GENG5512 MPE Engineering Research Project Part 2

Autonomous Solar Boat

Yanke Cheng
23242404
School of Engineering, University of Western Australia

Supervisor: Prof. Thomas Bräunl
School of Engineering, University of Western Australia

Co-Supervisor: Pierre-Louis Constant
School of Engineering, University of Western Australia

Word count: 6462

School of Engineering
University of Western Australia

Submitted: 22 May 2023

Abstract

Since 2017, the Robotics Lab in University of Western Australia (UWA) has been aiming at developing a solar-powered Autonomous Surface Vehicle (ASV). By middle of 2022, the mechanical structure and electrical design of the solar boat, as well as supporting test tools, have been basically developed. Though the boat can navigate properly according to the way points added in Ground Control Station (GCS) software QGroundControl (QGC), there is a lack of nautical information displayed inside. Current requirement is for the vehicle to have more sensors to enable it carrying out missions and have more customized ground station user interface, as well as display of nautical information. This project focuses on further development of Autonomous Solar Boat ground control software, which includes implementations of video streaming function, widgets for wind and depth indicators, and customization of map system in QGC. Furthermore, this paper introduces the map system, resource management, Fact system and video streaming function of QGC, as well as the procedures to develop and customize QGC based on Qt QML. Future research of the boat includes estimation and management of power consumption, installation and performance estimation of solar panel, and exploration of boat's usage to carry out special tasks such as underwater topographic survey, water quality monitoring.

Acknowledgement

The author would like to thank following people for advice and support they provide in the project:
Prof. Thomas Bräunl for organising the entire solar boat project, as well as his support and guidance during the procedure of doing Autonomous Solar Boat project.
Co-Supervisor Pierre-Louis Constant for his advice on software development of project and instructions on operating solar boat.
The UWA River Lab for their sponsorship in purchasing devices required for the project.

Table of contents

List of Figures

List of Tables

Acronyms and Abbreviations

QGC	QGroundControl(a ground control software)
Qt	A user interface design platform
QML	Qt user interface development language
GCS	Ground Control Station
RC	Remote Controller
GUI	Graphical User Interface
UAV	Unmanned Autonomous Vehicle
MAV	Micro Arial Vehicle
COG	Course Over Ground
SOG	Speed Over Ground
TWD	True Wind Direction
TWS	True Wind Speed
AWD	Apparent Wind Direction
AWS	Apparent Wind Speed
WGS84	World Geodetic System 1984
CVLC	Command-line VLC

Nomenclature:

A'_θ	Apparent wind direction in mathematical coordinates (deg)
h_θ	Heading angle of boat (deg)
R_θ	Zero reference angle of boat(deg)
P_θ	Platform-relative wind angle(deg)
T_u	Component of true wind in North-South direction(m/s)
T_v	Component of true wind in East-West direction(m/s)
$\boldsymbol{A}$	Apparent wind vector(m/s)
$\boldsymbol{C}$	Ground speed of boat(m/s)
$\boldsymbol{T}$	True wind vector(m/s)
T_θ	True wind direction(deg)

1. Introduction

1.1 Background

Since 2017, the University of Western Australia is aiming for creating an autonomously guided surface vehicle which uses renewable energy as power and can thus travel a long distance. The idea of solar boat project comes from the SeaCharger project, which aims to design an Unmanned Surface Vehicle (USV) to cross an ocean using solar power only, thus, to demonstrate the feasibility of a small, solar-powered vessel crossing ocean with solar power only. In 2016, SeaCharger began the task to travel from California to New Zealand, and finally failed when there were still 300 miles to New Zealand [1].

By the mid-point of 2022, a great amount of job has been done to get the boat ready for sailing and testing, included as follow:

1.Mechanical structure, which was redesigned in 2020, from previous form to a twin hull form has been developed, as shown in Figure 1.

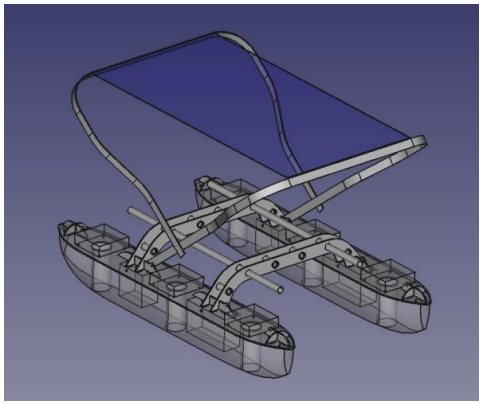

Figure 1: Twin Hull Design of Solar Boat [2]

2. Ground Control Station, including a power pack supporting power requirements from personal laptop, a Remote Controller (RC) used to control the boat manually, Wi-Fi modem and Bullet M5 Radio which together provide remote and stable wireless connection between boat and GCS, and a cart used for transportation of boat [3]. The control software is QGroundControl.

3. Pixhawk, used as micro controller to receive command from on-boat companion computer, collect and send data got from sensors to companion computer, and control the boat's behaviours. A GPS is installed and connected to Pixhawk to help with localization of boat.

4. Raspberry Pi 3B+, used as companion computer to read data inside Pixhawk, and send it to GCS laptop via the link set up. It is also used for creating video stream with Raspberry Pi camera on boat. Figure 2 below indicates the communication system of the boat.

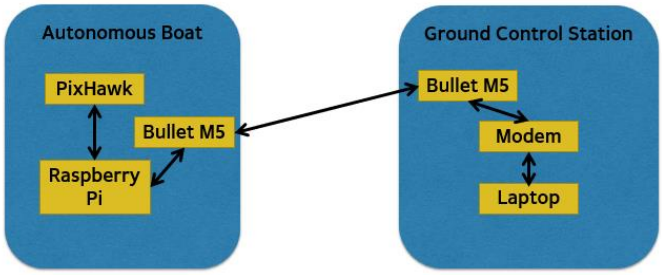

Figure 2: Communication System of Solar Boat

These features together make the testing of solar boat efficient and safe.

1.2 Current Situation

The sailing of boat mainly consists of two modes, Manual and Auto. While in Manual mode the boat simply controlled by a RC operated by a tester, Auto mode requires tester to first set up a task in QGC by adding way points or using mission planner, then sail the boat through way points given. The view inside QGC is displayed in Figure 3 as follow:

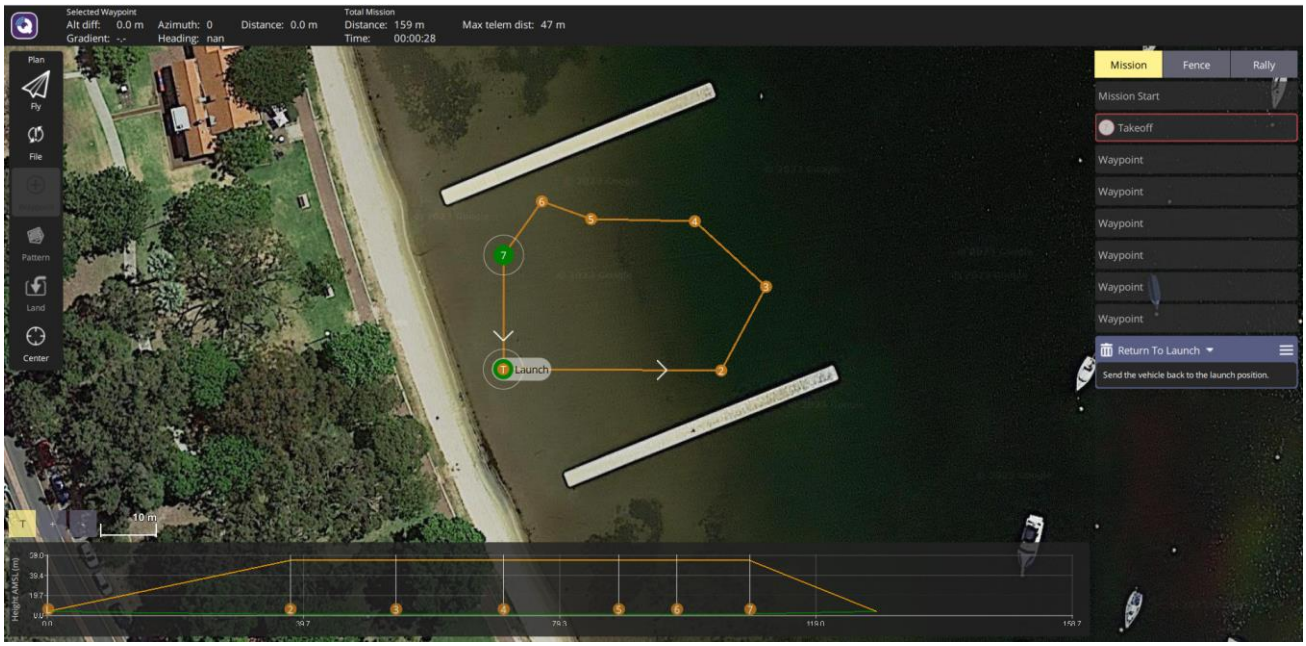

Figure 3: Mission Planning View in QGC

According to past tests, the boat can conduct missions set on QGC, and the path is relatively concise when the mission being conducted is not complex, as shown in Figure 4.

Figure 4: Path of Solar Boat in Simple Missions [3]

Though the boat is capable of sailing according to waypoints and tasks set up in QGC, and have a complete system to run a test, there are still aspects that can be improved and developed, listed as follow:

1. Improvement of software: Although QGC is a well-developed opensource software providing full flight control and mission planning functions for drones based on MAVLink, since the boat is designed for sailing on sea, there is a lack of nautical information inside to help with testers setting up missions, fixing boat's route and so on, as QGC is designed for drones. Moreover, with more sensors being installed on boat, though QGC has ability to display raw data collected from sensors, there are great demands to develop QGC from source with customized widgets and functions to transform raw data collected from sensors to concise and intuitive information displayed in Graphical User Interface (GUI). Another issue being faced by the boat is that the map system in QGC does not support nautical charts, while there are limited resources for nautical charts with opensource usage, which makes it a need to overlay information on nautical charts to current map.
2. Improvement of hardware: The current sensors installed on boat are basically useful for running of the boat, including GPS, Battery Monitor, and built in sensors on Pixhawk which detects the boat's headings, roll and pitch. However, for the voyage of solar boat, more important things to care about is the wind, and ability to detect whether there are obstacles underneath, which highlights the requirement to install wind and depth sensor.
3. Other aspects: Despite the installation of camera on boat, creating and streaming stable video from Raspberry Pi where the camera connected to ground station is also needed. As more and more sensors and facilities being installed on boat, many of them requires QGC to have the ability to display information they create, while QGC is a huge project with limited instructions on methods to customize it, thus make it a need to have easy methods to customize QGC.

1.3 Project Objectives and Scope

The development work of solar boat from 2022 to 2023 is divided into software part and hardware part respectively. Software part includes:

1. Development of indicators for depth and wind sensors, as well as implementing their function of transforming raw data collected to visualized information.
2. Video streaming from Raspberry Pi on boat to QGC software in Ground Control Station.
3. Customizing map in QGC, to make it able to display satellite map together with sea marks.

Hardware part includes:

1. Installation and testing of depth and wind sensors, including corresponding mechanical design for protection and solution of power consumption.
2. Integration of two additional thrusters.
3. Testing and adjustment of boat's control parameters.

This project focuses on the software part of development on solar boat.

1.4 Additional Literature Review

1.4.1 Ground Control System

Ground Control System are defined as architectures consist of requisite devices used to monitor, set commands, control Unmanned Autonomous Vehicles (UAV), and a GUI enabling pilots to operate remotely [4]. In mode details, the Paparazzi project from ENAZ states that basic components in GCS include a laptop connected to modem to keep data link with remote UAVs, and a RC transmitter for emergency [5].

1.4.2 QGroundControl

QGroundControl is a software that provides full flight control and setup of vehicle for vehicles based on PX4, Ardupilot or any other autopilot that communicates with MAVLink protocol [6] [7]. It has function of planning flight, vehicle set up, map display with the position, track of vehicle, video streaming, and supports for multiple platforms including Linux, Windows, OS X, iOS, and Android. QGC is an open-source software developed with Qt 5, secondary development can be done with Qt 5 to satisfy different needs. With advantages above, QGC is widely used as ground station software of UAVs, with various secondary development to satisfy different needs. In 2022, Luis Tirado et al apply QGC for controlling flight of UAVs using Px4 as flight controller, to reduce the training time for a worker operating drone to finish inspections of equipment and developed an interface for rapid parameterization of flights [8]. In 2018 and 2019, Cristian et al and Tristan et al extended QGroundControl to have automated mission planning function and autonomous multi agents function

respectively [9] [10].

1.4.3 MAVLink (Micro Air Vehicle Link) Protocol

MAVLink protocol is a higher-level open-source communication protocol based on serial communication, which is mainly used in the communication of Micro Aerial Vehicle (MAV) [11]. MAVlink is a set of rules for sending and receiving data commonly used by a small aircraft to communicate with a ground station (or other aircraft), and a checksum function is added. The wide application of the MAVLink protocol enables different vendors and developers to integrate and interoperate UAV systems. It provides a standard means of communication between UAVs and autopilot systems, enabling reliable and efficient data exchange and control between different components. Despite its features of light weight, expandability, and various message types defined inside [12], in [13], the author used Caesar cipher cryptography to encrypt MAVLink messages and send to MAV, and proved that during the establishment phase, an intruder could detect and break security system with no effort, which makes it an issue of the security of MAVLink. To solve this, in [14] and [15], different methods of encryption are performed based on MAVLink, to improve security of MAVLink and at the same time avoid the increment of memory and battery usage.

1.4.4 Applications of Autonomous Surface Vehicles

With the features of autonomy, flexibility and cost effectiveness, and capability of performing a variety of missions in complex and dangerous marine environments, Autonomous Surface Vehicles have been widely applied in area of aquatic plant species identification, water quality monitoring, oceanographic surveys and so on [16] [17] [18]. In [17] a system involving collection of raw data of aquatic plant species and recognition, classification of objects using deep learning methods was implemented, while there is a lack of real time response of the system, which can be implemented by installing an onboard computer processing system. The solar boat in contrast, has installed raspberry pi as companion computer, but only used for communication and video creation. Besides conducting marine scientific research, ASVs with camera installed can also be used in areas of marine monitoring and safety, detecting illegal fishing, marine pollution, or other potential safety threats. Being equipped with lifesaving, communication, and medical equipment, ASVs also play an important role in maritime rescue.

2. Method

The main development of project is based on QGC, with developing tool Qt 5. Due to the situation that QGC is updated constantly, while there is a lack of explanation of code structure in detail, and resources online is usually out of date, finding places to add items inside is time consuming. This part mainly explains the QML files and resources management inside QGC, and main things changed to achieve target of project. Discussion of video streaming system, features of indicators and consideration of customizing map is also covered.

2.1 Pre-requisite

The pre-requisite for software development of solar boat includes setting up of Qt 5 development environment, compiling and building of QGC source code and installation of GStreamer library, which is used by QGC to construct video streaming function. Guides and useful pages have been attached in Appendix A.

2.2 QGC Structure

The development of QGC follows a style of hybrid programming with C++ and QML in Qt. With extensive functionality and powerful libraries provided by Qt C++, developers can process complex logic and algorithms with high performance directly in C++, while usage of QML focuses on defining user interfaces and interactions, providing flexible layouts and visual effects capabilities.

2.2.1 QGC Application Structure (Top Level View)

Figure 5 below indicates the structure of QGC Application, with corresponding visual effects display function in QGC GUI.

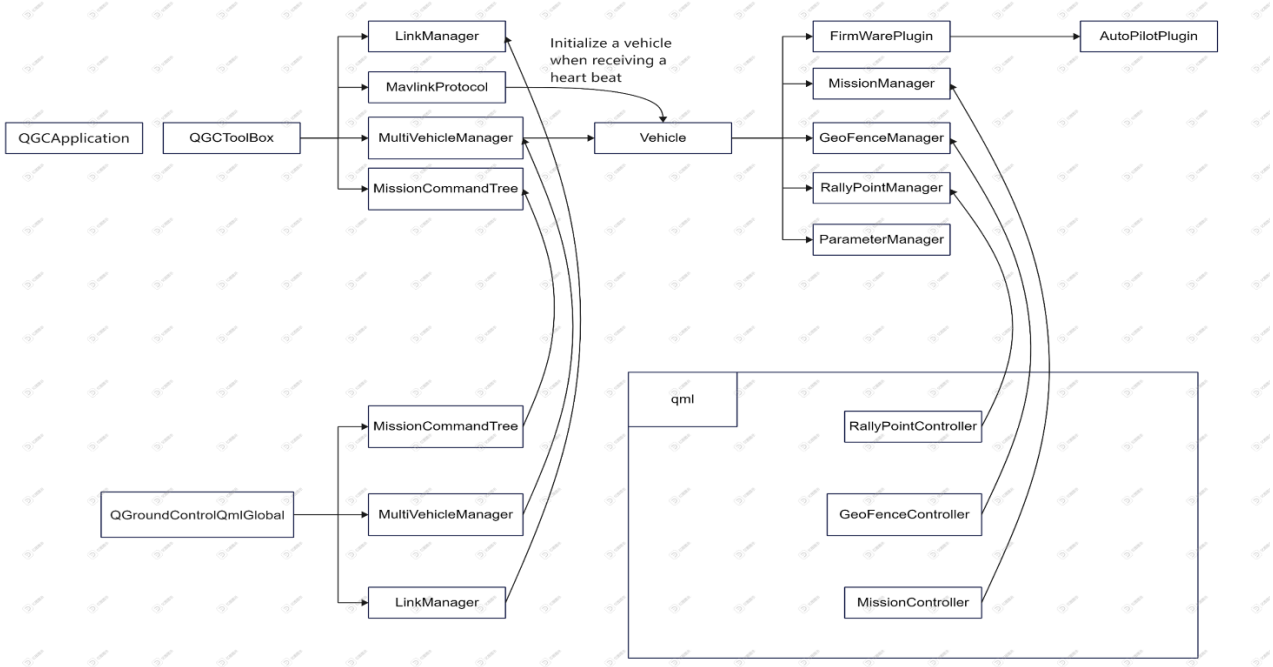

Figure 5: Structure of QGC Application

In QGC, the QGCApplication class is a central component that manages overall application lifecycle and provides various services and functionalities including initialization of application, management of user interface, communication between vehicle, management of plugins and so on. The QGCToolBox class is responsible for managing and providing access to various tools and panels in the application. It provides the infrastructure for integrating and interacting with various functionalities and displays within the application's user interface.

When the application is opened, the LinkManager class keeps a UDP connection to wait for a vehicle to connect [19]. Once a new known device is connected, it will create a new serial link connected to the vehicle, and the message bytes are sent through the link to MavlinkProtocol class, where message bytes are translated into MAVLink Message, if the received message is a heartbeat message, it will call MultiVehicleManager class to create an instance of new vehicle according to the message in the heartbeat. While the LinkManager class is used to receive message bytes, MavlinkProtocol class is used to translate the bytes into mavlink message, and call vehicle to handle the message through message processing functions in the MavlinkProtocol class.

On the user interface side, the QGroundControlQmlGlobal class is a global singleton object that provides access to all kinds of functionalities and resources within QML environment of the application. It serves as a bridge between the QML interface and the underlying C++ code of QGroundControl. Regarding the mission planning and execution of vehicle after connection, the

MissionController class, GeoFenceController class and RallyPointController class work in conjunction with the respective manager classes to handle the interaction and control of missions, geofences and rally points for unmanned vehicles.

2.2.2 Resources Management

In QGC, resources files such as images, fonts, translations and QML files are managed with the help of Qt resource system, which allows them to be embedded into the application, and to be easily accessible and portable. Definition and Organization of resources are completed by using .qrc files, which is an XML-based file that lists the resources along with their file paths within the project directory structure. Table 1 below shows the list of .qrc files and related categories of resources.

Table 1 List of .qrc files in QGC

Item	Usage
APMResources.qrc PX4Resources.qrc	Used to manage and organize the resources (such as images, QML files, fonts, etc.) that are specific to the APM autopilot (PX4 autopilot) and its associated features within the QGroundControl application.
InstrumentValueIcons.qrc	Provides a collection of icons representing various data types or values that are commonly used in QGroundControl's instrument panels.
UnitTest.qrc	Primarily used for organizing and including test-related files and assets used during the testing process.
airmap.qrc	Used for integrating the AirMap service into the QGroundControl application.
qgcimages.qrc	Contains a collection of resource paths and mappings that define the images to be included and accessed by QGroundControl.
qgcresources.qrc	Used to manage and organize various resources that are specific to project or customization.
qgroundcontrol.qrc	The main resource file that is responsible for managing and organizing various resources used within the application.

2.3 QGC Fact System

In QGC, vehicle data is managed in a flexible and powerful way using Fact System, which is designed to handle the communication and organization of data between QGC and UAV. Each Fact in the QGC Fact System represent individual parameters associated with the UAV, including telemetry values, settings, status indicators and so on. Fact Metadata, along with each Fact, describes the properties and behaviour of Fact related to it, and Fact Group manages a set of Facts. With the help of Fact System, each Fact can be directly linked to UI elements, be automatically updated in UI

elements when the associated values change.

2.4 Depth Indicator

Installation of depth sensor is referencing [20], which enables QGC to access data gotten from Pixhawk directly during design of depth indicator with help of QGC Fact System. The Fact groups and Fact being used in depth indicator are DistanceSensors and groundspeed. While implementation of depth indicator does not need complex logic calculation or algorithm, it can be simply done with QML only.

2.4.1 Objective Features of Depth Indicator

The objective features of a depth indicator are listed as follows:

1. Measurement Display: The depth indicator should be able to display depth measurements in a clear and readable format. This can be done either numerically or visually through a graphical representation.
2. Real-time Depth Monitoring: The depth indicator needs to continuously update displayed value as depth changes.
3. Maximum depth tracking: The indicator may also need to include a function of depth tracking, which enables system to record and display the deepest point reached during a sail and allow users to monitor for safety or reference purposes.

2.4.2 Depth Indicator Implementation

Implementation of depth indicator is mainly through usage of Qt ChartView class which provides a way to display various types of charts including line charts, bar charts and scatter charts. Display of measurements is implemented by setting the x-axis of chart to be scaled with boat's speed over ground, and y-axis (depth) to be scaled with maximum depth collected in history. Two strings are used to display current depth and maximum depth in history as well. Figure 6 below illustrates the final design of depth indicator.

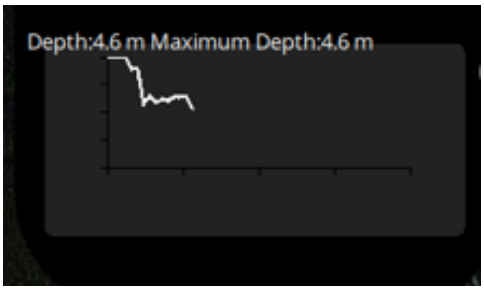

Figure 6 Design of Depth Indicator

List of codes to fix in QGroundControl is shown below in Table 2.

Table 2 Codes to fix for Depth Indicator

Item	Changes
Modify qgroundcontrol.qrc	Add to /qml: QGCDepthSensor.qml

Modify src/QmlControls/FlightMap/qmldir	Add modules: QGCDepthSensor.qml
Create src/FlightMap/Widets/QGCDepthSensor.qml	Add implementation of Depth Indicator
Modify src/FlightMap/Widgets/QGCInstrumentWidget	Add modules: QGCDepthSensor.qml

2.5 Wind Indicator

2.5.1 Design considerations

While some of wind sensors in the market have built in GPS, thus have the function to calculate the true wind according to vehicle's heading and speed, since the boat already has GPS installed, there is no need to have a wind sensor with built in GPS and Inertial Measurement Unit (IMU) under consideration of power consumption, weight, and cost. Moreover, taking advantages of QGC's feature that sensor data collected is convenient to access, implementing in this way enables the ability to calculate more accurate true wind speed and directions with the usage of data collected from other sensors.

The wind sensor being picked for solar boat outputs apparent wind and direction only. Final solution is to apply data collected from wind sensor, along with data from on boat GPS in QGC and calculate them before displaying.

2.5.2 Objective Features of Wind Indicator

The aim of wind indicator is to provide accurate and easily interpretable information about wind direction and wind speed. The objective features are listed below:

1. Display of wind speed and direction: Wind indicators should have arrows and compass points, with scales indicating angle. Those arrows should also have the function to rotate according to wind angles to reflect change of directions.
2. Calculation of true wind: The user interface should also calculate true wind and direction according to speed and heading information from IMU and movement information provided by GPS before visually display them.
3. Numerical Display: Calculation result for wind speed and directions need to be displayed numerically as well.

2.5.3 True Wind Calculation

Calculating the true wind is essential in many applications, particularly in the fields of aviation, marine navigation, and meteorology. Information of true wind will help in following aspects:

1. Navigation and Route Planning: Determining the actual wind direction and speed allow pilots, sailors to make informed decisions about course adjustments, sail, and optimal paths to reach their destinations efficiently and safely.
2. Performance Optimization: Since the sail and performance of boat highly relies on wind, by accurately calculating the true wind, operators can adjust their strategies and control inputs to maximize speed, efficiency, and overall performance.

In mobile applications of wind sensor, the measure station is moving when measuring the wind, which results in the situation that wind being measured is a vector sum of station's movement and true wind, therefore true wind can be obtained by subtracting station's motion from apparent wind, as indicated in Figure 7.

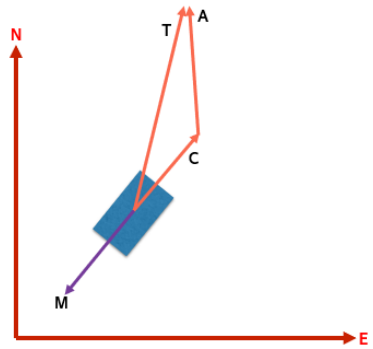

Figure 7 Vector Diagram of Apparent Wind Measurement

where **T** denotes vector of true wind, **A** denotes vector of apparentwind, **M** is the wind caused by boat's motion and **C** is the vector motion of boat over fixed earth.

In practical, the movement of boat usually is not where the boat is heading to due to effects of wind and water currents, means that the wind measured is indeed a platform-relative wind with same amplitude as apparent wind, and angle relative to zero reference angle which is defined as the angle between zero line of wind sensor and the bow of boat. Moreover, meteorological definition of wind is applied, which defines the wind direction as the direction where wind blows from, and thus make it a need to transform to mathematical coordinate system when calculating. Table 3 below indicates the list of parameters, their corresponding definitions and Fact/FactGroup inside QGC.

Table 3 List of parameters used in True Wind Calculation [21]

Parameter	Definition	Fact in QGC

Course Over Ground (COG)	The direction (relative to true north) the vessel actually moves over the fixed Earth;	vehicle.gps.courseOverGround
Speed Over Ground (SOG)	The speed at which the vessel moves in the direction of the COG;	vehicle.groundSpeed
Heading(H)	The direction to which the bow is pointing relative to true north;	vehicle.heading
Apparent Wind Direction (AWD) Apparent Wind Speed (AWS)	The wind vector with a speed referenced to the vessel and a direction referenced to true north;	vehicle.wind.direction vehicle.wind.speed
True Wind Direction (TWD) True Wind Speed (TWS)	Defined as a vector wind with a speed referenced to the fixed Earth and a direction referenced to true north;	-
Platform-Relative Wind	Defined as the wind vector measured relative to the ship;	-

Final Calculation of true wind mainly refers to [21], apparent wind angle in mathematical coordinate system is calculated as

$$A'_\theta = 270^\circ - (h_\theta + R_\theta + P_\theta) \tag{1}$$

where h_θ is the boat's heading, R_θ is the zero reference, P_θ is the wind direction measured and A'_θ is the apparent wind direction in mathematical coordinate system. Then the heading is transformed as follow:

$$C'_\theta = 90^\circ - C_\theta \tag{2}$$

where C_θ is course over ground, C'_θ is course over ground in mathematical coordinates. After this, the east-west component and north-south component T_u and T_v are calculated as

$$\begin{bmatrix} T_u \\ T_v \end{bmatrix} = \begin{bmatrix} \cos(A'_\theta) & \cos(C'_\theta) \\ \sin(A'_\theta) & \sin(C'_\theta) \end{bmatrix} \begin{bmatrix} |\boldsymbol{A}| \\ |\boldsymbol{C}| \end{bmatrix} \tag{3}$$

Finally, amplitude and direction of true wind is obtained using:

$$|\boldsymbol{T}| = (T_u^2 + T_v^2)^{1/2} \tag{4}$$

$$T_\theta = 270^\circ - \operatorname{atan}\left(\frac{T_v}{T_u}\right) \tag{5}$$

2.5.4 Implementation of Wind Indicator

The final design of wind indicator is shown in Figure 8.

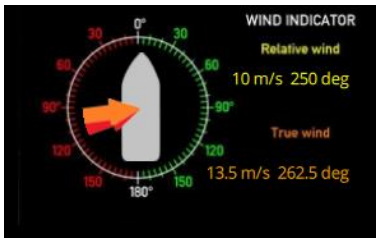

Figure 8 Design of Wind Indicator

As shown above, it has two rotational angles indicating the direction of true wind (the orange one) and relative wind (the red one). The scale on compass ring uses meteorological definition, where true north is defined as 0 angle, and clockwise is taken as positive direction. The numerical results for relative and true wind are displayed on the right. Parts to fix in QGC source code is shown in Table 4.

Table 4 Code to fix for Wind Indicator

Item	Changes
Modify qgroundcontrol.qrc	Add to /qml: QGCWindSensor.qml
Modify src/QmlControls/FlightMap/qmldir	Add modules: QGCWindSensor.qml
Create src/FlightMap/Widets/QGCWindSensor.qml	Add implementation of Wind Indicator
Modify src/FlightMap/Widgets/QGCInstrumentWidget	Add modules: QGCWindSensor.qml Change the combination of widgets for Wind Indicator, Compass, and Depth Sensor.

2.6 Map customization

The process of map displaying in QGC involves the following steps:

1. Selection of Map Provider: In the application, users are allowed to select a map provider from available options, including Mapbox, Google Maps and so on. The registration of providers table is done by QGCMapUrlEngine class.
2. Initialization of Map: After selection of map provider, QGC will initialize the map provider component, involving configuration of necessary settings such as API keys or authentication credentials, to establish a connection with selected map service. This is done by QGCMapEngine class.
3. Fetching of Map data: Map Provider component in QGC communicates with the selected map

service to fetch map data, including map tiles, satellite imagery, terrain information and other relevant data based on user's zoom level and location. The sources of tiles, including transformation of coordinates, and supporting APIs to retrieve the required map data are defined in various MapProvider classes, for example MapboxMapProvider, GoogleMapsMapProvider and GenericMapProvider.

4. Display of Map: After the map data is obtained, QGC displays the map within its user interface. This includes tasks like applying zoom levels, adjusting map's appearance, and rendering overlays. In QGC, display of map is included inside the Flightmap.qml.

Therefore, processes to customize the map in QGC are:

1. Find the suitable source of map type required and get the tile url of the tile server from official site of the type of map needed.
2. Create a MapProvider class or add class directly inside GenericMapProvider class, register and implement _geturl function and other supporting functions (ie. Coordinate transformation).
3. Add the _providerTable for customized map to register it and display its options inside QGC settings view.
4. If required to change the behaviour of QGC's display of maps, modify the Flightmap.qml file.

2.*6.1 OpenSeaMap*

Among choices of tile urls for nautical chart, the situation met is that many of the nautical chart providers do not support usage of their tiles for open-source usage, and the chart they can provide is only accessible from devices they provide, while both online and offline tile sets are not available.

OpenSeaMap is a collaborative mapping project focused on creating a free and open nautical chart of the world's seas and waterways. It provides detailed and up-to-date navigational information inside. The creation of its nautical chart is through utilization of crowd-sourced data and incorporation of various open data sources.

Figure 9 below indicates a sample nautical chart from OpenSeaMap.

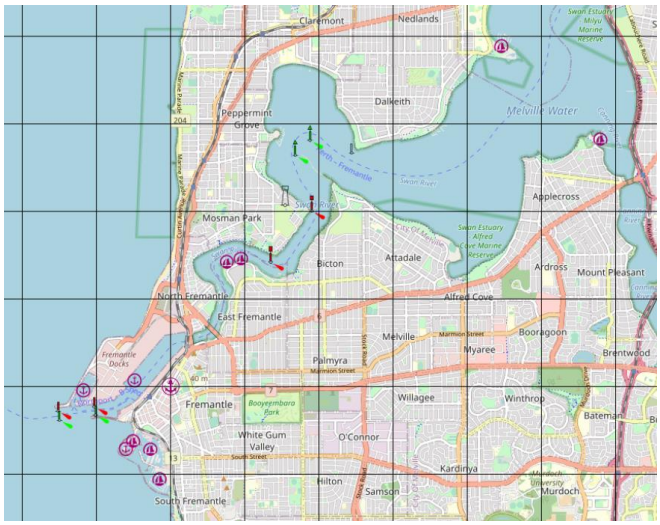

Figure 9 Sample Nautical Chart from OpenSeaMap [22]

2.6.2 Issues in adding OpenSeaMap

The way OpenSeaMap display its nautical chart is to present data in multiple levels with OpenLayers on the base map of OpenStreetMap, which enables it to contain all possible objects from OpenStreetMap. However, the tile url they provide only displays the information they overlay on OpenStreetMap, including aids to navigation, ports and so on, meaning that adding OpenSeaMap inside QGC need to overlay with other maps. Figure 10 shows the tiles get from OpenSeaMap tile server in QGC.

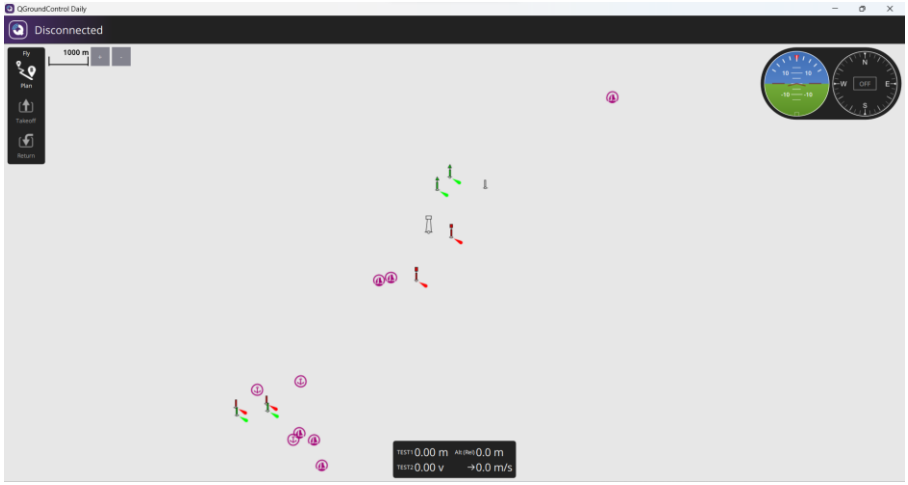

Figure 10 Tiles from OpenSeaMap

2.6.3 World Geodetic System 1984 (WGS84)

WGS84 coordinate system is a standard geodetic system used globally to define positions on Earth's surface. It is also the default coordinate system used by GPS and many other mapping and positioning technologies [23]. Because of different polices from different countries, usage of coordinates on map

may be added with offsets, as a method of enc encryption. Moreover, different company may have various ways of encryption for their map, thus makes it a need that, to add such map inside QGC especially when overlay with other maps, transformation of coordinate system of original map needs to be transformed to WGS84 coordinate system first.

2.6.4 Final Version of Customized Map

The final choice of customizing map is to overlay nautical information get from OpenStreetMap with Google Satellite Map. In this way, QGC can not only display nautical information with satellite map at the same time, but also to overlay other information based on OpenStreetMap such as wind, temperature, weather, and depth. Implementation of map is as shown in Figure 11 below:

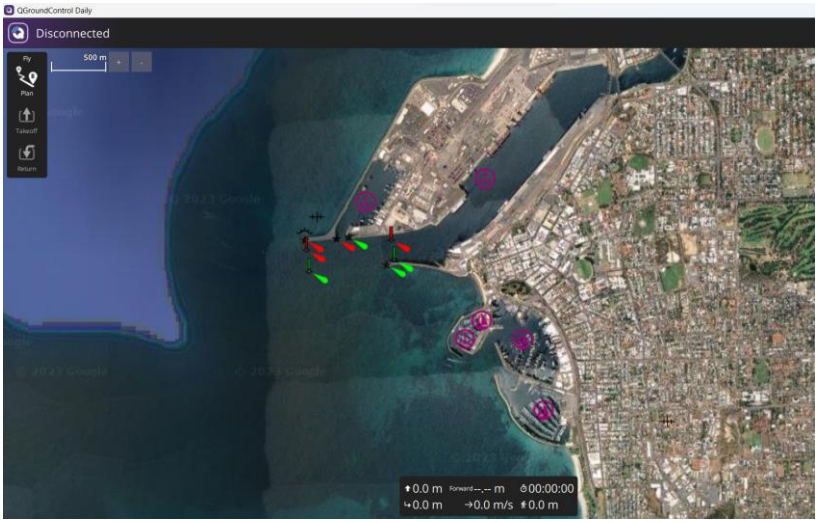

Figure 11 Implementation of Sea Map

Code to fix is shown in Table 5.

Table 5 Code to fix for Map Customization

Item	Changes
Modify GenericMapProvider.h	Add declaration of OpenSeaMapProvider class
Modify GenericMapProvider.cpp	Implement the _getURL function
Modify QGCMapUrlEngine.cpp	Add OpenSeaMap into _providersTable
Modify src/FlightMap/FlightMap.qml	Add functions of updating and overlaying sea marks.

2.7 Video Streaming

In QGC, the video streaming function is based on GStreamer, which is an open-source framework that provides a pipeline-based architecture for creating, processing and streaming media content. Usage of GStreamer library enables QGC to receive and decode multiple types of video sources including RTSP Video Stream, UDP h264 and UDP h265 Video Stream and so on. Besides, QGC itself can display video stream from camera on laptop running QGC.

2.7.1 Video Stream of Solar Boat

The video stream of solar boat is created by Raspberry Pi camera on boat. Figure 12 below indicates the video stream from on boat camera to top side computer running QGC, where camera is connected to raspberry pi through camera serial interface, and Raspberry Pi streams video to topside computer through UDP port.

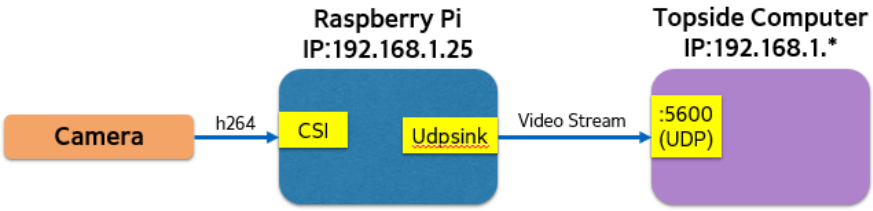

Figure 12 Video Stream from Raspberry Pi to Topside Computer

Due to the reason that source code of QGC does not include GStreamer library, when customizing QGC and video processing functions in QGC are needed, it is necessary to get GStreamer library intstalled correctly. From Raspberry Pi side, GStreamer is used to process and create different types of video stream according to requirements from users.

As the goal is to create video stream using Raspberry Pi camera and transmit it using Raspberry Pi, the video stream can be different types, or creation of which can be other libraries or Raspberry Pi built in functions. Final choice is to transmit video stream through Real Time Streaming Protocol (RTSP) over the network, with the help of command-line VLC (CVLC) pipe. Settings are attached in Appendix A.

3. Results and Discussion

3.1 Indicators Function Test

Test of depth indicator is carried out at Matilda Bay near UWA, where average depth is 3 meters. The goal is to test the functionality of depth indicator to read data collected from depth sensor through

usage of QGC Fact System and display it correctly. Since from hardware part, the wind sensor is installed, but electrical connection is not done, test of wind indicator is not revealed in the following test. Route and test result are shown in Figure 13.

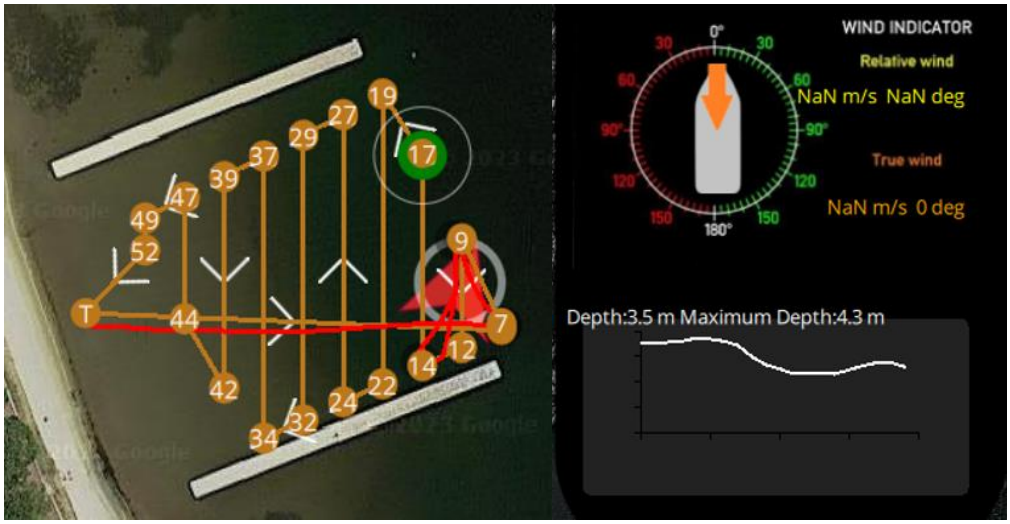

Figure 13 Route and Result of Indicators test

The figure on the left is test route of boat at Matilda Bay. From figure on the right, numerical result of depth, including record of maximum depth, is working correctly, and data collected are displayed and revealed in the line chart. For wind indicator side, though wind sensor is not installed, the "NaN" indicated in the numerical result display reveals that the Fact in the QGC for wind direction and speed is correctly accessed, otherwise QML code for numerical display of data will not be compiled, and nothing will be displayed.

Test of wind indicator is to use a set of data as follow:30 deg Heading, 45 deg Course Over Ground, 5m/s Ground Speed, 10 m/s Platform-based wind speed with 250 deg as wind direction, to calculate the true wind, and see if results can be displayed correctly and arrows can rotate accordingly. The theoretical result for wind speed is 13.5 m/s, with angle is 262.5 deg. The result from wind indicator has been shown in Figure 8, which is nearly the same as theoretical result.

3.2 Boat Payload Test

Besides the sail function of boat, the ability of boat to carry facilities conducting research tasks is also tested. Aim of the test is to add load on boat and determine the maximum weight of load the boat can carry. Test result is as shown in Figure 14.

Figure 14 Full Load Status of boat

The maximum load solar boat can carry is 14.75 kg. One point to mention is that the load needs to be put at the centre of boat, to ensure the weight is balanced over the whole boat.

3.3 Video Function Test

The aim of video function test is to know the performance of Raspberry Pi camera on boat, and the quality of video received at ground station. Test result is shown in Figure 15.

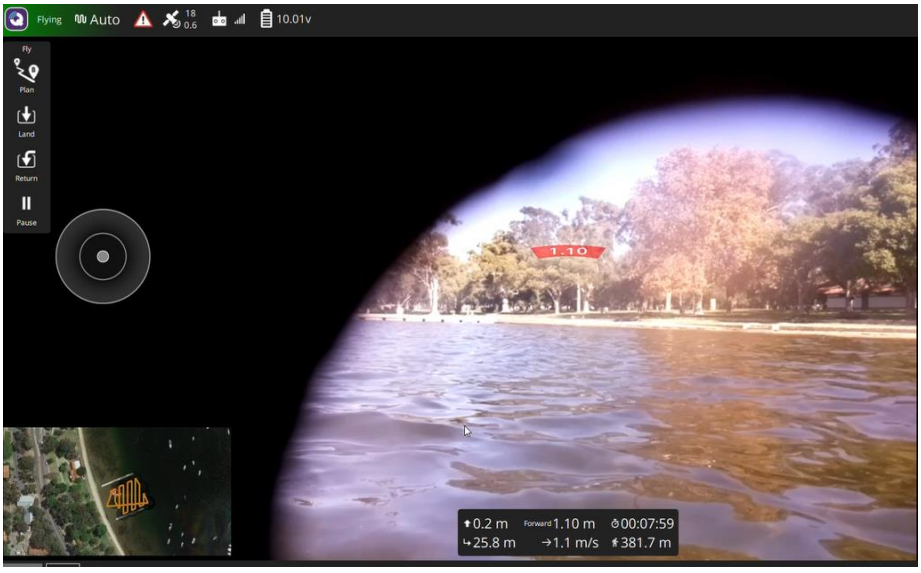

Figure 15 Video Get from Raspberry Pi

The ground station can receive continuous video from Raspberry Pi, though sometimes it will suddenly break off, then continues to stream. The functionality of QGC to capture photos and record video is verified to be working. Quality of video is relatively blurry but acceptable. This provides the ability of boat to conduct oceanographic research with usage of data collection and real-time processing.

4. Conclusions and Future Work

4.1 Conclusion

The aim of project was to customize QGC, including enabling it to display nautical information on map, having separate widgets to indicate and visualize the data collected from new installed depth and wind sensor, and complete the video stream function, to make the route planning and sailing of boat safer and more well-founded.

From the test, though wind indicator is not tested on boat due to the reason that wind sensor is not connected, basic functions of depth and wind indicator are proved to be implemented. Depth indicator, nautical chart and video stream are tested on boat. Implementation of depth and wind indicator, with other QGC widgets together provide direct and easy ways to observe the boat's behaviour and sailing status.

Customization of map in QGC for the solar boat is somehow not so essential at current stage according to boat's situation that the capacity of batteries on boat cannot supply the boat to sail on sea while majority sources of power for the boat are still batteries since the solar panel only provides a relatively small amount of power. However, the process in the project to customize the map provides an executable reference when it comes to adding other features on the map.

Completion of video streaming function provide the boat with ability to take photos or capture videos when sailing and save them either locally on Raspberry Pi on boat or remotely on Ground Control Station laptop, enabling the potential usage of boat to carry out tasks including detection of plastics, aquatic species and other research based on analysis of aquatic images.

Besides, the project also discusses about methods to customize QGC, which can be a reference for future development of solar boat since more and more sensors are to be installed on boat and requirement to develop widgets accordingly is of same importance.

4.2 Future Work

Despite installation of sensors, one issue being faced by the boat is that, currently batteries on boat are unable to supply stable voltages to support the working of boat, leading to a problem that boat the Raspberry Pi may not get enough voltage to run formally, which may cause the lost of communication between boat and ground station. Moreover, there is a lack of ways to control sensors on boat to turn on or off, presenting a problem that sensors will continuously consume power once the boat is turned on.

Moreover, it is worthy to explore usage of Raspberry Pi on the boat rather than using it only as a communication tool between ground station and boat. Examples include data collection, real-time analysis according to data collected and so on.

From the software side, integrating more features in QGC, especially based on boat, is also an interesting filed. The calculation of true wind can be more accurate with the boat's ability to collect data of roll and pitch from vessel. Map being used can be overlayed with more meteorological information. The installation of wind sensor is not completed, therefore test of wind indicator is also needed.

From hardware side, installation of wind sensor is of first demand. Besides, overall power consumption of boat, as well as the maximum time for the boat to run are needed. The power distribution on the boat may also need to be reviewed.

References

[1] D. McMillan, J. Zemp and T. Arbuckle, "SEACHARGER-SeaCharger Oceangoing Autonomous Boat," [Online]. Available: http://www.seacharger.com/.

[2] P. L. Constant, "Autonomous Surface Vehicles: Design, build of an ASV and autopilot integration," 2020. [Online]. Available: https://roblab.org/theses/2020-AutonomousBoat-Constant.pdf. [Accessed 17 05 2023].

[3] M. Connell, "Autonomous Surface Vehicles(ASV) Project: Design, build and upgrade of supporting hardware and collection of oceangraphic data," 2022. [Online]. Available: https://roblab.org/theses/2022-SolarBoat-Connell.pdf. [Accessed 17 05 2023].

[4] Z. Jia, "System Framework and Standards of Ground Control Station of Unmanned Aircraft System," Springer, Berlin, Heidelberg, 2011.

[5] G. Hattenberger, M. Bronz and M. Gorraz, "Using the Paparazzi UAV System for Scientific Research," 2014. [Online]. Available: https://repository.tudelft.nl/view/conferencepapers/uuid:b38fbdb7-e6bd-440d-93be-f7dd1457be60. [Accessed 17 5 2023].

[6] AiZheTeng, "QGroundControl Introductory tutorial: Software Introduction," 2018. [Online]. Available: https://www.ncnynl.com/archives/202103/4137.html. [Accessed 17 05 2023].

[7] qgroundcontrol.com, "QGroundControl User Guide," [Online]. Available: https://docs.qgroundcontrol.com/master/en/index.html. [Accessed 17 05 2023].

[8] T. Luis, V. Leonardo and R. Julio, "An interface based on QGroundControl for the rapid parameterization of flights from an embedded system for the control of an inspection drone," IEEE, Bogota, Colombia, 2022.

[9] R.-A. Cristian and C. David, "Extending QGroundControl for Automated Mission Planning of UAVs," *Sensors,* vol. 18, no. 7, p. 2339, 2018.

[10] D. Tristan, C. Noe, H. Ju-Hyeon and S. Hyo-Sang, "Implementation of Ground Control System for Autonomous Multi-agents using QGroundControl," *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems(RED UAS),* pp. 24-30, 2019.

[11] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith and M. Khalgui, "Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey," *IEEE Access,* vol. 7, pp. 87658-87680, 2019.

[12] mavlink.io, "MAVLink Developer Guide," [Online]. Available: https://mavlink.io/en/. [Accessed 17 05 2023].

[13] B. Rajatha, C. Ananda and S.Nagaraj, “Authentication of MAV communication using Caesar Cipher cryptography,” in *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials(ICSTM)*, Avadi, India, 2015.

[14] A. Azza, C. Omar, K. Anis, K. Mohamed and A. Tarek, “MAVSec: Securing the MAVLink Protocol for Ardupilot/PX4 Unmanned Aerial Systems,” in *2019 15th International Wireless Communications & Mobile Computing Conference(IWCMC)*, Tangier, Morocco, 2019.

[15] P. N, V. S and S. G, “Development of algorithms for MAV security,” in *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, India, 2016.

[16] A. Jose C. and C. Nuno A., “FASt - an antonomous sailing platform for oceanographic missions,” *OCEANS*, pp. 1-7, 2008.

[17] P. Maharshi, J. Shaphan, R. Rob, F. Scott and B. Gregory, “Autonomous Robotics for Identification and Management of Invasive Aquatic Plant Species,” *Applied Sciences*, vol. 12, no. 9, p. 2410, 2019.

[18] D. Matthew, G. Alistair and U. James, “An Autonomous Surface Vehicle for Water Quality Monitoring,” in *Australasian Conference on Robotics and Automation(ACRA)*, Sydney, Australia, 2009.

[19] Dronecode Project, “QGroundControl Developer Guide,” [Online]. Available: https://dev.qgroundcontrol.com/master/en/index.html. [Accessed 17 05 2023].

[20] Ardupilot, “Echologger ECT400 echosounder - Copter documentation,” [Online]. Available: https://ardupilot.org/copter/docs/common-echologger-ect400.html. [Accessed 17 05 2023].

[21] S. Shawn R., B. Mark A. and R. J. Sharp, “Establishing More Truth in True Winds,” *Journal of Atmospheric and Oceanic Technology*, vol. 16, no. 7, pp. 939-952, 1999.

[22] OpenStreetMap Foundation, “OpenSeaMap - The free nautical chart,” OpenSeaMap Community, [Online]. Available: https://map.openseamap.org/. [Accessed 17 05 2023].

[23] National Geospatial-Intelligence Agency, “NSG Documents Registry: Citation,” 08 07 2014. [Online]. Available: https://nsgreg.nga.mil/doc/view?i=4597. [Accessed 17 05 2023].

Appendix A

Useful pages:

QGroundControl Developer Page: Overview · QGroundControl Developer Guide

QGroundControl Source Code: GitHub - mavlink/qgroundcontrol: Cross-platform ground control station for drones (Android, iOS, Mac OS, Linux, Windows)

Ardupilot Wiki: Pixhawk Overview — Copter documentation (ardupilot.org)

QGroundControl Page in Stack Overflow: Newest 'qgroundcontrol' Questions - Stack Overflow

Set up for video streaming:

In Raspberry Pi terminal: raspivid -o - -t 0 -w 800 -h 600 -fps 12 | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8080/}' :demux=h264

In QGroundControl: