Sam Hessenauer
Ryan Kral
Corbin Adelman

Radio Collar Tracker Project

Abstract

As destruction of nature increases to make room for buildings, logging, and agriculture, it becomes necessary to measure the impact of the construction on the native animals. Drone tracking technologies in which automated UAVs are used to track a variety of animals, with radio collars, are an offered alternative to traditional tracking methods. By improving the GPS accuracy of the drone, we can create a more efficient model for tracking animals by offering a more precise projected position. Integrating Real Time Kinematic (RTK) GPS improves the system from meter-grade to centimeter-grade accuracy. Through testing, we found that stability of RTK GPS was tough to come by, but when working, a 45 times increase in accuracy was shown above traditional GPS.

Introduction

As society grows, populations increase, and demand for new infrastructure rises, the need for environmental impact reports rises. The reports are very costly due to their need to analyze air and water quality impact, cost analysis, and perhaps most importantly, the impact to threatened or endangered species. The Radio Collar Tracker project aims to offset the high cost of analyzing the threat to animals by eliminating thousands of man hours spent by ecologists and biologists on their foot tracking of animals. This is done by the use of a small unmanned aerial vehicle (UAV) equipped with a custom multi-frequency radio receiver payload. By flying the UAV over animal habitats, the UAV can capture a range of radio frequencies and amplitudes and then return the information it has captured for post-processing to determine the wildlife's location. This will allow for much more efficient surveys than walking and allows for more area to be surveyed more frequently. The purpose of this project is to increase the accuracy of the wildlife's location by improving the Global Positioning Satellite (GPS) onboard the UAV. The current system uses standard GPS which has a precision level of a few meters. It will then use this data and with post-processing, combine it with the gathered radio frequencies to determine the wildlife's location. Using offset data will produce an offset location which will become more and more inaccurate the farther away the UAV is from the animal.

To improve the accuracy of the system, we set out to integrate a new GPS method known as Real Time Kinematic (RTK) GPS. The RTK GPS approach increases the precision of the system from a few meters to a couple of centimeters. We chose to work on integrating RTK GPS into the system overall, because it would provide instant measurable and useful results. The RTK GPS hardware is developed by PIKSI. Included with the hardware is a GUI which allowed us to see the GPS positioning in real time. Additionally, the trackable measurements such as time, latitude, longitude and altitude, are outputted in a CSV file which allows us to compare RTK positioning with GPS positioning. The hardware combined with the software allows us to provide ecologists and biologists proof of the efficiency of the new system.

The rest of this paper will present the technical overflow and progression of our project. This includes a technical overview of how RTK GPS works, our approach to creating what is known as a "fixed RTK lock" through our various methods of testing, the python and C software we developed for scripting to streamline post-processing, and finally our planned milestones and our progression throughout the quarter. By the end of this paper, you should be able to identify how RTK GPS can positively impact a radio collar tracking application, and realize the costs and benefits associated with the approach that we took.

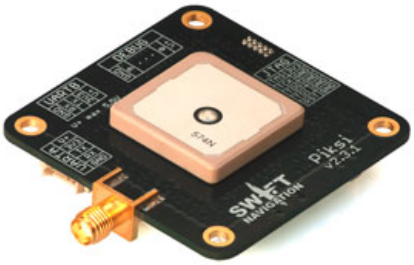
Sam Hessenauer
Ryan Kral
Corbin Adelman

Technical Material

For this project, the technical work completed can be broken down into the following sections: RTK GPS, RTK Testing, and Software Development and Integration.

Real Time Kinematic (RTK) GPS

Traditional Global Positioning System (GPS) uses the timing differences between signals transmitted from satellites to a receiver which then digitally processes the data in order to find a location. This traditional method however, has an accuracy error of approximately ~5m+. In Real Time Kinematic GPS, there is a Base station module on the ground as well as a Rover. As long as the Rover and the Base maintain at least 5 satellites in common, there can be a more accurate locational prediction of the Rover by adjusting the corrections determined by the Base station. This RTK solution can provide centimeter-grade accuracy of the position, and should cause a greater than 200 times increase in accuracy compared to traditional GPS. The major benefits are the extreme precision of the GPS unit for any application, with an option for real time tracking, it will be a crucial player in the future of drone technology. The downside to this technology is the cost, with traditional GPS modules costing ~$50 and high precision RTK GPS modules costing thousands of dollars which is a major setback for projects that need extremely accurate positioning. As a solution, we looked towards a module developed by Swift Navigations called Piksi. Piksi is an extremely low cost, two module RTK GPS system that is native to linux software. This technology is also open source, which allows for the GPS community help develop the board's systems along the way which will increase its evolution. The Piksi modules are also extremely small, with a form factor of only 53x53m. The RTK GPS system is low in power consumption, which allows for a reduction in battery weight needed on the aerial vehicle with a mere 500mW power consumption. Although, the Piksi RTK solution is still in development, it provides a promising opportunity in the nearby future and would greatly benefit the overall project.

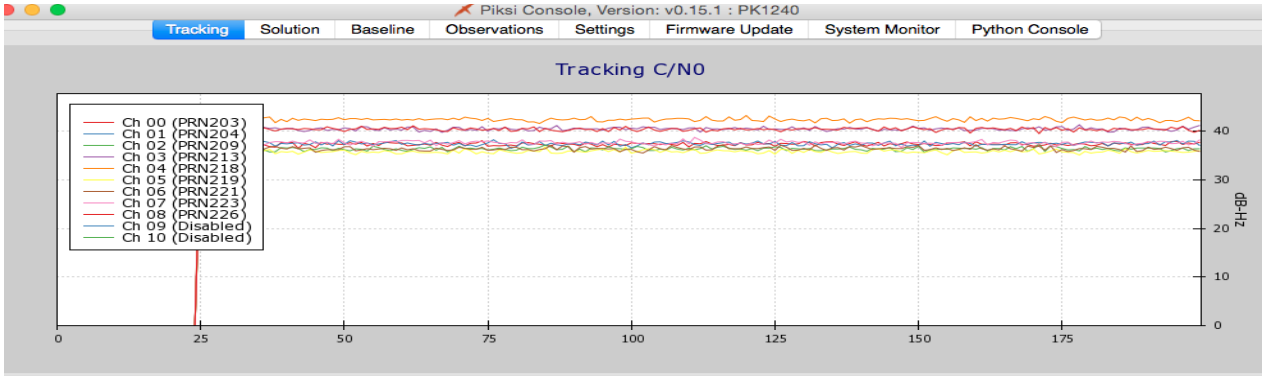


RTK Testing

After we received the Piksi modules, the next task was to install the firmware and console to run them. This proved to be a difficult because of major firmware installation issues that occurred since the Piksi compiler team was in the process of making a new installer. Files and folders got moved around, and after being incredibly active on the Piksi user forums, we finally located the files through the github open repository and posted an incredibly detailed installation tutorial for Mac OSX, Windows 7, Ubuntu, Debian Linux, and Raspberry Pi for users who also had difficulty installing the console. After the console was installed, we updated the Piksi hardware's firmware and plugged in the modules. The first few steps in getting a satellite connection was to run a simulation of the Piksi to make sure the basic parts of the modules are working. Below in Figure 1. is a screenshot of the the satellite gain versus time, with the different colors being different satellites. As you can see they all have a high gain and are connected to most of the satellites.

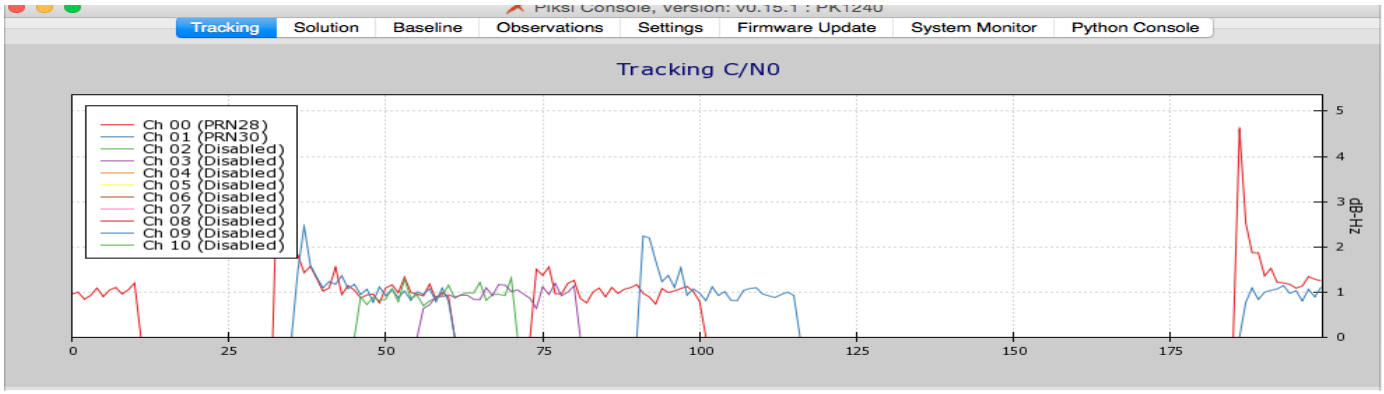
Sam Hessenauer
Ryan Kral
Corbin Adelman

Figure 1: Satellite DB Gain vs. Time - Simulation



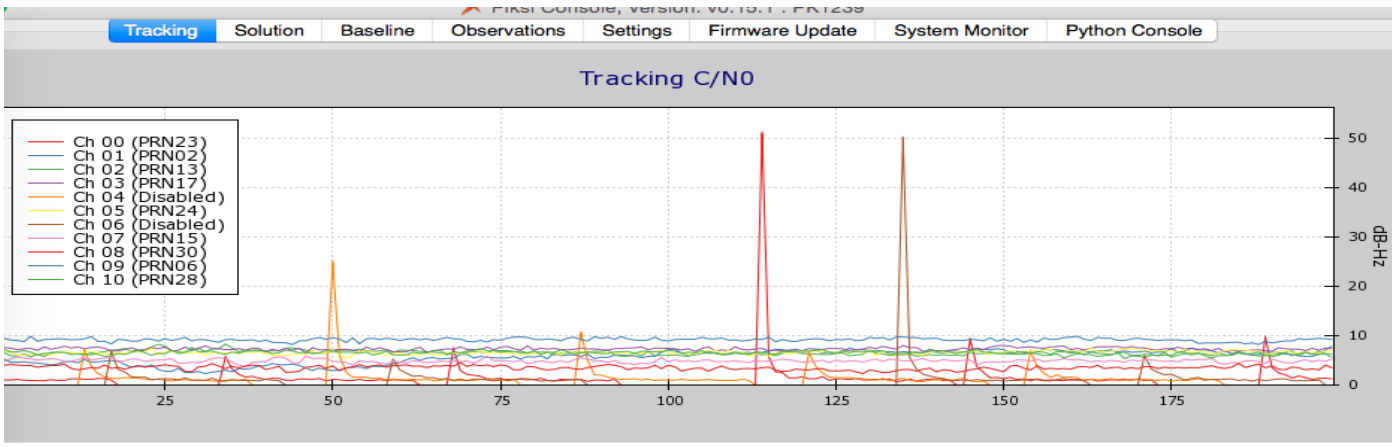
After the simulation was successful, we started recording the simulated data which was in a dated. CSV file in the console folder which contained timestamp, longitude, latitude, and altitude information. This was the data we needed to start the post processing code which will be discussed in the next section.
At the time, we were not given the external antennas for the modules, and struggled even receiving 2 satellite signals as shown below in Figure 2, after extensive research we attached the new antennas but there was improvement, but not nearly as much as what the simulation showed it should be.

Figure 2: Satellite DB Gain vs. Time - Actual Data



After such bad results, we decided to use a metal bench as a ground plane for our antenna because of feedback we received on the Piksi user's forum with using the newly acquired external antennas. This saw a more steady increase in data but still with a low gain. Shown below in Figure 3, we now have about 9 satellites connected, but the Db gain is still not as expected as shown in the simulation.

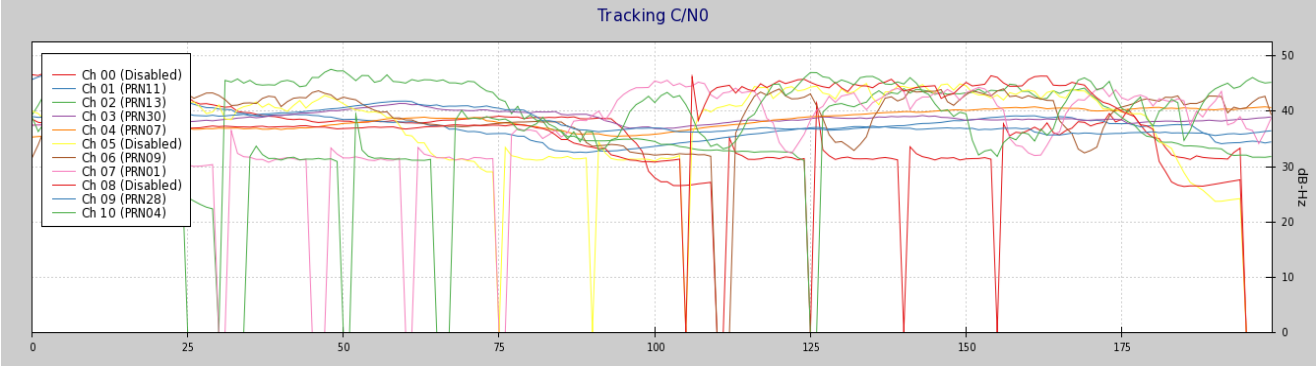
Figure 3: Satellite dB Gain vs. Time - With Bench Ground Plane and External Antenna



Sam Hessenauer
Ryan Kral
Corbin Adelman

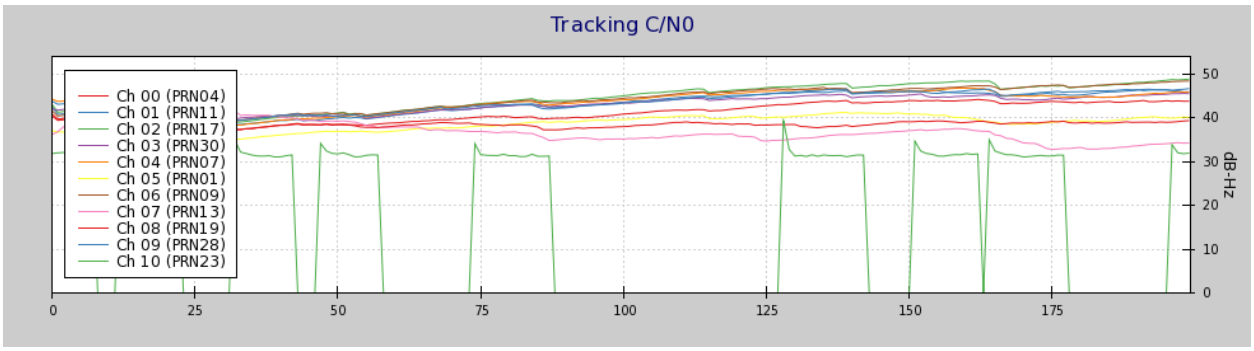
Each of these tests take a lot of time, most of the time spent on this project was attempting for the Piksi modules to connect to satellites, stabilize and then connect to each other. After many different combinations of weather and time of day, we did not achieve over a steady gain of about 10dB until the new Piksi Hardware Firmware version 16.0 was released. Upon installing the new firmware, we had an immediate major change response from the modules and is shown below in Figure 4, but without using any conductive ground plane surface.

Figure 4: Satellite DB gain vs. Time Firmware v16.0 without antenna boosting surface



This incredibly module response to the new firmware brought us new hope when almost all was lost. Next in order to try to get a more steady satellite lock, we decided to use the roof of Ryan's car as our ground plane because of its large surface area and highly conductive surface. Below in Figure 5, we can see an immense increase in reliability, gain, and stabilization in the Piksi Console

Figure 5: Satellite DB gain vs. Time Firmware v16.0 with Ryan's Car Roof as Ground Plane



Now that we had an ideal response from the Piksi Modules, we set out to try to get 'Fixed RTK' between the two modules, which would allow for the precise geolocated data. Each RTK lock would initially range between 15-22 minutes, and would immediately convert into a 'Float' when the rover started to move. This was our biggest issue and caused a lot of time towards our project. Waiting for the Piksi modules to initialize and then sync to troubleshoot Fixed RTK was a long and arduous process. Below in Figure 6, the base station had a static Fixed RTK connection to the rover. On the left under 'Mode', it is set to Fixed RTK, and the graph on the right is how close the base and rover's position is.

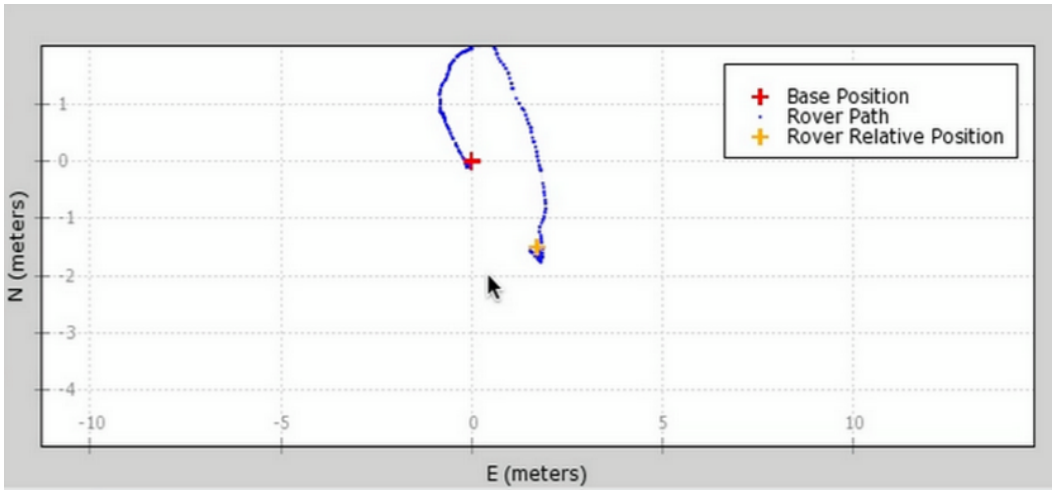
Sam Hessenauer
Ryan Kral
Corbin Adelman

Above is Figure 6: Base and Rover Position Map shows High Accuracy and Fixed RTK

Our next milestone was to get fixed RTK and hold it while we attempted to make a circle. After many hours of troubleshooting, and creating a Rover Rod to facilitate the rover's ground plane, we achieved Fixed RTK and got it tracking as we walked around the car. Below in Figure 7, you can see that we did finally get Fixed RTK working during one of the last weeks of class and we walked around a car.

Figure 7: Base and Rover Position Map Shows Path of Rover during Fixed RTK

By getting fixed RTK and a high precision path, we have made proof of concept that the Piksi does offer RTK positioning for a low cost, but does not have reliable results in varying conditions. Until the Piksi modules and firmware become more accurate and reliable, we do not recommend using this system.



Software Development and Integration

While the Piksi modules were being tested in the field, we took the generated CSV files from the simulated data and used it as a model for our software integration. The outputted data has a timestamp and its correlating longitude, latitude, and altitude in a listed form. In order to integrate the data with the already used system in place for the Radio Collar Tracking Project, we needed to create a C program that could be included in the post processing code that would allow access to the CSV file. The CSV first had to be found and imported to the C file, and then filtered so each of the values were stored in a different variable. For that, we created a python script that would take care of the parsing and storing. Next, we created the C program needed by the Radio Collar Tracking code in order to be able to call a function called getLocation( ) where the program would pass the time and be able to get back the exact longitude, latitude, and altitude at that specific moment.

When both the Python Script and the C program are imported into the C code running the main post-processing for the Radio Collar Tracking software, it allows for a seamless integration of data into their current system. Although we were getting questionable results from the Piksi modules, our main goal was to create a software integration system so that when the Piksi modules become more reliable and make more progress, it would be an incredibly easy integration into the current system. The goal was to make it so all one would need to do was to update the Piksi modules to the new firmware in the future and be able to run it on our system. By creating an interchangeable system, it would allow for easy to use and effortless integration for when Piksi technology became more accurate and reliable.

Sam Hessenauer
Ryan Kral
Corbin Adelman

Milestones

For this project, we used a set of milestones to keep on track and to prioritize our work tasks when things went wrong. In the table below you can see the milestones as we defined them. The tasks in bold were completed fully. The tasks which are italicized were completed to some extent, but require a bit of explanation. The remaining tasks were not completed as desired.

#	Priority 1 (low) - 5 (high)	Milestone Description	Due Date	Responsible People
1	**5**	Read documentation and get up to speed with the Piksi RTK GPS hardware.	4/21/2015	Corbin, Ryan, Sam
2	**4**	Demonstrate that the RTK GPS hardware works with two laptops in the field.	5/5/2015	Corbin, Ryan, Sam
2.5	*5*	*Demonstrate stable Fixed RTK lock with the introduction of a survey stick*	*5/19/2015*	*Corbin, Ryan, Sam*
3	*5*	*Integrate the RTK GPS in with the radio collar tracker hardware/software.*	*5/19/2015*	*Corbin, Ryan, Sam*
4	**4**	Evaluate comparative performance between the existing autopilot GPS and the Static RTK GPS.	5/26/2015	Corbin, Ryan, Sam
5	**3**	Finish RTK GPS callback script to finalize integration. Document our overall results.	6/2/2015	Corbin, Ryan, Sam
6	2	Investigate low cost RTK GPS hardware.	Stretch Goal	Corbin, Ryan, Sam
7	2	Compare the Piksi hardware with the low cost solution.	Stretch Goal	Corbin, Ryan, Sam

Milestones 1, 2, 4, and 5 were completed as expected. Here are details on what we did to meet these milestones:

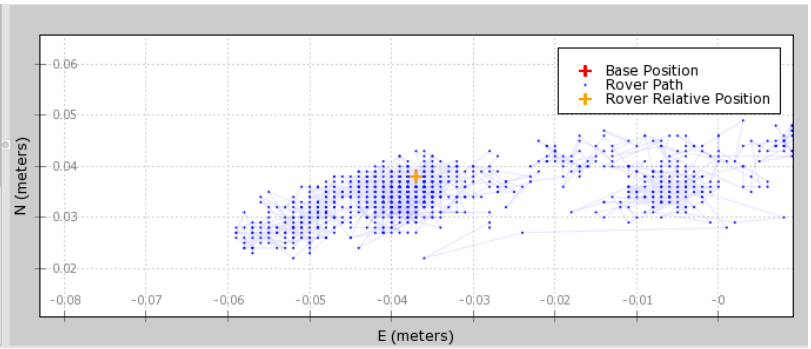
1. To get up to speed with the Piksi RTK GPS hardware and software, we spent some time reading and watching videos about the Piksi product, GPS, and how RTK GPS works. The following videos were watched:
    a. GPS Overview - https://m.youtube.com/watch?v=YjcfmZw23Wg
    b. RTK Overview - https://m.youtube.com/watch?v=zI59yyN7Tyw
    c. Beaglebone Black - https://www.youtube.com/watch?v=ciX08ysl6LE

    The following documents were read:
    a. Piksi Information - http://www.swiftnav.com/piksi.html

b. Piksi User Getting Started Guide - http://docs.swiftnav.com/wiki/Piksi_User_Getting_Started_Guide
c. Piksi Developer Getting Started Guide - http://docs.swiftnav.com/wiki/Piksi_Developer_Getting_Started_Guide
d. Piksi Forums - https://groups.google.com/forum/#!forum/swiftnav-discuss

2. To demonstrate that RTK GPS worked in the field with two laptops, we performed numerous tests on a wide variety of operating systems. We tested with Windows, Mac, and even with both laptops running on Linux. Here is a screenshot taken of the base station's Piksi console software which shows that we were able to achieve a "Fixed RTK" mode:

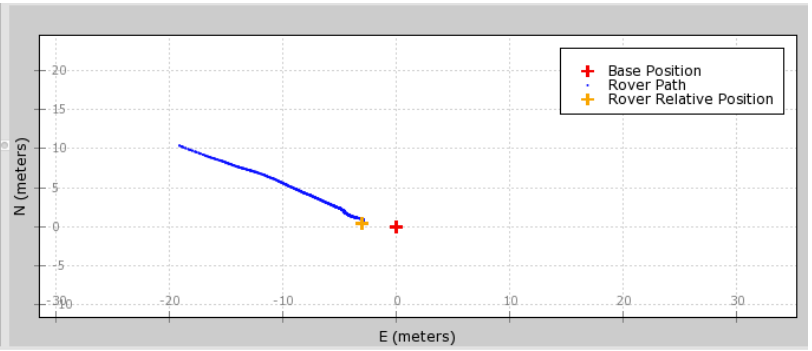
GPS ToW	423792.599725
N	0.038
E	-0.037
D	0.0
Dist.	0.0530377224247
Num. Sats.	8
Flags	0x01
Mode	Fixed RTK
IAR Num....	1



Though this shows nothing regarding the stability of the signal over time, it does show that we gained an RTK lock which was the goal of this milestone and the location of our previous position. For more information, you may view our previous milestone reports which detailed our successes on this milestone in depth.

4. To evaluate the performance of standard GPS with that of RTK GPS, we viewed data provided by the Piksi console. When we were in "Float" mode, we received images like the following:

GPS ToW	427695.199898
N	0.463
E	-3.027
D	0.828
Dist.	3.1721730722
Num. Sats.	7
Flags	0x00
Mode	Float
IAR Num....	1



"Float" mode is equivalent to the data obtained from standard GPS, without obtaining an RTK lock. In the image above, the rover position starts approximately 3-4 m away from the base station, then moves relative to its starting position. However, in reality, the rover was measured at only 10 cms away from the base station initially. When in "Fixed RTK" mode, the measured distance initially was far more accurate. We compared the error between what these graphs showed, and what we actually measured during testing for both standard GPS, and RTK GPS data to generate the following table:

	GPS	**RTK GPS (Piksi)**	**Precision Increase**
Expected Precision	3-4 m	~1.5 cm	~233x
Measured Precision	~4-5 m	~10 cm	~45x

The measured values were averages taken over four test trials conducted over the span of a week under varying environmental conditions. Though our measured precision, or range of error, was not as high as expected, we believe it was because we did not test in perfect weather conditions with

perfect ground planes during optimal times of day. Nevertheless, we were able to show a 45 times increase in precision which definitely makes the case that RTK GPS is worth investing in.

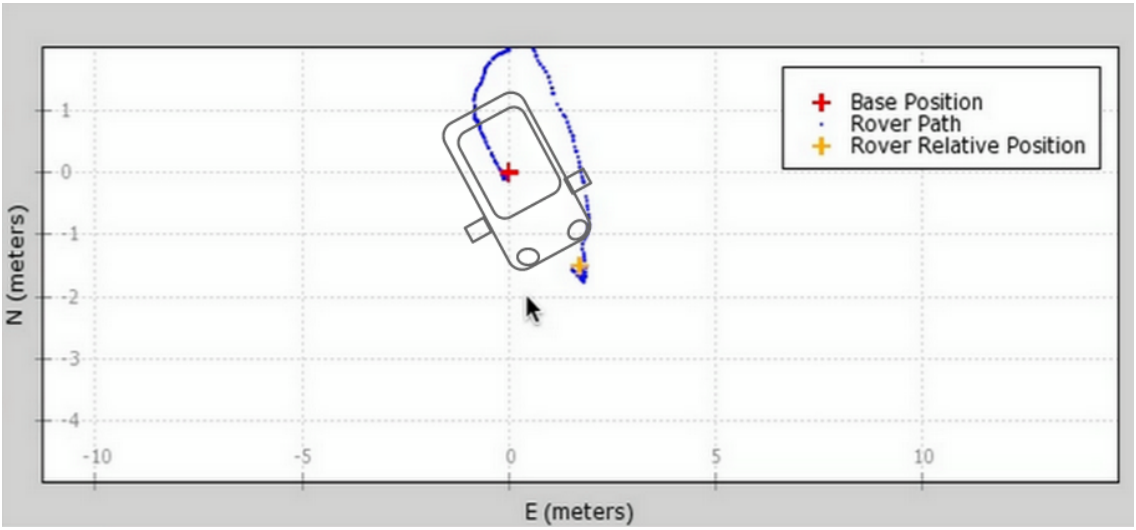
5. To finalize a callback script for integration purposes, we wrote code in Python. The purpose of this code was to allow someone writing Python code, to easily access the latitude, longitude, altitude, and timestamps of data collected from the RTK GPS units. We wrote a python file which can be included easily in other code files which performs the following tasks:
    a. Opens a CSV file which contains all of the RTK GPS data collected during a trial.
    b. Parses the contents of the CSV file into objects which each contain a timestamp, latitude, longitude, and altitude.
    c. Allows access to these objects from other files.

    The Python code which we wrote can easily be converted for use with C or any other programming language as it just needs to open a CSV file and parse it in the same manner as the Python which we developed. This was not difficult software to develop, but we wanted to include it as a milestone to show how easy integration will be once the Piksi hardware is shown to be more stable.

Milestones 2.5 and 3 were completed to some extent, but left some things to be desired in the end. Here are details on the work we did for these milestones:

2.5 To demonstrate stable “fixed RTK,” we developed the rover rod and attempted to find an ideal setup to hold an RTK lock for an extended period of time. The rover rod was made up of a selfie stick, a watch case, and the Piksi unit/GPS antenna. The selfie stick was purchased for $7 at a grocery store and served as a 3 foot extendable rod. We attached a watch case to the end of the stick which had a flat, 3” x 3” metal surface to act as a ground plane. We then stuck the GPS antenna to the watch case and hooked it up to the Piksi unit. This rover rod allowed us to get decent signal strength from the GPS antenna, and also allowed us to move the rover without causing human interference (the rod was extended 2-3 feet from our bodies).

In addition to developing the rover rod, we identified the ideal ground plane for the base station, a car roof. This provided us with the largest possible flat metal surface we could easily access, and gave us the best shot a a stable signal. We performed countless tests using these base and rover setups in both cloudy and sunny weather at various times of day. Here is an example of a test in which we walked around a car:



A slight misstep near the side mirror of the car caused a complete loss of the RTK lock. This one trial seemed to be indicative of the issues with the Piksi units overall. We believe we went above and beyond the actions that should have been required to meet this milestone. However, we were never able to maintain a stable RTK lock. The Piksi firmware went through an update which seemed promising for signal strengths, but even then the slightest hiccup (a gust of wind, car driving past, etc.) could cause our RTK lock to disappear entirely. We are convinced that despite the best possible efforts, the Piksi hardware is just not good enough to hold a stable RTK lock for long

periods of time in its current state. Thus, we felt that we completed as much work as possible on this milestone, but were unsuccessful in achieving our desired result. further information on this can be found in our project video.

3. Integration with the Piksi units into the radio collar tracker overall was an initial goal of the project; however, as we realized that maintaining a stable RTK lock would be an issue, we ruled full integration out of the picture. It is inadvisable to spend time and effort integrating and testing the Piksi units with the quadcopter, when we know that they don't even work reliable in a less intensive environment. Nevertheless, we wanted to do as much as possible towards integration so that in the future this task could be made easier for others. Thus, we developed scripts in Python which will start the Piksi software on the rover automatically, and allow it to collect data. For integration, this script need only be placed in a startup folder or something similar. The, as long as the port numbers are verified once initially, we will have automated the startup processes of the rover. The second integration task is exactly what we completed for milestone 5. We completed code which parses out the data collected from the RTK units and creates useful objects with the data. The remaining work for this step which we did not complete would be to install the Piksi console on the quadcopter and then perform testing. But, as mentioned before, these tasks were not relevant due to the stability issues faced.

Milestones 6 and 7 were not completed all the way. Here are details on why these milestones were not completed, and why that was acceptable for our project:

6. We did investigate an alternative way to get an RTK GPS signal but proved to be needing more development before use. Initially, we thought that once we had the Piksi fully integrated, then we would move on to a second RTK unit. However, because we had so many stability issues with the Piksi units, this was not as thorough as possible. Approximately halfway through the quarter we did decide to start moving on to new hardware, as firmware 15.0 on the Piksi was giving us terrible signal strengths and it was almost impossible to get any sort of RTK lock at all. However, just before we moved to new hardware, Piksi released a firmware update to version 16.0. This was extremely promising because we were now getting the signal strengths that we expected, and we were able to get an RTK lock much more easily. We then decided to continue focusing on the Piksi, rather than ordering entirely new hardware. As described above, the Piksi proved to still be unstable, but we were at least able to give an accurate description of all angles of the Piksi because we chose not to work on milestone 6. Additionally, we listed this milestone as a stretch goal and gave it low priority, so we believe that not completing this is not a reflection of our success with the project overall.
7. Because milestone 6 was not completed, no realistic comparison could be made between the Piksi units and any other RTK GPS units. Because this was a stretch goal, and for the reasons given for milestone 6, this incomplete milestone was deemed acceptable.

In addition to the milestones described above, we worked to complete a set of milestones for class papers, presentations, and video deadlines, but those were not included as all teams faced the same due dates. We also completed additional tasks which were outside the scope of our initial milestones. Specifically, we contributed greatly to the Piksi message boards. We posted messages detailing how to install the Piksi console software on Windows, Mac, Linux, and for the Raspberry Pi. We were also able to learn a ton about GPS from reading other message board posts and from communicating with a few members offline. Though this was not a milestone, we believe it was important work which helped the community surrounding the Piksi as it goes through beta testing.

Conclusion

For this project, the ultimate goal was to help biologists and ecologists save time and effort during the tracking of animals with radio collars. To do this, we felt strongly that integrating RTK GPS into the aerial radio collar tracker would instantly provide a measurable and useful improvement to the system overall. With proper integration, we would be able to convert standard GPS data from meter-grade precision to centimeter-grade precision. This would provide researchers with more accurate data to work with. We chose

to use Piksi RTK GPS units, developed by Swift Navigation, to reach our goal, despite the fact that these units were still in beta testing. This choice was made because these units were readily available at the start of our project and would not cause additional waiting time for new hardware to be purchased and delivered.

The milestones which we set for ourselves ultimately helped lead us as close as possible to full RTK GPS integration. However, along the way we ran into numerous obstacles which prevented us from reaching all of our milestones. Stability of the RTK GPS lock with the Piksi units was incredibly difficult to obtain, despite testing with multiple firmware versions, numerous ground planes, and in a wide variety of environmental conditions. We ultimately believe that the Piksi RTK GPS units are not far enough along in development to be useful in the radio collar tracker system overall. These units have stability issues which would be difficult to work with in a research environment and the benefits are not yet worth the extra work required to ensure valid data. Thus, we recommend to wait until future firmware updates are made by Swift Navigation for the Piksi RTK GPS units before integrating RTK GPS of this form into the system overall.

Despite the issues we faced with the Piksi RTK GPS units, we were still able to accomplish many useful tasks for this project. We showed that achieving a fixed RTK lock with the given hardware is possible, though not stable. Given future firmware updates, we have now documented the most effective way to test these units with a car acting as a ground plane for the base station, and a rover rod serving as the rover setup. We were also able to generate metrics which show that RTK GPS provides a 45 times improvement to the accuracy of standard GPS. This is extremely promising for future attempts at RTK GPS integration with this system. We developed software so that if the Piksi units ever are stable, integration into the larger system will be extremely simple. And finally, we learned a ton about GPS, RTK GPS, and working with products still in beta. We were active on message boards and were ultimately able to convey much of what we learned to the larger community of Piksi developers. Because of these accomplishments, we firmly believe that this project was successful overall.

References

Swift Navigation's Website: http://www.swiftnav.com/
GPS Overview - https://m.youtube.com/watch?v=YjcfmZw23Wg
RTK Overview - https://m.youtube.com/watch?v=zI59yyN7Tyw
Beaglebone Black - https://www.youtube.com/watch?v=ciX08ysl6LE
Piksi Information - http://www.swiftnav.com/piksi.html
Piksi User Getting Started Guide - http://docs.swiftnav.com/wiki/Piksi_User_Getting_Started_Guide
Piksi Developer Getting Started Guide - http://docs.swiftnav.com/wiki/Piksi_Developer_Getting_Started_Guide
Piksi Forums - https://groups.google.com/forum/#!forum/swiftnav-discuss