# Technical Specification

Autonomous Sailing Vessel

David Kelly - BSHC IOT - X13127390

14/5/2018

## Executive Summary

This report outlines the design of a software and hardware solution for automating sailing in a model craft. The report investigates the commercial and technical viability of using off-the-shelf components to implement a waterborne transportation solution. The system utilizes Internet of Things hardware such as the Raspberry Pi, Espressif ESP and Arduino platforms. The system will also use an android client for user input. In culmination, the system will let a user input coordinates through an Android client and the vessel's onboard software will make all other necessary inputs to guide the vessel to the destination.

The inspiration for this project came from looking at the environmental impact of current shipping practices along with the shockingly high costs involved with the process. While there are some commercial implementations into autonomous watercraft, there is a wide space to explore viability in low powered solar or wind-based solutions.

An overview of the system includes an Android client for user input. This will communicate with the vessel though the MQTT messaging service. The boat will have several ways of listening for commands including WiFi and LoRa Data.
The boat will parse the message payload and will make decisions based on the current environment, and through a set of predetermined behaviors, will make attempt to reach the user's destination. The boat itself will have a number of onboard sensors to aid in understanding the surrounding environment. These sensors include ultrasonic sensors, gyroscopes, pressure and temperature sensors and wind direction sensors.

## Introduction

The primary goal of the autonomous sailing vessel will be to transport a cargo across a body of water. Through the research, design, development and testing of this concept, a conclusion will be formed on the viability and scalability of the solution. The primary commercial application of the vessel will be in the transportation sector; moving non-perishable cargo. Another application may be aiding in marine biology, border patrol and search & rescue; areas that have seen some deployments of autonomous waterborne craft already. The low power requirements and long loiter time of autonomous should be quite a competitive edge in these sectors.

## Aims

The aim of the project is to develop a physical model craft and software solution capable of traversing water using rudder and sail. The model can be broken down to the following sections:

**Autonomous**
Sensors, devices and embedded software will provide collision-avoidance, control, navigation and pathfinding.

**Sailing**
The main method of propulsion will be by wind and other low power propulsion.

**Vessel**
The project will be self-contained and operate independently on water. A mobile interface exposing the craft's commands and telemetry will be developed in conjunction with the model.

## Objectives

To achieve the overarching goal of moving cargo across a body of water, these objectives will detail each step that will needed to be achieved to reach that goal.

1. **Model Craft** Production of physical model including traditional sailing control surfaces such as sail and rudder. A frame to contain cargo, sensors and controller hardware will be a core part of the design.
2. **On-board Control System** Development of sensor array, power solution, communications system and control system with the raspberry pi platform to enable environment detection and navigation.
3. **User Input System** Implementation of mobile application allowing command control over craft

## Deliverables

| Hardware | Embedded Software | Interface | Cloud Integrations |
|---|---|---|---|
| Model Vessel<br>Sensor System<br>Control Surface System<br>Communications System<br>Power System<br>Indicator System | Movement System<br>Sensing System<br>Command System<br>Communications System<br>Indicator System<br>Storage System | Android Command and Control Application | Google Maps<br>Blynk<br>AWS (Or Google Firebase) |

**Background**

From the earliest civilizations, the 15th century new-world explorers and right up to present day - sailing has played a key role in the progress of civilization and commerce. Traditionally wind and sail powered many early vessels. The famous Tea Clipper the Cutty Sark broke records going between London and Shanghai 19th century. At the turn of the century the ship began to lose out to the new Steam technology of the time. This marked the end of sail in the world of commerce as Steam provided a faster transportation solution. Mechanized seafaring vessels carrying the ISO Standard shipping container now make up over 90% of the cargo transport around the globe.

Autonomous travel is seeing huge growth in land-based craft. Tesla, Uber and Google are making huge developments in the autonomous driving space. While the area of autonomous waterborne craft is not seeing as much development in the transportation space, there is several projects currently underway.

**ONR and CARACaS**

Since 2014, the Office of Naval Research(ONR) has been developing solutions in the autonomous sailing space. Their first public demonstration of the technology included retrofitting small motor-powered patrol boats with their Control Architecture for Robotic Agent Command and Sensing (CARACaS) system. The system which was originally developed for the Mars Rover uses a portion of off-the-shelf parts to enable autonomous maneuverability and swarm functionality on the water. The use case of the retrofitted patrol boats is expected to be large vessel protection, pirate deterrent and limited offensive roles.

**Sea Hunter**

The latest development in autonomous naval defense, the Sea Hunter is a fully bespoke solution by Vigor Industrial for a Defense Advanced Research Projects Agency (DARPA) Anti-Submarine Warfare Program. Its trimaran hull and diesel engines give it great stability and huge transoceanic range. Its primary role is expected to be in mine clearing operations as well as surveillance and defense, often being compared to other existing naval craft in its operation capabilities. One of its most attractive metrics is the running costs - $15,000 against $700,000 of a comparable destroyer class vessel. As of February 2018, DARPA has concluded its research and has delivered the vessel to the ONR for further field trials and operational use.

**Open Source Underwater Glider**

Winner of the 2017 Hackaday Prize Alex Williams developed an innovative underwater submarine type of vessel that uses a selective flooding mechanism, wings and shifting internal weights to give the craft control. His project uses entirely off-the-shelf parts along with custom designed PCB boards and 3D Printed Hardware for the assembly. Alex states the he hopes a craft like this can be used in learning more about marine biology. The design has roots in a project from the Nonlinear and Autonomous Systems Lab in Michigan Tech University dating back as far as 2012.

**IRSC, WRSC and Mircotransat Challenge**

International Robotic Sailing Conference is a European based event celebrating the efforts of the public in autonomous sailing. Starting in Austria in 2008, the event hosts several challenges evaluating the autonomous vessels of both hobbyists and academic teams. Each year the vessels are challenged in several ways on their abilities including course following, collision detection, mapping and station keeping all within the confines of a closed waterway.

The World Robotic Sailing Conference is a similar conference established in the USA and hosts a number of similar evaluations. Over the years the events have come together in collaboration and hosted entrants from all over the world. The Microtransat Challenge is a ruleset posing one straightforward challenge to its entrants: Sail from Europe to the East Coast of the USA autonomously under wind power. Since 2010 the challenge has seen entries from engineering schools, private bodies and individuals. In the eight years of running, not one entry has ever made the journey citing loss of communications and crafts lost to fishing nets while making the transatlantic journey. The most successful entry in the challenge was by a private Norwegian company Offshore Sensing. Their entry Sailbuoy provides customers long loiter time environment sensing. They have completed voyages from Norway to Scotland and from Norway to the Faroe Islands. When it came to the transatlantic challenge, Sailbuoy made it 1,500 miles before the craft experienced software or sensor issues and began to sail in a circle.

In 2017 and ASPire project received international press coverage in the UK and Netherlands as it one of the first projects with the support of the Finnish Aland University and the European Regional Development Fund. It won in its class in the IRSC in 2017 and it is expected to make its first successful transatlantic crossing in 2018. One of the most unique aspects of the craft is that is has thermal imaging cameras for object avoidance and a class B AIS transponder to allow for information exchange with other ships with AIS, allowing it to inform nearby vessels that it is autonomous and to avoid or assist it if possible.

The vessel classification, rulesets and evaluations of the IRSC / WRSC provide a solid ground for measuring performance of any autonomous craft and will inform the evaluations and design of this project.

## Definitions, Acronyms, and Abbreviations

- IRSC - International Robotic Sailing Conference
- Pi - Raspberry Pi, Single Board Computer
- WiFi - Wireless Communication
- Arduino - Microcontroller
- ESC - Electronic Speed Controller. Controls power to motors
- LoRa - Low Bandwidth Long Range Wireless Communication
- Blynk - Control System for IOT devices
- Cayenne - Telemetry and Control system for IOT Devices
- MQTT - Message Queue Technology Transport. IOT standard for device control
- GPIO - General Purpose Input Output - Allows for sensing of voltages and powering devices.

### Materials and Technologies

A project in this domain has many options when it comes to achieving the end goals.

| **Function / Service** | **Chosen Product** |
|---|---|
| Development Languages | Python<br>Arduino C++<br>Android Java |
| Development Tools | VS Code<br>Android Studio<br>Arduino IDE |
| Microcontrollers and Single Board Computers | ESP 32<br>Arduino<br>Raspberry Pi |
| UI Prototyping Tools | Draw.io |
| Cloud Integrations | Google Maps<br>Blynk |
| Hardware Sensors, Actuators and Communications | Ultrasonic Sensor<br>DHT Sensor<br>Wind Angle Sensor<br>High Torque Servos |
| Materials | XPS Square Edge Insulation<br>Tronico Construction Kits |

**Python**

Scripting language Python is chosen for its close integration with the Raspberry Pi and high development velocity. With many IOT and Robotic libraries ready and available, it is the ideal glue for bring together the multiple parts of the vessels systems and decision making.

**Arduino C++**

The Arduino subset of C++ give the project simple to use commands and project structure for developing a robot type of interface. It contains many built-in libraries for interfacing with sensors and allows a developer to access control pins directly while maintaining a layer of abstraction about the hardware - allowing different hardware solutions to be trialed before integration.

**Android Java**

For a more bespoke user solution that will communicate over a multitude of radios like GSM, WiFi and SMS, a native Android application is the only way forward. With syntax very similar in some respects to Arduino C++, it will allow for mutual intelligibility between the two codebases and ease development load.

**VS Code**

The Windows developed Visual Studio Code (VS Code) is an open source code editor with a focus on speed and extendibility. It is gaining rapid popularity with developers and due to its excellent linting, debugging and Intellisense tools; a good platform to develop Python and Arduino C++.

**Arduino IDE**

The Arduino IDE comes with all the language support, compilers, code deployment, package and development board integration a developer may need. It is also open source and developed in such a way that it can be extended by the community.

While it may be used in its native form - its primary use in this project will be to feed functionality into VS Code which remaps the Arduino IDE into its own UI; bringing near one-to-one feature parity across the two editors, along with all the additional tools that VS Code natively has to offer.

**ESP 32**

This system on chip provides developers a platform to connect devices easily to the internet. With built in WiFi, Bluetooth and generous GPIO options, it will be used as the backbone of this project and will connect most of the embedded hardware together. It is also programmed in Arduino C++

**Arduino**

While is serves much of the functionality of the ESP 32, an Arduino Uno will be used in this project as a baseboard for a GSM/2G shield. Programmed in Arduino C++, this board and shield combination will allow for small amounts of data to be transmitted between the user and the vessel.

**Raspberry Pi**

The Raspberry Pi single board computer will perform two roles in the project. The storage system will be contained within the raspberry pi due to its abundance of USB Ports for data storage. The Pi will also functionally be the brains of the operation, using Python to listen for incoming messages from the subsystems and send back commands based on sensed and stored data.

**Blynk**

Blynk is a service that provides users simple connections and control over IOT devices. Including a Blynk configuration script lets you communicate with your IOT device via a smartphone application. Both devices will need an internet connection to communicate to the common MQTT server that Blynk provides. Blynk will be used to provide initial prototyping and control over the vessel while it is in development.

**Draw.io**

Draw.io is an open source cloud based diagram tool with integrations to google drive, and is used in the creation of mockups and use case models.

**Google Maps**

With the data being returned from the model craft including its GPS location, it is possible to send this data into the Google Maps platform to display its current location in real time on a map.

## Sensors

**Ultrasonic**

The ultrasonic sensor works by sending small pings into the environment and listening to the echo delay returning to the device. From this delay it can calculate distance.

**DHT22**

The DHT22 is a combination sensor that reads temperature data and humidity data in one device.

**Wind Angle Sensor - Rotary Encoder**

To measure wind, a small wind vane is constructed and mounted on top of a free moving rotary encoder. The encoder allows for angle measurement in the full 360-degree space.

# Requirements

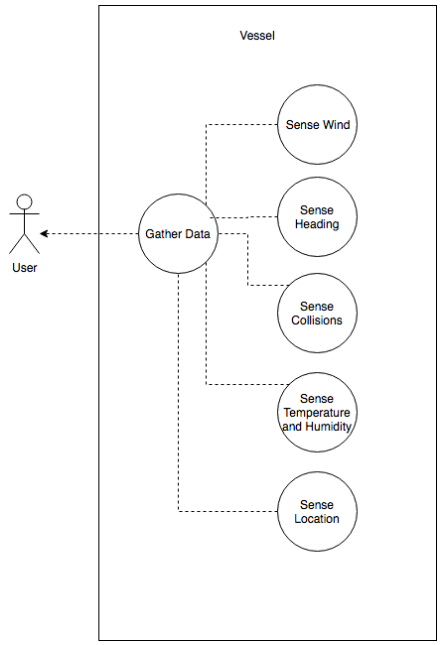
## User Requirements Definition

For the user, the vessel should have the following attributes.

In Non-Autonomous mode, the user should be able to use a simple mobile interface to maneuver the vessel. An internet connection will be needed on both the client device and the vessel.

In Autonomous mode, the user should allow for a user to enter a destination and the vessel will proceed to there and relay its status as best possible. An internet connection will be needed on both devices to pass coordinate information to the vessel. Once a course is set, the vessel should be able to proceed to its destination without user input.

## Functional Requirements




**Requirement 1 - Sense Environment**

**Description**

This subsystem will detect all possible environmental factors with the available sensors and make the data available to the other subsystems.

**Priority**

Very High

**Scope**

Sense environment and publish information.

**Use Case Diagram**

**Flow Description**

**Precondition**

System is powered on.

**Activation**

The system will begin sensing the environment from initial boot at a default polling rate.

**Main flow**

1. The system reads the values of all available sensors.

**Alternate flow**

A1. Power Saving - Change polling rate or turn off sensor.

1. The system will listen for an instruction from the C&C Subsystem.
2. Sensors are turned off or polling rate changed based on instructions.

**Exceptional flow**

1. The system will send a message to the C&C system if a read value is not in expected range.

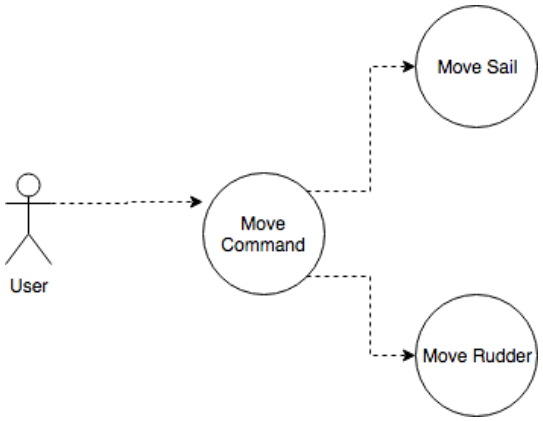
**Termination**

1. Sensing will stop when user has disconnected power to the boat.

**Post condition**

1. The system publishes values to the storage subsystem.
2. System continuously recalls itself.

**Requirement 2 - Move Actuators**




**Description**

This subsystem will control the movement of the surfaces directly.

**Priority**

Very High

**Scope**

Send power to actuators controlling sail, rudder and propeller.

**Use Case Diagram**

**Flow Description**

**Precondition**

System is powered on.

**Activation**

The system will begin listening for instructions from initial boot.

**Main flow**

1. The system will listen for instructions from the C&C system.
2. Based on received messages, system will activate the motors to move the surfaces.

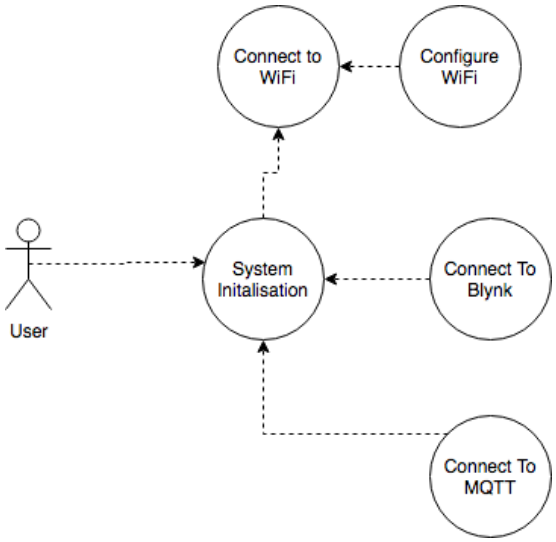
**Alternate flow**

None

**Post condition**

1. After movement has completed, the system will return to listening for messages from the C&C system.

**Requirement 3 - Communicate to Internet**




**Description**

This subsystem will receive messages from the user and relay back relevant telemetry through the use of 2G and WiFi.

**Priority**

High

**Scope**

Set active radios and transmit information to correct destination.

**Use Case Diagram**

**Flow Description**

**Precondition**

System is powered on.

The radios have already been preconfigured with the network. (WiFi credentials entered, 2G system has call credit and dedicated phone number.)

**Activation**

The system will attempt to establish a connection on boot.

**Main flow**

1. System will power both radios.
2. System will attempt to connect to networks through both radios.
3. On successful connection, the system will then listen on that radio for further instructions.

4. On receipt of instruction, the system will pass information on to the command and control subsystem.

**Alternate flow**

A1. Power Saver - Disable selected radio for period of time or entirely to save power

1. Based on received instruction from C&C system, disable radio.

**Exceptional flow**

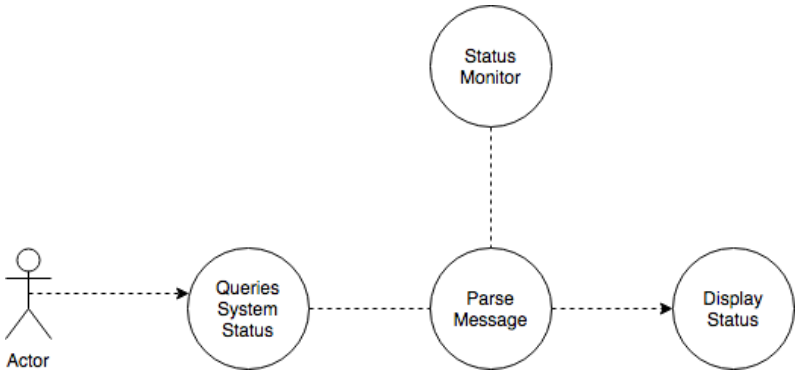
E1. No available connection

1. If system cannot establish a connection with the user, a message will be sent to the indicator system about system status.

**Post condition**

1. System continuously attempts to maintain connection unless instructed otherwise

**Requirement 4 - Indicator Status**




**Description**

This subsystem will receive messages from several subsystems and indicate to the usual visually the ongoing operations of the boat via LEDs.

**Priority**

Low

**Scope**

Receive messages and flash LEDs.

**Use Case Diagram**

**Flow Description**

**Precondition**

System is powered on.

**Activation**

The system will activate on boot.

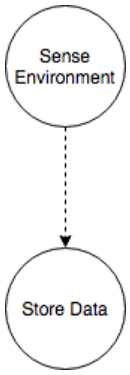
**Main flow**

1. System will listen for messages from all internal subsystems
2. System will parse message to determine origin and meaning of message.
3. On parse of instruction, the system will flash a sequence of LEDs to indicate it has received the message - informing the user of any potential errors.

**Post condition**

1. System continuously listen for instructions

**Requirement 5 - Store Information**




**Description**

This subsystem will record all sensor telemetry and make the data available to the other subsystems

**Priority**

Low

**Scope**

Store sensor data and send reports to other systems

**Use Case Diagram**

**Flow Description**

**Precondition**

System is powered on.

**Activation**

The system will activate on boot.

**Main flow**

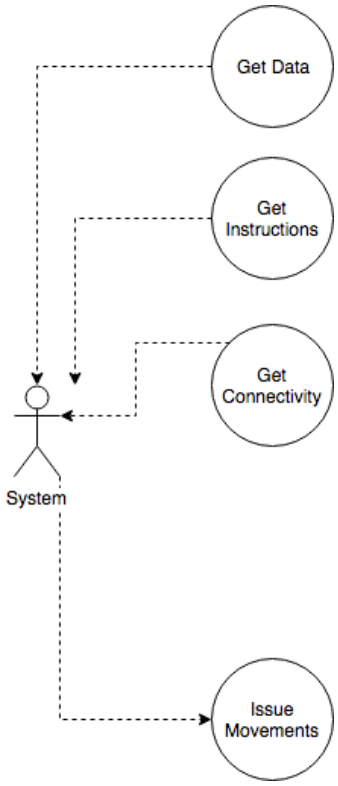
1. System will listen for sensor data messages.
2. System will store sensor data in easily queryable format.
3. System will return data to subsystems when requested.

**Alternate flow**

A1 Aggregate Reports - Help subsystems look at long view progress to destination.

1. On receipt of message with average flag, a average of a sensor value will be generated.
2. Average value will be returned to the subsystem.

**Requirement 6 - Command and Control Vessel**




**Description**

This subsystem will correlate all other systems together and based on available data or direct user instruction, send commands to actuators and sensors.

**Priority**

High

**Scope**

Parse gathered data and issue instructions.

**Use Case Diagram**

**Flow Description**

**Precondition**

System is powered on.

(Optional) System has connection to User Application.

System is gathering data.

**Activation**

The system will activate on boot and when a destination has been set by the user or has received a direct command from the user.

**Main flow**

1. System will communicate with storage and detect destination, heading, current location and sailing conditions.
2. Based on sailing algorithm and parsed information, issue move commands to sail and rudder subsystem.
3. Monitor progress to destination.

**Alternate flow**

A1 Manual Control - Through the use of the user application or cloud integration, user can enter direct commands to move actuators or switch power to sensors and subsystems.

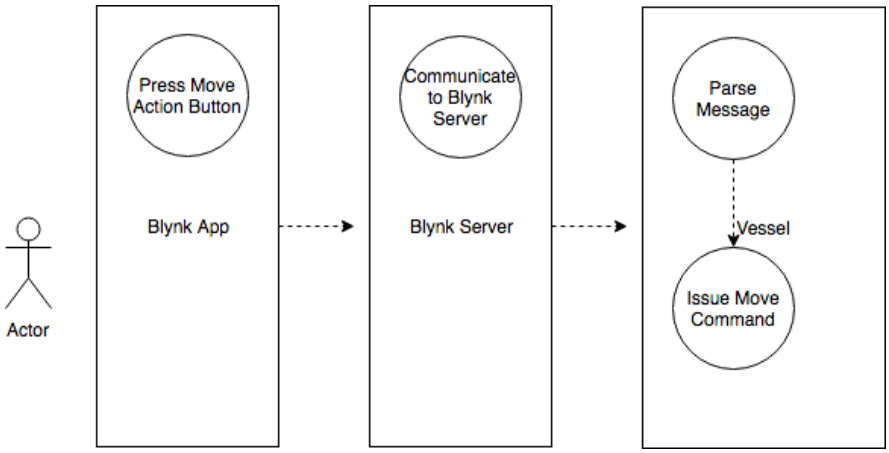
1. System will listen for direct user messages.
2. System override any previous commands or decision making and directly affect components on board the craft.

**Exceptional flow**

E1 Low Available Power - Handle situation where power use seems to exceed ETA to destination.

1. If command and control detects low available power, send messages to user to take action.
2. If user does not take direct control or issue new heading, assume no possible connection and make best judgement.

**Requirement 7 - Send Destination and Control Instructions**




**Description**

This subsystem will issue general or direct instructions to the vessel such as setting heading or moving control surfaces.

**Priority**

Medium

**Scope**

Accept user input and send to vessel

**Use Case Diagram**

**Flow Description**

**Precondition**

Application is installed on device.

Application has access to internet through radios.

System will listen for message from boat indicating it also has an internet connection and awaiting input.

(Optional) System has API Authorization with cloud integrations.

**Activation**

The system will activate on application launch.

**Main flow**

1. Present user with UI elements to accept destination input.
2. Send data to vessel via connected radio system

**Alternate flow**

A1 Manual Control - Move surfaces and configure sensors directly.

1. Present user with UI element to accept X/Y movement gestures and On/Off elements for direct control over vessel systems
2. Send data to vessel via connected radio system.

**Exceptional flow**

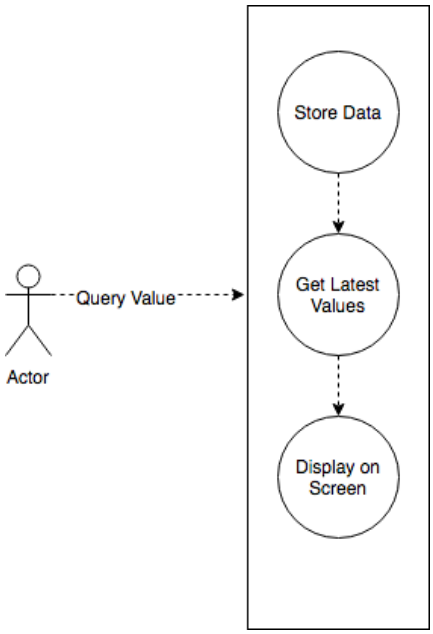
E1 Failure to communicate - Handle situation where instructions are not executed by vessel.

1. In event of data not being parsed by the system, enable timers within the system to engage a return to home function if no messages are received within the time window.

**Post condition**

System will continue to a parse data until vessel is switched off.

**Requirement 8 - Display Telemetry**




**Description**

This subsystem will receive the vitals of the vessel as the are published and help aid the user on sailing performance.

**Priority**

Medium

**Scope**

Display metrics to user

**Use Case Diagram**

**Flow Description**

**Precondition**

Application is installed on device.

Application has access to internet through radios.

System will listen for message from boat indicating it also has an internet.

**Activation**

The system will activate on application launch.

**Main flow**

1. Listen for incoming telemetry.
2. Display received results to user.

**Exceptional flow**

E1 Failure to communicate - Handle situation where no signals are received by vessel

1. Display to the user that the vessel is still listening for messages and awaiting input.

**Post condition**

System will continue to wait for messages until vessel is switched off or application is closed.

## Non-Functional Requirements

**Performance/Response Time Requirement**

In direct control mode, the control should emulate that of a radio controlled boat, drone or car. Input from the controls should be considered live and have an immediate input over the system.

In conditions where bandwidth is limited or at long range, it should be possible to relay small *instructions* rather than live continuous input. For example, one possible instruction could be “Turn the rudder left, power the motor to 30%. When the vessel is pointing North, return the rudder to the middle position and power the motors up to 100%”

**Availability requirement**

Availability in this project is a large consideration in this project. The vessel must continue to operate in adverse conditions where it may have lost radio communications or is expecting to lose power. In this sense the vessel must function in both high availability and low availability situations.

The mobile application will require a high availability of an internet connection via built in radios. Furthermore, integrations with 3rd parties are also required to be accessible always.

**Recoverability Requirement**

When waterborne, there are many situations that could occur that could be detrimental to the vessel. There could also be situations where one of more control system becomes non-functional and the pathfinding cannot proceed. Should the boat sense that is it not make clear progress towards its goal, it should inform the user that it needs intervention. Failing clear intervention, it should start either entering a return to home procedure or a low power GPS recording and 2G transmission drifting mode.

More generally, the system should be able to function with internet and low bandwidth connections and be able to restart systems if required.

**Robustness requirement**

The vessel has a high robustness requirement as it will be deployed in a hostile environment. With the high risk of flooding and erosion from sea water and fire from malfunctioning batteries, the design should properly contain components and generally maintain buoyancy in water.

**Security requirement**

Security is certainly requirement of this system. The vessel should only communicate over known WiFi networks, parse messages from known data sources and listen to instructions from known phone numbers while using SMS. Security on the application side can be considered more relaxed and accept the tradeoff that more importance is in getting data out and in of the application via any means possible.

(It may be considered at a later point that the vessel can attempt to connect to open WiFi

networks if it cannot connect to a known network as some ports like Dublin Port has an Open WiFi Network)

**Reliability requirement**

The system should be developed in such a way that the system can regain some functionality should a failure occur. This is assisted through the nature of the Arduino as programs can be very easily reloaded if there is any fault detected.

**Portability requirement**

The codebase for the system should have some elements of portability. If the code of the application is kept within the style of Arduino flavor C++, then the physical devices can be changed, upgraded and swapped to accommodate new features or design requirements.

**Extensibility requirement**

The system should take inspiration from the CARACaS system and allow for the simple addition and subtraction of components as the full solution scales up and down based on deployment needs.

## User Requirements

To deploy the boat, the user must have access to a number of tools for the sailing session to commence.

**Android Phone**

The Blynk application works best on an Android device. The user must also have permission from the owner to access the application (i.e. the user must have the unique link to access the application).

**Internet Access**

The phone device will need an internet connection and hotspot style capabilities for the vessel to receive initial instructions.

**Windows Laptop with Arduino IDE**

The boat will need to be tethered to a laptop to run initial checks on the system and see detailed output of the sensors from the serial monitor before it is deemed ready to sail.

## Environmental Requirements

### Arduino IDE

The Arduino IDE provides all the development functionality for scripting and communicating with the development boards. This can be running on Windows or Mac OS machines.

### Laptop

While a laptop is not essential for development, it is required for the setting up of a sailing session and the boat should be tested for its sensor outputs before the session begins. As such, a physical interfacing with the boat via USB should occur before voyage.

### SSH / VNC

For communication with Raspberry Pi's, an SSH or VNC client like VNC Viewer will be essential updating any code for the Raspberry Pi Platform.

### Internet Connectivity

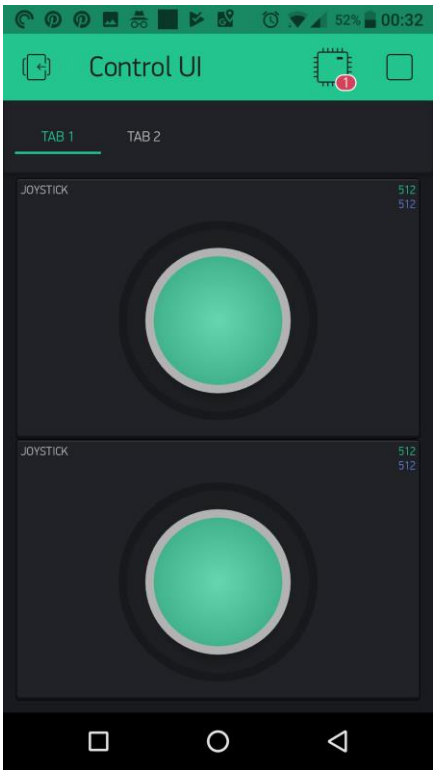
To test MQTT and Blynk, connectivity to the boat should be tested using through an available internet connection.

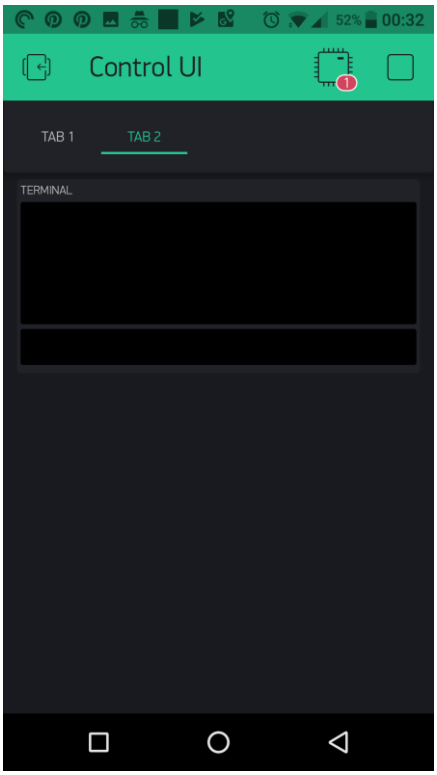
## Interface requirements & GUI

The interface requirements for the user application focus on functionality. The system requires.

- Text Input for coordinate entry.
- Map Integration for coordinate entry and route tracking.
- Switches for system control.
- Joystick Inputs for control surface actuation.

The interface for control of the vessel is provided through Blynk - an IOT cloud platform that provides messaging and control over embedded devices and exposes device functionality to the user through a android application builder.



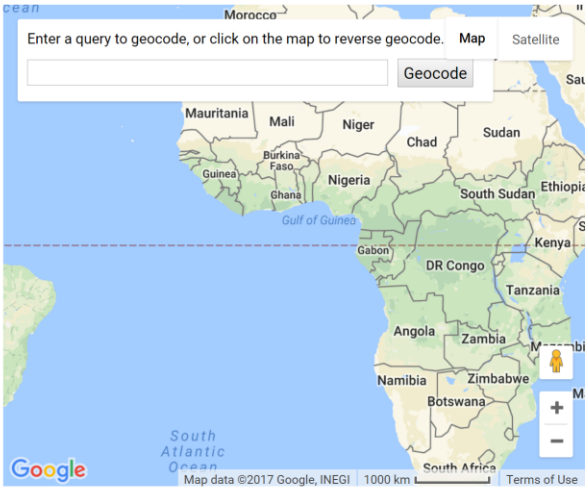




## Application Programming Interfaces (API)

The Blynk platform provides an API for MQTT like communication between the vessel and the android application along with other connected clients. Expanding on this further, the API provides virtual connectors and virtual pins for the user to hook functionality to. This functionality can have included making an on/off button, sliders or data displays. These virtual pins can be mapped to real pins on the Arduino thus providing the user an interface to the devices lowest level functions.

The GPS coordinates will also be passed from the boat to the applications and input to the google maps API to display the current location of the boat. The Google Maps API is able to handle the Latitude and Longitude from the boat use the same values as inputs to is display functionality, showing the exact reported location of the boat.

The Google Maps Geocoding API provides a direct way to access these services via an HTTP request. The following example uses the Geocoding service through the Google Maps JavaScript API to demonstrate the basic functionality.



## Manufacturing, Prototyping and Design

The code for all the embedded devices that make up the system is in Arduino Flavored C++ and is crafted in the Arduino IDE. The main functionality from the code is to take in sensor data, parse the data values and behave in a correct fashion to achieve the goal of sailing without continuous human input.

The sensor data is passed from a sensing board into a controller board which performs all the logic operations on the incoming date. Based on heading inputs from the user along with the sensor data, the main function of the application will return new values for the sail and rudder to help the boat execute a turn if needed and progress towards its destination. In addition to the main functionality, the user can manually pilot the boat if they wish using the android application.

Most Arduino-style programs follow a simple to follow define-setup-loop model of coding. At the top of each file, the inputs for that board are all declared and assigned. The setup initialized any external devices, objects and communication channels. Finally the loop is a block of code that continuously runs and typically updates values and forwards the values onto their destination.

The reason for this structure comes from the nature of the Arduinos themselves. On power up the device will run the Setup function once and follow on by running the loop infinitely for as long the device has power.

All the sensors rely on the digital pins of the Arduino for connectivity. The Gyroscope communicates on the I2C bus and the GPS sensor uses the Serial Bus for its communication. The system is powered off two 10,000 Milliamp/Hour batteries operating at the USB Standard 5 Volts and 1 Amp. There is an additional 9 Volt power supply that powers the servos separately as they have a very high peak power requirement that would disable some of the other devices if on the same power circuit. The system devices are connected through DuPont interconnect cables and breadboard circuits.

The models are created out of the XPS foam and are carved, sanded and sculpted with a variety of shaping implements.

The metal framing for component mount points is type of generic meccano construction kit that uses regular size screws and fittings to create the required housing shapes and structures.

**Prototype Version 1 - Airboat**

The initial physical prototype of the vessel was crafted during mid-December. This proof of concept tested multiple factors including viability of using an IOT device to control a sail, dealing with size and weight constraints, responsiveness of the system, communicating with the vessel wirelessly while on water.

The design represented minimum viable craft and served as a vertical slice of functionality. The system consisted of a single sheet of XPS Foam with mounting areas for a container and servos. The container housed a ESP 32 which had direct control over the actuators. The container contained a small power bank to power the ESP 32 and acted to offer some water resistance to the electronic components.

The ESP 32 connected to a local WiFi network and contained a Blynk integration. This allowed a user to manually control the servos. Mounted on the servos were more XPS foam formed to simulate a sail and rudder.

As the prototype was not yet ready for open waters, a propeller was added to the top side of the main foam board. This helped simulate an underwater propeller and moving the surfaces while in motion.

The prototype, now resembling an Airboat more than a sail boat, was then deployed into a container of water an allowed float freely for a period of hours. The tests revealed several outcomes.

- The closed cell XPS foam provides excellent buoyancy and weight was not going to a factor at this scale.
- The servos are quite exposed and an effort to keep water away from them should be investigated.
- While not the goal of the test explicitly, seeing the vessel move and turn in the container of water proved that this concept is viable and worth further exploring.
- The ESP 32 drained very little power while connected to WiFi and listening to Blynk commands. The battery drained after 30 minutes of powering the motor. It may be worth considering having two separate power systems - one for control and sensing and one for actuators.
- Waterproofness is a huge problem as the entire surface of the boat was damp while not ever being submerged. If the control systems were contained in a “hull” of a vessel, this “surface water” would accumulate and this would, at the very least, damage the control system.

**Prototype 2 - Clipper**

The second physical prototype builds on the lessons of the first and further refines all aspects of the vessel.

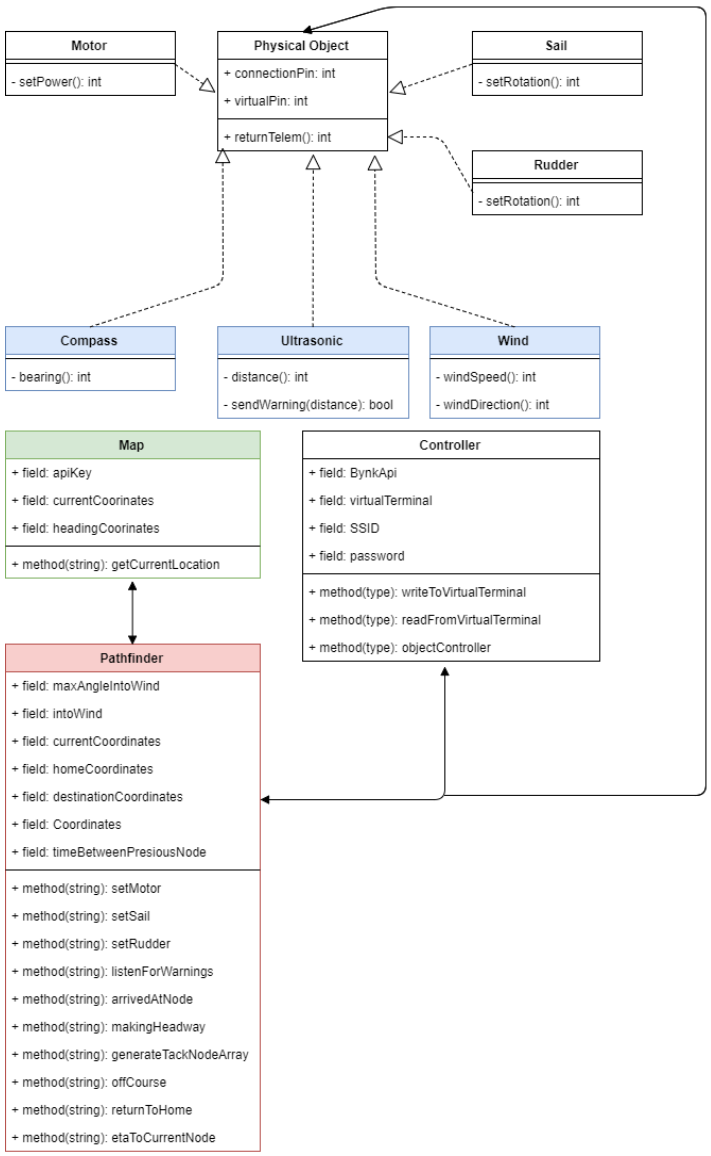
The hull of the vessel is now in a more traditional shape with a tapered bow, stern and keel. As the vessel has more depth, its buoyancy and carrying capacity is greatly increased. The hull features large carrying chambers which house the ballast and electronic systems along with two larger capacity batteries. The hull contains many metal modular support structures that provide strength and mounting points to critical components.

As buoyancy with the previous prototype was also an issue, additional space is available for ballast mounting helps maintain the boat in an upright position.

The vessel is near feature complete as it integrates all the sensing and support systems required for waterborne control and data feedback.

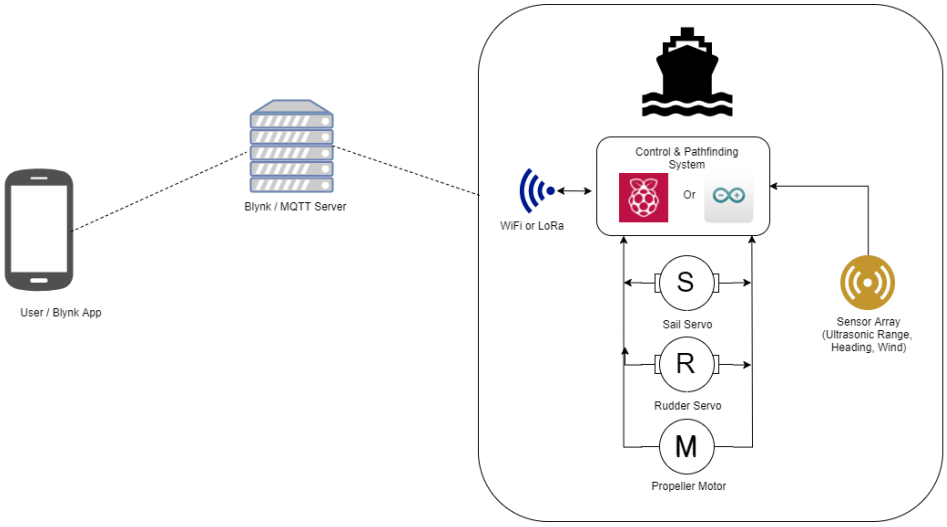
## Class Diagram

The following class diagram represents an overview of how each object interacts with each other . Each physical actuator or sensor will inherit from the physical object class. Each object can be controlled from either the main controller or the path finder, which will also have access to mapping functionality.


Motor
- setPower(): int
Physical Object
+ connectionPin: int
+ virtualPin: int
+ returnTelem(): int
Sail
- setRotation(): int
Rudder
- setRotation(): int
Compass
- bearing(): int
Ultrasonic
- distance(): int
- sendWarning(distance): bool
Wind
- windSpeed(): int
- windDirection(): int
Map
+ field: apiKey
+ field: currentCoorinates
+ field: headingCoorinates
+ method(string): getCurrentLocation
Controller
+ field: BynkApi
+ field: virtualTerminal
+ field: SSID
+ field: password
+ method(type): writeToVirtualTerminal
+ method(type): readFromVirtualTerminal
+ method(type): objectController
Pathfinder
+ field: maxAngleIntoWind
+ field: intoWind
+ field: currentCoordinates
+ field: homeCoordinates
+ field: destinationCoordinates
+ field: Coordinates
+ field: timeBetweenPresiousNode
+ method(string): setMotor
+ method(string): setSail
+ method(string): setRudder
+ method(string): listenForWarnings
+ method(string): arrivedAtNode
+ method(string): makingHeadway
+ method(string): generateTackNodeArray
+ method(string): offCourse
+ method(string): returnToHome
+ method(string): etaToCurrentNode

## System Architecture

This system architecture diagram shows at a high level the communication paths for data originating from the sensors, all the way to the controller application.



# Implementation

Commented [1]: code snippets, and explain whats going on, id probably take a significant line/s from the pathfinder class and explain how it works

The implementation of the project consists of a collection of scripts across the devices to act as a whole. As each device can run only one file at a time, the scripts will need to be flashed to each device before use.

### Hardware Test Script - Sensor System - Arduino Mega

The hardware test script is used to validate all the sensing functionality of the system. This file has no user input and is a simple initializing script that displays sensor status messages to the user.

In the script setup phase, it loads in all functionality libraries required the devices and initializes each device. The main loop sets the values of all the variable based on the incoming sensor data and publishes it to the serial monitor. The serial monitor output can be viewed by connecting the Arduino to the Arduino IDE.

### Communication and Control Script - Communication, Control and Movement Systems - ESP 32

The script has three core functions. First, it establishes connections with a known WiFi network. Second it establishes links with the user via the Blynk platform and relays the user input to the system. Finally it passes on move commands to the actuators, be they from the Blynk platform or from the decision making functionality.

### Sensing Script - Sensor System - Arduino Mega

This script functions much like the test script however its key difference is that it now sends the data across the serial bus to an awaiting control device to parse the data.

### Indication Script - Indicator System - Arduino

This script listens for messages from the control system and parses the data. After the parsing, various functions are called to indicate to the user via light and sound that the boat is performing operations.

### Display Script - Display System - ESP 32

The display script listens for messages from the control system and displays them on a small 16 character, 2 row monochrome display.

## Expansion on Pathfinder Class

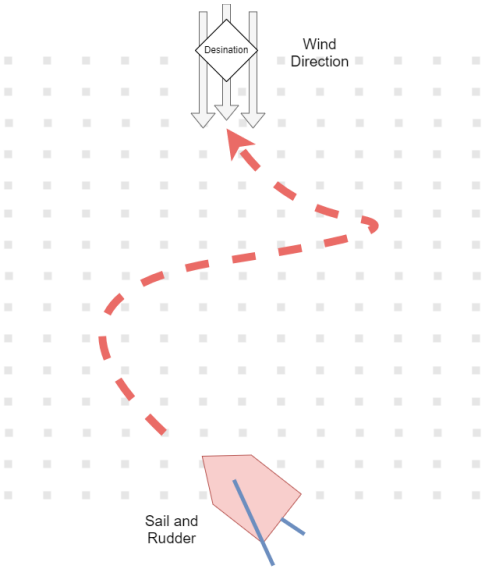
The simple response function provides most of the decision making while at sea.

**The problem with sailing**


Under motor powered sailing, pathfinding can be simplified to the following instructions:

- Get current location and destination coordinates
- Orient the vessel towards the destination
- Apply motor power until arrival

A part of this project will be to investigate feasibility into using wind and sail to propel the vessel, thus using electricity only computational power. When trying to reach a destination, the limitations of sailing have to be accounted for - not sailing into the wind being the most important.



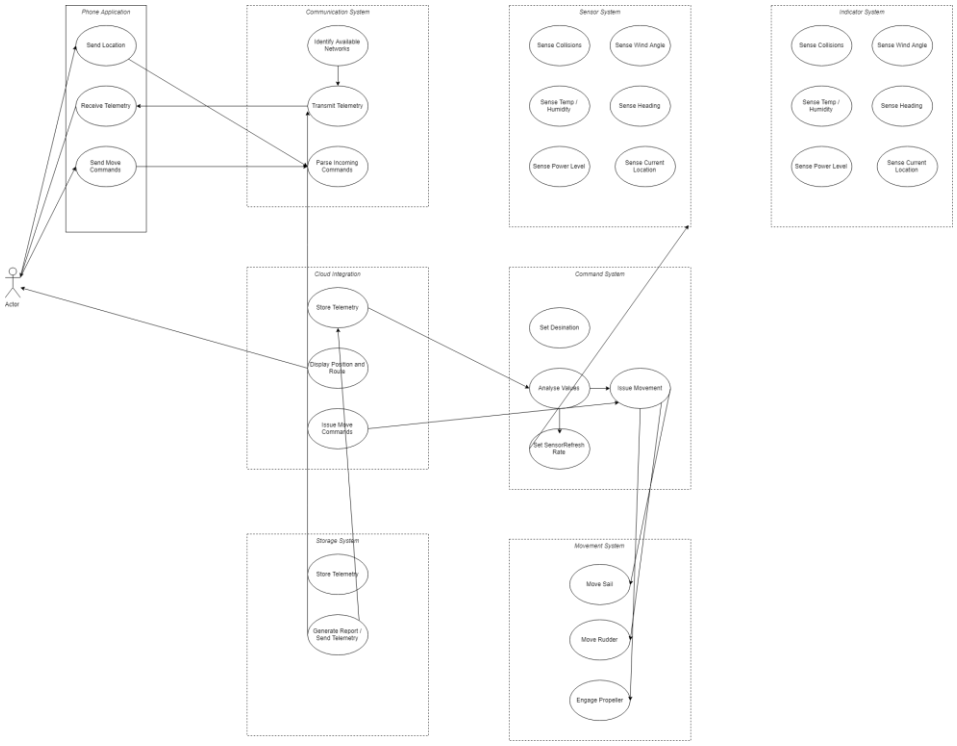

The solution implemented in the application is a multicase if statement to handle the main sailing scenarios like:

- Is our wind and destination dead ahead
- Is our heading into the wind
- In the wind behind us

Through simple if statements like this, the boat will be able to decide on each loop what procedure to take to navigate towards its goal.

A more novel solution for further research will divide the space between the destination and the vessel into a grid and proceed to the next best node on the grid. The best node will be determined by current wind heading, destination heading, and the max angle the ship can sail into the wind (a figure that is unique for every sailing vessel.)

## Overall Use Case Diagram




## Reflective Journals

### Development Journal October

During October my project manager Vikas Sahni was assigned and initial discussions took place around the goal of the project. Vikas made it clear that this project needed more research for its viability as I was still hashing out the finer aspects of the project. The goal for this month was to clarify direction and research the space around autonomous.

### Development Journal November

In November I presented my findings to my manager and we discussed the requirements for the project. The research into autonomous sailing proved very useful as several projects and conferences were discovered and served as a weather of knowledge within the problem space. The goal for the next month was documentation for the Christmas submission and some prototypes.

### Development Journal December

My personal focus for the month of December was on the prototype and equipment sourcing as I decided to build the model from scratch. A small codebase and IOT integration was created for the midpoint presentation along some associated documentation. The feedback from the presentation was clear that I needed write more about my research and shift some focus away from the prototype. Some feedback on the documentation phrasing was also provided. Over the Christmas period the document received some updates.

### Development Journal January

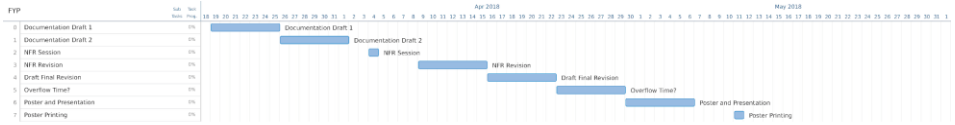
Very little development took place in January as this was exam week.

### Development Journal February

February was contained some updates to the codebase including some experiments with MQTT and Android. Most of the required hardware had arrived and a final assembly could be considered. One thing that posed quite a challenge in February was getting information from GPS and Gyroscopes. Further research into this was required.

### Development Journal March

During March myself and Vikas got stuck into the document and came up with a plan of action around refining documentation and research.




This timetable was set of actions for the next 30 or so days until the deadline. Within a week of this, the documentation was greatly revised and the timeline stayed accurate up until mid April.

### Development Journal April

Following on from the March update - near submission ready items were being created. Vikas now made it clear that he wanted to see more work on the code to get the vessel up and running. Following on from this the final hardware was decided and assembled.

## Testing

This is the questionnaire provided to test subjects for the user testing phase. During the user testing phase, participants were briefed on the project, give a guide of the systems and then asked to replicate the functionality.

| ID | TASK | Steps | Pass/Fail Yes/No | Issues |
|---|---|---|---|---|
| 1 | Power On System | 1. Connect USB Cables to Devices | | |
| 2 | Get Diagnostic Data | 1. Connect Device to Computer Via USB<br>2. Open Arduino IDE<br>3. Set Correct Comm Port<br>4. Open Serial Monitor and View Results | | |
| 3 | Enter WiFi Details | 1. Connect Device to Computer Via USB<br>2. Open Arduino IDE<br>3. Set Correct Comm Port<br>4. Enter available WiFi details in the control script.<br>5. Save and upload code to device. | | |
| 4 | Move Vessel | 1. Ensure Device has WiFi connection<br>2. Open Android App<br>3. Issue move commands on the joysticks | | |
| 5 | Set Autonomous Heading | 1. Ensure Device has WiFi connection<br>2. Open Android App<br>3. Enter numerical degree heading relative to north. | | |

User 1

| ID | Pass / Fail | Issues | Comments |
|---|---|---|---|
| 1 | Pass | | |
| 2 | Pass | Forgot Comm Port setting location | |
| 3 | Pass | | Found process tedious |
| 4 | Pass | | |
| 5 | Pass | Didn't know where north was exactly. | |

User 2

| ID | Pass / Fail | Issues | Comments |
|---|---|---|---|
| 1 | Pass | | The small usb cables can be fiddly |
| 2 | Pass | | |
| 3 | Pass | | |
| 4 | Pass | | |
| 5 | Pass | | |

User 3

| ID | Pass / Fail | Issues | Comments |
|---|---|---|---|
| 1 | Pass | | |
| 2 | Fail | Did not like IDE experience at all | |
| 3 | Fail | Did not like IDE experience at all | |
| 4 | Pass | | |
| 5 | Pass | Issue finding North | |

Testing was conducted with two friends and a family member with a prototype built of the system. The general feedback was that they enjoyed seeing the vessel turn and actuate on the input command. They all felt that the configuration process a major process and would preferred UI to enter data.

The key takeaways from the small experience testing are as follows:

- Create neater solution for WiFi configuration - perhaps a captive portal solution on power up.
- Have a simpler power cabling solution or perhaps a power switch design
- Remove the laptop entirely and bake in self tests to initial power on if possible.

## Lessons

The project posed many unexpected challenges and provided great learning experiences with each. Starting with the physical aspects, at many stages of the construction it became apparent that the ideal tool or material was not available, and a solution had to be implemented. For example, the specific foam material posed an interesting problem of being too buoyant - a problem I did not expect to encounter; in fact the opposite result was expected. The solution to add sand as ballast - provided the much-needed weight and stability was one that only became apparent during this design phase.

Another issue I encountered was seemingly random failures and resetting of the Arduino devices. The issue was localized down to occur only when the servos moved. The structure of the code seemed sound though only through experimentation did the issue become more apparent. By changing some of the timing functions and delaying the servo operations, the error disappeared. Following on from further research did it become clear that the servos were drawing too much power from the circuit and not leaving enough power for the Arduinos to work, hence resetting them.

On the software side of things, a similar issue occurred of random resets, but this time caused around the reading of an ultrasonic sensor. The way the sensor works is that it sends out an ultrasonic ping to the world and the delay in the response calculated. It eventually dawned on me that the entire loop was completing its run before the response has even come back. The computing adage about *the slowest thing a computer can do is wait for user input* has some resonance with this problem. In the milliseconds it took to wait for the echo, the Arduino was ready to execute another loop with another echo call. The solution to this was just like the servo experiments and a small delay was added to the loop to allow the echo to come back.

The user experience needs to be greatly refined at it was regrettable that the user tests took place at a later stage in the process their feedback was invaluable and aspects of the configuration that I had gotten used to are real pain points to a user and hinder market viability.

## Further Research

Through my research with the hardware, I was amazed at how much processing could be done with such low power devices. The space of autonomous water vehicles is a growing sector and interestingly, many of the smaller devices are having the same issues - getting lost at sea, as evidenced as no recorded autonomous sailing of the Atlantic Ocean. It is cost prohibitive at present to attach sensors like Lidar and Radar to small vessels and communication with a vessel while out at sea is expensive and not power efficient. One possible solution to this is to provide a virtual waterway with beacons that vessels can communicate with using low powered radio. The beacons could act as data storage and waypoints for a passing vessel.

The spirit of a solution like this comes from aviation. Planes travelling from Europe to the USA use the North Atlantic Tracks and the ICAO waypoints. As planes travelling over the sea have no visual indication or long-range radar, the planes follow preset paths each day while also using small digital beacons to orient themselves. Through these two mechanisms it is possible to turn chaos into order and send thousands of passengers each day across the Atlantic over the water safely. A similar solution with beacons and relays for autonomous vessels could greatly change the equation for small and low powered devices in autonomous travel.

## Conclusion

Looking at each objective within the project it is my opinion of the project hit each of the objectives it set out to achieve. The model vessel proves it is possible to create a sufficient housing for low powered IOT component at sea. The command and control system for the vessel is near feature complete and exposes much of the functionality required to get vessel sailing autonomously. The application and in general the user experience I feel is an area that needs further improvement. The process of tethering devices, flashing software and editing files to allow a user configure network access is not suitable for the average user. However, as a system to demonstrate the possibilities of the software and hardware integration, and to prove the business case behind autonomous sailing, I feel the vessel is a success.

# Appendix

## Project Proposal

Autonomous Sailing Vessel

David Kelly - BSHC IOT - X13127390

20/3/2017

# 1. Objectives

## The goal is to test the following concept:

Is it viable to exchange *time* for *money* and *energy consumption* when it comes to transport of cargo.

Traditional shipping via freighter is expensive. The largest consumers burn up to 500 tonnes of fuel per day, equaling $150,000 per day on fuel at current bunker fuel rates. Staffing is another high cost factor. Estimates for crew expenses per day average between $5000 - $7000 for bulk transport craft and $8000 - $10000 per day for tanker style craft.

In the financial crisis of 2008-2009, shipping companies really started feeling the pinch. In response they experimented with running their ships at slower than their design speed and increasing shipping times to save on costs.

By reducing the speed of the craft by 30%, they could reduce fuel consumption by up to 60%.

To compensate for the initial reduced capacity to deliver, the companies ran extra vessels on the shipping routes. (Notteboom, T. and P. Carriou, 2009)

Looking at this solution to adversity that market naturally adapted to, there is a solution I feel needs to be investigated: *If fuel and human cost are reduced to near zero, how much time could be afforded to shipping solutions.*

I will be developing one possible solution to this problem area in the domain of autonomous vehicles.

## The objective for this project will be to produce an Autonomous Sailing Vessel.

| Autonomous | Sail | Vessel |
|---|---|---|
| Sensors and embedded devices will provide collision avoidance, control, navigation and pathfinding. | The main method of propulsion will be by wind. | The project will be self contained and operate independently on water. |

In my implementation, I envision a craft that can use the power of the wind for primary transport. Telemetry and sensing will be powered by solar and battery stores. Finally, in situations requiring collision avoidance and low wind, thrusters can be deployed.

This implementation will be built iteratively and scaling on the axises of autonomy and control

Stage 1. Direct WiFi Control over powered motor

Stage 2. Direct WiFi Control over powered motor with sensors returning telemetry

Stage 3. Direct WiFi Control over powered motor with sensors overriding inputs (basic collision avoidance)

Stage 4. Return to home during deliberate loss of WiFi

Stage 5. Return to home during out of range loss of WiFi

Stage 6. Direct Control over sail and rudder

Stage 7. Direct Control over sail and rudder with sensors returning telemetry

Stage 8. Direct Control over sail and rudder with sensors overriding inputs and triggering motors (basic collision avoidance)

Stage 9. Return to home using motor, rudder and sail during deliberate WiFi loss

Stage 10. Return to home user out of range loss of WiFi

Stage 10 is the point which the solution is at minimum viable product. Additional stages are for investigation further innovation solution development.

Additional Stage 1. Test longer range communication solutions (2g, 3g, LoRa)

Additional Stage 2. Test long range controlled coastal sailing

Additional Stage 3. Test waypoint pathfinding

Additional Stage 4. Test sail craft performance metrics (Station Keeping, Lane Keeping)

Additional Stage 5. Test long range autonomous sailing

# 2. Background

Sailing has been the lifeblood of civilization for centuries. Earliest records date back to 3500 BCE detailing Egyptian vessels that use sails to provide propulsion. Developments in sailing have often been at the forefront of major milestones in history. Lateen Sails and Caravel Ships aided Vasco De Gama and Christopher Columbus in their expeditions. The Clipper Ship design allowed for the East India Company and the tea trade between Europe and China to flourish. Sail powered vessels fell out of favor at the turn of the century with the discovery of the steam engines and electricity. Innovations in naval design played a huge role in armed conflict in the 20th century. Inventions like the submarine, aircraft carrier and even the use of civilian craft during the evacuation of Dunkirk in 1940 only further demonstrates how the use of the sea by nations can have huge implications to society and world events.

In 2017 it is estimated that 90% of cargo transport around the world is performed on the sea. This has been facilitated by the ISO standardization of shipping containers in 1970 allowing for ease of loading and unloading in port and opening up a range of intermodal shipping options. Cargo shipping today is provided by a class of large, steel welded, fuel oil burning engine freighters. The largest ship class can carry over 500,000 tons. The ships typically have a lifespan of 30 years and it is estimated they contribute 2% to global carbon emissions.

Automated travel is a burgeoning marketing in 2017. Companies like Tesla, Uber and Google are all trying to solve autonomous transport on land. On sea, autonomous ships are gaining huge interest from companies like Rolls Royce and nations like Norway and Denmark. Google and Tesla are keeping a close eye on developments and supporting projects with their AI expertise to a handful of these projects. The US Navy already has autonomous patrol cruisers which assist in securing waterways and potentially dangerous situations at sea and plans to use them for more mundane missions like supply transport.

My particular interest area of autonomous sailing using traditional sails and wind is a well-defined problem domain and is not currently receiving much commercial interest. My

belief is that with innovations in sail and wing design, computing and with limited fossil fuels available, the use of renewable energy at sea needs to be investigated. Two autonomous sailing competitions take place every; a conference event in Norway, and an open entry competition which takes place from London to American or American to Ireland. The rulesets they lay out will provide the boundaries of my design research.

# 3. Technical Approach

This project consists of three areas which I need to develop.

A physical component, an IOT implementation component, and a Pathfinding / AI component.

On the physical side, technical approach will be to iteratively build models of increasing functionality and carry capacity. Models will initially be built out of a styrene material for its ease of crafting and high buoyancy. Additional mounting pieces will be 3D printed.

Sailing vessels can often be seen as complex craft with many sails, ropes and needing a large crew. This model will simplify the concept of a sailing vessel down to the following parts.

- Hull - Provides basic body shape and cargo storage
- Keel - Provides counterweight and resistance to sailing forces
- Sail - Provides propulsion
- Rudder - Provides heading
- Propeller - For emergency maneuverability and low wind testing

I will be modelling a wing sail as opposed to a traditional fabric sail as this will allow me to control the boat initially using only three servos and one motor.

For the IOT side I will be using a Raspberry Pi or Arduino to control the servos. In later iterations I will investigate adding sensors for collision avoidance, wind direction and investigate alternative communication methods to just Bluetooth and WiFi. Powering the controllers will be an array of batteries which will ideally be able to charge to a solar panel.

Finally, on the pathfinding side, I will need to implement a searching algorithm perform the following:

- Take as input a starting location and an end location
- Create an arbitrary grid between those two points
- Based on wind and sensor data, progress to the end location

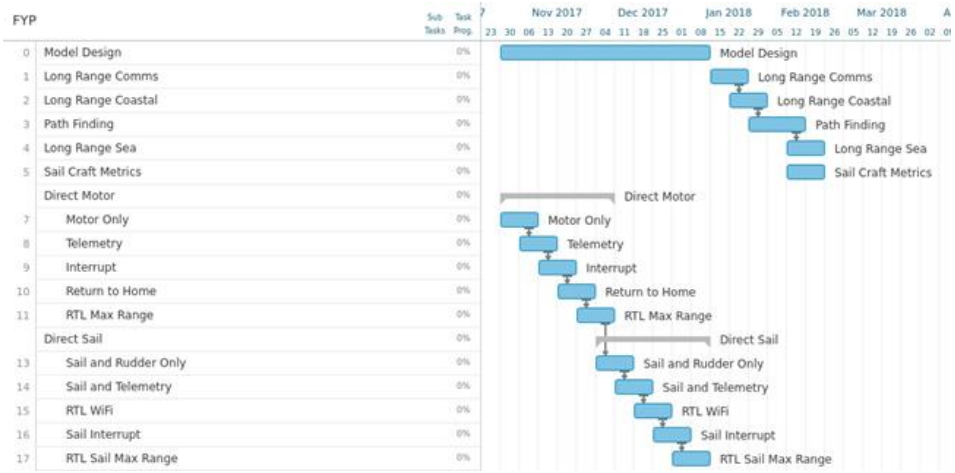
- Incentivizes progress to arbitrary waypoints on grid rather than direct path based on wind and sailing conditions

# 4. Special resources required

- STYROFOAM LB “Blue Foam” - Insulation and Model Making Material (Buoyant, Tensile and Fire Retardant)
- Raspberry Pi and Arduino
- Tools - Saw, Hot Wire Cutter, Soldering Equipment
- Sensor Devices incl GPS, LoRa Wan, SMS, 2G, 3G
- Actuators (Servos)
- Communication (Blynk will work for Midpoint)
- Waterproof Containers
- Waterproof Motors
- Body of water for testing

# 5. Project Plan

Gantt chart using Microsoft Project with details on implementation steps and timelines




# 6. Technical Details

Python or Arduino version of C.

Blynk for Android - A WYSIWYG Interface Builder for IOT Control

Under 2.4 Meters - 1.5 or less for Midpoint

Single Sail

Mono Hull

# 7. Evaluation

Given that sailing is a well understood problem domain, measuring the sailability of my craft will be against standard metrics derived from two sailing conferences around the world; the IRSC ruleset for competing sailing vessels, the WRSC ruleset for competitions. Logging and recording of data will also take inspiration from the ruleset Micro Transat open challenge.

Specifically I will be evaluating the project based on the following metrics:

Is it controllable

How far can it go

How fast can it travel

How much can it carry

How rough is the water it can sail through

How much energy does it consume

Can it navigate a narrow path

Can it avoid obstacles

Can it stay in one position in wind

Is it redundant or self righting

Is it cost effective

How much energy does it consume

Furthermore, I would like to investigate submitting the output of my research in autonomous sailing into the WRSC and the Autonomous Ship Technology Symposium to further contribute and gain feed from the autonomous sailing community.

# Bibliography

Notteboom, T. and P. Carriou, 2009. *"Fuel surcharge practices of container shipping lines: Is it about cost recovery or revenue making?".* Copenhagen, Proceedings of the 2009 International Association of Maritime Economists (IAME) Conference.