Autonomous Sailing Across the Great Salt Lake

Kyle Lemmon, Ted Goodell, Jim Squire, Bashar Al-Habash

***Abstract*—Robust autonomous ships provide a cheap and efficient approach for applications such as exploring hazardous waters that are unsafe for manned travel, mapping out vast areas of water using sensors, and providing autonomous transportation across water in a similar fashion to having autonomous vehicles travel roads. Our project sets to explore this technology by building an unmanned autonomous boat with a self-sustaining power source that will be able to operate for an undetermined period. Our goal is to have the ship be able to autonomously navigate across the Great Salt Lake in Utah.**

I. Introduction

One of the greatest benefits to humanity provided by computer technology is automation. Using computers to automate tasks normally done by humans has saved large sums of money and human lives. However, there are some jobs that are much more difficult to automate because they require software that can handle natural and uncontrolled environments with many variables to process. The area where this has generated the most interest is the automation of vehicles like automobiles and watercraft.

Interest in developing autonomous vessels for crossing oceans and other large bodies of water has recently increased in a similar fashion to interest in creating autonomous automobiles. Convenience and safety provided by autonomous automobiles are undoubtedly a main driving force for many researchers and global companies to perfect this technology. The differences between autonomous automobiles and autonomous vessels might not seem significant at first glance. However, when considering the different risks, safety measures, engineering, and computing needs of these two technologies, they begin to seem anything but similar.

Currently, a major key of the advancement by automobile companies, such as Tesla and Google, is having well-defined set of road regulation for the vehicles to follow. In contrast, water craft navigation rules are limited which creates safety concerns for automating them. With safety as a priority, we aim to develop an autonomous ship that will neither risk the safety of other ships, nor require the presence of an operator on-board. This will be accomplished by creating a 7 foot long, fully autonomous and self-sustained ship that will aid in the research and advancement of this evolving technology. This idea, inspired by the Microtransat competition that is later outlined in Section II, will help enlighten the feasibility and challenges of making human operated vessels obsolete.

While the Microtransat challenge focuses on boats crossing the Atlantic ocean, our focus will be to design and implement a ship that will integrate a few widely used and trusted technologies to cross the Great Salt Lake autonomously. Our ship will consist of our own fabricated hull, solar panels and batteries for power and an SoC board running Linux. The software we write for the board will take data from a GPS and other sensors and give commands to a rudder and drive motors to autonomously navigate to different pre-programmed waypoints on the lake.

In order to create our autonomous ship, we need to make sure that is fully mono-stable, and able to withstand the waves of The Great Salt Lake. The ship will have a waterproof compartment that will contain all of our electrical components, in order to prevent any water damage if the ship were to be submerged under water. Furthermore, the ship will have an easily accessible safety emergency stop button that will immediately shut down the motor and all other electrical components, reducing the risk of an accident during and after the our fabrication process.

For our systems software requirements, the ship will arithmetically navigate its course autonomously. To accomplish this, the ship is going track its own location and log it. This means that information such as, geographical location, velocity, and heading, will be continuously reported by the ship throughout its journey and then used by our, on-board, route algorithm to produce control commands to the hardware. The ship will also be equipped with an IP interface that allows for control commands to be issued by an external user and then executed by the ship remotely, over a local WiFi network as well as a terrestrial radio network. A dedicated web server and interface will be implemented to keep track of the information logged by the ship and to issue any external control commands. As we discuss later on in our stretch goals, we will adding interactive features to the web interface if time permits.

For our electrical requirements, we need a continuous power supply to the ship during its journey. Using rechargeable batteries and solar panels that are placed on top of the ship, the power will be independently supplied to the ship. During the night and when there is low direct sun contact to the solar panels, the ship is going to prioritize keeping the electronics powered on and not running the motor. To further conserve power, the ship will also turn off the motors when being drifter by the water along the desired route. Moreover, to comply with the maritime law, the ship's parameters will have white, red and green LED's to indicate its heading for other sea vessels. The white LED's will be placed along the stern, and red LED's along the port side, and green LED's along the starboard side of the boat.

There are more features that will help reduce risk of losing the ship throughout its journey and improve its performance, but depending on our progress speed and time limitations we are categorizing the following as time-permitting stretch goals:

- implementations of floating/stationary object sensors with

Manuscript received December 16, 2020

K, Lemmon, E Goodell, B. Al-Habash and J. Squire are Computer Engineering students at the University of Utah.

a collision detection and rerouting algorithm
- shallow water avoidance algorithm
- a dedicated web-server for the ship to issue real-time control commands and display status and images from a camera
- a total power loss recovery system
- allowing the IP interface to be usable over satellite radio

II. Background

Autonomous water crafts for different purposes and with different designs are being developed all around the world. Finland demonstrated a fully autonomous ferry in 2018 that traveled between Parainen and Nauvo [1]. Meanwhile, Norway is exploring creating autonomous container ships [1]. Other uses for autonomous ships include being able to map out large sections of the ocean or other bodies of water [2]. Autonomous ships can also be used as a method of garbage collection to clean the oceans. Potential benefits for autonomous watercraft include lower overall costs, increased safety, and being able to send ships to travel dangerous waters [2].

Microtransat is a competition that aims to stimulate this development by challenging teams to build small autonomous boats (maximum length of 2.4 meters) that can cross the Atlantic Ocean [3], and is the source behind our initial interest. This challenge was originally conceived in 2005 by Dr. Mark Neal of Aberystwyth University and Dr. Yves Briere of ISAE and was first attempted in 2010 by the *Pinta* developed by Aberystwyth University [4]. To date there have been 30 attempts, with only one of these (the *SB MET* engineered by Sailbuoy) successfully completing the voyage on the 26th of August in 2018 after traveling for almost 80 days [4].

There is great risk associated with crossing the Atlantic Ocean. Many Microtransat participants lost their autonomous ship by getting stuck in a fishing net, or by being picked up and taken by civilians and fishing ships [4]. This information, combined with the cost of travel for an Atlantic voyage, drove us attempt to cross the Great Salt Lake instead. Our ultimate purpose is not to win the Microtransat competition, but to create an interesting demonstration of what we have learned in our time as Computer Engineering students at the University of Utah.

III. Vessel Software and Hardware Design

A. Software Implementation

The system primarily consists of four software components each with a well defined scope, and firmly defined software interfaces. Software development is done against a well defined API, such that tests can be written prior to completion, and each component can be tested individually, with the interaction of other components able to be simulated. For the sake of simplicity and readability, each software component that is onboard the boat is named after the member of a ship's crew that would be responsible for carrying out the assigned tasks, were this a staffed ship as shown in Fig. 1.

The Linux based system run on the SOC uses a version of Debian Linux compiled for the SOC and board. Interfacing with off-board components was done through a combination

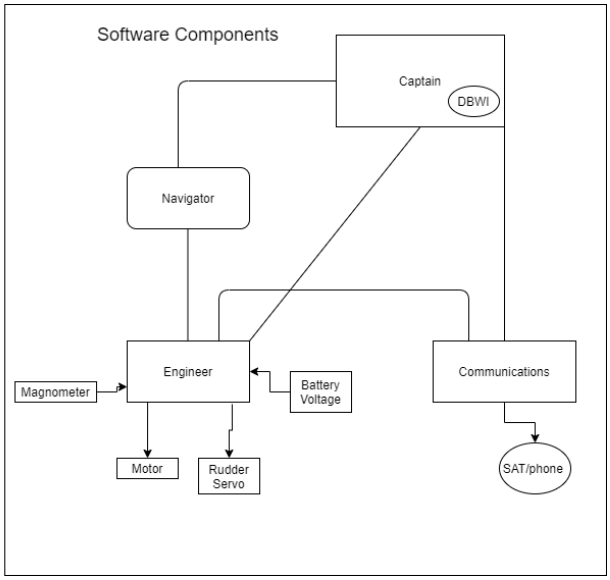

Fig. 1. Software component layout.

of file-mapped GPIO, PWM, memory-mapped I2C, UNIX sockets, and external software.

Our software was developed first by using header files to plan our high level component interfaces. This helped greatly when dividing the work, as the interfaces were well documented and implementation details were not necessary to design compatible software.

1) Engineer: The Engineer is responsible for interfacing with all hardware, including reading and tracking sensor values, and setting values for servos, actuators, and motors. Specifically these peripherals include the navigation light, bilge pump, rudder servo, thruster motor controllers, water sensor, magnetometer, accelerometer, charge controller, and GPS. The bilge pump control, navigation light, and water sensor all rely on simple GPIO to operate. The programming interface to the GPIO on the BealgeBone was provided through a library made by a GitHub user named Shabaz[5].

The accelerometer and magnetometer were connected using I2C and relied on the same aforementioned GPIO library. The code for the magnetometer not only retrieved sensor readings but also applied configuration data to its internal registers. The configuration data sets the internal analog amplifier to maximum and also enabled a digital filter that averaged the output across a large number of samples. These settings were necessary due to the weak strength of the earth's magnetic field and the low signal to noise ratio in the magnetic field in the boat. The code then places the magnetometer into burst mode which makes the magnetometer periodically generate new sensor data. Also, because there are unequal influences on each axis of the magnetometer it needs to be calibrated.

The calibration routine is based on an article written by Mike Tuupola and the pseudo code contained therein[6]. The algorithm keeps track of the maximum and minimum measurement on each axis of the two axis magnetometer. To correct for anomalous magnetic fields in the boat, also known as hard iron distortions, the algorithm averages the maximum

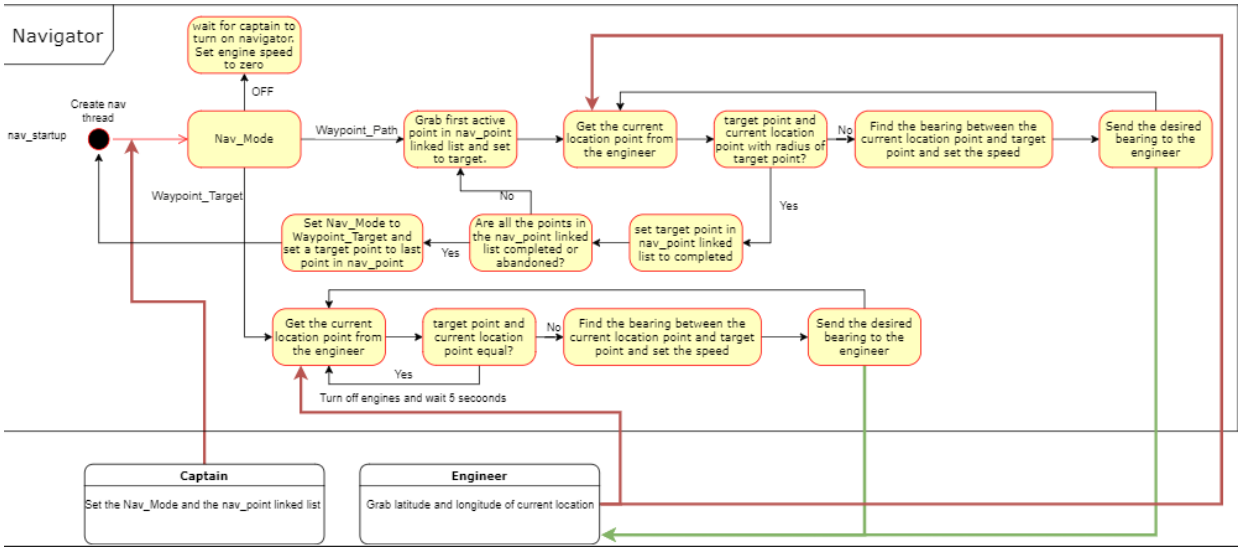

Fig. 2. Navigator flowchart.

and minimum of each axis and subtracts that average from new measurements. It also corrects for soft iron distortion that can come from larger ferrous objects like our keel. This is achieved by using the maximums and minimums to create scaling factors to remove any excess magnetic gain from each axis. The max and min values for each axis are saved to a configuration file.

The magnetometer is read in a loop inside of a separate thread. The thread waits for an external interrupt produced when the magnetometer sends a short pulse on its trigger line every time a new sensor reading is ready and then the heading is read from the sensor. Using this interrupt relies on a library called libgpiod. The heading of the boat retrieved from the magnetometer is then passed into a PID controller. If the engineer is in autonomous mode, then the rudder position will set to the output of the PID controller.

The GPS is also read in a separate thread using libgps. Libgps connects with a Linux service called gpsd which provides an abstraction layer for our USB based GPS. This thread takes the new data from the GPS and places it in a struct that is accessible from outside the engineer. The accelerometer is also read from this loop to determine if the boat is upside down.

The charge controller we chose was manufactured by Renogy and it featured an RS232 interface. The charge controller used the Modbus protocol over its RS232 interface which made it fairly easy to communicate with using libmodbus and a USB to serial adapter. The register mapping was publicly available online and we were able to retrieve much useful information including battery capacity and solar panel voltage.

The control of the rudder and thrusters was accomplished by changing the duty cycle on three PWM outputs of the Beagle Bone. The duty cycle was changed via an interface built into the Linux file system and function was created that allowed external code to change the output of a thruster by passing in a value between -100 and +100 to a function. The rudder was controlled is a similar manner but its limits were -90 to +90 which corresponded to the 180 degree range of the rudder servo.

2) Navigator: The Navigator's primary responsibility is to keep track of set of points on a course and issue commands to the Engineer to effectuate the course. The set of points, consisting of latitudes and longitudes, are set by the captain and the navigator stores the sequential points in a doubly linked list data structure. The navigator has 3 primary modes:

- Off mode, where the boat's speed and heading are set manually
- Course mode, where the navigator set the boat in autonomous mode
- Holding mode, where the navigator holds the position of the boat within a specified radius of a point

. The navigator will enter holding mode once the course has been completed or if set by the captain. The points on a course are marked with a status to show if the the point has been reached, if the point is currently being pursued, if the point is yet to be pursed, or if the point is abandoned and no longer valid. In the course and holding mode, the Navigator makes decisions about moving in the short term. These decisions are based on the current and target position of the boat which dictates the head and the speed of the boat given to the engineer. The Captain can tell the Navigator what parameters to minimize or maximize, such as to preserve power, or to get to the destination as fast as possible, or something in between. The navigator is started in single thread, and the flow chart seen in Fig. 2 shows the functionality described previously. Finally, the navigator saves its state when executing and recovers that saved state if the navigator is not shutdown safely by the captain, implying that our program has been reset after an ungraceful shutdown and our course should continue from where it was last saved and not from the start of the course.

3) Communications/Logs: The Communications component is responsible for aggregating information generated by the other components, transmitting that information when

appropriate, and receiving commands from off-board clients. This software component also serves to create logs from information sent from the other software modules as well as recording automatic status reports. All software modules can also send custom emergency messages to the Communicator to be logged and transmitted. The Communication module includes a TCP/IP server which can be used to receive commands and send status reports and other information over Wi-Fi for close range transmissions. The TCP/IP server could also be used for satellite internet for global communications, however a satellite modem has not yet be implemented due to cost and because it was not necessary for our demonstration and testing in Utah. The Communications component also can use the Automatic Position Reporting System (APRS) for receiving commands and sending status information. Transmissions from the boat will prioritize using the TCP/IP connection if one is present, otherwise they are sent using APRS.

APRS is an amateur digital radio format aimed at providing a location and status sharing network. In addition to sending out specifically formatted reports, radio operators can send messages to one another or to themselves on the network. There is a system of digital repeaters and internet gateways set up by other HAM radio operators which greatly broadens the network's reach. Packets are captured by gateways and broadcast on the APRS/TCPIP network, which the site APRS.fi pulls its information from. Our project uses this APRS.fi service for sending position reports which then updates a map with the boat's location on their website and app. APRS uses the AX.25 frequency keying link layer protocol, which requires a soundmodem. Our project uses the publicly available soundmodem software Dire Wolf. APRS allows some flexibility in parts of the message formats which allow us to receive commands on the boat and send status information back.

Commands use a simple format to either allow the modification of the current boats operation or to request specific status information. The Communications software is designed so that buffers containing these commands from either radio packets sent using APRS or by the TCP/IP server will be similarly formatted, which means both sources can be processed by the same function. These commands are parsed inside the communications module, and then any commands not related to communications or for status requests are sent to the Captain to be executed.

4) Captain: The Captain is in control of the high level health of the boat's systems, including the power system, navigation modes, off-boat command processing, and other tasks which did not fit into other software components. The Captain sets priorities for the other components, manipulating their state based on parameters like battery voltage, solar panel voltage, and commands from off-boat devices.

5) Telemetry Server: The telemetry server was designed as an always-online server that logs telemetry data sent to it by the boat's computer. This information is retrievable via web pages, with potential maps integration. Additionally, the course of the boat can be modified through through this server, and the information will be sent to the boat when the boat next checks in. Unfortunately, we did not have time to fully implement all of these features, though the basics functions of receiving and acknowledging telemetry data, and displaying it are implemented. This was never tested with the boat, however each component (ack-ing a packet, receiving/processing/displaying positions statuses, and choosing points on a map) was tested and worked.

6) Command Client: The Command Client was an idea for software that sits on a computer that has an IP connection the the boat's computer, and would allow remote control over IP of the boat. We found that our connection to the boat was significantly more reliable than we had originally thought, and we never found the need to implement this. We could simply read inputs from the terminal as the boat control software was being run on the BeagleBone over ssh with no difficulties.

B. Software Interfaces

The four on-board components communicate with each other via a hard-coded API. This allows for the testing of individual modules and simulations of individual modules. All of the software components were built against rigorously defined interfaces that allowed for blind testing. This permitted every software component to test its inputs and outputs individually without the need to be connected to the other systems. That is, the tests mocked the behaviour of the other components to thoroughly test each software component before the boat was ever built. The only exception to this rule was the Engineer component, which by necessity required testing against hardware.

C. Electrical

The electrical system can be broken into three groups: energy management, computing and sensors, and motor control. Each of these groups must interface with each other and will need to be stored in waterproof enclosures. The description of these groups are listed below.

1) Energy Management: The energy management electrical subsystem consists of solar panels, batteries, a solar charge controller. We had originally planned to use a charge sensor to read the battery charge, however the charge controller we used could communicate this information over a serial line. The purpose of this subsystem is to provide power to the craft in all but the most extreme cases.

There is a portion of the Captain software component that is also considered a part of the energy management system, and that component monitors the output of the charge sensor, and below a certain critical point, puts the system into a power conserving state. This is necessary to mitigate the risk of losing the craft due to loss of telemetry data. At the very least, the craft should always report its location so that it may be recovered.

The required critical components in the energy management system are solar panels in excess of 150 W maximum output, a solar charge controller, and a battery bank in excess of 450 Wh. We did not encounter difficulty sourcing a solar panel or charge controller, despite slightly increased demand due to COVID-19.

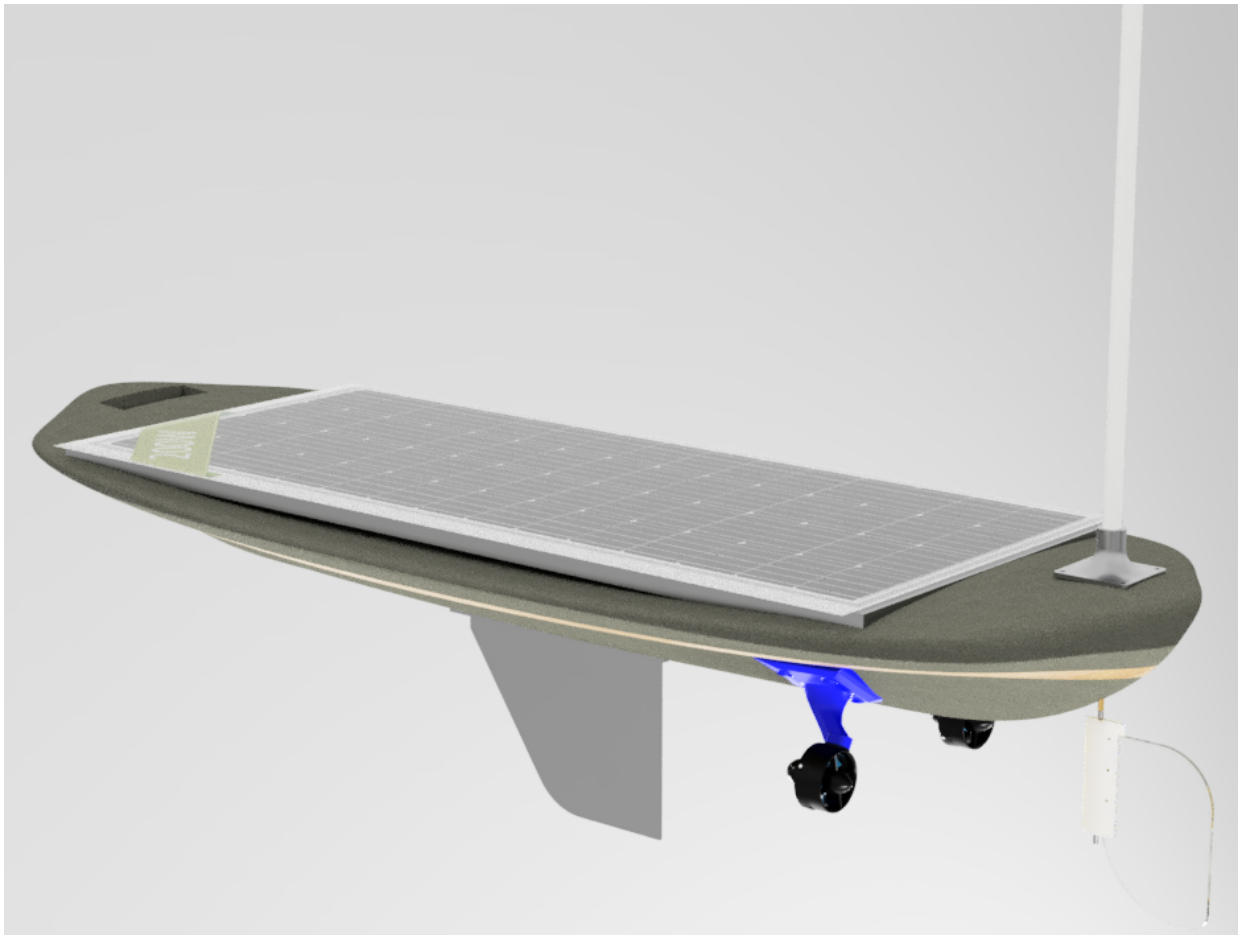

Fig. 3. Boat model from Autodesk Fusion 360.

2) Computing and sensors: The computer control subsystem consists mainly of an SOC board running Linux, with our custom software running on it. We have chosen to use a BeagleBone Green Wireless for this task, as we already have several available, and they are capable enough to handle all of the tasks we need to complete. We will also be using an Atmel ATSAMD51 based MCU for our backup processor to deliver telemetry in the event of a systems failure. We already have this MCU and a two-way radio which it will use to send telemetry information.

The sensors we need for autonomous navigation consist of a GPS and a magnetometer and we already have both acquired. The GPS will interface directly with our SoC using RS-232 serial communication. The magnetometer communicated using I2C. We will need to procure a 3G cell phone modem for remote IP communications. This will most likely also use RS-232 serial communication.

3) Power Delivery and Sensing: The power delivery system is the electrical system with the most components, however the crucial components are two brushless motors, two speed controllers, and one servo. We already have procured a single motor, speed controller, and servo, but have yet to determine exact criteria for these components. Once these current components have been tested and we are confident that they will meet our requirements, we will order another of each.

The purpose of two motors and one rudder is for redundancy. Given these three components, any one may fail and the craft still able to move and steer. The motors are mounted near the rear of the craft, one on the far left side, and one on the far right side, and the rudder is mounted in the rear of the craft, along the center line. In the event that a motor fails, the other will be able to provide propulsion, while the rudder can offset the misaligned center of thrust. In the event that the rudder fails, differential thrust on the two motors can be used to shift the center of thrust away from the center line of the boat, compensating for a potentially stuck rudder, and steering the boat.

D. Mechanical

The mechanical systems of the boat are mostly static, and are simplistic in nature. The shape of the hull and center of mass are designed with the idea that the boat only has a single stable position in the water, and will always right from a capsize or turbulent waters. The construction of the boat consists of a solid closed-cell foam interior, with sections hollowed out for batteries and electrical components, and a fiberglass and resin exterior. The foam interior makes the sinking of the boat unlikely, since there will always be significant displacement of water. The foam comes in boards that are significantly thinner than the depth of the hull, so we had to slice the CAD model

Fig. 4. Loading the boat to test at Decker Lake.

into layers, cut the layers with a CNC mill, and glue each layer together to form a hull out of the foam.

The 3d printed motor mounts are attached to a layer of plywood in the boat using threaded rods. The shaft connecting the rudder to the servo was also secured to the plywood, providing a rigid and secure fastening of the rudder components to the body of the craft. The rudder blade was cut from polycarbonate and secured to the rudder shaft with a 3d printed coupler.

There is a painted steel keel that is secured firmly to both the internal plywood layer and fiberglass exterior of the boat. This steel keel also helps with the stability of the boat both by lowering the center of mass, and by creating large drag vectors that oppose horizontal movement of the craft.

The boat is designed also to have a steeply increasing area of displacement such that changes in the weight of the boat have a disproportionately low impact on the drag of the craft in the water. This allows us a wide range of options if more battery capacity is required, or other hardware changes must be made.

To ensure that our boat would behave in water the way we expected, we 3d printed various prototypes and moved them in water. Also, we moved the boat in the water by pulling a string before attaching any of our electro-mechanical components to it to ensure its stability.

The three most difficult resources to acquire for this project were access to a large (at least 4'x8') CNC mill for cutting a positive mold for our hull, the space required to store and build the hull, and a welder and other equipment. Ted Goodell has access to such a CNC mill which we used when we were finished with our hull design, and Kyle Lemmon has welding equipment, a large backyard that can be used to work on the boat, and a garage that can store the boat.

We've also never fabricated a whole boat hull before. However, one of our team members (Kyle) has repaired boats using fiberglass and has experience welding. Also, Ted has experience with 3D modeling with an example of the boat shown in Fig. 3 and CNC machine operation. While this experience went a long way to reduce our risk, we also did research on design and construction methods before beginning fabrication.

E. Testing and Integration Strategy

To improve the chances of successful integration, we defined detailed software and hardware interfaces early on in the project. We planned different testing platforms at various stages of the project that involved connected all of our hardware using several progressive test benches. This included endurance tests of the motors connected to our power system to ensure that the motors would perform well with continuous use in water and that the power system would be able to sustain continuous use for multiple days. We also tested that data could be read from the sensors and charge controller and that the motors and rudder position servo could be effectively controlled using software. We also evaluated the software's feedback loop to control the rudder to maintain a heading by spinning the boat and checking the rudder response. The bilge pump motor and sensor were also tested, and after discovering undesired behavior a capacitor was added to the sensor lines to limit how fast the motor could switch on or off. These early bench tests gave us confidence to begin testing on the water.

Our water testing was conducted at Decker Lake in West Valley City, Utah. A padded wooden boat carrier was built for rolling our boat around, and it also allowed us to use a boat trailer as shown in Fig. 4. While testing at Decker Lake

the boat was initially tethered to ensure easy retrieval in case something went wrong. Here we tested our software with a laptop connected to the boat using Wi-Fi. During this testing we were able to identify and resolve several issues. Our GPS was not providing a reliable lock on the water which led to us modifying the antenna position to not be under the solar panel. We also discovered that without a lock GPS positions would be reported as not-a-number which would cause software that relied on those values to crash until this problem was identified and resolved. However after multiple days of testing we were able to successfully demonstrate that our boat was capable of autonomously navigating a series of waypoints.

F. External Risk Assessment

1) Potential Failures During Operation: In order to design our boat to be able to handle the extreme conditions of open waters, we conducted a risk assessment of system failures our boat may experience. Almost every scenario is tied to a specific system on the boat. Scenarios 1 and 8 are tied to the power systems consisting of the solar panels and batteries. Scenarios 4, 5, and 7 are related to the electrical hardware connections but they also can result from failure of the Engineer module of our software. Scenario 2 is tied the reliability of the software system as a whole. 3 and 9 are tied to the hull design of our boat.

Our risk assessment was performed by imagining potential failure scenarios given the nature of the project and our preliminary design. The failure scenarios are listed below and the number corresponding to each scenario is placed in a chart shown in Fig. 5 that compares the likelihood of that scenario without any mitigation and the severity of its consequences.

1) Total power loss
2) Software hang
3) Boat gets beached
4) Loss of propulsion
5) Loss of steering
6) Picked up by other people
7) Sensor failure
8) Electrical short due to salt water
9) High wind flips over watercraft

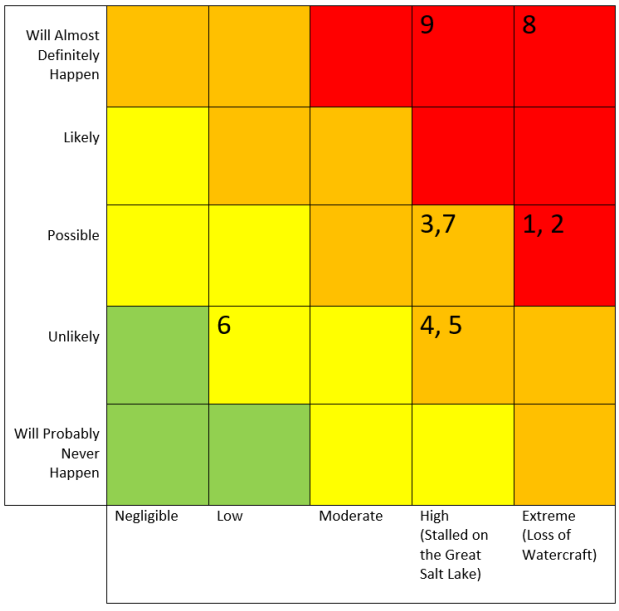

Fig. 5. Risk Assessment Chart

2) Mitigation Strategies: We came up with a few strategies to mitigate the risks listed above. Scenarios 1,2 and 8 are the most severe because they will result in a complete system failure and prevent us from recovering the watercraft since it will not be transmitting its location and will be adrift. To mitigate this scenario we are included a separate computer to act as a watchdog. It had its own battery and GPS and would periodically broadcast its location using a cellphone modem.

3,4,5, and 7 were mitigated by the remote communications systems on-board. While the boat is making its journey, we can monitor its progress and location. So, if something we're to happen that disabled its ability to navigate or move, we were able go retrieve it. Also, the watercraft was equipped with navigation lights in accordance with maritime law. This was done to aid us in locating the boat on the open water if the boat is unable to determine its own location.

To prevent scenario 9, we designed our boat to have a low center of mass. This was achieved with a heavy steel keel

Scenario 6 is very unlikely since our watercraft will be moving on a very sparse body of water. However, it is still possible. The boat has a durable label instructing passers by to not disturb the boat and it has our contact info should someone find the boat disabled on the water.

G. Group Management and Communications

To ensure that this project developed in a cohesive and timely manner we held weekly team meetings. We initially used Microsoft Teams, but due to difficulties in using this for both this project and professional employment use we switched to using Discord. Weekly meetings were in addition to times we scheduled to meet for the fabrication of the boat and software development. These meetings included discussions on individual team member's progress, any challenges they were currently facing, and information necessary for other team members to ensure that each endeavor was compatible. Discord was also used to post any new information or to ask questions related to development.

IV. Results and Final Analysis

A. Project Demonstration

For our project demonstration, we decided to carry out our final voyage on Utah lake instead of the Great Salt Lake. This was because of the shallowness around the edges of the Great Salt Lake, where we might not be able to reach our boat using our chase boat to retrieve it in the case of failures. Thus, on November 28, 2020 we took our boat shown in Fig. 6 to Utah Lake. We towed the autonomous boat behind our chase boat from the harbor towards the middle of the lake. Once there, we carried out some initial tests by having the boat follow a set of points away and back to us. We captured footage from both the on board camera recording in time lapse and remote

Fig. 6. Deomonstration of the autonomous boat at Utah Lake.

Fig. 7. Boat's autonomous path along Utah Lake.

footage from our personal cameras while following the boat. This footage was used in our video presentation of our boat, and we used this time to make sure that the boat functionalities are working as expected. Unfortunately, we soon discovered that our bilge pump stopped working due to a short circuit that happened the night before where we assessed the damaged parts but failed to test if the bilge pump was still operating. However, we managed to pump most of the water out manually by connecting the bilge pump straight to the batteries, and carried on with our voyage while testing the limitations of our water resistance. During the voyage the boat was able to travel autonomously for over 2 kilometers, with its path being updated live with APRS, which can be seen in Fig. 7. Finally, to finish off our voyage we set a path for our boat to follow which represents the letter 'U' as a remark for our university, but the on board GPS was damaged by the water and was unable to get a lock to report coordinates. This meant that our navigator was not able to carryout its functions, thus ending our voyage prematurely. While we learned a lot from our demonstration for the future, we clearly still have several issues to resolve.

B. Failure Analysis

In the end we were able to demonstrate the core functionality of our system and it was able to follow a course of waypoints autonomously. However, the final voyage on Utah Lake showed some critical issues in our hardware that we did not anticipate. Most significant was the failure of our GPS in water and our bilge pump. We had experienced problems with our GPS in the past but these were periodic and hard to replicate. So we assumed that it was not a hardware issue and could be resolved with software. This could have been handled better by performing a water sumbersion test with the GPS prior to the final voyage.

The bilge pump problem could have been addressed if we had tested the pump prior to the final voyage. We believe the bilge pump failure was cause by a wire shorting out on the heatsink for the transistor that controls the pump. In general, the wiring we used was solid core and difficult to work with. We might've been able to manage the wiring better if we had used more flexible wire.

There were also significant problems from RF interference from our main radio. There are three things we could do in the future to mitigate this problem. First, we could have

used shielded cable for all of our wiring. Second, we could've used a separate compartment for the radio transceiver. Third, we could place the antenna far away from the rest of the components of the boat.

V. Conclusion

Throughout this project we were able to create a fully autonomous, self sustainable watercraft. While there are more functionalities and improvements that can be integrated, we have learned many lessons for future progress. We are still determined to fix the issues brought forward during demonstration and testing, in order to enter our boat in the Microtransit challenge sometime in the future. Some of the issues that were overlooked by us during our planning and implementation process include: uninterrupted placement and water proofing for the GPS module, minimizing noise and signal interference from the APRS, and calculating the drift force vector in the Navigator to set the correct heading of the boat.

VI. Special Thanks

We had a little bit of outside help on this project. Isaac Doubek let us use his warehouse and tools to build our boat. In particular, he let us use his CNC machine and he designed and manufactured the waterproof box that holds our charge controller. Also, Diversified Metal Services in Salt Lake City cut the steel that made our keel. Additionally, John and Elizabeth Goodell made a financial contribution to the project. We are grateful to all of those who helped.

References

[1] N. P. Reddy, M. K. Zadeh, C. A. Thieme, R. Skjetne, A. J. Sørensen, S. A. Aanondsen, M. Breivik, and E. Eide, "Zero-Emission Autonomous Ferries for Urban Water Transport: Cheaper, cleaner alternative to bridges and manned vessels," *IEEE Electrification Magazine*, vol. 7, no. 4, pp. 32–45, Dec 2019.

[2] N. A. Cruz and J. C. Alves, "Autonomous sailboats: an emerging technology for ocean sampling and surveillance," in *OCEANS 2008*, Sep. 2008, pp. 1–6.

[3] Microtransat. The Microtransat Challenge. [Online]. Available: https://www.microtransat.org/

[4] ——. The Microtransat Challenge History. [Online]. Available: https://www.microtransat.org/history.php

[5] V. Shabaz. iobb library. [Online]. Available: https://github.com/shabaz123/iobb

[6] M. Tuupola. (2018, Feb) How to Calibrate a Magnetometer? [Online]. Available: https://appelsiini.net/2018/calibrate-magnetometer/