

ETSD037

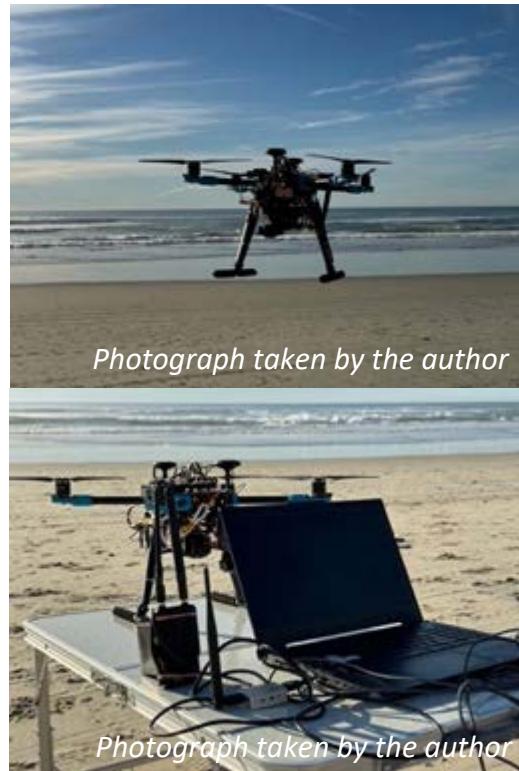
An Autonomous Unmanned Aerial Vehicle (UAV) System for Ocean Hazard Recognition and Rescue: Scout and Rescue UAV Prototypes

Research Notebook

Angelina Kim, The Bishop's School, La Jolla, California, U.S.A.



Photograph taken by the author



Photograph taken by the author

Table of Contents

<i>Figures</i>	6
<i>Tables</i>	9
<i>Abstract</i>	11
<i>Acknowledgements</i>	12
1. <i>Introduction</i>	13
2. <i>Review of Literature</i>	14
2.1. Ocean safety and lifeguarding.....	14
2.2. Lifeguard UAV application	15
2.3. Rip current detection	15
2.4. Type of VTOL UAVs	16
2.5. Over-actuated multicopter design	16
2.6. Rescue pulling as a slung load problem	19
2.7. FAA and local UAV regulation	20
3. <i>Statement of Problem</i>	21
4. <i>Hypothesis</i>	22
5. <i>Materials</i>	23
5.1. Scout UAV bill of materials	23
5.2. Rescue UAV bill of materials.....	24
5.3. Common support materials.....	28

6. Procedure	29
6.1. Lifeguard UAV requirements.....	29
6.2. Lifeguard UAV development strategy.....	30
6.3. UAV design common components.....	34
6.3.1. Flight controller.....	34
6.3.2. AI mission controller	36
6.3.3. Communication links	37
6.3.4. Battery	38
6.4. Scout UAV design, development, and operation.....	39
6.4.1. Scout UAV implementation and system block diagram.....	40
6.4.2. Motors and propellers	42
6.4.3. Mission management program.....	43
6.4.4. Scout UAV operation procedure	44
6.5. Rescue UAV design and build	45
6.5.1. Rescue UAV system block diagram.....	46
6.5.2. Extended and Symmetric In-Arm Pitch Axis.....	47
6.5.3. Actuator and propeller arrangement.....	50
6.5.4. Actuator and propeller.....	51
6.5.5. Servo motor for pitch and roll tilt rotor.....	53
6.5.6. In-arm pitch and roll design	54
6.5.7. Component layout.....	57
6.5.8. Rescue UAV design for folding	59
6.5.9. Material fabrication process.....	61
6.5.10. Building up	68
7. Results and Statistical Analysis.....	70

7.1.	Scout UAV ocean image collection	70
7.2.	Scout UAV flight analysis	74
7.3.	Rescue UAV roll and pitch demonstration.....	75
8.	<i>Analysis</i>	77
8.1.	Information-Weighted Average Differential Frame Displacement.....	77
8.1.1.	Differential frame displacement and error cross-correlation	77
8.1.2.	Ocean image analysis with DFD and ECC	80
8.1.3.	Information-Weighted Differential Frame Displacement.....	82
8.2.	Rip current detection with channel current analysis.....	87
8.2.1.	Ocean depth and risk model	87
8.2.2.	Rip current analysis with depth and risk models	89
8.3.	Rescue drone test flight.....	90
	<i>Conclusion</i>	93
	<i>Recommendations</i>	94
	<i>Bibliography</i>	95
	<i>Appendix A. Regulation compliance documents</i>	103
	<i>Appendix B. Tidal charts</i>	105
	<i>Appendix C. Images from Scout Missions</i>	108
	<i>Appendix D. Fabrication templates</i>	112
	<i>Appendix E. Computer programs</i>	125

Appendix F. Flight reports 197

Appendix G. Research progress notes..... 221

Figures

Figure 1. The scout UAV quadcopter in action for a flight mission at Del Mar beach	39
Figure 2. Scout UAV system block diagram	40
Figure 3. Top view of scout drone and components.....	40
Figure 4. Side view of the scout UAV	41
Figure 5. Back view of the scout UAV.....	41
Figure 6. Scout UAV mission management system.....	43
Figure 7. Rescue UAV under preflight check.....	45
Figure 8. Rescue UAV function block diagram.....	46
Figure 9. Conventional end-arm pitch axis and tilt angle.....	47
Figure 10. Proposed novel in-arm pitch axis and tilt angle.....	48
Figure 11. Maximum pitch angles with end-arm and in-arm pitch axes	49
Figure 12. Propeller trajectory with pitch and roll angle.....	50
Figure 13. Top actuator and propeller assembled on the pitch block	51
Figure 14. Roll and pitch drive servo motors assembled	54
Figure 15. In-arm pitch -45-degree coaxial rotor tilt CAD view.....	55
Figure 16. In arm pitch +45-degree coaxial rotor tilt CAD view.....	55
Figure 17. In-arm pitch block design and views with servo motor and inner gear	56
Figure 18. Arm roll angle drive design and servo motor.....	56
Figure 19. Body compartment opening for component placement	58
Figure 20. Power distribution board and ESC routing	58
Figure 21. Rescue UAV folding steps top view.....	59
Figure 22. Rescue UAV folding steps perspective view.....	59

Figure 23. Actuator and servo motor wiring plan with folding	60
Figure 24. An example CNC milling template for 300x400x2mm carbon fiber plate.....	61
Figure 25. Water-immersed CNC milling in progress.....	62
Figure 26. Water-immersed CNC milled carbon fiber board	63
Figure 27. 3D-printed servo motor spur gear reinforced with core screws.....	64
Figure 28. An example 3D-printing template for rescue UAV custom parts.....	65
Figure 29. 3D-printed rescue UAV custom components	66
Figure 30. Rescue drone build up progress photos.....	68
Figure 31. Ocean image collection flight plan	71
Figure 32. Mobile ground control station during scout mission.....	71
Figure 33. Example tide chart from NOAA used for scout planning	72
Figure 34. Ocean image during King Tide on 12/25/2023	73
Figure 35. Ocean image collected on 11/22/2023	73
Figure 36. PX4 Flight Review flight log analysis plots from scout mission #9	74
Figure 37. Pitch and roll actuation demonstration with a Python program.....	76
Figure 38. Rotor tilt across pitch angle range.....	76
Figure 39. Example ocean image section for comparison.....	79
Figure 40. DFD and ECC vector analysis results.....	79
Figure 41. Scout UAV ocean image for wave analysis	80
Figure 42. Zoomed views of focus areas for analysis	80
Figure 43. Water flow vectors from DFD in red and ECC in blue.....	81
Figure 44. Histograms of RGB intensity	82
Figure 45. 2D gradient images of the area #1 - 3	83

Figure 46. Information-weighted flow vector in red vs. before weighting in blue	84
Figure 47. Gradient density as information weight from the whole image.....	85
Figure 48. Information-weighted flow in red vs. unweighted flow in blue.....	86
Figure 49. Information-weighted incoming (blue) and receding (red) for vectors	86
Figure 50. Rip current depth and risk model	88
Figure 51. Rip risk model analysis results.....	89
Figure 52. Comparison of unweighted, depth, and depth+risk model rip flow.....	90
Figure 53. The rescue UAV actuator configuration as a coax hexacopter.....	90
Figure 54. Rescue UAV test flight in progress	91

Tables

Table 1. Summary of tilt rotor and 6 DOF UAVs.....	17
Table 2. Scout UAV major component bill of materials.....	23
Table 3. Rescue UAV major component bill of materials.....	24
Table 4. CNC machined components for rescue UAV.....	25
Table 5. 3D-printed components for rescue UAV.....	25
Table 6. Cables and connectors	26
Table 7. Rescue drone build screw list	26
Table 8. 3D CAD models from GRABCAD library	27
Table 9. Supporting computing devices.	28
Table 10. List of major equipment and tools for fabrication and assembly.....	28
Table 11. Lifeguard scout and rescue UAV requirements.	29
Table 12. Summary of research process - recursive rather than linear.....	31
Table 13. Comparison of PX4 and Ardupilot flight controller stack.....	34
Table 14. Pixhawk 6X major functions	35
Table 15. Computing platform comparison.....	36
Table 16. Lifeguard UAV communication radio links (SIYI, 2024)	38
Table 17. Battery specification comparison (GETFPV, 2024), (Amazon, 2024)	38
Table 18. Scout UAV thrust with AT2317 and APC11 (T-MOTOR, 2024).....	42
Table 19. Scout UAV beach operation procedure.....	44
Table 20. Rescue UAV thrust calculation with actuator and propeller.....	52
Table 21. Specification of Dynamixel X430-W250-T servo motor	53
Table 22. Water-immersed CNC milling process	61

Table 23. CNC templates and fabrication time.....	63
Table 24. 3D printing process steps.....	66
Table 25. 3D print templates and printing hours	67
Table 26. Scout UAV ocean image data collection missions.....	70

Abstract

Autonomous Unmanned Aerial Vehicles (UAVs) have potential to aid lifeguard operations as 82% of ocean-related deaths in California occur at unprotected beaches. However, commercial drones employed at select guarded beaches still require the manual lifeguard operation. This research developed an autonomous UAV system where a scout UAV can accurately detect rip currents early and activate a rescue UAV, which shortens the response time by dropping flotation devices and pulling victims to safety. The scout quadcopter was designed, built, and implemented with autonomous flights. The scout drone has wide-range triple radio communication links for control and manual override. Launch procedures and safety protocol were established. Nine ocean scout missions were conducted and collected 11.8k ocean images over 13.4km at San Diego beaches. Image analysis with a new information-weighted differential frame displacement vectors was conducted for optical flow and particle image velocimetry. The resulting wave displacement vectors used a new channel current flow analysis with depth and risk models to detect rip currents. For the rescue coaxial hexacopter, a novel in-arm pitch axis design was designed, built, and tested in flight. The rescue UAV spans 2.0m and weighs 12.3Kg with 44.1Kg total thrust for rescue missions. It can be folded into 1.1m for transportation. The in-arm pitch axis extends pitch angle by more than 65% from conventional end-arm pitch axes. The pitch axis realizes the most pitch angle up to ± 45 degrees with symmetric coaxial actuators when compared to conventional UAVs and has a wide roll angle up to ± 180 degrees.

Acknowledgements

As this project involved hands-on and iterative cycles of designing, printing, cutting, and building materials, I needed continuous access to a workplace that accommodated my computer, CNC machine, 3D printer and the assembly of drone parts. My parents graciously agreed to let me convert our garage into a machine shop. For the initial flight tests, I used our backyard and fortunately didn't experience any major incidents except damaging palm trees. I thank my parents for having kindly said that I just gave the trees an earlier leaf trimming than expected. I would like to extend my gratitude to my neighbors and fellow beachgoers whom I ran into during the pilot tests at my local beaches. Thank you for your kind understanding with noise and generous encouragement for this project.

Dr. Anthony Pelletier, our chemistry and biology teacher at my school, a researcher at Scripps, and sponsor of this project, provided unwavering support and encouragement throughout the process. Dr. Marcus Jaiclin, a computer science teacher at the school, also encouraged me to build a machine learning model for drones that laid the foundation of this research. For my computer science knowledge, I am thankful for the Harvard edX courses and community. For my mechanical skills, I credit my former First Tech Challenge robotics team (The Clueless #11212). For all mathematical knowledge and scientific processes applied to this project, I thank my current and former math and science teachers including Dr. Jay Zhao, Mr. Marcus Milling, Mr. Noble Kime, and Mr. Ben Duehr. Finally, I am indebted to anonymous proposal reviewers and members of engineering communities, such as the GSDSEF (Mr. Steve Rodecker), CSEF, the American Society for Nondestructive Testing, and the Institute of Electrical and Electronics Engineers (IEEE), for reviewing my previous research and providing valuable feedback that served as the groundwork for this project.

1. Introduction

Coastal communities in San Diego and California suffer from ocean fatalities. California has the most ocean incidents in the nation, and it has twice the number of incidents as the second-ranked state, Florida (United States Lifesaving Association, 2023). USLA reports that most of incidents occur at unguarded beaches, and that rip currents are major threats whether guarded or unguarded. I discovered that Unmanned Aerial Vehicles (UAVs) would be useful to reinforce lifeguard operation. Furthermore, an autonomous lifeguard UAV operation would overcome coverage, accuracy, and latency limitations. This research investigates the design requirements for lifeguard UAVs and system. It implements a desired scout and rescue UAV configurations and presents what lifeguard UAVs can offer to assist and expand lifeguard operation at our beaches.

2. Review of Literature

My literature review visited multiple areas of expertise, and they are categorized as follows:

- Ocean safety and lifeguarding (2.1)
- Lifeguard UAV application (2.2)
- Rip current detection (2.3)
- Types of UAVs (2.4)
- Over actuated multi-copter design (2.5)
- Rescue pulling as a slung load program (2.6)
- FAA and local UAV regulation (2.7)

2.1. Ocean safety and lifeguarding

In 2019, approximately 239,000 people died from drowning, the third-leading cause of unintentional injury and death (World Health Organization, 2023). Additionally, 50% of drowning victims are under the age of 30, and over 90% of drowning deaths occur in low- and middle-income countries (World Health Organization, 2023). Thus, lower socio-economic status is associated with the drowning (World Health Organization, 2023).

One critical factor against beach drowning is lifeguarding. Unguarded beaches have approximately five times more fatalities than the guarded beaches. It is also reported that most of fatalities are from rip currents, whether a beach is guarded or unguarded (United States Lifesaving Association, 2023). It is not recommended to swim at unguarded beaches for the

same reason (California Department of Parks and Recreation, 2023), and there are victims even at the Torrey Pines State Beach when unguarded (Kucher, 2022).

Lifeguard operation requires short response time. American Red Cross offers lifeguard training, and its manual states that a lifeguard should identify an incident within 30 seconds, and the victim should be rescued within 2 minutes, where ventilation and resuscitation are feasible (American Red Cross, 2023).

2.2. Lifeguard UAV application

UAVs are getting adopted to support lifeguard operation at select beaches. One example is to detect ocean predators (New York State, 2023). Those UAVs are still in early stages of development, and rely on manual steering and detection through visual examination.

For rescue operations, a simulation of flotation device delivery was researched for feasibility (Xiang, Hardy, Rajeh, & Venuthurupalli, 2016). The UAV is dedicated for rescue mission and controlled manually.

2.3. Rip current detection

Rip current detection has been tried with dedicated instruments such as acoustic wave (Nakano & Ishida, 2012) and doppler radars (Trizna, 2018). An observation from a stationary tower was compared with doppler radar-based particle image velocimetry (PIV) and optical flow (OF) to correlate rip current detection (Perkovic, Lippmann, & Frasier, 2009), (Puleo, Farquharson, Frasier, & Holland, 2003), and both of them performed consistent detection. Also,

a UAV-based PIV analysis was investigated to propose a displacement frame difference and to detect water flow (Dérian & Almar, 2017). Also, an error correlation has been used to perform optical PIV analysis (Holland, Puleo, & Kooney, 2001), (Hart, 1998). Lucas and Kanade technique based on least-squared error has been used commonly for optical flow analysis (Lucas & Kanade, 1981).

2.4. Type of VTOL UAVs

The survey of UAVs was limited to Vertical Take Off and Landing (VTOL) UAVs as the lifeguard tower is the only space lifeguard UAVs can be hosted. One of the mostly deployed VTOL with tilt rotor is V-22 Osprey tilt-rotor aircraft (Gung Ho Vids, 2023).

Thrust vectoring of the mono-copter actuator simplifies main actuator, while vectoring controlled by additional motors (bresh9019, 2023), (nicolalego, 2023). Tilt rotor coaxial mono-copter is the other implementation of vector thrusting, and it is likely more efficient and lighter than vector thrusting of rotational wings (VGR-Systems, 2024). One of the extreme cases of thrust vector UAV is bladeless UAV with air ducting with pipes (Rivellini, 2024), while it does not have thrust vector. It would be one of the safer UAVs that does not expose propeller blades outside the UAV.

2.5. Over-actuated multicopter design

Regular multicopters are under-actuated, and they have only 4 Degrees Of Freedom (DOF). There are multiple designs reported to achieve 6 DOF (Jiang & Voyles, 2013), and eventually perform omni-directional flight (Brescianini & D'Andrea, 2016).

Rotor tilt has been major focus to achieve such DOF. Rotor arms have roll around the arm axis with single actuator (Kamel, et al., 2018), (Ryll, Bulthoff, & Giordano, 2013) or coaxial configuration (Allenspach, et al., 2020). In addition to the roll, arm pitch angle has been added for dual-axis rotor tilt (Junaid, Sanchez, Bosch, Vitzilaios, & Zweiri, 2018), (Segui-Gasco, Al-Rihani, Shin, & Savvaris, 2013).

The pitch tilt as second rotor axis can avoid singularity angle during omnidirectional flight. The pitch angle implemented is highly limited by the design, and a bent arm design was proposed to increase pitch angle range to maximize pitch rotor tilt (Rajappa, Ryll, Bulthoff, & Franchi, 2015). The design is not practical and not applicable to coaxial thrust design, as it is not symmetric. The UAV photos and implementations of rotor tilt and configuration is summarized in Table 1.

Table 1. Summary of tilt rotor and 6 DOF UAVs.

Citation	Photo	Rotor tilt	Maximum pitch angle	Coaxial
(Brescianini & D'Andrea, 2016)	 <i>Photo taken from Brescianini & D'Andrea (2016, p.3261)</i>	n/a	n/a	No
(Ryll, Bulthoff, & Giordano, 2013)	 <i>Photo taken from Ryll, et. al. (2013, p.295)</i>	Roll	n/a	No

(Kamel, et al., 2018)		Roll	n/a	No
(Allenspach, et al., 2020)		Roll	n/a	Yes
(Segui-Gasco, Al-Rihani, Shin, & Savvaris, 2013)		Roll and pitch	<20	No
(Rajappa, Ryll, Bulthoff, & Franchi, 2015)		Roll and pitch	<40	No
(Junaid, Sanchez, Bosch, Vitzilaios, & Zweiri, 2018)		Roll and pitch	<20	No

Table created by the author

UAVs with high degrees of freedom need control schemes to manage roll and pitch in addition to rotor speed. A Newton-Euler translational and rotational dynamics of a rigid body equation is used to build a resource allocation in the flight controller (Allenspach, et al., 2020),

(Segui-Gasco, Al-Rihani, Shin, & Savvaris, 2013), (Kamel, et al., 2018), (Ryll, Bulthoff, & Giordano, 2013), (Rajappa, Ryll, Bulthoff, & Franchi, 2015), (Jiang & Voyles, 2013).

Tilt rotor multi-copters with only arm rolling suffer singularity at certain conditions.

There two singularities an omni-directional UAV can faces. A rank reduction singularity is when a UAV loses instantaneous controllability of forces and torques (F., Bicego, Ryll, & Franchi, 2018). A kinematic singularity is when the rotor thrust tilt angles have non-unique solutions (Bodie, Taylor, Kamel, & Siegwart, 2018). Pitch tilt rotors can have this problem unless a proper control algorithm utilizing pitch and roll angles is implemented.

2.6. Rescue pulling as a slung load problem

In addition to the dynamic control and omni-directional orientation and flight, pulling a victim out of water to safety needs a careful control scheme. It is similar to the so-called “slung load” problem, where an UAV transports a load attached through a cable (Yang, Xian, Cai, & Wang, 2024), (Yang & Xian, 2020).

A differential flatness has been useful for solving the control problem without solving differential control equations analytically. It uses flat outputs, and a finite order of derivatives are used to express system state and inputs (Sreenath, Lee, & Kumar, 2013).

For a rescue UAV to pull a victim, additional environments should be considered as design parameters. For example, wind, wave, rip currents, victim’s weight, and victim’s movement are unknown and vary dynamically.

2.7. FAA and local UAV regulation

Federal Aviation Administration (FAA) lays out regulations for unmanned aircraft systems (Federal Aviation Administration, 2023). There are limited statutory exceptions with basic sets of requirements (Federal Aviation Administration, 2023). Several rules are listed here:

- Keep your drone within visual line of sight
- Give way to and do not interfere with other aircraft
- A drone should weigh less than 55 pounds
- Fly at or under FAA-authorized altitudes in controlled airspaces
- Fly under 400 feet at uncontrolled airspace
- Take The Recreational UAS Safety Test (TRUST) and carry proof of test passage
- Register your drone, have a current FAA registration, and mark your drones
- Broadcast remote ID information
- Use B4UFLY apps to check the airspace control and availability (Federal Aviation Administration, 2024)

For nighttime UAV operation, UAVs should be equipped with anti-collision lighting visible from 3 statute miles (14 CFR Part 107, 2023).

Local governments have additional set of regulations. For example, the City of Del Mar allows drone flight over the ocean, but not over the beach (City of Del Mar, 2023).

FAA certification and registration documents are listed at Appendix A. Regulation compliance documents.

3. Statement of Problem

The following are problems researched to answer:

Main question:

- Can autonomous lifeguard UAVs save people's lives from ocean hazards?

Sub-questions:

- What are the requirements for the lifeguard UAVs and rescue system?
- What are the scout drone operation procedures?
- How can the scout drone analyze ocean images and detect rip currents?
- What are the rescue drone's design requirements to perform rescue operation?
- Which rescue drone design can overcome singularity during omnidirectional flight?

4. Hypothesis

It is hypothesized that:

- UAVs can be adapted and built to support lifeguard operation.
- For scout UAV:
 - UAVs can perform autonomous scout mission.
 - It can fly with a flight plan to collect ocean images.
 - It can analyze ocean images to show the water flow.
 - It can detect rip currents from water flow data.
 - It can have multiple radio links for control to stably perform missions under stress.
- For rescue UAV:
 - UAVs can be custom designed for lifeguard rescue mission.
 - It has enough thrust to carry inflatable flotation devices with a rope and to pull the victim to safety.
 - It can tilt rotors with roll and extended pitch angles for omni-directional orientation and flight.
 - It can be folded into a size that can be transported in passenger vehicles.

5. Materials

Materials and components for scout and rescue UAVs are listed separately in the following subsections.

5.1. Scout UAV bill of materials

The scout UAV is a quadcopter using a commercial frame and custom components list in Table 2. The table has major components and blocks only, and it does not include minor materials such as wires, connectors, zip ties, etc.

Table 2. Scout UAV major component bill of materials.

#	Description	Part #	Manufacturer or supplier	Count
1	Flight controller	Pixhawk 6X	Holybro	1
2	Flight controller expander	Pixhawk 6X baseboard	Holybro	1
3	Mission controller	JETSON Nano	NVIDIA	1
4	Camera gimbal, IP (HD)	A8	SIYI	1
5	Camera module, 10MP	Camera module 3	Raspberry Pi	1
6	IP radio (5GHz)	HM30	SIYI	1
7	Telemetry radio (2.4GHz)	Datalink	SIYI	1
8	RC radio (2.4GHz)	FM30	SIYI	1
9	Actuator	AT2317 KV880	T-Motor	4
10	Electronic speed control	BLHeli S 20A	Holybro	4
11	Propeller, 11 inch	MS1101L	T-Motor	4
12	Strobe	ARC V	Firehouse	2
13	Remote ID	Cube ID	IR-LOCK	1
14	GPS #1	M9N	Holybro	1
15	GPS #2	M8N	Holybro	1
16	Ethernet switch, 5 port	TL-SF1005D	TP-Link	1

17	Universal mount kit	Payload platform board	Holybro	2
18	Power distribution board	PDB01-V1.2	Holybro	1
19	Quadcopter frame	X500 V2	Holybro	1
20	Power module	PM02 V3 (12S)	Holybro	1
21	5V converter (3A)	3A	ShareGoo	1
22	4S Li-Ion battery	Dark Lithium V2	Upgrade Energy	6

*Excluding wires, connectors, zip ties, and double-sided tapes

Table created by the author

5.2. Rescue UAV bill of materials

The rescue UAV was custom designed with a novel extended pitch angle, and it was built from scratch. Major components listed in Table 3. In addition to major components, here are 86 custom-designed and CNC-machined carbon fiber plate parts and 320 3D-printed parts listed in in addition to the major components, and these are listed in Table 4 and Table 5.

Table 3. Rescue UAV major component bill of materials.

#	Description	Part #	Manufacturer or supplier	Count
1	Flight controller	Pixhawk 6X	Holybro	1
2	Flight controller expander	Pixhawk 6X baseboard	Holybro	1
3	Mission controller	JETSON Orin Nano	NVIDIA	1
4	Flight control power module	PM02 V3 (12S)	Holybro	1
5	Power distribution board	PDB500	APD	1
6	Camera gimbal, IP (HD)	A8	SIYI	1
7	IP radio (5GHz)	HM30	SIYI	1
8	Telemetry radio (2.4GHz)	Datalink	SIYI	1
9	RC radio (2.4GHz)	FM30	SIYI	1
10	Actuator	MN5008	T-Motor	12
11	Electronic speed control	51A BLHeli_32	Lumenier	12
12	Propeller, 18 inch	MF1806	T-Motor	12

13	12V converter (5A)	MATEKSYS 12S Pro BEC	Matek	3
14	Remote ID	Cube ID	IR-LOCK	1
15	GPS #1 and #2	M9N	Holybro	2
16	Ethernet switch, 5 port	TL-SF1005D	TP-Link	1
17	Servo motor	Dynamixel XL430-W250-T	Robotis	12
18	Servo expansion board	3P JST expansion board	Robotis	2
19	Servo USB control	U2D2 and Power Hub	Robotis	1
20	Ethernet switch, 5 port	TL-SF1005D	TP-Link	1
21	Pitch axis bearing	2mm / 5mm / 2.3mm	uxcell/Amazon	12
22	Roll axis bearing	22mm / 28mm / 16mm	uxcell/Amazon	12
23	Carbon fiber tube	22mm - 2mm thick	CNCCarbonFiber	12
24	6S LiPo battery	HRB 22.2V 3300mAh	Amazon	2

*Excluding zip ties and double-sided tapes

Table created by the author

Table 4. CNC machined components for rescue UAV.

#	Description	Size L (mm) x W (mm)	Thickness	Count
1	Body plate	332 x 290	4mm	2
2	Arm frame horizontal	379 x 69	4mm	12
3	Arm frame vertical long	342 x 46	4mm	6
4	Arm frame vertical short	135 x 46	4mm	6
5	Roll plate	198 x 80	2mm	6
6	Pitch side plate	256 x 170	2mm	12
7	ESC plate	53 x 30	2mm	12
8	Actuator plate	53 x 30	2mm	12
9	Pitch servo front plate	110 x 36	2mm	6
10	Pitch servo back plate	103 x 51	2mm	6
11	Pitch block guide	98 x 42	2mm	6
Total				86

Table created by the author

Table 5. 3D-printed components for rescue UAV.

#	Description	Filament length (m)	Weight (g)	Count
1	Leg socket	24.99	69	3

2	Shoe socket	10.25	28	3
3	Arm roll holder	3.85	11	24
4	Arm frame bracket	0.43	1	120
5	Pitch axis fixture	5.05	14	12
6	Pitch servo fixture	1.7	5	24
7	Fixture axis holder	0.48	1	36
8	Pitch axis stopper	0.11	0	12
9	Roll axis stopper	1.51	4	6
10	Motor plate bracket	0.77	2	24
11	ESC plate bracket	0.4	1	24
12	Pitch inner gear	10.11	28	6
13	Roll gear	3.28	9	6
14	Servo spur gear	0.65	2	12
15	JETSON Orin Nano mount kit	12.24	34	8
Total				320

Table created by the author

Table 6. Cables and connectors

#	Description	Usage
1	12AWG cable	Battery
2	14AWG cable	Motor
3	16AWG cable	Periphery
4	22AWG cable	Servo motor
5	XT150 connector	Actuator
6	XT60 connector	Battery, periphery
7	DuPont connector	ESC control

Table created by the author

Table 7. Rescue drone build screw list

Description	size	type	length	count	instance	per arm	total
Motor bracket	M2	Pan	8	6	2	12	72
Pitch gear	M2	Pan	14	7	1	7	42
ESC bracket	M2	Pan	6	6	2	12	72
Serve gear core	M2	Flat	8	3	2	6	36
Servo gear mount	M2	Flat	6	4	2	8	48

Servo gear mount	M2	Pan	6	4	2	8	48
Arm pitch fixture axis	M2	Pan	25	4	1	4	24
Arm pitch fixture axis	M2	Pan	40	2	1	2	12
Arm pitch fixture axis	M2	Pan	12	2	2	4	24
Arm pitch fixture axis	M2	Pan	10	2	2	4	24
Arm pitch fixture axis	M2	Pan	8	4	2	8	48
Arm pitch fixture axis	M2	Pan	14	2	1	2	12
Arm servo fixture assy	M2	Pan	25	8	2	16	96
Arm servo fixture assy	M2	Pan	40	4	2	8	48
Arm servo fixture	M2	Pan	25	1	2	2	12
Arm servo fixture	M2	Flat	12	2	2	4	24
Arm servo fixture	M2	Flat	12	2	2	4	24
Arm servo fixture	M2	Pan	10	2	4	8	48
Arm servo fixture	M2	Pan	8	2	4	8	48
Arm servo fixture	M2	Pan	6	4	4	16	96
Servo plate	M2.5	Flat	10	8	1	8	48
Arm roll gear	M2	Pan	30	1	2	2	12
Arm roll fixture	M2	Pan	40	4	2	8	48
Group plate	M2.5	Wafer	4	4	1	4	24
Group plate to body	M3	Pan	60	12	1	12	72
Extender to	M3	Pan	70	7	1	7	42
Extender, vertical	M3	Pan					
Extender, bracket	M3	Pan	10	5	4	20	120
Extender, bracket	M3	Pan	8	5	4	20	120
Extender, frame	M3	Pan					
Landing leg, frame	M3	Pan	10	4	3	12	12
Landing foot, leg	M3	Pan	8	1	12	12	12
Total							1368

Table created by the author

Table 8. 3D CAD models from GRABCAD library

#	Item	GRABCAD model
1	T-MOTOR MN5008	(PATEL, 2021)
2	Dynamixel XL430-W250-T	(f, 2023)
3	T-Motor 18x6 composite folding propellers	(Gibson, 2020)
4	Pixhawk 6X	(Tan, 2023)

Table created by the author

5.3. Common support materials

The following tables have common support materials used for both scout and rescue UAVs. Computing systems are listed in Table 9. The ground laptop was used for scout and rescue UAV flights. Servers were utilized to perform wave image analysis, rip current detection, and machine learning. Fabrication equipment and most used tools are listed in Table 10.

Table 9. Supporting computing devices.

#	Description	Model	CPU	GPU	Memory (GB)	Storage (TB)
1	Ground laptop	ASUS TUF Dash F15	i7-12650H	RTX3050Ti	32	1
2	Server #1	Custom build	i9-9900K	-	64	1
3	Server #2	Custom build	i9-12900K	RTX3090	128	1
4	Server #3	Custom build	i9-13900K	RTX4090	128	1

Table created by the author

Table 10. List of major equipment and tools for fabrication and assembly.

#	Item	Model
1	3D printer	Creality Ender 5 Plus
2	CNC machine	CNC 6040 4 Axis 2200W spindle
3	Battery charger	ISDT K4 AC 400W DC 600Wx2
4	Electric screwdriver	Populu 4V
5	Solder	Weller WLC100
6	Drill	BOSCH PS11-102
7	Filament dryer	SUNLU filament dryer

Table created by the author

6. Procedure

UAV design processes are arranged by elements for scout and rescue UAVs as procedure. Lifeguard UAV requirements and a framework for UAV development is described in 6.1 and 6.2. Scout UAV design and rescue UAV design and process are discussed in 6.3, starting with common components, and then at 6.4 and 6.5 respectively.

6.1. Lifeguard UAV requirements

Lifeguards' identification and response time constraints discussed in 2.1 set up basic UAV design requirements. The UAVs need to travel the full beach range every 30 seconds to first detect an incident. While it depends on the width of the beach, multiple UAVs will form a scout fleet to perform such visits frequently. On the other hand, a rescue mission to carry an inflatable flotation device (IFD) with a rope attached to the UAV and to pull a victim to safety requires much greater thrust than the scouting mission. By the nature of the missions, scouting requires continuous overlapping missions, while a rescue mission is an on-demand response based on a scout UAV's incident detection. Table 11 summarized scout and rescue UAV requirements. There are common items such as artificial intelligence (AI) capability, flight controllers, and reliable communication. Rescue UAV requires mechanical dynamics to allow omni-directional orientation and flight, in addition to larger scale thrust capability.

Table 11. Lifeguard scout and rescue UAV requirements.

Requirement	Solution		Affected design parameters
	Scout UAV	Rescue UAV	
AI capability	On-board mission controller		Payload, power
Precise control	Multiple IMUs and GPSs in FC		Payload, power
Flight duration	High-density, large-capacity Li-Ion	High current LiPo	Payload, thrust, power
Reliable communication	Wide-range and redundant radio links		Payload, mount area
Multiple camera	Camera gimbal and fixed mount camera		Payload, IP network
Payload	Up to 3Kg	Up to 20Kg	
Thrust for payload	6Kg	40Kg	Power
Omni-directional flight	-	Dual-axis tilt rotor	Payload, power, control
No orientation singularity	-	Extended and symmetric pitch rotor tilt	Payload, power, control
Carry lifesaving tool	-	Inflatable flotation device and rope	Payload, control
Pull victim to safety	-	Tilt rotor and multiple arms for thrust	Power, thrust
Reliable thrust	-	Coaxial hexacopter for redundancy	Design complexity, control
Transportation	Fit in passenger vehicle	Folded to fit in passenger vehicles	Design

Table created by the author

6.2. Lifeguard UAV development strategy

A systematic arrangement of project development phases is listed in Table 12. Several development frameworks and models, such as the waterfall model, rapid prototyping, and aircraft development lifecycle model, in engineering were reviewed, and three development phases are adopted for the research.

- Phase I: Design, certification, and patent

- Phase II: Prototyping, fabrication, and ML training
- Phase III: Pilot-testing and revision

Table 12. Summary of research process - recursive rather than linear

Phase	Deliverable	Duration	Location	Tools/mentor	Status
Preliminary phase 	-A deep reinforcement ML model developed for drone and tested in simulation -IEEE research paper (Kim, 2023a)	June 2022-May 2023	Home	Gazebo Python PX4Autopilot Harvard edx Dr. Jaclin	Completed
Phase I Design, certification, and patent 	<p>- Conceptual design a UAV Scout and Rescue system for Ocean safety - FAA Recreational pilot certification - Research manuscript for new mechanical design for rescue drone (Kim, under review) - CAD Design - Patent filed at USPTO (Kim, 2023b)</p> <p><i>Autonomous Scout Drone</i> - Software: Ground control station and application Custom program codes to control camera, recording, and forwarding. Open-source program for flight controller. - Hardware: Flight controller with expandable periphery I/O connection. AI processor for image processing. Brushless motor and propeller for additional thrust for speed and wind resistance. Two on-board cameras. One for aviation (front) and the other for beach (downward). Multiple radio connections – IP, telemetry, and RC for redundancy Two GPS receivers for accurate position and orientation</p>	June 2023-Dec 2023	Home	PowerPoint Excel Word FAA website Fusion360 QGroundControl Python JupyterNotebook Gstreamer Dr. Pelletier	Completed

	Omnidirectional Rescue Drone - Omni directional orientation and flight capability - In-arm design to increase actuator pitching for omnidirectional flight (Kim, under review) and roll without singularity - Redundant actuators - Strong enough thrust to pull a person - Rope, camera, sensor, multiple radio connections same as scout drone. - Peripheries same as scout drone - CAD design and assembly				
Phase II Prototyping (fabrication and ML training)	<p>Machine shop (garage) set up</p> <p>Autonomous Scout Drone Order parts: NVIDIA JETSON Nano development kit, 2317 motor and 11 inch propeller, SIYI A8 camera gimbal and Raspberry Pi camera module 3, IP radio, telemetry radio, and RC radio, GPS modules, 4S Li-Ion batteries, remote ID, strobe</p> <p>Build: Used universal mount platform, zip tie, double sided tape, and nuts and screws</p> <p>FAA Drone registration: FA3HRKM494 as 1st device</p>	Aug 2023- Feb 2024	Garage	CNC machine 3D Printing Drill Solder Zip tie FAA website Python Tensorflow Python numpy library Custom image wave feature extraction and	Fabrication completed Ocean image data collected for training Wave image data analyzed with RGB channel differential filtering Machine learning

	<p>Omnidirectional Rescue Drone</p> <p>Order parts: Many M2 and M3 screws and nuts. Reels of Nylon and 3D print glue. Carbon fiber plate and tube. Servo motor. Actuator and propeller. Roller and ball bearing, NVIDIA JETSON Orin development kit, wiring cables</p> <p>3D print parts: Nylon reels, custom inner gear, spur gear, brackets, arm holder, axis stopper, JETSON Orin mounts</p> <p>CNC milling: 2mm and 4mm carbon fiber plates. Body plate, arm plates, pitch plate, roll plate, motor and ESC plate</p> <p>Build: Pitch block, roll block, servo motor mount, body frame, and arm frame, solder wiring connectors, solder ESC and power distribution board.</p> <p>FAA Drone registration: FA3HRKM494 as 2nd device</p>			machine learning training algorithm Dr. Pelletier	model with wave image filtering
Phase III Pilot-testing and revision	<p>Autonomous Scout Drone</p> <p>Flight test</p> <p>Image data collection test</p> <p>Ocean image collected: 11.8k images across 13.8km beach mileage</p> <p>Image analysis: Python Jupyter Notebook. RGB color channel analysis. Cross-wave analysis. Signal filtering to distinguish wave front and breaks. Spatial fourier transform for wave detection.</p> <p>ML model build: Spatial and temporal wave analysis ML model under construction.</p> <p>Image data training: Analytic model and ML co-training planned.</p> <p>Beach scout mission for shoreline wave image datasets for image analysis and machine learning</p>	Dec 2023-current	Del Mar Beach Home Backyard Tennis court	Fusion360 CNC machine 3D Printing Dr. Pelletier	Scout Drone flight testing completed Rescue drone Initial flight testing completed Continued ML training to enhance accuracy of the threat detection model

	<p>Omnidirectional Rescue Drone</p> <p>Pitch and roll test</p> <p>Flight test</p> <p>Redesign: Pitch arm fixture had to be redesigned as 3D printer precision was not sufficient. Leg holder 3D print was too loose and shallow and it collapsed eventually. CNC holes were too tight, and needed additional drilling before the corrections were applied in milling.</p> <p>3D printed, CNC-ed, and rebuilt revised parts and spare back-up parts.</p> <p>Demonstrated pitch and roll control</p> <p>Performed incremental test flights</p> <p>IEEE research manuscript (Kim, under review)</p>			Continued prototype revision
--	---	--	--	------------------------------

*The next phases (scale up manufacturing, operation and maintenance) are not part of the current research but are planned for future endeavors

Table created by the author

6.3. UAV design common components

Both scout and rescue UAVs share the common flight controller, AI-capable mission controller platform, communication radio links, and batteries. This section details design choices for lifeguard UAV system implementation.

6.3.1. Flight controller

Two major flight controller platforms available are PX4 Autopilot (PX4, 2024) and Ardupilot (Ardupilot, 2024). They are comparable in many aspects, and they even run on the same the flight controller, such as Pixhawk. Both are open source, and they have extensive configurations in various form factors. Table 13 compares PX4 and Ardupilot (Akshata, 2024).

Table 13. Comparison of PX4 and Ardupilot flight controller stack

	PX4	Ardupilot
Start	2013	2007
Hardware	Pixhawk, Pixracer, SnapDragon Flight	APM2, Pixhawk, PXF, ErieBrain
Firmware	PX4 Autopilot	Ardupilot
Ground control system	QGroundControl	APM planner 2, Mission planner, Tower, MAVProxy
Strength	Precision, modularity	Versatility, community support
Licensing	BSD	GPL

Table created by the author

PX4 was adopted for the lifeguard UAV development due to its expansion capability and continuity from last year's simulation in the loop utilizing PX4 and Gazebo simulators (Kim, 2023).

The Pixhawk 6X flight controller and expansion baseboard were used in the research. The PX4 flight stack supports the controller seamlessly, and it has extensive input and output ports to realize scout and rescue UAVs and periphery components. It also offers multiple MAVLink (MAVLink, 2024) connections to the mission controller and ground control station for control and logging. Table 14 lists Pixhawk 6X's functionalities (PX4, 2024). It has multiple sensors and Inertia Measurement Units (IMUs) to control UAV stably and dual GPS receivers for precise flight plan execution.

Table 14. Pixhawk 6X major functions

Item	Description
FMU Processor	STM32H753 32 Bit Arm® Cortex®-M7, 480MHz, 2MB flash memory, 1MB RAM
IO Processor	STM32F100 32 Bit Arm® Cortex®-M3, 24MHz, 8KB SRAM
On-board sensors	Accel/Gyro: ICM-20649 or BMI088 Accel/Gyro: ICM-42688-P

	Accel/Gyro: ICM-42670-P Mag: BMM150 Barometer: 2x BMP388
I/O ports	16- PWM servo outputs 3 telemetry serial ports 2 GPS ports Ethernet port CAN expansion bus ports

Table created by the author

6.3.2. AI mission controller

AI capability is essential for lifeguard UAVs to be autonomous. For example, there could be multiple scout UAVs across a wide range of ocean, and a localized AI capability would reduce communication demand. Also, scout and rescue UAVs will continue their mission even when the communication link loses connection to ground control station temporarily. Table 15 lists common embedded computing platforms. For general computing and control, Raspberry Pi has solid footage (Raspberry Pi, 2024). Whenever the machine learning is engaged and a large number of floating-point calculations are involved, higher-end embedded computers such as NVIDIA Jetson platform offer competitive AI performance (NVIDIA, 2024).

Table 15. Computing platform comparison

	Raspberry Pi 4B	Jetson Nano	Jetson Orin Nano
CPU	Quad core Cortex-A72 (ARM v8) 64-bit SoC	Quad-Core Arm Cortex-A57 MPCore processor	6-core Arm Cortex-A78AE v8.2 64-bit CPU 1.5MB L2 + 4MB L3
Clock (GHz)	1.8	1.43	1.5
GPU	Broadcom VideoCore VI	128-core NVIDIA Maxwell™ architecture GPU	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores
GPU clock (MHz)	-	921	625

Memory	8GB LPDDR4-3200 SDRAM	4GB 64-bit LPDDR4 25.6GB/s	8GB 128-bit LPDDR5 68 GB/s
Power max (W)	7.6	10	15
Power typ (W)	3.4	5	7
TFLOPS	0.014	0.236	1.28
Geekbench single core	259	230	566
Geekbench multi-core	681	819	2974
AI performance (TOPS)		0.472	20
SpecInt rate		16	106
Linpack double	10.7	912.26	
Linpack single	20		
Video encoding		2x 1080p60 1x 4K30	3-4x 1080p30
Video decoding		1x 4K60 (H.265) 4x 1080p60 (H.265)	1x 4K60 (H.265) 2x 4K30 (H.265) 5x 1080p60 (H.265) 11x 1080p30 (H.265)

Table created by the author

6.3.3. Communication links

Both scout and rescue UAVs need stable communication channels for control and safety.

Three levels of communication links are adopted at both designs as listed in Table 16. MAVLink telemetry radio has the first layer of data connection between ground control station and the flight controller. Ground control station can request most flight data updates through MAVLink, and it can also override control with the attached joystick pad attached to control station. IP radio established an IP network between UAV and ground control station. The network allows video streaming and remote login to the mission controller. As the mission controller has local MAVLink to flight controller, it can force an override to flight controller as necessary. The last radio is remote control, and it receives manual remote controller input directly. The flight controller overrides all inputs when the manual remote controller takes control. As a safety measure, the UAV can stay or return to base when it loses all radio connection by flight

controller. Also, mission controller can have a custom program to instruct flight controller for a certain behavior when all radio links are lost.

The radios listed in Table 16 use frequency bands requiring operation privileges. As a ham radio operator with Federal Communications Commission (FCC) Technician license holder (call sign N6SEA), my operation of the radios complies with FCC regulation.

Table 16. Lifeguard UAV communication radio links (SIYI, 2024)

Function	Connection	Model	Range (Km)	Frequency (GHz)
MAVLink telemetry radio	Ground control and flight controller	SIYI Datalink	15	2.4
IP and video radio	Ground control, mission controller, and IP video camera	SIYI HM30	30	5
Remote control radio	RC controller and flight controller	SIYI FM30	30	2.4

Table created by the author

6.3.4. Battery

UAV can utilize different battery chemistry to perform better missions. A scout UAV needs to extend flight duration for the continuity of the scout missions. A Li-Ion battery offers higher energy density per weight than the LiPo battery as compared in Table 17. For the same capacity and series stack, Li-Ion has about 39% more energy density than LiPo. It will help extend flight duration as much. One assumption is the UAV will not draw more current than its current C-rating. A rescue drone will need stronger power delivery to produce thrust for rescue missions. A 6S LiPo battery with a high C-rating will meet such requirement.

Table 17. Battery specification comparison (GETFPV, 2024), (Amazon, 2024)

Brand	Upgrade Energy Dark Lithium VS Li-Ion	Lumenier LiPo	HRB LiPo
Configuration	4S	4S	6S
Capacity (mAh)	4200	4200	3300
C-rating	25	35	60
Weight (g)	282	392	467
Energy density (mAh/g)	14.9	10.7	7.1

Table created by the author

6.4. Scout UAV design, development, and operation

The scout UAV in Figure 1 implements a quadcopter with AI capable mission controller and secures payload capacity with motor and propeller selection. It also has triple radio links to ensure control and fallback mechanisms. It complies with FAA regulations with remote ID transmitter, registration tags, and 4-mile visibility light strobes. Following subsections will go over details beginning with requirements.

Figure 1. The scout UAV quadcopter in action for a flight mission at Del Mar beach



Photograph taken by the author

6.4.1. Scout UAV implementation and system block diagram

The components discussed were assembled to build a scout UAV over the frame as described in functional blocks diagram Figure 2.

Figure 2. Scout UAV system block diagram

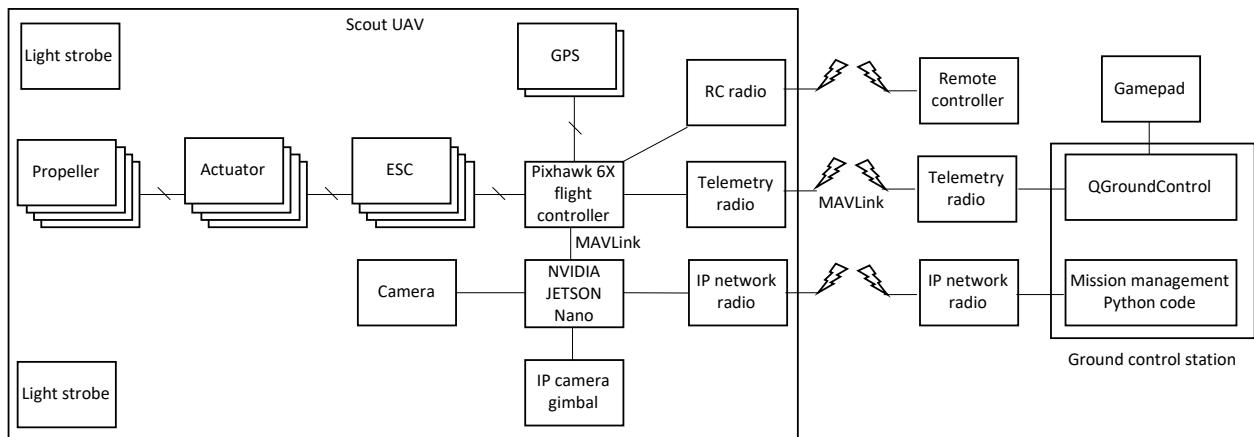
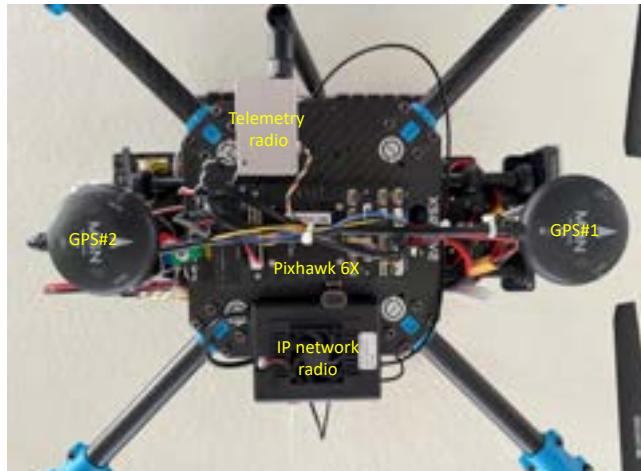


Diagram created by the author

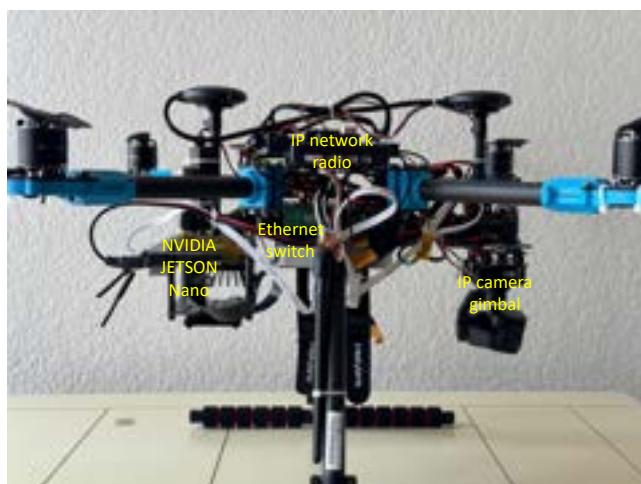
GPS receivers, gimbal camera, Raspberry Pi camera module 3, NVIDIA JETSON Nano, and flight controller were attached to the frame with mounts and screws. Other parts were fixed with double-sided tapes and zip ties. Wires were tied together to avoid flight interference. Placement and layout are shown in Figure 3, Figure 4, and Figure 5.

Figure 3. Top view of scout drone and components



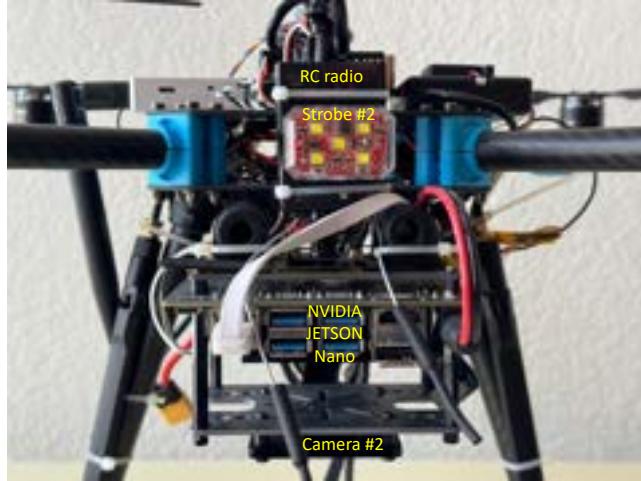
Photograph taken by the author

Figure 4. Side view of the scout UAV



Photograph taken by the author

Figure 5. Back view of the scout UAV



Photograph taken by the author

6.4.2. Motors and propellers

The scout UAV weighs about 2Kg without battery. The motor and propeller combinations should generate more than 3Kg payload thrust at 70% output. T-MOTOR AT2317 and APC11 provide thrust as listed in Table 18. The total thrust is about 4.6Kg at 70% throttle, and it should be able to carry additional battery capacity to extend flight duration.

Table 18. Scout UAV thrust with AT2317 and APC11 (T-MOTOR, 2024)

Throttle	Current (A)	Power (W)	Thrust (Kg)	Total thrust (Kg)
40%	4.8	71	0.59	2.4
45%	5.7	83	0.67	2.7
50%	6.6	97	0.74	3.0
55%	7.6	111	0.83	3.3
60%	8.9	130	0.93	3.7
65%	10.7	155	1.05	4.2
70%	12.4	180	1.16	4.6
75%	14.4	208	1.28	5.1
80%	16.7	240	1.40	5.6
90%	21.9	312	1.63	6.5
100%	23.4	332	1.67	6.7

Table created by the author

6.4.3. Mission management program

The mission controller can control flight controller through local MAVLink connection.

For the scout UAV beach data collection, multiple programs work together to facilitate beach image data collection as depicted at Figure 6. Several programs are initiated before scout mission starts by setting gimbal camera orientation, starting image collection from cameras, storing images, and streaming to ground control station. GPS coordinates and flight data are tagged to each image through a log file. Ground control station can login to NVIDIA Jetson Nano mission controller through Virtual Network Computing (VNC) to start programs during set up process. Ground control station starts to receive video streaming through IP network. All programs are listed as text file in Appendix E. Computer programs.

Figure 6. Scout UAV mission management system

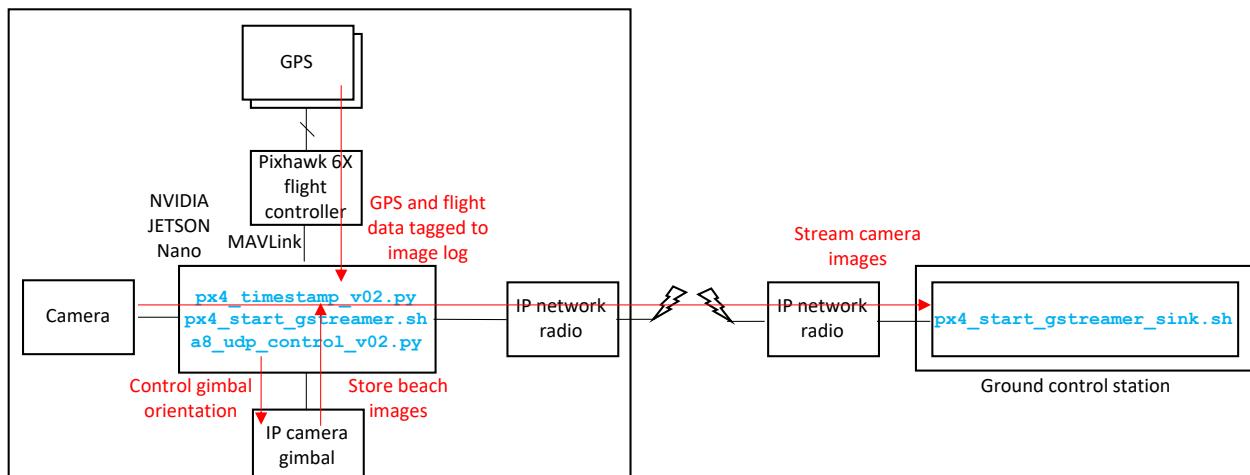


Diagram created by the author

6.4.4. Scout UAV operation procedure

The scout UAV was deployed at Del Mar beach to collect ocean image for offline analysis. A scout operation procedure was proposed and revised through the multiple test flights. Table 19 list the procedure checked during the data collection flight missions. It includes running the mission management programs through VNC login and alias.

Table 19. Scout UAV beach operation procedure

Phase	#	Items
Prepare	1	Checklist
	2	Drone
	3	Drone batteries, charged
	4	HM30 ground radio and batteries, charged
	5	2.4GHz datalink ground radio
	6	RC controller, charged
	7	Telemetry joystick, charged
	8	Folding table
	9	FAA and FCC documents
	10	Recoding camera (GoPro, iPhone)
	11	Linux laptop, charged
	12	Spare propellers
	13	Screw drive, hex wrench, zip tie
	14	Safety goggles or glasses
	15	QGroundControl offline map
Preflight	16	Set table
	17	Set laptop
	18	Set HM30 ground radio
	19	Set telemetry radio
	20	Secure surrounding safety boundary
	21	Start QGC
	22	Create flight plan
Flight	23	Install drone battery
	24	Power up drone, RC controller, ground radio
	25	Telemetry, data feed, and RC check
	26	Ping check from laptop (192.168.144.xx) 12 - Ground unit, 11 - Air unit 20 - JETSON, 25 - IP camera
	27	Upload flight plan
	28	VNC login to JETSON (192.168.144.20)

	29	Set recording folder
	30	Start JETSON recording ("rung")
	31	Start JETSON image GPS tagging ("runp")
	32	Start laptop video stream display command ("runs")
	33	Power on strobes
	34	Confirm safety
	35	Arm actuators
	36	Start flight path
	37	Land drone
	38	Disarm actuators
	39	VNC login to JETSON and shutdown ("shut")
	40	Power off drone and strobes
	41	Disconnect battery
Postflight	42	Maintain drone
	43	Check images recorded
	44	Review procedure

Table created by the author

6.5. Rescue UAV design and build

The Rescue UAV in Figure 7 implements coaxial hexacopter for thrust and actuator redundancy. It has a novel in-arm pitch axis design to increase maximum pitch angle by 65% from conventional end-arm pitch axis design. The design achieves ± 45 -degree pitch angle in addition to ± 180 -degree roll angle. It has the most pitch angle reported so far. With the extended and symmetric pitch angle, the UAV should have omni-directional orientation and flight capability.

Figure 7. Rescue UAV under preflight check

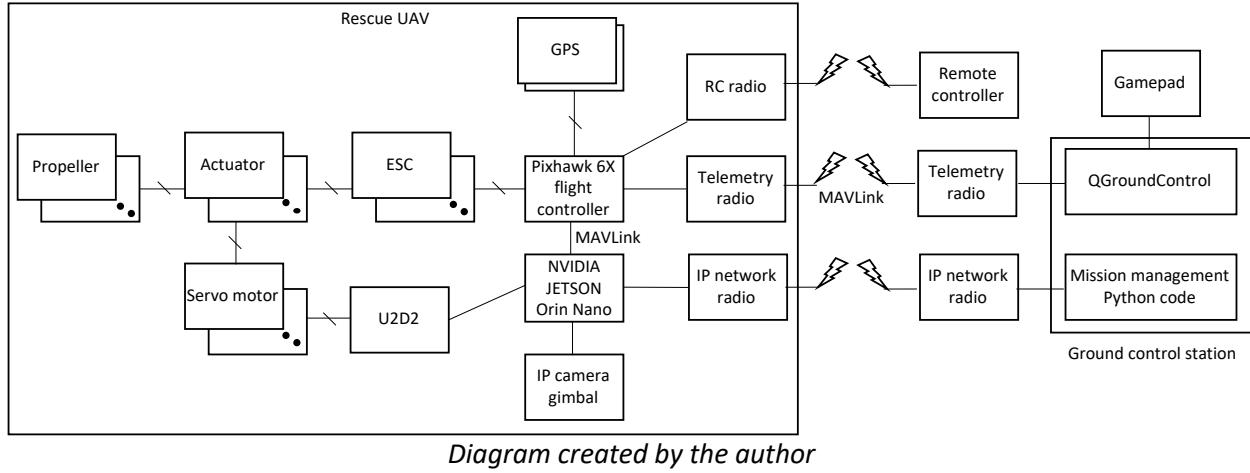


Photograph taken by the author

6.5.1. Rescue UAV system block diagram

The rescue UAV block diagram is similar to the scout UAV from function block perspective as shown in Figure 8. It has more actuator and propeller pairs for coaxial hexacopter configuration and servo motor control for pitch and roll tilt rotor by mission controller. The block diagram shows baseline configuration, and additional function blocks can be readily added to bring up full capability. One example is an inflatable flotation device and rope with trigger to release the device.

Figure 8. Rescue UAV function block diagram



6.5.2. Extended and Symmetric In-Arm Pitch Axis

A novel in-arm pitch axis design is developed in this research to extend pitch angle. As summarized in Table 1, previous pitch tilt rotor UAVs use end-arm pitch axis. The pitch angle for coaxial UAVs is limited to 20 degrees. One extreme design with bent arm is not practical, and it is not applicable to coaxial design.

The conventional end-arm pitch axis design and the novel in-arm pitch axis design are depicted in Figure 9 and Figure 10. By shifting pitch axis inward on the arm, actuator and propeller height increases when the pitch angle approaches arm and body. As a result, effective pitch angle increases by up to 65%.

Figure 9. Conventional end-arm pitch axis and tilt angle

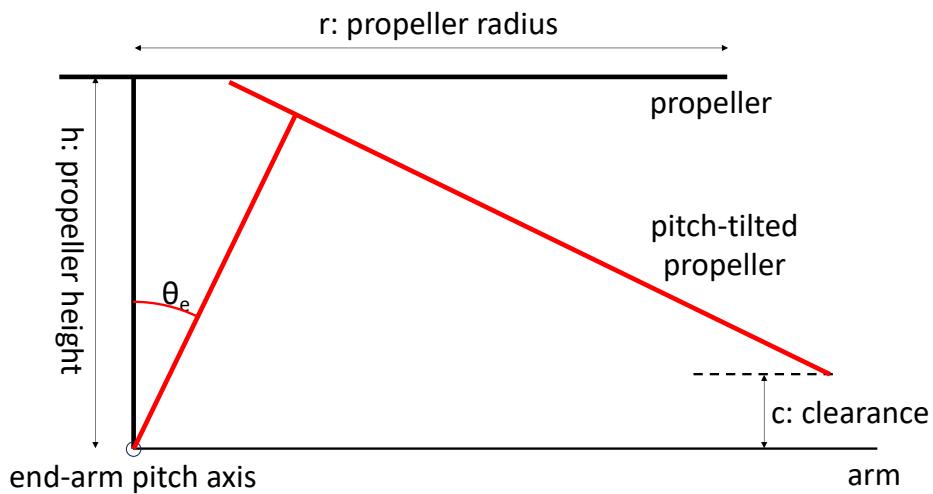


Diagram created by the author

Figure 10. Proposed novel in-arm pitch axis and tilt angle

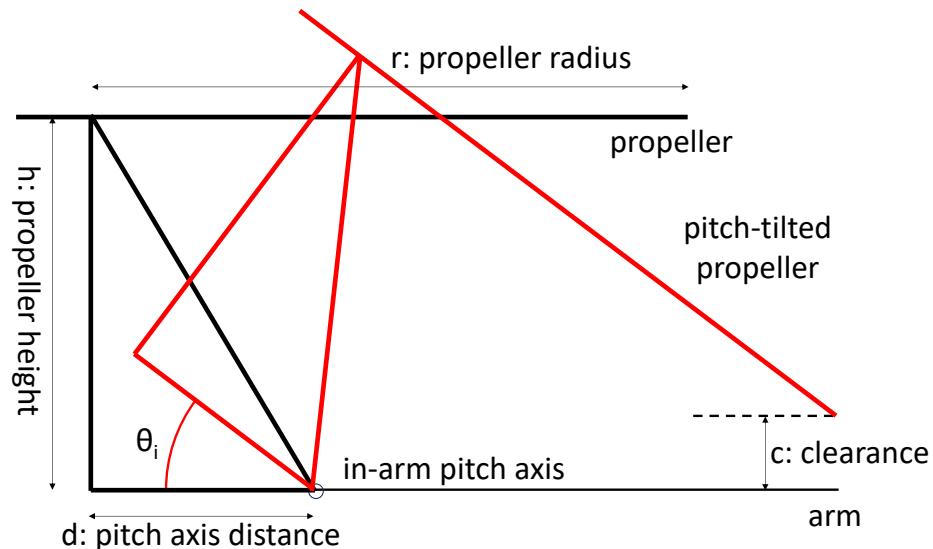


Diagram created by the author

The maximum pitch angle for end-arm pitch is calculated as (1):

$$\theta_e = \tan^{-1} \left(\frac{h}{r+c} \right) \quad (1)$$

The maximum pitch angle for in-arm pitch is solved with (2):

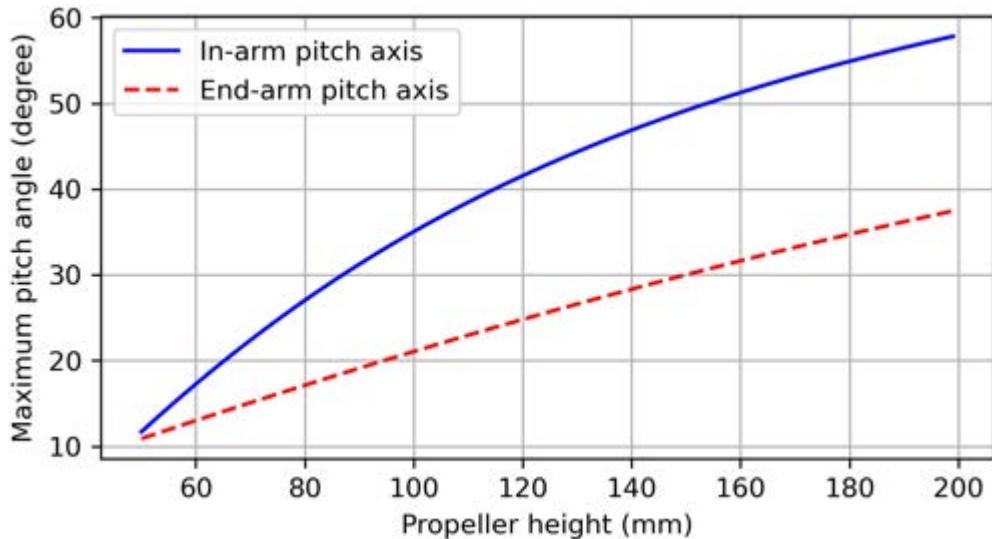
$$h \cos \theta_i + d \sin \theta_i = c + r \sin \theta_i \quad (2)$$

Using trigonometric identify and a quadratic equation, the maximum pitch angle is solved as (3), where $k = r - d$.

$$\theta_i = \sin^{-1} \left(\frac{-k + \sqrt{c^2 k^2 - (k^2 + h^2)(c^2 - h^2)}}{k^2 + h^2} \right) \quad (3)$$

Using (1) and (3), the maximum angles of the end-arm and in-arm pitch axes are compared in Figure 11. The propeller diameter is r is 9in, or 229mm, and the in-arm design used axis shift $d=140$ mm for the plot. If propeller height h is 140mm, the in-arm design has 46.9-degree maximum pitch angle, and the end-arm design has 28.3-degree maximum pitch angle. The in-arm design offers 65% more pitch angle than the end-arm design.

Figure 11. Maximum pitch angles with end-arm and in-arm pitch axes

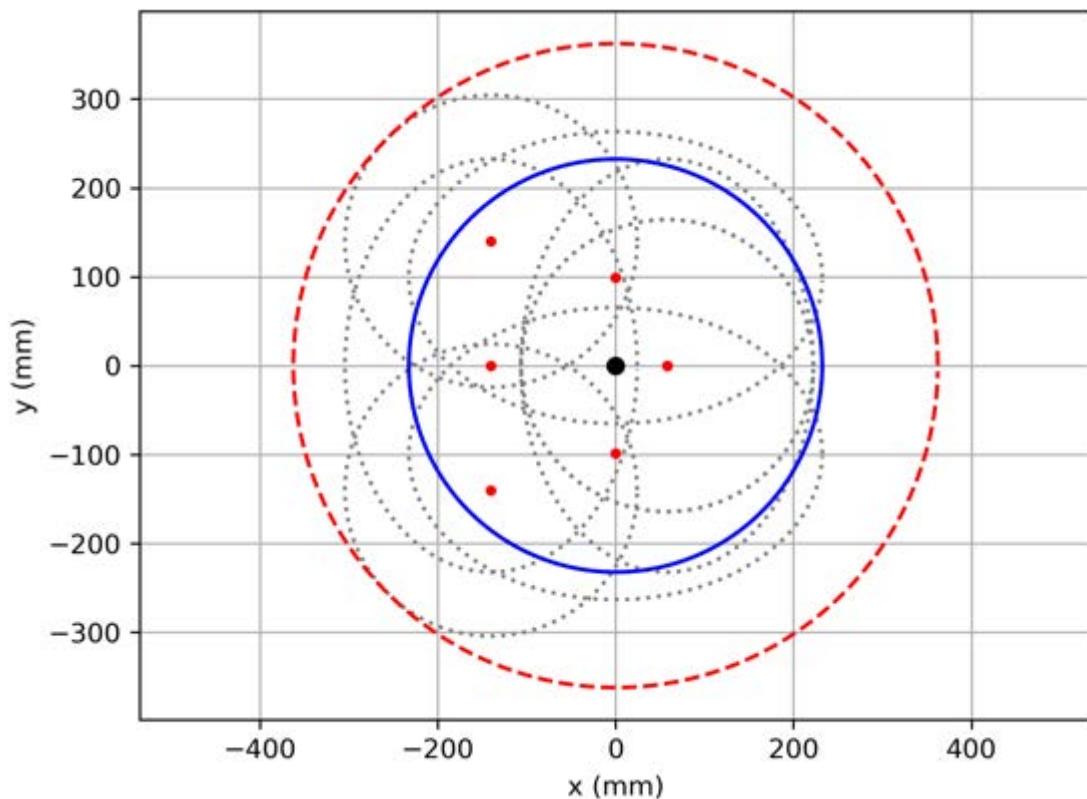


Plot created by the author

6.5.3. Actuator and propeller arrangement

Using in-arm pitch axis and roll, the actuators and propellers have more freedom to direct thrust to achieve desired control. At the same time, the propeller travels beyond the fixed rotor trajectory. As a result, the worst-case propeller trajectory should be considered to define spacing between propellers. Figure 12 shows the propeller path when pitch and roll angles are applied with the same set up as Figure 11. The red dash line is the maximum radius that propeller goes through, and the radius is 362mm. For omni-direction flight, a design will need to guarantee full freedom on pitch and roll angles of each arm to controller. Otherwise, propellers might collide at extreme tilt rotor angles.

Figure 12. Propeller trajectory with pitch and roll angle.



Plot created by the author

6.5.4. Actuator and propeller

The rescue UAV total thrust should be larger than 40Kg as defined in 6.1. T-MOTOR MN5008 KV340 motor and P18*6.1 18-inch propeller combination is used for the rescue UAV as assembled in Figure 13.

Figure 13. Top actuator and propeller assembled on the pitch block



Photograph taken by the author

Multiple actuators are put in coaxial hexacopter configuration for redundancy. The other approach could be using stronger thrust with a smaller number of actuators and larger propellers. However, they tend to require higher voltage battery and electronics. Also, smaller propellers are favored just in case for safety concerns. As calculated in Table 20 (T-MOTOR, 2024), the motor and actuator produce 22.3Kg total thrust at 60% throttle for a takeoff with 12.3Kg weight and additional lifesaving equipment, such as an inflatable flotation device. The bottom actuator and propeller produce about 25-35% less thrust in coaxial configuration due to turbulence. If the design has enough coaxial distance ratio Z/D , where Z is the distance between propellers and D is propeller diameter, the bottom actuation efficient will improve (Tyto Robotics, 2024). For the total thrust calculation, 70% efficiency was assumed for the bottom propellers as previously estimated (Tyto Robotics, 2024). To support thrust, UAV's current capacity should accommodate the expected current consumption.

Table 20. Rescue UAV thrust calculation with actuator and propeller

Throttle	Current (A)	Thrust (Kg)	6+6 total thrust (Kg)	Total current (A)
----------	-------------	-------------	-----------------------	-------------------

40%	3.8	1.0	10.5	45
45%	5.1	1.2	13.0	61
50%	6.9	1.5	16.2	83
55%	9.0	1.8	19.1	107
60%	11.1	2.1	22.3	133
65%	13.5	2.4	25.1	162
70%	16.0	2.7	28.1	192
75%	18.8	3.0	31.1	226
80%	22.0	3.3	34.3	264
90%	29.5	3.9	41.0	354
100%	33.5	4.2	44.1	402

Table created by the author

6.5.5. Servo motor for pitch and roll tilt rotor

For pitch and roll control of six coaxial rotors, twelve Dynamixel XL430-W250-T servo motors are arranged on the arm as assembled in Figure 14 (Robotis, 2024). The mission controller controls the servo motors through U2D2 USB communication control hub, and a shared serial communication link.

The motor's stall torque is 1.5N*m, or 15.3 Kg*cm. Considering the total thrust and the thrust is distributed to arms, the motor should be able to keep its position against torque applied to the servo motor during flight. The motor also needs up to 1.4A current, and power supply network should consider current capacity for all twelve motors.

Table 21. Specification of Dynamixel X430-W250-T servo motor

Item	Specification
Model	XL430-W250-T
MCU	Cortex-M3 72MHz, 32bit
Input voltage (V)	6.5 - 12.0
Stall torque (N*m)	1.5
Stall current (A)	1.4
Resolution (degree/pulse)	0.0879

Resolution step (pulse/rev)	4,096
Position sensor	Contactless absolute encoder, 12 bit AMS AS5601
Weight (g)	57
Gear ratio	258.5:1
ID	0 - 252
Protocol	Half duplex asynchronous serial communication 8 bit, 1 stop, no parity
Operating mode	Velocity Position Extended position PWM

Table created by the author

Figure 14. Roll and pitch drive servo motors assembled



Photographs taken by the author

6.5.6. In-arm pitch and roll design

The in-arm pitch axis and roll schemes were designed with Autodesk Fusion 360, and ± 45 -degree pitch tilt along roll tilt are performed at

Figure 15 and

Figure 16.

Figure 15. In-arm pitch -45-degree coaxial rotor tilt CAD view



Images created by the author

Figure 16. In arm pitch +45-degree coaxial rotor tilt CAD view



Images created by the author

The in-arm pitch angle is driven by a servo motor mounted at the end of the arm and an inner gear along the actuator axis as shown in Figure 17. 2mm carbon fiber plates hold the servo motor, and pitch block is guided by axis and guides at front and back of the servo motor. The front guide sits between side plate and the inner gear to keep the pitch angle plane maintained. A few screenshots removed parts blocking a view to show the gear engagement. The pitch axis has 2mm inner diameter and 5mm outer diameter ball bearings and stoppers to support the pitch

rotation. The inner gear has a capture hook at the end of the gear to keep the servo spur gear engaged with the inner gear if the spur gear exceeds the gear range. The inner gear to spur gear ratio is 306:12.

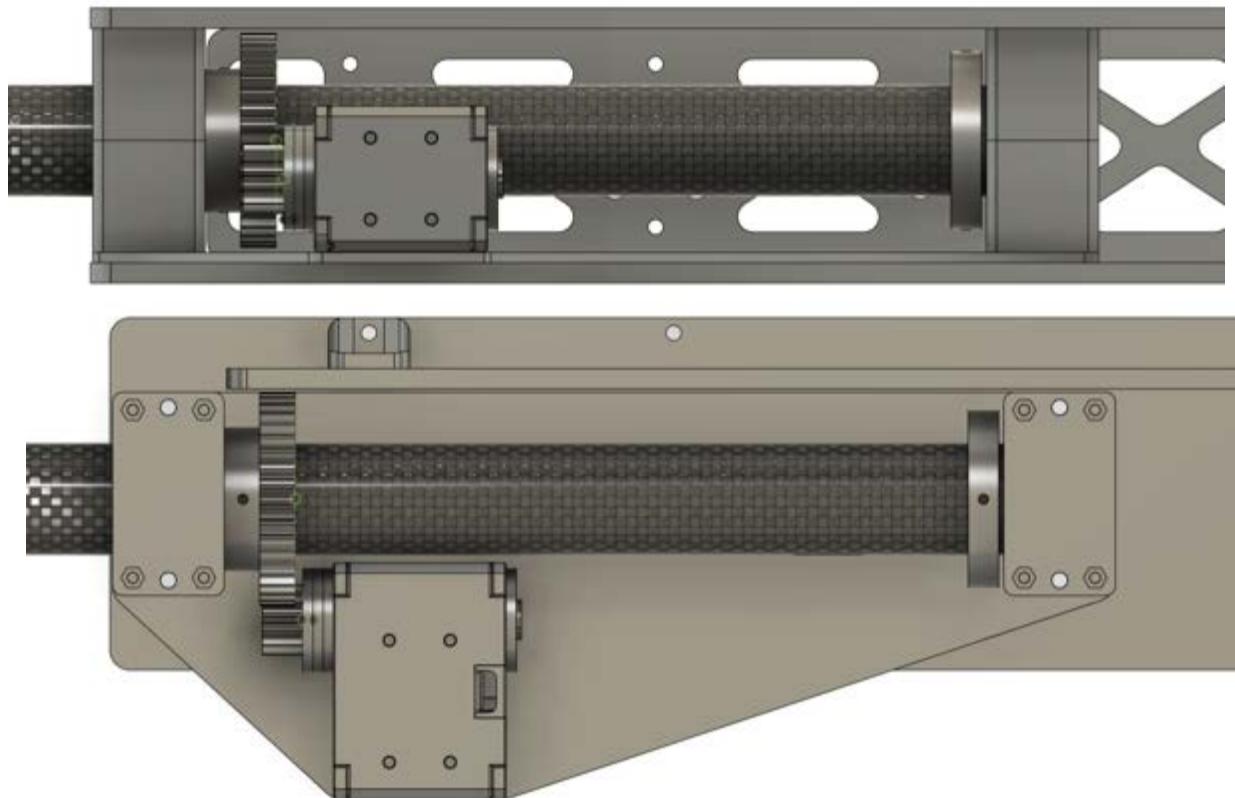
Figure 17. In-arm pitch block design and views with servo motor and inner gear



Images created by the author

The arm roll angle is maintained by the second servo motor along the arm axis as shown in Figure 18, where a few parts were hidden to reveal the gears. The arm is held by two holding blocks and 22mm inner diameter and 28mm outer diameter roller bearings. The arm pitch and roll block is modular and can be folded and detached for maintenance. The roll spur gear to servo motor spur gear ratio is 44:12.

Figure 18. Arm roll angle drive design and servo motor



Images created by the author

6.5.7. Component layout

Most electronics components are placed in the body frame, whose opening is shown in Figure 19. The power distribution board (PDB) is at the center of the body and receives battery power from the below. In Figure 20, twelve electronic speed control's (ESCs) are placed around PDB, and their wires are within 12cm of the power distribution board to avoid additional capacitors. Each ESC has three wires connected to an actuator, and each arm has two sets of ESC wiring connecting two ESCs to two brushless motors. ESC wirings are routed through the arm frame, then move out around the roll servo block, zip-tied to the arm tube. Also, ESCs are connected to Pixhawk PWM outputs at top body plate.

Figure 19. Body compartment opening for component placement

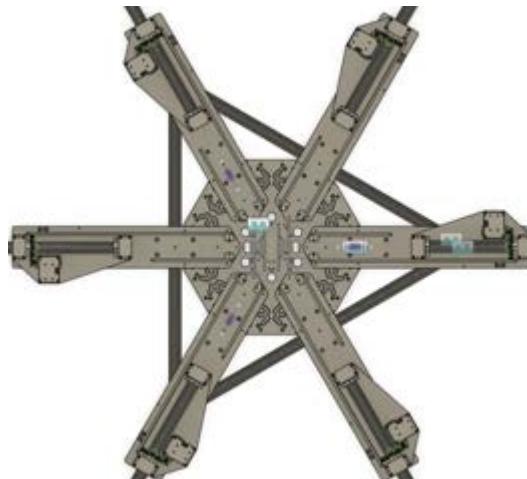
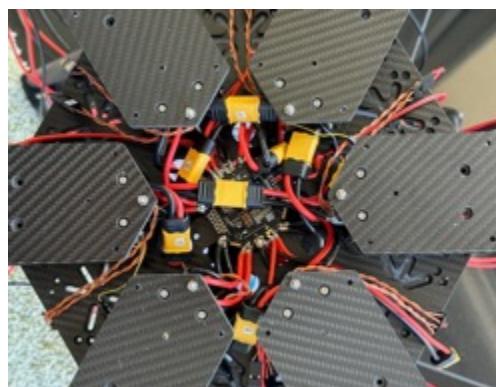


Image created by the author

Figure 20. Power distribution board and ESC routing



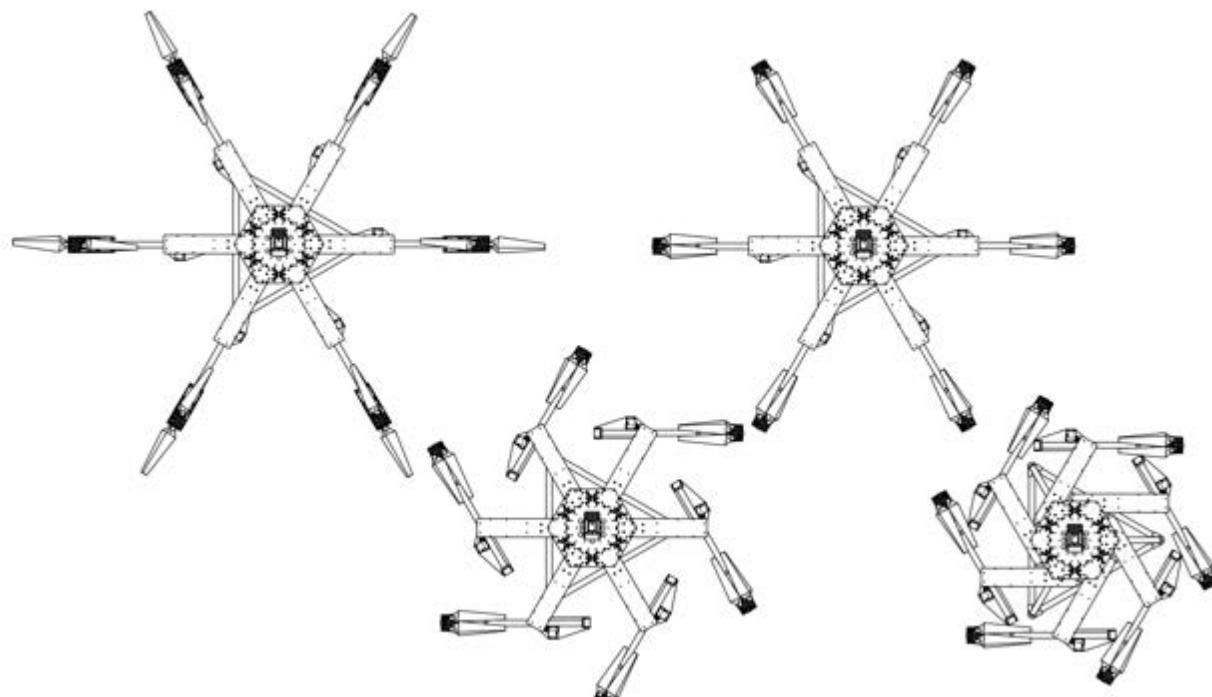
Photograph taken by the author

Additionally, the servo motor controller (U2D2 communication converter) and power supplier are centrally placed at the bottom of top body plate. Six three-wire control and power connectors are connected to a hub and the U2D2 communication converter. U2D2 connects to the mission controller on the top of the body frame. Pixhawk 6X flight controller sits at the center of the top body plate. All twelve ESCs, GPS, MAVLink, telemetry, and additional I/O wires connect to flight controller.

6.5.8. Rescue UAV design for folding

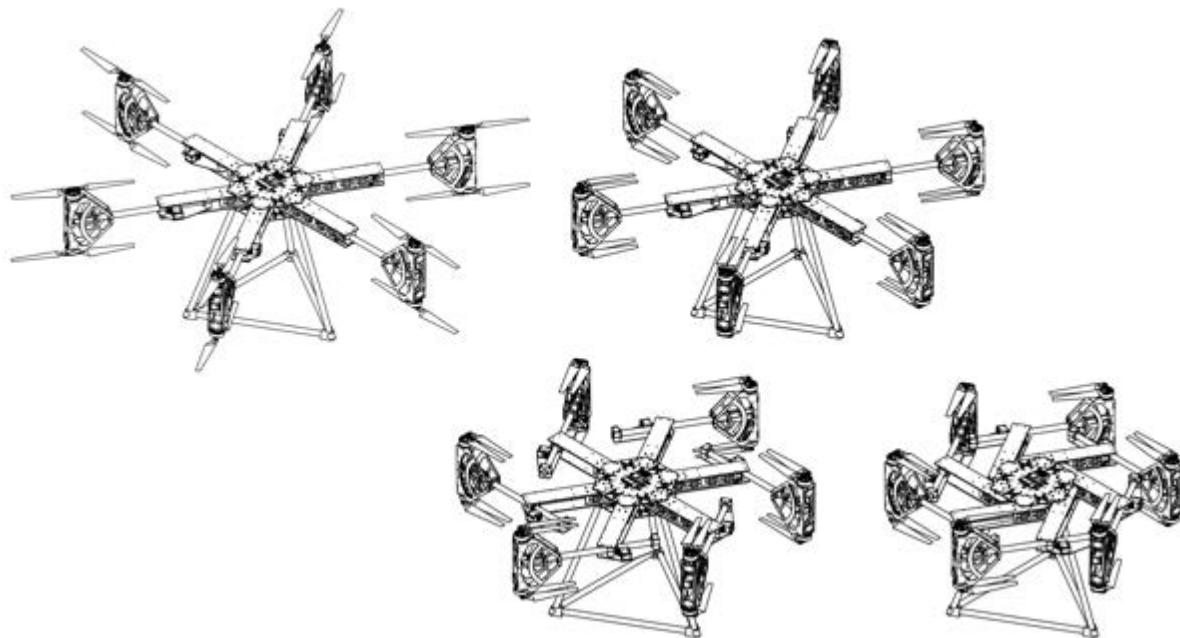
One of the rescue drone requirements from Table 11 is portability for transportation. The rescue drone wingspan is 2m to implement coaxial hexacopter configuration and thrust demand. The design provisioned folding to allow its transportation in passenger vehicles as shown in Figure 21 and Figure 22. Propellers are folded, pitch and roll tilt blocks are folded by 60 degrees, and arm frames are folded by 60 degrees around the body. The folded span is about 1.1m, and it is suitable for transportation with passenger vehicles.

Figure 21. Rescue UAV folding steps top view



Images created by the author

Figure 22. Rescue UAV folding steps perspective view



Images created by the author

To fold arm and arm frame, wiring needs planning to allow such movement after assembly. Wiring from ESC to actuator and from servo hub to servo motors should have additional run length to support folding as measured in Figure 23.

Figure 23. Actuator and servo motor wiring plan with folding

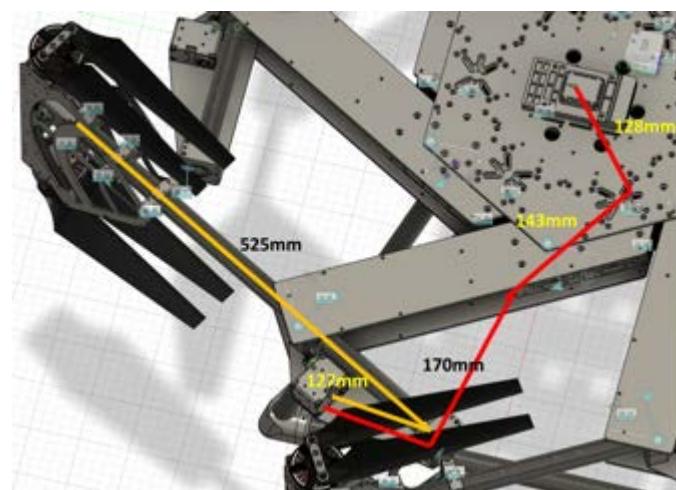


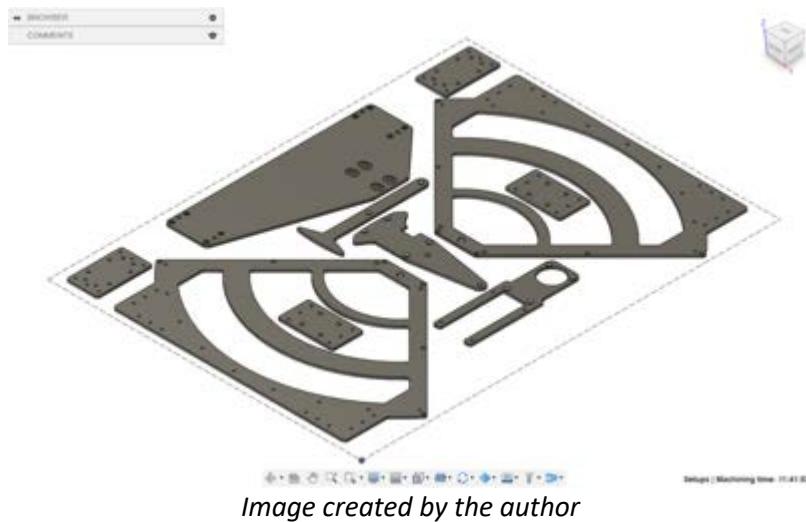
Image created by the author

6.5.9. Material fabrication process

The rescue UAV has been built from scratch. The structure support parts were CNC-milled from carbon fiber plates, and the connecting parts were 3D-printed out of Nylon.

Designed parts in Table 4 were laid out as templates as shown in Figure 24 to batch process the parts and to reduce machine and process set up overhead. Then they were repeatedly fabricated through CNC milling process. Portfolio of CNC templates used in the project are arranged in Appendix D. Fabrication templates.

Figure 24. An example CNC milling template for 300x400x2mm carbon fiber plate



After several attempts, CNC fabrication process was established as listed in Table 22. A carbon fiber board was placed on a waste board, then in a baking sheet. The boards were attached with double-sided tapes. Boards were put under water after Z-coordinate calibration.

Table 22. Water-immersed CNC milling process

#	Description
1	Cut 1mm plexiglass board to carbon fiber plate size
2	Remove tapes from carbon fiber plate
3	Attach double-sided tables to carbon fiber plate
4	Remove backing paper from double-sided tape
5	Remove cover from plexiglass board
6	Attach plexiglass board to double-sided tape
7	Remove cover from plexiglass board
8	Attach double-sided tables to carbon fiber plate
9	Remove backing paper from double-sided tape
10	Wipe clean baking sheet and dry
11	Align waste board with carbon fiber plate on baking sheet
12	Align baking sheet to CNC axis
13	Clamp baking sheet
14	Mount CNC end mill
15	Power on CNC controller
16	Perform X and Y position calibration
17	Perform Z calibration
18	Load G-code to CNC control program
19	Turn on spindle cooling pump
20	Pour water into baking sheet
21	Start CNC milling
22	Install side block panel

Table created by the author

Figure 25 shows water-immersed CNC milling set up and progress. Figure 26 shows a completed CNC milled carbon fiber board from the template in Figure 24. The template has all the 2mm parts for single arm assembly.

Figure 25. Water-immersed CNC milling in progress



Photograph taken by the author

Figure 26. Water-immersed CNC milled carbon fiber board



Photograph taken by the author

Table 23 lists CNC milling templates, number of times each part was fabricated, and number of hours carbon fiber boards were processed. A total of 16 carbon fiber boards were processed over 225 hours.

Table 23. CNC templates and fabrication time

#	Description	Plate size (mm ²)	Thickness (mm)	Mill bit	Duration (hour)	# of plates	Extended hours
1	Pitch and roll parts	300 x 400	2	1.8	11.7	8	93.6
2	Pitch and roll 3D parts	300 x 400	2	1.8	15.5	1	15.5
3	Pitch and roll 3D parts	200 x 300	2	1.8	7.6	1	7.6
4	Arm frame horizontal	300 x 400	4	2.5	10.0	3	29.9
5	Arm frame vertical long	300 x 400	4	2.5	26.6	1	26.6
6	Arm frame vertical short	300 x 400	4	2.5	12.0	1	12.0
7	Body plate top Arm frame vertical short	300 x 400	4	1.8	21.5	1	21.5
8	Body plate bottom	300 x 400	4	1.8	19.9	1	19.9
	Total					17	226.7

Table created by the author

Connection components and complex 3D parts were fabricated through 3D printing.

Nylon was the preferred filament for mechanical strength necessary for the assembly, movement, and flight under stress. I assumed Nylon would be strong enough to construct a prototype for conceptual demonstration, rather than regular deployment. In fact, the servo motor spur gear found itself not strong enough to deliver torque to roll and pitch gears. As a temporary solution, the spur gear has self-threaded flat screws at the core of the gear to strengthen the gear.

Figure 27. 3D-printed servo motor spur gear reinforced with core screws



Photograph taken by the author

Parts listed in Table 5 were arranged as templates as shown in Figure 28. The template has gears and arm holders with brim to support 3D printing stably. Comprehensive 3D print template portfolio is at Appendix D. Fabrication templates.

Figure 28. An example 3D-printing template for rescue UAV custom parts

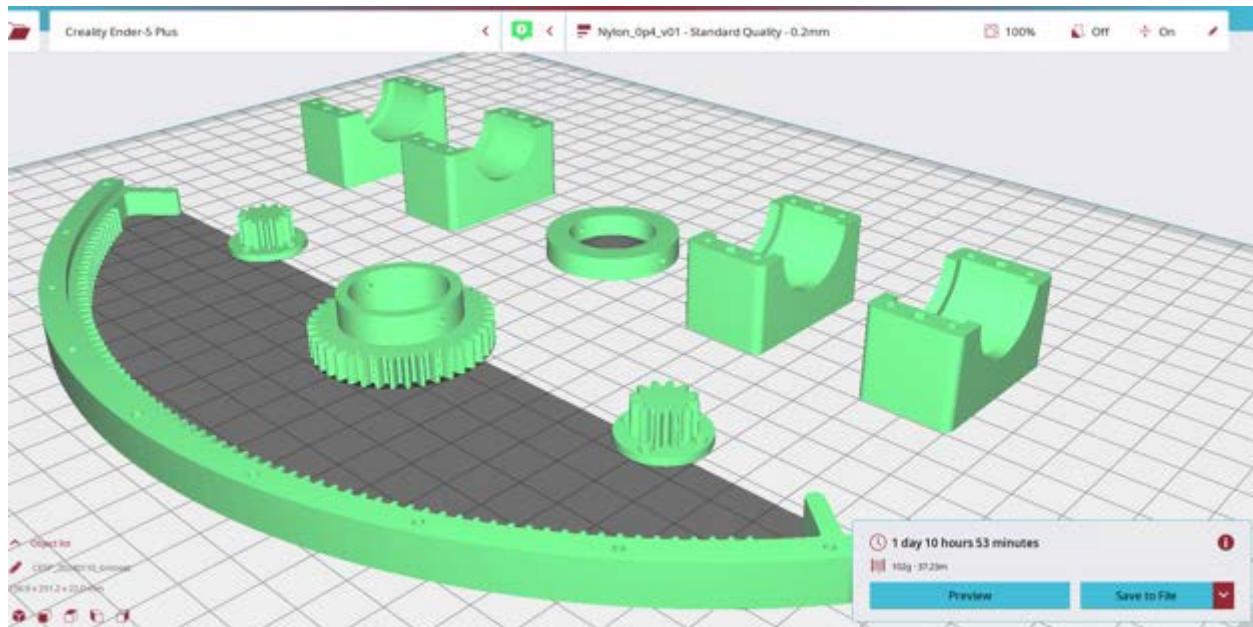


Image created by the author

After experimenting nozzle diameter, temperature, hot bed temperature, fill ratio, brim, layer thickness, etc., 3D-printing steps were decided as in Table 24. The dry box step was essential to prepare filament and stable 3D printing. Figure 29 shows several 3D-printed custom components. The brim was necessary for gear teeth to adhere to print bed.

Table 24. 3D printing process steps

#	Description
1	Dry filament reel at least for 12 hours
2	Clean print area with alcohol wipe
3	Apply adhesive glue to print area
4	Dry glue
5	Power on the 3d printer
6	Set nozzle temperature at 250C and print bed at 50C
7	Load filament to step motor
8	Load filament to nozzle
9	Clean up extruded filament
10	Calibrate Z height
11	Ensure nozzle height is less than 0.2mm with gauge
12	Perform leveling
13	Load mini SD card to card reader
14	Select G-code and start 3d print

Table created by the author

Figure 29. 3D-printed rescue UAV custom components



Photograph taken by the author

3D printed parts went through multiple revisions to improve fitting, assembly, and durability. Table 25 list major 3D printing templates and their print time approximated from slicer. More than 470 3D-printing hours were estimated to produce more than 300 custom parts.

Table 25. 3D print templates and printing hours

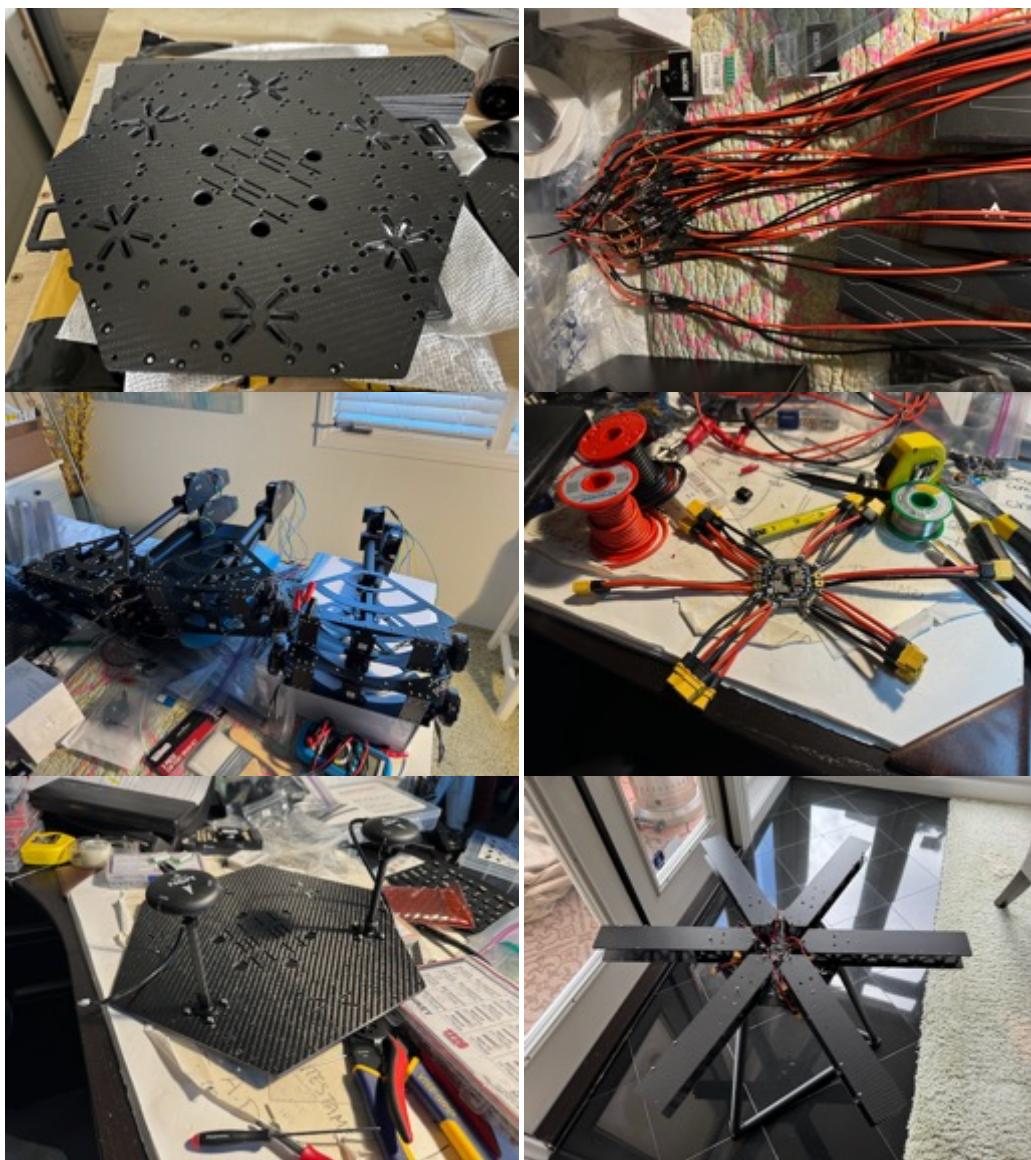
#	Description	# of parts	Weight (g)	Length (m)	Duration (hour)	# of print	Sum of hours
1	Leg holder v3	2	137	50.0	16.0	2	32.1
2	Leg holder v2	2	60	21.8	9.2	2	18.4
3	Motor and ESC bracket	24	37	13.4	13.0	3	39.0
4	Shoe holder	2	56	20.5	10.3	2	20.5
5	Pitch axis end	24	8	2.9	2.9	1	2.9
6	JETSON Orin mount	8	34	12.2	11.2	1	11.2
7	Pitch axis fixture	5	31	11.5	11.0	1	11.0
8	Arm roll gear	4	36	13.2	11.9	1	11.9
9	Leg and shoe holder v1	2	40	14.5	7.8	3	23.5
10	Arm frame bracket	25	28	10.3	10.1	4	40.3
11	Shoe holder v0	3	84	30.7	15.5	1	15.5
12	Leg holder version 0	3	39	14.1	8.1	1	8.1
13	Arm roll holder	4	42	15.4	9.5	5	47.5
14	Gear set	5	44	16.2	14.7	6	88.1
15	Arm fixture set	20	49	17.8	17.5	6	104.7
16	Servo motor riser and spur gear set	6	8	3.0	2.4	2	4.9
	Total						479.4

Table created by the author

6.5.10. Building up

The in-arm pitch and roll blocks were assembled to confirm feasibility. Body assembly involved wire preparation and soldering them to power distribution board and connectors. Leg and shoe holders needed re-design with tighter diameter to fit the 22mm tube for stable frame support.

Figure 30. Rescue drone build up progress photos





Photographs taken by the author

7. Results and Statistical Analysis

7.1. Scout UAV ocean image collection

The scout UAV has performed nine ocean image collection flight missions at Del Mar beaches between October and December 2023. Table 26 summarized all scout missions. Two missions did not collect any images due to warnings in mission controller or flight controller sensor. The scout UAV flew total 13.4km to collect 11.9k images out of the seven missions. Altitude was raised to 100m and the frame rate was also raised to 10 frames per second with later missions. 50m does not have wide enough coverage to analyze waves. 100m provided $175 \times 85 \text{mm}^2$ view, and it showed a worthy tradeoff between resolution and range.

Table 26. Scout UAV ocean image data collection missions

Flight #	Date	Time	Duration	Distance (km)	Altitude (m)	Take off	Frame per second	# of images	Data size (MB)
1	10/28/23	7:22	0:04:40	1.03	50	18th St.	1	0	0
2	10/28/23	7:39	0:04:48	1.04	50	18th St.	1	289	99.3
3	11/17/23	16:33	0:03:22	0.53	100	18th St.	1	88	27.9
4	11/17/23	16:42	0:06:45	1.52	100	18th St.	1	0	0
5	11/17/23	17:00	0:06:33	1.50	100	18th St.	1	299	58.5
6	11/22/23	8:19	0:08:38	2.09	100	27th St.	5	1950	499
7	11/22/23	9:01	0:07:27	1.75	100	18th St.	5	1660	441.5
8	12/25/23	9:43	0:08:29	2.04	100	27th St.	10	3995	2120
9	12/25/23	10:14	0:07:52	1.90	100	15th St.	10	3600	1900
Total				13.4				11881	5146

Table created by the author

Ocean data collection flight plans were set with QGroundControl (QGroundControl, 2024) as shown in Figure 31. It defines altitude, yaw, and speed of the UAV. Offline map was downloaded in advance to QGroundControl, and the plan was adjusted on site with fine tuning.

Figure 31. Ocean image collection flight plan

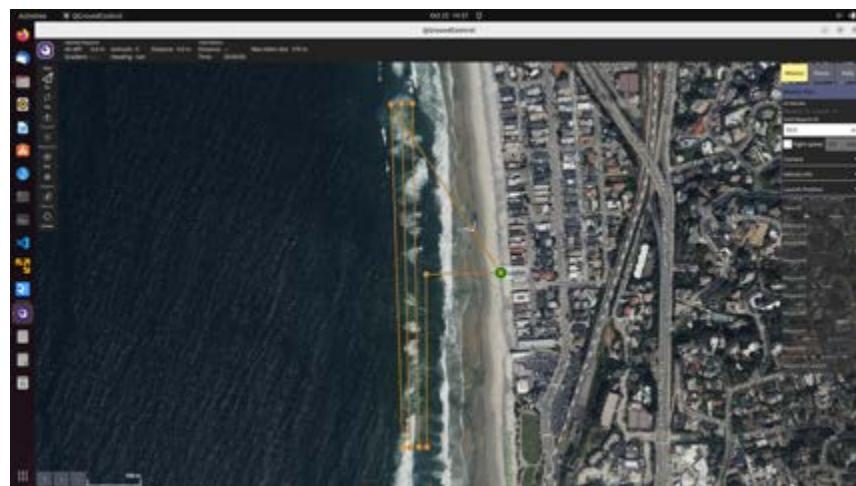


Image captured by the author

A mobile ground control station was set on a folding table as shown in Figure 32. Ground control laptop and radios were set on the table. Fine beach sand were avoided in wires and connectors, since fine sand causes unstable contacts. After the scout UAV powers up, IP network is used to start mission management programs through VNC log in.

Figure 32. Mobile ground control station during scout mission



Photograph taken by the author

Scout flight was planned when the tide was receding, as rip currents occur more at low tide (DUSEK, 2024). National Oceanic and Atmospheric Administration's tidal forecast chart at La Jolla beach was used to plan such tides in addition to weather conditions (NOAA Tides and Currents, 2024). An example tide chart is at Figure 33. San Diego faced King Tides on December 25, 2023, and two scout missions were conducted to record ocean images. All tidal charts used for scout missions are listed in Appendix B. Tidal charts.

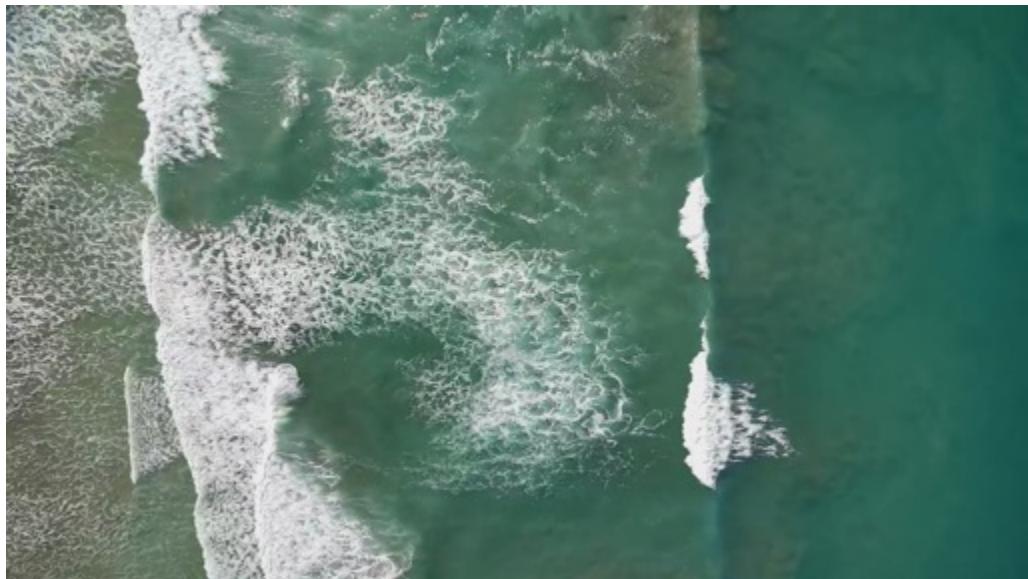
Figure 33. Example tide chart from NOAA used for scout planning



Image captured from NOAA (<https://tidesandcurrents.noaa.gov/>)

Different ocean tide conditions yield contrasting ocean water current views as compared in Figure 34 and Figure 35. Active tidal waves cause more dynamic behavior in wave breakage and back flow is stronger, which is more visible as sediment follows such flow. The ocean image snapshots from all missions are in Appendix C. Images from Scout Missions.

Figure 34. Ocean image during King Tide on 12/25/2023



Photograph taken by the author

Figure 35. Ocean image collected on 11/22/2023

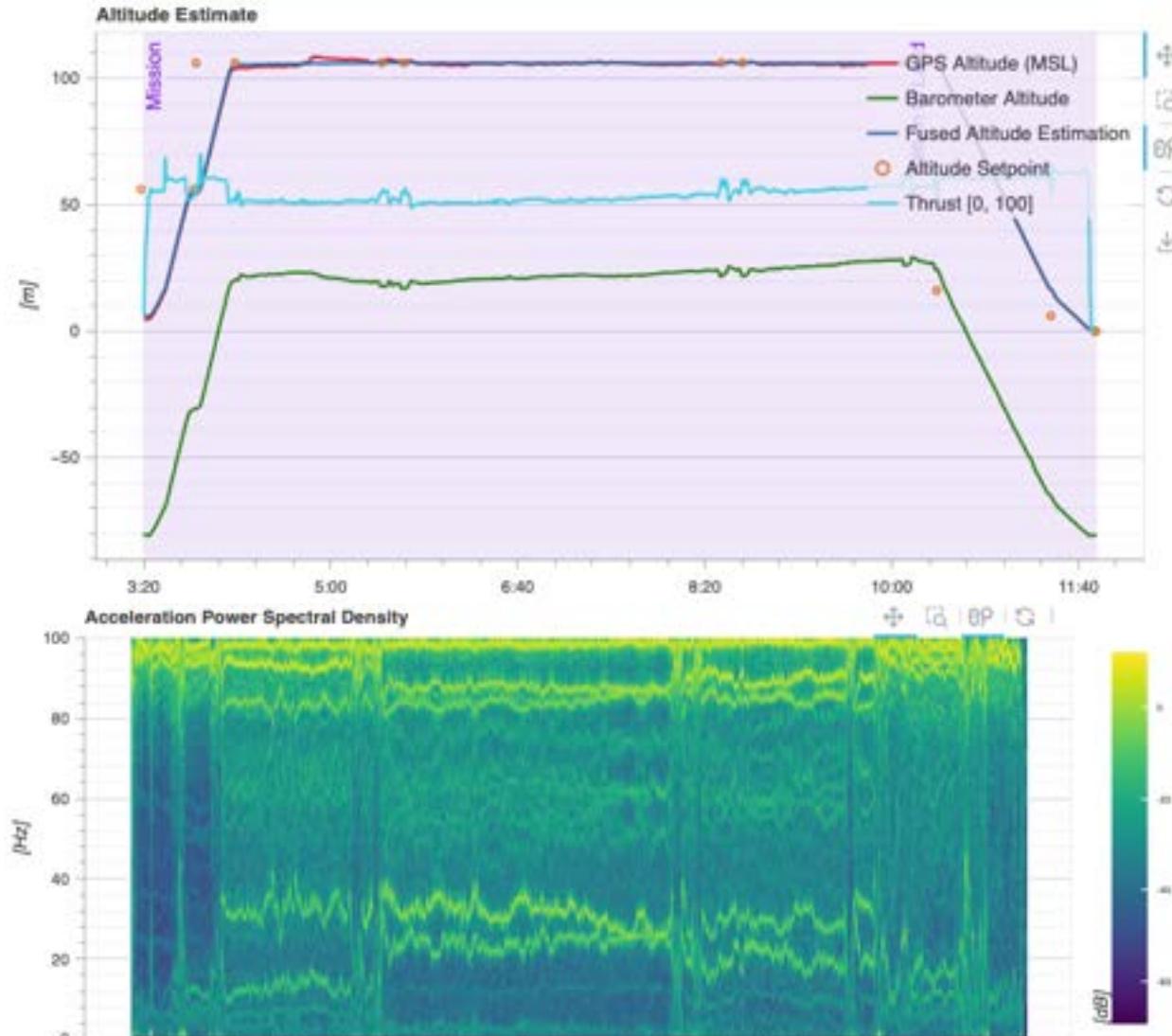


Photograph taken by the author

7.2. Scout UAV flight analysis

PX4 firmware records Pixhawk 6X's flight data to micro flash card for later analysis. A web-based PX4 Flight Review tool provides a quick review of flight dynamics when the log is uploaded (PX4 Flight Review, 2024). As shown in Figure 36, it covers basic flight information such as distance, altitude, and speed. Also, multi-copter specific dynamics information sets are analyzed to understand the control issues. Full flight analysis data sets for all scout missions are attached in Appendix F. Flight reports.

Figure 36. PX4 Flight Review flight log analysis plots from scout mission #9



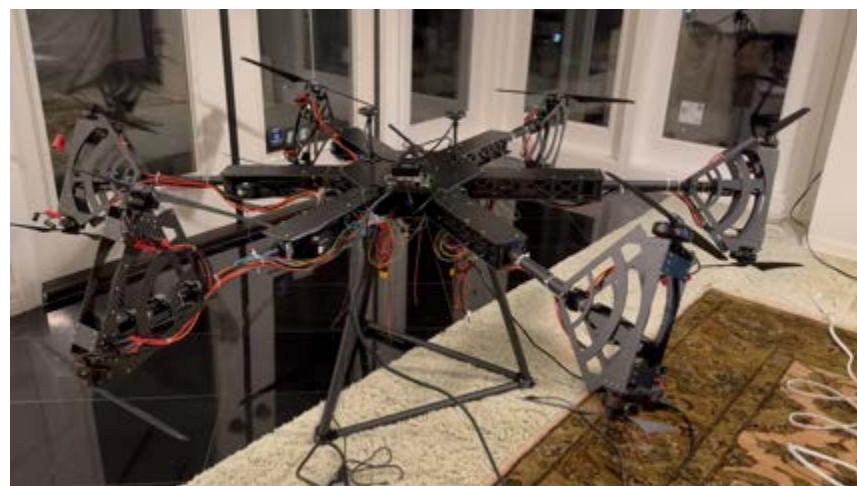
Images captured by the author from PX4 Flight Review (<https://review.px4.io/>)

7.3. Rescue UAV roll and pitch demonstration

The pitch and roll servo motors are commanded through U2D2 USB communication converter. A Python library for Dynamixel SDK was installed at JETSON Orin Nano mission controller for servo motor control (Robotis, 2024). A custom Python program was developed to adopt the SDK, and each motor was assigned with an ID corresponding to its arm actuator number. Extended position mode was used to drive gears and to achieve target position. The

program allows calibration of each motor to set a neutral position, while all motors can be controlled simultaneously. As shown in Figure 37, the pitch and roll were successfully performed with the program, and the target maximum tilt rotor feasibility was confirmed. Figure 38 shows one pitch block's angle sweep at extreme angles.

Figure 37. Pitch and roll actuation demonstration with a Python program



Photograph taken by the author

Figure 38. Rotor tilt across pitch angle range



Photographs taken by the author

8. Analysis

First, ocean image data is analyzed to extract wave and water flow in 8.1. Then rip currents are estimated based on the flow data in 8.2. Rescue UAV test flight results are discussed in 8.3.

8.1. Information-Weighted Average Differential Frame

Displacement

This section starts with the water optical flow algorithms from literature. Then they are applied to the real image sections to show the performance. A new information-weighted displacement is proposed as a solution to water flow analysis from information-deprived image tiles.

8.1.1. Differential frame displacement and error cross-

correlation

Literatures have two main optical flow particle image velocimetry algorithms – differential frame displacement (DFD) (4) (Dérian & Almar, 2017) and error cross correlation (ECC) function (5) and (6) (Puleo, Farquharson, Frasier, & Holland, 2003).

$$\mathbf{u}_D = \arg \min_{\mathbf{u}} \left\{ \int_A |\Delta I(\mathbf{x}, \mathbf{u})|^2 + k \int_A |f_{reg}|^2 \right\} \quad (4)$$

$$\mathbf{u}_E = \arg \max_{\mathbf{u}} \{e_c(\mathbf{u})\} \quad (5)$$

$$e_c(\mathbf{u}) = 1 - \frac{\int_A |I - I(t, \mathbf{u})|}{\int_A I + I(t, \mathbf{u})} \quad (6)$$

Both are similar as they minimize difference between the reference frame and search frame. ECC function has image intensity normalization in denominator. It is likely that DFD might benefit from such normalization, as there is no guarantee on the window intensity across the image and sequence. One of the DFD's benefit over ECC is noise suppression with regularization term in (4). It becomes clearer when the abstract terms are expanded in (7) with first-order regularization with gradient.

$$f_{reg} = ||\nabla \mathbf{u}|| \quad (7)$$

The algorithm uses vector \mathbf{u} to get solution u , and it becomes a recursive algorithm beginning with larger area, then moves into smaller area. Also, the algorithm becomes nonlinear as it goes through a complex search process. For simplicity, regularization was not used for DFD analysis. Both algorithms are compared by analyzing two sequential images at Figure 39. They are a portion of images at same coordinates taken 0.1 second apart from 100m altitude.

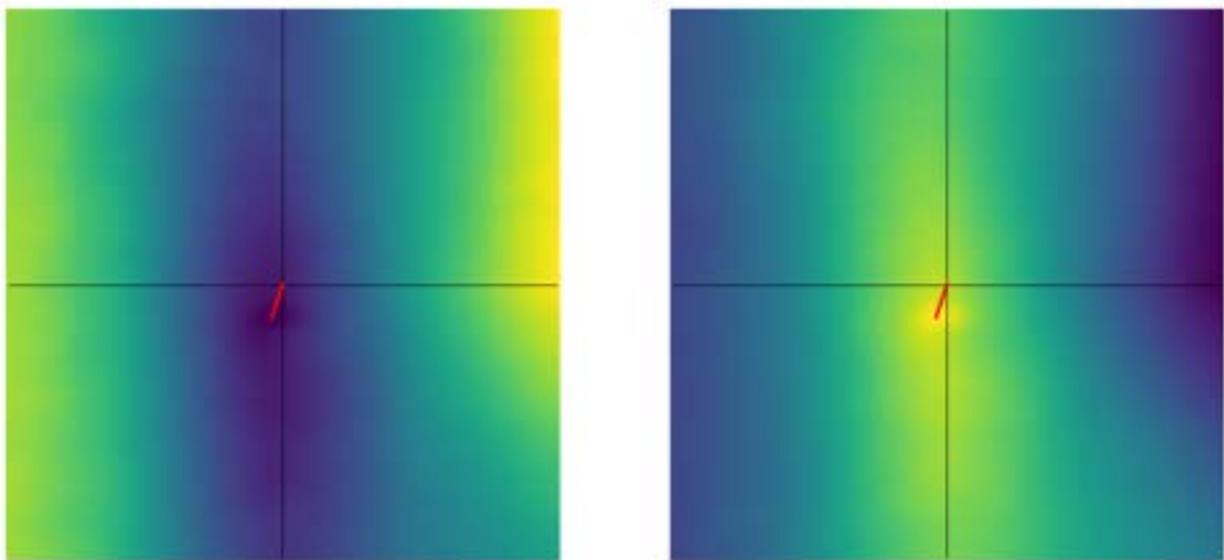
Figure 40 show DFD and ECC results for the center of tile of image with estimated displacement vectors. The DFD finds the minimum differences, while ECC looks for the maximum cross correlation. They yielded the same vector direction and displacement. The patterns look similar, if not the same, with inverted amplitude. The strong vertical direction movement comes from the scout UAV's flight distance between the two frames. The scout drone travelled at 5m/s, and two images are supposed to have about 5-pixel vertical shift. This shift will be applied to calculate the net water optical flow.

Figure 39. Example ocean image section for comparison



Photographs taken by the author

Figure 40. DFD and ECC vector analysis results

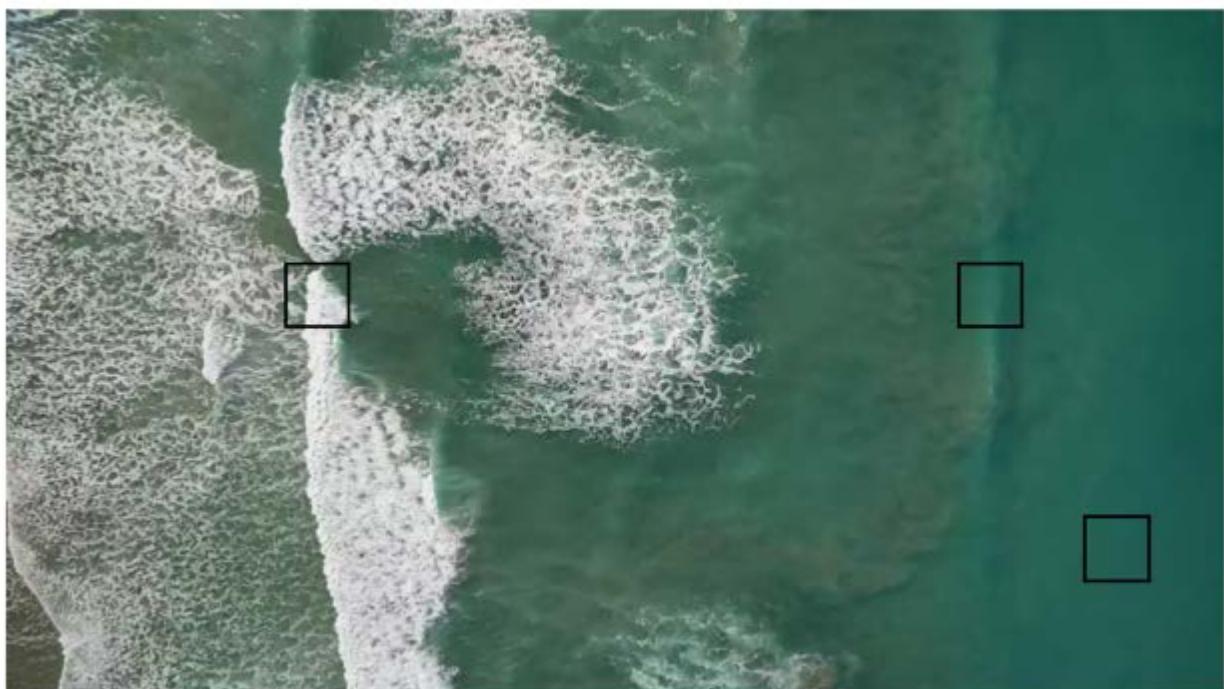


Images created by the author

8.1.2. Ocean image analysis with DFD and ECC

Both DFD and ECC can be compared for their performances and properties by applying them to the ocean image collection. Figure 41 is obtained from scout mission #9 over the King Tide. The square image sections will be analyzed with DFD and ECC algorithms.

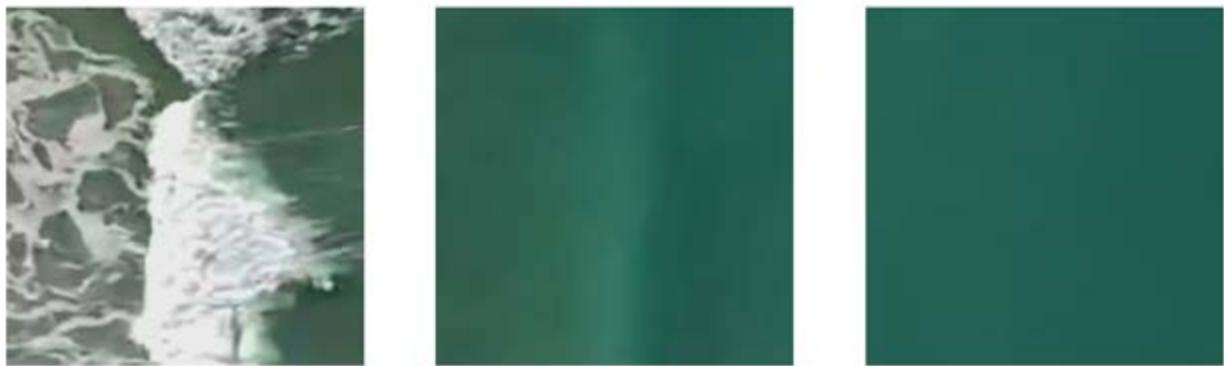
Figure 41. Scout UAV ocean image for wave analysis



Photograph taken by the author

As zoomed in Figure 42, three distinct wave phenomena are sampled at the image window: 1) wave breakage, 2) wave front, and 3) visually flat area.

Figure 42. Zoomed views of focus areas for analysis



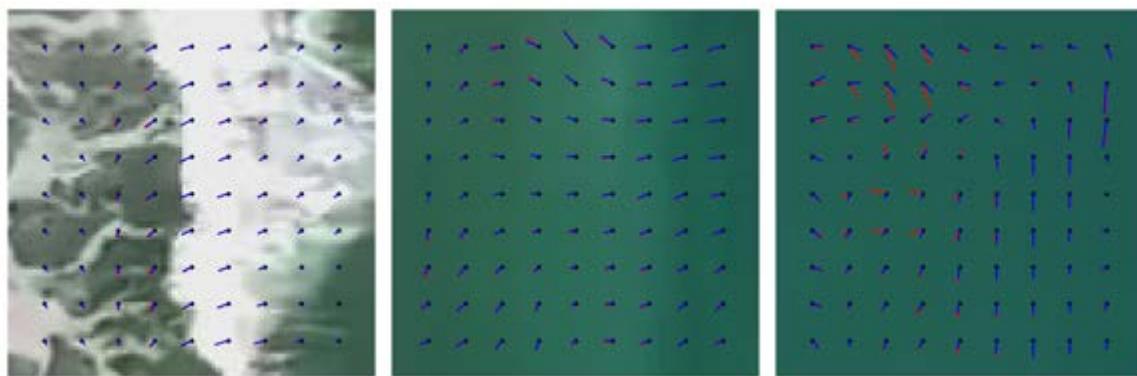
Photographs taken by the author

In practice, both DFD and ECC results have tangible variation at single point due to noise, and the outputs around the point are averaged to suppress noise and get meaningful results as expressed in (8). Averaging does not hurt the analysis, as the overall flow in the range of several meters are more useful to understand the water flow, rather than the 0.1m range flow from single pixel. DFD and ECC results for area #1-3 are plotted in

Figure 43. For comparison, DFD vectors are in red and ECC are in blue.

$$\bar{\mathbf{u}} = \frac{1}{B} \int_B \mathbf{u} \quad (8)$$

Figure 43. Water flow vectors from DFD in red and ECC in blue



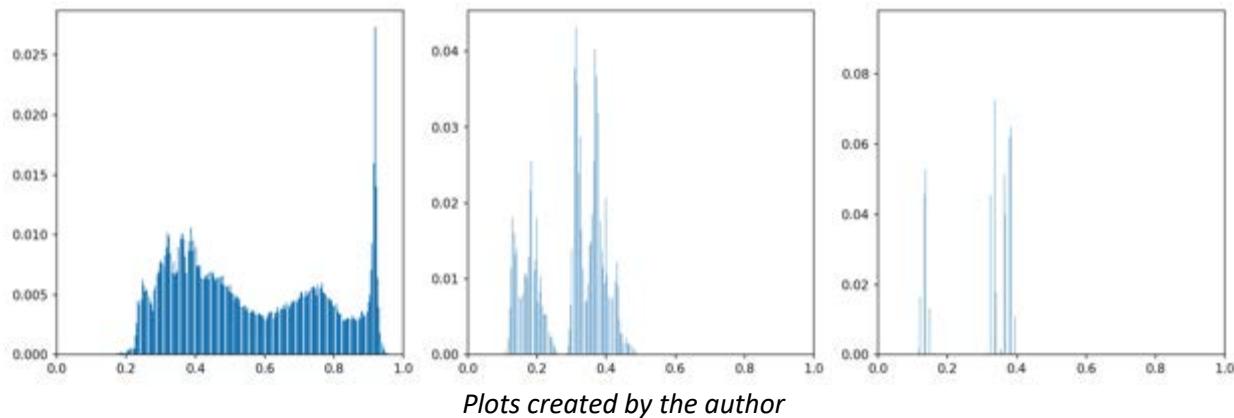
Photograph taken and images created by the author

For areas #1 and #2, both DFD and ECC have consistent results. Area #2 has wave front detected as left-pointing vectors, but there is a slight exaggeration in vertical direction. Area #3 shows errant behaviors for both. Also, 3) does not show much visual information, and the search algorithms must have found noisy and misleading results, while the flow vectors are strong. This behavior causes error in water flow estimation, then wrong rip current detections.

8.1.3. Information-Weighted Differential Frame Displacement

One of the root causes why area #3 resulted in extreme flow estimation is that the image does not have enough information to accurately extract a representing vector, not to mention any visual clues. Figure 44 shows RGB intensity histogram from each area. Area #3 lacks diverse distribution as in area #1 and 2, and it has narrow range of distribution.

Figure 44. Histograms of RGB intensity



The algorithms try to extract the best vector, assuming the given images have enough information to make a comparison. In practice, the given image section might not have such information, and the calculated vectors mislead in direction and amplitude. To address the issue, a new proposed solution in this research is to scale optical flow output vector by the relative amount of information in the image as in (9).

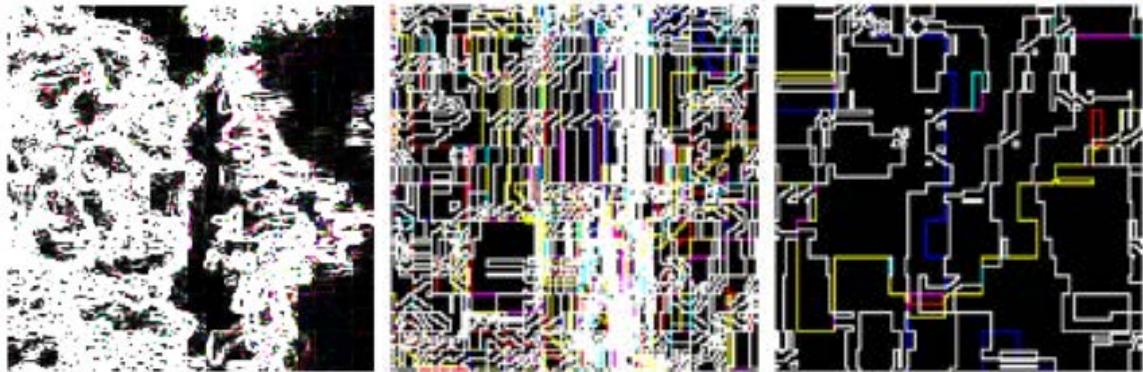
$$\mathbf{u}_w = w(I) \cdot \bar{\mathbf{u}} \quad (9)$$

The amount of information in an image can be estimated from several methods. The simplest is the ratio of standard deviation between the local image over the entire image in (10).

$$w(I) = \frac{\sigma_{\text{window}}}{\sigma_{\text{image}}} \quad (10)$$

Also, gradient of image shows the amount of information using a relatively simple matrix subtraction (Larkin, 2016) as analyzed in Figure 45. The comparison shows the gradient activity itself could indicate the amount of information, in addition to the amplitude of the gradient.

Figure 45. 2D gradient images of the area #1 - 3



Images created by the author

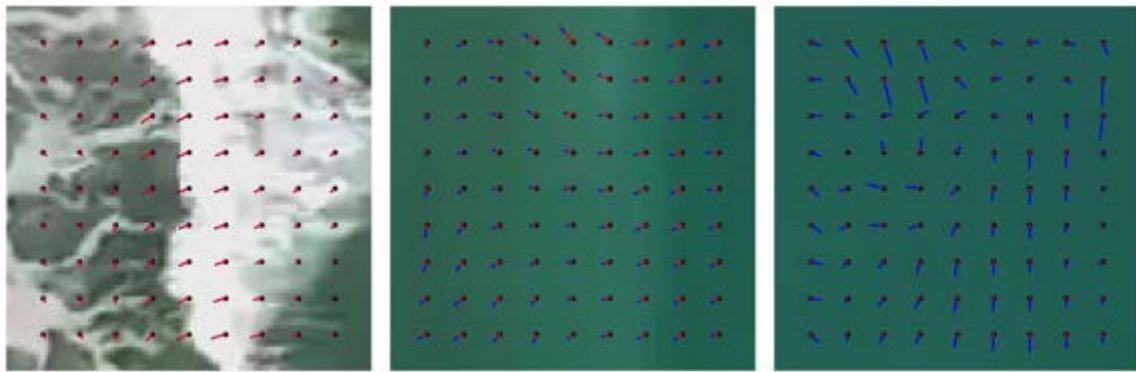
To consider the gradient activity, the non-zero gradient can be counted as gradient density as in (11). For the information weight, the ratio of gradient density between image window and entire image is used as in (12).

$$d(I) = \frac{1}{A_I} \sum_{|\delta| > 0} |\nabla I| \quad (11)$$

$$w(I) = \frac{d_{\text{window}(I)}}{d_{\text{entire image}}} \quad (12)$$

With the information-weighted scaling, the DFD vectors are suppressed when the image does not have significant features as shown in Figure 46. Original unweighted DFD vectors are in blue, and information weighted vectors are in red. The weighted vectors are scaled back when the gradient density is low, while they capture significant flow activities in area #1 and 2.

Figure 46. Information-weighted flow vector in red vs. before weighting in blue



Photograph taken and images created by the author

The benefit of information-weighted DFD water flow is clear when a complete image set is analyzed and compared. The entire image gradient density weight for DFD vector scaling is shown in Figure 47. With the scaling, the entire image comparison of weighted vs. unweighted DFD vectors are in Figure 48. The original DFD analysis produces strong and diverging vectors whenever the information becomes scarce. The information weighting successfully contains random direction and amplitude of DFD algorithm for a more meaningful water flow data.

Figure 47. Gradient density as information weight from the whole image

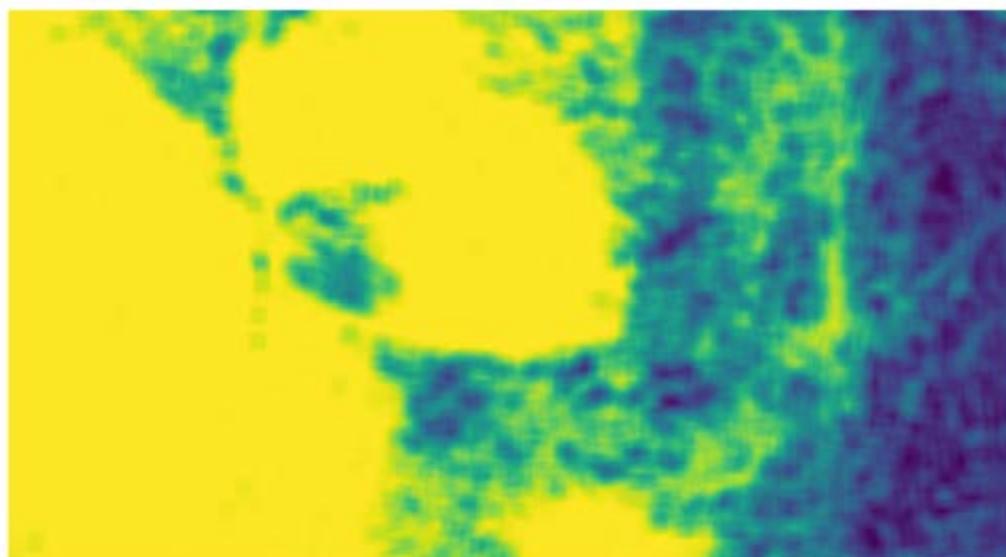
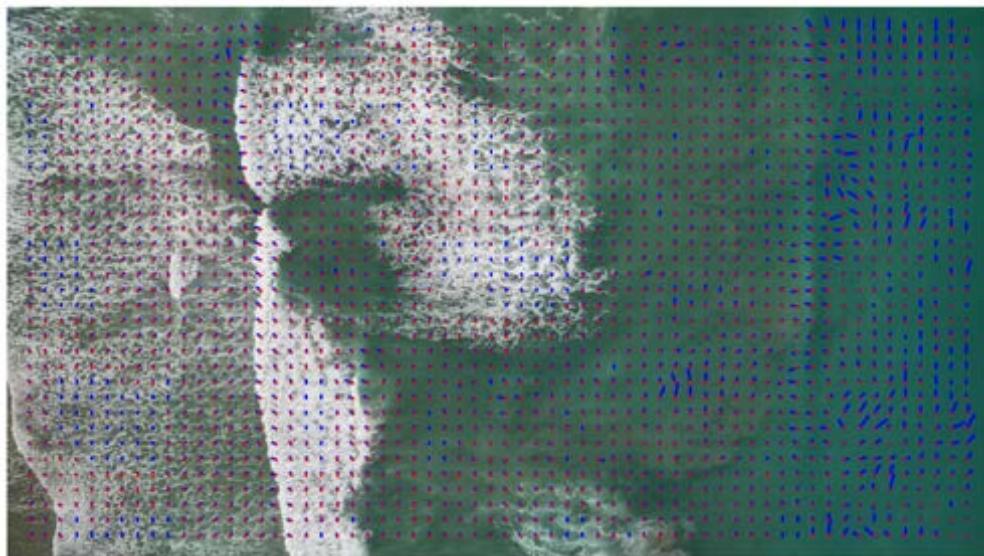


Image created by the author

Figure 48. Information-weighted flow in red vs. unweighted flow in blue



Photograph taken and image created by the author

The information-weighted DFD provides more reliable optical flow and particle image velocimetry as analyzed in Figure 49. The receding flows are in red and incoming flows are in blue in the figure. Using the flow information, further analysis can be performed to identify rip current and tide flow.

Figure 49. Information-weighted incoming (blue) and receding (red) for vectors



Photograph taken and image created by the author

8.2. Rip current detection with channel current analysis

8.2.1. Ocean depth and risk model

Rip currents are the biggest threat for scout UAV to detect. This section utilizes previously analyzed information-weighted optical water flow to estimate ocean flow, and to identify rip currents and extreme tide current. There are a few assumptions to start rip current channel analysis:

- Rip current occurs rather wide area greater than 1 meter
- Rip current is a relatively long-term behavior longer than 10 seconds
- Strong receding flow poses risks in addition to the rip currents

For a simpler analysis, it is also assumed that the scout UAV's flight path is orthogonal to the wave and flow.

To obtain overall water flow, cross-wave flow vectors out of the ocean water flow data can be added up to estimate a current channel's net flow as the simplest approach as in (12).

$$\text{flow} = \int_{\text{Time}}^{\square} \int_{\text{Channel}}^{\square} \int_{\text{Wave}}^{\square} u_w(t, \mathbf{x}) \quad (12)$$

One issue with (12) is, it treats flows at shallow and deep water same. Because deep water involves more water volume with the same flow than the shallow water, we need to introduce a depth model to (12) as shown in Figure 50. The water depth increases volume of the flow, and the current flow volume needs a depth as in (13), where $c(\mathbf{x})$ is a channel depth model, which can be obtained from the real underwater depth measurement, or a linear function of surface distance.

Figure 50. Rip current depth and risk model

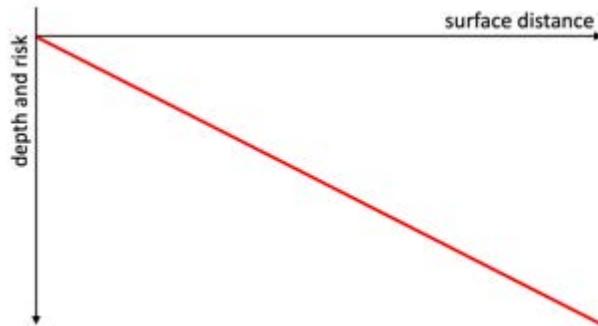


Diagram created by the author

$$\text{flow volume} = \int_{\text{Time}}^{\square} \int_{\text{Channel}}^{\square} \int_{\text{Wave}}^{\square} c(\mathbf{x}) \cdot u_w(t, \mathbf{x}) \quad (13)$$

Additionally, the depth also increases risk to swimmers and surfers, as they cannot react to the flow as they are in deeper water. As a result, with (14), the depth and risk together

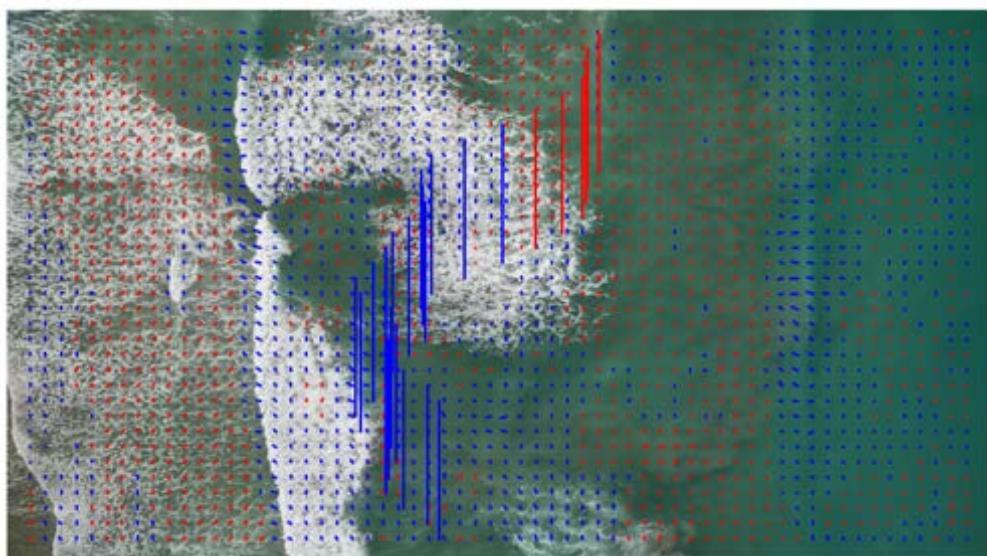
will make a quadratic function multiplied to the flow when rip current risk is assessed, where $r(\mathbf{x})$ is a quadratic function approximately.

$$\text{flow risk} = \int_{\text{Time}}^{\square} \int_{\text{Channel}}^{\square} \int_{\text{Wave}}^{\square} r(\mathbf{x}) \cdot u_w(t, \mathbf{x}) \quad (14)$$

8.2.2. Rip current analysis with depth and risk models

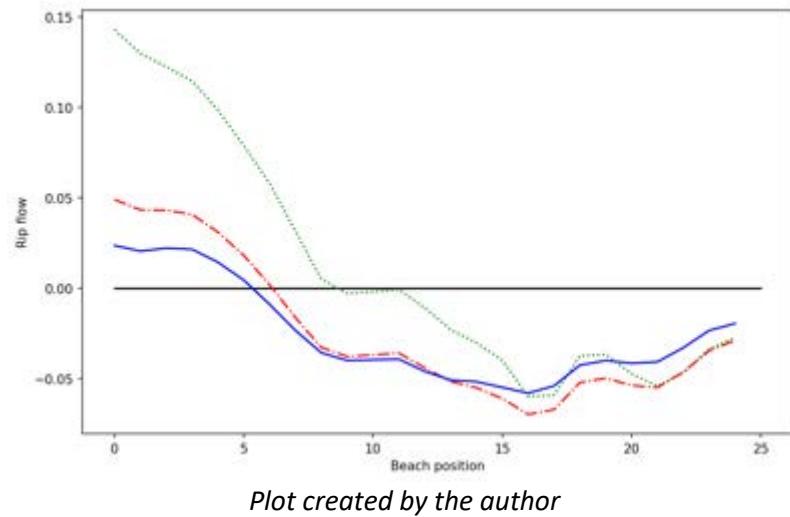
The flow risk model (14) is applied to the scout mission #9 images as in Figure 51. It averaged flow over 10 seconds at the same geographic position. Red lines are the channels where the average flow with depth and risk models show net ebb flow, and blue lines are net flood flow. The unweighted, depth model, and depth plus risk model rip flows are compared at Figure 52. The unweighted model analyzes that rip flow is larger than other two models. Rip currents are detected when a group of current channel exhibits strong ebb flow over a period of time.

Figure 51. Rip risk model analysis results



Photograph taken and image created by the author

Figure 52. Comparison of unweighted, depth, and depth+risk model rip flow



8.3. Rescue drone test flight

Pixhawk 6X flight controller was set up for coaxial hexacopter, and the designed arm length parameters were entered to the controller for control allocation as shown in Figure 53. The rescue UAV was able to take off, and Figure 54 shows its flight capture. The rescue UAV weighs 12.3Kg without battery, and the take off and test flight confirms its thrust is more than the UAV weight with margin as designed.

Figure 53. The rescue UAV actuator configuration as a coax hexacopter

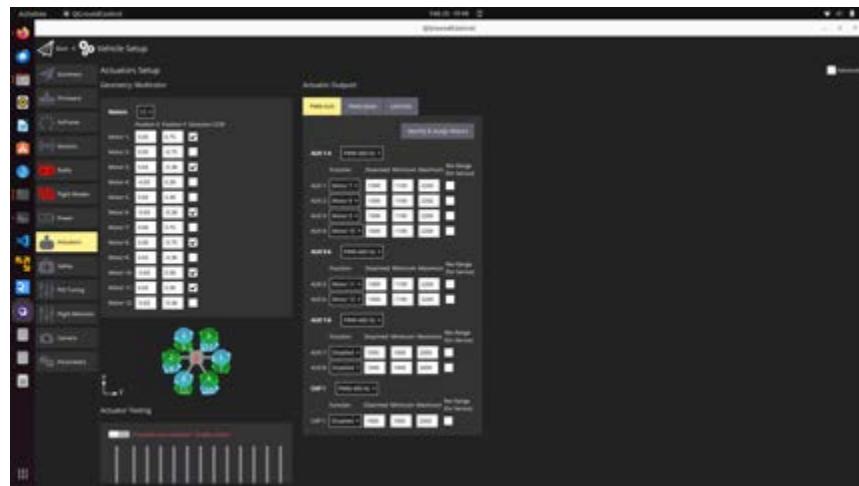


Image captured by the author

Figure 54. Rescue UAV test flight in progress



Photograph taken by the designated supervisor

The rescue UAV needs a control scheme that enables omni-directional flight. The Newton-Euler equations of translational and rotational dynamics of a rigid body in (15) should be set and solved with actuator and dual tilt rotor angles. The rescue UAV's extended pitch angle should be utilized when the equation is used to stay away from singularities.

$$\begin{bmatrix} mI_3 & 0 \\ 0 & J \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega \times (m \cdot v) \\ \omega \times (I \cdot \omega) \end{bmatrix} = \begin{bmatrix} F \\ M \end{bmatrix}$$

(15)

Pulling a victim out of water sounds straightforward, while it is a well known control problem called “slung load”, and it needs careful implementation. The weight of the victim, victim’s sudden force on the rope, ocean wave, rip current’s proximity, and wind are unknown or change dynamically. The rescue drone’s omni-directional orientation and flight should be solved and designed together with the slung load problem to maintain proper control during the rescue operation.

Conclusion

UAVs have strong potential to assist and extend lifeguard operation. UAVs are getting introduced to lifeguard operation, and they are in early adoption as they require manual pilot and visual examination. Lifeguard UAV requirements were discussed and proposed. A scout UAV and a rescue UAV were designed, fabricated, and demonstrated their basic functionality. The scout UAV is equipped with AI mission controller, strong thrust from actuator and propeller combination, triple redundant radio links, and large-capacity battery for extended flight duration. The scout UAV conducted nine scout missions at Del Mar beach to collect 11.8k beach images with 13.4km mileage. The collected images were used to develop an information-weighted optical flow to extract reliable ocean water flow from the images. A channel current flow estimation was performed over the optical flow, and a depth and risk model was applied to detect rip currents and tide flow. The rescue UAV implements a novel in-arm pitch axis for ± 45 -degree pitch axis in addition to ± 180 -degree roll. The maximum pitch angle improves pitch angle range by 65% from conventional design. The UAV was custom designed from scratch and built with 86 CNCed carbon fiber plate parts, 320 Nylon 3D printed parts, and 1,000 screws. The rescue UAV weighs about 12.3Kg without battery and wingspan is 2.0m. The rescue UAV can be triple folded to decrease diameter by 45% for transportation. The rescue UAV demonstrated roll and pitch angle sweep, and test flights were conducted. This research highlighted advantages of lifeguard scout and rescue UAV with their feasibility, while recommendations and challenges are listed as next steps.

Recommendations

Consult with lifeguards

- Interview local lifeguards for suggestions

Scout UAV

- Implement real-time wave analysis and rip current detection using mission controller

Rescue UAV

- Improve mechanical design
 - Metallic gears
 - Power and signal wiring
 - Leg and shoe to absorb shock
- Implement omni-directional control
- Implement slung load control along the omni-direction control to pull victim to beach

Bibliography

14 CFR Part 107. (2023, February 20). *PART 107—SMALL UNMANNED AIRCRAFT SYSTEMS*.

Retrieved from Code of Federal Regulations: <https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-107>

Akshata. (2024, February 20). *PX4 vs. ArduPilot: Choosing the Right Open-Source Flight Stack*.

Retrieved from The Droning Company: <https://www.thedroningcompany.com/blog/px4-vs-ardupilot-choosing-the-right-open-source-flight-stack>

Allenspach, M., Bodie, K., Brunner, M., Rinsoz, L., Taylor, Z., Kamel, M. S., . . . Nieto, J.

(2020). Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight. *The International Journal of Robotics Research*, 1305-1325.

Amazon. (2024, February 20). *Amazon*. Retrieved from Amazon: <https://www.amazon.com/>

American Red Cross. (2023, February 20). *Lifeguard Training*. Retrieved from American Red

Cross: <https://www.redcross.org/take-a-class/lifeguarding/lifeguard-training>

Ardupilot. (2024, February 20). *Ardupilot Versatile, Trusted, Open*. Retrieved from Ardupilot: <https://ardupilot.org/>

Bodie, K., Brunner, M., Pantic, M., Walser, S., Pfandler, P., Angst, U., . . . Nieto, J. (2019). An

Omnidirectional Aerial Manipulation Platform for Contact-Based Inspection.

Proceedings of Robotics: Science and Systems.

Bodie, K., Taylor, Z., Kamel, M., & Siegwart, R. (2018). Towards Efficient Full Pose

Omnidirectionality with Overactuated MAVs. *Proceedings of the 2018 International Symposium on Experimental Robotics*, (pp. 85-95).

Brescianini, D., & D'Andrea, R. (2016). Design, modeling and control of an omni-directional

aerial vehicle. *IEEE International Conference on Robotics and Automation (ICRA)*.

bresh9019. (2023, February 20). *Thrust vectoring drone*. Retrieved from YouTube:

<https://www.youtube.com/watch?v=u2cETOyuJ20>

California Department of Parks and Recreation. (2023, February 20). *Ocean Safety*. Retrieved from California Department of Parks and Recreation:

https://www.parks.ca.gov/?page_id=23792

City of Del Mar. (2023, February 14). *Drones*. Retrieved from City of Del Mar:

<https://www.delmar.ca.us/808/Drones>

Dérian, P., & Almar, R. (2017). Wavelet-Based Optical Flow Estimation of Instant Surface Currents From Shore-Based and UAV Videos. *IEEE Transactions on Geoscience and Remote Sensing*, 5790-5797.

DUSEK, G. (2024, February 20). *Rip Current Survival Guide*. Retrieved from NOAA Ocean Today: <https://oceantoday.noaa.gov/ripcurrentfeature/>

f, F. (2023, March 15). *Dynamixel XL430-W250-T*. Retrieved from GRABCAD:

<https://grabcad.com/library/dynamixel-xl430-w250-t-1>

F., M., Bicego, D., Ryll, M., & Franchi, A. (2018). Energy-Efficient Trajectory Generation for a Hexarotor with Dual- Tilting Propellers. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (pp. 6226-6232). Madrid, Spain.

Federal Aviation Administration. (2023, February 20). *Recreational Flyers and Community-Based Organizations*. Retrieved from Federal Aviation Administration:

https://www.faa.gov/uas/recreational_flyers

Federal Aviation Administration. (2023, February 14). *Unmanned Aircraft Systems*. Retrieved from Federal Aviation Administration: <https://www.faa.gov/uas>

- Federal Aviation Administration. (2024, February 20). *B4UFLY*. Retrieved from Federal Aviation Administration: https://www.faa.gov/uas/getting_started/b4ufly
- GETFPV. (2024, February 20). *GETFPV*. Retrieved from GETFPV: <https://www.getfpv.com>
- Gibson, D. (2020, August 23). *T-Motor 18x6 composite folding propeller pair CW CCW*. Retrieved from GRABCAD: <https://grabcad.com/library/t-motor-18x6-composite-folding-propeller-pair-cw-ccw-1>
- Gung Ho Vids. (2023, February 20). *V-22 Osprey Tilt-Rotor Aircraft In Action and Compilation*. Retrieved from YouTube: <https://www.youtube.com/watch?v=xnOw3clRIoI>
- Hart, D. P. (1998). High-Speed PIV Analysis Using Compressed Image Correlation. *Journal of Fluids Engineering*, 463-470.
- Holland, K., Puleo, J., & Kooney, T. (2001, December). Quantification of swash flows using video-based particle image velocimetry. *Costal Engineering*, pp. 76-77.
- Jiang, G., & Voyles, R. (2013). Hexrotor UAV platform enabling dexterous interaction with structures-flight test. *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*.
- Junaid, A. B., Sanchez, A., Bosch, J., Vitzilaios, N., & Zweiri, Y. (2018). Design and Implementation of a Dual-Axis Tilting Quadcopter. *Robotics*, 65.
- Kamel, M., Verling, S., Elkhatib, O., Sprecher, C., Wulkop, P., Taylor, Z., . . . Gilitschenski, I. (2018, October). The Voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle. *IEEE Robotics & Automation Magazine*, pp. 34-44.
- Kim, A. (2023). Deep reinforcement learning of position and Velocity PID control for rotational wing unmanned aerial vehicles. *5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*.

- Kucher, K. (2022, September 29). *Man whose body was found in water off Torrey Pines State Beach may have gone swimming from unguarded beach*. Retrieved from Del Mar Times: <https://www.delmartimes.net/news/story/2022-09-29/man-whose-body-was-found-in-water-off-torrey-pines-state-beach-may-have-gone-swimming-from-unguarded-beach>
- Larkin, K. G. (2016). Reflections on Shannon Information: In search of a natural information-entropy for images. *arXiv*.
- Lucas, B., & Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, (pp. 674-679).
- MAVLink. (2024, February 20). *MAVLink Developer Guide*. Retrieved from MAVLink: <https://mavlink.io/en/>
- Nakano, I., & Ishida, H. (2012). An acoustic monitoring system of the rip current based on passive reciprocal transmissions. *2012 Oceans - Yeosu*.
- National Oceanic and Atmospheric Administration. (2023, February 14). *NOAA TIDES & CURRENTS*. Retrieved from National Oceanic and Atmospheric Administration: <https://tidesandcurrents.noaa.gov/>
- National Oceanic and Atmospheric Administration. (2023, February 14). *NOAA TIDES & CURRENTS*. Retrieved from National Oceanic and Atmospheric Administration: <https://tidesandcurrents.noaa.gov/>
- New York State. (2023, February 14). *On Shark Awareness Day, Governor Hochul Distributes New Drones to Take a Bite Out of Dangerous Shark Encounters on Long Island Beaches*. Retrieved from New York State: <https://www.governor.ny.gov/news/shark-awareness-day-governor-hochul-distributes-new-drones-take-bite-out-dangerous-shark>

nicolalego. (2023, February 20). *3D PRINTED SINGLECOPTER WITH THRUST VECTORING.*

Retrieved from YouTube: https://www.youtube.com/watch?v=YdvwP_g6c2M

NOAA Tides and Currents. (2024, February 20). *NOAA Tide Predictions.* Retrieved from NOAA

Tides and Currents: https://tidesandcurrents.noaa.gov/tide_predictions.html

NVIDIA. (2024, February 20). *Embedded Systems with Jetson.* Retrieved from NVIDIA:

<https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/>

PATEL, D. (2021, February 25). *Antigravity MN5008.* Retrieved from GRABCAD:

<https://grabcad.com/library/antigravity-mn5008-1>

Perkovic, D., Lippmann, T. C., & Frasier, S. J. (2009). Longshore Surface Currents Measured by

Doppler Radar and Video PIV Techniques. *IEEE Transactions on Geoscience and Remote Sensing*, 2787-2800.

Puleo, J. A., Farquharson, G., Frasier, S. J., & Holland, K. T. (2003). Comparison of optical and

radar measurements of surf and swash zone velocity fields. *Journal of Geophysical Research Oceans*.

PX4. (2024, February 20). *Open Source Autopilot.* Retrieved from PX4: <https://px4.io/>

PX4. (2024, February 20). *PX4 Autopilot User Guide.* Retrieved from PX4 User Guide:

<https://docs.px4.io/main/en/>

PX4 Flight Review. (2024, February 20). *PX4 Flight Review.* Retrieved from PX4 Flight

Review: <https://review.px4.io/>

QGroundControl. (2024, February 20). *QGroundControl.* Retrieved from QGroundControl:

<http://qgroundcontrol.com/>

Rajappa, S., Ryll, M., Bulthoff, H. H., & Franchi, A. (2015). Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers. *IEEE International Conference on Robotics and Automation (ICRA)*.

Raspberry Pi. (2024, February 20). *Raspberry Pi*. Retrieved from Raspberry Pi:

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

Rivellini, S. (2024, February 20). *Bladeless Drone: First Flight*. Retrieved from YouTube:

<https://www.youtube.com/watch?v=5L6FSdUmEpg>

Robotis. (2024, February 20). *Dynamixel SDK*. Retrieved from Dynamixel SDK:

https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/

Robotis. (2024, February 20). *Robotis*. Retrieved from Robotis: <https://www.robotis.us/>

Ryll, M., Bulthoff, H. H., & Giordano, P. R. (2013). First flight tests for a quadrotor UAV with tilting propellers. *IEEE International Conference on Robotics and Automation*.

S. K., Lee, T., & Kumar, V. (2013). Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load. *52nd IEEE Conference on Decision and Control*.

Segui-Gasco, P., Al-Rihani, Y., Shin, H.-S., & Savvaris, A. (2013). A novel actuation concept for a multi rotor UAV. *International Conference on Unmanned Aircraft Systems (ICUAS)*.

SIYI. (2024, February 20). *SIYI*. Retrieved from SIYI: <https://shop.siyi.biz/>

Tan, J. K. (2023, April 11). *PIXHAWK 5X / PIXHAWK 6X*. Retrieved from GRABCAD:

<https://grabcad.com/library/pixhawk-5x-pixhawk-6x-1>

T-MOTOR. (2024, February 20). *AT2317 Long Shaft Outdoor 3D Trainer Glider Motor*.

Retrieved from T-MOTOR: <https://store.tmotor.com/goods.php?id=789>

T-MOTOR. (2024, February 20). *T-Motor MN5008 Antigravity Type 6-12S UAV Motor KV340*. Retrieved from T-MOTOR: <https://store.tmotor.com/product/mn5008-kv340-motor-antigravity-type.html>

Trizna, D. B. (2018). Coherent marine radar observations of rip current features with high temporal resolution. *OCEANS 2018 MTS/IEEE Charleston*.

Tyto Robotics. (2024, Februady 20). *Coaxial Configuration Setup and Testing*. Retrieved from Tyto Robotics: <https://www.tytorobotics.com/blogs/general-knowledge/coaxial-setup-and-testing>

United States Lifesaving Association. (2023, February 14). *American Lifeguard Rescue and Drowning Statistics for Beaches*. Retrieved from United States Lifesaving Association: <https://www.usla.org/page/Statistics>

VGR-Systems. (2024, February 20). *Coaxial drone*. Retrieved from YouTube: https://www.youtube.com/watch?v=6FU_HeXyI-Y

World Health Organization. (2023, July 25). *Drowning Fact Sheet*. Retrieved from World Health Organization: <https://www.who.int/news-room/fact-sheets/detail/drowning>

World Health Organization. (2023, February 20). *Health Topics - Drowning*. Retrieved from World Health Organization: <https://www.who.int/health-topics/drowning>

Xiang, G., Hardy, A., Rajeh, M., & Venuthurupalli, L. (2016). Design of the life-ring drone delivery system for rip current rescue. *2016 IEEE Systems and Information Engineering Design Symposium (SIEDS)*.

Yang, S., & Xian, B. (2020). Energy-Based Nonlinear Adaptive Control Design for the Quadrotor UAV System With a Suspended Payload. *IEEE Transactions on Industrial Electronics*, 2054-2064.

Yang, S., Xian, B., Cai, J., & Wang, G. (2024). Finite-Time Convergence Control for a Quadrotor Unmanned Aerial Vehicle With a Slung Load. *IEEE Transactions on Industrial Informatics*, 605-614.

Appendix A. Regulation compliance documents

Recreational drone certificate



Image captured by the author

FAA drone registration



**Federal Aviation
Administration**

Small UAS Certificate of Registration

REGISTERED OWNER: **Angelina Kim**

REGISTRATION NUMBER: **FA3HRKM494**

ISSUED: **06/28/2023**

EXPIRES: **06/28/2026**

This Small UAS Certificate of Registration **is not an authorization to conduct flight operations** with an unmanned aircraft. Operators of unmanned aircraft must ensure they comply with the appropriate safety authority from the FAA. To operate as a recreational flyer, a person must meet all of the statutory conditions of the exception for limited recreational operations of unmanned aircraft (49 U.S.C. 44809). Persons who do not meet all of the statutory conditions may not operate under the statutory exception for limited recreational operations of unmanned aircraft.

For U.S. citizens, permanent residents, and certain non-citizen U.S. corporations, this document constitutes a Certificate of Registration. For all others, this document represents a recognition of ownership.

To fly under the exception for recreational flyers you must:

- Have a current registration
- Fly only for recreational purposes
- Follow the safety guidelines of a community based organization
- Keep your drone within your visual line of sight
- Give Way and do not interfere with any manned aircraft
- Fly at or below 400' in controlled airspace and only with prior authorization
- Fly at or below 400' in uncontrolled airspace
- Comply with all airspace restrictions
- Pass The Recreational UAS Safety Test

Image captured by the author

FCC amateur radio license

Call Sign / Number N6SEA	Grant Date 07-07-2017	Expiration Date 07-07-2027	File Number 0007853118	Print Date 07-14-2017	Effective Date 07-07-2017
Operator Privileges Technician		Station Privileges PRIMARY	Here THIS LICENSE IS NOT TRANSFERABLE Special Conditions / Endorsements: NONE		
KIM, ANGELINA [REDACTED]					
AMATEUR RADIO LICENSE FCC Registration Number (FRN): 0026446757					
FCC 660 - August 2021					
<hr style="margin-top: 10px; margin-bottom: 5px;"/> (Licensee's Signature) FEDERAL COMMUNICATIONS COMMISSION					



Image captured by the author

Appendix B. Tidal charts

October 2023

Ocean data collection on 10/28

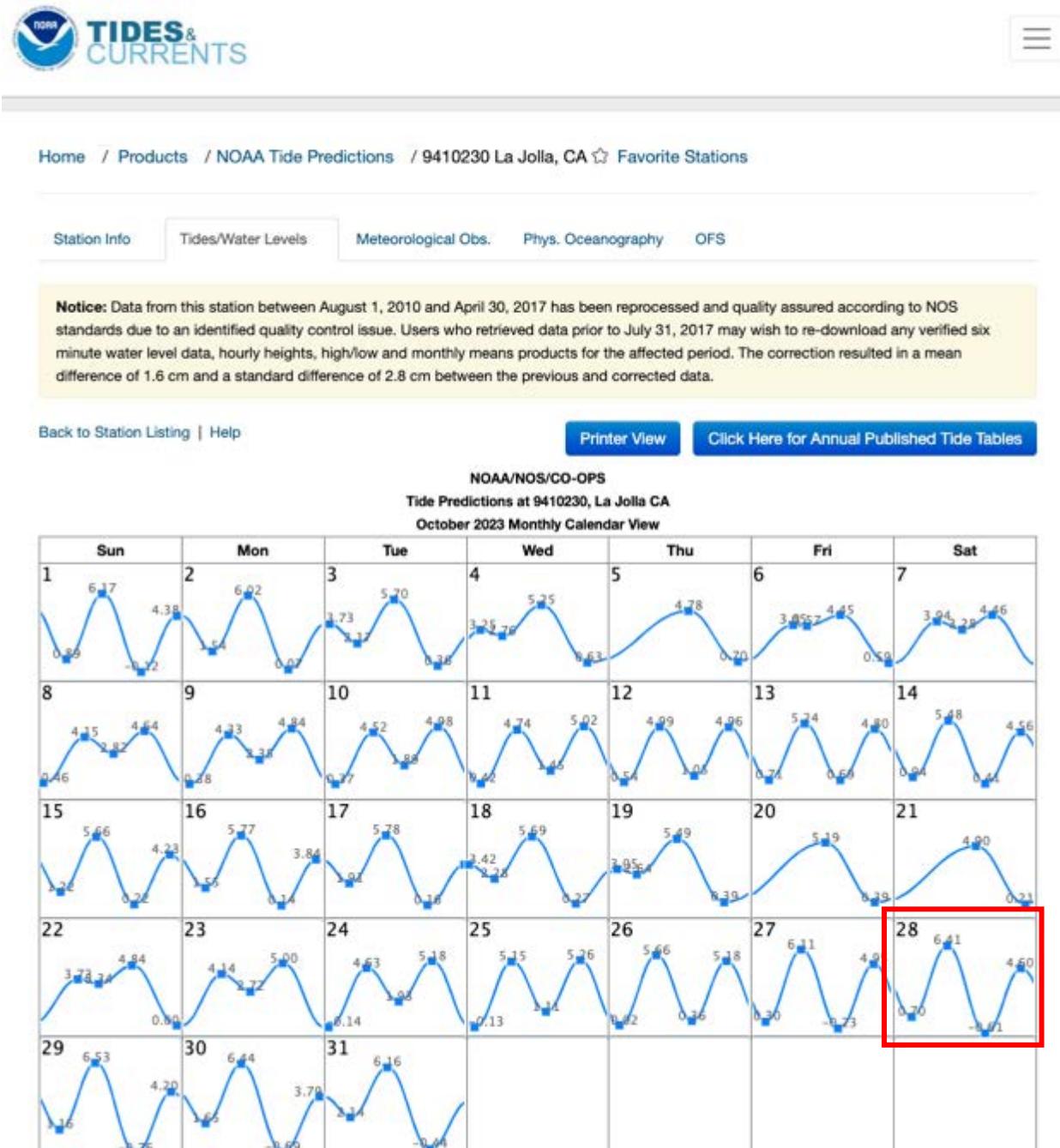


Image captured from NOAA(<https://tidesandcurrents.noaa.gov/>)

November 2023

Ocean data collection on 11/17 and 11/22

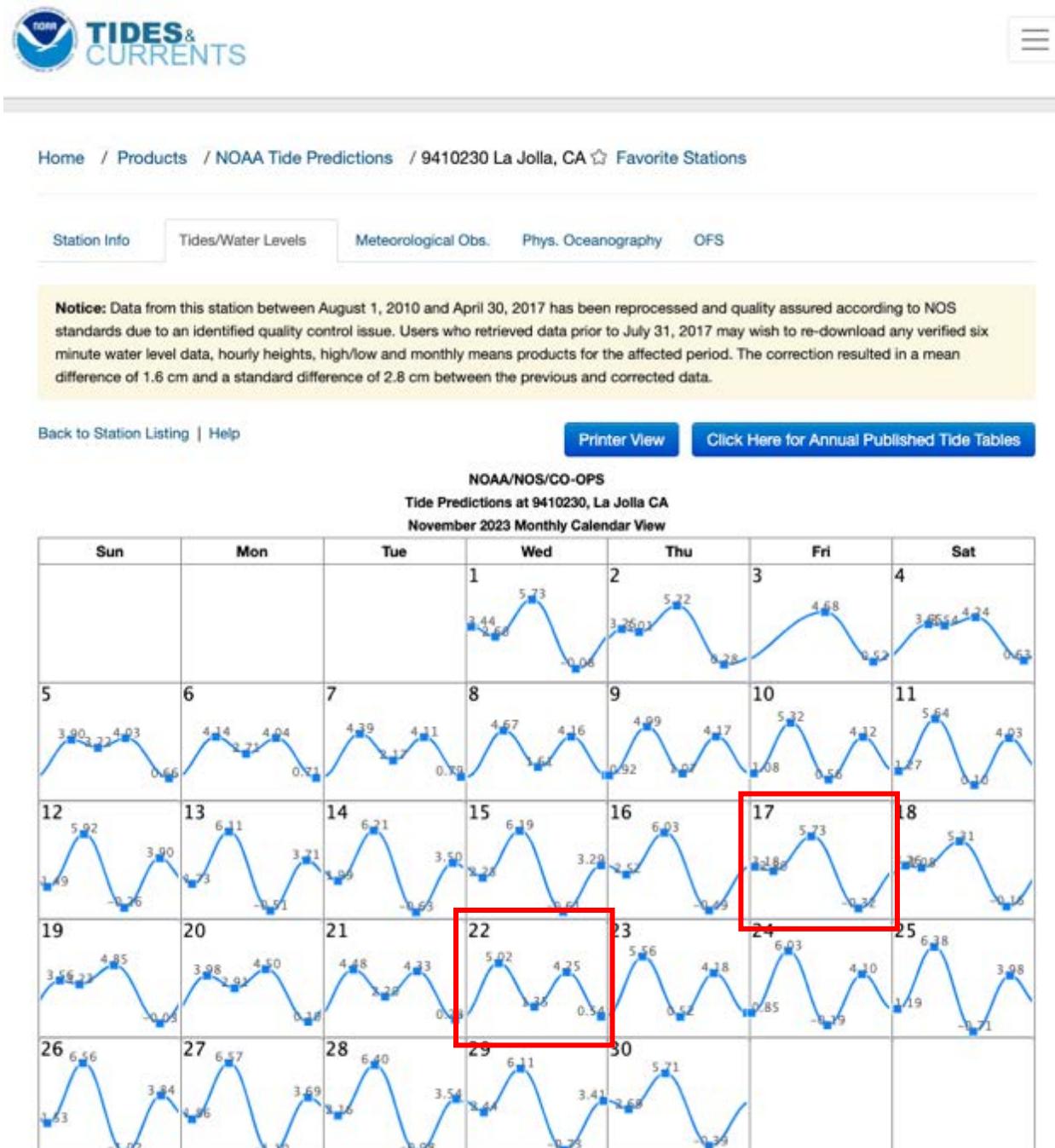


Image captured from NOAA(<https://tidesandcurrents.noaa.gov/>)

December 2023

Ocean data collection on 12/25

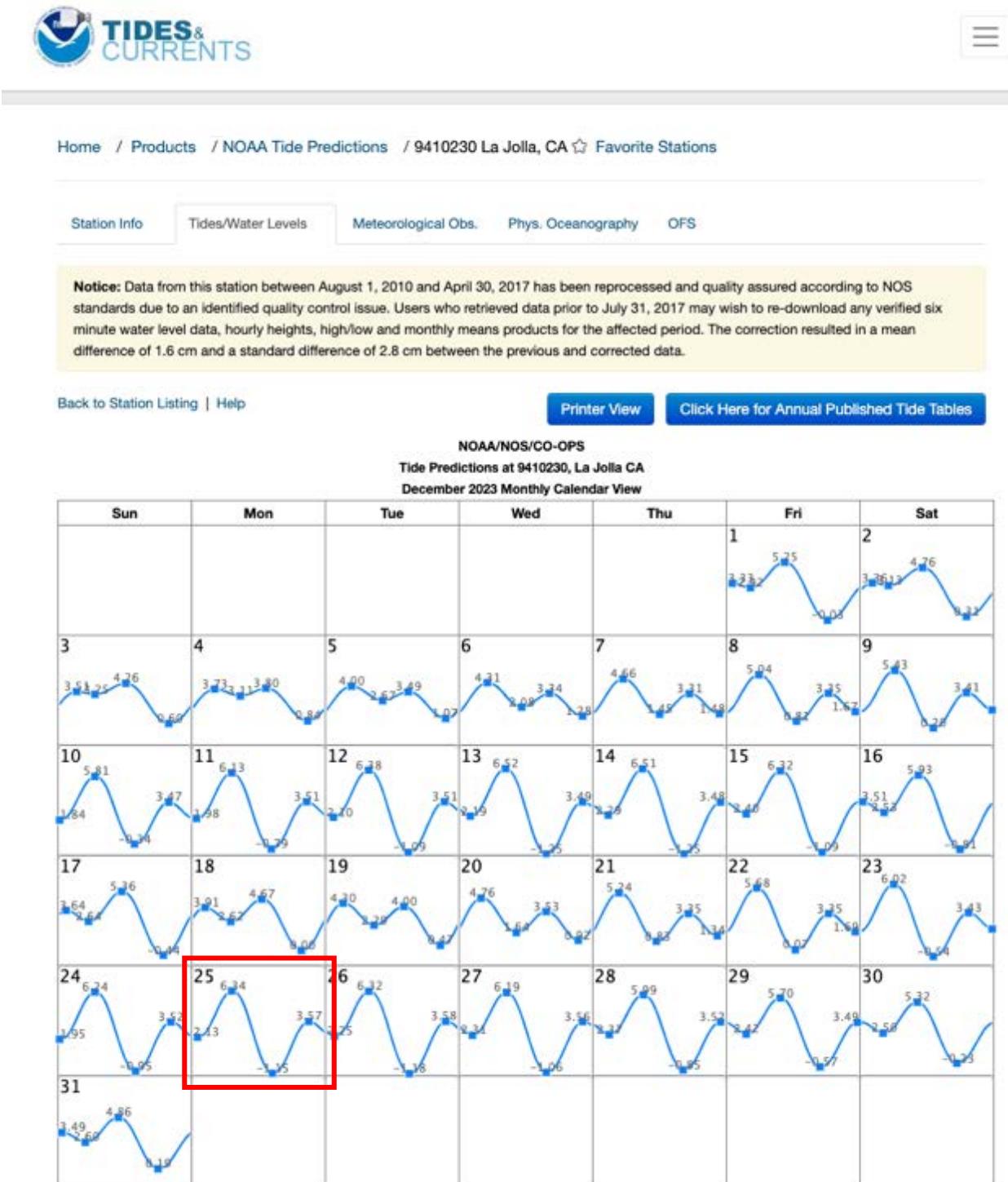


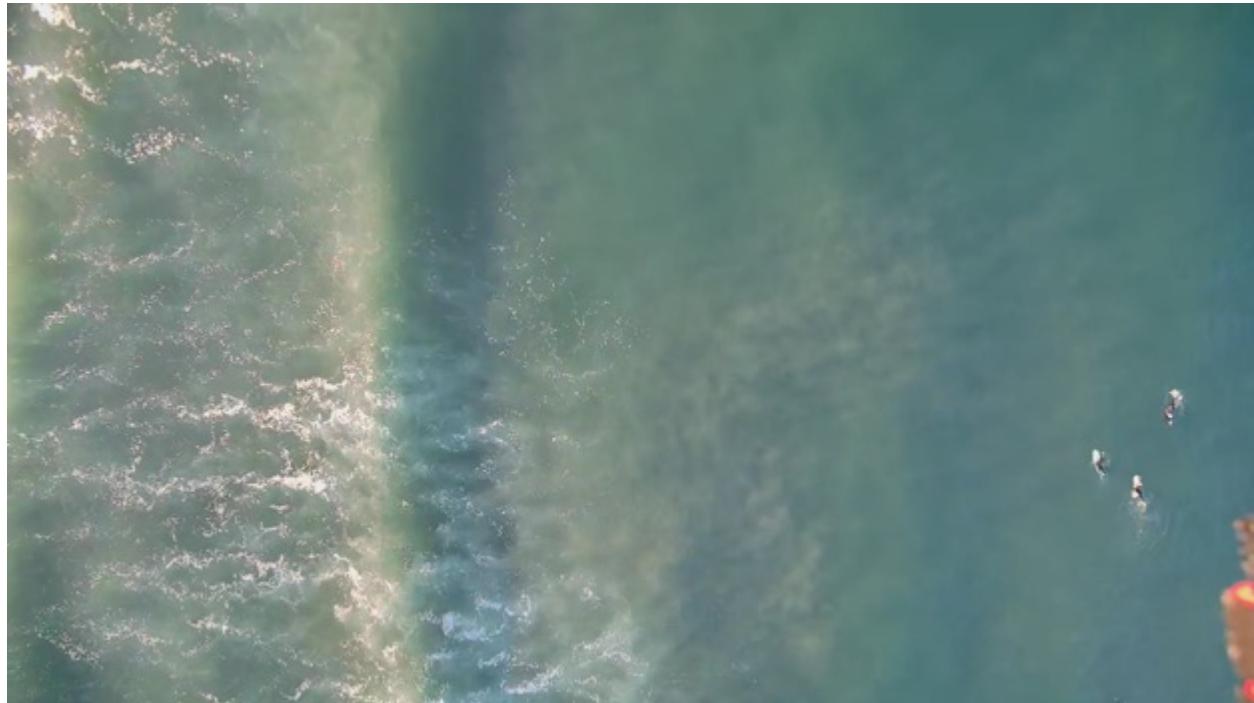
Image captured from NOAA(<https://tidesandcurrents.noaa.gov/>)

Appendix C. Images from Scout Missions

Flight #	Date	Time	Duration	Distance (km)	Altitude (m)	Take off	Frame per second	# of images	Data size (MB)
1	10/28/23	7:22	0:04:40	1.03	50	18th St.	1	0	0
2	10/28/23	7:39	0:04:48	1.04	50	18th St.	1	289	99.3
3	11/17/23	16:33	0:03:22	0.53	100	18th St.	1	88	27.9
4	11/17/23	16:42	0:06:45	1.52	100	18th St.	1	0	0
5	11/17/23	17:00	0:06:33	1.50	100	18th St.	1	299	58.5
6	11/22/23	8:19	0:08:38	2.09	100	27th St.	5	1950	499
7	11/22/23	9:01	0:07:27	1.75	100	18th St.	5	1660	441.5
8	12/25/23	9:43	0:08:29	2.04	100	27th St.	10	3995	2120
9	12/25/23	10:14	0:07:52	1.90	100	15th St.	10	3600	1900
Total				13.4				11881	5146

Table created by the author

Scout mission #2



Photograph taken by the author

Scout mission #3



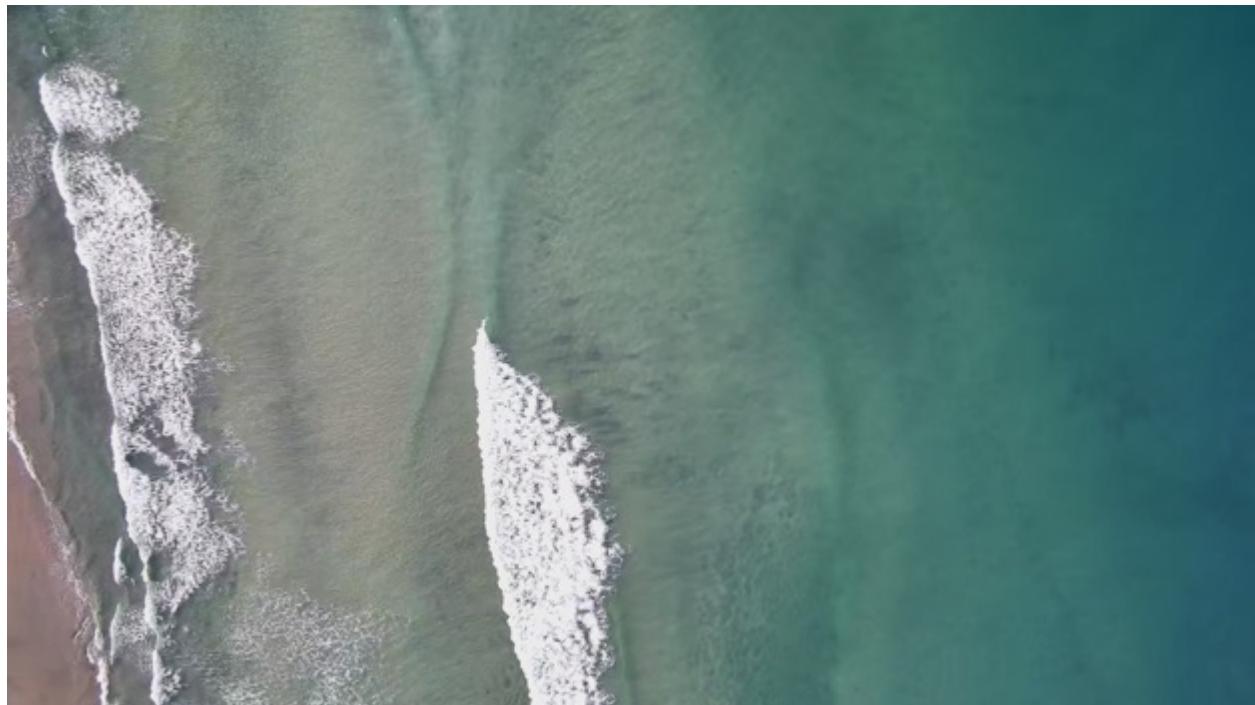
Photograph taken by the author

Scout mission #5



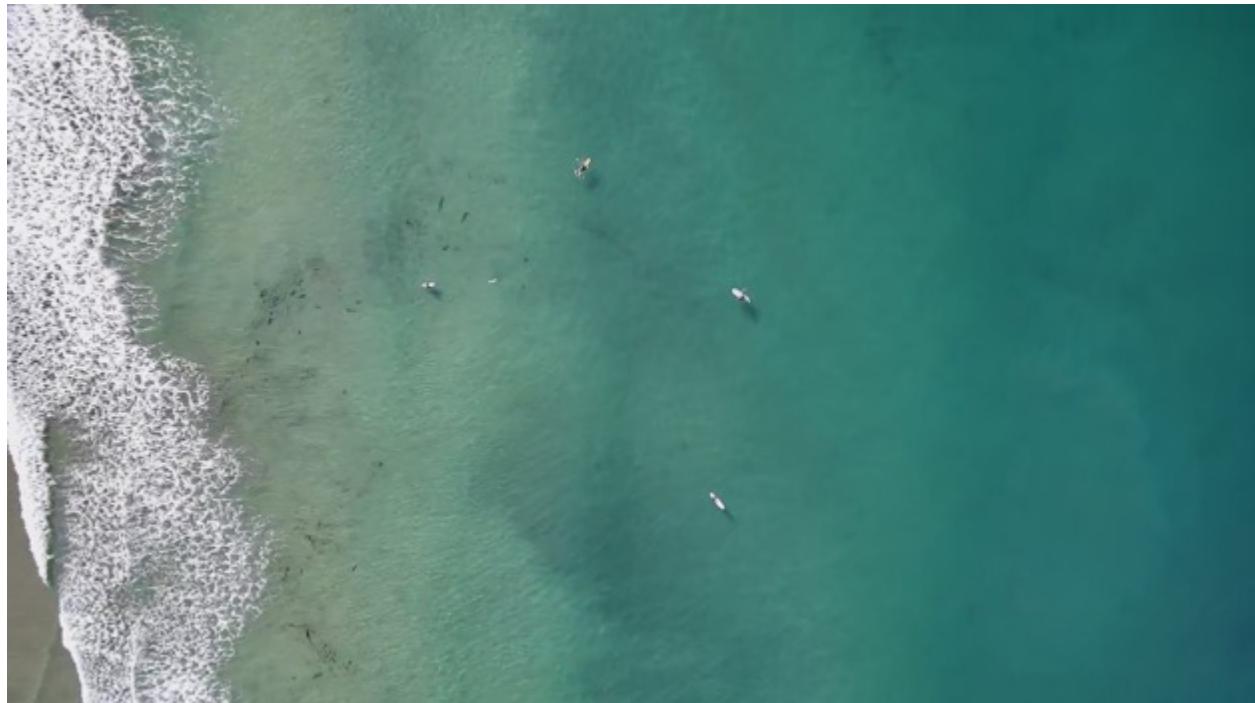
Photograph taken by the author

Scout mission #6



Photograph taken by the author

Scout mission #7



Photograph taken by the author

Scout mission #8



Photograph taken by the author

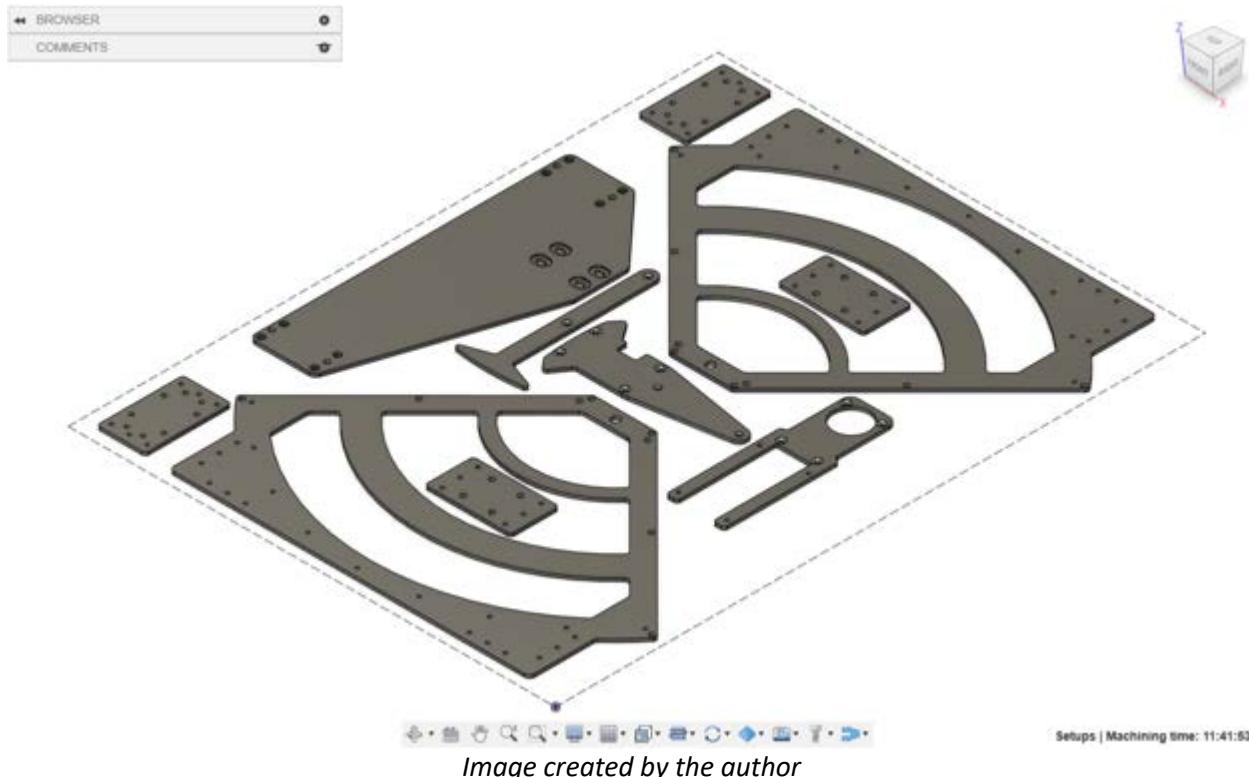
Scout mission #9



Photograph taken by the author

Appendix D. Fabrication templates

Pitch block (300x400x2mm)



Arm frame top and bottom (300x400x4mm)

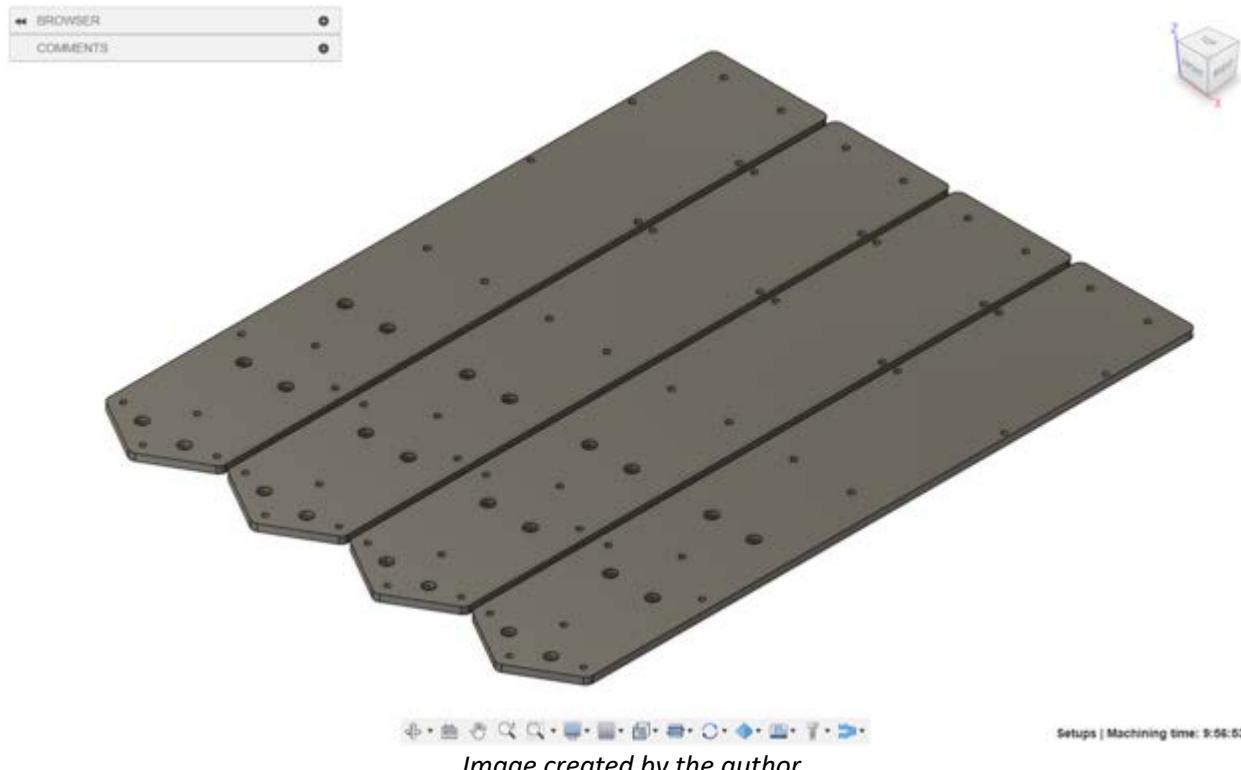


Image created by the author

Arm frame long vertical plate (300x400x4mm)

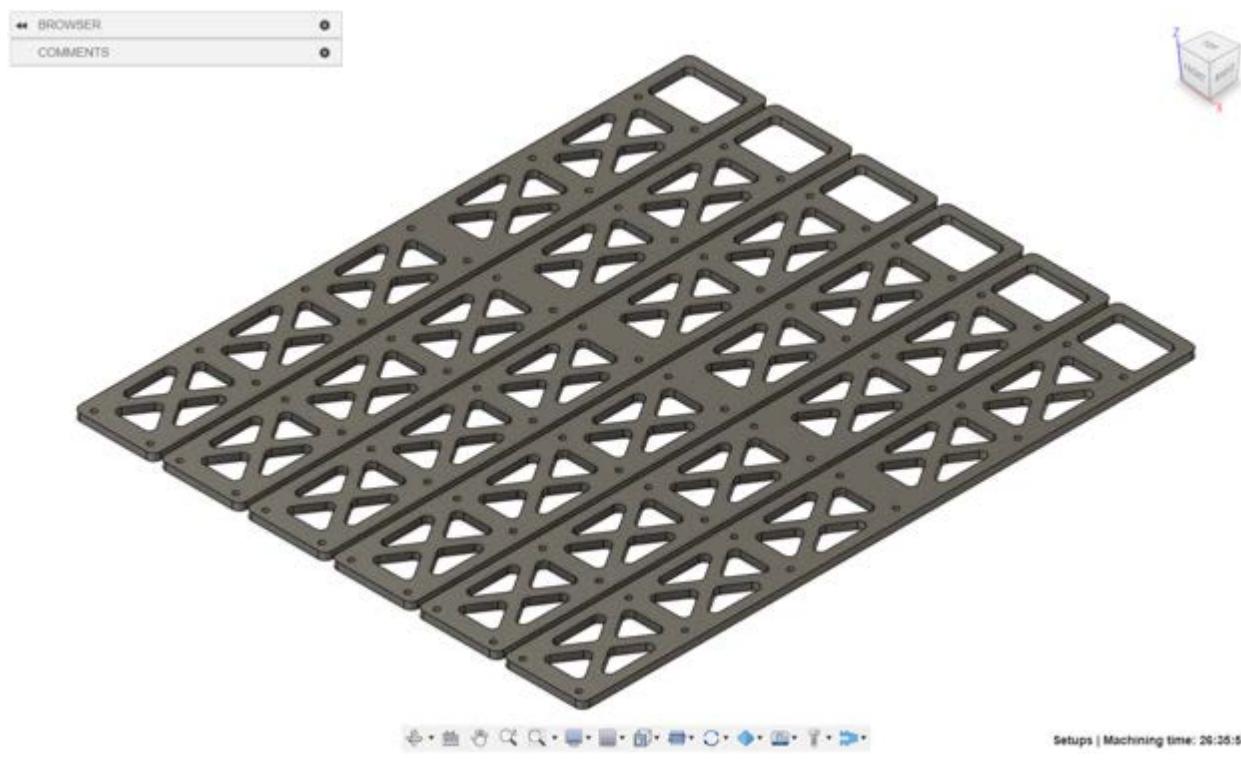
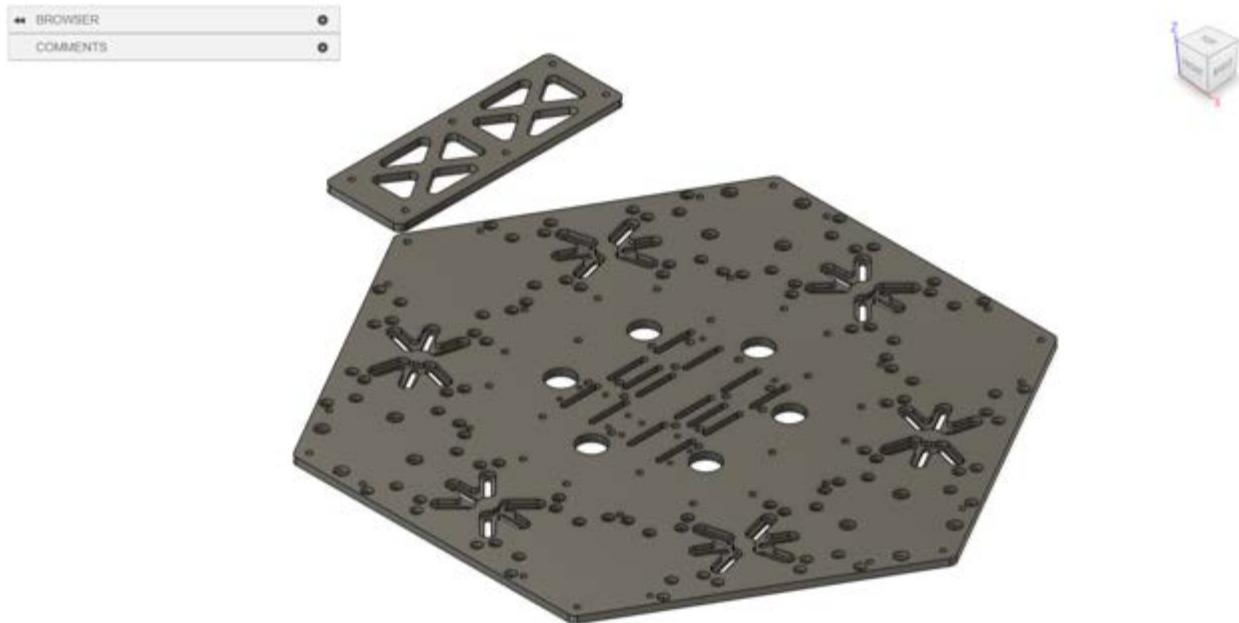


Image created by the author

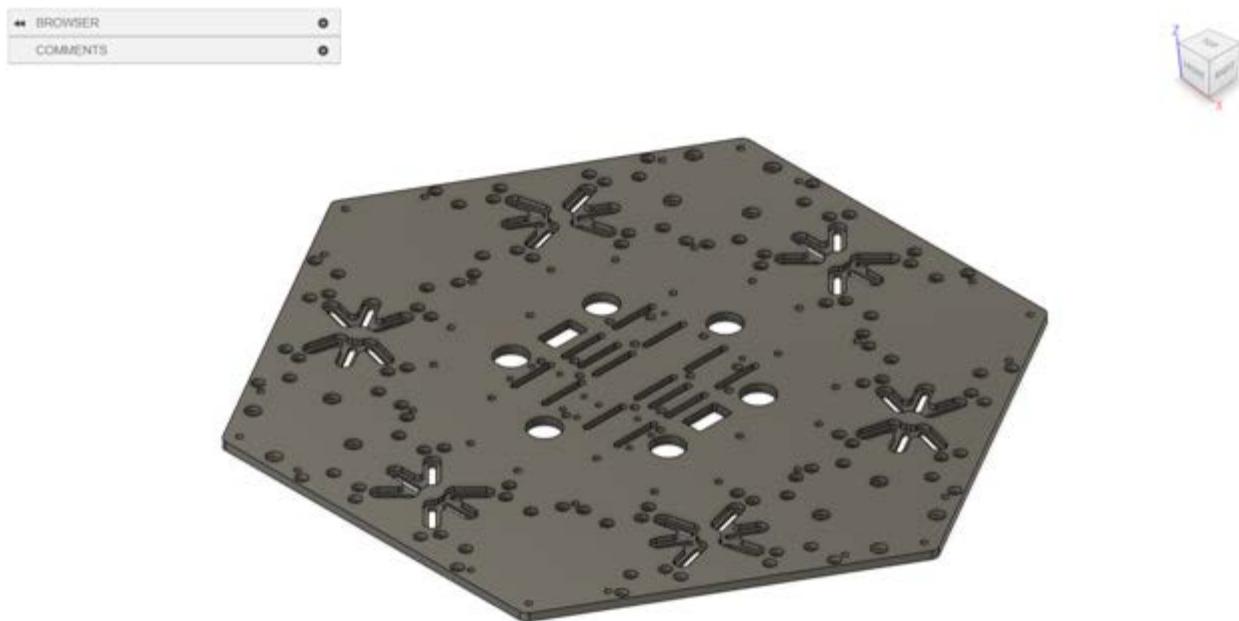
Body plate top and arm frame short vertical plate (300x400x4mm)



Setups | Machining time: 21:31:56

Image created by the author

Body plate bottom (300x400x4mm)



Setups | Machining time: 19:55:09

Image created by the author

Arm roll and pitch 3D part (300x400x2mm)

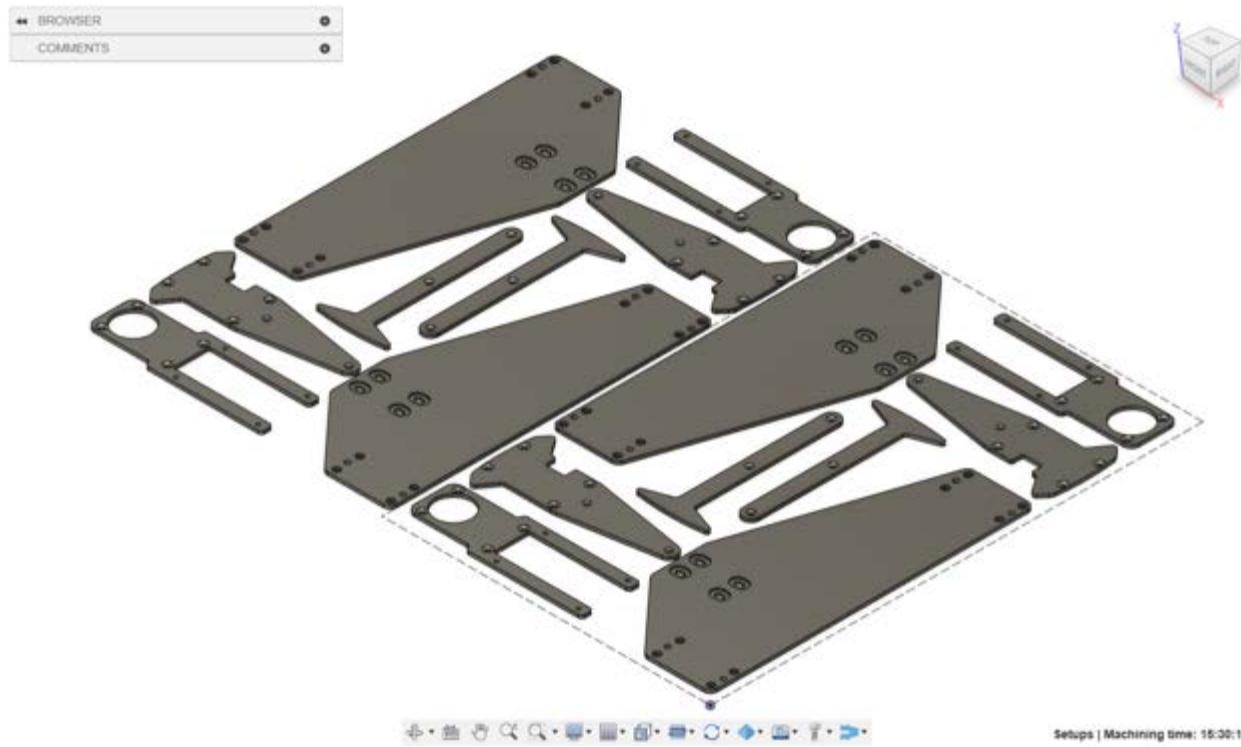


Image created by the author

Arm roll and pitch 3D part (200x400x2mm)

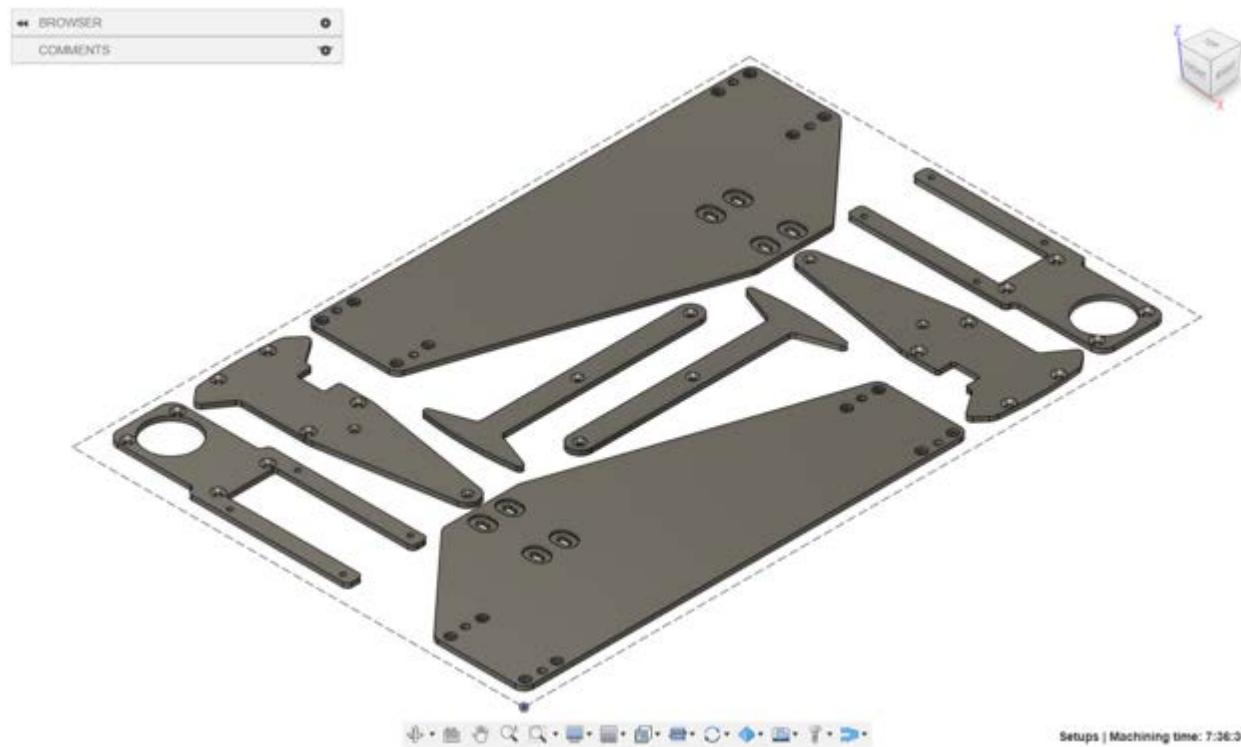


Image created by the author

Arm frame short vertical plate (200x300x4mm)



Image created by the author

3D print templates

Leg holder version 3

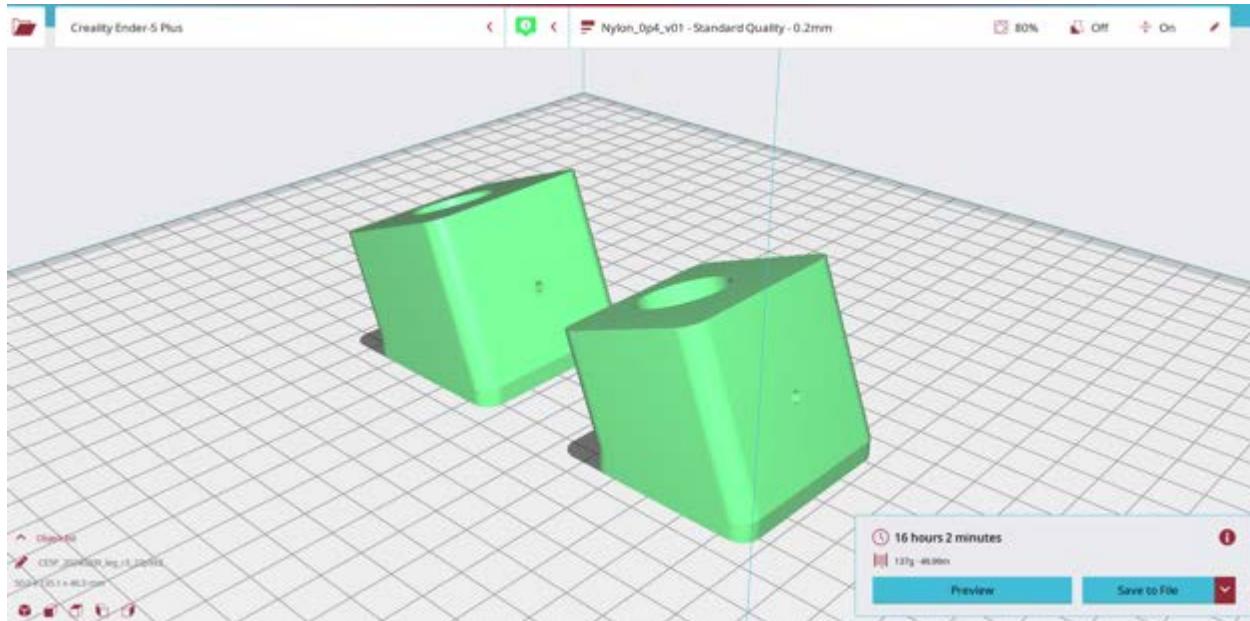


Image created by the author

Leg holder version 2

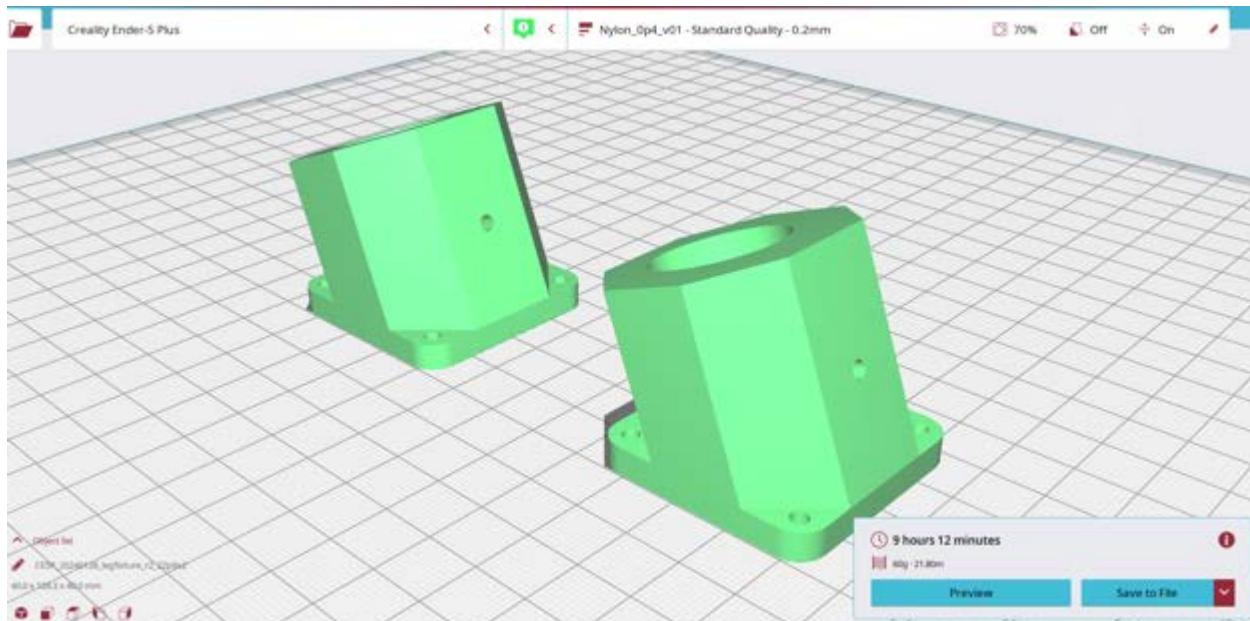


Image created by the author

Motor and ESC bracket

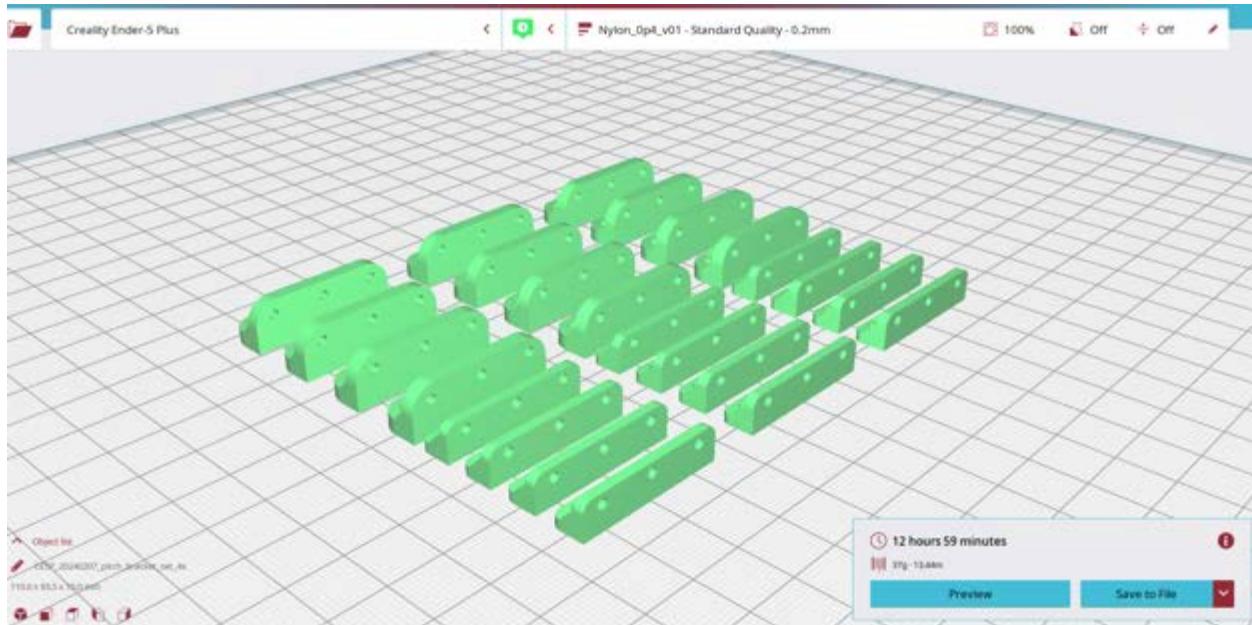


Image created by the author

Shoe holder

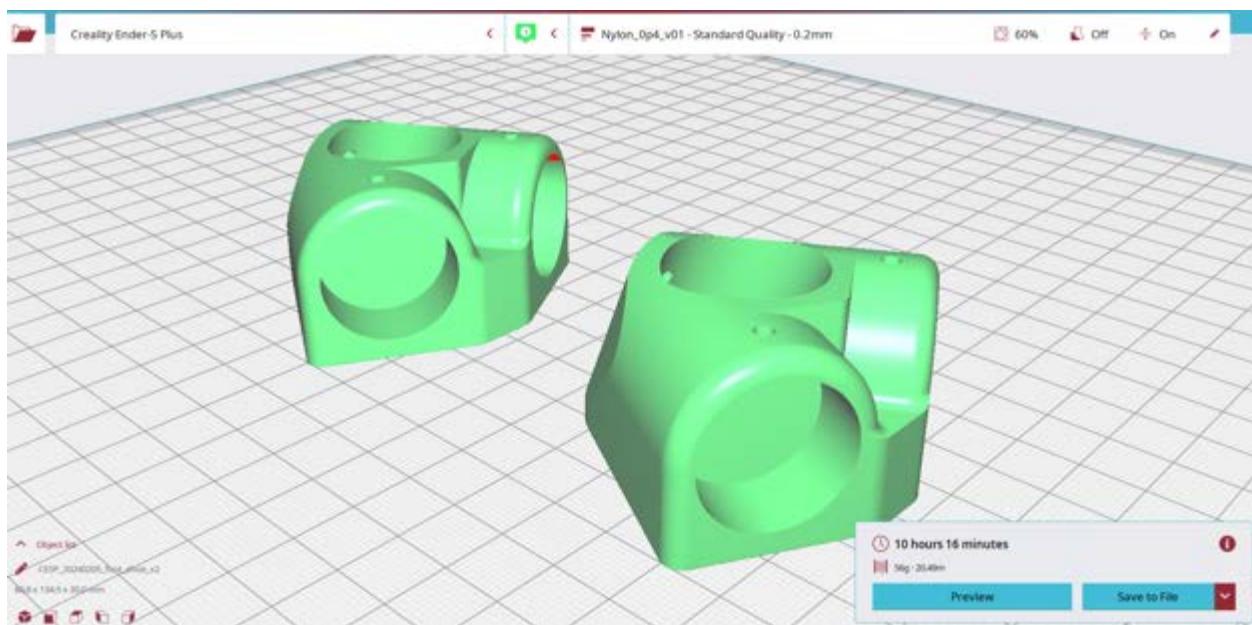


Image created by the author

Pitch axis end

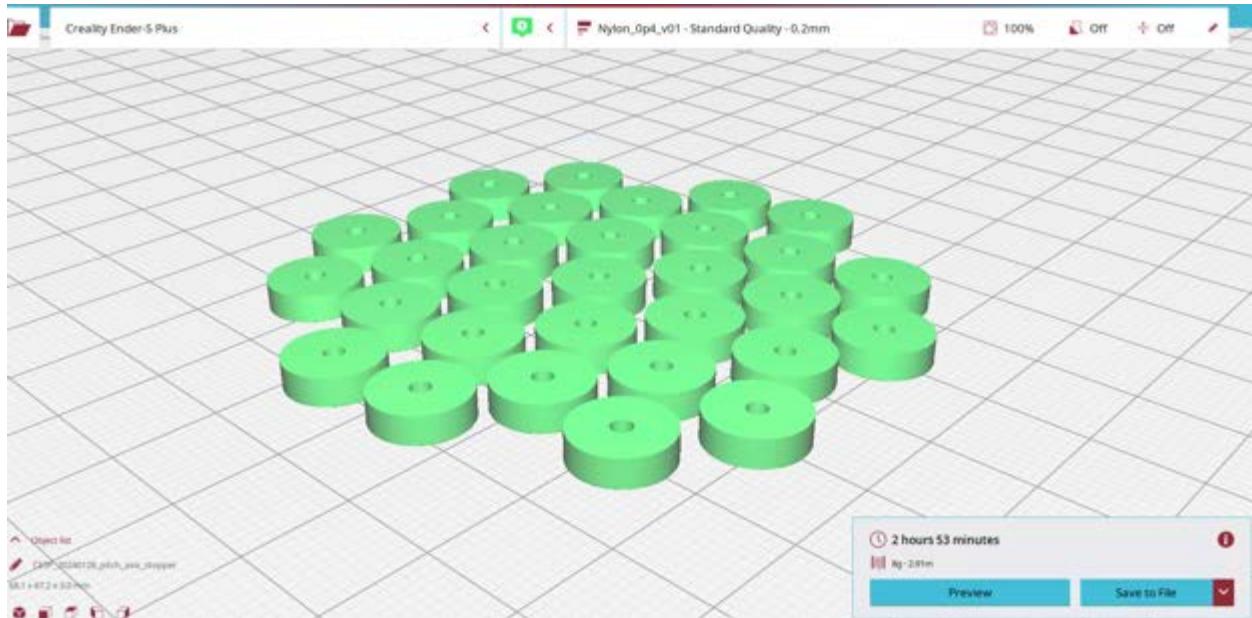


Image created by the author

JETSON Orin mount

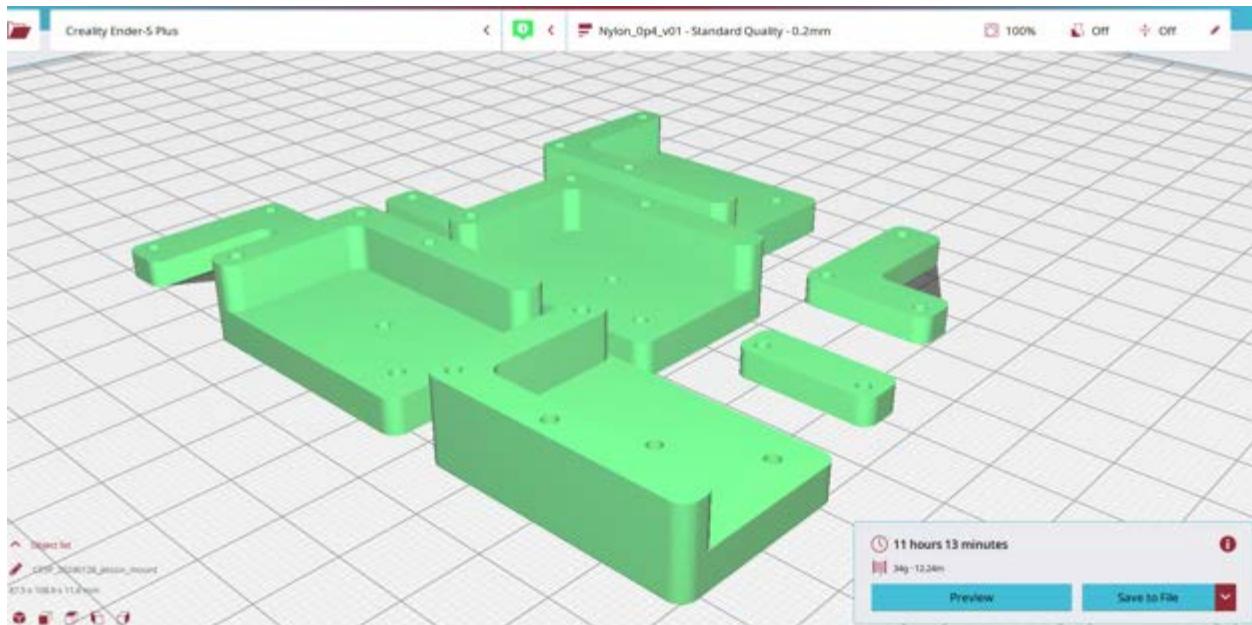


Image created by the author

Pitch axis fixture

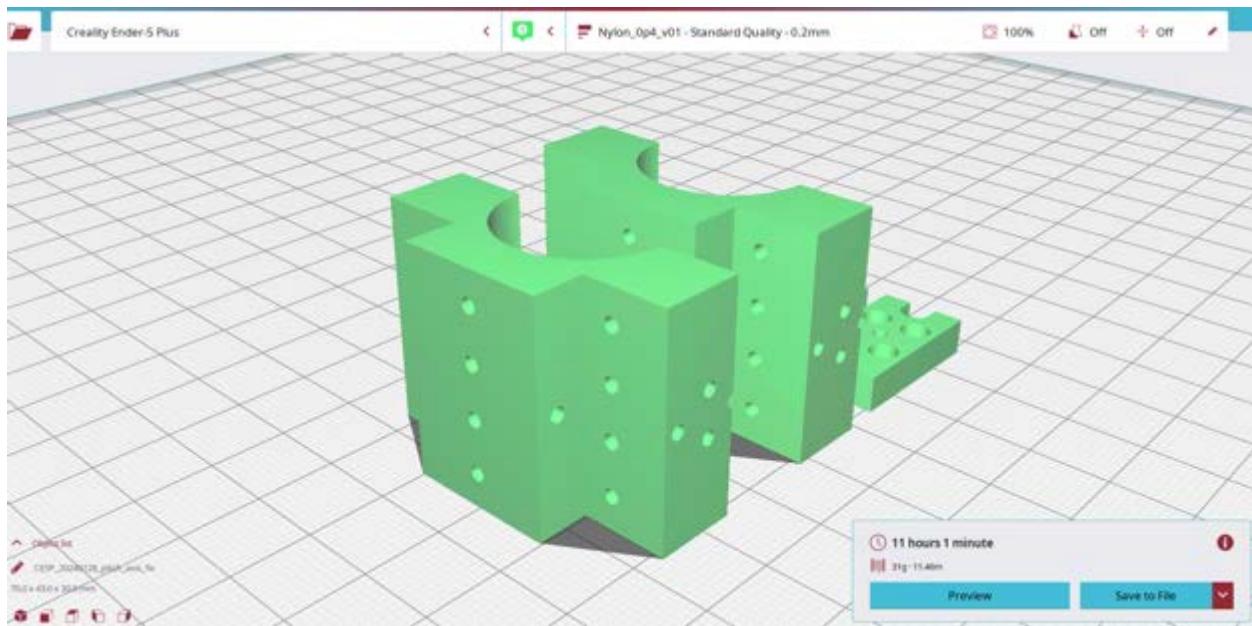


Image created by the author

Arm roll gear

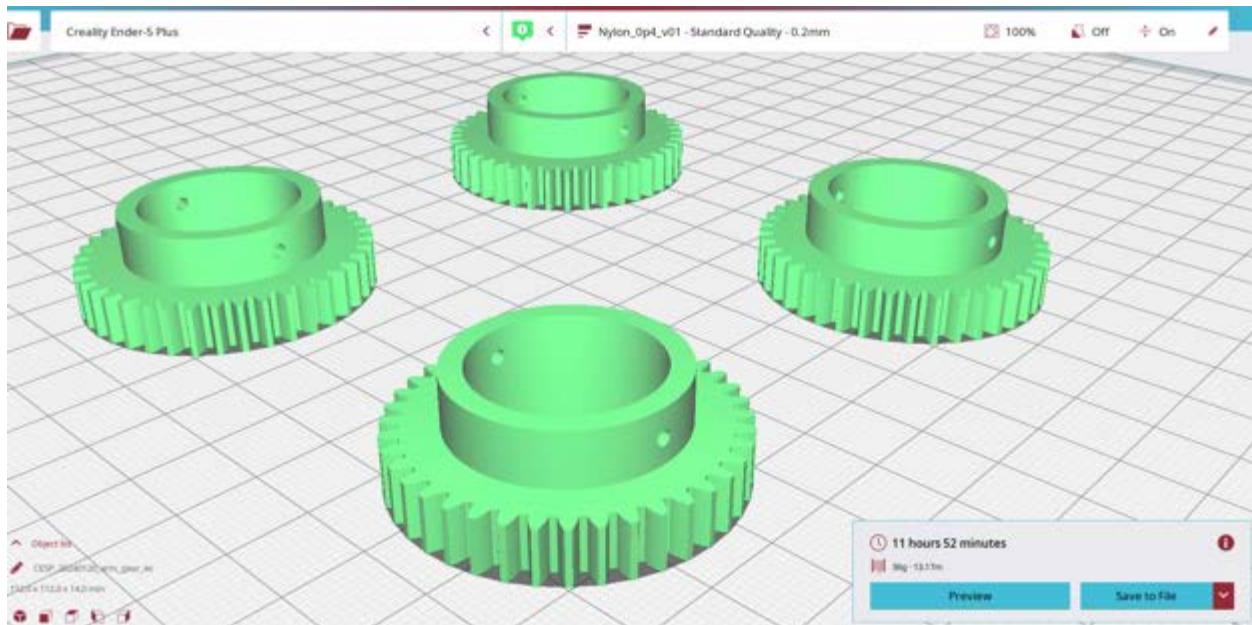


Image created by the author

Leg and shoe holder pair version 1

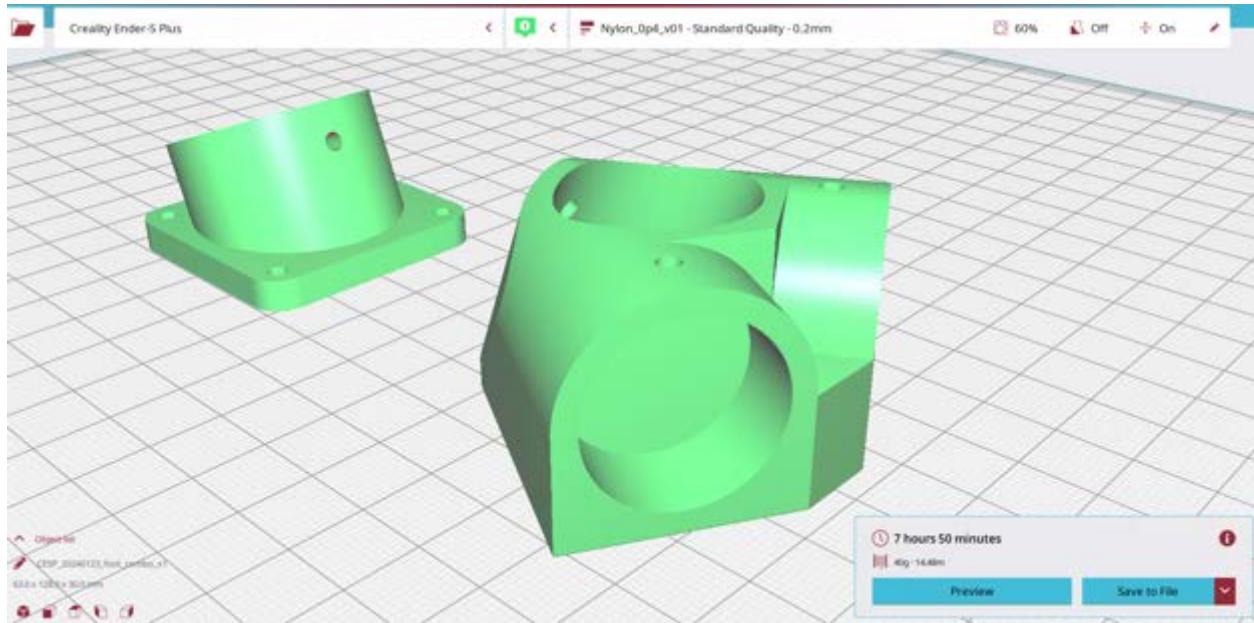


Image created by the author

Arm frame bracket

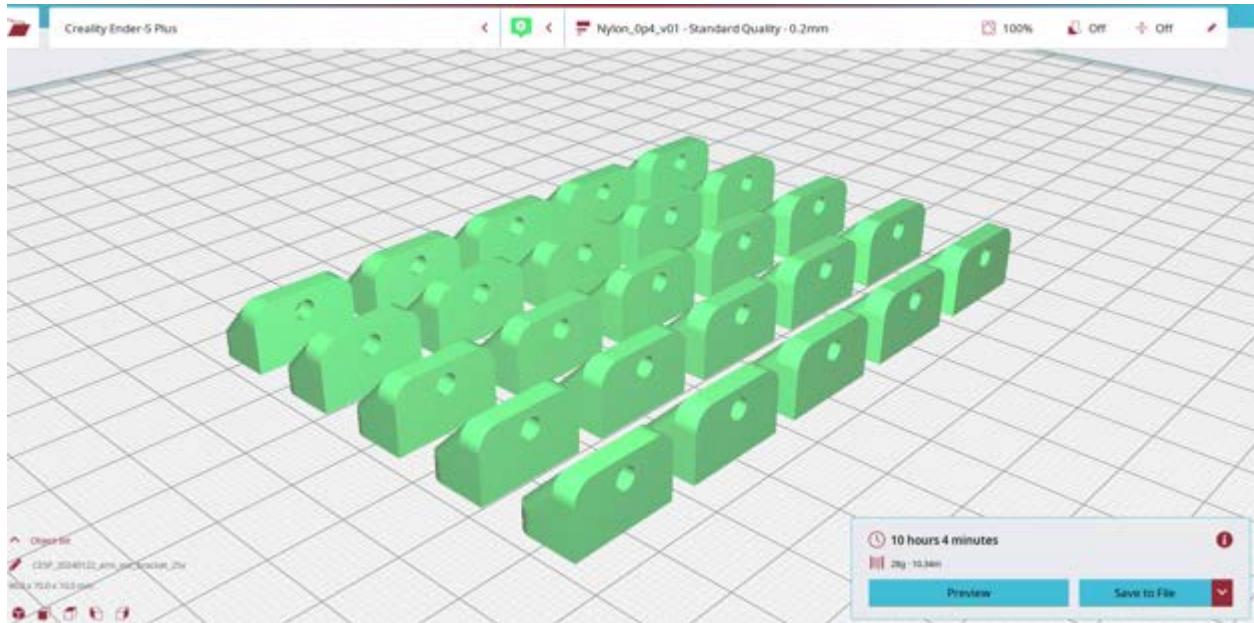


Image created by the author

Shoe holder version 0

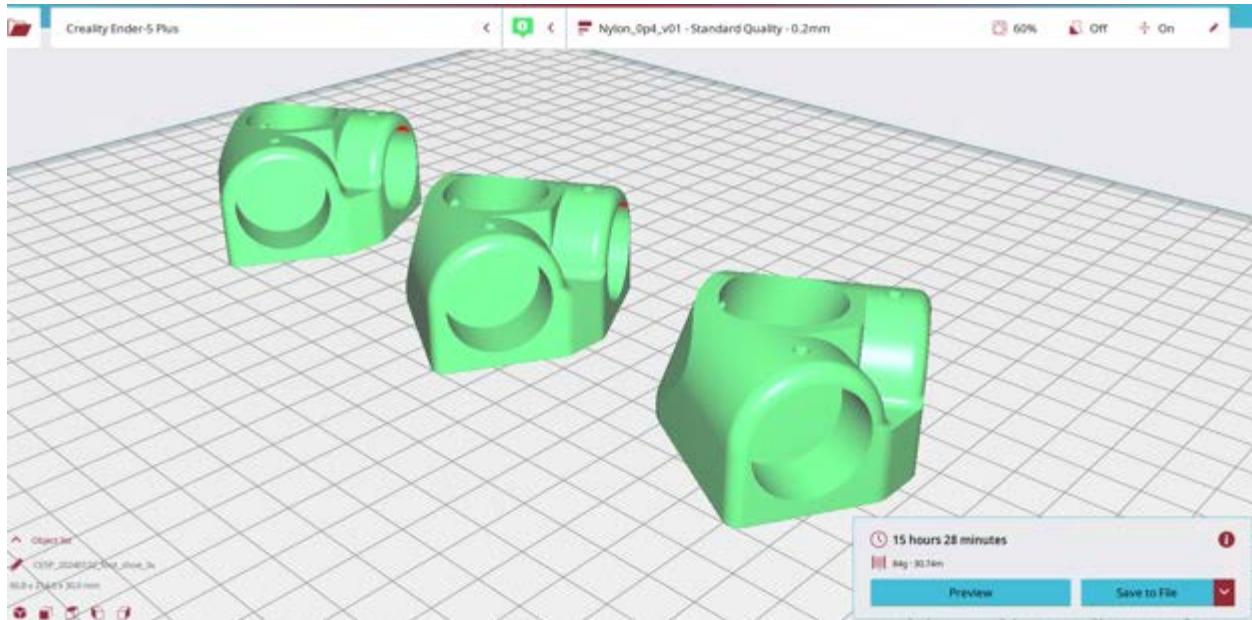


Image created by the author

Leg holder version 0

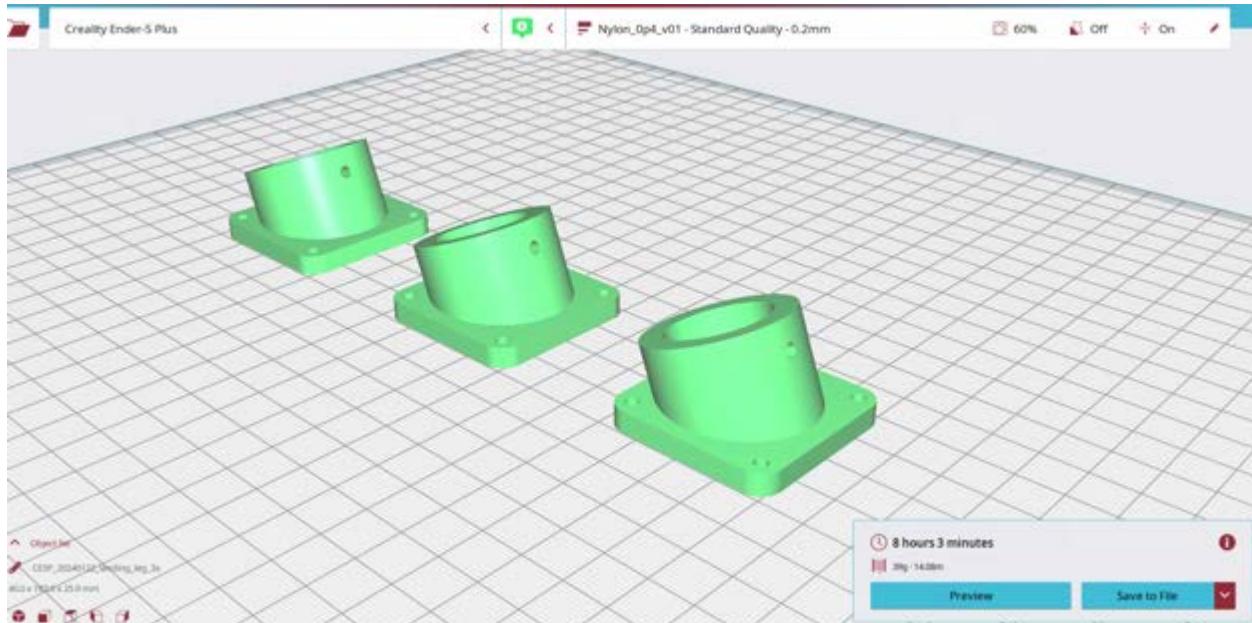


Image created by the author

Arm roll holder

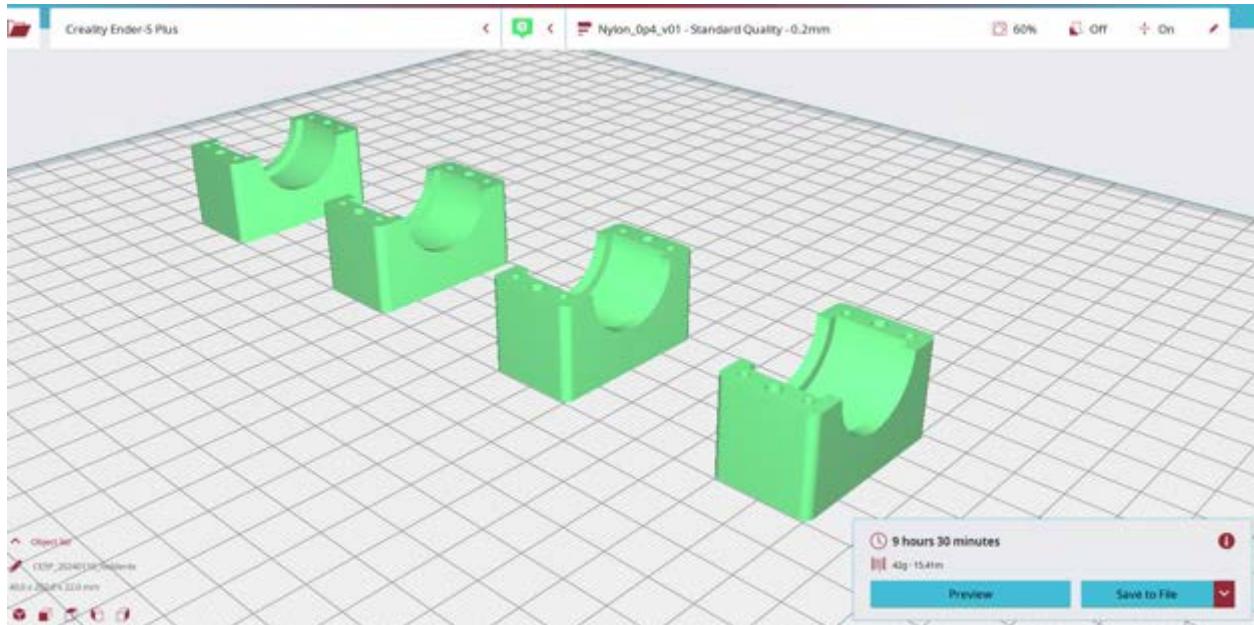


Image created by the author

Gear set

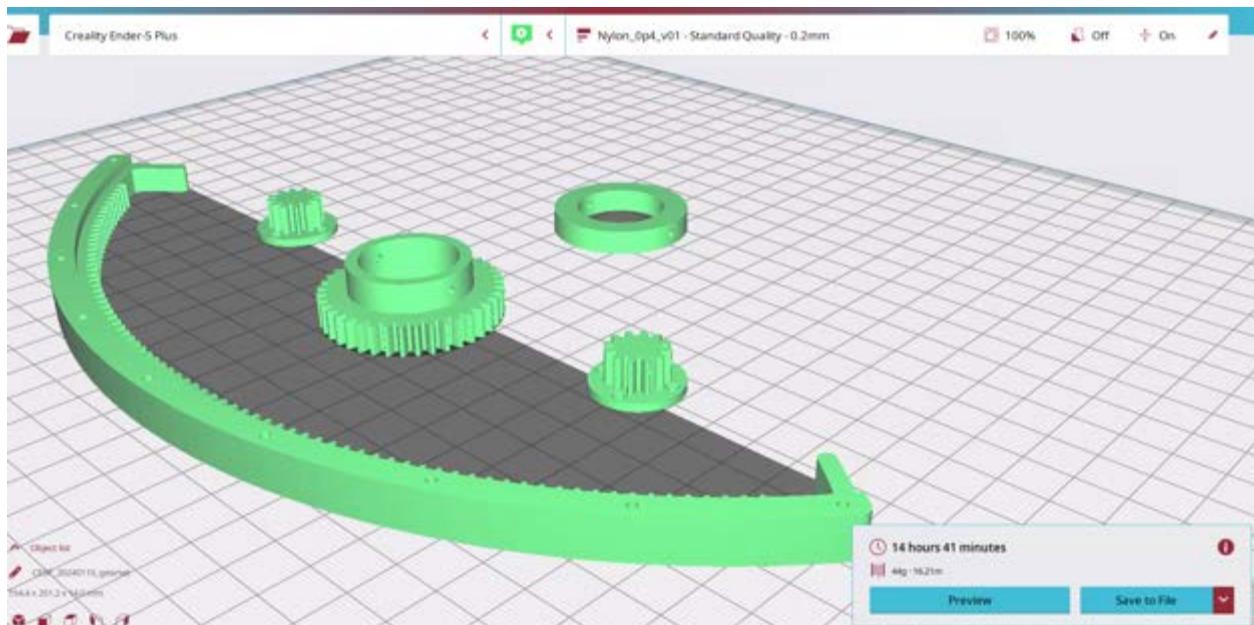


Image created by the author

Arm fixture set

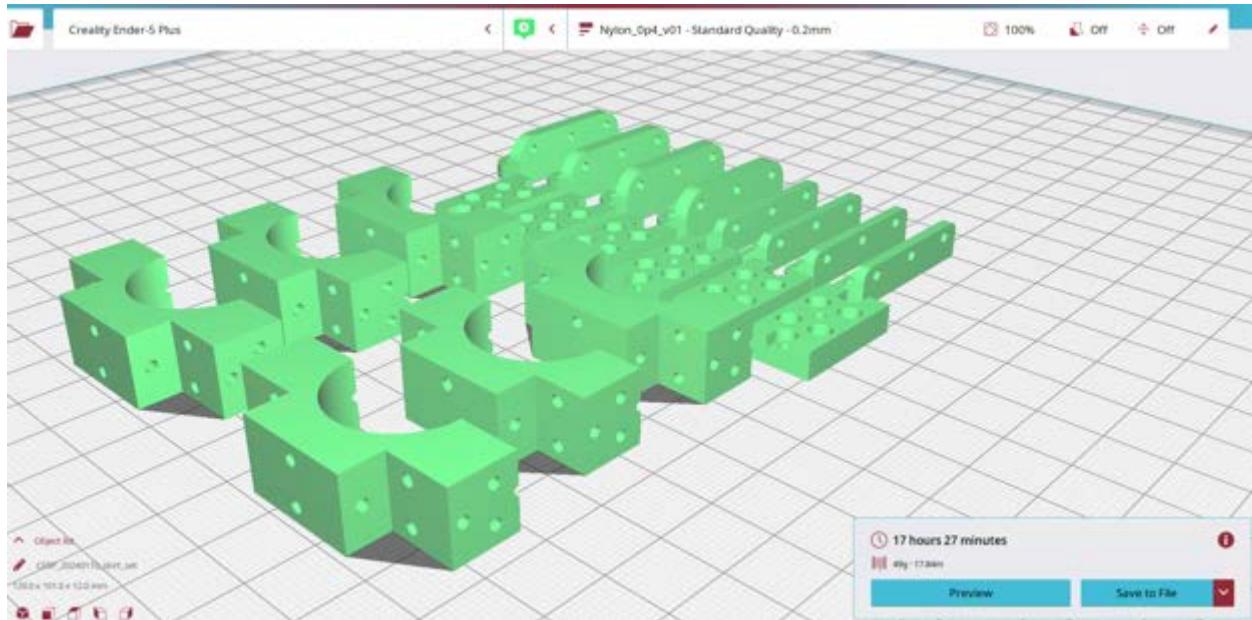


Image created by the author

Servo motor riser and spur gear set

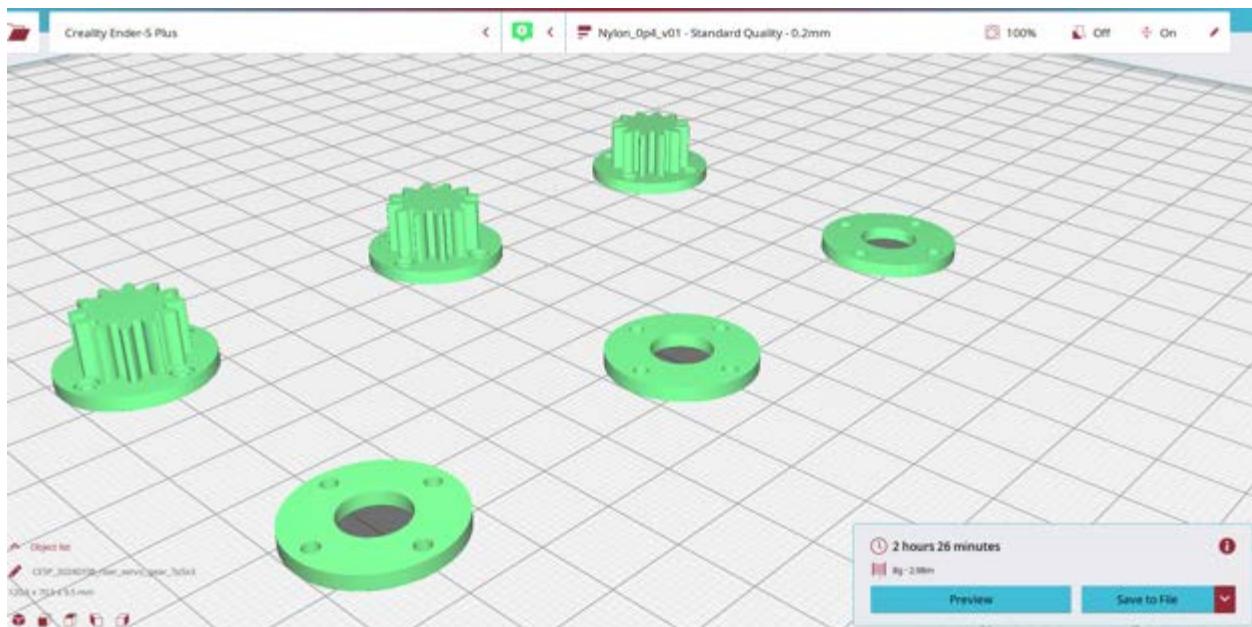


Image created by the author

Appendix E. Computer programs

All photographs were taken and images were created by the author

SCOUT DRONE IMAGE GEO-TAGGING CODE

px4_timestamp_v02.py

- Language: Python
- Host: NVIDIA JETSON Nano mission controller / Scout drone
- Usage:
 - Start this program before the scout mission to record GPS coordinates of images that saved to a folder
- Function:
 - Initiate MAVLink channel with Pixhawk flight controller through serial link
 - Request GPS coordinates to Pixhawk
 - Update GPS coordinate periodically
 - Maintain a log file
 - Monitor if there is any new image files
 - Log GPS coordinates and time stamp to all new images show up in the file system

```
from pymavlink import mavutil
import time, sys
import threading
import math
import os
import glob
import re
import numpy as np
import datetime

def wait_conn():
    """
    Sends a ping to stabilize the UDP communication and awaits for a response
    """
    msg = None
    while not msg:
        master.mav.ping_send(
            int(time.time() * 1e6), # Unix time in microseconds
            0, # Ping number
            0, # Request ping of all systems
            0 # Request ping of all components
        )
        msg = master.recv_match()
        print(msg)
        time.sleep(0.5)

def request_message_interval(message_id: int, frequency_hz: float):
    message = master.mav.command_long_encode(
```

```

        master.target_system, master.target_component,
        mavutil.mavlink.MAV_CMD_SET_MESSAGE_INTERVAL, 0,
        message_id, # The MAVLink message ID
        1e6 / frequency_hz, # The interval between two messages in microseconds. Set
        to -1 to disable and 0 to request default rate.
        0, 0, 0, 0, # Unused parameters
        0, # Target address of message stream (if message has target address fields).
0: Flight-stack default (recommended), 1: address of requestor, 2: broadcast.
    )
    master.mav.send(message)
    # Wait for a response (blocking) to the MAV_CMD_SET_MESSAGE_INTERVAL command and
print result
    response = master.recv_match(type='COMMAND_ACK', blocking=True)
    if response and response.command == mavutil.mavlink.MAV_CMD_SET_MESSAGE_INTERVAL
and response.result == mavutil.mavlink.MAV_RESULT_ACCEPTED:
        print("Command accepted")
    else:
        print("Command failed")

def request_message():
    request_message_interval(mavutil.mavlink.MAVLINK_MSG_ID_UTM_GLOBAL_POSITION, 20)
#340
    #request_message_interval(mavutil.mavlink.MAVLINK_MSG_ID_OPEN_DRONE_ID_BASIC_ID,
1) #12900
    #request_message_interval(mavutil.mavlink.MAVLINK_MSG_ID_OPEN_DRONE_ID_LOCATION,
1) #12901

def get_telemetry_data():
    ack_msg = master.recv_match(type='COMMAND_ACK')
    observation=[]
    t_param='UTM_GLOBAL_POSITION' #340
    try:
        UTM_GLOBAL_POSITION_lat = master.messages[t_param].lat
        UTM_GLOBAL_POSITION_lon = master.messages[t_param].lon
        UTM_GLOBAL_POSITION_alt = master.messages[t_param].alt
        UTM_GLOBAL_POSITION_vx = master.messages[t_param].vx
        UTM_GLOBAL_POSITION_vy = master.messages[t_param].vy
        UTM_GLOBAL_POSITION_vz = master.messages[t_param].vz

    UTM_GLOBAL_POSITION_data=[UTM_GLOBAL_POSITION_lat,UTM_GLOBAL_POSITION_lon,UTM_GLOBAL_P
OSITION_alt,
    UTM_GLOBAL_POSITION_vx,UTM_GLOBAL_POSITION_vy,UTM_GLOBAL_POSITION_vz]
    observation.extend(UTM_GLOBAL_POSITION_data) #count=45+3=48
except:
    print(t_param, ":", 'No message received')
return observation

def get_open_drone_id():
    ack_msg = master.recv_match(type='COMMAND_ACK')
    t_param='OPEN_DRONE_ID_BASIC_ID' #340
    try:
        OPEN_DRONE_ID_BASIC_ID_id_or_mac = master.messages[t_param].id_or_mac
        OPEN_DRONE_ID_BASIC_ID_uas_id = master.messages[t_param].uas_id
        print("OPEN_DRONE_ID_BASIC_ID_id_or_mac: ", OPEN_DRONE_ID_BASIC_ID_id_or_mac)
        print("OPEN_DRONE_ID_BASIC_ID_uas_id: ", OPEN_DRONE_ID_BASIC_ID_uas_id)
    except:
        print(t_param, ":", 'No message received')
    t_param='OPEN_DRONE_ID_LOCATION' #340
    try:
        OPEN_DRONE_ID_LOCATION_id_or_mac = master.messages[t_param].id_or_mac
        print("OPEN_DRONE_ID_LOCATION_id_or_mac: ", OPEN_DRONE_ID_LOCATION_id_or_mac)
    except:

```

```

print(t_param, ":", 'No message received')

print("Initiate mavlink")
boot_time = time.time()
#master = mavutil.mavlink_connection("/dev/ttyTHS1", baud=115200)
master = mavutil.mavlink_connection("udpout:192.168.144.19:14540")
print("Wait for connection")
wait_conn()
print("Wait for heart beat")
master.wait_heartbeat()
print("Heartbeat: (system %u component %u)" % (master.target_system,
master.target_component))
print("Request messages")
request_message()
print("1 second stand by")
time.sleep(1)
observation=[]
count=0
while len(observation)==0:
    observation=get_telemetry_data()
    print("UTM data: ", observation)
    time.sleep(1)
    count=count+1
    if count>10:
        request_message()
        print("Request messages")
        count=0

camera0_prefix='camera0_'
gimbal0_prefix='gimbal0_'
frame_rate=2
count_camera0_files=0
count_gimbal0_files=0
len_camera0_files=0
len_gimbal0_files=0
now=datetime.datetime.now()

root_path='/home/akim/px4_video/'
#dirFiles=os.listdir('/home/akim/px4_video/')

while len_camera0_files == 0 and len_gimbal0_files == 0:
    camera0_files=glob.glob(root_path+camera0_prefix+'*.jpg')
    camera0_files.sort(key=lambda f: int(''.join(filter(str.isdigit, f))))
    len_camera0_files=len(camera0_files)

    gimbal0_files=glob.glob(root_path+gimbal0_prefix+'*.jpg')
    gimbal0_files.sort(key=lambda f: int(''.join(filter(str.isdigit, f))))
    len_gimbal0_files=len(gimbal0_files)

print("image files detected")
now=datetime.datetime.now()
file=open(root_path+"timestamp_camera0.txt", "a")
file.write(str(now))
file.write('\n')
file.close()
file=open(root_path+"timestamp_gimbal0.txt", "a")
file.write(str(now))
file.write('\n')
file.close()

while True:
    camera0_files=glob.glob(root_path+camera0_prefix+'*.jpg')
    camera0_files.sort(key=lambda f: int(''.join(filter(str.isdigit, f))))
```

```

len_camera0_files=len(camera0_files)

gimbal0_files=glob.glob(root_path+gimbal0_prefix+'*.jpg')
gimbal0_files.sort(key=lambda f: int(''.join(filter(str.isdigit, f))))
len_gimbal0_files=len(gimbal0_files)

if count_camera0_files < len_camera0_files:
    UTM_data=get_telemetry_data()
    now=datetime.datetime.now()
    #print(camera0_files[len_camera0_files-1])
    filepath=camera0_files[len_camera0_files-1]
    filename=filepath[len(root_path)+len(camera0_prefix):len(filepath)] # remove
root path from file name
    #print(filename)
    fileidx=re.sub('\D','', filename)
    #print(fileidx)
    data_to_write=[fileidx,str(now.time())]
    data_to_write.extend(map(str, UTM_data))
    print(data_to_write)
    #UTM_record_camera0.append(fileidx)
    file=open(root_path+"timestamp_camera0.txt", "a")
    file.write(','.join(data_to_write))
    file.write('\n')
    file.close()
    count_camera0_files = len_camera0_files

if count_gimbal0_files < len_gimbal0_files:
    UTM_data=get_telemetry_data()
    now=datetime.datetime.now()
    #print(gimbal0_files[len_gimbal0_files-1])
    filepath=gimbal0_files[len_gimbal0_files-1]
    filename=filepath[len(root_path)+len(gimbal0_prefix):len(filepath)] # remove
root path from file name
    #print(filename)
    fileidx=re.sub('\D','', filename)
    #print(fileidx)
    data_to_write=[fileidx,str(now.time())]
    data_to_write.extend(map(str, UTM_data))
    print(data_to_write)
    #UTM_record_camera0.append(fileidx)
    #UTM_record_gimbal0.append(fileidx)
    file=open(root_path+"timestamp_gimbal0.txt", "a")
    file.write(','.join(data_to_write))
    file.write('\n')
    file.close()
    count_gimbal0_files = len_gimbal0_files

```

SCOUT DRONE VIDEO STORING AND STREAMING CODE

px4_start_gstreamer.sh

- Language: Linux Shell command
- Host: NVIDIA JETSON Nano mission controller / Scout drone
- Usage:
 - Start this program before the scout mission to start image collection and streaming flow
- Function:
 - Initiate image stream from IP camera gimbal and local Raspberry Pi camera module 3
 - Start saving images with index
 - Consolidate two images and reduce size
 - Send the consolidated image to ground control computer through IP streaming

```
gst-launch-1.0 \
  uridecodebin uri=rtsp://192.168.144.25:8554/main.264 source::latency=0 \
! tee name=t0_gimbal0 \
! nvvidconv \
! videorate \
! "video/x-raw(memory:NVMM),framerate=1/1" \
! nvvidconv \
! nvjpegenc \
! multifilesink location=~/px4_video/gimbal0_%d.jpg \
  nvarguscamerasrc sensor-id=0 \
! 'video/x-raw(memory:NVMM),width=4608,height=2592,framerate=10/1,format=NV12' \
! tee name=t0_camera0 \
! nvvidconv \
! videorate \
! "video/x-raw,framerate=1/1" \
! nvvidconv \
! nvjpegenc \
! multifilesink location=~/px4_video/camera0_%d.jpg \
  t0_gimbal0. \
! nvvidconv \
! 'video/x-raw(memory:NVMM),width=960,height=540,format=RGBA' \
! queue ! comp. \
  t0_camera0. \
! nvvidconv \
! 'video/x-raw(memory:NVMM),width=960,height=540,format=RGBA' \
! queue ! comp. \
  nvcompositor name=comp \
  sink_0::xpos=0 sink_0::ypos=0    sink_0::width=960 sink_0::height=540 \
  sink_1::xpos=0 sink_1::ypos=540  sink_1::width=960 sink_1::height=540 \
! nvvidconv \
! "video/x-raw(memory:NVMM),format=NV12" \
! videorate \
! "video/x-raw(memory:NVMM),framerate=10/1" \
! nvvidconv \
! nvjpegenc \
! rtpjpegpay \
! udpsink host=192.168.144.10 port=5010
```

SCOUT DRONE CAMERA GIMBAL SET UP CODE

a8_udp_control_v02.py

- Language: Python
- Host: NVIDIA JETSON Nano mission controller / Scout drone
- Usage:
 - Start this program before the scout mission to orient camera gimbal downward for beach image recording
- Function:
 - Initiate UDP connection to IP camera gimbal
 - Send a command to camera to look down with CRC check sum

```
#a8_udp_control_v02.py
import socket, sys

Camera1_IP='192.168.144.25'
Camera_UDP_Port=37260

# Create a UDP socket
Socket_Camer1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# Bind the socket to the port
Cameral_Address = (Camera1_IP, Camera_UDP_Port)
print("Camera IP address and port: ", Cameral_Address)

#UDP_MESSAGE = bytes.fromhex("55 66 01 01 00 00 00 08 01 d1 12") #Centering
#UDP_MESSAGE = bytes.fromhex("55 66 01 01 00 00 00 0c 03 57 fe") #Lock mode
#UDP_MESSAGE = bytes.fromhex("55 66 01 01 00 00 00 0c 04 b0 8e") #Follow mode
#UDP_MESSAGE = bytes.fromhex("55 66 01 04 00 00 00 0e 00 00 84 03 d7 20") #90 degree
pitch
#UDP_MESSAGE = bytes.fromhex("55 66 01 00 00 00 00 0d e8 05") #Acquire additude data

UDP_MESSAGE = bytes.fromhex("55 66 01 04 00 00 00 0e 00 00 7c fc 4f a4") #-90 degree
pitch
print("Data to camera : ", UDP_MESSAGE)
Socket_Camer1.sendto(UDP_MESSAGE, Cameral_Address) #Send message to UDP port
#Receive data from camera
data, address = Socket_Camer1.recvfrom(Camera_UDP_Port)
print("Data from camera: ", data, address)

UDP_MESSAGE = bytes.fromhex("55 66 01 01 00 00 00 0c 05 91 9e") #FPV mode
print("Data to camera : ", UDP_MESSAGE)
Socket_Camer1.sendto(UDP_MESSAGE, Cameral_Address) #Send message to UDP port
#There is no ack from A8 by definition
#Receive data from camera
#data, address = Socket_Camer1.recvfrom(Camera_UDP_Port)
#print("Data from camera: ", data, address)
```

GROUND CONTROL STATION VIDEO DISPLAY FROM SCOUT DRONE

px4_start_gstreamer_sink.sh

- Language: Linux Shell command
- Host: Ground control computer / Scout drone mission
- Usage:
 - Start this program before or during the scout mission to start video stream from the scout drone
 - It can be started any time as long as the scout drone has streaming turned on
- Function:
 - Make video sink to display UDP video streaming from the scout drone

```
gst-launch-1.0 \
  udpsrc port=5010 \
! application/x-rtp,encoding-name=JEPD \
! rtpjpegdepay \
! jpegdec \
! autovideosink
```

RESCUE DRONE TILT ROTOR SERVO MOTOR ABSTRACTION

dxl_uav_class.py

- Language: Python
 - Host: NVIDIA JETSON Orin Nano mission controller / Rescue drone
 - Usage:
 - Dynamixel servo control class
 - Used by **uav_control.py**
 - Function:
 - Abstract Dynamixel servo motor instance
 - Turn on / off servo motor torque
 - Read current position
 - Set goal position

```

#!/usr/bin/env python
import os, sys, tty, termios
from dynamixel_sdk import *

class dxl_uav:
    def __init__(self, ID):
        # Control table address
        # anything without "self." will not be used
        # just there for aesthetical purposes & as a reminder
        PROTOCOL_VERSION = 2.0
        self.ADDR_TORQUE_ENABLE = 64
        ADDR_LED_RED = 65
        ADDR_OPERATING_MODE = 11
        OP_VELOCITY_CTRL_MODE = 1
        self.ADDR_GOAL_VELOCITY = 104
        ADDR_VELOCITY_LIMIT = 44
        VELOCITY_LIMIT = 265
        self.ADDR_PRESENT_VELOCITY = 128
        LEN_LED_RED = 1
        self.ADDR_GOAL_POSITION = 116
        LEN_GOAL_POSITION = 4
        self.ADDR_PRESENT_POSITION = 132
        self.ADDR_OPERATING_MODE = 4
        LEN_PRESENT_POSITION = 4
        DXL_MINIMUM_POSITION_VALUE = 0
        self.MAXIMUM_POSITION_VALUE = 4095
        self.BAUDRATE = 57600
        self.DXL_ID = ID # Dynamixel#1 ID : 1
        self.orientation = 1 #1 or -1
        self.PRESENT_POSITION = 0
        self.POSITION_OFFSET = 0
        self.GOAL_POSITION = 0

        DEVICENAME = '/dev/ttyUSB0'
        self.TORQUE_ENABLE = 1 # Value for enabling the
torque
        self.TORQUE_DISABLE = 0 # Value for disabling the
torque
        #DXL_MOVING_STATUS_THRESHOLD = 20 # Dynamixel moving status
threshold

```

```

# Initialize PortHandler and PacketHandler
self.portHandler = PortHandler(DEVICENAME)
self.packetHandler = PacketHandler(PROTOCOL_VERSION)
# Open port
if self.portHandler.openPort():
    print(self.DXL_ID, ":Succeeded to open the port")
else:
    print(self.DXL_ID, ":Failed to open the port.")
# Set port baudrate
if self.portHandler.setBaudRate(self.BAUDRATE):
    print(self.DXL_ID, ":Succeeded to change the baudrate to ", self.BAUDRATE)
else:
    print(self.DXL_ID, ":Failed to change the baudrate.")

#turn on motor torque
def torqueOn(self):
    dxl_comm_result, dxl_error =
self.packetHandler.write1ByteTxRx(self.portHandler, self.DXL_ID,
self.ADDR_TORQUE_ENABLE, self.TORQUE_ENABLE)
    if dxl_comm_result != COMM_SUCCESS:
        print("torqueOn:%s" % self.packetHandler.getTxRxResult(dxl_comm_result))
    elif dxl_error != 0:
        print("torqueOn:%s" % self.packetHandler.getRxPacketError(dxl_error))

#turn off motor torque
def torqueOff(self):
    # Enable Dynamixel Torque
    dxl_comm_result, dxl_error =
self.packetHandler.write1ByteTxRx(self.portHandler, self.DXL_ID,
self.ADDR_TORQUE_ENABLE, self.TORQUE_DISABLE)
    if dxl_comm_result != COMM_SUCCESS:
        print("torqueOff:%s" % self.packetHandler.getTxRxResult(dxl_comm_result))
    elif dxl_error != 0:
        print("torqueOff:%s" % self.packetHandler.getRxPacketError(dxl_error))

#close motor IO port
def closePort(self):
    self.portHandler.closePort()

#get current motor position
def getCurrentPotision(self):
    self.PRESENT_POSITION, dxl_comm_result, dxl_error =
self.packetHandler.read4ByteTxRx(self.portHandler, self.DXL_ID,
self.ADDR_PRESENT_POSITION)
    if dxl_comm_result != COMM_SUCCESS:
        print(self.DXL_ID, ":getCurrentPotision:1:%s" %
self.packetHandler.getTxRxResult(dxl_comm_result))
    elif dxl_error != 0:
        print(self.DXL_ID, ":getCurrentPotision:2:%s" %
self.packetHandler.getRxPacketError(dxl_error))

#set motor position goal
def setGoalPotision(self, GOAL_POSITION_TARGET):
    self.GOAL_POSITION=GOAL_POSITION_TARGET
    dxl_comm_result, dxl_error =
self.packetHandler.write4ByteTxRx(self.portHandler, self.DXL_ID,
self.ADDR_GOAL_POSITION, self.GOAL_POSITION+self.POSITION_OFFSET)
    if dxl_comm_result != COMM_SUCCESS:
        print(self.DXL_ID, ":setGoalPotision:1:%s" %
self.packetHandler.getTxRxResult(dxl_comm_result))
    elif dxl_error != 0:
        print(self.DXL_ID, ":setGoalPotision:2:%s" %
self.packetHandler.getRxPacketError(dxl_error))

```

```

#set motor speed for velocity mode
def setSpeed(self,speed,turn):
    if speed >=250:
        speed=250
    if speed <=-250:
        speed=-250
    if turn==0:
        self.speedMotor1=speed
    if turn<0 and turn >-300:
        self.speedMotor1=speed + turn * self.DXL1_left
    if turn>0 and turn <300:
        self.speedMotor1=speed + turn * self.DXL1_rightt
    if turn>=300:
        self.speedMotor1=speed * self.DXL1_leftright
    if turn<=-300:
        self.speedMotor1= -speed * self.DXL1_leftright
    if self.speedMotor1>250:
        self.speedMotor1=250
    if self.speedMotor1<-250:
        self.speedMotor1=-250

def updateSpeed(self):
    dxl_comm_result, dxl_error =
self.packetHandler.write4ByteTxRx(self.portHandler, self.DXL_ID,
self.ADDR_GOAL_VELOCITY, self.speedMotor1 * self.DXL1_orientation)
    if dxl_comm_result != COMM_SUCCESS:
        print("1 %s" % self.packetHandler.getTxRxResult(dxl_comm_result))
    elif dxl_error != 0:
        print("2 %s" % self.packetHandler.getRxPacketError(dxl_error))

def getCurrentSpeed(self):
    self.speedNowMotor1, dxl_comm_result, dxl_error =
self.packetHandler.read4ByteTxRx(self.portHandler, self.DXL_ID,
self.ADDR_PRESENT_VELOCITY)
        # convert unsigned integer values to signed integer for readability
    if self.speedNowMotor1>1024:
        self.speedNowMotor1-=2**32

```

RESCUE DRONE TILT ROTOR PITCH AND ROLL CONTROL

uav_control_v1.py

- Language: Python
- Host: NVIDIA JETSON Orin Nano mission controller / Rescue drone
- Usage:
 - Control arm rotor tilt in pitch and roll
 - Manual control of rotor tilt
- Function:
 - Initiate USB ports to 6 pitch and 6 roll servo motors
 - Collect current position from all servo motors to make it a reference point
 - Offers a user keyboard interface to allow multiple modes of tilt rotor control
 - It can adjust individual arm's pitch and roll and make it a reference point as a calibration
 - It can sweep and rotate a relative or a global angular orientation

```
#!/usr/bin/env python
#UAV rotor pitch and roll tilt control
import os
import sys, tty, termios
from dxl_uav_class import dxl_uav
import math
import time

fd = sys.stdin.fileno()
old_settings = termios.tcgetattr(fd)

def getch():
    try:
        tty.setraw(sys.stdin.fileno())
        ch = sys.stdin.read(1)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    return ch

#initialize dynamixel motors in roll and pitch
DXL_roll=[]
DXL_pitch=[]
for i in range(6):
    DXL_roll.append( dxl_uav(i+1) )
    DXL_pitch.append( dxl_uav(i+11) )

#turn on the dynamixel motors
for i in range(6):
    DXL_roll[i].torqueOn()
    DXL_pitch[i].torqueOn()

mode=0
tiltX=0
tiltY=0
angleStep=1
positionStepX=0
positionStepY=0
scaleRoll=4096/360*40/12
scalePitch=4096/360*306/12
angle2rad=math.pi/180
```

```

#angular position of 6 arms
radArm=[0*angle2rad, 180*angle2rad, 120*angle2rad, 300*angle2rad, 60*angle2rad,
240*angle2rad]
timeDelay=0.1
sweepAngle=30
sweepStep=5

#Read current status and initialize to the present position offset
for i in range(6):
    DXL_roll[i].getCurrentPotision()
    DXL_pitch[i].getCurrentPotision()
    DXL_roll[i].POSITION_OFFSET=DXL_roll[i].PRESENT_POSITION
    DXL_pitch[i].POSITION_OFFSET=DXL_pitch[i].PRESENT_POSITION
    print("Status %d: %d/%d/%d %d/%d/%d" % (i+1, DXL_roll[i].PRESENT_POSITION,
DXL_roll[i].GOAL_POSITION, DXL_roll[i].POSITION_OFFSET, DXL_pitch[i].PRESENT_POSITION,
DXL_pitch[i].GOAL_POSITION, DXL_pitch[i].POSITION_OFFSET) )

print("-----")
print("0: all motors in same individual origin")
print("9: all motors as hexacoptor")
print("q: sweep by individual origin")
print("w: sweep by hexacopter center")
print("-: all motors to 0 position")
print("1-6: roll and pitch motor calibration")
print("m: save calibration")
print("J: left, L: right, I: forward, K: backward")
print("A: faster movement")
print("Z: slower movement")
print("ESC to quit")
print("-----")
while 1:
    charInput=getch()
    if charInput == chr(0x1b):
        break
    if charInput == "0":
        mode=20
    if charInput == "9":
        mode=29
    if charInput == "q":
        mode=27
    if charInput == "w":
        mode=28
    if charInput == "-": #reset position
        for i in range(6):
            DXL_roll[i].setGoalPotision(0)
            DXL_pitch[i].setGoalPotision(0)
        tiltX=0
        tiltY=0
    if charInput == "a":
        angleStep+=1
    if charInput == "z":
        angleStep-=1
    if charInput == "j":
        tiltX-=angleStep
    if charInput == "l":
        tiltX+=angleStep
    if charInput == "i":
        tiltY+=angleStep
    if charInput == "k":
        tiltY-=angleStep
    if charInput == "1":
        mode=1
        tiltX=0

```

```

    tiltY=0
if charInput == "2":
    mode=2
    tiltX=0
    tiltY=0
if charInput == "3":
    mode=3
    tiltX=0
    tiltY=0
if charInput == "4":
    mode=4
    tiltX=0
    tiltY=0
if charInput == "5":
    mode=5
    tiltX=0
    tiltY=0
if charInput == "6":
    mode=6
    tiltX=0
    tiltY=0
if charInput == "m":
    for i in range(6):
        DXL_roll[i].getCurrentPotision()
        DXL_pitch[i].getCurrentPotision()
        DXL_roll[i].POSITION_OFFSET=DXL_roll[i].PRESENT_POSITION
        DXL_pitch[i].POSITION_OFFSET=DXL_pitch[i].PRESENT_POSITION
if mode>0 and mode<7:
    positionStepX=int(tiltX*scaleRoll)
    positionStepY=int(tiltY*scalePitch)
    DXL_roll[mode-1].setGoalPotision(positionStepX)
    DXL_pitch[mode-1].setGoalPotision(positionStepY)
if mode==20:
    positionStepX=int(tiltX*scaleRoll)
    positionStepY=int(tiltY*scalePitch)
    for i in range(6):
        DXL_roll[i].setGoalPotision(positionStepX)
        DXL_pitch[i].setGoalPotision(positionStepY)
        #print("positionStepX: ",positionStepX)
        #print("positionStepY: ",positionStepY)
if mode==29:
    for i in range(6):
        tRoll=int(scaleRoll*(-
tiltX*math.sin(radArm[i])+tiltY*math.cos(radArm[i])))

tPitch=int(scalePitch*(tiltX*math.cos(radArm[i])+tiltY*math.sin(radArm[i])))
    DXL_roll[i].setGoalPotision(tRoll)
    DXL_pitch[i].setGoalPotision(tPitch)

if mode==27:
    tiltX=0
    tiltY=0
    positionStepX=int(tiltX*scaleRoll)
    positionStepY=int(tiltY*scalePitch)
    for i in range(6):
        DXL_roll[i].setGoalPotision(positionStepX)
        DXL_pitch[i].setGoalPotision(positionStepY)
    time.sleep(1)
    for k in range(0,sweepAngle+sweepStep,sweepStep):
        tiltX=k
        positionStepX=int(tiltX*scaleRoll)
        for i in range(6):
            DXL_roll[i].setGoalPotision(positionStepX)

```

```

        DXL_pitch[i].setGoalPotision(positionStepY)
        time.sleep(timeDelay)
    for k in range(0,360+sweepStep,sweepStep):
        tiltX=sweepAngle*math.cos(k*angle2rad)
        tiltY=sweepAngle*math.sin(k*angle2rad)
        positionStepX=int(tiltX*scaleRoll)
        positionStepY=int(tiltY*scalePitch)
        for i in range(6):
            DXL_roll[i].setGoalPotision(positionStepX)
            DXL_pitch[i].setGoalPotision(positionStepY)
            time.sleep(timeDelay)
    for k in range(sweepAngle,-sweepStep,-sweepStep):
        tiltX=k
        positionStepX=int(tiltX*scaleRoll)
        for i in range(6):
            DXL_roll[i].setGoalPotision(positionStepX)
            DXL_pitch[i].setGoalPotision(positionStepY)
            time.sleep(timeDelay)
    mode=0

if mode==28:
    tiltX=0
    tiltY=0
    positionStepX=int(tiltX*scaleRoll)
    positionStepY=int(tiltY*scalePitch)
    for i in range(6):
        DXL_roll[i].setGoalPotision(positionStepX)
        DXL_pitch[i].setGoalPotision(positionStepY)
    time.sleep(1)
    for k in range(0,sweepAngle+sweepStep,sweepStep):
        tiltX=k
        for i in range(6):
            tRoll=int(scaleRoll*(-
                tiltX*math.sin(radArm[i])+tiltY*math.cos(radArm[i])))

    tPitch=int(scalePitch*(tiltX*math.cos(radArm[i])+tiltY*math.sin(radArm[i])))
        DXL_roll[i].setGoalPotision(tRoll)
        DXL_pitch[i].setGoalPotision(tPitch)
    time.sleep(timeDelay)
    for k in range(0,360+sweepStep,sweepStep):
        tiltX=sweepAngle*math.cos(k*angle2rad)
        tiltY=sweepAngle*math.sin(k*angle2rad)
        for i in range(6):
            tRoll=int(scaleRoll*(-
                tiltX*math.sin(radArm[i])+tiltY*math.cos(radArm[i])))

    tPitch=int(scalePitch*(tiltX*math.cos(radArm[i])+tiltY*math.sin(radArm[i])))
        DXL_roll[i].setGoalPotision(tRoll)
        DXL_pitch[i].setGoalPotision(tPitch)
    time.sleep(timeDelay)
    for k in range(sweepAngle,-sweepStep,-sweepStep):
        tiltX=k
        for i in range(6):
            tRoll=int(scaleRoll*(-
                tiltX*math.sin(radArm[i])+tiltY*math.cos(radArm[i])))

    tPitch=int(scalePitch*(tiltX*math.cos(radArm[i])+tiltY*math.sin(radArm[i])))
        DXL_roll[i].setGoalPotision(tRoll)
        DXL_pitch[i].setGoalPotision(tPitch)
    time.sleep(timeDelay)
    mode=0
# report status
for i in range(6):

```

```
print("Status %d: %d %d" % (i+1, tiltX, tiltY) )
#DXL_roll[i].getCurrentPotision()
#DXL_pitch[i].getCurrentPotision()
#print("Status %d: %d/%d/%d %d/%d/%d" % (i+1, DXL_roll[i].PRESENT_POSITION,
DXL_roll[i].GOAL_POSITION, DXL_roll[i].POSITION_OFFSET, DXL_pitch[i].PRESENT_POSITION,
DXL_pitch[i].GOAL_POSITION, DXL_pitch[i].POSITION_OFFSET) )

#turn off and close
for i in range(6):
    DXL_roll[i].torqueOff()
    DXL_pitch[i].torqueOff()
    DXL_roll[i].closePort()
    DXL_pitch[i].closePort()
```

WORKSTATION SCOUT IMAGE DFD and ECC DEMONSTRATION

compare_DFD_CC_v01_GSDSEF_Note.ipynb

- Language: Python in Jupyter Notebook
- Host: Workstation
- Usage:
 - Beach image data analysis example plots
- Function:
 - Load an example beach image data
 - Perform differential frame displacement and error cross correlation analysis on a single tile
 - Find out the best match vector as displacement vector
 - Display DFD and ECC results
 - Show example area 1, 2, and 3 over the ocean image
 - Show a zoomed area image
 - Display gradient density image
 - Compare DFD vs. information-weighted DFD results
 - Show information-weighted DFD with incoming vs. receding flow

```
In [1]:
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import math
from numpy.fft import fft
import glob
#from matplotlib.image import imread
#import os
import time
In [2]:
pathImage='..../scout_video_09/'
filePrefix='gimbal0_'
fileSuffix='.jpg'
idxStart=2000
idxEnd=2500
In [3]:
#np.zeros((2,2), dtype='float', order='C')
In [19]:
idxFile=3
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_0 = mpimg.imread(pathImage + filename) / 256
#imgplot_0 = plt.imshow(img_0)

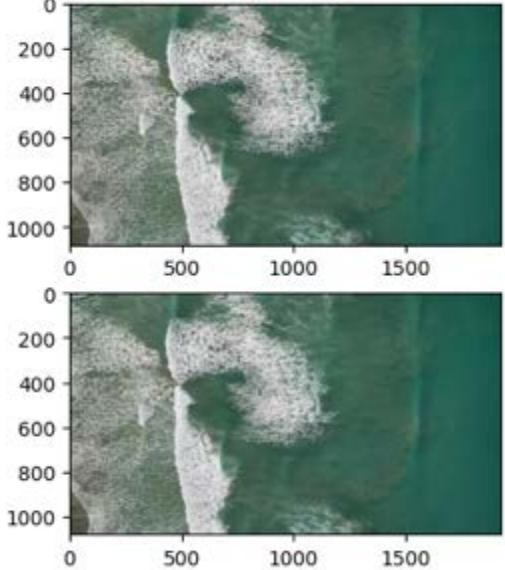
idxFile=4
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_1 = mpimg.imread(pathImage + filename) / 256

fig, (ax_0, ax_1) = plt.subplots(2)
imgplot_0 = ax_0.imshow(img_0)
imgplot_1 = ax_1.imshow(img_1)

plt.show()
```

```

print(np.min(img_0))
print(np.max(img_0))
print(img_0.shape)
print(img_0[1079,1919,2])
print(type(img_0[0,0,0]))


0.0078125
0.99609375
(1080, 1920, 3)
0.28515625
<class 'numpy.float64'>
In [43]:
tileSize=100
#tileX=1530
tileX=450
#tileX=1700
tileY=500
resX=1920
resY=1080
searchRange=50
searchStep=1
DFD=np.zeros((searchRange*2+1, searchRange*2+1))
ref = img_0[tileY:tileY+tileSize, tileX:tileX+tileSize, :]
cont = img_1[tileY:tileY+tileSize, tileX:tileX+tileSize, :]
#fig, (ax_0, ax_1) = plt.subplots(2)
#imgplot_0 = ax_0.imshow(ref)
#imgplot_1 = ax_1.imshow(cont)

f=plt.figure(dpi=100)
f.set_figwidth(10)
f.set_figheight(6)
plt.subplot(1,2,1)
plt.imshow(ref)
plt.axis('off')
plt.subplot(1,2,2)
plt.imshow(cont)
plt.axis('off')
plt.show()

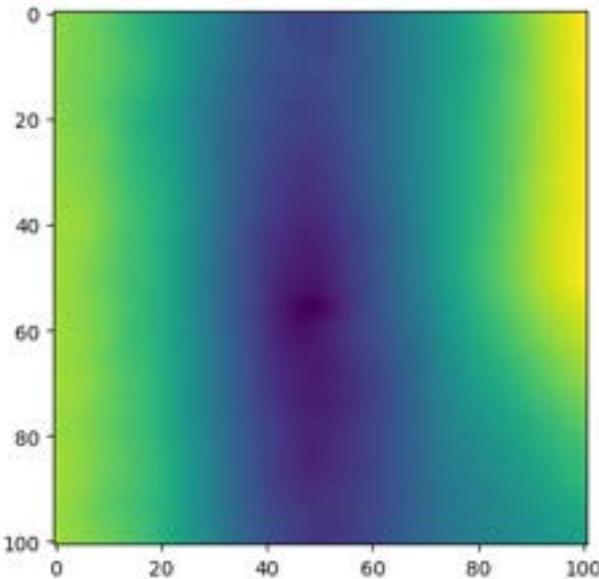
print(np.min(img_0))
print(np.max(img_0))

```

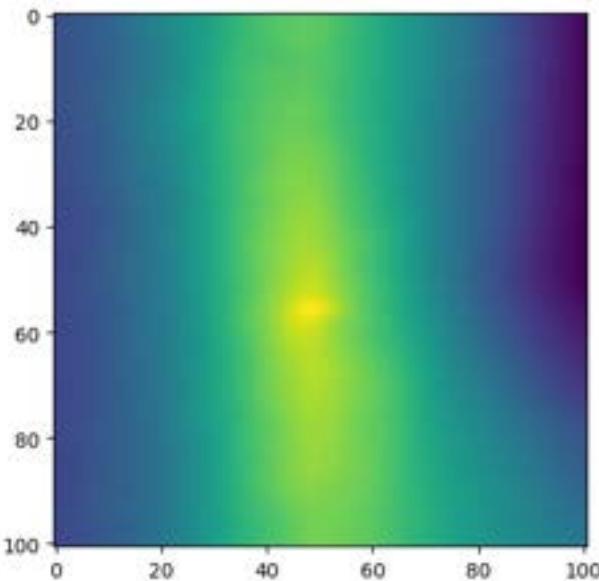
```
print(np.std(ref))
```



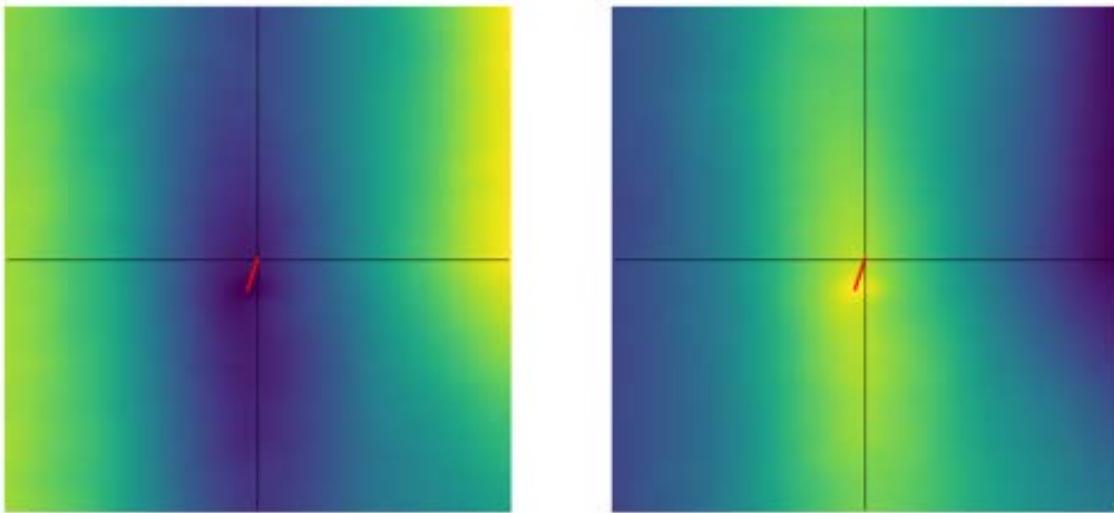
```
0.0078125
0.99609375
0.20333084997292822
In [21]:
#Differential Frame Displacement
for i in range(-searchRange, searchRange+searchStep, searchStep):
    for j in range(-searchRange, searchRange+searchStep, searchStep):
        compare = img_1[tileY+j:tileY+j+tileSize, tileX+i:tileX+i+tileSize, :]
        diff = np.mean((ref-compare)**2)
        #diff = np.mean((ref-compare)**2/(ref+compare)**2)
        DFD[j+searchRange, i+searchRange] = diff
print(compare.shape)
print(j+searchRange, i+searchRange)
#print(DFD[51,51])
imgplot_1 = plt.imshow(DFD)
print(range(-searchRange, searchRange+searchStep, searchStep))
print(DFD.max())
print(DFD.min())
print(np.unravel_index(DFD.argmin(), DFD.shape))
print(np.mean(DFD))
print(np.std(DFD))
(100, 100, 3)
100 100
range(-50, 51)
0.1297401865641276
0.007868874104817708
(56, 48)
0.06925222454556561
0.03010577885725318
```



```
In [22]:
#Error Cross Correlation from paper
ECC=np.zeros((searchRange*2+1, searchRange*2+1))
for i in range(-searchRange, searchRange+searchStep, searchStep):
    for j in range(-searchRange, searchRange+searchStep, searchStep):
        compare = img_1[tileY+j:tileY+j+tileSize, tileX+i:tileX+i+tileSize, :]
        #corr = np.sum(np.multiply(ref, compare))+np.multiply(ref-0.5, compare-0.5)+np.multiply(ref-1, compare-1)
        corr = np.sum(1 - np.sum(np.abs(ref-compare))) / np.sum(ref+compare) )
        ECC[j+searchRange, i+searchRange] = corr
print(compare.shape)
print(j+searchRange,i+searchRange)
#print(ECC[51,51])
imgplot_1 = plt.imshow(ECC)
print(range(-searchRange, searchRange+searchStep, searchStep))
print(ECC.max())
print(ECC.min())
print(np.unravel_index(ECC.argmax(), ECC.shape))
print(np.std(ECC))
print(np.mean(ref+compare))
print(np.mean(np.abs(ref-compare)))
(100, 100, 3)
100 100
range(-50, 51)
0.9560442066115911
0.7314408231104113
(56, 48)
0.0499996805493473
1.2684466145833333
0.2267294270833332
```



```
In [27]:  
f=plt.figure(dpi=150)  
f.set_figwidth(10)  
f.set_figheight(6)  
plt.subplot(1,2,1)  
plt.imshow(DFD)  
(tY,tX)=np.unravel_index(DFD.argmin(), DFD.shape)  
  
plt.plot([0,100],[50,50],'k',linewidth=0.5)  
plt.plot([50,50],[0,100],'k',linewidth=0.5)  
plt.plot([50,tX],[50,tY],'r')  
  
plt.axis('off')  
plt.subplot(1,2,2)  
plt.imshow(ECC)  
(tY,tX)=np.unravel_index(ECC.argmax(), ECC.shape)  
plt.plot([0,100],[50,50],'k',linewidth=0.5)  
plt.plot([50,50],[0,100],'k',linewidth=0.5)  
plt.plot([50,tX],[50,tY],'r')  
  
plt.axis('off')  
plt.show()
```



```
In [44]:  
f=plt.figure(dpi=100)  
f.set_figwidth(10)  
f.set_figheight(6)  
plt.imshow(img_0)  
  
X0=440  
Y0=400  
X1=X0+100  
Y1=Y0+100  
plt.plot([X0,X1],[Y0,Y0],'k')  
plt.plot([X0,X1],[Y1,Y1],'k')  
plt.plot([X0,X0],[Y0,Y1],'k')  
plt.plot([X1,X1],[Y0,Y1],'k')  
  
X0=1500  
Y0=400  
X1=X0+100  
Y1=Y0+100  
plt.plot([X0,X1],[Y0,Y0],'k')  
plt.plot([X0,X1],[Y1,Y1],'k')  
plt.plot([X0,X0],[Y0,Y1],'k')  
plt.plot([X1,X1],[Y0,Y1],'k')  
  
X0=1700  
Y0=800  
X1=X0+100  
Y1=Y0+100  
plt.plot([X0,X1],[Y0,Y0],'k')  
plt.plot([X0,X1],[Y1,Y1],'k')  
plt.plot([X0,X0],[Y0,Y1],'k')  
plt.plot([X1,X1],[Y0,Y1],'k')  
  
plt.axis('off')  
plt.show()
```



```
In [35]:  
tileSpan=15  
dimX=1920  
dimY=1080  
searchRange=15  
searchStep=1  
  
f=plt.figure(dpi=150)  
f.set_figwidth(10)  
f.set_figheight(6)  
  
plt.subplot(1,3,1)  
X0=440  
Y0=400  
X1=X0+100  
Y1=Y0+100  
iX0=X0-searchRange-tileSpan  
iX1=X1+searchRange+tileSpan  
iY0=Y0-searchRange-tileSpan  
iY1=Y1+searchRange+tileSpan  
  
img_ref = img_0[iY0:iY1, iX0:iX1]  
plt.imshow(img_ref)  
plt.axis('off')  
  
plt.subplot(1,3,2)  
X0=1500  
Y0=400  
X1=X0+100  
Y1=Y0+100  
iX0=X0-searchRange-tileSpan  
iX1=X1+searchRange+tileSpan  
iY0=Y0-searchRange-tileSpan  
iY1=Y1+searchRange+tileSpan  
  
img_ref = img_0[iY0:iY1, iX0:iX1]  
plt.imshow(img_ref)  
plt.axis('off')
```

```

plt.subplot(1,3,3)
X0=1700
Y0=800
X1=X0+100
Y1=Y0+100
iX0=X0-searchRange+tileSpan
iX1=X1+searchRange+tileSpan
iY0=Y0-searchRange+tileSpan
iY1=Y1+searchRange+tileSpan

img_ref = img_0[iY0:iY1, iX0:iX1]
plt.imshow(img_ref)
plt.axis('off')
Out[35]:
(-0.5, 159.5, 159.5, -0.5)

```



```

In [45]:
#####
#GSDSEF
#####
start = time.time()

tileSpan=15
dimX=1920
dimY=1080
searchRange=10
searchStep=1

noImages=2
idxStart=2000

idxFile=0
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_ref = mpimg.imread(pathImage + filename) / 256

m=0
print(pathImage + filePrefix +"DFD_" + str(m + idxStart) + ".npy")
#DFD=np.load(pathImage + filePrefix +"DFD_" + str(m + idxStart) + ".npy")
DFDx=np.load(pathImage + filePrefix +"DFDx_" + str(m + idxStart) + ".npy")
DFDy=np.load(pathImage + filePrefix +"DFDy_" + str(m + idxStart) + ".npy")
DFDEC=np.load(pathImage + filePrefix +"DFDEC_" + str(m + idxStart) + ".npy")

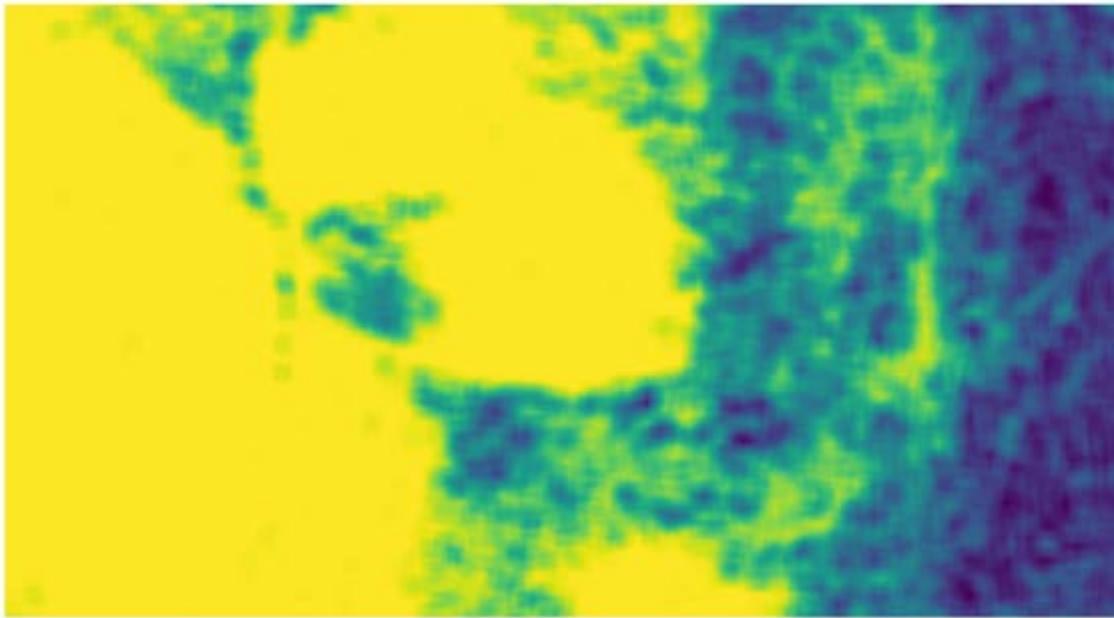
#c = 255 / np.log(1 + np.max(DFDEC))
#log_image = c * (np.log(DFDEC + 1))

f=plt.figure(dpi=100)
f.set_figwidth(10)
f.set_figheight(6)

plt.axis('off')
plt.imshow(DFDEC)

```

```
../scout_video_09/gimbal0_DFD_2000.npy
Out[45]:
<matplotlib.image.AxesImage at 0x7f0295163ee0>
```



```
In [47]:
#plot flow vector
#drone speed compensation
windowSpanX=20
windowSpanY=20
windowShiftX=30
windowShiftY=30
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz
pixelSize=0.1 #m
compensateY= speedY / frameRate / pixelSize
print("Compesantion vector Y: ", compensateY)

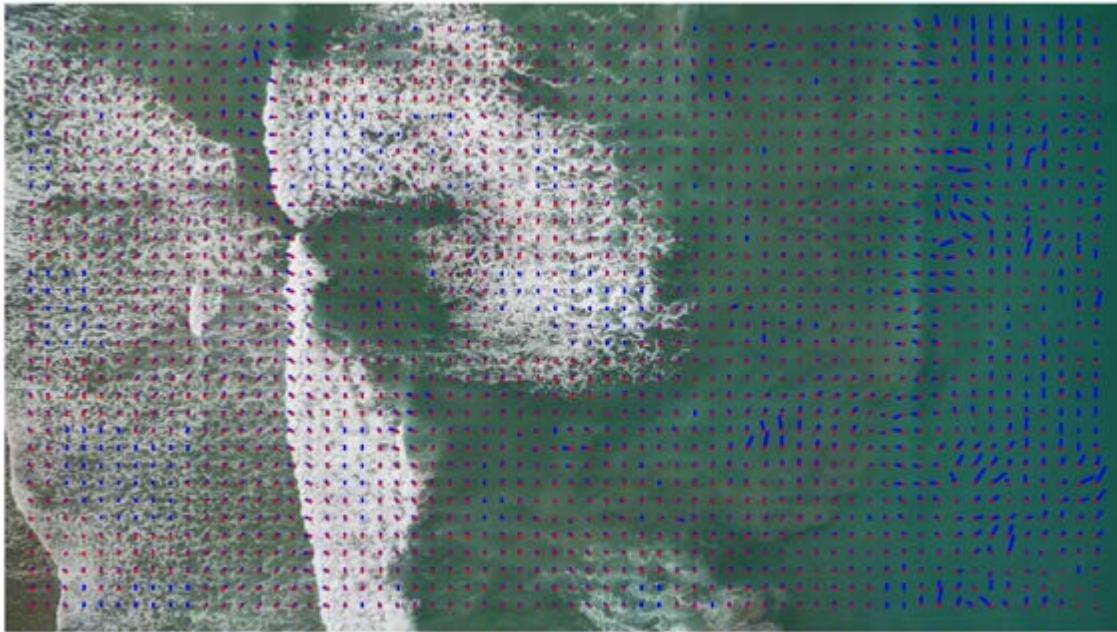
f=plt.figure(dpi=100)
f.set_figwidth(10)
f.set_figheight(6)

print("DFD shape: ", DFDx.shape)
(DFDheight,DFDwidth)=DFDx.shape
imgplot_1 = plt.imshow(img_ref)
for x in range(windowSpanY, DFDwidth-windowSpanX+1, windowShiftX ):
    for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
        vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        #print(vX0, vX0+vX , vY0, vY0+vY)
        if vX>0:
            lineColor='r'
        else:
            lineColor='b'
        plt.plot([vX0, vX0+4*vX], [vY0, vY0+4*(vY-compensateY)], 'b')
vC=np.mean(DFDEc[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
```

```

rE=vC
plt.plot([vX0, vX0+1*vX*rE], [vY0, vY0+1*(vY-compesantY)*rE], 'r')
plt.axis('off')
plt.savefig('DFD1_test.png',bbox_inches='tight', pad_inches = 0)
Compesantion vector Y:  5.0
DFD shape:  (1030, 1870)

```



```

In [48]:
#plot flow vector
#drone speed compensation
windowSpanX=20
windowSpanY=20
windowShiftX=30
windowShiftY=30
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz
pixelSize=0.1 #m
compesantionY= speedY / frameRate / pixelSize
print("Compesantion vector Y: ", compesantionY)

f=plt.figure(dpi=100)
f.set_figwidth(10)
f.set_figheight(6)

print("DFD shape: ", DFDx.shape)
(DFDheight,DFDwidth)=DFDx.shape
imgplot_1 = plt.imshow(img_ref)
for x in range(windowSpanY, DFDwidth-windowSpanX+1, windowShiftX):
    for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
        vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        #print(vX0, vX0+vX , vY0, vY0+vY)
        if vX>0:
            lineColor='r'
        else:

```

```
    lineColor='b'  
    plt.plot([vX0, vX0+4*vX], [vY0, vY0+4*(vY-compensateY)], 'b')  
  
    vC=np.mean(DFDEc[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])  
  
    rE=vC  
    plt.plot([vX0, vX0+1*vX*rE], [vY0, vY0+1*(vY-compensateY)*rE], lineColor)  
plt.axis('off')  
plt.savefig('DFD1_test.png',bbox_inches='tight', pad_inches = 0)  
Compensation vector Y: 5.0  
DFD shape: (1030, 1870)
```



In []:

WORKSTATION SCOUT IMAGE FOCUS AREA ANALYSIS

compare_DFD_CC_v02_area1.ipynb

- Language: Python in Jupyter Notebook
- Host: Workstation
- Usage:
 - Beach image data analysis example plots
- Function:
 - Load an example beach image data
 - Show the focus area to analyze
 - Perform and display DFD and ECC results
 - Show RGB intensity distribution
 - Show a gradient information from the area
 - Compare focus area with the whole image in analytics
 - Compare DFD vs. information-weighted DFD results

```
In [1]:
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import math
from numpy.fft import fft
import glob
#from matplotlib.image import imread
#import os
import time
In [2]:
pathImage='../../scout_video_09/'
filePrefix='gimbal0_'
fileSuffix='.jpg'
idxStart=2050
idxEnd=2500
In [3]:
idxFile=0
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_0 = mpimg.imread(pathImage + filename) / 256
#imgplot_0 = plt.imshow(img_0)

idxFile=1
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_1 = mpimg.imread(pathImage + filename) / 256

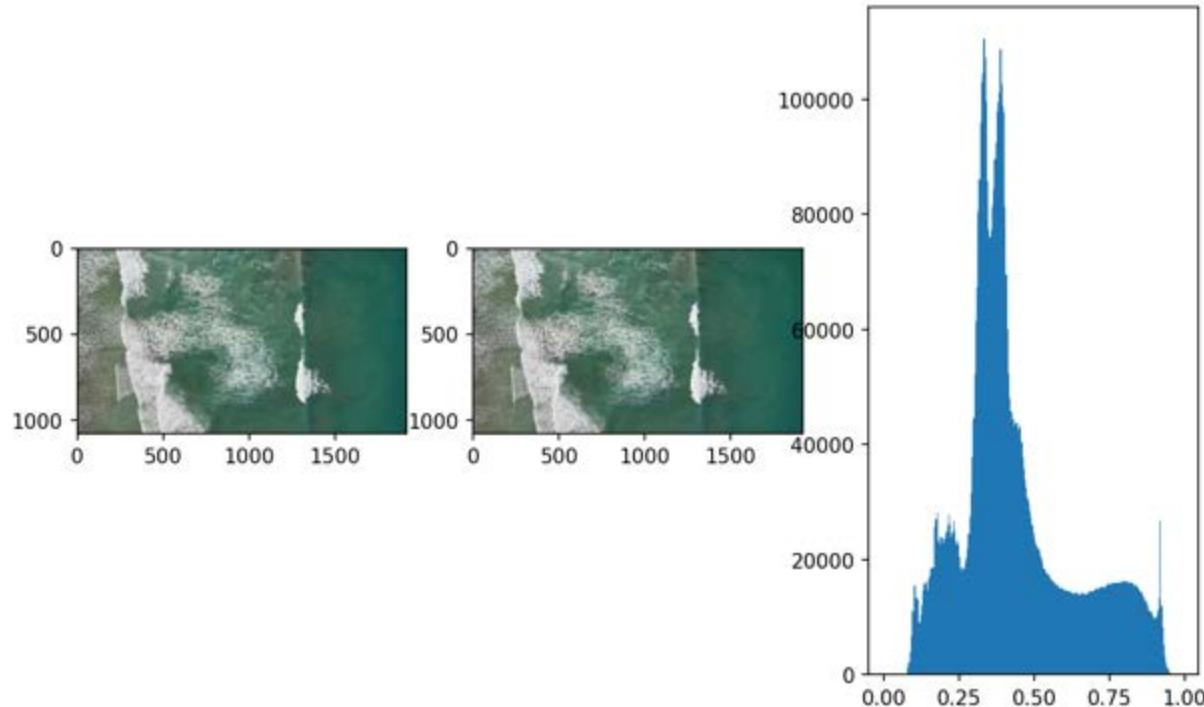
f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(6)
plt.subplot(1,3,1)
plt.imshow(img_0)
plt.subplot(1,3,2)
plt.imshow(img_1)
plt.subplot(1,3,3)
plt.hist(img_0.flatten(),255)
plt.show()

print(np.min(img_0))
```

```

print(np.max(img_0))
print(img_0.shape)
print(img_0[1079,1919,2])
print(type(img_0[0,0,0]))

```



```

0.0
0.99609375
(1080, 1920, 3)
0.296875
<class 'numpy.float64'>
In [4]:
#Section of image
start = time.time()

tileSpan=15
dimX=1920
dimY=1080
searchRange=15
searchStep=1

noImages=2
idxStart=2000

idxFile=0
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_ref = mpimg.imread(pathImage + filename) / 256
print(filename)

idxFile=1
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_compare = mpimg.imread(pathImage + filename) / 256
print(filename)

#image portion to reduce computation loading
X0=440
Y0=400

```

```

#X0=1500
#Y0=400

#X0=1700
#Y0=800

X1=X0+100
Y1=Y0+100

iX0=X0-searchRange+tileSpan
iX1=X1+searchRange+tileSpan
iY0=Y0-searchRange+tileSpan
iY1=Y1+searchRange+tileSpan

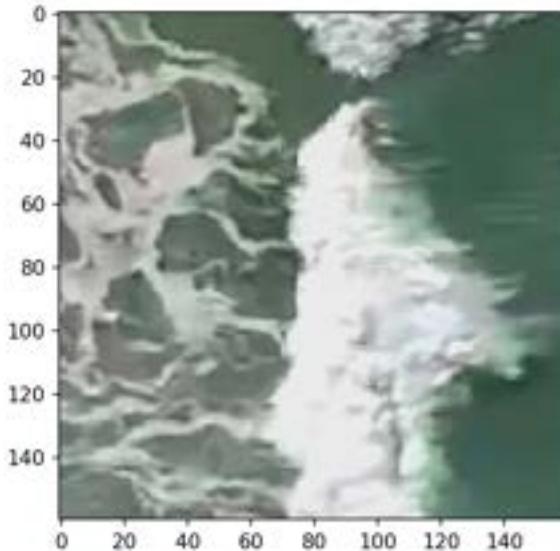
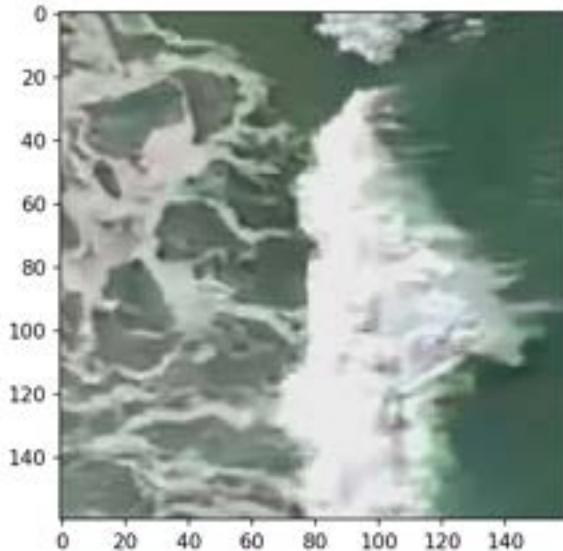
img_ref = img_ref[iY0:iY1, iX0:iX1]
img_compare = img_compare[iY0:iY1, iX0:iX1]
print(img_ref.shape)
(imgX,imgY,t)=img_ref.shape

DFD=np.ones((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
(dimDFDy,dimDFDx)=DFD.shape
print("DFD shape: ", DFD.shape)
DFDx=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
DFDy=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))

f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(6)
plt.subplot(1,2,1)
plt.imshow(img_ref)
plt.subplot(1,2,2)
plt.imshow(img_compare)

plt.show()
gimbalo_2000.jpg
gimbalo_2001.jpg
(160, 160, 3)
DFD shape: (100, 100)

```



```

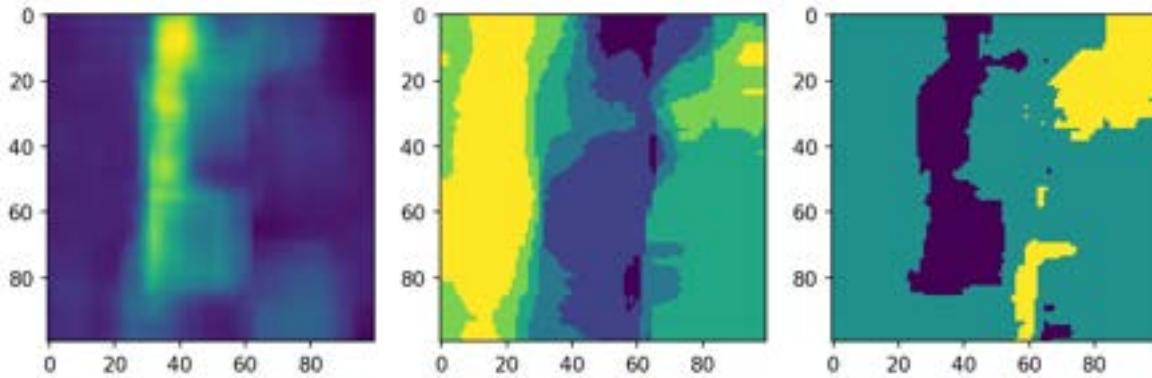
In [5]:
#DFD
start = time.time()
for x in range(searchRange+tileSpan, imgX - searchRange - tileSpan):

```

```

for y in range(searchRange+tileSpan, imgY - searchRange - tileSpan):
    ref = img_ref[y-tileSpan:y+tileSpan+1, x-tileSpan:x+tileSpan+1, :]
    #Differential Frame Displacement
    tDFD=np.zeros((2*searchRange+1, 2*searchRange+1))
    for i in range(-searchRange, searchRange+1, 1):
        for j in range(-searchRange, searchRange+1, 1):
            compare = img_compare[y-tileSpan+j:y+tileSpan+j+1, x-
tileSpan+i:x+tileSpan+i+1, :]
            diff = np.mean((ref-compare)**2)
            #diff = np.mean((ref-compare)**2/(ref+compare)**2)
            tDFD[j+searchRange, i+searchRange] = diff
    (vY, vX)=np.unravel_index(tDFD.argmin(), tDFD.shape)
    tX=x - searchRange - tileSpan
    tY=y - searchRange - tileSpan
    DFD[tY,tX]=tDFD.min()
    DFDx[tY,tX]=vX - tileSpan
    DF Dy[tY,tX]=vY - tileSpan
end = time.time()
print("run time: ", end - start)
run time: 68.50915741920471
In [6]:
f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(6)
plt.subplot(1,3,1)
plt.imshow(DFD)
plt.subplot(1,3,2)
plt.imshow(DFDx)
plt.subplot(1,3,3)
plt.imshow(DFDy)
Out[6]:
<matplotlib.image.AxesImage at 0x7f792647f6d0>

```



```

In [7]:
f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(6)

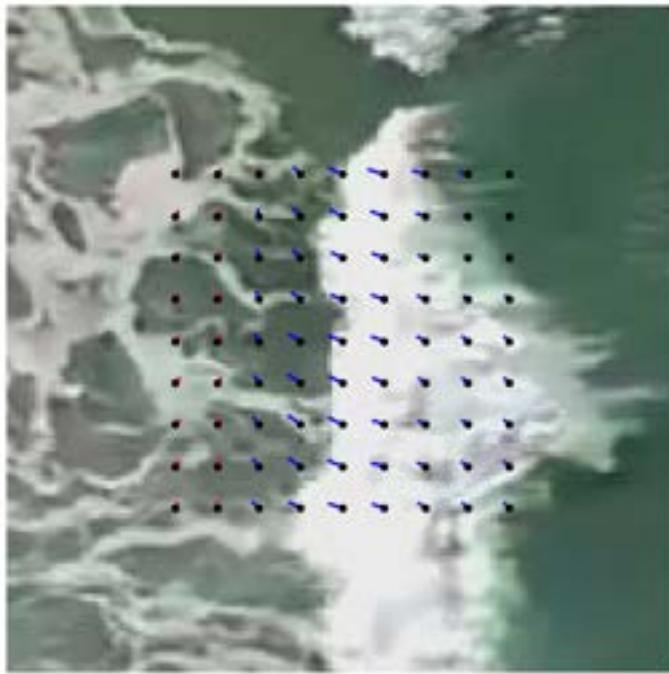
#plot flow vector
#drone speed compensation
windowSpanX=10
windowSpanY=10
windowShiftX=10
windowShiftY=10
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz
pixelSize=0.1 #m
compensateY= speedY / frameRate / pixelSize
print("Compesantion vector Y: ", compensateY)

```

```

print("DFD shape: ", DFD.shape)
(DFDheight,DFDwidth)=DFD.shape
imgplot_1 = plt.imshow(img_ref)
#for x in range(windowSpanX, DFDwidth-windowSpanX+1, 2*windowSpanX+1 ):
#    vX0=x + searchRange + tileSpan
#    plt.plot([vX0,vX0],[searchRange + tileSpan, dimY-searchRange-
#tileSpan],'k',linewidth=0.5)
#for y in range(windowSpanY, DFDheight-windowSpanY+1, 2*windowSpanY+1 ):
#    vY0=y + searchRange + tileSpan
#    plt.plot([searchRange + tileSpan, dimX-searchRange-
#tileSpan],[vY0,vY0],'k',linewidth=0.5)
for x in range(windowSpanY, DFDwidth-windowSpanX+1, windowShiftX ):
    for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
        vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        #print(vX0, vX0+vX , vY0, vY0+vY)
        if vX>0:
            lineColor='r'
        else:
            lineColor='b'
        plt.plot([vX0, vX0+1*vX], [vY0, vY0+1*(vY-compensateY)], lineColor)
        plt.plot([vX0],[vY0], '.k')
plt.axis('off')
plt.savefig('DFD1_test.png',bbox_inches='tight', pad_inches = 0)
Compesantion vector Y:  5.0
DFD shape:  (100, 100)

```



In [8]:

```

#Error Cross Correlation from paper
ECC=np.ones((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
(dimDFDy,dimDFDx)=ECC.shape
ECCx=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
ECCy=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))

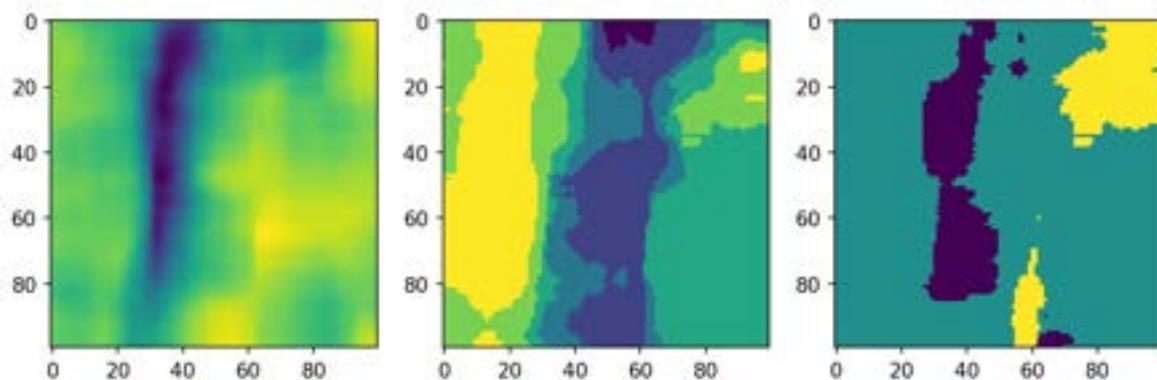
start = time.time()

```

```

for x in range(searchRange+tileSpan, imgX - searchRange - tileSpan):
    for y in range(searchRange+tileSpan, imgY - searchRange - tileSpan):
        ref = img_ref[y-tileSpan:y+tileSpan+1, x-tileSpan:x+tileSpan+1, :]
        #ECC
        tECC=np.zeros((2*searchRange+1, 2*searchRange+1))
        for i in range(-searchRange, searchRange+1, 1):
            for j in range(-searchRange, searchRange+1, 1):
                compare = img_compare[y-tileSpan+j:y+tileSpan+j+1, x-
tileSpan+i:x+tileSpan+i+1, :]
                corr = np.sum(1 - np.sum(np.abs(ref-compare)) / np.sum(ref+compare) )
                tECC[j+searchRange, i+searchRange] = corr
        (vY, vX)=np.unravel_index(tECC.argmax(), tECC.shape)
        tX=x - searchRange - tileSpan
        tY=y - searchRange - tileSpan
        ECC[tY,tX]=tECC.max()
        ECCx[tY,tX]=vX - tileSpan
        ECCy[tY,tX]=vY - tileSpan
end = time.time()
print("run time: ", end - start)
run time: 127.29741024971008
In [9]:
f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(6)
plt.subplot(1,3,1)
plt.imshow(ECC)
plt.subplot(1,3,2)
plt.imshow(ECCx)
plt.subplot(1,3,3)
plt.imshow(ECCy)
Out[9]:
<matplotlib.image.AxesImage at 0x7f797ad75ea0>

```



```

In [68]:
#####
#GSDSEF
#####
f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(6)

#plot flow vector
#drone speed compensation
windowSpanX=10
windowSpanY=10
windowShiftX=10
windowShiftY=10
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz

```

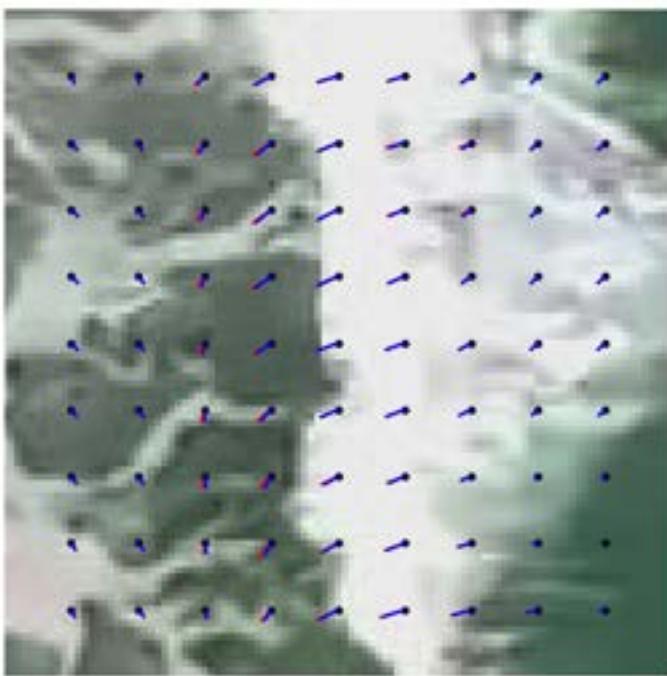
```

pixelSize=0.1 #m
compensateY= speedy / frameRate / pixelSize
print("Compesantion vector Y: ", compensateY)

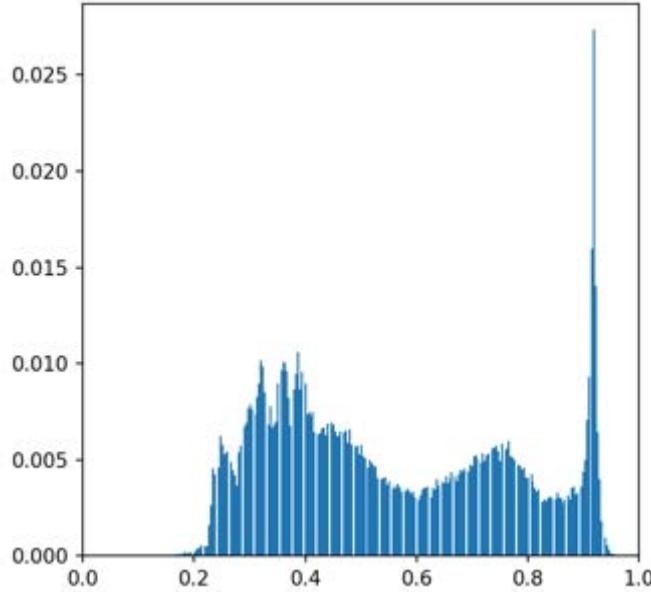
print("DFD shape: ", DFD.shape)
(DFDheight,DFDwidth)=DFD.shape
imgplot_1 = plt.imshow(img_ref)
(imgY,imgX,imgZ)=img_ref.shape
for x in range(windowSpanY, DFDwidth-windowSpanX+1, windowShiftX):
    for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
        #reference
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        plt.plot([vX0],[vY0], '.k')
        #DFD
        vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        if vX>0:
            lineColor='m'
        else:
            lineColor='c'
        plt.plot([vX0, vX0+1*vX], [vY0, vY0+1*(vY-compensateY)], 'r')
        #ECC
        vX=np.mean(ECCx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(ECCy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        if vX>0:
            lineColor='r'
        else:
            lineColor='b'
        plt.plot([vX0, vX0+1*vX], [vY0, vY0+1*(vY-compensateY)], 'b')

plt.axis('off')
plt.xlim(searchRange+tileSpan, imgX-searchRange-tileSpan)
plt.ylim(searchRange+tileSpan, imgY-searchRange-tileSpan)
plt.savefig('DFD1_test.png',bbox_inches='tight', pad_inches = 0)
Compesantion vector Y:  5.0
DFD shape:  (100, 100)

```

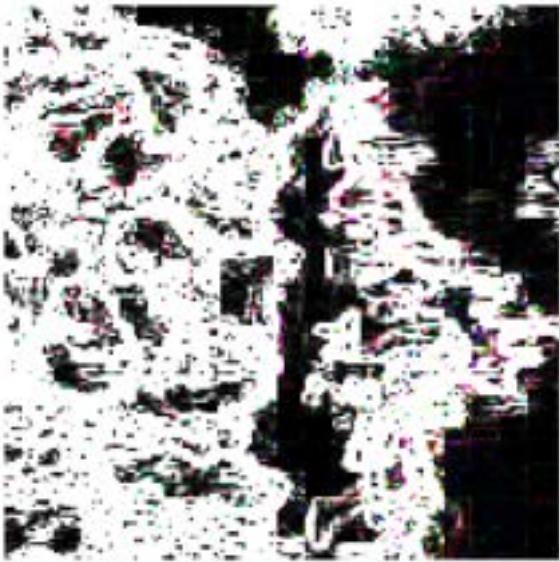


```
In [76]:  
#####  
#GSDSEF  
#####  
f=plt.figure(dpi=150)  
f.set_figwidth(5)  
f.set_figheight(5)  
weights = np.ones_like(img_ref.flatten()) / len(img_ref.flatten())  
  
plt.hist(img_ref.flatten(), 256, weights=weights)  
plt.xlim(0,1)  
Out[76]:  
(0.0, 1.0)
```



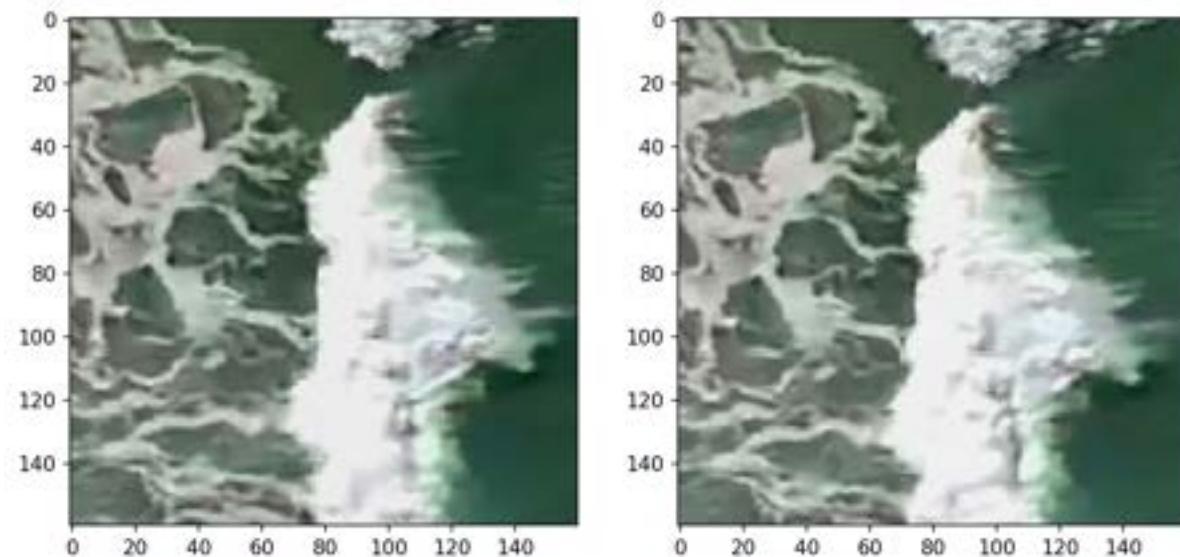
```
In [79]:  
#####  
#GSDSEF  
#####  
#information in section  
print("Image section-----")  
print("np.std(img_ref):", np.std(img_ref))  
(sY,sX,sZ)=img_ref.shape  
#gradient for information  
grad_ref_0a=img_ref[1:sY,0:sX-1,:]-img_ref[0:sY-1,0:sX-1,:]  
grad_ref_0b=img_ref[0:sY-1,1:sX,:]-img_ref[0:sY-1,0:sX-1,:]  
grad_ref_0c=img_ref[1:sY,1:sX,:]-img_ref[0:sY-1,0:sX-1,:]  
grad_ref_0=grad_ref_0a**2+grad_ref_0b**2+grad_ref_0c**2/np.sqrt(2)  
print("np.std(grad_ref_0): ", np.std(grad_ref_0))  
print("np.mean(grad_ref_0): ", np.mean(grad_ref_0))  
print("np.log10(np.std(grad_ref_0)): ", np.log10(np.std(grad_ref_0)))  
print("np.max(grad_ref_0): ", np.max(grad_ref_0))  
print("np.log10(np.max(grad_ref_0)): ", np.log10(np.max(grad_ref_0)))  
c = 255 / np.log(1 + np.max(grad_ref_0))  
log_image = c * (np.log(grad_ref_0 + 1))  
  
f=plt.figure(dpi=150)  
f.set_figwidth(5)  
f.set_figheight(5)  
  
plt.axis('off')  
plt.imshow(log_image)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
Image section-----
np.std(img_ref): 0.21441134901325956
np.std(grad_ref_0): 0.021988232740330647
np.mean(grad_ref_0): 0.0096106021110758
np.log10(np.std(grad_ref_0)): -1.6578096747758735
np.max(grad_ref_0): 0.5355482200282661
np.log10(np.max(grad_ref_0)): -0.2712014197912586
Out[79]:
<matplotlib.image.AxesImage at 0x7f791e5a2650>
```

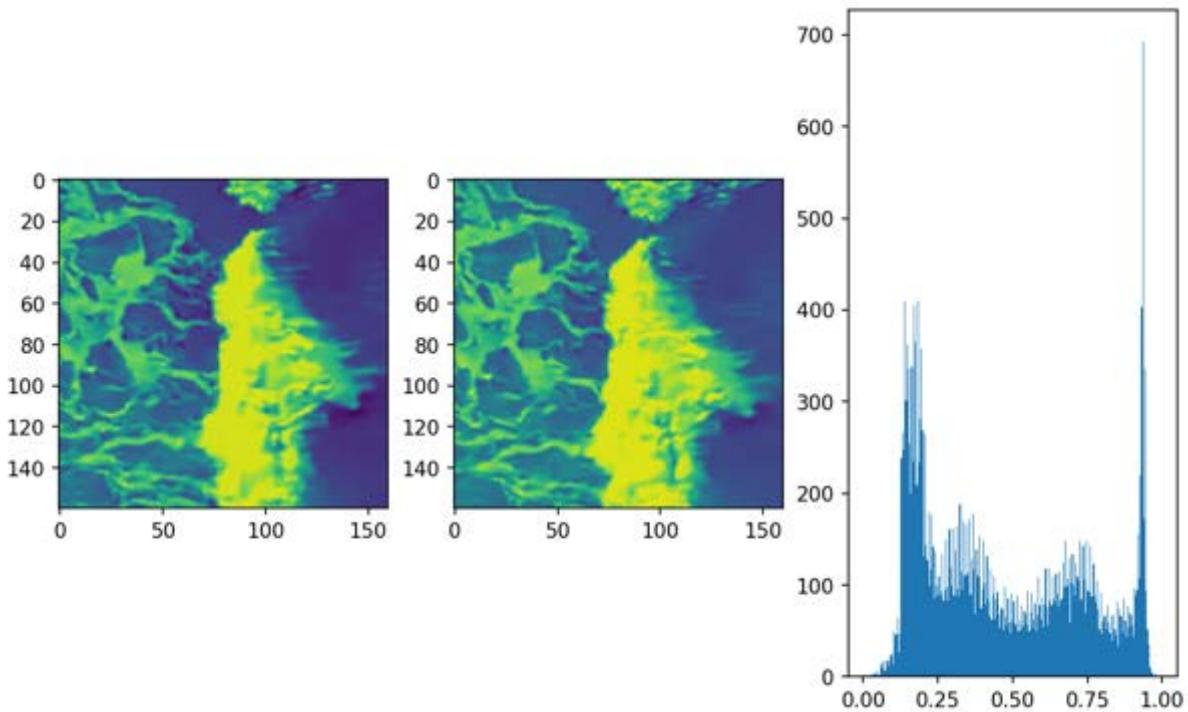


```
In [11]:
print(np.std(img_0))
print(np.std(ref))
print(np.min(ref))
print(np.max(ref))
ref_scale=(img_ref-np.min(img_ref))/(np.max(img_ref)-np.min(img_ref))
compare_scale=(img_compare-np.min(img_compare))/(np.max(img_compare)-
np.min(img_compare))
print(np.min(ref_scale))
print(np.max(ref_scale))
f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(6)

plt.subplot(1,2,1)
plt.imshow(ref_scale)
plt.subplot(1,2,2)
plt.imshow(compare_scale)
0.19232732813282308
0.16886650801558967
0.1640625
0.953125
0.0
1.0
Out[11]:
<matplotlib.image.AxesImage at 0x7f79260085b0>
```



```
In [12]:  
ref_gray=img_ref[:,:,:]+img_ref[:,:,:]+img_ref[:,:,:]  
ref_gray=(ref_gray-np.min(ref_gray))/(np.max(ref_gray)-np.min(ref_gray))  
compare_gray=img_compare[:,:,:]+img_compare[:,:,:]+img_compare[:,:,:]  
compare_gray=(compare_gray-np.min(compare_gray))/(np.max(compare_gray)-  
np.min(compare_gray))  
  
print(np.min(ref_gray))  
print(np.max(ref_gray))  
  
f=plt.figure(dpi=150)  
f.set_figwidth(10)  
f.set_figheight(6)  
  
plt.subplot(1,3,1)  
plt.imshow(ref_gray)  
plt.subplot(1,3,2)  
plt.imshow(compare_gray)  
plt.subplot(1,3,3)  
plt.hist(ref_gray.flatten(),255)  
plt.show()  
0.0  
1.0
```



```
In [62]:
#information in the overall image
print("Full image-----")
print("np.std(img_0): ", np.std(img_0))
#gradient for information
grad_img_0a=img_0[1:dimY, 0:dimX-1, :] - img_0[0:dimY-1, 0:dimX-1, :]
grad_img_0b=img_0[0:dimY-1, 1:dimX, :] - img_0[0:dimY-1, 0:dimX-1, :]
grad_img_0c=img_0[1:dimY, 1:dimX, :] - img_0[0:dimY-1, 0:dimX-1, :]
grad_img_0=grad_img_0a**2 + grad_img_0b**2 + grad_img_0c**2/np.sqrt(2)
grad_img_scale=grad_img_0
print("np.std(grad_img_0): ", np.std(grad_img_0))
print("np.mean(grad_img_0): ", np.mean(grad_img_0))
print("np.log10(np.std(grad_img_0)): ", np.log10(np.std(grad_img_0)))
print("np.max(grad_img_0): ", np.max(grad_img_0))
print("np.log10(np.max(grad_img_0)): ", np.log10(np.max(grad_img_0)))

#information in section
print("Image section-----")
print("np.std(img_ref):", np.std(img_ref))
(sY,sX,sZ)=img_ref.shape
#gradient for information
grad_ref_0a=img_ref[1:sY,0:sX-1,:]-img_ref[0:sY-1,0:sX-1,:]
grad_ref_0b=img_ref[0:sY-1,1:sX,:]-img_ref[0:sY-1,0:sX-1,:]
grad_ref_0c=img_ref[1:sY,1:sX,:]-img_ref[0:sY-1,0:sX-1,:]
grad_ref_0=grad_ref_0a**2 + grad_ref_0b**2 + grad_ref_0c**2/np.sqrt(2)
print("np.std(grad_ref_0): ", np.std(grad_ref_0))
print("np.mean(grad_ref_0): ", np.mean(grad_ref_0))
print("np.log10(np.std(grad_ref_0)): ", np.log10(np.std(grad_ref_0)))
print("np.max(grad_ref_0): ", np.max(grad_ref_0))
print("np.log10(np.max(grad_ref_0)): ", np.log10(np.max(grad_ref_0)))

#information in tile
print("Image tile-----")
print("np.std(ref): ", np.std(ref))
(sY,sX,sZ)=ref.shape
#gradient for information
```

```

grad_tile_0a=ref[1:sY,    0:sX-1, :] - ref[0:sY-1, 0:sX-1, :]
grad_tile_0b=ref[0:sY-1, 1:sX,    :] - ref[0:sY-1, 0:sX-1, :]
grad_tile_0c=ref[1:sY,    1:sX,    :] - ref[0:sY-1, 0:sX-1, :]
grad_tile_0=grad_tile_0a**2 + grad_tile_0b**2 + grad_tile_0c**2/np.sqrt(2)
print("np.std(grad_tile_0): ", np.std(grad_tile_0))
print("np.mean(grad_tile_0): ", np.mean(grad_tile_0))
print("np.log10(np.std(grad_tile_0)): ", np.log10(np.std(grad_tile_0)))
print("np.max(grad_tile_0): ", np.max(grad_tile_0))
print("np.log10(np.max(grad_tile_0)): ", np.log10(np.max(grad_tile_0)))

grad_tile_count = (grad_tile_0 > 0).sum()
(tY,tX,tZ)=grad_tile_0.shape
print("grad_tile_count, area ", grad_tile_count)
print("grad_tile area: ", tY, "*", tX, "*", tZ, "=", (tY*tX*tZ))
print("grad_tile_count/area: ", grad_tile_count/(tY*tX*tZ))

print(grad_img_0.shape)
print(np.max(grad_img_0))
print(np.min(grad_img_0))
#print(grad_img_0a)

f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(10)

plt.subplot(3,2,1)
plt.hist(grad_img_0.flatten(),255)
plt.yscale('log')
plt.subplot(3,2,2)
plt.hist(grad_ref_0.flatten(),255)
plt.yscale('log')

plt.subplot(3,2,3)
c = 255 / np.log(1 + np.max(grad_img_0))
log_image = c * (np.log(grad_img_0 + 1))
plt.imshow(log_image)
plt.subplot(3,2,4)
c = 255 / np.log(1 + np.max(grad_ref_0))
log_image = c * (np.log(grad_ref_0 + 1))
plt.imshow(log_image)

plt.subplot(3,2,5)
plt.hist(img_0.flatten(),255)
plt.yscale('log')
plt.subplot(3,2,6)
plt.hist(img_ref.flatten(),255)
plt.yscale('log')

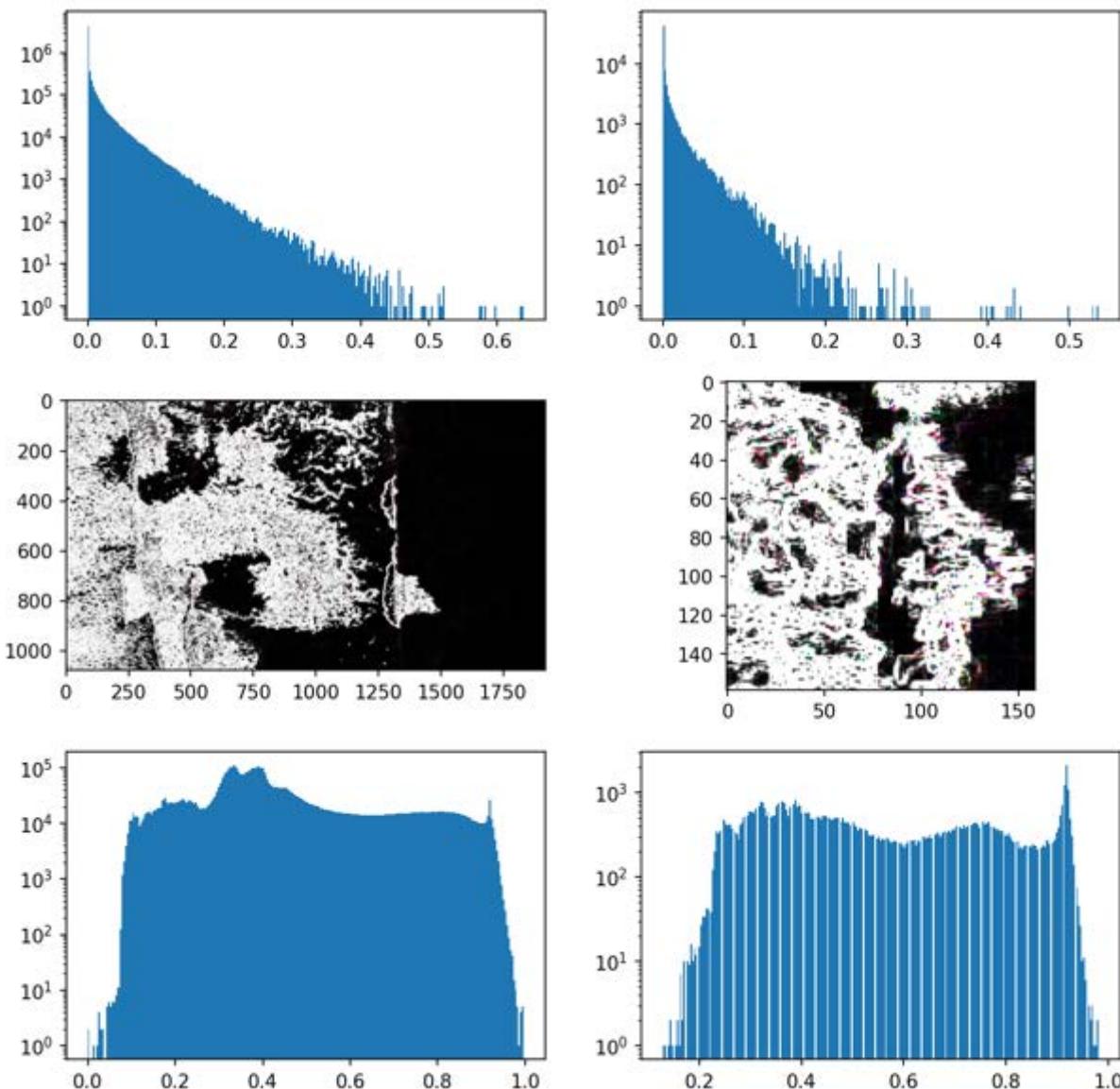
plt.show()
#print(grad_img_0[0:10,0:10,0])
Full image-----
np.std(img_0): 0.19232732813282308
np.std(grad_img_0): 0.01970766417000527
np.mean(grad_img_0): 0.007089085910298204
np.log10(np.std(grad_img_0)): -1.7053648469632052
np.max(grad_img_0): 0.6383041775008746
np.log10(np.max(grad_img_0)): -0.19497231323956482
Image section-----
np.std(img_ref): 0.21441134901325956
np.std(grad_ref_0): 0.021988232740330647
np.mean(grad_ref_0): 0.0096106021110758
np.log10(np.std(grad_ref_0)): -1.6578096747758735
np.max(grad_ref_0): 0.5355482200282661

```

```

np.log10(np.max(grad_ref_0)): -0.2712014197912586
Image tile-----
np.std(ref): 0.16886650801558967
np.std(grad_tile_0): 0.011171332947182724
np.mean(grad_tile_0): 0.004375367637702558
np.log10(np.std(grad_tile_0)): -1.9518950044073542
np.max(grad_tile_0): 0.14464968496178748
np.log10(np.max(grad_tile_0)): -0.8396825078819249
grad_tile_count, area 2493
grad_tile area: 30 * 30 * 3 = 2700
grad_tile_count/area: 0.9233333333333333
(1079, 1919, 3)
0.6383041775008746
0.0
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).

```



```

In [14]:
DFDS=np.ones((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))

```

```

#(dimDFDy,dimDFDx)=DFD.shape
#print("DFD shape: ", DFD.shape)
DFDSx=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
DFDSy=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
DFDSr=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))

#STD scaled vector
image_std=np.std(img_0)
#
start = time.time()
for x in range(searchRange+tileSpan, imgX - searchRange - tileSpan):
    for y in range(searchRange+tileSpan, imgY - searchRange - tileSpan):
        ref = img_ref[y-tileSpan:y+tileSpan+1, x-tileSpan:x+tileSpan+1, :]

        #Differential Frame Displacement with STD scaled vector
        tDFDS=np.zeros((2*searchRange+1, 2*searchRange+1))
        search_std = np.zeros((2*searchRange+1, 2*searchRange+1))
        #
        for i in range(-searchRange, searchRange+1, 1):
            for j in range(-searchRange, searchRange+1, 1):
                compare = img_compare[y-tileSpan+j:y+tileSpan+j+1, x-
tileSpan+i:x+tileSpan+i+1, :]
                diff = np.mean((ref-compare)**2)
                #diff = np.mean((ref-compare)**2/(ref+compare)**2)
                tDFDS[j+searchRange, i+searchRange] = diff
                search_std[j+searchRange, i+searchRange] = np.std(compare)
        (vY, vX)=np.unravel_index(tDFDS.argmax(), tDFDS.shape)
        tX = x - searchRange - tileSpan
        tY = y - searchRange - tileSpan
        DFDS[tY,tX]=tDFDS.min()
        DFDSx[tY,tX]=(vX - tileSpan) # * search_std[vY,vX] / image_std
        DFDSy[tY,tX]=(vY - tileSpan) # * search_std[vY,vX] / image_std
        DFDSr[tY,tX]= search_std[vY,vX] / image_std
end = time.time()
print("run time: ", end - start)
run time: 193.14682459831238
In [15]:
print(search_std[5,5]/image_std)
print(np.mean(search_std))
1.1343491884419719
0.16142135530110002
In [16]:
f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(6)

#plot flow vector
#drone speed compensation
windowSpanX=10
windowSpanY=10
windowShiftX=10
windowShiftY=10
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz
pixelSize=0.1 #m
compensateY= speedY / frameRate / pixelSize
print("Compesantion vector Y: ", compensateY)

print("DFD shape: ", DFD.shape)
(DFDheight,DFDwidth)=DFD.shape
imgplot_1 = plt.imshow(img_ref)
for x in range(windowSpanY, DFDwidth-windowSpanX+1, windowShiftX ):

```

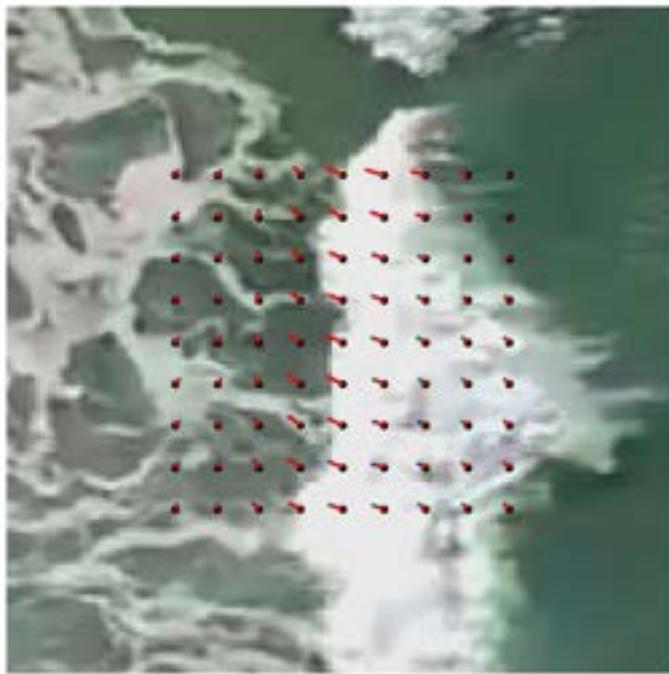
```

for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
    #reference
    vX0=x + searchRange + tileSpan
    vY0=y + searchRange + tileSpan
    plt.plot([vX0],[vY0], '.k')
    #DFD
    vX=np.mean(DFDx[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
    vY=np.mean(DFDy[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
    #if vX>0:
    #    lineColor='m'
    #else:
    #    lineColor='c'
    plt.plot([vX0, vX0+1*vX], [vY0, vY0+1*(vY-compensateY)], 'b')
    #ECC
    vX=np.mean(ECCx[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
    vY=np.mean(ECCy[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
    #if vX>0:
    #    lineColor='r'
    #else:
    #    lineColor='b'
    plt.plot([vX0, vX0+1*vX], [vY0, vY0+1*(vY-compensateY)], 'g')

    #DFDS
    vX=np.mean(DFDSx[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
    vY=np.mean(DFDSy[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
    vR=np.mean(DFDSr[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
    #if vX>0:
    #    lineColor='m'
    #else:
    #    lineColor='c'
    plt.plot([vX0, vX0+1*vX*vR], [vY0, vY0+1*(vY-compensateY)*vR], 'r')

plt.axis('off')
#plt.savefig('DFD1_test.png',bbox_inches='tight', pad_inches = 0)
Compesantion vector Y: 5.0
DFD shape: (100, 100)
Out[16]:
(-0.5, 159.5, 159.5, -0.5)

```



```
In [17]:
print(np.mean(DFDSx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])) 
print(np.mean(DFDSy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])) 
print(vX0,vY0)
print(compensateY)
-1.0
4.0
120 120
5.0
In [63]:
#Full image info
grad_img_0a=img_0[1:dimY, 0:dimX-1, :] - img_0[0:dimY-1, 0:dimX-1, :]
grad_img_0b=img_0[0:dimY-1, 1:dimX, :] - img_0[0:dimY-1, 0:dimX-1, :]
grad_img_0c=img_0[1:dimY, 1:dimX, :] - img_0[0:dimY-1, 0:dimX-1, :]
grad_img_0=grad_img_0a**2 + grad_img_0b**2 + grad_img_0c**2/np.sqrt(2)
grad_img_scale=grad_img_0
print("np.std(grad_img_0): ", np.std(grad_img_0))
print("np.mean(grad_img_0): ", np.mean(grad_img_0))
print("np.log10(np.std(grad_img_0)): ", np.log10(np.std(grad_img_0)))
image_info_max=np.max(grad_img_0)
image_info_max_log=np.log10(np.max(grad_img_0))
image_info_mean=np.mean(grad_img_0)
image_info_mean_log=np.log10(np.mean(grad_img_0))
image_info_std=np.std(grad_img_0)
image_info_std_log=np.log10(np.std(grad_img_0))
#
DFDE=np.ones((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
#(dimDFDy,dimDFDx)=DFD.shape
#print("DFD shape: ", DFD.shape)
DFDEX=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
DFDEy=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
DFDER=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
DFDEM=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
DFDEa=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))
DFDEC=np.zeros((imgY-2*searchRange-2*tileSpan, imgX-2*searchRange-2*tileSpan))

#STD scaled vector
```

```

image_std=np.std(img_0)
#
start = time.time()
#min_info=1
for x in range(searchRange+tileSpan, imgX - searchRange - tileSpan):
    for y in range(searchRange+tileSpan, imgY - searchRange - tileSpan):
        ref = img_ref[y-tileSpan:y+tileSpan+1, x-tileSpan:x+tileSpan+1, :]

        #Differential Frame Displacement with STD scaled vector
        tDFDE=np.zeros((2*searchRange+1, 2*searchRange+1))
        search_info_std = np.zeros((2*searchRange+1, 2*searchRange+1))
        search_info_mean = np.zeros((2*searchRange+1, 2*searchRange+1))
        search_info_max = np.zeros((2*searchRange+1, 2*searchRange+1))
        search_info_count = np.zeros((2*searchRange+1, 2*searchRange+1))
        #
        for i in range(-searchRange, searchRange+1, 1):
            for j in range(-searchRange, searchRange+1, 1):
                compare = img_compare[y-tileSpan+j:y+tileSpan+j+1, x-
tileSpan+i:x+tileSpan+i+1, :]
                diff = np.mean((ref-compare)**2)
                #diff = np.mean((ref-compare)**2/(ref+compare)**2)
                tDFDE[j+searchRange, i+searchRange] = diff
                #tile information from gradient
                (cY, cX, cZ)=compare.shape
                comp_grad_a = compare[1:cY, 0:cX-1, :] - compare[0:cY-1, 0:cX-1, :]
                comp_grad_b = compare[0:cY-1, 1:cX, :] - compare[0:cY-1, 0:cX-1, :]
                comp_grad_c = compare[1:cY, 1:cX, :] - compare[0:cY-1, 0:cX-1, :]
                comp_grad=comp_grad_a**2 + comp_grad_b**2 + comp_grad_c**2/np.sqrt(2)
                search_info_std[j+searchRange, i+searchRange] = np.std(comp_grad)
                search_info_mean[j+searchRange, i+searchRange] = np.mean(comp_grad)
                search_info_max[j+searchRange, i+searchRange] = np.max(comp_grad)
                search_info_count[j+searchRange, i+searchRange] = (comp_grad >
0).sum()
                #if np.std(comp_grad) < min_info and np.std(comp_grad) > 0:
                #    min_info=np.std(comp_grad)
                #print(np.log10(np.std(comp_grad)))
                #
                (vY, vX)=np.unravel_index(tDFDE.argmax(), tDFDE.shape)
                tX = x - searchRange - tileSpan
                tY = y - searchRange - tileSpan
                DFDE[tY,tX] = tDFDE.min()
                DFDEx[tY,tX] = (vX - tileSpan) # * search_std[vY,vX] / image_std
                DFDEy[tY,tX] = (vY - tileSpan) # * search_std[vY,vX] / image_std
                DFDER[tY,tX] = search_info_std[vY,vX]
                DFDEM[tY,tX] = search_info_mean[vY,vX]
                DFDEa[tY,tX] = search_info_max[vY,vX]
                DFDEC[tY,tX] = search_info_count[vY,vX] / (cY*cX*cZ)
end = time.time()
print("run time: ", end - start)
np.std(grad_img_0): 0.01970766417000527
np.mean(grad_img_0): 0.007089085910298204
np.log10(np.std(grad_img_0)): -1.7053648469632052
run time: 458.3000919818878
In [64]:
#image_info=np.std(grad_img_0)

f=plt.figure(dpi=150)
f.set_figwidth(10)
f.set_figheight(6)

#plot flow vector
#drone speed compensation
windowSpanX=10

```

```

windowSpanY=10
windowShiftX=10
windowShiftY=10
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz
pixelSize=0.1 #m
compensateY= speedY / frameRate / pixelSize
print("Compensation vector Y: ", compensateY)

print("DFD shape: ", DFD.shape)
(DFDheight,DFDwidth)=DFD.shape
imgplot_1 = plt.imshow(img_ref)
for x in range(windowSpanY, DFDwidth-windowSpanX+1, windowShiftX):
    for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
        #reference
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        plt.plot([vX0],[vY0], '.k')
        #DFD
        vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        #if vX>0:
        #    lineColor='m'
        #else:
        #    lineColor='c'
        plt.plot([vX0, vX0+1*vX], [vY0, vY0+1*(vY-compensateY)], 'b')
        #ECC
        vX=np.mean(ECCx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(ECCy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        #if vX>0:
        #    lineColor='r'
        #else:
        #    lineColor='b'
        plt.plot([vX0, vX0+1*vX], [vY0, vY0+1*(vY-compensateY)], 'g')

        #DFDS
        vX=np.mean(DFDSx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDSy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vR=np.mean(DFDSr[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        #if vX>0:
        #    lineColor='m'
        #else:
        #    lineColor='c'
        plt.plot([vX0, vX0+1*vX*vR], [vY0, vY0+1*(vY-compensateY)*vR], 'r')

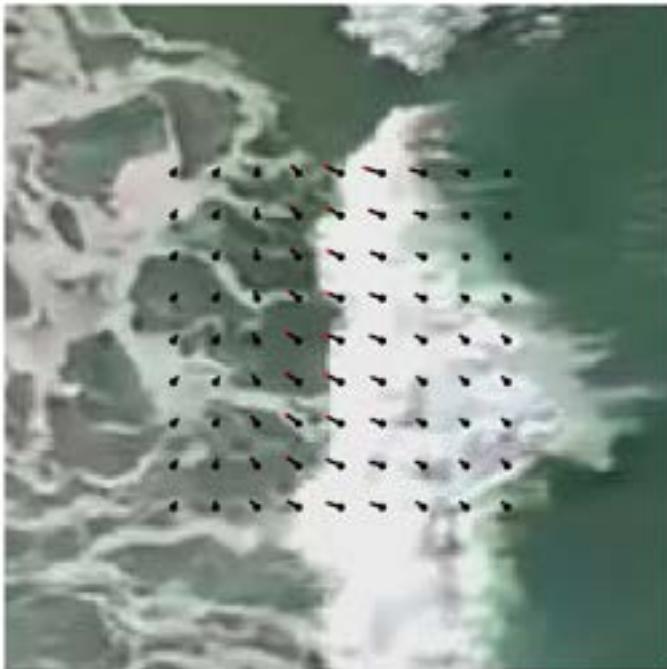
        #DFDE
        vX=np.mean(DFDEx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDEy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vR=np.mean(DFDER[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vM=np.mean(DFDEM[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vA=np.max(DFDEa[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vC=np.mean(DFDEC[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        #if vX>0:
        #    lineColor='m'
        #else:
        #    lineColor='c'
        rE=1/(1+image_info_max_log-np.log10(vA))
        rE=vC
        plt.plot([vX0, vX0+1*vX*rE], [vY0, vY0+1*(vY-compensateY)*rE], 'k')
        #print(rE)
        #print(vR)

```

```

plt.axis('off')
#plt.savefig('DFD1_test.png',bbox_inches='tight', pad_inches = 0)
Compesantion vector Y:  5.0
DFD shape:  (100, 100)
Out[64]:
(-0.5, 159.5, 159.5, -0.5)

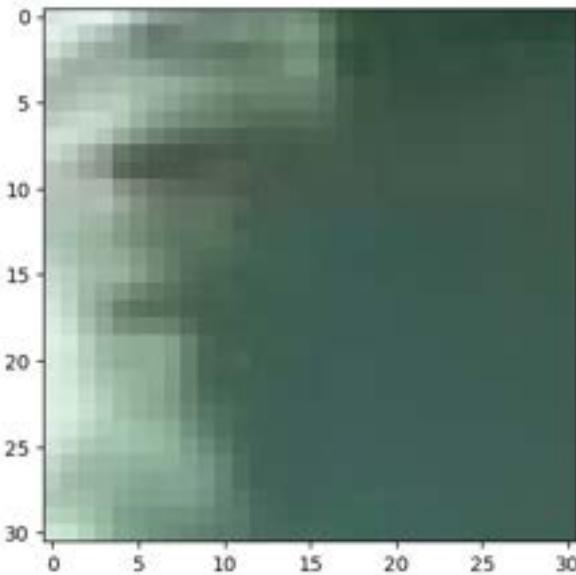
```



```

In [50]:
print(vR)
print(vR/image_info)
print(image_info_mean_log)
print(np.max(grad_img_0))
print(image_info_max_log)
print(np.log10(vA))
print(np.log10(vM))
0.0184027766747634
0.9337878155429559
-2.149409760544234
0.6383041775008746
-0.19497231323956482
-0.7981437358124197
-2.0070213044645113
In [55]:
print(image_info_mean_log)
print((image_info_mean_log-np.log10(vM)))
print(image_info_mean_log + image_info_std_log)
print(np.log10(vM) + np.log10(vA))
-2.149409760544234
-0.14238845607972284
-3.8547746075074394
-2.801751798798599
In [56]:
print(comp_grad.shape)
(30, 30, 3)
In [59]:
imgplot_1 = plt.imshow(ref)

```



```
In [80]:
#####
#GSDSEF
#####

#image_info=np.std(grad_img_0)

f=plt.figure(dpi=150)
f.set_figwidth(5)
f.set_figheight(5)

#plot flow vector
#drone speed compensation
windowSpanX=10
windowSpanY=10
windowShiftX=10
windowShiftY=10
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz
pixelSize=0.1 #m
compensateY= speedY / frameRate / pixelSize
print("Compesantion vector Y: ", compensateY)

print("DFD shape: ", DFD.shape)
(DFDheight,DFDwidth)=DFD.shape
imgplot_1 = plt.imshow(img_ref)
for x in range(windowSpanY, DFDwidth-windowSpanX+1, windowShiftX):
    for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
        #reference
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        plt.plot([vX0],[vY0], '.k')
        #DFD
        vX=np.mean(DFDx[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
        #if vX>0:
        #    lineColor='m'
        #else:
        #    lineColor='c'
        plt.plot([vX0, vX0+1*vX], [vY0, vY0+1*(vY-compensateY)], 'b')
```

```

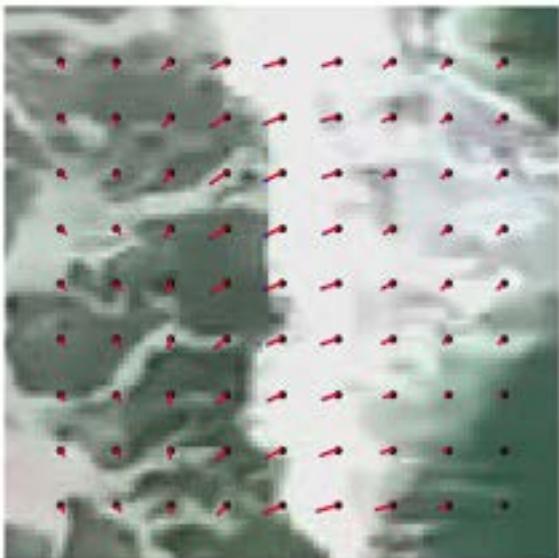
#ECC
 vX=np.mean(ECCx[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 vY=np.mean(ECCy[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 if vX>0:
 #    lineColor='r'
 else:
 #    lineColor='b'
 plt.plot([vX0, vX0+1*vX], [vY0, vY0+1*(vY-compensateY)], 'g')

#DFDS
 vX=np.mean(DFDSx[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 vY=np.mean(DFDSy[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 vR=np.mean(DFDSr[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 if vX>0:
 #    lineColor='m'
 else:
 #    lineColor='c'
 plt.plot([vX0, vX0+1*vX*vR], [vY0, vY0+1*(vY-compensateY)*vR], 'r')

#DFDE
 vX=np.mean(DFDEx[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 vY=np.mean(DFDEy[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 vR=np.mean(DFDER[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 vM=np.mean(DFDEM[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 vA=np.max(DFDEa[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 vC=np.mean(DFDEC[y>windowSpanY:y+windowSpanY,x>windowSpanX:x+windowSpanX])
 if vX>0:
 #    lineColor='m'
 else:
 #    lineColor='c'
 rE=1/(1+image_info_max_log-np.log10(vA))
 rE=vC
 plt.plot([vX0, vX0+1*vX*rE], [vY0, vY0+1*(vY-compensateY)*rE], 'r')
#print(rE)
#print(vR)

plt.axis('off')
#plt.savefig('DFD1_test.png',bbox_inches='tight', pad_inches = 0)
plt.axis('off')
plt.xlim(searchRange+tileSpan, imgX-searchRange-tileSpan)
plt.ylim(searchRange+tileSpan, imgY-searchRange-tileSpan)
Compesantion vector Y: 5.0
DFD shape: (100, 100)
Out[80]:
(30.0, 130.0)

```



In []:

WORKSTATION SCOUT IMAGE DIFFERENTIAL DISPLACEMENT ANALYSIS FOR OCEAN WATER OPTICAL FLOW

processing_DFD1_scout09_save_v03.ipynb

- Language: Python in Jupyter Notebook
- Host: Workstation
- Usage:
 - Beach image data analysis and generate machine learning training data
- Function:
 - Load an example beach image data
 - Sample a wave front tile to analyze
 - Perform differential frame displacement analysis on a single tile
 - Find out the best match vector as displacement vector
 - Perform whole image differential frame displacement analysis
 - Plot DFD value and vector elements
 - Plot DFD vectors found through the analysis
 - Apply scout drone relative movement compensation
 - Plot resulting DFD vectors as ocean water flow
 - Perform a series of sequential image analysis through the whole series of images
 - Save DFD data

```
In [1]:
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import math
from numpy.fft import fft
import glob
#from matplotlib.image import imread
#import os
import time
In [21]:
pathImage='../../scout_video_09/'
filePrefix='gimbal0_'
fileSuffix='.jpg'
idxStart=2000
idxEnd=2500
In [3]:
#np.zeros((2,2), dtype='float', order='C')
In [4]:
idxFile=3
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_0 = mpimg.imread(pathImage + filename) / 256
#imgplot_0 = plt.imshow(img_0)

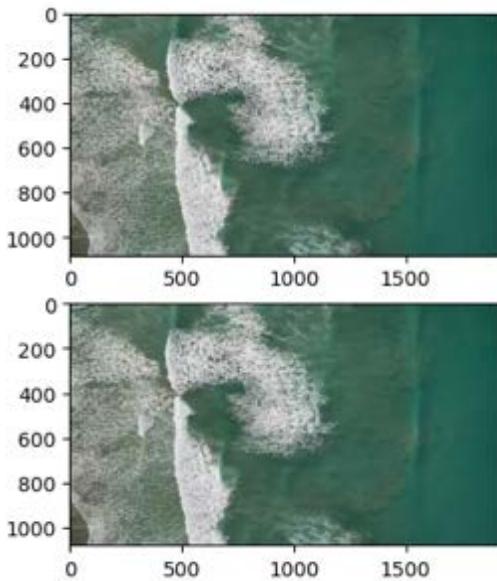
idxFile=4
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_1 = mpimg.imread(pathImage + filename) / 256

fig, (ax_0, ax_1) = plt.subplots(2)
imgplot_0 = ax_0.imshow(img_0)
```

```
imgplot_1 = ax_1.imshow(img_1)
```

```
plt.show()
```

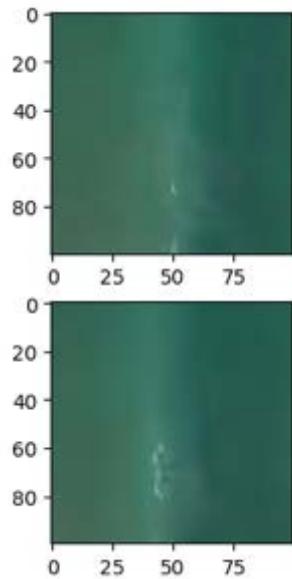
```
print(np.min(img_0))
print(np.max(img_0))
print(img_0.shape)
print(img_0[1079,1919,2])
print(type(img_0[0,0,0]))
```



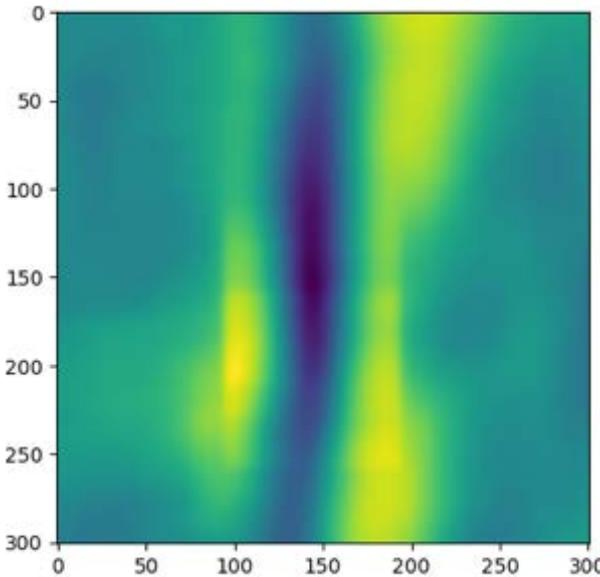
```
0.0078125
0.99609375
(1080, 1920, 3)
0.28515625
<class 'numpy.float64'>
In [5]:
tileSize=100
tileX=1500
tileY=500
resX=1920
resY=1080
searchRange=150
searchStep=1
DFD=np.zeros((searchRange*2+1, searchRange*2+1))
ref = img_0[tileY:tileY+tileSize, tileX:tileX+tileSize, :]
cont = img_1[tileY:tileY+tileSize, tileX:tileX+tileSize, :]
fig, (ax_0, ax_1) = plt.subplots(2)
imgplot_0 = ax_0.imshow(ref)
imgplot_1 = ax_1.imshow(cont)
```

```
plt.show()
```

```
print(np.min(img_0))
print(np.max(img_0))
```



```
0.0078125
0.99609375
In [6]:
for i in range(-searchRange, searchRange+searchStep, searchStep):
    for j in range(-searchRange, searchRange+searchStep, searchStep):
        compare = img_1[tileY+j:tileY+j+tileSize, tileX+i:tileX+i+tileSize, :]
        diff = np.mean((ref-compare)**2)
        DFD[j+searchRange, i+searchRange] = diff
print(compare.shape)
print(j+searchRange,i+searchRange)
print(DFD[51,51])
imgplot_1 = plt.imshow(DFD)
print(range(-searchRange, searchRange+searchStep, searchStep))
print(DFD.max())
print(DFD.min())
(100, 100, 3)
300 300
0.002429237365722656
range(-150, 151)
0.004801509094238281
0.00022349751790364582
```



```
In [7]:
print(np.unravel_index(DFD.argmax(), DFD.shape))
print(np.min(DFD))
(156, 144)
0.00022349751790364582
In [8]:
start = time.time()

tileSpan=15
dimX=1920
dimY=1080
searchRange=10
searchStep=1

noImages=2
idxStart=2000

idxFile=0
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_ref = mpimg.imread(pathImage + filename) / 256
DFD=np.ones((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))
(dimDFDy,dimDFDx)=DFD.shape
print("DFD shape: ", DFD.shape)
DFDx=np.zeros((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))
DFDy=np.zeros((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))

idxFile=1
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_compare = mpimg.imread(pathImage + filename) / 256

for i in range(-searchRange, searchRange+searchStep, searchStep):
    print("step i: ", i)
    for j in range(-searchRange, searchRange+searchStep, searchStep):
        refX1 = searchRange
        refX2 = dimX - searchRange
        refY1 = searchRange
        refY2 = dimY - searchRange
        #print("ref: ", refX1, refX2, refY1, refY2)
        ref = img_ref[refY1:refY2, refX1:refX2, :]
        #print("ref shape: ", ref.shape)
        diffX1 = searchRange + i
```

```

diffX2 = dimX - searchRange + i
diffY1 = searchRange + j
diffY2 = dimY - searchRange + j
#print("diff: ", diffX1, diffX2, diffY1, diffY2)
compare = img_compare[diffY1:diffY2, diffX1:diffX2, :]
diff3=(ref-compare)**2
diff=(diff3[:, :, 0]+diff3[:, :, 1]+diff3[:, :, 2]) / 3
#print("diff shape: ", compare.shape)
tSum=np.zeros((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))
#print("sum shape: ", sum.shape)
dimDiffY=diff.shape[0]
dimDiffX=diff.shape[1]
for x in range(-tileSpan, tileSpan + 1):
    for y in range(-tileSpan, tileSpan + 1):
        #print(tileSpan+y, dimDiffY-tileSpan+y)
        #print(tileSpan+x, dimDiffX-tileSpan+x)
        tSum += diff[tileSpan+y:dimDiffY-tileSpan+y, tileSpan+x:dimDiffX-
tileSpan+x]
tSum = tSum / (tileSpan + 1)**2
#print("sum complete")
for x in range(dimDFDx):
    for y in range(dimDFDy):
        if tSum[y, x]<DFD[y, x]:
            DFD[y, x]=tSum[y, x]
            DFDx[y, x]=i
            DFDy[y, x]=j

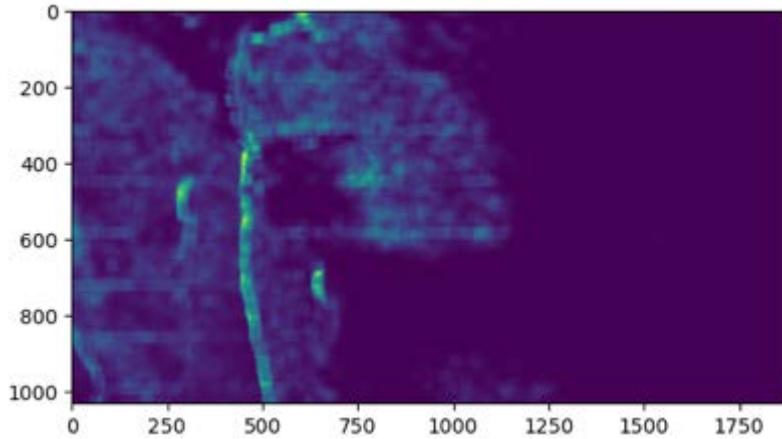
end = time.time()
print(end - start)
DFD shape: (1030, 1870)
step i: -10
step i: -9
step i: -8
step i: -7
step i: -6
step i: -5
step i: -4
step i: -3
step i: -2
step i: -1
step i: 0
step i: 1
step i: 2
step i: 3
step i: 4
step i: 5
step i: 6
step i: 7
step i: 8
step i: 9
step i: 10
2304.8531091213226
In [9]:
print(DFDy.min(), DFDy.max(), DFDx.min(), DFDx.max())
np.save(pathImage + filePrefix +"DFD" + str(idxFile + idxStart), DFD)
np.save(pathImage + filePrefix +"DFDx_" + str(idxFile + idxStart), DFDx)
np.save(pathImage + filePrefix +"DFDy_" + str(idxFile + idxStart), DFDy)
-10.0 10.0 -10.0 10.0
In [10]:
print(DFD.shape)
print(np.unravel_index(DFD.argmax(), DFD.shape))
print(np.min(DFD))
imgplot_1 = plt.imshow(DFD)

```

```
(1030, 1870)
```

```
(0, 1630)
```

```
0.0
```



```
In [11]:
```

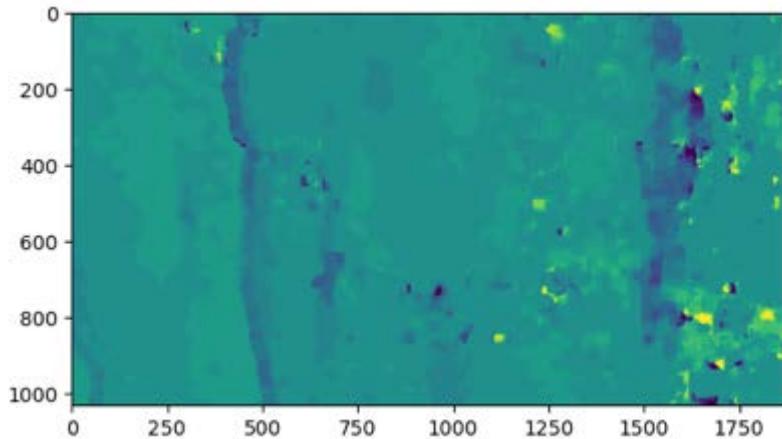
```
imgplot_1 = plt.imshow(DFDx)
```

```
np.max(DFDx)
```

```
np.mean(DFDx)
```

```
Out[11]:
```

```
0.02680961528477234
```



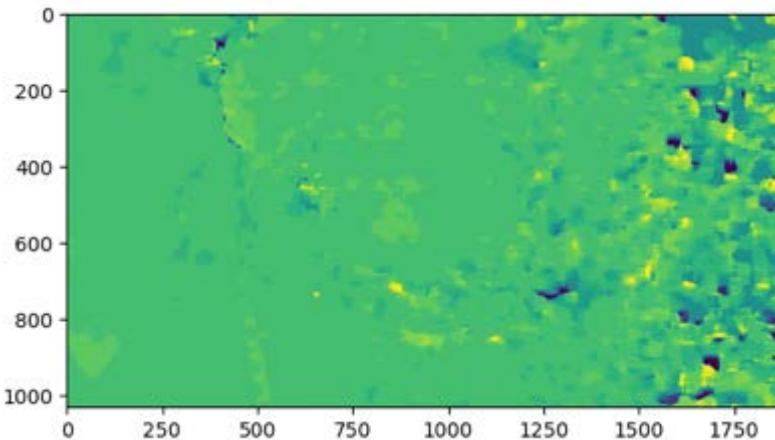
```
In [12]:
```

```
imgplot_1 = plt.imshow(DFDy)
```

```
np.mean(DFDy)
```

```
Out[12]:
```

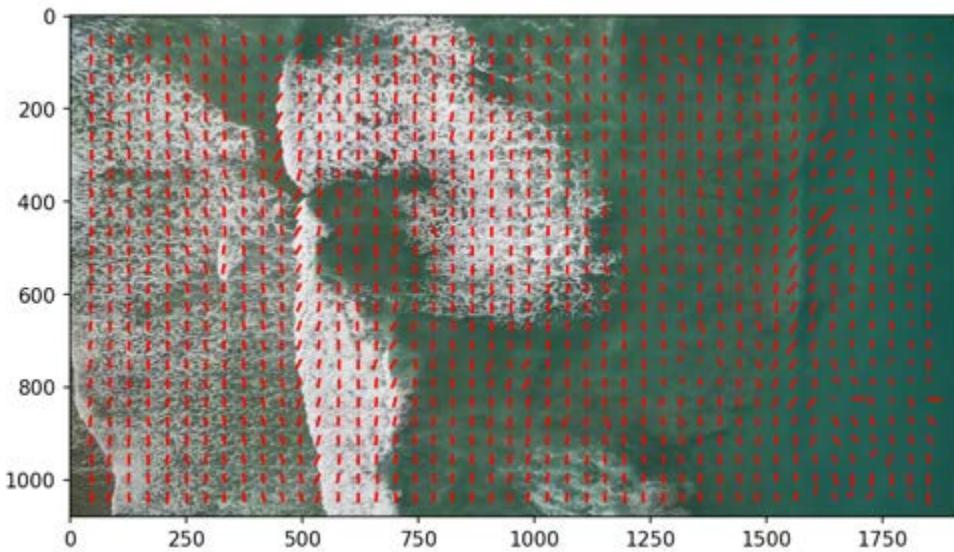
```
3.8405700638596127
```



```
In [13]:
#plot flow vector
windowSpanX=20
windowSpanY=20

f=plt.figure(dpi=150)
f.set_figwidth(8)
f.set_figheight(12)

print("DFD shape: ", DFD.shape)
(DFDheight,DFDwidth)=DFD.shape
#DFD=np.ones((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))
#DFDx=np.zeros((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))
#DFDy=np.zeros((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))
imgplot_1 = plt.imshow(img_ref)
for x in range(windowSpanX, DFDwidth-windowSpanX, 2*windowSpanX+1 ):
    for y in range(windowSpanY, DFDheight-windowSpanY, 2*windowSpanY+1 ):
        vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        #print(vX0, vX0+vX , vY0, vY0+vY)
        plt.plot([vX0, vX0+4*vX], [vY0, vY0+4*vY], 'r')
DFD shape: (1030, 1870)
```



```
In [20]:
```

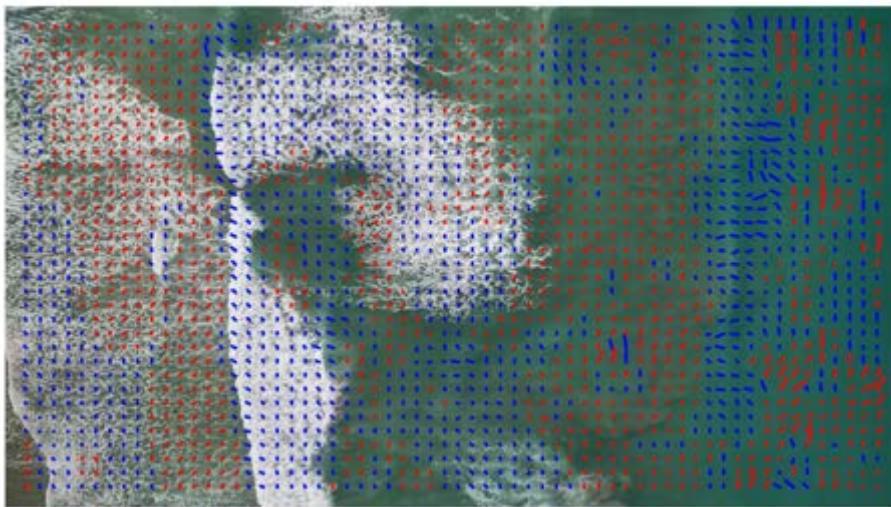
```

#plot flow vector
#drone speed compensation
windowSpanX=20
windowSpanY=20
windowShiftX=30
windowShiftY=30
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz
pixelSize=0.1 #m
compensateY= speedY / frameRate / pixelSize
print("Compesantion vector Y: ", compensateY)

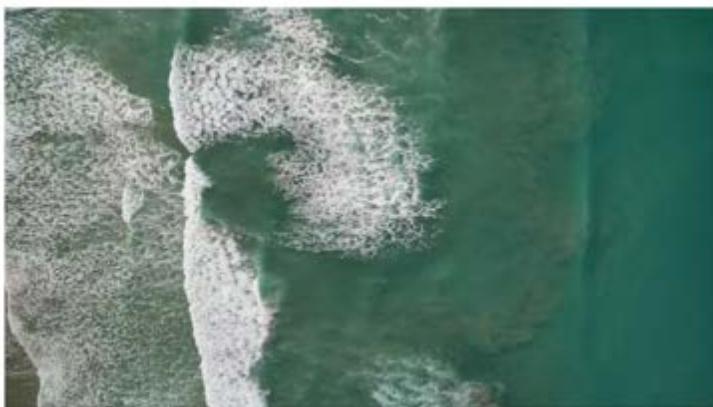
f=plt.figure(dpi=200)
f.set_figwidth(8)
f.set_figheight(12)

print("DFD shape: ", DFD.shape)
(DFDheight,DFDwidth)=DFD.shape
imgplot_1 = plt.imshow(img_ref)
#for x in range(windowSpanX, DFDwidth-windowSpanX+1, 2*windowSpanX+1 ):
#    vX0=x + searchRange + tileSpan
#    plt.plot([vX0,vX0], [searchRange + tileSpan, dimY-searchRange-
tileSpan], 'k', linewidth=0.5)
#for y in range(windowSpanY, DFDheight-windowSpanY+1, 2*windowSpanY+1 ):
#    vY0=y + searchRange + tileSpan
#    plt.plot([searchRange + tileSpan, dimX-searchRange-
tileSpan], [vY0,vY0], 'k', linewidth=0.5)
for x in range(windowSpanY, DFDwidth-windowSpanX+1, windowShiftX ):
    for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
        vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        #print(vX0, vX0+vX , vY0, vY0+vY)
        if vX>0:
            lineColor='r'
        else:
            lineColor='b'
        plt.plot([vX0, vX0+4*vX], [vY0, vY0+4*(vY-compensateY)], lineColor)
plt.axis('off')
plt.savefig('DFD1_test.png',bbox_inches='tight', pad_inches = 0)
Compesantion vector Y: 5.0
DFD shape: (1030, 1870)

```



```
In [18]:
img_ref = mpimg.imread(pathImage + filename) / 256
imgplot_1 = plt.imshow(img_ref)
plt.axis('off')
plt.savefig('DFD1_test.png',bbox_inches='tight', pad_inches = 0)
```



```
In [ ]:
#perform DFD analysis over images and save them
start = time.time()
imagePrefix='DFD1_gimbal0_'
imageSuffix='.png'

tileSpan=15
dimX=1920
dimY=1080
searchRange=10
searchStep=1

idxStart=0

for m in range(2029,3150):
    idxFile=m
    filename=filePrefix + str(idxFile + idxStart) + fileSuffix
    img_ref = mpimg.imread(pathImage + filename) / 256
    DFD=np.ones((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))
    (dimDFDy,dimDFDx)=DFD.shape
    print("File: ", filename)
    DFDx=np.zeros((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))
    DF Dy=np.zeros((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-2*tileSpan))
```

```

idxFile=m+1
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_compare = mpimg.imread(pathImage + filename) / 256

for i in range(-searchRange, searchRange+searchStep, searchStep):
    #print("step i: ", i)
    for j in range(-searchRange, searchRange+searchStep, searchStep):
        refX1 = searchRange
        refX2 = dimX - searchRange
        refY1 = searchRange
        refY2 = dimY - searchRange
        #print("ref: ", refX1, refX2, refY1, refY2)
        ref = img_ref[refY1:refY2, refX1:refX2, :]
        #print("ref shape: ", ref.shape)
        diffX1 = searchRange + i
        diffX2 = dimX - searchRange + i
        diffY1 = searchRange + j
        diffY2 = dimY - searchRange + j
        #print("diff: ", diffX1, diffX2, diffY1, diffY2)
        compare = img_compare[diffY1:diffY2, diffX1:diffX2, :]
        diff3=(ref-compare)**2
        diff=(diff3[:, :, 0]+diff3[:, :, 1]+diff3[:, :, 2]) / 3
        #print("diff shape: ", compare.shape)
        tSum=np.zeros((dimY-2*searchRange-2*tileSpan, dimX-2*searchRange-
2*tileSpan))
        #print("sum shape: ", sum.shape)
        dimDiffY=diff.shape[0]
        dimDiffX=diff.shape[1]
        for x in range(-tileSpan, tileSpan + 1):
            for y in range(-tileSpan, tileSpan + 1):
                #print(tileSpan+y, dimDiffY-tileSpan+y)
                #print(tileSpan+x, dimDiffX-tileSpan+x)
                tSum += diff[tileSpan+y:dimDiffY-tileSpan+y, tileSpan+x:dimDiffX-
tileSpan+x]
        tSum = tSum / (tileSpan + 1)**2
        #print("sum complete")
        for x in range(dimDFDx):
            for y in range(dimDFDy):
                if tSum[y,x]<DFD[y,x]:
                    DFD[y,x]=tSum[y,x]
                    DFDx[y,x]=i
                    DFDy[y,x]=j
end = time.time()
print(end - start)

#filename=filePrefix + str(idxFile + idxStart) + fileSuffix
np.save(pathImage + filePrefix +"DFD_" + str(m + idxStart), DFD)
np.save(pathImage + filePrefix +"DFDx_" + str(m + idxStart), DFDx)
np.save(pathImage + filePrefix +"DFDy_" + str(m + idxStart), DFDy)

windowSpanX=20
windowSpanY=20
windowShiftX=30
windowShiftY=30
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz
pixelSize=0.1 #m
compensateY= speedY / frameRate / pixelSize
#print("Compesantion vector Y: ", compensateY)

```

```

f=plt.figure(dpi=200)
f.set_figwidth(8)
f.set_figheight(12)

(DFDheight,DFDwidth)=DFD.shape
imgplot_1 = plt.imshow(img_ref)
for x in range(windowSpanY, DFDwidth-windowSpanX+1, windowShiftX):
    for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
        vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        #print(vX0, vX0+vX , vY0, vY0+vY)
        if vX>0:
            lineColor='r'
        else:
            lineColor='b'
        vX1=vX0+4*vX
        vY1=vY0+4*(vY-compensateY)
        if vX1<0:
            vX1=0
        if vY1<0:
            vY1=0
        if vX1>dimX-1:
            vX1=dimX-1
        if vY1>dimY-1:
            vY1=dimY-1
        plt.plot([vX0, vX1], [vY0, vY1], lineColor)
plt.axis('off')
filename=imagePrefix + str(m + idxStart) + imageSuffix
plt.savefig(pathImage + filename, bbox_inches = 'tight', pad_inches = 0)
plt.close()

File: gimbalo_2029.jpg
1816.0543775558472
File: gimbalo_2030.jpg
3456.280796289444
File: gimbalo_2031.jpg
5028.192597866058
File: gimbalo_2032.jpg
6547.916834592819
File: gimbalo_2033.jpg
8299.202639102936
File: gimbalo_2034.jpg
10044.22109246254
File: gimbalo_2035.jpg
11599.370234012604
File: gimbalo_2036.jpg
13066.518587589264
File: gimbalo_2037.jpg
14496.12367939949
File: gimbalo_2038.jpg
16149.747946739197
File: gimbalo_2039.jpg
17666.129164218903
File: gimbalo_2040.jpg
19223.922142982483
File: gimbalo_2041.jpg
20739.48153400421
File: gimbalo_2042.jpg
22226.710700511932
File: gimbalo_2043.jpg
23812.226888418198

```

File: gimbalo_2044.jpg
25422.758392333984
File: gimbalo_2045.jpg
26900.09226822853
File: gimbalo_2046.jpg
28395.788303375244
File: gimbalo_2047.jpg
30147.382929325104
File: gimbalo_2048.jpg
31781.119401693344
File: gimbalo_2049.jpg
33466.231439590454
File: gimbalo_2050.jpg
35157.80604124069
File: gimbalo_2051.jpg
36891.414430618286
File: gimbalo_2052.jpg
38489.708997011185
File: gimbalo_2053.jpg
40049.74000477791
File: gimbalo_2054.jpg
41574.27495241165
File: gimbalo_2055.jpg
43239.83487820625
File: gimbalo_2056.jpg
44831.117822647095
File: gimbalo_2057.jpg
46346.033939123154
File: gimbalo_2058.jpg
48182.31793999672
File: gimbalo_2059.jpg
49727.626991033554
File: gimbalo_2060.jpg
51300.64799642563
File: gimbalo_2061.jpg
52833.648718357086
File: gimbalo_2062.jpg
54552.876072883606
File: gimbalo_2063.jpg
56120.84501838684
File: gimbalo_2064.jpg
57741.27485251427
File: gimbalo_2065.jpg
59252.365793943405
File: gimbalo_2066.jpg
60993.72823882103
File: gimbalo_2067.jpg
62690.225244522095
File: gimbalo_2068.jpg
64438.18321061134
File: gimbalo_2069.jpg
65994.39007735252
File: gimbalo_2070.jpg
68806.37795114517
File: gimbalo_2071.jpg
71078.63471508026
File: gimbalo_2072.jpg
73408.82669043541
File: gimbalo_2073.jpg
75953.12694740295
File: gimbalo_2074.jpg
78582.11592555046
File: gimbalo_2075.jpg

```
81581.80021166801
File: gimbalo_2076.jpg
84110.01959252357
File: gimbalo_2077.jpg
86640.03912901878
File: gimbalo_2078.jpg
In [9]:
plt.savefig(filename,bbox_inches=0)
<Figure size 640x480 with 0 Axes>
In [ ]:
```

WORKSTATION SCOUT IMAGE RIP CURRENT ANALYSIS

rip_current_DFDc_v01_GSDSEF.ipynb

- Language: Python in Jupyter Notebook
- Host: Workstation
- Usage:
 - Beach image data analysis example plots
- Function:
 - Show images under analysis
 - Load DFD and information weights
 - Perform rip current analysis with depth and risk model
 - Show the rip current flow estimation
 - Compare unweighted, depth, and depth+risk model rip flow

```
In [1]:
# TensorFlow and tf.keras
import tensorflow as tf
print(tf.__version__)
from tensorflow import keras
# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import time
import math
import glob
import os
2024-02-25 18:24:24.791395: I tensorflow/core/util/port.cc:113] oneDNN custom
operations are on. You may see slightly different numerical results due to floating-
point round-off errors from different computation orders. To turn them off, set the
environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-02-25 18:24:24.814770: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already been
registered
2024-02-25 18:24:24.814791: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT
factory: Attempting to register factory for plugin cuFFT when one has already been
registered
2024-02-25 18:24:24.815341: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already
been registered
2024-02-25 18:24:24.819550: I tensorflow/core/platform/cpu_feature_guard.cc:182] This
TensorFlow binary is optimized to use available CPU instructions in performance-
critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
2024-02-25 18:24:25.235909: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38]
TF-TRT Warning: Could not find TensorRT
2.15.0
In [2]:
start = time.time()
#
imagePrefix='DFD1_gimbal0_'
```

```

imageSuffix=' .png'
pathImage=' ../scout_video_09/'
filePrefix='gimbal0_'
fileSuffix=' .jpg'
#
tileSpan=15
dimX=1920
dimY=1080
searchRange=10
searchStep=1
windowSpanX=20
windowSpanY=20
windowShiftX=30
windowShiftY=30
#compensation parameters
speedY=5 #m/s
frameRate=10 #Hz
pixelSize=0.1 #m
compensateY= speedY / frameRate / pixelSize
#Rip current analysis set up
ripNoImages=3
ripNoRowWindow=10 #windowSpanY and windowShiftY
#TF batch formation
tfBatchSize=2
tfIdxStart=2000
tfIdxEnd=2070
tfBatchEpoch=2
tfIteration=5
#
checkpointPath=". /cp_01/"
isExist = os.path.exists(checkpointPath)
if not isExist:
    os.mkdir(checkpointPath)
#cp_01: 4096, 1024, 256, 64, 2, SGD 0.00001
In [3]:
idxFile=0
idxStart=2000

#display two images

f=plt.figure(dpi=200)
f.set_figwidth(8)
f.set_figheight(12)
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
print(pathImage + filename)
img_0 = mpimg.imread(pathImage + filename) / 256

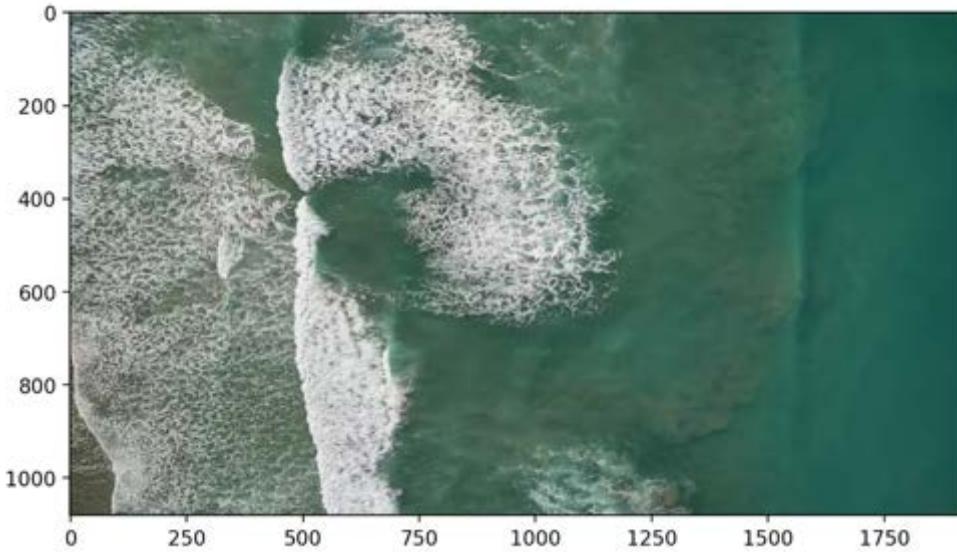
print(np.min(img_0))
print(np.max(img_0))
print(img_0.shape)
print(img_0[1079,1919,2])
print(type(img_0[0,0,0]))

imgplot_0 = plt.imshow(img_0)
plt.show()

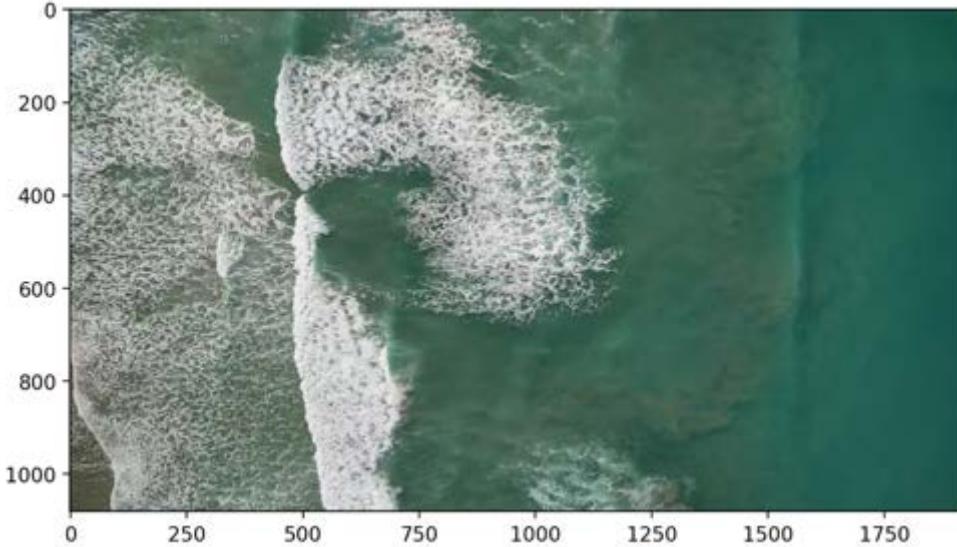
f=plt.figure(dpi=200)
f.set_figwidth(8)
f.set_figheight(12)
filename=filePrefix + str(idxFile + idxStart + 1) + fileSuffix
print(pathImage + filename)
img_0 = mpimg.imread(pathImage + filename) / 256
imgplot_0 = plt.imshow(img_0)

```

```
plt.show()
.../scout_video_09/gimbal0_2000.jpg
0.00390625
0.99609375
(1080, 1920, 3)
0.2890625
<class 'numpy.float64'>
```



```
.../scout_video_09/gimbal0_2001.jpg
```



```
In [4]:
n=2000
m=0
#DFD=np.load(pathImage + filePrefix +"DFD_" + str(n + m) + ".npy")
DFDx=np.load(pathImage + filePrefix +"DFDx_" + str(n + m) + ".npy")
DFDy=np.load(pathImage + filePrefix +"DFDy_" + str(n + m) + ".npy")
DFDEC=np.load(pathImage + filePrefix +"DFDEC_" + str(n + m) + ".npy")
```

```
(DFDheight,DFDwidth)=DFDx.shape
print(DFDheight, DFDwidth)
```

```
1030 1870
```

```
In [5]:
count=0
```

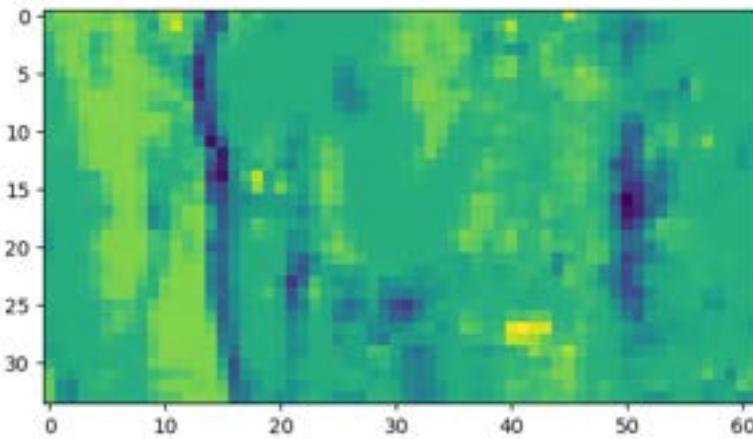
```

for x in range(windowSpanX, DFDwidth-windowSpanX+1, windowShiftX):
    count+=1
print("X count: ", count)
print("last x:" , x)
count=0
for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
    count+=1
print("Y count: ", count)
print("last y:" , y)

print((DFDwidth-windowSpanX*2)/windowShiftX)
print((DFDheight-windowSpanY*2)/windowShiftY)
svSx=int((DFDwidth-windowSpanX*2)/windowShiftX)
svSy=int((DFDheight-windowSpanY*2)/windowShiftY)

#Initialize Averaged DFD vector matrix
vectorSampleX=np.zeros((svSy+1, svSx+1))
vectorSampleY=np.zeros((svSy+1, svSx+1))
X count: 62
last x: 1850
Y count: 34
last y: 1010
61.0
33.0
In [6]:
listImage=[]
listVector=[]
n=2000
for m in range(0,ripNoImages):
    #load DFD vector data
    DFDx=np.load(pathImage + filePrefix +"DFDx_" + str(n + m) + ".npy")
    DF Dy=np.load(pathImage + filePrefix +"DFDy_" + str(n + m) + ".npy")
    DFDEC=np.load(pathImage + filePrefix +"DFDEC_" + str(n + m) + ".npy")
    (DFDheight,DFDwidth)=DFDx.shape
    idxX=0
    for x in range(windowSpanX, DFDwidth-windowSpanX+1, windowShiftX):
        idxY=0
        for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
            vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
            vY=np.mean(DFDy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
            vC=np.mean(DFDEC[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
            vectorSampleX[idxY, idxX]=vX*vC
            vectorSampleY[idxY, idxX]=vY
            idxY+=1
        idxX+=1
    #
    imgplot_1 = plt.imshow(vectorSampleX)
    #imgplot_2 = plt.imshow(vectorSampleY)
    print(idxY, idxX)
34 62

```



In [7]:

```

oceanDepthShallow=0.1
oceanDepthDeep=1
oceanDepthSlice=np.arange(oceanDepthShallow, oceanDepthDeep, (oceanDepthDeep-
oceanDepthShallow)/(svSx+1))
print(oceanDepthSlice.shape)
print(oceanDepthSlice)

oceanDepthMatrix=np.repeat([oceanDepthSlice], ripNoRowWindow, axis=0)
print(oceanDepthMatrix.shape)
print(oceanDepthMatrix)
(62,)
[0.1      0.11451613 0.12903226 0.14354839 0.15806452 0.17258065
 0.18709677 0.2016129  0.21612903 0.23064516 0.24516129 0.25967742
 0.27419355 0.28870968 0.30322581 0.31774194 0.33225806 0.34677419
 0.36129032 0.37580645 0.39032258 0.40483871 0.41935484 0.43387097
 0.4483871  0.46290323 0.47741935 0.49193548 0.50645161 0.52096774
 0.53548387 0.55      0.56451613 0.57903226 0.59354839 0.60806452
 0.62258065 0.63709677 0.6516129  0.66612903 0.68064516 0.69516129
 0.70967742 0.72419355 0.73870968 0.75322581 0.76774194 0.78225806
 0.79677419 0.81129032 0.82580645 0.84032258 0.85483871 0.86935484
 0.88387097 0.8983871  0.91290323 0.92741935 0.94193548 0.95645161
 0.97096774 0.98548387]
(10, 62)
[[0.1      0.11451613 0.12903226 0.14354839 0.15806452 0.17258065
  0.18709677 0.2016129  0.21612903 0.23064516 0.24516129 0.25967742
  0.27419355 0.28870968 0.30322581 0.31774194 0.33225806 0.34677419
  0.36129032 0.37580645 0.39032258 0.40483871 0.41935484 0.43387097
  0.4483871  0.46290323 0.47741935 0.49193548 0.50645161 0.52096774
  0.53548387 0.55      0.56451613 0.57903226 0.59354839 0.60806452
  0.62258065 0.63709677 0.6516129  0.66612903 0.68064516 0.69516129
  0.70967742 0.72419355 0.73870968 0.75322581 0.76774194 0.78225806
  0.79677419 0.81129032 0.82580645 0.84032258 0.85483871 0.86935484
  0.88387097 0.8983871  0.91290323 0.92741935 0.94193548 0.95645161
  0.97096774 0.98548387]
[0.1      0.11451613 0.12903226 0.14354839 0.15806452 0.17258065
 0.18709677 0.2016129  0.21612903 0.23064516 0.24516129 0.25967742
 0.27419355 0.28870968 0.30322581 0.31774194 0.33225806 0.34677419
 0.36129032 0.37580645 0.39032258 0.40483871 0.41935484 0.43387097
 0.4483871  0.46290323 0.47741935 0.49193548 0.50645161 0.52096774
 0.53548387 0.55      0.56451613 0.57903226 0.59354839 0.60806452
 0.62258065 0.63709677 0.6516129  0.66612903 0.68064516 0.69516129
 0.70967742 0.72419355 0.73870968 0.75322581 0.76774194 0.78225806
 0.79677419 0.81129032 0.82580645 0.84032258 0.85483871 0.86935484
 0.88387097 0.8983871  0.91290323 0.92741935 0.94193548 0.95645161
 0.97096774 0.98548387]

```



```

0.79677419 0.81129032 0.82580645 0.84032258 0.85483871 0.86935484
0.88387097 0.8983871 0.91290323 0.92741935 0.94193548 0.95645161
0.97096774 0.98548387]
[0.1 0.11451613 0.12903226 0.14354839 0.15806452 0.17258065
0.18709677 0.2016129 0.21612903 0.23064516 0.24516129 0.25967742
0.27419355 0.28870968 0.30322581 0.31774194 0.33225806 0.34677419
0.36129032 0.37580645 0.39032258 0.40483871 0.41935484 0.43387097
0.4483871 0.46290323 0.47741935 0.49193548 0.50645161 0.52096774
0.53548387 0.55 0.56451613 0.57903226 0.59354839 0.60806452
0.62258065 0.63709677 0.6516129 0.66612903 0.68064516 0.69516129
0.70967742 0.72419355 0.73870968 0.75322581 0.76774194 0.78225806
0.79677419 0.81129032 0.82580645 0.84032258 0.85483871 0.86935484
0.88387097 0.8983871 0.91290323 0.92741935 0.94193548 0.95645161
0.97096774 0.98548387]
[0.1 0.11451613 0.12903226 0.14354839 0.15806452 0.17258065
0.18709677 0.2016129 0.21612903 0.23064516 0.24516129 0.25967742
0.27419355 0.28870968 0.30322581 0.31774194 0.33225806 0.34677419
0.36129032 0.37580645 0.39032258 0.40483871 0.41935484 0.43387097
0.4483871 0.46290323 0.47741935 0.49193548 0.50645161 0.52096774
0.53548387 0.55 0.56451613 0.57903226 0.59354839 0.60806452
0.62258065 0.63709677 0.6516129 0.66612903 0.68064516 0.69516129
0.70967742 0.72419355 0.73870968 0.75322581 0.76774194 0.78225806
0.79677419 0.81129032 0.82580645 0.84032258 0.85483871 0.86935484
0.88387097 0.8983871 0.91290323 0.92741935 0.94193548 0.95645161
0.97096774 0.98548387]]
In [8]:
#amplify stronger vectors with squaring
#vXabs=np.abs(vectorSampleX)
#vectorInputX=np.multiply(vectorSampleX,vXabs)
vectorInputX=vectorSampleX
#print(vXsqr.shape)
#imgplot_1 = plt.imshow(vXsqr)
#print(np.sum(vXsqr))

#this is straight sum of x vector as rip current flow
ripValue=np.zeros((svSy-ripNoRowWindow+2,1))
for k in range(0,int(svSy)-ripNoRowWindow+2):
    tV=np.mean(vectorInputX[k:k+ripNoRowWindow,:])
    ripValue[k]=tV

#print(ripValue)
#print(len(ripValue))
#print(k+ripNoRowWindow)

#linear ocean depth model based rip current flow
oceanDepthShallow=0.1
oceanDepthDeep=1
oceanDepthSlice=np.arange(oceanDepthShallow, oceanDepthDeep, (oceanDepthDeep-oceanDepthShallow)/(svSx+1))
oceanDepthMatrix=np.repeat([oceanDepthSlice], ripNoRowWindow, axis=0)
ripDepth=np.zeros((svSy-ripNoRowWindow+2,1))
for k in range(0,int(svSy)-ripNoRowWindow+2):
    tV=np.mean( np.multiply( vectorInputX[k:k+ripNoRowWindow,:], oceanDepthMatrix) )
    ripDepth[k]=tV

#print(ripDepth)
#print(len(ripDepth))
#print(k+ripNoRowWindow)

#quadratic ocean risk model based rip current flow
oceanRiskSlice=np.multiply(oceanDepthSlice, oceanDepthSlice)
oceanRiskMatrix=np.multiply(oceanDepthMatrix, oceanDepthMatrix)
ripRisk=np.zeros((svSy-ripNoRowWindow+2,1))

```

```

for k in range(0,int(svSy)-ripNoRowWindow+2):
    tV=np.mean( np.multiply( vectorInputX[k:k+ripNoRowWindow,:], oceanRiskMatrix) )
    ripRisk[k]=tV

print(ripRisk)
print(len(ripRisk))
#rint(k+ripNoRowWindow)
[[ 0.0236601 ]
 [ 0.02051126]
 [ 0.02220663]
 [ 0.02152914]
 [ 0.01439843]
 [ 0.00446977]
 [-0.00896785]
 [-0.02341356]
 [-0.03575785]
 [-0.0400074 ]
 [-0.03962215]
 [-0.03931009]
 [-0.0460352 ]
 [-0.05083071]
 [-0.05177122]
 [-0.0548917 ]
 [-0.05809149]
 [-0.05402431]
 [-0.04257553]
 [-0.04002267]
 [-0.04154217]
 [-0.04074636]
 [-0.03288834]
 [-0.0233684 ]
 [-0.01951226]]
25
In [16]:
f=plt.figure(dpi=100)
f.set_figwidth(10)
f.set_figheight(6)

idxFile=0
filename=filePrefix + str(idxFile + idxStart) + fileSuffix
img_ref = mpimg.imread(pathImage + filename) / 256
imgplot_1 = plt.imshow(img_ref)

#draw average DFD vectors
for x in range(windowSpanX, DFDwidth-windowSpanX+1, windowShiftX ):
    for y in range(windowSpanY, DFDheight-windowSpanY+1, windowShiftY):
        vX=np.mean(DFDx[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vY=np.mean(DFDy[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vC=np.mean(DFDEC[y-windowSpanY:y+windowSpanY,x-windowSpanX:x+windowSpanX])
        vX0=x + searchRange + tileSpan
        vY0=y + searchRange + tileSpan
        #print(vX0, vX0+vX , vY0, vY0+vY)
        if vX>0:
            lineColor='r'
        else:
            lineColor='b'
        plt.plot([vX0, vX0+4*vX*vC], [vY0, vY0+4*(vY-compensateY)*vC], lineColor)

#draw neural flow line
#plt.plot([dimX / 2, dimX / 2], [0, dimY], 'k')
#plot rip current value
for i in range(len(ripValue)):
    vY0 = searchRange + tileSpan + windowSpanY + windowShiftY * i

```

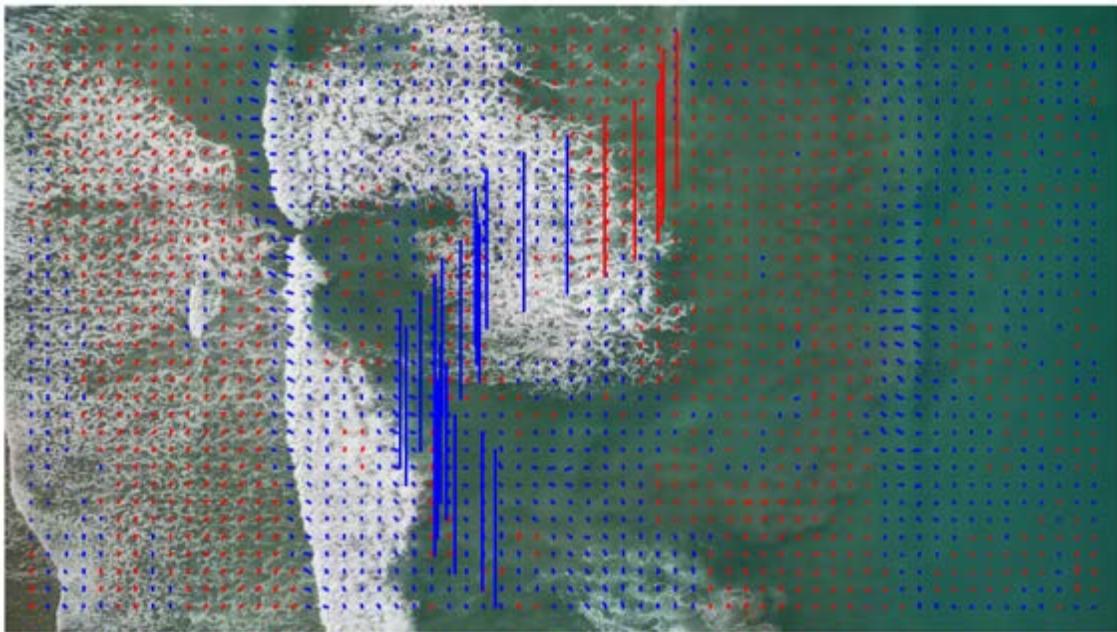
```

vY1 = searchRange + tileSpan + windowSpanY + windowShiftY * (i + ripNoRowWindow -
1)
vX0 = searchRange + tileSpan
xX1 = dimX - searchRange - tileSpan - windowSpanX
vXC = dimX / 2 + ripValue[i]*4000
if ripValue[i]>0:
    lineColor='c'
else:
    lineColor='g'
#plt.plot([vXC, vXC], [vY0, vY1], lineColor+":")

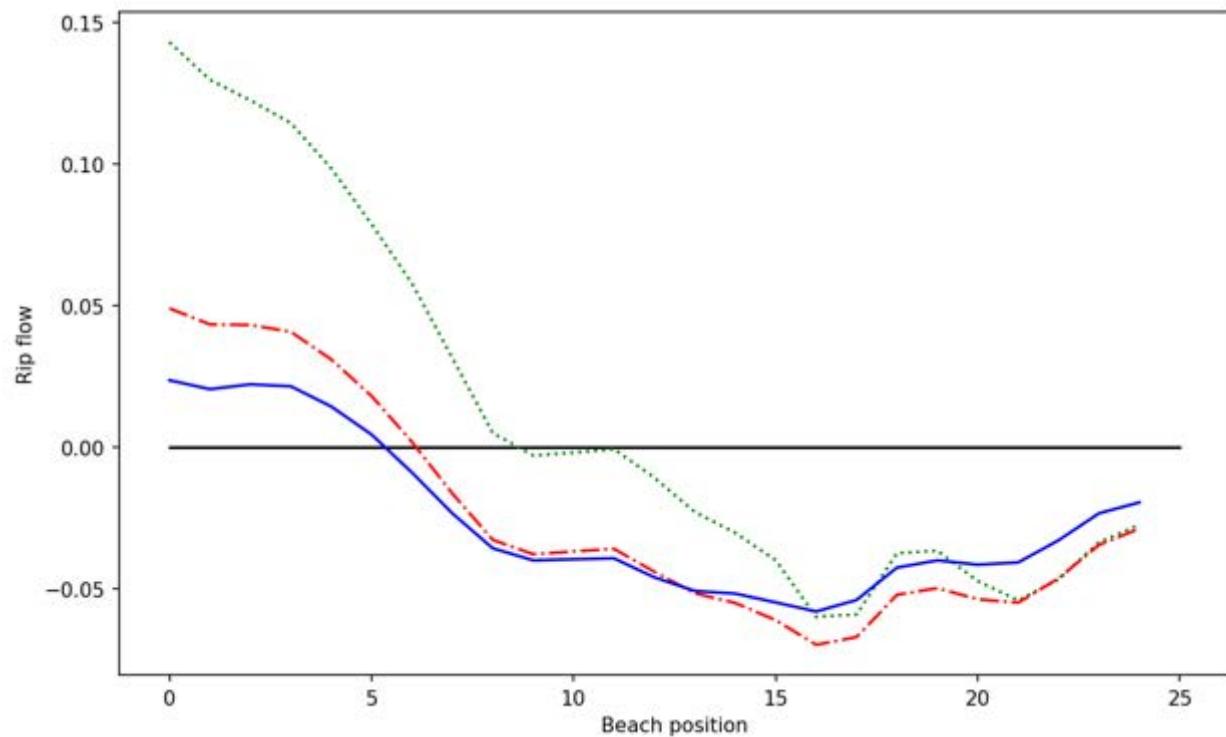
#plot rip depth value
for i in range(len(ripValue)):
    vY0 = searchRange + tileSpan + windowSpanY + windowShiftY * i
    vY1 = searchRange + tileSpan + windowSpanY + windowShiftY * (i + ripNoRowWindow -
1)
    vX0 = searchRange + tileSpan
    xX1 = dimX - searchRange - tileSpan - windowSpanX
    vXC = dimX / 2 + ripDepth[i]*4000
    if ripDepth[i]>0:
        lineColor='c'
    else:
        lineColor='g'
    #plt.plot([vXC, vXC], [vY0, vY1], lineColor+"--")
print(i)

#plot rip risk value
for i in range(len(ripValue)):
    vY0 = searchRange + tileSpan + windowSpanY + windowShiftY * i
    vY1 = searchRange + tileSpan + windowSpanY + windowShiftY * (i + ripNoRowWindow -
1)
    vX0 = searchRange + tileSpan
    xX1 = dimX - searchRange - tileSpan - windowSpanX
    vXC = dimX / 2 + ripDepth[i]*4000
    if ripRisk[i]>0:
        lineColor='r'
    else:
        lineColor='b'
    plt.plot([vXC, vXC], [vY0, vY1], lineColor)
print(i)
plt.axis('off')
24
24
Out[16]:
(-0.5, 1919.5, 1079.5, -0.5)

```



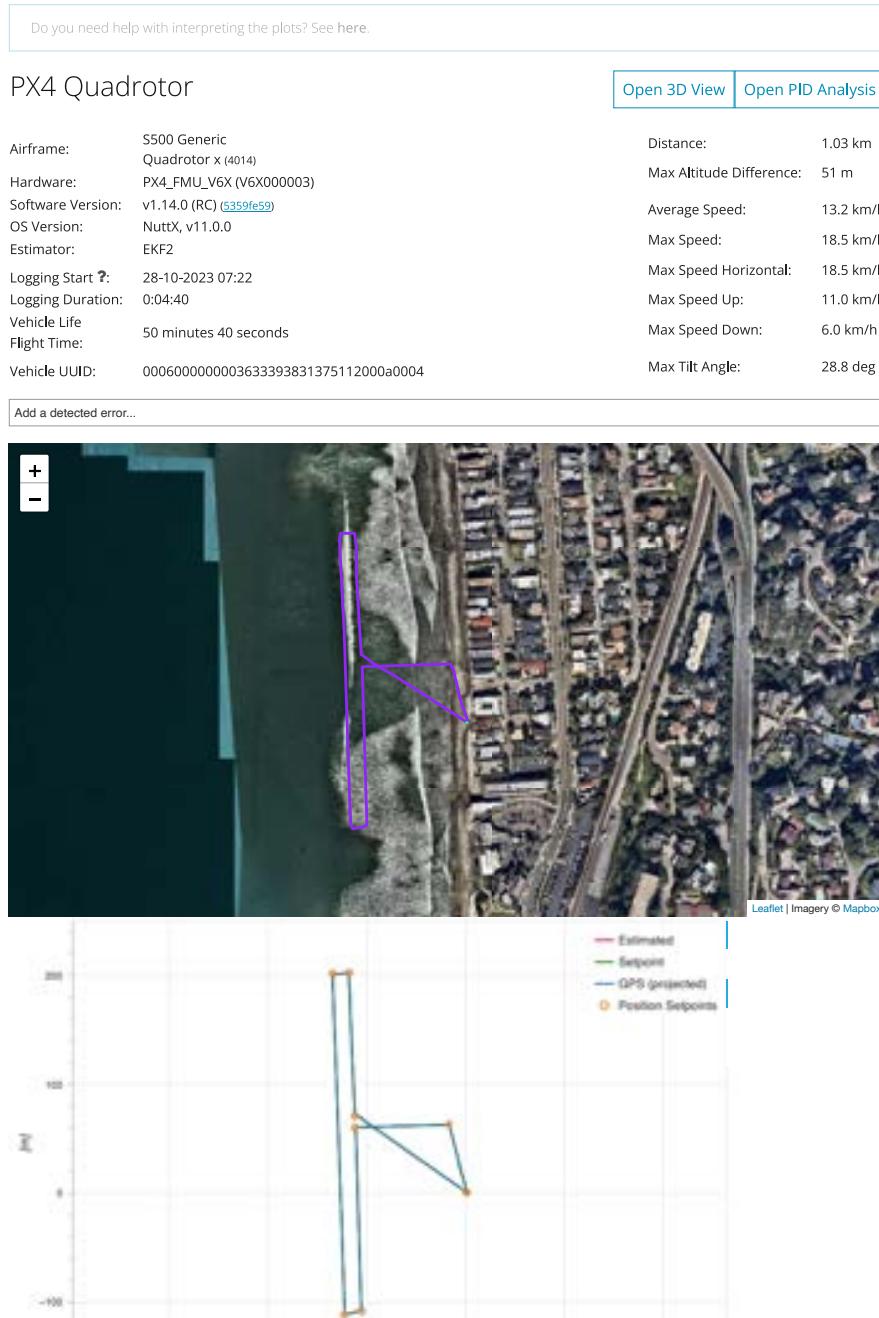
```
In [14]:  
f=plt.figure(dpi=150)  
f.set_figwidth(10)  
f.set_figheight(6)  
plt.plot([])  
plt.plot([0,len(ripRisk)], [0,0], 'k')  
plt.plot(ripValue, 'g:')  
plt.plot(ripDepth, 'r-.')  
plt.plot(ripRisk, 'b')  
plt.xlabel('Beach position')  
plt.ylabel('Rip flow')  
Out[14]:  
Text(0, 0.5, 'Rip flow')
```



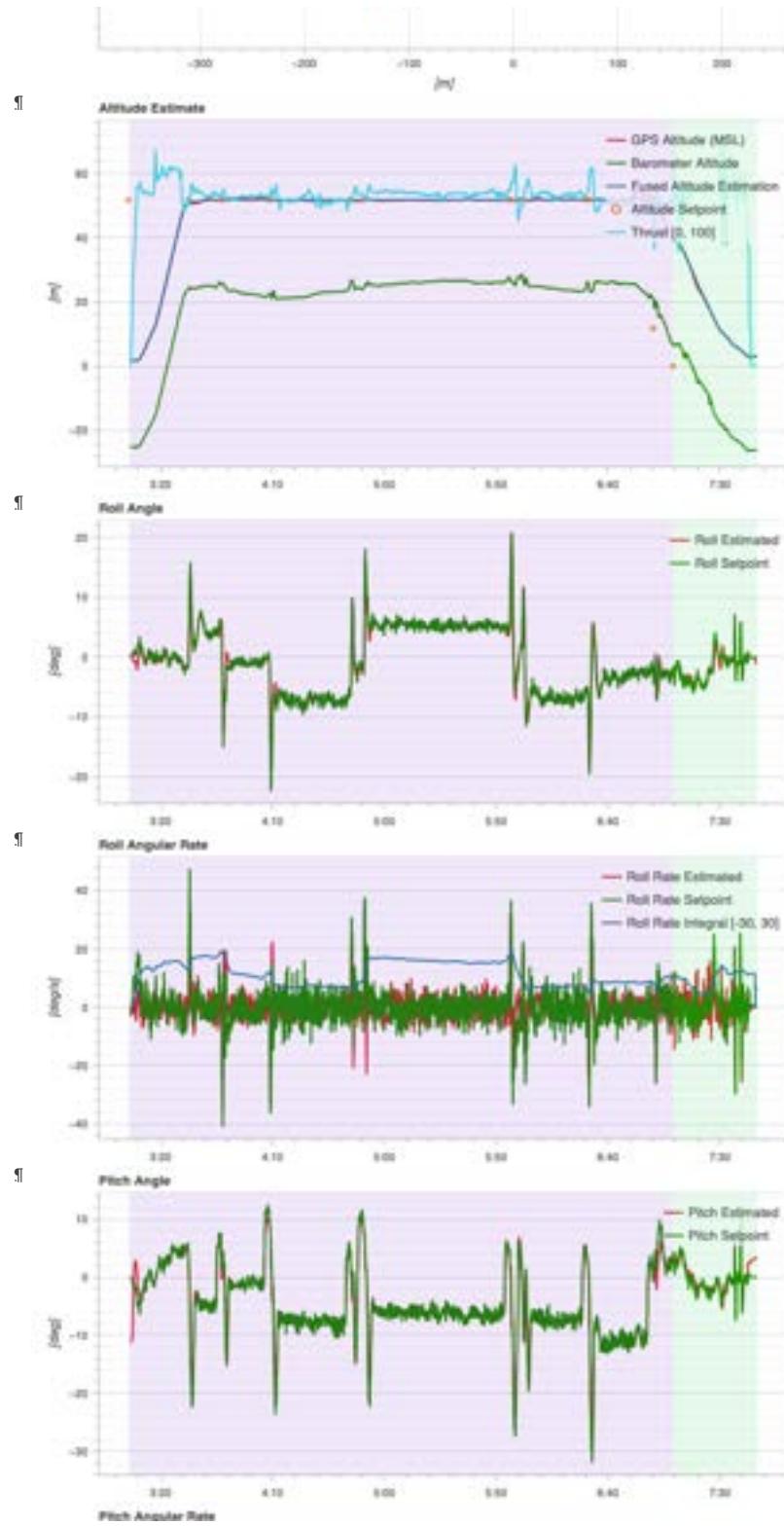
Appendix F. Flight reports

All images were created through PX4 Flight Review using scout drone flight data and captured by the author (<https://review.px4.io/>)

Scout flight #1 – page 1/8



Scout flight #1 – page 2/8 (pages 3-8 are omitted)



Scout flight #2 – page 1/8

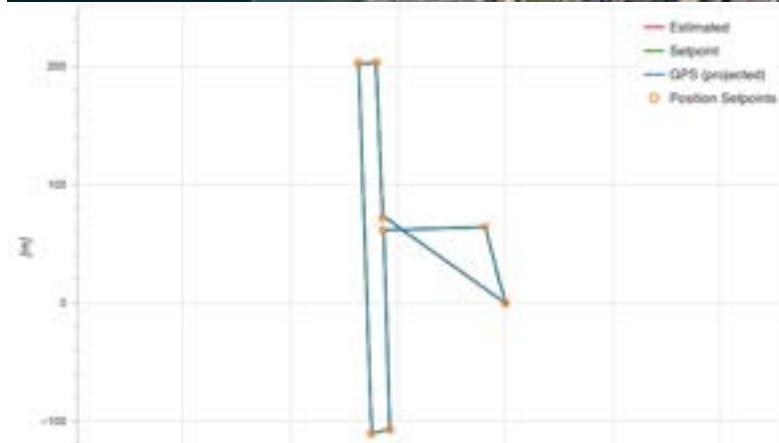
Do you need help with interpreting the plots? See here.

PX4 Quadrotor

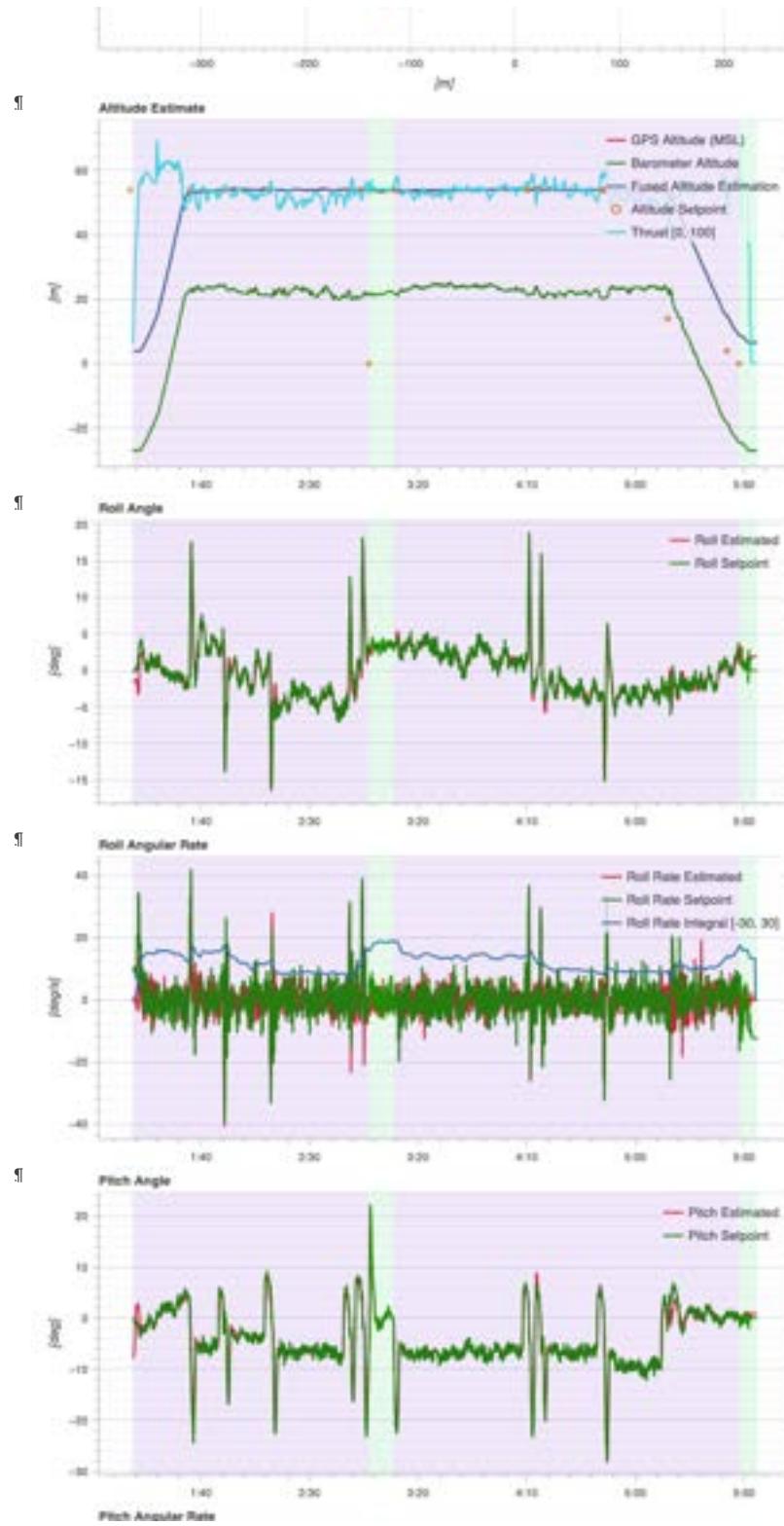
[Open 3D View](#) [Open PID Analysis](#)

Airframe:	S500 Generic Quadrotor X (4014)	Distance:	1.04 km
Hardware:	PX4_FMU_V6X (V6X000003)	Max Altitude Difference:	50 m
Software Version:	v1.14.0 (RC) (5359f659)	Average Speed:	12.9 km/h
OS Version:	Nuttx, v11.0.0	Max Speed:	18.4 km/h
Estimator:	EKF2	Max Speed Horizontal:	18.4 km/h
Logging Start ?:	28-10-2023 07:39	Max Speed Up:	11.1 km/h
Logging Duration:	0:04:48	Max Speed Down:	6.1 km/h
Vehicle Life	55 minutes 18 seconds	Max Tilt Angle:	27.5 deg
Flight Time:			
Vehicle UUID:	000600000003633393831375112000a0004		

Add a detected error...



Scout flight #2 – page 2/8 (pages 3-8 are omitted)



Scout flight #3 – page 1/8

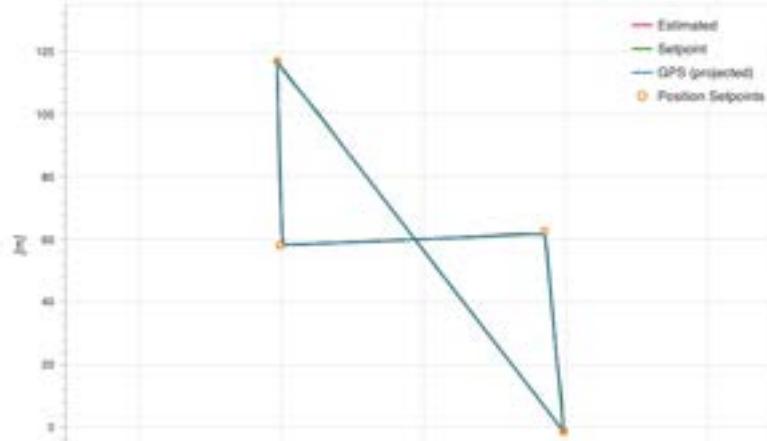
Do you need help with interpreting the plots? See here.

PX4 Quadrotor

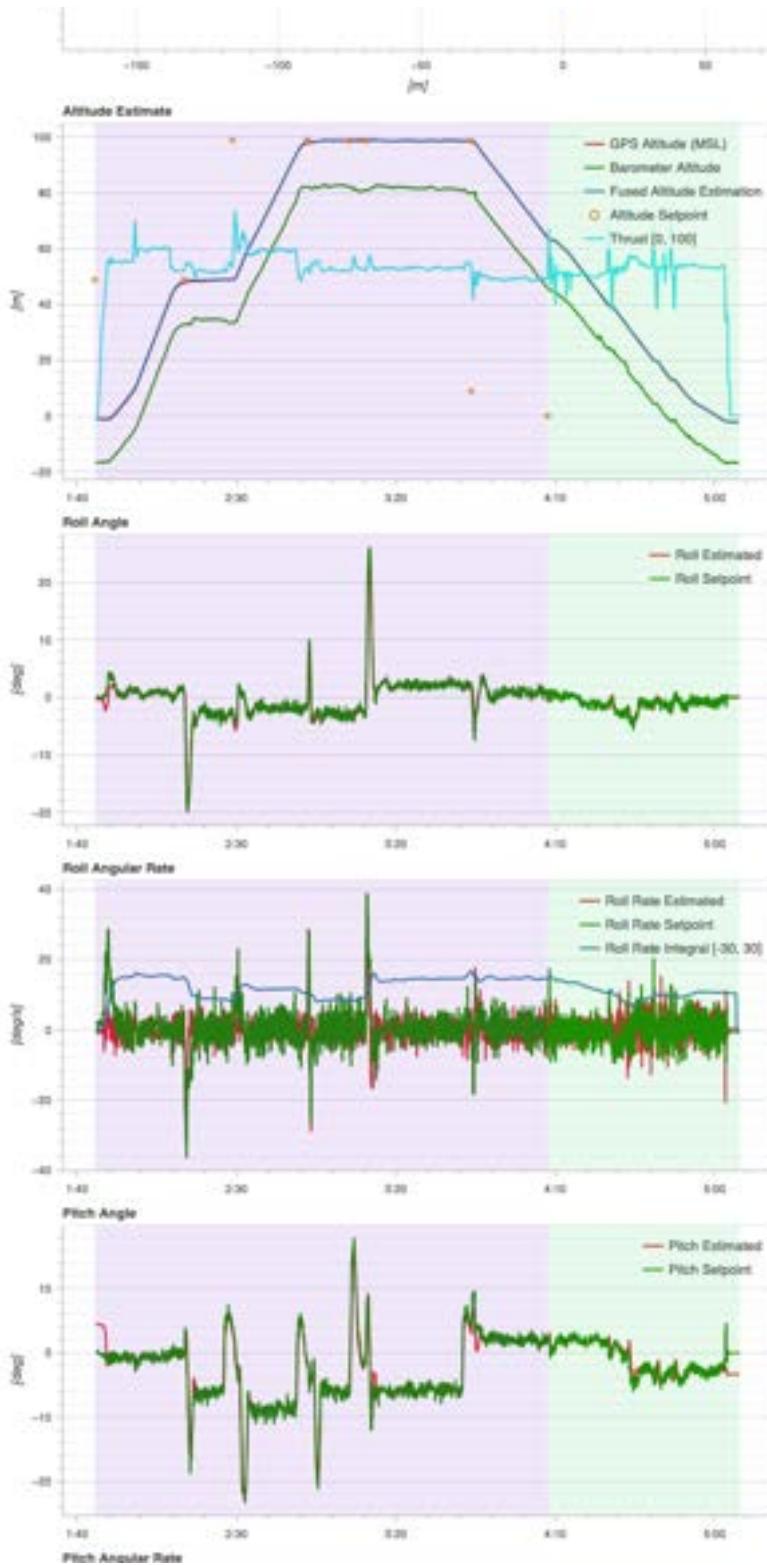
[Open 3D View](#) [Open PID Analysis](#)

Airframe:	S500 Generic Quadrotor X (4014)	Distance:	534.5 m
Hardware:	PX4_FMU_V6X (V6X000003)	Max Altitude Difference:	101 m
Software Version:	v1.14.0 (RC) (5359f659)	Average Speed:	9.4 km/h
OS Version:	Nuttx, v11.0.0	Max Speed:	20.0 km/h
Estimator:	EKF2	Max Speed Horizontal:	18.2 km/h
Logging Start ?:	17-11-2023 16:33	Max Speed Up:	10.8 km/h
Logging Duration:	0:03:22	Max Speed Down:	6.0 km/h
Vehicle Life	1 hours 4 seconds	Max Tilt Angle:	23.2 deg
Flight Time:			
Vehicle UUID:	000600000003633393831375112000a0004		

Add a detected error...



Scout flight #3 – page 2/8 (pages 3-8 are omitted)



Scout flight #4 – page 1/8

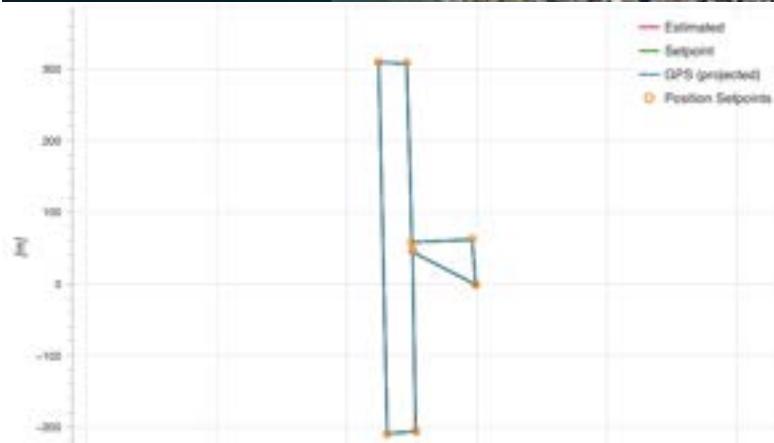
Do you need help with interpreting the plots? See here.

PX4 Quadrotor

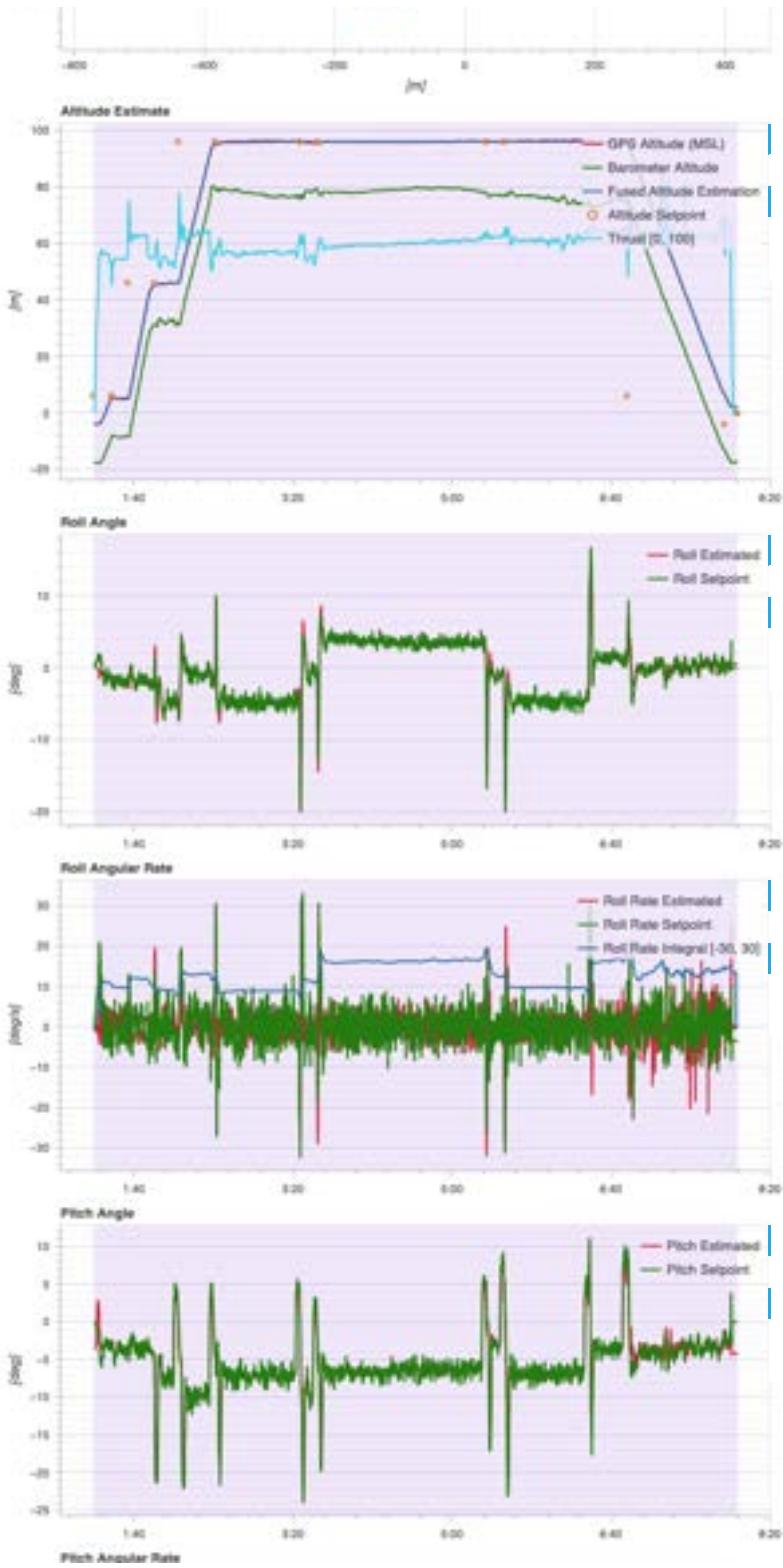
[Open 3D View](#) [Open PID Analysis](#)

Airframe:	S500 Generic Quadrotor X (4014)	Distance:	1.52 km
Hardware:	PX4_FMU_V6X (V6X000003)	Max Altitude Difference:	100 m
Software Version:	v1.14.0 (RC) (5359fe59)	Average Speed:	13.5 km/h
OS Version:	Nuttx, v11.0.0	Max Speed:	20.0 km/h
Estimator:	EKF2	Max Speed Horizontal:	18.3 km/h
Logging Start ?:	17-11-2023 16:42	Max Speed Up:	10.8 km/h
Logging Duration:	0:06:45	Max Speed Down:	6.9 km/h
Vehicle Life	1 hours 3 minutes 24 seconds	Max Tilt Angle:	23.6 deg
Flight Time:			
Vehicle UUID:	000600000003633393831375112000a0004		

Add a detected error...



Scout flight #4 – page 2/8 (pages 3-8 are omitted)



Scout flight #5 – page 1/8

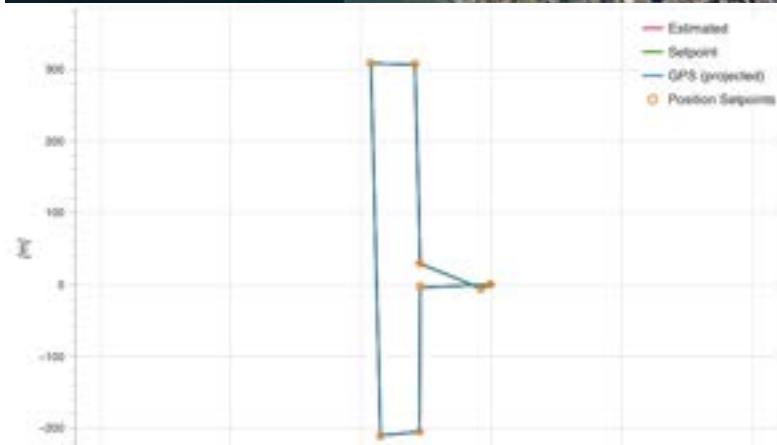
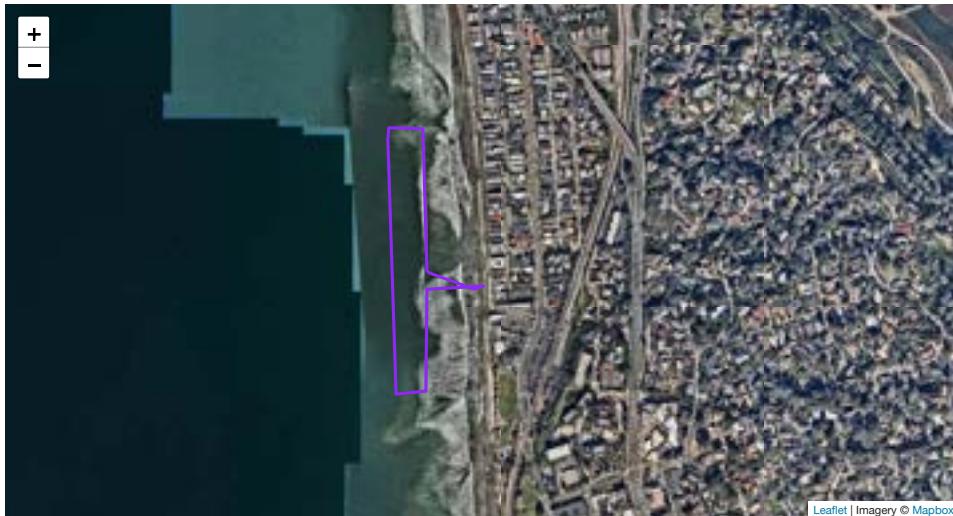
Do you need help with interpreting the plots? See here.

PX4 Quadrotor

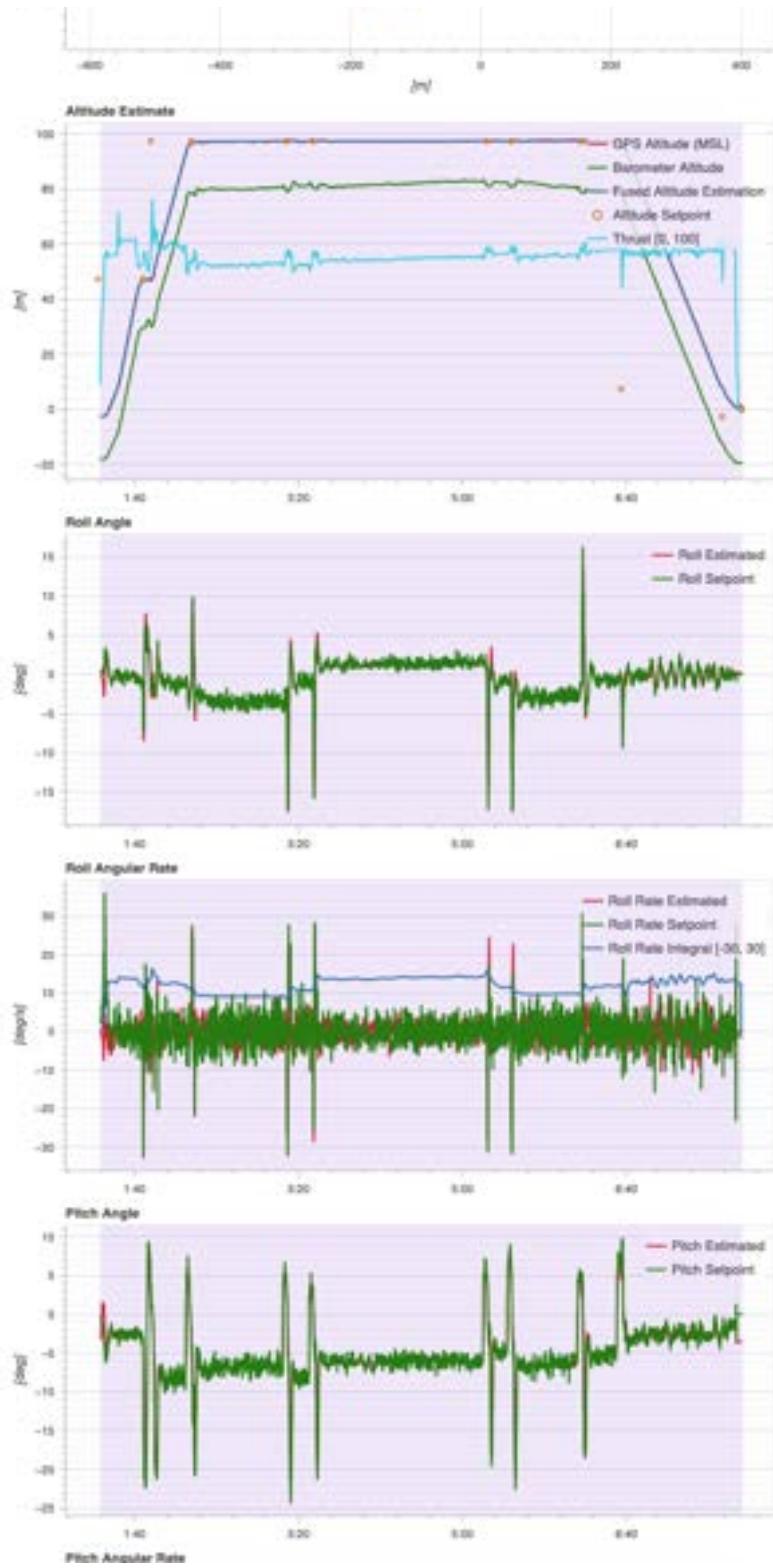
[Open 3D View](#) [Open PID Analysis](#)

Airframe:	S500 Generic	Distance:	1.50 km
Hardware:	Quadrotor x (4014)	Max Altitude Difference:	100 m
Software Version:	PX4_FMU_V6X (V6X000003)	Average Speed:	13.8 km/h
OS Version:	v1.14.0 (RC) (5359fe59)	Max Speed:	19.9 km/h
Estimator:	EKF2	Max Speed Horizontal:	18.3 km/h
Logging Start ?:	17-11-2023 17:00	Max Speed Up:	10.8 km/h
Logging Duration:	0:06:33	Max Speed Down:	6.4 km/h
Vehicle Life	1 hours 10 minutes 6 seconds	Max Tilt Angle:	22.6 deg
Flight Time:			
Vehicle UUID:	000600000003633393831375112000a0004		

Add a detected error...



Scout flight #5 – page 2/8 (pages 3-8 are omitted)



Scout flight #6 – page 1/8

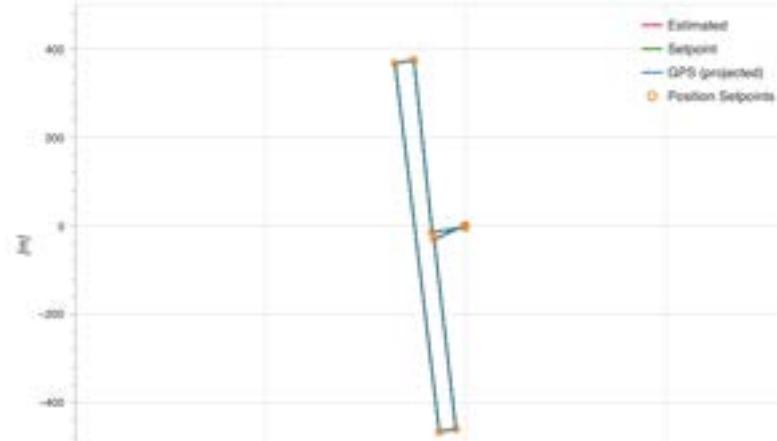
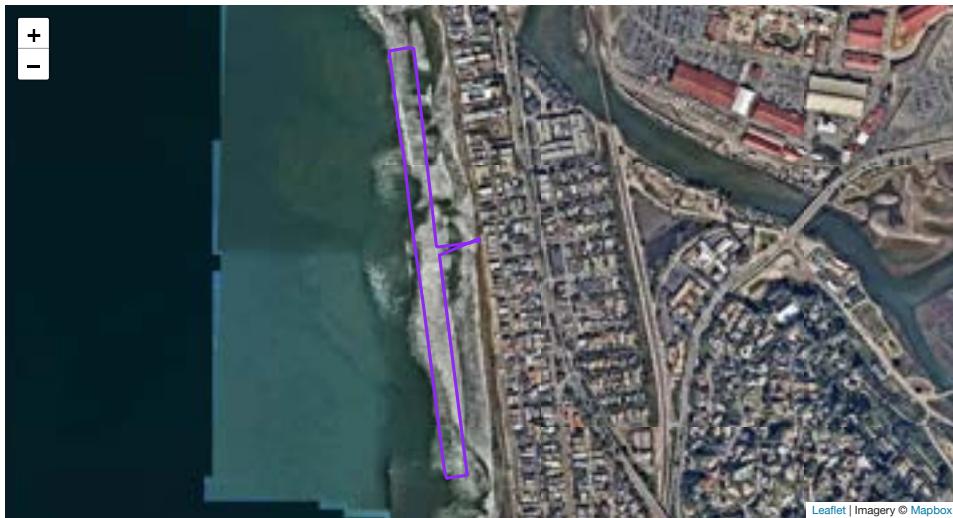
Do you need help with interpreting the plots? See here.

PX4 Quadrotor

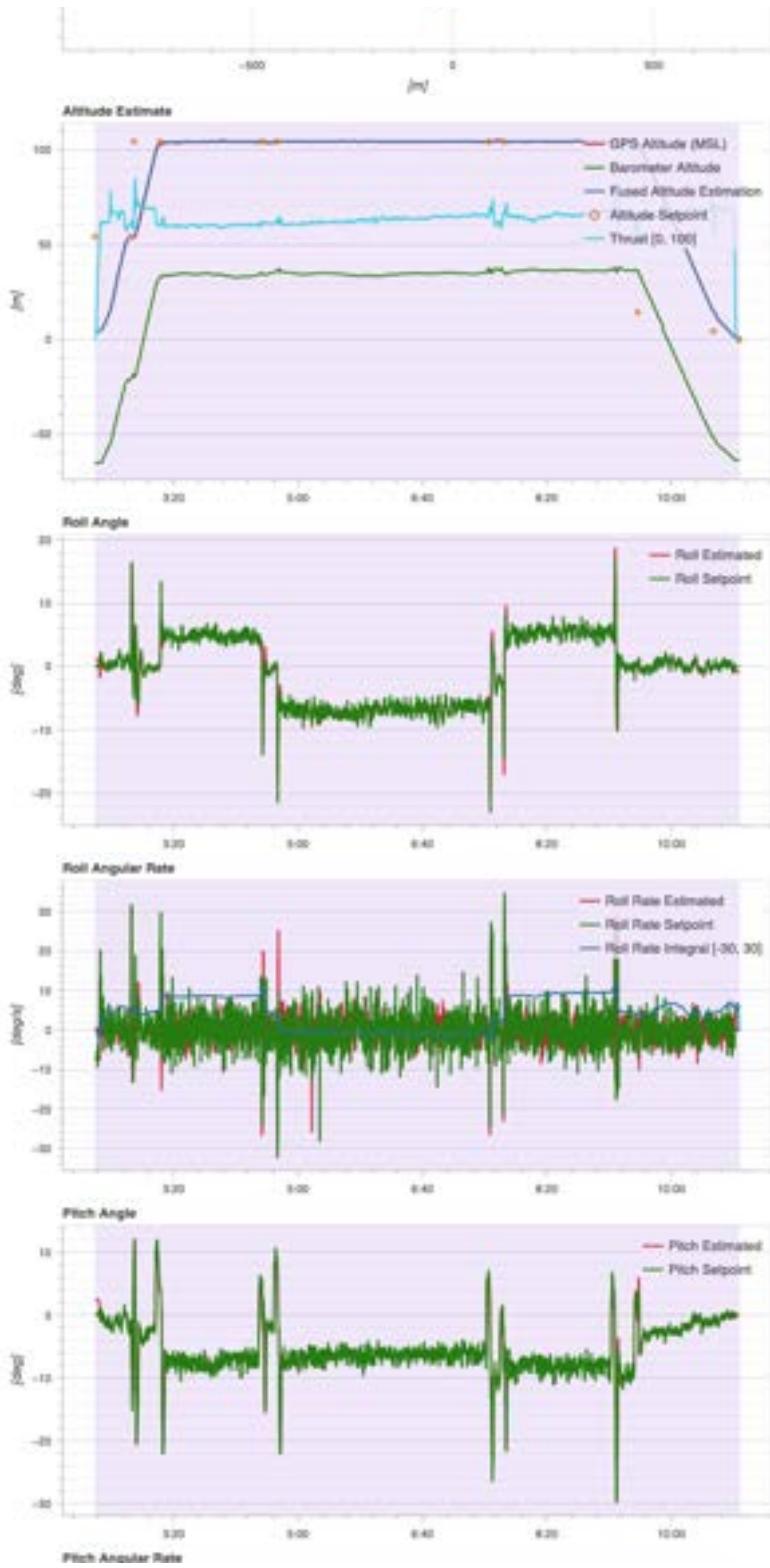
[Open 3D View](#) [Open PID Analysis](#)

Airframe:	S500 Generic Quadrotor X (4014)	Distance:	2.09 km
Hardware:	PX4_FMU_V6X (V6X000003)	Max Altitude Difference:	103 m
Software Version:	v1.14.0 (RC) (5359fe59)	Average Speed:	14.5 km/h
OS Version:	Nuttx, v11.0.0	Max Speed:	20.6 km/h
Estimator:	EKF2	Max Speed Horizontal:	19.1 km/h
Logging Start ?:	22-11-2023 08:19	Max Speed Up:	10.8 km/h
Logging Duration:	0:08:38	Max Speed Down:	5.8 km/h
Vehicle Life	1 hours 16 minutes 37 seconds	Max Tilt Angle:	29.1 deg
Flight Time:			
Vehicle UUID:	000600000003633393831375112000a0004		

Add a detected error...



Scout flight #6 – page 2/8 (pages 3-8 are omitted)



Scout flight #7 – page 1/8

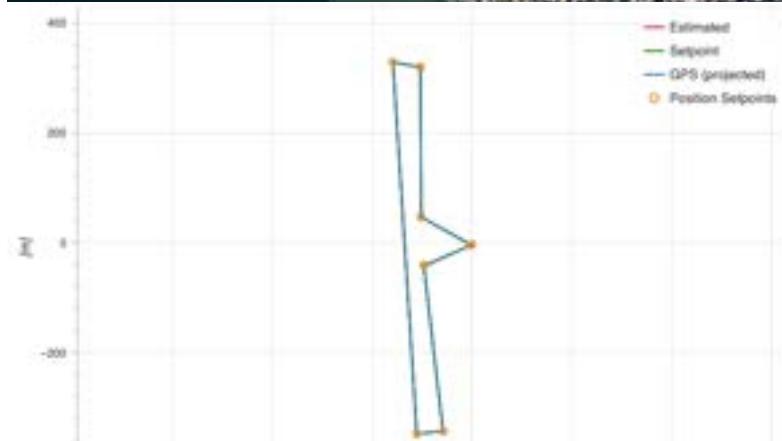
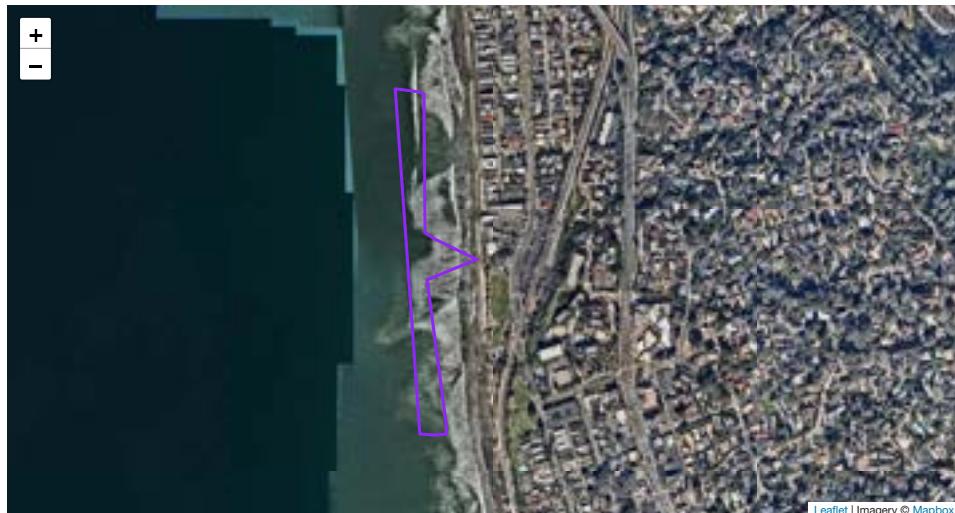
Do you need help with interpreting the plots? See here.

PX4 Quadrotor

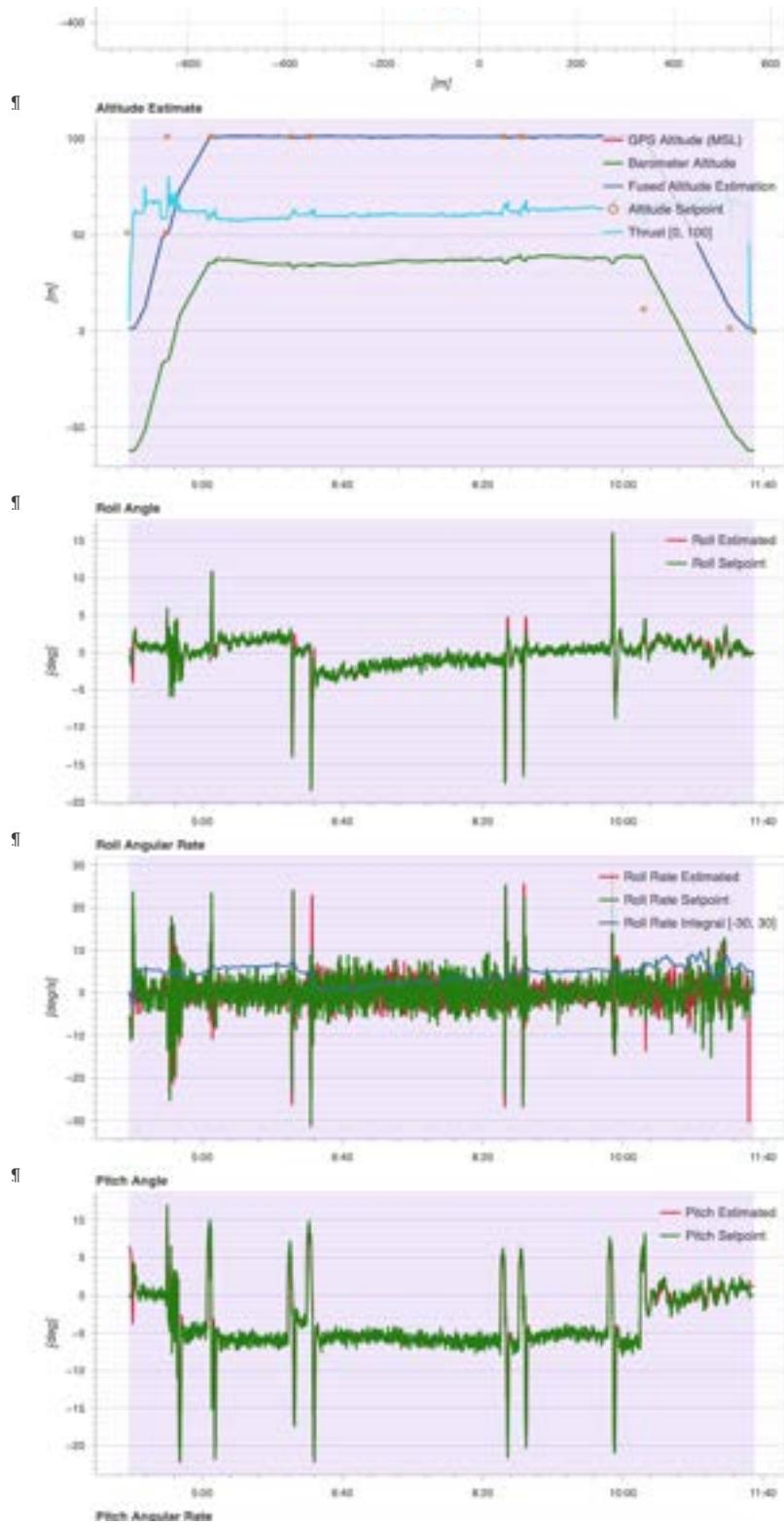
[Open 3D View](#) [Open PID Analysis](#)

Airframe:	S500 Generic Quadrotor X (4014)	Distance:	1.75 km
Hardware:	PX4_FMU_V6X (V6X000003)	Max Altitude Difference:	100 m
Software Version:	v1.14.0 (RC) (5359f659)	Average Speed:	14.0 km/h
OS Version:	Nuttx, v11.0.0	Max Speed:	18.8 km/h
Estimator:	EKF2	Max Speed Horizontal:	18.3 km/h
Logging Start ?:	22-11-2023 09:01	Max Speed Up:	10.9 km/h
Logging Duration:	0:07:27	Max Speed Down:	5.8 km/h
Vehicle Life	1 hours 25 minutes 13 seconds	Max Tilt Angle:	21.1 deg
Flight Time:			
Vehicle UUID:	000600000003633393831375112000a0004		

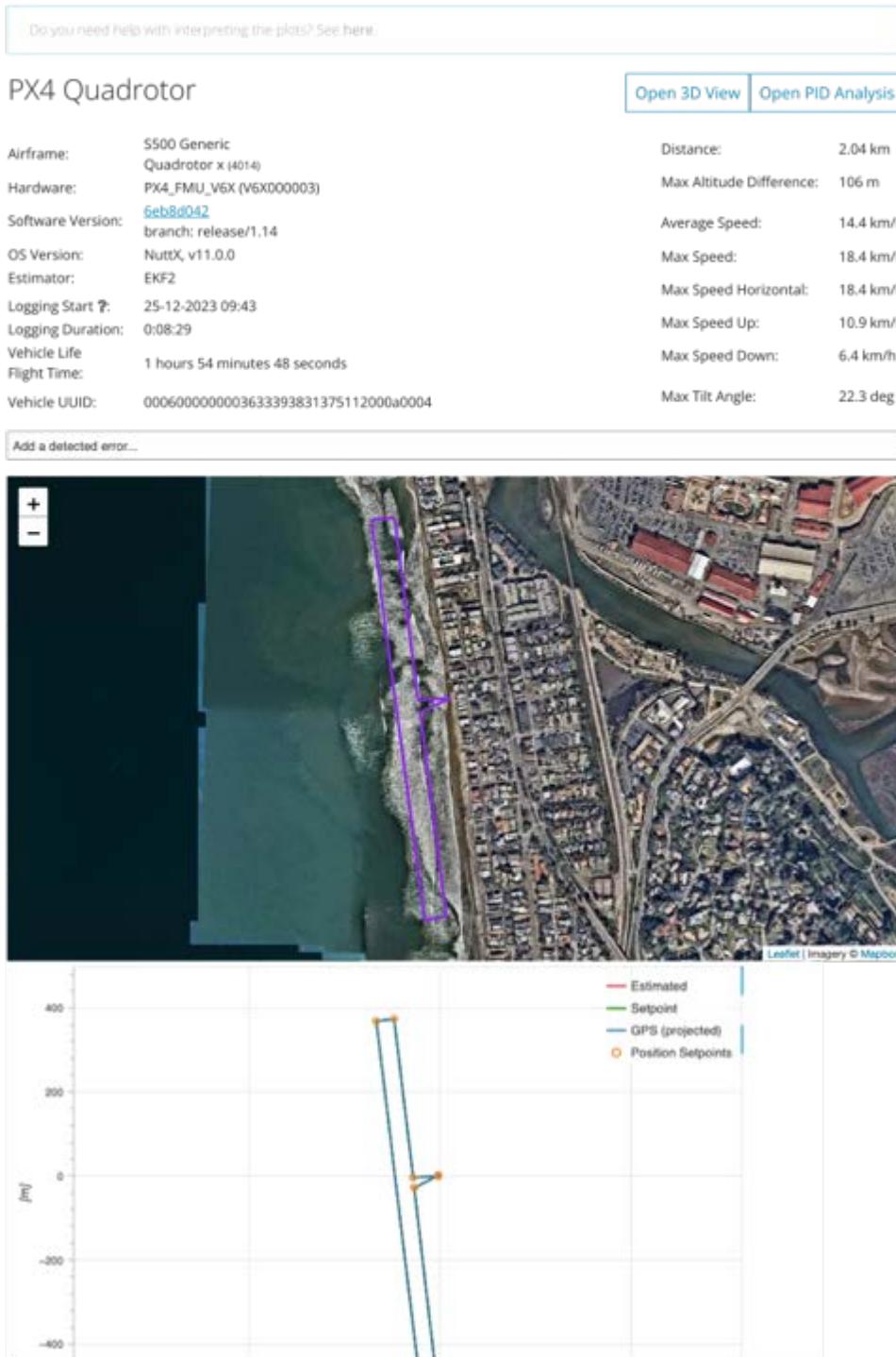
Add a detected error...



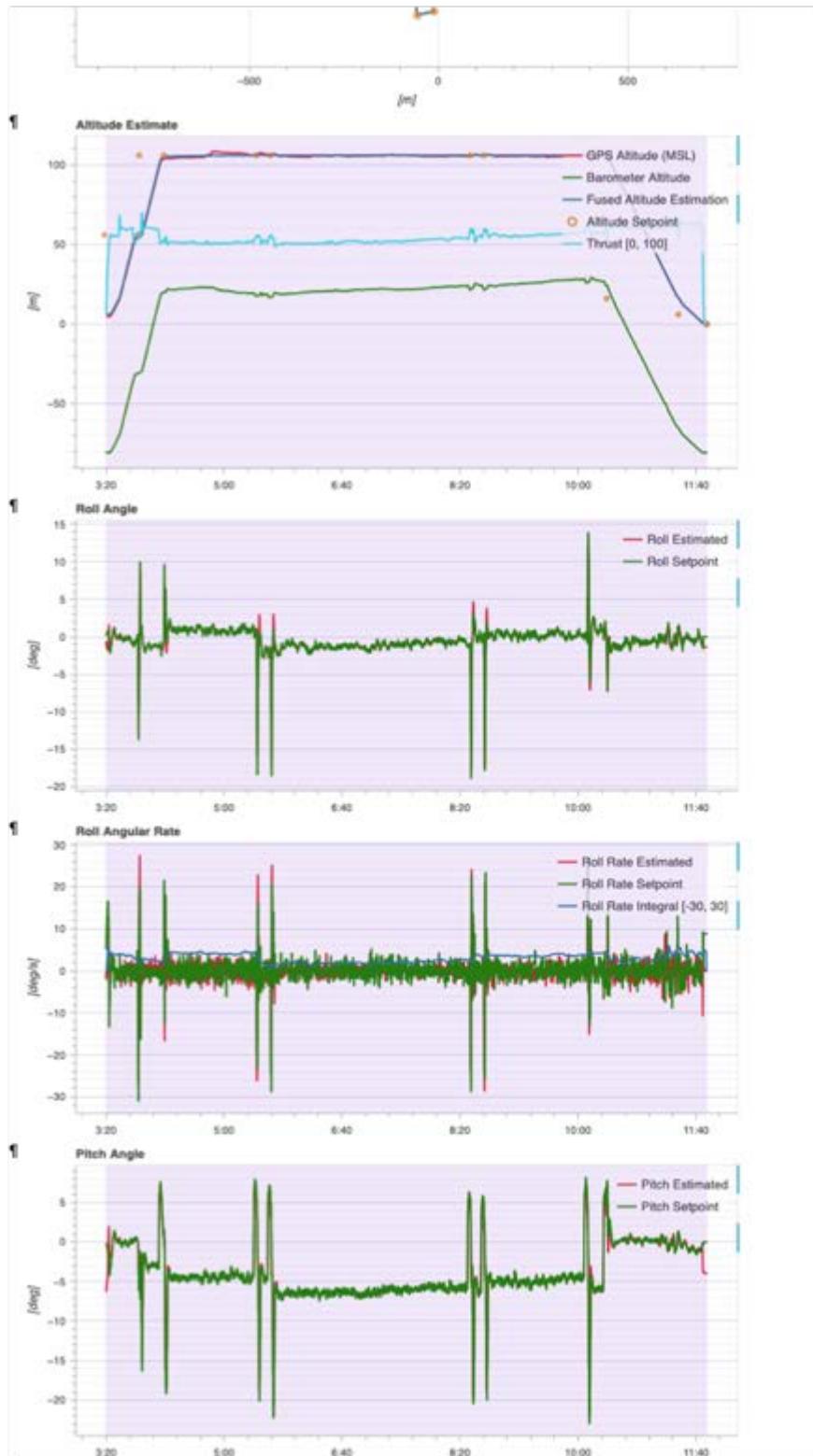
Scout flight #7 – page 2/8 (pages 3-8 are omitted)



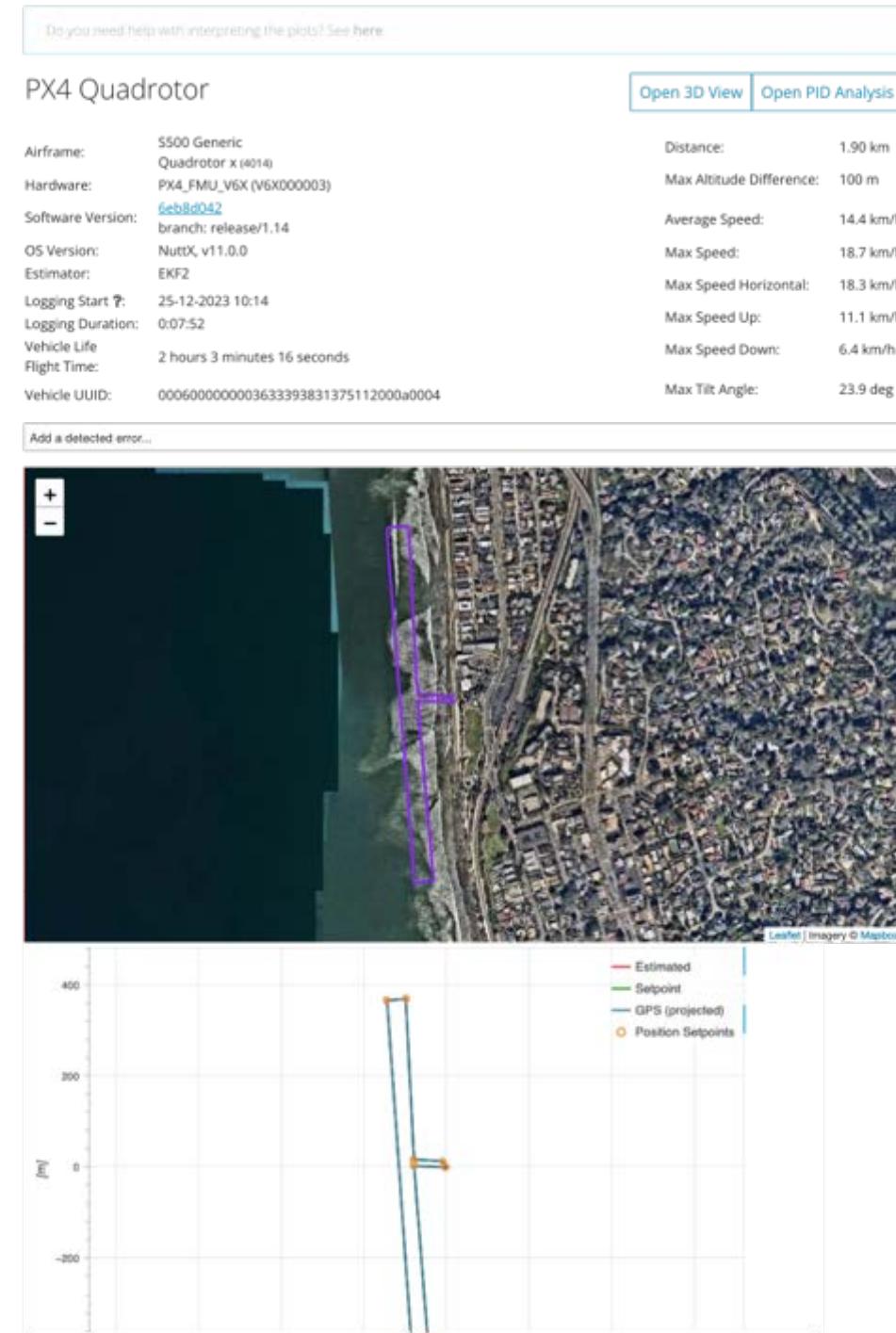
Scout flight #8 – page 1/8



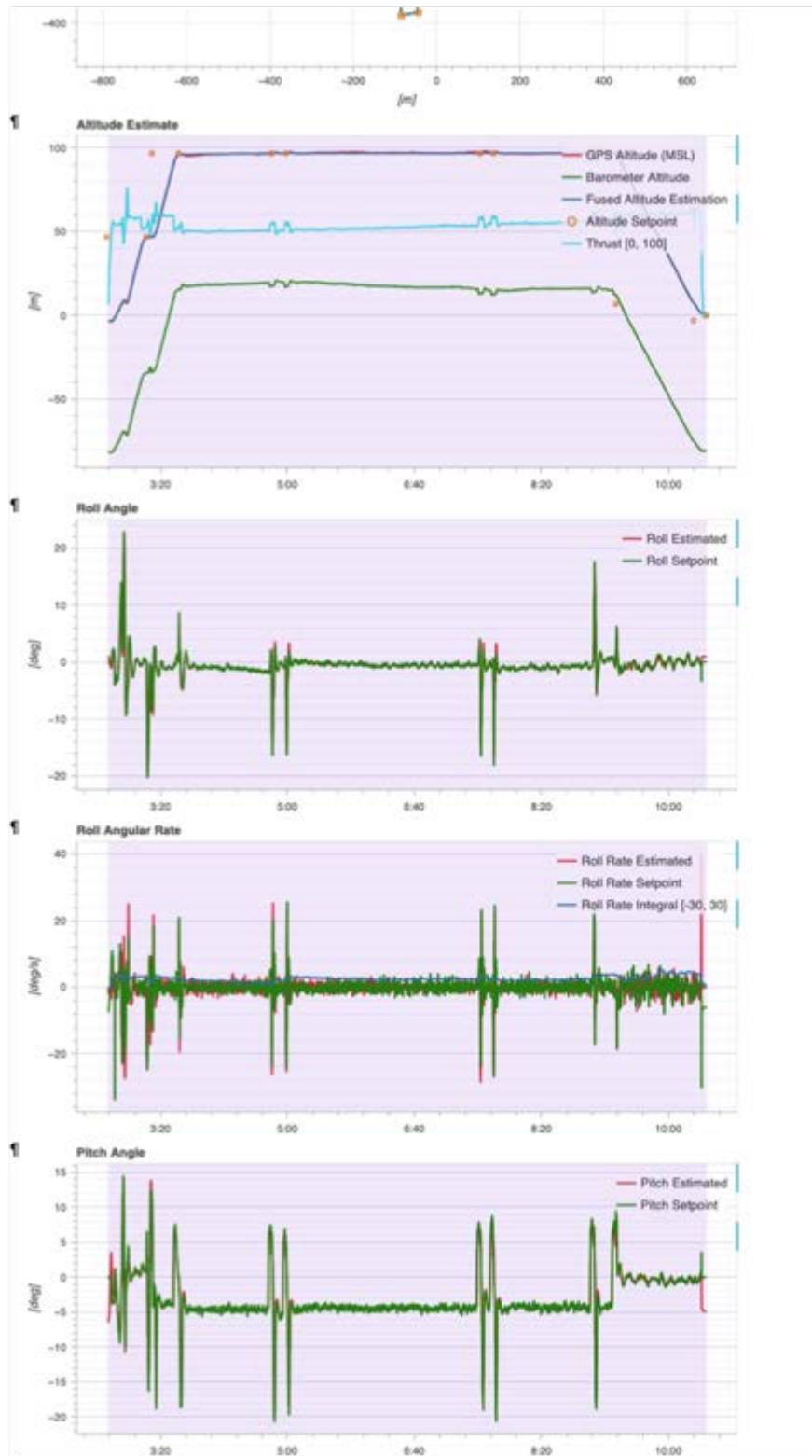
Scout flight #8 – page 2/8 (pages 3-8 are omitted)



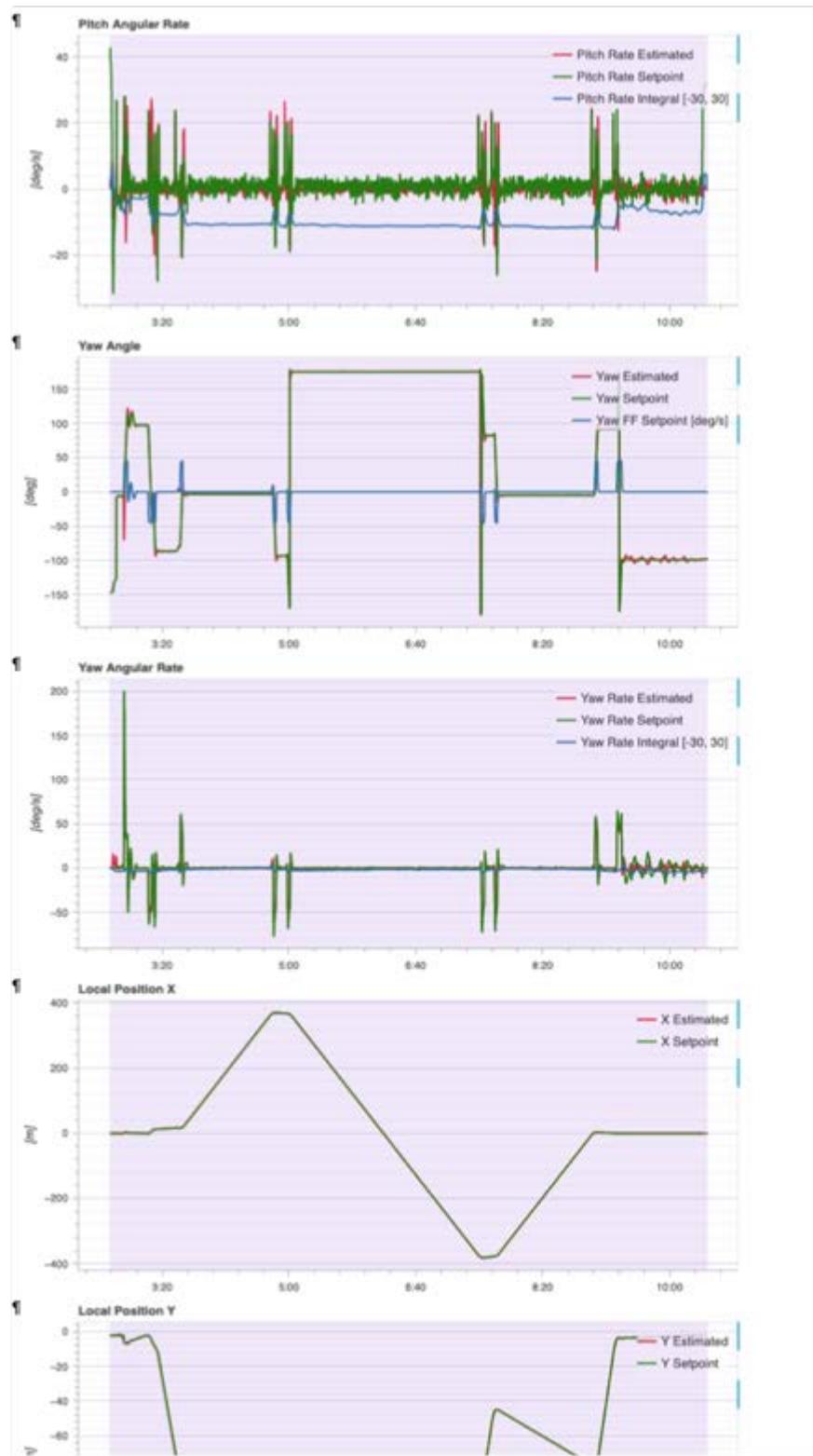
Scout flight #9 – page 1/8



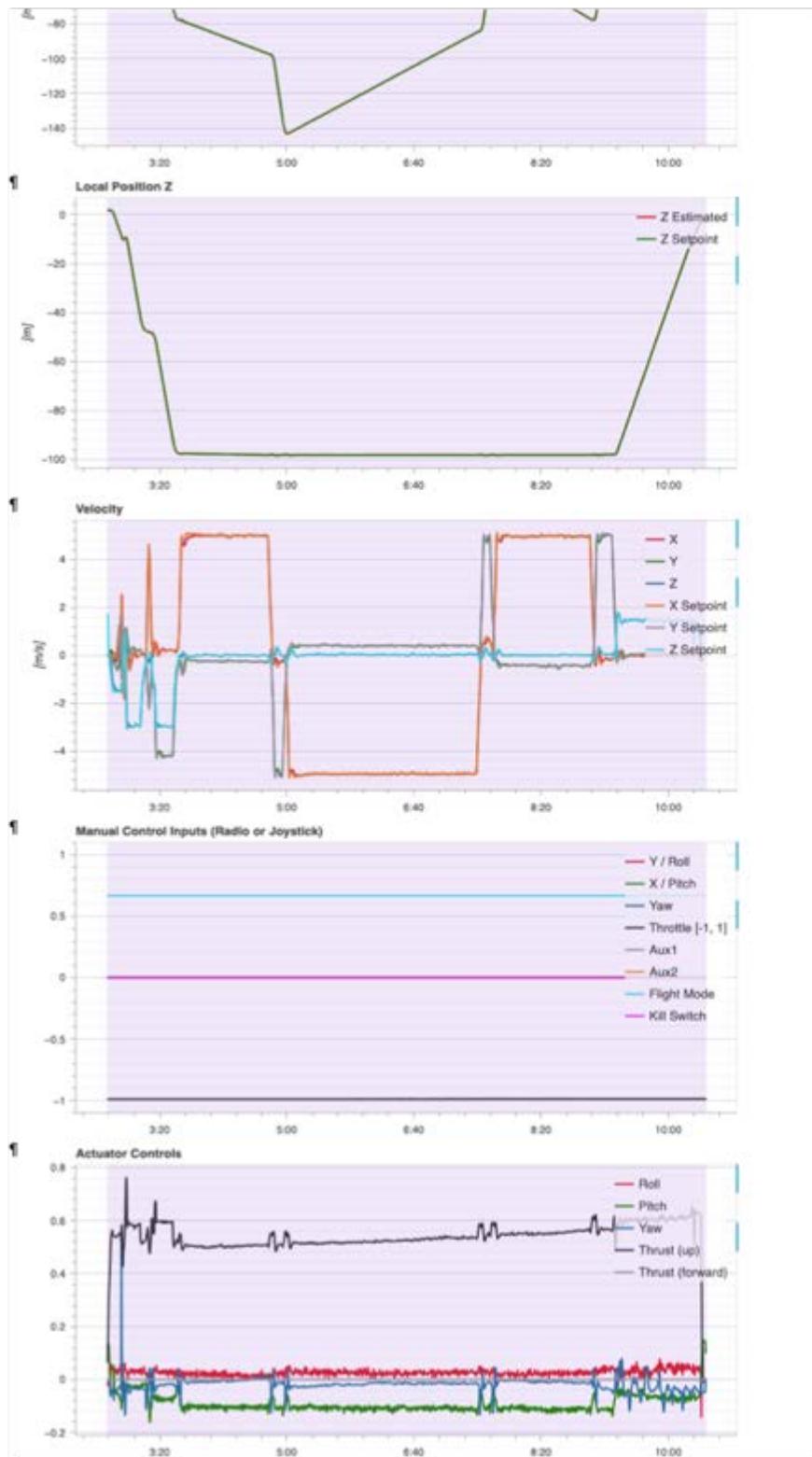
Scout flight #9 – page 2/8



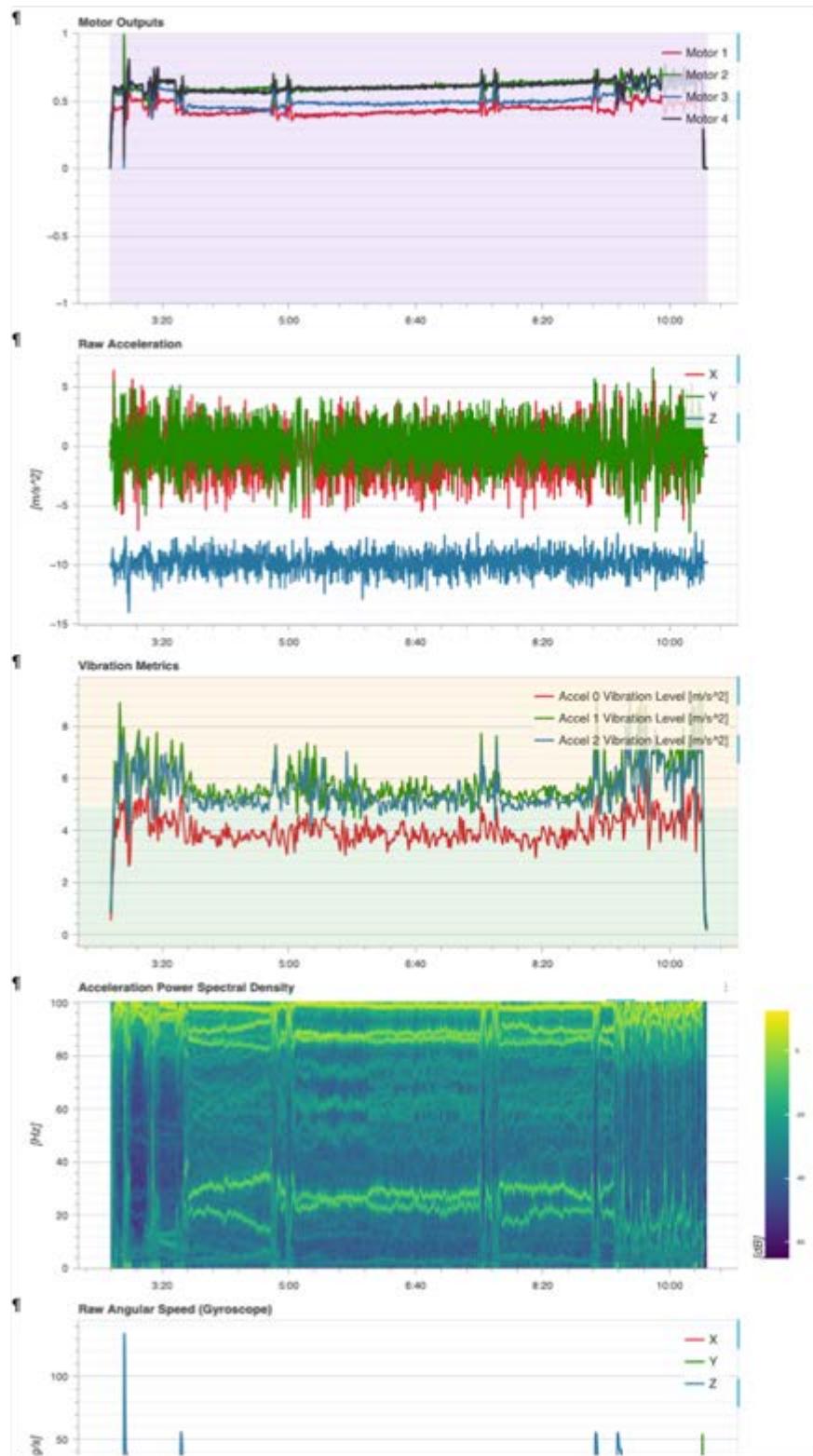
Scout flight #9 – page 3/8



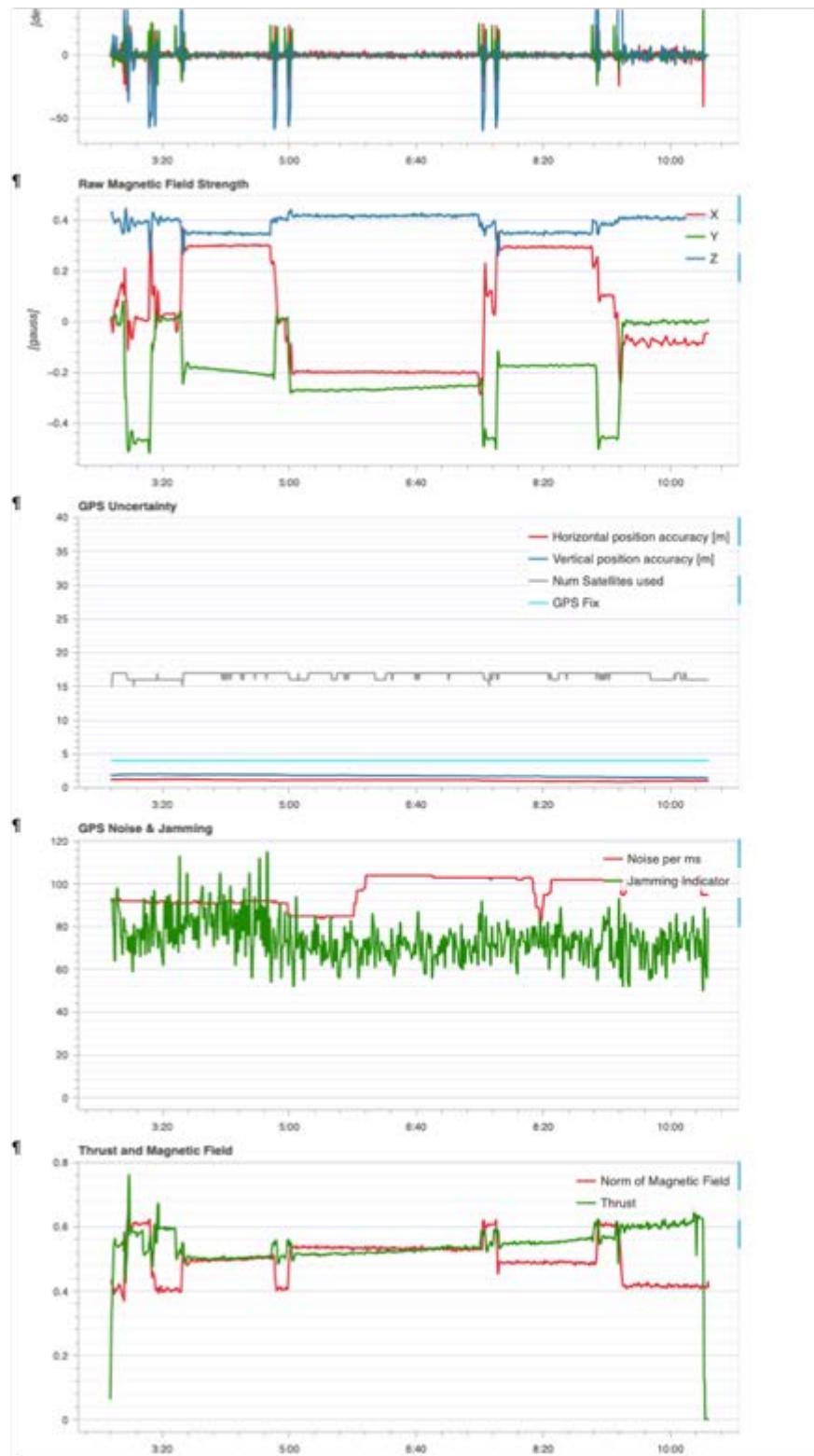
Scout flight #9 – page 4/8



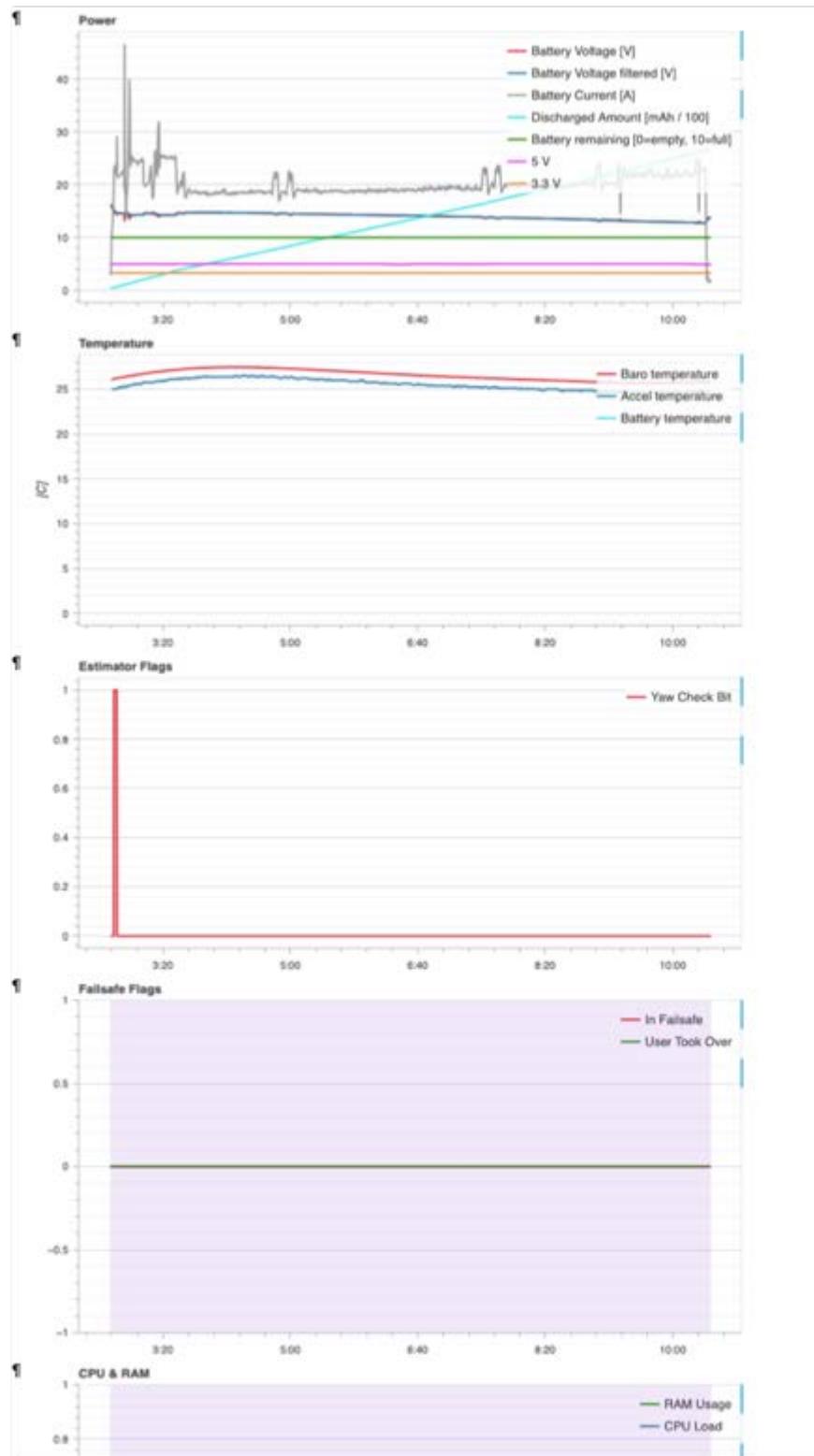
Scout flight #9 – page 5/8



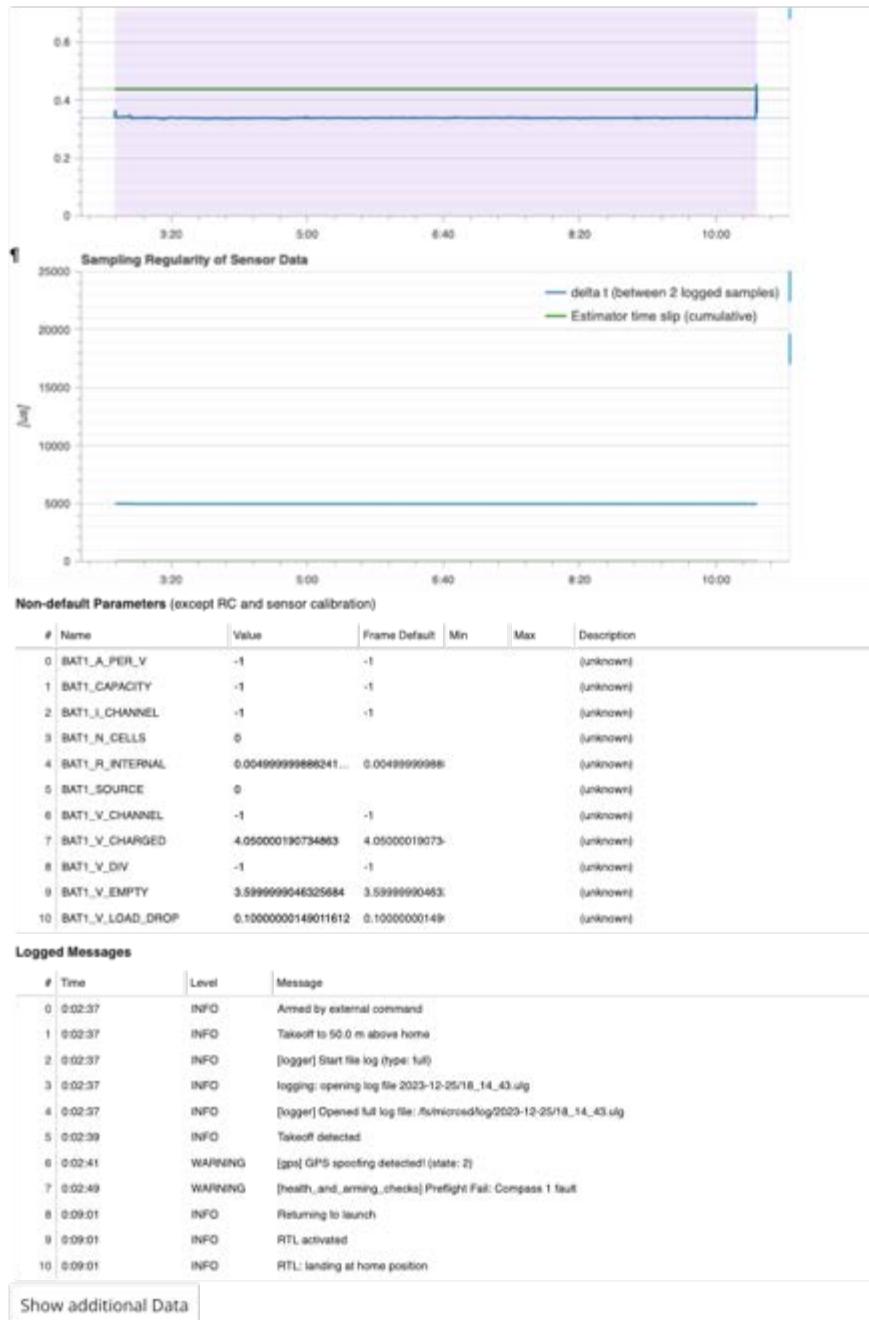
Scout flight #9 – page 6/8



Scout flight #9 – page 7/8



Scout flight #9 – page 8/8



Appendix G. Research progress notes

05/27/2023

Lifeguard drone ideas:

- Shark deterrent -- ultrasonic
- Drops buoy (?)
- Neon colors
- Drops line
 - o Drops it at shore
 - o Drags person
- Identifies...
 - o Rip tides
 - o Sharks
 - o Sting rays (?)
 - o Animals in general
- First aid kit
- Radio transmission to lifeguards & police
- Finds missing kids using facial recognition
- Going underwater
- Different number of propellers
- Solar powered
- Self-charging
- Water bottle
- Snacks
- Navigate caves
 - o Search operation
- Responds to text messages for help
- Leads lifeguard to person in danger
- Lights
- Finds animals too close to shore
 - o Turtles
 - o Dolphins
 - o Etc
- Floatable
- Self-operated
- Manually-operated
- Tide
- Speaker
 - o Lifeguard?
- Wire to beach
- Screen
- Cameras
- 2 drone collab, snapping point
- Sonic depth measurement

05/28/23

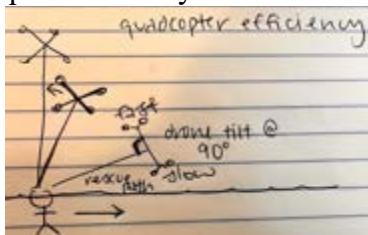
- TRUST (The Recreational UAS Safety Test) Completion Certificate
 - o Authentication Token: RECF29679944364
 - o Issued by: The Robotics Education & Competition Foundation
 - o To fly >.55lbs for fun without monetary intentions
- Prototype development kit: Holybro
 - o X500



○ *photograph taken by the author*

- Using it to practice drone flying, get feel of drone-building, identify drone components & connections, program, and gather data of ocean (rip currents waves, people, sea animals)

- Quadcopter efficiency for rescue efforts



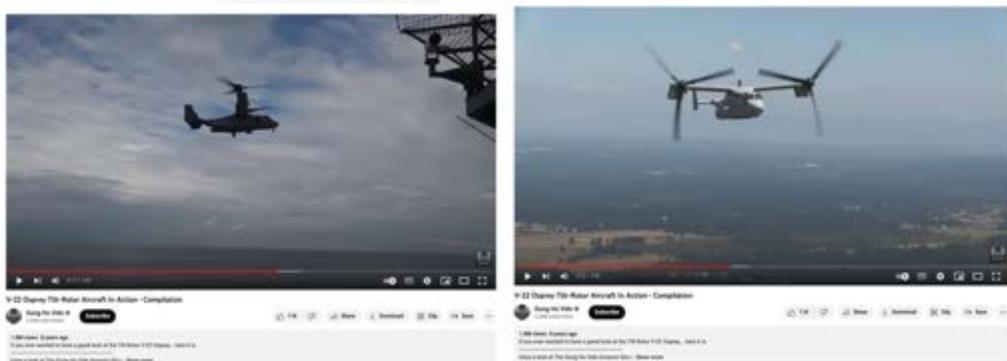
○ *diagram created by the author*

- Need design improvements
 - Strong motors? Different drone design?
 - Maximize drone force—tilt rotor (coaxial), thrust vector

V-22 Osprey Tilt-Rotor Aircraft In Action • Compilation: V-22 Osprey Tilt-Rotor

Aircraft In Action • Compilation

- Tilt rotor example



Images taken from “V-22 Osprey Tilt-Rotor Aircraft In Action • Compilation”
(<https://www.youtube.com/watch?v=xnOw3clRIoI>)

05/31/23

- Surveying flying mechanisms on YouTube
 - o **Thrust vectoring drone**
 - <https://www.youtube.com/watch?v=u2cETOyuJ20>
 - o 3D PRINTED SINGLECOPTER WITH THRUST VECTORING
 - https://www.youtube.com/watch?v=YdvwP_g6c2M
 - Not efficient thrust vectoring compared to F22 thrust vectoring because wind going through yellow drone

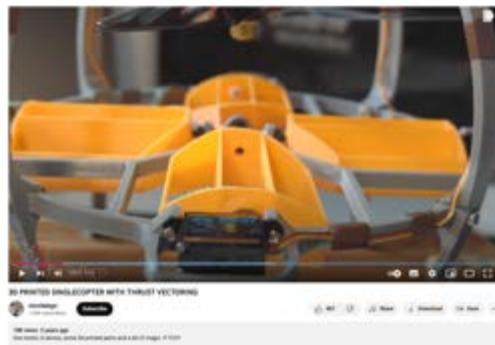


Image taken from “3D PRINTED SINGLECOPTER WITH THRUST VECTORING”
(https://www.youtube.com/watch?v=YdvwP_g6c2M)

- Coaxial = 2 propellers
- Coaxial drone
 - o https://www.youtube.com/watch?v=6FU_HeXyl-Y
 - o Coaxial
 - o Also tilt rotors

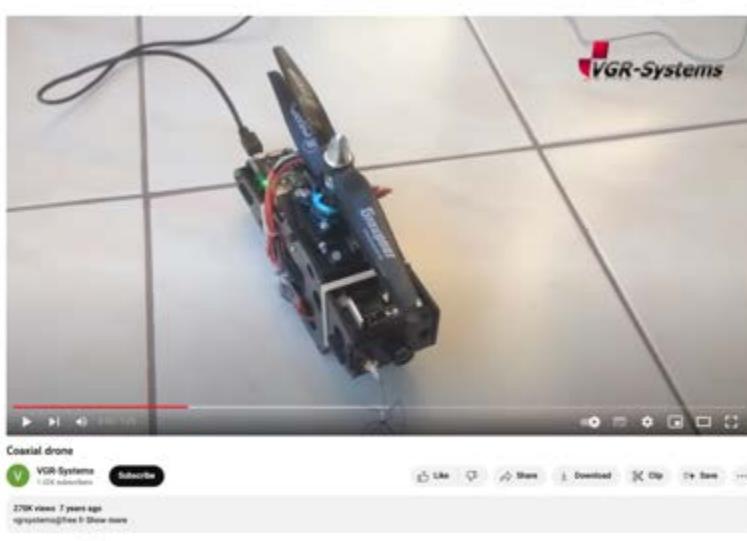


Image taken from “Coaxial drone” (https://www.youtube.com/watch?v=6FU_HeXyI-Y)

- Questions:
 - o Tri/hexa/penta/octacopter benefits/disadvantages?
 - o # of blades in propeller
 - o Thrust vectoring vs tilt rotors --> what's easier/more effective?
 - Tilt rotors = less difficult to program?
 - Which one pulls more? Need calculations
 - With wind, thrust vectors = worse bc wind holes on side
 - Initial guess = tilt rotors
 - Less bulky
 - Less affected by wind
- X500 Drone prototype connection drawing

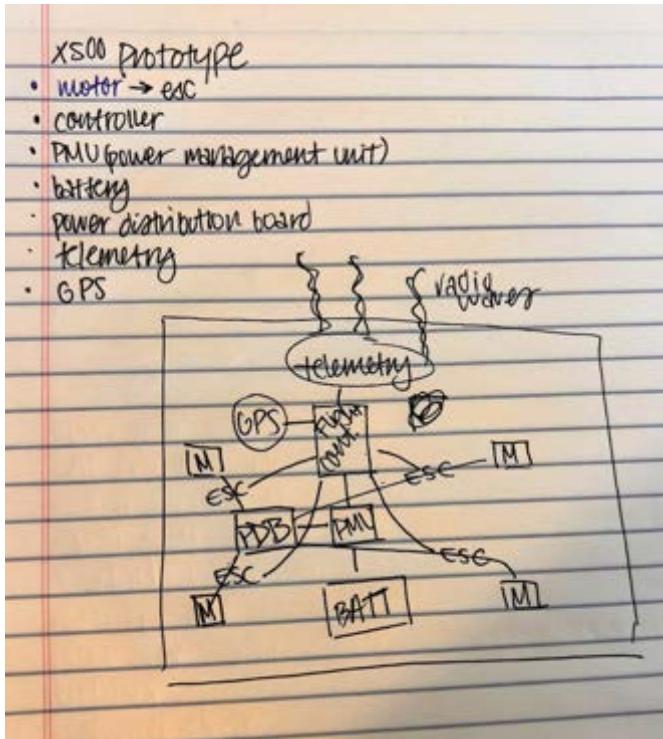


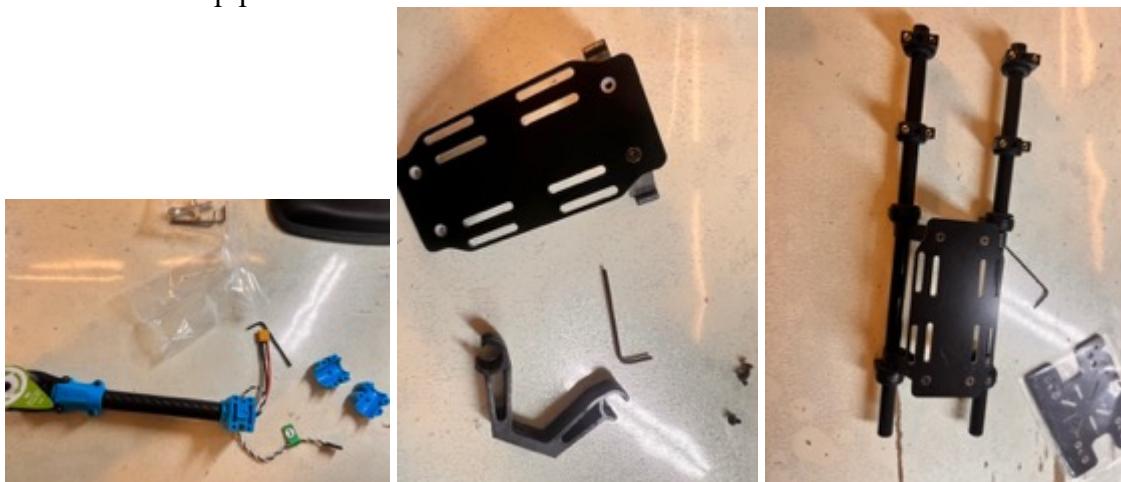
Diagram created and photograph

taken by the author

06/01/23

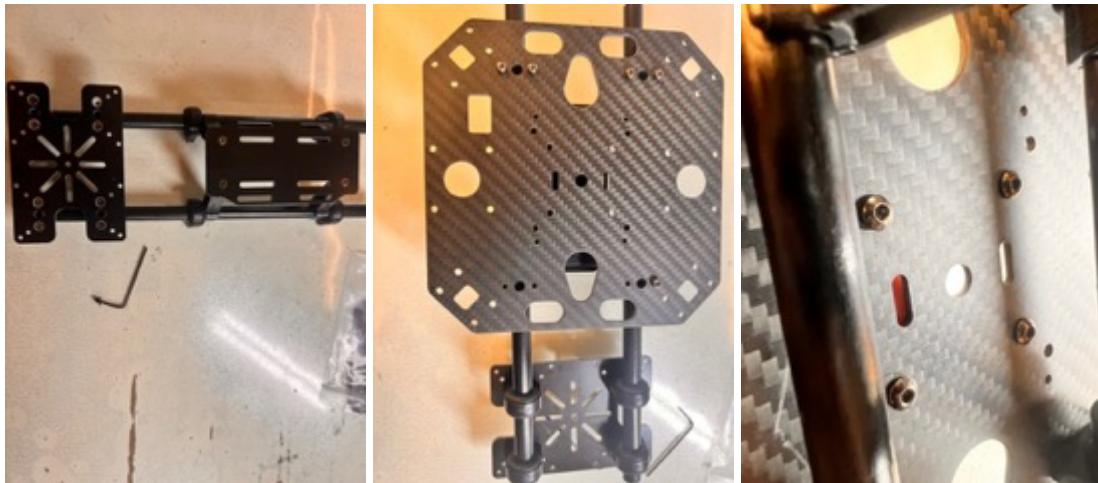
- Finished building battery holder, legs, main bottom plate, power distribution board, one bottom landing gear cross bar

Attempted to begin by building motors but then realized (1) most of it was already done by manufacturer and (2) in order to connect 2 blue pieces, I needed bottom main plate of drone to be fully built. Started to build battery holder. The battery holder plate also connects to the mission controller with pipes.



Photographs taken by the author

Mission controller plate. Bottom plate attached. Screws for power distribution board.



Photographs taken by the author

Power distribution board. There was an error in instruction diagrams I had to fix. Legs.



Photographs taken by the author

One landing gear cross bar attached.



Photograph taken by the author

Tomorrow (or next session) - attach other landing gear cross bar, screw in one hole that wasn't fitting, + everything else.

06/02/23

Screwed in a pesky screw. Most screws in drone are M2.5s or M3s.



Photograph taken by the author

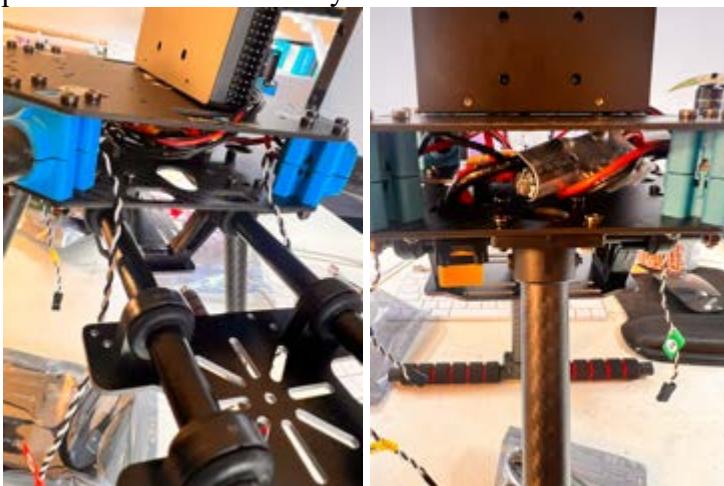
Secured both landing gear cross bars after attaching other one. Attached all motors and corresponding flange locknuts.



Photographs taken by the author

06/03/23

Attached power management unit. Connected motors to power distribution board. Realized flight controller needs nuts on bottom of top plate to be securely connected. Will take off top plate tomorrow to securely attach.



Photographs taken by the

author

06/04/23

<https://docs.holybro.com/autopilot/pixhawk-6x/dimensions> --> dimensions of flight controller

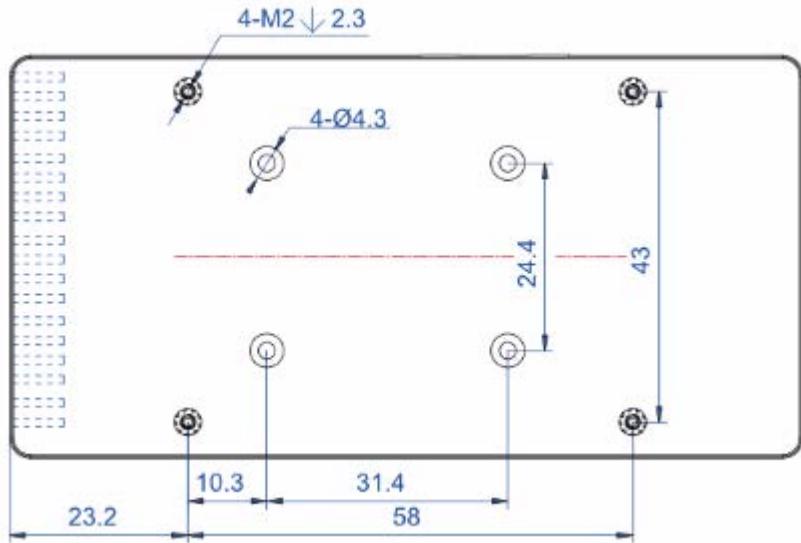


Image taken from

Holybro Docs (<https://docs.holybro.com/autopilot/pixhawk-6x/dimensions>)

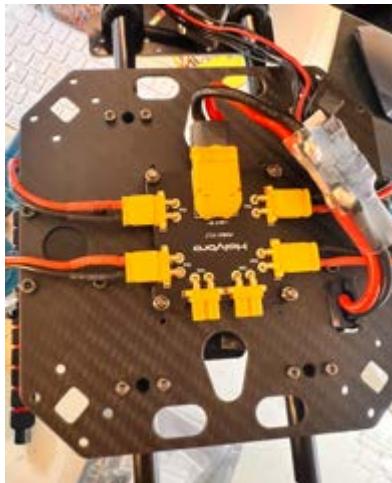
The bottom of flight controller

Took off top plate to connect flight controller to top plate. However wanted to include space between flight controller and plate to avoid damage. Also screws were too long so spacers (an interesting combination of M2.5 nuts, washers, and spacers for M2 screw) were necessary. And avoid jolting for gyroscope.



Photographs taken by the author

Also look I did it! Realized that motors 1 and 2 were switched so swapped them back and tidied up wiring which was a slight mess.



Photograph taken by the author

06/05/23

<https://oscarliang.com/lipo-battery-guide/>

<https://holybro.com/products/px4-development-kit-x500-v2>

Remounted motors and top plate

Mounted M8N GPS.

Attached battery pack pad and straps.



Photographs taken by the author

06/07/23

Connection/wiring diagram:

https://docs.px4.io/main/en/flight_controller/pixhawk6x.html

How to control drone through telemetry: <https://docs.holybro.com/telemetry-radio/sik-telemetry-radio-v3>

Connect one telemetry to px4 flight controller. Another to my computer w QGroundControl and controller.

Connecting telemetry to mission control: <https://ardupilot.org/planner/docs/mission-planner-installation.html>

Attached telemetry. Wired motors, GPS, & telemetry to flight control. Attached propellers.



Photographs taken by the author

06/08/23

ISDT K4 LiPo Charge/Discharge Cycle Mode Charger -

<https://isdtshop.com/products/isdt-k4>

- DC 600Wx2, 30A max current so can charge 8 batteries at once (4 on each side)
 - Charged batteries with 5A (max current for battery = 9A)
- Choosing the Best LiPo Battery Charger - <https://oscarliang.com/choose-lipo-battery-charger-power-supply/>

Operation confirmed when battery plugged in!!! AYYAYAYAYAYAYAYAYAY



Photograph taken by the author

But need XT60 cables because battery cable doesn't reach power connector cable

GPS takes 26 seconds to establish coordinates

Compilation & upload only work with Linux. Connection established with QGroundControl (just downloaded) and drone. Telemetry radios luckily connected with each other automatically and to QGroundControl also.



Photograph taken by the author

Drone works! Which I found by turning it on (propellers and all) in my room by telling it to takeoff. Not a great idea but it works and that's what matters! Saw my life flash before my eyes

06/11/23

What the drone should do:

- Can identify a person in danger
- Can alert lifeguards if someone is in danger
- Can drag someone out of danger

Deliverables from this project:

- Patent
- SW: Image processing at the beach (shark, person (not) in danger)
- ME: Drone can save person
 - o Prototype no work bc not powerful enough
 - o Design needs to be changed

Definitions:

- Drone 1: prototype
 - o To film and gain experience building/flying/knowing components
 - o Understand telemetry & sw stuff
- Drone 2: main vehicle
- Patent
- Will capture anything and everything and literally anythinggggg I can possibly implement
- But Drone 2 = only a few aspects bc limitations
- SW idea I want to implement
 - o Finding rip current/people/sharks
- ME idea I want to implement
 - o Rope/drop something/pull someone --> tilt rotor

Prototype:

- Cinderkit
- Almost ready to fly
- Camera? How to collect the data? And images and videos?
- Beach data collection plan BEFORE camera
 - o Need the whole system
 - o HAM Radio
 - o What data collection :skull:

- Manually? QGroundControl does it automatically w GPS?
- Heavy drone
 - So must register for FAA bc regulations
 - Need to have drone ID when Sep capitalism
- Know the area~~~
 - Where are the regulations
 - Where are the rip currents
- Tidal waves
 - When does it have more rip current
 - When is the best time for rip current

Start doing patent. Write down ideas & be more in-depth for the patent. Fly drone but not inside. Register for regulations. First do SW data collection and then patent.

Rip Current Science



Image taken from "Rip Current Science"
(<https://www.youtube.com/watch?v=RJ4hcaJ9ITY>)

Low tide

They often form at sand bars near shore

Water flows back into ocean thru deeper channels and away from shore

How to Spot a Rip Current



Image taken from “How to Spot a Rip Current”
(https://www.youtube.com/watch?v=PuAlDTC_gIQ)

Deeper, dark-colored water

Fewer breaking waves

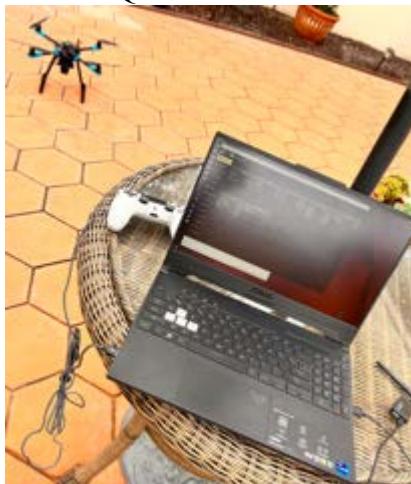
Rippled surface surrounded by smooth waters

Anything (eg flowy discolored sandy water) flowing out beyond the waves

06/12/23

Could not connect to the RC controller with the receiver

(https://holybro.com/products/radiomaster-r81-receiver?_pos=1&_sid=13ed26994&_ss=r)
So used QGroundControl and a PS4 controller joystick.



Photograph taken by the author

I realized that the target flight height on QGroundControl (2.5m) was too high and the drone started to cut through the palm trees



Photograph taken by the author

Then I tried to adjust height on QGroundControl but it still remained the same.

It was a windy day and the drone reaction to PS4 command was late. Latency bad. I could not have responded to wind well. :(

So drone started to go near house, hit house, and propellers damaged. Later, when testing to see if motors functioned, drone gave error message. And then I tested it again, no error message but some more damage.

Damage is shown below.



Photograph taken by the author

I took off one of the propellers. Learned not to do anything with motors after it crashes/break



Photographs taken by the author

Two screws not secured enough. Design issue caused motor to fall out.



Image taken by the author

QGroundControl said drone went this far during testing of motors to see if they still functioned after crash...when in reality I was holding it down at the backyard of my house. GPS house. GPS accuracy until satellites are established might not be good. Or, we need more accurate or secondary GPS.

The actuator should not be powered when the propeller or actuator fixture is damaged. Mechanical damage will be further progress if powered up.

Need wide open space to practice drone flight.

One leg landing foot is not sturdy when the drone crash lands. The end cap of the landing foot was off and the landing cushion was taken out. Need to glue it tightly.

PID's were not tuned at all.

To do next time

- PID tuning and battery calibration necessary
- Use flight planning with GPS
- Measure vertical and slanted thrust

Why receiver failed:

- Receiver link: <https://www.radiomasterrc.com/products/r81-receiver>
 - Connection between receiver and RC controller failed

- It's working tho
 - Signal format: Frsky D8 Compatible
- RC controller link: <https://www.radiomasterrc.com/products/tx16s-mark-ii-radio-controller>

- Singal format: ELRS (different from Frsky)
- ELRS vs Frsky no match
- ELRS is more advanced but Pixhawk 6X (flight controller) doesn't accept it so that's not good
- Must change RC controller to 4in1 type through hardware changes

6/14/23

JETSON Nano setup

Image download:

[https://developer.nvidia.com/embedded/downloads#?search=jetson%20nano&tx=\\$product,jetson_nano:~:text=Nano%20Developer%20Kit%20SD%20Card-,Image,-4.6.1](https://developer.nvidia.com/embedded/downloads#?search=jetson%20nano&tx=$product,jetson_nano:~:text=Nano%20Developer%20Kit%20SD%20Card-,Image,-4.6.1)



Photographs taken by the author

06/15/23

- JETSON needs to communicate with the computer using network because it attaches to the bottom of drone
- PCI Express wifi connecter intel --> but antenna is too big and HUGE bracket



Photograph taken by the author

- Using lightweight antenna now
 - Not using ethernet, wifi instead bc idt attaching a monitor, keyboard, & mouse on the beach would be fun
 - Remote login and use wifi
 - To do: get Jetson nano & Orin up to speed
 - Orin power supply input voltage: 5-19V. So can connect to 4S battery
 - Jetson Nano power supply input voltage: 5V, less flexible. Must connect to flight controller but then risky current budget
- Jetson Nano Current budget:

	Voltage (V)	Typ current (A)	Max current (A)	Typ power (W)	Max power (W)
Pixhawk 6X					
Telemetry	5		0.125		0.625
M8N GPS	5		0.15		0.75
R81 RC receiver					
JETSON Nano			2	5	10
PM02 V3 Power Module	5.2		3		15.6

Table created by the author

Raspberry vs Nano vs Orin performance comparison

	Raspberry Pi 4B	Jetson Nano	Jetson Orin Nano
CPU	Quad core Cortex-A72 (ARM v8) 64-bit SoC	Quad-Core Arm Cortex-A57 MPCore processor	6-core Arm Cortex-A78AE v8.2 64-bit CPU
Clock (GHz)	1.8	1.43	1.5
GPU	Broadcom VideoCore VI	128-core NVIDIA Maxwell™ architecture GPU	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores
GPU clock (MHz)		921	625
Memory	8GB LPDDR4-3200 SDRAM	4GB 64-bit LPDDR4 25.6GB/s	8GB 128-bit LPDDR5 68 GB/s
Power max (W)	7.6	10	15
Power typ (W)	3.4	5	7
Benchmark			
TFLOPS	0.014	0.236	1.28
Geekbench single core	259	230	566
Geekbench multi-core	681	819	2974
AI performance (TOPS)		0.472	20
SpecInt rate		16	106
Linpack double	10.7	912.26	
Linpack single	20		
Video encoding		2x 1080p60 1x 4K30	3-4x 1080p30
			1x 4K60 (H.265) 2x 4K30 (H.265)
Video decoding		1x 4K60 (H.265) 4x 1080p60 (H.265)	5x 1080p60 (H.265) 11x 1080p30 (H.265)

Table created by the author

- In terms of TFLOPS, Nano & Orin are >>>>
06/19/23
- Replacement parts (specifically motor connections) have arrived!! But no good screws (there are screws, but not the right ones)



Photograph taken by the author

- So ordered new stainless steel hex socket screws
 - o M3s: https://www.amazon.com/dp/B07VNDFYNQ?psc=1&ref=ppx_yo2ov_dt_b_product_details
 - o M2.5s: https://www.amazon.com/dp/B082XPZV1V?psc=1&ref=ppx_yo2ov_dt_b_product_details

Mounting plate on other side to put camera & Nvidia on it

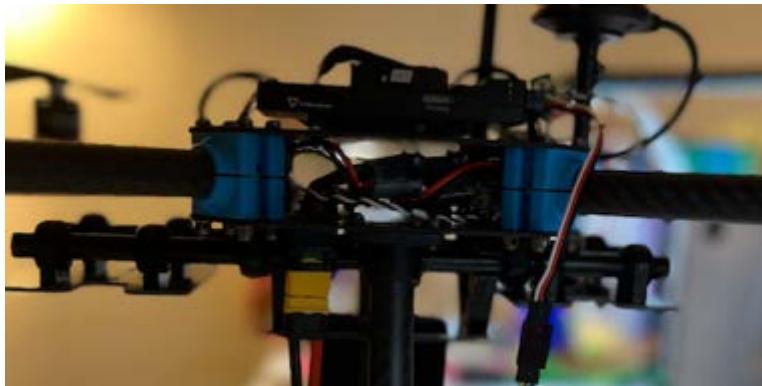


Photograph taken by the author



Photograph taken by the author

A photo of both plates attached



Photograph taken by the author

author

Landing gear reinforcement also added



Photograph taken by the author

Detached antenna cable from big SMA antennae



Photographs taken by the author

Attached lightweight antennae



Photographs taken by the author

Inserted M2 WiFi card & antennae into Nvidia Jetson Nano



Photograph taken by the author

Issue: Latest M2 WiFi card (Intel AX200) not compatible, requires higher version of Linux kernel than Ubuntu 18.04 (Jetson)

Ordered new WiFi card (8265NGW): arriving on weds

https://www.amazon.com/gp/product/B01MZA1AB2/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

06/20/2023

Patent idea development:

- Shark detection & deterrent
 - Identify shark
 - Identify a person near the shark within radius
 - Magnet—electric fields deter sharks
 - Ultrasonic devices—emit high-frequency sound waves to deter sharks
 - Drop ultrasonic probe from string
 - If they don't work, notify
- Drops
 - Rescue tube
 - Life jacket (like the ones on planes) for smaller drones
 - Self-inflatable when pull the tab
 - Dropped when see person in rip current (for >2 min?)
 - First aid kit
 - Water bottle
 - Food (protein bars)

- Neon colors: yellow, orange, pink, red, darker ones too?
 - o For lifeguards to easily see in the sky?
- Drops rope/rescue can/inflatable device with long rope attached to end & ...
 - o Drags person back to shore bc still holding other end of rope
 - o Brings rope to shore for people
 - to pull the person out
 - for lifeguard to follow rope & find them
 - o Rescue can/inflatable > rope bc don't need to pull person upwards bc rescue can floats itself!
- SW: Identifies...
 - o Need camera: probably two (one at front, one pointing downwards)
 - o Rip tides
 - o Animals
 - Sharks
 - Sting rays (?)
 - o PEOPLEEE
 - o Lifeguards
 - Red (clothes vs blood)
- Radio transmission to lifeguards & police
- Finds missing kids using facial recognition
- Going underwater
- Different number of propellers
- Solar powered
- Self-charging
- Navigate caves
 - o Search operation
- Responds to text messages for help
- Leads lifeguard to person in danger
- Lights
- Finds animals too close to shore
 - o Turtles
 - o Dolphins
 - o Etc
- Floatable
- Self-operated
- Manually-operated
- Tide
- Speaker
 - o Lifeguard?
- Wire to beach
- Screen
- Cameras
- 2 drone collab, snapping point
- Sonic depth measurement
- More ideas on 06/23/23

06/22/23

Ordered high temp masking tape. Katpon tape: Wire & cable wrap + protect electrical components during soldering. Polymide = insulator. Does not conduct electricity but good thermal conduction.

Tape on the motor wire



Photographs taken by the author

Added the bottom part of the motor that protected the naked wire

Moved GPS. Used to be at the back of the drone. But now it's on the side. Going to add another GPS for more accuracy later. Also it's good that there's distance between GPS and telemetry now. Because less RF inference. GPS = low-noise amplifier, needs to be away from other RF signal. No contaminate receiver



Photographs taken by the author

YAYYYYYY THE DRONE HAS BEEN FIXED. Time to go fly again!

But before...to do list with drone:

- Flight training at tennis courts
- Need to attach camera onto drone. Both cameras.
- Figure out how to do a planned flight on QGroundControl. Autonomous, scan beach, come back to original location.
 - o Can QGroundControl even do it in the first place?

- Study up!
- Mechanical: start designing the tilt rotor plan
 - o How to implement on drone???????
- Mechanical thoughts on tilt rotors:
- Servo vs DC:
 - o Servo allows for precise control (eg angular position, velocity, acceleration) (eg ZOSKAY 35kg high Torque Coreless Motor servo Metal Gear Digital and Stainless Steel Gear servo arduino servo for Robotic DIY,RC car (Control Angle 270°))
 - o Whereas DC motor is forward/backward motion only
- Motor encoder: provide info on DC electric shaft velocity/position
- Going to use DC because it's not bulky and fits within the arm rod smoothly (servo is rectangular)
- Materials: going to use carbon fiber tubes (eg FANCYWING 500mm (19.6 inches) 8mm x 10mm x 500mm Carbon Fiber Tubes Matte Surface 3K Roll Wrapped 100% Pure for Quadcopter Multicopter (2PCS))
 - o Tube 4 armz
 - o Fit something inside that tube and design along it. So servo motor not going to be easy
- DC Motor specification example: Greartisan DC 12V 1000RPM N20 High Torque Speed Reduction Motor with Metal Gearbox Motor for DIY RC Toys
 - o Rated Voltage: DC 12V
 - o Reduction Ratio: 1:30
 - o No-Load Speed: 1000RPM
 - o Rated Torque: 0.3Kg.cm
 - o Rated Current: 200mA
 - o D Shaped Output Shaft Size: 2.5*9.2mm (0.098" x 0.36") (D*L)
 - o Gearbox Size: 9 x 12 x 10mm (0.35 x 0.47 x 0.39inch) (L*W*H)
 - o Motor Size: 15.2 x 12 x 10mm (0.598 x 0.47 x 0.39) (L*W*H)
 - o Net weight: 9g
- Torque should be strong enough to overcome the resistance/weight of the drone

$$\tau = rF \sin \theta$$

Defintion of torque:

Notes:

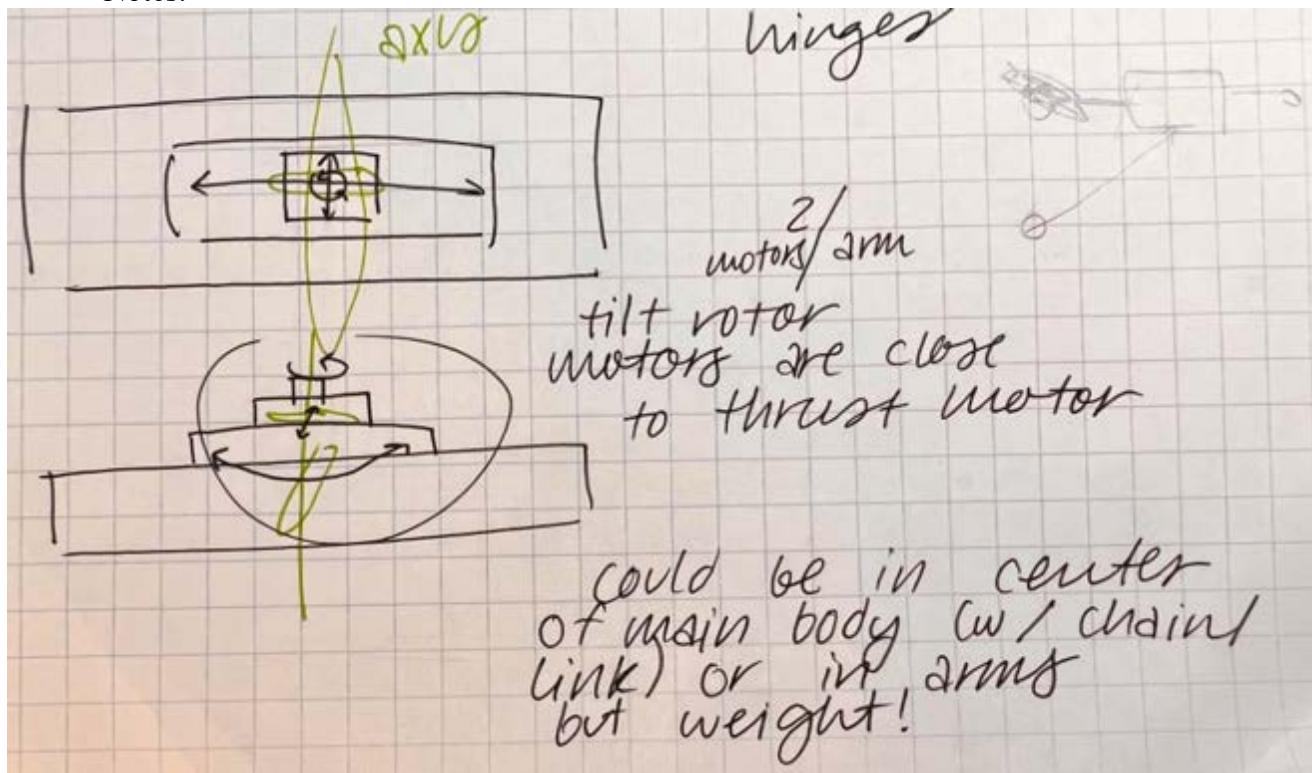


Diagram created and photograph taken by the author

06/23/23

HM30 – need air unit ethernet cable

- Shark detection & deterrent
 - o Identify shark
 - o Identify a person near the shark within radius
 - o Magnet—electric fields deter sharks
 - o Ultrasonic devices—emit high-frequency sound waves to deter sharks
 - Drop ultrasonic probe from string
 - o If they don't work, notify
- Drops
 - o Rescue tube
 - o Life jacket (like the ones on planes) for smaller drones
 - Self-inflatable when pull the tab
 - o Dropped when see person in rip current (for >2 min?)
 - o If drone goes into caves & someone trapped:
 - First aid kit
 - Water bottle (but kinda heavy)
 - Food (protein bars)
- Aesthetics
 - o Lights during the night and maybe during cave searches also
 - o Neon colors: yellow, orange, pink, red, darker ones too?
 - For lifeguards to easily see in the sky?
- Drops rope/rescue can/inflatable device with long rope attached to end & ...

- Drags person back to shore bc still holding other end of rope
- Brings rope to shore for people
 - to pull the person out
 - for lifeguard to follow rope & find them
- Rescue can/inflatable > rope bc don't need to pull person upwards bc rescue can floats itself! I love rescue cans
- SW: Identifies...
 - Need camera: probably two (one at front, one pointing downwards)
 - Rip tides-with given function
 - Animals
 - Sharks
 - Sting rays (?)
 - Seals
 - People
 - Really anything with computer vision ML training
 - Red
 - Lifeguards
 - Red clothes vs blood
 - Emergency vehicles (e.g., boats)
 - Anything big
 - Lifeguard tower
 - Cars (idk abt this)
- Radio transmission to lifeguards & police
 - If SW recognizes someone for a certain period of time is struggling in the water OR if there is a shark dangerously near shore/someone, the drone transmits lifeguard to lifeguard/police
 - Drone needs radio that is tuned to the same channel as the one emergency responders have
- Finds missing kids using facial recognition
 - Going to need footage before of the kid. Then track them down
 - Needs cameras
- Going underwater
 - I really don't think this is a good idea
 - But if it did it would need like a motor at its butt to propel it forwards...?
 - Also definitely like 3 cameras (forward, bottom, and backwards in case there's anything behind it. Because I would definitely not want to kill a fish)
- Charging: drone has very short battery life :(
 - Solar-powered
 - Solar cells connect to battery
 - A ton of solar cells to sufficiently support the drone...problem bc weight
 - Self-charging
 - Need charging ports at lifeguard station
 - Need CV to properly identify the charging ports & training beforehand.
 - Also probably need CV to get out of the charging port...or hand-program the path for the drone to get out
- Navigate caves

- Search operation
- Use lights
- Need another camera at the top of the drone because no bumpy in top of cave and maybe sides too because of stalactites
- Responds to text messages or calls for help
 - Needs to have its own phone # (both drone and users)
 - If it's an urgent issue, person in water danger
 - All connection goes to lifeguard station
 - Geolocation should be transmitted from phone so drone can be deployed immediately
 - Coordinates, go to ocean area to search
 - Either autonomous or manually operated
- Leads lifeguard to person in danger
- Finds animals too close to shore
 - Turtles
 - Dolphins
 - Etc
- Floatable
- Self-operated/autonomous
- Manually-operated by lifeguard
- Tide
- Speaker
 - Lifeguard?
- Wire to beach
- Screen
- Cameras
- 2 drone collab, snapping point
- Sonic depth measurement

List of items to bring when flying drone:

- Laptop computer
 - Only use Ubuntu when updating PX4 firmware
- PS4 controller/joystick
- RC plane controller
 - Charged batteries
- Box of tools and horror (“Development Kit”)
 - Mini hex wrenches
 - The thing that holds and twists nuts
- Both screw boxes
- Charged batteries for drone
- Drone
- Spare propellers
- Instruction manual
- Voltmeter
- Pliers
- Drone certificate recreational

- Another thing to take notes on & photos
- Phone app B4UFly in case flying in illegal area
- Ham radio certificates
- Sunglasses or eye goggles
- Hat
- Sunscreen
- Drink
- Mini table
- Telemetry H30
 - o Batteries
- MUST UPDATE ONCE ELECTRONIC SYSTEM BLOCK IS DONE
 - Electronic system block for drone:
- HM30: FULL HD DIGITAL IMAGE TRANSMISSION SYSTEM
- 30km range
- 150 ms low latency
 - o Low latency is important otherwise seeing drone fly late but data transmission is slow
- 1080 full HD
 - o Good enough resolution to drive drone
 - o Need 4-8 Mbps (4000-8000 kbps) to have good resolution
 - o Rn telemetry is 56 kbps 😊

Note: MAVLink (micro aerial vehicle link) in reality a combo of MAVLink, tcp/ip, and ethernet

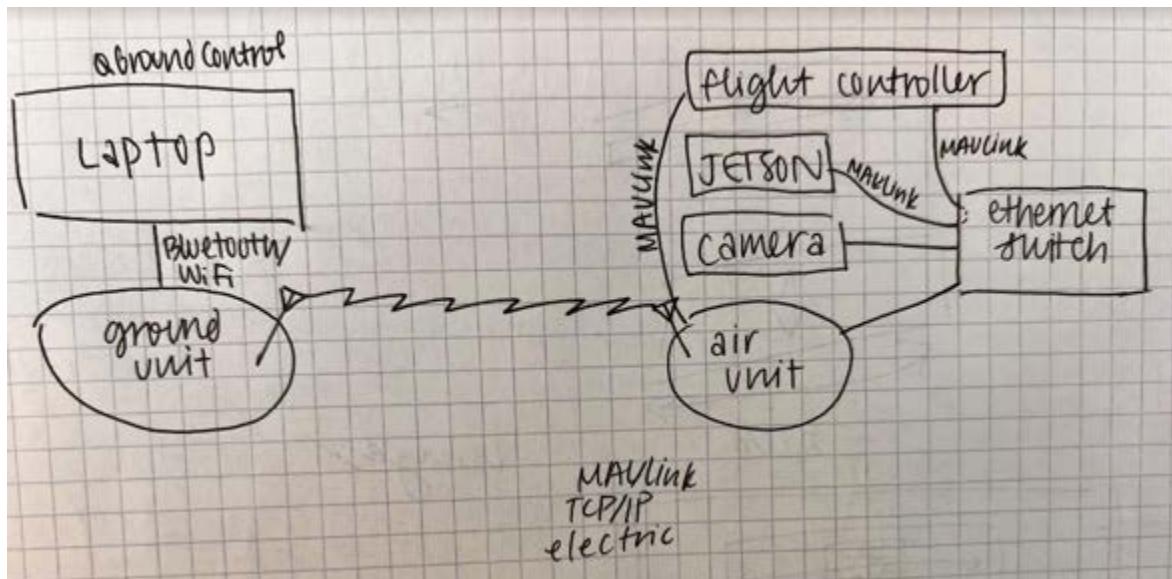


Diagram created and photograph taken by the author

Diagram of flight controller outputs:

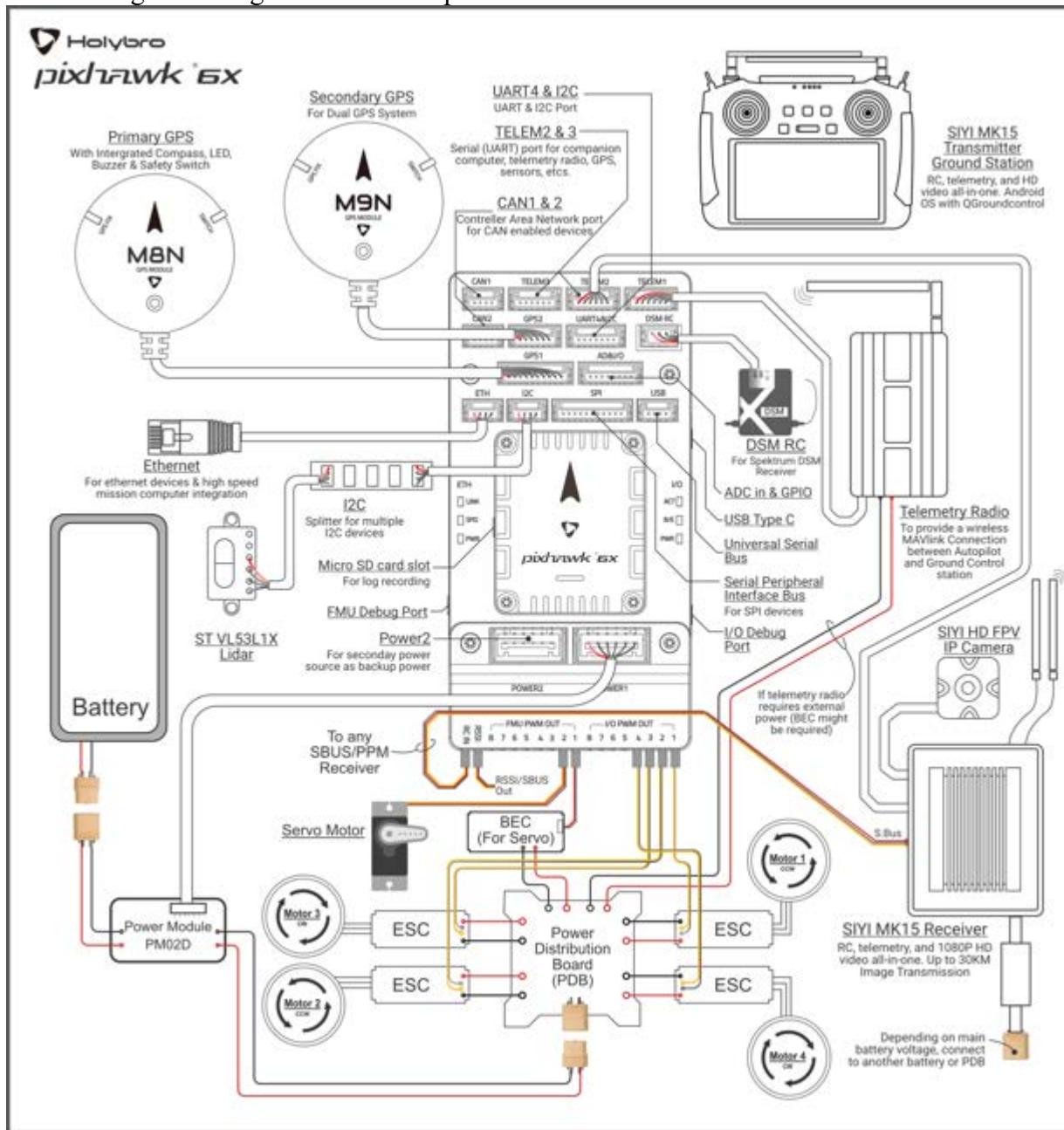


Image taken from PX4 User Guide

(https://docs.px4.io/main/en/flight_controller/pixhawk6x.html)

https://docs.px4.io/main/en/flight_controller/pixhawk6x.html

HM30 unboxing: [https://shop.siyi.biz/products/siysi-hm30?VariantsId=10748](https://shop.siyi.biz/products/siyi-hm30?VariantsId=10748)



Photographs taken by the author

HM30 ground transceiver:

Components of HM30:



Photographs taken by the author

Ground and air unit transceivers:

Batteries for HM30 ground controller transceiver:



Photographs taken by the author

Gimbal:

Cables & connectors for gimbal:



Photographs taken by the author

06/25/23

At beginning, QGroundControl kept giving error for arm calibration. So had to calibrate accelerometer, gyroscope, orientation, etc. They were preventing arm from passing test that initialization gave.

Wind is a factor. A big one. Sometimes when not touching joystick, drone moves on its own.

Battery runs out quickly. And when that happens, drone starts tilting over and crashing into ground during takeoff.

Tested thrust vector by hanging luggage-weight scale on a wire connected in between drone landing gear. Got 2.3 lbs for drone flying upwards but not completely accurate I think (wind, im a new driver, etc).

Also need to make sure drone battery is good when testing thrust. Will need to do more tests w vertical and sideway tests.

Also drone crashed into fence but thankfully only propellers were chipped a bit. But didn't affect flying.

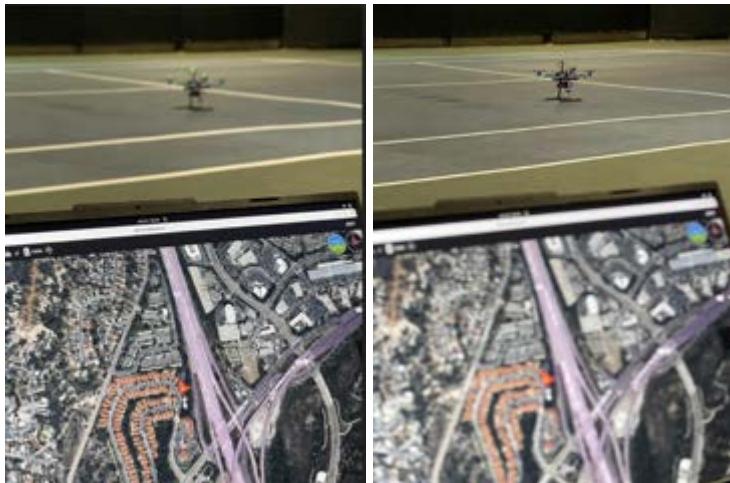
Must change battery level indicator because it's supposed to be ~15.89V but when it was 9.89V still marked as "100%" on QGroundControl.

Chipped propeller (bottom right)



Photograph taken by the author

QGroundControl setup:



Photographs taken by the author

author

Battery issue:



Before

Photograph taken by the author



After 😊

Photograph taken by the author

I also need to practice more driving.

06/26/23

- Shark detection & deterrent
 - o Identify shark
 - o Identify a person near the shark within radius
 - o Magnet—electric fields deter sharks
 - o Ultrasonic devices—emit high-frequency sound waves to deter sharks
 - Drop ultrasonic probe from string
 - o If they don't work, notify
- Drops

- Rescue tube
- Life jacket (like the ones on planes) for smaller drones
 - Self-inflatable when pull the tab
- Dropped when see person in rip current (for >2 min?)
- If drone goes into caves & someone trapped:
 - First aid kit
 - Water bottle (but kinda heavy)
 - Food (protein bars)
- Aesthetics
 - Lights during the night  and maybe during cave searches also
 - Neon colors: yellow, orange, pink, red, darker ones too?
 - For lifeguards to easily see in the sky?
- Drops rope/rescue can/inflatable device with long rope attached to end & ...
 - Drags person back to shore bc still holding other end of rope
 - Brings rope to shore for people
 - to pull the person out
 - for lifeguard to follow rope & find them (see lifeguard section)
 - Needs to anchor rope somehow so it doesn't get pulled away by current
 - Rescue can/inflatable > rope bc don't need to pull person upwards bc rescue can floats itself! I love rescue cans
- SW: Identifies...
 - Need camera: probably two (one at front, one pointing downwards)
 - Rip tides-with given function
 - Animals
 - Sharks
 - Sting rays (?)
 - Seals
 - People
 - Really anything with computer vision ML training
 - Red
 - Lifeguards
 - Red clothes vs blood
 - Emergency vehicles (e.g., boats)
 - Anything big
 - Lifeguard tower
 - Cars (idk abt this)
- Radio transmission to lifeguards & police
 - If SW recognizes someone for a certain period of time is struggling in the water OR if there is a shark dangerously near shore/someone, the drone transmits lifeguard to lifeguard/police
 - Drone needs radio that is tuned to the same channel as the one emergency responders have
- Finds missing kids using facial recognition
 - Going to need footage before of the kid. Then track them down
 - Needs cameras
- Going underwater

- I really don't think this is a good idea
- But if it did it would need like a motor at its butt to propel it forwards...?
- Also definitely like 3 cameras (forward, bottom, and backwards in case there's anything behind it. Because I would definitely not want to kill a fish)
- Charging: drone has very short battery life :(
 - Solar-powered
 - Solar cells connect to battery
 - A ton of solar cells to sufficiently support the drone...problem bc weight
 - Self-charging
 - Need charging ports at lifeguard station
 - Need CV to properly identify the charging ports & training beforehand.
 - Also probably need CV to get out of the charging port...or hand-program the path for the drone to get out
- Navigate caves
 - Search operation
 - Use lights
 - Need another camera at the top of the drone because no bumpy in top of cave and maybe sides too because of stalactites
- Responds to text messages or calls for help
 - Needs to have its own phone # (both drone and users)
 - If it's an urgent issue, person in water danger
 - All connection goes to lifeguard station
 - Geolocation should be transmitted from phone so drone can be deployed immediately
 - Coordinates, go to ocean area to search
 - Either autonomous or manually operated
- Leads lifeguard to person in danger
 - "Leading"
 - Need to pinpoint accurate location of the person in danger and remember it when getting lifeguard
 - Also need to know location of lifeguard tower
 - Need to alert lifeguard somehow (text? Signal?)
 - "Flagging"
 - Send signal to lifeguard
 - Lifeguard has drone detection device or just sees drone because it's brightly colored
 - Lifeguard goes to drone, who hovers over person in danger
 - Could also be pulling person up from water at same time
 - Rope—assuming person can float on their own
 - Drone gives person rope, person holds rope
 - Drag rope to shore & anchor it somehow
- Finds animals too close to shore
 - Turtles
 - Dolphins
 - Etc
- Floatable

- Self-operated/autonomous
- Manually-operated by lifeguard
- Tide
- Speaker
 - o Lifeguard?
- Wire to beach
- Screen
- Cameras
- 2 drone collab, snapping point
 - o In case there are 2 things happening
 - o Or if reinforcement is needed (weight, quantity of people)
- Sonic depth measurement

06/27/23

Beach data collection planning questions

- Where? Any restrictions?
 - o Del Mar Beach - I must fly over the ocean but not the beach:
<https://www.delmar.ca.us/808/Drones>
 - o Torrey Pines Beach – No
 - But can fly north (6th St) or south of it (Glider Port):
<https://torreypine.org/home-2-2/visit-the-reserve/protect-torrey-pines-reserve/drones/#:~:text=Drone%20flight%20IS%20permitted%20by,boundary%20is%20the%20Glider%20Port.>
- When?
 - o
 - No wind that can push the drone around
 - No rain
 - o Not too many people around (preferable but not necessary ig)
 - o Still sun left
 - o Tide changing: when it's going from highest tide to lowest tide because then there will be more rip currents
- Any legal requirements?
 - o FAA recreational flying certification
 - o HAM Radio license
 - o Get drone registered because weight & label drone with registration number
[https://www.faa.gov/uas/getting_started/register_drone#:~:text=Label%20Your%20Drone,a%20condition%20that%20is%20legible.\)](https://www.faa.gov/uas/getting_started/register_drone#:~:text=Label%20Your%20Drone,a%20condition%20that%20is%20legible.)
- What kind of data needed?
 - o Videos of the ocean with and without rip currents
 - o Videos of sharks
 - o Videos of people
- How much data needed?
 - o 5 hours
- Which device store data?
 - o JETSON

- How to plan flight, path, path length, speed, altitude?
 - Plan path: QGroundControl
 - ~4,000m path from Del Mar Beach. Starting around the train tracks and ending right before Del Mar Dog Beach
 - Limitation: drone needs to be in my line of sight. Bending (see plan 0 below) is a limitation
 - Telemetry data range (<https://holybro.com/collections/telemetry-radios/products/sik-telemetry-radio-v3?variant=41562952302781>)
 - 300m range 😞
 - Altitude: under 400 ft (legal restriction)
 - Use trig
 - Camera needs to capture the full rip current with some space at the edges.
 - Gimbal A8 FOV = 93°
 - What's the rip current size?
 - Can be 50 ft to 50 yards wide (<https://cslsa.org/content/Rip-Currents.html#:~:text=Rip%20currents%20can%20be%2050,or%20hundreds%20of%20yards%20offshore.>)
 - Speed: I do not know
 - Does it have to be slower to capture multiple waves?
 - Or can I just use photos to analyze rip currents
- How long is the flight?
 - 4000m
 - In time: NEED TO MEASURE BATTERY LIFE w decent margin
 - Need to calculate how much flight time margin against battery life
- Safety, contingency, and emergency plan
 - Don't put drone at limits of telemetry radio range (300m)
 - But the range changes depending on the weather (eg temperature, wind, seagull)
 - So need safety protocol: what does drone do if loses contact?
 - Drone needs to return back to original position where it took off
 - Make sure that setup is enabled in QGroundControl
- Based on the issues brought up, Plan 0 updated to become Plan 1
Plan 0

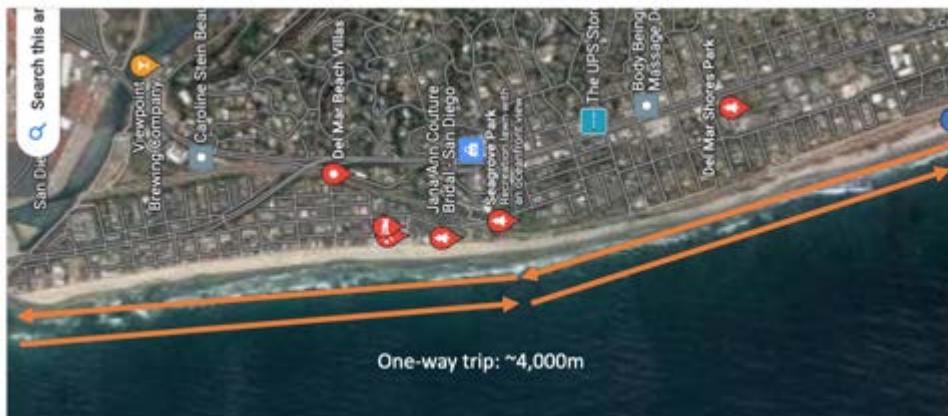


Image created by the author using QGC

Plan 1

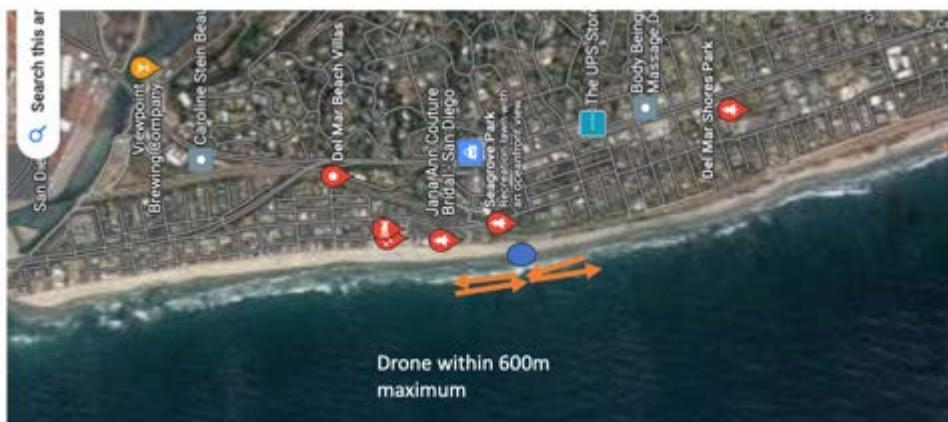


Image created by the author using QGC

TOMORROW; GET DRONE REGISTERED BC TAKES 16-20 DAYS FOR IT TO GO THRU

06/28/23

Main goal: get drone registered

Drone has been registered:



Small UAS Certificate of Registration

REGISTERED OWNER: **Angellina Kim**

REGISTRATION NUMBER: **FA3HRKM494**

ISSUED: **06/28/2023**

EXPIRES: **06/28/2026**

This Small UAS Certificate of Registration is not an authorization to conduct flight operations with an unmanned aircraft. Operators of unmanned aircraft must ensure they comply with the appropriate safety authority from the FAA. To operate as a recreational flyer, a person must meet all of the statutory conditions of law exception for limited recreational operations of unmanned aircraft (49 U.S.C. 44808). Persons who do not meet all of the statutory conditions may not operate under the statutory exception for limited recreational operations of unmanned aircraft.

For U.S. citizens, permanent residents, and certain non-citizen U.S. corporations, this document constitutes a Certificate of Registration. For all others, this document represents a recognition of ownership.

To fly under the exception for recreational flyers you must:

- Have a current registration
- Fly only for recreational purposes
- Follow the safety guidelines of a community-based organization
- Keep your drone within your visual line of sight
- Give Way and do not interfere with any manned aircraft
- Fly at or below 400' in controlled airspace and only with prior authorization
- Fly at or below 400' in uncontrolled airspace
- Comply with all airspace restrictions
- Pass The Recreational UAS Safety Test

Image captured by the author

06/29/23

Calculations for Beach Data Collection

	Siyi A8	GoPro Hero10 (Linear, standard/high)	Raspberry Pi Camera Module 3 (diagonal FOV)
FOV angle (degree)	93	87	75
Target coverage (m)	50	50	50
Altitude (m)	23.7	26.3	32.6
Camera pixels	4096	5120	4608
Pixel size (m)	0.0122	0.0098	0.0109

Table created by the author

Original calculations ^

I calculated these using trig. Eg for Siyi A8: $50/x = \tan(93^\circ/2) \rightarrow x = \text{altitude}$, $93^\circ = \text{FOV}$ (usually horizontal) angle

Found pixel size by doing target coverage / camera pixels. Eg for Siyi A8: $50 / 4096 = 0.0122$

2 issues:

1. I don't know how much coverage needed to get rip current
2. I don't know how much coverage needed to get animals

So I created 2 more tables

	Siyi A8	GoPro Hero10 (Linear, standard/high)	Raspberry Pi Camera Module 3 (diagonal FOV)
FOV angle (degree)	93	87	75
Target coverage (m)	100	100	100
Altitude (m)	47.4	52.7	65.2
Camera pixels	4096	5120	4608
Pixel size (m)	0.0244	0.0195	0.0217

Table created by the author

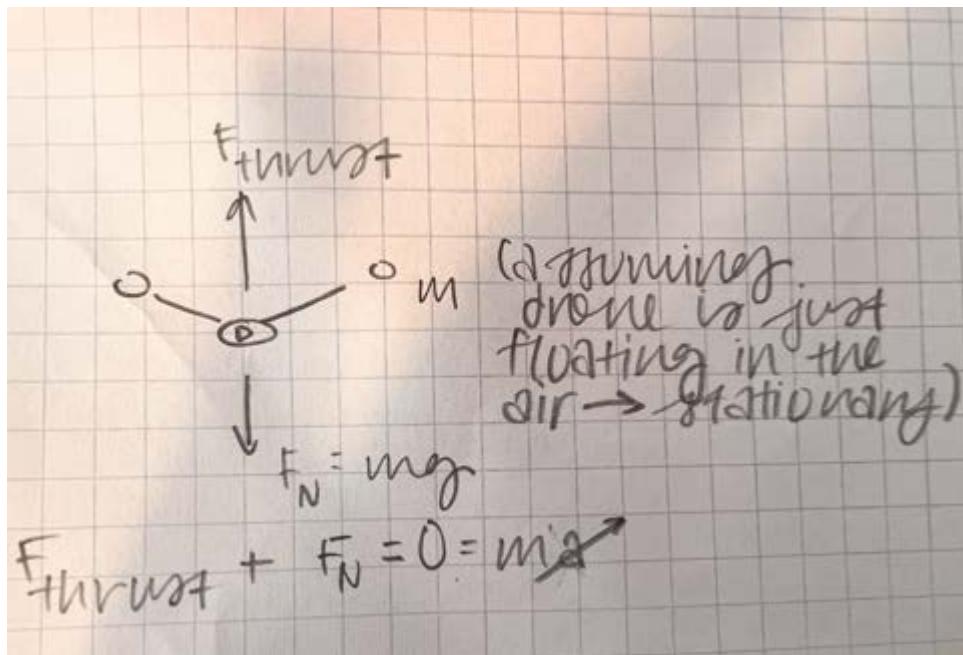
	Siyi A8	GoPro Hero10 (Linear, standard/high)	Raspberry Pi Camera Module 3 (diagonal FOV)
FOV angle (degree)	93	87	75
Target coverage (m)	150	150	150
Altitude (m)	71.2	79.0	97.7
Camera pixels	4096	5120	4608
Pixel size (m)	0.0366	0.0293	0.0326

Table created by the author

I'll need to do multiple rounds of videoing with fresh batteries at varying heights.

I'll pass through the same path but with different heights each time.

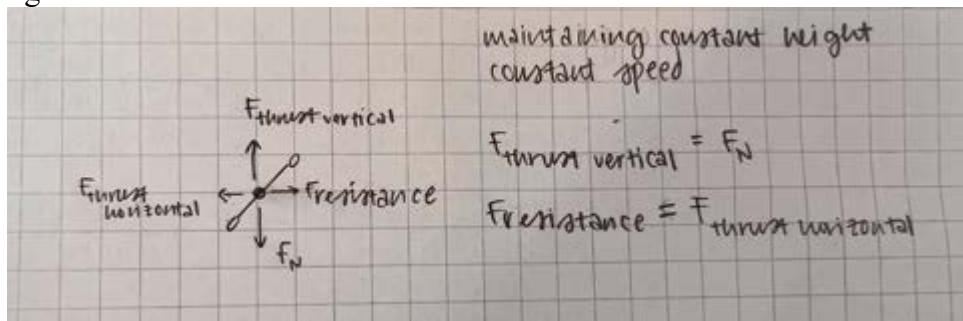
Force calculations on stationary drone:



Diagram

created and photograph taken by the author

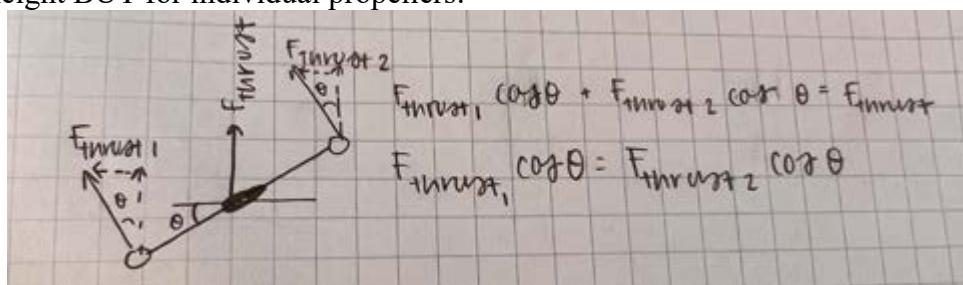
Force calculations on drone moving horizontally with constant speed but maintaining same height:



Diagram

created and photograph taken by the author

Force calculations on drone moving horizontally with constant speed but maintaining same height BUT for individual propellers:



Diagram

created and photograph taken by the author

Basic drone equations & summary of how it rolls/pitches/yaws: http://ffden-2.phys.uaf.edu/webproj/211_fall_2018/J-Rod_Maltos/physics_4th.html

Electronic speed controller stuff: <https://www.jouav.com/blog/electronic-speed-controller-esc.html>

Info on GoPro10: https://community.gopro.com/s/article/HERO10-Black-Digital-Lenses-FOV-Informations?language=en_US

Info on Raspberry Pi Camera Module 3: <https://www.raspberrypi.com/products/camera-module-3/>

Study tmrw: <https://towardsdatascience.com/demystifying-drone-dynamics-ee98b1ba882f>

(because something about my diagrams doesn't feel right...)

07/08/23

Looking at different types of drones contd.

[Meet the dazzling flying machines of the future | Raffaello D'Andrea](#)



Image taken from “Meet the dazzling flying machines of the future | Raffaello D'Andrea”, (<https://www.youtube.com/watch?v=RCXGpEmFbOw>)

[The astounding athletic power of quadcopters | Raffaello D'Andrea](#)



Image taken from “The astounding athletic power of quadcopters | Raffaello D’Andrea”, (<https://www.youtube.com/watch?v=w2itwFJCgFQ>)

07/11/23

Surf Zone Fatalities in US: NOAA and NWS

<https://www.weather.gov/safety/ripcurrent-fatalities>

Most fatalities from surf zone comes from rip currents

Nationwide Incident Data	
YEAR	2022
Total Agencies Reporting	148
Beach Attendance	12,685,010,374
Rescues	3,060,110
Preventive Actions	177,293,695
Medical Aids	8,639,136
Boat - Rescues	293,919
Boat - Passengers	441,058
Boat - Vessel Value	3,593,759,669
Drowning - Unguarded	3,602
Drowning - Guarded	757
Lost and Found	734,799
Public Safety Lectures	2,178,823
Students Attending	16,313,675

Image taken from USLA (<https://www.usla.org/page/Statistics>)

^^ US Lifesaving Association Data: <https://www.usla.org/page/Statistics>

New York uses drones to patrol for sharks amid rise in encounters:

<https://www.nbcnews.com/news/us-news/new-york-beach-officials-use-drones-search-sharks-spate-attacks-swimme-rcna93029>

Design of the Life-Ring Drone Delivery System for Rip Current Rescue:

- 93 seconds for lifeguards to get to person drowning/struggling maximum time
 - Some victims have survival times as low as 60 seconds
 - Thus drone must be swift and not survey 1 meter for 1 hour
- ^^

G. Xiang, A. Hardy, M. Rajeh and L. Venuthurupalli, "Design of the life-ring drone delivery system for rip current rescue," 2016 IEEE Systems and Information Engineering Design Symposium (SIEDS), Charlottesville, VA, USA, 2016, pp. 181-186, doi: 10.1109/SIEDS.2016.7489295.

Coaxial tilt-rotor hovers smoothly in any orientation:

<https://newatlas.com/drones/omnidirectional-tilt-rotor-drone/>

Benefit: drone can maintain any orientation

Under vs over actuated: The Effects of UAV Quadcopter Propeller Tilt Angle on Flight Stability: https://www.laccei.org/LACCEI2020-VirtualEdition/full_papers/FP316.pdf

More overactuation (AKA tilting) = improved flight stability

Benefits of tilt rotors:

<https://www.frontiersin.org/articles/10.3389/frobt.2022.1033715/full>

- Drone can get faster because less drag --> less affected by the wind

07/12/23

Dual-Axis Tilting Quadcopter:

https://www.researchgate.net/publication/328470521_Design_and_Implementation_of_a_Dual-Axis_Tilting_Quadcopter

<https://www.wevolver.com/specs/dual-axis.tilting.quadcopter>

Shows design of servos

<https://dronenodes.com/quadcopter-motor-propeller-rotation/>: useful for understanding what best ratios (e.g., thrust, propeller size...) of drone will be

07/13/23

Demonstrate SIYI HM30 video and data link

To tackle QGroundControl video issue (because it just didn't work): install programs following instructions under Linux

<https://github.com/mavlink/qgroundcontrol/blob/master/src/VideoReceiver/README.md>

```
list=$(apt-cache --names-only search ^gstreamer1.0-* | awk '{ print $1 }' | sed -e '/-doc/d' | grep -v gstreamer1.0-hybris)
```

```
sudo apt-get install $list
```

```
sudo apt-get install libgstreamer-plugins-base1.0-dev
sudo apt-get install libgstreamer-plugins-bad1.0-dev
```

After installation, QGC showed video control options

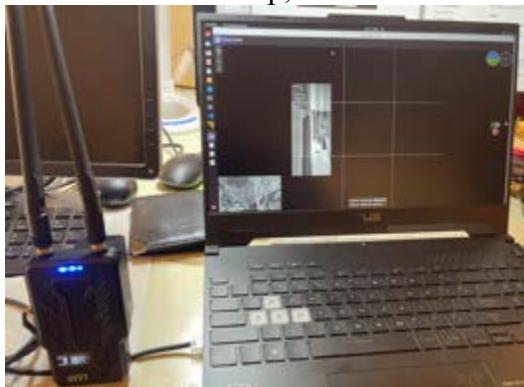


Photograph taken by the author

Not work: rtsp://192.168.144.25:8554/ch01.264 (IP address & port of IP camera...first camera I tried to use for testing bc simpler & less setup)

Worked somehow: rtsp://192.168.144.25:8554/main.264

Video format – mkv, mov, or mp4 doesn't have much difference on computer screen
Screen now shows up, but slow and late when updating ;(



Photograph taken by the author

Telemetry not working....perhaps connection wrong?

IP network = okay – ping reaches air unit 192.168.144.11 and IP camera 192.168.144.25

07/14/23

Telemetry connection not established

With Ubuntu 22.04, tried 1) Bluetooth, 2) Type-C USB, and 3) UDP network. 4) UART cannot be done bc adapter = not available

1) Bluetooth: SIYI-6Axxxx device appears. When selected and pin is entered, it is briefly connected but disconnects in a few seconds. So can't be used in QGC for telemetry

2) Type-C: Connected type-C USB to laptop. /dev/ttyACM0 is created. Create CommLink in QGC. When "Connect" is requested, QGC returns permission error

Added user to dialout group with

```
sudo adduser $USER $(stat --format="%G" /dev/ttyACM0 )
```

But...still getting the same permission error

Gave up Ubuntu for this telemetry issue and video issue.

Moved to Windows, downloaded QGC, tried 1) Bluetooth, 2) Type-C USB, and 3) UDP network.

1) Bluetooth: SIYI-6Axxxx device can be connected, but it is disconnected in a few seconds. Same as Ubuntu

2) Type-C USB: When plugged in, COM6 port shows in the device selection. At QGC, created COM6 port. “Connect” gives permission error. Suspect the same issue as Ubuntu

3) UDP network: There seems to be uplink data going, but not downlink. Still not working. Maybe the cable issue, where TX and RX are swapped? Need to check with customer service.



Photograph taken by the author

Need to update firmware? SIYI firmware version is 0.1.9. The latest is 0.2.0, but download does not work

Tried video link at QGC in Windows. Streaming worked & delay = low. Video is temporarily broken when chunk of image changes.

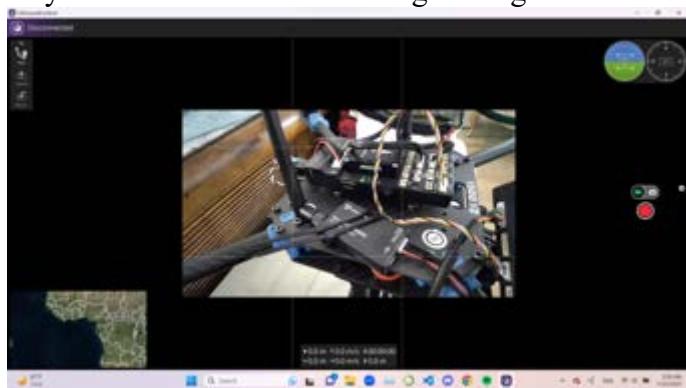


Image taken by the author

Telemetry connector:



Photograph taken by the author

Flight controller side: <https://docs.holybro.com/autopilot/pixhawk-baseboards/pixhawk-baseboard-ports>

Telem1, Telem2, Telem3 ports

Pin	Signal	Volt
1(red)	VCC	+5V
2(black)	TX7/5/2 (out)	+3.3V
3(black)	RX7/5/2 (in)	+3.3V
4(black)	CTS7/5/2 (in)	+3.3V
5(black)	RTS7/5/2 (out)	+3.3V
6(black)	GND	GND

Image taken from Holybro (<https://docs.holybro.com/autopilot/pixhawk-baseboards/pixhawk-baseboard-ports>)

Air unit connector: https://drive.google.com/drive/folders/12QkP0RelaofuNZdhwRwMy-EEdIEN_7Jf?usp=share_link

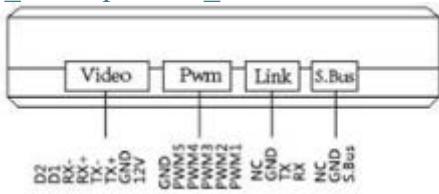


Image taken from SIYI (<https://shop.siyi.biz/>)

Connector mapping:

- FC #2 TX -> AU RX
- FC #3 RX -> AU TX

A Smart radio connection example

<https://doodlelabs.com/wp-content/uploads/2022/09/QGroundControl-and-Pixhawk-Advanced-Setup-v01020.pdf>

07/15/23

How to connect PX4 to JETSON:

<https://www.hackster.io/Matchstic/connecting-pixhawk-to-raspberry-pi-and-nvidia-jetson-b263a7>

<https://discuss.px4.io/t/connecting-a-pixhawk-2-4-8-to-nvidia-jetson-nano/14166>

<https://discuss.ardupilot.org/t/jetson-nano-pixhawk-cube-communication/73371/2>

Trying SIYI A8 gimbal

Removed the simple IP camera, and connected SIYI A8 gimbal. A8 functional. Video quality clearer than the simple IP camera. So, it works!!!



Photographs taken by the author

Next steps:

- 1) connect gimbal to JETSON through ethernet
- 2) JETSON record video from gimbal
- 3) connect HM30 air unit to ethernet (need Amazon order by 7/18)
- 4) form on-board network with ethernet switch to connect a) gimbal, b) JETSON, and c) air unit
- 5) JETSON forwards video to QGC through HM30 radio

07/16/23

Gimbal to JETSON connection

- Connected gimbal to TP-LINK ethernet switch
- Connected JETSON to ethernet switch
- Powered up JETSON and ethernet switch
- Assigned 192.168.144.20 to JETSON ethernet connection (to simulate HM30 ground unit)
- From JETSON, ping to 192.168.144.25 has <0.5ms delay

VLC not stable in JETSON. Cannot use it for RTSP display to confirm A8 video streaming

Gstreamer is an alternative while full screen:

<https://stackoverflow.com/questions/65725208/error-during-rtsp-streaming-using-gstreamer-on-the-jetson-nano>

gst-launch-1.0 uridecodebin uri=rtsp://127.0.0.1:8554/test ! nvoverlaysink

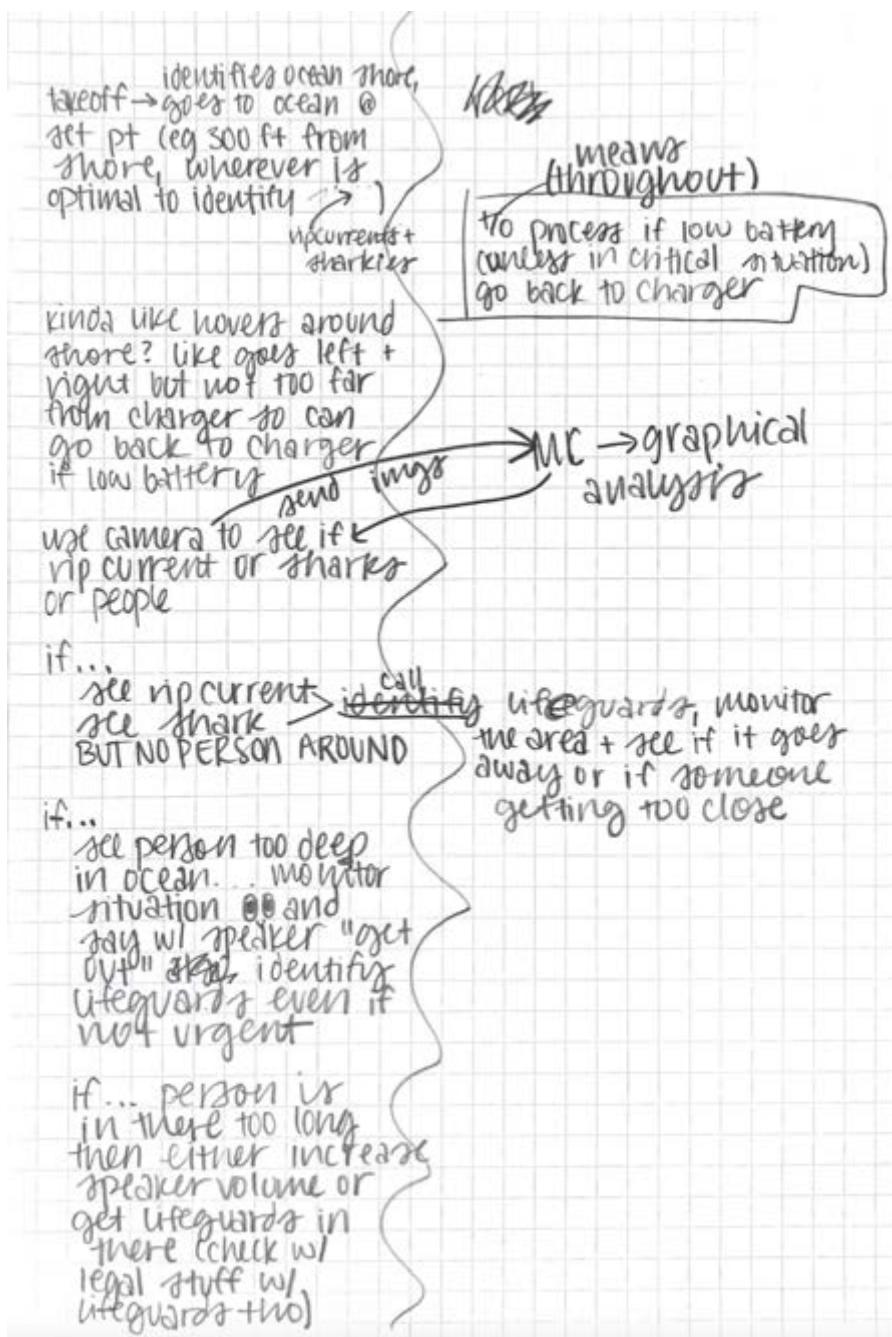
JETSON displayed stream video on monitor it was connected to!!!! Latency about 4s



Photograph taken by the author

07/17/23

Patent flowchart first draft:



Written and

photograph taken by the author

TCP/IP vs UDP:

- TCP/IP ensures that the data packet is received
 - o Slower
 - o JETSON SSH
- UDP is brute force, continuously sends data without confirmation that it has been received
 - o Faster
 - o HM30s use it

IPv4

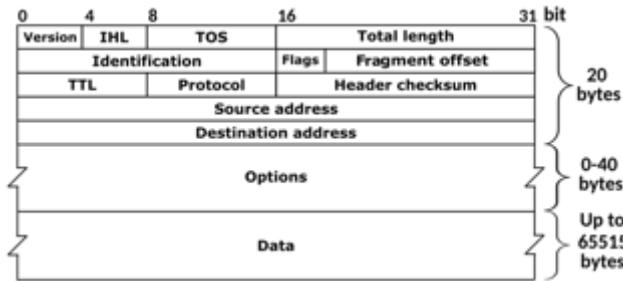


Image taken from Wikipedia

(https://en.wikipedia.org/wiki/Internet_Protocol_version_4)

- Each packet has source & destination address
- Packet broken into pieces which are sent

OSI model

by layer

7. Application layer [show]
6. Presentation layer [show]
5. Session layer [show]
4. Transport layer [show]
3. Network layer [show]
2. Data link layer [show]
1. Physical layer [show]

V*T*E

Screen captured from Wikipedia

(https://en.wikipedia.org/wiki/OSI_model)

- Ethernet & HM30 are physical layer
- All connections are treated the same at network layer. Does not care if it's air connection or wireless

Network diagram with IP addresses:

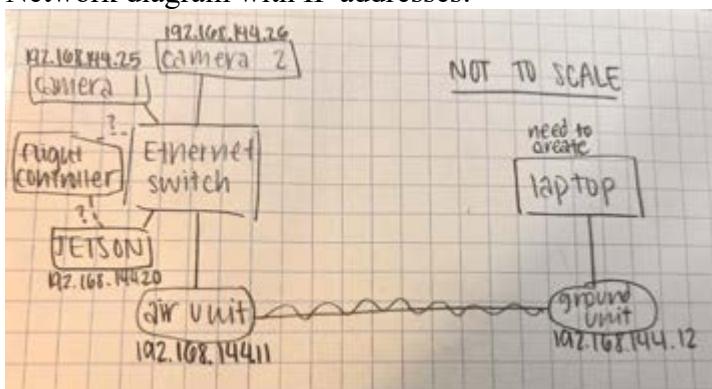


Diagram created and

photograph taken by the author

Essential: have JETSON record and store videos for data collection

Non-essential: send data collection from JETSON --> ethernet switch --> air unit --> ground unit --> laptop

GStreamer: “a pipeline-based multimedia framework that links together a wide variety of media processing systems to complete complex workflows.”

https://github.com/Cinderpe1t/TBS_Robotics_Low_Latency_Video_Transport_with_gstreamer

```
gst-launch-1.0 nvarguscamerasrc sensor-id=0 sensor-mode=4 \
    ! 'video/x-raw(memory:NVMM),width=1280,height=720,format=NV12,framerate=60/1' \
! nvvidconv flip-method=2 \
    ! 'video/x-raw(memory:NVMM),format=RGBA,width=640,height=360' \
! comp. nvarguscamerasrc sensor-id=1 sensor-mode=4 \
    ! 'video/x-raw(memory:NVMM),width=1280,height=720,format=NV12,framerate=60/1' \
! nvvidconv flip-method=2 \
    ! 'video/x-raw(memory:NVMM),format=RGBA,width=640,height=360' \
! nvcompositor name=comp sink_0::xpos=0 sink_0::ypos=0 sink_0::width=640 sink_0::height=360 \
sink_1::xpos=640 sink_1::ypos=0 sink_1::width=640 sink_1::height=360 \
! nvvidconv \
    ! 'video/x-raw(memory:NVMM),format=NV12,framerate=30/1' \
! nvjpegenc \
! rtpjpegpay \
! udpsink host=192.168.0.100 port=5010
```

Developing streaming with GStreamer

>>Displays RTSP streamed video for debugging

```
gst-launch-1.0 uridecodebin uri=rtsp://192.168.144.25:8554/main.264 ! nvoverlaysink
```

>>Save video image from the stream for image analysis

```
gst-launch-1.0 uridecodebin uri=rtsp://192.168.144.25:8554/main.264 ! nvjpegenc !
multifilesink location=~/px4_video/snapshot.jpg
```

>>Implement tee to perform two different tasks from the input stream

>>1: display stream, 2: save image from stream. what is the image update rate?

```
gst-launch-1.0 uridecodebin uri=rtsp://192.168.144.25:8554/main.264 ! tee name=stream_t \
! queue ! nvoverlaysink \
stream_t. ! queue ! nvjpegenc ! multifilesink location=~/px4_video/snapshot.jpg
```

>>Try RTSP sink, but JETSON does not have rtspsink. Did not work in the end

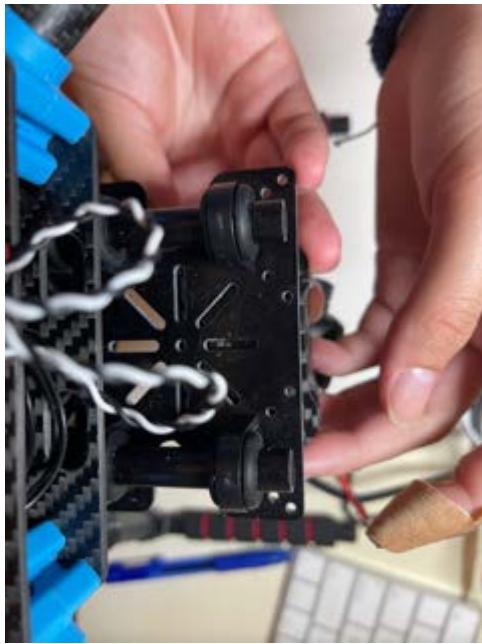
```
gst-launch-1.0 uridecodebin uri=rtsp://192.168.144.25:8554/main.264 ! tee name=stream_t \
! queue ! rtspsink service=5000 \
stream_t. ! queue ! nvjpegenc ! multifilesink location=~/px4_video/snapshot.jpg
```

>>UDP sink

```
gst-launch-1.0 uridecodebin uri=rtsp://192.168.144.25:8554/main.264 ! tee name=stream_t \
! queue ! udpsink host=192.168.144.21 port=5000 \
stream_t. ! queue ! nvjpegenc ! multifilesink location=~/px4_video/snapshot.jpg
```

Tomorrow:

- 1) Connect HM30 air unit to ethernet
- 2) Code the pitch of the gimbal to be tilted rather than 180°
- 3) Make sure mechanically connects to drone (crossing fingers that there's enough space to mount it)



Photograph taken by the designated supervisor

Just checked—drilling holes will not be necessary, gimbal holes match holes in drone

NVIDIA Jetson Nano GStreamer example pipelines

https://developer.ridgerun.com/wiki/index.php?title=Jetson_Nano/GStreamer/Example_Pipelines

How to use GStreamer tee

<https://stackoverflow.com/questions/45873407/gstreamer-tee-with-different-capabilities-on-each-branch>

<https://forums.developer.nvidia.com/t/gstreamer-record-and-view-video-simultaneously/183131>

How to change video rate

<https://gstreamer.freedesktop.org/documentation/videorate/index.html?gi-language=c>

07/19/23

Code the pitch of the gimbal to be tilted rather than facing forward

Gimbal user guide:

https://drive.google.com/drive/folders/19Hs92ClHiRWIjE9xS32Y6GToXNvGveLO?usp=share_link

Need CRC --> error-checking program

List of commands for gimbal:

0x01	Acquire firmware version
0x02	Acquire hardware ID
0x04	Auto focus
0x05	Manual zoom and auto focus
0x0F	Absolute zoom and auto focus
0x06	Manual focus
0x07	Gimbal rotation
0x08	Center
0x0A	Acquire gimbal configuration information
0x0B	Function feedback information
0x0C	Photo and video
0x0D	Acquire gimbal attitude
0x0E	Set gimbal control angle

Table created by the author

Command format:

Field	Index	Bytes	Description	Rotate	Centering	Lock mode	Set Gimbal angle
STX	0	2	0x6655	0x55	0x55	0x55	0x55
				0x66	0x66	0x66	0x66
CTRL	2	1	0: need ack, 1: ack_pack	0x01	0x01	0x01	0x01
DATA_LEN	3	2	DATA field byte length	0x02	0x01	0x01	0x04
				0x00	0x00	0x00	0x00
SEQ	5	2	Frame sequence (0 - 65535)	0x00	0x00	0x00	0x00
				0x00	0x00	0x00	0x00
CMD_ID	7	1	Command ID	0x07	0x08	0x0C	0x0E
DATA	8	DATA_LEN		0x64	0x01	0x03	0x00
				0x64			0x00
CRC16		2		0x3D	0xD1	0x57	0x84
				0xCF	0x12	0xFE	0x03
							...

Table created by the author

CRC (cyclic redundancy check: an [error-detecting code](#) commonly used in digital [networks](#) and storage devices to detect accidental changes to digital data) code in Python, migrated from C:

```
UDP_MESSAGE = bytes.fromhex("55 66 01 04 00 00 00 0e 00 00
7c fc")
#UDP_MESSAGE = bytes.fromhex("55 66 01 01 00 00 00 08 01")

crc_init = 0
crc = crc_init
for i in range(len(UDP_MESSAGE)):
    temp = (crc >> 8) & 0xff
    oldcrc16 = crc16_tab [UDP_MESSAGE[i] ^ temp]
    crc = (crc << 8) ^ oldcrc16
    crc = crc & 0xffff #limit to 16 bits
print("crc   :", hex(crc))
```

UDP command program:

```

import socket, sys

Camera1_IP='192.168.144.25'
Camera_UDP_Port=37260

# Create a UDP socket
Socket_Camer1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# Bind the socket to the port
Camera1_Address = (Camera1_IP, Camera_UDP_Port)
print("Camera IP address and port: ", Camera1_Address)

UDP_MESSAGE = bytes.fromhex("55 66 01 01 00 00 00 08 01 d1 12") #Centering
#UDP_MESSAGE = bytes.fromhex("55 66 01 01 00 00 00 0c 03 57 fe") #Lock mode
#UDP_MESSAGE = bytes.fromhex("55 66 01 01 00 00 00 0c 04 b0 8e") #Follow mode
#UDP_MESSAGE = bytes.fromhex("55 66 01 04 00 00 00 0e 00 00 84 03 d7 20") #90
degree pitch
#UDP_MESSAGE = bytes.fromhex("55 66 01 04 00 00 00 0e 00 00 7c fc 4f a4") #-90
degree pitch
#UDP_MESSAGE = bytes.fromhex("55 66 01 00 00 00 00 0d e8 05") #Acquire additude
data
#last 2 numbers are CRC obtained from previous program

print("Data to camera : ", UDP_MESSAGE)
Socket_Camer1.sendto(UDP_MESSAGE, Camera1_Address) #Send message to UDP port

#Receive data from camera
data, address = Socket_Camer1.recvfrom(Camera_UDP_Port)
print("Data from camera: ", data, address)

Socket_Camer1.close()

```

Run first program to get 2 CRC hexadecimal numbers and input them into UDP_MESSAGE for second program

Can point the camera directly to the ground as the drone is not always flat. Camera can always look at the ocean and dynamically adjusted.

Tomorrow: connect camera & JETSON & ethernet switch to air unit (actual drone) and see if it forwards to ground unit (computer & HM30s) --> so need to make ethernet cable



Photograph taken by the author

07/21/23

Make customer ethernet cable for A8 gimbal
 Gimbal's ethernet pin out from SIYI A8 manual
 Use straight-through connection to RJ45

<https://networkel.com/rj45-pinout-diagram/>

Originally planned to use ethernet crimping tool but decided to use existing ethernet cable and soldering

Cut flat ethernet cable to expose wires, solder to GH1.25 connector wires, then wrapped with polyimide masking tape

Flat ethernet cable:

https://www.amazon.com/dp/B07Z2T51DV?psc=1&ref=ppx_yo2ov_dt_b_product_details

GH1.25 connector kit:

https://www.amazon.com/dp/B07PBHN7TM?psc=1&ref=ppx_yo2ov_dt_b_product_details

Ethernet crimping tool:

https://www.amazon.com/dp/B0B7W9MHLV?psc=1&ref=ppx_yo2ov_dt_b_product_details



Image taken from SIYI (<https://shop.siyi.biz/>), Images taken from networkel (<https://networkel.com/rj45-pinout-diagram/>)

4 wires are used for ethernet: TX+, TX-, RX+, RX-

Confirmed ethernet connection with complete adapter



Photograph taken by the author

Next step: make custom ethernet cable for HM30 air unit

07/23/23

Considering attachment of air unit:

- How to power on (camera, JETSON, ethernet switch)? With only one battery

- Find voltage of each component
- Air unit HM30: up to 6S battery—directly attach to battery
- JETSON Nano: 5V, up to 2A
- Ethernet switch: 5V, up to .6A
- Camera A8: directly attach to battery
- Camera & air unit connected with XT30 connector (2 on drone)
- How power up JETSON & ethernet switch?
 - Flight controller: 5.2V
 - JETSON max is 5.25V—too close to margin
 - Not good if PM02 current connect (3A) to JETSON (2A) or ethernet switch (2.6A)
 - Need separate regulator that outputs 5V

Power supply planning

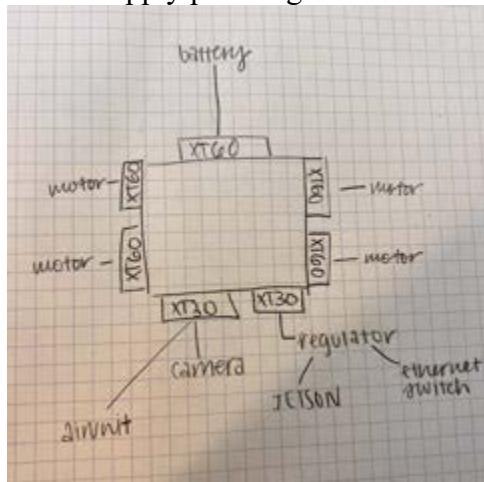


Diagram created and photograph taken by the author

Issue is do not have splitter for XT30 with air unit and camera. Need to order adapter cable for JETSON and ethernet to convert to XT30.



Photographs taken by the designated supervisor

^JETSON input power supply cable outer diameter and ethernet cable input power supply outer diameter

And order regulator, converts to lower voltage:

https://www.amazon.com/dp/B07DYXTX9H?psc=1&ref=ppx_yo2ov_dt_b_product_details

07/24/23

Current drone payload: 1 kg (<https://holybro.com/collections/x500-kits/products/px4-development-kit-x500-v2>)

Better drone: (<https://store.tmotor.com/goods-1294-MX860.html>)

- Payload = 9 kg
- Coaxial
- Covering over electrical system
- MN6009 KV:170
 - o “60” = radius of motor (mm)
 - o “09” = height of motor (mm)
 - o KV = speed of rotation (rd/min w propeller)

Hexacopter: (<https://store.tmotor.com/goods-843-M1500.html>)

- Payload = 15 kg
- 6 rotors, so bigger drone than coaxial one above

Motor example:

(https://www.aliexpress.us/item/3256802804732609.html?spm=a2g0o.store_pc_groupList.8148356.23.51c33ebann4ce5&pdp_npi=3%40dis%21USD%21US%20%24129.99%21US%20%24129.99%21%21%21%21%21%402103228816902507972672511e1360%2112000023101944022%21sh%21US%212627384811&gatewayAdapt=glo2usa)

- Expensive
- Single motor take-off thrust: 2kg
- Max thrust: 6.5kg
- If coaxial, thrust loss of 25%
- 4-motor max thrust: 11kg
- Powerful motors use 6S battery, unlike the current 4S battery I'm using
- Max propeller length: 22in, unlike current propeller length = 10in

Motor analysis specs:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	Model #	Motor size	KV	Idle current (A)	Rated voltage	Rated thrust (Kg)	Max Peak current (A)	Max Voltage (V)	Max Max thrust (Kg)	Max Max power (W)	Propeller (inch)	Weight (g)	Thrust /Weight	Thrust /Power	
Holybro	AIR2216II	2216	920	0.8	4S	1.3	17			272	T1045II				ht
T-MOTOR	V807	170										670			ht
T-MOTOR	U15 II	80										1740			ht
T-MOTOR	MN7005	7005	115			1.5		5		24		188	26.6		ht
T-MOTOR	MN4004	4004	300	0.2	4 - 6S		9		1.3	216	13 - 17	53	24.5	6	ht
T-MOTOR	MN4004	4004	400	0.2			12		1.3	300		53	24.5	4.3	
T-MOTOR	MN4006	4006	380	0.3	4 - 6S		16		2.3	380	13 - 16	68	33.8	6.1	ht
T-MOTOR	MN5006	5006	300	0.6	4 - 6S	1	19.63	23.09	2.996	500	17 - 18	108	27.7	6	ht
T-MOTOR	MN5006	5006	450	0.9	4 - 6S	1	26.28	22.89	3.218	650	14 - 15	108	29.8	5	ht
T-MOTOR	MN5006	5006	450		4 - 6S		24.82	14.96	2.609	371	17 - 18	108	24.2	7	ht
T-MOTOR	MN5008	5008			6 - 12S	1.2			4.2		17 - 18	135	31.1		ht
T-MOTOR	MN5208	5208	340		4 - 6S				4.1		16 - 18	145	28.3		ht
T-MOTOR	MN5212	5212	340	1.1	4 - 8S		31	24	4.355	744.7	15 - 18	205	21.2	5.8	ht
T-MOTOR	MN5212	5212	420	1.1	4 - 8S		59	24	5.918	1436.6	15 - 18	205	28.9	4.1	
T-MOTOR	MN6007	6007	160			2			5.5		21 - 22	172	32		
T-MOTOR	MN7005	7005													
T-MOTOR	MN8012	8012													
T-MOTOR	MN8014	8014													
T-MOTOR	MN8017	8017													
T-MOTOR	MN4120	4120	400	1.2	4 - 8S		47		4.3	1050		253			ht

Table created by the author

8/1/23

Enabling HM30 telemetry

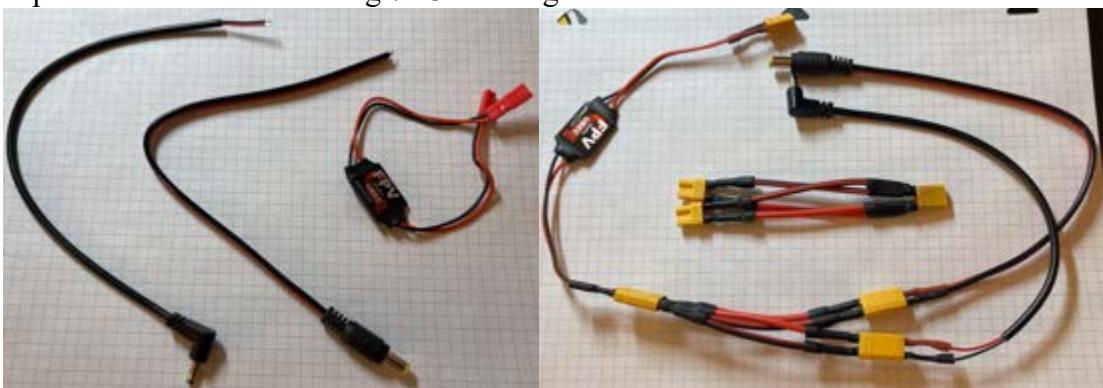
SIYI provided an instruction to enable UDP telemetry with HM90

<https://drive.google.com/drive/folders/1zr9lFAjsrGN-7Apwi9RoV9aQO2ZNM9Kh>

It shows an advanced firmware (0.2.4) can be installed through web interface, rather than USB.

Web interface's default admin passwords are missing. Stuck at login. Requested support to SIYI for default admin passwords.

Power connectors arrived for on-board electronics. Utilized XT30 plugs to make customer power connector according 7/23 drawing.



Photographs taken by the author

5V voltage-down regulator connects to power distribution board from 4S battery (16V). 5V voltage is separated to 1) ethernet adapter and 2) JETSON nano adapter. Confirmed both ethernet switch and JETSON nano are powered up with the connectors.

The other XT30 splitter is for 4S battery voltage between 1) HM30 air unit and 2) A8 mini gimbal. Need to test it.

A8 mini gimbal can record up to 4K 3840 x 2160 video to its own SD card in the slot, but it can send FHD 1920 x 1080 video only. So, if JETSON uses gimbal video or image only, it will be limited to FHD resolution only. Most of gimbals are limited FHD resolution

To have more resolution, need to enable hard-wired camera, such as Raspberry Pi camera module 3 with 12MP resolution.

<https://www.raspberrypi.com/products/camera-module-3/>

Raspberry Pi camera module 3 will need a cable and mounting fixture



Images taken from

Adafruit (<https://www.adafruit.com/>)

This fixture looks useful for a downward camera mounting

<https://www.adafruit.com/product/5721>

This fixture is useful for forward camera mounting

<https://www.adafruit.com/product/1434> (currently out of stock)

https://www.digikey.com/en/products/detail/adafruit-industries-llc/1434/10670006?so=82514583&content=productdetail_US&mkt_tok=MDI4LVNYSy01MDcAAAGNR9uqr_LdeMdLW1eqCwTBD5x-kH728ztgBy7SN_79MoDkWKfjshHZEZMV17ScA4eP_C7Ka_N7NAsYWRuCQqAvVtTiSJG_GACoCAk3R (back-ordered at Digikey)

Motor survey table was updated with more data points from web:

	Model #	Motor size	KV	Idle current (A)	Rated voltage	Rated thrust (Kg)	50%				Max				Propeller	Weight (g)	Thrust /Weight	50% Thrust /Power	Max Thrust /Power
							Current (A)	Voltage (V)	Thrust (Kg)	Power (W)	Current (A)	Voltage (V)	Thrust (Kg)	Power (W)					
Holybro	AIR2216B	2216	920	0.8	45	1.3	3.6	16	0.447	58	16.37	16	1.332	260	10	21	63.4	7.7	5.1
T-MOTOR	V807	170															670		
T-MOTOR	U15 II	80															1740		
T-MOTOR	MN7005	7005	115			1.5													
T-MOTOR	MN4004	4004	300	0.2	4 - 65		2.1	24	0.59	50	10.2	24	1.3	245	13 - 17	53	24.3	11.7	5.3
T-MOTOR	MN4004	4004	400	0.2	4 - 65		2.2	24	0.57	53	10.4	24	1.5	250	13 - 14	53	28.7	10.7	6.1
T-MOTOR	MN4006	4006	380	0.3	4 - 65		3.7	24	0.93	89	17.5	24	2.3	420	13 - 16	68	34	10.4	5.5
T-MOTOR	MN5006	5006	300	0.6	4 - 65	1	3.7	23.6	0.97	87	19.6	23.1	3	453	17 - 18	108	27.7	11.1	6.6
T-MOTOR	MN5006	5006	450	0.9	4 - 65	1	5.4	23.5	1.08	127	26.3	22.9	3.2	601	14 - 15	106	30.4	8.5	5.4
T-MOTOR	MN5006	5006	450		4 - 65		4.8	15.6	0.85	75	24.8	15	2.6	371	17 - 18	106	24.6	11.4	7
T-MOTOR	MN5008	5008	170	0.4	6 - 125	1.2	2.7	47.4	1.34	130	14.6	46.9	4.1	687	17 - 18	128	32	10.3	6
T-MOTOR	MN5008	5008	340	0.9	65		6.9	23.3	1.54	162	33.5	22.5	4.2	794	17 - 18	135	31.2	9.5	5.6
T-MOTOR	MN5008	5008	400	1.1	65		7	23.3	1.45	163	32.8	22.5	4	740	15 - 17	132	30.2	8.9	5.4
T-MOTOR	MN5208	5208	340		4 - 65		5.9	24	1.32	141	31.1	24	4.1	747	16 - 18	145	28.4	9.4	5.5
T-MOTOR	MN5212	5212	340	1.1	4 - 65		5.7	24	1.32	138	31	24	4.4	745	15 - 18	205	21.2	9.6	5.8
T-MOTOR	MN5212	5212	420	1.1	4 - 65		12.5	24	2.11	297	59	24	5.9	1437	15 - 18	205	28.9	7.1	4.1
T-MOTOR	MN6007	6007	160	0.5	125	2	5.4	47.7	2.47	257	23.8	47.3	6.4	1123	21 - 22	172	37.3	9.6	5.7
T-MOTOR	MN6007	6007	320	1	65		10.7	24.5	2.43	262	44.2	23.6	5.9	1041	21 - 22	159	37	9.3	5.7
T-MOTOR	MN7005	7005															161		
T-MOTOR	MN8012	8012																	
T-MOTOR	MN8014	8014																	
T-MOTOR	MN8017	8017																	
T-MOTOR	MN4120	4120	400	1.2	4 - 85														

Table created by the author

Need to use 50% thrust to calculate necessary thrust and weight for the drone.

Bladeless drone for safety

[Bladeless Drone: First Flight](#)



Image taken from "Bladeless Drone: First Flight"

(<https://www.youtube.com/watch?v=5L6FSdUmEpg>)

08/03/23

System diagram for patents

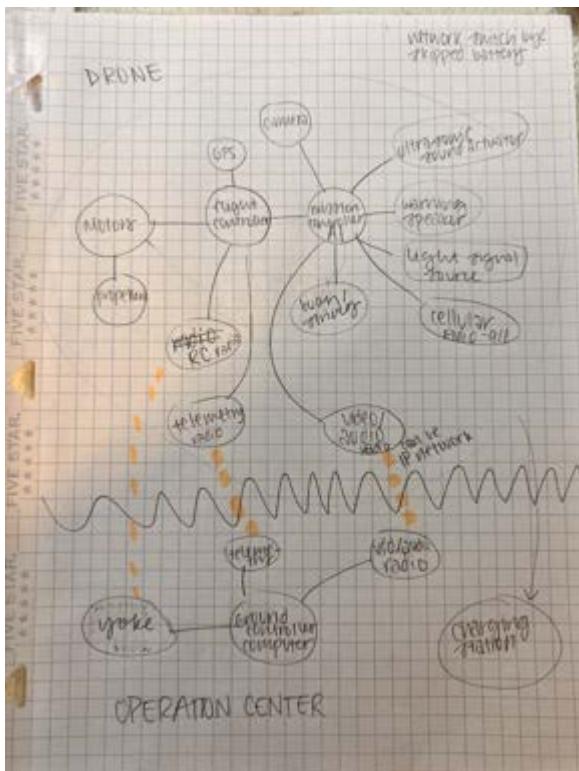
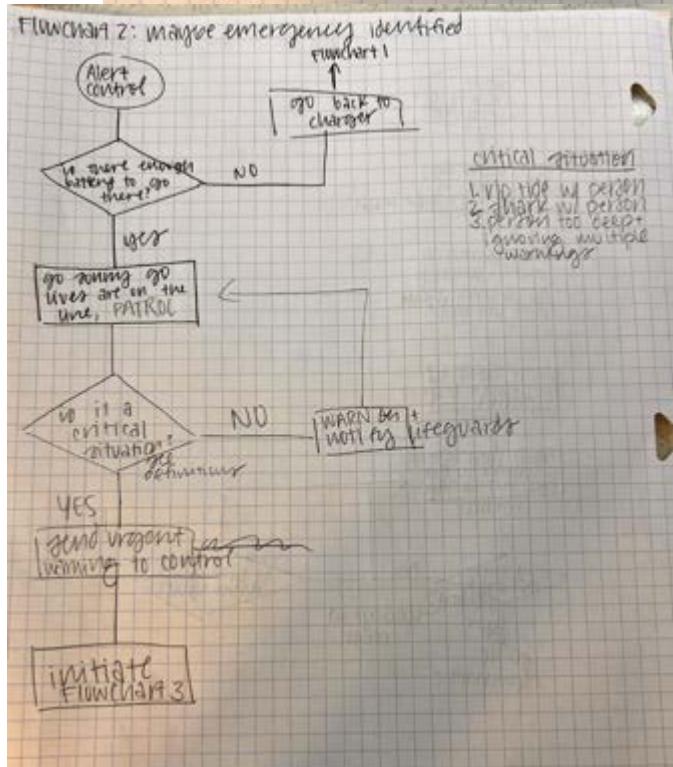
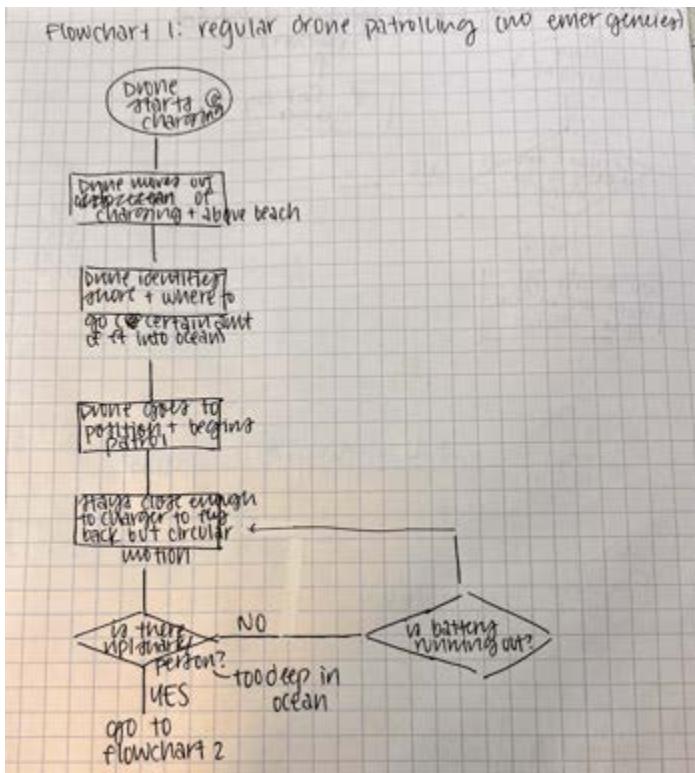


Diagram created and photograph taken by
the author

Flowchart for process:



Diagrams created and photographs taken by

the author

08/04/23

New materials arrived from Adafruit



Photograph taken by the author

- RPi camera module mount brackets
- RPi camera extension cables
- NoIR (No InfraRed filter) RPi camera for IR imaging

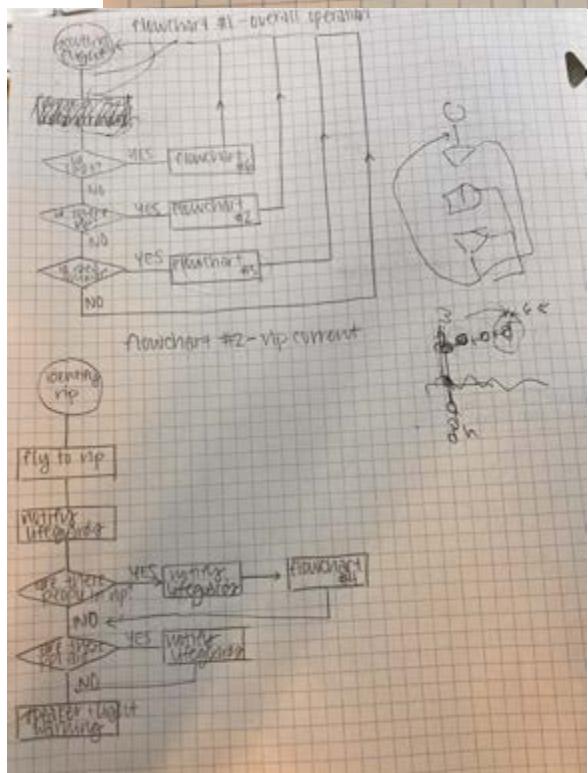
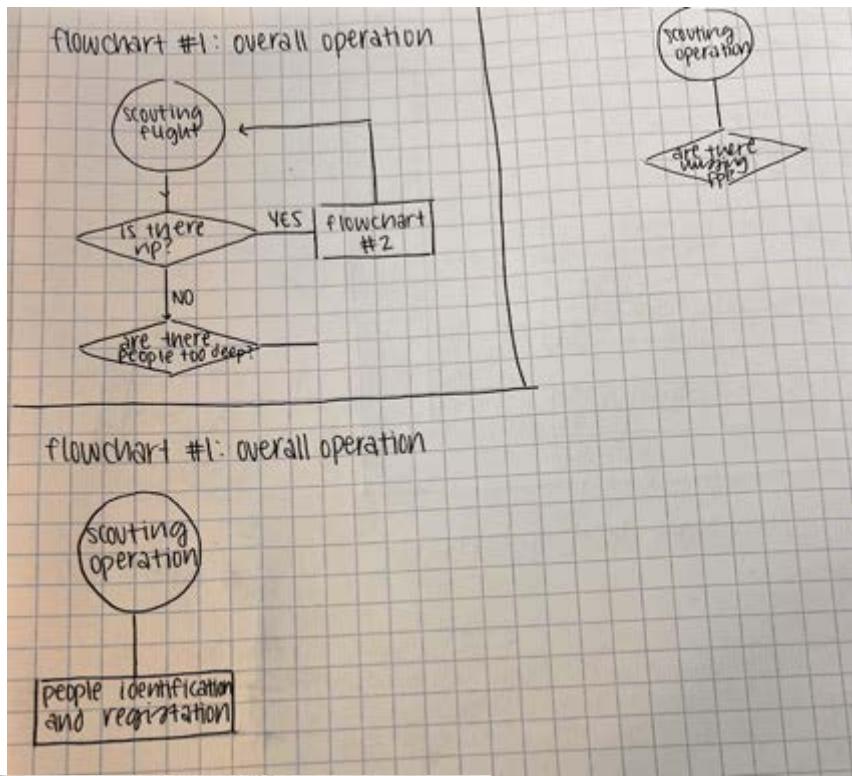
HM30 telemetry debugging

- SIYI emailed a confidential http web admin login password for firmware update
- Backed up existing firmwares through the SIYI Windows program
- Updated air unit and ground unit firmwares through http web interface
- Initial ethernet UDP test at Windows 11 laptop and QGroundControl did not work
- WiFi connection to HM30 was enabled. UDP still does not work.
- Confirmed the firmware versions with SIYI for correct configuration for UDP
 - HardID:6A02251757
 - FirmwareVer:0.1.9
 - SkyFirmareVer:5.2.4
 - **VideoVer:0.2.4 (this was updated)**
 - **SkyVidoeVer:0.2.2 (this was updated)**
 - LoaderVer:0.1.0
- Need to test UDP with Linux

08/05/23

Updated system flow chart:

Overall operation



Diagrams created and photographs taken by the author

Rescue operation

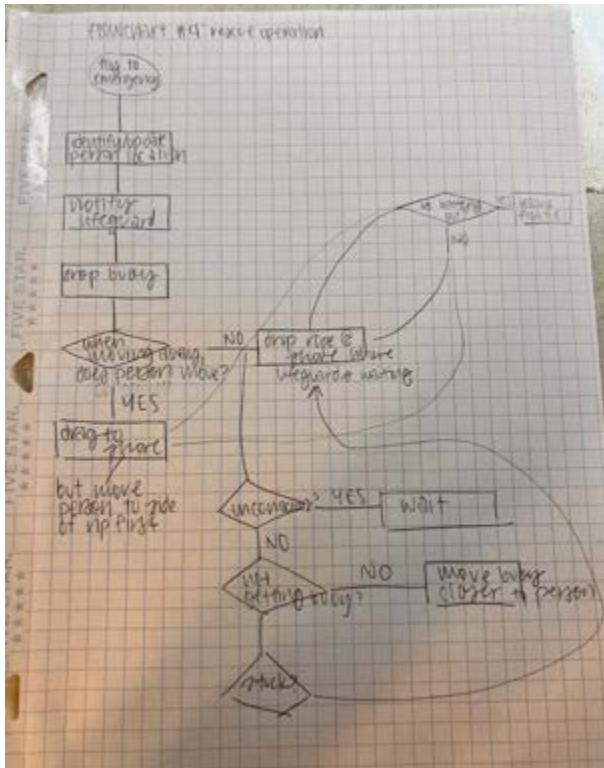
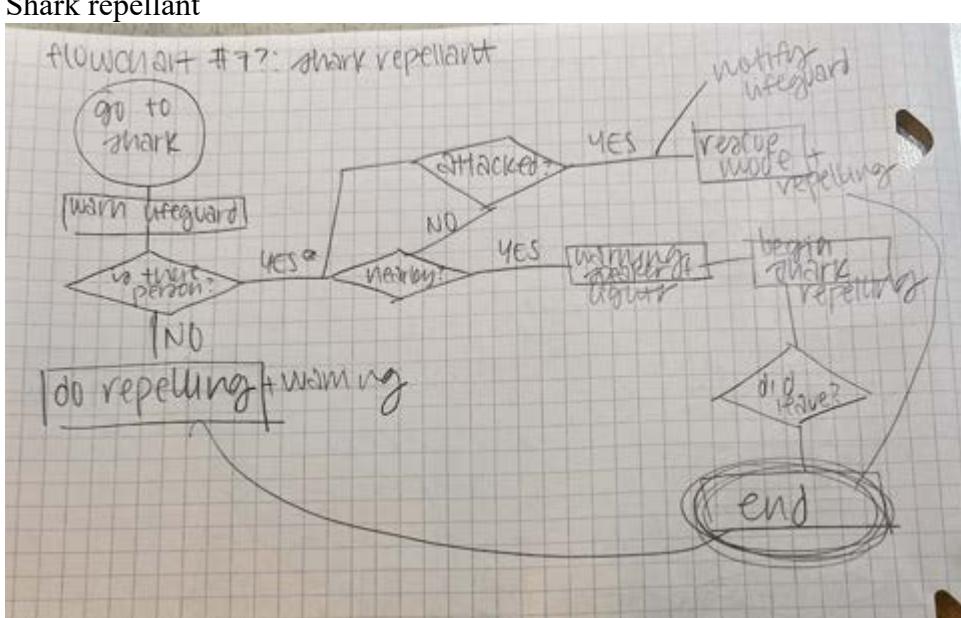


Diagram created and photograph taken by the author

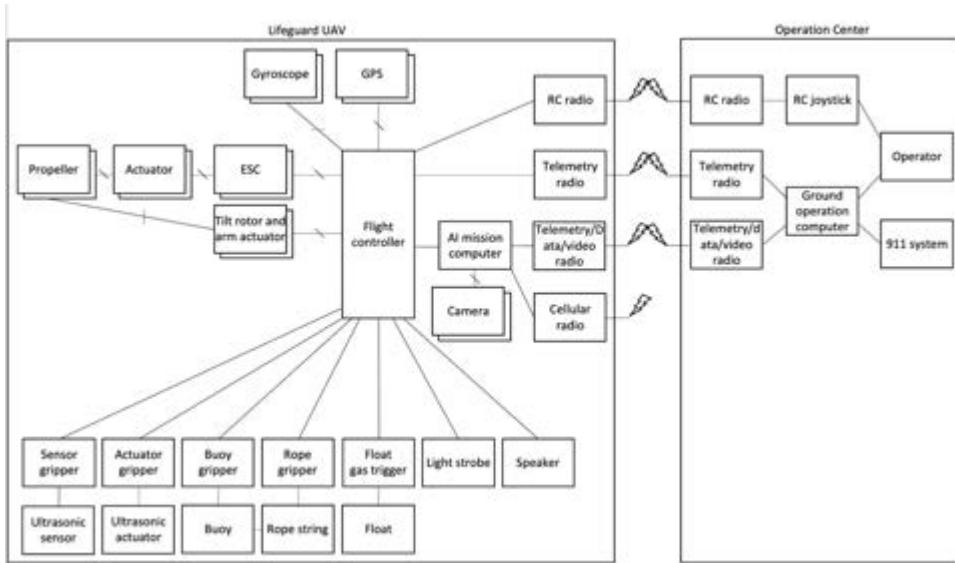
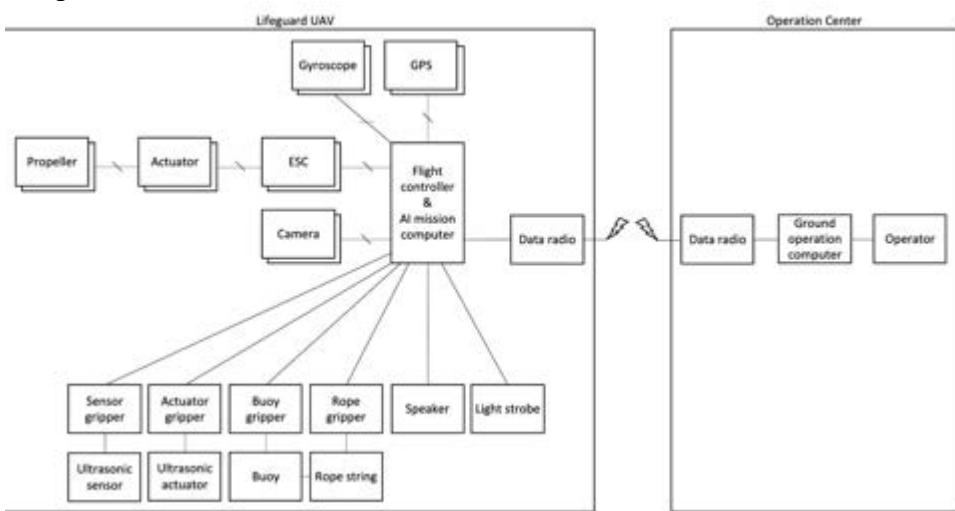
Shark repellent



Diagram

created and photograph taken by the author

Block diagram of system:

*Diagram**created by the author**Simplified:**Diagram**created by the author*

- f5efe4

08/06/23

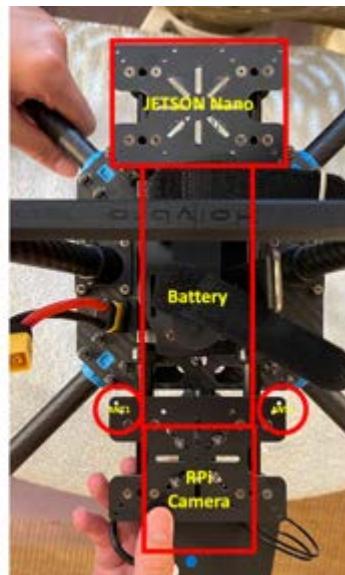
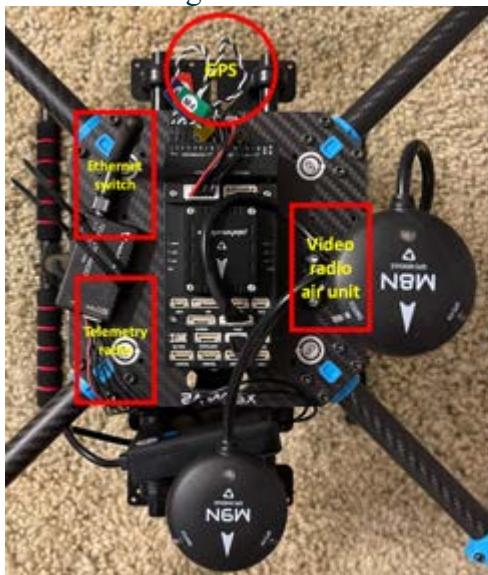
Beach outing documentation preparation. Laminated copies:

- FAA Small UAS Certificate of Registration, #FA3HRKM494
- The Recreational UAS Safety Test (TRUST) Completion Certificate, #RECF29679944364
- Amateur radio license, Technician, call sign N6SEA



Photograph taken by the author

Placement diagram



Photographs taken by the author

Getting longer bars (outer diameter 9.9mm, inner diameter 7.75mm) to make more vertical room for battery, gimbal, and camera in right photo

Moving GPS to different location (left photo) and attaching air unit, moving telemetry radio, and replacing where telemetry is now with Ethernet switch

Antennae locations in right photo.

Need to lower cameras to avoid landing legs in the video and image capture.

Wind resistance levels for drones:

Wind Resistance Level	Wind Speed	Indicators/effects
0	< 1 mph	Complete calmness
1	1-3 mph	Can move smoke in wind direction
2	4-7 mph	The wind felt on face, moves leaves
3	8-12 mph	Moves leaves and twigs
4	13-18 mph	Move dust off the ground
5	19-24 mph	Move small trees
6	25-31 mph	Sway large tree branches
7	32-38 mph	Resistance in walking against the wind
8	39-46 mph	Breaks twigs & small branches
9	47-54 mph	Blow slate from rooftops
10	55-63 mph	Break trees (a rare occurrence)
11	64-72 mph	Widespread damage (a rare occurrence)
12	> 73 mph	Destruction of structure(a rare occurrence)

Image taken from FlyThatDrone

(<https://flythatdrone.com/blog/drone-wind-resistance-levels-explained/>)

<https://flythatdrone.com/blog/drone-wind-resistance-levels-explained/>

Tilt rotor can more effectively overcome wind by minimizing body drag

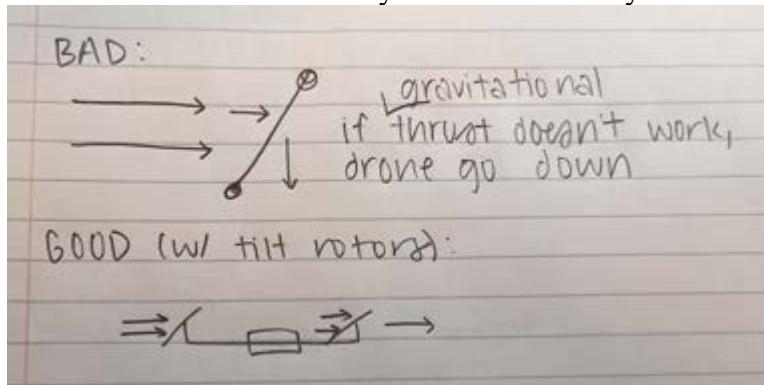


Diagram created and

photograph taken by the author

Flow charts updated

Flowchart #1
Overall operation

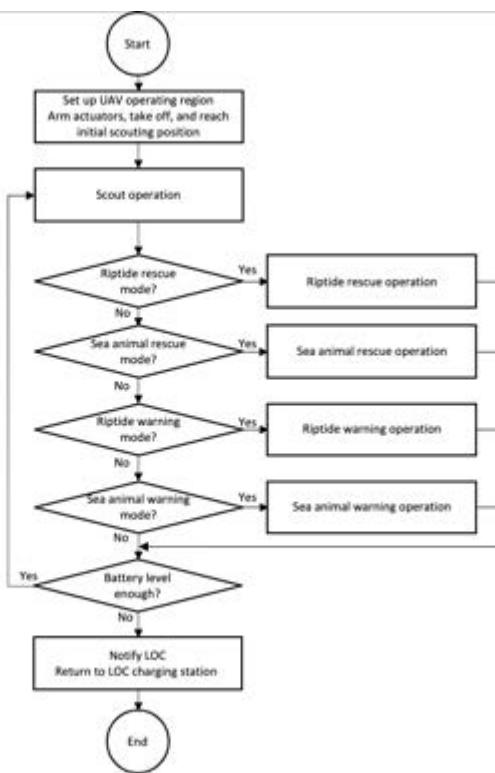


Diagram created by the author

author

Flowchart #2
Scouting operation

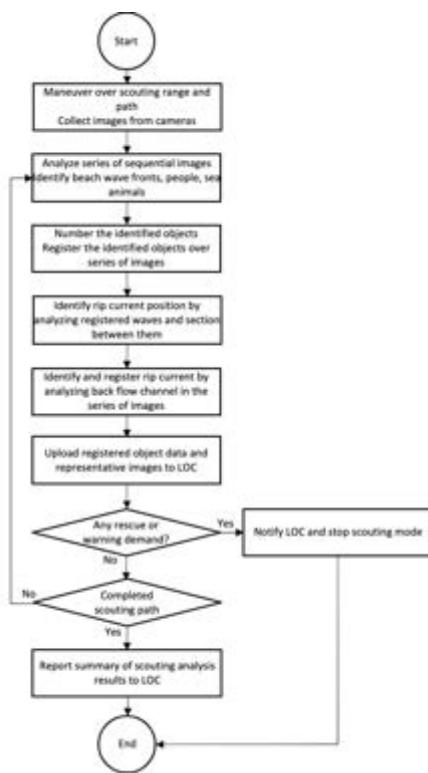


Diagram created by the author

Flowchart #3
Rip current rescue mode



Diagram created by the author

Flowchart #4
Sea animal rescue mode

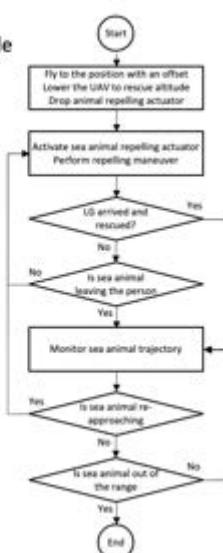


Diagram created by the author

08/07/23

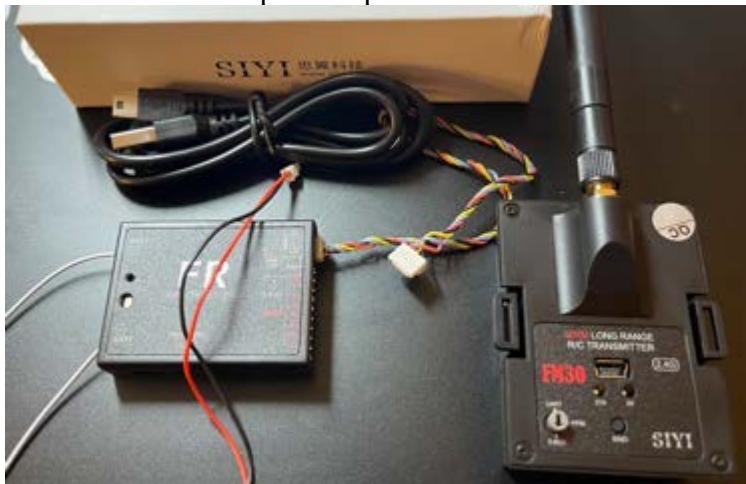
HM30 telemetry try at Linux

- Tried the update firmware at Ubuntu

- QGroundControl does not have connectivity to PX4
- So, it does not work. Need to communicate with vendor further
- For now, HM30 is useful for video and IP data transport only

FM30 telemetry radio arrived

- Ground unit can be powered up through USB port
- Air unit's 2 contact GH1.25 does not power up when connected to 4S battery
- Air unit is powered up through USB port, and it seems to be bonded with ground unit
- Need to make USB power supply connector
- Ground unit does not allow Linux to connect to Bluetooth
- At Windows, Linux, and Mac, Bluetooth is connected temporarily, but disconnected in a few seconds. Inquired with vendor.
- Need USB cable to power up FR receiver.



Photograph taken by the author

08/08/23

FM30 telemetry radio debug

- Confirmed firmware is the latest
- bluetoothctl program and trust, pair, and connect commands are not working well

08/09/23

Test drive Raspberry Pi Camera Module 3 on JETSON

<https://jetsonhacks.com/2019/04/02/jetson-nano-raspberry-pi-camera/>

- Initial run did not work. JETSON uses IMX219 sensor as a default interface to camera. It is compatible with Raspberry Pi Camera Module 2. But not with Camera Module 3.
- Run IO configuration utility
 - https://docs.nvidia.com/jetson/archives/l4t-archived/l4t-3273/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/hw_setup_jetson_io.html#wpID0E02D0HA
 - sudo /opt/nvidia/jetson-io/jetson-io.py
- Changed CSI to IMX477 dual, saved and rebooted

- akim@nano:~\$ gst-launch-1.0 nvarguscamerasrc ! Nvoverlaysink
- It seems to run, but image is blank.
- Other gstreamer commands at port 0 has the same results – blank screen, while gstreamer functions
- Search shows that JETSON NANO and ORIN NANO do not support Camera Module 3 yet.
- Until a solution is found, will need to back off to Camera Module 2 (8 mega pixels)
- Reverted CSI to IMX219
- Found this update
- <https://forums.developer.nvidia.com/t/ridgerun-rpi-camera-module-3-imx708-open-access-driver-for-nvidia-jetson-orin-nano-and-jetson-nano/261118>
- https://developer.ridgerun.com/wiki/index.php/Raspberry_Pi_Camera_Module_3_IMX708_Linux_driver_for_Jetson#Get_the_driver_patches
- Test sequence
- Display
 - SENSOR_ID=0
FRAMERATE=14
gst-launch-1.0 nvarguscamerasrc sensor-id=\$SENSOR_ID ! "video/x-raw(memory:NVMM),width=4032,height=3040,framerate=\$FRAMERATE/1" ! queue ! Nv3dsink
- JPEG snapshots
 - SENSOR_ID=0
FRAMERATE=14
gst-launch-1.0 nvarguscamerasrc num-buffers=1 sensor_id=0 ! 'video/x-raw(memory:NVMM), width=4608, height=2592, framerate=14/1, format=NV12' ! nvjpegenc ! filesink location=RidgeRun_test.jpg
- MP4 recording
 - SENSOR_ID=0
FRAMERATE=14
gst-launch-1.0 nvarguscamerasrc sensor_id=\$SENSOR_ID -e ! 'video/x-raw(memory:NVMM),width=4608,height=2592,framerate=\$FRAMERATE/1, format=NV12' ! nvvidconv ! "video/x-raw,width=1920,height=1080" ! x264enc ! qtmux ! filesink location=RidgeRun_out.mp4
- Eventually, worked with the driver installation
- Mount kit will be assembled later after cable length is confirmed

RPi CM3 mounting kit and JPEG screen capture with CM3



Photographs taken by the author

Connect Pixhawk 6X flight controller to JETSON Nano for telemetry and MAVLink
<https://www.hackster.io/Matchstic/connecting-pixhawk-to-raspberry-pi-and-nvidia-jetson-b263a7>

- Need a dedicated cable from telemetry to jumper pins
- Getting “link 1 down” message.
- Need remove ModemManager
- <https://www.elucidatedrones.com/posts/how-to-install-mavproxy/>
- Maybe connection is not good?
- Pin connection checks shows 0ohm at all three wires
- Need to check if TELEEM Port is working?
- Pixhawk default bit rate is 57600
 - <https://ardupilot.org/plane/docs/common-mission-planner-bluetooth-connectivity.html#:~:text=The%20default%20Baud%20Rate%20for,red%20LED%20when%20not%20connected.>
- JETSON UART default rate is 115200. Need to change
- To review bit rate set up
 - sudo stty -F /dev/ttyTHS1
 - Current speed is 115200 baud

08/10/23

JETSON to Pixhawk connection

- Need to change baud rate to 57600
 - <https://forums.developer.nvidia.com/t/how-do-i-change-the-baud-rate-please/40602/3>
- Installed setserial package
- setserial /dev/ttyTHS1 baud_base 57600
 - akim@nano:~\$ sudo setserial -a /dev/ttyTHS1
 - /dev/ttyTHS1, Line 1, UART: undefined, Port: 0x0000, IRQ: 64
 - Baud_base: 0, close_delay: 50, divisor: 0
 - closing_wait: 3000
 - Flags: spd_normal
 -
 - akim@nano:~\$ sudo setserial /dev/ttyTHS1 baud_base 57600
 - akim@nano:~\$ sudo setserial -a /dev/ttyTHS1
 - /dev/ttyTHS1, Line 1, UART: undefined, Port: 0x0000, IRQ: 64

- Baud_base: 57600, close_delay: 50, divisor: 0
 - closing_wait: 3000
 - Flags: spd_normal
- Still not working.
- Need to use sudo command.
- sudo adduser akim tty
- Still not working
- No error during boot
 - dmesg | grep serial
- sudo mavproxy.py --master=/dev/ttyTHS1 --baudrate=57600
- Need to check if Pixhawk telemetry is working
 - https://docs.px4.io/main/en/peripherals/serial_configuration.html
- When connected to PX4
 - MAV_0_CONFIG=101 for TELEM1
 - MAV_1_CONFIG=0 -> 102 for TELEM2
 - MAV_2_CONFIG=1000 for Ethernet
 - But I do not get any MAV_1* related set up parameters
- Tried JETSON connection with TELEM1
 - Still not working
 - MAV_0_RATE was 1200. Maybe the rate did not match with JETSON?

08/11/23

Dual GPS set up

- Pixhawk debugging through USB port. No battery connected for safety.
- https://docs.px4.io/main/en/gps_compass/#dual_gps
- Need to enable it. Attaching it does not enable automatically
- Use GPS2 port
- Connected Pixhawk through USB
- Firmware version is 1.13.2
- GPS_2_CONFIG was disabled. Now set to GPS2. Reboot
- Now GPS2 seems to work. Analyze Tools – MAVLink Inspector – GPS2_RAW has real-time updates
- GPS1 M8N's SWITCH red LED light blinking twice. Not sure why. Need user manual or inquire vendor

New batteries are available. They are battery #3 and #4. Charged and labeled them.

Drone ID from Cube arrived

Payload extension carbon fiber tube arrived

Installed Autodesk Fusion 360 as the mechanical design tool

Continuing MAVLink debug

- List of serial setup
 - cat /proc/cmdline
- USB connection from Pixhawk to JETSON
- Is shows up at USB device list
 - akim@nano:~\$ lsusb
 - Bus 002 Device 002: ID 0bda:0411 Realtek Semiconductor Corp.

- Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
- **Bus 001 Device 007: ID 3185:0035**
- Bus 001 Device 003: ID 25a7:fa61
- Bus 001 Device 002: ID 0bda:5411 Realtek Semiconductor Corp.
- Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
- But ls -l /dev/ttyUSB* does not show
- Pixhawk companion setup instruction:
- https://docs.px4.io/main/en/companion_computer/pixhawk_companion.html
- RPi example
- https://docs.px4.io/main/en/companion_computer/pixhawk_rpi.html

08/12/23

Patent submission at LegalZoom

- Finalized draft. Filled the forms as much as possible.
- Submitted order as #78705917
- Submitted drawings and original application files

08/13/23

08/17/23

SIYI 2.4G Datalink arrived

- It did not work at first power up
- Adjusted SER1_BAUD_RATE=115200
- Still does not work

08/21/23

First meeting with patent attorney for non-provisional utility patent filing

- Goal to file by 11/30

Things to do for beach data collection

#	Item	Beach data collection priority	New drone dev
1	Enable RPi camera forwarding to ground station	2	1
2	Enable local network login to enable remote JETSON login	1	1
3	JETSON connection to FC for MAVLink	2	1
4	Log GPS coordinates to images	2	1
5	Measure payload post length needed	1	1
6	Enable drone remote ID	1	1
7	Mount gimbal and RPi camera	1	1
8	Open park test ride	1	1

9	Flight path planning tool	1	1
10	Data collection checklist	1	1
11	Payload rod extension	1	1

Table created by the author

#2 enable local network login to enable remote JETSON login

- There are two different ways:
- #1 connect to JETSON through SIYI HM30 data network
 - It has wide range of communication as far as HM30 works
 - Login with VNC from laptop, to preserve operations and commands running while disconnected
 - It can compete with datalink capacity
- #2 connect to JETSON WiFi
 - Need to set JETSON as WiFi access point
 - Login with VNC from laptop
 - It has narrow range operation. Will be disconnected after take off
- Need to enable both
 - #2 as main
 - #1 as a back up

#5 and 7 mount camera system

- Gimbal might be able to mount directly to payload panel
- RPi camera need to mount below JETSON Nano
- JETSON Nano height is about 31mm with M2.5 holes
- Combination of M2.5 plastic stand off will work
 - Plastic is preferred than metal
 - Weight and stress

08/22/23

Drone flight checklist

Phase	Items	Check	Note
Prepare	Checklist		
	Drone		
	Drone batteries		
	Folding table		
	FAA and FCC documents		
	Recoding camera		
	Linux laptop		
	Propellers		
	Screw drive, hex wrench		
	Safety goggle		
Preflight	QGroundControl offline map		
	Set table		
	Set laptop		

	Set HM30 ground radio on tripod		
	Secure surrounding safety boundary		
	Create flight plan		
Flight	Install drone battery		
	Power up drone		
	Upload flight plan		
	VNC login to JETSON		
	Set recording folder		
	Start JETSON recording		
	Confirm safety		
	Arm actuators		
	Take off		
	Start flight path		
	Land drone		
	Disarm actuators		
	Power off drone		
	Disconnect battery		
Postflight	Maintain drone		
	Check images recorded		
	Review procedure		

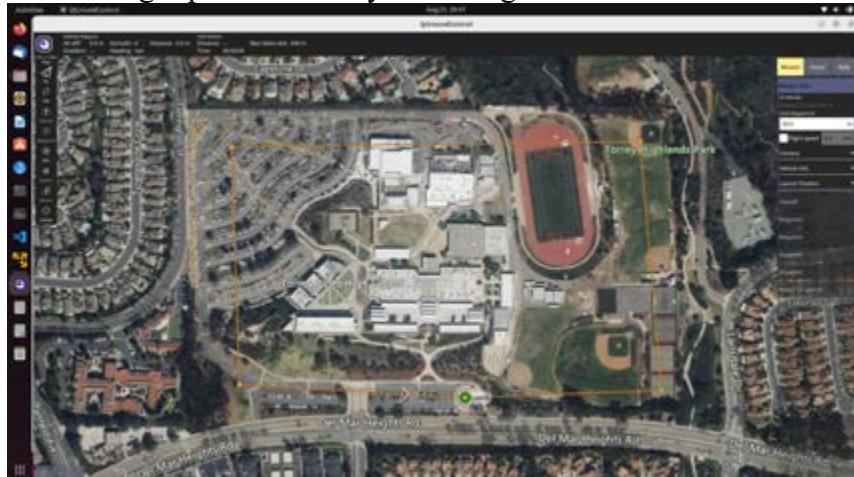
Table created by the author

QGC offline map

- Downloaded Del Mar Beach and Torrey Pines High School area for drone operation
- About 550MB data
WiFi connection is not available when ethernet cable is not connected
- Ethernet is available, but it is not connected once booted

08/23/23

Flight plan for Torrey Pines High School

*Image captured by the author*

- There is no guarantee that telemetry data connection will be effective during flight.
- School has many buildings that might block radio wave propagation

- Drone will initiate safety procedures if telemetry is out of range.

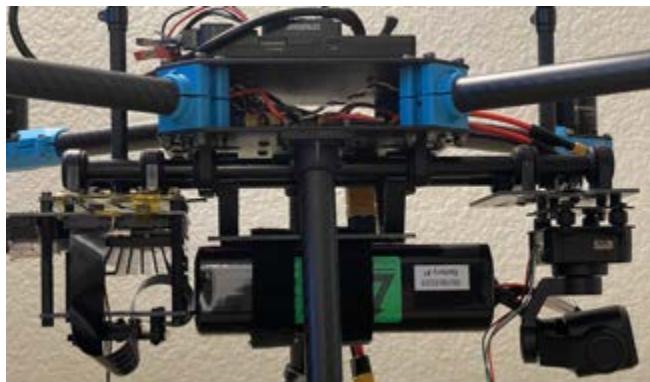
08/26/23

Asked patent attorney to proceed with drawing

08/27/23

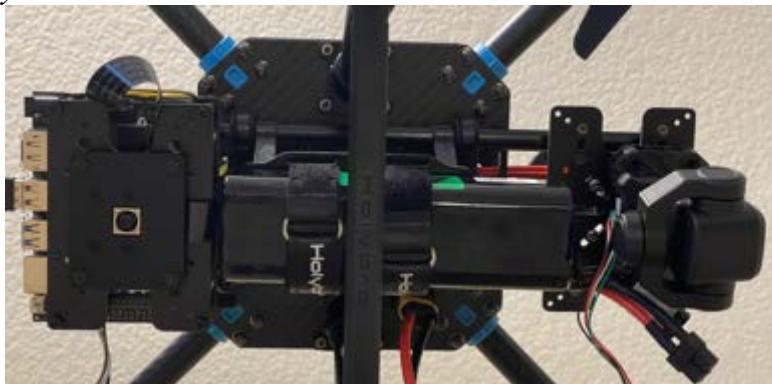
Camera mounting and drone customization

- Move GPS1 to the top of backward payload mount
- Mounted JETSON Nano on the backward payload mount
- Cross mounted GPS2 and gimbal at forward payload mount
- Gimbal is free hanging through rubber mounts
- Confirmed the battery fitting between JETSON nano and gimbal
- Shortened the JETSON RPi camera cable to 200mm from 300mm
- Rolled JETSON Nano WiFi antenna around backward payload mount
- Mounted RPi camera module 3 at the bottom of JETSON Nano
- Removed Telemetry radio for top surface rearrangement later
- Confirmed JETSON operation
 - External power supply, mouse, and display
 - Camera is upside down. Need to reverse
 - WiFi working well
 - Bottom view camera has clear view without obstruction from landing legs



Photographs

taken by the author



Photograph taken by the

author

Backward payload camera looking at bottom of the drone



Photograph taken by the author

Lidar distance sensor

- <https://holybro.com/products/st-vl53l1x-lidar>
 - <https://docs.px4.io/main/en/sensor/rangefinders.html>
 - <https://ardupilot.org/copter/docs/common-vl53l0x-lidar.html>
 - Connected to I2C port
 - It does not publish to QGC yet
 - Enabled SENS_EN_VL53L1X
 - https://docs.px4.io/v1.13/en/advanced_config/parameter_reference.html#SENS_EN_VL53L1X
 - Now the distance sensor works
 - Ground distance from bodyside is 19cm

Ethernet switch case was taken out to reduce volume and weight

- Need zip tie to fix the ethernet switch at landing gear



Photograph taken by the author

Remote ID setup



Photograph taken by the author

- Connected Cube ID at TELE2
- https://docs.px4.io/main/en/peripherals/remote_id.html
- MAV_1_CONFIG = TELE2
- MAV_1_MODE = Normal
- MAV_1_RATE = 0 (default sending rate for port).
- MAV_1_FORWARD = Enabled
- SER_TEL2_BAUD=57600
- Might need firmware 1.14. Current firmware is 1.13.2
- Not sure how to download 1.14 or need to compile?
- <https://docs.px4.io/main/en/releases/1.14.html>
- <https://github.com/PX4/PX4-Autopilot/releases>

08/28/23

Need to install PX4 package to install experimental firmware, which enables remote ID

- https://docs.px4.io/main/en/contribute/git_examples.html
- sudo apt install git
- git clone <https://github.com/PX4/PX4-Autopilot.git>
- git checkout v1.14.0-rc1
 - This is release candidate v1.14.0
 - <https://docs.px4.io/main/en/releases/1.14.html>
 - <https://github.com/PX4/PX4-Autopilot/releases>
- make px4_fmu-v6x_default
 - There is an error with cmake
- git submodule update --recursive
- make distclean
 - Still the same error
- Reinstall
- git clone https://github.com/PX4/PX4-Autopilot.git --recursive
- cd PX4-Autopilot
- make clean
- make distclean
- make submodulesclean
- Need to install cmake
- sudo apt install cmake
- make px4_fmu-v6x_default
- ModuleNotFoundError: No module named 'menuconfig'
- pip3 install kconfiglib

- [make px4_fmu-v6x_default](#)
 - These are missing
 - arm-none-eabi-g++
 - arm-none-eabi-gcc
 - arm-none-eabi-gcc
- Need to build tool chain
- bash ./PX4-Autopilot/Tools/setup/ubuntu.sh
- [make px4_fmu-v6x_default](#)
- Compilation worked out
- Need to upload/upgrade PX4 firmware
- Firmware upload/upgrade was successful



Image captured by the author

- Unfortunately, everything was reset. Need to reconfigure items configured before
- Enable GPS2: GPS_2_CONFIG
- Enable LIDAR distance sensor: SENS_EN_VL53L1X
- Remote ID:
 - MAV_1_CONFIG = TELE 2
 - Need to restart
 - MAV_1_MODE = Normal
 - MAV_1_RATE = 0 (default sending rate for port).
 - MAV_1_FORWARD = Enabled
 - SER_TEL2_BAUD=57600
- Working
 - GPS
 - LIDAR
- OPEN_DRONE_ID_* parameters are not showing up
- Might need official v1.14.0 release. RC1 might not have remote ID enabled.

- OPEN_DRONE_ID_* are messages, not parameters
- There is 1Hz OPEN_DRONE_ID_* messages.

		Message Component	OPEN_DRONE_ID_LOCATION (1280x1194)
		Count	84
1	GLOBAL_POSITION_INT	1.000	
2	GPS_RAW	0.000	
3	GPS_GLOBAL_ORIGIN	0.000	
4	GPS_RAW_INT	1.000	
5	HEARTBEAT	1.000	
6	HIGHRES_IMU	0.000	
7	HOME_POSITION	0.000	
8	LINE_FOLLOW_STATUS	1.000	
9	LOCAL_POSITION_NED	0.000	
10	DISTANCE	0.000	
11	OPEN_DRONE_ID_LOCATION	1.000	
12	PING	1.000	

Image captured by the author

Trying to setup ID

Console commands

https://dev.px4.io/master/en/middleware/modules_command.html

- param set MAV_ODID_ID_TYPE 1
- **ERROR** [param] Parameter MAV_ODID_ID_TYPE not found.
Need to set up ID and request through MAVLINK
- <https://mavlink.io/en/services/opendroneid.html>
- OPEN_DRONE_ID_BASIC_ID
- https://mavlink.io/en/messages/common.html#OPEN_DRONE_ID_BASIC_ID

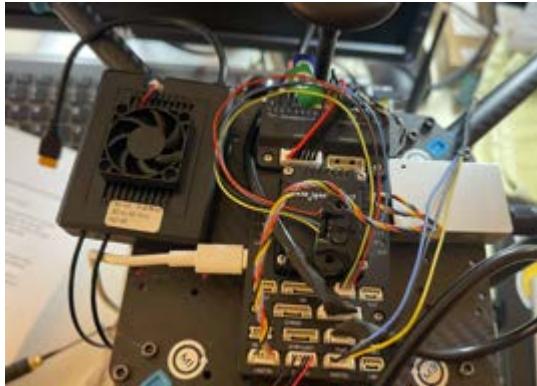
08/29/23

SIYI 2.4GHz Datalink 2nd try

- Set parameters according to manual
- https://drive.google.com/file/d/1AsQwuPGYdiO9NSQ4l5CXppwzfw_F6Rdc/view?usp=sharing
 - MAV_0_*
 - SER1_BAUD_RATE=115200
- Set QGC with ttyUSB0 connection
- It does not work
- Try firmware update
- https://drive.google.com/drive/folders/1UHKn-tGWBfaHdi6-1vckj2sDBNpf_2G9
- Version (0.1.4) are same as current version
- There 57600 and 115200 version
- Installed 57600 airunit and ground unit firmwares
- Use SIYI PC Assistant software
- Reduce data rate and reboot
 - SER1_BAUD_RATE=57600
- Now the telemetry connection works!!!

Drone assembly

- USB-C port blocks IP network air unit placement



Photograph taken by the author

- Need 90-degree angle adapter
- Still IMU debug port is somewhat blocked – it should be okay for now.
- Things to install
 - Ethernet switch
 - 2.4GHz telemetry datalink and antenna
 - IP network air unit and antenna
 - Distance sensor
 - Remote ID transponder

MAVLink onboard physical connection

- Two options
 - UART serial link, which failed to establish so far
 - <https://jetsonhacks.com/2019/10/10/jetson-nano-uart/>
 - <https://www.hackster.io/Matchstic/connecting-pixhawk-to-raspberry-pi-and-nvidia-jetson-b263a7>
 - Ethernet
 - https://docs.px4.io/main/en/advanced_config/ethernet_setup.html
- Last serial connection try, until ethernet switch is mounted
- Changed MAV_2_CONFIG = TELE3
- Reboot multiple times
- SER_TEL3_BAUD showed up. Set to 115200N1

At JETSON, following <https://jetsonhacks.com/2019/10/10/jetson-nano-uart/>

- git clone <https://github.com/JetsonHacksNano/UARTDemo>
- cd UARDemo
- systemctl stop nvgetty
- systemctl disable nvgetty
- udevadm trigger
- Reboot
- cd UARDemo
- sudo apt-get install python3-serial
- sudo python3 uart_example.py
- Output
 - UART Demonstration Program
 - NVIDIA Jetson Nano Developer Kit

- b'\xfe'
 - b'\x1e'
 - b'\xbff'
 - b'\x01'
 - b'\x01'
 -
 - b'\xf8'
 - b'\$'
 - b'\xb8'
 - b'\x10'
 - b'['
 - Exiting Program
- The serial link is working

Try MAVLink example from <https://www.hackster.io/Matchstic/connecting-pixhawk-to-raspberry-pi-and-nvidia-jetson-b263a7>
- MAVlink does not work
 - akim@nano:~\$ sudo mavproxy.py --master=/dev/ttyTHS1
 - Connect /dev/ttyTHS1 source_system=255
 - Log Directory:
 - Telemetry log: mav.tlog
 - Waiting for heartbeat from /dev/ttyTHS1
 - MAV> link 1 down
- Update baud rate to 115200
- https://ardupilot.org/mavproxy/docs/getting_started/examples.html
 - akim@nano:~\$ sudo mavproxy.py --master=/dev/ttyTHS1 --baud=115200
 - Connect /dev/ttyTHS1 source_system=255
 - Log Directory:
 - Telemetry log: mav.tlog
 - Waiting for heartbeat from /dev/ttyTHS1
 - MAV> Detected vehicle 1:1 on link 0
 - online system 1
 - LOITER> Mode LOITER
 - fence breach
 - AP: GCS connection regained
 - Received 1142 parameters
 - Saved 1143 parameters to mav.parm
- Now MAVLink through UART works
Enable VNC server at JETSON Nano
- Web references
 - <https://developer.nvidia.com/embedded/learn/tutorials/vnc-setup>
 - <https://stackoverflow.com/questions/74600691/nvidia-jetson-nano-vnc-connection-without-hdmi-plugged-in-headless>
 - <https://forums.developer.nvidia.com/t/how-to-setup-tigervnc-on-jetson-nano/174244>
- Key is to enable VNC in headless mode for drone operation
 - sudo apt-get update

- sudo apt-get upgrade
- sudo apt-get install nano
- sudo apt install tigervnc-standalone-server
- vncpasswd
- cd ~/.vnc
- sudo nano xstartup
 - !/bin/sh
 - export XDG_RUNTIME_DIR=/run/user/1000
 - export XKL_XMODMAP_DISABLE=1
 - unset SESSION_MANAGER
 - unset DBUS_SESSION_BUS_ADDRESS
 - xrdb /home//.Xresources
 - xsetroot -solid grey
 - gnome-session &
 - startlxde &
- sudo chmod 755 ~/.vnc/xstartup
- sudo touch /home//.Xresources
- cd /etc/systemd/system
- sudo nano vncserver@.service
 - akim@nano:/etc/systemd/system\$ sudo vi /etc/vnc.conf
 - akim@nano:/etc/systemd/system\$ sudo vi /etc/gdm3/custom.conf
 - akim@nano:/etc/systemd/system\$ cd
 - akim@nano:~\$ sudo systemctl daemon-reload
 - akim@nano:~\$ sudo systemctl enable vncserver@1
 - Created symlink [/etc/systemd/system/multi-user.target.wants/vncserver@1.service](#)
→ /etc/systemd/system/vncserver@.service.
 - akim@nano:~\$ sudo systemctl start vncserver@1
 - Job for [vncserver@1.service](#) failed because the control process exited with error code.
 - See "systemctl status [vncserver@1.service](#)" and "journalctl -xe" for details.
 - akim@nano:~\$ sudo systemctl enable vncserver@1
 - akim@nano:~\$ ps -ax |grep vnc
 - 8486 pts/1 S+ 0:00 grep --color=auto vnc
- After Nano reboot, JETSON is not found by RealVNC
- Ping to 192.168.1.34 fails. So network IP address is not assigned before login
- sudo vi /etc/gdm3/custom.conf
 - Added AutomaticLogin=akim
- Reboot
- Now ping works
- VNC connection is refused by JETSON

08/30/23

VNC is not working still – refused by JETSON

- Continuning error message was the screen geometry is not valid
- /etc/systemd/system/vncserver@.service
 - ExecStart=/usr/bin/vncserver :%i -depth 24 -geometry 1920×1080 -nolisten tcp

- Removed and simplified options
 - ExecStart=/usr/bin/vncserver :%i -depth 16
 - Depth is 16 to reduce color depth and data size
- Now RealVNC access works!



Image captured by the author

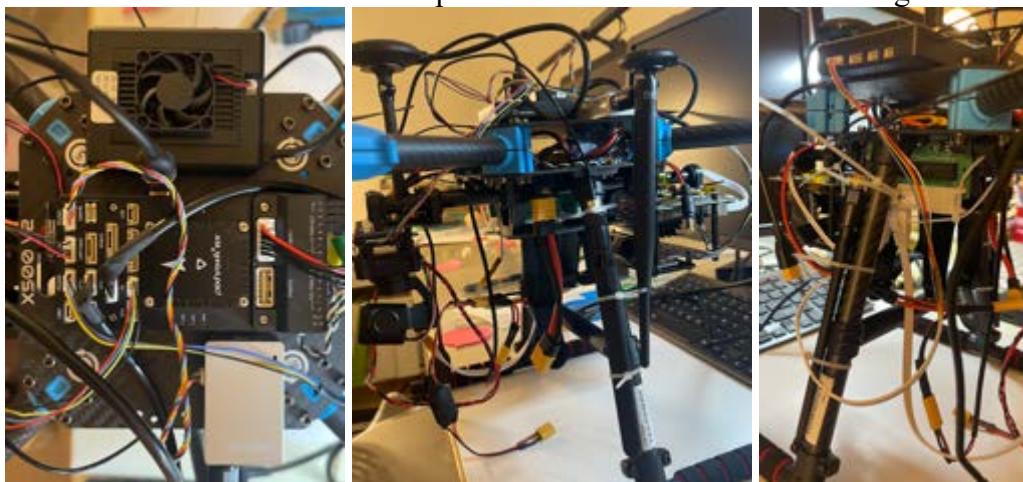
Make JETSON as WiFi hotspot to allow remote login from control station

<https://jetsonhacks.com/2021/10/05/wi-fi-hotspot-setup-nvidia-jetson-developer-kits/>

08/31/23

Mounting components

- Ethernet switch
- 2.4GHz datalink
- IP network air unit and antenna – used right angle USB-C adapter
- Power cables
- Distance sensor – double-sided tape does not stick. Screwed down at gimbal



Photographs taken by the author

Things to do

- Make XT30 splitter power cable as an extension adapter
- Attach remote ID
- Tidy up cables
- JETSON program function
- Turn gimbal to vertical position

- Setup gimbal mode
- Gstreamer
 - Collect gimbal and RPi camera images
 - Store images with index at every second
 - Forward images as video to control computer

09/01/23

Mounting

- Made one more set of XT30 splitter with a longer wire for HM30 air unit power connection
- Wire connection will be fixed after routine operation is confirmed
To do's
- Enable gstreamer through air IP network
- Install MAVLink library at JETSON
GStreamer set up
- Camera0 image saving standalone works
 - ```
gst-launch-1.0 \
 nvarguscamerasrc sensor-id=0 \
 ! 'video/x-
 raw(memory:NVMM),width=4608,height=2592,framerate=10/1,format=NV12' \
 ! tee name=camera0_original \
 ! nvjpegenc \
 ! filesink location=~/px4_video/camera0_image.jpg
```
- Camera0 display standalone with image size reduction works
  - ```
gst-launch-1.0 \
    nvarguscamerasrc sensor-id=0 \
    ! 'video/x-
        raw(memory:NVMM),width=4608,height=2592,framerate=10/1,format=NV12' \
    ! nvvidconv flip-method=2 \
    ! 'video/x-raw(memory:NVMM),format=RGBA,width=640,height=360' \
    ! queue ! nv3dsink
```
- Camera0 image saving and display together
 - ```
gst-launch-1.0 \
 nvarguscamerasrc sensor-id=0 \
 ! 'video/x-
 raw(memory:NVMM),width=4608,height=2592,framerate=1/1,format=NV12' \
 ! tee name=camera0_original \
 ! queue ! nvjpegenc \
 ! multifilesink location=~/px4_video/camera0_%d.jpg \
 ! camera0_original. \
 ! nvvidconv flip-method=2 \
 ! 'video/x-raw(memory:NVMM),format=RGBA,width=960,height=540' \
 ! queue ! nv3dsink
```
- Not sure if the video rate works well. It saves files and displays reduced image
- Camera0 WiFi transport
  - ```
gst-launch-1.0 \
```

- nvarguscamerasrc sensor-id=0 \
 ○ ! 'video/x-
 raw(memory:NVMM),width=4608,height=2592,framerate=10/1,format=NV12' \
 ○ ! nvvidconv flip-method=2 \
 ○ ! 'video/x-raw(memory:NVMM),format=RGBA,width=960,height=540' \
 ○ ! nvjpegenc \
 ○ ! rtpjpegpay \
 ○ ! udpsink host=192.168.1.32 port=5010
 ○ gst-launch-1.0 \
 ○ udpsrc port=5010 \
 ○ ! application/x-rtp,encoding-name=JEPD \
 ○ ! rtpjpegdepay \
 ○ ! jpegdec \
 ○ ! Autovideosink
- Succeeded
 

Photograph taken by the author
- Need to use space in control station. “tab” causes syntax error in gstreamer
 Merge two video sources
<https://forums.developer.nvidia.com/t/usage-of-nvcompositor-vs-videocompositor/191506>

09/02/23

IP addresses

- Control station: 192.168.144.10?
- Ground unit: 192.168.144.12
- Air unit: 192.168.144.11
- JETSON: 192.168.144.20
- Gimbal0: 192.68.144.25

Next step: Gimbal camera forwarding enablement

- Need standalone battery setup at drone to power up air unit and gimbal
- Use direct monitor connection and local key board at JETSON as first step
- Need to use VNC connection later
- After gimbal enabled, need to merge RPi camera 3 and gimbal to forward to ground control station
- Gimbal camera forwarding
- Gimbal functionality checked

- ifconfig shows eth0 at 192.168.144.20
 - Gimbal 90 degree python code works to rotate gimbal camera looking at bottom
- Gimbal0 display only works
 - gst-launch-1.0 \
 - uridecodebin uri=rtsp://192.168.144.25:8554/main.264 \
 - ! nvoverlaysink
- Gimbal image save only
 - gst-launch-1.0 \
 - uridecodebin uri=rtsp://192.168.144.25:8554/main.264 \
 - ! nvjpegenc \
 - ! multifilesink location=~/px4_video/gimbal0_%d.jpg
- Need to lower data screen refresh rate
- Giimbal forwarding only
 - gst-launch-1.0 \
 - uridecodebin uri=rtsp://192.168.144.25:8554/main.264 \
 - ! nvvidconv flip-method=2 \
 - ! 'video/x-raw(memory:NVMM),format=RGBA,width=960,height=540,format=NV12,framerate=10/1' \
 - ! nvjpegenc \
 - ! rtpjpegpay \
 - ! udpsink host=192.168.1.32 port=5010
 - gst-launch-1.0 \
 - udpsrc port=5010 \
 - ! application/x-rtp,encoding-name=JEPD \
 - ! rtpjpegdepay \
 - ! jpegdec \
 - ! Autovideosink
- Video is not showing up
- Alternatives
 - gst-launch-1.0 uridecodebin uri=rtsp://192.168.144.25:8554/main.264 !
 - nvoverlaysink
 - gst-launch-1.0 uridecodebin uri=rtsp://192.168.144.25:8554/main.264 ! nv3dsink
 - gst-launch-1.0 -v playbin uri=rtsp://192.168.144.25:8554/main.264
 - uridecodebin0::source::latency=300
 - gst-launch-1.0 rtspsrc location=rtsp://192.168.144.25:8554/main.264
 - latency=100 ! queue ! rtph264depay ! h264parse ! avdec_h264 ! videoconvert !
 - videoscale ! video/x-raw,width=640,height=480 ! Autovideosink
 - WARNING: from element /GstPipeline:pipeline0/GstURIDecodeBin:uridecodebin0: Delayed linking failed.
 - Additional debug info:
 - ./grammar.y(510): gst_parse_no_more_pads ():
 - /GstPipeline:pipeline0/GstURIDecodeBin:uridecodebin0:
 - failed delayed linking some pad of GstURIDecodeBin named uridecodebin0 to some pad of GstRtpH264Depay named rtph264depay0

- Issues
 - Gimbal output does not operate well as soon as the framerate is lower than 30
Gimbal setup changes
- Need to turn off “Video Output” option Off for best ethernet video performance
- Streaming resolution is stuck at HD (1280x760), while FHD is set. Reboot shows HD.
- GImbal and camera firmware updated
- Gimbal v0.3.0
- Camera v0.2.1
- Stream resolution saved as FHD is maintained after reboot.
Gimbal and camera compostion and forwarding
- References
 - https://developer.ridgerun.com/wiki/index.php/Jetson_Nano/Gstreamer/Example_Pipelines/Transforming
 - https://developer.download.nvidia.com/embedded/L4T/r24_Release_v2.1/Docs/Accelerated_GStreamer_User_Guide_Release_24.2.1.pdf
- Composition with tee's
- Frame rate do not match
- RBGA format is necessary for composition
- NV12 is necessary for JPEG encoding
- Need to evaluate best data format to use at each operation
 - Maybe framerate does not work well because of format?
- Things do not work
 - “nv3dsink” is not functional. Use “nvovertlaysink” during local machine test
- Transfer to control station through WiFi
 - gst-launch-1.0 \
 - uridecodebin uri=rtsp://192.168.144.25:8554/main.264 source::latency=0 \
 - ! nvvidconv \
 - ! 'video/x-
 - raw(memory:NVMM),width=960,height=540,format=RGBA,framerate=30/1' \
 - ! tee name=t_gimbal0 \
 - nvarguscamerasrc sensor-id=0 \
 - ! 'video/x-
 - raw(memory:NVMM),width=4608,height=2592,framerate=10/1,format=NV12' \
 - ! nvvidconv \
 - ! 'video/x-raw(memory:NVMM),format=RGBA,width=960,height=540' \
 - ! tee name=t_camera0 \
 - t_gimbal0. ! queue ! comp. \
 - t_camera0. ! queue ! comp. \
 - nvcompositor name=comp \
 - sink_0::xpos=0 sink_0::ypos=0 sink_0::width=960 sink_0::height=540 \
 - sink_1::xpos=0 sink_1::ypos=540 sink_1::width=960 sink_1::height=540 \
 - ! nvvidconv \
 - ! 'video/x-raw(memory:NVMM),format=NV12,framerate=10/1' \
 - ! nvjpegenc \

- ! rtpjpegpay \
- ! udpsink host=192.168.1.32 port=5010



Photograph taken by the author

- Transfer to control station through 5GHz IP network
 - gst-launch-1.0 \
 - uridecodebin uri=rtsp://192.168.144.25:8554/main.264 source::latency=0 \
 - ! nvvidconv \
 - ! 'video/x-
 - raw(memory:NVMM),width=960,height=540,format=RGBA,framerate=30/1' \
 - ! tee name=t_gimbal0 \
 - nvarguscamerasrc sensor-id=0 \
 - ! 'video/x-
 - raw(memory:NVMM),width=4608,height=2592,framerate=10/1,format=NV12' \
 - ! nvvidconv \
 - ! 'video/x-raw(memory:NVMM),format=RGBA,width=960,height=540' \
 - ! tee name=t_camera0 \
 - t_gimbal0. ! queue ! comp. \
 - t_camera0. ! queue ! comp. \
 - nvcompositor name=comp \
 - sink_0::xpos=0 sink_0::ypos=0 sink_0::width=960 sink_0::height=540 \
 - sink_1::xpos=0 sink_1::ypos=540 sink_1::width=960 sink_1::height=540 \
 - ! nvvidconv \
 - ! 'video/x-raw(memory:NVMM),format=NV12,framerate=10/1' \
 - ! nvjpegenc \
 - ! rtpjpegpay \
 - ! udpsink host=192.168.144.10 port=5010



Photograph taken by the author

Things to do

- Save image frame with reduced rate along the image/video transmission
- Initialize gimbal with Python code
- Streamline Python and gstream codes with custom image folder location per run

Change tee order. Save image and transfer to control station through WiFi. Low frame rate image saving and low frame rate composed image transmission

- gst-launch-1.0 \
- uridecodebin uri=rtsp://192.168.144.25:8554/main.264 source::latency=0 \
- ! tee name=t0_gimbal0 \
- ! nvvidconv \
- ! videorate \
- ! "video/x-raw(memory:NVMM),framerate=1/1" \
- ! nvvidconv \
- ! nvjpegenc \
- ! multifilesink location=~/px4_video/gimbal0_%d.jpg \
- nvarguscamerasrc sensor-id=0 \
- ! 'video/x-raw(memory:NVMM),width=4608,height=2592,framerate=10/1,format=NV12' \
- ! tee name=t0_camera0 \
- ! nvvidconv \
- ! videorate \
- ! "video/x-raw,framerate=1/1" \
- ! nvvidconv \
- ! nvjpegenc \
- ! multifilesink location=~/px4_video/camera0_%d.jpg \
- t0_gimbal0. \
- ! nvvidconv \
- ! 'video/x-raw(memory:NVMM),width=960,height=540,format=RGBA' \
- ! queue ! comp. \
- t0_camera0. \

- ! nvvidconv \
- ! 'video/x-raw(memory:NVMM),width=960,height=540,format=RGBA' \
- ! queue ! comp. \
- nvcompositor name=comp \
- sink_0::xpos=0 sink_0::ypos=0 sink_0::width=960 sink_0::height=540 \
- sink_1::xpos=0 sink_1::ypos=540 sink_1::width=960 sink_1::height=540 \
- ! nvvidconv \
- ! "video/x-raw(memory:NVMM),format=NV12" \
- ! videorate \
- ! "video/x-raw(memory:NVMM),framerate=10/1" \
- ! nvvidconv \
- ! nvjpegenc \
- ! rtpjpegpay \
- ! udpsink host=192.168.1.32 port=5010

09/03/23

Things to do

- Enable JETSON WiFi hotspot with fixed IP addresses at JETSON and laptop
- Make a drone set up code with Python and gstreamer
- Clean up power cables
- Clean up signal cables
- Enable remote ID information with MAVLINK
- MAVLink Python code to record position over time along the images
- Enable RC controller
- Enable RC controller with long range radio
- Attach long range radio
- Figure out onboard vs. RC controller priority
- JETSON WiFi hotspot
 - JETSON hotspot creates a WiFi network with WEP password
 - It does not create WiFi network effectively
 - There is no IP assigned to itself
 - Control station laptop does not see “nano” network at all
 - Cannot joint it as hidden network
 - “Edit connection” does not work and cannot configure further
 - Control station Ubuntu 22.04 can create a WiFi network with WPA encryption
 - It created it's own IP network 10.x.x.1
 - JETSON can see the network
 - JETSON cannot join the network, as it does not allow password entry
 - There is no guarantee JETSON will join control station network after all at field
 - So, no apparent solution
Use 5GHz IP network
 - Created px4_start.sh at root directory with gstreamer pipeline to control station through 5GHz (192.168.144.10)
 - Disconnect HDMI connector

- Created a VNC connection to JETSON through 5GHz IP network (192.168.144.20)
 - Connected to JETSON through 5GHz IP network with VNC
 - Started control station video gstreamer sink
 - Started JETSON px4_start.sh
 - Video streaming started
 - Closed VNC
 - Re-open VNC
 - Stopped video streaming
 - Confirmed video images were saved
- One problem: JETSON was not powered up by battery. Need to confirm why.
- JETSON is powered by battery and voltage converter power source.
 - The procedure was repeated.
 - Complete standalone operation was performed
- Things to improve
- Add gimbal lock mode and orientation Python code in the beginning
 - Make a quick shutdown script
- Wire tie down



Photographs taken by the author

px4_start.sh update

- Added gimbal setup with FPV lock and pitch -90 degree command
 - Enable MAVLink communication through UART
- Install pymavlink
- https://mavlink.io/en/mavgen_python/
- Pymavlink example codes. Most of them are based on IP network
- <https://www.ardusub.com/developers/pymavlink.html>
- Previous example: sudo mavproxy.py --master=/dev/ttyTHS1 --baud=115200
- Header should be:
 - `from pymavlink import mavutil
master = mavutil.mavlink_connection("/dev/ttyTHS1",
baud=115200)`
- Jupyter notebook install
 - <https://github.com/NVIDIA-AI-IOT/jetbot/wiki/>
 - <https://forums.developer.nvidia.com/t/jupyter-notebook-in-jetson-nano/72586/2>
 - sudo apt install nodejs npm
 - sudo apt install python3-pip
 - sudo pip3 install jupyter jupyterlab
 - sudo jupyter labextension install @jupyter-widgets/jupyterlab-manager
 - jupyter lab --generate-config
- Error: no module named packaging version
 - pip3 install packaging
- Error:
 - Pip3 install setuptools
- Need anaconda or archiconda
 - <https://github.com/yqlbu/archiconda3/blob/master/README.md>
- conda info --envs
- Jupyter does not work

09/04/23

MAVLINK prototype Python code using pymavlink

- Adopted MAVLink code from last year's MAVLink simulation suite
- Initial communication with Pixhawk was successful
 - Can create Python connection link
 - Can listen to heart beat
 - Can write a request for message and get acknowledgement
 - Can read overall message updates from Pixhawk, including GPS position
- Problems
 - MAVLink constants are not recognized
 - `mavutil.mavlink.MAVLINK_MSG_ID_UTM_GLOBAL_POSITION` is not recognized
- `UTM_GLOBAL_POSITION` is "The global position resulting from GPS and sensor fusion"
 - GPS1, GPS2, and IMU's
- Found that the message constants are for MAVLink2

- Set MAVLINK20=1 to change dialect
 - https://mavlink.io/en/mavgen_python/
- The code can read UTM_GLOBAL_POSITION from pymavlink import mavutil

```

import time, sys
import threading
import math
import os

def request_message_interval(message_id: int, frequency_hz: float):
    message = master.mav.command_long_encode(
        master.target_system, master.target_component,
        mavutil.mavlink.MAV_CMD_SET_MESSAGE_INTERVAL, 0,
        message_id, # The MAVLink message ID
        1e6 / frequency_hz, # The interval between two messages in microseconds. Set to -1 to
        disable and 0 to request default rate.
        0, 0, 0, 0, # Unused parameters
        0, # Target address of message stream (if message has target address fields). 0: Flight-stack
        default (recommended), 1: address of requestor, 2: broadcast.
    )
    master.mav.send(message)
    # Wait for a response (blocking) to the MAV_CMD_SET_MESSAGE_INTERVAL command and
    print result
    response = master.recv_match(type='COMMAND_ACK', blocking=True)
    if response and response.command == mavutil.mavlink.MAV_CMD_SET_MESSAGE_INTERVAL
    and response.result == mavutil.mavlink.MAV_RESULT_ACCEPTED:
        print("Command accepted")
    else:
        print("Command failed")

    def request_message():
        request_message_interval(mavutil.mavlink.MAVLINK_MSG_ID_UTM_GLOBAL_POSITION, 10)
#340
        request_message_interval(mavutil.mavlink.MAVLINK_MSG_ID_OPEN_DRONE_ID_LOCATION,
1) #12901
        def get_telemetry_data():
            ack_msg = master.recv_match(type='COMMAND_ACK')
            observation=[]
            t_param='UTM_GLOBAL_POSITION' #340
            try:
                UTM_GLOBAL_POSITION_lat = master.messages[t_param].lat
                UTM_GLOBAL_POSITION_lon = master.messages[t_param].lon
                UTM_GLOBAL_POSITION_alt = master.messages[t_param].alt
                UTM_GLOBAL_POSITION_vx = master.messages[t_param].vx
                UTM_GLOBAL_POSITION_vy = master.messages[t_param].vy
                UTM_GLOBAL_POSITION_vz = master.messages[t_param].vz

```

```

UTM_GLOBAL_POSITION_data=[UTM_GLOBAL_POSITION_lat,UTM_GLOBAL_POSITION_lon,UT
M_GLOBAL_POSITION_alt,
    UTM_GLOBAL_POSITION_vx,UTM_GLOBAL_POSITION_vy,UTM_GLOBAL_POSITION_vz]
observation.extend(UTM_GLOBAL_POSITION_data) #count=45+3=48
except:
    print(t_param, ":", 'No message received')
    return observation

print("Initiate mavlink")
boot_time = time.time()
master = mavutil.mavlink_connection("/dev/ttyTHS1", baud=115200)
master.wait_heartbeat()
print("Heartbeat: (system %u component %u)" % (master.target_system,
master.target_component))
print("Request messages")
request_message()
observation=get_telemetry_data()
print(observation)

```

Can use this code to add GPS location information along the image timeline.

Remote ID works at TELEM1 port, but not at TELEM2 port

When telemetry radio and remote ID are swapped, there is an error message at start

Might need to give up TELEM2 and use TELEM3, shifting MAV_2 to Ethernet

It is a flight controller TELEM2 problem. A new controller has TELEM2 port working.

Next step

- Set Remote ID to TELEM3
- Set MAVLink to Ethernet
- Make a new Ethernet cable – current cable is too short
- Set up a static Ethernet address at Pixhawk 6X
- Convert MAVLink code to UDP

09/05/23

Orignal plan

- Avoid using Ethernet
- Convert MAV_1 to TELEM3
- Covnert MAV_2 to UART4
- Pixhawk does not have an option to map MAV_x to UART4
New plan – cannot avoid using Ethernet
- MAV_1 is shifted to TELEM3 port
- Confirmed Cube ID broadcast drone ID with Drone Scanner
- MAV_2 is set to Ethernet
- Made a new cable with a longer connection between Ethernet switch and ETH port
IP addresses added for Pixhawk
- Control station: 192.168.144.10?
- Ground unit: 192.168.144.12

- Air unit: 192.168.144.11
- JETSON: 192.168.144.20
- Gimbal0: 192.68.144.25
- Pixhawk: 192.168.144.19

Pixhawk respond to ping at 192.168.144.19

Connection line is:

- master = mavutil.mavlink_connection("udpout:192.168.144.19:14540")
- Still confusing if it should be udpin or udpout

Updated code works once after each reboot. But not twice.

It needs a special wait code, in addition to heart beat

- def wait_conn():
 - Sends a ping to stabilish the UDP communication and awaits for a response
 - msg = None
 - while not msg:
 - master.mav.ping_send(
 - int(time.time() * 1e6), # Unix time in microseconds
 - 0, # Ping number
 - 0, # Request ping of all systems
 - 0 # Request ping of all components
 -)
 - msg = master.recv_match()
 - print(msg)
 - time.sleep(0.5)

Remote ID works. Confirmed with Drone Scanner. Need to set up ID's through MAVLink

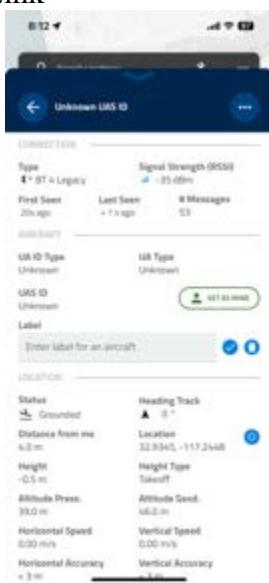


Image captured by the author

Next step

- Figure out why the MAVLink code works only once after reboot
- Set up remote ID through MAVLink
- Remove unused UART cable, tie up new ETH cable, attach remote ID to body

09/06/23

Trying to fix why MAVLink works only once after reboot

- Set up change to NORMAL and forward Enabled does to solve it
- Need both 1) mav.ping.send() response and 2) heartbeat to operate normally
- Need to enable jupyter notebook to avoid repeated rebooting
Install jupyter notebook
- <https://www.sahilramani.com/2021/11/how-to-setup-python3-and-jupyter-notebook-on-jetson-nano-faster>
- <https://www.sahilramani.com/2021/12/how-to-automate-setting-up-jupyter-notebook-on-the-jetson-nano/>
- It has scripted installation
- Stuck at solving environment – failed with initial frozen sove. Retrying with flexible solve. Failed with repodata from current_repodata.json, will retry with next repodata source.
- <https://stackoverflow.com/questions/74781771/how-we-can-resolve-solving-environment-failed-with-initial-frozen-solve-retry>
- Using libmamba as a solver is proposed solution
 - conda install -n base conda-libmamba-solver
 - conda install tensorflow --solver=libmamba
- Jupyter notebook runs now
How to set up OPEN_DRONE_ID_BASIC_ID?
- MAVLink command?
- During compilation?
- <https://mavlink.io/en/messages/common.html>
- MAVLink Type Enumerations
 - MAV_ODID_ID_TYPE
 - MAV_ODID_ID_TYPE_SERIAL_NUMBER
- MAVLink Messages
 - UTM_GLOBAL_POSITION (#340)
 - uas_id
- How to set it?

09/07/23

Back-up radio control override

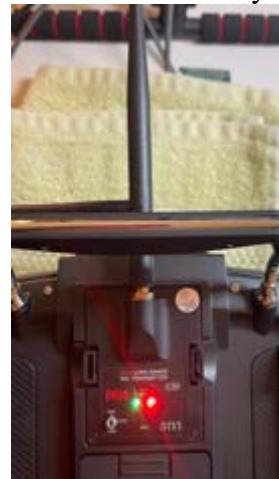
- Radiomaster TX16S MK II 4in1
- Radiomaster R81 receiver
- Frsky D8 mode Bonded and calibrated through QGC



Photographs taken by the author

author

- Need to extend range with 2.4GHz FM30 + FR30 radio
- Attached FM30 external unit to Radiomaster
- Replaced R81 receiver with FR30 receiver
- Enabled external unit
- Bonded FR receiver with FM30
- The replacement was transparent
- RC is recognized and operational
- TO DO's
- Need to attach FR30 receiver to the drone body



Photographs taken by the author

09/08/23

Research funding so far:

- GSDSEF Senior Division - Engineering: Electrical, Mechanical, and Robotics (EEMR)
Category 1st award, American Society of Non-Destructive Testing (San Diego Section)
Winner award \$250
- Amazon DeepRacer March 2023 Qualifier, Country-Specific Top Finisher top 10% award
\$50

- Amazon DeepRacer July 2003 Qualifier, Regional Top Finisher Prize \$400 – received today
- (TBD) NCWIT AspireIT Impact Award \$500

09/09/23

Things to do

- Purchase field set up folding table – 24x36?
 - Need chair or stool?
 - Tie down
 - Remote ID module
 - FM30 RC receiver
 - Wires
 - Program image vs GPS coordinate recording Python script
 - Enable remote ID
 - Practice Fusion 360
 - Study other drones
- Image timestamp recording script
- Having difficulty in sorting file
 - What if Gstreamer files to save a file and index
 - The code needs to know the exactly the last file saved by index and record UTM position with that index
 - List.sort() method does not treat filename as number
 - <https://stackoverflow.com/questions/33159106/sort-filenames-in-directory-in-ascending-order>
 - dirFiles.sort(key=lambda f: int("".join(filter(str.isdigit, f))))
 - dirFiles.sort(key=lambda f: int(re.sub("\D", "", f)))
 - <https://docs.python.org/3/library/re.html>

Timestamp UTM coordinate recording results

```
akim@nano:~$ startp
Camera IP address and port: ('192.168.144.25', 37260)
Data to camera : b'Uf\x01\x04\x00\x00\x0e\x00\x00|\xfc0\x4'
Data from camera: b'Uf\x02\x06\x00\x01\x00\x0e\x10\x00\x00\x00\x00\x00n\x47'
('192.168.144.25', 37260)
Data to camera : b'Uf\x01\x01\x00\x00\x00\x0c\x05\x91\x9e'
Initiate mavlink
Wait for connection
None
PING {time_usec : 1694294711424684, seq : 0, target_system : 255,
target_component : 0}
Wait for heart beat
Heartbeat: (system 1 component 0)
Request messages
Command accepted
1 second stand by
UTM data: [329345181, -1172448022, 16820, 2, 0, 1]
image files detected
['25', '329345188', '-1172448021', '16768', '2', '0', '0']
['26', '329345188', '-1172448021', '16768', '2', '0', '0']
['26', '329345295', '-1172448097', '14844', '0', '-1', '0']
```

```
[ '27', '329345296', '-1172448100', '14818', '0', '-1', '0']
[ '27', '329345296', '-1172448100', '14818', '0', '-1', '0']
```

```
akim@nano:~/px4_video$ cat timestamp_camera0.txt
25,329345188,-1172448021,16768,2,0,0
26,329345295,-1172448097,14844,0,-1,0
27,329345296,-1172448100,14818,0,-1,0
```

Need to add time stamp in addition to GPS coordinates

```
akim@nano:~$ startp
Camera IP address and port: ('192.168.144.25', 37260)
Data to camera : b'Uf\x01\x04\x00\x00\x0e\x00\x00|\xfc0\x4'
Data from camera: b'Uf\x02\x06\x00\x01\x00\x0e\x13\x00\x00\x00\x00\x00\x8ei'
('192.168.144.25', 37260)
Data to camera : b'Uf\x01\x01\x00\x00\x00\x0c\x05\x91\x9e'
Initiate mavlink
Wait for connection
None
PING {time_usec : 1694296908323156, seq : 0, target_system : 255,
target_component : 0}
Wait for heart beat
Heartbeat: (system 1 component 0)
Request messages
Command accepted
1 second stand by
UTM data: [329345075, -1172447985, 19027, -1, 1, 4]
image files detected
['0', '15:01:53.207100', '329345057', '-1172447982', '18891', '-2', '0', '4']
['0', '15:01:53.208073', '329345057', '-1172447982', '18891', '-2', '0', '4']
['1', '15:01:53.211278', '329345057', '-1172447982', '18891', '-2', '0', '4']
['1', '15:01:53.280744', '329345057', '-1172447982', '18888', '-2', '0', '4']
['2', '15:01:53.292098', '329345057', '-1172447982', '18883', '-2', '0', '4']
['3', '15:01:53.369614', '329345056', '-1172447982', '18881', '-2', '0', '4']
['2', '15:01:53.397751', '329345056', '-1172447982', '18879', '-2', '0', '4']
['3', '15:01:53.468104', '329345056', '-1172447982', '18875', '-2', '0', '4']
```

```
akim@nano:~/px4_video$ cat timestamp_gimbal0.txt
2023-09-09 15:01:53.163376
0,15:01:53.208073,329345057,-1172447982,18891,-2,0,4
1,15:01:53.280744,329345057,-1172447982,18888,-2,0,4
2,15:01:53.397751,329345056,-1172447982,18879,-2,0,4
3,15:01:53.468104,329345056,-1172447982,18875,-2,0,4
```

4S battery #1 was permanently damaged

- During on-board PX4 vs. JETSON code development, too much power was discharged
- It is recommended that each cell should stay above 3V, or 12V in 4S.
- It seems the whole voltage went down to 7V range.
- Charging with balancing cannot be done from charger error.
- Charger recognize the battery as 2S cell.
- For both 2S and 4S, charging w/o balancing is possible.
- Recharging try outs did not improve battery condition.

- Charger reduces charging current rapidly at 0.6A and stops automatically.
- Need to consider using 16V DC power supply in place of 4S battery
- Need measure battery voltage level each day.
- Do not leave the powered drone electronics system unattended.
Frame rate is too high or too low according to the time stamp. Image stamping is not in order - #2 and #3 are somehow mixed.
- Initial frame rate is higher than 2 frames/sec
- Later images were recorded frame is about 1 frame/sec exact
- Intended frame rate is 2 frames/sec.
- Not sure what happened

09/10/23

Prototype 2 planning: materials

Flight controller	Holybro Pixhawk 6X
Gimbal camera	SIYI A8 or ZR10
Fixed camera	Raspberry Pi Camera Module 3
GPS #1	Holybro M9N, or M10N
GPS #2	Holybro M9N, or M10N
Motor	MN5008-340
Propeller	P18*6.1
Battery	6S and ?
RC controller	Radio Master 16S MK II
RC radio	SIYI FM30
Telemetry ratio	SIYI 2.4G datalink
IP network radio	SIYI HM30
Remote IT	Cube ID
Ethernet switch	TP-Link 5 port ethernet switch
Power distribution board	?
Distance sensor	
Mission computer	JETSON Orin Nano
Electronic Speed Control	?

Table created by the author

Questions

- Drone design objectives
 - Wind resistance
 - Maximum speed
 - Maximum payload
- Number of arms: quad, hex, octa, etc.
 - For hex drone, co-axial drive can be supported by 2x ESC or 1x ESC
- Coaxial or not: 75% efficiency at bottom propeller
- Maximum payload with 70% thrust
- Thrust requirement
- Single vs. All-in-one ESC

- Total current needed at 70% and 100% thrust
- Battery requirement to support current consumption
- Wire requirement and rating
 - XT30: 30A
 - XT60: 60A
 - XT90: 100A
 - AS150: 150A
- GPS M10N vs. M9N
 - M9N has better accuracy than M10N
 - <https://holybro.com/collections/standard-gps-module/products/m10-gps>
 - Circular Error Probability (CEP)
 - M9N CEP=1.5m
 - M10N CEP=2.0m
- Servo motors
 - Dynamixels are convenient, but torque is low, and need additional gear box

09/11/23

Drone total weight vs. Thrust demand calculation

- Coaxial propeller has 75% thrust effectively (from T-Motor datasheet)

Drone weight (kg)	5	5	5	5
Payload (kg)	10	10	10	10
Total weight (kg)	15	15	15	15
# of arms	6	6	4	4
# of coax (75% of motor thrust)	6	0	4	0
Thrust needed per motor (kg)	1.43	2.50	2.14	3.75

Drone weight (kg)	7.5	7.5	7.5	7.5
Payload (kg)	15	15	15	15
Total weight (kg)	22.5	22.5	22.5	22.5
# of arms	6	6	4	4
# of coax (75% of motor thrust)	6	0	4	0
Thrust needed per motor (kg)	2.14	3.75	3.21	5.63

Drone weight (kg)	10	10	10	10
Payload (kg)	20	20	20	20
Total weight (kg)	30	30	30	30
# of arms	6	6	4	4
# of coax (75% of motor thrust)	6	0	4	0
Thrust needed per motor (kg)	2.86	5.00	4.29	7.50

Table created by the author

- Maximum take off total weight can be lifted by 70% motor thrust

Drone size and center area calculation

Propeller (inch)	18	15
------------------	----	----

Clearance	1	1
Quad size	40.4	33.8
Quad inner circle radius	4.4	3.8
Hexa size	56	47
Hexa inner circle radius	10	17
Octa size	67.6	56.8
Octa inner circle radius	15.8	13.4

Table created by the author

What kind of mechanical structures are necessary?

- Arm holding bearing and gear
- Tilting mechanism
 - Fixed tilt: simplest, limited angle
 - Parallel linked tilt: medium complexity, slightly more angle
 - Asymmetric linked tilt: most angle, most complex

Fixed tilt diagram

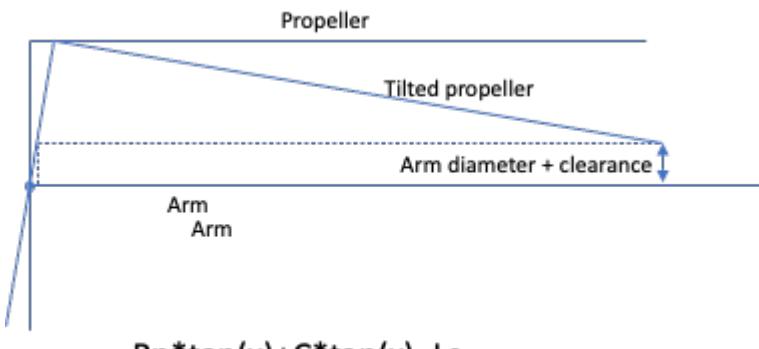


Diagram created by the

author

Parallel linked tilt

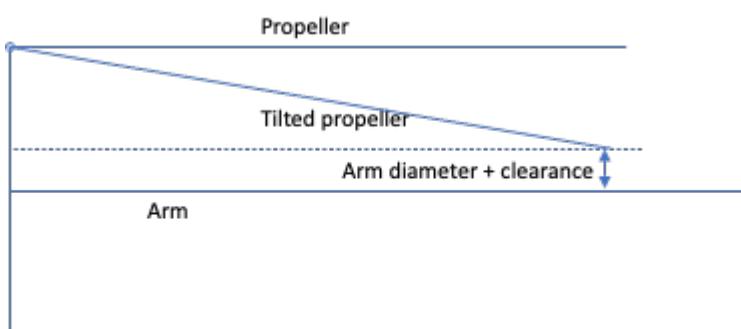


Diagram created by the

author

Propeller (inch)	18	15
Propeller radius (mm)	228.6	190.5
Clearance (mm)	15	15
Stand off from arm center (mm)	100	100
Arm radius (mm)	15	15
Arm + clearance	30	30
Fixed tilt		
Max angle (degree)	21.1	24.4
Parallel linked tilt		
Max angle (degree)	17.8	21.6
Parallel linked tilt has less angle unless stand off is high		
Propeller (inch)	18	15
Propeller radius (mm)	228.6	190.5
Clearance (mm)	15	15
Stand off from arm center (mm)	200	200
Arm radius (mm)	15	15
Arm + clearance	30	30
Fixed tilt		
Max angle (degree)	37.7	42.2
Parallel linked tilt		
Max angle (degree)	48.0	63.2

*Table created by the author***Asymmetric linked tilt**

- Need to raise propeller for tilt toward arm
 - Raise maximizes angle
- Away from arm does not have limit – up to 90 degrees

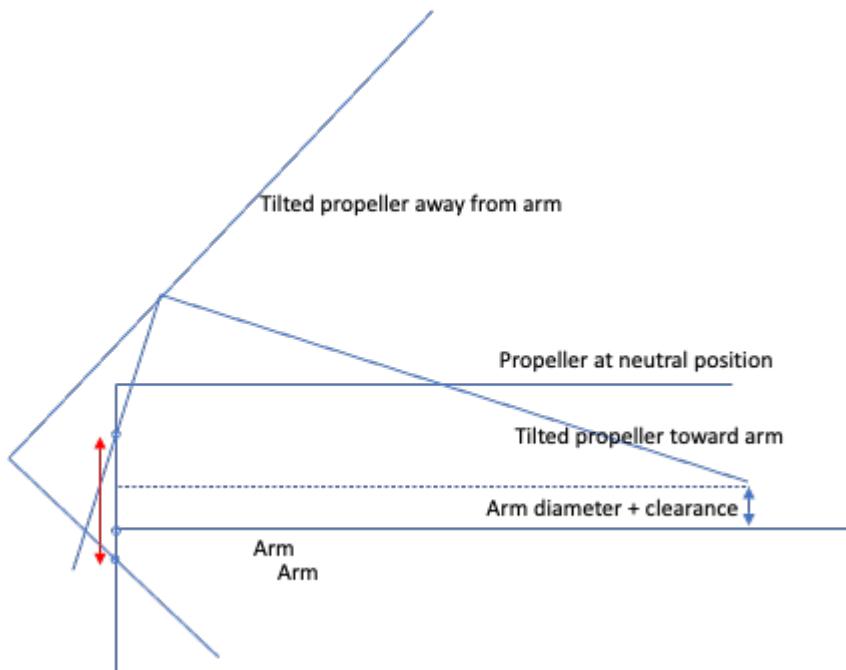


Diagram created by

the author

09/12/23

Hexa drone body board size

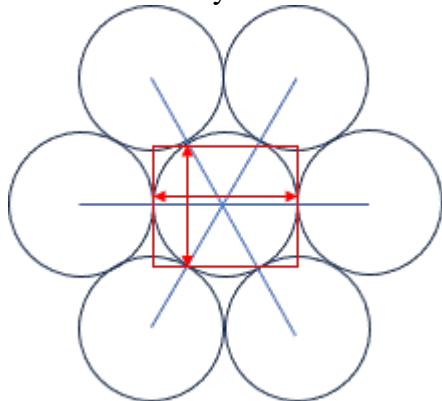


Diagram created by the author

Propeller (inch)	18	15
Propeller (mm)	457	381
60 degree body board size (mm)	396	330

Table created by the author

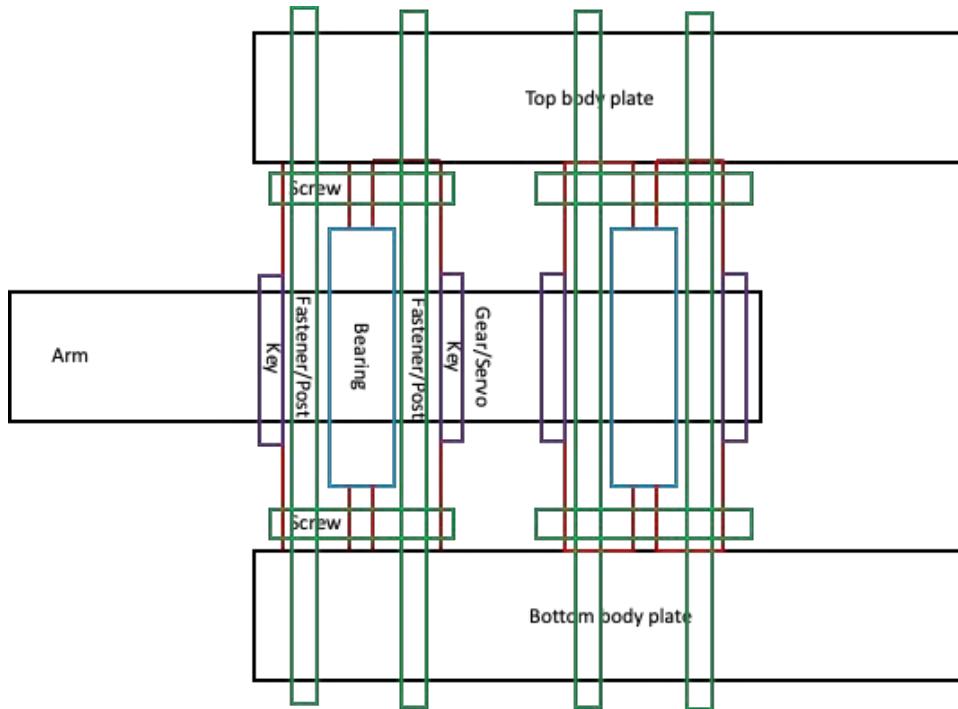
- With 18 inch propeller, board size is 457 x 396 mm
- It barely fits CNC machine with 4mm margin

Multi-level space arrangement

GPS	FC, MC	Radio
Arm/servo	Power	Arm/servo
Camera	Battery	Landing/antenna

Diagram created by the author

Arm rotation structure



Diagram

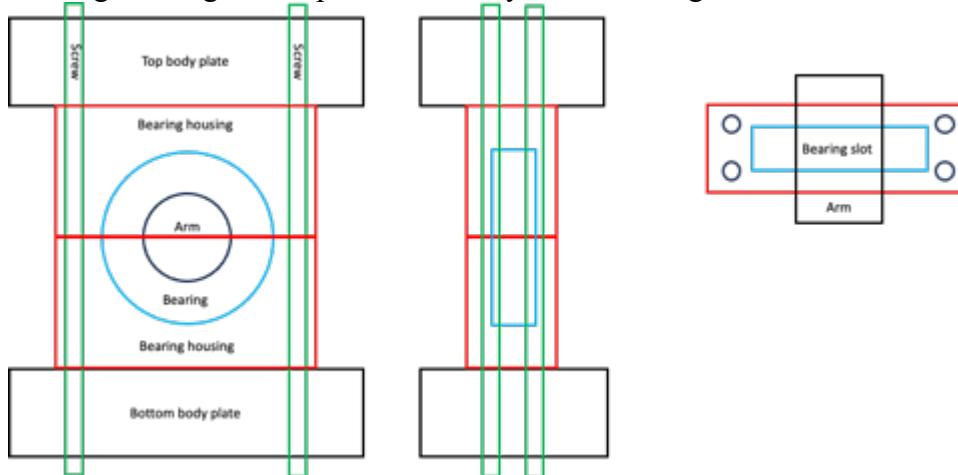
created by the author

Questions

- Is there a way to make vertical screw holes with CNC or others reliably?
- How to attach arm to bearing to avoid any slippage? Glue? Keys?
- How to attach gear to arm

09/13/23

Bearing housing with top and bottom symmetric design



Diagrams

created by the author

CNC will not be able to cut bearing slot smooth. But it should be okay.

Gear shaft assembly tutorial

<https://gearsolutions.com/departments/tooth-tips/when-someone-gives-you-the-shaft/>

Custom gear fab

<https://www.rushgears.com/>

Questions updated

- How to attach arm to bearing to avoid any slippage? Glue? Keys?
- How to attach gear to arm
- Which motor to drive arm rotation

Quad parallel tilt rotor example: <https://oscarliang.com/aimdroix-xray-tilt-rotor-fpv-quad/>



Image taken from OscarLiang

(<https://oscarliang.com/aimdroix-xray-tilt-rotor-fpv-quad/>)

09/14/23

Drone observation:

[VOLIRO - The Omnidirectional Hexacopter](#)



Image taken from "VOLIRO - The Omnidirectional Hexacopter"

(<https://www.youtube.com/watch?v=Sgd0i87r9-0>)

- Hexacopter with arm axis rotation
- Motor must be inside the body. The arm itself does not rotate
- Buffer wires are hanging in the middle of arm
- ESC at the body side of arm
- Later prototype rotates whole arm, leaving buffer wires at body
Rotational wing vs. Fixed wing for lifeguard mission
- Fixed wing can achieve single-charge whole day flight with solar batteries
- Fixed wing is better for survey and monitoring
- Fixed wing is not useful for warning or rescue missions though
3d printing for gears and fixtures
- Highly challenging to find right dimension and parts made of metal
- 3d printed parts has decent mechanical properties
- Nylon or PLA are recommended
- <https://3dsolved.com/best-3d-printing-filament-for-gears/#:~:text=Nylon%20is%20the%20best%20filament,the%20gears%20to%20work%20properly.>
- <https://all3dp.com/2/3d-printed-gears-get-the-gear-that-fits-your-needs/#:~:text=In%20terms%20of%20durability%2C%20nylon,make%20long%2Dlasting%20plastic%20gears.>
- Many articles and forums agree Nylon is best, rather than Polycarbonate or Carbon Fiber
- Nylon
 - Nylon hot end temperature: 230-280C
 - Hot bed: 110-120C
- Polycarbonate
 - Hot end: 260-325C
 - Bed: 100-150C
- Carbon fiber
 - Hot end: 245-270C
 - Bed: 80-90C
- Ender 5 plus:
 - Hot end: 260C max
 - Bed: 110C max
 - Need upgrade to print Polycarbonate
 - <https://3dsolved.com/ender-3-print-polycarbonate-nylon/>
- 0.2mm nozzle might be useful for gear printing with precision
 - <https://all3dp.com/2/ender-3-nozzle-size-which-sizes-are-supported/#:~:text=If%20you're%20looking%20at,also%20X%20and%20Y%20resolution.>
- Items to purchase
 - Nylon filament
 - 0.2mm nozzle
 - 3d print adhesive glue
- Servo motor
 - 35KG 360 degree for 1:2 gear ratio

- https://www.amazon.com/gp/product/B09F2ZXMXW/ref=ox_sc_act_title_1?smid=AVTJB76BDD27&th=1
 - Good for arm rotation drive
 - No load current: 0.12A at 7.4V
 - Stall current: 3.5A at 7.4V
 - Dimension: 55x20x43mm
- Servo motor with dual shaft
 - https://www.amazon.com/dp/B0C69RXXRJ/ref=sspa_dk_detail_1?pd_rd_i=B0C69RXXRJ&pd_rd_w=5DcHW&content-id=amzn1.sym.08ba9b95-1385-44b0-b652-c46acdff309c&pf_rd_p=08ba9b95-1385-44b0-b652-c46acdff309c&pf_rd_r=PV6H74YPGHNYS9YTK2KT&pd_rd_wg=GDz7s&pd_rd_r=5402e32e-dfd2-4507-a4ea-4212a537386d&s=toys-and-games&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWxfGhlbWF0aWM&spLa=ZW5jcnlwGVkUXVhbGlmaWVyPUExNjJMvJVIUVIOVTRRJmVuY3J5cHRIZElkPUEwODYwMDI5M0ZHWk5aS0xLRDA0TyZlbmNyeXB0ZWRBZEIkPUEwNDg4NzkxMzNNMzJKVFdZQkM1NyZ3aWrnZXROYW1lPXNwX2RldGFpbF90aGVtYXRpYyZhY3Rp249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsawNrPXRYdWU&th=1
- Servo controller
 - https://www.amazon.com/SongHe-Channel-interface-PCA9685-arduino-Raspberry/dp/B082QT9D5F/ref=sr_1_1_sspa?crid=1WWRARH0Z8EZB&keywords=servo+controller+voltage+converter&qid=1694724762&sprefix=servo+controller+voltage+converter%2Caps%2C141&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1
 - JETSON needs to send out I2C output to this
 - 16 channel
 - Current capacity?

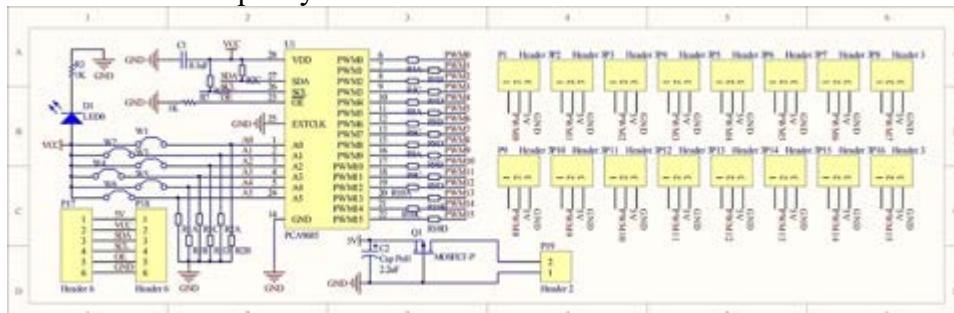


Image taken

from Adafruit (<https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf>)

- PCA9685 datasheet
 - <https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf>
 - This controller can handle up to 5V practically
 - JETSON I2C output is 3.3V
- Power supply for servo motor. Servo motor needs 7.4V voltage source
 - https://www.amazon.com/SoloGood-Independent-Suitable-Sutomotive-Aircraft/dp/B0C1YJ248R/ref=sr_1_2_sspa?crid=B3FX4ZDJTHG1&keywords=7.4v+6s+bec&qid=1694751740&sprefix=7.4v+6s+bec%2Caps%2C151&sr=8-2-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1
 - 8A and 16A peak

- https://www.amazon.com/Switch-Voltage-Stabilizer-7V-25-5V-Aircraft/dp/B0B34WWQ26/ref=sr_1_11?crid=1CTOINY64OKG3&keywords=7.4v+6s+converter&qid=1694751900&sprefix=7.4v+6s+converte%2Caps%2C147&sr=8-11
- 12A, 20A peak

09/15/23

Need to measure prototype drone carbon fiber material dimensions

- Arm diameter 15.94mm
- Arm thickness?
 - https://docs.px4.io/main/en/frames_multicopter/holybro_x500_pixhawk4.html
 - The photo suggests thickness is 1mm



Image taken from PX4 User Guide

Guide (https://docs.px4.io/main/en/frames_multicopter/holybro_x500_pixhawk4.html)

- Top body plate thickness 2.07mm
- Bottom body plate thickness 2.05mm
- Landing leg diameter 16.06mm
- Payload board 2.01mm
- Payload frame diameter 9.93mm, likely 1mm thick
- Top to bottom plate spacing: 28.01mm
- Bearing
 - Inner diameter/outer diameter/length = 16/35/11mm
 - https://www.amazon.com/WJB-6202-16-2RS-Bearing-1740lbf-Capacity/dp/B007HS8GKY/ref=sr_1_13?crid=2SSMCJ0X2US80&keywords=bearing+16mm&qid=1694824792&sprefix=bearing+16mm%2Caps%2C193&sr=8-13
 - 16/35/11 deep groove ball bearing
 - https://www.amazon.com/uxcell-6202-16-2RS-Groove-Bearings-Double/dp/B082PYQYPY/ref=sr_1_3?crid=3FKL8DMU25MB3&keywords=bearing%2B16mm%2Bbore&qid=1694825623&sprefix=bearing%2B16mm%2Caps%2C146&sr=8-3&th=1
 - 16/27/7 deep groove ball bearing
 - https://www.amazon.com/uxcell-16277-2RS-Groove-Bearings-Double/dp/B082PQ25DV/ref=sr_1_6?crid=2SSMCJ0X2US80&keywords=bearing+16mm&qid=1694824792&sprefix=bearing+16mm%2Caps%2C193&sr=8-6
 - 16/22/16 need roller bearing

- https://www.amazon.com/uxcell-Needle-Bearings-Needles-Bearing/dp/B0B5XR62XR/ref=sr_1_13?crid=3FKL8DMU25MB3&keywords=bearing%2B16mm%2Bbore&qid=1694825623&sprefix=bearing%2B16mm%2Caps%2C146&sr=8-13&th=1
- **16/22/16 need roller bearing - higher torque and rpm**
- https://www.amazon.com/uxcell-Bearings-Dynamic-14000rpm-Limiting/dp/B07V7WD82Z/ref=sr_1_14?crid=3FKL8DMU25MB3&keywords=bearing%2B16mm%2Bbore&qid=1694825623&sprefix=bearing%2B16mm%2Caps%2C146&sr=8-14&th=1
- 16/26/36 linear ball bearing
- https://www.amazon.com/dp/B0CDQK67C1/ref=sspa_dk_detail_4?pd_rd_i=B0CDQK67C1&pd_rd_w=nJ3os&content-id=amzn1.sym.0d1092dc-81bb-493f-8769-d5c802257e94&pf_rd_p=0d1092dc-81bb-493f-8769-d5c802257e94&pf_rd_r=JW04K6WZ84MGRK0Z01TG&pd_rd_wg=Diqu&pd_rd_r=c04b6d21-7868-462a-8071-8dbe2dcec791&s=industrial&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhawwy&th=1
- 16/28/70 square flange linear ball bearings
- https://www.amazon.com/uxcell-LMK16UU-Square-Flange-Bearings/dp/B07H976LHL/ref=sr_1_18?crid=3FKL8DMU25MB3&keywords=bearing%2B16mm%2Bbore&qid=1694825623&sprefix=bearing%2B16mm%2Caps%2C146&sr=8-18&th=1
- Carbon fiber tube
- 16x12
- https://www.amazon.com/uxcell-16x12x500mm-Airplane-Quadcopter-Wrapped/dp/B0951XSP1F/ref=sr_1_5?crid=34HQVSKZ1CMHK&keywords=carbon%2Bfiber%2Btube%2B16%2B12&qid=1694825211&s=industrial&sprefix=carbon%2Bfiber%2Btube%2B16%2B12%2Cindustrial%2C154&sr=1-5&th=1
- https://www.amazon.com/Piece-Pultruded-Strength-Hobbies-Projects/dp/B07KWFHV97/ref=sr_1_4?crid=34HQVSKZ1CMHK&keywords=carbon+fiber+tube+16+12&qid=1694825211&s=industrial&sprefix=carbon+fiber+tube+16+12%2Cindustrial%2C154&sr=1-4
- https://www.amazon.com/Carbon-Thickness-16mmx12mmx420mm-Wrapped-Surface/dp/B07GTSHTBTB/ref=sr_1_3?crid=34HQVSKZ1CMHK&keywords=carbon%2Bfiber%2Btube%2B16%2B12&qid=1694825211&s=industrial&sprefix=carbon%2Bfiber%2Btube%2B16%2B12%2Cindustrial%2C154&sr=1-3&th=1
- 3.3V - 5V level shifter
- https://www.amazon.com/HiLetgo-Channels-Converter-Bi-Directional-3-3V-5V/dp/B07F7W91LC/ref=sr_1_1_sspa?crid=229DP1PGTXCMP&keywords=3.3v+5v+logic+level+converter&qid=1694838654&sprefix=3.3v+5v+1%2Caps%2C155&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1
- Plan: Try JETSON Orin Nano I2C for servo control
- Python example
- <https://forums.developer.nvidia.com/t/setting-up-i2c-on-the-jetson-orin-nano/260141>
- Pin out
- <https://jetsonhacks.com/nvidia-jetson-orin-nano-gpio-header-pinout/>

- 3.3 - 5V interface might work as it is

09/16/23

Things to do for prototype 2

- JETSON Orin Nano set up
 - Flash SD card
 - Install system
 - Initial set up
 - Install programs - VScode
 - Set up MAVLink, jupyter, UART, I2C
 - Set up VNC server
 - Set up static IP on Ethernet
- Establish servo motor mechanism
- Enable nylon 3d printing
- Enable carbon fiber plate CNC machining
 - Vacuum suction
- Utilize Fusion360 tool
- Settle down arm rotation mechanism
- Settle down motor orientation mechanism
- Order material
- Science fair approval
- Update Dr. P for SRC approval
- Check patent progress
- Budget current flow through battery, wire, power distribution, ESC, and motor
- Electronics voltage requirements and current demand
- Field dry run in an open area

Drone transport requirement

- Car trunk width is 43 inches

Propeller (inch)	18	15
Clearance	1	1
Quad size longest	40.4	33.8
Quad width	37.0	31.0
Quad inner circle radius	4.4	3.8
Hexa length (long)	56	47
Hexa width	48	41
Hexa length with folding prop	38	32
Hexa width with folding prop	33	28
Hexa inner circle radius	10	17
Octa size	67.6	56.8
Octa inner circle radius	15.8	13.4

Table created by the author

- Need folding propeller to fit inside the car

Drone case study



Image taken from Youtube (<https://www.youtube.com/watch?v=Vy5Ky50eGJs>)

Image taken from Youtube (<https://www.youtube.com/watch?app=desktop&v=hAuNHW8MLE>)

Image taken from Youtube (https://www.youtube.com/watch?v=8HOQl_77CVg)

- Variable pitch propeller
- <https://www.youtube.com/watch?v=Vy5Ky50eGJs>
- Arm rotation – motor at arm end
- [A Novel Overactuated Quadrotor UAV: Modeling, Control and Experimental Validation](#)
- Synchronized tilt drone
- https://www.youtube.com/watch?v=8HOQl_77CVg

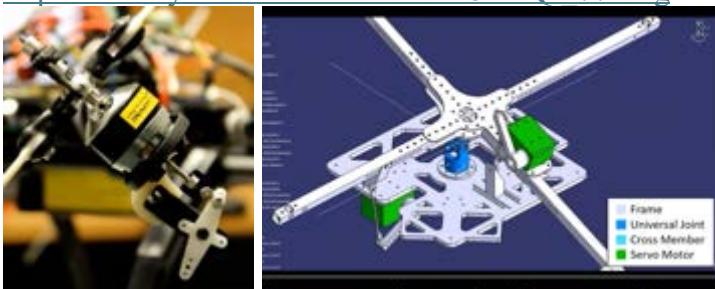


Image taken from Youtube (<https://www.youtube.com/watch?v=j0NJP18T1Es>)

Image taken from Youtube (<https://www.youtube.com/watch?v=-1MihNBS7Zw>)

- Arm rotation and rotor tilt
- <https://www.youtube.com/watch?v=j0NJP18T1Es>
- Thrust tilt
- <https://www.youtube.com/watch?v=-1MihNBS7Zw>

GSDSEF approval process needed to start data collection:

- How to participate: <https://www.gsdsef.org/students/how-to-participate>
- Student resource: <https://www.gsdsef.org/students/resources>
- ISEF Form 4: Human Participants form is not needed as an exempt study
 - <https://www.societyforscience.org/isef/international-rules/human-participants/>
 - Exempt Studies (Do Not Require IRB Pre-approval or Human Participants Paperwork)
 - Some studies involving humans are exempt from IRB pre-approval or additional human participant forms. Exempt projects for ISEF and affiliated fairs are:
 - [Data/record review studies \(e.g., baseball statistics, crime statistics\) in which the data are taken from preexisting data sets that are publicly available and/or](#)

- published and do not involve any interaction with humans or the collection of any data from a human participant for the purpose of the student's research project.
- Behavioral observations of unrestricted, public settings (e.g., shopping mall, public park) in which all of the following apply:
 - a. the researcher has no interaction with the individuals being observed
 - b. the researcher does not manipulate the environment in any way and
 - c. the researcher does not record any personally identifiable data.
 - ISEF Form 5A: Vertebrate Animal Form
 - It is not needed as it is an observation study

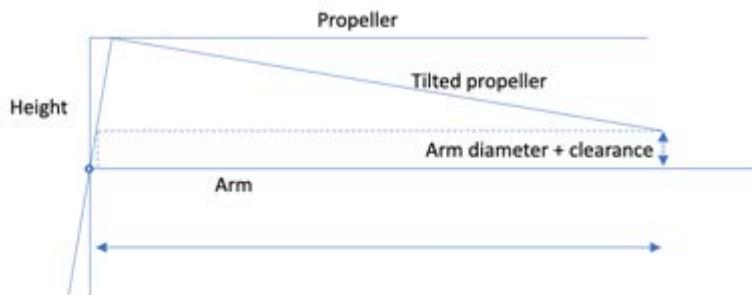
JETSON Orin Nano: SSD vs. SD card

- <https://jetsonhacks.com/2023/05/30/jetson-orin-nano-tutorial-ssd-install-boot-and-jetpack-setup/>
- SSD is faster in speed (up to 7000MB/s) and larger in capacity (2TB)
- SD card is slower (200MB/s) and limited in capacity (<1TB)
- SSD consumes about 2.5A current at 3.3V. It is a significant 8W power adder to JETSON itself
- JETSON Orin Nano maximum power is 15W

09/17/23

Tilt needs more careful distance calculation

- When stand off and rotor is tilt, the distance to body clearance need to consider additional displacement
- Also arm rotation should include additional span from standoff to avoid collision
- If 18 inch prop is too big, need to consider 17, 16, and 15 inch props.



$$Rp \cdot \tan(x) + C \cdot \tan(x) = Ls$$

$$H \cdot \cos(x) + P \cdot \cos(x) = \text{arm-to-body}$$

$$X = \text{atan}(Ls / (Rp + C))$$

Diagram

created by the author

Propeller (inch)	18	15
Propeller radius (mm)	228.6	190.5
Clearance (mm)	25	15
Stand off from arm center (mm)	100	100
Arm radius (mm)	15	15
Arm + clearance	40	30

Fixed tilt		
Max angle (degree)	20.4	24.4
Distance to body from anchor	249.1	214.8

Table created by the author

UAV weight needs more scrutiny

Component	Description	Tilt hexa coax			No tilt hexa coax			No tilt hexa		No tilt quad	
		Unit weight	Count	Sub total	Count	Sub total	Count	Sub total	Count	Sub total	
Flight controller	Holybro Pixhawk 6X	74	1	74	1	74	1	74	1	74	
Gimbal camera	SIYI AB or	95	0	0		0		0		0	
Gimbal camera	ZR10	381	1	381	1	381	1	381	1	381	
Gimbal camera	ZR30	628	0	0		0		0		0	
Fixed camera	RPI CM3	4	1	4	1	4	1	4	1	4	
IP network radio	SIYI HM30 air unit	74	1	74	1	74	1	74	1	74	
Mission computer	JETSON Orin Nano	176	1	176	1	176	1	176	1	176	
GPS #1	Holybro M9N	32	1	32	1	32	1	32	1	32	
GPS #2	Holybro M9N	32	1	32	1	32	1	32	1	32	
R/C radio receiver	SIYI FM30 FR receiver	20	1	20	1	20	1	20	1	20	
Telemetry radio	SIYI 2.4G datalink air unit	28	1	28	1	28	1	28	1	28	
Motor	MN5008-340	135	12	1620	12	1620	6	810	4	540	
Propeller	MF1806	37	12	444	12	444	6	222	4	148	
Propeller	MF1503	24	0	0		0		0		0	
Servo motor		60	12	720	0	0	0	0	0	0	
Remote IT	Cube ID	10	1	10	1	10	1	10	1	10	
Ethernet switch	TP-Link 5 port ethernet switch		1	0	1	0	1	0	1	0	
Power distribution board	4S		1	0	1	0	1	0	1	0	
Electronic Speed Control	?		12	0	12	0	6	0	4	0	
Battery	8400mAh 4S 14.8V 25C Li-ion	556	0	0	0	0	0	0	0	0	
Battery	8400mAh 6S 22.2V 25C Li-ion	849	2	1698	2	1698	1	849	1	849	
Frame, tube, cable	Tilt hexa coax	1000	1	1000							
Frame, tube, cable	No tilt hexa coax & non coax	600			1	600	1	600			
Frame, tube, cable	No tilt quad	400							1	400	
Total weight				6313		5193		3312		2768	

Table created by the author

- Question: which motor is more efficient with about 6kg weight
- Question: does tilt mechanism with additional weight pay off in flight performance?
 - Drag, pull
 - Center of weight and pull vs. Pulling direction alignment

09/18/23

Rotational tilt mechanism

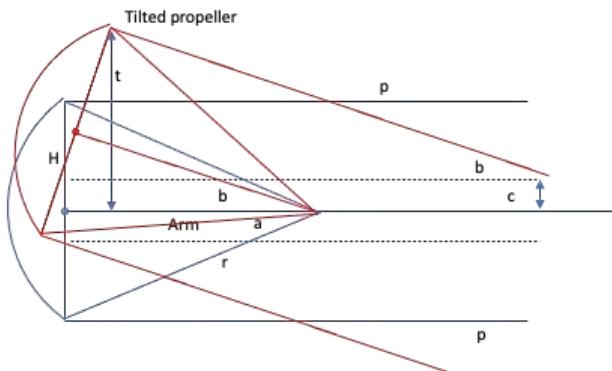


Diagram created by the author

Propeller (inch)	18	15
Propeller radius (mm)	228.6	190.5

Clearance (mm)	25	15
Stand off from arm center (mm)	100	100
Arm radius (mm)	15	15
Arm + clearance	40	30
Rotational tilt		
A=4-c^2/h^2	3.9375	3.9775
B=-2c/h	-0.5	-0.3
C=-4h^2-p^2	-92257.96	-76290.25
Rotational radius solution 1 (mm)	153.1	138.5
Rotational radius solution 2 (mm)	-153.0	-138.5
Max angle (degree)	40.8	46.2

Table created by the author

Tilt angle=b

$$h=r \sin(a)$$

$$t=r \sin(a+b)$$

$$c=t-p \sin(b)=r \sin(a+b)-p \sin(b)$$

Best case is a=b

$$c+p \sin(a)=r \sin(2a)=2r \sin(a) \cos(a)$$

$$c+p \cdot h/r=2r \cdot h/r \cdot \sqrt{1-(h/r)^2}$$

$$r \cdot c/h+p=2 \cdot \sqrt{r^2-h^2}$$

Rotational tilt can improve maximum angle to about 2 times under ideal conditions
Need to calculate the clearance to body

09/19/23

JETSON Orin Nano set up

- <https://developer.nvidia.com/embedded/learn/get-started-jetson-orin-nano-devkit#intro>
- Download JetPack 5.1.2
 - <https://developer.nvidia.com/embedded/jetpack>
- Wrote image to SD card
- Orin Nano needs displayport cable
- Initial set up was complete
- Install updates
- WiFi connection has issue with the main router. Connecting to guest network

*Photograph taken by the author*

09/23/23

Electronic Speed Control

- How to drive co-axial motors
- Each motor with ESC
 - Single output from Pixhawk is connected to two ESC's
 - Each ESC should have 35A capacity
 - 35A ESC: <https://www.getfpv.com/electronics/electronic-speed-controllers-esc/single-esc/lumenier-36a-blheli-32-32bit-2-6s-w-telemetry.html>
 - Q: DShot works in this manner?
- Two motors under single ESC
 - Direct output connection from Pixhawk
 - ESC outputs are connected to two motors
 - 80A ESC: <https://www.getfpv.com/electronics/electronic-speed-controllers-esc/single-esc/iflight-blitz-e80-80a-2-8s-blheli-32-single-esc.html>
- Q: Does Pixhawk support DShot?
- ESC telemetry requires additional wiring
Pixhawk DShot support
- https://docs.px4.io/v1.11/en/peripherals/esc_motors.html
- <https://docs.px4.io/v1.11/en/peripherals/dshot.html>
Hexa coax case
- <https://discuss.px4.io/t/dodecarotor-cox-wiring/11385>

09/24/23

Body plate placement diagram

- Hexagonal plate
- Arm with servo motors are hinged at a side for folding
- Folding exposes motor, gear and wiring for maintenance
- Other corners are also screwed to ensure sturdy assembly
- The end of the hexagon needs to be cut orthogonal to arm
- Tilt block will stop full folding

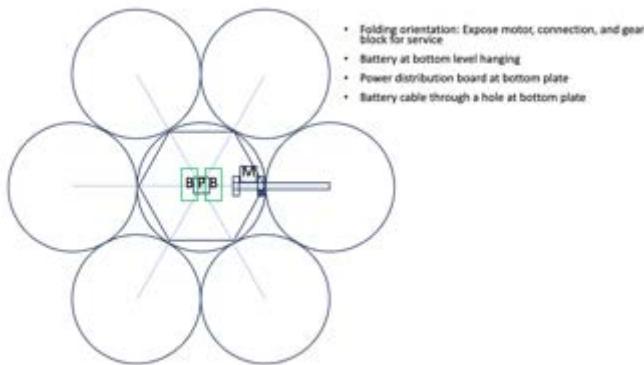


Diagram created by the author

10/01/23

Servo motors for tilt control

- Multiple-function servo motors with additional control is preferred
 - Simplify control
 - No encoder design needed

- Torque should greater than 5 Kg*cm
- XL430-W250-T would be a good candidate
- Voltage at 12 V max could be an issue

		Min V	Typ V	Max V	Torque (Kg*cm)	Stall torque (N*m)	Stall current (A)	No load speed (RPM)	No load current (A)	n (deg/pulse)	Gear ratio	Weight (g)
DYNAMIXEL	XL330-M077-T	3.7	5	6	2.2	0.22	1.5	363	0.15	0.0879	77.5	18
DYNAMIXEL	XL330-M288-T	3.7	5	6	5.3	0.52	1.5	104	0.15	0.0879	288.4	18
DYNAMIXEL	XL-320	6	7.4	8.4	4.0	0.39	1.1	114	0.18	0.293	238	17
DYNAMIXEL	XL430-W250-T	6.5	11.1	12	15.3	1.5	1.4	61	0.15	0.0879	258.5	57
DYNAMIXEL	AX-12A	9	11.1	12	15.3	1.5	1.5	59	0.14	0.293	254	55
DYNAMIXEL	AX-12W	9	11.1	12	2.0	0.2	1.4	470	0.26	0.293	32	53
DYNAMIXEL	MX-12W	10	12	14.8	2.0	0.2	1.4	470	-	0.0879	32	55
DYNAMIXEL	AX-18A	9	11.1	12	18.4	1.8	2.2	97	0.17	0.293	254	56

Table created by the author

Arm folding scheme

- Each arm block is modular
- Arm is folder along a pivot
- Expose motor, gear, and wiring
- Propellers should be folded too

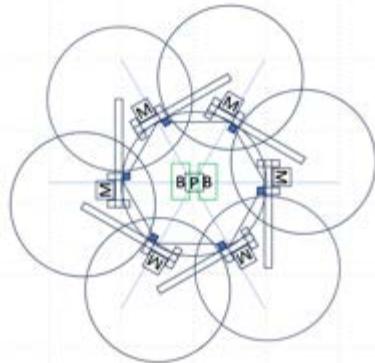


Diagram created by the author

Tilt block – view from side

- 3d print: tilt axis, motor fixture, servo to arm fixture part
- Carbon fiber plate CNC: Tilt block side wall, servo to arm fixture side wall

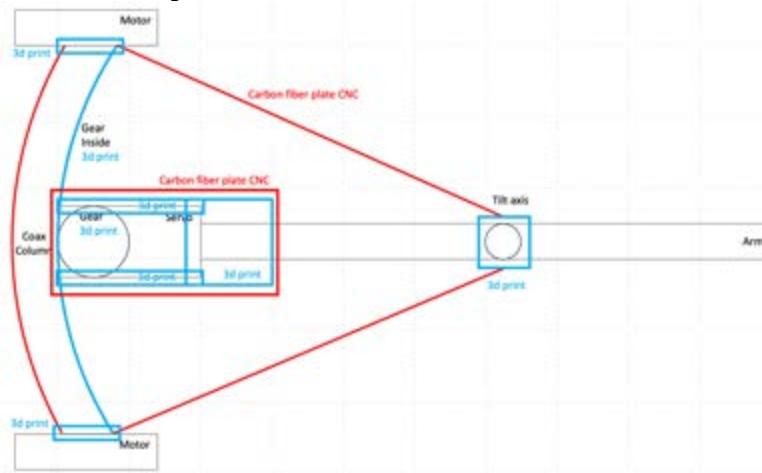


Diagram created by the author

author

Tilt block top view

- Carbon fiber plate for sidewalls to handle thrust from motor
- 3d printed parts connects plate to arm
- Power and signal cables will be routed outside along arm to allow fixes
- Need ESC placement near motor
- Servo side wall should guide tilt guide side wall to avoid twisting

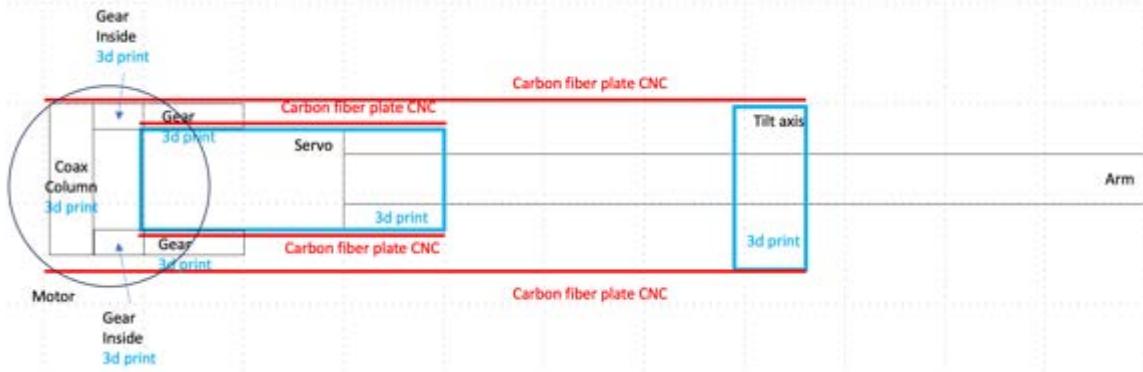


Diagram created by the author

10/02/23

Dynamixel power and communication connection

- U2D2: <https://www.robotis.us/u2d2/>
- U2D2 power hub: <https://www.robotis.us/u2d2-power-hub-board-set/>
- Expansion board: <https://www.robotis.us/3p-jst-expansion-board/>
- JST cables available are too short. Will need about 50 cm-long (18 inches) custom cables
- Star connection is likely because of distance
- How to generate 11.1 - 12V?
- It corresponds to 3S of Li-Ion battery
- Is motor powerful enough or strong enough torque?
- Need to use lower ratio gear to maximize torque

10/03/23

Carbon fiber tube strength calculator

- <https://design215.com/dcal/toolbox/tubing-calculator>
- 16/14/500 tube is 80% stronger than 16/15/500
- 22/18/500 is 760% stronger than 16/15/500
- 22/18/500 is 370% stronger than 16/14/500

10/04/23

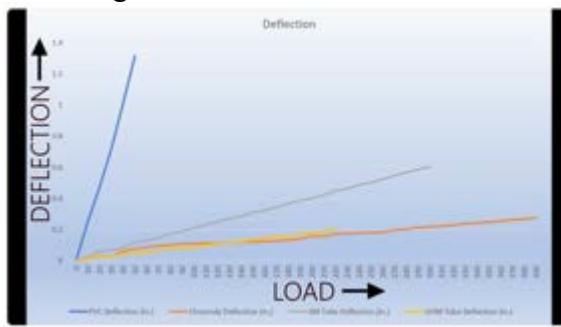
Tube test

[Carbon Fiber Tube Vs. Other Materials](#)



Image taken from Youtube (<https://www.youtube.com/watch?v=S6niIaIwCQg>)

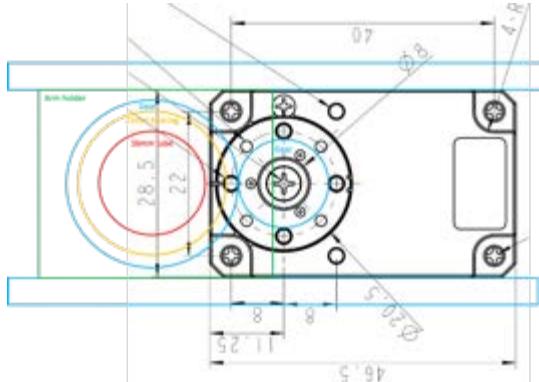
- PVC: Outer 0.8" / Inner 0.6" / 0.17lbs/ft -> 88lbs
 - Oak rod: Outer 0.13" / 0.75lbs/ft -> 148lbs
 - Aluminum: Outer 0.75", Inner 0.62" / 0.15lbs/ft -> 248lbs
 - Chromoly steel: Outer 0.75" / Inner 0.62" / 0.49lbs/ft -> >378lbs
 - Fiber glass: 0.75" / 0.625" / 0.1lbs/ft -> >378lbs
 - Carbon fiber w/ flas overwrap: 0.75" / 0.625" / 0.9lbs/ft -> >368lbs
 - Standard modulus carbon fiber: 0.75 / 0.625" / 0.09lbs/ft -> >378lbs
 - Ultra-high modulus carbon fiber: 0.75 / 0.625 / 0.1lbs/ft -> 288lbs
- Pressure bending
- PVC: 1.32" / 52lbs force
 - Chromoly steel: 0.27" / 388lbs force
 - Standard modulus carbon fiber: 0.6" / 300lbs
 - Ultra -high modulus carbon fiber: 0.2 / 219lbs



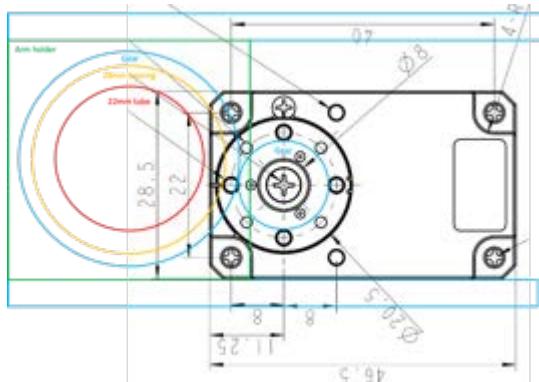
*Image taken from Youtube
(<https://www.youtube.com/watch?v=S6niIaIwCQg>)*

10/04/23

Arm rotation servo drive

*Diagram created by the author*

- Case 1
 - Motor height is 28.5mm
 - Arm holder axis is same as motor height, and they define plate height
 - Motor can be fixed with both top and bottom plates, along the two arm holders
 - 16mm tube works along 22mm bearing
 - 22mm tube will not fit
 - Diagram is for 16mm
 - Gear might need to slightly bigger to make space margin between tube and motor
 - Then the Gear to top and bottom margin might be low against top and bottom plates
- Case 2
 - Arm holder determines the top and bottom plate separation
 - 22mm tube with fit
 - 28mm pin roller bearing
 - Arm gear size determines arm holder height
 - Motor is fixed from the bottom or top plate only
 - Or need additional stand off from top and bottom plates

*Diagram created by the author*

Fusion 360 spur gear generation

- Gear parameters are:
 - 0.75 x 16 and 32
 - Fillet = 0.25, 0.25

- Gear radius: 6 and 12mm
- Arm tube (16mm) will not fit: $8 + 11.5 = 19.5 > 6+12 = 18$
- 34 teeth: diameter=25.5mm, outer diameter=27mm
 - Gear distance = $6 + 25.5/2 = 18.75$
- 36 teeth: diameter=27mm, outer diameter=28.5mm
 - Gear distance = $6 + 27/2 = 19.5\text{mm}$
 - It matches exactly. Not practical.

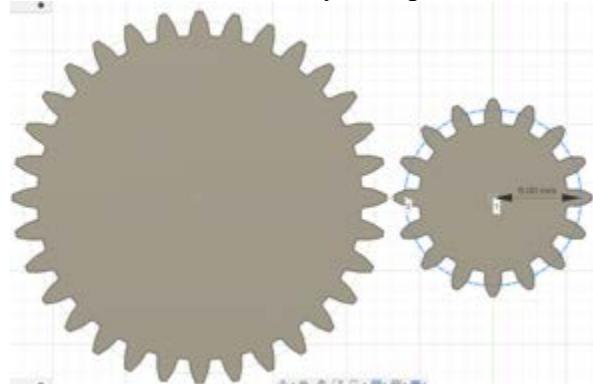


Image created by the author

Motor gear can be larger

- 28 teeth diameter=21mm, outer diameter=22.5mm, inner diameter=19.1
- Screws are at radius=8mm, with 2mm holes
- Just 1.1mm worst-case gap between gear inner diameter vs. Screw holes

10/05/23

Robotis Dynamixel motor and control+power hub arrived

- Mounted U2D2 on power hub
- Connected U2D2 and power hub with JST 3-pin connector
- Installed Dynamixel Wizard 2.0
- https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/
- Connect U2D2 to PC through USB cable
- U2D2 is recognized
- Scanning does not detect motor. Need motor power source?

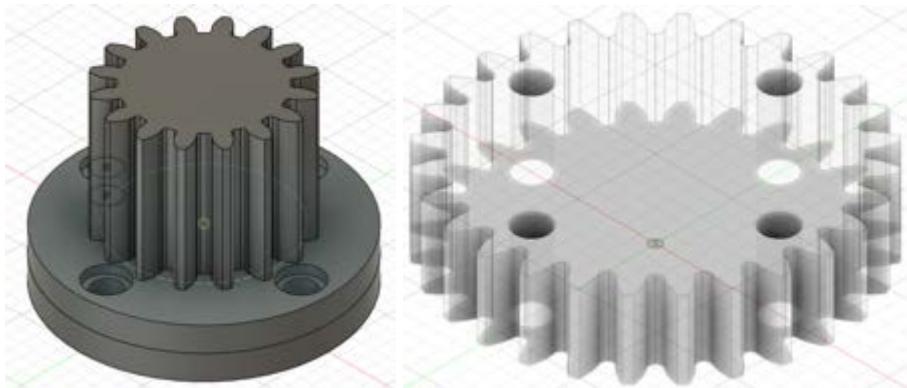


Photographs taken by the author

10/07/23

Pan head screws

- Motor horn assembly requires small screw head
 - M1.7 with 2.5mm head, 0.5mm thick
 - M2 with 3mm head, 0.6mm thick
 - <https://www.nbk1560.com/en-US/products/specialscrew/nedzicom/minaturescrew/SNZ-TBZ/SNZ-M2-TBZ/>
- Initial gear design
- Gear height is 10mm
 - Bottom riser is 2mm
 - Gear has 2mm riser
 - Arm gear needs to be either
 - At least 38 teeth for 16mm arm
 - At least 46 teeth for 22mm arm
 - Motor axis should be horizontal to arm axis for the minimum
 - If motor is fixed to bottom plate, need more complex calculations
 - Larger gear that encloses M2 screws might be an option



Images created by the author

the author

Arm gear			motor gear			motor edge to center	16mm arm			11mm arm		
teeth	dia	outer diameter	diameter	gear dist	radius		min dist	clearance	radius	min dist	clearance	
34	25.5	27	12	18.75	11.25	8	19.25	-0.5	11	22.25	-3.5	
36	27	28.5	12	19.5	11.25	8	19.25	0.25	11	22.25	-2.75	
38	28.5	30	12	20.25	11.25	8	19.25	1	11	22.25	-2	
40	30	31.5	12	21	11.25	8	19.25	1.75	11	22.25	-1.25	
42	31.5	33	12	21.75	11.25	8	19.25	2.5	11	22.25	-0.5	
44	33	34.5	12	22.5	11.25	8			11	22.25	0.25	
46	34.5	36	12	23.25	11.25	8			11	22.25	1	
48	36	37.5	12	24	11.25	8			11	22.25	1.75	

Table created by the author

Large gear case

Arm gear			motor gear 28 teeth			motor edge to center	16mm arm			22mm arm		
teeth	dia	outer diameter	diameter	gear dist	radius		min dist	clearance	radius	min dist	clearance	
34	25.5	27	21	23.25	11.25	8	19.25	4	11	22.25	1	
36	27	28.5	21	24	11.25	8	19.25	4.75	11	22.25	1.75	
38	28.5	30	21	24.75	11.25	8	19.25	5.5	11	22.25	2.5	
40	30	31.5	21	25.5	11.25	8	19.25	6.25	11	22.25	3.25	
42	31.5	33	21	26.25	11.25	8	19.25	7	11	22.25	4	
44	33	34.5	21	27	11.25	8			11	22.25	4.75	
46	34.5	36	21	27.75	11.25	8			11	22.25	5.5	
48	36	37.5	21	28.5	11.25	8			11	22.25	6.25	

Table created by the author

Arm gear



Image created by the author

Dynamixel test drive: Power supply enabled servo motor communication and drive.

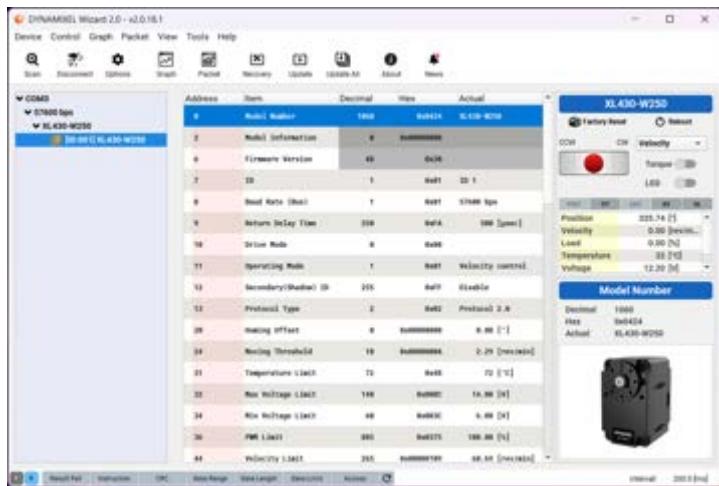


Image captured by the author

ESC and airframe configuration

- PX4 has dodecarotor or hexa coax configuration
 - https://docs.px4.io/main/en/airframes/airframe_reference.html#dodecarotor-cox
 - Need further wiring information
- PWM seems to use IO (MAIN) #1-6 and FMU (AUX) #1-6 connection
- DShot might be limited to FMU (AUX)
- Most of DShot ESC will support PWM input
 - <https://ardupilot.org/copter/docs/common-dshot-escs.html>
- Dodecarotor case
 - <https://discuss.px4.io/t/dodecarotor-cox-wiring/11385/2>

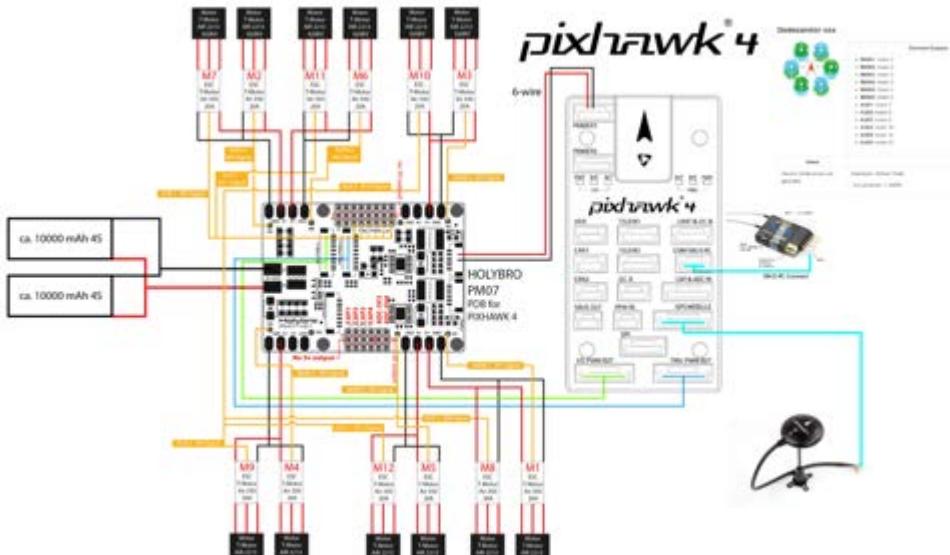


Image taken

from PX4 User Guide (<https://discuss.px4.io/t/dodecarotor-cox-wiring/11385/2>)

- Option 1: Use hex configuration
 - PWM: Use IO (Main) #1-6
 - DShot: Use FMU (Aux) #1-5
 - Use two motors per ESC with >70A current capacity
- Option 2: Use dodecacopter

- #1-6: IO (Main)
- #7-12: FMU (Aux)
- Use single ESC per motor with 35A current capacity
- Need to use PWM
- Q: Can ESC detect PWM
- Q: How the coax is drive is controller? Are they different – how?

10/08/23

GSDSEF proposal submitted for approval

Each motor needs dedicated ESC

- Lots of warnings
- <https://www.quora.com>If-I-have-two-identical-brushless-motors-running-off-of-the-same-ESC-can-one-of-them-run-in-the-opposite-direction-reverse#:~:text=It%20may%20work%C2%0but%20it,of%20one%20of%20the%20motors.>
- <https://drones.stackexchange.com/questions/588/can-i-run-two-motors-off-of-one-esc>
- Not recommended for brushless motor
Options are now
- Option 1: Use hex configuration
 - PWM: Use IO (Main) #1-6
 - DShot: Use FMU (Aux) #1-5
 - Each output connects to two ESCs
 - Bottom motor runs at opposite direction
 - Each motor has its own ESC >35A
 - Use PWM or Dshot
- Option 2: Use dodecacopter
 - #1-6: IO (Main)
 - #7-12: FMU (Aux)
 - Use single ESC per motor with 35A current capacity
 - Need to use PWM
 - Q: Can ESC detect PWM
 - Q: How the coax is drive is controller? Are they different – how?
- ESC vendor
- Amazon has \$40 4x ESCs
 - https://www.amazon.com/Electronic-Controller-Brushless-Multicopter-Quadcopter/dp/B09F3G4KNB/?encoding=UTF8&pd_rd_w=el6g9&content_id=amzn1.sym.5f7e0a27-49c0-47d3-80b2-fd9271d863ca%3Aamzn1.symc.e5c80209-769f-4ade-a325-2eaec14b8e0e&pf_rd_p=5f7e0a27-49c0-47d3-80b2-fd9271d863ca&pf_rd_r=SE360TZEGGEJRGXF1TTK&pd_rd_wg=2H1VT&pd_rd_r=3cb991c3-bec0-4905-8371-d419237809ae&ref_=pd_gw_ci_mcx_mr_hp_atf_m
 - Reviews have multiple malfunctions
- Need quality ESC
- Candidate
- <https://www.getfpv.com/lumenier-51a-blheli-32-32bit-2-6s-w-telemetry.html>

Using DShot might need to reconfigure airframe

- https://docs.px4.io/main/en/dev_airframes/adding_a_new_frame.html
- <https://docs.px4.io/main/en/config/actuators.html>
- <https://docs.px4.io/main/en/config/>

First Nylon 3d printing

- Replaced nozzle to 0.2mm
- Replaced PLA to Nylon filament
- Nozzle temp 250 and bed 60
- No glue yet
- It did not go well
- Need glue. Bed temp=65? Nozzle temp=255?
- Circular brim?



Photograph taken by the author

- Print #2
- Nozzle=260, bed=70, speed=20mm/s
- Lost two components due to delamination and clogging at 5%
 - Need to lower speed=10mm/s
- Remaining two components look good so far
- Two also stuck together, likely from delamination
- Print #3 – if it fails, surely it needs glue
- Speed=10mm/s
- Print failed

10/09/23

Gear parameter update

- Need to make the gear tooth larger
- Module = 1mm from 0.75mm
- Teeth = 12
- Outer diameter = 14mm
- Radial diameter = 12mm
- Applied 3d print glue
- Improved, but needs more adhesion.
- Will need brim
- First layer needs to be lower than current offset



Photograph taken by the author

10/10/23

Gear 3d print

- Temperature might be too high
- Travel speed should be faster
- Best results so far
- Bottom layer needs better adhesion
- Horizontal dimensions increase at higher layers



Photograph taken by the author

10/12/23

GSDSEF proposal approved for research!!!

Additional gear prints failed



Photograph taken

by the author

- Adhesion seems to be the root cause
- Thin bevel (right) did not work
- Things to change
- Dry the Nylon filament, even the brand new one
 - https://www.youtube.com/watch?v=r8TWV_3DBDc&t=328s

- Heated bed at 50 C
- Double-sided adhesive tape
- 0.4mm nozzle for stable print
- Bevel gear

10/15/23

Gear printed well

- Dry box helps
 - 24 hour drying at 70 C
 - Humidity drops to 14-15% from about 60-70%, later down to 12%
 - Large gear print best result so far
 - Bottom to top layer gradient is there still
 - Gear might need to be thicker
 - Might need supporting brim after all



Photographs taken by the author

- Nylon 3D print process
 - 24-hour drying at 70 C to drop humidity below < 15%
 - Add silica gel bags in the 3d printer enclosure
 - Clean bed with alcohol
 - Dry for 5 minutes
 - Apply Nylon glue or adhesive tape
 - Dry for 5 minutes
 - Heat the nozzle and hot bed at 250 C and 70 C
 - Load filament to nozzle
 - Sit at target temperature for 5-10 minutes
 - Set 0.1 mm offset from hot bed
 - Perform leveling

- Add 5x skirt to clean up nozzle
- Clean up nozzle before print starts
- Print

Small gear prints failed



Photograph taken by the author

- Likely bottom layer warpage blocked nozzle movement
- Will need brim to control warpage and yield

10/16/23

Gear set up: 25 degree pressure angle for 3d printed gear. Also servo motor gear has 12 teeth, and needs to be 25 degree angle.

- <https://www.instructables.com/A-Practical-Guide-to-FDM-3D-Printing-Gears/>
 - Pressure angle: 25
 - Module: 1
 - Number of teeth: 12
 - Backlash: 0.1
 - Root fillet radius: 0.3
 - Gear thickness: 10
- Assembly attempt

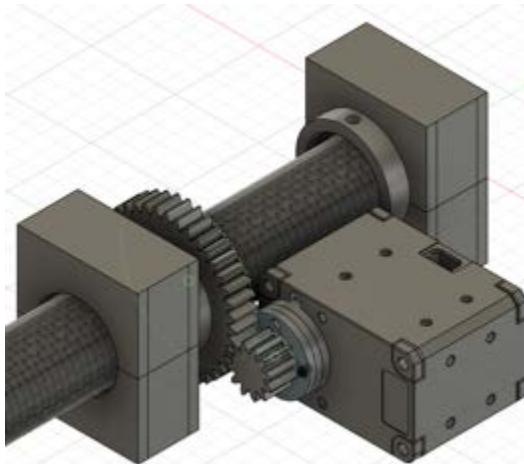


Image created by the author



Image created by the author

- Arm gear needs to have more than 36 teeth to have enough space between motor and arm
 - Settle down with 40 teeth
- Need to avoid capturing positions – use Joints
- Need to install internal gear app from Autodesk for tilt structure
22mm printed gear
- 21.82mm inner diameter for 22.1mm
- 24.8mm arm sleeve diameter
- 37.45 for gear outer diameter

10/18/23

Motor gear and riser print with default brim at Cura

- Brim was effective for adhesion
- 20.5 diameter was printed as 20.2
- 8.2mm opening was printed as 7.95mm
- Gear outer diameter is 13mm for 14mm design
- Need to widen opening diameter. Now 8.4mm to be sure
- Top of gear was somewhat smaller and less distinct



Photograph

taken by the author

Arm and motor set 3d print

- Lowered bed temperature to 50 C, hoping the vertical gradient profile is suppressed
- Added Silica gel bags (7, 100g each) to lower humidity inside the enclosure during print
10/19/23



Photograph taken by

the author

- Overall quality improved.
- Gear teeth at upper layers getting less pressure angle and ill defined.
- Maybe need to use 20 degree as before.

Gear parameter update

- Pressure angle: 20
- Module: 1
- Number of teeth: 12
- Backlash: 0
- Root fillet radius: 0.3
- Gear thickness: 7

If gear teeth are still raised at tip, need to try these:

- layer thickness at 0.08mm from 0.12mm, or 0.16mm
- Wall thickness to 0.8 from 0.4mm
- Layer pattern to concentric from line

Results were similar. Updating parameters

- layer thickness at 0.12mm, or 0.16mm
- Wall thickness to 0.8 from 0.4mm
- Layer pattern to concentric from line
- Outer wall speed at 10mm/s

Gear teeth quality improved. It looks like the key parameter is outer wall speed at 10mm/s from 20mm/s



Photograph taken by the author

Tilt gear design

- Custom inner gear design allows up to 250mm teeth. Same with single or double helix inner gears
- Use projection of regular spur gear for 300mm diameter
- Try 3d printing along arm gears

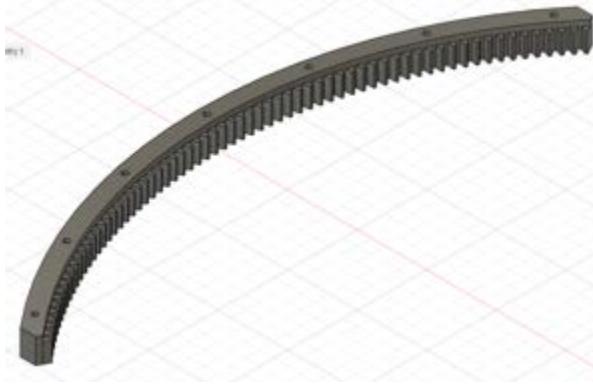


Image created by the author

10/21/23

Servo arm drive gear assembly

- Amazon's flat pan head M2 was in fact 2.4mm
- Used regular Philips M2
- Either raise gear height to 0.8 - 0.9mm
- Or enlarge screw head (3.8mm) seat



Photograph taken by the author

Original screws from Robotis fit much better.

- M2 screw head diameter is 3.25mm
- Sits tightly at the screw head placeholder
- Screw head is not noticeable



Photographs taken by the author

Drone take off and flight time rehearsal

- Drone bare weight w/o battery is 2kg
- Weight with battery is 2.5kg
The drone is too heavy to take off
- Need to reduce weight
- Items to drop
 - RC controller radio
 - Video radio
 - Reduced battery size
 - 4200mAh is 282g, from current 556g 8400mAh
 - Removed
 - XT30 power cable 11g
 - USB-C angled adapter 4g
 - Leg support bar 19g
 - FR RC receiver 21g
 - Air unit ethernet cable 5g
 - Air unit radio and antenna: 118g
 - Unused MAVLink serial cable: 3g
 - ~~XT60 extension power cable: 15g~~
 - Distance sensor: 5g

- XT30 split power cable: 11g
- Total 197g
- Weight with battery is 2.3kg
Again drone does not take off effectively.



Photograph taken by the author

- It is dragged on the surface
- Need to reduce weight further
- Next step is using 282g 4S battery from 556g 4S battery
Tilt block design update



Images created by the author

10/22/23

Arm design progress

- Added more tilt components
- Enabled tilt motion and arm rotation



Image created by the author

Tilt gear print was decent

*Photograph**taken by the author*

10/23/23

Current gear inventory

- Riser: 7
- 12-teeth gear: 5
- 40-teeth arm gear: 3

One of the new printed gear was broken during detach. There seems to be mechanical week at the gear to arm fixture interface

*Photograph taken by the author*

10/26/23

4200mAh battery at 282g, in place fo 8400mAh at 556g

- The drone was able to run more than 10 minutes, after GPS position acquisition
- Battery was at about 12.5V from 16V
- Charging added 3294mAh out of 4200mAh
- Can it run with Air unit radio and antenna?
- Scout drone motor upgrade
- Need more thrust to mount Air Unit radio, antenna, and RC remote controller
- AT2317 could be an upgrade within the current limit
- 50% thrust is about 750g over 450g with current motor
- Need a way to force CW and CCW
- Need new propellers
- Might need new ESC, as current ESC is 20A, and motor AT2317 consumes up to 23A max
- A 30A ESC will work: Lumenier 30A BLHeli_S ESC OPTO (2-4s) DSHOT

10/28/23

Current gear inventory

- Riser: 11
- 12-teeth gear: 9
- 40-teeth arm gear: 5
- Del Mar Beach first flight data collection
- Morning is good as there are fewer people at beach and parking is free until 8am
- It was about to heat the high tide
- Ethernet connection was not immediately recognized
 - Had to restart JETSON forcefully
- Sometimes PX4 did not allow arming
 - Restarting PX4 is easier, but still it might interfere with JETSON
- Battery usages are about less than 1300mAh
 - The battery should be able to run up to 2.5x more distance
- Altitude at 50m might not be sufficient.
 - Need to be at 75m and 100m next time
- Beach sand becomes challenging when gets into electronics
 - Need to keep cables away from sand to avoid contact issues
- High to low tide transition will show more rip current
 - High tide does not much room to maneuver at beach
- 1st run
 - Both camera and gimbal stopped recording in some time up to image ~100
 - GPS tagging logged up to about 30 images
- 2nd run
 - Only gimbal left images
 - RPi CM3 did not leave any images
 - GPS tagging did not produce any log

Example screen shot out of about 290 images captured:



Photograph taken by

the author

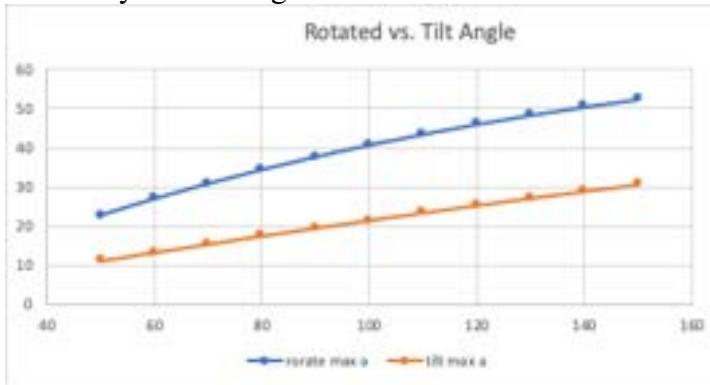
Flight plan for run #1 and #2:



Image captured by the author

Tilt rotor calculation

- Ideal case
 - 100mm height gives about 2x angle with rotated vs. Tilted
- Practical case
 - Rotate angle is less than motor's propeller angle
 - It is not trivial to solve it analytically
 - Need a numerical solver
 - Python coding needed



Plot created by the author

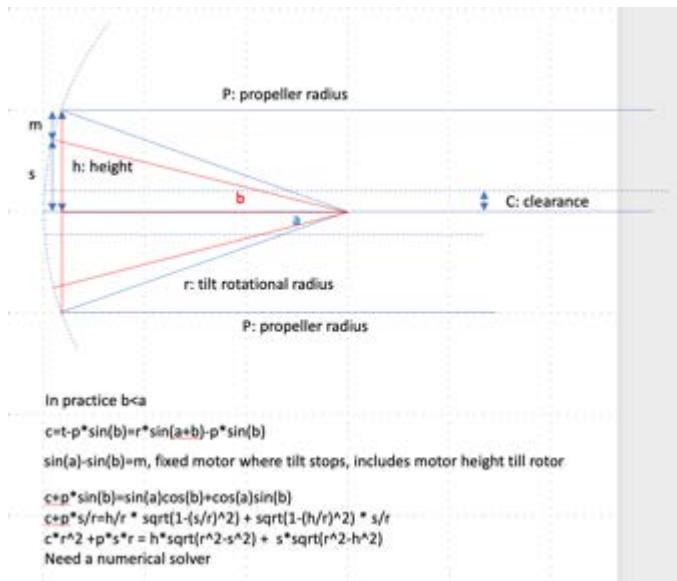
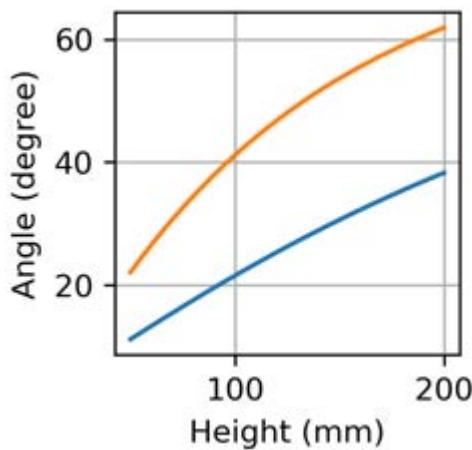


Diagram created by the author

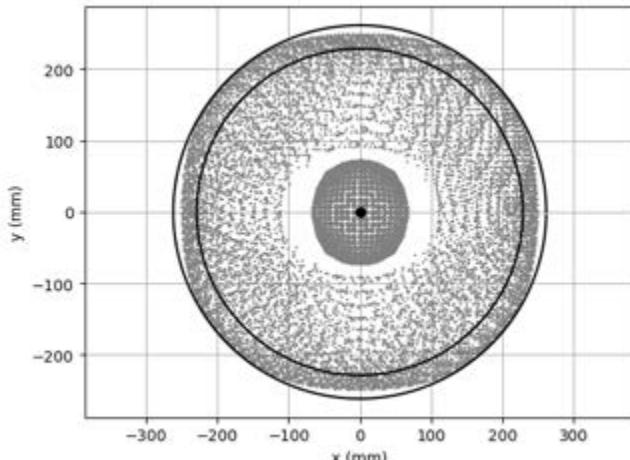
10/29/23



Plot created by the author

Propeller boundary over rotate and tilt

- Propeller height = 100mm
- 18 inch propeller can get to 261mm or 10.3inch radius with rotate and tilt maximally
- Tilt=>pitch up to +/- 40 degree
- Roate=>roll up to +/-45 degree

*Plot created by the author*

Current gear inventory

- Riser: 15
- 12-teeth gear: 9
- 40-teeth arm gear: 4
- Gear prints were not good – shrunked and stringed
- Laptop Linux has an issue:
 - Unexpected return from initial read: Volume Corrupt.
 - Failed to load image XX: Volume Corrupt
 - start_image() returned Volume Corrupt
 - <https://askubuntu.com/questions/1122261/unexpected-return-from-initial-read-volume-corrupt>
 - No error with fsck -f /dev/nvme1n1p2
 - Might EFI partition got corrupt
 - <https://help.ubuntu.com/community/Boot-Repair>
 - Boot-repair ISO image was installed on USB
 - It did not detect any issues
 - The image at Sourceforge was 2020 for 64-bit. A bit suspicious
 - Reinstalled Ubuntu LTS 22.04.03 on USB
 - Booted into USB, and install Boot-Repair
 - This time, Boot-Repair fixed a few thing
 - It shows NVRAM locked error
 - But the bootloader seems to work
- Windows boot require Bitlocker key all the day. Need to fix it
 - <https://superuser.com/questions/433048/bitlocker-issue-recovery-key-is-required-on-each-startup-after-installing-second>
 - Deactivate bitlocker
 - Reboot
 - Activate bitlocker

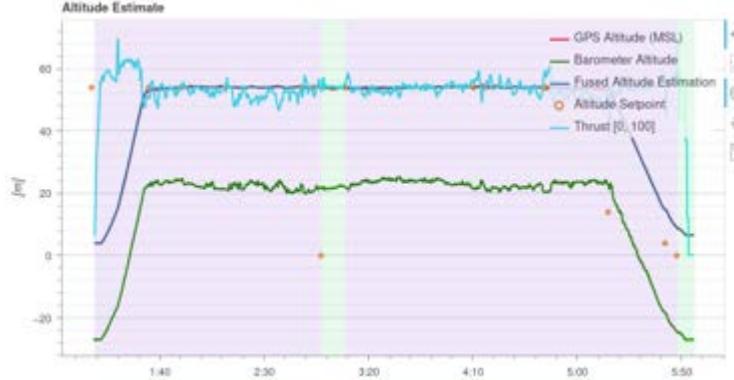
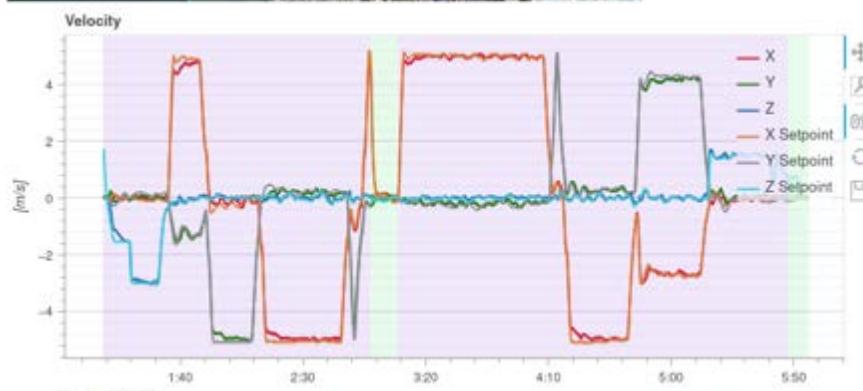
10/30/23

Flight Review (web) log analysis of 10/28 Del Mar Beach flight

PX4 Quadrotor

[Open 3D View](#) [Open PID Analysis](#)

Airframe:	S500 Generic	Distance:	1.04 km
Hardware:	Quadrotor x (4014)	Max Altitude Difference:	50 m
Software Version:	PX4_FMU_V6X (V6X000003)	Average Speed:	12.9 km/h
OS Version:	NuttX, v11.0.0	Max Speed:	18.4 km/h
Estimator:	EKF2	Max Speed Horizontal:	18.4 km/h
Logging Start ?:	28-10-2023 07:39	Max Speed Up:	11.1 km/h
Logging Duration:	0:04:48	Max Speed Down:	6.1 km/h
Vehicle Life	55 minutes 18 seconds		
Flight Time:			
Vehicle UUID:	0006000000003633393831375112000a0004	Max Tilt Angle:	27.5 deg


Images captured from PX4
Flight Review using flight date (<https://review.px4.io/>)

11/01/23

Propeller coordinate is -125, 134

- Practical radius is: 182.6mm
- Gear diameter is 150mm

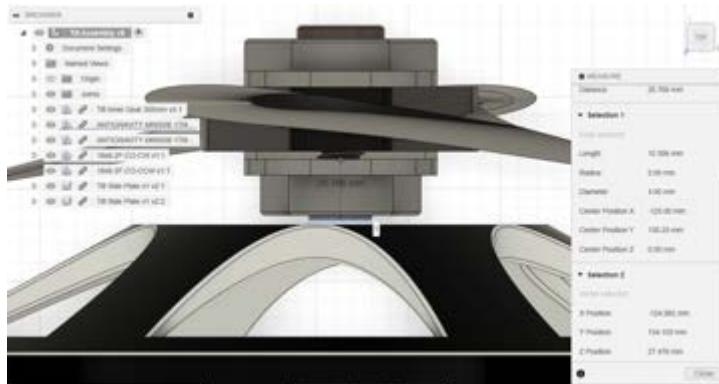
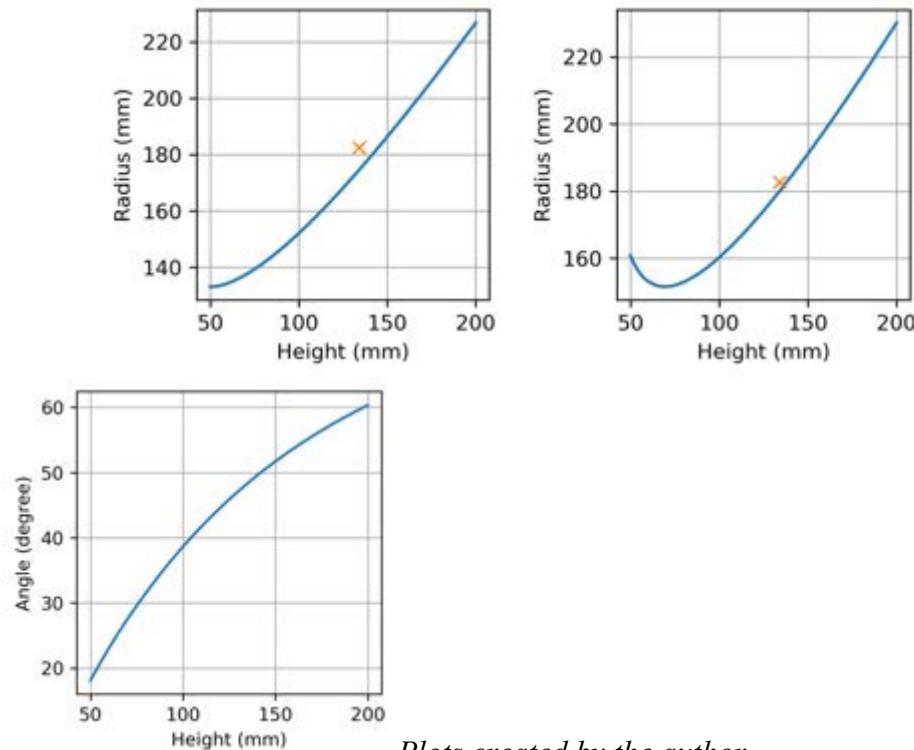


Image created by the author

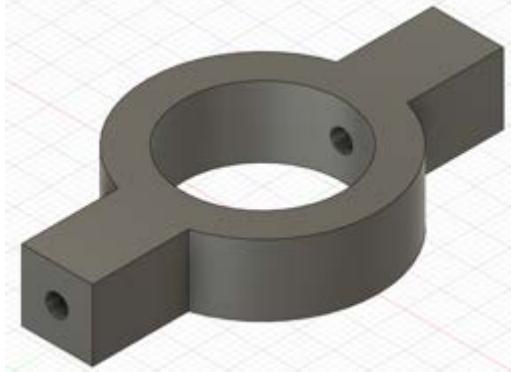
- Maximum gear rotation point is: -118, 94
- $h=134\text{mm}$
- Need to calculate s and m
- Calculate m
 - $(-118,94)/L(118,94)*182.6=(-142.8, 113.8)$
 - $m=113.8\text{mm}$
 - $s=h-m=20.2\text{mm}$
- Need to re-check calculation equation...
 - With the first plot, clearance is 25mm from the center of the arm
 - Current design plot in x mark is above the exact solution line
 - Radius is bigger than the height
 - Therefore the points above the line will have more safety margin or clearance to the arm
 - Points under the curve will be dangerous and propeller can hit the arm
 - Second plot has 35mm clearance from the center of the arm
 - Current design is closer to solution with slight margin
 - Current design point is conservative and could be a good compromise
 - It will turn up to 45 degrees in pitch

*Plots created by the author*

11/01/23

Need to design arm assembly parts

Single-piece pitch axis holder from two-piece design

*Image created by the author*

Tilt drive servo motor holder

- Need to hold servo motor
- Guide tilt rail

*Image created by the author*

- Motor back-side plate can be tilt guide.
 - Need to remove back-side idler assembly to be back-side tilt guide
 - Back-side assembly screws need to be flat for smooth guide plane
- Overall tilt thickness can be reduced
- Motor front panel needs to be separate from front tilt guide
 - Front panel need to open more for front-guide mount overlap opening
- One mitigation is to use top and bottom mounting
- Give up front plate, and use top and bottom plates + back plane for servo mounting
 - So that the width of tilt structure is a bit narrower and more stable
- Need to ensure 3d printed arm axis assembly is tight by itself rather than loose

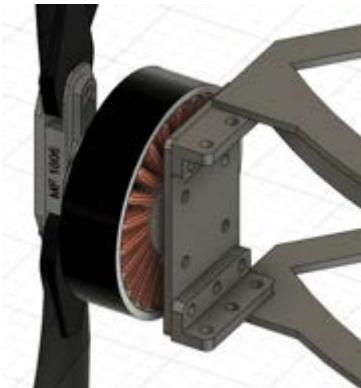
11/02/23

3d print quality degraded

- Need stronger adhesion
- Add more glue
- Raise bed temperature to 60C from 50C

*Photograph taken by the author*

Motor mount plate and bracket

*Image created by the author*

- Need to shift motor mount out of the tilt gear
- It increases tilt radius, giving less tilt angle
- Need to re-calculate tilt angle and margin
- Need to revise side plate to cover bracket mounting

11/03/23

Upgrade motor and propeller

- T-motor MS1101 11 inch propeller
- T-MOTOR AT2317 Long Shaft KV880
- Expect to re-mount additional radios for communication and control
- Replace motor and propeller after beach data collections to be safe

Vendor	Model #	Size	KV	Voltage	Weight	Propeller	Throttle	Voltage (V)	Current (A)	Power (W)	Thrust (Kg)
Holybro	AIR2216II	2216	920	4S		65 T1045II	30%	16	1.44	23	0.21
Holybro	AIR2216II	2216	920	4S		65 T1045II	40%	16	2.29	37	0.31
Holybro	AIR2216II	2216	920	4S		65 T1045II	50%	16	3.6	58	0.45
Holybro	AIR2216II	2216	920	4S		65 T1045II	70%	16	7.92	126	0.84
Holybro	AIR2216II	2216	920	4S		65 T1045II	100%	16	16.37	260	1.33
T-MOTOR	AT2317	2317	880	4S		79 APC 11	40%	14.7	4.8	71	0.59
T-MOTOR	AT2317	2317	880	4S		79 APC 11	45%	14.7	5.7	83	0.67
T-MOTOR	AT2317	2317	880	4S		79 APC 11	50%	14.6	6.6	97	0.74
T-MOTOR	AT2317	2317	880	4S		79 APC 11	70%	14.5	12.4	180	1.16
T-MOTOR	AT2317	2317	880	4S		79 APC 11	100%	14.2	23.4	332	1.67

*Photograph*

taken by the author

3d print quality does not improve with motor and arm gears

- Bottom adhesion is improved with glue

- After 2-3mm gear height, gear teeth begin to be raised up
- Solution?
 - Wall thickness = 0.4mm for 2 pass
 - Slowdown wall print speed at 5mm/s from 10mm/s
 - Bed temp = 50C
- Tilt components design progress
- Large 3d print components designs
- Motor and tilt block supports

11/04/23

3d print adhesion was very poor – not sure why. And again – adhesion is not good, especially at the tip of teeth.

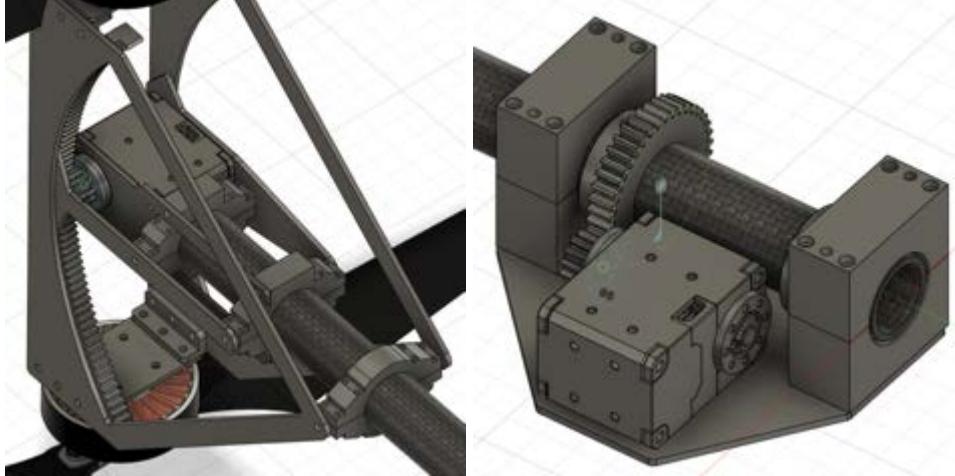


Photographs taken by the author

Tilt group has improved assembly

- Need to align the parts accurately, rather than roughly
- Separate
 - Servo motor fixture
 - Tilt block fixtures
- It might help re-align gear during assembly.

Arm axis bottom plate for folding arm and rotation/roll



Images

created by the author

Fusion 360 Manufacture CNC flow was tried

- 1.5mm or 2.0mm end mill will be the work horse bit

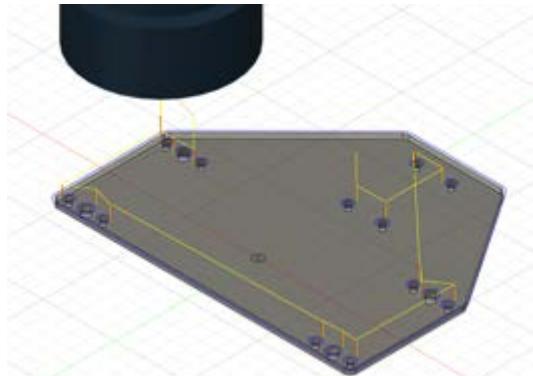


Image created by the author

Replaced 0.2mm nozzle with 0.4mm

- Cleaned up accumulated Nylon deposit around the heating element
- It makes successful arm gear print
- It also takes much less time
- Adhesion was too good, and it was hard to detach after print. Raise temperature to 80C to detach
- Might not need brim after all

11/05/23

Places where low-profile flat head screws are necessary

- Arm roll plate bottom
 - Tilt serve motor mount at back side
 - Need to unify to M2 screws only as much as possible
 - M2 is available up to 50mm
 - For now, the screws do not need to be embedded completely
 - They just limit the last few degrees of tile motion, until it hits the tilt plate
 - Need to enable motion simulation to see the impact of this blockage
Need flexible magnetic print bed for detaching
 - Detaching consumes too much time and cooling
 - Parts get damaged after prints
- ESC planning
- Motor mount plate needs zip tie or screw holes to mount ESC
 - Pitch plate needs holes to zip tie ESC cables along the rail
 - Need a cradle for ESC with AS150 connectors
 - Need to avoid pitch rotation mechanism

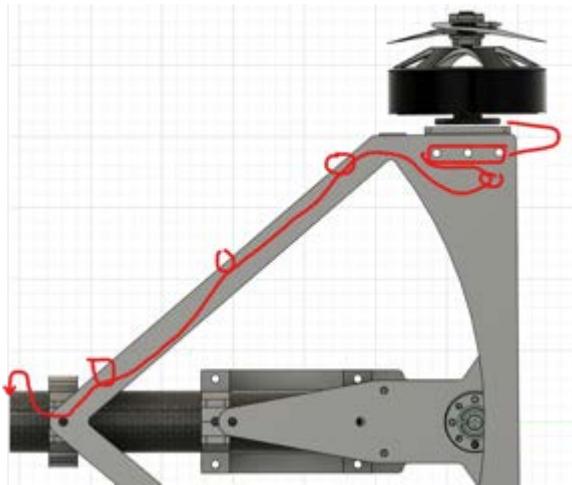


Image created by the author

11/06/23

Test 3d print of all pitch fixture parts

- Servo fixture block needs to be thicker in the corners

11/07/23

First arm assembly 3d print parts

- Motor plate bracket: decent print. One part broke during detach. It implies mechanical strength at the boundary is not good. Adding additional diagonal support to capture nuts in place.
- Servo motor hold fixture
 - M2 screw holes are too close to boundary (1mm).
 - The motor front plate mounting cuts much of the support. Need to make it thicker in depth and radius
- Pitch axis holder
 - M2 screw holes are too close to boundary
 - Need to be thicker in depth
- Arm body holder
 - Adhesion and warpage broke the structure.
 - Need to add more glue, set hot bed at 60C, add brim, and set infill density to 70% to reduce stress



Photographs taken by the author



Photographs taken by the author

11/08/23

Arm assembly screws

		size	type	height	exact	length	count	instance	per arm	total	ordered
Motor bracket	Side plate	M2	non flat	5	7	8	6	2	12	72	
Pitch gear	Side plate	M2	non flat	11.5	13.5	14	7	1	7	42	
Arm pitch axis fixture	Assembly, vertical	M2	non flat	11	13	14	4	1	4	24	
Arm pitch axis fixture	Assembly, vertical	M2	non flat	23.6	25	25	4	1	4	24	
Arm pitch axis fixture	Arm, vertical	M2	non flat	36.2	38	40	2	1	2	12	
Arm pitch axis fixture	Arm axis, horizontal	M2	non flat	57	59	60	1	1	1	6	
Arm servo fixture assy	Assembly, vertical	M2	non flat	11.1	13	14	4	2	8	48	
Arm servo fixture assy	Assembly, vertical	M2	non flat	22.1	24	25	4	2	8	48	
Arm servo fixture assy	Arm, vertical	M2	non flat	36	38	40	2	2	4	24	
Arm servo fixture	Servo plate, horizontal	M2	non flat	22.1	24	25	1	2	2	12	
Arm servo fixture	Guide plate, horizontal	M2	flat	15.9	18	20	2	1	2	12	100
Servo plate	Servo motor	M2.5	flat	11	11	10	8	1	8	48	100
Arm roll gear	Gear and ring	M2	non flat	28	30	30	1	2	2	12	
Arm roll fixture	Group plate	M2	pan	47	47	50	4	2	8	48	
Group plate	Servo motor	M2	flat	5	5	4	4	1	4	24	100
						length			total	ordered	
						8			72	100	
						14			114	120	
						25			84	100	
						30			12	30	
						40			36	60	
						50			48	60	
						60			6		

Table created by the author

Arm roll holder print

- Improved but still adhesion and distortion issue



Photograph taken by the author

- Corner needs to be rounded
- Continue 50C, 60% infill and glue application

Arm body holder print

- Corner rounding and 60° infill produced usable parts.
- Corners are slightly warped still
- Radial pin bearing height should be 16mm, not 12mm.
- Need a new 3d printing



Photograph taken by the author

11/09/23

Arm body holder print

- Not much adhesion issue
- Detach was tough, and raised bed temperature to 80C
- The part was warped during detach
- Need to find out better way to detach



Photograph taken by the author

11/10/23

Two 3d print beds

- Glass and flexible magnetic
- More time to cool down each bed after print
- Detach is much easier after cooling down
- Cooled down parts do not get warped, unlike raised temperature detach at 80C
- Flexible bed should help easier detach
- Flexible bed also need glue. Adhesion fails early without glue.
- Need to measure shrink and dimension from the parts
- Tilt gear prints are good.
 - One of two has a bit more stringing, but it should be useable



Photograph taken by the author

author

- Arm fixture set has better print and definition after increasing screw hole margin to >1.5mm from 1mm



Photograph

taken by the author

XL430 self-threaded holes:

- Opening is 2.1mm, and it needs M2.5mm screw. M2 screw will be freely in and out

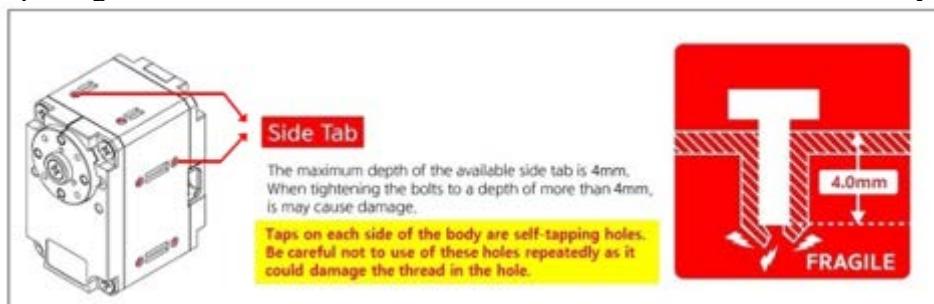


Image taken

from Robotis (<https://www.robotis.us/>)

- Wafer head screw for arm roll servo motor mount/wa
 - M2.5
 - Head diameter=7mm
 - Head thickness=0.7mm
 - Bottom plate design missed this screw head

Servo motor gears were successfully printed and detached with flexible bed

- Subsequent arm holder print did not go well from the initial layer
- Glue might be too dry. Need to apply more.

11/11/23

Overnight arm holder printed went well

- Glue needs to be wet to the bed surface
 - Nozzle height should be definitely lower than 0.2mm
 - Flexible bed makes it extremely easy to detach large and hard prints
- Printed part size measurements

- Printed dimensions are overall close to design. But the caliper measures the maximum of the layers, and it might not be able to capture possible shrink



Photographs taken by the author

- Inner diameter is smaller as the smallest of all the layers define it.
 - Both used 22.2mm to address shrink. Needs more room to improve fit.
 - A tighter fit is better than loose fit
 - Tube diameter is about 22.05mm
 - Gap at tube diameter is about 1.1 or 1.2mm
 - Use 22.5mm to close 0.94mm
 - The printed parts are mostly useable though



Photographs taken by the author

- Arm roll axis holder for a roller bearing placeholder
 - It used 28.1mm per original dimension, and needs a bit more margin
 - Bearing outer diameter is 28.00mm
 - Gap is about 1.2mm
 - Use 28.5mm to close 1.24mm gap -> 28.4mm to fit
 - It does not need to hold bearing tightly



Photograph taken by the author

- M2 screw can be self-threaded
- 2mm bearing fits M2 screw nicely



Photographs taken by the author

11/13/23

Arm fixture set print with adjusted arm tube opening

- Re-print with 22.5mm
- Tight fit. It will be tight when screwed in
- Bearing holder re-print with 28.4mm
- Makes a good and tight fit



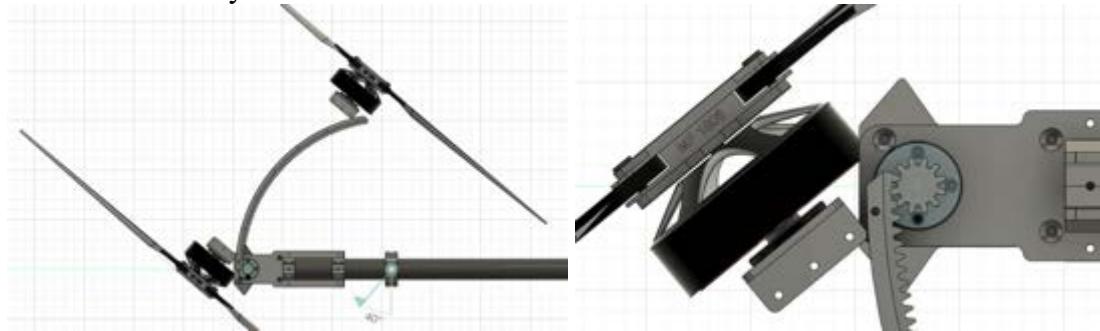
Photographs taken by the author

11/14/23

Tilt pitch limit

- Maximum is less than 40 degree

- Motor hits servo motor block
- Backside guide plate needs to be within the tilt gear diameter to avoid collision with motor
- Need to raise motor by about 5mm and shift away by about 5mm to support 40 degree pitch tilt
- Make minimum 40 degree as tilt target
- Need to make tilt gear does not lose servo completely at extreme angle, so that it can recover eventually



Images created by the author

11/15/23

Target tilt pitch angle: 45 degree

- Propeller height: 122mm
- Propeller radius: 173mm
- Current height = 129.3mm, radius = 192.8mm
- Previous gear: $r=150$
 - Outer diameter = $157*2 = 314\text{mm}$
- New gear
 - Outer diameter = $130*2 = 260\text{mm}$
 - Gear diameter = $123*2 = 246\text{mm}$
- Inner gear works as diameter is less than 250mm
 - Pressure angle = 20 degree

Target tilt pitch angle: 50 degree

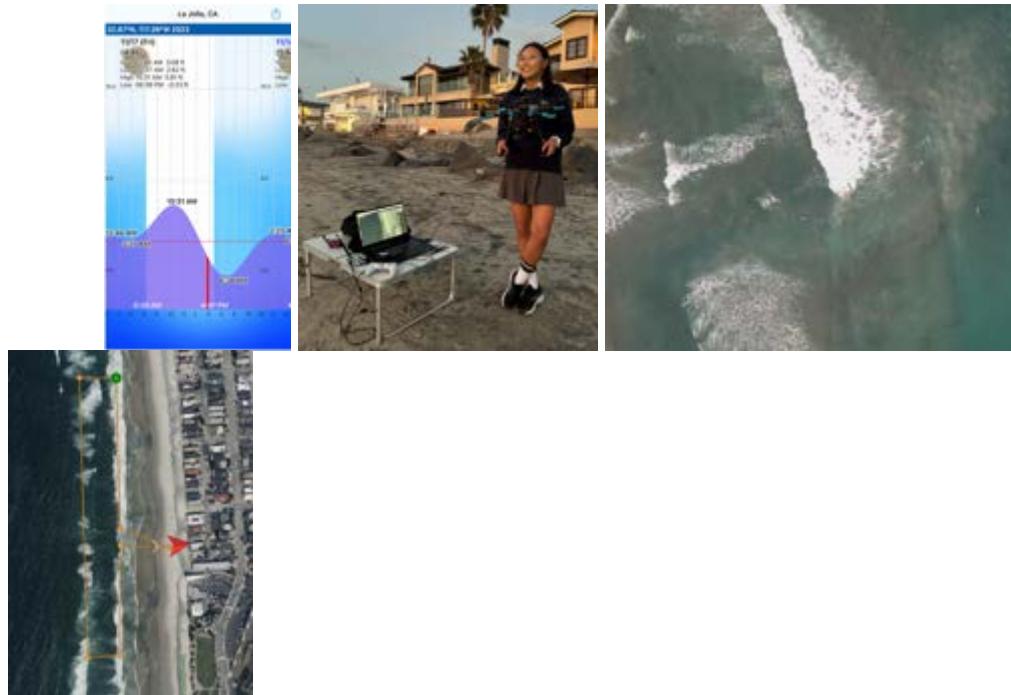
- Height: 142mm
- Radius: 185mm
- X offset: 119mm
- Tilt gear outer radius: $185 - 45 = 140\text{mm}$
- Tilt gear radius: $185 - 50 = 135\text{mm}$
- 50 degree pitch hits arm:
 - Need to re-confirm equation
 - Need to re-generate parameters

*Image created by the author*

11/17/23

2nd beach data collection at 4:30pm

- Tide was going low and the sunset was 4:45p
- RPi CM3 connector was disconnected at last flight – reconnected
- Run #3
 - Drone returned to take off location after a short flight
 - Not sure why it came back. Need to analyze log
 - Gimbal worked. RPi CM3 was out of focus. Need to fix it
- Run #4
 - Drone made a whole flight planned
 - Image storage folder was not named properly
 - No images were saved
- Run #5
 - Sun was almost set
 - Drone made a whole flight
 - Images were stored successfully
- Altitude at 100m seems to work better to cover waves

*Image captured by the author*

Photograph taken by the designated supervisor

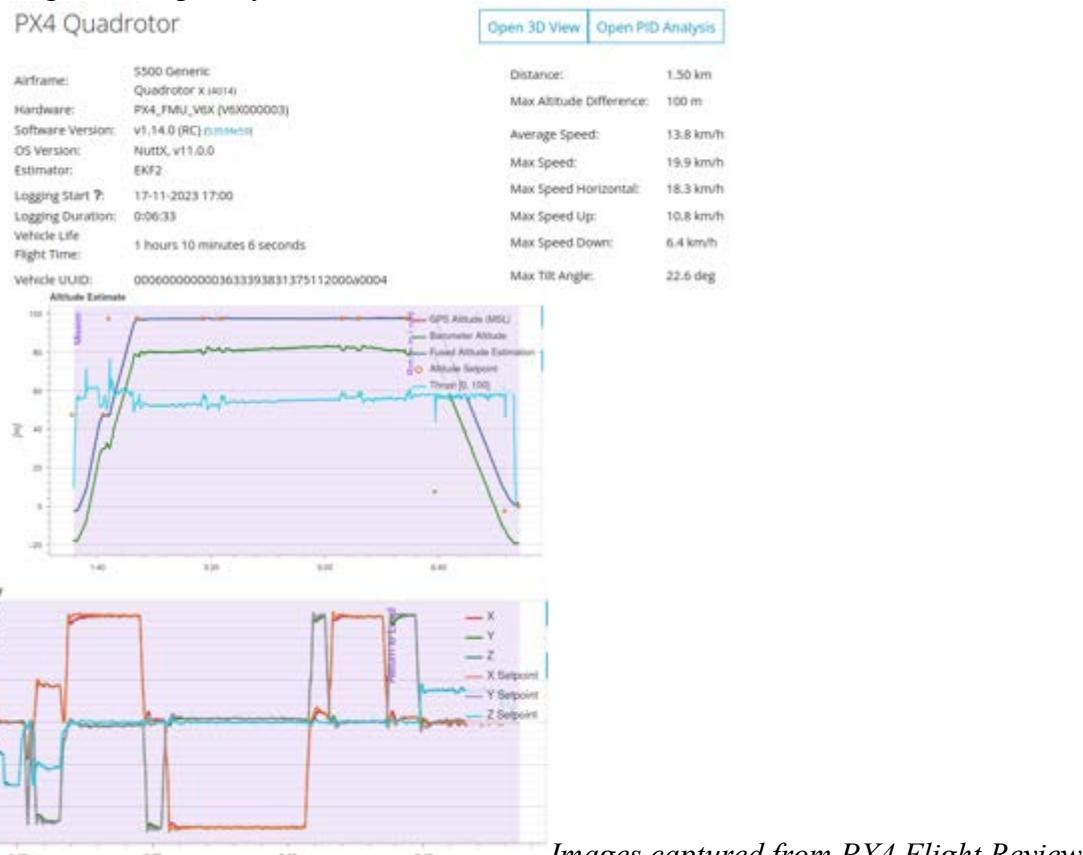
Photograph taken by the author

Image captured by the author

- Things to fix:
- GPS tagging did not work
 - Need to record heading too
 - Need to request more
 - Need to send gimbal command regularly
- Need to wear safety goggle
- RPi CM3 need to have infinite distance focus
- Need to simplify script running, especially name
- Need to simplify video folder naming. It should be created by script
- PX4 refused to arm actuators many times. Need to figure out why.

11/18/23

Flight #05 log analysis



(<https://review.px4.io/>)

Flight #04 failure log

- The “manual control lost” triggered fail safe return to take off position
- Guess: USB joystick control might have been insecure due to a possible table movement

Logged Messages			
#	Time	Level	Message
11	0:01:47	INFO	Takeoff detected
12	0:03:05	INFO	Manual control lost
13	0:03:05	WARNING	Failsafe activated due to manual control loss, triggering RTL in 5 seconds
14	0:03:05	INFO	Manual control regained after 0.4 s
15	0:03:10	WARNING	Failsafe activated due to manual control loss, triggering RTL
16	0:03:10	INFO	RTL activated
17	0:03:10	INFO	RTL: landing at home position
18	0:03:10	INFO	RTL: return at 99 m (100 m above destination)
19	0:03:43	INFO	RTL: descend to 9 m (10 m above destination)
20	0:04:07	INFO	Pilot took over using sticks
21	0:05:04	INFO	Landing detected

Image captured from PX4

Flight Review (<https://review.px4.io/>)

MAVLink does not work with Python

- Updated /fs/microsd/net.cfg and confirmed
 - netman show
 - DEVICE=eth0
 - BOOTPROTO=fallback
 - NETMASK=255.255.255.0
 - IPADDR=192.168.144.19
 - ROUTER=192.168.144.20
 - DNS=192.168.144.20
- Now JETSON can ping PX4
- Python still does not get MAVLink connection message
- Change UDP port to 14550, same as GCS
 - It went through the connection acknowledgement
- It seems the python code needs to wait for PX4 to initialize ethernet to make it functional
- Once the code runs, it has to be rebooted to be functional again
- Frame rate was 1 for gimbal0 and camera0
 - Increased it to 5 frames per second
- Increased frame rate seems to work
 - There are some residue files from camera0
 - Camera0 has focus issue

11/19/23

RPi CM3 autofocus does not work

- The driver does not support focusing
- <https://www.ridgerun.com/post/raspberry-pi-camera-module-3-driver-for-nvidia-jetson-orin-nano>
- <https://forums.developer.nvidia.com/t/ridgerun-rpi-camera-module-3-imx708-open-access-driver-for-nvidia-jetson-orin-nano-and-jetson-nano/261118>
- Ardu Cam might be good alternatives:

- <https://www.arducam.com/product-category/nvidia-jetson-nano-nx-officially-supported-sensors/page/2/>
- <https://www.arducam.com/product/arducam-high-quality-camera-for-jetson-nano-and-xavier-nx-12mp-m12-mount/>
- <https://www.arducam.com/product/arducam-12mp-imx477-motorized-focus-high-quality-camera-for-jetson-nano/>
On-board network topologies were broken when IP air unit was removed
- IP air unit worked as a router for PX4 UDP network
- Now PX4 considers JETSON as router
Improve gstreamer flow
- Now the script creates px4_video folder before gstreamer starts
- If there is a px4_video folder, it is renamed as px4_video_backup. Then a new px4_video folder is created
- Frame rate was 1. Now it is 5 to have ore images
Python GPS tagging
- Need more detailed logging and progress
- All the prints are written to file with time stamp at log file
- Previous code saved each GPS information to each file name.
- All logs are a stored at single file
Alias for easier launch
- “rung” for start_gstreamer.sh
- “runp” for start_python.sh
Px4 network initialization time
- It takes about 80 seconds for px4 to respond to ping and MAVLink can be established

11/20/23

ArduCAM 12M camera

- Compatible with Nvidia Jetson Nano and Xavier NX
- 1/2.3 Inch IMX477 Camera Module with M12 Mount Lens
- Manual focus
Configuration
- <https://docs.arducam.com/Nvidia-Jetson-Camera/Native-Camera/Quick-Start-Guide/#Software>
- sudo /opt/nvidia/jetson-io/jetson-io.py
 - Change to IMX477 at CSI, from IMX219
- /boot/extlinux/extlinux.conf changed
- But camera is not detected
- nvgstcapture-1.0 does not find a camera
- dmesg | grep IMX shows imx477 is getting registered
 - imx708 has register i2c error, but imx477 does not have an error
- i2cdetect does not show any cameras

11/21/23

IMX477 camera debug

- Try Jetson Orin Nano
 - It has a narrower ribbon cable
 - No other HW to confirm if the camera module is not defective
- Try older Jetpack 4.6.3
 - Install Jetpack 4.6.1 on a slower SD card
 - V4l2-ctl is missing.
 - Need to add apt-get repository
 - sudo apt-get install v4l-utils
 - v4l2-ctl –list-formats-ext shows available video format
 - There are errors that cannot recover
 - \$ gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw, format=GRAY8, width=1920, height=1080 ! videoconvert ! autovideosink
 - ERROR: from element /GstPipeline:pipeline0/GstV4l2Src:v4l2src0: Internal data stream error.
 - \$ gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw, format=RG10, width=1920, height=1080 ! videoconvert ! autovideosink
 - WARNING: erroneous pipeline: could not link v4l2src0 to videoconvert0, neither element can handle caps video/x-raw, format=(string)RG10, width=(int)1920, height=(int)1080
 - \$ dmesg | grep imx
 - [1.891206] imx477 8-001a: imx477_board_setup: error during i2c read probe (-121)
 - [1.898825] imx477 8-001a: board setup failed
 - [1.903322] imx477: probe of 8-001a failed with error -121
- Recover to RPi CM3 with the other unit
 - Re-install RidgeRun driver then reboot
 - Comment out IMX477 configuration at /boot/extlinux/extlinux.conf
 - Video output was recovered
- v4l2-ctl provides a few control options
 - <https://www.mankier.com/1/v4l2-ctl>
 - v4l2-ctl –list-ctrls-menus
 - No AF related items.

11/22/23

3rd Del Mar Beach flight and data collection



*Image captured by the author
Photograph taken by the author*

- Flight #6 and #7 conducted
 - Improved flight flow
 - Gstreamer and python launch alias rung and rump worked better
 - Waited 80 seconds for PX4 to establish MAVLink
 - It did not respond after all
 - Two locations
 - 27th street
 - 17th street
 - Longer flight
 - Higher frame rate at 5 frames/second from 1 frame/second
 - Wear eye protection glasses
 - Battery charging
 - #5: 2349mAh / 4200
 - #6: 2742mAh / 4200
- Problems encountered
- Flight #06 had warning: Accel1 clipping
 - It was overridden as the flight was normal
 - VNC did not register keyboard input in some instances
 - Not sure if it is JETSON or laptop
 - Had to restart system a few times
 - Mouse input worked
 - Solution: Make terminal icons to run rung and rump scripts
 - GPS tagging did not work
 - Need to understand why
 - RPi CM3 still does not have a good focus
 - Images are not useable
 - Images have ripple when propellers are actuated

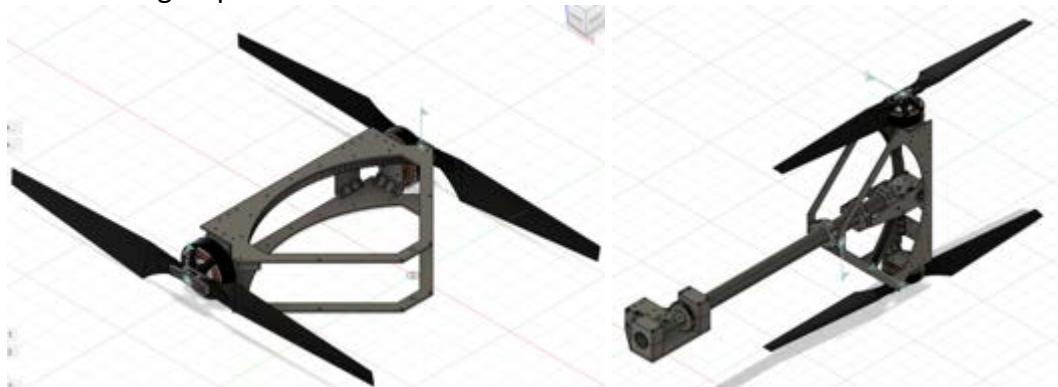
11/23/23

Previous set up - target tilt pitch angle: 50 degree

- Height: 142mm
- Radius: 185mm

- X offset: 119mm
- Tilt gear outer radius: $185 - 45 = 140\text{mm} \rightarrow 142\text{mm}$
 - 4mm rail for mounting = 138mm
- Tilt gear radius: $185 - 50 = 135$
- Equation update
- Pitch max angle and motor angle are same
- For 45 degree
- Height: 136mm \rightarrow increase to 140mm for 45 degree margin
- X offset: 136mm \rightarrow increase to 140mm for margin
- Radius: 192.5mm
- Tilt gear outer radius = 160mm for clearance from motor and motor bracket
- Tilt gear rail radius = 156mm
- Tilt gear radius = 153mm

Revised tilt group based on the calculation



Images

created by the author

- Maximum tilt pitch is about 47 degrees without obstruction
- Added ESC mount, and routing guides
- Need to re-think pitch axis when 5mm bearing is used
 - M3 70mm might be necessary – then need a different bearing.

11/24/23

Tilt gear print

- Poor print results, and both had adhesion issues
- Gear is larger than all others, and the glue was not fully covered.
- Brim was not applied



Photograph taken by the author

Tilt gear update

- Width of the gear increased by 2mm
 - The earlier print felt a bit filmsy

- M2 screws are centered at the mount rail
- Single tilt gear print, rather than two at a time

11/25/23

M2.5 screw wafer head dimension

- d = Thread Diameter: 2.5mm
- dk = Head Diameter: 7mm
- K = Head Height: 0.70mm
- Thread Pitch: 0.45mm

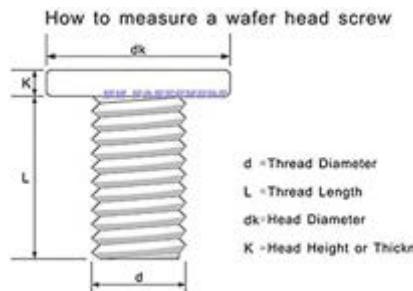


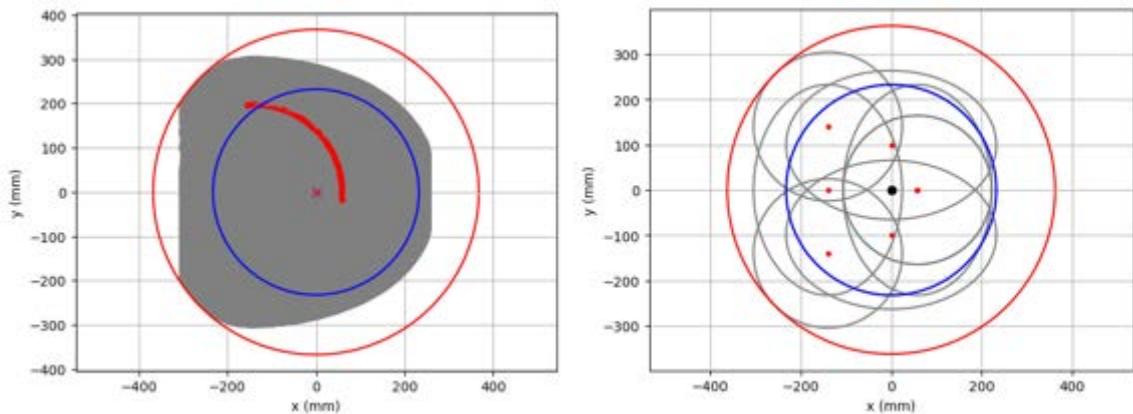
Image captured from Amazon

(<https://www.amazon.com>)

11/26/23

Calculated maximum radius of propeller with pitch and roll

- Updated propeller center coordinate calculation
 - $\text{centerX} = \text{Rcenter} * \text{np.cos}(\text{tiltAngle} + 45 * \text{np.pi}/180) - \text{XOffset}$
 - $\text{centerY} = \text{Rcenter} * \text{np.sin}(\text{tiltAngle} + 45 * \text{np.pi}/180) * \text{np.sin}(\text{rotateAngle})$
- 1-degree step at pitch and roll to search the maximum radius case
 - Pitch: -45 to 45
 - Roll: -90 to 90
- The maximum radius from the reference case is
 - Maximum radius: 367.5mm
 - Happens with +/-52 degree roll and 50 degree pitch



Images created by the author

- The maximum radius defines the size of the drone as it happens at inner side
 - For full freedom of pitch and roll at each propeller
 - 380mm: $12.5 \times 2 = 25$ mm margin between propellers

- 390mm: $22.5 \times 2 = 45$ mm margin between propellers
 - Arm axis clearance is about -306.6mm from the center of the reference case

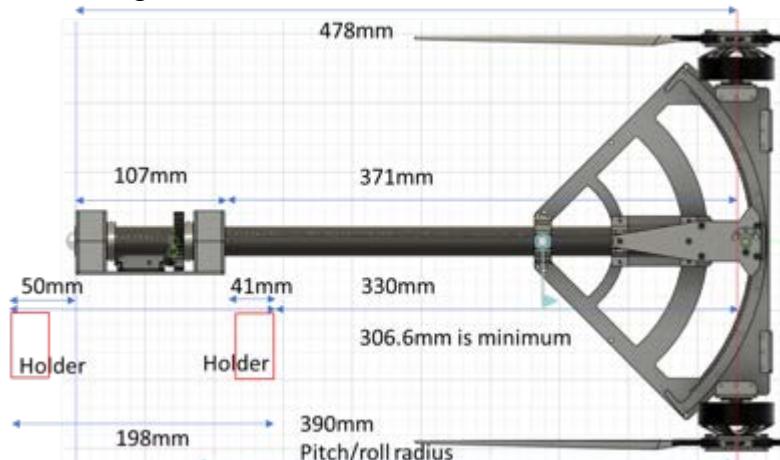


Image created by the author

11/28/23

CNC prep

- Plexiglass: 24 x 36 inch = 609.6 x 914.4 mm
 - As waste board
 - Cut to 24 x 18 inches
 - Thickness?
 - CNC area: 19 x 29 inch = 482.6 x 736.6 mm
 - Need to plexiglass board to fit CNC area
 - Use double-sided rug tape to fix
 - Waste board to CNC
 - Carbon fiber plate to waste board
 - Maximum CNC working area
 - $375 \text{ mm} \times 575 \text{ mm} = 14.8 \times 22.6 \text{ inch}$

Initial CNC try out

- 200x300 plate
 - 1mm end mill
 - 10000 rpm, 300mm/min feed rate, 100mm/min plunge rate

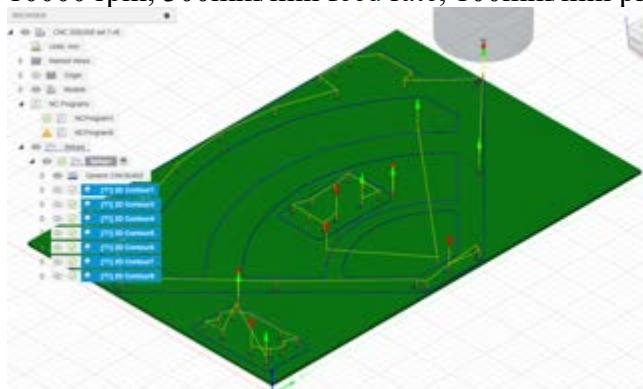


Image created by the author

Rescue drone CAD build up status

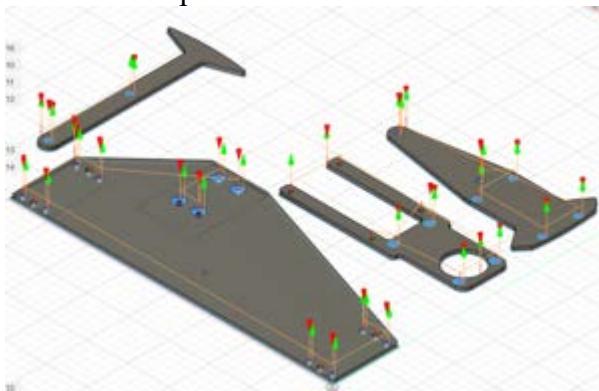
*Image created by the author*

- Maximum diameter is about 2m, or 1.98m
- Three folding possible
 - Propeller
 - Roll arm
 - Extension arm

11/30/23

CNC flow ready for set #2 with 3D cut

- Spiral for flat
- Radial for slope

*Image created by the author*

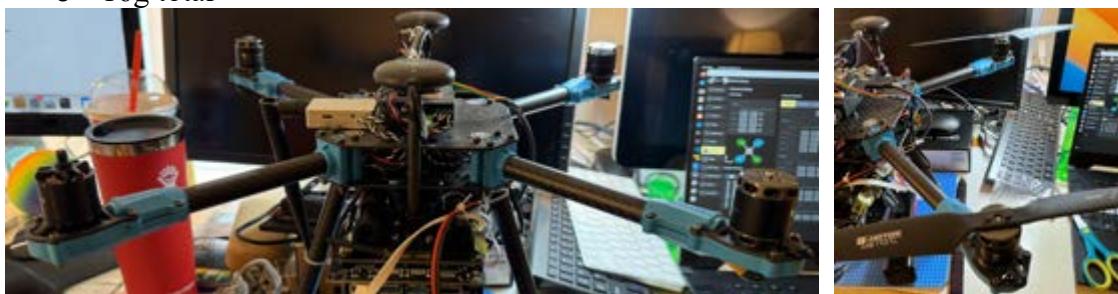
12/01/23

Scout drone motor and propeller upgrade

- Motors
 - 2317's AS150 connector should be converted to MR30
 - Same connection mapping for CW and CCW
 - CW: straight 3 wires
 - CCW: twist #1 and #2

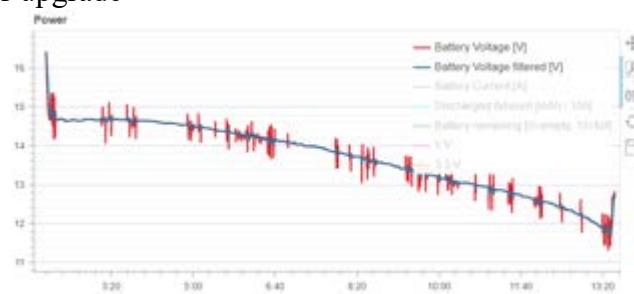
*Photographs**taken by the author*

- Weight comparison
- Actuator
 - 2216: 49g
 - 2317: 57g
- Propeller
 - 10 inch: 22g
 - 11 inch: 15g, excluding screw
- Remove actuator bottom cap and screws
 - 16g total

*Photographs taken by the author*

12/02/23

Test flight after motor and propeller upgrade

*Photograph taken by the author**Image captured from PX4 Flight Review (<https://review.px4.io/>)*

- Battery #5
 - Flight time about 11 minutes with battery #5 4200mAh
 - 3426mAh charged after flight out of 4200mAh capacity
- Large capacity battery issue
 - Two 8400mAh batteries does not take off
 - Battery voltage goes down to 6v immediately after propellers are rotating intermittently
 - They must be defective. Charging does not finish in normal manner.

First CNC milling

- 200x300 set #1
 - Not deep enough at 2.2mm
 - Discarded
 - Too much dust
- Next milling
 - Extend to depth to 2.7mm
 - Attach vacuum
 - There was a mistake with holes – not deep enough. Might need manual drilling



Photographs taken by the author

300x400 cnc set



Image created by the author

12/03/23

200x300 set #2 CNC milling

- Arm board holes were incomplete
- Arm board step holes were very shallow and needed repeated milling
- Possible mistake in milling depth in the original CNC instruction
- Or the plate had a slope to cause under milling at the board
- Arm board outside boundary was cut nicely though

- There was an incident where a mill bit penetrated the board, and eventually CNC platform
 - The mill bit might have caused uneven level on the other side of the board, where arm board was
 - Need to pull the bit stuck in the board. The end bit was embeded in the CNC platform.

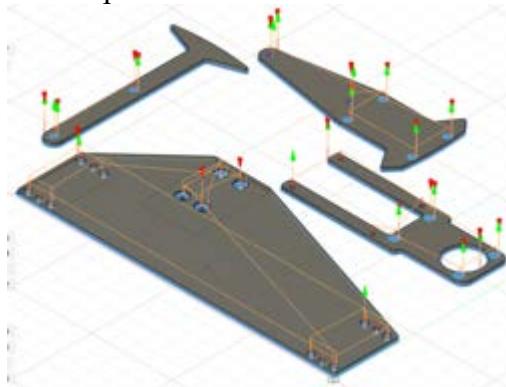


Image created by the author

Next 300x400 milling

- Reference point to top of the board
 - Do not need to add board thickness to auto Z calibration
- Stock thickness is same as board thickness
 - Simple calculation

12/04/23

Change material to Delrin (Acetal)?

- It is questionable if the dust is at safe level even after vacuum with dust bag and HEPA filter (95% filtering)
- Need water emersed milling with carbon fiber plate

12/08/23

Emersion milling

- Set up
 - Clamp
 - Bake tray
 - Double sided tape
 - 1mm waste board plexiglass
 - Double sided tape
 - Carbon fiber plate
- Process
 - Perform Z calibration
 - Z probe does not work if there is water
 - Reset machine coordinate – sometimes initial calibration trigger Z trip switch
 - Dry run
 - Add water 5 bottles
 - Align X and Y

- Check Y max limit
- Start milling
- Things to fix
 - Need new double sided tape
 - Tape itself sticks around the mill bit to cause lots of splash in the middle of milling
 - It breaks eventually due to the tape residue



Photographs taken

by the author

Strategy

- It is very hard to planarize 300x400 plate, plexiglass, and bake tray
- Might need to separate
 - Full 2D cut parts – use 300x400
 - 3D milling parts – use 200x300 plate

12/09/23

King tide in San Diego in December

<https://www.nbcstandiego.com/news/local/king-tides-roll-across-san-diego-waters/3370313/>

Emersion milling

- All intermediate step milling went through the 2mm board
 - Maybe the tape absorbed water and raised the whole platform?
- Changed tapes to 3M tapes with waterproof property
- 3M 9448A:
 - https://www.amazon.com/dp/B0BWZ1JCQB?ref=ppx_yo2ov_dt_b_product_details&th=1
 - Thinner tape to avoid adhesive sticking to mill bit
- 3Mseries 1600T PE foam tape:
 - https://www.amazon.com/dp/B09XLJ6CC9?ref=ppx_yo2ov_dt_b_product_details&th=1
 - Bottom tape needs thickness to avoid damage to bake tray
- Clamp needs to hold the bake tray plate
- Additional anchors were cut out of plexiglass pieces
 - Anchors took too much area, and it could collide with spindle
- Made additional narrow pieces for anchoring
 - Mill bits are breaking
- Reduced feed rate to 100mm/min from 150mm/min

- Still breaks
 - Reduced feed rate to 50mm/min from 100mm/min
 - Multiple pass depth to 0.5mm from 0.7mm
 - Milling run time becomes 12 hours
 - Bit broke after about 1 hour additional run
 - Vendors recommend to fasten the mill bit longer
 - It still breaks about in an hour
 - Maybe thicker mill bit at 1.5 or 1.8mm would be better?
- Milling parameter calculation
- <https://www.amanatool.com/pub/media/custom/upload/File-1436543087.pdf>
- Milling process
- Need to wet the tapes before Z calibration, to address Z changes?
 - 2D step milling does not work
 - Half steps are all punched through. Maybe use 3D milling?
 - Need to remove manual bottom height, and use contour without offset

12/11/23

One set milling finished

- Broke 4 of 1mm milling bits
- 2mm bit finalized cuts
- There are sharp edges. They will need smoothing



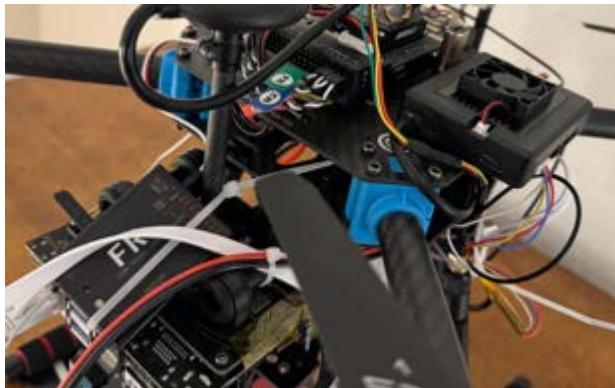
Photographs taken by the author

- Need to review waste board milling depth
- Need to repeat 3D milling parts to avoid full punch through
- Need to use 1.5mm or 1.8mm bit for whole milling process

12/14/23

Installed air IP radio with antenna and RC receiver

- Need to clean up wiring tighter to avoid propeller



Photograph taken by the author

12/15/23

3D milling parts on 200x300 board

- 2D step milling was successful
- During part boundary cut, clamp collapsed. Failed to cut all the parts.



Photograph taken by the author

- Need to enforce clamping
 - Horizontal shift needs to be blocked at the bottom of tray
 - There were 4 clamps over the tray plate using spacer. Need to tighten them further
 - Need to place 8 additional clamping. 2 stops at each side of the tray
- Next cut: 300x400 with full set milling
 - Use 1.8mm mill bit
 - Use 2D adaptive milling on screw steps
 - Use outside boundaries for screw slopes to achieve smooth slope

12/17/23

Whole arm set milling

- Added 8 more clamps. Total 12 clamps
- Thin double-sided tapes at board and waste board



Photographs taken by the author

- Finished all milling over about 12 hours. First successful finish without mill bit breakage
- South of the board got under milling slightly even with 2.5mm milling at 2mm board. There must have been a slope. Will need to clean up or drill the parts.
 - Milling depth should be increased to 2.6 or 2.7mm
 - Need to examine waste board damage with 2.5mm set up
 - 2D milling parts are okay and also easy to fix
 - 3D milling parts will be hard to fix comprehensively
 - Might need a separate 200x300 milling of 3D parts.
- Taking out parts took about an hour
 - Double-sided tape is not easy to remove part
 - Under-milled corner is tougher to remove
- Cleaning the sticky tape out from bake tray took about one hour
 - While tape held the parts and plate well, it is hard to remove
 - Thin sticky tape might not have much benefit in keeping plate flat
 - Thicker rug tape with better detaching might be better

12/17/2023

Bake tray cleaning

- It takes about an hour per each tray
 - Also, significant effort, goo-gone, soap, and paper towel
 - Need to use easier double-sided tapes to reduce efforts
- NOAA tidal forecast
- King tides around Christmas
 - <https://tidesandcurrents.noaa.gov/>

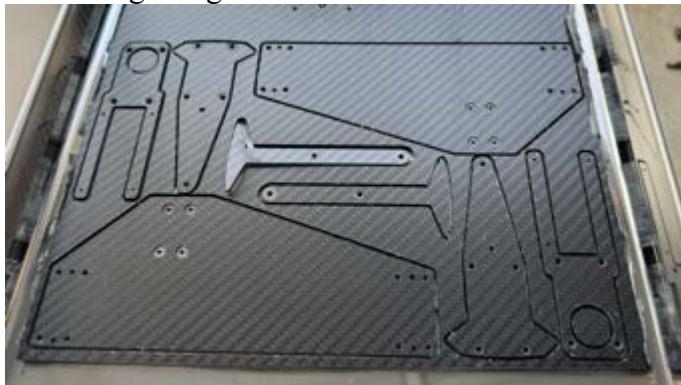


Image captured from

NOAA (<https://tidesandcurrents.noaa.gov/>)

12/19/2023

3D milling using 300x400 half board



Photograph taken by the author

- It went through good overall
- Detaching is much easier, can be done in 10 minutes with the rug double-sided tape

12/21/2023

Scout drone to-do's

- Essential
 - Dry run at home
 - Test flight with additional payload
 - Check remote video streaming
 - Remote control intervention
 - King tide beach data collection
- Good to have
 - GPS coordinate recording
 - Replace flight controller for serial MAVLink, from Ethernet
 - Add light strobe(s)
 - Add forward gimbal in place of RPi camera module 3

Rescue drone to-do's

- Build roll and pitch arm
 - Clean up CNC parts
 - Clean up 3D print parts
 - Drill holes
 - Test roll and pitch
- Build body and landing gear
 - CAD and CNC parts
 - Mounts for GPS, controller, power distribution, battery, etc.

Scout drone configuration update

- Video stream forwarding to ground station is enabled
 - IP air unit and ground radio pair works
 - VNC sometimes does not accept keyboard input at all
 - Not sure what causes it. Mouse works
 - The only way to get out is to reboot
- MAVLink connection does not work
 - Maybe need to reset px4 ip set up?
 - echo DEVICE=eth0 > /fs/microsd/net.cfg
echo BOOTPROTO=fallback >> /fs/microsd/net.cfg
echo IPADDR=192.168.144.19 >> /fs/microsd/net.cfg
echo NETMASK=255.255.255.0 >>/fs/microsd/net.cfg
echo ROUTER=192.168.144.11 >>/fs/microsd/net.cfg
echo DNS=192.168.144.11 >>/fs/microsd/net.cfg
 - JETSON cannot ping px4. Others (air unit, gimbal, ground unit responds)
 - Px4 fails to ping other devices
 - ERROR: sendto failed at seqno 0: 114

12/22/2023

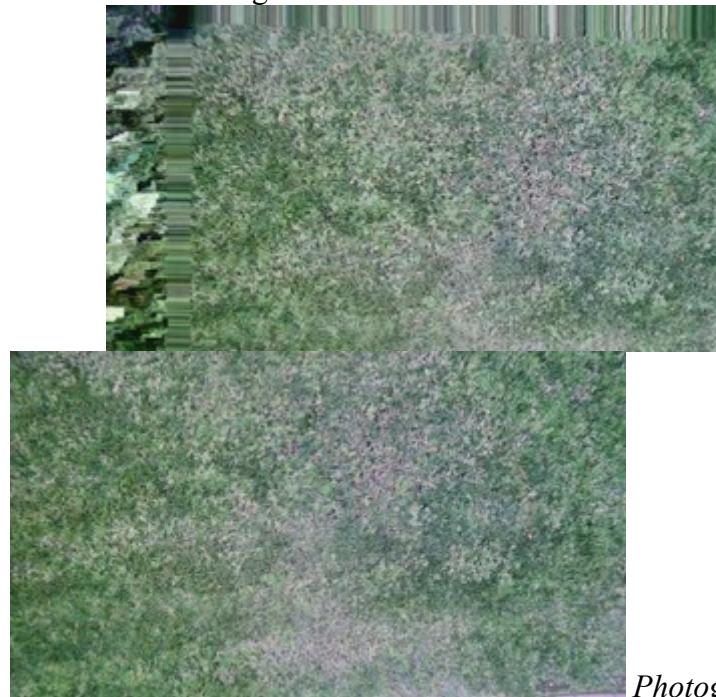
Rescue drone assembly

- Pitch gear seems to fit side plate
- Need to
 - Clean 3d printed parts out of brim
 - Label gears to avoid confusion later
 - Grind milled parts' sharp edges
- Double check TELEMr port configuration
- Bring back TELEMr – 3 as serial ports
 - Disable Ethernet
- Install v1.14 release upgrade
 - <https://github.com/PX4/PX4-Autopilot/releases>
 - https://docs.px4.io/main/en/dev_setup/building_px4.html
 - https://docs.px4.io/main/en/contribute/git_examples.html
 - git fetch origin release/1.14
 - git checkout release/1.14
 - make submodulesclean
 - make px4_fmu-v6x_default

- [1125/1125] Creating /home/akim/PX4-Autopilot/build/px4_fmu-v6x_default/px4_fmu-v6x_default.px4
- TELEM2 still not working
Scout drone test flight check points
- Set up
 - IP radio
 - Telemetry
 - RC controller
- Check points
 - Take off with payload
 - Video streaming
 - RC controller interception
 - Add additional payload
 - Flight duration and battery consumption

Test flight

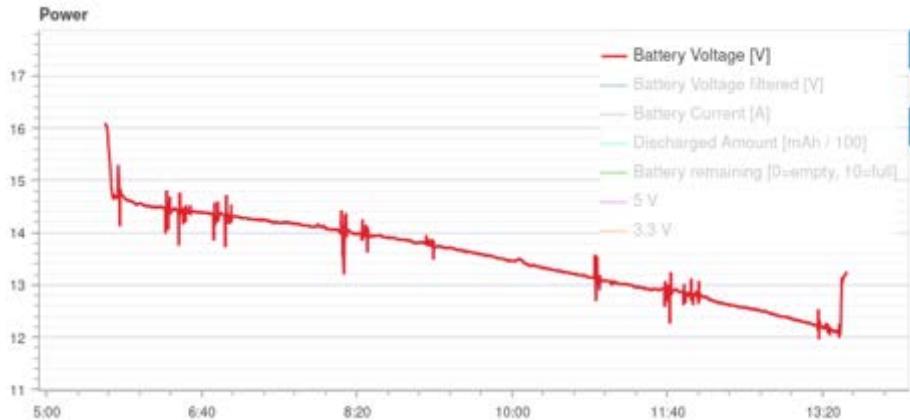
- The scout drone took off with the additional payload
 - Air unit radio and antenna
 - RC receiver
 - USB connector
- Air unit enabled wireless VNC log in to JETSON
 - Video streaming worked intermittently, especially during flight
 - A few images have incomplete boundary. It is temporary and recovers in about 10 images
 - Not sure about the root cause
 - Image 386 vs. 387



Photographs taken by the author

- Flight log duration: 7 minutes 58 seconds

- Battery charged 3251mAh
 - Battery was warm to hot after the flight
- There was USB disconnect issue
 - ttyACM or ttyUSB ports disappeared
 - USB connect came back after a new PX4 controller was connected through USB

*Image*

captured from PX4 flight review (<https://review.px4.io/>)

GPS tagging enabled with TELE3

- Remote ID is connected to TELEM2
- TELEM3 at 115200 BAUD
- GPS tagging works

12/23/2023

Scout drone update

- Clean up the wiring
- Added strobes at front and back
- It should assist visual tracking during flight, in addition to night or dawn flights

*Photograph taken by the author*

12/24/2023

Flight checklist updated with the latest configuration

Phase	Items	Check	Note
-------	-------	-------	------

Prepare	Checklist		
	Drone		
	Drone batteries, charged		
	HM30 ground radio and batteries, charged		
	2.4GHz datalink ground radio		
	RC controller, charged		
	Telemetry joystick, charged		
	Folding table		
	FAA and FCC documents		
	Recording camera (GoPro, iPhone)		
	Linux laptop, charged		
	Spare propellers		
	Screw drive, hex wrench, zip tie		
	Safety goggles or glasses		
	QGroundControl offline map		
Preflight	Set table		
	Set laptop		
	Set HM30 ground radio		
	Set telemetry radio		
	Secure surrounding safety boundary		
	Start QGC		
	Create flight plan		
Flight	Install drone battery		
	Power up drone, RC controller, ground radio		
	Telemetry, data feed, and RC check		
	Ping check from laptop (192.168.144.xx) 12 - Ground unit, 11 - Air unit 20 - JETSON, 25 - IP camera		
	Upload flight plan		
	VNC login to JETSON (192.168.144.20)		
	Set recording folder		
	Start JETSON recording ("rung")		
	Start JETSON image GPS tagging ("runp")		
	Start laptop video stream display command ("runs")		
	Power on strobes		
	Confirm safety		
	Arm actuators		
	Start flight path		
	Land drone		

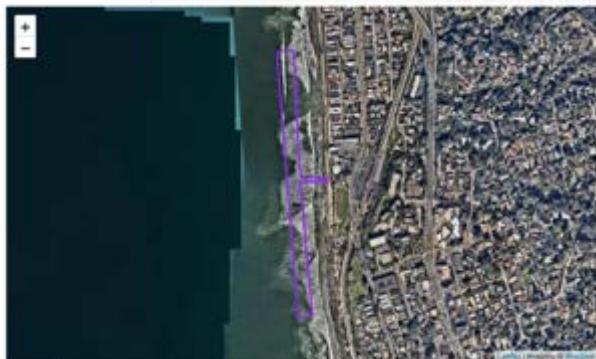
	Disarm actuators		
	VNC login to JETSON and shutdown ("shut")		
	Power off drone and strobes		
	Disconnect battery		
Postflight	Maintain drone		
	Check images recorded		
	Review procedure		

Table created by the author

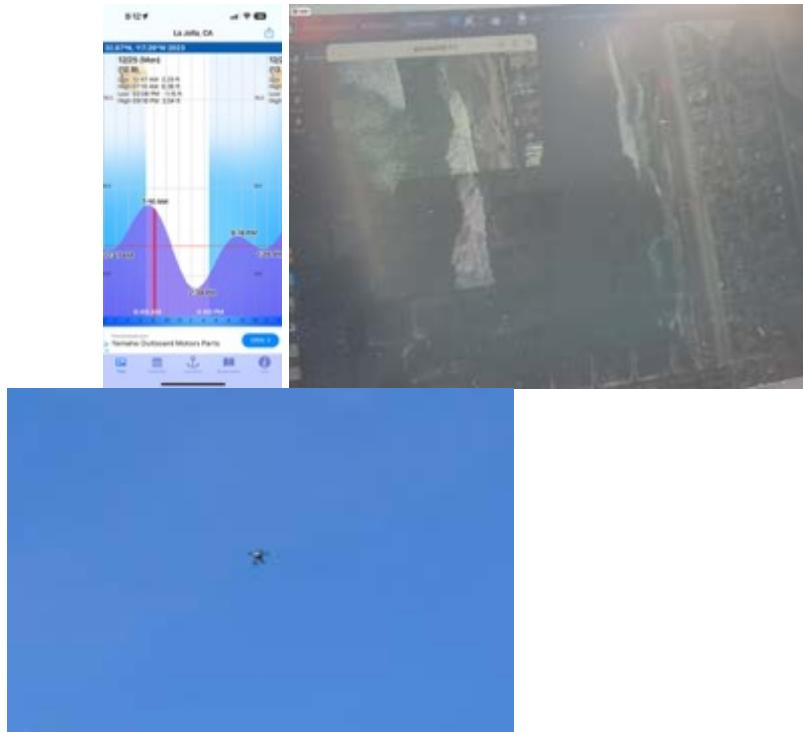
12/25/23

Beach data collection

- Flight #8 and #9 completed at Del Mar Beach. Flight #8 was at 26th-27th street, flight #9 was at 15th street. Went today because of king high tide, which was supposed to have more rip current.

*Images taken from PX4 flight review**(<https://review.px4.io/>)*

- Batteries #5 and #6 recharged after flight to 2829 and 2646 mAh.
- Tested new motors—more powerful, took up more battery power. Used new propellers
- Added RC controller (for emergencies)
- Live video G Streaming of cameras on drone
- Strobe was effective to assist vehicle tracking



*Image captured by the author
Photographs taken by the author
Flight log so far*

Flight #	Date	Time	Duration	Distance (km)	Altitude (m)	Location	Take off	Frame rate	# of images	Data size (MB)
1	10/28/23	7:22	0:04:40	1.03	50	Del Mar beach	18th street	1	0	0
2	10/28/23	7:39	0:04:48	1.04	50	Del Mar beach	18th street	1	289	99.3
3	11/17/23	16:33	0:03:22	0.53	100	Del Mar beach	18th street	1	88	27.9
4	11/17/23	16:42	0:06:45	1.52	100	Del Mar beach	18th street	1	0	0
5	11/17/23	17:00	0:06:33	1.50	100	Del Mar beach	18th street	1	299	58.5
6	11/22/23	8:19	0:08:38	2.09	100	Del Mar beach	27th street	5	1950	499
7	11/22/23	9:01	0:07:27	1.75	100	Del Mar beach	18th street	5	1660	441.5
8	12/25/23	9:43	0:08:29	2.04	100	Del Mar beach	27th street	10	3995	2120
9	12/25/23	10:14	0:07:52	1.90	100	Del Mar beach	15th street	10	3600	1900
10										
11										
			0:58:34	13.40					11881	5146.2

Table created by the author

- 7 effective flights, out of 9 total flights
- 11.8k images, excluding setup, takeoff and landing
- 13.4km flight mileage
- 58 minutes flight duration
- 5.1GB image data

Arm assembly try:

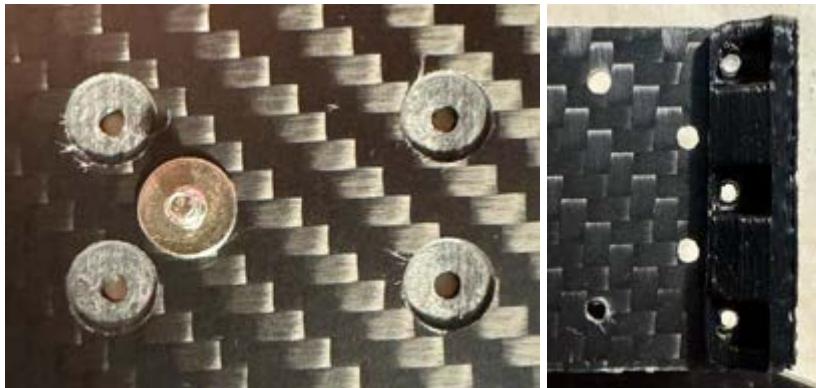
- Pitch gear is short. Need expansion in radius by 2-2.5mm
 - If force assembly, side plate bow
 - At 5th hole, 2mm opening completely misaligns
 - Linear length: $5 * 49.7 / 4 * \pi / 180 * 160 = 173.485$

- Scaling factor: $175.485/173.485 = 1.0115$, or 1.15%
- Last hole 8th offset = 280.777 vs 277.577 about 3mm difference
- Drill hole needs to be 2.05mm
- Arm holder is short. Need expansion by 1mm
 - Might use 1.2% expansion
 - Widen holes to 2.05mm
- Arm plate M2 hole is too tight
- Servo front plate need to widen M2 holes
- Need electric screw driver
- 3d printing
- Pitch gear first to fit
- Nylon is popping even when humidity is 15% at dryer and dried for several hours
- Need to dry overnight

12/26/23

Pitch gear

- Diameter at 2.1mm
- Expansion by 1.2% = 1.012
- 3d print is slightly larger than the milling parts
 - Need to try 1.1%
- Try 100% to be sure considering the big offset
3d print list
- Arm holder
- Servo holders
 - Wafer head screw diameter = 7.2mm
 - Thickness = 0.65mm
 - CNC opening = 6.21mm
 - Need to enlarge opening by 1mm with better centering
 - Measurements are: 6.46, 6.64, 6.27, 6.44
 - Try 8.2mm for margin
 - M2 head size also needs to be bigger by 0.5mm at least
 - Measurements are 3.44, 3.47, 3.55, 3.63
 - Try 4.6mm for margin
- Servo arm and tilt arm fixture set
- Motor and ESC brackets also need to be enlarged



Photographs taken by the author

12/28/23

Tilt gear printing

- 12/26 night printing failed from adhesion failure
- 12/27 night printing went well
- Original 101.2% expansion calculation was done based on an old and smaller radius print
 - Comparison with milled parts is not valid

Arm assembly

- 12/27 night printing tilt gear
 - Fits well with milled side plate
- Motor plate: self threaded + nut
 - Need to widen plate opening to 2.1mm
- ESC plate: self threaded
 - Need to widen plate opening to 2.1mm



Photographs

taken by the author

Fabrication

- CNC 200x300 3D parts with widened step opening and +0.1mm screw holes
 - Removed lead in and out
 - Reduced additional step opening to 4.2 and 7.6mm
- 3D print arm folder with +0.2mm screw holes

12/19/23

3d milling

- Holes are still tight and step holes are small
- Use “Stock to leave”
 - -0.1mm for holes
 - -0.2mm for slopes
 - -0.5mm for step holes
- 3d print
- Tilt and servo fixtures with 2.1mm vertical screw holes and 2.2mm horizontal holes

01/01/24

Ocean wave definitions

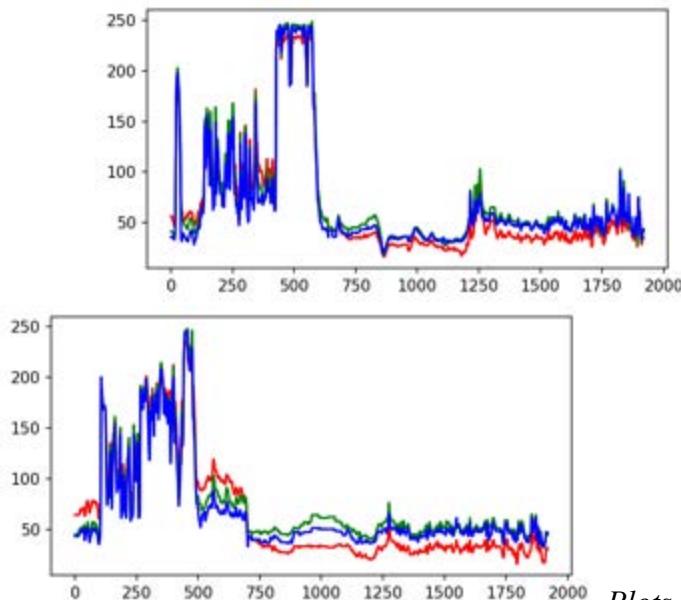
- <https://mrvanarsdale.com/marine-science/online-textbook/chapter-6-waves/>
- Wave crest
- Wave Trough
- Wave Height
- Wave Period
- Wavelength
- Breaking waves – plunging waves, spilling waves
- Swash
- Backwash
- Longshore currents

Beach image analysis



Plot created by the author

- RGB can be separated and scaled to maximum and minimum
- RGB balance shows water (green and blue) and sand (red)
- White – to dark transition from right side can be considered as approaching wave

*Plots created by the author*

01/02/24

Arm assembly progress

- Servo backplate needs to make a cut for motor cable
- Arm holder almost self-thread screws. Do not necessarily need nuts
- Change of plan for servo plate
 - Avoid drilling arm at axis. If needed drill vertically.

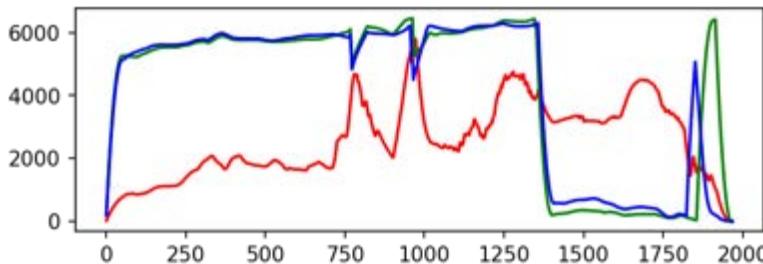
*Photographs taken by the author*

01/03/24

Arm assembly

- 22.6mm 3d print is still too tight to fit 22mm tube
- Printing 22.8mm opening
 - It is a bit too big

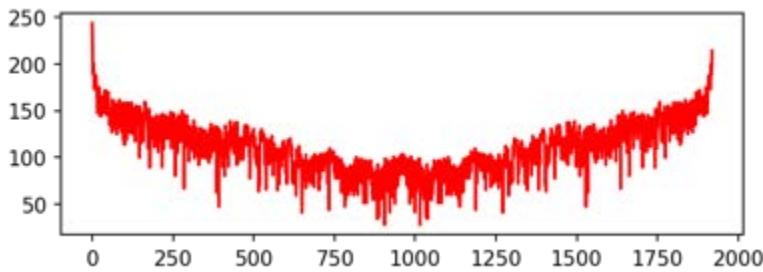
- There seems to be part to part differences
- Reduced 22.7mm and printing 2x to check fit
- 22.6mm ring does not print well
 - Extended outer diameter to 35mm to improve 3d print
- Wave analysis
RGB differential convolution might be useful to identify beach and water
- sandConvR=np.convolve(lineR,sandFilterR)
- sandConvG=np.convolve(lineR-lineG,sandFilterG)
- sandConvB=np.convolve(lineR-lineB,sandFilterB)



Plot created by the author

Fourier transform of a line of R

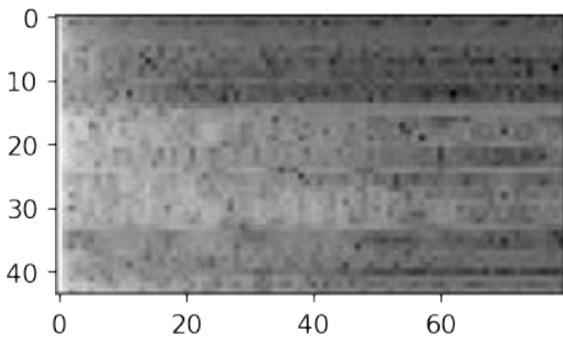
- Reveals frequency component with mirroring at center



Plot created by the author

Window and shift of a R line - spectrogram

- It can show wave breakage in white color and high frequency components



Plot created by the author



Plot created by the author

01/05/23

Landing gear design

- Triangular landing bars
- 50cm carbon fiber tubes
- 3D printed connectors
 - Need to confirm printing feasibility



Images created by the author

01/06/24

Main frame mounting

- Pixhawk 6x: <https://docs.holybro.com/autopilot/pixhawk-6x/dimensions>
 - GrabCAD drawing has 42.8 and 57.5mm

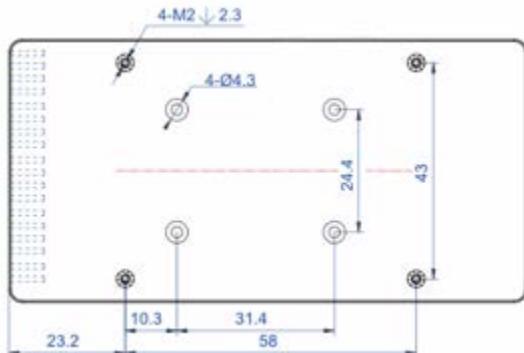


Image captured from Holybro

(<https://docs.holybro.com/autopilot/pixhawk-6x/dimensions>)

- APD PDB
 - M3 for 30.5mm x 30.5mm
- Battery strap holes
- GPS or mounting bracket
 - <https://docs.holybro.com/drone-development-kit/px4-development-kit-x500v2>
- U2D2 power hub
- Dynamixel distributors
- JETSON Orin Nano

- Camera mount x2
- Daughter plates
 - Upper for servo control
 - Lower for battery mount



Image created by author

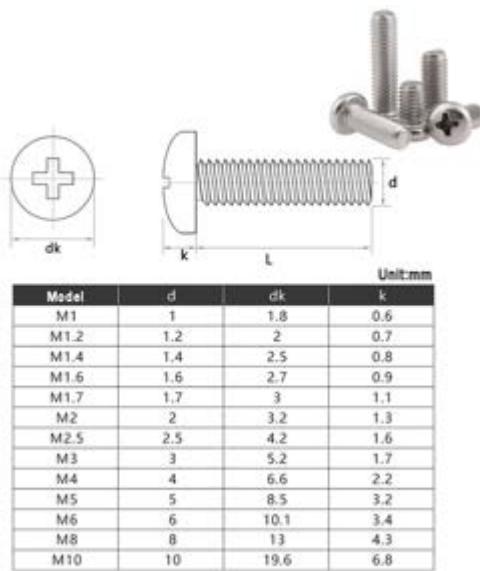


Image taken from Amazon

(<https://www.amazon.com/>)

APD PDB 500

- <https://docs.powerdrives.net/products/pdb>

01/08/24

Arm assembly

- Arm roll working – still need sliding screwing of the servo motor
- Pitch works with a simplified setup



Photographs taken by the author

Things to fix

- Servo holders need to be tighter
 - Diameter =22.4mm from 22.5mm
 - Axis hole diameter to 2mm from 2.2mm
- Pitch axis need to tighter to be an axis
 - Diameter =22.4mm from 22.5mm
 - Axis hole diameter to 2mm from 2.2mm
- Side plate needs to be larger to allow 5mm bearing
- Pitch gear needs to be lower at 7.5mm from 8mm to allow rotation
- Motor and ESC bracket with fully self-threaded assembly

01/10/24

Servo and tilt fixture update

- It assembles better and allows self-threading
- Need to tighten horizontal holes a bit
- Servo back plate to arm needs to use flat head light guide plate to avoid scratching side plate
300x400 cnc milling
- There was a stop, and it had to resume
- Overall good milling
- Arm plate stepped pockets were missing
Arm is ready to build duplicates
- Need to establish 24-hour 3d print and cnc milling cycle
Updated axis mount assembly
- So far, works good



Photographs taken by the author

01/11/24

Fab

- 300x400x2 cnc milling
 - Overall successful
 - Arm plate has one missing step pocket
- Arm fixture set 3d printing went well
- Start 2nd arm assembly
Updated screw table
- Total 1,368 screws if assembled fully

Description	Attach to	size	type	length	height	exact	count	instance	per arm	total
Motor bracket	Side plate	M2	Pan	8	5	7	6	2	12	72
Pitch gear	Side plate	M2	Pan	14	11.5	13.5	7	1	7	42
ESC bracket	Side plate	M2	Pan	6	4	6	6	2	12	72
Servo gear core	Motor	M2	Flat	8	8.5	8	3	2	6	36
Servo gear mount	Motor	M2	Flat	6	7.5	7	4	2	8	48
Servo gear mount	Motor	M2	Pan	6	6.5	6	4	2	8	48
Arm pitch fixture axis	Assembly, vertical	M2	Pan	25	23.6	25	4	1	4	24
Arm pitch fixture axis	Arm, vertical, drill	M2	Pan	40	36.2	38	2	1	2	12
Arm pitch fixture axis	Axis, longer	M2	Pan	12			2	2		24
Arm pitch fixture axis	Axis, longer	M2	Pan	10			2	2	4	24
Arm pitch fixture axis	Axis, shorter	M2	Pan	8			4	2	8	48
Arm pitch fixture axis	Pitch block mount	M2	Pan	14			2	3	2	12
Arm servo fixture assy	Assembly, vertical	M2	Pan	25	22.1	24	8	2	16	96
Arm servo fixture assy	Arm, vertical, drill	M2	Pan	40	36	38	4	2	8	48
Arm servo fixture	Servo plate, horizontal	M2	Pan	25	22.1	24	1	2	2	12
Arm servo fixture	Guide plate, horizontal	M2	Flat	12			2	2	4	24
Arm servo fixture	Back plate, horizontal	M2	Flat	12			2	2	4	24
Arm servo fixture	Axis, longer	M2	Pan	10			2	4	8	48
Arm servo fixture	Axis, longer	M2	Pan	8			2	4	8	48
Arm servo fixture	Axis, shorter	M2	Pan	6			4	4	16	96
Servo plate	Servo motor	M2.5	Flat	10	11	11	8	1	8	48
Arm roll gear	Gear and ring	M2	Pan	30	28	30	1	2	2	12
Arm roll fixture	Group plate	M2	Pan	40	47	47	4	2	8	48
Group plate	Servo motor	M2.5	Wafer	4	5	5	4	1	4	24
Group plate to body	Body	M3	Pan	60	54	54	12	1	12	72
Extender to	Frame	M3	Pan	70	62	62	7	1	7	42
Extender, vertical	Extender	M3	Pan							
Extender, bracket	Bracket	M3	Pan	10			5	4	20	120
Extender, bracket	Bracket	M3	Pan	8			5	4	20	120
Extender, frame	Body frame	M3	Pan							
Landing leg, frame	Tube	M3	Pan	10	7.5		4	3	12	32
Landing foot, leg	Tube	M3	Pan	8			1	12	12	32

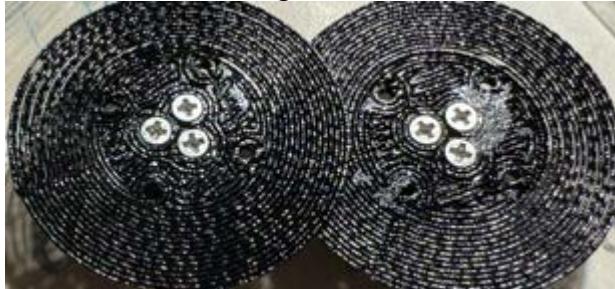
Table created by the author

01/12/24

Fab

- 300x400 milling successful
 - Missing step was fixed
- Gear set 3d print
 - Core-enforced servo motor gear

Enforce servo motor gear with screws



Photograph taken by the author

01/13/24

Fab

- 300x400 milling successful
 - Arm fixture 3d print successful
- Components to fab

		1	2	3	4	5	6
Motor mount	3D print	0	0	0	0		
Arm set	3D print	0	0	0	0		
Gear set	3D print	0	0				
Arm holder	3D print	0					
	CNC						
Side plate	2mm	0	0	0	0	0	
	CNC						
Arm plate	2mm	0	0				
	CNC						
Extension horiz	4mm						
	CNC						
Extension vert long	4mm						
	CNC						
Extension vert short	4mm						
	CNC						
Body	4mm						

Table created by the author

01/14/24

Fab

- Milling bit was broken
- Restarted from where it skipped
- Baking sheet was damaged
- Looks like there was a slope. Not sure why though.

- Took a while to take out the damaged baking sheet

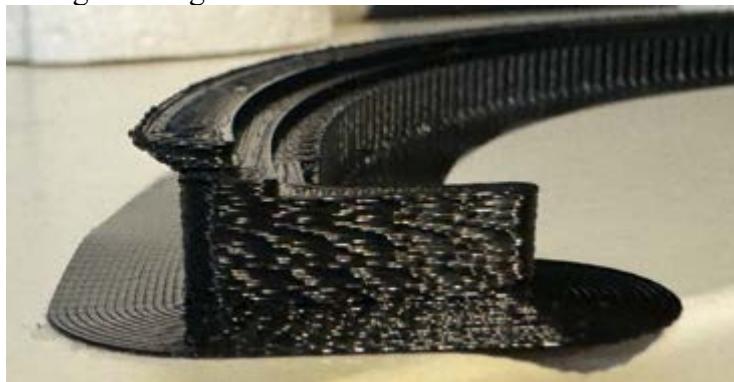


Photograph taken by the author

01/15/24

Fab

- 3d CNC milling went well
 - Finish 2mm fab
- Started 4mm arm extension board CNC milling
 - 4mm horizontal extension was milled good
 - Detaching was harder, as pieces larger
 - Slight under milling at extreme column, even with 0.7mm over milling
- 3d print
 - Extruder was shifted after certain step.
 - Whole prints except ring, are not useable.
 - Probably because initial height calibration is too aggressive
 - Dialing back 0.1mm height during Z calibration



Photographs taken by the author

01/18/24

Part fab status

		1	2	3	4	5	6
Motor mount	3D print	0	0	0	0	0	
Arm set	3D print	0	0	0	0	0	
Gear set	3D print	0	0	0	0	0	
Arm holder	3D print	0					

	CNC						
Side plate	2mm	0	0	0	0	0	0
	CNC						
Arm plate	2mm	0	0	0	0	0	0
	CNC						
Extension horiz	4mm	0	0	0	0		
	CNC						
Extension vert long	4mm	0	0	0	0	0	0
	CNC						
Extension vert short	4mm						
	CNC						
Body	4mm						

Table created by the author

01/19/24

Part fab status

		1	2	3	4	5	6
Motor mount	3D print	0	0	0	0	0	
Arm set	3D print	0	0	0	0	0	
Gear set	3D print	0	0	0	0	0	
Arm holder	3D print	0	0				
	CNC						
Side plate	2mm	0	0	0	0	0	0
	CNC						
Arm plate	2mm	0	0	0	0	0	0
	CNC						
Extension horiz	4mm	0	0	0	0		
	CNC						
Extension vert long	4mm	0	0	0	0	0	0
	CNC						
Extension vert short	4mm	0	0	0	0	0	0
	CNC						
Body	4mm						

Table created by the author

- CNC milling for arm extension failed as the plate was slipped

Dynamixel control and power wires

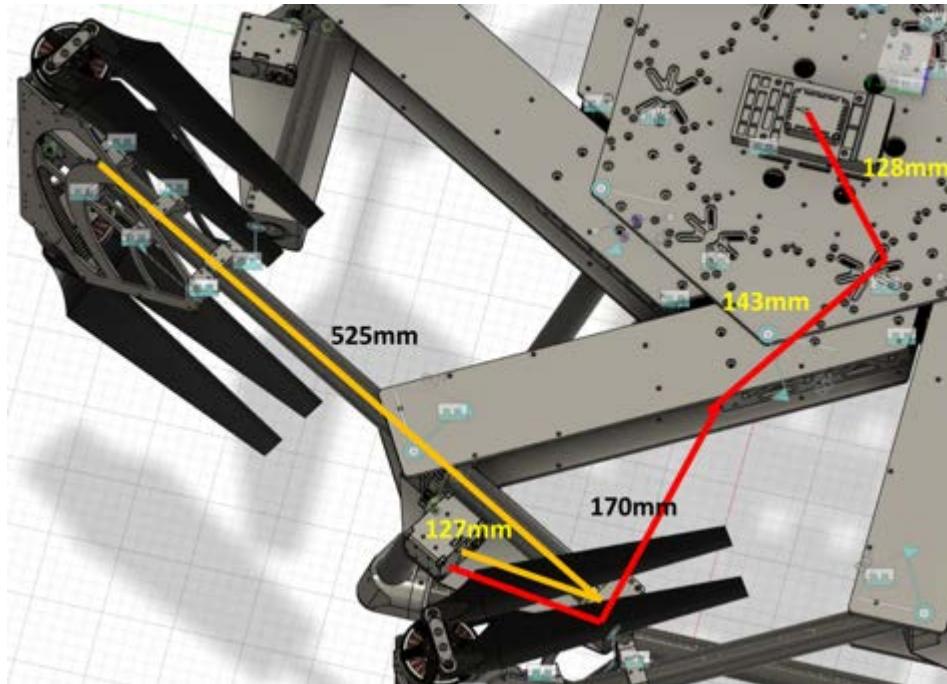


Image created by the author

- Pitch motor to roll motor: $525+127 = 652\text{mm} \rightarrow 700\text{mm}$
- Roll motor to controller: $127+170+143+128 = 568\text{mm} \rightarrow 600\text{mm} \rightarrow 700\text{mm}$
- Utilized provided connectors and cut to make them longer
 - Crimping takes time
 - Add 50mm, in addition to the given adapter length for margin
- Motor/ESC to PDB or Pixhawk: $86+163+390+170+143+128 = 1080\text{mm}$

Assembly order

- Connectors are quite big to go through a small hole
- Need to connectorize to be modular
- Soldering after wires are routed through holes in the tube

01/23/24

Leg holder 3d print

- Measured diameter is 21.8mm - too tight
 - Design diameter was 22.8mm
 - Next diameter should be 23.8mm
 - Need to expand foot shoe ink to 24mm
- Body frame plate CNC
- First plate made it after many tries
 - Second body plate is in CNC
 - Many 1.8mm mill bits are consumed

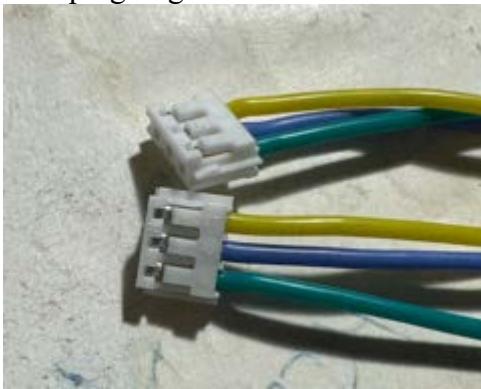


Photographs

taken by the author

Soldering vs crimping

- Crimping might work better and easier



Photograph taken by the author

01/24/24

3d printing

- Leg and shoe parts printed good. 2nd sec complete. Last set print started CNC milling
- Mill broke during outline
- Added XT60 connector hole, then restart milling



Photographs taken by the author

Assembly

Motor to ESC connection

- Straight connection rotates clockwise



Photograph taken by the author

Motor orientation for PX4 dodecacopter

https://bkueng.gitbooks.io/px4-devguide/content/en/airframes/airframe_reference.html

Dodecarotor cox

Common Outputs	
Name	
Generic Dodecarotor cox geometry	Maintainer: William Peale SYS_AUTOSTART = 24001

Image taken from

Dronecode (https://bkueng.gitbooks.io/px4-devguide/content/en/airframes/airframe_reference.html)

01/27/24

Motor mount needs a spacer to avoid friction with motor core



Photograph taken by the author

01/28/24

APD PDB500

- <https://powerdrives.net/pdb500>
- <https://docs.powerdrives.net/products/pdb/datasheets>

- <https://1471056808-files.gitbook.io/~files/v0/b/gitbook-x-prod.appspot.com/o/spaces%2FLqA1iwillddpVBXp8FT%2Fuploads%2FE3TA8lMde47Br4vnNQXH%2FPDB500v1.2.pdf?alt=media&token=e488e50a-0978-4ea9-b398-bafa9dceaa97>
- https://www.flyingtech.co.uk/sites/default/files/product_files/APD-F_Series-Manual.pdf
- <https://docs.powerdrives.net/downloads/3d-model-downloads>
- <https://docs.powerdrives.net/products/pdb>
Brushless motor wire length
 - 12 * 3 * 3.5 feet
 - 24V circuit (6S) will work with 16AWG with 30A and 5% voltage drop
 - https://www.viaircorp.com/v/wp-content/uploads/2017/01/VIAIR_wiregauge_chart.pdf
- MN5008 maximum current is 33A
ESC arrangement
 - De-solder ESC from motor
 - De-solder XT60 adapter from ESC power
 - Need to motor ESC cable first
 - Otherwise solder motor ESC cable after PDB assembly will be tough
- Then solder ESCs to PDB
Servo motor ID assignment
- Arm #3 pitch gear is loose. Fix axis might have damaged.
 - Need to re-print fix axis with extended coverage
- Arm #5's both pitch and roll motors are not responsive. Need to replace.
 - Cable wire order was wrong. Pitch motor was popped. Roll motor might have been damaged too.
- Arm #5's roll servo mount was not clean. It is working, but the servo gear might not be in the best shape

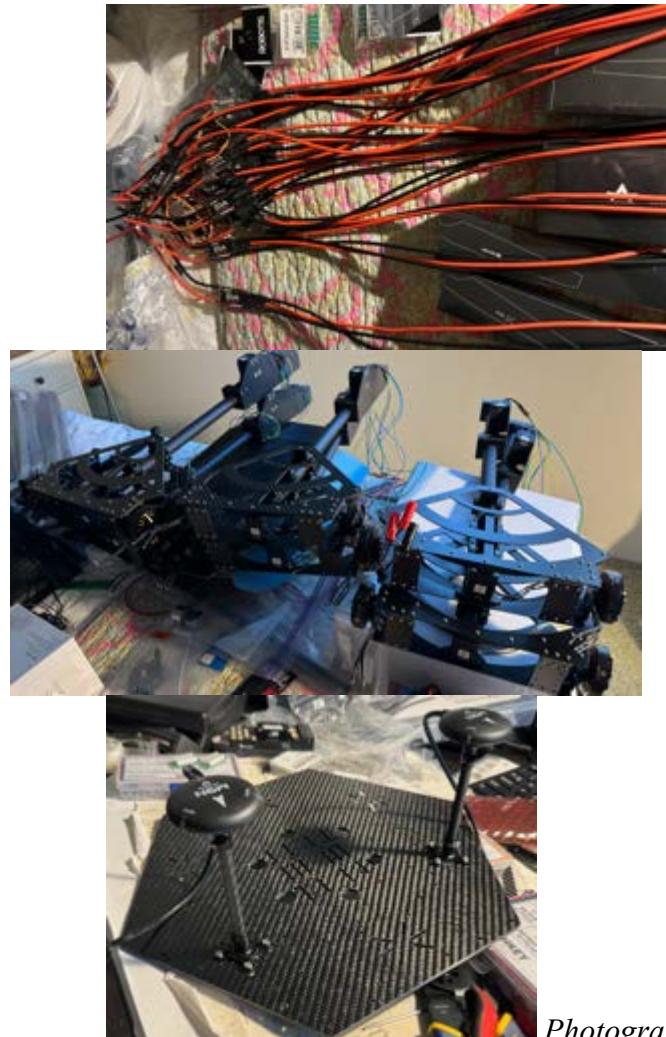
01/29/24

Assembly and preparation

- Solder ESC motor cable and connectors
- 6 arms are ready for arm extension
- GPS mount on top plate

01/30/24

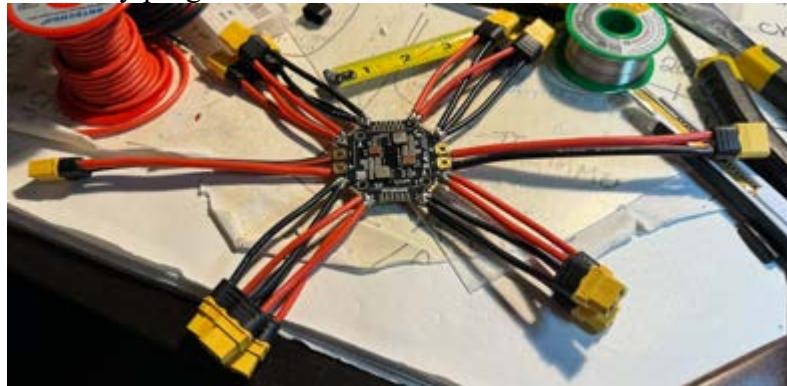
Assembly progress



Photographs taken by the author

01/31/24

Assembly progress



Photograph taken by the author

author

02/02/24

JETSON Orin reset

- Testing Dynamixel program
- It does not boot and stop at NVIDIA logo
 - After change dialout group for ttyUSB0
- Re-imaged JETPack 5.1.2
- Install VScode
- Headless mode
 - https://github.com/overclock98/Jetson_Nano_true_Headless_setup_without_hdmi_display
 - Need to disable encryption for VNC

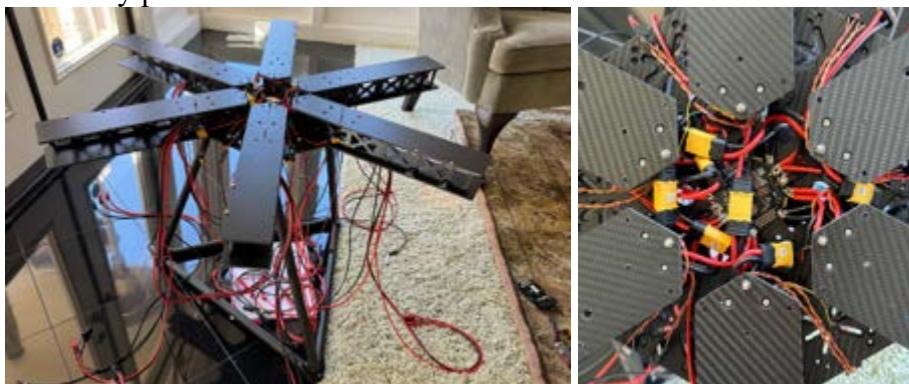
Assembly progress



Photographs taken by the author

02/03/24

Assembly process



Photographs taken by the author

Photographs



Photographs taken by the author

Test flight #1



Photographs

taken by the author

Leg was torn and a motor mount was broken



Photograph taken by the author

02/04/24

Test flight #2



Photograph

taken by the designated supervisor

Another propeller mount was torn:



Photographs taken

by the author

Test flight #3 – first official take off up to 5 inches.



Photograph taken by the author

But 4S battery failed to support thrust can crashed. A pitch block was broken.



Photograph taken by the author

02/05/24

Recovered from damages



Photograph

taken by the author

02/06/24

Tilt rotor control succeeded through Python



Photograph

taken by the designated supervisor

02/07/24

Flight test #4. Installed 6S LiPo batteries
Successfully took off



Photographs taken

by the author