# Base Station System for Tracking Powder Valley Nature Center Copperhead Snakes

ESE 498 Capstone Design Project

Project Duration: January through May of 2021

Submitted to Professor Wang and the Department of Electrical and Systems Engineering
May 9th, 2021


**Client:**
Dr. Benjamin Jellen (benjamin.jellen@uhsp.edu)
University of Health Sciences and Pharmacy in St. Louis

**Advisor:**
Dr. James Feher (jdfeher@wustl.edu)
Washington University in St. Louis

**Electrical Engineering Students:**
Josh Peck (joshpeck@wustl.edu)
Washington University in St. Louis

Dan Rosenburg (d.j.rosenburg@wustl.edu)
Washington University in St. Louis

Abbie Wolfe (abbiemwolfe@wustl.edu)
Washington University in St. Louis

# Abstract

We are working in collaboration with Dr. Benjamin Jellen of the University of Health Sciences and Pharmacy in St. Louis and another senior design group to create a system to track copperhead snakes living at the Powder Valley Nature Center. The system allows scientists to pinpoint a snake's exact location and obtain temperature readings for ecological studies. We utilize long range (LoRa) modulation technology in conjunction with a time difference of arrival (TDoA) algorithm to track a small implant lacking a GPS unit. The overall system consists of implantable tracking devices for the snakes, multiple base stations that receive data packets from the implants, and a main base station that relays these data packets to a TDoA algorithm, which calculates the location of the snakes to be uploaded to a user dashboard display. Additionally, a portable locating device is developed for in-the-field locating of the snakes.

# Introduction

Dr. Benjamin Jellen, Associate Professor of Bio Sciences at the University of Health Sciences and Pharmacy in St. Louis, is an ecologist researching copperhead snakes living at Powder Valley Nature Center located in Kirkwood, Missouri. Dr. Jellen and his team hope to better understand the snake's overall ecology and population dynamics, with an emphasis on changing the public's negative associations with venomous snakes. The current system requires Dr. Jellen to go out in the field with a tracking pole to locate each snake by hand, which is not ideal and is time consuming. Further, this method uses a tracking device with a high power consumption, requiring invasive battery replacement surgeries on the snakes every 8 months.

Two previous senior design groups worked on this project. The first group started in the spring of 2020 and established a LoRa radio connection between implantable trackers and a single base station. Their implant was too large to be placed in a snake and its power consumption was still high. The second group worked on the project in the fall of 2020 with the goal of decreasing the implant’s size and power consumption. Their tracker was much smaller, but it still used too much power due to an integrated GPS unit.

Our goal in this project is to create a system that minimizes the power consumption of the tracker so the snakes do not need frequent surgeries. We believe that to do this there cannot be a GPS unit in the snake implant. Our approach is to create a “localized GPS-like” system which uses trilateration to locate the snakes in a system resembling 4 GPS satellites. Specifically, we use a system of Adafruit Feathers (base stations) communicating through long range (LoRa) modulation technology to generate data packets including the timestamp. We then pass the data packets to a time difference of arrival (TDoA) algorithm that pinpoints the snake's approximate location. Due to error ranges in the TDoA approach in studies we examined, we also employ a secondary locating device which utilizes directional antennas to search for a more specific location of the snake. Our group’s focus is designing and implementing a system of base stations

and a user interface to obtain the snakes location, while a partnering team works to design the implantable tracking devices.

# Technical Approach

To keep the implants small with low power consumption, they cannot contain a GPS unit. To accomplish this, LoRa modulation technology and a time difference of arrival (TDoA) algorithm were used to calculate the coordinates of each tracking device for display on a web dashboard. LoRa is a low-power-wide-area network modulation technique that we use to transmit signals license-free at 915MHz.

The overall system block diagram shown in Figure 1 highlights the four stages in our method. First, the tracking devices in the snakes periodically send an omni-directional signal (as it has an omnidirectional antenna) to the base stations, each containing an Adafruit Feather MO with a LoRa radio module, that simultaneously receive the signal. Next, when the base stations receive this signal, they immediately record the arrival time as a timestamp (using the onboard GPS) and forward the information packet to the main base station over the LoRa network. Third, the main base station, containing an Adafruit Feather MO with a LoRa radio and an attached Raspberry Pi, receives the information packets and runs a TDoA algorithm. The algorithm calculates an approximate location for each snake. Finally, the Pi uploads the coordinates of each snake to the user dashboard. Additionally, when Dr. Jellen needs to go out and find the snakes, the approximate location calculated by the algorithm can be used in conjunction with the mobile locating device to further narrow down location.

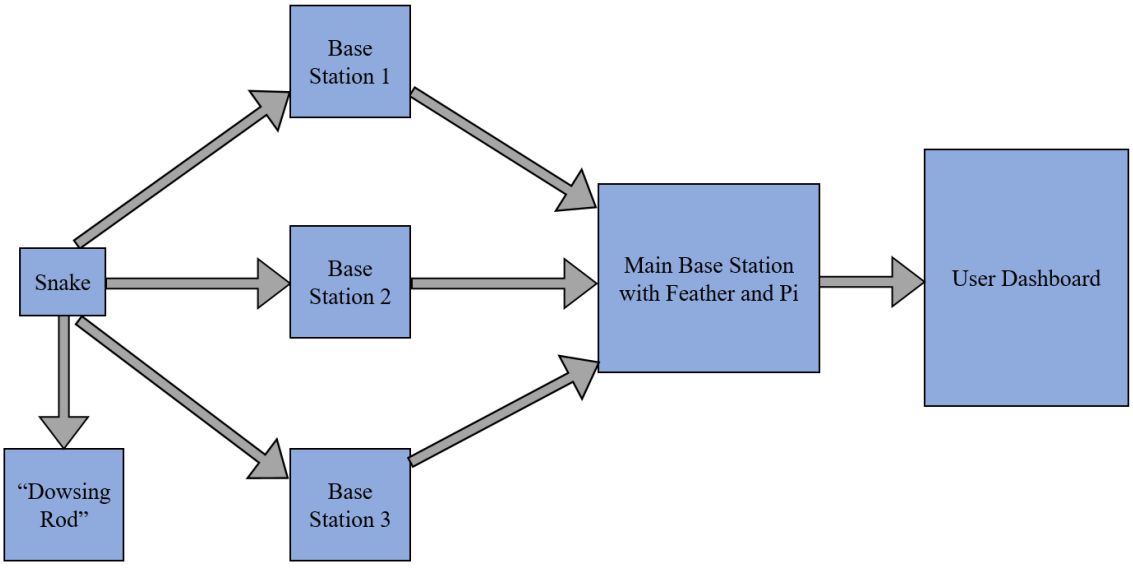


**Figure 1:** *Overall System Block Diagram*

The block diagram for the secondary base stations is shown in Figure 2. Each base station contains an Adafruit Feather MO with a RFM95 LoRa radio module that operates at 915 MHz.

They are each powered by a 5 watt (5 volt) solar panel and connected to an external battery. They also contain a GPS unit for a precise, synchronized clock and to set location, which is necessary for the TDOA algorithm computations.

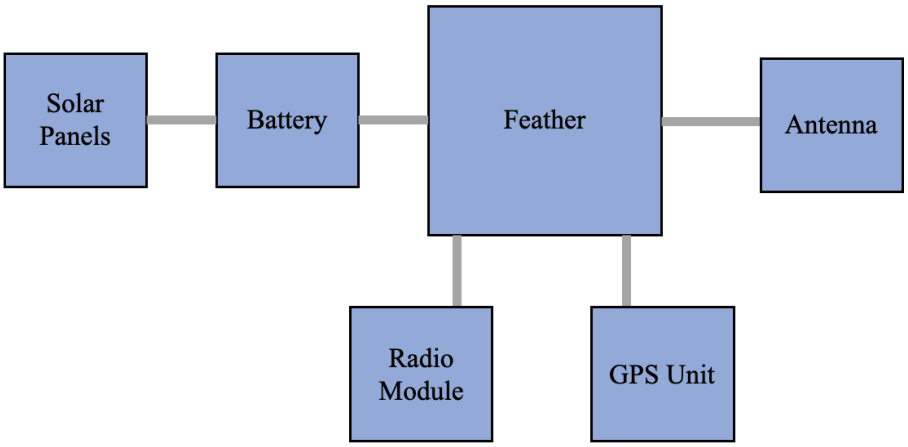


**Figure 2:** *Base Station Block Diagram*

The block diagram for the main base station is shown in Figure 3. In addition to the components in the secondary base stations, the main base station includes a Raspberry Pi. The Pi runs the algorithm that computes the implant location based on the data relayed from the base stations. It also uploads the computed solutions to a user dashboard, which displays the calculated locations of the snakes.

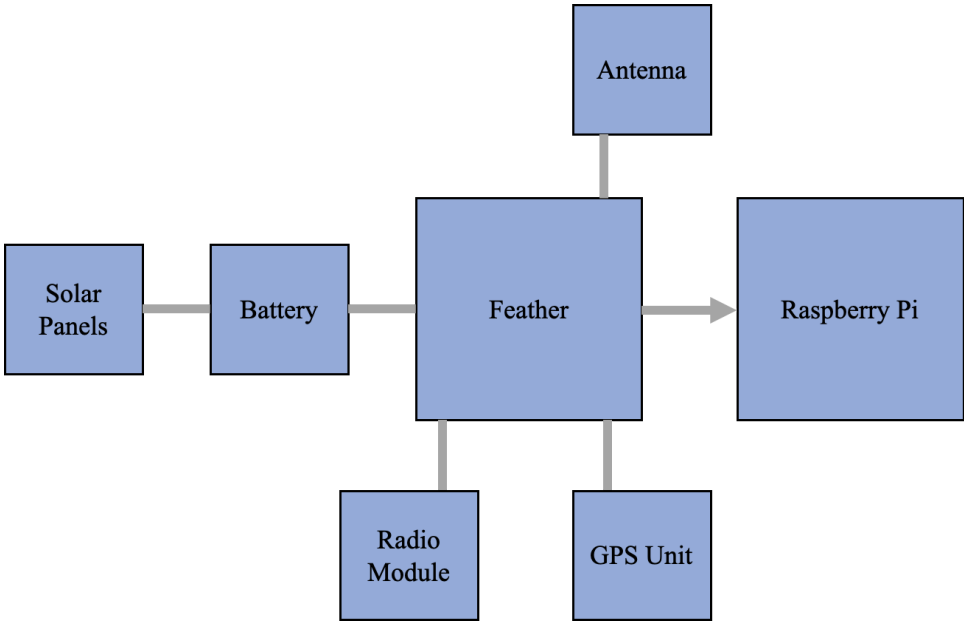


**Figure 3:** *Main Base Station Block Diagram*

The block diagram for the TDoA algorithm is shown in Figure 4. Before calculations begin, the raw data is sorted in order of snake IDs and timestamp, allowing the algorithm loop to iterate sequentially through all the data. The algorithm computes the location of the snake using the GPS coordinates of three base stations and the difference between arrival times of the signal to those base stations. The coordinates are converted to meters for the calculation, with 0ºN, 0ºE as the reference point, before reverting back to coordinates at the end. The algorithm uses the three closest base stations (and in our test implementations, exactly 3 stations) based on earliest arrival time, and uses Chan's Algorithm to compute the location. Chan's algorithm, "is an approximate realization of the maximum-likelihood estimator and is shown to attain the Cramer-Rao lower bound near the small error region." (Chan *et. al*), and is shown below:

$$\begin{bmatrix} x \\ y \end{bmatrix} = -\begin{bmatrix} x_{2,1} & y_{2,1} \\ x_{3,1} & y_{3,1} \end{bmatrix}^{-1} x \left\{ \begin{bmatrix} r_{2,1} \\ r_{3,1} \end{bmatrix} r_1 + \frac{1}{2} \begin{bmatrix} r_{2,1}^2 - K_2 + K_1 \\ r_{3,1}^2 - K_3 + K_1 \end{bmatrix} \right\}$$

where $T_{2,1} = T_2 - T_1$, $T_{3,1} = T_3 - T_1$, $c = 299792458.3$ (speed of light in m/s), $r_{2,1} = T_{2,1} * c$, $r_{3,1} = T_{3,1} * c$, $x_{2,1} = x_2 - x_1$, $y_{2,1} = y_2 - y_1$, $x_{3,1} = x_3 - x_1$, $y_{31} = y_3 - y_1$, $K_1 = x_1^2 + y_1^2$, $K_2 = x_2^2 + y_2^2$, and $K_3 = x_3^2 + y_3^2$. $T_n$ is the timestamp at the $n^{th}$ base station, $r_{n,m}$ is the difference in distances between base stations n and m and the implant, $x_{n,m}$ and $y_{n,m}$ is the distance between base stations broken into its vector components (both in meters), and $K_n$ is an intermediate variable. Solving this equation using the inputs from the three base stations produces the coordinates of the implant. To reduce the effect of noise on the time measurements and account for the limits of the timestamp's resolution, multiple measurements are taken in sequence, with the resulting locations averaged to reduce this variance. The location, as well as the other data such as the snake's temperature and battery voltage, are saved in a CSV file and transmitted to the user dashboard.

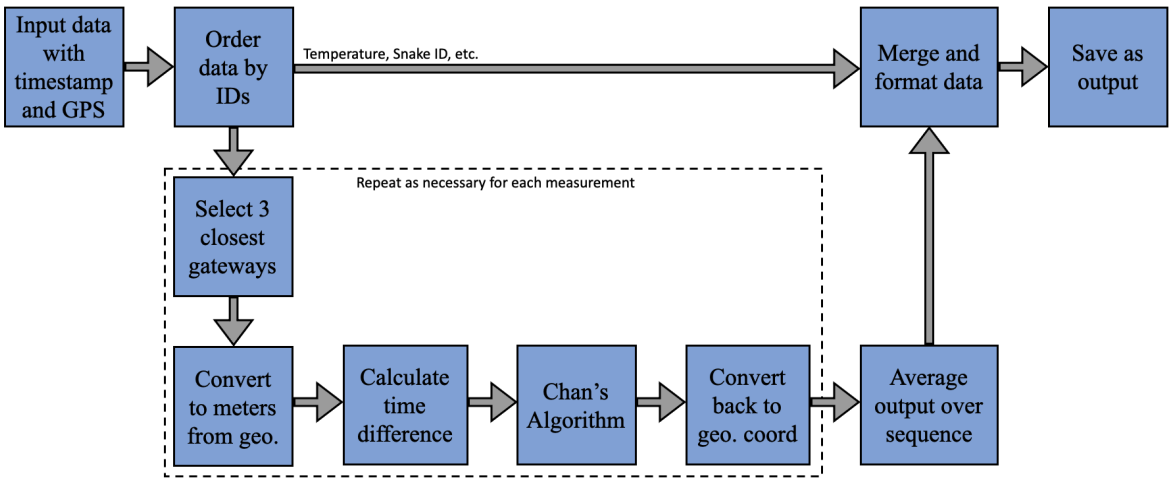


**Figure 4:** *Algorithm Block Diagram*

A user dashboard displays the site map with the last recorded location of each snake. In addition to the data visualization through the map, the output positional data is available to Dr. Jellen such that other analyses can be conducted using ArcGIS. This dashboard is hosted on the project website with our test data to illustrate how it might look. The dashboard is a primarily visual tool and does not have the ability to relay commands to either the base stations or the snakes. Rather, a "dowsing rod" we constructed will serve as the control panel side of this interface, functioning as both the control panel for the user dashboard and the locating device. This was done to keep all the code for the locating system at the Arduino level, rather than working backwards from the web interface to the Pi and then to the Feather network.

In addition to the base stations and user dashboard, we created the previously mentioned "dowsing rod", shown in Figure 5, which is a secondary sensing system. The TDoA algorithm allows for passive acquisition of the snakes' approximate locations, but has some error due to the nature of the system. However, the dowsing rod shrinks the possible locations of the snake considerably, by relying on an RSSI scheme. The control panel has the ability to switch the snakes from their "passive tracking" mode, in which the implants send a signal every four hours (although this is user configurable), to the "active tracking" mode. During active tracking, the snakes rapidly send out miniscule pings without any meaningful data. The dowsing rod, equipped with a directional Yagi-Uda antenna and OLED screen, displays the last received signal strength. The signal strength varies substantially with the direction of the antenna, as well as the distance between the transmitter and the antenna. From this, a hotter/colder sensing scheme is developed, where the highest RSSI values eventually point to the signal source: the snake.

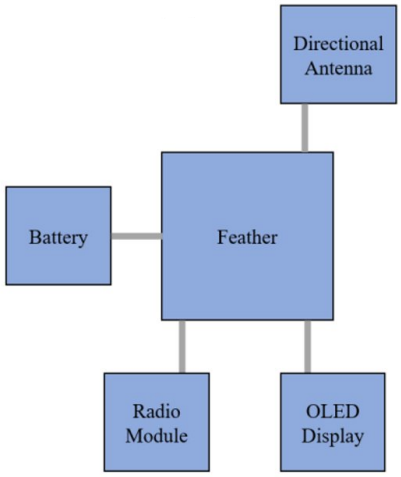


**Figure 5:** *"Dowsing Rod" Block Diagram*

# Code Logic

The programming implemented in the system falls into two groups: Feathers and Raspberry Pi. The code for the base station Feathers to communicate and record the GPS data is written in Arduino C. On the Raspberry Pi, the receiving algorithm and TDoA algorithm, as well

as the minor data cleaning tool, are written in Python. There are also two scheduled bash scripts that run Python scripts (specifically, the receiving algorithm and data cleaning tool). How this code works is described below.

## Feathers

The first items in the chain are the implantable devices (transmitters), which were developed by our partner team. In order to test our system, we utilize a Feather to act as a substitute for their implant. Our temporary transmitter consists of only a radio module (the Feather); the real implant would additionally contain a battery voltage monitor and temperature sensor. Our transmitter sends the relevant information to the base stations enclosed in each packet: the snake ID number, sequence count (the ID number of the current transmission set), ping count (the individual message in the set of 10 transmissions), the snake temperature (in our case a placeholder value), and the battery voltage (again, a placeholder value).

The base stations constantly loop through two functions: GPS information reading and millisecond timing updates. The GPS on each station reads new NMEA (National Marine Electronics Association) sentences containing location and time information. The NMEA sentence is a standardized format by which GPS data is relayed, including positional data and time, as well as other data irrelevant to this project. This information has a 10 Hz refresh rate (every 100 ms) and is stored on the Feather until a new sentence is read. Since the GPS time is restricted by refresh rate (due to our ability to poll it, not the precision of the GPS unit itself), the millisecond accuracy is limited to 100ms, which is insufficient for our purposes. Our second function determines a more accurate millisecond count by using the GPS millisecond value and extrapolating with the Feather's internal clock between new GPS information to get 1 millisecond accuracy. These two functions ensure that the Feather always has the most precise time data (that we are able to achieve) for each station. Our spatial resolution is determined by our GPS module, which has a resolution of decimal (rather than degree) minutes to four decimal places (equivalent to approximately 1 meter). When a packet is received from the transmitter, the base stations append the GPS data immediately (the current Arduino time as well as position) to the packet, such that the new packet contains all the information from the snake as well as the location of the base station and the time at which the packet was received. The base stations' known location and time of arrival are used later in the TDoA algorithm. Once the information has been appended, the next steps differ depending on whether the base station is the main station or one of the secondary units. The secondary units take this new packet and forward it to the main base station. The main base station simply prints the completed packet to serial output so it can be read by the Pi.

The Feather network uses addressability (through the use of packet headers) to ensure that messages are only being routed *from* the secondary stations *to* the main base station and so the main station can react differently to messages sent from transmitters and other base stations. Our addressing scheme is such that all the secondary base stations share one address (header),

the main base station and controller both have their own addresses, and the snake IDs addresses go up in series.

## Raspberry Pi

After the signal arrives at the main base station, the data packets are printed to the serial console of the Raspberry Pi via a hardwired USB connection. The entire process for the Python code on the Pi is depicted in Figure 6 below. The receiving Python algorithm processes the data packets to adjust the formatting before appending the packet to the end of the csv input data file corresponding to the snake's ID. The input files are separated by snake ID to eliminate the risk of data corruption when separating the processed raw input data. After a predetermined amount of time elapses since the last ping of the sequence for a snake, the ID of that snake is appended to the active snakes file, and the TDoA algorithm is called. The receiving algorithm is constantly running, awaiting future data.

The whole TDoA algorithm consists of three components: the preprocessing, the algorithm itself, and the postprocessing. During the preprocessing stage, the active snakes file is read first to determine which snakes have finished sending their sequence, meaning there is a complete set of data ready for processing. The input data files for these snakes are read and merged together in an array, and these files, including active snakes, are cleared out. No data is lost as the contents of the input data files read are logged to a master file. During the algorithm stage, the list of the raw input data is sorted by ping ID, then sequence ID, and finally by snake ID. This is so the algorithm can process data on multiple sequences and snakes at one time without missing any readings. A loop iterates through the entire array of data, calling the algorithm over each ping (consisting of 3 or more base station readings) and averages the output results by sequences, before finally saving them in an array. During the last part, the postprocessing, the newly determined snake locations are saved in up to 3 different csv files, which then are transmitted to the map. This data includes the latitude and longitude of the snake, its ID, the time of the most recent transmission, and the environmental data of voltage and temperature. The output data new csv file is updated to reflect the new locations of the snakes that had just been processed, leaving only the newest location for each snake even if two sequences of the same snake are calculated in the same algorithm run. All the calculated snake locations are logged in both the output data and the output data 14 days files, the latter only holding the previous 14 days of locations.

On the Pi there are two other sets of bash scripts, which are separate from the processing of the data. The first is a Python script that deletes input data from the master input log older than 3 days, as well as maintaining the rolling 14 day output file. As mentioned earlier, all previous results are still saved in the output data file. The second set of scripts ensures the Python code runs when needed. The bash scripts are scheduled using the built-in Linux utility Cron. Scheduled to run at midnight is a bash script that calls the old data cleanup tool so it runs once daily. The second script, set to run upon reboot and hourly, restarts the receiving algorithm after a Pi shutdown and checks hourly to ensure the receiving algorithm is still running, re-running it in

the event of a crash. Since the receiving algorithm is expected to be running continuously, it is critical to ensure it is always working. To clarify, the TDoA algorithm is only called by the receiving algorithm.

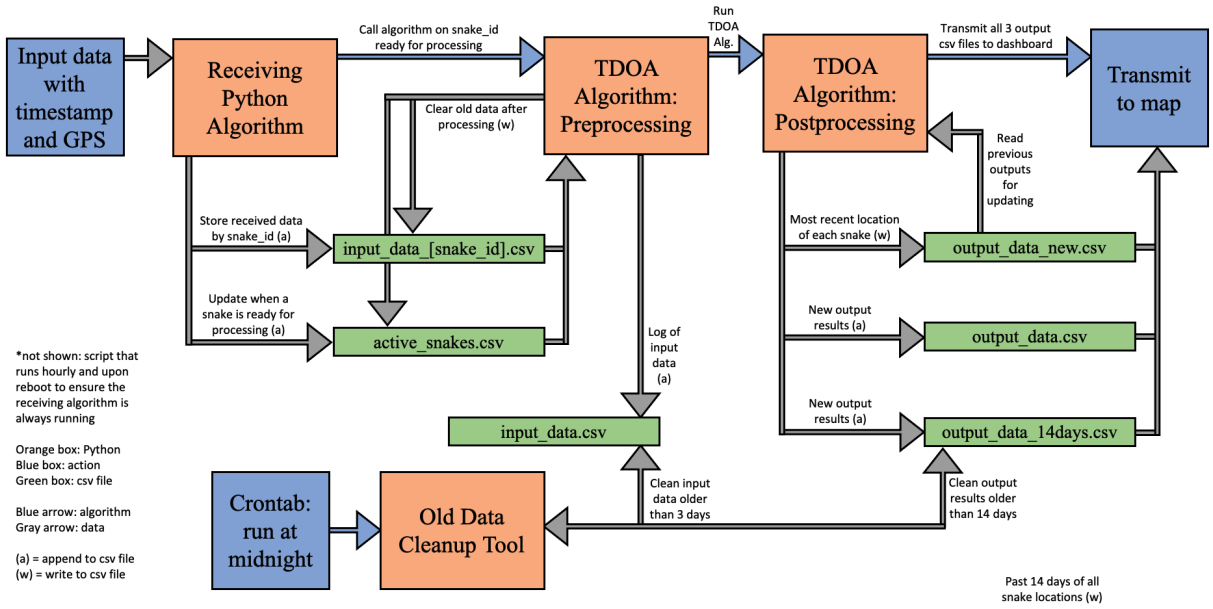


**Figure 6:** *Raspberry Pi Code and Data Flow Block Diagram*

# Data Collection and Procedures

To test the system and collect trial data before the completion of the implant, we repurposed the fourth base station's Feather to be a transmitter programmed to send data in a format identical to the real implant (leaving us with 3 base stations, the minimum needed for TDoA). For data collection purposes, the transmitter was programmed to send a specific number of pings when initialized by a button press on the physical device, with the set of pings constituting one sequence.

The first round of testing was completed inside an apartment with the purpose of verifying the chain of communication between the base stations. We checked that the data packets were received by the right device in the correct format. First, we tested the ability of the base stations to receive the transmitter's ping and append the GPS coordinates and timestamp of the arrival time to the data packets. This was verified by printing the data packets from each base station over serial to a computer.

We then checked that the secondary base stations could forward the data packets over LoRa to the main base station. At first, this was unsuccessful due to addressability issues. Implementing packet headers for addressability solved these problems and ensured the base stations only forwarded their packets to the main base station. Once fixed, the data packets were successfully transmitted to the main base station as verified by the example in Figure 8

```
DATA,20,3,0,71,10,2021-04-21 01:22:15.828, 3836.1394,N, 9014.3252,W
LDATA,20,3,0,71,10,2021-04-21 01:22:15.534, 3836.1394,N, 9014.3369,W
DATA,20,3,1,71,10,2021-04-21 01:22:19.660, 3836.1389,N, 9014.3252,W
DATA,20,3,1,71,10,2021-04-21 01:22:19.511, 3836.1421,N, 9014.3291,W
LDATA,20,3,1,71,10,2021-04-21 01:22:18.020, 3836.1394,N, 9014.3369,W
DATA,20,3,2,71,10,2021-04-21 01:22:22.536, 3836.1389,N, 9014.3262,W
DATA,20,3,2,71,10,2021-04-21 01:22:22.536, 3836.1389,N, 9014.3262,W
LDATA,20,3,2,71,10,2021-04-21 01:22:21.798, 3836.1394,N, 9014.3369,W
LDATA,20,3,2,71,10,2021-04-21 01:22:21.798, 3836.1394,N, 9014.3369,W
```

**Figure 8:** *Example of Successful Packet Transmission to Main Base Station*

The second round of testing involved moving beyond the apartment and onto Mudd Field, a large open field in the center of Washington University's Danforth Campus. Here, the full system was tested, from the transmitter pings and recording the timestamp to base station to base station communication and running the TDoA algorithm without user intervention. The pings were sent in sets of 10, spaced 500ms apart. The base stations were set up in a triangle enclosing around half of Mudd Field, with someone holding the transmitter (connected to a battery) within the enclosed region. Initially there were issues acquiring a GPS satellite fix, resulting in broken serial messages, but that problem was isolated and resolved. Figure 9 shows an example of a data packet successfully read on the Pi's serial console.

```
20,3,0,71,10,2021-04-21 01:22:15.828, 3836.1394,N, 9014.3252,W
```

**Figure 9** *Data Packets Printed in Serial Console of the Pi*

The transmitter was set to use a snake ID of 20. Figure 10 shows part of the input data file written by the receiving algorithm on the Raspberry Pi. More specifically, the contents came from the input_data_20.csv file, and the transmitter was on sequence 5.

```
2021-04-25 17:39:53.393,20,11,0,71,10,38.648755,-090.309912
2021-04-25 17:39:55.091,20,11,0,71,10,38.64834,-090.310595
2021-04-25 17:39:53.393,20,11,0,71,10,38.648755,-090.309912
2021-04-25 17:39:55.091,20,11,0,71,10,38.64834,-090.310595
2021-04-25 17:39:53.393,20,11,0,71,10,38.648755,-090.309912
2021-04-25 17:39:55.091,20,11,0,71,10,38.64834,-090.310595
2021-04-25 17:39:56.514,20,11,0,71,10,38.64825,-090.310433
2021-04-25 17:39:57.233,20,11,1,71,10,38.64834,-090.310595
2021-04-25 17:39:57.233,20,11,1,71,10,38.64834,-090.310595
2021-04-25 17:39:56.887,20,11,1,71,10,38.648758,-090.309912
2021-04-25 17:39:58.771,20,11,1,71,10,38.64825,-090.310433
```

**Figure 10:** *Snapshot of input_data_20.csv*

When the algorithm attempted to execute using this data, it failed, throwing a NonInvertibleMatrixError as seen in Figure 11, which occurs when there is repeat data. As can be seen above in Figure 10, we discovered bouncing of the signal between base stations at this point, resulting in repeated detections of signals at the main base station, complete with identical timestamps.

```
Error calculating TDOA for snake 20, sequence 11, ping 1: NonInvertibleMatrixError: Matrix det == 0; not invertible.
```

**Figure 11:** *Algorithm Error Message when Repeat Data is Present*

To run the algorithm for proof-of-concept despite this unforeseen problem, we manually parsed the data, removing repeat entries. Part of the cleaned data set is shown in Figure 12. It is important to note that the color differences are for readability to separate the data by sequences, and does not symbolize a difference in the data.

```
timestamp,snake_id,sequence_id,ping_id,temperature,voltage,latitude,longitude
2021-04-25 17:39:53.393,20,11,0,71,10,38.648755,-090.309912
2021-04-25 17:39:55.091,20,11,0,71,10,38.64834,-090.310595
2021-04-25 17:39:56.514,20,11,0,71,10,38.64825,-090.310433
2021-04-25 17:39:57.233,20,11,1,71,10,38.64834,-090.310595
2021-04-25 17:39:56.887,20,11,1,71,10,38.648758,-090.309912
2021-04-25 17:39:58.771,20,11,1,71,10,38.64825,-090.310433
2021-04-25 17:39:59.698,20,11,2,71,10,38.64834,-090.310595
2021-04-25 17:40:0.025,20,11,2,71,10,38.648758,-090.309912
2021-04-25 17:40:0.448,20,11,2,71,10,38.64825,-090.310433
2021-04-25 17:40:5.273,20,11,3,71,10,38.648758,-090.309912
2021-04-25 17:40:4.971,20,11,3,71,10,38.64834,-090.310595
2021-04-25 17:40:5.186,20,11,3,71,10,38.64825,-090.310433
2021-04-25 17:40:7.772,20,11,4,71,10,38.648758,-090.309912
2021-04-25 17:40:7.930,20,11,4,71,10,38.64834,-090.310595
2021-04-25 17:40:9.258,20,11,4,71,10,38.64825,-090.310433
2021-04-25 17:40:10.733,20,11,5,71,10,38.648758,-090.309912
2021-04-25 17:40:10.939,20,11,5,71,10,38.64834,-090.310595
2021-04-25 17:40:11.235,20,11,5,71,10,38.64825,-090.310433
2021-04-25 17:40:16.420,20,11,7,71,10,38.64834,-090.310595
2021-04-25 17:40:16.436,20,11,7,71,10,38.648758,-090.309912
2021-04-25 17:40:18.194,20,11,7,71,10,38.64825,-090.310433
2021-04-25 17:45:3.563,20,13,2,71,10,38.64834,-090.310595
2021-04-25 17:45:3.535,20,13,2,71,10,38.648763,-090.309912
2021-04-25 17:45:3.068,20,13,2,71,10,38.648303,-090.31045
2021-04-25 17:45:17.302,20,13,6,71,10,38.648763,-090.309912
2021-04-25 17:45:17.427,20,13,6,71,10,38.64834,-090.310595
2021-04-25 17:45:16.426,20,13,6,71,10,38.648303,-090.31045
2021-04-25 17:45:20.751,20,13,7,71,10,38.64834,-090.310595
2021-04-25 17:45:20.417,20,13,7,71,10,38.648763,-090.309912
2021-04-25 17:45:18.684,20,13,7,71,10,38.648303,-090.31045
2021-04-25 17:45:26.437,20,13,9,71,10,38.648763,-090.309912
2021-04-25 17:45:26.499,20,13,9,71,10,38.64834,-090.310595
2021-04-25 17:45:25.011,20,13,9,71,10,38.648303,-090.31045
```

**Figure 12:** *Cleaned Data Set from Mudd Field Testing*

The next step after the input data is collected should be the TDoA algorithm. Due to the problems with signal bouncing as described above, the data was manually cleaned as described above to remove the repeated lines, and then fed to the algorithm.

Execution of the TDoA algorithm failed, throwing a CantSolveMatrixError as shown in Figure 13. This occurs when the matrix of Chan's algorithm has no solution. Upon inspection of the data, it was discovered the difference in timestamps was much larger than expected. This is discussed in more detail later in the discussion section, but in summary the differences in arrival times should be on the order of milliseconds at the most, not entire seconds. This led to the conclusion that there was something else affecting the timestamps on the Feather base stations, and that the differences were not correlated with distance. To get around this issue for proof-of-concept purposes, we noticed that scaling the speed of light constant in the TDoA algorithm by $10^{-8}$ allowed for us to continue testing the algorithm as a temporary fix.

```
Error calculating TDOA for snake 20, sequence 11, ping 0: CantSolveMatrixError: IndexError at TDOA matrix solve step, possible timing issue.
Error calculating TDOA for snake 20, sequence 11, ping 1: CantSolveMatrixError: IndexError at TDOA matrix solve step, possible timing issue.
Error calculating TDOA for snake 20, sequence 11, ping 2: CantSolveMatrixError: IndexError at TDOA matrix solve step, possible timing issue.
Error calculating TDOA for snake 20, sequence 11, ping 3: CantSolveMatrixError: IndexError at TDOA matrix solve step, possible timing issue.
Error calculating TDOA for snake 20, sequence 11, ping 4: CantSolveMatrixError: IndexError at TDOA matrix solve step, possible timing issue.
Error calculating TDOA for snake 20, sequence 11, ping 5: CantSolveMatrixError: IndexError at TDOA matrix solve step, possible timing issue.
Error calculating TDOA for snake 20, sequence 11, ping 7: CantSolveMatrixError: IndexError at TDOA matrix solve step, possible timing issue.
```

**Figure 13:** *Algorithm Error Message when Time Difference is too Large*

The final step is uploading the calculated coordinates to the user dashboard. Our system was designed to take the solutions and automatically upload them. However, there were a couple of hiccups in this process. ArcGIS requires direct access to a hosted CSV file in order to update the locations without any user intervention. Cloud services, such as Google Drive, were not compatible with this method due to lack of Linux client support. We were, however, able to demonstrate the auto-update feature by editing the output CSV file hosted on WUSTL Box. The map updated as expected, but required user intervention.. A comprehensive uploading solution still needs to be determined in order to have autonomous data collection.

We attempted further modification and testing to the Feather base station code to try to remove bouncing and solve the timing issue, but could not find any easy solutions given the time constraints. Our initial assumption was that bouncing was due to reflection from close quarters indoors or perhaps due to multiple signals from multiple stations being received at the same time, but testing with only one station, and outside still caused bouncing. When only the main base station was used with the transmitter there were still occasional repeated transmissions. This is discussed in greater detail in the discussion and future endeavours sections.

# Results

## Algorithmic Calculations

Although we encountered issues during testing that prevented data from being collected that could be used in a final implementation, we were able to demonstrate a working system, with all components integrating together as intended. With this, there is sufficient data to show what would happen. As mentioned previously in the data collection section, we discovered signal bouncing that caused repeat data packets to show up at the main base station. We extracted a data

set to have no repeats, shown in Figure 11, to run in the algorithm. The data we gathered had a larger difference in timestamps than expected, so we scaled the speed of light constant in the algorithm by a factor of $10^{-8}$ in order to test the algorithm for proof-of-concept. Although this no longer represents what physically happened due to the timing issue, analysis can still be performed on it when assuming the timing was correct, as will follow. Due to this, we were not able to test the minimum number of pings needed when averaging to achieve optimal accuracy.

Below in Figure 14 represents the final output of one of the tials. The blue dot represents the actual position of the transmitter, as measured by using the current location function in Google Maps on an iPhone. The green dot represents the calculated location of the transmitter from Chan's algorithm (TDoA) using the collected data. Under the assumption the timing was accurate, this represents an error of 21 meters between the actual and calculated location of the transmitter. The red dots represent the locations of the three base stations during the test. The coordinates of these points can be found in Table 1.

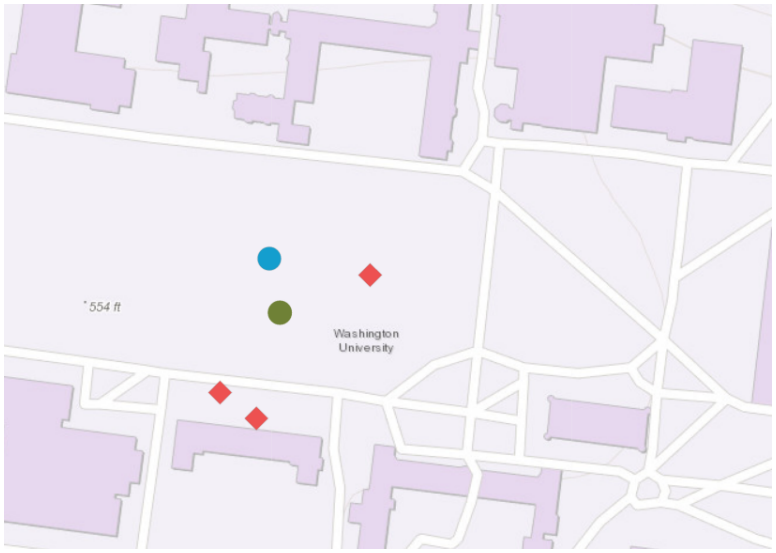


**Figure 14:** *ArcGIS Map Displaying the Calculated and Actual Data*

| Point | Latitude | Longitude |
|---|---|---|
| Actual Location (blue dot) | 38.648823 (N) | -90.310377 (W) |
| Calculated Location (green dot) | 38.648899 (N) | -90.305587 (W) |
| Base Station 1 (closest to the DUC) | 38.648254 (N) | -90.310425 (W) |
| Base Station 2 (closest to Simon Hall) | 38.648342 (N) | -90.310597 (W) |
| Base Station 3 (closest to Olin Library) | 38.648756 (N) | -90.30991 (W) |

**Table 1:** *Data Points for Markings in Figure 14*

## Power Consumption of Stations

Before purchasing a battery, we did back-of-the-envelope calculations to determine the power consumption of the base stations. Using a small USB port multimeter, the average current draw of the Feather and the GPS was in total 40 mA. This was determined using the known 5V input using USB charging and the mAh value over a one hour span. We purchased a 5V, 1A (5W) solar panel in addition to a solar charge controller connected between the panel and 3000 mAh battery.

Our theoretical runtime with a 3000 mAh battery (without any recharging) was 75 hours, or just over 3 days. It should be noted that this calculation smooths over any actual LoRa transmission done by the radio where the current draw can go as high as 100 mA. The multimeter we had did not allow for this to be observed, as the time resolution of the tester was too low. Since transmissions are so infrequent (only a couple total seconds every day), we believe the base power calculations are sufficient.

When hooked up to the solar cell and a charged battery, the base station functioned the entire testing period of 5 days, and could function for presumably much longer. Further passive testing could be conducted if we had more time, and more accurate power measurement technology would allow for a closer look.

Then we tested the cell and solar charger without any battery. The cell is 5V but the Feather operates on 3.7V logic and the step-down is handled by the charge controller. When the solar cell was used exclusively, the Feather was unable to remain on stably. Only when the panel was in full direct sunlight was the Feather able to remain on. Our average power draw was 0.15W, so the fact that the panel was unable to power the Feather is concerning. This could be due to the actual output of the panel being less than the 1W rating (as the rated value is the maximum value), or possible resistive drops from the charge controller. The implication of this is that the base stations may require a larger solar panel in order to operate perpetually. While our preliminary tests with the battery and solar cell seem to suggest an indefinite run time, the solar power may not be greater than the energy requirements of the load if not in direct sunlight. Placement of the solar panel is important and may require to be placed up high to ensure maximum power generation.

## “Dowsing Rod” RSSI Accuracy

Our homemade Yagi-Uda directional antenna proved to be fairly accurate and had significant attenuation outside of the main lobe depicted in the ideal radiation pattern below. The RSSI sensing scheme of following the lowest value (-22 dB being a near perfect signal strength directly next to and pointing at the transmitter) was a success with both ourselves being able to find a hidden transmitter (testing our ability to use the system), as well as the ability of an untrained person being able to locate the transmitter (testing how well the system works to an outsider). The exact radiation pattern of our antenna is still unknown (although it is anecdotally similar to the theoretical one), but the system is sufficient for finding transmitters. The gradations in signal strength over the course of a couple hundred meters (since this is the worst case

accuracy of our TDoA algorithm) were significant ranging from -108dB at the most distant and in the opposite direction of our transmitter to -22dB near the transmitter facing it. This gradient is a good size, allowing for significant variability to be determined due to direction even if distance remains constant, which is essential for a secondary sensing scheme like this.

# Discussion

The main objective of the project as outlined at the beginning of the semester was to create a tracking system for implantable devices that do not contain a GPS unit. We proposed doing so with a system of Adafruit Feathers with LoRa modules connected to a Raspberry Pi that runs a TDoA algorithm. We had success assembling the hardware for the system, establishing communication between all units, and running the algorithm with the appropriate data. Additionally, we proposed a dowsing rod for in the field testing along with weatherized enclosures. The dowsing rod works as planned with the exception of the ability for the user to change the active mode through the dashboard. This will instead be done directly through the dowsing rod and further transmitter code which utilizes addressability functions used in other applications such as the different behavior the main base station exhibits when receiving messages from a transmitter versus another base station. The communication between the transmitter worked on by our colleagues and the base stations we created was not successful. However, their SAMR34 board was able to receive our messages from the Feather (such as might occur when the “dowsing rod” switches the implant transmitters to active pinging). We made progress towards constructing the weatherized enclosures for the base stations, but once we became unsure of the status of a final system with working transmitters, we decided to pause that work. However, future work on this is straightforward and will be explained in the next section. Finally, we proposed a user dashboard using ArcGIS that would update Dr. Jellen on the snake's location and store this data. Our current user dashboard has the data separated by CSV files, but adding future functionality would primarily be done in ArcGIS (filtering the information through that, rather than from the CSV itself).

As we tested the system we ran into a number of complications. In the first round of apartment testing, the base stations could communicate with the transmitter but not each other. We determined that this was an addressability issue between the base stations and resolved it using additional libraries which allowed for addressable messages. While we were able to communicate between all of our devices, our partner team was unable to establish communication between the SAMR34 and Feather. This is an issue caused by the SAMR34 requiring an initial handshake between devices in order to establish a network of devices. This is more akin to LoRaWAN (a networking protocol built on top of LoRa communication) than LoRa (as our network communicated). All of our systems now speak to one another with addressable messages and functions associated with specific sender/receiver pairs, but we are unable to communicate with the SAMR34.

In the Mudd field round of testing we found two bugs in our system. The first was that if the GPS units did not fully acquire its position from the satellites (a “fix”), the data packets received by the main base station did not include latitude and longitude coordinates. This was resolved by using a conditional statement in the base station code to only receive a message if the GPS has a fix. Fortunately, once a GPS acquires a fix, it does not lose it. Combined with our solar cell and battery, the GPS also has a backup coin battery which ensures a fix is maintained.

Recording the timestamps is also a concern for a real-world implementation. As described above, we were able to get timestamp values down to single millisecond accuracy, but there were diverging timestamps between the three stations (far more than we might expect from the differing arrival times (which should be in the 10s of milliseconds, if they are even that high). Since each base station processes incoming messages identically and is equipped with identical hardware, there should not be this level of difference between the timestamp values. Investigating the source of this (possibly a hardware issue with the GPS modules themselves) is essential to creating an implementable solution.

The final obstacle we ran into was the bouncing of signals being sent by the base stations. Various attempts to reduce bouncing were undertaken. Using different send commands we tried to ensure that messages were not repeated. Since our messages did not exceed the maximum packet size, each message is only sent in one packet, so repeats cannot be due to this. Additionally, the code only allows for one serial print for each message, yet multiple, identical outputs occur. As to why this occurs, it is somewhat of a mystery. The current Arduino function used is the “sendToWait” function which delays sending on the transmitter end until the radio of the base station is not busy sending to the main station or receiving a transmission from another station or a snake. This function does have some kind of communication between the two stations, but should not trigger bouncing messages. Nevertheless, the issue persists. It is possible that bouncing is due to the environment; the apartment where testing was conducted had plaster walls with significant metal meshing within them. However, bouncing persisted in outdoor tests as well, possibly due to building materials on campus. Future work will need to be done to filter repeat messages, possibly at the Arduino level if it can be done quick enough such that other incoming messages are not missed.

Although the timestamps are not as precise as would be ideal and the signal bouncing was not resolved, we were able to manually extract the data so we had a set to run in the algorithm.

## Future Endeavors

While the main goals of this project were met, unforeseen complications need to be addressed by future groups to ensure that Dr. Jellen can effectively track copperhead snakes. The biggest problems to be solved by future groups are the communication between the transmitter and the base stations (mainly signal bouncing) and timing resolution. Cleaning this on the Arduino level prevents any complications on the algorithm end and ensures that we are passing clean data. Eliminating the bouncing makes the system more streamlined, as cleanup work does

not need to be performed by the algorithm. Further testing on bouncing could be done by checking the RSSI of the signal, as each subsequent bounce should have less signal strength.

We set out to create a user dashboard that would show the trackers' current locations and provide past data on where the snakes have traveled. Our dashboard shows the snake's current position and can be easily configured to show history. This would be done in ArcGIS and could utilize a master list of all the previously reported locations using symbology settings. A computer science project would be to upgrade the user dashboard and control panel to be more sophisticated and detailed. Auto-uploading of the output CSV files is an additional step to take to create a fully autonomous system. Additionally, we worked around our timing inaccuracies by scaling the speed of light, and while we know this is not a solution, it allowed us to test the algorithm to get proof-of-concept. In order to make the system reliable, the timing discrepancies will need to be remedied. The fact that our GPS units did not always display the same time (one base station repeatedly had a much larger difference in timestamp than we would expect) is a larger roadblock to implementing this on our current hardware. Even if we had completely synchronized clocks with 1 millisecond accuracy, we would still have overly-large errors because of how fast the speed of light is. We would ideally have microsecond timing, but this cannot be done directly using Arduino library functions. Achieving this could be a project in and of itself. Finally, since the system is not fully implementable, we did not complete weatherized enclosures for the base stations. When the entire system is reliably working this will need to be completed. Power consumption of the base stations look promising, but special attention will be needed when placing the stations (southward exposure, unobstructed view of the sun, etc.), and a larger solar panel may be needed if extended testing shows the base stations eventually run out of battery.

# Conclusion

The main outcome of this project was proof-of-concept for a trilateration TDoA algorithm and system. Furthermore, although the snake trackers do not talk to the base stations, we successfully established communication between a system of Adafruit Feather M0s with LoRa radios. After some additional refinement, this system can be used to track small reptiles in a vast number of applications. There are some considerations that must be made with the current hardware we are using (timing accuracy, bouncing issues) that can likely be rectified by future software improvements.

# Deliverables

This semester we developed a working tracking system for small reptiles. The system design is highlighted in this report to be turned into the ESE department and given to Dr. Benjamin Jellen. We also prepared a presentation and webpage for the senior design class. Finally, the following elements were built and tested as a part of the system.

First, there are **3 working base stations** (although we have a total of 5 Feathers) that receive signals from tracking devices implanted in snakes and then transmit these signals along with a timestamp and their GPS coordinates to the main base station. All three base stations have a 3D printed holder and can be powered by a solar panel and battery system as seen in Figure 15.

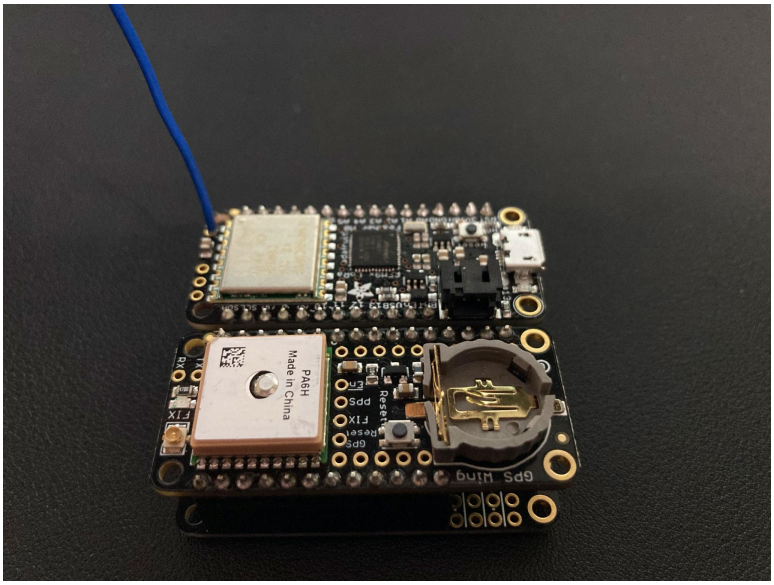


**Figure 15:** *Picture of a Base Station*

Next, we built a **main base station** consisting of an Adafruit Feather MO that receives data packets from the 3 base stations, and a Raspberry Pi that runs the TDoA algorithm and determines the snake's location. The main base station is shown in Figure 16.

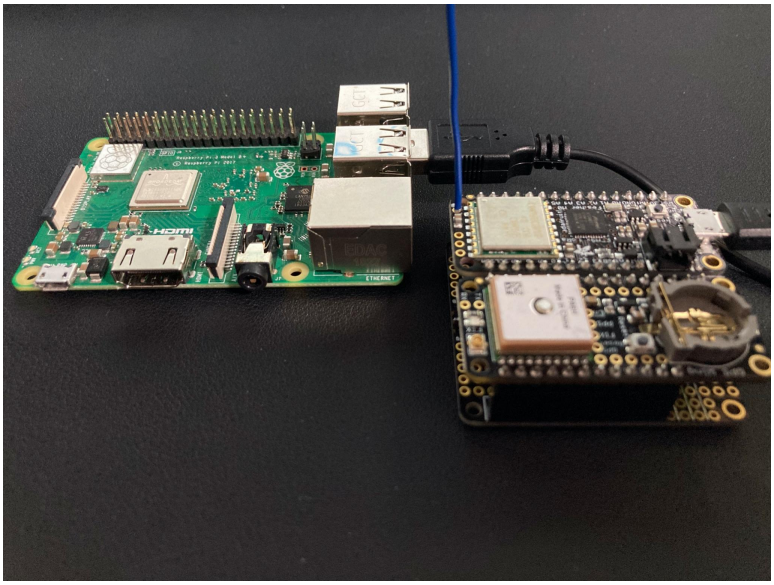

**Figure 16:** *Picture of the Main Base Station*

Additionally, we constructed a **portable "dowsing rod"** with an operating mode that allows for the snake's exact location to be found when enabled and used in the field as shown in Figure 17.

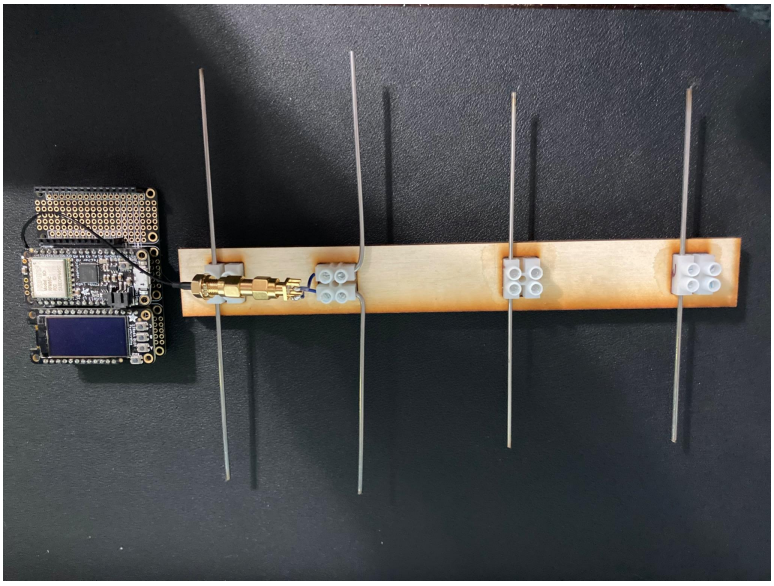

**Figure 17:** *Picture of the Portable Dowsing Rod*

Finally, there is a **user dashboard** using ArcGIS where the locational data of the snakes can be accessed. An example of the user dashboard is shown in Figure 18.

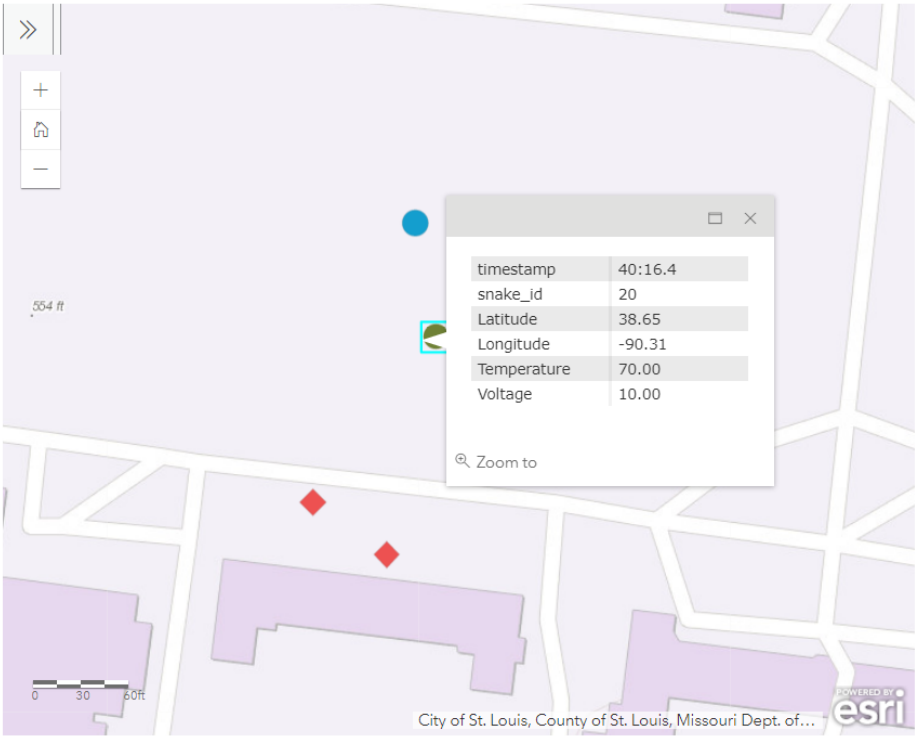

**Figure 18:** *Example of the User Dashboard*

# Schedule

Throughout the semester we predominantly stayed on schedule aside from a couple of unforeseen roadblocks we ran into. The final schedule we ended up following is shown in the Gantt chart in Figure 19.

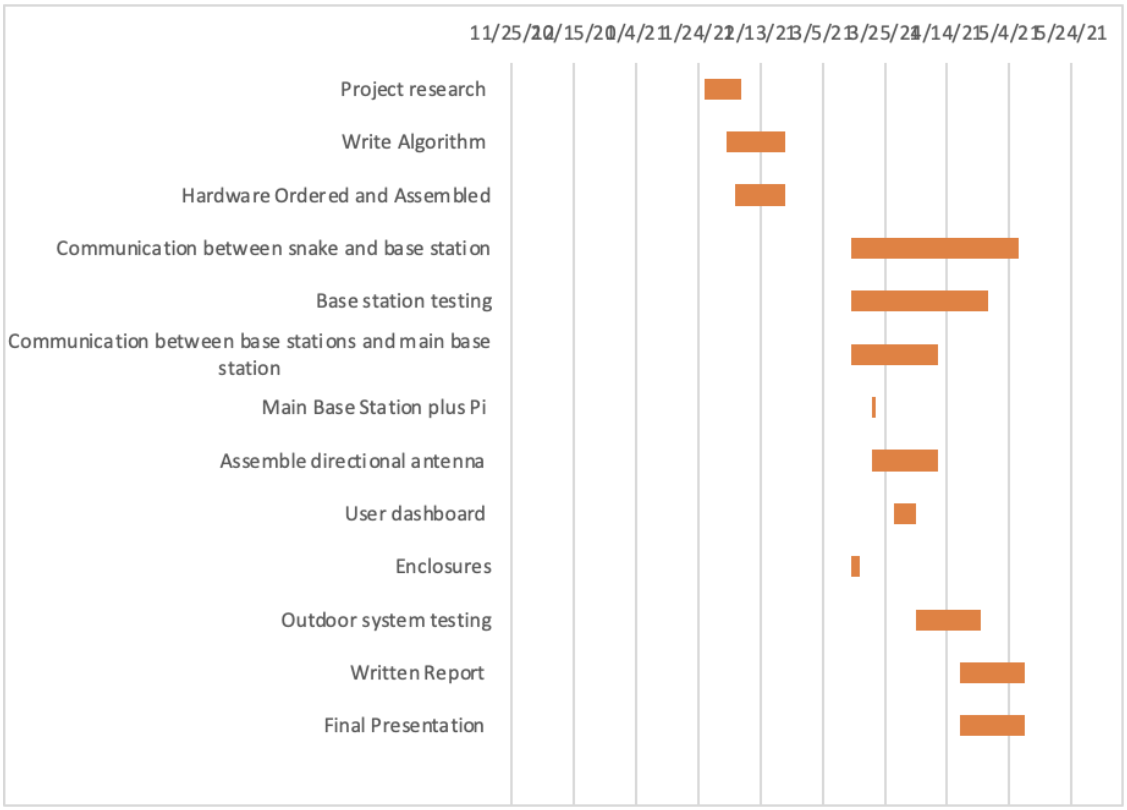


**Figure 19:** *Gantt Chart of Project Schedule*

# References

**Chan, Y, T., et al,**, "A simple and efficient estimator for Hyperbolic Location," *IEEE Transactions on Signal Processing, vol. 42, No. 8,* **August 1994**

**Choi, W., *et. al.***, "Low-Power LoRa Signal-Based Outdoor Positioning Using Fingerprint Algorithm," *International Journal of Geo-Information,* **2018** (11), https://doi.org/10.3390/ijgi7110440

**Fargas, B. C., *et. al.***, "GPS-free Geolocation using LoRa in Low-Power WANs," *Proceedings of 2017 Global Internet of Things Summit,* **2017**, https://doi.org/10.1109/GIOTS.2017.8016251

**Kays, R., *et. al.***, "Tracking Animal Location and Activity with an Automated Radio Telemetry System in a Tropical Rainforest," *The Computer Journal,* **2011**, https://doi.org/10.1093/comjnl/bxr072

**Podevijn, N., *et. al.***, "TDoA-Based Outdoor Positioning with Tracking Algorithm in a Public LoRa Network," *Wireless Communications and Mobile Computing,* **2018**, https://doi.org/10.1155/2018/1864209

**Pospisil, J., *et. al.***, "Investigation of the Performance of TDoA-Based Localization Over LoRaWAN in Theory and Practice," *Sensors,* **2020** (20), https://doi.org/10.3390/s20195464

**Zarlenga, Dan**. "Powder Valley Nature Center Reveals Results of Ongoing Snake Study at a Special Program Aug. 23." *Missouri Department of Conservation*, 12 Aug. 2019, https://mdc.mo.gov/newsroom/powder-valley-nature-center-reveals-results-ongoing-snake-study-special-program-aug-23