AVIATA: Autonomous Vehicle Infinite Air Time Apparatus

Axel Malahieude, David Thorne, Ryan Nemiroff, Bhrugu Mallajosyula, Yuchen Yao, James Tseng, Chirag Singh, Eric Wong, Willy Teav, Puneet Nayyar, Zehao Rong, Daniel Lin, Ziyi Peng, Jeff D. Eldredge

***Abstract*—We propose a system for indefinite flight time consisting of a central structure lifted by 8 autonomous multirotor vehicles. Vehicles can be attached and detached using multi-stage computer vision software and a custom mount. Decentralized communications and controls software allows the system to remain airborne even while replacing vehicles. Testing has shown the viability of the system, with successful static docking, position hold, and communications tests verifying the functionality of each individual subsystem. Future testing is to be conducted on the integration of the controls and docking subsystems. Subsequent work will focus on autonomy of the vehicle replacement process.**

I. Introduction

With the increased use of unmanned aerial vehicles (UAVs) for disaster response, package delivery, and environmental monitoring applications, the need for systems supporting payloads for extended flight time has increased drastically. We propose a flexible, low-cost solution to this problem, known as the Autonomous Vehicle Infinite Air Time Apparatus (AVIATA). AVIATA consists of a central frame lifted by 8 multirotor vehicles (drones), each of which can be detached and replaced without landing the central frame. This solution provides indefinite flight time by replacing low-battery drones with fully charged drones, keeping the drones charged without grounding the payload.

II. Related Work and Background

UAVs have been very well established in research, commercial business, as well as military uses. Recent developments in technology in the past decade have introduced drone swarms where multiple drones are flying in very close proximity. Notable projects include Perdix [1] being developed for the military and Intel's drone shows [2] used for entertainment purposes, where the drones must station keep very precise flight paths to achieve animations and graphics. These swarm systems are more computationally intensive than flying each drone independently, as each drone must sense and coordinate with nearby drones to not collide.

Another implementation of a drone swarm is where the vehicles are physically linked to create a single, larger vehicle. Projects include Link! [3], a NASA study to increase flight performance and ModQuad [4], a novel approach from the GRASP Lab at University of Pennsylvania to assemble drones mid-air using magnets. Other projects also involve the study on linking fixed wing aircraft at their wingtips and the resulting aerodynamic effects [5], [6]. This rigid swarm method yields a simpler control system as the swarm as a whole can be governed by a single vehicle controller.

III. Solution and Technical Work

A. Airframe

The role of the airframe subsystem is to design and build both the physical frame and the modifications to the commercial off-the-shelf drones to suit AVIATA's uses. AVIATA was designed around commercial off-the-shelf parts and other parts that can be fabricated using 3D printing methods for cost-effectiveness and ease of manufacturing. This was a particularly significant consideration due to the team's lack of lab and facility access due to the COVID-19 restrictions for a significant portion of this project's duration.

1) Hardware: The drone is built on a DJI Flame Wheel F550 hexacopter frame and is equipped with a Pixhawk 2.4.8 flight controller running a custom version of PX4 modified to handle various airframe configurations of AVIATA. The drone includes a rangefinder to more accurately determine its altitude above the ground.

The drone propulsion setup was determined through iterative testing after realization of the kit-provided motor and propellers did not produce sufficient thrust, particularly with the need for additional yaw control achieved through canting the motors at an offset angle of $10°$. With a 4S lithium polymer battery each drone is theoretically capable of producing $7.5kg$ of thrust.

The major COTS and custom designed components outfitted on each drone are listed by category below.

TABLE I: Notable Hardware Components in Each Drone

Category	Item	Quantity
Drone COTS Components		
Frame	F550 Hexacopter Frame	1
Frame	Universal Landing Skids	4
Propulsion	AIR2216 KV880 Brushless Motor	6
Propulsion	T1045 Propeller	6
Propulsion	SimonK 30A ESC	6
Propulsion	4S 5200mAh Lipo Battery	1
Control	Pixhawk 2.4.8 Kit	1
Control	M8N GPS and Compass	1
Control	V5 915 MHz 100mW Telemetry Radio	1
Control	TFmini-s Micro Lidar Module	1
Control and Docking	Raspberry Pi 4	1
Docking	Raspberry Pi Camera Module V2	1
Docking	S3003 Servo Motor	1
Drone Custom 3D Printed Components		
Frame	GPS Mount (frame dependent)	1
Propulsion	Canted Motor Mount	6
Control	Raspberry Pi Case	1
Docking	Camera Mount and Arm	1
Docking	Docking Mechanism	1

2) Frame Design: The AVIATA frame is the rigid, unpowered structure for which the COTS drones dock to, the payload is mounted, and the AprilTag [7] visual targets are placed for the drone's computer vision-based docking procedure.

The frame is comprised of two carbon fiber plates that sandwich eight square carbon fiber arms mounted radially in a spoke pattern. Square carbon fiber tubes were chosen for the advantages in torsional strength over circular tubes and the ease of alignment, and the A large (13.5 cm x 13.5 cm) central AprilTag target is mounted above the center plate, and the payload mount is mounted below the center plate. At each end of the arms is a 3D printed docking joint and a peripheral (6 cm x 6 cm) AprilTag target.

Fig. 1: CAD rendering of the AVIATA frame with the large central AprilTag target and docking joints (peripheral targets not rendered).

3) Docking Mechanism: The role of the docking mechanism is to facilitate the smooth integration of each drone to the central frame during switch-offs and provide a solid attachment to the system for stable flight.

The male connector, mounted on the individual COTS drones, houses a servo motor that drives 4 pins which lock the drone to the central frame. At the point of the male connector is an aluminum plate, which when fully mated with the female connector, creates a complete circuit that indicates that the docking procedure is complete.

The female connector, mounted on the central frame, opens with a large cone that provides a greater margin of error for the controls and encourages the mating of the male connection. As the male connector of the COTS drone slides into the female connector, grooves ensure that the mate is completed in the correct orientation. Once the aluminum plate contacts the bottom of the female connector and completes the circuit, the servo activates and locks the pins into place, securing the COTS drone to the central frame.

B. Controls

The role of the controls subsystem is to implement cooperative behavior to fly the AVIATA frame and achieve mission tasks while supporting drone docking and undocking. Additionally, autonomous procedures should ensure safety at all times, making emergency landings if needed, and commanding drones to dock and undock as required for keeping the frame airborne with sufficient battery levels. For the first phase of the project, we have focused on the flight capabilities of the linked vehicles, with more advanced autonomy yet to come. Because

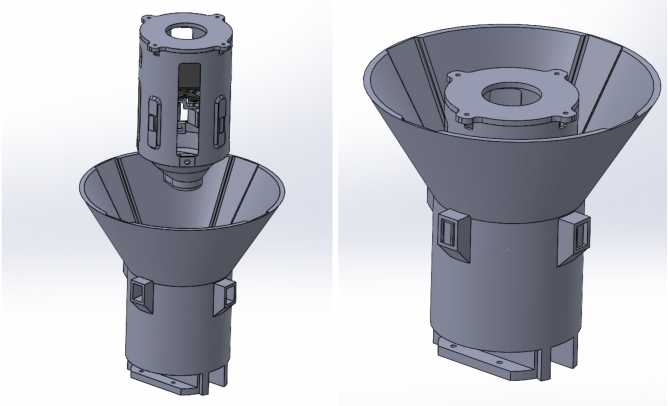

Fig. 2: CAD rendering of the AVIATA docking mechanism before and after docking complete.).

cooperative flight requires many of the same functionalities as standalone vehicles, a Pixhawk running PX4 Autopilot [8] was utilized on each drone to provide sensing and control capabilities, with only minor modifications required.

1) Overarching Concept: Cooperative flight is accomplished by rigidly attaching the drones to a central frame. This paves way to a simple realization: The frame is no different from a regular UAV, except that different actuators are controlled by different computers which must communicate. Therefore, we begin by approaching control exactly as standard multicopters do. This approach involves calculating a simplified version of the system dynamics, and finding an inverse of the dynamics to allow control. Of course, there are additional complications with linked aircraft that arise in the real-world, and these are elaborated on in the following sections. To further understand the problem and gather initial ideas for our approach, we looked to similar past achievements such as ModQuad by UPenn [4], another example of flying rigidly-linked multicopters with in-air attachment.

2) Control Allocation: To determine a control allocation, standard multicopters assume that the thrust force F produced by each rotor can be directly controlled, and that the torque τ produced around the rotor axis is proportional to its thrust ($\tau = kF$) [8]. Therefore, given k, the rotor locations, and each rotor's direction of rotation, linear dynamics for the vehicle can be derived using rigid body dynamics. For example, the matrix transformation in Equation 1 represents the dynamics of a quadcopter with arm length r and four rotors generating thrust $F_{1..4}$, with the left-hand side of the equation representing the net force and torque.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & -r & 0 & r \\ -r & 0 & r & 0 \\ -k & k & -k & k \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{1}$$

Standard multicopters generate a control allocation by calculating an inverse of these dynamics (also known as actuator effectiveness), and the same is done for AVIATA. Of course, the frame can take on many different configurations because

any docking location could be without a drone, so the control allocation for each configuration is precomputed and the appropriate one is selected when a drone docks or undocks in flight.

While force and torque can be used as inputs to the control allocation, this proves problematic when the drones dock and undock, since the mass of the frame changes abruptly, and so does the required thrust. A similar discontinuity occurs for torque. To solve this, we estimate the mass and moment of inertia for each configuration of the frame, and use that to calculate a control allocation that takes linear and angular acceleration as input.

Knowledge of the acceleration capabilities of the frame also serves as a good way to quantify its performance. For example, knowing how much thrust is required to hover (i.e. accelerate at 1 g), motor specifications can be appropriately chosen. Additionally, this helped us decide on a motor tilt angle of 10° to increase yaw torque, explained further in section IV-B4.

For some tests of AVIATA, a unique method for calculating inverse dynamics was used. Standard multicopters calculate the Moore-Penrose pseudoinverse of the dynamics to produce their control allocation [8]. While this would be adequate for our tests thus far, it can lead to premature motor saturation for lopsided configurations where one or more drones are absent from the frame. For this reason, we have experimented with a custom method for generating control allocations using linear optimization, with the help of the Python package CVXPY [9]. Our method optimizes for maximum thrust, and then optimizes for roll, pitch, and yaw torque when at 50% thrust (which should be close to hover thrust). In theory, this improves flight capability by prioritizing maximum achievable thrust and torque within motor constraints, as well as acknowledging that all maneuvering will occur near hover thrust. However, tests have shown that we must be careful not to apply too much stress on the less rigid parts of the frame, as discussed in section IV-B4, so further improvements will be made.

3) Cooperative Control: Control of the drones is based on the existing control architecture in PX4, shown in Figure 3. To fly the drones cooperatively, we choose a point in the control cascade at which to share data among all the drones. Before this point in the control loop, a "leader" drone is solely responsible for all decisions and computation, but once it shares data with the other "follower" drones, they each operate independently.

For various reasons, we have chosen to share the attitude & thrust setpoint ($\boldsymbol{q}_{\mathrm{sp}}$ combined with δT_{sp}) among the drones (from hereon referred to simply as "attitude setpoint"): The leader drone solely determines the attitude setpoint in each control cycle, and thereafter each drone independently executes attitude control. This decision is a tradeoff between the required communication throughput and the guarantee of agreement between drones. Communication throughput must go up for lower levels of control because they operate at higher rates, while the guarantee of agreement is lost as higher levels of control are assigned to run independently on each drone. Sharing the attitude setpoint is attractive because the 50Hz velocity controller that outputs the attitude setpoint is much slower than the 250Hz attitude controller and 1kHz attitude rate controller in the PX4 flight software (see Figure 3) [10]. We had also anticipated acceptable variability in attitude estimation, and additionally, the same method is used in the ModQuad project [4]. This is not to say that sharing the attitude setpoint is indisputably the best choice. For example, it may be acceptable to share the attitude rate setpoint, even if it cannot be communicated at 250Hz. Nevertheless, sharing the attitude setpoint has worked well thus far.

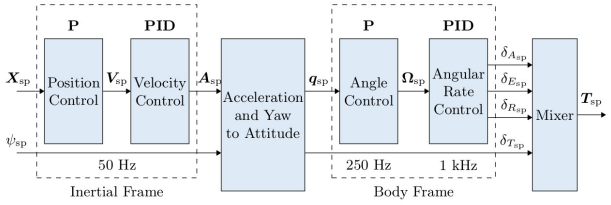

Fig. 3: PX4 Multicopter Control Architecture [10]

A couple complications arise from this. First, small measurement errors in compass heading result in fighting between drones over yaw control, but this is easily mitigated by rotating the attitude setpoint on each follower drone such that they all have the same yaw error as the leader drone. Similarly, drones will fight over control if offset from each other in roll and pitch, which may occur from mechanical imperfections and bending. As explained in section IV-B4, more testing is needed before moving forward on this potential issue.

Besides the attitude setpoint, other information must be communicated for the drones to fly cooperatively. Specifically, each drone must know the location of itself and all other docked drones on the frame in order to select the appropriate control allocation and use the appropriate actuator outputs from it, as well as make the necessary coordinate transformations to account for the docking joint angles. The current approach, assuming that humans carry out the initial drone placement, is to manually tell each drone where it is docked via a ground station. After takeoff, drones that dock or undock will broadcast an announcement to the other drones. By nature of the docking procedure, drones are commanded to dock to a specific location and will know if they have successfully docked to it, allowing all drones to have accurate knowledge of the configuration at all times. Note that this mode of operation is not yet tested because the system is not yet capable of in-air docking.

Another procedure is needed due to the fact that any drone, including the leader drone, can undock from the frame at any time. When this happens, the leader drone must notify one of the remaining docked drones that it should become the leader. This is a fairly simple operation, with a few features to ensure a smooth transition. First, the original leader drone does not undock until it has confirmed that the new leader has taken control. This is accomplished via the new leader responding with an acknowledgement after it has successfully transitioned modes. Communication latency may lead to both drones sending setpoints simultaneously, so this is mitigated by assigning an ID to each leader drone. This ID increments every time a new leader is assigned, so follower drones avoid confusion by only listening to the most recent leader they

are aware of. Tests outlined in section IV-B3 reveal additional improvements that could be made involving control continuity.

4) Simulation: Simulation has been a useful way to demonstrate proof-of-concept before testing on real hardware. This has been focused around demonstrating that the hardware specs and control loops are sufficient for flight. To accomplish this, a Python program was written to simulate rigid body dynamics and a simplified position controller, with the physical properties of the frame and rotors completely configurable. While the simulator ignores many real-world processes (for example, it assumes rotor thrust is directly controllable), it has allowed us to test various hardware specs, control allocations, and PID gains to get a rough idea of what might work in practice. Additionally, it is designed with modularity in mind so additional features and models can be added if desired.

Another simulation option exists, which is to use existing robotics simulation software such as Gazebo. In fact, there exists a package for simulating drones with the PX4 software in Gazebo [8]. However, we were unable to invest the time needed to configure these simulations to use multiple rigidly-connected vehicles, and prioritized real-world testing instead.

5) Software Implementation: Each physical drone carries a Pixhawk flight controller running a modified version of PX4 v1.11 and a Raspberry Pi running a custom C++ program. The C++ program interfaces over UART with the Pixhawk and over WiFi with the other drones' Raspberry Pi's. The modifications to PX4 and Raspberry Pi program allow for the cooperative behavior that has been outlined. The network interfaces on the Raspberry Pi's are configured to communicate via OLSRv2 mesh networking, and ROS2 is used to facilitate these communications, including those involving a ground station. All relevant software can be found via the AVIATA GitHub repository (see Appendix B).

C. Communication

The role of the communication subsystem is to provide a reliable, performant, and flexible network over which control commands and telemetry data can be transmitted. To avoid a single point of failure, a mesh network topology was chosen. This network topology has the added benefit of allowing direct communication between neighboring drones.

The Optimized Link State Routing Protocol Version 2 (OLSRv2) protocol chosen to implement this network [11]. As an existing network-layer solution, additional code implementation was not needed. This protocol handles neighbor detection and network topology control, providing the wireless communication requirements needed.

D. Docking

The role of the docking subsystem is to design and implement a software procedure for getting drones to and from AVIATA. Idle drones will receive a docking command from AVIATA when a fresh drone is required to replace a dying drone, at which point they will take off, fly to the system using GPS, and transition to a computer vision-based system for the finer aspects of mechanical docking before officially joining the swarm.

Computer vision is used because of uncertainty when using GPS and the high level of precision and accuracy required to dock a drone. On-board GPS guarantees location up to 2.5 meters, which is far from the centimeter-level accuracy needed. Although technologies such as RTK GPS can improve this, computer vision was deemed to be more robust to external factors such as slight movement of the airborne frame and possible communication delays in transmitting the GPS location of the drone's assigned docking location.

The general control flow during docking is a simple loop featuring a PID controller that takes a camera frame as its input and calculates new velocity setpoints for x, y, z, and yaw to output to the drone's flight controller. This code runs on a Raspberry Pi 4 on-board the drone with a direct wired connection to the Pixhawk flight controller.

Algorithm 1 Docking control flow

```
while errors > tolerance do
    Get next image from camera
    if AprilTag not detected then
        Handle failure
    end if
    Calculate new errors based on location of AprilTag
    Update PID controller
    Send new velocity setpoints to drone
end while
```

Docking occurs in two stages, allowing for the use of both a target large enough to be detected within the error of the GPS and a target small enough to fit in the camera's field of view at close range. In the first stage, the drone uses a large target placed in the center of the frame to position itself approximately halfway between the center of the frame and the intended docking slot, adjusting the drone's yaw to maximize the use of its horizontal field of view. The first stage of docking ends when the drone can reliably detect the smaller target near its intended docking slot. In the second stage, the drone positions itself horizontally above its intended docking slot before descending vertically into the slot. Failure to identify either target during docking is handled by a robust error-handling system, which determines whether to hold position, ascend, re-attempt an earlier stage, or abort docking based on the current drone state and previous target detections.

AprilTag [7] was chosen for this part of the docking subsystem due to its high accuracy and performance in determining the 3D location of a detected target in a given camera frame. This library allows us to reliably calculate how offset the drone currently is with respect to the target we wish to minimize our distance to, and this offset is the error used to update the PID controller.

Finally, a circuit controlled by the Raspberry Pi is used to detect when a drone has achieved docking. 2 GPIO pins on the board are used, one as output and the other as input. By placing a conductive material in the docking joint and wiring the GPIO pins such that the circuit is completed when the drone is fully coupled with the joint, a simple switch is created. When the switch is closed, the Raspberry Pi knows that the drone is ready to officially join the AVIATA swarm.

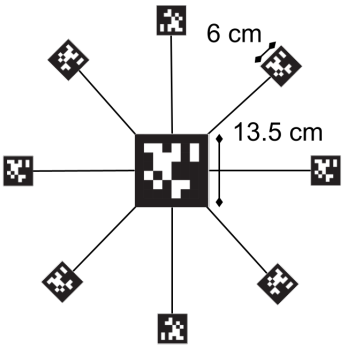

Fig. 4: Aerial diagram of AprilTag targets on the AVIATA frame. A docking joint is located at each peripheral target, and the central target is used for the first stage of docking.

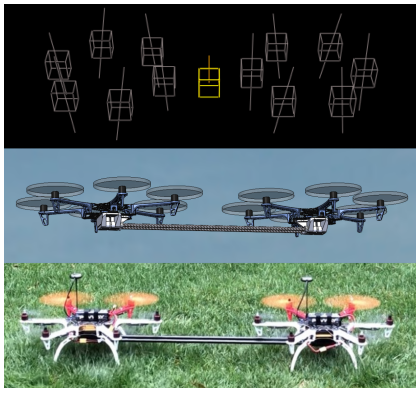

Fig. 5: Two-drone frame. The top image is its representation in the Python simulator, the middle image is the CAD model, and the bottom image is a photo of the real system.

Fig. 6: Four-drone frame in flight

IV. Experimental Results

A. Airframe

A final design for the proposed 8-drone airframe with docking mechanisms has not been implemented yet. Instead, several individual systems were used to test controls and docking in a simpler environment. The specific airframe setups are described more in their respective sections.

1) Future Work: Now that the 4-drone system and docking subsystem have been tested, the next step is to integrate the docking subsystem on to the 4-drone test. Once stable flight has been demonstrated with the docking mechanism, the team will move on to finalizing an 8-drone design capable of carrying a payload.

B. Controls

Two systems were built to test linked multirotor control. First, a two-drone frame was built by connecting two drones with carbon-fiber tubes attached to 3D-printed mounts. This is depicted in Figure 5. This design was chosen to serve as the simplest possible starting point for testing cooperative control of multiple rigidly-attached drones. The second system was a four-drone frame of equal size to the final eight-drone design, with differences being that there were four drones instead of eight, and static attachments were used rather than docking mounts. The temporary 3D-printed mounts and center support were connected using similar carbon-fiber beams as in the final design. The four-drone frame is depicted in Figure 14.

Next steps include outfitting the four-drone frame with actual docking mounts before finally moving on to the final frame design.

The DJI Flame Wheel F550 hexacopter design was used for all tests. For the two-drone apparatus, each drone was equipped with a 5200mAh 3S LiPo battery, HobbyPower 2212 920KV motors, and GemFan 1045 propellers. For the four-drone tests, a 5200mAh 4S LiPo battery was used with T-motor 2216 880KV motors and T-motor 1045 propellers. This hardware was chosen to be suitable for any potential continuation of the project.

1) Attitude Control: While we did not set many specific requirements for flight, the frame should be capable of holding position and following waypoints. To this end, the frame should have sufficient attitude control, which can be verified via flight logs and visual observation of the flights. In this section, the analysis will focus on the flight logs. See Appendix A for recorded flight videos that may provide further insight.

To test attitude control, the leader drone was commanded via manual control stick input from an RC transmitter. Trial and error was required in order to adjust the controller gains, but eventually we arrived at gains that worked adequately. Figure 7 shows the roll setpoint taken directly from the control stick compared with the estimated roll angle for each drone on the two-drone apparatus. Roll angle is analyzed here because it is the more difficult axis to control for the two-drone frame. This test, combined with observations from other tests, provides us with confidence that the multi-vehicle frames have comparable maneuverability to that of standalone multicopters. Although the commanded angles are small, this test exhibits adequate tracking for roll angle. Additionally, this demonstrates negligible latency and packet loss over the wireless mesh network. Because roll estimate is determined independently by each drone's on-board sensors, we expect the roll estimate plots to look identical and have no relative time difference. Ideal communication would yield the same for roll setpoint, but network latency would cause a relative

time shift in the roll setpoint on the follower drone. Luckily, no shift is observed, and the plots look nearly identical, so we can conclude negligible latency and packet loss.

After scaling up to four drones, no difference in network performance was observed. However, there was a larger offset in attitude between individual drones due to bending at the joints and other mechanical imperfections. Nevertheless, maneuverability was still adequate.

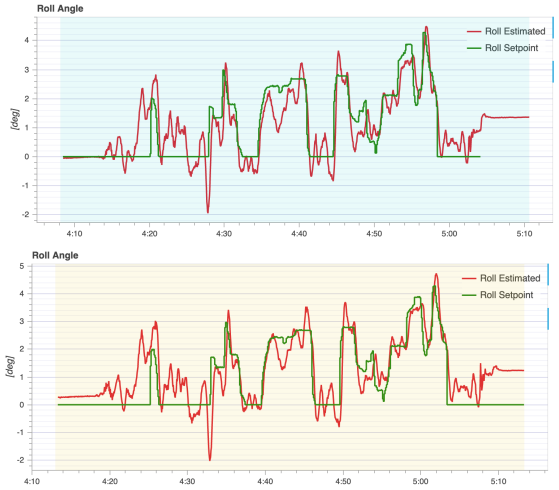

Fig. 7: Attitude control roll setpoint compared with estimated roll angle for each drone in the two-drone frame. The roll axis is perpendicular to the connecting beam between the drones. The top graph is for the leader drone, while the bottom graph is for the follower drone.

2) Position Hold: Holding position in flight is a basic yet vital requirement for this system. Although we have not set quantitative requirements here, we can report that both the two-drone and four-drone frames had the ability to hold position while remaining smooth and stable in flight, even in the presence of moderate wind. The two-drone frame exhibited similar position hold accuracy to that of a standalone drone.

There are a couple caveats to mention. First, after manually adjusting the yaw setpoint during a flight of the four-drone frame, the apparatus became unstable and exhibited large, low-frequency oscillations in roll and pitch. We believe this can be mitigated with further controller tuning and software improvements. Second, unrelated software bugs prevented us from achieving long-duration flights with the four-drone frame, so at this time there is less which can be said about its capabilities.

Videos demonstrating position hold can be found in Appendix A.

3) Leader Transfer: AVIATA has several unique requirements in order to enable cooperative control of a structure with indefinite flight time, one of which is passing the leadership role between drones in flight. We have focused on implementing leader transfer because it does not require docking and undocking during flight, which our hardware is not yet capable of.

Recall that the leader drone is responsible for position control, and determines an attitude setpoint for all drones on the frame. A "leader transfer" is the act of gracefully transferring this role: The new leader begins running position control and broadcasting attitude setpoints, while the previous leader stops position control and begins listening to attitude setpoints. In our tests, the new leader is commanded to hold its current position. While no state information needs to be transferred between the drones, they are expected to independently and adaptively estimate the thrust required to hover.

Test results from a flight of the two-drone frame revealed that we needed to modify PX4 to enable the hover thrust estimator in attitude control mode, as we saw the new leader begin commanding the default hover thrust, indicating it had not been updating its estimate.

Test results from a flight of the four-drone frame show good stability during the transfer. However, the new leader again started out commanding too little thrust (shown in Figure 8), leading to a small deviation in altitude. Ultimately, there was no danger to the integrity of the flight and a quick recovery was made, but it is clear our efforts to maintain continuity in thrust did not work as planned. More testing is needed, but it may be that the hover thrust estimator does not perform as we'd hoped, and transferring additional information (e.g. integrator buildup) may help ensure continuity.

Videos demonstrating leader transfer can be found in Appendix A.

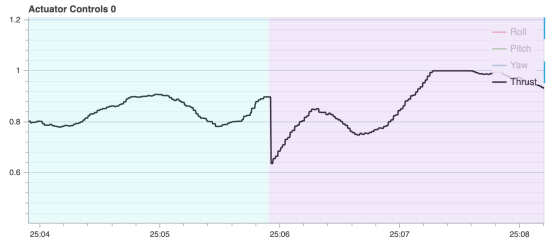

Fig. 8: Total thrust during leader transfer, from the perspective of the new leader. The background color change indicates when the vehicle becomes the leader. For the purposes of this analysis, the thrust can be considered unitless.

4) Problems and Mitigations: During the first tests with the two-drone frame, a couple of issues were noted. The first problem was inconsistent altitude control. We attributed this to typical uncertainty in the GPS and barometer when flying close to the ground, as we conducted tests just a few meters above the ground. We desired better performance so that flights would be more predictable, and so potential docking tests would be more likely to succeed. To improve altitude hold, we installed a TFmini LiDAR on two of the drones (note that only the leader drone needs to use it), and this was successful in achieving consistent altitude hold within a few centimeters.

The second problem was a lack of yaw control authority, made evident by motor saturation in the flight logs. This made sense because the two-drone frame, for example, has twice the yaw torque capability of a single drone, but far more than twice the moment of inertia in the yaw axis. To improve yaw control authority, we mounted each motor at a 10 degree angle to "amplify" its contribution to yaw torque on each drone. This preserved each drone's ability to fly on its own

while still creating ample opportunity for more force vectors to apply yaw torque on the combined frame. The improvement in maximum yaw torque was roughly 7-fold. These two solutions were spawned with the help of our CDR reviewers, and we would like to recognize them for their thoughtful suggestions.

The four-drone frame had its own unique problems. First, we noticed oscillations or "wobbling" of individual drones caused by torsion on the booms holding each drone (see Figure 9). This was clearly caused by torque applied by each drone in efforts to maintain attitude control for the frame, and is a good example of where we cannot assume perfect rigidity of the frame. In this case, the problem was mitigated by swapping the control allocation to one that applied far less torque on individual drones, in favor of relying more exclusively on the thrust difference between drones which puts less stress on the frame, and has much greater control effectiveness anyway. The mistake here was that the first control allocation we used was optimized to have the greatest possible net torque, with no regard to the weak points of the structure or wasting energy on motors with low effectiveness for a given torque axis. We switched the control allocation to the widely-used Moore-Penrose pseudoinverse to reduce the twisting forces. Generally, it appears that an energy-efficient control allocation also reduces stress on the structure, and the Moore-Penrose pseudoinverse is known to be energy efficient [4]. However, it does not work well for lopsided drone configurations which will arise when undocking drones from the full frame. Therefore, we plan to make improvements to our current optimization method to minimize the stress on the frame in addition to maximizing the net forces and torques.

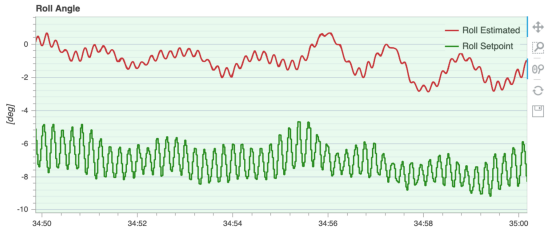

Fig. 9: Roll angle oscillations of one of the drones during a flight of the four-drone frame, captured over 10 seconds. The drone wobbled about the axis of the boom.

In addition, we saw a control issue arising from bending joints in the four-drone frame. Generally, issues can arise when the drones are offset from each other in attitude. An obvious way this can occur is bending where the booms connect to the central payload, as the drones lift the frame from the ends the booms. This creates a scenario like the one in Figure 10. While the effect was small during nominal testing, we had a chance to see detrimental effects after unrelated difficulties led to cracking the temporary 3D-printed central support of the four-drone frame. This led to significant bending that caused the frame to lose altitude shortly after takeoff. Looking at Figure 10, this is clearly explained by the conflict between attitude control, which wants to decrease the thrust of all drones, and altitude control which wants to maintain thrust. This conflict leads to undesirable behavior, especially because the controllers use integral control which will build up quickly. More testing is needed to determine if the problem persists when the frame is intact. If that is the case, an in-flight attitude calibration procedure will be implemented.

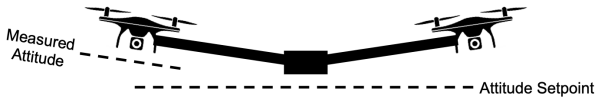

Fig. 10: Example of how drones may be offset from each other in attitude.

Lastly, the four-drone frame still had persisting yaw issues, even with all drone motors slightly canted at the 10 degree angle. In this case, we observed the frame would have a slow yaw drift in one direction (see Figure 11), which correlated to about one full revolution every 20 seconds. The four-drone frame was still able to be manually flown, but in position mode, this meant that the drones are not able to maintain it's heading, which are necessary for a the frame to autonomously hold position.

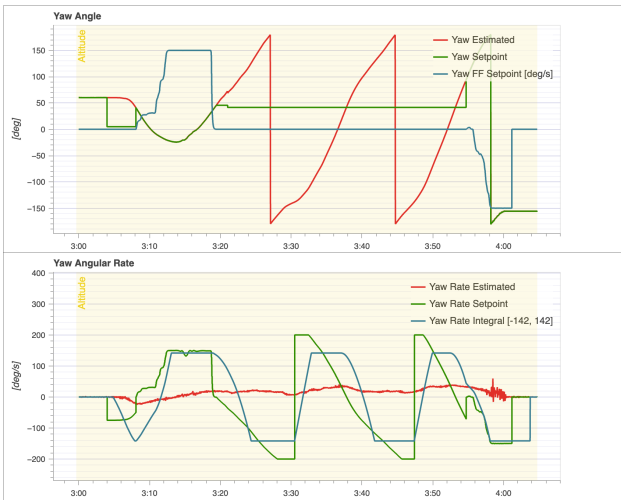

Fig. 11: Attitude control yaw setpoint compared with estimated yaw for the leader drone in the two-drone frame. Top graph is the yaw position, and the bottom graph is the yaw rate. The discontinuous jump corresponds to the graph wrapping around the 180 degree axis.

We hypothesize that this problem is in part due to the imperfection of the four-drone test assembly which causes the drones to have unaligned attitude readings among the four drones, which is related to the previous issue discussed. In this case, the misalignment is causing each of the AVIATA drone's onboard computers to improperly handle the attitude setpoints. We come to this conclusion from the present Pitch bias in the leader drone as seen in Figure 12. Each drone performs desaturation to prevent the motor control exceeding the motor's capabilities (i.e. going above 100% or below 0% thrust). Each drone performs its own desaturation calculation as though it were controlling the entire frame, but only

utilizes the motor outputs corresponding to its own motors. Performing desaturation on all the motor outputs is important for stability, but it only makes sense if the drones agree on the motor outputs. However, the drones share attitude setpoints, not motor outputs, so when they measure slightly different attitudes, they will have different motor outputs, and so each drone's prediction of the others' motor outputs is less meaningful. Additionally, the problem is exacerbated by the fact that yaw has a relatively low priority when PX4 performs desaturation. Thus, we see this yaw drift as a culmination of the attitude offset desaturation problem.

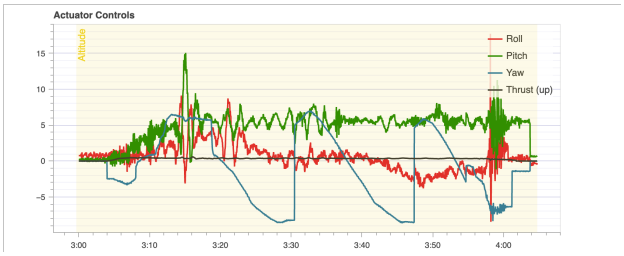

Fig. 12: Actuator controls in roll, pitch, and yaw, captured from the leader drone from same flight as Figure 11. The drone has a clear forward Pitch bias, as compared to the Pitch.

Some solutions to alleviate this issue require resolving the measured attitude offsets between the drones. Proposed solutions include calibrating the attitude as the average of the drones' attitudes and using this attitude for the calculations for attitude setpoints and desaturation, which should cause the motor outputs to have pretty good agreement. Another and more straightforward approach is simply to treat the leader drone's estimated attitude as the ground truth and give all the other drones the same attitude error. The downside of this solution is that it will rely more heavily on the integrator for position hold PID control. It may also make leader transfer less smooth as the offset error will be discontinuous. We plan to implement the first method and test the benefit of stability from the attitude offsets.

C. Communications

Only the physical performance of the communication system was tested, as simulation would be insufficient to determine network performance in practice (due to the importance of hardware performance on network behavior). The communications testing setup consisted of 3 physical devices, arranged in two configurations: a straight line, and an equilateral triangle. Randomized packets of size 1KB were sent with a random interval between 4500-5000 microseconds. The difference between send and receive time (the delay) was measured for each packet. The results of these tests are shown below.

While the delay for a one-hop node is significantly higher, the network performance is satisfactory for this application. We hypothesize the delay spikes observed could be caused by the message redistributing mechanism or signal interference. However, because these spikes are relatively infrequent, the network is reliable enough and the latency is sufficiently low for this application.

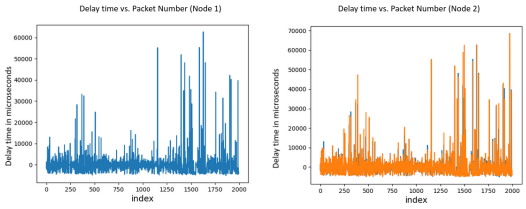

Fig. 13: Communications Test Results (Neighboring Nodes)

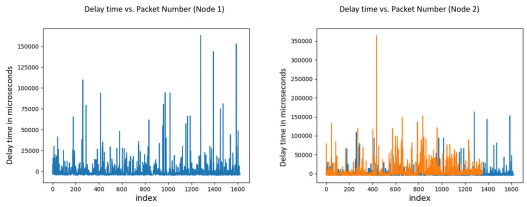

Fig. 14: Communications Test Results (1 Hop Distance)

D. Docking

The processes within the docking subsystem can be broken down into two main parts, characterized by the proximity of the drone to the target. Firstly, we test the multi-stage docking system, which is the stage during which the drone is far from its final destination. Secondly, we test the drone's precision and accuracy when it is within one meter from its docking location.

All of this was accomplished very early on in simulation. Using Gazebo, a program which simulates the same PX4 flight controller we use on the drones, we were able to prototype and test all of our code. The next step was to validate these results in physical tests.

1) Multi-stage docking: This test ensures the drone can get within range of the peripheral target when making its initial approach. The testing setup involved real-size targets pinned to the ground at the appropriate locations. The drone takes off to approximately 5 meters, find the central target and execute the 2-stage docking procedure, and initiate its landing sequence upon being within 35cm of the peripheral target. After doing some minor PID tuning to get steady flight, the drone was able to successfully execute the 2-stage docking procedure previously described. The main point of improvement is the speed at which the 2-stage procedure executes; the drone oscillated for roughly 20 seconds around the point at which it is supposed to transition to the second stage. This can be improved by either improving the PID controller, or loosening the tolerances. The latter would likely have little negative impact given that the drone did not oscillate far enough to lose sight of either target.

2) Precision Docking: The purpose of these tests was to fill in the gap left by the multi-stage docking test. The testing setup included a docking guide cone and drone attachment, which was 3D printed by the airframe team and is shown in Figure 16. The target is pinned to the ground next to the cone. The drone would take off to 1 meter, and execute stage 2 of docking. After each test, the PID controller was manually tuned by analyzing data from the logs. This data also led to

Fig. 15: A drone executing 2-stage docking, currently transitioning between the stages. The central tag is on the left, and the peripheral tag is 1 meter to its right.

developing new logic to increase precision, most notably a way of tying the drone's vertical and horizontal velocities together such that the drone stays within an imaginary inverted cone. This ensures the drone doesn't fly too far given its altitude to lose the target from its field of view. One item to improve upon is the drone's performance in windy conditions. The smallest gust of wind is enough to blow the drone by just a few centimeters, but at extremely low altitudes this is sufficient for the drone to lose the target from its field of view. Nevertheless, in calm conditions, repeatable results were obtained, so this test can be considered a success.

Fig. 16: Testing the precision and accuracy of the PID controller by attempting to land inside the guide cone.

3) Future Work: The next logical progression is to combine test series 1 and 2 to enable 2-stage docking with the guide cone at the end. The increased precision obtained from the PID tuning performed during the second test will likely help improve the performance of 2-stage docking as well. Afterwards, the drone will be ready for testing with the complete docking mechanism that the airframe team has developed.

V. Conclusion and Future Work

The proposed system for indefinite flight combines 8 drones cooperatively into a central structure, allowing dynamic replacement of individual drones in-air. Proof-of-concept testing has verified the core functionality of each subsystem, making this concept a viable avenue for further development.

The airframe consists of two parts: a central unpowered frame connecting the 8 drones, and a custom mount to which individual drones can attach. The central airframe, consisting of carbon fiber tubes surrouding carbon fiber plates, rigidly supports the 8 drones surrounding it. The docking mount uses a servomotor activated on docking by electrical contact to lock the docked drone into place. Initial testing, while limited, has been promising for the airframe. Four-drone flight tests confirmed the structural integrity of the central frame, and docking tests showed the viability of the overall docking subsystem.

The controls software subsystem is responsible for cooperatively supporting the required flight functionality of the AVIATA system. Controls allocations are calculated by a single leader using the inverse of the rigid body dynamics of the AVIATA system and distributed across the network. The controls system also handles the entrance and exit of drones to the system, adjusting controls allocations by the number of drones currently in the system. After initial proof-of-concept simulations, physical testing indicated that the core logic of the controls subsystem can provide the cooperative functionality required for the subsystem. While there exist some minor issues in the control subsystem, it appears that further testing could limit the impact of these issues.

The communications software subsystem relies on the OSLRv2 protocol without any significant code modification to provide a flexible, low-latency, and reliable mesh network topology. Performance testing confirmed that this network protocol and topology provided sufficient performance and reliability for this application.

The docking software subsystem uses a two-phase algorithm to precisely approach a drone's intended target. The AprilTag markers and corresponding library provide the 3D localization information required to update the local PID controller. Testing has shown that this combination provides the required precision to successfully dock a drone within the bounds of the docking mount.

Significant future testing is required in two areas to further develop this system. First, dynamic docking tests must be tested. Current docking tests focused on showing the potential of the docking subsystem, and were therefore confined to static tests. Future docking tests must include docking on the full airborne system. Second, the full system must be tested. The current tests only tested up to four drones attached to the system. The full eight-drone system still must be tested.

Future work is required in the automation of the cooperative controls system. Potential automation could replace low-battery drones with pre-charged drones or automatically assign certain drones to specific slots (or detect the drones in each slot). Further automation could also automatically generate controls mixers from individual drone data, allowing out-of-the-box integration of arbitrary drones into the system.

Appendix A
Supplemental Videos

- Single-drone position hold: [Link]
- Two-drone position hold (without canted motors or range sensors): [Link]

- Two-drone position hold and leader transfer (with canted motors and range sensors): [Link]
- Four-drone position hold (bad control allocation causing oscillations; instability after yaw input from RC): [Link]
- Four-drone leader transfer: [Link]
- Four-drone position hold (improved control allocation; cracked frame caused bending which presumably led to loss of altitude as explained in section IV-B4): [Link]
- Four-drone yaw bias (discussed in IV-B4), flight begins 2:15 in the video: [Link]

APPENDIX B
CODE REPOSITORY

All the code for AVIATA can be found at the following link:

- https://github.com/uas-at-ucla/aviata

ACKNOWLEDGEMENTS

These results are based upon work supported by the NASA Aeronautics Research Mission Directorate under award number 80NSSC20K1452. This material is based upon a proposal selected by NASA for a grant award of $10,811, subject to successful crowdfunding. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NASA.

The authors would also like to thank Dr. Brett Lopez for their valuable insights and advising.

REFERENCES

[1] "Department of defense announces successful micro-drone demonstration," Jan 2017.
[2] "Drone light shows powered by Intel," 2021.
[3] J. R. Cooper and P. M. Rothhaar, *Link!: Potential Field Guidance Algorithm for In-Flight Linking of Multi-Rotor Aircraft*.
[4] D. Saldaña, B. Gabrich, G. Li, M. Yim, and V. Kumar, "Modquad: The flying modular structure that self-assembles in midair," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 691–698, 2018.
[5] J. Quinlan, J. Pei, J. Cooper, R. Busan, P. Rothhaar, W. E. Milholen, T. Ozoroski, and C. Hartman, *Technical Challenges Associated with In-Air Wingtip Docking of Aircraft in Forward Flight*.
[6] J. R. Cooper and P. M. Rothhaar, "Dynamics and control of in-flight wing tip docking," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 11, pp. 2327–2337, 2018.
[7] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.
[8] "PX4 Autopilot." https://github.com/PX4/PX4-Autopilot. GitHub.
[9] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
[10] "PX4 controller diagrams." https://docs.px4.io/master/en/flight_stack/controller_diagrams.html.
[11] P. J. T. Clausen, C. Dearlove and U. Herberg, " The Optimized Link State Routing Protocol Version 2," RFC 7181, RFC Editor, April 2014.