Developing an Autonomous Aerial Package Transport System

New Mexico Supercomputing Challenge Final Report

April 5th, 2023

Team #28
Los Alamos High School

Team member:
Andres Iturregui

Teacher:
Allan Didier

Project Mentor:
Clint Hubbard

Executive Summary:

In the last decade, the idea of having packages delivered via autonomous drones has become very popular. Continual advancements in computational and aerospace technology make this goal easier to achieve each year. The successful implementation of delivery drones could have many benefits. The need for delivery drivers could be eliminated, and deliveries would be much faster. The trillion-dollar company Amazon has already developed UAVs (unmanned aerial vehicles) that are capable of transporting packages from a warehouse to a nearby location [6]. However, these deliveries are very expensive. Amazon's current generation of delivery drones cost an estimated $146,000 per unit, and deliveries are expected to cost Amazon a whopping $63 per package [7].

The goal of this project was to develop the hardware and software of a delivery drone that is significantly less expensive than Amazon's current delivery drones. This was achieved by using hobby-grade components designed for racing drones along with custom 3D-printed components. The drone is equipped with a carbon fiber frame, brushless motors, and electronic speed controllers to power these motors properly. A sonar sensor tells the drone its distance above the ground, and a GPS module provides location data. A mechanism to hold a package was designed using servo motors and a custom 3D-printed latch. A downward-facing camera is also present for recording video. Two separate flight computers are combined to maximize the functionality of the drone.

Under manual control, the drone is very stable in the air and flies as expected. After a flight controller replacement and many code changes, an autonomous flight was executed successfully. The autonomous flight consisted of the drone taking off with a package attached, flying to a predetermined GPS location, releasing the package at a lower altitude, and returning to the takeoff position. Furthermore, it was calculated that the drone could potentially carry a package weighing up to 618 grams.

The successful flights and results show that it is undoubtedly possible to create an autonomous delivery drone for a much lower price than Amazon does. The drone cost about $500, making it less than 1% of the price of an Amazon delivery drone. That being said, the drone could be improved by having higher quality sensors, a redesigned package pickup system, and more software features.

Problem/Solution

Modern delivery systems make the transportation of any type of package easier than it has ever been. Large semi-trucks and airplanes transport large numbers of packages long distances, and smaller trucks perform deliveries on a local level. These methods of delivery, however, raise several issues. For one, semi-trucks are bad for the environment because of their carbon dioxide emissions. Furthermore, deliveries can never be completed in under an hour using trucks. Fortunately, a new technology promises to solve these issues: drone delivery.

If implemented successfully, package delivery using UAVs (unmanned aerial vehicles) would both reduce carbon dioxide emissions and allow for deliveries to be completed as quickly as fast food delivery. Because of these benefits, Amazon has focused on developing delivery drones for nearly a decade [2]. They have even started deliveries in two U.S. cities using their UAVs that transport the package from an Amazon warehouse to the customer's household [6]. However, even these deliveries have a big flaw: price. Amazon's current drones cost an estimated $146,000 per unit, and each package costs about $63 to deliver [7].

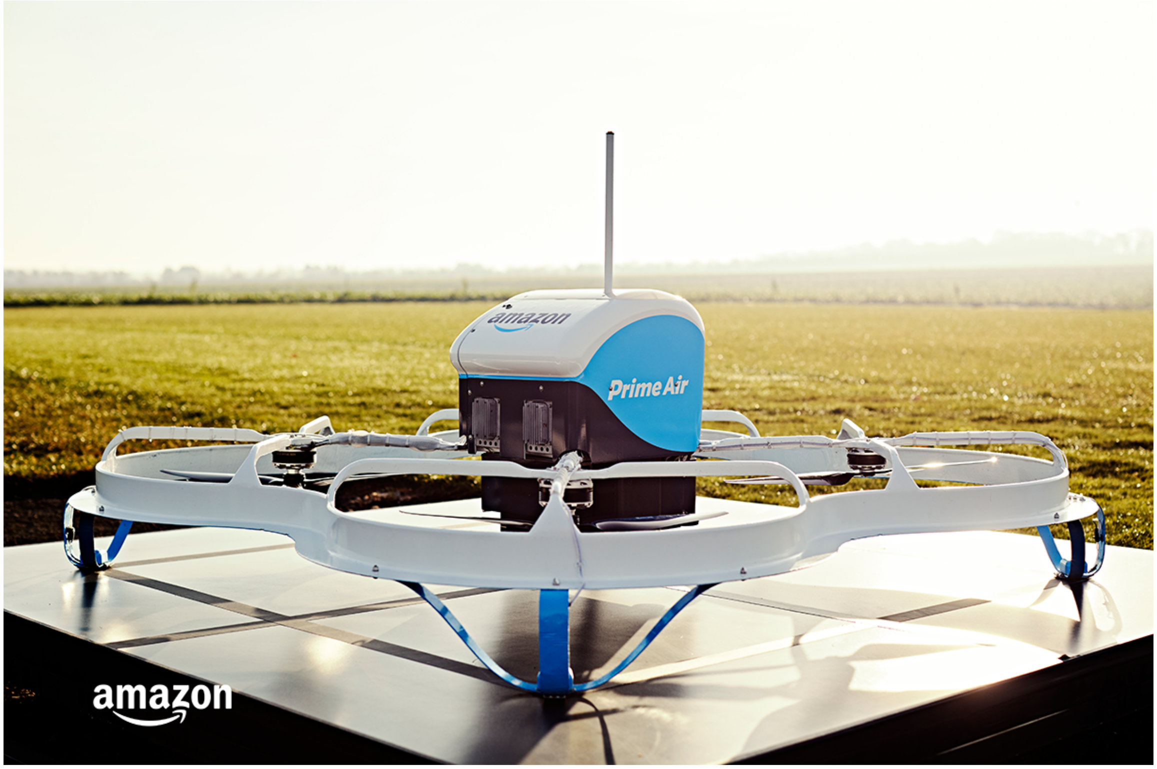

Figure 1. A quadcopter designed by Amazon [2]

The goal of this project was to develop a delivery drone significantly cheaper than Amazon's current model. This can be achieved by using functional hobby-grade components instead of expensive industrial equipment. The end goal was for the drone to take off with a package, travel to a particular location, release the package, and return to its takeoff position completely autonomously. Furthermore, manual remote control of the drone is also implemented to provide a safety backup for autonomous failures.

Hardware Design

When choosing and designing the hardware of the drone, several guidelines were kept in mind. For one, the hardware needed to be inexpensive yet still reliable and effective. Since each part of the drone was ordered or made separately, the components needed to be compatible with one another. Finally, the flight computers needed to be customizable and lightweight.

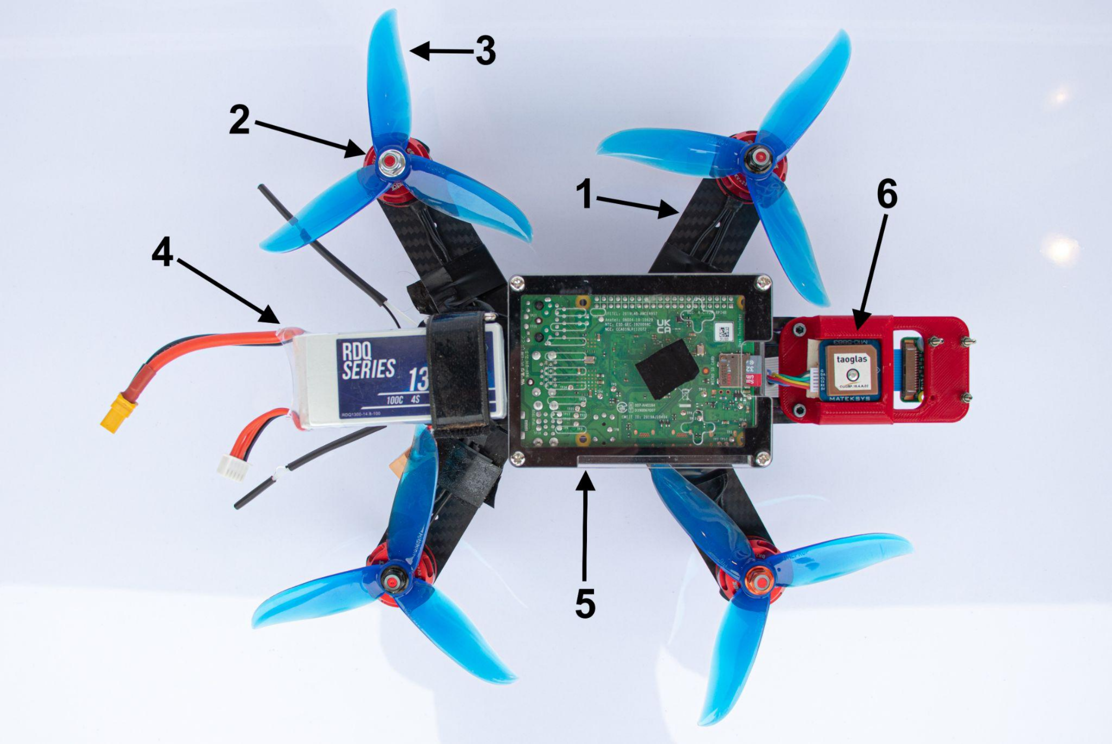

Figure 2.1. A top view of the quadcopter with labeled components.

1. Frame- The drone is held together using a carbon fiber frame designed for racing drones. Carbon fiber is advantageous for several reasons: it is lightweight, stiff, durable, and reasonably priced. It is considered a quadcopter frame because it supports the mounting of 4 motors. This was chosen over a 6-motor or 8-motor setup because having fewer motors is simpler and less obtrusive to the package pickup system.
2. Motors- The motors chosen are brushless motors designed for racing drones. Brushless motors were chosen over brushed motors because they are more powerful and last much longer. The motors are also larger than the motors on similarly-sized camera drones because the extra torque will be important for carrying a larger payload.
3. Propellers- The propellers chosen have a 5-inch diameter because this is the largest propeller that won't hit the frame. These propellers are also very lightweight and durable enough to survive a light crash.
4. Battery- A LiPo (lithium polymer) battery is used to power every component of the drone. The LiPo battery chemistry was chosen over other battery chemistries because these batteries can provide higher rates of current, which is important for carrying a package.
5. Raspberry Pi- A raspberry pi is a small yet powerful single-core computer that can be programmed using a variety of different languages. More information about its function can be found under *Flight Computers*.
6. GPS- A GPS (global positioning system) and compass module provides the flight controllers with latitude, longitude, and cardinal orientation. This data is crucial for successful autonomous navigation. It is held in place using a custom 3D-printed bracket.

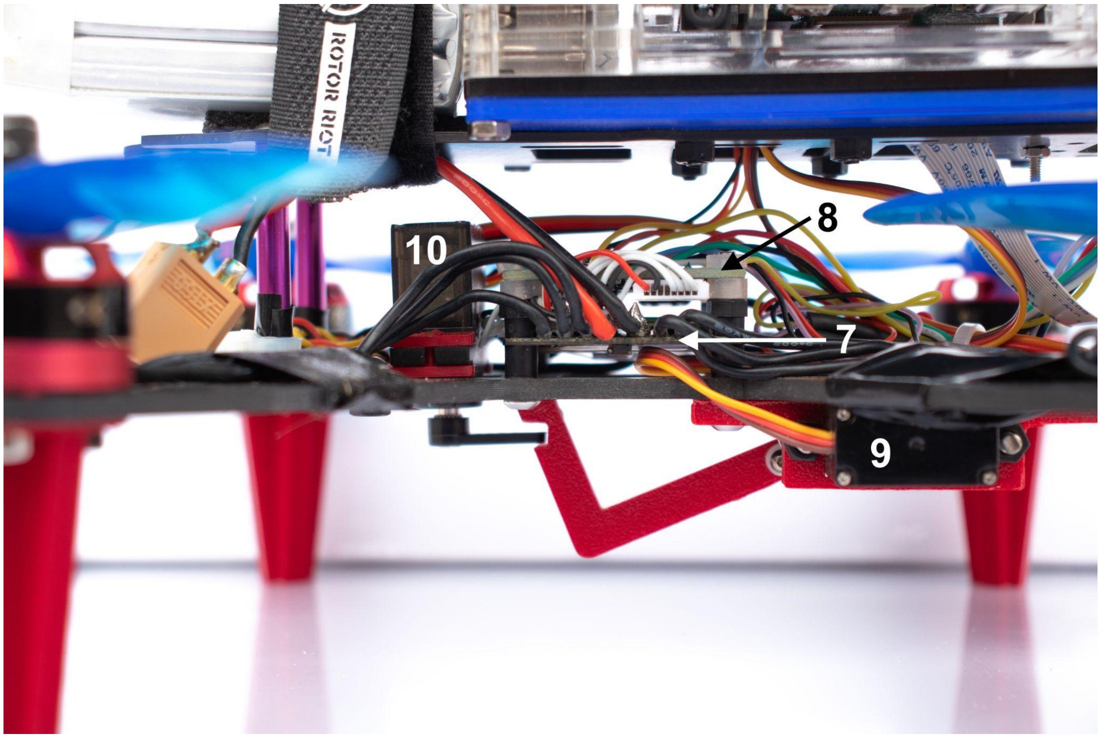

Figure 2.2. A side view of the quadcopter with labeled components.

7. ESC- The ESC (electronic speed controller) is a necessary component for powering brushless motors. The flight controller tells it how much power each motor should receive and it powers the motors accordingly.
8. Hobby-grade flight controller- The second flight controller on the drone is a Matek H743 Slim V3, which is a small microcontroller designed specifically for drones. Instead of being programmed with custom code, this controller runs an open-source software called Ardupilot. More information about this flight controller's function can be found under *Flight Computers*.
9. Package hook mechanism- The mechanism for holding and releasing packages consists of a servo motor with a custom 3D-printed hook attached to it.
10. Hook locking servo- When the hook mechanism is in the closed position (as shown in Figure 2.2), this servo can lock it into place, reducing the strain on the other servo.

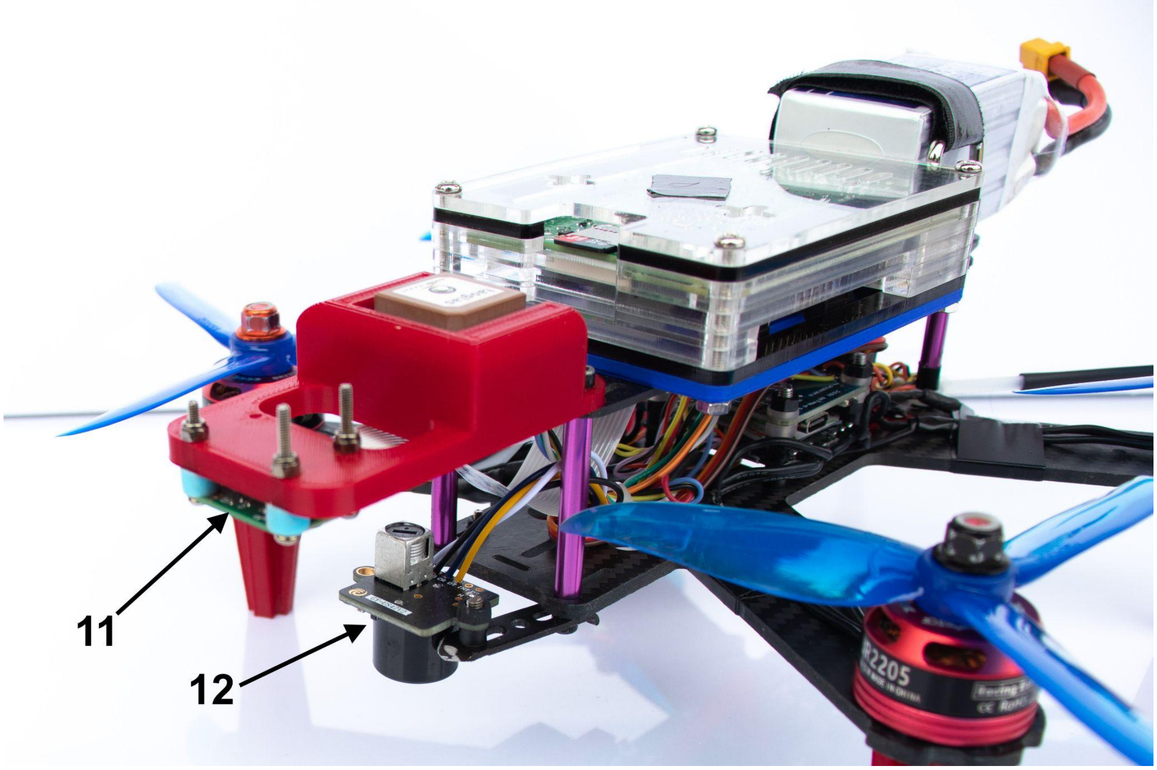

Figure 2.3. A front view of the quadcopter with labeled components.

11. Camera- A downward-facing camera is capable of recording video during the flight. It also provides the potential for advanced computer vision-based autonomous functions.
12. Sonar sensor- This downward-facing sensor can measure the distance between it and the ground. This is important because the drone needs an accurate altitude reading to successfully fly autonomously.

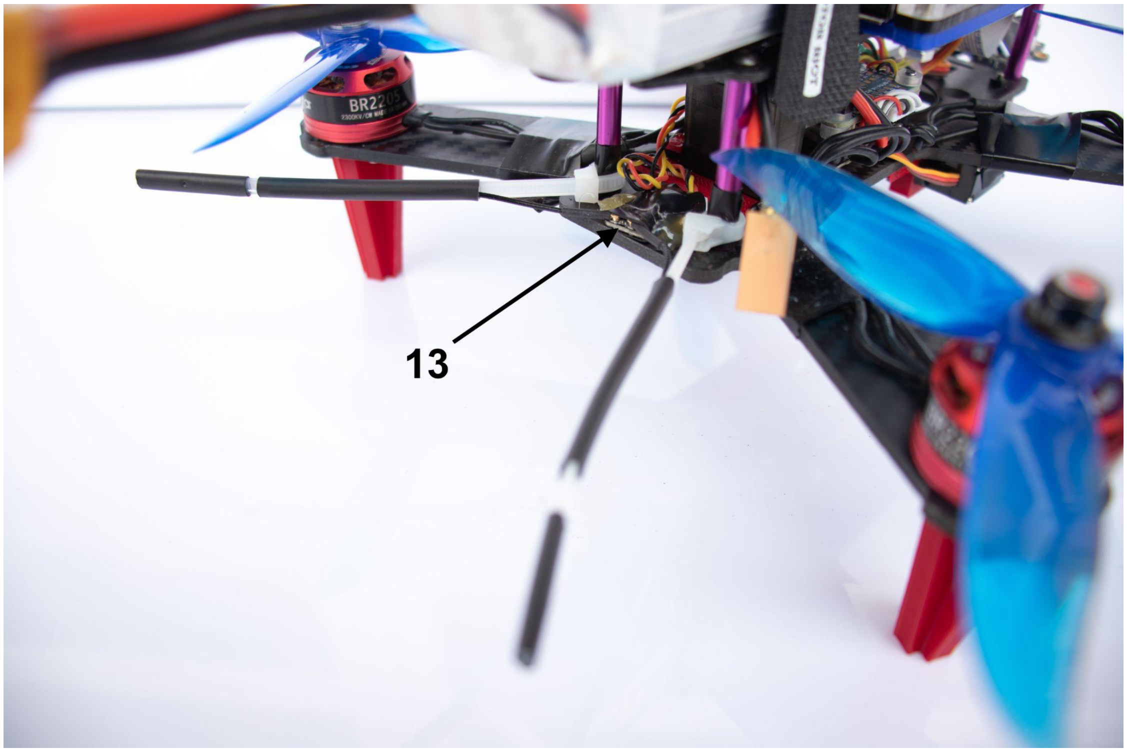

Figure 2.4. A back view of the quadcopter with labeled components.

13. Receiver- The receiver communicates with an RC (remote controller), which makes manually controlled flights possible. Manual control is important for being able to test the drone and take over an autonomous flight that went wrong.

Flight Computers

Unlike most hobby UAVs, the drone is equipped with not one, but two flight controllers. Both flight controllers have different strengths and weaknesses, so combining them maximizes the functionality of the quadcopter. Both flight controllers are communicating with one another at all times during the flight via a protocol called MAVLink. MAVLink (Micro air vehicle link) is very fast and efficient for transmitting and receiving data and is designed specifically for UAVs. [9].

The first flight controller, a Matek H743 Slim V3, is a small, hobby-grade flight controller designed specifically for UAVs. It runs a flight control software called Ardupilot, which is open-source and designed specifically for RC vehicles [3]. Because Ardupilot has been developed for over 10 years, it is well-refined and reliable [3]. The Matek flight controller is directly responsible for communicating with the GPS, compass, receiver, and sonar sensor. Using all of this data, it can spin the motors such that the drone behaves as expected and responds correctly to all RC inputs.

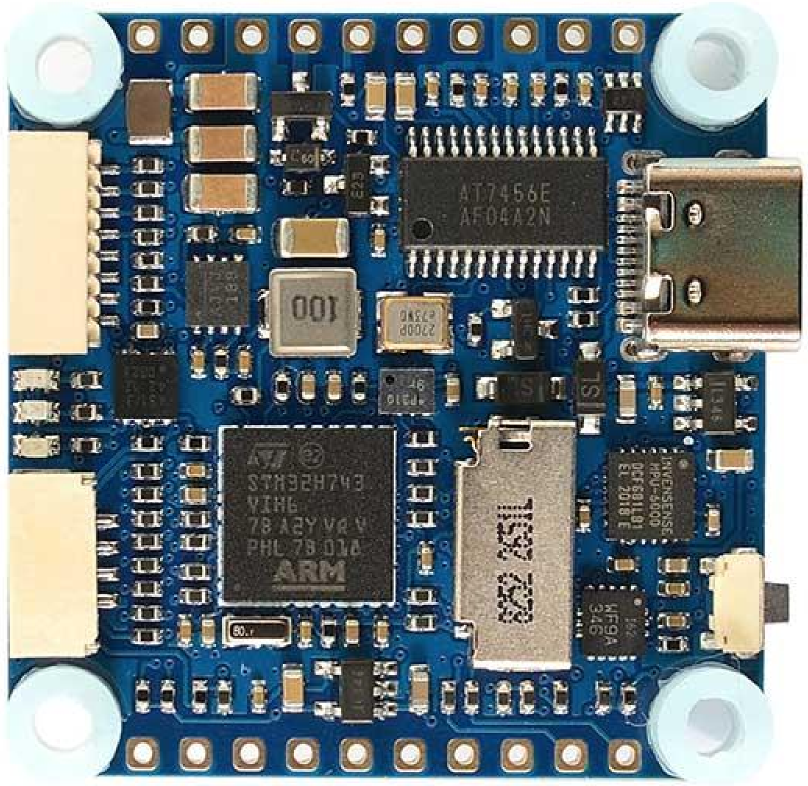

Figure 3. A close-up image of the Matek H743 V3 Slim. [8]

The second flight controller is a Raspberry Pi 4 Model B, which is a larger, single-core computer that is comparable in functionality to a laptop or desktop computer [12]. The Raspberry Pi runs a custom Python script that allows it to:

- Communicate with the Matek flight controller.
- Record and save video from the onboard camera.
- Control the servos that release or hold on to a package.
- Record data including GPS position, altitude, acceleration, battery voltage, autonomous flight status, and RC joystick positions.
- Send the Matek flight controller the GPS coordinates and altitude it should travel to.
- Tell the Matek flight controller whether it should be in autonomous or manual flight mode based on an RC switch's position.

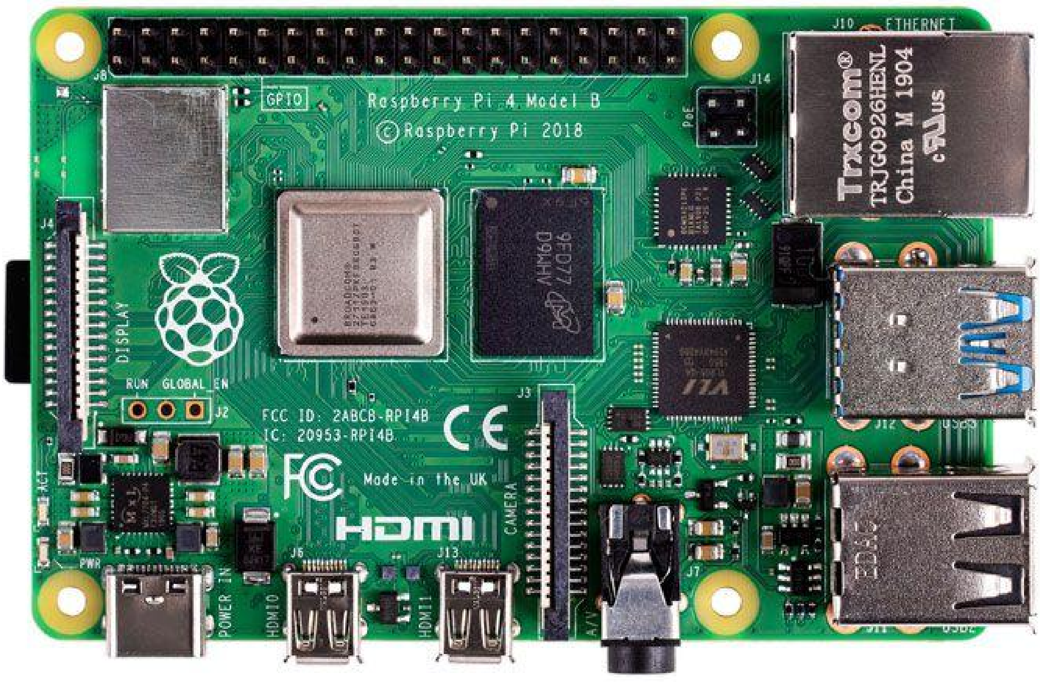

Figure 4. A close-up image of the Raspberry Pi 4 Model B. [11]

Flight Code

Throughout the entire flight, the Raspberry Pi is running a custom Python script. To simplify the launch procedure for the drone, the code is set up to automatically start running shortly after the Raspberry Pi is powered up. Furthermore, the code will stop running when a particular RC switch is held for 3 seconds. On startup, the code will run through the procedure below:

1. Configure the output pins that connect to the servos so that they can easily be accessed later in the code.
2. Attempt to connect to the Matek flight controller. If the connection fails, try once every 5 seconds until the connection is successful.
3. Create three separate files for recording data. The first file will record all of the raw, unedited messages that the Matek flight controller sends. The second file will record most of the data the flight controller sends in a spreadsheet format two times per second. The third file will record acceleration data in a spreadsheet format 100 times per second.
4. Send the Matek flight controller a message requesting that it send battery voltage, altitude, GPS position, RC positions, system status, and mission status twice per second. A message is also sent requesting that acceleration data be sent 100 times per second.
5. Send the Matek flight controller the mission plan. This is essentially a series of GPS locations and altitudes for the drone to travel sequentially.
6. Begin a segment of code that executes in a loop thousands of times per second. The code does the following during each iteration:
    a. Check for any new messages sent by the Matek flight controller.
    b. If there is a new message, extract the data from it and save it to the three files if it is the correct time to do so.
    c. If the RC switch that controls the package hook has been moved, power the servos accordingly.
    d. If the flight mode switch has been moved, send a message to the Matek flight controller requesting a flight mode change.
    e. If autonomous mode is enabled, start recording video.

f. If a waypoint has been reached, record that information to the three files and request that the next step in the mission is started.
g. If the mission is completed, stop recording on the camera.
h. Check if the code-ending RC switch has been held for more than 3 seconds. If it has, terminate the loop.

7. Once the loop is terminated, the code does a few more things before finishing:
    a. Stop the camera recording if the autonomous mission wasn’t completed successfully.
    b. Disable the servo motors.
    c. Save all three data files and close them.

Test Flights

To reach the end goal of completing autonomous package transportation, several test flights were performed. The videos of the test flights are linked in Appendix B.

Manual Control Test Flights

The first test flights completed were manual control test flights. The goal was to ensure that the drone was capable of flying and responding correctly to RC inputs before attempting any autonomous flight.

The drone flew successfully on its first flight; however, the pitch was inverted because attempting to make the drone fly forward made it fly backward and vice versa. This issue was easily fixed by changing a configuration setting in the Ardupilot software. Once this was changed, the drone flew very well—it was very responsive to RC inputs and quite powerful.

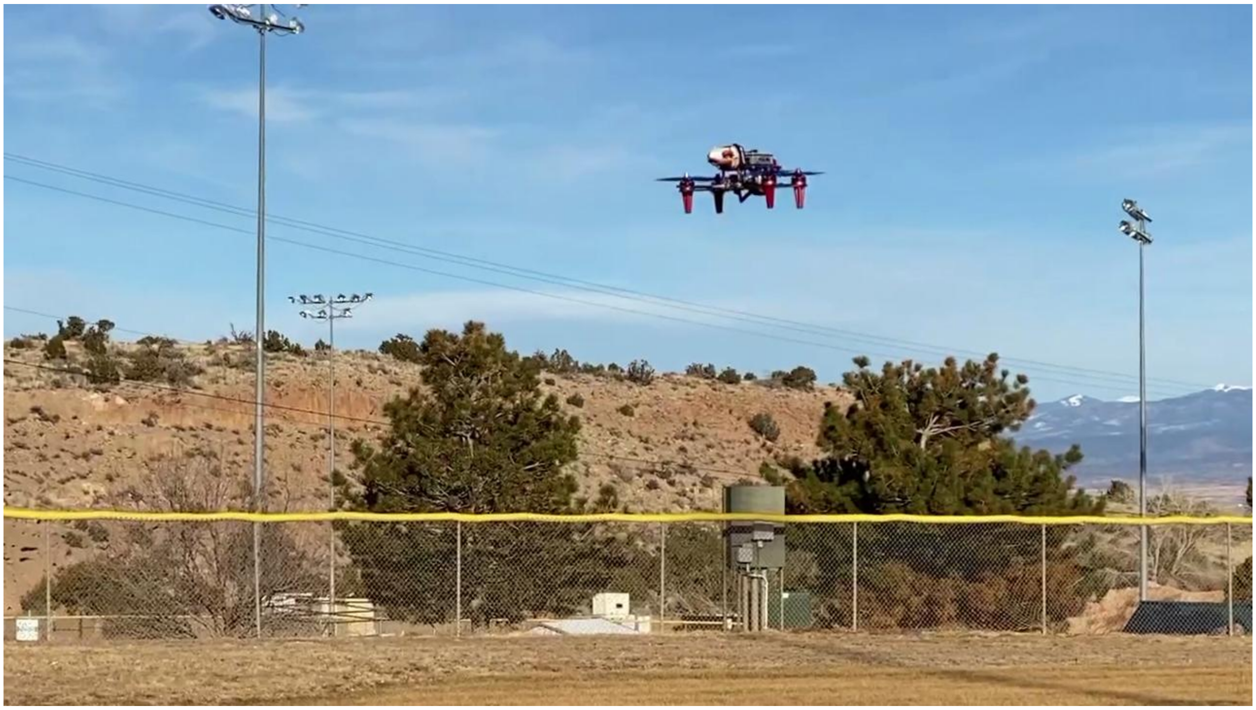

Figure 5. The drone is hovering under manual control.

Failed Autonomous Test Flights

During initial autonomous flight tests, there were countless failed flights. For most of these flights, the autonomous flight mode RC switch was enabled, but the drone wouldn't switch to autonomous flight. This issue occurred because the Python code sent the mission plan in an incorrect format.

After this was fixed, the next issue encountered was poor altitude control. The drone would take off and begin traveling to its destination, but it would do so at an altitude much higher than what was set. Once it reached the destination, it would have significant fluctuations in altitude. The cause of this was the hobby-grade flight controller using barometer altitude, which is measured using air pressure, instead of using the sonar sensor. Resolving this required a new flight controller that supported the use of a sonar sensor in autonomous flight.

Before a successful flight was reached, there was one final issue encountered. During the flight, the drone would take off successfully to the desired altitude of 3 meters and begin traveling to its destination, but it would quickly stop, turn around, and return to its takeoff position. The issue was revealed by the altitude data recorded by the Raspberry Pi.

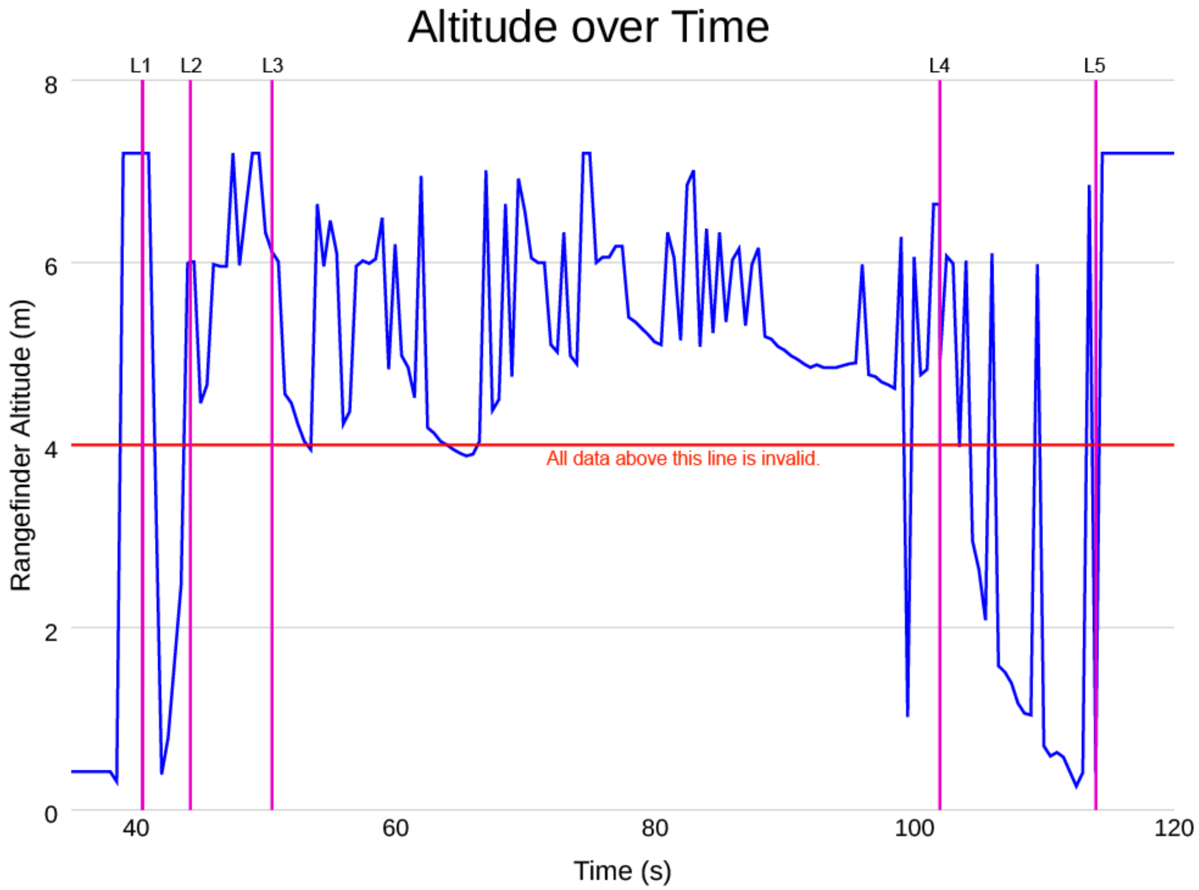

Figure 6. The altitude measured by the sonar sensor over time during the failed autonomous flight.

Figure 6 shows the altitude reading given by the sonar sensor throughout the flight. Because of the sensor's limited range, any readings given that are above 4 meters are invalid. We can see that the altitude is almost always invalid during the flight, which indicates why the drone returned to its home position. This feature is implemented in Ardupilot for safety reasons. The issue was resolved by reducing the drone's target traveling altitude from 3 meters to 1.5 meters.

Successful Autonomous Flight

After reducing the traveling altitude from 3 meters to 1.5 meters, a successful, fully autonomous flight was achieved. Before the flight, the drone had a small, lightweight package attached to the bottom of it. During the flight, the drone was able to take off to its target altitude, fly about 20-30 feet to its preset destination, reduce its altitude, release the package, return to its home position, and land smoothly.

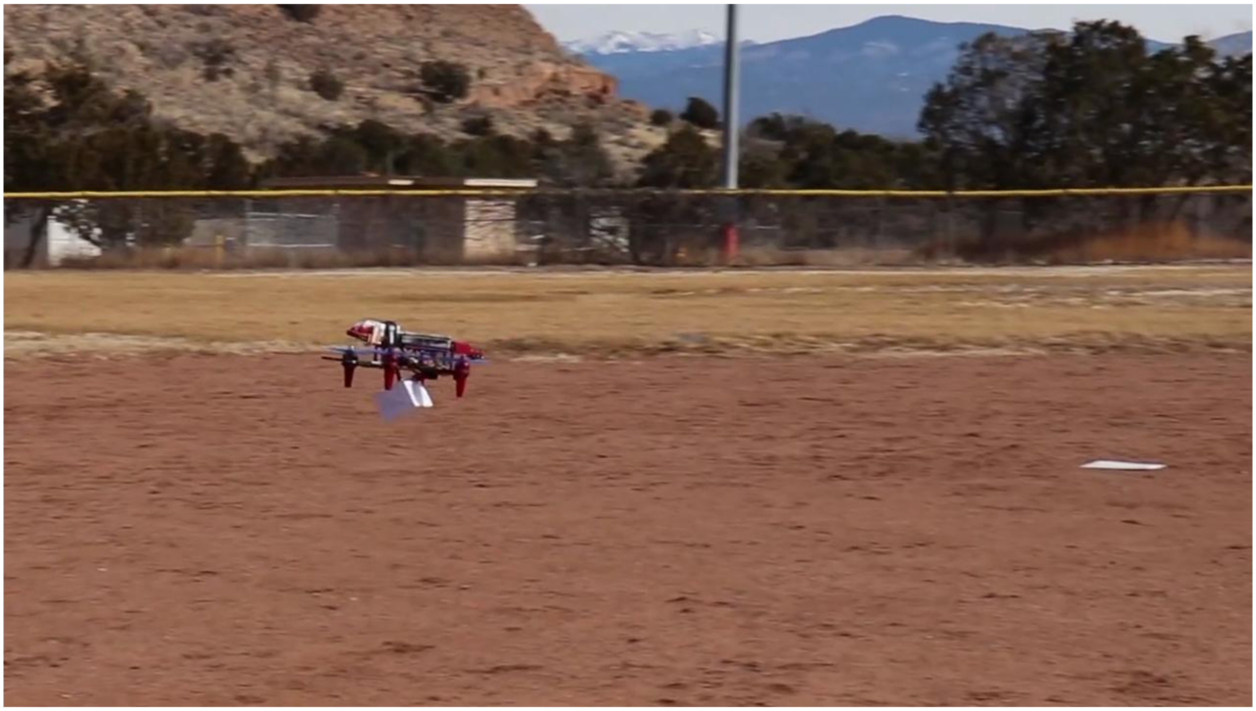

Figure 7.1. The drone flies towards its destination with the package attached underneath.

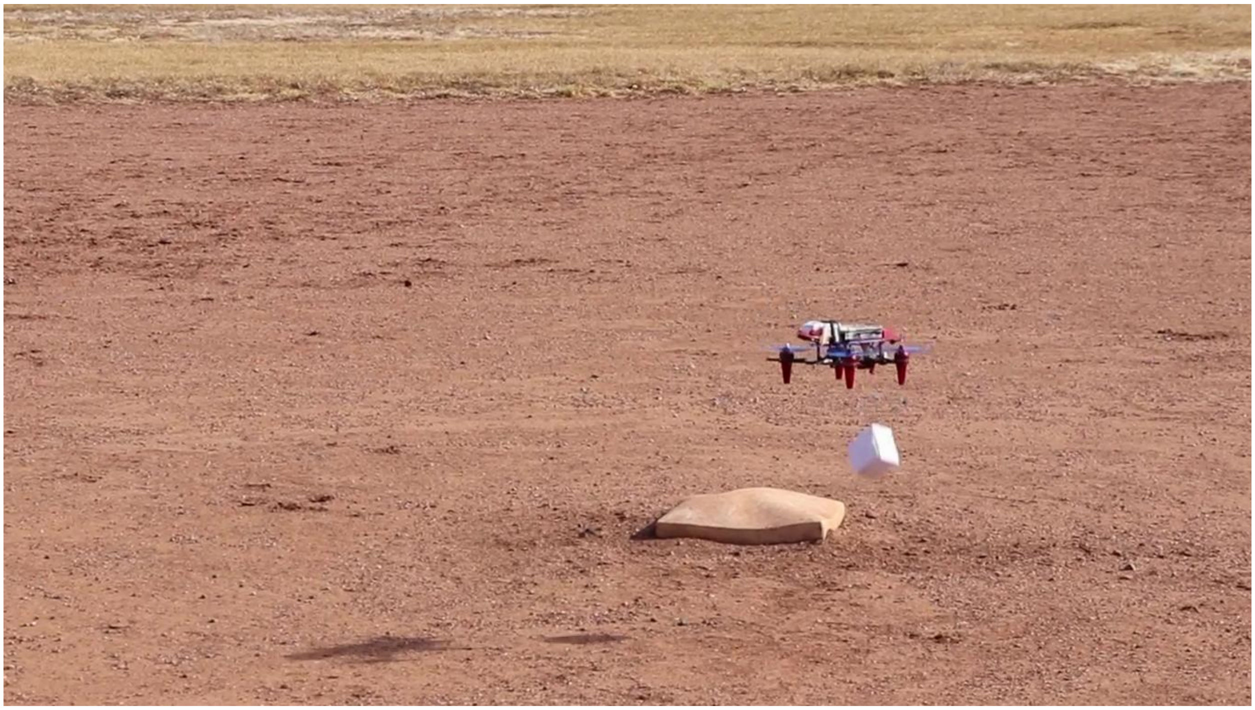

Figure 7.2. The drone releases the package at a lower altitude.

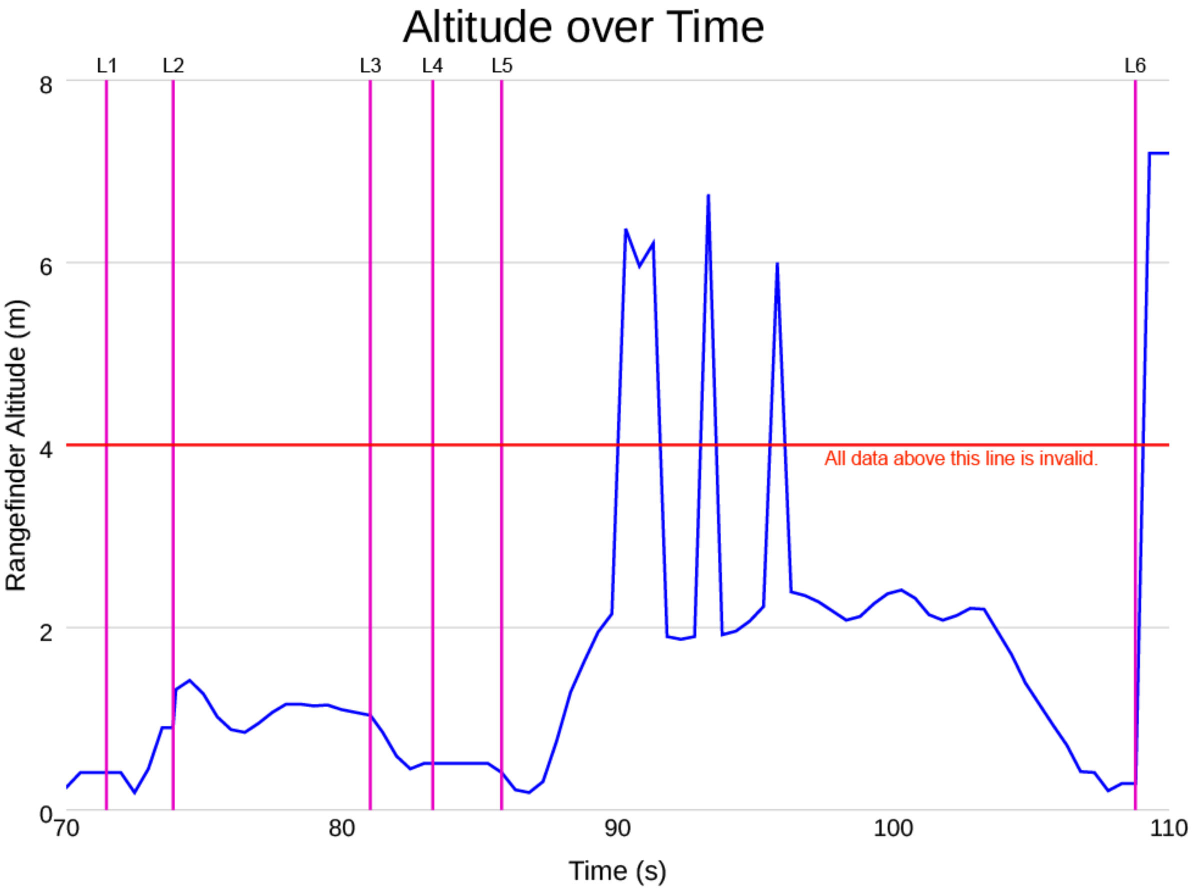

Figure 8. The altitude measured by the sonar sensor over time during the successful autonomous flight.

Figure 8 shows the altitude given by the sonar sensor throughout the flight. We can see that most of the data provided by the sensor is now valid because of the reduced altitude. Fortunately, the flight controller can tolerate invalid data for brief periods of time.

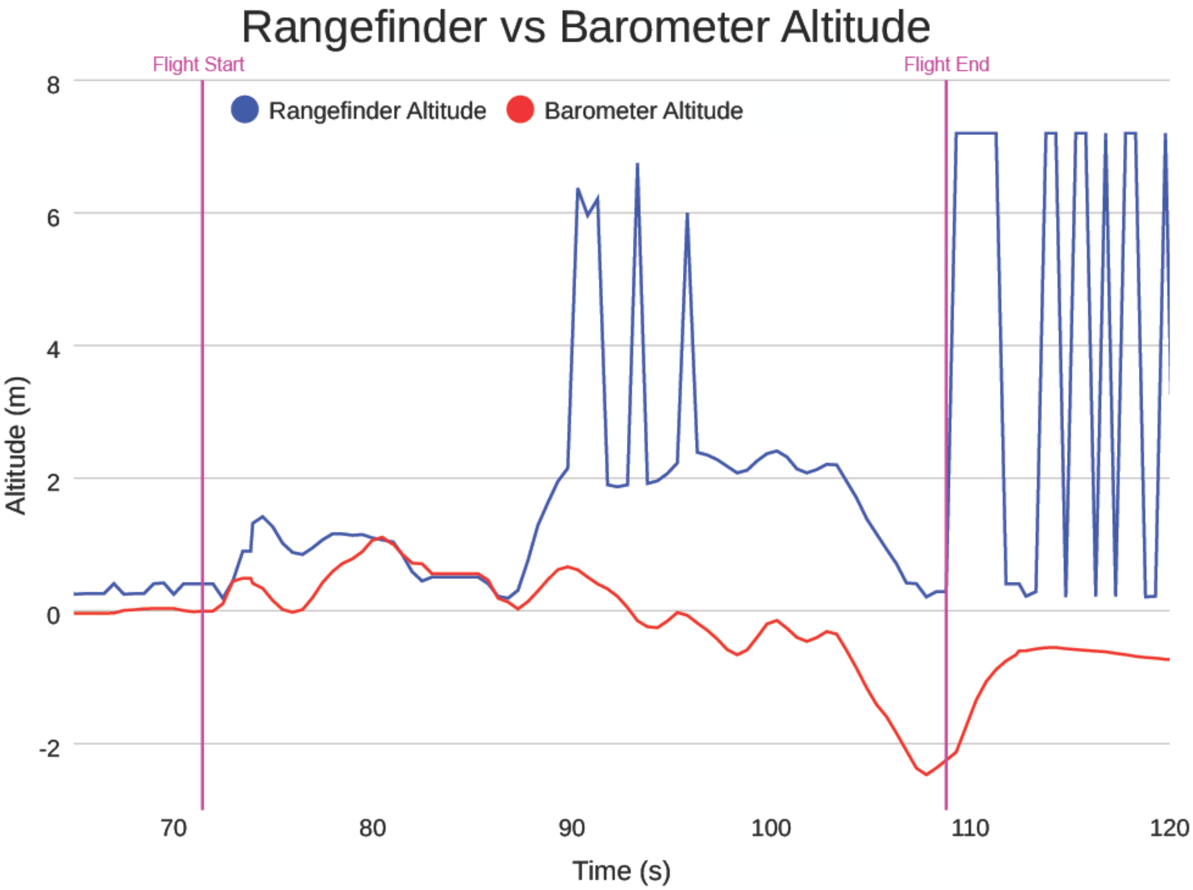

Figure 9. The altitude measured by the sonar sensor compared with the altitude measured by the barometer.

Figure 9 shows a comparison between the altitude measured by the sonar sensor and the altitude measured by the barometer. We can see that especially later in the flight, the barometer reading begins to drift and become inaccurate. Even though the sonar sensor isn’t always able to get a valid altitude reading, the valid readings it does get are much more accurate and precise than the barometer readings.

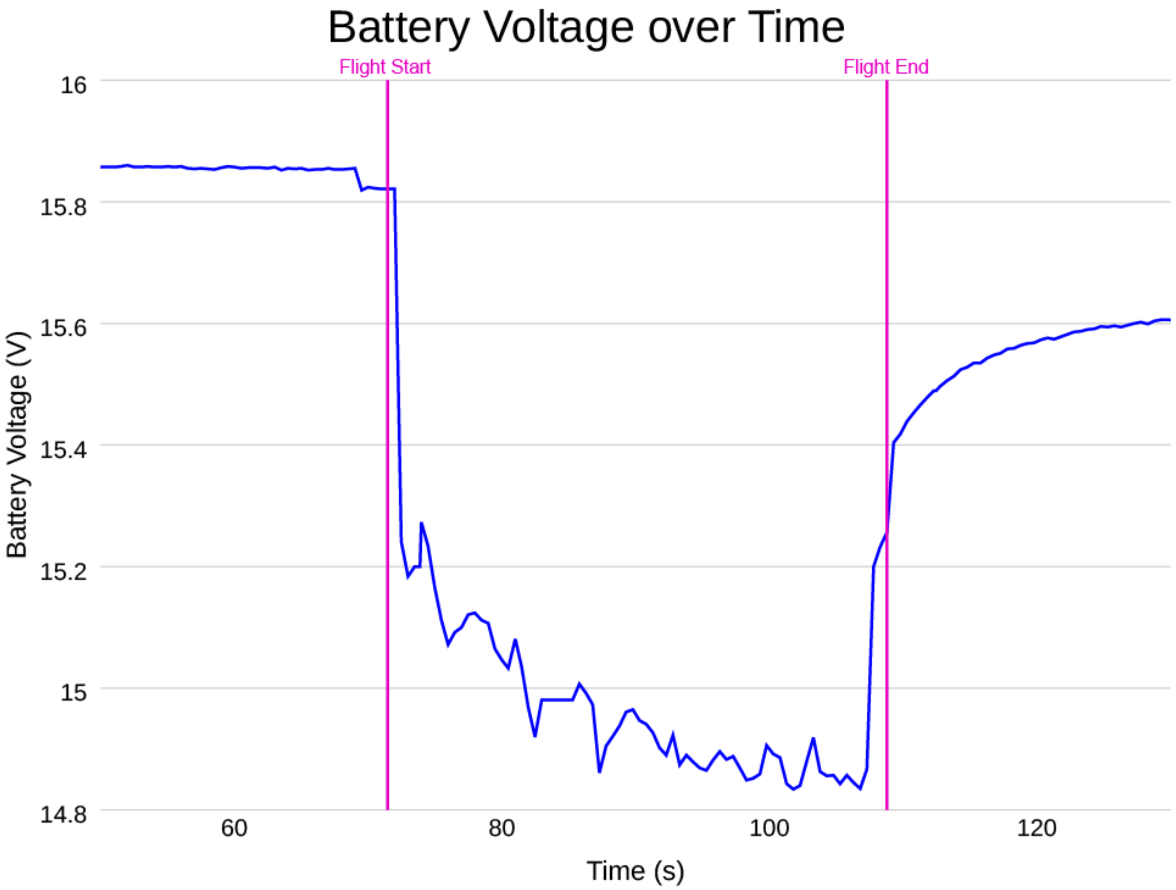

Figure 10. The battery voltage throughout the flight.

Figure 10 shows the voltage of the battery throughout the flight. As soon as the quadcopter takes off, the voltage sharply decreases before decreasing steadily throughout the flight. After the drone lands, the voltage rises before equalizing. This shows how the battery voltage is lower when the motors are demanding a lot of electrical current.

Acceleration Testing

While performing autonomous flight tests, the package used was very lightweight because of safety concerns. However, it is still important to see how much weight the drone could potentially carry. To determine this, maximum acceleration tests were performed. During the tests, the drone was controlled manually and set to hover. After a stable hover was reached, the throttle was set to maximum for about a second before the drone was brought back down slowly to a lower altitude. This process was repeated four times, with the Raspberry Pi recording acceleration measurements 100 times per second throughout the flight.

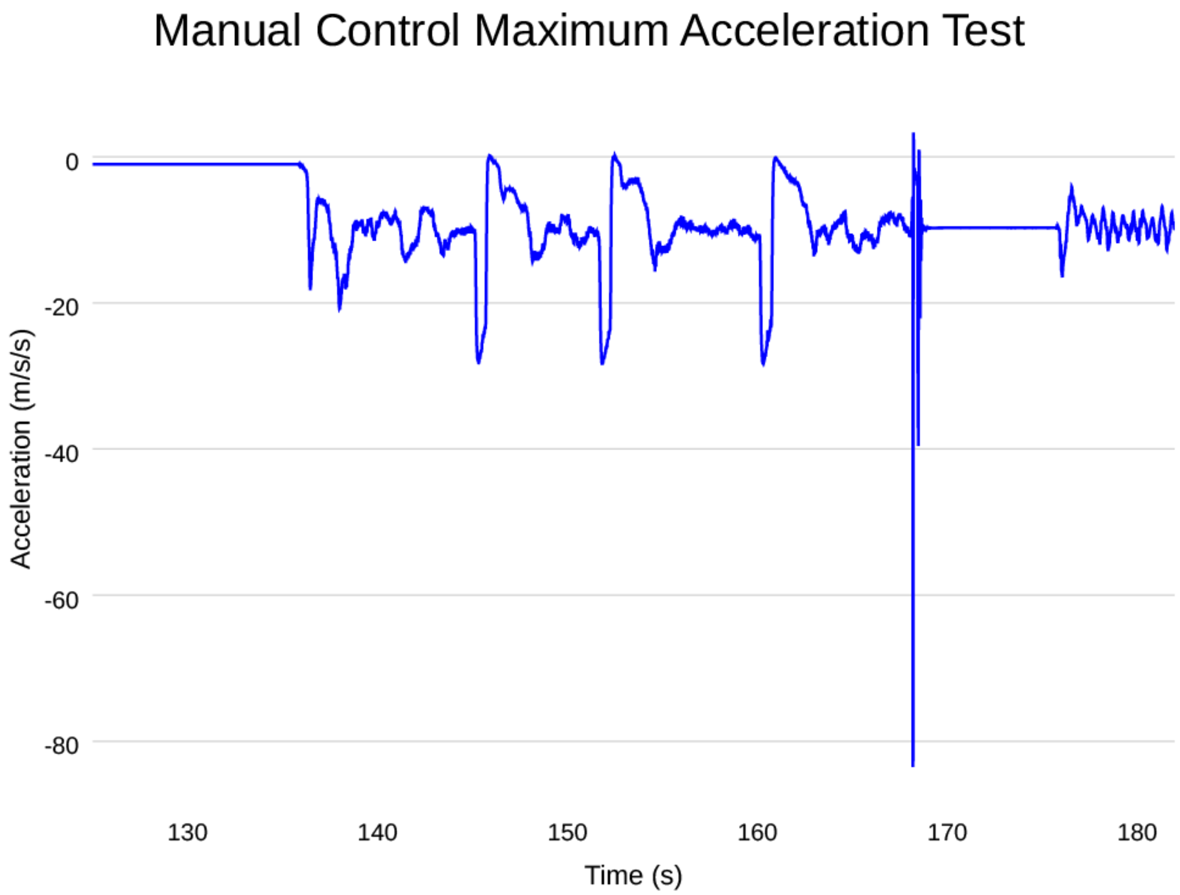

Figure 11. The acceleration measured during the manual acceleration test flight.

Test number	Maximum measured acceleration (m/s^2)
1	28.2
2	28.47
3	28.28
4	83.5
Average (not including test 4)	28.317

Figure 12. A table containing the maximum accelerations measured during the flight.

The first step in finding the maximum payload is calculating the average. Because the data from test 4 was a clear outlier, it was not used in the average. Furthermore, the acceleration reads 9.81 m/s^2 when the drone is laying flat on the ground, which means it is including acceleration due to gravity. This must be accounted for by subtracting 9.81 m/s^2 from the average, which gives us 18.507 m/s^2.

Next, we can find the net force acting on the drone using Newton's second law,

$$F_{net} = m \times a$$

where F_{net} is the net force acting on the drone, m is the mass of the drone, and a is the net acceleration of the drone. Since the drone mass is 0.669 kg and the acceleration is 18.507 m/s^2, we find that the net force acting on the drone is 12.381 N (newtons).

The net force on the drone is composed of two main forces: the force of the motors pulling the drone upwards and the force of gravity pushing the drone down. We can find the force of gravity acting on the drone using:

$$F_{gravity} = m \times 9.81\, m/s^2$$

This shows that the force of gravity is 6.563 N. We can find the force exerted by the motors using the equation:

$$F_{net} = F_{motors} - F_{gravity}$$

which gives us F_{motors} = 18.944 N.

To maintain control of the drone, it should have at least a 1.5:1 thrust-to-weight ratio. This means we can have a drone weight of up to 18.944 N / 1.5 = 12.629 N. We can then divide by acceleration due to gravity (9.81 m/s^2) to get our total maximum drone weight of 1.287 kg. Finally, we can subtract the drone’s initial weight of 0.669 kg to get a maximum payload weight of 0.618 kg. This means that the drone can fly properly while carrying a package weighing up to 618 grams.

Conclusion

To summarize the project, a drone was constructed using hobby-grade components and custom parts. A dual flight controller system allowed for autonomous flight to take place, and a latch design made the drone capable of holding and releasing packages that could potentially weigh up to 618 grams. Furthermore, the quadcopter cost about $500, making it 0.33% of the price of Amazon’s current delivery drone. Overall, the results show that it is possible to reduce the costs of delivery drones and make them more accessible to smaller companies. However, many improvements can be made to the drone to improve its overall performance and functionality:

- Upgrading the sonar sensor to a light-based sensor would allow for flights at higher altitudes.
- A forward-facing sensor could make the drone capable of obstacle avoidance.
- An improved package holding mechanism could set the package down more gently.
- The camera could be used to determine if dropping the package is safe.

Appendix A

List of Abbreviations and Symbols

a	Net acceleration of the drone
ESC	Electronic speed controller
$F_{gravity}$	The force exerted on the drone by gravity
F_{motors}	The force exerted by the motors
F_{net}	The net force acting on the drone
GPS	Global positioning system
LiPo	Lithium polymer
m	Drone mass
MAVLink	Micro air vehicle link
RC	Remote controller / Remote controlled
UAV	Unmanned aerial vehicle

Appendix B

URL Links to Flight Videos and Code

Video	URL
Manual control test flight	https://drive.google.com/file/d/1xDRlMNbT8KVDx1ZXClDrK5_HpKEo1KO5/view?usp=share_link
Last failed autonomous flight	https://drive.google.com/file/d/1aYcqf1iDtflcksIg_nN7iNMW6jgFJOg0/view?usp=sharing
Successful autonomous flight	https://drive.google.com/file/d/1pDVOCCU7aZKUkBdns47OvH32njr6mp7E/view?usp=sharing
Maximum acceleration test flight	https://drive.google.com/file/d/1Ix5KAadoGLhRZi4LbvTQuGmsPiYa4H_R/view?usp=share_link
Raspberry Pi Python code	https://drive.google.com/file/d/1xukzD_IZYwxGVunoFbH-MxuqTQMfWGGX/view?usp=sharing

Acknowledgments

I would like to thank my parents for their nonstop support and encouragement. I would also like to thank my mentor Clint Hubbard for his advice and my teacher sponsor Allan Didier.

References

[1] *Airspace Authorizations for Recreational Flyers*. (2023, February 16). Federal Aviation Administration. https://www.faa.gov/uas/recreational_flyers/authorization

[2] Amazon Staff. (2022, June 13). *Amazon Prime Air prepares for drone deliveries*. About Amazon. https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries

[3] ArduPilot. (2023). *ArduPilot*. ArduPilot.org. https://ardupilot.org/

[4] Autodesk. (n.d.). *Unify design, engineering, electronics, and manufacturing with Fusion 360*. https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR&tab=subscription&plc=F360

[5] Dean, G. (2021, December 7). *Drone racing: Everything you need to know*. Space.com. https://www.space.com/drone-racing-explained

[6] Haselton, T. (2022, June 13). *Amazon says it will begin delivering packages by drone in California later this year*. CNBC. https://www.cnbc.com/2022/06/13/amazon-to-begin-prime-air-drone-deliveries-in-california-this-year.html

[7] Iddenden, G. (2022, April 8). *Amazon's drone delivery will cost the company $63 per drop*. Charged Retail. https://www.chargedretail.co.uk/2022/04/08/amazon-prime-air-drone-delivery

[8] *Matek Flight Controller H743-SLIM V3*. (n.d.). Rotorgeeks. https://rotorgeeks.com/matek-flight-controller-h743-slim-v3

[9] *MAVLink Developer Guide*. (n.d.). https://mavlink.io/en/

[10] MaxAmps Lithium Batteries. (n.d.). *Lithium Polymer Charging/Discharging & Safety Information*. https://maxamps.com/pages/lipo-care-php

[11] Raspberry Pi 4 B (2 GB RAM). (n.d.). In *Elektor*. https://www.elektor.com/raspberry-pi-4-b-2-gb-ram

[12] Raspberry Pi Ltd. (n.d.). *Raspberry Pi 4*. Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/