

**To develop a Federated Learning Framework for Precision Aquaculture**

By

John Ling Tze Jun

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

Jan 2024

## REPORT STATUS DECLARATION FORM

**Title:** Develop a Federated Learning Framework for Precision Aquaculture

**Academic Session:** JAN 2024

I JOHN LING TZE JUN  
(CAPITAL LETTER)

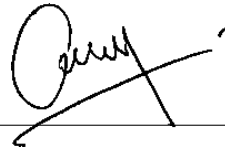
declare that I allow this Final Year Project Report to be kept in  
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

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:**

1, JALAN AMAN PERDANA  
11C/KU5 ,AMAN PERDANA , 41050  
KLANG, SELANGOR

Ts. Dr. Cheng Wai Khuen

\_\_\_\_\_  
Supervisor's name

**Date:** 11/04/2024

**Date:** 11/4/2024

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: 15/04/2024

**SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS**

It is hereby certified that JOHN LING TZE JUN (ID No: 21ACB02602) has completed this final year project/ dissertation/ thesis\* entitled "Develop a Federated learning Framework 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,




(JOHN LING TZE JUN)

\*Delete whichever not applicable

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**To develop a federated learning framework 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 : John Ling Tze Jun

Date : 15/04/2024

## **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 an Federated Learning framework for precision aquaculture. It is my first step to establish a career in cloud computing field. A million thanks to you.

Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course. Without them, it would be hard for me to cope with the struggles during this period.

## ABSTRACT

In the past few years, the growing population of the world has resulted in the increase in demand for aquaculture products, this has become one of the factors for depletion of wild fish population. In order to address the issue of depleting fish stocks from overfishing and environment pollution, precision aquaculture has been chosen as an alternative with potential to supply fish stocks to fulfil the current demands. Precision aquaculture is the cultivation of years of innovation and evolution in the aquaculture industry together with the use of different technologies to improve its efficiency and automation. However, this innovation is still vulnerable towards factors such as labour shortage, diversity of data and data privacy concerns. The water quality of aquafarms needs to be monitored periodically since it plays such an important role for determining the health and growth of fishes. The accuracy and experience of farmers to measure the water parameters in their farms determine the accuracy of the results but they are prone towards human error. Data also requires to be diverse in order to train machine learning models but cannot be shared across different aquafarms due to privacy concerns at the same time. Hence, this project aims to design a federate learning framework with an IoT Cloud solution for Precision Aquaculture for predicting water quality to improve the overall efficiency and effectiveness of aquaculture operations for production. Inside this paper, aquaculture farms will be referred as edge computing environments which will consist of IoT sensors and devices to obtain the real time water parameter readings from each farm and sending them into the cloud for further processing purposes using MQTT protocol. The data will also be processed locally using federated learning frameworks to train machine learning models which are aggregated from a main model located inside the cloud. After models have successfully been trained locally, they will then be sent back into the cloud for further storage, training and analysis to build the main model for predicting water quality and raising alerts to fish farmers along with recommended solutions. Finally, the collected data and predictions will be displayed in an interactive dashboard to assist farmers in making better data-driven decisions.

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>REPORT STATUS DECLARATION FORM</b>	<b>ii</b>
<b>FYP THESIS SUBMISSION FORM</b>	<b>iii</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>iv</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>TABLE OF CONTENTS</b>	<b>vi</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>xii</b>
<b>LIST OF SYMBOLS</b>	<b>xvi</b>
<b>LIST OF ABBREVIATIONS</b>	<b>x</b>
<b>CHAPTER 1 PROJECT BACKGROUND</b>	<b>1</b>
1.1 Introduction	1
1.2 Problem Statement	3
1.3 Motivation	5
1.4 Impact, significance and contribution	6
1.5 Report Organization	7
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>8</b>
2.1 Previous Works	8
2.1.1 Application of machine learning in intelligent fish aquaculture: A review	8
2.1.2 Aquasafe: A Remote Sensing, Web-Based Platform for the Support of Precision Fish Farming	11
2.1.3 Precision aquaculture by Fearghal O’Donncha and Jon Grant	13
2.1.4 ATLAS: Internet of Things Platform for Precision Aquaculture	15

2.1.5 IoT Based Smart Fish Farming Aquaculture Monitoring System	17
2.2 Comparison between previous aquaculture framework and proposed solution	19
2.3 Proposed Solutions	20
<b>CHAPTER 3 PROJECT SCOPE AND OBJECTIVES</b>	<b>22</b>
3.1 Project Scope	22
3.2 Project Objectives	23
3.3 Contributions	24
<b>CHAPTER 4 METHODS/TECHNOLOGIES INVOLVED</b>	<b>25</b>
4.1 System Architecture Diagram	25
4.2 Methodology	28
4.3 Project Workflow	29
4.3.1 Planning Phase	30
4.3.2 Analysis Phase	30
4.3.3 Design Phase	30
4.3.4 Implementation Phase	31
4.3.5 System Prototype	31
4.3.6 Implementation (Final)	31
4.4 Formulas for sensor Calibration	32
4.5 Technologies and Tools Involved	33
4.5.1 Laptop	33
4.5.2 Amazon Web Services (AWS)	33
4.5.3 Python Programming Language	34
4.5.4 Jupyter Notebook	34
4.5.5 AWS IoT Greengrass Core Software	35
4.5.6 MQTT Protocol	35
4.5.7 JSON	35
4.5.8 Raspberry Pi OS (Bookworm 64-bit)	36
4.5.9 Raspberry Pi Model 3 B+	36



4.5.10 Raspberry Pi T-cobbler	37
4.5.11 DS18B20 Waterproof Temperature Sensor	38
4.5.12 PH4502C Analog pH Sensor	39
4.5.13 ADS1115 Analog Digital Converter	40
4.5.14 Microsoft Power Bi	40
4.5.15 Arduino Uno	41
4.5.16 Flower Federated Learning Library	41
4.5.17 TensorFlow	42
4.6 Hardware Prototype Design	42
4.7 Schematic Diagram	43
4.7.1 pH Reading Module	43
4.7.2 Temperature Reading Module	44
4.7.3 T-Cobbler and LEDs	45
4.8 Project Timeline	46

## **CHAPTER 5 PRELIMINARY WORK** **51**

5.1 Initial Set-up	51
5.2 Raspberry Pi Setup (Edge Device)	52
5.2.1 Raspberry Pi Setup Command	52
5.3 Hardware Prototype set-up	54
5.4 Sensor Calibration	55
5.4.1 Adjusting 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	60
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	64
5.5.2.2 Rule-Based Alert Component	66
5.5.2.3 Data Synchronization Component	68
5.5.2.4 Federated Learning Client Component	69

5.6	Setting up AWS Cloud	71
5.6.1	AWS IoT Core	71
5.6.2	Amazon Lambda	72
5.6.3	Amazon DynamoDB	73
5.6.4	Amazon SNS	74
5.6.5	Amazon Athena	75
5.6.6	Amazon Simple Storage Service	76
5.6.7	Amazon Elastic Container Registry	77
5.6.8	Amazon Elastic Container Service	78
5.6.9	Amazon Virtual Private Cloud	78
5.6.10	Amazon Quicksight	80
5.7	Federated Learning Framework Implementation	81
5.7.1	Comparison of Federated learning Frameworks	83
5.7.2	Scheduled Deployment of Federated Learning Server	84
5.7.3	Setting up DNS Name	84
<b>CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION</b>		<b>86</b>
6.1	Hardware Deployment	86
6.2	System testing	89
6.2.1	Greengrass Components Testing	89
6.2.1.1	Sensor Reading and Processing Component	90
6.2.1.2	Rule-Based Alert Component	92
6.2.1.3	Data Synchronization Component	92
6.2.2	Greengrass Components Verification Test	94
6.2.3	AWS Cloud Testing	96
6.2.4	Federated Learning Module Testing	97
6.2.5	Federated Learning Module Verification test	101
6.3	Data Visualization	101
6.3.1	High Level Dashboard	103
6.3.2	Individual Level Dashboard	104

<b>CHAPTER 7 CONCLUSION</b>	<b>107</b>
<b>REFERENCES</b>	<b>110</b>
<b>APPENDIX</b>	<b>A-1</b>
APPENDIX A	A-1
APPENDIX B	A-2
APPENDIX C	A-3
<b>WEEKLY LOG</b>	<b>114</b>
<b>POSTER</b>	<b>120</b>
<b>PLAGIARISM CHECK RESULT</b>	<b>121</b>
<b>FYP2 CHECKLIST</b>	<b>123</b>

## LIST OF FIGURES

Figure Number	Title	Page
Figure 1.1	Per capita seafood consumption volume in rural households in China 2013-2021 (Statista, 2023)	1
Figure 1.2	Precision Aquaculture Market value forecast from Year 2022 to 2030.	2
Figure 1.3	Shows a Article about “Farmers and their data: An examination of farmer’s reluctance to share their data through the lens of the laws impacting smart farming”	3
Figure 2.1	Comparison of machine learning models and algorithms in different application fields.	8
Figure 2.2	Aquasafe Structure.	11
Figure 2.3	ATLAS Architecture Layers.	15
Figure 2.4	Block diagram of smart aquaculture system	17
Figure 4.1	AWS Cloud Architecture Diagram	25
Figure 4.2	Description of Steps in Architecture Diagram	25
Figure 4.3	Prototyping Development Methodology	28
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	39
Figure 4.8	ADS1115 Analog Digital Converter	40
Figure 4.9	Arduino Uno board	41
Figure 4.10	Graphical Design of IoT Device	42
Figure 4.11	Schematic Diagram of pH Reading Module	43
Figure 4.12	Schematic Diagram of Temperature Reading Module	44
Figure 4.13	Schematic Diagram od T-Cobbler and LEDs	45
Figure 4.14	Gantt Chart (1)	46
Figure 4.15	Gantt Chart (2)	47
Figure 4.16	Gantt Chart (3)	48

Figure 4.17	Gantt Chart (4)	49
Figure 4.18	Gantt Chart (5)	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	53
Figure 5.4	CLI displaying cgroup v1 and v2	53
Figure 5.5	CLI displaying TensorFlow 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 wire coiled onto outer metal casing	56
Figure 5.9	Highlighted area on reference electrode is Potentiometer	56
Figure 5.10	Arduino IDE for reading sensor voltage	57
Figure 5.11	Buffer Solution for pH 4.01 and pH7.00	57
Figure 5.12	pH 4.01 and pH 7.00 Buffer Solution with pH Probe immersed	58
Figure 5.13	Output Voltage of pH sensor from pH 4.01 and 7.00 Buffer Solution	58
Figure 5.14	pH reading of pH sensor for pH4.01 and 7.00 Buffer Solution	60
Figure 5.15	Raspberry Pi Registered as Greengrass Core Device	61
Figure 5.16	Raspberry Pi CLI indicating Greengrass Core Software in Installed	61
Figure 5.17	Greengrass Core Device inside IoT Thing Group	62
Figure 5.18	Component List for raspberrypi_aquaFarm	63
Figure 5.19	Deployment for raspberrypi_aquaFarm	63
Figure 5.20	Snapshot of artifact for com.example.sensorv5	64
Figure 5.21	Snapshot of recipe for com.example.sensorv5	64
Figure 5.22	Local Database function	65
Figure 5.23	Temporary Database function	65
Figure 5.24	AWS IoT Core MQTT Test Client	66
Figure 5.25	Snapshot of artifact for com.example.testAlarmv2	66
Figure 5.26	Snapshot of recipe for com.example.testAlarmv2	67

Figure 5.27	Snapshot of artifact for com.example.sync	68
Figure 5.28	Snapshot of recipe for com.example.sync	68
Figure 5.29	A Part of Artifact for Federated Learning Client Component com.example.flwrCLient2	69
Figure 5.30	Part of Recipe for Federated Learning Client Component com.example.flwrCLient2	69
Figure 5.31	IoT Rules for Message Routing	71
Figure 5.32	Lambda Function intermediateProcessing	72
Figure 5.33	DynamoDB with data passed in	73
Figure 5.34	Detail of SNS Topic	74
Figure 5.35	Details of Athena Data Source dynamo-quicksight	75
Figure 5.36	Table Specified in Associated Databases	75
Figure 5.37	Athena Query Editor Testing	76
Figure 5.38	Bucket List in S3	76
Figure 5.39	Docker files that will be uploaded into ECR as image repository	77
Figure 5.40	Detail of Task Definition Created from Docker Image	78
Figure 5.41	The Federated Learning Server Running as a Task in ECS Cluster	78
Figure 5.42	Detail of Security Group configurations for inbound rules	79
Figure 5.43	Detail of Network ACL configurations for inbound rules	79
Figure 5.44	List of Datasets in Quicksight	80
Figure 5.45	Quicksight Dashboard Developer Interface	81
Figure 5.46	Federated Learning Module from Architecture Diagram	81
Figure 5.47	Schedule Task with Target Task Definition	84
Figure 5.48	Configuration of Network Load Balancer flwrLoadBalancer	84
Figure 5.49	Configuration of Target Group flwrTargetGroup	84
Figure 6.1	Hardware Deployment at Aquafarm	86
Figure 6.2	Hardware Deployment at office tank	87
Figure 6.3	DS18B20 and PH4502C Sensor immersed in Water	87
Figure 6.4	Local Occupants	88
Figure 6.5	Green LED illuminating on IoT Device	89

Figure 6.6	MQTT Test Client receiving message from subscribed topic	90
Figure 6.7	Content of Main Database (Left) and Temporary Database (Right)	91
Figure 6.8	Email Message from AWS SNS	92
Figure 6.9	MQTT Receiving Message from subscribed synchronization topic	92
Figure 6.10	Data routing section from Architecture Diagram	96
Figure 6.11	Items in DynamoDB table environmentDB1	96
Figure 6.12	CloudWatch Event Log for Federated Learning Server (1)	97
Figure 6.13	CloudWatch Event Log for Federated Learning Server (2)	97
Figure 6.14	CloudWatch Event Log for Federated Learning Server (3)	98
Figure 6.15	Federated Learning Client Event Log	98
Figure 6.16	Global ML Model Weight uploaded in S3 bucket	99
Figure 6.17	Client Model Weight uploaded in S3 bucket	99
Figure 6.18	Federated Learning Module from Architecture Diagram	101
Figure 6.19	Data Visualization Module from Architecture Diagram	101
Figure 6.20	Main Page of Dashboard (High Level)	103
Figure 6.21	Individual Tank Detail Page of Dashboard (Individual Level)	104

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 2.1	Comparison between previous aquaculture framework and proposed solution	19
Table 4.1	Specifications of Laptop.	33
Table 5.1	Comparison of different Federated Learning Frameworks	83
Table 6.1	Greengrass Components Verification Test table	95
Table 6.2	Federated Learning Steps Verification Test table	101



## LIST OF SYMBOLS

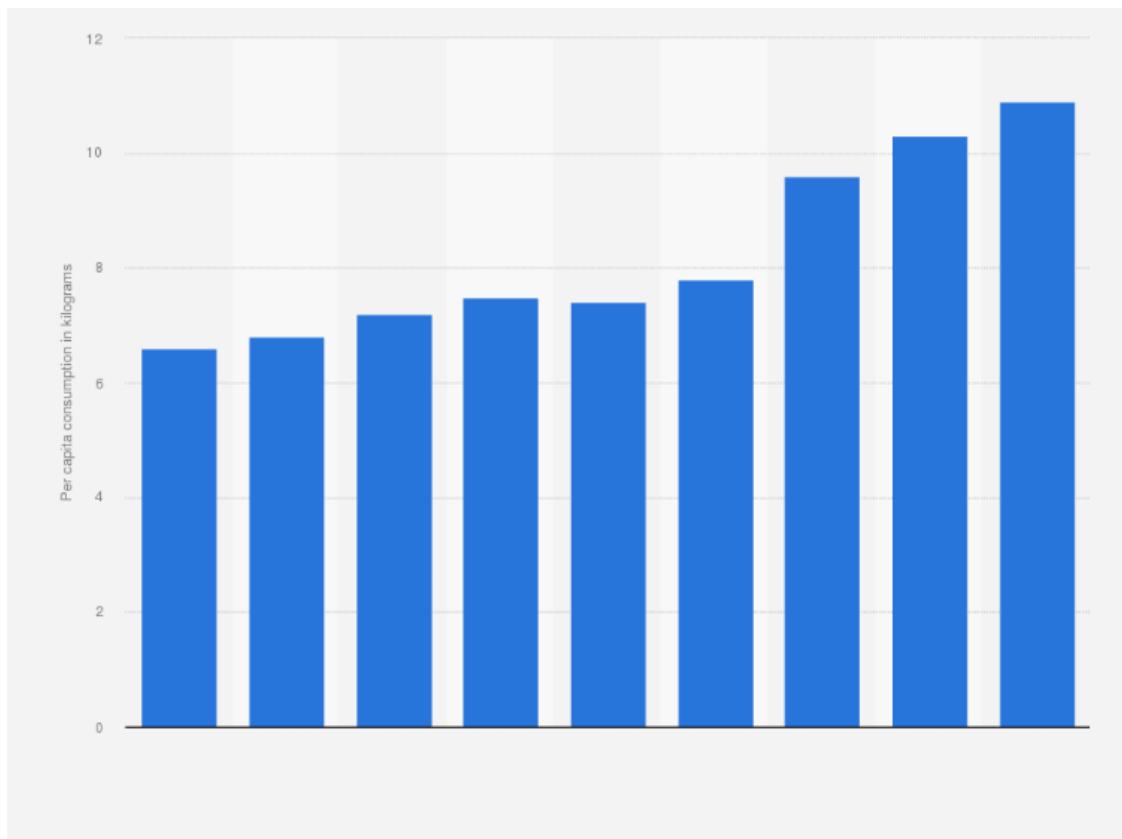
±	Plus/Minus sign
°C	Degree Celcius

## LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>UI</i>	User Interface
<i>IT</i>	Information technology
<i>IoT</i>	Internet of Things
<i>WSN</i>	Wireless Sensor Network
<i>GIS</i>	Geographical Information 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>FL</i>	Federated Learning
<i>RPC</i>	Remote Procedure Call
<i>BI</i>	Business Intelligent
<i>RNN</i>	Recurrent Neural Network
<i>ADC</i>	Analog to Digital Converter

# Chapter 1: Project Background

## 1.1 Introduction



**Figure 1.1 Per capita seafood consumption volume in rural households in China 2013-2021 (Statista, 2023)**

Since the very beginning of human civilization, aquaculture which is the farming of aquatic organisms such as fish and shellfish, has become a critical component of global food production. Sea creatures have long been a steady and preferred food source for human beings. As seen in Figure 1.1, it can be seen that the consumption of seafood per capita for each household in China from 2013 till 2021 has been increasing. It is undeniable that aquaculture plays a huge role in our lives for the ever growing demand for seafood as it is the fastest growing production sector in the world. However, the efficient management of aquaculture operations still remains a challenging risk, this is due to its requirement of precise monitoring and control to ensure the optimal growth, resource utilization and environmental sustainability is being maintained

## CHAPTER 1 PROJECT BACKGROUND

continuously and regulated. Precision aquaculture is the use of technology and data-driven approaches to optimize various aspects of fish and seafood production.



**Figure 1.2 Precision Aquaculture Market value forecast from Year 2022 to 2030 (Verified Market research, 2022)**

Based on Figure 1.2, it shows that the forecast for the Compound Annual Growth Rate (CAGR) between the year 2022 and 2030 for the global precision aquaculture Market is 13.66 percent increasing 772.95 million from 433.04 million. Whether we like it or not, the demand for precision aquaculture is on the rise and growing exponentially as time goes by.



Research paper

## Farmers and their data: An examination of farmers' reluctance to share their data through the lens of the laws impacting smart farming

**Figure 1.3 Shows a Article about “Farmers and their data: An examination of farmer’s reluctance to share their data through the lens of the laws impacting smart farming”**

While precision aquaculture offers numerous benefits, there are also several challenges which are associated with it such as data collection and management. Precision aquaculture relies heavily on collecting and analysing large amounts of data from various data sources such as sensor devices, cameras and environmental monitoring systems. Ensuring that the data which is collected is accurate and reliable is a challenging task especially in offshore or remote aquaculture facilities. Additionally, infrastructure will also be strained due to maintaining vast amounts of data which is used and requires advanced data management techniques. With that being said, a crucial problem with managing the data which has been collected is data privacy and security. Precision aquaculture involves collecting sensitive data from various sources, this data can be valuable and may contain proprietary or confidential information. The traditional way of handling privacy issues through centralized machine learning approaches has raised significant data privacy concerns and fish farmers have been reluctant to share their data due to various ownership reasons. This can be seen from an article which is published by NJAS Wageningen. Other than that, aquaculture facilities, especially those located in remote offshore areas, might have limited network connectivity, this raises issues if data needs to be transferred constantly to a central server which can be significantly impacted by network limitations for model training. Precision aquaculture operations can also vary in terms of environmental conditions and the type of species cultivated and farming practices. Data diversity and variability is beneficial for allowing models to be trained using data from many different diverse sources which will be able to reflect specific characteristics of different aquaculture facilities. This can lead to more accurate and adaptable models that caters to specific needs of

Bachelor of Computer Science (Honours)

Faculty of Information and Communication Technology (Kampar Campus), UTAR

## CHAPTER 1 PROJECT BACKGROUND

facilities. When only one source is used for data transfer and modelling, it easily becomes a single point of failure which is the same case with centralized approaches. Lastly, with the usage of a centralized model for machine learning, the model will not gain as much insight for precision aquaculture operations compared towards collaborative learning which allows for multiple facilities to contribute their own models but this is again faced with the recurring issue of data privacy and confidentiality which has been mentioned above. If we are able to solve these issues, particularly regarding data privacy, the machine learning model training for precision aquaculture can be further improved whilst avoiding issues concerning privacy, computational bottlenecks and resource intensiveness.

With that being said, the precision aquaculture industry in today's time is still heavily reliant on human labour for performing certain repetitive tasks manually. The actions that are performed by the farmers such as data collection through visual observations, or manual interpretations of data collected by oxygen meters for observing the conditions of fish. These collected data are then used for decision-making tasks in the form of operations on the farm but they are still susceptible towards risks such as errors in data by human collection, data leaks, incomplete data sets and outdated information. All of these risks are inevitable and can cause immense loss for the time and money invested by the farmers. The heavy reliance on human labour can also affect aquaculture farms if there is a decrease in human labour available such as during pandemic where the government implemented the lock down of foreign and local labour in Malaysia. As a result, the aquaculture industry will be heavily affected as well as the chain of supply for fish due to the lack of production capacity.

## CHAPTER 1 PROJECT BACKGROUND

### 1.3 Motivation

The various issues and challenges which are faced by the aquaculture industry all have potential to be solved through the use of IoT devices and cloud technologies. Precision aquaculture farms can be operated more efficiently through the help of smart technology to aid in their operations for monitoring and data collection tasks. When the process of aquaculture monitoring is streamlined and autonomous, the production of fish will also increase in both quality and quantity. However, there is still a lack of personnel working on solutions for this industry which is appropriate for use in aquaculture farms. The existing knowledge and technology for federated learning frameworks have yet to be used together with IoT in order to resolve the issue of data sharing and model training. Once these technologies are developed together, precision aquaculture for collecting data of regarding the water quality and living conditions of fish, real time fish health status as well as accurate data forecasting through training models trained through federated learning. The amount of labour which will be needed to operate a farm would also decrease significantly and multiple farms will be able to benefit from the implementation of federated learning to train their data knowing it is still secure and confidential. To accomplish this goal, an IoT-Cloud based solution which implements a federated learning framework is proposed for use in precision aquaculture for collecting real time data through IoT sensors that are located across multiple edge computing environments. Each edge computing environment will be able to collect and process real time data autonomously and locally at their own farms. The additional use of cloud architecture will also drastically compromise for the amount of computing power as well as storage that is needed for performing federated learning tasks and storing the data processed, real-time data visualization can also be achieved through the tools available in the cloud.

## CHAPTER 1 PROJECT BACKGROUND

### 1.4 Impact, significance and contribution

The aim of this project is to make use of various IoT sensors, cloud technologies and frameworks to reduce the need for human labour involvement in aquaculture operations while ensuring data confidentiality at the same time. By deploying sensors at each edge computing environments, the probability of fish populations infected by diseases can be reduced since the living conditions of the farms are constantly monitored. The accuracy of data collection is also increased compared towards traditional manual sampling which is not real time and unable to reflect the actual current states of the farm conditions. Furthermore, the use of federated learning models for analysis of collected data can help in forecasting to reduce the need for external decision making done by personnel on site for faster results.

At each aquaculture farm which is considered as an edge computing environment, the data collected by sensors will be stored locally and processed, this reduces latency since analysis and processing is done closer to the point it was generated. The data will only be transferred into the cloud when there is a connection to the network present, hence data will not have to traverse directly into the cloud. This is beneficial for aquaculture farms which are usually located at rural areas that have poor or even no network connectivity. The benefits of this is less bandwidth consumed for data transmission, lower costs for network connection and smaller size of data sent to cloud. As for the cloud, services such as compute services, database storage and machine learning tools are provided for the farmers to maintain their infrastructure.

The data which has been pre-processed will be stored locally inside the edge compute devices as well as synced into the cloud storages for backup. Machine learning models will also be deployed for model training to each edge environment for training their own models with their own data. This is all done through a federated learning framework approach for ensuring the data privacy is kept between the farms whilst still allowing the sharing of pre-trained models to be shared eventually. Once the models have been locally trained, they are transferred into the cloud where a central machine learning model is aggregated and updated with each of the new models received from the edge environments. This newly updated model will then be sent back to the edge environments for further model training, this process is repeated to eliminate the use of traditional centralized machine learning models where there are issues of accuracy and lack of training data.

In conclusion, the proposed solution of IoT cloud and federated learning for precision aquaculture will lower the costs of operations for aquaculture farms by reducing the amount of



## **CHAPTER 1 PROJECT BACKGROUND**

labour costs, cloud services also operate based on “pay-as-you-go” practices and do not incur additional upfront costing. The scalability of the farms will also increase with the use of cloud services to scale up or down easily depending on the size of the farms. Data driven decision-making will be achieved by this project to increase the efficiency and boost the production of aquaculture so that it meets the market demand.

### **1.5 Report Organization**

This report will consist of a total of 6 chapters, the first chapter will be about the introduction of this report, components such as motivation, contribution and problem statement are described in this section. Following by chapter 2 is the Literature Review where previous work and systems will be reviewed, and their features are compared with the proposed solution. The previous work will be organised by the date of systems introduced from oldest to latest to show the evolution of technology used for aquaculture. In chapter 3, project scope and objectives will be discussed further. Chapter 4 will be about the Methodology, which is used and the overall System design, the project workflow as well as the tools used in this project will also be elaborated. Chapter 5 will discuss about the Preliminary Work about the System Implementation and the results will also be shown as well. Finally, chapter 6 will be the conclusion of this project where all the sections are summarized and an overview of the latest state is provided.

# Chapter 2: Literature Review

## 2.1 System Review

### 2.1.1 Application of machine learning in intelligent fish aquaculture: A review

Application	DT	NB	SVM	ANN	KNN	EL	Machine vision	Other	Best
Biomass detection of fish	Size estimates			CNN,R-CNN, Mask R-CNN			Stereo Vision System		CNN:94.31%
	Weight estimates Count		SVM LS-SVM	CNN,BPNN CNN		RF	MV		CNN: 95.06% LS-SVM:97.40%
Recognition and classification of fish	Fish recognition		SVM	CNN,Faster R-CNN, R-CNN + LSTM CNN				YOLOv3, GMM, KDE, ViBe+BS	CNN-SVM: 99.45% CNN: average CV of 8.89%
	Age detection			CNN					CNN: 98.90% CNN: 99.70%
Behavioral analysis	Sex identification Fish species classification	DT NBM	SVM	CNN,BPNN,NIN	KNN	AdaBoost		YOLOv3	ANFIS: 98% CNN: 91.935% RNN: 89.89%
	Feeding behavior Group behavior Abnormal behavior		SVM	ANN,CNN,RNN CNN RNN			MV	ANFIS	
Water quality parameters prediction	Univariate prediction		SVM	ANN,RNN,LSTM, GRU,DBN,CNN				BMA	DBN
	Multivariate prediction			ANN,CNN,LSTM, SRU,ELM,Deep ESN, WNN		RF, XGBoost			RNN (LSTM)

**Figure 2.1 Comparison of machine learning models and algorithms in different application fields [22]**

In this research paper, the application of various machine learning algorithms in intelligent fish aquaculture is reviewed during the past 5 years since the paper was published. These applications mainly consisted of fish biomass detection, identification and classification of fish, behaviour analysis and water quality parameter prediction. The application models discussed in the paper were based on data acquisition, processing and algorithm implementation. Firstly, datasets were mainly acquired from online public datasets in formats of fish image and water quality parameter time series. As for data processing, S. Zhao et al [22] made use of image processing technology to preprocess images for regions and features of interest and also pre-processed time-series data to remove noises and obtain stationary data. As for machine learning algorithms, CNN models were principally involved in most applications along with other learning models such as SVM, BPNN, KNN, RF, AdaBoost, YOLOv3, RNN and XGBoost which can be seen in Figure 2.1.

## **CHAPTER 2 LITERATURE REVIEW**

### **Application of biological information evaluation**

Starting with application of biological information evaluation, the machine learning algorithm which had highest prediction precision for fish body length estimation was a CNN model which was able to resolve the issue of overlapping fish and non-overlapping fish. As for weight estimation, BPNN and CNN models had higher accuracy and better performance when compared to others such as SVM and RF. For counting, the accuracy of SVM was higher compared to CNN models when both were used to estimate the numbers with the results of SVM being 96.64%.

### **Application in species detection and classification**

As for species detection and classification, combination models along with CNN were mostly employed which indicates the wider used of CNN models in species classification. In parallel, SVM, NBM, KNN, YOLOv3, BPNN and AdaBoost models were also conducted where all achieved good classification results with accuracy rates over 90%. Regarding gender identification, CNN and DT models identified species with accuracy rates over 95%. With that being said, CNN models were also used in predicting fish age in otolith images and applied in species detection for fish information as well as shrimp and crab detection. Additionally, YOLOv3, SVM, Faster R-CNN and BM models were all used for fish detection achieving good results.

### **Application of behaviour analysis**

Behaviour analysis was used for evaluating fish behaviour and conducted with SVM, CNN, RNN and ANN models. CNN and CNN-LTSM models were integral for feeding behaviour analysis by effectively evaluating the appetite of fish. On the other hand, SVM model accurately quantified the feeding behaviour with accuracy of 0.95. When the set condition threshold was 0.5, ANN model accuracy for fish feeding intensity reached 100%. In group behaviour analysis, CNN model could determine the behaviour state of fish population and group behaviour. Finally for abnormal behaviour analysis, RNN model recognition achieved and accuracy of 89.89% and detected abnormal behaviour in fish through machine vision.

## **CHAPTER 2 LITERATURE REVIEW**

### **Application of water quality prediction**

In the application for water quality prediction, deep neural networks were extensively adopted with RNN model improving the model crucial in water quality prediction due to its ability to process time series. For dissolved oxygen prediction, LSTM, DBN, GRU and CN models were applied in combination with other models for DO content prediction, this in turn improved data processing capabilities and network depth thus demonstrating good prediction performance. In the case of water quality multi-parameter prediction and data loss, ELM and LSTM effectively predicted the content of water quality parameters in short term. Moreover, hybrid-based RF and XGBoost models predicted multi-parameter water quality. In the detection of toxic metals, ANN and LSTM-RNN predicted the content of heavy metals and COD with good detection results.

### **Strengths**

Machine learning technology provides relatively efficient methods for data processing in aquaculture, real-time monitoring, decision management and information extraction. The models developed will be more suitable with the improvement of aquaculture information transparency and quality of sensors and cameras. Neural networks and advanced complex machine learning algorithms can be used for learning tasks which aid in applications in aquaculture such as fish detection, classification, environmental analysis, behaviour analysis and disease prevention. Machine learning algorithms are able to work with vast amounts of data and produce relatively high precision and accuracy in predictions, hidden correlations between datasets could be uncovered to produce interesting insights to help farmers make decisions.

### **Weaknesses**

With the benefits brought from machine learning involvement in aquaculture, there were also shortcomings such as data imbalance and variability. This is due to the issue that the datasets covered in the project might only come from one data source and lack of variability. Incompleteness of datasets, missing data, low quality images and uncertainty of unique factors can cause imbalance and biases during machine learning implementation phases and produce inaccurate predictions. This problem is further highlighted by the fact that machine learning

## CHAPTER 2 LITERATURE REVIEW

algorithms tend to perform well on majority classes and struggling to classify minority class instances on the other hand. The variability of data due to external factors might challenge machine learning models ability to generalize well under different conditions.

### 2.1.2 Aquasafe: A Remote Sensing, Web-Based Platform for the Support of Precision Fish Farming

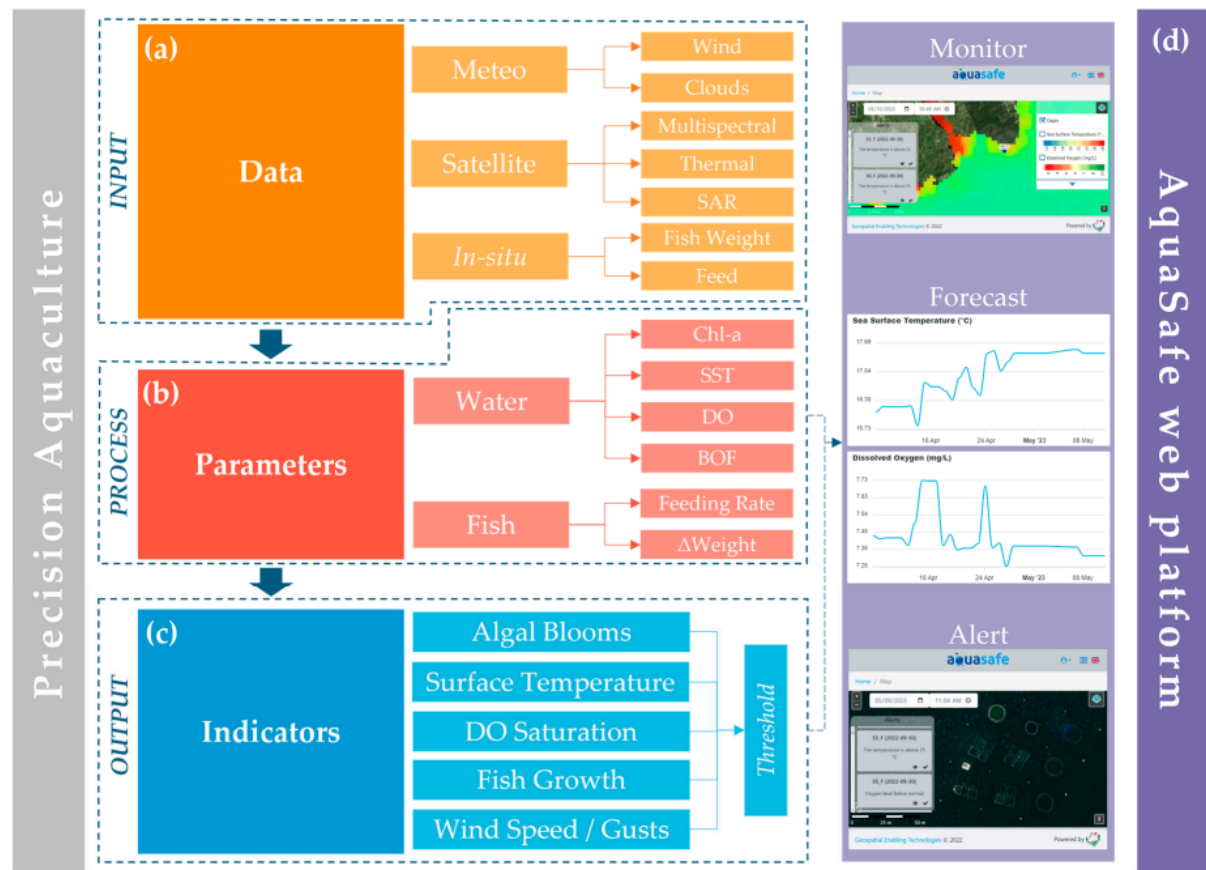


Figure 2.2 Aquasafe Structure [23]

This research examined how satellite technology could assist in the issues from fish farming and presented a web-based platform with the name “Aquasafe” that serves as a decision support system and monitors fish farming operations to provide warnings for potential risks. Satellite data, *in-situ* measurements, meteorological information were combined to farm operators insights and alerts for effective fish farm management. From Figure 2.2 we can see the four main pillars of the system being input data, estimated parameters, indicators and the web-based platform. Aquasafe utilizes data from satellite for thermal, SAR and multispectral data, in-situ measurements such as farmed species and growth rates and meteorological data like winds, cloud coverage and weather events were also put to use. As for the parameters, there were 2

## CHAPTER 2 LITERATURE REVIEW

categories which were those pertaining to water and the surrounding environments and those which pertained towards the farmed species and the farm itself. Parameters that effect the water were considered most crucial for example dissolved oxygen, water temperature and chlorophyll-a. As for parameters related to the farm, feeding rate, fish biomass and oxygen consumption were included. Five indicators were developed according to the parameters affecting survival and growth of farmed species being concentration of algae, sea surface temperature, saturation of dissolved oxygen, fish growth and wind speed and gusts. Appropriate thresholds were set to alert system administrators for potential threats. The platform for Aquasafe is based on a 3-tier architecture with the data tier supported by spatially enabled Relational Database Management System (RDBMS). The application tier implements Application Programming Interfaces for deployment of data curation, harvesting and product deriving mechanisms. Geospatial services based on Web mapping Services and Web Feature Service were also included. Finally, the web interface which is the presentation tier harnessed API's capabilities to derive the functionalities. The platform was developed with open-source software's such as PHP, Javascript libraries and PostgreSQL.

### Strengths

Aquasafe's enables real-time monitoring of wide range of parameters for farmers and managers whom might not have access and experience with web applications through a user friendly platform. Its strength lies in the ability to integrate geospatial information along with in-situ data for the comprehensive monitoring of environmental parameters, machine learning algorithms and remote sensing to provide real time information on environmental information and potential risks to aquaculture operations in fish farms. It aims to collaborate innovative technologies into the platform for continuous and comprehensive monitoring for more efficient and sustainable management of aquaculture facilities to provide opportunities for development of collaborative approaches and practices in marine aquaculture.

### Weaknesses

Aquasafe's approach of using a web-based platform for real time monitoring of fish farms and facilities might raise concerns regarding data and privacy of aquaculture facilities whom are using their service. Since satellite data and *in-situ* measurements are being used to help provide

## CHAPTER 2 LITERATURE REVIEW

real-time monitoring, some users might be worried about the privacy of the data used in their facilities since all the data are stored at a centralized RDBMS server for Aquasafe. Sensitive aquaculture data recorded might be vulnerable towards security breaches on the servers, some users might not want their data to be shared amongst others due to privacy and concern reasons too. This issue might scare users particularly large-scale facilities to use this system since the risk of their data falling into the hands of others, the lack of ownership of their own data outweighs the benefits brought by the suggested system.

### 2.1.3 Precision aquaculture by Fearghal O'Donncha and Jon Grant [21]

This research paper focused on DeepSense which is a big ocean data innovation environment powered by IBM with key components in commercialization of IoT technologies toward better management of fish farms. Hundreds of real-time underwater acoustic sensors were deployed at multiple fish farms take measurements daily and communicated to IBM cloud continuously informing animal and environmental conditions. A range of machine learning and mechanistic models for managing aquaculture operations were developed to simulate real-time predictions on fish biomass, health and mortality based on environmental stressors. Data assimilation was also made use in combination with mechanistic models to maintain accuracy. An IoT network was developed to integrated and complemented with a model management framework that enables tracking of functionalities, automatic subscription to data streams and relationships between different models. The objective was the supply of nutritionally appropriate feed at a rate and frequency which maximizes uptake by the fish while minimizing the environmental impacts together with informing health intervention practices.

#### **Strengths**

Precision aquaculture aims to improve the environmental sustainability and increase the production efficiency of farmers by implementing IoT devices, machine learning algorithms and sensor technologies. The impact of environmental aquaculture on feed management and waste pollution is reduced through optimization, risk of disease outbreak is also minimized at

## CHAPTER 2 LITERATURE REVIEW

the same time resulting from the analysis of machine learning. With the aid of cloud-based platforms, fish farmers will no longer require on-site computing infrastructure to monitor marine aquaculture. IBM cloud allows the users to access a wide range of cloud services such as data visualization, machine learning models and computing power without the hassle of purchasing or maintaining their own physical infrastructure.

### **Weaknesses**

Precision aquaculture's approach of using sophisticated technologies such as satellite-based monitoring, drone-based imaging and sensors requires significant upfront costs and investments particularly for the number of sensors deployed for real-time monitoring. Network connectivity between the sensors to the web-based platform also plays a crucial part for real-time monitoring and requires heavy amounts of resources to ensure reliable connectivity at all times. The system also requires large amounts of servers and infrastructure in order for it to support the vast amounts of data being stored and processed, this contributes another great amount towards the cost of the system and makes this concept less suitable for smaller-scale farmers due to the high costs for subscription towards its service. Interoperability also poses as a significant challenge which extends from legacy sensors data in onboard data loggers towards modern sensors reporting in proprietary format to dedicated cloud platforms.



## 2.1.4 ATLAS: Internet of Things Platform for Precision Aquaculture

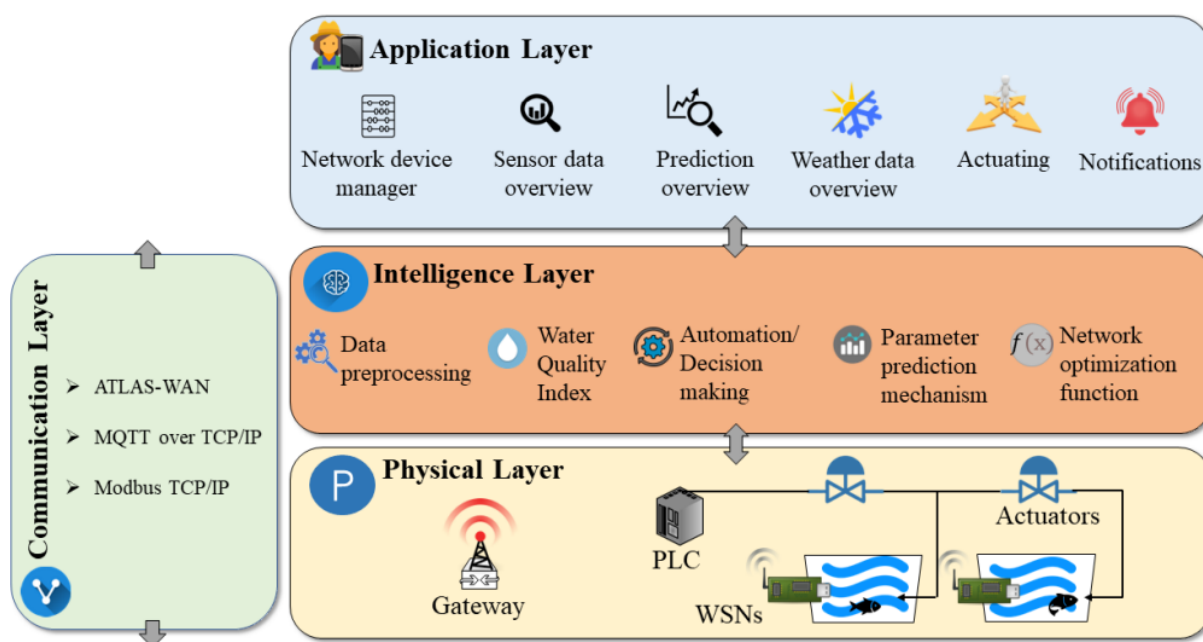


Figure 2.3 ATLAS Architecture Layers [24]

In this research paper, the authors presented an approach to capture heterogeneous requirements of an automated aquaculture monitoring system with the aid of IoT networks, cloud services, wireless sensor nodes (WSNs) and actuators called “ATLAS” [24]. The system mainly focuses in data collection and transfer through dedicated communication protocols, extracting and processing received data through machine learning algorithms, optimal decision making via optimization algorithms and easy-to-use interface for reflecting state of aquaculture. The ATLAS system architecture starts with WSNs measuring water quality parameters then transferring the data to the cloud server for processing and storing purposes. The data is then processed, stored and analysed in servers for decision making and reflect these actions through actuators. 4 layers have been derived for the system architecture shown in Figure 2.3 starting with the Physical layer which refers to the physical entities i.e., WSNs, gateway, actuators and programmable logical unit (PLC). Next is the communication layer which focuses on the protocols that is used for all communication purposes, they are in line with resource allocation performed by the next layer which is the intelligence layer. In the intelligence layer, all intelligent operations such as forecasting status of aquaculture and extraction of water quality index through utilization of data received from sensors, these services are real-time and assesses the states of aquaculture for optimal decision making. Lastly, the application layer contains interface for the ATLAS system which is accessed through internet. The users will be

## CHAPTER 2 LITERATURE REVIEW

informed regarding system conditions and visualizations of aquaculture conditions as well as manage network devices for actions and receive notifications.

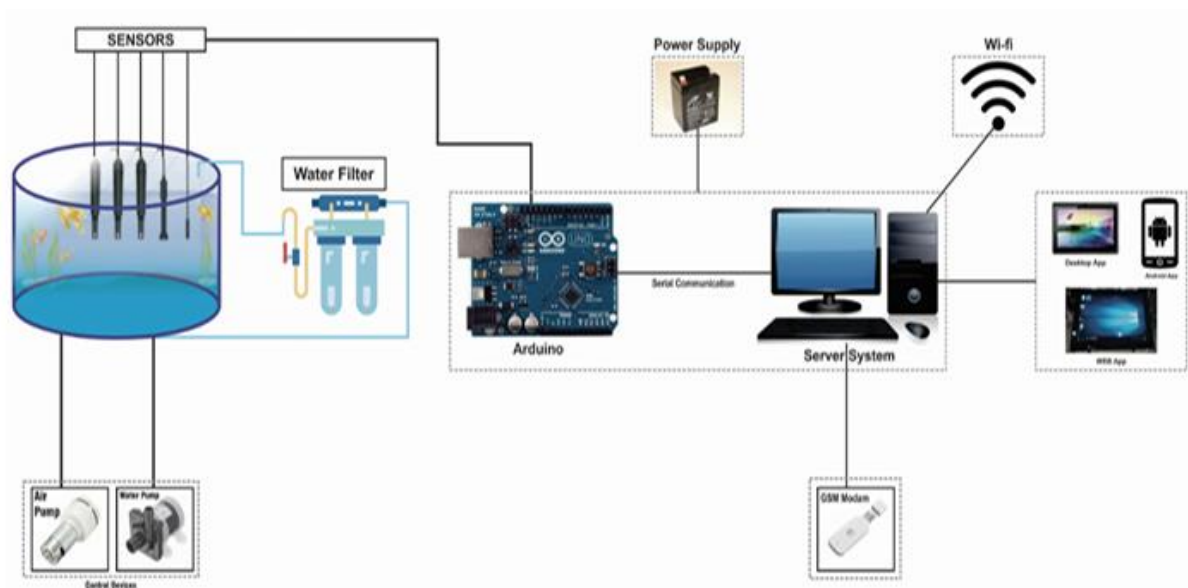
### **Strengths**

For this IoT precision aquaculture system, it made use of cutting edge technologies such as WSNs, PLC and actuators and communicates through a gateway towards a cloud-based server for processing. The implementation of a centralized intelligence layer for operations such as automation and decision making could potentially save up costs which is traditionally used in manual labour for monitoring. ATLAS also has a network optimization function which aims to adjust resources for supporting various functions in order to minimize power consumption of network nodes. This function regulates the transmission probability and transmit power of WSNs and guarantees performance of a single WSN by providing it with higher priority which makes the whole system much more customizable to suit the preference of its users.

### **Weaknesses**

Although the ATLAS precision aquaculture has its strengths, there are still some weaknesses found in this system. The main weakness of this system lies in the use of WSNs. Even though WSNs are easy to deploy, they are extremely vulnerable to malicious security attacks as they lack robust security systems. At the same time, even though WSNs can be large and span wide areas, this makes them a bigger access point for malicious attackers. Other than security issues, the network of WSNs are inexpensive since they require to be deployed frequently which leads to network latency and routing overhead problems due to the periodical toggling of power on the sensor nodes for operational lifetime extension.

### 2.1.5 IoT Based Smart Fish Farming Aquaculture Monitoring System



**Figure 2.4 Block diagram of smart aquaculture system [25]**

In this research paper, the authors proposed an IoT based smart aquaculture system to collect real-time data which will be used to optimize the production of aquaculture [25]. Low cost short range wireless sensors network was used in the system to automate the measurement and real-time monitoring of various different parameters on fish farms such as water pH levels, temperature and the behaviour of fish. The aim of the system was to help cope with the drawbacks of manual water testing which often resulted to the increase of death rate of fishes and also made significant impacts on the growth rate and internet access problems encountered in field areas. A range of different sensors were integrated, each having a unique purpose along with internet technologies ultimately combined with a user-friendly interface that will be used for interactions between the users through smartphones or website for monitoring purposes. GSM modem was used for sending messages to the operators when water quality exceeds a certain threshold through applications in their devices when internet connection was available. With that said, computers were also used to serve as servers to analyse the outputs from the sensors and act as a database at the same time for storing these data. The processed data will be analysed and then visualized to be displayed inside the mobile application or website for assisting decision making operations done by fish farmers.

## CHAPTER 2 LITERATURE REVIEW

### Strengths

The strength of the proposed system is that it allows fish farmers to have remote access to their farms for real-time monitoring of different parameters through the internet. A rule-based alarm is also very beneficial to alert the farmers regarding certain water parameters which have exceeded thresholds to avoid any issues that might affect the fishes. A dashboard for visualizing the real-time monitored data can directly impact the decision making process of the fish farmers by allowing them to make better timely decisions through accurate means. The cost of the system is also reduced since not much infrastructure will be consumed for processing and storage usage.

### Weaknesses

The proposed system has several weaknesses, the first one being it has a single point of failure since all the data is being stored inside one single device acting as a server and database. If the network connectivity from the sensors to the server is affected, real-time monitoring of parameters cannot be effectively achieved, this might lead to negative impacts on the fishes on the farms since the farmers will not be able to make critical decisions in time. Other than that, security concerns are also big risks concerning the system since local servers are used and can be vulnerable to cyber-attacks or physical damages. Lastly, the processing power and storage capacity of the system is very limited which reduces the scalability of the system. This becomes an issue when the scale of the fish farms increase and the system is not capable of handling the processing and storage operations anymore.

## CHAPTER 2 LITERATURE REVIEW

### 2.2 Comparison between previous aquaculture framework and proposed solution

Existing System Features	2.1.2	2.1.3	2.1.4	2.1.5	Proposed Solution
Scalability	Yes	No	Yes	No	Yes
Reliability	Yes	No	No	No	Yes
Cost Effective	No	No	Yes	Yes	Yes
Wireless Connection	Yes	Yes	Yes	Yes	Yes
Cloud based	No	Yes	Yes	No	Yes
Data Privacy	No	No	No	No	Yes
Real-Time data monitoring	Yes	Yes	Yes	Yes	Yes
Machine learning implementation	Yes	Yes	Yes	Yes	Yes
Decentralized data collection	No	No	No	No	Yes
Reduced communication cost	No	No	No	Yes	Yes
Data variability	Yes	No	Yes	No	Yes

Table 2.1 Comparison between previous aquaculture framework and proposed solution

### 2.3 Proposed Solutions

Having reviewed several precision aquaculture systems, there are several obvious limitations with their proposed design. One of the main concerning issues is that most of the reviewed systems are using a traditional centralized data collection approach, this approach has its benefits for certain purposes but also comes with its fair share of issues. One of the problems which is associated with centralized servers and data collection is the lack of data diversity and variability due to the limitations of geographic or environment constraints. This problem can affect the model which might cause limited generalization, outliers, biased representations, and ineffective anomaly detection due to the lack of variety in data sources and parameters used. This can be solved in the proposed solution through federated learning by aggregating data from multiple data sources each having their own local dataset and models training will also be done locally. However, another issue needs to be considered when diversity of data sources is introduced into the system, that is the lack of data privacy and security. This concern arises when sensitive data regarding aquaculture operations in different facilities is being collected and stored inside a centralized server, the facility operators might raise concerns regarding the level of privacy since their data is considered valuable. Centralized approaches are also especially vulnerable towards security concerns such as security breaches, data leakages and unauthorized access. In the proposed system, this problem can be easily avoided since federated learning is a decentralized approach so the data from different sources aren't stored under a single server, each aquaculture facility has complete ownership over their own data maintaining data sovereignty. Federated learning frameworks aggregate model updates which have been adjusted from local datasets and use encrypted communication to increase security measures, this reduces the risk for security breaches greatly.

With that being said, scalability is also a big issue found inside the reviewed systems, some are only capable of catering towards small-scale users and struggle to cope with the increasing data volumes and complexity, infrastructure from these systems will easily be exhausted or requires significantly more cost to upgrade or scale up. For the proposed system, federated learning's decentralized nature allows it to better cope with increasing numbers of devices and data sources before reaching its bottleneck. This is because computation is being distributed to local servers and devices, all processed local data will then contribute towards the overall model hence only more local devices will be needed to scale up. Latency and overhead can increase for systems using traditional approaches when more data is being transferred, this might affect the ability of systems to conduct crucial real-time monitoring of

## CHAPTER 2 LITERATURE REVIEW

aquaculture operations. The proposed system can avoid this issue by reducing the amount of data being transferred since only model updates need to be passed into the main server and most of the processing is done locally. Lastly, significant data collection and transmission costs might become a burden when using some of the reviewed systems, this is due to the need to continuously maintain and upgrade the sensors and data servers in order to sustain operations and different functionalities. The proposed system which implements federated learning framework, the same concepts apply towards this issue since lesser data will be transferred due to the compression before being sent as updates, another characteristic of federated learning is the ability for adaptive learning rates where devices with more data can be assigned smaller learning rates in turn requiring less updates for the data to achieve convergence. The amount of transmission cost is significantly reduced and bandwidth requirement will also be lowered.

# CHAPTER 3: PROJECT SCOPE AND OBJECTIVES

### 3.1 Project Scope

From the problem statement stated above, the implementation of a system used for precision aquaculture through traditional approaches is inefficient and can be further improved. With the traditional precision aquaculture system architectures, infrastructure is often exhausted and incurs unnecessary costing due to increasing amounts of data collection and processing. Other than that, concerns such as data privacy and security risks are also major issues found in traditional precision aquaculture systems such as centralized, hybrid, edge computing and wireless sensor networks. The aim of this project is to develop a federated learning framework in order to replace the conventional methods for machine learning model training in precision aquaculture systems. This is done so to improve the privacy of data collected and overall security of the system, transmission cost and latency should also be reduced through autonomously this implementation. This project mainly focuses on prawn as the aquaculture product, sensors and IoT devices will be used for obtaining real-time aquaculture data such as video image and water parameters from each edge autonomously. The data which is collected is then pre-processed locally at each edge to train local machine learning models which have been distributed by the main server, this is done to help aquaculture farmers do predictions for water quality trends, monitor and alert for threshold breaches and support in decision making by providing recommendations. The updated machine learning models are then sent to the main server which is residing inside the cloud. The main machine learning model is trained and data will be stored inside the cloud for visualization purposes in order through dashboards which are interactive in order to aid the aquaculture farmers in further aquaculture operations.



### 3.2 Project Objectives

- **To develop a federated learning model that is suitable for precision aquaculture.**

The main objective of this project is to develop a workable federated learning framework which will be used for precision aquaculture. This will be done by firstly having a generic baseline machine learning model which will be stored at a central server, copies of the model will then be shared around with multiple clients which are located at edge devices (aquaculture facilities) to be used for training with their local dataset containing water parameters and video images that they have generated from the data which is collected through the sensors and cameras+. The final outcome will be a workable federated learning framework.

- **To compare various approaches in enabling federated learning within an IoT-Cloud architecture.**

Another objective for this project is to compare between different approaches for implementing a federated learning framework. There are various approaches and variations which can be used for a federated learning framework ranging from vertical, horizontal, federated transfer, hierarchical, cooperative federated learning and much more. This objective aims to determine the most suitable approach to leverage collective knowledge in IoT-Cloud implementation through reasoning along with comparison with others. In order to achieve this objective, a sample of federated learning framework variations which are better suited for our project's datasets and requirements. After that, the selected approaches should be trained with the datasets, the final results of each approach should be assessed by comparing between factors and determining the most suited approach.

- **To develop an intelligent cloud-based dashboard interface for data visualization.**

This objective aims to develop an intelligent cloud-based dashboard interface that will be used for data visualization and analysis purposes. The dashboard should leverage the use of cloud infrastructure to be the primary data source so that it can provide near real-time visualization for the authorized users. The dashboard should also be interactive by allowing the users to explore the general information regarding each tank residing in the aquaculture farms which includes the tank status and amount of breach cases. Filters should also be added to the dashboard to go into detail for each tank to display the

## CHAPTER 3 PROJECT SCOPE AND OBJECTIVES

trends for water parameters. Additional features of the dashboard can be predictive analytics which are found in BI tools for better data analysis. This dashboard should support the users by improving their decision making through data-driven techniques.

### 3.3 Contributions

The contributions of this project seen from the perspective of researchers and workers in the aquaculture industry consists of many different benefits towards their daily operations and research areas. Some of the potential benefits gained from this project is better understanding regarding precision aquaculture knowledge, this is the result of collective insights provided by federated learning frameworks. With the help of collective understanding of best practices in aquaculture management through the sharing of machine learning models from different aquaculture facilities, aquaculture farmers will be able to implement better management and monitoring of their aquaculture farms. Along with this benefit, the privacy of their data also remains safe since none of the sensitive data is being shared with other, all data remains localized and models are also trained locally. Hence, aquaculture farmers still retain full ownership of the data collected inside their own aquaculture farms. Another great benefit is the reduce in costs for precision aquaculture management systems, this is due to the reduced network load from transferring data for real-time monitoring and communication. Only model updates from each aquaculture facility will be updated into the main model which lowers the need and amount for data transferring. Lastly, researchers who are interested in precision aquaculture can also benefit by joining their research models together with others since their proprietary data will not be shared.

# CHAPTER 4: Methods and System Design

## 4.1 System Architecture Diagram

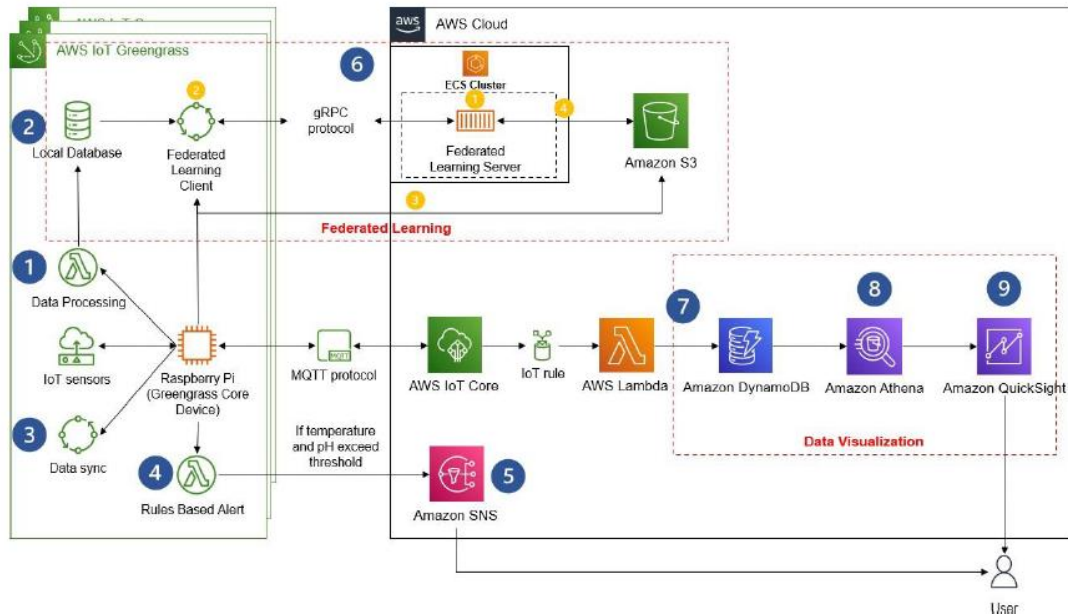


Figure 4.1 AWS Cloud Architecture Diagram

- 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

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.

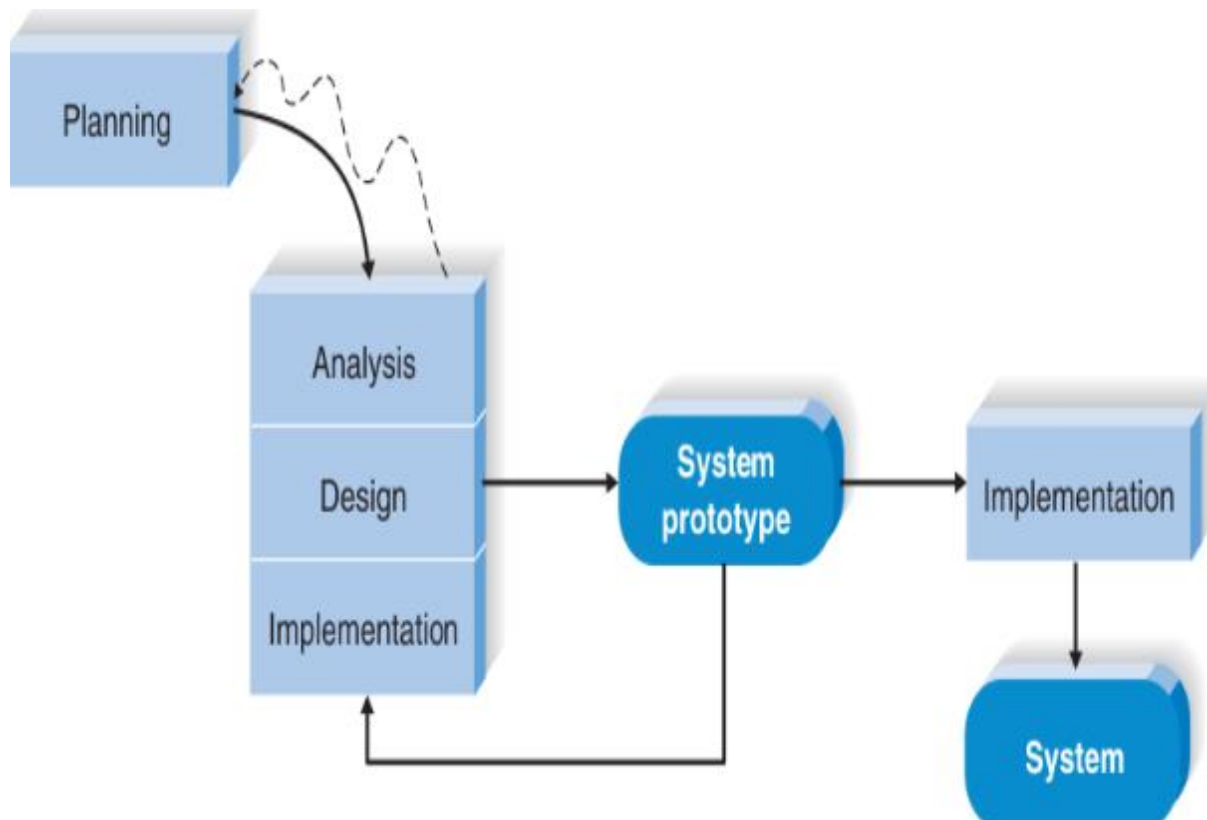
Starting from the edge environment, water parameter is obtained by using various type of IoT sensors. A raspberry pi is register as a **Greengrass core device** to run **Greengrass core software** which facilitates the deployment of lambda functions and custom components/algorithm. The local lambda function 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 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.

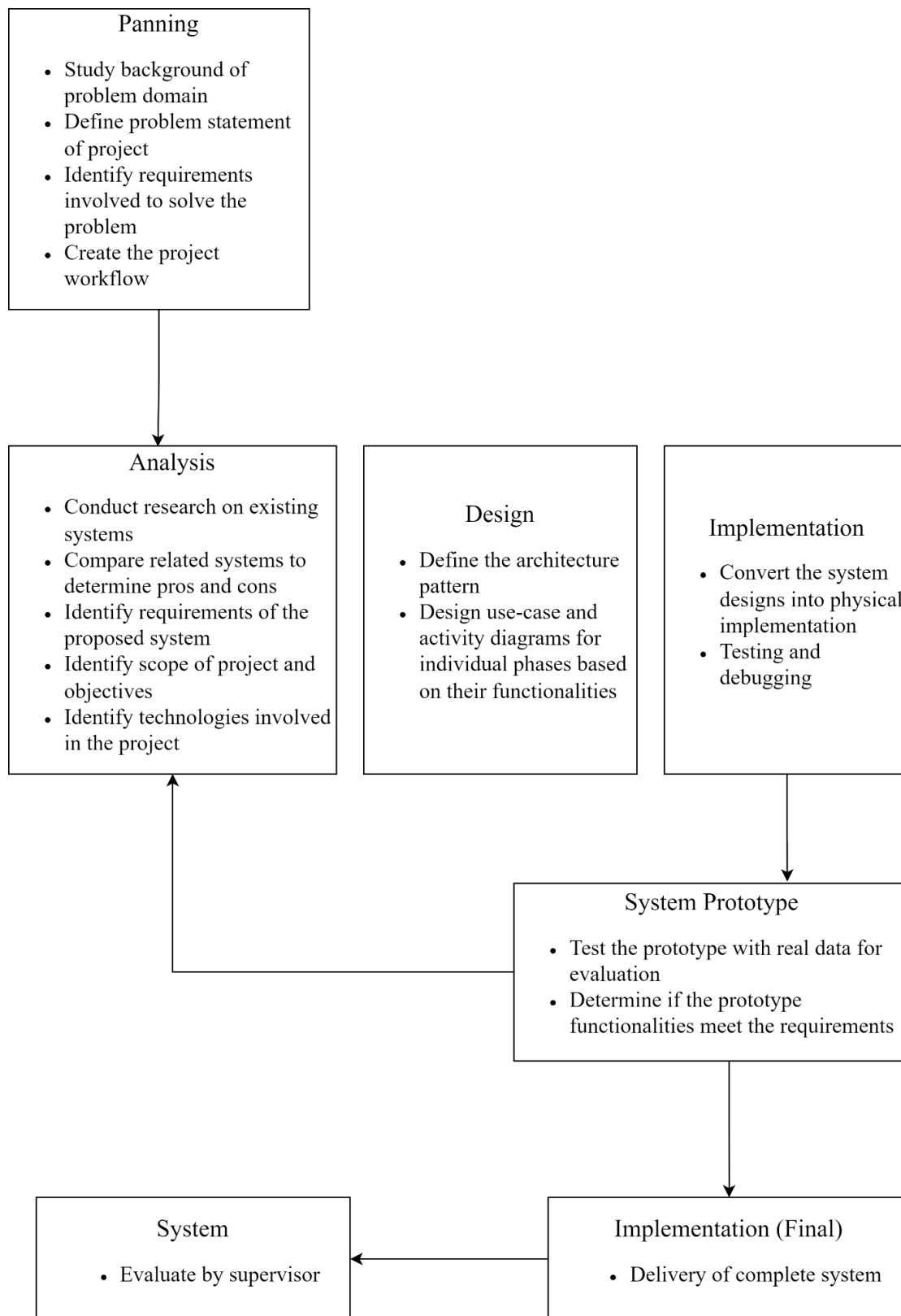
## 4.2 Methodology



**Figure 4.3 Prototyping Development Methodology**

The methodology which will be chosen to develop a federated learning framework for precision aquaculture RAD prototyping methodology. The system prototyping methodology performs its analysis, design and implementation phases concurrently by developing versions of the system for users to evaluate and give feedback. These versions are called prototypes and are repeated in a cycle continuously to explore design alternatives, in each cycle the prototype is refined. The cycle continues until enough functionality is provided from the prototype and can be implemented.

### 4.3 Project Workflow



### **4.3.1 Planning Phase**

The project starts off with the planning phase. In this phase, study is done on background regarding the problem domain of the project. After completion of background study, problem statement of the project will be formulated and methods involved for solving the problems will also be identified together. Then a project workflow can be constructed to get a better understanding of each phase inside the system development life cycle.

### **4.3.2 Analysis Phase**

After finishing the planning phase, the analysis phase begins by conducting research on exiting systems for identifying any possible chronological trends. Comparisons between these systems is then made to acknowledge their characteristics along with the pros and cons of these systems. Then the requirements for the proposed system are identified based on the advantages of certain reviewed systems and different approaches are also substituted in to replace the weaknesses. From the requirements gathered, a project scope can be identified along with the technologies that will be involved in this project.

### **4.3.3 Design Phase**

Following the analysis phase will be the design phase where the requirements outlined previously will be manifested into system specifications. We start by defining the architecture pattern that is well suited for our project system. Then the features and functionalities of the system will be used to construct use-case and activity diagrams.



### **4.3.4 Implementation Phase**

Once the design phase is completed, the design created will be realized into physical implementation according to the system specifications. The environment will be set up with related hardware and codes for algorithms. Different frameworks will each be tested out with datasets in order to find out their feasibility and test results for comparison. Once a suitable framework is chosen, the model will be integrated into the AWS cloud platform for training using data from edge devices. The results of the federated learning framework will be assessed through various performance metrics, a dashboard will also be developed for data visualization and displayed. Once all physical implementations are done, the system will become a prototype for testing and debugging.

### **4.3.5 System Prototype**

In this phase, the prototype will be tested using the collected data to be evaluated. Feedback will be given to determine whether the prototype satisfies the requirements or not. If the prototype doesn't meet its requirements, it will be refined and modified to meet the requirements.

### **4.3.6 Implementation (Final)**

A complete system which satisfies all requirements and is ready to be deliver and deployed in its specified environment.

#### 4.4 Formula for Sensor Calibration

pH calibration is very important in order to measure the accurate pH value, it involves comparing the output of the pH electrode with the known pH values of standard solutions. The pH meter will then use the calibration data to determine the relationship between pH electrode output and the pH values of the measured solution. The calibration coefficients, such as slope and intercept will be determined and then used to process the voltage output into pH units. This calibration process will be done on the PH4502C sensor for this project.

Below is the formula for measuring slope and intercept:

$$\text{Slope} = (\text{pH}_2 - \text{pH}_1) /$$

$$(\text{mV}_2 - \text{mV}_1) \text{Intercept} =$$

$$\text{pH}_1 - \text{Slope} \times \text{mV}$$

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 was done by involving various types of technological devices and components in order to develop a federated learning framework for precision aquaculture. The development of the framework will mainly be done on cloud services and integrated with edge devices.

#### 4.5.1 Laptop

Description	Specifications
Model	Lenovo Legion Y520-151KBN
Processor	Intel i7-7700HQ
Operating System	Windows 10
Graphic	NVIDIA GeForce GTX 1050
Memory	12GB DDR4 SDRAM
Storage	239 GB SSD, 932 GB HDD

**Table 4.1 Specifications of Laptop**

The laptop will be required for the configuration of AWS services for setting up the connection and monitoring of data between the main model of the system in AWS Cloud and edge participants. Computing power will also be required for writing and testing out various different federated learning frameworks using Jupyter Notebook to find out which one is the most suitable.

#### 4.5.2 Amazon Web Services (AWS)

Amazon Web Services (AWS) is the pioneering leader for cloud platforms industry that provides different kinds of cloud services which are on-demand for its users. Computing resources and storage infrastructure will be required in this project, services such as Amazon IoT Core, AWS Lambda, AWS IoT Greengrass, Amazon S3, Amazon Quicksight and Amazon SageMaker are the core services. Greengrass provides a Pub/Sub service that enables sending and receiving data from our edge environments to the cloud, AWS lambda will be ran on both edge environments and the cloud for different functions such as processing data and triggering events. Amazon DynamoDB will be used for storing and retrieving data and Amazon Quicksight is used to present the retrieved data in interactive dashboards. Amazon S3 will be

## CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

used for storing models from the edge environments and global model, Amazon Sagemaker will be used to train the global model and distribute to all edge environments.

### 4.5.3 Python programming language

Python will be the programming language used for the development of the proposed federated learning framework. This is because Python is a popular programming language which is supported for most programming tasks, Python is also easy to write and understand compared towards other programming languages. Other than that, python also has a wide range of libraries for federated learning, all edge devices and Lambda functions in the cloud will be written in Python, the training models will also be built using Jupyter Notebook.

### 4.5.4 Jupyter Notebook

Jupyter Notebook is an open-source web application that is used to share and create computational documents. It will be used for performing federated learning tasks, this includes testing out various different federated learning frameworks from different libraries. After testing is done, comparison between different frameworks will be done in order to select the most suitable framework to be used and further fine-tuned until satisfactory results are obtained. Jupyter Notebook will be ran on local device which does not consume any cloud computing resource to incur additional costing. Once the suitable framework is chosen, the model will be migrated into Amazon Sagemaker for training purposes and deployed.

### 4.5.5 AWS IoT Greengrass Core Software

AWS IoT Greengrass Core Software was installed in Raspberry Pi for it to extend the AWS functionality towards the edge environments. This was done in order for Raspberry Pi to act locally on data which was generated by the sensors using lambda functions and provides low latency on responding towards events. All of this is done to reduce the need for communication between AWS Cloud since the processing tasks will be done locally and only partial data is stored in the cloud, edge computing is achieved by installing AWS IoT Greengrass core software on edge devices to manage IoT devices locally.

### 4.5.6 MQTT Protocol

Message Queuing Telemetry Transport (MQTT) protocol is a lightweight messaging protocol that can be used for inter-machine communication. The lightweight design of MQTT protocol makes it suitable for IoT devices with limited processing power and bandwidth, it uses a Pub/Sub model where the client device publishes messages to a broker, which the messages are then received by devices that subscribe to it. This is coupled with event triggers which is suitable for real-time applications of the system. After data from sensors are collected and processed locally, they will be sent to AWS cloud through the MQTT protocol.

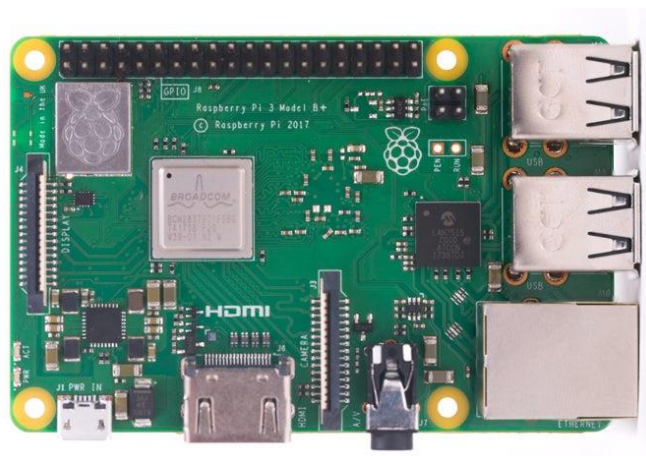
### 4.5.7 JSON

When custom software modules better known as “Component” are deployed into the edge devices, they can be separated into Artifact and Recipe. The artifact is the business logic of the component and includes scripts, static codes and resources. On the other hand, the recipe contains the metadata which describes the component such as its configurations, parameters, versions lifecycles and component dependencies. Recipes are preferred to be written in JSON language.

### 4.5.8 Raspberry Pi OS (Bookworm 64-bit)

Raspberry Pi OS is a Linux-based operating system that is specifically designed for Raspberry Pi. It was developed and maintained by the Raspberry Pi Foundation, this OS is chosen to be installed on the Raspberry Pi 4 Model B+ since the Lambda functions require a Linux-based OS to run on the edge devices. Lambda functions rely on cgroups feature of the Linux kernel due to AWS Greengrass Core v2 running as Docker container. The Docker uses cgroups for managing resource allocation and usage for containers. The 64-bit version is chosen since the TensorFlow library requires 64-bit OS to be installed and run as the edge device.

### 4.5.9 Raspberry Pi 3 Model B+



**Figure 4.4 Raspberry Pi 3 Model B+**

The development of the proposed system requires devices to connect communicate and manage other IoT devices and sensors at the edge environments. This central device will require sufficient compute power to run a portion of the tasks locally and also act as a gateway for communication with the cloud platform. With that being said, Raspberry Pi will be chosen as the core device that resides in the edge environments. Raspberry Pi 3 Model B+ with 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 specifications is chosen and is sufficient for this project. The 40-pin GPIO header will be extended to broadband for connecting the sensors with built-in wireless LAN for communication purposes.

### 4.5.10 Raspberry Pi T-Cobbler



**Figure 4.5 Raspberry Pi T-Cobbler**

Because The Raspberry Pi 3 Model B+ is short in GPIO pin header, the available pins will be occupied soon by different devices. This is why the use of Raspberry Pi T-Cobbler is crucial for extending the 40 GPIO pin header to a breadboard for allowing more wire to be connected to a single Raspberry Pi pin. The ribbon wire will be used as the connector between 40-pin of the Raspberry Pi and T-Cobbler with every GPIO pin header marked on it.

### 4.5.11 DS18B20 Waterproof Temperature Sensor



**Figure 4.6 DS18B20 Waterproof Temperature Sensor**

DS18B20 is a waterproof temperature sensor with a stainless-steel head that can be deployed underwater for sensing the temperature of its surrounding environment. It has 1-Wire interface sensor with a ground wire that needs to be connected to a microprocessor. Its measuring range is between  $-55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$  with  $\pm 0.5^{\circ}\text{C}$  reading accuracy in the range of  $-10^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ . An external library will be required to read the data for example DallasTemperature library, it requires 3.0 – 5.5V of input voltage to be powered up.



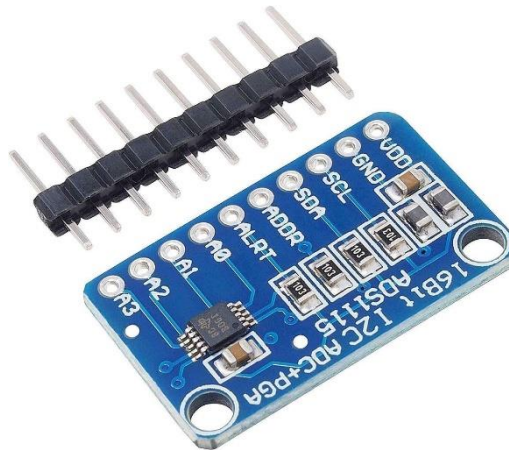
### 4.5.12 PH4502C Analog pH Sensor



**Figure 4.7 PH4502C Analog pH Sensor**

PH4502C Analog pH Sensor will be used to measure the pH value and temperature of the surrounding edge environment. It consists of a glass electrode probe that will be immersed in the liquid along with a reference electrode. The glass electrode will detect the changes in pH and generate voltage which is proportional to the acidity or basicity of the liquid. The reference electrode will provide stable voltage for ensuring accurate pH measurements, the range of pH is between 0PH – 14PH whilst the temperature is 0°C – 60°C with accuracy of  $\pm 0.1\text{pH}@25^\circ\text{C}$ . 5.00V of input voltage will be required to turn it on for data reading with less than 1 minute response time.

### 4.5.13 ADS1115 Analog Digital Converter



**Figure 4.8 ADS1115 Analog Digital Converter**

ADS1115 Analog Digital Converter is an analog to digital converter which will be used to convert analog signal into digital signal. Since Raspberry Pi GPIO pin is designed for receiving only digital signal, pH value for PH4502C sensor and ADS1115 will need to be converted. The four input channels on ADS1115 can be configured as single-ended or differential inputs with operating voltage ranging from 2.0V to 5.0V, the VDD will be connected to 5.0V pin on Raspberry Pi in our case. I2C interface will be used by ADS1115 for communicating with microcontroller and other digital devices.

### 4.5.14 Microsoft Power Bi

Microsoft Power Bi is a data visualization tools that can be used together with various kinds of data sources in order to present the data on an interactive dashboard, this dashboard in turn can provide business insights and intelligence towards the users. In this project, it will be used to present the data captured by the sensors from each edge environment, the dashboard will first be designed on Microsoft Power Bi before transitioning into AWS Quicksight.

### 4.5.15 Arduino Uno



**Figure 4.9: Arduino Uno board**

The Arduino Uno is an open-source microcontroller board equipped with sets of digital and analog input/output pins that can be used to interface other circuits. It is programmable through the Arduino IDE via a USB cable and barrel connector. In this project, the Arduino Uno board will be used to connect towards the pH4502c sensor to offset the voltage through its potentiometer in the pH sensor calibration process.

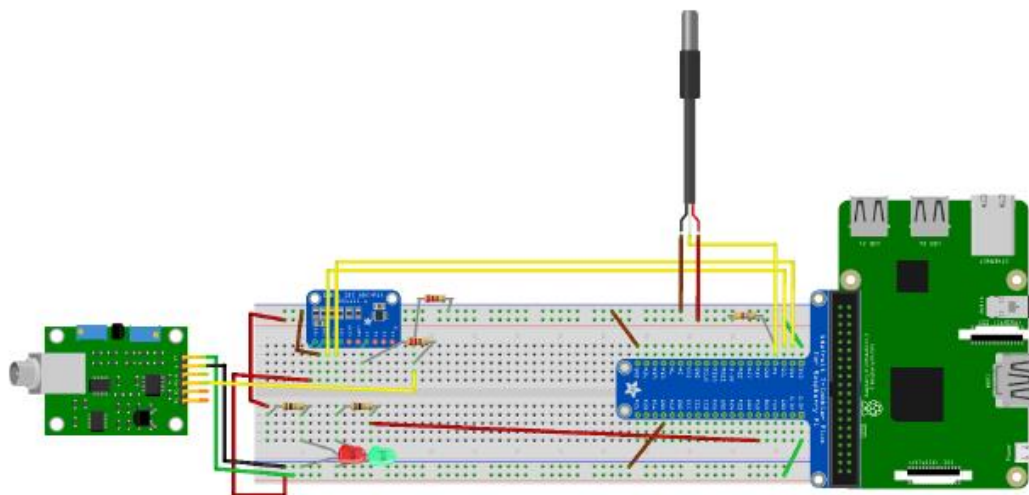
### 4.5.16 Flower Federated Learning Library

Flower federated learning library is an open-sourced library which is compatible with Python and TensorFlow which is designed for the development of federated learning systems. Federated learning is a paradigm for machine learning models to be decentralized through the training of global models and aggregation to edge devices or clients while maintaining data integrity by updating the training parameters only. In order to make use of Flower, its subclasses will be required for establishing client-side functionalities, as for the main server the default strategy is leveraged to orchestrate the federated learning algorithm. Flower also allows the creation of customized algorithms by defining the desired methods that are suitable for each user's needs.

### 4.5.17 TensorFlow

TensorFlow is an open-source machine learning framework which is widely used for training machine learning models. In this project, it will be utilized to develop a Recurrent Neural Network (RNN) model for the federated learning in edge and cloud environments. The purpose of the model is to predict the water parameter trends and offer foresight into the near-future water parameter trends to enhance the user's decision-making.

### 4.6 Hardware Prototype Design



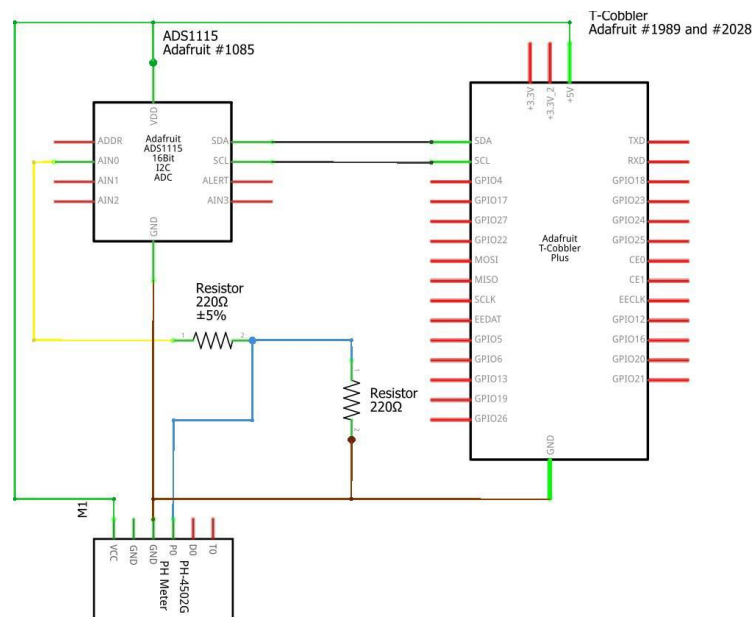
**Figure 4.10: Graphical Design of IoT Device**

The diagram shown above is an illustration of the prototype design of the IoT device that will be deployed onto the edge environments in a graphical format. The central connecting component that joins the Raspberry Pi machine and IoT sensors is the breadboard. One end of the breadboard is connected to a T-cobbler that represents the GPIO pins on the Raspberry Pi device. As for the IoT sensors, the pH and temperature sensors are connected to the breadboard using jumper wires, an analog-to-digital converter is also soldered and placed onto the breadboard together with LED lights. The main processing unit of the whole IoT device will be the Raspberry Pi with Greengrass Core Software installed, this enables the sensors to function locally without the need for triggers from AWS cloud. The component functions that will be performed locally include data collection through sensor readings, processing the captured data, establishing communication channels with AWS cloud and lastly storing and

sending the processed data. Additionally, federated learning client components will also be deployed on the IoT device to allow the federated learning process to be carried out.

### 4.7 Schematic Diagram

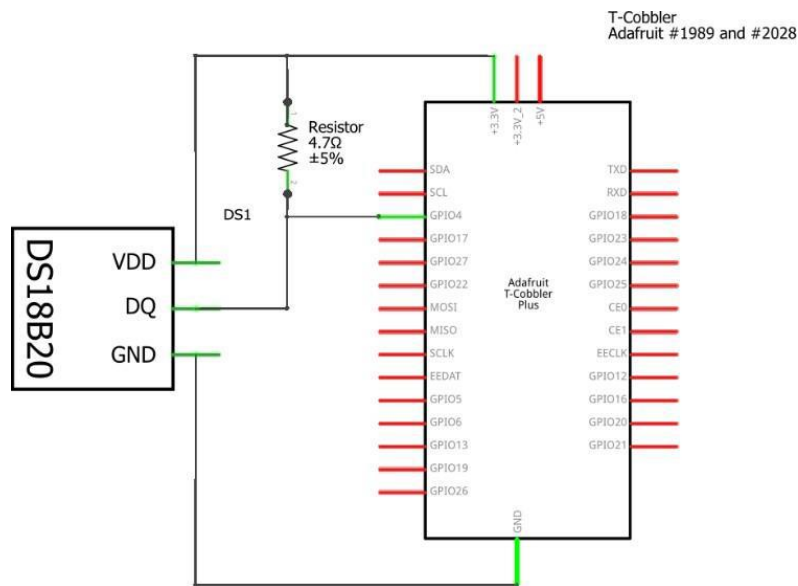
#### 4.7.1 pH Reading Module



**Figure 4.11: Schematic Diagram of pH Reading Module**

The diagram above illustrates the PH4502C pH reading module along with the ADS1115 ADC analog-to-digital converter. Sensor readings will be generated by the pH sensor when the data is being captured in the form of analog voltage values based on the pH level of the water. However, the GPIO pins on the Raspberry Pi are digital pins which means only binary (0 or 1) values can be read. Therefore, an analog-to-digital converter will be required to convert the analog voltage value into digital values for the Raspberry Pi device to comprehend and process. Both pH sensor and the ADC require a 5V pin connection. The AIN0 pin on the ADS1115 ADC will be connected to the P0 pin on the PH4502C sensor which outputs the analog signal for conversion purposes. Furthermore, the I2C (Inter-Integrated Circuit) interface of the Raspberry Pi needs to be enabled so that the SDA and SCL pins on the ADC will be able to communicate with the device.

### 4.7.2 Temperature Reading Module



**Figure 4.12: Schematic Diagram of Temperature Reading Module**

The diagram above depicts the DS18B20 waterproof sensor that is connected onto the T-cobbler. The sensor will require 3.3V to function, the data pins on the DS18B20 sensor will be connected to the GPIO 4 on the Raspberry Pi. The sensor will communicate with the Raspberry Pi device through the 1-Wire Protocol that needs to be enabled in the Raspberry Pi settings. This is a suitable setting because the default configuration of the GPIO 4 supports the 1-Wire protocol which makes it easier to communicate with the DS18B20 sensor without the need for much manual configurations.

4.7.3 T-Cobbler and LEDs

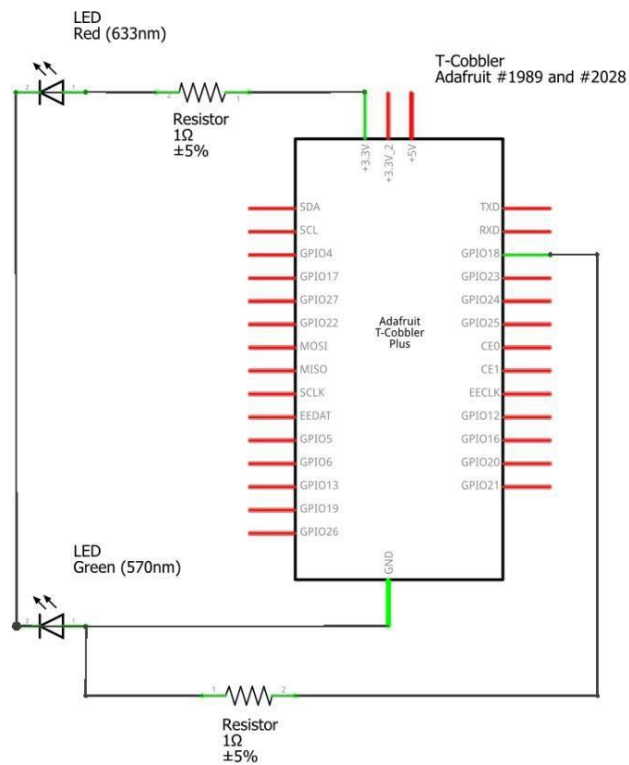


Figure 4.13: Schematic Diagram of T-Cobbler and LEDs

The figure above shows the T-Cobbler connected onto the 2 LEDs to perform indications of the IoT sensors during operations. The red LED is connected to the 3.3V pin and will light up to indicate when the device is connected to power. The green LED is connected to GPIO 18 pin and will illuminate when the sensors are activated to begin data capturing functions.

4.8 Project Timeline

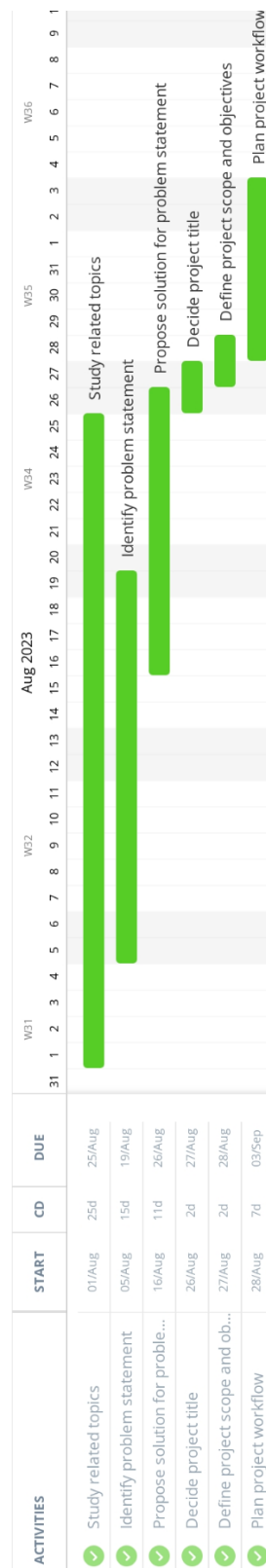


Figure 4.14: Gantt Chart (1)



# CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

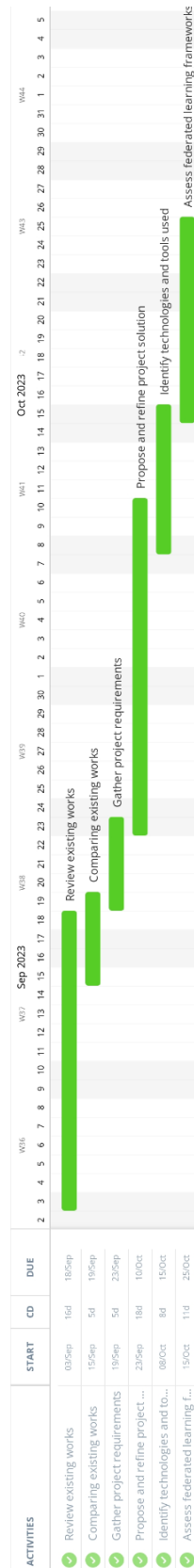


Figure 4.15: Gantt Chart (2)

# CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

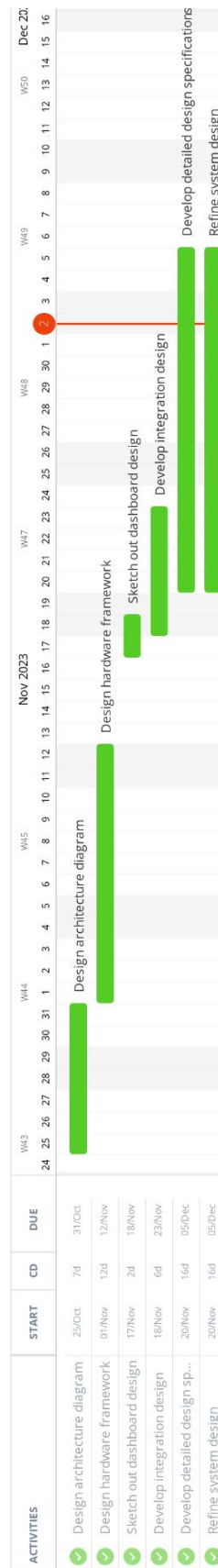


Figure 4.16: Gantt Chart (3)

# CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

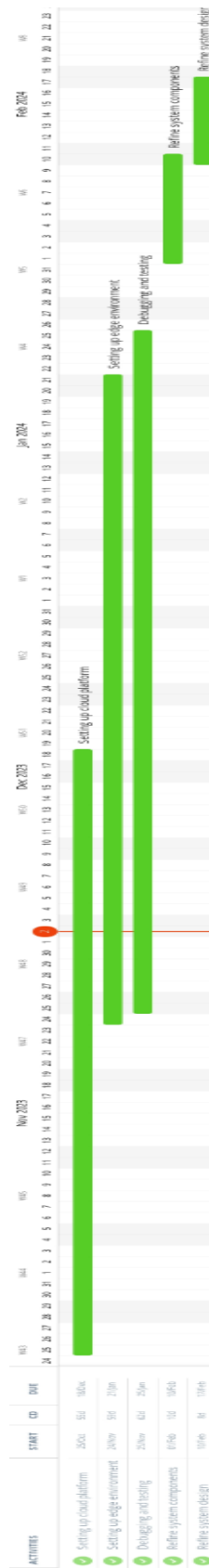


Figure 4.17: Gantt Chart (4)

# CHAPTER 4 METHODOLOGY AND SYSTEM DESIGN

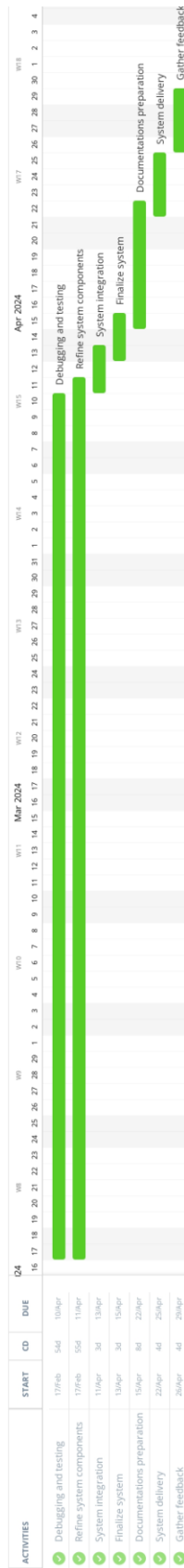


Figure 4.18: Gantt Chart (5)

## CHAPTER 5: Preliminary Work

### 5.1 Initial Set-up

Before developing a federated learning framework with IoT-Cloud Solution for Precision Aquaculture, the necessary software and installations have to be done beforehand and tools also have to be acquired prior from the actual development process:

1. Raspberry Pi OS (Release: 2023-02-21) was installed on Raspberry Pi and successfully booted.
2. Putty was installed to ssh to Raspberry Pi for enabling VNC Server.
3. RealVNC Viewer to control Raspberry Pi using laptop.
4. Java was installed on Raspberry Pi since it was required to install AWS IoT Greengrass core software.
5. An AWS root user account was registered.
6. A user account with “AdministratorAccess” was created using IAM following AWS best practices to access all AWS services.
7. Jupyter notebook was installed.
8. Microsoft Power Bi was installed.
9. All hardware and electronic components were procured and received.

### 5.2 Raspberry Pi Setup (Edge Device)

Prior to the beginning of the project, the below setup and installations must be done on the Raspberry Pi device to ensure the edge environment can function as it is intended.

1. Raspberry Pi OS Debian Bookworm 64-bit (Release 2024-3-15) has been installed on Raspberry Pi with SSH enabled and successfully booted.
2. Java default jdk has been installed on Raspberry Pi for installing AWS Greengrass Core software.
3. Raspberry Pi Cgroup reversion from v2 to v1 to run AWS Lambda functions locally.
4. Install the necessary dependencies and libraries for running the sensors.
5. Enabling the 1-Wire and I2C interfaces on Raspberry Pi configuration panel for the sensor communication.
6. TensorFlow is installed for primary machine learning library.
7. SQLite2 installed to run local database.

#### 5.2.1 Raspberry Pi Setup Command

Below shows the commands which are required to be run before the start of development.

1. Verify that java is installed and check its version
  - 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@rasberryp
```

Figure 5.2: CLI displaying content boot configuration file

- sudo sed -I -e "1 s/\$/ cgroup\_enable=memory cgroup\_memory=1 system.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 fs
ck.repair=yes rootwait quiet splash plymouth.ignore-serial- consoles cgroup_enabl
e=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 and v2

3. Install library for DS18B20 temperature sensor.
  - sudo pip3 install Adafruit\_DHT
  - sudo pip3 install w1 therm sensor
  - sudo raspi-config
    - Navigate to “Interface Options” and select “1-Wire”
    - Choose “Yes” to enable 1-Wire interface.
4. Install library for PH4502C pH sensor
  - sudo pip3 install Adafruit-circuitpython-ads1x15
  - sudo raspi-config
    - Navigate to “Interface Options” and select “I2C”
    - Choose “Yes” to enable I2C interface.
5. Reboot Raspberry Pi for the changes to take effect for 1-Wire and I2C
6. Install TensorFlow
  - Install the TensorFlow version compatible with Debian Bookworm 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'
```

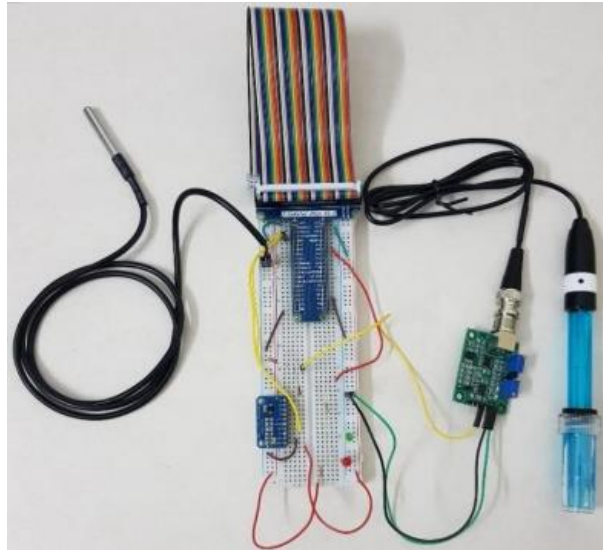
Figure 5.5: CLI displaying TensorFlow installed

## CHAPTER 5 SYSTEM IMPLEMENTATION

### 7. Install Sqlite3

- sudo apt update
- sudo apt full-upgrade
- sudo apt install sqlite3

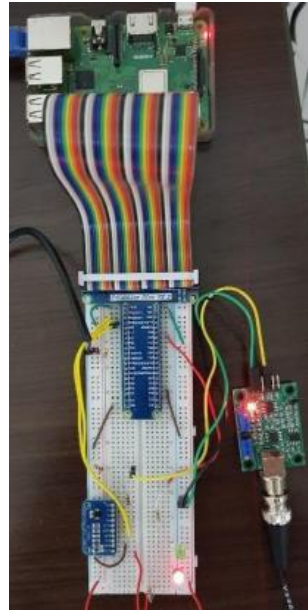
### 5.3 Hardware Prototype Set-up



**Figure 5.6: Hardware Implementation Based on Breadboard**

Figure 5.6 represent the hardware implementations which have been set up based on a breadboard for each edge environment. 2 sets of hardware implementations were set up to simulate the scenario of having multiple aquaculture farms at the same time. The breadboard is the foundation for connecting all the electronic components together as well as the IoT devices. 2 types of sensor devices were used in this implementation to collect real-time data for water parameters, they were the DS18B20 waterproof temperature sensor and PH4502C pH sensor. Since the output collected by the PH4502C sensor is in analog signals, it must be converted to digital signals to be interpreted by the Raspberry Pi using an ADS1115 analog converter. Furthermore, 2 LEDs were installed to indicate the activity of the edge environment, the red LED indicated that the breadboard received power and the green LED represented the sensors are active. Lastly, a T-Cobbler is used as an extension from the 4-pin GPIO located on the Raspberry Pi.





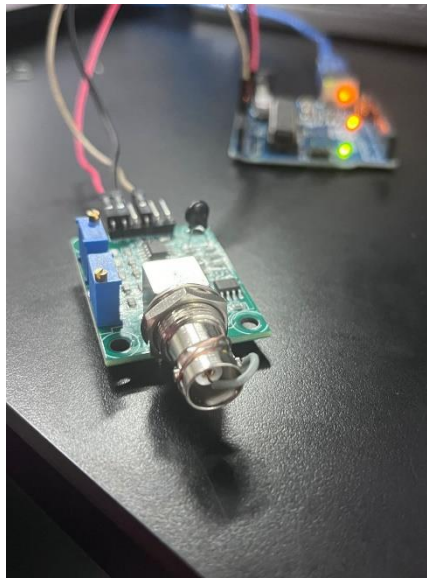
**Figure 5.7: IoT Devices receiving power**

Both sets of hardware implementations from figure 5.6 were connected onto a Raspberry Pi which is used to serve as the Greengrass core device at edge environments. AWS Greengrass core device acts as a central control unit for the local IoT device, data processing, messaging and caching will be done through the greengrass core device whether on or offline. Each edge environment will communicate with AWS Cloud through MQTT protocols. From figure 5.7 we can see the complete edge environment where the hardware from the breadboard is connected to the Raspberry Pi (Greengrass Core device), the red LED is turned on indicating the environment is active and power is connected.

### 5.4 Sensor Calibration

The calibration process preceding will be done on the PH4502C sensor and is crucial to ensure the sensor is able to accurately capture reliable pH measurements. The whole process is separated into various steps from voltage offset all the way to evaluating pH readings and calculating the slope.

### 5.4.1 Adjusting Sensor Voltage Offset



**Figure 5.8: BNC Connector with wire coiled onto outer metal casing**

In this step, the pH sensor will be forced to output a constant voltage of 7.0 which corresponds to a neutral pH reading. To achieve this effect, the BNC connector which connects towards the sensor probe will be shorted using a connecting metal wire from the inner connection towards the outer metal casing of the BNC connector which can be seen from figure 5.8.



**Figure 5.9: Highlighted area on reference electrode is Potentiometer**

Once the BNC connector has been shorted, the pH sensor will be dialed to 2.5 volts by adjusting the potentiometer which is located on the electrode module. From figure 5.9 we can see that the potentiometer is the one which is closer towards the BNC connector. The Arduino code is used in figure 5.10 to read the output voltage in the Arduino IDE, we can stop adjusting the potentiometer once the desired voltage is achieved.

## CHAPTER 5 SYSTEM IMPLEMENTATION



```
CODING_PH_SENSOR_CALIBRATION | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Arduino Uno
CODING_PH_SENSOR_CALIBRATION.ino
1 #define OFFSET
2
3 int pH_Value;
4 float Voltage;
5
6 void setup()
7 {
8   Serial.begin(9600);
9   pinMode(pH_Value, INPUT);
10 }
11
12 void loop()
13 {
14   pH_Value = analogRead (A0);
15   Voltage = pH_Value * (5.0/1023.0);
16   Serial.println(Voltage);
17   delay(500);
18 }
19

Output Serial Monitor x
Message (Enter to send message to 'Arduino Uno' on 'COM3')
2.50
2.51
2.51
2.50
2.51
2.51
2.51
2.51
2.50
2.51
2.50
2.50
```

Figure 5.10: Arduino IDE for reading sensor voltage

### 5.4.2 Mapping the pH Readings to Output Voltage



Figure 5.11: Buffer Solution for pH 4.01 and pH 7.00



**Figure 5.12: pH 4.01 and pH 7.00 Buffer Solution with pH Probe immersed**

For this next step, the buffer solution for different pH as in figure 5.11 will be used for the PH4502C sensor measurements. Two buffer solutions which are pH 4.01 and pH 7.00 were obtained and mixed with water. The BNC connector is now connected onto the sensor probe and the probe is immersed into the buffer solutions accordingly for some time to capture a stable output voltage reading, this step can be seen from figure 5.10 where the pH probe is immersed in the pH 4.01 buffer solution.

<pre> 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 </pre>	<pre> 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 </pre>
<pre> \$ cat 3.06 V 3.06 V 3.06 V 3.06 V 3.06 V 3.06 V 3.06 V 3.06 V </pre>	<pre> \$ cat 2.55 V 2.55 V 2.55 V 2.55 V 2.55 V 2.55 V 2.55 V 2.55 V </pre>

**Figure 5.13: Output Voltage of pH sensor from pH 4.01 and 7.00 Buffer Solution**

In figure 5.13 we can see the output voltage reading for both buffer solutions after immersing the pH probe in them for some time. The voltage reading for pH 4.01 on the left indicated 3.06 volts whilst pH 7.00 indicated 2.55 volts.

### 5.4.3 Calculate Slope and Intercept

After the voltage readings have been recorded, we can calculate the slope and intercept values as the coefficients used for pH calculation. When the pH prob detects the hydrogen ions in a solution, it generates a small electrical potential in response to the level of concentration. The potential released is measure as voltage output which will then be 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 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
Shell
4.08
4.08
4.08
4.08
4.08
4.08
4.08
4.08
4.08
4.08

```

```

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
Shell
6.92
6.92
6.92
6.92
6.92
6.92
6.92
6.92
6.92
6.92

```

Figure 5.14: pH reading of pH sensor for pH4.01 and 7.00 Buffer Solution

Finally, a python code is written with the formula below to read pH values from the sensors. Both pH 4.01 and 7.000 buffer solutions were reused to measure the result for the pH sensor readings. The readings for pH buffer 4.01 were 4.08 and pH 7.00 were 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 a feature which is located inside AWS IoT Core that is used to better manage each edge environment as a Greengrass Core device. For this project, the Raspberry Pi will be registered as the Greengrass Core device at each edge environment by installing the software on it. Currently, the Greengrass Core device is used to control both sensors on the device for data collection, send alert messages and sync data locally into the cloud. In the future, the local processing will be extended onto training machine learning models locally. The water parameters which were obtained from the sensors will be sent into AWS Cloud by using MQTT protocol.

## CHAPTER 5 SYSTEM IMPLEMENTATION

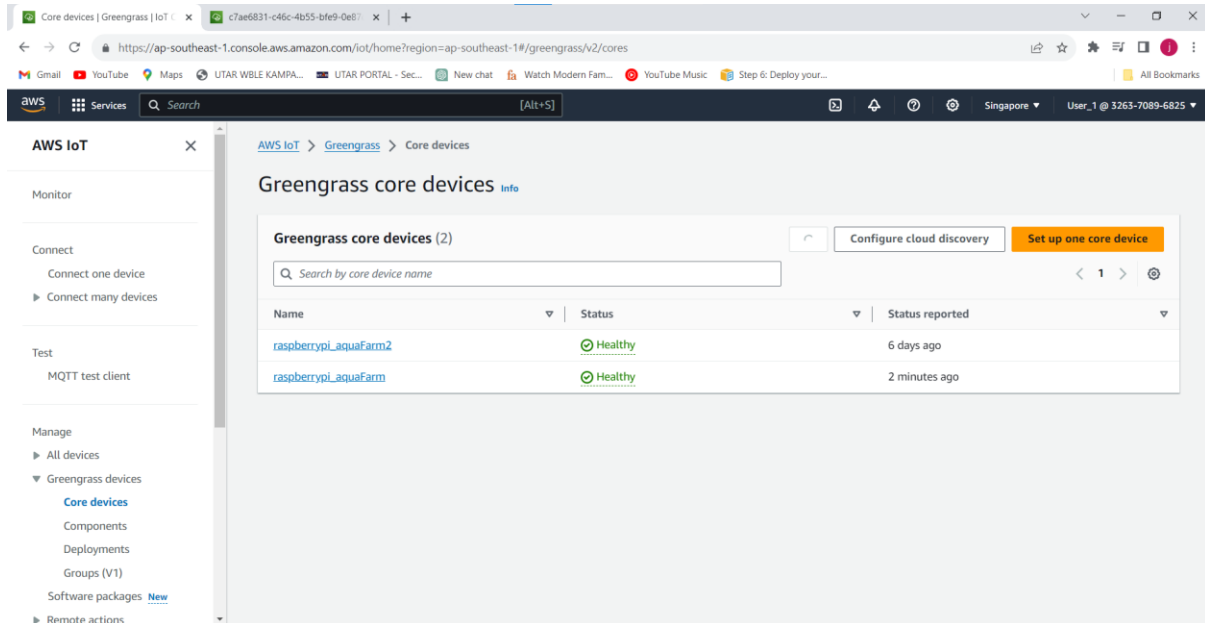


Figure 5.15: Raspberry Pi Registered as Greengrass Core Device

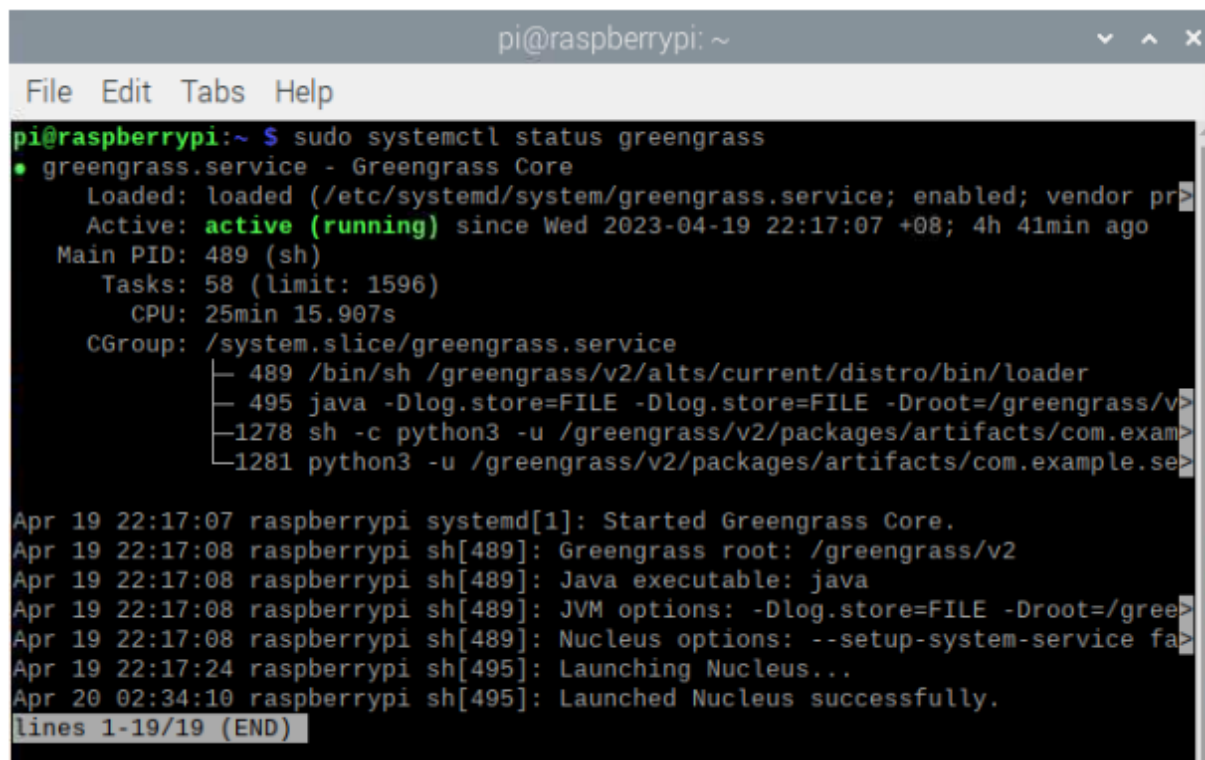
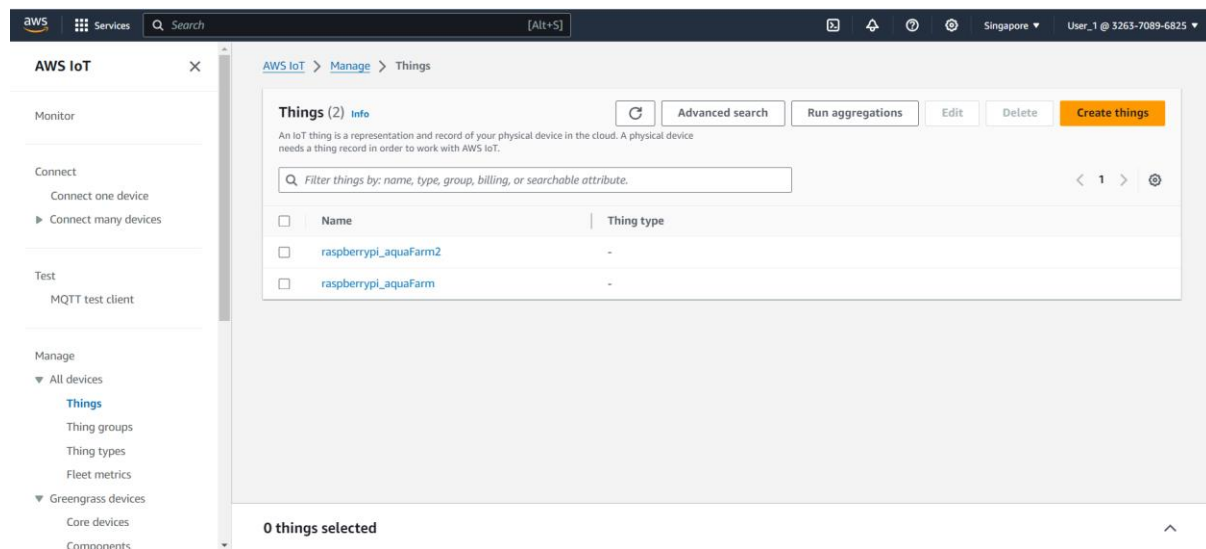


Figure 5.16: Raspberry Pi CLI indicating Greengrass Core Software in Installed

Figure 5.15 shows that both the Raspberry Pi have been registered in AWS IoT Core as Greengrass core devices with names “raspberrypi\_aquaFarm” and “raspberrypi\_aquaFarm2”. We can just follow the instructions which have already been provided in AWS to set up the Greengrass core devices by simply clicking “Set up one core device” on the core devices page.

## CHAPTER 5 SYSTEM IMPLEMENTATION

When the device has successfully been registered, it will be shown on the list of active devices in the AWS management console. On the Raspberry Pi CLI we can also use the cli commands to check that the software has been installed and Greengrass core is running and active such as in Figure 5.16.



**Figure 5.17: Greengrass Core Device inside IoT Thing Group**

From figure 5.17 we can see that both Raspberry Pi devices have also been registered in AWS IoT Things and each included in their own things group. Since Greengrass Core devices were still considered as IoT devices, registering them in Greengrass core will also automatically create them as Things, the user will be required to specify the Thing group for the Greengrass Core device when setting it up previously. In each Thing Group, the Things inside will be managed as a single entity and any policies or configurations which are applied will be broadcasted across all devices inside the group.

### 5.5.2 AWS Greengrass Component

AWS IoT Greengrass consists of components which are basically software modules which can be deployed and run on each individual Greengrass Core devices. Components can be categorized into public and custom components, public components are the ones which have been provided by AWS and are common to be used on core devices. On the other hand, AWS also allows users to create their own custom components which will have other specific requirements. Both of these components can be deployed onto specific core devices to have effect and take action for implementations in this project.



## CHAPTER 5 SYSTEM IMPLEMENTATION

Name	Dependency type	Version	Status	Status changed	Status reported	Latest deployment source
<a href="#">aws.greengrass.Nucleus</a>	Dependency	2.12.0	Finished	2 days ago	1 hour ago	Not reported
<a href="#">FleetStatusService</a>	Dependency	0.0.0	Running	2 days ago	1 hour ago	Not reported
<a href="#">UpdateSystemPolicyService</a>	Dependency	0.0.0	Running	2 days ago	1 hour ago	Not reported
<a href="#">aws.greengrass.TokenExchangeService</a>	Dependency	2.0.3	Running	2 days ago	59 minutes ago	<a href="#">4b91862b-2bed-44b3-b3d2-e0bb44a17a10</a>
<a href="#">TelemetryAgent</a>	Dependency	0.0.0	Running	2 days ago	1 hour ago	Not reported
<a href="#">DeploymentService</a>	Dependency	0.0.0	Running	2 days ago	1 hour ago	Not reported
<a href="#">aws.greengrass.LambdaManager</a>	Dependency	2.3.2	Running	2 days ago	1 hour ago	Not reported
<a href="#">aws.greengrass.Cli</a>	Root	2.12.0	Running	59 minutes ago	59 minutes ago	<a href="#">4b91862b-2bed-44b3-b3d2-e0bb44a17a10</a>
<a href="#">com.example.testAlarmv2</a>	Root	1.0.0	Running	59 minutes ago	59 minutes ago	<a href="#">4b91862b-2bed-44b3-b3d2-e0bb44a17a10</a>
<a href="#">com.example.sync</a>	Root	1.0.0	Running	59 minutes ago	59 minutes ago	<a href="#">4b91862b-2bed-44b3-b3d2-e0bb44a17a10</a>

**Figure 5.18: Component List for raspberry\_pi\_aquaFarm**

Name	Version
<a href="#">aws.greengrass.Cli</a>	2.12.0
<a href="#">aws.greengrass.LambdaLauncher</a>	2.0.12
<a href="#">aws.greengrass.LambdaRuntimes</a>	2.0.8
<a href="#">aws.greengrass.LocalDebugConsole</a>	2.4.1
<a href="#">aws.greengrass.SNS</a>	2.1.7
<a href="#">aws.greengrass.TokenExchangeService</a>	2.0.3
<a href="#">com.example.sensorv5</a>	1.0.0
<a href="#">com.example.sync</a>	1.0.0
<a href="#">com.example.testAlarmv2</a>	1.0.0

**Figure 5.19: Deployment for raspberry\_pi\_aquaFarm**

When we successfully install the Greengrass Core software on our core devices, some public components such as “aws.greengrass.Cli” and “aws.greengrass.TokenExchangeService” will be automatically deployed together to the Greengrass core device. When we want to deploy new components onto our core devices, we can opt to use the AWS management console or the

## CHAPTER 5 SYSTEM IMPLEMENTATION

Greengrass Core CLI. A deployment is the process for adding a list of new components to multiple or single devices at once depending on the number of Things in the group. Figure 5.18 shows the list of components which have been deployed onto raspberrypi\_aquaFarm which consists of both public and custom components.

### 5.5.2.1 Sensor Reading and processing component (com.example.sensorv5)

In order to capture the water parameter readings of the aquaculture farm using the local sensors we have deployed, a custom component has been developed in Greengrass which is named “com.example.sensorv5”. Figure 5.20 gives a brief overview of the artifact which is related to this component and figure 5.21 displays the recipe associated with it.



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

Figure 5.20: Snapshot of artifact for com.example.sensorv5



```
1 [{"RecipeFormatVersion": "2020-01-29",
2   "ComponentName": "com.example.sensorv5",
3   "ComponentVersion": "1.0.0",
4   "ComponentDescription": "sensorv5 MQTT Pub/Sub",
5   "ComponentPublisher": "Me",
6   "ComponentConfiguration": {
7     "DefaultConfiguration": {
8       "message": "switch",
9       "accessControl": {
10        "aws.greengrass.ipc.mqttproxy": {
11          "com.example.sensorv5:mqttproxy:1": {
12            "policyDescription": "Allows access to pub/sub to aquaFarm1/sensor.",
13            "operations": [
14              "aws.greengrass:PublishToIoTCore",
15              "aws.greengrass:SubscribeToIoTCore"
16            ]
17          },
18          "resources": [
19            "aquaFarm1/sensor",
20            "aquaFarm1/switch"
21          ]
22        }
23      }
24    }
25  }
26 ]
```

Figure 5.21: Snapshot of recipe for com.example.sensorv5

The functions which make up the “com.example.sensorv5” Greengrass component are mainly used to capture raw data from the sensors which have been deployed at 10-second intervals. In figure 5.20, we can see the function which is used to process raw data which is read from the

## CHAPTER 5 SYSTEM IMPLEMENTATION

DS18B20 temperature sensor, this involves extracting the temperature element from the raw data and converting it from millidegrees Celsius into degrees Celsius. As for the pH readings from the PH4502C sensor, the function shown in figure 5.21 is used to calculate the pH value by using the slope and intercept which have been obtained previously.

```
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 5.22: Local Database function**

```
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 5.23: Temporary Database function**

Furthermore, once the processed temperature and pH values have been recorded, the data will be paired together with additional data which is the timestamp and unique ID to represent the Greengrass core device. Once this is done, a function will be used to store the processed data into a local database in case network connectivity is unavailable. This is shown in figure 5.22 and 5.23 for a local temporary database. When the network connectivity is established once again, the processed data will then be stored inside the local main database and transformed into JSON format which is used for transmission. The transmission of data from edge devices onto the cloud is through a Pub/Sub service where messages are packaged and published onto their respective topics. In this project, the “aquaFarm1/sensor” topic is used to receive the messages and the “aquaFarm1/switch” topic is used to publish a message in order to turn on

## CHAPTER 5 SYSTEM IMPLEMENTATION

the sensors at the edge environments. The communication between the AWS IoT Core and component inside the edge device is facilitated through MQTT protocol and involves the subscription towards various topics for publishing. Lastly, JSON messages are also published onto the “loc/sensor” topic which is used for storing messages irrespective of network connection. These messages will be utilized by subsequent components.

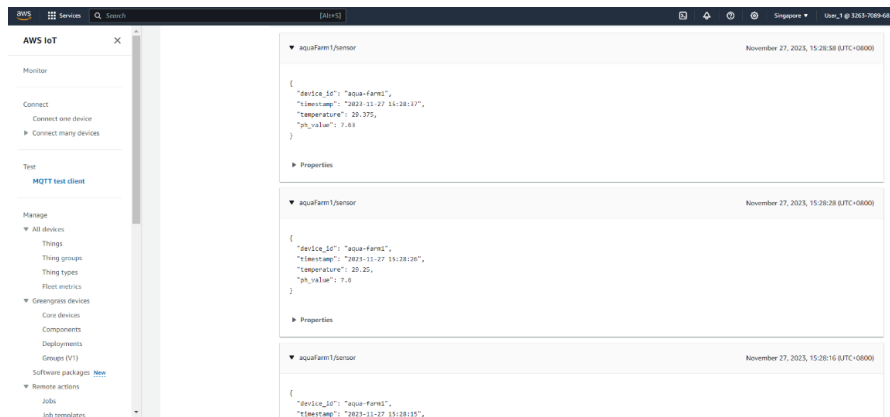


Figure 5.24: AWS IoT Core MQTT Test Client

In figure 5.24 we can see the usage of the AWS IoT Core MQTT Test Client to test out the communication between AWS IoT Core and the edge environment. The topic “aquaFarm1/sensor” has been subscribed to receive incoming messages and a JSON message is then published onto the “aquaFarm1/switch” topic to enable the component in edge environments to start capturing sensor data. The water parameters will then immediately be shown onto the console and the green LED on the hardware of the edge devices will also be turned on to indicate the sensors are active.

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

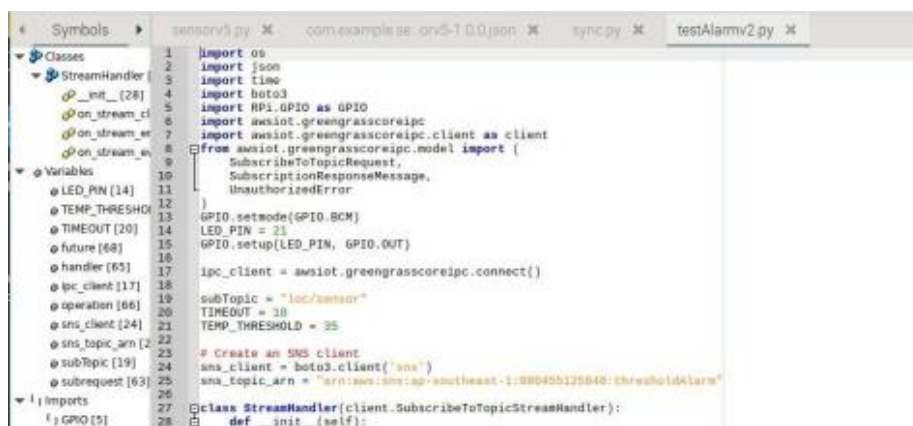


Figure 5.25: Snapshot of artifact for com.example.testAlarmv2  
Bachelor of Computer Science (Honours)  
Faculty of Information and Communication Technology (Kampar Campus), UTAR

## CHAPTER 5 SYSTEM IMPLEMENTATION



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

**Figure 5.26: Snapshot of recipe for com.example.testAlarmv2**

Both figures above show the artifact and recipe for the custom component “com.example.testAlarmv2”. This component is used to operate concurrently with the “com.example.sensorv5” component, it will evaluate the water parameters of the processed water by subscribing to the local topic “loc/sensor” which has been mentioned before. The messages which are received from the subscribed topic are parsed from JSON format to extract the temperature and pH values. These values are then compared against predefined thresholds to determine if they have exceeded the acceptable levels. When one of the water parameters have breached the threshold values, an alarm will be triggered and sent towards the user via the AWS SNS service. This component makes use of the boto3 library that enables direct API requests onto AWS services to allow the sending of the alarms towards users through SNS topics which have been created beforehand.

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 5.27: Snapshot of artifact for com.example.sync

```

1 {
2   "RecipeFormatVersion": "2020-01-25",
3   "ComponentName": "com.example.sync",
4   "ComponentVersion": "1.0.0",
5   "ComponentDescription": "sync MQTT Pub",
6   "ComponentPublisher": "Me",
7   "ComponentConfiguration": {
8     "DefaultConfiguration": {
9       "message": "sync",
10      "accessControl": {
11        "aws.greengrass.ipc.mqttproxy": {
12          "com.example.sensorv5.mqttproxy:1": {
13            "policyDescription": "Allows access to pub/sub to aquaFarm1/sync.",
14            "operations": [
15              "aws.greengrass#PublishToIoTCore",
16              "aws.greengrass#SubscribeToIoTCore"
17            ],
18            "resources": [
19              "aquaFarm1/sync"
20            ]
21          }
22        }
23      }
24    },
25    "Manifests": [
26      {
27        "Platform": {
    
```

Figure 5.28: Snapshot of recipe for com.example.sync

Figure 5.27 and 5.28 show the artifact and snapshot for the custom component “com.example.sync” which is a component used to perform data synchronization between the local main database residing in the edge devices onto the cloud database when a network connection has been obtained. This component is activated in 5-minute time intervals and makes use of the MQTT protocol along with boto2 library. When network connectivity has been restored, the component first checks if the temporary database is empty, if there is data the component begins comparing each row of data between the temporary database with data stored inside DynamDB. The absence of a row of data due to lack of connection can be

## CHAPTER 5 SYSTEM IMPLEMENTATION

identified since each row is unique. The identified row will then be converted to JSON format and published to the “aquaFarm1/sync” topic which is then sent to AWS IoT Core, every row in the temporary database is the deleted regardless if it is exiting in DynamoDB.

### 5.5.2.4 Federated learning Client Component (com.example.flwrClient2)



```
try:
    s3_client.download_file(bucket, key, temp_file_path)
    print(f"File download from S3 bucket: {bucket}, key: {temp_file_path}")
except Exception as e:
    print(f"Error download file from S3: {str(e)}")

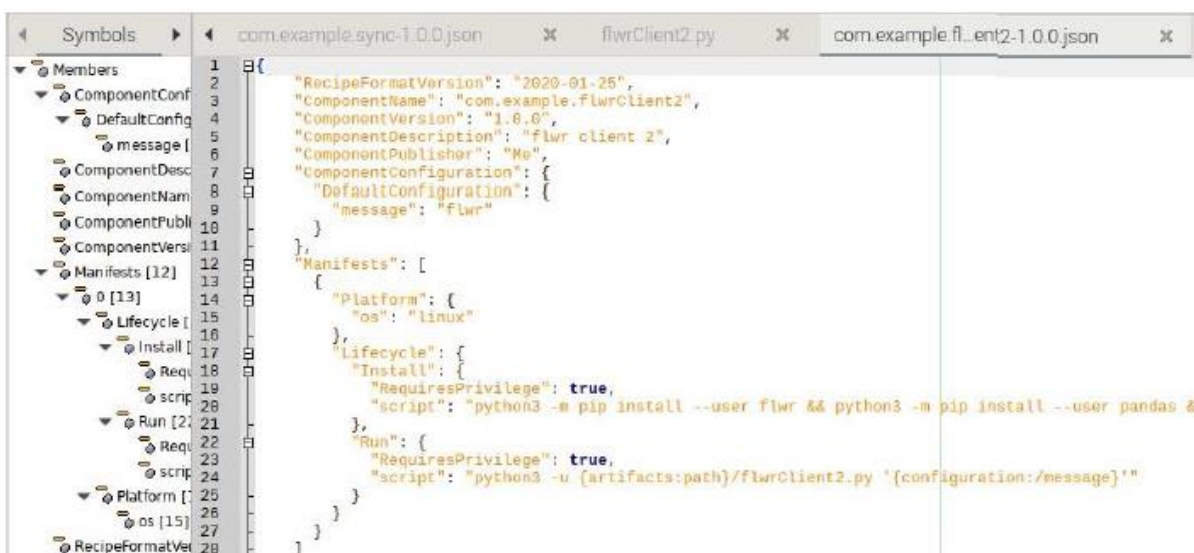
return temp_file_path

class numpyClient(fl.client.NumPyClient):
    def __init__(self, X_train, y_train):
        self.X_train = X_train
        self.y_train = y_train

    def get_parameters(self, config):
        print(f"(config)Numpy_Get_Parameter")
        bucket = 'flwr-testbucket'
        prefix = f"parameters/weights/{client_id}params.pkl"
        p = local_get_parameters()

        # Use a temporary file to store the model weights
        try:
            with tempfile.NamedTemporaryFile(dir='/tmp', delete=False) as fp:
                pickle.dump(p, fp)
                #logger.info(f"Dumped parameters to : {fp.name}")
                upload_to_s3(bucket, prefix, fp.name.replace('data', 'tmp'))
        except Exception as e:
            print(f"Error get_parameters: {str(e)}")
```

Figure 5.29: A Part of Artifact for Federated Learning Client Component  
com.example.flwrClient2



```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.flwrClient2",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "flwr client 2",
  "ComponentPublisher": "Me",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "message": "flwr"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "Install": {
          "RequiresPrivilege": true,
          "script": "python3 -m pip install --user flwr && python3 -m pip install --user pandas &&"
        },
        "Run": {
          "RequiresPrivilege": true,
          "script": "python3 -u {artifacts:path}/flwrClient2.py '{configuration:/message}'"
        }
      }
    }
  ]
}
```

Figure 5.30: A Part of Recipe for Federated Learning Client Component  
com.example.flwrClient2

## CHAPTER 5 SYSTEM IMPLEMENTATION

The federated learning process for this project will be conducted via a server which is located inside the cloud along with several client edge environments. The component represented from figure 5.29 and 5.30 above represent the artifact and recipe for the federated learning client at each edge environment. The main libraries that are used for this component will be the TensorFlow and Flwr libraries for federated learning and boto3 for communication purposes with cloud services.

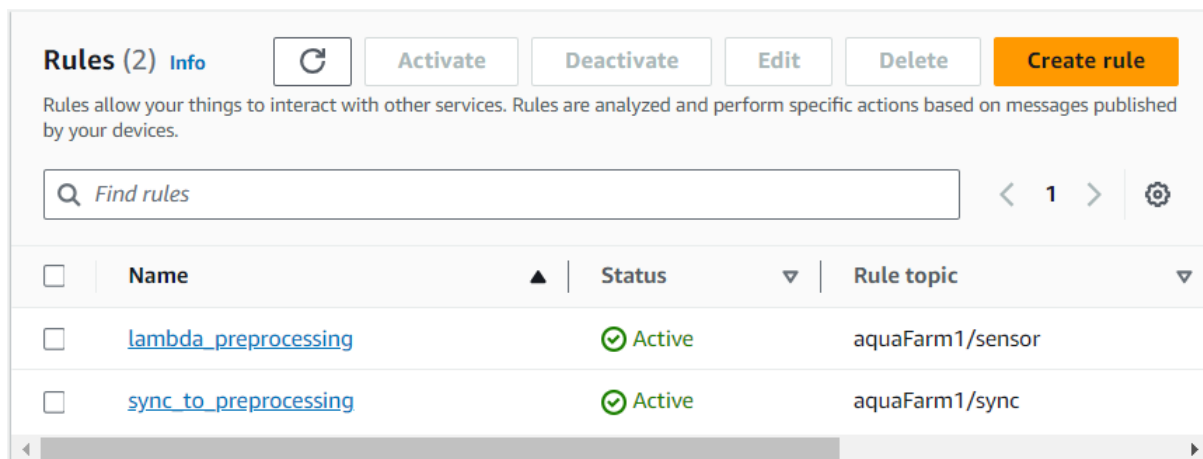
Each client will implement the subclasses available under Flwr's "flwr.client.NumPyClient" which is the easier option to be integrated. This specific subclass will define how the local client trains its model locally and performs evaluation once the federated learning server initiates the whole process. The local machine learning model training will be involving a Recurrent Neural Network (RNN) from the TensorFlow library by training the data which has been stored inside the local database of the raspberry pi machine. Once the local training is done and evaluated, the boto3 library will be used to communicate with the cloud services by retrieving and uploading the global models and aggregated model weights in Amazon S3.

The federated learning framework ensures that all clients across multiple edge environments are able to communicate with the server which is responsible for the orchestration of model aggregation through gRPC protocols. The clients will request access to the server through its DNS name and network ACL inbound rules which have been set up. More detailed explanation will be provided under the federated learning process section below.



## 5.6 Setting Up AWS Cloud

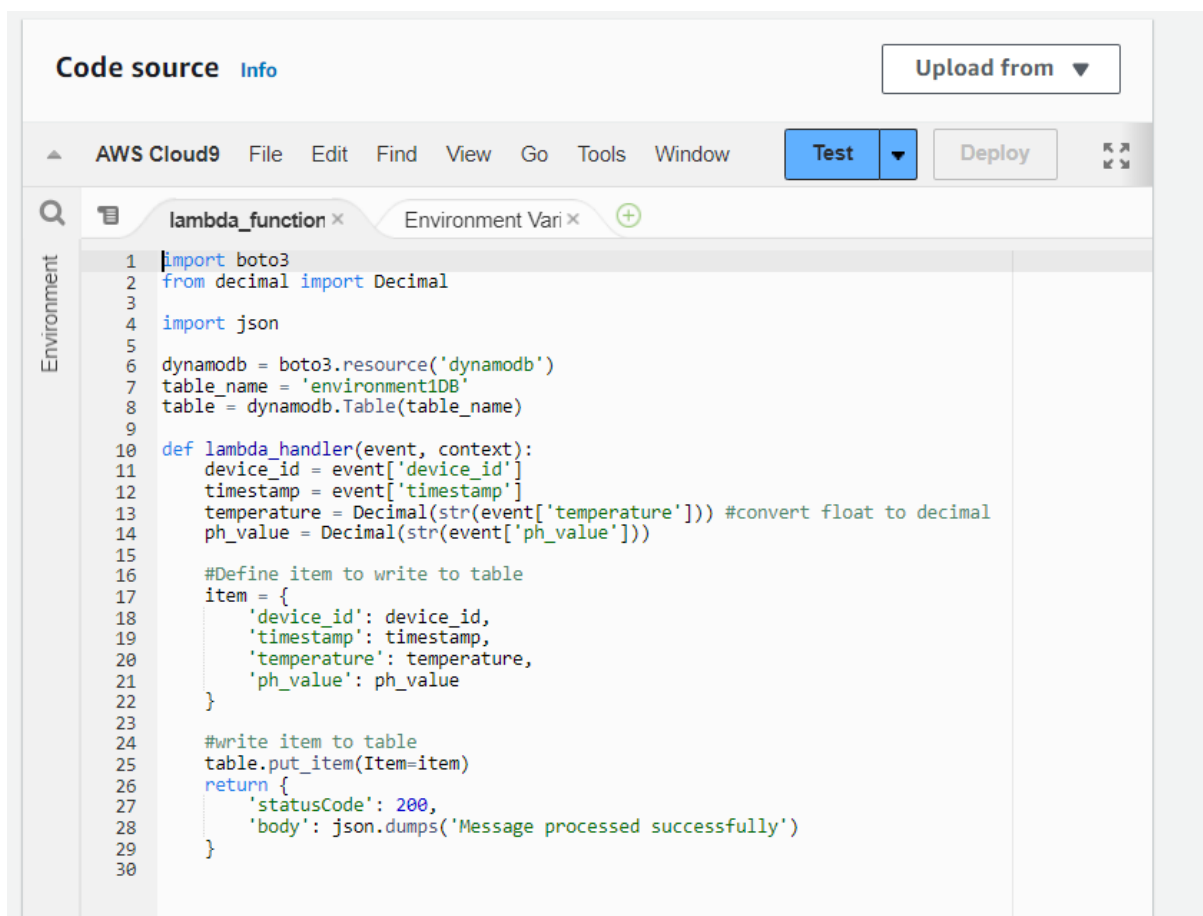
### 5.6.1 AWS IoT Core



**Figure 5.31 IoT Rules for Message Routing**

AWS IoT Core is the means for AWS Cloud to communicate with IoT devices, the use of AWS IoT Core in this project is to perform message routing for directing the sensor data which has been sent from Greengrass Core devices to other AWS services. IoT rules will be configured to direct these messages onto lambda functions. From figure 5.31 we can see that there are 2 IoT Rules which have been configured to subscribe to topics “aquaFarm1/sensor” and “aquaFarm1/sync” which will have messages published by the Greengrass Core devices and retrieve the data from these topics. The action of both rules is to direct the messages to Lambda for further processing.

## 5.6.2 Amazon Lambda



```

Code source Info
Upload from
AWS Cloud9 File Edit Find View Go Tools Window Test Deploy
lambda_function x Environment Vari x
1 import boto3
2 from decimal import Decimal
3
4 import json
5
6 dynamodb = boto3.resource('dynamodb')
7 table_name = 'environment1DB'
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     return {
27         'statusCode': 200,
28         'body': json.dumps('Message processed successfully')
29     }
30

```

**Figure 5.32: Lambda Function intermediateProcessing**

AWS Lambda is a simple serverless compute service that allows codes to be run in the language of your choice in response to an event that triggers it. The lambda function that can be seen in figure 5.32 is written in Python 3.9 runtime and deployed for processing incoming IoT rules triggers. This lambda function is named “intermediateProcessing” and will be receiving messages from IoT rules and storing them into a specific database in AWS DynamoDB. The input for this lambda function will be JSON message that contains the water parameters captured by the edge environment sensors, the message will then be split into individual items that represent the columns inside the database. This is done to ensure the data will be structured so it can be easily visualized later in a dashboard. Once the messages have been processed and the data is correctly formatted, the lambda function inserts the data into the specific DynamoDB database.

## 5.6.3 Amazon DynamoDB

Scan     Query

Select a table or index: Table - environment1DB
     
 Select attribute projection: All attributes

Filters: ▶

ⓘ This table has more items to retrieve. To retrieve the next page of items, choose **Retrieve next page**.

**Items returned (50)**

<input type="checkbox"/>	device_id (String)	timestamp (String)	ph_value	temperature
<input type="checkbox"/>	<a href="#">aqua-farm1</a>	2023-11-27 15:40:14	7.5	29.375
<input type="checkbox"/>	<a href="#">aqua-farm1</a>	2023-11-27 15:40:25	7.51	29.375
<input type="checkbox"/>	<a href="#">aqua-farm1</a>	2023-11-27 15:40:37	7.51	29.312
<input type="checkbox"/>	<a href="#">aqua-farm1</a>	2023-11-27 15:40:48	7.5	29.312
<input type="checkbox"/>	<a href="#">aqua-farm1</a>	2023-11-27 15:40:59	7.5	29.375
<input type="checkbox"/>	<a href="#">aqua-farm1</a>	2023-11-27 15:41:11	7.62	29.375
<input type="checkbox"/>	<a href="#">aqua-farm1</a>	2023-11-27 15:41:22	7.59	29.312

**Figure 5.33: DynamoDB with data stored in**

For this project, the storage service in AWS Cloud that was chosen to be used was AWS DynamoDB. DynamoDB is a database that can be used to store processed data with fast retrieval time. The data stored inside DynamoDB is usually used for various objectives such as real-time data analysis and machine learning. In our proposed solution, the data will be used for creating a dashboard further on in Amazon Quicksight with DynamoDB as its primary data source. Figure 5.33 shows the data items that have already been stored inside the table “environment1DB”, the device\_id is used as the partition key and timestamp as the sort-key which forms a composite key for unique identification of each row inside the database.

5.6.4 Amazon SNS

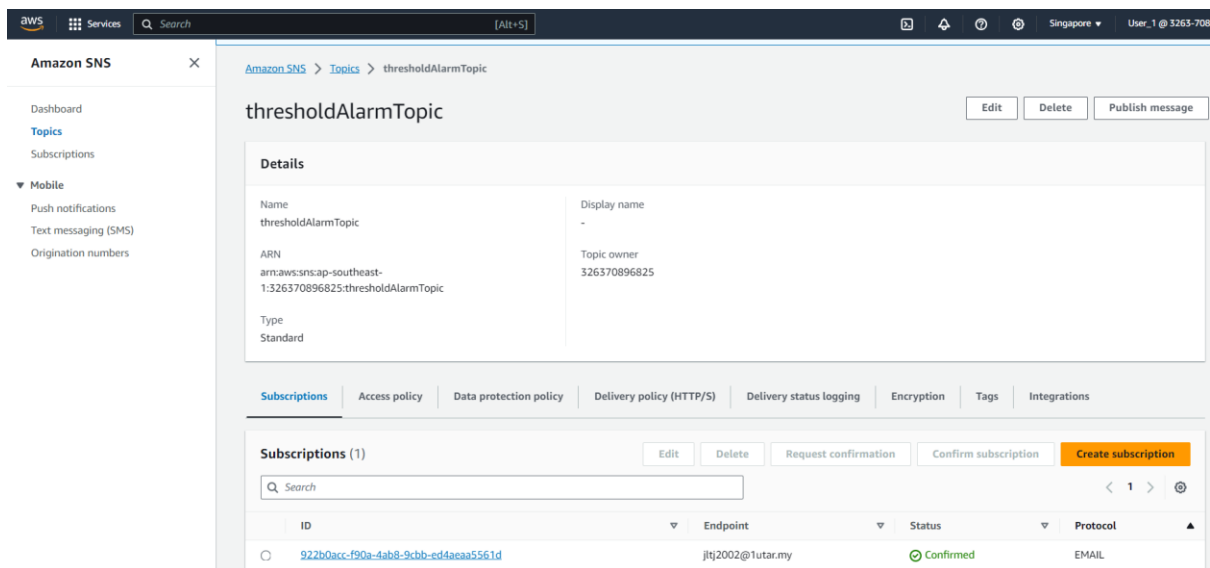
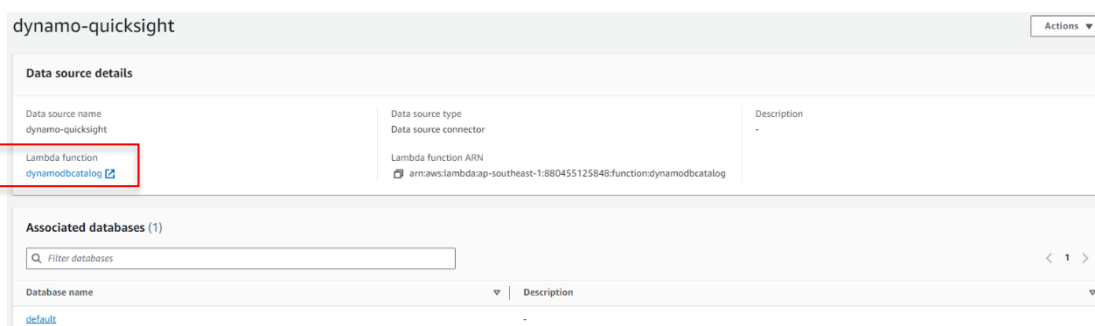


Figure 5.34: Detail of SNS Topic

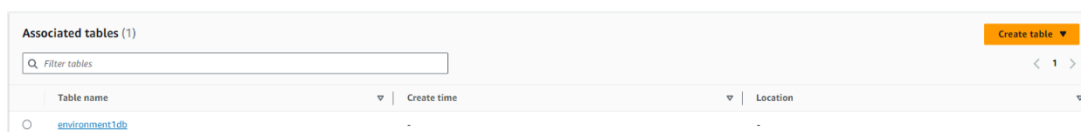
Amazon SNS is a managed messaging service which can be used to enable messages to be sent in different delivery methods to a distributed group of recipients. The available platforms are email, SNS and other channels. For our implementation, the SNS topic will be established as the destination for the “com.example.testAlarmv2” component on edge environment devices, it allows them to send messages for alerting the users when water parameters have been breached. In figure 5.34 we can see there is a subscription of a recipient who will receive the message protocol at a specific endpoint, in this case through email. When the alarm component is triggered, the boto3 library will invoke this service to the specific topic and subscriptions.

### 5.6.5 Amazon Athena

AWS Athena is a query service that is interactive and allows the execution of SQL queries from selected data sources in various different formats. For this project, AWS Athena has the responsibility of data communication between DynamoDB and Quicksight for data visualization purposes. The reason this service was chosen was due to the inability for NoSQL data stored in DynamoDB to be directly passed into Quicksight. DynamoDB will serve as the primary data source used for the visualization dashboard as well as the final backup for each of the edge environments.



**Figure 5.35: Details of Athena Data Source dynamo-quicksight**



**Figure 5.36: Table Specified in Associated Databases**

From the figures above, we can see the data source named “dynamodbQuicksight” was created to serve as the data source for Amazon Quicksight which queries its data from DynamoDB. A lambda function named “dynamodbcatalog” is also created to be the connector for facilitating federated queries from the associated database within DynamoDB. From the associated database, we can see that the table “environment1db” was chosen where all the processed data is being stored. All the data stored in this table will be queried out by Athena to be presented for visualization later on in the project.

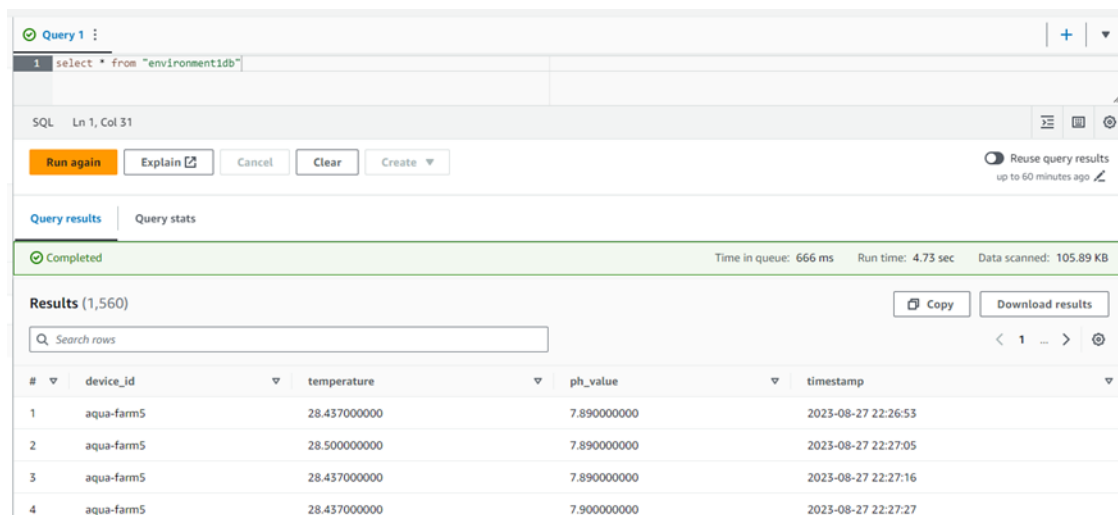


Figure 5.37: Athena Query Editor Testing

The figure above shows the query in Athena which was run on a simple SQL command through the query editor interface. The query (SELECT \* FROM “environment1db”) is ran and the lambda function is able to successfully retrieve the data from the chosen DynamoDB table as per the results displayed above.

### 5.6.6 Amazon Simple Storage Service (S3)

Amazon S3 is simple storage service which is scalable for object storage and allows storage for a wide range of data formats, for example images, files, documents and backups. All of these data are stored inside buckets for various use cases. For this project, the use of AWS S3 is for the intermediary during the federated learning process, it will facilitate the exchange and storage of model weights between the main federated learning server and the edge clients.

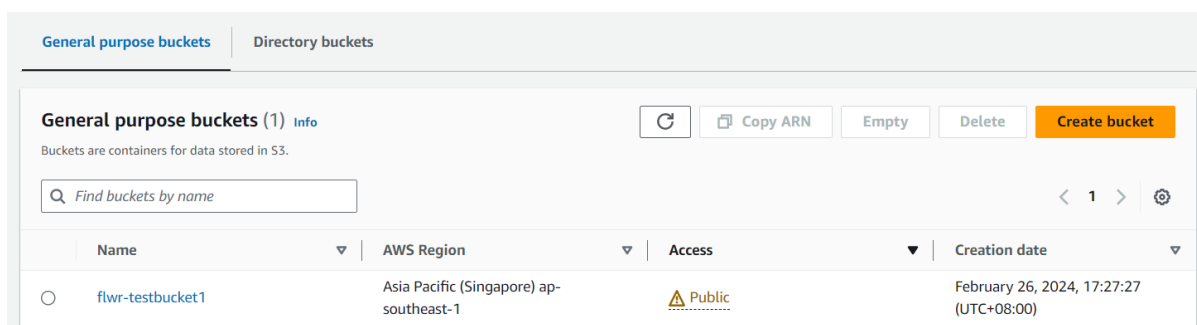


Figure 5.38: Bucket List in S3

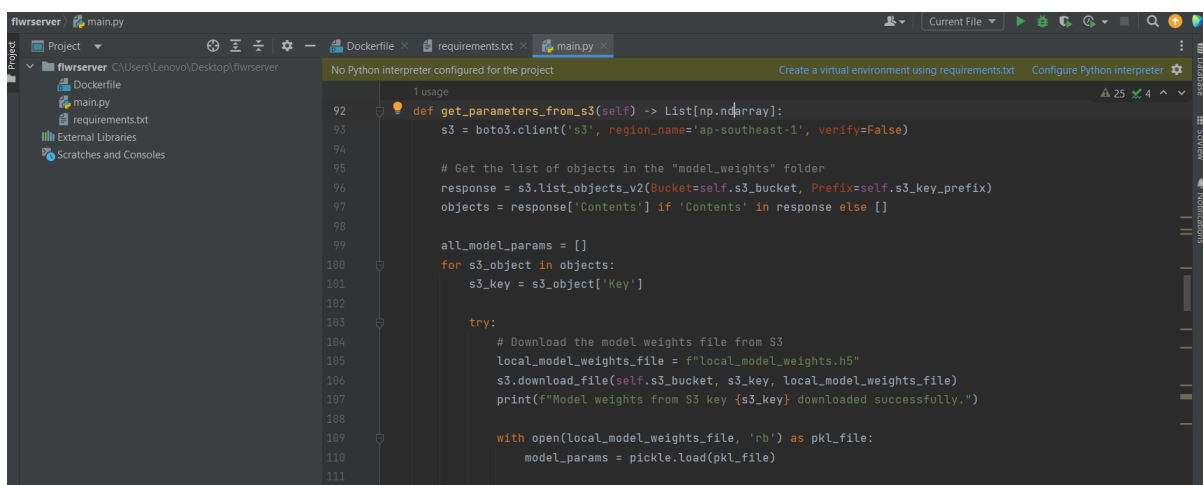
As seen from the figure above, our S3 bucket named “flwr-testbucket1” was created with versioning enabled due to the federated learning purposes. Bucket versioning in S3 will

## CHAPTER 5 SYSTEM IMPLEMENTATION

create new versions of an object stored when it is updated, the old version will still be retained. This is crucial for cases where the global ML model is negatively impacted by any client's model weights, we will still be able to rollback to a previous version of the model which is stable. The "flwr-testbucket1" bucket will be the central repository for storing both global and client ML model weights, it has the crucial role of storing objects for the server and client components during the federated learning process.

### 5.6.7 Amazon Elastic Container Registry (ECR)

Amazon ECR is a cloud service which is used for managing docker container registries. It allows the users to deploy their docker container images and integrate them seamlessly onto other AWS services such as ECS, Fargate and EKS. For this project, ECR will be used to upload the docker image files and server codes for federated learning server into the cloud to be stored as a private registry for further usage.



```
flwrserver main.py
Project
  flwrserver
    Dockerfile
    main.py
    requirements.txt
  External Libraries
  Scratches and Consoles

No Python interpreter configured for the project
Create a virtual environment using requirements.txt
Configure Python interpreter

92 def get_parametens_from_s3(self) -> List[np.ndarray]:
93     s3 = boto3.client('s3', region_name='ap-southeast-1', verify=False)
94
95     # Get the list of objects in the "model_weights" folder
96     response = s3.list_objects_v2(bucket=self.s3_bucket, Prefix=self.s3_key_prefix)
97     objects = response['Contents'] if 'Contents' in response else []
98
99     all_model_params = []
100     for s3_object in objects:
101         s3_key = s3_object['Key']
102
103         try:
104             # Download the model weights file from S3
105             local_model_weights_file = f"local_model_weights.h5"
106             s3.download_file(self.s3_bucket, s3_key, local_model_weights_file)
107             print(f"Model weights from S3 Key {s3_key} downloaded successfully.")
108
109             with open(local_model_weights_file, 'rb') as.pkl_file:
110                 model_params = pickle.load(pk1_file)
111
```

**Figure 5.39: Docker files that will be uploaded into ECR as image repository**

The figure above shows the codes used for our federated learning server opened in PyCharm. The main python file along with the dockerfile and requirements text file will be uploaded into AWS ECR. The docker container image will further be used for running our federated learning server by integrating it with AWS Fargate which is a serverless compute engine.

## CHAPTER 5 SYSTEM IMPLEMENTATION

### 5.6.8 Amazon Elastic Container Service (ECS)

Amazon ECS is a container orchestration service that is used to streamline the management, scaling and deployment of containerized applications. This service is used to host the federated learning server that will be the orchestrator of the federated learning process across all the edge environments that is located within the ECS cluster.

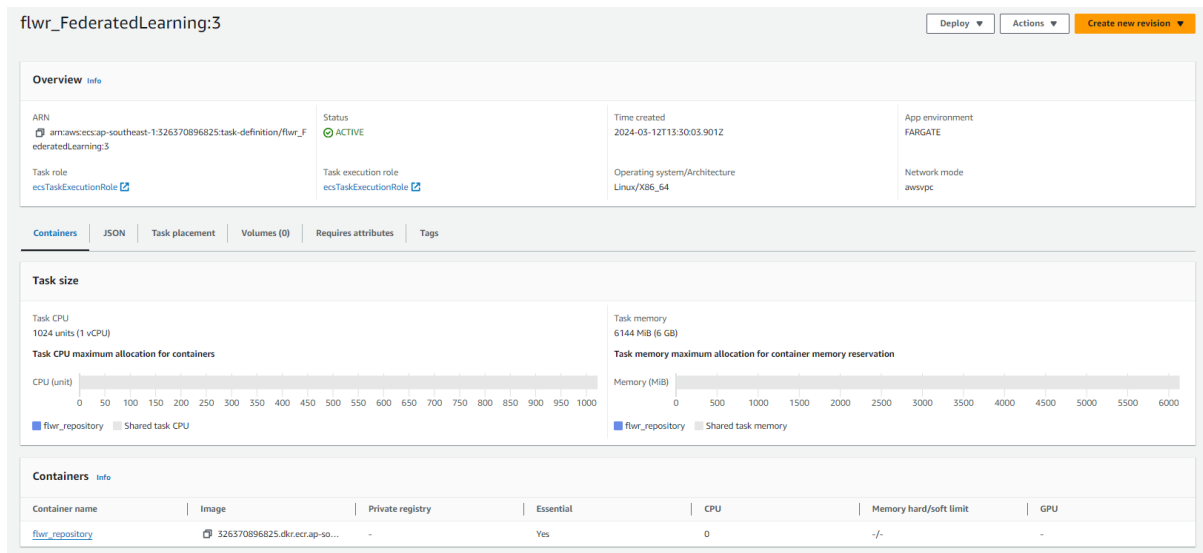


Figure 5.40: Detail of Task Definition Created from Docker Image

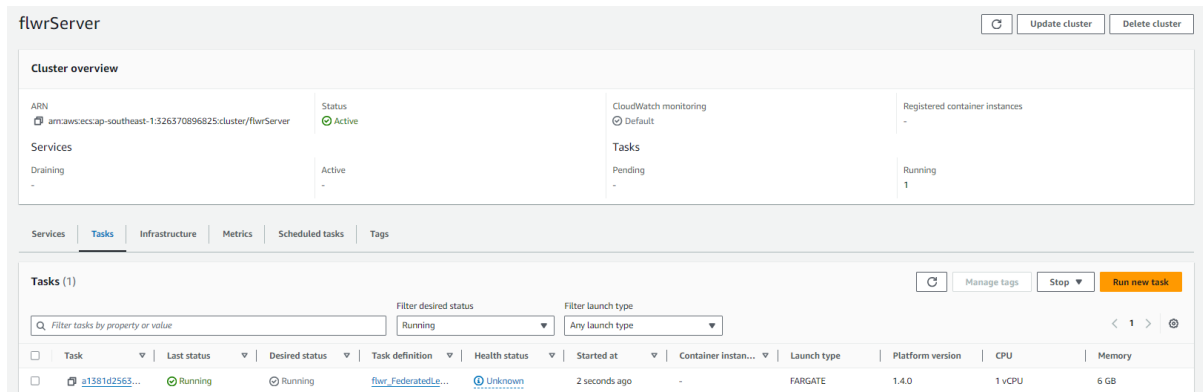


Figure 5.41: The Federated Learning Server Running as a Task in ECS Cluster

From the figure above we can see the task definition created in ECS through the use of Docker images which is stored in ECR repository. The resource requirements, parameters, configurations and network settings are specified inside the task definition. The containers section below specifies the Docker repository where the image is utilized from. The task definition will act as the federated learning server with 1 container however task definitions can usually accommodate up to maximum 10 different containers at a time.



### 5.6.9 Amazon Virtual Private Cloud

Amazon Virtual Private Cloud is a service that allows users to create isolated virtual networks within AWS cloud, users can define their own private virtual network topologies along with IP address ranges, subnets, network routing and gateways. This allows them to control their cloud environment’s network infrastructure with ease.

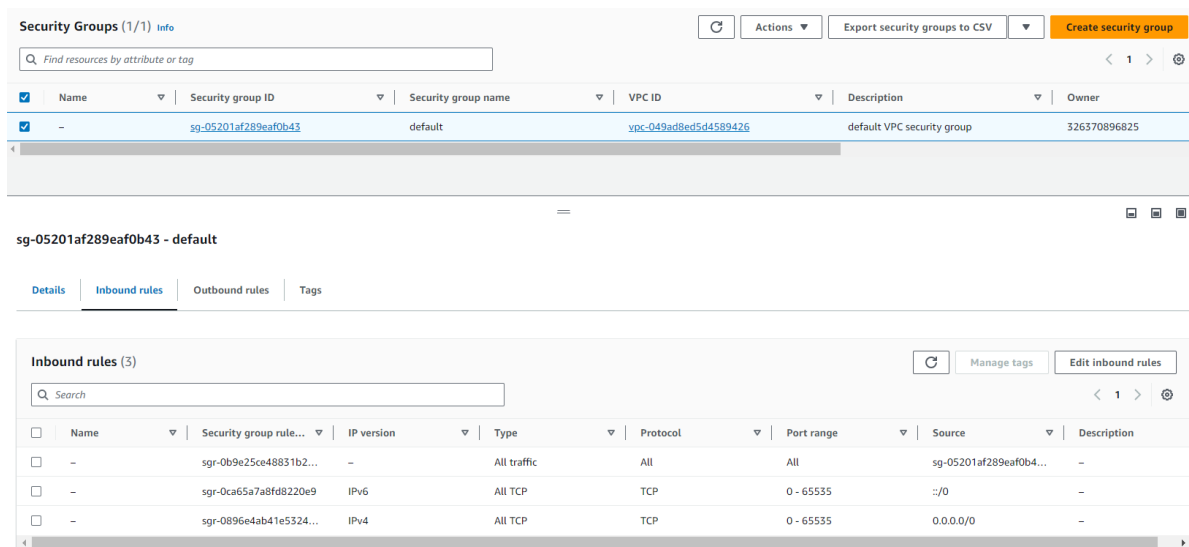


Figure 5.42: Detail of Security Group configurations for inbound rules

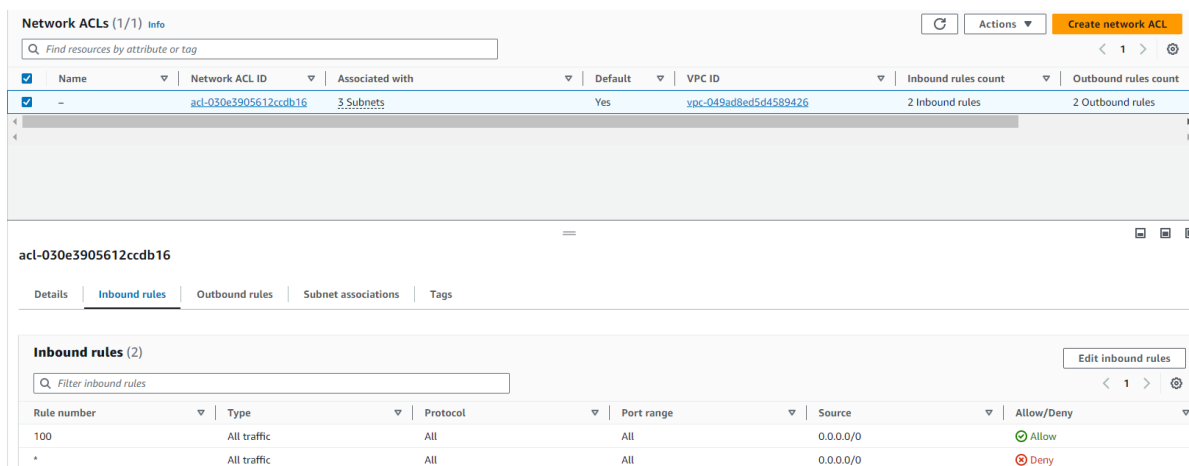


Figure 5.43: Detail of Network ACL configurations for inbound rules

From both figures above we can see that AWS VPC has been used in 2 ways for this project which is to first create 2 extra inbound rules for our default security group to allow IPv4 and IPv6 addresses to communicate with our network when their protocols are TCP, this will be associated with our ECS server which runs on the default security group to communicate with TCP messages on port 8080. The second use is to edit the inbound rules for the network ACL

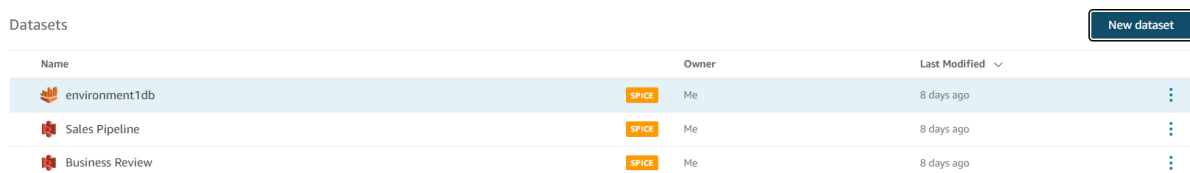
## CHAPTER 5 SYSTEM IMPLEMENTATION

to allow all traffic entering the default VPC for all subnets of our AWS region which is “ap-southeast-1” (Singapore).

### 5.6.10 Amazon Quicksight

Amazon Quicksight is a BI tool which can be used to create data visualizations through the analysis of data on the dashboards, this facilitates better decision making driven by data analysis. For this project, Quicksight is integrated with DynamoDB to visualize the processed data captured by the edge environments for near real-time data visualization on an interactive dashboard. The dashboard will also be designed so that the data presented will be compact and insightful by only displaying essential information, this ensures that the necessary information is presented towards the farmer to aid in their everyday decision making process.

The datasets that are passed into Quicksight will first be filtered and sourced in Athena by querying the specified DynamoDB table. From the figure below we can see that the table “environment1db” has been added as a dataset for the visualization process. Quicksight allows its users to include multiple datasets without regards of the source of these datasets.



The screenshot shows the 'Datasets' page in Amazon Quicksight. It features a 'New dataset' button in the top right corner. Below the header, there is a table with three columns: 'Name', 'Owner', and 'Last Modified'. The table lists three datasets: 'environment1db', 'Sales Pipeline', and 'Business Review'. Each dataset entry includes a small icon, a 'SPICE' status indicator, the owner's name ('Me'), and the last modified date ('8 days ago'). A vertical ellipsis menu is visible to the right of each row.

Name	Owner	Last Modified
environment1db	Me	8 days ago
Sales Pipeline	Me	8 days ago
Business Review	Me	8 days ago

**Figure 5.44: List of Datasets in Quicksight**

The figure above shows the interface used to develop the dashboard. The highlight of Quicksight is its vast selection of smart graphs and visualization options which users can choose from to present their datasets. Quicksight also allows users to create calculated fields from their existing datasets to derive new data insights. Basic forecasting features through the integration of ML-powered insights are also available in Quicksight if they are required. The end product which is a working interactive dashboard will then be published for authorized access by users.

## CHAPTER 5 SYSTEM IMPLEMENTATION

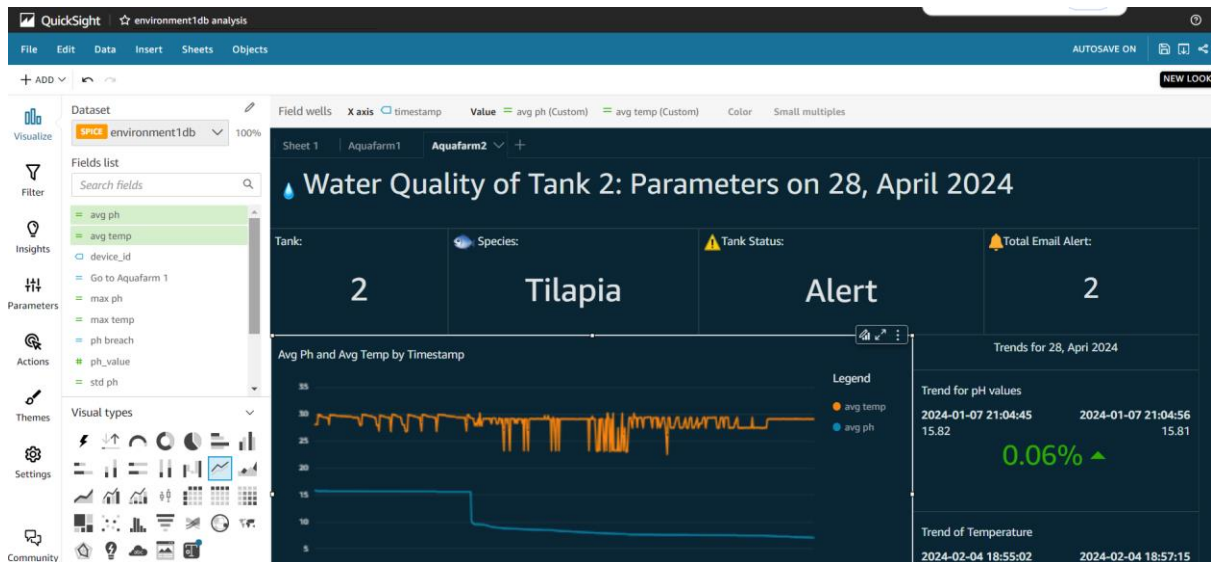


Figure 5.45: Quicksight Dashboard Developer Interface

### 5.7 Federated Learning Framework Implementation

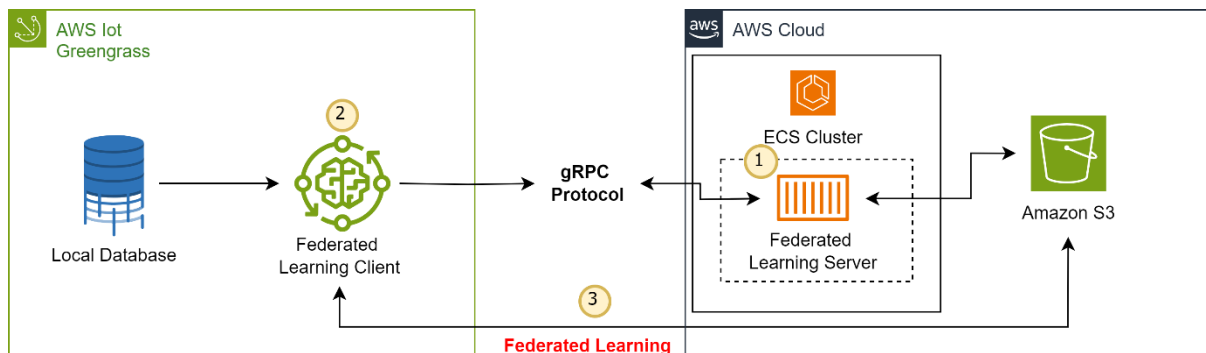


Figure 5.46: Federated Learning Module from Architecture Diagram

In figure 5.46 Above, we can see the architecture diagram for the federated learning framework which is implemented in this project. All the services and components needed for the framework has been prepared and added for service. At each of the edge environments, the federated learning process is represented by clients that are implemented through the use of “com.example.flwrClient2” greengrass component which is deployed along with the other components. Other than that, another major component deployed is the “com.example.sensorv5” component which is required for data capturing and storing in local database. As for the federated learning server which is implemented in the cloud, it will take the form of an ECS task definition which will be ready for deployment whenever possible. A

## CHAPTER 5 SYSTEM IMPLEMENTATION

ECS cluster named “flwrServer” will also be created to allocate the tasks based on the scheduled conditions which have been set. An S3 bucket called “flwr-testbucket1” will be used as the central repository for the aggregation of global and local client ML weights. This whole process is enabled through the communication between client and cloud server via gRPC protocol on port 8080. All of these services and components combined will form the whole federated learning process.

5.7.1 Comparison of Federated Learning Frameworks

There are several different federated learning frameworks which are available for use in the market currently, each of them have distinct differences in practice along with their own advantages and disadvantages. The framework which will be chosen to conduct the federated learning process in this project should be the most suitable framework to be integrated with the use of cloud services and edge environment qualities.

Feature	Flower	TensorFlow Federated (TFF)	PySyft
Programming Language	Python	Python	Python, PyTorch, TensorFlow
Scalability	High	High	High
Ease of Use	High	Moderate	Moderate
Customization	Extensive	Moderate	Moderate
Community Support	Growing	Active	Growing
Performance Optimization	Yes (communication efficiency)	Yes (optimizations)	Yes (secure multi-party computation)
Documentation	Comprehensive	Extensive	Moderate
Deployment Options	Various	Various	Various

Table 5.1: Comparison of different Federated Learning Frameworks

After comparing all the available frameworks in the market, Flower was chosen as the federated learning framework to be used in this project. Even though most of the frameworks discovered had similarities across many aspects of comparison, the main reason Flower was chosen was because of its ease of use. This is due to its high level of abstraction which lowers the overall complexity, higher level API's and tools are provided in Flower so that developers can focus on the ML task instead of the intricacies of the federated learning protocols. Another big reason is due to its wide community support and documentation, extensive documentation and tutorials can be found online which makes it easier for beginners to get started with the federated learning development along with the help of experienced researchers contributions for support.

### 5.7.2 Scheduled Deployment of Federated Learning Server

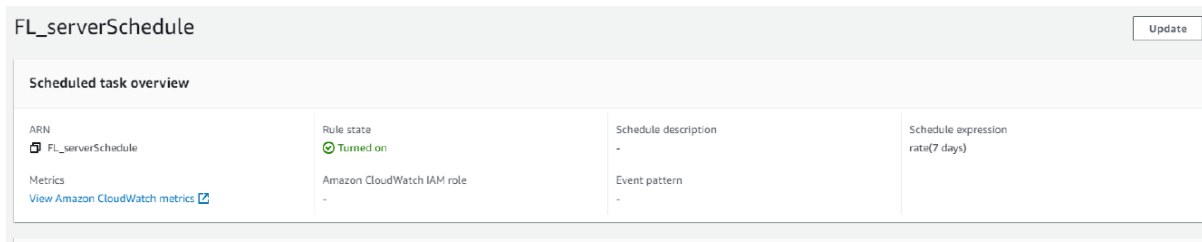


Figure 5.47: Schedule Task with Target Task Definition

For this project, the federated learning server that resides in the cloud will be initiated through a scheduled timestamp activation. The timestamp which was set for activating the process was one week so that the edge environments would have sufficient time to collect their local data and perform local ML training for model improvements. From the figure above we can see that the scheduler’s task is to dispatch the Fargate server with its defined task definition for the federated learning server to run when the condition set has been met.

### 5.7.3 Setting up DNS Name

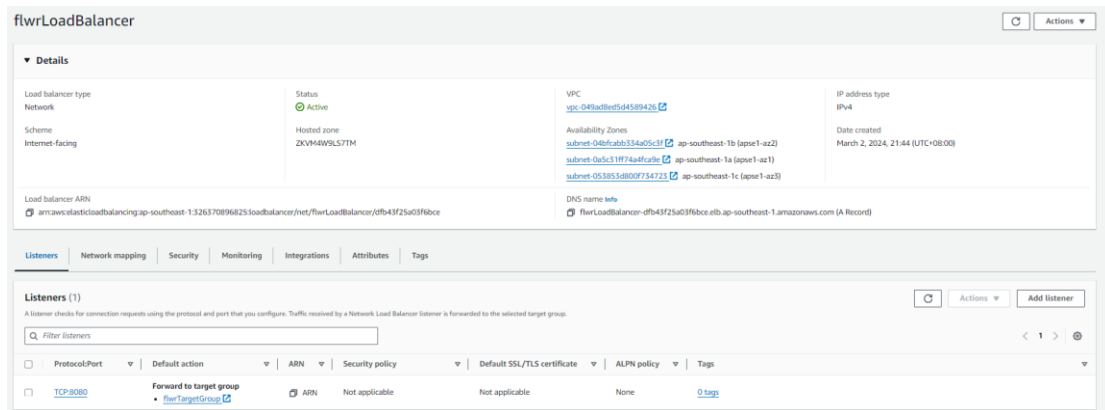


Figure 5.48: Configuration of Network Load Balancer flwrLoadBalancer

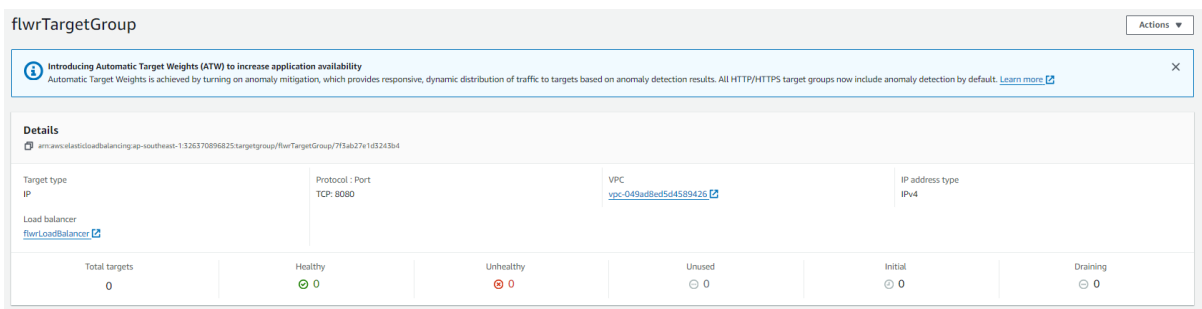


Figure 5.49: Configuration of Target Group flwrTargetGroup

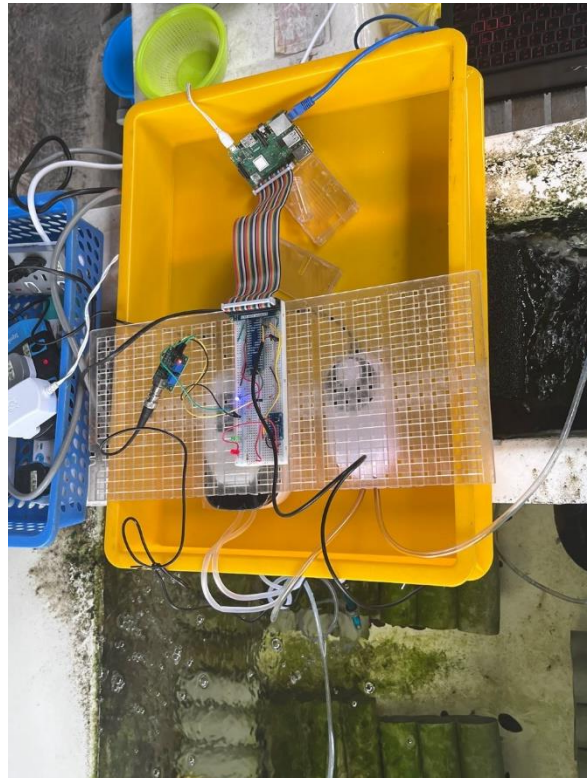
## CHAPTER 5 SYSTEM IMPLEMENTATION

After the federated learning server has been activated and running, it has to be accessible by the clients in the edge environments. The clients can access the server through its generated public IP address, however the IP address generated will be random and changing every time. Hence, without a specific IP address for each activation of the server, a DNS name is needed so that the clients can access the server without needing to specify its address each time.

To provide the server with a DNS name, an EC2 network load balancer was configured which can be seen in figure 5.48, it relies on the gRPC protocol on port 8080 to allow for communication between the clients and server. Other than that, in figure 5.49, we can also see the target group created so that the correct traffic will be forwarded onto the server through the load balancer. Once this is done, any incoming traffic on port 8080 which is not TCP will automatically be dropped to ensure only the correct protocol and port number traffic is allowed to reach the server. The DNS name will be associated with the load balancer to retrieve the server's IP address.

## CHAPTER 6: System Deployment and Evaluation

### 6.1 Hardware Deployment



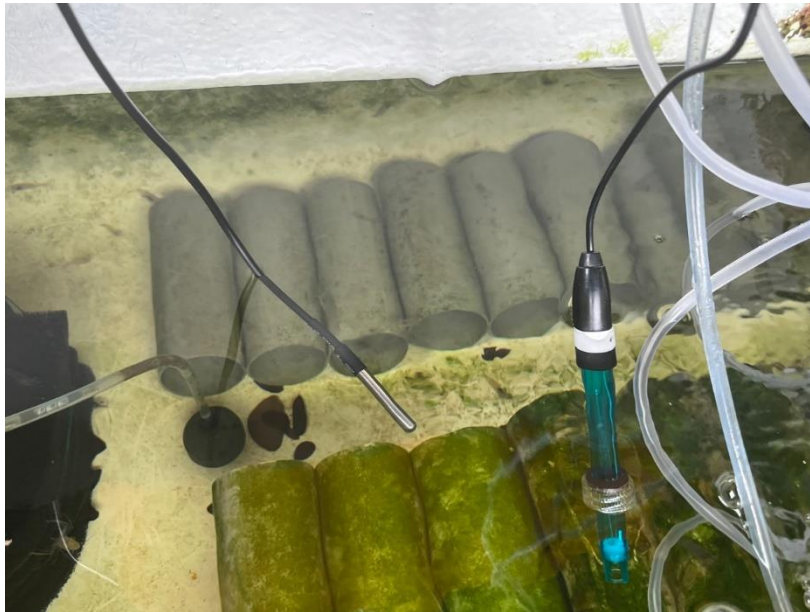
**Figure 6.1: Hardware Deployment at Aquafarm**



## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION



**Figure 6.2 Hardware Deployment at office tank**



**Figure 6.3: DS18B20 and PH4502C Sensor immersed in Water**



**Figure 6.4 Local Occupants**

Figure 6.1 above displays the entire edge environment IoT system being deployed in 2 different aquaculture tanks which are used for prawn farming. The first tank is filled with 80 freshwater big head prawns and some Gourami orange fishes along with aquatic plants to form a suitable living habitat for the prawns. It is also equipped with a water drainage and filtration system to recycle and filter the water in the tank, several oxygen pumps are also present to pump in oxygen for the prawns. As for the second smaller tank, it rears Langostino which are mini lobsters. The Raspberry Pi machine which acts as the Greengrass core device along with the integrated breadboard that holds the sensor devices is incorporated into both environments. The red LED lights on the breadboard indicate that the system is powered on but not activated, once the sensors are activated through MQTT protocols, the green and blue LEDs will light up. From figure 6.3 we can also see a close up of the DS18B20 temperature sensor and PH4502C sensors submerged into the water for data collection.

Both setups will be used to represent the edge environments, data collection, preprocessing and storage of water parameters will be conducted locally to mimic the daily operations of aquaculture farms using the Raspberry Pi and Greengrass components. The operations can also be performed without the need of network connectivity unless the data needs to be passed into the cloud for other tasks. When alarm breaches happen, an emergency network connection will be activated momentarily to trigger the alarm for the parameter breaches instead. The processed data for each edge environment will be visualized and presented onto a dashboard inside the cloud.

### 6.2 System testing

After the hardware has been successfully deployed into the aquaculture tanks, it is crucial to start performing comprehensive testing so that we can ensure all edge environments functionalities are able to be performed with the cloud services as intended. It is important for us to make sure that the federated learning framework can be performed as it should and the end result should be a working federated learning framework with a dashboard for processed data visualization.

#### 6.2.1 Greengrass Components Testing

This project consists of a total of 4 Greengrass private components which have all serve a specific purpose in each of the edge environments. They are the sensor reading and processing component “com.example.sensrov5”, a rule-based alert component that works together with it “com.example.testAlarmv2” and also a data synchronization component for data between the local and cloud databases “com.example.sync”. There is an additional component for federated learning client activation which will be tested separately. The 3 functions mentioned work together closely hence they will be tested to ensure the functionalities have been achieved.

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

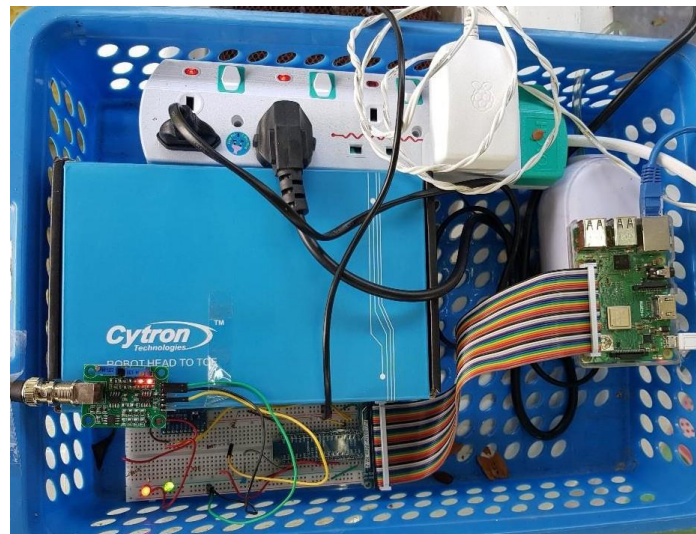


Figure 6.5: Green LED illuminating on IoT Device

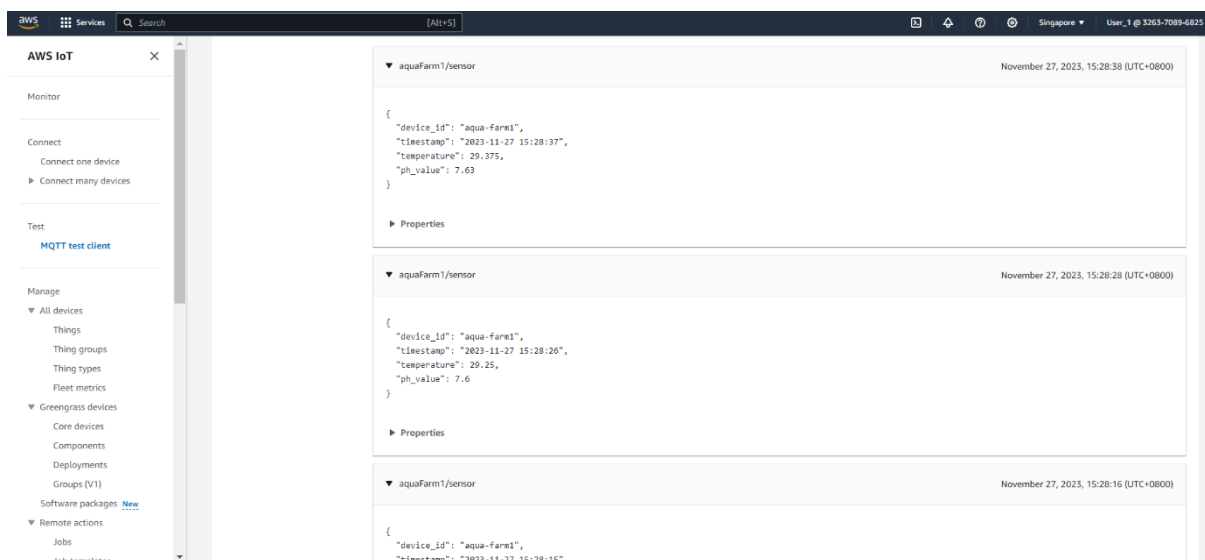


Figure 6.6: MQTT Test Client receiving message from subscribed topic

The figure above shows the sensor reading data being passed into the cloud when the component has been triggered through the MQTT test client in AWS console. Once the component is activated, the green LED on the device will start to illuminate to indicate that the sensors are switched on for data capturing. All the data which has been captured will be pre-processed and then stored into the local database. Simultaneously, the processed data will also be transmitted to the cloud for backup in the form of pub and sub modules in specific topics. This results in the image shown in figure 6.6 where the client receives the messages when they subscribe to the topic.

## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

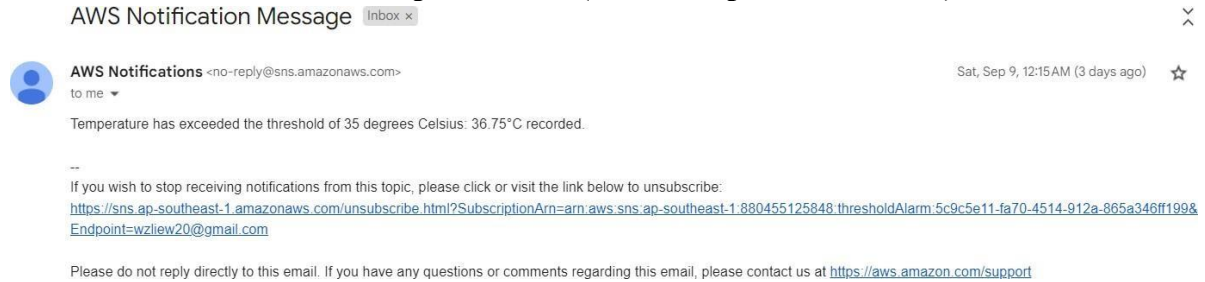
The figure consists of two side-by-side terminal windows. The left window, titled 'geany\_run\_script\_NLD5A2.sh', shows a Python script snippet: `conn = sqlite3.connect(db_file_path)`, `cursor = conn.cursor()`, and `cursor.execute('SELECT * FROM sensor_data')`. Below the script, a table of data is displayed with columns for farm name, timestamp, and two numerical values. The right window, titled 'geany\_run\_script\_HH2MB2.sh', shows a similar script snippet but with the query `cursor.execute('SELECT * FROM temp_sensor_data')`. It displays an identical table of data. Both tables contain 18 rows of data.

Farm Name	Timestamp	Value 1	Value 2
'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

**Figure 6.7: Content of Main Database (Left) and Temporary Database (Right)**

When there is an absence of network connectivity, the processed data will also be stored into the local database which is shown in the left side of the figure above. The data will be published to another topic which is called “loc/sensor” once the network connectivity is restored. The data is not only stored in the local database but also in a temporary database as shown in the right side of the figure above. The content in both databases are identical which means the component is functioning as it was intended. The purpose of the temporary database is to be used in the synchronization component for checking purposes when network connectivity is restored.

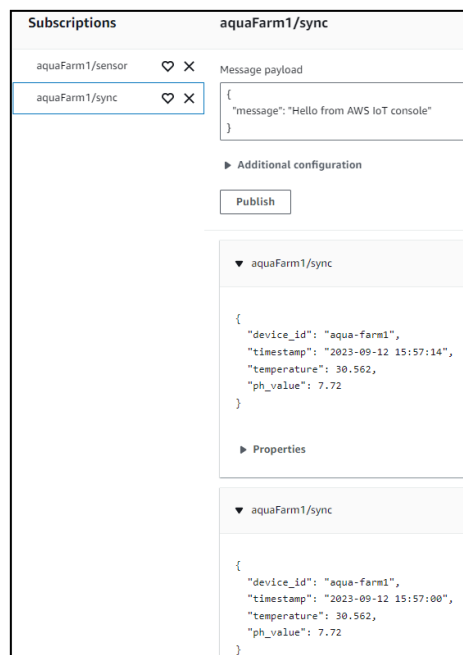
### 6.2.1.2 Rule-Based Alert Component Test (com.example.testAlarmv2)



**Figure 6.8: Email Message from AWS SNS**

At the same time, a rule-based alert component is also running which subscribes to the local topic “loc/sensor” which assesses the processed data to check if there have been any breaches of water parameters by comparing them against the preset threshold values. In the event of a water parameter breach, the component will trigger an alarm by sending an email to the user with the content displayed on figure 6.8. The email will inform the user regarding the particular parameter that was breached and the relevant parameter values for further reference.

### 6.2.1.3 Data Synchronization Component Test (com.example.sync)



**Figure 6.9: MQTT Receiving Message from subscribed synchronization topic**

As for the data synchronization component, this component runs on intervals of 5 minutes to check for internet connectivity status when there is an absence of network. Once an internet connection is detected, it will start by verifying if the temporary database on each edge device

## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

is empty. If there is data inside the database, the component will then initiate the synchronization process with the DynamoDB table in the cloud.

From figure 6.9 we can see that the topic “aquafarm1/sync” which is actively receiving messages from the synchronization component which confirms that it is functioning properly. This is not always the case because Greengrass has a message queuing functionality for MQTT messages which is used to store extensive amount of data during offline periods. Regardless, once the connection has been restored, the MQTT queue will be sent to the cloud. The advantage of the synchronization function is supporting the limited capacity of the MQTT queue to ensure the reliability of data synchronization.

## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

### 6.2.2 Greengrass Components Verification Test

Components	Condition	Functionality	Result	Pass / Fail	
<b>Data Reading and Preprocessing (6.1.2.1)</b>	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 temporary database (temp_sensro_data.db)	Processed data store into temporary database (temp_sensor_data.db)	Pass
			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	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	



## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

	-With network connection			
<b>Rule-Based Alert (6.1.2.2)</b>	-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
<b>Data Synchronization (6.1.2.3)</b>	-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 Test table**

6.2.3 AWS Cloud Testing

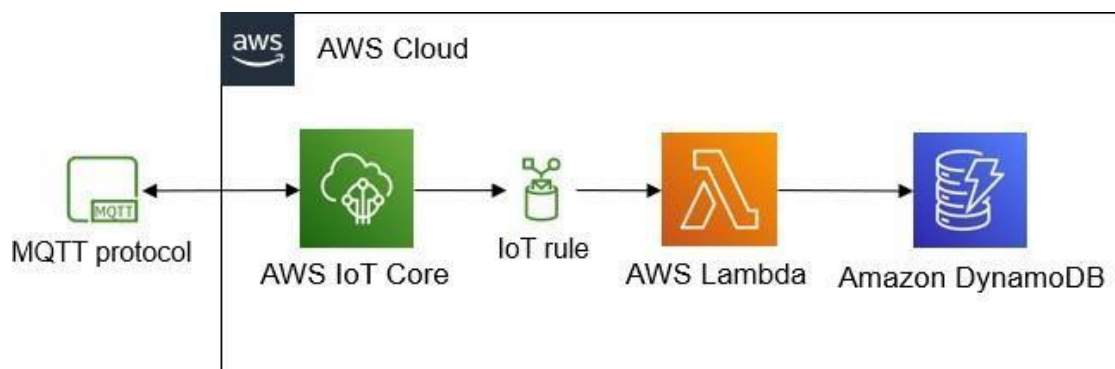


Figure 6.10: Data routing section from Architecture Diagram

The screenshot shows the AWS DynamoDB console interface. At the top, there are buttons for 'Scan' (selected) and 'Query'. Below these, there are dropdown menus for 'Select a table or index' (set to 'Table - environment1DB') and 'Select attribute projection' (set to 'All attributes'). There are also 'Filters' and 'Run' buttons. The main area displays a message: 'This table has more items to retrieve. To retrieve the next page of items, choose Retrieve next page.' Below this, a table shows 'Items returned (50)'. The table has four columns: 'device\_id (String)', 'timestamp (String)', 'ph\_value', and 'temperature'. The data rows show multiple entries for 'aqua-farm1' with timestamps from 2023-11-27 15:40:14 to 2023-11-27 15:41:22, and corresponding ph\_value and temperature readings.

device_id (String)	timestamp (String)	ph_value	temperature
aqua-farm1	2023-11-27 15:40:14	7.5	29.375
aqua-farm1	2023-11-27 15:40:25	7.51	29.375
aqua-farm1	2023-11-27 15:40:37	7.51	29.312
aqua-farm1	2023-11-27 15:40:48	7.5	29.312
aqua-farm1	2023-11-27 15:40:59	7.5	29.375
aqua-farm1	2023-11-27 15:41:11	7.62	29.375
aqua-farm1	2023-11-27 15:41:22	7.59	29.312

Figure 6.11: Items in DynamoDB table environmentDB1

The way to validate if the implementation for the AWS services were correct is to examine the target destination for the successfully captured and pre-processed data such as the structure depicted inside figure 6.10 above. Inside DynamoDB, the targeted table has been stored with the desired data values as we can see, this indicates that the AWS IoT configuration for subscribing to the topics “aquaFarm1/sensor” and “aquafarm/sync” together with the use of IoT rules for routing the subscribed messages towards a Lambda function for processing which in the end stores them into DynamoDB is functioning successfully as intended.

## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

### 6.2.4 Federated Learning Module Testing

The minimum number of active clients required to conduct a test in the federated learning module with the main server inside the cloud is 2, since we have 2 edge devices which are active clients, we can run the federated learning client which has already been deployed onto both edge devices. When the main server which is represented by the ECS cluster inside the cloud dispatches the federated learning server during scheduled intervals, the federated learning process should commence with the participation of 2 clients in total.

```
2024-03-23T17:40:48.358+08:00 2024-03-23 09:40:48.357951: I external/local_tsl/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2024-03-23T17:40:48.408+08:00 2024-03-23 09:40:48.408289: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one...
2024-03-23T17:40:48.408+08:00 2024-03-23 09:40:48.408348: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one...
2024-03-23T17:40:48.409+08:00 2024-03-23 09:40:48.409631: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one...
2024-03-23T17:40:48.417+08:00 2024-03-23 09:40:48.417898: I external/local_tsl/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2024-03-23T17:40:48.418+08:00 2024-03-23 09:40:48.418150: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-03-23T17:40:49.343+08:00 2024-03-23 09:40:49.343103: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2024-03-23T17:40:51.698+08:00 INFO flwr 2024-03-23 09:40:51.697 | app.py:163 | Starting Flower server, config: ServerConfig(num_rounds=3, round_timeout=None)
2024-03-23T17:40:51.713+08:00 INFO flwr 2024-03-23 09:40:51.713 | app.py:176 | Flower ECE: gRPC server running (3 rounds), SSL is disabled
2024-03-23T17:40:51.713+08:00 INFO flwr 2024-03-23 09:40:51.713 | server.py:89 | Initializing global parameters
2024-03-23T17:40:51.936+08:00 INFO flwr 2024-03-23 09:40:51.936 | server.py:276 | Requesting initial parameters from one random client
2024-03-23T17:45:55.905+08:00 INFO flwr 2024-03-23 09:45:55.904 | server.py:280 | Received initial parameters from one random client
2024-03-23T17:45:55.905+08:00 INFO flwr 2024-03-23 09:45:55.905 | server.py:91 | Evaluating initial parameters
2024-03-23T17:45:55.905+08:00 INFO flwr 2024-03-23 09:45:55.905 | server.py:104 | FL starting
2024-03-23T17:47:08.363+08:00 DEBUG flwr 2024-03-23 09:47:08.363 | server.py:222 | fit_round 1: strategy sampled 2 clients (out of 2)
2024-03-23T17:47:17.941+08:00 DEBUG flwr 2024-03-23 09:47:17.941 | server.py:236 | fit_round 1 received 2 results and 0 failures
2024-03-23T17:47:17.941+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
2024-03-23T17:47:17.983+08:00 warnings.warn(
/usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
```

Figure 6.12: CloudWatch Event Log for Federated Learning Server (1)

```
2024-03-23T17:47:17.983+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
2024-03-23T17:47:17.983+08:00 warnings.warn(
2024-03-23T17:47:17.998+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
2024-03-23T17:47:17.998+08:00 warnings.warn(
2024-03-23T17:47:18.016+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
2024-03-23T17:47:18.016+08:00 warnings.warn(
2024-03-23T17:47:18.032+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
2024-03-23T17:47:18.032+08:00 warnings.warn(
2024-03-23T17:47:18.049+08:00 DEBUG flwr 2024-03-23 09:47:18.049 | server.py:173 | evaluate_round 1: strategy sampled 2 clients (out of 2)
2024-03-23T17:47:19.625+08:00 DEBUG flwr 2024-03-23 09:47:19.625 | server.py:187 | evaluate_round 1 received 2 results and 0 failures
2024-03-23T17:47:19.625+08:00 WARNING flwr 2024-03-23 09:47:19.625 | fedavg.py:281 | No evaluate_metrics_aggregation_fn provided
2024-03-23T17:47:19.626+08:00 DEBUG flwr 2024-03-23 09:47:19.625 | server.py:222 | fit_round 2: strategy sampled 2 clients (out of 2)
2024-03-23T17:47:21.164+08:00 DEBUG flwr 2024-03-23 09:47:21.164 | server.py:236 | fit_round 2 received 2 results and 0 failures
2024-03-23T17:47:21.184+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
2024-03-23T17:47:21.184+08:00 warnings.warn(
2024-03-23T17:47:21.225+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
2024-03-23T17:47:21.225+08:00 warnings.warn(
2024-03-23T17:47:21.241+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
2024-03-23T17:47:21.241+08:00 warnings.warn(
2024-03-23T17:47:21.259+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amazonaws.com'. Use `verify=False` if you are aware of this warning.
2024-03-23T17:47:21.259+08:00 warnings.warn(
```

Figure 6.13: CloudWatch Event Log for Federated Learning Server (2)

## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

```

▶ 2024-03-23T17:47:21.275+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amaz...
▶ 2024-03-23T17:47:21.275+08:00 warnings.warn(
▶ 2024-03-23T17:47:21.308+08:00 DEBUG flwr 2024-03-23 09:47:21,308 | server.py:173 | evaluate_round 2: strategy sampled 2 clients (out of 2)
▶ 2024-03-23T17:47:22.021+08:00 DEBUG flwr 2024-03-23 09:47:22,021 | server.py:187 | evaluate_round 2 received 2 results and 0 failures
▶ 2024-03-23T17:47:22.021+08:00 DEBUG flwr 2024-03-23 09:47:22,021 | server.py:222 | fit_round 3: strategy sampled 2 clients (out of 2)
▶ 2024-03-23T17:47:23.679+08:00 DEBUG flwr 2024-03-23 09:47:23,678 | server.py:236 | fit_round 3 received 2 results and 0 failures
▶ 2024-03-23T17:47:23.696+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amaz...
▶ 2024-03-23T17:47:23.697+08:00 warnings.warn(
▶ 2024-03-23T17:47:23.728+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amaz...
▶ 2024-03-23T17:47:23.728+08:00 warnings.warn(
▶ 2024-03-23T17:47:23.748+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amaz...
▶ 2024-03-23T17:47:23.748+08:00 warnings.warn(
▶ 2024-03-23T17:47:23.768+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amaz...
▶ 2024-03-23T17:47:23.768+08:00 warnings.warn(
▶ 2024-03-23T17:47:23.781+08:00 /usr/local/lib/python3.9/site-packages/urllib3/connectionpool.py:1061: InsecureRequestWarning: Unverified HTTPS request is being made to host 'flwr-testbucket1.s3.ap-southeast-1.amaz...
▶ 2024-03-23T17:47:23.781+08:00 warnings.warn(
▶ 2024-03-23T17:47:23.796+08:00 DEBUG flwr 2024-03-23 09:47:23,796 | server.py:173 | evaluate_round 3: strategy sampled 2 clients (out of 2)
▶ 2024-03-23T17:47:24.450+08:00 DEBUG flwr 2024-03-23 09:47:24,450 | server.py:187 | evaluate_round 3 received 2 results and 0 failures
▶ 2024-03-23T17:47:24.450+08:00 INFO flwr 2024-03-23 09:47:24,450 | server.py:153 | FL finished in 86.54580242599999999
▶ 2024-03-23T17:47:24.451+08:00 INFO flwr 2024-03-23 09:47:24,451 | app.py:226 | app_fit: losses_distributed [(1, 1.1550804993489405), (2, 1.0982608100726074), (3, 1.0755262791693627)]

```

Figure 6.14: CloudWatch Event Log for Federated Learning Server (3)

```

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/tmpvo0tfghx
43/43 [=====] - 0s 2ms/step - loss: 2.2089

```

Figure 6.15: Federated Learning Client Event Log

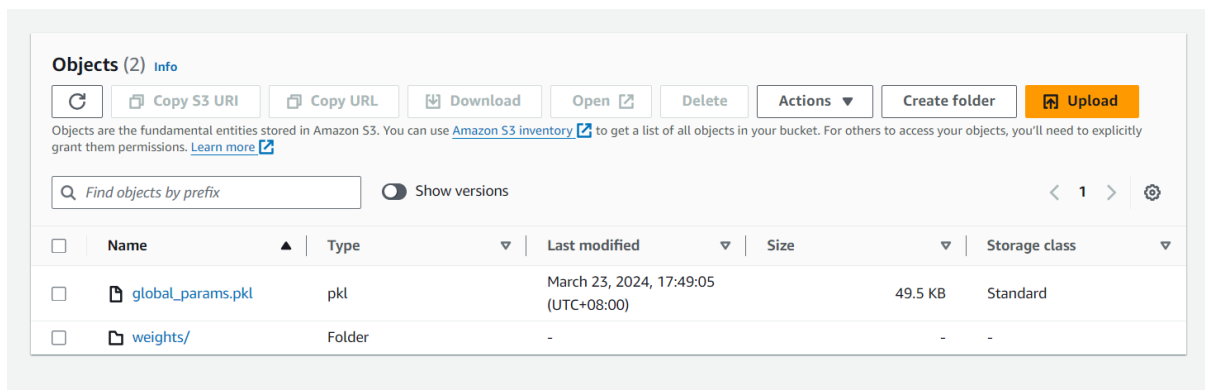


Figure 6.16: Global ML Model Weight uploaded in S3 bucket

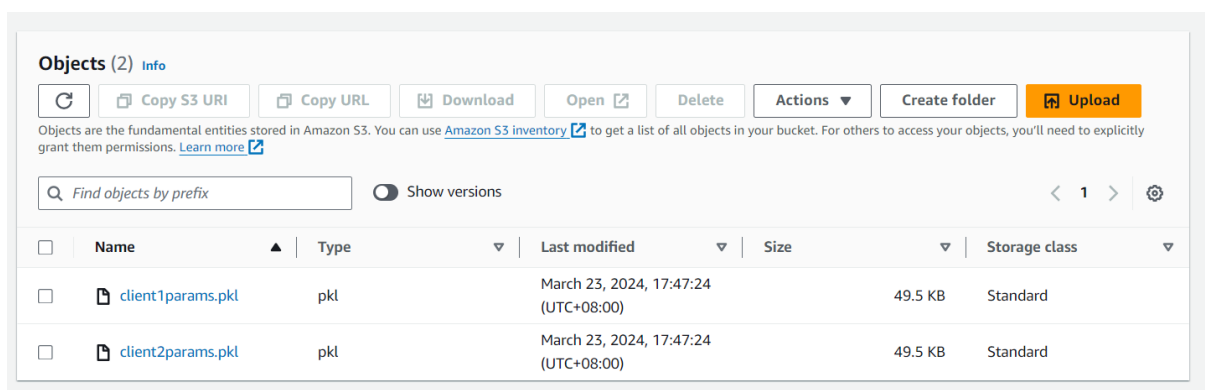


Figure 6.17: Client Model Weight uploaded in S3 bucket

From figures 6.12 to 6.14 above we can see the event logs for the ECS cluster inside AWS cloud which represents the actions ran on the main server for the federated learning process. Figure 6.15 is the client logs for each of the edge devices which are running the client processes individually. The server will be responsible for orchestrating the federated learning process by sending the necessary commands towards all the clients which are participating. The numbers highlighted in figure 6.12 and 6.15 are representing the steps which are performed during the federated learning process. The steps represent a single training round and will repeat for the total number of training rounds which are specified in the main server which is 3 times in our case.

## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

Steps:

**Step 1 (Server):** Server requests an initial ML model weight from a random client, this ML model will serve as the global model for the federated learning process.

**Step 1 (Client):** The random client which has been selected will upload its ML model weight to the S3 bucket in AWS.

**Step 2 (Server):** Server will orchestrate the federated learning training process through invoking the training function for all participating clients.

**Step 2 (Client):** When training is commenced, the global ML weights in S3 are downloaded to be used for local training using their own database data.

**Step 3 (Client):** After local training is completed, the clients will update the newly trained ML model into the same S3 bucket.

**Step 4 (Server):** The server will download the new ML model weights uploaded by all the clients and perform aggregation on the model to create a new global model. The result will be a new model with aggregated model weights which will be uploaded back into S3 to replace the previous global ML model.

6.2.5 Federated Learning Module Verification Test

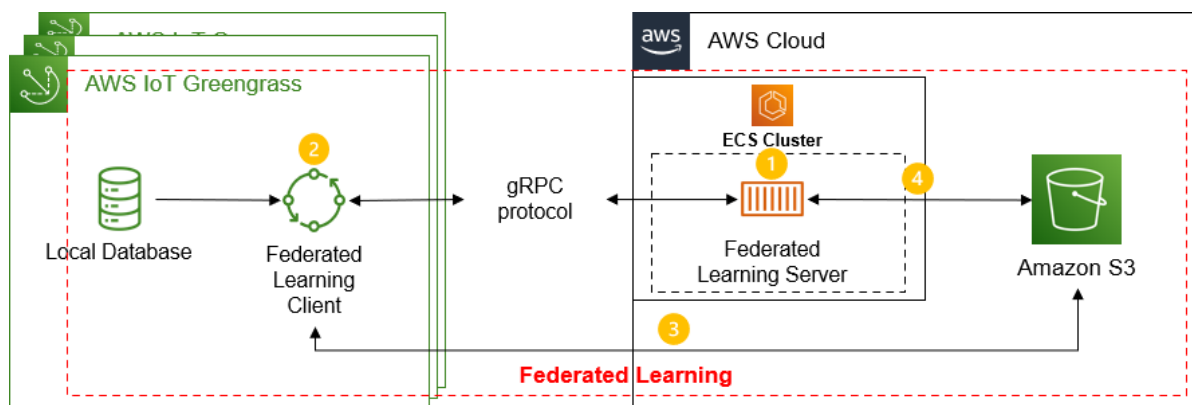
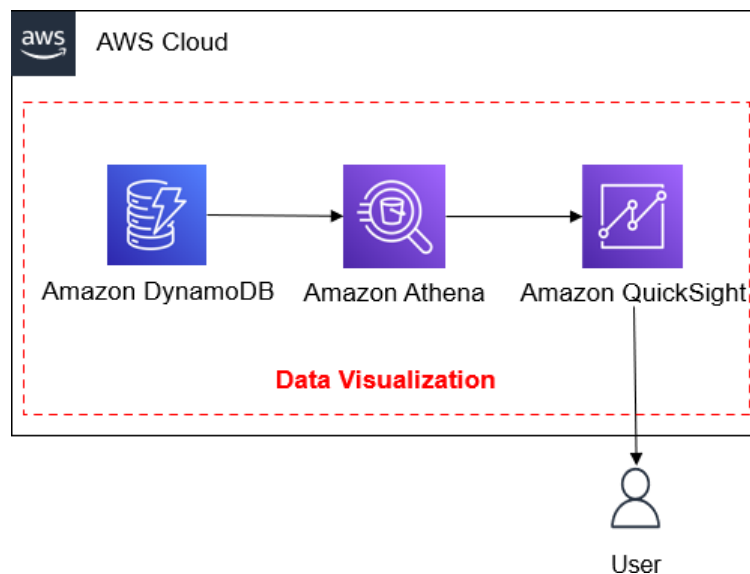


Figure 6.18: Federated Learning Module from Architecture Diagram

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

Table 6.2: Federated Learning Steps Verification Test table

## 6.3 Data Visualization



**Figure 6.19: Data Visualization Module from Architecture Diagram**

The data which has been collected from each of the individual edge environments have been processed and stored inside the specific DynamoDB table inside AWS cloud. As mentioned in the implementation section, Amazon Athena will be used to query these data from the specific DynamoDB table and turned into a data source in the suitable form to be used in Quicksight for visualization purposes. The final interactive dashboard will then be build using Quicksight and published for user to access their dashboard with data belonging to their own farm only. The design for the dashboard will follow various conditions which have been decided for preventing any redundant information presented to achieve a better data-driven decision-making process.

Design Considerations:

- Near real time information
- Prioritize Simplicity
- How is the farm doing?
- How critical is the overall situation?
- What is the health status of the farm?
- What caused breaches?
- Who caused the breaches?



## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

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

## 6.3.1 High Level Dashboard

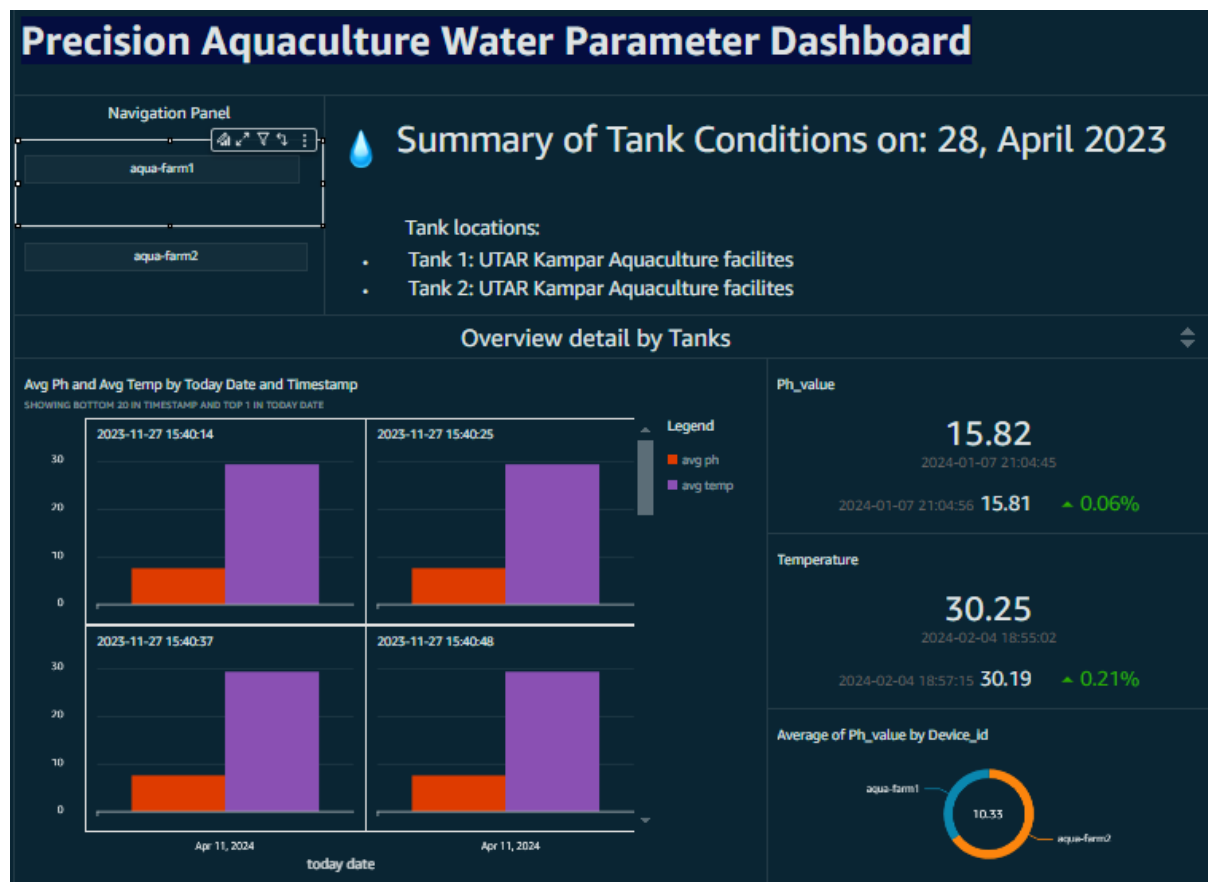


Figure 6.20: Main Page of Dashboard (High Level)

Dashboard Content:

- **Navigation Panel:** Navigate to individual tanks.
- **Date:** Inform the user of the date of the visualization.
- **Tank Locations:** Inform the user of the locations of the tanks visualized.
- **Average Water Parameters by Date and Timestamp:** Management consideration (E.g., Maintenance/Feed frequency)
- **Average of pH by Tank:** Provides proportional view (E.g., High percentage of critical values could reside in a tank)
- **Trend for pH values:** Act in advance before disasters happen.
- **Trend for water temperature:** Act in advance before disasters happen.

6.3.2 Individual Level Dashboard

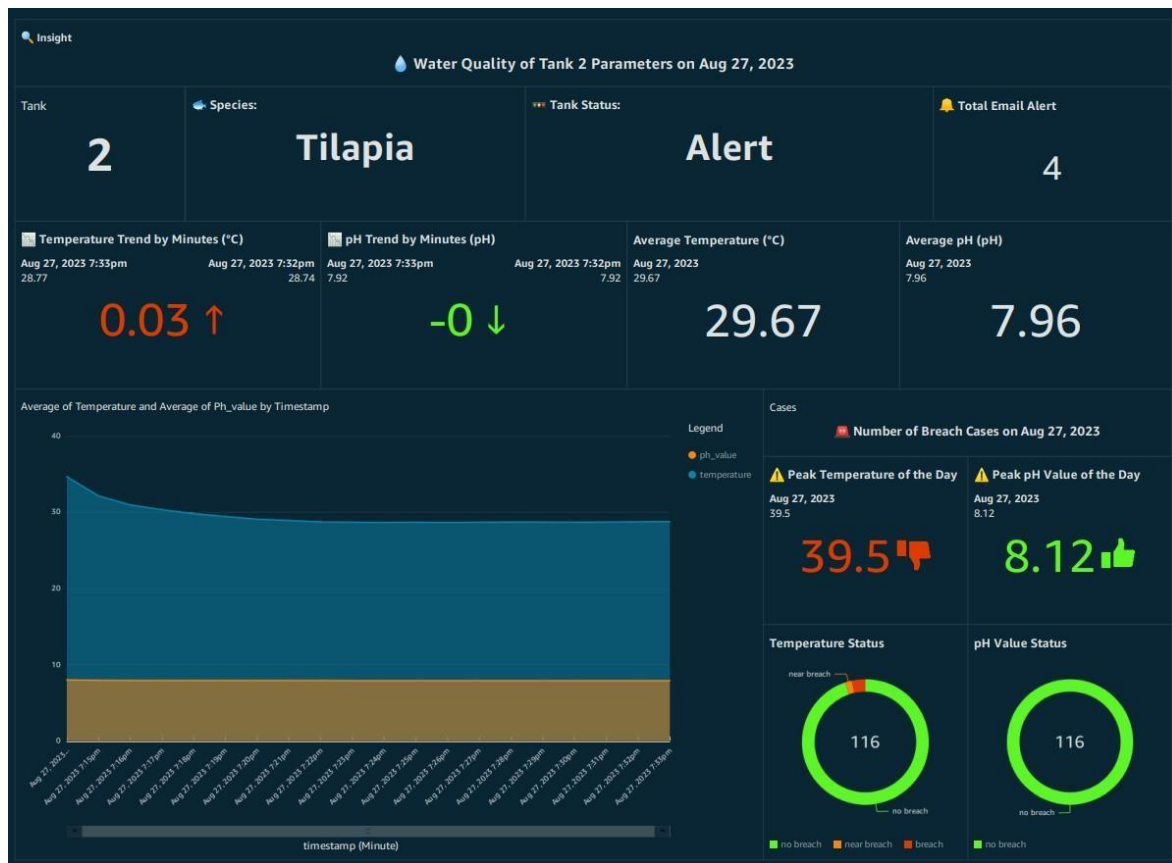


Figure 6.21: Individual Tank Detail Page of Dashboard (Individual Level)

Dashboard Content:

- **Date:** To inform the user of the date of the visualization.
- **Tank Number/ID:** Which tank’s information is being displayed.
- **Species:** Water requirements might differ for different species.
- **Health Status:** The urgency of the issue should be diagnosed or resolved.
- **Total Alert:** Number of breach cases.
- **Temperature Trend:** Stability (Inconsistent values may require attention to prevent stress on aquatic conditions)
- **pH value Trend:** Stability (Inconsistent values may require attention to prevent stress on aquatic conditions)
- **Average Temperature:** Management Considerations (E.g., Aging heater)
- **Average pH Value:** Management Considerations (E.g., Feed less to prevent water saturation)

## CHAPTER 6 SYSTEM DEPLOYMENT AND EVALUATION

- **Water Parameter Trend (Line Chart):** Derive insights from line trends (E.g., Is it reasonable if the trend for temperature to decrease from afternoon to evening?)
- **Peak Temperature:** The peak temperature can relate to certain incidents (E.g., water heater broken causes temperature to drop)
- **Peak pH value:** The peak temperature can relate to certain incidents (E.g., Contamination of water due to high values)
- **Number of Water Parameter Breaches:** Highlight farmers attention (Prioritize to solve these issues first)

# CHAPTER 7: Conclusion

In this project, the main motivation for “Developing a Federated Learning Framework for Precision Aquaculture” was to address the existing issues which are currently faced in the aquaculture industry of Malaysia. The aquaculture industry in Malaysia has yet to implement IT with various limitations such as the lack of knowledge and resources since the industry is predominantly occupied by medium to small-scaled operators. When compared against other giants of this industry such as China, the Malaysian way of relying on manual labour for daily operations and lack of decentralized data to be used for precision aquaculture operations exposes its disadvantages. The traditional way of aquaculture operations can be unreliable, costly and prone to human errors. With that being said, this project proposed a solution to provide a cost-effective, scalable and reliable method for aquaculture farmers to perform precision aquaculture whilst maintaining their data privacy in a decentralised manner through federated learning to monitor and manage operations. Since most aquaculture farms are usually located in remote areas, edge computing in a IoT setting is also implemented to ensure the availability in the absence of network connectivity which also suits the structure of federated learning. AWS Cloud was also chosen as the platform to carry out data visualization, backup and machine learning model training services for this project.

Furthermore, the system design which was chosen for this project was designed according to the objectives that required to be achieved. To ensure the feasibility and practicality of the system design, a preliminary prototype implementation was carried out. A framework has been developed based on the architecture diagram in order to validate and test out the system. Preliminary implementation was done in order to try out and test the functionalities and performance of the system under real-world conditions.

With the successful implementation of the second objective during Project 1, a strong solid foundation has been laid down for achieving the first and third objectives in Project 2. This is due to the fact that a working IoT edge environment which runs together with a cloud platform is required in order to perform the federated learning framework. The result of Project 1 were 2 sets of working IoT edge devices which were able to communicate with the cloud

## CHAPTER 7 CONCLUSION

services via MQTT protocols. Within each individual edge environment, Greengrass Core devices were registered with components for managing the client devices. These custom components functioned as the controls for performing local processing of data which have been captured. Inside AWS cloud, the captured data is passed through IoT rules after being received from a subscribed topic which was published by the Greengrass Core devices to further processing by Lambda functions and finally stored into tables inside DynamoDB.

As for Project 2, there were several important modifications which have been made towards the system architecture in order to achieve the first and third objectives as well as overcoming significant challenges faced along the way. The first objective of the project was achieved successfully by performing thorough research on the available federated learning frameworks available currently and comparing their pros and cons to decide on the most suitable framework to be used. After the specific framework has been decided, Flower federated learning documentation and tutorials online were leveraged to better understand how the entire process works. The server was hosted in the cloud with the integration of various AWS services to transform it into a service which can be accessible by all the clients. At each of the edge environments, essential functions required to run the federated learning process have also been implemented for the server to orchestrate the entire process by invoking some routines that need to be done by the clients. The whole federated learning process is executed through communication between clients and server in the form of gRPC protocols with the aid of intermediary AWS services to facilitate the entire federated learning process.

As for the third objective, an intelligent and interactive dashboard was also achieved through designing and constructing the dashboard in AWS Quicksight. The primary data source used for the dashboard was the processed data which was queried from the DynamoDB table storing them. The dashboard allows authorized users to check general information of the aquaculture farm in high level views or go into individual tanks in individual level view. This allows the visualization of near real-time data which is generated by edge environments for the users. The system was deployed onto several real aquaculture environments after the implementation phase to perform testing and evaluation. The testing criteria and environments were chosen to be as close to real world conditions as possible to mirror the performance of the system. Successful results were obtained during the testing phase which marks the success of this project.

## CHAPTER 7 CONCLUSION

As a conclusion, this project aims to address the various challenges which are faced by the Malaysian aquaculture industry by designing an IoT Cloud solution using edge computing together with federated learning framework. These technologies are still relatively new towards the industry thus making this project an opportunity for exploring and innovating deeper into this exciting topic. Federated learning has risen to become one of the trending technologies in the recent years, this project seeks to leverage its potential in the implementation of aquaculture operations. The ultimate goal of this project is to make contributions for the development and growth of the aquaculture industry in Malaysia by developing a solution which addresses the current issues and promotes a more sustainable and efficient practice.

## 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%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].
- [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].



## REFERENCES

- [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] 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/#:~:text=The%20aims%20of%20precision%20aquaculture,more%20evidence%2Dbased%20management%20decisions.> [Accessed: 13- Aug- 2022].
- [10] 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#:~:text=Edge%20is%20about%20processing%20data.> [Accessed: 11- Aug- 2022].
- [11] IBM, "What Is Edge Computing," Ibm.com, 2020. <https://www.ibm.com/cloud/what-is-edge-computing.> [Accessed: 17-Apr-2023].
- [12] "Federated learning," RapidMiner, 06-Sep-2022. [Online]. Available: <https://rapidminer.com/glossary/federated-learning/>. [Accessed: 18-Apr-2023].
- [13] 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.](https://www.researchgate.net/publication/233118805_The_use_of_information_technology_in_aquaculture_management.) [Accessed: 15- Aug- 2022].

## REFERENCES

- [14] 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].
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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\\_Control\\_for\\_Aquaculture\\_Based\\_on\\_Wireless\\_Sensor\\_Networks](https://www.researchgate.net/publication/262380185_Water_Quality_Monitoring_and_Control_for_Aquaculture_Based_on_Wireless_Sensor_Networks). [Accessed: 20-Aug- 2022].

## REFERENCES

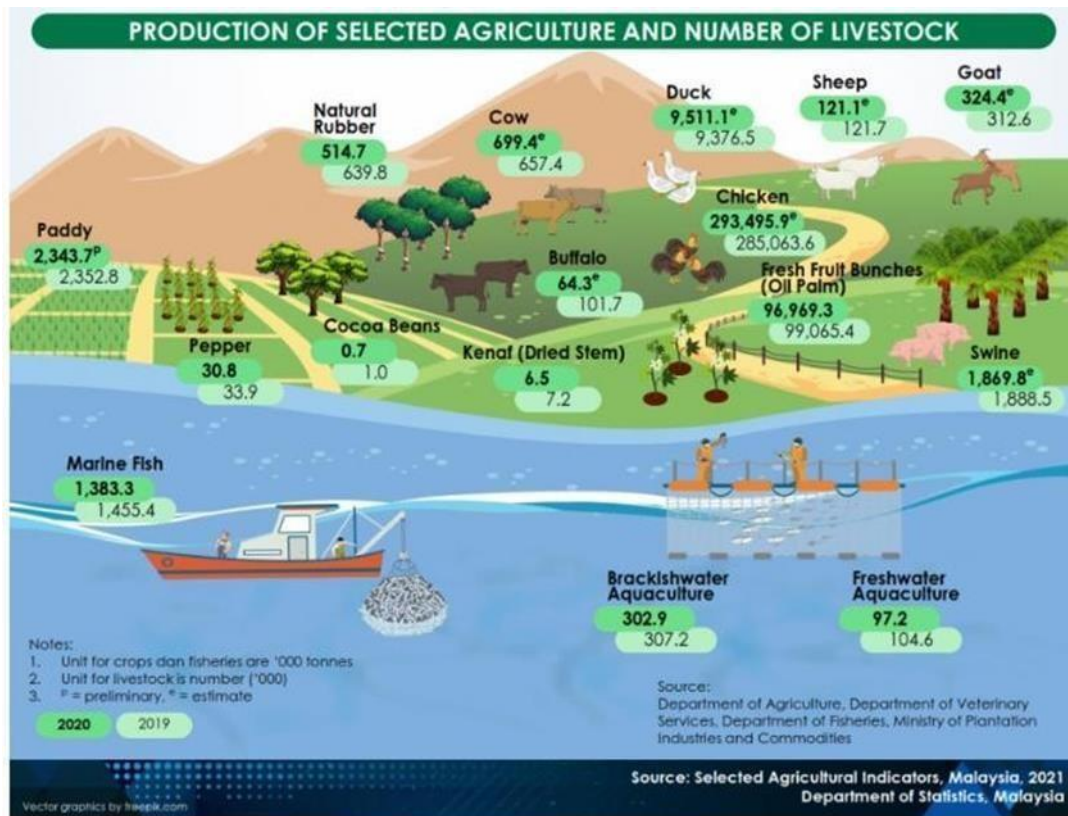
- [20] R. R. Teixeira, J. B. Puccinelli, L. Poersch, M. R. Pias, and V. M. Oliveira, "Towards precision aquaculture: A high performance, cost-effective IOT ..." [Online]. Available: [https://www.researchgate.net/publication/351869126\\_Towards\\_Precision\\_Aquaculture\\_A\\_High\\_Performance\\_Cost-effective\\_IoT\\_approach](https://www.researchgate.net/publication/351869126_Towards_Precision_Aquaculture_A_High_Performance_Cost-effective_IoT_approach). [Accessed: 15-Apr-2023].
- [21] F. O'Donncha and J. Grant, "precision aquaculture - researchgate." [Online]. Available: [https://www.researchgate.net/publication/339059062\\_Precision\\_Aquaculture](https://www.researchgate.net/publication/339059062_Precision_Aquaculture). [Accessed: 15-Apr-2023].
- [22] S. Zhao *et al.*, "Application of machine learning in intelligent fish aquaculture: A review," *Aquaculture*, vol. 540, p. 736724, Jul. 2021, doi: <https://doi.org/10.1016/j.aquaculture.2021.736724>.
- [23] A. Chatziantoniou, N. Papandroulakis, O. Stavrakidis-Zachou, S. Spondylidis, S. Taskaris, and K. Topouzelis, "Aquasafe: A Remote Sensing, Web-Based Platform for the Support of Precision Fish Farming," *Applied Sciences*, vol. 13, no. 10, p. 6122, Jan. 2023, doi: <https://doi.org/10.3390/app13106122>
- [24] V. K. Papanikolaou, S. A. Tegos, P. S. Bouzinis, D. Tyrovolas, P. D. Diamantoulakis, and G. K. Karagiannidis, "ATLAS: Internet of Things Platform for Precision Aquaculture," *2022 Panhellenic Conference on Electronics & Telecommunications (PACET)*, Dec. 2022, doi: <https://doi.org/10.1109/pacet56979.2022.9976375>
- [25] S. Karim, I. Hussain, A. Hussain, K. Hassan, and S. Iqbal, "IoT Based Smart Fish Farming Aquaculture Monitoring System," *International Journal on Emerging Technologies*, vol. 12, no. 2, pp. 45–53, 2021, Available: <https://www.researchtrend.net/ijet/pdf/8%20IoT%20Based%20Smart%20Fish%20Farming%20Aquaculture%20Monitoring%20System%20Sohail%20Karim%203461.pdf>

## REFERENCES

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

# APPENDIX

## Appendix A



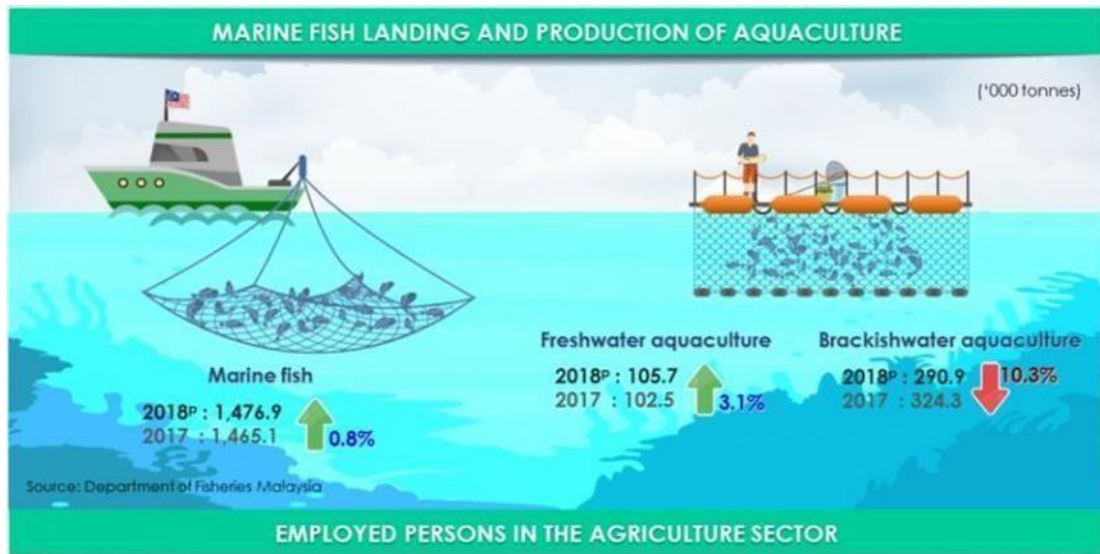
Production of selected agriculture sector 2019 (Department of Statistics Malaysia)

Source:

[https://www.dosm.gov.my/v1/index.php/index.php?r=column/cthemByCat&cat=72&bul\\_id=TDV1YU4yc1Z0dUVyZ0xPV0ptRlhWQT09&menu\\_id=Z0VTZGU1UHBUT1VJMF1paX\\_RRR0xpdz09](https://www.dosm.gov.my/v1/index.php/index.php?r=column/cthemByCat&cat=72&bul_id=TDV1YU4yc1Z0dUVyZ0xPV0ptRlhWQT09&menu_id=Z0VTZGU1UHBUT1VJMF1paX_RRR0xpdz09)

## APPENDIX

### Appendix B

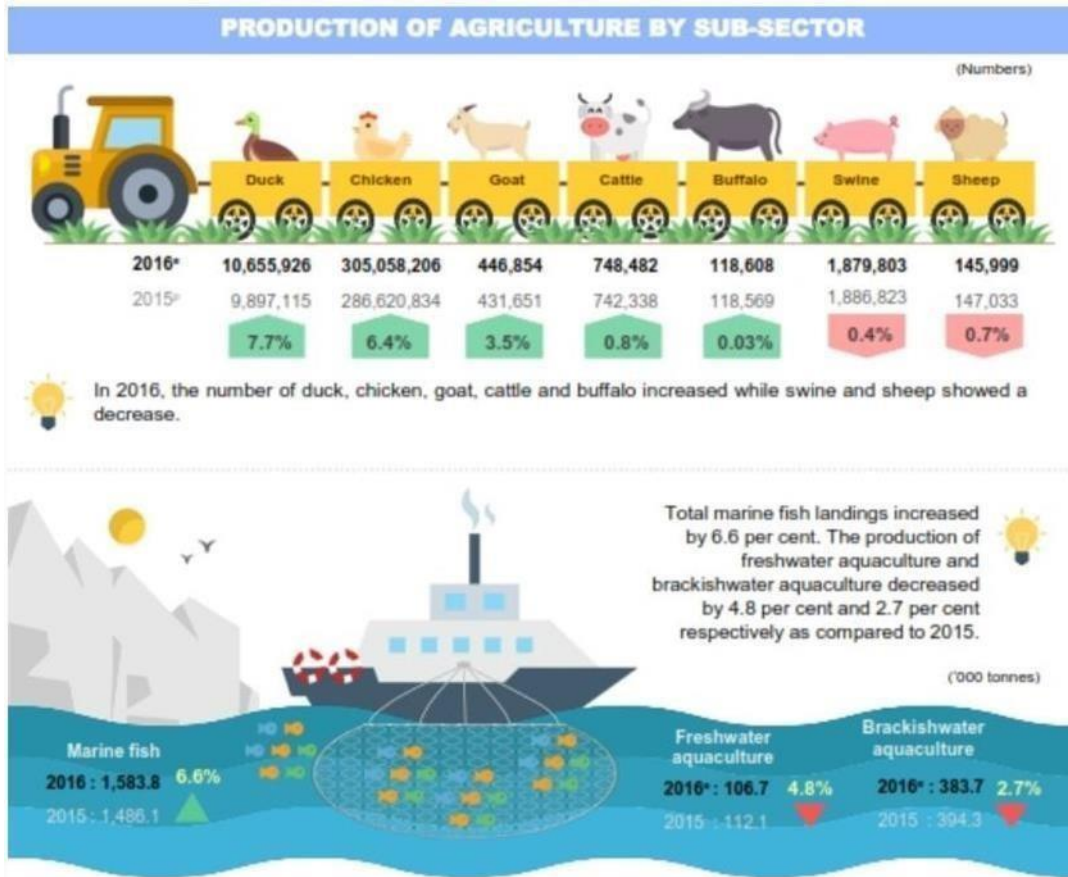


Production of selected agriculture sector 2018 (Department of Statistic Malaysia)

Source:

[https://www.dosm.gov.my/v1/index.php/index.php?r=column/cthemByCat&cat=72&bul\\_id=SEUxMEE3VFdBcDJhdUhPZVUxa2pKdz09&menu\\_id=Z0VTZGU1UHBUT1VJMF1paXRRR0xpdz09](https://www.dosm.gov.my/v1/index.php/index.php?r=column/cthemByCat&cat=72&bul_id=SEUxMEE3VFdBcDJhdUhPZVUxa2pKdz09&menu_id=Z0VTZGU1UHBUT1VJMF1paXRRR0xpdz09)

Appendix C



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

Source:

[https://www.dosm.gov.my/v1/index.php?r=column/cthemeByCat&cat=72&bul\\_id=MDNYUitINmRKcENRY2FvMmR5TWdGdz09&menu\\_id=Z0VTZGU1UHBUT1VJMF1paXRRR0pdz09#:~: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/cthemeByCat&cat=72&bul_id=MDNYUitINmRKcENRY2FvMmR5TWdGdz09&menu_id=Z0VTZGU1UHBUT1VJMF1paXRRR0pdz09#:~: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)

<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 4</b>
<b>Student Name &amp; ID: John Ling Tze Jun 21ACB02602</b>	
<b>Supervisor: Dr Cheng Wai Khuen</b>	
<b>Project Title: To Develop a Federated Learning Framework for Precision Aquaculture</b>	

### 1. WORK DONE

Completed OS configurations for second edge device.

### 2. WORK TO BE DONE

Configure Greengrass Core device software and components on second edge device.

### 3. PROBLEMS ENCOUNTERED

-

### 4. SELF EVALUATION OF THE PROGRESS

On Track



Supervisor's signature



Student's signature



## FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 6</b>
<b>Student Name &amp; ID: John Ling Tze Jun 21ACB02602</b>	
<b>Supervisor: Dr Cheng Wai Khuen</b>	
<b>Project Title: To Develop a Federated Learning Framework for Precision Aquaculture</b>	

### 1. WORK DONE

Successfully registered second Greengrass Core device and deployed custom components.

### 2. WORK TO BE DONE

Do testing on second edge device and study up on 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)

<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 8</b>
<b>Student Name &amp; ID: John Ling Tze Jun 21ACB02602</b>	
<b>Supervisor: Dr Cheng Wai Khuen</b>	
<b>Project Title: To Develop a Federated Learning Framework for Precision Aquaculture</b>	

### 1. WORK DONE

Finish writing federated learning server and client codes.

### 2. WORK TO BE DONE

Deploy federated learning component on both devices and finish up remaining AWS service configurations.

### 3. PROBLEMS ENCOUNTERED

Some AWS configurations are not functioning as intended.

### 4. SELF EVALUATION OF THE PROGRESS

On Track



Supervisor's signature



Student's signature

## FINAL YEAR PROJECT WEEKLY REPORT

*(Project II)*

<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 10</b>
<b>Student Name &amp; ID: John Ling Tze Jun 21ACB02602</b>	
<b>Supervisor: Dr Cheng Wai Khuen</b>	
<b>Project Title: To Develop a Federated Learning Framework for Precision Aquaculture</b>	

### 1. WORK DONE

Successfully deployed federated learning components and cloud server configured.

### 2. WORK TO BE DONE

Test out federated learning framework.

### 3. PROBLEMS ENCOUNTERED

-

### 4. SELF EVALUATION OF THE PROGRESS

Good Progress



Supervisor's signature



Student's signature

## FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 11</b>
<b>Student Name &amp; ID: John Ling Tze Jun 21ACB02602</b>	
<b>Supervisor: Dr Cheng Wai Khuen</b>	
<b>Project Title: To Develop a Federated Learning Framework for Precision Aquaculture</b>	

### 1. WORK DONE

Both edge devices can perform federated learning framework as intended with cloud server.

### 2. WORK TO BE DONE

Design and build interactive 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)*

<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 12</b>
<b>Student Name &amp; ID: John Ling Tze Jun 21ACB02602</b>	
<b>Supervisor: Dr Cheng Wai Khuen</b>	
<b>Project Title: To Develop a Federated Learning Framework for Precision Aquaculture</b>	

### 1. WORK DONE

Finished interactive dashboard. Also conducted final physical testing and troubleshooting for edge devices in aquafarm.

### 2. WORK TO BE DONE


Finish FYP report and prepare for presentation.

### 3. PROBLEMS ENCOUNTERED

-

### 4. SELF EVALUATION OF THE PROGRESS

On Track



Supervisor's signature



Student's signature

POSTER

**By John Ling Tze Jun**

# Federated Learning Framework for Precision Aquaculture

## Problem Statements

1. Aquaculture cultivation can cause stress and disease susceptibility in fish population.
2. Manual water sampling require more time and is prone to human error.
3. Precision aquaculture requires diverse and immense training data.
4. Traditional centralized machine learning models does not ensure data privacy.
5. Network connectivity of aquaculture farms in rural areas can be challenging.

## Objectives

1. To **develop a federated learning model** that is suitable for precision aquaculture.
2. To **compare various approaches** in enabling federated learning within an **IoT-Cloud architecture**.
3. To **assess the performances** on the proposed federated learning framework.

## Project Highlights

EDGE COMPUTING	FEDERATED LEARNING
<ul style="list-style-type: none"><li>• LOWER LATENCY AND IMPROVE RESPONSE TIME</li><li>• REDUCE BANDWIDTH CONSUMPTION</li><li>• LESS RELIANCE ON NETWORK CONNECTION</li><li>• REAL-TIME DECISION MAKING</li></ul>	<ul style="list-style-type: none"><li>• ENSURE DATA PRIVACY</li><li>• DIVERSE TRAINING DATA</li><li>• DECENTRALIZE APPROACH</li><li>• MULTIPLE EDGE CONTRIBUTE TO TRAINING MACHINE LEARNING MODEL</li></ul>

## PLAGIARISM CHECK RESULT

2102602\_FYP2

ORIGINALITY REPORT

<b>7</b> %	<b>4</b> %	<b>5</b> %	<b>2</b> %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

<b>1</b>	Shili Zhao, Song Zhang, Jincun Liu, He Wang, Jia Zhu, Daoliang Li, Ran Zhao. "Application of machine learning in intelligent fish aquaculture: A review", Aquaculture, 2021 Publication	<b>2</b> %
<b>2</b>	<a href="http://www.mdpi.com">www.mdpi.com</a> Internet Source	<b>1</b> %
<b>3</b>	<a href="http://www.comsoc.org">www.comsoc.org</a> Internet Source	<b>1</b> %
<b>4</b>	Vasilis K. Papanikolaou, Sotiris A. Tegos, Pavlos S. Bouzinis, Dimitrios Tyrovolas et al. "ATLAS: Internet of Things Platform for Precision Aquaculture", 2022 Panhellenic Conference on Electronics & Telecommunications (PACET), 2022 Publication	<b>&lt;1</b> %
<b>5</b>	<a href="http://eprints.utar.edu.my">eprints.utar.edu.my</a> Internet Source	<b>&lt;1</b> %
<b>6</b>	<a href="http://docs.aws.amazon.com">docs.aws.amazon.com</a> Internet Source	<b>&lt;1</b> %

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



**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	John Ling Tze Jun
<b>ID Number(s)</b>	21ACB02602
<b>Programme / Course</b>	BACHELOR OF COMPUTER SCIENCE (HONOURS)
<b>Title of Final Year Project</b>	To Develop a Federated Learning Framework for Precision Aquaculture

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)</b>
<b>Overall similarity index: <u>  7  </u> %</b>  <b>Similarity by source</b> Internet Sources: <u>  4  </u> % Publications: <u>  5  </u> % Student Papers: <u>  2  </u> %	<b>OK</b>
<b>Number of individual sources listed of more than 3% similarity: <u>  0  </u></b>	
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> (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

Name: Ts. Dr. Cheng Wai Khuen

Date: 12/4/2024

Signature of Co-Supervisor

Name: \_\_\_\_\_

Date: \_\_\_\_\_





**UNIVERSITI TUNKU ABDUL RAHMAN**  
**FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY**  
**(KAMPAR CAMPUS)**

**CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student Id	21ACB02602
Student Name	John Ling Tze Jun
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: 12/4/2024