2012

An extended duration hybrid powered hexcopter

Jason D. Winningham

AN EXTENDED DURATION HYBRID POWERED HEXCOPTER

by

JASON D. WINNINGHAM

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in
The Department of Electrical and Computer Engineering
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2012

In presenting this thesis in partial fulfillment of the requirements for a master's degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Jason D. Winningham (date)

THESIS APPROVAL FORM

Submitted by Jason D. Winningham in partial fulfillment of the requirements for the degree of Master of Science in Engineering in Computer Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate of the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Engineering in Computer Engineering.

Dr. David J. Coe	(Date)	Committee Chair
Dr. Jeffrey H. Kulick	(Date)	
Dr. Jennifer M. English	(Date)	
Dr. Robert Lindquist	(Date)	Department Chair
Dr. Shankar Mahalingam	(Date)	College Dean
Dr. Rhonda K. Gaede	(Date)	Graduate Dean

ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree Masters of Science in Engineering College/Dept. Engineering/Electrical and Computer Engineering

Name of Candidate Jason D. Winningham

Title An Extended Duration Hybrid Powered Hexcopter

This work attempts to extend the flight time of a UAS to allow for longer ranges, extended time on station, and larger payloads by adding lift capacity for an auxiliary power unit that generates electrical power using an internal combustion engine. The UAS is instrumented to examine power consumption during flights with various payload masses. Safety aspects of the larger UAS are examined, including a safety analysis of the flight control software.

Abstract Approval: Committee Chair
Dr. David J. Coe

Department Chair
Dr. Robert Lindquist

Graduate Dean
Dr. Rhonda K. Gaede

ACKNOWLEDGMENTS

I would like to thank Dr. David Coe for his guidance and support. Without it, I would not have been able to turn a flying weedeater into a thesis.

Thanks to Dr. Jeff Kulick for academic support, funding, and safety awareness. Yep, still have 10 fingers!

And especially thanks to Amy and Lauren, for putting up with me while I was either in the lab, in the shop, or ranting about a hexcopter.

-Jason

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE PAGE

LIST OF TABLES

TABLE PAGE

LIST OF SYMBOLS

SYMBOL	DEFINITION
$\eta_{control}$	control overhead factor(> 1)
η_{conv}	motor overhead factor (> 1)
ρ	density of air
V	voltage
I	current
R	resistance
$R_{DS_{ON}}$	MOSFET equivalent resistance between drain and source while on
P	power
T	thrust
C_T	thrust coefficient
C_P	power coefficient
FM	figure of merit

CHAPTER 1

INTRODUCTION

Many organizations have an interest in unmanned aerial systems (UAS). UAS have applications for the military, emergency services, law enforcement, environmental monitoring and research, agriculture, and infrastructure inspection. [1] [2] [3] A quadrotor helicopter (quadrotor) is one type of UAS. A typical quadrotor is a small rotary wing aircraft with four electric motors that requires no cyclic or collective pitch. [4] An example is shown in Figure 1.1.

A quadrotor differs from a traditional helicopter in that the propeller (prop) has a fixed pitch. Aircraft control is accomplished by varying the speed of one or more props, resulting in a corresponding change in thrust. [2] A helicopter prop must have the ability to vary pitch collectively, that is change the pitch applied to all blades equally, as well as a cyclic control mechanism, so that each blade's pitch is varied depending on its angle relative to the aircraft in order to achieve control and dynamic stability. [5] Quadrotor control is much simpler, mechanically, than a traditional helicopter swashplate, which has many moving parts to provide the cyclic and collective pitch control required for flight. This simplicity, illustrated in

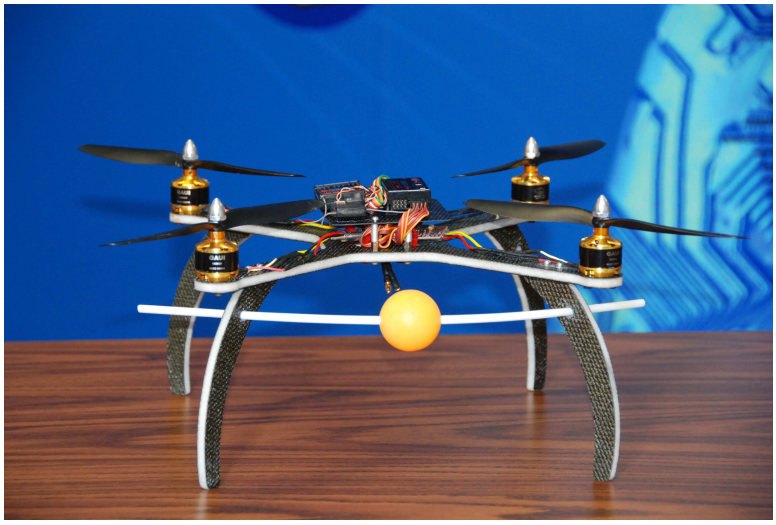


Figure 1.1: A modified Gaui 330X quadrotor

Figure 1.2, results in a more robust and easier to maintain drive system. In contrast, the swashplate for a model helicopter is shown in Figure 1.3.

Quadrotors provide advantages over fixed wing UAS. They can hover, are more maneuverable, and can fly more slowly than a fixed wing design. [2] Their drive train complexity is comparable to a fixed wing; the drive train on a single quadrotor arm is essentially identical to a fixed wing electric drive train mounted in a vertical orientation.

The cost of this maneuverability is that all of the quadrotor's lift comes directly from thrust, with a corresponding increase in power consumption. Another quadrotor disadvantage is control complexity. A quadrotor is an under-actuated system, with four actuators and six degrees of freedom [6], which results in a more complex control problem than a fixed wing aircraft.

Figure 1.2: A quadrotor prop and motor.

These advantages have made quadcopters popular research platforms. Extensive work is being done in the area of quadrotor control [7] [8] [9] [10] [6] [1] [11]. Additionally, work is being done in the areas of modeling [12] [6] [7], simulation [8] navigation [10], and trajectory control [11]. Multiple quadrotors can cooperate to perform a single task. [13]

Some quadrotor researchers are utilizing variable pitch props to achieve more agility than their fixed pitch counterparts [15] [16], even allowing inverted flight [17]. This additional control results in a slightly higher mechanical complexity and cost. The variable pitch prop is still much simpler than a standard helicopter design, since no cyclic control is required.

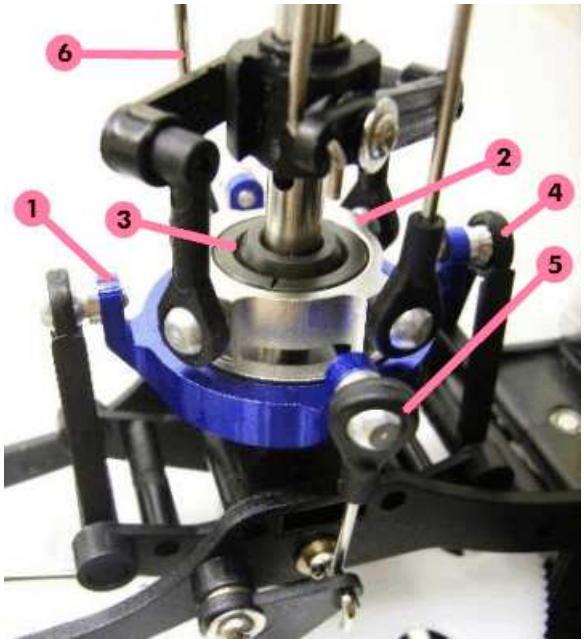


Figure 1.3: An RC helicopter swashplate [14]

The quadrotor definition can be extended to multicopters. Designs with 3, 4, 6, 8, and 10 props are available [18]. Most configurations have all props arranged symmetrically in a single plane. Some use a coaxial configuration, with 2 motors on top and bottom of each mounting structure.

Endurance is another disadvantage of multicopters. [2] Existing multicopters have short flight times even when carrying no payload. Extended flight times would allow a broader scope of missions since the aircraft can fly farther and stay on station longer. The goals for this project are to design a heavy lift multicopter capable of lifting a 5kg payload; to extend the duration of the multicopter by designing a hybrid auxiliary power unit driven by an internal combustion engine; and to perform a safety analysis of the flight computer software, since the ramifications of a control failure on a larger, longer-ranged aircraft are greater than that of a smaller short range aircraft.

CHAPTER 2

BACKGROUND

The goal for this project is to build a 6 rotor multicopter (hexcopter) capable of flying for more than one hour. Existing multicopters have typical flight times in the 15 to 20 minute range with no payload on the aircraft. Extended flight time allows the aircraft to operate over a larger area, fly longer distances, and gather more data. Table 2.1 shows example flight times and payloads for commercial UAS. Flight times drop dramatically when payloads are flown.

Lithium-polymer (li-po) rechargeable batteries are the typical power source for UAS. The basic cell is nominally 3.7V with a capacity relative to the physical size of the cell. Cells vary in size. Multiple cells are placed in series to attain higher voltages. Batteries are frequently referred to by the number of cells, not their voltage; e.g. a

Table 2.1: Commercial UAS payload and endurance, vendor specifications.

UAS	payload	flight time
Gaui 330X	350g	7.7 min
Gaui 500X	350g	7.1 min
Cinestar 6	167g	24 min
Vario Quad Copter	4000g	8 min

Table 2.2: Energy density for UAS power sources.

source	efficiency (%)	energy density $W \cdot h/kg$
Li-polymer battery [19]	95	333
methanol fuel cell [19]	35	1000
nitromethane fuel [19]	6	6061
gasoline [20]		12,333

3S battery contains three 3.7V cells in series, for a nominal 11.1V battery. Compared to other battery chemistries, li-pos are lightweight, capable of storing more energy, and able to discharge that energy at a higher rate without damage to the cells.

Li-po batteries have shortcomings as well. If the case breached or if the battery is overheated, overcharged, or charged too rapidly, the result may be a fire or explosion. While they are lightweight compared to other batteries, their energy density (defined as the energy per unit mass) does not compare favorably with other vehicle and aircraft power sources. Table 2.2 shows the energy density of various fuels used for small UAS.

2.1 Aircraft Requirements

The main requirement for the UAS design is to be capable of utilizing an Auxiliary Power Unit (APU) that produces electrical power sufficient to fly the aircraft with payload. The APU will consist of a hydrocarbon fueled internal combustion engine, generator, and appropriate power regulation. It will be used in conjunction

with a Power Management Unit (PMU) that will control the APU and integrate APU output with the aircraft's power system.

The aircraft will have batteries to provide power when the APU may not be able to supply adequate power, for example when maneuvering aggressively, and for emergency situations, so that a safe landing can be performed in case of loss of APU power generation. It is desirable to have excess generating capacity so that the PMU can also use the APU to power the payload and recharge batteries.

The flight computer must be capable of operating a 6 rotor multicopter (hexcopter). It must be an open source project so that the source code can be analyzed with tools commonly used for software safety and reliability testing. Autopilot functionality is highly desirable to ease the pilot's workload during extended flights.

2.2 Safety

Safety is a critical consideration in the design of the system. A typical multicopter design employs lithium-polymer batteries which contain significant electrical and chemical energy. The APU will be carrying flammable fuel. In addition, the mass (and corresponding kinetic energy) of the aircraft presents a hazard to personnel and property in the event of loss of control or power during flight.

"Software safety is a systems issue, not a software-specific issue." [21] The software used to control the aircraft is critical to the overall safety of the aircraft. The software will be analyzed to identify issues that may impact reliability and flight safety. Industry standard tools and coding standards will be used for the analysis.

2.3 Regulation

The Federal Aviation Administration (FAA) regulates airspace in the United States. Current FAA regulations prohibit public agencies from operating "Unmanned Aircraft Systems" without a "certificate of waiver" or "certificate of authorization" (CoA) for specific flight operations. These certificates are quite specific: the aircraft, pilot, area of operation, time of day, and other such operational details are specified. Congress has instructed the FAA [22] to update its regulations to integrate UAS into the National Air Space (NAS) and create a more expedient process for obtaining a CoA for UAS operation. Congress has also instructed the FAA to maintain the current hobbyist exemption. As a public institution, research conducted at UAHuntsville does not qualify for this hobbyist exemption.

2.4 Preliminary Work

Preliminary work was done with a commercial Gaui 330X quadrotor. The COTS unit was used for initial flight training. A composite frame was built to replace the factory frame. The Gaui was instrumented with current and voltage sensors and a telemetry system. The stock flight computer was also replaced with an ArduPilot Mega.

CHAPTER 3

UAS DESIGN

Like many engineering problems, the design process is iterative in nature. An abbreviated version of Leishman's [5] rotorcraft design cycle is shown in Figure 3.1. The extended flight duration of a multicopter is the primary requirement, and the APU is the proposed solution to the problem. With these inputs we create the initial aircraft and generator designs, verify by simulation that the aircraft can lift the generator and in turn consumes less power than the generator provides, and modify the aircraft and APU designs until the system meets requirements.

3.1 Theoretical Calculations

A momentum theory analysis [5] shows that the power supplied by a rotor is

$$P_{ideal} = \sqrt{\frac{T^3}{2\rho A}} \tag{3.1}$$

where P is the power required, T is thrust, ρ is the density of air, and A is the area of the rotor disk. Based on specifications of existing examples, it is estimated that six 14 inch rotors will provide sufficient lift for a gross aircraft weight of 10 kg, or

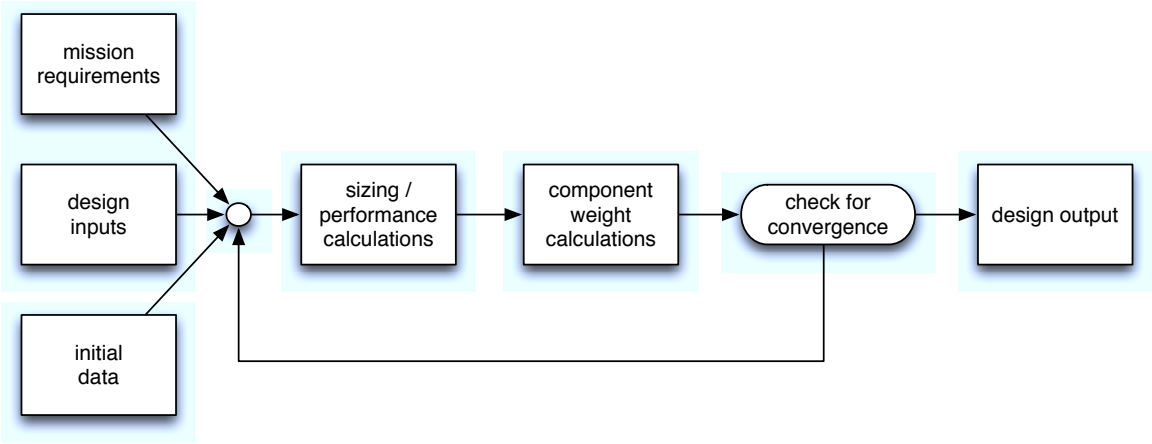


Figure 3.1: Design cycle for a generic rotorcraft [5].

1.67 kg per rotor. 1.67 kg of mass requires $1.67kg \cdot 9.8\frac{m}{s^2} = 16.3N$ of thrust per rotor in order for the aircraft to hover. ρ, the density of air, is approximately $1.225kg/m^3$ at $15°C$ at sea level. Substituting these values in Equation 3.1:

$$\begin{aligned} P_{ideal} &= \sqrt{\frac{(16.3kg \cdot \frac{m}{s^2})^3}{2 \cdot 1.225\frac{kg}{m^3} \cdot 0.099m^2}} \\ &= 133.6W \end{aligned}$$

So the minimum theoretical power required for a 10 kg hexcopter with 14 in rotors to hover is $6 \cdot 133.6W = 801.6W$.

Momentum theory gives power requirements of an ideal system. The assumption that the rotor is perfectly efficient is a source of error. The figure of merit (FM) of a rotor [5] is defined as

$$FM = \frac{P_{ideal}}{P_{measured}} \tag{3.2}$$

An estimated FM for a model airplane prop used on a multicopter is 0.5 [23].

The hexcopter will need additional power. The momentum theory calculation gives the power required to hover, but for the aircraft to ascend, fly laterally, and perform other maneuvers a control efficiency overhead is required. This overhead is represented by $\eta_{control}$.

There are also power losses for the electrical to mechanical power conversion. The motor does not perfectly convert all electrical power to mechanical power. The electronic speed controller (ESC) typically uses MOSFET transistors to switch power, which incur a loss represented by an equivalent resistance between the drain and source $R_{DS_{ON}}$. This switching loss and the motor conversion loss are represented by η_{conv}. These losses are combined into a single equation:

$$P_{total} = P_{ideal} \cdot \frac{1}{FM} \cdot \eta_{control} \cdot \eta_{conv} \tag{3.3}$$

Using the momentum theory calculation of 802W from Equation 3.2, an FM of 0.5, an estimated power overhead $\eta_{control}$ of 20% [24] and an estimated motor efficiency η_{conv} of 90%, [24],

$$\begin{aligned} P_{total} &= P_{ideal} \cdot \frac{1}{FM} \cdot \eta_{control} \cdot \eta_{conv} \\ &= 802W \cdot 2 \cdot 1.2 \cdot 1.1 \\ &\approx 2100W \end{aligned}$$

it is estimated that a minimum of 2.1kW of electrical power will be required.

Table 3.1: eCalc power estimate summary for 10kg GVW

	at hover	max
current per motor	29 A	51 A
current (total)	175 A	308 A
power per motor	325 W	548 W
power (total)	2054 W	3621 W

3.2 Simulation

Since the momentum theory calculation and power loss estimate indicate the available power and prop size are theoretically adequate, we proceed with an analysis using the commercial simulation tool eCalc [24]. eCalc takes into account more detailed motor specifications, prop properties, and other system components such as battery capacity, battery discharge rate, and ESC capacity. The eCalc simulation output for the 10kg gross vehicle weight (GVW) is shown in Figure 3.2. Power consumption data from the 10kg simulation is summarized in Table 3.1.

In addition to the 10kg GVW simulation, the aircraft with no APU will weigh 5kg. Examining the aircraft performance with no APU payload is useful so that the performance of the APU equipped aircraft can be compared with alternative configurations. The simulation results for the 5kg GVW are presented in Figure 3.3. Appendix B provides additional simulation outputs for 11kg and 12kg GVW configurations in Figure B.1 and Figure B.2, respectively.

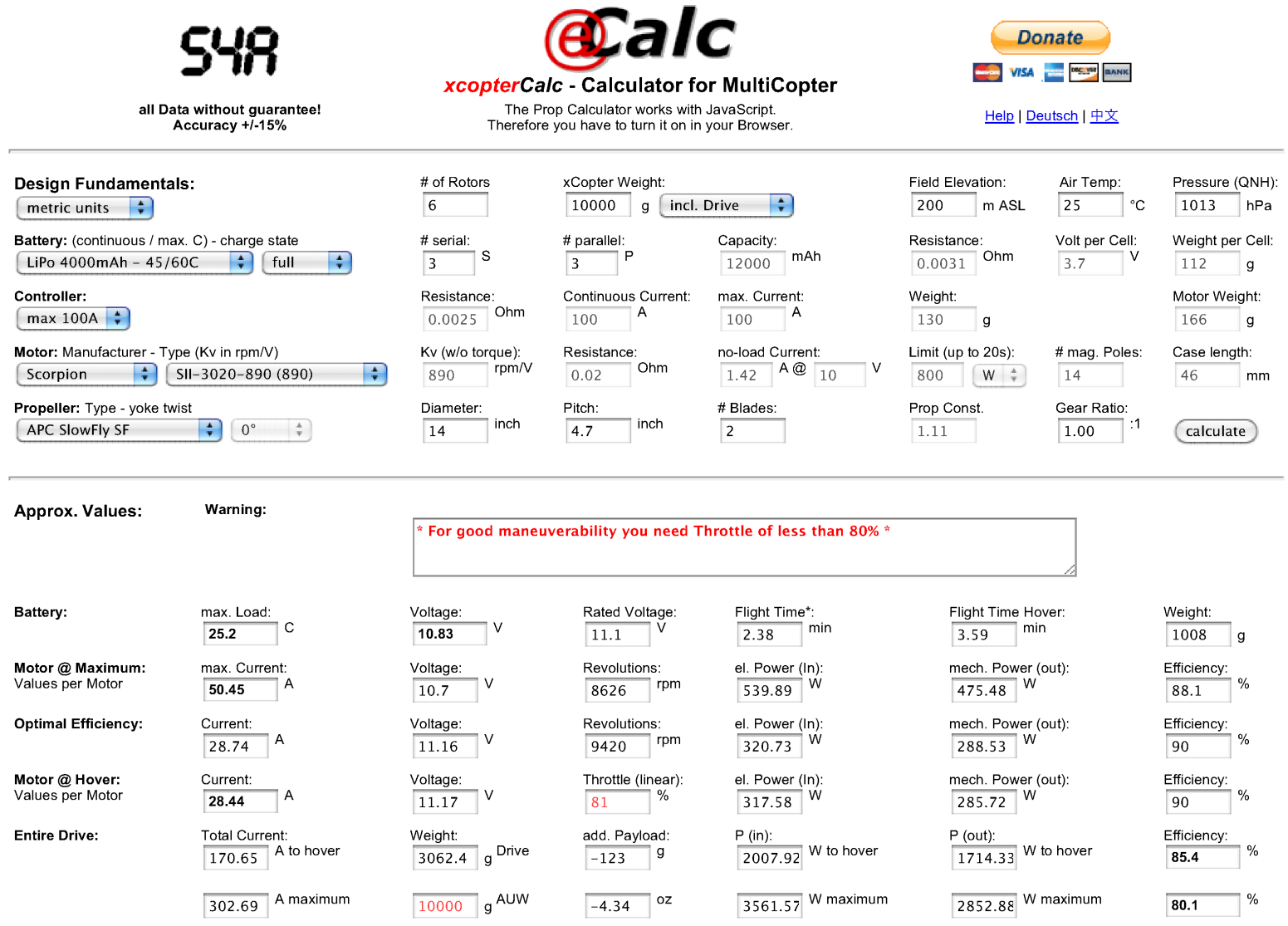


Figure 3.2: eCalc simulation output, 10kg GVW.

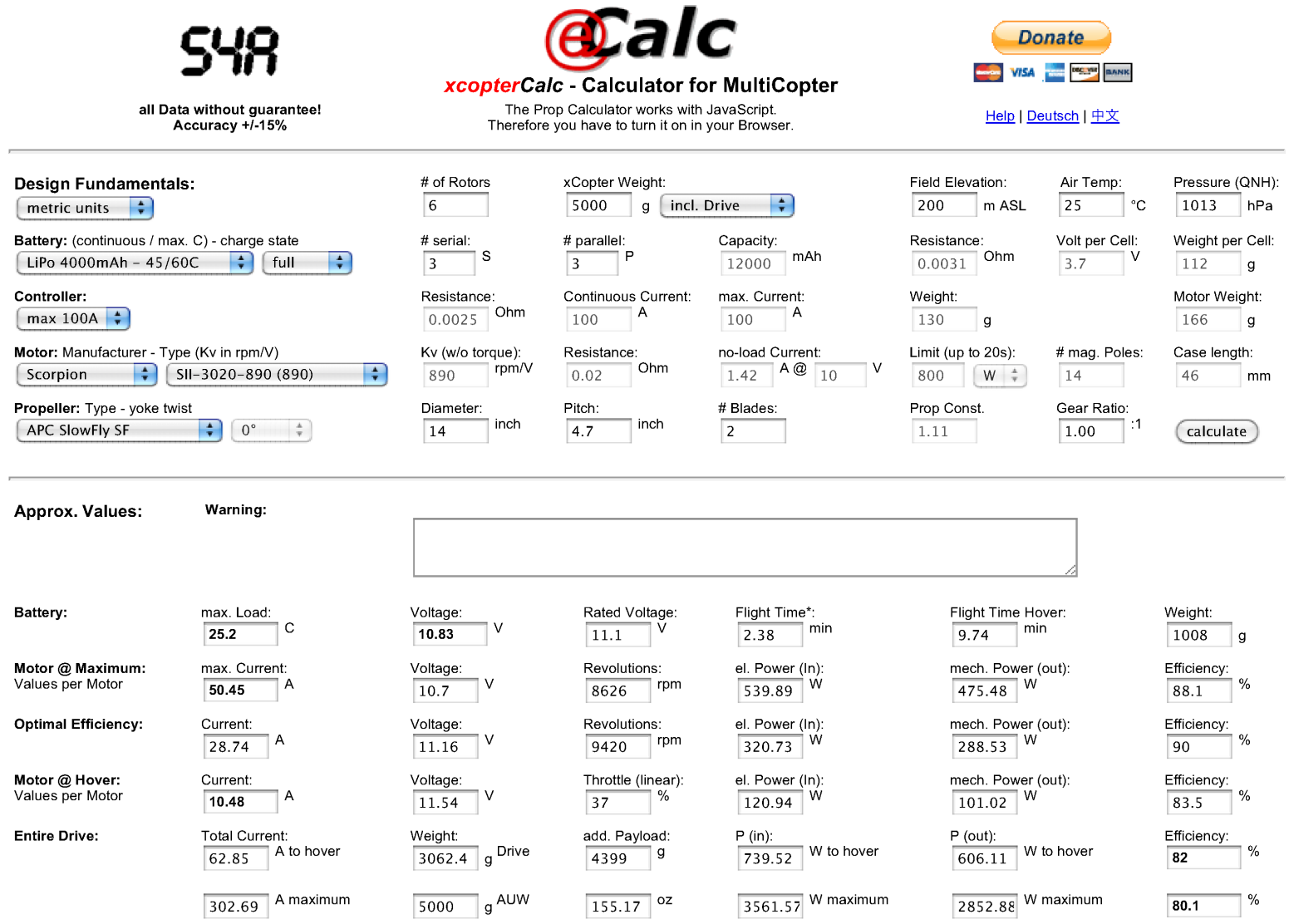


Figure 3.3: eCalc simulation output, 5kg GVW.

Table 3.2: Drive Train Components

APC 14x4.7SF and SFP props
Scorpion SII-3026-890 motor
Hobbyking SS 90A speed controller
Turnigy 4000mAh 3S (11.1V) 40C Lipo battery

3.3 Drive Train Component Selection

Components for the aircraft drive train are chosen based on their ability to meet specified requirements, cost and availability. ESC current capacity is specified with a safety factor of approximately 100%, since our extended duration flights are much longer than the use for which hobby ESCs are designed. The drive train components chosen are listed in Table 3.2 and shown in Figure 3.4.

Using an approach described in [23], data for several commercially available props has been compiled by Ananda [25]. While the exact prop chosen is not represented, thrust coefficient (C_T) and power coefficient (C_P) data for similar sizes of the same design is available. An alternate formula for the figure of merit is [5]

$$FM = \frac{{C_T}^{\frac{3}{2}}}{\sqrt{2}C_P} \tag{3.4}$$

Using this formula and data from Ananda [25] gives an FM of 0.57 for a slightly smaller prop in the same APC SlowFlyer family.

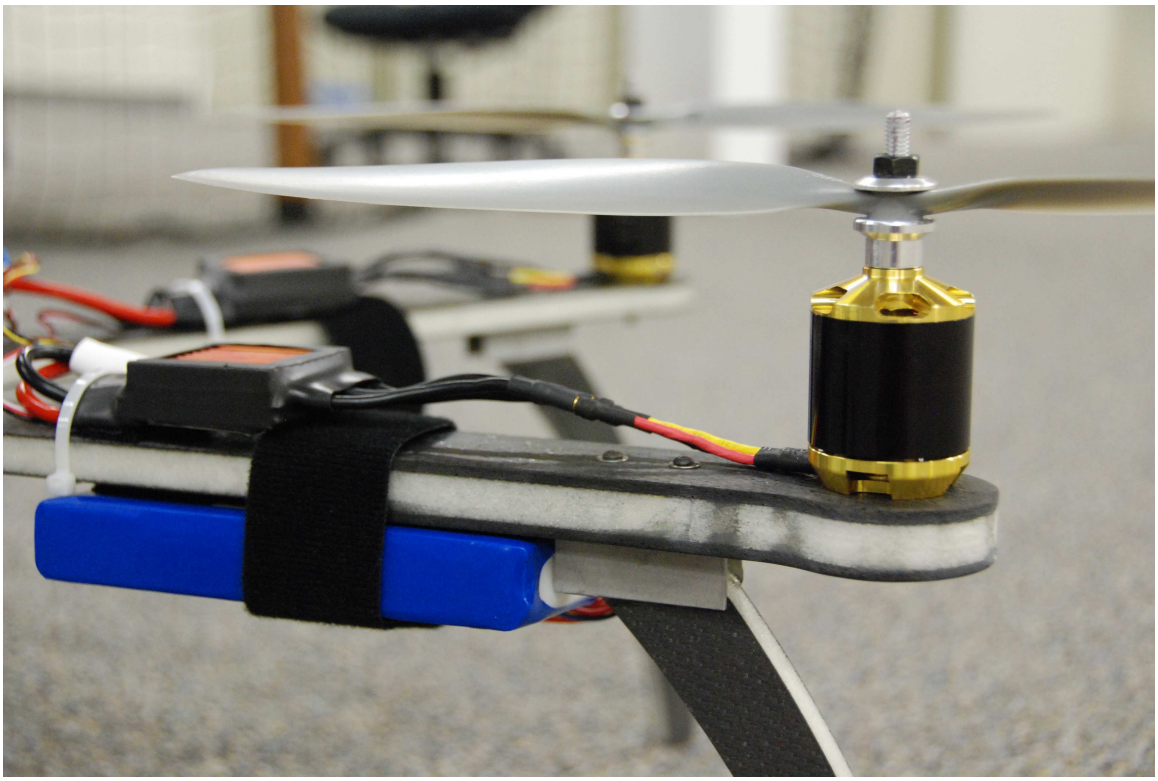


Figure 3.4: Drive train components shown mounted on the completed airframe.

The aircraft mass budget is shown in Table 3.3. The estimated values on the left were gathered from vendor data. If no vendor data was available, values were estimated based on data for similar components.

3.4 Avionics

The ArduPilot Mega (APM) is a commercial off the shelf (COTS) flight computer chosen. This device uses open source software, which facilitates the safety analysis of the flight computer code. The APM also has an autonomous mode, which may assist with extended flight test sessions.

Table 3.3: Aircraft Mass Budget

	est. unit mass (g)	qty	total mass (g)	actual unit mass (g)	qty	total mass (g)
frame	627	1	627	750	1	750
landing gear	23	6	138	36	6	216
motor	182	6	1092	221	6	1326
props	25	6	150	25	6	150
ESC	50	6	300	81	6	486
battery	175	6	1050	375	3	1125
avionics	250	1	250	250	1	250
total			3607			4503

The RC receiver, the primary flight control link, is a Spektrum A8000. Radios for APM telemetry and experimental telemetry are Digi 9XTend [26] 900MHz spread spectrum radios. They are chosen for their ability to operate on multiple channels without mutual interference and their range, which is much greater than the typical unlicensed ISM (Industrial, Scientific, and Medical) radio systems. Integration of the avionics with other aircraft systems is shown in Figure 3.5.

3.5 Power Consumption Data Acquisition Payload

The UAS is instrumented with a power monitoring and telemetry payload. This payload consists of current and voltage sensors to determine actual in-flight power consumption characteristics. These measurements are telemetered back to a ground station for storage.

The UAS power distribution system is designed to allow multiple battery configurations. In order to allow accurate power measurement while maintaining flexibil-

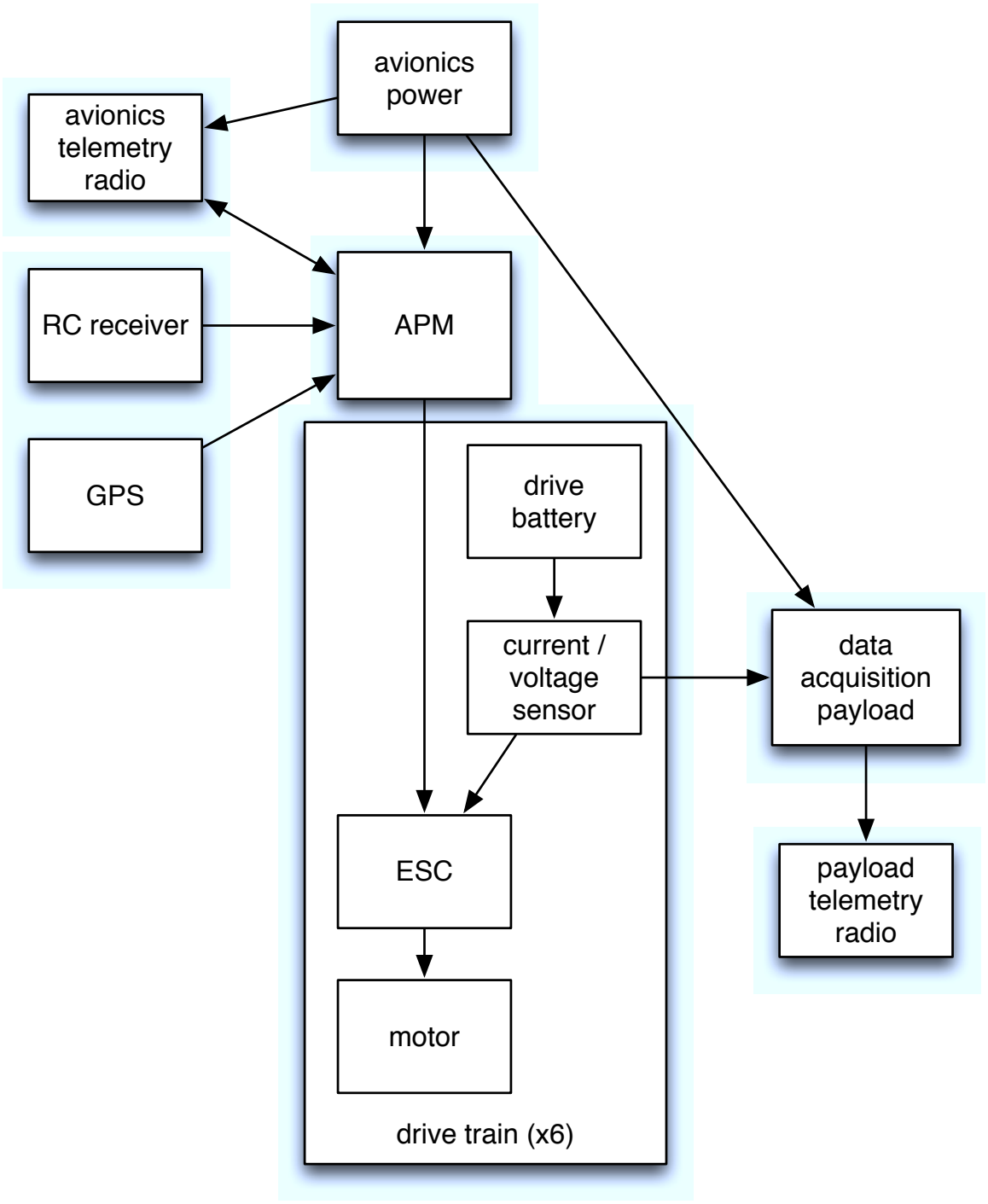


Figure 3.5: System block diagram. Note drive train components are replicated 6 times on this aircraft.

ity, each ESC power input is fitted with a current and voltage sensor. These analog sensors are the core of the data acquisition and telemetry system.

Each ESC power input is connected to a current and voltage sensor based on the Texas Instruments INA-169 [27] high side current shunt monitor. This is a current shunt amplifier that measures the voltage drop across a small value shunt resistor in series with the positive voltage supply. The components chosen allow current measurements up to 90A. In addition, the module also provides a resistor divider network to scale the input voltage down to a range appropriate for a 3.3V ADC input. A similar 45A version of this circuit is used to instrument the avionics and data acquisition subsystems power consumption.

The current sense and voltage sense outputs of the power sensors are connected to the analog inputs of an MCP3208 [28]. It is a 12 bit successive approximation analog to digital converter (ADC) that can sample at 100k samples/s. It provides a higher resolution than the ATmega's internal 10 bit ADC. Two MCP3208s provide a total of 16 analog inputs, 14 of which are in use. They are connected to the host microcontroller via the SPI (Serial Peripheral Interface) bus.

Multiple 9XTend radio networks can operate concurrently on different channels without interference, so the 9XTend was chosen to provide both the avionics telemetry and the payload telemetry. The data acquisition payload and ground station radios are configured to operate on channel 2. The radio is mounted in close proximity to the payload, so 5V TTL levels were used and no external RS232 level converter was required.

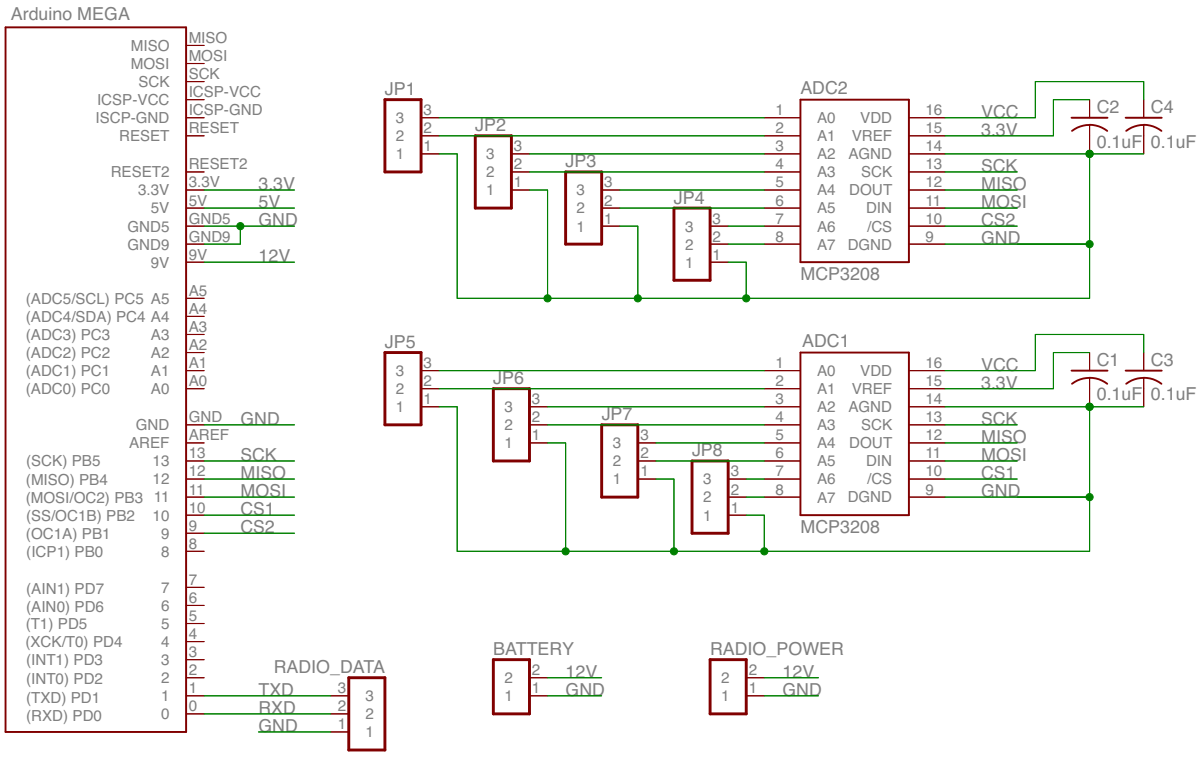


Figure 3.6: Prototype power data acquisition telemetry schematic.

The microcontroller for the telemetry system is the ATmega2560 [29]. It is an 8 bit AVR architecture that meets the data acquisition system requirements of an SPI interface to attach the ADCs and a UART (Universal Asynchronous Receiver-Transmitter) to interface with the radio. The hardware form factor is an Arduino MEGA microcontroller development board, chosen for its familiarity and the large supply of hardware and software components that are commercially available. Arduino daughterboards are called "shields". An Arduino MEGA prototyping shield, which consists of a large prototyping area and pads connecting to the MEGA pins,

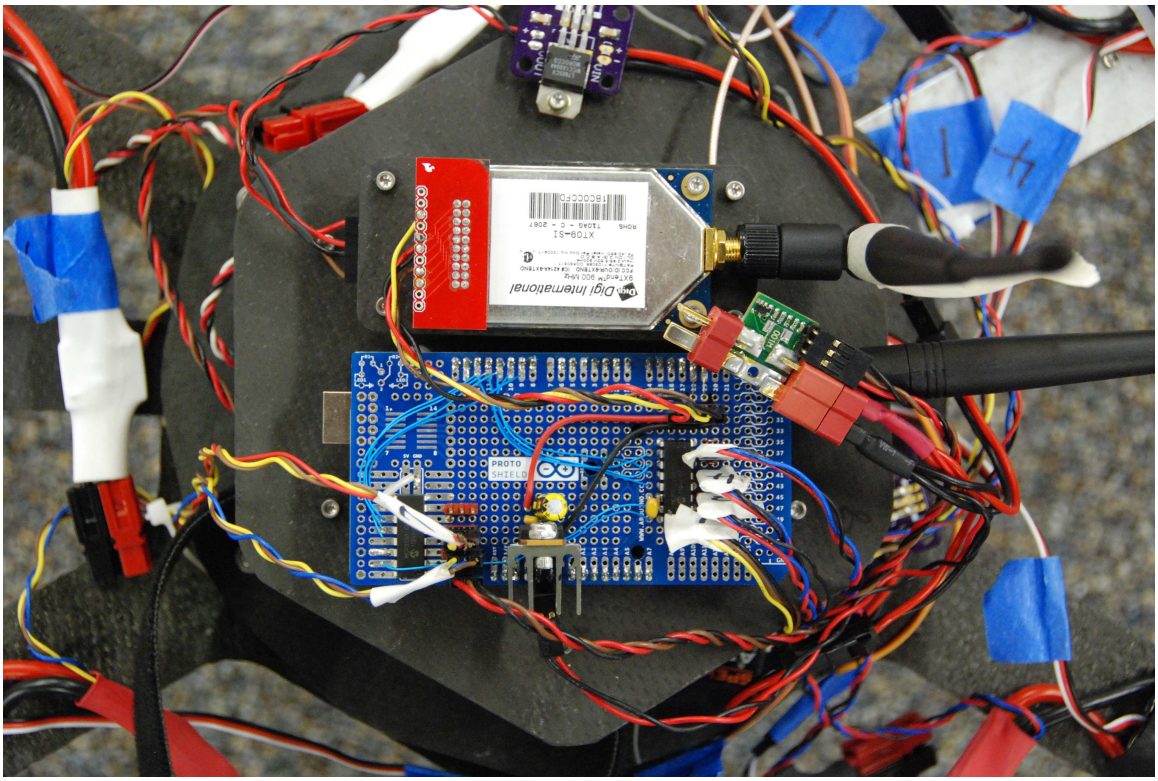


Figure 3.7: The power consumption data acquisition payload.

was used to mount the MCP3208s and pin headers for connections to the current sensors and radio. A voltage regulator circuit was also added to supply power to the radio, since the radio has no onboard regulator and the Arduino MEGA's onboard regulator does not have sufficient capacity to power the radio. The schematic for the shield components are in Figure 3.6. The completed payload is shown as flown in Figure 3.7.

The software was developed using the Arduino IDE version 22. The source code is included in Appendix D. Transfer functions for converting the ADC voltage measurements into current and voltage measurements were obtained from the Attopilot datasheet.

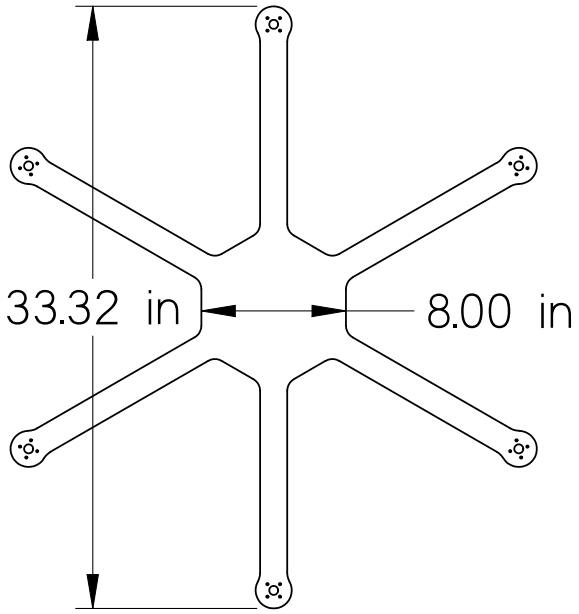


Figure 3.8: Airframe overall dimensions.

3.6 Airframe Design

The airframe is designed based on total mass requirements and the properties of the components specified. The composite frame is a unibody design constructed using carbon fiber and structural foam which results in a strong and lightweight frame. A modular frame design was considered, but the unibody design worked well on a smaller scale aircraft used for early experimentation. Minimum overall frame dimensions are driven by the prop length – the frame arms must be long enough so that adjacent props do not make contact. Figure 3.8 shows the major frame dimensions of the aircraft.

Landing gear design is driven by multiple factors. The frame must be held high enough so that any payload mounted below the body is not in contact with the ground. Additional height reduces ground effect issues, simplifying control during takeoff and landing. Excessive height reduces the stability of the aircraft when on the ground and increases mass and drag.

3.7 Airframe Construction

The first step in the construction process is to fabricate an oversized blank of the frame material. The frame consists of two layers of prepreg carbon fiber and one layer of film adhesive are applied to each side of a Rohacell structural foam core. The resulting layup is vacuum bagged with an aluminum plate and oven cured. The resulting blank is machined to final dimensions using a CNC milled template and a router table. This process is described in the following section.

3.7.1 Safety

Epoxy resins can cause allergic reactions by skin contact. Carbon fiber releases extremely fine particles when machined, ground, or sanded. Appropriate safety measures were followed in the use of material and tools. This includes use of safety glasses, breathing mask, gloves, and hearing protection as required. Safety instruction was received for all tools and machines used.

3.7.2 Mechanical Software Design Tools

Mechanical parts were designed using an academically licensed copy of the SolidEdge solid modeling package. Designs that were fabricated by CNC machining were processed using an academic license of SURFCAM. CNC machining was performed on either the Haas VF-1 or Haas TM-1 CNC mills located in UAHuntsville's Engineering Design & Prototyping Facility.

3.7.3 Foam Core Cutting and Assembly

The airframe construction utilizes a 0.4" thick Rohacell [30] structural foam core. Rohacell is a closed-cell rigid polymethacrylimide foam. It is machinable and thermoformable. The material is expensive, and if a single frame is cut from a solid sheet material waste is over 90%. In order to conserve material and reduce cost, the foam core was cut in segments and epoxied together to form an oversized blank for the layup. This reduced waste to less than 20%.

The foam was rough cut on the table saw into 6 rectangular blocks for the airframe arms, and a single square for the central body of the airframe. A tab was machined on the end of each arm to provide alignment with the center frame. The center frame was machined into a hexagon, with slots in each side to receive the arm tabs. Dimensions for the arm and center frame are shown in Figure 3.9 and Figure 3.10. This tab and slot arrangement provided alignment to ensure the parts did not slip during the assembly and epoxy cure process.

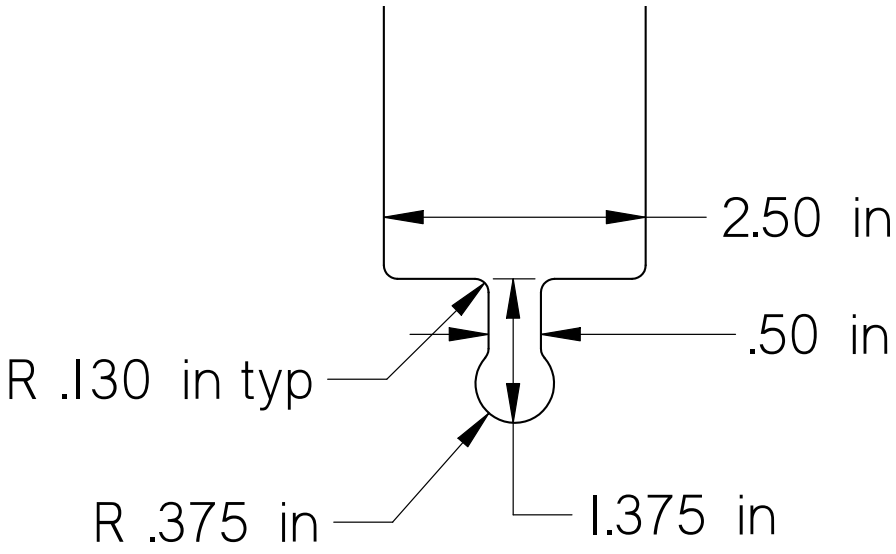


Figure 3.9: Foam arm detail.

The foam arms were attached to the center frame with 3M brand DP-460 epoxy. [31] DP-460 is a room temperature cure two part epoxy with a 60 minute work life and 24 hour cure time. The assembly was placed on a composite layup release film, covered with a layer of perforated release film so that excess epoxy could be removed, and placed in a vacuum bag with appropriate breather fabric. This assembly is shown in Figure 3.11. The vacuum bag process not only provides excellent clamping pressure, it allows excess epoxy to be removed from the assembly so that the resulting surface is flat. If excess epoxy was allowed to cure on the assembly, additional machining would be required to flatten the surface before continuing with the composite layup.

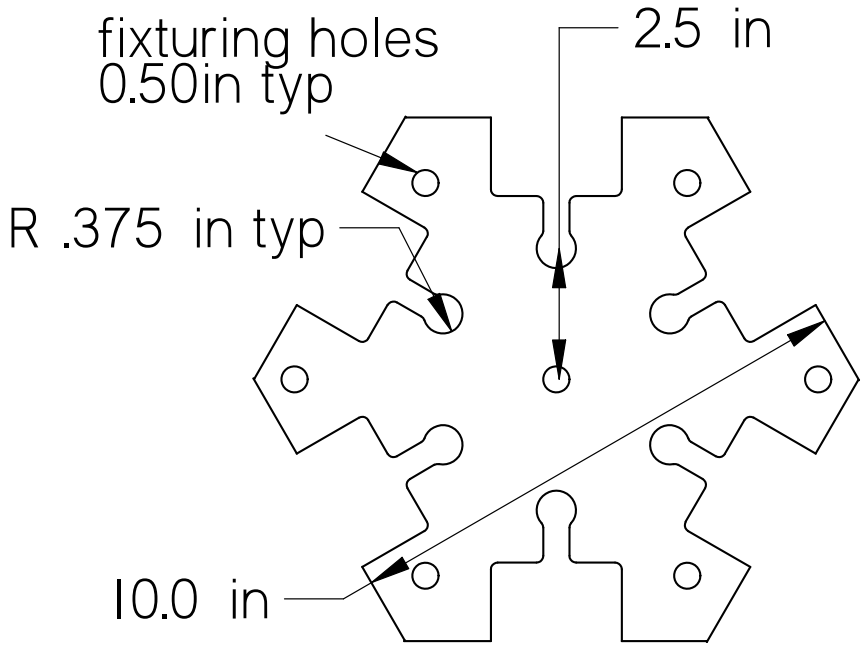


Figure 3.10: Foam body detail.

3.7.4 Static Test of Frame Strength

There was concern about the strength of the frame being reduced due to the joints in the foam core. Test samples of 0.215" and 0.7" Rohacell with a layer of film resin and 2 layers of carbon fiber prepreg were constructed. The 1" x 12" samples were made with foam cores that had either a butt joint or a half-lap joint bisecting the sample across the width in the middle of the sample.

Samples were tested by clamping one end of the sample to a rigid structure, then suspending a static load to the other end of the sample. Results are shown in

Figure 3.11: Rohacell foam core assembly, clamped for curing.

Table 3.4. The maximum load that could be applied with the test fixture was 55lb. The 0.7" samples did not fail with a 55lb load. Both the solid (control) and half-lap 0.215" samples failed at the point where the samples were clamped in the fixture. The joined sample did not fail at the joint. All 0.7" samples held the 55lb maximum load. Samples of the 0.4" material used for the aircraft frame were not available for static testing, but results of the 0.215" and 0.7" sample tests were considered adequate.

3.7.5 Composite Layup

Once the assembled foam core had cured, the composite layup could begin. The primary structural material for the airframe is LTM45EL/CF1803 [32] carbon

Table 3.4: Composite Layup Static Strength Test

sample	joint	max load	observation
0.215”	none	43lb	failed
0.215”	half-lap	48lb	failed (not at foam joint)
0.7”	half-lap	55lb	no failure
0.7”	butt	55lb	no failure

fiber prepreg matrix. “Prepreg” indicates that the fiber matrix is pre-impregnated with a metered amount of resin. Using this material is simpler than applying epoxy resin to a dry weave at layup time. The LTM45EL/CF1803 must be stored at $0°F$ and has a 5 to 6 day work life at $70°F$. The material should be allowed to come to room temperature before use.

Because the foam core is part of the layup, additional resin is needed. The amount of resin in the prepreg is sufficient for the carbon fiber matrix, but additional resin is needed to bond the foam to the rest of the layup. This additional resin is applied in the form of LTA45EL film adhesive. This material is a thin sheet of the same resin used in the LTM45EL/CF1803 carbon fiber.

The layup is assembled as follows. Slightly oversized pieces of film adhesive and carbon fiber are cut. One layer of film adhesive is applied to the foam core. Once the film adhesive is in place, a layer of carbon fiber is applied to the film, then a second layer of carbon fiber is applied over the first. The assembly is flipped over, and the same process is performed on the other side of the foam core so that the resulting composite sandwich consists of the following layers: carbon fiber, carbon fiber, film adhesive, Rohacell foam, film adhesive, carbon fiber and carbon fiber. Note that

both the film adhesive and the carbon fiber have a protective sheet on each side of the material. This sheet must be removed before applying the material to the layup. It is convenient to remove the protective backing from one side, apply the exposed face to the layup, and then remove the backing from the other side. It is also advisable to leave the top-most protective sheet until in place the release film is ready to be applied.

3.7.6 Vacuum Bagging

Once assembled, the layup is ready for the bagging process. The the layup is placed in a vacuum bag so that any air bubbles or voids are removed from the layup. Excess resin can also be removed by this process. For oven cured resins ensure that all release films, breather fabrics, bags, and other materials are rated for oven use.

First the layup is covered with a layer of non-perforated release film. The resin will not bond with this film. Failure to use a release film will result in epoxying the layup to anything else it touches during the cure process. Make sure any remaining protective backing is removed from the layup before applying the release film. The release film should be applied so that it is smooth, with no wrinkles or discontinuities that will telegraph into the surface of the finished layup.

Once the release film is applied the layup is placed on an aluminum plate. This provides a rigid structure to clamp the layup against so that the cured layup is flat. Any seams, dirt, or defects in the aluminum plate will telegraph into the surface of the cured layup.

Next a layer of breather fabric is placed on top of the layup. A section of breather should lead away from the layup so that a vacuum foot can be attached. The vacuum foot provides a port in the bag that allows the vacuum source to be connected. There should be enough breather fabric so that excess resin will be absorbed by the breather before it can reach the vacuum foot to avoid permanent damage to the foot.

Once the breather fabric is in place, a layer of vacuum bag material is used to cover the layup. Typically the entire stack, including the aluminum plate, is placed inside the bag. The bag material is available in rolls. Clay tape specifically designed for the vacuum bagging process is used to seal edges of the bag.

A vacuum of approximately 25in Hg is applied. Care should be taken to ensure the layup has not shifted during the bagging process, so that all composite layers are still aligned properly, the release film is still in place, and the breather and vacuum foot are still in place. Inspect the vacuum bag for leaks.

3.7.7 Curing

The bagged assembly is now ready for the curing oven. Due to equipment constraints the precise curing profile was not followed. Experimentation by other student groups has shown that a curing temperature of $225°F$ for a minimum of 90 minutes results in an acceptable layup. The bag is placed in the oven, with the vacuum foot disconnected and re-attached to vacuum feed-through connections inside the oven. The oven is started, but the 90 minute timer starts only after the oven has reached $225°F$. The vacuum should be monitored to ensure no leaks were introduced while placing the assembly in the oven. Note that other support materials may be

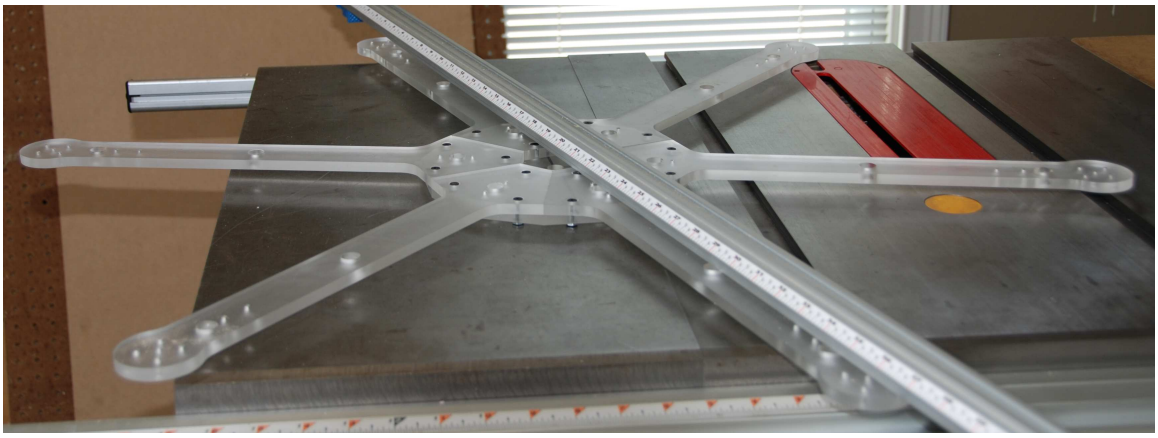


Figure 3.12: Checking the alignment of the assembled frame template.

substituted for aluminum, but material properties may require a longer cure time as they can retard heat flow to the layup. Once the layup has cured, the oven can be turned off and the vacuum source removed. Once the material has cooled enough to handle safely, the bag, breather, and release film layers should be removed.

3.7.8 Machining to Final Dimensions

Since the finished aircraft size is too large to machine in a single pass on either of the CNC mills available, an acrylic template was CNC machined in sections and bolted together. The assembled template is shown in Figure 3.12 is being checked for alignment. The composite blank was attached to the template and machined to final dimensions using a router table as shown in figure Figure 3.13. The acrylic template also provided a drill template so that holes for mounting motors, landing gear, and avionics were accurately located. Holes were drilled using a drill press. Hole dimensions are shown in Figure 3.14.

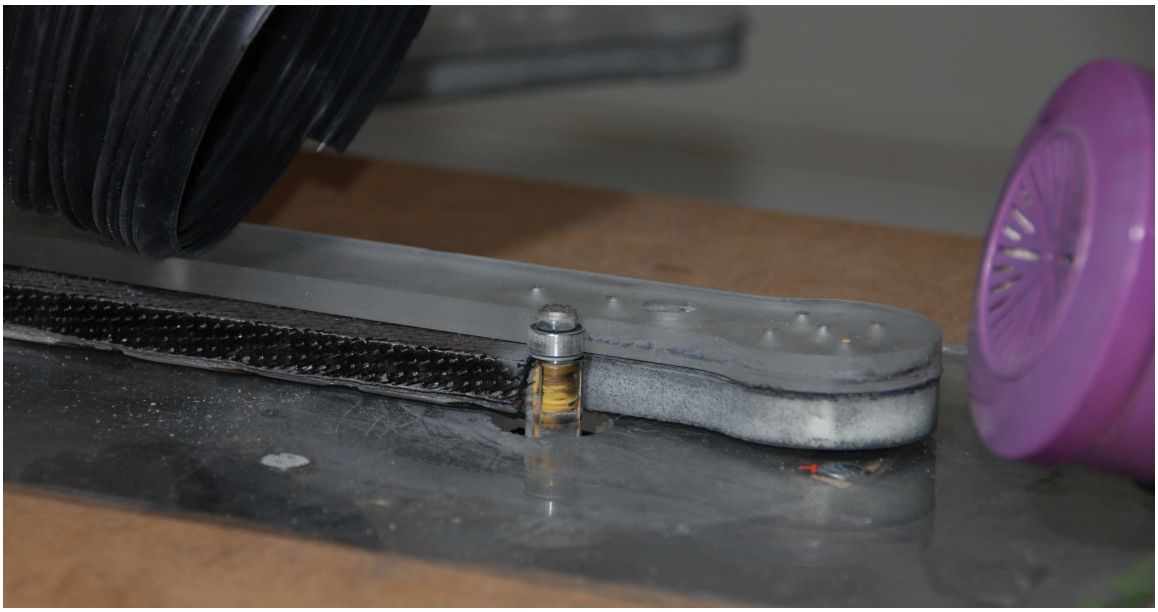


Figure 3.13: Cutting the composite layup to final dimensions using a router table.

The router is equipped with a $\frac{1}{2}$in diameter 1in long flush trim bit with a ball bearing guide. This is a standard woodworking tool, but the carbon fiber is very abrasive and tool life is short. Standard twist drills were used to drill mounting holes. The carbon fiber causes such aggressive wear that a router bit may only last for one or two airframes, and a single $\frac{1}{8}$in drill bit is quite dull after drilling the holes for a single airframe.

3.7.9 Landing Gear Construction

The landing gear consists of 6 legs, one mounted to each arm of the airframe. The legs were constructed using the same composite layup materials and machining techniques as the airframe. Since the attachment point for the landing gear is perpendicular to the layup, threaded inserts were epoxied into the foam core of the legs

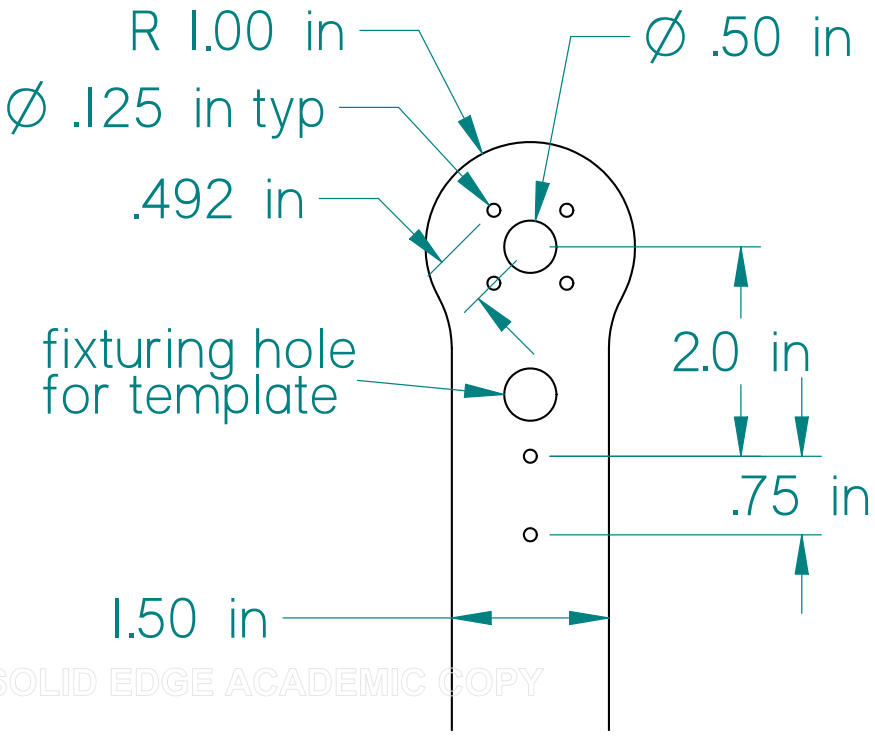


Figure 3.14: Airframe arm detail.

using DP-460. An aluminum bracket was machined to a U-channel and epoxied to the top of the leg to provide a carbon fiber to aluminum load bearing path. #4-40 stainless steel socket cap machine screws were used to attach the legs to the frame.

3.7.10 Avionics Mounting

Plates consisting of multiple layers of carbon fiber were used for mounting avionics. In addition to its other properties, carbon fiber is conductive, which provides a modicum of shielding for the flight computer against EMI from motors and ESCs, and RFI from the telemetry transmitters. Conversely the GPS, RC receiver,

and telemetry antennas were located above the carbon fiber to avoid RF shielding problems. Threaded aluminum spacers of appropriate lengths were used to mount the plates to the airframe.

3.7.11 Fasteners

Throughout the airframe, #4-40 stainless steel socket cap machine screws of appropriate lengths were used to connect the various components. One exception is the motor mounts, where the motors were already drilled and tapped to accept M3 machine screws. Screws installed so that the heads would contact the carbon fiber layup directly were fitted with standard diameter #4 stainless steel washers.

Every fastener must be secured using either a removable adhesive thread-lock [33] or a mechanically locking fastener, such as a lock nut with nylon insert. Any fasteners not positively retained will loosen due to vibrations, causing flight failures. The prop nuts are especially subject to loosening during flight.

3.8 Flight Testing and APM Tuning

Because the APM is typically used on smaller quadrotors, considerable tuning is required to achieve stable flight with the much larger hexcopter. Several PID (Proportional Integral Derivative) loops are used to provide aircraft control, and rate constants for these loops are exposed to the operator in the configuration tool. In order to reduce the opportunity for aircraft damage during this tuning, a Whitman training stand [2] was fabricated and is shown in Figure 3.15.

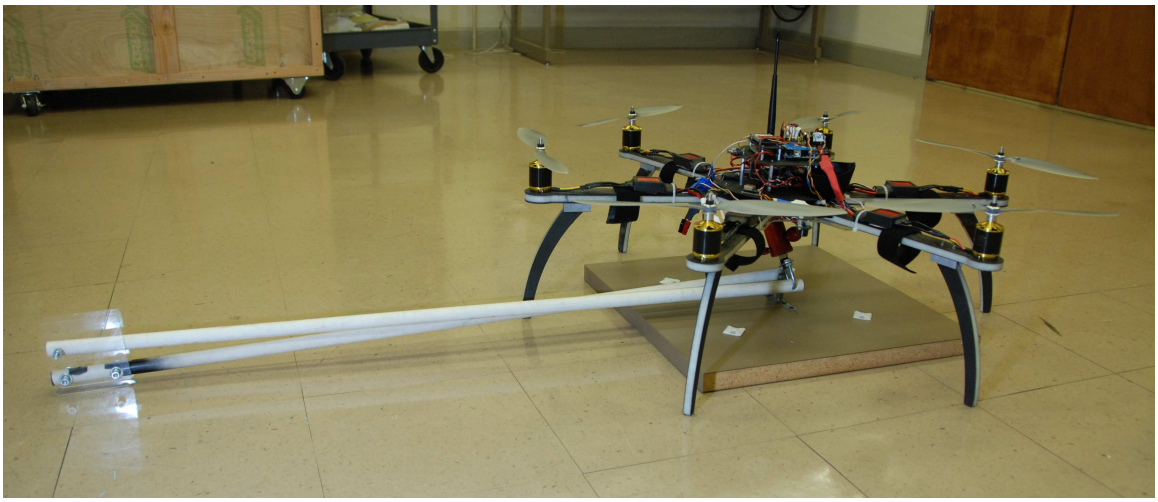


Figure 3.15: Hexcopter shown on the Whitman training stand.

Table 3.5: APM PID tuning parameters

parameter	P tuning value
Stabilize Roll	3.5
Stabilize Pitch	3.5
Rate Roll	0.04
Rate Pitch	0.04

Once tuning reached the point where some control was possible, plans were made to conduct further testing outdoors. Unfortunately FAA regulations do not allow research flights without a Certificate of Authorization (CoA). [3] This process requires a minimum of 60 business days. Test plans were modified to conduct test flights indoors to avoid violating FAA regulations.

The Mission Planner screen for configuring the APM tuning parameters is shown in Figure 3.16. Specific parameter changes are listed in Table 3.5. In addition

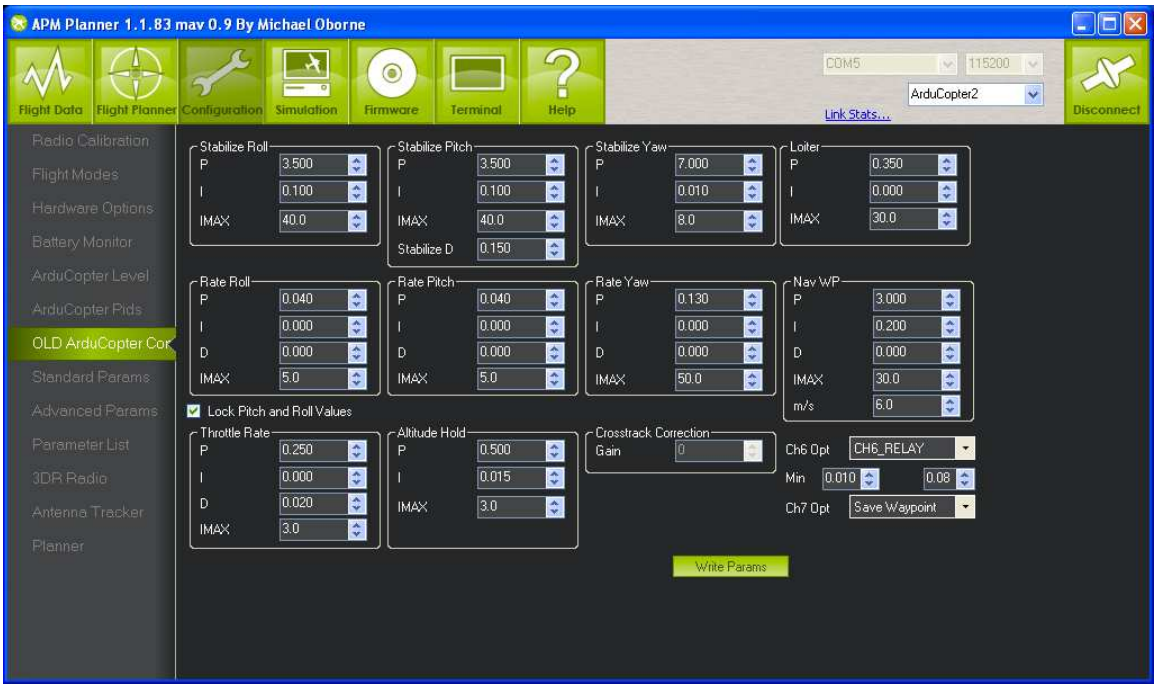


Figure 3.16: Mission planner configuration screen for APM tuning parameters.

to the PID tuning, the ESC update rate was changed to 490Hz. A complete list of APM parameters is provided in Appendix E.

3.9 Auxiliary Power Unit Design

The APU is designed using a commercially available gasoline fueled engine designed for large model aircraft mated with an alternator designed for UAV applications. Gasoline was chosen for its high energy density as illustrated in Table 2.2 The output of the alternator will be filtered and regulated by a separate power supply.

The 3W Modellmotoren 3W-28cs [34] is a 28.5cc 3.55HP (2.64kW) 1210g gasoline engine. It uses a 1:50 ratio mixture of oil and 98 octane gasoline. It has an

Table 3.6: APU Mass Budget

	unit mass (g)	qty	mass (g)
3W-28i motor	1210	1	1210
S675-600 alternator	1880	1	1880
regulator	500	1	500
fuel tank	58	1	58
gasoline (719.7 g/l)	719.7	0.5	359.85
mounting hardware	500	1	500
PMU	500	1	500
total			5007.85

electronic ignition system and requires an external power source between 6.0V and 8.4V. The power management unit will provide throttle control for the engine.

The generator is a Sullivan Products S676-600 [35] three phase brushless alternator rated for 900W to 2700W, depending on engine RPM. It weighs 1800g. The alternator will be custom wound to operate at the optimum speed for the engine so that it produces 12.5V after being rectified and filtered by the power supply.

The estimated APU mass budget is shown in Table 3.6. Fuel consumption remains an unknown; no sources were located that provided consumption data for this engine. Engine cooling, in the form of a small prop or external electric fan, may be required, but it is assumed that the drive rotors will provide enough airflow for adequate engine and alternator cooling during flight.

3.10 Power Management Unit Design

As shown in Figure 3.17 the power management unit (PMU) accepts power from the APU and on board batteries and supplies power to the ESCs, avionics, and

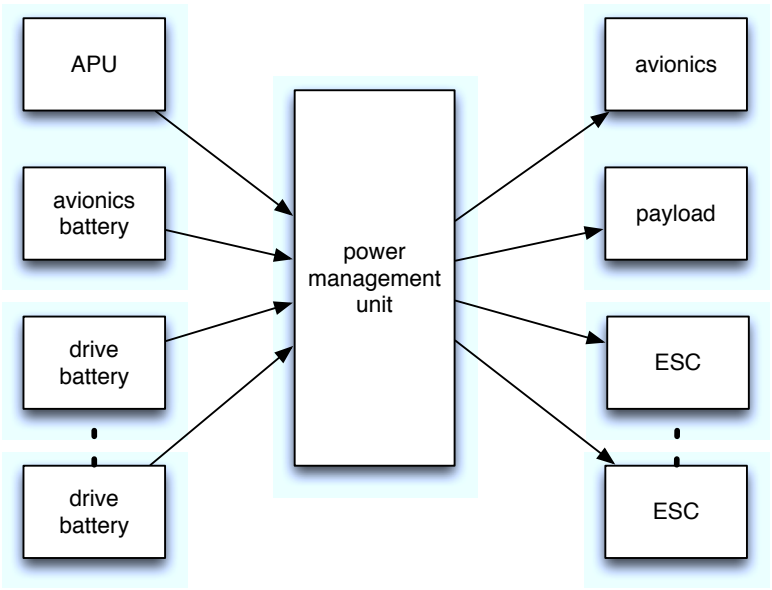


Figure 3.17: Power management unit functional integration with aircraft systems and APU.

payload. It controls the APU to maintain required system power. It monitors the power consumed by each ESC and switches the power supplied to individual ESCs between the APU and batteries as the demand from the individual ESC changes. Since the APU is a gasoline generator, increased demand is met by increasing the engine throttle setting - a process that takes time, hence the need for an instantaneous change to battery power.

The ESC power is supplied from one of two MOSFETs - one for the APU and another for the battery - as illustrated in Figure 3.18. Diodes were considered, but due to the current demands only diodes with large (0.6V) voltage drops were available. This voltage drop would result in a significant power loss - approximately 5% of system power. The voltage drop is also undesirable from a control standpoint, since motor speed is a function of input voltage. There is some power loss in the

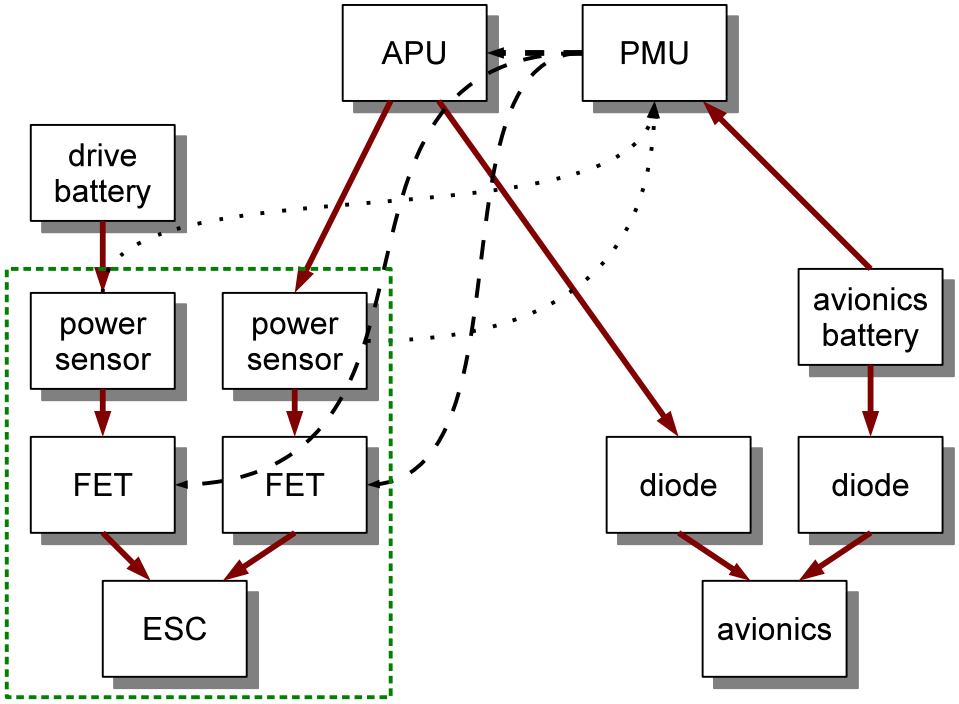


Figure 3.18: Power management unit block diagram. Solid lines indicate power, dashed lines are control signals, and dotted lines are sensor inputs. Drive train components are replicated 6 times; only a singe instance is shown in the dashed box at the lower left.

MOSFET, but power loss due to the typical FET $R_{DS_{ON}}$ dissipation is much less that a 0.6V diode loss.

Avionics power is supplied from either generator power or from a dedicated avionics battery. Since momentary loss of power to the avionics system can cause loss of control of the aircraft, a different management technique is implemented. The avionics battery and APU outputs are connected to the avionics power input via diodes. Power required by the avionics is much less than that of an ESC, so the resulting diode losses are acceptable.

It is desirable for the APU and PMU to supply power to future payloads. Payload power may be supplied by an arrangement similar to the drive train or avionics, or a separate payload battery may be supplied, depending on payload power requirements. The current data acquisition payload power requirement is small enough that it is connected to the avionics source in this design.

3.11 Ground Station

A ground station supports flight operations. The main components are a laptop computer running both the ArduPilot Mission Planner application and a serial data logging utility for the data acquisition payload. This power data is graphed in real time using the LiveGraph [36] utility so that the ground station operator can monitor battery levels and inform the pilot as needed.

The Mission Planner is used to configure the APM parameters, load autopilot missions, and give the ground station operator live data. The armed/disarmed status indicator and attitude indicator are the features used during flight operations. The flight data screen is shown in Figure 3.19. It can also be used to provide a control input so that the aircraft can be flown using a joystick attached to the computer and the APM telemetry link.

A Spektrum DX8 RC transmitter provides the flight control link. It is a 2.4GHz spread spectrum transmitter with telemetry receive capability. The DX8 is configured in airplane mode, standard practice for multicopter use. To simplify APM configuration, the DX8 inputs for aileron, elevator, rudder, and aux3 were reversed.

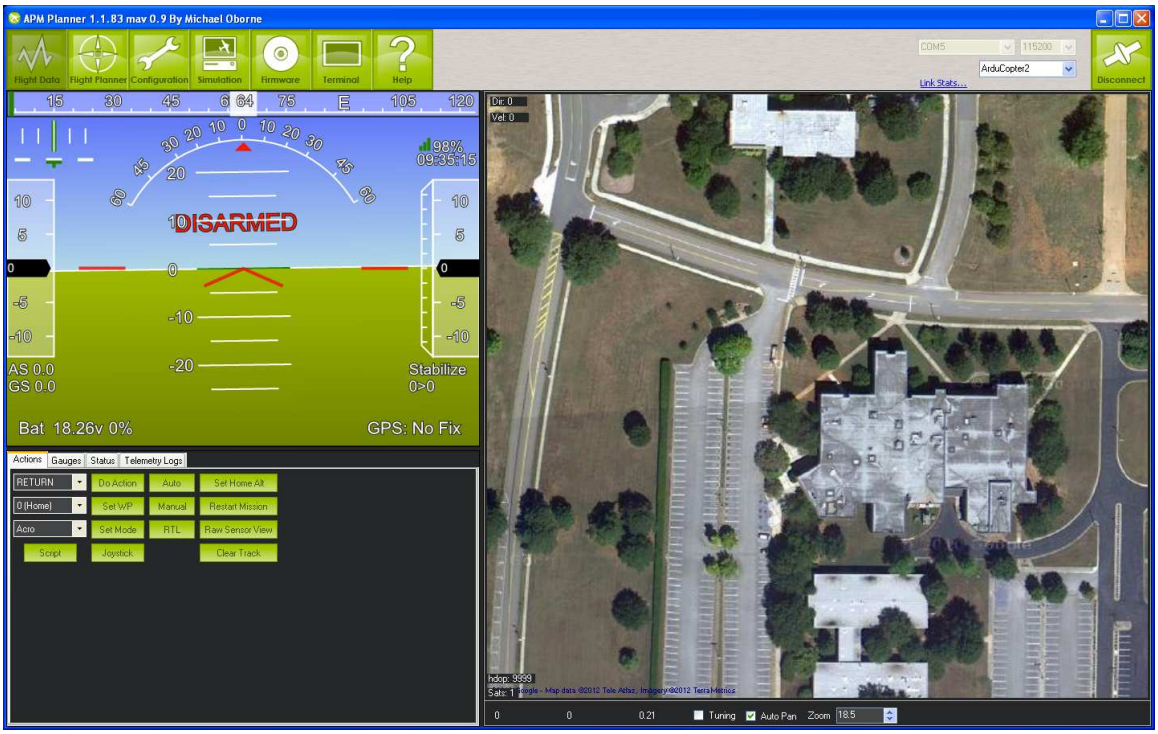


Figure 3.19: Mission planner flight data view.

This allowed use of the default parameters in the APM receiver configuration which reduced likelihood of errors during configuration.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Power Consumption During Flight

Flight tests were conducted with the instrumented hexcopter, shown in Figure 4.1. The test regime was to install three freshly charged batteries, launch the aircraft, and stay in the air until the battery voltage dropped to a minimum safe voltage or aircraft control deteriorated to unsafe levels due to loss of thrust. A ground station operator monitored battery voltage and reported to the pilot. The aircraft was launched as soon as was safely possible so as to avoid power loss by the ESCs of the idle system. A warning threshold of 10.5V and an immediate landing threshold of 9V were chosen to avoid battery damage. Preliminary work with the Gaui showed a sharp increase in the rate of voltage drop once the battery voltage was below 10V. A 0.5V margin was added to this number to ensure power remained to execute a safe landing. In addition to the battery voltage levels, the pilot may judge the aircraft had insufficient power for safe control.

Two primary aircraft weights were flown to measure power consumption and flight duration. One was the basic aircraft with no payload other than the power data acquisition system. The second profile was the same configuration with 5kg

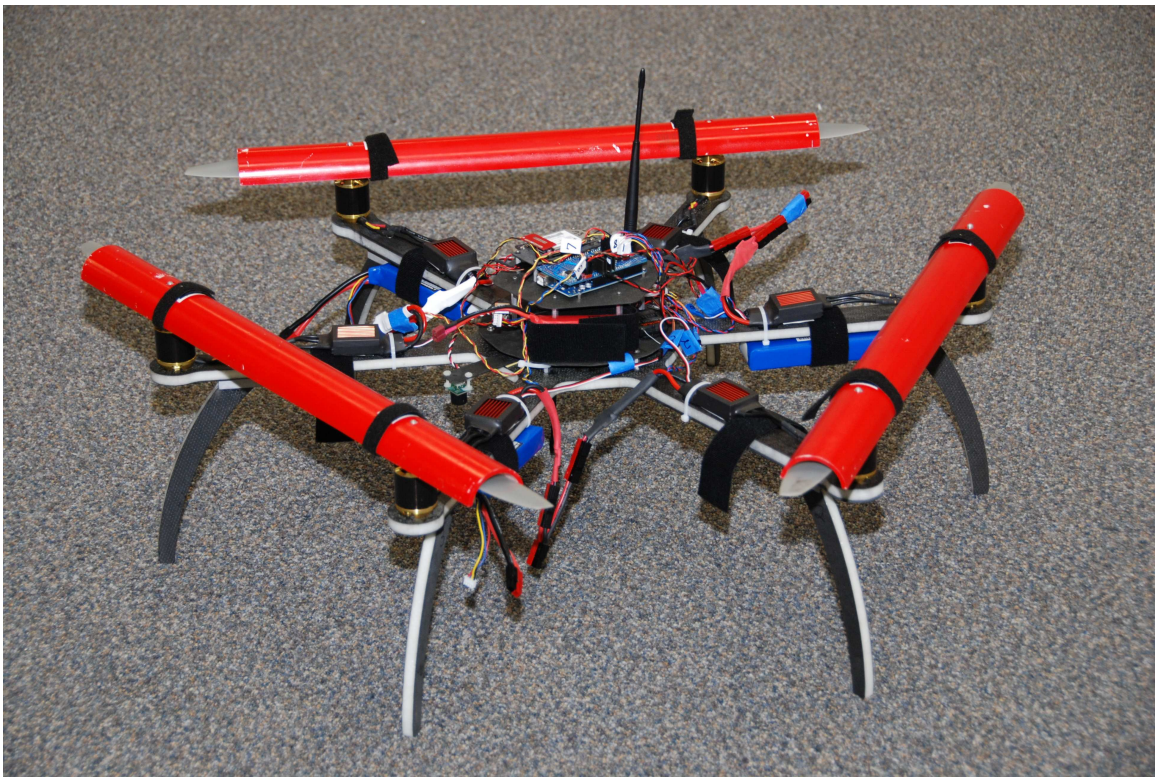


Figure 4.1: Hexcopter ready for test flight. Safety covers immobilize props and protect the operator while powering up the aircraft.

Table 4.1: Experimental flight results.

configuration	number of batteries	weight (GVW, kg)	flight time (min)	average power (W)
no payload	3	5.0	11.9	669
w/payload	3	10.0	4.0	1549
w/payload	3	11.0	2.9	1850
w/payload	3	12.0	1.2	2046

of ballast simulating the APU mass. Additional configurations with total aircraft weights of 11kg and 12kg were also flown to test the aircraft's ability to lift additional payload that could be used for fuel to further extend flight times. Results are listed in Table 4.1.

Raw data from the data acquisition payload is a stream of ASCII printable comma separated values. There are 15 fields: the time of the sample, in ms since microcontroller reset; the current (in amps) and voltage (in volts) for each of the 6 ESC DC input sensors, and current and voltage for the avionics battery output. This data was captured on the ground station laptop for storage and later analysis. Data was sampled at 4Hz. Samples were instantaneous measurements of their respective inputs. The ADC inputs were sampled sequentially and converted using the transfer functions provided by the vendor, then all values were transmitted to the ground station using the 9XTend payload radio.

In order to find total aircraft power consumption, the power at each current sensor is calculated and the resulting values are summed to calculate the total system power. DC power [37] is calculated using

$$P = VI \tag{4.1}$$

with units of Watts, Volts, and Amps, respectively. Total system power is calculated using

$$\begin{aligned} P_{total} &= P_{avionics} + \sum_{i=1}^{6} P_{ESC_i} \\ &= V_{avionics} \cdot I_{avionics} + \sum_{i=1}^{6} V_{ESC_i} \cdot I_{ESC_i} \end{aligned} \tag{4.2}$$

This calculation was performed for each sample recorded. Data that did not represent actual in-air flight data was identified by visual inspection and discarded. The

resulting data for the 5kg, 10kg, 11kg, and 12kg test flights are shown in Figure 4.3, Figure 4.4, Figure 4.5, and Figure 4.6, respectively. The per-sample power value for these in-flight samples were averaged and recorded as the average power consumption shown in Table 4.1. The observed average is also included in Figure 4.2, along with the calculated power using Equation 3.3 and data from eCalc simulations.

It is interesting to note that the observed power consumption is less than that predicted by both momentum theory and the simulation tool. One possible source of this discrepancy is the figure of merit for the prop. In the momentum theory calculation, we used an estimate of 0.5 for the FM. Data for similar props indicate the FM may be closer to 0.57. Another source of the discrepancy may be due to ground effects - the condition where the thrust required to hover is reduced by the proximity of the aircraft to the ground.

No data is available on fuel consumption. Research concerned with fuel consumption for small UAS sized engines emphasize that reliable fuel consumption data is only available by measuring a particular engine under the actual workload. The RC hobby community does have a rule of thumb: take the engine displacement in cc and use that value as the worst case fuel consumption in $\frac{ml}{min}$. The 3W 28cs engine displacement is 28cc. Using this value and converting volume to mass, the worst case fuel consumption is:

$$\begin{aligned} &= 28\frac{ml}{min} \cdot 0.719\frac{g}{ml} \cdot 60\frac{min}{hr} \\ &= 1200\frac{g}{hr} \end{aligned} \tag{4.3}$$

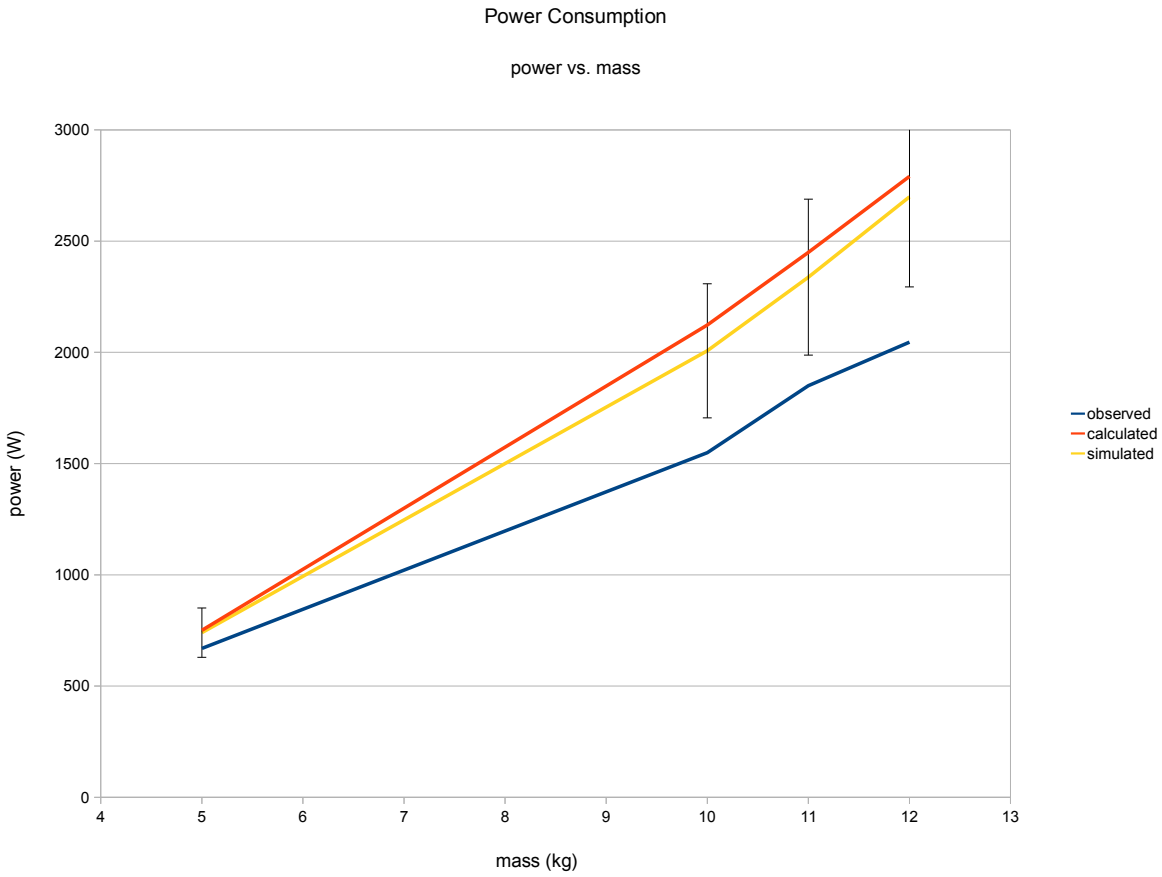


Figure 4.2: Mass vs. power summary.

The APU mass budget in Table 3.6 allows 360g for fuel resulting in a worst-case runtime of 18 minutes. Flight tests indicate the aircraft can lift an additional 2kg for a total worst-case flight time estimate of 2 hours with APU power.

Another approach to extend the flight time is to simply add 5kg of additional batteries. The 10kg GVW flight with 3 batteries lasted 4 minutes, which gives a battery consumption rate of $0.75\frac{batteries}{min}$. 13 additional batteries weigh $375g \cdot 13 = 4875g$. The 16 total batteries would provide an estimated flight time of $\frac{16batteries}{0.75\frac{batteries}{min}} = 21.3$ minutes.

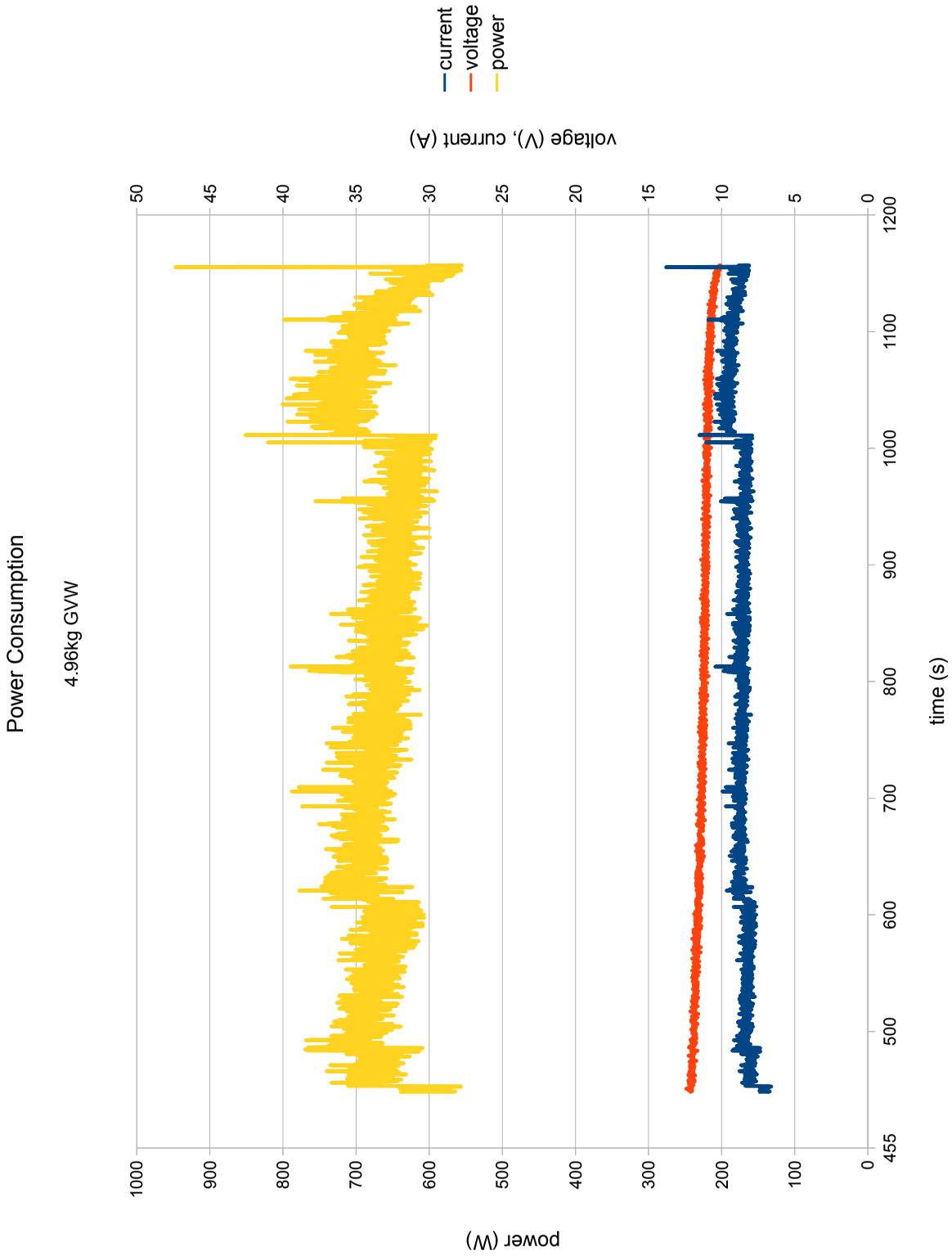


Figure 4.3: Current, voltage, and power for 5.0 kg GVW flight.

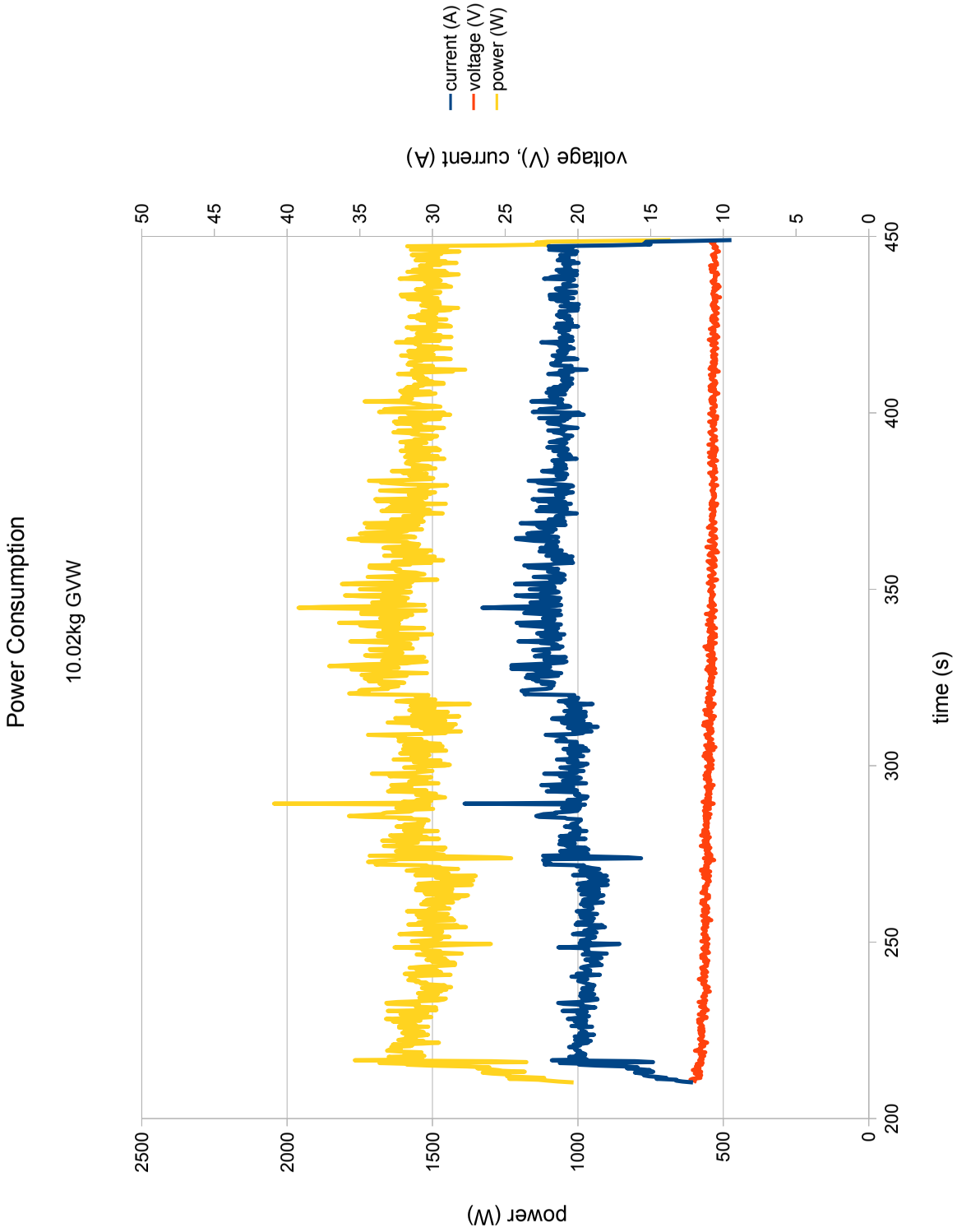


Figure 4.4: Current, voltage, and power for 10.0kg GVW flight.

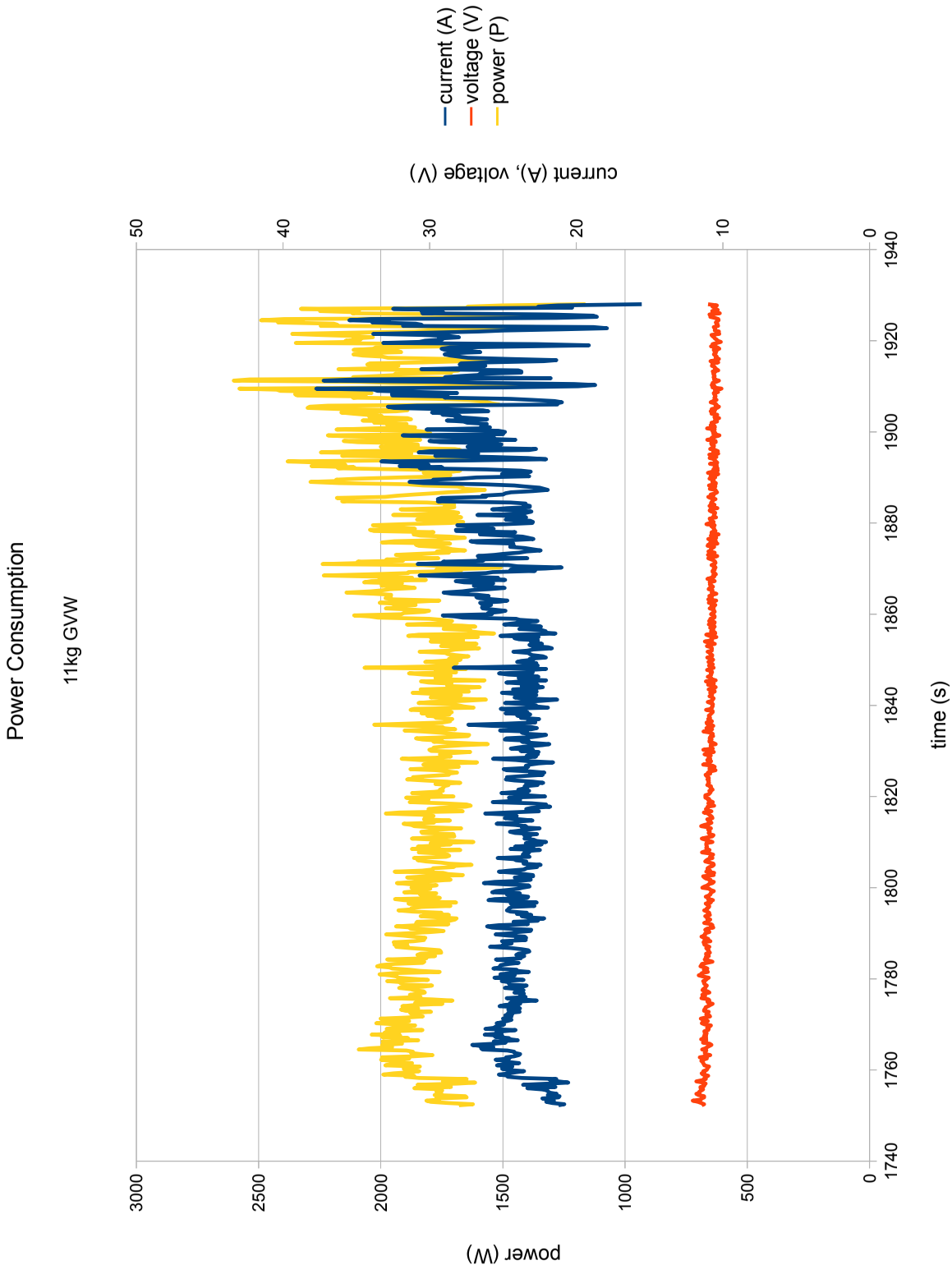


Figure 4.5: Current, voltage, and power for 11.0kg GVW flight.

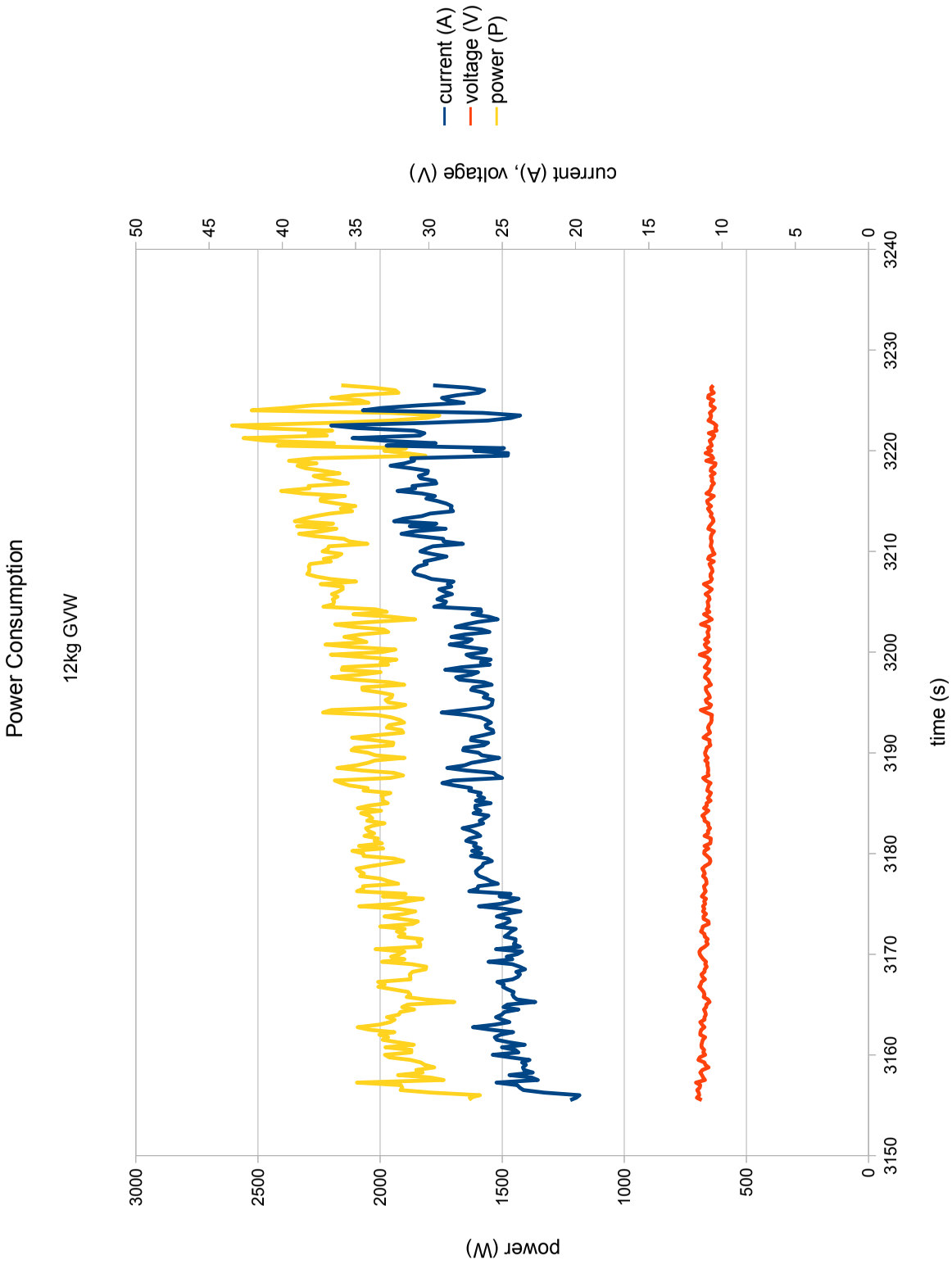


Figure 4.6: Current, voltage, and power for 12.0kg GVW flight.

4.2 ArduCopter Code Analysis

The APM hardware has different software to be installed, depending on the target vehicle. Code is available for ground vehicles, fixed wing aircraft, and rotorcraft (both traditional helicopter and multicopter). This section reports the analysis of the rotorcraft code, known as ArduCopter.

LDRA Testbed is an industry standard software suite that provides test, analysis, and requirements traceability for software systems. It can verify that code meets several industry standards, perform dynamic code analysis, and other tests. The Joint Strike Fighter (JSF) [38] coding standard was selected since it was designed for aircraft software systems. Many of the coding requirements for JSF and other standards are concerned with maintainability of code. Such standards are very important where the project will be handled by many programmers or a long time period. An examination of open source software projects has found them to be lacking in the area of test planning. [39]

The ArduCopter project is an open source project with a small number of programmers. Additionally, ArduCopter uses code from the Arduino project. The ArduCopter code is 13,019 lines after pre-processing by the Arduino suite, with an additional 3800 lines from the Arduino code base. No coding standard was enforced during its development, and much of the style does not adhere to JSF standards. As is typical of many open source software projects, no software requirements specification (SRS), software design document (SDD), or interface control document (ICD) were found making it more difficult to understand the structure and purpose of the code

modules, valid ranges of variables, etc. Thus, a complete safety analysis would require a significant reverse engineering effort which was outside the scope of this research.

A summary of JSF coding standard violations appears in Table 4.2, Table 4.3, and Table 4.4. Many of the violations are not pertinent to this safety analysis because adherence to the JSF standard was not a requirement for the developers. Additionally, the corresponding volume of output from the tool could mask violations that correspond to actual code defects.

To reduce the analysis output to a more useful level, many style related rules were ignored. For example, the number of spaces for indentation and the alignment of brackets are important for consistently readable code, but not directly critical for flight safety. Other code was flagged due to unusual code constructs resulting from the memory addressing in a Harvard architecture microcontroller. [40]

The ArduCopter code base is based on the Arduino environment, which uses C and C++ as its core, but performs automatic generation of function prototypes, includes libraries and provides other functionality on behalf of the user. Since not all the code is in a standard C/C++ file format, it was necessary to use the Arduino IDE to process the code into standard form and extract the output of the Arduino processing stage before the code is passed to the AVR-gcc compiler. Static analysis of the generated was performed using LDRA's Testbed [41] tool. No clearly identifiable defects were discovered.

Testbed's Quality Review Report is summarized in Figure 4.7. The code is scored in the areas of clarity, testability, and maintainability on a scale of 0 to 100%, with 100% representing no violations detected. Clarity examines coding style

Table 4.2: ArduCopter violations of JSF "SHALL" standards.

number of violations	standard
26	Use of break statement in loop.
1	Statement with no side effect.
4	#pragma used.
6	Logical comparison of pointers.
20	Macro replacement list needs parentheses.
1236	Basic type declaration used.
61	Casting operation on a pointer.
63	Casting operation to a pointer.
23	Volatile declaration.
135	Identifier not declared on new line.
240	Define used for numeric constant.
7	Class initialiser out of order.
17	Start of enumeration is upper case.
10	No copy constructor for class with pointers
10	No assignment operator for class with pointers
54	Lower case suffix to literal number.
1	Operator = doesn't return reference to *this.
439	Found #if, #ifdef, #else, #elif .
201	Filename in #include not in <>.
40	Found #ifndef.
25	Virtual class members need virtual destructor.
222	Arithmetic performed on unsigned values.
499	Use of C type cast.
5	Array parameter found.
109	Use of the NULL macro.
18	Enumeration is not all lower case.
46	Const variable is not all lower case.
4	Declaration of type not in header file.
7	Class member name reused.
27	Expression needs brackets.
44	Cast from integral type to pointer.
51	Array passed as actual parameter.
10	Inline or template function not in header.
3	Virtual member called in ctor/dtor.
13	Cyclomatic complexity greater than 20.
174	Global not initialised at declaration.
2	Constructor calls virtual function.

Table 4.3: ArduCopter violations of JSF "WILL" standards.

number of violations	standard
5	More than 7 parameters in procedure.
49	Else alternative missing in if.
1	Empty switch statement.
31	Use of pointer arithmetic.
38	Use of redundant cast.
8	Label is not part of switch statement (MR).
22	Non standard character in source.
238	No newline after semi colon.
26	Spaces round > or [] operators.
147	Space between unary operator and operand.
1794	{ or } not on line by itself.
158	Input line exceeds limit.
2273	... contents not indented by 4 spaces.
7	Access specifiers in invalid order.
1118	Use of numeric literal in expression.
229	Use of old style /* comments in C++
307	User name starts with underscore.
31	Start of class/struct/union/enum lower case.
2	Procedure exceeds 200 source lines of code.
61	Hexadecimal number with lower case char.
8	Empty initialisation exprsn in for loop.
434	Found #define.
3	Switch has missing or extra cases.
13	Functions defined in header file.
1	First parameter not on same line as function.
19	Pointer to function declared without typedef.
1410	} not aligned vertically below {.
525	Asterisk or ampersand not attached to type.
153	Function name is not all lower case.
820	Variable name is not all lower case.
1	Switch has only 1 case and default.
57	Name letters after first not lower case.
47	Constructor has insufficient initialisers.
9	Type declaration with variable definition.
7	Extern declaration is not in header file.
201	Parameter not declared on new line.
2	Variable should be defined once in only one file
156	Member function should be declared const.
7	Function return value not used.
13	Identifier is typographically ambiguous.

Table 4.4: ArduCopter violations of JSF "SHOULD" standards.

number of violations	standard
11551	Tab character in source.
136	Class data is not explicitly private.
1	Class definition not needed in file.
7	Header file name does not include class name.
3	File name does not contain defined class name.
1529	Non local declaration not in a namespace.
14	Inline member has more than 2 statements.
7	Pointer to pointer declared.

for consistency and compliance with good programming practice. Testability is a measure of features that may cause code to be more difficult to test. Maintainability examines code for features that make code more difficult to understand for humans who must modify the code. [42]

It should be noted that the Arduino compilation tools perform preprocessing on the code. Because the Arduino programs are not complete C++ programs until this preprocessing occurs, the LDRA tools will not give a useful assessment of the original code. This preprocessing results in a syntactically correct source file, but the style of the machine generated code results in poorer LDRA scores than the original code may warrant.

It is clear from the LDRA analysis that the code is under-documented. All source code files and functions scored poorly on clarity and maintainability due to this lack of documentation. Some results were of concern not because of a particular error condition, but because the resulting code can be difficult to understand, and as a result difficult to maintain. One particular example involves `goto`.

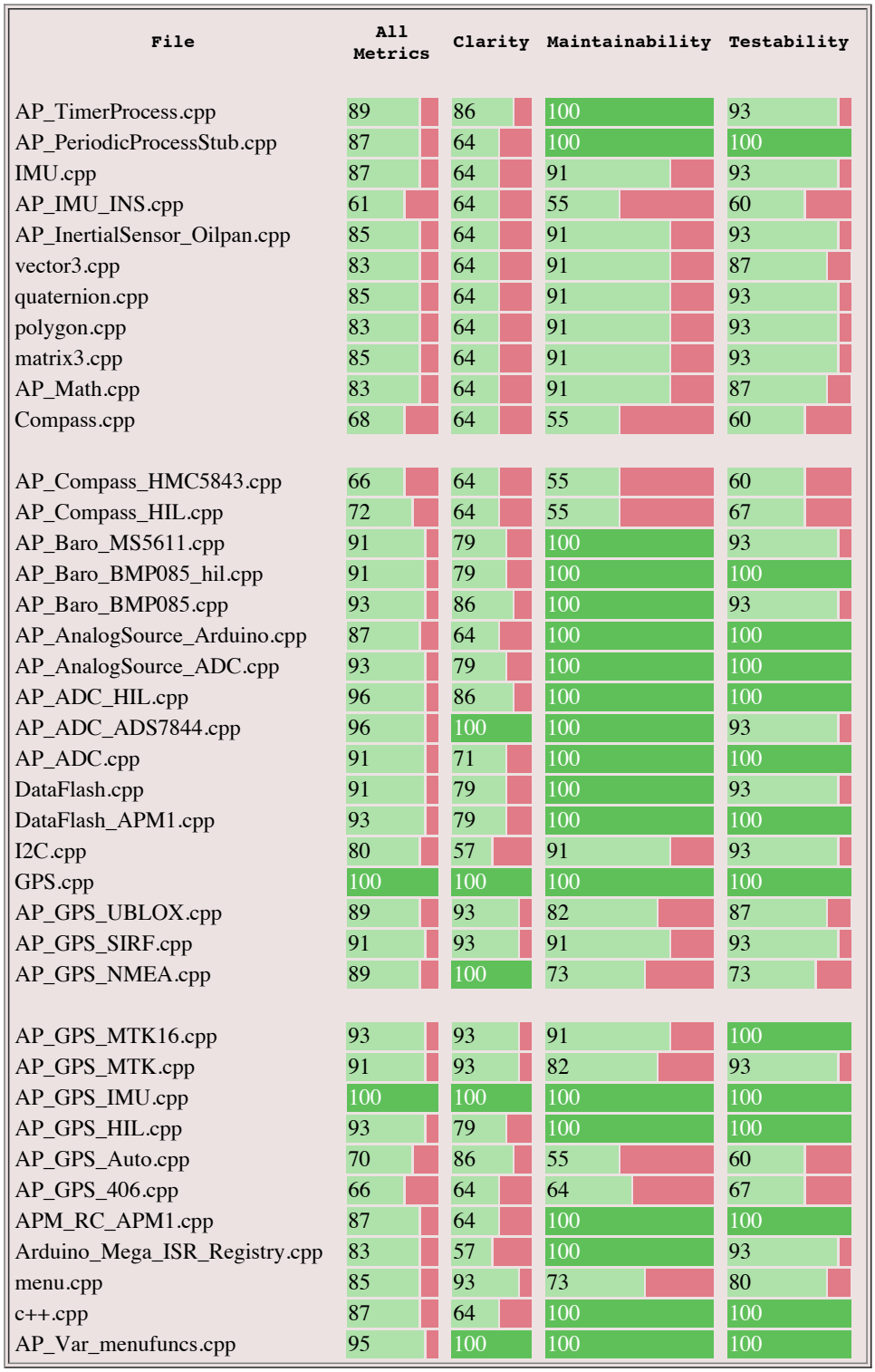
LDRA Testbed ® Quality Review Report

Group : arducopter-build

Overall Results - Percentage of metrics passing

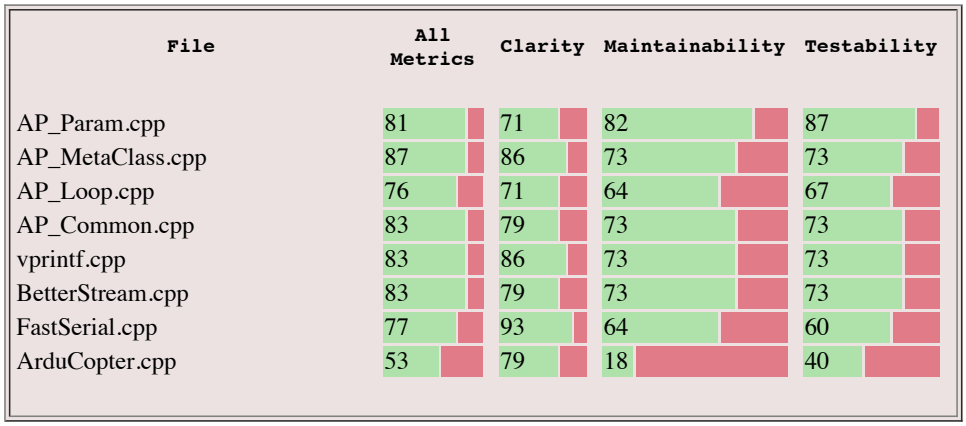
File	All Metrics	Clarity	Maintainability	Testability
arducopter-build	87	78	90	89
wiring_shift.c	91	79	100	100
wiring_pulse.c	87	93	73	87
wiring_digital.c	94	86	100	93
wiring_analog.c	100	100	100	100
wiring.c	98	100	100	93
WInterrupts.c	96	93	100	93
pins_arduino.c	100	100	100	100
SPI.cpp	91	79	100	100
WString.cpp	79	64	91	80
WMath.cpp	85	64	100	93
Tone.cpp	94	93	100	93
Print.cpp	87	64	100	93
main.cpp	83	57	100	100
HardwareSerial.cpp	100	100	100	100
AP_Declination.cpp	94	100	91	87
memcheck.cpp	87	64	100	100
AP_Relay.cpp	87	64	100	100
AP_OpticalFlow_ADNS3080.cpp	89	79	100	93
AP_OpticalFlow.cpp	93	86	100	93
AP_RangeFinder_MaxsonarXL.cpp	91	71	100	100
RangeFinder.cpp	91	71	100	100
AP_Motors.cpp	89	64	100	100
AP_MotorsHexa.cpp	98	100	100	100
RC_Channel.cpp	87	71	91	87
RC_Channel_aux.cpp	98	100	100	93
AC_PID.cpp	87	64	100	93
APM_PI.cpp	87	64	100	100
AP_AHRS_Quaternion.cpp	93	93	100	87
AP_AHRS_HIL.cpp	87	64	100	100
AP_AHRS_DCM.cpp	93	93	100	93

Figure 4.7: LDRA quality analysis summary graph.



File	All Metrics	Clarity	Maintainability	Testability
AP_TimerProcess.cpp	89	86	100	93
AP_PeriodicProcessStub.cpp	87	64	100	100
IMU.cpp	87	64	91	93
AP_IMU_INS.cpp	61	64	55	60
AP_InertialSensor_Oilpan.cpp	85	64	91	93
vector3.cpp	83	64	91	87
quaternion.cpp	85	64	91	93
polygon.cpp	83	64	91	93
matrix3.cpp	85	64	91	93
AP_Math.cpp	83	64	91	87
Compass.cpp	68	64	55	60
AP_Compass_HMC5843.cpp	66	64	55	60
AP_Compass_HIL.cpp	72	64	55	67
AP_Baro_MS5611.cpp	91	79	100	93
AP_Baro_BMP085_hil.cpp	91	79	100	100
AP_Baro_BMP085.cpp	93	86	100	93
AP_AnalogSource_Arduino.cpp	87	64	100	100
AP_AnalogSource_ADC.cpp	93	79	100	100
AP_ADC_HIL.cpp	96	86	100	100
AP_ADC_ADS7844.cpp	96	100	100	93
AP_ADC.cpp	91	71	100	100
DataFlash.cpp	91	79	100	93
DataFlash_APM1.cpp	93	79	100	100
I2C.cpp	80	57	91	93
GPS.cpp	100	100	100	100
AP_GPS_UBLOX.cpp	89	93	82	87
AP_GPS_SIRF.cpp	91	93	91	93
AP_GPS_NMEA.cpp	89	100	73	73
AP_GPS_MTK16.cpp	93	93	91	100
AP_GPS_MTK.cpp	91	93	82	93
AP_GPS_IMU.cpp	100	100	100	100
AP_GPS_HIL.cpp	93	79	100	100
AP_GPS_Auto.cpp	70	86	55	60
AP_GPS_406.cpp	66	64	64	67
APM_RC_APM1.cpp	87	64	100	100
Arduino_Mega_ISR_Registry.cpp	83	57	100	93
menu.cpp	85	93	73	80
c++.cpp	87	64	100	100
AP_Var_menufuncs.cpp	95	100	100	100

Figure 4.8: LDRA quality analysis summary graph (continued).



File	All Metrics	Clarity	Maintainability	Testability
AP_Param.cpp	81	71	82	87
AP_MetaClass.cpp	87	86	73	73
AP_Loop.cpp	76	71	64	67
AP_Common.cpp	83	79	73	73
vprintf.cpp	83	86	73	73
BetterStream.cpp	83	79	73	73
FastSerial.cpp	77	93	64	60
ArduCopter.cpp	53	79	18	40

Figure 4.9: LDRA quality analysis summary graph (continued).

Function `vprintf()` contains 9 `goto`s with 5 different targets in 421 lines of code. The `vprintf()` function is called indirectly 227 times (with an additional 73 calls in code currently commented out) throughout the ArduCopter code base. While there are apparently no errors in this code, the structure makes it more difficult to verify the code, and the ramifications of errors in a function called this extensively could be severe.

Another issue is the use of implicit type conversion. LDRA Testbed flagged many instances of implicit conversions in expressions involving both signed and unsigned integers. The errors resulting from a mistake in switching from signed to unsigned may be more likely to occur on the 8 bit ATmega2560, since standard integer sizes are smaller than on typical 32 bit systems.

4.3 Prop Collet Loss Incident

During initial hexcopter testing and APM tuning, a single prop was lost during flight. The failure occurred when the collet assembly attaching the prop to the motor shaft slipped off. Sufficient control remained to land the aircraft without damage. This incident supports the decision to use a hexcopter design rather than a quadrotor. Similar incidents during Gaui quadrotor flights resulted in immediate loss of control resulting in a crash.

It is unclear what caused the collet to slip. It is possible that the prop nut (which in turn tightens the collet) was tightened insufficiently. It is also possible that some contamination between the shaft and the collet provided enough lubrication to allow the collet to slip off at high motor speeds. No further incidents have been observed.

4.4 Uncontrolled Launch Incident

One flight session was terminated when the aircraft spontaneously powered up for approximately 1 second shortly after arming for flight. This incident resulted in major airframe damage. The incident occurred on the first flight after adding the second 9XTend radio to the aircraft. This observation led to the hypothesis that RFI was the cause of the anomaly.

An attempt to reproduce the problem was made. Props were removed for safety. After an hour of testing, the problem had not reproduced itself. Because the

APM automatically disarms after 30 seconds of inactivity, there is a short window in which the problem can occur.

Further RFI susceptibility testing was conducted using a 5W 2m amateur radio handheld transceiver. One instance of the anomalous behavior was observed; the motors powered up for approximately 1 second with no control inputs applied at the RC transmitter. In addition to the repetition of the anomalous throttle-up, sensor and RC receiver anomalies were observed in the presence of the RF source. Sensor reports of 20° pitch and 30° roll were observed while the aircraft was flat, level, and unmoving. The RC receiver status LEDs blinked and faded in a random and undocumented manner. This receiver behavior may indicate that RFI in the receiver caused false command inputs to the APM.

The 9XTend transceivers were reconfigured to reduce the RF output power to 10mW (from the default 1W). Several flights have been conducted since the modification, with no recurrence of the anomaly. A range test showed communications were reliable in excess of 125 ft, more than adequate for our indoor test environment. Additional range testing is required before outdoor flights of longer range are conducted. Other possible RFI reduction methods that were not implemented are presented in Table 4.5.

RF power loss is a function of distance squared. [43] Even small separations can have a noticeable effect in reducing RFI. Relocating the transceivers to the outer extremities of the airframe will decrease the amount of RF power from the transceivers at the APM and RC receiver. This relocation would require 20in of cabling between the transceiver and its data source. The system currently uses 5V TTL signaling be-

Table 4.5: Possible Avionics RFI Reduction Methods

relocate transceivers and antennas away from the RC receiver and APM
shield the APM
install ferrite cores on cables connected to the RC receiver
optoisolating the APM
use a less RFI susceptible RC receiver
use avionics telemetry link for control
shield motors and ESCs

tween the APM and the radio. This length, especially in light of the RFI problems, would require RS232 signal levels for reliable communications. Other interconnects even less susceptible to interference, such as RS42 differential signaling, may be necessary.

The use of optoisolators would reduce RFI and EMI that enters the APM via interface wiring. Optoisolation would require design of an interface or a design of a custom version of the APM hardware. Some COTS ESCs provide optoisolation to reduce EMI generated by ESC switching and that generated by the motors.

This incident also highlighted a less desirable aspect of the unibody frame design: it is more difficult to repair. If a modular design had been chosen, the damaged arms could have been replaced quickly and more easily than repairing the damaged unibody frame. This ease of repair appears to outweigh the modest increase in airframe mass, and patches to repair the unibody frame can outweigh the penalty of the modular frame.

CHAPTER 5

CONCLUSIONS

A hexcopter capable of lifting a 5kg payload has been designed, fabricated and flown. Detailed power consumption data has been recorded during each test flight, and flight endurance has been measured for the hexcopter with no payload, the target 5kg payload, and additional payloads of 6kg and 7kg to evaluate capacity for increased fuel payloads. The design meets predicted power consumption values. A hybrid APU has been designed that meets the power and weight requirements.

The use of open source software in safety critical systems raises serious safety concerns. While the static analysis of the ArduCopter flight computer code detected no flaws that would have a direct impact on flight safety, it is clear that some modules have serious readability, maintainability, and testability issues. The analysis was complicated by the lack of documentation for this open source product.

5.1 Future Work

Construction of the APU and PMU would complete the system and allow for full endurance testing. Obtaining a CoA from the FAA would allow this testing to take place, since no facility for indoor testing of an internal combustion engine

over extended periods of time is available. The safety analysis can be extended by performing dynamic analysis of the ArduCopter code.

The aircraft can also serve as a platform for noise reduction research. Passive noise reduction using custom prop designs is one area being explored. Active noise cancellation techniques may be enabled by the larger electrical power budget resulting from APU power generation.

Fuels other than gasoline can also be explored. The engine used in the APU design has a heavy fuel version which can use JP-5, JP-8 and D-2 diesel in active development. This is particularly attractive for military applications.

APPENDICES

APPENDIX A

ACRONYMS AND DEFINITIONS

ADC	analog to digital converter
Ah	Amp-hour
APM	ArduPilot Mega
CNC	computer numerical control machining
CoA	Certificate of Authorization (FAA)
COTS	commercial off the shelf
EMI	electromagnetic interference
ESC	electronic speed controller
GPS	global positioning system
GVW	gross vehicle weight
ISM	Industrial, Scientific, and Medical. Refers to a class of radio devices or frequencies that do not require the operator to obtain FCC licensing
mAh	milliamp-hour
MGVW	maximum gross vehicle weight

PID	proportional-integral-derivative
PMU	power management unit
prepreg	a composite material, e.g. carbon fiber or fiberglass, that is pre-impregnated with resin
RC	radio controlled
RFI	radio frequency interference
UAV	unmanned aerial vehicle
UAS	unmanned aerial system

APPENDIX B

ECALC SIMULATION OUTPUT

Additional output from the eCalc simulation tool for 11kg and 12kg GVW flights are in Figure B.1 and Figure B.2. Data for the 5kg and 10kg GVW flights are in Figure 3.3 and Figure 3.2, located in Chapter 3. This tool uses proprietary empirical data gathered for specific components. Availability of this simulation data influenced component selection.

`http://www.ecalc.ch/xcoptercalc_e.htm`

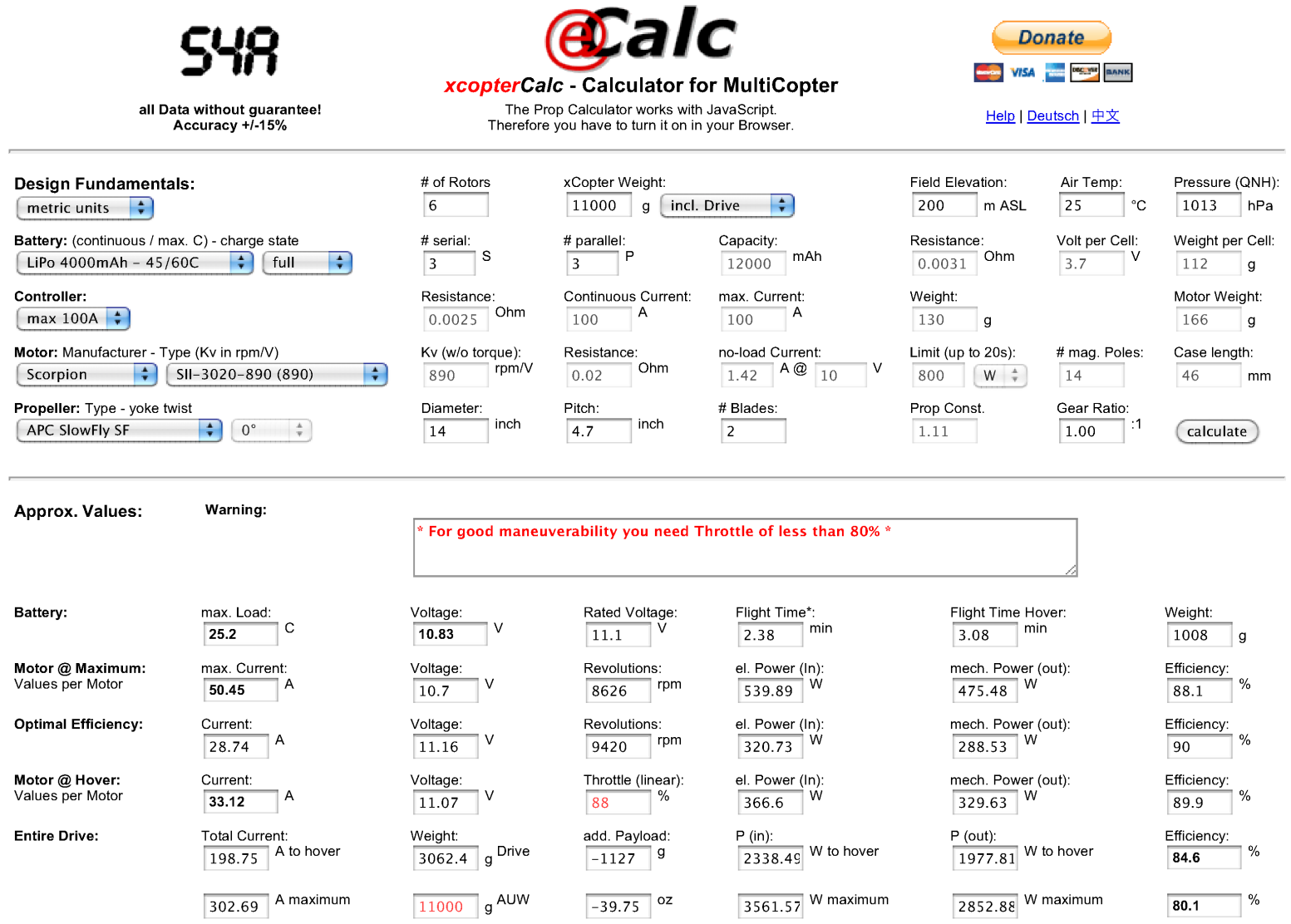


Figure B.1: eCalc simulation output, 11kg GVW.

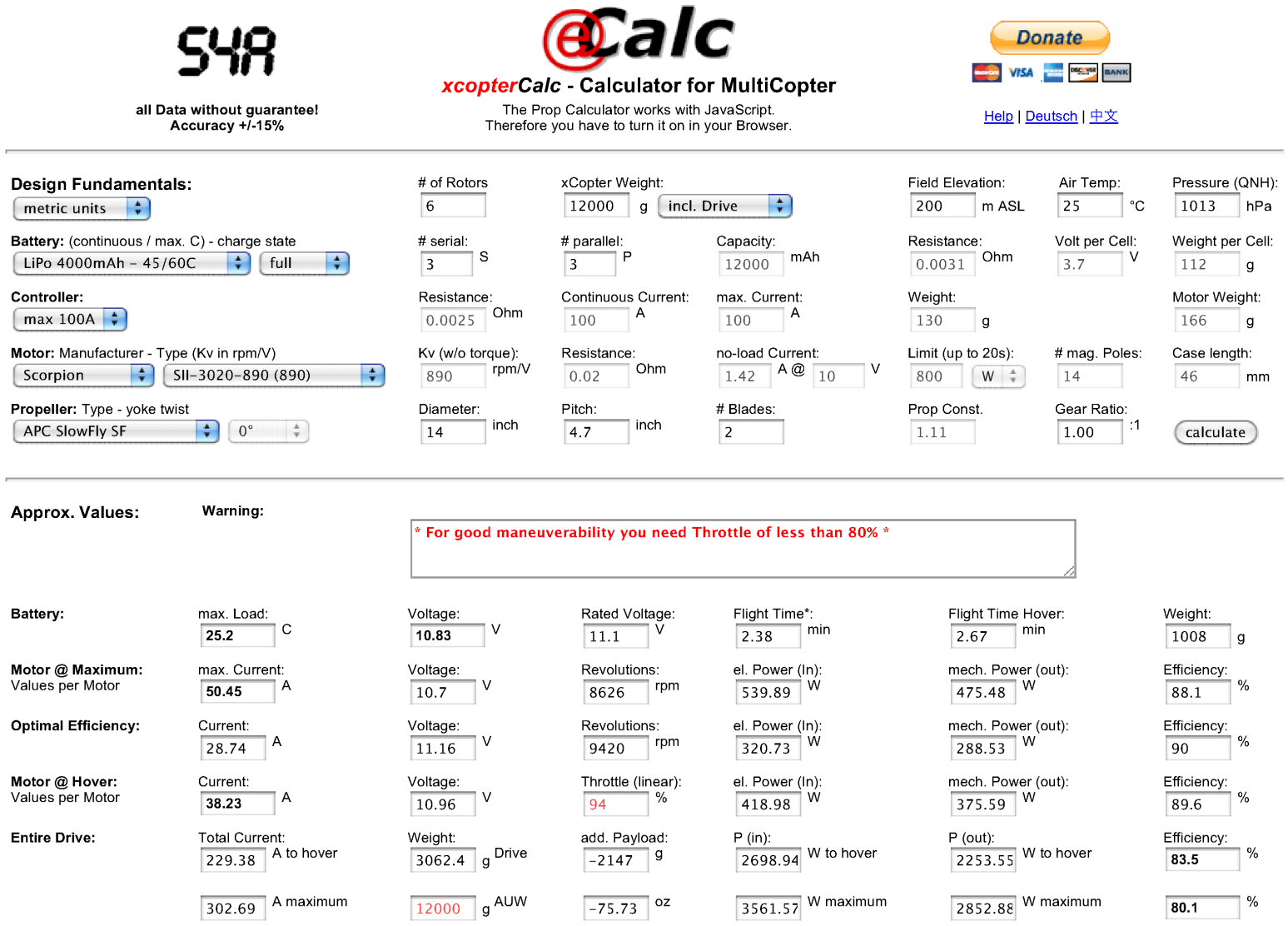


Figure B.2: eCalc simulation output, 12kg GVW.

APPENDIX C

FLIGHT OPERATIONS CHECKLISTS

C.1 Pre-Flight Checklist

☐ ensure flight area is safe, free of hazards and non-essential personnel

☐ inspect props for damage

☐ inspect aircraft for loose fasteners/connectors

☐ place aircraft on level launch site

☐ deploy avionics and payload antennas to vertical position

☐ install safety covers on props (x3)

☐ install and connect avionics battery

☐ verify aircraft ground station is communicating with aircraft

☐ verify payload ground station is communicating with aircraft and logging data

☐ verify video camera is recording

☐ install drive batteries

☐ power RC transmitter, ensure receiver link light is on

☐ verify aircraft ground station shows disarmed state

☐ remove safety covers (x3), **staying clear of props**

☐ verify flight area is safe, free of hazards and non-essential personnel

☐ arm APM, verify ARMED status on aircraft ground station

☐ announce launch to flight personnel

☐ begin flight operations

C.2 Post-Flight Checklist

☐ disarm APM

☐ verify disarmed state on aircraft ground station

☐ power off RC transmitter

☐ install safety covers (x3), staying clear of props

☐ remove drive batteries

☐ remove avionics battery

☐ terminate video recording

☐ terminate payload logging

APPENDIX D

DATA ACQUISITION PAYLOAD CODE

```
// #include <Wire.h>
// #include <avr/pgmspace.h>

// hex1 telemetry system
// sample 7 Attopilot 90A current sensors connected to
// two MCP3208 12 bit 8 channel ADCs
// Jason Winningham
// kg4wsv@gmail.com

// Microchip ADC code from Arduino playground, modified by jdw

// interval between samples, in ms
// e.g. 250 = 4Hz sample frequency
#define DATA_WRITE_INTERVAL 250

// ADC_STEPS is total possible ADC values, e.g. 2^n_bits
// 1024 for ATmega internal ADC, 4096 for 12 bit ADC
#define ADC_STEPS 4096

// MCP3208 8 channel ADC
#define NUM_ADC_CHANNELS 8
#define MCP_MOSI 11
#define MCP_MISO  12
#define SPICLOCK 13
#define SLAVE_SELECT_ADC2 9
#define SLAVE_SELECT_ADC1 10


int read_adc(byte cs, byte channel)
{
  int adcvalue = 0;
  //command bits - start, mode, chn (3), dont care (3)
```

```
  byte commandbits = B11000000;

  //allow channel selection
  commandbits|=((channel)<<3);

  digitalWrite(cs, LOW); //Select adc
  // setup bits to be written
  for (int i=7; i>=3; i--)
  {
    digitalWrite(MCP_MOSI,commandbits&1<<i);
    //cycle clock
    digitalWrite(SPICLOCK, HIGH);
    digitalWrite(SPICLOCK, LOW);
  }

  digitalWrite(SPICLOCK, HIGH);    //ignores 2 null bits
  digitalWrite(SPICLOCK, LOW);
  digitalWrite(SPICLOCK, HIGH);
  digitalWrite(SPICLOCK, LOW);

  //read bits from adc
  for (int i=11; i>=0; i--)
  {
    adcvalue+=digitalRead(MCP_MISO)<<i;
    //cycle clock
    digitalWrite(SPICLOCK, HIGH);
    digitalWrite(SPICLOCK, LOW);
  }
  digitalWrite(cs, HIGH); //turn off device
  return adcvalue;
}


////////////////////////////////////////////////////////////////////////////


// read the specified ADC pin, convert to voltage
// value according to the attopilot 90A current/voltage sensor
// we're using a 3.3V analog reference
float adc_to_atto90_voltage(int adc_in)
{
  float adc_volts;

  adc_volts = 3.3 * (float)adc_in / (float) ADC_STEPS;
  return adc_volts / 0.06369; // 63.69mv/V , per datasheet
```

```
}

////////////////////////////////////////////////////////////////////////////

// read the specified ADC pin, convert to current
// value according to the attopilot 90A current/voltage sensor
float adc_to_atto90_current(int adc_in)
{
  float adc_volts;

  adc_volts = 3.3 * (float) adc_in / (float) ADC_STEPS;
  return adc_volts / 0.0366; // 36.6mv/A , per datasheet
}



// convert the specified ADC value to voltage
// value according to the attopilot 45A current/voltage sensor
// we're using a 3.3V analog reference
float adc_to_atto45_voltage(int adc_in)
{
  float adc_volts;

  adc_volts = 3.3 * (float) adc_in / (float) ADC_STEPS;
  return adc_volts / 0.2423; // 242.3mv/V , per datasheet
}

// convert the specified ADC value to current
// value according to the attopilot 45A current/voltage sensor
float adc_to_atto45_current(int adc_in)
{
  float adc_volts;

  adc_volts = 3.3 * (float) adc_in / (float) ADC_STEPS;
  return adc_volts / 0.0732; // 73.2mv/A , per datasheet
}


////////////////////////////////////////////////////////////////////////////

void setup()
{
  pinMode(MCP_MOSI, OUTPUT);
  pinMode(MCP_MISO, INPUT);
  pinMode(SPICLOCK, OUTPUT);
```

```
  pinMode(SLAVE_SELECT_ADC1, OUTPUT);
  pinMode(SLAVE_SELECT_ADC2, OUTPUT);

  digitalWrite(SLAVE_SELECT_ADC1, HIGH);
  digitalWrite(SLAVE_SELECT_ADC2, HIGH);
  digitalWrite(MCP_MOSI, LOW);
  digitalWrite(SPICLOCK, LOW);

  Serial.begin(57600);
  Serial1.begin(57600);

  delay(20);
  Serial.flush();
  Serial1.flush();

  delay(20);
  Serial.println("hex1 telemetry");
  Serial1.println("hex1 telemetry");
}


void loop()
{
  static long next_data_write = 0;
  byte i;
  static int adc1[NUM_ADC_CHANNELS];
  static int adc2[NUM_ADC_CHANNELS];


  if (millis() < next_data_write)
  {
    return;
  }

  for (i = 0; i < NUM_ADC_CHANNELS; i++)
  {
    adc1[i] = read_adc(SLAVE_SELECT_ADC1, i);
    adc2[i] = read_adc(SLAVE_SELECT_ADC2, i);
  }

  Serial.print(millis());
  Serial1.print(millis());
  Serial.print(',');
  Serial1.print(',');
```

```
// motor 1
Serial.print(adc_to_atto90_current(adc1[0]));
Serial1.print(adc_to_atto90_current(adc1[0]));
Serial.print(',');
Serial1.print(',');
Serial.print(adc_to_atto90_voltage(adc1[1]));
Serial1.print(adc_to_atto90_voltage(adc1[1]));
Serial.print(',');
Serial1.print(',');

// motor 2
Serial.print(adc_to_atto90_current(adc1[2]));
Serial1.print(adc_to_atto90_current(adc1[2]));
Serial.print(',');
Serial1.print(',');
Serial.print(adc_to_atto90_voltage(adc1[3]));
Serial1.print(adc_to_atto90_voltage(adc1[3]));
Serial.print(',');
Serial1.print(',');

// motor 3
Serial.print(adc_to_atto90_current(adc2[2]));
Serial1.print(adc_to_atto90_current(adc2[2]));
Serial.print(',');
Serial1.print(',');
Serial.print(adc_to_atto90_voltage(adc2[3]));
Serial1.print(adc_to_atto90_voltage(adc2[3]));
Serial.print(',');
Serial1.print(',');

// motor 4
Serial.print(adc_to_atto90_current(adc1[4]));
Serial1.print(adc_to_atto90_current(adc1[4]));
Serial.print(',');
Serial1.print(',');
Serial.print(adc_to_atto90_voltage(adc1[5]));
Serial1.print(adc_to_atto90_voltage(adc1[5]));
Serial.print(',');
Serial1.print(',');

// motor 7
Serial.print(adc_to_atto90_current(adc2[4]));
Serial1.print(adc_to_atto90_current(adc2[4]));
Serial.print(',');
Serial1.print(',');
```

```
  Serial.print(adc_to_atto90_voltage(adc2[5]));
  Serial1.print(adc_to_atto90_voltage(adc2[5]));
  Serial.print(',');
  Serial1.print(',');

  // motor 8
  Serial.print(adc_to_atto90_current(adc1[6]));
  Serial1.print(adc_to_atto90_current(adc1[6]));
  Serial.print(',');
  Serial1.print(',');
  Serial.print(adc_to_atto90_voltage(adc1[7]));
  Serial1.print(adc_to_atto90_voltage(adc1[7]));
  Serial.print(',');
  Serial1.print(',');

  // avionics power
  Serial.print(adc_to_atto45_current(adc2[0]));
  Serial1.print(adc_to_atto45_current(adc2[0]));
  Serial.print(',');
  Serial1.print(',');
  Serial.print(adc_to_atto45_voltage(adc2[1]));
  Serial1.print(adc_to_atto45_voltage(adc2[1]));

  Serial.print('\n');
  Serial1.print('\n');
  next_data_write += DATA_WRITE_INTERVAL;

}
```

APPENDIX E

ARDUPILOT MEGA PARAMETERS

This is a complete list of the ArduPilot Mega / ArduCopter configuration parameters in use during testing.

Table E.1: APM parameters

parameter	value
ACRO_P	4.5
AHRS_YAW_P	0.4
ALT_HOLD_RTL	0
AMP_PER_VOLT	27.322
AUTO_LAND	20000
AUTO_SLEW	30
AXIS_ENABLE	0
AXIS_P	0.02
BATT_CAPACITY	4000
BATT_MONITOR	4
CAM_P_DZ	0
CAM_P_G	1
CAM_P_MAX	1900
CAM_P_MIN	1100
CAM_P_REV	1
CAM_P_TRIM	1500
CAM_R_DZ	0
CAM_R_G	1
CAM_R_MAX	1900
CAM_R_MIN	1100
CAM_R_REV	1
CAM_R_TRIM	1500
CH7_OPT	7
COMPASS_AUTODEC	1
COMPASS_DEC	0
COMPASS_LEARN	0
COMPASS_OFS_X	-75.294
COMPASS_OFS_Y	-22.121
COMPASS_OFS_Z	81.319
COMPASS_USE	1
ESC	0
FLOW_ENABLE	0
FLTMODE1	5
FLTMODE2	0
FLTMODE3	0
FLTMODE4	0
FLTMODE5	0
FLTMODE6	0
FRAME	1
HLD_LAT_I	0

parameter	value
HLD_LAT_IMAX	3000
HLD_LAT_P	0.35
HLD_LON_I	0
HLD_LON_IMAX	3000
HLD_LON_P	0.35
INPUT_VOLTS	4.7
LED_MODE	0
LOG_BITMASK	382
LOG_LASTFILE	0
LOITER_LAT_D	0
LOITER_LAT_I	0.2
LOITER_LAT_IMAX	3000
LOITER_LAT_P	2
LOITER_LON_D	0
LOITER_LON_I	0.2
LOITER_LON_IMAX	3000
LOITER_LON_P	2
LOW_VOLT	9.6
MAG_ENABLE	1
NAV_LAT_D	0
NAV_LAT_I	0.2
NAV_LAT_IMAX	3000
NAV_LAT_P	3
NAV_LON_D	0
NAV_LON_I	0.2
NAV_LON_IMAX	3000
NAV_LON_P	3
OF_PIT_D	0.12
OF_PIT_I	3.2
OF_PIT_IMAX	400
OF_PIT_P	2.5
OF_RLL_D	0.12
OF_RLL_I	3.2
OF_RLL_IMAX	400
OF_RLL_P	2.5
RATE_PIT_D	0
RATE_PIT_I	0
RATE_PIT_IMAX	500
RATE_PIT_P	0.04
RATE_RLL_D	0

parameter	value
RATE_RLL_I	0
RATE_RLL_IMAX	500
RATE_RLL_P	0.04
RATE_YAW_D	0
RATE_YAW_I	0
RATE_YAW_IMAX	5000
RATE_YAW_P	0.13
RC_SPEED	490
RC1_DZ	30
RC1_MAX	1926
RC1_MIN	1128
RC1_REV	1
RC1_TRIM	1531
RC2_DZ	30
RC2_MAX	1926
RC2_MIN	1126
RC2_REV	1
RC2_TRIM	1528
RC3_DZ	30
RC3_MAX	1926
RC3_MIN	1128
RC3_REV	1
RC3_TRIM	1133
RC4_DZ	40
RC4_MAX	1924
RC4_MIN	1125
RC4_REV	1
RC4_TRIM	1519
RC5_DZ	0
RC5_MAX	1928
RC5_MIN	1128
RC5_REV	1
RC5_TRIM	1926
RC6_DZ	0
RC6_MAX	1927
RC6_MIN	1127
RC6_REV	1
RC6_TRIM	1536
RC7_DZ	0
RC7_MAX	1498

parameter	value
RC7_MIN	1497
RC7_REV	1
RC7_TRIM	1498
RC8_DZ	0
RC8_MAX	1498
RC8_MIN	1497
RC8_REV	1
RC8_TRIM	1498
RTL_LAND	1
SERIAL3_BAUD	57
SIMPLE	9
SONAR_ENABLE	1
SONAR_TYPE	0
SR0_EXT_STAT	0
SR0_EXTRA1	0
SR0_EXTRA2	0
SR0_EXTRA3	0
SR0_PARAMS	50
SR0_POSITION	0
SR0_RAW_CTRL	0
SR0_RAW_SENS	0
SR0_RC_CHAN	0
SR3_EXT_STAT	0
SR3_EXTRA1	0
SR3_EXTRA2	0
SR3_EXTRA3	0
SR3_PARAMS	0
SR3_POSITION	0
SR3_RAW_CTRL	0
SR3_RAW_SENS	0
SR3_RC_CHAN	0
STAB_D	0.15
STAB_D_S	0.5
STB_PIT_I	0.1
STB_PIT_IMAX	4000
STB_PIT_P	3.5
STB_RLL_I	0.1
STB_RLL_IMAX	4000
STB_RLL_P	3.5
STB_YAW_I	0.01

parameter	value
STB_YAW_IMAX	800
STB_YAW_P	7
SUPER_SIMPLE	0
SYSID_MYGCS	255
SYSID_SW_MREV	118
SYSID_SW_TYPE	10
SYSID_THISMAV	1
THR_ALT_I	0.015
THR_ALT_IMAX	300
THR_ALT_P	0.5
THR_FAILSAFE	0
THR_FS_ACTION	2
THR_FS_VALUE	975
THR_MAX	1000
THR_MIN	0
THR_RATE_D	0.02
THR_RATE_I	0
THR_RATE_IMAX	300
THR_RATE_P	0.25
TRIM_THROTTLE	461
TUNE	9
TUNE_HIGH	89
TUNE_LOW	10
VOLT_DIVIDER	15.701
WP_INDEX	0
WP_LOITER_RAD	10
WP_MODE	0
WP_MUST_INDEX	0
WP_RADIUS	100
WP_SPEED_MAX	600
WP_TOTAL	1
XTRK_GAIN_SC	1

REFERENCES

[1] P. Castillo R. Lozano and A. Dzul. Stabilization of a mini rotorcraft with four rotors. 2005.

[2] Scott D. Hanford, Lyle N. Long, and Joseph F. Horn. A Small Semi-Autonomous Rotary-Wing Unmanned Air Vehicle (UAV).

[3] Federal Aviation Administration. FAA Unmanned Aircraft Systems Handbook. Technical report, July 2011.

[4] Paul Pounds and Robert Mahony. Design principles of large quadrotors for practical applications. In *2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[5] J.G. Leishman. *Principles of Helicopter Aerodynamics*. Cambridge Aerospace Series. Cambridge University Press, 2006.

[6] Tommaso Bresciani. *Modelling, Identification and Control of a Quadrotor Helicopter*. Masters thesis, Lund University, October 2008.

[7] Teppo Luukkonen. Modelling and control of quadcopter, August 2011.

[8] Michael David Schmidt. *Simulation and Control of a Quadrotor Unmanned Aerial Vehicle*. Masters thesis, University of Kentucky, 2011.

[9] Andrew Neff. *Linear and Non-Linear Control of a Quadrotor UAV*. Masters thesis, Clemson University, May 2007.

[10] Menno Wierema. *Design, Implementation and Flight Test of Indoor Navigation and Control System for a Quadrotor UAV*. Masters thesis, Delft University of Technology, December 2008.

[11] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[12] P. McKerrow. Modelling the draganflyer four-rotor helicopter. In *IEEE International Conference on Robotics and Automation*, April 2004.

[13] D. Mellinger, M. Shomin, N. Michael, and V. Kumar. Cooperative grasping and transport using multiple quadrotors. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, Nov 2010.

[14] photo by Guido Brüscher. Wikipedia Commons `http://upload.wikimedia.org/wikipedia/commons/0/00/Taumel_142_b.png`. 2006.

[15] M. Cutler, N. Kemal Ure, B. Michini, and J. P. How. Comparison of fixed and variable pitch actuators for agile quadrotors. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Portland, OR, August 2011. (AIAA-2011-6406).

[16] B. Michini, J. Redding, N. K. Ure, M. Cutler, and J. P. How. Design and flight testing of an autonomous variable-pitch quadrotor. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2978 – 2979. IEEE, May 2011.

[17] M. Cutler and J. P. How. Comparison of fixed and variable pitch actuators for agile quadrotors. 2012.

[18] Max Levine. ArduCopter frame configurations `http://code.google.com/p/arducopter/wiki/AC2_Multi`. 2012.

[19] Shyam Menon, Nathan Moulton, and Christopher Cadou. Development of a dynamometer for measuring small internal-combustion engine performance. *Journal of Propulsion and Power*, 23(1), January-February 2007.

[20] George Thomas. Overview of Storage Development DOE Hydrogen Program. Review, DOE, Sandia National Laboratories, Livermore, CA, May 2000.

[21] Federal Aviation Administration. *FAA System Safety Handbook*. December 2000.

[22] U.S. Congress. H.R. 658: FAA Modernization and Reform Act of 2012.

[23] John B. Brandt and Michael S. Selig. Propeller Performance Data at Low Reynolds Numbers. In *49th AIAA Aerospace Sciences Meeting*, Orlando, FL, January 2011.

[24] eCalc. xcopterCalc for MultiCopter with fixed pitch prop `http://ecalc.ch/xcoptercalc_e.htm?ecalc`. April 2012.

[25] Gavin Ananda. UIUC Propeller Database `http://www.ae.illinois.edu/m-selig/props/propDB.html`. June 2012.

[26] MaxStream, Lindon, UT. *9XTend OEM RF Module*, January 2007.

[27] Texas Instruments, Dallas, TX. *High-Side Measurement Current Shunt Monitor*, November 2005.

[28] Microchip Technology Inc., Chandler, AZ. *2.7V 4-Channel/8-Channel 12-Bit A/D Converters with SPI Serial Interface*, 2002.

[29] Atmel Corporation, San Jose, CA. *Atmel AVR 8-Bit Microcontroller*, May 2012.

[30] Evonik Industries, Darmstadt, Germany. January 2011.

[31] 3M Adhesives Division, St. Paul, MN. *3M Scotch-Weld Epoxy Adhesives DP-460*, 1997.

[32] Umeco Structural Materials (Derby) Ltd., Heanor, Derbyshire, UK. *LTM45-1 Component Prepreg*, 2012.

[33] Henkel AG & Co., Düsseldorf, Germany. *Loctite 242*, January 2012.

[34] 3W-Modellmotoren GmbH, Rödermark, Germany. *Engine Manual, 28i / CS*, 2002.

[35] Sullivan Products, Baltimore, MD. *900 to 2700 Watt Brushless Alternator S675600*, 2011.

[36] LiveGraph. LiveGraph: the real-tme data graph plotter http://www.live-graph.org/. June 2012.

[37] J. David Irwin. *Basic Engineering Circuit Analysis*. Macmillian Publishing Company, New York, 3 edition, 1989.

[38] Lockheed Martin Corporation. *Joint Strike Fighter Air Vehicle C++ Coding Standards for the System Development and Demonstration Program*, 2005.

[39] S.Suomalainen. Open-Source Components in Safety Critical Systems. Autumn 2004.

[40] Alan Clements. *Microprocessor Ssytems Design*. PWS Publishung Company, Boston, MA, 3 edition, 1997.

[41] LDRA. *LDRA Testbed*. LDRA Technology, Southport, UK.

[42] LDRA. *LDRA TBvision Reference*. LDRA Technology, Southport, UK, 2010.

[43] Constantine A. Balanis. *Antenna Theory: Analysis and Design*. John Wiley and Sons, Inc., 2 edition, 1997.