Collaborative Control of Ground and Aero Vehicles

ECE 445 Final Report

Team 19
Alvin Sun (yixiaos3)
Jialiang Zhang (jz23)
Mingda Ma (mingdam2)

TA: Andrew Chen
Spring 2021
May 1, 2021

Contents

1 Introduction

1.1 Background and Motivation

Package delivery has become one of the essential services required by people's everyday life. Despite a well developed infrastructure, the package delivery system is very cost inefficient in the last few miles of each delivery because of traffic and sub-optimal ground transportation planning. Autonomous delivery over drone networks has become one of the new trends which can drastically reduce this last-mile delivery cost. However, drone-network on its own is very difficult to scale up due to the lack of battery life for all commercial drones, especially when carrying payload. To actually have such a system deployed in big cities, we could take advantage of the large ground vehicle network which already exists with ride-share companies like Uber and Lyft as well as public transportation networks such as buses and mailing / delivery services. The rooftop of an automobile has plenty of space to hold packages and a drone delivery network can drastically expands its readability by being able to pick-up and drop packages onto moving ground vehicles. We can then optimize for flight time and efficiency while having minimal interference with the automobile's route. An overview of the proposed framework is shown in Figure 1.

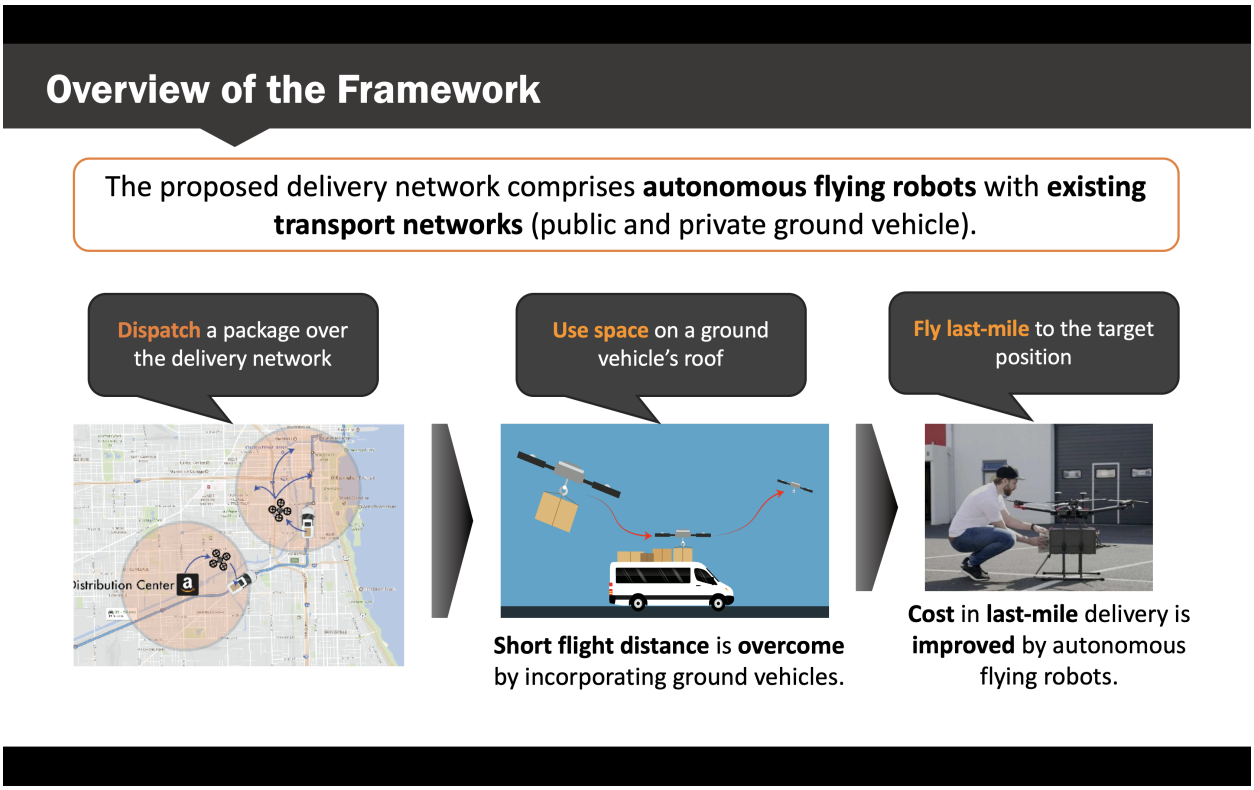


Figure 1: Framework Overview [1]

1.2 Challenges

While the proposed framework can dramatically increase delivery coverage and efficiency, the problem of safely docking a drone onto ground vehicles in motion remains a big challenge. We aim at implementing the idea in a lab environment by developing a decentralized multi-agent control system that automatically synchronizes a drone with a moving ground vehicle when in close proximity. As a proof of concepts, the project takes the assumptions that vehicle states (such as its position and orientation) can be accurately estimated. The infrastructure of the lab, the drone, and the ground vehicle will be provided by the support of our generous sponsor, Professor Naira Hovakimyan. We will achieve the synchronized motion through a collaborative peer-to-peer control scheme. More specifically, the ground vehicle will estimate its own trajectory several seconds into the future, and will periodically send the trajectory to the drone. Since the drone cannot acquire absolute position read from the motion capture system, the ground vehicle is also in charge of estimating the drone's poses (through motion capture). The drone will then optimize its current control to track this future trajectory.

2 Design

All designs for the project is carried out under a laboratory setting as illustrated in Figure 2. Hardware for both the drone (crazyflie) [2] and the ground vehicles (Jackal) [3] is provided by commercially available robotic research platform. We will design a separate piece of visualization hardware to help understand the performance of the control synchronization. An overall design block diagram is shown in Figure 3. The following sections will go into details of the design for each sub-component.

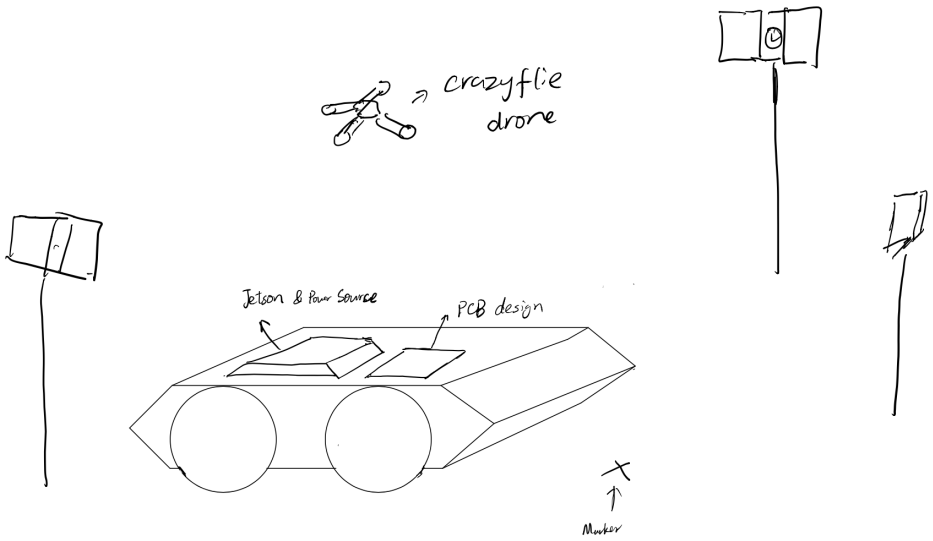


Figure 2: Physical Design

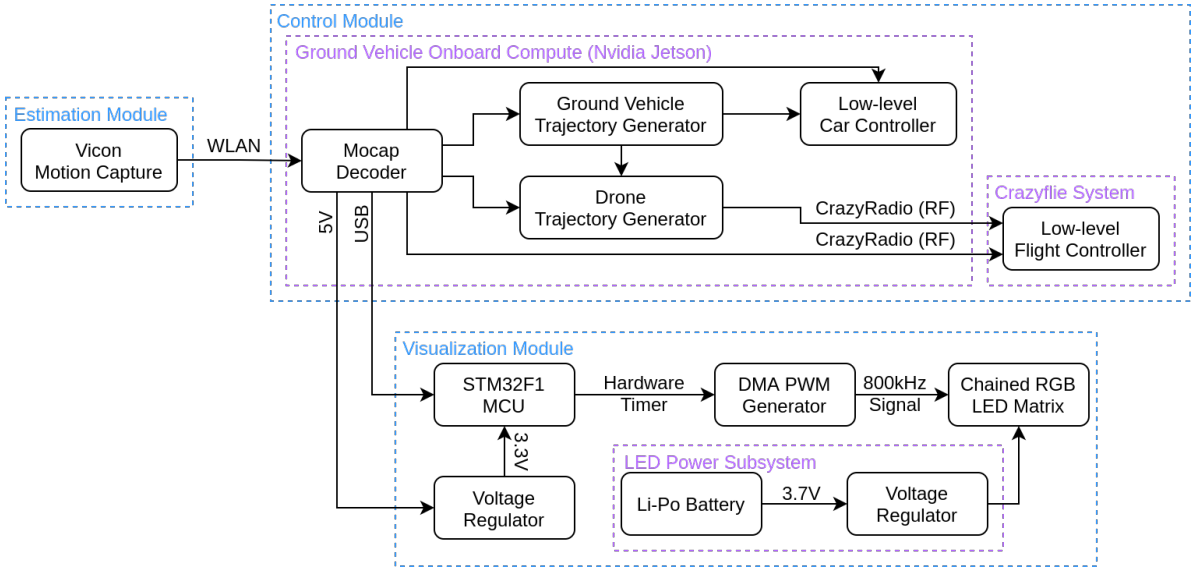


Figure 3: Block Diagram

2.1 Estimation Module

The Vicon motion capture system is a commercialized indoor localization system that takes advantage of multi-view high-speed imaging technologies. To configure it properly, we will need to install reflective markers to both ground and aero vehicles and calibrate their extrinsic poses before using the localization data coming from the system. The calibration problem is shown in Figure 4 [1]. Since the Vicon measurement can give accurate estimate for both T_{wj} and T_{wc}, we can calibrate for the relative transform using

$$T_{jc} = T_{wj}^{-1} T_{wc}. \tag{1}$$

To reduce the noise of the measurements, the calibration procedure is done by averaging 1000 measurements in the tangent space of T_{jc}.

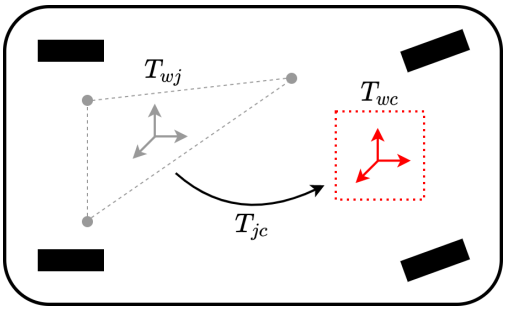


Figure 4: Pose Calibration

2.2 Control Module

This module is in charge of stabilizing both the ground vehicle and the drone while commanding them to stay within close proximity of each other. To achieve this, we will need to implement controllers for both vehicles as well as high-level trajectory generator to collaborate between the two. Since the drone has built-in firmware for low-level controller and estimator, the following sections will focus instead on the control of the ground vehicle and the trajectory generator. All control related computes happen on board the ground vehicle with the Jetson TX2 embedded platform.

2.2.1 Ground Vehicle (Jackal) Controller

The low-level controller for the Jackal is provided as a differential-drive controller, meaning that we can move the vehicle by commanding a linear velocity and a angular velocity. Due to the non-holonomic constraint of differential-drive vehicles, there is no simple linear control law that stabilizes the vehicle. As a result, a slightly modified version of a proportional controller is used to achieve trajectory tracking for the Jackal bot. The controller produces

[1] Note the variable naming convention that T_{ab} denotes an $SE(3)$ transformation from frame a to frame b. Also for the subsequent equations, w, j, and c is abbreviations for the world frame, jackal frame, and the drone (crazyflie) frame respectively.

both linear and angular velocity commands from the positional error e_x as well as heading error e_θ. The errors are calculated as

$$e_x = \|x - x_t\|_2 \tag{2}$$
$$e_\theta = \theta - \angle(x_t - x) \tag{3}$$

where (x, θ) describes the position and heading state of the Jackal, and x_t is the target position the controller is trying to track. A visual illustration is included in Figure 5. The control law is then simply applying some control gain, K_v and K_ω, on both of the errors.

$$v = K_v e_x \tag{4}$$
$$\omega = K_\omega e_\theta \tag{5}$$

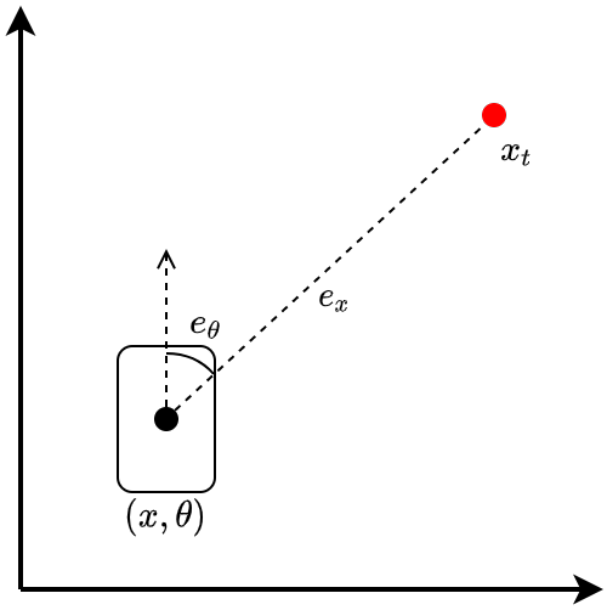


Figure 5: Caption

2.2.2 Trajectory Generator

Since the built-in controller on-board the drone is a linear controller, there is a certain amount of lag between the time that the commanded way-point is sent out and the time that the drone reaches the target. To counteract such control delay, we will need to send a future way-point to the drone so that the drone can synchronize well with the current Jackal position after applying that control delay. To achieve this, we implemented a constant twist [4] motion model to predict the motion of the ground vehicle a few hundred milliseconds into the future. The body twist of a $SE(3)$ transform falls onto the tangent space of the pose, which can also be viewed as the time derivative of the 3D pose on the $SE(3)$ manifold. The reason for using constant twist motion model instead of the simpler constant velocity motion model is that the twist representation encodes angular velocity as a screw motion, so

that the prediction will consider certain amount of curvature. The body twist is numerically differentiated (in tangent space) through consecutive pose measurement coming from the motion capture system. An illustration of consecutive measurement is shown in Figure 6. The dotted red curves represents the future prediction at each measurement time step using the constant twist motion model. The instantaneous body twist at each time instance, $V_j(t_i)$,

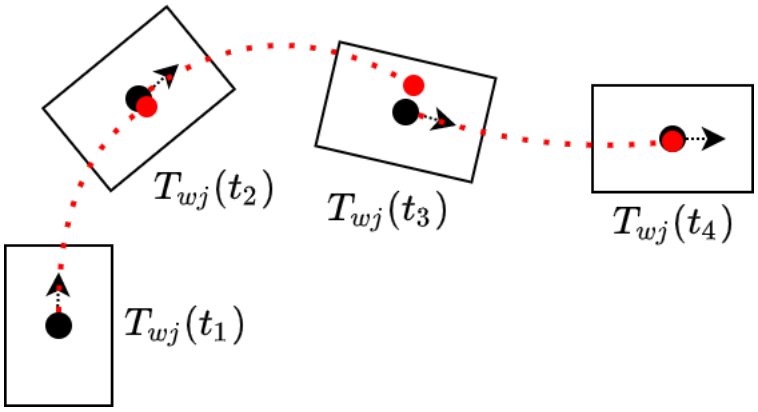


Figure 6: Jackal Trajectory Prediction

is obtained by the following numerical differentiation

$$V_j(t_i) = \frac{\log(T_{wj}(t_{i-1})^{-1} T_{wj}(t_i))}{t_i - t_{i-1}} \tag{6}$$

where log is taking the logarithm map of a $SE(3)$ pose. However, numerical differentiation amplifies noise especially with small time step. To ameliorate this problem, we applied a running average filter on those twist estimates to reduce such noises. Let h be such filter, and we obtain $\tilde{V}_j$, the filtered twist estimate

$$\tilde{V}_j(t) = h(t) * V_j(t). \tag{7}$$

Finally, using the latest available pose at t_i, we can apply the constant twist motion model and extrapolate to make future pose predictions at any time $t > t_i$.

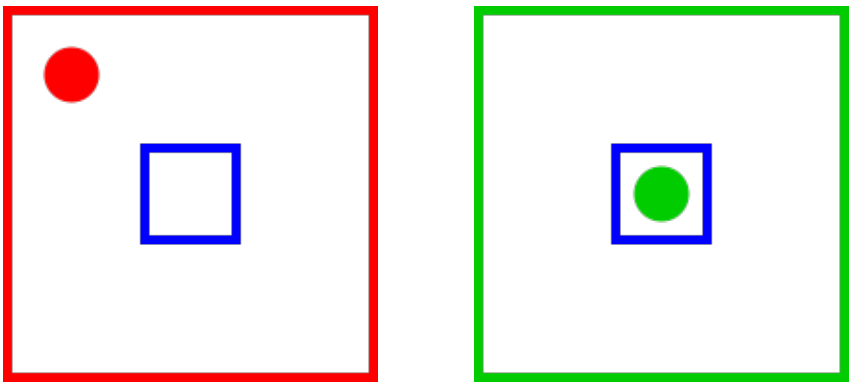
$$\hat{T}_{wj}(t) \approx T_{wj}(t_i) \exp(\tilde{V}_j(t_i)(t - t_i)) \tag{8}$$

We then manually tuned the control delay (which is the same as the prediction horizon) so that the synchronization error between the desired landing position and the actual drone pose is minimized. The final value we pick for the delay is 0.5 seconds.

2.3 Visualization Module

To help users observe the synchronization process between the two vehicles, we implemented a visualization module in our project. The visualization module consists of an LED matrix subsystem and an embedded software system. In the matrix consists of 64 LEDs, a 4-LED square represents the position of the drone. The outer frame on the matrix represents the

edge of the ground vehicle and the inner frame represents the optimal landing position for the drone. The outer frame and the 4-LED square will turn red when the drone is not within the proximity required for docking and will turn green when they are well-aligned. The inner frame will constantly stay blue. When the drone is too far away from the ground vehicle (0.6m), however, the whole LED matrix will turn off automatically to save power. An illustrative drawing for the visualization pattern is shown in Figure 7.



(a) Bad Alignment (b) Good Alignment

Figure 7: Alignment Indicator Illustration

2.3.1 Timer Frequency Design

The Neopixel 5050 LED [5] is controlled by a 800 kHz duty-cycle-varying PWM [2] signal. To generate such signal without consuming software clock cycles, we designed the embedded software to run on one of its hardware timer with DMA [3] channels built in. According to Figure 8, the main frequency for the micro-controller is 72 MHz. With prescaler division, the clock frequency at Timer 2, which is the hardware timer we use for generating the control signal, is 8 MHz. We can control the signal frequency and duty cycle by manipulating the values of ARR (Auto Reload Register) and CCR (Compare and Capture Register) on the micro-controller. More specifically, ARR holds the value which a counter resets at. CCR, on the other hand, holds the value that will be used in controlling the duty cycle. We can control the final refresh rate of our matrix by using the following equations: let f_o be the original frequency available (8MHz), f_c be the final clock frequency, $N_a a$ be the ARR value, N_c be the CCR value, and T_{on} be the high-voltage time for each duty cycle, then we have

$$f_c = \frac{f_o}{N_a} = 800\text{kHz} \tag{9}$$

$$T_{on} = \frac{1}{\frac{N_c}{N_a} \cdot f_c} \tag{10}$$

In our project, we set N_a to 10 to obtain the overall 800 kHz output signal. We also set N_c to 5 for digital one and 2 for digital zero. Plugging in those numbers and the corresponding T_{on}

[2] Pulse Width Modulation
[3] Direct Memory Access

for the two cases, we get 0.6μs and 0.15μs for the high voltage time respectively. According to the Neopixel datasheet [5], those two duty cycle periods fall within the required range for on and off signal detection.

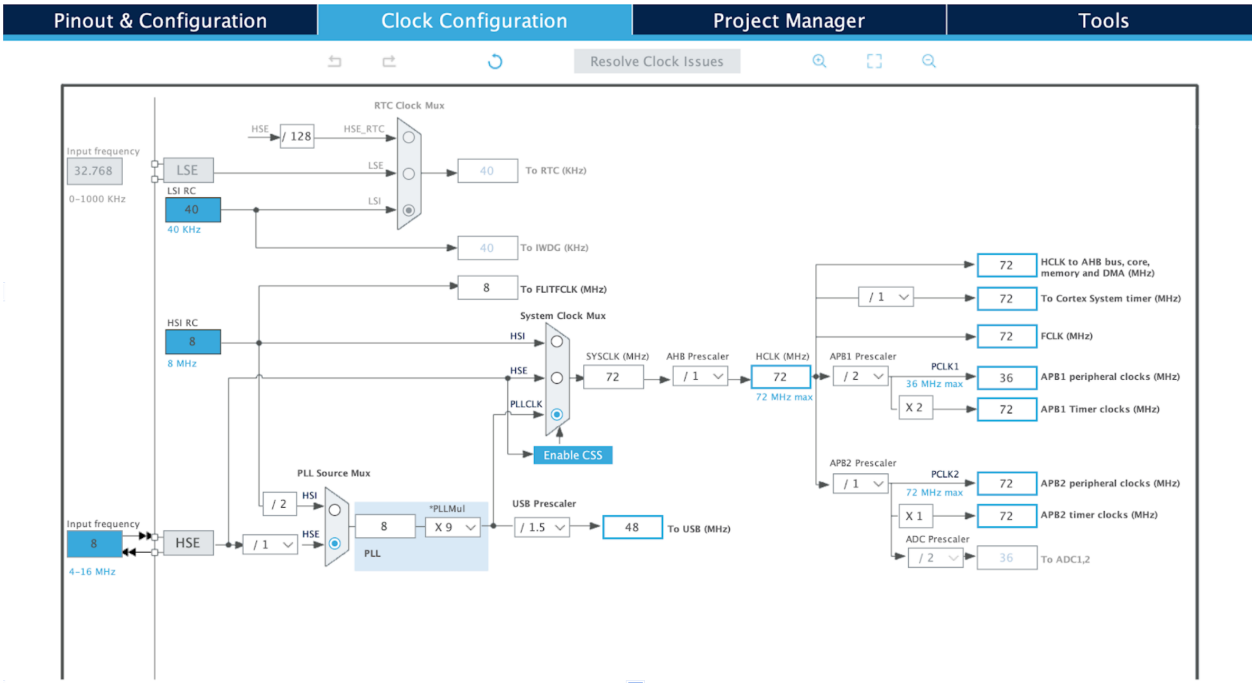


Figure 8: Clock Frequency Setting in CubeMX

2.3.2 Hardware system

The hardware system is designed to show the relative positions of the drone to the center point of landing/docking area on top of the ground vehicle by lighting LEDs. This subsystem mainly consists of two parts: LED matrix and STM32F103 micro-controller.

The LED matrix consists of 64 NeoPixel LEDs, all of which are connected in series as shown in Figure 18. These LEDs are powered by a Li-Po battery and is stable enough to function correctly thanks to our voltage regulator. The detailed specifications of the battery and regulator can be found in Appendices C.6 and C.7 correspondingly, and the schematics for them are shown in Figure 17.

The STM32F103 micro controller is in charge of getting information from upstream modules through USB serial port. The controller's schematics is as shown in Figure 15. This chip is also able to generate PWM waves directly through AAR and CCR (as explained in Section 2.3.1). Comparing with other commonly seen micro-controller such as Arduino, STM32F1 operates on higher frequency (72 MHz for our specific chip) and contains better internal timer and DMA support. As a result, we are able to achieve outputting high frequency signal without consuming CPU cycles. Moreover, STM32 also has its own built-in USB controller, saving developers the work for building serial communication and external

wire hazards. Together with these hardware components, we were able to present an accurate visualization of our synchronization between the drone and the ground vehicle.

2.3.3 Software System

The embedded software system is based on the code base generated by CubeMX. It enables the serialized communication between motion capture module and our STM32F103 MCU on the software level. As depicted in Figure 9, we used two threads. The USB driver thread is specifically in charge of receiving x and y coordinates from motion capture module and transform the received coordinates by 90 degrees counter-clockwise, since the orientation of the motion capture system and LED matrix systems is different. It then marks the two received coordinates as global variable so that the freeRTOS thread is able to use it. Finally, the USB driver thread set the signal to 1 so that freeRTOS thread can be waken up. It then processes the coordinates and decide what color to light up for each of the LEDs.

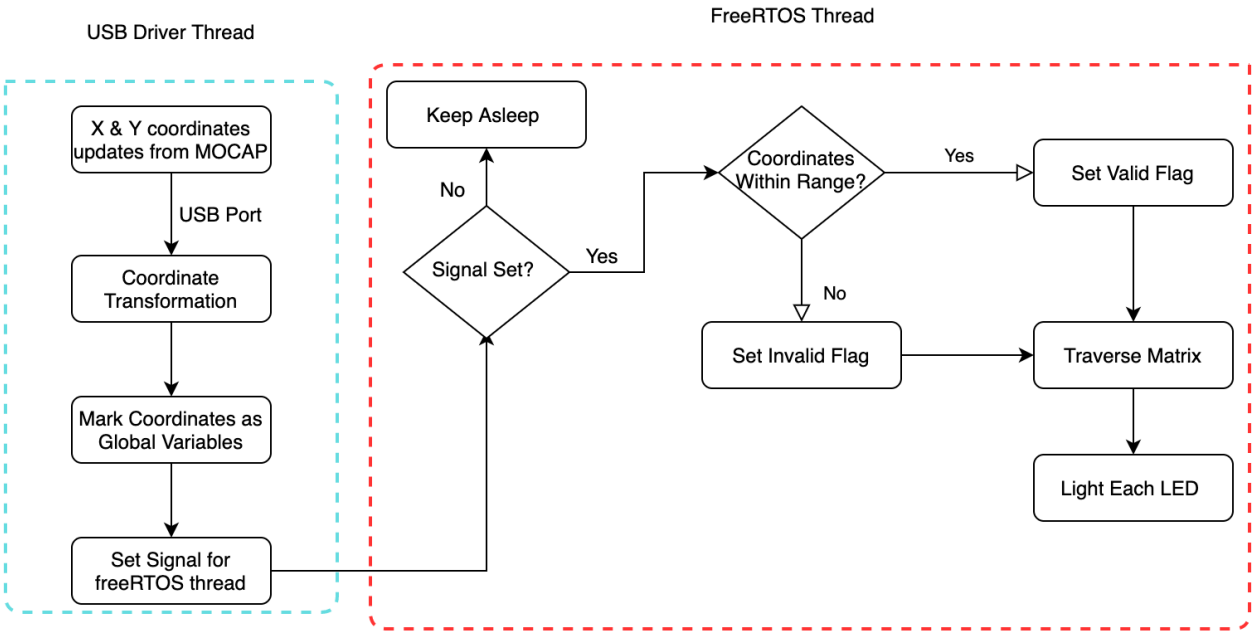


Figure 9: Flowchart of the Software System

3 Results and Verification

In this section, we will perform both qualitative and quantitative analysis to verify the success criteria of our design. For a more detailed list of requirement and verification table, see Appendix C.

3.1 Control Performance

3.1.1 Jackal Control

Since we do not assume all ground vehicles to be autonomously controlled in reality, the trajectory tracking performance for the Jackal is not the focus for this project. Nevertheless, we implemented a simple proportional controller for the Jackal bot to track time-parametrized smooth trajectories within certain error bound. As shown in Figure 10, there is a noticeable amount of control lag as well as tracking error when comparing with the desired trajectory. However, the overall tracking performance is good enough for it to mimic an autonomously planned vehicle following some arbitrary trajectory.

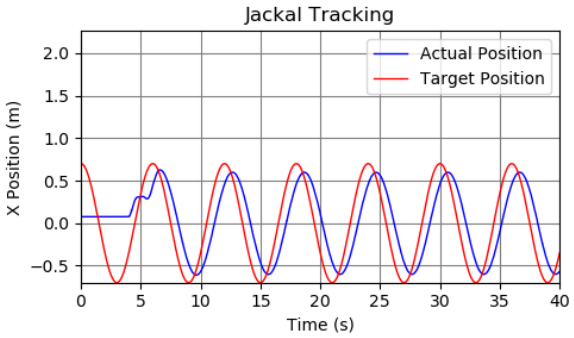


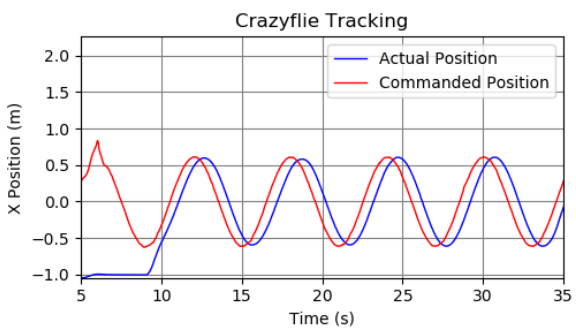
Figure 10: Jackal Tracking – X Coordinate

3.1.2 Crazyflie Control

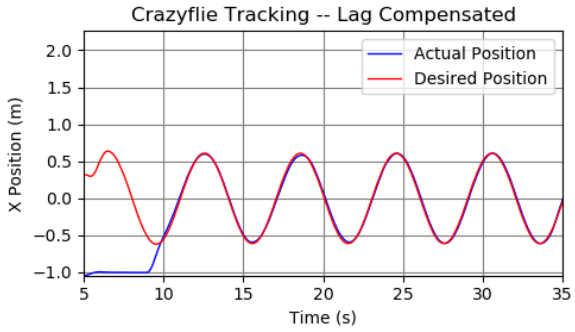
The flight control, on the other hand, is much more critical to have the drone dock accurately onto the desired landing spot. However, the on-board flight controller does not exhibit any predictive capability, which means there will also be a certain amount of control lag. Figure 11a shows that the tracking performance on the pre-planned circular trajectory is pretty good, but with some fixed delay to the commanded trajectory. Applying our proposed constant twist motion model compensation, we obtain close-to-ideal tracking performance which exhibits very low to zero tracking delay, as shown in Figure 11b. For the manually controlled Jackal, Figure 12 shows a similar comparison that demonstrated the qualitative effectiveness of our motion model lag compensation.

3.1.3 Synchronization Error

Using the lag compensated controller with the constant twist prediction model, the achieved tracking error is far below our proposed safe landing tolerance, which is 20 cm in radius.

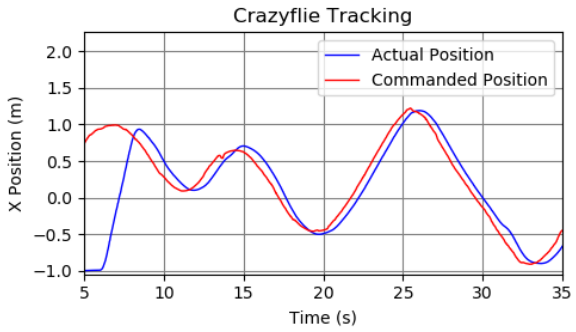


(a) Without Lag Compensation

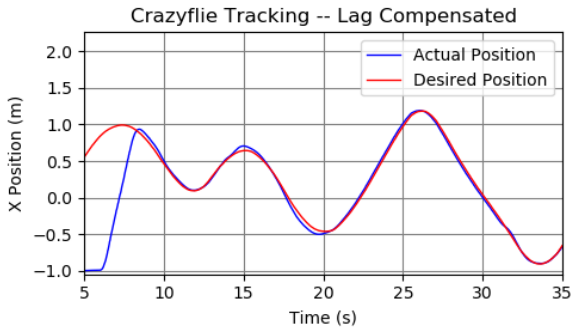


(b) With Lag Compensation

Figure 11: Crazyflie Tracking of Pre-planned Trajectory – X Coordinate



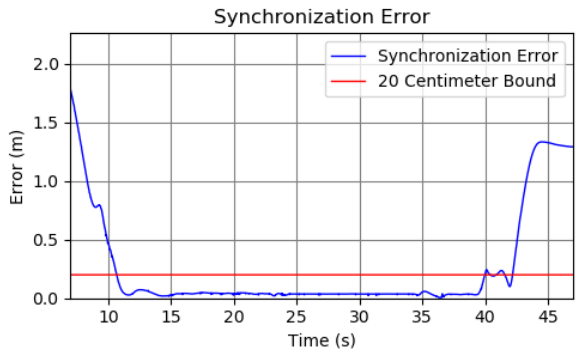
(a) Without Lag Compensation



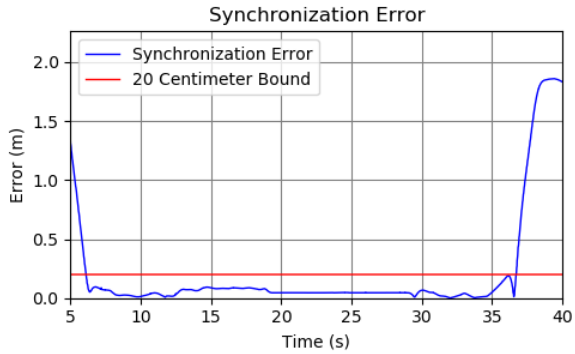
(b) With Lag Compensation

Figure 12: Crazyflie Tracking of Manual Trajectory – X Coordinate

Figure 13 shows that during the synchronization and landing phase, the relative position error between the drone and the desired landing spot on the Jackal stays below the tolerance line for both autonomously controlled and manually controlled Jackal.



(a) Autonomous Jackal



(b) Manually Controlled Jackal

Figure 13: Overall Synchronization Error

3.2 Visualization Results

Our visualization module is able to present the position of the drone with respect to center point of landing area. As shown in Figure 14.a, the visualization module has a red outer frame and a red 4-square object representing the drone when the alignment between the drone and the ground vehicle is not accurate. Moreover, as shown in Figure 14.b, the module completely shuts down when the drone is too far away. The module also functions correctly when the drone is within the accepting proximity to land, as shown in Figure 14.c. After analysis of videos taken,we found that our visualization module has a latency of around 10ms, and is hard to capture by the human eyes.

(a) In Range, Unable to Land (b) Drone Out of Range (c) In Range, Ready to Land

Figure 14: Visualization for Different Alignments

4 Cost Analysis

4.1 Material

Aside from basic lab infrastructures (the ground vehicle, the drone and the Vicon system), the material costs of our projects are listed in Table 1.

Component	Attributes	Quantity	Unit Price ($)	Total Price ($)
Capacitor	22pF	10	0.07	0.72
	0.1μF	20	0.08	1.52
	1μF	100	0.02	2.2
	10μF	10	0.14	1.44
Resistor	150Ω	10	0.05	0.54
	360Ω	20	0.07	1.44
	10KΩ	10	0.05	0.54
Voltage Regulator	3.3V@800mA	3	0.42	1.26
	5V@3A	2	1.26	2.52
Micro Controller	STM32F103	4	6.57	26.28
LED	NeoPixel 100-Pack	1	39.95	39.95
	10mA 0805	10	0.14	1.44
Level Shifter	8-Channel	3	1.1	3.3
Tactile Switch		10	0.31	3.12
Micro USB Header	R/A	4	0.68	2.72
JST Header	Vertical	3	0.17	0.51
	R/A	3	0.17	0.51
Crystal Oscillator	8MHz	5	0.44	2.2
Total				92.21

Table 1: Material Costs

4.2 Labor

Our estimated development cost is $40/hour, 10 hours/week for three people. This semester is 16 weeks long, therefore our total cost for development is:

$$3 * 40 * 10 * 16 * 2.5 = 48000$$

5 Conclusion

5.1 Accomplishment

In this project, we have successfully demonstrated the proof of concept of a possible solution to the last mile problem in modern drone package delivery system. Our collaborative control system achieved close proximity control between the ground and aero vehicle as well as a demonstration of robust docking onto moving ground vehicles. Even though we have made many assumptions that may not be available so easily to us in reality (e.g., the pose estimate of both the ground vehicle and the drone), we still managed to show that this solution is completely viable given enough known variables in the system.

5.2 Future Work

As mentioned in Section 5.1, we have made many assumptions. In real world, however, we might have to rely on computer vision to actually measure these parameters without introducing too much burden and memory bandwidth to the central server (or control system). Therefore one future direction we can think of is how to extrapolate the control algorithm to add a computer vision flavor. Another difficulty comes from actually scaling up the whole system. For instance, if we have multiple drones in the same area, how can we plan their motions so that each of them can be used efficiently and safely. In addition, the communication between different drones and the central server requires a significant amount of consideration before implementation. Last but not least, the whole system, on the highest level, could be generalized as an optimization problem, hence how to deploy such drones and their bases so that the maximum efficiency can be achieved remain a question.

5.3 Ethics and Safety

Although our project by itself casts little to no ethics or safety concerns because it is in a lab environment with comprehensive safety measures, as a proof of idea, it may raise the following issues:

- **Conflicts of Interests:** The successful deployment of such networks may significantly reduce the needs for labors in relevant industries, taking jobs from workers, and causing conflicts between companies and workers / unions. Such consequences could go against #3 of the IEEE Code of Ethics "to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist." [6] We currently do not have a solution for this and consider it far beyond our control.

- **Possible Unlawful Misuse:** Such a autonomous delivery system might offer more vacant for smuggling, taking advantage of the unsupervised time before the packages reaching the destinations, whereas increasing the difficulty for tracking such crimes. Such consequences, together with the next two in the list, would go against #1 of the IEEE Code of Ethics "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger

the public or the environment." [6] To avoid such unlawful activities and minimize their damage, recording every delivery specifications and detecting for contraband before the package is sent into the autonomous system is recommend.

- **Potential Hazard to Public Safety:** Aerial vehicles might cause serious secondary injuries under potential misbehavior of the ground vehicles since the drone can cause heavy impact and consequent explosion under high speed. UAV-related incidents are not unusual in today's society as shown by [7]. These experiments [8] suggest the serious aftermaths. In response, we should advocate that drivers to drive safely or use reliable auto vehicle systems to minimize the possibility of accidents as well as to build a emergent evasion response for the drones.

- **Privacy Concern:** In industries, the cyber-security measurement at ending terminals such as the drones could be overlooked. A breach can cause serious violation to public privacy. Potential misuse includes stalking and leaking private information. To protect the civic privacy, the whole system should be protected by reliable hardware / software security such that it is maintained and examined periodically.

With aforementioned concerns, some positive aspects are listed below:

- **Productivity:** Without doubt, autonomous delivery systems could tremendously increase the productivity. This benefit and the next point, help us to develop #1 of the IEEE Code of Ethics [6].

- **Service and User Experience:** Without human intervention, the delivery systems would avoid much mistakes of express and significantly improve the user experience.

- **Social Progress:** The wide use of such a system could push the progress of our society in many aspects, such as productivity, economy, legislation, cyber-security, and so on. This complies with #2 of the IEEE Code of Ethics "to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems." [6]

References

[1] G. B. Haberfeld, A. Gahlawat, and N. Hovakimyan, "Safe sampling-based air-ground rendezvous algorithm for complex urban environments," *CoRR*, vol. abs/2103.07519, 2021.

[2] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 37–42, 2017.

[3] C. R. Inc, "Jackal unmanned ground vehicle." https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/, 2021. Accessed: 2021-04-05.

[4] Wikipedia contributors, "Screw theory — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Screw_theory&oldid=1016439037, 2021. [Online; accessed 3-May-2021].

[5] Shenzhen LED Colors, *LED Color SK6812 Technical Data Sheet*, 4 2015. Rev. 3.

[6] IEEE, "Ieee code of ethics." https://www.ieee.org/about/corporate/governance/p7-8.html. Accessed: 2021-02-18.

[7] "List of uav-related incidents." https://en.wikipedia.org/wiki/List_of_UAV-related_incidents, Dec 2020. Accessed: 2021-02-18.

[8] E. Tegler, "What happens when a drone crashes into your face?." https://www.popularmechanics.com/flight/drones/a28774546/drone-head-collision/, Aug 2019. Accessed: 2021-02-18.

Appendices

A Software Design

All software are maintained publicly on Github.

A.1 Embedded Software

See https://github.com/alvinsunyixiao/embedded-example/tree/jz/usb.

A.2 Control Software

See https://github.com/alvinsunyixiao/crazy_land/tree/alvin/plot.

B PCB Design

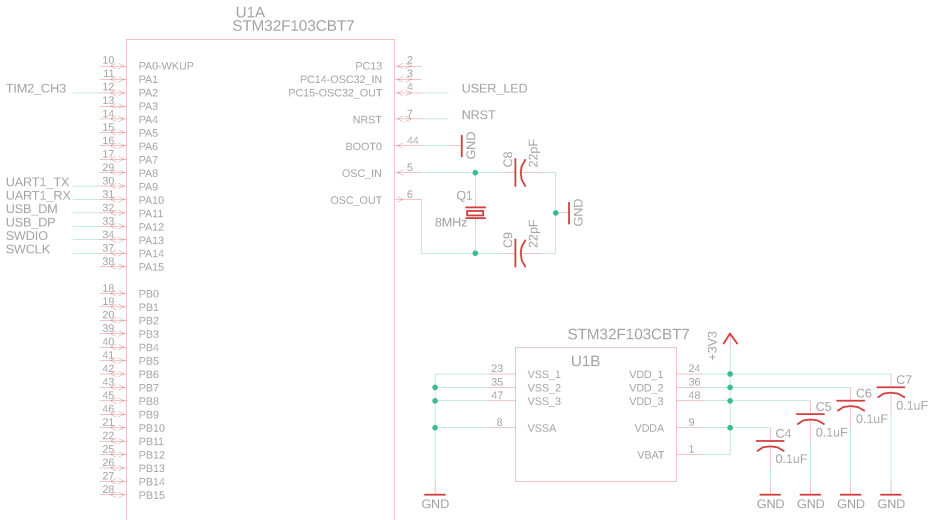


Figure 15: Schematics - STM32F103 Micro Controllers and Peripherals

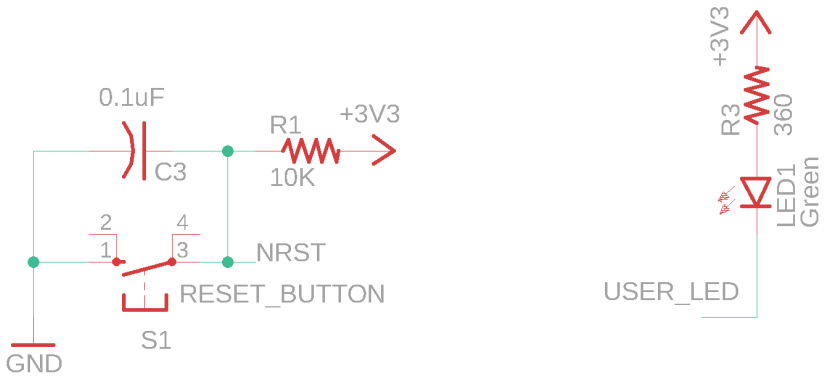


Figure 16: Schematics - User I/O

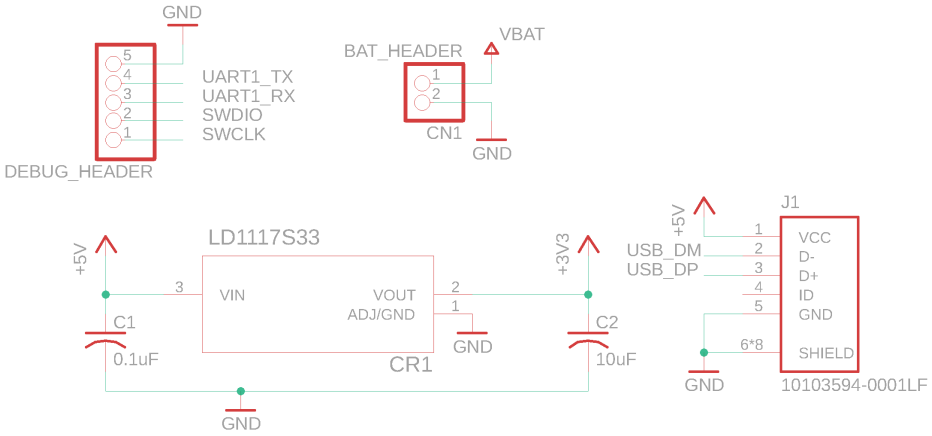
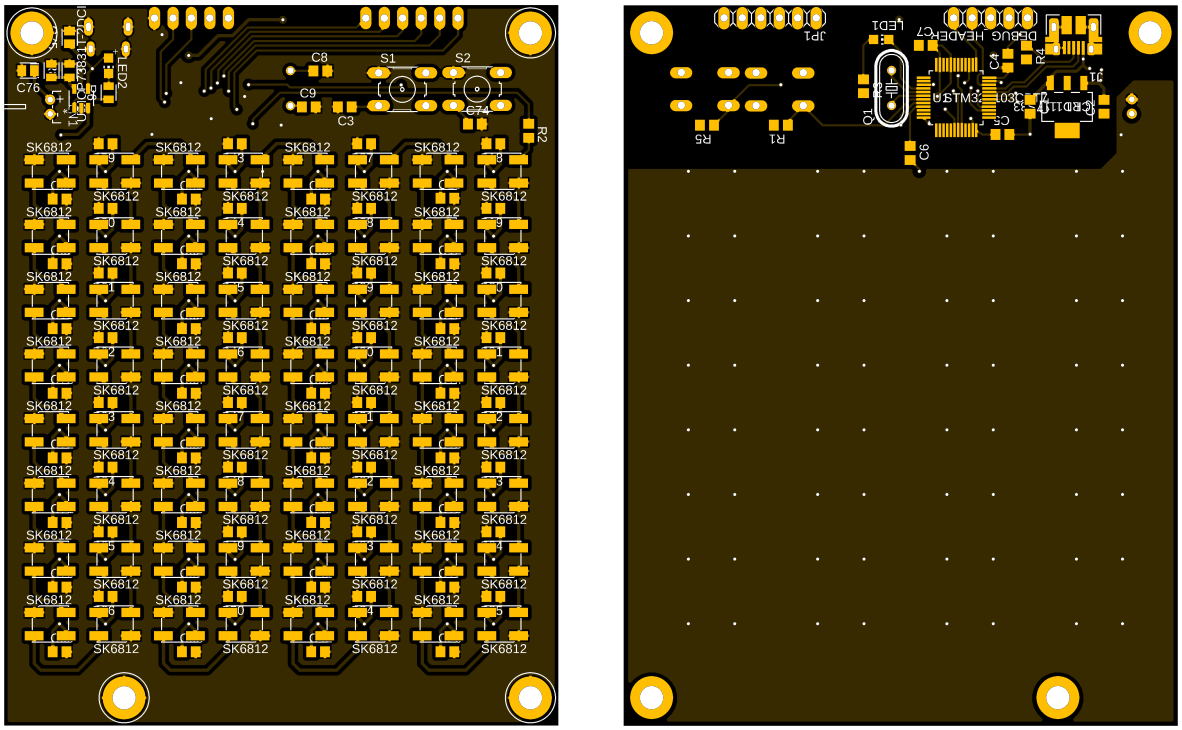


Figure 17: Schematics - Connection Headers

Figure 18: Schematics - LED Matrix



(a) Top Side (b) Bottom Side

Figure 19: PCB Layout

C Requirement & Verification Tables

C.1 Control Module-Mocap Decoder

Requirements	Verification	Result
The Vicon system should be able to locate the given object accurately and broadcast its corresponding coordinates to all the devices within the predetermined communication channel based on the local Wi-Fi system.	Specify a certain channel and use a PC to access it. Print the value and check if it matches with the object's coordinates in Vicon.	Yes

Table 2: RV Table for Vicon and Mocap Decoder

C.2 Control Module-Ground Vehicle Trajectory Generator

Requirements	Verifications	Result
This module should be able to generate a sequence of preplanned way points that are inside the bound of the Vicon motion capture system.	Numerically test all generated way points against the allowed rectangular boundary defined by the Vicon motion capture system and make sure non of them are outside the boundary. Also plot the generated trajectories before executing the low-level trajectory tracker to ensure the way points resembles simple geometries such as circles and polygons.	Yes

Table 3: RV Table for Ground Vehicle Trajectory Generator

C.3 Control Module-Low-level Car Controller

Requirements	Verifications	Results
The controller should be able to connect to and pair the ground vehicle with a PS4 controller and allow the user to move the car with it.	By taking the measurements from Vicon, we can get the actual trajectory that the controller executed. We can then compare the execution with the generated trajectory and verify that the controller does track achieve those way points within 20cm radius accuracy.	Yes

Table 4: RV Table for Low-level Car Controller

C.4 Control Module-Drone Trajectory Generator

Requirements	Verifications	Results
1. The trajectory generator should allow the drone to follow the car within a proximity of 30cm of the vehicle.	1a. Start to time the process when the drone and the car starts to move.	
	1b. After the drone successfully synchronize with the car, move the car in random directions and check if the drone could still follow it autonomously.	Yes
	1c. Check the visualization module on the car to make sure it is always green.	
2. The synchronization process mentioned above should be done within 20 seconds.	2. If step 2 is finished, check the timer to make sure the process takes less than 20 seconds.	Yes

Table 5: RV Table for Drone Trajectory Generator

C.5 Control Module-Low-level Flight Controller

Requirements	Verifications	Results
The controller should be able to connect to and pair the drone with a controller and allow the user to move the drone with it.	1. Pair the controller. 2. Try to move the drone with the controller in all directions.	Yes

Table 6: RV Table for Low-level Flight Controller

C.6 Visualization Module-Power Regulator

Requirements	Verifications	Results
The voltage regulator should be able to handle voltage of 4.5V-5.5V and convert it to a DC voltage at around 3.3V.	1. Connect the voltage regulator with a 5V power source. 2. Measure the output voltage and make sure it is within the range listed in requirements.	Yes

Table 7: RV Table for Power Regulator

C.7 Visualization Module-Li Po Battery

Requirements	Verifications	Results
1. The battery should be able to provide a voltage of 3.2-4.2V to fulfill the requirements of the voltage regulator.	1. Connect the battery with an multimeter. 2. Make sure the voltage across the battery is within the required voltage range. 3. Connect the battery with a resistor of 2.5 Ohms.	Yes
2. The battery should be able to provide a current of no less than 1.92A. Since each battery is used to power 32 LED lights, each with an operation current of 60mA.	4. Measure the current using the multimeter to make sure that the current is higher than 1.92A.	

Table 8: RV Table for Li-Po Battery

C.8 Visualization Module-STM Controller

Requirements	Verifications	Results
1. The MCU should be able to take an input voltage of 3.3V± 0.1V.	1. Connect the MCU with a voltage source of the given required voltage.	Yes
2. With the given voltage range, the MCU should output a voltage between -0.5V to -1V as logic "0" to the LED and output a voltage above 3.3V as logic "1" to the LED.	2. Measure the output pins that are going to be used for LEDs individually and confirm that each of them fulfills the required voltage.	Yes

Table 9: RV Table for STM32F1 controller

C.9 Visualization Module-Chained RGB LED Matrix

Requirements	Verifications	Results
1. The LED can operate under a voltage of 3.7±0.1V.	1. Connect the LED to a power source with a voltage that fulfills the given requirements.	Yes
2. The data input voltage of the LED should be able to interpret a voltage above 3.7V as a logic "1" in its output, and interpret a voltage below -0.5V as a logic "0" in its output.	2. Feed the Din pin with a voltage around 3.7V and check if the light is on.	Yes
	3. Feed the Din pin with a voltage around -1 to -0.5V and check if the light is off.	Yes

Table 10: RV Table for Chained RGB LED Matrix

C.10 Software System

Requirements	Verifications	Results
1. The software system can freely pass commands to lower level systems	1. Use software to light an LED with certain features (colors, duration etc.).	Yes
2. The latency of the software system does not cause updating/flashing frequency of LEDs to lag 50ms.	2. Take a video and analyze each frame. Verify that the updating frequency of LED matrix is at least 20Hz (20 frame per second).	Yes

Table 11: RV Table for Software System