UNIVERSITI UTARA MALAYSIA

PRACTICUM REPORT

SEM 2 2019/2020 (A192)

AQUAPONIC MONITORING SYSTEM

(AquMoSys)

BY:

GHIFARI AHMAD ZAKI

SCHOOL OF COMPUTING
COLLEGE OF ARTS AND SCIENCES

AQUAPONIC MONITORING SYSTEM (AquMoSys) BASED ON IoT (Internet of Things)

MCMC-UUM

MAKERSPACE LABORATORY

SCHOOL OF COMPUTING, COLLEGE OF ARTS & SCIENCES

UNIVERSITI UTARA MALAYSIA, SINTOK KEDAH 06010

This report is prepared to fulfil the requirement of

STIX3912 Practicum

By:

AHMAD ZAKI GHIFARI

243809

ACKNOWLEDGEMENT

Alhamdulillahi rabbil ‘aalamiin, give thanks and praise to Allah Subhanahu Wa Ta'ala for giving me the opportunity and strength to complete the internship at MCMC-UUM Makerspace Lab, School of Computing, College of Arts and Sciences, University Utara Malaysia. I also want to express my gratitude and affection to my parents in Indonesia who have provided a lot of advice and support in completing internship for 6 months. I would like to thank the University Utara Malaysia, 3.5 years I have studied here, a lot of knowledge and experience that I have gained so that I can complete this internship. Thank you very much to Dr. Mustafa Ali Abuzaraida, as my supervisor at UUM, Mr. Suwannit Chareen and Mr. Ahmad Hanis Mohd Shabli for allowing me to take part in the internship at MCMC-UUM MakerSpace Lab and giving me lots of valuable knowledge, experience and contributions. I apologize for all the mistakes and shortcomings while undergoing internship here. Finally, to all those who have directly or indirectly assisted and helped me in undergoing this internship. May Allah Subhanahu Wa Ta'ala repay all your kindness and sincerity with the best kind of reward.

SCHOOL OF COMPUTING COLLEGE OF

ARTS AND SCIENCES

FEBRUARY 2020

DISCLAIMER

STIX3912 PRACTICUM

I am responsible for the accuracy of all opinion, technical comment, factual report, data, figures, illustrations, and photographs highlighted in this report. I bear full responsibility that the report submitted has been reviewed and subject to copyright ownership rights. University Utara Malaysia will not bear any liability for the accuracy of any comment, report, and other technical and information, and the copyright or ownership right claims.

AHMAD ZAKI GHIFARI

243809

ABSTRACT

Computing that surrounds us today is a far cry from its humble beginnings where one personal computer was shared by everybody at home. Nowadays, computers are more than just tools, they are ubiquitous assistants in our day-to-day lives. Over the last decade, the range of Internet-connected devices has grown significantly, fueled in part by the exponential growth of applications and new cloud-based services. Put together, these trends are leading to a future marked by what some call the Internet of Things (IoT): a world where millions of people, processes, and devices (from fridges and cars to heart rate monitors, energy tools, and traffic monitors) are connected in ways where the sum is greater and smarter than the individual parts. Aquaponics is a system where plants and fish grow together. By combining the two systems, recycling occurs so that the waste from the aquaculture system is the input of the hydroponic system. In aquaponics, it is essential to continuously monitor the environment to ensure optimal growth of the fish and the plant. AquMoSys is Aquaponic Monitoring System consisting of 4 (four) sensors connected with ESP32 microcontroller namely Temperature sensor, Turbidity sensor, Humidity sensor and pH sensor as well as mobile applications based on Flutter, Firebase Realtime Database and Firebase Cloud Firestore. Firstly, after the all the four (4) sensors connected to ESP32 and it also connected to WiFi connection which was set up when the program script was uploaded. It has four main features, which are Animated Realtime Graph, Push Notification, Email Alert, and we can manually set the threshold for each of the sensors. Evolutionary prototyping model (also known as breadboard) will be used with other necessary steps added to it. This system uses evolutionary prototyping because this type of prototyping is a model that allows the developer to continuously refine the system as other requirements are getting clearer. Visual Studio Code and Arduino IDE was used to write the code. All the components were connected and soldered in the stripboard.

Keyword: Aquaponic, IoT, AquMoSys, Fish, Plant, Flutter.

TABLE OF CONTENTS

TITLE PAGE..i
DECLARATION..ii
ACKNOWLODGMENT……………………………………………………iii
DISCLAIMER…………………………………………………… iv
ABSTRACT……………………………………………………v
LIST OF FIGURES……………………………………………...vii-viii
LIST OF TABLES……………………………………………viii
ABBREVIATION……………………………………………...ix
CHAPTER 1: INTRODUCTION.. ..1
1.1 Organization Background ....……1
1.2 Vision……………………………………………………3
1.3 Mission…………………………………………………….3
1.4 Objective……………………………………………………3-4
1.5 Summary.. 4
CHAPTER 2: PROJECT DESCRIPTION.. 5
2.1 Project Background 5
2.2 Problem Statement.. 6
2.3 Project Objectives.. 6
2.4 Project Scope 6-7
2.5 Significance of Project.. 8
2.6 Weakness and Recommendation.. 8
2.7 Summary.. 9
CHAPTER 3: METHODOLOGY.. 10
3.1 Overall Phases 10
3.2 Description of Each Phase 10-11
3.3 Summary.. 12
CHAPTER 4: CONCLUSION.. 13
REFERENCES………………………………………..………...………..14-15
APPENDICES……………………………………………………16-47

LIST OF FIGURES

Figure 1: School of Computing Logo 1

Figure 2: MCMC-UUM Makerspace Lab Logo 1

Figure 3: School of Computing Organization Chart 1

Figure 4: AquMoSyS Logo 5

Figure 5: Workflow of Evolutionary Prototyping Logo 10

Figure 6: Gantt Chart of AquMoSys 16

Figure 7: Scheduling & Deliverables of AquMoSys 16

Figure 8: Nodemcu 32S-ESP32 17

Figure 9: Humidity sensor (DHT11) 17

Figure 10: DS18B20 waterproof version 18

Figure 11: Turbidity sensor by DF Robot 18

Figure 12: pH sensor V.2 by DF Robot 19

Figure 13: Universal Mobile Charger (Micro USB) 19

Figure 14: Schematic of all connected components 20

Figure 15: Use Case diagram of AquMoSys 20

Figure 16: Splashscreen page of AquMoSys 23

Figure 17: Dashboard page of AquMoSys 23

Figure 18: Drawer page of AquMoSys 24

Figure 19: Sign In page of AquMoSys 24

Figure 20: Threshold Setting page of AquMoSys 25

Figure 21: Notification page of AquMoSys 25

Figure 22: About page of AquMoSys 26

Figure 23: Realtime graph and 24 hours data history page of Temperature sensor 26

Figure 24: Realtime graph and 24 hours data history page of Turbidity sensor 27

Figure 25: Realtime graph and 24 hours data history page of pH sensor 27

Figure 26: Realtime graph and 24 hours data history page of Humidity sensor.... …………………………...28

Figure 27: Email alert & Push Notification page of AquMoSys .... …………………………...28

Figure 28: Turbidity, DS18B20 waterproof, and pH sensors................................ …………………………...29

Figure 29: Nodemcu 32S-ESP32 soldered to the stripboard. DHT11, Turbidity, pH and Temp sensor…...29

Figure 30: Aquaponic Ecosystem was created using plastic pipe and big plastic drum……………..……...29

LIST OF TABLES

Table 1: Functional Requirement's List...…………………………..20

Table 2: Non-Functional Requirement's List…………………………………………………...……21

ABBREVIATION

SOC	School of Computing
MCMC	Malaysia Communications and Multimedia Commissions
UUM	University Utara Malaysia
SDK	Software Development Kit
IoT	Internet of Things
AquMoSys	Aquaponic Monitoring System
WiFi	Wireless Fidelity
IDE	Integrated Development Environment
UI	User Interface
PCB	Printed Circuit Board
USB	Universal Serial Bus
E-Mail	Electronic Mail
IR 4.0	Industrial Revolution 4.0
NTU	Nephelometric Turbidity Unit

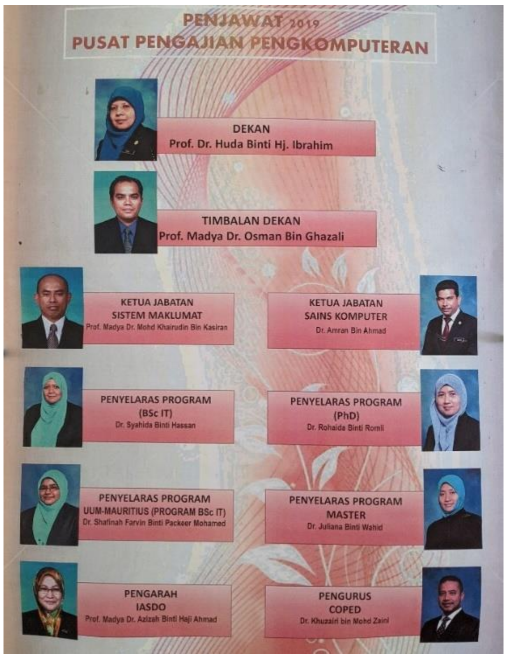
1.0 INTRODUCTION

This chapter presents an overview of the School of Computing, as well as the UUM-MCMC MakerSpace Lab where I undergo my internship since February – August 2020. This chapter is organized as follow, which is in section 1.1 describes the organization background which consists of the organization's history and organization chart. In section 1.2, discusses about organization's vision, section 1.3 discusses about organization's mission, section 1.4 discusses about organization's objectives and in section 1.5 is summarization of the chapter.

1.1 Organization Background

Figure 1: School of Computing Logo

Figure 2: MCMC-UUM Makerspace Lab Logo


PENJAWAT 2019
PUSAT PENGAJIAN PENGKOMPUTERAN

DEKAN
Prof. Dr. Huda Binti Hj. Ibrahim

TIMBALAN DEKAN
Prof. Madya Dr. Osman Bin Ghazali

KETUA JABATAN
SISTEM MAKLUMAT
Prof. Madya Dr. Mohd Khairudin Bin Kasiran

KETUA JABATAN
SAINS KOMPUTER
Dr. Amran Bin Ahmad

PENYELARAS PROGRAM
(BSc IT)
Dr. Syahida Binti Hassan

PENYELARAS PROGRAM
(PhD)
Dr. Rohaida Binti Romli

PENYELARAS PROGRAM
UUM-MAURITIUS (PROGRAM BSc IT)
Dr. Shafinah Farvin Binti Packeer Mohamed

PENYELARAS PROGRAM
MASTER
Dr. Juliana Binti Wahid

PENGARAH
IASDO
Prof. Madya Dr. Azizah Binti Haji Ahmad

PENGURUS
COPED
Dr. Khuzairi bin Mohd Zaini

Figure 3: School of Computing Organization Chart

School of Computing (SOC) was first established on March 1, 1990 as School of Information Technology. In July 2011, the School of Information Technology was renamed as School of Computing. The main aim of the school is to provide a conducive environment for the acquisition and dissemination of computing knowledge and skills.

Computing that surrounds us today is a far cry from its humble beginnings where one personal computer was shared by everybody at home. Nowadays, computers are more than just tools, they are ubiquitous assistants in our day-to-day lives. Over the last decade, the range of Internet-connected devices has grown significantly, fueled in part by the exponential growth of applications and new cloud-based services.

Put together, these trends are leading to a future marked by what some call the Internet of Things (IoT): a world where millions of people, processes, and devices (from fridges and cars to heart rate monitors, energy tools, and traffic monitors) are connected in ways where the sum is greater and smarter than the individual parts. The power and potential of all these connected and smart technologies offers us the freedom and flexibility to do things when, how and where we want.

IoT is regarded as one of the main backbones of Industrial Revolution 4.0 (IR 4.0). It has been predicted by McKinsey Global Institute that it will generate up to $11T in value to the global economy by 2025. The IoT is about much more than just smart refrigerators or lifestyle enhancement. It also includes a corporate side, enabling organizations to collect and analyze data from sensors on manufacturing equipment, pipelines, weather stations, smart meters, delivery trucks and other types of machinery.

There are many players in Malaysia who have started their IoT initiative, either in the academic or industry players. Majority of them are focusing on the smart gadgets, especially in terms of creating, packaging, optimizing, producing in masses, power saving, long-life battery, better sensor reading and communication, etc. Although all the initiatives are very important in creating a robust and complete ecosystem in the IoT roadmap, we are still very much lacking in managing and analyzing IoT data in business applications (industrial analytics) and other important areas. It is imperative for businesses to harness the data, analyze it for data-driven insights, and find the business in their data to achieve these better outcomes.

UUM has possess a very strong record in the field of business data analytics and big data. It can be offered to fill in this gap in bringing Malaysia towards IR4.0. Getting there involves establishing a data supply chain to properly and quickly mobilizing data for consumption, developing a robust analytical platform - enabling data to flow easily, quickly and usefully through the entire organization – and eventually throughout each company's ecosystem of partners – companies are on their way to realizing the true value hidden in data.

Next, to reap the full benefits of the IoT, it is important to develop a robust analytical platform that brings together the capabilities around sensor-driven computing, industrial analytics, and intelligent machine applications. A myriad of analytics tools can sift through and process data once they have been programmed with the parameters. Whatever the tools used, the ideal goal is to put a system in place that can automate the analytics process of collecting and processing raw data, which can then be used to deliver business insights.

In realizing the importance and need of digital-enabled solutions in today's world, School of Computing, Universiti Utara Malaysia has proposed the MCMC-UUM My Makerspace Laboratory to encourage research and publication in related fields. The vision of the laboratory is to act as the innovation hub in the emerging technology of Internet of Things through democratization of innovation, realization of prototype, and products before deployment of the IoT devices [1].

1.2 Vision

To be acknowledged as a center of excellence and main reference in Enterprise Computing [2].

1.3 Mission

To actualize the aspiration of SOC as the center of excellence and main reference in (i) teaching and learning, (ii) research and innovation, (iii) publication and consultation, and (iv) professional training in the Enterprise Computing related areas [3].

1.4 Objectives

1.4.1 School of Computing (SOC)

a) To provide opportunities for undergraduates, masters and doctoral studies in computing and other relevant disciplines [4]

b) To expand the knowledge of national development and assist it through the research and consultation activities [5].

c) To produce graduates with sufficient integrated knowledge encompassing the program offered by the School of Computing [6].

d) To produce graduates who will become experts in their career choice [7].

e) To develop a creative, innovative, and wider understanding of computing theory and practice, in line with the globalization and modernization of Computing [8].

f) To achieve an economic excellence and development with the improvement of quality and skills in computing thus providing expertise and knowledge workers for public or private sectors [9].

g) To contribute towards the enhancement of knowledge and social development by producing intellectuals and professionals [10].

h) To establish professional relationship and collaboration with the private and public sectors towards developing expertise in the relevant fields of knowledge [11].

1.4.2 MCMC-UUM Makerspace Lab

a) To create innovative applications and domain capability across verticals for country’s needs such as Smart Community, Smart Business, Smart Agriculture, etc [12].

b) To build industry-capable talent, start-up community and entrepreneurial ecosystem for IoT [13].

c) To provide an ecosystem for innovation to thrive and embrace entrepreneurship [14].

d) To energize research mind-set and reduce costs in research and development by providing neutral and interoperable, multi-technology stack laboratory facilities [15].

e) To reduce import dependency on IoT components and promote indigenization [16].

f) To provide environment for product creation, testing and also for validation & incubation [17].

1.5 Summary

This chapter describes about the overall background of School of Computing & MCMC-UUM Makerspace Lab, which includes Organization Background, Vision, Mission, and Objectives. The next chapter (Chapter 2), will be discussing about Project Descriptions which consists of Project Background, Problem Statement, Project Objective, Project Scope, Significance of Project, and Weaknesses & Recommendation.

2.0 PROJECT DESCRIPTION

In this chapter, the overall of project description is explained. Every sub-chapters are described in detail and clear explanation. In section 2.1 describes about Project Background, section 2.2 explains about Problem Statement, section 2.3 discusses about Project Objectives, section 2.4 discusses Project Scope, section 2.5 explains about Project Significance and lastly, section 2.6 describes the Weakness and Recommendation.

2.1 Project Background

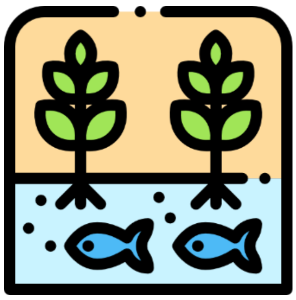


Figure 4: AquMoSys's Logo

Aquaponics is a system of aquaculture (aquaculture) and plants (hydroponics) together in a recirculated ecosystem or benefit that uses natural bacteria to convert feces & fish food residues into plant nutrients [18]. In other words, aquaponics is a system where plants and fish grow together. By combining the two systems, recycling occurs so that the waste from the aquaculture system is the input of the hydroponic system. By combining aquaculture into aquaponics, wastes discharged into nature are minimal [19]. So, arguably the aquaponics system is an environmentally friendly system [20].

Internet of Things (IoT) is a computing concept about everyday objects that are connected to the internet and can identify themselves to other devices. A concept where an object can transfer data through a network without requiring human-to-human or human-computer interaction [21].

AquMoSys is Aquaponic Monitoring System consisting of 4 (four) sensors connected with ESP32 microcontroller namely Temperature sensor, Turbidity sensor, Humidity sensor and pH sensor as well as mobile applications based on Flutter, Firebase Realtime Database and Firebase Cloud Firestore. Firstly, after the all the four (4) sensors connected to ESP32 and it also connected to WiFi connection which has been set up when the program script was uploaded. It has four main features, which are Animated Realtime Graph, Push Notification, Email Alert, and we can manually set the threshold for each of the sensors.

2.2 Problem Statement

In aquaponics, it is essential to continuously monitor the environment to ensure optimal growth of the fish and the plant, by paying attention to pH of the water, turbidity of the water, water temperature and humidity of the surrounding air.
This can take a lot of time and we need to find alternatives to be able to monitor the aquaponics system practically without disrupting our other activities wherever and whenever we want. Therefore, on the basis of this problem AquMoSys was finally created.

2.3 Project Objectives

This project aims to:

a) To identify the requirements of Aquaponic monitoring system.
b) To design and develop Aquaponic monitoring system.
c) To implement Firebase Realtime Database into Aquaponic monitoring system.

2.4 Project Scope

Aquaponic monitoring system will be developed for those who needs to continuously, and remotely monitor their aquaponic system in real time update through their own smartphones.

2.4.1 Target User

AquMoSys is aimed at anyone who has an aquaponics system and wants convenience and practicality in its management and maintenance.

2.4.2 Functionality

a) Users can view the real-time update of the sensors reading for every 10 minutes interval, it displayed by using animated graph. They also can see all the data recorded in the last 24 hours and all recorded data in previous days.

b) AquMoSys has 2 (two) types of notification namely Push notification and E-mail alert. When users launch AquMoSys app, by default push notification will be automatically turn on, but they also can turn it off through Notification menu. To use E-mail alert, users are required to input/login by using their E-mail address and automatically E-mail alert option will be available in Notification menu. Same with push notification, users also can turn the E-mail alert off.

c) Users can manually set each threshold value from each sensor reading according to the value they want. After a certain threshold value has been set, Push notification and E-mail alert will automatically notify the users if the sensors reading is above or below the threshold value.

2.4.3 Tools

a) AquMoSys is mobile-based application which is Android platform. It was developed using Flutter, open-source UI software development kit created by Google.

b) Firebase Realtime Database was chosen to store all recorded sensors reading. Every 10 minutes interval, all 4 (four) connected sensors through ESP32 microcontroller will retrieve and send the data to Firebase Realtime Database while to store the E-mail address used by the users, Firebase Cloud Firestore was opted.

c) To program ESP32 microcontroller and all four (4) connected sensors; Turbidity sensor, Temperature sensor, Humidity sensor, and pH sensor, Arduino IDE (Integrated Development Environment) was used to write the program script and upload it to ESP32 by using 115200 baud rates.

d) Fritzing software was used to create a schematic of all components so that it can be used for our reference when soldering them into stripboard.

e) To assemble all the components that mentioned above, a stripboard was used then soldered all those components based on the schematic which has been created before.

f) To power the ESP32 microcontroller, standard charger/mobile charger which has micro-USB port is used.

2.4.4 Duration of Project

The total duration of the project is 6 months, during the practical period. For detail scheduling and deliverables, a Gantt chart will be attached in Appendix B.

2.5 Project Significance

Aquaponic monitoring system will be able to monitor the water temperature, humidity, and pH level to ensure optimal growth of fish and plant. Using AquMoSys application, the condition of our aquaponic can always be monitored easily, just through a smartphone and can still do other activities without having to check the condition every minute.
Using ESP32 system on a chip microcontroller with integrated WIFI, it is connected to four types of sensors which are Turbidity sensor (NTU), Waterproof Temperature sensor (°C), Humidity Sensor (%), and pH sensor. Android was chosen as the platform of this mobile application and Flutter as open-source UI software development kit created by Google. To store all the data records of sensor's reading, Firebase Realtime Database is used. Firebase Cloud Firestore is also used to store the email records when users have signed in to get email alert. Animated real time graph is used to display all the data records in real time update for every 10 minutes interval.

2.6 Weakness and Recommendation

2.6.1 Weakness

Currently, AquMoSys is only focused for monitoring purpose. For further development, controlling function can be added to enrich the features, such as balancing the pH of water automatically when it is not stable also notify the user when the plant is ready to harvest.

2.6.2 Recommendation

At this moment pH level, water turbidity, water temperature and humidity level just can be displayed in term of giving the information to the users. When the reading of the sensors is above or below the threshold value that has been set, push notification and email alert will notify the users through their smartphones and action can be taken properly. However, this action still must depend on the user to be executed. By adding a control function, system can automate what actions will be taken to restore the condition to normal. As an example, we can add a heater in the system so that when the water temperature is unstable or exceeds the threshold value, the heater will stabilize the water temperature to normal conditions according to the threshold value that has been set. The same goes for the pH level, if the ph sensor's reading is exceeds (above/below) the threshold value, a mini water pump can be used to pump a certain liquid to stabilize the pH level of the water.

2.7 Summary

In a nutshell, this chapter has described about the project description, problem statement, project objectives, project scope, project significance, weakness, and recommendation of the project. AquMoSys should provide convenience and practicality for users in monitoring the condition of the aquaponic system. As technological sophistication develops, especially in mobile applications and IoT such as Android and Flutter, AquMoSys can be very useful in the management and maintenance of aquaponics systems. Furthermore, the next chapter (chapter 3) will be discusses about methodology, i.e. overall phases, description of phases, and summary.

3.0 METHODOLOGY

This chapter explains about the methodology which is used to create and developed AquMoSys.

It consists several sub-chapters namely, (3.1) Overall Phases, (3.2) Description of Each Phases and (3.3) Summary.

3.1 Overall Phases

Evolutionary prototyping model (also known as breadboard) will be used with other necessary steps added to it. This system uses evolutionary prototyping because this type of prototyping is a model that allows the developer to continuously refine the system as other requirements are getting clearer [22].

There are six steps in Evolutionary Prototyping model as shown below in Figure 4,

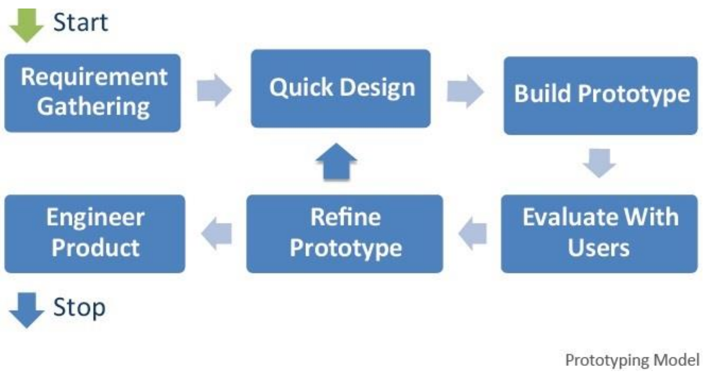


Figure 5: The workflow of Evolutionary Prototyping

3.2 Description of Each Phases

a) Requirement Gathering

This phase involves the collection and analysis of all information necessary to create the system. The information received will then be analyzed to identify the system requirements. In this phase, all functional and non-functional requirements were gathered to get sufficient information and model.

b) Quick design

This phase involves the design of the system, look, and feel and how each component interacts with each other in the form of a use case diagram and the schematic of the system. After that, a low-fidelity prototype will be created. In this phase, the basic interface of this flutter mobile application was designed and created. In addition, any additional features were also added to enrich the features of this mobile application. Basic arrangement to place all the components (NodeMCU ESP32S microcontroller, Turbidity sensor, pH sensor, Waterproof Temperature sensor, and Humidity sensor) in the PCB was designed based on the size of the components. Getting the small arrangement was prioritized to get the smallest size of the PCB.

c) Build prototype

After the design is done, the first rendition of the prototype will be built. The system will be developed using Visual Studio Code, Flutter SDK and Arduino IDE. After the app's interface and system design was finished, the prototype was started to build. Visual Studio Code and Flutter SDK was used to create the mobile application prototype while Arduino IDE was used to create the code script of the system and upload it into NodeMCU ESP32S microcontroller. To power the ESP32S, standard mobile phone micro-USB charger is used.

d) Evaluate with users

The system will then be brought and shown to the supervisor/colleagues to evaluate and seek opinions on what could be added to the system, or what could possibly improve it. In this phase, after finishing the prototype, the overall system was tested by the lecturer/supervisor to see whether the system meets all the requirements or not.

e) Refine Prototype

When feedback is received, and the system requires improvements, it will be refined to fulfil all the added requirements to it. When the prototype is finished, it will be shown again for evaluation. This phase and the previous phase will be repeated several times until the prototype fulfils all the requirements of the system. After being tested by the lecturer/supervisor, the feedbacks gathered were implemented in the system.

f) Engineer Product

When the system is perfect and requires no further improvements, the system is ready to use. After all the feedbacks were implemented, AquMoSys is ready to use.

3.3 Summary

In short, this chapter has discussed about overall phases in Evolutionary prototyping, and how each phase has been worked and executed. In the next chapter (chapter 4), the conclusion for all chapter will be summarized.

4.0 CONCLUSION

In the end, the development of this IoT-based aquaponics monitoring system project and this Android application has resulted in benefits that can help users manage and monitor their aquaponics system.

Evolutionary prototyping model was chosen because this model allows the developer to continuously refine the system as other requirements are getting clearer.

The creation and development of this project from the start has presented quite a big challenge in completing this project in just 6 months. Even when this project was first assigned to me, I did not have any basic knowledge related to making this project. Started everything from scratch, such as what components will be used in making this project, how the series of all the existing components can be arranged properly and neatly, how all the existing components can be connected and can function properly, and how specifications of each component. Once done, how can I program these components to be connected and all the sensors that used can function properly without the slightest error. After everything has been confirmed that it can run and function, some of the existing sensors must also be calibrated before they can be used so that the value reading is correct.

In order to be monitored and monitored anytime and anywhere, an android application that uses the Flutter UI was created to display all readings from each sensor which is hereinafter called AquMoSys. Animated graphics are used to enhance the appearance of recorded data, besides that all data that has been recorded in the last 24 hours can also be viewed using this AquMoSys application. Not only that, even all the data that has been retrieved in the previous days can also be displayed easily and conveniently by users. In addition, notification and email alert features are also included in this mobile application. AquMosys users can manually set the threshold value of each sensor reading as desired. To be able to get e-mail alerts, users are required to sign in first using an email address.

In accordance with the objectives and scope of the project mentioned in the previous chapter, AquMoSys is believed to have fulfilled all the requirements that were designed before this mobile application was made, namely as a system to monitor aquaponics in real time updates, every 10 (ten) minutes intervals for each sensor will take a reading of each existing parameter. Equipped with Animated Graph, Push Notification & E-mail Alert, and Threshold Setting.

REFERENCES

[1] MCMC-UUM MakerSpace Lab. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/faqs/research-units/mcmc-uum-makerspace-lab

[2] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[3] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[4] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[5] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[6] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[7] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[8] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[9] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[10] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[11] Mission & Vision. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/about-us/mission-vision

[12] MCMC-UUM MakerSpace Lab. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/faqs/research-units/mcmc-uum-makerspace-lab

[13] MCMC-UUM MakerSpace Lab. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/faqs/research-units/mcmc-uum-makerspace-lab

[14] MCMC-UUM MakerSpace Lab. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/faqs/research-units/mcmc-uum-makerspace-lab

[15] MCMC-UUM MakerSpace Lab. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/faqs/research-units/mcmc-uum-makerspace-lab

[16] MCMC-UUM MakerSpace Lab. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/faqs/research-units/mcmc-uum-makerspace-lab

[17] MCMC-UUM MakerSpace Lab. (n.d.). Retrieved August 06, 2020, from http://www.soc.uum.edu.my/faqs/research-units/mcmc-uum-makerspace-lab

[18] Murad, S. A. Z., Harun, A., Mohyar, S. N., Sapawi, R., & Ten, S. Y. (2017). Design of aquaponics water monitoring system using Arduino microcontroller. doi: 10.1063/1.5002442

[19] Vernandhes, W., Salahuddin, N., Kowanda, A., & Sari, S. P. (2017). Smart aquaponic with monitoring and control system based on iot. 2017 Second International Conference on Informatics and Computing (ICIC). doi: 10.1109/iac.2017.8280590

[20] Haryanto, Ulum, M., Ibadillah, A. F., Alfita, R., Aji, K., & Rizkyandi, R. (2019). Smart aquaponic system-based Internet of Things (IoT). Journal of Physics: Conference Series, 1211, 012047. doi: 10.1088/1742-6596/1211/1/012047

[21] Haryanto, Ulum, M., Ibadillah, A. F., Alfita, R., Aji, K., & Rizkyandi, R. (2019). Smart aquaponic system-based Internet of Things (IoT). Journal of Physics: Conference Series, 1211, 012047. doi: 10.1088/1742-6596/1211/1/012047

[22] Sherrell L. (2013) Evolutionary Prototyping. In: Runehov A.L.C., Oviedo L. (eds) Encyclopedia of Sciences and Religions. Springer, Dordrecht

APPENDICES

A. Gantt Chart

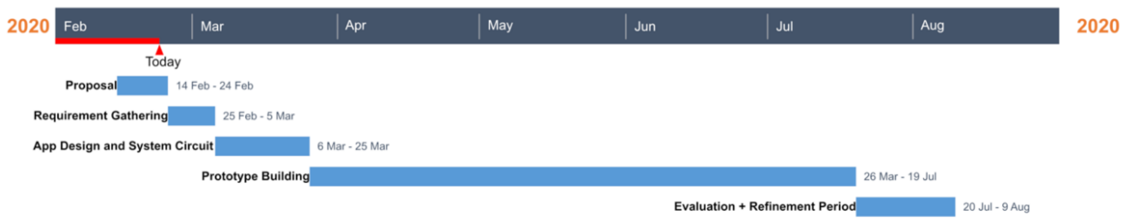


Figure 6: Gantt Chart of AquMoSys

B. Scheduling & Deliverables

START DATE	END DATE	DESCRIPTION	Days	Deliverable
14/02/2020	24/2/2020	Proposal	10	Proposal Document
25/02/2020	5/3/2020	Requirement Gathering	10	Functional Requirement
6/03/2020	25/3/2020	App Design and System Circuit	20	Interface Design + Circuit Schematic
26/3/2020	19/7/2020	Prototype Building	109	Prototype
20/7/2020	9/8/2020	Evaluation Refinement Period	20	Feedback + Final Product

Figure 7: Scheduling & Deliverables of AquMoSys

C. Components (Microcontroller & Sensors)

a) Nodemcu 32S – ESP32

This component is used as the main processor of AquMoSys as well as all necessary sensors will be connected to this board. To power this board, it can simply use standard micro-usb charger, or it also can use battery.

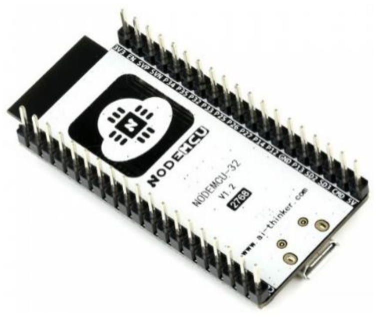


Figure 8: Nodemcu 32S-ESP32

b) Humidity sensor (DHT11)

This sensor is used to measure the humidity of AquMoSys. It has 3 pins connected to Nodemcu 32S board in order to function properly. The three pins consist of GND (Ground) pin, VCC pin, and Data pin.

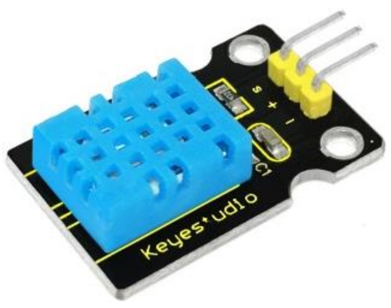


Figure 9: Humidity sensor (DHT11)

c) Waterproof Temperature sensor (DS18B20)

This sensor is waterproof version of DS18B20, so it can be immersed in the water. It is used to measure either the temperature of the water or the surrounding air. Same as DHT11, it also has 3 cables or pins namely VCC pin, GND, and Data pin.

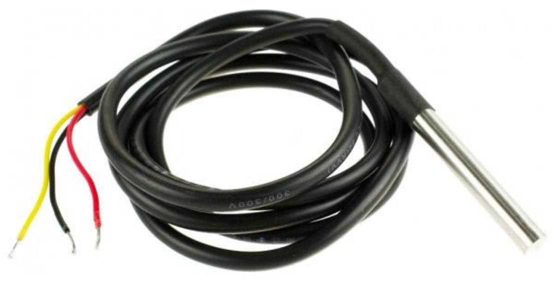


Figure 10: DS18B20 waterproof version

d) Turbidity sensor

This sensor is used to measure the turbidity or how clear a water is. Using NTU units, which stands for Nephelometric Turbidity Unit. It has 5-volt operating voltage to function properly. It also has GND, VCC and Data pins.

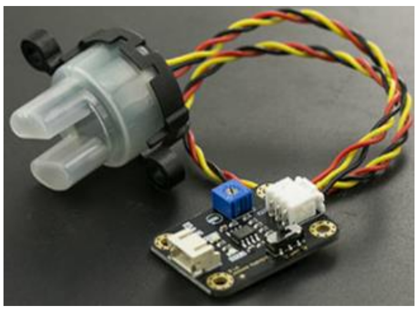


Figure 11: Turbidity sensor by DF Robot

e) pH sensor

This sensor is used to measure pH level of the water. It has range measurement starting from 1 to 14. It has 3.3-5-volt operating voltage to function properly. Same as turbidity sensor, it also has 3 different cables namely VCC, GND and Data.

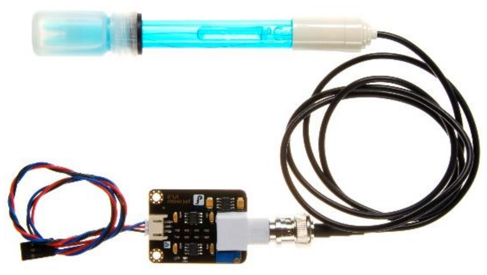


Figure 12: pH sensor V.2 by DF Robot

f) Mobile charger (Micro-USB)

This standard mobile charger is used to power Nodemcu 32S-ESP32. It has micro type B port and just attach it into Nodemcu.

Figure 13: Universal Mobile Charger (Micro USB)

D. Component Schematic

This schematic shows how each of all components are connected. As seen in the schematic below, each of VCC and GND of all four (4) sensors are connected to VIN and GND pin respectively. For the data pin, each of all four sensors have been already assigned into digital or analog pin in Nodemcu 32S-ESP32 namely DHT11 sensor was assigned to pin 32 of Nodemcu 32S-ESP32, DS18B20 waterproof sensor was assigned to pin 33 of Nodemcu 32S-ESP32, pH sensor was assigned to pin 35 of Nodemcu 32S-ESP32 and Turbidity sensor was assigned to pin 34 of Nodemcu 32S-ESP32.

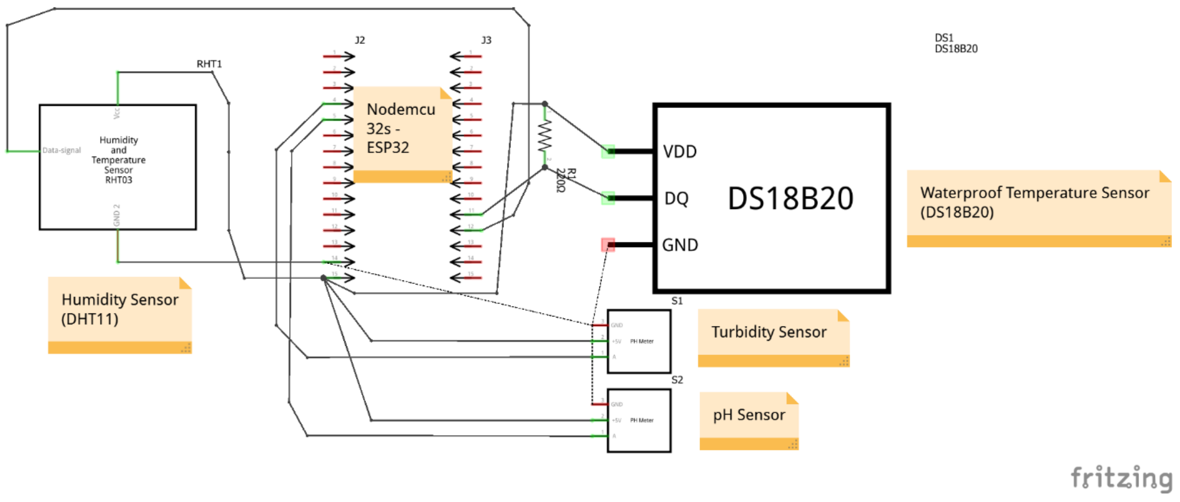


Figure 14: Schematic of all connected components

E. Use Case Diagram

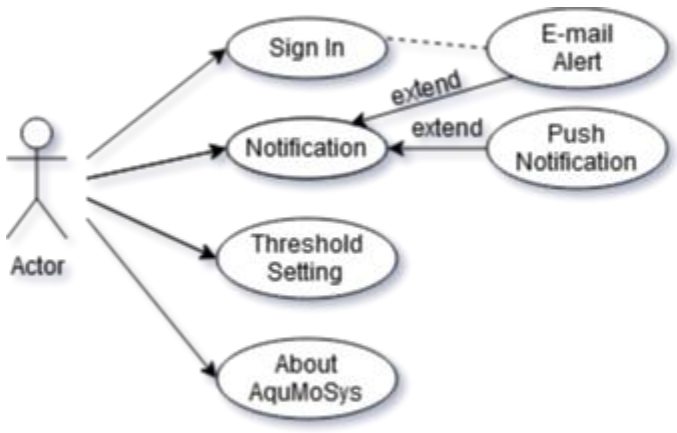


Figure 15: Use Case diagram of AquMoSys

F. Requirements

Below are the functional requirements and non-functional requirement of the system. In the priority column, the following short hands are used:

- ❖ M – Mandatory requirements (something the system must do).
- ❖ D – Desirable requirements (something the system preferably should do).
- ❖ O – Optional requirements (something the system may do).

a) Functional Requirement's List

No.	Requirement		Priority
1.	**Log in.**		
	1.1	Users shall input/log in using their e-mail address to get the e-mail alert.	O
2.	**Notification**		
	2.1	Users shall turn on the push notification.	O
	2.2	Users shall turn off the push notification.	O
	2.3	Users shall turn on the e-mail alert.	O
	2.4	Users shall turn off the e-mail alert.	O
3.	**Threshold Setting** .		
	3.1	Users shall manually set and update the desired threshold value.	M
4.	**About AquMoSys**		
	4.1	Users shall view the detail description about AquMoSys app.	M

Table 1: Functional Requirment's List

b) Non-functional Requirement's List

No.	Requirement		Priority
1.	**Performance**		
	1.1	AquMoSys should only use minimum data usage to run its function.	M
2.	**Security**		
	2.1	The retrieved value of Turbidity, Temperature, Humidity, pH level, and timestamp will be stored in Firebase Realtime Database.	M
	2.2	The e-mail used by users for logging in to get e-mail alert will be stored in Firebase Cloud Firestore.	M
3.	**Compatibility** .		
	3.1	AquMoSys should run properly in Android platform with minimum version of Android 8.0 (Oreo).	M

4.	**Usability & Reliability**		
	4.1	AquMoSys should be represented in a simple and understandable user interface.	M
	4.2	AquMoSys should be easy to operate.	M
	4.3	The Firebase Realtime Database, Cloud Firestore and Nodemcu 32s – ESP32 should be online anytime.	M

Table 2: Non-Functional Requirement’s List

G. User Interface

a) Splash screen

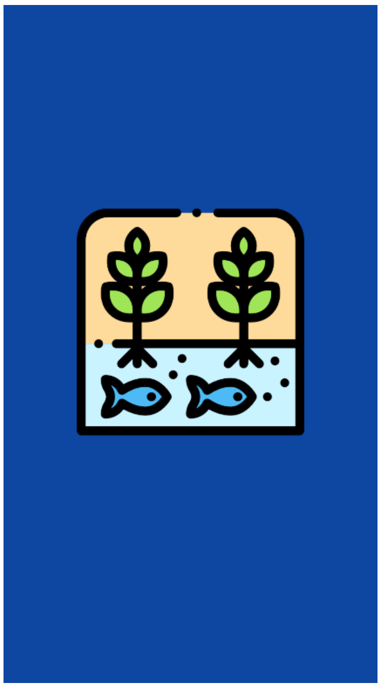


Figure 16: Splashscreen page of AquMoSys

b) Dashboard

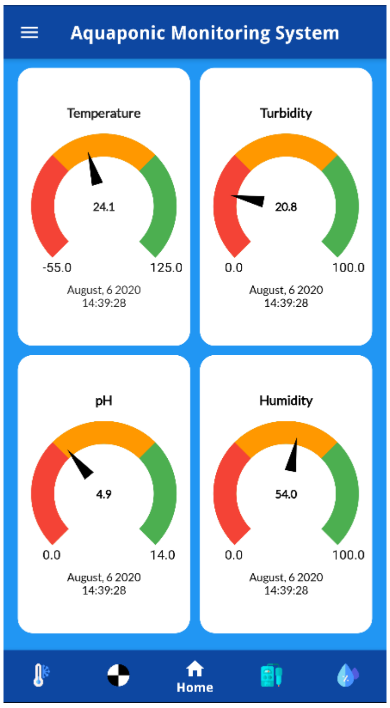


Figure 17: Dashboard page of AquMoSys

c) Drawer

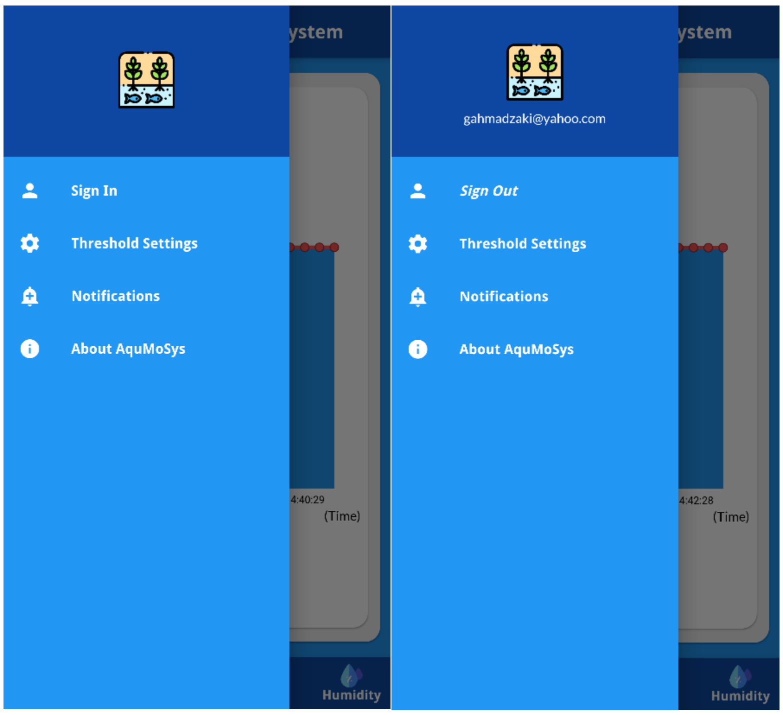
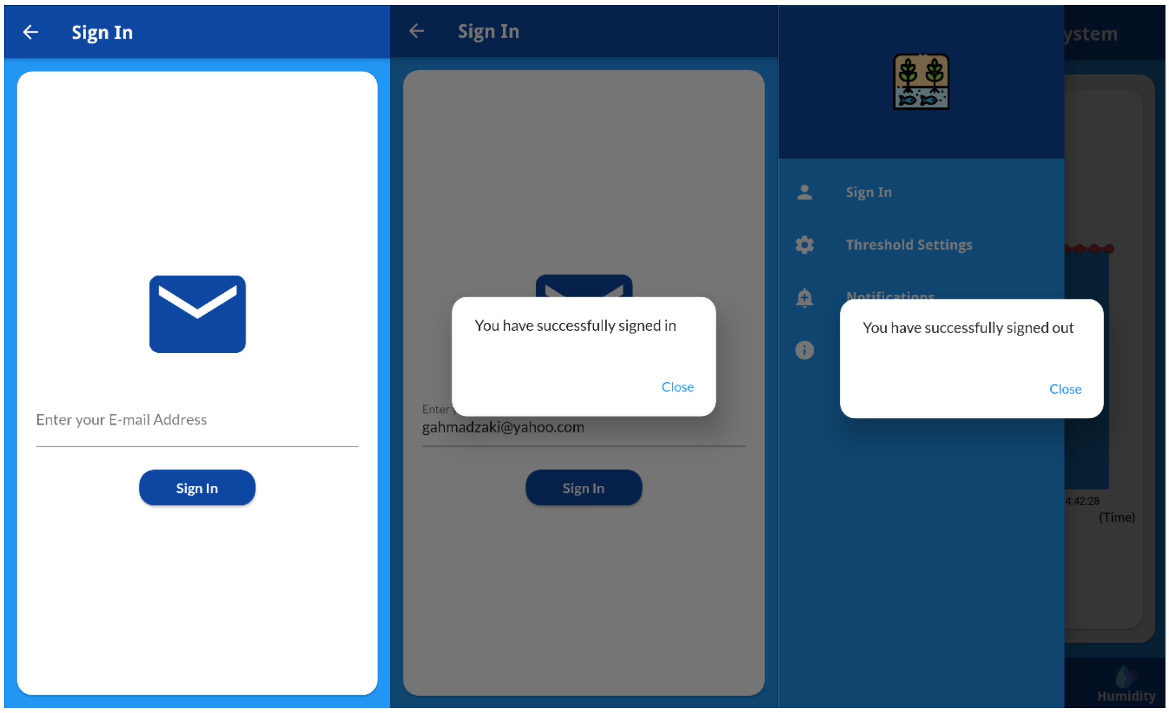


Figure 18: Drawer page of AquMoSys

d) Sign In



You have successfully signed in

You have successfully signed out

Figure 19: Sign In page of AquMoSys

e) Threshold Setting

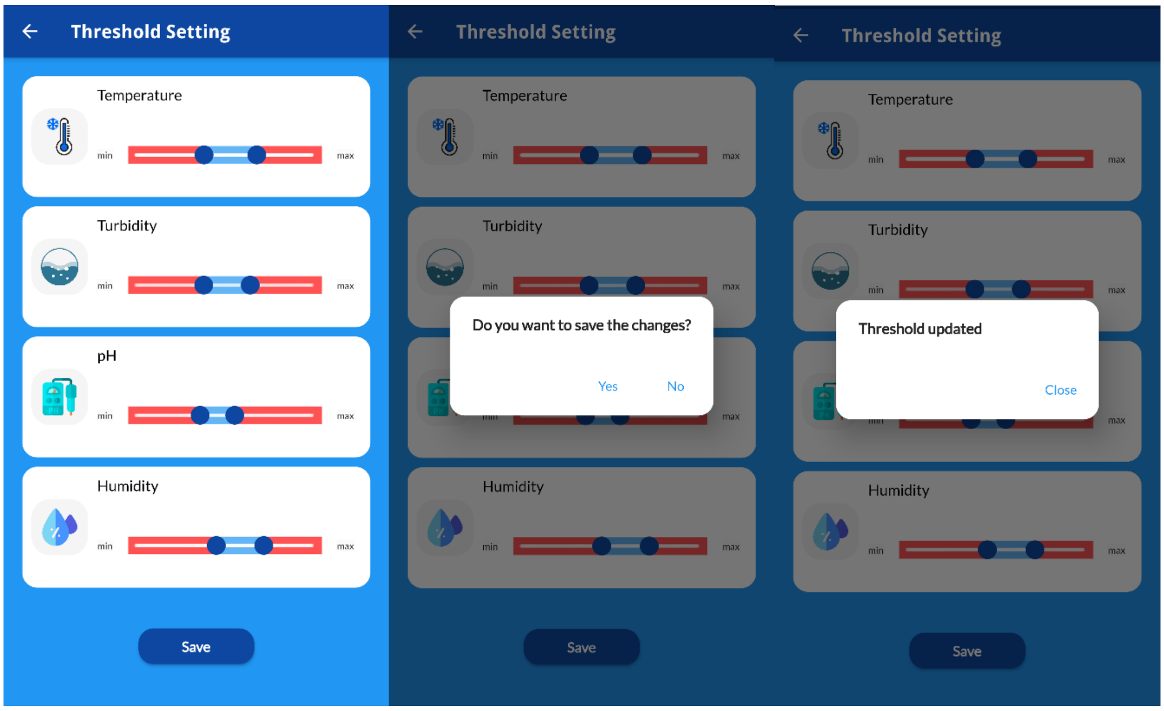


Figure 20: Threshold Setting page of AquMoSys

f) Notification

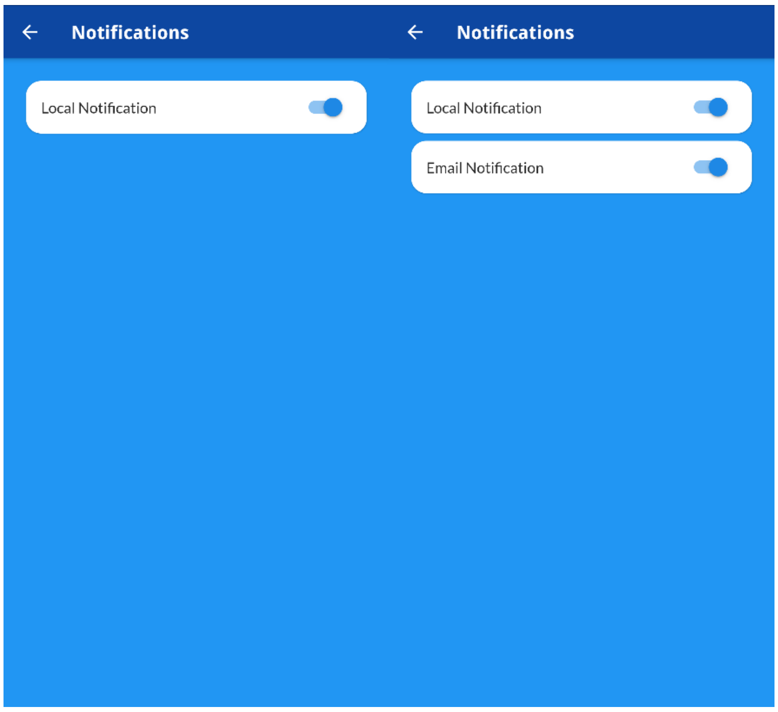
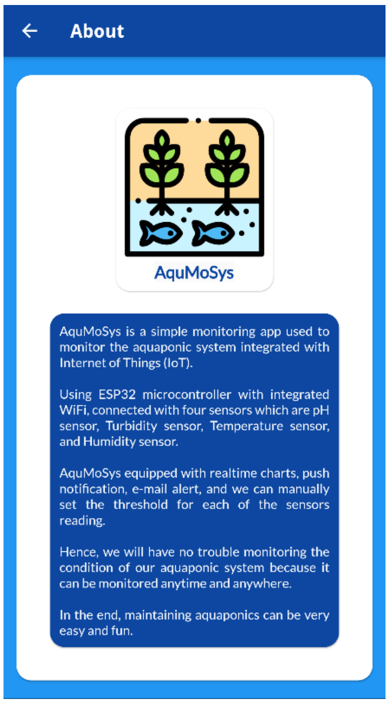


Figure 21: Notification page of AquMoSys

g) About AquMosys



AquMoSys is a simple monitoring app used to monitor the aquaponic system integrated with Internet of Things (IoT).

Using ESP32 microcontroller with integrated WiFi, connected with four sensors which are pH sensor, Turbidity sensor, Temperature sensor, and Humidity sensor.

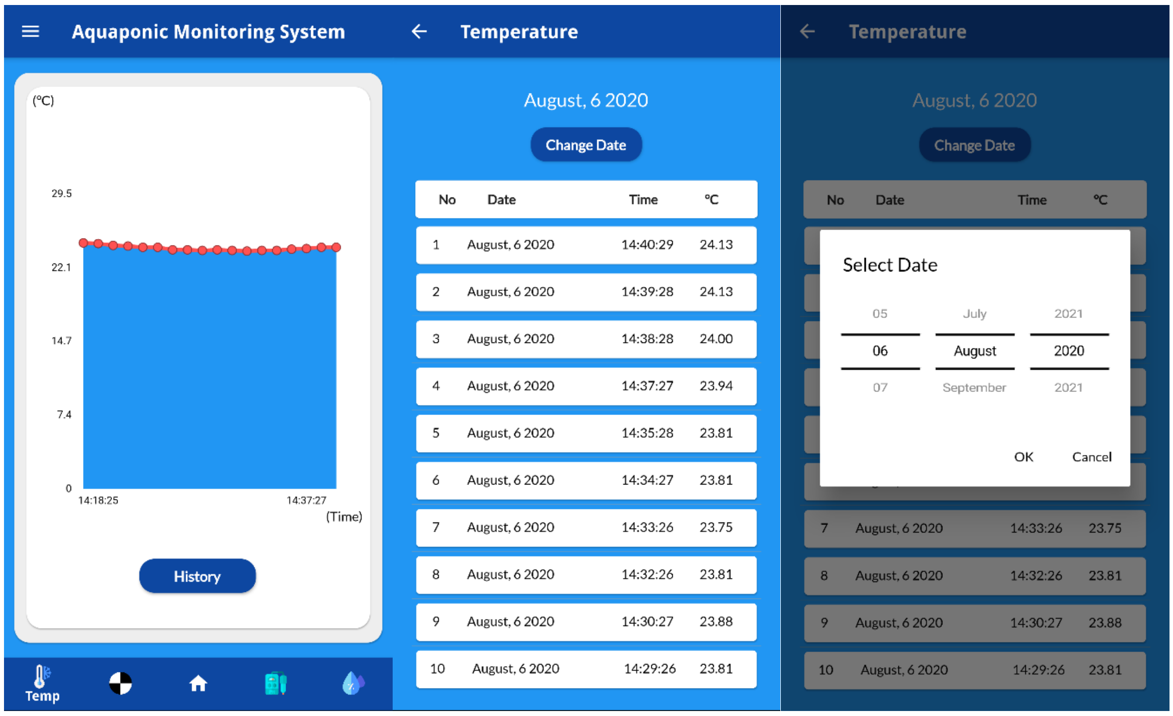
AquMoSys equipped with realtime charts, push notification, e-mail alert, and we can manually set the threshold for each of the sensors reading.

Hence, we will have no trouble monitoring the condition of our aquaponic system because it can be monitored anytime and anywhere.

In the end, maintaining aquaponics can be very easy and fun.

Figure 22: About page of AquMoSys

h) Temperature



August, 6 2020

Change Date

No	Date	Time	°C
1	August, 6 2020	14:40:29	24.13
2	August, 6 2020	14:39:28	24.13
3	August, 6 2020	14:38:28	24.00
4	August, 6 2020	14:37:27	23.94
5	August, 6 2020	14:35:28	23.81
6	August, 6 2020	14:34:27	23.81
7	August, 6 2020	14:33:26	23.75
8	August, 6 2020	14:32:26	23.81
9	August, 6 2020	14:30:27	23.88
10	August, 6 2020	14:29:26	23.81

August, 6 2020

Change Date

Select Date

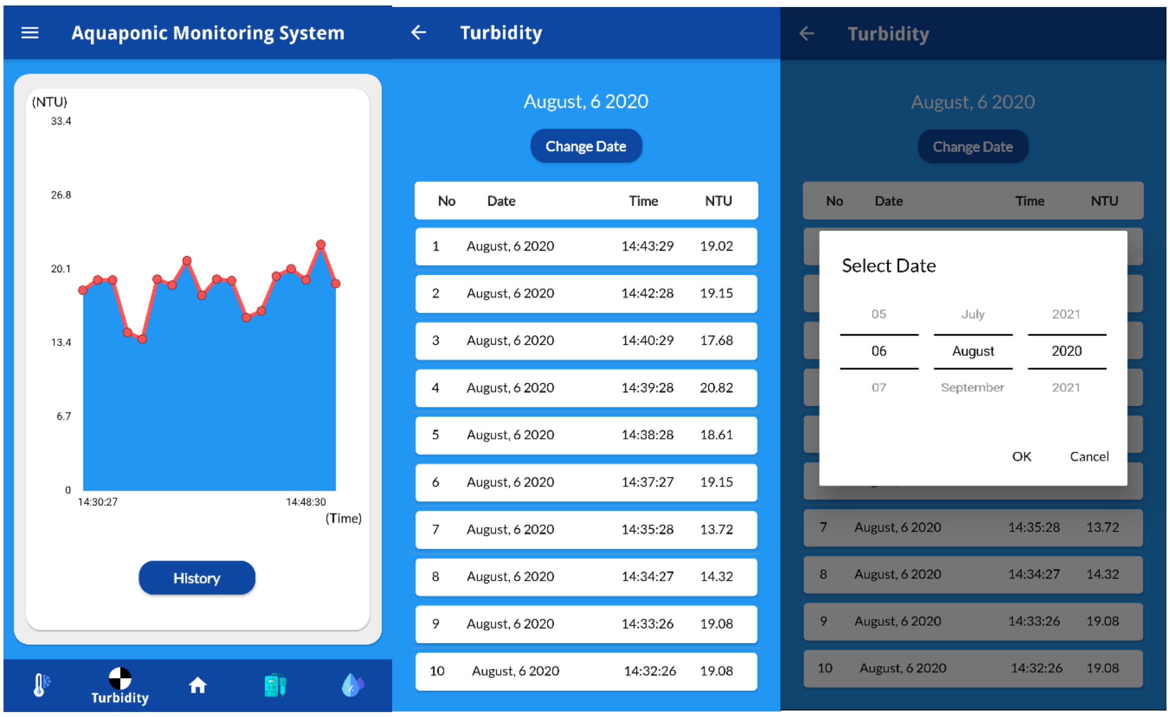
05	July	2021
06	August	2020
07	September	2021

OK Cancel

No	Date	Time	°C
7	August, 6 2020	14:33:26	23.75
8	August, 6 2020	14:32:26	23.81
9	August, 6 2020	14:30:27	23.88
10	August, 6 2020	14:29:26	23.81

Figure 23: Realtime graph and 24 hours data history page of Temperature sensor

i) Turbidity



Turbidity

August, 6 2020

Change Date

No	Date	Time	NTU
1	August, 6 2020	14:43:29	19.02
2	August, 6 2020	14:42:28	19.15
3	August, 6 2020	14:40:29	17.68
4	August, 6 2020	14:39:28	20.82
5	August, 6 2020	14:38:28	18.61
6	August, 6 2020	14:37:27	19.15
7	August, 6 2020	14:35:28	13.72
8	August, 6 2020	14:34:27	14.32
9	August, 6 2020	14:33:26	19.08
10	August, 6 2020	14:32:26	19.08

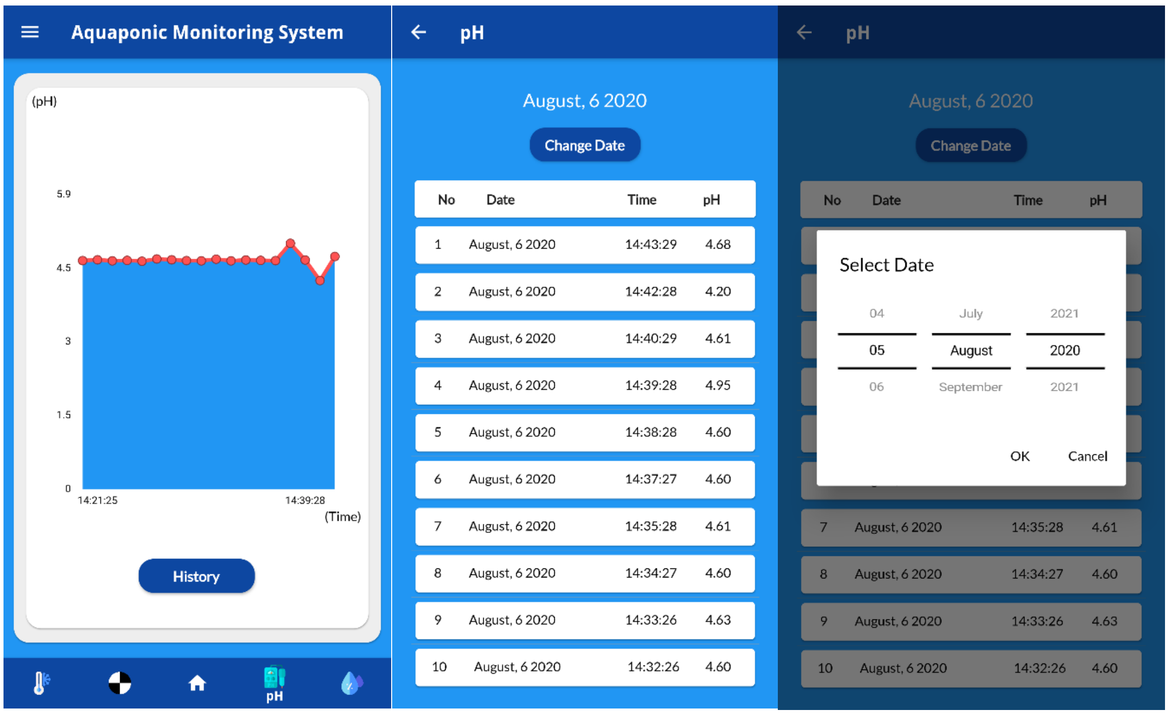
Select Date

05	July	2021
06	August	2020
07	September	2021

OK Cancel

Figure 24: Realtime graph and 24 hours data history page of Turbidity sensor

j) pH



pH

August, 6 2020

Change Date

No	Date	Time	pH
1	August, 6 2020	14:43:29	4.68
2	August, 6 2020	14:42:28	4.20
3	August, 6 2020	14:40:29	4.61
4	August, 6 2020	14:39:28	4.95
5	August, 6 2020	14:38:28	4.60
6	August, 6 2020	14:37:27	4.60
7	August, 6 2020	14:35:28	4.61
8	August, 6 2020	14:34:27	4.60
9	August, 6 2020	14:33:26	4.63
10	August, 6 2020	14:32:26	4.60

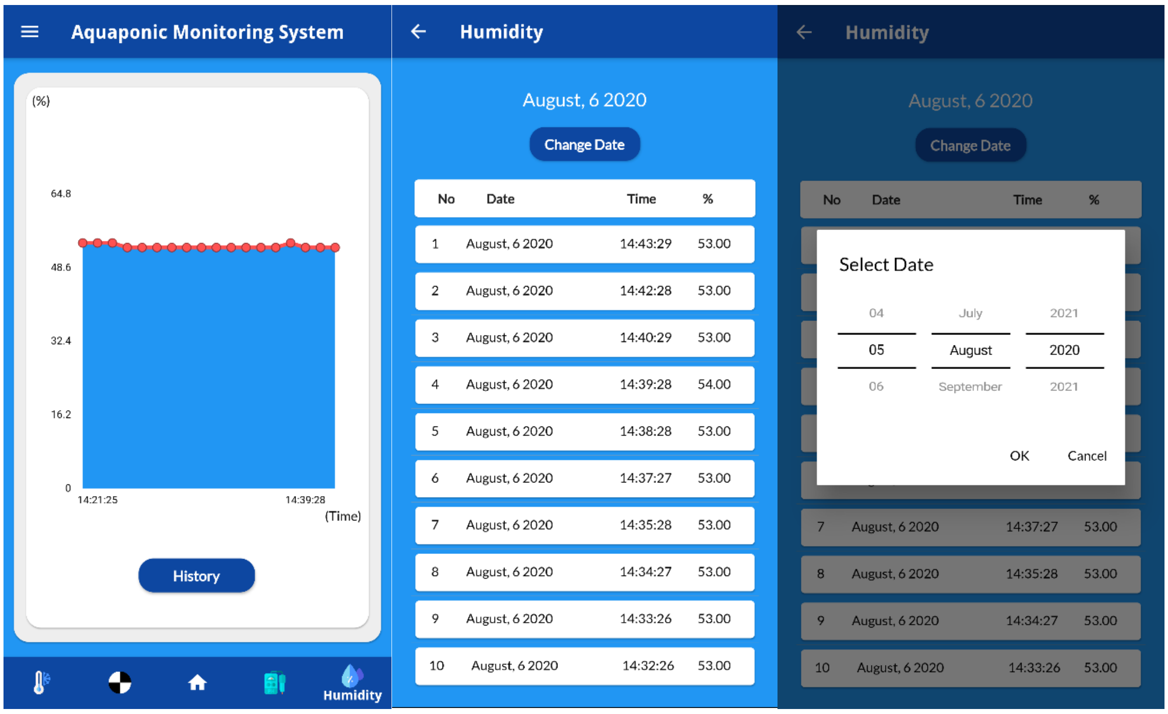
Select Date

04	July	2021
05	August	2020
06	September	2021

OK Cancel

Figure 25: Realtime graph and 24 hours data history page of pH sensor

k) Humidity

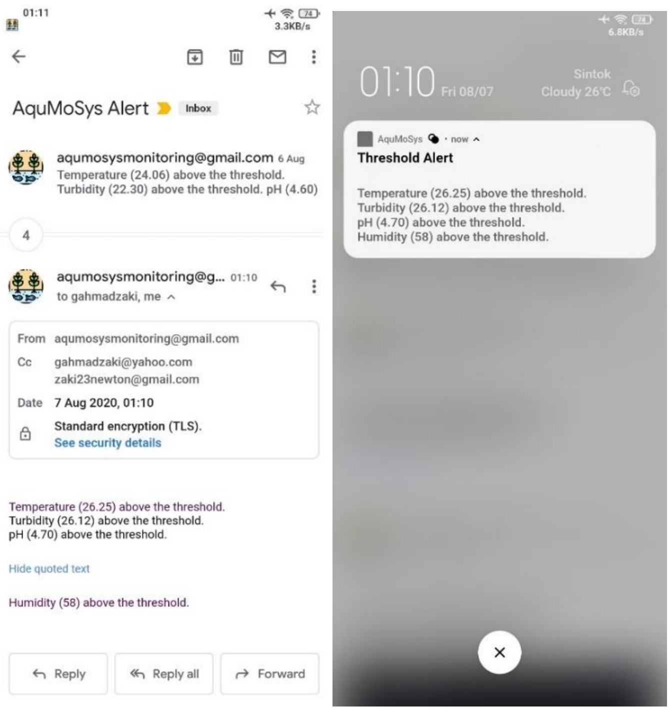


No	Date	Time	%
1	August, 6 2020	14:43:29	53.00
2	August, 6 2020	14:42:28	53.00
3	August, 6 2020	14:40:29	53.00
4	August, 6 2020	14:39:28	54.00
5	August, 6 2020	14:38:28	53.00
6	August, 6 2020	14:37:27	53.00
7	August, 6 2020	14:35:28	53.00
8	August, 6 2020	14:34:27	53.00
9	August, 6 2020	14:33:26	53.00
10	August, 6 2020	14:32:26	53.00

No	Date	Time	%
7	August, 6 2020	14:37:27	53.00
8	August, 6 2020	14:35:28	53.00
9	August, 6 2020	14:34:27	53.00
10	August, 6 2020	14:33:26	53.00

Figure 26: Realtime graph and 24 hours data history page of Humidity sensor

l) Push Notification & E-mail Alert



AquMoSys Alert

aqumosysmonitoring@gmail.com 6 Aug
Temperature (24.06) above the threshold.
Turbidity (22.30) above the threshold. pH (4.60)

4

aqumosysmonitoring@g... 01:10
to gahmadzaki, me

From aqumosysmonitoring@gmail.com
Cc gahmadzaki@yahoo.com
zaki23newton@gmail.com
Date 7 Aug 2020, 01:10
Standard encryption (TLS).
See security details

Temperature (26.25) above the threshold.
Turbidity (26.12) above the threshold.
pH (4.70) above the threshold.

Hide quoted text

Humidity (58) above the threshold.

AquMoSys · now
Threshold Alert

Temperature (26.25) above the threshold.
Turbidity (26.12) above the threshold.
pH (4.70) above the threshold.
Humidity (58) above the threshold.

Figure 27: Email alert & Push Notiffication page of AquMoSys

m) Components After Being Assembled (Soldered)

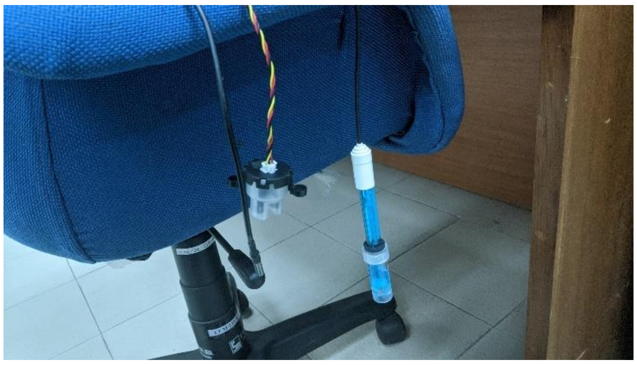


Figure 28: Turbidity, DS18B20 waterproof, and pH sensors

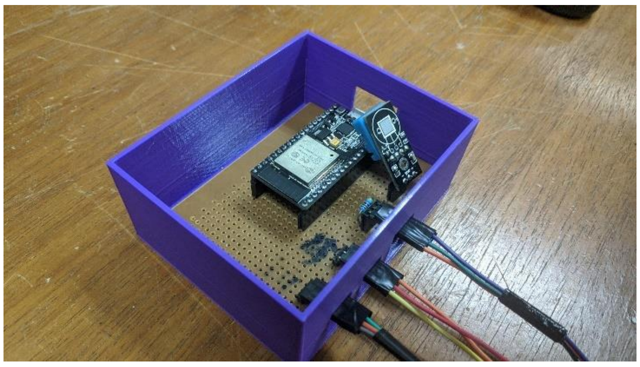


Figure 29: Nodemcu 32S-ESP32 soldered to the stripboard. DHT11, Turbidity, pH and Temperature sensor

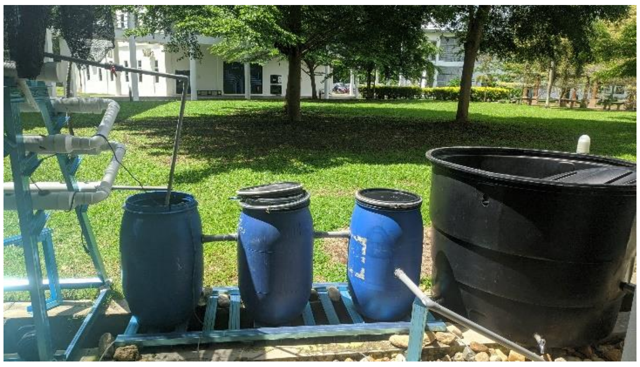
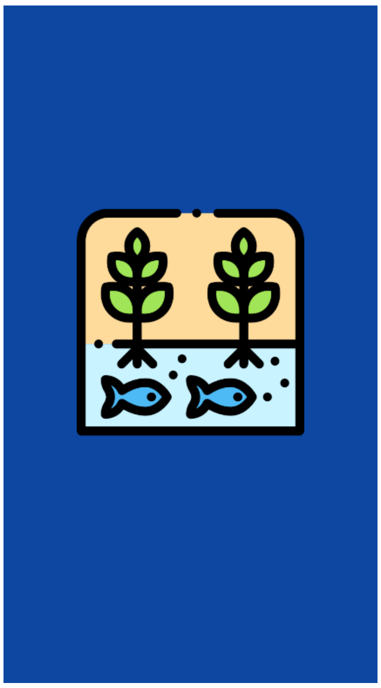


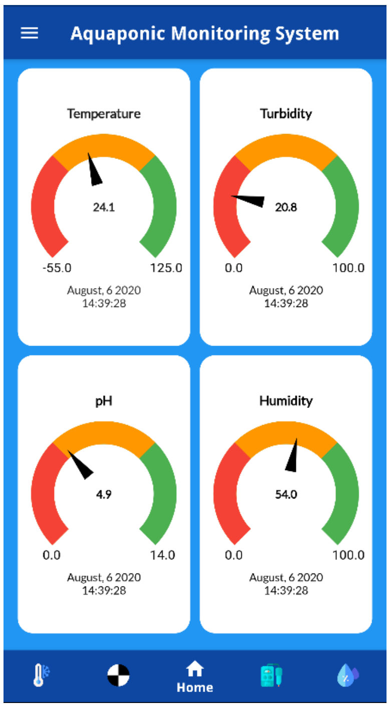
Figure 30: Aquaponic Ecosystem was created using plastic pipe and big plastic drum

n) AquMoSys Operating Instructions

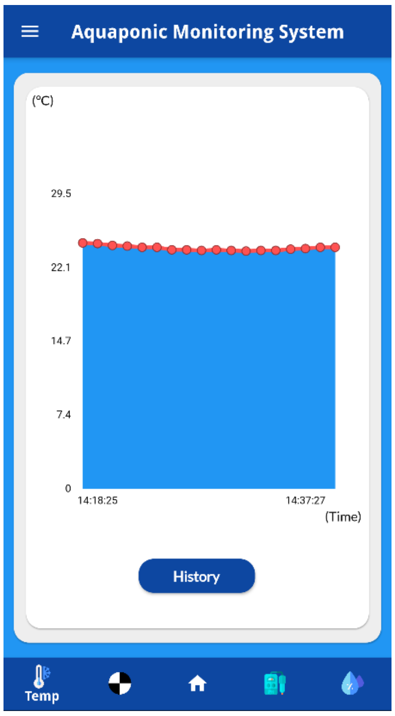
1. Launch AquMoSys app in your smartphone. When it is opening, a splashcreen page will appear as seen below.



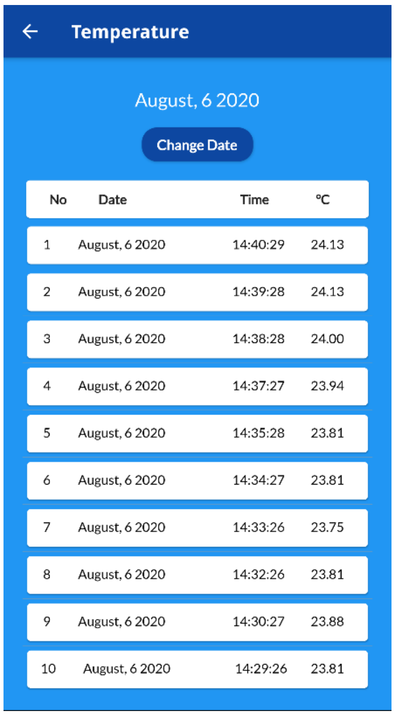
2. After that, a main dashboard page will appear as seen below. In this page, you can see four (4) gauges represents each of sensor's current condition. Each of gauge can be selected and it will direct to real-time chart page.



3. When one of the gauges is selected, for example temperature gauge. It will turn to real-time chart page as seen below. It has a history button to show more data in last 24 hours that retrieved for every 5 minutes. We can also see a detail value by just touch and hold the blue area in the charts.

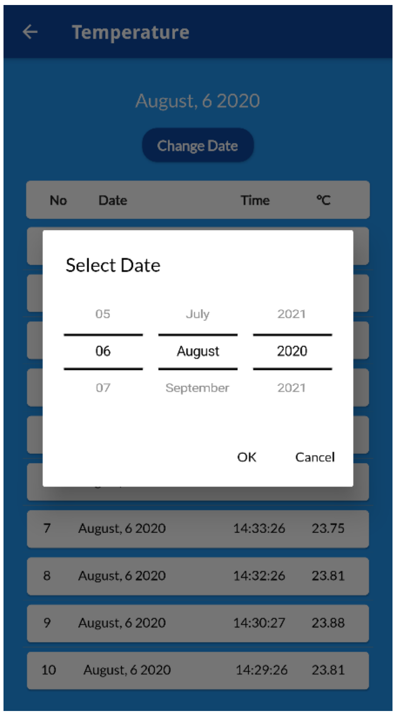


4. When "History" button is selected, it will direct to list of last 24 hours data retrieved as seen below. We can also see the previous data retrieved in another day by selecting "Change Date" button.



No	Date	Time	°C
1	August, 6 2020	14:40:29	24.13
2	August, 6 2020	14:39:28	24.13
3	August, 6 2020	14:38:28	24.00
4	August, 6 2020	14:37:27	23.94
5	August, 6 2020	14:35:28	23.81
6	August, 6 2020	14:34:27	23.81
7	August, 6 2020	14:33:26	23.75
8	August, 6 2020	14:32:26	23.81
9	August, 6 2020	14:30:27	23.88
10	August, 6 2020	14:29:26	23.81

5. When “Change Date” button is selected, a new page will appear as seen below. We can manually change the desired date and select “OK” when finished. New data list will appear based on date we have chosen.


Temperature

August, 6 2020

Change Date

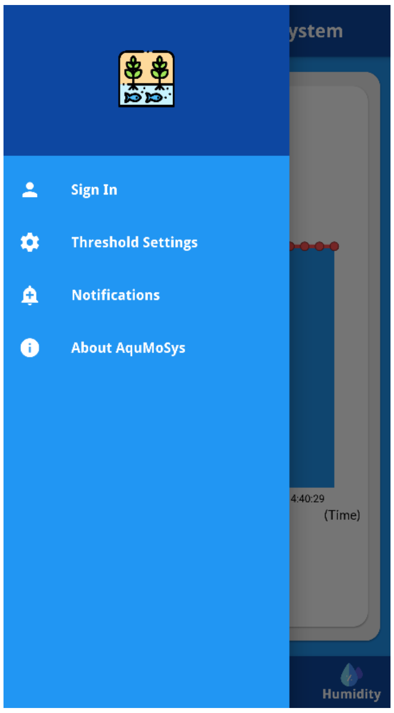
Select Date

05	July	2021
06	August	2020
07	September	2021

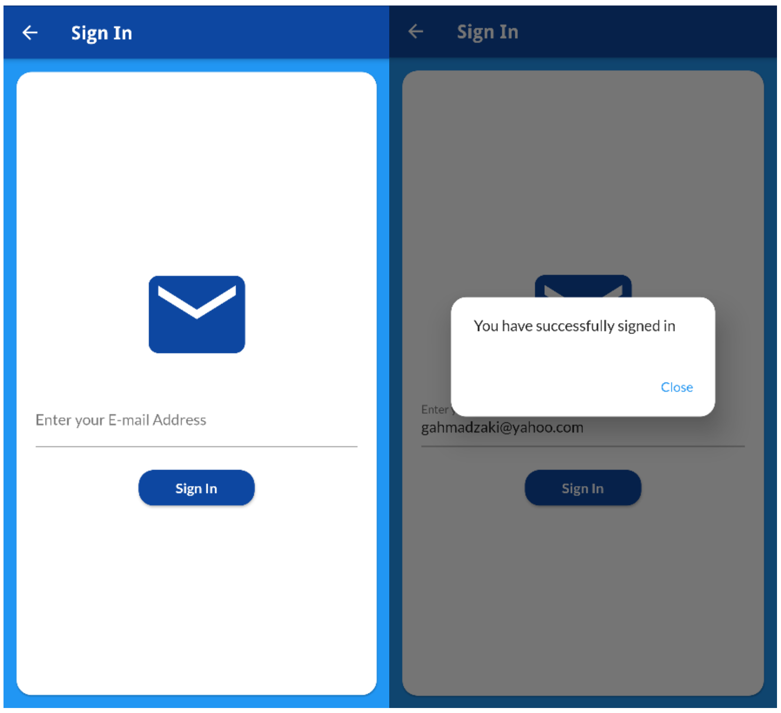
OK Cancel

No	Date	Time	°C
7	August, 6 2020	14:33:26	23.75
8	August, 6 2020	14:32:26	23.81
9	August, 6 2020	14:30:27	23.88
10	August, 6 2020	14:29:26	23.81

6. Back to main dashboard, select “Three white line” button in top left corner. Drawer page will appear as seen below. It has several menus namely, “Sign In”, “Threshold Setting”, “Notifications”, and “About AquMoSys”.

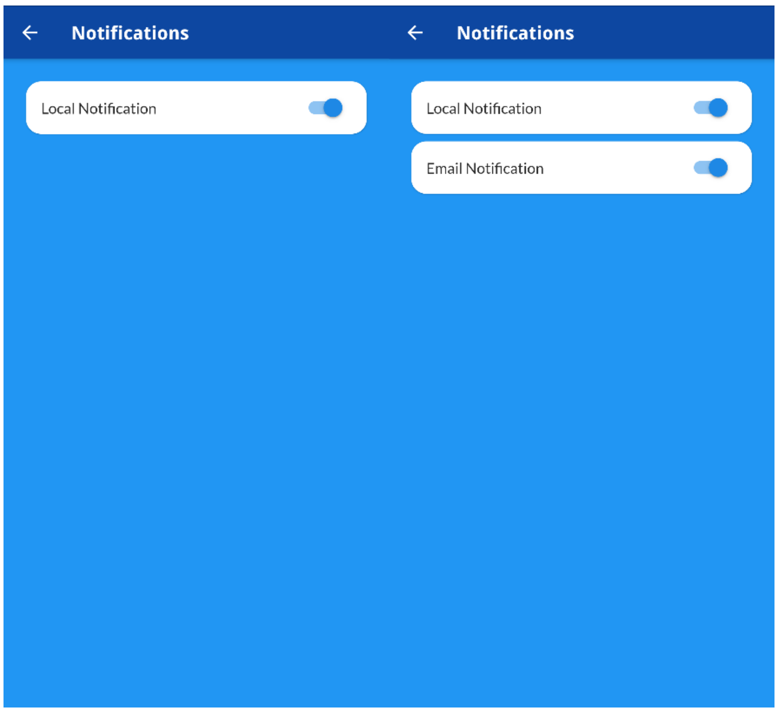


7. Select "Sign In" menu to enter our email in order to get the email alert if the condition of Aquaponic is not stable. After input the email, click "Sign In" button and pop up message will appear, and you have successfully signed in as seen below.

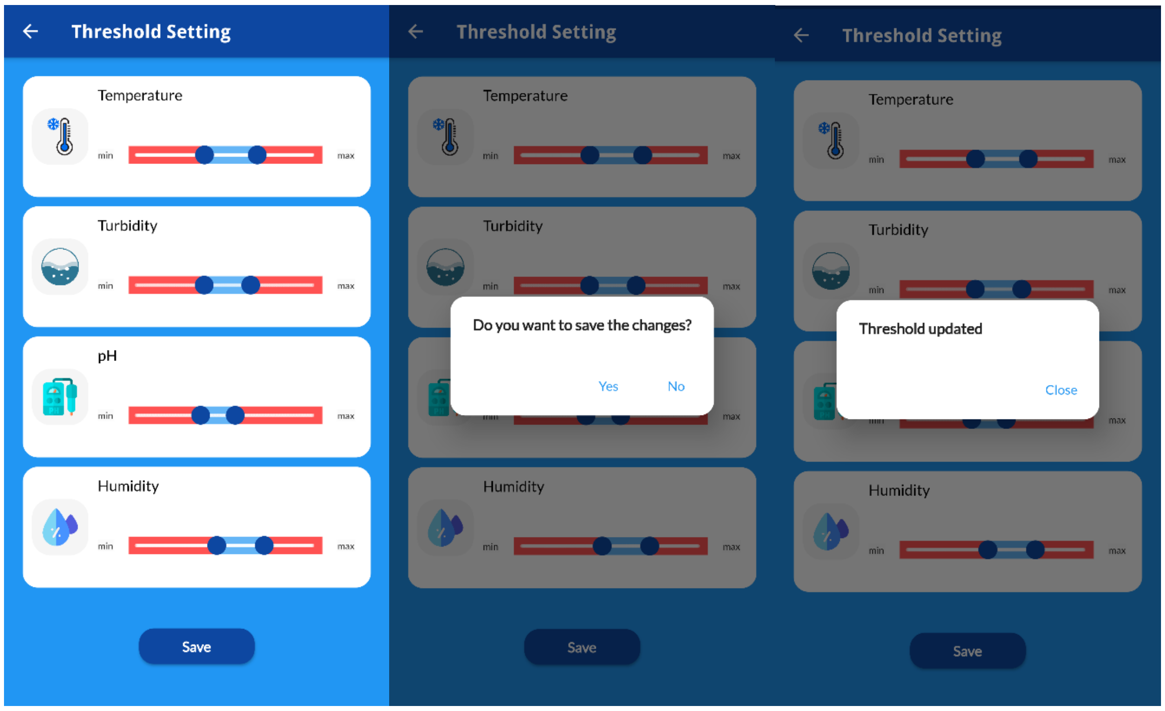


You have successfully signed in

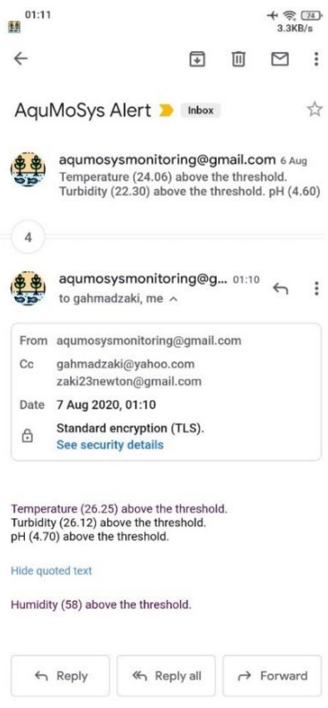
8. To see the changes, back to drawer page, select "Notification" menu. As seen below additional menu namely "Email Alert" will appear because we have already signed in using our email. Otherwise, when you do not sign in, "Email Alert" menu will not appear. We can also manually turn on or turn off for each of notifications.



9. To set the threshold value for each of sensor's reading, back to drawer page, select "Threshold Setting" menu and new page will appear as seen below. As we can see, each of sensors is represented by one slider function. We can freely move the both sides of the slider to set the minimum (left slider) and maximum (right slider) of the threshold sensor's value. Red color on the slider represents the value exceeds (above or below) the threshold. Blue color represents the minimum and maximum of the threshold value. After finished changing the value, we need to save the updated threshold by simply selecting "Save" button and pop message will appear to confirm our action, select OK and the threshold has been successfully updated.



10. Since we have already signed in using the email, updated the threshold as mentioned in previous step. Now, we can get a push notifcation and an email alert when one or any of the sensor's condition is not stable as seen below. When all the sensors simultaneously shows unstable condition, an email alert or push notification will show up to notify us.



AquMoSys Alert Inbox

aqumosysmonitoring@gmail.com 6 Aug
Temperature (24.06) above the threshold. Turbidity (22.30) above the threshold. pH (4.60)

4

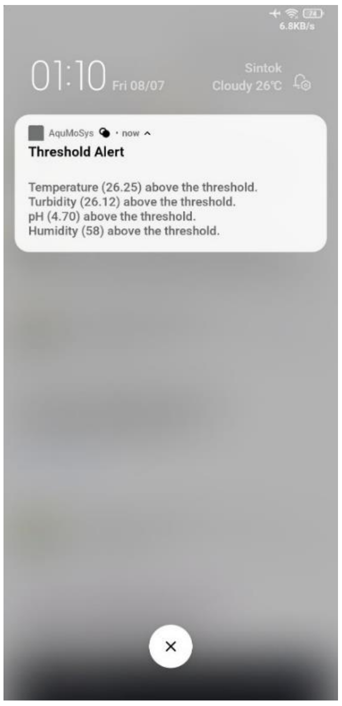
aqumosysmonitoring@g... 01:10
to gahmadzaki, me

From	aqumosysmonitoring@gmail.com
Cc	gahmadzaki@yahoo.com zaki23newton@gmail.com
Date	7 Aug 2020, 01:10
	Standard encryption (TLS). See security details

Temperature (26.25) above the threshold.
Turbidity (26.12) above the threshold.
pH (4.70) above the threshold.

Hide quoted text

Humidity (58) above the threshold.

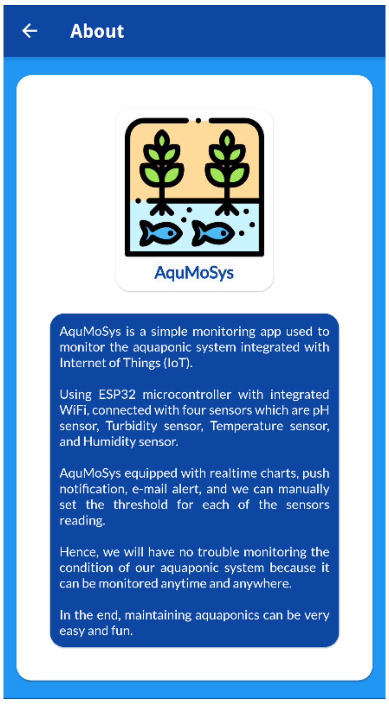


AquMoSys · now

Threshold Alert

Temperature (26.25) above the threshold.
Turbidity (26.12) above the threshold.
pH (4.70) above the threshold.
Humidity (58) above the threshold.

11. Lastly, "About AquMoSys" menu shows us about the brief description of AquMoSys as seen below.



AquMoSys is a simple monitoring app used to monitor the aquaponic system integrated with Internet of Things (IoT).

Using ESP32 microcontroller with integrated WiFi, connected with four sensors which are pH sensor, Turbidity sensor, Temperature sensor, and Humidity sensor.

AquMoSys equipped with realtime charts, push notification, e-mail alert, and we can manually set the threshold for each of the sensors reading.

Hence, we will have no trouble monitoring the condition of our aquaponic system because it can be monitored anytime and anywhere.

In the end, maintaining aquaponics can be very easy and fun.

H. Source Code (Flutter)

a) lit_data.dart

```
import 'dart:convert';

List<LitData> bannerModelFromJson(String str) =>
List<LitData>.from(json.decode(str).map((x) => LitData.fromJson(x)));

String litDataToJson(List<LitData> data) =>
json.encode(List<dynamic>.from(data.map((x) => x.toJson())));

class LitData {

  LitData({
      this.data,
      this.timestamp
  });


    final double data;
    final int timestamp;



    factory LitData.fromJson(Map<String, dynamic> json) => LitData(
       data : json['data'],
       timestamp : json['timestamp'],
    );

    Map<String, dynamic> toJson() =>
      {
        'data' : data,
        'timestamp' : timestamp
      };
}
```

b) about.dart

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

class AboutPage extends StatelessWidget {
  const AboutPage({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.blue[500],
      //FONTCHANGE appbar
      appBar: AppBar(
        title: Text('About',
            style: GoogleFonts.droidSans(
                textStyle: TextStyle(
              fontWeight: FontWeight.bold,
            ))),
),
body: SingleChildScrollView(
  child: Card(
    margin: EdgeInsets.symmetric(horizontal: 14, vertical: 20),
    shape:
        RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
    child: Padding(
      padding: EdgeInsets.all(32.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Card(
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(16)),
            color: Colors.white,
            child: Container(
              padding: EdgeInsets.all(10),
              decoration:
                  BoxDecoration(borderRadius: BorderRadius.circular(16)),
              child: Column(
                children: <Widget>[
                  Image(
                    image: AssetImage(
                      'assets/icon/splashscreen.png',
                    ),
                    height: 150,
                    width: 150,
                  ),
                  SizedBox(
                    height: 5,
                  ),
                  Material(
                      child: Text(
                    'AquMoSys',
                    style: TextStyle(
                      fontWeight: FontWeight.bold,
                      fontSize: 18,
                      color: Colors.blue[900],
                    ),
                  ))
                ],
              ),
            ),
          ),
          SizedBox(
            height: 16,
          ),
          Card(
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(16)),
            color: Colors.blue[900],
            child: Container(
              padding: EdgeInsets.all(10),
                    child: Center(
                      child: Text(
                        'AquMoSys is a simple monitoring app used to monitor the aquaponic system
integrated with Internet of Things (IoT).\n \nUsing ESP32 microcontroller with integrated
WiFi, connected with four sensors which are pH sensor, Turbidity sensor, Temperature
sensor, and Humidity sensor.\n \nAquMoSys equipped with realtime charts, push
notification, e-mail alert, and we can manually set the threshold for each of the sensors
reading. \n \nHence, we will have no trouble monitoring the condition of our aquaponic
system because it can be monitored anytime and anywhere. \n \nIn the end, maintaining
aquaponics can be very easy and fun.',
                        textAlign: TextAlign.justify,
                        style: TextStyle(color: Colors.white, fontSize: 14),
                      ),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

c) chart.dart

```
import 'package:aquamonitoring/pages/wrapper.dart';
import 'package:flutter/material.dart';
import 'package:fl_chart/fl_chart.dart';
import 'package:intl/intl.dart';
import 'package:aquamonitoring/models/lit_data.dart';

class ChartWidget extends StatefulWidget {
  ChartWidget({Key key, this.lists, this.judul, this.param}) : super(key: key);

  final List<LitData> lists;

  final String judul;
  final String param;

  @override
  _ChartWidgetState createState() => _ChartWidgetState();
}

class _ChartWidgetState extends State<ChartWidget> {
  double maxVal(data) {
    dynamic max = data.first;
    data.forEach((e) {
      if (e.data > max.data) max = e;
    });
    print(max.data);
    return max.data;
  }
```

```
double minVal(data) {
  dynamic max = data.first;
  data.forEach((e) {
    if (e.data < max.data) max = e;
  });
  print(max.data);
  if (max.data > 0) return 0;
  return max.data;
}

double _top;
double _left;

List<Color> gradientColors = [
  Colors.blue,
];

@override
Widget build(BuildContext context) {
  //UNTUK MENDAPATKAN UKURAN LAYAR
  double tWidht = MediaQuery.of(context).size.width;
  double tHeight = MediaQuery.of(context).size.height;

  //TEMPAT MENGUBAH POSISI LABEL
  //SESUAIKAN DENGAN KETINGGIAN SUMBU Y
  //
  //ACUAN PENGUBAHAN
  //0.00x DARI UKURAN LAYAR AGAR RESPONSIVE DI SEMUA HP
  switch (widget.judul) {
    case ('pH'):
      _top = tHeight * 0.007;
      _left = tWidht * 0.04;
      break;
    // KODINGAN INI PAKAI JIKA CUSTOMISASI BERBEDA DARI DEFAULT

    case ('Turbidity'):
      _top = tHeight * 0.007;
      _left = tWidht * 0.015;
      break;
    case ('Temperature'):
      _top = tHeight * 0.007;
      _left = tWidht * 0.015;
      break;
    case ('Humidity'):
      _top = tHeight * 0.007;
      _left = tWidht * 0.015;
      break;
    //DEFAULT DI PAKAI UNTUK POSISI YANG SAMA
    default:
      _top = tHeight * 0.007;
      _left = tWidht * 0.015;
  }
```

```
return Card(
  color: Colors.grey[200],
  margin: EdgeInsets.symmetric(horizontal: 11, vertical: 16),
  shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.spaceAround,
    children: <Widget>[
      Card(
        //WARNA CARD
        color: Colors.white,
        shape:
            RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
        //POSISI CARD
        margin: EdgeInsets.symmetric(horizontal: 13, vertical: 5),
        child: Column(
          children: <Widget>[
            Stack(
              children: <Widget>[
                // Card(
                //   color: Colors.lightBlueAccent.withOpacity(0.9),
                Row(
                  mainAxisAlignment: MainAxisAlignment.start,
                  children: <Widget>[
                    Expanded(
                      child: Container(
                        //UKURAN CHART
                        //UBAH 0.8 DAN 0.72 SESUAI KEMAUAN
                        //MAKSUDNYA UKURAN LAYAR DI KALIKAN DENGAN 0.XX
                        width: MediaQuery.of(context).size.width * 0.82,
                        height: MediaQuery.of(context).size.height * 0.72,
                        decoration: const BoxDecoration(),
                        child: Padding(
                          padding: const EdgeInsets.only(
                              right: 20.0,
                              left: 20.0,
                              top: 35,
                              bottom: 120),
                          child: LineChart(
                            mainData(),
                          ),
                        ),
                      ),
                    ),
                  ],
                ),
                // ),
                Positioned(
                    top: _top,
                    left: _left,
                    child: Material(
                        color: Colors.transparent,
                        child: Text(widget.judul,
                            style: TextStyle(color: Colors.black)))),
                Positioned(
                left: tWidht * 0.79,
                top: tHeight * 0.565,
                child: Material(
                    color: Colors.transparent,
                    child: Text('Time',
                        style: TextStyle(color: Colors.black)
                        //fontWeight: FontWeight.bold),
                        ))),
            Positioned(
              //POSISI HISTORY
              top: tHeight * 0.62,
              left: tWidht * 0.29,
              child: MaterialButton(
                minWidth: MediaQuery.of(context).size.width * 0.3,
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(16),
                ),
                onPressed: () {
                  Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) => Wrapper(
                              judul: widget.judul,
                              param: widget.param,
                              page: 'details')));
                },
                padding: EdgeInsets.symmetric(
                  vertical: 5,
                ),
                color: Colors.blue[900],
                child: Text(
                  'History',
                  style: TextStyle(
                    fontSize: 16,
                  ),
                ),
                textColor: Colors.white,
              ),
            ),
          ],
        ),
      ],
    ),
  ),

  // Expanded(child: SizedBox()),
 ],
),
);
}

LineChartData mainData() {
  return LineChartData(
    gridData: FlGridData(
  show: false,
  drawVerticalLine: false,
),
titlesData: FlTitlesData(
    show: true,
    leftTitles: SideTitles(
        textStyle: const TextStyle(
            color: Colors.black,
            //fontWeight: FontWeight.bold,
            fontSize: 13),
        showTitles: true,
        reservedSize: 28,
        margin: 12,
        interval: (maxVal(widget.lists) * 1.5) / 5),
    bottomTitles: SideTitles(
        textStyle: const TextStyle(
            color: Colors.black,
            //fontWeight: FontWeight.bold,
            fontSize: 12),
        showTitles: true,
        getTitles: (value) {
          switch (value.toInt()) {
            case 1:
              return DateFormat('H:mm:s').format(
                  DateTime.fromMillisecondsSinceEpoch(
                      widget.lists[0].timestamp));
            case 15:
              return DateFormat('H:mm:s').format(
                  DateTime.fromMillisecondsSinceEpoch(
                      widget.lists[14].timestamp));
            case 28:
              return DateFormat('H:mm:s').format(
                  DateTime.fromMillisecondsSinceEpoch(
                      widget.lists[29].timestamp));
          }
          return '';
        })),
borderData: FlBorderData(
    show: false,
    border: Border.all(color: const Color(0xaaddaaff), width: 1)),
minX: 0,
maxX: widget.lists.length.toDouble(),
minY: minVal(widget.lists) * 1.5,
maxY: maxVal(widget.lists) * 1.5,
lineBarsData: [
  LineChartBarData(
    colors: [
      Colors.redAccent,
    ],
    spots: widget.lists.asMap().entries.map((e) {
      return FlSpot(e.key.toDouble(), e.value.data.toDouble());
    }).toList(),
    isCurved: true,
    preventCurveOverShooting: true,
          curveSmoothness: 0,
          dotData: FlDotData(show: true),
          barWidth: 4,
          isStrokeCapRound: false,
          belowBarData: BarAreaData(
            show: true,
            colors:
                gradientColors.map((color) => color.withOpacity(0.8)).toList(),
          ),
        ),
      ],
    );
  }
}
```

d) detail_screen.dart

```
import 'package:aquamonitoring/providers/data_provider.dart';
import 'package:flutter/material.dart';
import 'package:aquamonitoring/models/lit_data.dart';
import 'package:flutter_holo_date_picker/flutter_holo_date_picker.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:intl/intl.dart';
import 'package:provider/provider.dart';
//import 'package:time_formatter/time_formatter.dart';

import '../models/lit_data.dart';

class DetailScreen extends StatefulWidget {
  DetailScreen({Key key, this.lists, this.judul}) : super(key: key);
  final List<LitData> lists;
  final String judul;
  @override
  _DetailScreenState createState() => _DetailScreenState();
}

class _DetailScreenState extends State<DetailScreen> {
  final scaffoldKey = GlobalKey<ScaffoldState>();

  var date = DateTime.now();
  var min;
  var max;
  var unit;
  var appbarTitle;

  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    double tWidht = MediaQuery.of(context).size.width;
    //double tHeight  = MediaQuery.of(context).size.height;
```

```
var index = 0;
var dataProvider = Provider.of<DataProvider>(context);
switch (widget.judul) {
  //GANTI JUDUL DI HISTORY
  //GANTI UNIT
  //HISTORY UNIT CHANGE
  //APPBAR ganti appBarTitle
  case 'Temperature':
    min = dataProvider.minTemp;
    max = dataProvider.maxTemp;
    unit = '°C';
    appbarTitle = 'Temperature';

    break;
  case 'Turbidity':
    min = dataProvider.minTurb;
    max = dataProvider.maxTurb;
    unit = 'NTU';
    appbarTitle = 'Turbidity';
    break;
  case 'Humidity':
    min = dataProvider.minHum;
    max = dataProvider.maxHum;
    unit = '%';
    appbarTitle = 'Humidity';
    break;
  case 'pH':
    min = dataProvider.minPh;
    max = dataProvider.maxPh;
    unit = 'pH';
    appbarTitle = 'pH';
    break;
  default:
    min = -200;
    max = 300;
}

print('min: $min');
print('max: $max');

List<LitData> list = [];
widget.lists.forEach((element) {
  var time = DateTime.fromMillisecondsSinceEpoch(element.timestamp);
  if (time.day == date.day &&
      time.month == date.month &&
      date.year == time.year) {
    list.add(element);
  }
});

print(list.length);

return Scaffold(
key: scaffoldKey,
backgroundColor: Colors.blue[500],
//FONTCHANGE appbar
appBar: AppBar(
    title: Text(appbarTitle,
        style: GoogleFonts.droidSans(
            textStyle: TextStyle(
          fontWeight: FontWeight.bold,
        )))),
body: Container(
  margin: EdgeInsets.all(10),
  padding: EdgeInsets.all(10),
  child: Column(
    children: [
      MaterialButton(
        onPressed: () {},
        child: Text(
          DateFormat('MMMM, d yyyy').format(date).toString(),
          style: TextStyle(
            color: Colors.white,
            fontSize: 20,
            //fontWeight: FontWeight.bold,
          ),
        ),
      ),
      RaisedButton(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(16),
        ),
        color: Colors.blue[900],
        child: Text(
          "Change Date",
          style: TextStyle(
            color: Colors.white,
            fontSize: 15,
            fontWeight: FontWeight.bold,
          ),
        ),
        onPressed: () async {
          var datePicked = await DatePicker.showSimpleDatePicker(
            context,
            initialDate: DateTime(DateTime.now().year,
                DateTime.now().month, DateTime.now().day),
            firstDate: DateTime(2020),
            lastDate: DateTime(DateTime.now().year + 1),
            dateFormat: "dd-MMMM-yyyy",
            locale: DateTimePickerLocale.en_us,
            looping: true,
          );

          final snackBar = SnackBar(
              content: Text(
                  "Date Successfully Picked ${DateFormat('MMMM, d
yyyy').format(datePicked)}"));
```

```
    scaffoldKey.currentState.showSnackBar(snackBar);

    setState(() {
      date = datePicked;
    });
  },
),
SizedBox(
  height: 10,
),
Expanded(
  child: Column(
    children: <Widget>[
      Card(
          child: Container(
        height: 40,
        child: Row(
          mainAxisAlignment: MainAxisAlignment.start,
          children: <Widget>[
            Padding(
              padding: EdgeInsets.only(left: tWidht * 0.06),
              child: Text(
                'No',
                style: TextStyle(fontWeight: FontWeight.bold),
              ),
            ),
            Padding(
              padding: EdgeInsets.only(left: tWidht * 0.08),
              child: Text('Date',
                  style: TextStyle(fontWeight: FontWeight.bold)),
            ),
            Padding(
              padding: EdgeInsets.only(left: tWidht * 0.29),
              child: Text('Time',
                  style: TextStyle(fontWeight: FontWeight.bold)),
            ),
            Padding(
              padding: EdgeInsets.only(left: tWidht * 0.12),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: <Widget>[
                  Text(unit,
                      style:
                          TextStyle(fontWeight: FontWeight.bold)),
                ],
              ),
            )
          ],
        ),
      )

          // ListTile(
          //   title: Text(
          //     'No\t\t\t\t\t\tDate\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\tTime',
          //     style: TextStyle(
          //       fontWeight: FontWeight.bold,
          //     ),
          //   ),
          //   trailing: Text(
          //     unit,
          //     style: TextStyle(
          //       fontWeight: FontWeight.bold,
          //     ),
          //   ),
          // ),
          ),
      list.length == 0
          ? Container(
              margin: EdgeInsets.only(
                top: 20,
              ),
              child: Center(
                child: Text(
                  'No Data Available',
                  style: TextStyle(
                    color: Colors.white,
                    fontWeight: FontWeight.bold,
                    fontStyle: FontStyle.italic,
                  ),
                ),
              ),
            )
          : Expanded(
              child: ListView(
                children: ListTile.divideTiles(
                  color: Colors.grey,
                  tiles: list.map((data) {
                    index++;
                    var textStyle = TextStyle(
                        color: Colors.black, fontSize: 14);

                    if (data.data < min || data.data > max) {
                      textStyle = TextStyle(
                          color: Colors.red, fontSize: 14);
                    }

                    return Card(
                        margin: EdgeInsets.all(5),
                        child: Container(
                          width: MediaQuery.of(context).size.width *
                              0.8,
                          height: 40,
                          child: Row(
                            mainAxisAlignment:
                                MainAxisAlignment.spaceBetween,
                            children: <Widget>[
                              SizedBox(
                                          width: 2,
                                        ),
                                        Padding(
                                          padding: EdgeInsets.only(
                                              right: tWidht * 0.05),
                                          child: Text(
                                            "$index\t\t\t\t\t\t ${DateFormat('MMMM, d
yyyy').format(DateTime.fromMillisecondsSinceEpoch(data.timestamp))}",
                                            style: textStyle,
                                          ),
                                        ),
                                        Padding(
                                          padding: EdgeInsets.only(
                                              right: tWidht * 0.03),
                                          child: Text(

'\t\t\t\t\t\t\t\t${DateFormat("H:mm:s").format(DateTime.fromMillisecondsSinceEpoch(data.timestamp)).toStri
ng()}',
                                            style: textStyle,
                                          ),
                                        ),
                                        Padding(
                                          padding: EdgeInsets.only(
                                              bottom: 3,
                                              right: tWidht * 0.06),
                                          child: Text(
                                            data.data.toStringAsFixed(2),
                                            style: textStyle,
                                          ),
                                        ),
                                      ],
                                    ),
                                  )
                                  // ListTile(
                                  //   leading:
                                  //   Column(
                                  //     crossAxisAlignment: CrossAxisAlignment.end,
                                  //     mainAxisAlignment:
                                  //         MainAxisAlignment.center,
                                  //     children: <Widget>[
                                  //       Text(
                                  //         "$index\t\t\t\t ${DateFormat('MMMM, d
yyyy').format(DateTime.fromMillisecondsSinceEpoch(data.timestamp))}",
                                  //         style: textStyle,
                                  //       ),
                                  //     ],
                                  //   ),
                                  //   title: Column(
                                  //    crossAxisAlignment: CrossAxisAlignment.center,
                                  //     mainAxisAlignment:
                                  //         MainAxisAlignment.center,
                                  //     children: <Widget>[
                                  //       Text(
                                  //
```

```
'\t\t\t\t\t\t\t\t\t${DateFormat("H:mm:s").format(DateTime.fromMillisecondsSinceEpoch(data.timestamp)).toSt
ring()}',
                                              //        style: textStyle,
                                              //      ),
                                              //    ],
                                              //   ),
                                              //   trailing: Column(
                                              //    crossAxisAlignment:
                                              //        CrossAxisAlignment.start,
                                              //    mainAxisAlignment:
                                              //        MainAxisAlignment.start,
                                              //    children: <Widget>[
                                              //      SizedBox(height: 17,),
                                              //      Text(
                                              //        data.data.toStringAsFixed(2),
                                              //        style: textStyle,
                                              //      ),
                                              //    ],
                                              //   ),
                                              // ),
                                              );
                                          }).toList(),
                                        ).toList(),
                                      ),
                                    ),
                              ],
                            ),
                          ),
                        ],
                      ),
                    ),
                  );
                }
              }
```

e) home.dart

```
import 'package:aquamonitoring/fcm.dart';
import 'package:aquamonitoring/providers/data_provider.dart';
import 'package:aquamonitoring/providers/screen_provider.dart';
import 'package:customgauge/customgauge.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:provider/provider.dart';

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    var data = Provider.of<DataProvider>(context);
    var screenProvider = Provider.of<ScreenProvider>(context);

    return StreamBuilder(
        stream: FirebaseDatabase.instance.reference().child('AquMoSys').onValue,
                        builder: (context, snapshot) {
                          print(snapshot.data);
                          if (snapshot.hasData) {
                            DataSnapshot dataValues = snapshot.data.snapshot;

                            Map<dynamic, dynamic> values = dataValues.value;

                            List daftar = values.values.toList();

                            daftar.sort((a, b) {
                              return b['timestamp'].compareTo(a['timestamp']);
                            });

                            var lastData = daftar[0];
                            print(lastData);

                            print(DateTime.fromMillisecondsSinceEpoch(daftar[0]['timestamp']));

                            var notifText = '';

                            if (lastData['Temperature'] < data.minTemp) {
                              notifText = notifText +
                                  ('\nTemperature (${lastData['Temperature'].toStringAsFixed(2)}) below the
threshold. ');
                            } else if (lastData['Temperature'] > data.maxTemp) {
                              notifText = notifText +
                                  ('\nTemperature (${lastData['Temperature'].toStringAsFixed(2)}) above the
threshold. ');
                            }

                            if (lastData['Turbidity'] < data.minTurb) {
                              notifText = notifText +
                                  ('\nTurbidity (${lastData['Turbidity'].toStringAsFixed(2)}) below the threshold. ');
                            } else if (lastData['Turbidity'] > data.maxTurb) {
                              notifText = notifText +
                                  ('\nTurbidity (${lastData['Turbidity'].toStringAsFixed(2)}) above the threshold. ');
                            }
                            if (lastData['pH'] < data.minPh) {
                              notifText = notifText +
                                  ('\npH (${lastData['pH'].toStringAsFixed(2)}) below the threshold. ');
                            } else if (lastData['pH'] > data.maxPh) {
                              notifText = notifText +
                                  ('\npH (${lastData['pH'].toStringAsFixed(2)}) above the threshold. ');
                            }

                            if (lastData['Humidity'] < data.minHum) {
                              notifText = notifText +
                                  ('\nHumidity (${lastData['Humidity']}) below the threshold. ');
                            } else if (lastData['Humidity'] > data.maxHum) {
                              notifText = notifText +
                                  ('\nHumidity (${lastData['Humidity']}) above the threshold. ');
                            }

                            if (notifText.isNotEmpty) {
```

```
  sendNotification(notifText);
}

return Container(
  width: MediaQuery.of(context).size.width,
  height: MediaQuery.of(context).size.height,
  padding: EdgeInsets.all(10),
  child: SingleChildScrollView(
    child: Wrap(
      crossAxisAlignment: WrapCrossAlignment.center,
      alignment: WrapAlignment.spaceAround,
      runAlignment: WrapAlignment.spaceAround,
      runSpacing: 10,
      children: [
        InkWell(
          onTap: () {
            screenProvider.setCurrentIndex(0);
          },
          child: Container(
            width: MediaQuery.of(context).size.width * 0.45,
            height: MediaQuery.of(context).size.height * 0.375,
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.circular(16),
            ),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                Text(
                  "Temperature",
                  style: TextStyle(
                    fontSize: 14,
                    fontWeight: FontWeight.bold,
                    // color: Colors.white,
                  ),
                  textAlign: TextAlign.center,
                ),
                SizedBox(
                  height: 10,
                ),
                CustomGauge(
                  // defaultSegmentColor: Colors.white,
                  // needleColor: Colors.white,
                  startMarkerStyle: TextStyle(
                    // color: Colors.white,
                    color: Colors.black,
                  ),
                  endMarkerStyle: TextStyle(
                    // color: Colors.white,
                    color: Colors.black,
                  ),
                  gaugeSize:
                      MediaQuery.of(context).size.width * 0.4,
      minValue: -55,
      maxValue: 125,
      segments: [
        GaugeSegment(
          'Low',
          (125 - -55) / 3,
          Colors.red,
        ),
        GaugeSegment(
          'Medium',
          (125 - -55) / 3,
          Colors.orange,
        ),
        GaugeSegment(
          'High',
          (125 - -55) / 3,
          Colors.green,
        ),
      ],
      currentValue: lastData['Temperature'].toDouble(),
      valueWidget: Text(
        "${lastData['Temperature'].toDouble().toStringAsFixed(1)}",
        style: TextStyle(
          fontSize: 12,
          fontWeight: FontWeight.bold,
          // color: Colors.white,
        ),
        textAlign: TextAlign.center,
      ),
    ),
    Text(
      DateFormat('MMMM, d yyyy')
          .format(DateTime.fromMillisecondsSinceEpoch(
              lastData['timestamp']))
          .toString(),
      style: TextStyle(
        fontSize: 12,
        // color: Colors.white,
      ),
      textAlign: TextAlign.center,
    ),
    Text(
      DateFormat('HH:mm:ss')
          .format(DateTime.fromMillisecondsSinceEpoch(
              lastData['timestamp']))
          .toString(),
      style: TextStyle(
        fontSize: 12,
        // color: Colors.white,
      ),
      textAlign: TextAlign.center,
    ),
  ],
),
  ),
),
InkWell(
  onTap: () {
    screenProvider.setCurrentIndex(1);
  },
  child: Container(
    width: MediaQuery.of(context).size.width * 0.45,
    height: MediaQuery.of(context).size.height * 0.375,
    decoration: BoxDecoration(
      color: Colors.white,
      borderRadius: BorderRadius.circular(16),
    ),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.center,
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Text(
          "Turbidity",
          style: TextStyle(
            fontSize: 14,
            color: Colors.black,
            fontWeight: FontWeight.bold,
          ),
          textAlign: TextAlign.center,
        ),
        SizedBox(
          height: 10,
        ),
        CustomGauge(
          defaultSegmentColor: Colors.black,
          needleColor: Colors.black,
          startMarkerStyle: TextStyle(
            color: Colors.black,
          ),
          endMarkerStyle: TextStyle(
            color: Colors.black,
          ),
          gaugeSize:
              MediaQuery.of(context).size.width * 0.4,
          minValue: 0,
          maxValue: 100,
          segments: [
            GaugeSegment(
              'Low',
              (100 - 0) / 3,
              Colors.red,
            ),
            GaugeSegment(
              'Medium',
              (100 - 0) / 3,
              Colors.orange,
            ),
            GaugeSegment(
              'High',
              (100 - 0) / 3,
              Colors.green,
            ),
          ],
          currentValue: lastData['Turbidity'].toDouble(),
          valueWidget: Text(
            "${lastData['Turbidity'].toDouble().toStringAsFixed(1)}",
            style: TextStyle(
              fontSize: 12,
              color: Colors.black,
              fontWeight: FontWeight.bold,
            ),
            textAlign: TextAlign.center,
          ),
        ),
        Text(
          DateFormat('MMMM, d yyyy')
              .format(DateTime.fromMillisecondsSinceEpoch(
                  lastData['timestamp']))
              .toString(),
          style: TextStyle(
            fontSize: 12,
            color: Colors.black,
          ),
          textAlign: TextAlign.center,
        ),
        Text(
          DateFormat('HH:mm:ss')
              .format(DateTime.fromMillisecondsSinceEpoch(
                  lastData['timestamp']))
              .toString(),
          style: TextStyle(
            fontSize: 12,
            color: Colors.black,
          ),
          textAlign: TextAlign.center,
        ),
      ],
    ),
  ),
  ),
  InkWell(
    onTap: () {
    screenProvider.setCurrentIndex(3);
    },
    child: Container(
    width: MediaQuery.of(context).size.width * 0.45,
    height: MediaQuery.of(context).size.height * 0.375,
    decoration: BoxDecoration(
      color: Colors.white,
      borderRadius: BorderRadius.circular(16),
    ),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.center,
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Text(
          "pH",
          style: TextStyle(
            fontSize: 14,
            color: Colors.black,
            fontWeight: FontWeight.bold,
          ),
          textAlign: TextAlign.center,
        ),
        SizedBox(
          height: 10,
        ),
        CustomGauge(
          defaultSegmentColor: Colors.black,
          needleColor: Colors.black,
          startMarkerStyle: TextStyle(
            color: Colors.black,
          ),
          endMarkerStyle: TextStyle(
            color: Colors.black,
          ),
          gaugeSize:
              MediaQuery.of(context).size.width * 0.4,
          minValue: 0,
          maxValue: 14,
          segments: [
            GaugeSegment(
              'Low',
              (14 - 0) / 3,
              Colors.red,
            ),
            GaugeSegment(
              'Medium',
              (14 - 0) / 3,
              Colors.orange,
            ),
            GaugeSegment(
              'High',
              (14 - 0) / 3,
              Colors.green,
            ),
          ],
          currentValue: lastData['pH'].toDouble(),
          valueWidget: Text(
            "${lastData['pH'].toDouble().toStringAsFixed(1)}",
            style: TextStyle(
              fontSize: 12,
              color: Colors.black,
              fontWeight: FontWeight.bold,
            ),
            textAlign: TextAlign.center,
                ),
              ),
              Text(
                DateFormat('MMMM, d yyyy')
                    .format(DateTime.fromMillisecondsSinceEpoch(
                        lastData['timestamp']))
                    .toString(),
                style: TextStyle(
                  fontSize: 12,
                  color: Colors.black,
                ),
                textAlign: TextAlign.center,
              ),
              Text(
                DateFormat('HH:mm:ss')
                    .format(DateTime.fromMillisecondsSinceEpoch(
                        lastData['timestamp']))
                    .toString(),
                style: TextStyle(
                  fontSize: 12,
                  color: Colors.black,
                ),
                textAlign: TextAlign.center,
              ),
            ],
          ),
        ),
      ),
      InkWell(
        onTap: () {
          screenProvider.setCurrentIndex(4);
        },
        child: Container(
          width: MediaQuery.of(context).size.width * 0.45,
          height: MediaQuery.of(context).size.height * 0.375,
          decoration: BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.circular(16),
          ),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Text(
                "Humidity",
                style: TextStyle(
                  fontSize: 14,
                  color: Colors.black,
                  fontWeight: FontWeight.bold,
                ),
                textAlign: TextAlign.center,
              ),
              SizedBox(
                height: 10,
      ),
      CustomGauge(
        defaultSegmentColor: Colors.black,
        needleColor: Colors.black,
        startMarkerStyle: TextStyle(
          color: Colors.black,
        ),
        endMarkerStyle: TextStyle(
          color: Colors.black,
        ),
        gaugeSize:
            MediaQuery.of(context).size.width * 0.4,
        minValue: 0,
        maxValue: 100,
        segments: [
          GaugeSegment(
            'Low',
            (100 - 0) / 3,
            Colors.red,
          ),
          GaugeSegment(
            'Medium',
            (100 - 0) / 3,
            Colors.orange,
          ),
          GaugeSegment(
            'High',
            (100 - 0) / 3,
            Colors.green,
          ),
        ],
        currentValue: lastData['Humidity'].toDouble(),
        valueWidget: Text(
          "${lastData['Humidity'].toDouble().toStringAsFixed(1)}",
          style: TextStyle(
            fontSize: 12,
            color: Colors.black,
            fontWeight: FontWeight.bold,
          ),
          textAlign: TextAlign.center,
        ),
      ),
      Text(
        DateFormat('MMMM, d yyyy')
            .format(DateTime.fromMillisecondsSinceEpoch(
                lastData['timestamp']))
            .toString(),
        style: TextStyle(
          fontSize: 12,
          color: Colors.black,
        ),
        textAlign: TextAlign.center,
      ),
      Text(
                          DateFormat('HH:mm:ss')
                            .format(DateTime.fromMillisecondsSinceEpoch(
                              lastData['timestamp']))
                            .toString(),
                          style: TextStyle(
                           fontSize: 12,
                           color: Colors.black,
                          ),
                          textAlign: TextAlign.center,
                         ),
                        ],
                       ),
                      ),
                     ),
                    ],
                   ),
                  ),
                 );
                }

                return Center(
                 child: CircularProgressIndicator(),
                );
               });
          }
         }
```

f) notification_screen.dart

```
import 'package:aquamonitoring/fcm.dart';
import 'package:aquamonitoring/providers/auth_provider.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

class NotificationScreen extends StatefulWidget {
 @override
 _NotificationScreenState createState() => _NotificationScreenState();
}

class _NotificationScreenState extends State<NotificationScreen> {
 @override
 Widget build(BuildContext context) {
  var authProvider = Provider.of<AuthProvider>(context);

  Future getNotif() async {
   var tempLocal = true;
   var tempEmail = true;

   await Firestore.instance
     .collection('tokens')
     .document(getToken())
     .get()
```

```
      .then((value) {
    if (value.data != null) {
      tempLocal = value.data['value'];
    }
  });

  if (authProvider.currentEmail.isNotEmpty) {
    await Firestore.instance
        .collection('emails')
        .document(authProvider.currentEmail)
        .get()
        .then((value) {
      if (value != null) {
        tempEmail = value.data['value'];
      }
    }).catchError((onError) {});

    print(tempEmail);
  }

  print('tempLocal: $tempLocal');
  print('tempEmail: $tempEmail');

  return {
    'local': tempLocal,
    'email': tempEmail,
  };
}

return Scaffold(
  backgroundColor: Colors.blue[500],
  //FONTCHANGE appbar
  appBar: AppBar(
    title: Text('Notifications',
        style: GoogleFonts.droidSans(
            textStyle: TextStyle(
          fontWeight: FontWeight.bold,
        ))),
  ),
  body: FutureBuilder(
    future: getNotif(),
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.done) {
        var local = snapshot.data['local'];
        var email = snapshot.data['email'];

        return Padding(
          padding: const EdgeInsets.all(20.0),
          child: Column(
            children: [
              Card(
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(16)),
                color: Colors.white,
                child: SwitchListTile(
                  value: local,
                  title: Text(
                    'Local Notification',
                    style: TextStyle(
                        // color: Colors.white,
                        ),
                  ),
                  onChanged: (value) async {
                    await Firestore.instance
                        .collection('tokens')
                        .document(getToken())
                        .updateData({'value': value});
                    setState(() {});
                  },
                ),
              ),
              (authProvider.currentEmail.isEmpty)
                  ? SizedBox()
                  : Card(
                      shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(16)),
                      color: Colors.white,
                      child: SwitchListTile(
                        value: email,
                        title: Text(
                          'Email Notification',
                          style: TextStyle(
                              // color: Colors.white,
                              ),
                        ),
                        onChanged: (value) async {
                          await Firestore.instance
                              .collection('emails')
                              .document(authProvider.currentEmail)
                              .updateData({'value': value});
                          setState(() {});
                        },
                      ),
                    ),
              Expanded(
                child: SizedBox(),
              ),
              // MaterialButton(
              //   minWidth: double.infinity,
              //   color: kPrimary,
              //   padding: EdgeInsets.symmetric(
              //     vertical: 10,
              //   ),
              //   child: Text(
              //     'Save',
              //     style: TextStyle(
              //       color: Colors.white,
              //       fontSize: 20,
                              //    ),
                              //   ),
                              //   onPressed: () {},
                              // ),
                            ],
                          ),
                        );
                      }
                      return Center(
                        child: CircularProgressIndicator(),
                      );
                    },
                  ),
                );
               }
              }
```

g) setting_screen.dart

```
import 'package:aquamonitoring/providers/data_provider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_range_slider/flutter_range_slider.dart' as frs;
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

class SettingScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    var dataProvider = Provider.of<DataProvider>(context);

    return SettingInitial(dataProvider);
  }
}

class SettingInitial extends StatefulWidget {
  final DataProvider _dataProvider;
  SettingInitial(this._dataProvider);

  @override
  _SettingInitialState createState() => _SettingInitialState();
}

class _SettingInitialState extends State<SettingInitial> {
  final _scaffoldKey = GlobalKey<ScaffoldState>();
  double minTemp;
  double maxTemp;
  double minTurb;
  double maxTurb;
  double minPh;
  double maxPh;
  double minHum;
  double maxHum;

  @override
```

```
void initState() {
  super.initState();

  print('init');
  print(widget._dataProvider.minTemp);
  minTemp = widget._dataProvider.minTemp;
  maxTemp = widget._dataProvider.maxTemp;
  minTurb = widget._dataProvider.minTurb;
  maxTurb = widget._dataProvider.maxTurb;
  minPh = widget._dataProvider.minPh;
  maxPh = widget._dataProvider.maxPh;
  minHum = widget._dataProvider.minHum;
  maxHum = widget._dataProvider.maxHum;
  setState(() {});
}

@override
Widget build(BuildContext context) {
  minTemp = widget._dataProvider.minTemp;
  maxTemp = widget._dataProvider.maxTemp;
  minTurb = widget._dataProvider.minTurb;
  maxTurb = widget._dataProvider.maxTurb;
  minPh = widget._dataProvider.minPh;
  maxPh = widget._dataProvider.maxPh;
  minHum = widget._dataProvider.minHum;
  maxHum = widget._dataProvider.maxHum;

  changeFunction(int index, double min, double max) {
    print(index);
    print(min);
    print(max);
    switch (index) {
      case 0:
        minTemp = min;
        maxTemp = max;
        break;
      case 1:
        minTurb = min;
        maxTurb = max;
        break;
      case 2:
        minPh = min;
        maxPh = max;
        break;
      case 3:
        minHum = min;
        maxHum = max;
        break;
      default:
    }

    // setState(() {});
  }
```

```
Future<void> updated() async {
  return showDialog<void>(
    context: context,
    barrierDismissible: true, // user must tap button!
    builder: (BuildContext context) {
      return AlertDialog(
        shape:
            RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
        content: SingleChildScrollView(
          child: ListBody(
            children: <Widget>[
              Text(
                'Threshold updated',
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                ),
              ),
            ],
          ),
        ),
        actions: <Widget>[
          FlatButton(
            child: Text('Close'),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
        ],
      );
    },
  );
}

Future<void> dialog() async {
  return showDialog<void>(
    context: context,
    barrierDismissible: true, // user must tap button!
    builder: (BuildContext context) {
      return AlertDialog(
        shape:
            RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
        content: SingleChildScrollView(
          child: ListBody(
            children: <Widget>[
              Text(
                'Do you want to save the changes?',
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                ),
              ),
            ],
          ),
        ),
        actions: <Widget>[
          FlatButton(
            child: Text('Yes'),
            onPressed: () {
              Navigator.of(context).pop();
              updated();
            },
          ),
          FlatButton(
            child: Text('No'),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
        ],
      );
    },
  );
}

handleSubmit() async {
  await widget._dataProvider.setData(
    (minTemp),
    (maxTemp),
    (minTurb),
    (maxTurb),
    (minPh),
    (maxPh),
    (minHum),
    (maxHum),
  );

  dialog();
}

return Scaffold(
  key: _scaffoldKey,
  backgroundColor: Colors.blue[500],
  //FONTCHANGE appbar
  appBar: AppBar(
      title: Text('Threshold Setting',
          style: GoogleFonts.droidSans(
              textStyle: TextStyle(
            fontWeight: FontWeight.bold,
          )))),
  body: Container(
    padding: EdgeInsets.all(20),
    child: Column(
      children: <Widget>[
        Threshold(
          0,
          'temperature',
          'Temperature',
          minTemp,
          maxTemp,
  changeFunction,
),
// Threshold(
//   'temperature',
//   'Temperature Maximum',
//   maxTemp,
// ),
SizedBox(
  height: 10,
),
Threshold(
  1,
  'turbiditiy',
  'Turbidity',
  minTurb,
  maxTurb,
  changeFunction,
),
// Threshold(
//   'turbiditiy',
//   'Turbidity Maximum',
//   maxTurb,
// ),
SizedBox(
  height: 10,
),
Threshold(
  2,
  'ph-1',
  'pH',
  minPh,
  maxPh,
  changeFunction,
),
// Threshold(
//   'ph-1',
//   'pH Maximum',
//   maxPh,
// ),
SizedBox(
  height: 10,
),
Threshold(
  3,
  'humidity',
  'Humidity',
  minHum,
  maxHum,
  changeFunction,
),
// Threshold(
//   'humidity',
//   'Humidity Maximum',
//   maxHum,
          // ),
          Expanded(
            child: SizedBox(),
          ),

          MaterialButton(
            minWidth: MediaQuery.of(context).size.width * 0.3,
            onPressed: () {
              handleSubmit();
            },
            color: Colors.blue[900],
            padding: EdgeInsets.symmetric(
              vertical: 10,
            ),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(16),
            ),
            child: Text(
              'Save',
              style: TextStyle(
                color: Colors.white,
                fontSize: 15,
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
          SizedBox(
            height: 20,
          ),
        ],
      ),
    ),
  );
 }
}

class Threshold extends StatefulWidget {
  final index;
  final image;
  final title;
  final minValue;
  final maxValue;
  final function;

  Threshold(this.index, this.image, this.title, this.minValue, this.maxValue,
      this.function);

  @override
  _ThresholdState createState() => _ThresholdState();
}

class _ThresholdState extends State<Threshold> {
  double minValue;
  double maxValue;
double min;
double max;

var division;

@override
void initState() {
  super.initState();
  minValue = widget.minValue;
  maxValue = widget.maxValue;

  switch (widget.index) {
    case 0:
      min = -55;
      max = 125;
      division = (max - min) * 2;
      break;
    case 1:
      min = 0;
      max = 100;
      division = (max - min) * 1;
      break;
    case 2:
      min = 0;
      max = 14;
      division = (max - min) * 10;
      break;
    case 3:
      min = 0;
      max = 100;
      division = (max - min) * 2;
      break;
    default:
  }
}

@override
Widget build(BuildContext context) {
  var format;
  return Container(
    padding: EdgeInsets.all(10),
    decoration: BoxDecoration(
      color: Colors.white,
      borderRadius: BorderRadius.circular(16),
    ),
    child: Row(
      children: <Widget>[
        Container(
          margin: EdgeInsets.only(
              // bottom: 20,
              ),
          padding: EdgeInsets.all(10),
          decoration: BoxDecoration(
            color: Colors.grey[100],
    borderRadius: BorderRadius.circular(20),
  ),
  child: Image.asset(
    'assets/icon/${widget.image}.png',
    width: 40,
    fit: BoxFit.fitWidth,
  ),
),
SizedBox(
  width: 10,
),
Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    Text(
      widget.title,
      style: TextStyle(
        fontSize: 16,
        color: Colors.black,
      ),
    ),
    SizedBox(
      height: 20,
    ),
    Row(
      children: <Widget>[
        Material(
            child: Text(
          'min',
          style: TextStyle(fontSize: 10),
        )),
        Container(
          height: 70,
          width: MediaQuery.of(context).size.width * 0.58,
          child: SliderTheme(
            //COLOR SLIDER
            data: SliderTheme.of(context).copyWith(
              overlayColor: Colors.blue[300].withOpacity(0.3),
              activeTrackColor: Colors.blue[300],
              inactiveTrackColor: Colors.redAccent,
              thumbColor: Colors.blue[900],
              trackHeight: 19.0,

              //valueIndicatorColor: const Color(0xFF0175c2),
            ),
            child: frs.RangeSlider(
              min: min,
              max: max,
              lowerValue: widget.minValue,
              upperValue: widget.maxValue,
              divisions: division.toInt(),
              showValueIndicator: true,
              valueIndicatorMaxDecimals: 1,
              valueIndicatorFormatter: (int index, double value) {
```

```
            String twoDecimals;
            switch (widget.title) {
              case ('Temperature'):
                {
                  format = ' \u00b0C';
                  twoDecimals = value.toStringAsFixed(1);
                }
                break;
              case ('Turbidity'):
                {
                  format = ' NTU';
                  twoDecimals = value.toStringAsFixed(0);
                }
                break;
              case ('pH'):
                {
                  format = ' pH';
                  twoDecimals = value.toStringAsFixed(1);
                }
                break;
              case ('Humidity'):
                {
                  format = ' %';
                  twoDecimals = value.toStringAsFixed(1);
                }
                break;
              default:
                twoDecimals = value.toStringAsFixed(2);
            }

            return '$twoDecimals $format';
          },
          onChanged:
              (double newLowerValue, double newUpperValue) {
            // setState(() {
            //   widget.minValue = newLowerValue;
            //   widget.maxValue = newUpperValue;
            // });
          },
          onChangeStart:
              (double startLowerValue, double startUpperValue) {
            print(
                'Started with values: $startLowerValue and $startUpperValue');
          },
          onChangeEnd:
              (double newLowerValue, double newUpperValue) {
            print(
                'Ended with values: $newLowerValue and $newUpperValue');
            widget.function(
                widget.index, newLowerValue, newUpperValue);
          },
        ),
      ),
    ),
    Material(
        child: Text(
      'max',
      style: TextStyle(fontSize: 10),
    )),
  ],
),

// Row(
//   crossAxisAlignment: CrossAxisAlignment.center,
//   children: [
//     MaterialButton(
//       onPressed: () {
//         setState(() {
//           widget.value.text =
//               (double.parse(widget.value.text) - 1).toString();
//         });
//       },
//       minWidth: 0,
//       padding: EdgeInsets.all(4),
//       color: kPrimary,
//       child: Icon(
//         Icons.remove,
//         color: Colors.white,
//       ),
//     ),
//     Padding(
//       padding: EdgeInsets.symmetric(
//         horizontal: 10,
//       ),
//       child: Container(
//         height: 20,
//         width: MediaQuery.of(context).size.width * 0.3,
//         child: TextFormField(
//           controller: widget.value,
//           onChanged: (value) {
//             if (!double.parse(value).isNaN) {
//               setState(() {
//                 widget.value.text = value;
//               });
//             }
//           },
//           keyboardType: TextInputType.number,
//           style: TextStyle(
//             color: Colors.white,
//             fontSize: 20,
//           ),
//           textAlign: TextAlign.center,
//           textAlignVertical: TextAlignVertical.center,
//           decoration: InputDecoration(),
//         ),
//       ),
//     ),
//     MaterialButton(
            //      onPressed: () {
            //        setState(() {
            //          widget.value.text =
            //              (double.parse(widget.value.text) + 1).toString();
            //        });
            //      },
            //      minWidth: 0,
            //      padding: EdgeInsets.all(4),
            //      color: kPrimary,
            //      child: Icon(
            //        Icons.add,
            //        color: Colors.white,
            //      ),
            //    ),
            //  ],
            // ),
           ],
         )
        ],
      ),
    );
  }
}
```

h) sign_in_screen.dart

```
import 'package:aquamonitoring/providers/auth_provider.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:email_validator/email_validator.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';

class SignInScreen extends StatefulWidget {
  @override
  _SignInScreenState createState() => _SignInScreenState();
}

class _SignInScreenState extends State<SignInScreen> {
  final formKey = GlobalKey<FormState>();
  final emailController = TextEditingController(text: '');

  @override
  Widget build(BuildContext context) {
    var authProvider = Provider.of<AuthProvider>(context);

    Future<void> loginDialog() async {
      return showDialog<void>(
        context: context,
        barrierDismissible: true, // user must tap button!
        builder: (BuildContext context) {
          return AlertDialog(
            shape:
                RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
        content: SingleChildScrollView(
          child: ListBody(
            children: <Widget>[
              Text('You have successfully signed in'),
            ],
          ),
        ),
        actions: <Widget>[
          FlatButton(
            child: Text('Close'),
            onPressed: () {
              Navigator.of(context).pop();
              Navigator.of(context).pop();
            },
          ),
        ],
      );
    },
  );
}

handleSubmit() async {
  await Firestore.instance
      .collection('emails')
      .document(emailController.text)
      .setData(
    {'email': emailController.text, 'value': true},
  );

  await authProvider.setUser(emailController.text);

  loginDialog();

  // Navigator.pop(context);
}

return Scaffold(
  backgroundColor: Colors.blue[500],

  //FONTCHANGE appbar
  appBar: AppBar(
    title: Text('Sign In',
        style: GoogleFonts.droidSans(
            textStyle: TextStyle(
          fontWeight: FontWeight.bold,
        ))),
  ),
  body: Padding(
    padding: EdgeInsets.all(10),
    child: Card(
      shape:
          RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
      child: Padding(
        padding: EdgeInsets.symmetric(
          horizontal: 20,
        ),
        child: Center(
          child: Form(
            key: formKey,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Icon(
                  Icons.email,
                  size: MediaQuery.of(context).size.width * 0.3,
                  color: Colors.blue[900],
                ),
                SizedBox(
                  height: 20,
                ),
                TextFormField(
                  controller: emailController,
                  validator: (value) {
                    if (value.isEmpty) {
                      return 'E-mail cannot be empty';
                    }
                    if (!EmailValidator.validate(emailController.text)) {
                      return 'Invalid E-mail Address';
                    }
                    return null;
                  },
                  decoration: InputDecoration(
                    labelText: 'Enter your E-mail Address',
                  ),
                ),
                SizedBox(
                  height: 20,
                ),
                MaterialButton(
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(16)),
                  onPressed: () {
                    if (formKey.currentState.validate()) {
                      handleSubmit();
                    }
                  },
                  minWidth: MediaQuery.of(context).size.width * 0.3,
                  color: Colors.blue[900],
                  padding: EdgeInsets.symmetric(
                    vertical: 10,
                  ),
                  child: Text(
                    'Sign In',
                    style: GoogleFonts.lato(
                      textStyle: TextStyle(
                        color: Colors.white,
                        fontSize: 15,
                        fontWeight: FontWeight.bold,
                            ),
                           ),
                          ),
                         ),
                        ],
                       ),
                      ),
                     ),
                    ),
                   ),
                  ),
                );
              }
            }
```

i) wrapper.dart

```
import 'package:flutter/material.dart';
import 'package:aquamonitoring/widgets/chart.dart';
import 'package:aquamonitoring/pages/detail_screen.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:aquamonitoring/models/lit_data.dart';

class Wrapper extends StatefulWidget {
  Wrapper({Key key, this.judul, this.param, this.page}) : super(key: key);

  final String judul;
  final String param;
  final String page;

  @override
  _WrapperState createState() => _WrapperState();
}

class _WrapperState extends State<Wrapper> {
  final dbRef = FirebaseDatabase.instance.reference().child('AquMoSys');
  @override
  Widget build(BuildContext context) {
    return StreamBuilder(
        stream: dbRef.onValue,
        builder: (context, AsyncSnapshot<Event> snap) {
          List<LitData> lists = [];

          final res = [];

          if (snap.hasData) {
            lists.clear();
            res.clear();
            DataSnapshot dataValues = snap.data.snapshot;

            Map<dynamic, dynamic> values = dataValues.value;

            List daftar = values.values.toList();
    daftar.sort((a, b) {
      return a['timestamp'].compareTo(b['timestamp']);
    });

    for (var item in daftar) {
      var prod = {
        'data': item[widget.param] != null
            ? item[widget.param].toDouble()
            : 0.0,
        'timestamp': item['timestamp']
      };

      res.add(prod);
    }

    // print(res);
    // print('test');

    res.reversed;

    List<LitData> listsObjcs =
        res.map((e) => LitData.fromJson(e)).toList();

    lists = listsObjcs.reversed.toList();

    var index = 0;
    List<LitData> chartList = [];
    lists.forEach((element) {
      index++;
      if (index <= 18) {
        chartList.add(element);
      }
    });

    lists = lists.reversed.toList();
    chartList = chartList.reversed.toList();

    print(DateTime.fromMillisecondsSinceEpoch(listsObjcs[0].timestamp));
    print(DateTime.fromMillisecondsSinceEpoch(
        listsObjcs[listsObjcs.length - 1].timestamp));

    if (widget.page == 'chart') {
      return new ChartWidget(
          lists: chartList, judul: widget.judul, param: widget.param);
    } else {
      return new DetailScreen(
          lists: lists.reversed.toList(), judul: widget.judul);
    }
  }

  return Center(
    child: CircularProgressIndicator(),
  );
});
```

```
 }
}
```

j) auth_provider.dart

```
import 'package:aquamonitoring/fcm.dart';
import 'package:flutter/widgets.dart';
import 'package:shared_preferences/shared_preferences.dart';

class AuthProvider with ChangeNotifier {
String _currentEmail = '';
String get currentEmail => _currentEmail;

String _currentToken = '';
String get currentToken => _currentToken;

getInitialUser() async {
SharedPreferences prefs = await SharedPreferences.getInstance();
_currentEmail = (prefs.getString('currentEmail') ?? '');
_currentToken = (prefs.getString('currentToken') ?? getToken() ?? '');
print('get initial user');
notifyListeners();
}

setUser(newEmail) async {
SharedPreferences prefs = await SharedPreferences.getInstance();
_currentEmail = newEmail;
await prefs.setString('currentEmail', newEmail);
print(_currentEmail);
notifyListeners();
}

setToken(newToken) async {
SharedPreferences prefs = await SharedPreferences.getInstance();
_currentToken = newToken;
await prefs.setString('currentToken', newToken);
print(_currentToken);
notifyListeners();
}
logout() async {
SharedPreferences prefs = await SharedPreferences.getInstance();
_currentEmail = '';
await prefs.setString('currentEmail', '');
notifyListeners();
}
}
```

k) data_provider.dart

```
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';

class DataProvider extends ChangeNotifier {
double _minTemp = -55;
double _maxTemp = 125;
double _minTurb = 0;
double _maxTurb = 100;
double _minPh = 0;
double _maxPh = 14;
double _minHum = 0;
double _maxHum = 100;

double get minTemp => _minTemp;
double get maxTemp => _maxTemp;
double get minTurb => _minTurb;
double get maxTurb => _maxTurb;
double get minPh => _minPh;
double get maxPh => _maxPh;
double get minHum => _minHum;
double get maxHum => _maxHum;

setData(double minTe, double maxTe, double minTu, double maxTu, double minP,
double maxP, double minH, double maxH) async {
print('minnte');
print(minTe);

_minTemp = minTe;
_maxTemp = maxTe;
```

```
_minTurb = minTu;
_maxTurb = maxTu;
_minPh = minP;
_maxPh = maxP;
_minHum = minH;
_maxHum = maxH;

SharedPreferences prefs = await SharedPreferences.getInstance();

await prefs.setDouble('minTemp', minTe);
await prefs.setDouble('maxTemp', maxTe);
await prefs.setDouble('minTurb', minTu);
await prefs.setDouble('maxTurb', maxTu);
await prefs.setDouble('minPh', minP);
await prefs.setDouble('maxPh', maxP);
await prefs.setDouble('minHum', minH);
await prefs.setDouble('maxHum', maxH);

print(prefs.getDouble('minTemp'));

notifyListeners();
}

getInitialData() async {
SharedPreferences prefs = await SharedPreferences.getInstance();
_minTemp = prefs.getDouble('minTemp') ?? _minTemp;
_maxTemp = prefs.getDouble('maxTemp') ?? _maxTemp;
_minTurb = prefs.getDouble('minTurb') ?? _minTurb;
_maxTurb = prefs.getDouble('maxTurb') ?? _maxTurb;
_minPh = prefs.getDouble('minPh') ?? _minPh;
_maxPh = prefs.getDouble('maxPh') ?? _maxPh;
_minHum = prefs.getDouble('minHum') ?? _minHum;
_maxHum = prefs.getDouble('maxHum') ?? _maxHum;
print('get initial data');
print(prefs.getDouble('minTemp'));
print(_minTemp);
print('yoi');
notifyListeners();
```

```
}
}
```

l) screen_provider.dart

```
import 'package:flutter/material.dart';

class ScreenProvider extends ChangeNotifier {
int _currentIndex = 2;

int get currentIndex => _currentIndex;

setCurrentIndex(int newIndex) {
_currentIndex = newIndex;
notifyListeners();
}
}
```

m) chart.dart

```
import 'package:aquamonitoring/pages/wrapper.dart';
import 'package:flutter/material.dart';
import 'package:fl_chart/fl_chart.dart';
import 'package:intl/intl.dart';
import 'package:aquamonitoring/models/lit_data.dart';

class ChartWidget extends StatefulWidget {
ChartWidget({Key key, this.lists, this.judul, this.param}) : super(key: key);

final List<LitData> lists;

final String judul;
final String param;

@override
_ChartWidgetState createState() => _ChartWidgetState();
}

class _ChartWidgetState extends State<ChartWidget> {
double maxVal(data) {
dynamic max = data.first;
data.forEach((e) {
if (e.data > max.data) max = e;
});
print(max.data);
return max.data;
}

double minVal(data) {
dynamic max = data.first;
data.forEach((e) {
if (e.data < max.data) max = e;
});
print(max.data);
if (max.data > 0) return 0;
return max.data;
}

double _top;
double _left;

List<Color> gradientColors = [
Colors.blue,
];

@override
Widget build(BuildContext context) {
//UNTUK MENDAPATKAN UKURAN LAYAR
double tWidth = MediaQuery.of(context).size.width;
double tHeigth = MediaQuery.of(context).size.height;
String titleCustom;

//TEMPAT MENGUBAH POSISI LABEL
//SESUAIKAN DENGAN KETINGGIAN SUMBU Y
//
//ACUAN PENGUBAHAN
//0.00x DARI UKURAN LAYAR AGAR RESPONSIVE DI SEMUA HP
switch (widget.judul) {
case ('pH'):
_top = tHeigth * 0.007;
_left = tWidth * 0.015;
```

```
titleCustom = "(pH)";
break;
// KODINGAN INI PAKAI JIKA CUSTOMISASI BERBEDA DARI DEFAULT
case ('Turbidity'):
_top = tHeigth * 0.007;
_left = tWidth * 0.015;
titleCustom = "(NTU)";
break;
case ('Temperature'):
_top = tHeigth * 0.007;
_left = tWidth * 0.015;
titleCustom = "(\u00b0C)";
break;
case ('Humidity'):
_top = tHeigth * 0.007;
_left = tWidth * 0.015;
titleCustom = "(%)";
break;
//DEFAULT DI PAKAI UNTUK POSISI YANG SAMA
default:
_top = tHeigth * 0.007;
_left = tWidth * 0.015;
}

return Card(
color: Colors.grey[200],
margin: EdgeInsets.symmetric(horizontal: 11, vertical: 16),
shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
child: Column(
mainAxisAlignment: MainAxisAlignment.spaceAround,
children: <Widget>[
Card(
//WARNA CARD
color: Colors.white,
shape:
```

```
RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
//POSISI CARD
margin: EdgeInsets.symmetric(horizontal: 13, vertical: 5),
child: Column(
children: <Widget>[
Stack(
children: <Widget>[
// Card(
// color: Colors.lightBlueAccent.withOpacity(0.9),
Row(
mainAxisAlignment: MainAxisAlignment.start,
children: <Widget>[
Expanded(
child: Container(
//UKURAN CHART
//UBAH 0.8 DAN 0.72 SESUAI KEMAUAN
//MAKSUDNYA UKURAN LAYAR DI KALIKAN DENGAN 0.XX
width: MediaQuery.of(context).size.width * 0.82,
height: MediaQuery.of(context).size.height * 0.72,
decoration: const BoxDecoration(),
child: Padding(
padding: const EdgeInsets.only(
right: 20.0,
left: 20.0,
top: 35,
bottom: 120),
child: LineChart(
mainData(),
),
),
),
),
],
),
// ),
Positioned(
top: _top,
left: _left,
child: Material(
color: Colors.transparent,
child: Text(titleCustom,
style: TextStyle(
color: Colors.black,
)))),
Positioned(
left: tWidth * 0.77,
top: tHeigth * 0.56,
child: Material(
color: Colors.transparent,
child: Text(
'(Time)',
style: TextStyle(
color: Colors.black,
),
))),
Positioned(
//POSISI HISTORY
top: tHeigth * 0.62,
left: tWidth * 0.29,
child: MaterialButton(
minWidth: MediaQuery.of(context).size.width * 0.3,
shape: RoundedRectangleBorder(
borderRadius: BorderRadius.circular(16),
),
onPressed: () {
Navigator.push(
context,
MaterialPageRoute(
builder: (context) => Wrapper(
judul: widget.judul,
param: widget.param,
page: 'details')));
},
padding: EdgeInsets.symmetric(
vertical: 5,
),
color: Colors.blue[900],
child: Text(
'History',
style: TextStyle(
fontSize: 15,
fontWeight: FontWeight.bold,
),
),
textColor: Colors.white,
),
),
],
),
],
),
),
// Expanded(child: SizedBox()),
],
),
);
}
LineChartData mainData() {
return LineChartData(
gridData: FlGridData(
show: false,
drawVerticalLine: false,
),
titlesData: FlTitlesData(
show: true,
leftTitles: SideTitles(
textStyle: const TextStyle(
color: Colors.black,
//fontWeight: FontWeight.bold,
fontSize: 11),
showTitles: true,
reservedSize: 28,
margin: 12,
interval: (maxVal(widget.lists) * 1.5) / 5),
bottomTitles: SideTitles(
textStyle: const TextStyle(
color: Colors.black,
//fontWeight: FontWeight.bold,
fontSize: 11),
showTitles: true,
getTitles: (value) {
switch (value.toInt()) {
case 1:
return DateFormat('H:mm:s').format(
DateTime.fromMillisecondsSinceEpoch(
widget.lists[0].timestamp));
case 15:
return DateFormat('H:mm:s').format(
DateTime.fromMillisecondsSinceEpoch(
widget.lists[14].timestamp));
case 28:
return DateFormat('H:mm:s').format(
DateTime.fromMillisecondsSinceEpoch(
widget.lists[29].timestamp));
}
return '';
})),
borderData: FlBorderData(
show: false,
border: Border.all(color: const Color(0xaaddaaff), width: 1)),
minX: 0,
maxX: widget.lists.length.toDouble(),
minY: minVal(widget.lists) * 1.5,
maxY: maxVal(widget.lists) * 1.5,
lineBarsData: [
LineChartBarData(
colors: [
Colors.redAccent,
],
spots: widget.lists.asMap().entries.map((e) {
return FlSpot(e.key.toDouble(), e.value.data.toDouble());
}).toList(),
isCurved: true,
preventCurveOverShooting: true,
curveSmoothness: 0,
dotData: FlDotData(show: true),
barWidth: 4,
isStrokeCapRound: false,
belowBarData: BarAreaData(
show: true,
colors:
gradientColors.map((color) => color.withOpacity(1)).toList(),
),
),
],
);
}
}
```

n) fcm.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:dio/dio.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:mailer/mailer.dart';
import 'package:mailer/smtp_server.dart';

var _token;

FirebaseMessaging _firebaseMessaging = FirebaseMessaging();

generateToken() async {
await _firebaseMessaging.getToken().then((token) {
_token = token;
print(_token);
});
}

getToken() {
return _token;
}

sendEmail(text) async {
var emails = [];
await Firestore.instance.collection('emails').getDocuments().then(
(value) {
value.documents.forEach((element) {
if (element.data['value'] == true) {
emails.add(element.data['email']);
}
});
},
);

String username = 'aqumosysmonitoring@gmail.com';
String password = 'UUM_AquMoSys2020';

final smtpServer = gmail(username, password);
// Creating the Gmail server

// Create our email message.
final message = Message()
..from = Address(username)
// ..recipients.add('zaki23newton@gmai.com') //recipent email
..ccRecipients.addAll(emails) //cc Recipents emails
```

```
// ..bccRecipients
// .add(Address('bccAddress@example.com')) //bcc Recipents emails
..subject = 'AquMoSys Alert' //subject of the email
..text = text; //body of the email

try {
final sendReport = await send(message, smtpServer);
print(
'Message sent: ' + sendReport.toString()); //print if the email is sent
} on MailerException catch (e) {
print(
'Message not sent. \n' + e.toString()); //print if the email is not sent
// e.toString() will show why the email is not sending
}
}

sendNotification(text) async {
var emails = [];
await Firestore.instance.collection('emails').getDocuments().then(
(value) {
value.documents.forEach((element) {
if (element.data['value'] == true) {
emails.add(element.data['email']);
}
});
},
);

String username = 'aqumosysmonitoring@gmail.com';
String password = 'UUM_AquMoSys2020';

final smtpServer = gmail(username, password);
// Creating the Gmail server

// Create our email message.
final message = Message()
..from = Address(username)
// ..recipients.add('zaki23newton@gmai.com') //recipent email
```

```
..ccRecipients.addAll(emails) //cc Recipents emails
// ..bccRecipients
// .add(Address('bccAddress@example.com')) //bcc Recipents emails
..subject = 'AquMoSys Alert' //subject of the email
..text = text; //body of the email

try {
final sendReport = await send(message, smtpServer);
print(
'Message sent: ' + sendReport.toString()); //print if the email is sent
} on MailerException catch (e) {
print(
'Message not sent. \n' + e.toString()); //print if the email is not sent
// e.toString() will show why the email is not sending
}

await Firestore.instance.collection('tokens').getDocuments().then((value) {
var tokens = value.documents;
print(tokens[0].data);
tokens.forEach((token) async {
if (token.data['value'] == true) {
await Dio().post(
'https://fcm.googleapis.com/fcm/send',
options: Options(
headers: {
'Authorization':
'key=AAAAa8lKikM:APA91bFQ5Hc0Y2BrArNF8FE8DYj96uveDT0kQPpZMfTyKwaLai
x9tArowwTlF0h6rJ2ThIp99pgc4uQuX60sKcBpTccjWCQGcXhQ4-
IgVfKWdpCdF9qzukrc-z_pw-vPj3tDm3VRnOdS',
},
),
data: {
"notification": {
"title": "Threshold Alert",
"body": text,
"click_action": "FLUTTER_NOTIFICATION_CLICK",
},
// "data": {"name": "Coba", "age": 25},
"to": token['token']
},
).then((value) {
print(value.data);
return true;
}).catchError((e) {
print(e);
});
}
});
});
Firestore.instance.collection('emails').getDocuments().then((value) {
var emails = value.documents;
print(emails[0].data);
print(emails.length);
});
print(text);
}
```

o) main.dart

```
import 'dart:ui';
//import 'package:aquamonitoring/constant.dart';
import 'package:aquamonitoring/pages/about.dart';
import 'package:aquamonitoring/pages/home.dart';
import 'package:aquamonitoring/pages/notification_screen.dart';
import 'package:aquamonitoring/pages/setting_screen.dart';
import 'package:aquamonitoring/pages/sign_in_screen.dart';
import 'package:aquamonitoring/pages/wrapper.dart';
import 'package:aquamonitoring/providers/auth_provider.dart';
import 'package:aquamonitoring/providers/data_provider.dart';
import 'package:aquamonitoring/providers/screen_provider.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:custom_splash/custom_splash.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';
import 'package:flutter/services.dart';
import 'fcm.dart';

void main() {
runApp(MyApp());
}

class MyApp extends StatelessWidget {
@override
Widget build(BuildContext context) {
SystemChrome.setPreferredOrientations([
DeviceOrientation.portraitUp,
]);
return MultiProvider(
providers: [
ChangeNotifierProvider(create: (_) => DataProvider()),
ChangeNotifierProvider(create: (_) => ScreenProvider()),
ChangeNotifierProvider(create: (_) => AuthProvider()),
],
child: MaterialApp(
debugShowCheckedModeBanner: false,
title: 'AquMoSys',
theme: ThemeData(
//FONTCHANGE body
//GANTI JENIS FONT DI SINI SESUAI DI WEBSITE fonts.google.com
//GANTI DENGAN NAMA YANG ADA "TEXTTHEME"
//JIKA TIDAK ADA BERARTI BELUM TERDAFTAR DI LIBRARY
textTheme: GoogleFonts.latoTextTheme(
Theme.of(context).textTheme,
),
primaryColor: Colors.blue[900],
visualDensity: VisualDensity.adaptivePlatformDensity,
),
home: CustomSplash(
imagePath: 'assets/icon/splashscreen.png',
backGroundColor: Colors.blue[900],
animationEffect: 'zoom-in',
logoSize: 200,
home: MainPage(),
duration: 2500,
type: CustomSplashType.StaticDuration,
),
// AnimatedSplash(
// imagePath: 'assets/icon/splashscreen.png',
// home: MainPage(),
// duration: 2500,
// type: AnimatedSplashType.StaticDuration,
// ),
),
);
}
}

class MainPage extends StatefulWidget {
@override
_MainPageState createState() => _MainPageState();
}

class _MainPageState extends State<MainPage> {
@override
Widget build(BuildContext context) {
var authProvider = Provider.of<AuthProvider>(context);
var dataProvider = Provider.of<DataProvider>(context);

return HomePage(authProvider, dataProvider);
}
}

class HomePage extends StatefulWidget {
```

```
final AuthProvider authProvider;
final DataProvider dataProvider;

HomePage(this.authProvider, this.dataProvider);

@override
_HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
String _message = '';

final FirebaseMessaging _firebaseMessaging = FirebaseMessaging();

getInit() async {
await generateToken();
await Firestore.instance
.collection('tokens')
.document(getToken())
.get()
.then(
(value) async {
print('get token');
print(value.data);
print('get token');
if (!value.exists) {
await Firestore.instance
.collection('tokens')
.document(getToken())
.setData(
{
'token': getToken(),
'value': true,
},
).then(
(value) {
print('token added');
},
);
}
},
).catchError((onError) async {
print(onError);
await Firestore.instance
.collection('tokens')
.document(getToken())
.setData(
{
'token': getToken(),
'value': true,
},
).then(
(value) {
print('token added');
},
);
});
await widget.authProvider.getInitialUser();

await widget.dataProvider.getInitialData();
}

void getMessage() {
print(_message);
_firebaseMessaging.configure(
onMessage: (Map<String, dynamic> message) async {
print('onMessage');
print('on message $message');
setState(() => _message = message["notification"]["title"]);
}, onResume: (Map<String, dynamic> message) async {
print('onResume');
print('on resume $message');
setState(() => _message = message["notification"]["title"]);
}, onLaunch: (Map<String, dynamic> message) async {
```

```
print('onLaunch');
print('on launch $message');
setState(() => _message = message["notification"]["title"]);
});
}

@override
void initState() {
super.initState();
getInit();
}

// @override
Widget child = Container(
child: Center(
child: Text('Page Not Found'),
),
);

Widget build(BuildContext context) {
var authProvider = Provider.of<AuthProvider>(context);
var screenProvider = Provider.of<ScreenProvider>(context);
var currentIndex = screenProvider.currentIndex;

Future<void> logoutDialog() async {
print('current email: ${authProvider.currentEmail}');

await Firestore.instance
.collection('emails')
.document(authProvider.currentEmail)
.updateData({'value': false});
setState(() {});

await authProvider.logout();

return showDialog<void>(
context: context,
barrierDismissible: true, // user must tap button!
builder: (BuildContext context) {
```

```
return AlertDialog(
shape:
RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
content: SingleChildScrollView(
child: ListBody(
children: <Widget>[
Text('You have successfully signed out'),
],
),
),
actions: <Widget>[
FlatButton(
child: Text('Close'),
onPressed: () {
Navigator.of(context).pop();
},
),
],
);
},
);
}
switch (currentIndex) {
case 0:
child = Wrapper(
judul: 'Temperature',
page: 'chart',
param: 'Temperature',
);
break;
case 1:
child = Wrapper(
judul: 'Turbidity',
page: 'chart',
param: 'Turbidity',
);
break;

case 2:
child = Home();
break;

case 3:
child = Wrapper(
judul: 'pH',
page: 'chart',
param: 'pH',
);
break;
case 4:
child = Wrapper(
judul: 'Humidity',
page: 'chart',
param: 'Humidity',
);
break;
}

buildSignIn() {
if (authProvider.currentEmail.isEmpty) {
return ListTile(
leading: Icon(
Icons.person,
color: Colors.white,
),
title: Text(
'Sign In',
style: GoogleFonts.droidSans(
textStyle: TextStyle(
color: Colors.white,
fontSize: 15,
fontWeight: FontWeight.bold,
),
),
),
onTap: () {
Navigator.push(context,
MaterialPageRoute(builder: (context) => SignInScreen()));
},
);
} else {
return ListTile(
leading: Icon(
Icons.person,
color: Colors.white,
),
title: Text(
'Sign Out',
style: GoogleFonts.droidSans(
textStyle: TextStyle(
color: Colors.white,
fontSize: 15,
fontWeight: FontWeight.bold,
fontStyle: FontStyle.italic,
),
),
),
onTap: () {
logoutDialog();
},
);
}
}
var bottomNavigationBar2 = BottomNavigationBar(
backgroundColor: Colors.blue[900],
type: BottomNavigationBarType.fixed,
currentIndex: currentIndex,
onTap: (int index) => screenProvider.setCurrentIndex(index),
selectedItemColor: Colors.white,
unselectedItemColor: Colors.white,
showUnselectedLabels: false,
items: [
BottomNavigationBarItem(
backgroundColor: Colors.white,
icon: new Image.asset(
'assets/icon/temperature (2).png',
height: 25,
width: 25,
),
title: Text(
'Temp',
//FONTCHANGE BOTTOM
style: GoogleFonts.droidSans(
fontWeight: FontWeight.bold,
),
)),
BottomNavigationBarItem(
backgroundColor: Colors.white,
icon: new Image.asset(
'assets/icon/turbid.png',
height: 25,
width: 25,
),
title: Text(
'Turbidity',
//FONTCHANGE BOTTOM
style: GoogleFonts.droidSans(
fontWeight: FontWeight.bold,
),
)),
BottomNavigationBarItem(
backgroundColor: Colors.white,
icon: Icon(
Icons.home,
color: Colors.white,
),
title: Text(
'Home',
//FONTCHANGE BOTTOM
style: GoogleFonts.droidSans(
fontWeight: FontWeight.bold,
),
),
),
BottomNavigationBarItem(
backgroundColor: Colors.white,
icon: new Image.asset(
'assets/icon/ph-1.png',
height: 25,
width: 25,
),
title: Text(
'pH',
//FONTCHANGE BOTTOM
style: GoogleFonts.droidSans(
fontWeight: FontWeight.bold,
),
)),
BottomNavigationBarItem(
backgroundColor: Colors.white,
icon: new Image.asset(
'assets/icon/humidity.png',
height: 25,
width: 25,
),
title: Text(
'Humidity',
//FONTCHANGE BOTTOM
style: GoogleFonts.droidSans(
fontWeight: FontWeight.bold,
),
),
)
],
);
return Scaffold(
backgroundColor: Colors.blue[500],
//FONTCHANGE appbar
appBar: AppBar(
//FONTCHANGE
title: Text(
'Aquaponic Monitoring System',
style: GoogleFonts.droidSans(
textStyle: TextStyle(
fontWeight: FontWeight.bold,
),
),
),
// actions: <Widget>[
// IconButton(
// icon: Icon(Icons.settings),
// onPressed: () {
// Navigator.push(context,
// MaterialPageRoute(builder: (context) => SettingScreen()));
// },
// )
// ],
),
body: SafeArea(
child: child,
),
bottomNavigationBar: bottomNavigationBar2,
drawer: Drawer(
child: Container(
color: Colors.blue[500],
child: ListView(
children: <Widget>[
DrawerHeader(
decoration: BoxDecoration(
color: Colors.blue[900],
),
child: Center(
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
SizedBox(
width: 60.0,
height: 60.0,
child: ClipPath(
child: Image.asset(
'assets/icon/splashscreen.png',
scale: 0.5,
fit: BoxFit.cover,
),
),
),
SizedBox(
height: 10,
),
authProvider.currentEmail.isEmpty
? SizedBox()
: Text(
authProvider.currentEmail,
style: TextStyle(
color: Colors.white,
),
),
],
),
),
),
buildSignIn(),
ListTile(
leading: Icon(
Icons.settings,
color: Colors.white,
),
title: Text(
'Threshold Settings',
style: GoogleFonts.droidSans(
textStyle: TextStyle(
color: Colors.white,
fontSize: 15,
fontWeight: FontWeight.bold,
),
),
),
onTap: () {
Navigator.push(context,
MaterialPageRoute(builder: (context) => SettingScreen()));
},
),
ListTile(
leading: Icon(
Icons.add_alert,
color: Colors.white,
),
title: Text(
'Notifications',
style: GoogleFonts.droidSans(
textStyle: TextStyle(
color: Colors.white,
fontSize: 15,
fontWeight: FontWeight.bold,
),
),
),
onTap: () {
Navigator.push(
context,
MaterialPageRoute(
builder: (context) => NotificationScreen()));
},
),
ListTile(
leading: Icon(
Icons.info,
color: Colors.white,
),
title: Text(
'About AquMoSys',
style: GoogleFonts.droidSans(
textStyle: TextStyle(
color: Colors.white,
fontSize: 15,
fontWeight: FontWeight.bold,
),
),
),
onTap: () {
Navigator.push(context,
MaterialPageRoute(builder: (context) => AboutPage()));
},
),
],
),
),
),
);
}
}
```

p) pH_TempESP32

```
#define WIFI_SSID "UUMWiFi_Guest"
#define WIFI_PASSWORD ""
#define FIREBASE_HOST "aquamonitoring-edc92.firebaseio.com" // host di isi
#define FIREBASE_AUTH "lSXha7n2ShBeTaQzSHEqBaeVtqp9QLL2tLqbXibA" // auth di
  isi

#define ESPADC 4096.0   //the esp Analog Digital Convertion value
#define ESPVOLTAGE 3300 //the esp voltage supply value
#define pin_ph          35
#define pin_turbidity   34
#define pin_dht         32
#define pin_ds          33
#define pin_led         13
#define DHTTYPE         DHT11


DHT dht(pin_dht, DHTTYPE);
OneWire oneWire(pin_ds);
// Pass our oneWire reference to Dallas Temperature sensor
DallasTemperature sensors(&oneWire);

FirebaseData firebaseData;

float voltage, volt, ntu, phValue, temperature, humidity, turbidity;


void read_all()
{
  static unsigned long timepoint = millis();
  if (millis() - timepoint > 1000U) //time interval: 1s
  {
    timepoint = millis();
    //voltage = rawPinValue / esp32ADC * esp32Vin
    voltage = analogRead(pin_ph) / ESPADC * ESPVOLTAGE; // read the voltage
    phValue = ph.readPH(voltage, temperature); // convert voltage to pH with temperature
  compensation
    Serial.print(temperature);
    Serial.print(" C ");
    Serial.print(humidity);
    Serial.print(" % ");
    Serial.print(phValue);
    Serial.print(" P ");
    Serial.print(turbidity);
    Serial.print(" NTU ");
    Serial.println();
  }
  ph.calibration(voltage, temperature); // calibration process by Serail CMD
}

// https://wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189
/*void read_turbidity() {
  int sensorValue = analogRead(pin_turbidity);// read the input on analog pin 0:
  float voltage = sensorValue * (5.0 / 1024.0); // Convert the analog reading (which goes from
   0 - 1023) to a voltage (0 - 5V):
  turbidity = voltage;
  }*/

// https://www.teachmemicro.com/arduino-turbidity-sensor/

void read_turbidity1() {
  volt = 0;
  for (int i = 0; i < 800; i++)
  {
    volt += ((float)analogRead(pin_turbidity) / 1023) * 5;
  }
  volt = volt / 800;
  volt = round_to_dp(volt, 1);
  if (volt < 2.5) {
    ntu = 3000;
  } else {
    ntu = -1120.4 * sq(volt) + 5742.3 * volt - 4353.8;
  }
  turbidity = ntu;
}

float round_to_dp( float in_value, int decimal_place )
{
  float multiplier = powf( 10.0f, decimal_place );
  in_value = roundf( in_value * multiplier ) / multiplier;
  return in_value;
}

void setup() {
  Serial.begin(9600);
  pinMode(pin_led, OUTPUT);
  for (int i = 0; i < 10; i++) {
    digitalWrite( pin_led, digitalRead( pin_led ) ^ 1 );
    delay(100);
  } digitalWrite( pin_led, 0);
  sensors.begin();
  ph.begin();
  Serial.println("|");
}

DynamicJsonDocument doc(1024);
```

```
JsonObject sensorTime = doc.createNestedObject("timestamp");
void loop() {

  sensors.requestTemperatures();
  temperature  = sensors.getTempCByIndex(0);
  humidity = dht.readHumidity();
  read_turbidity1();
  read_all();

  doc["Temperature"] = "temperature";
  doc["Humidity"] = humidity;
  doc["pH"] = phValue;
  doc["Turbidity"] = turbidity;
  sensorTime[".sv"] = "timestamp";

  if (Firebase.push(firebaseData, "Aqua", doc))
  {
    Serial.println("Data Sent ");
  }

  // handle error
  else {
    Serial.print("pushing /temperature failed:");
    Serial.println(firebaseData.errorReason());
    return;
  }
}
```