Operation BEARTRAC

AE 100, Team CANDY

Authors

Colin Hoang, Anuj Kakde, Nithika Kiruveedula, Daniela Uyuni, Yamilex Cruz Ramirez

Supervisors

Dr. Hilary Johnson, Dr. Derek Lang, Ting-Hao Wang, Jed Roach

Contents

1 Abstract

1.1 Mission Overview

The purpose of the mission is to design and showcase an aircraft that will be able to autonomously perform various search missions, carry and operate an optical payload, and successfully transmit telemetry data to the CubeSat. The optical payload will be able to search for and find Oski in the form of a 1 m^2 ground target within a 2-acre area at Richmond Field Station. The CubeSat team will relay commands and imagery to the aircraft in order to aid the aircraft's search.

1.2 ConOps/Flight Profile

The entire mission will last about 100 minutes. The power limitations of the flight profile are 10 minutes of data retrieval and 11 minutes of autonomous flight. The 10 minutes of data retrieval will include preflight data from the CubeSat and initial path planning within the search area. The 11 minutes of flight will occur about the bounded area previously determined, wherein the aircraft will locate the various search targets. The rest of the available time, the aircraft will wait at a designated location for the CubeSat to relay new search areas, which will repeat for a number of rounds until Oski is found.

1.3 Critical Assumptions

We assume that we will be flying in flat terrain, with no obstacles blocking the view of Oski. In addition, we will assume we are flying at around 15 miles per hour winds at RFS, with the expectation that winds will not exceed 25 miles per hour. The data transmission will be done in the order of aircraft to CubeSat to ground station. Operations will be limited to daytime, with no rain, clear visibility, and limited GPS interference.

1.4 UAV Wiring Diagram

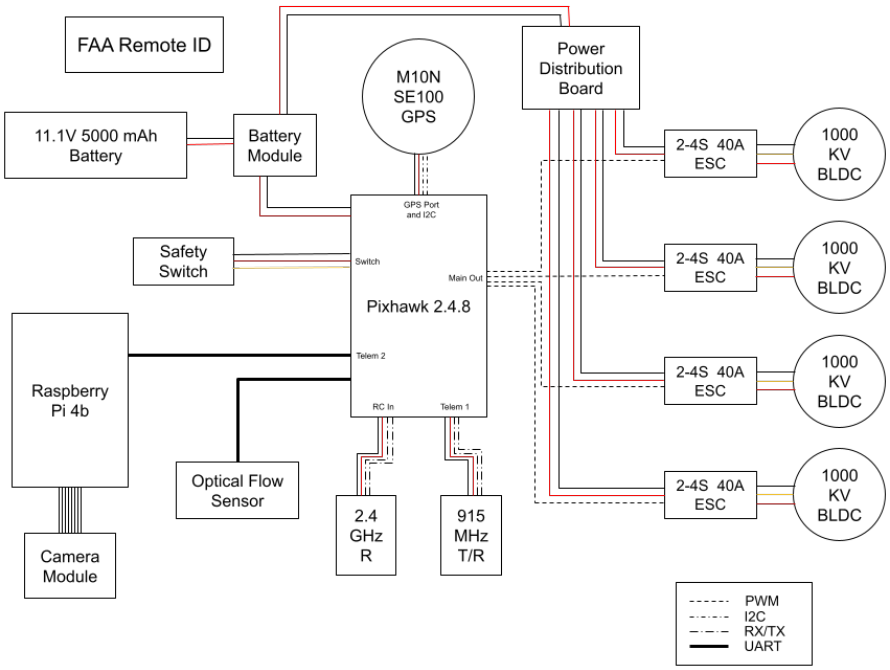

Figure 1: UAV wiring diagram

2 Subsystems

2.1 Airframe

2.1.1 Function

In the overall functionality of a drone, the primary function of its airframe is to serve as its structural foundation by housing, protecting, and connecting the variety of other components while supporting flight loads. The structure and properties of the airframe are integral to ensuring stable flight, as they define the drone's aerodynamic shape. In addition to aerodynamic shape, other key functions of the airframe include structural integrity, component mounting, protection, weight distribution and balance, and configuration.

2.1.2 Components

The main structural components of a quadcopter airframe are the frame base plates, four arms, motor mounts, and the landing gear and legs. The entirety of the frame weighs about 456 grams.

The frame base plates serve as the central body where we are able to mount components like the flight controller, PPM encoder, I2C Bus, 2.4 GHz antenna, 915 MHz antenna, and power distribution board. The frame base plates are made of carbon fiber, which provides lightweight strength, stiffness, and durability for the drone, in addition to vibration-dampening capabilities and aerodynamic efficiency. To best support the heavier load provided by the battery whilst concentrating most of the load towards the center of the drone, in alignment with the frame base plates, we designed a housing bracket to accommodate the battery and Raspberry Pi. The bracket, which was 3D-printed using PLA filament, clips onto the rods that run under the top base plate, supporting the load of the battery by securing it in a slot provided by the bracket.

The four arms extend outward from the frame base plates and hold the motors. The arms are also made of carbon fiber, with their stiffness and low mass being crucial in order to maintain frame stiffness and resist bending and twisting as they transmit thrust forces to the central frame. Additionally, the arms route wiring from the ESCs to motors, with the ESCS being ziptied to the bottom of the arms.

The motor mounts are located at the ends of the arms, ensuring the motors are held securely. The mounts allow for motor loads to be securely transferred to the frame without placing too much stress on the carbon arms, as the motors produce high torque and continuous vibration amidst thrust force. Additionally, the motor mounts maintain correct motor geometry by ensuring perpendicularity between the motor shaft and arm, enabling consistent thrust vectors across all four motors.

Landing gear and legs provide ground clearance, stabilize the drone on uneven terrain, and absorb impact loads to protect critical components during the impact of takeoff and landing. The legs and landing gear are made of carbon fiber as well, with the landing gear being reinforced with foam to optimize shock absorption during landing and prevent bouncing or tipping.

2.1.3 Prototyping and Testing

Our airframe was constructed using the 500-X4 Quadcopter Frame Kit, purchased from Amazon. When choosing a frame kit, we believed this option stood out as it offers a strong balance of weight efficiency, structural integrity, and compatibility with typical flight controllers needed for our mission. Its 3k carbon fiber construction provided a high strength-to-weight ratio, supporting the mass and payload constraints of our mission while minimizing structural vibration and flex. The 500 mm wheelbase and compatibility with Pixhawk flight controllers provide space and interface flexibility for our payload and avionics integration, and the included landing gear enables protection of components during landing. In prototyping and testing the airframe, we aimed to validate that our chosen airframe configuration is strong enough to sustain thrust loads, provides adequate stiffness for stable flights, and integrates appropriately with other subsystems.

One of our most significant initial analyses was verification of the center of gravity, leading us to design a battery bracket housing. A quadcopter's flight stability heavily depends on maintaining a precise center of gravity location with respect to the propeller plane. For multirotor configurations, an optimal center of gravity typically lies slightly below the motor plane within a few millimeters of the roll and pitch axes. Due to components of our other subsystems adding asymmetric mass to the frame, management of the center of gravity became a key point, as corrective torque using additional power draw must be applied when the center of gravity shifts too far in one direction. To address this concern, we conducted various hand calculations

on our drone's estimated center of gravity and then built on these calculations using Ansys Mechanical. We imported our CAD into Ansys Mechanical, assigning appropriate material properties to each component of the frame and other subsystem components.

Figure 2: Isometric view of drone CAD model

Figure 3: Top view of drone CAD model

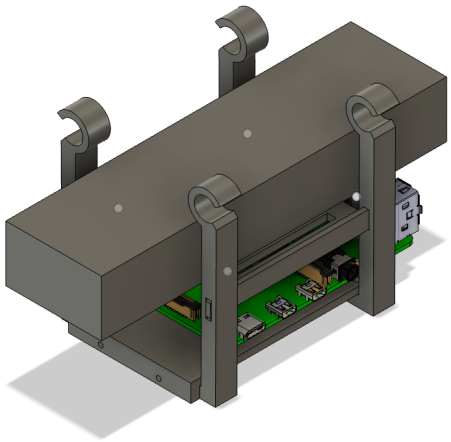

Figure 4: CAD model of battery harness and undercarriage

Once material assignments were completed, the Center of Gravity tool was implemented, and when solved, the results provided coordinates of the center of gravity in the model coordinate system. We found our battery to be the heaviest movable component and the largest contributor to center of gravity variability. To ensure center of gravity stability, we designed a rigid battery bracket that constrains the battery in all degrees of freedom and ensured that in Ansys Mechanical, it maintained the center of gravity at about the center of the frame in the X-Y plane. Additionally, a slotted design ensures the center of gravity remains centered, while also allowing the battery to be removed for charging.

A more thorough structural analysis of the airframe was then conducted utilizing Ansys Mechanical as well, addressing the overall structural performance of our airframe by importing the assembly, assigning

appropriate material properties, and applying thrust, landing, and payload loads. A solid mesh was generated, with tetrahedral elements for plates and brackets and additional local mesh refinement at areas with high stress gradients (motor mounts, arm joints, landing gear attachments). Static structural simulations allowed us to analyze stress, strain, and potential deformation under thrust and battery weight to ensure components do not flex excessively or exceed allowable stress. Modal analysis allowed us to compute the first several natural frequencies of the frame to prevent resonance, to ensure cleaner IMU data, better flight stability, and reduced vibration for imaging payloads. Using these analyses, we were able to identify the motor-mount interfaces, arm-to-center plate joints, landing gear attachment points, and areas surrounding the battery and payload mounts as potential areas of high stress. These regions were shown to carry higher load concentrations due to thrust and maneuvering, with geometric features such as screw holes and cutouts increasing stress concentrations. The landing gear was displayed as the highest hotspot, so we reinforced this area using malleable support to avoid damage prior to subjection to landing impacts.

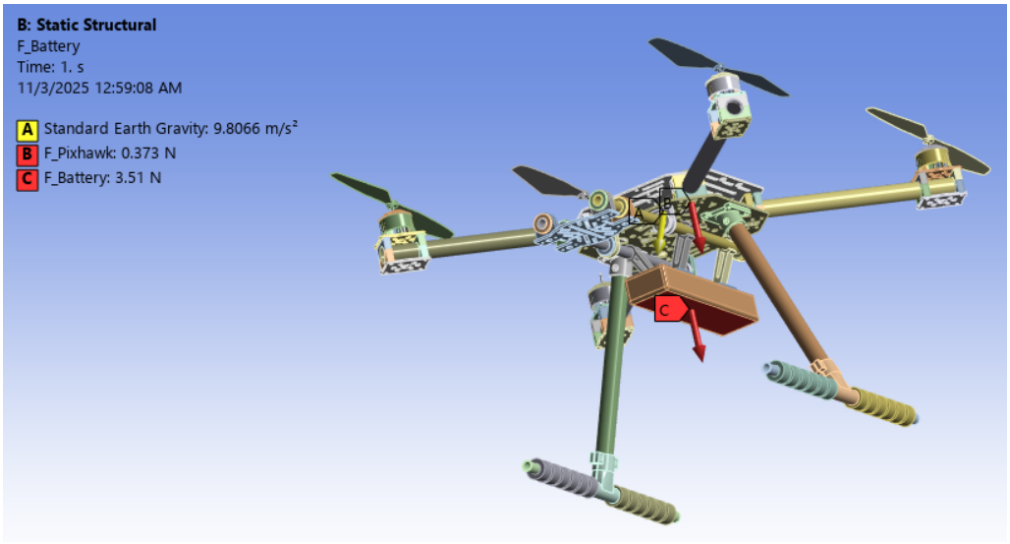

Figure 5: Drone CAD model in Ansys for analysis

2.1.4 Constraints

Our airframe design was primarily constrained by mass limits, center of gravity placement, structural strength requirements, vibration characteristics, and integration with other subsystem components. Structural analysis and material choice addressed structural integrity, center of gravity, and vibration,the while mass limits were analyzed using our mass and power budget as we aimed to maintain an ideal thrust-to-weight ratio. In integration with other components, space within central plates influenced wiring routes while limiting the choice in component dimensions.

2.2 Power

2.2.1 Function

The power subsystem serves as the critical energy source for all of the UAV's operations during the mission. Its primary functions include supplying 11.1 V power to all four brushless motors and ESCs for propulsion during flight operations. It is responsible for providing stable 5V power to the avionics (Pixhawk flight controller, GPS, and sensors), communications equipment (915 MHz and 2.4 GHz antennas, FAA transponder), and payload systems (Raspberry Pi 4B, camera module, and cooling system). For the duration of the mission, the power system also enables 11 minutes of autonomous flight plus 10 minutes of ground-based data transmission per operation cycle. For safety and redundancy, it also supports any failsafe operations, including return to launch (RTL), automatic landing, and emergency power management protocols. The power system also

maintains sufficient energy reserves to complete three full mission cycles while preserving a minimum 20% safety margin as specified in mission requirements.

2.2.2 Components

The main components of the power system include: A primary lithium battery (3S LiPo 5000mAh, 11.1V, 120C), 4 ESCs (40A with BEC, 2-4S compatible), and 1 power distribution board that is integrated on the frame of the drone.

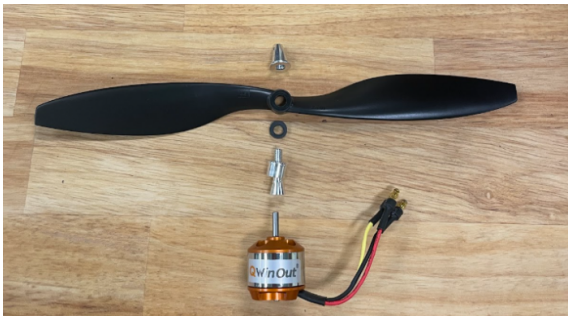

Figure 6: Motor and propeller setup

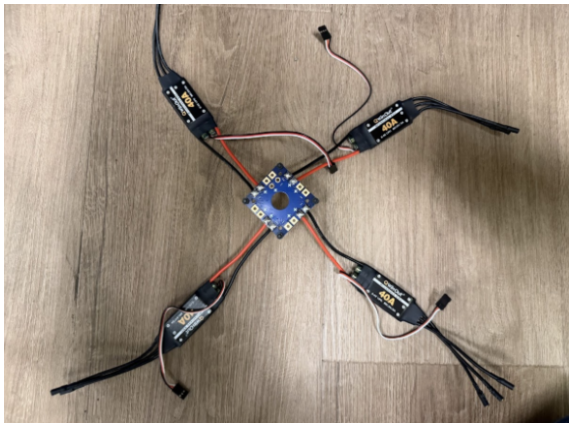

Figure 7: Power distribution board connected to ESCs

2.2.3 Process and Analyses

Our power system follows a hierarchical distribution model beginning with the primary power path, followed by the secondary power path. In the primary Power Path (11.1V), where the battery connects to the main power distribution board integration in the X400 frame, the four separate power leads distribute the 11.1V directly to each ESC, then each ESC will regulate power delivery to the brushless motors based on the PWM control signals from the Pixhawk. In the secondary power path (5V), each ESC's integrated BEC steps down 11.1V to the regulated 5V. The 5V power buses supply the Pixhawk flight controller, the M8N GPS module, 915 MHz antenna, optical flow sensor, Raspberry Pi 4B with the cooling fan, and the Arducam 12MP camera.

The first step in ensuring we would have sufficient Power for the duration of the mission was doing the analysis to support the selection of our motors. For a stable quadcopter flight, the standard thrust-to-weight ratio is 2:1. With a total expected mass of 1.39 kg we knew we needed to be able to produce approximately 2.78 kg of thrust. According to the data sheet of the A2212 1000KV Brushless motors we selected, the recommended voltage was 11.1V (3S LiPo), with 1045 propellers (10 inch diameter with a 4.7 inches of pitch), we expected to see an operating current of 15.76A, 885g of thrust, with max load current of 15.6A but we expected to see an average load current of 7.5A when producing enough thrust to maneuver for the duration of the mission.

We developed 2 comprehensive power budgets to ensure sufficient energy capacity for the mission. The first power budget was the components that would be powered by the Battery, in this case, the 4 motors and the 4 ESCs.

Battery Power Budget Tracker							
Component	**Qty**	**Operating Voltage (V)**	**Current (A)**	**Power Draw (W)**	**Active Time (mins)**	**Energy per Cycle (Wh)**	**Notes**
A2212 Brushless Motors + ESCs	4	11.10	7.5	83.25	9	49.95	9 Minutes of Flight, Hover + Manuevering
			Total Energy per Cycle (Wh)				49.95
			Total Energy per Cycle + 3% Tolerance (Wh)				51.45
Battery provides 11.1V at 5000 mAh so total of				55.5 Wh of capacity for one mission			TRUE
Mission is	3		cycles long so the total energy for the mission is (Wh)				154.35

Figure 8: Battery Power Budget Tracker

From our Battery Power Budget, we established that a single cycle would require 51.35 Wh per cycle, and with our chosen battery, we could reliably have a battery capacity of 55.5 Wh per fully charged battery. Our 3% tolerance factor already accounts for the worst-case operating conditions, and thus under nominal conditions, we expect the system to maintain adequate reserves with the planned battery swap at the end of every cycle to ensure mission completion.

We decided to implement the second power source, the power bank, to ensure the Raspberry Pi would never shut down, even during battery swaps. It was more time efficient for our system to ensure the Raspberry Pi had continuous power to reduce the amount of time needed for setup between breaks and to mitigate the risk of any potential data loss before being able to successfully send it to the CubeSat Team.

The second Power Budget we developed was for the components that would be powered by the Power Bank: Pixhawk 2.4.8 Controller, M8N GPS, 915 MHz Antenna, FS-iA6B 2.4 GHz Receiver, Raspberry Pi 4b + Cooling, and the Arducam 12 MP Camera. We planned for all components, with the exception of the camera, to run for the entire 14-minute period, including flight and grounded transmitting.

From our analyses, we realized we needed to adjust the flight time for the mission and needed to incorporate battery swaps. As a result of this, we had to incorporate battery swaps between cycles. Our 3% tolerance factor already accounts for the worst-case operating conditions, and thus under nominal conditions, we expect the system to maintain adequate reserves with the planned battery swap at the end of every cycle to ensure mission completion.

Power Bank Power Budget Tracker							
Component	**Qty**	**Operating Voltage (V)**	**Current (A)**	**Power Draw (W)**	**Active Time (mins)**	**Energy per Cycle (Wh)**	**Notes**
Pixhawk 2.4.8 Controller	1	5	0.25	1.25	14	0.29	9 minutes of Flight, 5 minutes grounded transmitting
M8N GPS	1	5	0.1	0.5	14	0.12	Always broadcast GPS position throughout 14 min period
915 Mhz Antenna	1	5	0.03	0.15	14	0.04	Always transmitting throughout 14 min period
FS-iA6B 2.4 GHz Receiver	1	5	0.03	0.15	14	0.04	Always transmitting throughout 14 min period
Raspberry Pi 4b + Cooling	1	5.00	1.1	5.5	14	1.28	Operating throughout the mission to generate path plan and object detection
Arducam 12 MP Camera	1	5.00	0.2	1	9	0.15	Image capture for 9 min period of flight
			Total Energy per Cycle (Wh)				1.91
			Total Energy per Cycle + 3% Tolerance (Wh)				1.97
Power Bank provides 5V at 10000 mAh so total of				50 Wh of capacity			TRUE
Mission is	3		cycles long so the total energy for the mission is (Wh)				5.91

Figure 9: Power Bank Budget Tracker

2.2.4 Constraints

One of the biggest constraints was that the power subsystem alone accounted for the most mass from a single system. In our mass budget tracker, we needed to choose a battery that would provide sufficient battery while still maintaining a 2:1 ratio for thrust. In our budget estimates, 40.9% of our mass was from our power components. Despite this, our budgets were able to help us decide on the proper motors, propellers, and battery to meet our needs.

2.3 Sensors and Controls

2.3.1 Function

The sensor subsystem provides all of the real-time environmental, positional, and motion data needed for the drone to navigate through space, maintain stable flight, and carry out the mission. The sensors measure the drone's altitude, attitude, position, velocity, and environmental features, feeding this data into the Pixhawk flight controller, which then computes control outputs

2.3.2 Components

The Pixhawk 2.4.8 Flight Controller contains 3-axis accelerometers, 3-axis gyroscopes, 3-axis magnetometers, and a barometer. These core flight sensors provide attitude (roll, pitch, and yaw), angular rates, vertical position (barometric altitude), and heading (compass). The GPS module provides global position, velocity, time synchronization, and heading.

2.3.3 Prototyping and Testing

Before installing any sensors on the airframe, we connected each one to the Pixhawk and Raspberry Pi for standalone testing. With this initial verification, we were able to gauge accuracy in values such as IMU readings, magnetometer heading stability, and barometer altitude noise level, and identify wiring or orientation errors.

With regards to the calibration of sensors, we performed the necessary calibrations through QGroundControl. For example, accelerometer calibration ensured proper axis orientation, and gyroscope calibration helped to remove initial position bias to ensure clean inputs to the flight controller's EKF. When the system was later fully constructed in conjunction with other subsystems, we further monitored EKF status indicators, such as GPS health, in QGroundControl to make sure all sensors were fused correctly and that the flight controller was receiving reliable and consistent multi-sensor data. In full-system integration, where we checked EKF health consistency using arm/disarm tests and flight tests, we began to encounter faults in the barometric altitude calculated by the Pixhawk. To remedy this issue, we altered parameters AHRS_GPS_USE and EK3_OGN_HGT_MASK in QGroundControl to use GPS instead for height calculations and apply corrections to local positioning. To validate the switch to reliance on GPS altitude, we conducted barometer analysis in two scenarios: a stationary bench test and a stationary test with the motors running. In the stationary bench test with no motors running, we aimed to establish a baseline for the accuracy and noise of the barometer by placing the drone on a flat surface. With the drone not facing any mechanical or aerodynamic disturbances, we logged altitude readings for several minutes with the system completely idle. We then calculated the altitude range, standard deviation, and drift over sixty seconds, with normal performance being displayed under no-load conditions. However, when we ran a stationary test with motors running (no propellers), we began to see rapid oscillations in altitude, correlating to motor vibration transferring through the airframe into the Pixhawk and causing pressure fluctuations. Height estimates became much more unstable as vibration produced false altitude changes in the barometer, which relies on tiny pressure differentials. Upon transitioning the EKF height source from barometer to GPS using parameters AHRS_GPS_USE and EK3_OGN_HGT_MASK, the variance in altitude in stationary tests with motors running decreased dramatically since the barometer could not provide as reliable data under motor vibration.

2.3.4 Constraints

The sensor subsystem is largely constrained by vibration limits, magnetic interference, airflow disturbances, and reliable GPS signal access. Furthermore, sensor placement is hindered by weight distribution, axis alignment consistency, and access to high-current wiring. In order to ensure accurate EKF fusion and consequent accuracy in navigation and stability, the sensor subsystem must receive proper amounts of power, maintain stable mounting, and mitigate environmental factors that corrupt data inputs to the flight controller.

2.4 Imaging

2.4.1 Function

The imaging subsystem utilizes a camera to complete the objective of locating and geo-tagging the QR code image on the field.

2.4.2 Components

Our imaging payload consists of a 12MP Arducam Camera Module 3 with autofocus. The camera was connected to the Raspberry Pi 4B using native connections and built-in communication protocols. On the backend, the QR code recognition was done by implementing the built-in OpenCV QRCodeDetector class.

2.4.3 Prototyping and Testing

QR code detection was first evaluated using a small, handheld QR code to validate the algorithm's ability to detect targets at varying distances. After establishing baseline performance, a full-scale replica of the 1 m^2 QR code was printed and used to determine the maximum reliable detection range. The algorithm consistently detected the QR code up to a distance of approximately 30 feet, beyond which detection became unreliable.

After integrating the camera onto the drone, flight tests were conducted inside the drone cage, during which the system attempted to detect the QR code at varying altitudes while in flight. The subsystem was subsequently tested at the Richmond Field Station, where additional experiments were performed to evaluate QR code detection at greater heights.

Figure 10: Successful detection and outlining of QR code

```
QR Code Detection #45
========================================
Timestamp: 2025-12-10 13:02:59.991
QR Data: https://forms.gle/N6vwcJ2HbMj2JTq56

GPS Coordinates:
  Latitude:  37.91581700°
  Longitude: -122.33528210°
  Altitude:  22.38 m

========================================
```

Figure 11: Successful image geo-tagging and decoding

The QR code detection system required significant stability to reliably lock onto and scan QR codes from drone-captured images. To address initial instability, the computer vision pipeline was modified to capture still images whenever a QR code was believed to be in frame, though this produced many noisy or irrelevant images. A post-processor script, QR_PostProcessor.py, runs at the end of each flight to filter these images. It loads all images from the source directory, validating the files based on edge density, contrast, pattern variance, and pyzbar-based QR detection. Valid images are cropped to QR regions using detected green

triangle boundaries, and the sharpest high-confidence crop is selected. The final output includes a binary QR array of the scanned QR image to be transmitted to the ground station.

Figure 12: Fusion Binary Image of QR code

2.4.4 Constraints

While the QR detection algorithm remained reliable in identifying the physical location of the QR code, it struggled to consistently decode the information embedded within the code. This limitation was primarily due to image instability, as the camera was rigidly mounted to the drone without a gimbal, resulting in significant motion-induced blur and vibration. Additionally, the drone's flight dynamics were characterized by sharp, non-smooth movements, further degrading image quality and negatively impacting QR code decoding performance.

2.5 Data Transmission

2.5.1 Function

The data transmission subsystem is the method the drone communicates with the ground station and flight operator to send and receive commands and data that inform and dictate the drone's flight and operations.

2.5.2 Components

In total, we used four different communication methods to interact with the drone. The flight operator communicated with the drone using a 915 MHz SiK radio antenna to view and calibrate flight parameters, as well as to monitor the flight profile through QGroundControl. A 2.4 GHz RC transmitter and receiver were used to send manual commands to the drone and served as a safety backup in case of an emergency stop.

Channels 5 and 6 on the controller were programmed and bound to switches. Channel 5 was configured as a two-position switch for Arm and Disarm functions, with Disarm serving as the emergency stop that cuts power to all motors. Channel 6 was set as a three-position switch to control the drone's flight modes. The three assigned flight profiles were Stabilize, Altitude Hold, and Guided.

During autonomous flight, the controls were set to Guided mode to execute the flight plan provided by the Raspberry Pi. Since we were unable to fully complete testing of the autonomy code on the drone, we temporarily replaced Guided with Loiter mode, which provided a position lock for safer manual flying.

The ground operator communicated with the Raspberry Pi over Wi-Fi. Through SSH, the operator could access the Raspberry Pi's command-line interface during flight to modify flight profile code in real time and view telemetry data from both the Pixhawk and Raspberry Pi.

The final communication method used a 915 MHz LoRa antenna to link the ground station and the CubeSat. The LoRa module was programmed to receive three coordinate pairs from the CubeSat team

and store them as input data for autonomous flight planning. It also transmitted the coordinates and URL associated with scanned QR codes back to the CubeSat team.

As for intra-drone communication from the Raspberry Pi to Pixhawk, telemetry and commands were sent through the MAVLink protocol using the pymavlink library with Python. The connections were made through the telem2 port on the Pixhawk, utilizing Rx and Tx pins on the Raspberry Pi.

2.5.3 Prototyping and Testing

Communication between the ground station and the drone via LoRa was tested by transmitting example packets containing coordinate pairs between the two devices. The sample code used for this data transmission test can be found on GitHub in the LoRa Communications folder, labeled 915Receiver.py and 915Transmitter.py. To validate the range of the 915 MHz, LoRa, and 2.4 GHz antennas, we performed a link budget analysis to verify the drone's effective communication range.

2.5.4 Constraints

Due to the 915 MHz antennas of all teams broadcasting publicly, this led to commands of another team arming our drone, as well as interfering with our ability to conduct testing while their antennas were broadcasting. Another limitation we didn't foresee was the limited range of the WiFi communication between the Raspberry Pi and the ground operator.

3 Major Technical Trades

3.1 Propeller Type

One of the technical trades we considered was using 1045 or 9450 propellers. The main considerations between the propellers are static thrust, efficiency, integration, and power draw; however, putting more emphasis on efficiency and integration. Using these parameters, we decided to use 1045 propellers, which have a diameter of 10", which creates more static thrust at the same RPM ($T \propto R^2$ roughly) compared to the 9450 propellers; however, larger diameter propellers add more weight to the drone. Ultimately, we decided that the weight gain was minimal enough to justify the increased thrust. In addition, the larger diameter of the 1045 propellers correlates to improved hover efficiency, which is enhanced by better integration with 2212 brushless motors in comparison to the 9450. However, we still must consider that higher diameters necessitate greater power and torque demand at the same RPM.

3.2 Quadcopter vs. Hybrid VTOL vs. Fixed-Wing

Another major technical trade-off for our team was initially choosing whether or not to build a quadcopter, a hybrid VTOL, or a fixed-wing. The main considerations that factored into our decision were stability, imaging quality, build/test time, Pixhawk integration, and cost, with a higher emphasis placed on stability and build/test time. Amongst those three considerations, the quadcopter ranked highest, with the hybrid VTOL and fixed-wing following. The quadcopter has better stability for imaging missions due to its hover capability, which will be necessary for QR code detection. In addition, we considered the quadcopter to have the shortest build/test timeline, which was critical to the 11/17 build completion deadline. In addition, the stationary hover capability of the quadcopter allowed for the best imaging quality. The quadcopter is also the most compatible with the Pixhawk since it has well-documented/tested autopilot capabilities for multi-rotor aircraft. In addition, given our timeline, the quadcopter has a lower technical risk, considering it has simpler aerodynamics than a hybrid VTOL or fixed-wing.

4 Challenges and Limitations

4.1 Risk Matrix

No.	Risk Summary	L	C	Strat.	Risk Statement
1	Comms/Packet Loss	2	5	M	During data transmittance to cubesat, packet may be lost.
2	Component Failures	2	4	M	In flight, components such as motors or ESC's may be faulty and fail.
3	Landing Failure	2	4	M	During landing, landing gear may fail to bear the load of UAV landing or UAV will not achieve a "smooth" landing.
4	High Winds at RFS	1	4	A	On test day, winds may exceed 25 mph, which the UAV is not designed to withstand.
5	Loss of Power	3	5	M	In flight, if our battery experiences drop in voltage too quickly or if failsafe mode is triggered it may cause unstable flight or crash.
6	GPS noise/drift	1	3	W	May cause inaccurate position or velocity estimates during operation and lead to deviations on flight path.
7	Development Delay	3	4	M	Reduces test and build time and ability to establish realistic timelines pertinent to project

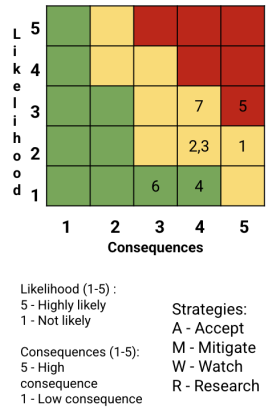

Figure 13: Risk Matrix

The key risks we identified in our risk matrix are Communications/Packet loss, component failures, landing failure, high winds at RFS, loss of power, GPS noise/drift, and development delay. The risk involving communications/packet loss is that during data transmittance from UAV to CubeSat, the data packet may be lost, and a strategy we identified for this issue is incremental testing. The risk posed by component failures includes failure of motors or ESC during flight, which can be mitigated through incremental testing and monitored during flight. The risk involving landing failure is that during landing, the landing mechanism may fail to bear the load of the UAV, or the UAV will not achieve a "smooth" landing, both of which have the potential to lead to structural/component failure. This can be mitigated by reinforcing the landing gear during flight by the pilot. Another risk identified is high-speed winds at the RFS on demo day, with the UAV not being designed to withstand more than 25 mph winds. Our most likely/consequential risks were identified as loss of power, GPS noise/drift, and communications loss.

4.2 Mitigation Strategies for Future Directions

Throughout the course of the semester and leading up to the final demonstration, we identified several mitigation strategies that would have accelerated project development. Strategies to prevent component failures included protecting hardware from heat exposure, avoiding GPS-dependent flight modes indoors, and carefully verifying propeller orientation before testing. During initial drone cage testing, we failed to check propeller mounting orientation, resulting in inverted propellers that caused flight failure and propeller damage. A second critical error involved relying on GPS-based altitude hold mode within the drone cage, where GPS signal degradation caused the drone to ascend uncontrollably into the roof, further damaging propellers. Most significantly, on demonstration day, extended sun exposure caused our Raspberry Pi to overheat, corrupting its drive and eliminating our ability to demonstrate QR code scanning capabilities. This thermal failure, previously unencountered in our development cycle, highlighted the need for thermal management considerations in field deployments.

Additional timeline-related mitigation strategies included scheduling earlier flight testing at Richmond Field Station (RFS), implementing more incremental subsystem testing, and coordinating earlier with the CubeSat team on communication protocols. Our RFS testing was limited to two days—one focused on pilot flight evaluation in wind conditions and the other on validating QR code detection at operational altitudes. Extended RFS access would have enabled proper autonomous flight validation and complete system integration testing. Similarly, our development process would have benefited from more rigorous incremental component testing before full system integration, as we encountered multiple setbacks attempting to integrate subsystems that had not been individually validated. Finally, we deferred CubeSat communication integration

until the night before the demonstration, when the CubeSat team completed their code. Earlier coordination with the CubeSat team would have allowed sufficient time to debug the drone-CubeSat interface and validate end-to-end telemetry.

4.3 Timeline Comparison

Referencing our timeline, we were able to build and assemble the drone by 11/10. We were also able to begin testing and coding our Raspberry Pi and camera around 11/6, and flight testing in the Hesse drone cage also began around 11/13. However, we did not test telemetry transmission and receiving until the week of the demonstration due to the cubesat team being unable to test with us until the night before the demonstration.

5 Reflections

5.1 Colin Hoang

- Technical Decisions: A big technical decision was with the on-board computer paired with our Pixhawk in order to do in-flight computation and telemetry/command. Our two options were: using the simple Raspberry Pi Operating System or Ubuntu to run commands through Robot Operating System (ROS). About halfway through our project, we decided to pivot from setting up our Raspberry Pi with the Raspberry Pi OS and instead attempted to set it up with Ubuntu and ROS. The benefits we had in mind were that the new system was more robust for tasks like path-planning and pairing with camera detection. However, we ultimately decided to abandon that direction since we were limited on time and the physical hardware capabilities of the Raspberry Pi to run heavier computation. Looking back now, we definitely made the right choice. Our end product on the Raspberry Pi OS was able to send path-planning commands to the drone as well as reliably run a QR detection algorithm. There is definitely more functionality and robustness associated with the Ubuntu/ROS setup, but in the interest of time and scope of this project, our choice worked out.

- Testing Reality vs. Theory: As I was flying our drone and Team 3's drone, I quickly realized that the altitude fluctuations in the GPS caused the drone to act very erratically and automatically accelerate upwards on Altitude-Hold mode. In theory, the drone should have held its altitude somewhat reliably. However, we did not consider the sensor noise and unreliability of the GPS module. We should have originally tracked the altitude as we were flying in Stability mode and seen the behavior of the altitude readings. We eventually fixed this later with sensor fusion.

- Team Management: Our team organized work sessions mainly in the Aero Lounge and had meetings over Zoom if there was no physical work required. Everyone was very good about attending and participating. I think one thing that worked very well was repeating weekly meetings that everyone allocated time for. This forced us to work on the project bit by bit and make progress over the weeks. I think one thing that definitely caused us stress and shifted us off course was the unexpectedly long work hours to figure out the Pixhawk flight controls and get the drone flying in the first place. I think one thing I would definitely keep for future group work is the weekly time-allocated meetings.

- I think that I grew in the sense that I am now able to learn to adapt to new things and fill in positions that I once felt like I was never able to before. Going into this project, I knew nothing about drones or onboard computing within drones. The whole process involved first researching current work on drones and using my resources, such as Ting-Hao, to learn more about it, and then developing methods to implement my knowledge. I feel much more comfortable handling novel tasks and projects, as well as working with others to make progress.

5.2 Anuj Kakde

- **Technical Decision:** One major technical decision I contributed to was selecting the motors and electronic speed controllers (ESCs) for the drone's power system. We evaluated options including 3S versus 4S LiPo batteries, 30A versus 40A ESCs, and 1000 KV versus 1400 KV brushless motors. The

final choice of a 3S battery, 40A ESCs, and 1000 KV motors was driven by component availability, power efficiency needs, and flight duration goals: the 3S battery discharges more slowly than a 4S (reducing heat and extending runtime), while its voltage proved sufficient to lift the drone's weight without requiring excess current capacity. The 40A ESCs provided a safety margin over the 30A option for reliability under variable loads, matching the conservative 3S voltage rating. Lower-KV 1000 motors also drew fewer amps than 1400 KV equivalents at equivalent thrust, minimizing battery drain and thermal stress. These selections balanced performance, safety, and practicality effectively, and I would make the same choices if restarting the project.

- **Testing Reality vs. Theory:** One clear case where testing diverged from our theoretical predictions was the drone's flight duration versus our initial power budget analysis. Our calculations projected only about 9 minutes of flight time based on conservative estimates of motor and ESC power draw. In reality, the drone achieved 13-15 minutes, significantly exceeding expectations.

  This mismatch revealed that our power models overestimated ESC current draw, likely due to conservative assumptions about efficiency losses and hover throttle conditions. While the outcome was positive, as it provided a substantial safety margin, it highlighted how pessimistic modeling ensured mission reliability but masked true performance potential.

  To catch this earlier next time, we would measure actual ESC power consumption under hover conditions during ground testing: connect the battery to a multimeter with motors spinning at 40-50% throttle, then refine the power budget model with real data before flight integration.

- **Team Management:** Our team frequently communicated tasks and responsibilities, dividing subsystems among group members so everyone had meaningful work to do. We typically met during class or lab sessions in the Aero lounge, providing ample time for incremental progress. Most of the time, when someone was working on a hardware aspect of the project, other team members provided support, troubleshooting assistance, and help building subsystems.

  The constant communication and collaborative nature, where everyone understood and contributed to all aspects of the project, even beyond their designated tasks, is something I would repeat in future school projects, as it helped everyone learn and deepen their understanding of class-related topics. One concrete process change I'd implement next time is scheduling flight testing at the RFS much earlier, since real incremental progress and validation could really only occur there, as the drone cage was too small. Testing these incremental changes in actual flight conditions would be something I'd implement in future projects.

- **Personal Growth:** One way I personally grew as an engineer this term was by applying theoretical concepts to practical design and validation decisions. This came out during drone assembly, where I used flight controls and stability principles to position the center of mass directly above the flight controller, ensuring accurate IMU readings without skew. I also incorporated structural failure points into the drone legs, protecting critical components during crashes. This experience taught me to not just apply engineering principles but to design with intentional failure modes in mind from the start. Moving forward, I'll make this philosophy a standard checklist item in every design review for future projects.

5.3 Nithika Kiruveedula

- Technical Decision: One major technical decision I partook in was how the altitude for flight control was estimated, and overall sensor fusion. Initially, we relied on Pixhawk's onboard barometer as the primary source for height, which is a common and typically sound choice for smaller UAVS. The alternate option we considered was to switch the EKF height source to GPS by adjusting parameters. What ultimately drove the decision was experimental evidence and consultation from Ting-Hao. During stationary tests with motors running, we observed large oscillations due to vibration-induced pressure fluctuations, degrading altitude estimates significantly. We saw much more stable results in these conditions upon switching to GPS-based height estimation. Upon reflection, this may not have been the "safe" choice, as it is typical practice to utilize the Pixhawk's onboard barometer. However, given the pressure conditions

we were observing, it was the correct engineering decision given the time and material constraints in our control.

- Testing Reality vs. Theory: One instance where testing diverged from our structural predictions was when our landing gear brackets developed cracks after a hard landing. Based on our structural analyses in Ansys Mechanical, the landing gear and its attachment points were predicted to withstand landing loads within allowable limits. Our model primarily focused on static and quasi-static loading under vehicle weight, assuming relatively smooth landings with minimal vertical acceleration. In actual practice, testing revealed much higher and impulsive landing loads than projected. During less-controlled landings, as we were still in the process of configuring vehicle controls, high-impact forces were transmitted through the legs of the vehicle into the landing gear brackets. This impact caused localized cracking at points near fasteners with high stress concentrations due to clamping force. This experience revealed the incompleteness in our structural simulation, and in future modeling, I would simulate failure modes more accurately by modeling landing as an impact event rather than a quasi-static case.

- Team Management: Our team organized work primarily by subsystem ownership to tackle the variety of complex components involved, which worked really well early on in the project timeline. Each member was responsible for the design and analysis of a subsystem, which allowed us to make parallel progress, especially in early phases where we were planning out subsystem details. However, this structure revealed a slight weakness in integration. With subsystems being designed and tested in an individual capacity, interface issues did not arise until later on, such as mechanical load transfer and sensor mounting effects. These issues were not discovered until late in the timeline, leading to additional roadblocks in integration closer to the testing and demonstration deadline, as fixes in one subsystem often introduced new issues. A concrete process improvement I would implement in a future workflow is scheduling incremental interface checkpoints earlier on, where partial subsystems are intentionally integrated early on to validate compatibility in mechanical, electrical, and data scopes. If we had practiced this earlier, we could have reduced the degree of late-stage surprises through proactive system-level validation. On the contrary, I think one positive aspect of our team's workflow was communication. Our documentation practices were organized and thorough, and we consistently communicated to keep up with official and internally set deadlines.

- Personal Growth: One area where I experienced significant personal growth through this project was in my overall understanding of flight controls. Although I had taken a few controls courses prior to this course, I viewed flight controls largely as an intimidating "black box" that somehow took inputs to stabilize the vehicle. As we progressed through system design, testing, and integration, I was able to engage more deeply with how control loops and feedback actually functioned and played out in real life. Adjusting parameters and diagnosing issues with behaviors such as flight mode behavior helped me connect theoretical control concepts to real system behaviors. While this experience allowed me to grow a lot in becoming more comfortable in control behavior and reasoning, I also realized how dependent system performance is on assumptions made in initial system design about sensor quality and structural dynamics. I hope to continue growing in this area of knowledge by studying control systems more formally and pairing that knowledge with future projects involving testing to validate control assumptions I make.

5.4 Daniela Uyuni

- Technical Decision - One major technical decision was choosing between 1045 and 9450 propellers, which is detailed in the major technical trades section of the report. One of the main tradeoffs was having propellers with a larger diameter in order to increase thrust, however, at the detriment of adding more weight to the drone. Thrust is roughly equal to the square of the radius of the propeller, which is desirable to overcome the rest of the weight on our drone. We decided that although an increased radius size would lead to more weight, the thrust generated by the increased radius was worth the almost negligible weight addition. Reflecting on this, I think I would make the same decision again, as the weight did not affect the drone, and the thrust was ideal.

- Testing reality vs. Theory- A moment in testing that did not match analysis expectations was our landing stand breaking after a few rounds of flight testing. While analysis showed that the drone chassis would be able to withstand. After multiple rounds of flight testing, the brackets on either side of the stand underwent a substantial amount of stress when landing, leading to cracks. We attempted to first mitigate this by gluing it back together with super glue; however, when we would attach the bracket back on and put the screws back in, the stress of putting the screws back in would cause the glue to become undone. We decided instead to use tape to hold the brackets together, which worked well enough to be able to keep them together for the demonstration. What we would do differently next time is perhaps use a different material for the brackets or attempt to screw things slightly less tightly so as not to cause more stress on the brackets.

- Team Management - I think our team did a good job of organizing our work, meeting up regularly to work, and organizing and documenting all work. I think what worked well for us was making a Google Drive for all project-related documentation from the start, which made co-working very easy. I think it also helped that we all had very honest and frequent communication with each other, which made our goals and concerns very clear across the team.

- Personal Growth- I think I personally grew as an engineer a lot technically, especially when it came to building and controlling a drone. In terms of building a drone, I learned about a lot of new things that I would've never previously considered, such as propeller choices, their orientation, and their power consumption. In addition, I was not familiar with how a drone controller was "made" and what kind of modes the controller had that eased flight control.

5.5 Yamilex Cruz Ramirez

- Technical Decisions: One of the major technical decisions I partook in was the decision to incorporate battery switches as part of the mission, as opposed to having continuous flight for the 3 QR code searches. Soon after making an early iteration of the power budgets, we realized that to meet the project requirements of 15 minutes of continuous flight time without our chosen frame and motors, we would need a battery with a much higher capacity, which would, in turn, add an increasing amount of weight to our drone. While scaling up our battery could have been a viable option, it would also mean we would be pushed beyond the trust capacity of our current motors, and we would have to choose larger, and in turn heavier, heavier motors to stay within our 2:1 thrust ratio. It became clear that our design would be significantly heavier and expensive. We chose to instead segment the mission into three separate flights with battery swaps in between each search zone, using a lighter battery that would provide us with sufficient time to search for one QR code. Looking back at this, I think it was the right choice because it allowed us to maintain a drone design that was feasible for our budget, with a small compromise of incorporating a 30-second battery swap between each search.

- Testing reality vs. theory: One of the moments where testing did not match our expectations from our assumptions was when we first started to test the QR code detection vs testing the QR detection during flight. In the Aero lounge, we thought our testing approach was sufficient to yield great results in flight. We were holding the camera steady at various distances and had great rates of detection. However, when we sought to test this in our test flights, especially those at Richmond Field Station, our images of attempted detections showed constant motion blur and jitter that we hadn't encountered in our static tests. We then realized that our ground testing did not consider the conditions of flight, from vibrations from the motors, small oscillations from the flight controller correction position, or wind gusts. Our assumptions treated the camera as a stable platform, so even if we had validated the QR-code detection algorithm, it was difficult to see the same performance in the integrated system. Looking back, I would add something as simple as moving the mounted camera a lot more in an attempt to simulate flight conditions.

- Our team was very good at establishing communication norms, both in how we were going to communicate with each other via text and in how we wanted to keep all of our materials organized. This early organization made it easier to let each other know when we were meeting, and everyone was willing to always have consistent meetings, which allowed us to stay up to date with any new deadlines, both

self-imposed and major class deadlines. We established shared drives, and it made all of our materials accessible as we added more components. Having lab meetings in Hesse also helped us consistently work on the project. We had a good distribution of work when it came to any submission we had to do, and even when one person was working on what was their subsystem, everyone would come in to provide moral support and help if possible. This structure was something that I enjoyed because I could learn a lot from my fellow teammates, and it helped us become a closer-knit group. One process I'd consider implementing sooner that would have helped our testing process would have been organizing trips to the Richmond Field Station a lot sooner; it would alleviate the need to have very long days of testing and having to fix issues on demand without access to resources in Hesse.

- I definitely grew and learned a lot in this class, especially with such a daunting final project. I feel like this class was one where we needed to use online resources and reach out to our GSI, Ting-Hao, to be able to make progress faster. One of my biggest takeaways was to build a system as if we expect the worst conditions. We were able to build a reliable system because we were thorough early on in our development. This was a good project to apply concepts that had previously only been theoretical, and it definitely built my confidence in what can be accomplished in a narrow timeline with a dedicated group.

6 Full List of Components

- 500 X-4 500mm Carbon Fiber Center Plate Quadcopter Frame Kit
- 2.4 GHz RC Controller and Transmitter
- QWinOut 4Pcs Motor+ESC+Prop
- Pixhawk 2.4.8 Kit
- Raspberry Pi 4B
- Arducam Raspberry Pi Camera 3
- FAA Transponder
- Optical Flow Sensor
- M2, M3, M4 Screw/Nut Set
- 3S LiPo Battery 5000mAh 11.1V 120C
- 10,000mAh Power Bank
- Pixhawk 2.4.8 GPS
- Radio Telemetry Kit 915 Mhz
- QWinOut 4Pcs Motors (Extras)
- 1045 Propellors (Extras)
- XT60 Power Switch

7 Github Repository (Code), Wiring Diagram & Subsystem Documentation

`https://github.com/anujndk/AE100-TeamCANDY.git`
`https://drive.google.com/drive/folders/1JqnIvmsYmc3AtT_d-3wu9Ui33Hc9U9au?usp=drive_link`