YAŞAR UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

COMP4910 Senior Design Project 1, Fall 2020
Supervisor: HÜSEYİN HIŞIL

AQUALITY: Smart Water Sensor Platform
Final Report

19/12/2022

By:
19070001008, Doğu Alpay
18070001033, Mert Sadık Sezigen
17070001013, Berkin Akbıyık

PLAGIARISM STATEMENT

This report was written by the group members and in our own words, except for quotations from published and unpublished sources which are clearly indicated and acknowledged as such. We are conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism according to the University Regulations. The source of any picture, graph, map or other illustration is also indicated, as is the source, published or unpublished, of any material not resulting from our own experimentation, observation or specimen collecting.

Project Group Members:

Name, Lastname	Student Number	Date
Doğu Alpay	19070001008	
Mert Sadık Sezigen	18070001033	
Berkin Akbıyık	17070001013	

Project Supervisors:

Name, Lastname	Department	Date
HÜSEYİN HIŞIL	Computer Engineering	

ACKNOWLEDGEMENTS

We would like to thank our supervisor for her assistance and comments to improve our project structure.

We are also grateful that the Microver Electronics and the founder Umit Kayacik for their suppliance. Also, thanks to Batuhan Kaya who works for Microver as a hardware engineer, for his personal interest in our project and important suggestions.

KEYWORDS

Embedded systems, aquarium, water quality, water temperature, circuit design, Amazon Web Services, over the air software update.

ABSTRACT

When you chat with the people around you, when it comes to the aquarium, you will see that almost everyone is either interested in the aquarium or has an acquaintance who is interested. How many people know how vast and scientific a hobby such a well-known hobby in our country is? Among the reasons for most of us to start aquarium hobby; There are features such as having an aquarium at home when we are born, the desire to have a living thing, the parents' desire to give responsibility to their children, the pleasant appearance of the aquarium and giving people peace of mind. Aquality enhances the aquarium hobby, preventing fish from wasting away and providing a better aquarium experience. It has water quality sensors and calculates the quality of the water according to its temperature and TDS (Total Dissolved Solid) value. It can connect to internet by Wi-Fi and communicate with cloud web server. It has the ability to update software over the air. It also has a LAN server that will allow the user to change the network and user settings.

ÖZET

Çevrenizdeki kişilerle sohbet ettiğinizde konu akvaryuma gelince görürsünüz ki hemen herkes ya akvaryum ile ilgilenmiştir ya da ilgilenen bir tanıdığı vardır. Ülkemizde bu kadar iyi bilinen bir hobinin ne kadar uçsuz bucaksız, ne kadar bilimsel bir hobi olduğunu acaba kaç kişi biliyor? Çoğumuzun akvaryum hobisine başlama sebepleri arasında; doğduğumuzda evde akvaryum olması, bir canlıya sahip olma isteği, ebeveynlerin çocuklarına sorumluluk vermek istemesi ile akvaryumun görüntüsünün hoşa gitmesi ve insana huzur vermesi gibi özellikleri yatıyor. Aquality akvaryum hobisini geliştirerek balıkların telef olmasını engeller ve daha iyi bir akvaryum deneyimi sunar. Su kalitesi sensörlerine sahiptir ve suyun kalitesini sıcaklığına ve TDS(Total Dissolved Solid) değerine göre hesaplar. Wi-Fi ile internete bağlanabilir ve bulut web sunucusu ile iletişim kurabilir. Yazılımı havadan güncelleme yeteneğine sahiptir. Ayrıca, kullanıcının ağ ve kullanıcı ayarlarını değiştirmesine izin verecek bir LAN sunucusuna sahiptir.

Table Of Contents

LIST OF FIGURES

LIST OF TABLES

Abbreviations & Descriptions

Abbreviation	Description
OTA	Over the air. We used it to mention over-the-air software updates. It means that the device will be able to get software updates from us if it is connected to the internet.
TFT	Thin film transistor. It is widely used in LCD screens. It tells about type of the display panel.
Buzzer	Buzzer is a vibrating device that makes beep noise to interact with the user in specific ways.
MCU	Microcontroller unit. It is an integrated circuit to deal with specific tasks. We can see it as a little computer that is used in relatively small projects.
IDE	Integrated development environment
Eclipse	Eclipse is an open-source IDE (integrated development environment) which makes it very easy to build, deploy and manage a software.
TDS	Total dissolved solid. The unit is PPM (parts per million).
LVGL	Light and Versatile Embedded Graphics Library
FreeRTOS	FreeRTOS is a real-time operating system which is for developing light weight real time applications in embedded systems. Controlling a rocket is a good example of that kind of application. It makes it possible to work with multiple tasks simultaneously.
RTOS	Real-time operating system.
ESP32	ESP32 is the microcontroller unit model that we will use in this project which is developed by Espressif company in China.
ESP-IDF	ESP-IDF is Espressif's official IoT Development Framework. It comes with its own self-sufficient SDK for any kind of application. It works with FreeRTOS preferably.
Espressif IDE	Espressif IDE is an Eclipse fork, which includes ESP-IDF framework in it. So it is a gold mine for us. Because normally ESP-IDF does not come with an IDE environment, it has a command line interface.
AWS	Amazon Web Services. It is a service that makes it possible to control any kind of project from the cloud.
AWS S3	S3 is a file server service. We will be storing over the air update files with that service.
AWS IoT Core	IoT Core is an IoT service. We will generate a device on that service and run any kind of process on that device. Therefore, we will be communicating with IoT Core servers.
IoT	Internet of things
LAN	Local area network
LAN server	A server that is hosted by the network source device.
DS18B20	DS18B20 is a temperature sensor, and it can reach the digital data resolution up to 12bits and has ±0.5°C accuracy from -10°C to +85°C.

1. INTRODUCTION

1.1. Description of the Problem

Those who deal with aquariums often encounter problems such as greening of the water and algae formation, inability to maintain the water temperature, and the loss of fish due to the increase in the waste rate in the water. Due to the lack of a useful system that instantly monitors the temperature and cleanliness of the water, this causes both material and moral damages.

There is a detailed description of the problem in Appendix A: Requirement Specifications Document.

1.2. Project Goal(s)

The main purpose of our product, develop a smart water sensor platform for aquariums or other water tanks. It will collect diagnostics data and share it with the user through AWS. The device will also be able to receive OTA software updates when connected to a local Wi-Fi network with a stable internet connection. It will have a small TFT screen that displays some diagnostics and device states, and a desktop application will be available for monitoring the device diagnostics. The device will help the user monitor water quality and will alert the user through a physical buzzer and AWS tools if the water quality is poor.

1.3. Project Output(s)

COMP 4910 outputs are:

- Project assignment form
- RSD v1.0 and v2.0 document
- DSD v1.0 document
- Final Report
- Project poster
- Project web page
- Project presentation

1.4. Project Activities and Schedule

In our project, we will be submitting the following items at the specified times: the first version of the problem definition, the project assignment form, the first and second versions of the requirements specification documents, version one of the design specification document, the final report, the project presentation, and the project poster.

Activity (Started Date)	Explanation
Problem Definition (20.09.2022)	The problem was discussed and identified as a team on 25.09.2022.
PAF (26.09.2022)	Project assignment form is prepared and submitted on 29.09.2022.
RSD v.1.0 (22.10.2022)	The first version of the requirements specification document was prepared on October 22, 2022 and submitted on November 15, 2022.
RSD v.2.0 (18.11.2022)	Second version of the requirement specification document is prepared on 03.12.2022.
DSD v.1.0 (04.12.2022)	The design specification document is prepared on 12.12.2022
Final Report (13.12.2022)	The final report was prepared on 15.12.2022
Project Presentation (16.12.2022)	
Project Poster (17.12.2022)	

2. DESIGN

2.1. High Level Design

We will connect to S3 for file sharing and IoT Core for cloud communications, two different AWS services. These are S3 and IoT Core. They will provide us a gateway for cloud communications and over the air software updates.
We will have two sensors such as TDS and water temperature sensors. By using them we will get data about the water and warn the user when needed by using a buzzer. With an on board TFT display screen, the user will be able to see the diagnostics by checking the screen whenever he/she wants, or the user may also see the diagnostics by using our desktop application, in real time.

Below you can see a brief system design diagram.

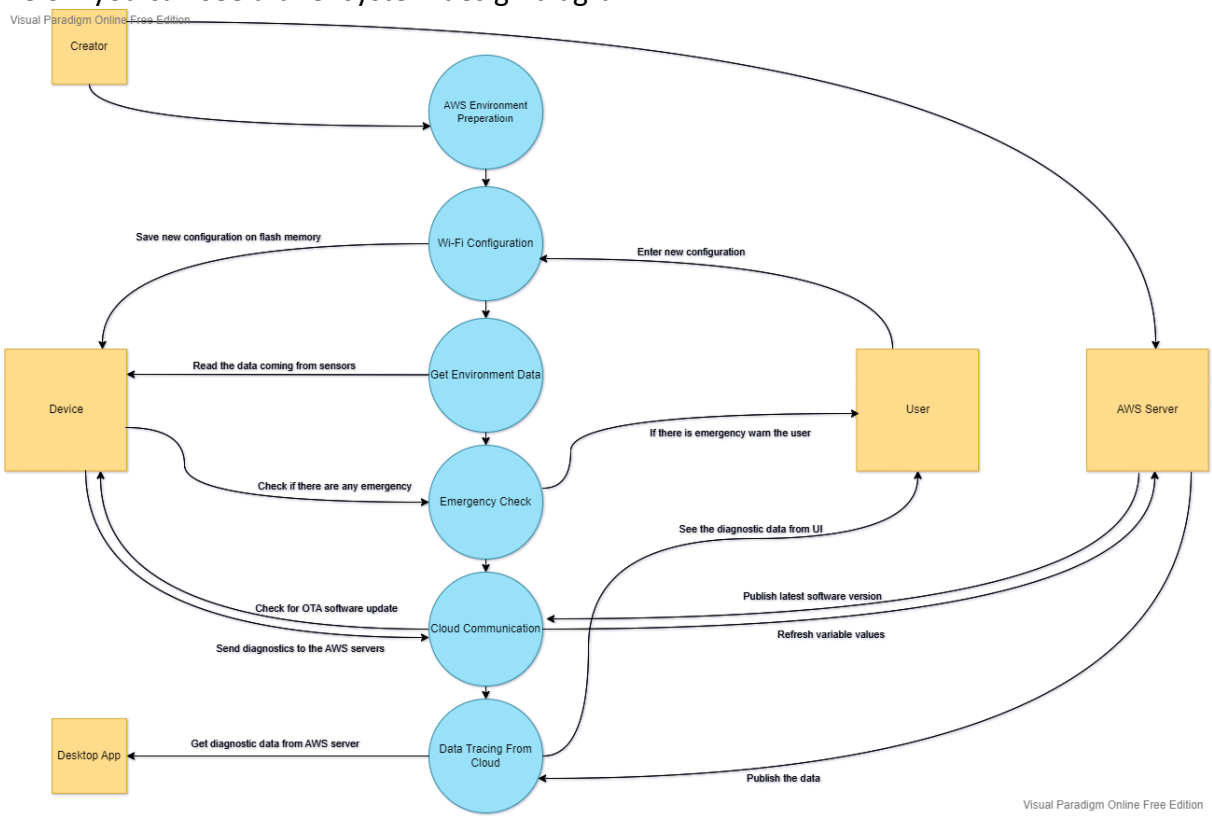

Figure 1 - System Context Diagram

2.2. Detailed Design

In detail, we can mention the hardware layer and real time operating system design.

Two environmental inputs from the sensors, such as total dissolved solid and water temperature, are available on the hardware layer. As environment outputs, we have a buzzer and a display screen. which are all managed by our ESP32-C3 microcontroller. The hardware layer schematic for Aquality is shown below.

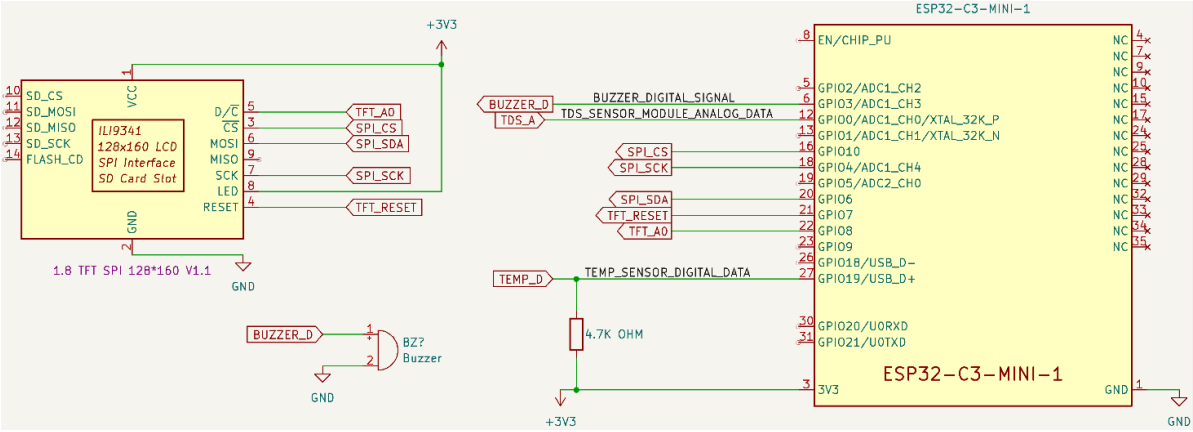

Figure 2 - Hardware Layer Schematic

We'll be working on an embedded environment at the software layer. Without an operating system, we were able to code, but doing so would be improper. As a result, we will build Aquality using the FreeRTOS operating system. In comparison to more complicated operating systems like Linux, FreeRTOS is a real-time operating system with less complexity but less functionality. FreeRTOS will be able to schedule and complete our small number of jobs because we will be concentrating on a single application and won't have any more work to accomplish.

The task structure is more clearly displayed below.

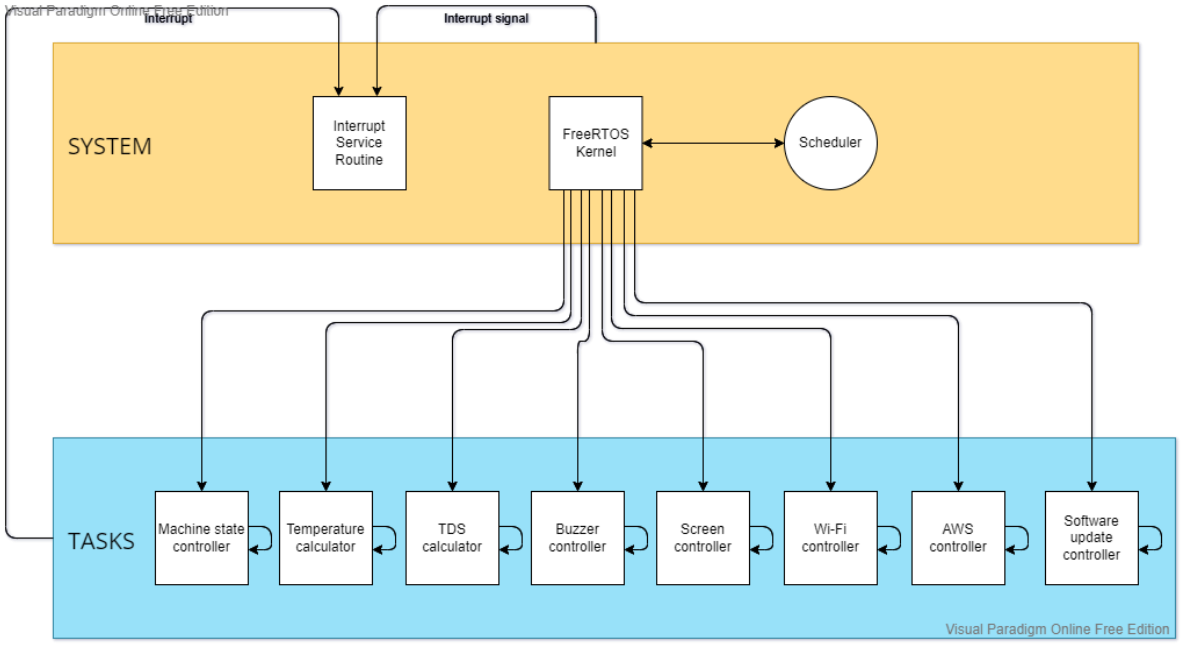

Figure 3 - Software Task Structure

2.3. Realistic Restrictions and Conditions in the Design

- In our design, there is no extra security other than the products and services are using.
- The desktop application may not support more than 500 device data, but it is unlikely for a user to have 500 devices, at least for now.
- There may be power supply problems since we fully trust the onboard USB connector.

3. IMPLEMENTATION, TESTS and TEST DISCUSSIONS

3.1. Implementation of the Product

To successfully implement the product, we will use Espressif IDE and ESP-IDF framework that Espressif MCUs are fully compatible with. By choosing the ESP-IDF framework we simply accept using a real-time operating system for handling our tasks in a half simultaneous way of processing. That RTOS is FreeRTOS. And FreeRTOS provides us a well-documented SDK for any possible application.

You can find an image below that shows assembled look of our system.

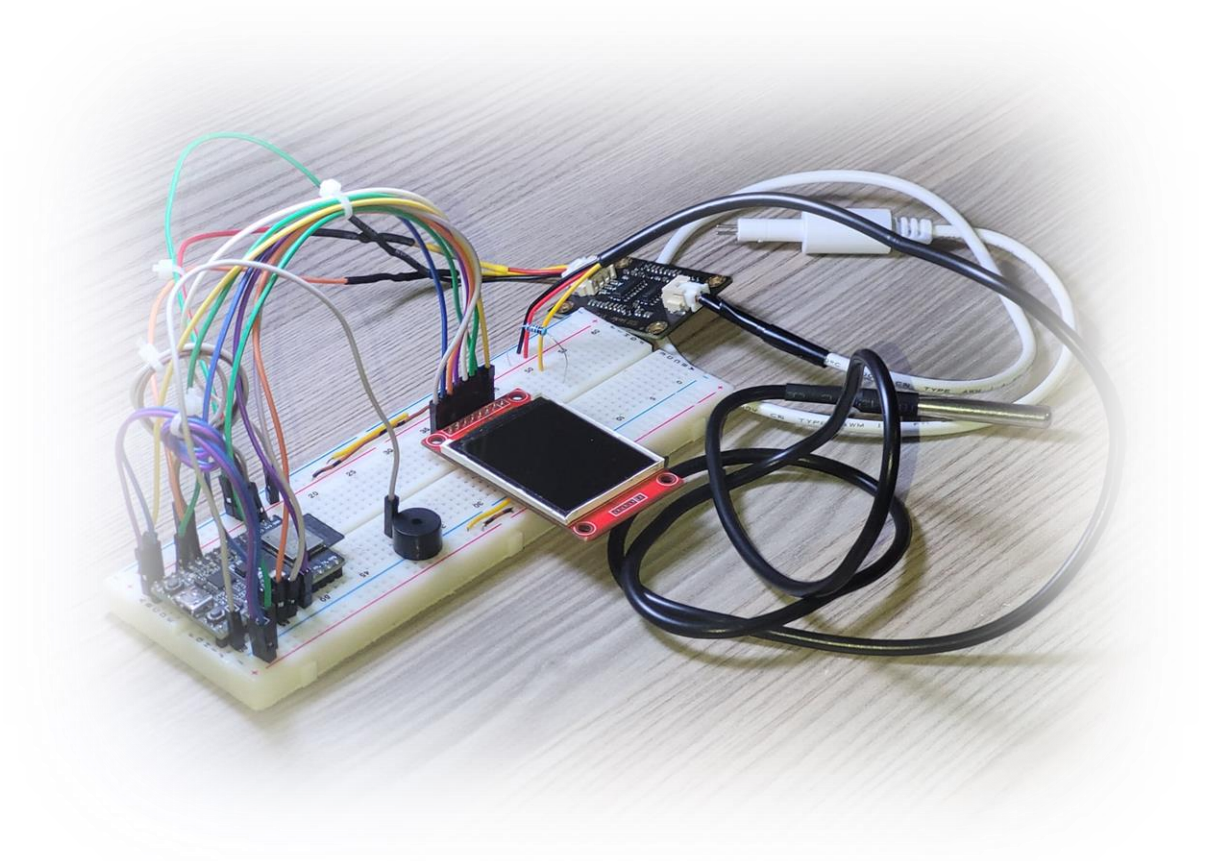

Figure 4 - System Look

3.2. Tests and Results of Tests

The most crucial phase of a project's development is its testing phase. Since this project involves several hardware communication layers, we will test our product specifically for it.

- There will be tests of the environment, hardware, and software.
- Through environmental tests, we will check whether our components are strong and effective enough to match hardware specifications.
- Each use case of our software requirements will be tested in software tests, while voltage and currents at specific spots on the schematic will be tested in hardware tests.
- The tests and anticipated outcomes are defined in the tables. For software tests, at least 1000 test cycles must be performed before the results may be considered complete.

Below you can find the hardware & environmental tests.

Test	Expected Result
Leave the sensor probes in 50cm depth of a water tank for one day.	The sensors probes shouldn't be worn out, broken or took water inside. Should be working properly and with the same precision.
Leave the system open for one day.	Each component on the system should be working fine. Any kind of screen, buzzer, sensor, or development board issue is not tolerable.
Check the peripherals' power consumption.	Peripherals must meet the power consumption equality with datasheet of itself.
Check the voltage values of each power line on the schematic.	Every voltage measurement should meet the requirement of the specific peripheral.
Use different power supply for better check on the powerline stability.	The voltage measurements and power consumptions must match with the above tests. There should not be any difference on the system stability.
Screen test with random animations and LVGL functionalities for 1 minute.	The screen should not flicker and freeze. The fps should appear on the screen and there should be an average of bigger than 20 frames per second. The colours should look fine. There shouldn't be any teleportation or lose on the UI objects.
Compare TDS value with a certified TDS sensor device.	The TDS values should match. -+5 PPM precision is expected.
Compare temperature value with certified waterproof temperature sensor device.	The temperature values should match. -+0.5 degrees Celsius precision is expected.
Check the buzzer activation for 1 minute.	The buzzer sound should be the same. The sound should not stutter or freeze.

Below you can find the software tests.

Test	Expected Result
Change Wi-Fi SSID & password and try to connect to the Wi-Fi with new SSID & password. (10 times)	The device should automatically connect to the Wi-Fi with given new SSID & password.
Change Wi-Fi hotspot password and try to connect to hotspot with new password. (10 times)	We should be able to connect to the device's Wi-Fi hotspot with the new password. When we try to access the configuration page should appear correctly.
Connect to AWS	Device should automatically connect to the AWS server if the configuration is done, and an internet connection is valid.
Connect to AWS IoT Core	Device should automatically connect to the AWS IoT Core server if the configuration is done, and an internet connection is valid.

Connect to AWS S3	Device should automatically connect to the AWS S3 server if the configuration is done, and an internet connection is valid.
Get software update (10 times)	The device should get the updates and reset the device in 1 minute if there are any new version of the software.
Change screen state and watch the transitions with the sensor values.	Every object of UI should appear correctly, and the functions of the screen should keep working.
Publish diagnostic data on 10 different shadows from the lower to the maximum values. (10 times)	All scope of the data should appear on the desktop application dashboard of Aquality one by one.
Listen to the published data on the above task. (10 times)	All scope of the data should appear on the desktop application dashboard of Aquality one by one.
Try to enter invalid value to the configuration variables. (10 times)	The system should not accept invalid scope of data. There should be a filter system for configuration inputs.
Watch the FreeRTOS tasks with Segger SystemView tool. (10 times)	Every task should be running as expected. There shouldn't be any task that is not working or working in an unexpected way. System memory and disk usage should not be high at any time. The context switching operations must be flawless.

4. CONCLUSIONS

4.1. Summary

In conclusion, the aquarium hobby is a popular and scientific pursuit that is enjoyed by many people. There are various reasons why individuals may be drawn to this hobby, including a desire to have a living thing in their home, a desire to give their children responsibility, and the calming effects of an aquarium. Aquality is a smart water sensor platform that aims to improve the aquarium hobby by providing water quality information and alerts to the user. This device connects to the internet via Wi-Fi and can receive over-the-air software updates, and also includes a LAN server for changing network and user settings. In addition, a desktop application is available for monitoring device diagnostics. Overall, Aquality aims to enhance the aquarium experience and prevent fish from wasting away by ensuring the water quality is optimal.

We have identified the main goal of the project and discussed the features that should be included. We have also created documents that outline the system requirements, including the Requirement specification and Design specification, in order to guide the development process and ensure that all necessary features are included. We have also identified the technologies and tools that we will use for the development of the project, including

Amazon Web Services, the C programming language, and the Espressif IDE. We have made significant progress in planning the project and are ready to move forward with development. After careful research, we have chosen to use a development board with an ESP32 microprocessor for the circuit of our project. We searched for the lowest cost display that this processor could best control and decided to use a 128x160 TFT display. Since we did not have many options for the remaining sensors and materials, we provided the ones that fit our budget best.

4.2. Cost Analysis

Week	Dates	Doğu Alpay		Mert Sadik Sezgen		Berkin Akbıyık		-		Total Weekly Effort in Man-Hours
		Work Done	Total Hours Spent	Work Done	Total Hours Spent	Work Done	Total Hours Spent	Work Done	Total Hours Spent	
Week 1	15.09.2022 - 06.10.2022	Identified the group members	10	Identified the group members	10	Identified the group members	10			30,00
Week 2		Identified the project topic	10	Identified the project topic	10	Identified the project topic	10			30,00
Week 3		Researched about possible examples	10	Researched about possible examples	11	Researched about possible examples	10			31,00
Week 4		PAF designed	15	PAF designed	15	PAF designed	15			45,00
Week 5	10.10.2022 - 16.10.2022	Started to the RSD v1.0 document, introduction is written & discussed	15	Started to the RSD v1.0 document, introduction is written & discussed	15	Started to the RSD v1.0 document, introduction is written & discussed	15			45,00
Week 6	17.10.2022 - 23.10.2022	Research about peripherals and electronic components that we will use in project. Functional requirements are written & discussed. Non-functional requirements are written & discussed. References addition to the document. Abbrevations are defined and added to the document.	15	Research about AWS and AWS services that we will use. Functional requirements are written & discussed. Non-functional requirements are written & discussed. Abbrevations and table of contents addition to the document.	15	Worked on Wi-Fi requirements specifically. Functional requirements are written & discussed. Non-functional requirements are written & discussed. List of figures addition to the document.	10			40,00
Week 7	24.10.2022 - 30.10.2022		10		12		15			37,00
Week 8	31.10.2022 - 06.11.2022		10		10		10			30,00
Week 9	07.11.2022 - 13.11.2022		15		13		12			40,00
Week 10	14.11.2022 - 20.11.2022	Use case diagrams are designed.	20	Use case diagrams are designed.	15	Use case diagrams are designed.	15			50,00
Week 11	21.11.2022 - 27.11.2022	Started DSD v1.0, added introduction. Researched aboud possible software design strategies and structures. Context diagram added to the document. Worked on software subsystem design and software test design sections.	14	Started DSD v1.0, added introduction. Researched aboud possible hardware design strategies and structures. Worked on hardware subsystem design and hardware test design sections. Finalized the DSD document.	15	Started DSD v1.0, added introduction. Researched aboud AWS structural design. Worked on system design section.	15			44,00
Week 12	28.11.2022 - 04.12.2022		15		13		12			40,00
Week 13	05.12.2022 - 11.12.2022	Prepared final report in detail.	15	Prepared final report. Prepared website.	13	Prepared final report. Prepared poster.	10			38,00
Week 14	12.12.2022 - 16.12.2022		10		15		11			36,00
Week 15	19.12.2022	Some little details added to the documents. And prepared presentation.	8	Some little details added to the documents. And prepared presentation.	8	Some little details added to the documents. And prepared presentation.	10			26,00
Week 17										0,00
Week 18										0,00
Total Effort in Man-Hours			192,00		190,00		180,00		###	562,00
Total Effort in Man-Days			24,00		23,75		22,50		###	70,25

4.3. Benefits of the Project

There are several benefits to using a Aquality which measures water temperature and water cleanliness for aquariums:

Improved fish health: Proper water temperature and cleanliness are essential for the health and well-being of aquarium fish. By regularly monitoring these conditions, you can ensure that your fish are living in a healthy environment.

Early detection of problems: By continuously monitoring the water temperature and cleanliness, you can quickly identify any problems that may arise and take action to correct them. This can help prevent serious issues from developing and potentially save the lives of your fish.

Reduced maintenance: Regularly checking the water temperature and cleanliness manually can be time-consuming and labor-intensive. Using a device to automate this process can save you time and effort, allowing you to focus on other aspects of aquarium maintenance.

Peace of mind: Knowing that your fish are living in optimal conditions can give you peace of mind and allow you to enjoy your aquarium hobby with confidence.

4.4. Future Work

In future, we may want to add some functional peripherals that can be triggered remotely to the device to make Aquality fully interactive and autonomous. Some examples for these peripherals would be solenoid valve for controlling waterflow, water leak sensor, and chemical material or fish food ejectors.

Other than that, the display could be a bigger one and capacitive touch could also be an option.

References

- Aquality Requirements Specification Document v1.3
- Aquality Design Specification Document
- https://www.freertos.org/Documentation/RTOS_book.html

APPENDICES

APPENDIX A: REQUIREMENTS SPECIFICATION DOCUMENT

COMP4910 Senior Design Project 1, Fall 2022
Advisor: Hüseyin Hışıl

AQUALITY: Smart Water Sensor Platform Requirements Specifications Document

18.12.2022
Revision 2.3

19070001008, Doğu Alpay
18070001033, Mert Sadık Sezigen
17070001013, Berkin Akbıyık

Revision History

Revision	Date	Explanation
1.0	19/11/2022	Initial requirements are set
1.1	24/11/2022	Functional requirements, non-functional requirements, flows, data requirements are set
1.2	05/12/2022	UML diagrams and some new requirements/use cases are added. Some requirements/use cases are changed.
2.0	14/12/2022	Abbreviations explained, system schematic and peripheral images added
2.1	15/12/2022	Data requirements are explained. Table of figures added.

2.2	16/12/2022	Changed the data type of water temperature sensor that we will read according to the datasheet.
2.3	17/12/2022	Added new variables to the data requirements table.

Table of Contents

Abbreviations & Descriptions

Abbreviation	Description
OTA	Over the air. We used it to mention over-the-air software updates. It means that the device will be able to get software updates from us if it is connected to the internet.
TFT	Thin film transistor. It is widely used in LCD screens. It tells about type of the display panel.
Buzzer	Buzzer is a vibrating device that makes beep noise to interact with the user in specific ways.
MCU	Microcontroller unit. It is an integrated circuit to deal with specific tasks. We can see it as a little computer that is used in relatively small projects.
IDE	Integrated development environment
Eclipse	Eclipse is an open-source IDE (integrated development environment) which makes it very easy to build, deploy and manage a software.
TDS	Total dissolved solid. The unit is PPM (parts per million).
LVGL	Light and Versatile Embedded Graphics Library
FreeRTOS	FreeRTOS is a real-time operating system which is for developing light weight real time applications in embedded systems. Controlling a rocket is a good example of that kind of application. It makes it possible to work with multiple tasks simultaneously.
RTOS	Real-time operating system.
ESP32	ESP32 is the microcontroller unit model that we will use in this project which is developed by Espressif company in China.
ESP-IDF	ESP-IDF is Espressif's official IoT Development Framework. It comes with its own self-sufficient SDK for any kind of application. It works with FreeRTOS preferably.
Espressif IDE	Espressif IDE is an Eclipse fork, which includes ESP-IDF framework in it. So it is a gold mine for us. Because normally ESP-IDF does not come with an IDE environment, it has a command line interface.
AWS	Amazon Web Services. It is a service that makes it possible to control any kind of project from the cloud.
AWS S3	S3 is a file server service. We will be storing over the air update files with that service.
AWS IoT Core	IoT Core is an IoT service. We will generate a device on that service and run any kind of process on that device. Therefore, we will be communicating with IoT Core servers.
IoT	Internet of things
LAN	Local area network
LAN server	A server that is hosted by the network source device.
DS18B20	DS18B20 is a temperature sensor, and it can reach the digital data resolution up to 12bits and has ±0.5°C accuracy from -10°C to +85°C.

1. Introduction

1.1 Purpose

The purpose of this document is to determine the needs for developing a smart water sensor platform that is going to collect diagnostics from an aquarium or any kind of water tank and then share the data with the user through AWS. This device will also get OTA software update whenever a new version of software is uploaded to the AWS file server if the device is connected to the local Wi-Fi network with a stable internet connection. The device will also have a small TFT screen that will show some diagnostics and some device states on it. We will also develop a desktop application to monitor the device diagnostics. The device will help the user to follow water quality and warn the user by using a physical buzzer and AWS tools when the water quality is poor.

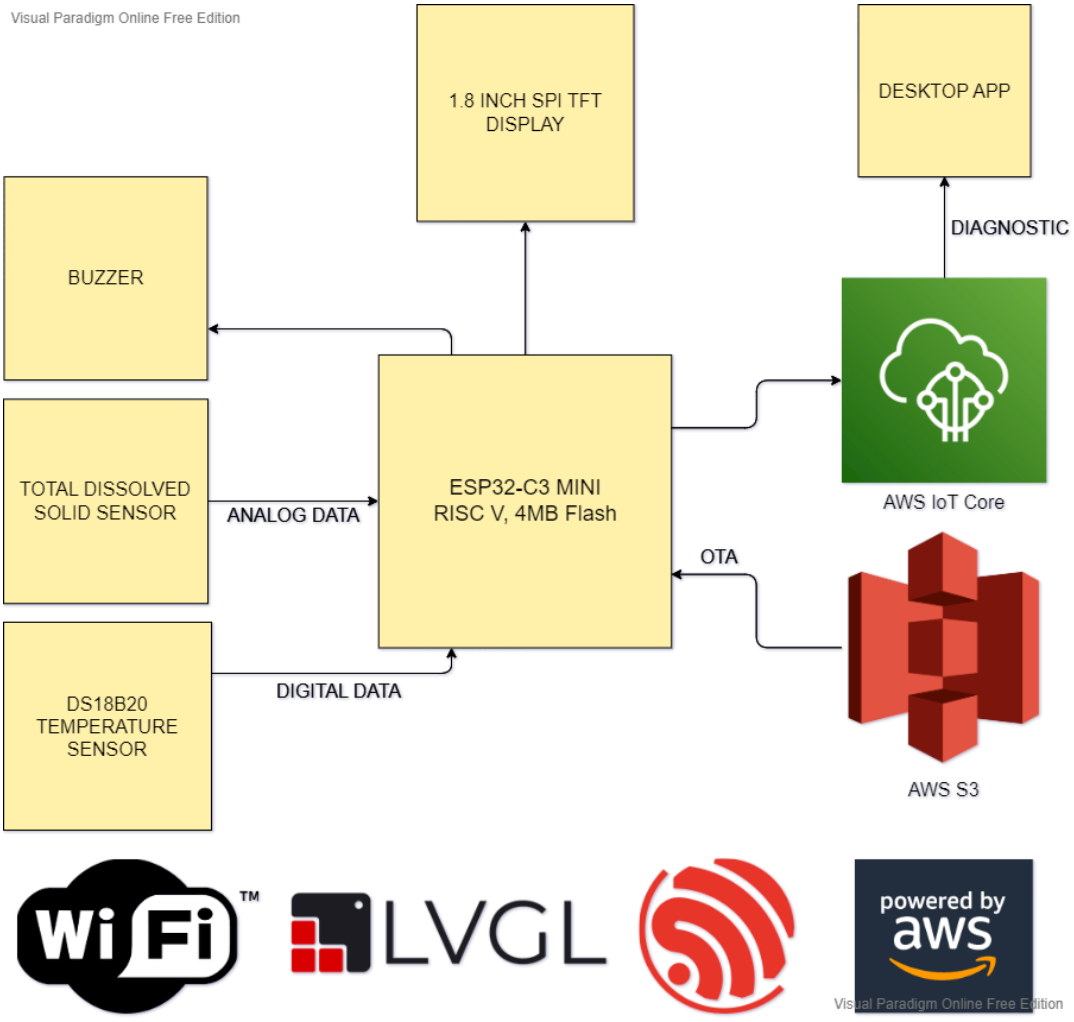

Figure 5 - System Blueprint

1.2 Intended Audience

This document is useful for students who are working on this project and their advisors as much as the customer. Since the students will do most of the job this document will be important most of the time for them.

1.3 Scope

➢ **Development Environment**

We will be using ESP32 MCU while developing this product. Since we want to work with C programming language, we will be developing on ESP-IDF framework that is supplied by Espressif company to develop bare metal C projects on their MCUs. There is also an IDE that is supplied by Espressif again, called Espressif IDE. That IDE runs Eclipse in the background. In the meantime, the MCU will include FreeRTOS since it is fully supported.

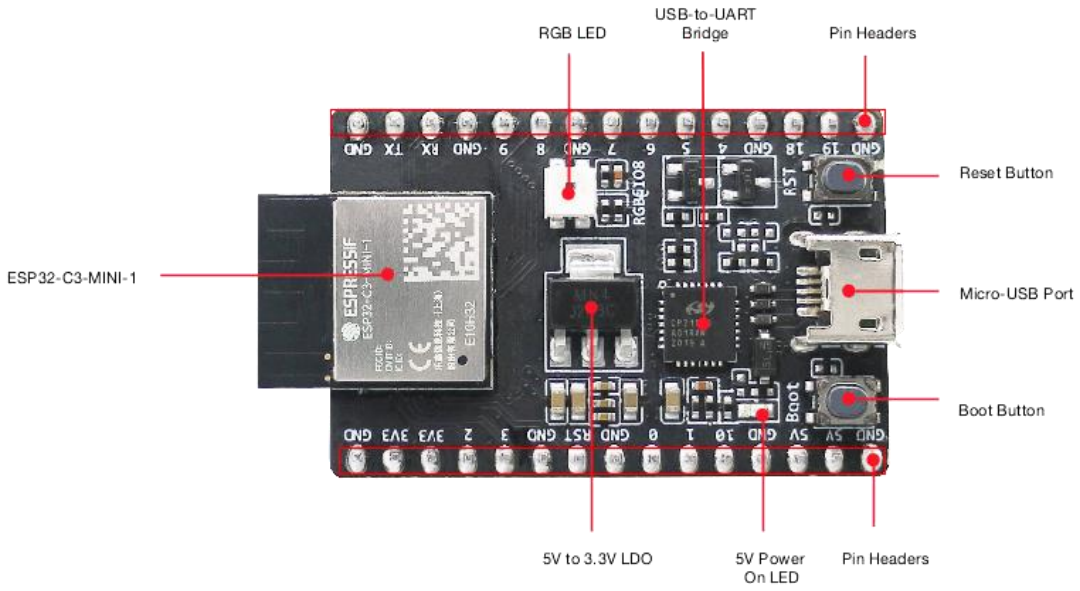

Figure 6 - ESP32-C3-DevKitM-1 Front

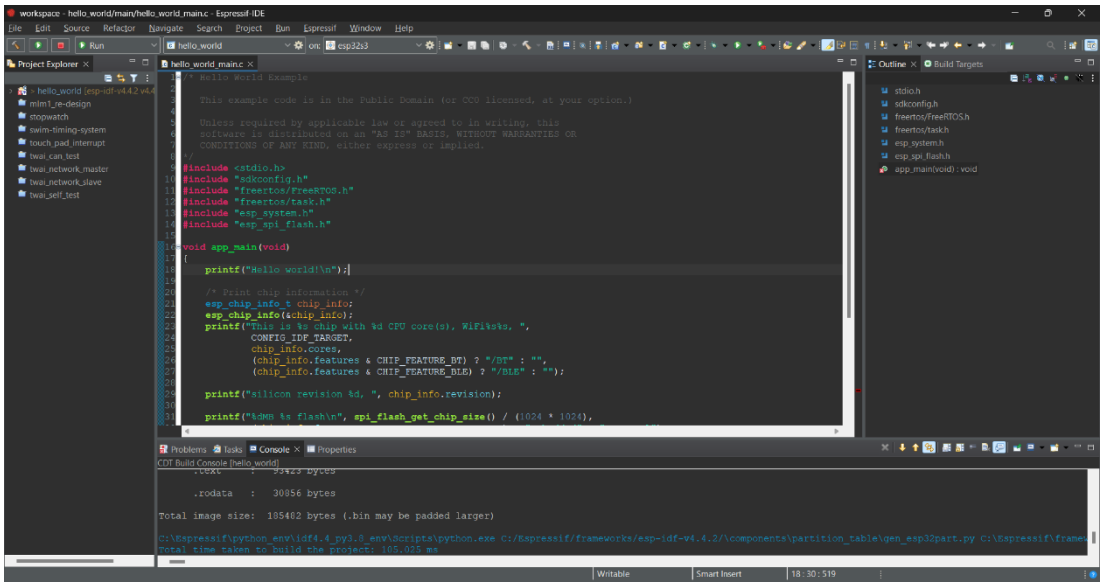

Figure 7 - Espressif IDE

- **Sensors & Peripherals Signaling**

The signalization and data extraction out of the sensors is the main work of this product. Therefore, we need clean communication between the MCU and other peripherals. For that we need to understand how TDS and temperature sensors work and how do we calculate the meaningful data by using their values, since they are the only sensors of product. We will have one buzzer as output and use it for warning the user when the water quality is poor.

Figure 8 - TDS Sensor Module & Probe

Figure 9 – DS18B20 Waterproof Temperature Sensor

Figure 10 - Buzzer

➢ **Wi-Fi Connection**

The ESP32 MCU has a built in Wi-Fi module and antenna that supports 2.4 GHz communication frequency. We will be creating a LAN web server to make it possible to communicate with the user to get some data, like home Wi-Fi connection information etc. Then the device will try to connect to the local Wi-Fi, if the Wi-Fi has internet connection the device will be able to connect to AWS services and will work as expected.

➢ **Amazon Web Services**

We will be using two different services of AWS. These are AWS IoT Core for diagnostics sharing and AWS S3 as a file server for OTA software updates. AWS IoT Core will also let us monitor the diagnostics of each device separated from each other with their unique ID's.

➢ **UI Design**

Since we will use a TFT screen, we will need a graphics library to make development easy for us. Little VGL is the graphics library that we are looking for. We will use it as ported version for ESP32 MCUs. LVGL also supplies a UI designer that is called SquareLine Studio. We will be using that designer for UI development.

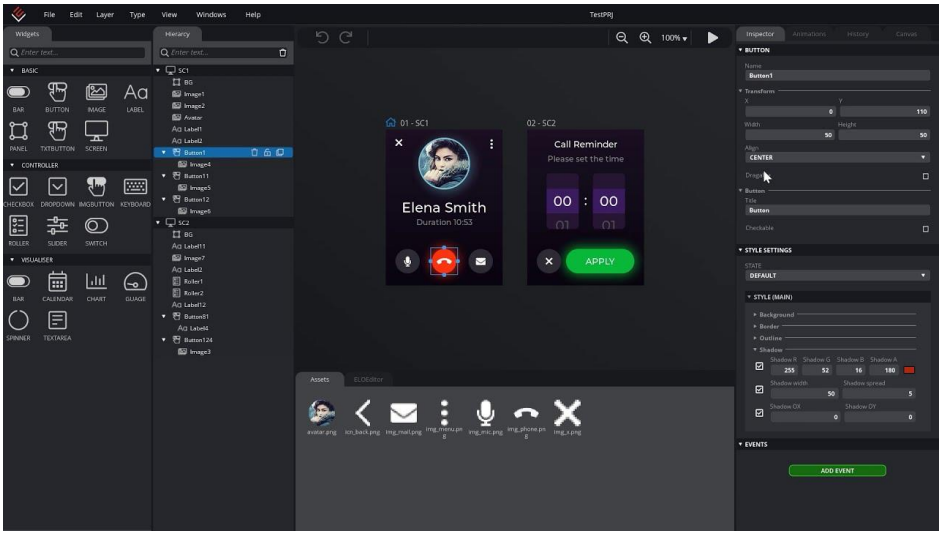

Figure 11 - SquareLine Studio

Figure 12 - 1.8-inch SPI TFT LCD Display Module, 128x160

➢ **Desktop Application Development**

The desktop application will be developed in Qt framework. It will connect to AWS IoT Core service by using AWS IoT Device SDK for Python. Each user will be able to monitor his/her own devices' diagnostics only.

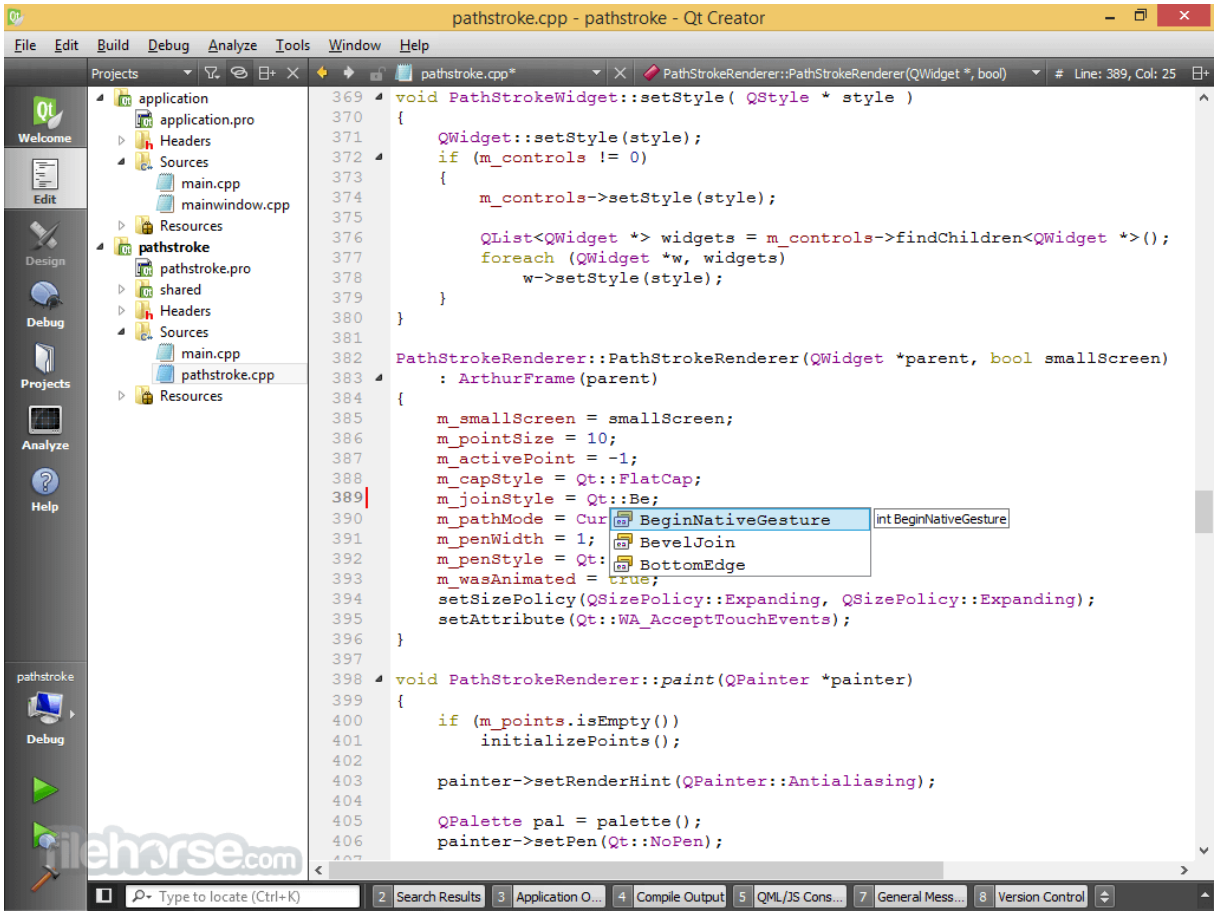

Figure 13 - Qt Creator & Designer

1.4 Overview

The rest of the RSD contains the details about requirements and functions. The RSD is organized with the standards of IEEE RSD Standards.

2. Requirements & Use Cases

2.1 Functional Requirements

2.1.1 Sensors & Peripherals Signaling

No.	Requirement	Use Case
1	To read analog TDS value from TDS sensor probe	Get analog TDS value
2	To read digital temperature value from temperature sensor probe	Get digital temperature value
3	To filter the noisy analog values while reading	Filter the analog values
4	To calculate meaningful TDS value by using some offsets and current temperature	Calculate meaningful TDS value
5	To calculate meaningful temperature value by using some offsets	Calculate meaningful temperature value
6	To use buffer while reading analog values	Use buffer
7	To calculate the average values from their buffers to get stable results on higher refresh rates	Get average for each buffer of value
8	To activate the buzzer for warning the user	Activate the buzzer

2.1.2 Wi-Fi Connection

No.	Requirement	Use Case
1	To set up Local Area Network which allows connected users to configure the device	Create LAN server
2	To change the Wi-Fi SSID and password, so that the device can connect to the internet by using those	Change Wi-Fi SSID & password
3	To change only the password of the device Hotspot for making the structure a bit more secure and configurable	Change Hotspot password
4	To connect internet through local Wi-Fi network	Connect internet through Wi-Fi
5	To connect hotspot, for being able to change network configurations	Connect hotspot
6	To save the network configurations on SPIFFS	Save network config
7	To change network configurations such as Wi-Fi SSID & password and hotspot password	Change network config

2.1.3 Amazon Web Services

No.	Requirement	Use Case
1	To configure the device to connect to AWS IoT.	Register AWS account
2	To setup the services and the devices will be connected to AWS by saving necessary information	Setup AWS services & devices
3	To get device certificate while creating a new device on IoT Core	Get device certificate
4	To get device private key while creating a new device on IoT Core	Get device private key
5	To get Amazon Root CA while creating a new device on IoT Core	Get Amazon Root CA
6	To publish a message to a topic with diagnostics in it	Publish message to a topic
7	To upload the latest version of software image to the S3 server	Upload new software image
8	To update the new version of software download link on the S3 server software update text file as a JSON value	Update new software link
9	To get OTA software update	Get software update

2.1.4 TFT Display UI Design

No.	Requirement	Use Case
1	To design the overall UI style and positioning	Design the UI
2	To make UI look modernized	Use effects on UI
4	To show TDS value	Show TDS value
5	To show water temperature value	Show temperature value
6	To show overall water quality by using a variable symbol	Show water quality
7	To show internet connection & hotspot status	Show network status
8	To show essential data on screen	Show essentials

2.1.5 Desktop Application Development

No.	Requirement	Use Case
1	To connect Amazon Web Service (IoT Core)	Connect AWS IoT Core
2	To pull TDS value data	Pull TDS value
3	To pull water temperature data	Pull temperature value
4	To calculate water quality data	Pull water quality
5	To show TDS value	Show TDS value (Desktop)
6	To show water temperature value	Show temperature value (Desktop)
7	To show calculated water quality	Show water quality (Desktop)

2.2 Data Requirements

Critical data are listed int the table below with their explanation of why we need them and their specific requirements, if there are any.

Variable Name	Explanation and requirements
tdsValue	Data of the total dissolved solid value, the unit will be PPM (parts per million). It is required to be calculated by using the analog output of the TDS sensor module.
tempValue	Data of the water temperature, the unit will be degrees Celsius. It is required to be calculated by using the digital output of the temperature sensor.
wifiSSID	Data of an in-range Wi-Fi SSID of a network that has a connection to the internet. Data type is a set of characters. To be able to use AWS functionalities, this data must be set by the user by using configuration page of the device by connecting to the Wi-Fi hotspot that is generated by our device.
wifiPassword	Data of an in-range Wi-Fi password of a network that has a connection to the internet. Data type is a set of characters. To be able to use AWS functionalities, this data must be set by the user by using configuration page of the device by connecting to the Wi-Fi hotspot that is generated by our device.
hotspotSSID	Data of the Wi-Fi hotspot SSID. That network is generated by our device. Data type is a set of characters. This data cannot be changed since it will be including the unique ID/serial number of the micro controller unit.
hotspotPassword	Data of the Wi-Fi hotspot password. That network is generated by our device. Data type is a set of characters. This data will come with a default value. While configuring the other Wi-Fi options, also this data may be changed.
deviceUniqueID	Data of the unique ID or the serial number of the MCU. This data is permanent and cannot be changed, on default. It is written on the chip in factory. We will use it for the unique Wi-Fi hotspot SSID.
currentScreen	Data of the screen state. It will have different enumerations for different machine states. We will use it in the screen controller task most critically.
softwareVersion	Float. Data of the current version of the software. We will use it while comparing our software version with the latest software version that is on the cloud AWS S3 server.
avarageVoltage	Float. This is a parameter that we will use while calculating analog inputs such as TDS and temperature. In our

	system we can see that it will be somewhere close to 3.20 Volts.
tdsBuffer[]	Set of floats. This buffer is for collecting the most recently collected analog data and calculating the average value for a smooth output. We will use it for TDS calculation.
tempBuffer[]	Set of floats. This buffer is for collecting the most recently collected digital data and calculating the average value for a smooth output. We will use it for temperature calculation.
awsCaCert	Set of characters. This is a certificate called “CA (certificate authority)” that is needed to be able to connect AWS servers. We can get that certificate from AWS’s official website or right after generating a device on AWS IoT Core.
awsDeviceCrt	Set of characters. Device certificate is a certificate that is generated for our AWS IoT Core device for one time at the beginning. We need to save that to use that device. If we lose it, we need to generate a new device on AWS IoT Core and delete the previous one.
awsPrivateCrt	Set of characters. Device private key is a key that is very important, it is like a password for our AWS IoT Core device. With that key and our device certificate, anyone can connect to our AWS server and use it for their own purposes. It is also generated for our AWS IoT Core device for one time at the beginning. We need to save that to use that device. If we lose it, we need to generate a new device on AWS IoT Core and delete the previous one.
awsHostAddrS3	Set of characters. It is for directing our OTA update check requests to our AWS S3 server and the JSON that holds the specific information about the software updates.
awsClientID	Set of characters. It is the name of our AWS IoT Core device. We use it while connecting to the AWS server.
awsIotEndpoint	Set of characters. This is a region specific, special endpoint address to our AWS IoT Core services.
machineState	Enumeration. This will hold the state of the machine. The states will be normal, temperature warning, TDS warning etc.

buzzerState	Boolean. This will hold the state of the buzzer. If the buzzer is active, it is true. If the buzzer is inactive, it is false.
currentScreen	Uint8_t. This will work in parallel with machineState variable but there may be extra screens.
wifiConnectionStatus	Boolean. This will be true if the Wi-Fi connection is valid. If not, false.
hotspotConnectionStatus	Boolean. This will be true if the Wi-Fi hotspot server is working. If not, false.
tdsExpectedUpper	Uint16_t. This is the upper limit of user defined TDS scope for getting warnings with buzzer activation.
tdsExpectedLower	Uint16_t. This is the lower limit of user defined TDS scope for getting warnings with buzzer activation.
tempExpectedUpper	Float. This is the upper limit of user defined temperature scope for getting warnings with buzzer activation.
tempExpectedLower	Float. This is the lower limit of user defined temperature scope for getting warnings with buzzer activation.

2.3 Use Case Diagrams

Below you can see the diagnostics use case diagram that has 4 device peripherals in it. TDS sensor and temperature sensor transmit the data to the device. We put the data in buffer and get the average value for each buffer. Then we use these data to find real temperature and TDS values. By using that information, the system decides to activate or deactivate the buzzer.

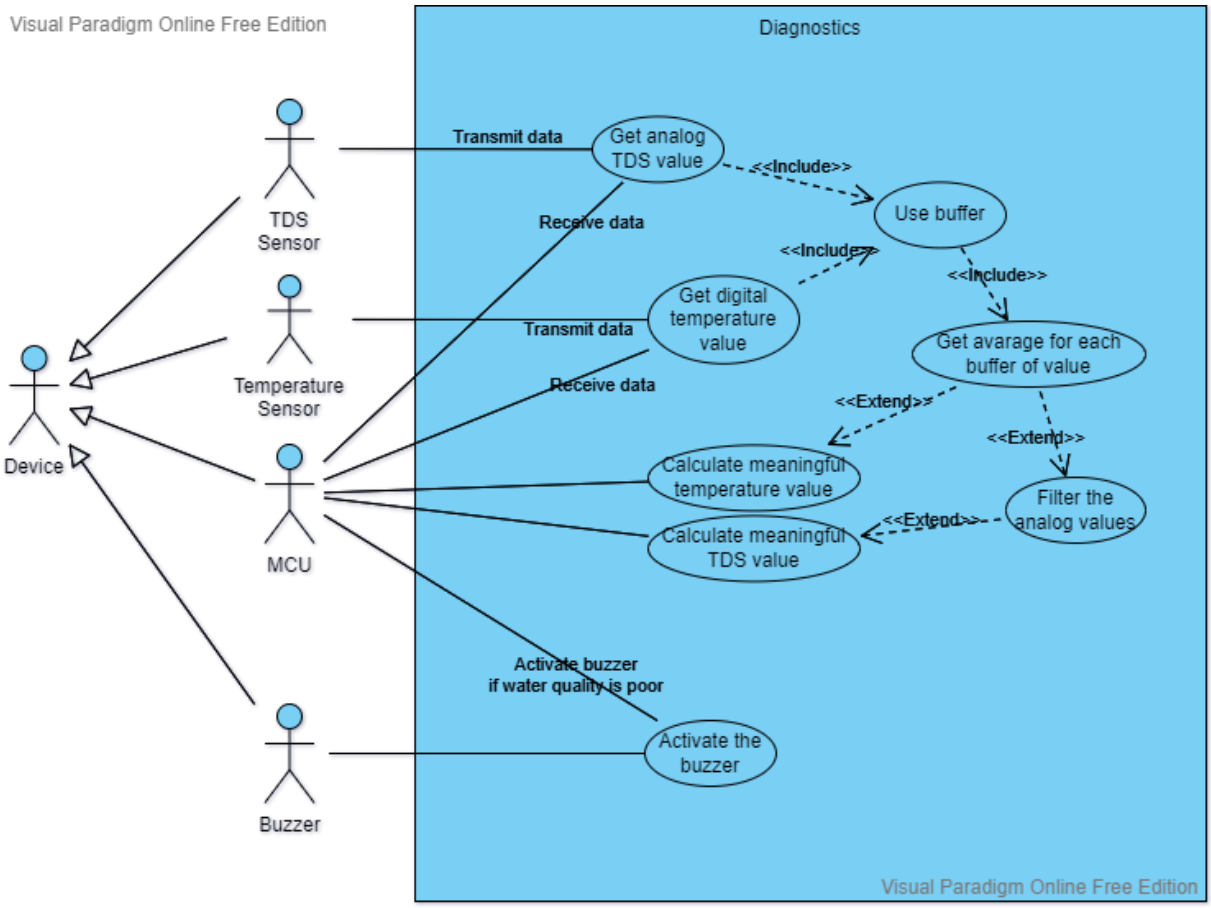

Figure 14 - Signalization Use Case Diagram

Below you can see the Wi-Fi use case diagram with a user and MCU with Wi-Fi module on it. First, the system should create a LAN server therefore there is going to be a Wi-Fi hotspot for the user to connect and then reconfigure the device by saving it on flash memory of MCU. Then the device can also connect to another Wi-Fi network and try connecting to the internet through that network.

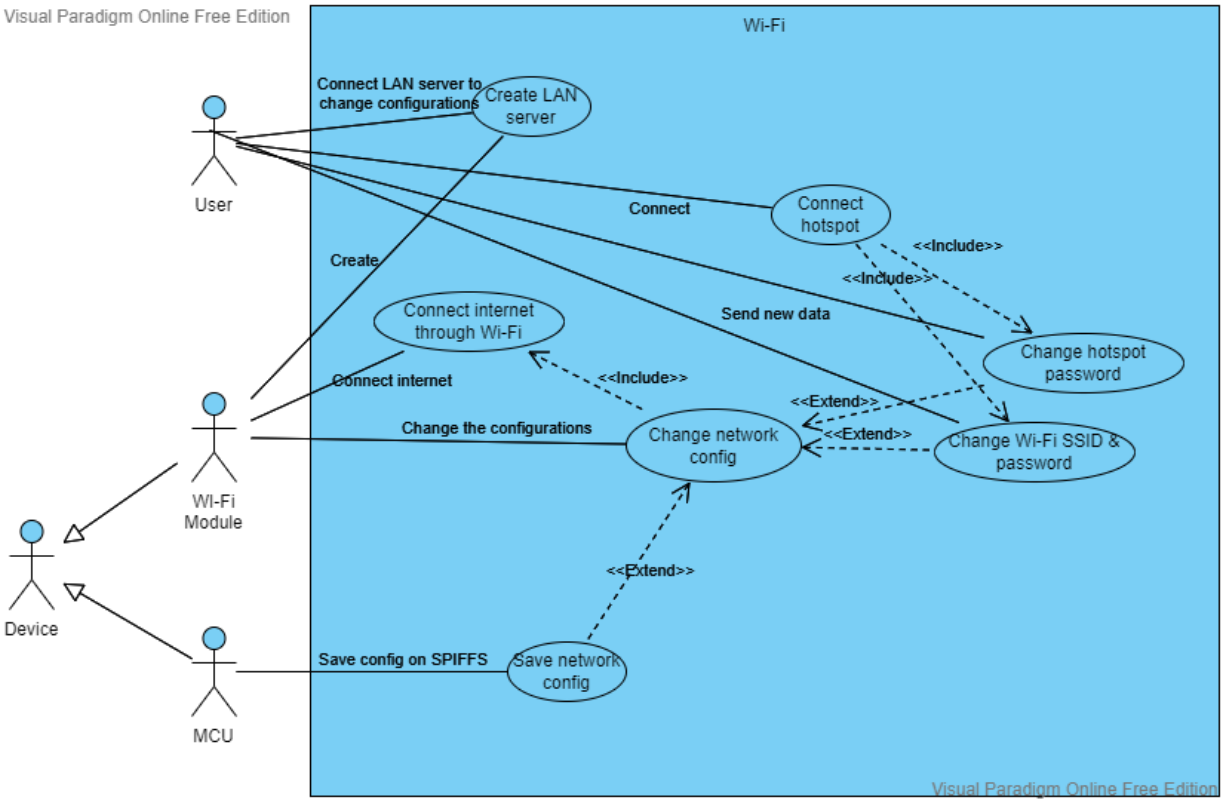

Figure 15 - Wi-Fi Use Case Diagram

Below you can see the AWS use case diagram with a user, device, and creator. First, as a creator, we should register an AWS account and then set up the AWS services that we are going to use and create a thing in IoT Core service of AWS. After creating a thing, we can get device certificate, private key, and Amazon Root certificate. With those certificates we can create different shadows on that thing as a copy of that thing and then publish message over them as diagnostics. Then the user will be able to see those diagnostics over the Aquality desktop app.
We will also be able to upload new software versions to the AWS S3 server and update the new software image link. By using that link the device can get the software update over the air.

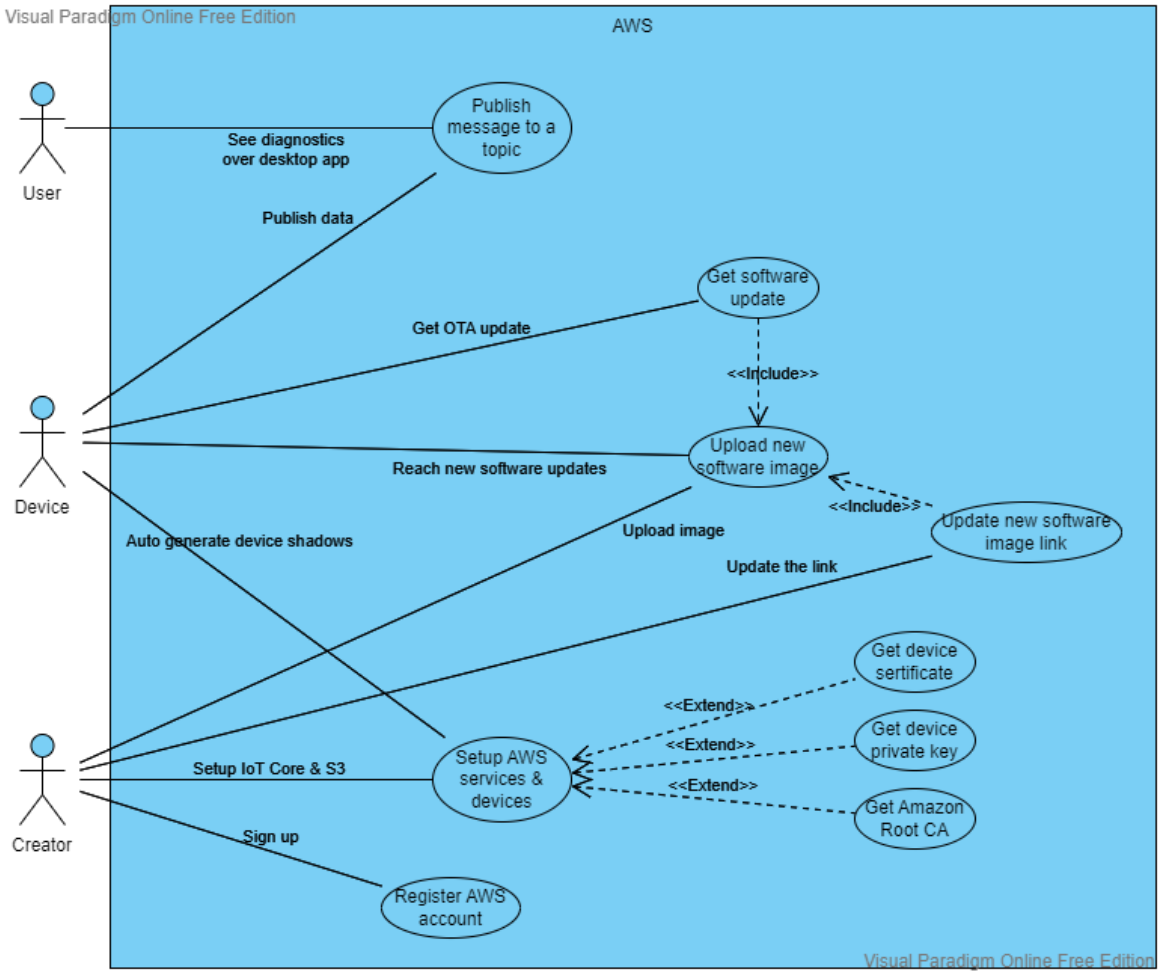

Figure 16 - AWS Use Case Diagram

Below you can see the TFT display user interface use case with a user, TFT display, and a creator. First, we will design the UI elements and use them with additional effects such as animations etc. Then the system and the TFT display will show the essential data such as water quality, network status, TDS, and temperature values. And the user will be able to easily see those information.

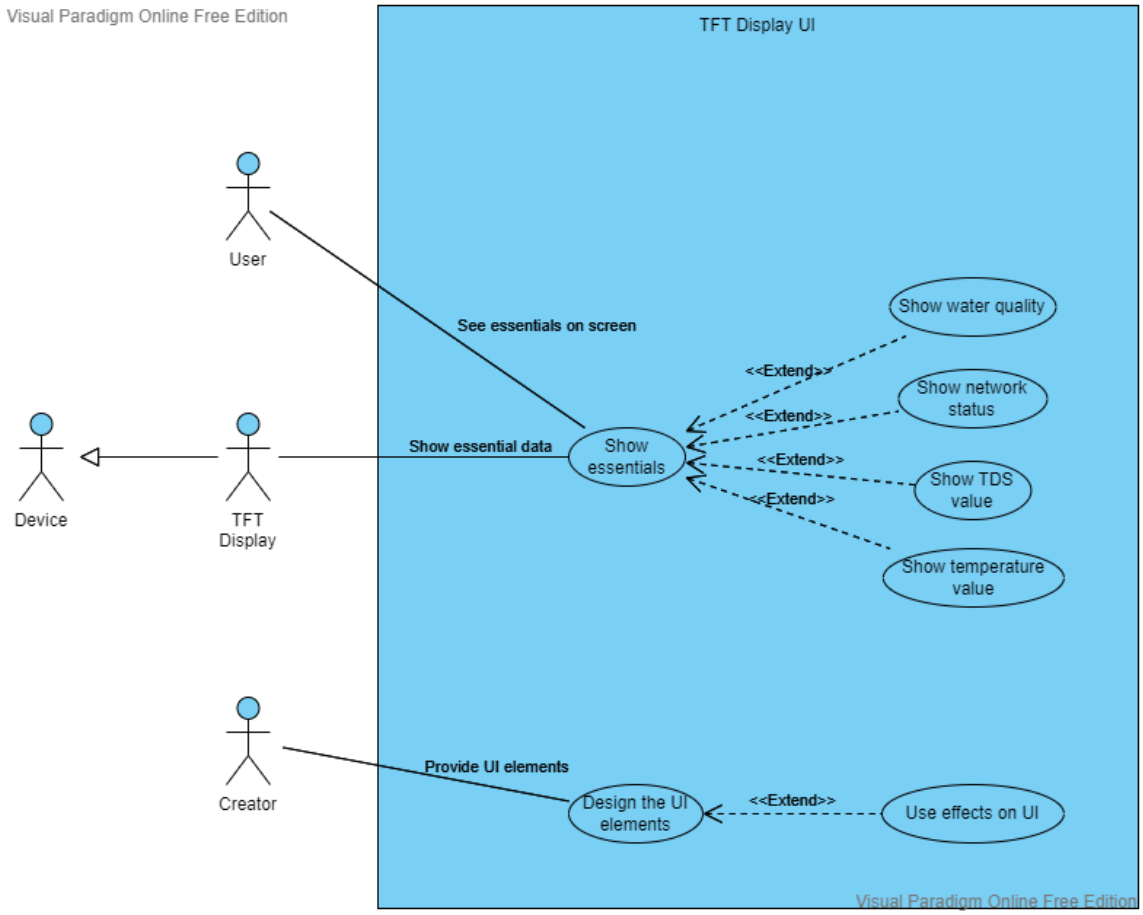

Figure 17 - Display UI Use Case Diagram

2.4 Development & Event Flows

First, we need to develop the UI and screens that we will see in order, by using SquareLine Studio. After we can extract the source code for LVGL library and edit the items that are located on the UI. After that, we can focus on the sensor inputs and buzzer control according to the values. Then we can handle the AWS functionalities such as S3 OTA update and IoT Core diagnostics but before we need to be able to configure the Wi-Fi SSID and passwords. Finally with the development of the desktop monitoring system our product will be done.

The product will refresh the screen by sending at least 15-20 fps to the TFT screen. At the same time, we will be getting input values from TDS and temperature sensors and calculating their meaningful values. According to those values, the screen items will be changed. If the water quality is out of accepted scope by the user -as it will be set while configuring the device when connected to the LAN server, we will activate the buzzer for an increasing amount of time once in an hour while the water quality is getting worse.
The device will have its own LAN server. That server will handle the configuration of the Wi-Fi and Hotspot settings, as well as the water quality scope. Then it can save the configurations on its flash.
While also doing them, the device will also try to connect to the AWS services and check if there are software updates first, then it will start to share diagnostics.

Finally, the desktop application will connect to AWS services, then it will monitor the device diagnostics that the user has.

2.5 Non-functional Requirements

No.	Requirement
1	To make the system durable/reliable by testing and validating the device for each possible scenario
2	To keep the web communication secure by using HTTPS if possible
3	To make the system fast by choosing the right components and developing a robust software by using RTOS features correctly.
4	To create the UI for 128x160 resolution screen
5	To have a stable internet connection
6	To make sure that the device will be located away from the water itself

3. References

- Davis et al., "Identifying and measuring quality in a software requirements specification," [1993] Proceedings First International Software Metrics Symposium, 1993, pp. 141-152, DOI: 10.1109/METRIC.1993.263792.

- "IEEE Guide for Software Requirements Specifications," in IEEE Std 830-1984, vol., no., pp.1-26, 10 Feb. 1984, DOI: 10.1109/IEEESTD.1984.119205.

COMP4910 Senior Design Project 1, Fall 2022
Advisor: Hüseyin Hışıl

AQUALITY: Smart Water Sensor Platform
Design Specification Document

18.12.2022
Revision 1.3

19070001008, Doğu Alpay
18070001033, Mert Sadık Sezigen
17070001013, Berkin Akbıyık

Revision History

Revision	Date	Explanation
1.0	08/12/2022	Introduction and system context diagram added.
1.1	12/12/2022	System design explained. Hardware layer and schematics added.
1.2	15/12/2022	Software layer and tasks are explained.
1.3	18/12/2022	Test designs are added and explained in detail.

Table of Contents

Introduction

With Aquality, we are trying to help people who care about their water quality for any kind of purpose. We came up with an idea of calculating the overall quality of water by using two parameters such as total dissolved solid in the water and temperature of the water. The device is also going to have a warning system with a buzzer on it. It will ring when the water quality is dropped below the assigned level. The device will be transmitting Wi-Fi hotspot signals, connecting to a Wi-Fi network and sending the diagnostics to Amazon Web Services. By connecting to Wi-Fi hotspot, simply going to an IP address from the web browser will let the user change Wi-Fi connection preferences such as network ID and password. If the system can connect to the network correctly according to those preferences, it will start sending data simultaneously. Aquality will also have a small display on it and it will show hotspot status, internet connection status, water quality, water temperature and total dissolved solid in the water. The internet connection and hotspot icons will appear on the permanent top bar, the other things will appear on the screen one by one.
In addition Aquality will have its own diagnostic tracing application. It will simply show diagnostics from all the user's devices with decent internet connection.

1. System Design

1. We will have 2 cloud services that we will connect to, and these are S3 for file server and IoT Core for cloud communications.
   We will be able to upload a new system image to the S3 server and update the software on each device that has an internet connection.
   On the IoT Core service, we will have a thing device that is going to be generated for Aquality specifically. This device has a specific topic path in our personal server that includes a shadow name in it. If we just send a message to a new and different shadow name, it just creates a new shadow like below.
   - The first device: "$aws/things/AQUALITY/shadow/name/TestDevice"
   - New generated device: "$aws/things/AQUALITY /shadow/name/NewDevice"

   Instead of these basic names, we will automatically choose this name by using the unique ID of the chip. Each device at the first time of connection to the internet, will generate a shadow for itself automatically.
   After that we can simply communicate through those shadow paths. We can send & receive live data.

2. The diagnostic data will be collected in specific protocols.
    - The TDS sensor module will be sending us analog values with 3.3 Volts reference. We will read sample analog values into a buffer with at least 50 slots. Then we will get the average 16-bit unsigned integer value.
    - The temperature sensor DS18B20 will be sending the values in Maxim's exclusive 1-Wire bus protocol that implements bus communication using one control signal. We will also use a buffer for it, so it stays stable.

3. We will control the buzzer by simply applying high or low voltage to GPIO3 pin. If you continue reading, you can see the schematics and pinout for more details.

4. We will be refreshing the TFT screen for at least 20 frames per second. Lower is not acceptable.

- Below, you can see the context diagram of the system.

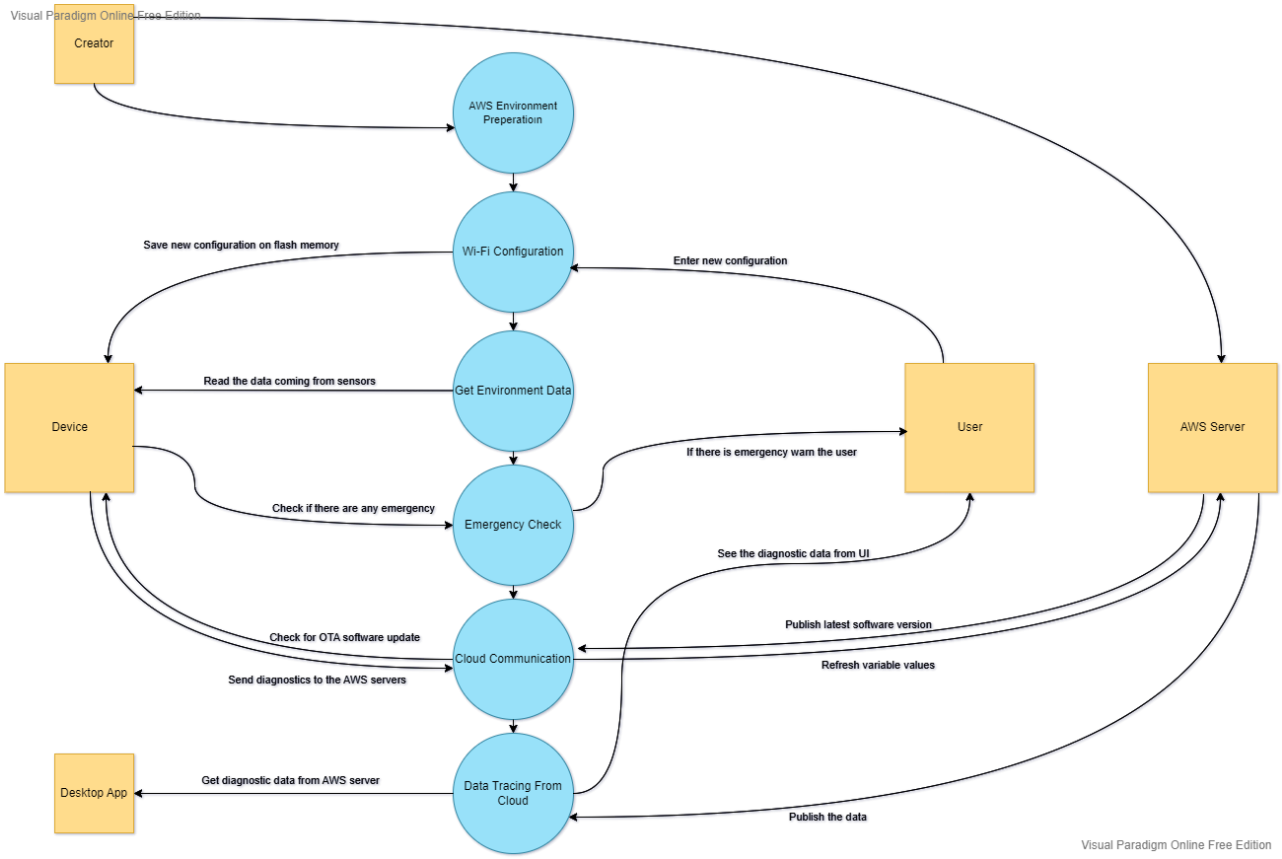

Figure 18 - System Context Diagram

2.1 Hardware Subsystem

On the hardware layer, we have two environment inputs from the sensors such as total dissolved solid and temperature of water. We have a buzzer and a display screen as environment outputs. Which are all controlled by our micro controller unit ESP32-C3. You can see the schematic for Aquality hardware layer below.

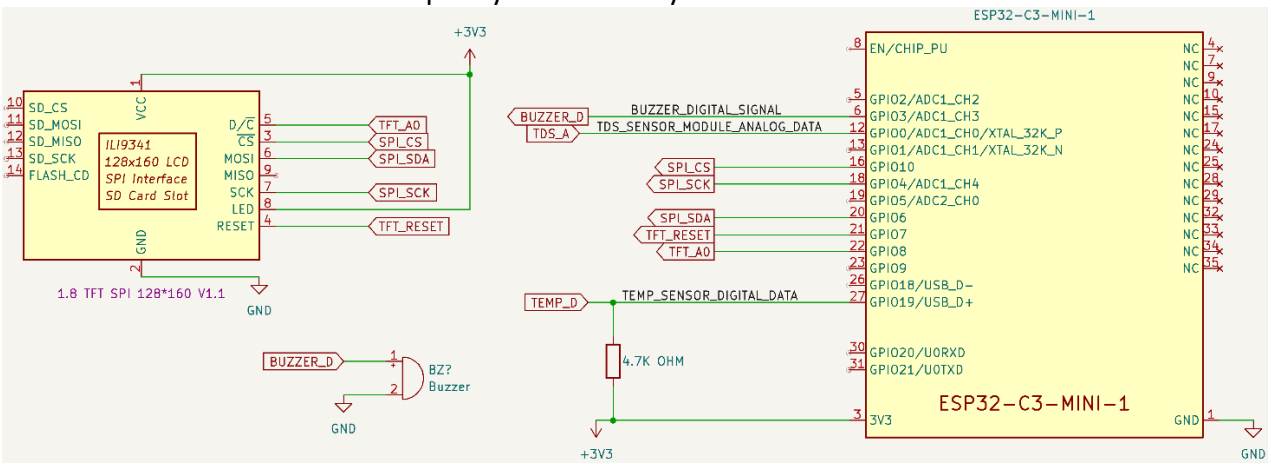

Figure 19 - Hardware Layer Schematic

Since we are using the development board of mentioned ESP32 micro controller unit, we didn't use any kind of resistor or other elements on the schematics -except the one between TEMP_D and +3V3- because the appropriate elements are used on the development board as we need. Therefore, we will only connect the pins according to the schematic on the figure.

In future, if we want to implement the hardware on another custom PCB design, we will need other elements on the schematics.

2.2 Software Subsystem

2.2.1 Embedded Software Subsystem

On the software layer we will be working on an embedded environment. We were able to code without an operating system, but this would be unprofessional. Therefore, we will be developing Aquality on a lightweight operating system, FreeRTOS. FreeRTOS is a real-time operating system with lower amount of complexity but also low capabilities compared to more complex operating systems such as Linux. Since we will be focusing on only one application without any kind of additional work to do, FreeRTOS will be able to schedule and process our limited number of tasks.

Our application's tasks are;

- Machine state controller
- Temperature calculator
- TDS calculator
- Buzzer controller
- Screen controller
- Wi-Fi controller
- AWS controller
- Software update controller

Below you can see the task structure more clearly.

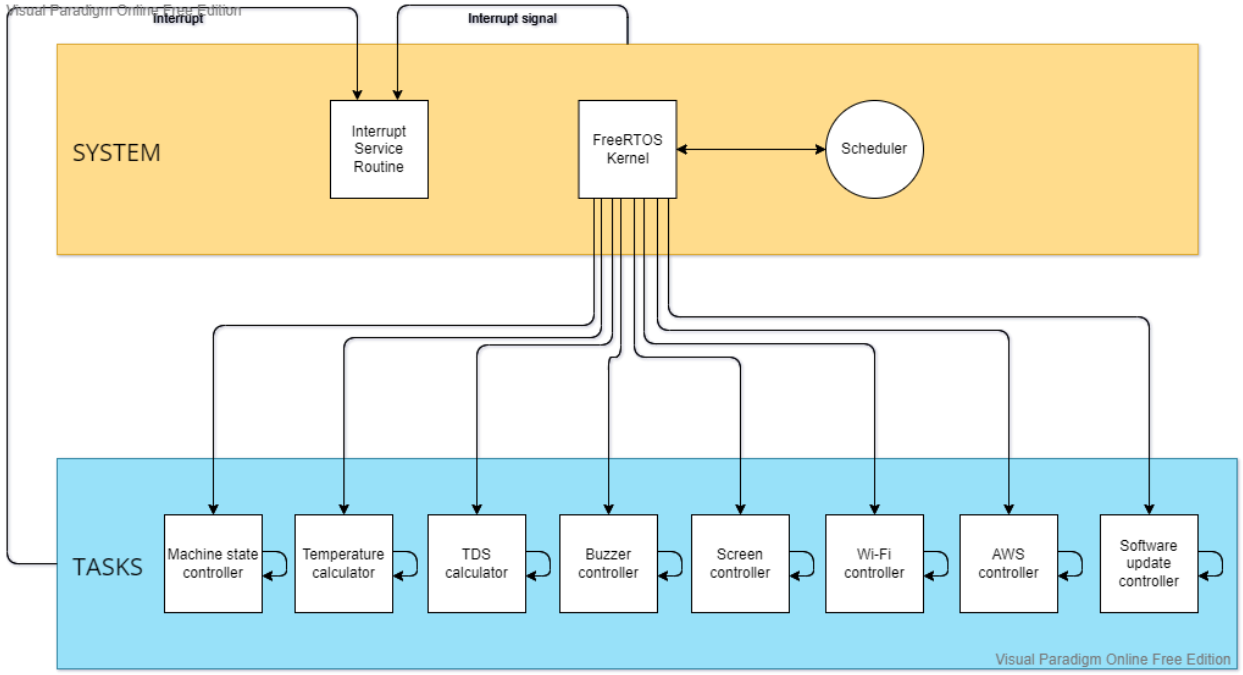

Figure 20 - Software Task Structure

You can find detailed explanations of the operating system tasks below.

Task Name:	Machine state controller
Summary:	This task checks critically important global variables and decides the state of the machine accordingly.
Basic Flow:	1. If the temperature warning flag is set, change the machine state accordingly. 2. If the TDS warning flag is set, change machine state accordingly.
Alternative Flows:	none
Preconditions:	Temperature value calculated. TDS value calculated.
Postconditions:	Machine state set.
Important Variables:	machineState

Task Name:	Temperature calculator
Summary:	This task calculates the water temperature in real time while also reading digital data that is coming from the temperature sensor.
Basic Flow:	1. Read digital data. 2. Fill into the buffer. 3. If the buffer is full, get the average value and write on the temperature variable. 4. If the temperature value is out of expected space, set a warning flag.
Alternative Flows:	If the temperature value is not out of the expected space, clear the warning flag.
Preconditions:	avarageVoltage is given.
Postconditions:	Temperature value calculated.
Important Variables:	tempValue tempBuffer avarageVoltage

Task Name:	TDS calculator
Summary:	This task calculates the TDS value in real time while also reading analog data that is coming from the TDS sensor module.
Basic Flow:	1. Read analog data with ADC. 2. Fill into the buffer. 3. If the buffer is full, get the average value and calculate the real TDS value by also using temperature value and write on the TDS variable. 4. If the TDS value is out of the expected space, set a warning flag.
Alternative Flows:	If the TDS value is not out of the expected space, clear the warning flag.
Preconditions:	Temperature value calculated. avarageVoltage is given.
Postconditions:	TDS value calculated.

Important Variables:	tdsValue tdsBuffer tempValue avarageVoltage

Task Name:	Buzzer controller
Summary:	This task checks the water quality by using environmental data and if needed activates the buzzer to warn the user.
Basic Flow:	1. Check warning flags and activate/deactivate the buzzer as needed. 2. Change the buzzer state variable accordingly.
Alternative Flows:	• If a temperature value warning flag is set, activate the buzzer. If not, deactivate. • If a TDS value warning flag is set, activate the buzzer. If not, deactivate.
Preconditions:	TDS value calculated. Temperature value calculated.
Postconditions:	none
Important Variables:	tdsValue tempValue buzzerState

Task Name:	Screen controller
Summary:	This task refreshes the screen frames according to the machine state by using appropriate data for that specific machine state.
Basic Flow:	1. Check the screen timer. 2. If the timer is up, change the screen state. 3. If TDS state, show the TDS screen. 4. Else if temperature state, show the temperature screen. 5. Show the Wi-Fi and Wi-Fi hotspot status at the top bar.
Alternative Flows:	• If the machine state is set for temperature or TDS warning, warn the user on the screen accordingly.
Preconditions:	TDS value calculated. Temperature value calculated. Machine state set.
Postconditions:	Screen frame refreshed.
Important Variables:	currentScreen tdsValue tempValue buzzerState wifiConnectionStatus hotspotConnectionStatus

Task Name:	Wi-Fi controller
Summary:	This task controls the Wi-Fi connections and makes sure that there are no connection issues. Checks if the Wi-Fi hotspot, Wi-Fi connection with internet and LAN server for configuration work as expected.
Basic Flow:	1. Check Wi-Fi hotspot and LAN server status and change the variable accordingly. 2. If not working properly reset Wi-Fi hotspot. 3. Check Wi-Fi connection status and change the variable accordingly. 4. If not connected and haven't tried to reconnect more than 10 times in last one hour, reconnect. 5. If connected to Wi-Fi but no internet connection, reconnect.
Alternative Flows:	none
Preconditions:	LAN server page HTML structure variables are given. Device configuration is done by user.
Postconditions:	Device configuration is done by user.
Important Variables:	wifiConnectionStatus hotspotConnectionStatus wifiSSID wifiPassword hotspotSSID hotspotPassword tdsExpectedUpper tdsExpectedLower tempExpectedUpper tempExpectedLower

Task Name:	AWS controller
Summary:	This task controls the AWS connections and makes sure that there are no connection issues.
Basic Flow:	If internet connection is valid and not connected to AWS, connect to AWS. If connected to AWS, check for software updates. If a software update is valid, set a warning flag. If connected to AWS, send diagnostic data to IoT Core thing shadow.
Alternative Flows:	none
Preconditions:	wifiConnectionStatus is connected. Device configuration is done by user. TDS value calculated. Temperature value calculated.
Postconditions:	Software update is checked. Diagnostic is published.
Important Variables:	awsCaCert awsDeviceCrt awsPrivateCrt awsHostAddrS3 awsClientID awsIotEndpoint softwareVersion deviceUniqueID tempValue tdsValue

Task Name:	Software update controller
Summary:	This task updates the software if there is a newer version of software package that is on the AWS S3 server.
Basic Flow:	1. Check the software update flag. 2. If it is set, get the image link, and start update. 3. Restart the device.
Alternative Flows:	none
Preconditions:	Software update is checked.
Postconditions:	Software update is done.
Important Variables:	awsHostAddrS3 softwareVersion awsCaCert awsDeviceCrt awsPrivateCrt

FreeRTOS task states may need to be changed according to the situation. Therefore, we need to consider the task state machine diagram of FreeRTOS. Below you can see the original image that is shown in FreeRTOS documentation.

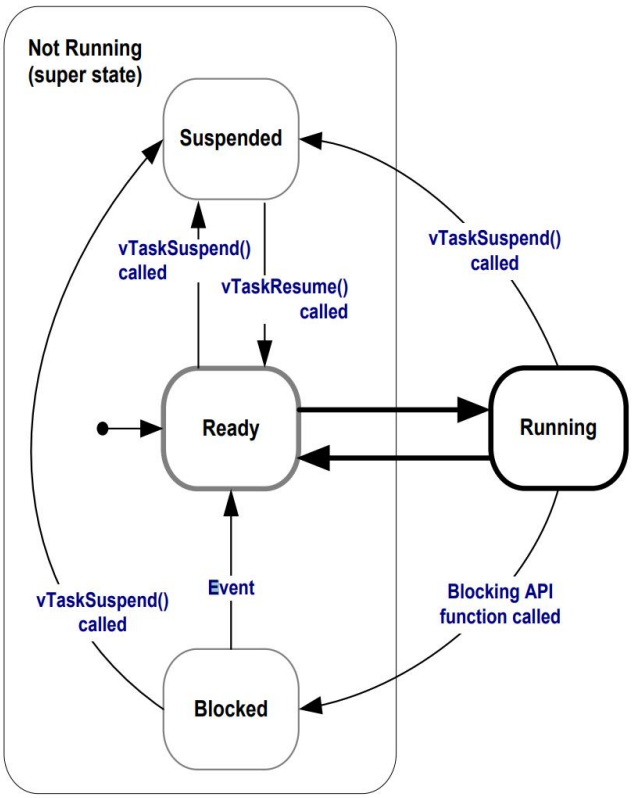

Figure 21 - Task State Machine of FreeRTOS

2.2.2 Desktop Application Software Subsystem

Desktop application will be a simple one-page application. It will have a table on the screen and some filtering & sorting options. Since we will develop that application with Qt environment, we will use PyQt5 framework. Therefore, we will be using Python while developing the desktop application. It will simply refresh the live data for each device that is coming from the AWS IoT Core server.

5. System Test Design

Tests are the most important part of the development part of a project. We will be testing our product in a customized way since this project includes different layers of hardware communication. There will be environmental & hardware tests and software tests.

- In environmental tests we will test our components to see if they are durable and efficient enough to meet the hardware requirements.
- In hardware tests we will be testing our voltage and currents at specific points on the schematic.
- In software tests we will be testing each use case of our software requirements.

In the tables we will define the tests and the expected results. The results must be observed at 100% ratio with at least 1000 amount of test cycles for software tests and at least 10 amount of test cycles for hardware & environmental tests. If the test cycle amount is given in the box, than the above amounts are not important for that specific test.

3.1 System Hardware & Environmental Test Design

Test	Expected Result
Leave the sensor probes in 50cm depth of a water tank for one day.	The sensors probes shouldn't be worn out, broken or took water inside. Should be working properly and with the same precision.
Leave the system open for one day.	Each component on the system should be working fine. Any kind of screen, buzzer, sensor, or development board issue is not tolerable.
Check the peripherals' power consumption.	Peripherals must meet the power consumption equality with datasheet of itself.
Check the voltage values of each power line on the schematic.	Every voltage measurement should meet the requirement of the specific peripheral.
Use different power supply for better check on the powerline stability.	The voltage measurements and power consumptions must match with the above tests. There should not be any difference on the system stability.
Screen test with random animations and LVGL functionalities for 1 minute.	The screen should not flicker and freeze. The fps should appear on the screen and there should be an average of bigger than 20 frames per second. The colors should look fine. There shouldn't be any teleportation or lose on the UI objects.
Compare TDS value with a certified TDS sensor device.	The TDS values should match. -+5 PPM precision is expected.
Compare temperature value with certified waterproof temperature sensor device.	The temperature values should match. -+0.5 degrees Celsius precision is expected.
Check the buzzer activation for 1 minute.	The buzzer sound should be the same. The sound should not stutter or freeze.

3.2 System Software Test Design

Test	Expected Result
Change Wi-Fi SSID & password and try to connect to the Wi-Fi with new SSID & password. (10 times)	The device should automatically connect to the Wi-Fi with given new SSID & password.
Change Wi-Fi hotspot password and try to connect to hotspot with new password. (10 times)	We should be able to connect to the device's Wi-Fi hotspot with the new password. When we try to access the configuration page should appear correctly.
Connect to AWS	Device should automatically connect to the AWS server if the configuration is done, and an internet connection is valid.
Connect to AWS IoT Core	Device should automatically connect to the AWS IoT Core server if the configuration is done, and an internet connection is valid.
Connect to AWS S3	Device should automatically connect to the AWS S3 server if the configuration is done, and an internet connection is valid.
Get software update (10 times)	The device should get the updates and reset the device in 1 minute if there are any new version of the software.
Change screen state and watch the transitions with the sensor values.	Every object of UI should appear correctly, and the functions of the screen should keep working.
Publish diagnostic data on 10 different shadows from the lower to the maximum values. (10 times)	All scope of the data should appear on the desktop application dashboard of Aquality one by one.
Listen to the published data on the above task. (10 times)	All scope of the data should appear on the desktop application dashboard of Aquality one by one.
Try to enter invalid value to the configuration variables. (10 times)	The system should not accept invalid scope of data. There should be a filter system for configuration inputs.
Watch the FreeRTOS tasks with Segger SystemView tool. (10 times)	Every task should be running as expected. There shouldn't be any task that is not working or working in an unexpected way. System memory and disk usage should not be high at any time. The context switching operations must be flawless.

6. References

- Aquality RSD Document v2.3
- https://www.freertos.org/Documentation/RTOS_book.html