REAL-TIME MONITORING WATER QUALITY MONITORING SYSTEM ON BOARD USV

CHE KU MUHAMMAD DANIEL AIMAN BIN CHE KU MAZLAN

Bachelor of Engineering Technology Electrical

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	:	CHE KU MUHAMMAD DANIEL AIMAN BIN CHE KU MAZLAN
Date of Birth		
Title	:	REAL-TIME MONITORING WATER QUALITY MONITORING SYSTEM ON BOARD USV
Academic Session	:	20/21

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☐	OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_______________________	_________________________
(Student's Signature)	(Supervisor's Signature)
	Dr. Zainah Md Zain
New IC/Passport Number	Name of Supervisor
Date:25 JANUARY 2021	Date:

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

SUPERVISOR’S DECLARATION

I hereby declare that I have checked this thesis and, in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Engineering Technology in Electrical.

____

(Supervisor’s Signature)

Full Name	: Dr. Zainah Md. Zain
Position	: Senior Lecturer
Date	:

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

____

(Student's Signature)

Full Name	: CHE KU MUHAMMAD DANIEL AIMAN BIN CHE KU MAZLAN
ID Number	: TB17019
Date	: 25 JANUARI 2021

REAL-TIME MONITORING WATER QUALITY MONITORING SYSTEM ON BOARD USV

CHE KU MUHAMMAD DANIEL AIMAN BIN CHE KU MAZLAN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Engineering Technology in Electrical

Faculty of Electrical & Electronics Engineering Technology

UNIVERSITI MALAYSIA PAHANG

JANUARY 2021

ACKNOWLEDGEMENTS

First and foremost, praises and thanks to Allah, the Almighty, for His showers of blessings throughout my Senior Design Project I and II to complete the project successfully.

I would like to express my deep and sincere gratitude to my final year project supervisor, Dr Zainah Binti Md Zain for giving me the opportunity to do this project and providing invaluable guidance throughout this research. Her vision, sincerity and motivation have deeply inspired me. She has taught me the methodology to carry out the project and to present the project works as clearly as possible. It was a great privilege and honour to work and study under her guidance. I am extremely grateful for what he has offered me.

I am extremely grateful to my parent for their love. prayers, caring and sacrifices for educating and preparing me for my future. I am very thankful to my group members, Tanesh A/L Maran and Abdul Qaiyum Bin Haji Ramli for the cooperation and teamwork in completing the project.

Finally, the appreciation goes to whoever has help in the process of the project direct or indirectly to make this project a success.

ABSTRAK

Unmanned Surface Vehicle (USV) seperti namanya adalah alat untuk mengangkut barang atau objek ke lokasi yang dimaksudkan. Ia tidak memerlukan campur tangan manusia untuk operasinya. Kenderaan permukaan tanpa pemandu (USV) adalah kapal pembawa air yang mampu beroperasi di permukaan air tanpa ada pengendali manusia di dalamnya. Pada mulanya, ia dibuat dengan memasang semula kawalan radio berawak sebelumnya, pelbagai kenderaan permukaan tanpa pemandu yang dibina kini tersedia. Agar USV dapat beroperasi tanpa manusia, ia mesti dikawal secara tanpa wayar. Projek ini meneroka penggunaan IoT, seperti komunikasi tanpa wayar, pangkalan data awan dan aplikasi mudah alih. Wi-Fi digunakan sebagai media komunikasi untuk mengawal USV melalui aplikasi mudah alih. Projek ini juga merangkumi pengembangan aplikasi mudah alih mudah menggunakan MIT App Inventor. Tujuan aplikasi mudah alih adalah untuk membuat platform mudah alih untuk mengawal USV dan untuk menunjukkan hasil sensor pada aplikasi tersebut. Wi-Fi juga telah digunakan untuk memuat naik dan memuat turun data dari dan ke pangkalan data awan. Mikrokontroler yang digunakan dalam projek ini adalah WEMOS MEGA + WIFI, gabungan Arduino Mega dan cip Wi-Fi terbina dalam (ESP8266) untuk menyambung ke Wi-Fi. USV didorong oleh motor tanpa sikat sebagai pengawal baling-baling oleh Pengawal Kelajuan Elektronik dan servo mikro sebagai kemudi untuk menavigasi di dalam air. Sensor yang digunakan untuk memantau kualiti air adalah sensor pH, sensor suhu kalis air DS18B20 dan sensor kekeruhan. Kemudian semua data dari sensor dibaca oleh Arduino Mega dan kemudian data dihantar ke pangkalan data pelayan awan oleh ESP8266. Pangkalan data yang digunakan dalam projek ini adalah ThingSpeak, pangkalan data siap pakai yang dapat dihubungkan dengan mudah ke sistem, dan data yang dikumpulkan akan digunakan untuk muncul di platform lain, seperti aplikasi mudah alih. Hasil dari projek tersebut adalah pengembangan sistem USV yang terdiri daripada perkakasan, perisian dan aplikasi mudah alih yang dapat digunakan untuk memantau, mengendalikan dan memantau USV dan sensornya.

ABSTRACT

Unmanned Surface Vehicle (USV) as the name suggest is a means of transporting goods or objects to the intended location. It does not require human intervention for its operation. Unmanned surface vehicles (USVs) are water-borne vessels capable of operating on the surface of the water without any human operator on board. Originally created by retrofitting previously manned radio controls, a wide range of purpose-built unmanned surface vehicles are now available. In order for the USV to operate without a human being, it must be controlled wirelessly. This project explored the use of IoT, such as wireless communication, cloud database and mobile applications. Wi-Fi is used as a communication medium to control the USV over a mobile application. The project also includes the development of simple mobile applications using the MIT App Inventor. The aim of the mobile app is to create a mobile platform to control the USV and to show the results of the sensors on the application. Wi-Fi has also been used to upload and download data from and to the cloud database. The microcontroller used in this project is WEMOS MEGA+WIFI, a combination of Arduino Mega and a built-in Wi-Fi chip (ESP8266) to connect to Wi-Fi. The USV is driven by a brushless motor as the propeller controller by the Electronic Speed Controller and micro servo as the rudder to navigate in the water. The sensors used for monitoring water quality are the pH sensor, the DS18B20 waterproof temperature sensor and the turbidity sensor. Then all the data from the sensors is read by the Arduino Mega and then the data is sent to the cloud server database by the ESP8266. The database used in this project is ThingSpeak, a ready-made database that can be easily connected to the system, and the data collected will be used to appear on the other platform, such as a mobile application. The result of the project is the development of a USV system consisting of hardware, software and a mobile application that can be used to monitor, control and monitor the USV and its sensors.

TABLE OF CONTENT

DECLARATION

TITLE PAGE

ACKNOWLEDGEMENTS ii

ABSTRAK iii

ABSTRACT iv

TABLE OF CONTENT v

LIST OF TABLES viii

LIST OF FIGURES ix

LIST OF ABBREVIATIONS xi

CHAPTER 1 INTRODUCTION 1

1.1 Project Background 1

1.2 Problem Statement 3

1.3 Objectives 3

1.2.2 Scope 4

CHAPTER 2 LITERATURE REVIEW 5

2.1 Aluminium hull USV for coastal water and seafloor monitoring 5

2.2 Development of an Unmanned Surface Vehicle Platform for Autonomous Navigation in Paddy Field 6

2.3 Radar Based Collision detection developments on USV ROAZ II 7

2.4 Development of Navigation System for Unmanned Surface Vehicle by Improving Path Tracking Performance 8

2.5 Water Quality Monitoring System Based on IOT 9

2.6 Water Monitoring and Analytic Based Thingspeak 10

CHAPTER 3 METHODOLOGY **11**

3.1 Introduction 11

3.2 USV Hardware Design 11

3.2.1 USV Design 12

3.2.2 Mechanical System 13

3.2.2.1 Thruster 13

3.2.2.1 Rudder 14

3.3 Flowchart 15

3.4 Block Diagram 18

3.5 Software Configuration and Development 19

3.5.1 WEMOS MEGA+WIFI 19

3.5.2 Mega 2560 Coding 22

3.6 ESP32 Cam 26

3.7 Mobile Application 27

3.8 Circuit Diagram 36

3.7.1 Circuit diagram for mechanical parts and ESP32-Cam 36

3.7.2 Circuit diagram for water quality monitoring system 37

3.9 ThingSpeak configurations 36

CHAPTER 4 RESULTS AND DISCUSSION **40**

4.1 Introduction 40

4.2 Experimental Testing 40

4.2.1 USV Mechanical System 40

4.2.2 pH Sensor Result 41

4.2.3 Turbidity Sensor Result 43

4.2.4 Temperature Sensor Result 44

4.3 Problem Encounter 45

CHAPTER 5 CONCLUSION 47

5.1 Introduction 47

5.2 Future Recommendation 47

REFERENCES 49

APPENDICES A MEGA 2560 CODE 51

APPENDICES B ESP 8266 CODE 57

APPENDICES C ESP32 CODE 63

APPENDICES D GANTT CHART 65

APPENDICES E PROJECT COST 66

LIST OF TABLES

Table 3.1 Microcontroller Specifications 21

LIST OF FIGURES

Figure 1.1 Sampling Process of Kim Kim River 1
Figure 2.1 ALANIS USV 5
Figure 2.2 ALANIS USV performing basic guidance and control tests. 5
Figure 2.3 Unmanned surface vehicle platform 6
Figure 2.4 The 6-D of motion of the USV platform in geodetic coordinate system 6
Figure 2.5 Target detected and leaving closest approach zone 7
Figure 2.6 ROAZ II Autonomous Surface Vehicle 7
Figure 2.7 Shows the location on the google map 8
Figure 2.8 USV platform searching the target platform location 8
Figure 2.9 System design model of water quality monitoring system using IOT 9
Figure 2.10 Water Monitoring and Analytic Based Thingspeak 10
Figure 2.11 Thingspeak cannel of proposed system 10
Figure 3.1 USV design 12
Figure 3.2 Thruster and ESC 13
Figure 3.3 Rudder 14
Figure 3.4 Flowchart of the overall system 15
Figure 3.5 Flowchart of the Water Quality Monitoring System 16
Figure 3.6 Flowchart of USV Control System 17
Figure 3.7 Block diagram of the overall USV system 18
Figure 3.8 WEMOS MEGA+WIFI 19
Figure 3.9 DIP switch configuration for mode selection 20
Figure 3.10 Switch for serial selection 22
Figure 3.11 Example coding of sending data to ESP 8266 22
Figure 3.12 Credentials parameter of the Wi-Fi 23
Figure 3.13 Webserver setup 23
Figure 3.14 Command to be send to Arduino Mega 23
Figure 3.15 Channel number and write API key for ThingSpeak 24
Figure 3.16 Example of codes for turbidity sensor 25
Figure 3.17 ESP32 Cam diagram 26
Figure 3.18 The block codes example for welcome page 27
Figure 3.19 The GUI of welcome page 28
Figure 3.20 Camera function of the ESP32 Cam 29

Figure 3.21	Blocks for Camera function	29
Figure 3.22	The control page of the mobile application	30
Figure 3.23	Blocks for GO button	30
Figure 3.24	Example blocks to send data to ThingSpeak	31
Figure 3.25	Example blocks for result page	32
Figure 3.26	Example blocks for single result function	33
Figure 3.27	Example blocks for combine results function	34
Figure 3.28	Sensor data in table form	35
Figure 3.29	Sensor data in graphical form	35
Figure 3.30	Circuit diagram for mechanical system and ESP32-Cam	36
Figure 3.31	Circuit diagram for water quality system	37
Figure 3.32	ThingSpeak channel created	38
Figure 3.33	Channel details of pH sensor	38
Figure 3.34	ThingSpeak link to access the channel detail	39
Figure 4.1	Experimental testing in a pool	40
Figure 4.2	The application view when testing	41
Figure 4.3	pH sensor result on ThingSpeak.	42
Figure 4.4	pH sensor acquire	42
Figure 4.5	Turbidity sensor result on ThingSpeak	43
Figure 4.6	Turbidity sensor acquire	43
Figure 4.7	Temperature sensor result on ThingSpeak	44
Figure 4.8	Temperature sensor acquire	44
Figure 4.9	Video feed on application	45
Figure 4.10	Flush function on application	46
Figure 4.11	ESP32 Cam block on MIT App Inventor	48

LIST OF ABBREVIATIONS

USV	Unmanned Surface Vehicle
IoT	Internet of Things
BOD	Biochemical Oxygen Demands
COD	Chemical Oxygen Demands
WQI	Water quality index
DO	Dissolved oxygen
SS	Suspended solids
AN	Ammonia
Wi-Fi	Wireless Fidelity
ROAZ	Regionaal Overleg Acute Zorgketen
PVC	Polyvinyl chloride
USB	Universal Serial Bus
EEPROM	Electrically Erasable Programmable Read-Only Memory
IP	Internet Protocol
LCD	Liquid Crystal Display
GUI	Graphical User Interface
ESC	Electronic Speed Controller
UART	Universal Asynchronous Receiver/Transmitter

CHAPTER 1

INTRODUCTION

1.1 Project Background

Water is one of our main sources of living and currently, we are facing worldwide water pollution crisis. Water quality is a large environmental problem and one of humanity's grand challenges. In 2019, Johor state government faced a lot of losses and had hard time to treat all those polluted Kim Kim River due to chemical substances. This has become one of the major problems, as the cost of the treatment is expensive. The government had to allocated an emergency fund of RM 6.5 millions to speed up the cleaning work of Kim Kim River as it has affected a lot of citizen. Figure 1.1 shows the sampling process of the Kim Kim River.(The Straits Times,2019)

Figure 1.1: Sampling Process of Kim Kim River

The results of this water test were measured using eight parameters including the Biochemical Oxygen Demands (BOD) and Chemical Oxygen Demands (COD) and Total Suspended Solids water levels. The highest BOD levels of dead rivers between 88-98 and reading 98 were categorized as highly polluted rivers. But the Kim Kim River readings (Circular 11 sample location) are 821. (Star Online,2019)

A total of 5,848 children and adults were treated since March 8 2019 due to toxic waste contamination in the Kim River, Pasir Gudang, Johor. Luckily none were reported dead due to this contamination. (The Sun Daily,2019)

Water quality index (WQI) determines the water quality of the aquatic ecosystem. Some of the water quality parameters are as dissolved oxygen (DO), biochemical oxygen demand (BOD), chemical oxygen demand (COD), suspended solids (SS), ammonia (AN) and pH. These parameters are constantly monitored to ensure the water quality.

Water sampling by using Unmanned Surface Vehicle (USV) is one of the effective ways to get the real-time water quality index. This USV will be equipped with sensors that are related to water quality monitor. These sensors can communicate with each other to collect and transmit data from the surrounding environment. It’s been used widely for monitoring habitat, natural disaster and many more. Those collected data will be transmitted to base stations. These USVs are equipped with multiple sensors and they are capable of transmitting data at the same time. And more importantly, they can be configured to be low or high power to save power and they are small in size compared to other type of wireless device. (WSD,2019)

This project uses the WEMOS MEGA+WIFI integrated with water quality sensors and USV’s controller to monitor the water quality on real time basis. The data monitored all be transmitted to a remote base station (mobile application) to allow data analysis take place. Based on the analyzed data, the quality of the polluted waste can be identified and a solution to improve the quality can be proposed.

1.2 Problem Statement

Manual sampling is a traditional method of water sampling that used widely in water sampling process to measure the water quality. Lately, the development of USV give the researchers an idea to equip the USV with water sampling system to measure the water for water quality analysis purpose. However, the existing platforms are too expensive and no wholly open-source USV platforms that allows modification for diverse purposes.

1.3 Objective

The objectives of this project are to achieve the following;

1. To design and develop an USV that low in cost and attached with onboard water quality monitoring system.

2. To develop a water quality monitoring system that can send data real time from the sensors on the USV to the platform.

3. To develop remote control system via mobile application to control the USV and display the data from the sensor in real time.

1.4 Scope

The main target of this project is to design a framework or a platform that is able to monitor the water quality in term of its pH level, temperature, turbidity and remotely or automated system for the USV based on the Android application through a notification to inform the user. Apart from that, the system must be efficiently easy to set up and portable based on the requirement needed by the user, portable and user friendly. To counter the issues mentioned above, several measures have been implemented for this project:

1. The coverage of this system is limited to the range of connection between the transmitter and the receiver.
2. The USV needs to be resistant, fast and high maneuverability to avoid the malfunction when it is running.

CHAPTER 2

LITERATURE REVIEW

2.1 Aluminium hull USV for coastal water and seafloor monitoring

This journal tells about Construction and Development of the Aluminium Autonomous Navigator for Intelligent Sampling (ALANIS), an unmanned surface vehicle (USV) developed by the Autonomous robotic systems and control group of CNR-ISSIA Ge-nova basically for coastal monitoring. The onboard automation system of the rubber boat shaped aluminium vessel manages the steering and throttle of a conventional outboard motor on the base of user desires and measurements supplied by the navigation package. An automatically controlled winch is devoted to deployment and recovery of scientific instrumentation through a suitable hole in the vehicle prow. (M. Caccia,2009)

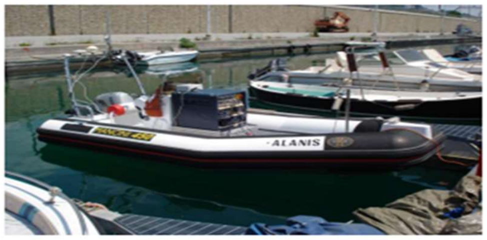


Figure 2.1 ALANIS USV

Figure 2.2 ALANIS USV performing basic guidance and control tests.

2.2 Development of an Unmanned Surface Vehicle Platform for Autonomous Navigation in Paddy Field

This project carried out to develop an unmanned surface vehicle (USV) platform for autonomous navigation in the paddy field. The surface vehicle used in this research was a radio controlled air propeller vessel that had been modified into an unmanned surface vehicle platform. A GPS compass system was attached to the top of the USV platform as the navigation system to provide the position and heading angle. The USV platform can autonomously navigate to the predefined navigation map. From the GPS trajectory data of the map-based navigation experiment, the in-system root mean square (RMS) lateral error from the target path was observed to be less than 0.45 m, and the in-system RMS heading error was 4.4 degree or less. The purpose of the research is to realize the autonomous weeding, intelligent fertilization or paddy growth management based on this USV platform.(Y Liu,2016)

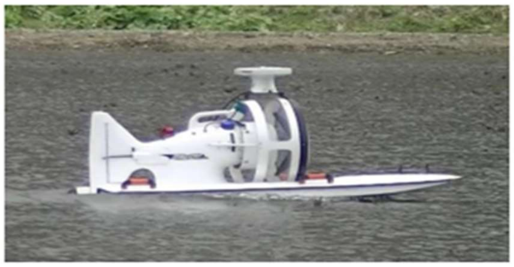


Figure 2.3 Unmanned surface vehicle platform

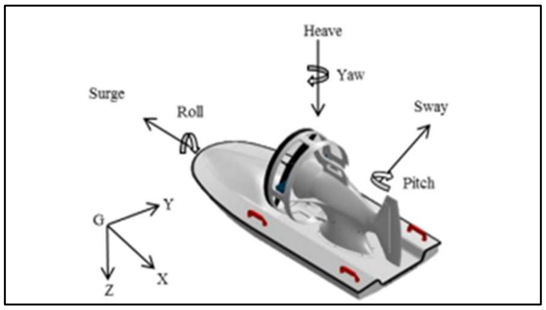


Figure 2.4 The 6-D of motion of the USV platform in geodetic coordinate system

2.3 Radar Based Collision detection developments on USV ROAZ II

This work presents the integration of obstacle detection and analysis capabilities in a coherent and advanced C&C framework allowing mixed-mode control in unmanned surface systems. The collision avoidance work has been successfully integrated in an operational autonomous surface vehicle and demonstrated in real operational conditions. We present the collision avoidance system, the ROAZ autonomous surface vehicle and the results obtained at sea tests.(C. Almeida, 2009)

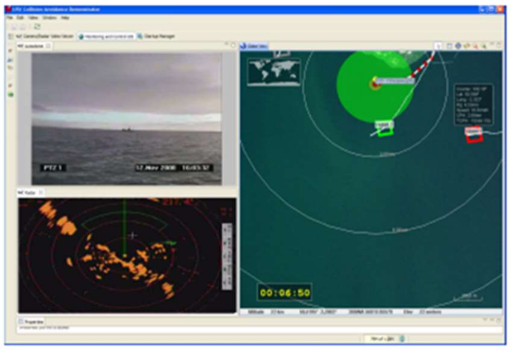


Figure 2.5 Target detected and leaving closest approach zone

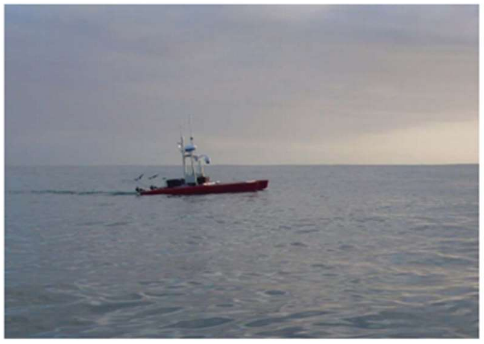


Figure 2.6 ROAZ II Autonomous Surface Vehicle.

2.4 Development of Navigation System for Unmanned Surface Vehicle by Improving Path Tracking Performance

A fundamental of a fully unmanned vehicle entails the use of Global Positioning System (GPS) and sensors module that emits USV a series of a waypoint for moving towards the target. In other words, GPS provides an accurate data location longitude and latitude for monitoring purposes. However, this real-time tracking path needs to extend their application for moving in curvature motion. Based on this fact, the real-time autonomous navigation system of USV will improve in this research by implementing the mathematical equation that will communicate with the GPS sensor. (Puteri Nur Farhanah, 2020)

Figure 2.7 Shows the location on the google map

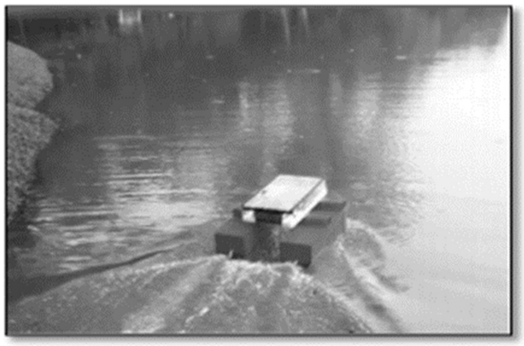


Figure 2.8 USV platform searching the target platform location

2.5 Water Quality Monitoring System Based on IOT

The researchers present a design and development of a low-cost system for real time monitoring of the water quality in Internet of Things (IoT). The system consist of several sensors is used to measuring physical and chemical parameters of the water. The parameters such as temperature, pH, turbidity and flow sensor of the water can be measured. The measured values from the sensors can be processed by the core controller which is Atmega328. Finally, the sensor data can be viewed on internet using WIFI system. (I. Y. Amran,, 2017)

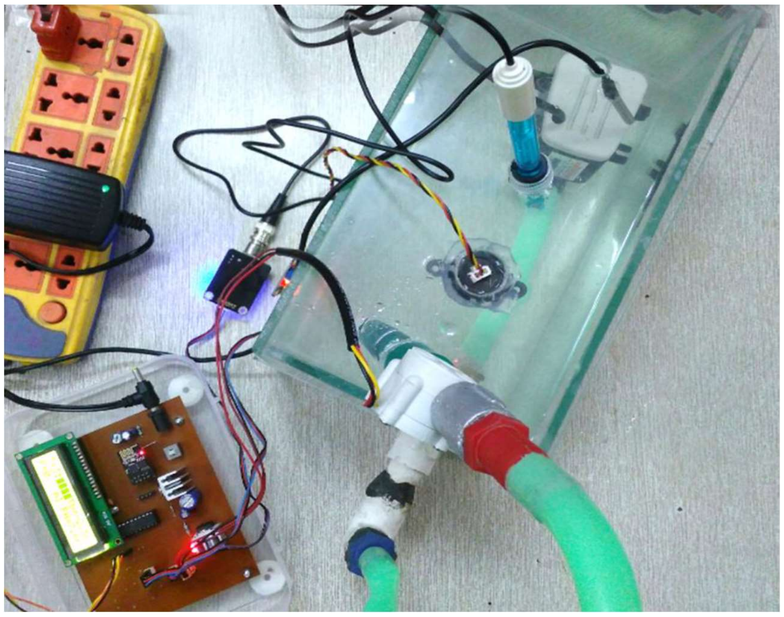


Figure 2.9 System design model of water quality monitoring system using IoT

2.6 Water Monitoring and Analytic Based Thingspeak

This paper presents a water quality monitoring using IoT based Thingspeak platform that provides analytic tools and visualization using MATLAB programming. The proposed model is used to test water samples using sensor fusion technique such as TDS and Turbidity. The data will then be uploaded online to Thingspeak platform to monitor and analyse. The system notifies authorities when there are water quality parameters out of a predefined set of normal values. A warning will be notified to user by IFTTT protocol. (Abbas Hussain, 2020)

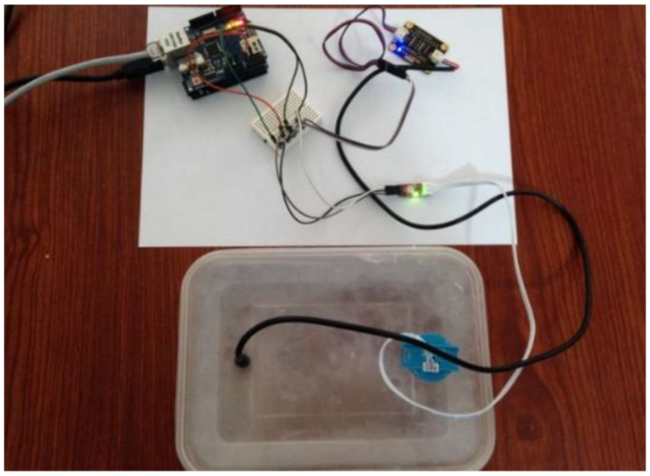
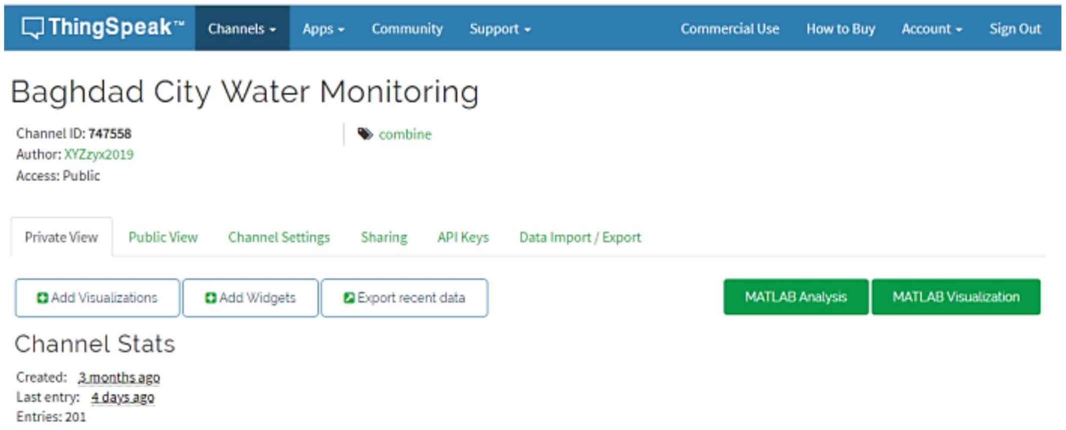


Figure 2.10 Implementation of the proposed system



Baghdad City Water Monitoring

Channel ID: 747558
Author: XYZzyx2019
Access: Public

combine

Channel Stats

Created: 3 months ago
Last entry: 4 days ago
Entries: 201

Figure 2.11 Thingspeak cannel of proposed system

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, the methodology of the project will be explained in details on its platform design selection, flowchart, block diagram and the software configuration

First, the hardware design as it is an important aspect of the overall project. The hardware selection is done based on the literature review done in Chapter 2. Next, the flowchart of the system and the block diagram of the system is explained to show the overall system and its functions. Then the software configuration will be explained how the USV software is developed using 2 platform that are MIT App Inventor and Arduino IDE. Finally, the use of ThingSpeak will also be explained.

3.2 USV Hardware Design

The USV design are based on the hull design, the mechanical system and the controller type chosen.

3.2.1 USV Design

The USV we designed based on the Solidwork drawing. The sizes and dimensions were determined before we apply to the hands-on works. The USV were designed using plywood. We cut the plywood in segments of parts and attached all of them with screws and wood glue. Then, the USV were coated with the fibreglass and the resin. The purpose of the fiberglass and resin were to ensure the USV is waterproof and able to float in the water. The fibreglass process took a lot of effort since we had to coat the USV with 3 layers of fiberglass. The layer was sanded every time each layer is completely dried.

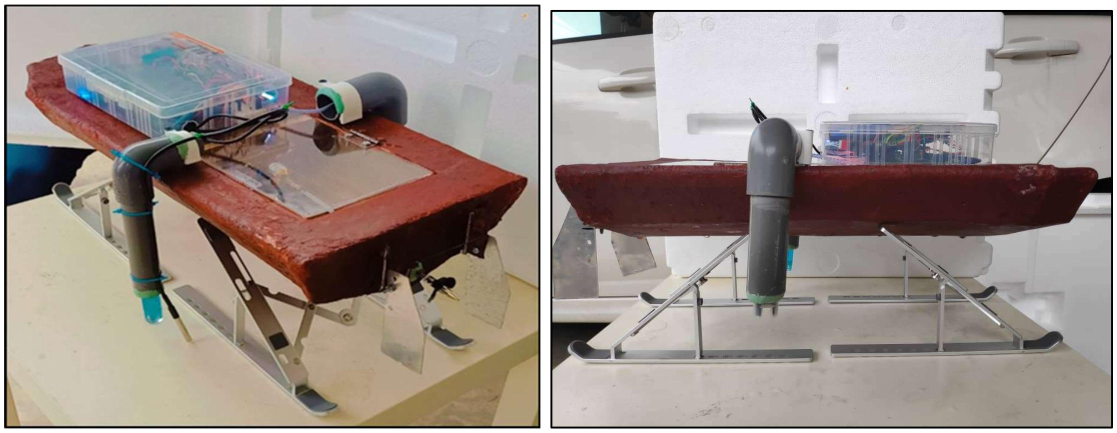


Figure 3.1 USV that has been developed

The sensors were mounted using the PVC pipe to hold the sensors and wires. This ensures that the wires will not be touching any water. This is one of our safety precautions to avoid electrocution when operating the USV. The microcontroller which is WEMOS MEGA+WIFI and electrical connections were put inside the plastic case.

3.2.2. Mechanical System

The mechanical system of our USV consists of two parts which is the thruster and the rudder. The concept of the system will be explained below.

3.2.2.1 Thruster

The thruster of our USV used a brushless motor and an Electronic Speed Controller (ESC). The motor is powered by a Li-Po 11.1 V, 2200mAh battery. The motor rating is 2200kV 6T. The motor is attached to a 25cm motor shaft and a propeller.

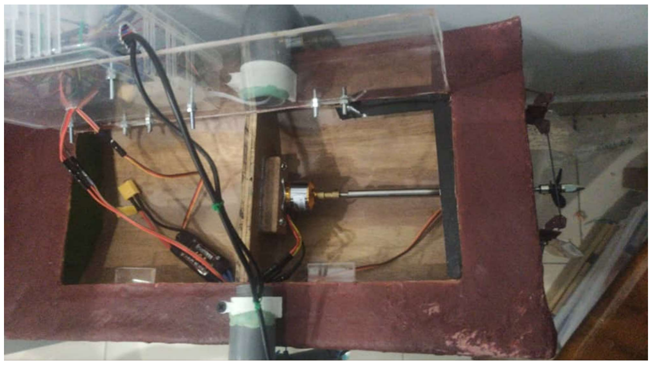


Figure 3.2 Thruster and ESC

3.2.2.2 Rudder

The rudder was built using the aluminium sheets and door hinges. The door hinges were attached to the back of the USV. The sheets size is around 100mm x 40mm. The rudder is moved by a SG90 micro servo. The angle of rotation of the servo were set to be 120°.

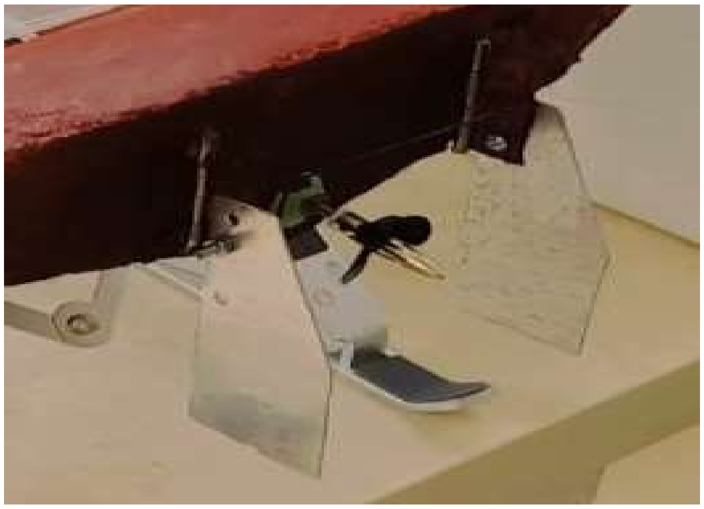


Figure 3.3 Rudder

3.2.3 Controller

The USV is controlled by using an android application that was built using MIT App Inventor. This was done by using a wireless local area network. A modem with internet connection is needed in order to create the access point (AP) so that the USV hardware can connect to the internet. When the USV is powered on, the microcontroller will initialize a connection to the internet. This will create a webserver that will be used as the controller medium.

3.3 Flowchart

This flowchart shows the working principle of the overall system including the android application and the Arduino coding.

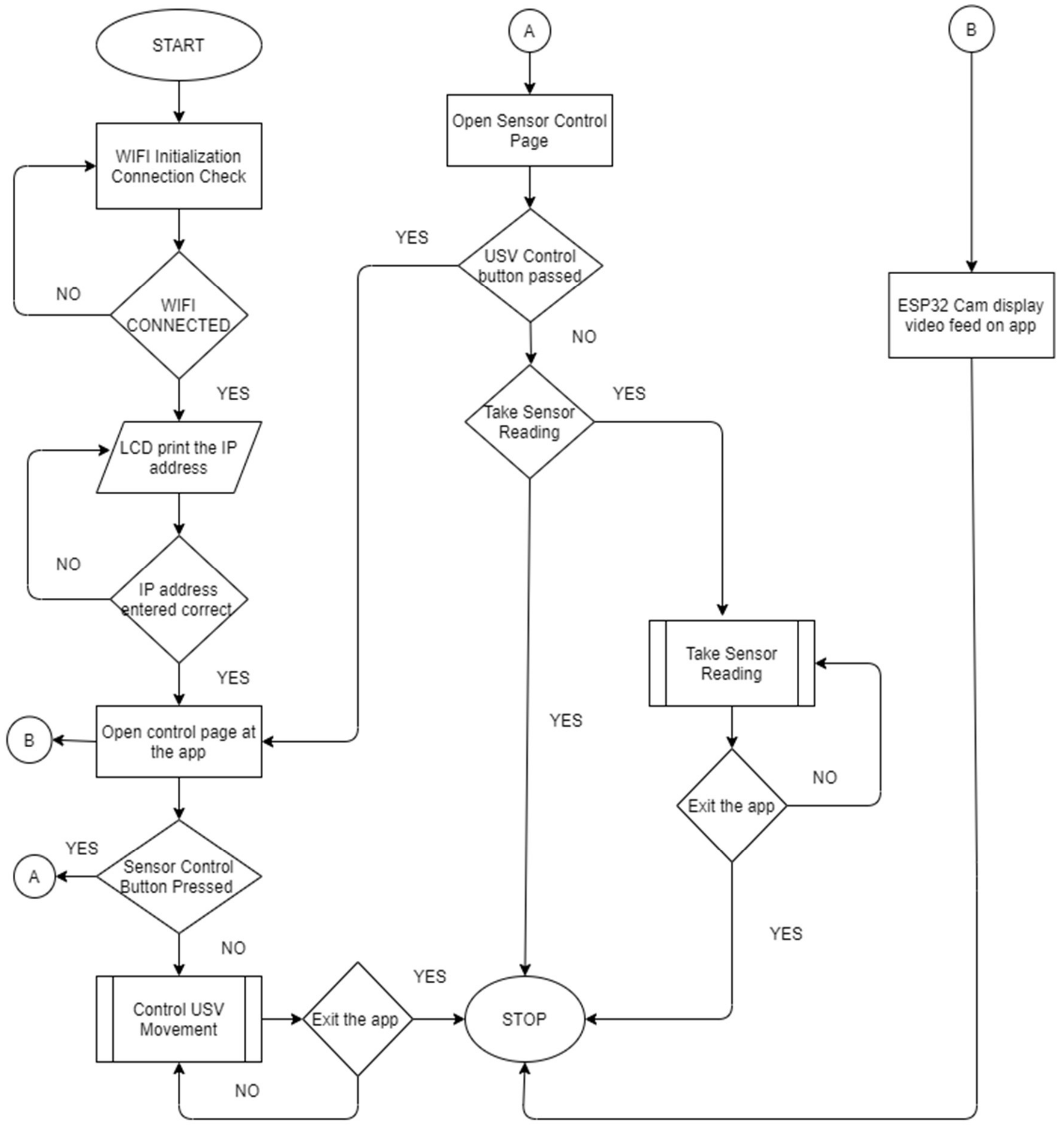


Figure 3.4 Flowchart of the overall system

In Figure 3.4, the flowchart shows the overall system and its subfunctions such as USV Controller and Water Quality system.

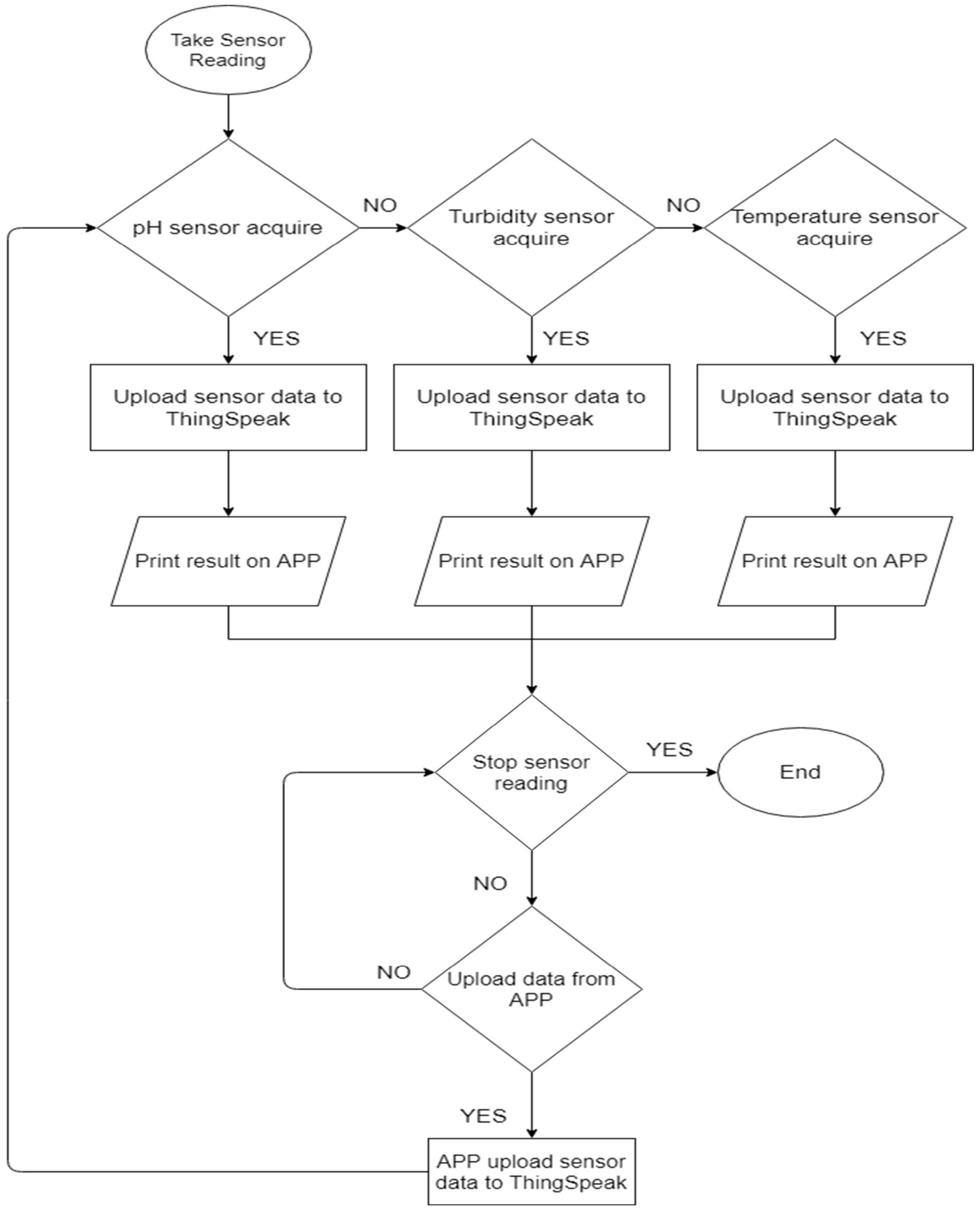


Figure 3.5 Flowchart of the Water Quality Monitoring System

Figure 3.5 shows the subfunction for the Water Quality Monitoring System which takes the data from pH sensor, turbidity sensor and temperature sensor.

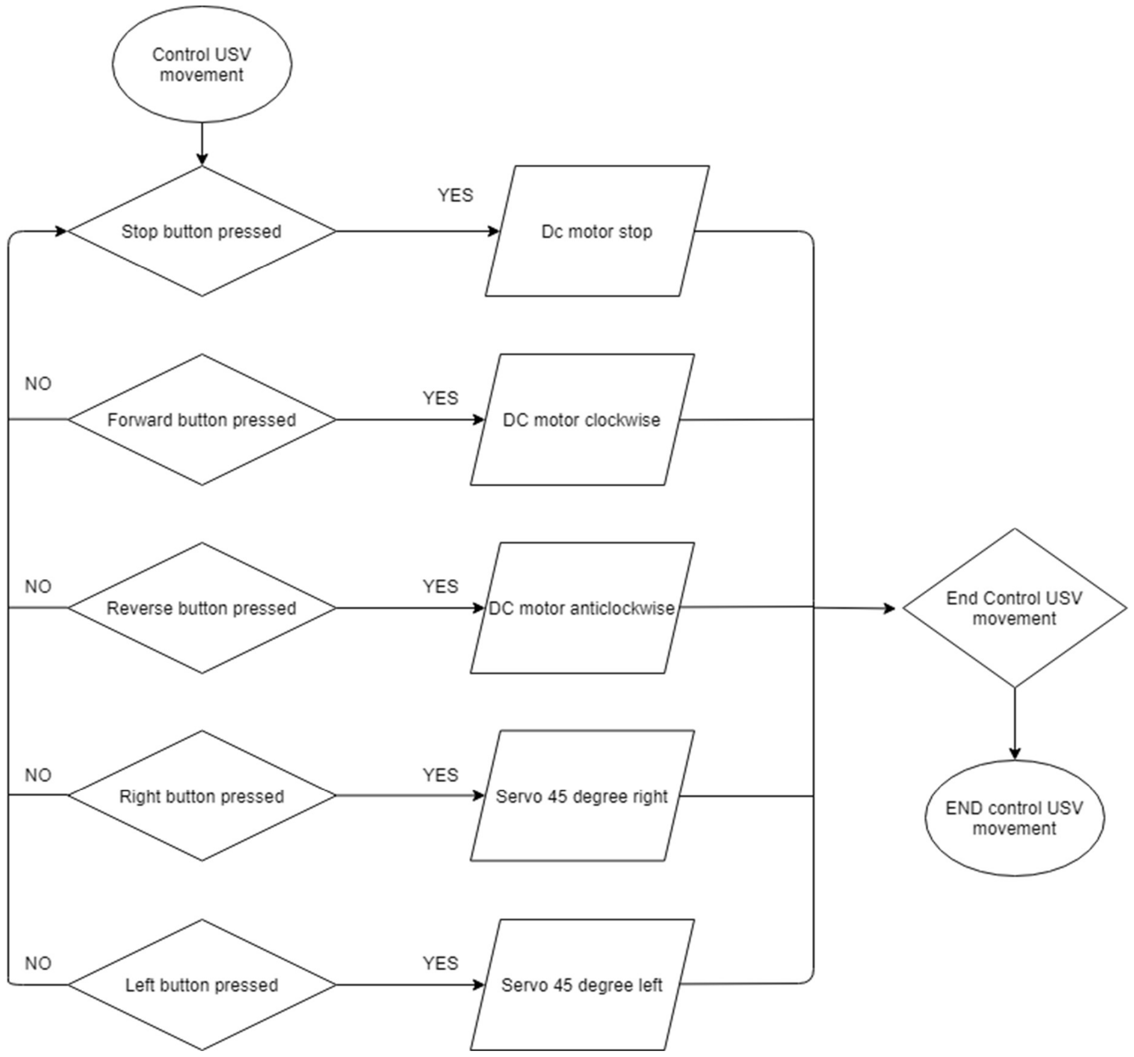


Figure 3.6 Flowchart of USV Control System

In Figure 3.6 shows the flowchart of USV Control System which controls the brushless motor as the thruster and the micro servo as the rudder.

3.4. Block Diagram

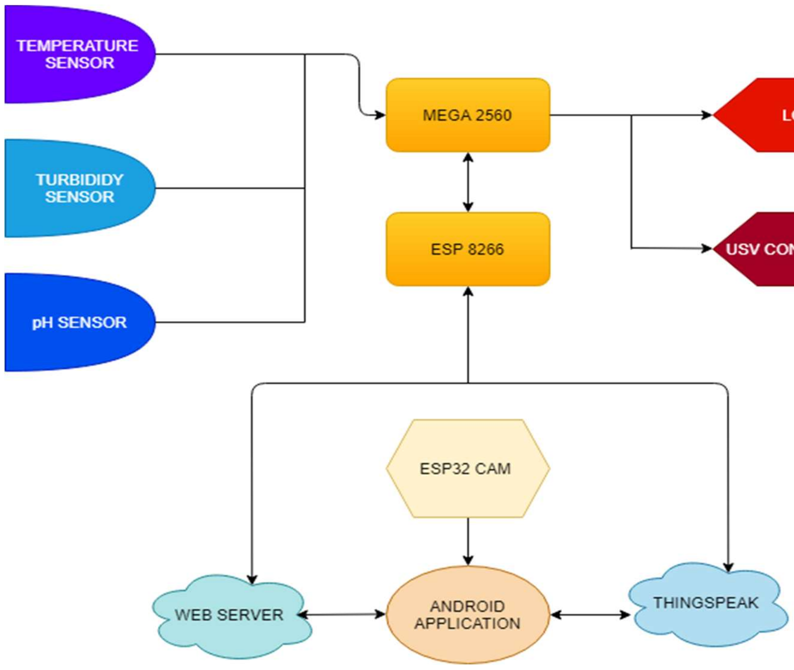


Figure 3.7 Block diagram of the overall USV system

Figure 3.7 shows the block diagram of the overall USV system based on the WEMOS MEGA+WIFI which is built in with MEGA 2560 and ESP 8266. The sensors, LCD and USV controller were programmed in the MEGA 2560. The ESP 8266 will communicate with the Web Server, Thingspeak and the android application.

3.5 Software Configuration and Development

3.5.1 WEMOS MEGA+WIFI

This project uses the WEMOS MEGA+WIFI as the microcontroller. It allows flexible configurations for connections for the connection between Atmega2560, ESP8266 and USB serial. The configurations are sent using the DIP switches This is because it is built in with MEGA 2560 and ESP 8266. By using this microcontroller, we can reduce the complication in connecting the MEGA 2560 and ESP 8266.

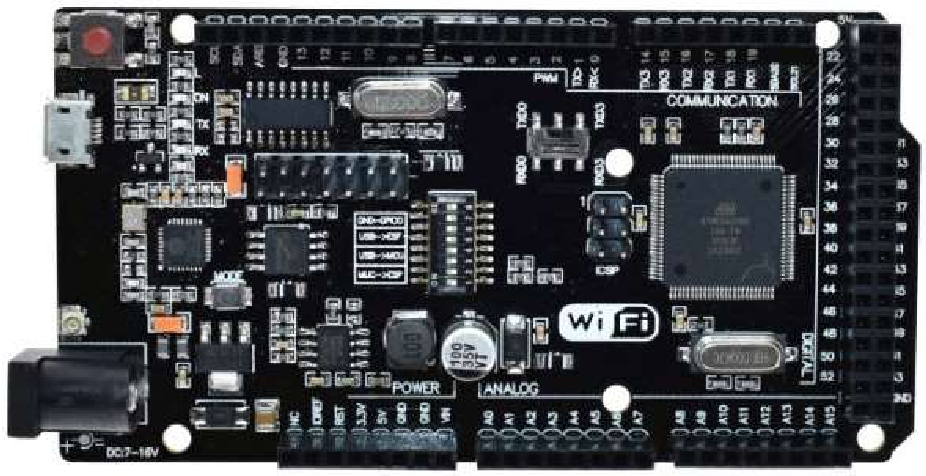
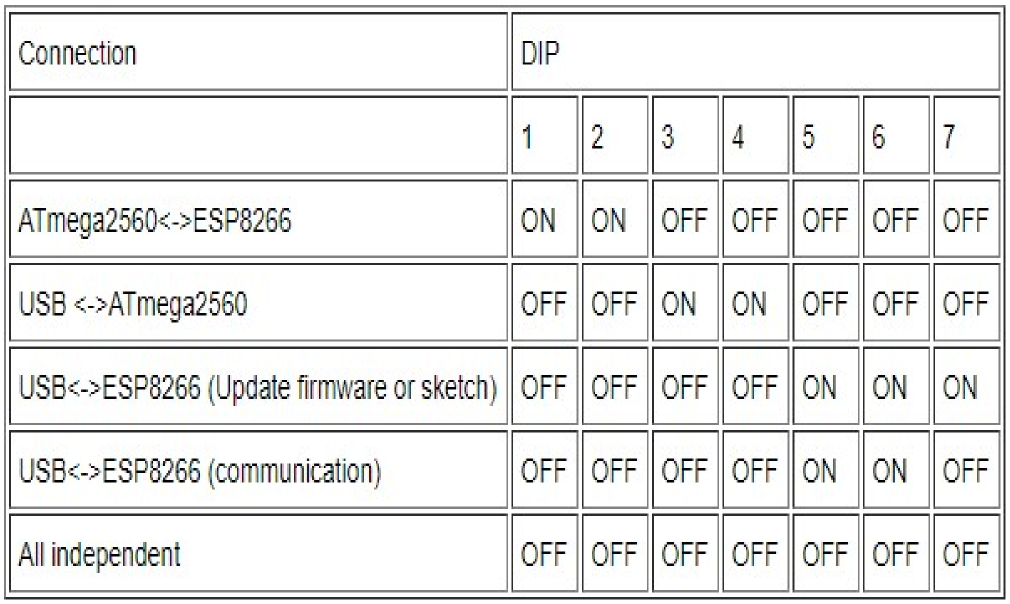
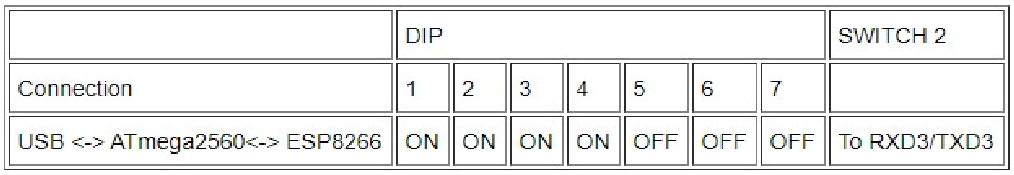


Figure 3.8 WEMOS MEGA+WIFI



Connection	DIP						
	1	2	3	4	5	6	7
ATmega2560<->ESP8266	ON	ON	OFF	OFF	OFF	OFF	OFF
USB <->ATmega2560	OFF	OFF	ON	ON	OFF	OFF	OFF
USB<->ESP8266 (Update firmware or sketch)	OFF	OFF	OFF	OFF	ON	ON	ON
USB<->ESP8266 (communication)	OFF	OFF	OFF	OFF	ON	ON	OFF
All independent	OFF	OFF	OFF	OFF	OFF	OFF	OFF



	DIP							SWITCH 2
Connection	1	2	3	4	5	6	7	
USB <-> ATmega2560<-> ESP8266	ON	ON	ON	ON	OFF	OFF	OFF	To RXD3/TXD3

Figure 3.9 DIP switch configuration for mode selection

The coding is uploaded differently which are to MEGA 2560 and ESP 8266. This MEGA 2560 will take care of the sensors and motors that controls the USV function. ESP 8266 will handle the Wi-Fi connectivity and host the webserver that are used to communicate with the mobile application and ThingSpeak.

Microcontroller	MEGA 2560	WEMOS MEGA+WIFI
Processor	Atmega2560	Atmega2560, ESP 8266
Operating Voltage	5/7-12V	5/7-12V,3v3
CPU Speed	16MHz	16MHz, 80MHz
Analog in/Out	16/0	16/0, 1/0
Digital IO/PWM	54/15	54/15
EEPROM [kB]	4kb	4kb
SRAM	8kb	8kb,64kb
Flash [kB]	356kb	32Mb, 8Mb
USB	Regular	CH340G
UART	4	4, 1/Wi-Fi

Table 3.1 Microcontroller Specifications

This WEMOS MEGA+WIFI also have serial connection which are Serial 0 – Serial 3. In this project, we use the Serial 3 of Arduino Mega is used to connect with Serial 1 of the ESP 8266 by switching the switch to the TXD3/RXD3.

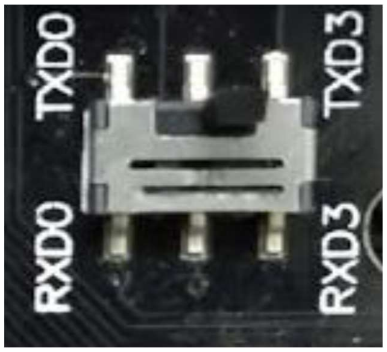
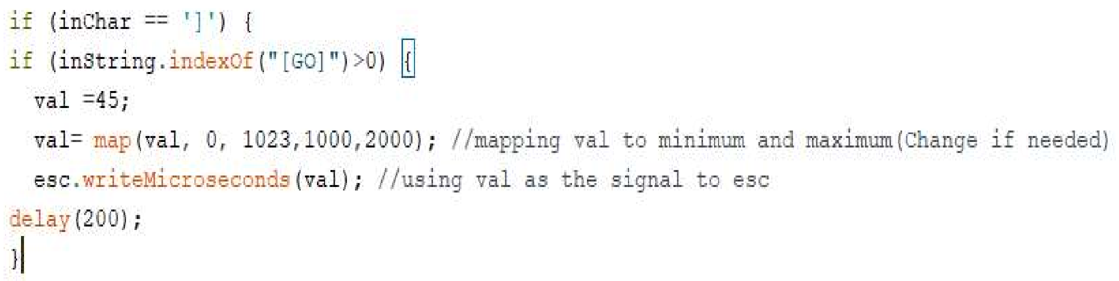


Figure 3.10 Switch for serial selection

3.5.2 Mega 2560 Coding

As mention before, the DIP switch is turned on based on the Figure 3.9 to upload the coding to the Mega 2560. The MEGA 2560 is set to receive command from the ESP 8266 via the Serial 3 read function. The data received are in a form of string or char. Then, Mega 2560 will identify the command received to do a specific code for the particular command. In Figure 3.11 shows that if the Mega 2560 received the command [GO], it will process the codes that turn on the brushless motor.

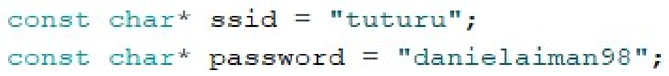


```
if (inChar == ']') {
if (inString.indexOf("[GO]")>0) {
  val =45;
  val= map(val, 0, 1023,1000,2000); //mapping val to minimum and maximum(Change if needed)
  esc.writeMicroseconds(val); //using val as the signal to esc
delay(200);
}
```

Figure 3.11 Example coding of sending data to ESP 8266

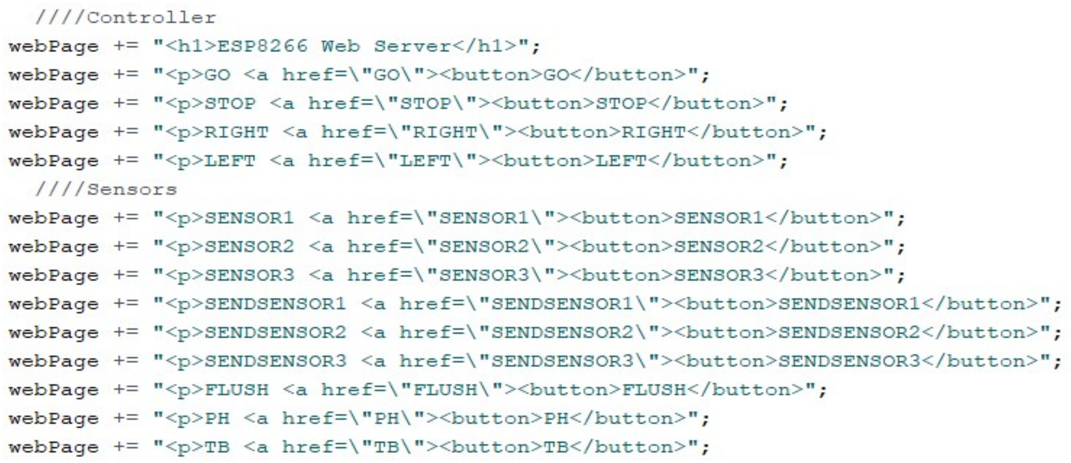
3.5.3 ESP 8266 Coding

The DIP switches are configured based on the Table 3.1 to upload the coding to ESP 8266. Figure 3.12 shows the Wi-Fi credentials needed to be filled for it to connect to the local Wi-Fi. Then, a webserver and its components need to be created. The webserver acts as the middle man between the android application and the ESP 8266. The webserver created the URL using the IP address. The webserver and its codes are shown in Figure 3.13. The button is set so after being press a specific URL path will be access by the webserver. The specific URL path will trigger the specific command to be print by the ESP 8266 serial print as in Figure 3.14.



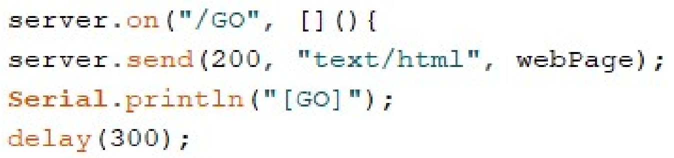
```
const char* ssid = "tuturu";
const char* password = "danielaiman98";
```

Figure 3.12 Credentials parameter of the Wi-Fi



```
  ////Controller
webPage += "<h1>ESP8266 Web Server</h1>";
webPage += "<p>GO <a href=\"GO\"><button>GO</button>";
webPage += "<p>STOP <a href=\"STOP\"><button>STOP</button>";
webPage += "<p>RIGHT <a href=\"RIGHT\"><button>RIGHT</button>";
webPage += "<p>LEFT <a href=\"LEFT\"><button>LEFT</button>";
  ////Sensors
webPage += "<p>SENSOR1 <a href=\"SENSOR1\"><button>SENSOR1</button>";
webPage += "<p>SENSOR2 <a href=\"SENSOR2\"><button>SENSOR2</button>";
webPage += "<p>SENSOR3 <a href=\"SENSOR3\"><button>SENSOR3</button>";
webPage += "<p>SENDSENSOR1 <a href=\"SENDSENSOR1\"><button>SENDSENSOR1</button>";
webPage += "<p>SENDSENSOR2 <a href=\"SENDSENSOR2\"><button>SENDSENSOR2</button>";
webPage += "<p>SENDSENSOR3 <a href=\"SENDSENSOR3\"><button>SENDSENSOR3</button>";
webPage += "<p>FLUSH <a href=\"FLUSH\"><button>FLUSH</button>";
webPage += "<p>PH <a href=\"PH\"><button>PH</button>";
webPage += "<p>TB <a href=\"TB\"><button>TB</button>";
```

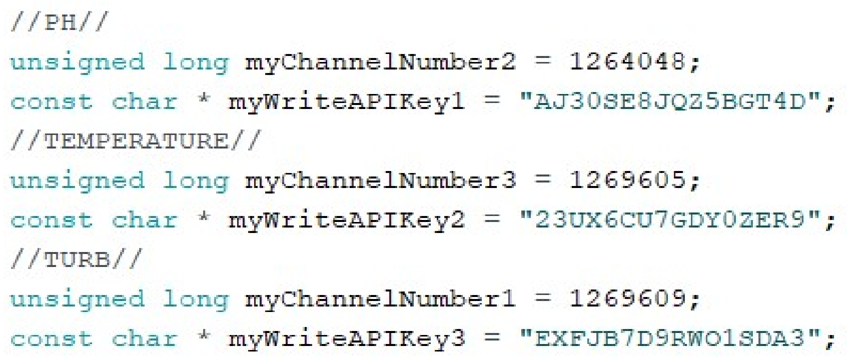
Figure 3.13 Webserver setup



```
server.on("/GO", [](){
server.send(200, "text/html", webPage);
Serial.println("[GO]");
delay(300);
```

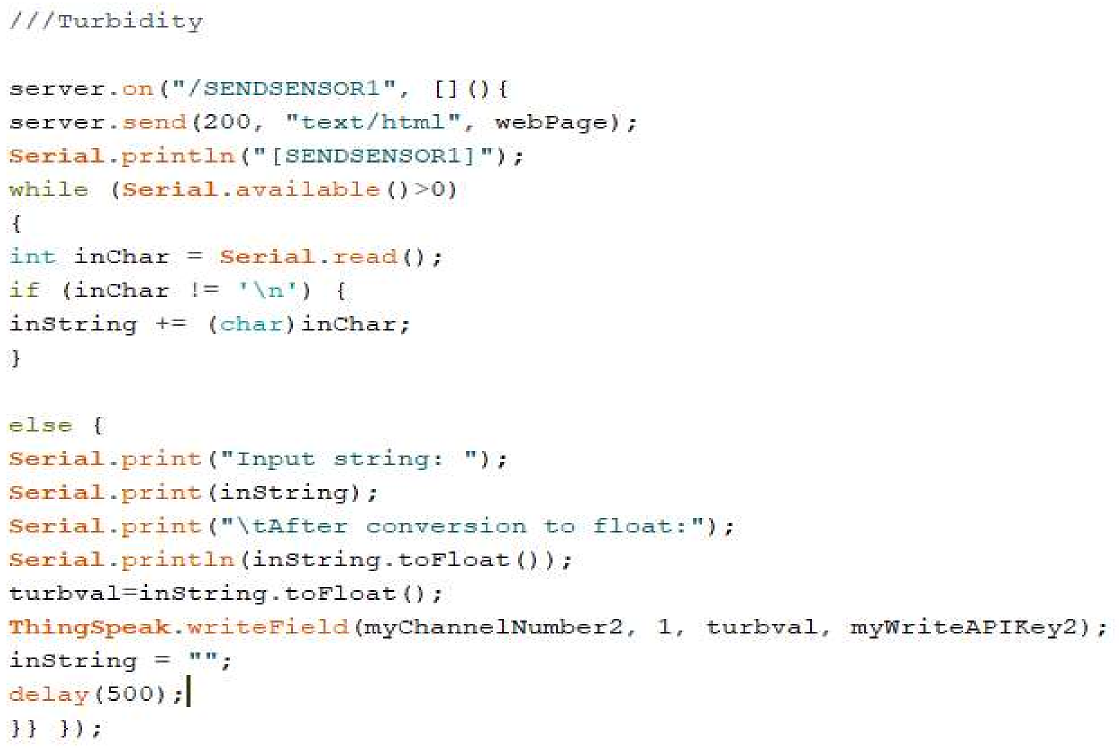
Figure 3.14 Command to be send to Arduino Mega

Next, to upload the acquired sensor data, the data will need to be received by the ESP8266. The data sent by Mega 2560 are received in form of string or char then it will be converted to float before can be upload to the ThingSpeak. Figure 3.15 show the channel number and write API key for the specific field needed to be set for the specific sensor.



```
//PH//
unsigned long myChannelNumber2 = 1264048;
const char * myWriteAPIKey1 = "AJ30SE8JQZ5BGT4D";
//TEMPERATURE//
unsigned long myChannelNumber3 = 1269605;
const char * myWriteAPIKey2 = "23UX6CU7GDY0ZER9";
//TURB//
unsigned long myChannelNumber1 = 1269609;
const char * myWriteAPIKey3 = "EXFJB7D9RWO1SDA3";
```

Figure 3. 15 Channel number and write API key for ThingSpeak.



```
///Turbidity

server.on("/SENDSENSOR1", [](){
server.send(200, "text/html", webPage);
Serial.println("[SENDSENSOR1]");
while (Serial.available()>0)
{
int inChar = Serial.read();
if (inChar != '\n') {
inString += (char)inChar;
}

else {
Serial.print("Input string: ");
Serial.print(inString);
Serial.print("\tAfter conversion to float:");
Serial.println(inString.toFloat());
turbval=inString.toFloat();
ThingSpeak.writeField(myChannelNumber2, 1, turbval, myWriteAPIKey2);
inString = "";
delay(500);
}} });
```

Figure 3. 16 Example of codes for turbidity sensor

3.6 ESP32 Cam

In this project, ESP32 Cam is used as the IP camera which will show the video feeds on the android application of the view from the front of the USV. The ESP32 board is equipped with camera module, OV2640. This board supports the image Wi-Fi upload and built in with smallest 802.11b/g/n Wi-Fi BT SoC module.

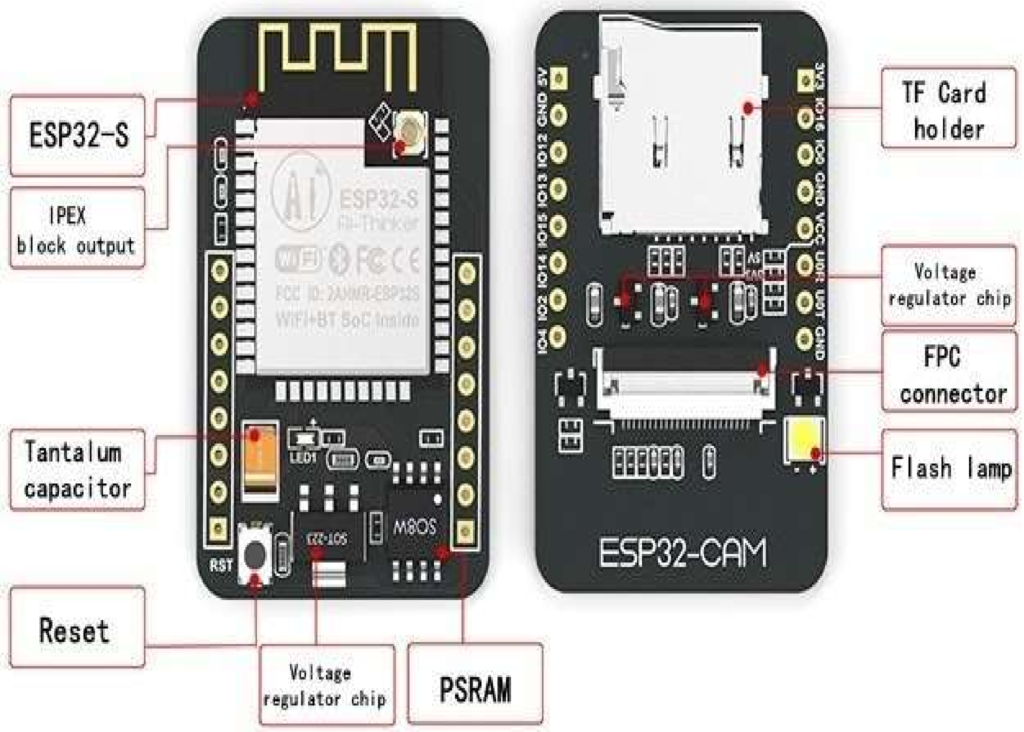
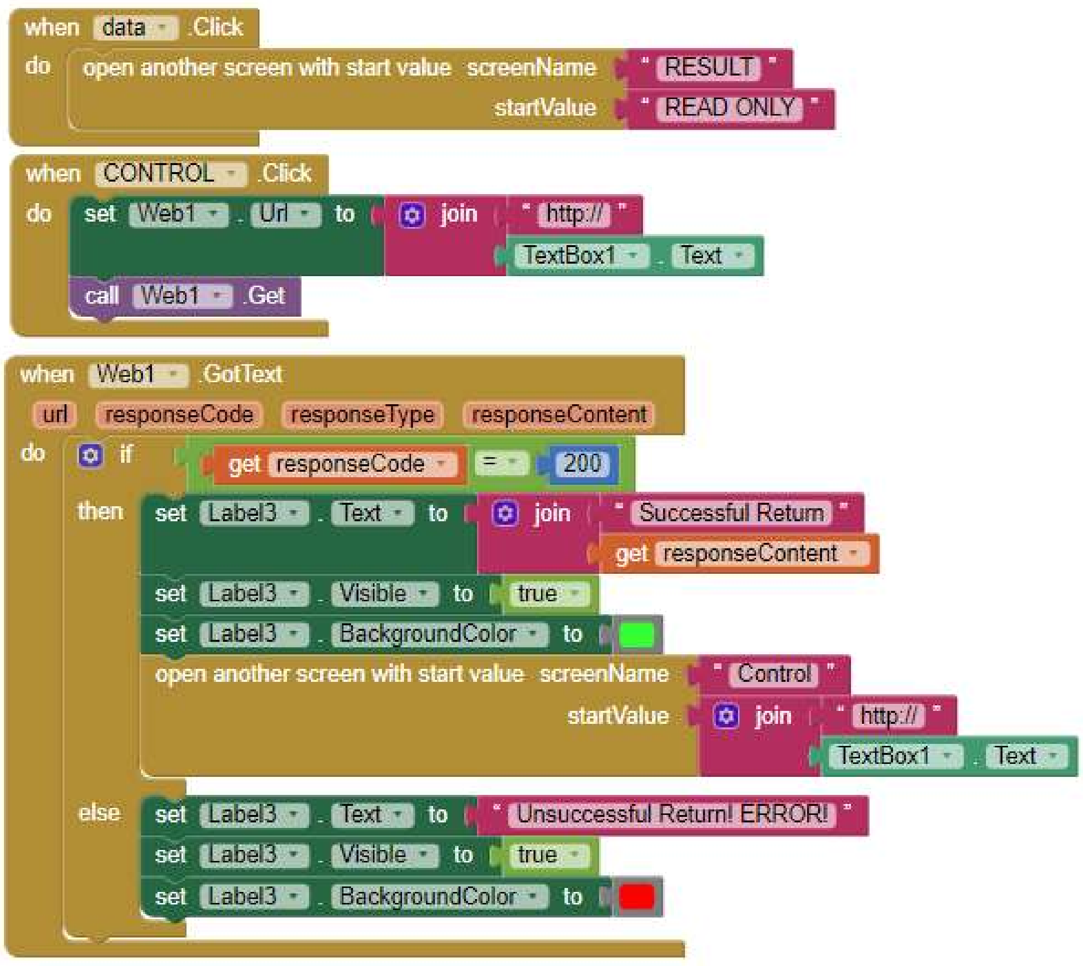


Figure 3.17 ESP32 Cam diagram

3.7. Mobile Application

The application is developed using MIT App Inventor, an online platform to create applications by dragging and dropping the components into a design view and using a visual blocks language to program the application behaviour. The Graphical User Interface (GUI) of the mobile application consist of 3 pages which are welcome page, control page, and result page.



```
when data .Click
do  open another screen with start value  screenName  " RESULT "
                                          startValue  " READ ONLY "

when CONTROL .Click
do  set Web1 . Url to  join  " http:// "
                             TextBox1 . Text
    call Web1 .Get

when Web1 .GotText
 url  responseCode  responseType  responseContent
do  if  get responseCode = 200
    then  set Label3 . Text to  join  " Successful Return "
                                      get responseContent
          set Label3 . Visible to  true
          set Label3 . BackgroundColor to 🟩
          open another screen with start value  screenName  " Control "
                                                 startValue  join  " http:// "
                                                                   TextBox1 . Text
    else  set Label3 . Text to  " Unsuccessful Return! ERROR! "
          set Label3 . Visible to  true
          set Label3 . BackgroundColor to 🟥
```

Figure 3.18 The block codes example for welcome page

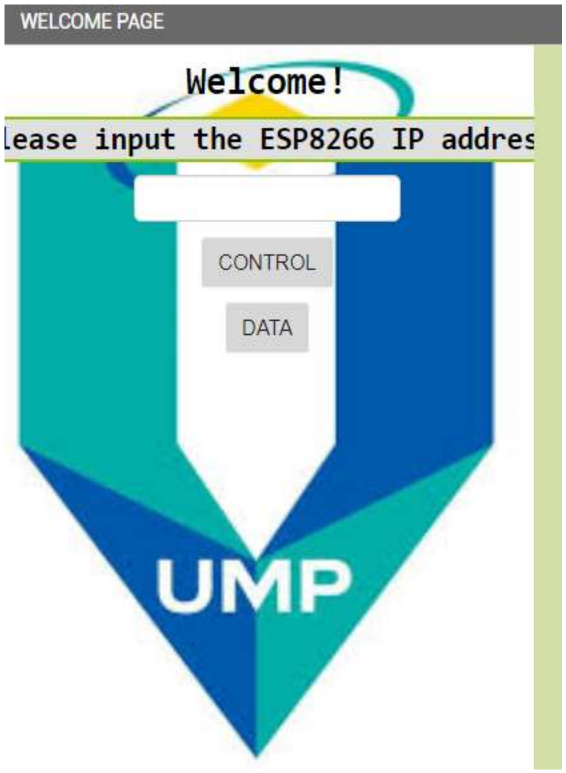


Figure 3.19 The GUI of welcome page

The user will need to enter the IP address printed on the LCD mounted on the USV in order to control it but if the user only want to view the previous acquired data, the user can press the data button. The control button will connect the application to the IP address entered and access the IP address as URL hostname. The button will then bring the user to the next page which is the control page.

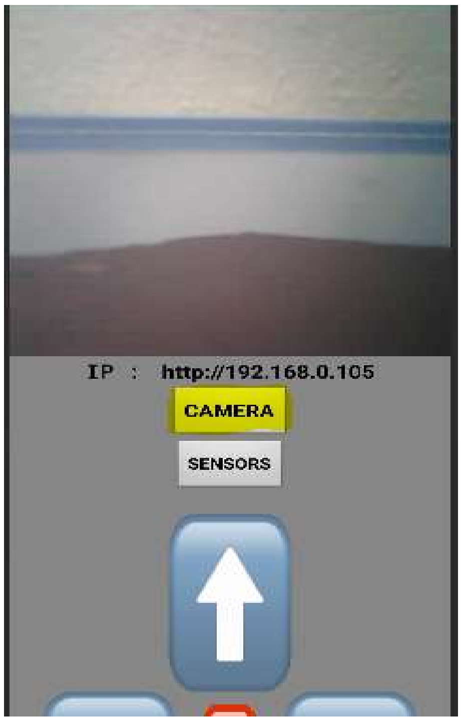
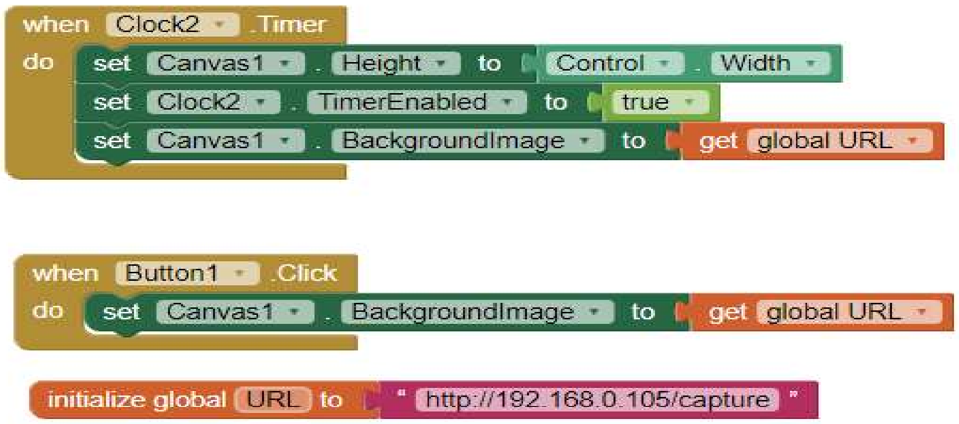


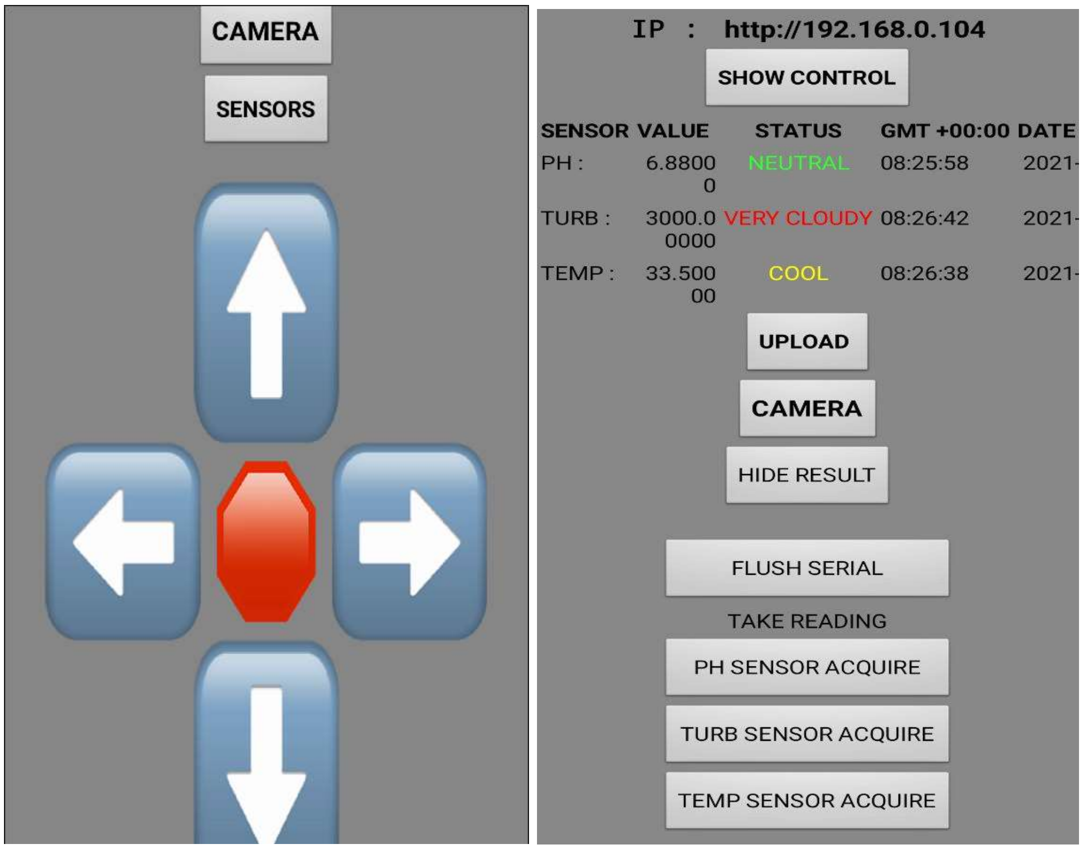
Figure 3.20 Camera function of the ESP32 Cam



```
when Clock2 .Timer
do  set Canvas1 . Height to Control . Width
    set Clock2 . TimerEnabled to true
    set Canvas1 . BackgroundImage to get global URL

when Button1 .Click
do  set Canvas1 . BackgroundImage to get global URL

initialize global URL to " http://192.168.0.105/capture "
```

Figure 3.21 Blocks for Camera function

In Figure 3.21 shows the blocks for the ESP32 Cam that will show the feeds of the USV front view. The ESP32 Cam is connected to the local Wi-Fi and established an IP address. The IP address is used to connect the camera function of the application to the ESP32 Cam.



SENSOR	VALUE	STATUS	GMT +00:00	DATE
PH :	6.88000	NEUTRAL	08:25:58	2021-
TURB :	3000.00000	VERY CLOUDY	08:26:42	2021-
TEMP :	33.50000	COOL	08:26:38	2021-

Figure 3.22 The control page of the mobile application.

The Figure 3.22 show sthe control page of the USV where we can control the USV movement. The specific buttons will accesed the specific URL. For example, if the user pressed the forward button, the application will access the URL as 192.168.0.104/GO. This action will give command to ESP 8266 to print the specific command to MEGA 2560. Then the MEGA 2560 will execute specific process for moving the USV.

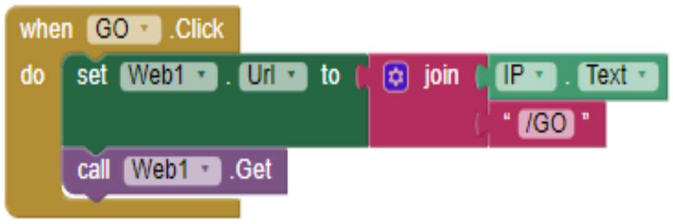
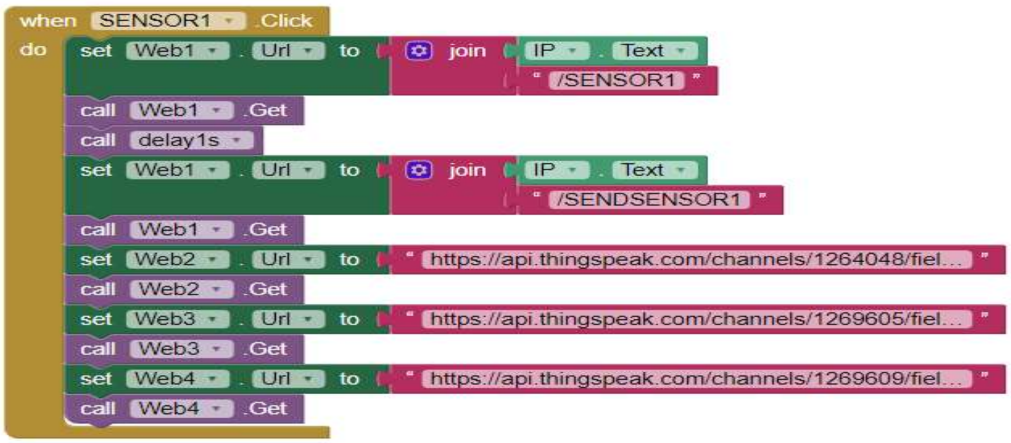


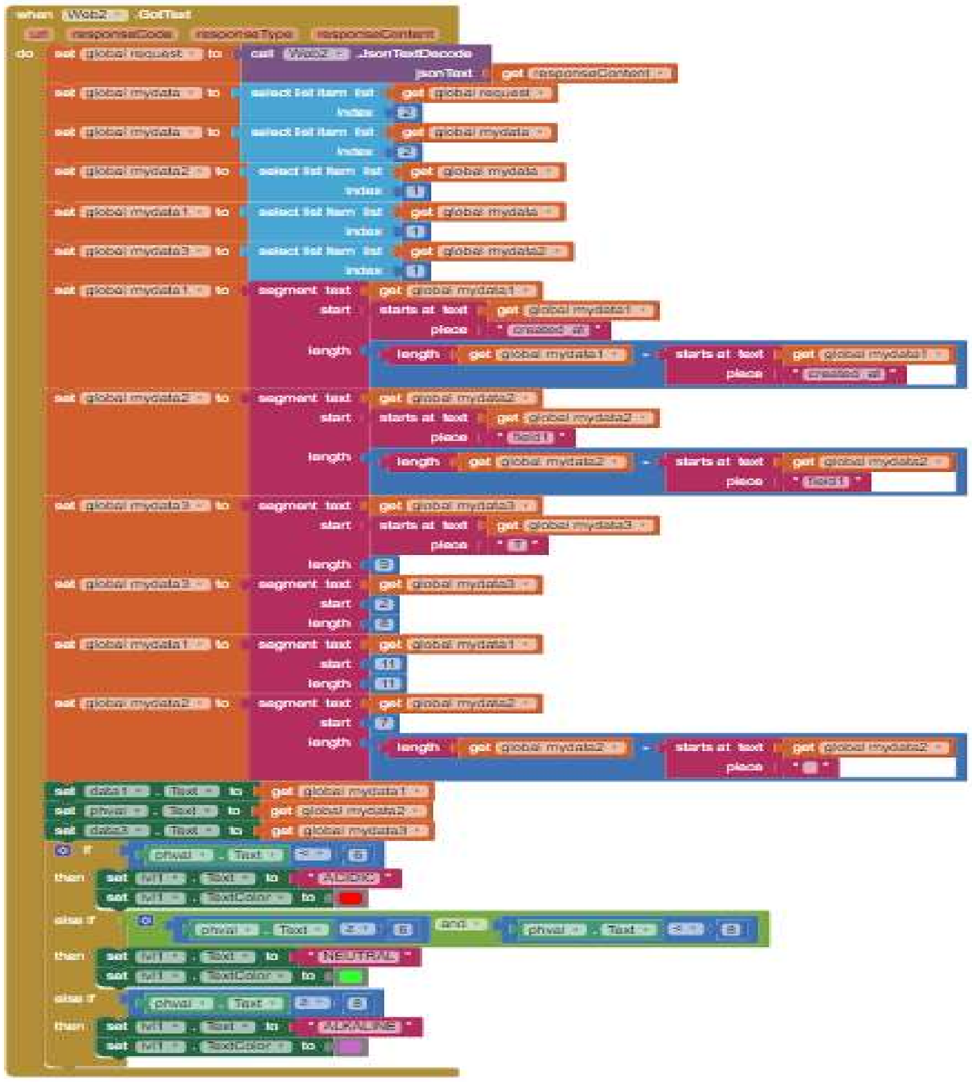
Figure 3.23 Blocks for GO button



```
when SENSOR1 .Click
do  set Web1 . Url to  join  IP . Text
                             " /SENSOR1 "
    call Web1 .Get
    call delay1s
    set Web1 . Url to  join  IP . Text
                             " /SENDSENSOR1 "
    call Web1 .Get
    set Web2 . Url to " https://api.thingspeak.com/channels/1264048/fiel... "
    call Web2 .Get
    set Web3 . Url to " https://api.thingspeak.com/channels/1269605/fiel... "
    call Web3 .Get
    set Web4 . Url to " https://api.thingspeak.com/channels/1269609/fiel... "
    call Web4 .Get
```

Figure 3.24 Example blocks to send data to ThingSpeak

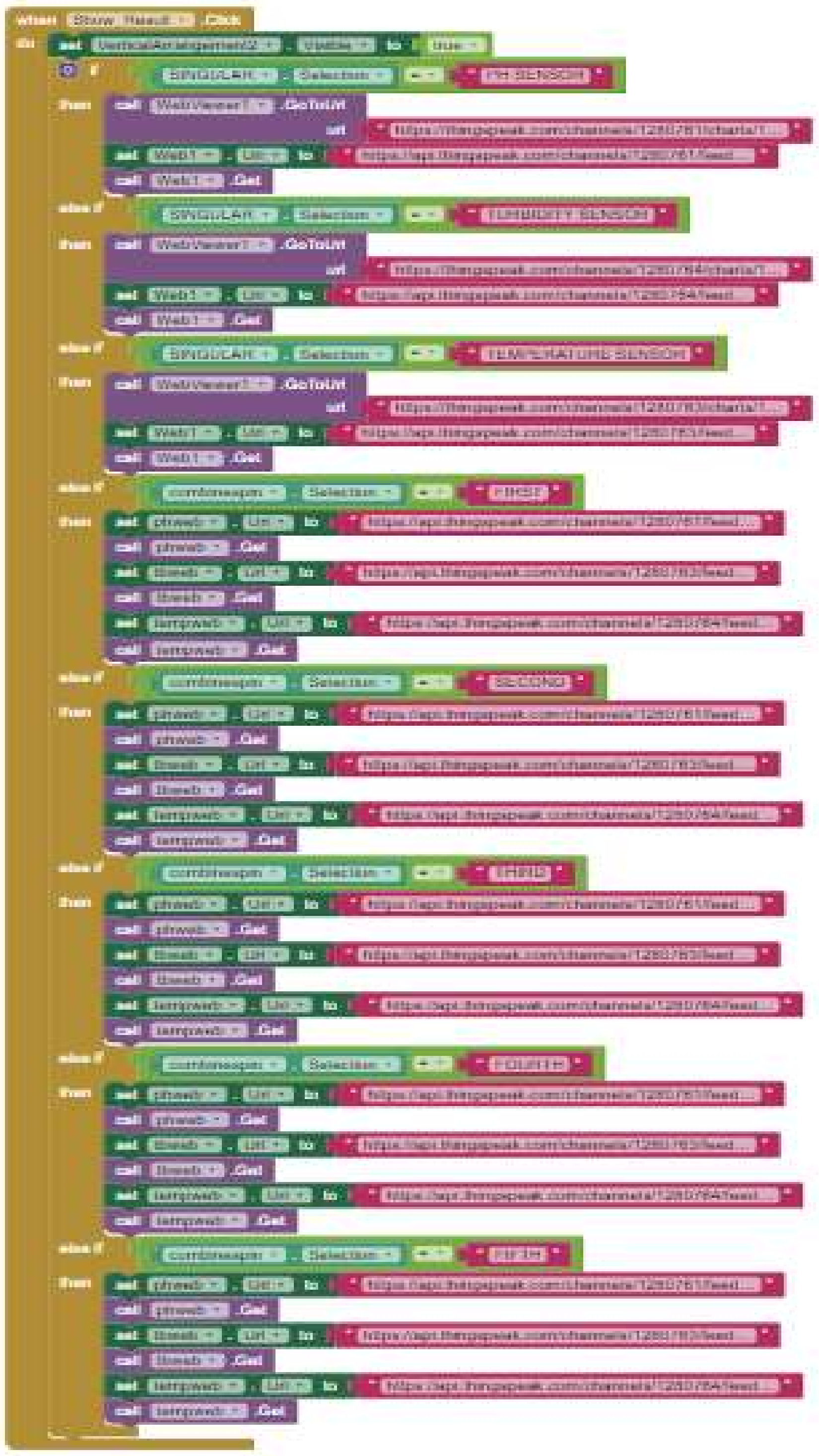
Figure 3.24 shows the blocks that works to send the sensor data to the ThingSpeak. This link https://api.thingspeak.com/channels/1264048/fields/1.json?results=1 contain the unique channel id on the ThingSpeak. For example, the channel id for this example is 126048. The application will access the ThingSpeak channel through its channel id.



```
when Web2 .GotText
  url  responseCode  responseType  responseContent
do  set global request to  call Web2 .JsonTextDecode
                                   jsonText  get responseContent
    set global mydata to  select list item  list  get global request
                                            index 2
    set global mydata to  select list item  list  get global mydata
                                            index 2
    set global mydata2 to  select list item  list  get global mydata
                                             index 1
    set global mydata1 to  select list item  list  get global mydata
                                             index 1
    set global mydata3 to  select list item  list  get global mydata2
                                             index 1
    set global mydata1 to  segment  text  get global mydata1
                                    start  starts at  text  get global mydata1
                                                      piece " created_at "
                                    length  length  get global mydata1  -  starts at  text  get global mydata1
                                                                                      piece " created_at "
    set global mydata2 to  segment  text  get global mydata2
                                    start  starts at  text  get global mydata2
                                                      piece " field1 "
                                    length  length  get global mydata2  -  starts at  text  get global mydata2
                                                                                      piece " field1 "
    set global mydata3 to  segment  text  get global mydata3
                                    start  starts at  text  get global mydata3
                                                      piece " f "
                                    length 8
    set global mydata3 to  segment  text  get global mydata3
                                    start 2
                                    length 2
    set global mydata1 to  segment  text  get global mydata1
                                    start 11
                                    length 11
    set global mydata2 to  segment  text  get global mydata2
                                    start 7
                                    length  length  get global mydata2  -  starts at  text  get global mydata2
                                                                                      piece " " "
    set data1 . Text to  get global mydata1
    set phval . Text to  get global mydata2
    set data3 . Text to  get global mydata3
    if  phval . Text  <  6
    then  set lvl1 . Text to " ACIDIC "
          set lvl1 . TextColor to 🟥
    else if  phval . Text  ≥  6  and  phval . Text  ≤  8
    then  set lvl1 . Text to " NEUTRAL "
          set lvl1 . TextColor to 🟩
    else if  phval . Text  ≥  8
    then  set lvl1 . Text to " ALKALINE "
          set lvl1 . TextColor to 🟪
```

Figure 3.25 Example blocks for result page

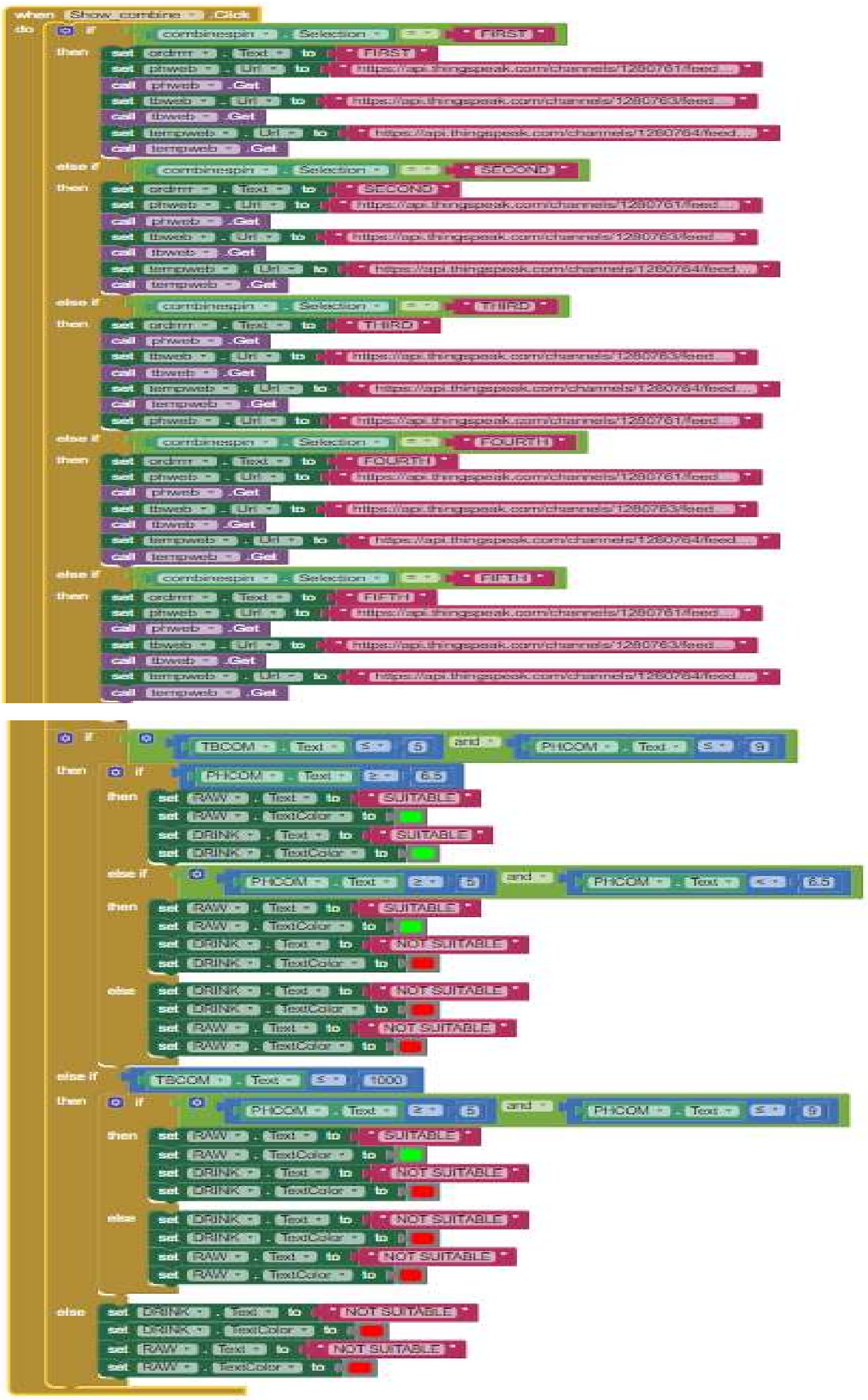
Figure 3.25 shows the blocks of the result page. This block will show the user the status of the acquired sensor data. For example, for the pH sensor data, the application will show the user the value of the pH and whether it is acidic, alkaline or neutral.



```
when Show_Result .Click
do  set VerticalArrangement2 . Visible to true
    if      SINGULAR . Selection = "PH SENSOR"
    then    call WebViewer1 .GoToUrl
                        url "https://thingspeak.com/channels/1280761/charts/1..."
            set Web1 . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call Web1 .Get
    else if SINGULAR . Selection = "TURBIDITY SENSOR"
    then    call WebViewer1 .GoToUrl
                        url "https://thingspeak.com/channels/1280784/charts/1..."
            set Web1 . Url to "https://api.thingspeak.com/channels/1280784/feed..."
            call Web1 .Get
    else if SINGULAR . Selection = "TEMPERATURE SENSOR"
    then    call WebViewer1 .GoToUrl
                        url "https://thingspeak.com/channels/1280783/charts/1..."
            set Web1 . Url to "https://api.thingspeak.com/channels/1280783/feed..."
            call Web1 .Get
    else if combinespin . Selection = "FIRST"
    then    set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280783/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280784/feed..."
            call tempweb .Get
    else if combinespin . Selection = "SECOND"
    then    set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280783/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280784/feed..."
            call tempweb .Get
    else if combinespin . Selection = "THIRD"
    then    set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280783/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280784/feed..."
            call tempweb .Get
    else if combinespin . Selection = "FOURTH"
    then    set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280783/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280784/feed..."
            call tempweb .Get
    else if combinespin . Selection = "FIFTH"
    then    set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280783/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280784/feed..."
            call tempweb .Get
```

Figure 3.26 Example blocks for single result function

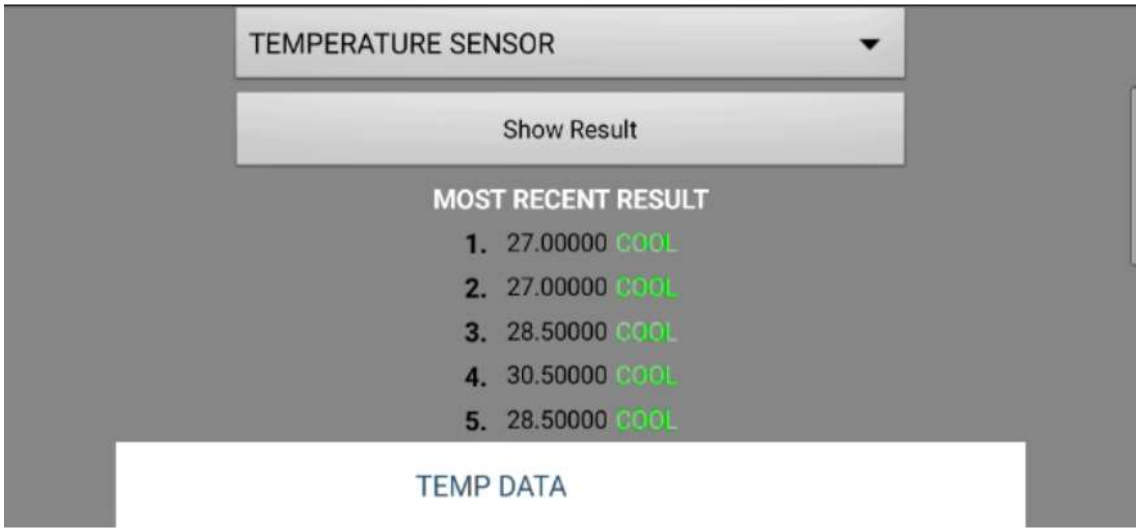
In Figure 3.26 shows the blocks for single result function which will retrieve the acquired sensor data from the ThingSpeak. The data will be retrieved based on each of ThingSpeak channel for each sensor.



```
when Show_combine .Click
do  if      combinespin . Selection = "FIRST"
    then    set ordrrr . Text to "FIRST"
            set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280763/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280764/feed..."
            call tempweb .Get
    else if combinespin . Selection = "SECOND"
    then    set ordrrr . Text to "SECOND"
            set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280763/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280764/feed..."
            call tempweb .Get
    else if combinespin . Selection = "THIRD"
    then    set ordrrr . Text to "THIRD"
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280763/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280764/feed..."
            call tempweb .Get
            set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
    else if combinespin . Selection = "FOURTH"
    then    set ordrrr . Text to "FOURTH"
            set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280763/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280764/feed..."
            call tempweb .Get
    else if combinespin . Selection = "FIFTH"
    then    set ordrrr . Text to "FIFTH"
            set phweb . Url to "https://api.thingspeak.com/channels/1280761/feed..."
            call phweb .Get
            set tbweb . Url to "https://api.thingspeak.com/channels/1280763/feed..."
            call tbweb .Get
            set tempweb . Url to "https://api.thingspeak.com/channels/1280764/feed..."
            call tempweb .Get

    if      TBCOM . Text ≤ 5  and  PHCOM . Text ≤ 9
    then    if      PHCOM . Text ≥ 6.5
            then    set RAW . Text to "SUITABLE"
                    set RAW . TextColor to 🟩
                    set DRINK . Text to "SUITABLE"
                    set DRINK . TextColor to 🟩
            else if PHCOM . Text ≥ 5  and  PHCOM . Text < 6.5
            then    set RAW . Text to "SUITABLE"
                    set RAW . TextColor to 🟩
                    set DRINK . Text to "NOT SUITABLE"
                    set DRINK . TextColor to 🟥
            else    set DRINK . Text to "NOT SUITABLE"
                    set DRINK . TextColor to 🟥
                    set RAW . Text to "NOT SUITABLE"
                    set RAW . TextColor to 🟥
    else if TBCOM . Text ≤ 1000
    then    if      PHCOM . Text ≥ 5  and  PHCOM . Text ≤ 9
            then    set RAW . Text to "SUITABLE"
                    set RAW . TextColor to 🟩
                    set DRINK . Text to "NOT SUITABLE"
                    set DRINK . TextColor to 🟥
            else    set DRINK . Text to "NOT SUITABLE"
                    set DRINK . TextColor to 🟥
                    set RAW . Text to "NOT SUITABLE"
                    set RAW . TextColor to 🟥
    else    set DRINK . Text to "NOT SUITABLE"
            set DRINK . TextColor to 🟥
            set RAW . Text to "NOT SUITABLE"
            set RAW . TextColor to 🟥
```

Figure 3.27 Example blocks for combine results function

The result page enabled user to view 5 previous results. The user needs to choose which sensor to show the result or show the combine result. The Figure3.28 shows the 5 recent results and Figure 3.29 shows the acquired sensor data from the ThingSpeak on the application.



TEMPERATURE SENSOR

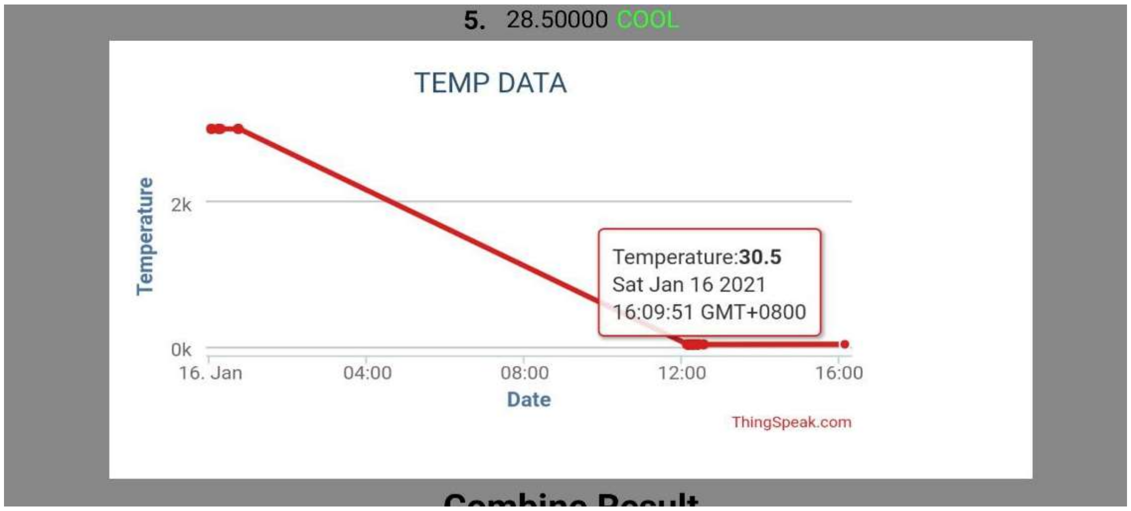
Show Result

MOST RECENT RESULT

1. 27.00000 COOL
2. 27.00000 COOL
3. 28.50000 COOL
4. 30.50000 COOL
5. 28.50000 COOL

TEMP DATA

Figure 3.28 Sensor data in table form



5. 28.50000 COOL

Temperature:**30.5**
Sat Jan 16 2021
16:09:51 GMT+0800

Figure 3.29 Sensor data in graphical form

3.8 Circuit Diagram

This part shows the circuit connection that completes the USV system. The circuit consists of two parts which are for mechanical parts and ESP32 Cam and, the water quality monitoring system.

3.8.1 Circuit diagram for mechanical parts and ESP32-Cam

We have use WEMOS MEGA+WIFI as the microcontroller to connect the mechanical parts and ESP32-Cam. Based on the diagram, the microcontroller power will be supplied a power bank. The Electronic Speed Controller (ESC) and SG90 Micro Servo are connected to the WEMOS MEGA+WIFI. Then, the ESP32-Cam is connected to the USB TTL serial. The USB TTL serial will then be connected to the power bank too.

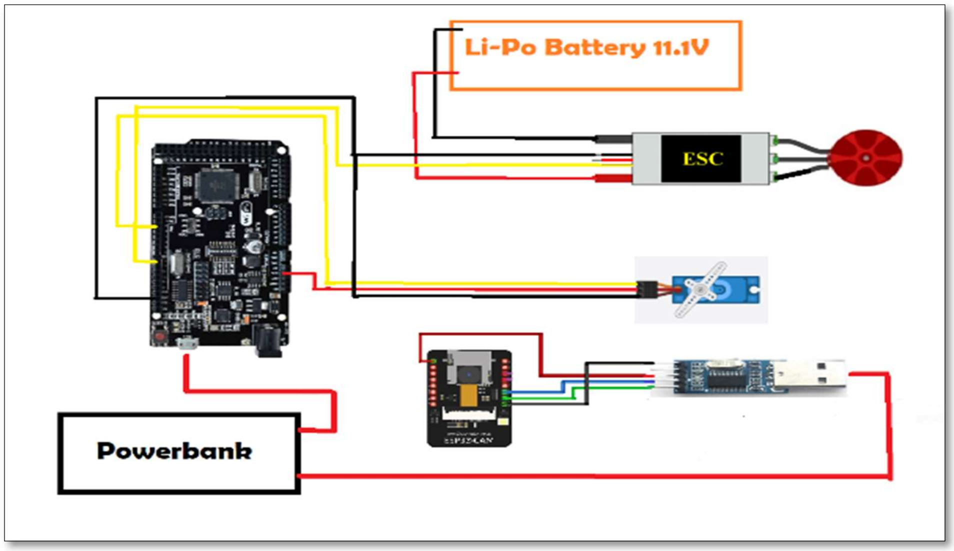


Figure 3.30 Circuit diagram for mechanical system and ESP32-Cam

3.8.2 Circuit diagram for water quality monitoring system

We have used three sensors related to water quality, which is analog pH sensor, analog turbidity sensor and temperature sensor. These sensors are connected to WEMOS MEGA+WIFI. The LCD I2C is functioning to display the IP address when the WEMOS MEGA+WIFI is power up. The 4.7K Ohm resistor is used as a pull up resistor. This resistor will ensure the voltage between Ground and Vcc is actively controlled when the WEMOS MEGA is on.

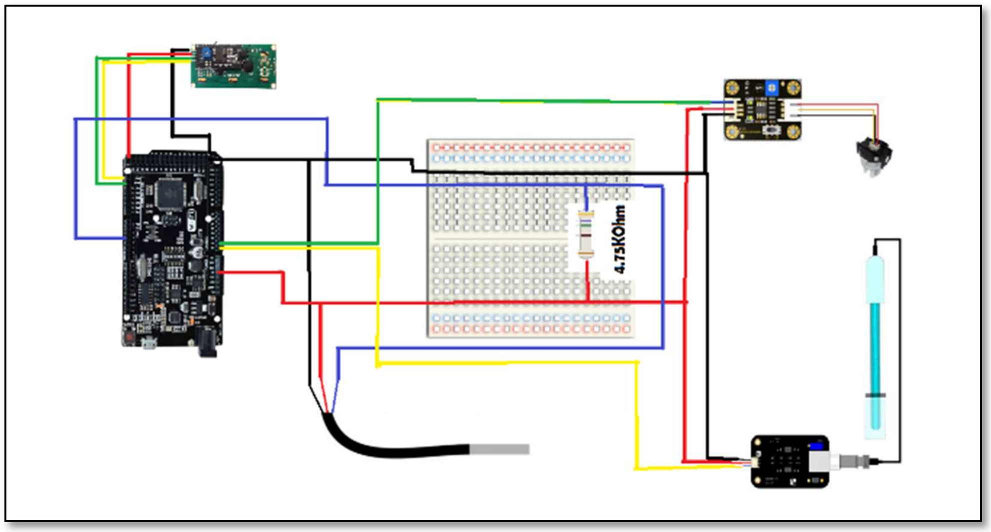
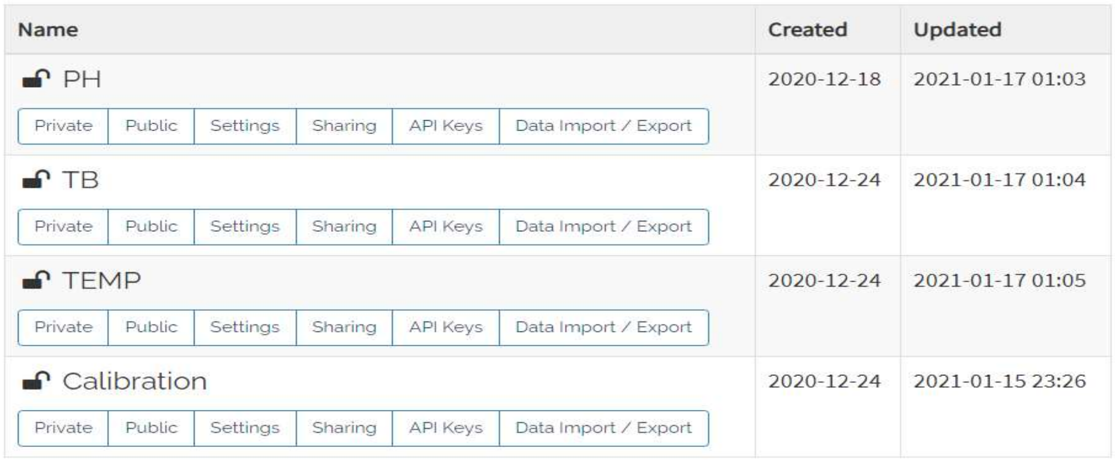


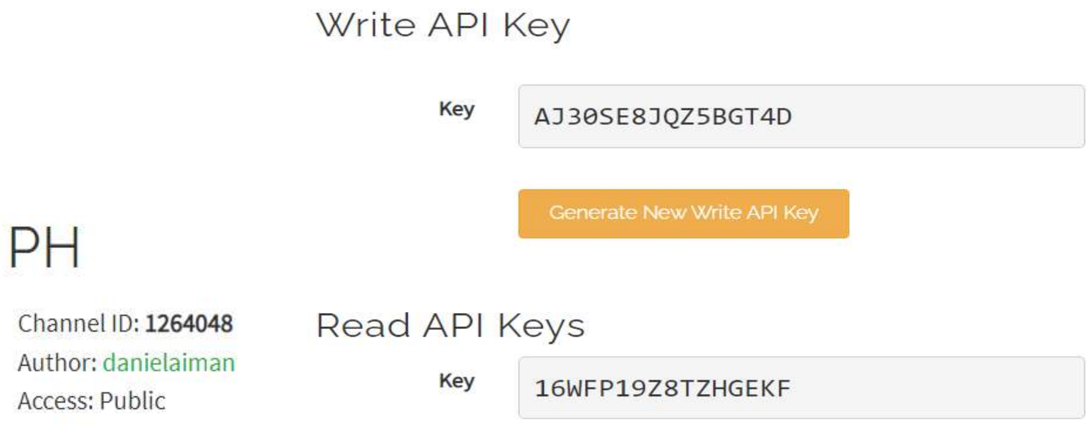
Figure 3.31 Circuit diagram for water quality system

3.9 ThingSpeak Configurations



Name	Created	Updated
PH	2020-12-18	2021-01-17 01:03
TB	2020-12-24	2021-01-17 01:04
TEMP	2020-12-24	2021-01-17 01:05
Calibration	2020-12-24	2021-01-15 23:26

Figure 3.32 ThingSpeak channel created



PH

Channel ID: **1264048**
Author: danielaiman
Access: Public

Write API Key

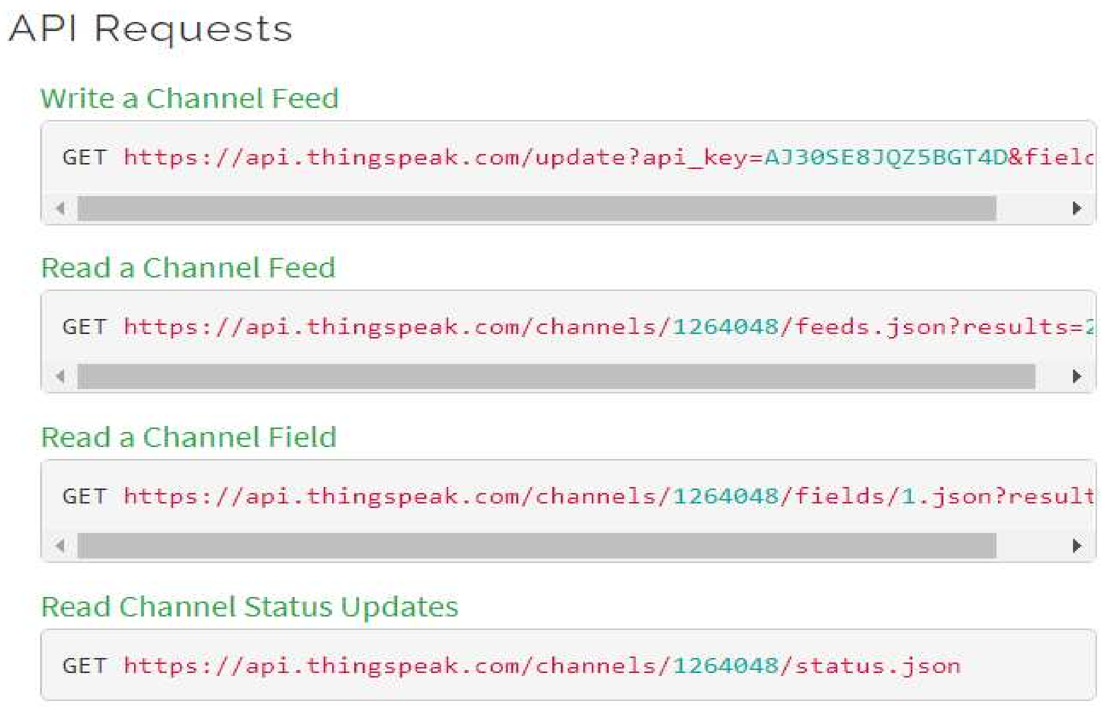
Key: AJ30SE8JQZ5BGT4D

Read API Keys

Key: 16WFP19Z8TZHGEKF

Figure 3.33 Channel details of pH sensor

Figure 3.32 and Figure 3.33 shows the created channel for each sensor. Each channel has its unique channel id and API key. This channel id and API keys is used to determine the place to store the acquired data. Write API key is used to write the acquired data to the channel while the read API keys is used to retrieved or read the acquired data from the ThingSpeak from the application.



API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=AJ30SE8JQZ5BGT4D&fielc
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/1264048/feeds.json?results=2
```

Read a Channel Field

```
GET https://api.thingspeak.com/channels/1264048/fields/1.json?result
```

Read Channel Status Updates

```
GET https://api.thingspeak.com/channels/1264048/status.json
```

Figure 3.34 ThingSpeak link to access the channel detail.

Figure 3.34 shows the link that is used to access the ThingSpeak channel. This link is entered in the blocks of the application on MIT App inventor. The only difference for each sensor is the channel id and API keys. The id and API keys can be changed based on the created channels.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

This chapter will explain about the results of the testing and the analysis of the testing.

4.2 Experimental Testing

4.2.1 USV Mechanical System

The experimental test was conducted to make sure that the USV and the developed application were working correctly. Based on the testing, the USV and the application works as desired. The USV is able to communicate with the application. The USV is able to get the command sent by the mobile application. For example, when we pressed the forward button on the application, the USV is able to processed it and move forward. As for the sensors, the sensors are able to work perfectly fine when the acquire sensor button is pressed. The USV can take the sensor readings of the water and send it to the application.

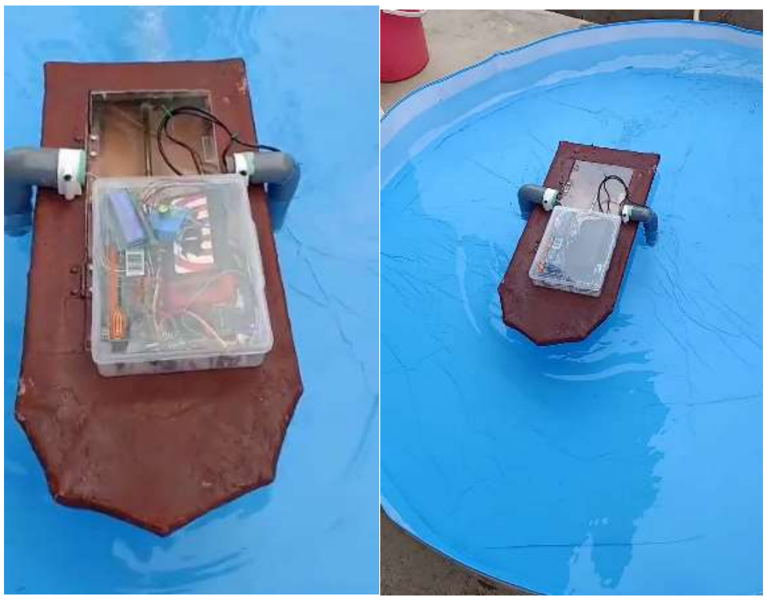


Figure 4.1 Experimental testing in a pool

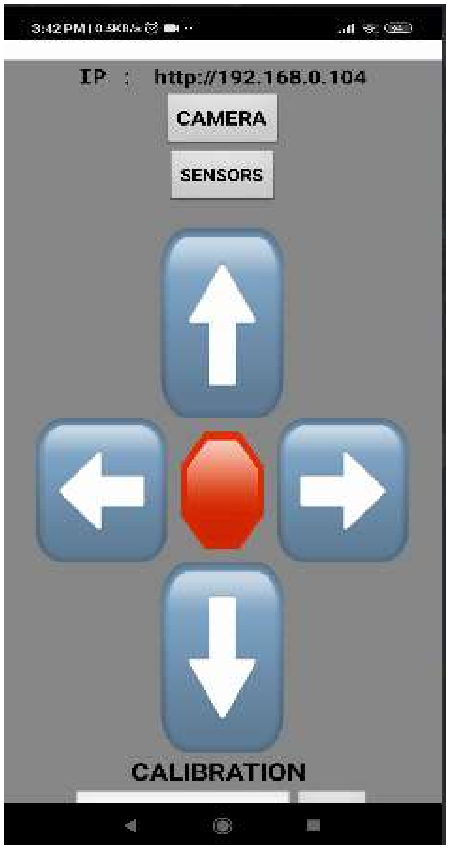


Figure 4.2 The application view when testing

4.2.2 pH Sensor Result

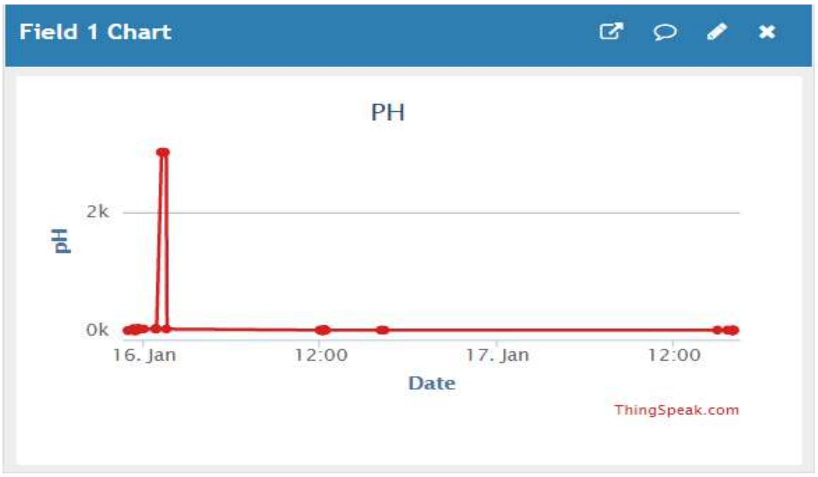
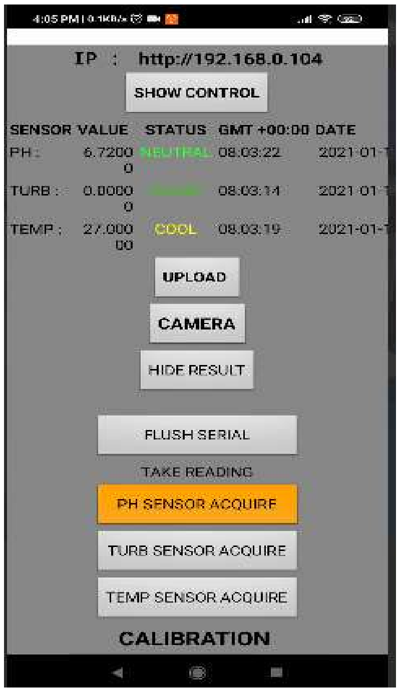


Figure 4.3 pH sensor result on ThingSpeak.



IP : http://192.168.0.104

SENSOR	VALUE	STATUS	GMT +00:00	DATE
PH :	6.72000	NEUTRAL	08:03:22	2021-01-
TURB :	0.00000	[illegible]	08:03:14	2021-01-
TEMP :	27.00000	COOL	08:03:19	2021-01-

Figure 4.4 pH sensor acquire

Figure 4.3 shows the result when the acquire pH sensor button is pressed. The acquire sensor button will take the sensor reading and send it to the ThingSpeak channel. The acquired sensor data can also be viewed on the application.

4.2.3 Turbidity Sensor Result

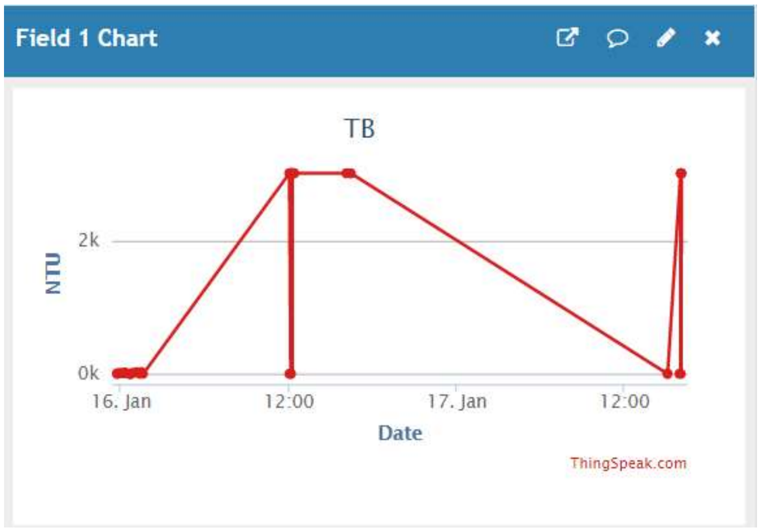
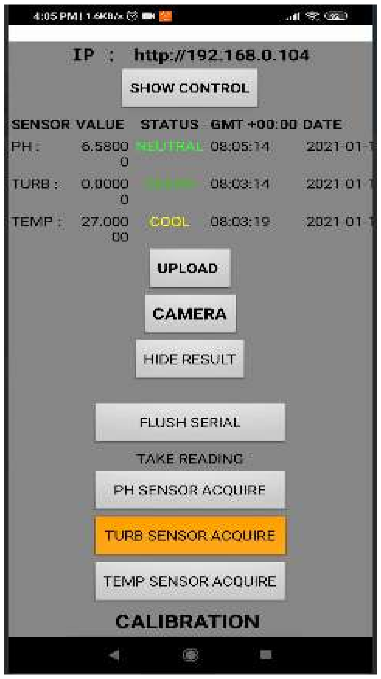


Figure 4.5 Turbidity sensor result on ThingSpeak



IP : http://192.168.0.104

SENSOR	VALUE	STATUS	GMT +00:00	DATE
PH :	6.58000	NEUTRAL	08:05:14	2021-01-1
TURB :	0.00000	[illegible]	08:03:14	2021-01-1
TEMP :	27.00000	COOL	08:03:19	2021-01-1

Figure 4.6 Turbidity sensor acquire

Figure 4.5 shows the result when the acquire turbidity sensor button is pressed. The acquire sensor button will take the sensor reading and send it to the ThingSpeak channel. The acquired sensor data can also be viewed on the application.

4.2.4 Temperature Sensor Result

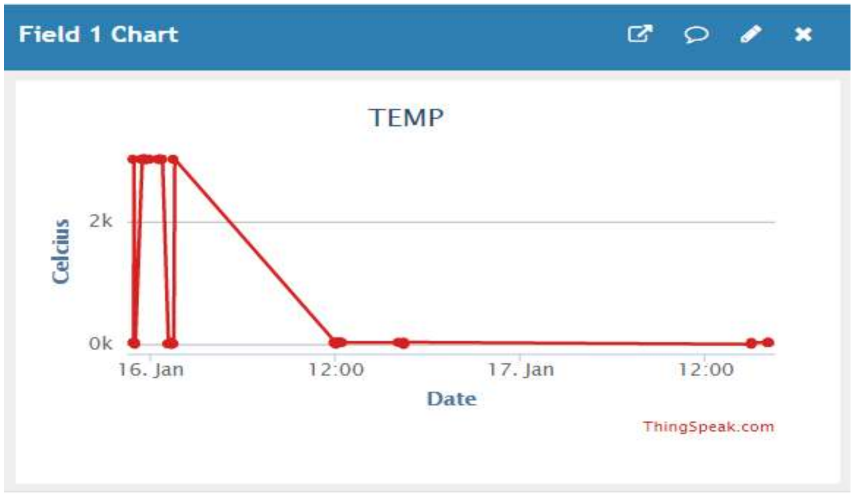
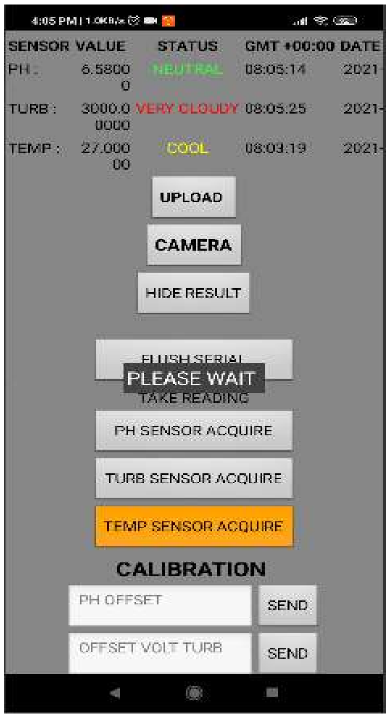


Figure 4.7 Temperature sensor result on ThingSpeak



```
SENSOR VALUE     STATUS       GMT +00:00 DATE
PH :    6.5800   NEUTRAL      08:05:14    2021-
        0
TURB :  3000.0   VERY CLOUDY  08:05:25    2021-
        0000
TEMP :  27.000   COOL         08:03:19    2021-
        00
```

Figure 4.8 Temperature sensor acquire

Figure 4.7 shows the result when the acquire temperature sensor button is pressed. The acquire sensor button will take the sensor reading and send it to the ThingSpeak channel. The acquired sensor data can also be viewed on the application.

4.3 Problem Encounter

In the testing phase, there are a few problems that have been faced. The problem are caused by the software and coding.

First, the ESP32-Cam function that give video feeds makes the application a bit lag. This happened because we used the feeds in a form of snapshot in .mjpeg format. The application was stuttered because the feeds is refreshed in every 1 second. This might also be happening because of different IP addresses are used. This is because the ESP32 Cam will print a different IP address while the WEMOS MEGA+WIFI uses a different IP address.

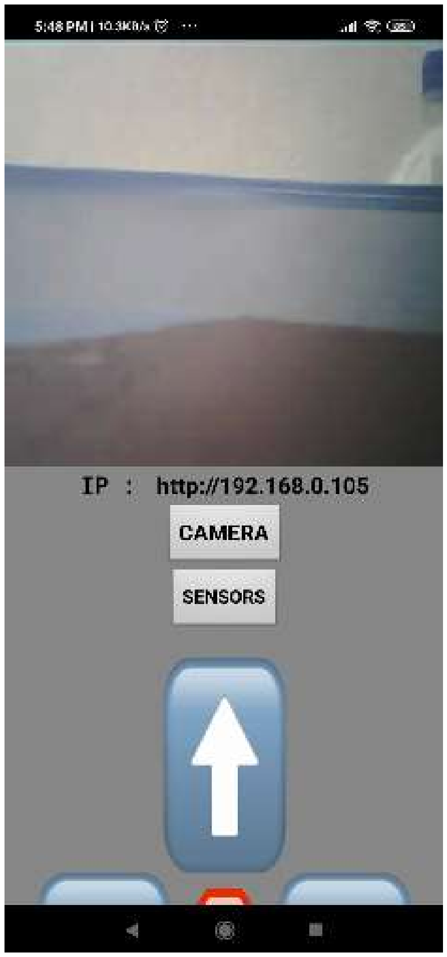
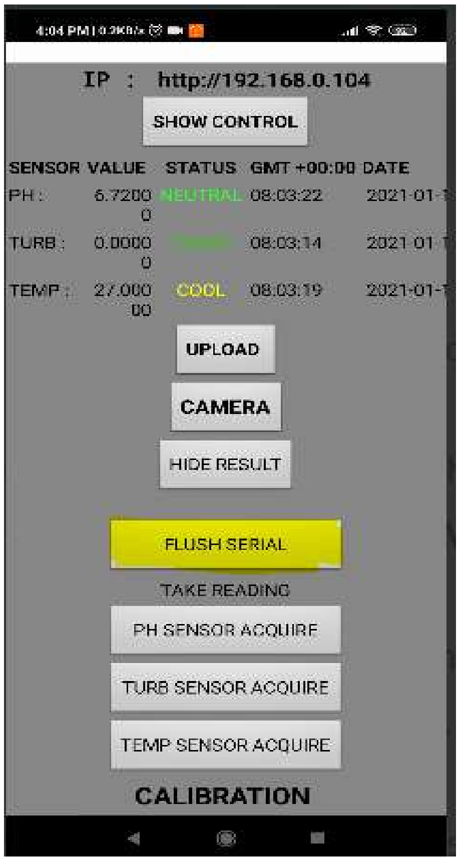


Figure 4.9 Video feed on application

Second, the problem encountered was when the UPLOAD button was pressed, it somehow uploaded the sensor data into the wrong channel and sometimes the data did not upload at all. This error is caused by various reasons. One of them might be the delay or the connection strength of the Wi-Fi. So, the data that should be uploaded earlier might be uploaded with the new data at the same time. In order to counter the problem, new channels of ThingSpeak are created using a second account. This channels only serve its purpose to received data after the UPLOAD button is pressed. The flush function is created to flush the delayed data in between acquiring the sensor data


IP : http://192.168.0.104

SENSOR	VALUE	STATUS	GMT +00:00	DATE
PH :	6.72000	NEUTRAL	08:03:22	2021-01-1
TURB :	0.00000	[illegible]	08:03:14	2021-01-1
TEMP :	27.00000	COOL	08:03:19	2021-01-1

Figure 4.10 Flush function on application

The flush function is created to flush the delayed data in between acquiring the sensor data. This will clear the delayed data from sending them to ThingSpeak.

CHAPTER 5

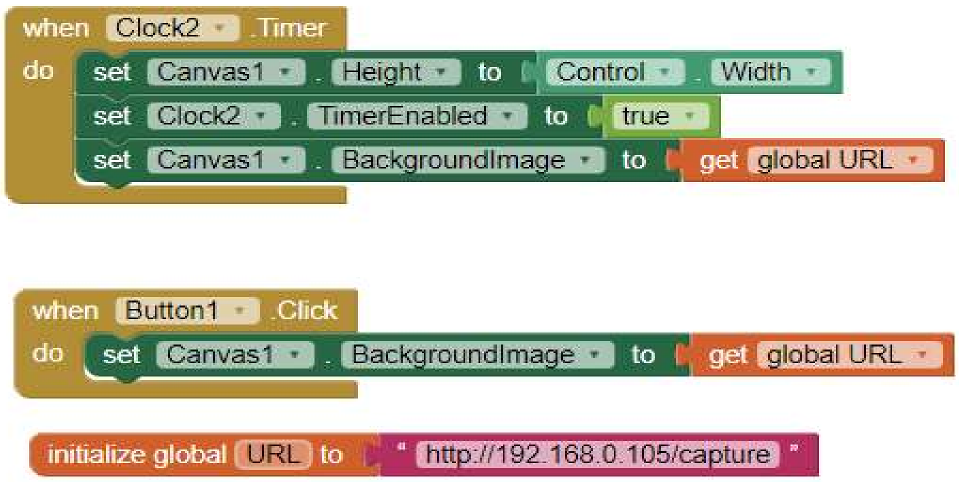
CONCLUSION

5.1 Introduction

The result of the project is the development of a USV system consisting of hardware, software and a mobile application. The USV hardware is being built in time and managed to obtain some real-world data from the nearby lake. System software works fine with minimal error or delay. The mobile application of the system also works perfectly without any noticeable glitch.

5.2 Future Recommendation

For future use, the USV can be improved in some aspects. One of them is the range of the connectivity. The connectivity of the current system is limited to the connectivity range of the Wi-Fi modem which is around 10 meters. To improve this, the antenna can be added to the system to extend the connectivity to at least 100 meters. This can ensure that the USV will not be losing internet connection while in field test.



```
when Clock2 .Timer
do  set Canvas1 . Height to Control . Width
    set Clock2 . TimerEnabled to true
    set Canvas1 . BackgroundImage to get global URL

when Button1 .Click
do  set Canvas1 . BackgroundImage to get global URL

initialize global URL to " http://192.168.0.105/capture "
```

Figure 5.1 ESP32 Cam block on MIT App Inventor

On the other hands, the function of the ESP32 Cam on the application can be improved. Since, the current system makes the application stuttering, the blocks of the application on the MIT App Inventor can be improvised. Based on the he current system of the ESP32 Cam on the application, the IP address is determined beforehand in the blocks as shown in the Figure 5.1. For example, the improvement can be made with asking the user to enter two IP addresses which are for ESP8266 and ESP32 Cam in the welcome page.

The other improvement is the USV can be mounted with the GPS module. The USV location can be traced by using the GPS. This can also avoid the loss of USV while losing connection with the Wi-Fi. This can also help the user to pinpoint the location of the acquired sensor data.

Finally, the problem we encountered is the IP addresses for the ESP32 Cam and ESP8266 sometimes printed as the same address. So, when this happened, the application can only show the video feed and cannot control the USV system. To counter this, the ESP32 Cam can be designated a fixed IP address beforehand.

REFERENCES

Cleaning up toxic river Sungai Kim Kim in Pasir Gudang to cost S$2.16 million. (2019, March 14). The Straits Times. https://www.straitstimes.com/asia/se-asia/cleaning-up-toxic-river-sungai-kim-kim-in-pasir-gudang-to-cost-s216-million

The Sun Daily. (2019, March 20). Victims of chemical pollution treated since March 8. Retrieved from https://www.thesundaily.my/local/5-848-victims-of-chemical-pollution-treated-since-march-8-CN708613

Water Supplies Department. (2019, April 1). Trial use of unmanned surface vessel (USV) system for water quality monitoring and sampling at Impou. (n.d.). https://www.wsd.gov.hk/en/core-businesses/water-quality/my-drinking-water-quality/unmanned-surface-vessel-system/index.html

M. Caccia, M. Bibuli, R. Bono, Ga. Bruzzone Gi. Bruzzone and E. Spirandelli Consiglio Nazionale delle Ricerche Istituto di Studi sui Sistemi Intelligenti per l'Automazione Via De Marini, 6 16149 Genova, Italy. (n.d.). Aluminum hull USV for coastal water and seafloor monitoring - IEEE conference publication. IEEE

Wonse Jo, Yuta Hoashi, Lizbeth Leonor Paredes Aguilar, Mauricio Postigo-Malaga, José M.Garcia-Bravo, Byung-Cheol Min. A low-cost and small USV platform for water quality monitoring. 2019 The Authors. Published by Elsevier Ltd.

Yufei Liu. Noboru Noguchi. Takeshi Yusa. (n.d.). Development of an unmanned surface vehicle for autonomous navigation in a Paddy Field

Carlos Almeida*, Tiago Franco**, Hugo Ferreira*, Alfredo Martins*, Ricardo Santos**, José Miguel Almeida*, João Carvalho*, Eduardo Silva*. (n.d.). (PDF) Radar based collision detection developments on usv roaz II.

PUTRI NUR FARHANAH MOHD SHAMSUDDIN1, MUHAMAD ARIFPIN MANSOR1, *, ROSHAHLIZA M. RAMLI1, RAJA MARIATUL QIBTIAH RAJA JAAPAR2. (2020, April 1). (PDF) Development of navigation system for unmanned surface vehicle by improving path tracking performance.

I. Y. Amran, K. Isa, H. A. Kadir, R. Ambar, N. Syila, A. Aziz, A. Kadir, M. Haniff and A. Mangshor, "Development of Autonomous Underwater Vehicle for Water Quality Measurement Application," pp. 1-16.

J. Liu, J. Sun, J. Wang, G. Huang, J. Jing, X. Xiang, X. Chen and Y. Zhang, "Comparison of effects on volatile and semi-volatile organic compounds in water by different sampling methods," 2009 International Conference on Energy and Environment Technology, ICEET 2009, vol. 2, no. November 2008, pp. 480-483, 2009.

Fernando K Tecnologia, Arduino Mega com WiFi Embutido ESP8266, 2017.

Sathish Pasika, Sai Teja Gandla. (2020, July). Smart water quality monitoring system with cost-effective using IoT. ScienceDirect.com | Science, health and medical journals, full text articles and books. https://www.sciencedirect.com/science/article/pii/S2405844020309403

Abbas Miry, Gregor Alexander Aramice. (2020, August). (PDF) Water monitoring and analytic based ThingSpeak. ResearchGate.

Miry, A. and Aramice, G., 2021. *Water monitoring and analytic based thingspeak*. [online] Academia.edu. Available at: <https://www.academia.edu/43785373/Water_monitoring_and_analytic_based_ThingSpeak> [Accessed 28 January 2021].

V., V., & Gaikwad, D. (2017). Water Quality Monitoring System Based on IOT. Research India Publications.

International Journal of Electrical and Computer Engineering (IJECE). (2019). Water monitoring and analytic based thingspeak.

Fernando K Tecnologia, *Arduino Mega com WiFi Embutido ESP8266,* 2017.

I. Y. Amran, K. Isa, H. A. Kadir, R. Ambar, N. Syila, A. Aziz, A. Kadir, M. Haniff and A. Mangshor, "Development of Autonomous Underwater Vehicle for Water Quality Measurement Application," pp. 1-16.

Niel Andre cloete, Reza Malekian and Lakshmi Nair, Design of Smart Sensors for Real-Time Water Quality monitoring, ©2016 IEEE conference.

APPENDIX A

ARDUINO CODE

ARDUINO MEGA 2560 CODE

```
#include <MemoryFree.h>
#include <EEPROM.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
String inString;
/*******PHSENSOR************/
#define SensorPin A1
float Offset=0 ;
#define samplingInterval 20
#define ArrayLenth 40
int pHArray[ArrayLenth];
int pHArrayIndex = 0;
static unsigned long samplingTime = millis();
static float pHValue, voltage;
/*******TEMPERATURESENSOR************/
#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 3
DeviceAddress thermometerAddress;
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature tempSensor(&oneWire);
/*******TURBIDITYSENSOR************/
int sensorPin = A0;
float volt;
float ntu;
float tboffset=0;
/*******SERVO LEFT AND RIGHT*******/
#include <Servo.h>
Servo servo;
/*********DC MOTOR*************/
int val;
Servo esc;

void setup() {

  Serial.begin(115200);
  Serial3.begin(115200);
  lcd.init();
  lcd.init();
  // turn on LCD backlight
```

```
lcd.backlight();
/*******TURBIDITYSENSOR************/
pinMode(sensorPin,INPUT);
/*******TEMPERATURESENSOR************/
Serial.println("DS18B20 Temperature IC Test");
Serial.println("Locating devices...");
tempSensor.begin(); // initialize the temp sensor
if (!tempSensor.getAddress(thermometerAddress, 0))
Serial.println("Unable to find Device.");
else {
Serial.print("Device 0 Address: ");
printAddress(thermometerAddress);
Serial.println();
}
tempSensor.setResolution(thermometerAddress, 9);// set the temperature resolution (9-12)

/*****SERVO LEFT AND RIGHT********/
  servo.attach(2);
  servo.write(0);
  delay(2000);
/*************************DC
MOTOR*****************************************/
  esc.attach(8); //Specify the esc signal pin,Here as D8
  //esc.writeMicroseconds(1000); //initialize the signal to 1000
}
void loop() {
  volt = 0;
for(int i=0; i<500; i++)
{
volt += ((float)analogRead(sensorPin)/1024)*5;
}
volt = (volt/500)+ tboffset;
if(volt < 2.5){
ntu = 3000;
}else if(volt >=4.2){
ntu=0;
}else{
ntu = (-1120.4*(volt*volt))+(5742.3*volt)-4352.9;
}
if (millis() - samplingTime > samplingInterval)
{
pHArray[pHArrayIndex++] = analogRead(SensorPin);
if (pHArrayIndex == ArrayLenth)pHArrayIndex = 0;
voltage = avergearray(pHArray, ArrayLenth) * 5.0 / 1024 + 0.05;
pHValue = 3.5 * voltage + Offset;
samplingTime = millis();
}}
void serialEvent3() {
```

```
while (Serial3.available()) {
char inChar = Serial3.read();
Serial.write(inChar);
inString += inChar;
lcd.setCursor(0,0);
lcd.print(inString);
/***************CONTROL***************/

if (inChar == ']') {
if (inString.indexOf("[GO]")>0) {
  val =45;
  val= map(val, 0, 1023,1000,2000); //mapping val to minimum and maximum(Change if
needed)
  esc.writeMicroseconds(val); //using val as the signal to esc
delay(200);
}
else if (inString.indexOf("[LEFT]")>0) {
servo.write(0);
delay(200);
}
else if (inString.indexOf("[RIGHT]")>0) {
servo.write(120);
delay(200);
}
else if (inString.indexOf("[STOP]")>0) {
 servo.write(0);
 val =LOW;
  val= map(val, 0, 1023,1000,2000); //mapping val to minimum and maximum(Change if
needed)
  esc.writeMicroseconds(val); //using val as the signal to esc;
  servo.write(90);
delay(200);
}
/***********************PH SENSOR*********************/
if (inString.indexOf("[PH]")>0) {
Serial.print("Input string: ");
Serial.print(inString);
Serial.print("\tAfter conversion to float:");
Serial.println(inString.toFloat());
Offset=inString.toFloat();
inString = "";
delay(500);
}
if (inString.indexOf("[TB]")>0) {
Serial.print("Input string: ");
Serial.print(inString);
Serial.print("\tAfter conversion to float:");
Serial.println(inString.toFloat());
tboffset=inString.toFloat();
inString = "";
delay(500);
}
/*******TURBIDITYSENSOR************/
else if (inString.indexOf("[SENSOR1]")>0) {
Serial.println("Voltage="+String(volt)+" V Turbidity="+String(ntu)+" NTU");
Serial3.println(ntu);//print to esp8266
delay(500);
}
/*******TEMPERATURESENSOR************/
else if (inString.indexOf("[SENSOR2]")>0) {
tempSensor.requestTemperatures(); // request temperature sample from sensor on the one
wire bus
displayTemp(tempSensor.getTempC(thermometerAddress)); // show temperature on display
delay(500);
}
/*******PHSENSOR************/
else if (inString.indexOf("[SENSOR3]")>0) {
Serial.print("pH: ");
Serial.println(pHValue, 2);
Serial3.println(pHValue,2);//print to esp8266
}
else if (inString.indexOf("[FLUSH]")>0) {
Serial3.println("0000");//print to esp8266
}
else{
Serial.println("Wrong command");
}
inString = "";
}}}

/*******TEMPERATURESENSOR************/
void displayTemp(float temperatureReading) { // temperature comes in as a float with 2
decimal places
// show temperature °C
Serial.println(temperatureReading); // serial debug output
Serial3.println(temperatureReading,2); //print to esp8266
Serial.print("°");
Serial.print("C ");
}

// print device address from the address array
void printAddress(DeviceAddress deviceAddress){
for (uint8_t i = 0; i < 8; i++){
if (deviceAddress[i] < 16) Serial.print("0");
Serial.print(deviceAddress[i], HEX);
}}
```

```
/*******TURBIDITYSENSOR************/
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;
}
/*******PH SENSOR************/
double avergearray(int* arr, int number)
{
int i;
int max, min;
double avg;
long amount = 0;
if (number <= 0)
{
Serial.println("Error number for the array to avraging!/n");
return 0;
}
if (number < 5) //less than 5, calculated directly statistics
{
for (i = 0; i < number; i++)
{
amount += arr[i];
}
avg = amount / number;
return avg;
}
else
{
if (arr[0] < arr[1])
{
min = arr[0]; max = arr[1];
}
else
{
min = arr[1]; max = arr[0];
}
for (i = 2; i < number; i++)
{
if (arr[i] < min)
{
amount += min; //arr<min
min = arr[i];
}
else
{
if (arr[i] > max)
{
amount += max; //arr>max
max = arr[i];
}
else
{
amount += arr[i]; //min<=arr<=max
}}}
avg = (double)amount / (number - 2);
}
return avg;
}
```

APPENDIX B

ARDUINO CODE

ESP 8266 CODE

```
#include <ESP8266WiFi.h>
#include "ThingSpeak.h"
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
MDNSResponder mdns;
/*******SERVO LEFT AND RIGHT*******/
#include <Servo.h>
Servo servo;
/*********DC MOTOR*************/
int val;
Servo esc;
float phcab, tbcab;
char data;
const char* ssid = "tuturu";
const char* password = "danielaiman98";
WiFiClient client;
//PH//
unsigned long myChannelNumber2 = 1264048;
const char * myWriteAPIKey1 = "AJ30SE8JQZ5BGT4D";
//TEMPERATURE//
unsigned long myChannelNumber3 = 1269605;
const char * myWriteAPIKey2 = "23UX6CU7GDY0ZER9";
//TURB//
unsigned long myChannelNumber1 = 1269609;
const char * myWriteAPIKey3 = "EXFJB7D9RWO1SDA3";
//calibration//
char* readAPIKeycab = "1ZNUY26TJCNJULBY";
float phval, turbval, tempval;
ESP8266WebServer server(80);
String webPage = "";
String inString;
void setup(void){
  ////Controller
webPage += "<h1>ESP8266 Web Server</h1>";
webPage += "<p>GO <a href=\"GO\"><button>GO</button>";
webPage += "<p>STOP <a href=\"STOP\"><button>STOP</button>";
webPage += "<p>RIGHT <a href=\"RIGHT\"><button>RIGHT</button>";
webPage += "<p>LEFT <a href=\"LEFT\"><button>LEFT</button>";
  ////Sensors
webPage += "<p>SENSOR1 <a href=\"SENSOR1\"><button>SENSOR1</button>";
webPage += "<p>SENSOR2 <a href=\"SENSOR2\"><button>SENSOR2</button>";
webPage += "<p>SENSOR3 <a href=\"SENSOR3\"><button>SENSOR3</button>";
webPage += "<p>SENDSENSOR1 <a
href=\"SENDSENSOR1\"><button>SENDSENSOR1</button>";
webPage += "<p>SENDSENSOR2 <a
href=\"SENDSENSOR2\"><button>SENDSENSOR2</button>";
webPage += "<p>SENDSENSOR3 <a
href=\"SENDSENSOR3\"><button>SENDSENSOR3</button>";
webPage += "<p>FLUSH <a href=\"FLUSH\"><button>FLUSH</button>";
webPage += "<p>PH <a href=\"PH\"><button>PH</button>";
webPage += "<p>TB <a href=\"TB\"><button>TB</button>";
delay(1000);
Serial.begin(115200);
WiFi.begin(ssid, password);
ThingSpeak.begin(client); // Initialize ThingSpeak
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print("IP address: ");
Serial.print(" ");
Serial.print(WiFi.localIP());
if (mdns.begin("esp8266", WiFi.localIP())) {
}
server.on("/", [](){
server.send(200, "text/html", webPage);
});
server.on("/GO", [](){
server.send(200, "text/html", webPage);
Serial.println("[GO]");
delay(300);
});
server.on("/REVERSE", [](){
server.send(200, "text/html", webPage);
Serial.println("[REVERSE]");
delay(300);
});
server.on("/RIGHT", [](){
server.send(200, "text/html", webPage);
Serial.println("[RIGHT]");
delay(300);
});
server.on("/LEFT", [](){
server.send(200, "text/html", webPage);
Serial.println("[LEFT]");
delay(300);
});
server.on("/STOP", [](){
server.send(200, "text/html", webPage);
Serial.println("[STOP]");
delay(300);
});
```

```
server.on("/SENSOR1", [](){
server.send(200, "text/html", webPage);
Serial.println("[SENSOR1]");
});
server.on("/SENSOR2", [](){
server.send(200, "text/html", webPage);
Serial.println("[SENSOR2]");
});
server.on("/SENSOR3", [](){
server.send(200, "text/html", webPage);
Serial.println("[SENSOR3]");
});
server.on("/FLUSH", [](){
server.send(200, "text/html", webPage);
Serial.println("[FLUSH]");
while (Serial.available()>0)
{
int inChar = Serial.read();
if (inChar != '\n') {
inString += (char)inChar;
}
else {
Serial.print("Input string: ");
Serial.print(inString);
Serial.print("\tAfter conversion to float:");
Serial.println(inString.toFloat());
inString = "";
delay(500);
}}
});
server.on("/PH", [](){
server.send(200, "text/html", webPage);
Serial.println("[PH]");
phcab = ThingSpeak.readFloatField( 1269611, 1, readAPIKeycab );
Serial.println(phcab, 2);
});
server.on("/TB", [](){
server.send(200, "text/html", webPage);
Serial.println("[TB]");
tbcab = ThingSpeak.readFloatField( 1269611, 2, readAPIKeycab );
Serial.println(tbcab, 2);
});

///Turbidity

server.on("/SENDSENSOR1", [](){
server.send(200, "text/html", webPage);
Serial.println("[SENDSENSOR1]");
```

```
while (Serial.available()>0)
{
int inChar = Serial.read();
if (inChar != '\n') {
inString += (char)inChar;
}

else {
Serial.print("Input string: ");
Serial.print(inString);
Serial.print("\tAfter conversion to float:");
Serial.println(inString.toFloat());
turbval=inString.toFloat();
ThingSpeak.writeField(myChannelNumber2, 1, turbval, myWriteAPIKey2);
inString = "";
delay(500);
}} });

//// Temperature

server.on("/SENDSENSOR2", [](){
server.send(200, "text/html", webPage);
Serial.println("[SENDSENSOR2]");
while (Serial.available()>0)
{
int inChar = Serial.read();
if (inChar != '\n') {
inString += (char)inChar;
}
else {
Serial.print("Input string: ");
Serial.print(inString);
Serial.print("\tAfter conversion to float:");
Serial.println(inString.toFloat());
tempval=inString.toFloat();
ThingSpeak.writeField(myChannelNumber3, 1, tempval, myWriteAPIKey3);
inString = "";
delay(500);
}}});
////pH
server.on("/SENDSENSOR3", [](){
server.send(200, "text/html", webPage);
Serial.println("[SENDSENSOR3]");
while (Serial.available()>0)
{
int inChar = Serial.read();
if (inChar != '\n') {
inString += (char)inChar;
}
else {
Serial.print("Input string: ");
Serial.print(inString);
Serial.print("\tAfter conversion to float:");
Serial.println(inString.toFloat());
phval=inString.toFloat();
ThingSpeak.writeField(myChannelNumber1, 1, phval, myWriteAPIKey1);
inString = "";
delay(500);
}}});
server.begin();
}}
void loop(void){
server.handleClient();
}
```

APPENDIX C

ARDUINO CODE

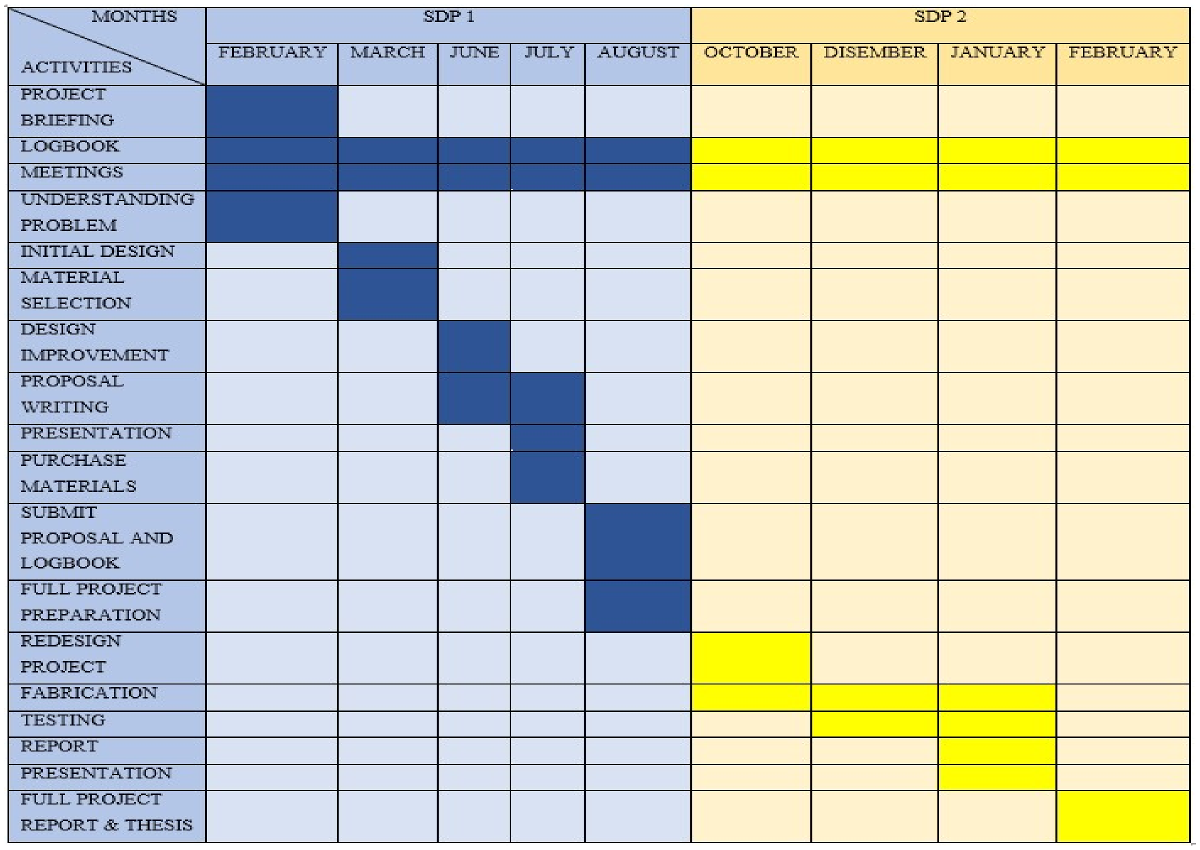
ESP32 CAM CODE

```
#include <esp_camera.h>
#include <WiFi.h>

//
// WARNING!!! Make sure that you have either selected ESP32 Wrover Module,
//           or another board which has PSRAM enabled
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h"  //disable brownout problems
#include "camera_pins.h"

const char* ssid = "tuturu";
const char* password = "danielaiman98";

void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  //initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);//flip it back
    s->set_brightness(s, 1);//up the blightness just a bit
    s->set_saturation(s, -2);//lower the saturation
  }
  //drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(10);
}
```

APPENDIX D

GANTT CHART

SDP1 &SDP2



MONTHS / ACTIVITIES	SDP 1					SDP 2			
	FEBRUARY	MARCH	JUNE	JULY	AUGUST	OCTOBER	DISEMBER	JANUARY	FEBRUARY
PROJECT BRIEFING	■								
LOGBOOK	■	■	■	■	■	■	■	■	■
MEETINGS	■	■	■	■	■	■	■	■	■
UNDERSTANDING PROBLEM	■								
INITIAL DESIGN		■							
MATERIAL SELECTION		■							
DESIGN IMPROVEMENT			■						
PROPOSAL WRITING			■	■					
PRESENTATION				■					
PURCHASE MATERIALS				■					
SUBMIT PROPOSAL AND LOGBOOK					■				
FULL PROJECT PREPARATION					■				
REDESIGN PROJECT						■			
FABRICATION						■	■	■	
TESTING							■	■	
REPORT								■	
PRESENTATION								■	
FULL PROJECT REPORT & THESIS									■

APPENDIX D

PROJECT COST

No	Materials	Unit/s	Price
1	Liquid pH sensor (Gravity:Analog pH Sensor/MeterKit For Arduino)	1	RM158.00
2	Turbidity sensor (Gravity:Analog Turbidity Sensor For Arduino)	1	RM59.90
3	WEMOS MEGA+WIFI	1	RM53.74
4	Rc Boat Shaft Kit 3.17mm Flexible Axle+Propeller for Rc Brushless Electric Boat	1	RM39.00
5	LCD1602 Serial IIC I2C 16x2 Liquid Crystal	1	RM10.50
6	D18B20 Temperature sensor	1	RM8.00
7	LiPo Rechargeable Battery 11.1V 2200mAh 30C	1	RM49.90
8	2-3 Cell LIPO Balance Charger	1	RM49.0
9	ESC 40A support 2-4S	1	RM37.60
10	Servo motor (SG90 Micro Servo)	3	RM14.70
11	ESP32 CAM WiFi + Bluetooth Development Board with OV2640 Camera Module	1	RM41.90
12	40 Ways Male to Male Jumper Wire	1	RM2.50
13	40 Ways Male to Female Jumper Wire	1	RM2.50
14	A5 Acrylic sheet 3mm X 210mm X 148mm	1	RM5.50
15	Fibreglass Resin 1kg + Hardener 25ml	1	RM19.50
16	Fiberglass	1	RM30.66
17	Rc Brushless Motor A2212/6T-2200KV- DXW	1	RM18.00
18	1.2mm Black Oxidised (Rust Proof) Push Rod-4pcs (26CM)		RM3.50
19	PVC Elbow Connector	2	RM5.00
20	PVC Pipe	1	RM5.00
21	Cable Clip	2	RM5.00
22	Plywood 3mm	1	RM15.00
		Grand Total	RM634.40