

Designing an IoT-Cloud Solution for Precision Aquaculture

BY

LIEW WEI ZHENG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS) INFORMATION SYSTEMS

ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2023

REPORT STATUS DECLARATION FORM

Title: Designing an IoT-Cloud Solution for Precision Aquaculture

Academic Session: JUNE 2023

I LIEW WEI ZHENG

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

2, JALAN BUNGA TANJUNG

.TAMAN SETIA , 43000, _____

KAJANG, SELANGOR _____

Ts. Dr. Cheng Wai Khuen

Supervisor's name

Date: 13/09/2023

Date: 14/9/2023

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN

Date: 13/09/2023

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that **LIEW WEI ZHENG** (ID No: **20ACB05319**) has completed this final year project/ dissertation/ thesis* entitled “*Designing an IoT-Cloud Solution for Precision Aquaculture*” under the supervision of _____ Ts Dr CHENG WAI KHUEN _____ (Supervisor) from Faculty of Information And Communication Technology.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



(LIEW WEI ZHENG)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled “**Designing an IoT-Cloud Solution for Precision Aquaculture**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : LIEW WEI ZHENG

Date : 13 SEPTEMBER 2023

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr Cheng Wai Khuen who has given me this bright opportunity to engage in designing an IoT- Cloud Solution for precision aquaculture. It is my first step to establish a career in cloud computing field. Greatly appreciate the opportunity.

Finally, I'm feeling grateful to anyone providing support during this project. Without them, it is hard to go through the struggle during this period.

ABSTRACT

In recent years, the demand of aquaculture product is increasing due to the growing population of the world and the depletion of wild fish population. To address the depleting fish stocks caused by overfishing and environment pollution, aquaculture will become an alternate source to supply fish stocks to fulfil the demand. Aquaculture has been evolved throughout times by utilizing different technology to improve efficiency and increase production. But, even with the implementation of technology, aquaculture industry is still vulnerable to factor such as labour shortage and inaccurate data analysis. In this case, water quality needs to be monitor periodically because it plays an important role in determine the health and growth of fish. Measuring water quality involve fish farmer collect sample from ponds and pass them to an examiner to perform water testing. The accuracy of the analyse result are determine by the experience of the examiner and prone to human error. Thus, this project designs an IoT-Cloud Solution for Precision Aquaculture in predicting water quality to improve the efficiency and effectiveness of aquaculture operations to increase the overall production. In this paper, an aquaculture farm will be referred as an edge computing environment consisting of IoT sensors and devices used to obtain water parameters from the farm, processed them and store them locally as well as send to cloud for other uses. Sensors of various purpose are used to collect real time water parameter so that the data can be transmit to cloud for analysis via edge device using MQTT protocol. Edge computing can be used to process data locally, reducing the bandwidth requirements for transmitting large amounts of data to the cloud and also reduces the latency in receiving results from the analysis from cloud. While the cloud can store the data as a backup in unstructured format and later serves as data for visualise graphically on a dashboard. The federated learning framework also involves, employing a client-server architecture. In this setup, the edge environment functions as the client, which trains a local machine learning model using local dataset. Meanwhile, the server aggregates the model weights from multiple clients to create a more powerful, global model without accessing the individual client's data. The trained model from the server is then redistributed back to the clients. This process allows clients to have an improved model while ensuring the privacy of their data remains intact.

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	xii
LIST OF TABLES	xvi
LIST OF SYMBOLS	xvii
LIST OF ABBREVIATIONS	xviii
CHAPTER 1 PROJECT BACKGROUND	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Motivation	4
1.4 Impact, significance and contribution	4
1.5 Report Organization	6
CHAPTER 2 LITERATURE REVIEW	7
2.1 Previous Works	7
2.1.1 The use of Information Technology in Aquaculture Management	7
2.1.2 Water Quality Monitoring and Control for Aquaculture Based on Wireless Sensor Networks	10
2.1.3 IoT Based Smart Fish Farming Aquaculture Monitoring System	12
2.1.4 Application of machine vision systems in aquaculture with emphasis on fish	14
2.1.5 Toward Precision Aquaculture: A high performance,	16

cost effective IoT approach	
2.1.6 Precision Aquaculture by Fearghal O’Donncha and Jon Grant	18
2.1.7 Recent Advancements in Deep Learning Frameworks For Precision Fish Farming Opportunities, Challenges, and Application	20
2.2 Chronological trend of reviewed system	22
2.3 Comparison between the previous aquaculture framework and proposed solution	24
2.4 Comparison between previous machine learning method and machine learning in proposed solution	25
2.5 Proposed Solutions	25
CHAPTER 3 PROJECT SCOPE AND OBJECTIVES	27
3.1 Project Scope	27
3.2 Project Objectives	28
CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN	30
4.1 System Architecture Diagram	30
4.2 Methodology	32
4.3 Project Workflow	33
4.3.1 Planning Phase	34
4.3.2 Analysis Phase	34
4.3.3 Design Phase	34
4.3.4 Implementation Phase	34
4.3.5 System Prototype	35
4.3.6 Implementation (Final)	35
4.4 Formula of Single-Point pH Calibration	35
4.5 Technologies and Tools Involved	36
4.5.1 Laptop	36
4.5.2 Raspberry Pi 3 Model B+	36
4.5.3 Raspberry Pi T-Cobbler	37
4.5.4 DS18B20 Waterproof Temperature Sensor	38

4.5.5 PH4502C Analog pH Sensor	38
4.5.6 ADS1115 Analog to Digital Converter	39
4.5.7 Raspberry Pi OS Debian Bullseye (64-bit)	40
4.5.8 Amazon Web Services (AWS)	40
4.5.9 AWS IoT Greengrass Core Software	41
4.5.10 MQTT Protocol	41
4.5.11 Python Programming Language	41
4.5.12 JSON	42
4.5.13 Jupyter Notebook	42
4.5.14 Amazon QuickSight	42
4.5.15 gRPC Protocol	43
4.5.16 Flower Federated Learning Library	43
4.5.17 TensorFlow	43
4.6 Hardware Prototype Design	44
4.7 Schematic Diagram	45
4.7.1 T-Cobbler and LEDs	45
4.7.2 pH Reading Module	46
4.7.3 Temperature Reading Module	47
4.8 Project Timeline	48
CHAPTER 5 SYSTEM IMPLEMENTATION	51
5.1 Initial Set-up	51
5.2 Raspberry Pi Setup (Edge Device)	51
5.2.1 Raspberry Pi Setup Command	52
5.3 Setting Up Hardware	54
5.4 pH Sensor Calibration	55
5.4.1 Adjusting the Sensor Voltage Offset	56
5.4.2 Mapping the pH Readings to Output Voltage	57
5.4.3 Calculate the Slope and Intercept	59
5.4.4 Evaluate the pH Readings	59
5.5 Setting Up Edge Environment	60
5.5.1 AWS IoT Greengrass	60
5.5.2 AWS Greengrass Component	62

5.5.2.1	Sensor reading and processing component (com.example.sensorv5)	65
5.5.2.2	Rule-Based Alert Component (com.example.testAlarmv2)	69
5.5.2.3	Data Synchronization Component (com.example.sync)	70
5.5.2.4	Federated Learning Client (com.example.flwrClient2)	72
5.6	Setting Up AWS Cloud	73
5.6.1	AWS IoT Core	73
5.6.2	Amazon SNS	74
5.6.3	Amazon Lambda	75
5.6.4	Amazon DynamoDB	76
5.6.5	Amazon Athena	77
5.6.6	Amazon S3	78
5.6.7	Amazon Elastic Container Service (Amazon ECS)	79
5.6.8	Amazon QuickSight	81
5.7	Federated Learning Module Implementation	82
5.7.1	Scheduling Deployment of Federated Learning Server	83
5.7.2	Setting Up DNS Name	83
CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION		85
6.1	Hardware Deployment	85
6.2	System Testing	87
6.2.1	Greengrass Components Testing	87
6.2.1.1	Sensor Reading and Processing Component Testing (com.example.sensorv5)	88
6.2.1.2	Rules-Based Alert Component Testing (com.example.testAlarmv2)	90
6.2.1.3	Data Synchronization Component Testing (com.example.sync)	90
6.2.2	Greengrass Components Verification Test	91

6.2.3 AWS Cloud Testing	93
6.2.4 Federated Learning Module Testing	94
6.2.5 Federated Learning Module Verification Test	96
6.3 Data Visualization	97
6.3.1 High Level Dashboard	98
6.3.2 Individual Level Dashboard	100
CHAPTER 7 CONCLUSION	102
REFERENCES	104
APPENDIX	A-1
Appendix A	A-1
Appendix B	A-2
Appendix C	A-3
WEEKLY LOG	112
POSTER	118
PLAGIARISM CHECK RESULT	119
FYP2 CHECKLIST	121

LIST OF FIGURES

Figure Number	Title	Page
Figure 1.1	The statistic of aquaculture sector of Malaysia in the year 2018 and 2019	1
Figure 1.2	Aquaculture production in Malaysia from year 2015 to 2020	1
Figure 1.3	The emerging of highly contagious viral disease, Tilapia Lake Virus (TiLV) and its impact on Tilapia farming industry	2
Figure 2.1	Block diagram of the Wireless Sensor Network	10
Figure 2.2	Block diagram of the smart aquaculture system	12
Figure 2.3	Simplified network architecture	16
Figure 2.4	Four pillars of precision aquaculture framework	18
Figure 2.5	The effect of water quality on harvest result	20
Figure 4.1	AWS Cloud Architecture Diagram	30
Figure 4.2	Description of Steps in Architecture Diagram	30
Figure 4.3	Prototyping Development Methodology	32
Figure 4.4	Raspberry Pi 3 Model B+	36
Figure 4.5	Raspberry Pi T-Cobbler	37
Figure 4.6	DS18B20 Waterproof Temperature Sensor	38
Figure 4.7	PH4502C Analog pH Sensor	38
Figure 4.8	ADS1115 Analog to Digital Converter	39
Figure 4.9	Graphical Design of IoT Device	44
Figure 4.10	Schematic Diagram of T-Cobbler and LEDs	45
Figure 4.11	Schematic Diagram of pH Reading Module	46
Figure 4.12	Schematic Diagram of Temperature Reading Module	47
Figure 4.13	Gantt Chart (1)	48
Figure 4.14	Gantt Chart (2)	49
Figure 4.15	Gantt Chart (3)	50
Figure 5.1	CLI displaying cgroup v2	52

Figure 5.2	CLI displaying content boot configuration file	52
Figure 5.3	CLI displaying updated content boot configuration file	52
Figure 5.4	CLI displaying cgroup v1	52
Figure 5.5	CLI displaying TensorFlow version installed	53
Figure 5.6	Hardware Implementation Based on Breadboard	54
Figure 5.7	IoT Devices Receiving Power	55
Figure 5.8	BNC Connector with Paper Clip Inserted and Alligator Clip Connected to the Outer Metal Casing of BNC Connector	56
Figure 5.9	The Highlighted Area on Reference Electrode is Potentiometer	56
Figure 5.10	The Highlighted Area on Reference Electrode is Potentiometer	57
Figure 5.11	pH 4.01 and pH 7.00 Buffer Powder for Buffer Solution	57
Figure 5.12	pH 4.01 and pH 7.00 Buffer Solution with pH Probe Immerse in It	58
Figure 5.13	The Output Voltage of pH Probe from pH 4.01 and pH 7.00 Buffer Solution (Left to Right)	58
Figure 5.14	The pH reading of pH sensor from pH 4.01 and pH 7.00 Buffer Solution (Left to Right)	59
Figure 5.15	Raspberry Pi Registered as Greengrass Core Device	60
Figure 5.16	Raspberry Pi CLI Indicating Greengrass Core Software are Installed on It	61
Figure 5.17	Greengrass Core Device as a IoT Thing	61
Figure 5.18	Greengrass Core Device inside IoT Thing Group	62
Figure 5.19	Component List of raspberrypi_AquaFarm1	63
Figure 5.20	Deployment for raspberrypi_AquaFarm1	64
Figure 5.21	Raspberry Pi CLI Command to Deploy Custom Component	64
Figure 5.22	Raspberry Pi CLI Command to Display Component List	65
Figure 5.23	A Snap of Artifact for Component com.example.sensorv5	66

Figure 5.24	A Snap of Recipe for Component com.example.sensorv5	66
Figure 5.25	DS18B20 Data Processing Function	66
Figure 5.26	PH4502C Data Processing Function	67
Figure 5.27	Local Database Function	67
Figure 5.28	Temporary Database Function	68
Figure 5.29	AWS IoT Core MQTT Test Client	69
Figure 5.30	A Part of Artifact for Component com.example.testAlarmv2	69
Figure 5.31	A Part of Recipe for Component com.example.testAlarmv2	70
Figure 5.32	A Portion of Artifact for Component com.example.sync	70
Figure 5.33	A Portion of Recipe for Component com.example.sync	71
Figure 5.34	A Part of Artifact for Federated Learning Client Component com.example.flwrClient2	72
Figure 5.35	A Part of Recipe for Federated Learning Client Component com.example.flwrClient2	72
Figure 5.36	IoT Rules for Message Routing	73
Figure 5.37	The Action Indicating Lambda Function in IoT Rule	73
Figure 5.38	Detail of SNS Topic	74
Figure 5.39	Lambda Function intermediateProcessing	75
Figure 5.40	environment1DB Table Configuration	76
Figure 5.41	Details of Athena Data Source dynamo-quicksight	77
Figure 5.42	Table Specified in Associated Databases	77
Figure 5.43	Athena Query Editor Testing	78
Figure 5.44	Bucket List in S3	78
Figure 5.45	Detail of Task Definition Created from Docker Image	79
Figure 5.46	The Federated Learning Server Running as a Task in ECS Cluster	80
Figure 5.47	List of Datasets in QuickSight	81
Figure 5.48	QuickSight Dashboard Developer Interface	81
Figure 5.49	Federated Learning Module from Architecture Diagram	82
Figure 5.50	Schedule Task with Target Task Definition	83
Figure 5.51	Configuration of Network Load Balancer testLB	83

Figure 6.1	The Entire View of Fish Tank with IoT Hardware	85
Figure 6.2	The Hardware Deployment	85
Figure 6.3	DS18B20 and PH4502C Sensor in Water	86
Figure 6.4	Local Occupants	86
Figure 6.5	Green LED Illuminating on IoT Device	88
Figure 6.6	MQTT Test Client Receiving Message from Subscribed Topic	88
Figure 6.7	Content of Main Database (Left) and Temporary Database (Right)	89
Figure 6.8	Email Message from AWS SNS	90
Figure 6.9	MQTT Receiving Message from Synchronize Synchronization Topic	90
Figure 6.10	Data Routing Part from Architecture Diagram	93
Figure 6.11	Items in DynamoDB Table environment1db	93
Figure 6.12	CloudWatch Event Log for Federated Learning Server	94
Figure 6.13	Federated Learning Client Event Log	95
Figure 6.14	Global ML Model Weight Uploaded in S3 Bucket	95
Figure 6.15	Client Model Weight Uploaded in S3 Bucket	95
Figure 6.16	Federated Learning Module from Architecture Diagram	96
Figure 6.17	Data Visualization Module from Architecture Diagram	97
Figure 6.18	Main Page of Dashboard (High Level)	98
Figure 6.19	Individual Tank Detail Page of Dashboard (Individual Level)	100

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Comparison between the previous aquaculture framework and proposed solution	24
Table 2.2	Comparison between previous machine learning method and machine learning in proposed solution	25
Table 4.1	Specifications of laptop	36
Table 6.1	Greengrass Components Verification Testing Table	91
Table 6.2	Federated Learning Steps Verification Test	97

LIST OF SYMBOLS

±	Plus/Minus Sign
°C	Degree Celcius

LIST OF ABBREVIATIONS

<i>IoT</i>	Internet of Things
<i>AI</i>	Artificial Intelligence
<i>IT</i>	Information Technology
<i>DSS</i>	Decision Support System
<i>UI</i>	User Interface
<i>ES</i>	Expert Systems
<i>GIS</i>	Geographical information system
<i>WSN</i>	Wireless Sensor Network
<i>MVS</i>	Machine Vision System
<i>RAD</i>	Rapid Application Development
<i>AWS</i>	Amazon Web Services
<i>ML</i>	Machine Learning
<i>DL</i>	Deep Learning
<i>Pub/Sub</i>	Publish/Subscribe
<i>GPIO</i>	General-Purpose Input/Output
<i>RPC</i>	Remote Procedure Call
<i>BI</i>	Business Intelligence
<i>RNN</i>	Recurrent Neural Network
<i>ADC</i>	Analog to Digital Converter

Chapter 1: Project Background

1.1 Introduction

SEKTOR AKUAKULTUR AQUACULTURE SECTOR	2018		2019		PERUBAHAN Changes (%)	
	Kuantiti (MT) Quantity	Nilai (RM B) Value B	Kuantiti (MT) Quantity	Nilai (RM B) Value B	Kuantiti Quantity	Nilai Value
Air Tawar Freshwater	101,270	0.71	104,602	0.78	3.29%	9.25%
Air Payau / Marin Brackishwater	116,112	2.29	119,069	2.46	2.55%	7.18%
Rumpai Laut Seaweed	174,083	52.22 mil	188,111	65.84 mil	8.06%	26.11%
JUMLAH TOTAL	391,465	3.06	411,782	3.30	5.19%	8.10%

Figure 1.1: The statistic of aquaculture sector of Malaysia in the year 2018 and 2019

[1]

Aquaculture industry in Malaysia comprise of different sectors, from edible to non-food commodities. Based on figure 1.1, the edible food fish are break into freshwater, brackish water/marine and seaweed. While the non-food type of fish are ornamental fish and aquatic plants. Among the freshwater fish, Tilapia, Catfish and Prawn are the popular freshwater species for local fish farm. For brackish water fish, Grouper, Sea bass, Tiger Prawn, Shellfish and Seaweed provides the most commercial value to the market.

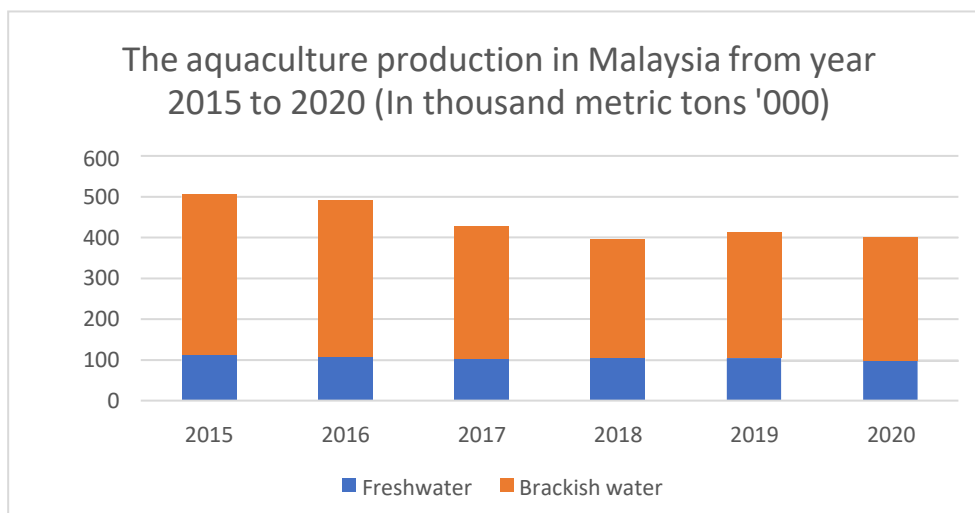


Figure 1.2 Aquaculture production in Malaysia from year 2015 to 2020

CHAPTER 1 PROJECT BACKGROUND

Figure 1.2 displays the aquaculture production in Malaysia from year 2015 to 2020. The output of aquaculture industry in Malaysia are going downhill from 2015 to 2018 and remain unstable throughout the year 2018 to 2020 with no significant changes. There are several factors that contributed toward the output of local aquaculture industry such as diseases, water quality, feeding rate and frequency, number of culturist and others.

1.2 Problem Statement

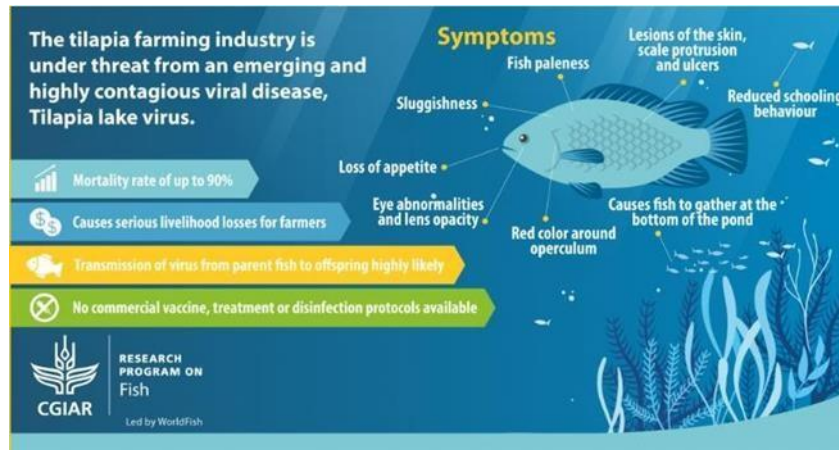


Figure 1.3: The emerging of highly contagious viral disease, Tilapia Lake Virus (TiLV) and its impact on Tilapia farming industry [2]

Cultivating of aquatic organisms, especially fish can be happened in an artificially built controlled environment like ponds and fish tanks or in a more natural environment inshore or offshore waterbody using cages. All these methods usually involve large volume of fishes cramped in a relatively small space and unusual environment. Therefore, it poses a significant threat where fish are often stressed and susceptible to any disease causing the local fish populations extremely vulnerable to any infectious diseases [3]. Other factor that can potentially contributes toward weaken immune system of fish are water quality of the aquaculture environment.

According to figure 1.3, the emerging of Tilapia Lake Virus (TiLV) are threatening the world's second most farmed fish and one of the most farmed freshwater fish in Malaysia which could cause significate damage to aquaculture industry worldwide. Being able to diagnose diseases happened in fish population involve daily monitoring of fish behaviour for any sign of sickness [4]. But the process in diagnosing aquatic animals for any clinical sign is close to impossible due to the nature of fish make them hard to visualize closely and inspect their health status. In relative terms, diseases control is more challenging in

CHAPTER 1 PROJECT BACKGROUND

aquaculture compared to the land-based livestock since the environment where fish lives make them hard to observe using the conventional ways [5]. Often, the needed to catch samples of fish for health examination are necessary to determine the health status of the fish population. But fish cannot be cached easily without causing stress on them, and potentially affect the result of the health examination.

Besides, diseases are also prevalent in fish populations in the wild, whether in freshwater or marine environments. However, in the natural environment, water flows freely and dead fish are rapidly consumed by predators, which helps to reduce the risk of disease transmission [6]. As a result, the effects of disease are often less noticeable in the wild compared to aquaculture systems, where water often circulates within a closed system which is also factors that affect immune respond of fish [7]. Moreover, fish farmer often measures water quality by manually collecting water sample in a centralize filtering system or every pond periodically. The accuracy of manual water quality test depends on the skill and experience of the tester.

Furthermore, today's aquaculture industry still highly dependent on manual labouring especially tasks to obtained data about the fish and the environment. From collecting data using visual observation by farmer or data acquisition tools such as water quality test kit, oxygen meter and others to interpret the collected data into the status and condition of fish and environment [8]. The interpretation of data will then assist the farmer in decision making regarding the operation of the farm, human error is inevitable during the processes. The labour-intensive process make aquaculture vulnerable to labour shortage problems. For example, the implementation of Movement Control Order (MCO) by Malaysia government during covid-19 affected the supply of foreign labour into Malaysia [9]. Consequently, without sufficient workforce engaging in aquaculture industry, the fish supply chain will be affected because of the limited production capacity.

CHAPTER 1 PROJECT BACKGROUND

1.3 Motivation

The challenges face in aquaculture industry must be solves to improve the production to address the increasing demand of global market. Fish farms need an efficient way to monitor the health and the environment condition of the fish that will eventually influence the production. The lack of manpower in the market made it hard to find an adequate personal for the relative job position. Moreover, the difficulties in measure water quality made the accuracy unprecise, the collected sample did not represent the current state of the water quality. Other than that, the conventional ways in measure water quality are done periodically and often involve high resources in maintain the water quality. By introducing precision aquaculture, it helps farmer to facilitate and automate operations in their farm based on the collected data from sensor and other automate technology [10]. It also reduced the need for labour, fast detection of fish's health status, real time water quality monitoring and provided more accurate data for forecasting and decision making. To realized it, an IoT-Cloud solution in precision aquaculture is proposed to collect real time data using IoT devices and sensors in edge computing environment. Edge computing process a portion of data locally within the aquaculture environment, in close proximity to the source of the data [11]. This approach differs from relying solely on cloud resources where it requires more cost and network bandwidth to transmit data as well as introduced latency. The cloud is utilized for tasks that require high computing power and served as storage of pre-processed data, particularly for machine learning and data visualization purposes.

1.4 Impact, significance and contribution

This project involves the use of various IoT sensors, devices, and technologies to reduce the need for human presence in aquaculture operations. This is particularly relevant to the current labour shortage in the market. By applying sensors for data acquisition, the mortality rate of fish populations due to stress from manual fish catching for sampling purposes can be greatly reduced, as the weakened immune systems of fish are less susceptible to disease. In addition, the use of IoT sensors enables the collection of more accurate field data in real-time, compared to manual sampling, which often fails to represent the current state of the aquaculture environment. The use of machine learning in data analysis and forecasting also reduces the need for experienced personnel and provides faster results for decision-making by farmers.

CHAPTER 1 PROJECT BACKGROUND

The data collected by sensors are pre-processed locally and used for training local machine learning models in an edge computing environment, which significantly reduces latency by processing data and performing analysis closer to the point where it's generated [12]. This eliminates the need to traverse over a network to a cloud. Since most aquaculture farms are located in remote areas with poor network connectivity in Malaysia, this method provides several benefits, such as lower costs of data transmission and bandwidth consumption, as only the significant data are sent to the cloud. Meanwhile, the cloud provides services to farmers, primarily database and computing power, without the need to operate and maintain infrastructure.

The pre-processed data is stored in a cloud database for backup and other purposes, while locally trained machine learning models are later used for federated learning. Assuming the cloud infrastructure is shared by several unrelated aquaculture farms, the federated learning approach ensures data privacy, as all the machine learning models are trained locally at their respective edge environments without exchanging data with each other [13]. Over the cloud, a centralized machine learning model is updated with the aggregated model from every edge to create a robust and precise model that will be received by every edge again. The idea of federating learning solves the issue of insufficient data to train an accurate and powerful model as the central model benefits from trained model at every edge using their own data.

In summary, the IoT cloud solution for precision aquaculture increases the scalability of aquaculture operations by allowing on-demand cloud services that enable operations to scale up or down easily. At the same time, the overall operation costs are lower because of reduced labour costs and the only payment required is for the rate of cloud service used. This project also increases the efficiency of farm operations through the implementation of IoT devices and machine learning that support data-driven decision-making to boost aquaculture production to meet the growing market demand.

CHAPTER 1 PROJECT BACKGROUND

1.5 Report Organization

This report consists of 6 chapters, the first chapter will be the Introduction of this report where motivation of this project is described achieve and the contribution of it. Chapter 2 is Literature Review, various previous work is reviewed and the features are compared with the proposed system. The previous work is organized into a chronological order to depict the evolution of technology use in aquaculture. In Chapter 3, the project scope and objective are elaborated. Chapter 4 will be the Methodology and System Design, where the methodology and project workflow are planned, and the system design are elaborated. Chapter 5 is the part where System Implementation are report, the realization of architecture diagram are shown in this part. Chapter 6 will be the System Deployment and Evaluation, where the completed implementation is deployed to real tank for testing. Lastly, Chapter 7 is the conclusion of this project with overview of the latest state.

Chapter 2: Literature Review

2.1 Previous Works

2.1.1 The use of Information Technology in Aquaculture Management

Omar F. El-Gayar [14], published a research paper regarding the impacts of advancement in IT field towards aquaculture industry. To address the growing demand of fish protein and the depletion of wild fish population, the adoption of advanced IT in aquaculture industry for better management of facilities are necessary to increase the efficiency and productivity. Besides, due to the nature of the physical, biological and environments of aquaculture, the decision-making process as well as management are fairly complex. Thus, by adapting different IT solution in this industry which will potentially benefits the management of aquaculture facilities are crucial. The solutions being reviewed from various author are namely instrumental and control, computerized data management, artificial intelligence, decision making systems and expert systems, image processing and pattern recognition, and geographical information system.

Instrumentation and process control

Process control helps in automating the control of an aquaculture production process using instrumentation techniques. Instrumentation provides feedback information about the system which the information will then analysed for proper action to be carried out to control the system to get desired result. For instance, the application of instrumentation on environmental monitoring and control ensures the effect of environment parameters including pH value, temperature, oxygen level, salinity and others are in optimum level to avoid the negative effect on aquaculture production [15]. Visual sensors are connected to a computer to stored collected information about the environment factors and later will be analysed using image processing and recognition algorithms. Once the monitored environment did not achieve the prescribed optimal range of environment parameter, alarm will be raised to inform the operator for further actions.

CHAPTER 2 LITERATURE REVIEW

Data Management

Aquaculture involve a series of complex processes from environment optimization to fish status assessment before ready to harvest. During the process, large amount of data needs to be constantly tracked, with concern on the quality and availability of the data to avoid inappropriate decision made cause by the unreliable data. Regard to the needs, computer is most suitable for performing data management tasks such as storing, analysing, retrieving and presenting data with many available “off the shelf” data management software in the market. For instance, a microcomputer-based data management system composing a desktop microcomputer for data storing, database worksheet, pocket computer for field data monitoring, modem and printer to download the collected data remotely to the desktop are proposed [16]. The data of environment and fish are recorded periodically into the system for analysing.

Decision Support Systems

A DSS created a model with UI and database management system for aquaculture operator to increase the effectiveness of decision making process in operating the farm. El-Gayar et al [17], propose a prototype DSS that aid the operator in decision making regarding the planning and development of the farm. The system consists of three components, a model containing the tasks and objectives, a database to store the recorded data, and a dialog component providing UI to display the other two components and handle interaction between user and system. DSS assists operator in decision making on operational and managerial task of the farm based on models and data stored inside the system.

Artificial Intelligence and Expert Systems

AI can be defined as development of tools and techniques that mimic human intelligence in problem solving using computer science, logic and mathematics. While ES also known as knowledge-based systems, is a program with capabilities of AI using knowledge associated with inferential procedures to solve problems in certain problem domain. Nevertheless, AI are just emerging, the application of AI in aquaculture industry is not obvious. An application automates the recirculating aquaculture system using control system connected with an ES

CHAPTER 2 LITERATURE REVIEW

shell are propose by Lee [18] and it integrate the management of the aquaculture operations and filtration system. The author emphasize that the application can achieves real-time monitoring and control in aquaculture system similar to that perform by human.

Image Processing and Pattern Recognition

The use of computer algorithms to process digital images to make it better for further processing is called image processing. Other than that, pattern recognition extracts some features of an image and then the features will be used for recognition by utilizing statistical technique or AI. Patrell, Neufeld and Savage [19] present a pattern recognition system to capture images of fish in marine sea cage environment. The system comprises of three video cameras integrated with computer-based image analysis system without using artificial lightning. Since the system are still in development phase, the early result of the system is rather inaccurate with the ability to count only 5% of the actual size of fish population in the marine sea cage.

Geographical Information Systems

The geographical factor plays an important role in aquaculture, it determined the suitable species to be farm in a specific environment by analysing the scenarios and criteria of the target environment. GIS composing computer hardware and software for the purpose of recording, storing, analysing and present the geographical data which use in the planning phase of aquaculture development to choose the optimum sites for aquaculture activities. The application of GIS in aquaculture are demonstrate by Ross, Mendoza and Beveridge [20] in the selection of suitable site for salmon aquaculture in coastal area of Scotland. The criteria to be evaluated are the water current, water quality, natural shelter, and bathymetry. The result from the evaluation recommends that 6.4% of the total are a suitable for salmon farming.

CHAPTER 2 LITERATURE REVIEW

2.1.2 Water Quality Monitoring and Control for Aquaculture Based on Wireless Sensor Networks

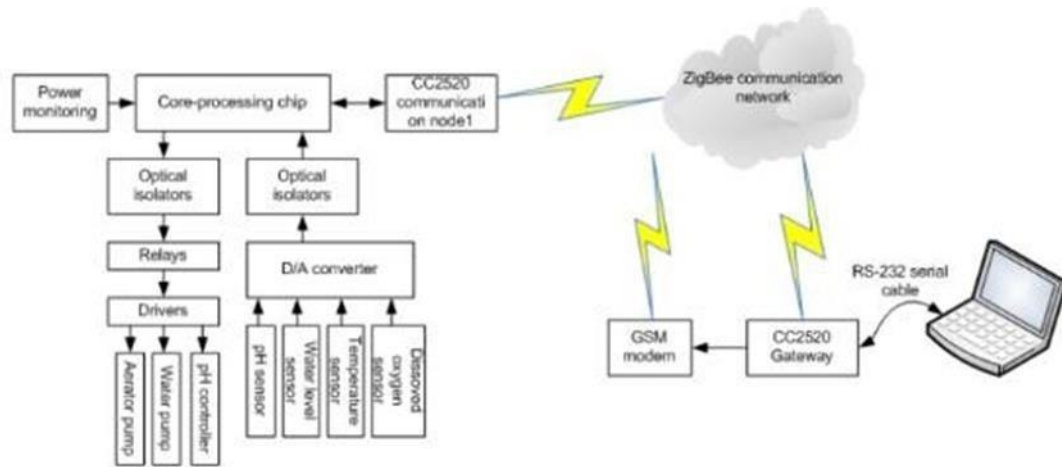


Figure 2.1: Block diagram of the Wireless Sensor Network [21]

In this paper, the implementation of WSN is used to solve the problem of data transmission under difficult circumstances in aquaculture. The WSN are connected closely with the physical environment allowing sensors to provide real time data collected directly from the environment to end user with the use of wireless communication technology called ZigBee. Multiple sensors of various type will be used for the purpose of monitoring information about water quality parameters including pH value, temperature, oxygen level and salinity at every 3 minutes intervals. The data will then send to a host computer via gateway through wireless communication network to perform data processing, analysis and visualisation. Host computer act as a central monitoring platform with necessary data analysis and monitoring program installed on it. The host computer displays the results and sometimes alarm the operator with short message if the water quality is not according to the standard [21].

CHAPTER 2 LITERATURE REVIEW

Strength

For this WSN based smart aquaculture system, data collection and transmission are done automatically by using sensors and ZigBee communication network. The system uses a centralized control system that receive data from sensors. The host device allows remote monitoring water quality in real-time to enables quick detection of any changes in the water quality which can improve the efficiency and effectiveness of aquaculture management. Implementation wireless sensor networks also a cost-effective alternative to traditional monitoring methods, mainly on reducing labour cost.

Weakness

The water quality monitoring and control system for aquaculture using WSN has its strength. However, the system also has some weaknesses, WSN have very limited range, which may require the installation of additional sensors or equipment to cover larger aquaculture facilities. WSN also vulnerable to signal loss, or environmental factors that may affect the accuracy and completeness of the data collected. Moreover, it also requires a stable power supply, which may be a challenge for remote or off-grid aquaculture facilities. The system also generates relatively large amounts of data that need to be stored, processed, and analysed, which can be a challenge for a host device.

2.1.3 IoT Based Smart Fish Farming Aquaculture Monitoring System

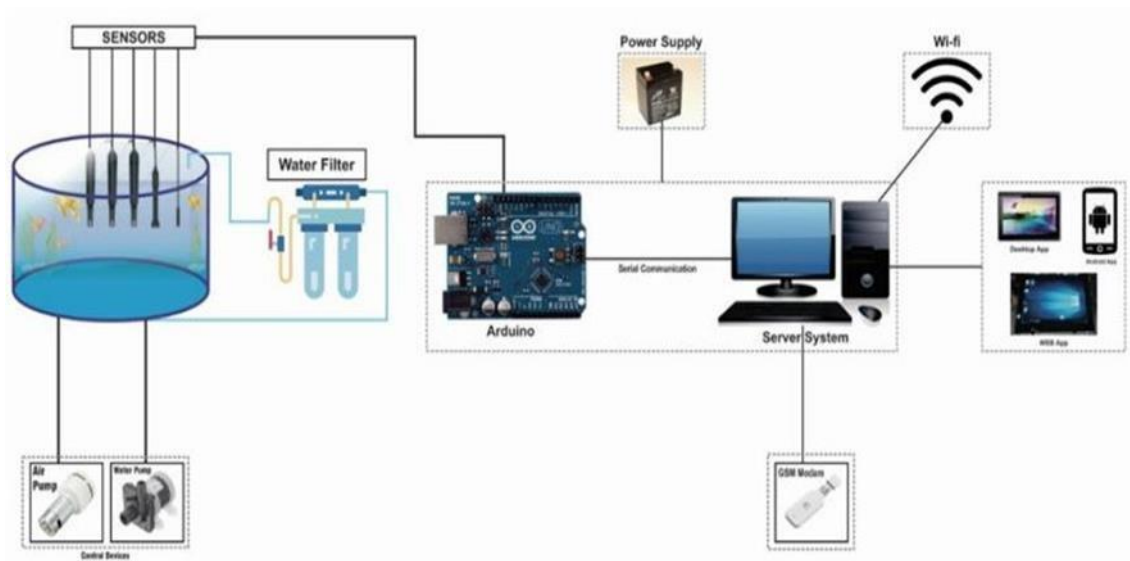


Figure 2.2: Block diagram of the smart aquaculture system [22]

An IoT based smart aquaculture solution are proposed by Karim et al [22] to obtain real time data accurately to maximize the production. This project is focus on low cost by using short range wireless sensors network to help fish farmer monitor and measure water quality like pH, temperature and behaviour of fish as well as control in real time. The motivation of the system is to cope the drawbacks of manual water testing which lead to the increasing death rate of fish and affect their growth rate as well as internet access problem in the field areas. The system integrated varieties of sensors of different purpose and internet technology combined with a user-friendly interface for monitoring and interaction between the system and operators on smartphone or website. A GSM modem is used to send message to the operator when the water quality crosses the safe range without internet otherwise the operator monitors the water parameters through website or application from their devices if internet connection available. Besides, an ordinary computer was served as a server to analyse output values from sensors and act as a database to store data. Processed data are visualized on mobile app, desktop app, and website to assist farm operator in decision making.

CHAPTER 2 LITERATURE REVIEW

Strength

The system provides remote access for farmer to perform real-time monitoring of water quality, motion and other important parameters across the network. It also uses a rule- based alarm to alert farmer if any of the water parameter exceed threshold, the alert message is presented on a dashboard along with the real time water parameter which helps the farmers to take timely action. A server is used to compute, manage and store the sensors data and the necessary data could be retrieved by user by consuming minimum cost of internet.

Weakness

The proposed IoT-based smart fish farming aquaculture monitoring system uses a local server as the central hub for data processing and storage. However, this approach has some potential weaknesses. Firstly, the local server may have limited processing power and storage capacity, which could affect the scalability of the system as the number of sensors and data points increase. Secondly, since the local server is a single point of failure for the system, any issues or downtime could potentially disrupt the monitoring and management of the fish farm, leading to negative impacts on fish health and productivity. Moreover, during maintenance, the system could go offline, leading to potential loss of productivity. Finally, the use of a local server may pose data security risks, as it could be vulnerable to cyber-attacks or physical damage.

CHAPTER 2 LITERATURE REVIEW

2.1.4 Application of machine vision systems in aquaculture with emphasis on fish

Monitoring fish behaviour and status during pre-harvest to prevent severe loss caused by diseases and stress issues using optical sensors and MVS, the system also contributes to assessment, prediction as well as measurement of fish quality [23]. In this research, the authors emphasize the benefits of using both underwater and surface-based cameras to obtain images and videos of fish to determine their welfare. According to studies on their behaviour, they are better than conventional methods that cause unnecessary stress on the fish population. MVS uses image analysis, which involves two steps: image acquisition and image processing on physical objects to construct descriptions and information for them, while video tracking monitors the object's activity and analyses it automatically. Both image and motion analysis are used to quantify behavioural parameters of objects without human presence. For example, when fish are infested by parasites, they will exhibit increasing leaping behaviour. The application of MVS provides insight for farm operators in decision-making according to the intended circumstances.

Strength

The article provides a comprehensive review of the state-of-the-art in MVS applied to aquaculture, with a particular focus on fish. The use of optical sensors to collect data on fish and their behaviour without causing harm or stress to the fish is particularly important for analysis purposes, where it is important to minimize any negative impact on the subjects to be analysed. Besides, it also automates the collection and analysis of data, reducing the need for human intervention and increasing efficiency since monitoring individual fish manually would be time-consuming and impractical. The use of MVS for measuring fish size, weight, and observing behaviour will be more accurate and consistent than manual measurements.

CHAPTER 2 LITERATURE REVIEW

Weakness

This article focuses almost exclusively on fish and does not cover other aspects of aquaculture such as shellfish which is also major product of aquaculture. Besides, implementing and maintaining machine vision systems can be technically challenging, especially for producers who do not have a background in engineering or computer science. Moreover, the article does not discuss the potential economic costs associated with implementing MVS in aquaculture, which could be a significant barrier to adoption for some producers considering its involving AI and some expertise.

2.1.5 Toward Precision Aquaculture: A high performance, cost effective IoT approach

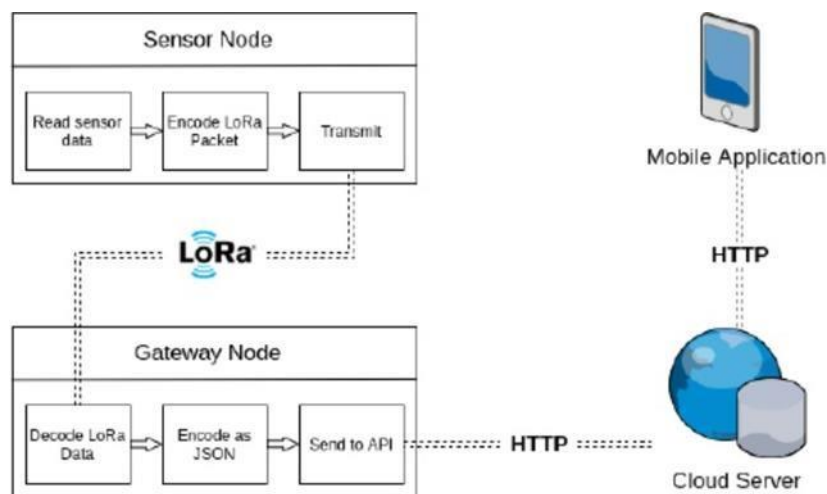


Figure 2.3: Simplified network architecture [24]

This paper describes a project focused on developing a networked sensor system for aquaculture that addresses power and communication limitations in both inland-based and offshore facilities [24]. The system based on implementation of IoT devices and incorporates machine learning models into local devices to enhance aquaculture activities by providing real-time data on water quality visualization for decision making. The proposed system consists of two levels of data interaction, a local layer, and an external layer. The local layer is responsible for communication between sensor nodes and the gateway node over LP-WAN and LoRa technology. LoRa was chosen due to its low energy consumption and enhanced communication coverage. The sensor data is encoded in a LoRa packet and transmitted to a gateway node within range. The gateway node then forwards the sensor data via HTTP to an external cloud server. On the other hand, the external layer is responsible for communicating the sensor data to the cloud infrastructure via gateway nodes, which can use either WiFi or cellular networks. The cloud service will allow for real-time data analysis and decision-making in aquaculture, improving the efficiency and sustainability of farming activities.

CHAPTER 2 LITERATURE REVIEW

Strength

As a precision aquaculture prototype itself, it benefits from the implementation of IoT device as well as the use of cloud platforms. It enables real-time monitoring of various parameters, including water quality, fish behaviour, and feeding patterns, providing farmers with the ability to make data-driven decisions. The system is also scalable and cost-effective, making it accessible to small-scale aquaculture farmers who may not have the resources to invest in expensive technology. The authors propose that the system can be easily customized to meet the specific needs of different farmers and can be expanded as the business grows. One of the major benefits of the system is the use of LoRa transmitters for communication, which have low energy consumption and enhanced communication coverage. This makes it suitable for use in farms located in remote areas. Additionally, data is processed in the cloud, reducing the need for on-site computing resources.

Weakness

Despite its potential to address issues in aquaculture, the system also has some limitations. One of the main vulnerabilities of the system is its dependence on internet connectivity. The sensors collect data which is then transmitted over the internet to the cloud for processing and visualization. This means that any disruption to the network could render the system inoperable and prevent farmers from accessing the latest data. Additionally, the transmission of raw data to the cloud is costly in terms of network bandwidth and can introduce latency. This could negatively impact the real-time monitoring and decision-making capabilities of the system. Machine learning models also embedded in some of the IoT nodes. However, to improve the accuracy of these models, a sufficient amount of data is required. As the system is relatively small in scale, the amount of data generated may be limited, which impact the ability to train a better model.

2.1.6 Precision Aquaculture by Fearghal O’Donncha and Jon Grant

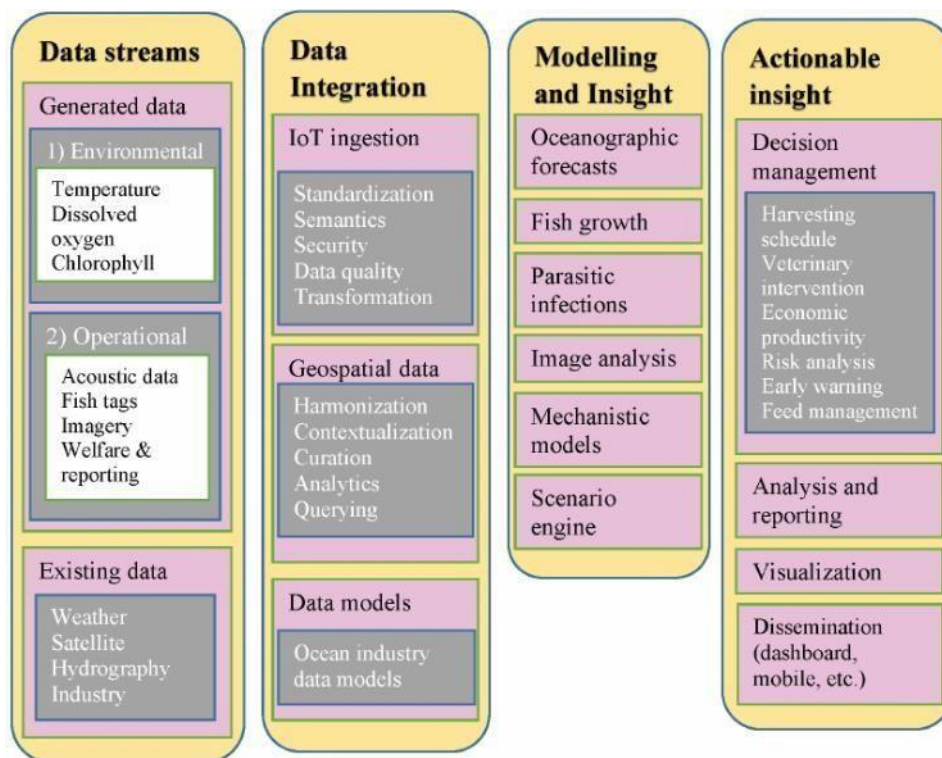


Figure 2.4: Four pillars of precision aquaculture framework [25]

This article is about ongoing effort to develop an autonomous framework to manage farm by implementing precision aquaculture concept across a number of marine fish farm in Canada [25]. The data streams source involves an IoT network of sensors, including underwater wireless acoustic sensors to measure environmental parameters, “CageEye” acoustic systems for individual fish tracking and positioning as well as external datasets such as satellite observations and weather data. All generated data are communicated to IBM cloud. For the data integration, modelling and insight, IoT platform is used to manage various machine learning models and integrate data from sensors and other sources. The machine learning models provide real-time predictions on fish health, biomass, and mortality based on environmental stressors, and inform on outbreaks of parasitic infections. Data assimilation is used to combine mechanistic models with observations to maintain accuracy. Actionable insight consists of real time condition from collected data as well as prediction result from machine learning models. The aim is to use the insight to optimize feeding while minimizing environmental impacts and inform health intervention practices. However, the author mention that the development of sensors and machine learning models is still in its early stages, and

CHAPTER 2 LITERATURE REVIEW

further research and innovation are needed to fully realize the potential of precision aquaculture.

Strength

The whole concept of precision aquaculture is to increased production efficiency and improved environmental sustainability by implementing IoT devices, machine learning and technologies to assist farmers monitor and manage their production systems more effectively. Precision aquaculture also help reduce the environmental impact of aquaculture by optimizing feed management, reducing waste and pollution, and minimizing the risk of disease outbreaks thanks to the analysis from the use of machine learning. Besides, the massive amount of data obtained from various source allow more powerful machine learning model to be trained and deploy for different circumference. Additionally, the use of cloud computing provides a cost-efficient option for farmers by eliminating the need for on-site computing infrastructure. The use of IBM cloud allows the access to variety of cloud services including computing power to train machine learning model, data visualization and reporting tools without the need of maintaining too much physical infrastructure.

Weakness

The scale of precision aquaculture involved are relatively large with the utilization of more sophisticated technologies such as drone-based imaging and satellite-based monitoring that will not normally be seen in smaller farm. The implementation of precision aquaculture in the article can require significant upfront investment, particularly the cost of sensors and monitoring systems. It also requires reliable internet connectivity since it rely on computing resources located in cloud. Besides, the need of specialized technical expertise to maintain the marine facilities and IoT devices. As a result, this concept is less applicability to small-scale farmers because most of the precision aquaculture technologies discussed in the article may not be cost-effective for smaller scale farmers.

2.1.7 Recent Advancements in Deep Learning Frameworks for Precision Fish Farming Opportunities, Challenges, and Applications

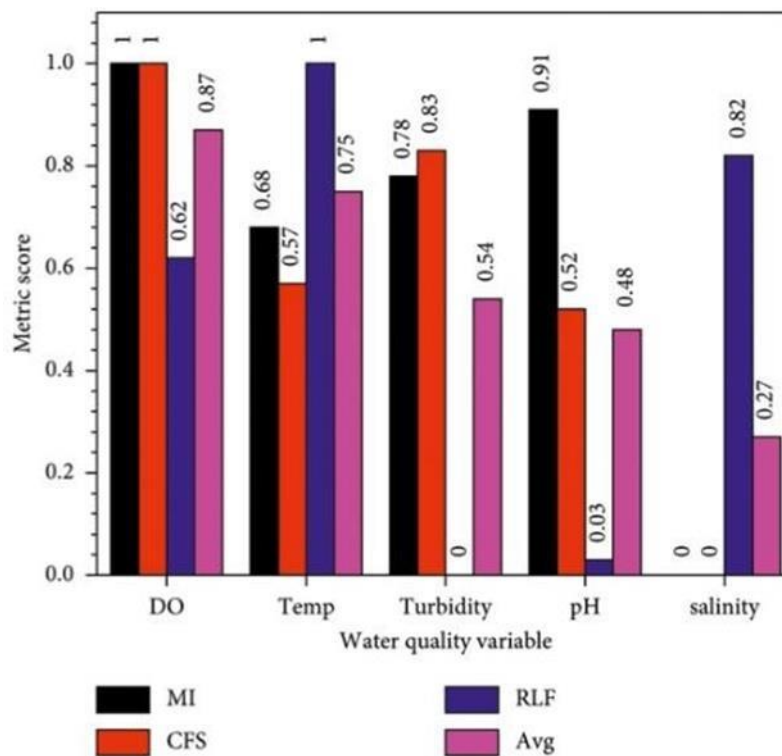


Figure 2.5: The effect of water quality on harvest result [26]

This article discusses the application of deep learning and machine learning in smart fish farming [26]. Although the article is focusing on exploring the technical specifics of techniques, including data, algorithms, and performance, and how they can improve efficiency, productivity, and sustainability in aquaculture rather than precision aquaculture concept. One main thing is it mentions the use of IoT sensors to collect water parameters such as dissolved oxygen, pH, temperature, turbidity and salinity in smart fish farming. The data collected by the sensors is then analysed using machine learning and deep learning algorithms to provide insights and make predictions as well as the source for model training. Figure 2.5 shows the metric score of water quality in affecting growth and yield of livestock, it supports why machine learning can be used to maintain water quality variables to keep it within optimal boundaries. They also reveal challenges and limitations of DL and ML-based adaptive fish farming techniques, including the need for labelled training datasets, high computational power requirements, and costly equipment. Despite these limitations, DL and ML have the potential

CHAPTER 2 LITERATURE REVIEW

to extract features automatically, produce high-precision processing outcomes, and improve over time.

Strength

The article stated that using Machine Learning (ML) and Deep Learning (DL) in aquaculture, can help improve various aspects of aquaculture operations. ML and DL can be used to analyse large amounts of data collected from sensors, cameras, and other sources in fish farms. This data can be used to optimize feeding, water quality, and other environmental conditions to promote better fish growth, health, and survival rates. The trained models can also be used to detect and predict disease outbreaks, monitor fish behaviour, and identify abnormalities in fish growth patterns. The benefits of applying ML and DL in aquaculture include increased efficiency, reduced costs, and improved sustainability. By optimizing fish growth and health, aquaculture operations can produce more fish with less feed and resources. This can help reduce the environmental impact of aquaculture and increase the availability of seafood for human consumption. Additionally, the use of ML and DL can help reduce the risks associated with disease outbreaks, which can result in significant economic losses for fish farmers.

Weakness

As the article mentioned, the need for labelled datasets to train an accurate ML and DL model depend on the quality and quantity of dataset where collection and labelling of large amounts of data can be time-consuming and expensive. Besides, machine learning and deep learning algorithms require significant computing power to train and operate. It can be challenging for smaller farms or those with limited access to costly high- performance computing infrastructure. This assumption was made as the article not mentioning the usage of cloud services which is on demand computing resources that pay at the rate they used. Moreover, the access of limited datasets of fish species because the data obtain are only limited to the species the farm have. This situation made the use of ML and DL algorithms may not be equally effective for all fish species.

CHAPTER 2 LITERATURE REVIEW

2.2 Chronological trend of reviewed system

The application of information technology in aquaculture industry can be trace back to the previous decades. Various kind of technologies has been applied to increase the efficiency and production of aquaculture as well as minimize the waste. The implementation of information technology in aquaculture operations can be range from obtaining data from using sensors, data transmission network, data processing and analysis, data storing and data visualization to assist farm operator in decision making.

Data collection from the field can be done by using wide range of sensors of different purpose and it is directly connected to a computer [15]. The data collected from sensors will then need to be downloaded and store in a meaningful structure to make it available for later use. Thus, the computer itself are served as database using some commercial off the shelf database management software that support data storing and database worksheet [16]. With the stored data, DSS is used to visualize data with UI for farm operator and provide an interactive control to help in data monitoring and decision making based on the data obtained [17]. The system from the late 80s and 90s are fairly limited in term of efficiency and effectiveness, most of the sensors involve are either wired or the collected data need to be downloaded manually to computer for further processing. Besides, most of the workload are concentrated on a personal computer from data storing to data presentation, the computer will be the single point of failure.

After the advancement of information technology in 21st century using WSN are feasible to make all the devices connect remotely. Wireless sensors are used to obtain real time data from the fish farm and directly send to host computer using wireless communication technology like ZigBee rather than having a complex physical infrastructure like its predecessor [21]. But the host computer still acts as a central unit for data storing, processing and also data presentation. In contrary, Karim et al [22] propose a smart aquaculture solution based on IoT to obtain real time field data. A computer served as a server for data analysis and data storing are used, but the data processing and data visualization are separated using application as well as website. For this proposed system, it did not utilize internet connection due to the consideration on limited internet access from the project's environment.

On the other hand, ES the first successful form of AI is used to automate data analysis process using control system [18]. It is integrated with a filtration system to perform control if the data from sensors are not according to the standard. Image processing and pattern recognition are also used to extract information from image capture from video sensors [19].

CHAPTER 2 LITERATURE REVIEW

The system uses image recognition to perform counting on the fish population, but with poor accuracy. The usage of AI is still uncommon and most of the system are for research purpose or prototype because of technology constraint at the moment. Jump back to 2017, (Saberioon et al) proposed MVS using optical sensor to analyse fish behaviour to determine their health status. MVS involve image acquisition and image processing to construct a description on the fish by comparing their behaviour with a model [23]. But one can argue that the accuracy of the system because the only parameter uses to determine fish health are image and video obtain from sensors, water quality such as pH value, temperature, biomass, oxygen level and salinity also play an important role in influence fish welfare.

In recent times, precision aquaculture has gained significant traction, with many farms adopting this approach to aquaculture. This can be observed in [24], which describes a networked sensor system designed for aquaculture and [25], which highlights the implementation of precision aquaculture concept in several farms in Canada. Precision aquaculture leverages IoT technologies such as sensors and cloud platforms, to collect field data, which is then analysed using cloud-based services and AI to provide data-driven insights [27]. [26] highlights how the deployment of machine learning or deep learning can significantly improve aquaculture production. The data

*models used in precision aquaculture. [24] and [25] are both examples of precision aquaculture. However, there are some differences between the two. [24] describes a smaller and more scalable design that is suitable for smaller farms, while [25] involves the deployment of precision aquaculture practices on an economical scale that can only be achieved by larger companies. Nevertheless, the primary objective of precision aquaculture is to optimize the efficiency, sustainability, and profitability of aquaculture operations, while simultaneously minimizing the environmental impact.

CHAPTER 2 LITERATURE REVIEW

2.3 Comparison between the previous aquaculture framework and proposed solution

Existing System Features	2.1.2	2.1.3	2.1.5	2.1.6	Propose Solution
Scalability	No	No	Yes	No	Yes
Reliability	No	No	No	No	Yes
Cost Effective	Yes	Yes	Yes	No	Yes
Wireless Connection	Yes	Yes	Yes	Yes	Yes
Low Bandwidth Consumption	Yes	Yes	Yes	No	Yes
Lower Latency	No	No	Yes	No	Yes
Cloud Based	No	No	Yes	Yes	Yes
Real-time data monitoring	Yes	Yes	Yes	Yes	Yes
Remote Sensors Controlling	No	Yes	Yes	Yes	Yes
Machine Learning Implementation	No	No	Yes	Yes	Yes

Table 2.1 Comparison between the previous aquaculture framework and proposed solution

CHAPTER 2 LITERATURE REVIEW

2.4 Comparison between previous machine learning method and machine learning in proposed solution

Previous Method	2.1.4	2.1.7	Proposed Solution
Federated Learning	No	No	Yes
Data Privacy	No	No	Yes
IoT as Data Source	No	Yes	Yes
Periodically Fine-Tuning to Improve Accuracy	Yes	Yes	Yes
Aggregating Trained Model of Different Source	No	No	Yes
Less Reliance on Local Data Availability	No	No	Yes

Table 2.2 Comparison between previous machine learning method and machine learning in proposed solution

2.5 Proposed Solutions

According to the reviewed system above, each of them have some limitations due to technology constraint at that time or the system are design based on certain circumference. The earlier reviewed system relies on a single computer to serve as database, data processing as well as data visualize platform, this made the system vulnerable to any disaster or human error and become a single point of failure and potentially loss of valuable information about the farm. Besides, data visualization on the WSN can only be achieve on through the central physical device. Although both of the reviewed precision aquaculture system utilizing cloud services, they are less reliable because rely on transmitting sensor data to cloud for processing and vulnerable to network disruption. Moreover, the application of AI in the form of expert system in early day to perform data analysis, image processing and pattern recognition technique also used but their accuracy are questionable. With the advancement of hardware and software, AI are able to achieve batter accuracy in data analysis. Different machine learning algorithm are uses nowadays in aquaculture including MVS where it applies pattern recognition to study fish behaviour to determine their health status. Machine learning often fine-tune periodically using

CHAPTER 2 LITERATURE REVIEW

data collected in the environment where they deployed. In some case, the dataset is not large enough to improve model through training. Sometimes, model is trained on cloud using its computing power, but this method doesn't ensure the data privacy of the farm.

Therefore, the proposed system will design a precision aquaculture using IoT alongside with cloud and federated learning. In the proposed system, IoT device and sensors located within an edge environment are used as a primary way to collect real time environment parameters. An edge device will act as a core to perform pre- processing on the collected sensor data, the processed data can be used to train a local machine learning model. Local machine learning model will analyse and make prediction on the sensor data to provide some insights regarding the data. Only the processed data, machine learning result and trained machine learning model will be transmitted to cloud for storing via edge device. The use of edge computing benefits from its low bandwidth requirement and low latency because the data are process locally as well as only the relevant data are sent to cloud. This method is suitable for aquaculture since most of them are usually located at remote are with unstable network connectivity. Besides, cloud service also brings many benefits, it is usually provided by third party which they control and maintain the infrastructures. Hence, cloud database is used to store processed data and trained model and where they're available for retrieval by any devices anytime. It also a backup solution making the system more reliable and disaster proof which some reviewed systems are otherwise. The sensor data along with analysis results will be visualize on a dashboard that can be access by authorize user. A centralize machine learning model are used to aggregating trained model from different edge to create a more robust and powerful model before distributing back to every edge. The method used is called federated learning, the centralize model still can benefits from the variety of model trained with their own data without even have access to the data.

Chapter 3: Project Scope and Objectives

3.1 Project Scope

The problem statement above stated that conventional way to determine fish welfare and measure water quality are inefficient and labour intensive. Often, to determine welfare of fish involve in catching them to obtain sample, but this process will make the fish population stress which further affect the overall production. The process of obtaining water sample for quality test are also tedious, it involves collecting sample from every pond and require an experience tester to perform water quality testing and prone to human error. This project aims to replaces conventional way by implementing IoT cloud solution, it improves the productivity and efficiency in management of aquaculture operation. The system is focus on fish stock and potentially applicable to other aquaculture product such as prawns and crabs. IoT devices and sensors will be used to obtain real time field data such as water parameters and video image without human involvement within the edge and then pre-process them locally. The processed data can use to train local machine learning model and deploy locally to perform intelligence tasks such as make prediction of the water quality trend and alert fish farmer for possible threshold breach as well as provide recommendation if necessary. After that, processed data and trained model will be sent to cloud using an edge device. Cloud storage and database will used to store all the data and machine learning model from various edge environment for visualization and federated learning. Besides, visualization is done by presenting pre-process water parameters along with other prediction results graphically on a dashboard for farm operator to perform data driven decision making.

3.2 Project Objectives

1. Design and develop an IoT-Cloud platform for precision aquaculture.

The objective is to develop an IoT-Cloud platform that supports precision aquaculture by integrating an edge computing environment with a cloud platform. Within the edge computing environment, various IoT devices consisting of sensors, cameras and other equipment will be utilized to gather real-time data on fish and their environment that serves as the primary source of data. The raw data collected will be pre-processed and store locally, with only the relevant data being transmitted to the cloud platform via an edge device, such as a server machine gateway. A machine learning model located within the edge computing environment can leverage the locally pre-processed data for model training purposes. The cloud platform will manage and store the pre-processed data as backup and trained model weights from various edge in a database according to its intended purpose, making it readily available whenever needed. This approach reduce latency since data processing and analysis are performed closer to the source of data. Besides, it consumes less network bandwidth and cost saving because only the necessary data are transmitted to the cloud.

2. Perform data visualization and analysis using an intelligent cloud-based dashboard interface.

This objective is to develop an intelligent cloud-based dashboard interface that combines data visualization and analysis capabilities. The dashboard will leverage cloud infrastructure as the primary source of data while providing near real-time visualization to authorized users. The dashboard offers interactive visualizations, allowing users to explore general information about all the tanks in the aquaculture farm including status of every tank and number of breach cases. Besides, it able to apply filters and drill down into details of individual tank displaying water parameter trends and the overall situation. Intelligent features, such as predictive analytics offer by BI tools, enhance data analysis by enabling users to forecast near-future trends. The dashboard offers a brief but insightful information to users, supporting data-driven decision-making for management.

3. Implement federated learning in distributed edge environments.

Another objective is to implement a federated learning approach in distributed edge environments. Once the conditions are met, a federated learning server hosted in the cloud will serve as a coordinator, orchestrating the federated learning process across the edge environments. Each edge environment will train its own local ML model using its respective sensor data, transmitting only the model weights to the cloud-based server for aggregation, resulting in a centralized global model. Subsequently, the global model weights are distributed back to each edge environment. This process will be repeated according to the predetermined number of training rounds set prior to the federated learning process.

Chapter 4: Methodology and System Design

4.1 System Architecture Diagram

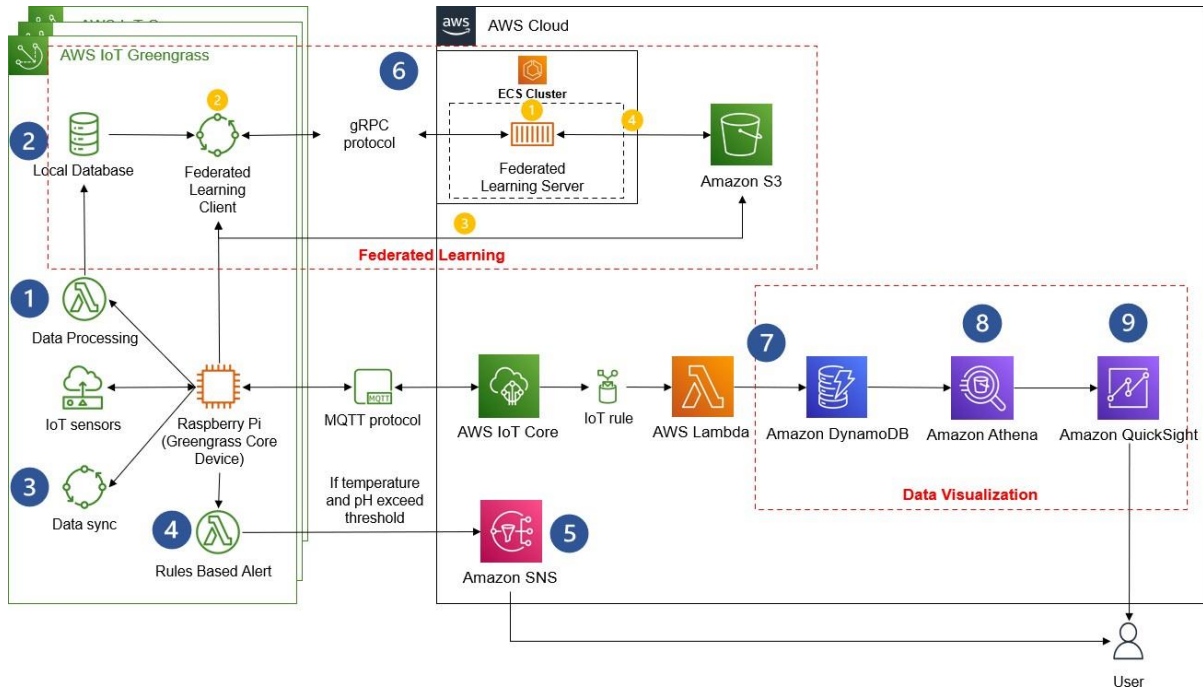


Figure 4.1: AWS Cloud Architecture Diagram

- 1 • The raspberry pi kick start the sensors reading and process data, pH value and temperature are collected every 10 seconds regardless the network connection
- 2 • The sensors data are store in local storage, a relational database regardless the network connection
- 3 • **If network present:** periodically check the temporary storage to perform data syncing with DynamoDB
- 4 • While the sensors reading data, a rules-based condition will be evaluated
- 5 • If temperature or pH reading exceed acceptable threshold, an alert will be sound via email to user
- 6 • Federated Learning
 - 1 • Server initialize a global model weight and upload to S3
 - 2 • Client download global model weight and perform local training using local data
 - 3 • Client upload trained local weight to S3
 - 4 • Server perform model aggregation using weights from clients and upload aggregated weight as global model back to S3
- 7 • **If network connection present:** the sensor data will be store into DynamoDB located in cloud as a backup source, else data will be duplicate into a temporary storage
- 8 • Athena query data store in DynamoDB to serve as a data source for visualization
- 9 • A near real time interactive dashboard visualize data for farm monitoring

Figure 4.2: Description of Steps in Architecture Diagram

CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

Figure 4.1 above shows the architecture separate into two area, **AWS IoT Greengrass** which represent the edge environment and **AWS Cloud** serves as shared infrastructure that are accessible by different edge environment. **AWS IoT Greengrass** is a service that enable the creation, deployment and management of our edge runtime.

Starting from the edge environment, water parameter is obtained by using various type of IoT sensors. A raspberry pi is register as a **Greengrass core device** to run **Greengrass core software** which facilitates the deployment of lambda functions and custom components/algorithm. The local lambda function controls the IoT sensor readings and pre-process incoming sensor data before storing it into a local relational database for further use. Simultaneously, the sensor data is sent to the cloud for backup as well as visualization purpose. Additionally, a data synchronization component periodically activates to sync the data in the local database with the cloud database in case of network reconnection. Furthermore, a rule-based alert system detects any water parameters that exceed preset thresholds and triggers **Amazon SNS** to send emails to alert users. Moreover, the **Greengrass core device** also acts as a gateway to transmit data and communicate with the cloud using the **MQTT protocol**.

On the other hand, a **Fargate container** is located inside an **ECS cluster** in the cloud to host a server for federated learning, serving as a coordinator for the entire federated learning process. Besides, **Amazon S3** serves as a proxy between clients and server in the process, storing global and client ML weights. When the federated learning process starts, the server initializes an ML model, and its weights are uploaded to **S3** as a centralized global model. Federated learning clients on different edge environments download the global model weights from **S3** and perform local ML training using data stored in their local databases. The trained ML model weights from various edge environment are then stored back in the **S3 bucket**, which the server uses to perform weight aggregation, combining all the weights into a single global update and uploaded back to **S3** as the global ML mode. This process repeats until the training round ends. All communication between servers and clients is done through the **gRPC protocol**.

Meanwhile, inside the **AWS Cloud**, **AWS IoT Core** receives data sent from the Greengrass core device using the Pub/Sub service. The sensor data is then sent to a lambda function for further processing via an **IoT rule**, which defines the lambda function responsible for handling

CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

data from the Pub/Sub service in **IoT Core**. In this case, the sensor data is processed and categorized based on their source and stored in **Amazon DynamoDB**. **DynamoDB** is chosen for its low latency in data retrieval, which is beneficial for providing near-real-time visualization. **AWS Athena** serves as the query engine that retrieves data from **DynamoDB**, enabling seamless initialization and data extraction and transformation, later serving as a data source for visualization. **Amazon QuickSight** is used to build an interactive dashboard utilizing data from **AWS Athena** to provide quick insights and analytics for users. Authorized users can then access the dashboard with only the relevant data presented graphically for data-driven decision-making.

4.2 Methodology

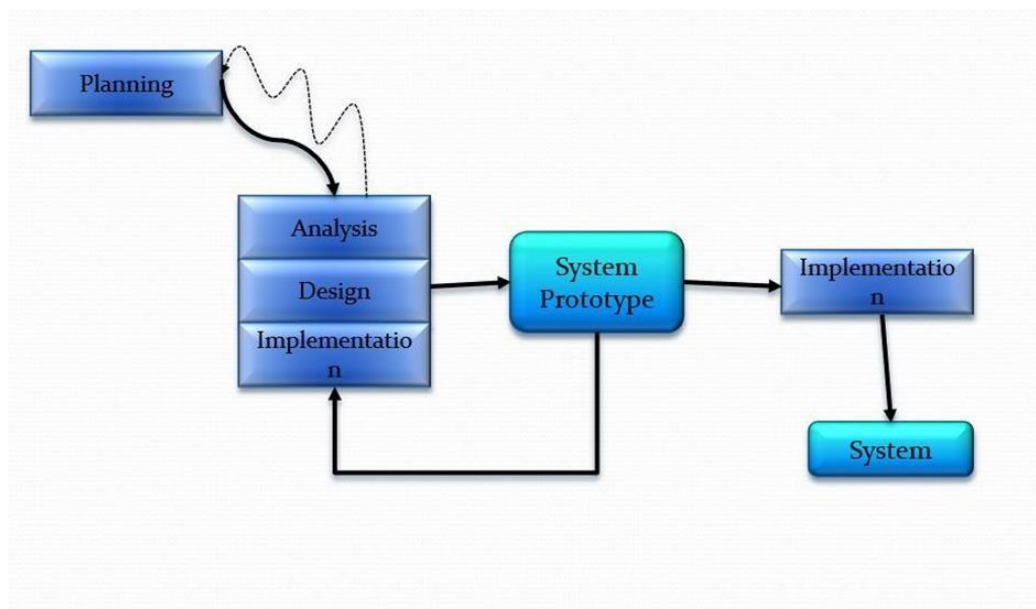
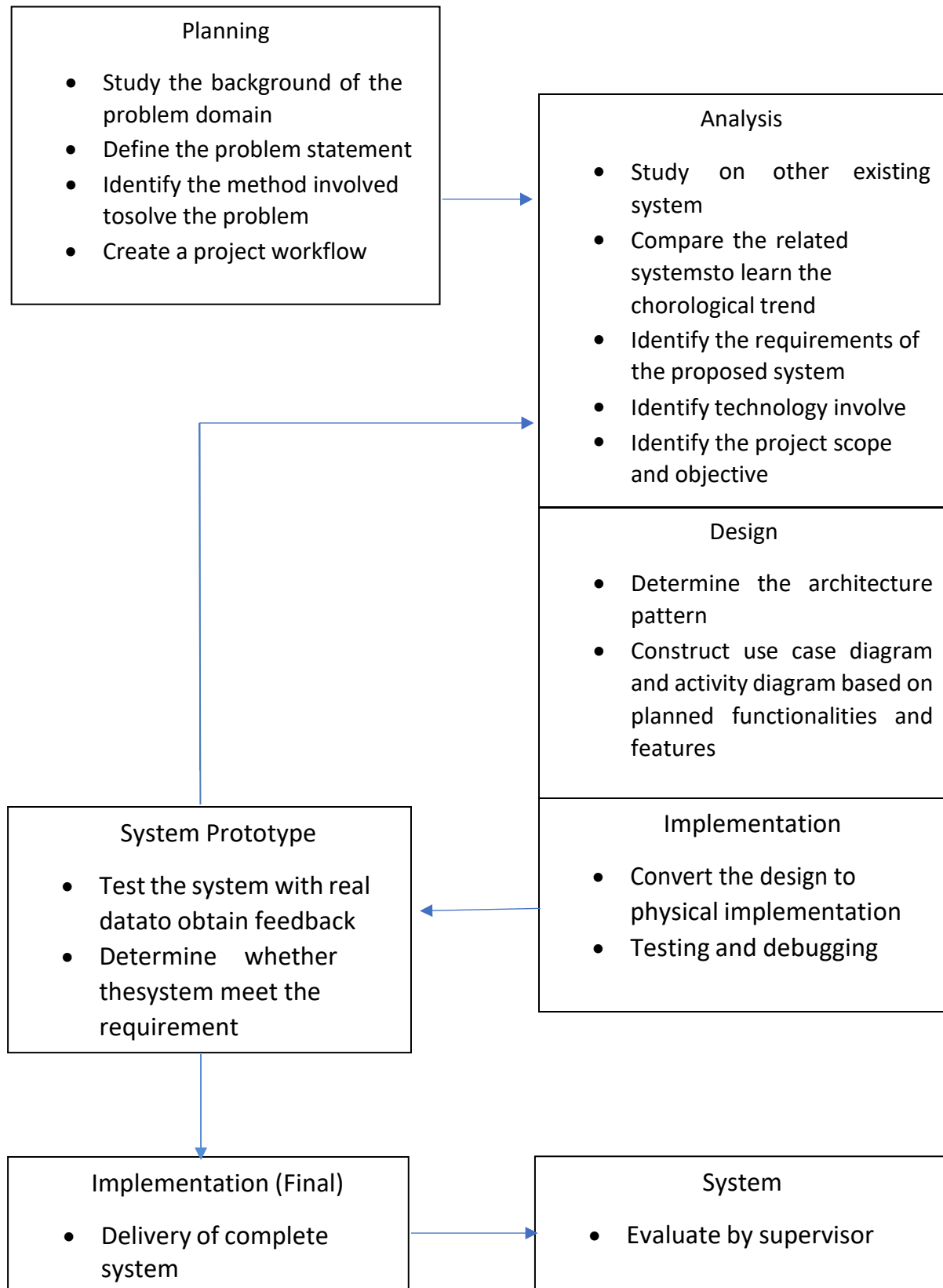


Figure 4.3: Prototyping Development Methodology

The IoT-Cloud Solution for Precision Aquaculture will be developed using RAD prototyping as methodology. This prototyping-based methodology will start with an initial planning. After the planning phase, analysis, design and implementation are performed concurrently and repeatedly in a cycle to refine the prototype. Prototypes are produced in every cycle to obtain feedback and refine the system until the desired outcome is achieved. As the system satisfies all the defined requirements, it is ready to be implemented.

4.3 Project Workflow



CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

4.3.1 Planning Phase

This project starts with planning phase, study is conducted to gain insight regarding the problem domain of this project. After the study, the problem statement is defined and a solution with appropriate method involved to solve the problem will be identified. Then a project workflow will be constructed to shows the model of system development life cycle.

4.3.2 Analysis Phase

After the planning phase, the requirements of the system need to be defined by reviewing the existing systems to learn their chronological trend. Comparison is made on the reviewed systems to acknowledge what solutions are implemented to solve the problem throughout the times and the characteristic of the systems. Then, the requirements of the system are identified based on the existing system with the use of different approach to eliminate their weaknesses. From the defined requirements, we identify the technology involved in the project as well as the project scope and objectives.

4.3.3 Design Phase

The next phase after analysis phase is design phase, the requirements of the proposed system are turned into system specifications. The specification in term of features and functionalities are used to construct a use case diagram and activity diagram. After that, a suitable architecture pattern is assigned to this system.

4.3.4 Implementation Phase

Once the design phase is completed, the design will be realized to physical implementation according to the system specifications. Setting up the environment with related hardware and coding the algorithm will be done. A machine learning model will be trained with related dataset. AWS will be the platform for the integration of trained model and also as a database to store the incoming data from sensors. A dashboard will be developed for data visualization and display on a computer. After the physical implementation, the system will become a prototype for testing and debugging.

CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

4.3.5 System Prototype

In this phase, the prototype will be test using the collected data to obtain feedback. From the feedback, determine whether the system satisfy the requirements or else the system will be refined and modify to become other until it meets requirements.

4.3.6 Implementation (Final)

A complete system that satisfies all the requirement ready to be deliver and deploy in the environment.

4.4 Formula of Single-Point pH Calibration

pH calibration is a critical step in the accurate measurement of pH value. Calibration involves comparing the output of the pH electrode with known pH values of standard solutions. The pH meter uses the calibration data to determine the relationship between the output of the pH electrode and the pH values of the solutions being measured [28]. The calibration coefficients, such as slope and intercept, are determined during this process and used to convert the voltage output of the pH electrode into pH units. The PH4502C pH sensor involved in this project require to go through calibration process in order to acquire an accurate pH measurement.

Below is the formula for measuring slope and intercept:

$$\text{Slope} = (\text{pH2} - \text{pH1}) / (\text{mV2} - \text{mV1})$$

$$\text{Intercept} = \text{pH1} - \text{Slope} \times \text{mV1}$$

Below is the formula for measuring pH value:

$$\text{pH} = \text{slope} \times (\text{mV at calibration point}) + \text{intercept}$$

where:

pH1/pH2 are the measured pH of first and second calibration point.

mV1/mV2 are the voltage output of first and second calibration point.

4.5 Technologies and Tools Involved

This project involved using various kind of IoT devices and sensor as well as electronic components to construct a precision aquaculture environment. The operation of the prototype will mainly control by using cloud services, including the activation of sensors in edge environment. Some other software will be use in assisting the development of the proposed system.

4.5.1 Laptop

Description	Specifications
Model	Acer Predator PH315-53
Processor	Intel i5-10300H
Operating System	Windows 10
Graphic	NVIDIA GeForce RTX 2060 6GB GDDR6
Memory	16GB DDR4 SDRAM
Storage	512 GB SSD, 1TB SATA HDD

Table 4.1 Specifications of laptop

The laptop is required for the control of Raspberry Pi using Ethernet cable and SSH via VNC Viewer. Besides, it also uses for configuration of the AWS services including register the Raspberry Pi as Greengrass core device and monitoring the flow of data between edge environment and AWS Cloud. It's computing power also use for testing various machine learning algorithm using Jupyter Notebook to find out the most suitable algorithm.

4.5.2 Raspberry Pi 3 Model B+



Figure 4.4: Raspberry Pi 3 Model B+

CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

The development of the proposed system requires a device act as central device to connect, communicate and manage other IoT device and sensor within the edge environment. The central device needs sufficient computing power to run portion of operation locally and serve as a gateway to communicate with cloud platform. Therefore, Raspberry Pi computer are chosen as a core device for the edge environment, specifically Raspberry Pi 3 Model B+. Raspberry Pi 3 Model B+ having a 1.4GHz 64-bit quad-core processor, 1GB RAM, dual-band wireless LAN, Bluetooth 4.2/BLE, Gigabit Ethernet, 40-pin GPIO header and other specification, which is sufficient for this project. Additionally, Raspberry Pi 3 Model B+ provide decent computing power to perform local processing, the 40-pin GPIO header can be extended to a breadboard to connect sensors and with build-in wireless LAN to communicate with cloud platform with a relatively low cost to acquire it. In short, this Raspberry Pi device will be the core device inside edge environment to perform local processing and control other IoT devices as well as sent and receive packet from cloud platform.

4.5.3 Raspberry Pi T-Cobbler



Figure 4.5: Raspberry Pi T-Cobbler

Raspberry Pi 3 Model B+ have relatively limited number of GPIO pin header, 40 of it in total. For this proposed system, a few wired sensors are connected to the Raspberry Pi as the power source and data reading, the available pin are quickly occupied by different device. Thus, the use of Raspberry Pi T-Cobbler to extend the 40 GPIO pin header to a breadboard allow more wire to be connected to a single Raspberry Pi pin. The ribbon wire is use as the connector between the 40-pin of Raspberry Pi and the T-cobbler with every GPIO pin header name mark on it.

4.5.4 DS18B20 Waterproof Temperature Sensor



Figure 4.6: DS18B20 Waterproof Temperature Sensor

DS18B20 is a waterproof temperature sensor with stainless steel head that can be deployed underwater to obtain the temperature of the environment. It is a 1-Wire interface sensor with only 1 wire along with ground wire needed to be connected to a microprocessor. The measuring range of the sensor is between -55°C to $+125^{\circ}\text{C}$ with $\pm 0.5^{\circ}\text{C}$ reading accuracy in the range of -10°C to $+85^{\circ}\text{C}$. To read the data from the sensor, an external library is required; the most popular library is the Dallas Temperature library. The sensor comes with 3 wires, namely Vcc, GND, and Data, and requires 3.0-5.5V input voltage to power up.

4.5.5 PH4502C Analog pH Sensor



Figure 4.7: PH4502C Analog pH Sensor

CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

PH4502C is an analogue pH sensor use to measure pH value and temperature of the deployed environment. The PH4502C sensor consists of a glass electrode probe which are immersed in the liquid being measured and a reference electrode. The glass electrode probe detects changes in pH and generates a voltage proportional to the acidity or basicity of the liquid. The reference electrode module provides a stable voltage reference to ensure accurate pH measurements. The measuring range of this pH sensor is between 0PH - 14PH while the measuring range for temperature is 0°C - 60°C with accuracy of $\pm 0.1\text{pH}@25^\circ\text{C}$. This pH sensor requires 5.00V input voltage to turn on for data reading with a response time of less than 1 minutes.

4.5.6 ADS1115 Analog to Digital Converter

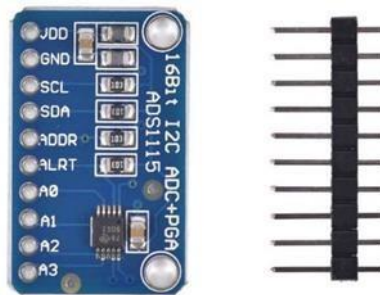


Figure 4.8: ADS1115 Analog to Digital Converter

ADS1115 is an analog to digital converter (ADC) for the conversion of analog signal into digital signal. Since all Raspberry Pi GPIO pin are designed to receive only digital signal, to read pH value using PH4502C sensor, ADS1115 are used to convert the analog voltage signal of pH sensor into digital signal. The ADS1115 features four input channels that can be configured as single-ended or differential inputs. It has a relatively wide operating voltage range from 2.0V to 5.5V, therefore the VDD of it are connected to 5.0V pin on Raspberry Pi for our case. Besides, I2C interface are used by ADS1115 for communication with a microcontroller or other digital device.

4.5.7 Raspberry Pi OS – Debian Bullseye (64-bit)

Raspberry Pi OS is a Linux-based operating system designed specifically for the Raspberry Pi. It is the official operating system for the Raspberry Pi, developed and maintained by the Raspberry Pi Foundation. The Debian Bullseye 64-bit are the OS chosen specifically to be installed on Raspberry Pi 3 Model B+ due to the TensorFlow library to be use require 64-bit OS and lambda functions require Linux-based OS to run on edge device. The lambda functions rely on cgroups feature of the Linux kernel because AWS Greengrass Core v2 runs as a Docker container, and Docker uses cgroups to manage resource allocation and usage for containers.

4.5.8 Amazon Web Services (AWS)

AWS is a well know Cloud platform that offers a wide variety of on-demand services for provisioning computing resources quickly and cost-effectively. The AWS services involved in the proposed system include AWS IoT Core, AWS Lambda, Amazon SNS Amazon DynamoDB, Amazon S3, AWS Athena, Amazon ECS and Amazon QuickSight. AWS IoT Core is a key service used to manage IoT devices within the AWS IoT. Greengrass edge environment. It provides a Pub/Sub service that enables sending and receiving data from the edge environment. Also, the deployment of various component that enable the edge environment to work independently with limited connection with cloud. The AWS Lambda is running in both edge environment and cloud, the function allowing response to various events, including IoT events and trigger action to send alert using Amazon SNS. Amazon DynamoDB is a database that provides fast data retrieval and querying. AWS Athena will query data from DynamoDB table and serve as a data source for visualization. Therefore, the Amazon QuickSight will retrieve data from it to present them on an interactive dashboard. Amazon S3 is an object storage to store trained ML model weights from edge environment and a global model weight. Lastly, Amazon ECS will host a Fargate server that will coordinate the federated learning process between the server in cloud and client in edge.

4.5.9 AWS IoT Greengrass Core Software

The AWS IoT Greengrass core software is installed in Raspberry Pi to extend the AWS functionality to edge environment. This made it possible for Raspberry Pi to act locally on the data generated from sensor using lambda function, providing low latency on responding an event. Besides, it reduces the need of constantly communication with AWS Cloud as most processing jobs are done and stored locally and only the relevant data are stored into cloud database as backup. Hence, edge computing is made possible by installing AWS IoT Greengrass core software on an edge device to manage other local IoT devices.

4.5.10 MQTT Protocol

Message Queuing Telemetry Transport (MQTT) protocol is a lightweight messaging protocol used for machine-to-machine communication. The lightweight design of MQTT protocol made it suitable for IoT devices with relatively limited processing power and bandwidth. The MQTT protocol uses a Pub/Sub model, where client device publishes messages to a broker, which then the messages are received by devices that subscribe to it. After the collected data are pre-processed locally, it will be sent to AWS Cloud by using MQTT protocol.

4.5.11 Python Programming Language

Python is used as a main programming language in the development of this proposed system. It is chosen because of easy to understand and write, and widely supported by most programming tasks. Both lambda function running on edge device and AWS Cloud are written in Python among other language options. Besides, the custom code deployed in edge device to read sensor data are also written using Python. Furthermore, it is also used in Jupyter Notebook to build and test a ML model.

CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

4.5.12 JSON

The deployment of custom software modules, also known as “Component,” into edge devices consists of two parts, namely Artifact and Recipe. The Artifact is the custom code that is written in any programming language, while the recipe contains metadata that describes the component. The recipe also specifies the component's configuration parameters, version, lifecycle, component dependencies, and platform compatibility. Recipes can be written in either JSON or YAML, with JSON being the preferred language.

4.5.13 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows users to create and share computational documents. It will use to perform machine learning tasks, including testing various machine learning algorithm from different library. The most suitable model for the proposed system will be selected to further fine-tuned until a satisfactory result is obtained. Since Jupyter Notebook utilizes the computing power of the local device, all the testing can be done on the device without incurring any additional cost. This is because Amazon SageMaker utilizes cloud computing power, which may require payment. Once all the necessary testing and training are completed on the chosen model, it can be migrated over to Fargate container running on Amazon ECS for deployment.

4.5.14 Amazon QuickSight

Amazon QuickSight is a cloud-based business intelligence (BI) and data visualization service. However, when integrating with AWS services like Amazon S3 and Athena, the process is more straightforward compared to connecting with external data sources. In our specific use case, AWS Athena will serve as the primary data source, facilitating the creation of an interactive dashboard that offers a comprehensive overview of all the tanks within an edge environment. The dashboard can be further customized to provide detailed insights at the individual tank level, presenting trends in water parameters, the number of alerts triggered, and the overall status of each specific tank. These insights enable users with the data they need to make informed decisions and take appropriate actions based on their observation.

4.5.15 gRPC Protocol

gRPC is an open-source, high-performance RPC framework developed by Google. The Flower federated learning library primarily relies on the gRPC protocol to facilitating communication between clients and server. This protocol is particularly well-suited for federated learning scenarios, where establishing efficient and reliable communication channels between edge devices, acting as clients, and cloud-based servers is crucial. Given gRPC are design for building efficient and distributed systems, it serves as a foundation for orchestrating federated learning processes across edges and cloud.

4.5.16 Flower Federated Learning Library

Flower federated learning is an open-source library designed to simplify the development of federated learning systems. Federated learning is a machine learning paradigm that allows training a global machine learning model across decentralized edge devices or clients while keeping the training data on those devices, preserving privacy and reducing the need to centralize data. To use Flower, it requires us to implement its subclass with three methods for establishing the client-side functionality. On the server side, we can leverage the default strategy, which encapsulates the federated learning approach or algorithm. However, if neither of these options aligns with our specific use cases, Flower provides the flexibility to create customized algorithms by defining the necessary methods that suit the needs.

4.5.17 TensorFlow

TensorFlow is an open-source machine learning framework that is widely used for building and training machine learning models. It is an essential component in this project, where its capabilities are leveraged to develop Recurrent Neural Network (RNN) models for federated learning in our edge environments and cloud. The RNN is developed for the purpose of predicting water parameter trends that offering foresight into near-future water parameter trends and enhancing the decision making of users.

4.6 Hardware Prototype Design

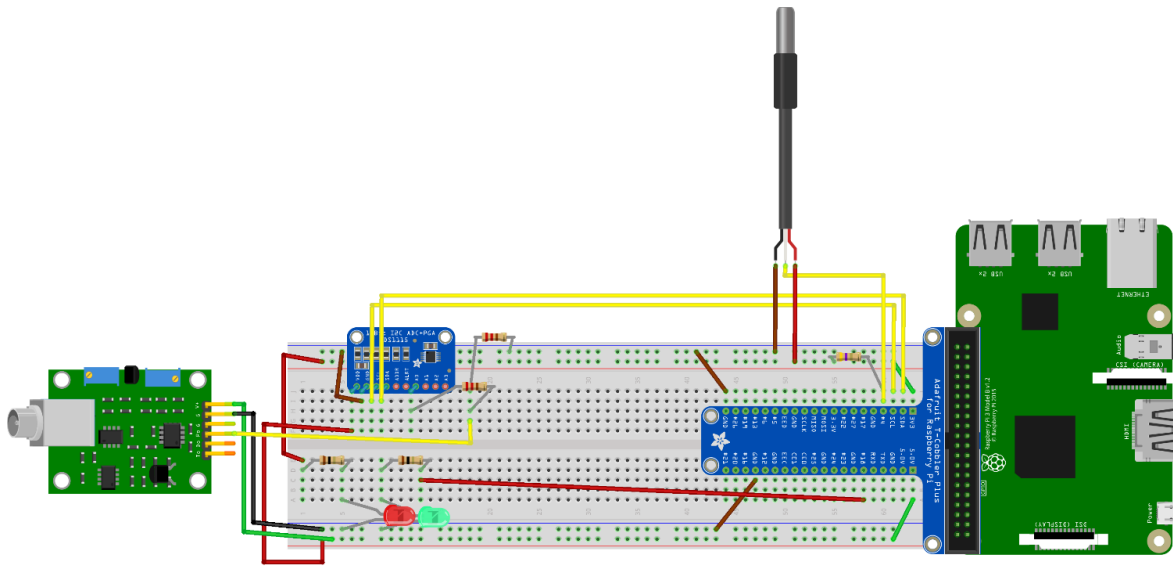


Figure 4.9: Graphical Design of IoT Device

The diagram above illustrates the hardware prototype design of the IoT device in a graphical format. The central component of this IoT device is a breadboard, which is connected to a Raspberry Pi via a T-Cobbler that represents the GPIO pins of the Raspberry Pi. The breadboard is equipped with a pH sensor, an analog-to-digital converter, a temperature sensor, and several LEDs, all connected to the T-Cobbler using jumper wires. The Raspberry Pi serves as the primary processing unit and is installed with the Greengrass Core Software, enabling it to function locally without heavy reliance on AWS cloud. With the Greengrass Core Software, the Raspberry Pi can perform various local tasks, including reading sensor data, processing data, storing data, and establishing communication with the AWS cloud. Furthermore, additional Greengrass components can be added as needed to expand the device's functionality.

4.7 Schematic Diagram

4.7.1 T-Cobbler and LEDs

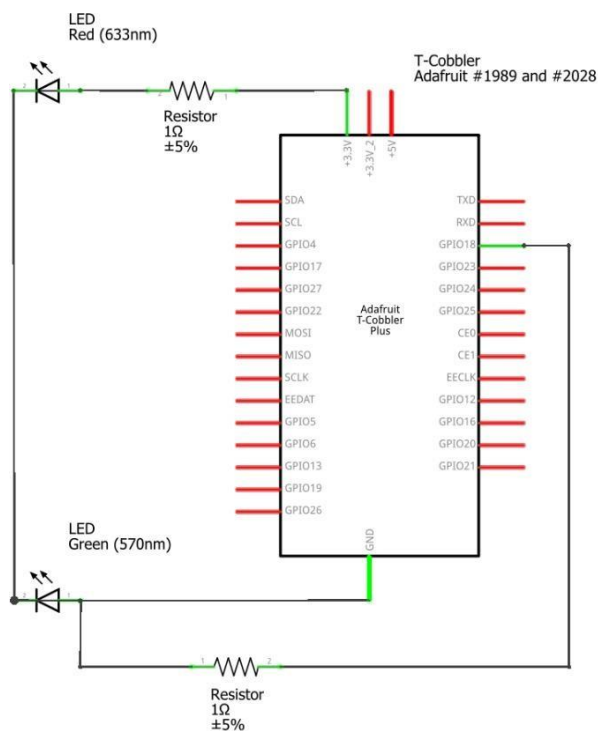


Figure 4.10: Schematic Diagram of T-Cobbler and LEDs

Figure 4.9 displays two LEDs connected to the T-Cobbler, serving a simple function within this IoT device as indicators of its operation. The red LED is connected to the 3.3V pin and serves as a straightforward power indicator, illuminating when the IoT device is receiving power. The green LED is connected to GPIO 18 pin and activates when the sensors begin reading data.

4.7.2 pH Reading Module

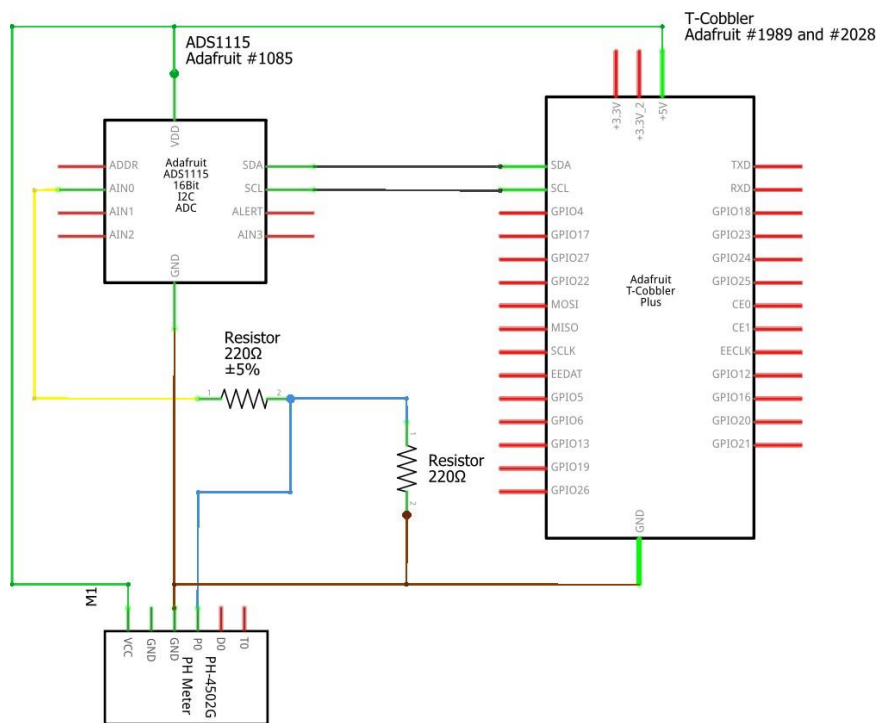


Figure 4.11: Schematic Diagram of pH Reading Module

Diagram above depicts the pH reading module with PH4502C sensor and ADS1115 ADC. The PH4502C sensor generates an analog voltage signal that varies based on the pH level of the water being measured. Conversely, Raspberry Pi GPIO pins are digital pins, capable of reading only binary values (0 or 1) or high and low voltage levels. Therefore, the ADS1115 ADC is used to convert the analog voltage into a digital value that the Raspberry Pi can comprehend and process. Both the PH4502C and ADS1115 require a connection to the 5V pin. The P0 pin of the PH4502C sensor outputs the analog signal, which is then directed to the AIN0 pin on the ADS1115 ADC for conversion. Additionally, the ADS1115 features SDA and SCL pins, which are integral components of the I2C (Inter-Integrated Circuit) communication interface, enable communication with the Raspberry Pi.

4.7.3 Temperature Reading Module

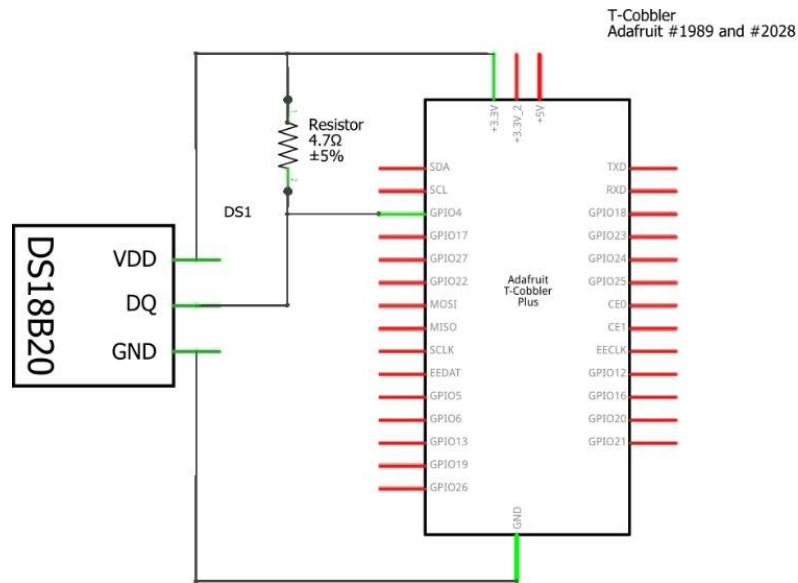


Figure 4.12: Schematic Diagram of Temperature Reading Module

The diagram above shows the DS18B20 waterproof sensor connected to a T-Cobbler. The sensor requires a 3.3V voltage to function. The data pin on the DS18B20 is connected to GPIO 4. This choice is because the DS18B20 uses the 1-Wire protocol to communicate with the Raspberry Pi. GPIO 4 on a Raspberry Pi is configured to support the 1-Wire protocol by default, which makes it possible to connect the DS18B20 sensor without having to manually configure the GPIO pin for 1-Wire communication.

CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

4.8 Project Timeline

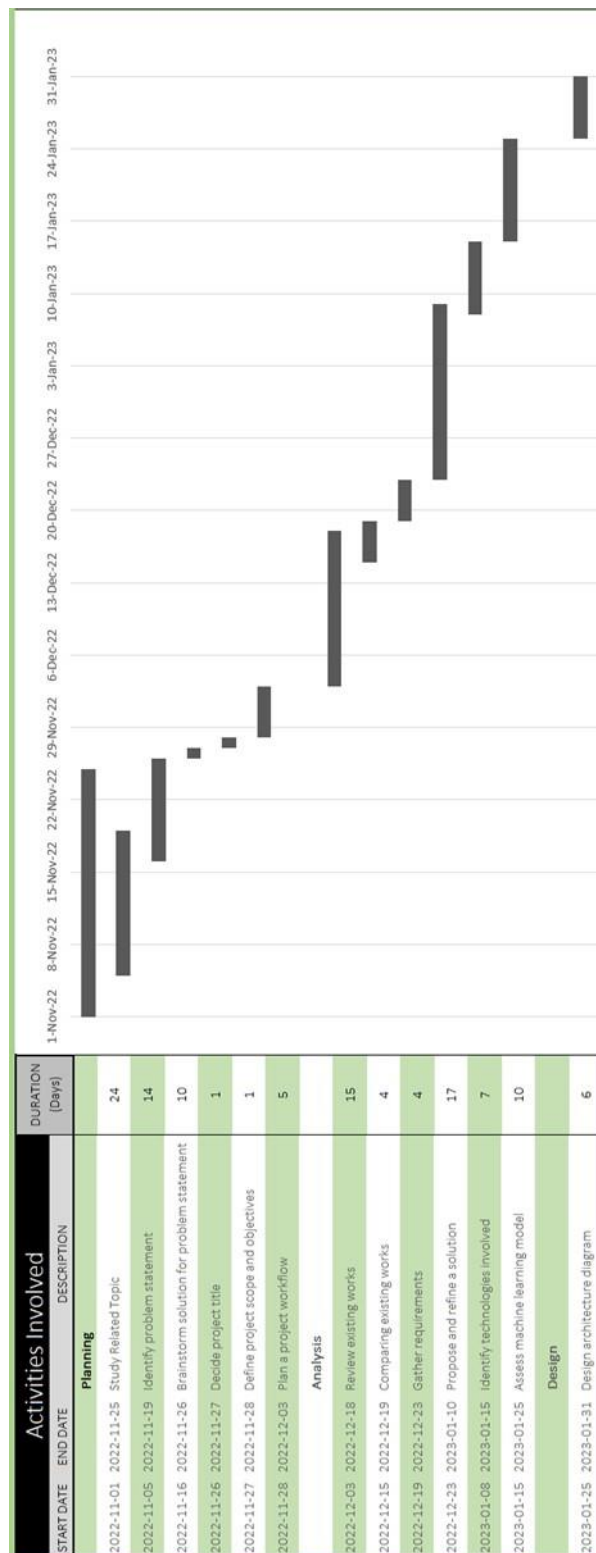


Figure 4.13: Gantt Chart (1)

CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

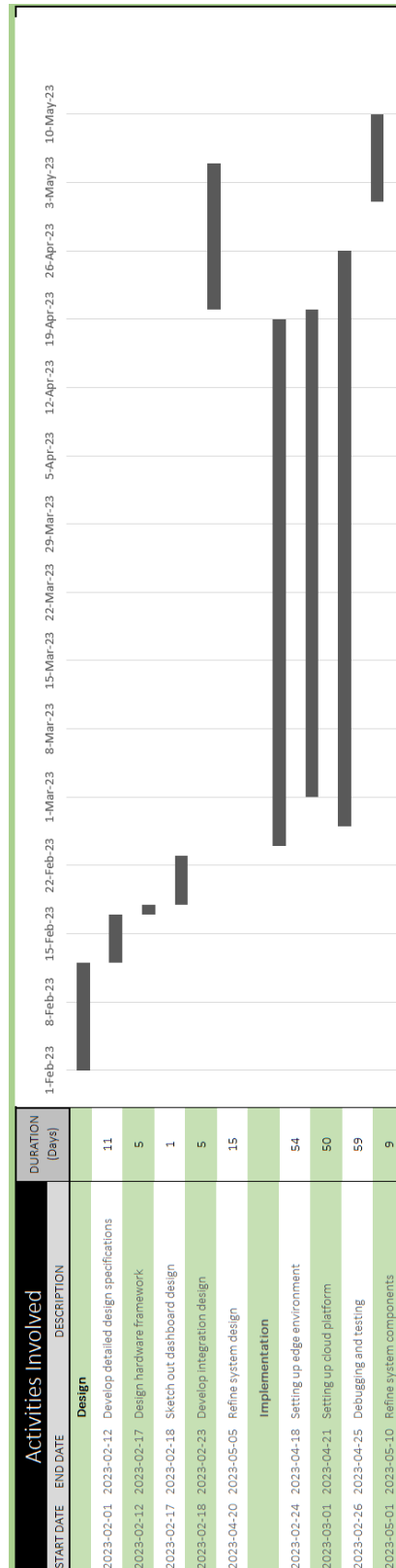


Figure 4.14: Gantt Chart (2)

CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

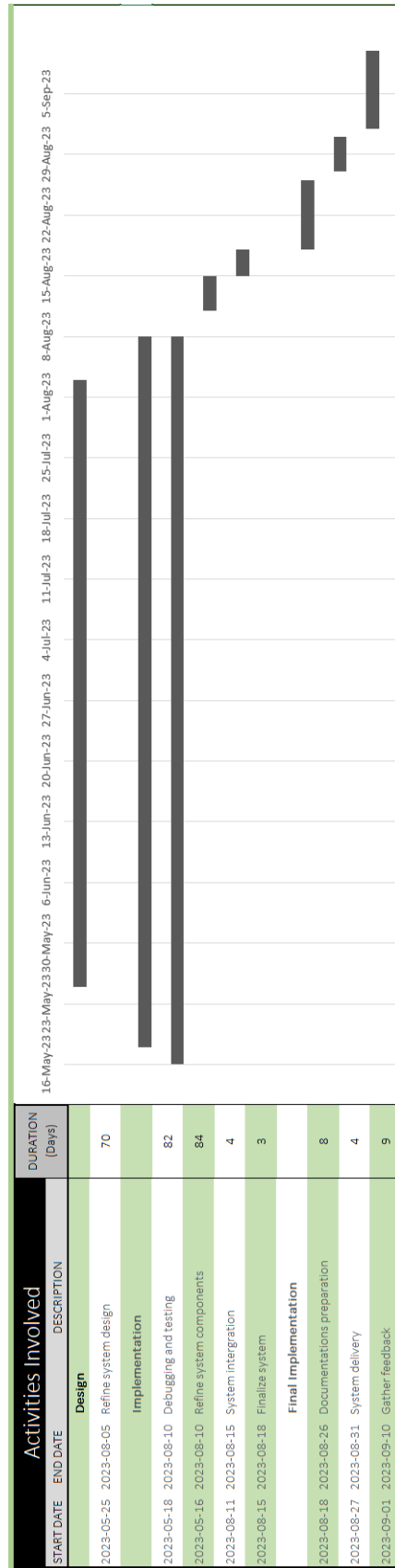


Figure 4.15: Gantt Chart (3)

CHAPTER 5: System Implementation

5.1 Initial Set-up

Before starting to develop an IoT-Cloud Solution for Precision Aquaculture, all the necessary software is installed on personal laptop and the tools involved have been acquired prior the development process:

1. Putty use to enable VNC Server on Raspberry Pi.
2. VNC Viewer to control Raspberry Pi using laptop.
3. An AWS root account has registered.
4. A user is created following AWS best practices with “AdministratorAccess” that have the access to all AWS services.
5. An AWS QuickSight account is registered on user account.
6. Jupyter Notebook installed.
7. All the hardware and electronic components has procured and received.

5.2 Raspberry Pi Setup (Edge Device)

Prior starting the project, below is the abstract description of setting up the Raspberry Pi to ensure that the edge environment work as intended.

1. Raspberry Pi OS Debian Bullseye 64-bit (Release: 2023-05-03) has installed on Raspberry Pi and successfully booted.
2. Java installed on Raspberry Pi as it required to install AWS IoT Greengrass core software.
3. Cgroup reversion from v2 to v1 as required to run AWS Lambda function locally.
4. Install the necessary library uses to control devices and sensors.
5. Enable 1-Wire and I2C interface on raspberry pi configuration panel as require for the sensor involved.
6. TensorFlow are installed as our primary machine learning library.
7. Sqlite3 installed as the local database.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.2.1 Raspberry Pi Setup Command

Below is the command require to run before deployment of the AWS Greengrass components.

1. Verify the java version and install it.
 - java -version
 - sudo apt install default-jdk
2. Cgroup reversion from v2 to v1.
 - findmnt -lo source,target,fstype,options -t cgroup,cgroup2

```
pi@raspberrypi:~ $ findmnt -lo source,target,fstype,options -t cgroup,cgroup2
SOURCE TARGET          FSTYPE OPTIONS
cgroup2 /sys/fs/cgroup cgroup2 rw,nosuid,nodev,noexec,relatime,nsdelegate,memory
```

Figure 5.1: CLI displaying cgroup v2

- cat /boot/cmdline.txt

```
pi@raspberrypi:~ $ cat /boot/cmdline.txt
console=serial0,115200 console=tty1 root=PARTUUID=f9d6eda3-02 rootfstype=ext4 fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consolespi@raspberryp
```

Figure 5.2: CLI displaying content boot configuration file

- sudo sed -i -e "1 s/\$/ cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0/" /boot/cmdline.txt
- cat /boot/cmdline.txt

```
pi@raspberrypi:~ $ cat /boot/cmdline.txt
console=serial0,115200 console=tty1 root=PARTUUID=f9d6eda3-02 rootfstype=ext4 fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0pi@raspberrypi:~ $
```

Figure 5.3: CLI displaying updated content boot configuration file

- sudo reboot
- findmnt -lo source,target,fstype,options -t cgroup,cgroup2

```
pi@raspberrypi:~ $ findmnt -lo source,target,fstype,options -t cgroup,cgroup2
SOURCE TARGET          FSTYPE OPTIONS
cgroup2 /sys/fs/cgroup/unified cgroup2 rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/systemd cgroup rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/memory cgroup rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/freezer cgroup rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/cpu,cpuacct cgroup rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/pids cgroup rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/perf_event cgroup rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/net_cls,net_prio cgroup rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/cpuset cgroup rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/devices cgroup rw,nosuid,nodev,noexec,relatime,
cgroup /sys/fs/cgroup/blkio cgroup rw,nosuid,nodev,noexec,relatime,
```

Figure 5.4: CLI displaying cgroup v1

CHAPTER 5 SYSTEM IMPLEMENTATION

3. Install library for DS18B20 temperature sensor.
 - sudo pip3 install Adafruit_DHT
 - sudo pip3 install w1thermsensor
 - sudo raspi-config
 - Navigate to "Interfacing Options" and select "1-Wire."
 - Choose "Yes" to enable the One-Wire interface.
4. Install library for PH4502C pH sensor
 - sudo pip3 install adafruit-circuitpython-ads1x15
 - sudo raspi-config
 - Navigate to "Interfacing Options" and select "I2C."
 - Choose "Yes" to enable the I2C interface
5. Reboot Raspberry Pi for the changes in step 3 and 4 to take effect.
 - Sudo reboot
6. Install TensorFlow
 - Install the TensorFlow version compatible to the Debian Bullseye 64-bit OS.
 - Verify the installation of TensorFlow on Raspberry Pi.

```
pi@raspberrypi:~ $ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.__version__
'2.10.0'
```

Figure 5.5: CLI displaying TensorFlow version installed

7. Install Sqlite3
 - sudo apt update
 - sudo apt full-upgrade
 - sudo apt install sqlite3

5.3 Setting Up Hardware

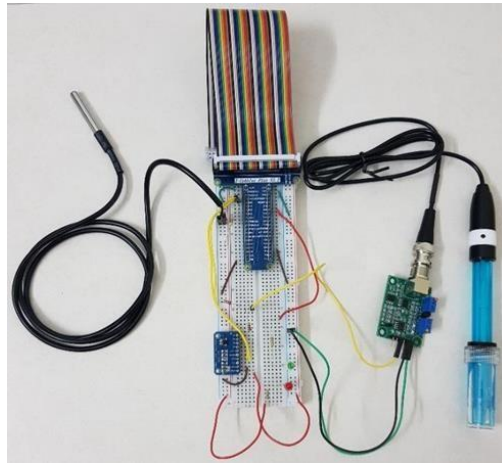


Figure 5.6: Hardware Implementation Based on Breadboard

Figure 5.6 represent the hardware implementation based on a breadboard for this edge environment, the breadboard is the foundation for all the electronic components and IoT devices used. The sensors involved in this implementation to obtain real time water parameters is DS18B20 waterproof temperature sensor and PH4502C pH sensor. The output of the PH4502C pH sensor is analog signal, which must be converted to a digital signal before it can be read and processed by Raspberry Pi. Therefore, an ADS1115 is used to convert analog signal from the pH sensor and converts it into a digital signal that can be read and interpreted by Raspberry Pi. Furthermore, two LED are used to indicate the activity of the edge environment, red LED indicate the breadboard receive power and green LED represent the sensors are active. Finally, a T-Cobbler is an extension from the 40-pin GPIO on Raspberry Pi.

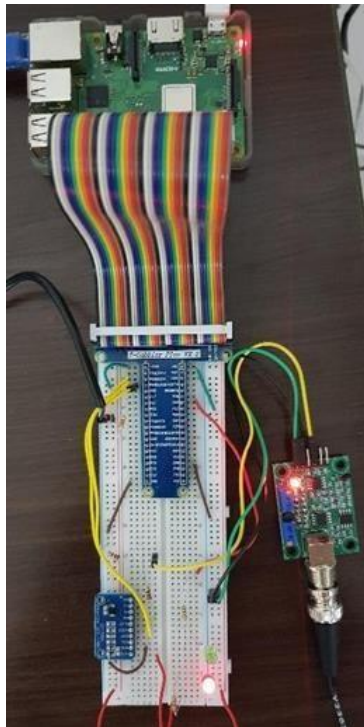


Figure 5.7: IoT Devices Receiving Power

The hardware implementation from figure 5.6 will be connected to a Raspberry Pi, which will serve as the Greengrass core device. The Greengrass core device enables the control of local IoT devices, local processing, messaging, and data caching even without an internet connection. It will also be responsible for communicating with AWS Cloud using the MQTT protocol. Figure 5.7 represents the edge environment, which consists of the Greengrass core device (the Raspberry Pi) and various other devices that are connected to it. The red LED on some devices indicate that the edge environment is active.

5.4 pH Sensor Calibration

The following calibration process is an essential step in setting up PH4502C sensor as it ensures accurate and reliable pH measurements. The process involved adjusting the voltage offset of PH4502C, mapping the pH reading to output voltage, calculate the slope and intercept and finally evaluate the pH reading.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.4.1 Adjusting the Sensor Voltage Offset

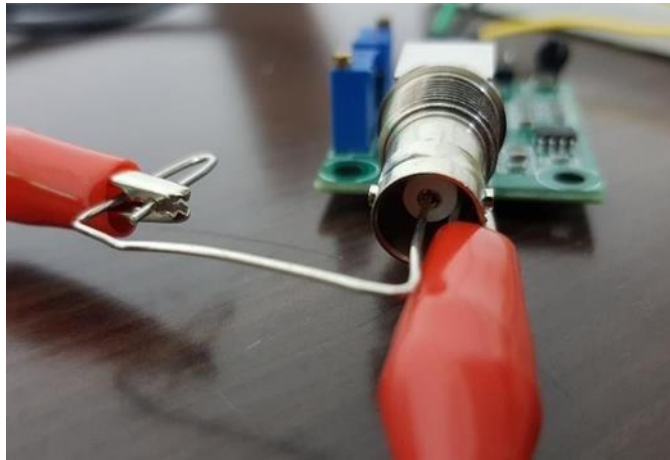


Figure 5.8: BNC Connector with Paper Clip Inserted and Alligator Clip Connected to the Outer Metal Casing of BNC Connector

The idea behind this step is to force the pH sensor to output a constant voltage of 7.0, which corresponds to a neutral pH reading. This is achieved by shorting the BNC connector using a metal paper clip and connecting alligator clips to the paper clip and the outer metal casing of the BNC connector as shown on Figure 5.8.

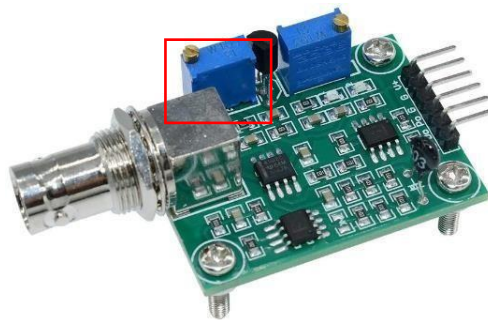


Figure 5.9: The Highlighted Area on Reference Electrode is Potentiometer

Once the BNC have been shorted, we dial the pH sensor to 2.5 volts by adjusting the potentiometer on the reference electrode module. Figure 5.10 below indicate the Python code use to read the output voltage, we stop adjusting the potentiometer when it reached 2.5 volts.

CHAPTER 5 SYSTEM IMPLEMENTATION

```
<untitled> x pHCalibrate.py x
1 import board
2 import busio
3 import time
4 import sys
5 import adafruit_ads1x15.ads1115 as ADS
6 from adafruit_ads1x15.analog_in import AnalogIn
7
8 # Setup
9 i2c = busio.I2C(board.SCL, board.SDA)
10 ads = ADS.ADS1115(i2c)
11
12 def read_voltage(channel):
13     while True:
14         buf = list()
15
16         for i in range(10): # Take 10 samples
17             buf.append(channel.voltage)
18         buf.sort() # Sort samples and discard highest and low
19     --
```

```
Shell
ADS1115 channel 0-3: 0
Adjust potentiometer nearest to BNC socket to ~2.50V
Starting readings. Press CTRL+C to stop...
2.5 V
2.5 V
2.5 V
2.5 V
2.5 V
```

Figure 5.10: The Highlighted Area on Reference Electrode is Potentiometer

5.4.2 Mapping the pH Readings to Output Voltage



Figure 5.11: pH 4.01 and pH 7.00 Buffer Powder for Buffer Solution

In this step we need to map the reported voltages from PH4502C to two pH measurements. Thus, I've prepared two pH buffer solutions using pH buffer powder, specifically pH 4.01 and pH 7.00 shown on figure 5.11 as they are commonly used as the pH measurements. Now we can replace the alligator clips with pH probe.

CHAPTER 5 SYSTEM IMPLEMENTATION



Figure 5.12: pH 4.01 and pH 7.00 Buffer Solution with pH Probe Immerse in It

After connecting the pH probe with the BNC, we can start to record the output voltage of the pH sensor in the pH 4.01 and pH 7.00 buffer solution. The pH probe is immersed in both buffer solution respectively for some time allowing the pH sensor to stabilize before reading the output voltage. It is demonstrated in figure 5.12 as the pH probe is placed in pH 7.00 buffer solution with the pH 4.01 buffer solution on the left.

```
4 import sys
5 import adafruit_ads1x15_ads1115 as ADS
6 from adafruit_ads1x15_analog_in import AnalogIn
7
8 # Setup
9 i2c = busio.I2C(board.SCL, board.SDA)
10 ads = ADS.ADS1115(i2c)
11
12 def read_voltage(channel):
13     while True:
14         buf = list()
15
16         for i in range(10): # Take 10 samples
17             buf.append(channel.voltage)
18         buf.sort() # Sort samples and discard high
19         buf = buf[2:-2]
20         avg = (sum(map(float,buf))/6) # Get average
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

pH	Output Voltage (V)
4.01	3.06
4.01	3.06
4.01	3.06
4.01	3.06
4.01	3.06
4.01	3.06
4.01	3.06
4.01	3.06
4.01	3.06
4.01	3.06
7.00	2.55
7.00	2.55
7.00	2.55
7.00	2.55
7.00	2.55
7.00	2.55
7.00	2.55
7.00	2.55
7.00	2.55
7.00	2.55

Figure 5.13: The Output Voltage of pH Probe from pH 4.01 and pH 7.00 Buffer Solution (Left to Right)

Figure 5.13 depicts the output voltage reading for both buffer solution. On the left, the voltage reading for pH 4.01 buffer solution stabilize on 3.06 volts while voltage for pH 7.00 buffer solution recorded remain uniform on 2.55 volts. In conclusion, pH 4.01 reported 3.06 volts (3.06, 4.01) and pH 7.00 reported 2.55 volts (2.55, 7.00).

CHAPTER 5 SYSTEM IMPLEMENTATION

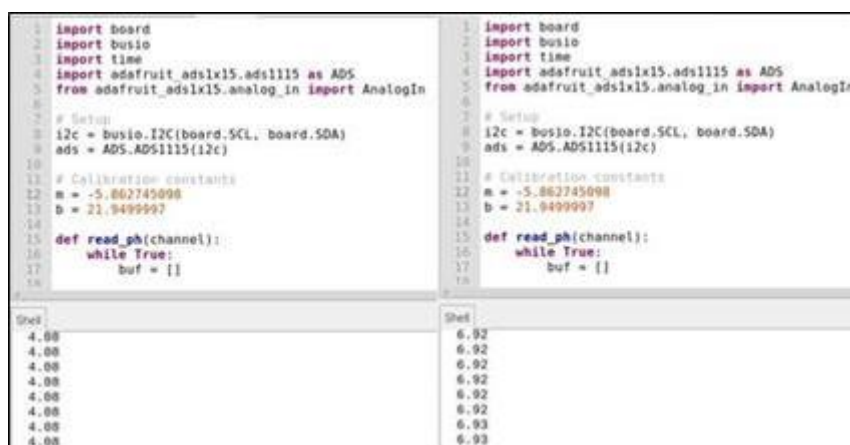
5.4.3 Calculate the Slope and Intercept

The slope and intercept are the calibration coefficients used in the calculation of pH value from the voltage output of a pH electrode probe. pH electrodes probe generates a small electrical potential in response to the concentration of hydrogen ions in a solution. This potential is measured by a pH meter as a voltage output, which is then converted into pH units using the calibration coefficients.

$$\begin{aligned}\text{Slope} &= (\text{pH2} - \text{pH1}) / (\text{mV2} - \text{mV1}) \\ &= (7.00 - 4.01) / (2.55 - 3.06) \\ &= 2.99 / (-0.51) \\ &= -5.862745098\end{aligned}$$

$$\begin{aligned}\text{Intercept} &= \text{pH1} - \text{Slope} \times \text{mV1} \\ &= 4.01 - (-5.862745098)(3.06) \\ &= 21.9499997\end{aligned}$$

5.4.4 Evaluate the pH Readings



```
1 import board
2 import busio
3 import time
4 import adafruit_ads1x15_ads1115 as ADS
5 from adafruit_ads1x15_analog_in import AnalogIn
6
7 # Setup
8 i2c = busio.I2C(board.SCL, board.SDA)
9 ads = ADS.ADS1115(i2c)
10
11 # Calibration constants
12 m = -5.862745098
13 b = 21.9499997
14
15 def read_ph(channel):
16     while True:
17         buf = []
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Shot	Shot
4.08	6.92
4.08	6.92
4.08	6.92
4.08	6.92
4.08	6.92
4.08	6.92
4.08	6.92
4.08	6.93
4.08	6.93

Figure 5.14: The pH reading of pH sensor from pH 4.01 and pH 7.00 Buffer Solution (Left to Right)

The final step is to ensure the pH measurement are accurate and satisfy our requirements. A Python code is written with the formula below to read pH value from the sensor. Both pH 4.01 and pH 7.00 buffer solution are reused to measure the result of pH readings. From left to right on figure 5.14, the pH reading for pH 4.01 and pH 7.00 buffer solution are reported at pH 4.08

CHAPTER 5 SYSTEM IMPLEMENTATION

and pH 6.93 respectively. This result satisfies the requirement as the PH4502C having $\pm 0.1\text{pH}@25^\circ\text{C}$ accuracy.

$$\begin{aligned}\text{pH} &= \text{slope} \times (\text{mV at calibration point}) + \text{intercept} \\ &= -5.862745098 \times (\text{mV at calibration point}) + 21.9499997\end{aligned}$$

5.5 Setting Up Edge Environment

5.5.1 AWS IoT Greengrass

AWS IoT Greengrass is one of the feature of AWS IoT Core, it allows us to manage our edge environment via an Greengrass core device. In our case, Raspberry Pi will be the Greengrass core device with Greengrass core software installed on it. In the current implementation, the Greengrass core device are implemented with several Greengrass components that serves different function in this project. These components granting the ability to the Greengrass core device to function autonomously, reducing its dependence on network connectivity and cloud resources. Functions such as sensor data collection, preprocessing, and local storage within a database as well as rule-based algorithms to trigger alarms can be operate without network connectivity. When a stable network connection becomes available, the Greengrass core device synchronizes data with a cloud-based database and also engage in federated learning processes using locally pre-processed data with cloud-based servers. In the future, new custom Greengrass components can be deploy as needed to expand and enhance the capabilities of our edge environment to adapt the future requirement.

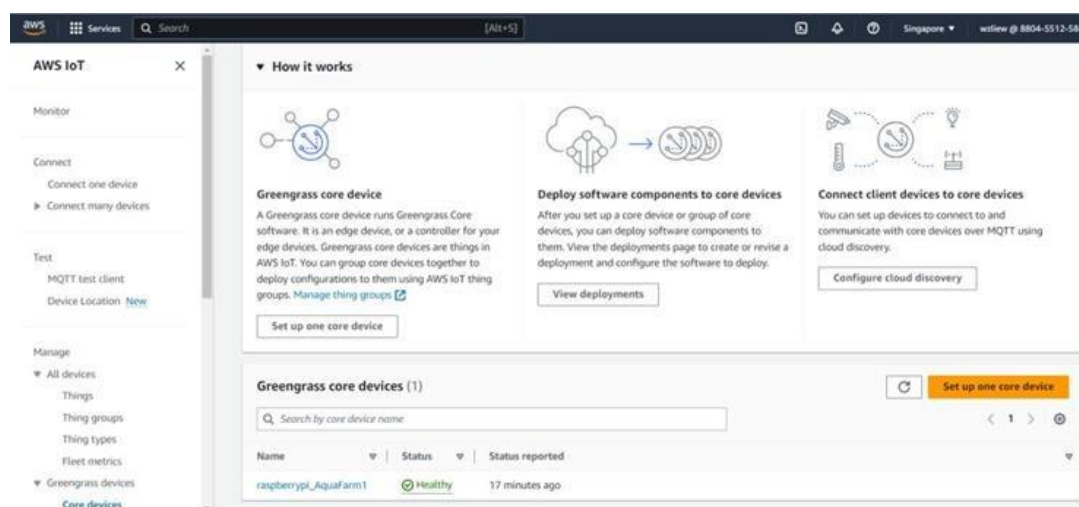


Figure 5.15: Raspberry Pi Registered as Greengrass Core Device


```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo systemctl status greengrass
● greengrass.service - Greengrass Core
   Loaded: loaded (/etc/systemd/system/greengrass.service; enabled; vendor pr
   Active: active (running) since Wed 2023-04-19 22:17:07 +08; 4h 41min ago
   Main PID: 489 (sh)
   Tasks: 58 (limit: 1596)
   CPU: 25min 15.907s
   CGroup: /system.slice/greengrass.service
           └─ 489 /bin/sh /greengrass/v2/alts/current/distro/bin/loader
              └─ 495 java -Dlog.store=FILE -Dlog.store=FILE -Droot=/greengrass/v
                 └─ 1278 sh -c python3 -u /greengrass/v2/packages/artifacts/com.exam
                    └─ 1281 python3 -u /greengrass/v2/packages/artifacts/com.example.se

Apr 19 22:17:07 raspberrypi systemd[1]: Started Greengrass Core.
Apr 19 22:17:08 raspberrypi sh[489]: Greengrass root: /greengrass/v2
Apr 19 22:17:08 raspberrypi sh[489]: Java executable: java
Apr 19 22:17:08 raspberrypi sh[489]: JVM options: -Dlog.store=FILE -Droot=/gree
Apr 19 22:17:08 raspberrypi sh[489]: Nucleus options: --setup-system-service fa
Apr 19 22:17:24 raspberrypi sh[495]: Launching Nucleus...
Apr 20 02:34:10 raspberrypi sh[495]: Launched Nucleus successfully.
lines 1-19/19 (END)
    
```

Figure 5.16: Raspberry Pi CLI Indicating Greengrass Core Software are Installed on It

Figure 5.15 displays there’s a Greengrass core device name “raspberrypi_AquaFarm1” registered in AWS IoT Core, it is in fact the Raspberry Pi. To set the Raspberry Pi as Greengrass core device, we just need to follow the instruction provided step by step upon clicking the “Set up one core device” button in the core devices page. After completing the steps, the Raspberry Pi will be shown in the Greengrass core devices list. From Raspberry Pi CLI like figure 5.16, it indicates the Greengrass core software are installed in Raspberry Pi.

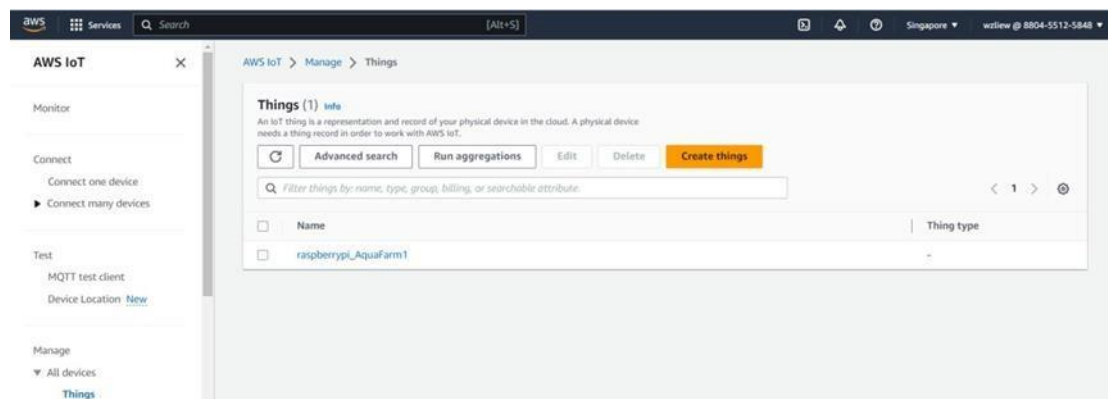


Figure 5.17: Greengrass Core Device as a IoT Thing

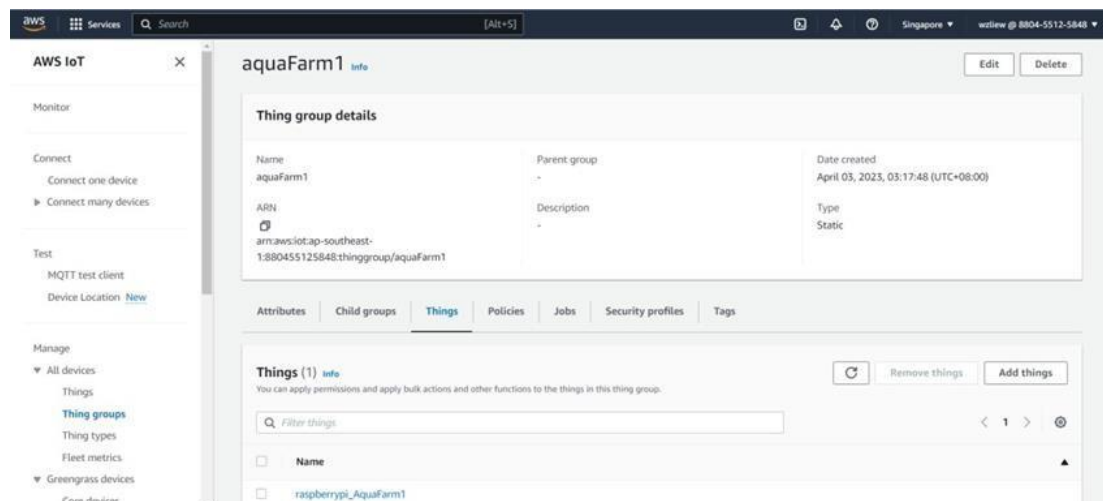


Figure 5.18: Greengrass Core Device inside IoT Thing Group

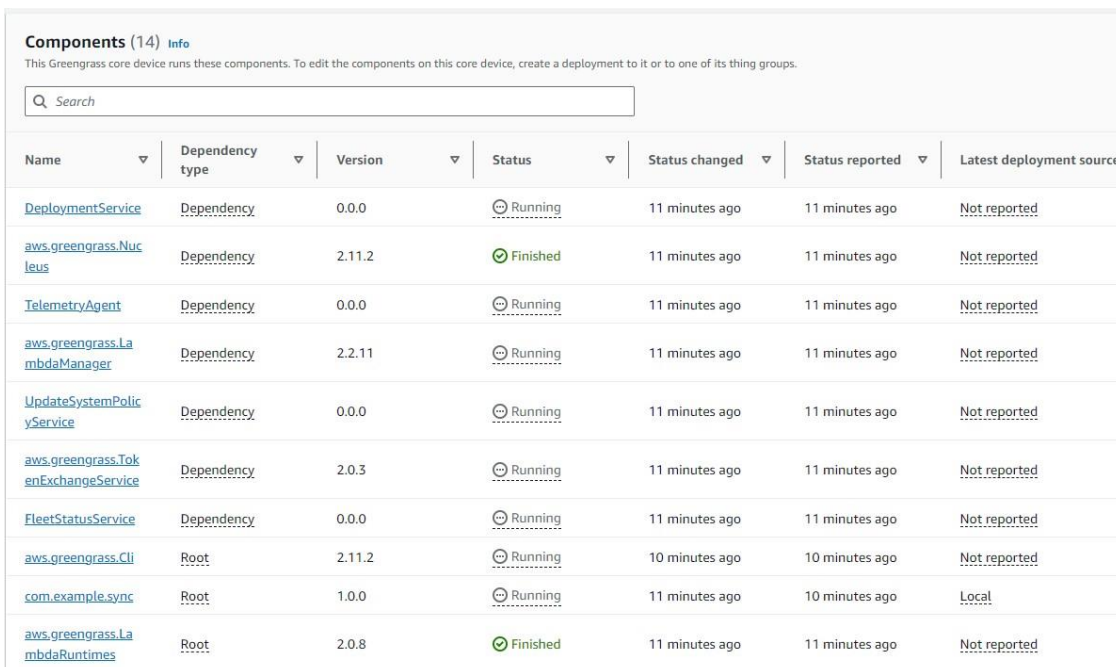
Based on Figure 5.17 and Figure 5.18, the Raspberry Pi is also registered as a Thing in AWS IoT Core and is included in a Thing Group. Since a Greengrass core device is also considered an IoT device, it is automatically added to the Thing list when registered as Greengrass core device. If a Thing Group is specified during the "Set up one core device" process, the Raspberry Pi will be added to that Thing Group. Thing Group allow the management of multiple devices as a single entity and apply policies and configurations across all devices in the group.

5.5.2 AWS Greengrass Component

The components of AWS Greengrass are software modules that run on a Greengrass Core Device. AWS provides several public components that are commonly used on the core device. AWS Greengrass also allows users to develop their own custom components based on their specific requirements. Both public components and custom components will be use for the implementation. One of the custom Greengrass components on the core device is a sensor reading component. This component's primary function is to gather sensor data, perform preprocessing, and store the processed data into local database. Simultaneously, the sensor reading component also directing the sensor data the sensor to a rule-based alarm component and trigger alert if the water parameters exceed predefined thresholds. Additionally, a data synchronization component is implemented to sync local data with cloud database. Furthermore, a federated learning client will also be integrated as a Greengrass component.

CHAPTER 5 SYSTEM IMPLEMENTATION

In AWS IoT Greengrass, a component is comprised of two parts, namely the **artifact** and the **recipe**. The **artifact** is a package that contains the code and dependencies for the component ready to be deployed or distributed. On the other hand, the **recipe** is a YAML or JSON file that describes the configuration of the component. It specifies the name and version of the component, the artifact to use, and any dependencies on other components.



Name	Dependency type	Version	Status	Status changed	Status reported	Latest deployment source
DeploymentService	Dependency	0.0.0	Running	11 minutes ago	11 minutes ago	Not reported
aws.greengrass.Nucleus	Dependency	2.11.2	Finished	11 minutes ago	11 minutes ago	Not reported
TelemetryAgent	Dependency	0.0.0	Running	11 minutes ago	11 minutes ago	Not reported
aws.greengrass.LambdaManager	Dependency	2.2.11	Running	11 minutes ago	11 minutes ago	Not reported
UpdateSystemPolicyService	Dependency	0.0.0	Running	11 minutes ago	11 minutes ago	Not reported
aws.greengrass.TokenExchangeService	Dependency	2.0.3	Running	11 minutes ago	11 minutes ago	Not reported
FleetStatusService	Dependency	0.0.0	Running	11 minutes ago	11 minutes ago	Not reported
aws.greengrass.Cli	Root	2.11.2	Running	10 minutes ago	10 minutes ago	Not reported
com.example.sync	Root	1.0.0	Running	11 minutes ago	10 minutes ago	Local
aws.greengrass.LambdaRuntimes	Root	2.0.8	Finished	11 minutes ago	11 minutes ago	Not reported

Figure 5.19: Component List of raspberrypi_AquaFarm1

When the Greengrass core software is installed, some components are automatically deployed on the Greengrass core device. However, deployment can also be managed through the AWS CLI or the AWS Management Console. A deployment is a process that deploys a list of components to multiple devices at once. Figure 5.19 shows the list of components installed through different deployment methods on the raspberrypi_AquaFarm1, it includes both public components and custom components.

CHAPTER 5 SYSTEM IMPLEMENTATION

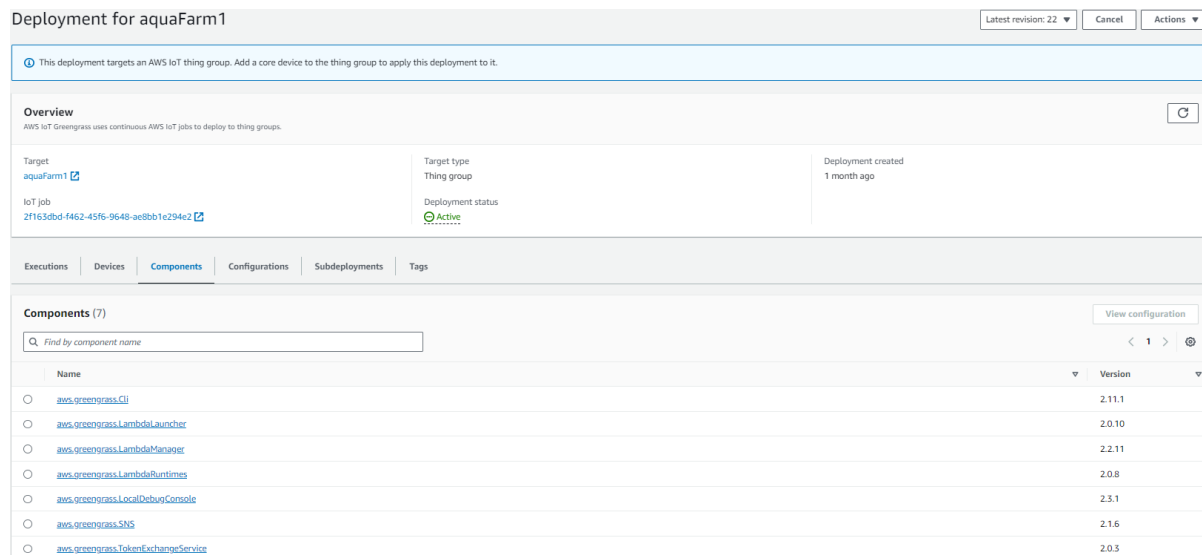


Figure 5.20: Deployment for raspberrypi_AquaFarm1

In Figure 5.20, the deployment of public components using the AWS Management Console is depicted. These public components are prerequisites for the subsequent deployment of custom components. Also, some of the public components may have dependencies on other public components.

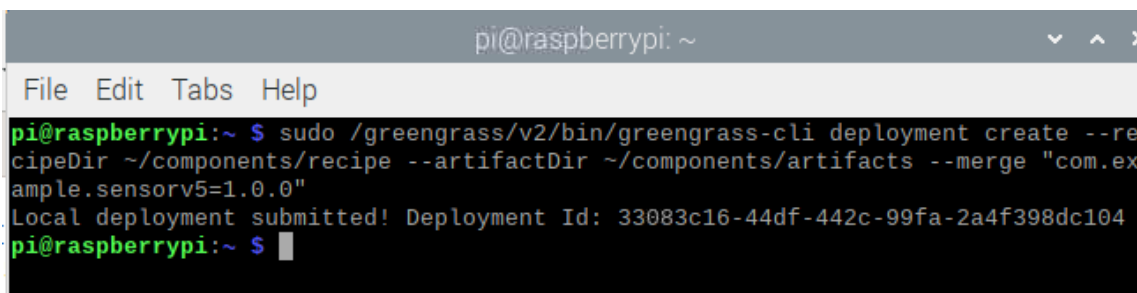


Figure 5.21: Raspberry Pi CLI Command to Deploy Custom Component

While Figure 5.21 displays the CLI command to deploy a custom component named “com.example.sensorv5” directly on Raspberry Pi. The component artifact and recipe are located inside the Greengrass core device, this command just simply deploy it.

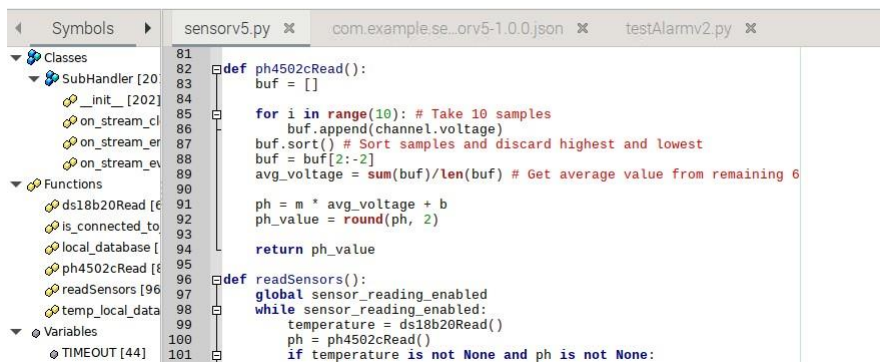
CHAPTER 5 SYSTEM IMPLEMENTATION

```
pi@raspberrypi:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Components currently running in Greengrass:
Component Name: TelemetryAgent
  Version: 0.0.0
  State: RUNNING
  Configuration: null
Component Name: UpdateSystemPolicyService
  Version: 0.0.0
  State: RUNNING
  Configuration: null
Component Name: aws.greengrass.TokenExchangeService
  Version: 2.0.3
  State: RUNNING
  Configuration: {"activePort":38657,"port":0.0}
Component Name: com.example.sensorv5
  Version: 1.0.0
  State: RUNNING
  Configuration: {"accessControl":{"aws.greengrass.ipc.mqttproxy":{"com.example.sensorv5:mqttproxy:1":{"operations":["aws.greengrass.PublishToIoTCore","aws.greengrass.SubscribeToIoTCore"],"policyDescription":"Allows access to pub/sub to aquaFarm2/sensor","resources":["aquaFarm2/sensor","aquaFarm2/switch"]}},aws.greengrass.ipc.pubsub":{"com.example.MyLocalPubSubComponent:pubsub:1":{"operations":["aws.greengrass.PublishToTopic"],"policyDescription":"Allows access to publish/subscribe to all topics","resources":["loc/sensor"]}},"message":"switch"}}
Component Name: aws.greengrass.Nucleus
  Version: 2.11.2
  State: FINISHED
  Configuration: {"awsRegion":"ap-southeast-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFrequencySeconds":"15","envStage":"prod","fleetStatus":{"periodicStatusPublishIntervalSeconds":86400.0},"greengrassDataPlaneEndpoint":"","greengrassDataPlanePort":"8443","httpClient":{"","iotCoreEndpoint":"c2vm61188brj30.credentials.iot.ap-southeast-1.amazonaws.com","iotDataEndpoint":"abfknoded83ou-sts.iot.ap-southeast-1.amazonaws.com","iotRoleAlias":"GreengrassV2TokenExchangeRoleAlias"},"jvmOptions":{"-Dlog.store=FILE","logging":{"","matt":{"spooler":{"},"networkProxy":{"proxy":{"},"platformOverride":{"},"runWithDefault":{"posixShell":"sh","posixUser":"ggc_user:ggc_group"},"telemetry":{"}}
Component Name: aws.greengrass.Cli
  Version: 2.11.2
  State: RUNNING
  Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":null}
Component Name: aws.greengrass.LambdaRuntimes
  Version: 2.0.0
  State: FINISHED
  Configuration: {}
Component Name: DeploymentService
  Version: 0.0.0
  State: RUNNING
```

Figure 5.22: Raspberry Pi CLI Command to Display Component List

After the component are deployed successfully, the “com.example.sensorv5” are listed in the component list as depicted in figure 5.22 with status “RUNNING”. In some case, it is possible the component is in “BROKEN” state, this mean the component need to be fixed before being deploy again. The components list on the Raspberry Pi CLI are same as the components list shown on Figure 5.19 on AWS Management Console. The subsequent custom components involved will be deploy in the same way.

5.5.2.1 Sensor reading and processing component (com.example.sensorv5)



```
81
82
83 def ph4502cRead():
84     buf = []
85     for i in range(10): # Take 10 samples
86         buf.append(channel.voltage)
87         buf.sort() # Sort samples and discard highest and lowest
88         buf = buf[2:-2]
89         avg_voltage = sum(buf)/len(buf) # Get average value from remaining 6
90
91     ph = m * avg_voltage + b
92     ph_value = round(ph, 2)
93
94     return ph_value
95
96 def readSensors():
97     global sensor_reading_enabled
98     while sensor_reading_enabled:
99         temperature = ds18b20Read()
100         ph = ph4502cRead()
101         if temperature is not None and ph is not None:
```

Figure 5.23: A Snap of Artifact for Component com.example.sensorv5

CHAPTER 5 SYSTEM IMPLEMENTATION



Figure 5.24: A Snap of Recipe for Component com.example.sensorv5

To gather water parameter readings from the local sensors and execute the required preprocessing tasks, we developed a custom Greengrass component named “com.example.sensorv5”. Figure 5.23 provides a brief overview of the artifact related to this component, while Figure 5.24 displays the recipe associated with it.

```
def ds18b20Read():
    sensor_file = None
    for file in os.listdir('/sys/bus/w1/devices/'):
        if file.startswith('28-'):
            sensor_file = os.path.join('/sys/bus/w1/devices', file, 'w1_slave')
            break

    if sensor_file is None:
        print("Error: Sensor file not found")
        return None
    else:
        with open(sensor_file, 'r') as file:
            sensor_data = file.read()

        # Parse the temperature reading from the raw data
        temperature_line = sensor_data.split("\n")[1]
        temperature = float(temperature_line.split(" ")[9].strip()[2:]) / 1000

    return temperature
```

Figure 5.25: DS18B20 Data Processing Function

```

def ph4502cRead():
    buf = []

    for i in range(10): # Take 10 samples
        buf.append(channel.voltage)
    buf.sort() # Sort samples and discard highest and lowest
    buf = buf[2:-2]
    avg_voltage = sum(buf)/len(buf) # Get average value from remaining 6

    ph = m * avg_voltage + b
    ph_value = round(ph, 2)

    return ph_value

```

Figure 5.26: PH4502C Data Processing Function

The primary function of this component is to read raw data from sensors and preprocess it into a human-readable format at a 10-second interval, achieving this by employing the two functions outlined below. In figure 5.25, the function responsible for processing raw data from the DS18B20 temperature sensor, which involves extracting the temperature element from the raw data and converting it from millidegrees Celsius to degrees Celsius. On the other hand, for the pH readings obtained from the PH4502C sensor, the previously acquired slope (m) and intercept (b) values from the pH calibration process are used to calculate the pH value, as demonstrated in figure 5.26.

```

def local_database(data):
    conn = sqlite3.connect('sensor_data.db')
    cursor = conn.cursor()

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS sensor_data (
            device_id TEXT,
            timestamp TEXT,
            temperature DECIMAL,
            ph_value DECIMAL,
            PRIMARY KEY (device_id, timestamp)
        )
    ''')

    cursor.execute('''
        INSERT INTO sensor_data (device_id, timestamp, temperature, ph_value)
        VALUES (?, ?, ?, ?)
    ''', (data['device_id'], data['timestamp'], data['temperature'], data['ph_value']))

    logging.info("successfully commit in database")

    db_filename = 'sensor_data.db'
    db_path = os.path.abspath(db_filename)
    print(f"The sensor database file will be created at: {db_path}")
    logging.info(f"The sensor database file will be created at: {db_path}")

    conn.commit()
    conn.close()

```

Figure 5.27: Local Database Function

CHAPTER 5 SYSTEM IMPLEMENTATION

```
def temp_local_database(data):
    conn = sqlite3.connect('temp_sensor_data.db') # Use a separate temporary database
    cursor = conn.cursor()

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS temp_sensor_data (
            device_id TEXT,
            timestamp TEXT,
            temperature DECIMAL,
            ph_value DECIMAL,
            PRIMARY KEY (device_id, timestamp)
        ''')

    cursor.execute('''
        INSERT INTO temp_sensor_data (device_id, timestamp, temperature, ph_value)
        VALUES (?, ?, ?, ?)
    ''', (data['device_id'], data['timestamp'], data['temperature'], data['ph_value']))

    conn.commit()
    conn.close()
```

Figure 5.28: Temporary Database Function

Subsequently, the processed temperature and pH values are added with additional data, such as a timestamp and a unique ID representing the Greengrass core device, before being stored in a local database. In figure 5.27, a function used to store processed data into the local database. In cases where network connectivity is unavailable, an additional function, as shown in figure 5.28, is executed to store data in a local temporary database.

When network connectivity is established, the processed data will only store in the local main database. Additionally, the processed data is transformed into JSON format for transmission via the Pub/Sub service, enabling storage in a cloud-based database. This message package is subsequently published to their respective topic, which in this case is “aquaFarm1/sensor”. To activate the sensor within the code, this component subscribes to the topic "aquaFarm1/switch," where it receives messages in JSON format that instruct the sensor to turn on or off. The communication between this component and AWS IoT Core is facilitated through the Pub/Sub service of the MQTT protocol, which involves subscribing to and publishing messages to specific topics.

Furthermore, the JSON messages are also published to the “loc/sensor” topic, a local Pub/Sub interface used for communication between components, irrespective of the network connection. Messages published to the "loc/sensor" local topic will be utilized by other components concurrently.

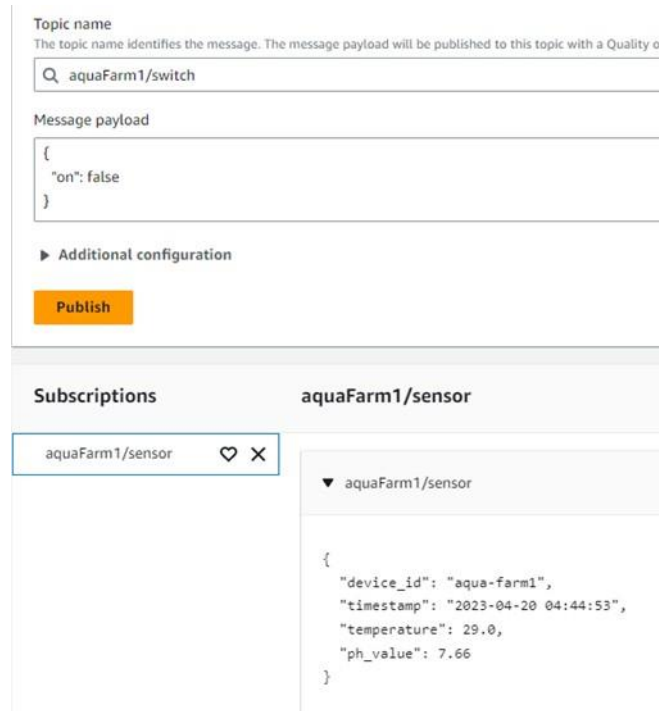


Figure 5.29: AWS IoT Core MQTT Test Client

Figure 5.29 shows the usage of MQTT Test Client in AWS IoT Core to test the communication between edge environment and AWS IoT Core. We first subscribe to the topic “aquaFarm1/sensor” and publish a JSON message to topic “aquaFarm1/switch” to activate the sensor reading. The sensor will read the water parameter immediately and the result are shown on the console. At the same time, the green LED on the hardware will be turn on as well, indicating the sensor are actively reading data.

5.5.2.2 Rule-Based Alert Component (com.example.testAlarmv2)

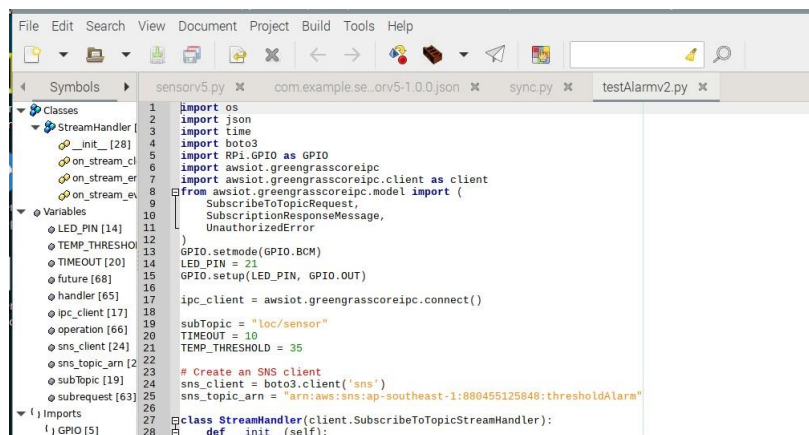
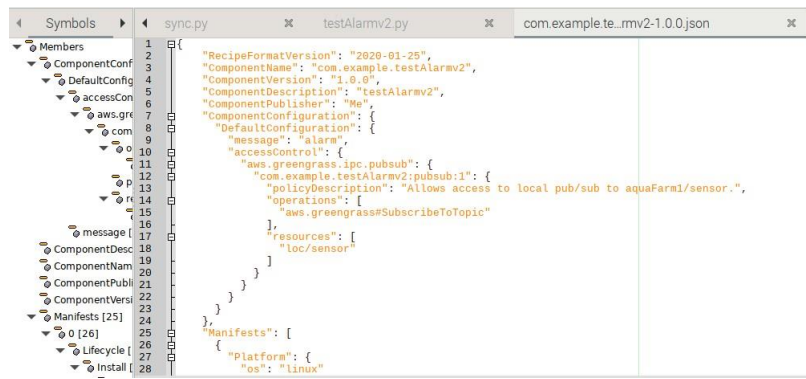


Figure 5.30: A Part of Artifact for Component com.example.testAlarmv2

CHAPTER 5 SYSTEM IMPLEMENTATION



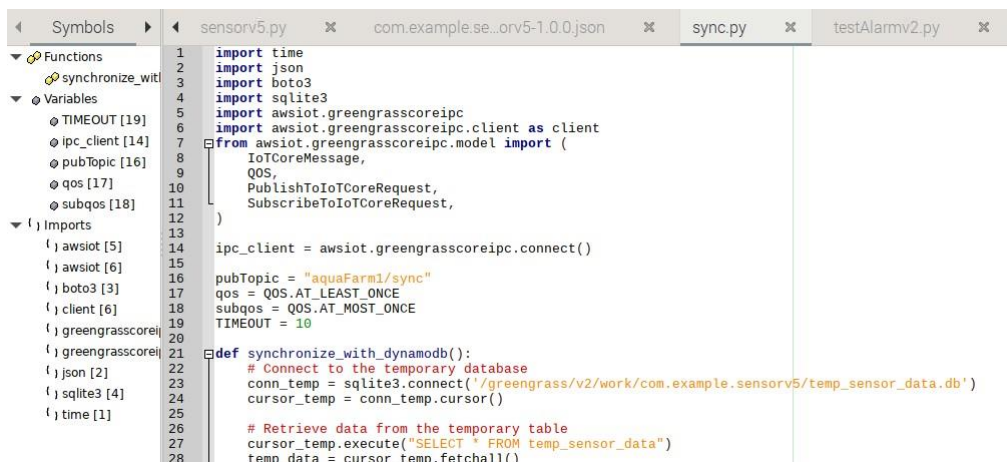
```
1 {
2   "RecipeFormatVersion": "2020-01-25",
3   "ComponentName": "com.example.testAlarmv2",
4   "ComponentVersion": "1.0.0",
5   "ComponentDescription": "testAlarmv2",
6   "ComponentPublisher": "me",
7   "ComponentConfiguration": {
8     "DefaultConfiguration": {
9       "message": "alarm",
10      "accessControl": {
11        "aws.iot.core:publish": {
12          "com.example.testAlarmv2:pubsub:1": {
13            "policyDescription": "Allows access to local pub/sub to aquaFarm1/sensor.",
14            "operations": [
15              "aws.iot.core:SubscribeToTopic"
16            ],
17            "resources": [
18              "loc/sensor"
19            ]
20          }
21        }
22      }
23    },
24    "Manifests": [
25      {
26        "Platform": {
27          "os": "linux"
28        }
29      }
30    ]
31  }
```

Figure 5.31: A Part of Recipe for Component com.example.testAlarmv2

Figure 5.30 and figure 5.31 above displays the artifact and recipe of “com.example.testAlarmv2”, respectively. This component operates concurrently with the “com.example.sensorv5” component, and its purpose is to evaluate the processed water parameters by subscribing to the local topic “loc/sensor”, which is published by the “com.example.sensorv5” component. The messages received from the subscribed topic in JSON format are parsed to extract only the temperature and pH values. These values are then checked against predefined thresholds to determine if they exceed acceptable levels. In the event that the water parameters breach these thresholds, an alarm is triggered and sent to the user via AWS SNS.

To facilitate this functionality, the “com.example.testAlarmv2” component utilizes the boto3 library, which enables direct API requests to AWS services, allowing it to send alarms to the user through the SNS topic previously created.

5.5.2.3 Data Synchronization Component (com.example.sync)



```
1 import time
2 import json
3 import boto3
4 import sqlite3
5 import awsiot.greengrasscoreipc
6 import awsiot.greengrasscoreipc.client as client
7 from awsiot.greengrasscoreipc.model import (
8     IoTCoreMessage,
9     QOS,
10    PublishToIoTCoreRequest,
11    SubscribeToIoTCoreRequest,
12 )
13
14 ipc_client = awsiot.greengrasscoreipc.connect()
15
16 pubTopic = "aquaFarm1/sync"
17 qos = QOS.AT_LEAST_ONCE
18 subqos = QOS.AT_MOST_ONCE
19 TIMEOUT = 10
20
21 def synchronize_with_dynamodb():
22     # Connect to the temporary database
23     conn_temp = sqlite3.connect('/greengrass/v2/work/com.example.sensorv5/temp_sensor_data.db')
24     cursor_temp = conn_temp.cursor()
25
26     # Retrieve data from the temporary table
27     cursor_temp.execute("SELECT * FROM temp_sensor_data")
28     temp_data = cursor_temp.fetchall()
```

Figure 5.32: A Portion of Artifact for Component com.example.sync

CHAPTER 5 SYSTEM IMPLEMENTATION

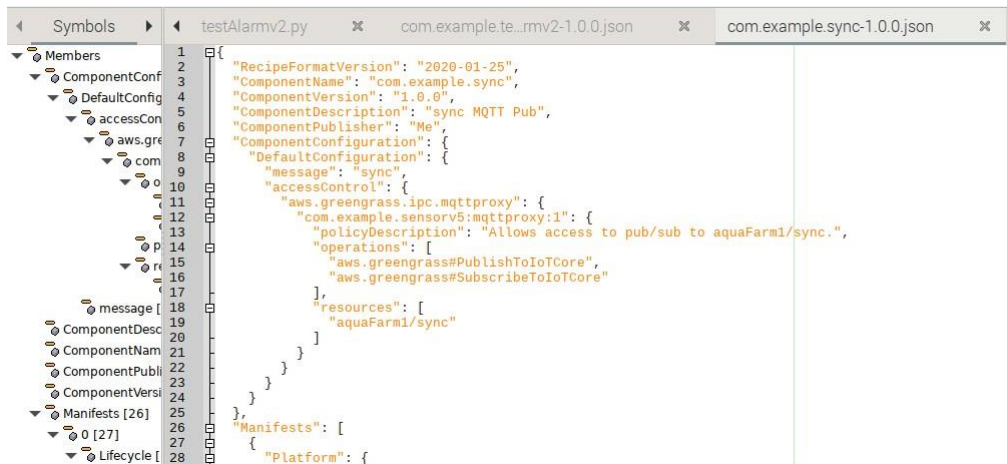


Figure 5.33: A Portion of Recipe for Component com.example.sync

In the absence of network connectivity, the processed data is stored in both a local main database and a temporary database. This component is activated at 5-minute intervals to perform data synchronization with the cloud database when a network connection becomes available. Figures 5.32 and 5.33 provide snapshots of the artifact and recipe of “com.example.sync”, respectively. This component makes use of both the MQTT protocol and the boto3 library.

When the network connection is restored, this component first checks whether the temporary database is empty. If it contains data, the component begins comparing each individual row in the temporary database with the data stored in DynamoDB. Since each row is unique, the absence of a row in DynamoDB indicates that the processed data was not properly stored due to the lack of an internet connection. In such cases, the row is converted to JSON format and published to the “aquaFarm1/sync” topic, then sent to AWS IoT Core. Subsequently, every row in the temporary database is deleted, regardless of whether it already exists in DynamoDB.

5.5.2.4 Federated Learning Client (com.example.flwrClient2)

```

118 try:
119     s3_client.download_file(bucket, key, temp_file_path)
120     print(f"File download from S3 bucket: {bucket}, key: {temp_file_path}")
121 except Exception as e:
122     print(f"Error download file from S3: {str(e)}")
123
124 return temp_file_path
125
126 class numpyClient(fl.client.NumPyClient):
127     def __init__(self, X_train, y_train):
128         self.X_train = X_train
129         self.y_train = y_train
130
131     def get_parameters(self, config):
132         print(f"(config)Numpy_Get_Parameter")
133         bucket = 'flwr-testbucket'
134         prefix = f"parameters/weights/{client_id}params.pkl"
135         p = local_get_parameters()
136
137         # Use a temporary file to store the model weights
138         try:
139             with tempfile.NamedTemporaryFile(dir='/tmp', delete=False) as fp:
140                 pickle.dump(p, fp)
141                 #logger.info(f"Dumped parameters to : {fp.name}")
142                 upload_to_s3(bucket, prefix, fp.name.replace('data', 'tmp'))
143         except Exception as e:
144             print(f"Error get_parameters: {str(e)}")
145

```

Figure 5.34: A Part of Artifact for Federated Learning Client Component com.example.flwrClient2

```

1 {
2   "RecipeFormatVersion": "2020-01-25",
3   "ComponentName": "com.example.flwrClient2",
4   "ComponentVersion": "1.0.0",
5   "ComponentDescription": "flwr client 2",
6   "ComponentPublisher": "Me",
7   "ComponentConfiguration": {
8     "DefaultConfiguration": {
9       "message": "flwr"
10    }
11  },
12  "Manifests": [
13    {
14      "Platform": {
15        "os": "linux"
16      },
17      "Lifecycle": {
18        "Install": {
19          "RequiresPrivilege": true,
20          "script": "python3 -m pip install --user flwr && python3 -m pip install --user pandas &&"
21        },
22        "Run": {
23          "RequiresPrivilege": true,
24          "script": "python3 -u {artifacts:path}/flwrClient2.py '{configuration:message}'"
25        }
26      }
27    }
28  ]
}

```

Figure 5.35: A Part of Recipe for Federated Learning Client Component com.example.flwrClient2

Federated learning is conducted through a cloud-based server and several edge environments, each equipped with its own federated learning client, as illustrated in Figures 5.34 and 5.35, representing part of the artifact and recipe of the client. The federated learning client relies on three main libraries, namely TensorFlow, Flower, and boto3.

The client implements subclasses of Flower's “flwr.client.NumPyClient” which, while easier to implement, offers less flexibility. These subclasses define how local training and evaluation are executed and enable the federated learning server to initiate the local training process. Local ML model training, which involves a Recurrent Neural Network (RNN) from TensorFlow, utilizes data stored locally in the local database. The boto3 library is employed to interact with AWS S3, enabling the retrieval of global ML model weights and the storage of local ML model weights in the cloud.

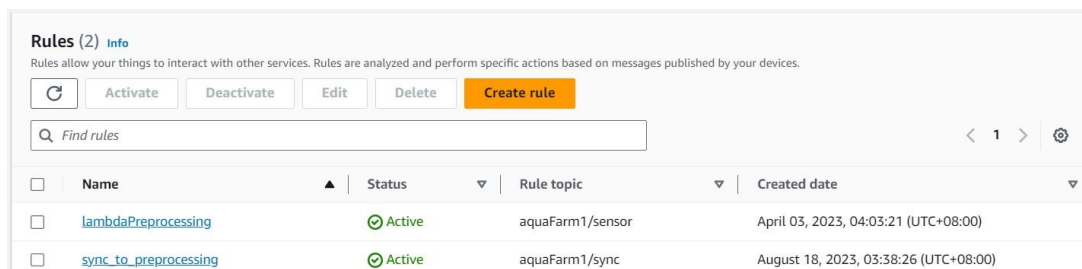
CHAPTER 5 SYSTEM IMPLEMENTATION

This federated learning client maintains a similarity across multiple edge environments and communicates with the server responsible for orchestrating the federated learning process, using the gRPC protocol. The client is initialized by specifying the previously implemented subclass and the server's DNS. More detailed insights into the entire federated learning process will be provided in the dedicated section.

5.6 Setting Up AWS Cloud

5.6.1 AWS IoT Core

AWS IoT Core serves as the central communication hub connecting any Greengrass core device to the AWS Cloud. It offers an interface equipped with various features and capabilities to facilitate IoT device management, data processing, and communication. The previously registered core devices and deployments are managed through this interface. In this implementation, AWS IoT Core takes on additional responsibilities, such as message routing, to efficiently channel sensor data sent by core devices to other AWS services. Consequently, IoT rules named “lambdaPreprocessing” and “sync_to_preprocessing” have been created to direct messages from core devices to specific lambda functions for further processing and synchronization.



The screenshot shows the AWS IoT Core Rules console. At the top, there are buttons for 'Activate', 'Deactivate', 'Edit', 'Delete', and 'Create rule'. Below these is a search bar labeled 'Find rules'. A table lists two rules:

<input type="checkbox"/>	Name	Status	Rule topic	Created date
<input type="checkbox"/>	lambdaPreprocessing	Active	aquaFarm1/sensor	April 03, 2023, 04:03:21 (UTC+08:00)
<input type="checkbox"/>	sync_to_preprocessing	Active	aquaFarm1/sync	August 18, 2023, 03:38:26 (UTC+08:00)

Figure 5.36: IoT Rules for Message Routing



The screenshot shows the AWS IoT Core Actions console. It displays a single action configuration:

Service	Action
Lambda	Send a message to a Lambda function

Figure 5.37: The Action Indicating Lambda Function in IoT Rule

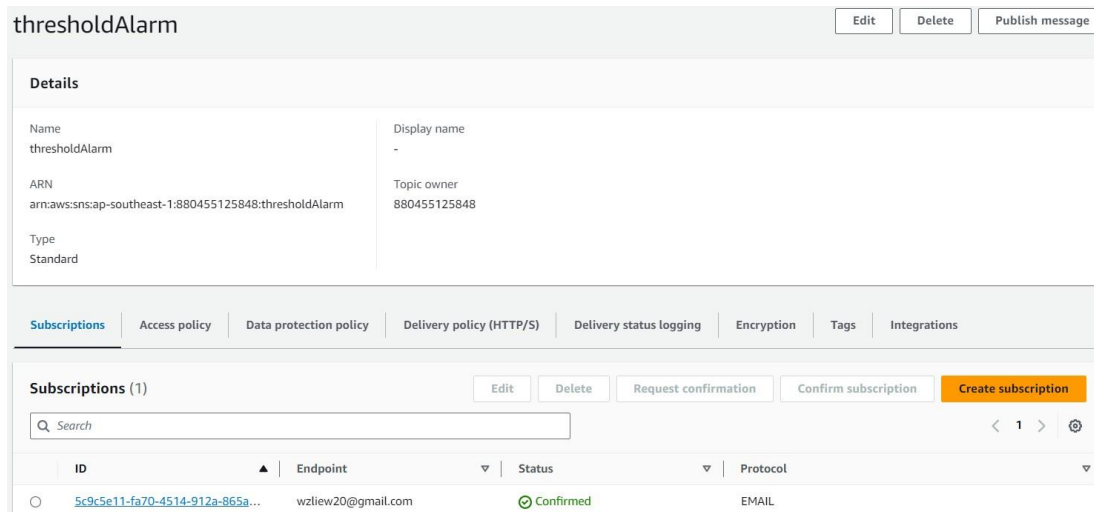
CHAPTER 5 SYSTEM IMPLEMENTATION

Based on Figure 5.36, both IoT Rules, “lambdaPreprocessing” and “sync_to_preprocessing” subscribe to the topics “aquaFarm1/sensor” and “aquaFarm1/sync”, respectively. These topics are published by the Greengrass core device, and both IoT Rules able receive message when there is a network connection on the core device.

The action taken by these IoT Rules is to route the message to an Amazon Lambda function for further processing, as illustrated in Figure 5.37, where the action is indicated as “Lambda”. The specific Lambda function to which the IoT Rules direct the message must be defined, and in this case, “intermediateProcessing” has been chosen for this purpose.

5.6.2 Amazon SNS

Amazon SNS is a fully managed messaging service that enables the sending of messages or notifications to a distributed group of recipients through various delivery methods, such as email, SMS, and other channels. In this implementation, an SNS topic will be established to act as the destination for the “com.example.testAlarmv2” component on core devices, allowing it to send messages to alert users when water parameters are breached.



The screenshot displays the Amazon SNS console interface for a topic named "thresholdAlarm". At the top, there are buttons for "Edit", "Delete", and "Publish message". Below this is a "Details" section with the following information:

Name	thresholdAlarm	Display name	-
ARN	arn:aws:sns:ap-southeast-1:880455125848:thresholdAlarm	Topic owner	880455125848
Type	Standard		

Below the details are tabs for "Subscriptions", "Access policy", "Data protection policy", "Delivery policy (HTTP/5)", "Delivery status logging", "Encryption", "Tags", and "Integrations". The "Subscriptions (1)" section is active, showing a search bar and a table of subscriptions:

ID	Endpoint	Status	Protocol
5c9c5e11-fa70-4514-912a-865a...	wzliw20@gmail.com	Confirmed	EMAIL

Figure 5.38: Detail of SNS Topic

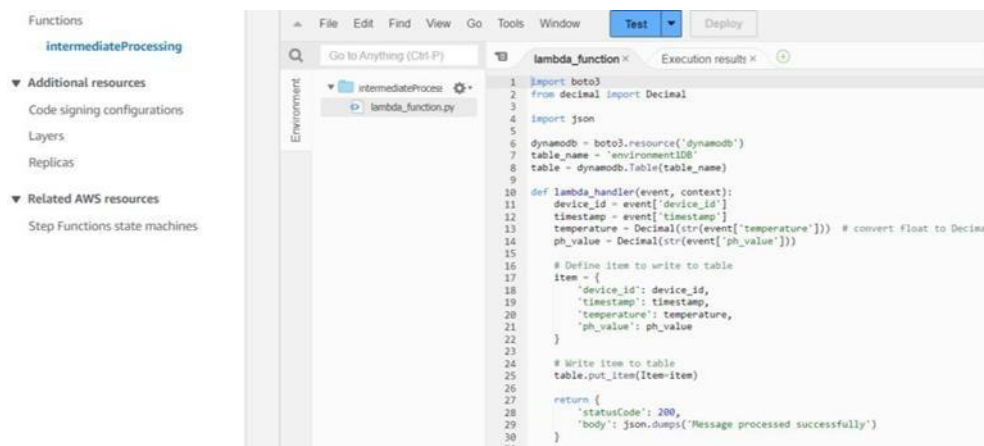
Figure 5.38 displays the content of the “thresholdAlarm” topic along with a list of subscriptions below. These subscriptions serve as the recipients who will receive messages, and in this case, email is the chosen message protocol with a specified endpoint. Therefore, whenever the “com.example.testAlarmv2” component on a core device needs to send a message to trigger an

CHAPTER 5 SYSTEM IMPLEMENTATION

alarm to the user, it utilizes the boto3 library to directly invoke this service with this specific topic.

5.6.3 Amazon Lambda

Amazon Lambda is a serverless computing service, it allows code to be run in response to an event. The following Lambda function are written in Python 3.9 runtime and deployed to process the incoming data from IoT Rules.



```
1 import boto3
2 from decimal import Decimal
3
4 import json
5
6 dynamodb = boto3.resource('dynamodb')
7 table_name = 'environment108'
8 table = dynamodb.Table(table_name)
9
10 def lambda_handler(event, context):
11     device_id = event['device_id']
12     timestamp = event['timestamp']
13     temperature = Decimal(str(event['temperature'])) # convert float to Decimal
14     ph_value = Decimal(str(event['ph_value']))
15
16     # Define item to write to table
17     item = {
18         'device_id': device_id,
19         'timestamp': timestamp,
20         'temperature': temperature,
21         'ph_value': ph_value
22     }
23
24     # Write item to table
25     table.put_item(Item=item)
26
27     return {
28         'statusCode': 200,
29         'body': json.dumps('Message processed successfully')
30     }
```

Figure 5.39: Lambda Function intermediateProcessing

Figure 5.39 illustrates the structure and function of the “intermediateProcessing” AWS Lambda function. This particular Lambda function is responsible for processing messages received from IoT rules and storing them in a specified database in Amazon DynamoDB. The input message to this Lambda function is a JSON object containing information about the water parameters obtained from sensors. The function then processes this input message by splitting it into individual items that represent columns in the DynamoDB database. This ensures that the data is structured in a way that can be easily visualized on a dashboard. Once the message has been processed and the data has been formatted correctly, the Lambda function then inserts the data into the specified DynamoDB database.

5.6.4 Amazon DynamoDB

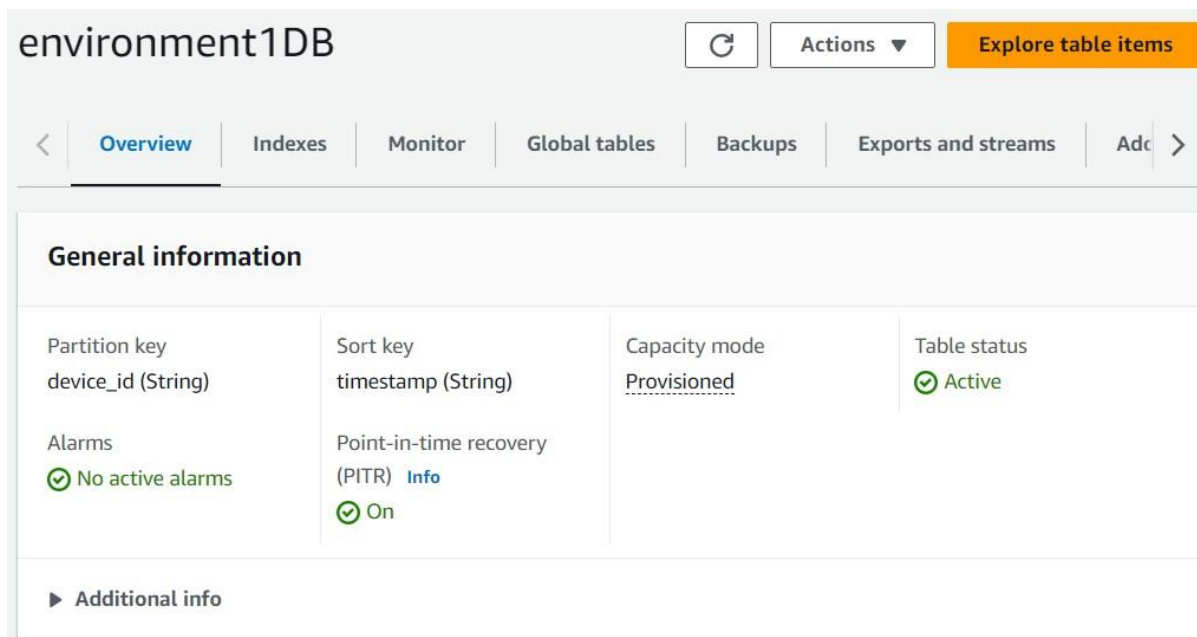


Figure 5.40: environment1DB Table Configuration

Amazon DynamoDB was chosen as the database to store processed data due to its fast retrieval speed. The processed data stored in DynamoDB can be used for various purposes, such as real-time data analysis and machine learning. In this proposed system, Amazon QuickSight is used to visualize the data in near real-time, DynamoDB will be the primary data source for it. Figure 5.40 shows the configuration of the “environment1DB” table in database. The device_id is used as the partition key and the timestamp is used as the sort-key, together forming a composite key that uniquely identifies each row in the table. This table are used to store the incoming processed data from Lambda function and ready to be visualize.

5.6.5 Amazon Athena

Amazon Athena is an interactive query service that enables the execution of SQL queries on data stored in various formats. In our implementation, Athena plays a crucial role as an intermediary between DynamoDB and QuickSight. This choice stems from certain challenges posed by DynamoDB when it comes to querying and visualizing NoSQL data directly. DynamoDB serves as our primary data source for visualization and will also function as a backup for the edge environment.

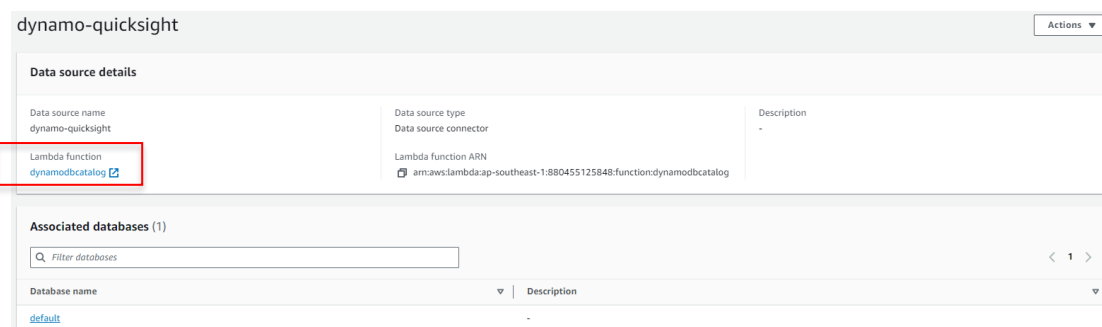


Figure 5.41: Details of Athena Data Source dynamo-quicksight

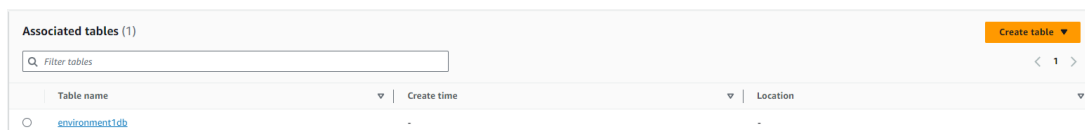


Figure 5.42: Table Specified in Associated Databases

In figure 5.41, we have created an Athena data source named “dynamo-quicksight” to serve as the data source for Amazon QuickSight while querying data from DynamoDB. As highlighted in the figure, the Lambda function labeled “dynamodbcatalog” is an AWS Lambda-DynamoDB connector, utilized to facilitate federated queries from the associated database. Within the associated database, specified in figure 5.42, we have identified the “environment1db” table where our processed data is stored. Consequently, Athena conducts queries on the data contained within the “environment1db” table and subsequently becomes the data source for visualization in QuickSight.

CHAPTER 5 SYSTEM IMPLEMENTATION

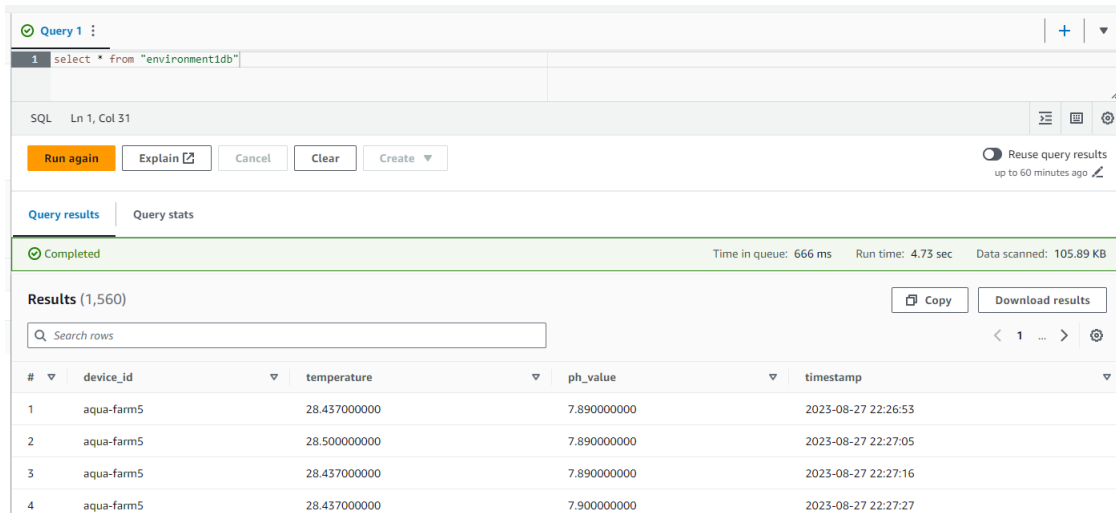


Figure 5.43: Athena Query Editor Testing

The figure above displays the query editor interface of Athena. The query (`SELECT * FROM "environmentldb"`) are ran to ascertain whether the Lambda-DynamoDB connector able to successfully retrieve data from the specified DynamoDB table. The results indicates that the items within the "environmentldb" table were successfully queried by Athena.

5.6.6 Amazon S3

The Amazon S3 is a scalable object storage service that allow wide variety of data, such as files, documents, images, and backups, to be store within buckets. It can be accommodating vast range of use cases. In our specific scenario, it plays a crucial role as an intermediary for the federated learning process, facilitating the exchange of model weights between server and clients.

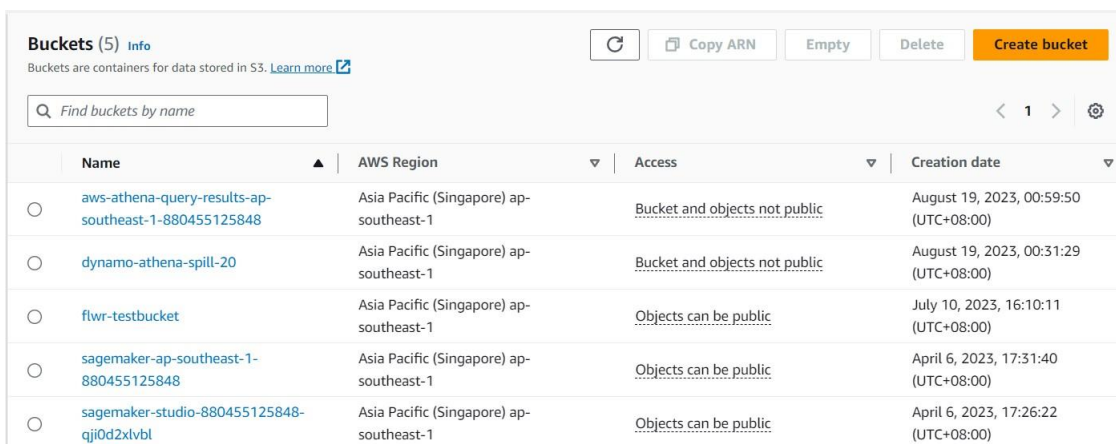


Figure 4.44: Bucket List in S3

CHAPTER 5 SYSTEM IMPLEMENTATION

For our use case, an S3 bucket named “flwr-testbucket” was created with versioning enable specifically for the federated learning process. Bucket versioning configuration on S3 creates a new version of an object when it gets updated, while retaining the old version. In this case, if the global ML model are negatively impacted by client's ML model weights, reversion to a previous version of the model is possible by making rolling back to a known-good state. The other buckets in the list are generated by Athena and SageMaker automatically. The “flwr-testbucket” serves as the central repository for storing both global ML model weights and client ML model weights. It plays a dual role, being utilized by both the server and the client components in the federated learning process.

5.6.7 Amazon Elastic Container Service (Amazon ECS)

Amazon ECS is a container orchestration service that streamline the deployment, management, and scaling of containerized application. The ECS will be used to host a federated learning server that serves as the orchestrator for the federated learning process across various edge environments within ECS cluster. To achieve this, the server source code is containerized using Docker, encapsulating it in a Docker image. After that, the Docker image are push to Amazon Elastic Container Registry (ECR) repository.

flwr_Server:3 Deploy ▾ Actions ▾

Overview [Info](#)

ARN arn:aws:ecs:ap-southeast-1:880455125848:task-definition/flwr_Server:3	Status ACTIVE	Time created 2023-08-06T18:06:10.660Z	App environment FARGATE
Task role ecsTaskExecutionRole	Task execution role ecsTaskExecutionRole	Operating system/Architecture Linux/X86_64	Network mode awsvpc

[Containers](#) | [JSON](#) | [Storage](#) | [Tags](#)

Task size

Task CPU 2 vCPU	Task memory 8 GB
--------------------	---------------------

Containers [Info](#)

Container name	Image	Private registry	Essential	CPU	Memory hard/soft limit	GPU
flwr_Server	880455125848.dkr.e...	-	Yes	0	-/-	-

Figure 5.45: Detail of Task Definition Created from Docker Image

CHAPTER 5 SYSTEM IMPLEMENTATION

The Docker image stored in the ECR repository is used to defining task definitions in ECS. These task definitions include various parameters, including configurations, resource requirements, and networking settings. In figure 5.45, is the detail of task definition named “flwr_Server” along with its version. Additionally, the containers specified below this task definition indicate the specific Docker image it utilizes. This task definition will become our federated learning server with 1 container. Notably, a task definition can accommodate a maximum of 10 different containers.

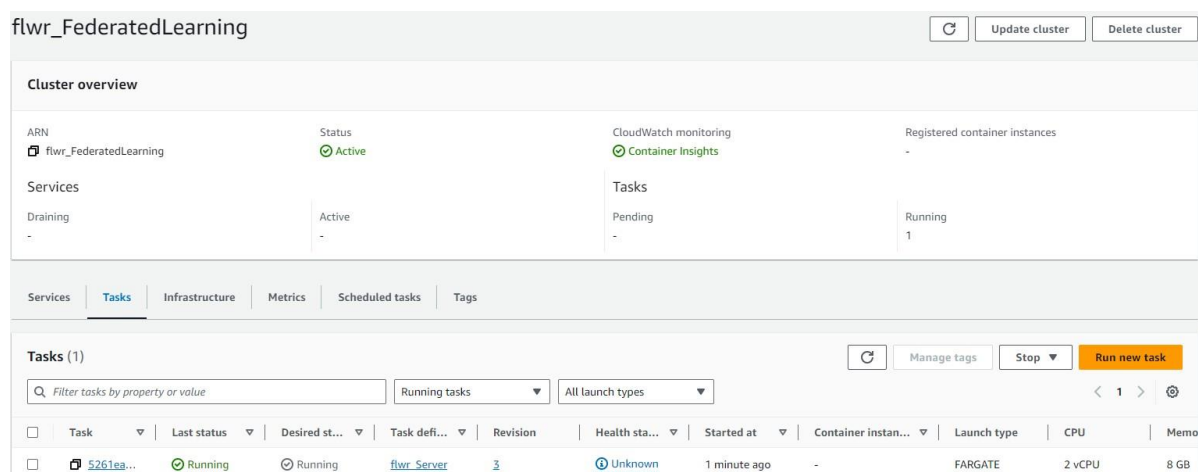


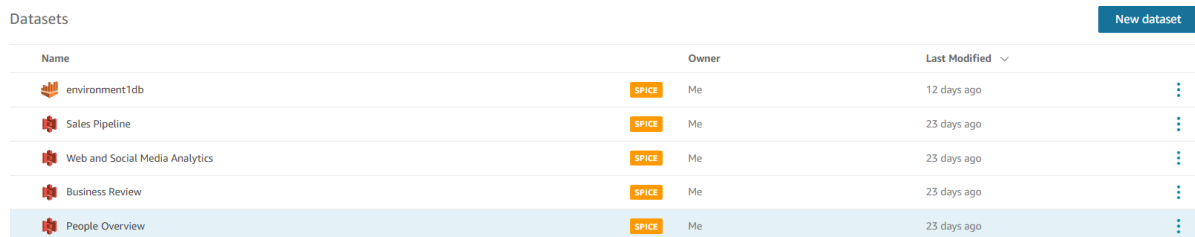
Figure 5.46: The Federated Learning Server Running as a Task in ECS Cluster

The task definitions defined will be executed either as a task or as an ECS service within the ECS cluster. In this implementation, AWS Fargate is employed, given that the federated learning server currently demands a modest level of computational power. As illustrated in figure 5.46, the federated learning server is launched as a single task to test the configuration in the “flwr_FederatedLearning” cluster. After the federated learning process completed, manual reconfiguration is needed to initiate further tasks. Alternatively, the federated learning server can also be deployed as an ECS service, where it will automatically launch new tasks based on predefined conditions.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.6.8 Amazon QuickSight

Amazon QuickSight is a BI and data visualization tool that enables the analysis and visualization of data, facilitating data-driven decision-making. It will be integrate with DynamoDB, which visualize processed data in near real-time on an interactive dashboard. The dashboards are designed to be compact and insightful, displaying only the essential information necessary for informed decision-making without overwhelm the user.



Name	Owner	Last Modified
environment1db	Me	12 days ago
Sales Pipeline	Me	23 days ago
Web and Social Media Analytics	Me	23 days ago
Business Review	Me	23 days ago
People Overview	Me	23 days ago

Figure 5.47: List of Datasets in QuickSight

The datasets used for the dashboard are sourced from Athena, which retrieves data by querying the specified DynamoDB table. As illustrated in the figure5.46 above, the Athena data source has been successfully added as a dataset named “environment1db” for visualization purpose. Besides, it’s allows to include multiple datasets, irrespective of whether the data sources are internal or external.

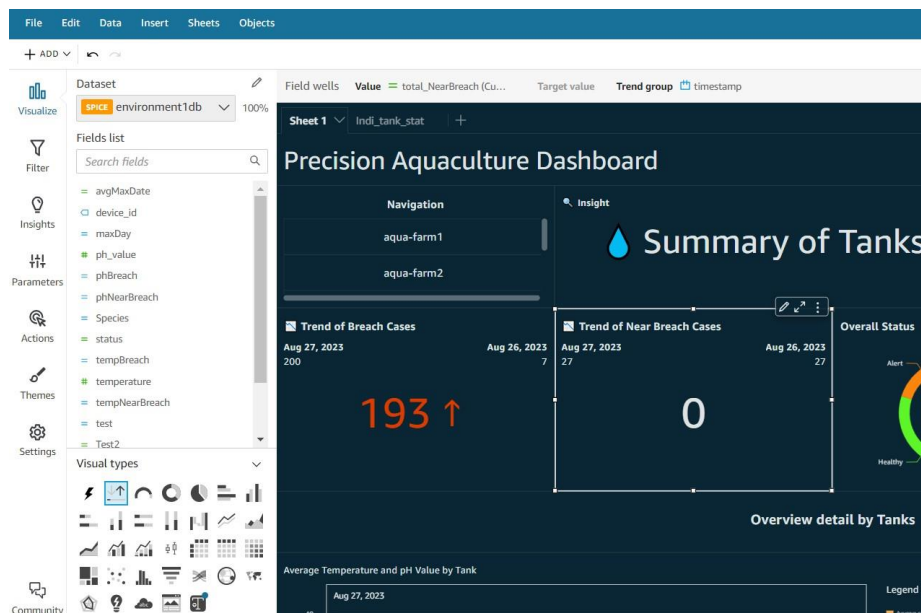


Figure 5.48: QuickSight Dashboard Developer Interface

CHAPTER 5 SYSTEM IMPLEMENTATION

The figure 5.47 depicts the dashboard developer interface, incorporates with essential features and visual options that simplified the creation of interactive dashboards. In addition to using the original data from datasets, calculated fields can be created to derive new data from the existing dataset. Furthermore, QuickSight also integrates intelligent features, enabling the use of ML-powered insights, including basic forecasting capabilities. After finish building the dashboard, it will be published to access by the authorize user.

5.7 Federated Learning Module Implementation

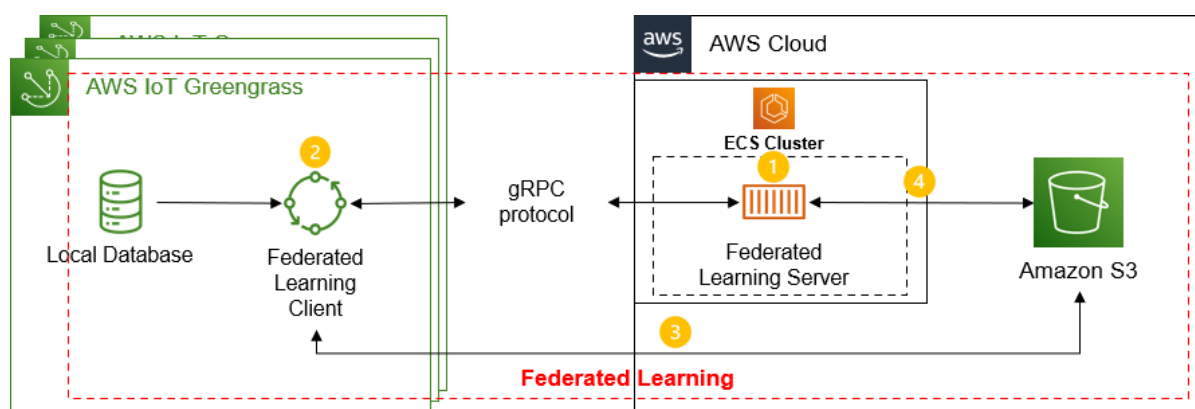


Figure 5.49: Federated Learning Module from Architecture Diagram

In figure 5.48 displays the federated learning module from the architecture diagram. The components and services required to perform federated learning have been prepared and ready to be put in service. In the edge environment, the federated learning client is represented by Greengrass component “com.example.flwrClient2” deployed on the core devices. Besides, the local database store locally processed data from “com.example.sensorv5” component. In the cloud, the federated learning server has been defined as an ECS task definitions, making it ready for deployment. While the ECS cluster “flwr_FederatedLearning” are used to allocating new tasks based on condition for federated learning. The S3 bucket “flwr-testbucket” serves as a central repository for the exchange and storage of global and client ML weights update. The communicate between client in edge and server in cloud utilized gRPC protocol on port 8080. With these components and services in place, the federated learning process can be activated anytime.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.7.1 Scheduling Deployment of Federated Learning Server

The screenshot displays the 'FL_serverSchedule' configuration in the AWS CloudWatch console. It includes an 'Update' button in the top right corner. The 'Scheduled task overview' section shows the following details:

ARN	Rule state	Schedule description	Schedule expression
FL_serverSchedule	Turned on	-	rate(7 days)

Additional details include Metrics (View Amazon CloudWatch metrics), Amazon CloudWatch IAM role, and Event pattern.

The 'Schedule targets (1)' section features a search bar and a table with the following data:

Target name	Task definition	Launch type	Amazon ECS target	Target role	Number of tasks
FL_server1	flwr_Server3	FARGATE	flwr_FederatedLear...	ecsEventsRole	1

Figure 5.50: Schedule Task with Target Task Definition

In this implementation, we initiate the activation of federated learning process by scheduled timestamp. It's planned to activate the federated learning server of every week, as this frequency allow sufficient data required for the ML model to make substantial improvements to be gathered. The scheduler's primary task is to dispatch the federated learning server on Fargate with the defined task definition as observed in Figure 5.49 when condition is met.

5.7.2 Setting Up DNS Name

The screenshot shows the configuration details for a Network Load Balancer named 'testLB'. The 'Details' section includes the following information:

Property	Value
Load balancer type	Network
Status	Active
VPC	vpc-08978670183ef5199
IP address type	IPv4
Scheme	Internet-facing
Hosted zone	ZKVM4W9LS7TM
Availability Zones	subnet-0fd597454634d662a ap-southeast-1a (apse1-az1)
Date created	August 27, 2023, 13:44 (UTC+08:00)
Load balancer ARN	arn:aws:elasticloadbalancing:ap-southeast-1:880455125848:loadbalancer/net/testLB/0d3733e6fadbb50f
DNS name	testLB-0d3733e6fadbb50f.elb.ap-southeast-1.amazonaws.com (A Record)

The 'Listeners (1)' section shows a single listener configuration:

Protocol:Port	Default action	ARN	Security policy	Default SSL cert	ALPN policy	Tags
TCP:8080	Forward to target group	testTG	Not applicable	Not applicable	None	0 tags

Figure 5.51: Configuration of Network Load Balancer testLB

CHAPTER 5 SYSTEM IMPLEMENTATION

Once the federated learning server is set up, the next step is to make it accessible by multiple clients. Federated learning clients access the server using the server's IP addresses. However, when tasks are dispatched on the ECS cluster, they will randomly assign whatever IP addresses available in the pool. To simplify the federated learning process without the need of specifying server IP addresses in every client component, a DNS name is needed for clients to access the server.

To achieve this, a network load balancer is established as shown in Figure 5.50, as the Flower federated learning library relies on the gRPC protocol on port 8080 for communication between clients and the server. The load balancer requires a target group so traffic can be forward to the federated learning server, by specifying that it should only receive traffic with TCP protocol on port 8080. Consequently, any traffic other than TCP will be dropped by the load balancer, ensuring that only traffic on port 8080 reaches the server. The network load balancer is assigned a DNS name so that clients can use to retrieve the server's IP address.

CHAPTER 6: System Deployment and Evaluation

6.1 Hardware Deployment

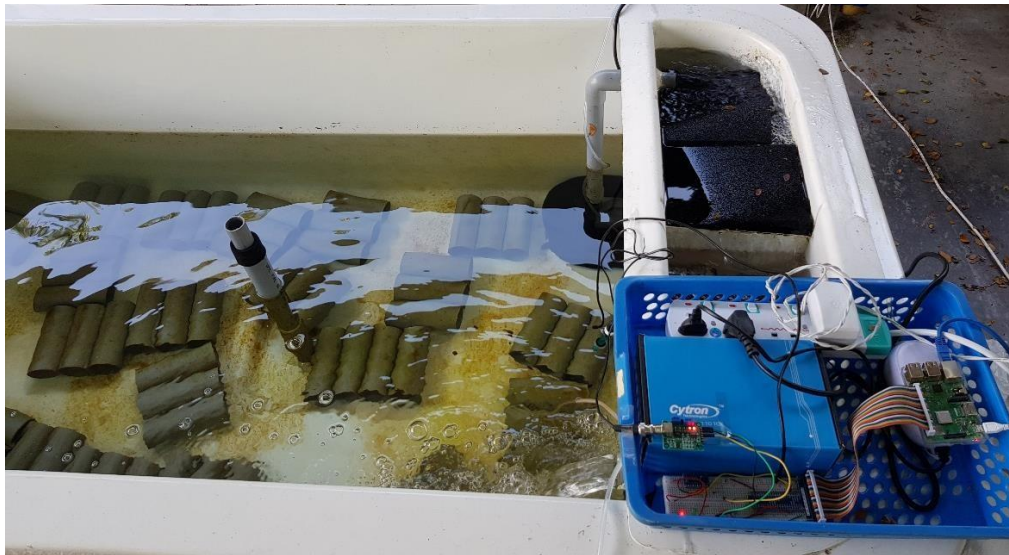


Figure 5.1: The Entire View of Fish Tank with IoT Hardware

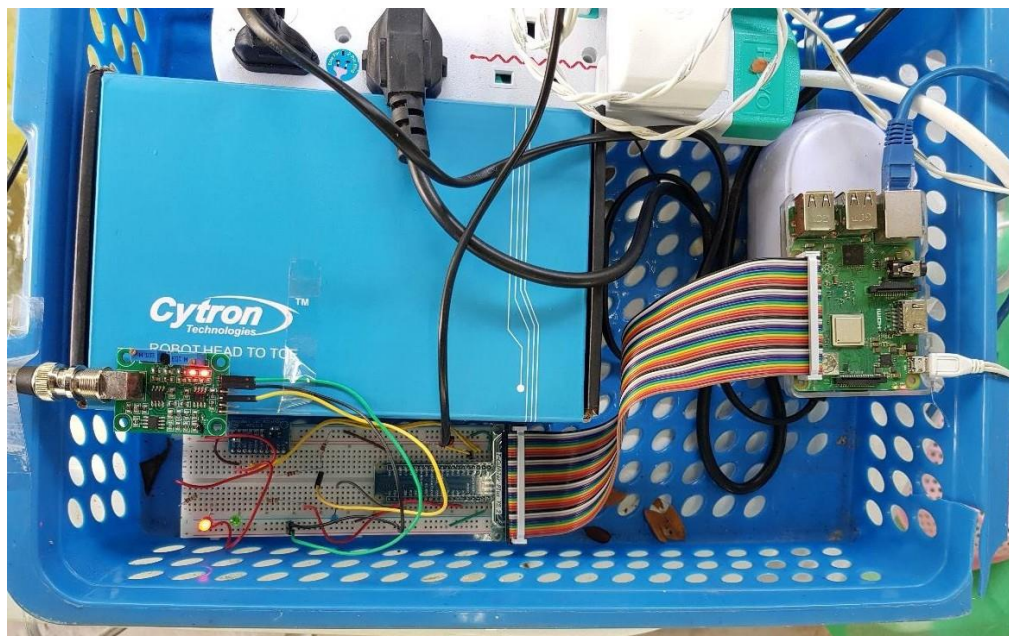


Figure 6.2: The Hardware Deployment

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION



Figure 6.3: DS18B20 and PH4502C Sensor in Water



Figure 6.4: Local Occupants

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

Figure 6.1 shows the entire IoT system deployed in an aquaculture tank for prawn farming. The tank itself is equipped with a water drainage system, filtration system, and an oxygen pump. The Raspberry Pi which is the Greengrass core device is incorporated into the setup, extended with a breadboard to accommodate the sensors and components, as depicts in figure 6.2. The LEDs involved serve as indicators for the system's status. For instance, a red LED indicates that the system is powered on, while a green LED means that the system is actively collecting sensor data. In Figure 6.3, it can be observed that the DS18B20 temperature sensor and PH4502C sensor are partially submerged in the water to collect water parameters.

This setup serves as the testing environment for the edge. All data collection, pre-processing, and storage are conducted locally by the Raspberry Pi with the Greengrass components deployed in it. This means that the system can operate to perform essential tasks even without network connectivity, but some function likes federated learning and synchronization require internet. The processed data from this tank will be visualized on a dashboard.

6.2 System Testing

Following the deployment of the hardware into the aquaculture tank, it is important to conduct comprehensive testing to ensure that both the edge environment functionalities and the cloud services perform as intended. The ultimate goal is to verify that the system functions seamlessly, with the end result being the successful conduct of federated learning process and visualization of processed data on a dashboard.

6.2.1 Greengrass Components Testing

The system is deployed with four Greengrass components, each serving its own specific function. The sensor reading and processing component "com.example.sensorv5," works in interdependently with the rules-based alert component "com.example.testAlarmv2." Additionally, the data synchronization component, known as "com.example.sync," is closely associated with these components. Therefore, they will be tested together to ensure their interaction and functionality work as intended.

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

6.2.1.1 Sensor Reading and Processing Component Testing (com.example.sensorv5)

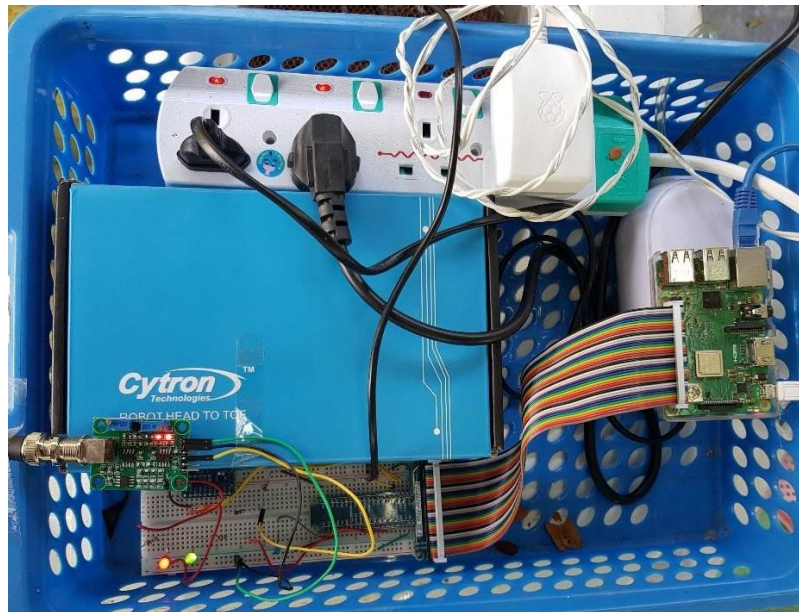


Figure 6.5: Green LED Illuminating on IoT Device

Message payload

```
{
  "on": "true"
}
```

► Additional configuration

Publish

Subscriptions **aquaFarm1/sensor**

Subscription	Message payload
aquaFarm1/sensor ♥ ✕	<pre>{ "message": "Hello from AWS IoT console" }</pre> <p>► Additional configuration</p> <p>Publish</p>

▼ aquaFarm1/sensor

```
{
  "device_id": "aqua-farm1",
  "timestamp": "2023-09-12 15:39:37",
  "temperature": 30.375,
  "ph_value": 7.7
}
```

► Properties

Figure 6.6: MQTT Test Client Receiving Message from Subscribed Topic

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

In Figure 6.5 shows once the sensor reading and processing component is activated, the green LED illuminates, indicating that the system is actively collecting water parameters. In scenarios with network connectivity, these water parameters go through pre-processing will be stored in a local database. Simultaneously, the processed data is transmitted to the cloud for backup by publishing it to a topic. As illustrated in Figure 6.6 in the cloud environment, an MQTT test client actively receives messages from the subscribed topic publish by the component.

Furthermore, the processed data is also published to a local topic “loc/sensor” for use by the rules-based alert component. To validate the data's successful storage in the local database, a query of the local database content will be performed.

```
try:
conn = sqlite3.connect(db_file_path)
cursor = conn.cursor()

cursor.execute('SELECT * FROM sensor_data')
```

Location	Time	Temp (C)	Humidity (%)
'aqua-farm1'	'2023-09-12 15:53:27'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:53:41'	30.5	7.72
'aqua-farm1'	'2023-09-12 15:53:54'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:54:07'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:54:21'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:54:34'	30.5	7.71
'aqua-farm1'	'2023-09-12 15:54:47'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:55:01'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:55:14'	30.625	7.7
'aqua-farm1'	'2023-09-12 15:55:28'	30.625	7.7
'aqua-farm1'	'2023-09-12 15:55:41'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:55:54'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:56:08'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:56:20'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:56:33'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:56:47'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:57:00'	30.562	7.72
'aqua-farm1'	'2023-09-12 15:57:14'	30.562	7.72

```
try:
conn = sqlite3.connect(db_file_path)
cursor = conn.cursor()

cursor.execute('SELECT * FROM temp_sensor_data')
```

Location	Time	Temp (C)	Humidity (%)
'aqua-farm1'	'2023-09-12 15:53:27'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:53:41'	30.5	7.72
'aqua-farm1'	'2023-09-12 15:53:54'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:54:07'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:54:21'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:54:34'	30.5	7.71
'aqua-farm1'	'2023-09-12 15:54:47'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:55:01'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:55:14'	30.625	7.7
'aqua-farm1'	'2023-09-12 15:55:28'	30.625	7.7
'aqua-farm1'	'2023-09-12 15:55:41'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:55:54'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:56:08'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:56:20'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:56:33'	30.562	7.7
'aqua-farm1'	'2023-09-12 15:56:47'	30.562	7.71
'aqua-farm1'	'2023-09-12 15:57:00'	30.562	7.72
'aqua-farm1'	'2023-09-12 15:57:14'	30.562	7.72

Figure 6.6: Content of Main Database (Left) and Temporary Database (Right)

In the absence of internet connectivity, the processed data will still store in the local database, as depicted in Figure 6.XX (Left) and will only publish to the local topic “loc/sensor”. However, the data processed during this period will also be simultaneously recorded in an additional temporary database, as shown in Figure 6.XX (Right). It's worth emphasizing that the content of both databases is identical, indicating that this component is functioning as intended. When network connectivity is restored, the content of the temporary database will be synchronized with the cloud database.

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

6.2.1.2 Rules-Based Alert Component Testing (com.example.testAlarmv2)

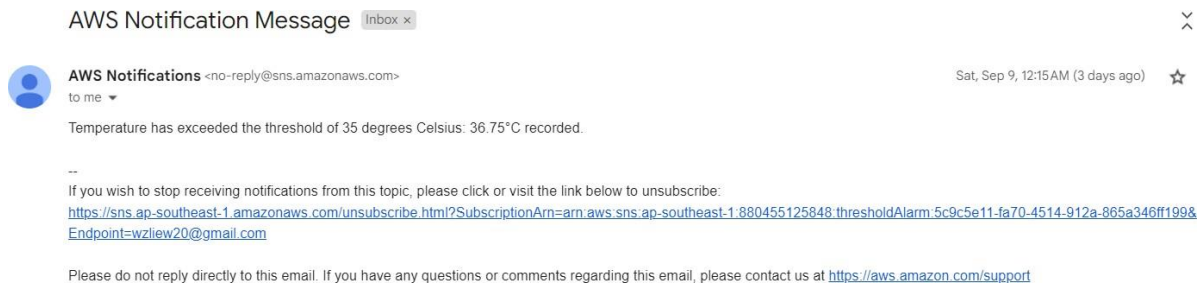


Figure 6.8: Email Message from AWS SNS

Concurrently, the rules-based alert component is subscribed to the local topic "loc/sensor" to assess whether any breaches have occurred in the water parameters, based on predefined thresholds. In the event of a breach, this component triggers an alarm by sending an email to the user, with the email content displayed in Figure 6.8. This email serves to inform the user that a particular parameter has exceeded its threshold, providing the relevant parameter value for reference.

6.2.1.3 Data Synchronization Component Testing (com.example.sync)

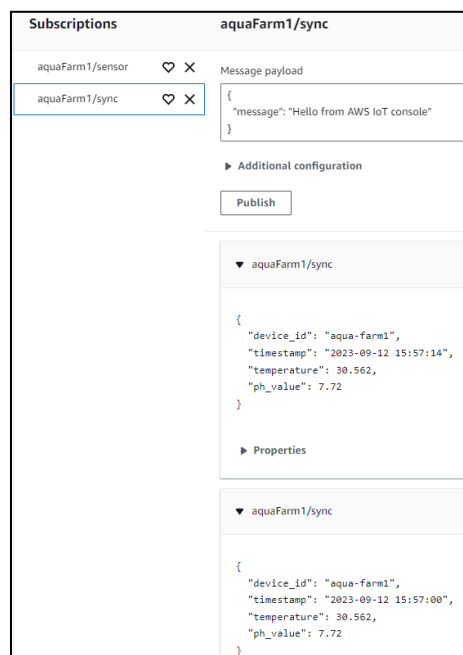


Figure 6.9: MQTT Receiving Message from Subscribe Synchronization Topic

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

The data synchronization component operates on a schedule, checking for internet connectivity every 5 minutes interval. When an internet connection is detected, it first verifies whether the temporary database is empty. If the temporary database contains data, the component initiates the synchronization process with the specified DynamoDB table in the cloud.

Figure 6.9 displays the topic “aquaFarm1/sync” in the AWS IoT Core MQTT Test Client, which actively receives messages from the synchronization component, confirming its proper functionality. However, in certain scenarios, the data in the temporary database may not be synchronized immediately. This is because Greengrass have a MQTT message queuing functionality to store an extensive amount of data during offline periods.

Nevertheless, when the connection is restored, the messages in the MQTT queue are sent to the cloud. But, the MQTT queue has a limited capacity, thus there’s the need of a data synchronization component to ensure reliable data synchronization.

6.2.2 Greengrass Components Verification Test

Components	Condition	Functionality	Result	Pass/Fail
Data Reading and Preprocessing (6.1.2.1)	- With network connection	Subscribe to topic “aquaFarm1/switch” for sensors activation/deactivation	Subscribe to topic “aquaFarm1/switch” for sensors activation/deactivation	Pass
	- Sensor reading activated	Read raw data from DS18B20 and PH4502C sensor and preprocess into meaningful format.	The raw data from sensors are processed into a meaningful format.	Pass
	- No network connectivity requirement	Processed data store into local database (sensor_data.db)	Processed data store into local database (sensor_data.db)	Pass
		Processed data store into	Processed data store into temporary	Pass

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

		temporary database (temp_sensor_data.db)	database (temp_sensor_data.db)	
		Package processed data into JSON message and publish to local topic "loc/sensor"	Package processed data into JSON message and publish to local topic "loc/sensor"	Pass
	- Sensor reading activated - With network connection	Package processed data into JSON message and publish to topic "aquaFarm1/sensor"	Package processed data into JSON message and publish to local topic "aquaFarm1/sensor"	Pass
Rules-Based Alert (6.1.2.2)	- No network connectivity requirement	Subscribe to local topic "loc/sensor" to retrieve message for water parameters evaluation	Subscribe to local topic "loc/sensor" to retrieve message for water parameters evaluation	Pass
	- With network connection	Send message alert to user informing water parameters breach happen using email through API call Amazon SNS	Send message alert to user informing water parameters breach happen using email through API call Amazon SNS	Pass
Data Synchronization (6.1.2.3)	- With network connection	Synchronize the valid items in temporary database with DynamoDB	Synchronize the valid items in temporary database with DynamoDB	Pass

Table 6.1: Greengrass Components Verification Testing Table

6.2.3 AWS Cloud Testing

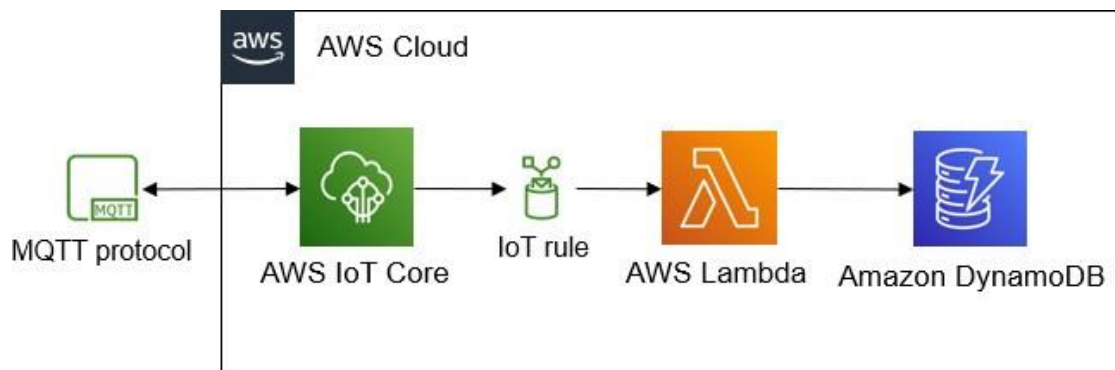


Figure 6.10: Data Routing Part from Architecture Diagram

<input type="checkbox"/>	device_id (String)	timestamp (String)	ph_value	temperature
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:41:19	7.69	30.437
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:41:08	7.71	30.375
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:40:56	7.71	30.375
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:40:45	7.71	30.5
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:40:34	7.71	30.5
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:40:23	7.72	30.437
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:40:11	7.69	30.437
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:40:00	7.71	30.437
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:39:49	7.71	30.437
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:39:37	7.7	30.375
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:39:26	7.71	30.375
<input type="checkbox"/>	aqua-farm1	2023-09-12 15:39:14	7.71	30.312

Figure 6.11: Items in DynamoDB Table environment1db

A way to validate the correct implementation of the part of AWS services depicted in figure 6.10 is to examine the target destination, which is DynamoDB. AWS IoT is configured to subscribe to topics “aquaFarm1/sensor” and “aquafarm/sync” and used IoT rules to route the incoming messages to a Lambda function for processing before storing them into DynamoDB. The successful storage of processed data in the DynamoDB table, as illustrated in Figure 6.11, indicate that all AWS services involved are functioning as intended.

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

6.2.4 Federated Learning Module Testing

To conduct a test on the federated learning module, a minimum of 2 active clients is required. Therefore, Amazon SageMaker is used to simulate a second client, running the same federated learning client deployed on it. Consequently, when the ECS cluster dispatches a federated learning server at scheduled intervals, the federated learning process can commence with the participation of both clients.

2023-09-13T00:00:33.112+08:00	2023-09-12 16:00:33.112194: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use...
2023-09-13T00:00:33.112+08:00	To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compil...
2023-09-13T00:00:33.949+08:00	2023-09-12 16:00:33.949862: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2023-09-13T00:00:35.642+08:00	INFO flwr 2023-09-12 16:00:35,642 app.py:148 Starting Flower server, config: ServerConfig(num_rounds=3, round_timeout=N...
2023-09-13T00:00:35.659+08:00	INFO flwr 2023-09-12 16:00:35,659 app.py:168 Flower ECE: gRPC server running (3 rounds), SSL is disabled
2023-09-13T00:00:35.659+08:00	INFO flwr 2023-09-12 16:00:35,659 server.py:86 Initializing global parameters
2023-09-13T00:00:35.930+08:00	INFO flwr 2023-09-12 16:00:35,930 server.py:273 Requesting initial parameters from one random client
2023-09-13T00:02:31.057+08:00	INFO flwr 2023-09-12 16:02:31,057 server.py:277 Received initial parameters from one random client
2023-09-13T00:02:31.057+08:00	INFO flwr 2023-09-12 16:02:31,057 server.py:88 Evaluating initial parameters
2023-09-13T00:02:31.057+08:00	INFO flwr 2023-09-12 16:02:31,057 server.py:101 FL starting
2023-09-13T00:06:32.002+08:00	DEBUG flwr 2023-09-12 16:06:32,002 server.py:218 fit_round 1: strategy sampled 2 clients (out of 2)
2023-09-13T00:07:13.609+08:00	DEBUG flwr 2023-09-12 16:07:13,609 server.py:232 fit_round 1 received 2 results and 0 failures
2023-09-13T00:07:13.628+08:00	/usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1056: InsecureRequestWarning: Unverified HTTPS request is being made. ...
2023-09-13T00:07:13.720+08:00	DEBUG flwr 2023-09-12 16:07:13,720 server.py:168 evaluate_round 1: strategy sampled 2 clients (out of 2)
2023-09-13T00:07:20.817+08:00	DEBUG flwr 2023-09-12 16:07:20,817 server.py:182 evaluate_round 1 received 2 results and 0 failures
2023-09-13T00:07:30.618+08:00	INFO flwr 2023-09-12 16:07:30,617 server.py:147 FL finished in 299.560512197
2023-09-13T00:07:30.618+08:00	INFO flwr 2023-09-12 16:07:30,618 app.py:218 app_fit: losses_distributed [(1, 2.231629783031987), (2, 2.172189305810367...
2023-09-13T00:07:30.618+08:00	INFO flwr 2023-09-12 16:07:30,618 app.py:219 app_fit: metrics_distributed_fit {}
2023-09-13T00:07:30.618+08:00	INFO flwr 2023-09-12 16:07:30,618 app.py:220 app_fit: metrics_distributed {}
2023-09-13T00:07:30.618+08:00	INFO flwr 2023-09-12 16:07:30,618 app.py:221 app_fit: losses_centralized []
2023-09-13T00:07:30.618+08:00	INFO flwr 2023-09-12 16:07:30,618 app.py:222 app_fit: metrics_centralized {}
2023-09-13T00:07:30.682+08:00	File uploaded to S3 bucket: flwr-testbucket, key: parameters/global_params.pkl
2023-09-13T00:07:30.682+08:00	Model weights from S3 key parameters/weights/client1params.pkl downloaded successfully.
2023-09-13T00:07:30.682+08:00	Model weights from S3 key parameters/weights/client2params.pkl downloaded successfully.
2023-09-13T00:07:30.682+08:00	Error aggregation: setting an array element with a sequence. The requested array has an inhomogeneous shape after 2 dimensi...
2023-09-13T00:07:30.682+08:00	Model weights from S3 key parameters/weights/client1params.pkl downloaded successfully.
2023-09-13T00:07:30.682+08:00	Model weights from S3 key parameters/weights/client2params.pkl downloaded successfully.
2023-09-13T00:07:30.682+08:00	Error aggregation: setting an array element with a sequence. The requested array has an inhomogeneous shape after 2 dimensi...
2023-09-13T00:07:30.682+08:00	Model weights from S3 key parameters/weights/client1params.pkl downloaded successfully.
2023-09-13T00:07:30.682+08:00	Model weights from S3 key parameters/weights/client2params.pkl downloaded successfully.
2023-09-13T00:07:30.682+08:00	Error aggregation: setting an array element with a sequence. The requested array has an inhomogeneous shape after 2 dimensi...

Figure 6.12: CloudWatch Event Log for Federated Learning Server

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

```

INFO flwr 2023-09-12 16:02:30,776 | grpc.py:49 | Opened insecure gRPC connection (no certificate)
DEBUG flwr 2023-09-12 16:02:30,820 | connection.py:42 | ChannelConnectivity.IDLE
DEBUG flwr 2023-09-12 16:02:30,833 | connection.py:42 | ChannelConnectivity.READY
/opt/conda/lib/python3.7/site-packages/boto3/compat.py:82: PythonDeprecationWarning: Boto3 will
dates, bug fixes, and security updates please upgrade to Python 3.8 or later. More information
r-aws-sdks-and-tools/
warnings.warn(warning, PythonDeprecationWarning)
{}Numpy_Get_Parameter
File uploaded to S3 bucket: flwr-testbucket, key: parameters/weights/client2params.pkl
{}Numpy_Fit_Parameter
File download from S3 bucket: flwr-testbucket, key: /tmp/tmpaoub774t
Epoch 1/10
43/43 [=====] - 3s 6ms/step - loss: 12.9183
Epoch 2/10
43/43 [=====] - 0s 4ms/step - loss: 2.2277
Epoch 3/10
43/43 [=====] - 0s 5ms/step - loss: 2.1685
Epoch 4/10
43/43 [=====] - 0s 5ms/step - loss: 2.1420
Epoch 5/10
43/43 [=====] - 0s 4ms/step - loss: 2.1490
Epoch 6/10
43/43 [=====] - 0s 4ms/step - loss: 2.1453
Epoch 7/10
43/43 [=====] - 0s 4ms/step - loss: 2.2368
Epoch 8/10
43/43 [=====] - 0s 4ms/step - loss: 2.2317
Epoch 9/10
43/43 [=====] - 0s 3ms/step - loss: 2.2668
Epoch 10/10
43/43 [=====] - 0s 3ms/step - loss: 2.2324
File uploaded to S3 bucket: flwr-testbucket, key: parameters/weights/client2params.pkl
{}Numpy_Evaluate_Parameter
File download from S3 bucket: flwr-testbucket, key: /tmp/tmpvo0tfgfx
43/43 [=====] - 0s 2ms/step - loss: 2.2089
DEBUG flwr 2023-09-12 16:07:30,645 | connection.py:139 | gRPC channel closed
INFO flwr 2023-09-12 16:07:30,647 | app.py:215 | Disconnect and shut down

```

Figure 6.13: Federated Learning Client Event Log

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	global_params.pkl	pkl	September 13, 2023, 00:14:09 (UTC+08:00)	49.5 KB	Standard
<input type="checkbox"/>	weights/	Folder	-	-	-

Figure 6.14: Global ML Model Weight Uploaded in S3 Bucket

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	client1params.pkl	pkl	September 13, 2023, 00:07:29 (UTC+08:00)	49.5 KB	Standard
<input type="checkbox"/>	client2params.pkl	pkl	September 13, 2023, 00:07:30 (UTC+08:00)	49.5 KB	Standard

Figure 6.15: Client Model Weight Uploaded in S3 Bucket

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

Figures 6.12 and 6.13 display the log events of the federated learning server and federated learning clients respectively, on Amazon CloudWatch. The server is responsible for orchestrating the federated learning process by sending commands to all participating clients. The numbers in Figure 6.12 and Figure 6.13 represent the steps within the federated learning process. The server will repeat the step below based on the configured training round.

Steps:

Step 1 (Server): Server requests an initial ML model weight, which serve as the global model from a random client to start the federated learning process.

Step 1 (Client): The random selected client uploads its ML model weight to S3 bucket.

Step 2 (Server): Server orchestrates the training process by invoking the training function on all participating clients.

Step 2 (Client): The global ML weights are downloaded from S3 bucket and used for training using data from its own database.

Step 3 (Client): After training, client updates the newly trained local ML model to S3 bucket.

Step 4 (Server): The server downloads the ML model weights from all clients and performs aggregation to create a new global model. The resulting aggregated model weights are then uploaded back to S3 as the updated global ML model.

6.2.5 Federated Learning Module Verification Test

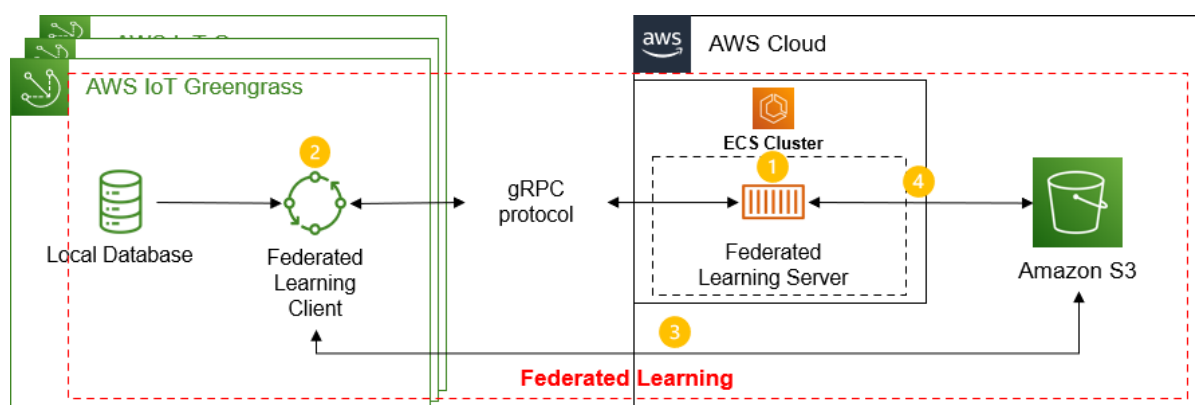


Figure 6.16: Federated Learning Module from Architecture Diagram

Steps	Description	Success/Fail
Step 1	Initialize global model using ML model weight from a random client	Success
Step 2	Clients perform local ML model training using local data	Success
Step 3	Clients update the trained local ML model weights to S3 bucket	Success
Step 4	Server perform model aggregation using client ML model weights and upload to S3 bucket	Success

Table 6.2: Federated Learning Steps Verification Test

6.3 Data Visualization

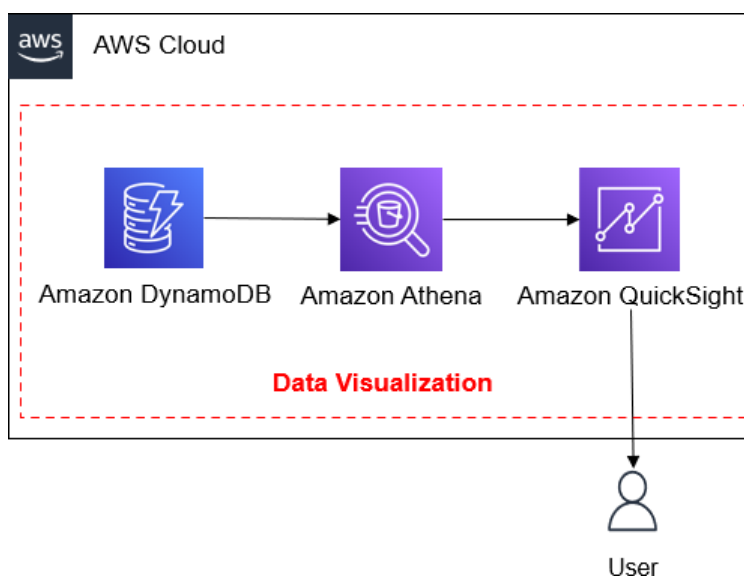


Figure 6.17: Data Visualization Module from Architecture Diagram

At this moment, the data collected from edge environment are processed and store in a DynamoDB table. As cover in the implementation section, Amazon Athena is used to query the data from DynamoDB and serve as a data source for QuickSight to visualise. The interactive dashboard build using QuickSight will be published to access by user, accessing only the data belong to their farm. The dashboard is design following several considerations to prevent redundant information present to user in order to achieve data-driven decision making.

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

Design Considerations:

- Prioritize Simplicity
- Near real time information
- How's the farm doing?
- How critical is the overall situation?
- How much not in healthy status?
- Who cause it?
- What causes it?

The Pages of Dashboard:

- Entire Aquaculture Farm (High Level)
- Individual Tank Detail (Individual Level)

6.3.1 High Level Dashboard

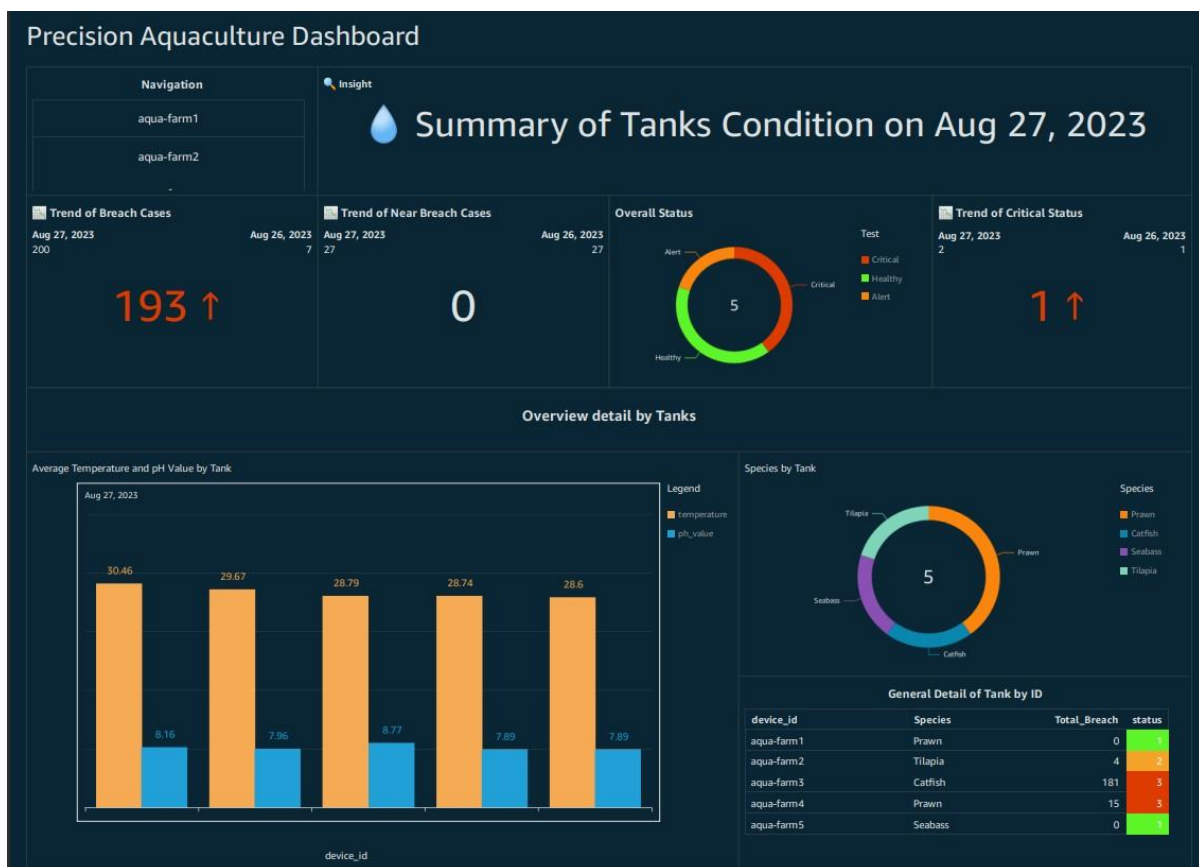


Figure 6.18: Main Page of Dashboard (High Level)

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

Dashboard Content:

- **Navigation Panel:** Navigate to individual tank.
- **Date:** The date of the information visualizes to user.
- **Trend for Breach Cases:** Need to diagnose what cause the issue (E.g., spike on breach cases)
- **Trend for Near Breach Cases:** Act in advance before the worst happened.
- **Overall Health Status of All Tanks:** Provides proportional view (E.g., High percentage of red mean something devastating happening in the farm)
- **Trend of Critical Status:** As indication of does the issue really solved?
- **Average Water Parameters by Tank:** Management consideration (E.g., Maintenance/Feed frequency)
- **Species by Tank:** Provides proportional view (E.g., High critical status might indicate pandemic on majority species)
- **General Detail of Tank:** Provide the number of breach cases and health status so farmer can dive into the details of individual tank.

6.3.2 Individual Level Dashboard

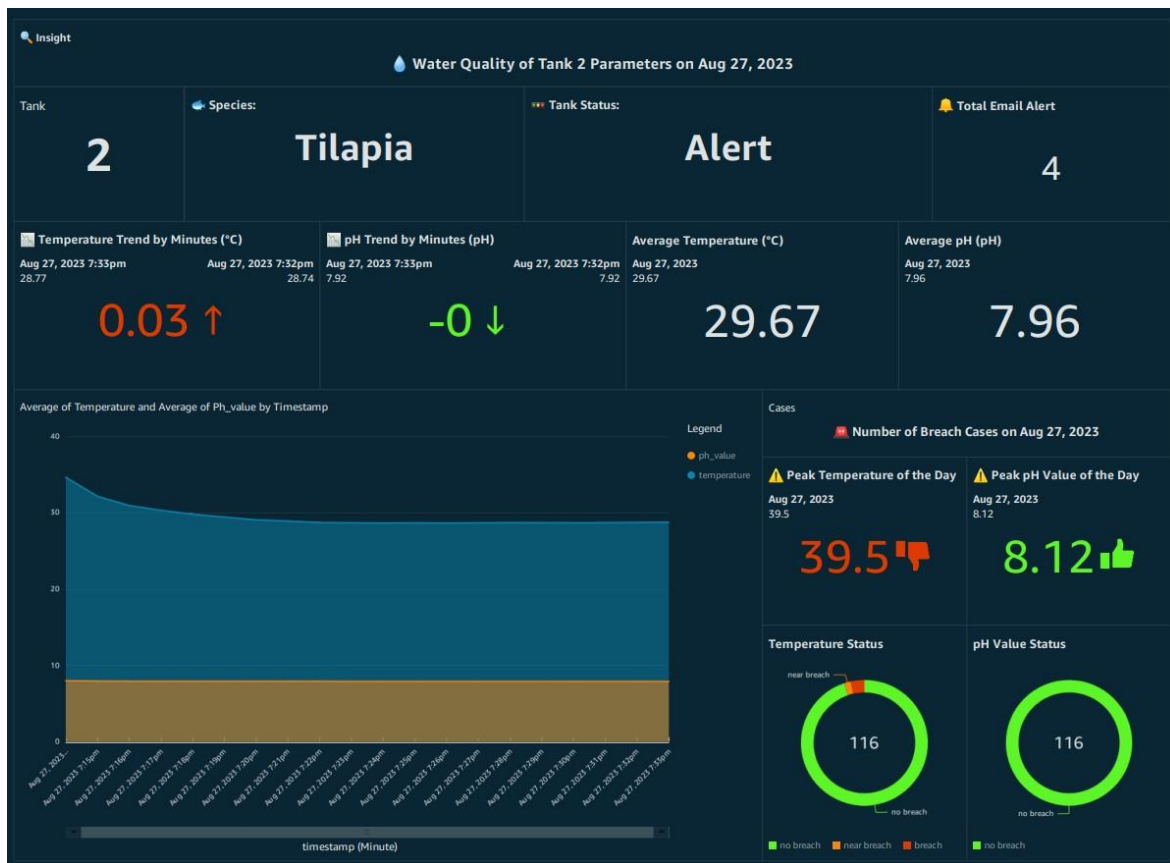


Figure 6.19: Individual Tank Detail Page of Dashboard (Individual Level)

Dashboard Content:

- **Date:** The date of the information visualizes to user.
- **Tank Number/ID:** Which tank’s information are being observe.
- **Species:** Different water condition requirement for different species.
- **Health Status:** The urgency of the issue should be diagnosed or solve.
- **Total Alert Sound:** Also known as the number of breach cases.
- **Temperature Trend:** Stability (Inconsistent need closer attention to prevent stress on aquatic life)
- **pH Value Trend:** Stability (Inconsistent need closer attention, signalling water quality issue need to address)
- **Average Temperature:** Management Considerations (E.g., Aging heater)
- **Average pH Value:** Management Considerations (E.g., Feed less quantity more frequent)

CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

- **Water Parameters Trend (Line Chart):** More information can be derived from line trend (E.g., Its normal temperature going down steadily from evening to night/ Pattern recognition)
- **Peak Temperature:** The level temperature can tell what kind of incident happened (E.g., Broken heater cause temperature goes down more than usual)
- **Peak pH Value:** The level pH value can tell what kind of incident happened (E.g., High peak value might indicate contamination)
- **Number of Temperature Breaches:** Derive focus from farmer (Prioritize solve this issue)
- **Number of pH Value Breaches:** Derive focus from farmer (Prioritize solve this issue)

CHAPTER 7: Conclusion

The motivation behind this project, “Designing an IoT-Cloud Solution for Precision Aquaculture” is mainly to address the issues faced in aquaculture industry, particularly in Malaysia. The use of IT in the Malaysian aquaculture industry is limited due to the predominance of small and medium-scale operators, who typically have limited resources for investing in IT solutions. As a result, they often rely on manual labour for daily operations, which can be unreliable, costly, and prone to errors. Thus, the proposed solution is providing a cost-effective, scalable and reliable way for aquaculture operators to monitor and manage their operations. Considering most of the aquaculture farm are located in remote area, edge computing is used to ensure the availability even having poor internet connection. While The AWS Cloud are used to for data backup, visualization and training machine learning model in this project.

The system design of the project is designed subsequently according to the objectives needed to be achieved. To ensure the feasibility and practicality of the system design in this project, a preliminary implementation was carried out on Project 1. A framework was developed based on the architecture diagram to facilitate testing and validation of the system. The purpose of the preliminary implementation was to test the system's functionality and performance for further improvement.

With the successful implementation of the first objective in Project 1, it has laid a framework for achieving the second and third objective for Project 2, as it is based on the platform from the first objective. In first objective, we successfully set up an edge environment and a cloud platform using AWS. The communication between the edge environment and the AWS Cloud was achieved through the MQTT protocol. Within the edge environment, a Greengrass core device was registered to manage the client devices in the environment. The local processing was achieved through the deployment of custom components. In the AWS Cloud, IoT rules were used to receive incoming data by subscribe to the topic published by the Greengrass core device to obtain data and further process it uses lambda function to store the data in DynamoDB.

CHAPTER 7 CONCLUSION

In Project 2, significant modifications have been introduced to the system architecture to address the requirements of the second and third objectives, as well as to overcome challenges previously encountered. Project 2 involved the implementation of both the edge environment and AWS Cloud components based on the new system architecture. Subsequently, the second objective was achieved by designing and constructing an interactive dashboard. This dashboard serves the purpose of visualizing data in near-real-time, leveraging processed data generated by the edge environments and available to the authorise user.

For the third objective, a federated learning framework was developed, which operates on a client-server architecture. The server is hosted in the cloud, and security measures have to be considered to ensure the server able to provide service to client. Various AWS services have been integrated to transform the server into service accessible by the client. As for the edge environment, it has been implemented with essential functions required by the federated learning process. These functions are critical for the server to orchestrate federated learning operations by invoking the necessary routines. Consequently, federated learning is executed through communication between the client and server, facilitated by the gRPC protocol, with select AWS services serving as intermediaries to facilitate this process.

After the implementation phase are completed, the system is deployed to a real aquaculture environment for rigorous testing and evaluation. The purpose of this deployment is to assess whether the system functions effectively in real-world conditions, mirroring its performance during the implementation phase. Successful results from this real-world testing phase mark the project as a success.

In conclusion, this project aims to address the challenges faced by the aquaculture industry in Malaysia by designing an IoT cloud solution using edge computing and federated learning. The combination of these two technologies is still relatively uncommon in this industry, making this project an opportunity to explore and innovate. Federated learning has been a trending topic in recent years, and this project seeks to leverage its potential in aquaculture. Ultimately, the goal is to contribute to the growth and development of the Malaysian aquaculture industry by providing a solution that addresses current issues and promotes sustainable and efficient practices.

REFERENCES

REFERENCES

- [1] "Annual Statistics - Department of Fisheries Malaysia Official Portal", Department of Fisheries Malaysia Official Portal, 2019. [Online]. Available: <https://www.dof.gov.my/en/resources/i-extension-en/annual-statistics/>. [Accessed: 09- Aug- 2022].
- [2] "Global tilapia industry under threat from highly contagious disease", WorldFish, 2019. [Online]. Available: <https://www.worldfishcenter.org/blog/global-tilapia-industry-under-threat-highly-contagious-disease>. [Accessed: 09- Aug- 2022].
- [3] B.L. Nicholson, "Fish Diseases in Aquaculture", Thefishsite.com, 2006. [Online]. Available: <https://thefishsite.com/articles/fish-diseases-in-aquaculture#:~:text=Infectious%20diseases%20pose%20one%20of,and%20spread%2%20of%20infectious%20diseases>. [Accessed: 10- Aug- 2022].
- [4] Banrie, "An introduction to fish health management", Thefishsite.com, 2013. [Online]. Available: <https://thefishsite.com/articles/an-introduction-to-fish-health-management>. [Accessed: 10- Aug- 2022].
- [5] E. Peeler and N. Taylor, "The application of epidemiology in aquatic animal health -opportunities and challenges", BMC, 2011. [Online]. Available: <https://veterinaryresearch.biomedcentral.com/articles/10.1186/1297-9716-42-94>. [Accessed: 11- Aug- 2022].
- [6] N. O. A. A. Fisheries, "Aquaculture fish health," NOAA, 29-Dec-2022. [Online]. Available: <https://www.fisheries.noaa.gov/content/aquaculture-fish-health>. [Accessed: 21- Apr-2023].

REFERENCES

- [7] “Fish immune system - Improve aquaculture production,” Veterinaria Digital. <https://www.veterinariadigital.com/en/articulos/fish-immune-system-how-to-improve-aquaculture-production-through-immunity/>. [Accessed: 11- Aug- 2022].
- [8] M. Føre, "Precision fish farming: A new framework to improve aquaculture, Part 1 - Responsible Seafood Advocate", Global Seafood Alliance, 2019. [Online]. Available: <https://www.globalseafood.org/advocate/precision-fish-farming-a-new-framework-to-improve-aquaculture-part-1/>. [Accessed: 11- Aug- 2022].
- [9] C. Tian Hee, "Weather, labour shortage affects fish supply in Malaysia, say industry players as prices go up", CNA, 2022. [Online]. Available: <https://www.channelnewsasia.com/asia/malaysia-fish-supply-weather-labour-shortage-price-hike-2717796>. [Accessed: 12- Aug- 2022]
- [10] A. Pounds, "Precision aquaculture, part 1: Data and evidence-based management -Responsible Seafood Advocate", Global Seafood Alliance, 2021. [Online]. Available: <https://www.globalseafood.org/advocate/precision-aquaculture-part-1-data-and-evidence-based-management/#%3A~%3Atext%3DThe%20aims%20of%20precision%20aquaculture%2Cmore%20evidence%2Dbased%20management%20decisions>. [Accessed: 13- Aug- 2022].
- [11] I. Edge, “What Is Edge Computing & Why Is It Important? | Accenture,” Accenture.com, 2022. <https://www.accenture.com/us-en/insights/cloud/edge-computing-index#%3A~%3Atext%3DEdge%20is%20about%20processing%20data>. [Accessed: 11- Aug- 2022].
- [12] IBM, “What Is Edge Computing,” Ibm.com, 2020. <https://www.ibm.com/topics/edge-computing>. [Accessed: 17-Apr-2023].
- [13] “Federatedlearning,” RapidMiner, 06-Sep-2022. [Online]. Available: <https://rapidminer.com/glossary/federated-learning/>. [Accessed: 18-Apr-2023].

REFERENCES

- [14] O.F. El-Gayar, "The use of information technology in aquaculture management", Research Gate, 1997. [Online]. Available: https://www.researchgate.net/publication/233118805_The_use_of_information_technology_in_aquaculture_management. [Accessed: 15- Aug- 2022].
- [15] J.W. Zahradnik, "Status and Perspectives in the Instrumentation of Aquacultural Facilities", Science Direct, 1987. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1474667017591521>. [Accessed: 15- Aug- 2022].
- [16] Varvarigos, P. and M.T. Hone, "An inexpensive microcomputer-based data record keeping system for the individual fish farm", in Automation and Data Processing in Aquaculture (Ed.), J.G Balchen, NY:Pergamon Press, 1978, pp. 263-272.
- [17] El-Gayar, O.F., P.S. Leung and L. Rowland, "An aquacultural development decision support system (ADDSS): a preliminary design", presented at 25th Conference of the World Aquaculture Society, New Orleans, USA, 1994.
- [18] Lee, P.G., "Computer automation for recirculating aquaculture systems", in Techniques for Modern Aquaculture (Ed.), J.K. Wang, St Joseph, Minnesota: American Society of Agriculture Engineers, 1993, pp. 61-70.
- [19] Naiberg, A., J. Petrell, C.R. Savage and T.P. Neufeld, "A non-invasive fish size assessment method for tanks and sea cages using stereo video", in Techniques for Modern Aquaculture (Ed.), J. K. Wang, St Joseph, Minnesota: America Society of Agriculture Engineers, 1993, pp. 372-381.
- [20] Ross, L.G., E.A. Mendoza and M.C.M., The application of geographical information systems to site selection for coastal aquaculture: An example based on salmonid cage culture, Aquaculture 112:165-178, 1993.

REFERENCES

- [21] D.S. Simbeye, "Water Quality Monitoring and Control for Aquaculture Based on Wireless Sensor Networks", Research Gate, 2014. [Online]. Available: [https://www.researchgate.net/publication/262380185_Water_Quality_Monitoring_and Contr ol_for_Aquaculture_Based_on_Wireless_Sensor_Networks](https://www.researchgate.net/publication/262380185_Water_Quality_Monitoring_and_Contr ol_for_Aquaculture_Based_on_Wireless_Sensor_Networks). [Accessed: 20- Aug- 2022].
- [22] S. Karim, I. Hussain, A. Hussain, K. Hassan and S. Iqbal, "IoT Based Smart Fish Farming Aquaculture Monitoring System", Researchtrend.net, 2021. [Online]. Available: <https://www.researchtrend.net/ijet/pdf/8%20IoT%20Based%20Smart%20Fish%20Farming%20Aquaculture%20Monitoring%20System%20Sohail%20Karim%203461.pdf>. [Accessed: 20- Aug- 2022].
- [23] M. Saberioon, A. Gholizadeh and P. Cisar, "Application of Machine Vision Systems in Aquaculture with Emphasis on Fish: State-of-the-Art and Key Issues", Research Gate, 2017. [Online]. Available: [https://www.researchgate.net/publication/288516072_Application_of_Machine_Vision Syste ms_in_Aquaculture_with_Emphasis_on_Fish_State-of-the-Art_and_Key_Issues](https://www.researchgate.net/publication/288516072_Application_of_Machine_Vision_Syste ms_in_Aquaculture_with_Emphasis_on_Fish_State-of-the-Art_and_Key_Issues). [Accessed: 23- Aug- 2022].
- [24] R. R. Teixeira, J. B. Puccinelli, L. Poersch, M. R. Pias, and V. M. Oliveira, "Towards precision aquaculture: A high performance, cost-effective IOT ..." [Online]. Available: https://www.researchgate.net/publication/351869126_Towards_Precision_Aquaculture_A_Hi gh_Performance_Cost-effective_IoT_approach. [Accessed: 15-Apr-2023].
- [25] F. O'Donncha and J. Grant, "precision aquaculture - researchgate." [Online]. Available: https://www.researchgate.net/publication/339059062_Precision_Aquaculture. [Accessed: 15- Apr-2023].
- [26] G. Kaur, N. Adhikari, S. Krishnapriya, S. G. Wawale, R. Q. Malik, A. S. Zamani, J. Perez-Falcon, and J. Osei-Owusu, "Recent advancements in deep learning frameworks for precision fish farming opportunities, challenges, and applications," Journal of Food Quality, 07-Feb-2023. [Online]. Available:

REFERENCES

<https://www.hindawi.com/journals/jfq/2023/4399512/#conclusion>. [Accessed: 16- Apr-2023].

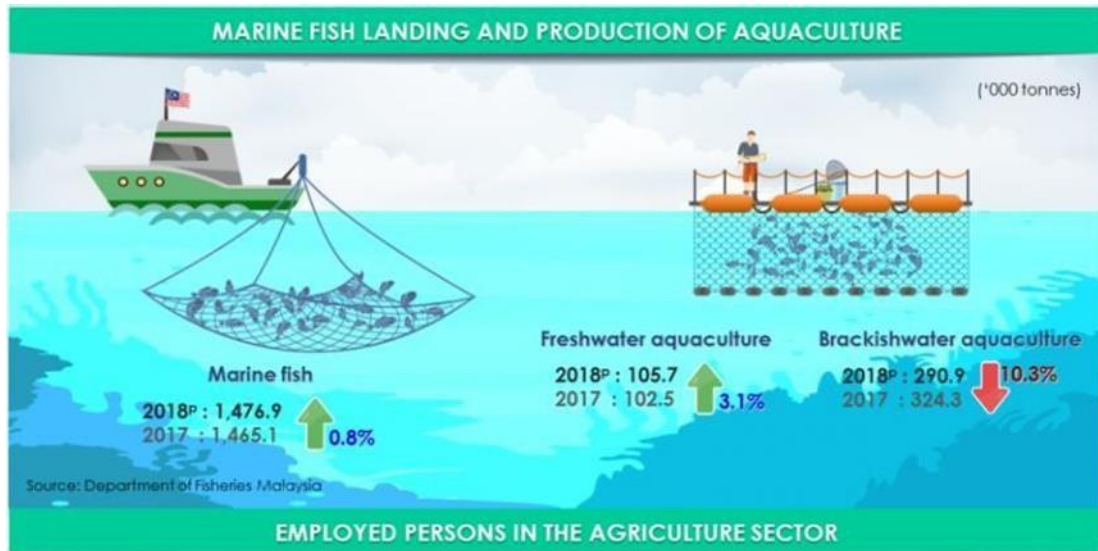
[27] A. Pounds, Assistant Researcher in Aquatic Food Security Institute of Aquaculture, and [107, “Precision Aquaculture, part 1: Data and evidence-based management - responsible seafood advocate,” Global Seafood Alliance, 23-Mar-2023.

[Online]. Available: <https://www.globalseafood.org/advocate/precision-aquaculture-part-1-data-and-evidence-based-management/>. [Accessed: 16-Apr-2023].

[28] “Why calibrate your pH meter and how!,” OroCommerce. [Online]. Available: <https://www.instrumentchoice.com.au/why-calibrate-your-ph-meter-and-how/#%3A~%3Atext%3DA%20pH%20calibration%20is%20the%2Ccharacteristics%20of%20your%20pH%20sensor>. [Accessed: 19-Apr-2023].

APPENDIX

Appendix A



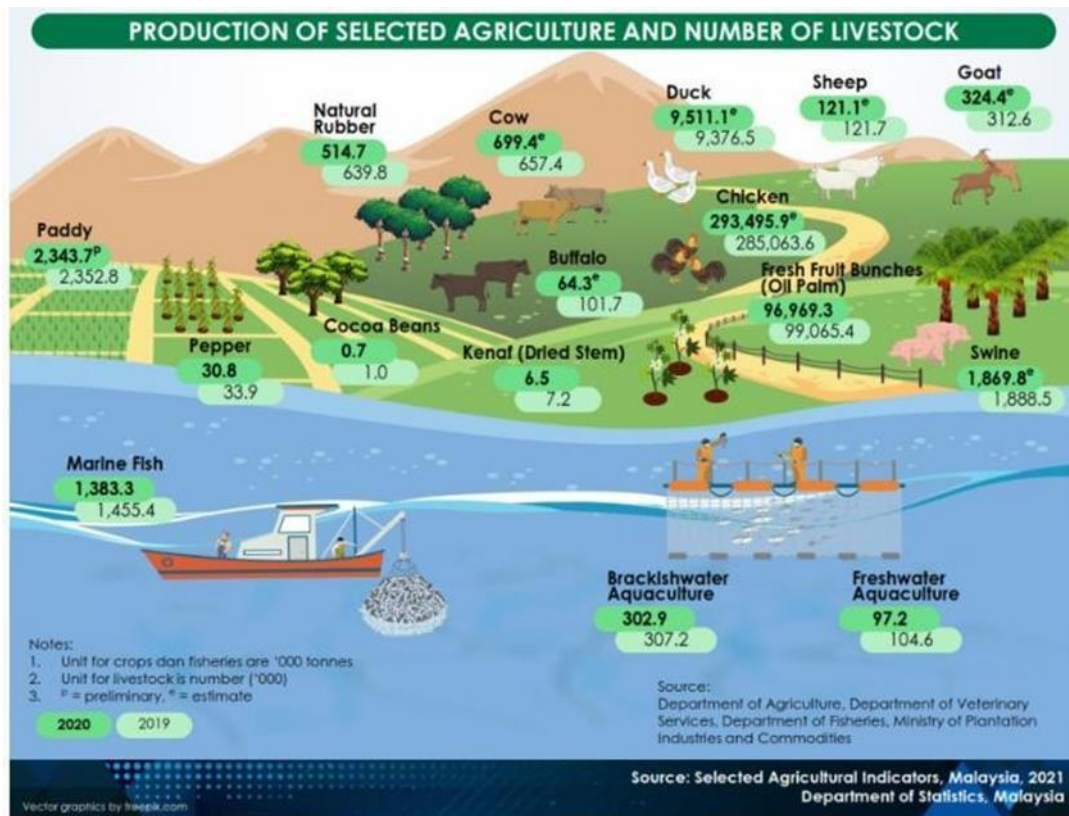
Production of selected agriculture sector 2018 (Department of Statistic Malaysia)

Source:

https://www.dosm.gov.my/v1/index.php/index.php?r=column/cthemByCat&cat=72&bul_id=SEUxMEE3VFdBcDJhdUhPZVUxa2pKdz09&menu_id=Z0VTZGU1UHBUT1VJMF1paXR00xpdz09

APPENDIX

Appendix B



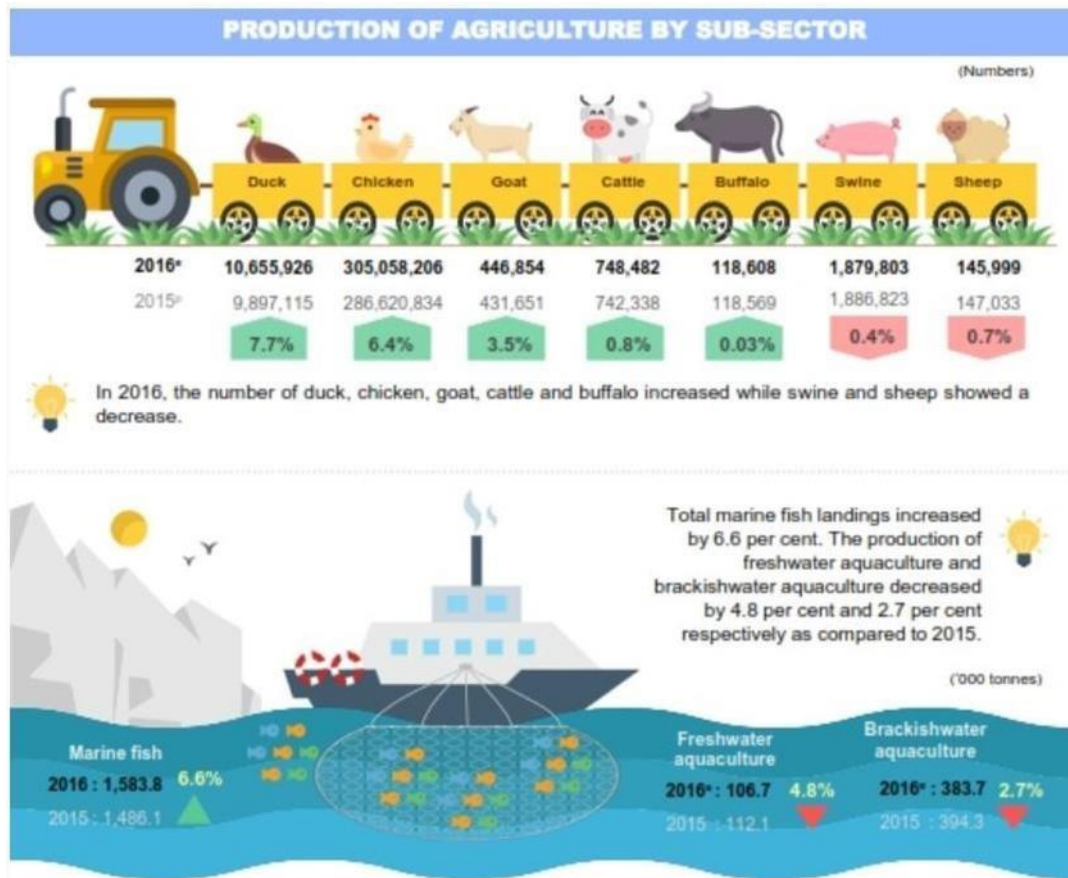
Production of selected agriculture sector 2019 (Department of Statistic Malaysia)

Source:

https://www.dosm.gov.my/v1/index.php/index.php?r=column/cthemByCat&cat=72&bul_id=TDV1YU4yc1Z0dUVyZ0xPV0ptRlhWQT09&menu_id=Z0VTZGU1UHBT1VJMF1paXRRR0xpdz09

APPENDIX

Appendix C



Production of selected agriculture sector 2016 (Department of Statistic Malaysia)

Source:

[https://www.dosm.gov.my/v1/index.php?r=column/cthemByCat&cat=72&bul_id=MDNYUitINmRKcENRY2FvMmR5TWdGdz09&menu_id=Z0VTZGU1UHBUT1VJMFlpaXRRR0pdz09#:~:text=Agriculture%20sector%20grew%208.1%20per%20cent%20in%202016&text=5%20billion%20to%20the%20Gross,%25\)%20and%20rubber%20\(7.1%20%25\).](https://www.dosm.gov.my/v1/index.php?r=column/cthemByCat&cat=72&bul_id=MDNYUitINmRKcENRY2FvMmR5TWdGdz09&menu_id=Z0VTZGU1UHBUT1VJMFlpaXRRR0pdz09#:~:text=Agriculture%20sector%20grew%208.1%20per%20cent%20in%202016&text=5%20billion%20to%20the%20Gross,%25)%20and%20rubber%20(7.1%20%25).)

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	Study week no.: 2
Student Name & ID: LIEW WEI ZHENG 2005319	
Supervisor: DR CHENG WAI KHUEN	
Project Title: Designing an IoT-Cloud Solution for Precision Aquaculture	

1. WORK DONE

Problem resolved Cgroup reversion from version 2 to version 1 and other issue.

2. WORK TO BE DONE

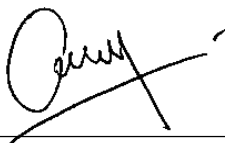
Implement the federated learning framework.

3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

On Track



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	Study week no.: 4
Student Name & ID: LIEW WEI ZHENG 2005319	
Supervisor: DR CHENG WAI KHUEN	
Project Title: Designing an IoT-Cloud Solution for Precision Aquaculture	

1. WORK DONE

Trail and error different federated learning library and revise the system architecture diagram.

2. WORK TO BE DONE

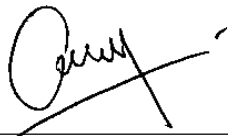
Implement both the federated learning client and server.

3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

On Track



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	Study week no.: 6
Student Name & ID: LIEW WEI ZHENG 2005319	
Supervisor: DR CHENG WAI KHUEN	
Project Title: Designing an IoT-Cloud Solution for Precision Aquaculture	

1. WORK DONE

- Federated learning client and server code successfully tested.
- Deploy code on edge environment and cloud.

2. WORK TO BE DONE

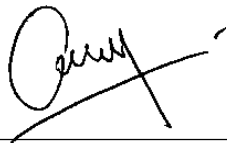
Troubleshoot the issue encounter.

3. PROBLEMS ENCOUNTERED

Federated learning client and server fail to communicate.

4. SELF EVALUATION OF THE PROGRESS

Behind Schedule



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	Study week no.: 8
Student Name & ID: LIEW WEI ZHENG 2005319	
Supervisor: DR CHENG WAI KHUEN	
Project Title: Designing an IoT-Cloud Solution for Precision Aquaculture	

1. WORK DONE

Federated learning framework successfully implemented as work as intended.

2. WORK TO BE DONE

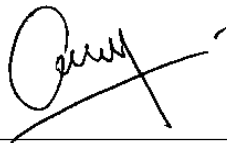
Design and build a dashboard.

3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

On Track



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	Study week no.: 10
Student Name & ID: LIEW WEI ZHENG 2005319	
Supervisor: DR CHENG WAI KHUEN	
Project Title: Designing an IoT-Cloud Solution for Precision Aquaculture	

1. WORK DONE

- Dashboard has been complete and publish
- System architecture revise

2. WORK TO BE DONE

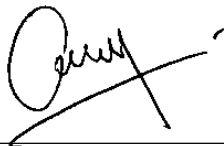
- Physical deployment of the system
- Prepare FYP report

3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

On Track



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	Study week no.: 12
Student Name & ID: LIEW WEI ZHENG 2005319	
Supervisor: DR CHENG WAI KHUEN	
Project Title: Designing an IoT-Cloud Solution for Precision Aquaculture	

1. WORK DONE

- Physical deployment testing and troubleshoot.
- Finalize the system architecture diagram

2. WORK TO BE DONE

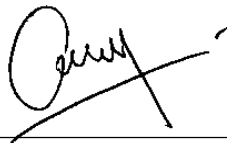
Continue for the FYP report preparation.

3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

On Track

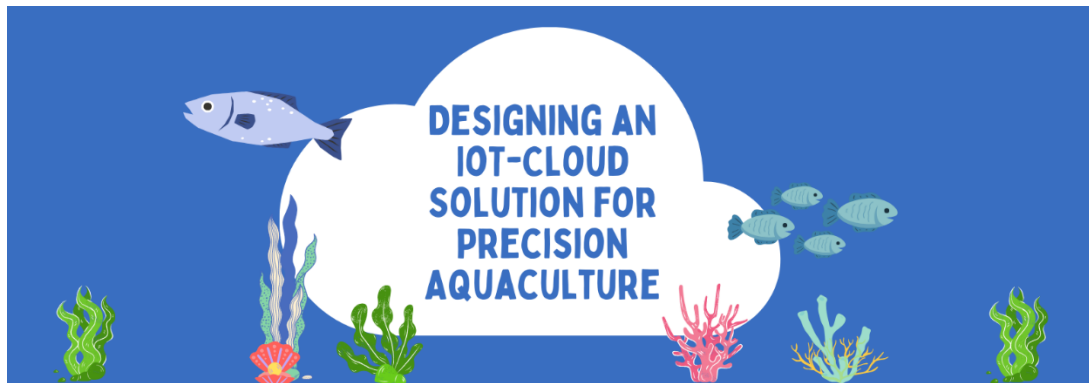


Supervisor's signature



Student's signature

POSTER



DESIGNING AN IOT-CLOUD SOLUTION FOR PRECISION AQUACULTURE

PROBLEM STATEMENT

Heavy Labor Dependency:

- Contributes significantly to aquaculture costs.
- Vulnerable to labor shortages.

Manual Water Sampling:

- Time-consuming and prone to errors.

Reliance on Visual Observations:

- Limited use of testing equipment for environment monitoring.

Data Interpretation:

- Critical for decision-making.

OBJECTIVES

IoT-Cloud Platform for Precision Aquaculture

- Develop an IoT-Cloud system integrating sensors and edge computing.
- Local data processing and storage.

Intelligent Cloud Dashboard for Data Visualization

- Cloud-based interactive dashboard near real-time data visualization.

Federated Learning in Edge Environments

- Local model training.
- Cloud-based server as coordinator.
- Enhancing aquaculture management with collaborative machine learning.


PROJECT HIGHLIGHT

EDGE COMPUTING

- Lower latency and improve respond time
- Reduce bandwidth consumption
- Less reliance on network connection
- Real-time decision making

FEDERATED LEARNING

- Ensure data privacy
- Diverse training data
- Decentralize approach
- Multiple edge contribute to the training of a machine learning model



INTERACTIVE DASHBOARD

- Near real time access to crucial aquaculture data.
- Interactive charts and graphs for quick insights
- Monitor the status of all tanks at a glance
- Empower aquaculture management with actionable insights

BY LIEW WEI ZHENG



PLAGIARISM CHECK RESULT

FYP_TURNITIN_SUBMIT

ORIGINALITY REPORT

8%

SIMILARITY INDEX

6%

INTERNET SOURCES

4%

PUBLICATIONS

4%

STUDENT PAPERS

PRIMARY SOURCES

1

scholar.dsu.edu

Internet Source

1%

2

docs.aws.amazon.com

Internet Source

1%

3

Submitted to Malta College of Arts, Science and Technology

Student Paper

1%

4

eprints.utar.edu.my

Internet Source

<1%

5

www.hachonline.com

Internet Source

<1%

6

Submitted to Kingston University

Student Paper

<1%

7

fict.utar.edu.my

Internet Source

<1%

8

Asif Abbasi. "AWS Certified Data Analytics Study Guide", Wiley, 2020

Publication

<1%

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



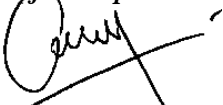
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	LIEW WEI ZHENG
ID Number(s)	20ACB05319
Programme / Course	Bachelor of Information Systems (Honours) Information Systems Engineering
Title of Final Year Project	Designing an IoT-Cloud Solution for Precision Aquaculture

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u>8</u> % Similarity by source Internet Sources: <u>6</u> % Publications: <u>4</u> % Student Papers: <u>4</u> %	OK
Number of individual sources listed of more than 3% similarity: <u>0</u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.



Signature of Supervisor

Signature of Co-Supervisor

Name: Ts. Dr. Cheng Wai Khuen

Name: _____

Date: 14/9/2023

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

**FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)**

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB05319
Student Name	LIEW WEI ZHENG
Supervisor Name	DR CHENG WAI KHUEN

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 13/09/2023