DATA ACQUISITION
MARINE VEHICLE
SEMI-AUTONOMOUS WATER TESTING AND DATALOGGING

Grecia Roman
Clinton Mazone
Danton Wyatt
Michael Genova
PROJECT OVERVIEW

• **Objective**
  • To provide water quality data using a semi-autonomous marine vehicle, reducing the time and resources required compared to traditional testing methods

• **Sensors and Data Logging**
  • Mike Genova

• **Navigation System**
  • Clinton Mazone

• **Marine Vehicle**
  • Grace Roman

• **Propulsion**
  • Danton Wyatt
DATALOGGER AND SENSORS

MIKE GENOVA
DATALOGGER MICROCONTROLLER – ESP32

- 240 Mhz dual-core processor
- Wi-Fi + Bluetooth built in
- 3.3v logic can be powered by a single 18650 Li-po
- Uses Arduino IDE
- Plenty of GPIO and comm ports
- Extensive libraries with good documentation
SENSOR SELECTION

- DS18B20 Waterproof Digital Temp Sensor
- 5v Analog Ph Sensor
- 5v Analog TDS sensor (Total Dissolved Solids)
- 5v Analog Turbidity Sensor
- BOSCH BME280 Digital Ambient Air Sensor
3D PRINTED BRACE

- A custom brace was needed to secure the sensors in such a way that would not interfere with the propulsion and navigation systems.
- Diameter of each sensor was measured and organized into a row from smallest to largest.
- Autodesk Fusion 360 software was used to create the bracket.
- The part was 3D printed at the CSUB FabLab.
DATA FLOW

Digital Data

Digital Sensors (temperature, Barometric pressure, humidity)

Analog Sensors (Ph, TDS, turbidity)

Diagram created with draw.io
LTE cellular module added for remote connectivity

Schematic Created with Fritzing Design Software
DATALOGGER CONTROL BOX REVISION 2
THE PROBLEM WITH IOT CELLULAR TECH

• Most available devices only support 2G connections (such as the Adafruit FONA808).
• The Adafruit 3G version is still under development and has a lot of bugs.
• While the hardware is still supported by its manufacturer, the cell towers they use are being decommissioned in the USA.
• I had purchased this module and a sim card that supported 2G networks. However I learned that in Bakersfield the last remaining 2G towers from T-Mobile have been shut down recently.
• More research needed to be done on cellular IoT connectivity.
• Rather expensive at $80, for something that isn’t practical.
• Cellmapper.net is a useful tool to check what cellular services are in your area.
• Originally, I had seen all the towers and assumed they were online, it was later that I found out that the towers were offline since October 2019.
• I could have used a different microcontroller from Arduino.CC that has LTE built in, but the code would have to be re-written and the control box would have to be re-wired due to the different GPIO pinouts. This would set us back significantly on our schedule. Also, Wi-Fi is not available on that module.
• Back to the drawing board we go, to look for another cellular solution...

Note: Tower offline
THE SOLUTION

• Newer 5G technologies exist for IoT platforms, mainly NB-IOT and CAT-M1.
• North America uses CAT-M1 technology which is supported by Verizon and AT&T.
• SIMCOM Technologies developed the SIM7000 LTE chip.
• Timothy Woo, a young engineer from Georgia, created a breakout board for the SIM7000 chip and rewrote the open source Adafruit library all by himself. The modified library works with both adafruit’s boards, and the botletics LTE board.
• The Botletics SIM7000A board is affordable and available to anyone for $65.
The library worked with the example code given but crashed when implemented into my main program. Clearly it needed some modification.

There are over 3200 lines of code in the library, and a single line of code somewhere was causing a crash. Arduino IDE does not have a built-in debugger, so an alternate method of troubleshooting was needed.

The ArduinoTrace library, created by programmer Benoit Blanchon was the best tool for debugging Arduino IDE. Source code can be found on Github. He is also the creator of one of the most used Arduino libraries to date - ArduinoJSON. (see below). The ArduinoJSON library was used in the Wi-Fi version of our datalogger to serialize data over Wi-Fi.

It was found that Timothy Woo’s library modifications did not account for large strings of data being returned from a server such as in our project.

Response from the server was truncated and caused the program to crash.

Debugging revealed that a small 500 ms delay was all that was needed. Thanks Benoit!
ENERGY SAVING TECHNIQUES – THE D24V22F5 REGULATOR

- Pololu makes high quality electronics for robotics that are made in the USA (Las Vegas).
- The enable pin allows the regulator output to be turned on and off via a microcontroller, drawing less than 5 microamps during standby.
- Produces a clean reference voltage of exactly 5.00 VDC needed for the sensors.
- It was found that other regulators produced an output voltage of 5.2-5.7 VDC in order to eliminate a brown out scenario when the regulator is delivering high current loads. This is great for a microcontroller supply, but not for a sensor VCC reference. The higher VCC voltage causes sensor error.
- Small enough to fit anywhere.
- Low cost - $5.00

With all the bugs finally worked out, we could finally begin analyzing energy consumption.

Due to the process of multiple states and multiple components, all with different current consumption, estimating battery life and battery capacity requirements can be difficult.

Average current is very difficult to determine in this scenario.

The Adafruit article written by Mike Stone on energy budgeting proved very helpful.

Measure voltage drops, current, and time for each component and convert to Joules.

This makes estimating battery requirements easy.
With the energy budget technique, I estimate 19 days of use between charging. This is achieved with only two 3.7v 18650 cell batteries in parallel.
There is too much code to show all of it, but here are some interesting pieces.

- Sensor data is converted to a string and stored in a buffer.
- The buffer is sent to the cloud via LTE with the fona.postData command.
- Coding for datalogger began in December 2019.
- Documentation by Benoit Blanchon was the major contributing factor to success in the datalogger portion of the project. Without his example code we would be far behind schedule.

- 562 lines of main Arduino code
- 3300 lines in the library
- After upload, the system goes to sleep.

```cpp
dtostrf(h20_temp, 6, 2, h20TempBuff);
dtostrf(h20_ph, 1, 2, phBuf);
dtostrf(h20_tds, 1, 2, tdsBuf);
dtostrf(h20_turbidity, 1, 0, turbBuf);
dtostrf(relative_humidity, 1, 2, rhBuf);
dtostrf(baro_pressure, 1, 2, baroBuf);
dtostrf(bat_voltage, 1, 2, battBuff);
dtostrf(latitude, 1, 6, latBuff);
dtostrf(longitude, 1, 6, longBuff);
dtostrf(altitude, 1, 1, altBuff);
```
• Sample screenshots of the datalogger information stored in the Adafruit.io IoT cloud server
• Unfortunately, the PH sensor is rather erratic, and the turbidity sensor was damaged by water infiltration.
• Next page has the link to live data
Current datalogging session was started with a full battery charge 4/24/2020 at 9pm.

https://io.adafruit.com/mgenova79/dashboards/damv
## MATERIALS FOR DATALOGGER (old vs new)

<table>
<thead>
<tr>
<th>MATERIALS</th>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP32</td>
<td>$20</td>
<td>$20</td>
</tr>
<tr>
<td>Ph sensor</td>
<td>$50</td>
<td>$50</td>
</tr>
<tr>
<td>TDS sensor</td>
<td>$12</td>
<td>$12</td>
</tr>
<tr>
<td>Turbidity sensor</td>
<td>$5</td>
<td>$5</td>
</tr>
<tr>
<td>temp sensor</td>
<td>$2</td>
<td>$2</td>
</tr>
<tr>
<td>BME280 ambient sensor</td>
<td>$8</td>
<td>$8</td>
</tr>
<tr>
<td>I2C ADC</td>
<td>$15</td>
<td>$15</td>
</tr>
<tr>
<td>level shifter</td>
<td>$4</td>
<td>$4</td>
</tr>
<tr>
<td>5v regulator</td>
<td>$8</td>
<td>$8</td>
</tr>
<tr>
<td>Li-po batteries</td>
<td>$20</td>
<td>$20</td>
</tr>
<tr>
<td>Enclosure</td>
<td>$15</td>
<td>$15</td>
</tr>
<tr>
<td>misc wire etc</td>
<td>$10</td>
<td>$10</td>
</tr>
<tr>
<td>Adafruit.io subscription</td>
<td>$10/month</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>$169</td>
<td>$234</td>
</tr>
</tbody>
</table>
NAVIGATION
CLINTON MAZONE
PROBLEM

• The vessel must reach target location(s) without active oversite or control by an operator.
• To solve this, we will design a system which will achieve the following:
  1. Take one or multiple sets of target coordinates from the user
  2. Calculate a heading based on current and target location
  3. Use an IMU to monitor the vessel's orientation
  4. Control a rudder and a motor which will steer and propel the vessel in the correct direction
THE GPS MODULE

• Communicates with satellites to determine:
  • Location of module in Longitude and Latitude
  • Other useful data which we won’t make much use of in this project.
• Uses serial communication to send data to other devices.
ORIENTATION MODULES

• Used in Aerospace and other applications which require orientation data
• Uses 3 Components:
  • 3-Axis Gyroscope
  • 3-Axis Accelerometer
  • 3-Axis Magnetometer
• Gathered data is processed by a “filter” to give us:
  • Yaw
  • Pitch
  • Roll
• This is essentially a digital compass
THE BNO-055 MODULE

• An orientation module designed by Bosch
• Uses a built-in microcontroller to perform the complicated calculations (i.e. the “filter”)
  • Also frees up the master device (the Arduino) to focus on other tasks
• Communicates via I²C protocol
THE ARDUINO MKR GSM 1400

- Cellular communication module
  - Communicates via the GSM network
  - Easy way to send commands (SMS or internet)
- 48MHz clock speed
- Two serial ports
  - Useful for communicating with the GPS and the PC at the same time
- Built-in LiPo Battery port
- Requires a LiPo battery to provide the high current consumed by the cellular module
COMPUTING TARGET
HEADING

\[ Y = \sin(\text{targetLon} - \text{currentLon}) \times \cos(\text{targetLat}) \]
\[ X = (\cos(\text{currentLat}) \times \sin(\text{targetLat})) \]
\[ - (\sin(\text{currentLat}) \times \cos(\text{targetLat})) \]
\[ \times \cos(\text{targetLon} - \text{currentLon}) \]
\[ \text{targetHeading} = \text{atan2}(Y, X) \]
## MATERIALS

<table>
<thead>
<tr>
<th>Component</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino MKR GSM 1400 w/antennae</td>
<td>$70</td>
</tr>
<tr>
<td>Adafruit Ultimate GPS</td>
<td>$40</td>
</tr>
<tr>
<td>Adafruit BNO055 Fusion Breakout</td>
<td>$35</td>
</tr>
<tr>
<td>Active GPS Antennae</td>
<td>$15</td>
</tr>
<tr>
<td>2500mAh LiPo Battery</td>
<td>$15</td>
</tr>
<tr>
<td>Waterproof box</td>
<td>$10</td>
</tr>
<tr>
<td>Various connectors and wires</td>
<td>~$30</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>$215</strong></td>
</tr>
</tbody>
</table>
**ARDUINO NAVIGATION SOFTWARE FLOWCHART**

- **Pre-setup:**
  - Declare global variables and waypoint array
  - Setup timers
  - Set magnetic declination for region
  - Add radial offset for compass, in case compass is pointing a different direction than the front of the boat.

- **Setup:**
  - Initialize GPS and IMU
  - Connect to cell network
ARDUINO NAVIGATION SOFTWARE FLOWCHART

• Main loop:
  • Constantly listens for new data from the GPS
  • Other functions are on a timer:
    • Every 10 seconds: Check SMS messages.
    • Every 2 seconds: Recalculate target bearing and distance.
    • Every 0.1 seconds: Check current compass heading and adjust rudder angle and motor speed if needed.
COMPUTING BEARING IN C++

Bearing required to reach target from current location can be given by:

\[ Y = \sin(\text{targetLon} - \text{currentLon}) \times \cos(\text{targetLat}) \]

\[ X = (\cos(\text{currentLat}) \times \sin(\text{targetLat})) - (\sin(\text{currentLat}) \times \cos(\text{targetLat}) \times \cos(\text{targetLon} - \text{currentLon})) \]

\[ \text{targetHeading} = \text{atan2}(Y, X) \]
DISTANCE CALCULATION IN C++

Distance from current location to target is given by:

\[
A = \sin^2\left(\frac{\text{targetLat}-\text{currentLat}}{2}\right) + \cos(\text{currentLat}) \cdot \cos(\text{targetLat}) \cdot \sin^2\left(\frac{\text{targetLon}-\text{currentLon}}{2}\right)
\]

Distance = \(2 \cdot \text{atan2}( \sqrt{A} , \sqrt{1 - A} ) \) \* radiusOfEarth

{ radius of the earth = 6,371,000 meters }

```cpp
/*
 * Calculates distance from current location to target location in meters.
 */

double getDistance(double targetLat, double targetLon, double currentLat, double currentLon)
{
    targetLat *= PI/180;  // convert from degrees to radians
    targetLon *= PI/180;
    currentLat *= PI/180;
    currentLon *= PI/180;

    double a = (pow(sin((targetLat-currentLat)/2),2) + (cos(currentLat)*cos(targetLat)*pow(sin((targetLon-currentLon)/2),2)));
    double c = 2*atan2(sqrt(a),sqrt(1-a));
    return (6371000 * c);  // radius of earth is 6,371 km
}
```
SMS COMMANDS

- SMS commands allow simple control over the navigation system.
- Currently available commands:
  - Add waypoint (simply send it any latitude and longitude coordinate)
  - GO ('G' or 'g')
  - STOP ('S' or 's')
  - CLEAR WAYPOINTS ('C' or 'c')
SMS COMMANDS

- SMS commands allow simple control over the navigation system
- Currently available commands:
  - Add waypoint (simply send it any latitude and longitude coordinate)
  - GO ('G' or 'g')
  - STOP ('S' or 's')
  - CLEAR WAYPOINTS ('C' or 'c')
SMS COMMANDS: ADD WAYPOINT

Text Message
Today 6:40 PM

35.342432, -119.101799

Waypoint added!

GPS and compass nav system basic test!
CONNECTED!
CURRENT WAYPOINTS:
{ 35.342432, -119.101799 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
{ 0.000000, 0.000000 }
Navigation on?: 0
SMS COMMANDS: GO

Waypoint added!

Navigation started!
SMS COMMANDS: STOP

Waypoint added!

Navigation started!

Navigation stopped!

Location in ddm.mm: 3520.1450, 119.056.7324
CURRENT LOCATION: 35.336084, -119.095540
Target location: 35.342432, -119.101799
Distance to target: 905.83 meters
Current heading: 193.0626
Target heading: 321.1757
Angle difference: 128.13
Calibration status: 10.9, 0.0
Yaw: 0.5125
Pitch: 0.7500
Roll: 2.3750

Location in ddm.mm: 3520.1453, 119.056.7324
CURRENT LOCATION: 35.336088, -119.095540
Target location: 35.342432, -119.101799
Distance to target: 905.48 meters
Current heading: 193.0626
Target heading: 321.1757
Angle difference: 128.11
Calibration status: 10.9, 0.0
Yaw: 0.5125
Pitch: 0.7500
Roll: 2.3750

CURRENT WAYPOINTS:
( 35.342432, -119.101799 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
( 0.000000, 0.000000 )
Navigation on?: 0
SMS COMMANDS:
CLEAR WAYPOINTS
NAVIGATION SYSTEM ENCLOSURE
NAVIGATION ENCLOSURE

- Waterproof on/off switch to disconnect battery
- Waterproof cellular antennae port
- Waterproof USB port for debugging and charging battery
- Cable gland to connect to propulsion box
MARINE VEHICLE PROTOTYPE
GRACE ROMAN
MARINE VEHICLE OVERVIEW

- Original Plan - Submersible Vehicle
  - This was the original plan until we discovered that it is much harder to send radio signals through water than air.
  - Drones are typically not operated by remote control and are completely autonomous, navigating with onboard computers and sensors. This idea was too intricate given our time frame for the project.
  - We did not have enough time for this type of implementation.

- New Plan - Floating Vehicle
  - Easier to build in given time frame.
  - Fastest solution to begin water testing.
## MATERIAL FOR MARINE VEHICLE

<table>
<thead>
<tr>
<th>Component</th>
<th>Qty</th>
<th>Cost/Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>4” PVC Pipe</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>5/16 Threaded Rod</td>
<td>2</td>
<td>$2.92</td>
</tr>
<tr>
<td>4” 90D Elbow</td>
<td>2</td>
<td>$4.98</td>
</tr>
<tr>
<td>36X3” Aluminum Angle Bar</td>
<td>2</td>
<td>$4.79</td>
</tr>
<tr>
<td>36X1” Aluminum Angle Bar</td>
<td>2</td>
<td>$5.96</td>
</tr>
<tr>
<td>36X1” Flat Aluminum Bar</td>
<td>2</td>
<td>$7.58</td>
</tr>
<tr>
<td>Misc</td>
<td>-</td>
<td>$38.50</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>$90.96</strong></td>
</tr>
</tbody>
</table>
BUOYANT FORCE

\[ F_{buoyant} = \rho g V_f \]

\( \rho = \text{fluid density} \)

\( g = 9.8 \frac{m}{s^2} \)

\( V_f = \text{volume of displaced fluid} \)

\( m_{object} \) (mass of the object)

\( \rho_{object} \) (density of the object)

\( \rho_{fluid} \) (density of the fluid)

Gravity

Buoyancy
BUOYANT FORCE

• Buoyancy calculations were essential for max load capacity.
• Calculation results: 36” X 4” diameter pipes required for a 20 lbs load.
• Once we had max load parameters this served as a guide for purchasing frame materials and electronic components.

\[ F_{buoyant} = \rho g V_f \]
• \( \rho = \text{fluid density} \)
• \( g = 9.8 \frac{m}{s^2} \)
• \( V_f = \text{volume of displaced fluid} \)
MARINE VEHICLE ASSEMBLY

• 4” X 36” PVC Pipe
• Flat Aluminum bars for frame
• Aluminum angle bars for mounting frame to PVC pipes.
MARINE VEHICLE ASSEMBLY

- 1” X 28” Aluminum bars for frame.
- 5/16 Threaded rod for frame stability and mount for motor.
MARINE VEHICLE ASSEMBLY

- Acrylic sheet for mounting all electronic components.
- Laser cut mounting guides for bolts.
FIRST PRESENTATION RESULTS

Future Updates

- Add 3D printed end caps and rudders.
- Switch from 90-degree elbows to 45-degree.
- Test on water
• Vehicle is now assembled for testing.
• Minor modifications are still required:
  • Such as rudders. Rudder length needs to be increased. Once rudders have appropriate dimension modification prototypes will be 3D printed using PLA plastic.
RUDDERS
RUDDERS

- Rudders were designed using Autodesk Fusion 360.
- The rudders are attached to a block that can fit a 5mm shaft that will be used for steering in conjunction with the navigation system.
UPDATED MARINE VEHICLE

Max Weight Capacity= 20lbs
UPDATED MARINE VEHICLE

• Marine Vehicle carrying 20lbs.
• Here we can see the marine vehicle is able to carry 20lbs load effortlessly.
• The water level reaches the half-way point of the PVC pipes as predicted by buoyancy calculations.
MARINE VEHICLE IN ACTION

Future updates are:

• the modification of rudders is required.
• Waterproofing marine vehicle.
• Testing in non-ideal conditions.
PROPULSION & STEERING

DANTON WYATT
PROPULSION

• Motor
  • Motor was completed
  • Made of electric motor and gearbox from old angle grinder
  • 3D printed propeller
MAKING OUR MOTOR

- Disassembled this angle grinder
MAKING OUR MOTOR

- Removed windings, weights, fan, and insulation
MAKING OUR MOTOR

- Removed windings, weights, fan, and insulation
MAKING OUR MOTOR

• Machined an additional 5mm shaft for attaching to motor coupling
MAKING OUR MOTOR

- Attached 3D printed fittings and covered drive shaft with pvc pipe
MAKING OUR MOTOR

• Brazed threaded rod onto nut to attach propeller to gearbox
MAKING OUR MOTOR

- Assembled with electric motor and attached to vehicle
CONTROLLING OUR MOTOR

• Will use data from navigation module to determine when to run motor
• Need to pulse width modulate
  • Will reduce current needed and extend battery life
  • Must use a motor driver
Motor Driver

- Looked into using L298N and DRV8871 motor drivers
  - Have around a 2V voltage drop
  - Motor would not start until around 60-70% duty cycle for PWM
  - Neither gave the performance we desired
CONTROLLING OUR MOTOR

Motor Driver

• Decided to use VN5019 motor driver
  • Has a negligible voltage drop
  • Motor will start even at duty cycles of 10% with PWM
  • Will allow us to conserve more power from batteries
  • Testing indicates that a 20% duty cycle will sufficiently move our craft
STEERING

Dual Rudder System

- Uses two 3D printed rudders (one on each pontoon)
- Controlled by a single stepper motor and belt system
- Uses an A4988 StepStick driver to control stepper motor
Considerations

• Rudders begin to lose steering efficiency around 35°
  • Need to limit travel of rudders
• Stepper motors have no default home position
  • Need to keep track of position of rudders
STEERING

Relevant code to turn left

• RudderPos variable will serve as an index for the position of the stepper motor

• Rudders will automatically adjust when the vehicle is turning toward its target heading

```c
int rudderPos = 0; //Global variable to track position of the rudders. Should remain in the range of -35 to 35.

if (degTurn < -35) { //Check to see if a turn more than 35 degrees to the left is required
    digitalWrite(dirPin, LEFT); //Set the direction of the stepper motor to turn rudders to the left
    for(rudderPos; rudderPos > degTurn; rudderPos--){ //Turn rudders to the left until they reach max number of steps (35)
        if (rudderPos == -35) break; //If the rudder is at the max turn break out of the loop
    } else { //If rudder is not at max turn step with stepper motor
        digitalWrite(stepPin, HIGH);
        delay(1);
        digitalWrite(stepPin, LOW);
        delay(1);
    }
} //rudderPos will be decremented after each iteration of loop

if (rudderPos < degTurn) { //Check to see if the rudder needs to start returning to position 0
    digitalWrite(dirPin, RIGHT); //Change direction of stepper motor to step back toward position 0
    for(rudderPos; rudderPos < degTurn/ rudderPos++){ //Step back toward position 0 if required turn is between -35 and 0 degrees
        if (rudderPos == 35) break; //Break out of loop if the rudder is at max turn position
    } else { //Step once with stepper motor
        digitalWrite(stepPin, HIGH);
        delay(1);
        digitalWrite(stepPin, LOW);
        delay(1);
    }
} //rudderPos will be incremented after each iteration of loop
```
STEERING

Relevant code to turn right

- RudderPos variable will serve as an index for the position of the stepper motor
- Rudders will automatically adjust when the vehicle is turning toward its target heading

```cpp
if(degTurn > 35) { // Check to see if a turn more than 35 degrees to the right is required
digitalWrite(dirPin, RIGHT); // Set direction of stepper motor to turn rudders to the right
for(rudderPos; rudderPos < degTurn; rudderPos++) { // Turn rudders to the left until they reach max number of steps (35)
  if(rudderPos == 35) break; // If the rudder is at max turn break out of the loop
else // If rudder is not at max turn then take a step with stepper motor
  digitalWrite(stepPin, HIGH);
delay(1);
digitalWrite(stepPin, LOW);
delay(1);
} // rudderPos will be incremented after each iteration of loop

if(rudderPos > degTurn) { // Check to see if the rudder needs to start returning to position 0
  digitalWrite(dirPin, LEFT); // Change direction of stepper motor to turn rudders back toward position 0
  for(rudderPos; rudderPos > degTurn; rudderPos--) { // Step back toward position 0 if required turn is between 0 and 35 degrees
    if(rudderPos == -35) break; // Break out of the loop if the rudder is at max turn
else // If rudder is not at max turn take a step
  digitalWrite(stepPin, HIGH);
delay(1);
digitalWrite(stepPin, LOW);
delay(1);
} // rudderPos will be decremented after each iteration of loop
```
STEERING

Future tasks

• Rudders need a way to calibrate and set home position when starting the vehicle
  • Mechanical switches
  • Optical sensors
  • Hall effect sensors
COMBINING HARDWARE AND NAVIGATION SOFTWARE
PROCESS OVERVIEW

• Take GPS and compass data
• Determine the required correction to reach destination
• Turn on motor and set speed
• Adjust stepper motor to control rudders to steer
• Turn off motor when destination is reached
CONTROLLING MOTOR AND SPEED

- Motor will turn on when a GPS waypoint is received
- Will run at full speed until 10m away
- Will run at half speed until 3m away
- Will stop when within 3m of GPS coordinate and then take readings

```c
if (distanceToTarget > 10) {
    motorSpeed = fullSpeed;
} else {
    motorSpeed = halfSpeed;
}
if (distanceToTarget < 3) {
    // Stop motor and get sensor readings
    motorSpeed = 0;
    analogWrite(motorControlPin, motorSpeed);
```
ISSUES FROM INITIAL TESTING

- Stepper motor would "bounce" back and forth when not adjusting rudders
- Rudders would easily turn one way but not the other
FIXING THE ISSUES

STEPPER MOTOR BOUNCE

• Assumed issue was software related
• Bouncing when on course trying to make minor corrections
• Solution was to create a 2° dead zone around target heading
• Stepper motor would be put into sleep mode when within 2° of target heading
• This prevent stepper motor from rapidly stepping back and forth
FIXING THE ISSUES

ASYMMETRIC RUDDER TURNING
• Assumed issue was software related
• Debugging showed software was functioning as intended
• Issue was related to the belt tensioners on the steering system
• Removed one tensioner and changed its position
• Ended up simplifying steering code as well
STEERING CODE CHANGE

BEFORE

• Stepper motor would make small adjustments if course correction was small

AFTER

• Stepper motor turns rudders to a set position regardless of size of course correction

```c
if(rudderPosition < correction){
    digitalWrite(stepDirection, LOW);
    for(rudderPosition; rudderPosition < correction; rudderPosition++){
        if(rudderPosition >= 35) break;
    }
    digitalWrite(stepEnable, HIGH);
    getCorrection(targetHeading, currentHeading);
}
```

```c
if(rudderPosition < newRudderPosition){
    incrementDirection = 1;
    digitalWrite(stepDirectionPin, LOW);
} else{
    incrementDirection = -1;
    digitalWrite(stepDirectionPin, HIGH);
}
for( rudderPosition != newRudderPosition; rudderPosition += incrementDirection){
    digitalWrite(stepEnablePin, HIGH);
    delay(stepDelay);
    digitalWrite(stepEnablePin, LOW);
    delay(stepDelay);
```
CONNECTING EVERYTHING TOGETHER

FINALIZING PROTOTYPE FOR TESTING
CONNECTING EVERYTHING TOGETHER

Arrows represent wiring harnesses that connect systems together:

- **Navigation Controller**
- **Motor control box**
- **Data-logger (independent of other systems)**
- **Sensor Array**
- **Stepper Motor (Rudder Control)**
- **Drive Motor**
• Our design is modular, so connections between control boxes are necessary.
• To simplify disassembly we used waterproof connectors such as the Delphi Metri-Pack 150, and other various connectors from Adafruit and home depot.
• The Delphi connectors are automotive grade and must be crimped and assembled by hand.
STEPPER AND DRIVE MOTOR CONNECTORS

• Ideal brand power plug connectors were used between the motor and the motor control box. They can handle high current and are well insulated
• The stepper motor requires 4 wires. The Adafruit weatherproof 4 wire connector was used to connect the stepper motor to the motor control box
MOUNTING CONTROL BOXES

- All control boxes are now wired up, mounted to the frame, and secured with industrial grade Velcro.
- The boxes can easily be removed and serviced by unplugging the connectors and separating the Velcro.
NAVIGATION TESTS

ALL MEMBERS
THE LAND TEST

• Points were chosen at various points at Truxtun Park
  • We sent the GPS points the boat via SMS, and started navigation
  • We then walked the boat towards whichever direction the rudders were attempting to steer the boat.

• Results:
  • Success!
  • Made a few tweaks to the rudder angles and the navigation sensitivity.
THE LAND TEST
LAKE TEST 1

• Points were chosen at various points at Truxtun Lake
  • We sent the GPS points the boat via SMS, and started navigation
  • We then walked the boat towards whichever direction the rudders were attempting to steer the boat.

• Results:
  • Motor speed setting would not propel the boat fast enough for the rudders to be effective. We increased the speed and started again in lake test 2.
LAKE TEST 2

- Coordinates were chosen at various points at Truxtun Lake
  - We sent the GPS points the boat via SMS, and started navigation
  - We then launched the boat and sent it the Go command ('G') via SMS to start the navigation program.
LAKE TEST 2 VIDEOS
LAKE TEST 2 RESULTS

- The navigation program seemed to work properly, and the motor speeds were sufficient.
- Interference from wind and currents prevented the boat from reaching its destination and pushed it into some plants where it got tangled. Modifications to the boat frame will need to be made.
LAKE TEST 2 VIDEO LINKS

- HTTPS://YOUTUBE.COM/AER44KZLQV (FROM DRONE)
- HTTPS://YOUTUBE.COM/_QWNF4K1MIC (FROM BOAT)

FULL PROJECT PLAYLIST:
HTTPS://WWW.YOUTUBE.COM/PLAYLIST?LIST=PLKAWA3EJRKQ59IKXWH1VIJO3MERQEU8-J
COVID-19 IMPACT

- Most of the project had been completed before the shelter in place order, so sufficient testing came to a standstill.
- Since the project was completely assembled, the remainder of the work was mostly code modification.
- Alternate methods were used to discuss improvements to the project.
- Coding was something that could be done at home. What we did was upload to OneDrive and another team member would upload the file to the microcontroller to test.
- Webcam and Zoom were used to show the effects of code changes to other members of the group.
MECHANICAL ISSUES
AFTER LAKE TEST

GRECIA ROMAN
SUGGESTIONS FOR IMPROVEMENTS TO BOAT FRAME

After testing on the lake, we discovered the following areas for improvement:

• Add fins to the front side of the boat for course stability and wind resistance.
• Increase size of the rudders for more turning power.
• Stronger braces to support steering belt tension, which would tweak rudder shaft angles.
PROBLEMS AND SOLUTIONS – WATERPROOFING THE GEARBOX

- Water was entering the gearbox and corroding the bearings ultimately causing bearing seizure.
- New improved sealed bearings were installed, which also failed, twice.
- Coating all mating surfaces with silicone sealant to prevent water penetration was required to ensure this problem would not happen again.
BONUS VIDEO: RESCUE MISSION