DE-42 (DC & SE)

NUST COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING

AQUA CONTROL - WATER BILLING & QUALITY ASSESSMENT SYSTEM

A PROJECT REPORT

DE-42 (DC & SE)

Submitted by

NS MUHAMMAD JAHANGIR

ASC MUHAMMAD AKIF HUSSAIN

NS ALI IQBAL

ASC TALHA BIN IRFAN

BACHELORS

IN

COMPUTER ENGINEERING

YEAR

2024

PROJECT SUPERVISOR

DR. SAJID GUL KHAWAJA

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,
ISLAMABAD, PAKISTAN

YEAR 2024

Certification

This is to certify that Muhammad Jahangir, 350955 and Ali Iqbal, 343273 and Muhammad Akif Hussain, 359498 and Talha Bin Irfan, 359497 have successfully completed the final project AQUA CONTROL - WATER BILLING & QUALITY ASSESSMENT SYSTEM, at the College of Electrical and Mechanical Engineering,(CEME) to fulfill the partial requirement of the degree 42.

Signature of Project Supervisor
Dr. Sajid Gul Khawaja
Associate Professor

Sustainable Development Goals (SDGs)

SDG No	Description of SDG	SDG No	Description of SDG
SDG 1	No Poverty	**SDG 9**	Industry, Innovation, and Infrastructure
SDG 2	Zero Hunger	SDG 10	Reduced Inequalities
SDG 3	Good Health and Well Being	SDG 11	Sustainable Cities and Communities
SDG 4	Quality Education	SDG 12	Responsible Consumption and Production
SDG 5	Gender Equality	SDG 13	Climate Change
SDG 6	Clean Water and Sanitation	SDG 14	Life Below Water
SDG 7	Affordable and Clean Energy	SDG 15	Life on Land
SDG 8	Decent Work and Economic Growth	SDG 16	Peace, Justice and Strong Institutions
		SDG 17	Partnerships for the Goals

Sustainable Development Goals

Complex Engineering Problem

Range of Complex Problem Solving

	Attribute	**Complex Problem**	
1	Range of conflicting requirements	Involve wide-ranging or conflicting technical, engineering and other issues.	X
2	Depth of analysis required	Have no obvious solution and require abstract thinking, originality in analysis to formulate suitable models.	
3	Depth of knowledge required	Requires research-based knowledge much of which is at, or informed by, the forefront of the professional discipline and which allows a fundamentals-based, first principles analytical approach.	X
4	Familiarity of issues	Involve infrequently encountered issues	X
5	Extent of applicable codes	Are outside problems encompassed by standards and codes of practice for professional engineering.	X
6	Extent of stakeholder involvement and level of conflicting requirements	Involve diverse groups of stakeholders with widely varying needs.	
7	Consequences	Have significant consequences in a range of contexts.	
8	Interdependence	Are high level problems including many component parts or sub-problems	X

Range of Complex Problem Activities

	Attribute	**Complex Activities**	
1	Range of resources	Involve the use of diverse resources (and for this purpose, resources include people, money, equipment, materials, information and technologies).	X
2	Level of interaction	Require resolution of significant problems arising from interactions between wide ranging and conflicting technical, engineering or other issues.	X
3	Innovation	Involve creative use of engineering principles and research-based knowledge in novel ways.	X
4	Consequences to society and the environment	Have significant consequences in a range of contexts, characterized by difficulty of prediction and mitigation.	
5	Familiarity	Can extend beyond previous experiences by applying principles-based approaches.	X

Dedicated to our advisor, Dr. Sajid Gul Khawaja, whose guidance was the main reason behind the successful completion of this humongous task. To our parents as they were the main reason behind our hard work. Thank you, everyone for making this a reality.

Acknowledgment

We would like to express our sincere gratitude and appreciation to all those who have contributed to the completion of this thesis.

First and foremost, we would like to thank ALLAH Almighty and acknowledge His blessings through the completion of this project. We are profoundly grateful for the wisdom and knowledge bestowed upon us by Allah. His divine guidance has been the ultimate source of inspiration and motivation, helping us overcome challenges and providing clarity during moments of uncertainty.

We would like to extend our deepest thanks to our supervisor Dr Sajid Gul Khawaja, whose guidance, support, and valuable insights have been instrumental throughout this industrial project. Their expertise and encouragement have helped shape the direction of this work and have provided us with invaluable guidance at every step. His valuable feedback at every step helped us polish our skills and critical thinking to achieve our project deliverables. It has been a great honor for us to conclude our project and thesis under his supervision.

We would like to express our heartfelt appreciation to our parents for their unwavering support and encouragement throughout this endeavor. Their love, understanding, and belief in our abilities have provided us with the motivation and strength to overcome challenges and persevere. We are truly grateful for their presence in our life.

In conclusion, this thesis would not have been possible without the collective efforts, guidance, and support of the individuals mentioned above. Their contributions have significantly shaped this research project, and we are sincerely grateful for their involvement.

Abstract

The project named "Water Billing & Quality Assessment" provides a computerized solution to improve the efficiency of water billing and quality assessment. Instead of using agents (employees) who visit homes to record water readings manually, this system uses sensor technology integrated with micro-controllers to gather and process the data remotely.

The project is centered on utilizing a STM 32 micro-controller, a series of 32-bit ARM Cortex-M based micro controllers , which uses a YFS-201 sensor for billing and pH and turbidity sensors for quality assessment. The proposed system is intended to receive and transmit data to mobile and Web applications.

The project objectives are multifaceted. Our primary objective is to execute the cpp code on the STM micro controller which transmits data from sensors. An essential aspect of achieving this objective is using the LoRaWAN technology to transmit data through LoRaWAN servers, which support a large number of devices per gateway, making it highly scalable for massive IoT deployments. Furthermore, we emphasize establishing a mobile and web application interface to display the data being transmitted through the entire structure.

The project encompasses both the development of hardware and software. A hardware system utilizing the STM micro controllers connected to the billing and quality sensors as discusses above are developed. The controllers are integrated with the LoRaWAN technology to transmit data through LoRaWAN servers to the mobile and web applications via cloud. The system's performance is evaluated based on value accuracy from the IoT devices, real-time processing capabilities, and the quality of its visual output.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

The Internet of Things(IoT) is a transformative technological concept which has been of significant attention in for many years because of it's wide range of applications in fields like transportation, Healthcare, Industrial, and smart home systems. The use of agents(employees) to record every piece of data diminishes the accuracy of data and results in time loss.

With the of IoT, this becomes comparatively easy as instead of relying on agents to read data physically (face to face) everything gets done remotely via the Iot devices which increases the efficiency and accuracy of data being recorded. With the fast progress of hardware technologies, there is an increasing demand for efficient and real-time data monitoring systems capable of dealing with dynamic and difficult conditions.

By using a STM32-micro controller for real-time data monitoring, this removes the use of any manual labour for data reading and collection. The IoT device monitors and records data to be sent towards database remotely instead of employees taking readings from water meters 1 house at a time.

The primary goal of this project is to create a smart transmission system that takes data from the IoT device in real-time, transmits the data through LoRaWAN servers and analyses it to produce a visual output display on the mobile and web applications.

Finally, the goal of this project is to create smart home IoT system for real-time data monitoring by transmitting it from the IoT device through the LoRaWAN servers to the web and mobile apps via cloud. The project's achievements have the potential to reduce the use of manual labour , significantly enhancing data monitoring and recording of data without the use of any physical agents (employees).

1.2 Problem Statement

This project addresses the problem of the use of old and obsolete data monitoring systems in Pakistan, especially in homes as agents(employees) are required to physically access the

meter to record the water readings of the homes. Such that not also be able to determine the water's quality if it is safe to use it or not.

This causes delay in readings as some times employee might not come to the customers house and take readings which causes an issue in water usage readings and billing statistics. Whilst also not determining if water is clean to be used or not.

The purpose of this project is to overcome these restrictions by creating a smart home IoT system based on real time STM32-micro controller readings. With its ARM Cortex-M CPU integrated with the LoRaWAN module, the STM platform provides a strong and versatile platform for fast data transmission. This project seeks to construct a system that can transmit data from he IoT device through the LoRaWAN servers, analyze the data and generate a visual output in real-time in mobile and web application respectively.

1.3 Objectives

The objectives we have to complete in order to make to the project fully operational are:

- To Create an IoT node for Billing using STM32 micro controller.
- To Create an IoT node for Quality using STM32 micro controller.
- Use the LoRaWAN module integrated on the micro controller to transfer data from IoT node.
- Establish connectivity of IoT nodes to the Things Network.
- Develop a transmission protocol to transfer data from the Things Network to Firebase Database.
- Develop a mobile app using flutter Dart frameworks for data visualization.
- Develop a web app using flutter Dart frameworks for remote monitoring.

This project seeks to contribute to the field of Internet of Thing's (IoT) and improve the development of efficient and high-performance smart home IoT systems by addressing these objectives and problems.

1.4 Scope

The goal of this project is to create a smart home IoT system for real-time water data monitoring. The system will be built to take data from micro-controllers in the form if pulses, convert them to real world value(Litres). To achieve hardware data processing and real-time speed, the project will take advantage of the STM32-micro controllers platform's capabilities, notably the combination of an ARM CPU and LoRaWAN module. By delivering efficient and high-quality data points, the suggested system attempts to remove any manual labour required for this system.
The project will focus on the following aspects mainly within its scope:

- ***Hardware-Software Integration:***
  The project will require the integration of software and hardware components. The STM32's platform's ARM CPU will handle control flow and data monitoring and conversion, while the LoRaWAN module will be used to transmit the data across the LoRaWAN servers .

- ***LoRaWAN Connectivity:***
  The major goal of the project is to create a transmission protocol that can transfer the data from device to cloud. This is done by registering the IoT device on the Things Network by inputting the device's EUI, App EUI and App Key, such that when registered the LoRaWAN module embedded on the micro-controller can access the LoRaWAN servers via the things network and send the data towards cloud via integration's .

- ***Cloud Database Storage:***
  The project's goal is to transmit data towards the Firebase cloud via the Things Network but this is done when the data transmitted from the things network is sent to the pipe-dream workflow via web-hooks integration. Here the data is received in a JSON body using a HTTP trigger, it is analyzed and then processed to extract the relevant data from the body using a Python custom script and that script connects the pipe-dream workflow to Firebase Database by providing the Firebase credentials and stores the relevant data in storage.

- ***Data Visualisation:***
  The system will be built to visualize the output data onto the mobile and web applications respectively such that the mobile application is for customers only and will show daily and monthly water usage as well as monthly bills.

  The Web application will be for admin and customer such that functionality for customer will remain the same but for admin the data will be shown as sum monthly water usages of all customers as well as their bills and the quality aspects of the water which are turbidity and pH.

1.5 Organization of Report

The final year project report is divided into 5 chapters. The remaining part of this document is organized as follows:

- Chapter 2 delves into a detailed study of the smart home IoT system. It also gives a review of existing literature on similar projects and techniques, laying the foundation for the project.

- Chapter 3 offers an overview of the system architecture and the proposed design of our project. It explain the working of our project and how it is implemented.

- Chapter 4 offers an exploration of the system architecture and the hardware components used for our Billing & Quality IoT node. It explain the working of the hardware structure and data transmission of the IoT nodes with the help of images discussing the components required, pin connections etc.

- Chapter 5 the focus is on the transmission protocol of the project in order how to transmit data from the device via LoRaWAN servers to the Firebase Cloud.
- Chapter 6 explains the data visualization of the project in which how data is retrieved from the Firebase Database and displayed for Customer in the mobile application & for Administration as well as Customer in Web Application.
- Chapter 7 gives a summary of the entire project, as how it contributes to the field of IoT using LoraWan technology rather than Wi-Fi.

Chapter 2

Background

This chapter covers background of the relevant components and their details which are required in the development of our solution.

2.1 System Implementation

2.1.1 Overview of STM Micro-controllers

2.1.1.1 Wide Range of Options:

The STM32 family offers a vast selection of micro controllers, ranging from low-power to high-performance variants. This allows designers to choose a model that precisely fits their needs, without having to compromise on performance or power consumption.

2.1.1.2 Performance:

STM32 micro controllers are based on the ARM Cortex-M series, which are known for their high performance and efficiency. They offer a variety of core options (Cortex-M0, M0+, M3,M4,M7,and M33),catering to different performance and power requirements.

2.1.1.3 Extensive Peripheral Set:

These micro controllers come with a rich set of peripherals, including multiple I2C, SPI, UART interfaces, ADCs, DACs, and advanced timers. This reduces the need for external components and simplifies system design.

2.1.1.4 Low Power Consumption:

Many STM32 models are designed with low power consumption in mind, featuring multiple low-power modes and flexible power management. This makes them suitable for battery-powered and energy-efficient applications.

2.1.1.5 Development Ecosystem:

STMicroelectronics provides a comprehensive anduser-friendly development ecosystem, including the STM32CubeMX for peripheral configuration and code generation, STM32-CubeIDE for development, and extensive libraries and middleware (e.g., STM32Cube HAL and LL APIs).

2.1.1.6 Community and Support:

STM32 micro controllers are widely spread and have many applications; therefore, there are a lot of tutorials, forum discussions, application notes, and other sources for using these micro controllers. This support network can help cut down the time needed for development and to minimize time spent on analyzing the problem.

2.1.1.7 Cost-Effectiveness:

However, considering all the features supported by STM32 micro controllers, these are relatively affordable devices. This makes them ideal for use in both personal and business related activities; be it for leisure or for profit making purposes.

2.1.1.8 Scalability and Compatibility:

STM micro controllers are designed in such away that all members within the same family can be used interchangeably with minimal differences in pin-outs hence facilitating easy up gradation from one version of another within the same family of STM32 micro controllers without much changes. This allows for necessary adjustments to be made in the future and for designs to be as adaptable, or fluid, as possible.

2.1.1.9 Advanced Features:

High - end STM32 models have some additional capabilities such as support of real-time operating system (RTOS), hardware cryptography, as well as connectivity interfaces such as USB, Ethernet, CAN and others which are critical for contemporary EMBS applications.

These all make the STM32 micro controllers as one of the most used micro controllers in today's applications ranging straight from basic embedded to high end applications. [1]

2.1.2 Sensors Configuration

In order to monitor & determine the usage and quality of the water, we are using different sensors for our system.

2.1.2.1 Water Billing Sensor

For this Purpose we are using a YFS-201 sensor as it is a water flow sensor that generates Pulses or Interrupts using the Hall effect. It has a hard plastic body and a small turbine inside that rotates when liquid flows through it with a flow rate range of 1 to 30L/ min. Its speed changes with different rates of flow.

The module has three pins: Power, Ground, and Digital output. YF-S201 consumes very little current and can work with an allowing pressure of 1.75 MPa.

It can measure upto 1to 30 L/min, The operational or working voltage is usually between 5V and 18V DC. Its working temperature range of -25 to 80. The sensor body is made from hard plastic so it also doesn't corrode.[2]

2.1.2.2 Water Quality Sensors

For this Purpose we are using a pH sensor and a turbidity sensor:

pH Sensor: In pH sensors measure solution pH by comparing the electrical potentials of a pH-sensitive system and a stable reference system. The sensing system features a pH-sensitive glass bulb that alters its voltage based on hydrogen ion concentration. This potential is measured by a sensing electrode, with a potassium chloride (KCl) solution conducting electricity between the glass and electrode.[3]

Turbidity Sensor: Turbidity tells us about the clearness of the water.It uses an LED beam light and a a light detector.It sends the light beam into the water,this light will be scattered from the suspended particles in water, Light detector placed at the 90 degree will detects the number of light reflected back from the particles.

The measuring unit of Turbidity is Nephelometric Turbidity Unit (NTU). Greater the NTU value means more suspended solids in water,lesser the NTU value means lesser the suspended solids in water.The sensors reads analogue value from (0-1023)which is then converted to corresponding voltage value and then from voltage to NTU.It also comes with the external module which is Turbidity detection module.

2.1.3 Transmission Protocol

For this we are using a TTN network for the transmission of data towards the cloud platform as discussed in Chapter 5.

- First by registering the device onto the network. [4]
- Then creating a webhook integration to pipedream. [5]

- Then Connecting the pipdream workflow to the Firebase Database using a python script. [6]

2.1.4 Data Visualization

For this, we are developing a mobile and web app for data visualization using Android Studio as an IDE, Flutter Framework [7] and Dart Language as discussed in 6.

Using Firebase Tools such as:

- **Authentication** for Login and Sign-up. [8]
- **Storage** for Image saving. [9]
- **Firestore** for information storing. [10]

2.2 Summary

Chapter 2 begins by providing an overview of how will we implement our IoT water system using the ideas of different techniques as discussed below:

- A LoRa-based Low-power Smart Water Metering System [11]
- Sensor System for Real-time Water Quality Monitoring [12]
- Water Quality Monitoring System: A Smart City Application With IoT Innovation [13]

Overall, Chapter 2 serves as a resource for understanding the different water monitoring systems whilst also discussing our own. It covers the basics of water billing & quality methods with an emphasis on the LoRaWan technology. This chapter equips readers with the necessary knowledge to explore and utilize different water meter solutions in the IoT world.

Chapter 3

System Schema and Architecture

We have designed a system-level diagram that depicts complete information about the entire project, as presented in the Figure 3.1 .

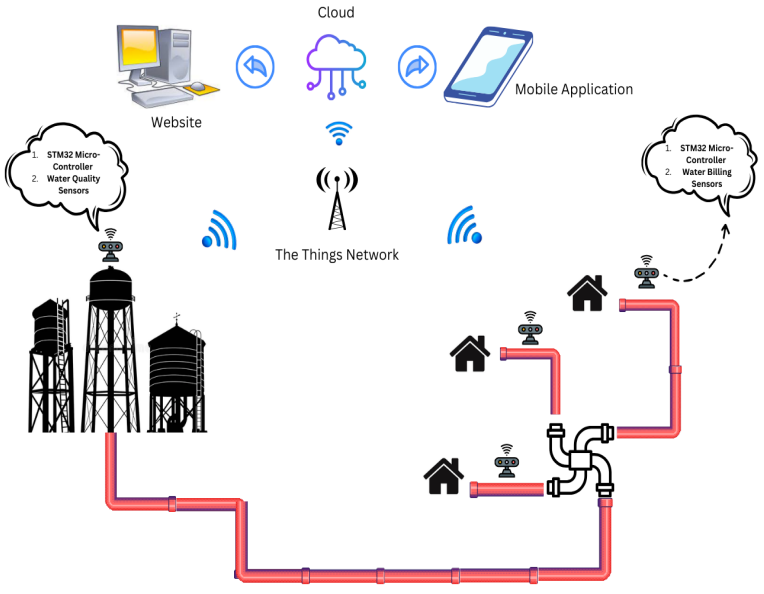


Figure 3.1: System Level Diagram

It is designed so that even a layman could easily understand the overview and integration of our whole project from IoT node to web and mobile application. The flow of the

complete system is depicted in Figure 3.1.

In the first step, we will be working on the acquisition layer for example In the first phase of this layer we started making our first IoT Node by integrating the water flow sensor to calculate the amount of water being consumed by any entity. The sensor used for this purpose is YF-S201. It generates the interrupts as the water flows through it. We have attached the sensor output pin to the STM32 Microcontroller that monitors and controls all the data coming from the sensor. In the next phase, we started working on the Second IoT node by fusing the water quality sensors such as pH and water turbidity sensors into our STM32 micro-controller.

Now once we get our desired data from the sensor to the micro-controller the next milestone is to transmit data without any physical connection to the server to process and transfer further to mobile and web apps. So for this purpose, we have used LoRaWan technology, the reason behind the choice of this technology was because its power consumption is very low and its long-range capability we started integrating loRawan libraries and necessary files to send the data acquired by our micro-controller to the Things Network which is part of our Network Layer. Using the technology of LoRawan our IoT node can now easily connect to the internet through the air by Network Console.

The Things Network console is a free platform for IoT devices where we can register our device free of cost and our device can communicate with the rest of the world all over the internet.

Data packets transmitted by the device manifest on The Things Stack board in JSON format. When the data from the IoT node reaches The Network console it is in the form of packets that contain all the necessary information such as Dev EUI, JoinEUI, AppkEY, etc that help identify which IoT node has transferred this packet. The received packet also contains a variable named payload that contains the value that is read and processed by our microcontroller from the attached sensors. The payload is in the form of an encoded base-64 string to forward to any other platform over the internet such as pipedream we

must decode the payload back to its original decimal value. For this purpose, we have provided a Payload formatter in JavaScript that converts the base64 back to a decimal value.

To transfer the data over the air to the Things console we are using the Dragino gateway that provides a path to connect the IoT devices to the internet remotely over the frequency of Asia-As-923.

To establish connectivity between The Things Stack and Pipedream, a web-hooks integration is utilized. This integration necessitates an endpoint URL and a Downlink API Key. The endpoint URL is generated from Pipedream, while the Downlink API Key is created in the API Key section.

First, the Things Network is connected to the pipedream workflow via a webhook connection & the workflow is connected to our Cloud Firestore via the custom Python script using the Firebase service credentials.

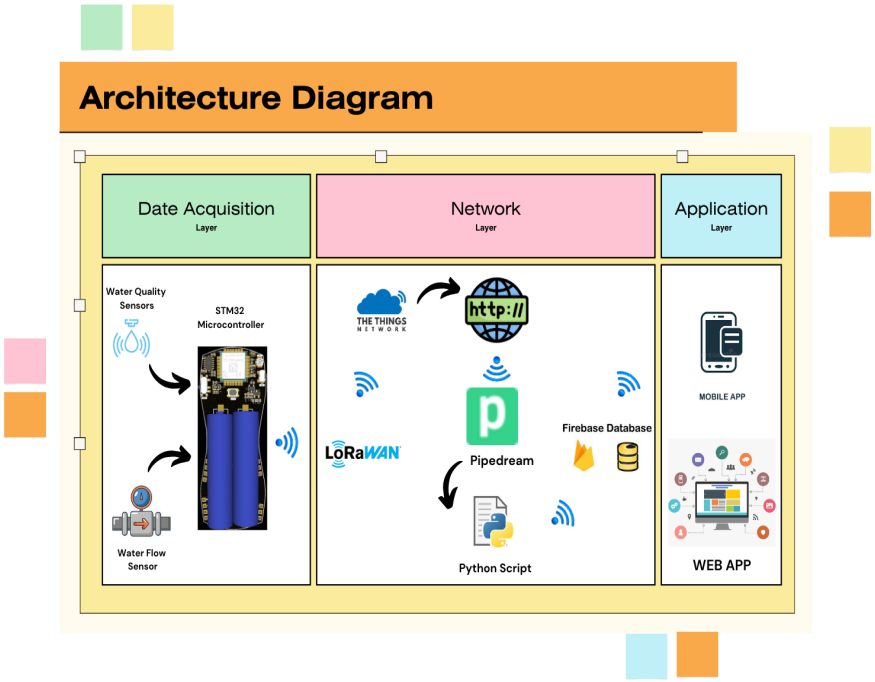


Figure 3.2: Architecture Diagram

The entire discussion above is visually represented by the Architecture level Diagram in

the Figure 3.2 shown above.

We have included a mobile application for visualizing the real-time assessment of water consumption by customers. This app allows customers to monitor their daily water usage and home billing. By tracking their consumption, customers can identify and curb excessive water use, helping to prevent waste and manage their bills more effectively. This approach promotes water conservation and encourages responsible usage.

We have also developed a web application for administrators and staff to monitor and control water quality. The user-friendly interface allows easy viewing of water quality metrics, enabling real-time actions if the water becomes unsuitable. Additionally, the admin can adjust water prices and billing rates through the front end. Furthermore, the system provides an option for viewing the number of customers connected to our network.

Chapter 4

Acquisition Layer

In this chapter, we will discuss the data acquistion layer which make use of our two IoT nodes, 1. Water Billing node and 2. Water Quality node. The chapter focuses on the integration of sensors with our Wise Node (STM micro-controller) custom-made by RISETech.

4.1 Hardware and Components

The following elements were chosen to keep in mind the cost as well as high efficiency to come up with this IoT node.

1. Microcontroller (STM32 Iot WiseNode)
2. YF-S201 Flow Sensor
3. PH Sensor
4. Turbidity Sensor
5. Potentiodivider Circuit
6. Connectivity Wires

7. Two 3D printed Boxes to fix Sensors and the WiseNode

8. One Acrylic Glass Box for the Water Quality Sensors

4.1.1 STM32 Iot WiseNode

We are using a Custom build (made by CEME) STM32WL55JC1 microcontroller4.1 that operates on low power(3.3V) and has a LoraWAN chip integrated into it. The schematic design is shown in Figure 4.1.

Figure 4.1: WiseNode

4.1.1.1 Software For IoT

STM32CubeIDE We are using STM32CubeIDE 4.2 for programming the microcontroller. STM32CubeIDE is a complete multi-OS development software, that was a part of the STM32Cube software company.STM32CubeIDE is an advanced C and C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors

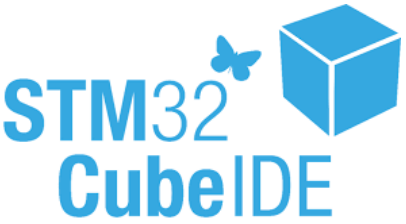


Figure 4.2: STM32CubeIDE

STM32CubeProg STM32CubeProgrammer 4.3 (STM32CubeProg) is a complete multi-OS software tool for programming STM32-related products. It provides an easy-to-use and efficient environment for reading, writing, and verifying device memory through both the debug interface (JTAG and SWD) and the bootloader interface (UART, USB DFU, I2C, SPI, and CAN).

Figure 4.3: STM32CubeProgrammer

4.1.2 YF-S201 Flow Sensor

The YF-S201 shown in Figure 4.4 is a water flow sensor that generates Pulses or Interrupts using the Hall effect. It has a hard plastic body and a small turbine inside that rotates when liquid flows through it with a flow rate range of 1 to 30L/min. Its speed changes with different rates of flow.

Figure 4.4: YF-S201

Figure 4.5 shows the schematic design of YF-201 sensor attached with the WISENode.

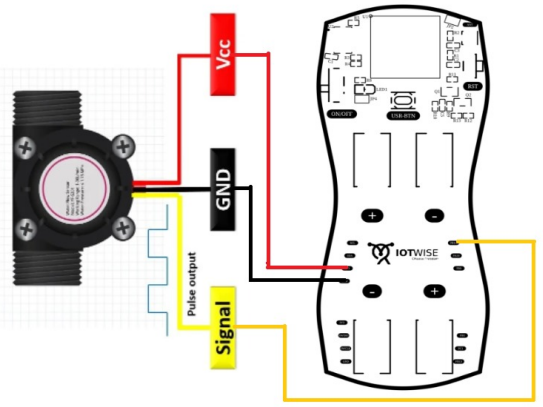


Figure 4.5: Schematic diagram of YF-S201

4.1.3 PH Sensor

The pH sensors4.6 measure solution pH by comparing the electrical potentials of a pH-sensitive system and a stable reference system. The sensing system features a pH-sensitive glass bulb that alters its voltage based on hydrogen ion concentration. It gives a reading between 0-14.

Figure 4.6: PH Sensor

The given Figure 4.7 is the schematic diagram of PH Sensor.

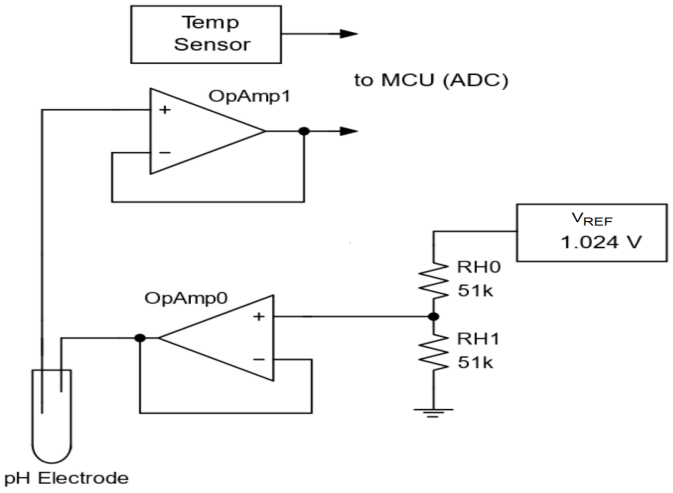


Figure 4.7: Schematic Diagram

4.1.4 Turbidity Sensor

Turbidity4.8 tells us about the clearness of the water. It uses an LED beam light and a light detector. It sends the light beam into the water, this light will be scattered from the suspended particles in the water, a Light detector placed at 90 degrees will detect the number of light reflected back from the particles.

Figure 4.8: Turbidity Sensor

As shown in the Figure 4.9.

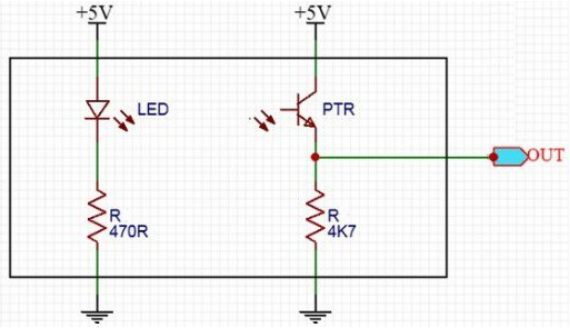


Figure 4.9: Schematic Diagram

4.2 Billing Node-Sensor Integration

The first step in integrating the sensor into the microcontroller was finding the right pin to attach the sensor output by viewing the datasheet of our microcontroller. Since our sensor generates pulses we need to set a pin as GPIO EXTI shown in Figure 4.10.

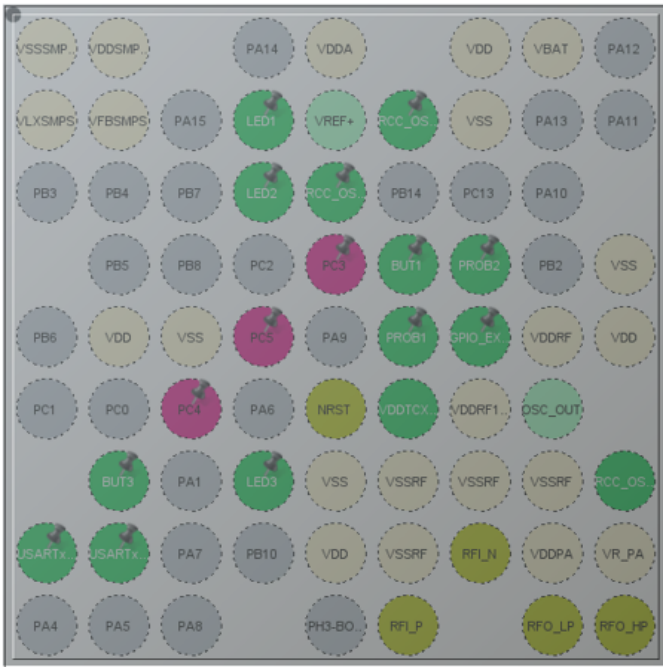


Figure 4.10: Wise Node Pin configuration

Ater setting up the pin as GPIO EXTI,then enabling the GPIO EXT14.11 at Falling edge trigger detection because YF-S201 generates interrupt and when interrupts occurs it generate a pulse so it will count the number of pulses by counting the number of falling

edges.

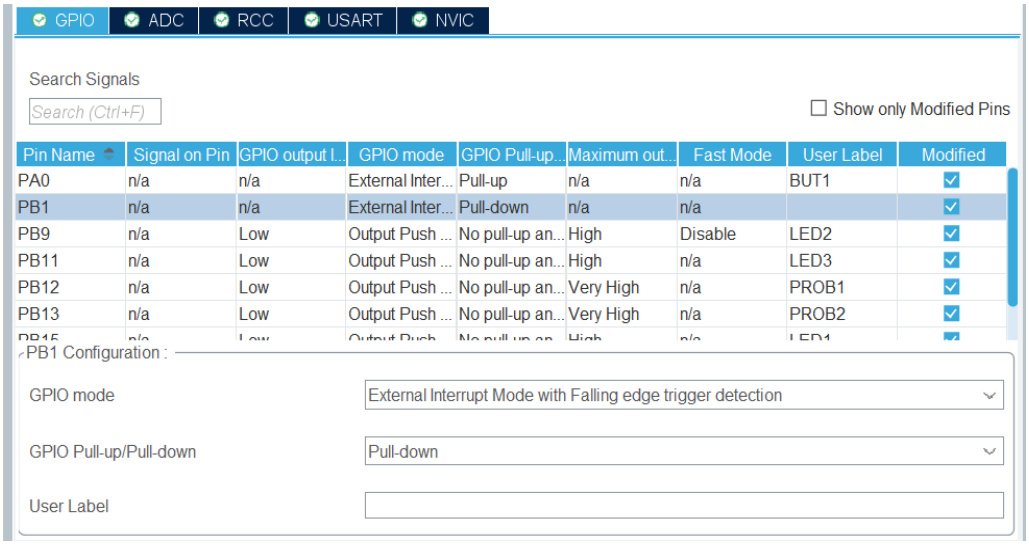


Figure 4.11: GPIO Configuration

Later we provided the Ground and VCC = 3.3v and ground to the sensor got the sensor reading to our microcontroller and stored it in a variable.

4.3 Quality Node Sensor Integration

4.3.1 Calibration of PH Sensor

Firstly I calibrated the PH sensor using the Arduino by Connecting the inner side of the BNC connector with the Outer body, then reading the Analogue value from the Circuitry, it gave 3.3V (3.3V due to the potential divider) because of the potential divider adjusting the potentiometer integrated on thye circuitry I have to drop the voltage to half (1.65V) to adjust its 0 Point.

4.3.2 Hardware Connections(sensors with wiseNode)

Coming on to the integration part of the sensors we know that both sensors give Analogue Output and both sensors require 5V but the wiseNode which we are using operates on

3.3V so we have to use the Potentio divider as shown in Figure 4.12 to drop the voltage from 5V to 3.3V.To convert this we are Two resistors of 2.2k and 1k Then connecting VCC and GND to both sensors.

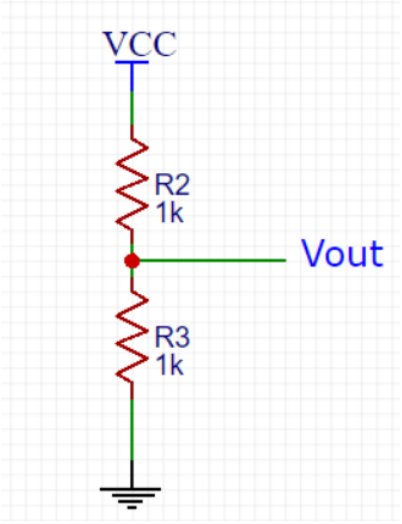


Figure 4.12: Potentiodivider Circuit

Since our sensors give analog Signal so looking at the Documentation of the wiseNode, we have connected the PH sensor with the PB14 pin of the wiseNode and the Turbidity sensor with the PA10 pin of the wiseNode, keeping in mind that these both have ADC channels. As shown in the Figure 4.13 Below

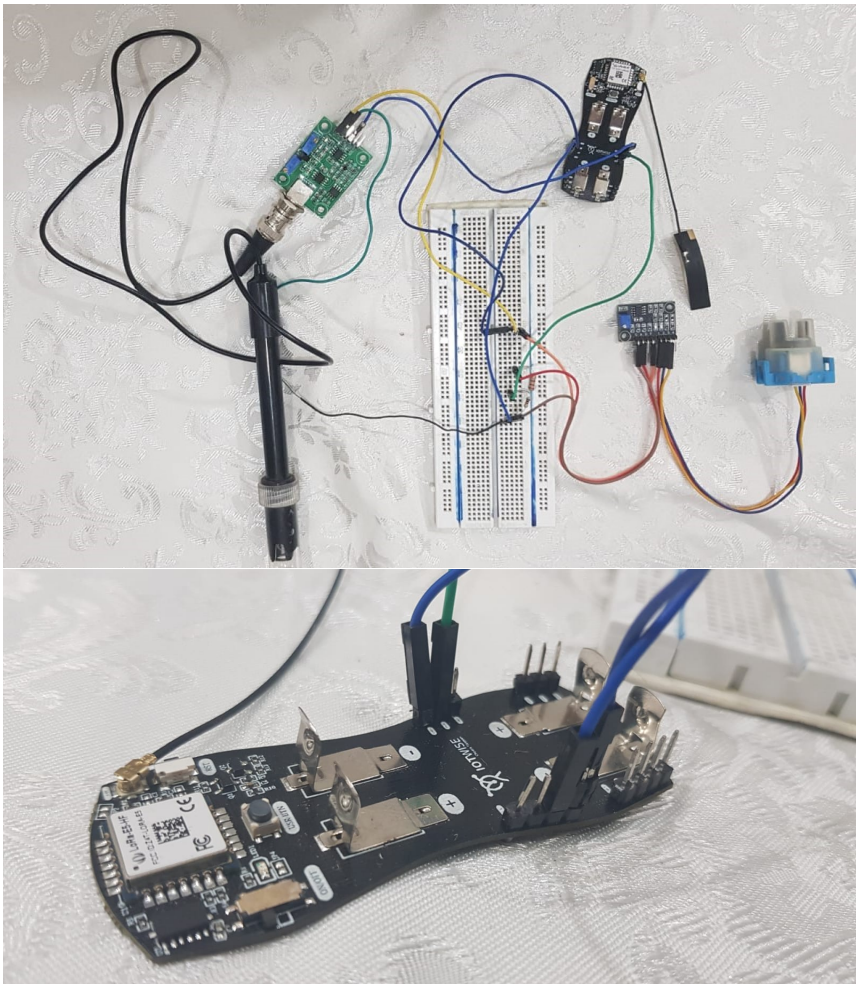


Figure 4.13: Hardware Connections

4.3.3 STMCubeide

Firstly creating the new Project and modifying the IOC file 4.14 according to our WiseNode.

Figure 4.14: IOC file

To enable the ADC channels 4.15, the PA10 pin should be set as `ADC_IN6`, and the PB14 pin should be set as `ADC_IN1` in the IOC file.

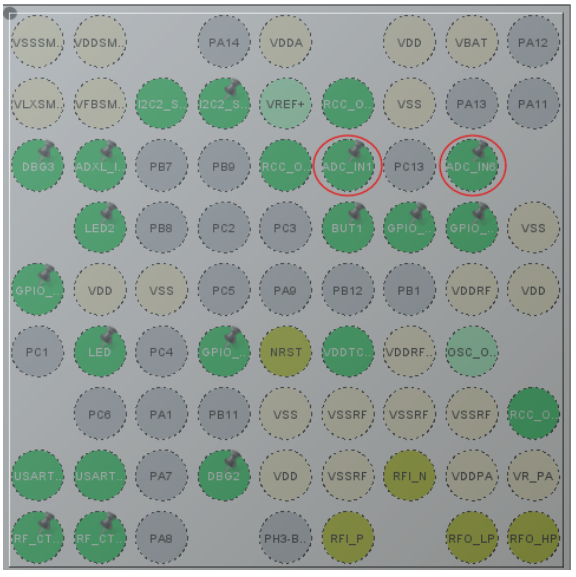


Figure 4.15: Enabling ADC pins

Then set the ADC configuration 4.16 by enabling the Scan Conversion Mode and Continuous Conversion Mode because we want to read the sensor data from pins we have enabled as ADC and want to continuously convert the data after regular intervals of time

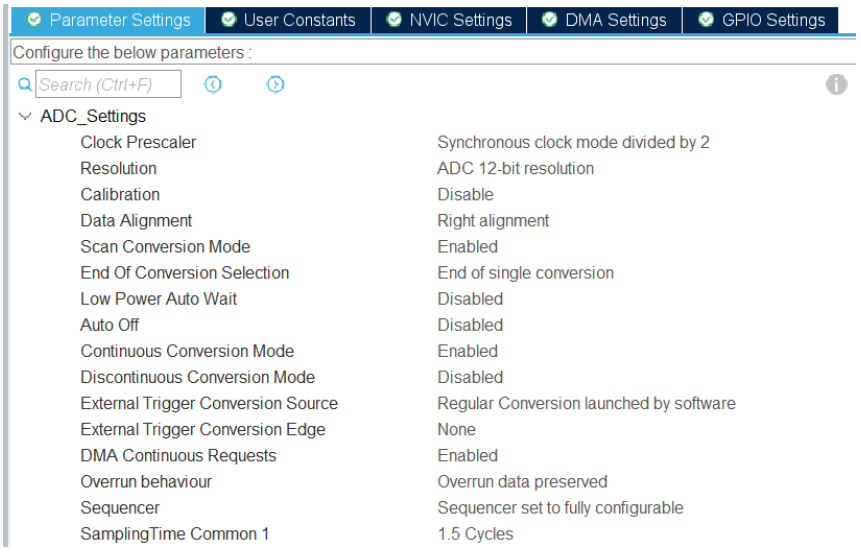


Figure 4.16: Setting up Configuration for ADC

4.3.3.1 Scan Conversion Mode

Scan Conversion mode allows multiple analogue signals to be converted into digital values allowing ADC to handle this automatically and allow sequential conversion of each channel. It also plays an important role by improving the efficiency of data acquisition from sensors making it a suitable way for the system involving multiple sensors.

4.3.3.2 Continuous Conversion Mode

Continuous Conversion mode allows continuous conversion of data from sensors. It is very useful when we need to see real-time monitoring of data where the signal needs to be constantly observed, It also reduces the need for the CPU to repeatedly start the conversion.

Combining these 2 modes will allow continuous and multiple sensor data conversion with higher efficiency.

We are using the DMA method because our microcontroller has only 1 ADC in it that's why for multiple ADC conversion we have to use (Direct Memory Access) DMA.

4.3.3.3 Direct Memory Access (DMA)

DMA, which stands for Direct Memory Access, is a feature found in many microcontrollers, including the STM32 series. This feature enables efficient data transfer between peripherals and memory without involving the CPU. The DMA controller functions as an intermediary between peripherals and memory, facilitating direct data transfers. By taking over the task of moving data between peripherals and memory, the DMA controller frees up the CPU to perform other operations. Operating independently, the DMA controller can transfer data in various modes, such as single, circular, 4.17(in our case we are using circular)

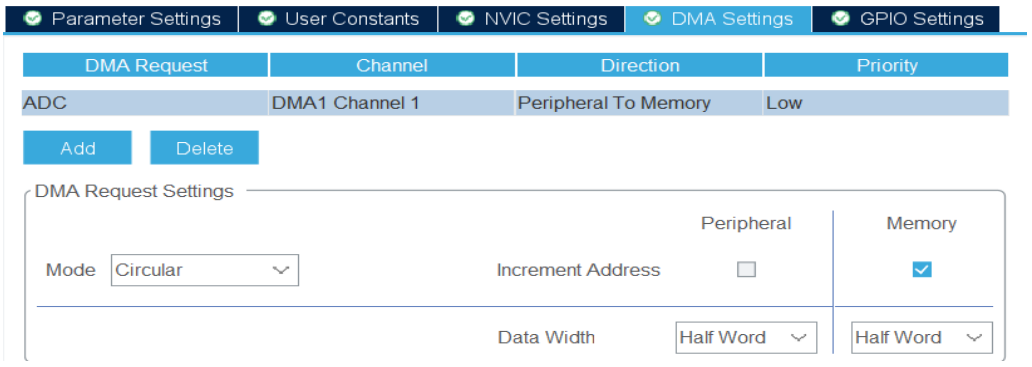


Figure 4.17: Adding DMA

or burst mode. It can handle data transfers to and from peripherals, memory, or even between different memory locations.

Utilizing DMA4.18 can significantly enhance the performance of a microcontroller, especially in scenarios requiring frequent and high-speed data transfers. This includes tasks like audio processing from an ADC, data logging, or communication protocols such as UART, SPI, or I2C.

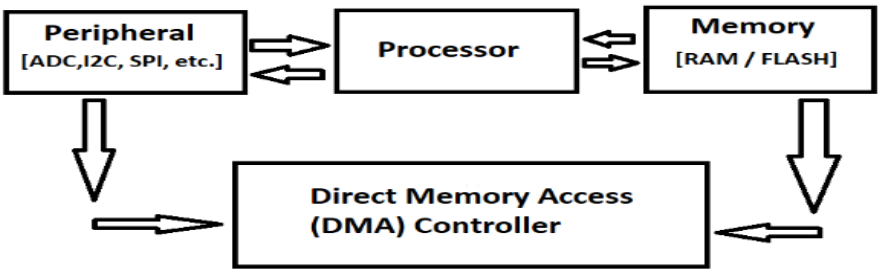


Figure 4.18: DMA

After setting the the environment we have generated the code and have done the required changes in the code to get the values.

4.4 Final Setup of Hardware

4.4.1 Billing Node

This includes the final Hardware setup after fixing of sensors and microcontroller and the node are ready to be deployed.4.19

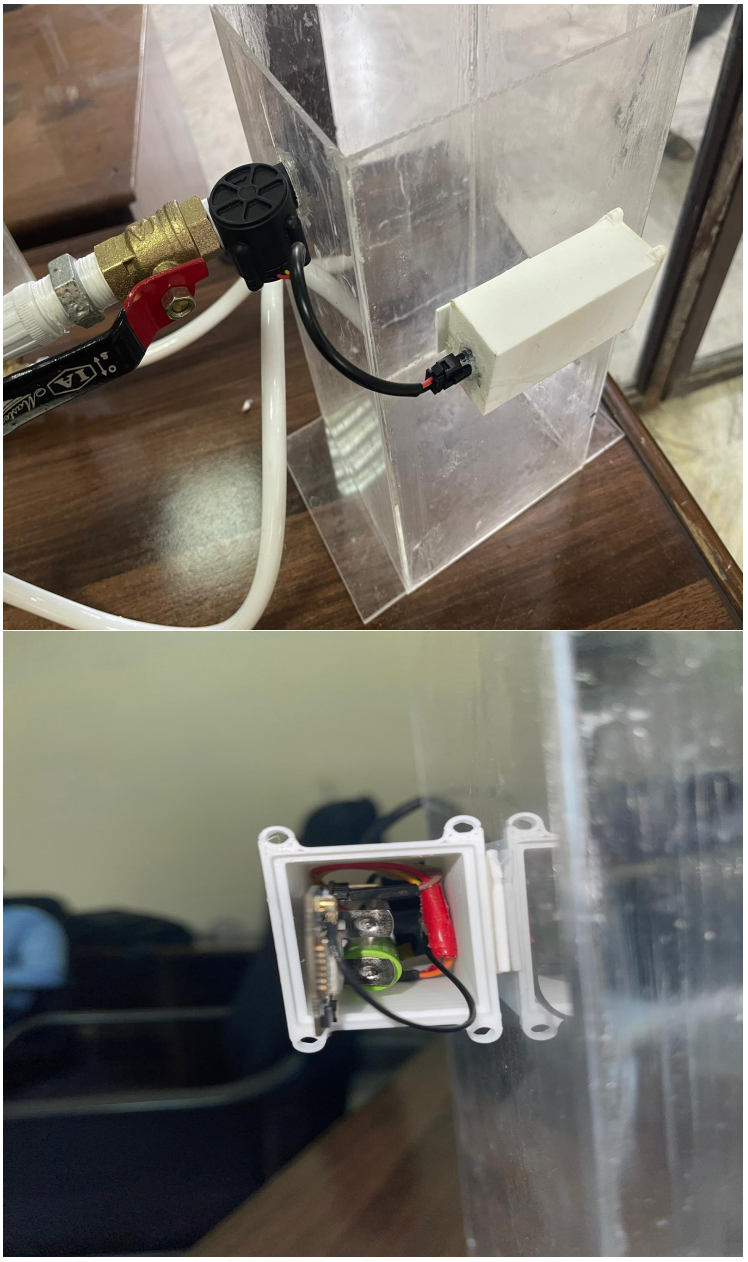


Figure 4.19: Final Setup for Billing node

4.4.2 Quality Node

The Figure 4.20shows the the complete Hardware setup for quality node.

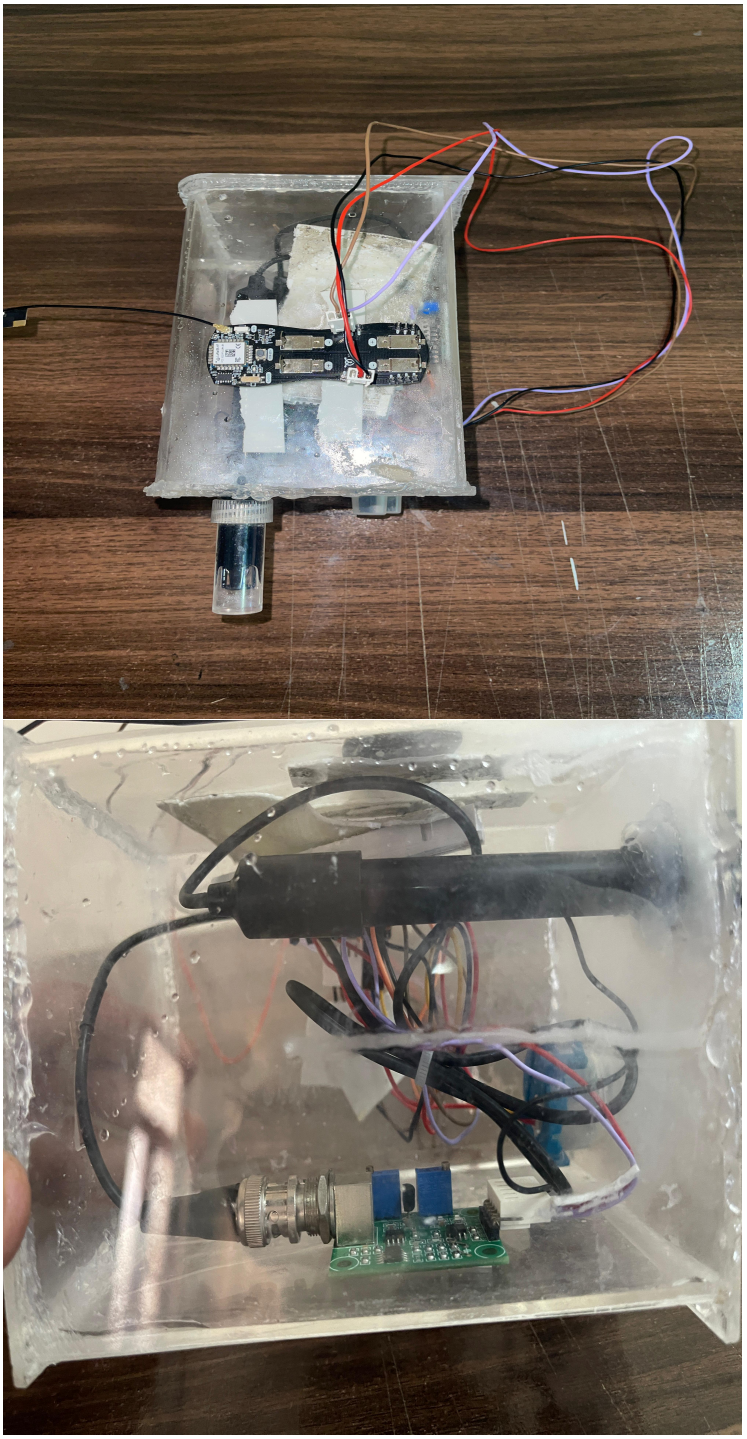


Figure 4.20: Final Setup for Quality node

4.5 Power Consumption

4.5.1 Billing Node

We are using power from battery sources such as two AAA battery cells4.21 as shown in the figure below.

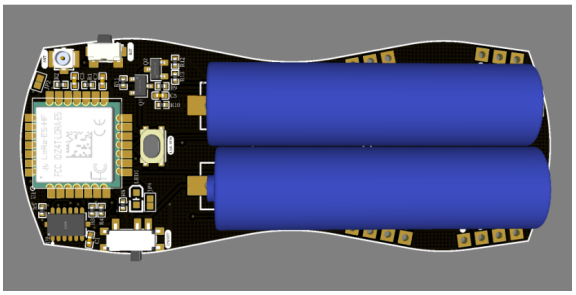


Figure 4.21: MCU With Batteries

These two AAA battery cells combine to give us around 1500 mAh of power. The Batteries we are using are from Xiaomi Mi brand Z17 alkaline dry batteries and they have a shelf power for up to 7 years.

To calculate the power consumed by our sensor and microcontroller we have to assume some assumptions or consider some scenarios as discussed below:

4.5.1.1 First Scenario: Low Sensor Usage

In this scenario, we assume that the sensor is active for only one percent of the time, with the Micro Control Unit (MCU) in sleep mode for the remaining ninety-nine percent. The MCU has a sleep current of 2.1 microamperes (WOR mode), while the YF-S201 sensor operates at its maximum current of 15mA.

The formula to calculate the average current consumption can be given as:

$$I_{\text{avg}} = (I_{\text{sleep}} \times T_{\text{sleep}}) + (I_{\text{active}} \times T_{\text{active}})$$

Calculating the average current for the Billing Sensor.

$$I_{\text{avg}} = ((2.1\mu A \times 0.99)) + (15.0021mA \times 0.01)$$

$$I_{\text{avg}} = 0.1521mA$$

Now we need to calculate the battery life, and the formula for the battery life can be given as:

$$\text{Battery Life} = \frac{\text{Battery Capacity}}{\text{Average Current}}$$

As we know our batteries give us an accumulative capacity of 1500 mAh and the average current is 0.1521 mA. Hence we have a battery life equal to:

$$\text{Battery Life } = \frac{1500 \text{ mAh}}{0.1521 \text{ mA}}$$

Given a battery capacity of 1500mAh and an average current of 0.1521mA, the battery life is approximately 9,860 hours or 411 days.

4.5.1.2 Second Scenario: Sensor Operating Continuously

In this scenario, we assume that the sensor is operating continuously or for ninety-nine percent of the time and that our Micro Control Unit is in sleep mode for only one percent of the time. The Micro Control unit takes as low as 2.1 Microamperes sleep current (WOR mode) and assuming YF-S201 is operating at its 100% and draws its maximum current i.e. 15mA.

Since the sensor is operating continuously at 15mA and that of MCU sleep time is very negligible at about 0.21A, Hence the average current will be:

$$I_{\text{avg}} \approx 15mA$$

Now we need to calculate the battery life, and the formula for the battery life can be given as:

$$\text{Battery Life} = \frac{\text{Battery Capacity}}{\text{Average Current}}$$

As we know our batteries give us an accumulative capacity of 1500 mAh and the average current is 15mA. Hence we have a battery life equal to:

$$\text{Battery Life} = \frac{1500 \text{ mAh}}{15 \text{ mA}}$$

$$\text{Battery Life} \approx 100 \text{ hours or } 4.17 \text{ days}$$

4.5.1.3 Third Scenario: Normal Water Consumptions

In this scenario, we are assuming that the efficiency of the sensor is only 25 % which means it will now drain a max current of 3.75 mA instead of 15mA as of in previous when it was operating at 100 % efficiency. Moreover, we are also assuming that the MCU will go to sleep every 22 hours out of 24 hours since usually it takes 1 or 2 hours to fill up any household water tanker.

We are also considering MCU to be active during this time and its current draw is assumed to be about 15 μ A. Hence the current during the two hours when the MCU and Sensor both are in active mode will be 3.765 mA. and the time percentage will be $\frac{2}{24}$. We have also assumed that MCU will be in sleep mode for 22 hours so the time percentage will be $\frac{22}{24}$. The formula to calculate the average current consumption can be given as:

$$I_{\text{avg}} = (I_{\text{sleep}} \times T_{\text{sleep}}) + (I_{\text{active}} \times T_{\text{active}})$$

$$I_{\text{avg}} = (3.765mA \times \frac{2}{24}) + (2.1\mu A \times \frac{22}{24})$$

$$I_{\text{avg}} \approx 0.3139mA$$

Now we need to calculate the battery life, and the formula for the battery life can be given as:

$$\text{Battery Life} = \frac{\text{Battery Capacity}}{\text{Average Current}}$$

As we know our batteries give us an accumulative capacity of 1500 mAh and the average current is 0.3139 mA. Hence we have a battery life equal to:

$$\text{Battery Life } = \frac{1500mAh}{0.3139mA}$$

Battery Life $\approx$ 4777.3 hours or 199.05days

4.5.2 Quality Node

As quality node includes 2 sensors:

1. Maximum current of PH Sensor is 20mA
2. Maximum current of Turbidity Sensor is 40mA

As the Quality node sensors will be in use continuosly so it will operates 99 percent of the time so the sleep time of the MCU is 1 percent that is negligible so the average current will be 0.6mA(0.4mA+0.2mA).As the battery is 1500mAH.We also include 30 percent line losses so the battery capacity will remain 1050mAH.

Calculating the Battery capacity using the formula

$$\text{Battery Life} = \frac{\text{Battery Capacity}}{\text{Average Current}}$$

The battery will last for 2500Hr or 41 days and if we include line losses then it will last for 1750Hr or 29 days.

4.6 Summary

In this chapter, we have discussed about the two IoT nodes including a brief discussion on all the sensors and the WiseNode (microcontroller) used in our two nodes. We have also mentioned the software we are using to program the WiseNode.We have also added detailed steps for the Integration of sensors including all the steps in hardware and software to be done to get the values from the sensors.Last but not least we have also included the battery consumption for the 2 nodes.

Chapter 5

Network Layer

This chapter delves into the intricacies of data transmission within the smart home IoT system. Its aim is to provide a comprehensive overview of how data transmission functions, including the analysis and processing of data prior to reaching the database.

5.1 Data Transmission Protocol

The connectivity flow within the protocol is depicted in Figure 5.1.

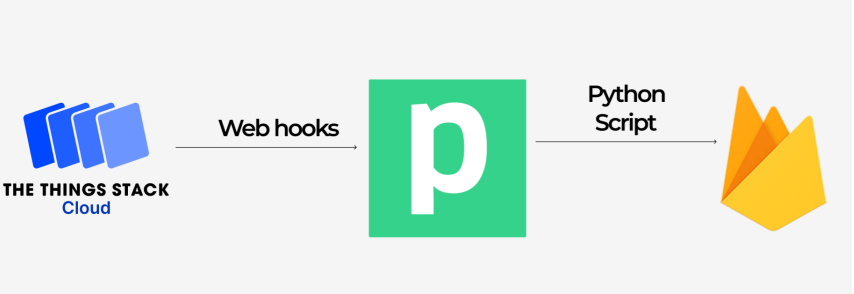


Figure 5.1: Data Transmission Protocol

5.1.1 The Things Stack/Network

5.1.1.1 Introduction

The Things Stack/Network serves as a LoRaWAN Network server facilitating the connectivity, management, and monitoring of devices, gateways, and end-user applications. Its primary objective is to ensure the security, scalability, and reliability of data routing throughout the network.

5.1.1.2 Setting up The Things Stack

Upon logging into The Things Stack, the initial step involves creating an application. Subsequently, within the "End Devices" section, devices are registered by inputting the Device EUI, App EUI, and App Key. Once registered, devices can utilize the LoRaWAN servers via the embedded LoRaWAN module.

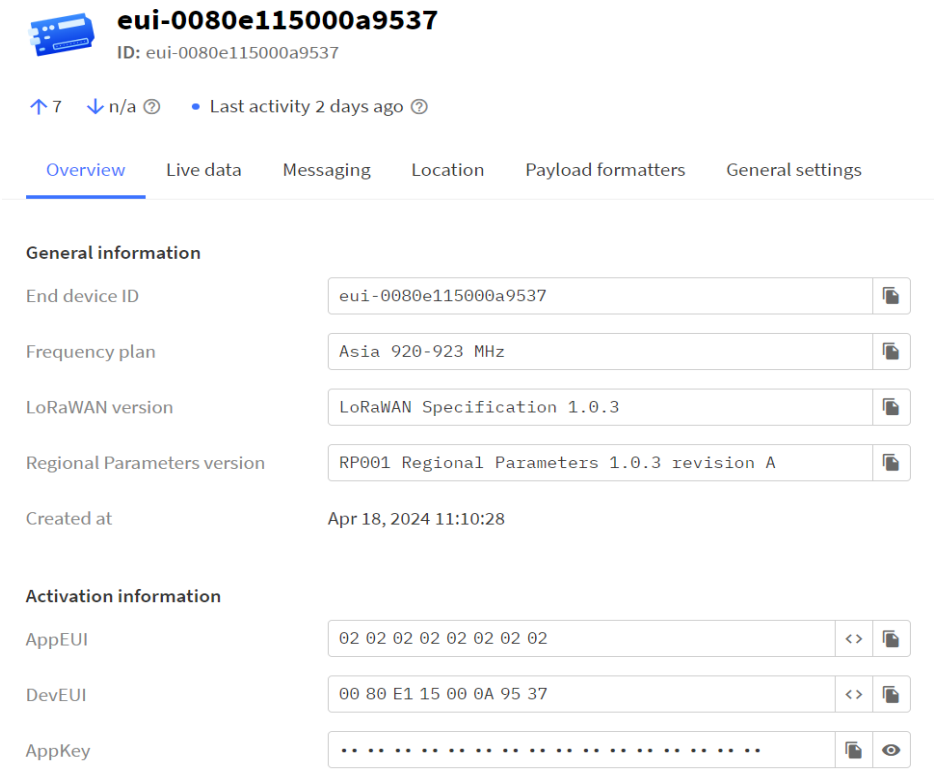


Figure 5.2: End Device Registration

5.1.1.3 Data Packet Reception

Data packets transmitted by the device manifest on The Things Stack board in JSON format. These packets contain the payload, which comprises the actual data sent from the device. As the received data is in hexadecimal form, a payload formatter is employed to convert it into decimal form. Subsequently, it is divided by 450, considering that there are 450 pulses per liter.

Figure 5.3: Payload Transmission

5.1.1.4 Integration to Pipedream

To establish connectivity between The Things Stack and Pipedream, a web-hooks integration is utilized. This integration necessitates an endpoint URL and a Downlink API Key. The endpoint URL is generated from Pipedream, while the Downlink API Key is created in the API Key section.

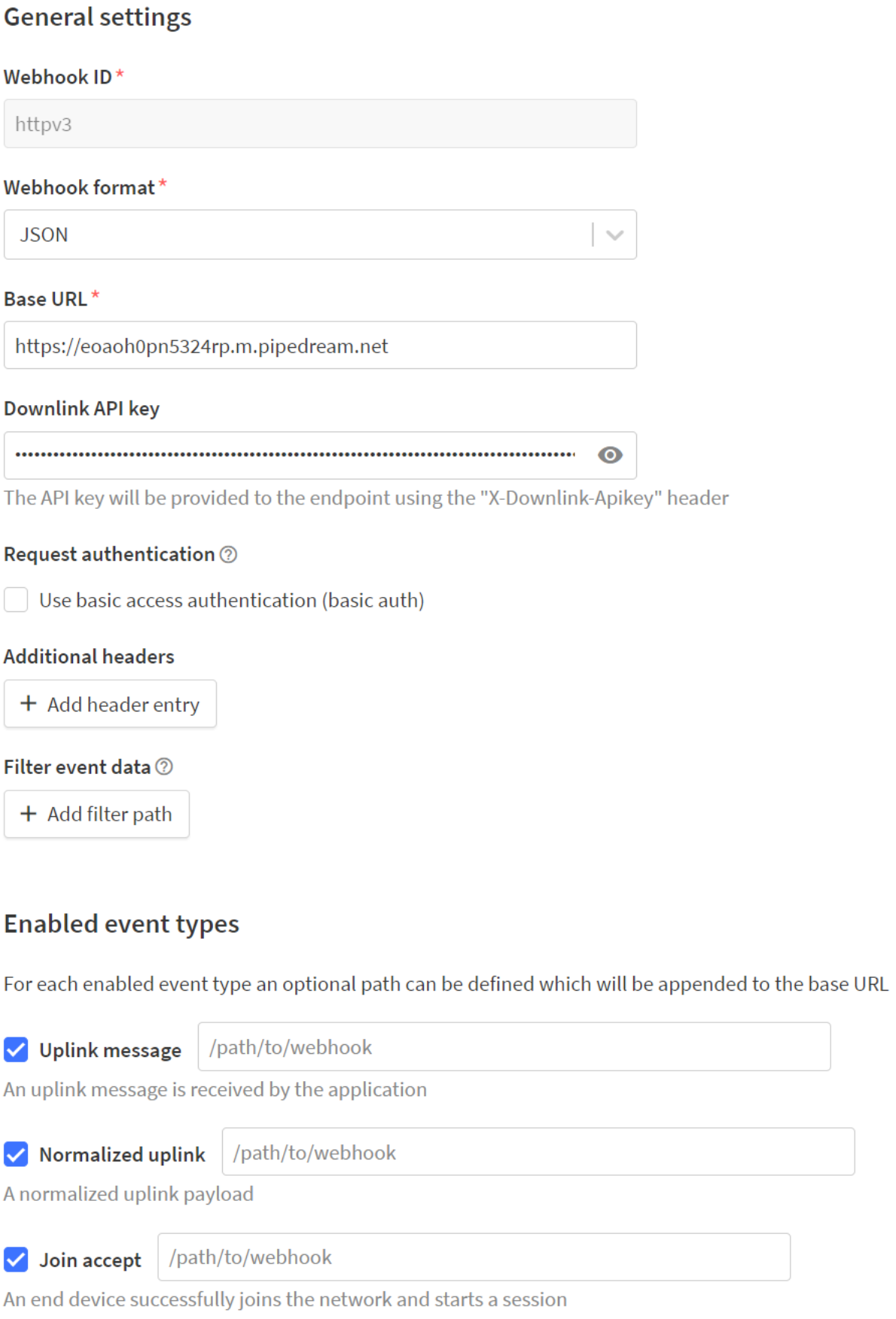


Figure 5.4: Web-hook Integration

5.1.2 Pipedream Workflow

5.1.2.1 Introduction

Pipedream serves as a cloud-based integration and automation platform, enabling users to connect various APIs, services, databases, and applications by creating custom workflows.

5.1.2.2 Workflow Creation

The initial step involves creating a project followed by a workflow within it. The workflow offers various HTTP options, among which "full HTTP data request" is selected.

5.1.2.3 Things Stack Connectivity to Pipedream

To connect The Things Stack to Pipedream, an endpoint URL is required for the webhooks integration. This URL is generated within the HTTP request in the workflow body.

5.1.2.4 Connectivity to Firebase

To link the workflow to the Firebase database, a Python custom script is utilized for data transfer from Pipedream to the database. The Firebase admin library is employed, requiring the service credentials of the Firebase account to establish connectivity.

```python
import firebase_admin
from firebase_admin import credentials, firestore
from datetime import datetime, timedelta
import pytz

try:
    # Try to get the app instance
    app = firebase_admin.get_app()
except ValueError:
    # Initialize Firebase with credentials and a unique app name
    cred = credentials.Certificate({
        "type": "service_account",
        "project_id": "digital-water-billing-system",
        "private_key_id": "093d630fd92e1ac614ebbe1682ac9ed8b540a87f",
        "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFA
        "client_email": "firebase-adminsdk-orcro@digital-water-billing-system.iam.
        "client_id": "104291328075654925633",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs
        "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509
        "universe_domain": "googleapis.com"
    })
    app = firebase_admin.initialize_app(cred, name='digital-water-billing-system')

# Access Firestore database using the initialized app instance
db = firestore.client(app=app)
```

Figure 5.5: Firebase Connection

Upon receiving data from The Things Stack, Pipedream processes it to determine daily and monthly usage. These parameters are crucial for both customers and administration purposes. Quality parameters such as turbidity and pH are displayed solely for administration.

5.1.3 Firebase

5.1.3.1 Firebase Connectivity

The connection to Firebase is established by providing service credentials, generated when a private key is created.

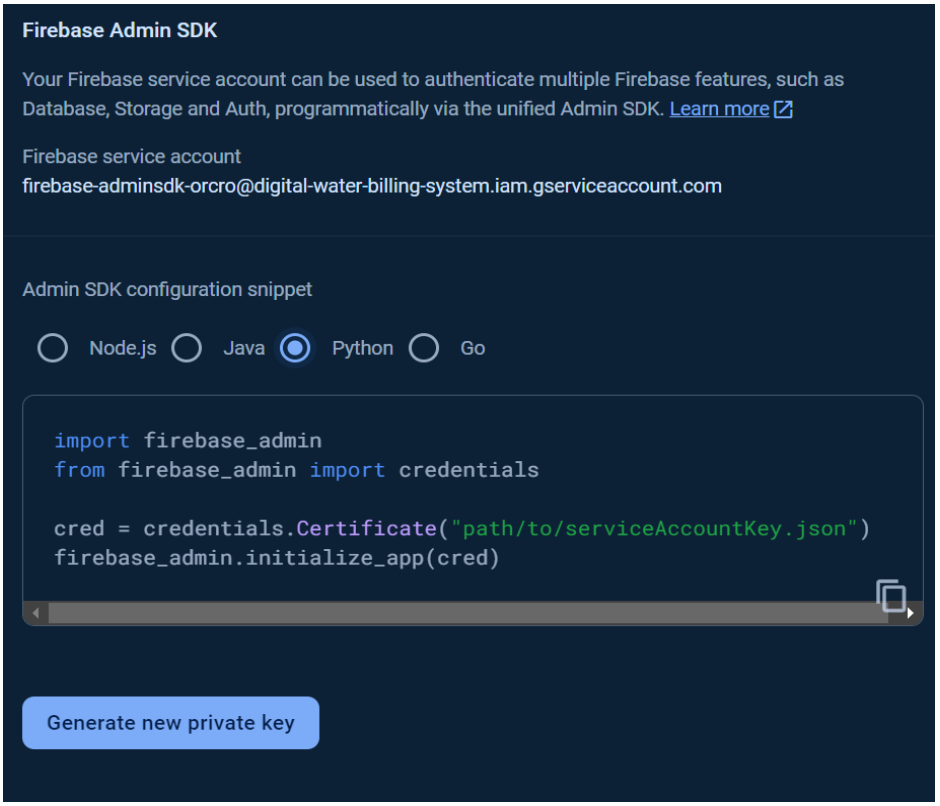


Figure 5.6: Service Credentials

5.1.3.2 Customer Database

Before transmitting and storing data in the database, a connection is established. To determine which IoT device is supposed to send data to which user, a verification process is used. In this process, the IoT device EUI (Extended Unique Identifier) is cross-referenced with the device EUI stored in the user credentials. If a match is found, the data is stored under that user's ID.

To calculate daily water usage, a Date Time function (in Pakistan Standard Time) is used. The live data for the day is accumulated until midnight, after which data for the next day is stored separately.

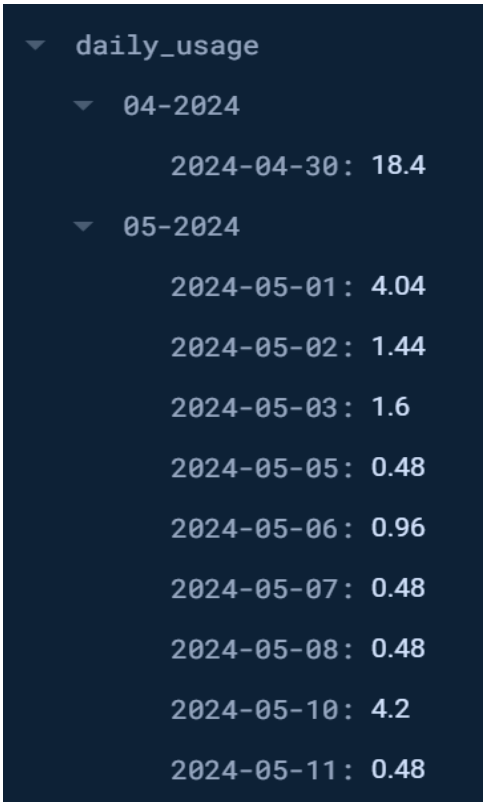


Figure 5.7: Customer Daily Usage

The same logic applies to monthly usage calculations. Daily data is continually added up as new data arrives. When the new month starts, the total usage for that month starts, and the process starts anew to ensure accuracy.

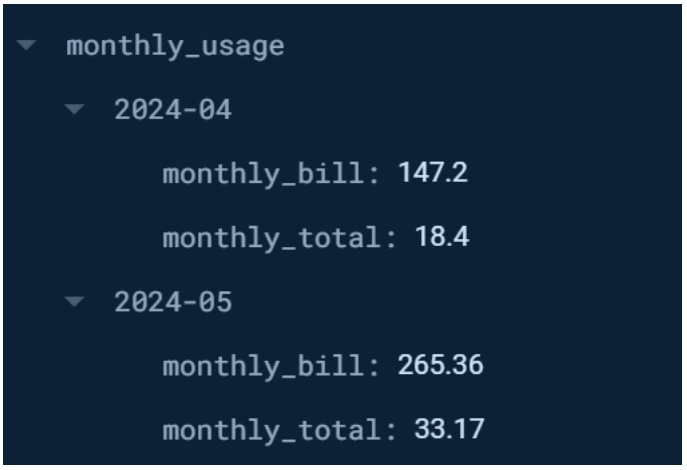


Figure 5.8: Customer Monthly Usage & Bill

To determine the monthly bill, the administration credentials include a field known as

"billing Rate," which specifies the cost of water per litre. The monthly total water usage is multiplied by the billing Rate to calculate the monthly bill.

The billing Rate field is shown as below:

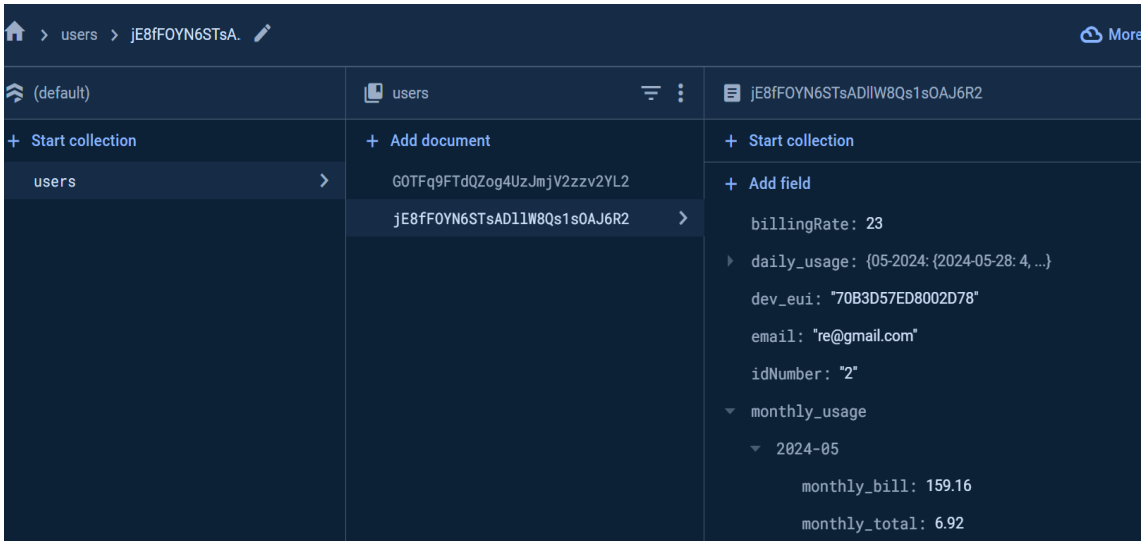


Figure 5.9: Billing Rate Field

With this the monthly usage which is stored, appears as depicted in Figure 5.9 :

5.1.3.3 Administration Database

The quality parameters, pH and turbidity, are stored in the administration credentials and are updated every time a new packet arrives. This ensures the values remain accurate and current.

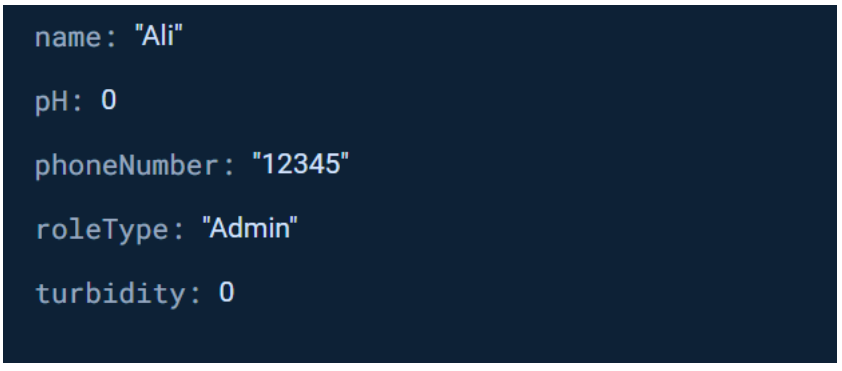


Figure 5.10: Water Quality Parameters

To calculate daily water usage, a Date Time function (in Pakistan Standard Time) is used. The daily usage data for each customer is accumulated and stored in the administration credentials for that day.

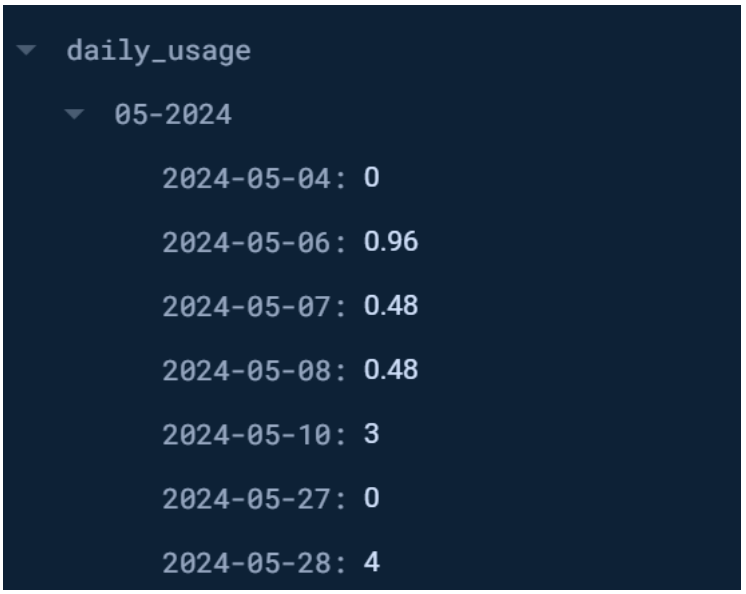


Figure 5.11: Administration Daily Usage

To calculate monthly water usage, a Date Time function (in Pakistan Standard Time) is used. The monthly usage data for each customer is accumulated and stored in the administration credentials for that month, continually to ensure data accuracy.

Figure 5.12: Administration Monthly Usage

With this, the transmission protocol loop, as illustrated in Figure 5.1, is complete.

5.2 Summary

In short, this chapter delves into the network layer of our project by explaining the transmission protocol in detail. First, the Things Network is connected to the pipedream workflow via a webhook connection & the workflow is connected to our Cloud Firestore via the custom Python script using the Firebase service credentials.

Chapter 6

Application Layer

In order to complete the data visualization objectives of the project, we need to develop user interfaces to retrieve data from the Firebase Database as discussed in Chapter 5. This involves developing both a mobile and a web application.

This chapter provides a comprehensive explanation of the development of the mobile and web applications, their connectivity with the Firebase Database, and the methods used to retrieve data.

6.1 Mobile Application Development

6.1.1 Development Overview

Flutter is an open-source UI development kit used to create cross-platform applications from a single codebase for the Web, Android, iOS, and more. We are using Android Studio IDE for the Flutter framework and Dart as the programming language for the development of our mobile application.

6.1.1.1 Data Visualization

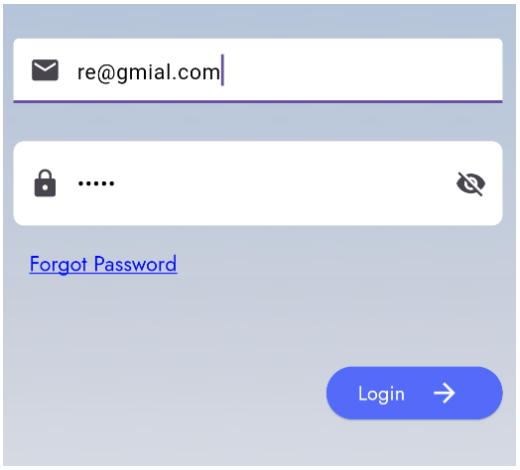
To visualize data, we have developed multiple screens for the app:

- Dashboard Screen: Displays the water usage and bill for the current month.
- Water Usage Screen: Shows the daily and monthly water usage.
- Billing History Screen: Displays the water bills of previous months and the current month.

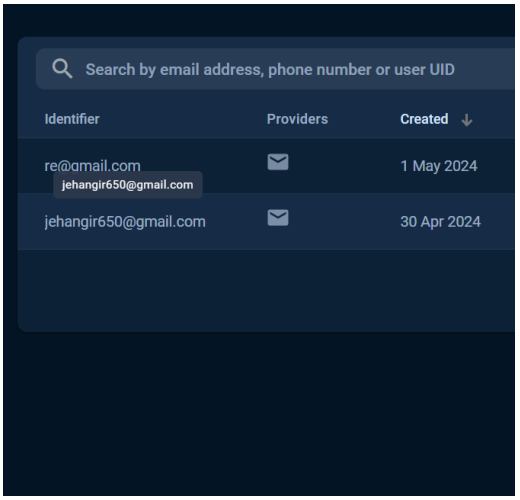
6.1.2 Role-Based Access Control

Upon when user signs in the , user type is verified and they are directed to the **Dashboard** page as our mobile app is only for customers.

- User Authentication: This screen allows users to log into the app using their credentials, which are verified against Firebase Authentication as shown in Figure 6.1 below:



(a) Email and Password Input



(b) User credentials

Figure 6.1: Cross checking User credentials with Email and Password input

6.1.3 Customer Interface

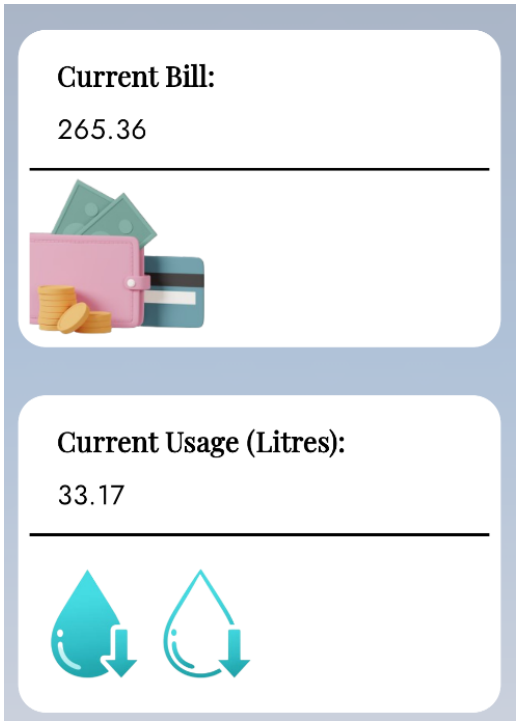
After logging in, customers are directed to their dashboard, where they can access personal information and details about their meter:

- Data Presentation: For data visualization, we're developing multiple screens listed

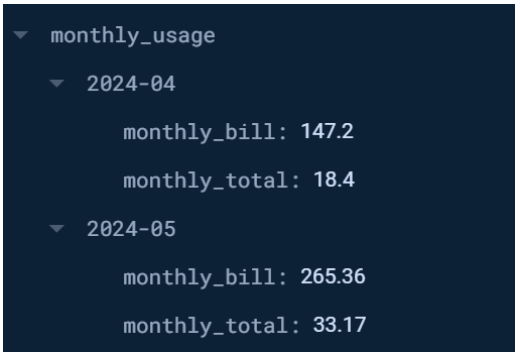
below.

- The **Dashboard Page** displays water usage and bill information for the current month.
- The **Water Usage Page** provides insights into daily and monthly water usage.
- The **Billing History Page** showcases water bills for previous and current months.
- The **Live Data Page** offers real-time data of water flow through the meter.

- Customer Dashboard: Upon sign-in, customers are greeted with the dashboard presenting their monthly bill and water usage. Figure 6.2a & 6.2b shows snippet of Customer Sign-in Page.



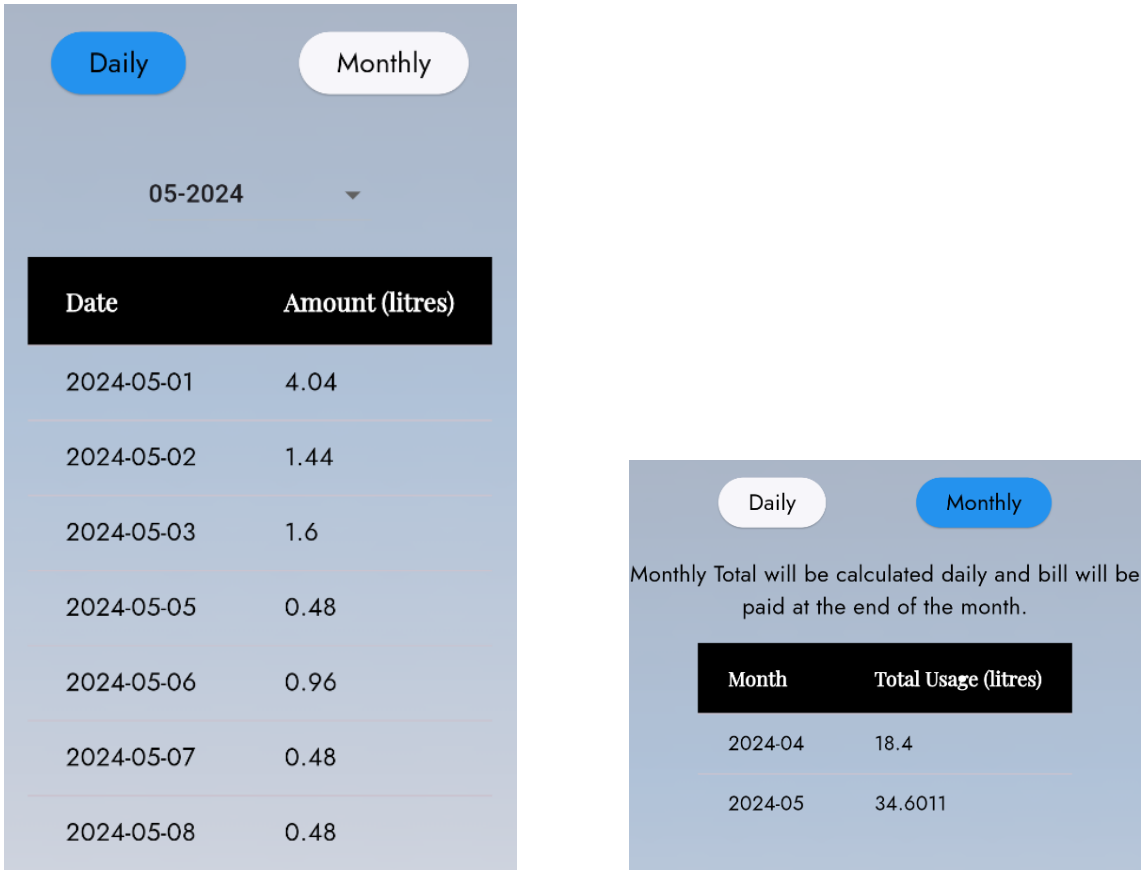
(a) Current Month Usage



(b) Monthly Data in Database

Figure 6.2: Dashboard Screen with Current Bill & Usage

- Usage History: This page displays total water consumption for the current month and day. Figure 6.3a and 6.3b shows snippet of User-history page.

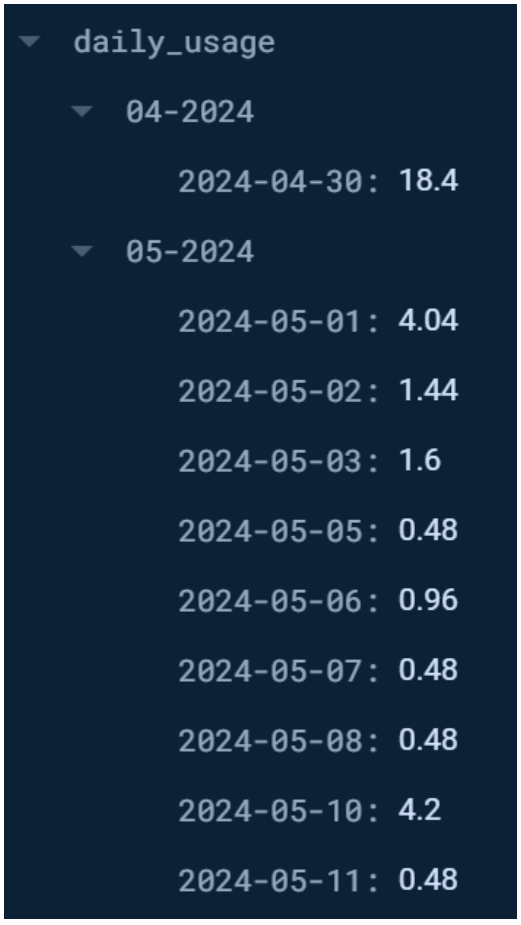


(a) Daily Usage

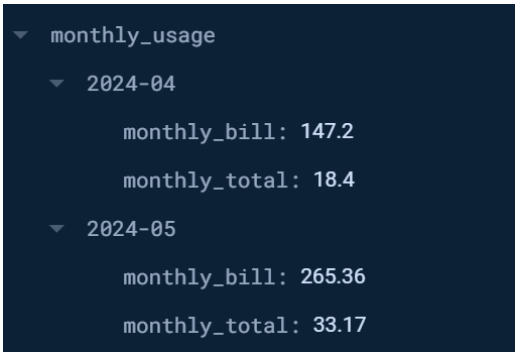
(b) Monthly Usage

Figure 6.3: Usage Screens

This screen retrieves data from Firebase for daily and monthly usage, storing it in table formats as shown in Figure 6.4.



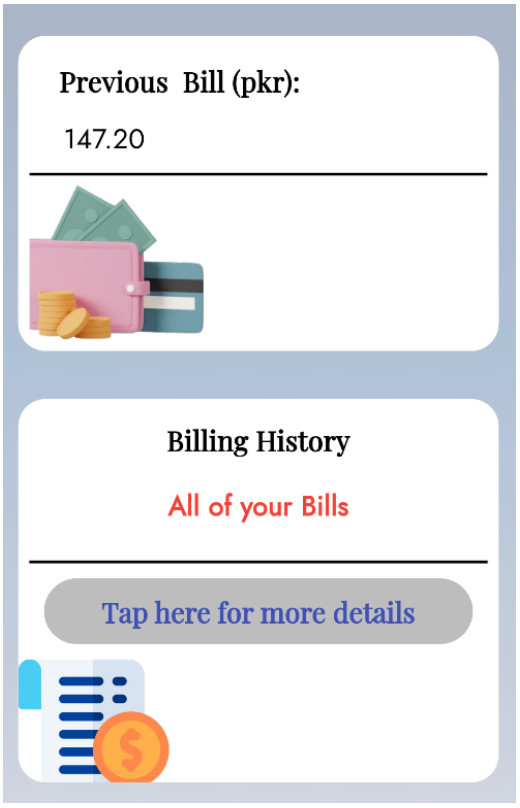
(a) Daily Firebase Storage



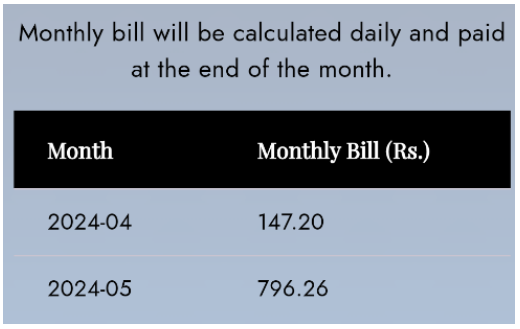
(b) Monthly Firebase Storage

Figure 6.4: Firebase Storage Data

- Billing History: Here, customers can view their total bill for the current month and access a list of previous bills. Figure 6.5 show snippet of billing history and billing list pages.



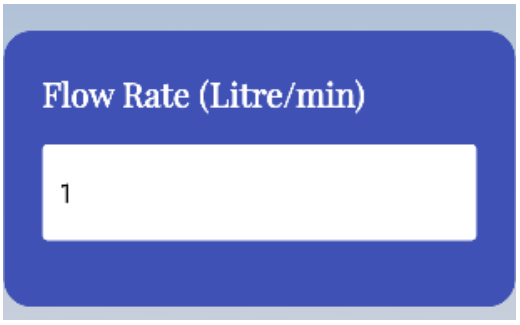
(a) Previous Month Bill



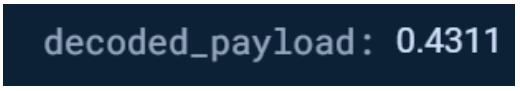
(b) All Months Bill

Figure 6.5: Billing Screens

- Live Data: This page presents real-time water flow data from the customer's meter. Figure 6.6 shows snippet of live data stream page.



(a) Live Flow Screen



(b) Live flow data in Firebase

Figure 6.6: Live Data Screens

6.1.3.1 Utility Screens

These screens serve auxiliary functions for our website:

- My Account: This page displays details about the current user, including username, email, phone number, customer type, and customer ID as shown in Figure 6.7.

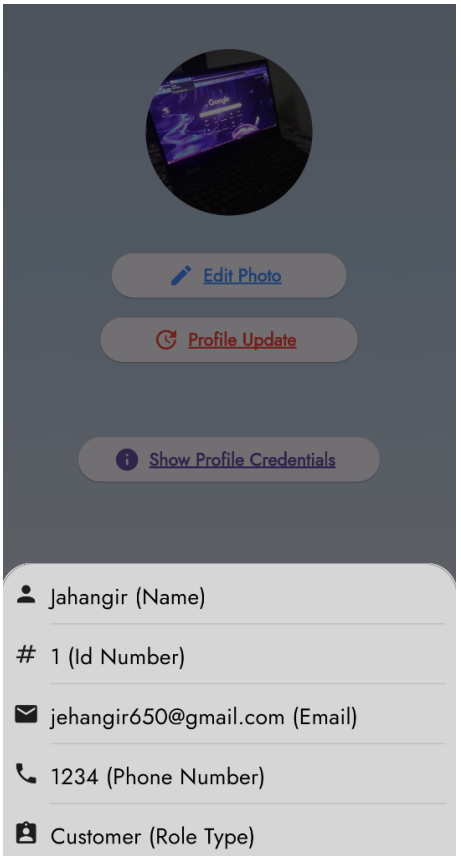


Figure 6.7: User Profile Credentials

- Contact Us Screen: This screen facilitates communication with our team via email, Google Maps, and phone dialer as shown in Figure 6.8.

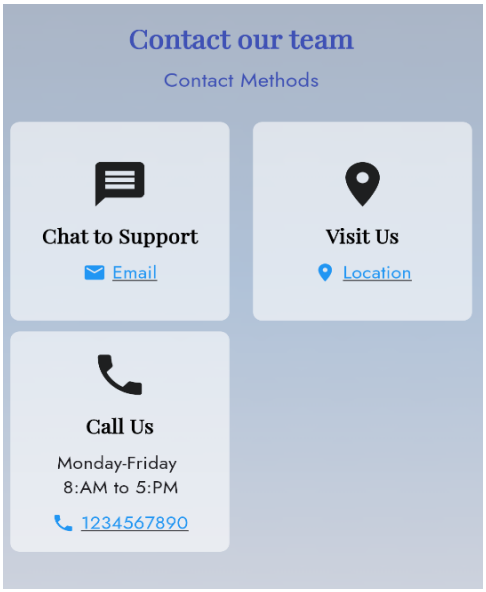


Figure 6.8: Contact Options

- Forgot Password Screen: This screen allows users to reset their passwords by sending a reset email through Firebase Authentication as shown in Figure 6.9.

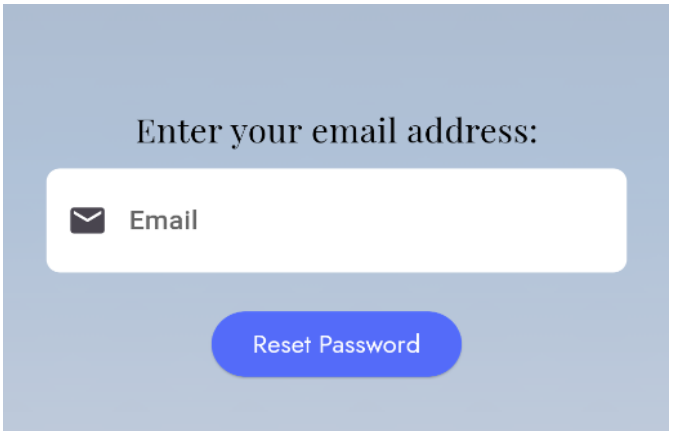


Figure 6.9: Email Reset

6.2 Web Application Development

We have discussed the development of our mobile app for data visualization in the previous section. Now, we will discuss the development of the website in detail.

6.2.1 Development Overview

We use Flutter web for developing the website to maintain the same code base and uniformity in appearance with the mobile app.

6.2.1.1 Data Visualization

We've chosen Flutter for our website development. Flutter is an open-source UI Development Kit known for its ability to create cross-platform applications for various systems, including Web, Android, and iOS.

We'll utilize the Visual Studio Code IDE for the Flutter framework and code in Dart for our web app development.

6.2.2 Role-Based Access Control

Upon user sign-in to the website, their user type is verified, and they're directed to the appropriate pages:

- User Authentication: This page serves as the initial interface of our website for users. Here, users input their email address and password for authentication. Figure 6.10 shows snippet of Sign-in Page.

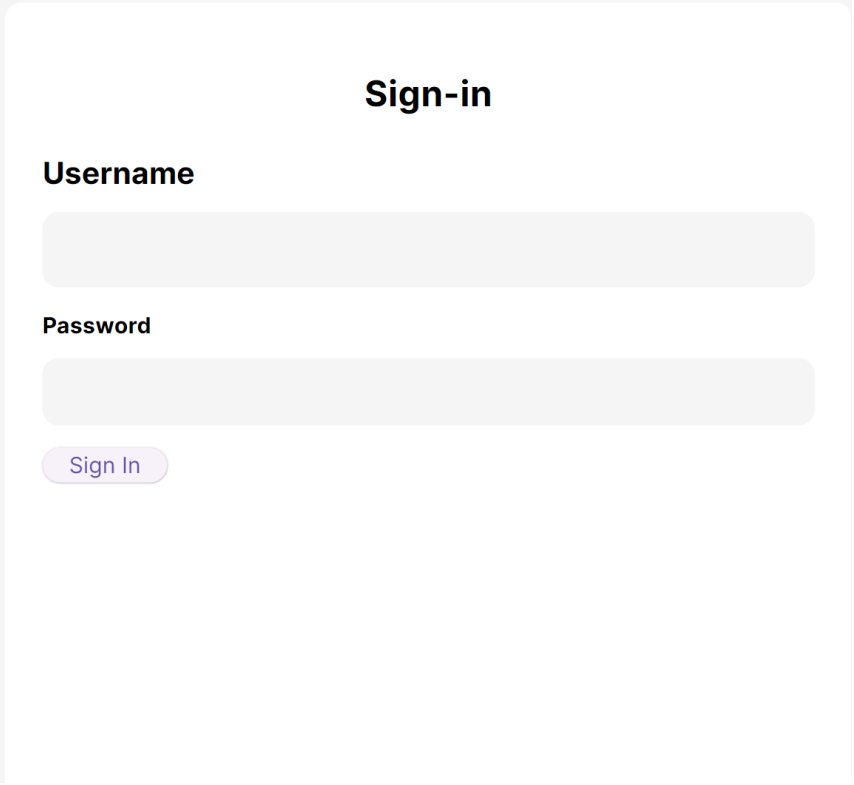


Figure 6.10: Snippet of Sign In Page

6.2.3 Customer Interface

After logging in, customers are directed to their dashboard, where they can access personal information and details about their meter:

- Data Presentation: For data visualization, we're developing multiple screens listed below.
  - The **Dashboard Page** displays water usage and bill information for the current month.
  - The **Water Usage Page** provides insights into daily and monthly water usage.
  - The **Billing History Page** showcases water bills for previous and current months.
  - The **Live Data Page** offers real-time data of water flow through the meter.

- Customer Dashboard: Upon sign-in, customers are greeted with the dashboard presenting their monthly bill and water usage. Figure 6.11 shows snippet of Customer Sign-in Page.

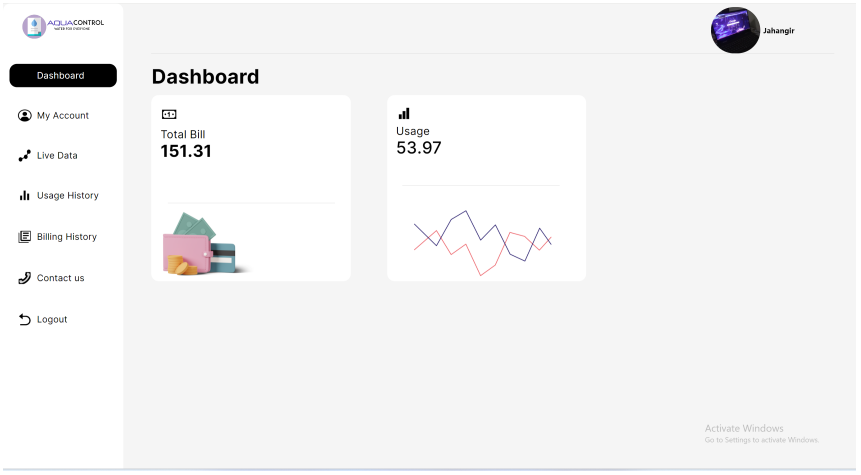


Figure 6.11: Customer Dashboard snippet

- Usage History: This page displays total water consumption for the current month and day. Figure 6.12 shows snippet of User-history page.

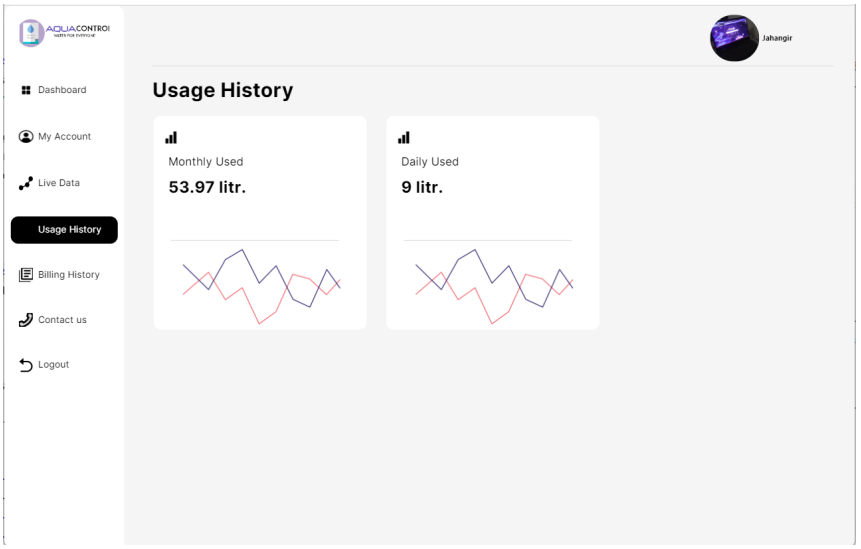


Figure 6.12: Usage History snippet

- Billing History: Here, customers can view their total bill for the current month and access a list of previous bills. Figure 6.13 and 6.14 show snippet of billing history and billing list pages.

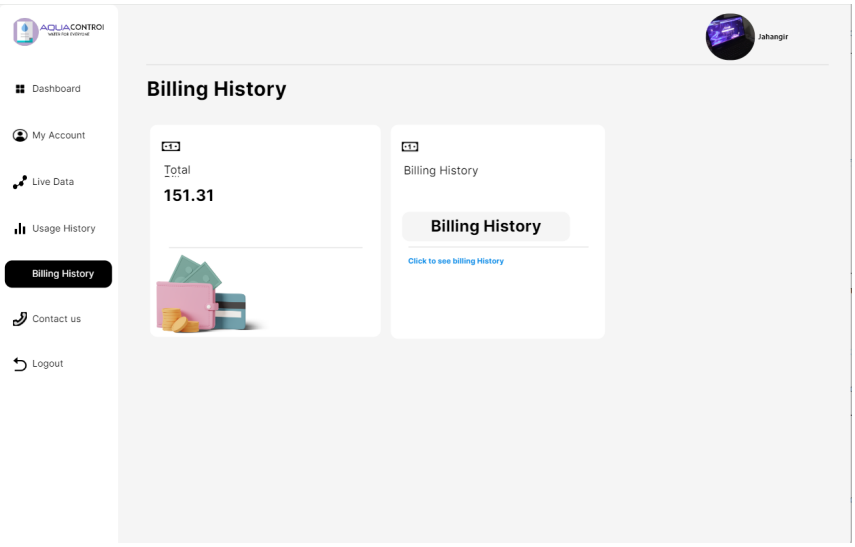


Figure 6.13: Billing History

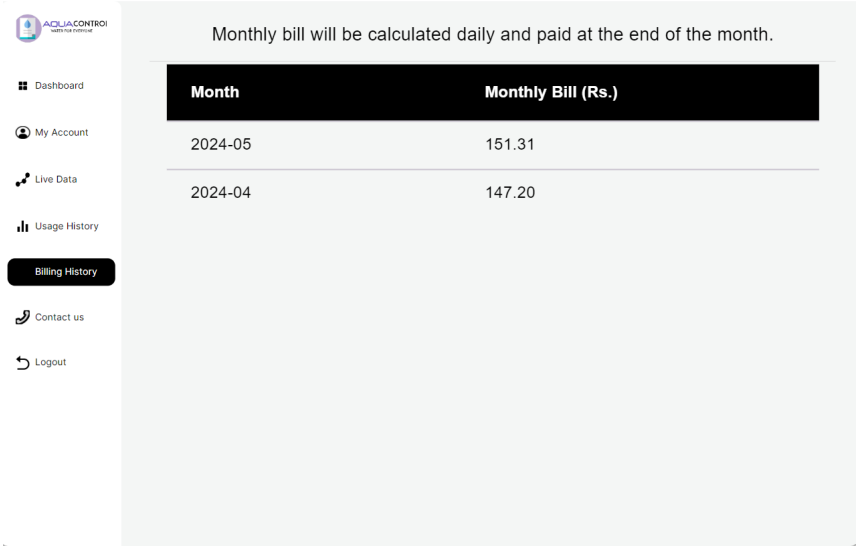


Figure 6.14: Billing List

- Live Data: This page presents real-time water flow data from the customer's meter. Figure 6.15 shows snippet of live data stream page.

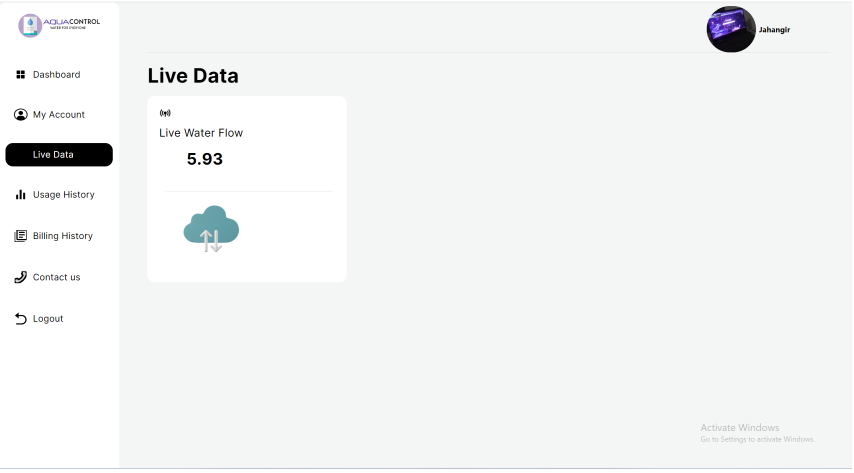


Figure 6.15: Live Data

6.2.3.1 Utility Screens

These screens serve auxiliary functions for our website:

- My Account: This page displays details about the current user, including username, email, phone number, customer type, and customer ID. Figure 6.16 shows snippet of utility screen showing My Account.

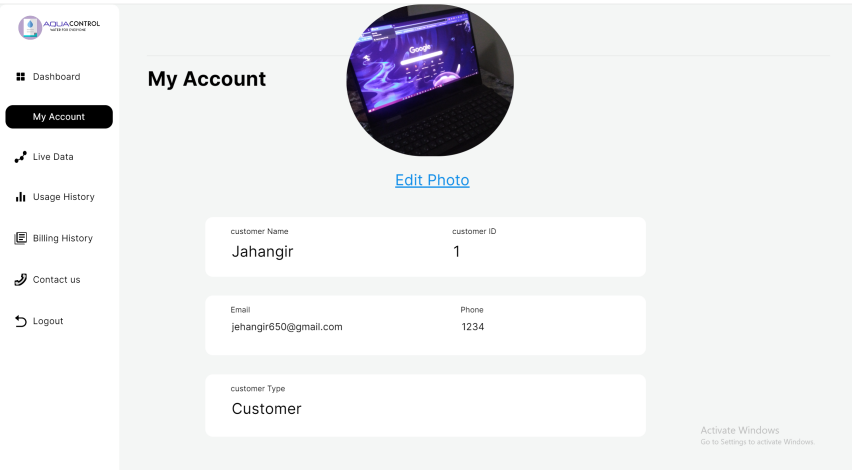


Figure 6.16: My Account

- Contact Us Screen: This screen facilitates communication with our team via email, Google Maps, and phone dialer. Figure 6.17 shows snippet of utility screen showing Contact Us.

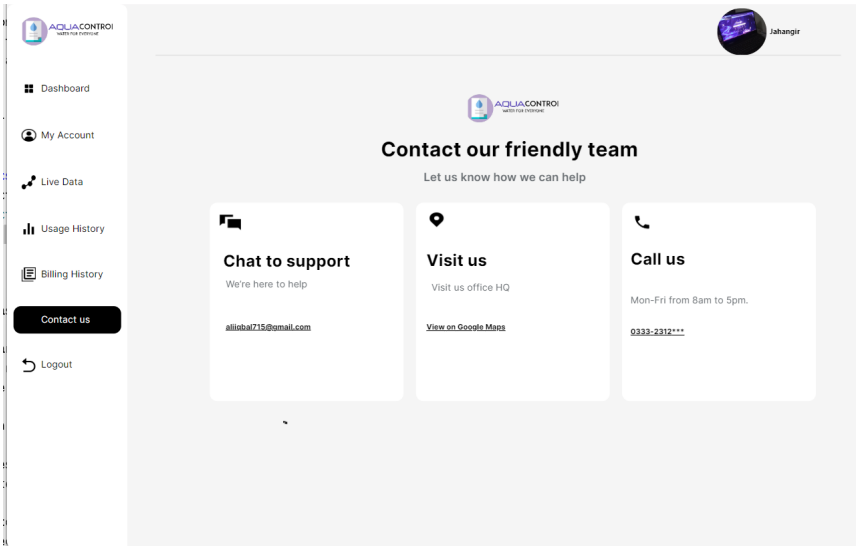


Figure 6.17: Contact Options

6.2.4 Admin Interface

Upon admin login, they're directed to the admin dashboard, providing insights into total water consumption and bills for the entire system and all customers:

- Data Presentation: Similar to the customer interface, we're developing multiple screens for data visualization.
  - The **Dashboard Page** displays water usage and bill information for the current month.
  - The **Water Usage Page** provides insights into daily and monthly water usage.
  - The **Billing History Page** showcases water bills for previous and current months.
  - The **Users Page** displays real-time data of water flow through the meters.
- Admin Dashboard: Upon sign-in, the admin is presented with the dashboard showing total monthly bills, customer count, meter count, and accumulated monthly water usage for all customers. Figure 6.18 shows snippet of Admin Dashboard.

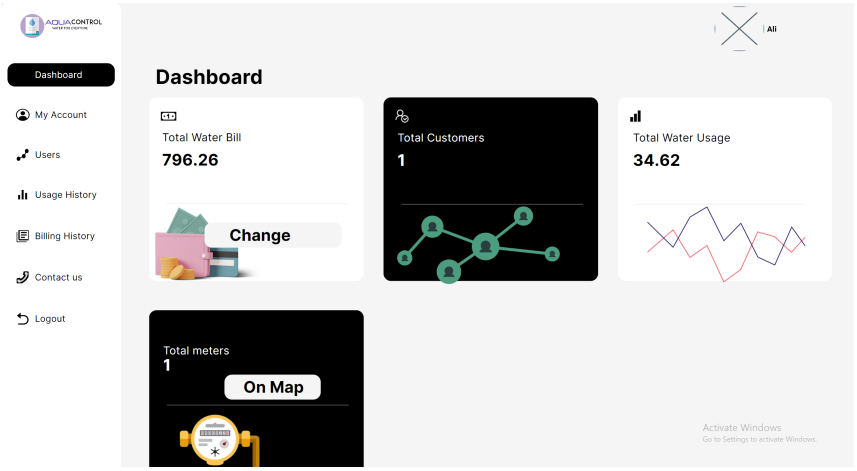


Figure 6.18: Admin Dashboard

- Usage History: This page displays total water consumption for the current month by all customers, daily water usage, and water quality at the main reservoir. Figure 6.19 shows snippet of utility screen showing Usage History.

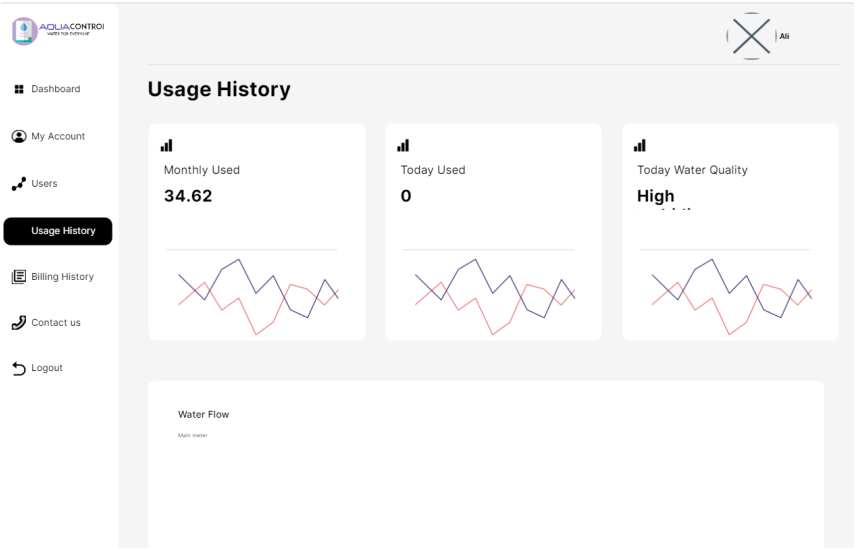


Figure 6.19: Usage History

- Billing History: Admins can view total bills for the current month and access billing history for all users. Figure 6.20 shows snippet of Billing History while Figure 6.21 shows the Admin Billing List Page.

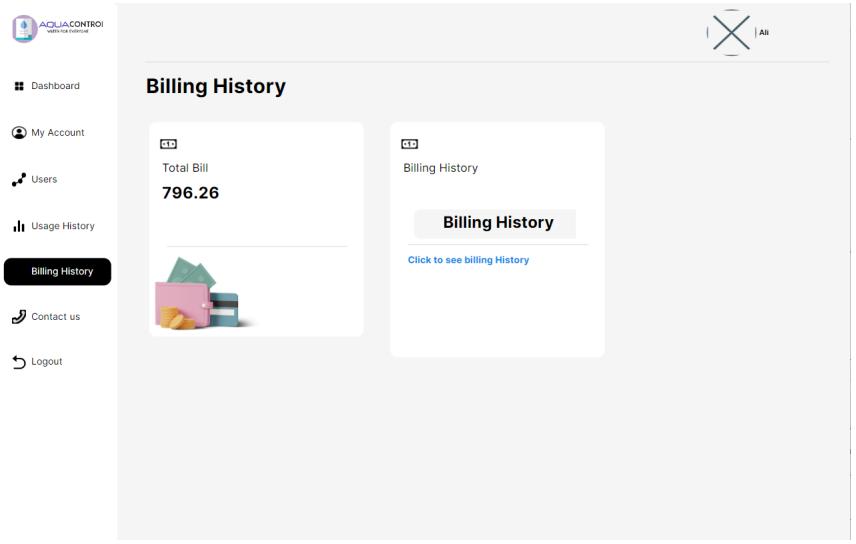


Figure 6.20: Billing History

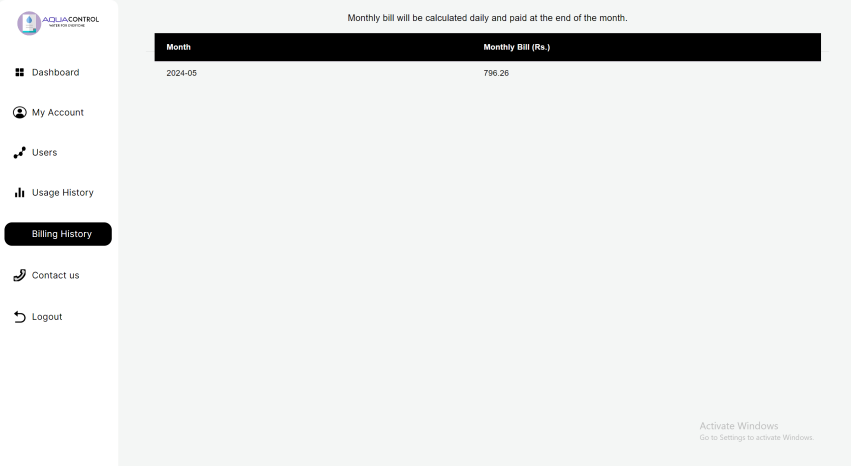


Figure 6.21: Billing List

- Users: The users page lists all registered customers along with their names, emails, and device IDs. Figure 6.22 shows snippet of User details.

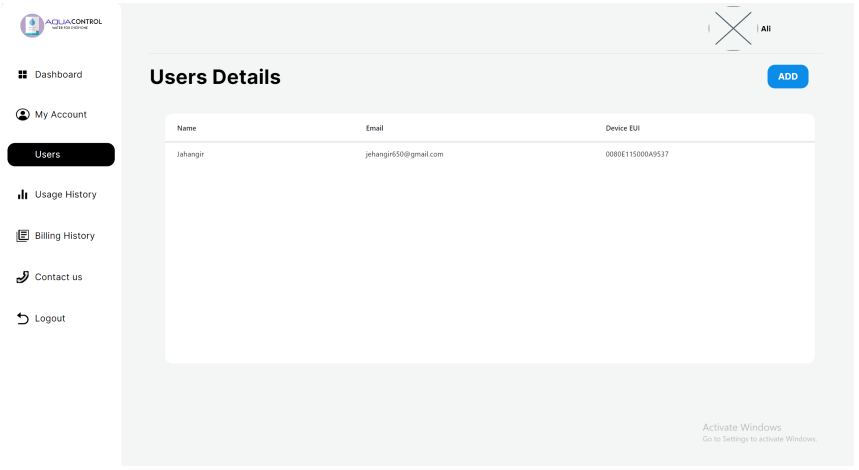


Figure 6.22: Snippet of Details of Users View on Admin Panel

– User Registration: Admins can add new users by providing necessary details such as name, password, ID, phone, email, device EUI, and user type.

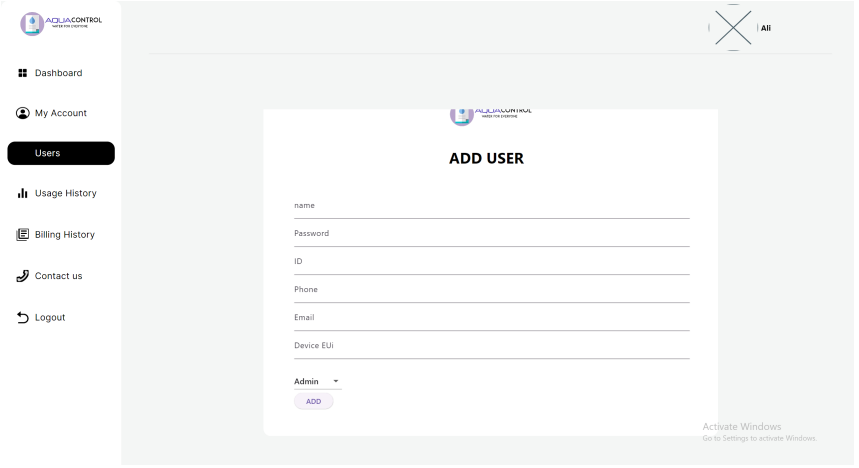


Figure 6.23: User Registration

6.2.5 Utility Screens

These screens serve auxiliary functions for our website:

- My Account: This page displays details about the current user, including username, email, phone number, customer type, and customer ID.

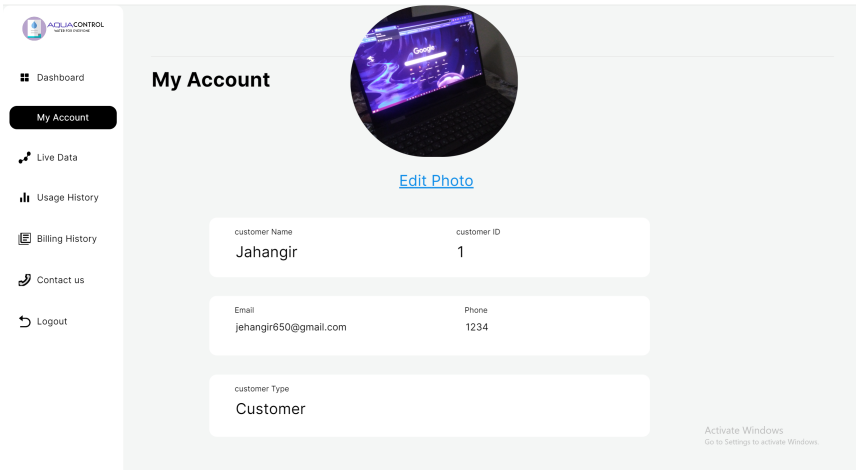


Figure 6.24: My Account

- Contact Us Screen: This screen facilitates communication with our team via email, Google Maps, and phone dialer.

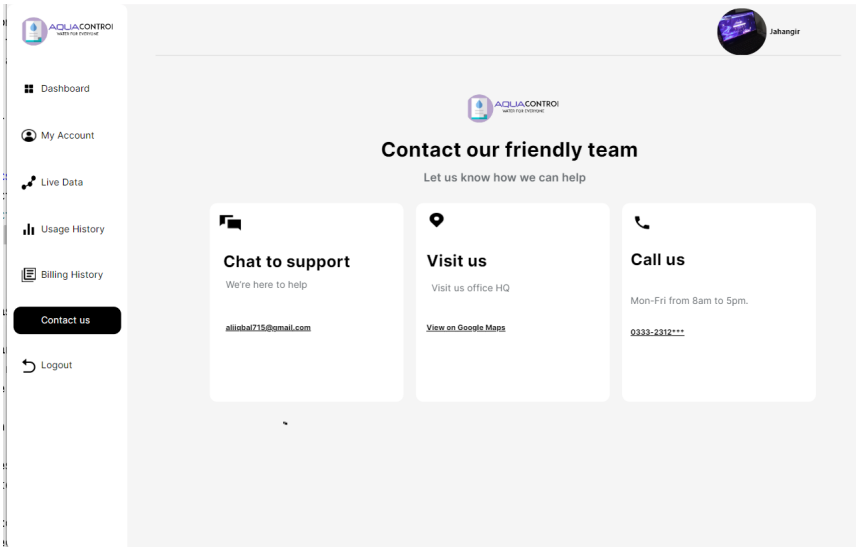


Figure 6.25: Contact Options

6.3 Summary

In this Chapter, we have discussed about the data visualization of our project as how the data being retrieved from Cloud Firestore and being displayed on the mobile and web app for each user respectively.

Also explaining the logic of each screen, detailing how each functions and whether it relies on Firebase. Highlight the importance of each screen to our app.

Chapter 7

Conclusions & Future Work

In this chapter, we will discuss the conclusions that we have reached. The transmission protocol was one of the major topics of our project. As we are introducing a new method of transmitting data instead of using Wi-Fi we are using LoraWan technology for the transmission of data from one end point to the other.

We Will also be discussing about how to improve out project in the future by using different technology such that other people when implement a system like this can make improvements to it.

7.1 Conclusions

7.1.1 Summary of Main Milestones

The main goal of our project was to create and build a smart home IoT water system. We worked our way through different problems and tried alternative platforms until we could create a system that could implement the system effectively. Here is the summary of the main milestones we have completed:

7.1.2 Milestones Accomplished

7.1.2.1 Device Integration with LoraWan:

The STM32-micro controller has a LoraWan module/chip embedded on it as the chip then allows the device be connected to the Things Stack and use LoraWan servers to transmit the data.

7.1.2.2 Transmission Protocol:

We then implemented a transmission protocol to send data from the IoT device to the Things Network. From there to the Pipedream workflow and analyze and process the data in the workflow to store it in the Firebase Database.

7.1.2.3 Data Visualization:

We also developed a mobile and web application for data visualization for customer and administration to see the statistics related to water usage and water quality.

7.2 Future Work

In order to further improve the project, multiple sensors can be used to detect more components of water, to determine if water is clean to use or not.

7.2.1 Usage Water

For better measurement of Usage Water, we can use the sensors below:

7.2.1.1 Temperature Sensors:

Measure the temperature of water to determine if water is set at a specific temperature according to the environment to make it easy for customers to use it.

7.2.1.2 Conductivity Sensors:

Measure the electrical conductivity of water in order to determine the concentration of dissolved salts & impurities.

7.2.1.3 Oil-in-Water Sensors:

In order to detect the presence of oil & hydrocarbons in water, in order to determine if safe to use or not.

7.2.2 Drinking Water

Such that by the using different sensors, people can also use this project to determine if water is safe to drink or not:

7.2.2.1 Chlorine Sensors:

Measure the chlorine concentration in water as chlorine is used as a disinfectant in water to kill insects.

7.2.2.2 Conductivity Sensors:

Measure the electrical conductivity of water in order to determine the concentration of dissolved salts & impurities.

7.2.2.3 Microbial Sensors:

Detect the presence of bacteria, viruses etc to determine if water is drinkable or not.

7.2.2.4 Oil-in-Water Sensors:

In order to detect the presence of oil & hydrocarbons in water, in order to determine if safe to use or not.

7.2.2.5 Dissolved Oxygen Sensors:

To detect the dissolved oxygen levels in the water as high oxygen levels determine that the water is drinkable and low levels do not.

Bibliography

[1] M. Hasan, "STM32 ADC tutorial using DMA with HAL Code Example | Embedded There — embeddedthere.com." https://embeddedthere.com/stm32-adc-tutorial-using-dma-with-hal-code-example/. [Accessed 03-06-2024].

[2] "mantech.co.za." http://www.mantech.co.za/datasheets/products/yf-s201_sea.pdf. [Accessed 03-06-2024].

[3] E. Fahad, "pH meter Arduino, pH Meter Calibration, DIYMORE pH Sensor Arduino Code — electroniclinic.com." https://www.electroniclinic.com/ph-meter-arduino-ph-meter-calibration-diymore-ph-sensor-arduino-c [Accessed 03-06-2024].

[4] "Adding Devices — thethingsindustries.com." https://www.thethingsindustries.com/docs/devices/adding-devices/. [Accessed 03-06-2024].

[5] "Webhooks — thethingsindustries.com." https://www.thethingsindustries.com/docs/integrations/webhooks/. [Accessed 03-06-2024].

[6] "Python - Pipedream — pipedream.com." https://pipedream.com/docs/code/python. [Accessed 03-06-2024].

[7] "Flutter documentation — docs.flutter.dev." https://docs.flutter.dev/. [Accessed 03-06-2024].

[8] "Get Started with Firebase Authentication on Android — firebase.google.com." https://firebase.google.com/docs/auth/android/start. [Accessed 03-06-2024].

[9] "Get started with Cloud Storage on Android | Cloud Storage for Firebase — firebase.google.com." https://firebase.google.com/docs/storage/android/start. [Accessed 03-06-2024].

[10] "Get started with Cloud Firestore | Firebase — firebase.google.com." https://firebase.google.com/docs/firestore/quickstart. [Accessed 03-06-2024].

[11] Y. Ye, Y. Yang, L. Zhu, J. Wang, and D. Rao, "A lora-based low-power smart water metering system," pp. 301–305, 2021.

[12] E. Simonoska, D. C. Bogatinoska, I. Dimitrievski, and R. Malekian, "Sensor system for real-time water quality monitoring," pp. 114–119, 2023.

[13] K. Shanmugam, M. E. Rana, D. Tan Zi Xuen, and S. Aruljodey, "Water quality monitoring system: A smart city application with iot innovation," pp. 571–576, 2021.