Encrypting and decrypting (secure) messages add computational overhead.

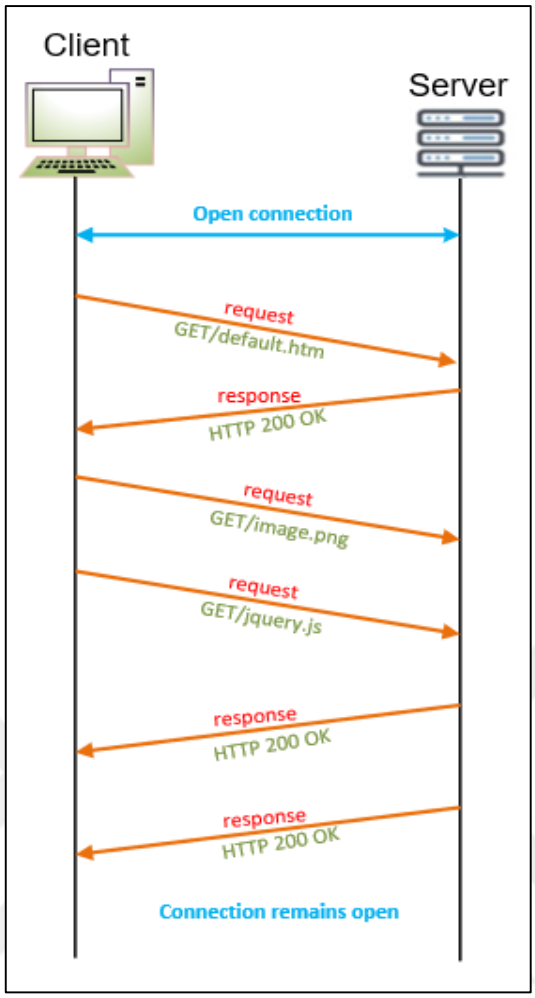

Figure 12. HTTP message control flow

Table 3 shows the comparison between the Internet protocols MQTT and HTTP.

Table 3. MQTT vs. HTTP: A Comparison of IoT Messaging Protocols

Criteria	MQTT	HTTP
Year	1999	1991
Architecture	Client/Broker	Client/Server
Abstraction	Publish/Subscribe	Request- Response
Header Size	2 Byte	8 Byte
Message Size	Undefined & small (up to 256 MB maximum size)	Undefined and large (based on the web server's or programming language's technology)
Semantics/ Methods	Connect, Disconnect, Publish, Close, Subscribe, Unsubscribe	Post, Get, Head, Patch, Put, Connect, Options, Delete
QoS /Reliability	QoS 0 - At most once (Fire-and-Forget), QoS 1 - At least once, QoS 2 - Exactly once	Limited (via Transport Protocol - TCP)
Transport Protocol	TCP (MQTT-SN can use UDP)	TCP
Security	TLS/SSL	TLS/SSL
Default Port	1883/ 8883 (TLS/SSL)	80/ 443 (TLS/SSL)
Encoding Format	Binary	Text
Licensing Model	Open Source	Free

CHAPTER THREE
METHODOLOGY

3.1. Monitoring system overview

Automation that is both cost-effective and effective in meeting oil demand can be achieved through intelligent design. The IoT can be used to better manage energy use in the residential, commercial, and industrial sectors. SCADA systems for pipeline monitoring and control are typically expensive and difficult to maintain. An Arduino UNO, Node-MCU, and Node-RED/MQTT SCADA system is depicted in figure 13 for monitoring and controlling appliances from a distance via local Wi-Fi. Wi-Fi is a short-range wireless transmission system that allows radio signals to connect to the Internet. With Wi-Fi, consumers may easily access the Internet at anywhere and anytime. Many different application scenarios necessitate the creation of data and service-sharing infrastructures. There are services that can be shared between many applications, such as anomaly detection in sensed data at the Application layer. Applications that are available as a hosted service over the Internet and may be accessed by anyone with an Internet connection are referred to as software-as-a-service (SaaS). One of the most complete forms of cloud computing services is SaaS, often known as cloud application services, which delivers an entire program controlled by a provider via a web browser. Temperature, leakage, liquid flow, pressure, and flame detection are just few of the parameters that can be monitored by this system. As a result, users can monitor various equipment remotely and make decisions based on the information from sensors. As a result, sensor data will be sent to the Node-Red dashboard, figure 14 depicts the proposed system's block diagram. Sensors collect data on the pipeline system's many parameters and send it to Arduino. The Arduino's job here is to collect sensor data from a variety of sensors and send it to NodeMCU. The Inter-Integrated Circuit (I2C) communication protocol, commonly known as the 2-wire protocol since it only uses two wires, the Tx and Rx lines, delivers data from Arduino to NodeMCU. Arduino is required to connect the sensors that provide digital or analog output signals, the digital data is sent from NodeMCU to the MQTT broker.

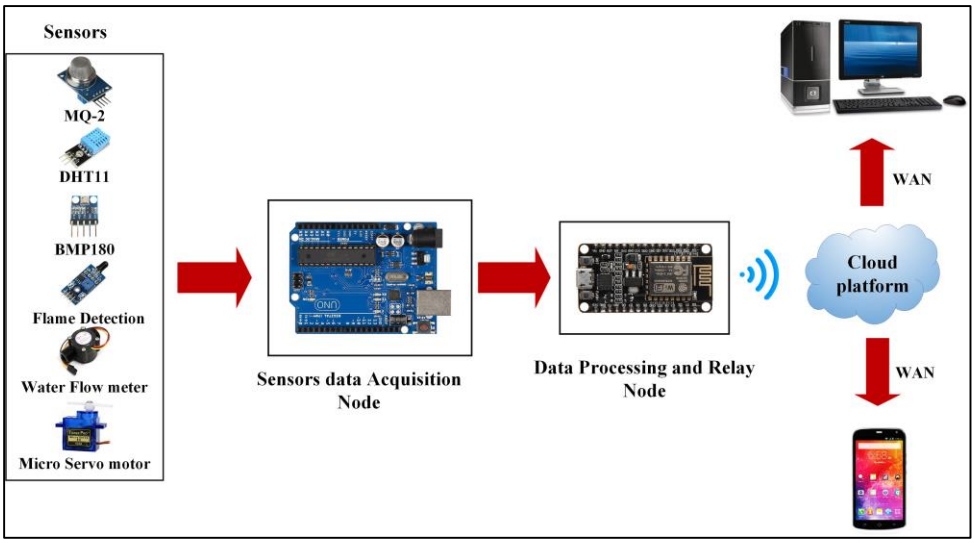

Figure 13. General System Proposed

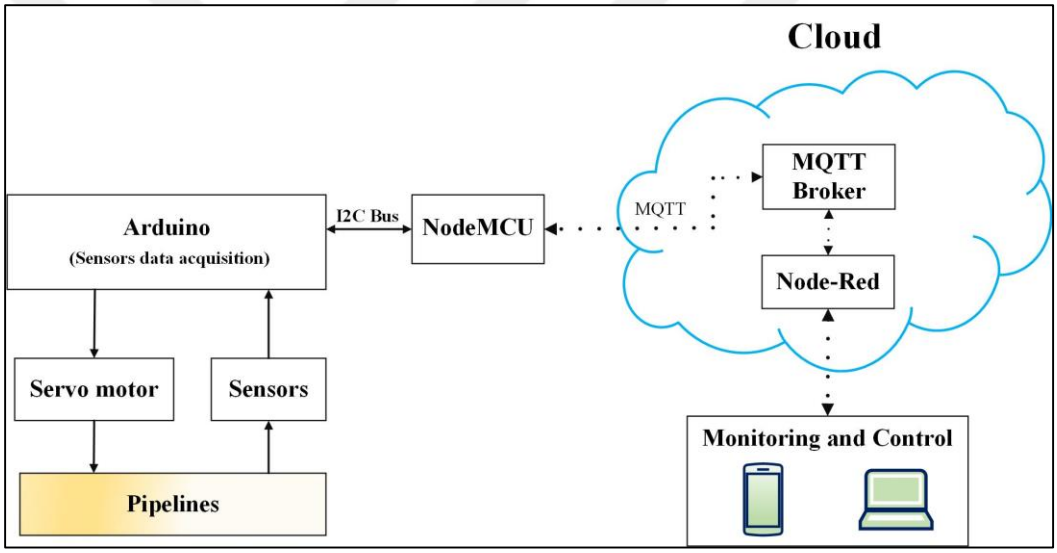

Figure 14. System block diagram

The MQTT protocol is used to communicate between the MQTT broker and the NodeMCU. Although the Secure Socket Layer/ Transport Layer Security (SSL/TLS) encryption is an option, because MQTT is designed to be a lightweight protocol, encryption may not be employed in all instances. In this case, the client sends a username and password in clear text to the server for authentication. Entities that use MQTT are classified as MQTT brokers and clients, respectively. As a MQTT broker and data visualization tool, Node-RED is a must-have. Because of this, the MQTT broker may be monitored and data sent to using a graphical user interface. Node-RED also connects to the MQTT broker through MQTT. Monitored and controlled systems are application-based systems that can include a wide range of devices such as

computers as well as mobile phones. Figure 15 depicts the Arduino code flowchart, In the setup section of the Arduino work, the first step is to define the sensors, functions sent and received, devices, and pins as variables. NodeMCU and some sensors are connected to the Arduino through I2C. Arduino UNO reads the data in the loop part after the sensors and nodes are attached to it. The Arduino-connected NodeMcu sends or receives data in accordance with the appropriate functions, then performs the necessary actions on the incoming data. NodeMCU can connect to the internet and send sensor data to a MQTT broker using the MQTT publish and subscribe protocol after receiving sensor data from Arduino. NodeMCU can also be used to accept control data from a MQTT broker and send it to Arduino via the MQTT protocol. The data received from NodeMCU is subsequently processed by Arduino. This allows for data monitoring as well as control of the Arduino's I/O pins, these steps are shown in figure 16.

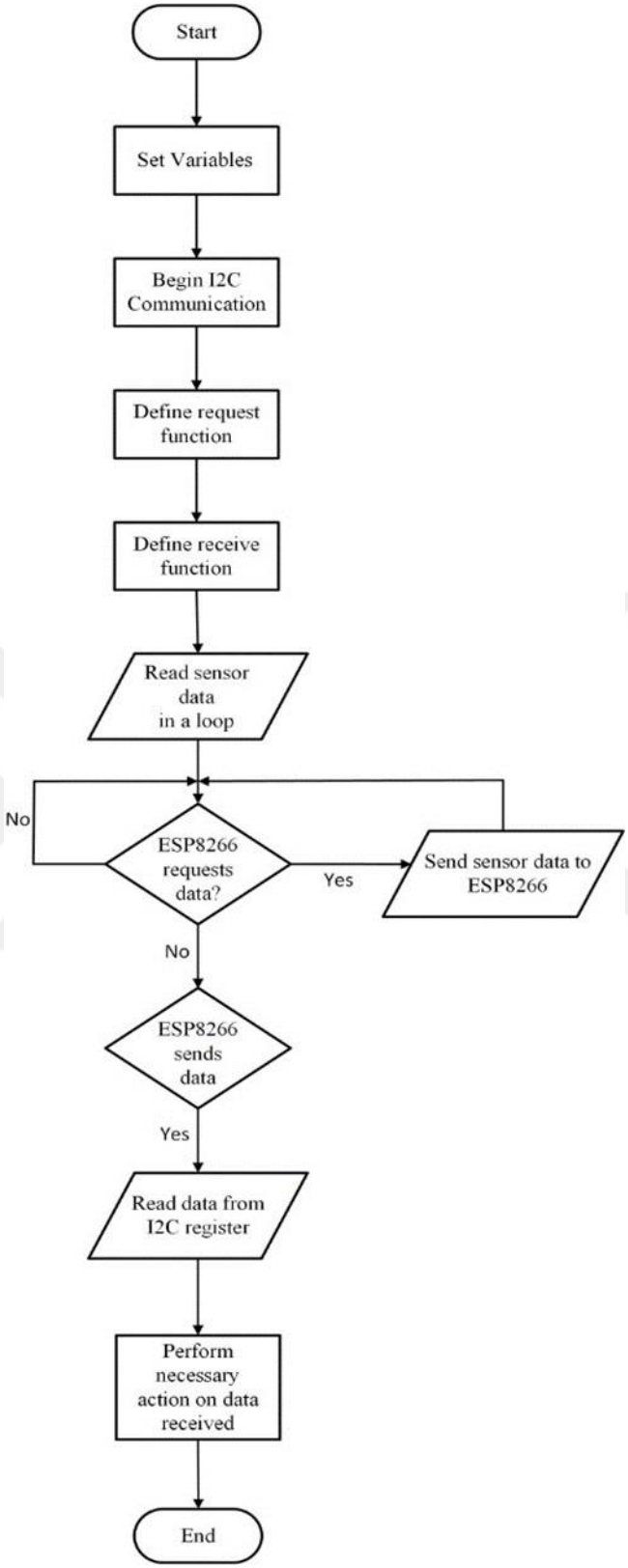

Figure 15. Flowchart showing workflow for Arduino

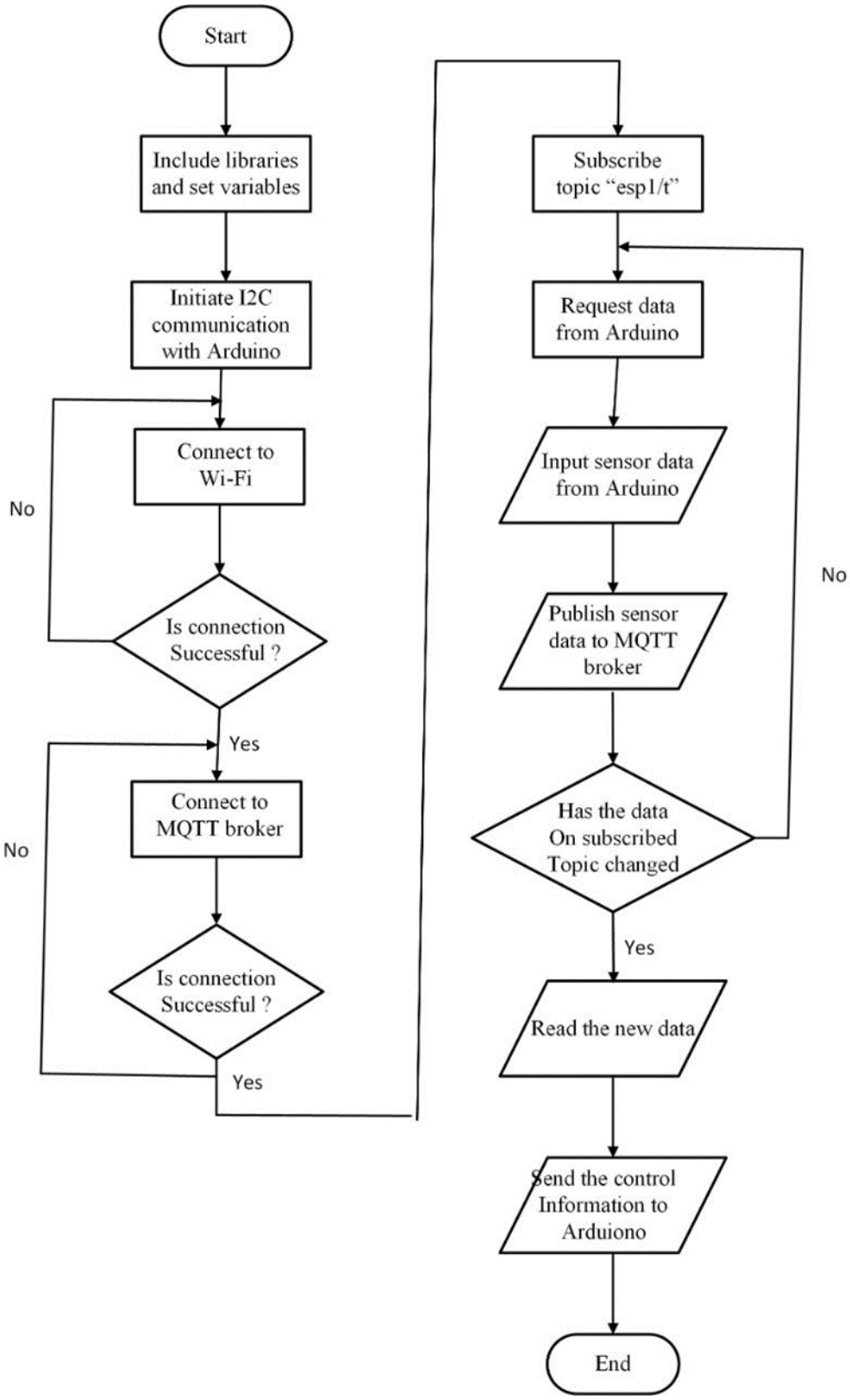

Figure 16. Flowchart showing workflow for Node-MCU ESP8266

A variety of sensors are included in the design of the pipeline monitoring system. Figure 17 shows how the sensors are linked to the Arduino UNO, connecting cables are used to link NodeMCU to Arduino. Wi-Fi is supported by the NodeMCU microcontroller. Once the MQTT server has been set up and connected to the NodeMCU, the sensor data can be displayed on a website or mobile application.

The suggested system is separated into two sections. The sensors and microcontrollers are in the first section, while the MQTT broker and data display are in the second section.

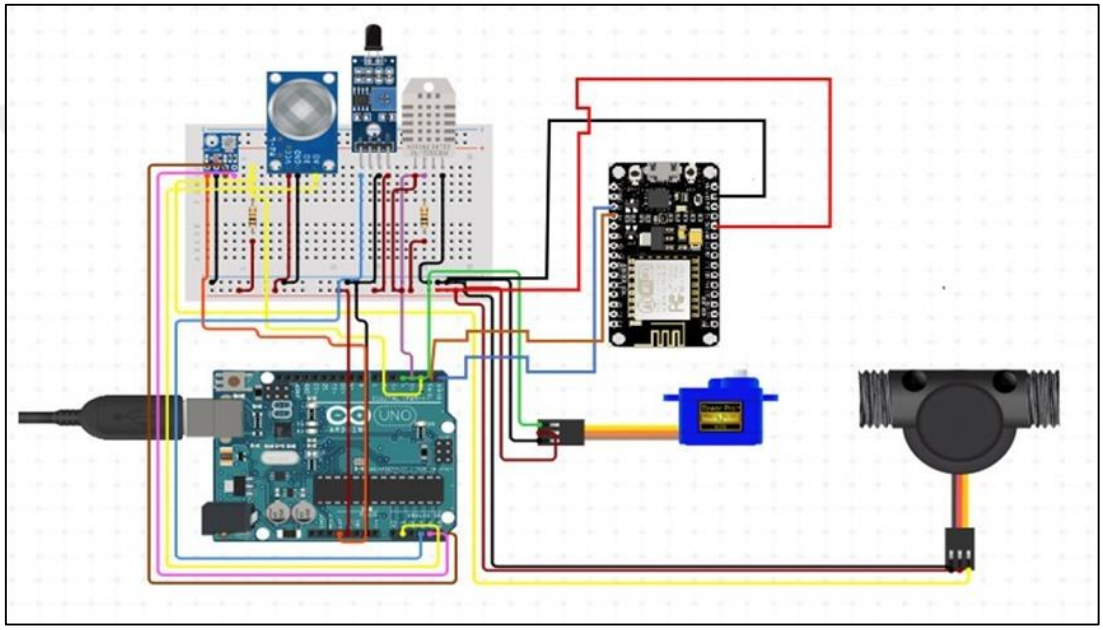

Figure 17. Design of the proposed system's circuit diagram

3.2. Proposed system structure

The proposed IoT-based open-source SCADA structure equipment and networking plans are shown in this section, the proposed SCADA system is depicted in Figure 18 as a block diagram, with an Arduino Uno addressed by RTU and a NodeMCU ESP8266 serving as an interface between devices and the cloud for sharing data between the client and sensors via the web using the MQTT protocol. An Arduino Uno with a bunch of sensors to monitor temperature, pressing factor, stream and gas spillage, and a fire recognition sensor in case of a fire close to oil pipelines. A servo motor acts as a valve, opening and closing the oil pipeline in the event of a fire, gas leakage, or any other problems.

The Arduino code is made by the Arduino environment improvement program, and data is sent via the MQTT protocol. In advising transportation shows, MQTT is a Client-Server Publish/Subscribe protocol.

It's an open and lightweight, designed to be simple for both distributors and allies to implement. These characteristics make it ideal for use in a variety of situations, including those that are required in this research. MQTT's lightweight nature and twofold impression are two of its positive qualities, which help it to dominate when moving data wirelessly. In contrast to widely used conventions such as HTTP, MQTT has a negligible amount of overhead.

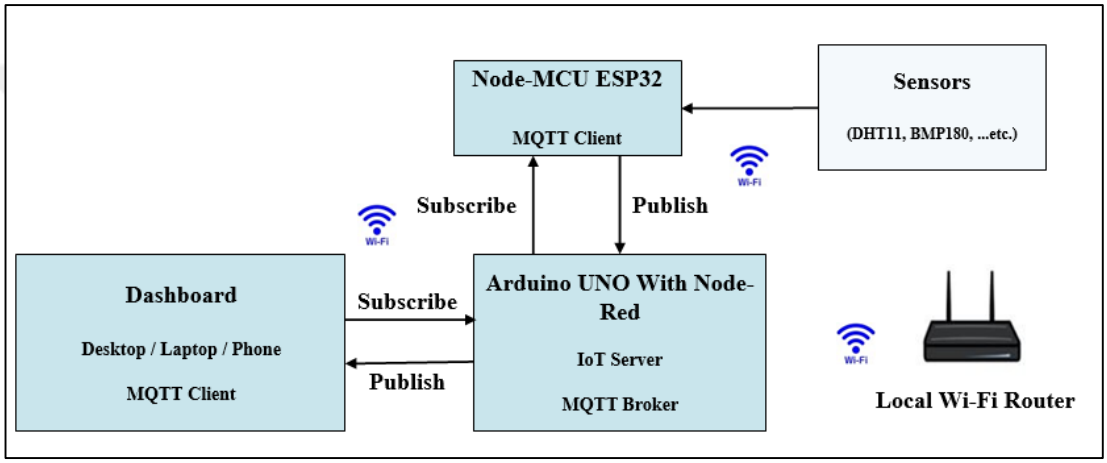

Figure 18. The proposed SCADA system block diagram

3.2.1. System Hardware

3.2.1.1. Arduino UNO

It is an Alf and Vegard's RISC processor (AVR) based microcontroller based on the Microchip ATmega 328P family and manufactured by Arduino. This board includes cutting edge sets and straight forward data which is attached and interfaced with different augmentation sheets and electronic circuits (Matijevic, M., and Cvjetkovic, V., 2016), As shown in figure 19, Arduino-based (I/O) pins can be connected to a variety of sensors, devices, and electronic circuits.

Figure 19. Arduino UNO

In total, the board has 14 General-Purpose Inputs and Outputs (GPIOs), six of which are Pulse-Width Modulation (PWM), six of which are Analog, and six of which are USB B type. A USB Type B cable or a 9-volt battery can be used to supply the board with an input voltage of 7 to 20 volts. A computer, another Arduino board, or other microcontrollers can be connected to the Arduino Uno via a number of workspaces. Pins 0 Receive (RX) and 1 Transmit (TX) of the ATmega328 have been modernized to provide sequential Universally Asynchronous-Receiver/Transmitter Transistor- Transistor Logic (UART TTL) (5V) correspondence (TX). On the board, an ATmega16U2 is used for this sequential correspondence, which appears as a virtual COM port for programming on a computer system. (Badamasi Y. A., 2014).

3.2.1.2. Node MCU ESP8266

Node MCU ESP8266 is an open-source hardware, started with firmware for Espressif Systems' ESP8266 Wi-Fi system on chip and gear that used the ESP-12 module. The ESP32 32-bit MCU is then added to the mix later. The ESP8266 Wi-Fi Module is a free System-on-a-Chip (SoC) with a built-in TCP/IP stack that allows any microcontroller to connect to the internet. The ESP8266 can work with an application or delegate all Wi-Fi limit checking to another processor. Each ESP8266 module comes pre-programmed with AT request set firmware, allowing it to be connected to Arduino and have a Wi-Fi range similar to that of a Wi-Fi Shield (which is scarcely out of the box), the ESP8266 Wi-Fi Node-MCU Module is shown in figure 20. The ESP8266 module is a low-cost board with a large and rapidly growing community. The device's onboard processing is extremely powerful, and its storage capacity allows for seamless integration with sensing modules and other devices via its GPIOs while processes are running (Wan Z., Song Y. and Cao Z., 2019).

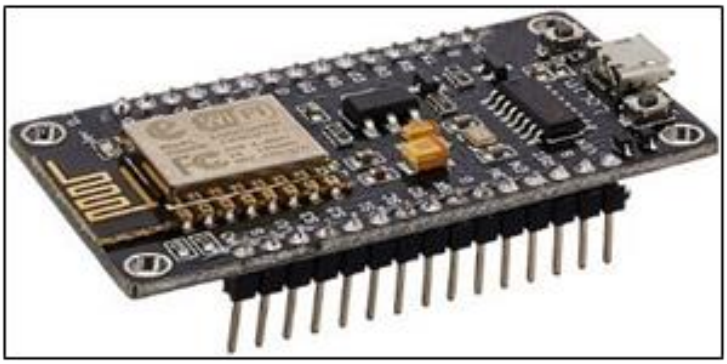

Figure 20. Node-MCU Module ESP8266 Wi-Fi

3.2.1.3. BMP180 Sensor

The BMP XXX series includes the BMP180 sensor. Barometric pressure or atmospheric pressure is the sole purpose of these devices. The BMP180 has a high-accuracy sensor designed for customer use. Nothing but the weight of air is represented by barometric pressure. The weight of the air can be felt in any place where there is a lot of air. Using a BMP180 sensor, the data is fed into a computer to produce a computerized yield (Ch S., Abdul and Pendem, 2020). Associations at this level are called "primary." To begin, connect the Arduino's VIN pin to the 3.3V power pin and GND to the ground. The pins used for I2C communication are what will be used for the time being. Each Arduino board has a different number of I2C pins that must be connected correctly (Kodali, R. K., and Mandal, S., 2016). Figure 21 shows the structure of the module BMP180 sensor.

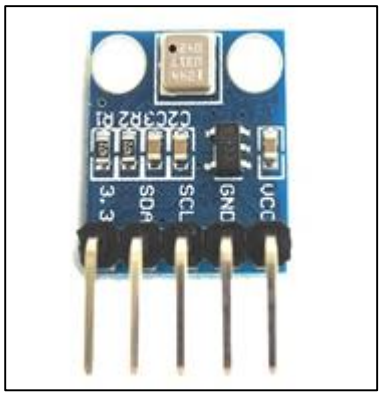

Figure 21. Module BMP180 Sensor

3.2.1.4. DHT11 Sensor

Capacitive sensor material is used in this hardware module to measure the relative mugginess and temperature. A thermistor with a negative temperature coefficient estimates the parameters. In terms of efficiency and dependability, it has the potential to last for a very long time. Figure 22 shows the structure of the DHT11 sensor. A thermistor and a capacitive mugginess component are both included in this sensing module. It has two terminals, one of which is held in place by a dampness substrate. It's a dielectric, which is why it's so common. The capacitance of the sensor changes as a result of changes in environmental parameters such as mugginess. The IC calibrates the output based on the measured change (Sharmila, F. M. et al, 2019). Temperature is measured using the Negative Temperature Coefficient concept, which is similar to the thermistor. As the temperature rises, so does the value give to the reduction in obstructions. Using semiconductor ceramics or polymers, this sensing module is able to measure the temperature change with the smallest possible change in temperature. The DHT11 temperature sensor module has a temperature range of 0 to 50 degrees Celsius with a two-degree tolerance. This module has a dampness range of 20% to 80%, with a 5% tolerance. This module's sampling frequency is one hertz. Three to five volts is the recommended operating voltage. Temperature measurement uses a maximum of 0.3 milliamps (mA) (Abhyankar, V. et al, 2019).

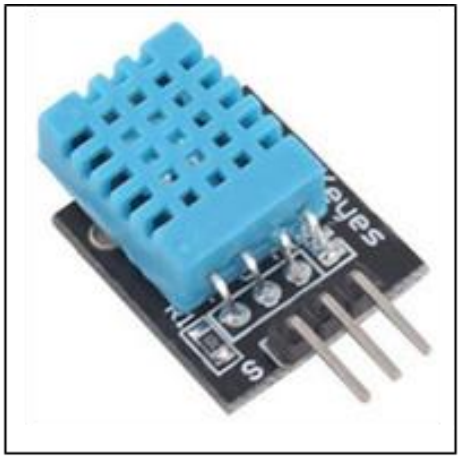

Figure 22. Sensing Module DHT11

3.2.1.5. Gas Sensor – MQ2

A combustible gas sensor is in use here. Smoke can also be detected by it. In all places, it identifies the groups of combustible gas, and its output is tracked as a voltage. From 300 to 10,000 ppm of gas can be measured with this device. Liquid Petroleum Gas, Iso-Butane and propane, methane and liquor can also be measured with this instrument. In domestic and industrial settings, this sensor can be used to identify supplies that have been spilt, as well as in a compact gas locator (Catalina H et al, 2016). Figure 23 shows the Module gas sensor.

Figure 23. Module Gas Sensor MQ2

A combustible gas sensor is in use here. Smoke can also be detected by it. In all places, it identifies the groups of combustible gas, and its output is tracked as a voltage. From 300 to 10,000 ppm of gas can be measured with this device. Liquid Petroleum Gas, Iso-Butane and propane, methane and liquor can also be measured with this instrument. In domestic and industrial settings, this sensor can be used to identify supplies that have been spilt, as well as in a compact gas locator (Catalina H et al, 2016).

An aluminum-oxide-based clay sensor with a tin dioxide-coated detecting component is housed in a hardened steel network. Four legs of the detecting component are connected to each other. Two leads are in charge of warming up the detecting component, while the other is in charge of generating yield signals. Oxygen is adsorbed on the outer layer of a sensing material when it is heated to a high temperature in air. The flow of current is effectively halted due to the benefactor electrons in tin oxide's attraction to the oxygen atoms. In the presence of decreasing gases, oxygen molecules react with adsorbed oxygen's surface thickness to reduce its thickness. As a result of current flowing through the sensor, it can now read simple voltage values (Pandey, R. C. et al, 2017).

3.2.1.6. Flame Detector

By comparing the signal from this sensor to a reference signal, it can determine whether or not there is a fire. The establishment's response to a well-known fire is critical. Because of the components, it uses to distinguish the fire, a fire identifier can often react faster and more precisely than a smoke or warmth indicator. Module Flame Detector shown is in figure 24.

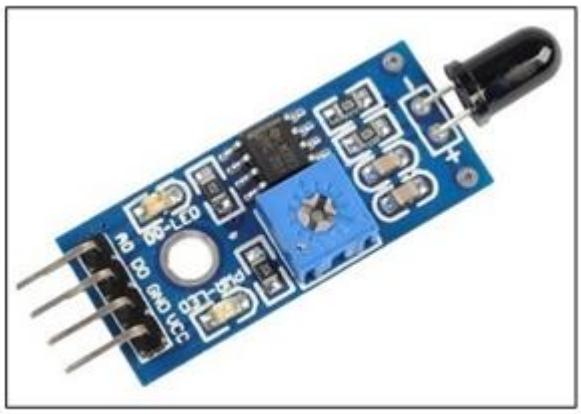

Figure 24. Module Flame Detector

There are a variety of methods for recognizing a fire. UV locator, near-IR bunch identifier, infrared finder, infrared warm cameras, UV/IR marker, etc. are just a few examples. The sensor module's Photodiode (an IR gatherer) picks up a small amount of infrared light when the fire consumes it. When the IR Receiver detects a change in voltage, it uses an Op-Amp to see if the yield pin (DO) gives 0V(LOW), and if there is no fire, the yield pin will be 5V. (HIGH). It is employing an IR-based fire sensor in this endeavor. Because of the high speed and high fragility of the YG1006 sensor (an NPN silicon phototransistor), it is used in this device. Perception ranges from 760 to 1100 nm, with an area point of about 60 degrees, in the range of infrared light. The potentiometer can be tinkered with to alter the sensitivity. There is a working voltage somewhere between 3.3- and 5-volts DC, with an electronic yield of somewhere in that range (Sasmita, E. S., Rosmiati, M., and Rizal, M. F., 2018) (Diwanji, M., Hisvankar, S., and Khandelwal, C., 2019).

3.2.1.7. Flow Water Sensor

The Arduino flow is based on the "Hall Effect" principle. Electromagnetism and electric current induce a voltage difference in transverse conductors. Flow meters use Hall Effect and have a rotor in the shape of a small fan/propeller that is arranged

in the liquid's flow path for measurement. Figure 25 depicts the YF-S201 water flow sensor. (Singh, P., and Saikia, S., 2016).

Figure 25. Flow Water Sensor

Water rushes through the flow sensor, and as a result, the rotor's fans are pushed by the water flow. The rotor's shaft is connected to a Hall Effect sensor. A current-flowing coil and a magnet are used to attach to the rotor's shaft. A voltage or pulse is generated as the rotor turns. One pulse is generated for each liter of liquid that passes through the sensor in a minute when using the YF-S201 water flow sensor, which uses the same technology. Variations in a magnetic field are seen because of the magnet attached to the rotor shaft The Arduino counts the number of pulses it generates and displays this information. A simple formula is used to calibrate this flow-based pulse reading in L/hr. Three wires are connected to the water flow sensor: a red wire for 5V Vcc, a black wire for GND, and a yellow wire for the signal/pulse. The flow sensor's Vcc and Gnd pins are connected to the Arduino's Vcc and GND pins, and the pulse line is connected to one of the Arduino's digital pins.

3.2.1.8. SG90 Servo Motor

Armatures can only be moved at a fixed angle by servo motors, or servo mechanisms. In order to rotate the SG90 devices, shown in figure 26, a pulsed signal whose width corresponds to the servo motor's rotation is used. In order to rotate the SG90 servo, signals with pulse widths between 0.5ms and 2.5ms, at 20ms between pulses, are required. pulse width modulation is a close relative of the pulsed signal. Colour-coded red, brown, and white wires connect the servo motor to power, ground, and signal. For millisecond-long periods of time, a servo motor draws hundreds of milliamps of current (Amali, L. Y., and Batan, I. M. L.).

Figure 26. SG90 Servo Motor

3.2.2. Software of the proposed system

3.2.2.1. MQTT Protocol

This is a Publish/Subscribe in convention. It is a clear and lightweight informing instrument that is intended for confined gadgets, problematic organizations and low transfer speed. As it gives a basic association between the specialist, worker and customers, it is considered reasonable for this proposed plan (ESP8266 microcontroller, cell phones and PC). Where clients are permitted to distribute information to themes for information or subscribe in points. Then, at that point, the client disseminates the information to any customer who has subscribed to this subject. In this task, the ESP8266 microcontroller publishes the information subscribed from the sensors for certain particular subjects to the MQTT intermediary. While PCs and mobiles show this information distributed to the worker. Administrative control in this undertaking is conceivable in the MQTT protocol. Figure 27, shows the architecture and how data is exchanged between publisher and subscriber by MQTT protocol. Unlike the traditional client-server approach, in which a client communicates directly with an endpoint, MQTT clients are split into two groups: A sender (referred to as a publisher in MQTT) and a consumer that receives the data (an MQTT subscriber). The publisher and the subscriber do not know anything about each other, and, in reality, are never in direct communication with each other. A third component (an MQTT broker), operates like a ‘traffic cop’, routing messages from the publisher to any endpoints functioning as subscribers.

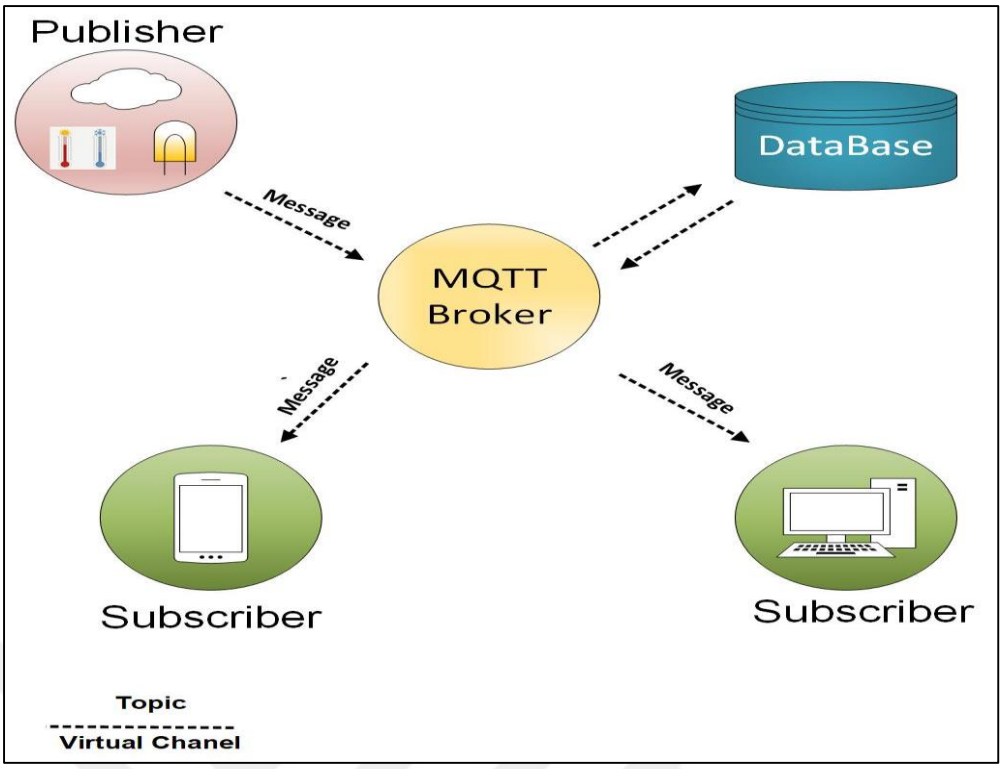

Figure 27. Data exchange between publisher and subscriber by MQTT protocol

3.2.2.2. Authentication of MQTT protocol

The rapid growth of the IoT and cyber-physical systems (CPS) has resulted in high demand for smart gadgets that are equipped with sensors that collect data from their surroundings, process it, and send it to remote sites for further analysis. The widespread use of IoT devices, as well as the pressures of device development time to market, have highlighted security and privacy concerns.

One of the most significant components in the design of a communication protocol in an IoT-based system is an effective and secure authentication method. MQTT, as one of the most widely used messaging protocols in the IoT industry, provides simple authentication by username and password. The proposed pipeline monitoring system includes the design and implementation of token-based MQTT protocol authentication in restricted devices. As seen in figure 28, it requires the internet client to submit a username and password before access to Node-RED on the server is authorized, hence boosting security. The adminAuth attribute secures the Node-RED admin Application Programming Interface (API). If this variable is not set, anyone with network access to Node-RED can use the Node-RED admin API.

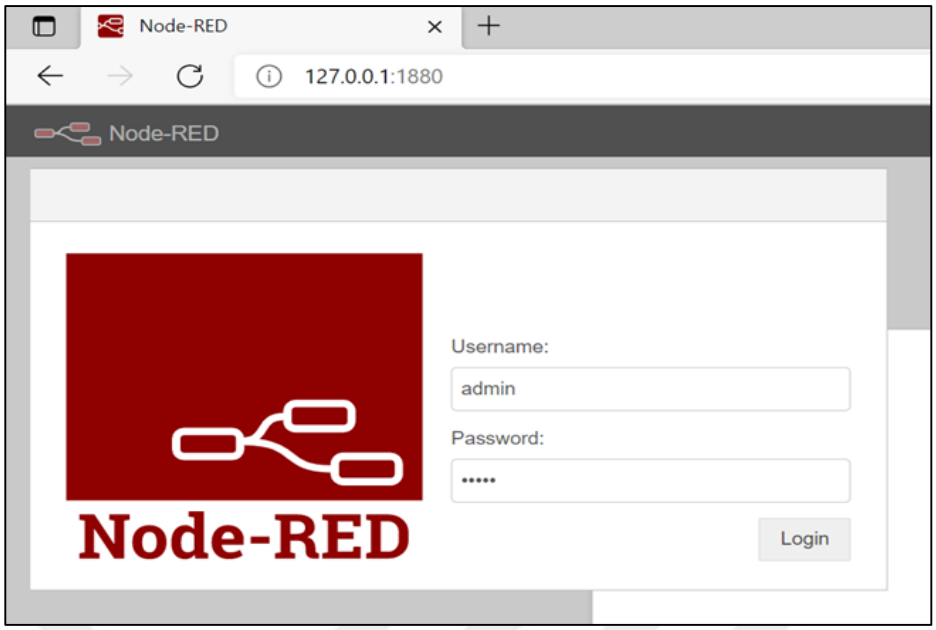

Figure 28. Authentication MQTT protocol for Node-Red

3.2.2.3. Node-Red

International Business Machines Corporation (IBM) created the Node-Red platform, which is a Visual Programming Language (VPL) platform. The goal of Node-RED is to allow users to construct IoT applications utilizing graphically depicted software components. Node-RED works with a variety of hardware, including Arduino, Raspberry Pi, and the Beagle Bone Black. Android, Windows, and Mac are among the operating systems it supports. IBM Bluemix and Microsoft Azure are two cloud systems that can be employed. Several API are available for the building of IoT applications, including runtime execution APIs, graphical user APIs, Shell Prompt APIs, Storage APIs, and HTTP. It also has the ability to integrate with external APIs that may be deployed via the Dashboard component. JavaScript Object Notation (JSON), JavaScript, HyperText Markup Language (HTML), Comma Separated Values (CSV), and Extensible Markup Language (XML) format are among the data representations supported (Ferencz and Domokos, 2019).

3.2.2.3.1. Node-RED flows

In addition to being a programming tool, Node-RED serves as an execution platform for applications built using the Node.js runtime. IoT, web services, and other Node-RED applications require the use of the flow editor. Node.js powers both the flow editor and the website it is hosted on. In fact, the main purpose of Node-RED is

a Node.js-powered web application that serves as a flow editor. A browser-based flow editor is at work here. Drag the node that wants to utilize from the palette to the workspace and select it. Create an application by connecting nodes to each other using a wiring. An application can be deployed to a target runtime with just a single click for the developer. Installing new nodes created by developers is as simple as adding them to the palette already populated with various nodes. Figure 29 depicts the proposed system's flow wire connections.

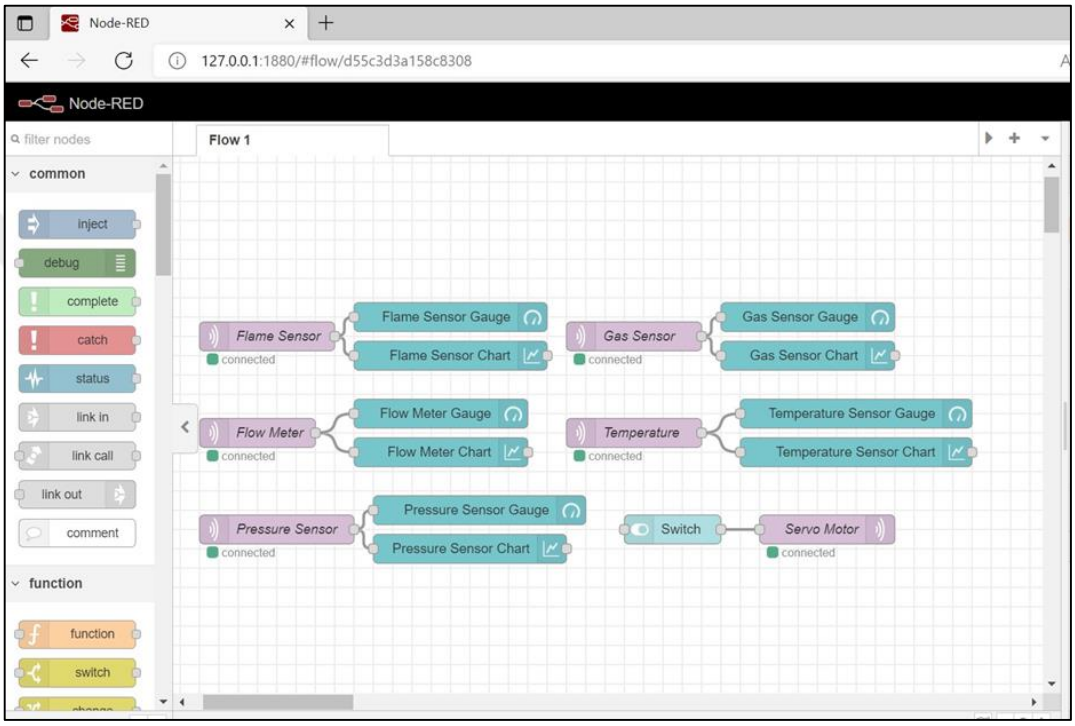

Figure 29. Node-Red Flow

3.2.2.3.2. Web Dashboard

The dashboard's major goal is to allow users to alter and gather data in a graphical interface that is easy to read. Each interface displays data in a clear and ordered manner. As a result, the necessity of a simpler control panel is to convert a set of difficult-to-read data into clearer, easier-to-read images, allowing for quicker decisions and a better understanding of what's going on. The use of a dashboard is critical in all areas, as different companies and factories use different solutions to track their performance in various industries.

The Node-Red tool is used to create a web application, through which a dashboard is created to monitor and control the system, as this tool supports the MQTT protocol. Node-Red offers a dashboard, which is a local web application. The web server is set

locally by installing the tool on the computer and defining it. Based on sensor data, displays three different types of user interfaces (metrics, scripts, and charts). Furthermore, as illustrated in figure 30, the dashboard is a dynamic online application that can be seen in any web browser. The dashboard's design should be considered as a grid. The widget is the primary element, and it is organized into groups. The group element's width is scaled in units, with one unit equalizing 48 pixels by default. The width of the tools is likewise set to automatic by default, which means it will be as the group's width.

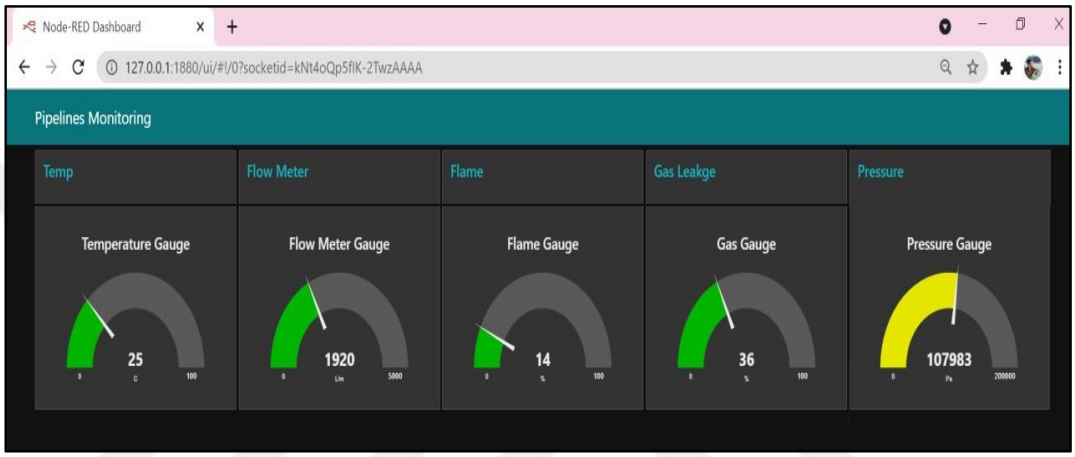

Figure 30. Web Dashboard of the proposed system

3.2.2.4. IoTMQTTPanel Phone Application

IoTMQTTPanel is an internet of things application that offers remote monitoring and control of electronic equipment via its iOS and Android phones. This application enables the management and visualization of IoT projects using the MQTT protocol. It provides a dashboard from which users may develop graphic interfaces by combining various widgets. Sensor data can also be stored and displayed by the application. It comprises libraries for a wide range of hardware platforms, including Raspberry Pi, NodeMCU, Arduino, SparkFun, and others. App, Server, and Libraries are the three most important components of the IoTMQTTPanel. The software can help with the design of the user interface. All communication between the app and the hardwires are handled by the server. And libraries allow hardware to communicate with the server via commands, figure 31 displays the dashboard for monitoring sensors for the proposed system in the smart phone application.

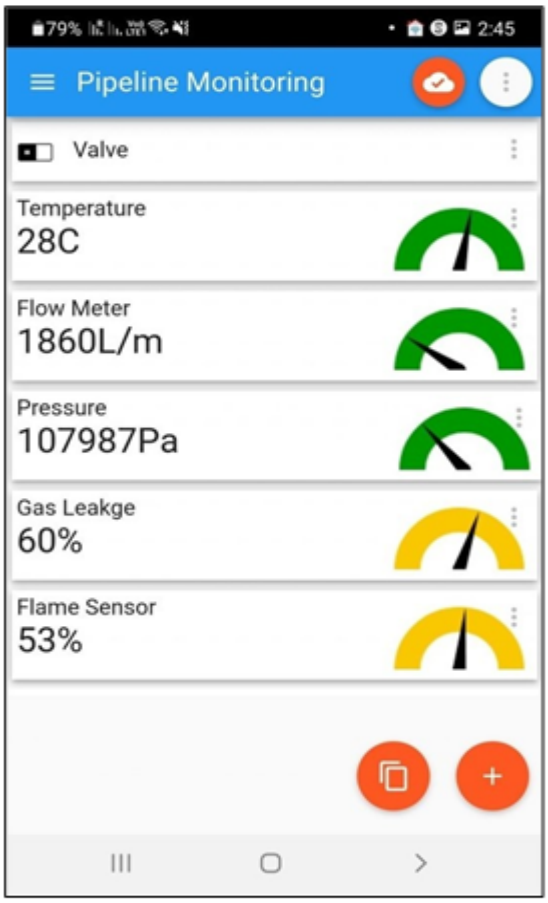

Figure 31. Sensors read via IoTMQTTPanel application

3.3. Proposed pipelines monitoring system

Chemical industries are currently attempting to identify and offer reliable methods for conserving their fluid resources and monitoring their pipeline systems. They go over all of its benefits in depth, demonstrating that it is a cost-effective way for the sector to treat its oil and gas while also increasing productivity. This research emphasizes the importance of IoT Pipelines management. As a result, leaks and damage along the oil pipeline will be easier to detect. To create a smart oil distribution network, a dynamic programming-based solution method is provided. The IoT oil monitoring scenario is used in this solution, improves the oil system's robustness and may be adapted to other monitoring domains with ease.

The system uses cloud computing to monitor and control oil pipeline. All the sensor readings are saved on a separate web server. Local web-server has the ability to show the data. As depicted in figure 32, a web browser is used to monitor oil pipelines. To construct an IoT architecture, several sensor nodes and smart tools are connected and

employed. Sensors and embedded systems that take advantage of the Internet of Things can be used to provide intelligent administration and monitoring. The suggested method is built on a variety of sensor nodes, such as standard sensor nodes that are connected and mounted on pipelines. The temperature sensor DHT11 is used to assess whether the pipe or the surrounding environment has a normal or abnormal temperature. Sensors are used to detect gas leaks and a fire detector sensor in that region, in addition to a flow meter for calculating the speed of oil flow via pipelines and a pressure sensor for knowing the pressure of the liquid inside the pipe. In addition to the sensors already mentioned, servo motors act as valves to open and close the pipe.

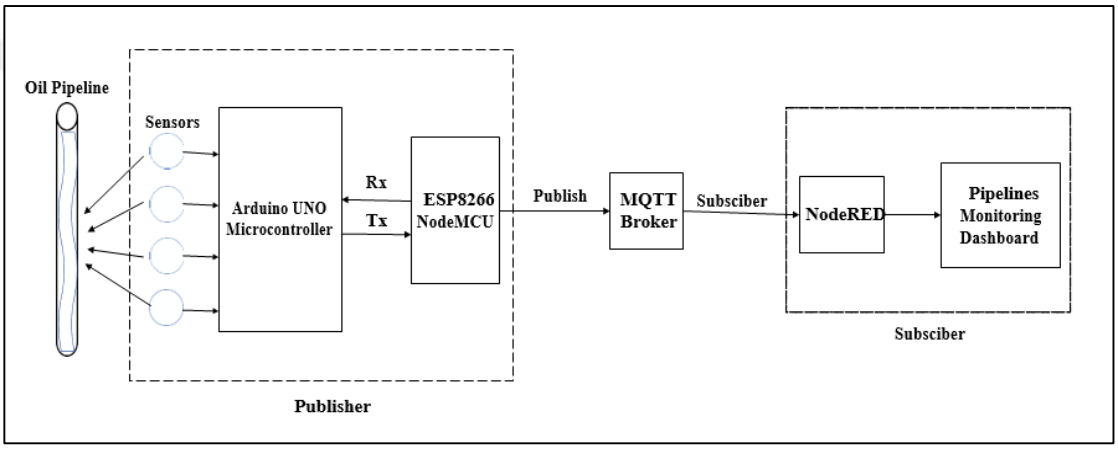

Figure 32. Oil Pipelines Monitoring System Architecture

Figure 33 illustrates the proposed pipeline monitoring system's working environment, as well as the dashboard of NodeRED, which displays sensor readings on a computer screen.

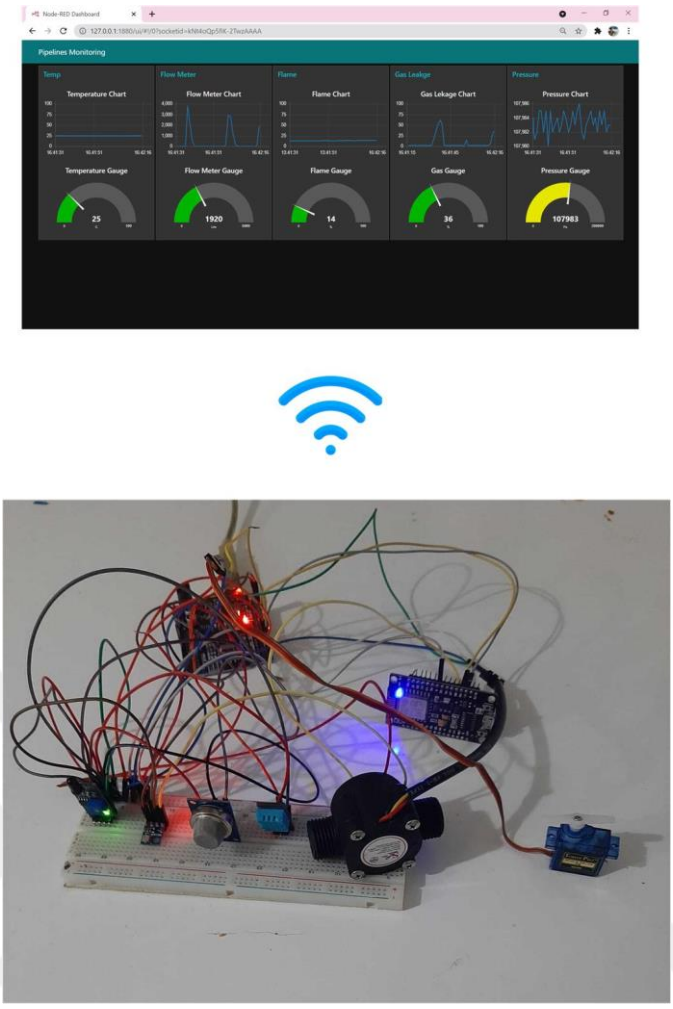

Figure 33. Experimental system

CHAPTER FOUR

SIMULATION MODULE

4.1. SIMULATION PLATFORM

The simulation of the proposed system aims to monitor special waves for the protocols used, which shows the delay of data transmission, the amount of data transmitted, and the amount of oscillations for the MQTT and HTTP protocols in a simulated environment. For extensive algorithm level and modelling simulation, LabView is used for model simulation. In addition, using Arduino and a temperature sensor, a circuit is created in the Proteus program for simulation. The Arduino environment is used to program the circuit used in this proposal. Selecting the right circuit parameters in the simulation environment allows the delay time and data packet size to be monitored. In the simulated scenario, the MQTT protocol and the HTTP protocol are compared in terms of data transmission delay time between the server and the client, as well as the size of the data packet delivered between the client and the server. A circuit is designed in the Proteus program to send data in real-time to a circuit designed in the LabView program for the MQTT protocol, here the Proteus circuit is considered as a client and the circuit designed in the LabView is considered as a server, as well as a data platform is made in the firebase to send data in real-time to the HTTP protocol circuit is also designed in LabView, finally the protocols are compared with each other. Noting that the data sent to the two protocols is the same type and amount of data generated from two different sources which makes the comparison between the two protocols valuable.

4.1.1. PROTEUS

For electronic design automation, the Proteus Design Suite is a closed-source software toolset. It is largely employed for the creation of electronic circuit designs and prints for use in the manufacture of printed circuit boards. Using this software may capture schematics, run simulations, and create Printed Circuit Board (PCB) layouts. Depending on the size of the created designs and the simulation requirements for microcontrollers, they are available in various formats. When designing a PCB, Proteus Design Suite's circuit design feature is utilized for model simulation. As a result, it is an essential part of the design. The proteus microcontroller simulation works by modifying the microcontroller component of the design with a hex or debug

file. Any analog/ digital electronic equipment linked to it is then co-simulated. This enables it to be used in a variety of project prototyping applications, Including IoT projects, motor control, temperature management, and user interface creation.

An Arduino Uno microcontroller is used in the proposed circuit. This microcontroller is connected to various sensors such as the temperature and humidity sensors, as indicated in figure 34. Communication Port (COM) is also utilized to construct a virtual port in order to connect this circuit to a virtual port configured in the LabView in order to transport the temperature and humidity sensor data to the LabView's interface.

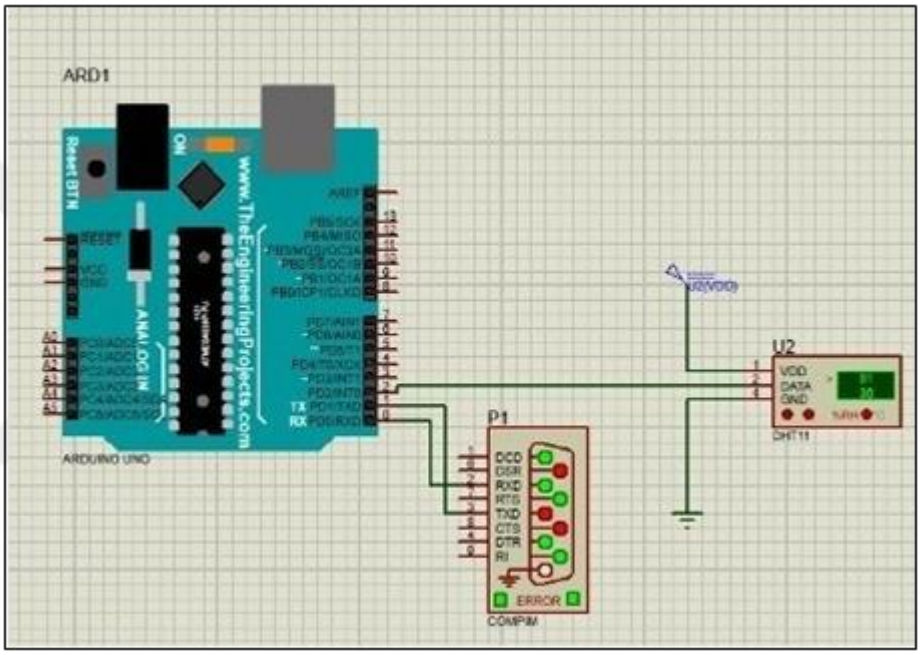

Figure 34. Simulation using DHT11 sensor in Proteus

COMPIM model is used to simulate the virtual COM interfaces that can be used to create a virtual serial port using Bluetooth or USB connectivity. The electrical circuit processes serial signals once they have been recorded and buffered. The computer's serial ports will receive all serial data generated by the Central Processing Unit (CPU) or the Universal Asynchronous Receiver-Transmitter (UART) type. Virtual serial ports can be created with USB connectivity using a variety of technical solutions. It is possible to convert baud rates when using the COMPIM model. Hardware and software can be used in conjunction to verify the device's virtual and physical properties. The virtual serial port is used to send temperature changes to LabVIEW. LabVIEW is used to present the results.

4.1.2. LabVIEW

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a visual programming language system design framework and development environment. The creation of user interfaces is a part of the LabVIEW development process. LabVIEW programs and subroutines are virtual instruments (VIs). Each VI includes a block diagram, a front panel, and a connecting pane. In other block diagrams, which are referred to as VIs, they are used as sub-stacks. The front panel is made up of controls and indicators. Inputs are controls that allow a user to give the VI information. Indicators are outputs: based on the inputs, they suggest or represent the VI's outcomes. On the rear panel, which is a block diagram, is the graphical source code.

On the back panel, all objects placed on the front panel will display as terminals. The rear panel also contains structures and functions that operate on controls and provide data to displays. The Functions palette can be used to place structures and functions on the back panel. Nodes are the aggregate term for controls, indications, structures, and functions. Two controls and an indicator, for example, could be attached to the addition function, causing the indicator to display the sum of the two controls. As a result, a virtual instrument can be run as a program with the front panel acting as a user interface, or as a block diagram node with the front panel defining the node's inputs and outputs via the connector pane. This means that each VI can be double-checked before being used as a function in a larger program.

Using the free LabVIEW Interface for Arduino (LIFA) toolbox, a LabVIEW developer can simply exchange data with the ever-popular Arduino microcontroller. LabVIEW sends serial commands to an I/O engine on the Arduino, and the Arduino reacts by providing the needed data or action.

4.1.2.1. MQTT Library for LabVIEW: LVMQTT

Among different libraries, two libraries are referenced to have the option to utilize them in the LabVIEW advancement climate. One of them carries out more prominent security working with SSL authentications, while the other, more fundamental, doesn't execute said security, leaving the security of correspondence, to the intrinsic in the TCP/IP stack. With the MQTT correspondences, it has been chosen to utilize the fundamental library that doesn't present SSL security. In this library, there is a set of VI and subVI necessary for the MQTT communication. They all have error handling, presenting error inputs and outputs. If an error occurred at the input of the

VI, it spreads on exit immediately. The following sections will try to mention the most important parts of each of the VIs necessary for the correct operation of the protocol.

4.1.2.1.1. MQTT_Connect.vi

To use MQTT, a client must establish a connection with the broker before it can exchange publications and subscriptions with the broker. A "CONNECT" message has been defined to accomplish this goal. Other connection parameters include a client ID, which the broker can use to identify a connected client. The broker makes use of this client Id, This VI is responsible for setting up the association between the broker and the client. Figure 35 shows the VI symbol and information boundary set and the vital yields for additional handling, featuring the ID of (association ID out) and the sort of association (Connection Status).

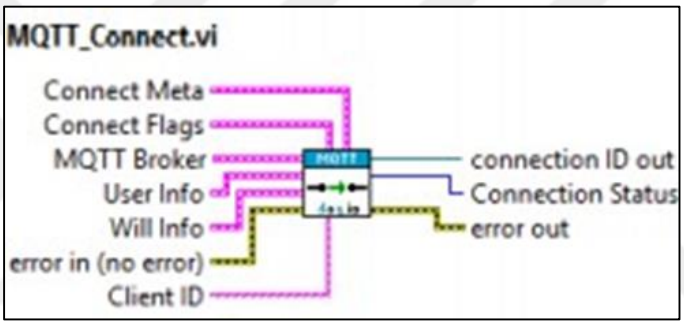

Figure 35. MQTT_Connect.vi function

4.1.2.1.2. MQTT_Disconnect.vi

DISCONNECT is a control packet that can be delivered by a client to the server as the last control packet in order to indicate that the client has successfully disconnected from the server. This VI is responsible for making the correct disconnection, by the client, between the client and the broker. In figure 36, the icon and entry to the VI is shown.

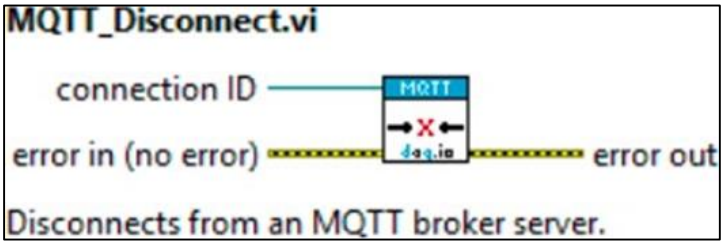

Figure 36. MQTT_Disconnect function

4.1.2.1.3. MQTT_PingReq.vi

When there is no control packet to be delivered to the server, a client can send a PingReq to inform the server of its presence. The existence of this packet informs the server that the client is still alive. It is also used to exercise the network and to show that the network connection is active when exercising the network. This VI is responsible for sending a ping solicitation to the broker to keep experience the association between the client and the broker and that a detachment isn't made undesirable. In figure 37, the VI information and yields are shown. These yields are excessive for additional handling.

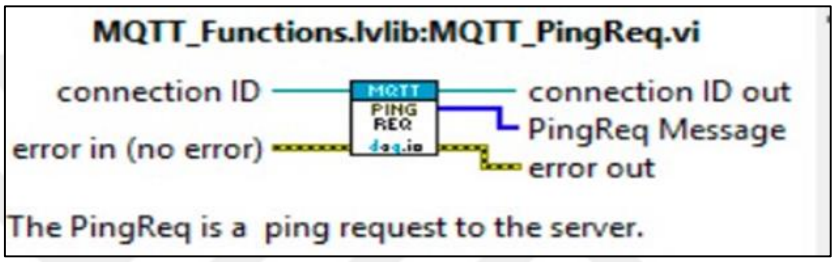

Figure 37. MQTT_PingReq.vi function

4.1.2.1.4. MQTT_Publish.vi

PUBLISH is a packet that both the server and the client can send to carry an application message. This VI is accountable for sending the publish message to the broker. In figure 38, the VI symbol is shown, where the boundaries of the passage and exit thereof. This yield should be equivalent to the information, considering which is the Identifier of the association.

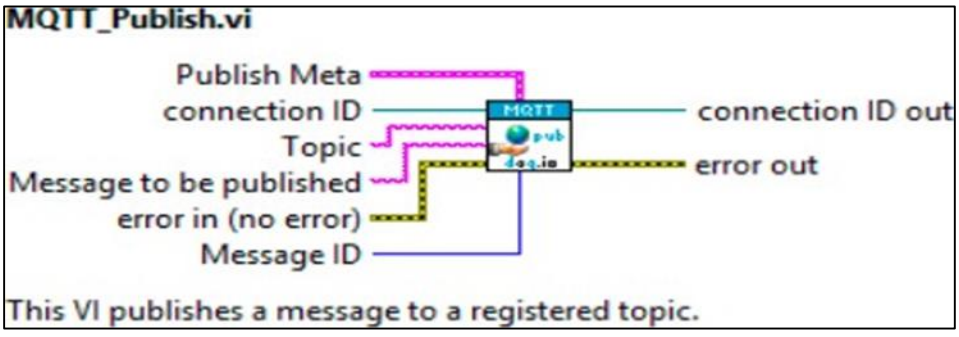

Figure 38. MQTT_Publish.vi function

4.1.2.1.5. MQTT_Subscribe.vi

When a client sends the command SUBSCRIBE, it can be used to generate one or more Subscriptions, which are used to register a client's interest in one or more

topics. A maximum QoS with which the server can send application messages to the client is also specified in the SUBSCRIBE packet (for each Subscription). This VI is accountable for subscribe into one or a few topics in the broker by the Client, to begin getting the messages distributed in that. Figure 39 shows the VI symbol and its feedback and yield boundaries. If no error occurs, the association ID from the information is spread to the yield.

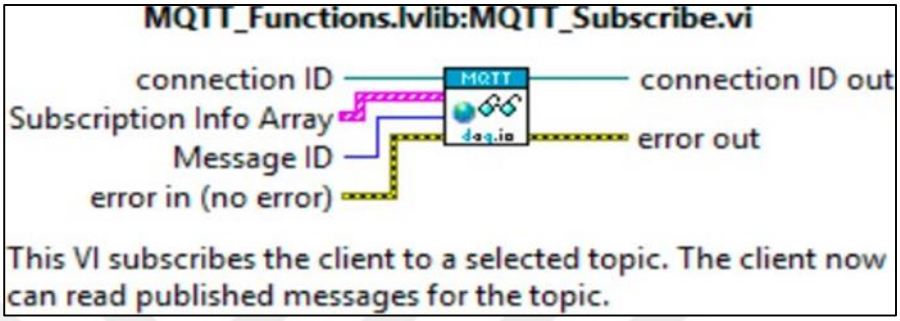

Figure 39. MQTT_Subscribe.vi function

The circuit is designed for the MQTT protocol in LabVIEW, as shown in figure 40, where a set of LabView functions is used like the logic gates, and the timer that acts as a time parameter, as well as some functions that are called from the MQTT library such as connect and disconnect to connect the client to the server, also used subscribe and publish functions in order to receive the temperature sensor data from the Proteus to the LabView. The port that is used in the designed circuit is 80 which does not provide safety.

After the circuit designed in the Proteus application is connected to the circuit designed in LabView, real-time data is sent from Proteus to LabView to measure the delay time and size of the received data packet.

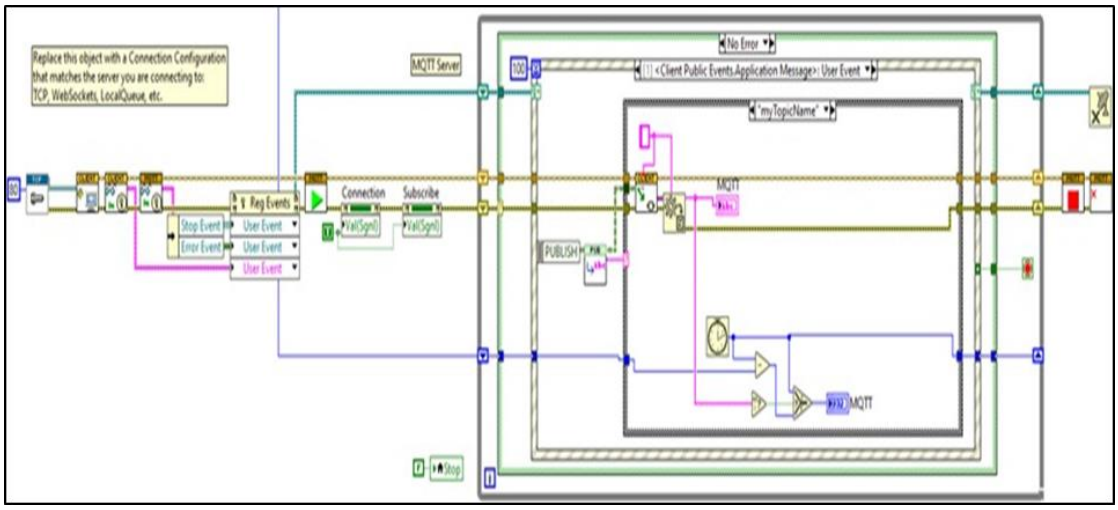

Figure 40. Block diagram of MQTT protocol

4.1.2.2 HTTP Library for LabVIEW: LVHTTP

When downloading the library of the HTTP protocol to the LabVIEW, it will get a set of functions for the protocol, where each function has certain tasks. Figure 41 shows the POST.VI function, which works to send a request to create new data from the web.

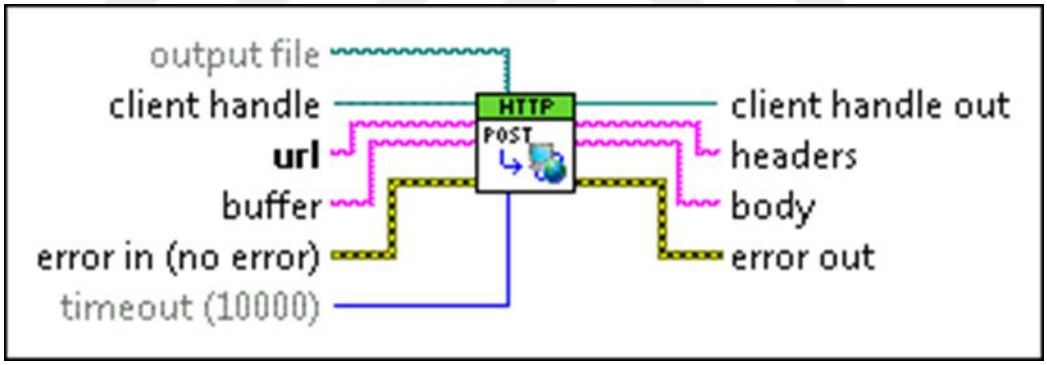

Figure 41. HTTP POST function

Figure 42 explains the PUT function for sending a request to modify the data sent from the web. Where it has inputs and outputs similar to POST.VI function but differ in task.

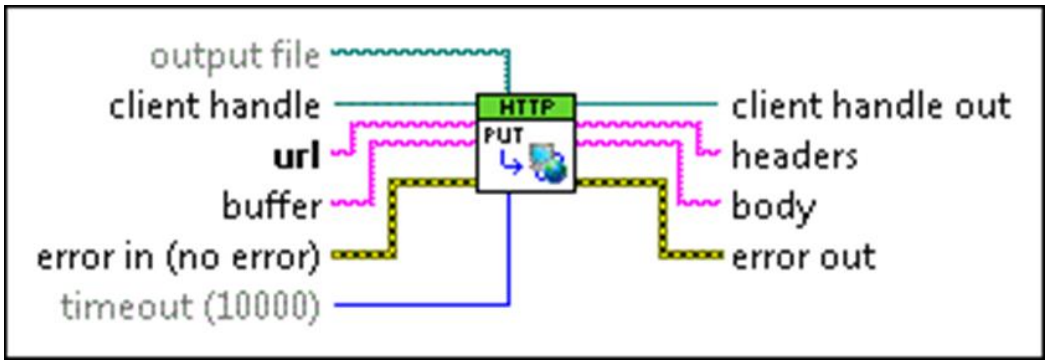

Figure 42. HTTP PUT function

Figure 43, shows the GET.VI function, which sends a request from the server to the web for readings data. It contains inputs and outputs similar to POST.VI function and PUT.VI function, but they differ in tasks.

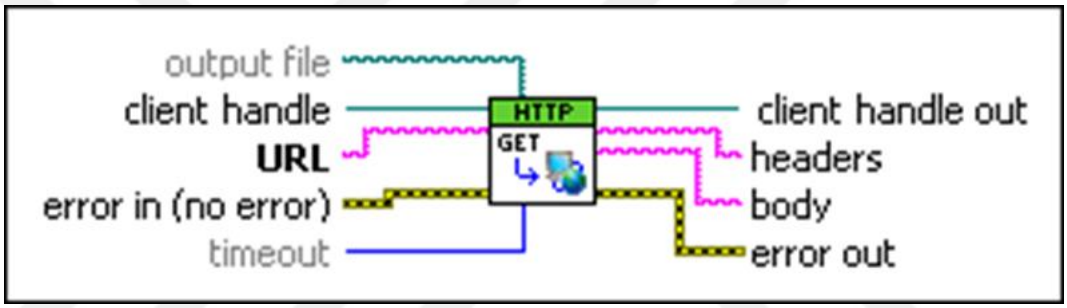

Figure 43. HTTP GET function

Input and Output for all functions mentioned above:

- When a server returns body data, an “output file” indicates where this data should be saved. The VI does not save the body file since it does not specify an output file.
- "The client handle" provides the client handle to be associated with the Web request. it can connect several HTTP client VIs using client handles while keeping authentication credentials, HTTP headers, and cookies. When making a single Web request without any persistent data, such as headers or credentials, a client handle is not necessary.
- This VI delivers a web request to the web page, server, or web service specified by the “URL”.
- Error situations that occur before this node runs are described by "error in." In terms of functionality, this input delivers standard error.

- The "timeout" parameter sets the amount of time in milliseconds that the Web request must wait for a response from the server before it times out. 10000 milliseconds are the default setting.
- The client handle connected with the Web request is returned by "client handle out." To connect many HTTP client VIs while keeping authentication credentials, HTTP headers, and cookies, use client handles. When making independent Web requests without persistent data such as headers or credentials, client handles are not required.
- The header fields provided by the server are returned by "headers".
- "body" is a function that returns the body data that is returned by the server.
- "error out" is a variable that holds information about errors. The standard error-out functionality is provided by this output.

The circuit is designed for the HTTP protocol, where the GET function is used, as shown in figure 44. A database is created by the Firebase platform, where this platform supports sending data in real time. Virtual data set in the platform is sent to the circuit designed for the protocol, then the delay time and size of the received data packet are measured in waves.

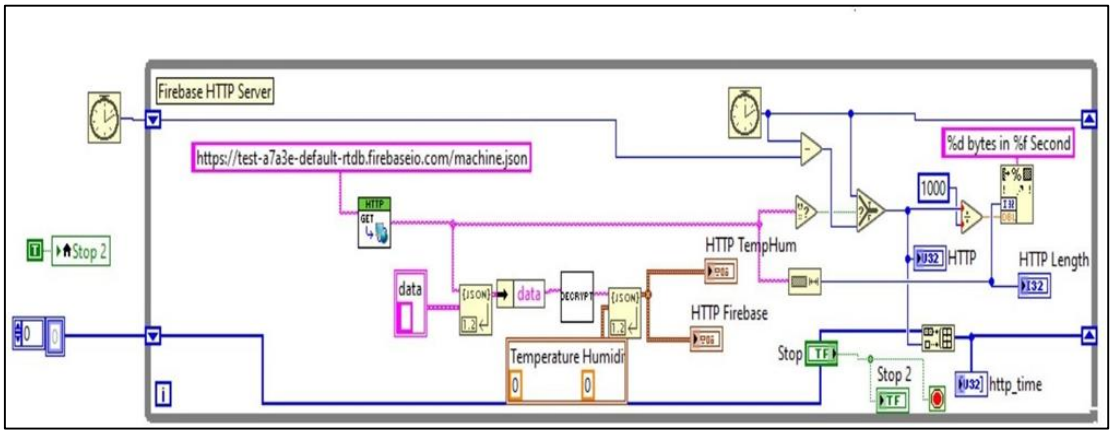

Figure 44. Block diagram of HTTP protocol circuit in LabView

CHAPTER FIVE
IMPLEMENTATION AND SIMULATION RESULTS

5.1. Hardware Implementation Results

The usual network design in an IoT system is as follows: First, the terminal device or sensor gathering data or signals. For devices that cannot be connected to the Internet or the Intranet network, the sensor first sends the collected information to the IoT gateway, which then delivers the information to the server. Some devices, such as mobile phones, have their own functionalities for connecting to the network.

The definition of each sensor, the fundamentals of a remote oil pipeline monitoring system, how to construct them, and communication and IoT technologies have all been explored in earlier chapters. This chapter will go over the results that are acquired.

5.1.1. Temperature measurement

Thermowells are hollow tubes having a closed one end and a threaded other. They are permanently mounted in pipes so that temperature measuring sensors can be added to measure the liquid's temperature. When the power supply is turned on, all gadgets are powered up. Temperature sensors detect the current temperature and send it to the Arduino UNO. It employs a DHT11 temperature sensor, which collects temperature and humidity data and sends it to the NodeMCU once it has been processed by the Arduino UNO, the signal output of this sensor is a digital signal. The data format used by DHT11 is a single bus data format. That is, input/output two-way transmission is completed by a single data port. The size of a single packet is 5 bytes (40 Bit). The communication time is under 3 milliseconds. The system starts working as soon as the power is turned on, and data is collected at regular intervals. As illustrated in figure 45, the results are shown in the webserver, in figure (b) the x-axis of the real-time curve is represented, and the y-axis of the curve is amplitude (temperature in Celsius unit), in figure (a), the figure shows the temperature gauge at time 20:32:36.

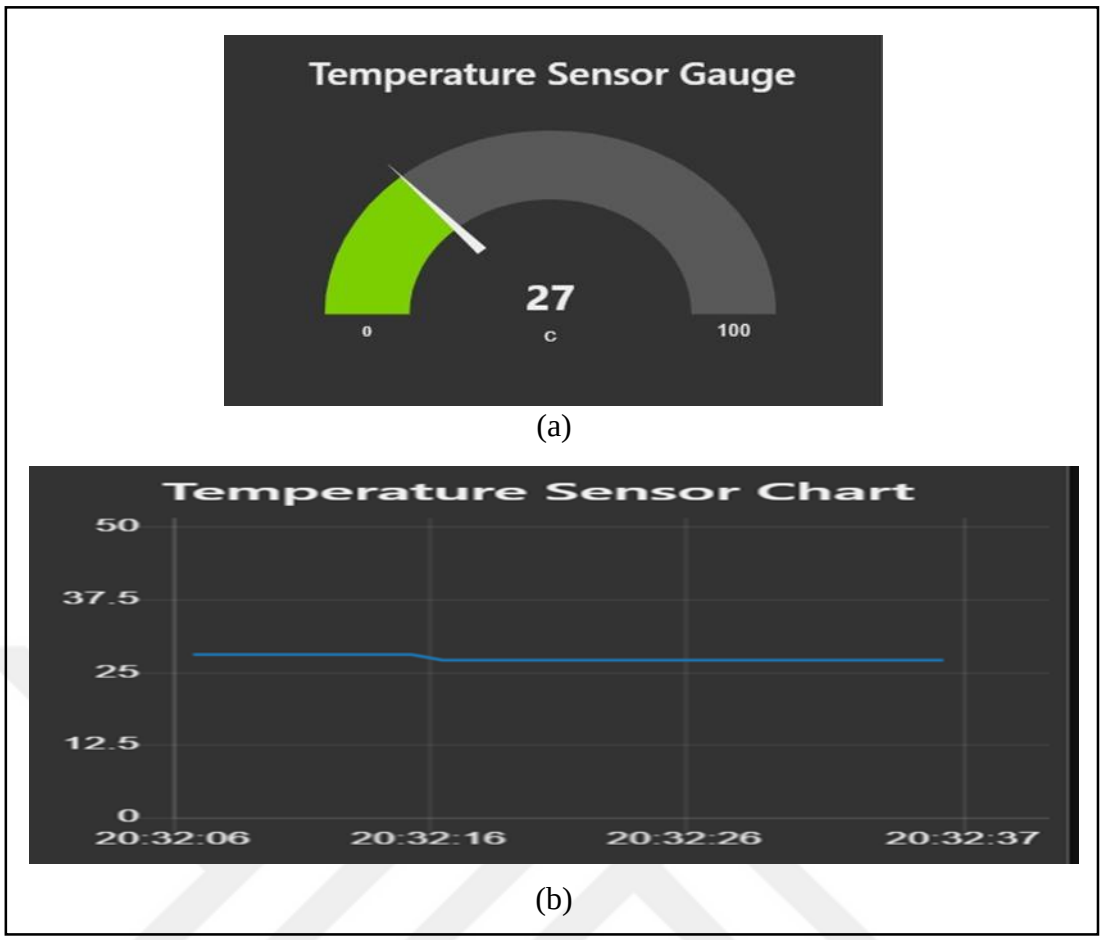

Figure 45. Temperature result in web-server

5.1.2. Flow meter measurement

Liquid flow meters measure the velocity of the liquid flowing through the tube, the unit of measurement used is l/min, and it is sent to the Arduino UNO. Here the YF-S201 Hall effect water flowmeter is used as the fluid flow sensor inside the tube, the rotor blades rotate using energy from the flow stream, and the rotor rotates faster as the velocity of the liquid increases, the data output signal acquired by this sensor is digital. The results appear in the web server as shown in figure 46, the curve in figure (b) represents the velocity of the fluid inside the pipe, where the x-axis of the curve represents the real time to read the data, while the y-axis represents the capacity value (liters per minute), in figure (a) shows the flow velocity gauge at the time 14:14:25.

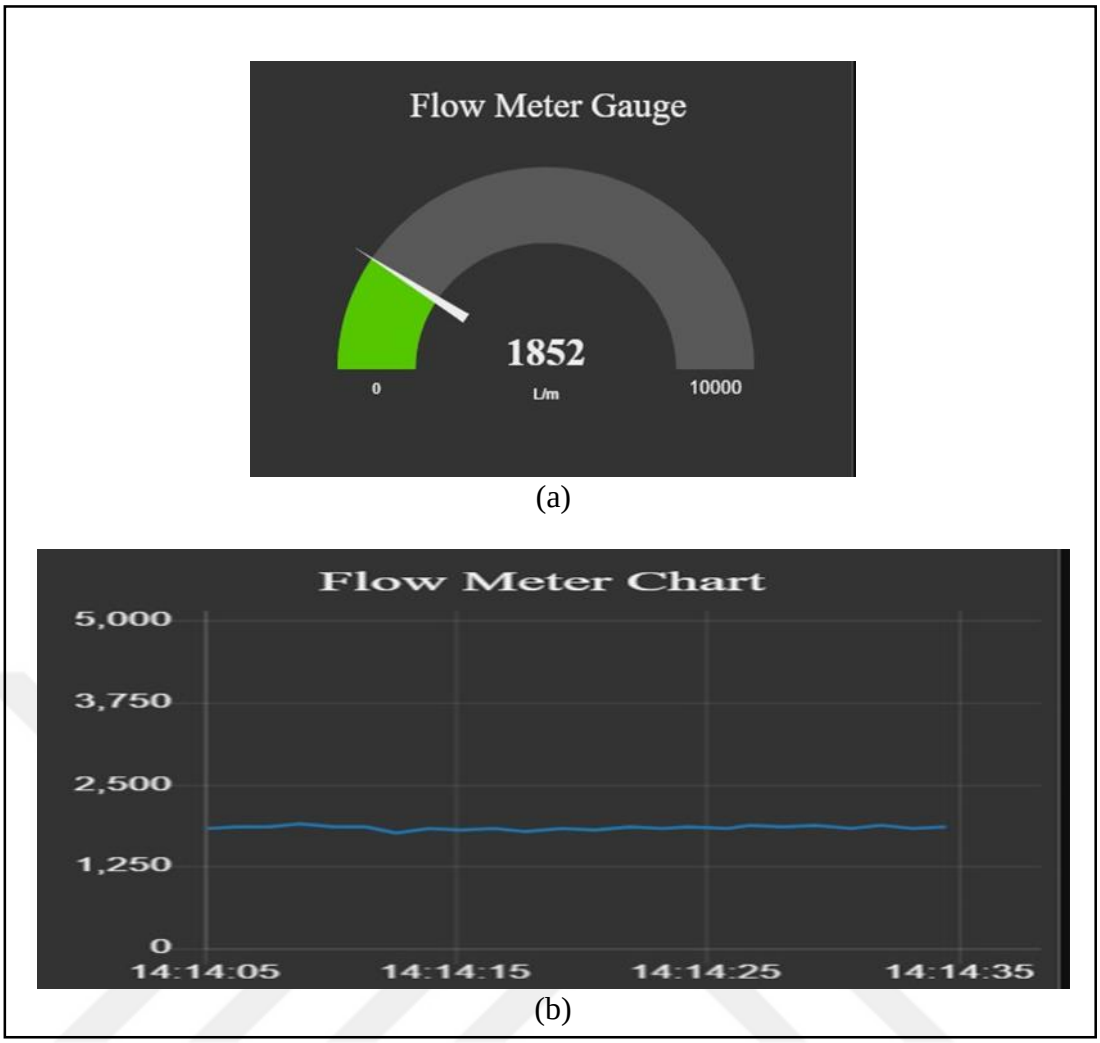

Figure 46. Flow meter result in web-server

5.1.3. Pressure measurement

Damages caused by impulsive events typically result in a pressure pulse traveling in both directions through the liquid in the pipe. These pressure pulses can be noticed and quantified from a long distance away. The information included in these recorded pulses can be utilized to monitor pipelines in order to detect and pinpoint damages. The pressure sensors measure the current amount of pressure and then send it to the Arduino Uno. Here the pressure sensor BMP180 is used as a sensor to measurement, the data output signal acquired by this sensor is analog. The results appear in the web server, as shown in figure 47, the x-axis of the curve in figure (b) represents the real-time, the y-axis of the curve represents the real-time pressure sensor data (Pascals unit). figure (a) shows the pressure gauge in pascals unit at the time 03:28:54.

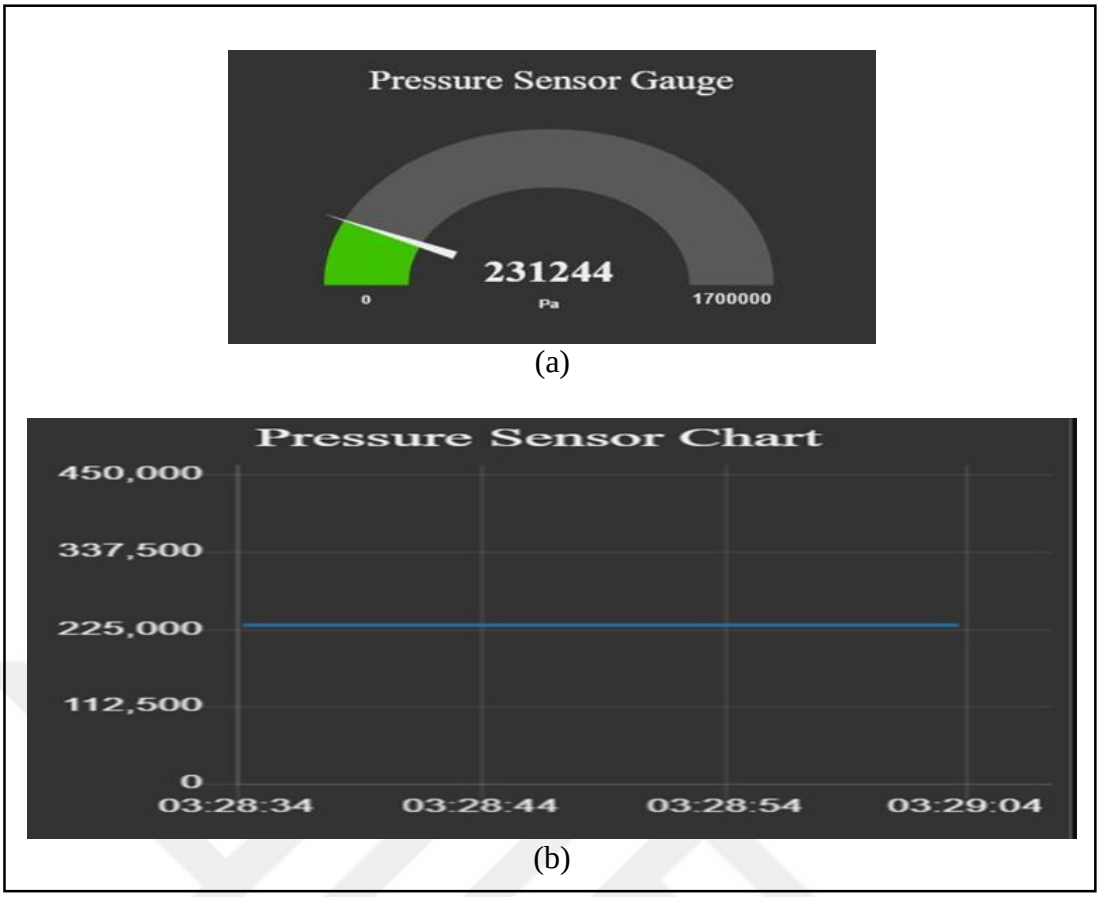

Figure 47. Pressure result in web-server

5.1.4. Gas leaking measurement

Gas sensors detect current gas leaks and send them to the Arduino UNO. Here Gas-MQ2 is used as a sensor that senses the amount of gas leakage, the output signal of the data acquired from the gas leak sensor is analog signal, the results appear in the web server, as shown in figure 48, where the x-axis of the curve in figure (b) represents real time For the data acquired from the sensor, the y-axis of the curve represents the percentage of gas leakage, the percentage is directly proportional to the gas leakage, where the greater the amount of gas leakage, the greater the percentage amount, and vice versa. figure (a) shows the gauge of gas leakage at time 00:13:14.

.

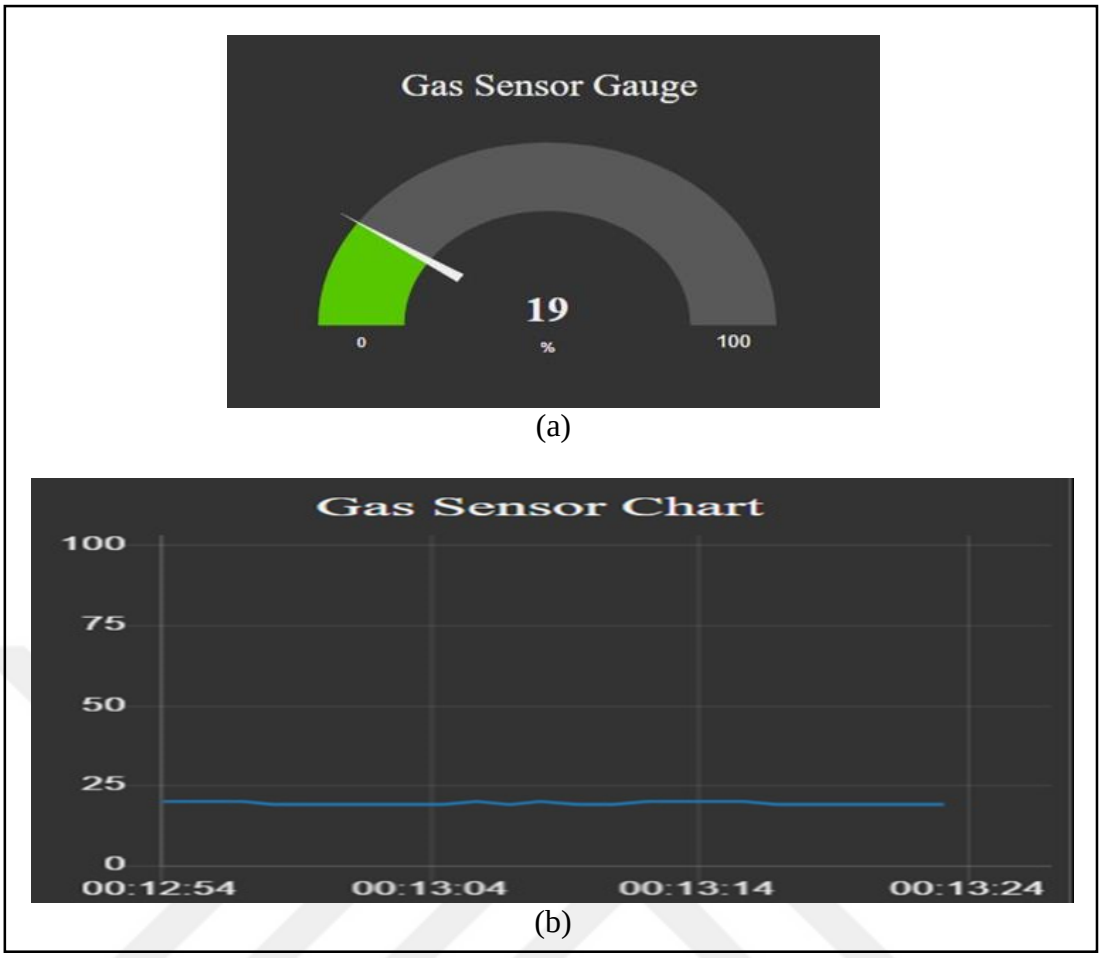

Figure 48. Gas detect result in web-server

5.1.5. Flame detector measurement

Flame detection technology is the first technology used in the explosion proof system. How to quickly and correctly identify fire incidents and transmit explosion information to the monitoring unit in the event of a gas explosion in a pipeline branch is a central topic of study for flame detection technology. Fire sensors detect fires if they occur and send them to the Arduino UNO. Here the flame detector is used as a sensor that senses fires, the sensor's data output signal is analog. The results appear in the web server, as shown in figure 49, the curve in figure (b) represents the acquired data from the sensor, where the x-axis of the curve represents the real-time data of the sensor, the y- axis is the percentage of a distance between the node of system and the fire place, each the higher the percentage means the location of the fire is close to the node of system, figure (a) shows the gauge of flame detection at time 21:49:27.

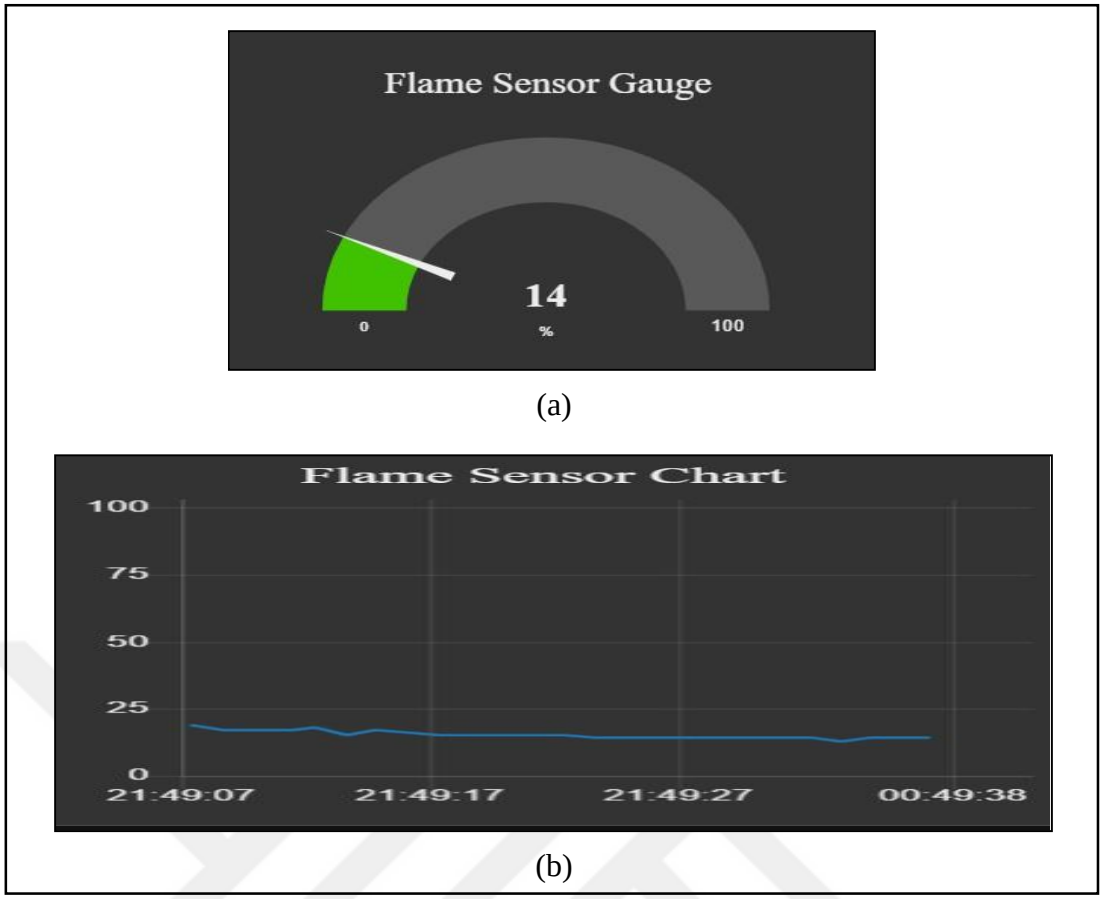

Figure 49. Flame detect result in web-server

5.2. Smartphone application

The front-end and back-end make up the application architecture. The term "back-end" refers to a server (cloud computing) that houses a database that saves IoT data as well as software that does not interact directly with users. The front-end is a client part. It includes sensor gauges and graphs for the proposed system. This application automatically refreshes for new readings at regular intervals. The mobile application can display sensor information in real time.

In figure 50, the temperature information is shown, the x-axis of the curve in figure (b) represents the real time that the temperature sensor data is acquired, the y-axis represents the temperature value of the sensor, the temperature gauge is also displayed, in figure (a) the temperature gauge shows temperature at time 23:47:47.

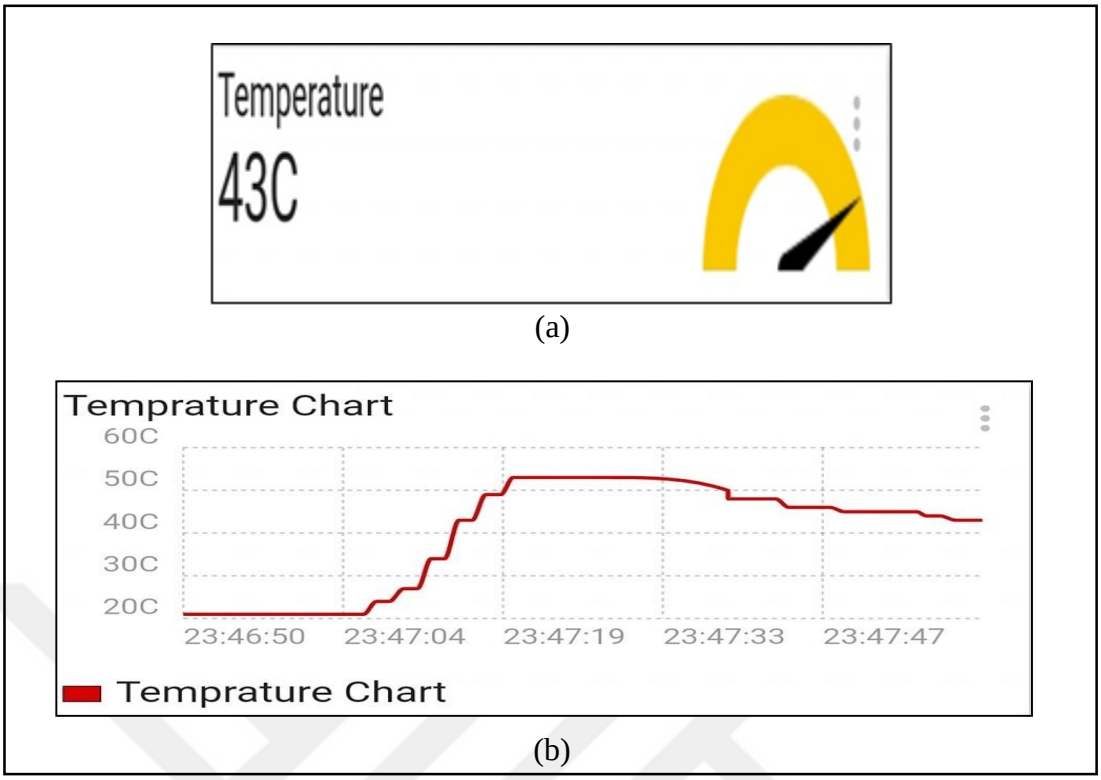

Figure 50. Temperature result in mobile application

In figure 51 the flow meter information is shown, the x-axis of the curve in figure (b) represents the real time that the liquid flow meter data was acquired, the y-axis represents the fluid flow velocity (liters/min). The flow velocity is also displayed as a gauge, in figure (a) the flow velocity gauge is shown at time 18:52:22.

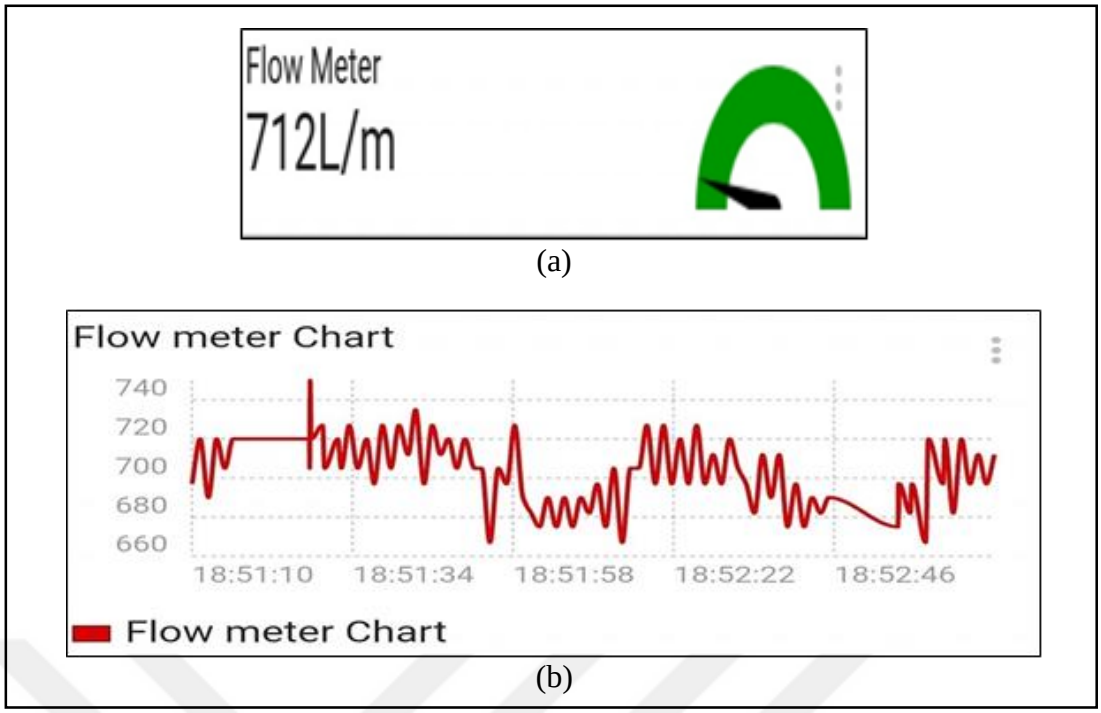

Figure 51. Flow meter result in mobile application

In figure 52 shows the pressure information, the x-axis of the curve in figure (b) represents the real time that the pressure sensor data was acquired, the y-axis represents the pressure value. The pressure value is displayed as a gauge as well, in figure (a) the gauge shows the pressure value at the time 03:36:11.

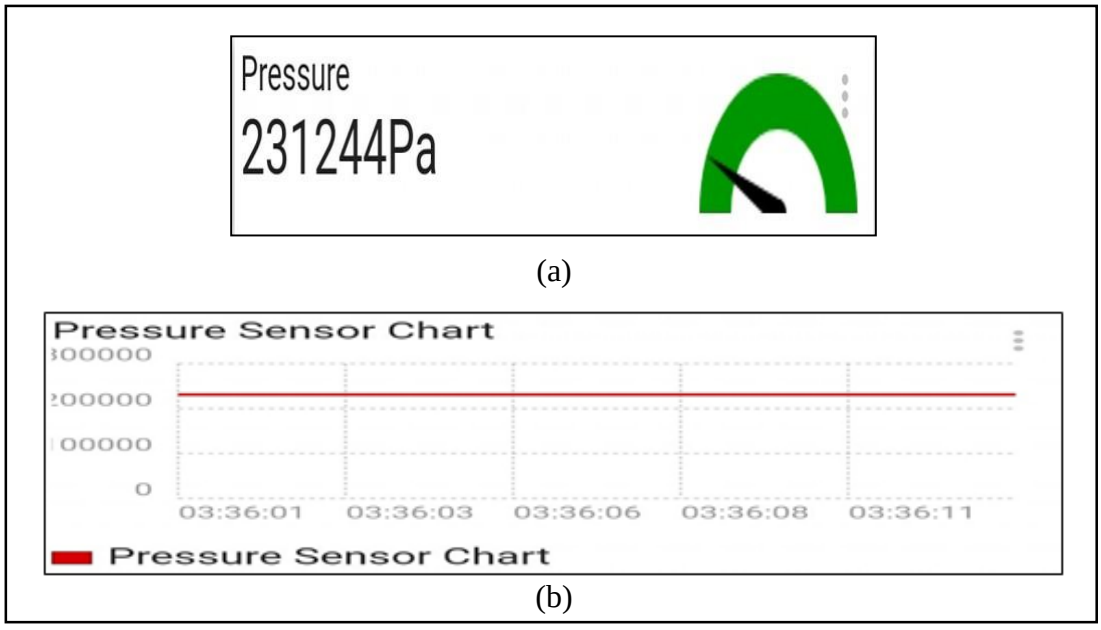

Figure 52. Pressure result in mobile application

In figure 53, the information of the gas leakage sensor is shown, the x-axis of the curve in figure (b) represents the real time that the gas leakage sensor data is acquired, the y-axis represents the percentage of the gas leakage. The gas leakage is also displayed as a gauge, in figure (a) showing the gas leakage rate at the time 00:23:06.

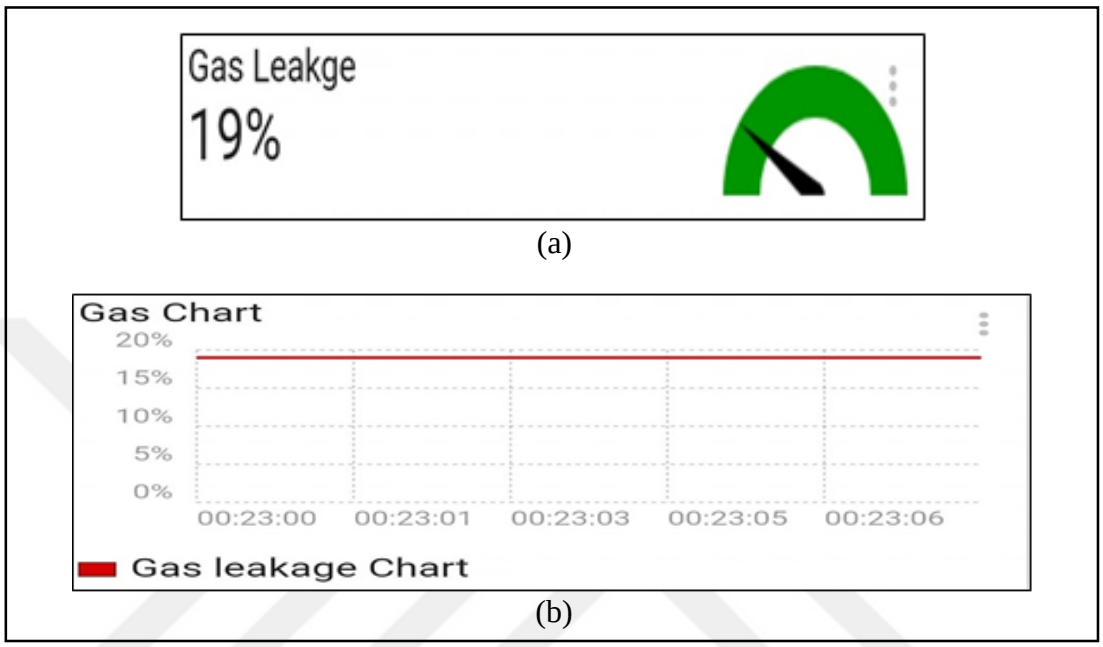

Figure 53. Gas detect result in mobile application

In figure 54 shows the flame detection information, the x-axis of the curve in figure (b) represents the real time that the flame detection sensor data was acquired, the y- axis represents the percentage of the distance between the flame location and the system node. The flame detection ratio is also displayed as a gauge, in figure (a) the flame detection gauge at the time 00:53:00 is shown.

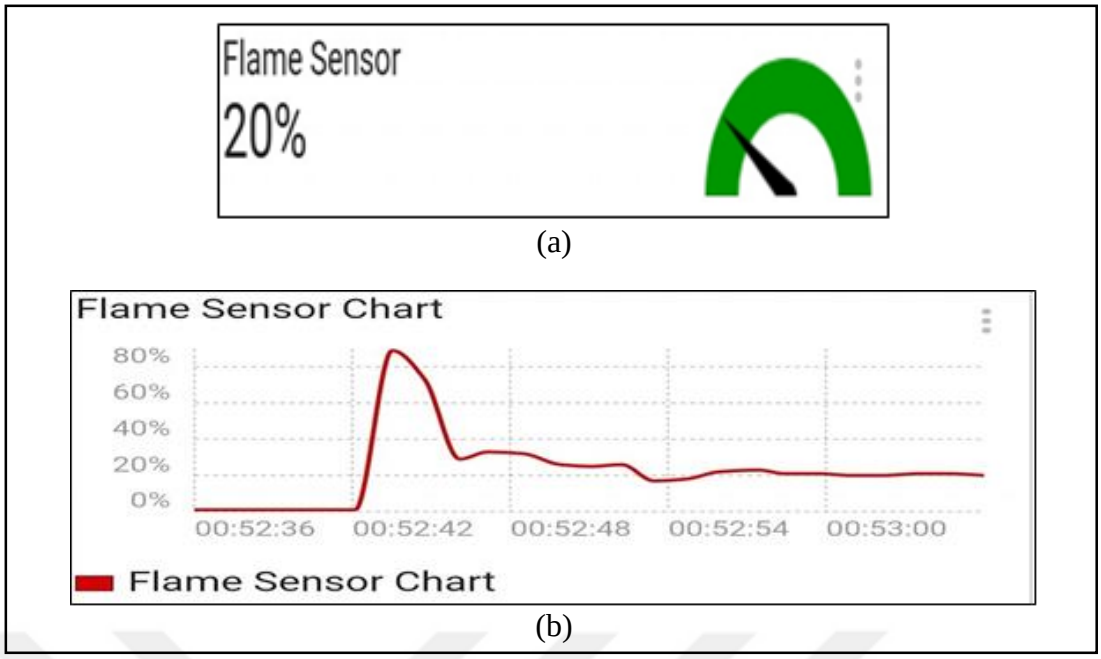

Figure 54. Flame detect result in mobile application

5.3. Results of simulation

Some communication protocols, such as MQTT, can be used instead of HTTP in order to reduce the size of the server's memory footprint. HTTP has a huge header because every time data is transferred, a TCP packet is sent to connect/disconnect, hence the more data is sent, the more traffic is generated.

The header of MQTT is relatively small, and it can also send and receive the next data while maintaining the TCP connection, so it can suppress the total data traffic more than HTTP.

In addition, when using MQTT, one should also pay attention to that, while maintaining the TCP connection of MQTT, the data should be sent and received. Because MQTT reduces the amount of communication by maintaining a TCP connection if disconnect the TCP connection every time data communication is performed, MQTT will perform the connection and disconnection processing every time data is sent, just like HTTP, but the result will increase communications volume. A comparison has been tested between MQTT and HTTP protocols in terms of delay, size of packets, consumption power, and oscillations.

Figure 55, explained the difference between HTTP protocol and MQTT protocol in terms of delay, the x-axis of the curve represents the number of samples that are being transported, the y-axis represents the delay time of the protocols when transporting samples, MQTT protocol has better outperforms than HTTP protocol. Where MQTT

protocol has less delay than HTTP protocol. The large delay in time when transferring data in HTTP is due to the large protocol header size compared to MQTT, default HTTP header size is 8 bytes where MQTT has a small header size of only 2 bytes.

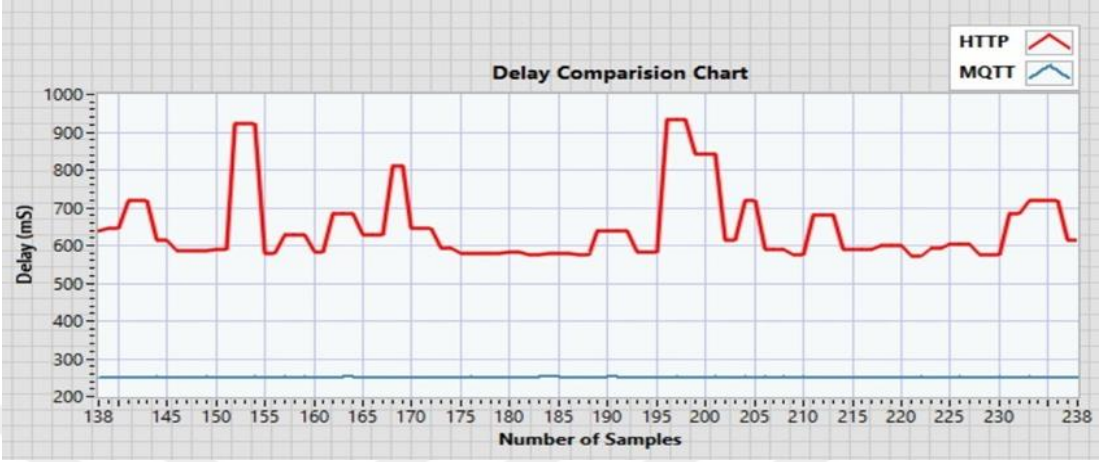

Figure 55. The comparison of delay between MQTT and HTTP

Figure 56, explained the difference between HTTP protocol and MQTT protocol in terms of size packet, the x-axis of the curve represents the number of samples that are being transported, The y-axis represents the size of the packet transmitted by the protocol, HTTP protocol has better outperforms than MQTT protocol. Where HTTP has 260 bytes of the packet, While MQTT has only 13 bytes. The reason for the small packet size in MQTT is because it uses binary format, while HTTP uses ASCII (American Standard Code for Information Interchange) format.

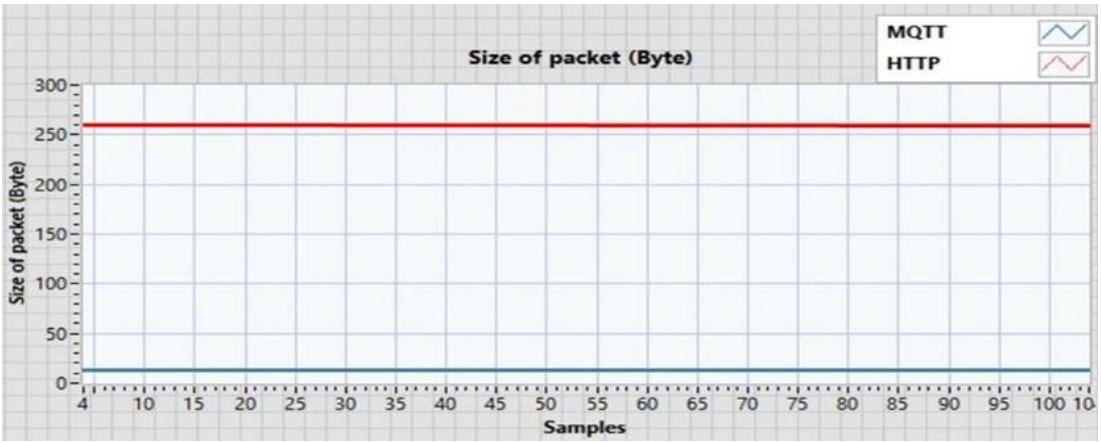

Figure 56. Comparison of the size packet between MQTT and HTTP

Figure 57 clarified the amount of oscillations inside MQTT and HTTP protocols, where the oscillations in MQTT are less than HTTP, that's mean the MQTT protocol has efficiency much more than HTTP protocol, where the MQTT protocol

supports QoS, unlike the HTTP protocol. The reliability of the protocols depended on the oscillations, whenever the oscillations are high leads the reliability to be low and vice versa.

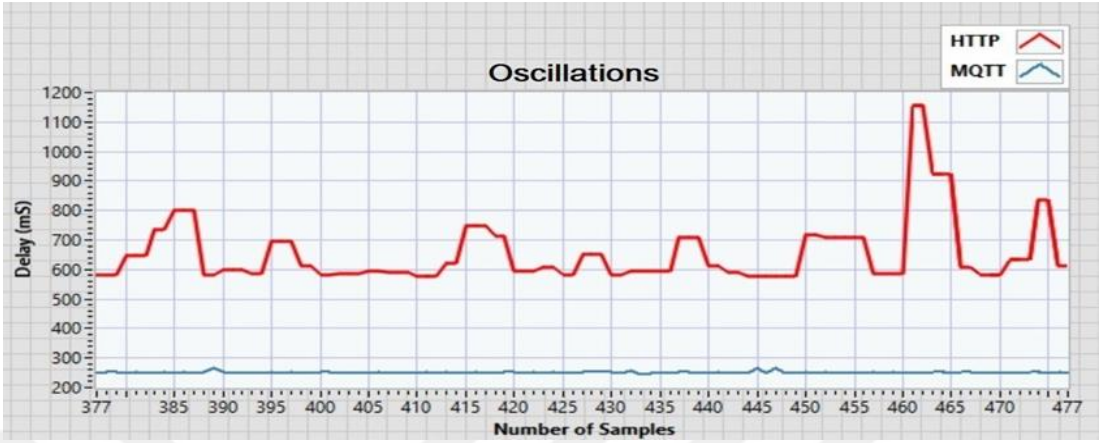

Figure 57. Comparison of the oscillations between MQTT and HTTP

Figures 58, 59, are explained the temperature and humidity in HTTP protocol and MQTT protocol. Where figure 44 shows the temperature and humidity for the MQTT protocol while figure 45 shows the temperature and humidity for HTTP protocol.

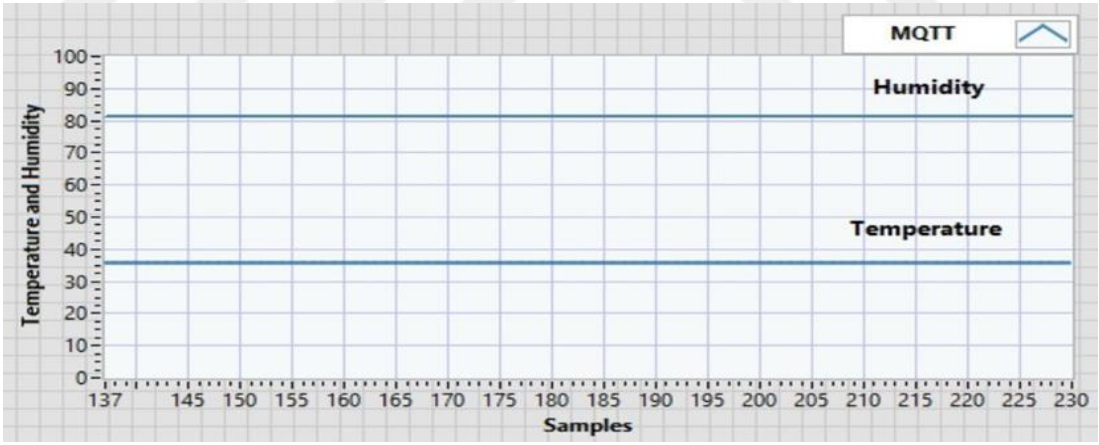

Figure 58. The levels of Temperature & Humidity in MQTT protocol

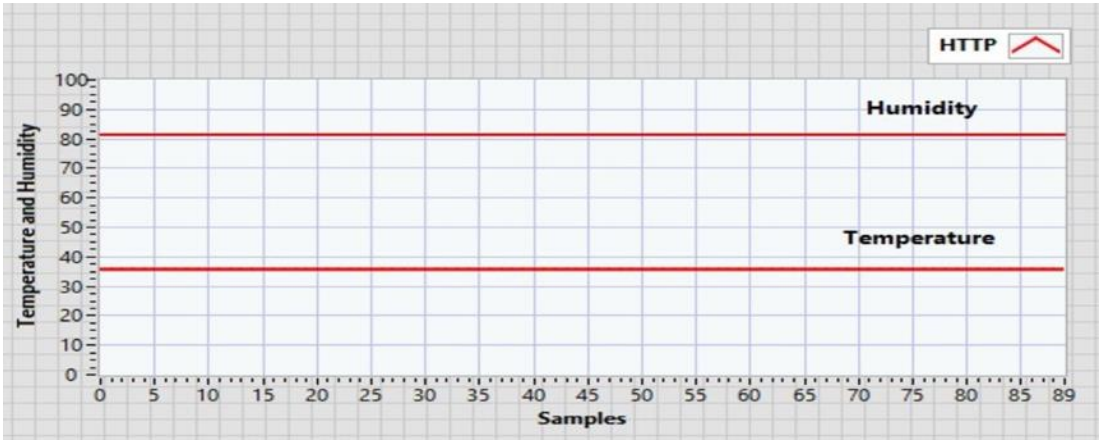

Figure 59. The levels of Temperature & Humidity in HTTP protocol

In tables 4,5, explained the difference between MQTT and HTTP protocols in terms of Samples per Minute, Delay, Size of Packet, Reliability, Oscillations, and power consumption.

Finally, Through curves and tables. It deduces that the MQTT protocol is superior to the HTTP protocol in several aspects, the most important of which is the speed of data transfer between the server and the client. the MQTT protocol consumes less power compared to the HTTP protocol when transferring samples. MQTT is more reliable and has less oscillation in response time. In terms of the length of the packet, the HTTP protocol has better than the MQTT protocol.

Table 4. Shows the results of a comparison between MQTT and HTTP

	HTTP	**MQTT**
Samples per Minute	89	236
Maximum delay(ms)	1685	254
Minimum delay (ms)	570	249
Avg. Delay (ms)	663.6629	250.23
Size of Packet (Byte)	260	13

Table 5. Shows the results of a comparison between MQTT and HTTP

	HTTP	**MQTT**
Delay	High	Low
Size of Packet	High	Low
Reliability	Low	High
Oscillations	High	Low
Power consumption	High	Low

CHAPTER SIX
CONCLUSIONS AND FUTURE WORK

6.1. CONCLUSIONS

In this study, an investigation is conducted on the application of the Internet of Things in industrial fields. Recently, interest in the IoT has been given to petroleum pipeline systems that monitor pipelines remotely using the Internet, in order to enhance their monitoring process by using the communication opportunity offered by the Internet.

The design of the IoT oil pipeline monitoring system is characterized by its low cost, flexibility, and reliability compared to classical SCADA systems. The proposed IoT-based SCADA system is suitable for remote monitoring and control of the oil pipeline system. Where the sensors are monitored in real-time via a computer or via a smartphone application through dashboards. The proposed system is implemented in the laboratory environment using various sensors and microcontrollers and displays the results. The system operates in three stages. The first stage is the system monitoring of temperature, pressure, gas leakage, fire detection, and the amount of liquid flow inside the pipe, carrying the data to the database storage using the MQTT and Node-Red protocol installed on the NodeMCU, and then to a dashboard connected to the database. In the second stage, the system controls the servo motor whereby the user can control its movement using a microcontroller (NodeMCU). Finally, in the third stage, supervisory control is achieved. The end-user can monitor and control the system through the web interface or the mobile application. The cost of the developed system is low, easy to operate, and simply integrated with the devices of the proposed system.

A simulation of the MQTT protocol for the proposed system is carried out using the Proteus and the LabView programs to make a comparison between the MQTT protocol with the HTTP protocol, the comparison included the delay time, the transmitted data packet size, and oscillations. In sum, the MQTT protocol outperforms the HTTP protocol in many aspects, so the MQTT protocol is a suitable protocol to be used in the proposed system in this thesis.

6.2. FUTURE WORK

There are several suggestions for future development, including:

1. Using Low-Power Wide-Area Network (LPWAN) instead of Wi-Fi as communication technology.
2. More flexible and intelligent sensors can be used at a low cost, which can be combined with the proposed system to make it more reliable, allowing it to be used and distributed on a large scale.
3. Focus on developing an IoT architecture in pipeline monitoring system with integration of new protocols such as CoAP, XMPP etc.

REFERENCES

Aba, E. N., Olugboji, O. A., Nasir, A., Olutoye, M. A., & Adedipe, O. (2021). Petroleum pipeline monitoring using an internet of things (IoT) platform. SN Applied Sciences, 3(2), 1-12

Abhyankar, V., Singh, A. G., Paul, P., Mehta, A., & Vidhya, S. (2019, March). Portable Autonomous Rain Prediction Model Using Machine Learning Algorithm. In 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN) (pp. 1-4). IEEE.

Abomhara, M., & Køien, G. M. (2014, May). Security and privacy in the Internet of Things: Current status and open issues. In 2014 international conference on privacy and security in mobile systems (PRISMS) (pp. 1-8). IEEE.

Aghenta, L. O., & Iqbal, M. T. (2019). Low-cost, open source IoT-based SCADA system design using thinger. IO and ESP32 thing. Electronics, 8(8), 822. pp 1-5.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. IEEE communications surveys & tutorials, 17(4), 2347-2376.

Al-Masri, E., Kalyanam, K. R., Batts, J., Kim, J., Singh, S., Vo, T., & Yan, C. (2020). Investigating messaging protocols for the Internet of Things (IoT). IEEE Access, 8, 94880-94911.

Amali, L. Y., & Batan, I. M. L. (2020). Design of a Patient Wrist Rehabilitation Device with Servo Motor Drive. JMES The International Journal of Mechanical Engineering and Sciences, 4(2), 28-37.

Anusha, D., Sarma, P. M., & SandhyaRani, M. N. (2013). Appliance remote control using Arduino. International Journal of Latest Trends in Engineering and Technology (IJLTET), 2(4), 35-41.

Avhad, M., Divekar, V., Golatkar, H., & Joshi, S. (2013, December). Microcontroller based automation system using industry standard SCADA. In 2013 Annual IEEE India Conference (INDICON) (pp. 1-6). IEEE.

Badamasi, Y. A. (2014, September). The working principle of an Arduino. In 2014 11th international conference on electronics, computer and computation (ICECCO) (pp. 1-4). IEEE.

Borhani, M., Liyanage, M., Sodhro, A. H., Kumar, P., Jurcut, A. D., & Gurtov, A. (2020). Secure and resilient communications in the industrial internet. In Guide to Disaster-Resilient Communication Networks (pp. 219-242). Springer, Cham.

Bryce, R., Shaw, T., & Srivastava, G. (2018, July). Mqtt-g: A publish/subscribe protocol with geolocation. In 2018 41st international conference on telecommunications and signal processing (TSP) (pp. 1-4). IEEE.

Butun, I. (2020). Industrial IoT. Springer International Publishing.

Ch, S. M., Abdul, N. M., & Pendem, S. (2020). Smart jacket for health monitoring using LabVIEW. Materials Today: Proceedings. pp 1-6

Chavala, L. N., Singh, R., & Gehlot, A. (2022). Performance evaluation of LoRa based sensor node and gateway architecture for oil pipeline management. International Journal of Electrical and Computer Engineering, 12(1), 974.

Chen, A., & Dong, H. (2021, May). A Pipeline On-line Detection and Control System Based on the IoT. In 2021 IEEE 4th International Conference on Electronics Technology (ICET) (pp. 1095-1098). IEEE.

Chopra, K., Gupta, K., & Lambora, A. (2019, February). Future internet: The internet of things-a literature review. In 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon) (pp. 135-139). IEEE.

Diwanji, M., Hisvankar, S., & Khandelwal, C. (2019, September). Autonomous fire detecting and extinguishing robot. In 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT) (pp. 327-329). IEEE.

Feng, C., Palleti, V. R., Mathur, A., & Chana, D. (2019, February). A Systematic Framework to Generate Invariants for Anomaly Detection in Industrial Control Systems. In NDSS.

Ferencz, K., & Domokos, J. (2019). Using Node-RED platform in an industrial environment. XXXV. Jubileumi Kandó Konferencia, Budapest, 52-63.

Goel, A., & Mishra, R. S. (2009). Remote data acquisition using wireless-SCADA system. International Journal of Engineering (IJE), 3(1), 58-65.

Grgić, K., Špeh, I., & Heđi, I. (2016, October). A web-based IoT solution for monitoring data using MQTT protocol. In 2016 international conference on smart systems and technologies (SST) (pp. 249-253). IEEE.

Grigorik, I. (2013). Making the web faster with HTTP 2.0. Communications of the ACM, 56(12), 42-49.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.

Gupta, B. B., & Quamara, M. (2020). An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols. Concurrency and Computation: Practice and Experience, 32(21), e4946.

Han, N. S. (2015). Semantic service provisioning for 6LoWPAN: powering internet of things applications on Web (Doctoral dissertation, Institut National des Télécommunications).

Hassan, Q. F., & Madani, S. A. (Eds.). (2017). Internet of things: Challenges, advances, and applications. CRC Press.

Houimli, M., Kahloul, L., & Benaoun, S. (2017, December). Formal specification, verification and evaluation of the MQTT protocol in the Internet of Things. In 2017 International conference on mathematics and information technology (ICMIT) (pp. 214-221). IEEE.

Ismail, A. A., Hamza, H. S., & Kotb, A. M. (2018, December). Performance evaluation of open source iot platforms. In 2018 IEEE global conference on internet of things (GCIoT) (pp. 1-5). IEEE.

Kao, K. C., Chieng, W. H., & Jeng, S. L. (2018, March). Design and development of an IoT-based web application for an intelligent remote SCADA system. In IOP Conference Series: Materials Science and Engineering (Vol. 323, No. 1, p. 012025). IOP Publishing.

Khera, M. N., & Balguvhar, S. (2011). Design of microcontroller based wireless SCADA system for real time data. UACEE International Journal of Advances in Electronics Engineering, 2(1). pp 106-108

Kodali, R. K., & Mandal, S. (2016, December). IoT based weather station. In 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT) (pp. 680-683). IEEE.

Kodali, R. K., & Soratkal, S. (2016, December). MQTT based home automation system using ESP8266. In 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC) (pp. 1-5). IEEE.

Lojka, T., & Zolotová, I. (2014, September). Improvement of human-plant interactivity via industrial cloud-based supervisory control and data acquisition system. In IFIP International Conference on Advances in Production Management Systems (pp. 83-90). Springer, Berlin, Heidelberg.

Lu, X. (2014). Supervisory Control and Data Acquisition System Design for CO2 Enhanced Oil Recovery. Electrical and Computer Sciences.

Maheswari, C., Perinchery, A. A. B., Priyanka, E. B., Ambika, K. S., Narmatha, S., Prenitha, A., & Monisha, M. (2021, February). Review on Online Monitoring and Control in Industrial Automation–An IoT Perspective. In IOP Conference Series: Materials Science and Engineering (Vol. 1055, No. 1, p. 012034). IOP Publishing.

Mahmood, M. K., & Al-Naima, F. M. (2011). An internet based distributed control systems: A case study of oil refineries. Energy and Power Engineering, 3(03), 310-316.

Mahmood, M. K., Al-Naima, F. M., & Uzunoglu, N. K. (2012, April). Design and simulation of a data transmission network for industrial control system subject to reliability improvement. In 2012 International Conference on Future Communication Networks (pp. 23-28). IEEE.

Mahmoud, R., Yousuf, T., Aloul, F., & Zualkernan, I. (2015, December). Internet of things (IoT) security: Current status, challenges and prospective measures. In 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST) (pp. 336-341). IEEE.

Matijevic, M., & Cvjetkovic, V. (2016, February). Overview of architectures with Arduino boards as building blocks for data acquisition and control systems. In 2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV) (pp. 56-63). IEEE.

McClanahan, R. H. (2002, May). The benefits of networked SCADA systems utilizing IP-enabled networks. In 2002 Rural Electric Power Conference. Papers Presented at the 46th Annual Conference (Cat. No. 02CH37360) (pp. C5-1). IEEE.

Medrano, K., Altuve, D., Belloso, K., & Bran, C. (2018, October). Development of SCADA using a RTU based on IoT controller. In 2018 IEEE International Conference on Automation/XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA) (pp. 1-6). IEEE.

Mehta, B., Rengarajan, D., & Prasad, A. (2012). Real time patient tele-monitoring system using LabVIEW. International Journal of Scientific & Engineering Research, 3(4), 1-11.

Mononen, T., & Mattila, J. (2017, November). A low-cost cloud-extended sensor network for supervisory control. In 2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM) (pp. 502-507). IEEE.

Moustafa, N., Turnbull, B., & Choo, K. K. R. (2018). An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Internet of Things Journal, 6(3), 4815-4830.

Mumtaz, S., Alsohaily, A., Pang, Z., Rayes, A., Tsang, K. F., & Rodriguez, J. (2017). Massive Internet of Things for industrial applications: Addressing wireless IIoT connectivity challenges and ecosystem fragmentation. IEEE Industrial Electronics Magazine, 11(1), 28-33.

Naik, N., & Jenkins, P. (2016, July). Web protocols and challenges of web latency in the web of things. In 2016 Eighth International Conference on ubiquitous and future networks (ICUFN) (pp. 845-850). IEEE.

Naik, N., Jenkins, P., Davies, P., & Newell, D. (2016, December). Native web communication protocols and their effects on the performance of web services and systems. In 2016 IEEE International Conference on Computer and Information Technology (CIT) (pp. 219-225). IEEE.

Nasir, M. A., Naidoo, L., Shahbaz, M., & Amoo, N. (2018). Implications of oil prices shocks for the major emerging economies: A comparative analysis of BRICS. Energy Economics, 76, 76-88.

Osabutey, D., Obro-Adibo, G., Agbodohu, W., & Kumi, P. (2013). Analysis of risk management practices in the oil and gas industry in Ghana. Case study of Tema Oil Refinery (TOR). European journal of business and management, 5(29), 139-149.

Pandey, R. C., Verma, M., Sahu, L. K., & Deshmukh, S. (2017). Internet of things (IOT) based gas leakage monitoring and alerting system with MQ-2 sensor. International Journal of Engineering Development and Research, 5(2), 2135-2137.

Phuyal, S., Bista, D., Izykowski, J., & Bista, R. (2020). Design and implementation of cost-efficient SCADA system for industrial automation. International Journal of Engineering and Manufacturing, 10(2), 15.

Pierleoni, P., Concetti, R., Belli, A., & Palma, L. (2019). Amazon, Google and Microsoft solutions for IoT: architectures and a performance comparison. IEEE access, 8, 5455-5470.

Pirbhulal, S., Zhang, H., E Alahi, M. E., Ghayvat, H., Mukhopadhyay, S. C., Zhang, Y. T., & Wu, W. (2017). A novel secure IoT-based smart home automation system using a wireless sensor network. Sensors, 17(1), 69.

Priyanka, E. B., Maheswari, C., & Thangavel, S. (2021). A smart-integrated IoT module for intelligent transportation in oil industry. International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 34(3), e2731.

Priyanka, E. B., Thangavel, S., & Gao, X. Z. (2021). Review analysis on cloud computing based smart grid technology in the oil pipeline sensor network system. Petroleum Research, 6(1), 77-90.

Priyanka, E. B., Thangavel, S., & Gao, X. Z. (2021). Review analysis on cloud computing based smart grid technology in the oil pipeline sensor network system. Petroleum Research, 6(1), 77-90.

Prokhorov, A. S., Chudinov, M. A., & Bondarev, S. E. (2018, January). Control systems software implementation using open-source SCADA-system OpenSCADA. In 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (pp. 220-222). IEEE.

Puviarasi, R., Dineshkumar, T., Babu, V. S., & Partheeban, P. (2021, July). Design and Development of Smart Portable System for Underground Pipeline Leakage Monitoring Based on IoT. In Journal of Physics: Conference Series (Vol. 1964, No. 6, p. 062065). IOP Publishing.

Rakas, S. V. B., Stojanović, M. D., & Marković-Petrović, J. D. (2020). A review of research work on network-based scada intrusion detection systems. IEEE Access, 8, 93083-93108.

Rani, P. J., Bakthakumar, J., Kumaar, B. P., Kumaar, U. P., & Kumar, S. (2017, March). Voice controlled home automation system using Natural Language Processing (NLP) and Internet of Things (IoT). In 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM) (pp. 368-373). IEEE.

Rao, B. S., Rao, K. S., & Ome, N. (2016). Internet of Things (IoT) based weather monitoring system. international journal of advanced research in computer and communication engineering, 5(9), 312-319.

Sahadevan, A., Mathew, D., Mookathana, J., & Jose, B. A. (2017, June). An offline online strategy for IoT using MQTT. In 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud) (pp. 369-373). IEEE.

Sajid, A., Abbas, H., & Saleem, K. (2016). Cloud-assisted IoT-based SCADA systems security: A review of the state of the art and future challenges. IEEE Access, 4, 1375-1384.

Sarierao, B. S., & Prakasarao, A. (2018, April). Smart healthcare monitoring system using mqtt protocol. In 2018 3rd International Conference for Convergence in Technology (I2CT) (pp. 1-5). IEEE.

Sasmita, E. S., Rosmiati, M., & Rizal, M. F. (2018, December). Integrating Forest Fire Detection with Wireless Sensor Network Based on Long Range Radio. In 2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC) (pp. 222-225). IEEE.

Sayed, K., & Gabbar, H. A. (2017). SCADA and smart energy grid control automation. In Smart energy grid engineering (pp. 481-514). Academic Press.

Shahzad, A., Musa, S., Aborujilah, A., & Irfan, M. (2014). The SCADA review: system components, architecture, protocols and future security trends. American Journal of Applied Sciences, 11(8), 1418.

Sharmila, F. M., Suryaganesh, P., Abishek, M., & Benny, U. (2019, March). Iot Based Smart Window using Sensor Dht11. In 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS) (pp. 782-784). IEEE.

Shi, W., Yang, Z., & Li, K. X. (2013). The impact of crude oil price on the tanker market. Maritime Policy & Management, 40(4), 309-322.

Shoja, S., & Jalali, A. (2017, December). A study of the Internet of Things in the oil and gas industry. In 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI) (pp. 0230-0236). IEEE.

Singh, J., Tiwari, M., & Shrivastava, M. (2013). Industrial automation—A review. International journal of Engineering Trends and Technology, 4, 3516-3520.

Singh, P., & Saikia, S. (2016, December). Arduino-based smart irrigation using water flow sensor, soil moisture sensor, temperature sensor and ESP8266 WiFi module. In2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC) (pp. 1-4). IEEE.

Statista, I. H. S. (2018). Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions). URL: https://www. statista. com/statistics/471264/iot-number-of-connected-devicesworldwide/(Consulté 17/05/2020).

Stouffer, K. A., Falco, J. A., & Scarfone, K. A. (2011). Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc). pp 1-241

Sultana, T., & Wahid, K. A. (2019). Choice of application layer protocols for next generation video surveillance using internet of video things. IEEE Access, 7, 41607-41624.

Trung, D. (1995, September). Modern SCADA systems for oil pipelines. In Industry Applications Society 42nd Annual Petroleum and Chemical Industry Conference (pp. 299-305). IEEE.

Wan, Z., Song, Y., & Cao, Z. (2019, March). Environment dynamic monitoring and remote control of greenhouse with ESP8266 NodeMCU. In 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) (pp. 377-382). IEEE.

Williams, R. I. (1992). Handbook of SCADA systems: for the oil & gas industry.

Wood, L., Holliday, P., Floreani, D., & Psaras, I. (2009, October). Moving data in DTNs with HTTP and MIME. In 2009 International Conference on Ultra Modern Telecommunications & Workshops (pp. 1-4). IEEE.

Wu, Q., Chen, X., Yu, H., Liu, Q., & Yang, Y. (2020, March). Real-time data visualization method for oil pipeline monitoring based on Internet of Things. In IOP Conference Series: Materials Science and Engineering (Vol. 768, No. 5, p. 052124). IOP Publishing.

Yadav, G., & Paul, K. (2021). Architecture and security of SCADA systems: A review. International Journal of Critical Infrastructure Protection, 100433.

Yang, Y., McLaughlin, K., Sezer, S., Littler, T., Im, E. G., Pranggono, B., & Wang, H. F. (2014). Multiattribute SCADA-specific intrusion detection system for power networks. IEEE Transactions on Power Delivery, 29(3), 1092-1102.

ANNEXES

ANNEXES A

```
#include <Servo.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <Wire.h>
#include <BMP180.h>
#include <FlowMeter.h>

int pin_gas = A0;
int pin_Flame = A1;
int flow_pin = 2;
int temp_pin = 3;
int servo_pin = 9;

int gas_value;
int Flame_value;
int flow_value;
float Temperature_value;
float Pressure_value;

int flow_frequency;     // Measures flow sensor
pulsesint flow_minute;  // Calculated litres/hour
long flow_currentTime;
long flow_cloopTime;
int flow_value_;

BMP180 myBMP(BMP180_ULTRAHIGHRES);
DHT temp (temp_pin, DHT11);
Servo myservo;

void setup()
{
  Serial.begin(9600);
  Serial.setTimeout(100);
  myservo.attach(servo_pin);
  temp.begin();
  myBMP.begin();

  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(flow_pin, INPUT);
  digitalWrite(flow_pin, HIGH);
```

```
Serial.begin(9600);
attachInterrupt(0, flow_interrupt, RISING);
sei();
flow_currentTime = millis();
flow_cloopTime = flow_currentTime;
}
void loop()
{
gas_value = gas(); Flame_value = Flame();
flow_value = flow();
Temperature_value = Temperature();
Pressure_value = Pressure();
servo();
Serial.print(";esp1/g,"); Serial.print(gas_value);
Serial.println(";");
Serial.print(";esp1/ff,"); Serial.print(Flame_value);
Serial.println(";");
Serial.print(";esp1/t,");
Serial.print(Temperature_value);Serial.println(";");
Serial.print(";esp1/p,"); Serial.print(Pressure_value);
Serial.println(";");
Serial.print(";esp1/f,"); Serial.print(flow_value);
Serial.println(";");
Serial.println();
delay(1000);
}
void servo()
{
String rx =
Serial.readStringUntil(';');if (rx
== "on")
{
myservo.write(90);
digitalWrite(LED_BUILTIN, 1);
}
if (rx == "off")
{
myservo.write(0);
digitalWrite(LED_BUILTIN, 0);
}
```

ANNEX B

```
# include <PubSubClientTools.h>
#include <MqttWildcard.h>
#include <PubSubClient.h>
#include <StringSplitter.h>
#include <ESP8266WiFi.h>
const char *ssid = "Mohammed"; // Enter your WiFi name
const char *password = "10203040"; // Enter WiFi password
const char *mqtt_broker = "mqtt.iotindustries.sk";
const int mqtt_port = 1883;
const char *mqtt_user = "";
const char *mqtt_password = "";
WiFiClient espClient;
PubSubClient client(espClient);
void setup()
{
pinMode(LED_BUILTIN, OUTPUT);
digitalWrite(LED_BUILTIN, 1);
Serial.begin(9600);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED)
{
delay(1000);
digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
}
client.setServer(mqtt_broker, mqtt_port);
client.setCallback(callback);
while (!client.connected())
{
client.connect("esp8266-client", mqtt_user, mqtt_password);
digitalWrite(LED_BUILTIN, 0); delay(2000);
}
client.publish("esp1", "start"); client.subscribe("esp1/ms");
}
void callback(char *topic, byte *payload, unsigned int length)
{
}
```

```
{
for (int i = 0; i < length; i++)
{
Serial.print((char)payload[i]);
}
}
void loop()
{
client.loop();
String topic;
String messag;
String rx = Serial.readStringUntil(';');
if (rx != "")
{
StringSplitter *splitter = new StringSplitter(rx,
',', 2);
topic = splitter->getItemAtIndex(0);
messag = splitter->getItemAtIndex(1);
Serial.print(topic);
Serial.print(" : ");
Serial.println(messag);
client.publish(topic.c_str(), messag.c_str());
}
}
```

RESUME

Personal Information

Surname, name : Omar Mohammed Mahdi AL MUBARAK

Nationality : Iraq

Education

Degree	Education Unit	Graduation Date
Master	Electrical-electronic engineering	2022
Bachelor	Electronic and Control Engineering Techniques	2016/2017
High School	Al-MA'ARY School	2012/2013

Work Experience

Year	Place	Title
-	-	-

Foreign Language

Arabic - English

Publications