

Memo

The following changes were made to the Final Design Package:

- Corrected all misspellings and grammatical errors
- Updated Final Design Package to include GitHub URL
- Modified ART-001 to clarify commands
- Added TP-004 (Preflight Checklist) for better future reference

XPRIZE Wildfire: Team Snowflake

Final Project Executive Summary March 28, 2025

Lucas Bons, Anthony Cardenas, Vincent Carter, Nina Chao, Jadyn Christensen,
Joshua Crookston, Isaac Davies, Blake Folsom, Anna Holm, Janie Linford, Jonah Lowther,
Tristan Mott, Jacob Wilkins, Israel Zenteno

1. Project Overview

The XPRIZE Wildfire competition is a 4-year, \$11 million initiative aimed at promoting innovation in firefighting technologies to mitigate destructive wildfires. The competition seeks to revolutionize current wildfire management practices by developing advanced technologies that can swiftly and accurately detect, characterize, and suppress wildfires before they escalate. As participants, BYU, in collaboration with students from the Norwegian University of Science and Technology (NTNU), is designing, building, and testing two unmanned aircraft systems (UAS): a high-altitude scanner and a low-altitude responder.

Our contribution to the XPRIZE Wildfire competition involves equipping two specialized UAS for wildfire management. For the high-altitude drone, an RC Anaconda, we are integrating a fire recognition algorithm with thermal cameras to detect fires from significant heights. For the low-altitude drone, a Hero VTOL fixed-wing quad-rotor, we are implementing the same thermal camera-based fire recognition algorithm, complemented by a deployment mechanism to deliver a suppressant within a precise radius of the detected fire. This project targets controlled, campfire-sized fires less than 3 feet in diameter.

Our project serves as a critical foundation for advanced wildfire detection and prevention, allowing future teams to build upon this innovation and drive ongoing enhancements in response strategies and technology. By aligning our efforts with the broader goals of the competition, we have developed essential tools and capabilities that will contribute to effective wildfire management and suppression.

2. Brief Description of Design

The final design of our drone system consists of two pre-designed drone kits: the RC Anaconda and the Hero PNP. These platforms serve as a reliable foundation for our project, which focuses on integrating non-stock electronics, enabling autonomous flight capabilities, and incorporating fire detection and suppression features. The overall system setup, including connection and wiring diagrams for the electronic components of each drone, is documented in ART-011 of the final design package.

For communication and control, detailed instructions on integrating the Raspberry Pi with the flight controller and ROS 2 Jazzy are provided in GD-001, ensuring seamless communication between hardware and software. To enable real-time fire detection, thermal cameras were mounted on each drone, with CAD files and mounting details available in ART-012 and ART-013. The code flow governing the fire detection algorithm is documented in ART-006, explaining the sequence of operations that allow the system to detect and respond to potential fire hazards effectively.

Additionally, a custom drop mechanism was implemented on the low-altitude drone to allow for the system's fire suppression capabilities. This mechanism, which uses a pin servo mount and

pin attached to the underside of the drone, is detailed in ART-014. The design allows for precise deployment of a fire suppression agent when a fire is detected.

3. Summary of Final Performance

Our final design successfully met the key success measures established for the project, demonstrating both functional and operational effectiveness in fire detection and suppression (See REQ-003 for Key Success Measures). Through extensive testing and validation, we confirmed that each subsystem performed as intended, ensuring the system's overall reliability. The high-altitude drone, equipped with a thermal camera and fire detection algorithm, was able to identify and track fire hotspots while autonomously executing a lawnmower flight pattern. GPS coordinates of detected fires were successfully transmitted, validating the drone's ability to relay critical information for wildfire response. The effectiveness of this detection system was verified through controlled flight tests, detailed in the TP-002 artifact.

The low-altitude drone also met its key objectives, demonstrating the ability to autonomously navigate to a fire's GPS location and execute a suppression drop. During the flight tests, we confirmed that the drone's drop mechanism functioned as expected, deploying an object within close proximity of the target coordinates. While initial tests indicated minor variability in drop accuracy due to environmental factors such as wind and altitude, refinements in flight path optimization and deployment timing improved precision in later trials.

In addition to meeting the core success measures, our system underwent 'Flight Path' tests to assess navigation accuracy and operational efficiency. The high-altitude drone consistently executed its predefined lawnmower scanning pattern, ensuring thorough coverage of the target area for fire detection. Meanwhile, the low-altitude drone successfully navigated to designated GPS coordinates, demonstrating its ability to respond promptly to detected fires. Throughout testing, the fire recognition algorithm was refined, minimizing false positives and ensuring reliable thermal detection. The system's overall performance highlights its potential as a valuable tool in autonomous wildfire response, laying the groundwork for future improvements in detection accuracy, suppression effectiveness, and real-world deployment.

4. Conclusion and Recommendations

Our project successfully met its key success measures through comprehensive, individual component verification. The high-altitude drone demonstrated its ability to detect fire hotspots while flying autonomously in a systematic lawnmower pattern, effectively transmitting GPS coordinates of identified fires. The low-altitude drone showcased robust autonomous flight capabilities and successfully deployed a payload at a designated GPS location. These achievements validate the core functionalities of our multi-drone fire detection and response system.

For future development, we recommend conducting integrated system testing where all components operate simultaneously in a continuous trial, providing a more realistic assessment

of system performance under operational conditions. Additionally, starting flight testing early is essential to identify potential issues and address them before they become critical, as non-fatal crashes can be repaired and used as learning opportunities. We anticipate several areas for enhancement, including extended drone flight time, improved deployment accuracy, and enhanced communication reliability. A key goal is the implementation of full autonomy, involving the development of autonomous landing and takeoff capabilities to transform the current semi-autonomous system into a fully autonomous solution. These advancements would significantly enhance the system's operational effectiveness and real-world applicability for fire detection and response scenarios.

5. Content Summary of Final Design Package


The contents of our Final Design Package include the following:

XPRIZE Wildfire Team GitHub URL

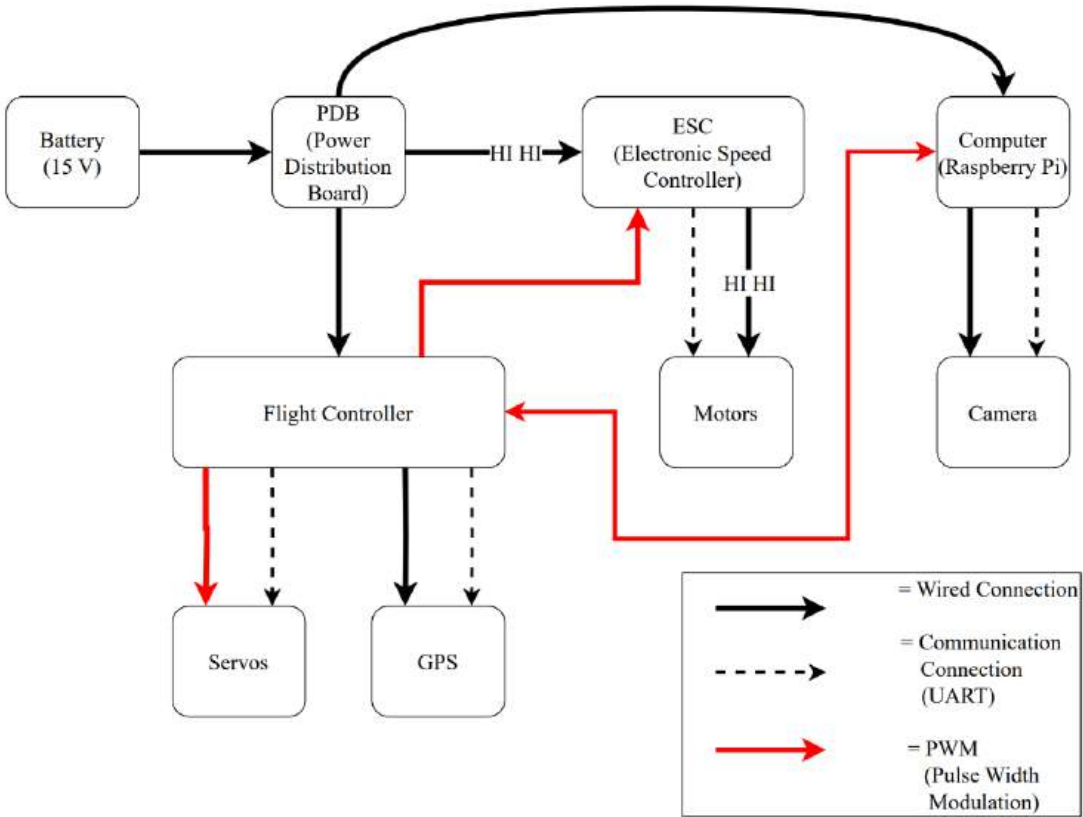
https://github.com/XPRIZE-Snowflake/main_drone_software


- **Assembly Instructions/Software User Guides**
 - **Rpi5 Integration with PixracerPro and Ros 2 “Jazzy” (GD-001)** – Instructions for initially integrating the Raspberry Pi 5 with PixracerPro and Ros 2 Jazzy.
 - **QGroundControl Quick Start Guide (GD-002)** – Instructions for using QGroundControl with a PixracerPro for both the high- and low-altitude drones.
 - **System Layout (ART-002)** – Illustrates the arrangement and physical positioning of all electric components within the high-altitude drone.
 - **Demo Flight Plan (ART-007)** – Demonstration of an example flight plan in QGroundControl.
- **Bill of Materials (REQ-002)** – An itemized list of the materials and components required for the system and their pricing
- **Software Bill of Materials (REQ-004)** – A detailed inventory of all software components, libraries, and dependencies.
- **System Block Diagrams**
 - **Drone Communications Strategy Block Diagram (ART-003)** – Illustrates the general drone communications setup.
 - **System Progress Overview (ART-008)** – Outlines the high-level overview of the system operation, including functionality working and functionality in progress.
 - **XPRIZE System Overview (Professional) (ART-011)** – Maps out the high-level overview of the full system operation, including the full circuit and communication integration.
- **Circuit Diagrams**
 - **High-Altitude Drone Wiring Diagram (ART-001)** – Illustrates wiring connections between the components of the high-altitude drone.
 - **Low-Altitude Drone Wiring Diagram (ART-009)** – Illustrates wiring connections between the components of the low-altitude drone.

- **Coding Documentation**
 - **High and Low Altitude UAV Code Outline (ART-006)** – Explains the code flow for fire detection, camera integration, and suppressant deployment in the High-Altitude and Low-Altitude Drones
 - See also **Rpi5 Integration with PixracerPro and Ros 2 “Jazzy” (GD-001)** in **Software User Guides**
- **Test Procedures**
 - The following procedures, unless otherwise specified, are included in the **Central Test Plan Document (TP-002)**:
 - **Scanner Flight Tests** – Verifies the functionality and accuracy of the fire detection system during flight
 - **Suppressant Deployment Flight Tests** – Ensures the fire suppressant system activates and deploys correctly
 - **Flight Path Tests** – Confirms that the aircrafts follow the intended flight paths accurately (lawn-mower path for the high-altitude drone, direct path to the fire for the low-altitude drone)
 - **Code Debugging Tests** – Identifies and resolves software errors to ensure system reliability
 - **Other Individual Component Test Procedures** – Details on any other testing methodologies we used to verify the performance of additional system components
 - **Thermal vs. Low-Light Camera Comparison for UAS Fire Detection (TP-001)** – Procedure compares the performance of a thermal camera and a low-light camera for fire detection in various lighting conditions.
 - **Test Procedure, Thermal Camera Validation on Drone with Raspberry Pi 5 Running ROS 2 (TP-003)** – Procedure to validate the functionality of a thermal camera connected to a Raspberry Pi 5 running ROS 2, ensuring compatibility, power adequacy, and proper video feed visualization.
 - **Long-Term Safety Plan (ART-015)** – The year-long safety plan that we used for all of our test flights to mitigate risk.
 - **Preflight Checklist (TP-004)** – Checklist for all materials and tests needed before an active flight
- **Custom Part Drawings and/or Files**
 - **High Altitude Camera Mount Render (ART-012)** – Render and CAD file for custom camera mount of the high-altitude drone.
 - **Low Altitude Camera Mount Render (ART-013)** – Exploded assembly render and CAD file for custom camera mount of the low-altitude drone.
 - **Drop Mechanism Render (ART-014)** – Assembly render and CAD file for the pin servo mount and pin used to deploy suppressant from the low-altitude drone.
 - **Raspberry Pi Mount (ART-016)** – Render and CAD file of the Raspberry Pi mount inside the low-altitude drone.

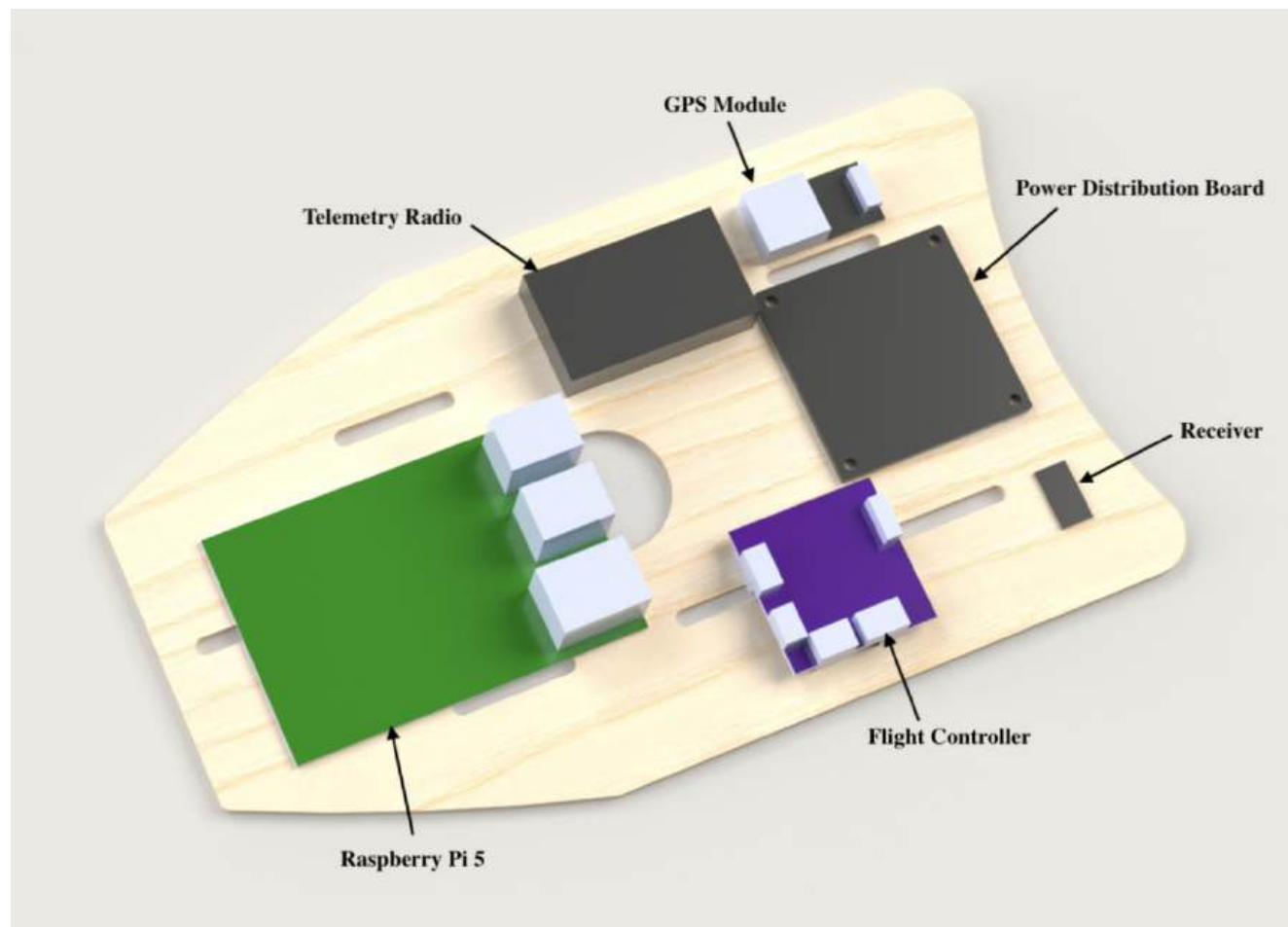
Artifact ID: ART-001	Artifact Title: High Altitude Drone Wiring Diagram	
Revision: 02	Revision Date: 2024-11-12	
Prepared by: Jadyn Christensen	Checked by: Jacob Wilkins	
Purpose: Illustrate how power will be distributed to all peripherals in the High-altitude drone		

Revision History			
Revision	Revised by	Checked by	Date
01	Jadyn Christensen	Jacob Wilkins	2024-10-28
02	Jacob Wilkins	Joshua Crookston	2024-11-12



Artifact ID: ART-002	Artifact Title: Navigation System Layout		
Revision: 05	Revision Date: 2025-03-10		
Prepared by: Nina Chao		Checked by: Jonah Lowther	
Purpose: This render displays the original navigation system layout in the high-altitude Anaconda RC plane, including all electronic components. The photo below is our actual setup. Components are mounted on a fiberboard using Velcro, which slides into the back of the plane and is secured with Velcro. The batteries are attached with Velcro to a balsa wood board at the front of the plane.			

Revision History			
Revision	Revised by	Checked by	Date
01	Nina Chao	Jonah Lowther	2024-10-16
02	Nina Chao	Jacob Wilkins	2024-10-21
03	Jacob Wilkins	Joshua Crookston	2024-10-28
04	Joshua Crookston	Jonah Lowther	2025-02-25
05	Nina Chao	Jonah Lowther	2025-03-10

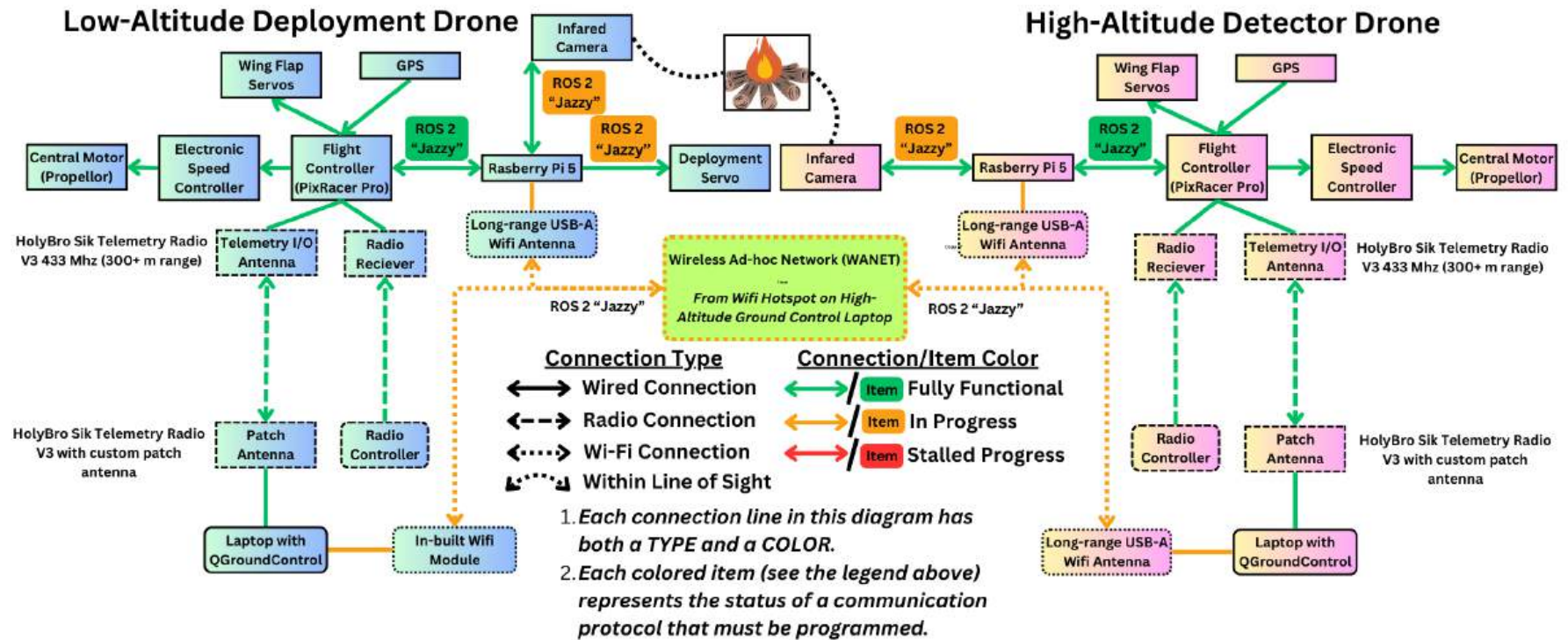



Artifact ID: ART-003	Artifact Title: Drone Communications Strategy
Revision: 08	Revision Date: Feb 25, 2025



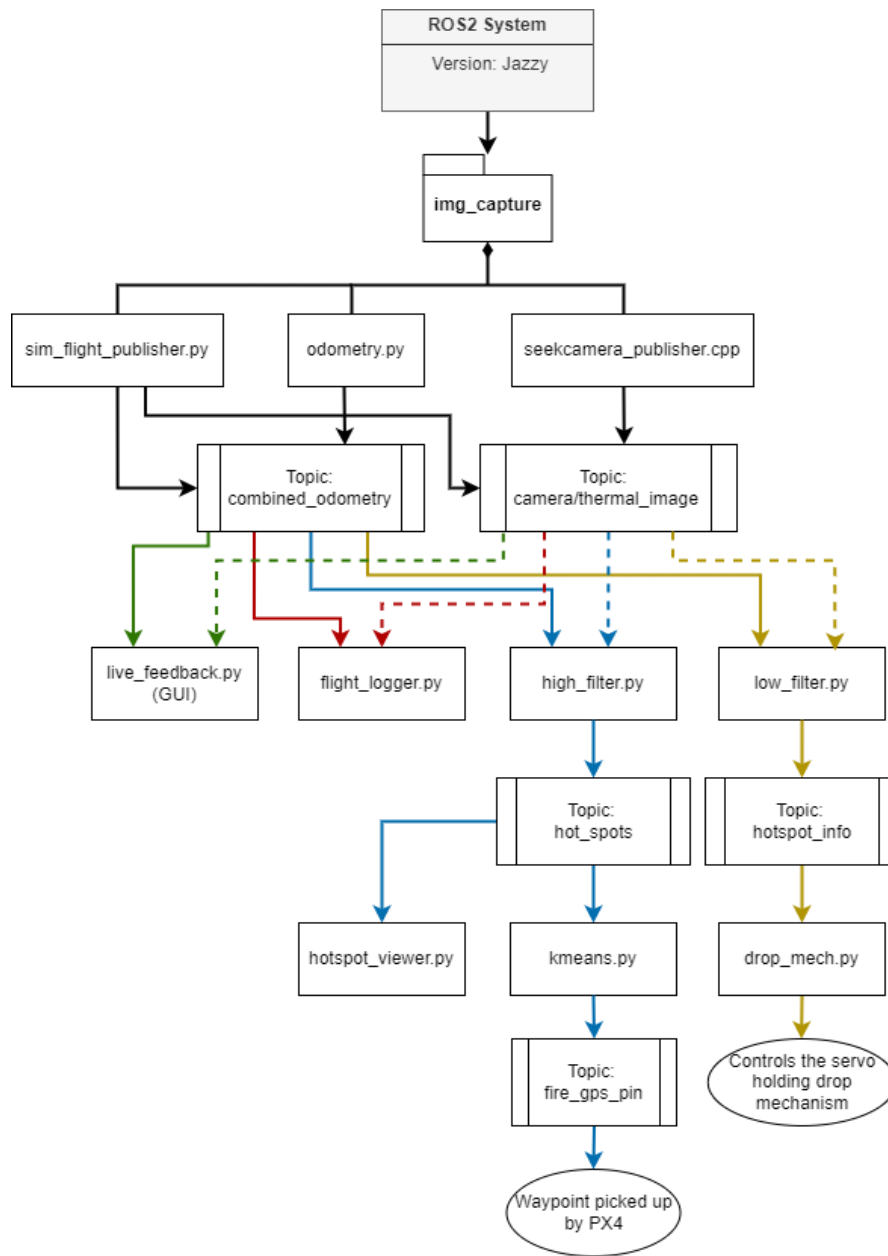
Prepared by: Joshua Crookston	Checked by: Jacob Wilkins
Purpose: Outlines general drone communications setup as well as problems to overcome in the future with potential proposed solutions.	

Revision History			
Revision	Revised by	Checked by	Date
01	Joshua Crookston	Israel Zenteno	Oct 10, 2024
02	Joshua Crookston	Israel Zenteno	Oct 14, 2024
03	Joshua Crookston	Israel Zenteno	Oct 18, 2024
04	Joshua Crookston	Jacob Wilkins	Oct 21, 2024
05	Joshua Crookston	Jacob Wilkins	Nov 18, 2024
06	Joshua Crookston	Jacob Wilkins	Jan 31, 2025
07	Joshua Crookston	Israel Zenteno	Feb 6, 2025
08	Joshua Crookston	Jonah Lowther	Feb 25, 2025



Artifact ID: ART-006	Artifact Title: High-altitude and Low-altitude Drone Code Overview		
Revision: 01	Revision Date: 2025-02-20		
Prepared by: Anthony Cardenas		Checked by: Isaac Davies	
Purpose: Explains the code flow for fire detection in the High-Altitude and Low-Altitude UAV, in relation to the Detection Code Block Diagram			

Revision History			
Revision	Revised by	Checked by	Date
01	Isaac Davies	Janie Linford	2025-02-20



High Altitude UAV Code Overview

Definitions:

Node - Code Block (shown with boxes)

Topic - Message Channel (shown with blocks with margins)

Publish – Send data to a topic

Subscribe – Collect data from a topic

Code Environment:

The high and low altitude UAVs are using Ros2 to separate tasks for Fire Detection. Ros2 is a common robotic code platform used for communication between code blocks that run concurrently. Our Code Block Diagram details the Ros2 node/topic communication. Our code is in python due to the range of libraries it provides, especially for image processing. The image input from the Seek Thermal Camera is done in C++, since the API connection is written in that language. It uses the "roscpp" library which allows us to use the C++ code in the Ros2 environment. The odometry input from the pix4 is done through the "px4_msgs" library, which is a public repository used to communicate with the pix4. All code is kept in a GitHub Repository called "XPRIZE-Snowflake".

Code Overview:

Node 1: seekcamera_publisher.cpp

Activates the Seek Thermal camera, and publishes the image data to the "camera/thermal_image" topic. This code uses the prebuilt API from the Seek Thermal company to communicate with the camera.

Node 2: odometry_publisher.py

Collects data from the "/fmu/out/vehicle_altitude" and "/fmu/out/vehicle_gps_position" topic that the pix4 publishes messages to. Then, it collects and combines the data into a single Json message that is sent to the "/combined_odometry" topic. This data includes latitude, longitude, altitude, pitch, roll, yaw, and a timestamp.

Node 3: high_alt_filter.py

Intakes images from the camera and odometry data from the "camera/thermal_image" and "/combined_odometry" topics, respectively. It starts by saving the first odometry data as a variable called "home_location". Once, it reaches an altitude above 30 meters, it matches the image and odometry data using their timestamps, and filters the images looking for the hottest spot. It returns the location of the hotspot relative to local GPS coordinates, and publishes the data to a topic called "hot_spots".

Node 4: k-means.py

Intakes GPS locations from the "hot_spots" topic and saves them to an array of hot spot locations. It then parses through locations and clusters them with the closest hot spot locations, which is an algorithm called k-means. This allows the data collected from several images to be averaged, improving accuracy of the GPS coordinates for each fire. When a k-means cluster has more coordinates than a set threshold, set by "min_size", the GPS coordinate is sent to a topic called "fire_gps_pin". In the future, these locations will be sent to qgroundcontrol and will be saved in flight_logger.py.

Node 5: low_alt_filter.py

Intakes images from the camera and odometry data from the “camera/thermal_image” and “/combined_odometry” topics, respectively. It starts by saving the first odometry data as a variable called “home_location”. Once, it reaches an altitude above 30 meters, it matches the image and odometry data using their timestamps, and filters the images looking for the hottest spot. It returns the location of the hotspot relative to the UAV, and publishes the data to a topic called “/hotspot_info”.

Node 6: drop_mech.py

Intakes local odometry data and user commands from the topics “/hotspot_info” and “servo_command”, respectively. It uses that data to determine when to open the drop mechanism. In automatic mode, it opens the drop mechanism when the hotspot is within 5 meters. In manual mode, it opens the drop mechanism when the user sends a command to the “/servo_command” topic. The Pi uses the “python3-gpiozero” and “python3-rpi.gpio” libraries to control the servo. It can run by itself, no other nodes need to be run. The default settings are manual mode and the servo is set to close.

The following commands can be sent to the “/servo_command” topic:

Switching to automatic or manual mode:

```
ros2 topic pub /servo_command std_msgs/msg/String "{data: \"auto\"}"
```

```
ros2 topic pub /servo_command std_msgs/msg/String "{data: \"manual\"}"
```

Controlling the servo:

```
ros2 topic pub /servo_command std_msgs/msg/String "{data: \"close\"}"
```

```
ros2 topic pub /servo_command std_msgs/msg/String "{data: \"open\"}"
```

Node 7: flight_logger.py

Code that subscribes to the “camera/thermal_image” and “/combined_odometry” topics, and logs the data collected during runtime. It pairs data from the camera and pix4 using their timestamps. On exit, it saves all data to a file called “flight_logs/high_alt_images_{date}.npy”. Doesn’t need to be run for “high_alt_filter.py” to work. For the moment, high and low altitude share this logger since they perform the same function on different Raspberry Pis that have their own logs.

Node 8: sim_flight_publisher.py

Simulates a flight path and publishes to the “camera_themal/image” and “/combined_odometry” topics. It is used to test the “high_alt_filter.py” node. This should not be run with “seek_camera_publisher.cpp” and “odometry_publisher.py” nodes. This path is premade and copied into the pi in a file called “sim_data/video_matrix16.npy”.

Node 9: live_feedback.py

This node subscribes to the “camera/thermal_image” and “combined_odometry” topics. This node filters the images using the “OpenCV” library and pairs it with odometry data. It then shows them on a GUI, so the user can see live camera data and filtered data. This node often crashes with low network connections, so it is separated from the “high_alt_filter.py” node. This allows all other nodes to continue. This node needs to be run with putty, which can open a GUI from your computer monitor.

See “FullSetupDocumentation” for how to run “live_feedback.py” node.

Using the Code:

In order to run each node, the following commands need to be sent from the terminal:

1. Enter the Pi wirelessly:

a. Option 1: In the lab, with Wi-Fi

ssh username@10.2.118.167

b. Option 2: In the field, without Wi-Fi

Create a hotspot from your computer or phone

On a windows computer go to: settings->network and internet->mobile hotspot

Find the new ip address of the Pi and enter using ssh:

ssh username@{ip_address}

2. Go to the ros2 workspace and build the package:

```
cd ros2_ws
```

```
colcon build --symlink-install --packages-select img_capture
```

```
source install/setup.bash
```

Note: If the pix4_msgs are creating an error, run:

```
colcon build --symlink-install --packages-select px4_msgs
```

3. Run the desired nodes:

Generic node activation:

```
ros2 run {package} {node}
```

a. Option 1: For testing in the lab:

ros2 run img_capture sim_flight_publisher.py

ros2 run img_capture high_alt_filter.py

ros2 run img_capture k-means.py

ros2 run img_capture live_feedback.py (in Putty, see FullSetupDoc)

b. Option 2: For testing in the field:

ros2 run img_capture seekcamera_publisher

ros2 run img_capture odometry_publisher.py

ros2 run img_capture flight_logger.py

ros2 run img_capture live_feedback.py (in Putty, see FullSetupDoc)

c. Option 3: Launch files

ros2 launch img_capture new_img_capture.launch.py [optional arguments]

ros2 launch img_capture new_img_capture.launch.py sim:=false log:=false high:=true

Here are the possible arguments:

sim:=false log:=false high:=true

sim – false: launches odometry_publisher.py and seekcamera_publisher

sim – true: launches sim_flight_publisher.py

log – true: launches flight_logger.py

log – false: doesn't launch flight_logger.py

high – true: launches high_alt_filter.py and kmeans.py

high – false: launches low_alt_filter.py and drop_mech.py

Print – log: only sends messages to the logger


Print – screen: prints messages directly on the terminal

Print – both: prints to screen and saves to the logger

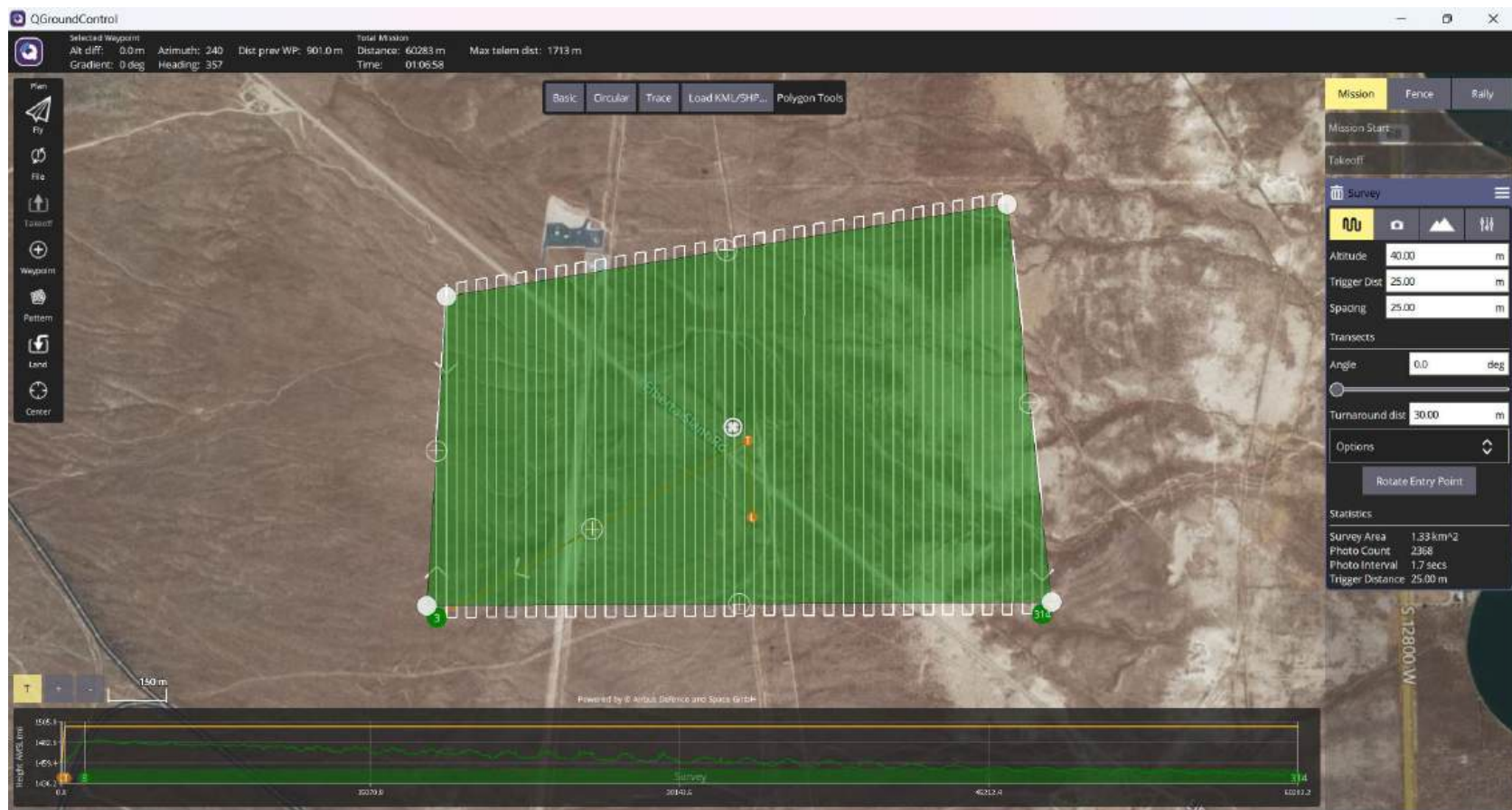
4. Shutdown the scripts

pgrep -f img_capture | xargs kill -9

This command will stop running code and save the log from flight_logger.py

Artifact ID: ART-007	Artifact Title: Demo High-altitude Drone Flight Plan		
Revision: 02	Revision Date: 2025-02-25		
Prepared by: Jonah Lowther		Checked by: Nina Chao	
Purpose: The standard lawnmower flight plan that will be used with our high-altitude drone to detect fire hotspots over a certain area.			

Revision History			
Revision	Revised by	Checked by	Date
01	Jonah Lowther	Nina Chao	2024-12-02
02	Joshua Crookston	Jonah Lowther	2025-02-25




Artifact ID: ART-008	Artifact Title: System Progress Overview
Revision: 08	Revision Date: 2025-02-11

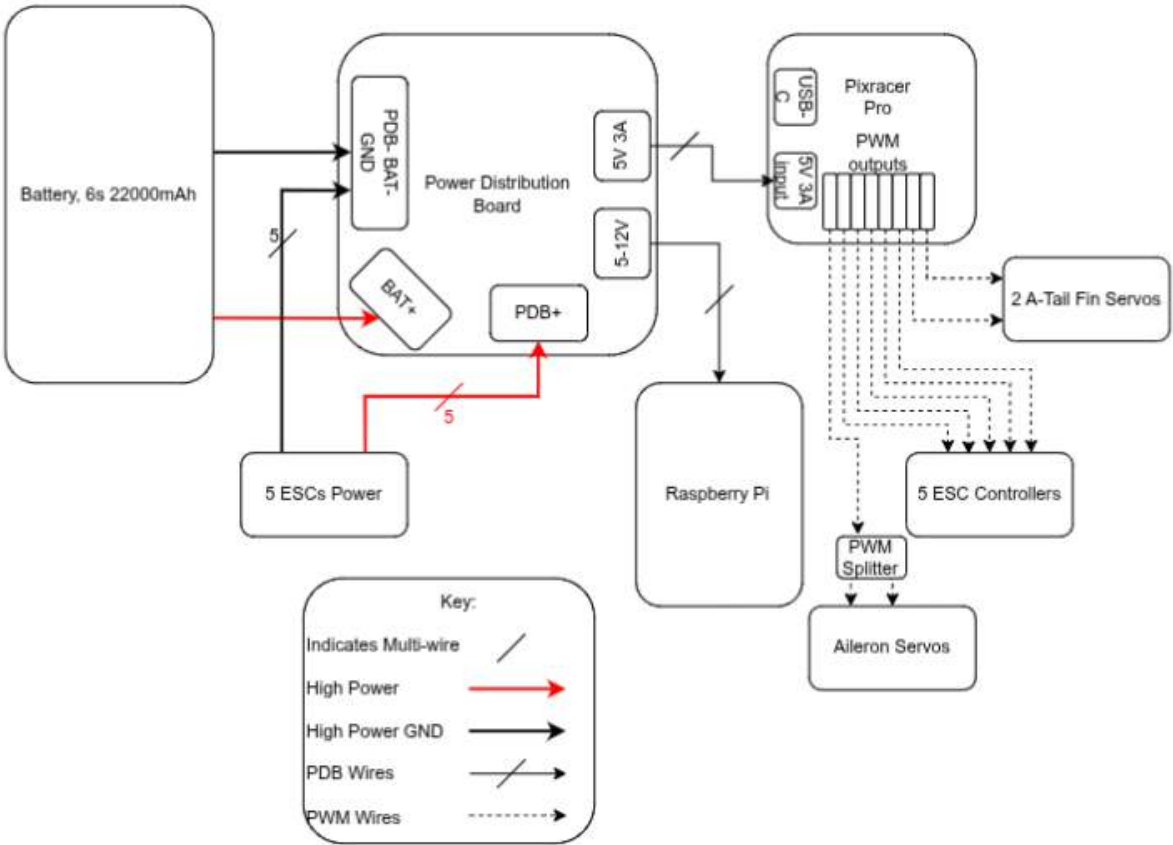


Prepared by: Isaac Davies	Checked by: Jonah Lowther
Purpose: Outlines general system setup as well as problems to overcome in the future with potential proposed solutions.	

Revision History			
Revision	Revised by	Checked by	Date
01	Isaac Davies	Jonah Lowther	2025-01-30
02	Isaac Davies	Jonah Lowther	2025-01-31
03	Isaac Davies	Jonah Lowther	2025-02-03
04	Isaac Davies	Jonah Lowther	2025-02-04
05	Isaac Davies	Jonah Lowther	2025-02-05
06	Isaac Davies	Jonah Lowther	2025-02-06
07	Israel Zenteno	Jonah Lowther	2025-02-07
08	Isaac Davies	Israel Zenteno	2025-02-11

Artifact ID: ART-009	Artifact Title: Low Altitude Drone Wiring Diagram	
Revision: 01	Revision Date: 2025-01-16	
Prepared by: Anna Holm		Checked by: Jacob Wilkins
Purpose: A diagram that outlines the wiring of the low-altitude drone, the HERO PNP		

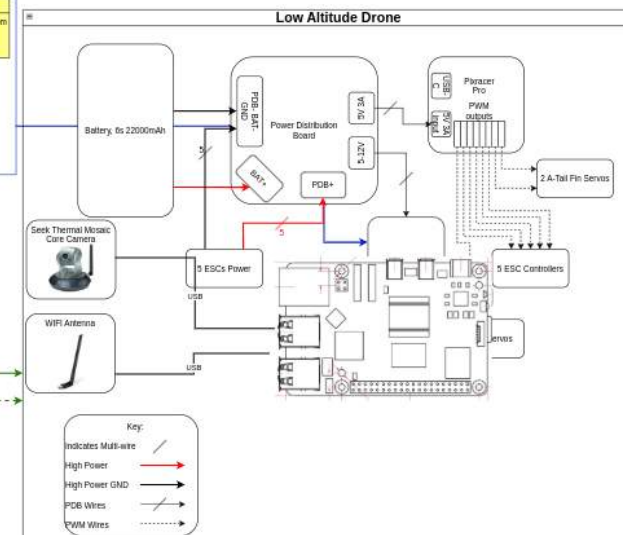
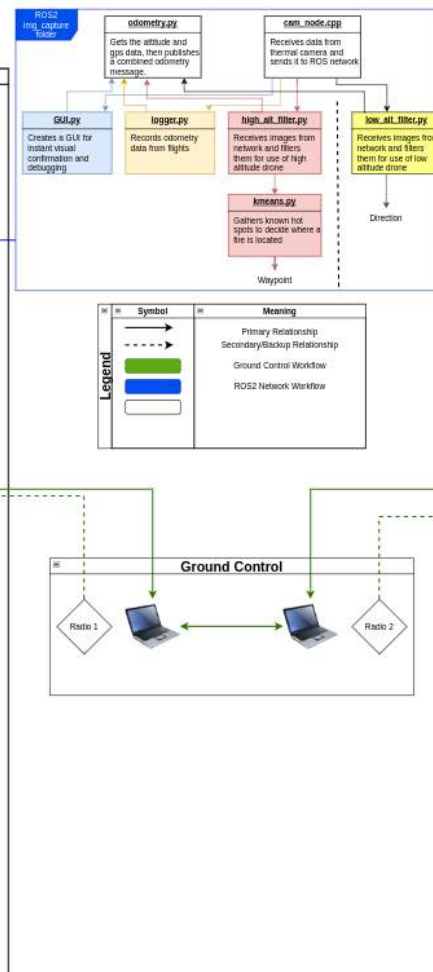
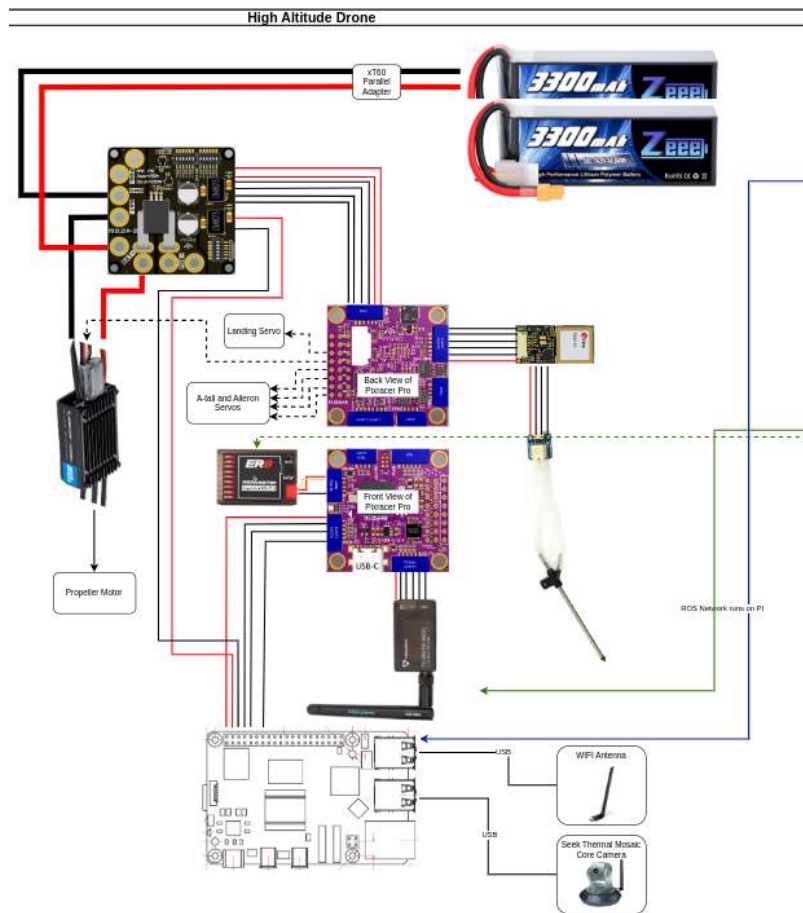
Revision History			
Revision	Revised by	Checked by	Date
01	Anna Holm	Jacob Wilkins	2025-01-16




Artifact ID: ART-011	Artifact Title: XPRIZE System Overview (Professional)
Revision: 01	Revision Date: 2025-02-25
Prepared by: Isaac Davies	Checked by: Jonah Lowther
Purpose: Outlines general system setup with high-quality connection diagrams, including wiring diagrams	

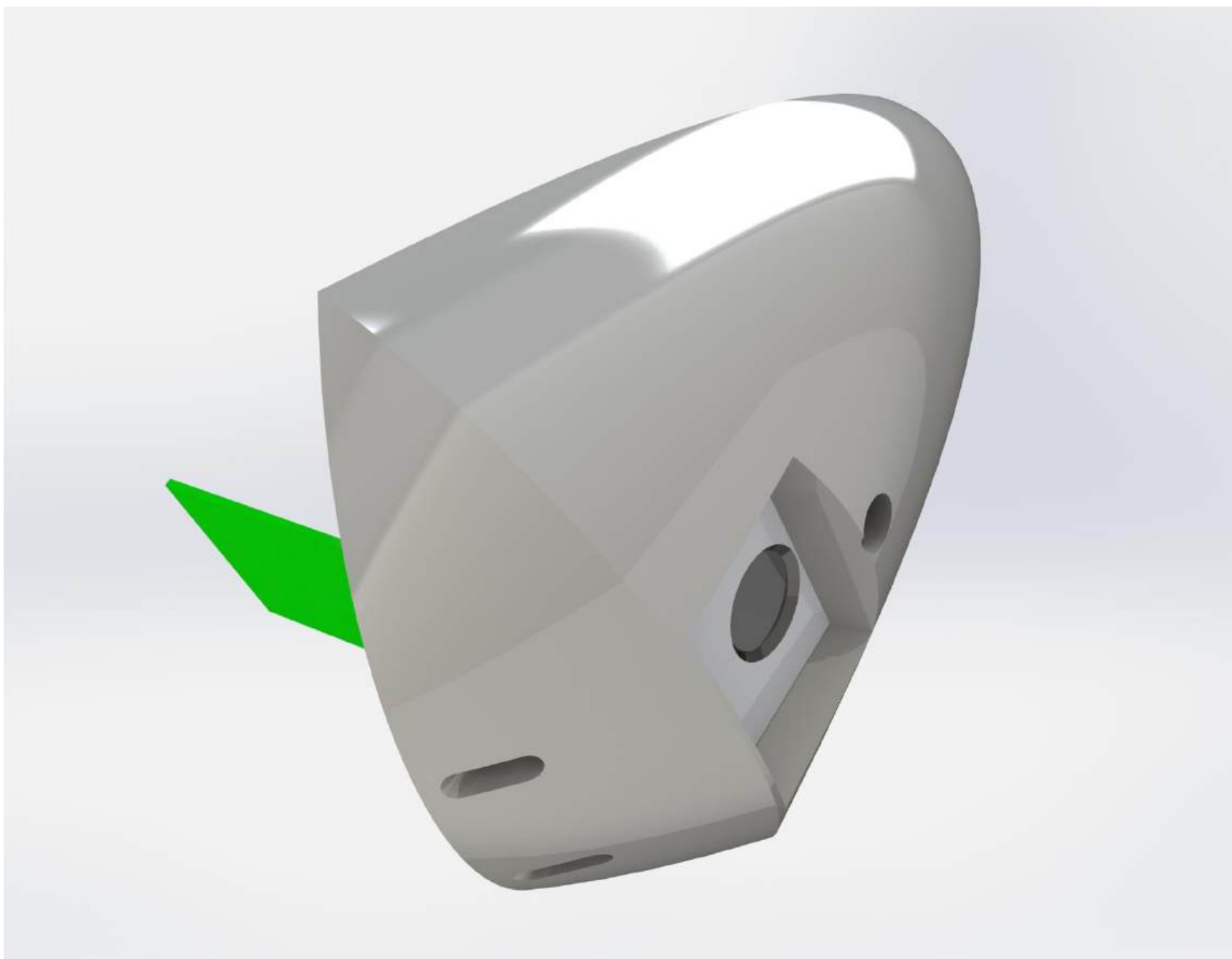



Revision History			
Revision	Revised by	Checked by	Date
01	Isaac Davies	Jonah Lowther	2025-02-25



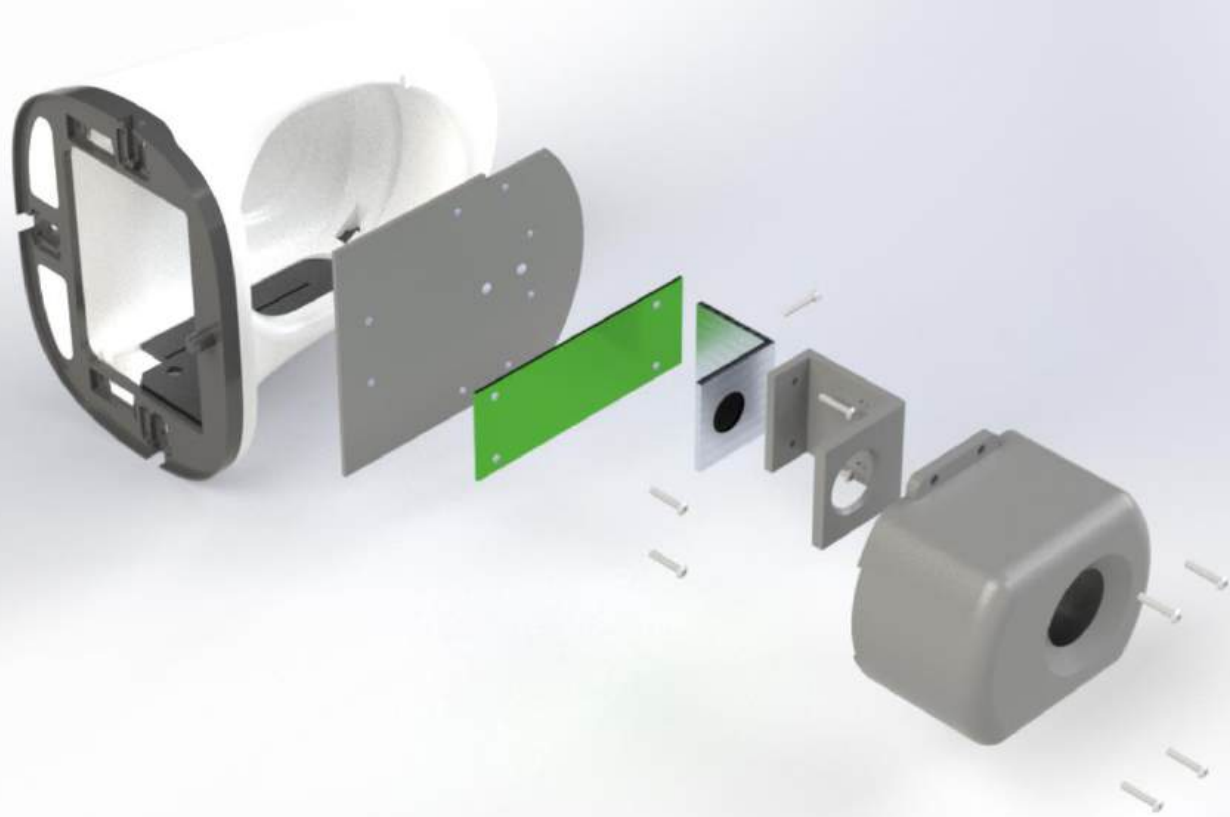
Artifact ID: ART-012	Artifact Title: High Altitude Camera Mount Assembly		
Revision: 01	Revision Date: 2025-02-24		
Prepared by: Blake Folsom		Checked by: Joshua Crookston	
Purpose: Assembly shows the camera mount with the camera in it. The camera mount was made to fit the high-altitude drone (Anaconda RC fixed wing airframe).			


Revision History			
Revision	Revised by	Checked by	Date
01	Blake Folsom	Joshua Crookston	2025-02-24



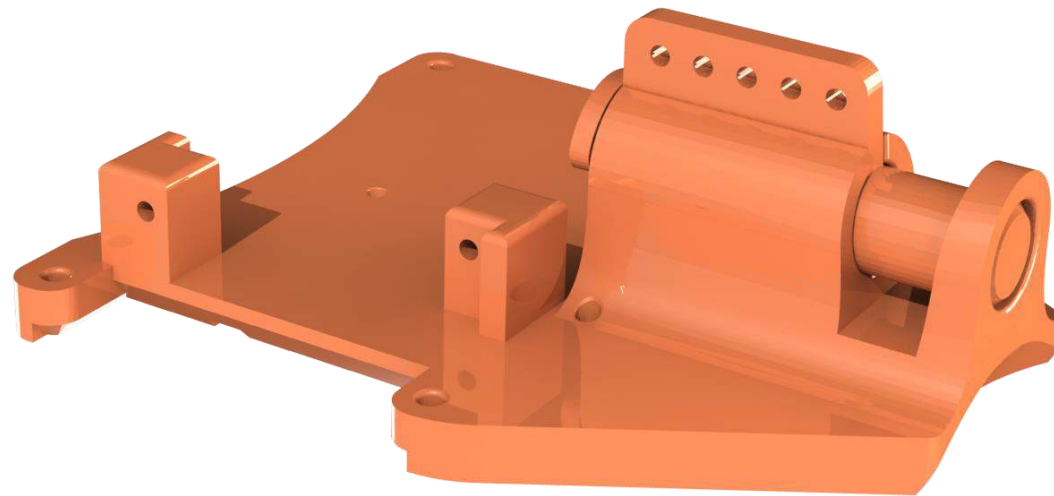
Artifact ID: ART-013	Artifact Title: Low Altitude Camera Mount Assembly		
Revision: 01	Revision Date: 2025-02-24		
Prepared by: Nina Chao		Checked by: Blake Folsom	
Purpose: Assembly shows an exploded view of the low altitude camera mount with the camera in it. The camera mount was made to fit the low altitude drone (Hero PNP).			


Revision History			
Revision	Revised by	Checked by	Date
01	Nina Chao	Blake Folsom	2025-02-24
02	Nina Chao	Blake Folsom	2025-03-05



Artifact ID: ART-014	Artifact Title: Proxy Fire Suppressant Drop Mechanism		
Revision: 01	Revision Date: March 5, 2025		
Prepared by: Jacob Wilkins		Checked by: Anna Holm	
Purpose: Assembly render of the pin servo mount and pin used to release proxy fire suppressant from the underbelly of the low-altitude drone.			

Revision History			
Revision	Revised by	Checked by	Date
01	Jacob Wilkins	Anna Holm	March 5, 2025
02	Jacob Wilkins	Anna Holm	March 10, 2025



Artifact ID: ART-015	Artifact Title: Long-Term Safety Plan		
Revision: 01	Revision Date: 2025-03-12		
Prepared by: Nina Chao		Checked by: Jonah Lowther	
Purpose: This is the year-long safety plan we used for all our test flights. This safety plan was approved by Bryant Brown, Steve McLean, and Darin Childers from BYU Risk Management. Additional approval from Bryant to set fires for camera testing at Kiwanis Park and test fly the drones at Rock Canyon Park was received later.			

Revision History			
Revision	Revised by	Checked by	Date
01	Nina Chao	Jonah Lowther	2025-03-12

Safety Plan

Any job, task, activity, or event can benefit from having a safety plan that contains information about identifying and controlling hazards, responding to emergencies, and ensuring that affected individuals are properly trained.

Purpose and Scope

This Safety Plan is used with the following job, task, activity, or event:

Where will it be used?

The expected user(s) of this plan is/are:

This safety plan is intended for _____ use, and expires on _____

Responsible Persons

The person(s) responsible for ensuring compliance with this Safety Plan is/are:

This plan was written by _____ on this date _____

It is due for review by _____ on this date: _____

Hazard Identification

Has a Job Hazard Assessment (JHA) been completed for this activity? Yes No

Is there a written Standard Operating Procedure (SOP) for completing this activity? Yes No

Have there been any safety incidents or 'near miss' events associated with this activity? Yes No

Are any of the following potential hazards present or likely to be present? (check all that apply)

Hazardous Chemicals (physical and/or health hazards)	Rotating Equipment
Hazardous Atmosphere (IDLH, oxygen deficient, etc.)	Flying Debris/Particles
Hazardous Energy (electromagnetic, pneumatic, pressure, etc.)	High Noise
Ergonomic (repetitive motion, body position, etc.)	Slips/Trips/Falls
Readiness to work (fatigue, stress, distractions, etc.)	Lift/Move/Transport
Blood Borne Pathogens & Infectious Agents	Confined Space
Fire/Steam/Hot Work	Crushed/Caught In
Pinch Points/Sharp Edges	Animal Handling
Other:	

Hazard Controls and Safe Practices

Which of the following will be used to mitigate or eliminate the hazards associated with this activity?

- ☐ Substitution/Elimination (Use less hazardous alternatives or eliminate the hazard completely)
- ☐ Engineering Controls (Mechanical safety devices)
- ☐ Administrative Controls (Safety Policies, Standard Operating Procedures, Safety Data Sheets)
- ☐ Personal Protective Equipment (Goggles, Gloves, Apron, Face Shield, Respiratory Protection, Hearing Protection, Fire-Resistant Apparel, Safety Shoes, etc.)
- ☐ Proper Authorization
- ☐ Training/Supervision
- ☐ Correct tools and equipment
- ☐ Increased Awareness
- ☐ Appropriate Scale/Scope
- ☐ Machine Guarding
- ☐ Time/Shielding/Distance
- ☐ Barriers/Cones/Tape
- ☐ Chemical Storage
- ☐ Waste Management
- ☐ Guardrails/Fall Arrest
- ☐ Lock Out Tag Out
- ☐ Confined Space Permit
- ☐ Defensive Driving
- ☐ Other: _____

Describe in detail any controls identified above, including how & where they will be used:

What conditions necessitate immediately stopping the activity?

- Uncontrolled Fire Detection: If fire becomes larger than our size limitations or exceeds the boundaries of the fire pit, the fire will be extinguished and the activity will cease.

Emergency and Incident Response

If help is needed, call: 911, BYU Police Dispatch at (801) 422-2222, and/or

In case of emergency...

is responsible for completing an online Incident Report, available at

risk.byu.edu or by [clicking here](#).

Safety-related or other pertinent information associated with this task that is not identified above (if applicable)

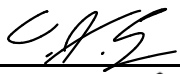
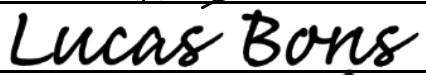
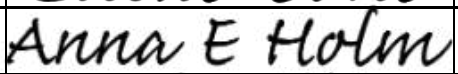
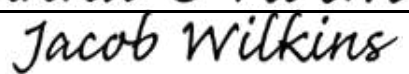

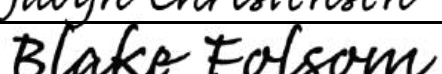
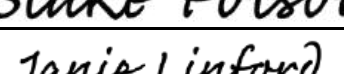

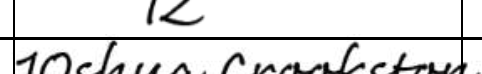
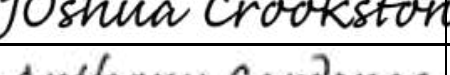
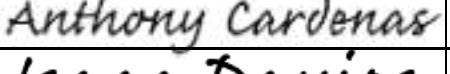
Communication, Training, and Recordkeeping


The following individuals, work teams, or groups require training about this Safety Plan:

Safety plan training is required before beginning the associated task, after any significant changes to the plan, and at the following intervals:

By signing below, I certify that:


1. I have been trained on the content of this Safety Plan.
2. I have had an opportunity to ask questions and resolve concerns about this plan.
3. I will use this plan, and other resources as needed, to help me work safely and minimize injury, loss, or damage.

Name (print legibly)	Signature	Training Completion Date
Jonah Lowther		11/25/2024
Lucas Bons		11/25/2024
Anna Holm		11/25/2024
Jacob Wilkins		11/25/2024
Jadyn Christensen		11/25/2024
Blake Folsom		11/25/2024
Janie Linford		11/25/2024
Israel Zenteno		11/25/2024
Joshua Crookston		11/25/2024
Anthony Cardenas		11/25/2024
Isaac Davies		11/25/2024

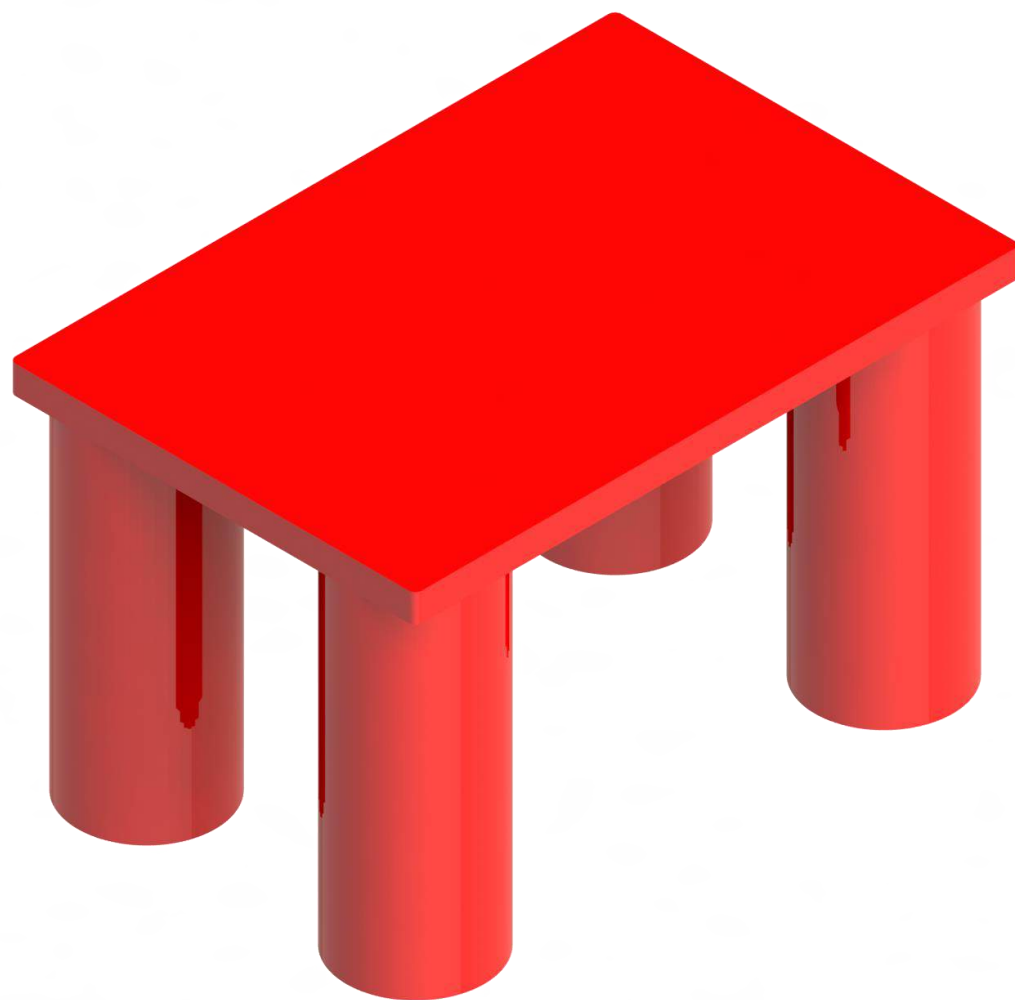
Nina Chao  11/25/2024


Tristan Mott  11/25/2024

Vincent Carter  11/25/2024

Artifact ID: ART-016	Artifact Title: Raspberry PI Mount		
Revision: 02	Revision Date: 2025-03-19		
Prepared by: Jacob Wilkins		Checked by: Blake Folsom	
Purpose: The mount is made of PLA material and is glued inside the fuselage of the low altitude drone. The Raspberry Pi is secured to the mount with a velcro strip.			

Revision History			
Revision	Revised by	Checked by	Date
01	Jacob Wilkins	Blake Folsom	2025-02-24
02	Jacob Wilkins	Blake Folsom	2025-03-19



Artifact ID: ART-019	Artifact Title: High Altitude Camera Mount Drawing		
Revision: 01	Revision Date: February 24, 2025		
Prepared by: Blake Folsom		Checked by: Nina Chao	
Purpose: Gives key dimensions for the high-altitude camera mount. This drawing could be used to recreate the mount. However, the geometry is complex, and the CAD files should be used and relied on. (CAD files: https://byu.box.com/s/vil5nlui6tnnyj08t3dsop6s4aceqpif)			

Revision History			
Revision	Revised by	Checked by	Date
01	Nina Chao	Blake Folsom	March 25, 2025

4

3

2

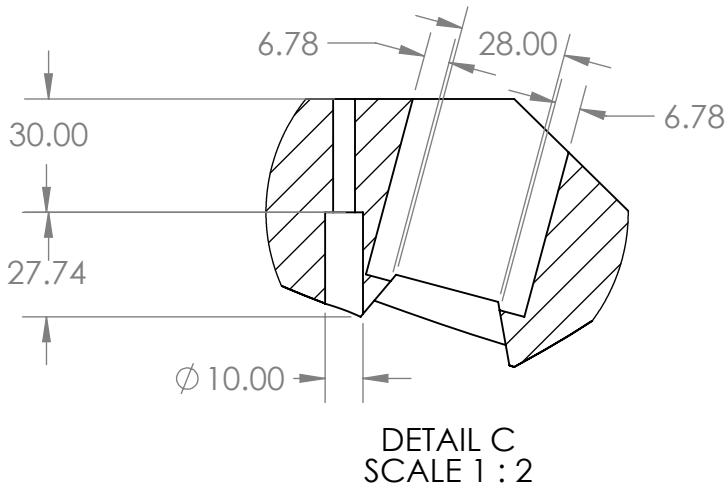
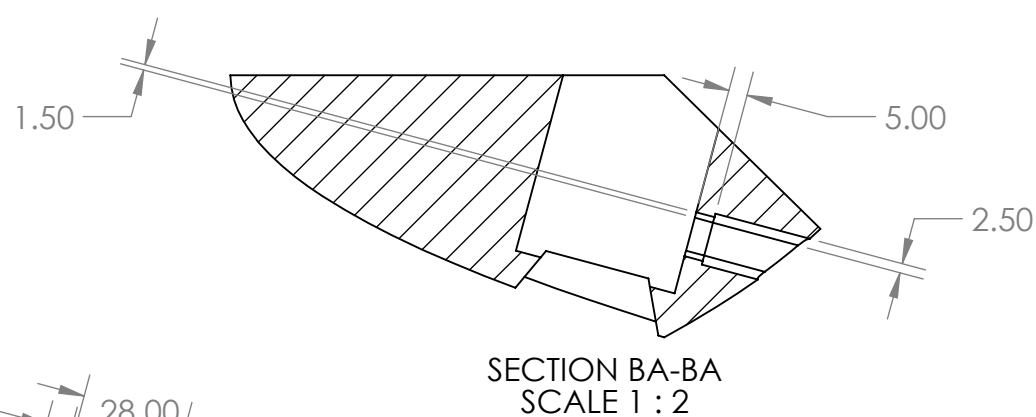
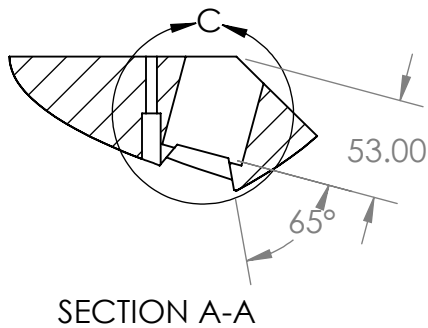
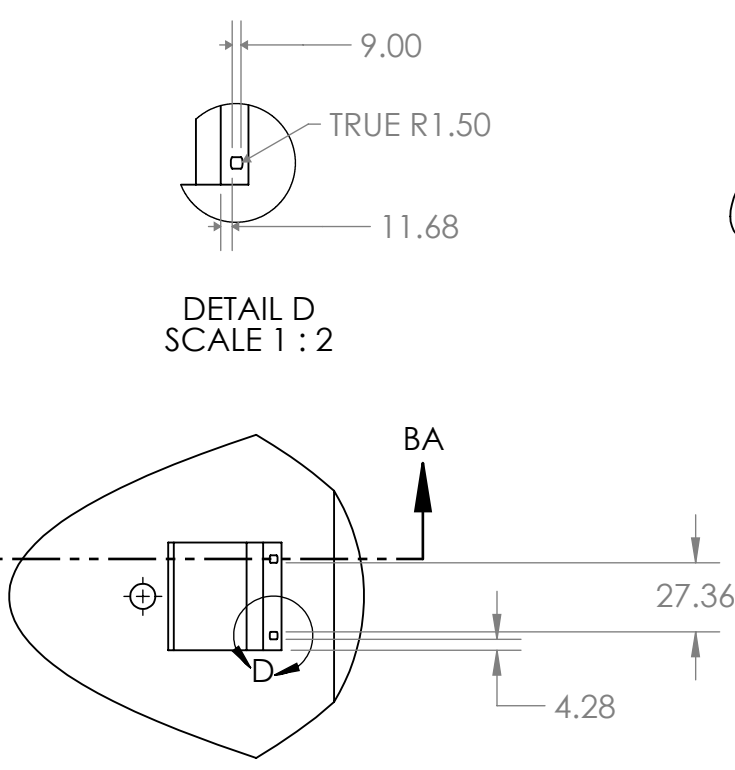
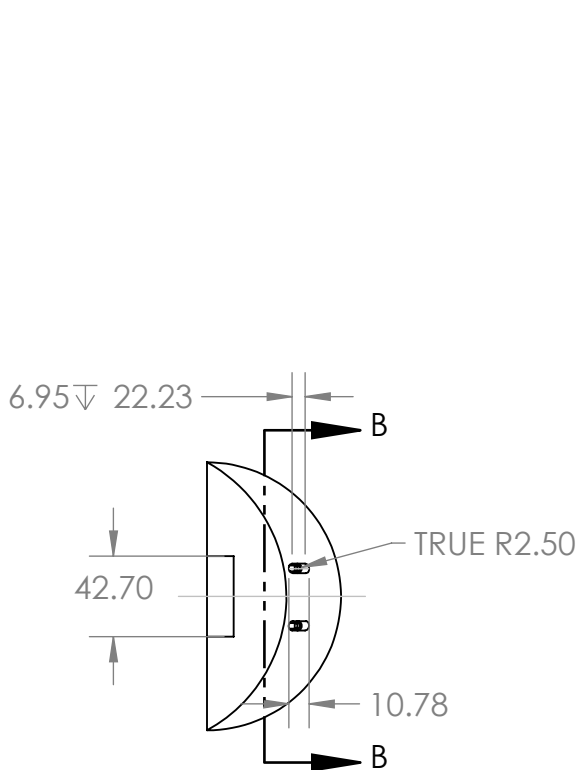
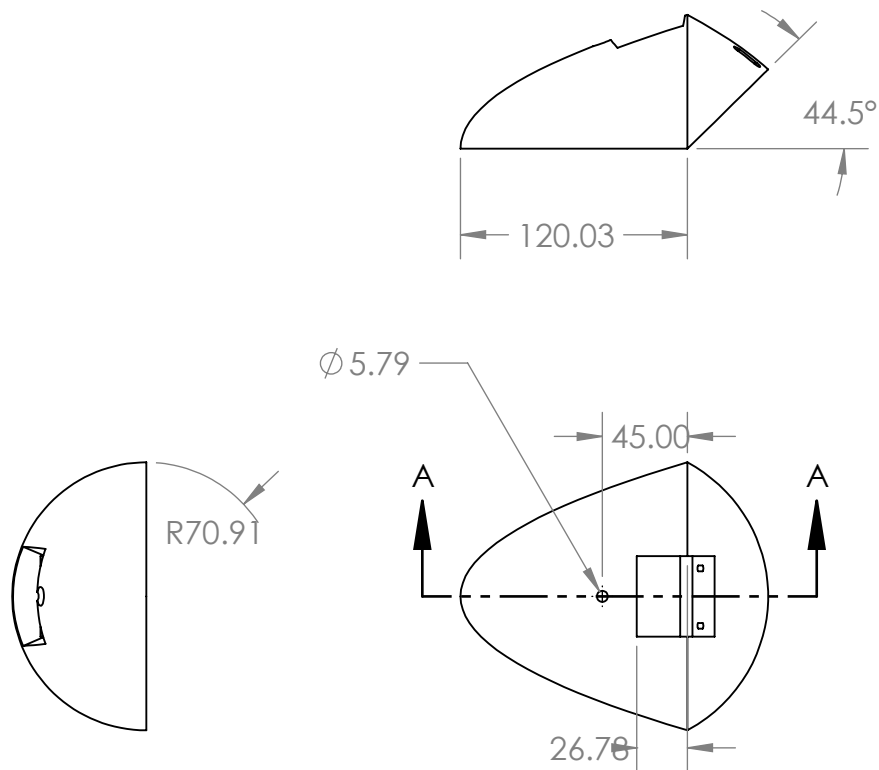
1

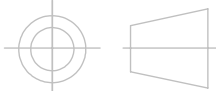
B

B

A

A




UNLESS OTHERWISE SPECIFIED TOLERANCES ARE		NAME	DATE	Capstone-XPRIZE Wildfire	
INCH:		DRAWN BY:	B.FOLSOM		
MM:		CHECKED BY:	N.CHAO	3/25/25	PART NAME:
0.0 = ± 0.06		UNITS:	Millimeters		HIGH ALT. CAMERA MOUNT
0.00 = ± 0.06		MATERIAL:	PLA		
0.000 = ± 0.009		FINISH:			
ANGULAR:		COMMENTS:			
0.000 = ± 0.05					
0.0 = ± 1					
0.00 = ± 0.5					
					
DO NOT SCALE DRAWING		SIZE		PART NO.	REV
		B		1	A
		SCALE: 1:4		WEIGHT:	SHEET 1 OF 1

3

2

1

Artifact ID: ART-020	Artifact Title: Low Altitude Camera Mount Drawing		
Revision: 01	Revision Date: March 26, 2025		
Prepared by: Blake Folsom		Checked by: Nina Chao	
Purpose: Gives key dimensions for the Low-altitude camera mount. This drawing could be used to recreate the mount. If needed see CAD file:(https://byu.box.com/s/a5oytlj8etgj9xv5vvqfr7qxuwtz4pn7)			

Revision History			
Revision	Revised by	Checked by	Date
01	Nina Chao	Blake Folsom	March 26, 2025

4

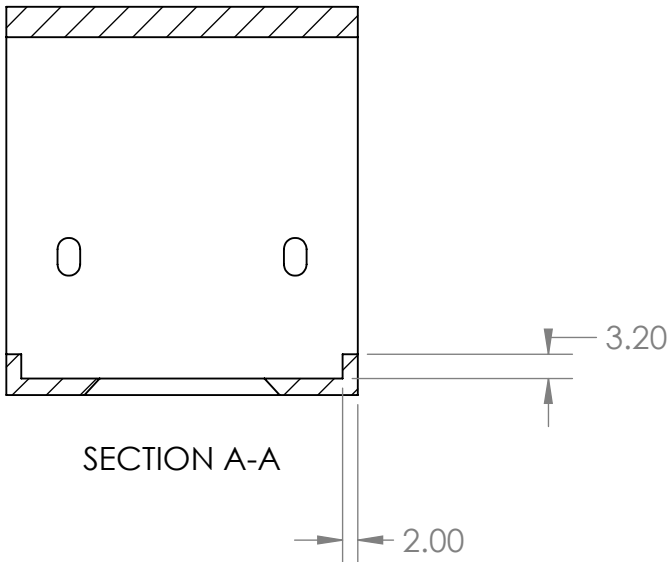
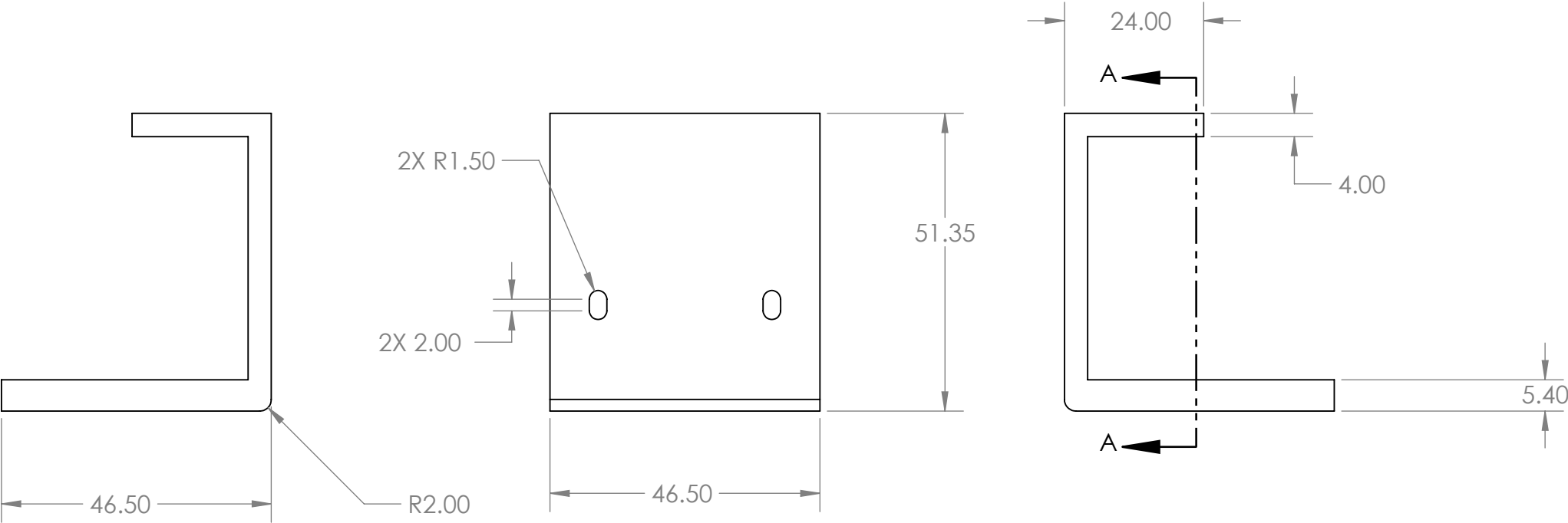
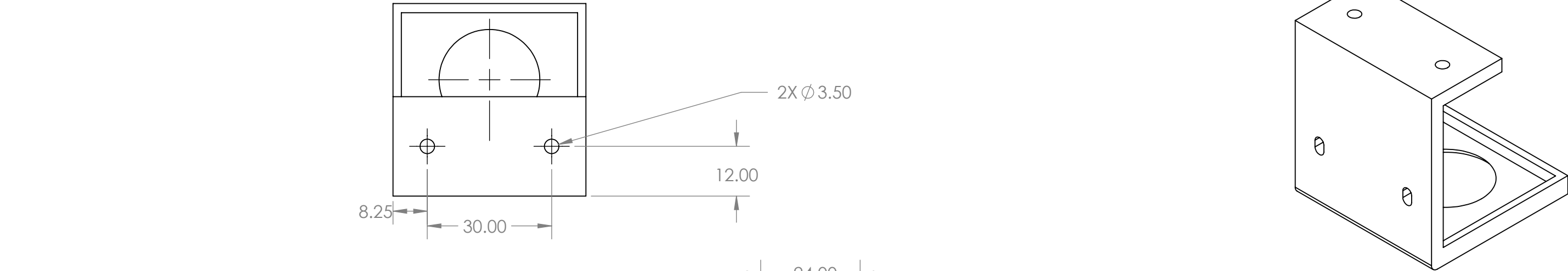
3

2

1

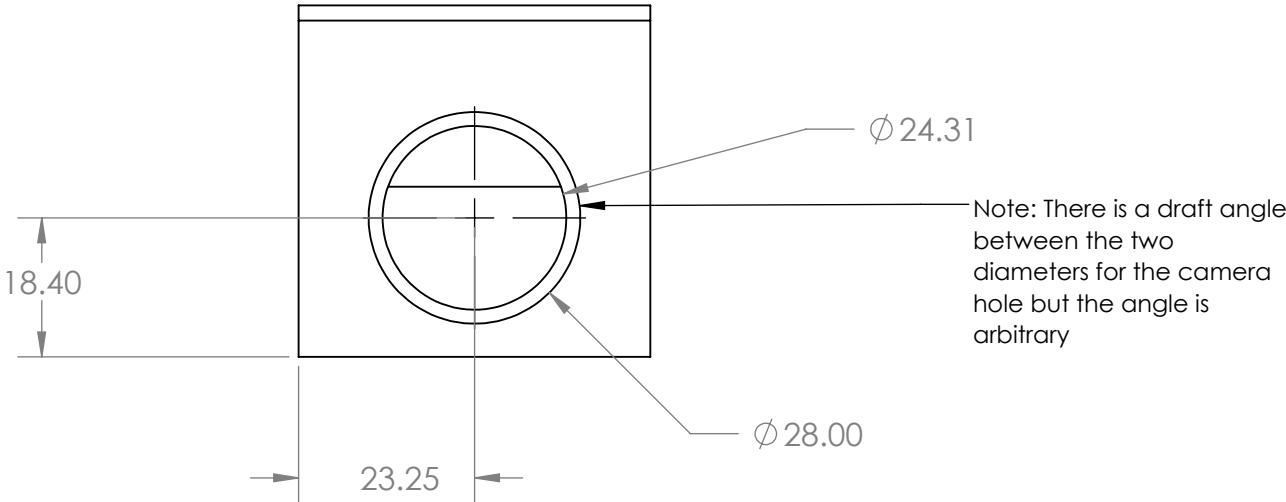
B

B



A

A




UNLESS OTHERWISE SPECIFIED TOLERANCES ARE		NAME	DATE	Capstone-XPRIZE Wildfire PART NAME: LOW ALT. CAMERA MOUNT	
INCH:	MM:	DRAWN BY:			
0.0 = ± 0.06	0. = ± 0.50	CHECKED BY:			
0.00 = ± 0.06	0.0 = ± 0.25	UNITS:			
0.000 = ± 0.009	0.00 = ± 0.15	MATERIAL:		SIZE PART NO. REV	
ANGULAR:	0.000 = ± 0.05	FINISH:			
0.0 = ± 1		COMMENTS:		SCALE: 1:1 WEIGHT: SHEET 1 OF 1	
0.00 = ± 0.5					
DO NOT SCALE DRAWING					

3

2

1

Artifact ID: GD-001	Artifact Title: Instructions for Rpi5 Integration with PixRacer Pro and Ros 2 Jazzy		
Revision: 13	Revision Date: 2025-03-04		
Prepared by: Israel Zenteno		Checked by: Janie Linford	
Purpose: Instructions for initially integrating the Raspberry pi 5 with PixRacer Pro and Ros 2 Jazzy, establishing communication between the PixRacer Pro and its companion computer.			

Revision History			
Revision	Revised by	Checked by	Date
01	Israel Zenteno	Janie Linford	2024-10-16
02	Joshua Crookston	Isreal Zenteno	2024-10-21
03	Janie Linford	Joshua Crookston	2024-11-22
04	Israel Zenteno	Joshua Crookston	2024-11-25
05	Janie Linford	Israel Zenteno	2025-01-13
06	Janie Linford	Joshua Crookston	2025-01-14
07	Janie Linford	Israel Zenteno	2025-01-15
08	Janie Linford	Israel Zenteno	2025-01-17
09	Joshua Crookston	Janie Linford	2025-01-22
10	Joshua Crookston	Janie Linford	2025-01-23
11	Joshua Crookston	Israel Zenteno	2025-01-27
12	Janie Linford	Joshua Crookston	2025-02-04
13	Joshua Crookston	Israel Zenteno	2025-03-04
14	Joshua Crookston	Anthony Cardenas	2025-04-07

Initial Setup Guide

for Raspberry Pi 5 with PixRacer
Pro and ROS 2 Jazzy

Author: Team 56 “Snowflake”

Date: 2025-03-04

Table of Contents

Initial Setup

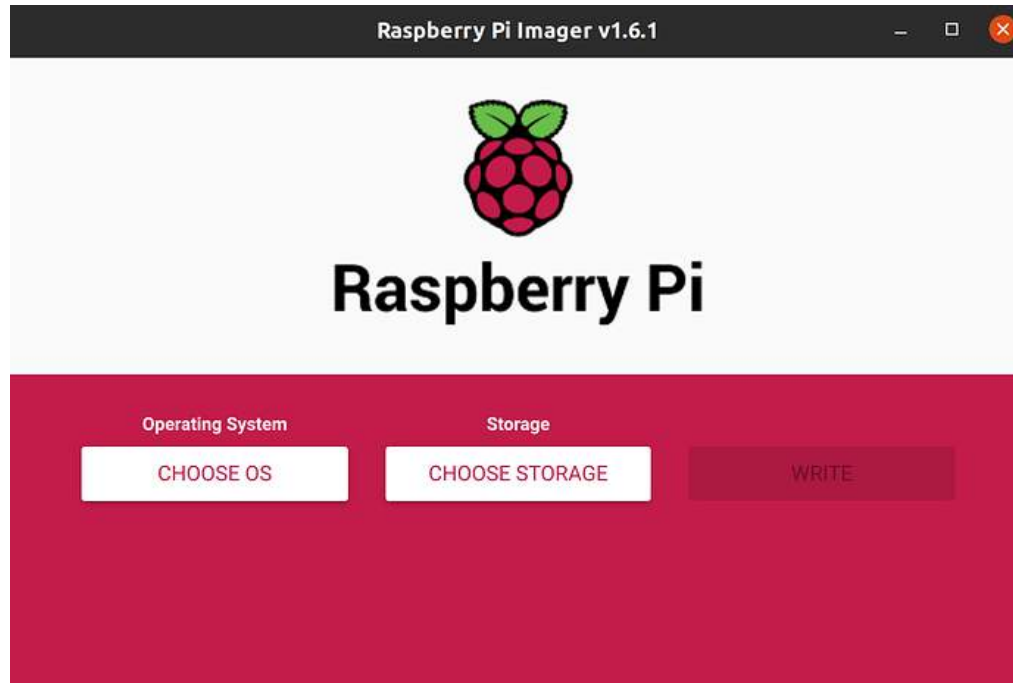
Flash SD Card with Ubuntu 24.04 Server.....	4
Boot Ubuntu and Set Up SSH.....	6
Enable UART0 on RPi	6
ROS 2 “Jazzy” Installation	7
ROS 2 Workspace Setup	7
PixRacer Pro Setup.....	8
RPi and PixRacer Pro Wiring	12
uXRCE_DDS Agent Installation.....	13
uXRCE_DDS Agent Activation	13
SSH setup with GUI to view Images/Graphs.....	15
Servo Control Using RPi	15
Appendix	16

Flash SD Card with Ubuntu 24.04 Server

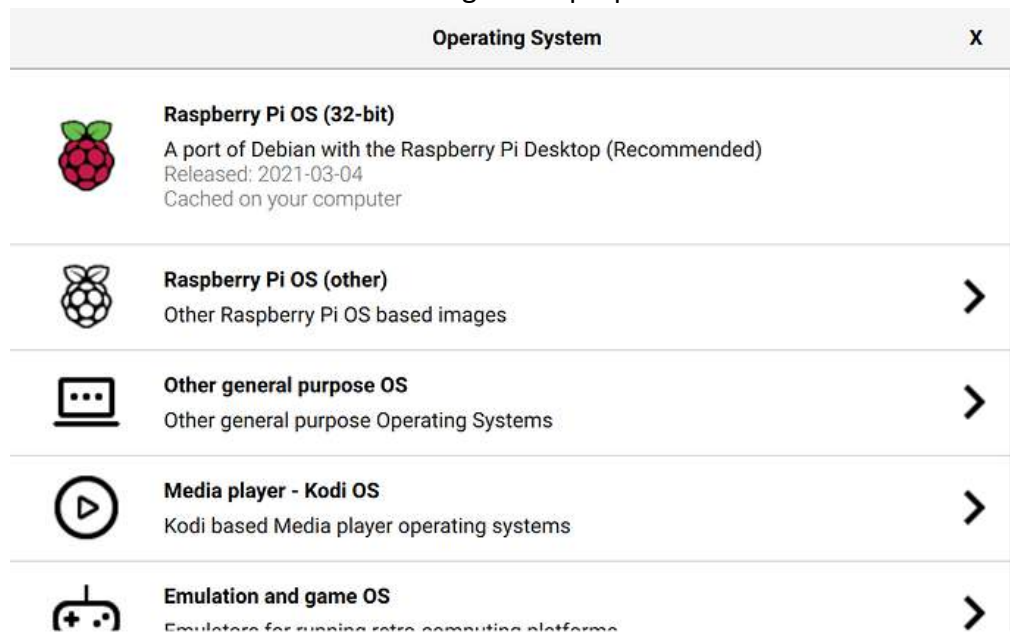
1. Install the right Raspberry Pi Imager for your operating system:

<https://www.raspberrypi.com/software/>

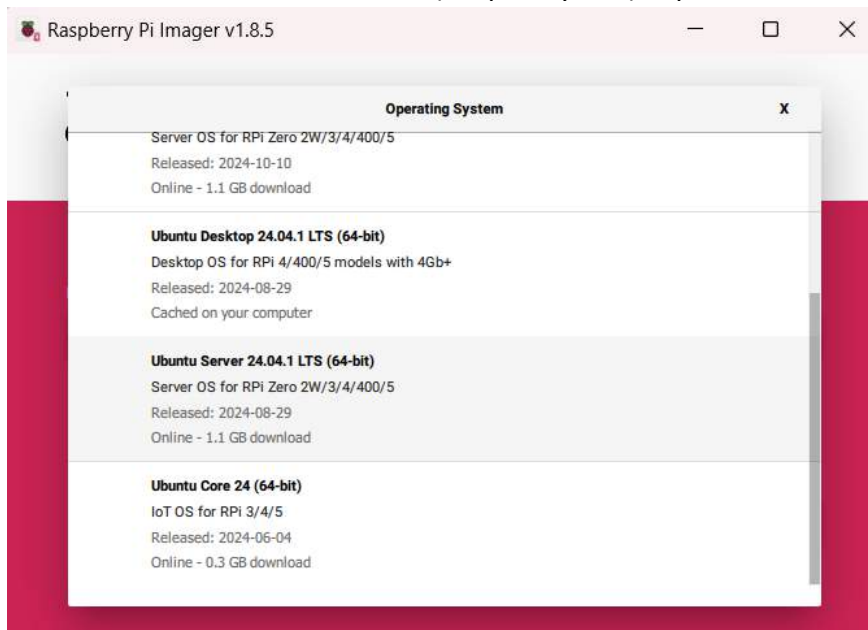
2. Start the Imager and open the “CHOOSE OS” menu



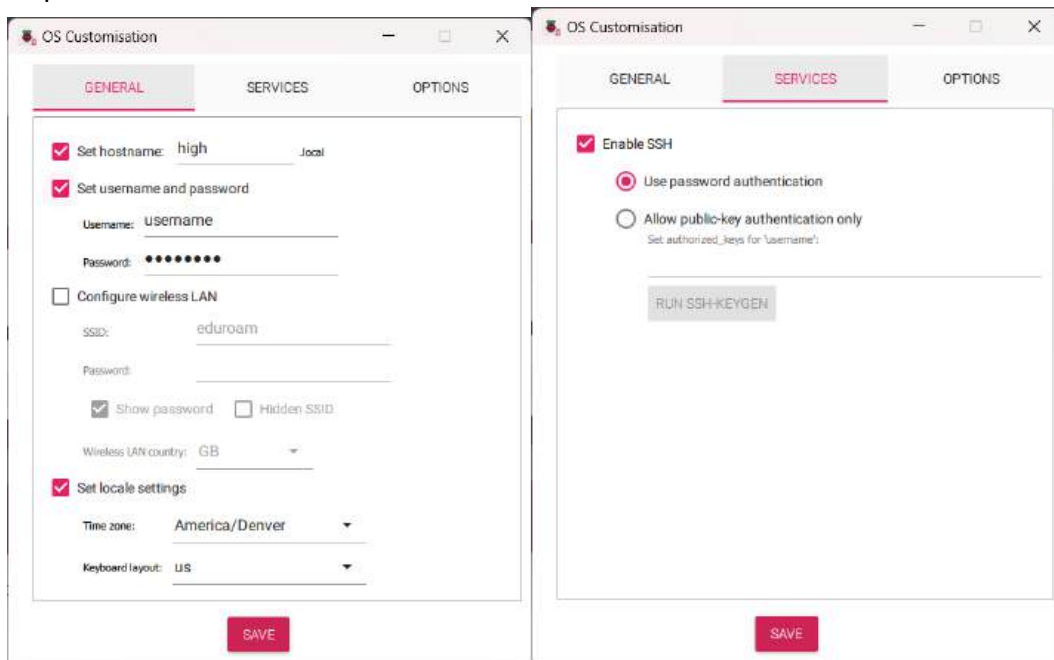
3. Scroll down the menu click “Other general-purpose OS”



4. Select the “Ubuntu Server 24.04 (Raspberry Pi 5)” option



5. Open the “SD Card” menu and select the microSD card you have inserted, and click “WRITE”. Once complete, make sure your Pi is off and insert this SD card.
6. Go through the OS customization options to configure the OS.
Note. you are unable to connect to eduroam this way due to the authentication it requires.



Boot Ubuntu and Set Up SSH

1. Ensure your HDMI screen and a USB keyboard are plugged into the Raspberry Pi 5 before plugging in and powering on
2. Complete the boot process and prompted setup
3. Update the system:
 - i. `sudo apt update -y && sudo apt upgrade -y && sudo apt full-upgrade -y`
4. Install essential tools and dependencies:
 - i. `sudo apt install -y git curl wget build-essential openssh-server software-properties-common python3-pip`
5. Enable and start ssh:
 - i. `sudo systemctl enable ssh`
 - ii. `sudo systemctl start ssh`
6. Reboot the system:
 - i. `sudo reboot`
7. After reboot, check the IP address:
 - i. `ip addr`

Enable UART0 on RPi

On Raspberry Pi 5, the primary UART appears on the Debug header. All other UARTs are disabled by default and need to be enabled. We specifically need to enable UART0.

1. Install raspi-config:
 - i. `sudo apt-get install raspi-config`
2. Open raspi-config:
 - i. `sudo raspi-config`
3. In raspi-config:
 - i. Go to "Interface Options"
 - ii. Select "Serial Port"
 - iii. Choose "No" to disable serial login shell
 - iv. Choose "Yes" to enable the serial interface
 - v. Select "Finish" and choose to restart the Raspberry Pi
4. After reboot, edit the boot configuration file:
 - i. `sudo nano /boot/firmware/config.txt`
5. Enable UART0 by adding the following lines of code to the bottom of the file:
 - i. NOTE: Other instructions will tell you to put `"dtoverlay=disable-bt"`, but doing so on the Raspberry Pi 5 will not let you connect to the PixRacer Pro!
 - ii. `enable_uart=1`

- iii. `dtoverlay=uart0`
- 6. Exit the file by typing `ctrl+x`, `ctrl+y`, and enter
- 7. Reboot the RPi:
 - i. `sudo reboot`
- 8. Verify the serial port is available:
 - i. `ls /dev/ttyAMA0`

The serial port should now be available as `/dev/ttyAMA0` (also accessible as `/dev/serial0`).

ROS 2 “Jazzy” Installation

1. Set up ROS 2 Jazzy repositories on the RPi:
 - a. `sudo add-apt-repository universe`
 - b. `sudo apt update && sudo apt install -y curl gnupg lsb-release`
 - c. `sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg`
 - d. `echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null`
2. Install ROS 2 Jazzy base and development tools:
 - a. `sudo apt update`
 - b. `sudo apt install -y ros-jazzy-ros-base ros-dev-tools`

RESUME INSTRUCTIONS HERE

3. Install Python dependencies:
 - a. `sudo apt install pipx`
 - b. `pipx install --user --break-system-packages -U empy==3.3.4 pyros-genmsg setuptools`
4. Install colcon build system and additional tools:
 - a. `sudo apt install -y python3-colcon-common-extensions python3-rosdep`
 - b. `python3-vcstool`
 - c. `sudo rosdep init`
 - d. `rosdep update`

ROS 2 Workspace Setup

1. Create a ROS 2 workspace and clone necessary repositories on the RPi:
 - a. `mkdir -p ~/ros2_ws/src`
 - b. `cd ~/ros2_ws/src`
 - c. `git clone https://github.com/PX4/px4_msgs.git`

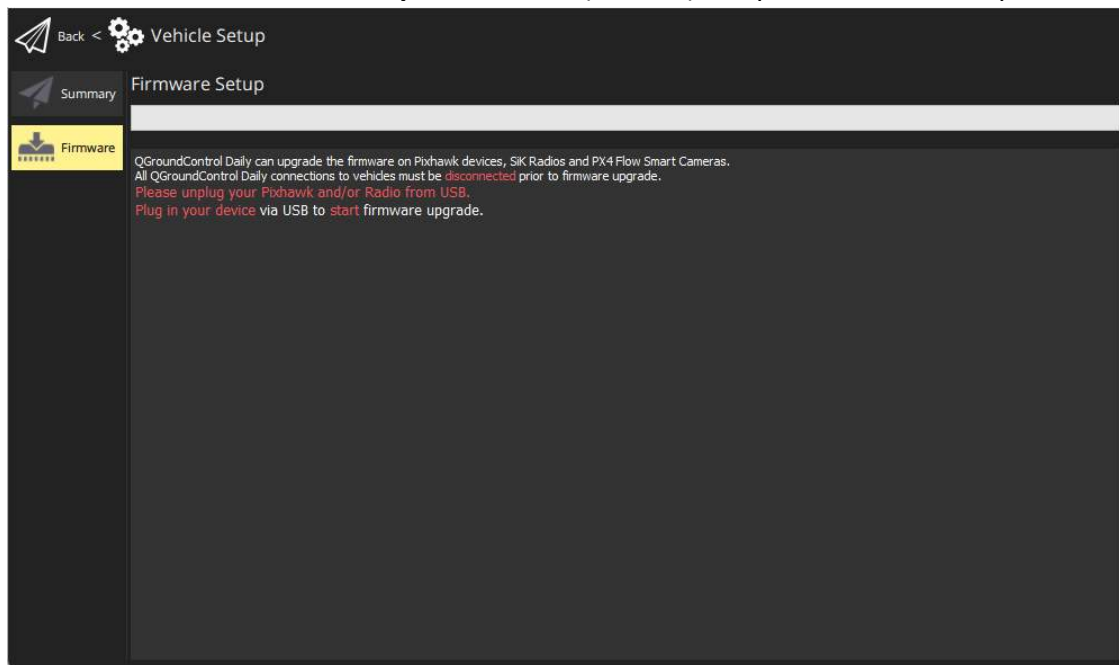
- d. `git clone https://github.com/PX4/px4_ros_com.git`
 - e. `cd ..`
- 2. Install dependencies and build the workspace:
 - a. `source /opt/ros/jazzy/setup.bash`
 - b. `sudo rosdep init`
 - c. `rosdep update`
`rosdep install --from-paths src --ignore-src -r -y`
 - d. `export MAKEFLAGS="-j4"`
 - e. `colcon build --parallel-workers 4`
- 3. Set up environment to source ROS 2 and workspace on terminal startup:
 - a. `echo "source /opt/ros/jazzy/setup.bash" >> ~/.bashrc`
 - b. `echo "source ~/ros2_ws/install/setup.bash" >> ~/.bashrc`
 - c. `source ~/.bashrc`
- 4. Verify the ROS 2 setup:
 - a. `echo $ROS_PACKAGE_PATH`
 - b. `ros2 --version`

PixRacer Pro Setup

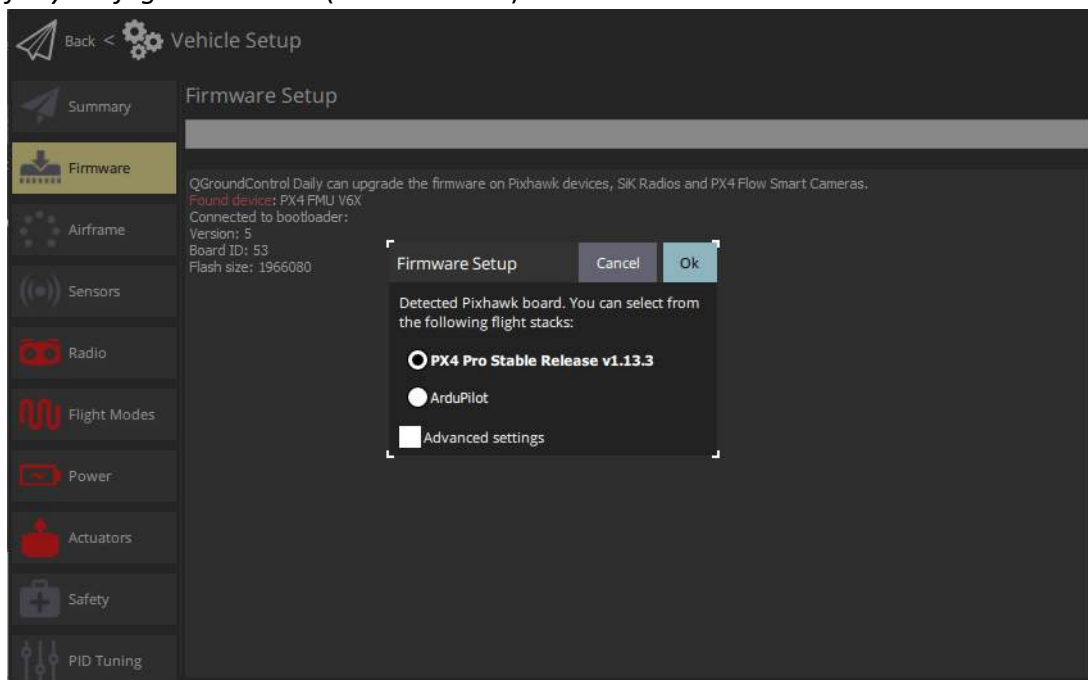
Download QGroundControl onto a computer you will use to connect to your PixRacer Pro to download the firmware: https://docs.qgroundcontrol.com/master/en/qgc-user-guide/getting_started/download_and_install.html

1. Start *QGroundControl* and connect the vehicle

2. Select "Q" icon > **Vehicle Setup** > **Firmware** (sidebar) to open *Firmware Setup*

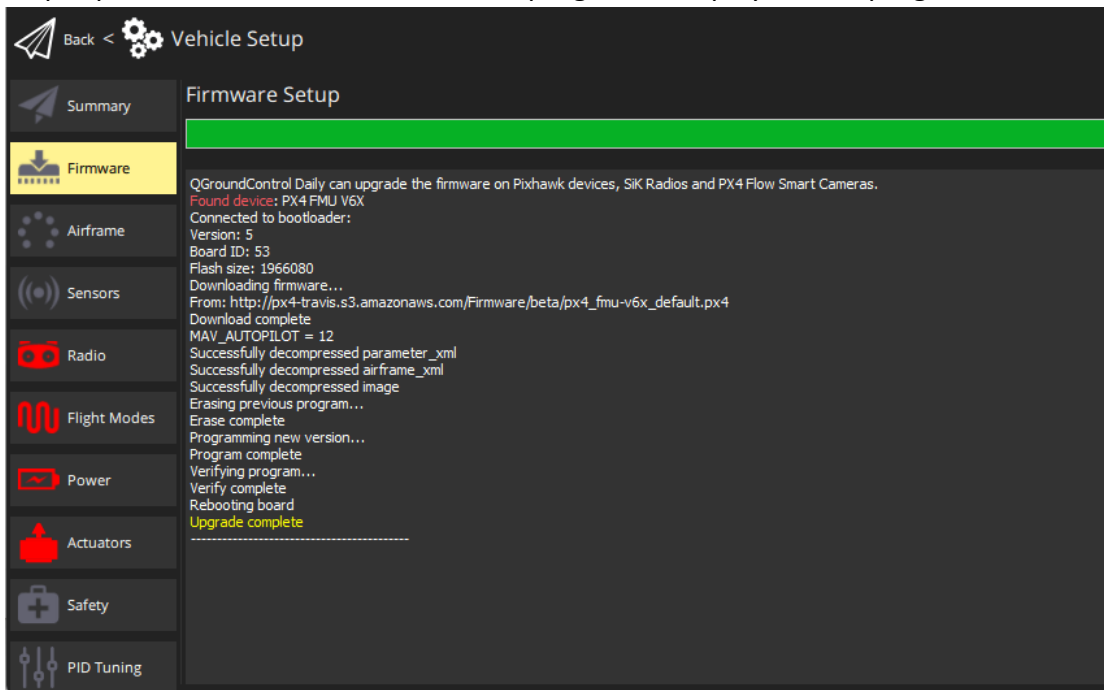


3. Connect the flight controller directly to your computer via USB
4. Select the **PX4 Pro Stable Release vX.x.x** option to install the latest stable version of PX4 for your flight controller (autodetected)



5. Click the **OK** button to start the update. The firmware will then proceed through a number of upgrade steps (downloading new firmware, erasing old firmware etc.). Each

step is printed to the screen and overall progress is displayed on a progress bar.



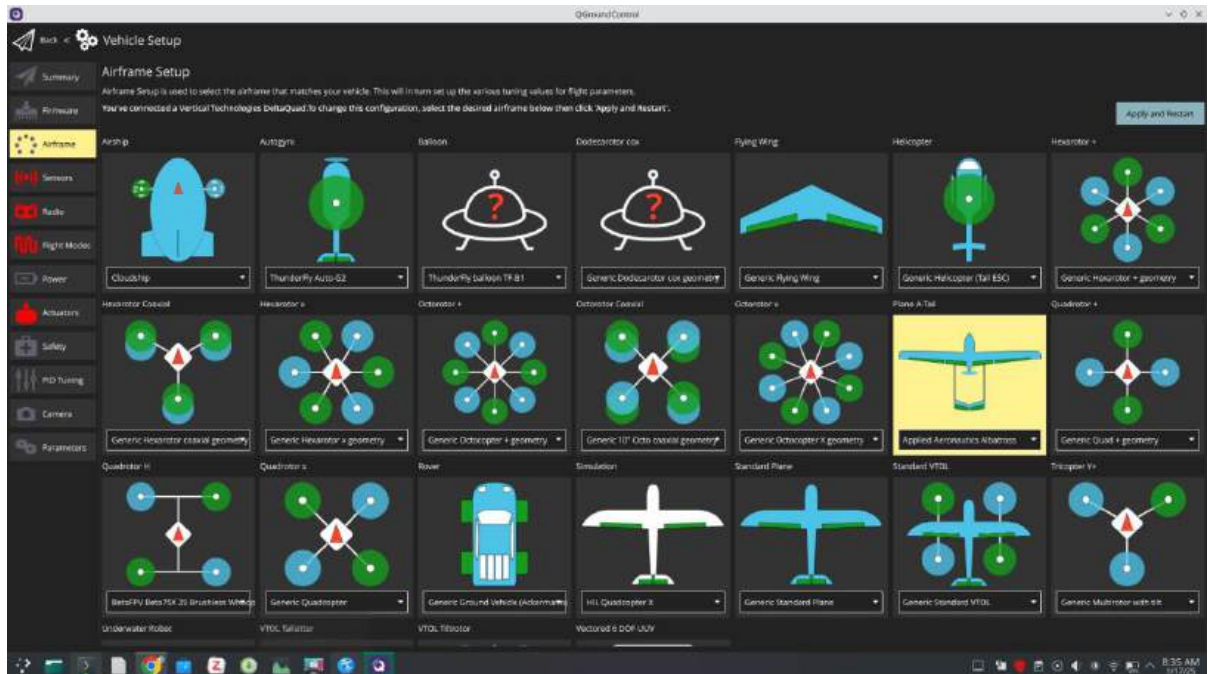
6. Once the firmware has completed loading, the device/vehicle will reboot and reconnect.

After installing the firmware you need to select a vehicle type and frame configuration. The vehicle type can be found here:

https://docs.px4.io/main/en/airframes/airframe_reference.html

Once you determine the vehicle type this can be set in Qgroundcontrol.

1. Select "Q" icon > Vehicle Setup > Airframe (sidebar) to open Airframe Setup.
2. Select the broad vehicle group/type that matches your airframe and then use the dropdown within the group to choose the airframe that best matches your vehicle.



The example above shows *Applied Aeronautics Albatross* which is what we used for our High-Altitude drone.

3. Click Apply and Restart. Click Apply in the following prompt to save the settings and restart the vehicle.

The PX4 firmware then needs to be configured to use ROS 2 instead of MAVLINK. To update parameters, follow this guide: https://docs.px4.io/main/en/advanced_config/parameters.html. The following parameters need to be changed:

```
MAV_1_CONFIG = 0 (Disabled)
UXRCE_DDS_CFG = 102 (TELEM2)
SER_TEL2_BAUD = 921600
```

MAV_1_CONFIG=0 and UXRCE_DDS_CFG=102 disable MAVLink on TELEM2 and enable the uXRCE-DDS client on TELEM2, respectively. The SER_TEL2_BAUD rate sets the comms link data rate. Check that the uxrce_dds_client module is now running. You can do this by running the following command in the QGroundControl MAVLink Console:

- uxrce_dds_client status

If the client module is not running you can start it manually in the MAVLink console:

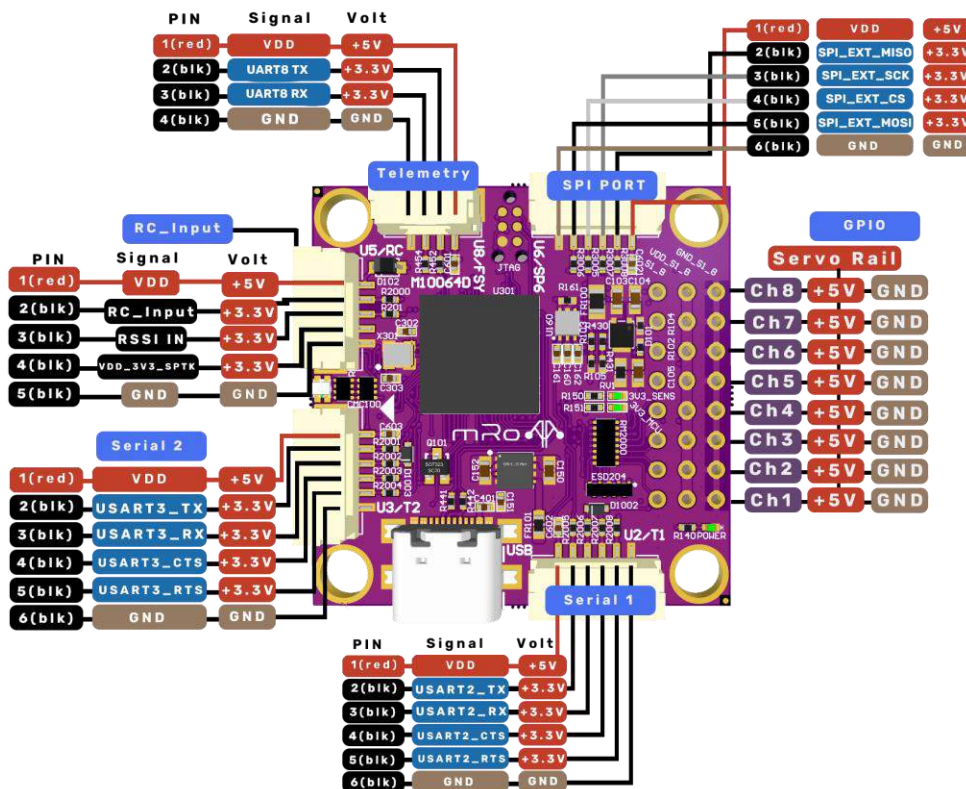
- uxrce_dds_client start -t serial -d /dev/ttyAMA0 -b 921600

RPi and PixRacer Pro Wiring

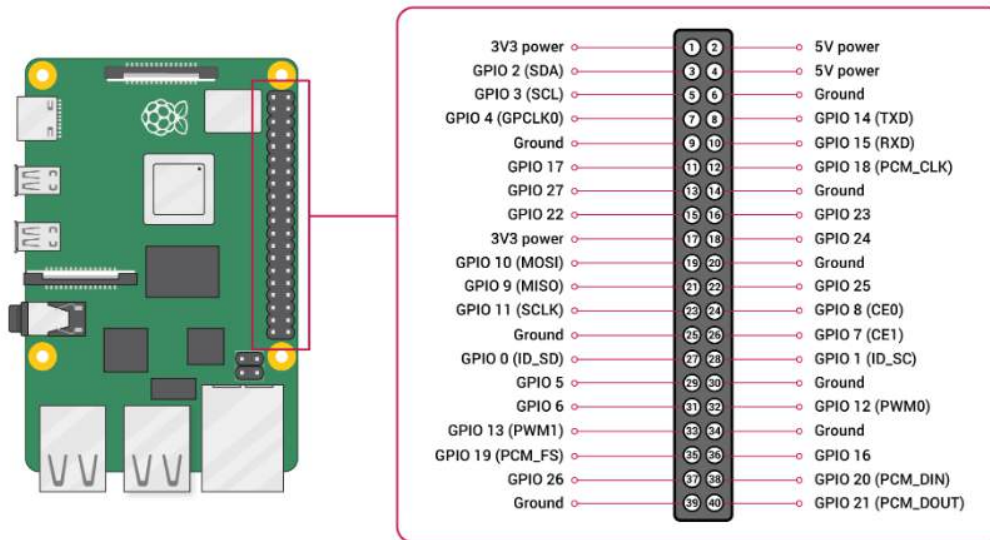
Wire up the serial connection between the RPi and PX4 that is to be used for offboard control. Connect the PixRacer Pro TELEM2 TX/RX/GND pins to the complementary RXD/TXD/Ground pins on the RPi GPIO board:

PX4 TELEM2 Pin	RPi GPIO Pin
UART5_TX (2)	RXD (GPIO 15 - pin 10)
UART5_RX (3)	TXD (GPIO 14 - pin 8)
GND (6)	Ground (pin 6)

The following diagram shows the top view of the PixRacer pro, the TELEM2 port pins are the “Serial 2” pins.



This diagram shows the port pins for the RPi.



uXRCE_DDS Agent Installation

On the RPi, install the Micro XRCE-DDS Agent to communicate with the PixRacer Pro.

1. Install dependencies:
 - a. `sudo apt install -y cmake gcc g++ git libssl-dev`
2. Clone and build Micro-XRCE-DDS-Agent:
 - a. `cd ~`
 - b. `git clone https://github.com/eProsima/Micro-XRCE-DDS-Agent.git`
 - c. `cd Micro-XRCE-DDS-Agent`
 - d. `mkdir build`
 - e. `cd build`
 - f. `cmake ..`
 - g. `make -j4`
 - h. `sudo make install -j4`
 - i. `sudo ldconfig /usr/local/lib/`
 - j. `cd ../..`

uXRCE_DDS Agent Activation

Start the uXRCE_DDS agent in the RPi terminal. To run the Micro-XRCE-DDS Agent, use the following command on the RPi:

- `sudo ~/Micro-XRCE-DDS-Agent/build/MicroXRCEAgent serial --dev /dev/ttyAMA0 -b 921600`

Both the agent and client should be running, and you should see activity on both the MAVLink console and the RPi terminal. You can view the available topics using the following command on the RPi:

- `source /opt/ros/jazzy/setup.bash`
- `ros2 topic list`

Make sure to rebuild your workspace after making any changes by doing the following:

- `cd ~/ros2_ws`
- `colcon build`
- `source ~/.bashrc`

To automatically start the MicroXRCE Agent on Ubuntu 24.04 startup, you can create a systemd service.

1. Create a new systemd service file:
 - a. `sudo nano /etc/systemd/system/microxrce-agent.service`
2. Add the following content to the file:
 - a. NOTE: PLEASE MAKE SURE TO CHANGE THE USERNAME IN THE FILE PATH TO YOUR USER USERNAME, OR IT WILL NOT WORK
[Unit]
Description=MicroXRCE Agent
After=network.target

[Service]
ExecStart=/home/my_username/Micro-XRCE-DDS-Agent/build/MicroXRCEAgent
serial --dev /dev/ttyAMA0 -b 921600
Restart=always
User=root

[Install]
WantedBy=multi-user.target
3. Reload systemd to recognize the new service:
 - a. `sudo systemctl daemon-reload`
4. Enable the service to start on boot:
 - a. `sudo systemctl enable microxrce-agent.service`
5. Start the service:
 - a. `sudo systemctl start microxrce-agent.service`
6. Check the status with:
 - a. `sudo systemctl status microxrce-agent.service`
7. Reboot the system:
 - a. `sudo reboot`
 - b. `ros2 topic list`

- c. `ros2 topic echo /fmu/out/vehicle_status`

This setup ensures that the agent runs with root privileges, which may be necessary for accessing the serial device. This should allow the MicroXRCE Agent to automatically start on system boot.

SSH setup with GUI to view Images/Graphs

On Raspberry Pi:

- Install X11 Forwarding Tools
`sudo apt update && sudo apt upgrade -y`
`sudo apt install x11-apps xauth xorg`
- Edit the SSH Configuration File
`sudo nano /etc/ssh/sshd_config`
 - Ensure the following lines are set and/or uncommented in the file
`X11Forwarding yes`
`X11DisplayOffset 10`
`X11UseLocalhost yes`
- Restart the SSH Service
`Sudo systemctl restart ssh`

On Windows Laptop:

- Install an X server, such as Xming or VcXsrv.
 - Configure the X server to allow connections and start it before using PuTTY.
- **Open PuTTY**
 - Start PuTTY and enter the Ubuntu machine's hostname or IP address in the **Host Name (or IP address)** field.
- **Enable X11 Forwarding**
 - In the left-hand menu, navigate to **Connection > SSH > X11**.
 - Check the box labeled **Enable X11 forwarding**.
 - Enter `localhost:0` in the **X display location** field (this is usually the default).
- **Save the Session**
 - Go back to the **Session** category, give the session a name, and click **Save** to save the settings.

Servo Control Using RPi

Install the following on the RPi:

- `sudo apt install python3-gpiozero`

- `sudo apt-get install python3-rpi.gpio`

Attach the Servo PWM pin to GPIO 25 (pin 22 on the RPi). The Servo needs to be powered by an external 5V power source because it draws too much current from the RPi. The RPi and the external power source do need to share the same ground, however. Once the Servo has the correct wiring, run the following in a python script:

```
from gpiozero import Servo
from time import sleep

servo = Servo(25)

val = -1

try:
    while True:
        servo.value = val
        sleep(0.1)
        val = val + 0.1
        if val > 1:
            val = -1
except KeyboardInterrupt:
    print("Program stopped")
```

Appendix

Useful links divided by section

Flash SD Card with Ubuntu 24.04

- <https://ubuntu.com/tutorials/how-to-install-ubuntu-desktop-on-raspberry-pi-4#2-prepare-the-sd-card>

Boot Ubuntu Desktop and Set Up SSH

- <https://averagelinuxuser.com/how-to-install-and-use-ssh-on-linux/>

Enable UART0 on RPi

- <https://www.raspberrypi.com/documentation/computers/configuration.html#cm1-cm3-cm3-and-cm4>

ROS 2 “Jazzy” Installation & ROS 2 Workspace Setup

- ROS 2 installation and workspace setup:
<https://docs.ros.org/en/jazzy/Installation/Ubuntu-Install-Debs.html#install-ros-2>

→ ROS 2 examples: <https://github.com/ros2/examples>

PixRacer Pro Setup

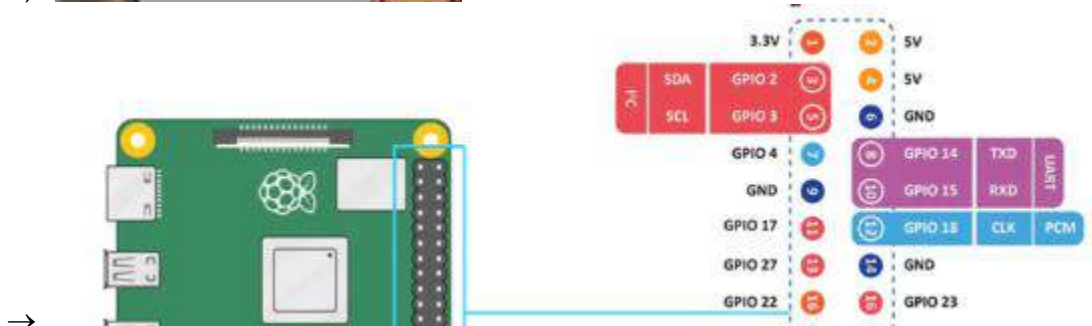
- Qgroundcontrol firmware installation:
<https://docs.px4.io/main/en/config/firmware.html#install-stable-px4>
- Qgroundcontrol vehicle selection:
<https://docs.px4.io/main/en/config/airframe.html>
- Airframe type reference:
https://docs.px4.io/main/en/airframes/airframe_reference.html
- How to change parameters in Qgroundcontrol guide:
https://docs.px4.io/main/en/advanced_config/parameters.html
- TELEM 2 parameter config:
https://docs.px4.io/main/en/companion_computer/pixhawk_rpi.html#ros-2-and-uxrce-dds

uXRCE_DDS Agent Installation & uXRCE_DDS Agent Activation

- https://docs.px4.io/main/en/companion_computer/pixhawk_rpi.html#ros-2-and-uxrce-dds

RPi and PixRacer Pro Wiring

- RPi and PixRacer pro wiring:
https://docs.px4.io/main/en/companion_computer/pixhawk_rpi.html#wiring
- PixRacer Pro hardware layout: <https://docs.3dr.com/autopilots/pixracer-pro/#downloads>




Extra links

ROS 2 “Jazzy” Integration with PX4

- ROS 2 Integration with PX4: <https://docs.px4.io/main/en/ros2/>
- uXRCE-DDS (PX4-ROS 2/DDS Bridge): https://docs.px4.io/main/en/middleware/uxrce_dds.html
- uORB Message Reference:
- https://docs.px4.io/main/en/msg_docs/
- ROS 2 User Guide (with PX4): https://docs.px4.io/main/en/ros2/user_guide.html
- PX4-Autopilot/IntegrationTests: https://github.com/PX4/PX4-Autopilot/tree/main/integrationtests/python_src/px4_it/mavros

PX4

- PX4 Autopilot User Guide: <https://docs.px4.io/main/en/>
- MAVLink Messaging: <https://docs.px4.io/main/en/middleware/mavlink.html#mavlink-overview>
- UAV Data Transmission and Protocols PowerPoint: <https://robo-labor.ee/img/cms/projektid/UAV%20Data%20Transmission%20and%20Communication%20Protocols.pdf>
- General PixRacer Documentation: https://bkueng.gitbooks.io/px4-user-guide/content/en/flight_controller/pixracer.html

Artifact ID: GD-002	Artifact Title: QGroundControl Guide	
Revision: 1	Revision Date: 2025-03-05	
Prepared by: Jadyn Christensen	Checked by:	
Purpose: Instructions for using QGroundControl with a Pixracer Pro on both the High Altitude and Low Altitude Drone		

Revision History			
Revision	Revised by	Checked by	Date
01	Jadyn Christensen		2025-03-05

QGroundControl

Initial Setup and Usage Guide

Author: Team 56 “Snowflake”

Date: 2025-03-05

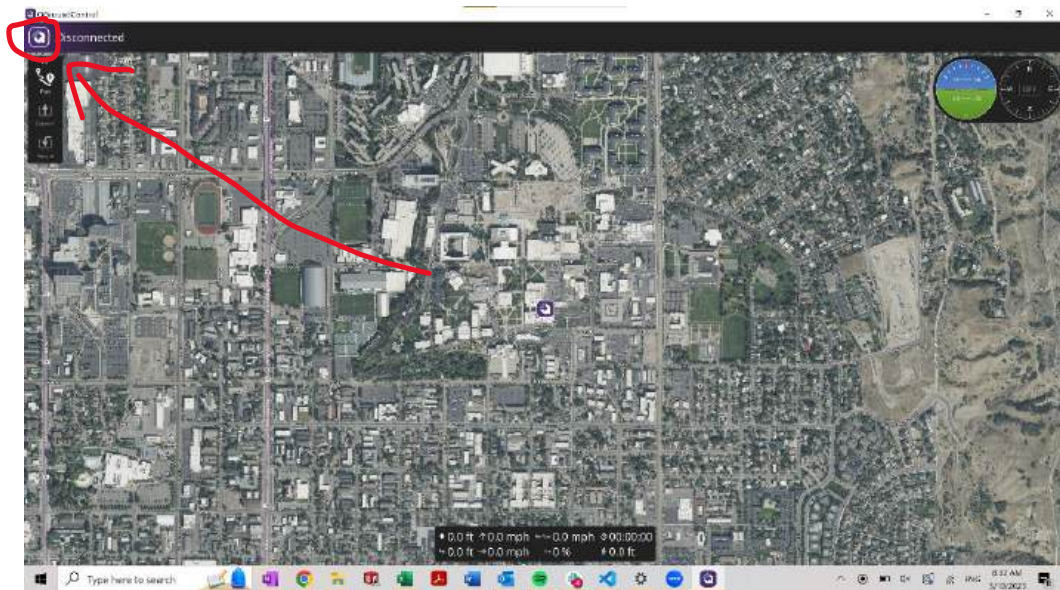
Table of Contents

QGroundControl	2
Flash Pixracer Pro.....	4
Choose the vehicle type.....	6
Binding radio controller to onboard receiver	7
Output Channels	7
Sensor Calibrations	7
Flight Modes	7
Safety Parameters.....	7
Other Parameters that may become useful	7
Configure firmware to use ROS 2	7
Appendix	8

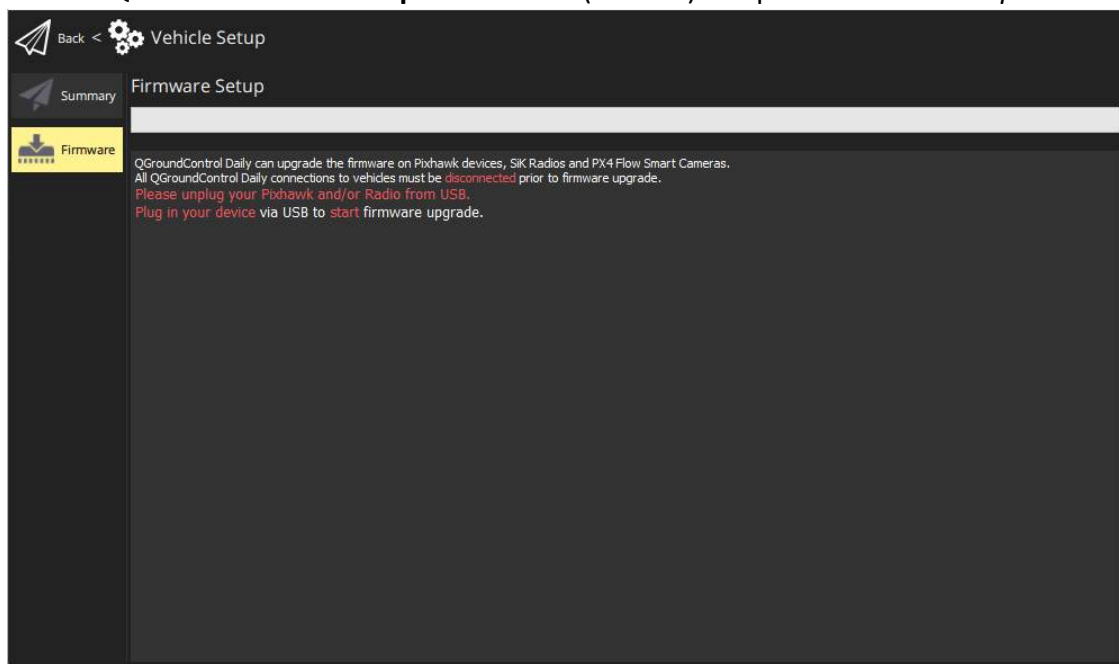
Flash Pixracer Pro

Download QGroundControl onto a computer you will use to connect to your PixRacer Pro to download the firmware: https://docs.qgroundcontrol.com/master/en/qgc-user-guide/getting_started/download_and_install.html

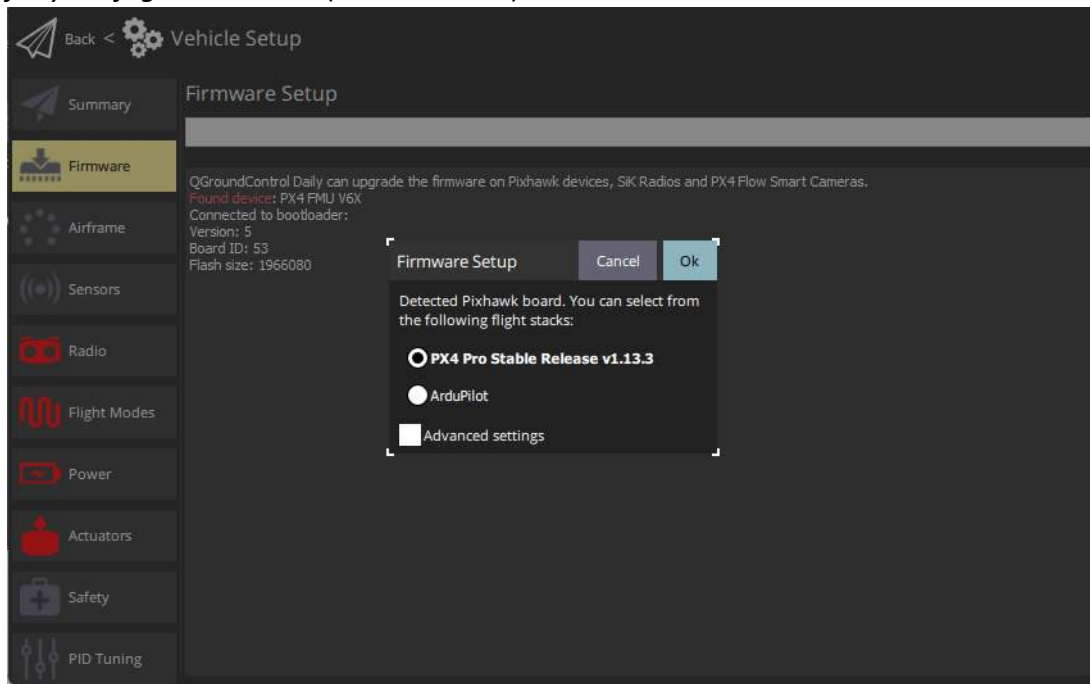
1. Start *QGroundControl*
2. The splash screen appears as below. To access the options menu, click the Q icon in the top left corner of the screen.



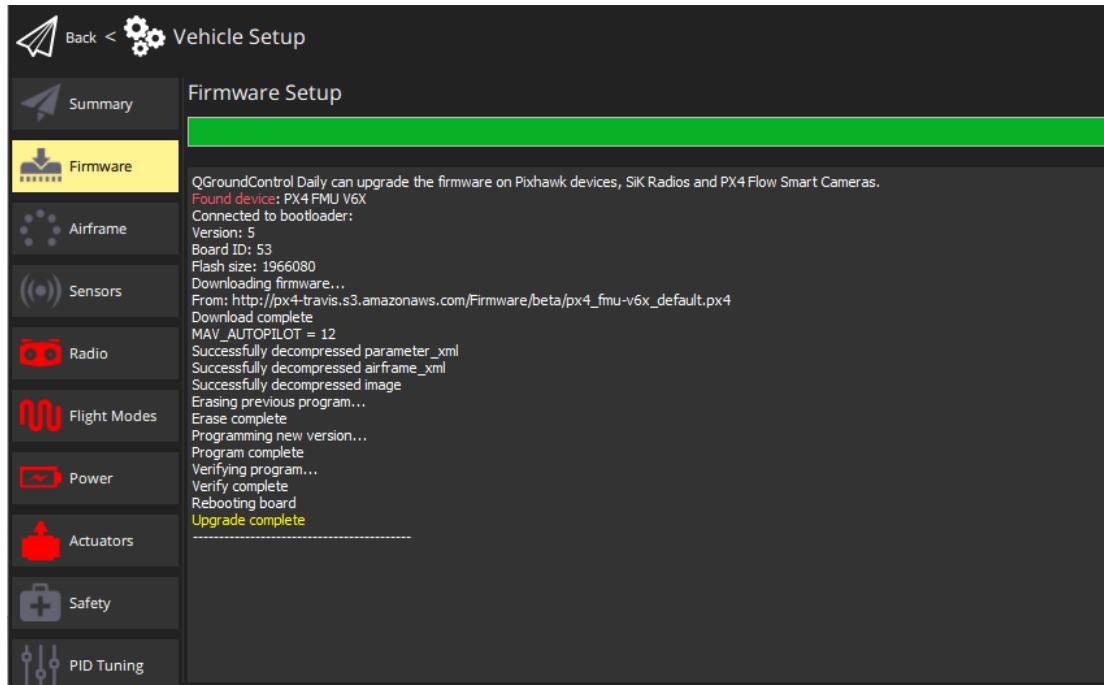
3. Select "Q" icon > **Vehicle Setup** > **Firmware** (sidebar) to open *Firmware Setup*



4. Connect the flight controller directly to your computer via USB
5. Select the **PX4 Pro Stable Release vX.x.x** option to install the latest stable version of PX4 *for your flight controller* (autodetected)



6. Click the **OK** button to start the update. The firmware will then proceed through a number of upgrade steps (downloading new firmware, erasing old firmware etc.). Each step is printed to the screen and overall progress is displayed on a progress bar.



7. Once the firmware has completed loading, the device/vehicle will reboot and reconnect.

Choose the vehicle type

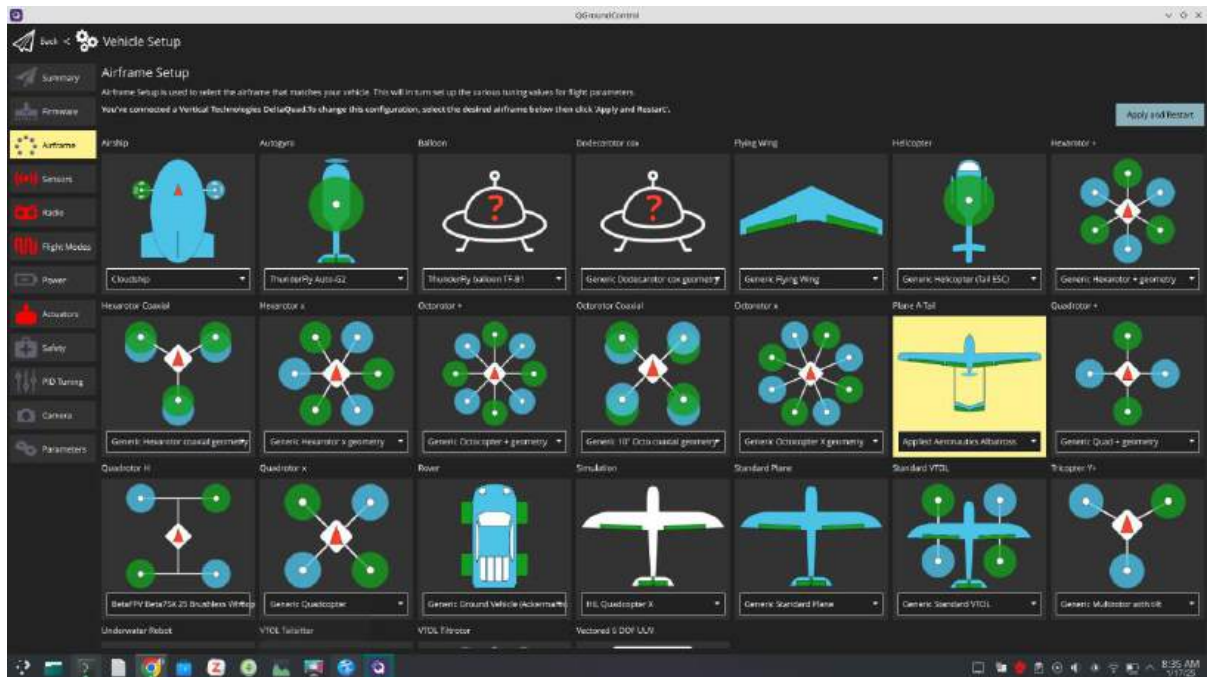
Now that the flight controller is flashed, you can connect to the drone either through USB-C connection or wirelessly through a ground radio as long as the radio is connected. See ART-001 and ART-009 for wiring diagrams.

You now need to select a vehicle type and frame configuration. The different options can be found here: https://docs.px4.io/main/en/airframes/airframe_reference.html

For this project, we used the “Applied Aeronautics Albatross” for the high-altitude drone because it matches the a-frame tail design of the Anaconda. We used “Generic Standard VTOL” for the low altitude drone.

The steps to set this up are below

1. Select "Q" icon > Vehicle Setup > Airframe (sidebar) to open Airframe Setup.
2. Select the broad vehicle group/type that matches your airframe and then use the dropdown within the group to choose the airframe that best matches your vehicle.



The example above shows *Applied Aeronautics Albatross* selected

3. Click Apply and Restart. Click Apply in the following prompt to save the settings and restart the vehicle.

Sensor Calibrations

On the sensor tab in Qgroundcontrol prior to each test flight ensure that you precede through the accelerometer, compass, level horizon, and gyroscope. **We suggest that these be performed before every flight.**

1. Follow the instructions on each test. They are very clear and self explanatory.

It's also important to do an ESC calibration. That is found in the power tab, not the sensor tab.

1. Unplug battery, connect to flight controller via USB.
2. Hit start test.
3. Plug in battery.

This test tends to be extremely sensitive to how quickly one connects the battery. We suggest that there be 2 people, one to hit "start test", and a second to immediately connect a battery.

Binding radio controller to onboard receiver

We followed this guide here to bind the receiver and the radio: <https://www.expresslrs.org/quick-start/binding/>

Once it is bound, ensure that the drone is in communication with qgroundcontrol by going to the radio tab. You should be able to see inputs making a difference in the levels.

Output Channels

On the model tab, you'll see that there are many options. Here, each output channel (i.e. left aileron, pusher motor, etc.) need to be associated with an input command. This takes a little bit of a guess and check.

1. Figure out which input channels are associated with which movement of the radio. Pushing the left joystick left and right is one channel, up down is another, and the same for the right joystick. Each switch on the radio is also associated with one.
2. Assign each input channel to an output. Our outputs are shown below.
3. This is also the section where you can set servo ranges. Find what PWM value is the center of each servo travel, and set that as the center of the servo range.

Configure firmware to use ROS 2

The PX4 firmware now needs to be configured to use ROS 2 instead of MAVLINK. To learn how to find and update parameters, see this guide:

https://docs.px4.io/main/en/advanced_config/parameters.html.

1. To configure the ROS2 connection, set the following configurations to their respective settings.

```
MAV_1_CONFIG = 0 (Disabled)
UXRCE_DDS_CFG = 102 (TELEM2)
SER_TEL2_BAUD = 921600
```

2. MAV_1_CONFIG=0 disables MAVLink on TELEM2 and UXRCE_DDS_CFG=102 enables the uXRCE-DDS client on TELEM2. SER_TEL2_BAUD rate sets the comms link data rate.
3. Check that the uxrce_dds_client module is now running. You can do this by running the following command in the QGroundControl MAVLink Console
`uxrce_dds_client status`
4. If the client module is not running you can start it manually in the MAVLink console:
`uxrce_dds_client start -t serial -d /dev/ttyAMA0 -b 921600`

Appendix

Useful links divided by section

Flash SD Card with Ubuntu 24.04

- <https://ubuntu.com/tutorials/how-to-install-ubuntu-desktop-on-raspberry-pi-4#2-prepare-the-sd-card>

Boot Ubuntu Desktop and Set Up SSH

- <https://averagelinuxuser.com/how-to-install-and-use-ssh-on-linux/>

Enable UART0 on RPi

- <https://www.raspberrypi.com/documentation/computers/configuration.html#cm1-cm3-cm3-and-cm4>

ROS 2 “Jazzy” Installation & ROS 2 Workspace Setup

- ROS 2 installation and workspace setup:
<https://docs.ros.org/en/jazzy/Installation/Ubuntu-Install-Debs.html#install-ros-2>
- ROS 2 examples: <https://github.com/ros2/examples>

PixRacer Pro Setup

- Qgroundcontrol firmware installation:
<https://docs.px4.io/main/en/config/firmware.html#install-stable-px4>
- Qgroundcontrol vehicle selection:
<https://docs.px4.io/main/en/config/airframe.html>
- Airframe type reference:
https://docs.px4.io/main/en/airframes/airframe_reference.html

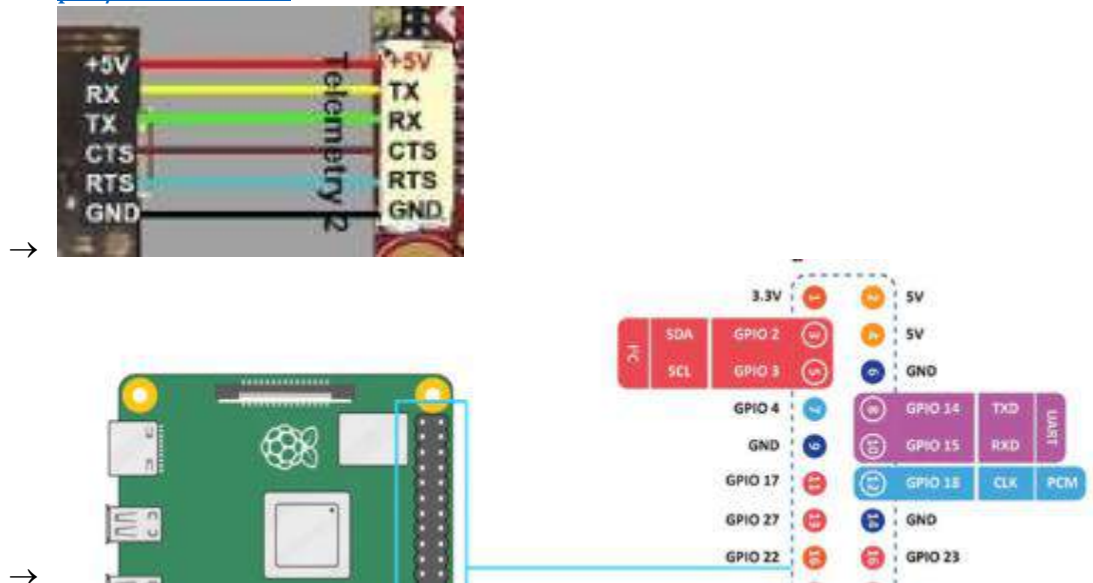
- How to change parameters in Qgroundcontrol guide:
https://docs.px4.io/main/en/advanced_config/parameters.html
- TELEM 2 parameter config:
https://docs.px4.io/main/en/companion_computer/pixhawk_rpi.html#ros-2-and-uxrce-dds

uXRCE_DDS Agent Installation & uXRCE_DDS Agent Activation

- https://docs.px4.io/main/en/companion_computer/pixhawk_rpi.html#ros-2-and-uxrce-dds

RPi and PixRacer Pro Wiring

- RPi and PixRacer pro wiring:
https://docs.px4.io/main/en/companion_computer/pixhawk_rpi.html#wiring
- PixRacer Pro hardware layout: <https://docs.3dr.com/autopilots/pixracer-pro/#downloads>



Extra links

ROS 2 "Jazzy" Integration with PX4

- ROS 2 Integration with PX4: <https://docs.px4.io/main/en/ros2/>
- uXRCE-DDS (PX4-ROS 2/DDS Bridge):
https://docs.px4.io/main/en/middleware/uxrce_dds.html
- uORB Message Reference:
https://docs.px4.io/main/en/msg_docs/
- ROS 2 User Guide (with PX4): https://docs.px4.io/main/en/ros2/user_guide.html

- PX4-Autopilot/IntegrationTests: https://github.com/PX4/PX4-Autopilot/tree/main/integrationtests/python_src/px4_it/mavros


PX4

- PX4 Autopilot User Guide: <https://docs.px4.io/main/en/>
- MAVLink Messaging: <https://docs.px4.io/main/en/middleware/mavlink.html#mavlink-overview>
- UAV Data Transmission and Protocols PowerPoint: <https://robo-labor.ee/img/cms/projektid/UAV%20Data%20Transmission%20and%20Communication%20Protocols.pdf>
- General PixRacer Documentation: https://bkueng.gitbooks.io/px4-user-guide/content/en/flight_controller/pixracer.html

Troubleshooting

Occasionally (haha maybe more than that) the Pixhawk enters strange states. In this case, we suggest trying to reflash the firmware.


The MAGICC lab was an incredibly helpful resource for setting up drones.

Artifact ID: REQ-001	Artifact Title: Gantt Chart	
Revision: 01	Revision Date: 2024-10-02	
Prepared by: Anthony Cardenas/Israel Zenteno	Checked by: Israel Zenteno	
Purpose: A visual document for planning, scheduling, and tracking project tasks and timelines. It is important to note that daily changes and updates are made apart from the major revisions listed below.		

Revision History			
Revision	Revised by	Checked by	Date
01	Anthony Cardenas	Israel Zenteno	2024-10-02
XX	There are changes almost daily, so no changes will be tracked any further.		

Phase 6		Finalize Project					
Task 5.4	Flight Test 3	14-Feb-25	10	28-Feb-25	100%		
	Successful VTOL Flight	14-Feb-25	10	7-Mar-25	25%	All	
	Fully Validate Detection	14-Feb-25	10	7-Mar-25	50%	All	
	Drop mechanism complete	31-Jan-25	10	7-Mar-25	100%	Airframe	
Task 5.5	Flight Test 4	14-Feb-25	10	7-Apr-25	100%		

Complete fulfillment of requirements individually		14-Feb-25	10	7-Apr-25	100%	All
Task 6.1	Refine, Optimize and test	2-Mar-25	2	3-Mar-25	90%	
Task 6.2	Final Design review with Pod instructor	4-Mar-25	2	5-Mar-25	0%	
Task 6.3	Final Design Package DRAFT (file required)	6-Mar-25	2	9-Mar-25	100%	
Task 6.4	Presentation Slides (file required)	10-Mar-25	2	11-Mar-25	100%	
Task 6.5	Executive Summary and artifacts (file required)	12-Mar-25	2	13-Mar-25	100%	
Task 6.6	Final project review	14-Mar-25	2	17-Mar-25	0%	
Task 6.7	Design Fair Dry-Run with Pod Instructor	18-Mar-25	2	19-Mar-25	0%	
Task 6.8	Design Fair	20-Mar-25	2	23-Mar-25	0%	
Task 6.9	Finalize design package (file required)	24-Mar-25	2	25-Mar-25	0%	
Task 6.10	Final presentation with Sponsor	26-Mar-25	2	27-Mar-25	0%	
Task 6.11	Deliver project to sponsor	28-Mar-25	2	31-Mar-25	0%	
Task 6.12	Check out clearance	1-Apr-25	2	2-Apr-25	0%	

Artifact ID: REQ-002	Artifact Title: Bill of Materials	
Revision: 05	Revision Date: February 2, 2025	
Prepared by: Jonah Lowther	Checked by: Blake Folsom	
Purpose: A list identifying all materials and components required to construct two autonomous drones. It is important to note that daily changes and updates are made apart from the major revisions listed below.		

Revision History			
Revision	Revised by	Checked by	Date
01	Jonah Lowther	Blake Folsom	September 19, 2024
02	Blake Folsom	Israel Zenteno	September 25, 2024
03	Joshua Crookston	Jacob Wilkins	November 14, 2024
04	Joshua Crookston	Jacob Wilkins	December 2, 2024
05	Joshua Crookston	Israel Zenteno	February 2, 2025

Part ID	Part Name	Description	Qty	Revision	Vendor	Part #	Cost	Completion
High-Altitude Drone Only								
S-001	RMRC Anaconda – KIT	Basic High-Altitude Drone Airframe	1	1	Ready Made RC	3239	\$249.99	100.00%
S-002	EMAX GT2826-860KV Brushless Motor	High-Altitude Component	1	1	Ready Made RC	83364	\$34.99	100.00%
S-003	EMAX ES9258 27g Metal Gear Digital Servo	High-Altitude Component	3	1	Ready Made RC	83623	\$50.97	100.00%
S-005	EMAX ES3054 17g Metal Gear Digital Servo	High-Altitude Component	4	1	Ready Made RC	83624	\$39.96	100.00%
S-006	RMRC Anaconda - Flap Hardware Kit	High-Altitude Component	1	1	Ready Made RC	3455	\$4.99	100.00%
S-007	APC 13x6.5EP Pusher Prop	High-Altitude Component	1	1	Ready Made RC	945	\$5.73	100.00%
S-014	FLYFUN 60A V5 ESC (3S-6S)	Electronic Speed Controller	1	2	Hobby Wing	30214101	\$46.99	100.00%
S-022	Digital Airspeed Sensor Kit Differential PITOT for PX4 Pixhawk Autopilot Flight	Airspeed Sensor	1	3	Amazon.com	B01J7NCML0	\$48.89	100.00%
S-010	XT60 Parallel Adapter	Power Adapter	4	2	Ready Made RC	82594	\$11.16	100.00%
S-004	Zeee 14.8V 4S Lipo Battery 50C 3300mAh Soft Case Battery with XT60 Plug for RC Airplane Helicopter RC Boat UAV Drone FPV RC Car Truck Boat(2 Packs)	High-Altitude Component	1	1	Amazon.com	Not Listed	\$62.09	100.00%
S-029	SanDisk 32GB MAX ENDURANCE UHS-I microSDHC Memory Card with SD Adapter	Another high quality SD card	1	2	B & H Photo Video	Not Listed	\$12.99	100.00%
Low-Altitude Drone Only								
S-025	Makeflyeasy Hero Drone 2180mm UAV VTOL - Type: PNP	Low-Altitude Drone Framework with VTOL capabilities	1	3	UAV Model	Not Listed	\$1,499.00	100.00%
S-031	Lumenier 22000mAh 6s 20c Lipo Battery	Larger Lipo battery for Low-altitude drone	1	1	GetFPV.com	2445	\$299.99	100.00%
High-Altitude Drone & Low-Altitude Drone								
Power Distribution								
S-018	High current PWR	Power Distribution Board	2	3	3D Robotics	M10121	\$259.80	100.00%
GPS								
S-008	Lumenier SAM-M10Q GPS Module - NDAA	GPS Module	2	2	GetFPV.com	22281	\$159.98	100.00%
Flight Controllers								
S-013	PixRacer Pro	Flight Controllers	2	2	3D Robotics	M10064D	\$699.80	100.00%
S-032	SanDisk 32GB (Pack of 2) Ultra microSDHC UHS-I Memory Card (2x32GB) with Adapter - SDSQUA4-032G-GN6MT [New Version]	Backup SD cards for Pixracer Pros	1	1	Amazon.com	Not Listed	\$13.49	100.00%
Flight Companion Computers								
S-015	Raspberry Pi 5 - 8 GB RAM	On-board Computers	2	2	Adafruit	5813	\$160.00	100.00%
S-019	27W USB-C Power Supply Compatible with Raspberry Pi 5, PD Adapter 5.1V 5A Power Supply Type-C Power Adapter for Raspberry Pi 5 Model B 8GB/4GB (White)	Plug-in power supply for Raspberry Pi 5	1	3	Amazon.com	B0D28KNT7H	\$9.99	100.00%
S-028	SanDisk 64GB MAX ENDURANCE UHS-I microSDXC Memory Card with SD Adapter	Upgraded SD Cards for the Rasperry Pi 5's so even with inconsistent power, the SD cards will not get corrupted	2	3	B & H Photo Video	#SASQQVR06461 • MFR #SDSQQVR-064G-AN61A	\$31.98	100.00%
Radio Remote Control of Drone								
S-009	RadioMaster Zorro Radio Controller - 4-in-1 Multi-Protocol	Drone RC Controllers	2	2	GetFPV.com	17658	\$233.98	100.00%
S-024	900mah 3.7v Li-ion 18350 Battery for Zorro Radio Controller (2pcs)	Batterys for Drone RC Controllers	2	3	Radio Master	18350	\$19.98	100.00%
S-011	R81 V2 Receiver - FCC Region	Radio Reciever for Flight Controller	2	2	NewBeeDrone	Not Listed	\$15.98	100.00%

Building and Testing

S-023	Woodland Scenics FBA_WOOST1444 Glue, White	Glue to construct airframe	1	3	Amazon.com	Not Listed	\$21.99	100.00%
S-020	Standard Wood Supply Firewood Bundle	Build fire to test cameras	2	3	Home Depot	86675500010	\$13.94	100.00%
S-021	Diamond Strike-a-Fire Starter Matches	Start fire to test cameras	1	3	Home Depot	4878911003	\$5.98	100.00%

Ground Control Station Control of Drone (Radio)

S-016	TUOLNK 4PCS SMA Coax Connector Cable Gender Changers,SMA Male/Female to RP-SMA Male/Female Coax Adapter Low Loss Coaxial Connector for Radio Antenna, Audio - 4 pcs	Adapters for SiK Telemetry Radios to use a new antenna (only for Ground Control Stations)	1	3	Amazon.com	SMA Straight Adapter	\$6.87	100.00%
S-017	12dBi 433MHz Antenna High Gain Radio Antenna SMA Male Connector Omni-Directional Antenna Wireless Antenna for Wireless Sensor Network Router Modem Pack of 2	Long-range radio antenna for ground station (to be connected to the SiK Telemetry Radio)	1	3	Amazon.com	Not Listed	\$9.68	100.00%
S-012	SiK Telemetry Radio V3 - 500 mW 433 MHz	Plug-in Radio Modules for Drones and Ground Control Stations	2	2	HolyBro	17014	\$125.98	100.00%

Ground Control Station Control of Drone (Wi-Fi)

S-030	BrosTrend AX1800 WiFi 6 Linux Compatible WiFi Adapter for PC and Raspberry Pi 2+ w/Longer Range WiFi Antenna & AC650 Linux WiFi Adapter USB Wireless Adapter with 5dBi Antenna	Set of long-range usb ONE wifi adapter and ONE receiver for ground control station and raspberry pi 5	1	1	Amazon.com	Not Listed	\$51.99	100.00%
-------	--	---	---	---	------------	------------	---------	---------

Cameras

S-026	S304SP Mosaic Core Starter Kit 320x240, 57HFOV, 9HZ	Two long-wave fire detection cameras	2	1	Seek Thermal	S304SP	\$798.00	100.00%
-------	---	--------------------------------------	---	---	--------------	--------	----------	---------


Total Cost: \$5,007.15

Artifact ID: REQ-003	Artifact Title: Requirements Matrix	
Revision: 06	Revision Date: 2025-03-07	
Prepared by: Israel Zenteno		Checked by: Jacob Wilkins
Purpose: Outline the project requirements set by the sponsor.		



Revision History			
Revision	Revised by	Checked by	Date
01	Blake Folsom	Jacob Wilkins	2024-09-6
02	Jonah Lowther	Jacob Wilkins	2024-09-20
03	Israel Zenteno	Joshua Crookston	2024-09-23
04	Jonah Lowther	Israel Zenteno	2024-09-27
05	Joshua Crookston	Jonah Lowther	2025-02-20
06	Joshua Crookston	Jonah Lowther	2025-03-07

[illegible]

Artifact ID: REQ-004	Artifact Title: Software Bill of Materials	
Revision: 03	Revision Date: 2025-02-11	
Prepared by: Joshua Crookston		Checked by: Jacob Wilkins
Purpose: A list identifying all software components required to construct two autonomous drones according to our requirements. It is important to note that daily changes and updates are made apart from the major revisions listed below.		

Revision History			
Revision	Revised by	Checked by	Date
01	Joshua Crookston	Jacob Wilkins	2024-11-18
02	Joshua Crookston	Israel Zenteno	2025-02-10
03	Israel Zenteno	Joshua Crookston	2025-02-11

Software Bill of Materials (SBOM)

Project Information

Date Created: November 18, 2024

Last Updated: February 10, 2025

1. Component: Raspberry Pi Imager (Deb Package)

- Name: Raspberry Pi Imager
- Version: Latest (as of download date)
- Format: .deb
- Source URL:
https://downloads.raspberrypi.org/imager/imager_latest_amd64.deb
- Dependencies:
 - libc6
 - libstdc++6
 - libgcc1
 - Other standard libraries required for Debian-based systems.

2. Component: Raspberry Pi Imager (Windows Executable)

- Name: Raspberry Pi Imager
- Version: Latest (as of download date)
- Format: .exe
- Source URL: https://downloads.raspberrypi.org/imager/imager_latest.exe
- Dependencies:
 - Windows system libraries (e.g., .NET Framework, Visual C++ Redistributables, if required).

3. Component: ROS 2 Jazzy Desktop

- Name: ROS 2 Jazzy Desktop
- Version: Jazzy Release
- Format: Debian Packages (.deb)
- Source URL: [ROS Jazzy Installation](#)
- Dependencies:
 - software-properties-common
 - curl
 - Locale setup (en_US.UTF-8)
 - ROS 2 GPG key and apt repository setup
 - System libraries:
 - libpython3-dev
 - build-essential

4. Component: QGroundControl

- Name: QGroundControl
- Version: Latest (as of download date)
- Formats: Executable for Windows, Linux binaries
- Source URL: [QGroundControl](#)
- Dependencies:
 - For Linux:
 - Qt libraries (qt5-default, qtdeclarative5-dev)
 - GStreamer (gststreamer1.0*)
 - Multimedia libraries.
 - For Windows:
 - Bundled dependencies in the installer.

5. Component: uXRCE-DDS Agent and Client (PX4-ROS2/DDS Bridge)

- Name: uXRCE-DDS Agent and Client
- Version: Latest (as of download date)
- Source URL: [PX4 uXRCE-DDS Documentation](#)
- Dependencies:
 - eProsima Micro XRCE-DDS library
 - Build tools:
 - cmake
 - make
 - System libraries:
 - libfastcdr-dev
 - libfastrtps-dev
 - ROS2 integration dependencies:
 - ROS2 workspace with px4_msgs cloned and built using colcon.

6. Component: Mission Planner

- Name: Mission Planner
- Version: Latest (as of download date)
- Formats: Windows executable, Linux via MONO runtime
- Source URL: [Mission Planner Installation](#)
- Dependencies:
 - For Windows:
 - DirectX
 - .NET Framework
 - Visual C++ Redistributables
 - For Linux:
 - MONO runtime.

7. Component: Preware Remote Pilot by ASA

- Name: Preware Remote Pilot by ASA
- Version: Latest (as of download date)
- Platform: Android application
- Source URL: Available on Google Play Store or official ASA website.
- Dependencies:
 - Android OS compatibility based on app-defined API levels.

8. Component: PX4_msgs

- Name: PX4_msgs (ROS2 Message Definitions for PX4)
- Version: Latest (as of download date)
- Source URL: [PX4 ros com GitHub Repository](#)
- Dependencies:
 - ROS2 workspace setup.
 - Synchronization with PX4 firmware versions for compatibility.

9. Component: PX4_ros_com

- Name: PX4_ros_com (PX4 to ROS2 Bridge)
- Version: Latest (as of download date)
- Source URL: [PX4 ros com GitHub Repository](#)
- Dependencies:
 - Direct dependency on px4_msgs.
 - ROS2 workspace setup.
 - Build tools like colcon.

10. Python Libraries

The following Python libraries are required:

Library Name	Version	Source/Installation Command
numpy	Latest	pip install numpy
pandas	Latest	pip install pandas
scipy	Latest	pip install scipy
rclpy	ROS2-specific	Installed via ROS2 setup instructions

Additional standard Python libraries used include:
os, math, json, time, threading, collections.deque, and others.

Libraries to include in Python Files:

```
import os
import math
import json
import time
import threading
import tkinter as tk
from collections import deque
from datetime import datetime
```

```
import numpy as np
import pandas as pd
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from scipy.ndimage import gaussian_filter
```

```
import rclpy
from rclpy.node import Node
from rclpy.qos import QoSProfile, ReliabilityPolicy
```

```
from sensor_msgs.msg import Image
from std_msgs.msg import String
from px4_msgs.msg import VehicleCommand, VehicleAttitude, SensorGps
```

```
from builtin_interfaces.msg import Time
```

11. C++ Libraries


The following C++ libraries are used in conjunction with ROS2 and PX4:

Library Name	Source/Description
<chrono>	Standard C++ library
<cstdio>, <cstring>	Standard C++ library

<rclcpp/rclcpp.hpp>	ROS2 C++ client library
<sensor_msgs/msg/image.hpp>	ROS2 message type for sensor data
Seek Thermal SDK	Includes headers like <seekcamera.h>

Libraries to include for C++ Files:

```
#include <chrono>
#include <cstdio>
#include <cstring>
#include <rclcpp/rclcpp.hpp>
#include <sensor_msgs/msg/image.hpp>
// Seek Thermal SDK (C includes)
extern "C" {
    #include <seekcamera/seekcamera.h>
    #include <seekcamera/seekcamera_manager.h>
    #include <seekcamera/seekcamera_error.h>
    #include <seekcamera/seekcamera_frame.h>
    #include <seekcamera/seekcamera_version.h>
    #include <seekframe/seekframe.h>
}
```

Artifact ID: TP-001	Artifact Title: Thermal vs. Low-Light Camera Comparison for UAS Fire Detection		
Revision: 02	Revision Date: 2025-03-06		
Prepared by: Anthony Cardenas, Isaac Davies		Checked by: Tristan Mott	
Purpose: The purpose of this test is to compare the performance of a thermal camera and a low-light camera for fire detection in various lighting conditions. The results will determine the most suitable camera for use on our UAS, evaluating its effectiveness at different distances during both daytime and nighttime conditions. Additionally, this test aims to establish light threshold values for fire detection based on filtered footage analysis.			

Revision History			
Revision	Revised by	Checked by	Date
01	Isaac Davies	Tristan Mott	2024-10-22
02	Joshua Crookston	Tristan Mott	2025-03-06

References		
Artifact ID	Revision	Title
ART-006	01	Detection Code Overview

Test Procedure:

Thermal vs. Low-Light Camera Comparison for UAS Fire Detection

1. Test Objective

The purpose of this test is to compare the performance of a thermal camera and a low-light camera for fire detection in various lighting conditions. The results will determine the most suitable camera for use on our UAS, evaluating its effectiveness at different distances during both daytime and nighttime conditions. Additionally, this test aims to establish light threshold values for fire detection based on filtered footage analysis.

2. Test Equipment & Setup

2.1 Required Equipment

- Thermal camera
- Low-light camera
- Stable mounting platform for cameras
- Fire source (controlled, 2 ft (0.6 m) in diameter)
- Fire-starting materials (wood, fire starter)
- Fire suppressant (water, extinguisher)
- Computer with image processing software and filtering code
- Safety equipment for fire handling

2.2 Setup Steps

1. **Verify Camera Functionality:**
 - a. Confirm that both cameras are operational and capable of recording video.
 - b. Ensure proper storage or transmission of recorded footage.
2. **Mounting and Stability:**
 - a. Securely attach cameras to a stable platform.
 - b. Ensure cameras are properly aligned for consistent fire detection testing.

3. Test Execution

3.1 Fire Setup & Observation Distance

1. Safely ignite a controlled fire measuring **2 ft (0.6 m) in diameter** in a designated fire pit.
2. Maintain environmental safety and confirm wind conditions do not interfere with the test.

3.2 Camera Comparison Procedure

1. Position the first camera **100 meters away** from the fire.
 - a. Record footage using the **thermal camera**.
 - b. Switch to the **low-light camera** and record footage.
2. Move to **200 meters away** and repeat the process for both cameras.
3. Move to **400 meters away** and repeat the process for both cameras.
4. Ensure each camera records for a sufficient duration to capture usable data.

3.3 Data Collection & Image Processing

1. Download recorded footage from both cameras.
2. Apply the fire detection filtering code to each video.
3. Analyze detection performance using filtered images:
 - a. **Unfiltered video:** Raw footage from both cameras.
 - b. **Filtered video:** Processed footage using fire detection filters.
 - c. **Auto-scaled video:** Adjusted contrast and brightness for better fire visibility.
4. Evaluate fire visibility at each distance based on filtering results.

4. Test Validation Criteria

Camera	Distance	Fire Detected? (Y/N)	Filtered Visibility (Y/N)
Thermal	100m		
Low-light	100m		
Thermal	200m		
Low-light	200m		
Thermal	400m		
Low-light	400m		

Additional Analysis

- Compare detection accuracy for both cameras before and after filtering.
- Assess the effectiveness of fire detection filtering algorithms.
- Identify the maximum reliable detection distance for each camera.

5. Test Completion & Documentation

1. **Record Observations:** Document test results, including video analysis and detection accuracy.
2. **Log Errors & Issues:** Note any hardware/software failures, latency, or video quality problems.
3. **Save Processed Footage:** Store unfiltered and filtered videos for reference.
4. **Shutdown Procedures:** Safely power down all equipment.

6. Safety Considerations

- Ensure fire safety measures are in place (fire extinguisher, controlled burn area).
- Maintain a safe distance from the fire.
- Follow all capstone safety guidelines.

7. Test Status & Reporting

Upon test completion, document results and submit a test report detailing:

- Camera performance comparisons
- Observed video feed quality
- Detection accuracy at different distances
- Effectiveness of filtering in enhancing fire detection
- Identified issues and recommendations

Artifact ID: TP-002	Artifact Title: Central Flight Test Document
Revision: 01	Revision Date: Nov 6, 2024




Prepared by: Jacob Wilkins	Checked by: Jonah Lowther
Purpose: Contains testing plans for all flights, communications, and ROS 2 nodes as well as test results.	

Revision History			
Revision	Revised by	Checked by	Date
01	Jacob Wilkins	Jonah Lowther	Nov 6, 2024
XX	There are changes almost daily, so no changes will be tracked any further.		

Flight #	Date	Successful tests	Failed tests	Post Flight Notes	Location
1	1/17/2025	None	preflight failed	SD card failed requiring us to take SD card from raspberry pi, aileron servo wasn't attached correctly, and a-tail servo failed randomly.	Elberta, Slant Road

2	1/24/2025	<p>Succesfully got high altitude drone in the air but transmitter lost connection to the drone. However, luckily the untested failsafe worked for loss of connection and the drone correctly came back and circled from where it took off the drone was immediatly landed. A new and better tranmitter is needed.</p>	Provo,Rock Canyon Park
3	2/7/2025	<p>High Altitude drone did not fly due to extreme wind conditions. But the camera effectively detected the fire , Transmitter needed to be on crossfire mode to use newly purchased tranmitter. Once transmitter was on the mode using the new transmitter range from transmitter to drone was accebttable.</p>	Provo,Rock Canyon Park
4	2/18/2025	<p>Low Altitude Drone on VTOL mode fell about 20 feet because the motors cut off due to a failsafe trigered by an error:"Attitude Invalid" seemed like the vehicle estimators were not running</p>	Provo,Rock Canyon Park
5	2/21/2025	<p>High Altitude drone was flown effectively for about a 5 minute flight. The drone succesufuly switched from manual mode to autonomous mode during the flight and succesfully followed the correct waypoint path autonomously. A fire was made and the drone flew by the fire many times to collect fire data. When the drone was being taken down for landing there was traffic on the road and the drone lost battery and got extensive damage with the fuelsulage splitting into two pieces and dammage on the wings, however, electronics were still intact. Unfortuanntly, the drone lossing power wiped the data on the Rasberry Pi and no critical fire data was saved from the thermal camera.</p>	Elberta, Slant Road

6	3/11/202 5	None	After setting up a fire and testing the raspberry pi, camera, and comms systems of each drone, we attempted to connect the controller to the high-altitude drone to fly above the fire to capture video data. Unfortunately, the drone kept losing connection. We then attempted to fly the high-altitude drone and that drone, once armed, would switch into autonomus mode and our only functioning control was the disarm switch. We were unsuccessful in our flight, but we were able to successfully reconnect to the drones without issues the following day. No damage was done to either drone.	Provo, Rock Canyon Park
7	3/21/202 5	None	We successfully tested the autonomous dropping mechanism	Elberta, Slant Road

Artifact ID: TP-003	Artifact Title: Test Procedure, Thermal Camera Validation on Drone with Raspberry Pi 5 Running ROS 2		
Revision: 01	Revision Date: 2025-04-06		
Prepared by: Joshua Crookston		Checked by: Jacob Wilkins	
Purpose: The purpose of this test is to validate the functionality of a thermal camera connected to a Raspberry Pi 5 running ROS 2, ensuring compatibility, power adequacy, and proper video feed visualization. This test does not evaluate any image processing algorithms but focuses solely on the operational aspects of the camera.			

Revision History			
Revision	Revised by	Checked by	Date
01	Joshua Crookston	Jacob Wilkins	2025-04-06

Test Procedure

Thermal Camera Validation on Drone with Raspberry Pi 5 Running ROS 2

1. Test Objective

The purpose of this test is to validate the functionality of a thermal camera connected to a Raspberry Pi 5 running ROS 2, ensuring compatibility, power adequacy, and proper video feed visualization. This test does not evaluate any image processing algorithms but focuses solely on the operational aspects of the camera.

2. Test Equipment & Setup

2.1 Required Equipment

- Thermal camera
- Raspberry Pi 5 with ROS 2 installed
- Power source (ensure sufficient wattage for Raspberry Pi 5 and camera)
- Drone (if applicable) or stable mounting platform
- Computer with ROS 2 and GUI for monitoring
- Fire source (controlled, 2 ft (0.6 m) in diameter)

- Safety equipment for fire handling

2.2 Setup Steps

1. Verify Camera Compatibility:

- Confirm that the thermal camera is supported by ROS 2 and Raspberry Pi 5.
- Install necessary ROS 2 packages and drivers.
- Verify camera recognition in ROS 2 using the following commands:

```
ros2 topic list
```

```
ros2 topic echo /<camera_topic>
```

- Ensure the camera streams data properly in ROS 2.

2. Power Verification:

- Connect the camera to Raspberry Pi 5 and ensure sufficient power supply.
- Monitor voltage and current consumption to confirm stable operation.

3. Mounting and Connectivity:

- Securely attach the camera to the Raspberry Pi and ensure a stable data connection.
- Mount the Raspberry Pi onto the drone or a stationary platform.

3. Test Execution

3.1 Fire Setup & Observation Distance

- Safely ignite a controlled fire measuring **2 ft (0.6 m) in diameter**.
- Move **200 ft (61 m) away** from the fire while maintaining line-of-sight.
- Ensure environmental safety and confirm wind conditions do not interfere with the test.

3.2 Live Video Feed Verification

- Open the ROS 2 GUI to display the live thermal feed.
- Verify that the camera is streaming properly by observing:
 - Unfiltered video** (raw thermal data).
 - Filtered video** (preliminary filtering applied).
 - Auto-scaled video** (adjusted contrast for better visibility).
- Observe the fire in the live feed and confirm visibility.

3.3 Filter Adjustment & Parameter Tuning

Using real-time feed adjustments, modify the following parameters to isolate fire visibility:

Parameter	Description	Adjustment Steps
-----------	-------------	------------------

Grayscale Threshold	Cuts out pixels below a certain grayscale value	Increase until only fire remains visible
Value Addition	Adds a constant value to unfiltered pixels	Adjust to enhance contrast
Gaussian Blur	Applies a blur based on a sigma value	Tune to reduce noise while maintaining fire visibility
Final Threshold	Cuts out pixels below a new grayscale threshold	Fine-tune for optimal fire isolation

- Adjust parameters incrementally while monitoring live feed until only the fire remains visible.

4. Test Validation Criteria

Test Step	Expected Outcome
Camera initialization	Camera is detected by ROS 2 and streams data
Power verification	Camera operates without power failures
Live feed display	Unfiltered, filtered, and auto-scaled video feeds are viewable
Fire detection	Fire is visible at 200 ft (61 m) distance
Parameter tuning	Fire is isolated using filter adjustments

5. Test Completion & Documentation

- Record Observations:** Document test results, including screenshots of video feeds at different filter settings.
- Log Errors & Issues:** Note any hardware/software failures, latency, or video quality problems.
- Save Configuration Settings:** Store optimal parameter values for future reference.
- Shutdown Procedures:** Safely power down the Raspberry Pi and camera.

6. Safety Considerations

- Ensure fire safety measures are in place (fire extinguisher, controlled burn area).
- Maintain a safe distance from the fire.
- Verify drone stability (if applicable) before flight.

7. Test Status & Reporting

Upon test completion, document results and submit a test report detailing:

- Camera functionality status
- Observed video feed quality
- Optimal filter parameters
- Identified issues and recommendations

Artifact ID: TP-004	Artifact Title: Preflight Checklist	
Revision: 01	Revision Date: 2025-03-16	
Prepared by: Lucas Bons		Checked by: Jonah Lowther
Purpose: The checklist of items to bring and procedures to check before an active flight test		



Revision History			
Revision	Revised by	Checked by	Date
01	Lucas Bons	Nina Chao	2024-03-16

Flight Date:

High Altitude

Complete?	Action:	Tools, materials Needed:	Details:	Who can do it?
	Assemble wings	3/32 hex		Anyone
	Ensure connection to Pi	Pi, laptop, extra jumper cables, wiring diagram (on hood)	Insert Pi into drone, insert wifi antenna, connect jumpers from Flight Controller, connect to laptop through wifi	Israel, Anthony, Isaac, Tristan, Janie
	Ensure camera capabilities	Laptop		Anthony, Tristan, Isaac
	Drone battery level verification	Multimeter		Anyone
	Remote battery level verification	Controller		Brian
	Receiver Test	Controller	Power Drone on, test receiver range	JBJ
	Calibrate accelerometer	Laptop - QGroundControl	Stationary dance	JBJ (Jonah, Brian, Jadyne) + Anyone
	Calibrate compass	Laptop - QGroundControl	Spinning dance	JBJ + Anyone
	Calibrate gyro	Laptop - QGroundControl	Set on level ground	JBJ + Anyone
	Calibrate level horizon	Laptop - QGroundControl	Set on level ground	JBJ + Anyone
	Acquire GPS signal	Laptop - QGroundControl	Power up, take laptop and plane into the open and validate gps lock via Q ground control	JBJ + Anyone
	Create flight plan	Laptop - QGroundControl	Create and load flight plan	JBJ

Attach wings to fuselage	Phillips head screwdriver		Anyone
Right aileron servo test	Controller		Brian
Left aileron servo test	Controller		Brian
Right flap servo test	Controller		Brian
Left flap servo test	Controller		Brian
Front wheel servo test	Controller		Brian
Attach rear prop	Rear Prop, Nut, wrench		Lucas, Anna, Jady, Jonah, Brian
Final visual inspection			
Insert lid	Lid		Anyone
Center of gravity test		Pick up drone under wings and ensure balance	JB + Airframe integration
Set up ROS nodes	Laptop	Launch ROS nodes and start collecting data	Anthony, Tristan, Isaac
Start fire	Matches/lighter, wood		Anyone

Flight Date:

Low Altitude

Complete?	Action:	Tools, materials Needed:	Details:	Who can do it?
	Ensure connection to Pi	Pi, laptop, extra jumper cables, wiring diagram (on hood)	Insert Pi into drone, insert wifi antenna, connect jumpers from Flight Controller, connect to laptop through wifi	Israel, Anthony, Isaac, Tristan, Janie

Ensure camera capabilities	Laptop		Anthony, Tristan, Isaac
Drone battery level verification	Multimeter		Anyone
Remote Battery Level Verification	Controller		Brian
Receiver Test	Controller	Power Drone on, test receiver range	JBJ
Drop servo test	Laptop	Something to drop	Anthony, Tristan, Isaac
Calibrate accelerometer	Laptop - QGroundControl	Stationary dance	JBJ (Jonah, Brian, Jadyn) + Anyone
Calibrate compass	Laptop - QGroundControl	Spinning dance	JBJ + Anyone
Calibrate gyro	Laptop - QGroundControl	Set on level ground	JBJ + Anyone
Calibrate level horizon	Laptop - QGroundControl	Set on level ground	JBJ + Anyone
Acquire GPS signal	Laptop - QGroundControl	Power up, take laptop and plane into the open and validate gps lock via Q ground control	JBJ + Anyone
Create flight plan	Laptop - QGroundControl	Create and load flight plan	JBJ
Screw in rear lid	Rear lid screws, .05 in hex wrench		Airframe Integration
Attach wings to fuselage			Airframe Integration
Attach tail fins			Airframe Integration
Attach rear prop	Nut, cap, wrench, prop	Screws on opposite of normal direction	Lucas, Anna, Jadyn, Jonah, Brian
Right aileron servo test	Controller		Brian

Left aileron servo test	Controller		Brian
Right flap servo test	Controller		Brian
Left flap servo test	Controller		Brian
Insert Front Lid	Front Lid	Has retractable pin to secure	Anyone
Center of Gravity Verification		Pick up drone under wings and ensure balance	JBJ + Airframe integration
Final visual inspection			
Set up ROS nodes	Laptop	Launch ROS nodes and start collecting data	Anthony, Tristan, Isaac

Additional Tools/Materials: Do we have it?

Multimeter

Servo tester

Wifi antenna

Tweezers

Hex sets

Painters tape

Ballast

Extra jumper cables

Foam tack glue

Speed tape

Velcro

USB cables

Phillips head screwdriver

Controllers

All props (4x quad rotors, 2x tail props)

Nuts

Crescent wrench

Wings, tail fins, lids

Rear lid screws

Matches/lighter

Wood

Fire extinguisher

gallon of water

Charged Batteries

scissors