Development of an Autonomous, Fault-Tolerant Multi-Agent System for Surveying Alligator Populations in North Carolina

By Aidan Young

Advised by Dr. David Johnston

ABSTRACT

North Carolina is the uppermost recognized range for the American Alligator, but a warming climate may push their range further northward. This project aims to automate traditional alligator eyeshine surveys by using uncrewed surface vessels (USVs). Because the environments that alligators are found in can be hazardous, including the alligators themselves, I (1) developed a theoretical command-and-control framework to coordinate multiple USVs in a way that is resilient to individual malfunction and (2) constructed small, inexpensive, prototype USVs to implement this framework using Raspberry Pi computers and Arduino microcontrollers. The proposed framework addresses partitioning of complex water bodies & coordinating multiple USVs with unreliable communication networks to effectively survey the study area. Although motivated by alligator surveys, the project is generalizable to other species or environments.

INTRODUCTION

The American Alligator (*Alligator mississippiensis)* is not just a charismatic megafauna, but a keystone species in the ecosystems it resides in (Mazzotti et al., 2009). As a large predator, it regulates the populations of other animals via top-down control, in turn protecting the plants that form the basis of ecosystems. The deep holes that they dig, called gator holes, serve as water reservoirs for other animals and plants during droughts (*American alligator*). They reside mostly in freshwater and brackish water bodies like lakes, rivers, and marshes, but have been observed in coastal saltwater, despite not being physiologically adapted for it like crocodiles (*Alligator, American*).

North Carolina represents the northernmost boundary for the American Alligator in the continental U.S. Alligators are more common in southern North Carolina than northern, and their range appears to be dictated by temperature; alligators in North Carolina grow more slowly and reproduce less often than those further south (*Alligator, American*). The survival challenges that North Carolina alligators face is visible in their populations. A few thousand alligators are thought to reside in North Carolina, whereas over 100,000 are reported in South Carolina, just below North Carolina (Moore, 2024; *Alligator Hunting Season Report,* 2023).

As the effects of climate change influence global temperatures, warming year-round temperatures and, in particular, less harsh winters imply both greater survivability odds for alligators within their current range and also access to new areas that were previously uninhabitable. A study by Ryberg and Lawing, which modeled current alligator distributions as well as future distributions under various climate projections, substantiates this idea. They found that alligator ranges are expected to shift northward---alligators will move northward into North Carolina but also lose habitat in Florida due to prohibitively warm temperatures. Ryberg and

Lawing also caution that, although alligators as a species have survived climatic shifts throughout history, they did so by adjusting their ranges, not by adapting to these shifts. The time scale on which temperatures are expected to rise is orders of magnitude faster than in the past; it is unclear whether alligator populations will be able to respond in time to the temperature changes that are projected to occur in the upcoming decades (Ryberg & Lawing, 2018).

Improved alligator surveying and monitoring methods are thus necessary for both areas where alligator presence is expected to increase and those expected to decrease. North Carolina is especially relevant as a state with relatively little alligator presence up to the present day. Alligator management resides in a strange balance. Although alligators themselves are no longer a threatened species, having recovered from a history of hunting, they are federally classified as "threatened due to similarity of appearance" to the American crocodile (*Crocodylus acutus*) (*Endangered and Threatened…*, 2019). In North Carolina, however, very limited hunting is allowed by permit in specific counties as a means of population reduction (*Alligator, American*). These permits further necessitate proper surveying to ensure adequate and responsible population management.

There are many ways of surveying alligators, the foremost being aerial nest surveys and spotlight surveys. Aerial nest surveys are the best way of establishing an accurate population estimate, where the number of nests is a proxy for the number of alligators in an area. Spotlight surveys leverage the fact that alligator eyes reflect light, making alligators easy to detect at nighttime, equipped with a light source. Spotlight surveys serve to establish a minimum population; when alligator lengths are collected along with raw counts, the age structure of the population can also be estimated (Woodward & Moore; *Managing Alligator Populations*; Balkcom et al.).

Although aerial drones have been used recently to conduct aerial surveys of crocodilians, this project aims to create a boat-based system to automate typical spotlight surveys (Scarpa & Piña, 2019). Drones have limited use in areas with high vegetation coverage, which compose some, although not all, of alligator habitat. By using uncrewed surface vessels (USVs), there is the potential to survey areas inaccessible from above.

Alligators, however, are documented to attack drones, including USVs (Price, 2023). This project, in addition to developing a prototype USV for conducting these surveys autonomously, aims to create a protocol to coordinate multiple robots to conduct a survey at once. The main priorities of this multi-agent system are (1) parallelization and (2) redundancy. In ideal circumstances, multiple robots surveying at once will complete a given area quicker; this speed is relevant when considering that these surveys must practically occur at nighttime. If one or more USVs are destroyed, the remaining USVs can still complete the survey area.

The use of multiple drones or agents to complete surveys is not novel. There is plenty of interesting work on theoretically optimal ways to coordinate the movements of multiple agents at once. However, there is little work that addresses the reality of surveying in these adverse environments; it is not possible to solve for optimal paths in advance, because the destruction of a robot mid-survey would still result in a nonoptimal survey. Furthermore, it is also difficult to calculate optimal paths in real time, because a potentially disconnected communication network (due to destroyed robots or an environment that disrupts radio signals) means that, although possible, it is not guaranteed for a single USV or base station to know every other USV's position and other state variables, such as battery life, at a given time. This project thus aims to create a movement and communication protocol that, while not perfectly optimal, serves as a pragmatic basis for completing actual multi-agent surveys in hazardous environments.

Furthermore, the project recognizes that cost should also be a factor when creating USVs that are ultimately expendable. Because there is no guarantee for a single robot to make it home and in the interest of still getting usable data from the area it can survey before failure, this project aims to embed as much computing as possible onto the USVs themselves to complete in real-time. In the age of cloud computing and AI driven by large language models, this project instead necessitates computationally inexpensive code, to both be run in real-time and on cheap single board computers such as the Raspberry Pi.

METHODS & MATERIALS

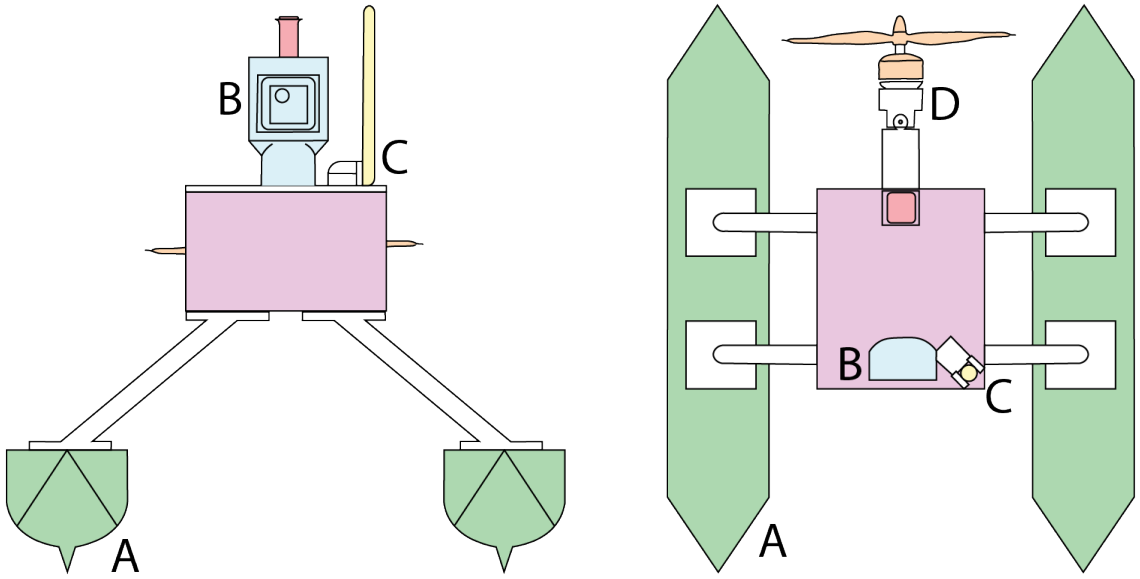
Component List

In addition to PLA filament, access to a 3D printer, and access to a soldering iron, the following components are used to construct the USV prototype:

- Raspberry Pi 4 Model B, 2GB RAM
- Adafruit Feather RP2040 with RFM95 LoRa Radio
- 915Mhz Antenna
- uFL to SMA connector
- Adafruit Triple-axis Magnetometer MMC5603
- Adafruit Ultimate GPS
- Raspberry Pi Camera 3 (Wide NOIR)
- Raspberry Pi Camera Cable
- Generic Brushless DC drone motor
- Electronic Speed Controller (ESC) compatible with the above motor
- Generic drone propeller compatible with the motor
- SG90 Servo motor
- Blomiky 9.6V 2000mAH NiMH battery
- Tamiya to bare wire connector
- UBEC Step-down converter (5V at 3A)

- OVONIC 3S LiPo Battery 5200mAh 11.1V
- Many multicolor jumper wires
- Electrical tape
- ½ lb. of aluminum pellets or other similar material
- 2 USB-A to USB-C adapters
- Flex Seal Liquid Rubber Sealant Coating
- 4-pin JST connector
- Singular skateboard bearing
- Superglue

USV Construction



The USV itself is a catamaran supported by two deep-v hulls (A). This design choice was motivated by the need for having an elevated camera (B) for imaging, and the catamaran construction allows for high stability and limited roll even if a large amount of the boat is out of the water. Not only is the camera high above the water, but the choice to put the propeller above water means even more of the USV will be above the water and further motivates the stability of the catamaran construction. The choice to use an air propeller instead of a water propeller is

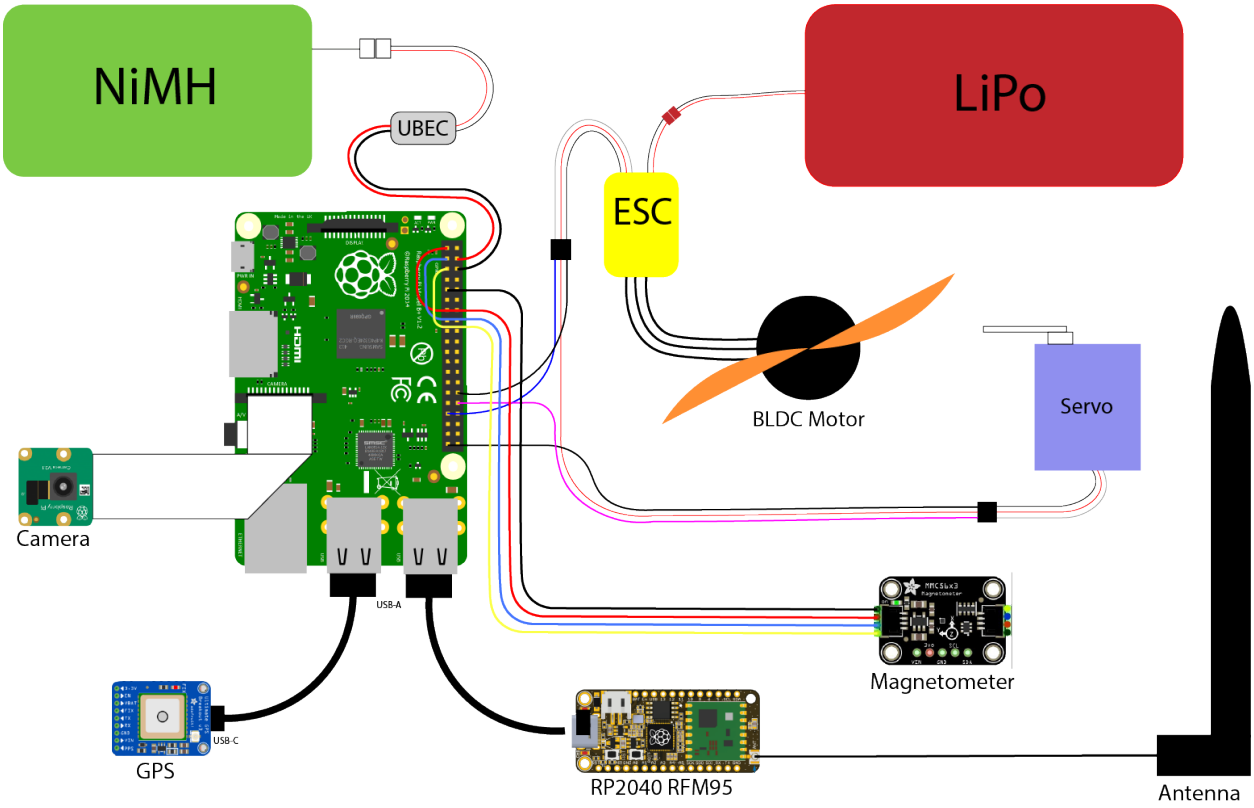
manifold: standard drone motors are cheap and readily available, less exposure to water extends their lifetimes, and having the motor above the water prevents it from becoming fouled with underwater plants or the propeller hitting the bottom when in shallow areas. The motor is mounted onto a piece that is glued to a servo on top and pressed into a ball bearing on the bottom. The servo motor controls the angle of this piece and thus the angle of the motor. The combined servo and motor allow the boat to be able to both move and control direction.

The frame for the USV was modeled using the online software TinkerCAD; although not the most advanced software, it allowed to rapid iteration. The models for the motor mount and the camera mount in specific were borrowed from pre-existing, open source models by Thingiverse user @BuildSomeStuff and Printables user @DocWeebl, respectively (*Raspberry pi camera…*; Thingiverse.com, *RC airboat by buildsomestuff*).

The model is then 3D printed piece by piece using PLA filament. Aside from attaching the moving arm of the servo and the BLDC motor, every part can be attached using superglue or a similar glue.

The motor and batteries being in the main box above the water raises the center of mass, so approximately a ¼ pound of aluminum pellets are added to each hull to lower the center of mass closer to the waterline. Because 3D printed PLA is watertight, Flex Seal Liquid Rubber Sealant Coating is sprayed thoroughly onto the fully constructed hull to ensure the submerged pieces of the USV remain watertight. Although the boat lacks a true keel, the ridge at the very bottom of the hull, which stretches across the entire hull, serves the similar purpose of preventing drifting and keeping the USV on course.

Electronic Wiring



The Raspberry Pi serves as the flight controller and as the onboard computer, reading values and interpreting values from the GPS and magnetometer, receiving and sending packets via serial with the RP2040-driven LoRa radio, and taking and analyzing images using the camera. The power supply for the motors are separate from the rest of the electronics. This is because the BLDC motor requires lots of power and at sporadic intervals, particularly when the thrust being set by the Raspberry Pi changes drastically over short periods. In contrast, the rest of the electrical components require less power but much more steadily. The LiPo battery thus powers the motors whereas a less powerful but reliable NiMH battery powers the Raspberry Pi and all the components it connects to. The wiring of all the grounds are interchangeable with other grounds, but since the magnetometer uses I2C and the motors use PWM, that wiring is not.

A script publically available on GitHub by user nliaudat was used to calibrate the magnetometer to correct for hard and soft iron errors (nliaudat, *Ellipsoid fitting using python numpy to calibrate magnetometers*).

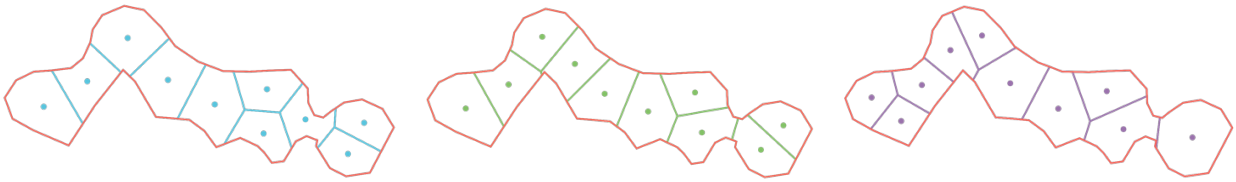
Preprocessing Algorithms

The Duke Reclamation Pond traced into a simpler polygon

Survey water bodies can be traced into two-dimensional polygons for use in the preprocessing steps.

Lloyd's algorithm, which is a clustering algorithm that is also used for creating partitions, is applied with a target number of n partitions passed in as a parameter. This algorithm functions by iteratively calculating the Voronoi diagram for a given set of points, shifting each point towards the centroid of its respective Voronoi cell, and recalculating the Voronoi diagram for the newly positioned points. The number of iterations is also passed in as a parameter to Lloyd's algorithm. In practice, $i=100$ was used; greater numbers did not result in much change. Because the Voronoi diagram creates a set of cells, where each cell c_i represents the portion of the overall

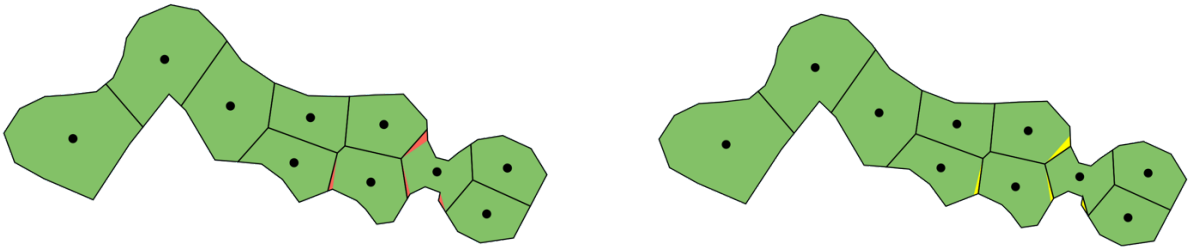
space that is closest to a given point p_i, an emergent property is that the polygons are consequently of roughly even sizing and typically convex (Bozkaya et al., 2023). In practice, this can be done using Python's `shapely` package, particularly the `voronoi_diagram` method (*Shapely*).



Three different outputs of Lloyd's Algorithm with the same starting conditions (10 points, 100 iterations)

Although Lloyd's algorithm can create a valid partition, it is not guaranteed to, because a desired property of each of the cells in the partition being "strongly-visible." Strong-visibility is when, for a cell c_i, there is at least one point that can "see" (have an unobstructed, direct path to) the entirety of the permiter of c_i. By extension, this means that this point can "see" every other point in the polygon. For this purpose, it is important that the centroid of the cell is one of the points that is strongly visible. This is relevant for the pathfinding logic discussed later.

Strong-visibility can be guaranteed by, for each cell, calculating which parts of the cell the centroid cannot "see." If the centroid can see everything, then there is no change needed. Else, for each region, the centroid cannot see, as there might be multiple, attempt to merge them into an adjacent cell, only actually merging if that region can be added while keeping the adjacent cell strongly-visible still. If this is not possible for any adjacent cell, the region is turned into its own new cell.

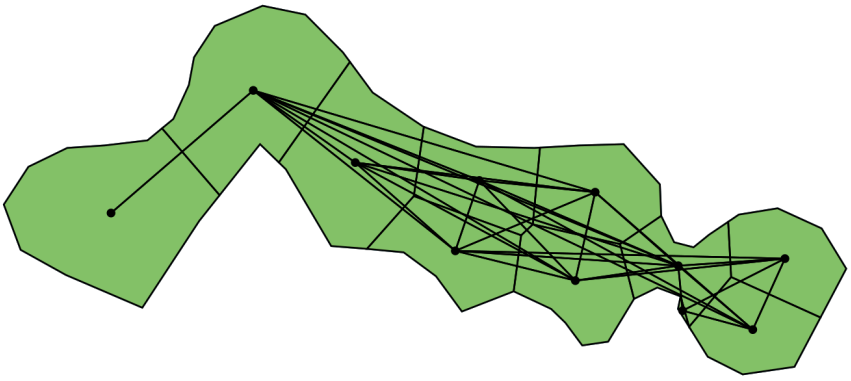


These two figures display the nonvisible portions of cells in red (left) and the when they are merged with other cells or split into their own (right)

The centroids of the final output of Lloyd's algorithm can be passed further into the next step, which is to construct a navigation mesh for the robots to be able to easier pathfind around the survey region. This algorithm simply takes every possible pair of centroids and, for each pair, creates a line between those two points and checks to see whether this line is contained within the overall survey polygon or not. If the line is contained, then it means that there is an unobstructed path between the two points and this path should be included in the navigation mesh, with the length of the line. If not, then the path is obstructed and so is not a valid path to be included. This step is also done using the `shapely` package and, optionally, the `buffer` method can be used to shrink in order to not count paths that might geometrically be valid but, in practice, go unacceptably close to the shoreline, as there might be concerns with depth or other obstacles too close to shore. The output of this algorithm can be used to create a weighted, undirected graph. It is not, however, guaranteed to create a connected graph, wherein every node is reachable from every other node, which is required to create a usable navigation mesh.

Thus, an additional step is required. For each pair of cells in the overall diagram, first check to see whether they are adjacent (this can be done in `shapely` by taking the union of the two cells and checking to see whether it is type `Polygon` [adjacent] or `MultiPolygon` [not adjacent]). If they are adjacent, then find the midpoint of the edge that they share and connect their centroids via that midpoint. A simply pseudocode for the above algorithm, where :

```
constructNavigationMesh(area, centroids, cells):
    let n = size of list centroids
    create an empty set of edges E
    create an empty set of nodes V
    add each centroid in centroids to V as new nodes
    for i in 0 to n
        for j in i+1 to n
            path = line between centroids[i] and centroids[j]
            if area contains path
                dist = euclidean distance of path
                add path as an edge to E with a weight of dist
            else if cells[i] and cells[j] are adjacent
                shared_edge = intersection of cells[i] and cells[j]
                midpoint = midpoint of line shared_edge
                add midpoint to V as a new node
                dist_i = euclidean distance between centroids[i] and midpoint
                dist_j = euclidean distance between centroids[j] and midpoint
                add edge btwn centroids[i] and midpoint to E w/ weight dist_i
                add edge btwn centroids[j] and midpoint to E w/ weight dist_j
            end if
        end for
    end for
    return V, E
```



Full navigation mesh

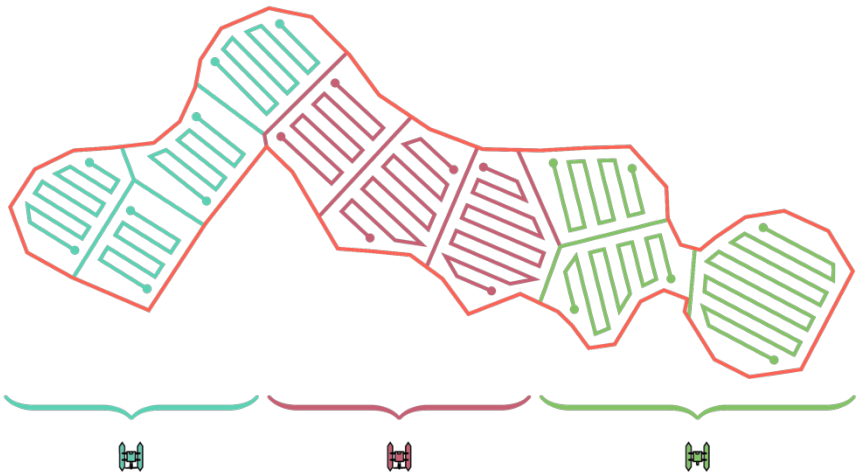
The graph output from this algorithm is suited for Djiktra's Algorithm, which finds the optimal path between two nodes in a weighted, undirected graph (Huang et al., 2009). Although more complicated algorithms such as A* could be applied, the number of nodes and edges is, in a real-world survey, unlikely to be large enough such that the performance boost is worth the added complexity. The importance of the strong-visibility can now be appreciated. Because an unobstructed, direct path is guaranteed between the centroid of cell and any other point within that cell and there is also a guaranteed path, reasonably efficient path between the centroids of every cell, then there is a guaranteed unobstructed path between every point in the entire survey area. A path between a random $point_i$ in $cell_i$ and a random $point_j$ in $cell_j$ is thus:

$$point_i \rightarrow centroid_i \rightarrow path\ between\ centroid_i\ and\ centroid_j \rightarrow centroid_j \rightarrow point_j$$

Survey paths for each cell can be calculated algorithmically by taking shrinking in every direction by buffer amount *b* (amount, not percentage). Then, longest edge of each cell is found. The cell is rotated such that this longest edge is horizontal, and then $\left\lfloor \frac{width\ of\ rotated\ cell}{\mathrm{b}} \right\rfloor$ transect lines are drawn vertically, or perpendicular to the original longest edge, at *b* intervals. The transect cells are then connected in a lawnmower pattern.

The final step of preprocessing is to split the overall partitions into *u* groups, where *u* is the number of USVs available to conduct the survey. Currently, this is done using the Fluid Communities algorithm implemented by the NetworkX `asyn_fluidc` method. This algorithm is favored because it takes the number of target groups (or communities) as a parameter, unlike other community-finding algorithms where the number of output communities is not something that can be specified. Each USV is assigned as the "leader" of one of the communities at random.

Community-finding algorithms are used so that, ideally, the elements in a given group are all close together, meaning that, once the leader of that grouping has completed a survey of one of the elements in its group, it will be very close to next element to do a survey of.



Cells split evenly among three USVs

Mesh Radio & Communication Protocol

The RP2040 RFM95 runs on a script written in C++ (*Getting Started with Arduino*). The code heavily leverages the RadioHead Packet Radio library by AirSpayce. By using specifically the `RHMesh` class, the USV system can form a mesh network. The mesh network means that

packets can be sent across the radios of multiple intermediary USV to finally reach the destination, even if the sender and receiver are themselves out of range. Furthermore, the route between the sender and receiver is automatically discovered and, if the network changes the route is no longer valid, then a new route can be discovered (*Radiohead: ...*). This packet radio can achieve long distance of multiple kilometers, but with the tradeoff being very small maximum packet sizes of 252 bytes (*Adafruit RFM69HCW and RFM9X Lora Packet Radio ...*). These radios can therefore not feasibly transmit audio or video, but can be used to send latitude, longitude, heading, and other information about a USV's current state, as well as request instructions or respond to requests.

When a USV sends a broadcast out, one of the variables it includes in the packet is its current tally of surveyed partitions, as list `completed` where `completed[i] = True` means that $cell_i$ has been surveyed and `completed[i] = False` means it has not yet been surveyed. When another USV receives this broadcast, it compares its list with the list it receives. If there are any cells in the received list that are marked as `True` which the USV has as `False`, it sets its value to `True`, which is just an OR operation. This is a kind of gossip protocol, whereby the peer-to-peer communication of information leads to a more unified agreement of the state of a system (GeeksforGeeks, 2024).

The leader designation algorithm described prior is critical to the communication and coordination protocol. A USV surveys all the partitions it is a leader of without needing to communicate with other robots, although it listens to other USV's broadcasts that it is in range of to keep track of the current survey state. When there are no more cells in the group a USV is the leader of, it moves to an adjacent group area and sends a request to the leader of that group to survey a polygon. The leader responds affirmative if the polygon has polygon has not been

surveyed along with an authority number. If there is no response at all, the requesting USV assumes that something happened with the original leader, and assumes leadership of the group, but with a lower authority number than the original leader. This way, if a third USV requests to survey the group and the original leader comes back in range or online, the third USV will receive two responses, one from the original leader and one from the USV that assumed new leadership. But, the third USV can tiebreak the potentially conflicting instructions based on the authority number associated with each response.

Control Loop

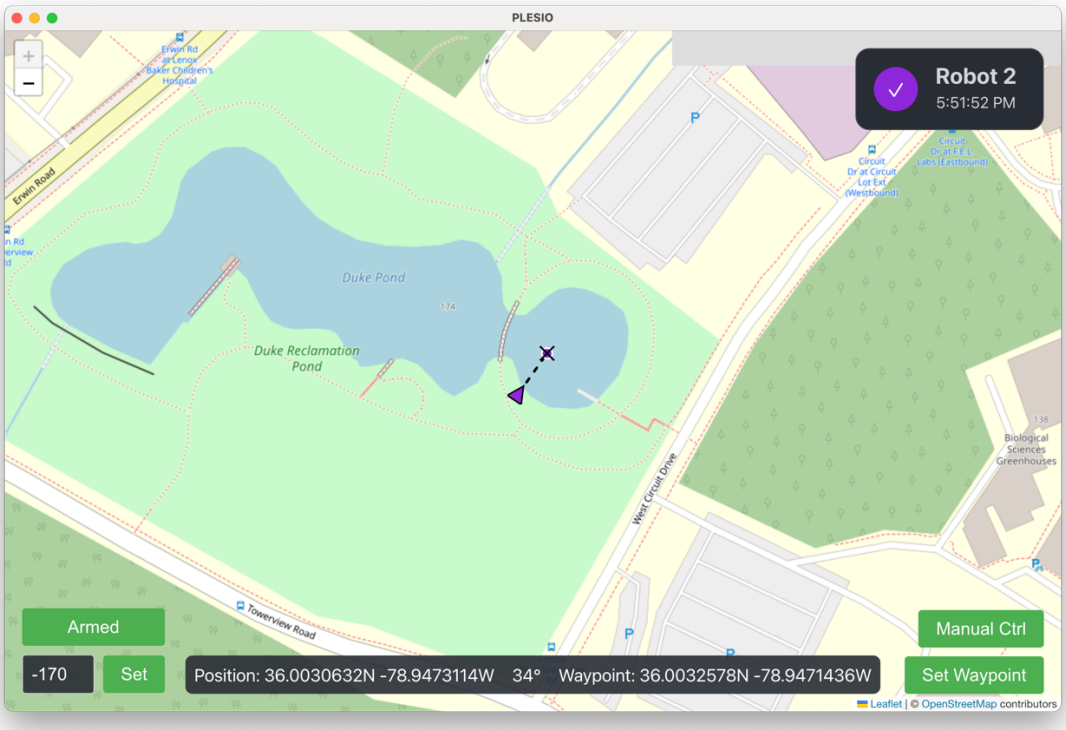
The main script that the Raspberry Pi runs is written in Python. The autonomous pathfinding logic of the Raspberry Pi, at its core, is a simple loop in which, the script first checks the radio for any incoming messages and parses them. The script reads the current GPS and magnetometer values. It checks to see whether the USV is within an acceptable threshold distance of the current waypoint. If it is, then it considers the waypoint as having been reached and sets the current waypoint to the next waypoint in the queue. The script then sends any messages as well, either a response or to broadcast its status if enough time since the last broadcast has passed. The script then checks the arm state to determine whether the motors should be running. If the USV is armed, the script calculates the heading error, which is difference between the current heading and the actual angle between the USV's position and the next waypoint. It maps this error to a servo angle, to turn the boat towards the necessary heading. The servo angle is then mapped to a thrust value, where the straighter the motor, the stronger the thrust. The more extreme the turn, the weaker the thrust, to prevent rapid movement and overcorrection. Although thrust PWM values range between 1000-2000 microseconds (um), the

code sets the max thrust to be 1300um, because larger values caused the propeller to fly off the motor.

Ground Control Software

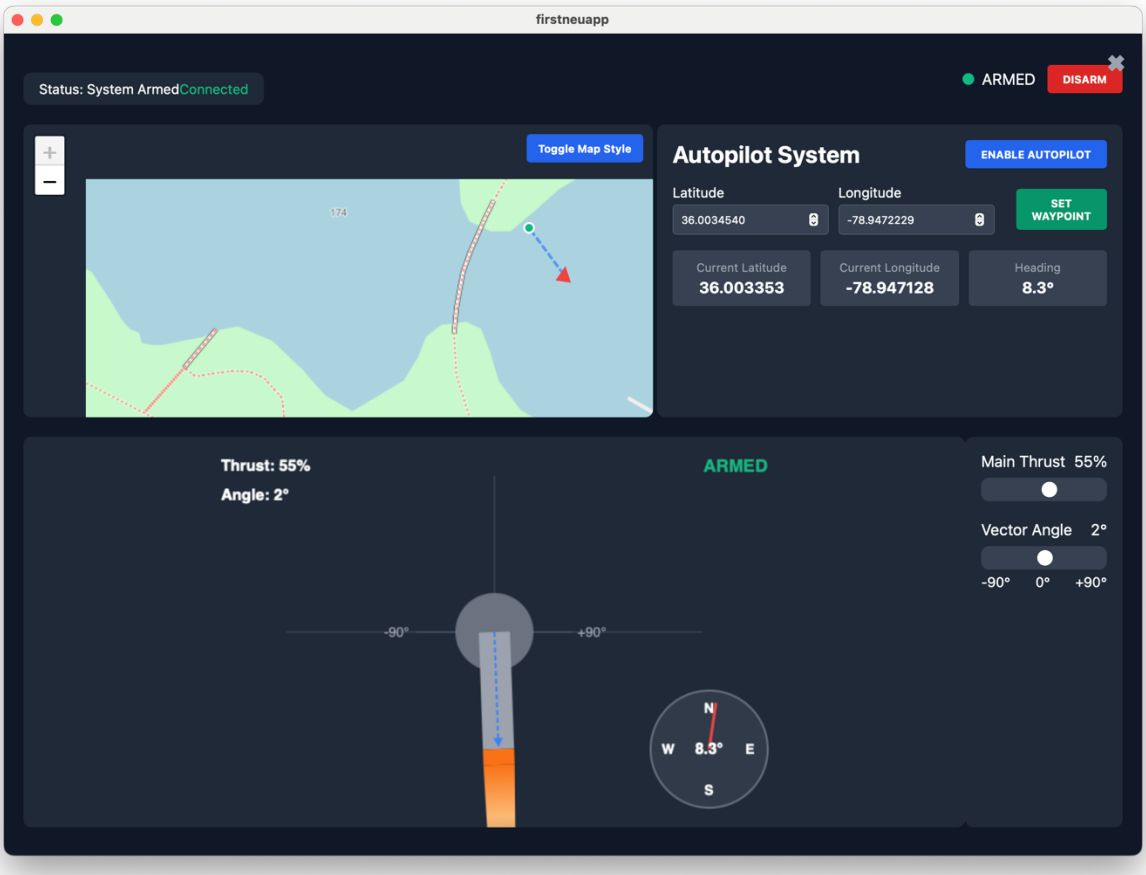
Although pre-existing softwares exist to control USVs and other autonomous vehicles, such as QGroundControl and ArduPilot, they rely on the MavLink communication software (*QGC*; *ArduPilot*).

Furthermore, neither software supports monitoring of multiple autonomous vehicles at once. To more flexibly communicate with the Raspberry Pi, two softwares were written to be lightweight, ground control stations for interfacing with the USV. One was written using React and Electron, and sends and receives information via a LoRa packet radio, which must be plugged into the computer running the software. In order to prevent clogging the network and dropping packets, USVs communicating via LoRa broadcast their status every ten seconds. This information is then displayed on the software.



LoRa-based custom ground control software

A second software was also written using Neutralino and vanilla Javascript to create a second application to both set autonomous waypoints, as well as control the USV manually, by sending data over a WiFi network using HTTP requests to a Python Flask server endpoint running on the USV's Raspberry Pi. In practice, the WiFi network can be an access point hosted by the Raspberry Pi itself. The much higher bandwidth of WiFi allows data to be sent much more frequently, which makes manual control feasible and is useful for debugging, as the USV state can be observed in real-time.



Wi-Fi-based ground control software

RESULTS

The USV handles well in the water. When the servo is rotated a full 90 degrees in either direction, the USV is able to rotate itself in place. At the most powerful thrust setting, the prow of both hulls began to tilt downward and go beneath the water. In testing, this did not cause the USV to flip or tip over, but there may still be a risk of that occurring if the boat were to go faster. Aside from this issue, the single drone motor is very capable of providing enough thrust to move at an adequate survey speed of 1-2 meters per second.

The USV is successfully able to drive itself autonomously via LoRa-sent instructions as well as drive manually using instructions sent over WiFi. Both programs are also able to monitor the USV's position and log progress, although the WiFi-based one is much more useful for debugging and monitoring due to the much more frequent update rate. The WiFi range of the access point hosted by the Raspberry Pi is quite weak and cannot support a mesh network, which only makes it useful for testing single USVs fairly close to shore.

The magnetometer, when properly calibrated, also appears to function well and accurately. The GPS, however, can initially place the USV well but, when the USV begins moving, does not accurately update to the USV's current position. This makes it difficult to further evaluate the ability of the USV prototype to conduct missions with multiple waypoints. In contrast to the LoRa antenna and the magnetometer, which are both isolated from the rest of the electronics, the GPS sits within the main box along with the rest of the electrical components.

Augmenting Lloyd's algorithm to create strongly-visible cells is a tradeoff, in which the equality of polygon size is deprioritized in favor of strong-visibility. This is acceptable because there is no intrinsic reason for each cell to be of similar area. In contrast, it is important for a USV to be able to move between every point in a survey region.

DISCUSSION & NEXT STEPS

The use of drones and other robots for studying animals and conducting surveys is on the rise. Aerial drones have seen a wide variety of usages for terrestrial and aquatic animals, and are often faster, less invasive, and sometimes cheaper than equivalent, alternative methods (Pedrazzi et al., 2025). In contrast, USVs have seen less practical use up to this point for wildlife surveys, because the angles provided by cameras are at a less useful angle for detecting obstacles and objects, although in the case of eyeshine surveys the low angle is more practical than those provided by drones (Huang et al., 2023). USVs also have the potential to enter environments impassible to aerial drones and, with weight being less of a constraint, can afford larger batteries and thus extended survey time compared to aerial drones.

Because the use case of USVs lie in more complex areas where drone flight is difficult, they will operate in areas where data transmission is less guaranteed. Autonomy is thus more important with these surveys, as is redundancy. This system serves as the basis for a computationally lightweight, cost-effective, and relatively easy-to-implement way to conduct autonomous surveys. Although the project was initially motivated by surveying alligators, the core of the project ultimately lies in the pragmatic communication and movement protocols, whose applications extend far beyond alligator surveys, and even the USV platform itself.

This project is not complete, and there are many potential next steps to further improve it. Overall, there was limited opportunity to test and validate the functionality of the USV. The lack of alligators in Durham made it impossible to take any pictures of alligators in the field, so it was not feasible to create an eye detection script. Making more USVs to test whether multiple can practically communicate with each other is another aspect to test.

On the physical side of the project, an external and active antenna should be added to the GPS to get better GPS data or the GPS should be upgraded outright. To allow the USV to move efficiently at high speeds, more ballast should be added to the hulls and the hulls should be made larger to support the added weight of the ballast.

On the preprocessing side, the current method of ensuring strongly-visible cells is fairly naïve and does not produce particularly elegant partitions. There is a need for a better way to guarantee strongly-visible survey cells while still being able to survey as much of the overall area as possible. There has been pre-existing work on a similar topic that adds a different step after Lloyd's algorithm to partition a convex polygon into many convex polygons, but in this case the input is not guaranteed to be a convex polygon, so more work is needed (Campillo et al., 2024). Another potential approach might could use a totally separate partitioning method, such as Delaunay triangulations, but it is unclear whether such methods would result in polygons suitable for creating transect surveys within.

For the Raspberry Pi control loop, the current method for calculating the angle and thrust values, while functional, could be improved. A PID-type controller that accounts for more factors aside just heading error could allow the USV to more accurately follow the path, especially when facing adverse conditions like current and wind.

The custom ground control software, while functional, has a lot of opportunity for further development. Specifically, there is a need to a ground control software that is able to control multiple surveying drones at once. This wireframe, while a useful way to monitor the prototype, could give more information than current location, next waypoint, and heading, such as overall path for each robot, as well as information about the overall mesh network topology. Furthermore, more commands could be issued to robots, such as return to launch or hold a

location. The preprocessing steps currently done via a Python script could be incorporated into the ground control software, using the software as a GUI for the preprocessing steps.

CONCLUSION

This project further represents many different aspects combined into one basic, yet functional, system. Each aspect has been carefully considered to be as inexpensive as possible while still preserving the core, required function. Each aspect remains distinct from the others and so can be improved or modified to serve a use case without requiring an overhaul of the entire system. The preprocessing algorithms can be expanded upon or redone without needing to change the radio, and the materials or layout of the frame can be upgraded without needing to change the code or electronics. In situations where cost is a barrier to entry for technologies, this project presents a low-cost robot with an easily acquired parts list. Furthermore, the redundancy aspect of this project, which is not typically seen in the context of wildlife surveys, opens the door for applying more autonomous surveys in more complex and hostile areas, where drone failure, while not optimal, does not compromise and survey and can be gracefully handled.

REFERENCES

ArduPilot. (n.d.). *Ardupilot*. ArduPilot.org. https://ardupilot.org/

Balkcom, G., Menken, T., Johannsen, T., Parnell, I. B., Howze, B., Horan, R., & Waters, G. (n.d.). Georgia's Alligator Management Plan (2021-2030). https://georgiawildlife.com/sites/default/files/wrd/pdf/management/2010_Alligator_Management_Plan.pdf

Bozkaya, E., Karatas, M., & Eriskin, L. (2023). Heterogeneous wireless sensor networks: Deployment Strategies and coverage models. *Comprehensive Guide to Heterogeneous Networks*, 1–32. https://doi.org/10.1016/b978-0-323-90527-5.00009-5

Campillo, M., González-Lima, M. D., & Uribe, B. (2024). An algorithmic approach to convex fair partitions of convex polygons. *MethodsX*, *12*, 102530. https://doi.org/10.1016/j.mex.2023.102530

Endangered and Threatened Wildlife and Plants; Regulations Pertaining to the American Alligator (Alligator mississippiensis). Federal Register. (2019, January 19). https://www.federalregister.gov/documents/2021/01/19/2021-01012/endangered-and-threatened-wildlife-and-plants-regulations-pertaining-to-the-american-alligator

GeeksforGeeks. (2024, May 29). *Gossip protocol in disrtibuted systems*. https://www.geeksforgeeks.org/gossip-protocol-in-disrtibuted-systems/

Getting Started with Arduino. Docs.arduino.cc. (n.d.). https://docs.arduino.cc/learn/starting-guide/getting-started-arduino/

Huang, C.-Y. (Ric), Lai, C.-Y., & Cheng, K.-T. (Tim). (2009). Fundamentals of algorithms. *Electronic Design Automation*, 173–234. https://doi.org/10.1016/b978-0-12-374364-0.50011-4

Huang, T., Chen, Z., Gao, W., Xue, Z., & Liu, Y. (2023). A USV-UAV cooperative trajectory planning algorithm with Hull Dynamic Constraints. *Sensors*, *23*(4), 1845. https://doi.org/10.3390/s23041845

lady ada. (n.d.). Adafruit RFM69HCW and RFM9X Lora Packet Radio ... https://cdn-learn.adafruit.com/downloads/pdf/adafruit-rfm69hcw-and-rfm96-rfm95-rfm98-lora-packet-padio-breakouts.pdf

Managing Alligator Populations. TPWD. (n.d.). https://tpwd.texas.gov/huntwild/wild/species/alligator/management/index.phtml

Mazzotti, F. J., Best, G. R., Brandt, L. A., Cherkiss, M. S., Jeffery, B. M., & Rice, K. G. (2009). Alligators and crocodiles as indicators for restoration of Everglades Ecosystems. *Ecological Indicators*, *9*(6). https://doi.org/10.1016/j.ecolind.2008.06.008

Moore, E. (2024, July 5). *We have lots of alligators in NC. Here's where you're most likely to see one (+ what to do)*. Where are alligators found in NC? https://www.charlotteobserver.com/news/state/north-carolina/article289741194.html

nliaudat. (n.d.). *Ellipsoid fitting using python numpy to calibrate magnetometers*. GitHub. https://github.com/nliaudat/magnetometer_calibration

North Carolina Wildlife Resources Commission. (n.d.). *Alligator, American*. Alligator, American | NC Wildlife. https://www.ncwildlife.gov/species/alligator-american

Pedrazzi, L., Naik, H., Sandbrook, C., Lurgi, M., Fürtbauer, I., & King, A. J. (2025). Advancing Animal Behaviour Research using drone technology. *Animal Behaviour*, *222*, 123147. https://doi.org/10.1016/j.anbehav.2025.123147

Price, M. (2023, June 12). *Alligator snatched drone during sheriff's office training class, Florida video shows*. Miami Herald. https://www.miamiherald.com/news/state/florida/article276315021.html

Project WILD. (n.d.). *American alligator*. Back from the Brink. https://www.fishwildlife.org/download_file/view/1010/2057

QGC. QGroundControl. (n.d.). https://qgroundcontrol.com/

Radiohead: Radiohead Packet Radio Library for embedded microprocessors. (n.d.). https://www.airspayce.com/mikem/arduino/RadioHead/index.html

Raspberry pi camera mount - camera module 3 version. Raspberry Pi Camera Mount - Camera Module 3 Version par DocWeebl | Téléchargez gratuitement un modèle STL | Printables.com. (n.d.). https://www.printables.com/fr/model/368788-raspberry-pi-camera-mount-camera-module-3-version

Ryberg, W. A., & Lawing, A. M. (2018). GENETIC CONSEQUENCES AND MANAGEMENT IMPLICATIONS OF CLIMATE CHANGE FOR THE AMERICAN ALLIGATOR (ALLIGATOR MISSISSIPPIENSIS). In *American Alligators: Habitats, Behaviors, and Threats*. essay, Nova Science Publishers, Inc. Retrieved from https://www.researchgate.net/publication/324360226_GENETIC_CONSEQUENCES_AND_MANAGEMENT_IMPLICATIONS_OF_CLIMATE_CHANGE_FOR_THE_AMERICAN_ALLIGATOR_ALLIGATOR_MISSISSIPPIENSIS.

Scarpa, L. J., & Piña, C. I. (2019). The use of drones for conservation: A methodological tool to survey caimans nests density. *Biological Conservation*, *238*, 108235. https://doi.org/10.1016/j.biocon.2019.108235

Shapely. (n.d.). https://shapely.readthedocs.io/en/stable/

South Carolina Department of Natural Resources. (2023). Alligator Hunting Season Report 2023. https://dnr.sc.gov/wildlife/alligator/pdf/2023PublicAlligatorHuntingSeasonReport.pdf

Thingiverse.com. (n.d.). *RC airboat by buildsomestuff*. Thingiverse. https://www.thingiverse.com/thing:6072240

Woodward, A. R., & Moore, C. T. (n.d.). Statewide Alligator Surveys. https://usgs-cru-individual-data.s3.amazonaws.com/cmoore/tech_publications/Woodward%20%20Moore%20(1990)%20Statewide%20Surveys%20FR-1.pdf