Multi-Drone Synchronization Using Time Delay Based Commands

Jadyn Chowdhury*, Logan Maue*, Justin Magsino*, and Aiden Stewart*
University of Illinois at Urbana-Champaign, Urbana, IL, 61801, U.S.A.

Stemming from technologies required to perform novel drone light shows as a form of public entertainment, this report aims to show scalable techniques for drone control. This is all encompassing, walking through hardware fixes, software writing, and proper command structure required to perform these tasks. A central computer will be used to control three Crazyflie quadcopters in synchronous motion that is tunable to a song or lights housed on separate units. This synchronous motion is achieved by implementing time delays to the commands that will allow both drones to move at the same global time, through the swarm library. All of this shows how to scale up to larger systems of drones. Additionally, a demonstration of the techniques developed here is shown in a miniature drone show synchronized to music.

I. Nomenclature

p_x	=	position of the drone in the x-direction (m)
p_y	=	position of the drone in the y-direction (m)
p_z	=	position of the drone in the z-direction (m)
ψ	=	yaw
θ	=	pitch
Φ	=	roll
Q	=	weight matrix of the LQR cost function
R	=	weight matrix of the LQR cost function

II. Introduction

A. Opening Statement

The field of multi-drone synchronization has gained significant attention in recent years, with applications spanning entertainment, infrastructure inspection, and environmental monitoring. The ability to control multiple drones in coordinated movements has transformative potential, allowing for dynamic displays, efficient task execution, and creative performances. This report aims to explore a new approach to multi-drone synchronization using time delay-based commands, enabling drones to operate with high precision and consistency while maintaining visual coherence. In this project, three Crazyflie quadcopters will be synchronized using commands sent from a single laptop, an approach that mirrors the setup needed in larger drone systems, where managing each drone from a separate device would be impractical. Controlling multiple drones from a central unit minimizes hardware complexity and enhances operational efficiency, but it also introduces unique synchronization challenges. We anticipate that these challenges will manifest in slight timing discrepancies between drones, and the goal is to address them through carefully calibrated commands and managing of data packets. By synchronizing drones with controlled commands, this project provides a flexible and scalable alternative to centralized or motion-capture-based systems, demonstrating a novel method for multi-drone coordination with applications in both practical and performance contexts. In addition to controlling multiple drones, we also intend to control the lights on the drones to simulate what could be expanded for a light show. By controlling the lights on the drones rather than using a light deck, we're able to utilize measurements from the flow deck rather than having to rely solely on data from the motion capture system.

* Undergraduate, Aerospace Department, Grainger College of Engineering

B. Organization of Report

This report is organized around three main sections: weekly milestones based on the three-week schedule the content for this report was based on, results from the project after completion of the milestones, and a discussion surrounding implementation of the results and future variations that could result from this project. The milestones section is divided into three parts, one for each working week, detailing the goal for the week, the challenges faced, and ultimately the result for each goal. The results section will go over the final demonstration flight, what was expected of it and how the overall performance relates to other factors. Finally, the discussion section talks about what those results could mean for future projects in this field or variations that could have made it into this report.

C. Related Work and Research Contributions

Previous work, such as the centralized control systems explored in *Implementation of Communication Between Two Autonomous Drones* (Scott Bout Et al., Fall 2022) and *Implementation of Formation Flight Between Heterogeneous Drones* (Berg Et al., Fall 2023), has demonstrated the use of centralized intelligence or motion capture to synchronize multiple drones. Additionally, *Enable Communication Between Two Drones to Allow for Autonomous Synchronized Flight* (Amoruso Et al., Fall 2023) investigates synchronized drone swarms, which can replace traditional fireworks displays by choreographing drones in pre-programmed patterns. These studies support our first two milestones and goals, where we establish central control of two Crazyflie drones and synchronize their movements using time delay adjustments. By building on these methods, this project aims to address the synchronization issues that arise when coordinating drones from a single device, particularly focusing on reducing timing discrepancies through precise delay calibration.

Drone-Created Art: from PNG to X, Y, Z (Carrigan Et al., Fall 2023) extends drone use to creating 3D shapes and art, transforming flight paths into visual displays. This work exemplifies the potential for drones to be used creatively, as they respond to precise positioning commands to generate intricate designs. In our project, this research informs Milestone 3, where synchronized drones perform in pre-arranged patterns to explore the aesthetic potential of time delay-based coordination in 3D space achieving similar goals.

Expanding on interactive applications, *Drone Control Response to Sound Frequencies* (Elser Et al., Fall 2023) demonstrates drones reacting to audio cues, converting sound frequencies into physical movements. This research underscores the potential for real-time synchronization with external stimuli, laying the groundwork for future extensions where drones could respond to live music or create sound-reactive visuals, such as a dynamic "soundbar" display.

D. Key Points for Reader Attention

Readers should focus on the project's approach to achieving multi-drone synchronization through time delay-based commands from a single laptop, addressing timing challenges without multiple control devices. Key areas of interest include the calibration of time delays to minimize discrepancies, the flexibility of this setup for scaling to larger drone systems, and the potential applications in synchronized performances and coordinated aerial tasks. All of these rely on the correct implementation of the swarm library as well as using custom-built controllers and observers. There is also an extensive section within the discussion about future improvements to the project and better implementation strategies.

III. Weekly Milestones and Project Process

For the week of November 11th, by the end of the day Friday, the hardware and software were implemented to control two drones through one laptop. This allowed the ability to create and send commands from one central unit to both drones, greatly reducing any possible errors in the synchronization of their movements to come in later weeks. Within this step the aim was to investigate the possible solution of using one radio transmitter on the same laptop. This route took a bit of figuring out for the required hardware configuration, then the logistics of sending out a fly command through the software that was needed to develop for correct implementation. The largest technical difficulty in successfully completing this project was assumed to be ensuring that the drones are receiving commands at the same time. To go about this process, there were a couple of possible implementations that were explored. By working with the software, the easier solution was finding a way to simultaneously send out commands from the computer itself. Another possibility that was investigated was using the mocap system to aid in determining how and when the commands are received. The expected results of working and synchronized commands for each drone will set up for the weekly milestone where there will be implementation for the time delay into drone commands.

Initially the hardware would have to be configured to make this project possible. Initial ideas revolved completely around central computers for all required communication as this would be the easiest way for system-wide communication. This is also how professional light shows control their drones, using one computer with multiple radio transmitters that talk to multiple drones themselves. The first idea was to use multiple radio transmitters to talk

to each drone, but limitations in the client quickly made that not an option. When trying to implement this, the drone's client on the computer would simply choose its preferred radio and not switch off that transmitter. This caused the problem of repeating errors of "hardware already in use," as it would try to switch frequencies to talk to another drone. Stemming from this, the investigation into using one radio was started. Changing the individual drone's Uniform Resource Identifier (URI) was going to be the solution, as this was the unique "password" for lack of a better term to talk to a specific drone. It is made up of two parts, the radio frequency and an address that was unique to each drone. But after editing them, all three drones were on the same radio frequency with varying addresses that were easy to distinguish between each other. Below are what was chosen:

radio://0/69/2M/E7E7E7E701
radio://0/69/2M/E7E7E7E702
radio://0/69/2M/E7E7E7E703

Here radio designates the tool the client would use to call send out the commands, 69 is the radio frequency that was assigned in lab section, and the last digit is what signifies a unique drone. An added benefit of doing this was that there would be no more interference with frequencies close to that as the address would be different and other computers would not be able to communicate with the drones. Changing the drone's URI is straightforward and can be done swiftly with the built in cfclient in the software package that comes with the Crazyflie quadrotors, in fact it is already talked about in the lab manual for AE483.

This configuration proved to be useful for finding a software solution to giving the drones commands from one central location. With the current format for sending drone commands through the local drone client within the flight.py code, the radio transmitter would have to alternate between each URI and there would be a time delay incurred in the processing power plus the frequency at which the actual signal would be sent. Obviously not ideal, a new class of controlling the drones would need to be used. This led to the Crazyflie swarm library which already had a time delay function built in, including the use of a multi-processing configuration to send the drone commands at close to simultaneous instances, or at least close enough to not notice. However, this software package introduces a new class of controlling drones, the commander class. This proved to be a great steppingstone for the project as it needed a reliable signal sending method with low latency and consistent hardware configuration for scalability.

Next, for the week of November 18th, by the end of the day Friday, the implementation of the drone swarm's observers was completed to integrate motion capture data. In longer flights, drones that do not implement motion capture data into the observer will eventually exhibit drift in their flight commands, as they would be operating under their own relative information. By using mocap data, the drones can assuredly employ data from the objective world frame. We will have to address integrating the separate origin points of the different drones into their flight plans, as the Qualisys world frame is relied upon. Another challenge will be making sure that the correct data packets are sent, to reliably get their information from the mocap data. Working with these data packets will be a significant challenge, and ensuring the correct information is sent to the correct drone will require reworking of the original code base, including making changes to cfclient. After this milestone, the drones were ready to be flown together precisely and accurately, implementing data from the mocap.

So far in the laboratory assignments, the drones have been flying with limited information feedback from the actual world, this has led to some issues with non-precise control movements. This is not something that might have catastrophic effects for flying multiple drones at proximity to each other. To combat this, implementing motion-capture data to drones will have to be done. This was accomplished by changing the system of governing equations for the observable states within the drone's firmware. It went from six observable states to nine, a huge improvement and makes every state measurable. The change is due to there now being an extra information deck on the drones, the active marker deck. Before inputs could only come from the flow deck which would give height and lateral velocity data. But with the active marker deck, the drone knows its exact location in the room, along with its exact orientation with respect to space and its roll, pitch, and yaw angles. Now, each drone does have its own unique observer, as each one needs to be tuned to their own unique constants of motion. Implementing the custom controller and observer for each drone was kind of complicated with the swarm library as the commander class had to be edited for a Boolean to be passed through to choose whether the custom firmware was active or the default firmware was used.

As previously mentioned, there was an issue with the implementation of the new library and class for controlling the drone. This, of course, would extend to the sending of data packets from the motion-capture system as editing the command package within commander would entail a lot of software changes when a simpler solution exists. There is a way to send package information embedded into the cfclient package. This is mostly used for extraneous information like light control or outside observers on the laptop that does not send with the command packages. With this the central command unit can collect mocap data, package it into a six variable data package for the drone to receive between flight commands. This is straightforward and involves minimal changes to the firmware on the drone and a function within the flight code. First the header file, controller_ae483.h, needs to be edited to include six variables instead of three. This is done within the struct AE483Data function. Then the controller needs to be edited to take the six variables and log them within its data grab while in flight. Finally, the flight code needs to take the measurements

from the motion-capture system, package it into a data packet, and send that data packet to the drone. To reiterate, this occurs outside of the commands data packet unlike the default set up from lab. The implication of doing it this way is that we are sending out six data packets in a row as opposed to just three or even just one if the drone is flying according to its default configuration.

Finally, for the week of December 2nd, by the end of the day Friday, we executed a performance where the drones fly in various sequences and directions. To accomplish this performance, we commanded the drones to fly synchronously with each other, along with continuing to integrate the improved observers. The goal is to create an aesthetically pleasing, intricate, and interesting flight plan. The thought is to mimic some of the flight paths and maneuvers that would be required in larger drone shows. In a real drone show, thousands of drones need to fly in formation along specified predetermined flight paths. If able, we may see if we can borrow a third drone from another classmate and even make a flight sequence with three drones together. Additionally, we demonstrated control of the inbuilt lights on the drones themselves. Through this proof of concept, we showed how this could potentially be scaled up one day in the future.

During the initial flight tests with the three drones, the controller for one drone was not smooth enough to handle the commands as they required more forgiveness for errors. After this was done, figuring out the flight paths of all three drones was more difficult than expected. They had a few restrictions, like not flying them over one another as this would cause the z-ranger on the flow deck to send a bad input to the drone's observer resulting in a crash. Additionally, they were not allowed to be within 5 cm of each other as this would cause a collision and two drones would crash. To simulate the drone show movements, there were synchronous movements as well as movements in different directions as this would simulate changing shape or effects. Without a light deck the drones seemed to not be in the spirit of being drone show ready, to counter this the command packet was edited to include a command that would change the lights on the drone to display in orange and blue, not blinking too.

IV. Results of the Final Flight Demonstration

For the final flight demonstration, all 3 drones were given a set of flight commands to fly in different directions and heights at the same time. To prevent collisions and the drones from overlapping with one another and causing a crash, the drones were set at a distance about 1 meter from one another and given varying commands in different directions so that they wouldn't interact with one another as seen in Fig. 1. The performance results for the 3 drones were recorded upon completion of the sequence.

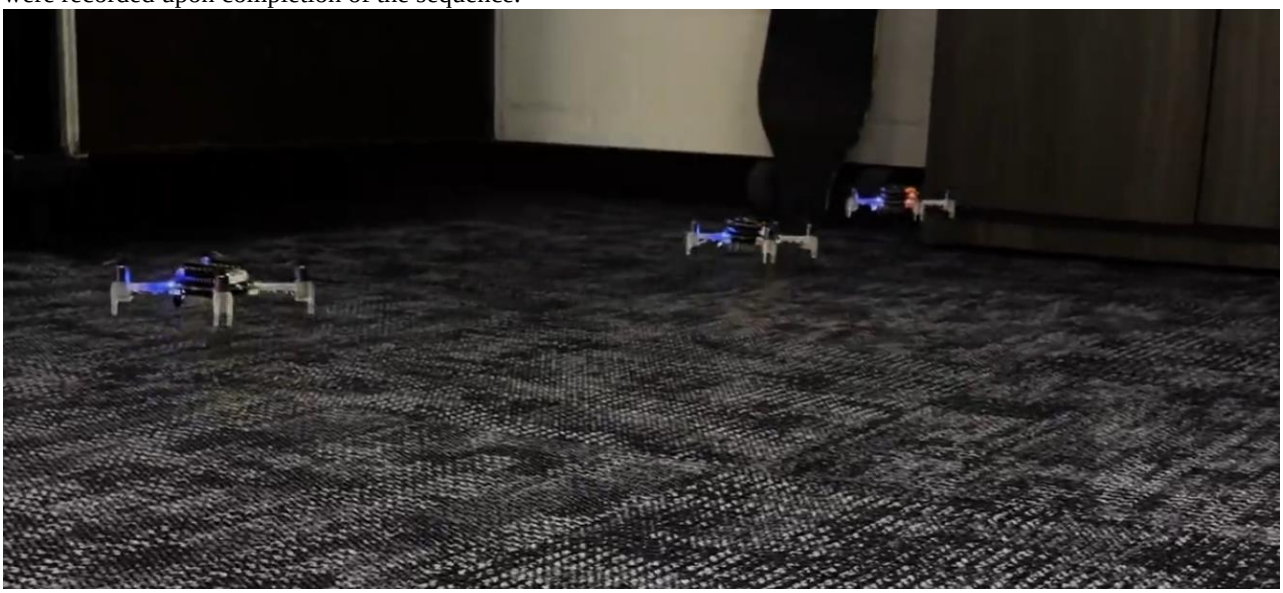

Fig. 1 Picture of the setup of all 3 drones at different distances

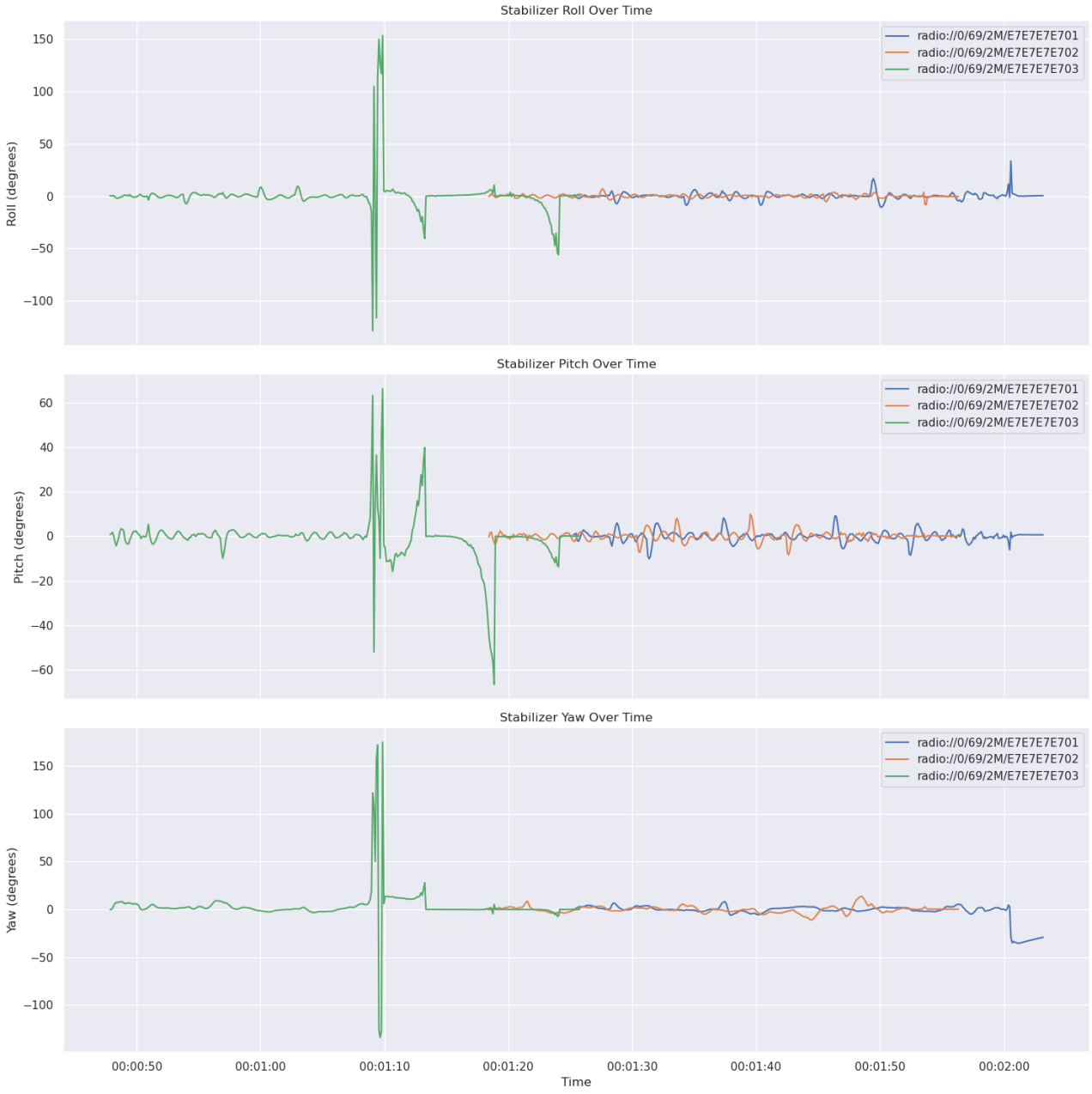

Fig. 2 Plot of the drone's orientation with respect to time

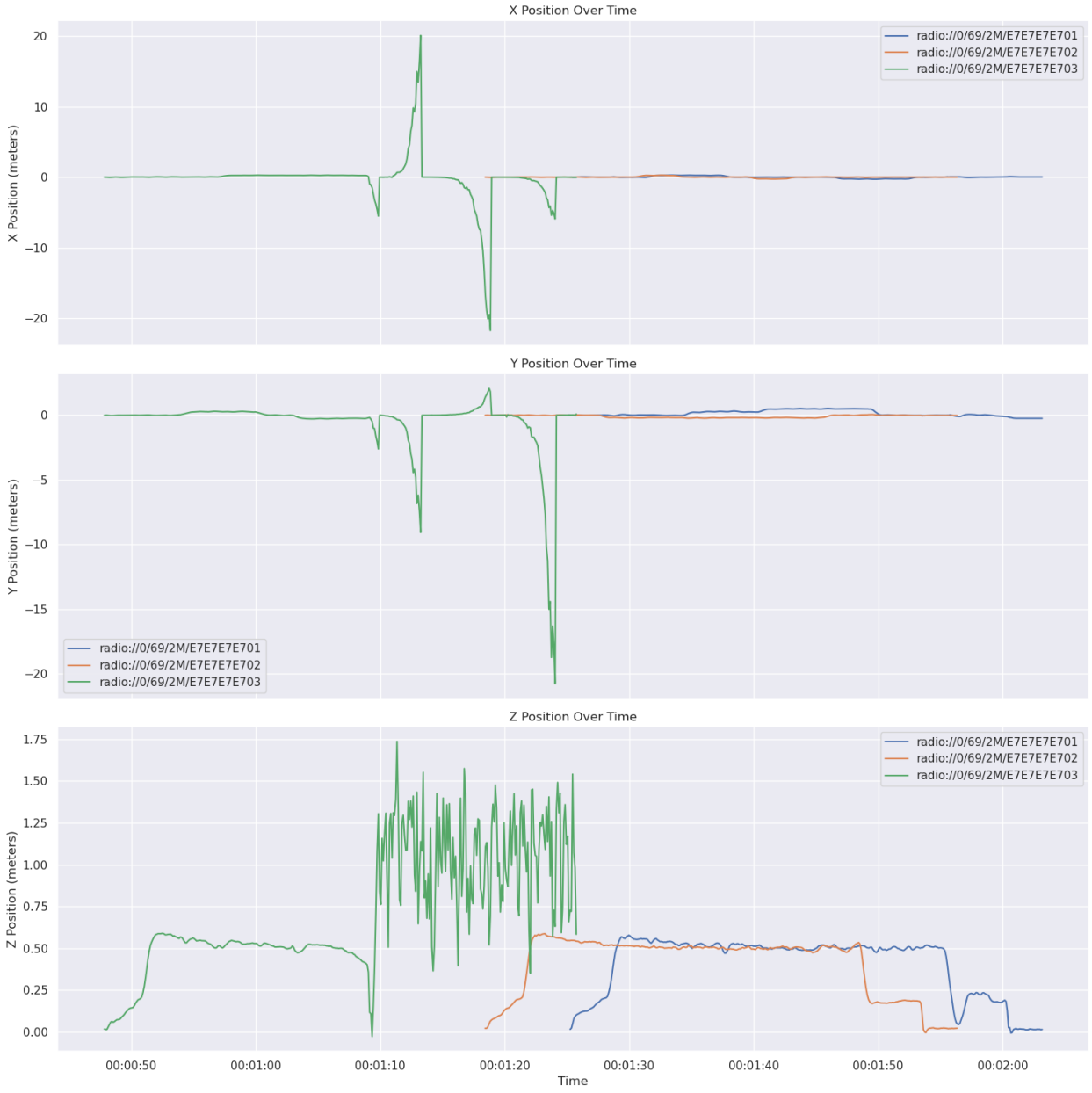

Fig. 3 Plot of the drone's position with respect to time

As seen in Fig. 2 and Fig. 3, at a certain point during the drone's flight time, the 3rd drone failed and crashed while drones 1 and 2 accomplished their respective sequences. The plots of the positions and orientations of the drones shows that initially all 3 drones were indeed accomplishing the movements we had intended them to perform, however, during a certain command, drone 3 failed and crashed while the remaining 2 continued. Upon further investigation and research, our group deduced that a potential reason for the 3rd drone crashing mid-flight is due to the radio not having enough bandwidth to continue sending 3 simultaneous commands to each of the 3 drones. Prior to the final test, we encountered this issue with the drone that was crashing changing during each test we conducted. After doing more research on swarm and debugging our sequences, it was determined that the reason a crash occurs is most likely due to the lack of proper bandwidth. As a result, moving forward we decided to remove the 3rd drone that crashed mid-flight from our plots to properly showcase the other 2 drone's results. While the 3rd drone was removed for the plots, we were able to successfully showcase that flying 3 drones from one radio is possible.

Fig. 4 Plot of the 2 drone's orientation with respect to time

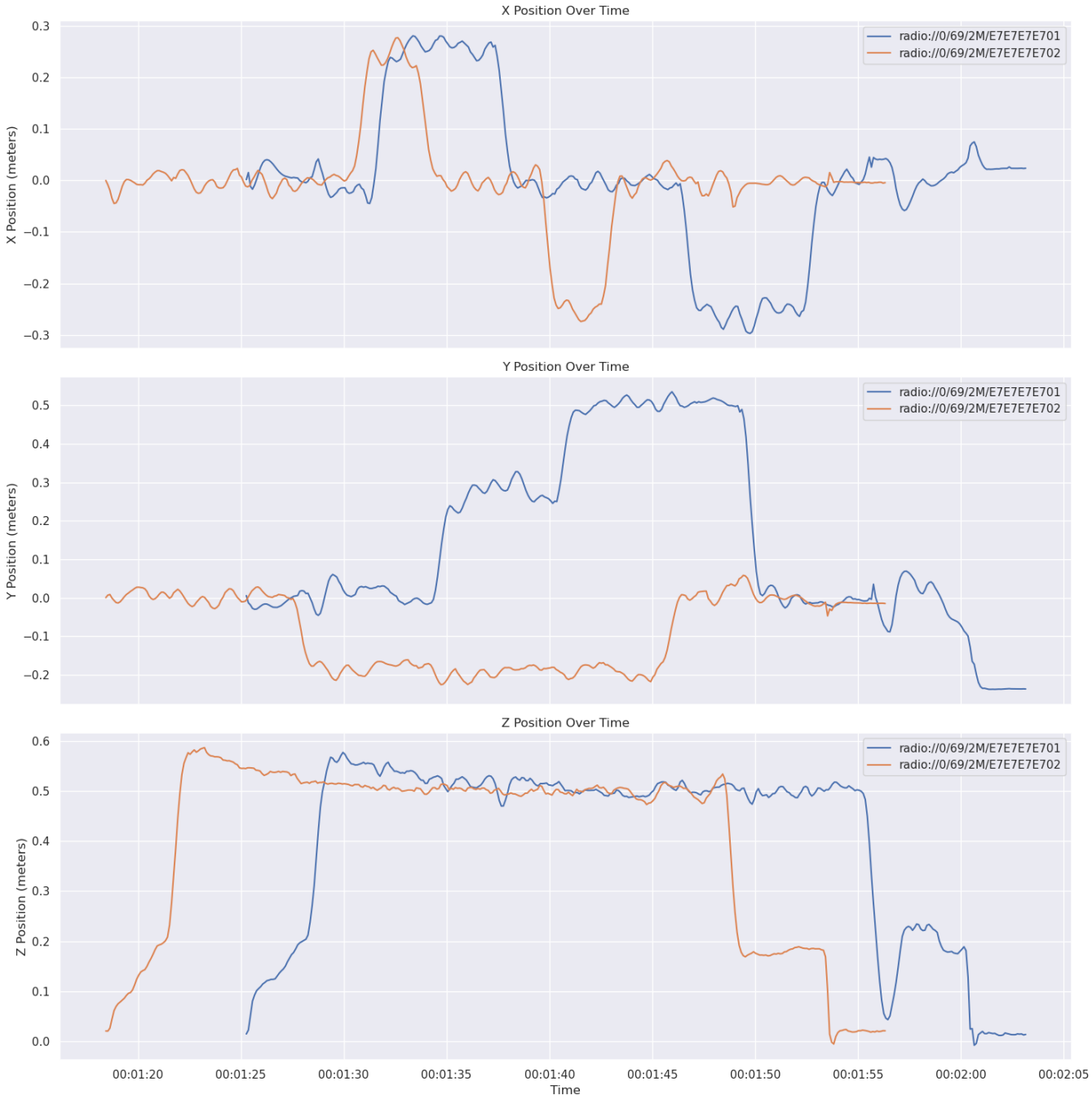

Fig. 5 Plot of the 2 drone's position with respect to time

Fig. 4 and Fig. 5 display the orientation and position plots of the 2 drones which accomplished the sequences without crashing. With the 3rd drone which crashed removed, it can be more easily observed for both drones the movements and stability. In Fig. 4, it can be seen that for each of the 3 directions: ψ, θ, and Φ, the drones are relatively stable. The implementation of our custom observer and controller is visible as the orientation graphs visualize the drones trying to maintain balance to prevent it from completely rotating in one direction as opposed to the consistency of the default observer and controller. After testing different weights for the Q and R matrices, we were successfully able to select values which allowed all 3 drones to fly relatively stable while utilizing our custom observer and controller. In Fig. 5, it shows the movements of the drones in the p_x, p_y, and p_z directions. Through these 3 graphs, it is worth noting how both drones can execute the same commands and fly to the same height or distance at different times and end the sequence in the same place, with only drone 1 having a little drift in the y direction. This showcases the use of the

motion capture system to allow the drones to know their position and execute the movements based on their current placement.

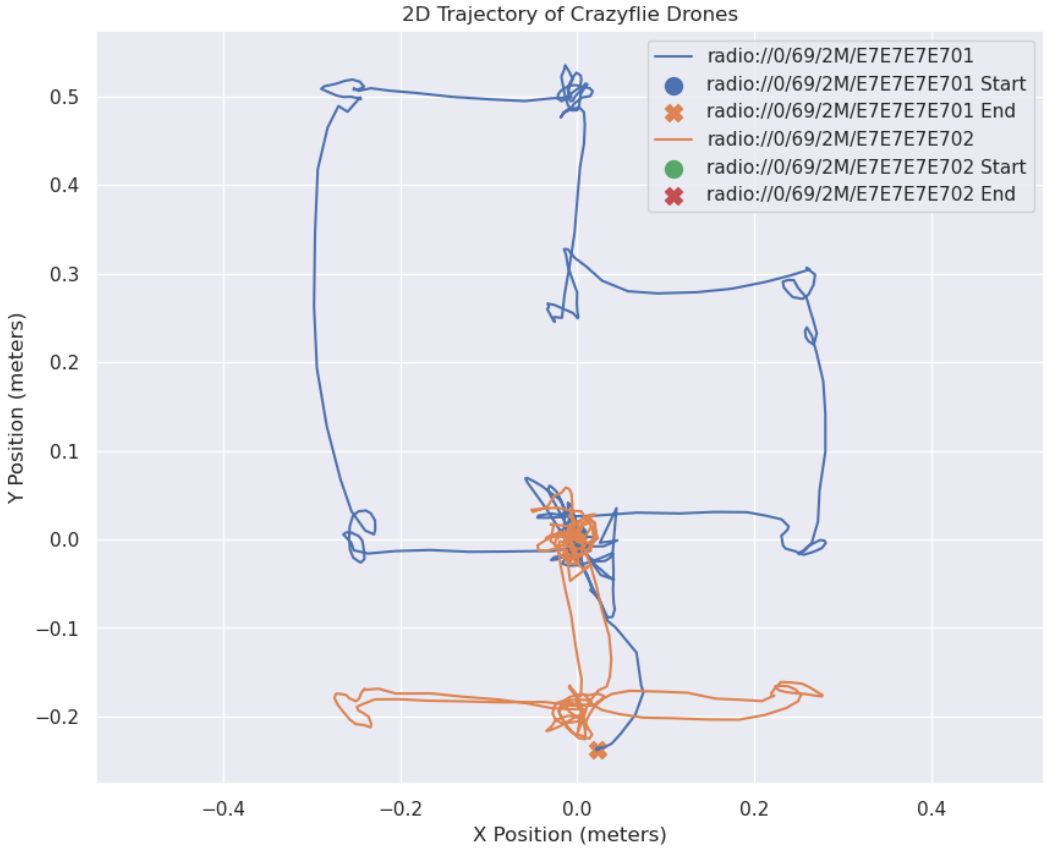

Fig. 6 Plot of the 2D trajectory for 2 drones

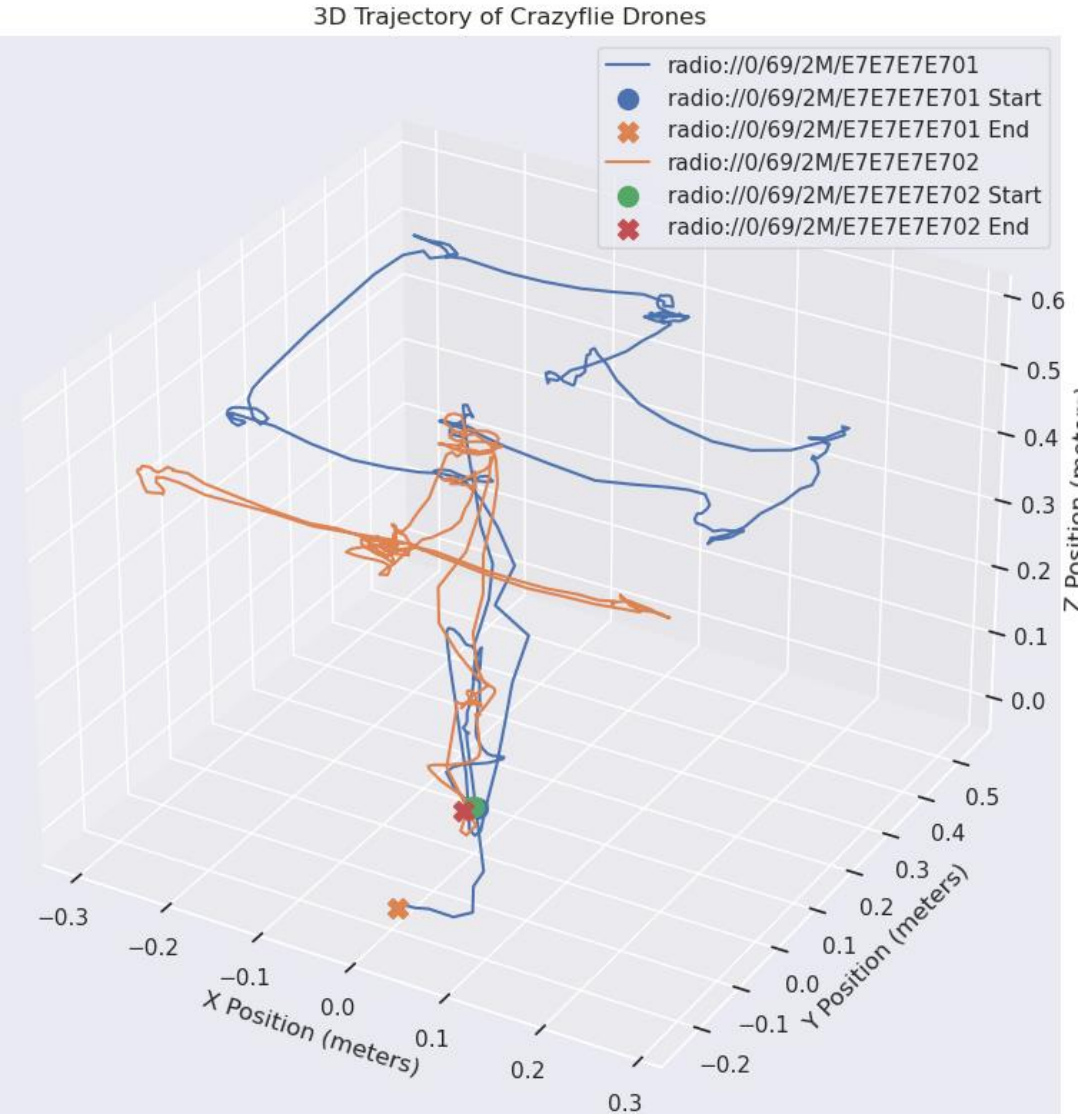

Fig. 7 Plot of the 3D trajectory for 2 drones

By removing the drone that crashed mid-flight, the trajectories of the 2 drones that successfully finished the sequences can be observed in Fig. 6 and Fig. 7. As seen in both the 2D and 3D trajectories, both drones were able to accomplish their movements with a little drift from their initial positions. Through the use of the motion capture system for the drone's position, both drones were able to complete their actions with minimal drift and swaying. The reliance on the motion capture system along with the sequence commands greatly contributed to the drones not encountering one another which would create a series of problems, most often crashes and the breaking of propellers.

Fig. 8 All 3 drones with the lights on

In addition to the drones flying simultaneously, they were also able to successfully fly with the lights inputted. Rather than using a light deck, a command was used to have the drones display different colored lights as seen in Fig. 8. This proved successful as after a few light checks, the drones flew simultaneously and with the intended lights during the duration of the sequences without blinking. Although the colors chosen were blue and orange, the colors could be changed for an actual drone performance to further entertain the audience.

V. Discussion and Future Consideration

If we had wanted to, and had access to three more drones, another further step we could have relatively easily included would have been having two computers with one radio transmitter, each flying three drones. The easy method of synchronization could have been simply running the programs at the same time, by clicking the enter button on both computers together. Still, since there would be two computers, the flights may not have been perfectly working together. To address this issue, we could have accessed the world time clock from a package and ensured that both computers started executing commands at the same time from the world clock.

At the center of this project is the goal that the results of this project could be expanded to fly hundreds or thousands of drones together in configuration, ideally with one control point. Another idea that had been considered by our group was having multiple computers each with a different radio each separately controlling one drone. The implementation of this idea for two drones could have been relatively simple, as it would not have required many changes to the existing code base. We would have needed to find some way to make sure that they both flew at the same time and started executing orders at the same time. Another possible consequence of this could be that although the two computers started flying the drones at the same time, the later commands may have started to execute with a slight difference in time. That process could have kept occurring until there was a noticeable discrepancy between drone flight commands. A possible fix to this issue could once again be going with the world clock package and syncing to that time. Ultimately, though, having one laptop and one radio for each drone was discarded. The purpose of this project was to demonstrate the possibility of being able to scale the two drones up to many, many more drones. Needing one thousand laptops to fly a drone swarm of one thousand drones is unnecessarily complicated. Alternatively, instead of using full-sized computers to operate each drone with one radio, smaller Raspberry Pi's could have been programmed to be able to handle the execution of flight commands to each drone, greatly decreasing the unwieldiness of physical large laptops.

We found that most demonstrations of large drone configurations typically employed multiple radios, with each radio controlling multiple drones. The maximum number of drones we were able to successfully fly in sequence was three. We believe that this threshold is related to the communication data rate of the radio that we were given. Though more than sufficient to send packets for one drone, the radio does have an upper limit for how much data can be sent.

If we could access a radio with a higher communication data rate, then we would be able to determine if this factor was the limiting one.

Even with only three drones, there were multiple times where the commands sent to each drone would end up not going through correctly, as a drone could simply end up randomly crashing. Each of the drones individually could fly the prespecified flight path, and we continuously checked the components to determine if that was a source of failure as well. Simply put, flying all three drones without an erroneous command being processed by any one of the three drones was difficult. Another possible implementation would be working with the Swarm library source code and improving the thread management of code itself. While the communication data rate itself is a limiting factor for how many drones can be controlled, how the data is formatted and sent is also essential. If less data needed to be sent to each drone at a time, then the number of drones and the performance of the drones in flight could be improved.

For added visual effects that could eventually be added later, there already exists two other decks that can be bought from Bitcraze. The Buzzer deck makes buzzing tones, similar to a phone's buzzing noises. Frankly, the group was not as impressed by the Buzzer deck, as the sounds were described as shrill and annoying. The Buzzer deck's noise is reminiscent of attempting to study in one's room while there is an active construction site outside. Another alternative to adding sound is using an external speaker and syncing it in some way with the flight commands for the drones. The LED-ring deck has entertaining lights that can be individually programmed, along with headlights facing forward that are also programmable. Flying the drones with the LED-ring decks in the dark is another way that the drones can be more visually entertaining, as the lights especially stand out in the dark. Another possibility is using external party, such as laser pointers, that would be synced to be change with the flight. There already exist market options where laser lights can be made to change with changes in sound/music. So, if music or specific tones were synced with the drones in flight, light controls that are affected by sound could be used in conjunction, and a practical application in the drone lab could be created.

For the best use of the LED-ring deck, it would have to be attached to the bottom of the drone. Unfortunately, adding this deck to the bottom of the drone would mean it would have to be under the Z-ranger deck. The LED-ring deck would not be able to be put over the Z-ranger deck because that would block the laser range finder. Even if the Z-ranger deck was put over the Buzzer deck, then the power supply being drained faster, and the extra total weight would have to be taken into consideration. The adding or switching out of other decks could result in the weight distribution changing. For optimal results, the moments of inertia calculations may have to be retabulated. However, as was done when we would switch out the mocap marker deck with the base deck, we would keep the same equations. So, the same equations could still have a chance of functioning, but the controller may be operating under slightly faulty conditions. Additionally, the removal of the Z-ranger deck would necessitate the editing of the equations of motion, as that piece of information would be lost. Then, after the changing of that state being observable in the state space model was accounted for, new LQR equations would have to be calculated. Finally, those LQR equations would have to be implemented into the C code for the drone to fly.

An additional path that could be explored would be sending the full flight path to the drones all at the same time. One way this could have possibly been done would be by implementing a fully hardcoded flight path into the C code. Considerations would have to be made for whether all this information could have been stored in the drone itself. Then, the flight file could have simply told all three drones to start running through their hardcoded flight sequences at the same time.

VI. Conclusion

Our group was able to meet the requirements for our project. For our final test flight, we demonstrated that we were capable of not only flying two drones in step with each other, but also that we were able to add a third drone, flying all three together in a pre-planned sequence. Additionally, the demonstration at the beginning of the flight showed that we were able to control the drones' inbuilt lights. We made a visually pleasing sequence of movements that were interesting to watch. We laid out a plan that could eventually be expanded into larger Crazyflie drone shows using our custom controllers and observers.

VII.Acknowledgments

We would like to acknowledge and thank the teaching team along with Dr. Bretl for providing us with the knowledge and support to accomplish this project. Specifically, we would like to thank our individual teaching assistants for their time during and outside of their assigned laboratory hours, Rihan Aaron Dsilva and Hongrui Zhao.

Special thanks to Destiny Fawley for being flexible with us and opening the lab early before office hours for our flight demonstration.

VIII.References

[1] Scott Bout, Konstantinos Morakalis, William O'Connor, and Aayush Patel, "Implementation Of Communication Between Two Autonomous Drones", AE483 Final Project Report, University of Illinois at Urbana-Champaign, Fall 2022.

[2] Calvin Berg, Fuad Samhouri, Leela Herena and Pranav Banerjee, "Implementation of Formation Flight Between Heterogeneous Drones", AE483 Final Project Report, University of Illinois at Urbana-Champaign, Fall 2023.

[3] Jake Amoruso, Kameron Jackson, Nick O'Brien, and Arturo Saucedo, "Enable Communication Between Two Drones to Allow for Autonomous Synchronized Flight", AE483 Final Project Report, University of Illinois at Urbana-Champaign, Fall 2023.

[4] Caleb Carrigan, Anna Hylbert, Eric Monson, and Matt Taylor, "Drone-Created Art: from PNG to X,Y,Z", AE483 Final Project Report, University of Illinois at Urbana-Champaign, Fall 2023.

[5] Chloe Elser, Hsien-Kuei Chang, Devanshi Narayan and Talon Gay, "Drone Control Response to Sound Frequencies", AE483 Final Project Report, University of Illinois at Urbana-Champaign, Fall 2023.

IX. Appendix

Data and code were submitted as an attachment to this report and are available upon request.

E. Appendix A.1: Flight Code

F. Appendix A.2: Flight Analysis

G. Appendix B: Firmware (C)