April 2017

ACAP: The Autonomous Cargo Aircraft Project

Benjamin Kanzler Gillette
Worcester Polytechnic Institute

Erik Ilan Nadel
Worcester Polytechnic Institute

Keshuai Xu
Worcester Polytechnic Institute

Killian Ceara Henson
Worcester Polytechnic Institute

Nicholas Charles Cyganski
Worcester Polytechnic Institute

ACAP: The Autonomous Cargo Aircraft Project

A Major Qualifying Project submitted to the Faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree in Bachelor of Science in Electrical Engineering, Computer Science, Robotics Engineering, Manufacturing Engineering, and Mechanical Engineering by

Nicholas Cyganski | Benjamin Gillette | Killian Henson

Xavier Little | Erik Nadel | Zhaochen Ding

Tyler Nickerson | Keshuai Xu

Date: 4/27/17

Project Advisors:

Professor Fred Looft, Advisor

Professor Mike Ciaraldi, Co-Advisor

Abstract

Development of autonomy in fly-by-wire aircraft has long been limited to multi-million dollar systems, or large remotely operated vehicles due to regulatory restrictions and the difficulty of designing a system around existing light aircraft. This report summarises the development of an experimental, optionally piloted aircraft with the capability of fully autonomous decisionmaking and controlled flight. Particular time is spent discussing the safety features of the mechanical and electrical systems, as well as a novel control system design for light aircraft autopilots. This phase of the project included successful independant ground tests of all electromechanical systems, and simulator tests of all autoflight software. The next phase of the project will include flight testing.

Acknowledgements

Professor Fred J. Looft

We would like to sincerely acknowledge with much appreciation the crucial role of Professor Fred J. Looft, who advised our project, devoted time to listen to our progress, gave kind guidance, provided us with resources, and encouraged us to pursue our goals bravely.

Professor Michael J. Ciaraldi

The computer science team would like to thank Professor Michael Ciaraldi for advising the development of the flight computer software. His knowledge from previous projects helped us approach design problems that were new to the team and guide us to an optimal solution. We also would like to highlight his fantastic sense of humor that made MQP meetings the highlight of our day.

Joe St. Germain

The ACAP team would like to thank Joseph St. Germain for all of the advice and guidance he provided over his years as WPI's Robotics lab manager. Any time a student wound up struggling with a piece of hardware that just would not cooperate, Joe was there with a suggestion on how to approach the problem. Throughout this project, he helped to provide us with the tools and equipment we needed to conduct our initial testing.

Table of Contents

Table of Appendices

Table of Authorship

Executive Summary	All
Appendix A	Heather Cummings
Appendix B - Aircraft Integration	Killian Henson
Appendix C - Power and Electronic Control Systems	Benjamin Gillette
Appendix D - Design of the Safety Officer	Keshuai Xu
Appendix E - FMC Backend	Erik Nadel, Nicholas Cyganski
Appendix F - Software Frontend	Tyler Nickerson, Nicholas Cyganski
Appendix G - Computer Vision	Nicholas Bradford
Appendix H - TCAS	Zhaochen Ding

1 Introduction

1.1 Forward

The introduction section of this report serves as a unifying summary of the work completed on the project and its motivation. The accompanying appendices to follow the introduction are in-depth explanations of design methodology, results, and regulatory requirements for each of the disciplines involved; aviation, electrical engineering, computer science, and mechanical engineering.

1.2 Background

The Autonomous Cargo Aircraft Project (ACAP) is a student project effort to configure and demonstrate the world's first gate-to-gate autonomous civilian aircraft, including start-up through shutdown and all phases between, including ground operations and enroute navigation. The outcome will demonstrate the viability of autonomous aircraft operations within civilian airspace between 400' and FL180 without modification of existing airspace architecture beyond the implementation of FAA NextGen principles before 2020.

Currently only about 100 airports are used for mass cargo delivery in the United States. Commercialization of autonomous light aircraft technology could expand the system to over 8000 US airports and reduce the size of aircraft and cargo capacity for which delivery by air is economical. Simplifying the freight forwarding system that has been unaltered for 200 years would also decrease cargo delivery time and cost by factors of 5 and 4 respectively, allowing cargo delivery to less developed areas; national and international, eliminating risk to pilots' lives in dangerous operations such as disaster response or humanitarian efforts.

Development risk will be minimized by retrofit of a FAA certified aircraft designed by Piper Aircraft Inc, and Supplementary Type Certificate (STC) components. Autonomous operation tests will be conducted with a crew of two (pilot and flight engineer) to address FAA restrictions and mitigate risk. Gate-to-gate, Visual Flight Rules operation will employ only GPS, inertial reference, and computer vision technologies for real-time navigation.

1.3 Industry Background

Over an eighteen month period, real-time simulation (part of a separate project) has proven the economic and logistic viability of augmenting existing freight forwarding services in the contiguous United States with a fleet of small autonomous aircraft ranging in wingspan from 30ft to 80ft. Freight forwarding refers to the intermediate steps in the cargo delivery chain from origin to long distance carrier, and from the carrier to the parcel's final destination. In today's market, freight bundlers such as FedEx, UPS, and DHL attempt to complete their vertical integration, but the majority of shipping in the U.S. is still dependant on freight forwarding by rail and truck after arriving at intermodal terminals or airports, as pictured under "traditional freight forwarding" below. Considered the missing link to the infamous Amazon drone delivery, the ACAP method aims to simplify this.

Figure 1: "The ACAP Method" Simplified Shipping with Autonomy"

There are nearly 20,000 airports in the U.S., meaning that suburban residents are typically no further than a 20 minute drive away from their nearest airport. The majority of aircraft in our existing cargo fleet are restricted to only the 50 largest airports in the U.S. due to their size. A fleet of small autonomous aircraft have been shown in simulation to decrease freight forwarding times by a factor of four, and costs by a factor of five. Although beyond the scope of this simulation, the losses associated with palletizing cargo for shipment by ground transportation, and road traffic congestion near larger airports have also been theorized to reduce overall shipping costs over traditional methods. At current ground shipping profit margins, a single aircraft of this size is expected to pay for the cost of its conversion within ~600 hours of operation as a cargo carrying aircraft, or ~120 days if the aircraft is flying 5hrs/day.

1.4 Project Description

The development of autonomous aircraft systems have been limited to low-cost remote controlled aircraft or multi-million dollar commercial platforms since their inception. This Major Qualifying Project aims to outfit an existing four seat general aviation aircraft with all the systems necessary to fly as an autonomous aerial vehicle for the purpose of cargo transportation. The product of this MQP will be a first-of-its-kind unmanned aerial vehicle (UAV) capable of unassisted gate-to-gate travel, including taxi, takeoff, navigation, landing, and parking. Many of the systems and practices developed for this project have never been practically implemented, and can serve as proof of concept for future autonomous aircraft design.

The project began with the purchase of a Piper PA-28-140 Cherokee aircraft (see section 2.1). The interior of the aircraft was removed with the exception of the two front seats, and flight controls, thus revealing all essential aircraft systems. Electrically driven actuators were fixed to the airframe to control all mechanical aircraft subsystems. Sensors to facilitate the autonomous operation of the aircraft were mounted to the fuselage, including a pitot-static system, magnetometer, inertial reference unit, GPS receiver, ground proximity LIDAR, and various engine monitoring systems. The electrical subsystems of the aircraft, such as lights, fuel pump, and starter, were controlled by the onboard computer mounted in the existing avionics bay of the aircraft with a touchscreen interface for the flight engineer in the co-pilot seat. All actuators and electrical systems with control of the aircraft were outfit with software failsafes and physical overrides and breakaways. The pilot in command (PIC), seated in the aircraft, will always be able to activate critical aircraft systems or take manual control of the aircraft in the event that the computer system does not perform as intended.

During the planning and completion of all mechanical and electrical alterations, the team sought guidance of a Designated Airworthiness Representative, with the intent to seek experimental certification of the aircraft from the Federal Aviation Administration under CFR §21.191(a) Experimental Certificates: Research and Development, which is defined as "Testing new aircraft design concepts, new aircraft equipment, new aircraft installations, new aircraft operating techniques, or new uses for aircraft." The certification of the aircraft would have allowed aerial testing, in addition to the ground based testing, which requires no certification.

2 Aircraft Platform

2.1 Certified Aircraft

There exist a number of examples in recent history of both certified aircraft and purposebuilt research models being employed in the development of civilian autonomous aircraft technology. The noticeable trend among these is that purposebuilt models exist between gross takeoff weights (MTOW) of zero to 200 pounds, with one of the largest being the Area-I PTERA, a jet turbine powered scale model of a Boeing 737-800. Above the 200 pound mark, where one might consider the test platform to be a "real" aircraft, rather than a hobbyist drone, FAA certified aircraft lead the way. Most notably, some of the largest aircraft used to test unmanned systems include BAE System's Jetstream 31, NASA Dryden's Sabreliner 65, and Piaggio Aerospace's P.180 Avanti, with wingspans around 50ft and 15,000lbs MTOW.

There are many advantages of using certified aircraft at larger weight categories, but the most influential to this project are as follows:

- The airframe and powerplant are already certified to carry human flight crews.
- The flight characteristics are well understood and numerically documented.
- Only one additional FAA certification is required to fly in United States airspace.
- The public's image of certified aircraft is welcoming rather than objectionable.

The aircraft platform chosen for the project was a 1972 Piper PA-28 Cherokee 140. The Cherokee was chosen, as it is one of the best selling light aircraft of all time, making maintenance resources easy to locate and low cost. Given the monetary resources available to this project through private, academic, and government funding, the project's total budget was estimated at $40,000, with half of that being allocated for aircraft acquisition. The aircraft chosen was purchased for $17,300, including a $550 pre-purchase inspection, from the Perception Prime Flight School in Beverly Massachusetts. The aircraft was engaged in flight training until the purchase, as such, the aircraft was receiving scheduled 100hr maintenance and kept to the highest operational standards after its arrival at the school in 2012. Being that the aircraft was in annual, it was flown to Worcester Regional Airport, where it would be kept in a quonset style hangar, managed by the airport's fixed base operator (FBO), Rectrix Aerodrome Centers Inc.

As the aircraft is going to remain certified as its original model and category for the duration of the project, all original specifications and flight characteristics will remain unaltered, as they appear below.

N15726 (Figure 2)

1972 Piper PA-28 Cherokee 140 "Cruiser"

Powerplant: 2,000hr TBO 150hp Lycoming O-360-A4A

Propeller: Sensenich fixed-pitch, 2-blade, 76" dia.

Length: 23'6" Height: 7' 4" Wingspan: 30'

Wing Area: 160ft^2

4-Seat Empty weight: 1,375 lb

Gross Weight: 2,400 lb

Fuel capacity, std 50 gal (48 gal usable)

Cruise Power, Best Economy: 124 kts

Stall Speed: 48 kts (dirty) 58 kts (clean)

Climb Performance: 750 fpm

Ceiling: 13,000 ft Range: 455nm

2.2 Limitations & Regulations

As a certified, privately owned aircraft, N15726 must be operated in accordance with the Federal Aviation Regulations (FAR) Title 14: Aeronautics and Space (CFR) §91 "General Operating and Flight Rules" for the time being. In general, this means that the aircraft will not deviate from its original flight envelope, airframe and powerplant operating characteristics, and documented limitations in the Pilot's Operating Handbook (POH), and posted placards within the aircraft.

Upon retrofit of the aircraft beyond the modifications and maintenance provided for by FAR §91, the aircraft will then specifically fall into §91.319 "Aircraft having experimental certificates: Operating limitations," stating the following:

"Each person operating a provisionally certificated civil aircraft shall operate within the prescribed limitations displayed in the aircraft or set forth in the provisional aircraft flight manual or other appropriate document. However, when operating in direct conjunction with the type or supplemental type

certification of the aircraft, that person shall operate under the experimental aircraft limitations of §21.191 of this chapter and when flight testing, shall operate under the requirements of §91.305 of this part."

Experimental certification of an aircraft is never a guaranteed process, nor is it recommended for for most applications where a Supplemental Type Certificate (STC) might be obtained to allow for the installation of a specific part or modification to the aircraft. For larger scale systems, such as autopilots, that require testing while in flight before certification, there exists specific provisions for testing in FAR §21, which also encompasses homebuilt and kit aircraft under advisory of the Experimental Aircraft Association (EAA).

2.3 Modifications & Re-Certification

In order to operate an aircraft in United States Airspace with modifications or additional features beyond those certified by the manufacturers original airworthiness certificate, a new airworthiness certificate, or STC must be obtained for structural components, and adhere to FAA Technical Standard Order (TSO) for avionics and aircraft instruments. In the case of Unmanned Aerial Systems (UAS), or in this case specifically, an Optionally Piloted Aircraft (OPA), the FAA allows for several different certification options, depending on the desired operations under the certification. They are as follows:

"A civil UAS cannot be operated in air commerce in the National Airspace System unless there is an appropriate and valid airworthiness certificate issued for that UAS. U.S. registration is a prerequisite for the issuance of an airworthiness certificate.

Eligible individuals may apply for the following:

- *21.17b type certificate for special class aircraft and a 21.183 standard airworthiness certificate for special class aircraft*
- *21.25 type certificate for restricted category aircraft and a 21.185 special airworthiness certificate in the restricted category*
- *21.191 special airworthiness certificate in the experimental category for the purposes of research and development, crew training, and market survey*
- *21.197 special flight permit for the purpose of production flight testing new aircraft"*

This project will be seeking certification under §21.191(a):

"Research and development. Testing new aircraft design concepts, new aircraft equipment, new aircraft installations, new aircraft operating techniques, or new uses for aircraft."

Experimental airworthiness certificates are issued by Designated Airworthiness Representatives (DAR) appointed by the FAA to examine aircraft, award airworthiness and assign operating limitations, such as limiting homebuilt aircraft to a 25nm radius of their home airport until 40hrs of single pilot testing has been completed. Although there is a checklist of items provided by the FAA for a DAR to take into consideration when examining an aircraft, the final decision is solely in the hands of the DAR. The process is described in detail in Advisory Circular (AC) 20-27G "Certification and Operation of Amateur-Built Aircraft". Figure 9 from AC 20-27G is shown below to outline the process.

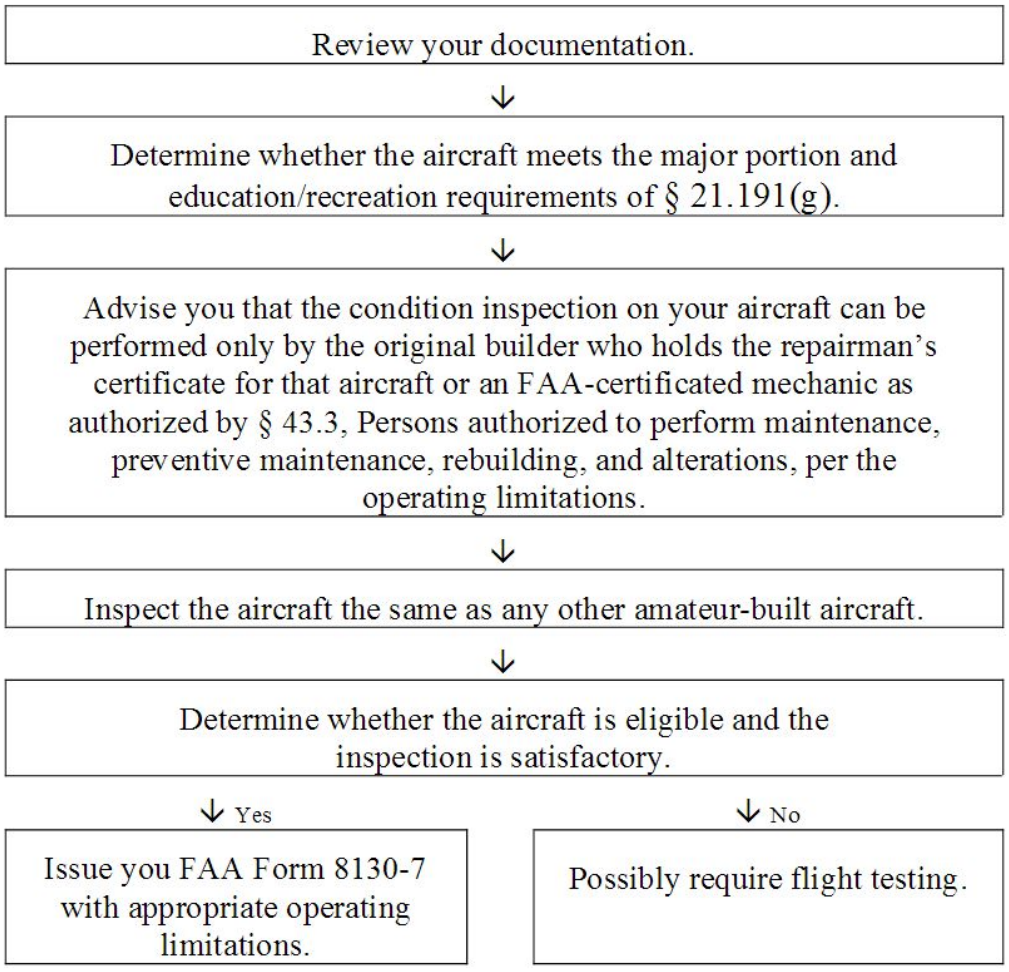


Figure 3: Experimental Aircraft Certification Flow Chart

Upon issuance of Form 8130-7, the DAR will assign limitations for Phase-1 flight testing of the aircraft. Phase-1 flight testing of a previously certified aircraft requires a minimum duration of controllability testing. Phase-1 testing in general is described as:

"Fly[ing] the aircraft over water or sparsely populated areas with light air traffic in accordance with §91.305, Flight test areas, appropriate for the applicant to show the aircraft is controllable throughout its normal range of speeds and maneuvers and that the aircraft has no hazardous operating characteristics or design features."

When the prescribed flight testing duration with a minimum certified flight crew of one, for a Piper Cherokee, is complete, the Pilot in Command (PIC) will *"endorse the aircraft logbook and maintenance records with a statement that [they] have complied with the requirements of §91.319(b)."* The aircraft may then continue to fly under any the Phase-2 limitations established by the DAR under the command of any qualified flight crew in that model of aircraft. For the purposes of research and development, N15726 is certified for flight testing of UAS with a minimum flight crew of two; one PIC controlling conventional aircraft systems and ensuring safe flight, and one flight engineer, who is intimately knowledgeable of the modifications and new systems being tested, as well as basic aviation knowledge applicable to FAR *§91 operations.*

Since a large part of the certification process is convincing the DAR of the safety of the aircraft systems and modifications made, modification of original aircraft systems should be kept to an absolute minimum, especially those concerning the following.

- the flow of air over the aircraft's skin or control surfaces
- the structural integrity of the original airframe
- the required flight equipment for Visual Flight Rules (VFR), as defined in FAR §91.205

Measures taken to minimize these changes will be discussed throughout this document, but it is important to note that the resale of the aircraft after the testing period requires its recertification back to its original airworthiness certificate. Any changes made need to be rolled back, and modified parts replaced. In some instances, this has the potential to cost as much as the purchase of the aircraft, as the maintenance must be completed under the supervision of an Airframe and Powerplant mechanic (A&P). Documentation of all these alterations are not only required for resale, but also make for a much more convincing case for the experimental airworthiness examination.

2.4 Documentation

Under traditional FAR §91 rules, there is a limited set of physical alterations that an aircraft owner or operator may make without the appropriate mechanic's certification, such as changing tires, adding or removing interior aesthetic elements, or performing an oil change, since none of these would alter the performance of the aircraft. In some cases, the proper entry must still be added to the aircraft's logbooks, of which there are typically three; airframe, engine, and propeller. Since the installation and removal of equipment in this project will take place solely on the interior of the aircraft, a note was placed on the current page of the airframe log, indicating the all entries from the date of most recent purchase forward will be recorded in a separate binder, also containing the technical specifications of all equipment installed by the project team.

The documentation included in this binder, which is to be on board the aircraft at all times of operation, include the following sections:

1. Continued Airframe Logbook
2. Experimental Mechanical Schematics
3. Experimental Electrical Schematics
4. Experimental Avionics & Sensors
5. Experimental Weight & Balance
6. Experimental Operating Limitations
7. Detailed Modifications Log

Until the time of the experimental certification, no additional entries will be recorded in the Continued Airframe Logbook section, only in the Detailed Modifications Log, for review by the Designated Airworthiness Representative. The remaining sections will be up to date with any equipment currently installed in the aircraft at the time of operation, and written with the methodology that a knowledgeable individual in all the major technical fields would be able to recreate the system in an identical aircraft; and therefore, be able to construct a complete picture of the changes that have been made to this aircraft. The Experimental Operating Limitations section will be considered a supplement to FAA Form 8130-7 and FAR §21, organized by the flight engineers based on, and reviewed by, the engineers of each system installed in the aircraft. All flight crew members are expected to have new operating limitations committed to memory, but it will be considered the duty of the flight engineer, for

the purposes of cockpit resource management, to ensure that the aircraft is properly configured to remain within these limitations during the immediately expected phases of flight.

In a separate document, checklists will be prepared in collaboration with all engineering team leaders that encompass every aspect of system operation available to the flight crew from inspecting the system prior to engine start, through the securing of the aircraft. These checklists will not be a supplement to current checklists, rather they will be a completely new set of checklists, also including all items necessary for the safe operation of the original certified aircraft. These checklists are as follows:

1. Preflight Inspection (Outside)
2. Preflight Inspection (Inside)
3. Before Engine Start
4. Engine Start
5. Calibration & Equipment Status
6. Before Taxi
7. Run-Up & System Checks
8. Before Take-off
9. Cruise
10. Engaging Autoflight
11. Disengaging Autoflight
12. Before Descent
13. Before Landing
14. After Landing
15. Securing Aircraft

Emergency checklists will also be included, written with the same methodology for each symptom of the expected emergencies, both of the original aircraft, and of the experimental systems. Each will be depicted with an easy to recognize symbol, or combination of symbols, and are as follows:

1. Fire During Engine Start
2. Engine Failure In Flight
3. Engine Fire In Flight
4. Wing Fire In Flight
5. Electrical Failure In Flight
6. Electrical Fire In Flight (Rear)

7. Electrical Fire In Flight (Forward)
8. Autopilot Will Not Disengage
9. Pitot-Static Failure
10. Smoke In The Cockpit
11. Carbon Monoxide Detected
12. Emergency Descent
13. Communications Failure

Troubleshooting checklists will be provided by each engineering team for their respective systems. Troubleshooting checklists are not intended to be used in an emergency situation, but in event that a system is not behaving as expected, and troubleshooting is necessary to complete mission objectives, or return the aircraft safely to the ground.

2.5 Design Requirements

In order to satisfy the inspection requirements described above, no flight control mechanism may be irreversibly modified, and no non-critical system may be modified to the extent that it alters the performance of any existing aircraft gauges, sensors, controls, or avionics. In general, this can be interpreted as the attachment of new systems, physically, and electrically, must be done in a way that allows the original system to operate in exactly the same way as it would without the presence of the new system. For example, attaching an autopilot actuator to the existing T-Bar must be done in a way that does not require altering the T-Bar, or any other components, in any way other than replacing them with an identical duplicate. An acceptable method in this case would be a custom clamp around the intersection of the T-Bar, supporting a mechanism that does not interfere with the operation of the original controls in any way. Electrically, this can be interpreted as the addition of systems that sense the current status of another system, or that alter that status in a way that does not prevent the critical operation of the system in the case of an emergency. Additionally, the pitot-static and vacuum systems can be modified, using approved parts for the original aircraft, so long as the experimental equipment does not modify the performance of their respective systems and gauge readings available to the flight crew.

Since safety is of the highest priority throughout the design of these new aircraft systems, each added system must have a minimum of two methods by which the flight crew can disable it operation at any time, and return the original aircraft system to its uninhibited operation, as stated above. These two methods must be of fundamentally different types, such as pushing a button to disable the autoflight

mechanisms, and physically pulling a pin to release control of the aircraft. Tried and tested techniques to enhance safety, and features associated with safety, will be implemented for each of these instances. As an example, a master electrical cut-off switch handle would be available to the flight crew, and within reach at all times. Emergency equipment will also be available as a last resort during system failure, such as fire extinguishers, and cable cutters.

While still maintaining all safety aspects of the project, a basic set of operating parameters must be obtained by the collective systems, consisting of mechanical equipment, electrical equipment, and computer software. Systems were designed with the methodology that mechanical expectations, such as control surface actuation speed, resolution, and force, would drive the electrical system requirements, and those would in-turn, drive the software interface requirements. The first, and most basic of these requirements is defined by FAA document 14CFR §25.143(d), where the maximum control surface forces are defined for modifications of existing aircraft, and design of new aircraft. It is notable that these forces are for single-pilot certified aircraft only, and defined by the primary point of biomechanical contact with the control; therefore, the position in the control surface actuation system to which an actuator is attached makes an enormous difference to the forces required. As an example, the mechanical advantage provided by the bell-crank on the end of the flap handle in a PA-28 type aircraft is almost 20:1, meaning that actuating the flap handle at the same point as a human might only require 50lbs of force, while actuating the control cable directly might require as much as 1000lbs of force. The requirements for the primary flight controls for short and long durations as per the FAA are shown below, as the most basic of guidelines for mechanical design. No guidance is provided by the FAA on the exact durations, or how to interpret “short” or “long” term applications of force, so it has been left to the team to define them, based on pilot experience, as being more or less than five seconds in duration.

Long Term Application
- Pitch: 10lbs
- Roll: 5lbs
- Yaw: 20lbs

Short Term Application
- Pitch: 50lbs
- Roll: 25lbs
- Yaw: 150lbs

3.1 Mechanical Engineering

Mechanical Systems were added to the plane to control a variety of systems including the control surfaces. The three primary control surfaces are the rudder, stabilator, and ailerons which control the yaw, pitch, and roll respectively. Each control surface has it's own motor and gear box specifically chosen to fit the requirements of that system. Each mechanism includes an electromechanically actuated clutch system, which can be disengaged manually, as well as a last resort pull pin to disable each system. Figure 4 shows a base cockpit model before additional systems were added.

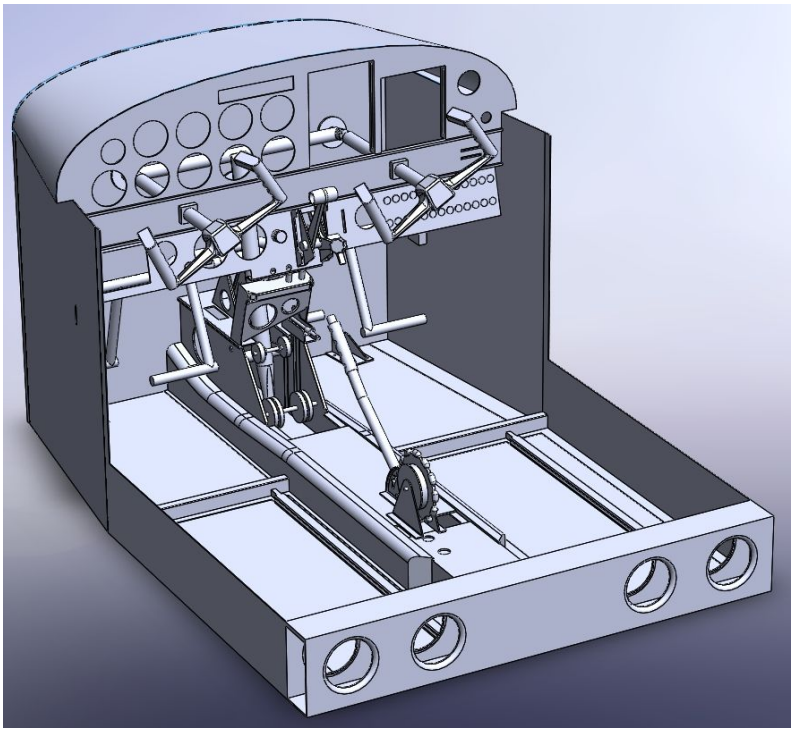


Figure 4: Aircraft Cockpit Model

The aviation industry has a long history of making new technology work with old, which is why we still have sixty year old airplanes in the sky today. This project continues the tradition, as we retrofit a Piper Cherokee 140 to be cable of autonomous flight. Mechanically speaking, this project poses many interesting challenges. There are numerous requirements that all need to be met with numerous restrictions. These included; meeting force and speed requirements, minimizing modifications to the aircraft, incorporating safety features, easy of manufacturing and size limitations. For this project the team had access to basic manufacturing equipment, such as CNC machines, Lathes, 3D printers and a laser cutter, which allowed us to design and create our own parts. Overall it was important to keep in

mind the limitations of the machine shop and machines we had access to so that we could design systems that were actually capable of being made.

The pilot controls the aircraft's pitch by moving the yoke forward and backwards, which rotates the T-bar mechanism that is connected to a series of pulleys and cables that controls the movement of the stabilator. This mechanism is can be seen in figure 5, below.

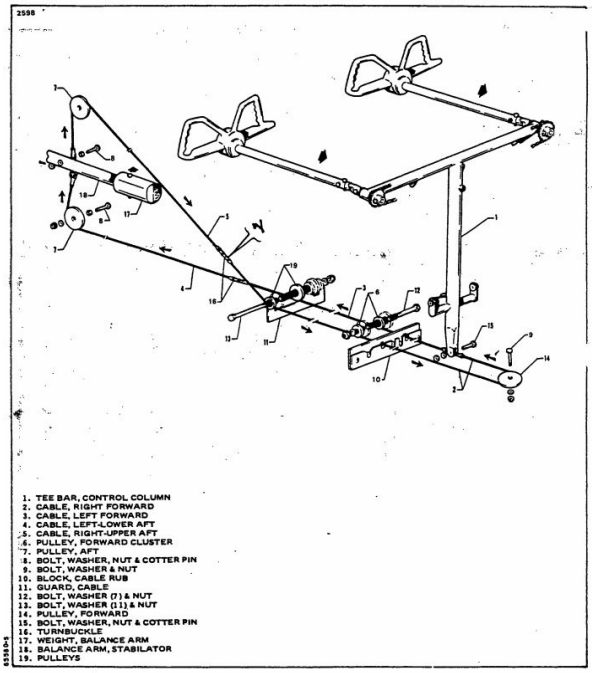


Figure 5: Original Pitch Control Cables

A VEX bag motor with a 10:1 versaplanetary gearbox, driving a belt and tensioner system to allow for the eccentric motion of the T-Bar. This is then mounted onto a custom gearbox with a 1.458:1 reduction and a custom dog gear clutch system. The pitch mechanism is theoretically capable of moving the stabilator from one extreme to the other in about 0.15 seconds while applying 50lbs of force at the yoke handles.

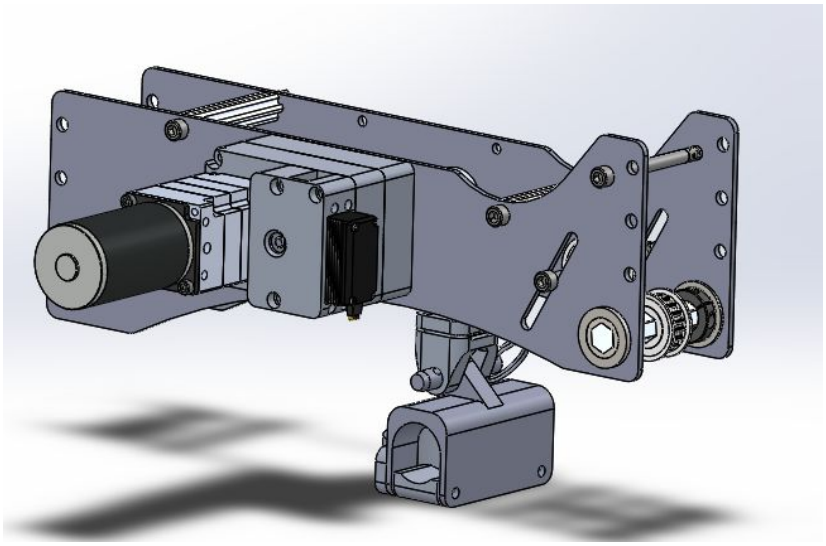


Figure 6: Pitch Control Mechanism

The roll of the aircraft is controlled via a cable and chain system that maneuvers its way around the T-Bar. A double sprocket was added to the back of the original co-pilot side yoke, and controlled via another Bag motor, versaplanetary gearbox, clutch, and sprocket. At the yoke handles, the system pictured below produces a maximum of 25 lbs of force, and is theoretically capable of actuating the ailerons from one end to the other in about 0.15 seconds.

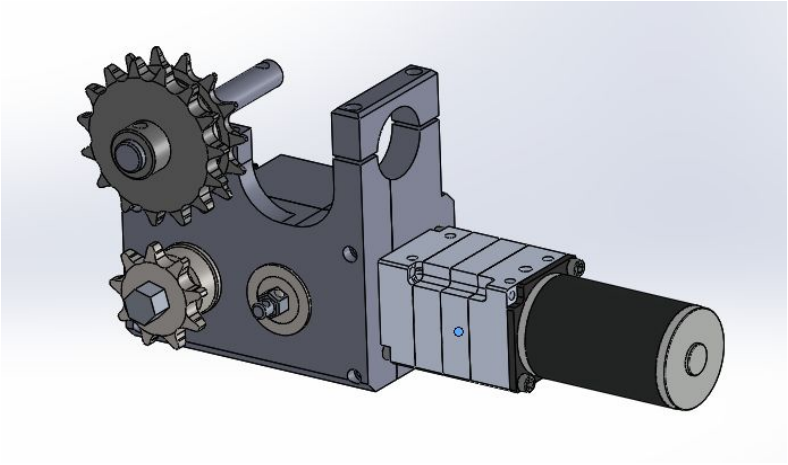


Figure 7: Roll Control Mechanism

Lastly, the yaw of the aircraft is controlled via rudder pedals, driving cables once again. Our mechanism connects a sprocket to the top of pedal assembly and rotates the mechanism forward and back

similarly to how a pilot's feet would be used to accomplish the same task. This system fits behind the pedals and connects to the floor of the airplane through one of the engine mounts. The mechanism produces 1200 in-lbs of torque, or approximately one moose-inch of torque (1170in-lbs). The system uses a VEX bag motor and a 70:1 versaplanetary gearbox. The next stage is a custom gearbox with a 400:1 ratio, removable shaft, and a clutch mechanism. The total actuation time is approximately 0.3 seconds.

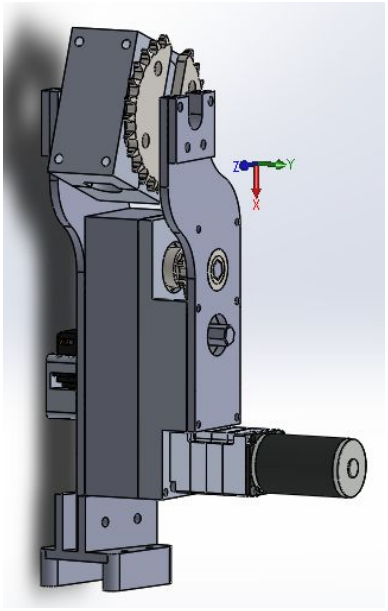


Figure 8: Yaw Control Mechanism

3.2 Computer Science

Below is a diagram for the system architecture that controls the aircraft. Starting from the flight engineer, inputs are received through the flight management computer (FMC) and sent to the FMC_IO which interprets data from the FMC into movements for the aircraft controls . Using these components, the flight engineer has fly-by-wire control of the aircraft. For flight without input from the flight engineer, the flight management computer needs to obtain context from the environment. This is done through communication via the message proxy. Through the message proxy, the FMC_IO can report aircraft control positions to the FMC. Other auxiliary systems that communicate with the FMC provide vision and environmental context to the aircraft.

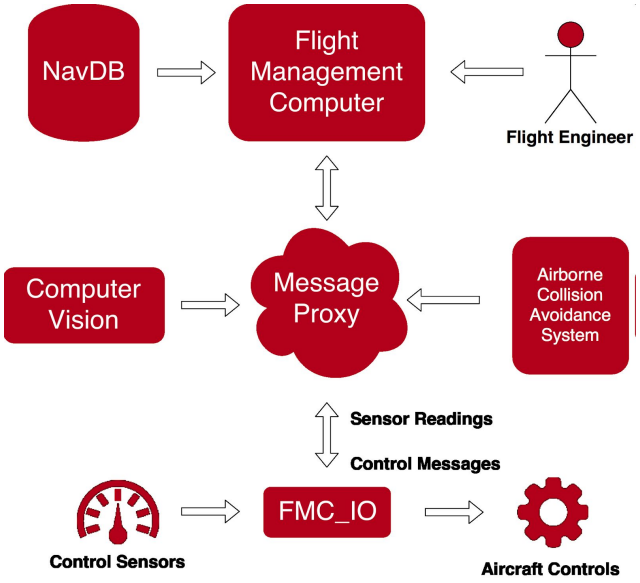


Figure 9: Software Architecture

The FMC acts as a central hub for the aircraft. It is also the control center for the flight engineer. From the FMC, the flight engineer can see where the aircraft is, the positions of the physical controls, and directly control the aircraft. The other role of the FMC is to provide navigation and context. The role of a navigation system is fulfilled by generating waypoints from the aircraft's origin to its destination. This includes ground navigation along taxiway as well as air navigation. For context, the FMC uses information from attached auxiliary systems. This includes estimated pitch and bank from the computer vision systems, taxiway lane information, and locations of nearby aircraft from the TCAS. The context can be used to assist in following navigation waypoints created by the FMC. Tied together, the flight engineer has a complete picture of the aircraft and its state.

The FMC sends/receives incoming information to/from auxiliary systems and manages control scheduling, audio cues, spatial sensors, and the state machine. Sending and receiving information is done through the message proxy. A diagram of how these all work together is presented in Figure 9. This information can come from the computer vision system, the ACAS, FMC_IO, and any additional system added in the future. The NavDB seen in the top left of Figure 9 provides airport information and is the only auxiliary system that does not use the message proxy. The FMC_IO, located at the bottom of Figure 9, interprets serialized control messages into physical movements and sends back the sensed position of a physical device. The FMC uses the state machine to decide which routines to execute throughout the phases of flight. Each state in the state machine is tied to a set of routines that are loaded upon entering the affiliated state. These routines tell the aircraft what to do given specific conditions or to perform

general tasks throughout the flight. Following a modular design like the frontend, routines allow custom actions to be added to the backend with minor changes to the code base.

The flight engineer interacts with the FMC through an interface on the right hand side of the monitor (see Appendix F). The interface is based on existing flight displays found in Boeing aircraft and utilizes vector graphics to support high-quality rendering at a variety of resolutions. The frontend also features an intuitive debugging panel that gives the flight engineer full control of all mechanical assets including real-time sensor readings of all devices. Along with debugging, the flight engineer can see the tasks that will be executed by the FMC during flight, including their progress. With a modular design, any additional information required can be added with minor changes to the code base. The integrated navigation display, and glass panel display are shown below.

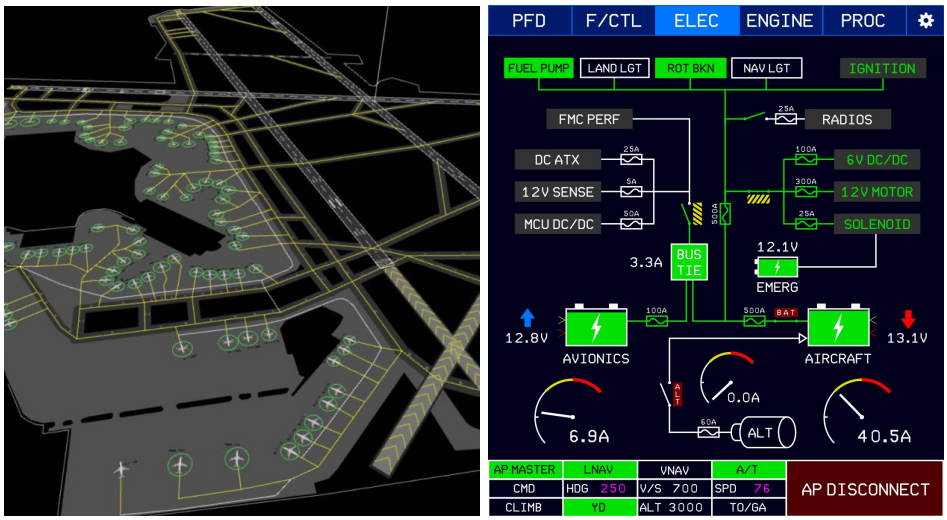


Figure 10&11: Combined Navigation and Engine Monitoring Displays

Lastly, though autopilots exist for small and large aircraft, none are able to maintain constant control of an airframe during all phases of flight, especially as close to the edges of the performance envelope, as is required by light aircraft in the air, while autonomous ground navigation presents another challenge altogether. A Linear Quadratic Gaussian control loop, generated with the MATLAB system identification toolbox, using data recorded from the aircraft estimates the required control inputs to achieve a desired flight path derived from several different parameters of flight, including vertical speed, airspeed, and bank angle. An example simulation of both lateral navigation, and vertical navigation combined is shown below, as the aircraft plans a path and flies from its starting location and orientation to follow a defined path.

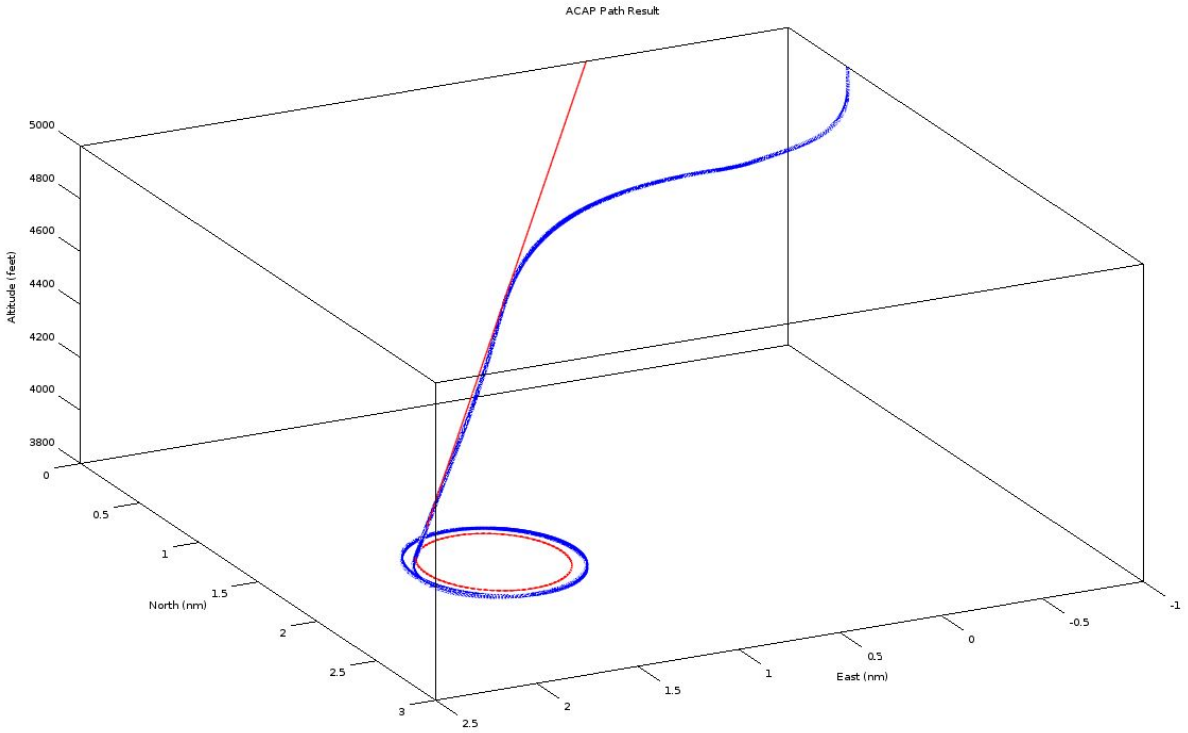


Figure 12: Controlled Descent into Hold Simulation

3.3 Electrical Engineering

These motor controllers receive commands from the Flight Management Computer (FMC) pictured below in figure 13, a custom-built desktop computer designed to provide maximum computational power at maximum efficiency. Careful budgeting of electrical power allows all of the electrical systems to operate on the power provided by the aircraft alternator, while budgeting enough power on the average flight to still charge the aircraft battery for the next starting. By powering systems directly from the alternator rather than a separate bank of batteries, the aircraft's range is not limited by the need to recharge, as long as the plane has fuel, the system can keep working. Throughout a flight, the FMC has access to several sensors to obtain any necessary about the aircraft's state, including GPS location, airspeed, engine conditions, altitude, and environmental variables, such as humidity and temperature. Communication with these sensors and operation of the various aircraft controls are coordinated by a network of ten microcontrollers, ranging from dedicated motor control units to general-purpose I/O units and system-on-chip devices that provide high-speed I/O, the majority of which are located in the forward avionics bay, pictured in figure 14.

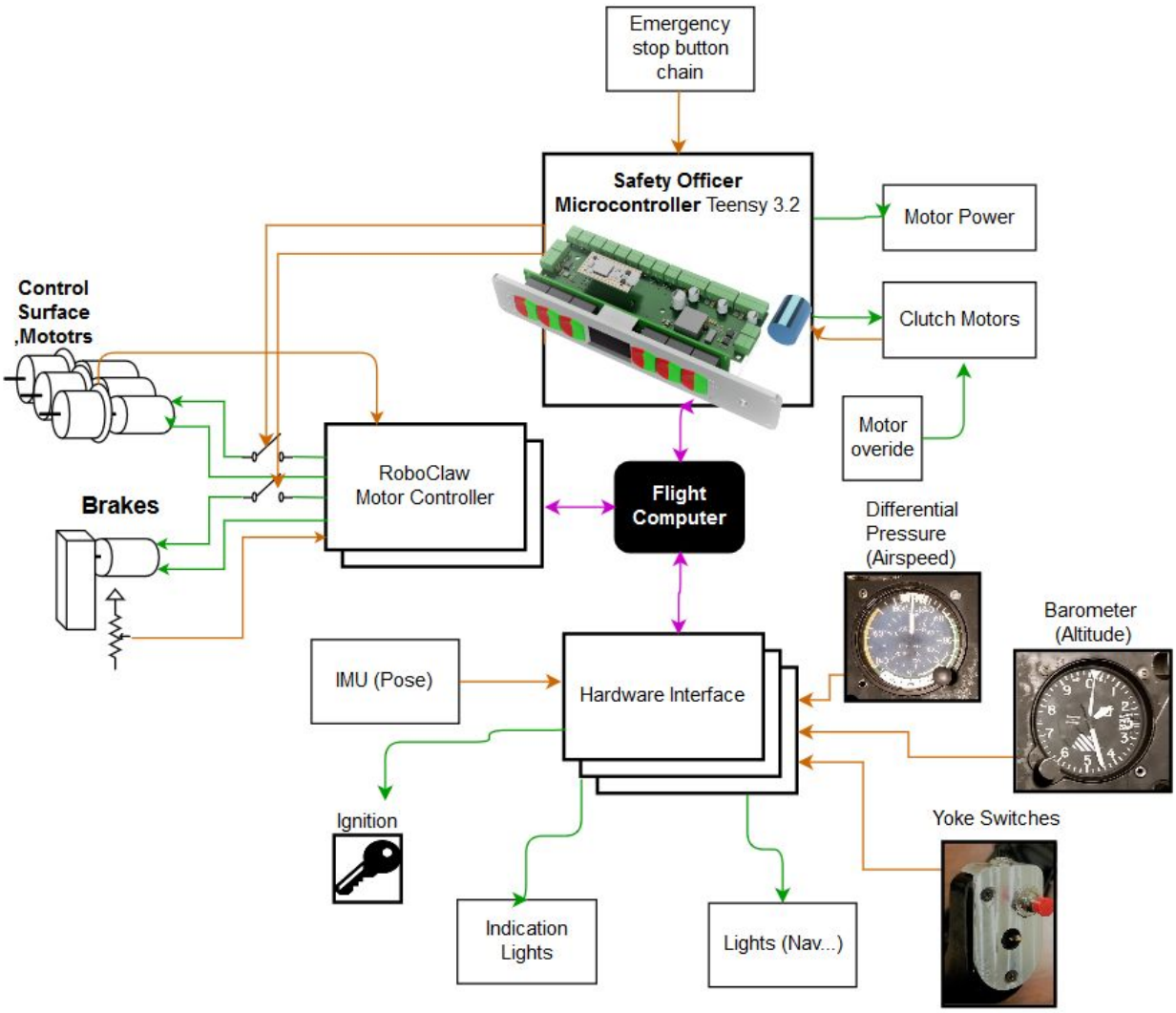


Figure 13: Electrical System Architecture

Figure 14: Forward Avionics Bay

One of these microcontrollers, mounted in the front panel, acts as a "Safety Officer," monitoring the most important parts of a network of safety features integrated into the aircraft, such as current sensors, thermistors, and manual disconnects. The Safety Officer directly controls the actuation of three mechanical clutches, which allow the pilot to engage and disengage the main actuators at any time with the press of a button on the yoke. In addition, in-line relays allow the pilot to cut motor power at any time, and an easily accessible breaker box provides access to a master power kill switch. As a last line of defense, a clutch override system allows the pilot to manually disengage the autopilot even in the event of a computer failure. Every circuit in the aircraft runs through a circuit breaker to reduce the risk of an electrical fire. This combination of overlapping safety systems and accurate control hardware combine to create a reliable, practical, and effective aircraft control system.

ACAP

AUTONOMOUS CARGO AIRCRAFT PROJECT

EXPERIMENTAL PA-28-140 OWNER'S HANDBOOK

Note:

The Information provided in this booklet has been edited from a Piper 1973 Cherokee Cruiser Owner's Handbook. It is not to be used for commercial purposes and has been created as an example handbook for the experimental aircraft for autonomous flight made from a Piper Cherokee, PA-28-140, with the tail number N15726 for the Autonomous Cargo Aircraft Project at the Worcester Polytechnic Institute. I do not claim rights not authorship to information provided for the manual flight specifications, designs, controls, manual procedures, or any information that can be directly quoted from the Piper 1973 Cherokee Cruiser Owner's Handbook.

This manual is not to be used as a guide for flight instruction. The main purpose of this handbook is to provide information on the added autonomous systems, their controls, additional safety procedures for use of the autonomous controls, and updated aircraft specifications due to these additions.

Section I

Specifications

Performance

Weights

PowerPlant

Fuel and Oil

Baggage

Dimensions

Landing Gear

Added Components and Their Weights and Datum

Removed Components and Their Weights

Section 1
Specifications

Performance

The Following Performance figures apply to the specified experimental Cherokee flown at a gross weight of 2150 pounds under standard conditions at sea level, or stated altitude. Any deviation from standard equipment may result in changes in performance and shall be discussed in the figures.

Gross Weight

Take-off (ft)(flaps up)	
Take-off distance over 50-ft obstacle (ft)(flaps up)	
Best Rate of Climb Speed (mph)	89
Service Ceiling (ft)	10,950
Absolute Ceiling	13,000
Top Speed (mph)	139 (142*)
Cruising Speed (75% Power, sea level)(mph)	121 (124*)
Optimum Cruising Speed (75% power, 7,000 ft, mph)	132 (135*)
Instructional Power Cruising Speed (60% Power, Sea level)	108 (110*)
Fuel Consumption (gal per hour 75%)	8.4
Fuel Consumption (gal per hour 50%)	5.6

*** with wheel fairings**

Cruising Range (75% power, sea level, mi)	495 (710**)
Cruising Range (75% Power, 7,000 ft, mi)	540 (780**)
Optimum Cruising Range (55% Power, 10,000 ft)	670 (930**)
Instructional Power Cruising Range (60% Power Sea Level)	575 (800**)
Stalling Speed (flaps down, mph)	55
Landing Roll (flaps down, ft)	535*

*** this value applies only for the conditions indicated on the landing distance versus density altitude chart**
**** With Reserve Fuel (50 gal)**

Weights

Gross Weight (lbs)	2150
Empty Weight (standard)(lbs)	1490
USEFUL LOAD (standard)(lbs)	660

Power Plant

Engine - Lycoming	O-320-E3D
Rated Horsepower and Speed (rpm)	150 at 2700
Bore (inches)	5.125
Stroke (inches)	3.875
Displacement (cubic inches)	319.8
Compression Ratio	7:1
Dry Weight (lbs)	276
Oil Sump Capacity (qts)	8
Propeller (Sensenich)	M74DM

Fuel and Oil

Fuel Capacity (US gal) Standard	36
Fuel Capacity (US gal) Reserve	50
Oil Capacity (qts)	8
Fuel, Aviation Grade (minimum octane)	80/87

Baggage

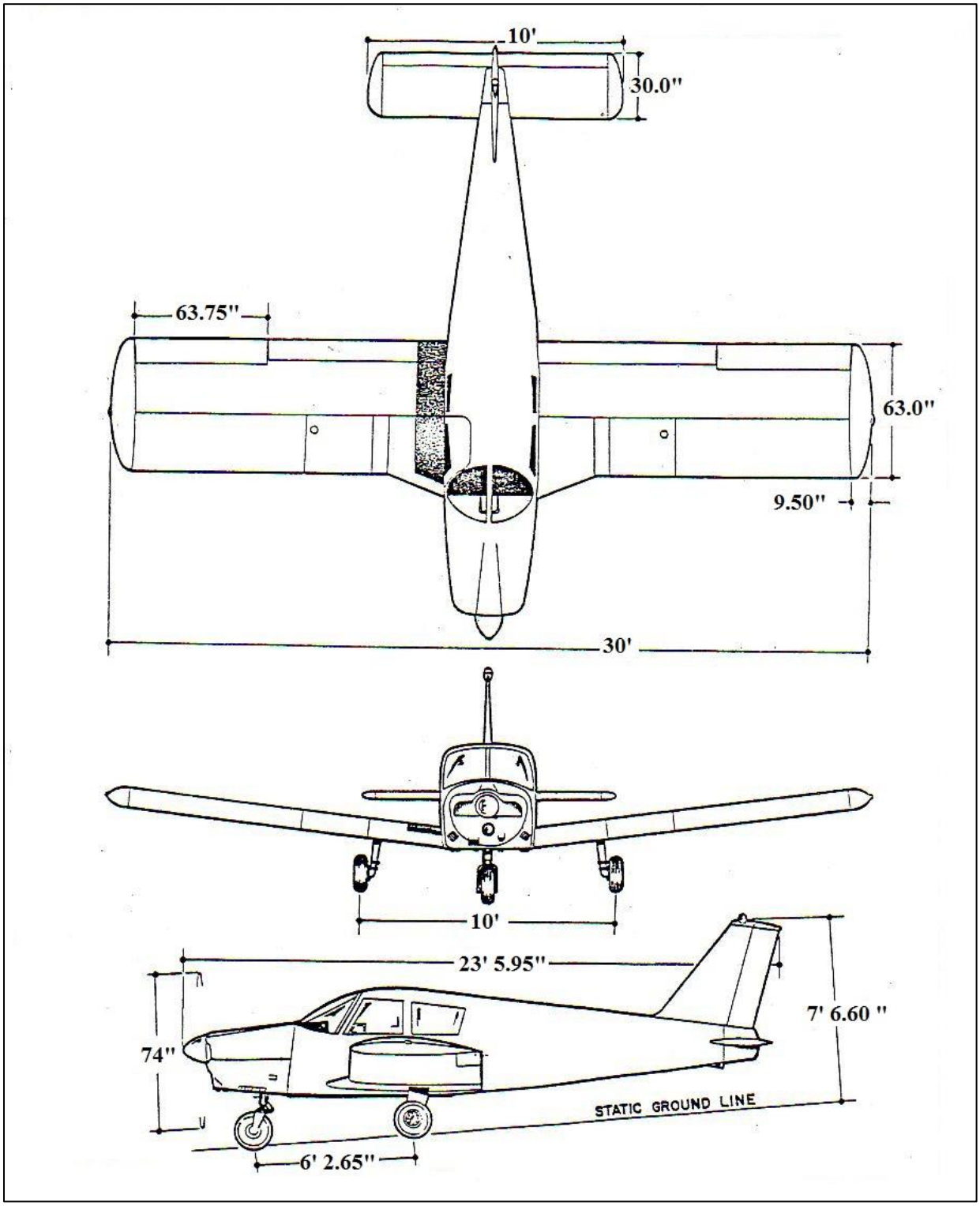
This aircraft is not designed to carry baggage as there are electronic systems which reside in the original baggage compartment necessary to autonomous flight.

Dimensions

Wing Span (ft)	30
Wing Area (sq ft)	160
Wing Loading (lbs per sq ft)	13.4
Length (ft)	23.3
Height (ft)	7.3
Power Loading (lbs per hp)	14.3

Landing Gear

Wheel Base (ft)		6.2
Wheel Tread (ft)		10.0
Tire Pressure (lbs)	Nose	24
	Main	24
Tire Size	Nose (4 ply rating)	6.00 x 6
	Main (4 ply rating)	6.00 x 6


10'
30.0"
63.75"
63.0"
9.50"
30'
10'
23' 5.95"
7' 6.60 "
74"
STATIC GROUND LINE
6' 2.65"

Added Components and Their Weights and Datum

Pitch Mechanism	9 lbs at 0 in
Roll Mechanism	6 lbs at -3 in
Yaw Mechanism	8 lbs at -8 in
Touchscreen	3 lbs at 3 in
Engine Controller	2 lbs at 4 in
Throttle/Mixture Servos	3 lbs at 3 in
Avionics Mounting Hardware	3 lbs at 1 in
Forward Avionics	4 lbs at 0 in
Switches/Lights	1 lb at 3 in
Safety Officer Board	1 lb at 3 in
Flap Motor	5 lbs at 8 in
Brake Motor	3 lbs at 2 in
Electric Trim/Flap Servo/Cable Sensors	3 lbs between front and rear seats
Flight Computer	10 lbs at rear seat
Junction Box	6 lbs at rear seat
Aft Avionics	4 lbs at rear seat
Fire Extinguisher	4 lbs at rear seat
Avionics Battery	30 lbs rear seat
Forward Wire	2 lbs at radio distance
Middle Wire	3 lbs at pilot distance
Aft Wire	2 lbs at passenger Distance

Removed Components and Their Weights

Rear Passenger Seats and their Hardware (pounds)	15
Narco MK12 (pounds)	5.8
Garmin GTX 250XL (pounds)	3.24
Terra Model TM23 Marker Beacon Receiver (pounds)	0.25
TKM Model MG200 (pounds)	6
TKM Model MC60 (pounds)	1.3
Bendix Kind KA134 Audio Panel (pounds)	0.8
Auxiliary Overhead Speaker (pounds)	2

Section II

Design Information

Engine and Propeller

Structures

Landing Gear

Control System

Fuel System

Electrical System

Heating and Ventilation System

Cabin Features

Optional Equipment

Section II

Design Information

Engine and Propeller

The Lycoming O-320-E3D engine installed in the experimental Cherokee PA-28-140 is rated at 150 horsepower at 2700 rpm. The compression ratio for this engine is 7 to 1 and it requires at minimum 80/87 octane fuel. The engine is equipped with direct drive or optional geared drive starter, a computer driven starter for autonomous flight, a 60 ampere alternator, dual magnetos, vacuum pump drive, diaphragm type fuel pump, and a float carburetor.

Exhaust gases are carried through a system constructed of heavy gauge stainless steel which incorporates heater shrouds to provide cabin heat, defrosting, and carburetor deicing.

The propeller used on the PA-28-140 is a Sensenich M74DM fixed-pitch aluminum alloy unit. The diameter is 74 inches with a standard pitch of 58 inches. All performance figures are based on the standard 58 inch pitch propeller.

Cowling on the Cherokee is designed to cool the engine in all normal flight conditions, including protracted climb, without the use of cowl flaps or cooling flanges.

The throttle quadrant containing the manual throttle and mixture controls is located in the lower center of the instrument panel. This quadrant contains a friction lock on the right side which prevents the creeping of the controls. The throttle and mixture control motors for autonomous flight are located behind the throttle quadrant and may be overridden by applying manual force to the controls. The motors will continue in attempting to control the throttle and mixture, so continuous force must be applied to override the system.

To the right of the quadrant is the carburetor heat control that provides maximum carburetor heat when fully ON. Air passes through a highly efficient dry type filter when the carburetor heat is OFF.

Structures

All structures are constructed of aluminum alloy and are designed to withstand load factors well in excess of the normal requirements. All exterior surfaces are primed with etching primer and painted with acrylic lacquer.

The wings are attached to each side of the fuselage by inserting the butt ends of the respective main spars into a spar box carry through. The spar box carry through is an integral part of the fuselage structure. This provides, in effect, a continuous main spar with splices at each side of the fuselage. There are also fore and aft attachments at the rear spar and at an auxiliary front spar.

The wing airfoil section is a laminar flow type, NACA65_2-415 with the maximum thickness about 40% aft of the leading edge. This permits the main spar carry through structure to be located under the rear seat providing unobstructed space ahead of the rear seat location for the electrical systems for autonomous flight.

Landing Gear

The three landing gears use a Cleveland 6.00 x 6 wheel, the main wheels being provided with brake drums and Cleveland single disc hydraulic brake assemblies. The nose wheel and the main gear both use 6.00 x 6 four ply tires. All the tires have tubes.

The nose gear is steerable manually through a 44° arc by use of the rudder pedals. A spring device is incorporated in the rudder pedal torque tube assembly to aid in rudder centering and to provide rudder trim. For autonomous use, the rudder pedals may be operated by a gear box mounted between the top of the pedal bar and the floor. The nose gear steering mechanism also incorporates a hydraulic shimmy dampener.

The oleo struts are of the air-oil type with normal extension being 3.25 inches for the nose gear and 4.5 inches for the main gear under normal static (empty weight of airplane plus full fuel and oil) load.

The manual brakes are actuated by a hand lever and a master cylinder which is located below and near the center of the instrument panel – below the throttle quadrant. The parking brake is incorporated in the lever brake and is operated manually by pulling back on the lever and depressing the knob attached to the top of the handle. To release the parking brake, pull back on the brake lever to disengage the catch mechanism then allow the handle to swing forward provided the automatic brake is not engaged.

For autonomous flight, there is a linear lead screw mechanism to engage the brakes. The brakes are actuated by this mechanism through the linear lead screw providing forward pressure on the hand lever. The parking brake is actuated automatically due to continuous forward pressure on the hand lever by the mechanism. In autonomous use, the computer system will reverse the motor direction and return the linear lead screw to the full back position along with the hand lever. Should the automatic brakes fail in the engaged position, the brake cannot be disengaged manually. To override the automatic brake, a forceful pull on the brake hand lever will detach the hand lever for the linear lead screw.

Control System

Dual Controls are provided as standard equipment with a cable system used between the controls and the surfaces. The horizontal tail is of the all movable slab type, with an anti-servo tab acting as a longitudinal trim tab. It is manually actuated by a control wheel on the floor between the front seats. It is automatically actuated by a motor and chain mechanism attached to the center console behind the trim tab above the cables. The stabilator provides extra stability and control with less size, drag, and weight than conventional tail surfaces. The differential action of the ailerons tends to eliminate adverse yaw in turning maneuvers and reduces the amount of coordination required in normal turns.

The stabilator may be manually operated by pushing and pulling the control wheel to the desired position. It is automatically controlled by a belt system which imitates the push and pull motion on the control wheel. The ailerons are manually operated by rotating the control wheel and are automatically operated by a double sprocket connected to a gear box and the original chain of the control wheel.

The flaps may be manually operated by pulling the handle to one of three predetermined extension positions of 10, 25, and 40 degress and are operated automatically by a simple winch system located below the manual flap handle. The flaps are balanced for light operating forces

and spring loaded to return to the up position. A past-center lock incorporated in the actuating linkage holds the flap when it is in the up position so that it may be used as a step on the right side. The flap will not support a step load except when in the full up position, so it must be completely retracted when used as a step.

Fuel System

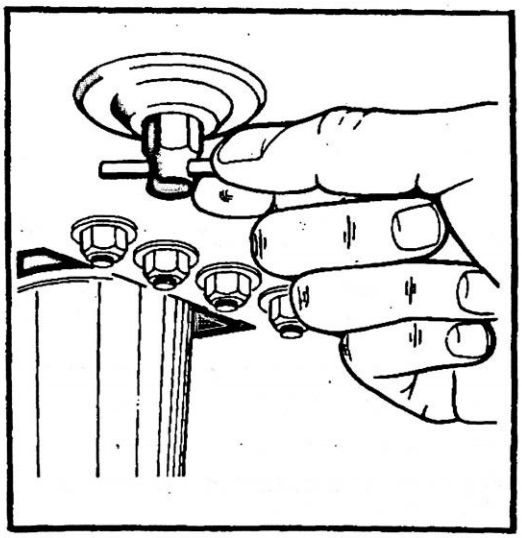
Fuel is stored in two twenty-five gallon tanks which are secured to the leading edge structure of each wing by screws and nut plates to allow easy removal for service or inspection.

The standard quantity of fuel is 36 gallons for the Cherokee 140. To obtain the standard quantity of fuel, fill the tanks to the bottom of the filler neck indicator.

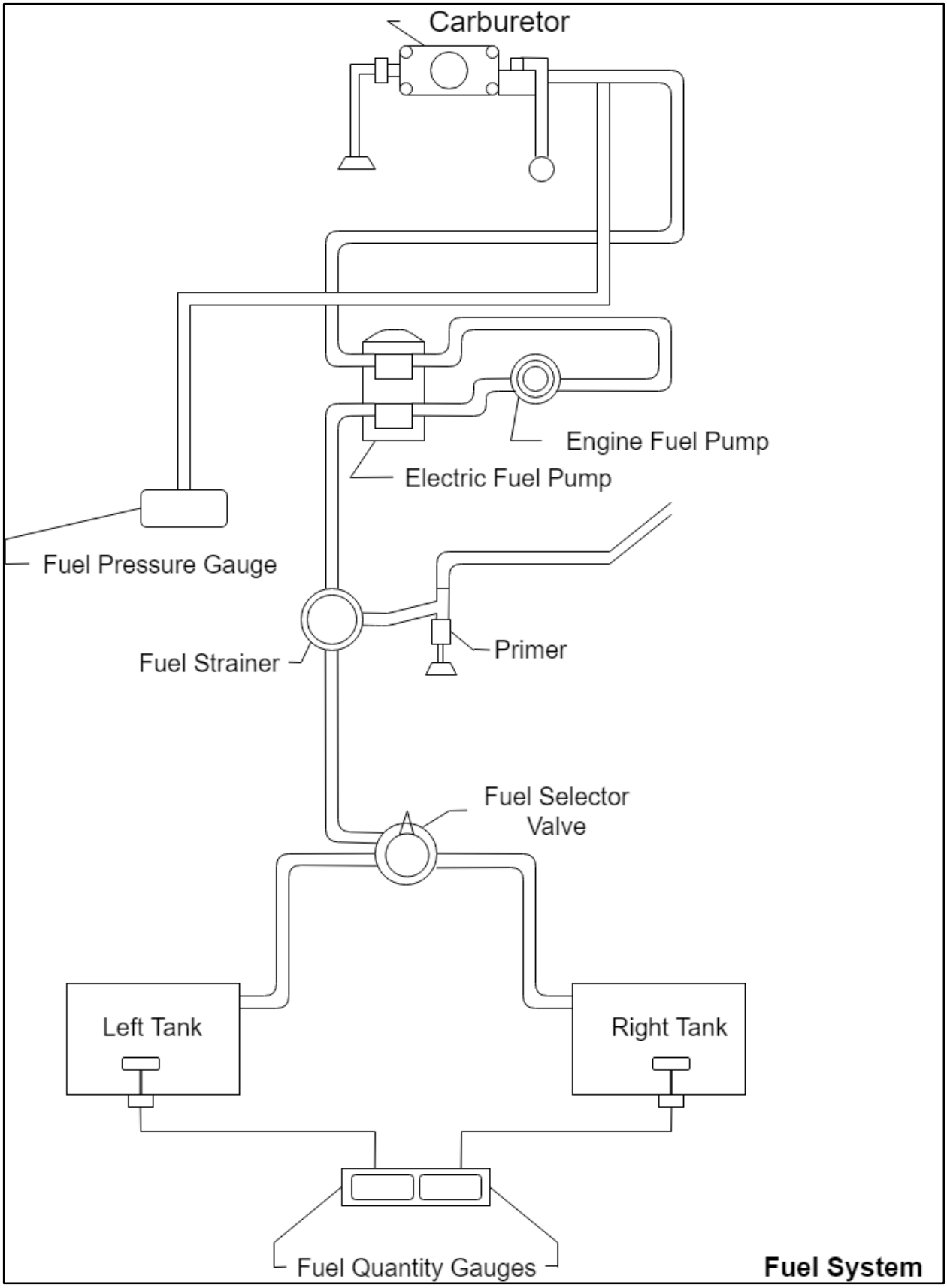
An auxiliary electric fuel pump is provided for use in case of failure of the engine driven fuel pump. The electric pump should be on for all take offs and landings and when switching tanks.

The fuel strainer is equipped with a quick drain and is located on the front lower left corner of the firewall. This strainer should be drained regularly to check for water or sediment accumulation. To drain the lines from the tanks, the tank selector valve must be switched to each tank in turn, with the electric pump on, and the gascolator drain valve opened. Each tank has an individual quick drain located at the bottom, inboard, rear corner.

Fuel quantity and pressure are indicated on gauges located in the engine gauge cluster on the left side of the instrument panel.



Fuel Strainer


Carburetor
Engine Fuel Pump
Electric Fuel Pump
Fuel Pressure Gauge
Fuel Strainer
Primer
Fuel Selector Valve
Left Tank
Right Tank
Fuel Quantity Gauges
Fuel System

Electrical System

The electrical system includes a 12 volt 60 ampere alternator, two batteries, voltage regulator, overvoltage relay, master switch relay, safety officer, and junction box. The original battery and master switch relay are located beneath the baggage compartment floor and the second battery is located to the left of the original battery. Access for service or inspection is obtained by raising the hinged floor panel. The regulator and overvoltage relay are located on the forward left side of the fuselage behind the instrument panel.

Electrical switches are located on the right center instrument panel, and the circuit breakers are located on the lower right instrument panel and on the junction box. A rheostat switch on the right side of the switch panel controls the navigation lights and the intensity of the instrument panel light.

Standard electrical accessories include manual starter, computer driven starter, electric fuel pump, stall warning indicator, cigar lighter, and ammeter. Navigation lights, anti-collision lights, landing light, and instrument panel lighting are offered as optional accessories.

The Flite Liner includes as standard electrical accessories: manual starter, computer driven starter, electrical fuel pump, stall warning indicator, ammeter, navigation lights, anti-collision light, landing light, and instrument panel light.

Circuit provisions are made to handle a complete complement of communications and navigational equipment.

The alternator system offers many advantages over the generator system. The main advantage is full electrical power output at much lower engine RPM and results in improved radio and electrical equipment operation. Since the alternator output is available all the time, the battery will be charging almost continuously. This will make cold weather starting easier.

In generator systems**,** the ammeter indicates battery discharge. In the Cherokee electrical system the ammeter displays in amperes the load placed on the alternator. With all electrical equipment except the master switch in the OFF position, the ammeter will indicate the amount of charging current demanded by the battery. As each item of electrical equipment is turned on, the current will increase to a total appearing on the ammeter. This total includes the battery. The maximum continuous load for night flight with radios on is about 30 amperes. This 30 ampere value plus approximately 2 amperes for a fully charged battery will appear continuously under these flight conditions.

The master switch is a split switch with the left half operating the master relay and the right half energizing the alternator. The switch is interlocked so that the alternator cannot be operated without the battery. For normal operation, be sure both halves are turned on.

Maintenance on the alternator should prove to be a minor factor. Should service be required, contact the local piper dealer. Do not take off with a fully discharged battery as 3 volts are needed to excite the alternator.

Heating and Ventilating System

Heat for the cabin interior and the defroster system is provided by a heater muff attached to the exhaust system. The amount of heat desired can be regulated with the controls located on the far right side of the instrument panel. If unusual odors are noticed, the heat should be shut off

and the system inspected for leaks. The air flow may be regulated between the front and rear seats by the use of the levers located on top of the heat ducts next to the control console.

Fresh air inlets are located in the leading edge of the wing at the intersection of the tapered and straight sections. A large adjustable outlet is located on the side of the cabin near the floor at each seat location, overhead air outlets are offered as optional equipment. Cabin air is exhausted through an outlet located below the rear seat floor panel.

On Flite Liner overhead air outlets are not offered as optional equipment.

Cabin Features

The instrument panel of the Cherokee is designed to accommodate the customary advanced flight instruments and all the normally required powerplant instruments. The artificial horizon and directional gyro are vacuum operated through use of a vacuum pump installed on the engine, while the turn indicator is electrically operated. A vacuum gauge is mounted on the far right side of the instrument panel. A natural separation of the flight group and power group is provided by placing the flight group in the upper instrument panel and the power group in the sub panel center. The radios and circuit breakers located on the right hand instrument panel have extra circuits provided for a complete line of optional radio equipment. The microphone is located on the console cover.

The front seats are adjustable fore and aft, and recline for pilot-passenger comfort and ease of entry and exit.

A single strap shoulder harness controlled by an inertia reel is standard equipment for the front seats. The shoulder strap is routed over the shoulder adjacent to the windows and attached to the inboard lap strap in the general area of the occupants inboard hip.

A check of the inertia reel mechanism is made by pulling sharply on the strap. The reel will lock in place under this test and prevent the strap from extending. Under normal movement the strap will extend and retract as required.

Section III

Operating Instructions

Preflight

Starting Engine

Warm Up

Ground Check

Take-Off

Climb

Stalls

Cruising

Maneuvers

Approach and Landing

Stopping Engine

Mooring

Weight and Balance

Operating Tips

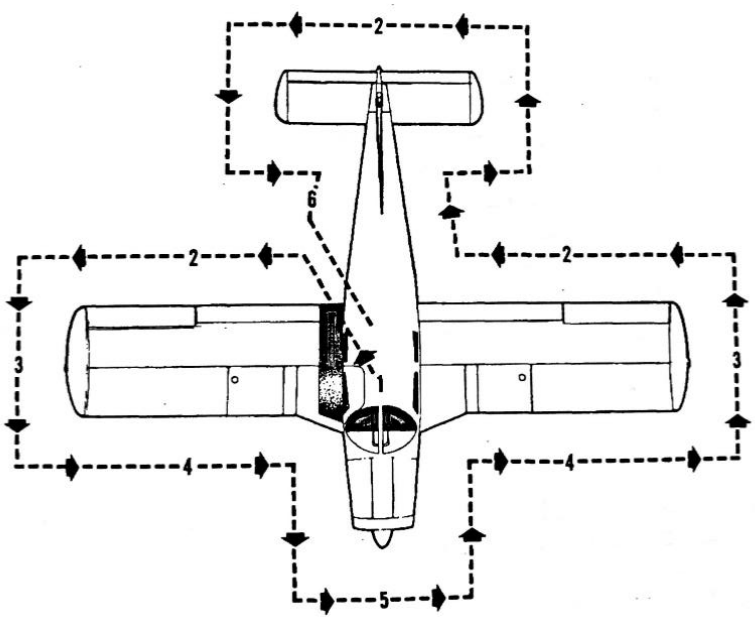
Section III

Operating Instructions

Preflight

The preflight will be conducted manually. The airplane should be given a thorough visual inspection prior to each flight. Particular attention should be given to the following items:

1. A. remove seat belt securing control wheel
   B. Master switch ON
   C. Check fuel quantity indicators (two tanks)
   D. Master switch and ignition OFF



2. A. Check for external damage, operational interferences of control surfaces or hinges
   B. Insure that wings and control surfaces are free of snow, ice, or frost

3. A. Visually check fuel supply, secure caps
   B. Drain all fuel system sumps and lines
   C. Check that fuel system vents are open

4. A. Check landing gear shock struts for proper inflation
   B. Check tires for cuts, wear, and proper inflation
   C. Check brake blocks and discs for wear and damage

5. A. Inspect windshield for cleanliness
   B. Check the propeller and spinner for defects or nicks
   C. Check for obvious fuel or oil leaks
   D. Check oil level, 8 quarts maximum (insure dipstick is properly seated)
   E. Inspect cowling and inspection covers for security
   F. Check nose wheel tire for inflation, wear

G. Check nose wheel shock strut for proper inflation
H. Check for foreign matter in air inlets

6. A. Stow tow bar and control locks if used
   B. Check baggage for proper storage and security
   C. Close and secure the baggage compartment door

7. A. Upon entering the aircraft ascertain that all primary flight controls operate properly
   B. Close and secure the cabin door
   C. Check that required papers are in order and in the aircraft
   D. Fasten seat belts and shoulder harness. Check function of inertia reel

Starting Engine

Manual Operations for Every Case

1. Set parking brake ON
2. Set the Carburetor heat control in the full COLD (OFF) position
3. Select the desired tank with fuel selector valve

Manually Starting engine when cold

1. Open throttle approximately ¼ inch
2. Turn the master switch ON
3. Turn the electric fuel pump ON
4. Move the mixture control to FULL RICH
5. Engage the starter by rotating magneto switch clockwise and pressing in
6. When the engine fires, advance throttle to desired setting. If the engine does not fire within five to ten seconds, disengage starter and prime with one to three strokes of the priming pump. Repeat the starting procedure

Manually Starting Engine when Hot

1. Open the throttle approximately ½ inch
2. Turn the Master Switch ON
3. Turn the electric fuel pump ON
4. Put the mixture control in IDLE CUT-OFF
5. Engage the starter by rotating magneto switch clockwise and pressing in. When the engine fires, advance the mixture control and move the throttle to the desired setting

Manually Starting Engine when Flooded

1. Open the throttle full
2. Turn the master switch ON
3. Turn the electric fuel pump OFF
4. Put mixture control in IDLE CUT-OFF
5. Engage the starter by rotating magneto switch clockwise and pressing in. When the engine fires, advance the mixture control and retard the throttle.

When the engine is firing evenly, advance the throttle to 800 RPM. If oil pressure is not indicated within thirty seconds, stop the engine and determine the trouble. In cold weather it will take a few seconds longer to get an oil pressure indication. If the engine has failed to start, refer to the "Lycoming Operating Handbook, Engine Troubles and Their Remedies."

Starter manufacturers recommend that cranking periods be limited to thirty seconds with a two minute rest between cranking periods. Longer cranking periods will shorten the life of the starter.

Automatic Starting of Engine (engine must be warm for automatic starting)

1. Ensure magneto safety switches are grounded
2. Turn the Master Battery switch ON
3. Junction Box main power ON
4. Check circuit breakers on Junction Box
5. Power up the computer
6. Safety ignition switch ON
7. Monitor gauges and systems as computer begins the process of completing items found on the manual checklist for a Normal Start

When the engine is firing evenly, the automatic system will advance the throttle to 800 RPM. If oil pressure is not indicated within thirty seconds, automatic system will follow the process to stop the engine and the trouble can be evaluated once the system has been disengaged. The automat start is not advised for use in cold weather as it will take a few seconds longer to get an oil pressure indication, therefore the system my determine that there is an unresolved issue where there is none. If the engine has failed to start, refer to the "Lycoming Operating Handbook, Engine Troubles and Their Remedies."

Warm-up

As soon as the engine starts after a manual starting procedure, the oil pressure should be checked. If no pressure is indicated within thirty seconds, stop the engine and determine the trouble. In cold weather it will take a few seconds longer to get an oil pressure indication. Warm-up the engine at 800-1200 RPM.

Take-off may be made as soon as ground check is completed, providing that the throttle may be opened fully without backfiring or skipping, and without reduction in engine oil pressure.

Ground Check

For manual Flight, check the magnetos at 2000 RPM by switching from Both to right then back to both before switching to left. Differential drop should not exceed 50 RPM while the total drop on either magneto should not exceed 175 RPM.

Check both the oil temperature and pressure. The temperature may be low for some time if the engine is being run for the first time of the day, but as long as the pressure is within limits the engine is ready for take-off.

Carburetor heat should also be checked prior to take-off to be sure that the control is operating properly and to clear any ice which may have formed during taxiing. Avoid prolonged ground operation with carburetor heat ON as the air is unfiltered.

Operation of the engine driven fuel pump should be checked while taxiing or during pretake-off engine run up by switching off the electric fuel pump and observing fuel pressure. The electric fuel pump should be on during take-off to prevent loss of power should the engine driven pump fail. The engine is warm enough for take-off when the throttle can be opened without the engine faltering.

For automatic flight, the system will continuously monitor the indicators through installed sensors and will follow the manual ground check items in a checklist format. It will run through each step while monitoring gauges and indicators for the indication of any trouble.

Take-off

Just before take-off the following items should be checked for manual flight:

1. Fuel on proper tank
2. Electric fuel pump - ON
3. Engine Gauges checked
4. Flaps - set
5. Carburetor heat - OFF
6. Mixture - set
7. Seat Backs erect
8. Fasten belts/harnesses
9. Trim tab - set
10. Controls - free and correct
11. Door – latched

In autonomous flight, the computer system will follow each item on the above checklist while monitoring the voltage of each of the batteries as well as the gauge readings through installed sensors. Should the system encounter a problem, it will initiate the appropriate response procedure to either alleviate the problem or return to its starting position.

In the conventional take-off procedure set the trim control slightly aft of neutral. Allow the airplane to accelerate to 50 to 60 miles per hour, then ease back on the wheel enough to let the airplane fly itself from the ground. Premature or excessive raising of the nose will result in a delayed take-off. After take-off let the aircraft accelerate to the desired climb speed by lowering the nose slightly.

Short Field, Obstacle Clearance

Lower the flaps to 25° (second notch), accelerate to 55-60 miles per hour and ease back on the control wheel to rotate. After breaking ground, accelerate to the best and of climb speed, 78 miles per hour. Slowly retract the flaps when the obstacle has been cleared and continue to climb at 89 miles per hour.

For autonomous flight, if the computer receives and input that the runway is a short field with an obstacle to be cleared, the system will execute the previously listed procedure automatically.

Short Field, No Obstacles

Lower the flaps to 25° (second notch), accelerate to 55-60 miles per hour and ease back on the control wheel to rotate. Accelerate to best rate of climb speed, 89 miles per hour. Slowly retract the flaps while climbing out.

For autonomous flight, if the computer receives and input that the runway is a short field, the system will execute the previously listed procedure automatically.

Soft Field, No Obstacles

Lower the flaps to 25° (second notch), accelerate aircraft and pull nose gear from ground as soon as possible, lift off at the lowest possible airspeed. Accelerate just above the ground to best rate of climb speed, 89 miles per hour. Climb out while slowly retracting the flaps.

For autonomous flight, if the computer receives and input that the runway is a soft field, the system will execute the previously listed procedure automatically.

Soft Field, Obstacle Clearance

Lower the flaps to 25° (second notch), accelerate aircraft, pull nose gear off as soon as possible and lift off at lowest possible airspeed. Accelerate just above the ground to best angle of climb speed, 78 miles per hour, to climb past obstacle clearance height, continue climb while - accelerating to best rate of climb speed, 89 miles per hour, and slowly retract the flaps.

For autonomous flight, if the computer receives and input that the runway is a soft field with an obstacle to be cleared, the system will execute the previously listed procedure automatically.

Climb

The best rate of climb airspeed at gross weight is 89 miles per hour while the best angle of climb airspeed is 78 miles per hour. At lighter than gross weight these speeds are reduced. For a manned flight the recommended enroute climbing speed of 100 miles per hour provides increased visibility over the nose. Shallow turns of a few degrees will also aid forward visibility during climb out. The autonomous system will use the same enroute climbing speed of 100 miles per hour for the purpose of integration with other aircraft, however it is not necessary for increased visibility.

Stalls

Stall characteristics of the Cherokee are conventional. Visual stall warning is provided by a red light located on the left side of the instrument panel which is turned on automatically between 5 and 10 miles per hour above stall speed along with an aural stall horn. Gross weight stalling speed with power off and full flaps is 55 miles per hour at 2150 pounds. With flaps up this speed is increased by 9 miles per hour.

Intentional spins are prohibited in the normal category airplane. In manual flight, lazy eights and chandelles may be performed in the normal category provided a 60° angle of bank and/or a 30° angle of pitch is not exceeded. The autonomous system is not currently rated for commercial maneuvers and caution must be given when performing these maneuvers in this modified aircraft. For approved maneuvers and entry speeds refer to the Flight Manual.

Stall Speed Table		
Angle of Bank	Flaps 40°	Flaps Retracted
0°	55 MPH	64 MPH
20°	57 MPH	66 MPH
40°	63 MPH	73 MPH
60°	78 MPH	91 MPH
Power Off - Gross Weight 2150 lbs.		

Cruising

The cruising speed is determined by many factors including power setting, altitude, temperature, loading, and equipment installed on the airplane.

The normal cruising power is 75% of the rated horsepower of the engine. True airspeeds which may be obtained at various altitudes and power settings can be determined from the charts in Section IV of this handbook. The computer controlled system can obtain true airspeeds through evaluation of inputs such as density altitude with current readings of altitude and power setting for current and accurate airspeed outputs.

Use of the mixture control in cruising flight reduces fuel consumption significantly, especially at higher altitudes. The mixture should be leaned during cruising operation above 5000 feet altitude and at pilot's discretion at lower altitudes when 75% power or less is being used. If any doubt exists as to the amount of power being used, the mixture should be in the FULL RICH position for all operations under 5000 feet. When level flight has been established by the autonomous system, the mixture control will be adjusted by the system based on atmospheric conditions and the desired fuel economy in cruise. In other phases of flight the mixture will be adjusted to the appropriate level based on procedures and the cruise conditions.

To lean the mixture, pull the mixture control until the engine becomes rough, indicating that the lean mixture limit has been reached in the leaner cylinders. Then enrich the mixture by pushing the control towards the instrument panel until engine operation becomes smooth. Motors behind the throttle quadrant allow the system to automatically control the mixture position.

The continuous use of carburetor heat during cruising flight decreases engine efficiency. Unless icing conditions in the carburetor are severe, do not cruise with the heat on. Apply full carburetor heat slowly and only for a few seconds at intervals determined by icing severity. The autonomous system should not be operated in conditions where icing may occur, the system is capable of checking for icing and applying the carburetor heat as necessary.

In order to keep the airplane in best lateral trim during cruising flight, the fuel should be used alternately from each main tank. It is recommended that one main tank be used for one hour after take-off; the other main tank used until nearly exhausted, then return to the first main tank. The autonomous system cannot switch tanks automatically with the fuel selector switch, however there is a fuel selector setting in the computer system

Maneuvers

The airplane is approved for certain aerobatic maneuvers, provided it is loaded with the approved weight and center of gravity limits and that the aircraft is manned for such maneuvers. (See airplane flight manual). The maneuvers are spins, steep turns, lazy eights, and chandelles. The autonomous system will not perform such maneuvers, manual override is necessary to perform the above list of maneuvers.

Approach and Landing

Manual Landing Checklist

1. Fuel on proper tank
2. Mixture - RICH
3. Electric fuel pump ON
4. Seat backs erect
5. Flaps - set (115 MPH)
6. Fasten belts/harnesses

The airplane should be trimmed to an approach speed of about 85 miles per hour with the flaps up. The flaps can be lowered at speeds up to 115 miles per hour, approach speed is reduced by 3 miles per hour for each notch of flaps used. Carburetor heat should not be applied unless there is an indication of carburetor icing, since the use of carburetor heat causes a reduction in power which may be critical in case of a go-around. Full throttle operation with carburetor heat on is likely to cause detonation.

The amount of flap used during landings and the speed of the aircraft at contact with the runway should be varied according to the landing surface and existing conditions, both windwise and loadwise. It is generally good practice to contact the ground at the minimum possible safe speed consistent with existing conditions.

Normally, the best technique for short and slow landings is to use full flap and enough power to maintain the desired airspeed and approach flight path. Mixture should be full rich, fuel on the fullest tank, carburetor heat off, and electric fuel pump on. Reduce the speed during the flareout and contact the ground close to the stalling speed (55 to 65 mph). After ground contact hold the nose wheel off as long as possible. As the airplane slows down, drop the nose and apply the brakes. There will be less chance of skidding the tires if the flaps are retracted before applying the brakes. Braking is most effective when back pressure is applied to the control wheel , putting most of the aircraft weight on the main wheels. In high wind conditions, particularly in strong crosswinds, it may be desirable to approach the ground at higher than normal speeds with partial or no flaps.

Stopping Engine

At the Pilot's discretion, the flaps should be raised and the electric fuel pump turned off. After parking, the radios should be turned off and the engine stopped by pulling the mixture control to idle cut-off. The throttle should be left at full aft to avoid engine vibration while stopping. Then the magnetos and master switches should be turned off. In autonomous engine shut off, the computer will automatically perform the previously mentioned steps.

Mooring

The Cherokee should be moved on the ground with the aid of the nose wheel tow bar provided with each plane and secured in the baggage compartment. Tie downs may be secured to rings provided under each wing, and to the tail skid. The aileron and stabilator control wheel shaft are secured by looping the seat belt through the control wheel and pulling it tight. The rudder is held in position by its connections to the nose wheel steering, and normally does not have to be secured. The flaps are locked when in the full up position, and should be left retracted.

Weight and Balance

It is the responsibility of the owner and pilot to determine that the airplane remains within the allowable weight vs. center of gravity envelope while in flight for manned flight and before the autonomous system is engaged for automatic flight. For weight and balance data see the airplane manual and weight and balance form supplied with each airplane.

Operating Tips

The following operating tips are of particular value in the operation of the Cherokee 140, particularly for manual flight.

1. Learn to trim for take-off so that only a very light back pressure on the wheel is required to lift the airplane off the ground.
2. The best speed for take-off is about 60 MPH under normal conditions. Trying to pull the airplane off the ground at too low an airspeed decreases the controllability of the airplane in event of engine failure.
3. Flaps may be lowered at airspeeds up to 115 MPH. To reduce flap operating loads, it is desirable to have the airplane at a slower speed before extending the flaps
4. Before attempting to reset any circuit breaker, allow a two to five minute cooling off period
5. Before starting the engine, check that all radio switches, light switches, and the pitot heat switch are in the OFF position so as not to create an overloaded condition when the starter is engaged.
6. The overvoltage relay is provided to protect the electronics equipment from a momentary overvoltage condition (approximately 16.5 volts and up), or a catastrophic regulator failure. In the event of a momentary condition, the relay will open and the ammeter will indicate "0" output from the alternator. The relay

may be reset by switching the ALT switch to OFF for approximately 30 seconds and then returning the ALT switch to ON. If after recycling the ALT switch the condition persists, the flight should be terminated as soon as practical, reduce the battery load to a minimum.

7. The vacuum gauge is provided to monitor the pressure available to assure the correct operating speed of the vacuum driven gyroscopic flight instruments, it also monitors the condition of the common air filter by measuring the flow of air through the filter.

   If the vacuum gauge registers lower than 5" +/- .10" Hg at 2000 RPM, the following items should be checked off before flight.

   1. Common air filter, could be dirty or restricted
   2. Vacuum lines could be collapsed or broken
   3. Vacuum pump, worn
   4. Vacuum regulator, not adjusted correctly. The pressure, even though set correctly, can read lower under two conditions: (1) Very high altitude, above 12,000 feet, (2) Low engine RPM usually on approach or during training maneuvers. This is normal and should not be considered a malfunction.

Section IV

Emergency Procedures

Introduction

Junction Box

Safety Officer Board

Override

Ground Operations

Take-Off

In Flight

- **Power Off Landing**
- **Fire**
- **Loss of Oil Pressure**
- **Loss of Fuel Pressure**
- **High Oil Temperature**
- **Alternator Failure**
- **Engine Roughness**
- **Spins**
- **Open Door**

Section IV

Emergency Procedures

Introduction

This section contains procedures that are recommended if an emergency condition should occur during ground operation, take-off, or in flight. These procedures are suggested as the best course of action for coping with the particular condition described, but are not a substitute for sound judgement and common sense, contrary to the autonomous system's course of action in an emergency. Since emergencies rarely happen in modern aircraft, their occurrence is usually unexpected, and the best corrective action may not always be obvious. Pilots should familiarize themselves with the procedures given in this section and be prepared to take appropriate action should an emergency arise.

Most basic emergency procedures, such as power off landings, are a part of normal pilot training. Although these emergencies are discussed herein, this information is not intended to replace such training, but only to provide a source of reference and review, and to provide information on procedures which are not the same for all aircraft. It is suggested that the pilots review standard emergency procedures periodically to remain proficient in them.

Junction Box

The junction box is located between and slightly back from the front seats. The junction box contains the circuit breakers for the autonomous systems as well as the master kill switch.

Safety Officer Board

The safety officer board maintains a constant monitor of all autonomous systems including motor function and battery voltage to ensure system is running smoothly. The safety officer board has its own backup battery separate from the aircraft power, this allows the safety officer board to remain functional should there be a power loss in the aircraft.

The safety officer board has an LED display that will signal if battery voltages drop too low. Should this happen before take-off, the flight should be abandoned and the systems inspected.

The safety officer board has relays that it may actuate to interrupt the main motors; this allows the safety officer board to cut power to the motors should there be a malfunction that keeps the motors running. The safety officer board is the first method to override the autonomous system. Should the safety officer board fail, there are three position switches for manual control of the safety officer board. The upper position for the switch allows the safety officer to maintain control, the middle (neutral) position will keep the safety officer board disconnected, and the lower position for the switch will disengage the safety officer board and will run a back driving motor for the autonomous mechanisms. The switch will not remain in the lower position if released, it will return to the neutral position. In order to keep back driving the motors, the switch

must be pushed down to the lower position continuously. The three switches control the three clutch motors for the main flight controls: ailerons, stabilator, rudder.

Override

In all instances of an emergency with a pilot in the aircraft, the autonomous system should be overridden and manual control maintained. Become familiar with the override of each system in case of necessary manual override. The following steps can be taken to ensure that the system is overridden in order to perform emergency maneuvers and landings:

1. Should an emergency arise, an input to the safety officer board to disconnect all automatic systems will disconnect the motors so that manual flight may be obtained
2. If safety officer board does not perform as stated above, push safety officer switch to bottom position, release to neutral position to ensure motors remain disconnected
3. Pitch Override – pull string connected to clutch, if motor continues, pull gear pin
4. Roll Override - pull string connected to clutch, if motor continues, pull gear pin
5. Yaw Override - pull string connected to clutch, if motor continues, pull gear pin
6. Flaps – power will be cut to the flap mechanism if the safety officer is disengaged, can be moved by manual force
7. Mixture and Throttle – power to motors will be cut if the safety officer is disengaged, can be moved by manual force
8. Rudder Trim – can be moved manually without any override
9. Stabilator Trim – can be moved manually without any override

Ground Operations

Engine Fire During Start

Engine fires during start are usually the result of over priming. The procedures below are designed to draw the excess fuel back into the induction system:

1. If engine has not started
    a. Mixture - Idle Cut-Off
    b. Throttle - Open
    c. Turn engine with starter (This is an attempt to pull the fire into the engine
2. If engine has already started and is running, continue operating to try to pull the fire into the engine
3. In either case stated in (1) and (2), f the fire continues longer than a few seconds, the fire should be extinguished by the best available external means
4. If external fire extinguishing is to be applied:
    a. Fuel Selector Valves - OFF
    b. Mixture - Idle Cut-Off

Take-Off

Engine Power Loss During Take-Off

The proper action to be taken if loss of power occurs during take-off will depend on circumstances.

1. If sufficient runway remains for a normal landing, land straight ahead
2. If insufficient runway remains, maintain a safe airspeed and make only a shallow turn to avoid obstructions. Use of flaps depends on circumstances. Normally, flaps should be fully extended for touchdown.
3. If you have gained sufficient altitude to attempt a restart, prodeed as follows
    a. MAINTAIN SAFE AIRSPEED
    b. FUEL SELECTOR - SWITCH TO ANOTHER TANK CONTAINING FUEL
    c. ELECTRIC FUEL PUMP - CHECK ON
    d. MIXTURE - CHECK RICH
    e. CARBURETOR HEAT - ON

Note:
If engine failure was caused by fuel exhaustion, power will not be regained after tanks are switched until empty fuel lines are filled, which may require up to ten seconds

If power is not regained, proceed with the POWER OFF LANDING procedure.

In Flight

Engine Power Loss in Flight

Complete engine power loss is usually caused by fuel flow interruption, and power will be restored shortly after fuel flow is restored. If power loss occurs at low altitude, the first step is to prepare for an emergency landing (see POWER OFF LANDING). Maintain an airspeed of at least 80 MPH IAS, and if altitude permits, proceed as follows:

1. Fuel Selector - switch to another tank containing fuel
2. Electric Fuel Pump - ON
3. Mixture - RICH
4. Carburetor Heat - ON
5. Engine Gauges - check for an indication of the cause of Power Loss
6. Primer - check locked
7. If no Fuel Pressure is indicated, check tank selector position to be sure it is on a tank containing fuel.
8. Carburetor heat - OFF
9. Electric Fuel Pump - OFF

If the above steps do not restore power, prepare for an emergency landing. If time permits:

1. Ignition switch - “L” then “R” then back to “BOTH”
2. Throttle and Mixture - Different Settings (This may restore power if problem is too rich or too lean of a mixture, or partial fuel system restriction)
3. Try another fuel tank - (Water in the fuel could take some time to be used up, and allowing the engine to windmill may restore power. If power loss is due to water, fuel pressure indications will be normal)

Note:
If engine failure was caused by fuel exhaustion, power will not be regained after tanks are switched until empty fuel lines are filled, which may require up to ten seconds

If power is not restored, proceed with POWER OFF LANDING procedures.

Power Off Landing

If loss of power occurs at altitude, trim the aircraft for best gliding angle (80 MPH IAS)(Air Cond. OFF) and look for a suitable field. If measures are taken to restore power are not effective, and if time permits, check your charts for airports in the immediate vicinity; it may be possible to land at one if you have sufficient altitude. If not possible, notify the FAA by radio of your difficulty and intentions. If another pilot or passenger is aboard, let them help.

When you have located a suitable field, establish a spiral pattern around this field. Tre to be at 1000 feet above the field at the downwind position, to make a normal approach. Excess altitude may be lost by widening your pattern, using flaps or slipping, or a combination of these.

Touchdowns should normally be made at the lowest possible airspeed with full flaps.

When committed to landing:

1. Ignition - OFF
2. Master Switch - OFF
3. Fuel Selector - OFF
4. Mixture - Idle Cut-Off
5. Seat Belt tight, Shoulder Harness in Place

Fire

The presence of fire is noted through smoke, smell, and heat in the cabin. It is essential that the source of the fire be promptly identified through instrument readings,

character of the smoke, or other indications, since the action to be taken differs somewhat in each case.

Source of fire, Check:

1. Electrical Fire (Smoke in the cabin)
    a. Master Switch - OFF
    b. Vents - OPEN
    c. Cabin Heat - OFF
    d. Land as soon as possible
2. Engine Fire (In Flight)
    a. Fuel Selector - OFF
    b. Throttle - Closed
    c. Mixture - Idle Cut-Off
    d. Heater - Off (In all cases of fire)
    e. Defroster - Off (In all cases of fire)
    f. If terrain permits, land immediately

The possibility of an engine fire in flight is extremely remote. The procedure given above is general and pilot judgement should be te deciding factor for action in such an emergency.

Loss of Oil Pressure

Loss of oil pressure may either be partial or complete. A partial loss of oil pressure usually indicates a malfunction in the oil pressure regulating system, and a landing should be made as soon as possible to investigate the cause, and prevent engine damage.

A complete loss of oil pressure indication may signify oil exhaustion or may be the result of a faulty gauge. In either case, proceed toward the nearest airport, and be prepared for a forced landing. If the problem is not a pressure gauge malfunction, the engine may stop suddenly. Maintain altitude until such time as a dead stick landing can be accomplished. Don't change power settings unnecessarily, as this may hasten complete power loss.

Depending on the circumstances, it may be advisable to make an off airport landing while power is still available, particularly if other indications of actual oil pressure loss, such as sudden increase in temperatures, or oil smoke, are apparent, and an airport is not close.

If engine stoppage occurs, proceed to POWER OFF LANDING.

Loss of Fuel Pressure

1. Electric Boost Pump - ON
2. Fuel Selector - Check on full tank

If the problem is not an empty fuel tank, land as soon as practicable, and have engine driven fuel pump checked.

High Oil Temperature

An abnormally high oil temperature indication may be caused by a low oil level, an obstruction in the oil cooler, damaged or improper baffle seals, a defective gauge, or other causes. Land as soon as practicable at an appropriate airport and have the cause investigated.

A steady rapid rise in oil temperature is a sign of trouble. Land at the nearest airport and let a mechanic investigate the problem. Watch the oil pressure gauge for an accompanying loss of pressure.

Alternator Failure

Loss of alternator output is detected through a zero reading on the ammeter. Before executing the following procedure, insure that the reading is zero and not merely low by actuating an electrically powered device, such as the landing light. If no increase in the ammeter reading is noted, alternator failure can be assumed.

1. Reduce electrical load
2. Alternator circuit breakers - Check
3. "Alt" switch - Off (for 30 seconds), Then On

If the ammeter continues to indicate no output, or alternator will not stay reset, turn off "Alt" switch, maintain minimum electrical load and land as soon as practical. All electrical load is being supplies by the battery.

Engine roughness is usually due to carburetor icing which is indicated by a drop in RPM, and may be accompanied by a slight loss of airspeed or altitude. If too much ice is allowed to accumulate, restoration of full power may not be possible; therefore, prompt action is required.

1. Carburetor heat ON (See Note), RPM will decrease slightly and roughness will increase. Wait for a decrease in engine roughness or an increase in RPM, indicating ice removal. If no change in approximately one minute, return carburetor heat to COLD. If the engine is still rough, try steps below:
    a. Mixture - adjust for maximum smoothness. Engine will run rough if too rich or too lean
    b. Electric Fuel Pump - ON
    c. Fuel Selector - Change to other tank to see if fuel contamination is the problem
    d. Engine Gauges - Check for abnormal readings. If any gauge readings are abnormal, proceed accordingly.
    e. Magneto switch - "L" then "R" then back to "BOTH". If operation is satisfactory on either magneto, proceed on that magneto at reduced power, with mixture full rich, to a landing at the first available airport.

    If Roughness persists, prepare for a precautionary landing at pilot's discretion.

Note:

Partial Carburetor heat may be worse than no heat at all, since it may partially melt ice, which will refreeze in the intake system. When using carburetor heat, therefore, always use full heat, and when ice is removed return the control to the full cold position.

Spins

Intentional spins are prohibited in the normal category airplane and the utility category airplane when air conditioning is installed. For approved maneuvers as a utility category airplane, refer to the Flight Manual.

1. Throttle - idle
2. Rudder - Full opposite to direction of rotation
3. Control wheel - full forward
4. Rudder - neutral (when rotation stops)
5. Control wheel - as required to smoothly regain level flight attitude

Open Door

The cabin door on the Cherokee is double latched, so the chances of it springing open in flight at both the top and bottom are remote. However, should you forget the upper latch, or not full engage the lower latch, the door may spring partially open. This will usually happen at take-off or soon afterward. An open door will not affect normal flight characteristics, and a normal landing can be made with the door open.

If both upper and lower latches open, the door will trail slightly open, and airspeed will reduce slightly.

To close the door in flight, proceed as follows:

1. Slow aircraft to 100 MPH IAS
2. Cabin vents - Close
3. Storm Window - Open
4. If upper latch is open - latch, If lower latch is open - open top latch, push door further open, and then close rapidly. Latch top latch.

   A slip in the direction of the open door will assist in latching procedure.

Section V

Performance Charts

Altitude Conversion Chart

Take Off Distance vs. Density Altitude

Rate of Climb vs. Density Altitude

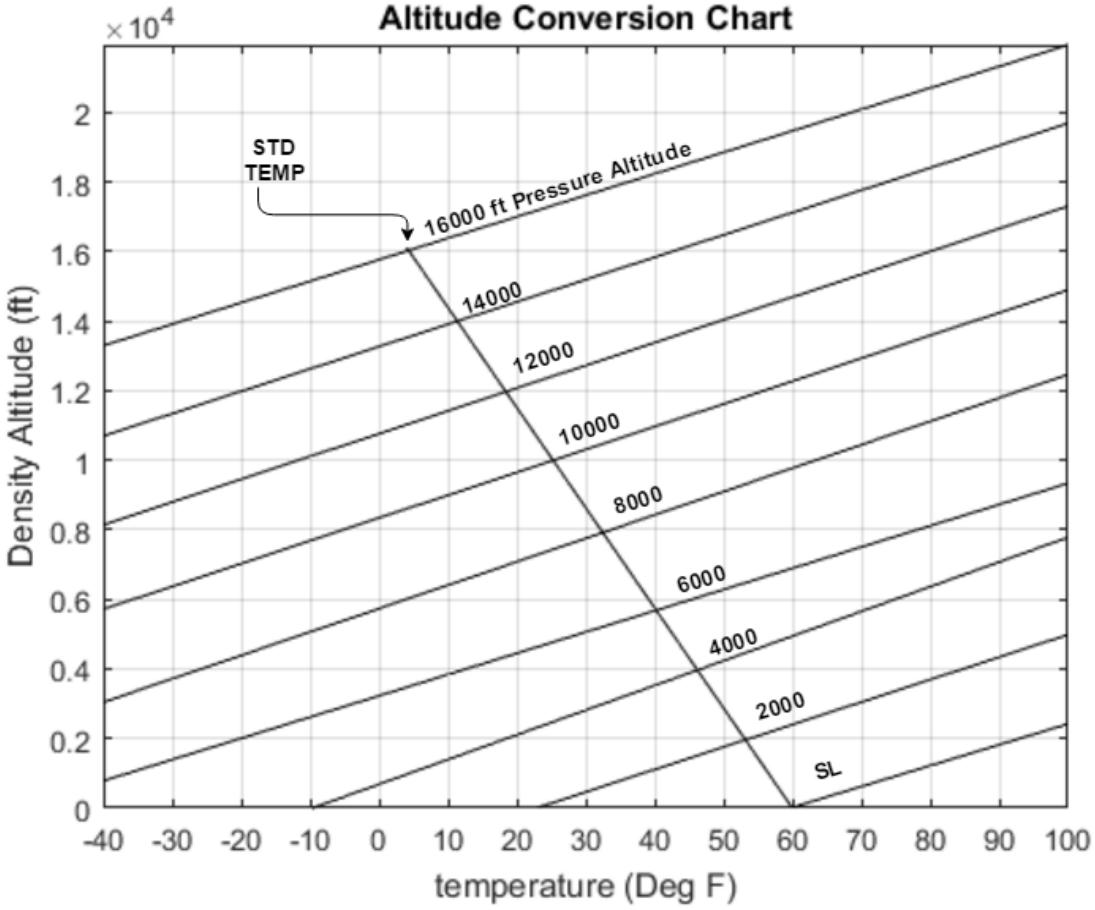
True Airspeed vs. Density Altitude

Range vs. Density Altitude

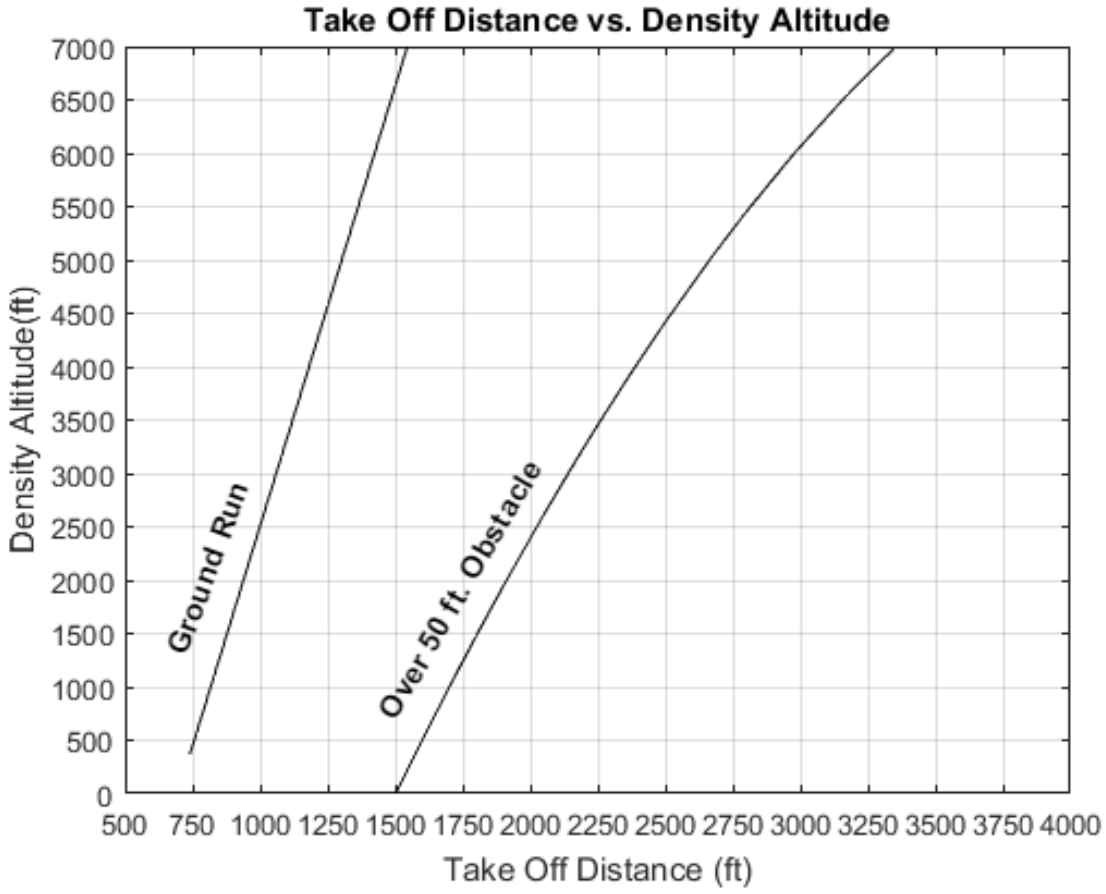
Power vs. Density Altitude

Landing Distance vs. Density Altitude

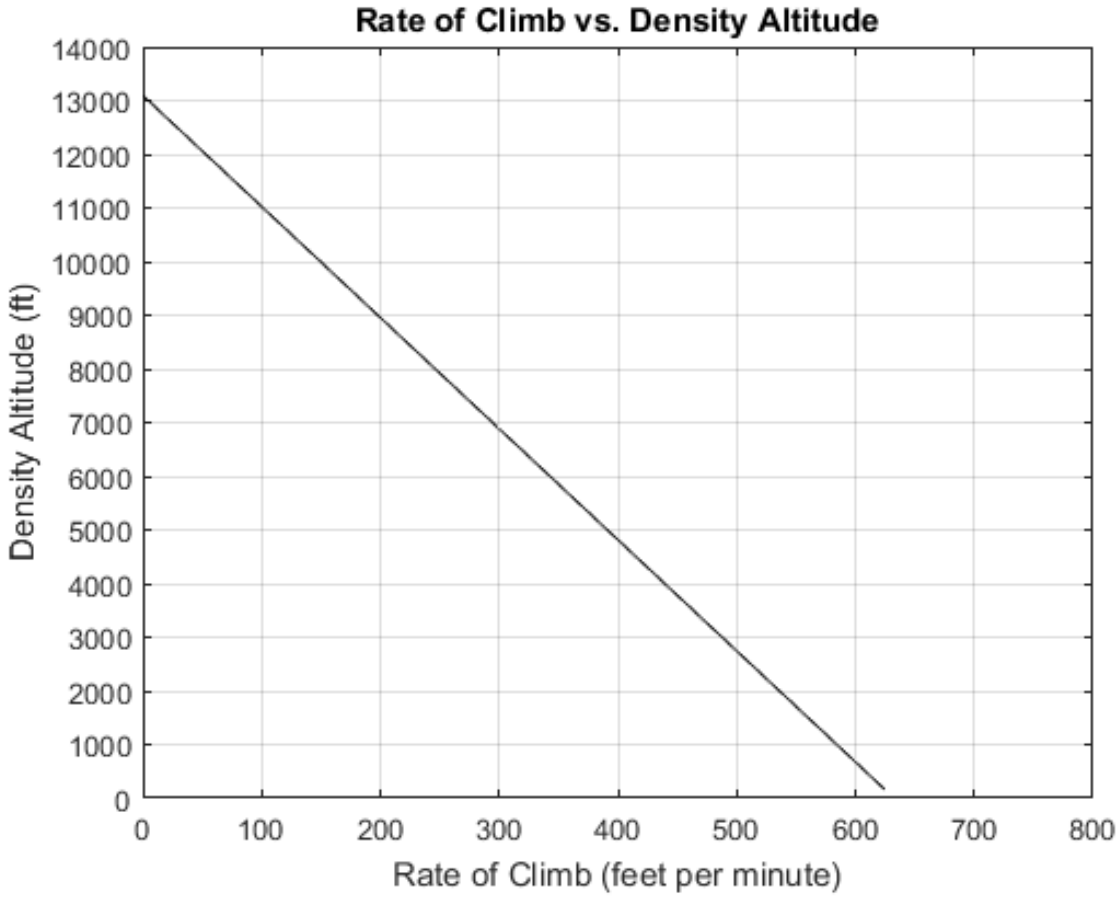
Glide Distance vs. Altitude



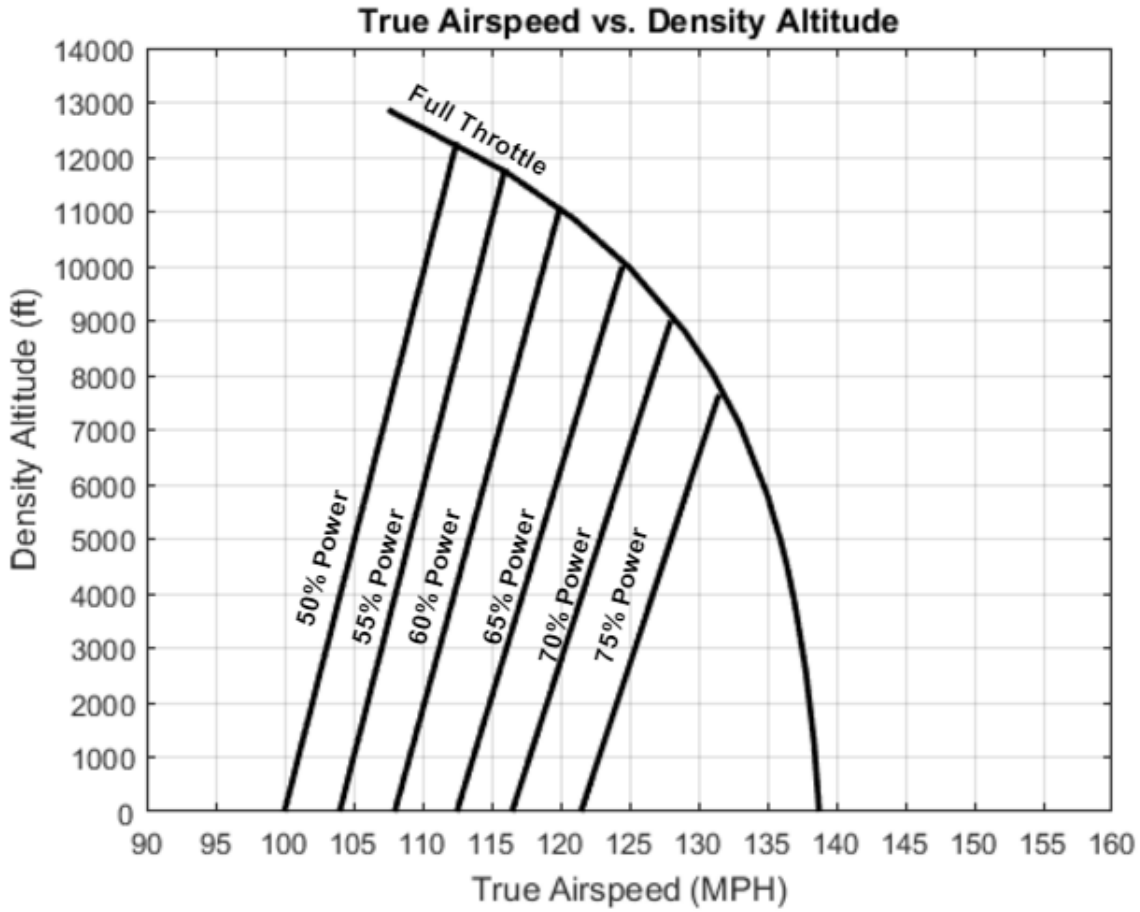
This chart should be used to determine density altitude from existing temperature and pressure altitude conditions for use with performance charts

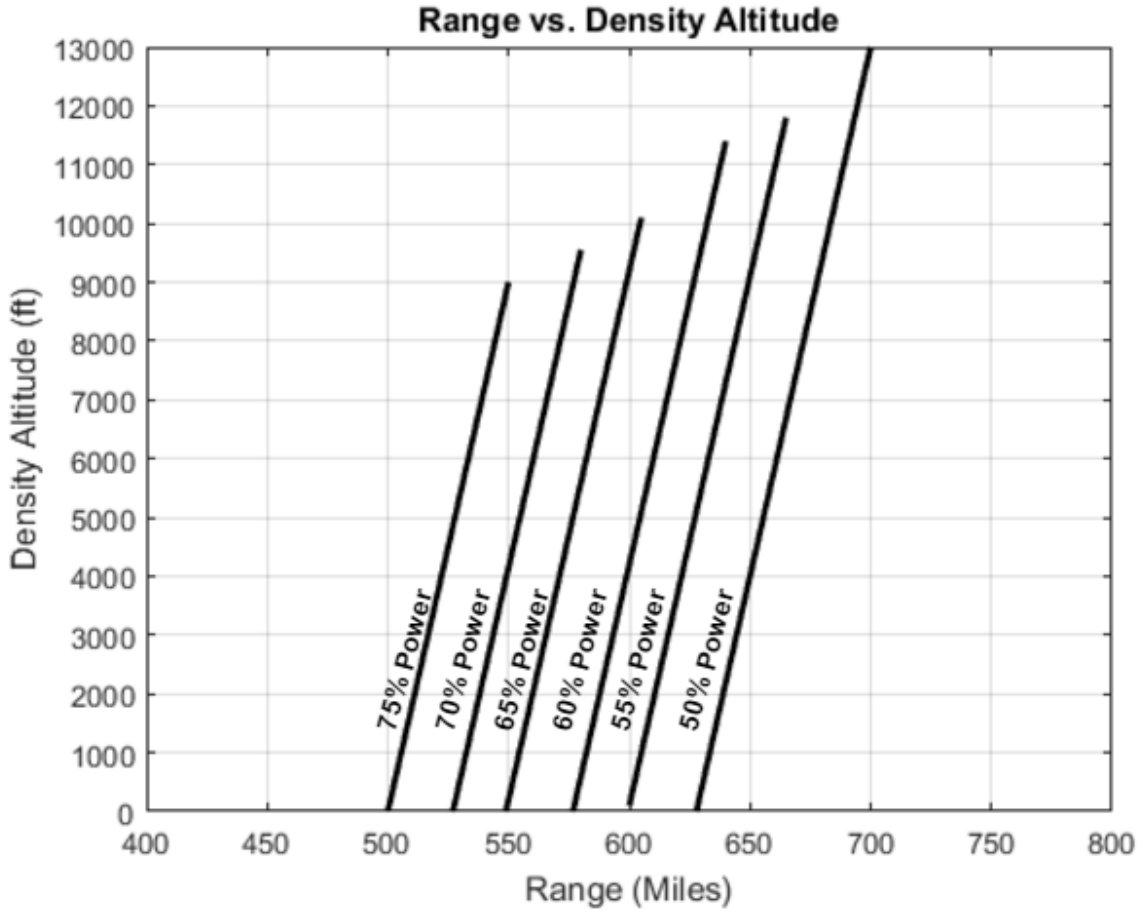


1950 lbs gross weight
Extrapolation of this chart above 7000 ft is invalid.

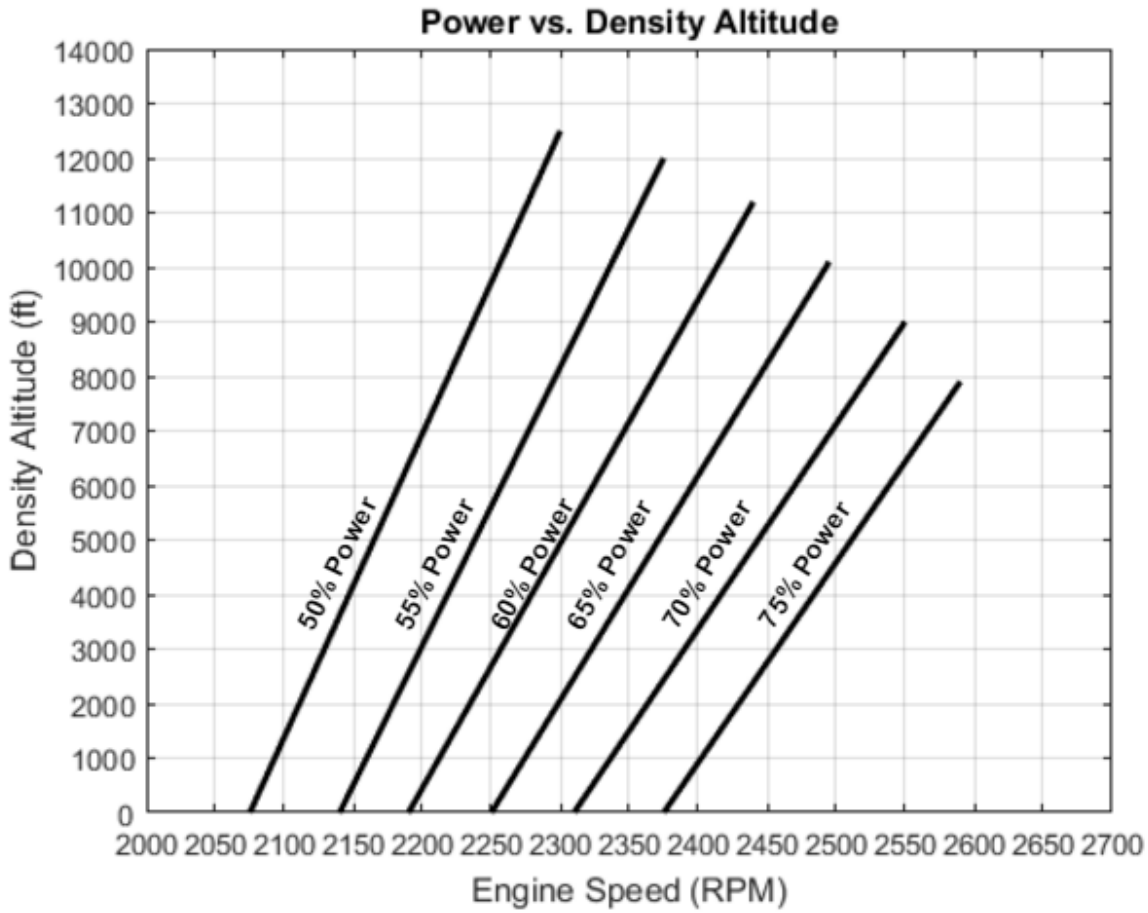


Power Full – Throttle

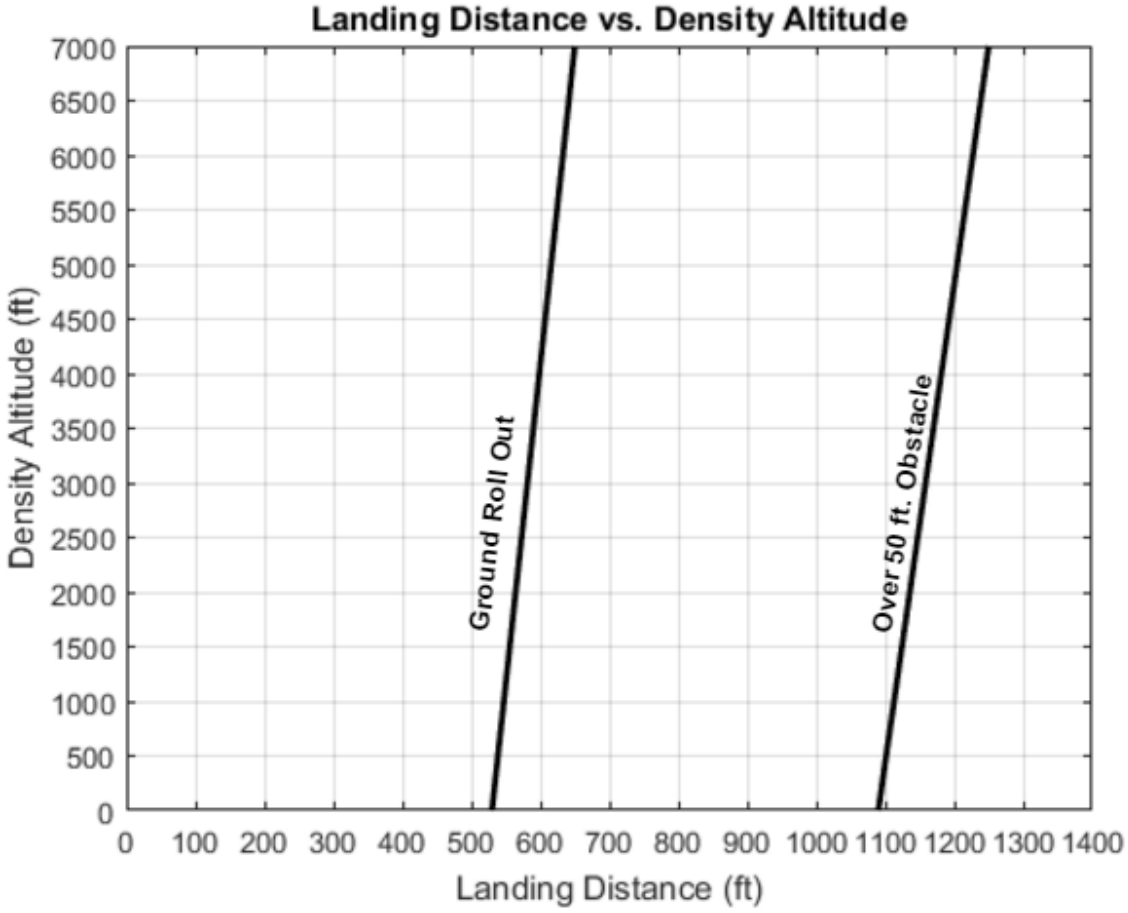

True Airspeed vs. Density Altitude
Density Altitude (ft)
True Airspeed (MPH)
Full Throttle
50% Power
55% Power
60% Power
65% Power
70% Power
75% Power
14000
13000
12000
11000
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160



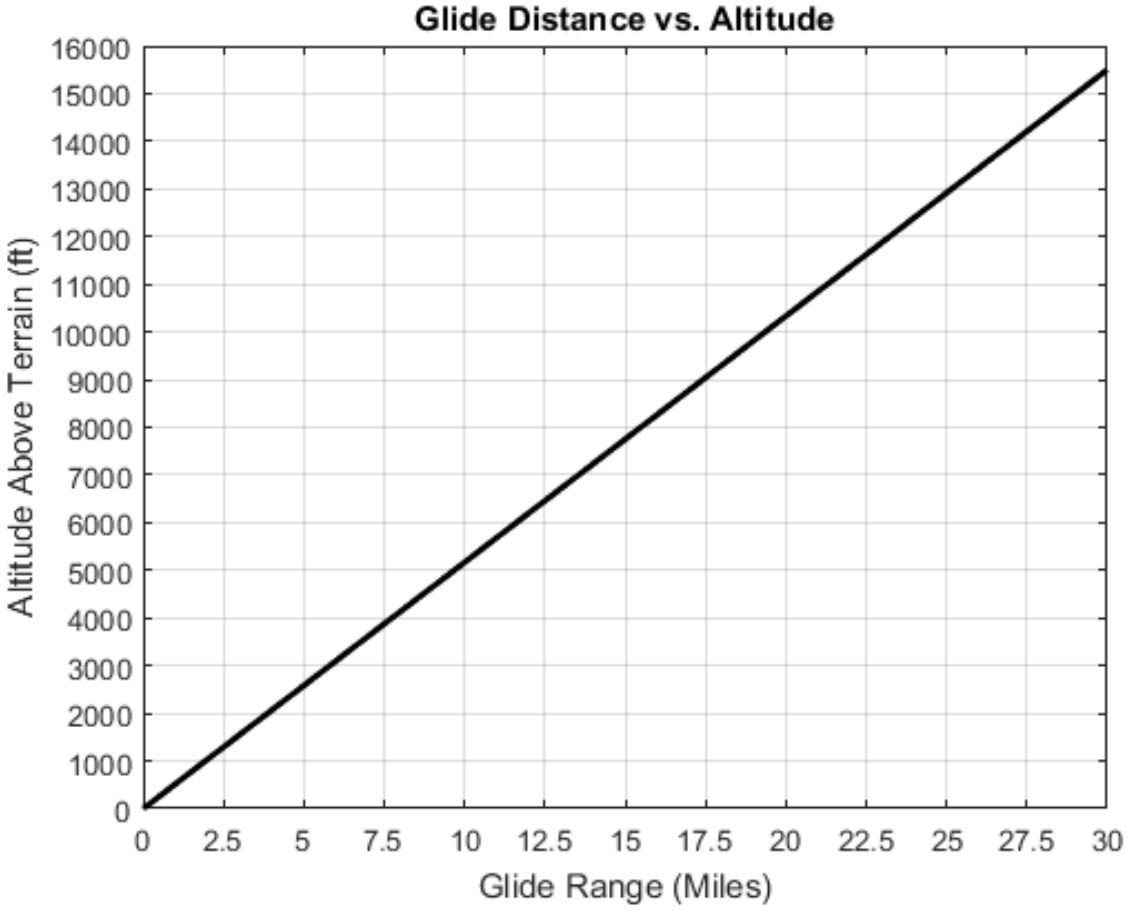
Using a standard Fuel amount of 36 Gallons



50% Power – 75 hp – 5.6 GPH
55% Power – 82 hp – 6.2 GPH
60% Power – 90 hp – 6.7 GPH
65% Power – 97 hp – 7.3 GPH
70% Power – 105 hp – 7.9 GPH
75% Power – 112 hp – 8.4 GPH



Flaps 40 degrees, Power off, Paved Level Dry Runway, No Wind, Maximum Braking, Short Field Effort, Gross Weight 2150 lbs



2150 lbs
03 MPH
Prop Windmilling
0 degree of flaps
No wind

Section VI

General Maintenance

Tire Inflation

Battery Service

Brake Service

Landing Gear Service

Fuel and Oil Requirements

Care of Air Filter

Care of Windshield and Windows

Leveling and Rigging

Serial Number Plate

Robotic Components

Section VI

General Maintenance

This section contains information for minor maintenance of the airplane. For further maintenance assistance refer to the Cherokee 140 Service Manual. Any complex repairs or modifications should be accomplished by a Piper Certified Service Center or equivalent.

Tire Inflation

For maximum service from the tires, inflate them to the proper pressure of 24 pounds for all three wheels. Interchange the tires on the main wheels, if necessary, to produce even wear. All wheels and tires are balanced before original installation, and the relationship of the tire, tube, and wheel should be maintained, if at all possible. Out of balance wheels can cause extreme vibration on take-off. In the installation of new components, it may be necessary to rebalance the wheel with the tires mounted.

Battery Service

The 12 volt battery is located in a stainless steel container under the baggage compartment floor. This container should be drained occasionally by opening the rubber cap on the drain tube. Check the battery for proper fluid level (below the baffle plates) and use a hydrometer to determine the density of the battery fluid.

If the battery is discharged, charge it before take-off as three volts are needed to excite the alternator. To recharge, start at a 4 ampere rate and finish with a 2 ampere rate. Quick charges are not recommended.

Brake Service

The brake system is filled with MIL-H-5606 (petroleum base) hydraulic fluid. This should be checked at every 100 hour inspection and replenished when necessary by filling the brake reservoir on the firewall to the indicated level. If the system as a whole has to be refilled with fluid, this should be done by filling with the fluid under pressure from the brake end of the system. This will eliminate air from the system as it is being filled.

If after extended service the brake blocks become worn excessively, they are easily replaced with new segments. Brake clearances require no adjustments.

Landing Gear Service

Main wheels are easily removed by taking off the hut cap, axle hut, and the two bolts holding the brake segment in place, after which the wheel slips easily from the axle.

Tires are removed from the wheels by first deflating the tire, removing the three through bolts, and separating the wheel halves.

Landing gear oleo struts should be checked for proper strut exposures and fluid leaks. The required extensions for the strut when under normal static load (empty weight of airplane plus full fuel and oil) is 3.25 inches for the nose gear and 4.50 inches for the main gear. Should

the strut exposure be below that required, it should be determined whether air or oil is required by first raising the airplane on jacks. Depress the valve core to allow air to escape from the strut housing chamber. Remove filler plug and slowly raise the strut to full compression. If the strut has sufficient fluid it will be visible up to the bottom of the filler plug and will then only require proper inflation.

Should fluid be below the bottom of the filler plug hole, oil should be added. Replace the plug with valve core removed, attach a clear plastic hose to the valve stem of the filler plug and submerge the other end in a container of hydraulic fluid (MIL-H-5606). Fully compress and extend the strut several times thus drawing fluid from the container and expelling air from the strut chamber. To allow fluid to enter the bottom chamber of the main gear strut housing, the torque link assembly must be disconnected to let the strut be extended a minimum of 10 inches. (The nose gear torque links need not be disconnected.) Do not allow the strut to extend more than 12 inches. When air bubbles cease to flow through the hose, compress the strut full and again check fluid level. Reinstall the valve core and filler plug, and the main gear torque links if disconnected. With fluid in the strut housing at the correct level, attach a strut pump and to the air valve and inflate the oleo strut to the correct height when the airplane is on the ground.
In jacking the airplane for landing gear or other servicing use two hydraulic jacks and a tail stand. Place 350 pounds of ballast on the base of the tail stand before jacking up the airplane. The hydraulic jacks should be placed under the jack points on the bottom of the wing and the airplane should be jacked up until the tail skid is at the correct height to attach the tail stand. With the tail stand attached continue raising the airplane to the desired height.

Fuel and Oil Requirements

Aviation grade 80/87 Octane (minimum) fuel must be used in this aircraft. The use of lower grades can cause serious engine damage in a very short period of time, and is considered of such importance that the engine warranty is invalidated by such use.

The oil capacity of the Lycoming O-320 series engines is 8 quarts and the minimum safe quantity is 2 quarts. It is recommended that the oil be changed every 50 hours or sooner under favorable conditions. The following grades are recommended for the specified temperatures:

Average Ambient Air Temperature For Starting	Single Viscosity Grade	Multi - Viscosity Grades
Above 60˚ F	SAE 50	SAE 40 or SAE 50
30˚ to 90˚ F	SAE 40	SAE 40
0˚ to 70˚ F	SAE 30	SAE 40 or 20W-30
Below 10˚ F	SAE 20	SAE 20W-30

Care of Air Filter

The carburetor air filter must be cleaned at least once every 50 hours. Under extremely adverse conditions of operation it may be necessary to clean the filter daily. Extra filters are inexpensive and a spare should be kept on hand and used as a rapid replacement.

The filter manufacturer recommends that the filter be tapped gently to remove dirt particles. Do not blow out with compressed air.

Care of Windshield and Windows

A certain amount of care is needed to keep the plexiglass windows clean and unmarred. The following procedure is recommended:

1. Flush with clean water and dislodge excess dirt, mud, etc. with your hand.
2. Wash with mild soap and water. Use a soft cloth or sponge, do not rub.
3. Remove oil, grease, or sealing compounds with a soft cloth and kerosene.
4. After cleaning, apply a thin coat of hard polishing wax. Rub lightly with a soft cloth.
5. A severe scratch or mar may be removed using jeweler's rouge to rub out the scratch, smoothing, and then applying wax.

Leveling and Rigging

Leveling the aircraft for purposes of weighing or rigging is accomplished as follows:

1. Partially withdraw two machine screws located immediately below the left front side window. These screws are leveling points and the airplane is longitudinally level when a level placed on the heads of these screws indicates level.
2. To put the airplane in a longitudinally level position on scales, first block the main gear oleos in the fully extended position, then deflate the nose wheel tire until the proper attitude is obtained. For rigging only, the airplane may be placed on jacks for leveling.
3. To level the airplane laterally, place a level across the baggage compartment floor along the read bulkhead

Rigging: although the fixed flight surfaces cannot be adjusted for rigging purposes, it may be necessary upon occasion to check the position of these surfaces. The movable surfaces all have adjustable stops, as well as adjustable turnbuckles on the cables or push-pull tubes, so that their range of travel can be altered. The positions and angular travels of the various surfaces are as follows:

1. Wings: 7° dihedral, 2° washout
2. Stabilator travel: 18° up, 2° down, tolerance +/- 1°
3. Fin should be vertical and in line with the center of fuselage
4. Aileron travel: 30° up, 15° down, tolerance +/- 2°
5. Flap travel: 10°, 25°, 40°, tolerance +/- 2°
6. Rudder travel: 27° right and left, tolerance +/- 2°
7. Stabilator tab travel: 3° up, 12° down, tolerance +/- 1°

Cable tensions for the various controls are as follows

- Rudder: 40+/-5 lbs
- Ailerons: 40+/-5 lbs
- Flaps: 10+/-1 lb
- Stabilator: 40+/-5 lbs
- Stabilator Trim: 10+/-1 lb

Serial Number Plate

The serial number plate is located near the stabilator on the left side of the airplane and also at the cabin entrance. Refer to this number for service or warranty matters.

Robotic Components

No serviceable parts. Do not attempt any maintenance or servicing on the additional robotic components of this experimental aircraft. Any and all servicing on these parts should be done by the appropriate specialist.

EXPERIMENTAL PA-28-140 CHECKLISTS

Note:

The Information provided in this booklet has been edited from a Piper 1973 Cherokee Cruiser Checklist. It is not to be used for commercial purposes and has been created as an example for the experimental aircraft for autonomous flight made from a Piper Cherokee, PA-28-140, with the tail number N15726 for the Autonomous Cargo Aircraft Project at the Worcester Polytechnic Institute. I do not claim rights not authorship to information provided for the manual flight specifications, designs, controls, manual procedures, or any information that can be directly quoted from the Piper 1973 Cherokee Cruiser Checklists.

Table of Contents

Speeds for Normal Operation

V_x	74 MPH	= Best Angle of Climb Speed
V_y	85 MPH	= Best Rate of Climb Speed
$V_{best\ l/d}$	83 MPH	= Best Glide (lift/drag)
V_s	64 MPH	= Stall Speed, Normal Configuration
V_{so}	55 MPH	= Stall Speed, Landing Configuration
V_{fo}	115 MPH	= Maximum Flap Extension Speed
V_a	129 MPH	= Maneuvering Speed (at Gross Weight)
V_{no}	140 MPH	= Maximum Structural Cruising Speed
V_{ne}	171 MPH	= Never Exceed Speed

100 MPH = Normal Climb Out

74 MPH = Short Field Take Off, Flaps 25˚

85 MPH = Normal Landing Approach, Flaps Up

76 MPH = Normal Landing Approach, Flaps 40˚

76 MPH = Short Field Approach, Flaps 40˚

Emergency Procedures

Disengaging Autonomous System

1. Input to Safety Officer Board to disconnect all automatic systems
2. If Safety Officer Board Fails
    a. Push Safety Officer Switch to BOTTOM
    b. Release to NEUTRAL
    c. Pitch Override – pull string
    d. Roll Override – pull string
    e. Yaw Override – pull string
    f. Flaps – manual force
    g. Mixture – manual force
    h. Throttle – manual force
    i. Rudder Trim – manual force
    j. Stabilator Trim – manual force
3. If clutch removal fails
    a. Pitch Override – pull gear pin
    b. Roll Override – pull gear pin
    c. Yaw Override – pull gear pin

Engine Failure During Take Off

1. Disconnect automatic system
2. Throttle – IDLE
3. Brakes – APPLY
4. Mixture – IDLE CUT-OFF
5. Ignition Switch – OFF
6. Master Switch – OFF

Engine Failure Immediately After Takeoff

1. Disconnect automatic system
2. Airspeed - MAINTAIN SAFE AIRSPEED
3. Fuel Selector Valve - SWITCH TANK
4. Electric Fuel Pump - ON
5. Mixture - RICH
6. Carburetor Heat – ON

Engine Failure During Flight

1. Disconnect automatic system
2. Airspeed – 83 MPH
3. Carburetor Heat – ON
4. Fuel Selector Valve – SWITCH TANKS
5. Mixture – RICH
6. Master Switch – ON
7. Auxiliary Fuel Pump – ON
8. Ignition Switch – BOTH

9. Primer – IN AND LOCKED

Note:
If engine failure was caused by fuel exhaustion, power will not be regained after tanks are switched until empty fuel lines are filled, which may require up to ten seconds.

Landing Without Engine Power

1. Disconnect automatic system
2. Airspeed – 83 MPH
3. Radio – MAYDAY on 121.5 MHz
4. Mixture – IDLE CUT-OFF
5. Fuel Selector Valve – OFF
6. Ignition Switch - OFF
7. Master Switch – OFF
8. Door – OPEN

Touchdown should be made at lowest possible airspeed with full flaps.

Spins

1. Disconnect automatic System
2. Throttle - IDLE
3. Rudder – FULL OPPOSITE TO ROTATION
4. Control Wheel – FULL FORWARD
5. Rudder – NEUTRAL WHEN ROTATION STOPS
6. Control Wheel – REGAIN LEVEL FLIGHT

Electrical Fire in Flight

1. Disconnect automatic system
2. Master Switch - OFF
3. Vents – OPEN
4. Cabin Heat and Air – OFF
5. Land – AS SOON AS POSSIBLE

Engine Fire in Flight

1. Disconnect Automatic System
2. Fuel Selector Valve – OFF
3. Throttle – CLOSED
4. Mixture – IDLE CUT-OFF
5. Cabin Heat and Air – OFF
6. Defroster – OFF
7. Forced Landing – EXECUTE

Engine Fire During Start on Ground

1. Cranking – CONTINUE

If Engine Starts

1. Power – 1800 RPM for a few minutes
2. Engine – SHUTDOWN AND INSPECT
3. Fuel Selector – OFF
4. Master Switch – OFF
5. Ignition Switch – OFF

If Engine Fails to Start

1. Passengers – EVACUATE
2. Engine –SECURE
3. Mixture – IDLE CUT-OFF
4. Master Switch – OFF
5. Ignition Switch – OFF
6. Fuel Selector Valve – OFF
7. Fire – EXTINGUISH

Preflight Inspection

Cockpit

1. Required Documents – AVAILABLE
2. Control Lock – RELEASE
3. Flight Controls – FREE AND CORRECT
4. Mixture and Throttle – IDLE CUT OFF
5. Master Switch – ON
6. Fuel Quantity Gauges – CHECK
7. Lights – ON
8. Stall Warning – CHECK
9. Master Switch – OFF
10. Ignition Switch – OFF
11. Flaps – DOWN

Right Wing

1. Flap – SECURE AND UNDAMAGED
2. Aileron – FREEDOM OF MOVEMENT
3. Wing Tip and Light – UNDAMAGED
4. Aileron Counterweight – UNOBSTRUCTED
5. Tiedown – REMOVED
6. Fuel Tank – CHECK LEVEL, CAP SECURE
7. Tank Drain – DRAIN – SECURE
8. Fuel – PROPER COLOR, NO WATER

9. Landing Gear – UNDAMAGED
10. Tire – PROPERLY INFLATED
11. Chocks – REMOVED

Nose

1. Windshield – CLEAN AND UNDAMAGED
2. Engine Oil Level – 2 QTS MIN, 8 QTS MAX
3. Engine Oil Dipstick – SECURED
4. Propeller and Spinner – SECURE AND UNDAMAGED
5. Cowling – SECURE AND UNDAMAGED
6. Landing Light – SECURE AND UNDAMAGED
7. Nose Gear – UNDAMAGED
8. Tire – PROPERLY INFLATED
9. Tow Bar – REMOVED AND STOWED
10. Chocks – REMOVED
11. Alternator Belt – PROPER TENSION
12. Fuel Sump Drain – DRAIN, SECURE

Left Wing

1. Fuel Tank – CHECK LEVEL, CAP SECURE
2. Tank Drain – DRAIN, SECURE
3. Fuel – PROPER COLOR
4. Landing Gear – UNDAMAGED
5. Tire – PROPERLY INFLATED
6. Chocks – REMOVED
7. Wing Tip and Light – UNDAMAGED
8. Aileron Counterweight – UNOBSTRUCTED
9. Tiedown – REMOVED
10. Pitot Mast – UNDAMAGED
11. Aileron – FREEDOM OF MOVEMENT
12. Flap – SECURE AND UNDAMAGED

Empennage

1. Elevators – FREEDOM OF MOVEMENT
2. Rudder – FREEDOM OF MOVEMENT
3. Trim Tabs- SECURE, UNDAMAGED
4. Tail Cone and Light – SECURE, UNDAMAGED
5. Tie Down- REMOVED

Before Starting Engine

1. Preflight Inspection – COMPLETE
2. Flaps – UP
3. Seat Belts, Shoulder Harness – ADJUSTED, LOCKED
4. Door – LATCHED
5. Radio, Autopilot, Electrical Equipment – OFF
6. Parking Brake – SET
7. Fuel Selector Valve – FULLEST TANK
8. Controls – CHECK
9. Circuit Breakers – CHECK

Engine Start (Cold)

Manual in All Cases

1. Parking Break – SET
2. Carburetor Heat – OFF
3. Fuel Selector Valve – FULLEST TANK

Manual Start

1. Master Switch - ON
2. Electric Fuel Pump ON
3. Mixture - FULL RICH
4. Fuel Pressure - CHECK
5. Prime – AS REQUIRED
6. Throttle-OPEN ¼ INCH
7. Propeller – CLEAR
8. Ignition Switch – START
9. Oil Pressure – CHECK
10. Engine – WARM UP 1000 RPM
11. Radios – ON
12. Transponder – STANDBY

Automatic Start

1. Ensure magneto safety switches are grounded
2. Master Battery switch - ON
3. Junction Box main power - ON
4. Check circuit breakers on Junction Box
5. Power up the computer
6. Safety ignition switch - ON
7. Monitor gauges and systems as computer begins the process of completing items found on the manual checklist for a Normal Start

Before Taxi

1. Monitor Autonomous system as it performs the following (may also be done manually)
2. Brakes – TEST
3. Lights – AS REQUIRED
4. Radios – ON, CHECKED

Before Take Off

1. Monitor Autonomous system as it performs the following (may also be done manually)
2. Lights – AS REQUIRED
3. Flight Instruments – SET
4. Flight Controls – FREE AND CORRECT
5. Throttle – 2000 RPM
6. Magnetos – CHECK
7. Carburetor Heat – ON, CHECK, OFF
8. Engine Instruments – IN GREEN ARC
9. Vacuum Gauge – 4.9-5.1
10. Throttle – 1000 RPM
11. Trim Tab – SET
12. Mixture – FULL RICH
13. Auxiliary Fuel Pump – OFF, CHECK, ON
14. Parking Brake – RELEASE
15. Transponder – ON-ALT

Normal Take Off

1. Monitor Autonomous system as it performs the following (may also be done manually)
2. Flaps – UP
3. Carburetor Heat – OFF
4. Auxiliary Fuel Pump – ON
5. Throttle – FULL OPEN
6. Rotate – 50-60 MPH
7. Climb Speed – 85 MPH

Short Field Take Off, Obstacle Clearance

1. Monitor Autonomous system as it performs the following (may also be done manually)
2. Flaps - 25˚ (second notch)
3. Carburetor Heat – OFF
4. Auxiliary Fuel Pump – ON
5. Brakes - HOLD
6. Throttle – FULL OPEN
7. Brakes – RELEASE

8. Rotate – 55-60 MPH
9. Initial Climb Speed – 74 MPH

Climb

Normal Climb

V_x	74 MPH	= Best Angle of Climb Speed
V_y	85 MPH	= Best Rate of Climb Speed

Cruise

1. Monitor Autonomous system as it performs the following (may also be done manually)
2. Electric Fuel Pump – OFF
3. Power – 2200-2600 RPM
4. Trim Tab – SET
5. Mixture – ADJUST
6. Fuel Selector – SWITCH AS NEEDED, THIS IS MANUAL

Before Landing

1. Seats, Belts, Shoulder Harnesses – ADJUST, LOCK
2. Fuel Selector – FULLEST TANK
3. Monitor Autonomous system as it performs the following (may also be done manually)
4. Mixture – FULL RICH
5. Auxiliary Fuel Pump – ON
6. Parking Brake – OFF
7. Flaps – AS REQUIRED, BELOW 115 MPH
8. Airspeed – 85 MPH
9. Landing Light – ON

After Landing

1. Monitor Autonomous system as it performs the following (may also be done manually)
2. Flaps – UP
3. Auxiliary Fuel Pump – OFF
4. Mixture – LEAN MAX RPM

5. Landing Light – OFF
6. Carburetor Heat – OFF
7. Transponder – STANDBY

Shut Down, Securing Airplane

1. Monitor Autonomous system as it performs the following (may also be done manually)
2. Electrical Equipment – OFF
3. Mixture – IDLE CUT OFF
4. Throttle -
5. Ignition – OFF
6. Master Switch – OFF
7. Parking Brake – SET (IF REQUIRED)
8. Chocks/Tiedowns – INSTALLED
9. Parking Brake – OFF

Specifications

2150	lbs	= Maximum Gross Weight
1490	lbs	= Basic Empty Weight
444	lbs	= Payload With Standard Fuel
8.4	gph	= Fuel Flow @ 75% Power

50	gal	=	Total Fuel Capacity
36	gal	=	Standard Fuel Capacity
100LL		=	Fuel Octane Rating (Blue)
8	qts	=	Oil Sump Capacity
2	qts	=	Minimum Oil Quantity

150	hp	=	Lycoming O-320
12	V	=	Battery
3.8g		=	Positive Limit Load Factor
0g		=	Negative Limit Load Factor

Appendix B
Aircraft Integration Systems

Written By: Killian Henson

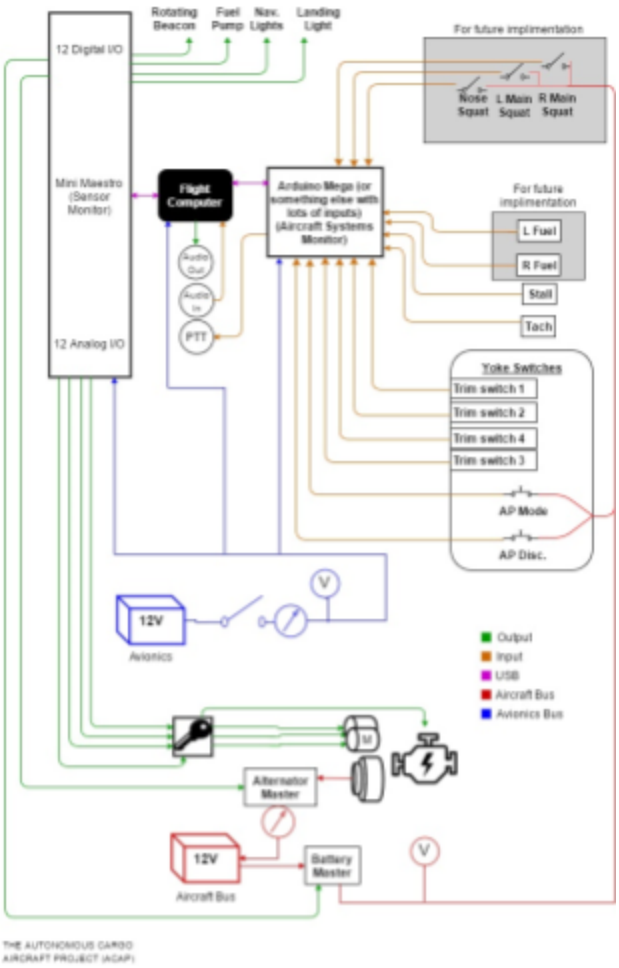


Figure A1: Aircraft Integration Hookup Chart

B1 System Description

The aircraft integration system primarily consists of existing systems which need to be tapped into to transmit and receive information to and from the flight computer. However it also includes expanding on existing systems and clearing space for new components.

Table of Contents

B2 Making the Space for Added Systems

Part of the aircraft integration was to make enough room for all of the new systems in the aircraft by removing unnecessary systems and to clean up the existing wiring. Due to safety, time, and efficiency, as much of the current working systems was left as possible. Only systems that were not needed or in the way, were moved or removed, respectively.

The first task that had to happen so that the teams could asses the space and layout behind the panel of the right side, was to remove the unnecessary avionics and those that would be moved to a different location. This included the entire Radio Stack including the Transponder, NAV/COM, Radio and Audio Panel. The transponder was moved to the top of the new space, and the rest was left open to accommodate the touch screen.

Figure B2: Original Aircraft Wiring

The second task was to fix the wiring that was in the aircraft, because when we bought the plane it was poorly completed as can be seen in figure B2. There were many wires that did not go anywhere, which is common, so that the entire system does not have to be rewired when a new instrument is installed. However with the amount of wiring and mechanical systems that needed to be added, it was determined that removing wires in favor of more usable space was preferable. Unfortunately there were more issues than just extra wires with what was there. Through removing the nav-stack and unnecessary wires it was determined that there were many cables and wires which were attached incorrectly or not connected to anything at all. One of the radios was broken when originally tested and it was discovered that the coax cable ran out of the back, around a bundle of wires and back into the same radio. Additionally there were coax cables which were wrapped around bundles and neither end was attached to anything at all. All of these were removed and the messy wiring redone to maximize space and efficiency.

B3 Instrument Air System (Pitot Static System)

The Instrument Air System provides pitot and static pressure for the Airspeed Indicator, Altimeter, and Vertical Speed Indicator. This system is used to measure the speeds and altitude of the aircraft.

This system is comprised of a pitot mast/knife which is located on the bottom of the left wing in addition to the three panel gauges and all of the tubing connecting the system. There are two holes on the pitot mast marked in figure B3; the hole marked 1 is the ram air speed and the hole marked 2 is the static air. There is tubing running from the holes on the knife to behind the instrument panel where they connect with their respective instruments as depicted in figures B4 and B5. As can be seen in figure B4 there is also an alternate static valve. This is located in the cockpit underneath the pilot's side of the panel. If the pilot suspects that the static port is not reporting correctly he/she would use the alternate valve instead.

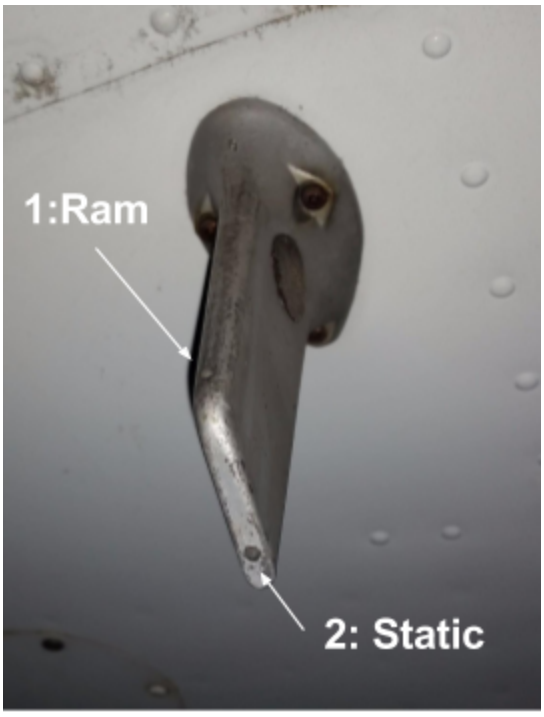


Figure B3: Pitot Knife

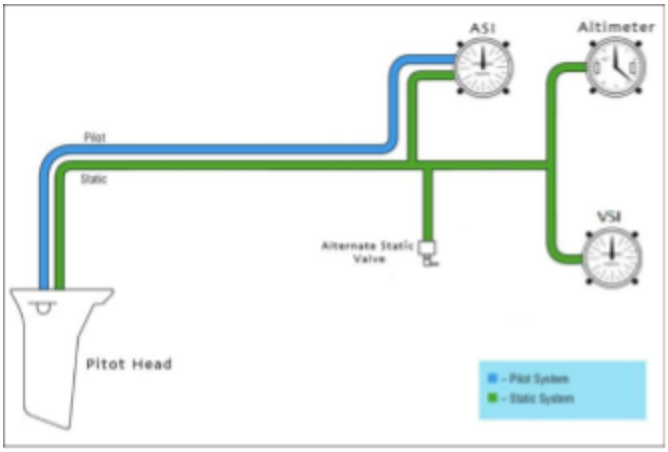
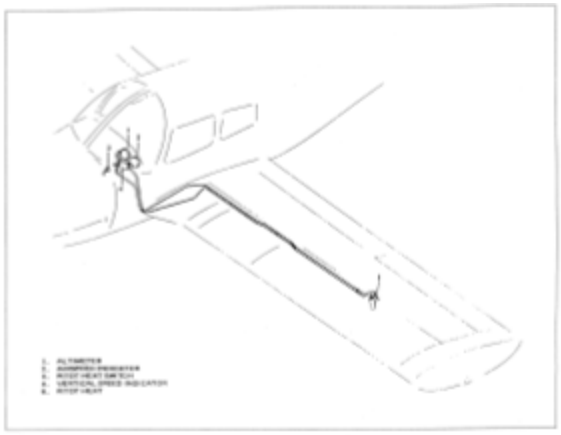


Figure B4: Instrument Air System Diagram [1]

This system is critical to instrument flight and necessary for the flight computer to fly safely. Because of the sensitive nature of this system, the utmost care had to be taken when modifying it to reduce the risk of causing a leak in the system which would cause inaccurate readings on the speed and altitude gauges.

To minimize the risk of a leak in the system each junction was sealed with either a hose clamp and/or shrink tubing. Wherever a tube met with one of the plastic plumbing joints, a hose clamp was secured around it to hold it in place. Additionally wherever tube sizes were reduced, (with or without a plastic junction), shrink tubing was sealed to reduce risk of a leak in that area. This was tested on segments of test tubing to ensure that the shrink tubing would hold once the tubing was pressurized.



B5: POH Pitot Static System Diagram [2]

The primary issue with modifying this system was the difficulty finding what parts were used in the original aircraft, because those can be assumed to be reliable for use in this system. Unfortunately, the aviation industry is set up in a way that it is somewhat hard to find parts which are known to be used in aircrafts. Aircraft Spruce is one of the few places that does have most of the parts one could want, however they are often overpriced and sometimes don't have exactly the right sizes. However, if the right product is found on their website, US plastic has more options for the same products, so to reduce shipping costs this project got tubing supplies from US plastic because they had more items which were needed and therefore reduced the shipping costs.

Below is a list of parts used to modify the Instrument Air System.

Part Number	Image	Use	URL
T1		• 1/8" NPT x 1/4" Hose ID Black HDPE Tee • Replaces Elbow, mounting onto the Mode C Altitude Encoder • Static Air System	https://www.usplastic.com/catalog/item.aspx?itemid=27811
T2		• 3/16 in Black Nylon Tee • Spliced into the tube going into the Airspeed Indicator • Ram Air System	https://www.usplastic.com/catalog/item.aspx?itemid=27626
T3		• ¼ to ⅛ in Reduction Tube • Static Air System	https://www.usplastic.com/catalog/item.aspx?itemid=35458
T4		• 3/16 to ⅛ in Reduction Tube • Ram Air System	https://www.usplastic.com/catalog/item.aspx?itemid=35456
T5		• ⅛ in Black Nylon Tubing • Goes to Differential Pressure Sensor	https://www.usplastic.com/catalog/item.aspx?itemid=119659
T6		• 5/16 to ⅝ in Worm Drive Hose Clamps • Used to clamp junctions	https://www.usplastic.com/catalog/item.aspx?itemid=32763

As seen in Figure B6 part T1 connects to the Mode C Altitude Encoder which converts static pressure to information for the transponder. This is attached at the end of the static line after the six-pack instruments. This was determined to be the best place to splice into the system as it did not require very many additions to to the line. The elbow that was previously connected was replaced with a tee (part T1), then the new line out was reduced in size, using part T3, and channeled to a differential pressure sensor which relays the static pressure to the computer system through part T5.

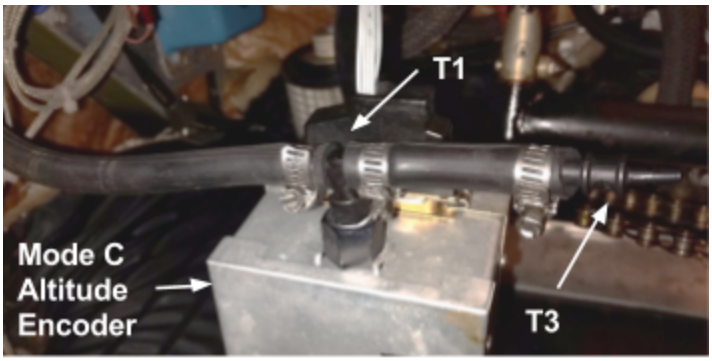


Figure B6: Static Line

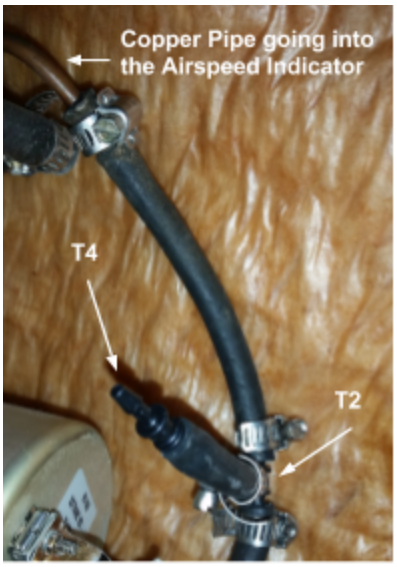


Figure B7: Ram Air Line

The ram air speed was read from just before it goes into the Airspeed Indicator. The upper left corner of Figure B7 shows the copper pipe which is connected to the Airspeed Indicator. The tube coming off of that pipe was spliced into using a tee (part T2) then reduced down to the proper size, using part T4, to read into the sensors which relayed airspeed to the computer through part T5.

B4 Stall Warning Indicators

The stall warning indicator(s) are for alerting the pilot whenever the plane is stalling. In an airplane, a stall is when the airplane's angle of attack reaches its critical angle of attack which is defined by when the angle is such that the lift on the wings decreases. This creates a situation where unless corrected for quickly, the plane will fall out of the sky without the lift needed to keep it aloft.

There is a metal tab on the left wing which is angled such that it closes a circuit when the angle of attack is equal or greater than the stall angle. This causes the light and or buzzer to activate in the cockpit. When N15726 was bought it only had a light and for safety the pilot decided the light should be replaced with a more reliable LED and that a buzzer should be added. One reason for the added buzzer is because the LED is located next to the window which can dull the effect of the light. Additionally in the case of an emergency audible warning is more reliable since the cabin crew will be monitoring other systems and knowing if the plane is going into a stall immediately is essential for a fast recovery.

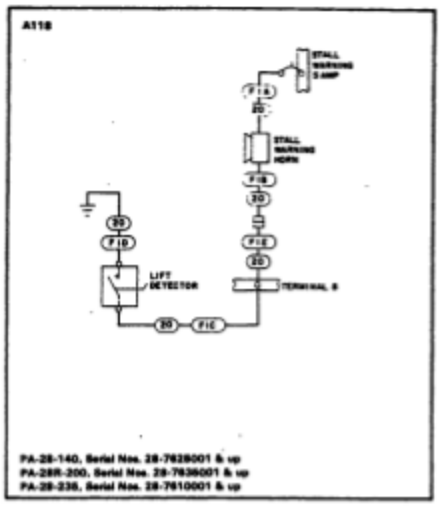


Figure B8: POH Stall Warning System [3]

The electrical system is designed to close the circuit which activates the stall warning indicators. Figure B8 depicts the circuit design in the Pilot's Operating Handbook or POH. N15726's system originally did not have the stall warning horn and instead had only a light. Our new design adds the LED and horn in parallel where figure B8 shows the stall warning horn.

There are some important features to note when testing and wiring this system. This system like many others in the plane is grounded when the master switch is off. When the master is off the master ground contactor grounds the live wire on the Stall Warning indicator. This means that if the polarity of the wiring is being tested when no power is running through the system the wires will appear reversed from what they actually are when the master is on. In the case that the system is wired backwards, it will be indicated when the master switch is on, the LED will flicker and the horn will not make any noise. Additionally if the system is tested with a battery from the aircraft ground to the computer output wire on the "power" line, it will appear to work properly. In this case the easiest way to fix the problem is to reverse the lines coming from the stall warning tab on the wing.

B5 Yoke Switches

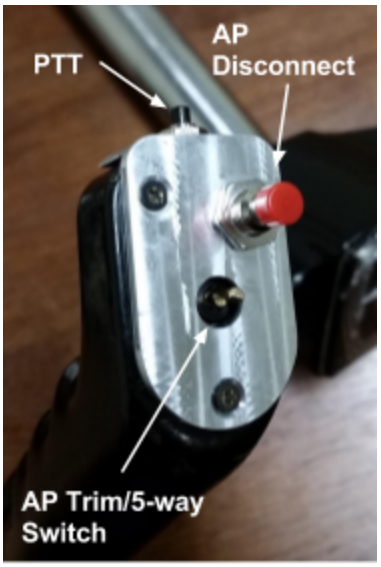


Figure B9: Yoke Switches

All aircraft have a Push To Talk (PTT) switch on the left yoke for communicating with nearby traffic and Air Traffic Control (ATC). However, for safety and convenience it was decided to add a number of switches and buttons to the yoke. In addition to the PTT switch, there is now an Autopilot (AP) mode switch, AP Disconnect/Emergency Shutoff Switch, and trim switches so that the cabin crew can nudge the AP without having to disconnect it. Additionally after switch selection there is added an encoder which can be programed for an additional function if one is deemed necessary.

The primary difficulty with selecting switches was the size constraints. Effectively six buttons needed to fit within the confines of a box approximately 1in by .75in. Additionally one of these switches needed to be perpendicular to the others to act as the PTT switch which is traditionally mounted on the front of the yoke so as to be activated by the pilot's pointer finger.

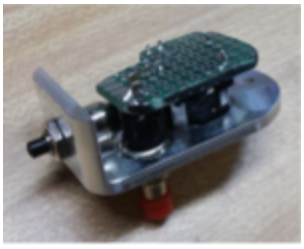


Figure B10: Switches soldered onto a protoboard and mounted

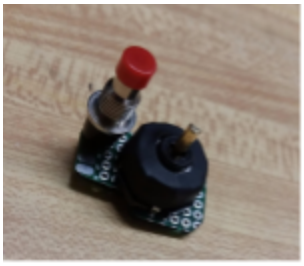


Figure B11: Shrink Wrap around the 5-way switch

Pin	Color
GND	Solid Brown
AP Disconnect	Striped Brown
Push To Talk	Solid Green
5-way Center	Striped Green
Encoder A	Solid Blue
Encoder B	Striped Blue
5-way A	Solid Orange
5-way B	Striped Orange
5-way C	Solid (dark) Orange
5-way D	Striped (dark) Orange

Table B1: Yoke Wire Color Code

Once the general positioning of the switches was determined, the next step was to design a mount which would leave the necessary room while holding all of the switches securely in the yoke. Once the mounts were made, the switches were soldered onto a protoboard cut to fit in the hole in the yoke (figure B10) and shrink wrap was used on the 5-way switch to ensure that there

would be no shorts with the exposed metal on the sides (figure B11). All of the ground wires were soldered together and wires were attached to each pin on the protoboard. Table B1 describes the color coordination for the yoke switches which go to a microcontroller.

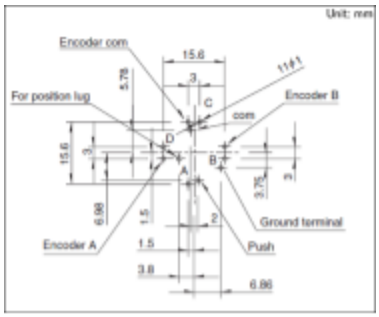


Figure B12: 5-Way Switch Schematic [4]

B6 Tachometer Circuit

In order to make sure the aircraft engine was functioning properly, the Flight Management Computer needed a way to determine the engine speed. An electronic tachometer was developed to collect this data (the tachometer circuit is shown in Figure B13). This tachometer worked by detecting the voltage spikes produced by the magnetos when firing the spark plugs. The voltage level across the primary coil of the magnetos were measured from the magneto connections on the ignition switch (connected to Mag_In in the schematic). This signal, a ~200 V decaying sinusoid (shown in figure B14), was scaled down to a logic-level signal with a high-impedance voltage divider and a Zener diode. An analog integrator and one-shot filter were then used to remove any transients and produce a single digital output pulse indicating a magneto trigger (Trigger_Out in the schematic). This output passed to a dedicated microcontroller packaged with the sensor circuit (an ATMEGA2560 on an

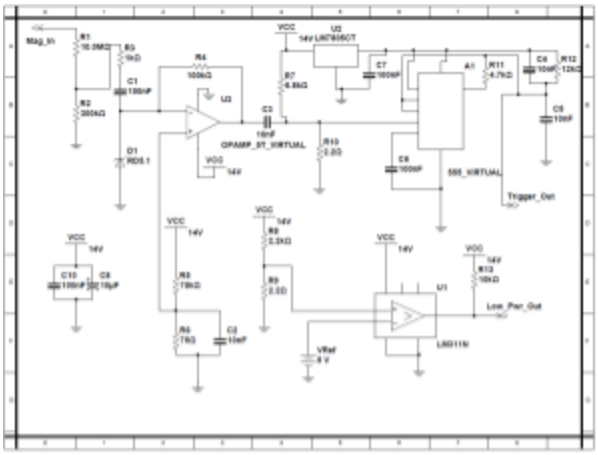


Figure B13: Tachometer Circuit

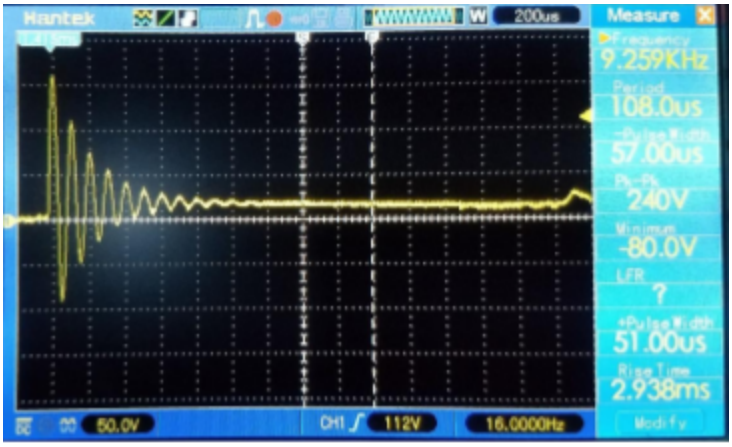


Figure B14: ~200 V decaying sinusoid on an Oscilloscope

Arduino development board), triggering a software interrupt. The microcontroller counted the number of magneto firings detected in a rolling time window and used this to calculate the engine speed. The microcontroller also tracked the total elapsed engine run time and recorded this information to non-volatile memory, allowing the total aircraft engine time to be tracked. Additional circuitry, consisting of an analog comparator, voltage divider, and high-capacity capacitor, was included to allow the microcontroller to detect a power loss and record the engine time data safely before shutting down.

The tachometer circuit proved successful in lab tests; it was able to properly detect the timing of a scaled-down voltage signal similar to that produced by the magnetos in the plane, calculate the correct engine speed, detect power losses, and record engine time to memory for continuous tracking. During initial testing in the aircraft, an unidentified issue (likely an accidental short circuit due to a loose wire) damaged the voltage regulation circuitry on the microcontroller development board. In response, the damaged board was replaced, the circuit mounted in an enclosure, and all wires were secured and properly terminated. Unfortunately, due to time constraints, final testing with the repaired circuit were not completed.

B7 Audio Intercom System

Figure B15: Audio Intercom Plug

The Audio Intercom System needed to be tapped into, so that the computer can record and eventually send messages to nearby traffic and to Air Traffic Control. First the system was continuity checked between the intercom plug (figure B15) and where each line was going to. To help with this we used figure B16 and table B2 from the intercom operating instructions. Once the system's integrity was verified we connected the computer to the needed ports for it to listen and transmit. To do this the computer listens to the intercom through the back seat passenger's headphone jack. To transmit it triggers the PTT and "talks" through the hand mic jack which is located under the center of the panel. For messages to the cabin crew, it outputs audio to the overhead speaker via an external audio amplifier.

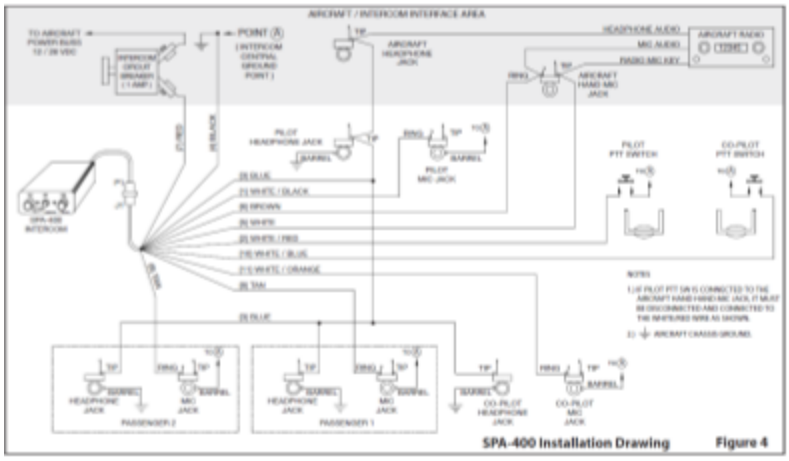


Figure B16: Intercom Schematic [5]

Table 1

Plug Pin #	Wire Color	Function	Connect To:
1	White / Black	Pilot Mic Input	Ring Terminal of Pilot Intercom Mic Input Jack
2	White / Red "A	Pilot Transmit Switch Input	Pilot Transmit Switch (PTT) (Switch to Ground to Transmit)
3	Blue "B	Headphone - Radio and Intercom Outputs	Radio Headphone Output and all Headphone Jacks
4	Black "C	Ground	Hand Mic Jack Ground Terminal (Post A)
5	White "D	Transmit Relay (Key) Control Output	Tip Terminal of Aircraft Mic Jack or Key Input of Aircraft Radio or Audio Panel
6	Brown	Mic Audio Output	Ring Terminal of Aircraft Mic Jack or Input of Aircraft Radio or Audio Panel
7	Red "E	12 VDC through 24 VDC Power Input	Intercom Circuit Breaker
8	Tan "F	Passenger #1 Mic Input	Ring Terminal of Passenger #1 Mic Jack
9	Tan "F	Passenger #2 Mic Input	Ring Terminal of Passenger #2 Mic Jack
10	White / Blue	Copilot Transmit Switch	Copilot Transmit Switch (PTT)
11	White / Orange	Copilot Mic Input	Ring Terminal of Intercom Copilot Mic Input Jack
12			No Connection

* See Wiring Instructions

Table B2: Intercom Color Code [5]

B8 Aircraft Lights

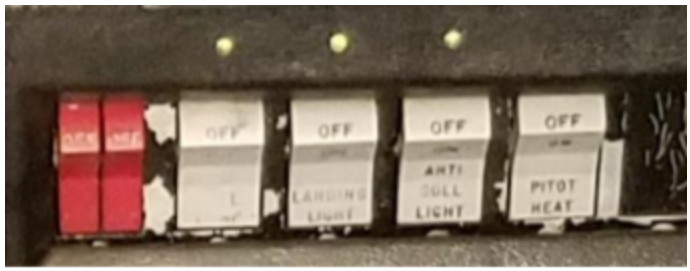


Figure B17: LEDs over Master Switch panel

The aircraft lights and fuel pump have to be modified to allow their status to be read and changed by the flight computer. This involves a critical-on manual override, so that if the switch is manually turned on, the computer can not turn it off, but whilst the switch is off, the computer can control it's status. This design was chosen after talking with the pilot, for safety purposes. With these types of switches it is most important that they are not turned off in flight, however having them turned on unexpectedly will cause minimal problems.

The application of this was very simple. The wires which go to the computer were attached to the switch output. This allows the computer to run power directly to the output, bypassing the switch. If the switch is on, the computer can not keep the power from flowing to the output. The master switch is purposefully not wired in this way because that is the only thing keeping the plane from being able to fully start itself which would be incredibly dangerous.

For each of the switches which the computer can control, there is also an LED wired from the power to ground. This will allow the cabin crew to know the actual current status of each circuit regardless of the physical switch status.

B9 Additional Lights and Sensors


Figure B18: Panel Lights and Switches

There are additional lights and sensors in the plane to monitor a variety of conditions.

The indicator panel, shown in figure A18, has five lights which can be programed to indicate different errors or notifications such as illuminating whenever the radio is transmitting. It also has three double throw-double pull switches with an off position. They are ON-Off- Momentary switches which are for disengaging the three primary surface controller motor clutches.

Additionally the Barometer and Outside Air Temperature was measured on one device located in the tail to measure outside pressure and air temperature.

The team also added an additional GPS unit located in the forward avionics bay. This is for the computer to know its location because it can't easily interface with the GPS in the plane.

B10 For Future Implementation

B10.1 Landing Gear Squat Switches

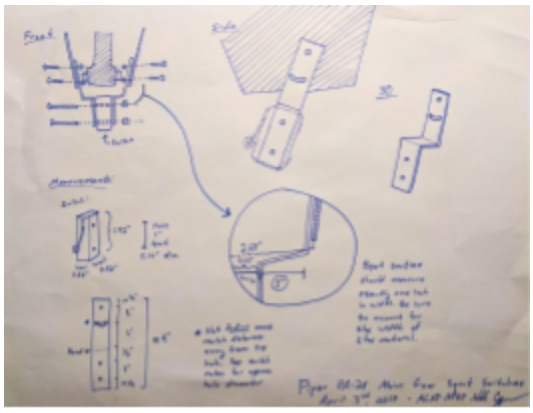


Figure B19: Squat Switch Mount Design

There will be three squat switches on the landing gear to monitor if each gear is in contact with the ground. This will be accomplished by the ME team mounting them so that when the shocks compress, the switches are triggered. The wires will need to be protected and run from the switches up through a hole in

either the wing or body of the aircraft for the main gear. The nose gear squat switch will have to be protected against heat and carefully placed, as it will run up into the engine compartment and through one of the premade holes in the firewall.

B10.2 Fuel Tank Indicators

Figure B20: Fuel Gauges

The Left and Right Fuel Quantity Transmitters (figure B20) are mounted inside of the fuel tanks. Their outputs go through the wings of the aircraft, to the left side of the cabin and run forward to the Fuel Level Indicators. These are electrical signals coming from the ohmmeter connected to the Transmitters. This allows direct interception of the signal for input to the computer. Figure B21 from the PA-28 Service Manual gives resistances which will allow for verification of system integrity and easy computer integration.

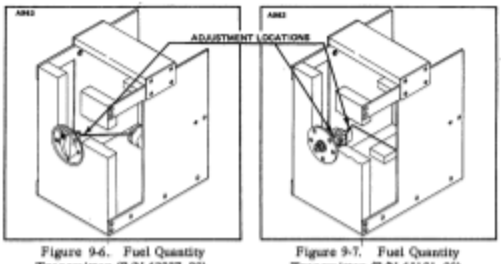


Figure 9-6. Fuel Quantity Transmitter (P/N 62037-00)

Figure 9-7. Fuel Quantity Transmitter (P/N 66101-00)

TABLE IX-II. FUEL QUANTITY TRANSMITTER CALIBRATION TOLERANCES

UNIT	POSITION	RESISTANCE
P/N 62037-00 (Metal Float)	Empty	.00 to 0.5 ohms
	Full	29.6 to 31.3 ohms
P/N 66101-00 (Rubber Float)	Empty	240 + 20, -0 ohms
	Full	33.5 + 0, - 4.5 ohms

Figure B21: POH Fuel Quantity Transmitter [6]

Works Cited

[1]"Pitot-Static System - N476DS", *Flygerry.com*, 2017. [Online]. Available: http://www.flygerry.com/aircraft-systems/pitot-static-system. [Accessed: 20- Apr- 2017].

[2]”The Cherokee Cruiser and Flight Liner Pilot’s Operating Manual”, Piper, 1973, pp. 2-13.

[3]”Piper Cherokee Service Manual”, Piper, 1995, pp. 4H23.

[4]"RKJXT1F Series - Basic information", *Alps.com*, 2017. [Online]. Available: http://www.alps.com/prod/info/E/HTML/MultiControl/Switch/RKJXT/RKJXT1F42001.html. [Accessed: 18- Mar- 2017]

[5]”SPA-400 TSO Installation and Operating Manual”, Sigtronics Co., San Dimas, CA, pp. 2.

[6]”Piper Cherokee Service Manual”, Piper, 1995, pp. 3C21.

Appendix C: Power and Electronic Control Systems

Table of Contents

Overview

This appendix describes the electrical and electronic systems added to the aircraft (as opposed to those used to interact with or replace existing avionics equipment, which are discussed in Appendix B). It begins with a brief overview of the systems and their intended functionality. This is then followed by more detailed overviews of the designs of each of the three main subsystems: the actuation systems, the power distribution network, and the control hardware. The actuation systems consisted of the motors and servos used to physically move the control surfaces. The control hardware was the collection of electrical and electronic devices used process the control software and make the actuators and aircraft electronics operate as intended. The power distribution network (shown in Figure C1) included the regulation, protection, and wiring hardware needed to operate the electrical system from the aircraft power system. The appendix concludes with a discussion how the system actually performed in testing and recommendations for future work.

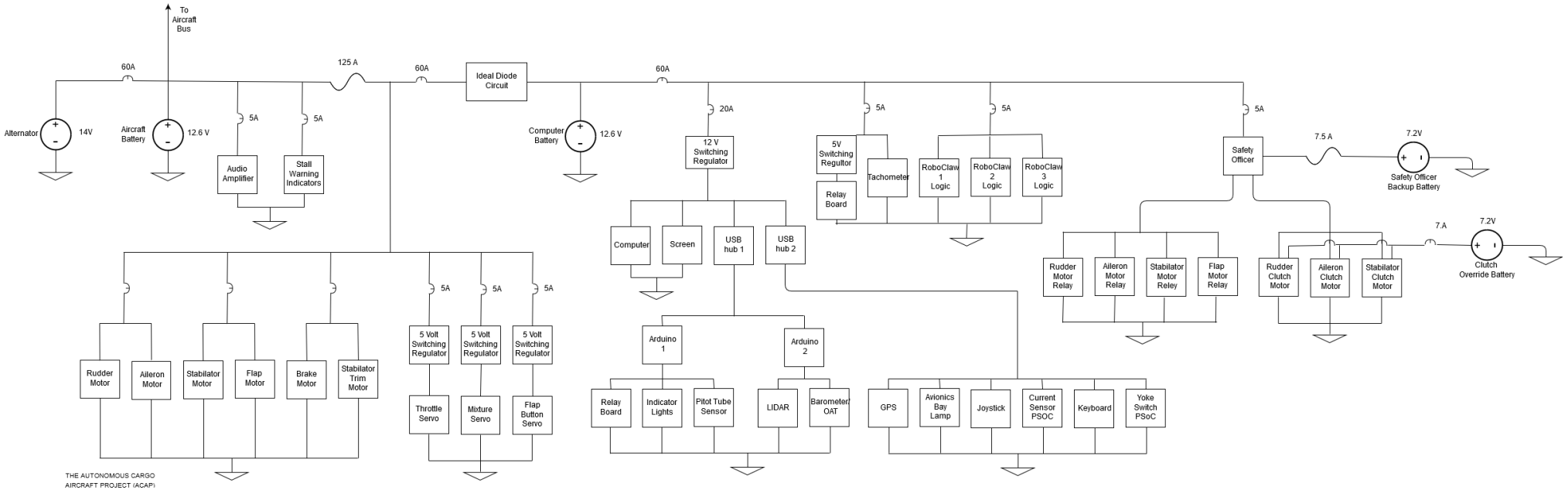


Figure C1: Power Distribution System schematic

C1. Actuation

Basic operation of the aircraft required control of at least the ailerons, stabilator, and rudder. More advanced control would have also required the flaps, throttle, and brakes. Full control would have required the mixture and trim settings. Actuation systems were designed for each of these systems except the trim wheels.

Most of the actuators used were DC electric motors. The four control surfaces (ailerons, stabilator, rudder, and flaps) were all operated by BAG motors: 12 V motors with peak power outputs of 149 W [C1] each. The brakes, which did not need to counter aerodynamic resistance, were operated by a smaller motor. The positions of the rudder, stabilator, and ailerons were monitored using VersaPlanetary Integrated Encoders: 10-bit Hall Effect quadrature encoders [C2] mounted in the motor gearboxes. The high resolution of these encoders, combined with the gear reduction from the transmission, provided a final position resolution of several thousand encoder ticks across the range of motion of the actuators, allowing for accurate speed and position control. The flaps, which did not need to be controlled as precisely, were operated through a simpler method—dead reckoning based on running the flap motor at a set speed for a set amount of time from a known starting position. Limit switches were mounted in the aircraft to disable motors if the actuators were ever driven to the end of their ranges of motion in order to prevent damage to the aircraft from faulty controls or jammed motors. The throttle and mixture levers were operated with servos. The reliable, integrated closed-loop position control provided by servos greatly simplified the operation of these controls.

C2. Control Hardware

C2.1. Flight Management Computer

The Flight Computer had to provide the maximum possible computational power in order to run the complex control theory, computer vision, and system integration operations required to operate an autonomous aircraft. It also had to operate on the minimum possible input power because of the 60 A limit on the aircraft alternator. This need for high-power, high-efficiency voltage regulation presented a difficult design challenge. Multiple designs were considered for the flight computer itself in order find the best tradeoff between power and efficiency, and multiple power supply options were considered for each possible design.

Flight Computer Layout Options

When designing the flight computer power supply, the possibility of using a laptop rather than a desktop computer was discussed as a possible way to simplify the power supply design. There were several possible advantages to using a laptop: because laptops run on batteries, they are typically designed to be extremely efficient; they are designed to receive power from a single-voltage DC power source; and their built-in batteries would allow them to continue to operate even if aircraft power were lost. However, laptops also present additional difficulties: their higher efficiency often comes at the cost of significantly lower computational power than comparable desktop equipment; the DC voltage level required to power most laptops is around 19 V, a level not common in most other applications; they are less customizable than desktops; and they are often more expensive than comparable desktops. Laptop and desktop options were both considered because of the different power supply options available for each format.

Self-Regulating DC Supplies

The first power supply options considered were commercially available power supplies capable of powering the flight computer directly from the 10 V-16 V aircraft electrical system. The prospect of such a supply was attractive because of the simplicity of using a single contained unit and the reliability of a commercially manufactured system. One such unit for powering a desktop was the PSTC-12200I power supply from PowerStream. This unit was designed to act as a standard ATX power supply when powered from a 9 V-18 V source and could provide an output of 200 W. Unfortunately, this supply and other similar supplies were relatively inefficient, with a maximum efficiency of 65%. [C3] This was well below the 85% needed to provide 200 W of output power without exceeding the computer power budget.

The laptop equivalents to such fully-contained supplies were laptop car chargers. These units were made to power laptops from a 12 V car electrical system, essentially the same type of system found in the aircraft. Unfortunately, no commercially available models were found that could provide more than 120 W, and available laptops that drew less than 120 W were far less computationally powerful than the desktop systems under consideration.

Externally Regulated DC Supplies

In addition to regulated DC supplies, unregulated DC supplies were examined. These supplies were only available for desktops and were designed to convert power from a regulated DC power source (usually 12 V) to the required ATX power levels. The most prominent such

supplies were the picoPSU line. There were several attractive features of supplies such as this, including the extraordinarily compact form factor, but by far the most notable was the efficiency: the manual for the picoPSU lists an efficiency of greater than 96%. [C4] The main disadvantage of such converters was the need for a separate voltage regulator unit to convert the 10 V-14 V aircraft power to a steady 12 V. Additionally, no converters of this type were found that were rated for more than 160 W of output power--less than the 200 W desired for powering a desktop.

AC Supplies

One option that was considered briefly was the possibility of using a power inverter to convert the 10-14 V DC aircraft power to the standard 120 V_{rms}, 60 Hz mains power level. A setup such as this would have been quite versatile, as it would have allowed the flight computer to be powered from any standard power AC power supply. Such supplies are widely available for both desktops and laptops, as they are typically the standard power source for either type of system. However, this idea was dismissed without much consideration because of concerns with the inefficiency that would be introduced by converting a 12 V DC source up to 120 V_{rms} AC and then back down to 12 V, 5 V, and 3.3 V ATX power levels or 19 V laptop power levels.

Custom Power Supplies

A final option that was considered was the possibility of designing and building a custom DC power supply. A standard laptop supply would have provided a single 19 V source and so could have been constructed as a simple boost converter. A desktop supply would have been more complex, but still possible. A standard ATX power supply provides four power levels: a +12 V rail, a +5 V rail, a +3.3 V rail, and a -12 V rail. The 5 V and 3.3 V sources could have been obtained from commercially available wide-input buck converters and the -12 V source could have been obtained from the +12 V rail with a simple 1:1 isolating module, leaving only the +12 V source, which would have required the construction of a dedicated buck-boost converter. Such a power supply could be designed to fit the required task perfectly--switching converters can achieve efficiencies in excess of 90%, and a custom supply could have been designed to provide as much power as was required. The main disadvantages of such an approach were greatly increased development time and effort and a much higher risk factor: if the converter were improperly designed, it could take weeks to construct a new converter from a corrected design.

Selected Design

A desktop setup was chosen for the flight computer design. This construct allowed for the selection of much more powerful hardware than would have been available in a similarly priced laptop. A picoPSU externally regulated desktop power supply was selected as the main power supply because of its high efficiency, along with a 200W voltage regulator rated to produce a stable 12 V output from an input voltage of 10 V-34 V; however, because the highest-power picoPSU could not provide enough power to fully operate the computer, some modifications were made. An examination of the manuals for different picoPSU models showed that the only apparent difference between the higher and lower power picoPSU supplies was the current rating of the +12 V rail. Further examination revealed that the picoPSU 12 V rail was simply a switched connection to the 12 V picoPSU supply input. Therefore, in order to provide greater maximum current, the picoPSU's +12 V rail was bypassed. The picoPSU +12 V output was used to control a relay directly connecting the flight computer +12 V input to the 12 V regulator output. Because the ATX specification required all voltage levels be maintained within 5% of the nominal values and the regulator was rated to maintain its output within ~1% of 12 V, this was considered an acceptable source for directly powering the computer. [C5][C6] A simplified schematic of the modified power supply circuit is shown in Figure C2.

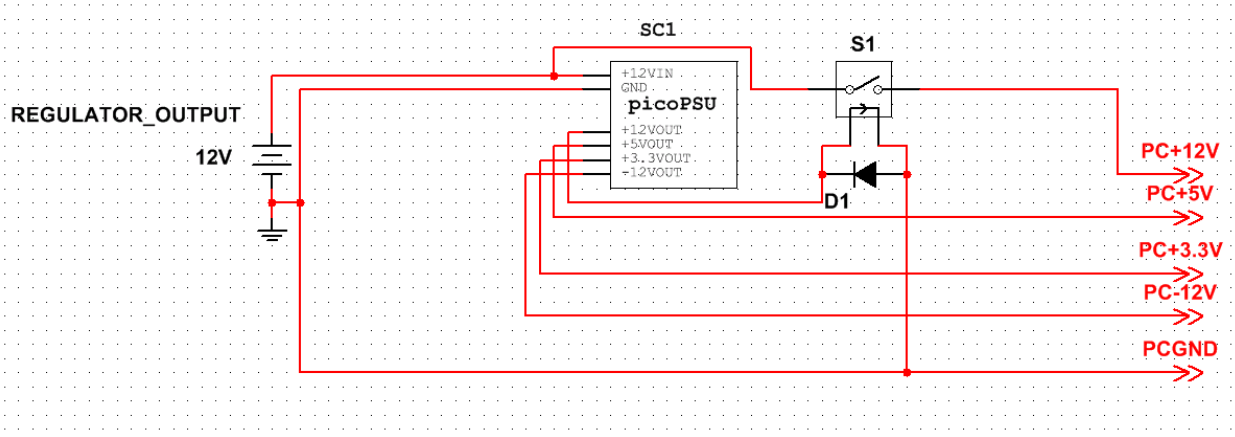


Figure C2: FMC Power Supply schematic

C2.2. Microcontrollers

The FMC controlled the rest of the electronics in the plane through a series of nine microcontrollers: three RoboClaw motor controllers, three ATMEGA2560s on Arduino development boards, and a Mini Maestro servo controller.

The RoboClaw motor controllers each provided control over two of the actuator motors. 45 A models were used for the BAG motors, while a 15 A model was used to operate the smaller brake motor. Commercially available motor controllers were chosen over custom designed control circuitry because custom circuitry would have been difficult and time-consuming to make and because the RoboClaw controllers came with pre-written control software that greatly simplified the process of control over USB. The RoboClaws read in actuator position data from the quadrature encoders and used this data to provide closed-loop position control of the motor-driven actuators based on commands from the FMC, using a PD control system tuned using the RoboClaw configuration software.

The Mini Maestro servo controller was selected for similar reasons as the RoboClaws. The Maestro provided an easy way to produce PWM servo control signals based on USB commands, and its many output channels allowed for several servos to be controlled over a single USB line.

Arduino development boards were used for most basic I/O. This was primarily because of the simple-to-use software interface, which most people working on the project were familiar with and could easily use. Two Arduino Megas were used (one in the front of the plane, one in the back). The forward Arduino was used to drive LED indicator lights in the front panel, to read in the analog airspeed value from the pitot system pressure sensor, and to operate a set of eight relays controlling: the computer push-to-talk switch; the landing lights; the anti-collision light; the navigation lights, the fuel pump, the left and right magnetos, and the starter motor. The rear Arduino was used to interface with the I2C barometer/outside air temperature sensor mounted in the tail of the aircraft. A third Arduino Mega was used as part of a tachometer sensor circuit (described in Appendix B).

Computer Interface to the Yoke Switches

The yoke switches communicates with the flight management computer (FMC) though USB HID interface. A Cypress PSoC 5LP microcontroller was programmed to appear as a 24-button USB joystick, allowing the FMC software to easily incorporate the switches without the need of extra drivers. All switch and encoder signals are debounced in hardware using the PSoC's universal digital block (UDB). Figure C3 shows the USB descriptor used to emulate a joystick.

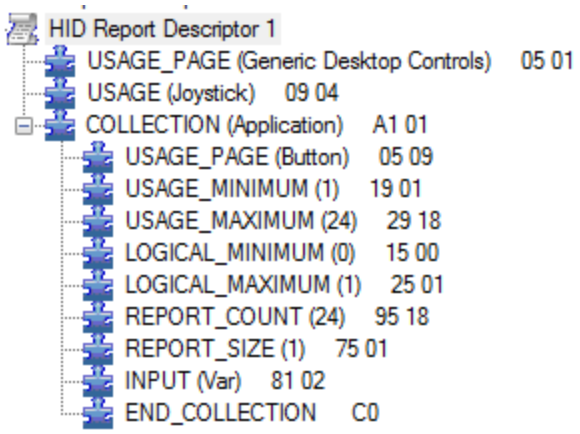


Figure C3: USB descriptor to emulate a joystick

Computer Interface to the Current Sensors

Three LEM HO-S series Hall Effect current sensors were used to sense the battery charging and discharging current. Each of the current sensors outputs a single-ended voltage signal with a reference voltage. The sensitivity is 16 mV/A for the 50 A sensor and 8 mV/A for the 100 A sensors. A Cypress PSoC 5LP microcontroller was used to read the sensors. The on-board delta-sigma ADC is configured in 16 bit differential mode with the internal reference, providing a resolution of 0.06 mV. An analog mux was used to multiplex the three pairs of input signal. The current sense readings are sent to the flight management computer via a raw USB HID protocol, and the over-current condition is sent to the Safety Officer through the CAN bus.

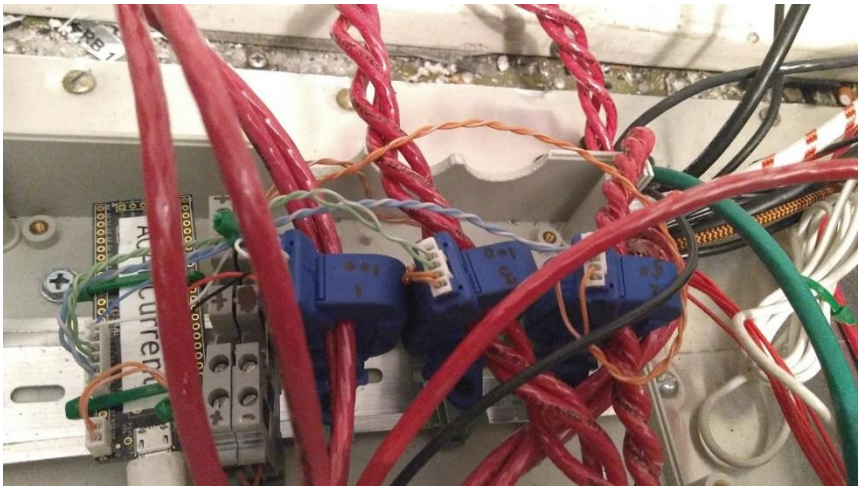


Figure C4: Current sensors and the Cypress PSoC 5LP microcontroller

C3. Power Distribution Network

The aircraft electrical system was the primary power source for all major systems added to the aircraft. Other options, such as a high-capacity battery or a generator, were deemed impractical either because they would limit flight duration due to their lower capacity or because they presented safety and practicality issues. The aircraft electrical system consisted of a 12.6 V lead-acid battery charged by a 14 V alternator powered by the engine. The alternator was the main power source, and was rated to provide up to 60 A of current. The alternator could power any of the systems as long as the engine was running, so the total energy capacity was limited only by the amount of fuel the aircraft could carry. The aircraft battery was capable of powering very high-power loads for short periods; it ordinarily did so when operating the starter motor. The battery in the plane was rated to provide 440 A cold-crank (that is, it can provide 440 A for at least 30 s without the battery voltage dropping below 7.2 V).

In order to ensure that the alternator could provide enough power to operate all systems during a sustained flight, the power consumption of each electrical system was analyzed. The systems were then adapted to ensure that the total average and total peak power draws were within the ranges that could be provided by the aircraft electrical system.

Continuous Load

Table C1 below shows the predicted total current draw of the electrical systems that were to operate continuously. The current draw was based on power consumption while running at the 14 V alternator voltage.

Constant current draw			Current (A)
	Aircraft		
		Alternator winding	3
		Anti-collision light	3.5
		Engine gauges	1
		Elec. Turn & bank	0.5
		Master solenoid	0.8
	Computer		
		Flight computer	17
		µCU's & sensors	1
		Relays	1
	Total		**27.8**

Table C1: Continuous Load Analysis

The constant current load on the aircraft electrical system was predicted to be 28.8 A. Current sink values for the aircraft components were obtained from the Piper Cherokee Service Manual as the load values for system components on the PA-28-140 operating at continuous duty cycle. In order to reduce power consumption, the system was limited to operation in daylight during fair-weather conditions (air temperature above freezing). Limiting the aircraft to warm-weather flight eliminated the need to use the pitot heater, which would have drawn an additional 13.2 A when in use. Limiting the aircraft to daylight flight eliminated the need to use several high-powered lights: the position light (4 A), the instrument light (1 A), and the cabin dome reading light (0.6 A). The cigar lighter was also not used, as it drew 8 A when in use and was a completely unnecessary accessory. Limiting the aircraft's operational window to remove these five components reduced the theoretical constant current consumption by 18.2 A and the maximum theoretical transient current consumption by 26.8 A. The computer power was calculated for a PC with an 85% efficient 200 W power supply constantly running at full load. . The current consumption figures for the microcontrollers, sensors, and relays were rough estimates based on datasheets for representative components.

Transient Load Analysis

Table C2 below shows predicted worst-case peak current draw for the systems on the Aircraft Battery. The current draw was based on power consumption while running at the 12 V battery voltage.

Transient current draw			Current (A)
	Constant current draw		27.8
	Aircraft		
		Landing lights	8
		Fuel pump	0.5
		Radio	2.5
	Actuators		
		Main actuators	265
		Servos	10
	Total		**313.8**

Table C2: Transient Load Analysis

The maximum transient current load on the aircraft electrical system was predicted to be 290.8 A. This load was calculated as the sum of constant current draw and the maximum possible current draw for all transient loads. The current sink values for the landing lights and fuel pump when running at 12 V were obtained from current draw values at 14 V listed in the

Piper Cherokee Service Manual through Ohm's law based on the assumption that they can be modelled as pure resistive loads. Main actuator current values were calculated as the sum of five BAG motors, each with a 53 A stall current, running at stall simultaneously. Servo current draw values were calculated as the draw from six servos, each drawing a maximum of 2.5 A from a 6 V DC power supply operating at 80% efficiency.

It was highly unlikely that this load would ever be reached for any extended time during actual operation. The only time the main actuator motors were likely to be operating at stall was very briefly as they begin moving from a complete stop, and the maximum load would only occur if all five actuators were activated simultaneously. Because any current spikes approaching this maximum transient value would have been very short-lived (most likely less than a second in duration) this current could be provided by the aircraft battery, which was rated to produce 440 A for brief periods. The act of drawing so much current would likely cause the voltage at the positive terminal of the aircraft battery to drop somewhat, so in order to prevent critical systems from shutting down during these periods, flight compute, and all autopilot systems except for the main actuators and servos were to be powered from a separate battery that was to be disconnected from the alternator and aircraft battery whenever a large current spike occurred, then reconnected to charge once the spike subsided. Because the autopilot electronics were not to be connected during high-load scenarios, they were not included in this calculation.

Expected Average Current Draw

Table C3 below shows the expected average current draw of the electrical systems over the course of a flight. The current draw was based on power consumption while running at the 14 V alternator voltage.

Average current draw			Current (A)
	Constant current draw		27.8
	Aircraft		
		Radio	0.5
	Actuators		
		Servos	4
		Main actuators	15
	Total		**47.3**

Table C3: Expected Average Load

The expected average current load on the power system was 48.3 A. This was calculated as the sum of the constant current load with the expected average current draw for the radio,

servos, and main actuators. The landing lights and fuel pump were omitted because they run so infrequently that they will have little impact on the overall average power consumption over the course of a flight.

The battery in the aircraft electrical system was not intended to be used as a power source; it was only present to act as a short-term source of high current and to stabilize the voltage output from the alternator. The 60 A alternator was the only actual power source on the aircraft, so in order for the aircraft to be capable of continuous flight, the average power consumption had to be below the maximum amount of power the alternator can generate. The 48.3 A draw of the system was within the normal operating range of the alternator, so the system was deemed able to maintain flight unlimited by availability of electrical power.

Voltage Regulation

A major function of the power distribution system was to ensure to provide sufficient power at the voltage levels required to operate the different equipment present in the aircraft system. This equipment could be broadly divided into four categories based on the type of power source required for reliable operation: the main actuators, which required a high-power source; the servos, which required a moderate-power regulated source; the microcontrollers and sensors, which required low-power regulated sources; and the Flight Management Computer, which required high-power regulated source. The first three of these systems were relatively simple to design; the fourth, the flight computer power supply, was considerably more complex.

Aircraft Electrical System

As previously discussed, the aircraft's electrical power system consisted of a 14 V rectified alternator and a 12.6 V lead-acid battery. While the electrical system included basic voltage regulation to maintain the alternator output at 14 V, significant variation was still possible both due to the AC nature of the alternator and to the possibility for transient current draws that could temporarily exceed the maximum alternator output, resulting in a voltage drop on the main aircraft power rail. To account for these variations, power supplies were designed to account for an input voltage that could range from ~10 V-16 V (or ~12 V-16 V for systems powered from the semi-isolated computer battery).

Basic Power Supplies

Of the three basic power systems, the main actuator power source was the simplest. The main actuators and the motor controllers chosen to operate those actuators were capable of

operating over a wide range of input voltages--the RoboClaws were rated to operate from a 6 V-34V source. This wide range allowed the motors and associated controllers to be powered directly from the unregulated 14 V aircraft electrical system.

The servos, which contained embedded electronics to provide position control, required a regulated 5-6 V power source. This regulation was obtained through commercially available servo power supplies. The selected supplies were DC-DC step-down switching converters, designed to provide a regulated 5 V or 6 V output from a higher-voltage (<8 V-36 V) power source. Switching converters were selected because of their much greater efficiency relative to linear voltage regulators when driving high-current loads. A separate 2.5 A supply was used for each servo so that each servo could be deactivated independently by switching of its power supply.

The Arduino Mega microcontroller and Maestro servo controller were both capable of operating from a 5 V power source and drew relatively little power. Both boards were capable of obtaining this power from a basic USB connection. As both boards were to remain connected to the Flight Computer via USB during normal operation, no additional power source was needed for these boards.

Computer Battery Isolation

Requirements

The computer battery provides power for the flight computer, microcontrollers, and voltage-sensitive avionics including the radio. To enable charging, the computer battery needs to be connected in parallel with the alternator and the aircraft battery. When the aircraft battery output voltage dips due to transient high current draw from the starter and the main actuators, the computer battery shall maintain its voltage within the normal input range of the microcontroller POL (Point-of-Load) DC/DC converters and the computer DC/DC converter. The circuit shall switch 20 A continuous and 200 A peak current, and 9-14 V voltage.

Design

The required logic can be easily satisfied with a diode. V_1 connects to the alternator and aircraft battery. V_2 connects to the computer battery. In normal operation $(V_1 – V_2 > V_f)$, the current flows from V_1 to V_2, and it charges the computer battery and supplies power to the attached systems. When V_1 dips $(V_1 – V_2 < V_f)$, the diode stops the reverse current flow and maintains voltage.

This design is practically infeasible because even with a low forward voltage high power Schottky diode (IXYS DSS2x101-015A), the forward voltage reaches above 500 mV at 20 A. [C7] This means the diode would continuously dissipate 10 W of heat. The forward voltage drop also prevents the battery from being fully charged, causing the lead-acid battery to operate at suboptimal conditions.

Another design that have been considered is a relay controlled by voltage difference between V1 and V2 (Figure C5). This design enables bidirectional control of current flow. More complex control logic can be implemented with a microcontroller. Heat loss is minimized. However, high power relays and contactors are inherently expensive and slow. Most of the relays that will meet the power requirements take tens of milliseconds to actuate [C8], which is ineffective for transient voltage protection.

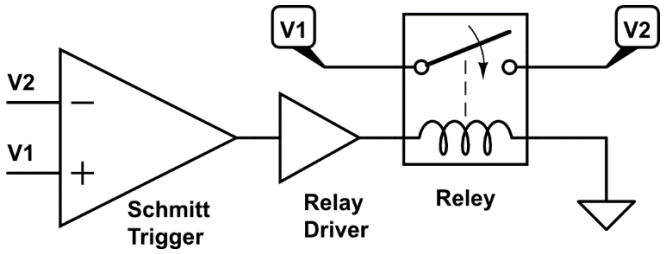


Figure C5

The response time requirement can be satisfied by changing the switching element from a relay to a MOSFET. Driving an N-channel MOSFET that has 12V voltage at Source requires more complex circuit and additional power supply, because the MOSFET needs Gate voltage $V_{gs(th)}$ above Source voltage to turn on. Linear Technology LTC4359 Ideal Diode controller with Reverse Input Protection provides an integrated solution to this problem (Figure C6). LTC4359 emulates an ideal diode by switching a MOSFET depending on voltage difference between Vin and Vout. [C9]

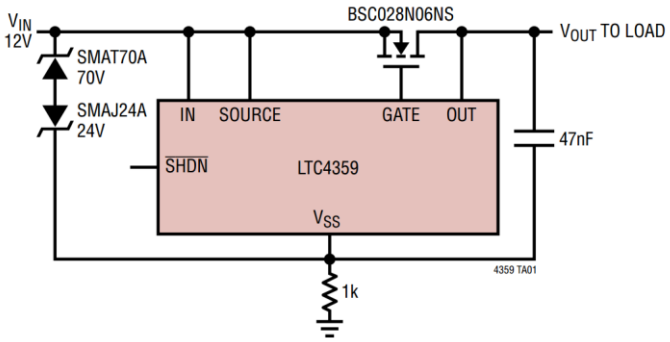


Figure C6: Ideal Diode example circuit

IXYS IXTN600N04T2 N-channel MOSFET was chosen for the switching element. It is rated for 600 A Drain to Source current and 40V Drain to source voltage [C10]. It is in a chassis mount package with screw terminals, which is convenient for connecting heavy gauge wires and heatsink. With the Drain to Source resistance of 1.3 mΩ [C10], the IXTM600N042T2 only dissipates 0.52W of heat at 20 A. Thermal calculation shows only 5 °C temperature rise with 5 °C/W heatsink at 20A, and 60 °C temperature rise at 200 A. The LTC4359 and IXTN600N04T2 combination was chosen because it provides a simple solution that meets all requirements.

The reverse flow circuit (Figure C6) is identical to the typical application in the LTC4359 schematic [C9]. The LTC6703 circuit is intended to visually indicate whether the MOSFET has been turned on by sensing the gate voltage V_{gs}. The LTC6703 is a comparator with an internal 400 mV reference. The design is incorrect because the source of the MOSFET is not at ground potential. This problem can be solved by making the comparator measure the differential voltage between the gate and source instead of between gate and ground. Because the reverse flow protection module is installed inside an opaque box, the visual indication is not needed. Fuse F1 was intentionally blown to remove power from the LTC6703 comparator.

Figures C7 and C8: Ideal Diode board layout; Ideal Diode circuit

C1. Testing, Results, and Recommendations

Full testing was conducted of the aileron, stabilator, and rudder mechanisms. These systems all performed well enough to allow for a fly-by-wire demonstration in which the aircraft taxied around for several minutes in a "fly-by-wire" configuration—controlled by an onboard flight engineer using a joystick connected to the FMC. During the demonstration, the flight engineer demonstrated full motion of the ailerons, rudder, and stabilator, and demonstrated full deployment of the flaps with stops at each flap setting. This demonstration also acted as a test of the flight computer, power distribution system, and battery isolator, all of which appeared to work as intended. The test pilot on board during the demonstration described the performance of the aircraft under fly-by-wire as somewhat "jerky," but not at all violent, always predictable, and always responsive to the flight engineer's control. The control precision could likely improve significantly if alterations were made to reduce backlash in the mechanical systems, particularly in the clutches, as this would reduce control hysteresis. With additional time for more testing, minor alterations like improved tuning of the PID/PD controllers could likely also produce improvement in control responsiveness. Figure C9 shows the aircraft operating under fly-by wire control during the demonstration.

Figure C9: Fly-by-wire taxiing demonstration

Flaps

Construction of the flap mechanism was not yet complete at the time of the demonstration. The mechanism for depressing the flap handle button to retract the flaps was never installed, and no position sensor was installed to monitor the position of the flap handle. Even so, the flap mechanism worked well under open-loop control, smoothly deploying the flaps, stopping at each flap position, and retracting the flaps if the flap handle button was first manually depressed. This excellent performance suggests that if a flap button mechanism and position sensor were added, the flap mechanism would be capable of full fly-by-wire operation.

Throttle and Mixture

During preliminary testing, the mixture actuator worked perfectly, moving the mixture lever in response to movement of a joystick. The throttle mechanism, however, caused the servo controller to lose its connection to the computer when the throttle servo ran quickly. The throttle

and mixture servos were both removed to troubleshoot this problem, and while the exact cause was never found, a faulty servo appeared likely. Unfortunately, neither servo could be remounted in time for the demonstration test this. Even so, the behavior of the mixture servo was promising for the servo control system, suggesting that further troubleshooting would be worthwhile.

Battery Isolator

At one point during testing, several hours using the computer without running the engine began to deplete the battery. At this point, the battery isolator correctly connected the aircraft battery to maintain the voltage level and keep the computer running.

Flight Computer

Extensive lab testing revealed an initial problem with the flight computer power supply, in which the inrush current produced when the computer first turned on caused a voltage drop at the FMC power supply input, shutting the supply off. In response, a 2.2 mF electrolytic capacitor was placed across the PSU input to suppress the transient voltage drop. This fixed the problem. During normal operation, the computer greatly outperformed efficiency expectations: even during lab stress-tests using graphics-intensive software, the total current draw of the computer never exceeded 4 A—less than 25% of the predicted consumption.

Appendix C References

[C1]. "BAG Motor". *VEX Robotics*. N.p., 2017. Web. 25 Apr. 2017.

[C2]. "Versaplanetary Integrated Encoder". *VEX Robotics*. N.p., 2017. Web. 25 Apr. 2017.

[C3]. (2016). 1u high 200 watt 12 volt dc input pc atx-24 pin or atx-12 style power supply, [Online]. Available: http://www.powerstream.com/DC-PC-12V-200I.htm.

[C4]. Picopsu-160-xt quick installation guide, version 1.0d, Mini-Box.com. [Online]. Available: http://resources.mini-box.com/online/PWR-PICOPSU-160-XT/PWR-PICOPSU-160-XT-manual.pdf.

[C5]. Power supply design guide for desktop platform form factors, Rev. 1.2, Intel Corporation, Feb. 2008. [Online]. Available: http://www.formfactors.org/developer/specs/Power_Supply_Design_Guide_Desktop_Platform_Rev_1_2.pdf.

[C6]. Vhk200w dc-dc converter datasheet, CUI Inc, Dec. 2013. [Online]. Available: http://www.cui.com/product/resource/vhk200w.pdf.

[C7]. Power schottky rectifier , DSS 2x101-015A, IXYS, 2004

[C8]. G. A. Peterson and T. Graff, Relay actuation circuitry , US Patent 5,055,962, Oct. 1991

[C9]. Ideal diode controller with reverse input protection, LTC4359, LT 0514 Rev. B, Linear Technology, 2012.

[C10]. Trencht2 gigamos power mosfet , IXTN600N04T2, IXYS, Oct. 2012.

Appendix D

Design of the Safety Officer

Contents

Author: Keshuai Xu

D.1 Overview

The Safety Officer module monitors the system state, handles critical human interface, communicates with the Flight Management Computer, and actuates safety features (clutch mechanism and actuator power) to return the control of the aircraft to the human pilot. It locates in the instrument panel, featuring six illuminated push buttons, four indicator lights, and an OLED display (Figure D.1). In the rear of the unit, it has connectors for three servo motors, seven relays, two power module disable signals, an external autopilot disconnect switch chain, a USB, and a CAN bus (Figure D.2).

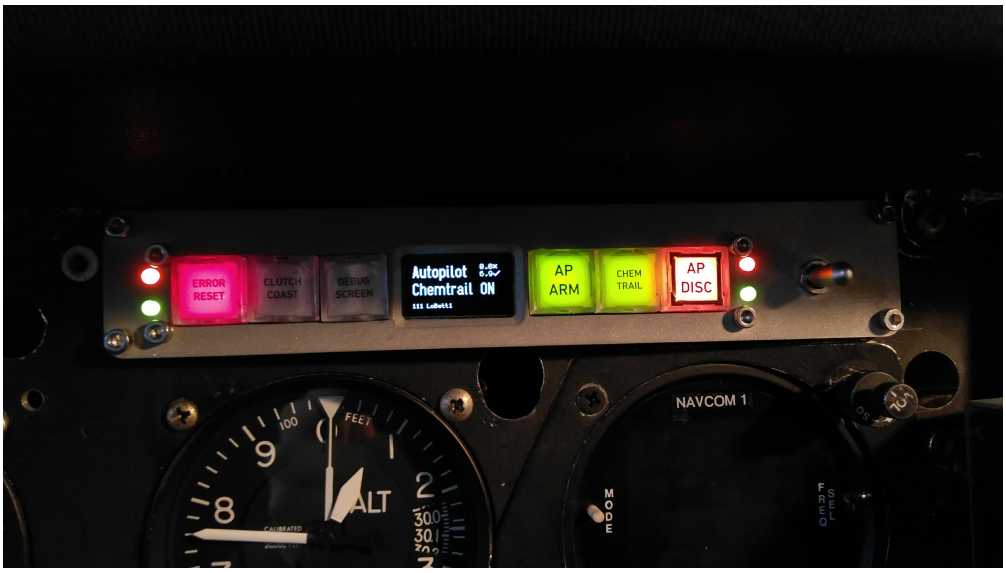


Figure D.1: The Safety Officer running in the aircraft

D.2 Safety Features in the Actuation System

The safety features in the actuation system are shown in Table D.1. The principle for the design is that the human pilot must be able to either easily back-drive the motor, or mechanically disconnect the motor from the rest of the aircraft.

Relays All the DC motors driven by H-bridges have a relay in series with the motors to cut power. A normally-open SPST relay is inserted in series with the motor armature, between the two half bridges of the H-bridge driver (Figure D.3). The reason for this design instead of placing the relay on the motor power rail before the H-bridge drivers is, when the power is cut and the motor is being back-driven, the back EMF created by the motor can feed into the power rail through the H-bridge MOSFET body diodes, which results in increased back-driving force and unintentionally powering up other motors.

The coils of all actuator power off relays are powered by a single power rail with a manual switch. In the event of relay contacts do not open due to a microcontroller logic failure, the pilot can turn off the power to force open the relay contact.

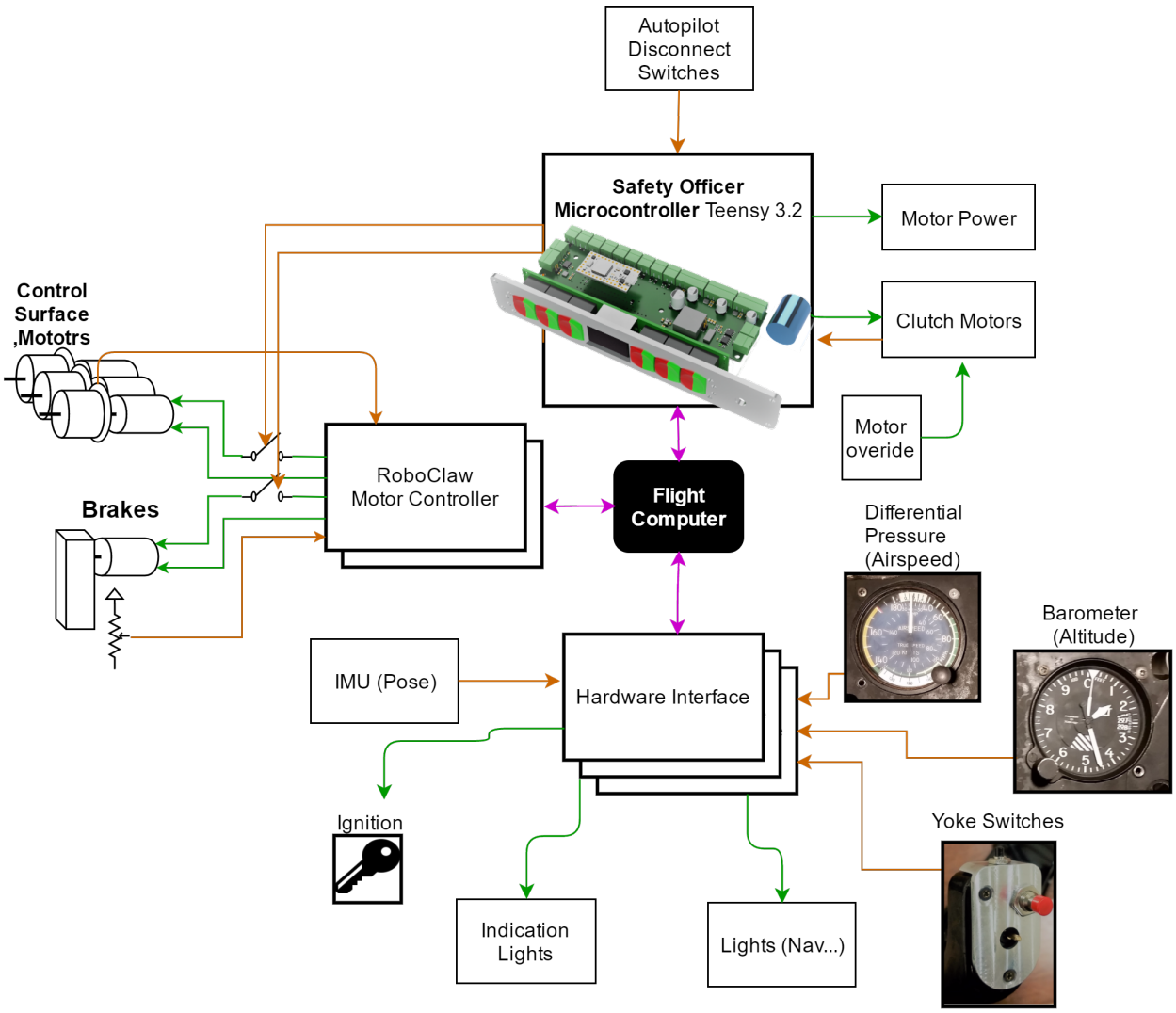


Figure D.2: Connections of the Safety Officer

Clutches The aileron, the stabilator, and the rudder motors are connected to the rest of the aircraft via clutch mechanisms due to their high-power and high-back-driving-force nature. The clutch mechanism is a pair of dog gears driven by an RC servo motor (Figure D.4). The clutch can be meshed when the dogs are aligned, and can be disengaged with less than 5 N of force under full load.

RC Servo Power Supply Disable Each RC servo motors in Table D.1 gets power from a dedicated DC-DC converter, stepping down from aircraft battery voltage to 5 V or 6 V depending on the servo. Pololu 5V, 5A Step-Down Voltage Regulator D24V50F5 (Figure D.5) or a similar variant was used. The Pololu D24V50F5 features an enable pin. The enable pin is internally pulled up to the input voltage. When the enable pin is pulled below 1 V, the voltage regulator turns off the output. This feature enables the Safety Officer to control the power of the RC servos.

Actuator	Driver Type	Safety Feature
Aileron	H-bridge	Relay and Clutch
Stabilator	H-bridge	Relay and Clutch
Rudder	H-bridge	Relay and Clutch
Flaps	H-bridge	Relay
Stabilator Trim	H-bridge	Relay
Brakes	H-bridge	Relay
Rudder Trim	RC Servo	Power supply disable
Throttle	RC Servo	Power supply disable
Mixture	RC Servo	Power supply disable
Carb Heat	RC Servo	Power supply disable
Flap Handle Button	RC Servo	Power supply disable

Table D.1: Actuator system safety features

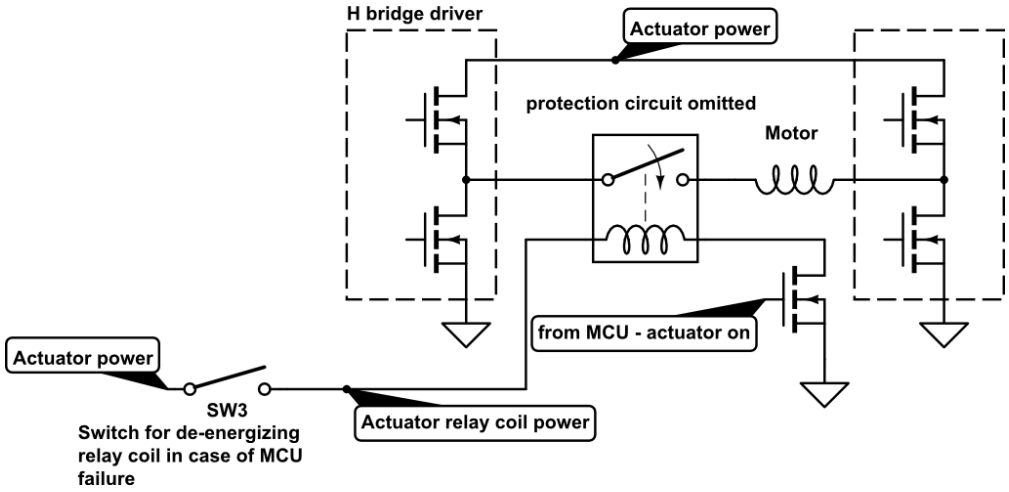


Figure D.3: DC motor power off relay and driving circuit

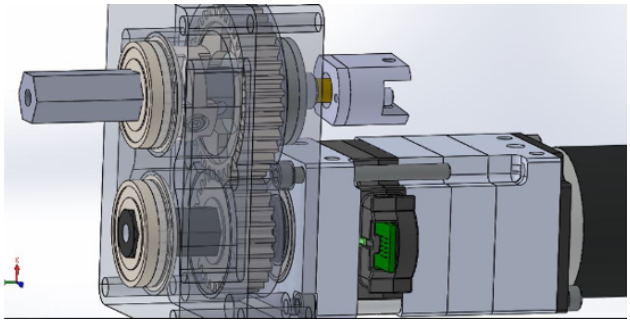


Figure D.4: A CAD screen shot of an early design showing the clutch mechanism

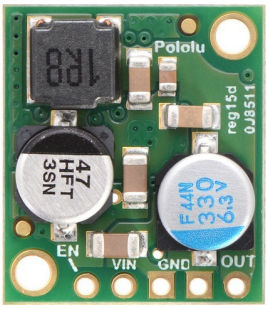


Figure D.5: Pololu 5V, 5A Step-Down Voltage Regulator D24V50F5 [1]

D.3 Safety Features in the Ignition Control

Author: Benjamin Gillette

For the Flight Management Computer to start the aircraft engine, it needed to be able to control the magnetos and the starter motor. This in itself was not difficult (the devices could all be switched through relays), but giving control of the engine to the computer presented multiple safety hazards. These hazards were addressed by developing a system of switches to ensure that the pilot could override the computer and manually set the starter and magneto states at any time.

The main difficulty presented by the ignition safety system was the fact that the safe failure mode was different depending on whether the plane was flying or on the ground. When flying, the system had to default to leaving the engine on to ensure that the aircraft could continue to fly; when on the ground, the engine had to be kept from accidentally turning on, as the propeller could have injured unsuspecting ground personnel. This was accomplished using a combined series-parallel connection of switches as shown in Figure D.6.

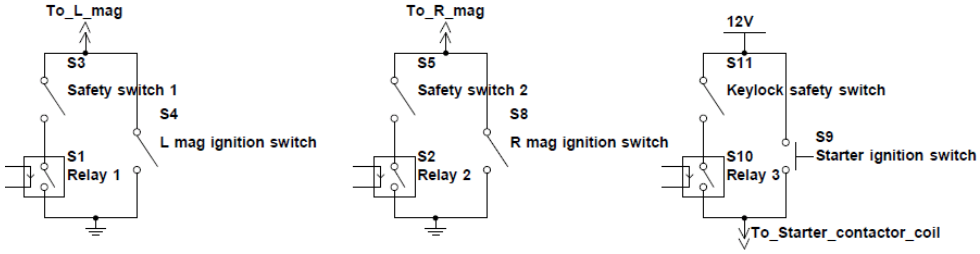


Figure D.6: Schematic of the Ignition Safety Interlock

The computer-controlled relays were placed in parallel with the ignition switches, and additional physical switches were placed in series with the relays. Opening the switches allowed the computer to be completely disconnected from the system, returning the aircraft controls to their original functions. For additional safety, a key lock switch was used for the starter motor safety switch to ensure that the engine could not easily start without a key.

D.4 State Machine Design

The top-level logic of the Safety Officer is based on a finite state machine (FSM). A simplified state transition diagram is shown in Figure D.7. The software initializes in *Not Armed* state. When the human pilot presses the *Autopilot Arm* push button, the FSM transits into *Armed* state. When the Flight Management Computer (FMC) commands to start autopilot, the FSM transits into *Autopilot* state. In any state, when autopilot disconnect is commanded through FMC or one of the push buttons, the FSM transits into *Not Armed* state. In *Not Armed* and *Armed* states, the motors remain off, and the clutches are disengaged. In the *Autopilot* state, the motors are on and controlled by the FMC, and the clutches are engaged. The goal of this state machine design is to ensure the autonomous control only starts with human pilot's consensus and can be disabled at any time.

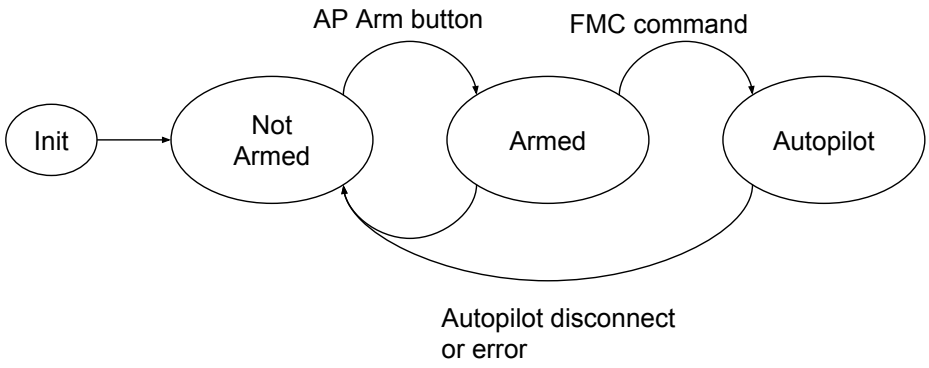


Figure D.7: State transition diagram of the Safety Officer

Each of the clutch servo motors has an FSM. It is shown in Figure D.8. In *Out* state, the clutch is disengaged, and in *In* state, the clutch is engaged. The FSM initializes in *Moving Out* state. The clutch motor disengages the clutch mechanism in full power and transits into *Out* state upon success. When *Autopilot* is triggered in *Out* state, the clutch transit into *Moving In* state, where the clutch motor applied gentle pressure to the clutch mechanism, and the control surface motor gently rotates until the clutch meshes, which transits the FSM into *In* state. In any state except *Moving Out* state, the *Autopilot Disconnect* trigger will transit the FSM into *Moving Out* state. This design ensures the clutch can be disengaged instantly in any state. Even in *Out* state, the *Autopilot Disconnect* trigger will cause the clutch to retry disengaging, in case the clutch mechanism did not hold its position in *Out* state.

If the clutch mechanism fails to mesh in *Moving In* state before time out, the FSM transits into *Moving Out* state and raise an error message. If *Moving Out* times out, the FSM transits into *Error* state. The human pilot may recover from the error by retry disengaging the clutch.

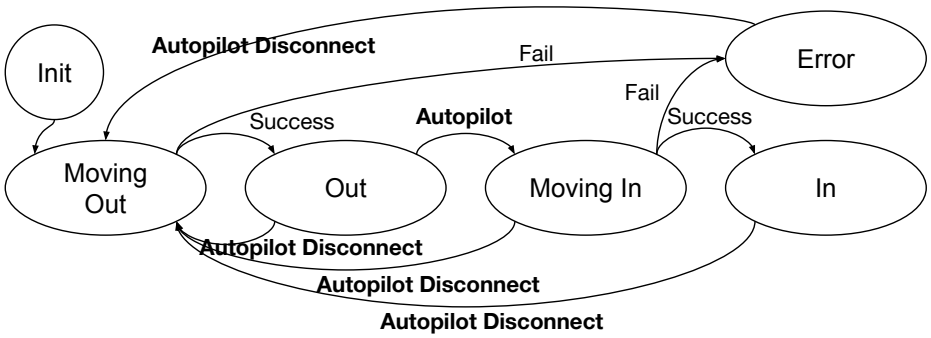


Figure D.8: State transition diagram of clutch servo motors

D.5 Actuator Disconnect Relays and Driving Circuit

The actuator power off relays electrically disconnect the motors from H-bridge drivers. The relays are required to cut off 60 A DC current with a highly inductive load. TE Connectivity Potter & Brumfield Relays V23134J1052D642 are used in the physical implementation.

To prolong the life of the relay and reduce electromagnetic interference (EMI), the arc across the contacts generated by cutting off inductive load must be suppressed. An RC network and a TVS diode are placed in parallel with the relay contact (Figure D.9). The TVS (Transient Voltage Suppression) diode should have a breakdown voltage higher than maximum back EMF generated by the motor when back-driven. The TVS diode clamps the voltage across the relay contact to 24V, and the RC network dissipates the arcing energy. The RC network is omitted in the final implementation because the qualifying physical components are unrealistically large, and a TVS diode alone can provide enough protection for the relays.

The driving circuit on the Safety Officer board is an array of N-channel MOSFETs with flyback diodes driven by microcontroller GPIOs. Opto-isolation for the MOSFET gate signal was considered but omitted due to the low and predictable inductance in the relay coil. The relays are powered from the main battery power only. Loss of main battery power disconnects the relay and enables the motors to be safely back-driven.

D.6 Clutch Servo Driving Circuit

The aileron, stabilator, and rudder motors have clutches for decoupling the motors from the yoke and control surfaces when the human pilot takes control. An RC servo motor is chosen to drive each clutch mechanism because the RC servo has a high torque motor and gearbox with a position feedback sensor in a compact package. However, the integrated driving circuit pre-installed in the RC servo is unsuitable for our application because it does not provide an interface to directly control the torque of the motor and read back the position. Simply commanding the servo with PWM signal to mesh the clutch will cause the servo to operate in full power stall condition and will cause it to overheat. To address this problem,

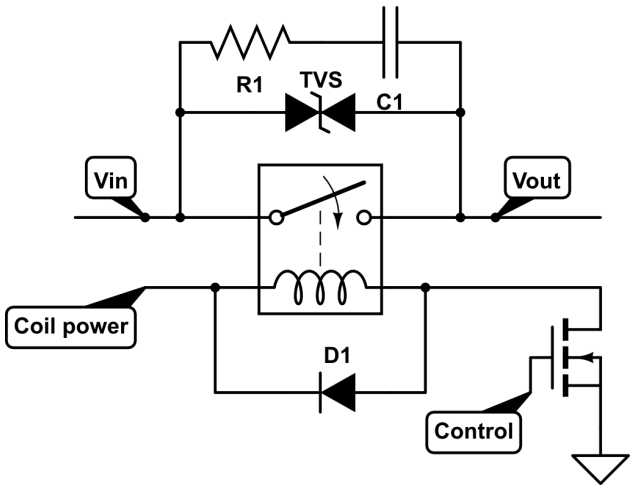


Figure D.9: Relay arc suppression circuit

the pre-installed driver circuit in the RC servos was removed and all the signals were exposed to the Safety Officer board (Figure D.10). The safety officer board included 3 motor driver channels for the motors and 3 ADC channels for the potentiometers. The motor torque is set by the duty cycle of the pulse width modulation (PWM) signal, and the motor position is read from the potentiometer with the on-chip ADC. With the position feedback, the position of the clutch servo is close-loop controlled with a PID controller.

Each channel of the driver circuit (Figure D.11) uses a Texas Instruments DRV8835 Dual Low-Voltage H-Bridge IC to drive a motor. Each channel in the DRV8835 supplies up to 1.5 A current, and two channels can be connected in parallel for 3 A driving current[2]. The torque of the motor is set by PWM on IN1 and IN2 ports in the schematic. The POT port connects to an ADC pin on the microcontroller.

The DRV8835 are powered from a 5 V 10 A output DC-DC converter, offering consistent performance under varying input power conditions. The input voltage ranges from 7 V to 14 V.

Texas Instruments DRV8835 is selected because it integrates the control logic, the gate driver, the FETs, and the protection circuits into a tiny package. It has a good balance between the size and the on resistance.

The clutch motors can be externally overridden when the Safety Officer fails. Each motor connects to a DPDT on-off-(on) switch. The on side connects to the Safety Officer, and the momentary (on) side connects to a backup battery. In normal operation the switches are left at on position, allowing the Safety Officer to control the clutch motors. In Safety Officer fail condition, the pilot toggles the switch to the (on) position, and use the battery power to drive the clutch to the disengage direction. When the pilot release the switch from the (on) position, it rests at off position and keeps the clutch motor off.

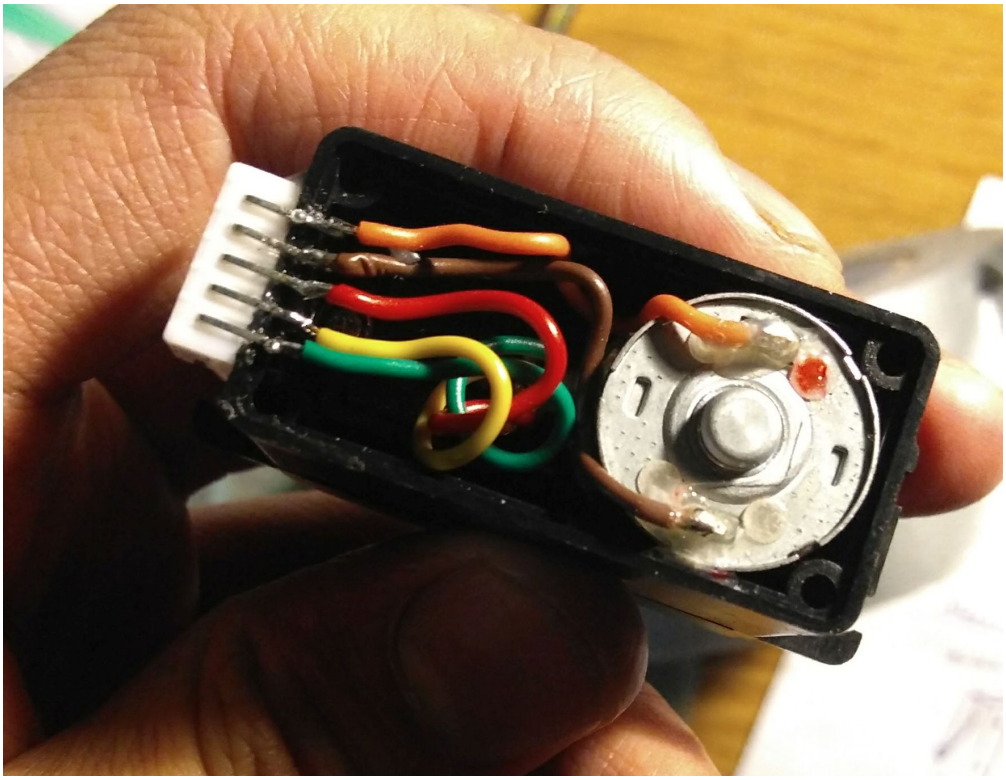


Figure D.10: Modified clutch servo motor. The rear cap was removed to show the wiring

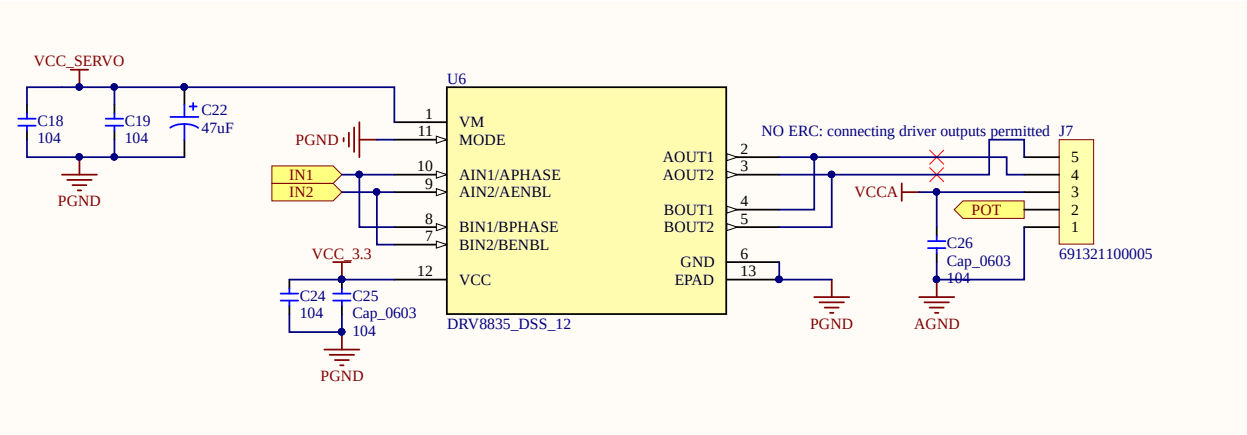


Figure D.11: Clutch servo driver circuit

D.7 RC Servo Power Supply Disable

Each of the RC servos motor for the throttle, the mixture, the flap button, the carburetor heat, and the rudder trim is powered by a DC-DC converter with enable signal. The enable signal is pulled up to input voltage with a 100k resistor. The corresponding microcontroller GPIO is configured in open drain mode. When the enable signal is pulled below 1 V from the microcontroller, the DC-DC converter shuts down[1] and turns off the servo motor.

The microcontroller operates at 3.3 V logic level, and applying 12 V will possibly cause the GPIO output driver to fail in SCR latch-up condition. The K20P64M72SF1 datasheet [3] did not mention the existence and ratings of the ESD diodes. For an extra layer of safety, a 3.3 V Zener diode (Figure D.12) is used in each channel to clamp down the voltage.

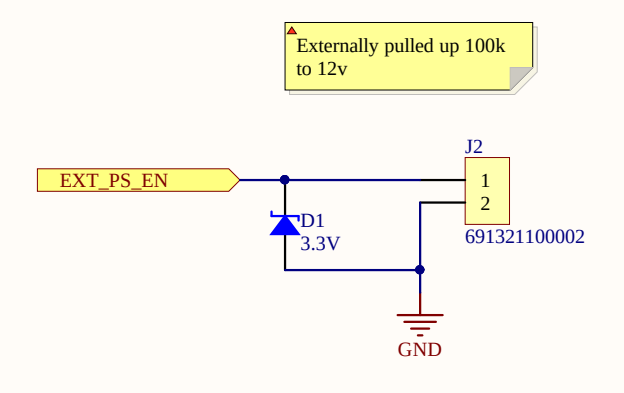


Figure D.12: RC servo power supply disable circuit

D.8 Redundant Power Supply

The safety officer must be able to disengage the clutches when the main battery power is turned off, in the case of a system malfunction or an electrical fire. The highest-priority power supply is from the aircraft battery. When the aircraft battery power supply fails, a 7.2 V Ni-MH battery seamlessly switches in and supplies power to the Safety Officer. Figure D.14 shows the schematic for the power supply.

LTC4417 Prioritized PowerPath™Controller was selected to control the power path switching. Figure D.13 shows the switching waveform. LTC4417 normally sources power from the aircraft battery. When the aircraft battery level too low (below 12 V), LTC4417 switches to the backup power. The 12 V threshold is set to protect the aircraft battery from costly over-discharge damage when the Safety Officer is left on while the aircraft is not running. The operating voltage of the Safety Officer is between 7 V and 17 V, limited by the operating range of the DC-DC power supplies. The Safety Officer can safely reject prolonged over-voltage up to 30 V, limited by the breakdown voltage of the MOSFETs. Over-voltage on one input channel causes the LTC4417 to switch to the other input channel, and it does not disrupt the normal operation of the Safety Officer. Higher voltage transients are handled by TVS diodes at the input. These protection features are critical because the aircraft power rail is expected

to be dirty caused by the motors and the alternator. The power switch of the Safety Officer controls the enable pin of LTC4417, which is internally pulled up and grounded through a locking toggle switch on the front panel.

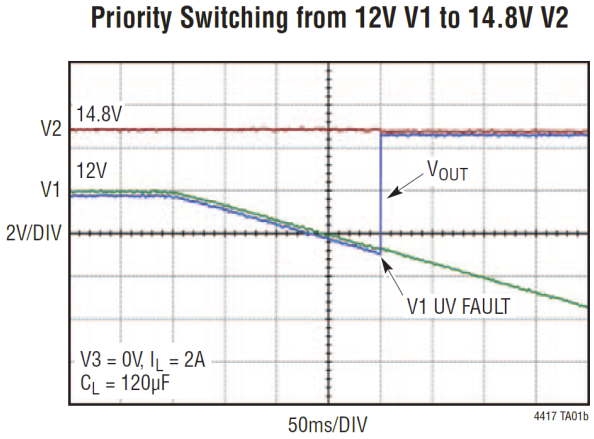


Figure D.13: LTC4417 power path switching from 12V V1 to 14.8V V2 [4]

LMZ21701 is a 1 A switch mode DC-DC converter. It supplies the 5 V rail, which powers the LED lights for the front panel, the buzzer, and the LDO on the Teensy board. TPS73633 LDO drops from 5 V to 3.3 V for logic level and analog power supply.

LMZ12010 is an 10 A switch mode DC-DC converter set to 5 V output. It powers the three clutch servo motors at approximately maximum 2.5 A each. It can be disabled from the microcontroller to ensure motor off when intended.

The input voltages are sensed with on-chip ADC using simple resistor dividers followed by RC filters. The voltage readings are used for sensing power failures and estimating battery levels.

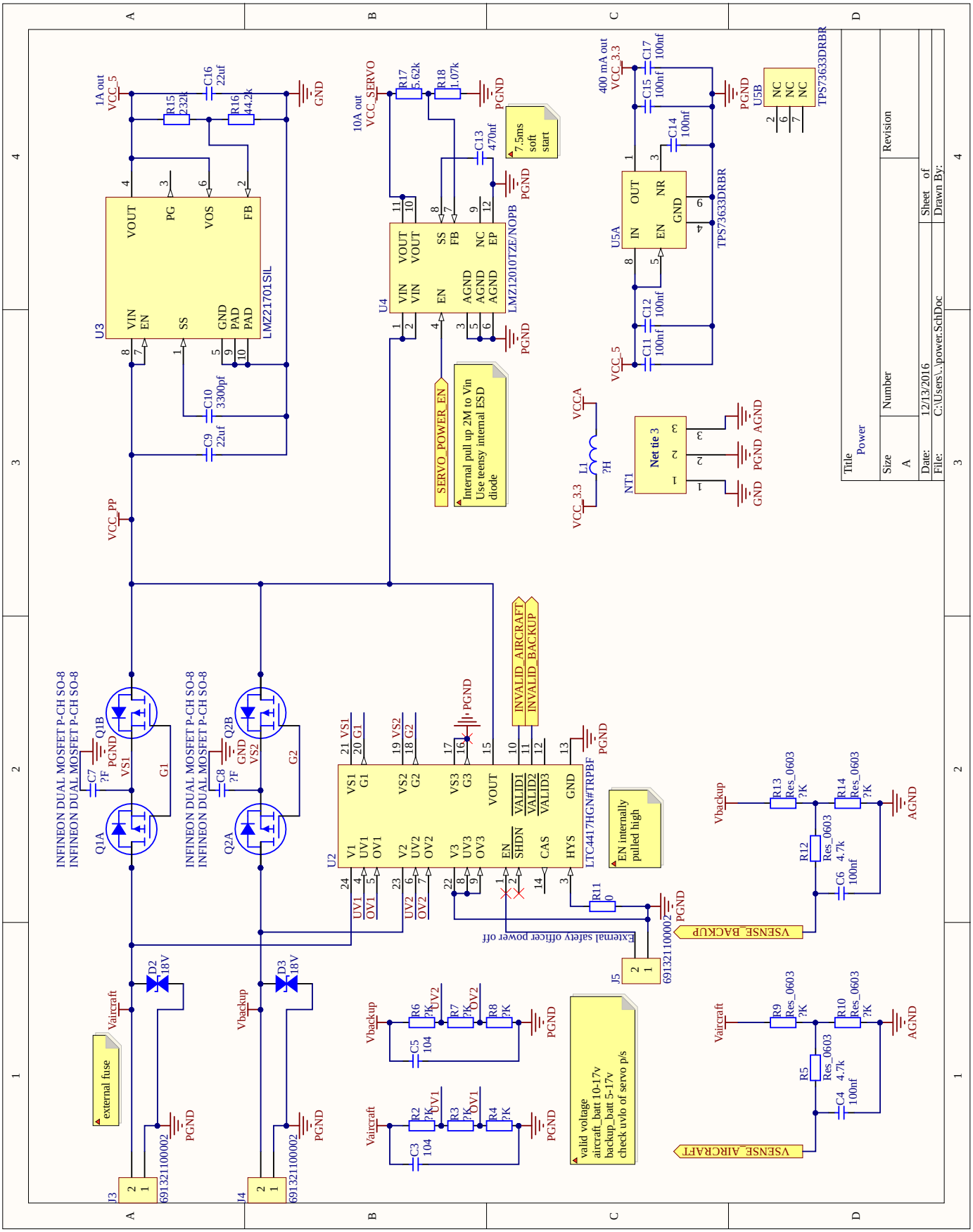


Figure D.14: Redundant power supply for the safety officer

D.9 Front Panel Switches, Display, Audio, and Lighting

The front panel consists of six NKK UB2 illuminated push buttons, four LEDs, and an SSD1306 OLED dot matrix display (Figure D.1). Each of the push button includes one red and one green LED backlight.

NKK UB2 push buttons were selected because they feature strong tactile feedback and bright light indication. Each of the button signals is RC debounced and connects to a microcontroller GPIO. Dedicated GPIO for buttons instead of array scanning enables the button press to be handled in an interrupt.

Texas Instruments TLC5926 16-Bit Constant-Current LED Sink Driver was used to drive 16 push button lights and indicator LEDs. Constant-current LED driver ensures uniform LED brightness. It is equivalent to a serial in parallel out shift register with a power stage, and it communicates with the microcontroller through SPI.

SSD1306 OLED dot matrix display was selected because of its small form factor, high brightness, and high resolution. It is visible in direct sunlight. The 128×96 resolution allows a large amount of information to be shown at the same time. It communicates with the microcontroller through SPI.

A Texas Instruments TPA6211A1 3.1-W Mono, Fully Differential, Class-AB Audio Amplifier is built into the Safety Officer to provide audible feedback. It was designed to drive a surface transducer mounted to the front panel. However, tests show that the 3.1 W audio output is not loud enough to be heard over the engine noise, and the surface transducer is removed from the design to save space. The audio output can be routed through the intercom instead.

D.10 Microcontroller and Connectivity

Teensy 3.2 development board with Freescale K20P64M72SF1 was selected because it is Arduino compatible, has integrated USB and CAN bus controller, and has a small footprint.

USB The Safety Officer communicates with the flight management computer (FMC) via USB HID protocol. It sends a packet to the FMC every 10 ms containing the status and battery voltages. It expects to receive a packet from FMC containing autopilot status every 10 ms. If the timing is not satisfied, the Safety Officer assumes the FMC is malfunctioning or disconnected and triggers autopilot disconnect.

USB HID protocol is chosen over USB Serial because USB HID uses INT (interrupt) transfer type instead of BULK. USB is never real-time. There is no guarantee on latency in either INT or BULK transfer type. However, with INT, the operating system is asked to poll the device at a certain interval. This non-realtime is acceptable because the operating system (Ubuntu Linux) that runs the FMC is not real-time after all. USB HID provides excellent performance in our test. USB HID data are packetized by the protocol instead of in

the application in USB Serial. The error detection and correction is handled at the hardware level by the USB controller.

CAN bus The CAN bus is designed for integrating remote current and temperature sensor to monitor the aircraft status. The sensors and microcontrollers are as far as 2 m away from the Safety Officer. SPI and I^2C are not reliable at this distance. The real-time nature of CAN bus ensures critical information such as over-current and over-temperature alarm are responded immediately. Texas Instruments TCAN1051HGV CAN transceiver is used.

D.11 PCB Layout and Mechanical Design

The physical unit must fit in a 1.187" × 7.0" rectangular hole in the instrument panel. The goal of the layout and mechanical design was to minimize the physical size of the unit and to maximize the ease of installation.

Two-board construction was needed because the vertical space (front panel board) barely fits all the switches and the OLED display. A horizontal board (main board) was added for the non-user-interface components. Figure D.15 shows the physical construction. Both the front panel and the main board are two layers. The main board has 2 oz copper for increased current capacity. The front panel board has regular 1 oz copper. They are connected with pin headers and sockets.

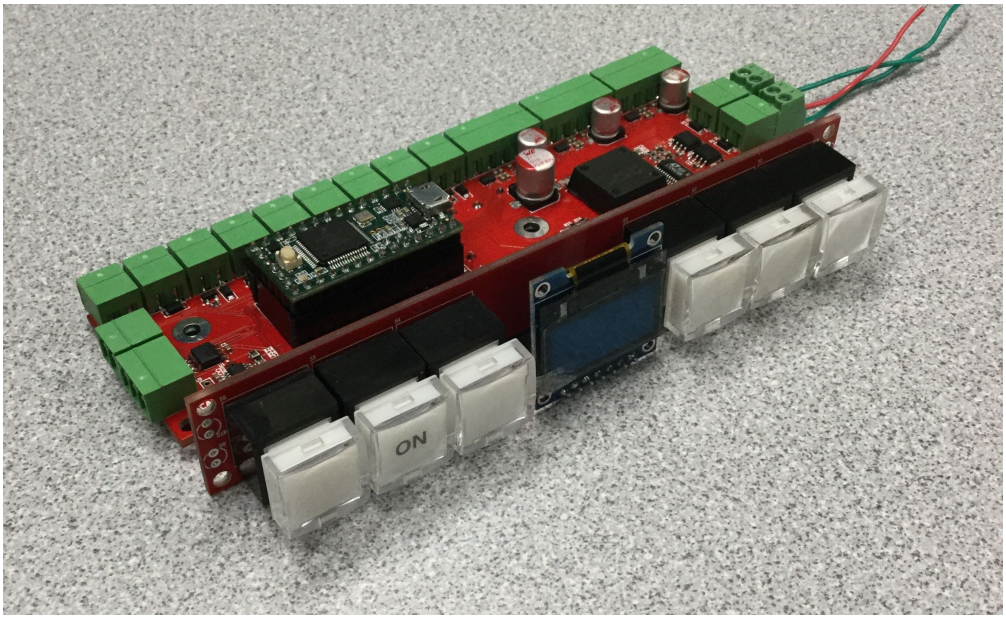


Figure D.15: Physical construction of the Safety Officer

The rear connectors are Würth Elektronik WR-TBL Series 3211 pluggable screw terminals (Figure D.16). They simplify the wiring process, and they can be easily unplugged when needed.

Figure D.17 and D.18 shows the layout of the two PCB. Good thermal management, decoupling, and grounding practices were obeyed.

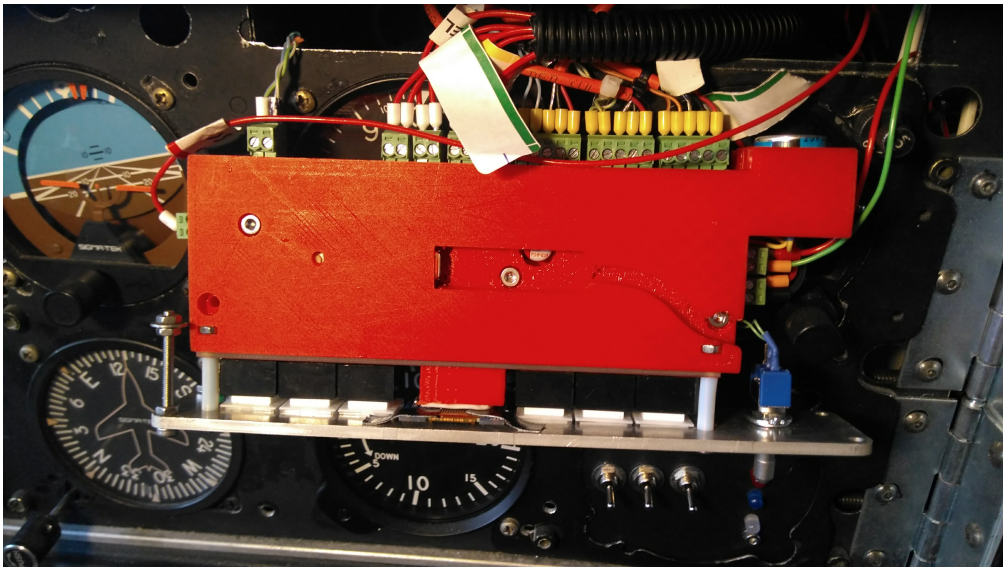


Figure D.16: Pluggable screw terminals on the Safety Officer

The PCBs were hand pick-and-placed and reflowed in a toaster oven.

The CAD model (Figure D.19) of the PCB was imported from Altium Designer into Autodesk Fusion 360 to design the enclosure and the front panel. A 3D printed plastic enclosure and a CNC milled aluminum front panel was designed and manufactured. 3D printed plastic enclosure allows complex geometry and provides maximum support for the circuit board and components. Strain relief was added to support the USB cable. The front panel was sandblasted for a non-glaring finish. Figure D.20 shows the finished product.

D.12 Embedded Software Design

The embedded software has a timer ISR running every 10 ms for the most critical tasks, including the FSMs detailed in Section D.4, the closed-loop motor control described in Section D.6, USB communication, and fault condition detection. All push buttons are handled in higher-priority interrupts. CAN communication is handled in interrupt. The main loop communicates with the display and lights.

D.13 Test Results

The Safety Officer consistently disconnects all actuators listed in Table D.1 within 500 ms under full load. The redundant power, user interface, and communication functioned as designed.

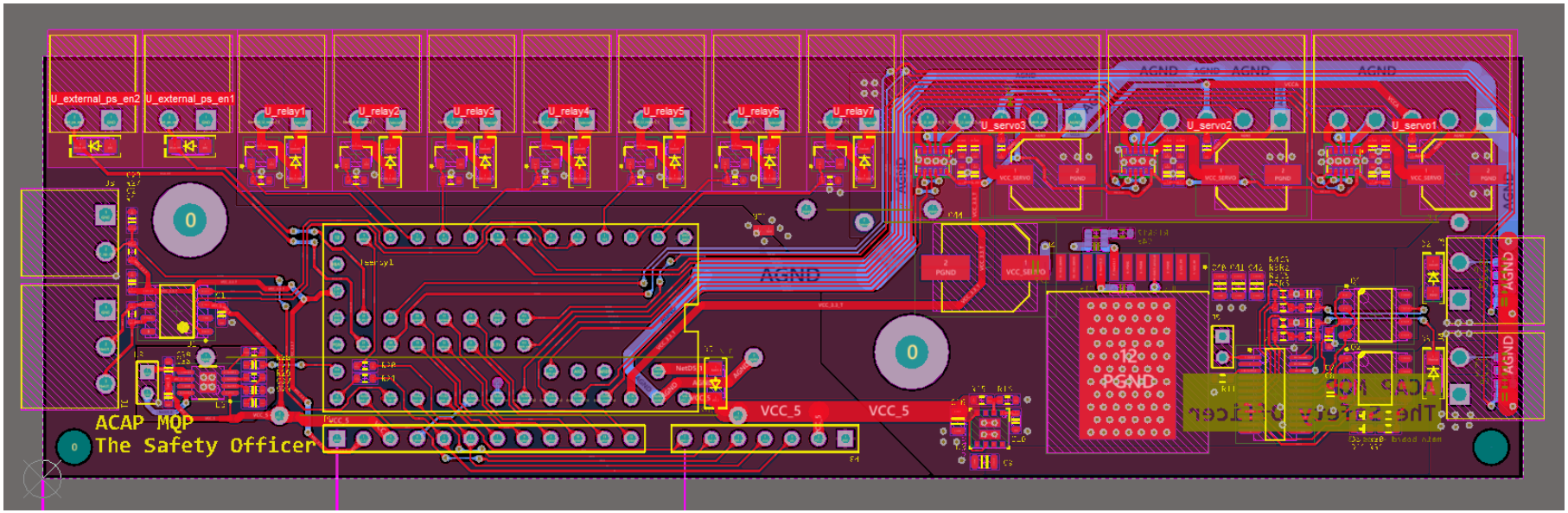


Figure D.17: Layout of the main board

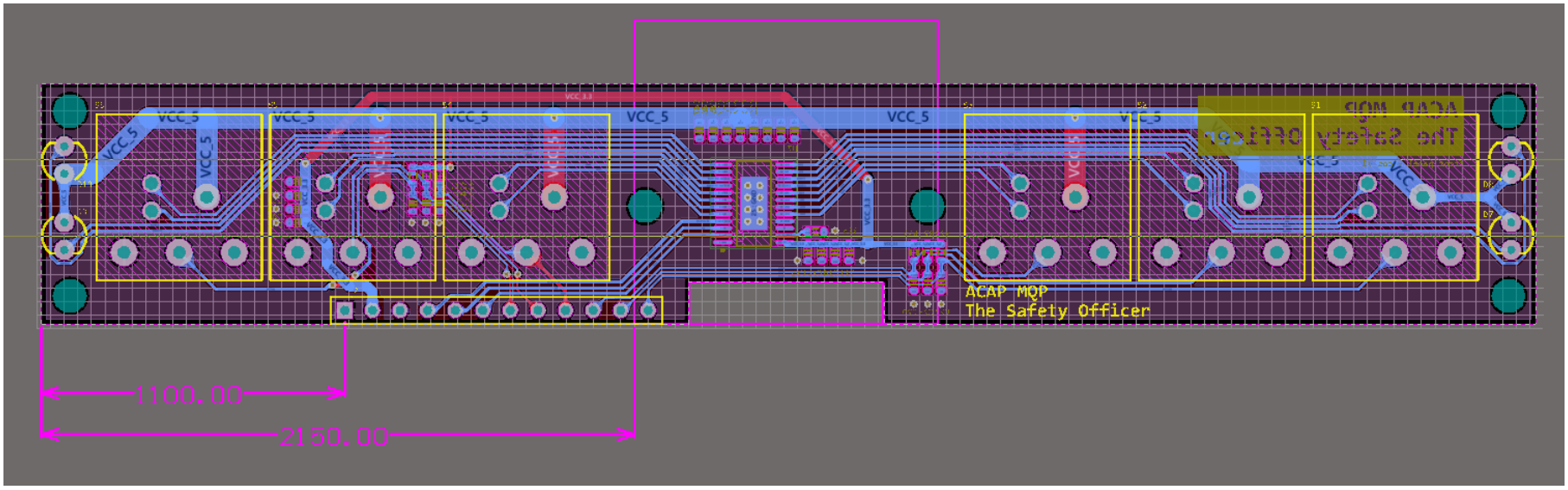


Figure D.18: Layout of the front panel board

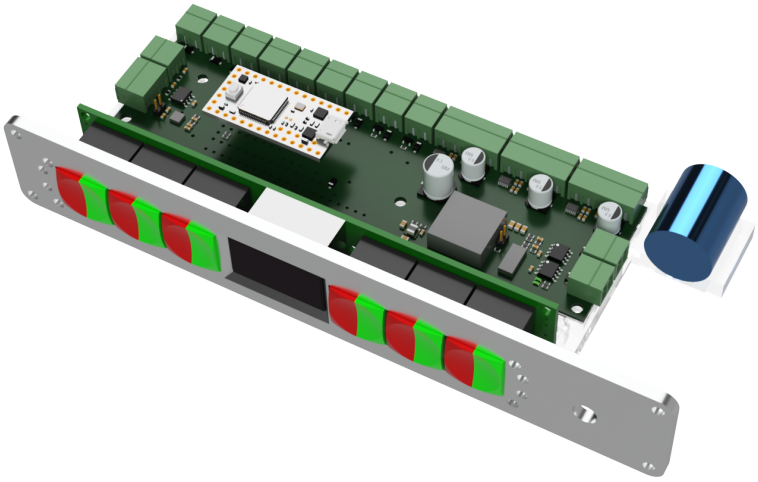


Figure D.19: CAD model of the Safety Officer

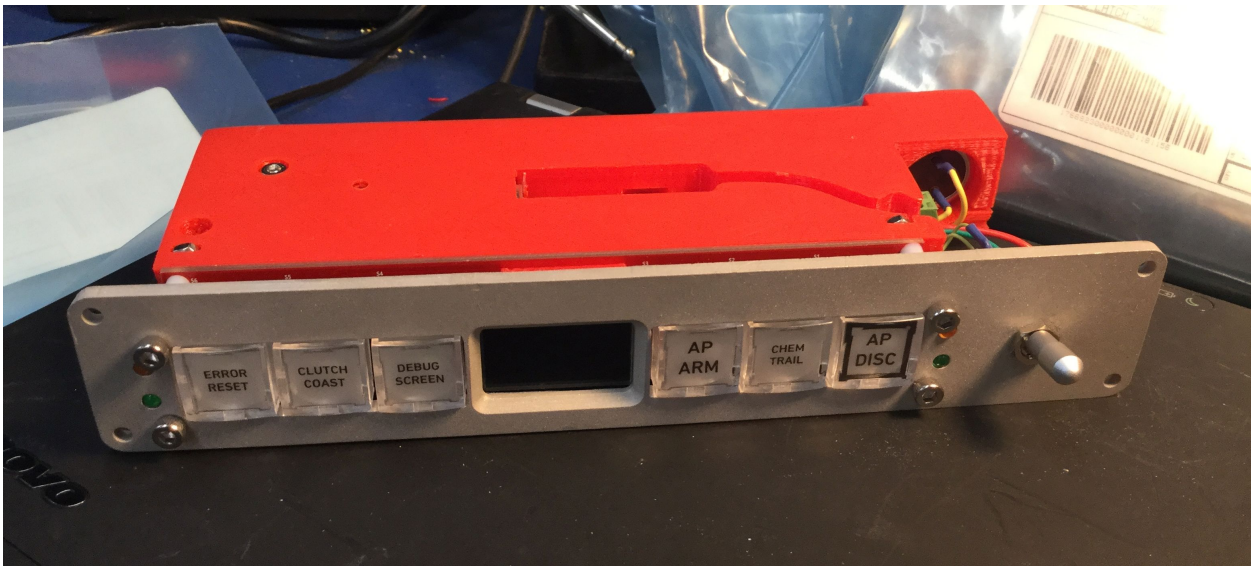


Figure D.20: Assembled Safety Officer

D.14 PCB and Software Sources

The PCB and related documents can be found at
`https://github.com/urill/acap-safetyofficer`
The embedded software source code can be found at
`https://github.com/urill/acap-safety-officer-firmware`

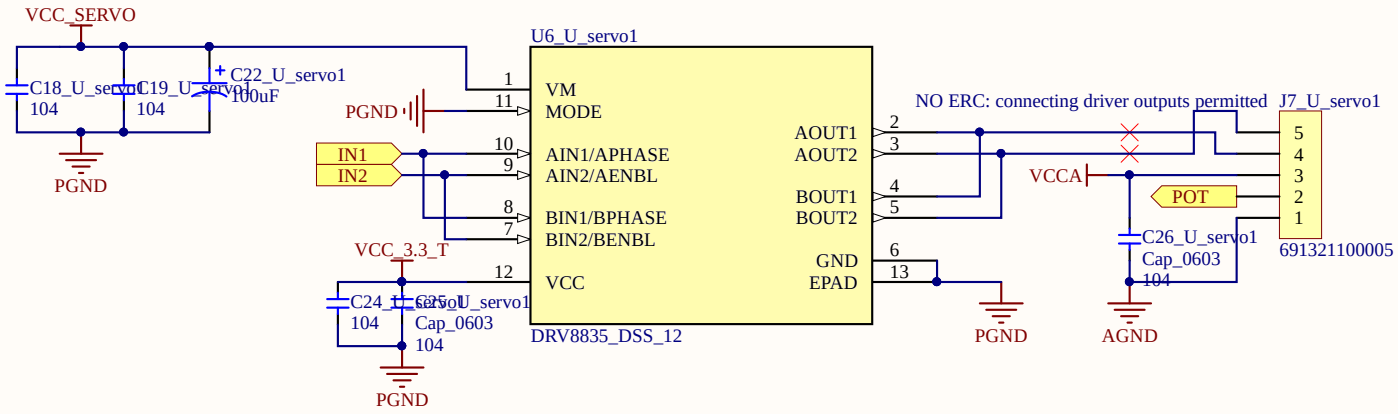
Bibliography

[1] (2016). Pololu 5v, 5a step-down voltage regulator d24v50f5, [Online]. Available: `https://www.pololu.com/product/2851`.

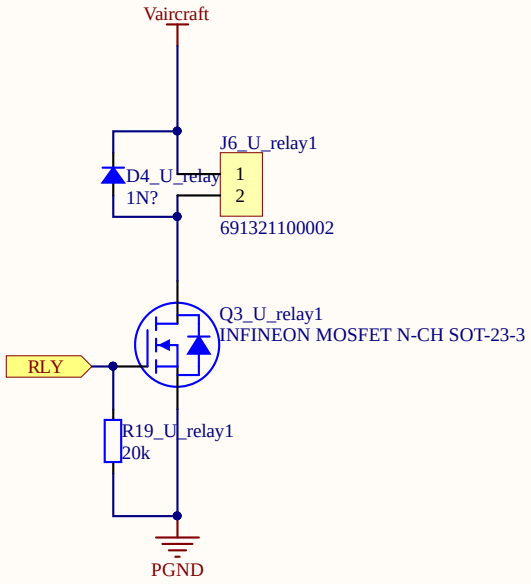
[2] *Drv8835 dual low-voltage h-bridge ic*, DRV8835, REVISED MAY 2016, Texas Instruments, May 2012.

[3] *K20 sub-family*, K20P64M72SF1, Rev. 3, Freescale Semiconductor, Nov. 2012.

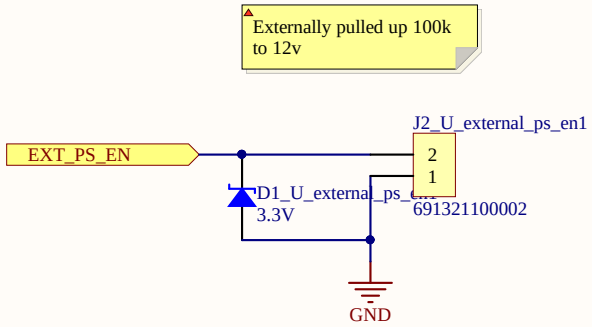
[4] *Ltc4417 prioritized powerpath controller*, LTC4417, LT 1112, Linear Technology, 2012.



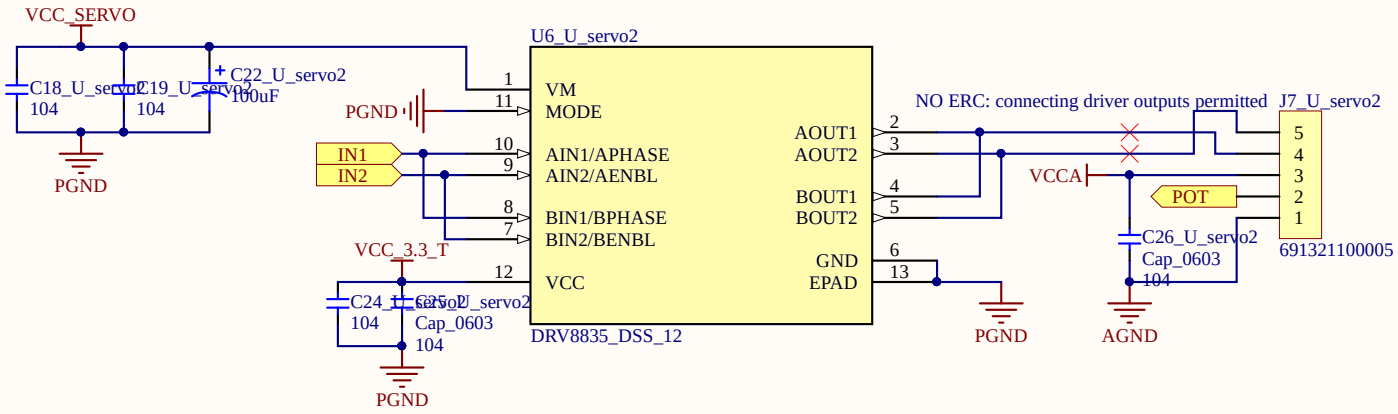
Title: Gutted servo driver + pot connection		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\servo.SchDoc		Drawn By:



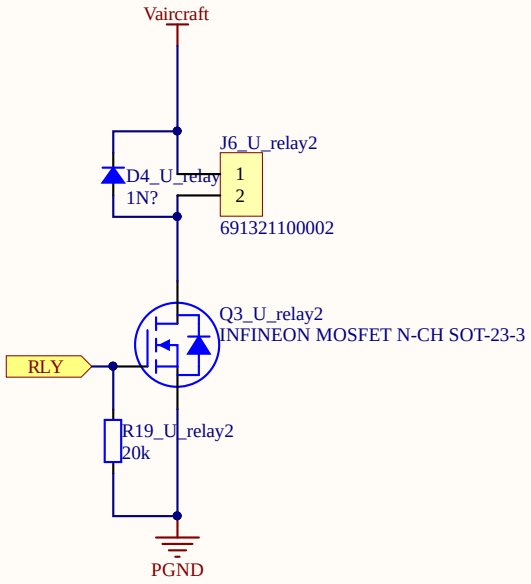
Title: Relay driver		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\relay.SchDoc		Drawn By:



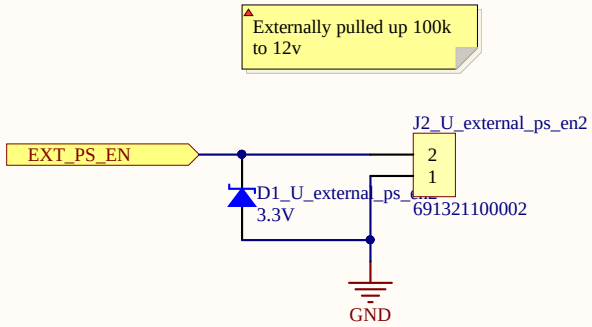
Title		
Size: A	Number	Revision
Date: 1/20/2017		Sheet of
File: C:\Users\..\external_ps_en.SchDoc		Drawn By:



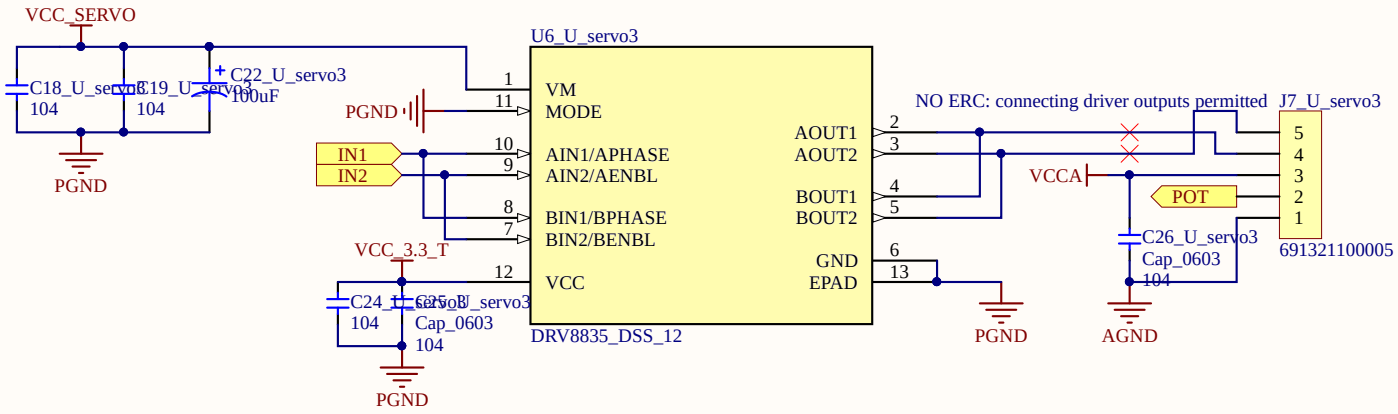
Title: Gutted servo driver + pot connection		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\servo.SchDoc		Drawn By:



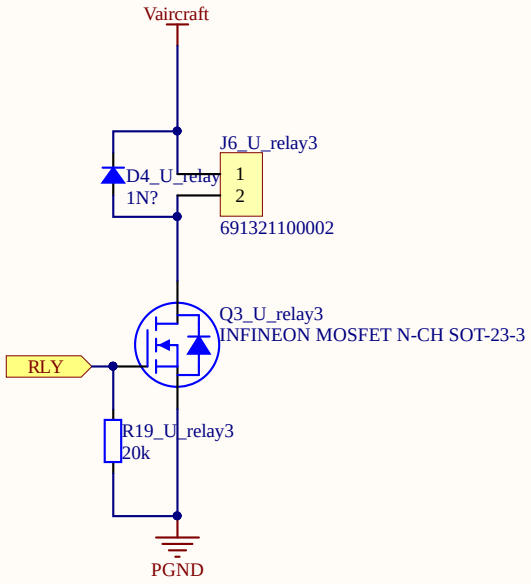
Title: Relay driver		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\relay.SchDoc		Drawn By:



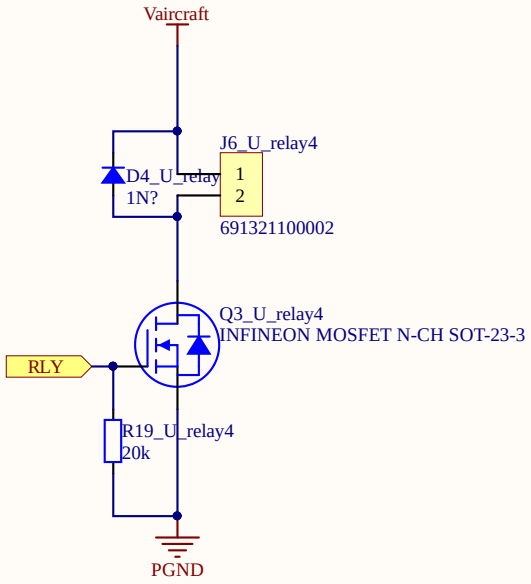
Title		
Size: A	Number	Revision
Date: 1/20/2017		Sheet of
File: C:\Users\..\external_ps_en.SchDoc		Drawn By:



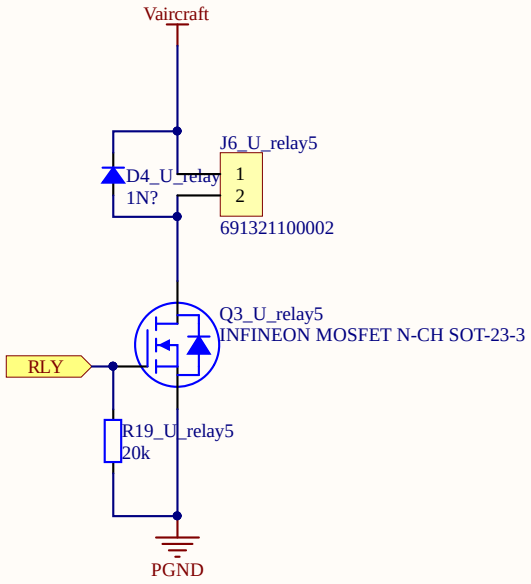
Title: Gutted servo driver + pot connection		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\servo.SchDoc		Drawn By:



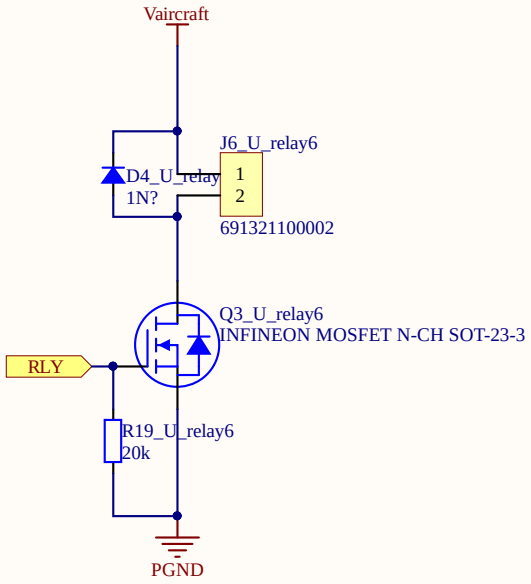
Title: Relay driver		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\relay.SchDoc		Drawn By:



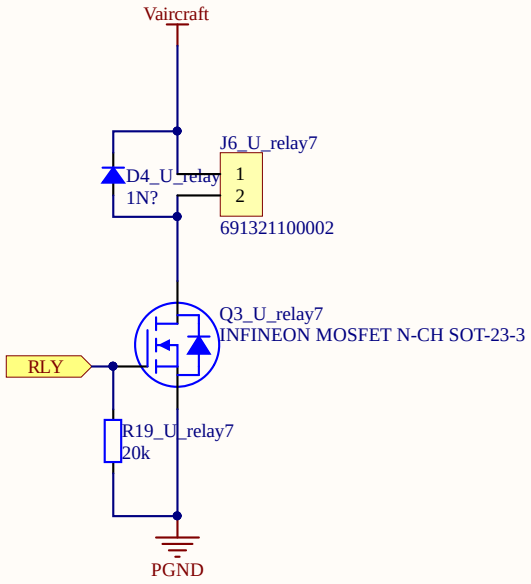
Title: Relay driver		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\relay.SchDoc		Drawn By:



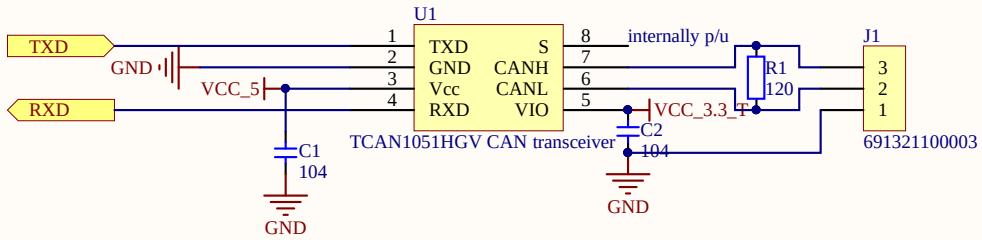
Title: Relay driver			
Size: A	Number:		Revision:
Date: 1/20/2017		Sheet of	
File: C:\Users\..\relay.SchDoc		Drawn By:	



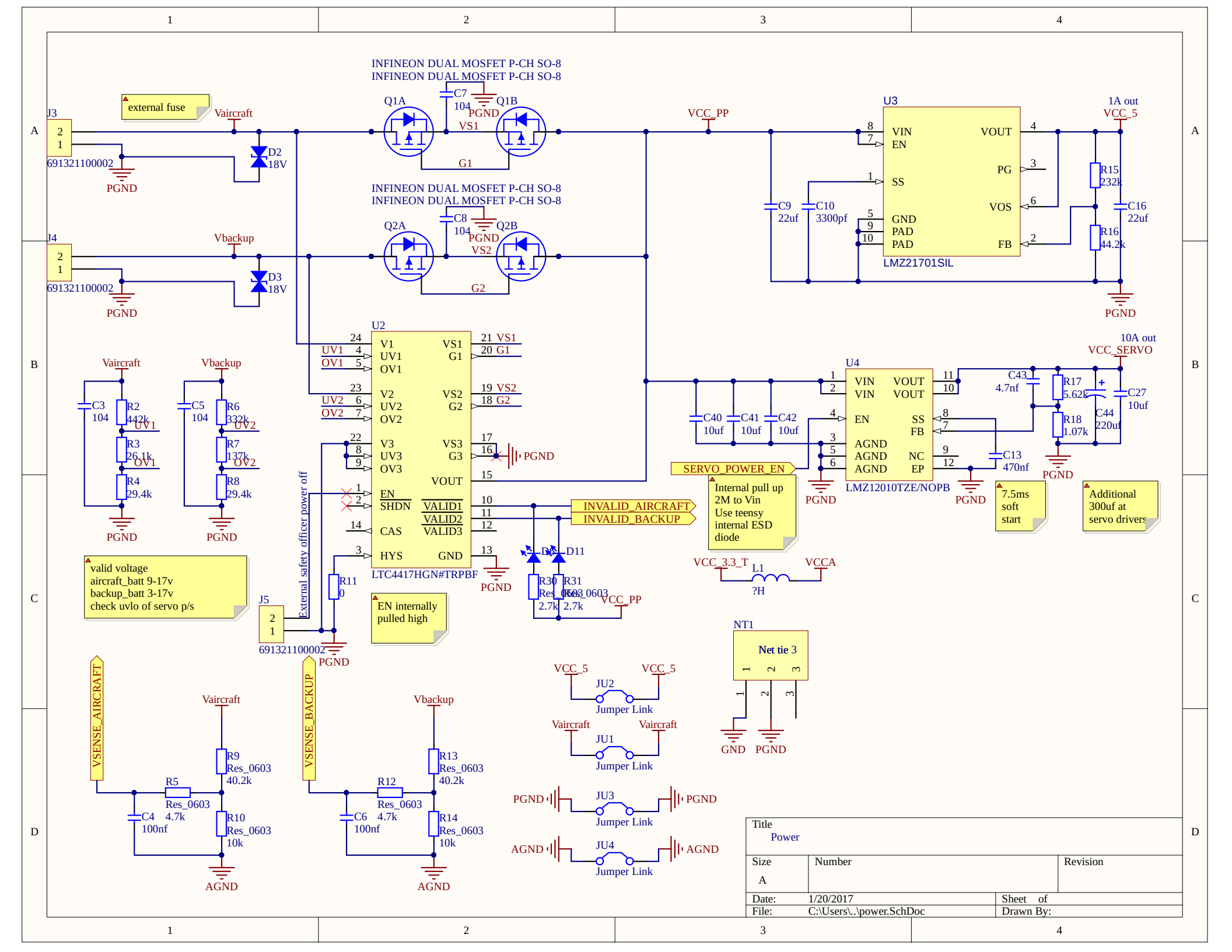
Title: Relay driver		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\relay.SchDoc		Drawn By:

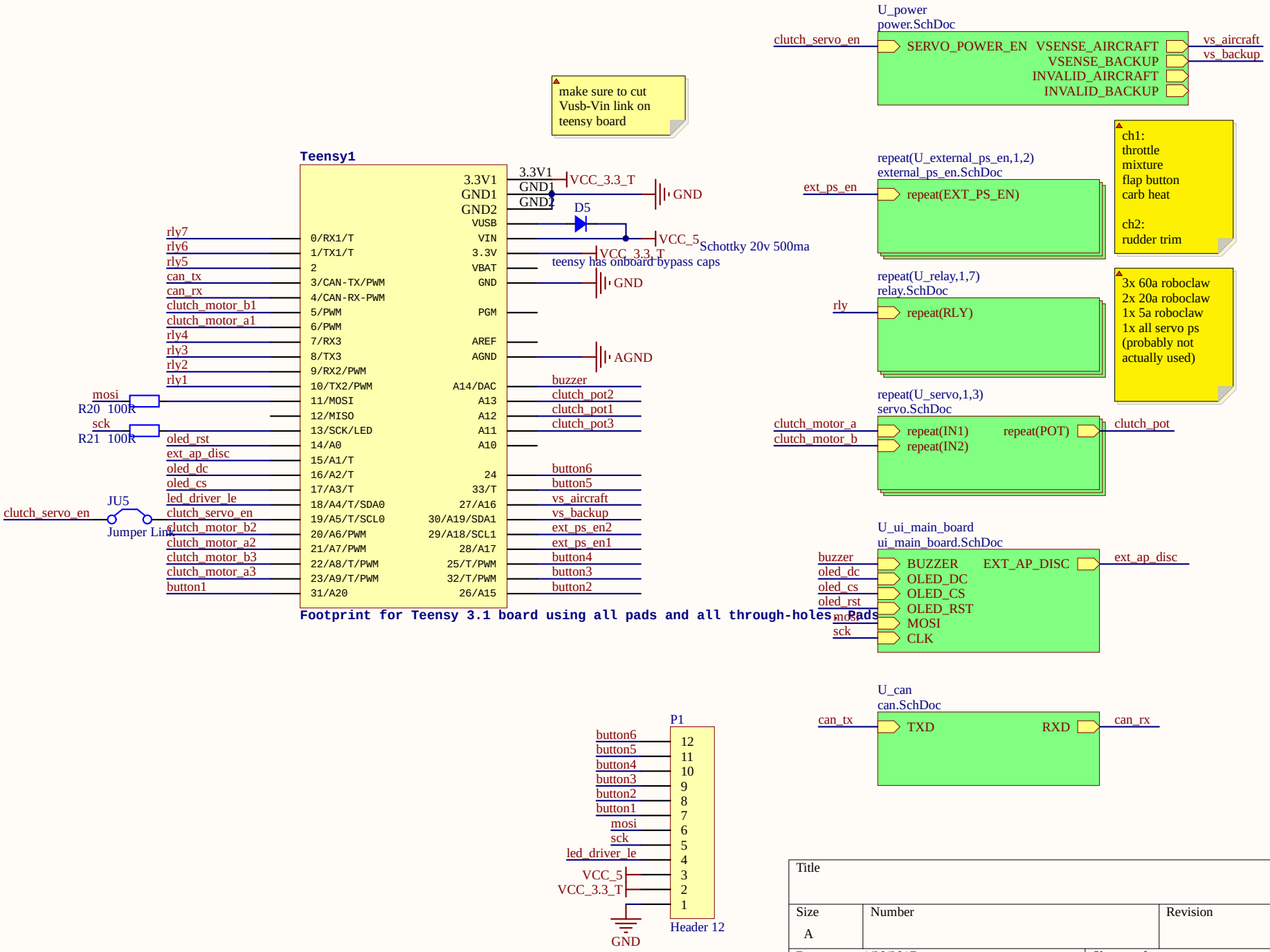


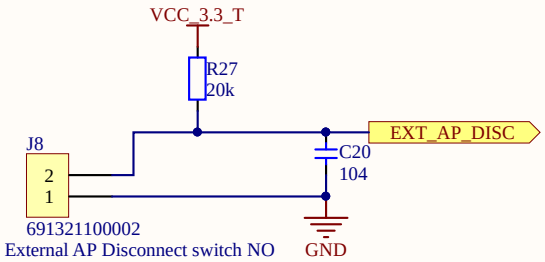
Title: Relay driver		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\relay.SchDoc		Drawn By:

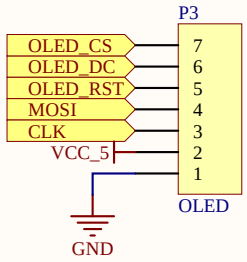


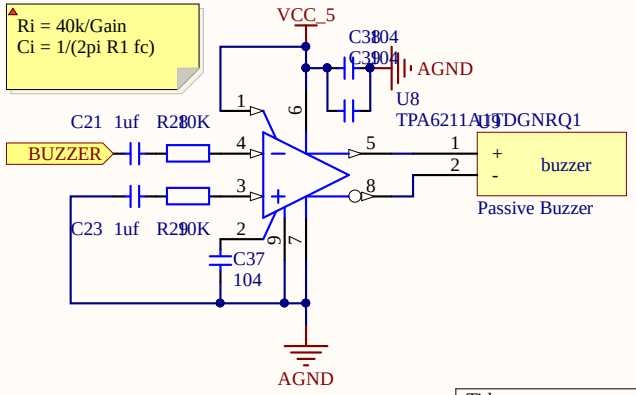
Title		
Size: A	Number	Revision
Date: 1/20/2017		Sheet of
File: C:\Users\..\can.SchDoc		Drawn By:


INFINEON DUAL MOSFET P-CH SO-8
INFINEON DUAL MOSFET P-CH SO-8
Q1A
Q1B
C7
104
PGND
VS1
G1
INFINEON DUAL MOSFET P-CH SO-8
INFINEON DUAL MOSFET P-CH SO-8
Q2A
Q2B
C8
104
PGND
VS2
G2
external fuse
J3
2
1
691321100002
PGND
Vaircraft
D2
18V
J4
2
1
691321100002
PGND
Vbackup
D3
18V
VCC_PP
U3
VIN
EN
SS
GND
PAD
PAD
VOUT
PG
VOS
FB
LMZ21701SIL
C9
22uf
C10
3300pf
R15
232k
R16
44.2k
C16
22uf
1A out
VCC_5
PGND
U2
V1
UV1
OV1
V2
UV2
OV2
V3
UV3
OV3
EN
SHDN
CAS
HYS
VS1
G1
VS2
G2
G3
VOUT
VALID1
VALID2
VALID3
GND
LTC4417HGN#TRPBF
PGND
INVALID_AIRCRAFT
INVALID_BACKUP
Vaircraft
Vbackup
C3
104
R2
442k
UV1
R3
26.1k
OV1
R4
29.4k
C5
104
R6
332k
UV2
R7
137k
OV2
R8
29.4k
valid voltage
aircraft_batt 9-17v
backup_batt 3-17v
check uvlo of servo p/s
External safety officer power off
J5
2
1
691321100002
R11
0
EN internally
pulled high
D10
D11
R30
R31
2.7k
2.7k
VCC_PP
U4
VIN
VIN
EN
AGND
AGND
AGND
VOUT
VOUT
SS
FB
NC
EP
LMZ12010TZE/NOPB
C40
10uf
C41
10uf
C42
10uf
C43
4.7nf
C13
470nf
R17
5.62k
R18
1.07k
C44
220uf
C27
10uf
10A out
VCC_SERVO
SERVO_POWER_EN
Internal pull up
2M to Vin
Use teensy
internal ESD
diode
7.5ms
soft
start
Additional
300uf at
servo drivers
VCC_3.3_T
L1
?H
VCCA
NT1
Net tie 3
GND
PGND
VSENSE_AIRCRAFT
VSENSE_BACKUP
R5
Res_0603
4.7k
R9
Res_0603
40.2k
C4
100nf
R10
Res_0603
10k
AGND
R12
Res_0603
4.7k
R13
Res_0603
40.2k
C6
100nf
R14
Res_0603
10k
AGND
VCC_5
VCC_5
JU2
Jumper Link
Vaircraft
Vaircraft
JU1
Jumper Link
PGND
PGND
JU3
Jumper Link
AGND
AGND
JU4
Jumper Link
Title
Power
Size
A
Number
Revision
Date:
1/20/2017
Sheet of
File:
C:\Users\..\power.SchDoc
Drawn By:

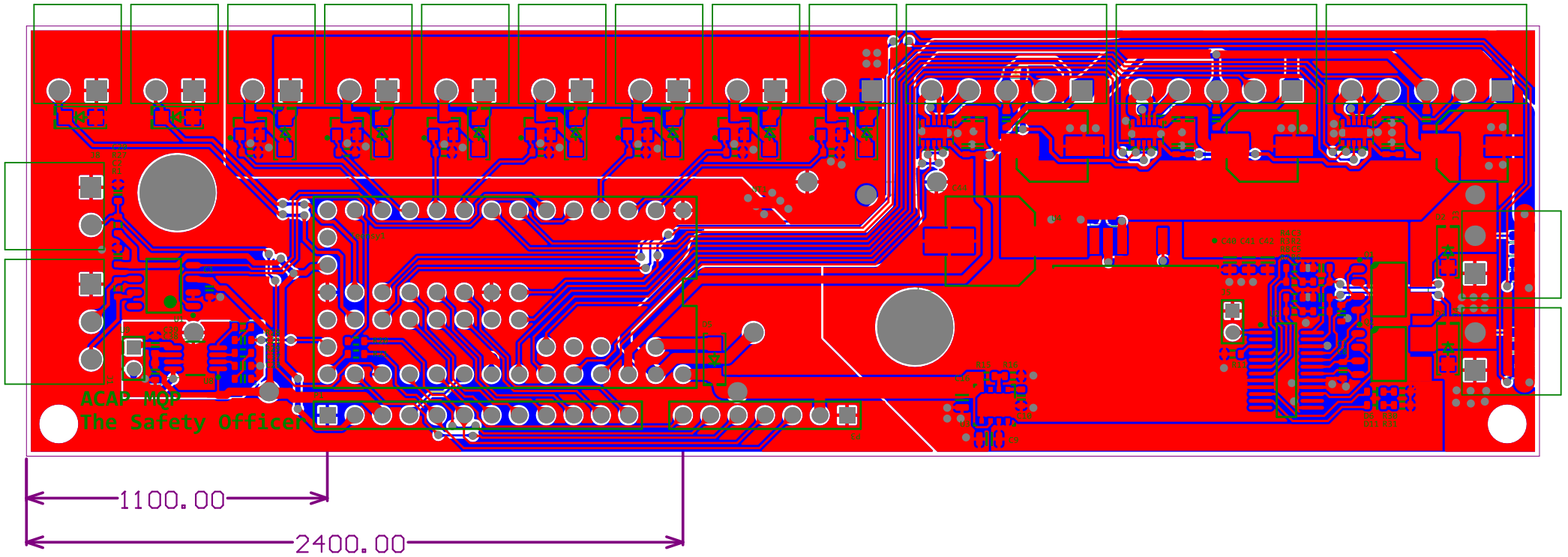

make sure to cut
Vusb-Vin link on
teensy board
Teensy1
3.3V1
GND1
GND2
VUSB
VIN
3.3V
VBAT
GND
PGM
AREF
AGND
A14/DAC
A13
A12
A11
A10
24
33/T
27/A16
30/A19/SDA1
29/A18/SCL1
28/A17
25/T/PWM
32/T/PWM
26/A15
0/RX1/T
1/TX1/T
2
3/CAN-TX/PWM
4/CAN-RX-PWM
5/PWM
6/PWM
7/RX3
8/TX3
9/RX2/PWM
10/TX2/PWM
11/MOSI
12/MISO
13/SCK/LED
14/A0
15/A1/T
16/A2/T
17/A3/T
18/A4/T/SDA0
19/A5/T/SCL0
20/A6/PWM
21/A7/PWM
22/A8/T/PWM
23/A9/T/PWM
31/A20
VCC_3.3_T
GND
D5
VCC_5
Schottky 20v 500ma
VCC_3.3_T
teensy has onboard bypass caps
GND
AGND
rly7
rly6
rly5
can_tx
can_rx
clutch_motor_b1
clutch_motor_a1
rly4
rly3
rly2
rly1
mosi
R20 100R
sck
R21 100R
oled_rst
ext_ap_disc
oled_dc
oled_cs
led_driver_le
clutch_servo_en
clutch_motor_b2
clutch_motor_a2
clutch_motor_b3
clutch_motor_a3
button1
JU5
clutch_servo_en
Jumper Link
buzzer
clutch_pot2
clutch_pot1
clutch_pot3
button6
button5
vs_aircraft
vs_backup
ext_ps_en2
ext_ps_en1
button4
button3
button2
Footprint for Teensy 3.1 board using all pads and all through-holes Pads
P1
button6
button5
button4
button3
button2
button1
mosi
sck
led_driver_le
VCC_5
VCC_3.3_T
GND
12
11
10
9
8
7
6
5
4
3
2
1
Header 12
U_power
power.SchDoc
clutch_servo_en
SERVO_POWER_EN
VSENSE_AIRCRAFT
VSENSE_BACKUP
INVALID_AIRCRAFT
INVALID_BACKUP
vs_aircraft
vs_backup
repeat(U_external_ps_en,1,2)
external_ps_en.SchDoc
ext_ps_en
repeat(EXT_PS_EN)
ch1:
throttle
mixture
flap button
carb heat
ch2:
rudder trim
repeat(U_relay,1,7)
relay.SchDoc
rly
repeat(RLY)
3x 60a roboclaw
2x 20a roboclaw
1x 5a roboclaw
1x all servo ps
(probably not
actually used)
repeat(U_servo,1,3)
servo.SchDoc
clutch_motor_a
clutch_motor_b
repeat(IN1)
repeat(IN2)
repeat(POT)
clutch_pot
U_ui_main_board
ui_main_board.SchDoc
buzzer
oled_dc
oled_cs
oled_rst
mosi
sck
BUZZER
OLED_DC
OLED_CS
OLED_RST
MOSI
CLK
EXT_AP_DISC
ext_ap_disc
U_can
can.SchDoc
can_tx
TXD
RXD
can_rx
Title
Size
A
Number
Revision
Date: 1/20/2017
Sheet of
File: C:\Users\..\teensy.SchDoc
Drawn By:

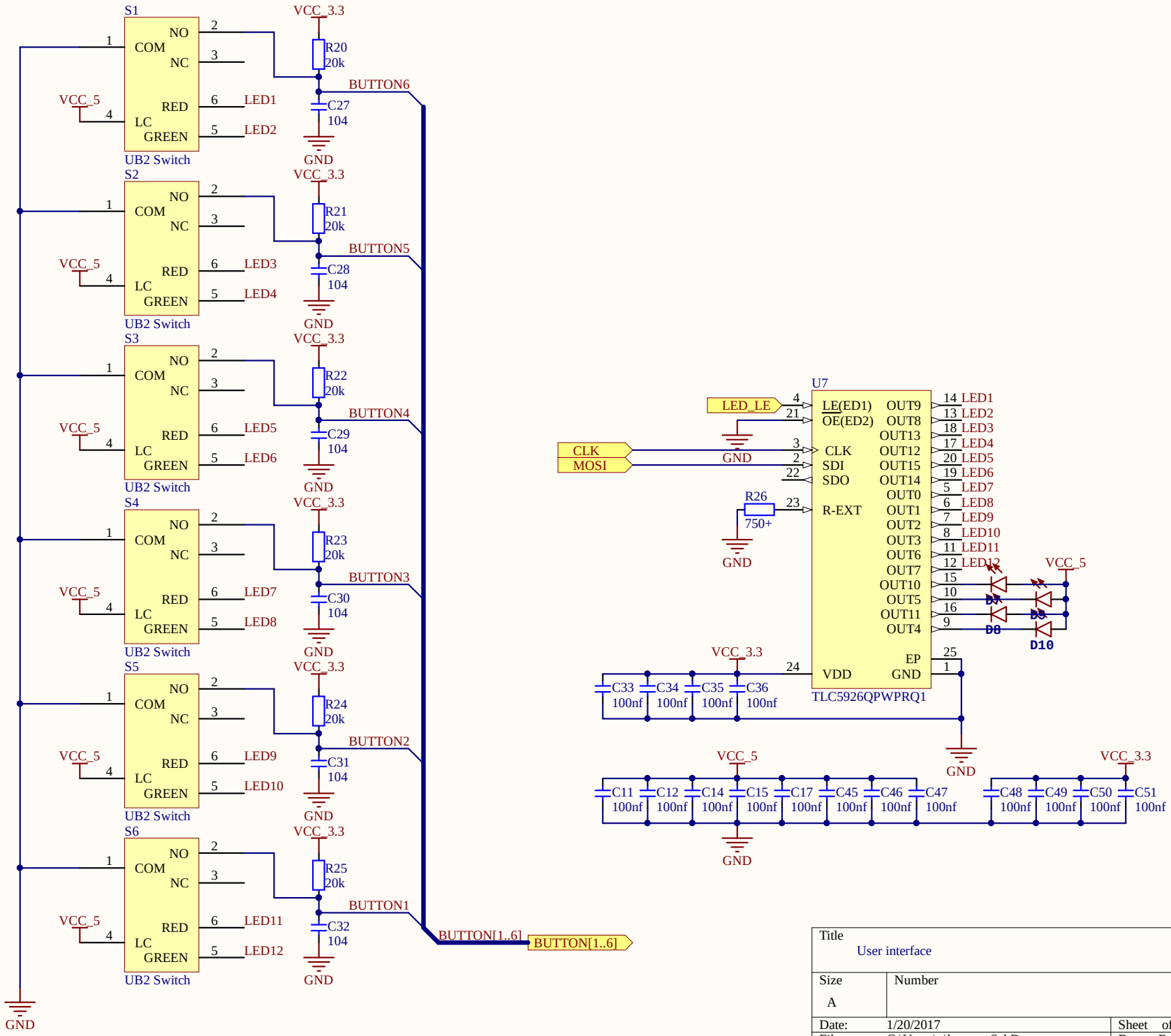




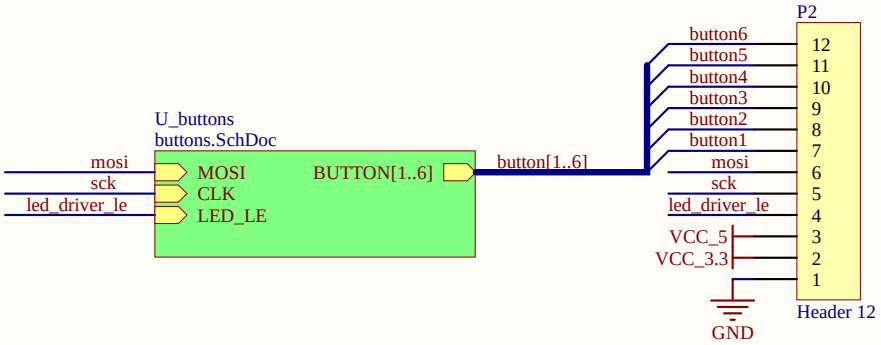


Title		
Size: A	Number	Revision
Date: 1/20/2017		Sheet of
File: C:\Users\..\ui_main_board.SchDoc		Drawn By:

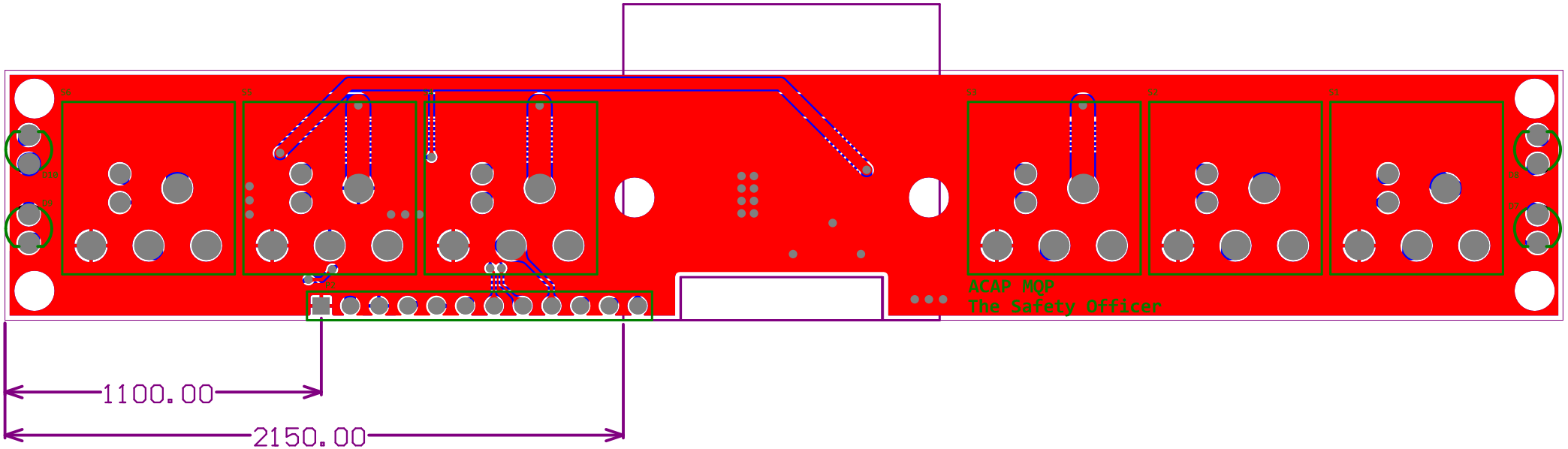

ACAP MQP
The Safety Officer
1100.00
2400.00



Title: User interface		
Size: A	Number:	Revision:
Date: 1/20/2017		Sheet of
File: C:\Users\..\buttons.SchDoc		Drawn By:



Title			
Size A	Number		Revision
Date:	1/20/2017	Sheet of	
File:	C:\Users\..\frontpanel_interconnect.SchDoc	Drawn By:	


S6
S5
S4
S3
S2
S1
D10
D9
D8
D7
J2
ACAP MQP
The Safety Officer
1100.00
2150.00

Appendix E

FMC Backend

Written By: Erik Nadel, Nicholas Cyganski

Table of Contents

E1 Overview

Written in java, the software backend for the ACAP project knits together components to achieve full control of the aircraft. By communicating with the electrical engineering work, the backend provides an interface for controlling the aircraft through software. The overall design goal for the backend is a modular, flexible, and robust system that allows components to be added or removed without instability or hassle. Modularity is achieved with abstraction, flexibility is achieved through dynamic components, and robustness is achieved through fail safes.

This project is intended to continue beyond the initial team's efforts. Being modular allows for new contributors to test new components without changing existing code. Another important aspect of the backend regarding new teams is the efficiency of bringing up a new development environment. The ACAP backend is bundled with all the resources required to run on a new system. All dependencies are downloaded with a single command and reduce the time spent setting up a new computer. Style checking is enforced through build scripts and keeps the code style consistent across commits. Utilities that enable quick development such as hot swapping are also bundled with the environment to prevent having to restart to see changes. Tied together with version control, the goal is to create an all-in-one system.

E 1.1 Flight Management Computer (FMC)

The program starts out by initializing the GLWorld, which is the 3d view seen on the left side of the FMC interface. The first object created by the FMC is the UserAircraft. The UserAircraft singleton is used throughout the FMC to control devices, manage sensor readings, and access the mode control panel input. Once the GLWorld is loaded, the core logic threads are started. These include the control scheduler, message handler, spatial sensors, GPS, audio cue handler, and state machine. Once started, these threads handle their own logic and the world drawing is handled in the GLWorld.

E 1.1.1 Control Scheduler

The Control Scheduler thread iterates through the task queue and runs each group of tasks given by order of priority. These tasks are created through the Tasks tab by clicking on the + button. They can also be added programmatically.

E 1.1.2 Message Handler

The message handler, otherwise known as the SubscribeThread, interprets received messages. The messages are received through the locally running ZMQ instance. By default, this ZMQ instance is running on localhost bound to port 5560.

Messages are handled based on their type. The Message handler loads handlers from the acap.fmc.procedure.messages package. Each handler implements the IMessageHandler class. The corresponding message handler for an incoming message stores the received data and runs the action defined in the message handler. For example, the ControlMessageHandler interprets all sensor messages from the aircraft and sends the readings to the proper device specified in the message.

E 1.1.3 Spatial Sensors/GPS

Currently, we use Phidget USB sensors to get GPS readings, pitch, and roll. The spatial sensor threads take data from these sensors and put them through a Kalman filter. After filtering the sensor data, the refined data is stored in the UserAircraft class.

E 1.1.4 Audio Cues (Aural Handler)

The Aural handler manages audio cues that need to be played during any phase of flight. The thread loads all classes found in the fmc.procedure.aural package. These classes implement a common IAudioCue interface. The aural handler thread then checks the condition of each loaded cue and will play the corresponding audio file.

E 1.1.5 State Machine

The state machine is responsible for loading routines related to each phase of flight. These routines are user defined and can be found in acap.fmc.state.fsm.routines. At the transition of each state, the state machine thread checks the classes found in the routines package and kicks off any IRoutine that returns a relevant state that matches the next state. The routine is then executed either one time, forever (until the FMC shuts down), or until a specific state is reached. Example routines can be found in the routines package.

E2 Design Methodology

For this project, we started from an existing simulation codebase developed by Nicholas Cyganski. From the existing simulation, we refactored the project into a more extensible and managed codebase. This project resides in the world package and is used to provide visuals to the flight engineer.

The refactoring was broken down into five steps: integrating version control, decoupling the codebase from the Processing IDE, adding dependency management, applying the MVC (model view controller) pattern, and finally general cleanup. Adding version control was a critical part of the refactor. Without proper version control, there is a difficulty in merging different feature changes from multiple users and keeping track of breaking changes/bugs. Our choice of version control was git because of its ease of use and large user base. The second step of decoupling the codebase from the Processing IDE gives the team more control over the execution of the sim and the libraries used. The Processing IDE provides a wrapper that allows for easy execution of java programs that use the Processing library by adding the proper class path parameters needed to link the Processing library. However, the drawback of the added convenience is that using the Processing IDE does not support the java source extension and cannot be managed like a java project typically would. As a result, we ended up with a purely Java codebase that does not rely on Processing IDE to run.

After decoupling the Processing IDE, the next step was to add dependency management. For the project, we chose to use Gradle as our dependency manager. The dependency management helps the development team keep track of dependencies, install dependencies, and locate breaking dependency updates. Adding any dependencies to the codebase would then only require one additional line in the dependency manager. The added benefit is that updating additional changes is done in one command. This one command philosophy applies to the installation of dependencies as well.

E 2.1 Organization

After integrating dependency management, reorganizing the code base was the next step. The state of the codebase at the start of the project was functional, but not organized. Organizing this codebase was critical to any future development because it allows for easy understanding of

what role a class has. Another benefit of organization is the ability for the non-computer science members to be able to suggest features, make comments on structure, and critique pending changes to the backend.

To keep the repository organized, the backend was separated into two major packages, fmc (flight management computer) and world. The world package contains the original (refactored) simulation code and the fmc package houses the logic for route calculation, communication with electrical components, and any other task that assists the aircraft in flight. Some of the models in the world are used in the fmc due to needing certain models such as airports and runways to calculate routes.

E3 Class Organization

One contributor to the extensibility of a project is clear class organization. For the flight management computer, the packages were separated by their functionality. Classes organized by their purpose allows for easy debugging and ease of navigation throughout the code base. For example, any routing functionality is in the routing package. Cases where a class is utilized for multiple core functions are placed into a utility package. While helper classes are considered a bad "smell" [1] [2], writing these helper classes following an OO paradigm alleviates the procedural programming smell of utility classes.

There are two root packages: *fmc* (flight management computer) and *world*. The *fmc* package encompasses all classes related to collecting data from the aircraft and sending data to the aircraft. The *world* package consists of all classes related to the virtual world that the pilot sees on the flight computer display.

E 3.1 Flight Management Package

The flight management package is separated into seven packages: controls, gui, hid, procedure, routing, sensors, and state. Each of these packages contains a set of classes for performing a functionality related to its parent package name. The classes that exist outside of these sub-packages contain constants used throughout the backend of the flight management computer and the singleton class that represents the aircraft being controlled by the flight management computer.

E 3.1.1 Controls

The controls package contains classes for communicating with the controls on the aircraft. The roles vary from generating control messages to setting the readings received from sensors in the aircraft. This package also contains the logic for executing tasks found by the control scheduler.

E 3.1.2 HID (Human Input Devices)

The HID package contains all logic for HID that can be connected to the flight management computer. Currently, the only supported input device is a joystick.

E 3.1.3 GUI

The GUI package contains the framework for the frontend of the flight management computer. This package also contains the applets used to display the status of the individual flight controls. The frontend also serves as a location for the user to input parameters for route calculation and toggle settings for the auto pilot.

E 3.1.4 Procedure

The procedure package contains classes that handle the logic for changing phases of flight. The conditions to check for changing phases are also handled in the procedure package. Along with the finite state machine, the procedure package contains the definition of the Task class. The classes that handle sending and receiving control/reading messages are also found in the procedure package. The difference from the controls package is that the logic in procedure only receives the messages, but the controls package handles the decoding of the message and passing the readings data to the appropriate location.

E 3.1.5 Routing

The logic for calculating ground and air routes is found in the routing package. The routing utilities are exposed to the user through the user interface of the flight management computer where the user enters taxi instructions and a specified route. Combining the outputs of the air route and ground waypoints, the route is displayed through the virtual world showing a path that the aircraft is expected to take.

E 3.1.6 Sensors

The sensors package handles the interfacing between USB sensors attached to the flight management computer and directs their data to corresponding variables in system. Any data retrieved through these sensors passes through a Kalman filter. This filter helps to reduce outliers and show smooth data. The classes within this package update data by calling the appropriate method from the user aircraft singleton that corresponds with the variable being updated by a sensor.

E 3.1.7 State

The state package holds the singleton classes that manage the state of the overall aircraft. Any variables used in routing calculation, performance calculation, and real time navigation are located within this package. A singleton is used to manage the state because several packages require access to the current state of the aircraft to perform tasks. The singleton pattern also gives the ability to properly control the setting of variables and easily reset the state if needed by recycling the instantiated object. Rather than having a class of static variables accessed through static methods, the singleton approach preserves the object-oriented focus in the backend design.

E 3.1.8 Constants

Outside of the packages is a single class that contains constants. These constants are referenced during various duties of the backend. These constants also assist with development by giving the ability for a developer to turn off specific sensors or devices that might not be available on the development machine. This class also contains settings for the message communication that control where to read messages from and where to send messages to. Simply put, the constants class contains all the settings for the backend.

E 3.2 World Package

The world package contains classes related to drawing the virtual space and classes that represent real world objects. The world package is divided into three groups to represent a model view controller pattern. Additional packages include utilities and plugins used to add visuals to the world.

E 3.2.1 Controller

The controller package contains the logic for manipulating the camera that the flight management computer displays on the left side of the screen. The current controllers in the world include a mouse listener for manipulating the camera with the mouse and a controller for the list of airports displayed in the debug menu to set camera position based on the airport identifier.

E 3.2.2 Model

Any world related model is kept in the world model package. The models included in the world model package include aprons, runways, taxi nodes, aircraft, and any other element that is used to navigate the world. Models related to taxiways, such as the apron and taxi path, are used in constructing ground navigation routes. For air navigation routes, the runways and other models that implement the BaseAeroObject are used.

E 3.2.3 View

The view package contains the logic for drawing the classes found in the model package. At this point in time, the only models shown in the world are airports and aircraft. Additional views related to the models would be found here if added in the future.

E 3.2.4 Plugin

The plugin package was created for displaying data from the fmc package that is not related to the world itself. For example, a calculated ground route. Any Runnable class placed into this folder with a draw method will be added to the drawing loop and be shown to the user. Critical for modular development, having a plugin folder allows for developers to add visuals to the display without changing the existing codebase.

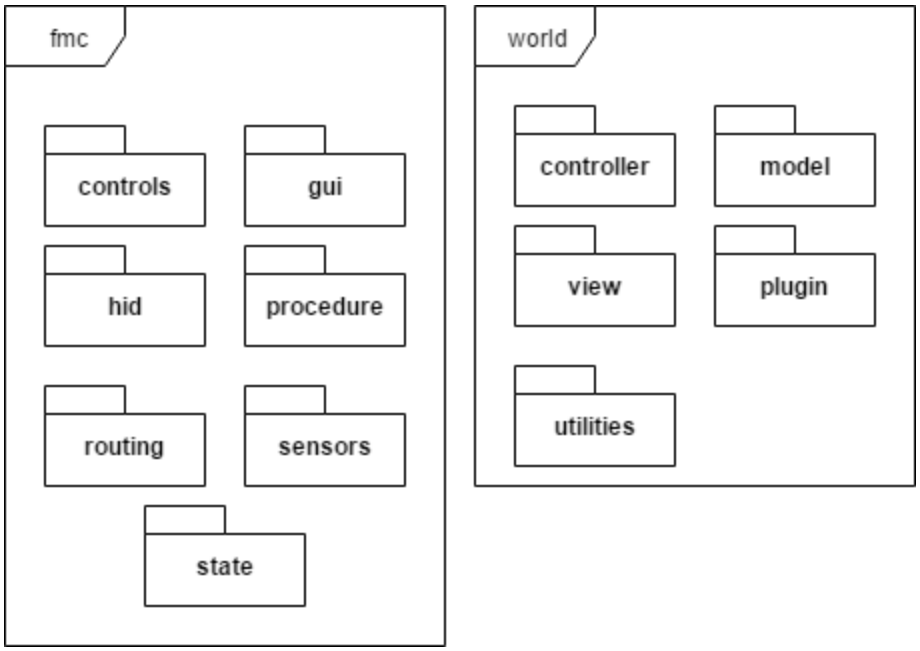


Figure 1E

Figure 1E shows an overview of all the packages that exist in the flight management computer.

E4 Procedure Stack

The procedure stack for the flight management computer is based on message passing. Any procedure that needs to be executed by a physical component is serialized using the Protobuf library into a message type recognizable by the software interfacing with the motors (FMC_IO). For debugging, the controls found in the debugger menu send messages instantly to the plane and set the position of a specified component accordingly. For real operation, the messages sent grouped up in task groups and executed by priority.

E 4.1 Components

E 4.1.1 Task Queue

The flight management computer maintains a singleton task queue that can be manipulated by any other class. The intent of the task queue is to be used globally by any class

such that new tasks can be added without any trouble.

E 4.1.2 Control Scheduler

The control scheduler is a soft time task scheduler. Upon startup, the control scheduler constantly loops with the role of retrieving the next task, executing the next task, and then waiting for the task to complete before moving on. The current set interval between iterations is 100ms. The current tasks handled by the control scheduler are implementations of our ITask interface which uses the Java runnable Interface as the parent class.

E 4.1.3 Tasks

Tasks handled by the control scheduler are implementations of the ITask interface. The ITask interface dictates how a task must be implemented. Each task in the control scheduler must have a method to check the status of the task, check whether or not the task is done, and the run method (inherited from the runnable type). The default task type takes in a device name and the desired percentage. The associated run method serializes and sends the message to the FMC_IO. The status checks for the default task are based off sensor readings from received messages. As of now, the status checks are used to show progress in the user interface.

E 4.1.4 Messages

The procedure stack on the FMC communicates with the microcontrollers via ZMQ using serialized messages. This message format is shown in Figure 2E The object is constructed using Protobuf generated classes and a string is generated from the constructed object.

```
message ControlMessage {
  enum component{
    ALL = 0;
    STABILIATOR = 1;
    RUDDER = 2;
    THROTTLE = 3;
    MIXTURE = 4;
    CARB_HEAT = 5;
    BRAKES = 6;
    PITCH_TRIM = 7;
    RUDDER_TRIM = 8;
    FLAPS = 9;
    LEFT_MAGNETO = 10;
    RIGHT_MAGNETO = 11;
    STARTER = 12;
    LANDING_LIGHT = 13;
    ROTATING_BEACON = 14;
    FUEL_PUMP = 15;
    AILERON = 16;
    FLIGHT_COMPUTER = 17;
    STABILATOR_TRIM = 18;
    NAVIGATION_LIGHT = 19;
    FLAP_BUTTON = 20;
    FLAP_HANDLE = 21;
  }

  component TargetComponent = 1;
  component SourceComponent = 2;
  double position = 3;
  bool isSetup = 4;
  double maxPosition = 5;
  double minPosition = 6;
  double velocity = 7;
  double smoothness = 8;

}
```

Figure 2E – Control Message Format Description

In a message, the intent is the source component to tell the target component what the position reading is or what position to set. A component is chosen from the component enumerator that is also located in the Protobuf definition. For example, if the flight engineer requested that the LEFT_MAGNETO was turned on, the source component would be the flight computer with a target of LEFT_MAGNETO. The indicated position would be set as 100% (On position). For sensor data, the target component is set as the flight computer instead. In the flight management computer, we assume all data with the target set as the flight computer to be sensor

readings. Using this assumption, we can ignore any message not targeted for the flight computer.

The other variables such as min/max position, velocity, and smoothness are used to setting up the target component. These parameters are for fine tuning the electrical components that move the plane's physical controls. The flow of these messages is described in Figure 3E – Flow of execution for Tasks

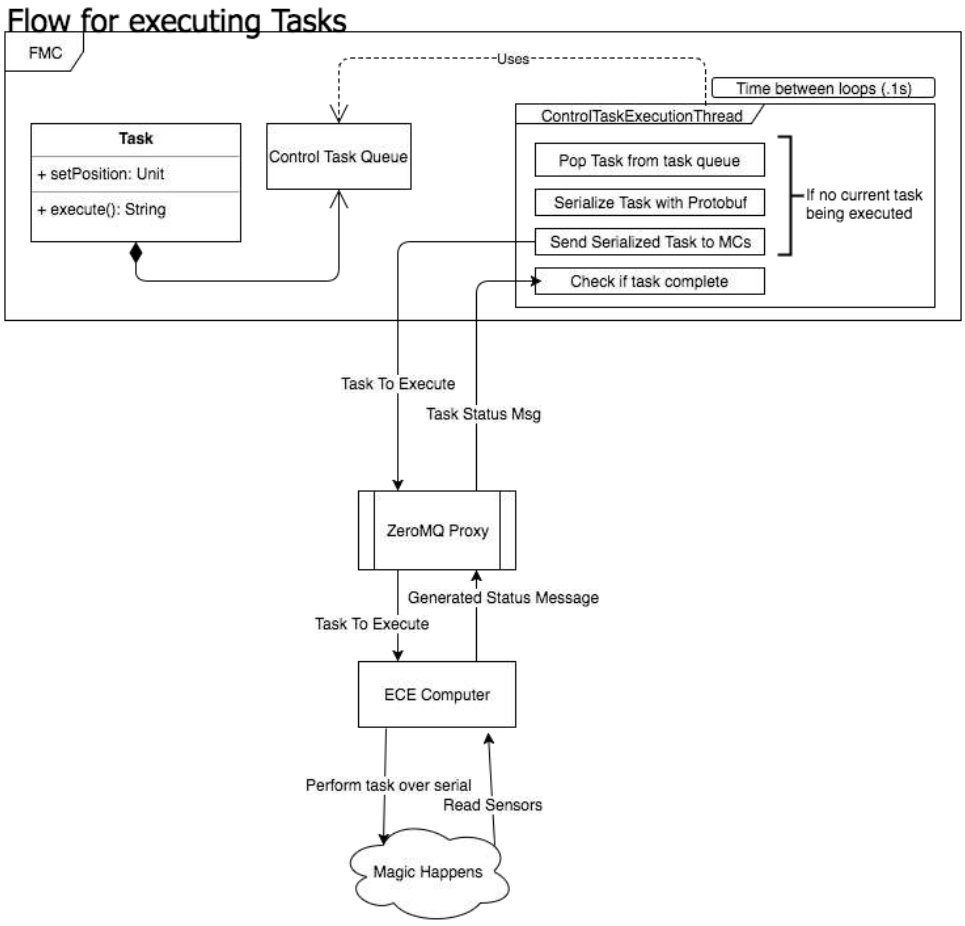


Figure 3E – Flow of execution for Tasks

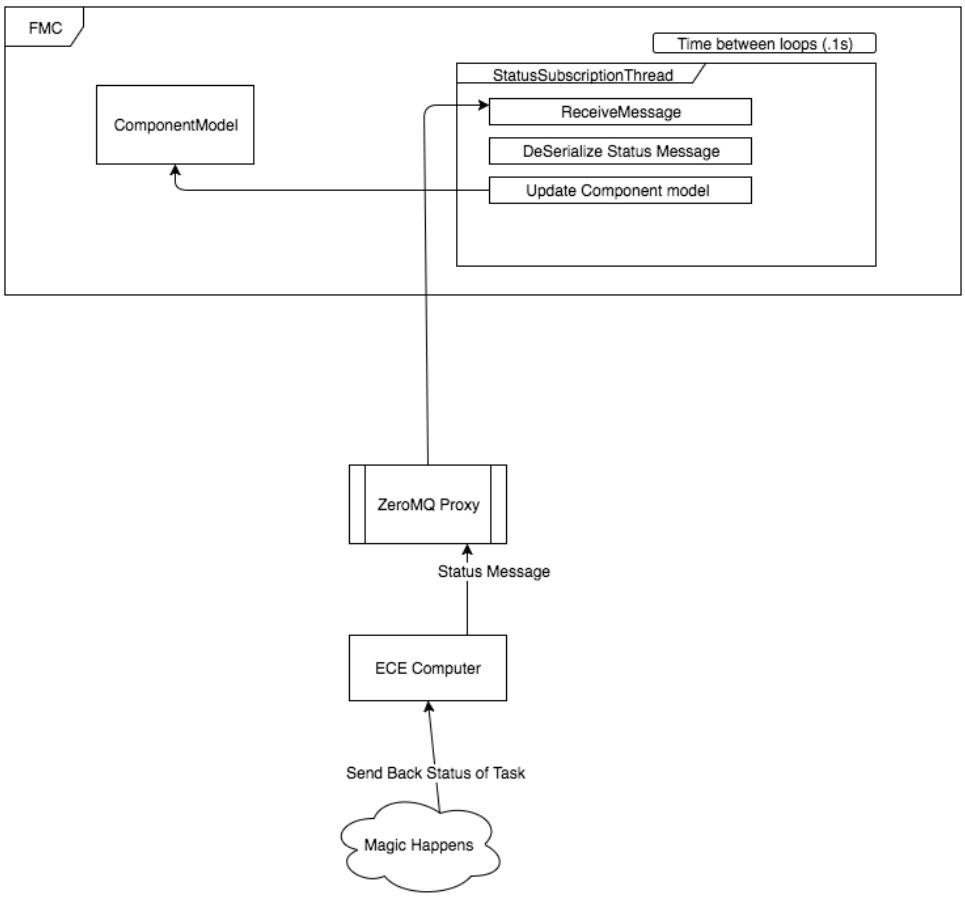


Figure 4E – Flow of capturing sensor readings

E 4.2 State Machine

A portion of the procedure stack deals with a finite state machine. From gate to gate, the aircraft goes through several different states on the way to its destination. These states help manage which routines are run and what conditions to check for during each phase of flight. Each state in the state machine has a unique check method that tells the state machine when to transition to the next state. For example, during the startup phase, the plane will transition to the taxi phase once taxi clearance has been acquired. This taxi clearance check is unique to the startup phase and cannot be triggered from any other state due to the transitions being one way.

The state machine also provides more context to the plane's current situation. This context can be used for activating audio cues, avoiding unnecessary checks, and other tasks that would benefit from the state. Avoiding unnecessary checks can save precious computing power for higher priority tasks. For instance, if a condition only needs to be checked during landing

phase, there is no point in checking for said condition during any other phase of flight.

From a modular standpoint, the state machine provides flexibility in replacing subroutines or correcting errors. If a specified action needs to be added during the initial approach, a developer can simply add a routine that is run during the initial approach without having to modify any existing code. If the aircraft keeps performing an action incorrectly during the takeoff routine, the action can be pinpointed easily by looking at any routine running during the takeoff.

E 4.2.1 Phases and their Transitions

The flight management computer contains twelve phases: startup, taxi, takeoff, initial climb, cruise, descent, initial approach, final approach, landing, rollout, and taxi parking. Each of these phases can only transition to either itself, or to the next phase of flight. There is no transition to a previous state. With each phase is a bundled check method that is constantly queried to determine when to transition to the next state. For example, during the startup phase, the machine checks for taxi clearance to transition to the taxi phase. In the taxi phase, the machine checks to see if the aircraft has reached the hold short before the takeoff runway to transition to the takeoff phase. The rest of the phases follow a similar pattern and can be observed in the StateMachine class inside the *acap.fmc.procedure.fsm* package.

E 4.2.2 Implementation

The current library used for maintaining the state machine is StatefulJ [3]. This library was chosen because it is simple to set up and allowed for easy transition set ups. The state machine is first defined with a set of phases with a corresponding name and Boolean indicating if the state is a final state. After defining the states, the state objects created are added to the state machine object instantiated. With states defined, the next step is to create transitions between each state. For all phases, a transition is invoked with a 'CHECK_PHASE_CHANGE' action. This action occurs every 100 milliseconds and runs a corresponding phase check to test if the conditions are right to transition into the next phase. The methods can be replaced with any method that returns a Boolean and allows for easy reprogramming if state transitions need to be tweaked in the future.

E5 ZMQ Setup

For passing messages between the flight management computer and any external component, we needed to first figure out what the most efficient method of communication would be between the two endpoints. We decide to use ZeroMQ as our messaging server for passing messages between the two endpoints. Zero MQ is a distributed messaging system that can connect code across any platform/language with the ability to carry message across inproc, IPC, TCP, TIPC, and multicast [4]. It also supports architecture patterns such as pub-sub, push-pull, and router-dealer. We chose ZMQ because the library is simple and has no added bloat.

E 5.1 Subscriber-Publisher Pattern (Sub-Pub)

The subscriber-publisher pattern addresses the multicast problem of group-messaging. A publisher sends out each message to "all of many", such that all subscribers receive the same message. The sub-pub pattern is aimed at scalability such that large volumes of messages sent out to recipients do not choke the publisher if network conditions are in our favor [4]. Sub-pub can handle a large amount of traffic since subscribers don't talk back to publishers (this talking back known as back chatter). When publishers don't need to worry about processing responses, more computing power can be used on sending out messages efficiently. Removing back-chatter has several downsides though [4]:

- A publisher cannot tell when a subscriber is connected
- Subscribers cannot control the publisher
- Publishers don't know if a subscriber has dropped

The biggest downside of these is the idea that the publisher will keep on sending messages without confirmation from the subscriber. Since each end will need to be sending/receiving messages, we would be able to deal with the problem of detecting whether a subscriber has dropped by checking for heartbeat messages. For now, we do not have heartbeat messages implemented, but they are planned to be added in the future.

For the subscribers/publishers, we also wanted to have the nodes connect to a central server so that the host does not have to know the addresses of all the endpoints that need to receive a message. Using a proxy, we can accomplish this behavior and end up with the following diagram found in Figure 5E.

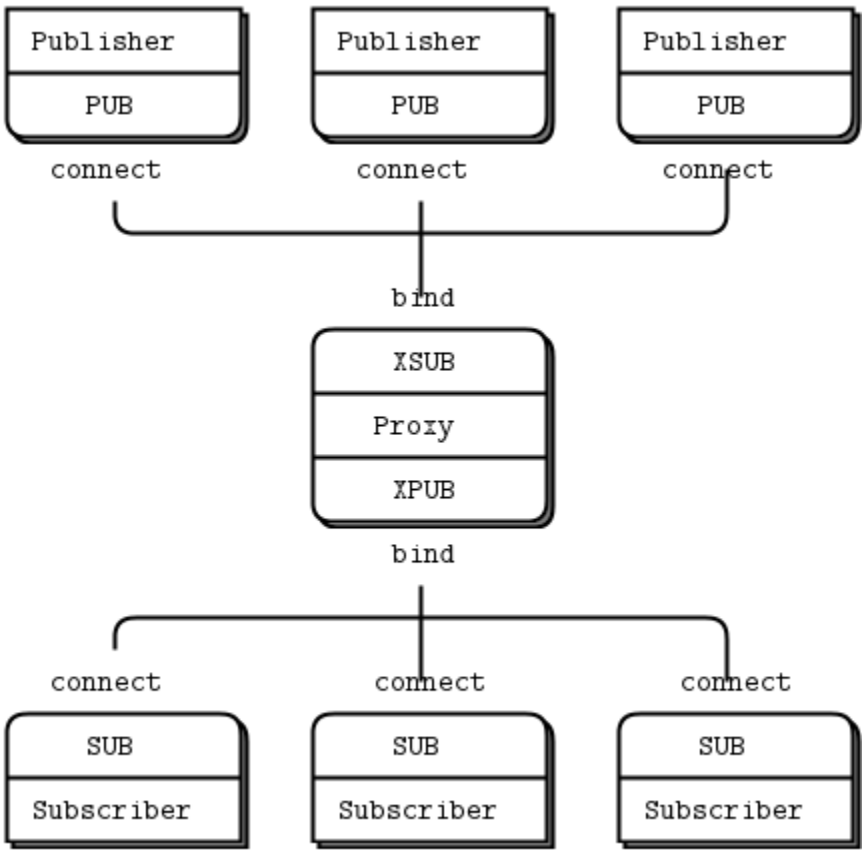


Figure 5E

In Figure 5E, the subscribers connect to a central proxy node and receive messages from the XPUB socket. This XPUB socket acts as a proxy publisher, delivering messages from publishers connected to the XSUB socket. Whenever a publisher sends a message to the XSUB socket on the proxy, the proxy relays the message out to the subscribers. The perks of having a proxy are simple: we can add as many publishers and subscribers without having to worry about connections to each one. Also, we can process the messages on the proxy if any messages needed to be tweaked before they were sent out to subscribers.

For our current setup, the ZMQ proxy server will be running on the flight management computer, but could be run anywhere (if the publisher/subscriber nodes can connect). In the future, there may be a reason to switch to a request-reply pattern, but for now we have not needed the capabilities.

E6 Hardware Interfacing

For interfacing with the hardware, all actions are executed over serialized messages. Using Protobuf, the messages can be constructed as normal objects and serialized with a single command. The flight engineer controls the hardware through the GUI, but the GUI utilizes the component controller class to translate the input into actions. For the classes that deal with hardware interfacing, we separated the role of sending controls and storing device data to allow either to be swapped out. The device data is stored in the devices class and can only set/get information about an affiliated hardware component.

E 6.1 Component Controller

The component controller class provides the interface for setting a controller's position. Each instance of the component controller class represents a single position change. The created instance is then serialized into a control message (see procedure stack) and sent to the electrical components translated into physical movements using the *serialize* method. Having a single instance of an object represent a single action allows for the component controller to be used for task execution. For handling the several types of devices that need to receive messages, the constructor of the component controller takes in an enumerator indicating the target component.

E 6.2 Devices

The devices class manages the current state of the physical controls. At the time of writing, the variables managed are the min position, max position, and current position. The interface provided by this class allows for the changing of a device's state. Since the states of all devices can be received from the sensors attached to the devices, the devices class is always instantiated with a clean slate upon startup of the flight management computer. The state is not stored and restored from each session due to the potential inaccuracy of sensor readings from an older set of data.

The states of the devices are maintained by a hashmap with the key being the device enumerator value found in the Protobuf message structure definition (Figure 2E). The hashmap maps each device enumerator to an array of Objects. An individual type is not defined as the array to allow the ability to store varying types that represent the state of a device. For instance,

if a device needed to maintain a numerical representation of a sensor reading along with a string value or a newly defined type, the varying types can be supported without hassle. Adding support for new data in the device state array is done by adding an instantiation of the hashmap, a constant that represents the position in the status array, and a public method that allows for easy access and manipulation of the new status.

```
for (component c : SerialMessage.ControlMessage.component.values()) {
  map.put(
      c, new Object[] {DEFAULT_MIN, DEFAULT_MAX, DEFAULT_POSITION, new FlightSimInterface(c)});
}
}
```

Figure 6E

The public methods for accessing this state data utilize a constant that indicates where in the state array the corresponding state data is found. The abstraction provided by the public method allows the data to be accessed by only knowing the name of the component.

```
public void setControllerMax(component c, double value) { map.get(c)[MAX] = value; }
```

Figure 7E - Example snippet where the controller's max value is accessed with the defined constant MAX.

The same principle of needing to know only the name of the component applies to the entire device class. With the names being based off the Protobuf device list, any added in the Protobuf file will not cause errors in the existing device class.

E7 Aerial Routing

To achieve autonomy, an aerial routing system (ARS) needed to be created to guide the plane from runway to runway. The role of the aerial routing system is to indicate when to climb and for how long, create points for traversing a traffic pattern, guide the plane on when to start turns for desired headings, and other tasks that will assist in guiding the aircraft. By generating a set of waypoints with real longitude/latitude coordinates, the problem is reduced to a "Follow the line" challenge.

To start, the ARS calculates the takeoff point. The takeoff is calculated from the waypoint that the aircraft lifts off the ground. From this waypoint, the ARS calculates the

waypoint for the aircraft to stop ascending. This waypoint is calculated using the target altitude, climb rate performance, and indicated air speed. This information can be used to construct a solvable triangle. Using these variables, the climb angle is calculated and the horizontal component of the distance is combined with the climb rate to return where the aircraft climb ends.

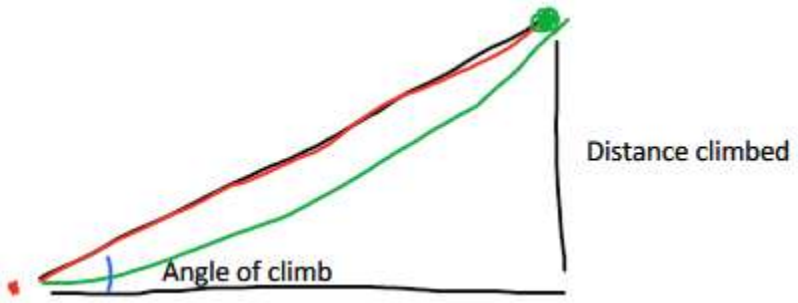


Figure 8E

In Figure 8E, the green dot indicates the return of the takeoff calculations. The green line represents what actual traversal would look like without instantaneous velocities. The horizontal black line shows the horizontal distance and the vertical shows the vertical distance. The blue line shows the angle of climb. After the takeoff point is found, the ARS continues with creating waypoints to navigate to a destination.

The next step for the ARS is to guide the aircraft to be at the proper heading towards the destination. By default, the ARS coordinates a turn at 25 degrees of bank. To perform an efficient turn, the ARS needs to tell the aircraft when to start. A turn is based around the waypoint with a vector going into the waypoint and a vector going out of the waypoint. The directions of the vectors represent the heading before and after the turn. Initially, a turning waypoint represents an instantaneous turn. With the start heading and end heading, the ARS will replace the instantaneous turning waypoint with two waypoints that represent the start and end of the turn.

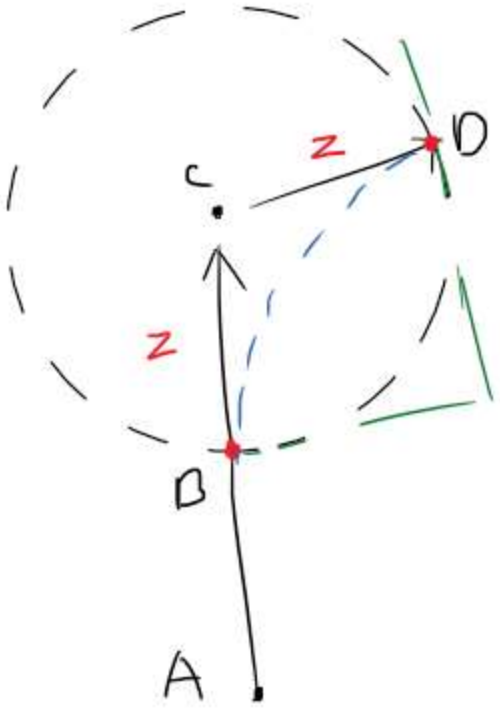


Figure 9E

To find the starting point, the ARS calculates the time of the turn using the air speed and bank angle. In our case, the bank angle would be 25 degrees (optimal bank angle for our aircraft). With the bank angle and airspeed, the ARS retrieves the time to execute the turn with Equation 1E. The time of the turn is calculated with the difference in the start/end headers (A and D) divided by the rate of the turn. This time is used with the airspeed to get the distance away from point C (instantaneous turn point) in Figure 8E which is value Z in the figure. This distance from the instantaneous turn is used for the start and end points of the turn.

$$\omega = \frac{1,091 \tan \theta}{V}$$

Equation 1E

After turning on course, waypoints are created at any location between the home and destination airports. Any waypoint between the home and destination airport would be a NAVAID entered by the user to follow a recommended flight path. The last set of waypoints generated from the ARS indicate the entry pattern at the destination airport.

When entering a pattern, the ARS can calculate the pattern waypoints for uncontrolled and controlled airports. There are two expected types of pattern entry in an uncontrolled airport: 45-degree entry onto midfield downwind and 225 "descending teardrop" onto the 45 degree entry joining midfield downwind. For a controlled airport, the entered leg would be indicated by the user. In the case of an uncontrolled airport, the ARS will anticipate which type of entry will be used and fly directly to the point of entry to avoid unnecessary fuel burn and maneuvers near the airport. Since the traffic pattern of an airport is standard, generating the waypoints is the same for any runway.

E8 Ground Routing

A core component of the ACAP project is autonomous ground navigation. Automated ground navigation is difficult in aviation due to obstacles (wingtips), taxi instructions, and strict taxiway markings. The first step to achieving automated ground navigation is building a set of waypoints along a taxiway for the aircraft to follow. The next step is creating a module to follow the line. This section focuses on the creation of the waypoint set.

E 8.1 Core Problem

The problem of navigating an airport is that the aircraft needs knowledge of where an indicated taxiway is. Another problem is that not every taxiway is going to be given a label. The solution that we came up with is to represent the airport as an undirected-weighted graph. The airport data used in the flight management computer represents airport taxiways as nodes with an assigned coordinate. The airport data does not connect any of these nodes, but labels what taxiway the node is a part of.

```
<TaxiwayPoint
    index="0"
    type="HOLD_SHORT"
    orientation="FORWARD"
    lat="42.2683028876781"
    lon="-71.8738053739071"/>
```

Figure 10E – Example Taxiway point

Using the data from the airports, we can create a graph using these taxiway nodes as the nodes for the graph. Due to overlapping nodes from intersecting taxiways, the name of the node cannot be set to the corresponding taxiway. Instead, the name of the node is based on the latitude and longitude. This ensures that a single node is created for intersecting taxiways which would otherwise cause the graph to be disconnected. Each node is assigned a set of properties that is used in the creation of a ground route: taxiway name, type, and holdshort. A holdshort is a marker that tells the aircraft to stop and wait for clearance before proceeding. The taxiway name attribute is used to create a route from a set of taxiway instructions. The type attribute helps navigate along unnamed taxiways, and the holdshort attribute is a Boolean which indicates whether the node contains a holdshort mark.

For parking spaces, the airport data contains the points for parking spaces located in the airport. These are added to the graph as and are connected to the rest of the nodes. These parking nodes help the aircraft find parking spaces at an airport.

Once all the nodes are obtained from the airport data, the next step in creating the graph is linking the nodes. The airport data groups taxiway points with paths that indicate the start and end of a series of nodes.

```
<TaxiwayPath
 type="PATH"
  start="7"
  end="8"
 width="21.34M"
weightLimit="500000"
drawSurface="FALSE"
 drawDetail="FALSE"
 surface="ASPHALT"
  name="0"
 centerLine="TRUE"
  />
```

Figure 11E

The start and end numbers are based off the indices of taxiway points. The taxiway path above defines a path consisting of taxiway points with an index of 7 and 8. By traversing these taxiway path objects, the nodes are connected by grabbing the nodes of the corresponding index and adding an edge between them. In this example, an edge is added between taxi nodes 7 and 8.

After connecting all the nodes, a completed graph is created. Figure 12E shows the graph generated from the KBOS airport beside the airport diagram. This completed graph can be used to find a set of waypoints to navigate, given taxi instructions.

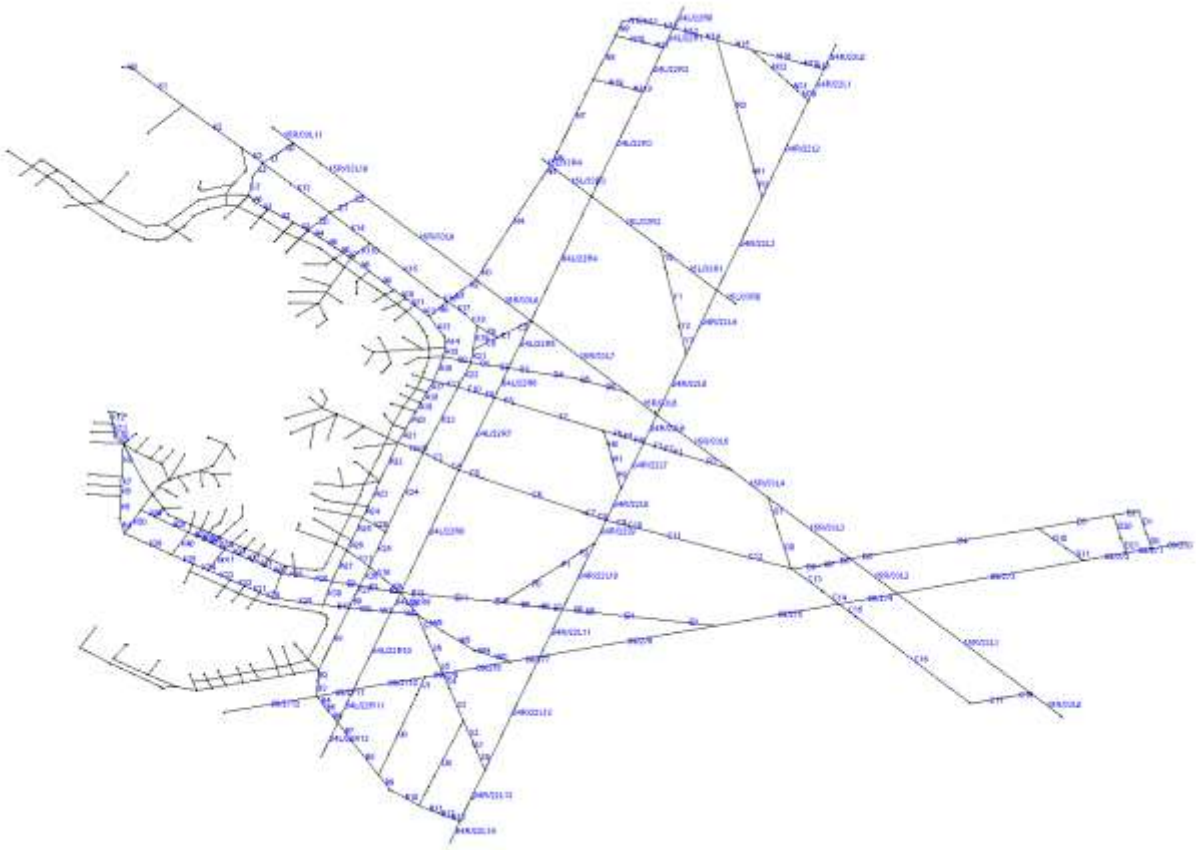


Figure 12E -Boston Logan Airport Graph

E 8.1.1 Traversing the Graph

With an airport deconstructed into a graph, navigating around the airport becomes a graph traversal problem. For finding the shortest path to a specific point at the airport, the shortest path can be computed using Dijkstra's algorithm. This works for un-towered airports, but for a towered airport, finding the shortest path to the target runway does not work. At a towered airport, a set of taxiway instructions need to be followed. For following taxiway instructions, a shortest path needs to be computed between each taxiway in the instruction set.

The core of the ground navigation is taking a taxi instruction set and translating it into a set of waypoints given an airport graph. The first step is splitting the instructions to get the individual taxiway identifiers found in the instruction set. Starting from a parking spot, the shortest path is found to the first taxiway in the instruction set. This shortest path is found by performing a breadth first search from the starting point for the node with the attribute matching

the next taxiway identifier. Given this node, the Dijkstra shortest path is computed between the two nodes. To ensure the path follows the instructions correctly, the Dijkstra algorithm is modified to ignore edges not on the current taxiway. For leaving a parking space, this would prevent the shortest path from including a detour to an unintended taxiway. The weight used in computing the shortest path is the distance between the nodes. This pattern is repeated between each pair of taxiways in the instruction until the end is reached. The result is a subset of the airport graph with edges that only contain the path indicated by the input string.

Figure 13E shows the taxiway instruction "K K3 E K F 15R/33L G D D1 09/27" being parsed at KBOS. This instruction is entered through the mode control panel discussed in Appendix F. The start of the path is at parking space A. The red dots indicate the end of a shortest path search. In this example, a total of ten searches are performed to find the correct path.

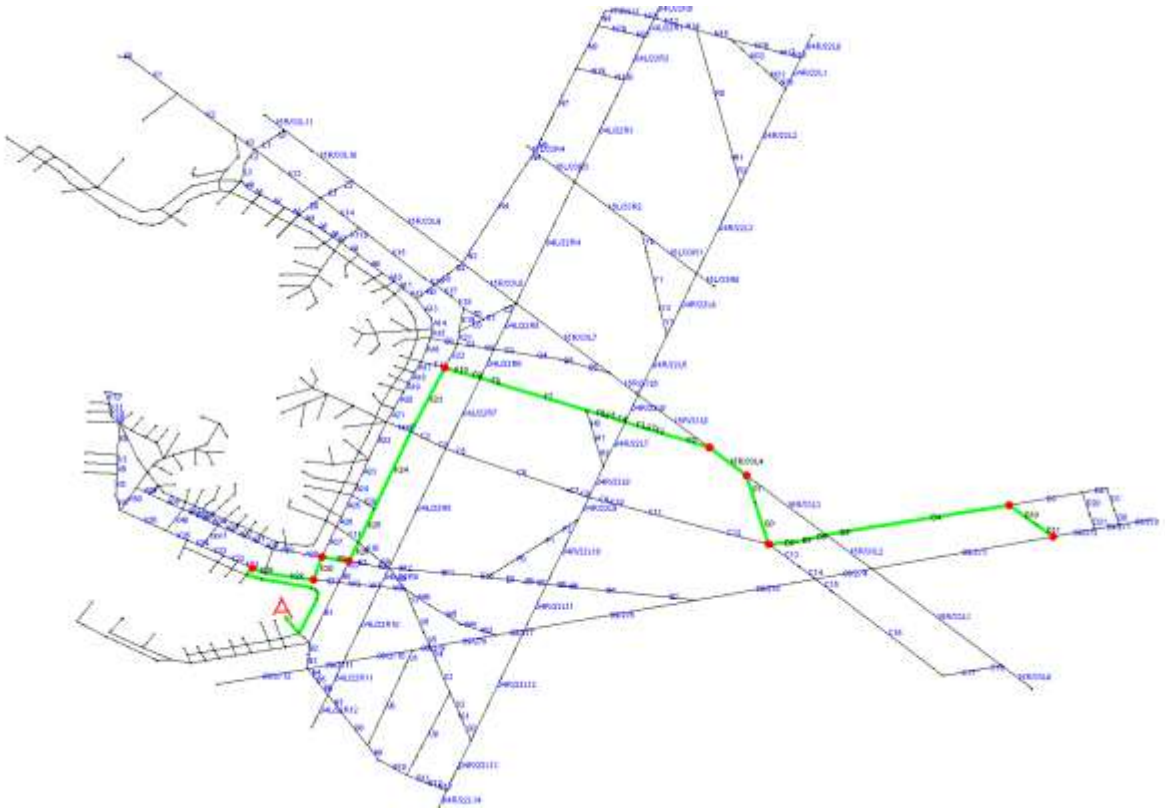


Figure 13E - Instructions "K K3 E K F 15R/33L G D D1 09/27"

E9 Weather Information

Weather data is critical to automated flight. Deciding which runway to pick in specific weather conditions allows for safe landings and takeoffs. The FAA uses the international standard METAR for communicating the current weather at a specific airport. A METAR message typically looks like:

KJAX 020256Z 02003KT 10SM TSRA OVC01OCB SCT100 BKN130 18/17 A2996

Details of a METAR message will not be discussed in this section. Instead, we present our methodology for collecting this data and interpreting it for the flight computer.

Writing code to interpret this data would be a waste of time given that several systems exist with the ability to represent a METAR message in an easily readable format. Therefore, we used an API provided by the Aviation Weather Center [5] to gather this data. The API provided by the Aviation Weather Center returns METAR data in an XML format and interprets the METAR data such that data elements returned include temperature (in Celsius) and the wind speed. In querying the API, the parameters include a station string (for specifying the station to retrieve weather from), a time range, lon-lat rectangle, radial distance, and flight path. These parameters allow for gathering weather based on several situations. Station string specifies the weather station to retrieve the METAR data from. The time range specifies the time interval of the requested METAR data. In our case, this would always be set to include one to two hours before the current hour. One to two hours is chosen for the case that the API reported no weather within the past hour. A lon-lat rectangle can be given to gather all METAR data within a specified area. The radial distance can be passed in to gather all METAR data within the radial distance of a lon-lat point and flight path allows for the retrieval of all METAR data on a specified flight path of waypoints.

These parameters allow us to gather data for a few specific situations the aircraft might be in. For example, if we needed to get the weather for the current area, but were not sure what airports were nearby, we could use the radial distance parameter to get the weather. When approaching the destination airport, we would only need to use the station string to get the weather needed to pick the runway to land on (if one is not assigned to the aircraft). Other

parameters have not needed to be used now, but might find a use case in the future.

The parameters are passed in via the URL of the query. For example,

https://www.aviationweather.gov/adds/dataserver_current/httpparam?dataSource=metars&requestType=retrieve&format=xml&radialDistance=20;-104.65,39.83&hoursBeforeNow=3

will get the METAR data within 20 nautical miles of longitude -104.65 latitude 39.83 up to 3 hours before now. The result we get back is in XML.

Figure 14E – Example Weather Data

JAXB (Java Architecture XML Binding) [6] allows for the creation of java classes based on XML schema. With this, we used the XML schema posted by Aviation Weather to generate the java classes used to represent the retrieved METAR data. This allows for easy access to specific METAR data without having to write helper classes to parse the XML data. After creating the java classes for storing the METAR data, the general interface for retrieving weather data was implemented.

For our weather interface, we implemented the ability to grab the nearest METAR data given an airport identifier, and a lon-lat. Finally, there is also a function to pick a runway based on the current wind conditions and the specified minimum runway required, max crosswind component, and max headwind component. Grabbing the nearest METAR data involves using the API discussed previously and taking advantage of the radial distance parameter. For picking a runway, the function takes in a METAR object and given airport as well as the requirements

specific to the aircraft for picking a runway (minimum runway needed, max headwind component, and max crosswind component). Using this information, the runway selection function first gathers the wind direction and speed from the METAR data given. Then there is a check to see if the winds are fast enough to consider crosswind/headwinds. If the winds are negligible, then the runway with the smallest heading differential is chosen based on the wind direction. When the winds are strong enough to cause considerable crosswinds/headwinds, the cross-wind component and headwind components are calculated. Once calculated, the function searches for the smallest heading difference in the wind direction as well as ensuring that the runway does not exceed the max head/crosswind components. The function then returns a runway object.

Works Cited

1. R. Bagby, "Helper Classes Are A Code Smell," 10 June 2012. [Online]. Available: http://www.robbagby.com/posts/helper-classes-are-a-code-smell/. [Accessed February 2017].

2. M. Ward, "Avoiding Utility Classes," 21 May 2012. [Online]. Available: https://github.com/marshallward/marshallward.org/blob/master/content/avoid_util_classes.rst. [Accessed February 2017].

3. StatefulJ, "StatefulJ Framework," 2016. [Online]. Available: https://github.com/statefulj/statefulj.

4. iMatix Corporation, "ZeroMQ Guide," 2014. [Online]. Available: http://zguide.zeromq.org. [Accessed August 2016].

5. Aviation Weather Center, "Aviation Weather Center," US Dept of Commerce, [Online]. Available: http://www.weather.gov/.

6. Oracle, "Java Architecture for XML Binding (JAXB)," [Online]. Available: http://www.oracle.com/technetwork/articles/javase/index-140168.html.

Appendix F
Software Frontend

Written By: Tyler Nickerson & Nicholas Cyganski

Table of Contents

F1 Design Methodology

During NASA's retrofit of the Space Shuttle fleet in the 1990's, the shuttles were equipped with "glass panels," meaning that the primary instrumentation was replaced with digitally generated imagery, projected on CRT screens. Glass panels made their first debut in the

Figure F1: Comparison of Traditional vs. Glass Cockpits

civilian aviation market with the Beechcraft 2000 Starship, and quickly spread to all corners of the industry. Today, many pilots work hard, and spend thousands transitioning from traditional cockpits, populated with mechanical “steam” gauges, to modern glass cockpits with minimal buttons and switches, which have been replaced with widescreen LCD screens. A Beechcraft King Air 200 cockpit is shown below, before and after being converted to a glass panel cockpit. The difference in equipment and how information is displayed to the flight crew dictates a significant change in the flight crews thinking that is required to process information and react quickly.

Glass panel architecture offers the avionics designers much more freedom when trying to design intuitive cockpit displays, but the designs are not always well received by the pilots who actually have to use them. Rather than subscribe to the mentality of any one aircraft or avionics designers, this project attempts to blend together several different cockpit styles to create an interface between man and machine that satisfies the flight crew, while providing the debugging tools and new features required by this prototype autonomous vehicle. In many ways, the design of the flight engineer’s station, in the right seat of the aircraft, resembles that of modern drone ground control stations, shown below. For obvious reasons, a ground station will likely become the eventual evolution of the flight engineer’s equipment. However, currently a flight engineer must be present in the cockpit during all navigation.

Figure F2: Remotely Operated Aircraft Instrumentation

F2 Safety Considerations

Beyond displaying the information in a format that can be easily understood by the flight crew, there are a limited number of features on the touchscreen displays that must be accessible and perform consistently for the flight crew whenever they need. The most predominantly featured of these is the autopilot disconnect button, and associated autopilot status indicators, which are all visible to the flight crew independent of which mode or data is being displayed on the screen at any given time. There are a myriad of ways by which to disconnect various autoflight features, the most commonly used of which will always be the disconnect switch located on the pilot's side yoke. Should the electrical connection to any of the hardware become inoperative in flight, the crew should be able to disable all autoflight features from the touch screen interface, meaning that there is no additional hardware between the human interface and the hardware to be disconnected. The autoflight mode indicators also provide the flight crews with an interface much like the Mode Control Panels (MCP) of larger aircraft, with persistent logic and numerical representations of what the aircraft is trying to achieve. The combination of these two displays resides at the bottom of all multi-function displays, as seen below.

Additional interlocks are also provided by the GUI, which must be in agreement with other hardware settings and mechanical interlocks to allow for the flight computer to take control of more dangerous aircraft functions, such as moving the main actuators, and starting the engine.

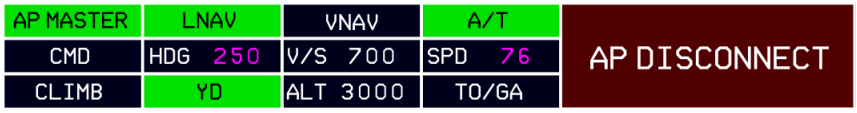


Figure F3: GUI Lower Third - Autoflight Indicators

F3 Launcher

Given that there are several programs that must be running and configured with each other to bring the aircraft into an operable condition, it would add several minutes of pre-flight preparation and monotonous procedures if the flight crew were to conduct this setup at the beginning of every flight. To avoid these undesirable effects, a launcher will be included, allowing the flight crew to select from several modes, and conduct the setup of all programs automatically. These modes will include a simple choice between simulated flight for debugging purposes, as well as actual flight. In addition, the launcher will include a series of checkboxes that will retain their settings from the previous flight, allowing for some of the more dangerous systems to be disabled, such as the main actuators, or the engine ignition circuitry.

F4 Graphical User Interface

The original simulator on which the flight management software was based used the Processing library for Java in conjunction with ControlP5 to render the 3D world, as well as the controls necessary for modifying the world (toggles, sliders, dropdowns, etc.). Although Processing remained a suitable framework for displaying the environment outside the cockpit, the ControlP5 component library resting on top of it proved cumbersome as the software requirements began to expand. Most notably, ControlP5 positions components using absolute coordinates, meaning specific X-Y coordinates need to be specified for each component drawn. Not only does this positioning system prevent the software layout from easily adapting to multiple screen resolutions, but container elements such as panels and grids become much more difficult to implement. Likewise, the initialization code for the ControlP5 components in the original software predominately consisted of duplicate code blocks with minor modifications in each. Our ultimate goal in designing the new GUI was to maximize code and component reuse without the need for duplicating chunks of code.

Initially, we entertained the notion of replacing the existing system with a strictly web-based

architecture, replacing the Processing framework with a WebGL alternative. This would have allowed us to build reusable components in frameworks such as Facebook's React while maintaining a universal stylesheet for every component on the page. However, due to time limitations and the amount of work required to fully replace Processing, this idea was dismissed. As a result, we attempted to instead integrate the principles behind such a web-based system into the existing Java model. We identified these principles as follows:

1. The presence of a component library in which common system controls such as buttons, toggles, and dropdowns may be reused with minimal effort while maintaining a consistent styling (padding, margins, color, etc.)

2. The ability to create responsive layouts in which components automatically wrap, expand, or scroll depending on the dimensions of the window on which they are displayed

3. The existence of a universal color palette which will maintain a consistent look and feel throughout the software

Due to the original system's lack of containers and layouts, our first task was to replace ControlP5 as the primary component library. Our first choice was JavaFX, due to its support for CSS as a styling language and its mission to act as a framework for developing rich internet applications runnable on any platform [1]. To implement JavaFX, the existing Processing GLWindow object would have needed to instead be loaded into a canvas object and run inside a JavaFX application. However, we quickly discovered that our only option for doing so was via JOGL, a Java binding for the OpenGL API [2]. JOGL provides a NewtCanvasAWT wrapper object, which loads OpenGL applets as canvas objects in the Abstract Window Toolkit (AWT) for Java, a framework dating back to the launch of Java in 1995. While it is possible to combine JavaFX and AWT components in the same application, the official documentation suggests that doing so would often cause a significant decrease in application performance, thus partially defeating the benefits of using JavaFX [3]. As a result, Java Swing was adopted as the primary GUI toolkit in place of JavaFX, considering it is the original successor to AWT but continues to support AWT components.
While seemingly a straightforward task, wrapping the existing OpenGL applet in a canvas object proved particularly difficult. Internally, Processing runs a series of setup function calls within a primary setup() method once the applet's initialization stack is called. Therefore,

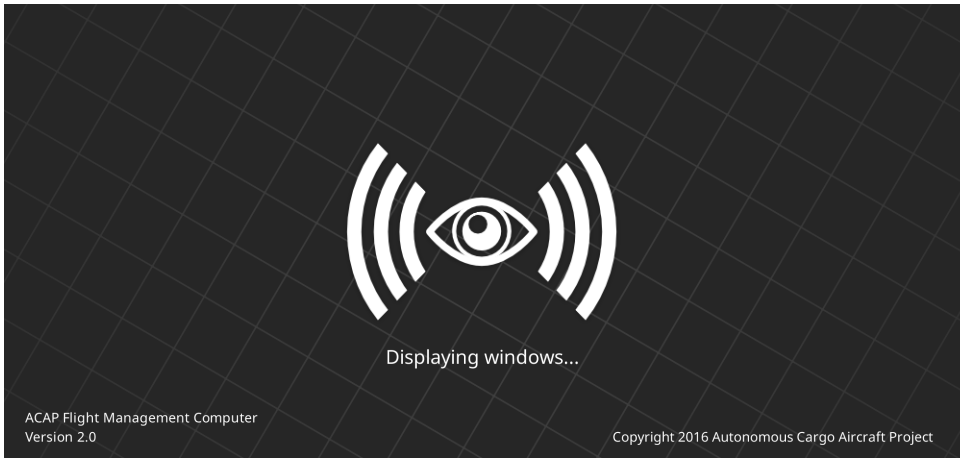


Figure F4: ACAP Launcher Splash Screen

simply wrapping the applet in a canvas object and running its underlying thread caused the application to crash almost immediately, as this approach jumps over all setup calls. To add to this complexity, Processing's encapsulation structure renders it impossible to simply call these setup methods from the main applet class. Therefore, several of Processing's internal, decompiled methods were copied into the main applet's class. While this muddled the applet's code by combining lower-level initialization steps with higher-level function calls, it was necessarily in keeping Processing as the main render engine of the flight computer. As the GUI became more complex and multiple Processing applets were added to the software, these decompiled methods were eventually extracted to an abstract class to encourage code reuse (see "Engine & Instrument Displays").

By incorporating Swing as an additional library required at runtime, the overall launch time of the application was greatly delayed. To compensate for this, a splash screen was developed to inform the user of the application's initialization progress. All initialization code is run on an independent thread separate from the splash screen interface to ensure there is a minimal delay in displaying the splash screen. The separate worker thread then reports progress back to the splash screen, causing the screen's text to update.

Once the interface structure was fully refactored with the new AWT canvas, work began to migrate all existing ControlP5 components to Swing components. This task was relatively simple, as the main Processing applet utilized a singleton pattern and was controlled by model classes, allowing the entirety of the 3D world to be modified without touching the Processing applet directly. The first Swing components that were added to the new interface were two combo-boxes that modified the airport view and the aircraft view, respectively, as this was the

only existing functionality in the original applet. However, it was quickly realized that the camera would eventually follow the aircraft object as it navigated throughout the 3D world, so functionality to change the aircraft view was dropped.

F5 Primary Flight & Navigation Displays

Taking inspiration from general cockpit design, the left half of the widescreen touch interface is reserved for the Primary Flight Display (PFD) and Navigational Display (ND), allowing the flight crew to observe how the aircraft's perceived situation and intentions react to changes made on the right half of the screen, which will contain all other settings.
This first mockup of the PFD, created by the flight crew, is predominantly Boeing inspired, with a few Airbus motifs in the speed and altitude tapes, and the wider vertical speed indicator area. The most significant differences between existing cockpit displays and this PFD is the inclusion of radio frequencies, and a more prominent angle of attack meter. Active communications and navigation radio frequencies are shown on the PFD to indicate what the flight computer believes it should be tuned to in real time, if it had authority over radio tuning, which is not the case in this prototype aircraft. Communications frequencies change at prescribed times by air traffic control, or at the discretion of the pilot throughout the flight, so it is imperative that an autonomous aircraft can make those same decisions while traversing airspace. Since smaller aircraft can be expected to traverse a much larger range of angle of attack during the typical flight, as compared to the rather docile changes associated with larger commercial aircraft, the angle of attack meter takes center stage in this small prototype aircraft.
In contrast to the PFD, the ND in this prototype aircraft is quite a departure from existing equipment. The ND is a depiction of the 3D world surrounding the aircraft, including airport grounds, taxiways, runways, route, and in-flight transition points. The camera has three modes; aircraft, airport, and free, which all allow zooming, and rotation or panning. In aircraft mode, the camera is bound to the aircraft, in the center of the screen, depicted in 3D. The camera will follow the aircraft as it flies, or taxis on the ground. Airport mode positions the camera at the origin of the selected airport, from which it can be zoomed, rotated, or panned about to plan a taxiway route, or visualize an approach. Lastly, free mode can start at the origin of the current flight plan, and can be moved about with complete freedom to visualize

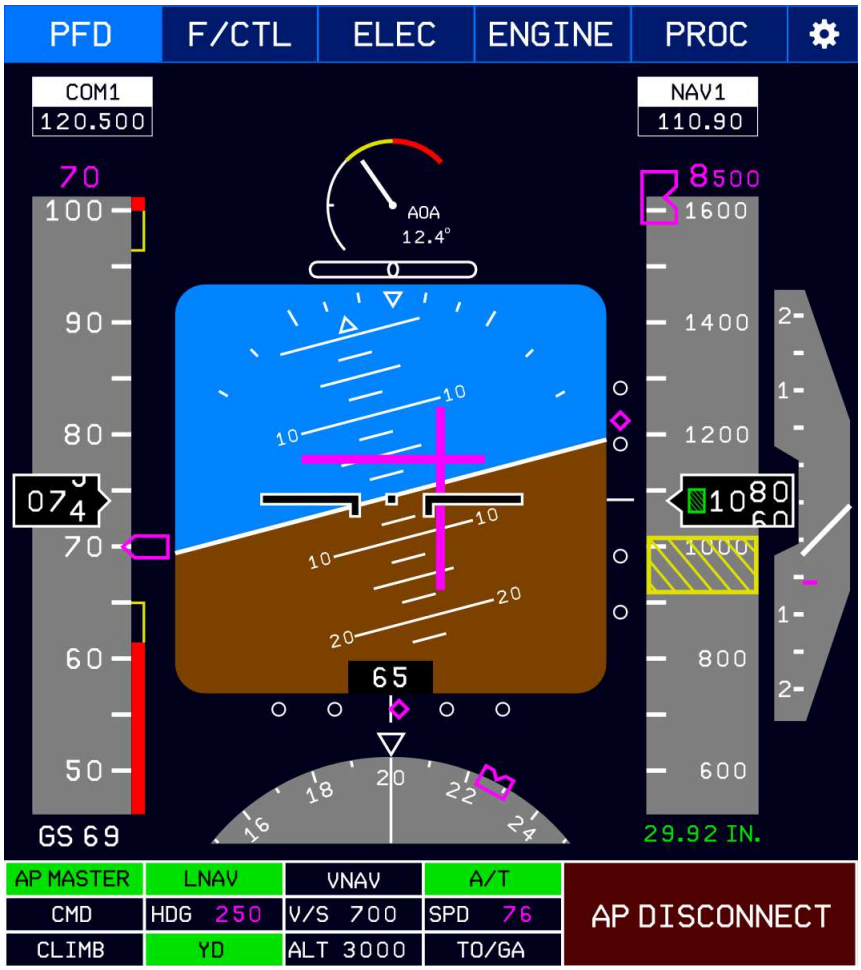


Figure F5: Primary Flight Display

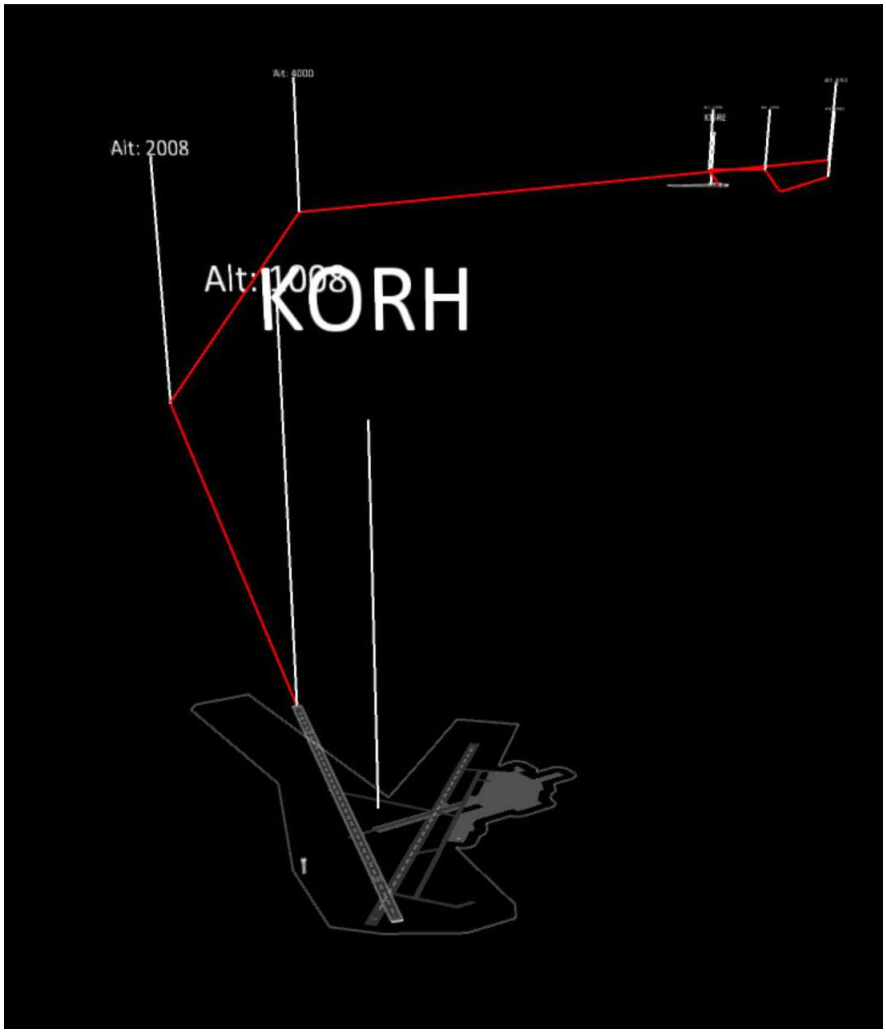


Figure F6: Navigation Display Aerial Routing Example

any of the above, but will likely be of particular use in examining the planned route ahead of the aircraft. Airport and various navigational aids can be shown in detail, only when zoomed out, or not at all. Points used in the active flight plan will always be shown. The planned path of the aircraft, transition points, and a history path are shown in the 3D world, and can be observed while adjusting performance parameters on the right half of the screen. Below is an example flight plan departing Worcester Regional Airport in Massachusetts, and landing at the uncontrolled Orange Municipal airport, also in Massachusetts.

Aircraft ground navigation is also depicted, including runway markings, taxiway markings and signage, vehicle pathways, aprons, and parking stands. Other aircraft are also shown in this image of Logan International Airport in Boston Massachusetts, occupying all the parking spaces, forcing the aircraft to be assigned a parking location.

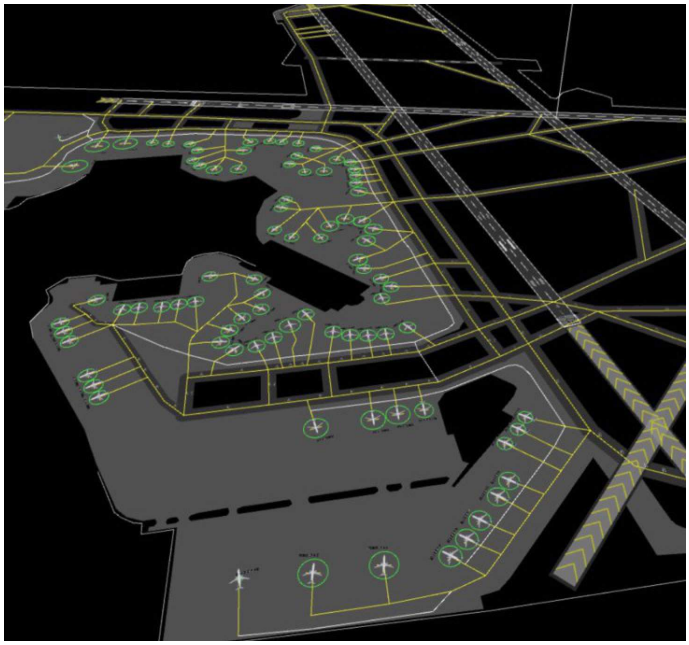


Figure F7: Navigation Display Ground Example

F6 Engine & Instrument Displays

The engine display is more heavily inspired by traditional cockpits to make for easy readability, and to more intuitively depict caution and danger ranges of the various systems, shown on radial gauges in red and yellow. There are several other features related to the powerplant shown on this display, such as the ignition and magneto firing, mixtures, carburetor heat, fuel flow, and aircraft hobbs and tach meters.

The tachometer takes front and center on this display, showing the user the current engine output shaft RPM, and the trendline, radially. The throttle setting, regardless of actual output RPM is shown as a percentage of throttle position, under the radial gauge. Likewise, the exact position of the mixture handle is shown as a red bar, ranging from “lean” at 0%, to “rich” at 100%. Four mixture set points are also shown along the bar, labeled with the first letter of the mixture setting; Stop, Taxi, Economy, and Performance.

Carburetor heat is a toggleable button, allowing the flight crew to actuate the carburetor heat lever, and a display of the observed drop in RPM when the leaver is deployed. Using a history of similar type aircraft, a percentage of carburetor icing will be calculated, and displayed, indicating the amount of ice present in the carburetor intake, from none, to a fully choked engine at sea level.

The fuel system is depicted with both tanks, the selector valve, and both fuel pumps. All fuel

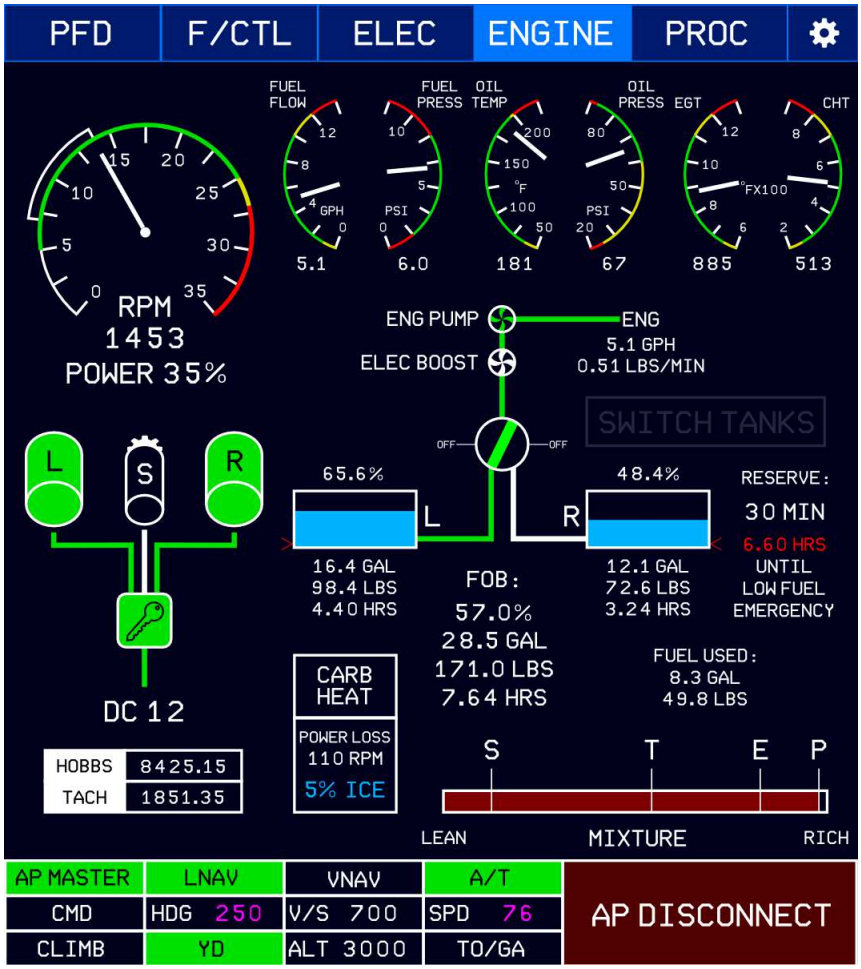


Figure F8: Engine Instruments Display

statistics are shown numerically, such as: endurance, fuel flow rate, weight, quantity, and reserve. The fuel quantity in each tank is also shown graphically. Since switching fuel tank feeding will have to be performed manually in this prototype if the flight exceeds a duration which one tank can support, a large “switch tanks” indicator will illuminate to indicate that it is the optimal time to switch.

The electrical interface takes its inspiration from Embraer and Airbus interfaces, with a focus on the battery charging circuit. Based on the sensed configuration of all autoflight electronics in the aircraft, active lines are depicted in green, along with the systems that they supply power to. Inactive lines and equipment are shown in white. Fuses and their values are shown solely for reference. Interactive elements on the electronics page include: the fuel pump, all lighting, and the bus tie switch, which connects the batteries at the beginning of the flight. Four physical switches in the aircraft are shown on the display, but must be physically thrown in the aircraft to alter their position. These include; the alternator master, battery master, two

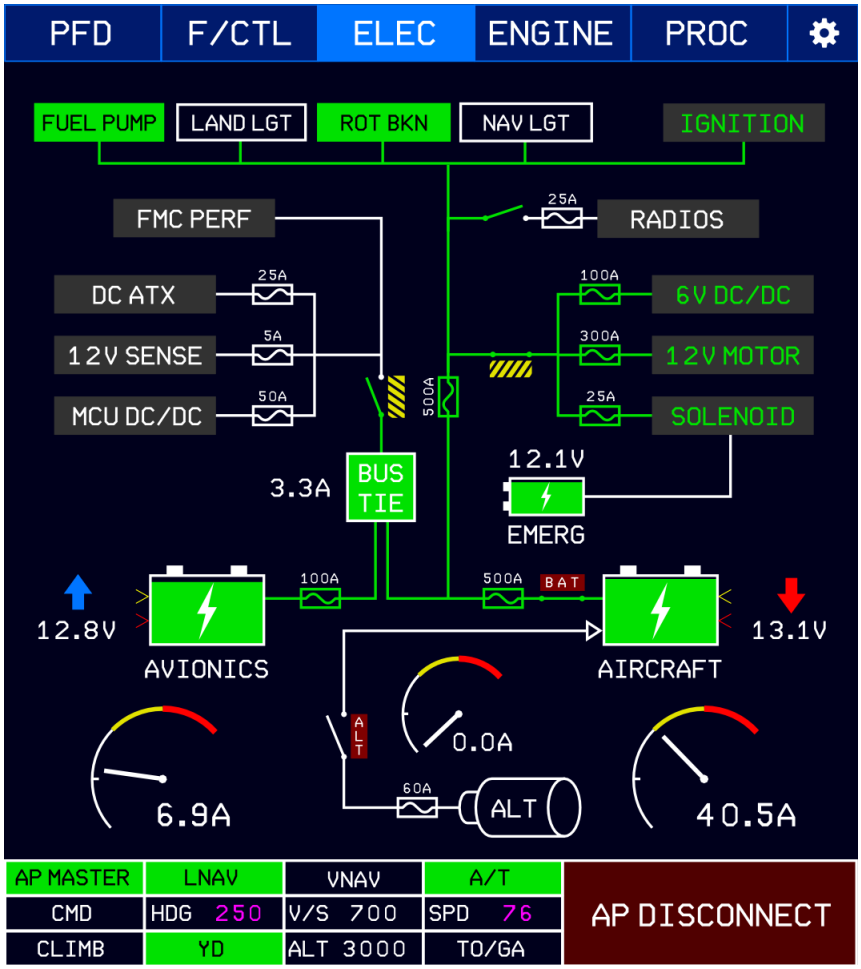


Figure F9: Electrics Status Display

emergency disconnects, and the avionics master switch. Digital ammeters show the current being drawn from each battery on a radial gauge, and the current being supplied by the alternator to the charging circuit. Caution and danger ranges for each of these meters show currents which would either overload the circuitry, or no cause a loss of battery voltage overtime. Battery voltages are shown for each battery with trend arrows to show their real time charging statistics. Critical battery voltages where the flight computer would no longer be able to properly convert voltages for the motherboard, and where the lead-acid batteries would sustain damage, are shown with small chevrons next to the graphical indication of battery voltage level.

The flight controls interface provides the flight crew with a graphical representation of every moving part of the aircraft, both their actual positions, and what is being commanded by the autoflight system. A forward profile outline of the aircraft is dotted with control surface position indicators for rudder, ailerons, stabilator, brakes, and trims. Green indicators for

primary controls, and blue for trims, show the current sensed position of the controls, while white diamonds show the commanded positions from the autoflight systems. Some flight controls have limitations based on the indicated airspeed of the aircraft. If the aircraft is above the specified maneuvering speed in the pilot's operating handbook, then the unacceptable part of the surfaces ranges will be greyed out, and movement of the surfaces into that range will be disallowed. The wheels of the aircraft are equipped with squat switches, which trigger whenever the landing gear suspension is not fully extended. A wheel in contact with the ground is indicated in green. The flap position indicator functions in a similar way, indicating actual position with a depiction of the flap and airfoil, and commanded position with a white diamond. Flap positions outside of the specified flap operating range are shown in red. Each actuator has an associated interactive button on the bottom, right-hand side of the screen. White indicators show inactive systems, which can be activated by the user. A yellow, "armed" state indicates that the actuator is capable of movement, while green indicates that the actuator is currently in transit. Opposite the actuator indicators are the status indicators for each sensor system onboard the aircraft, including: pitot-static, GPS, gyro/IRS, LIDAR, camera, outside and cabin air temperatures.

To print this level of sophisticated display graphics inside a Swing application, we departed from the traditional methods of utilizing 2D graphics paint methods and chose to base each display off scalable vector graphic (SVG) images. Originally, the display system was to be built using strictly SVG images, employing the Apache Batik SVG library to modify the underlying document object model (DOM) structure every time a component was updated. However, this approach quickly broke down, as the varying graphics programs used in creating the SVG files led to ambiguity in their DOM structures. As a result, Batik often had difficulty interpreting or displaying the files, leading to scaling issues and untraceable error messages. To avoid further complications, we began to entertain the notion of converting the DOM hierarchy into a Java 2D graphics class. We began by testing the primary flight display, using the open-source Flamingo SVG Transcoder to generate Java code from the SVG image. Yet the resulting class was an estimated 15,000 lines long, creating serious lag in the editors used by the computer science team. Additionally, the generated code, while successfully being able to display the image, was borderline incoherent and extremely difficult to navigate. Finally, we decided that defaulting to SVG-based Processing applets would be the most time-efficient and effective approach to creating the desired panels. The previously-mentioned

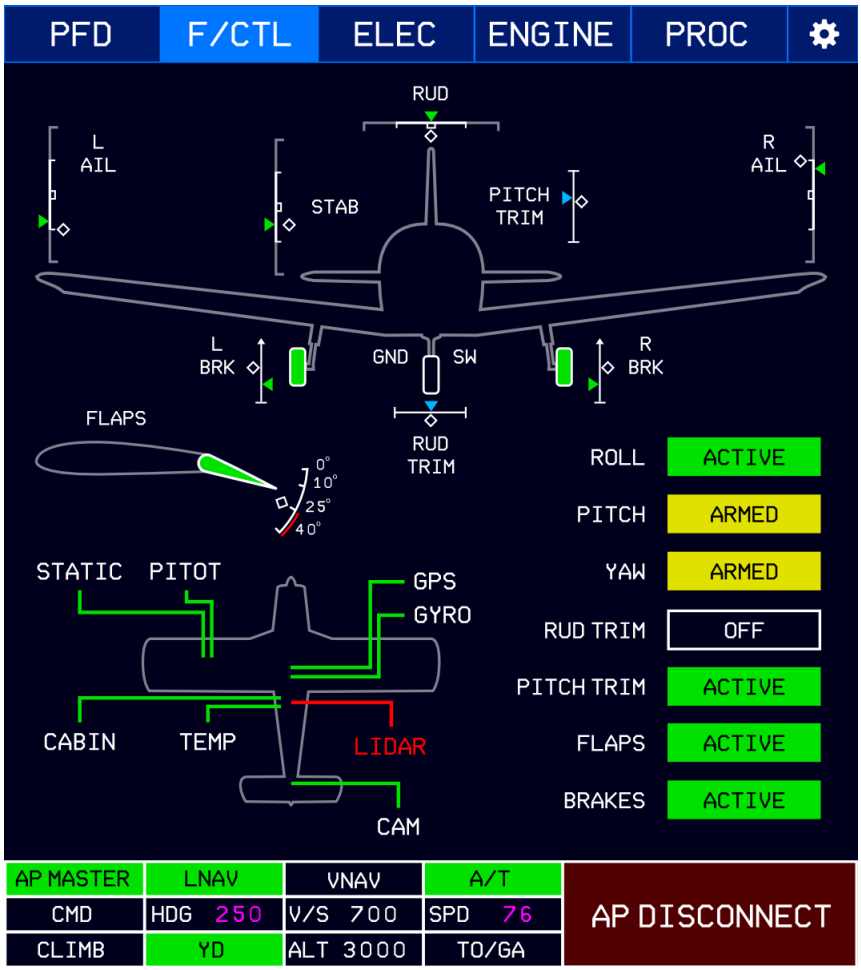


Figure F10: Flight Controls Status Display

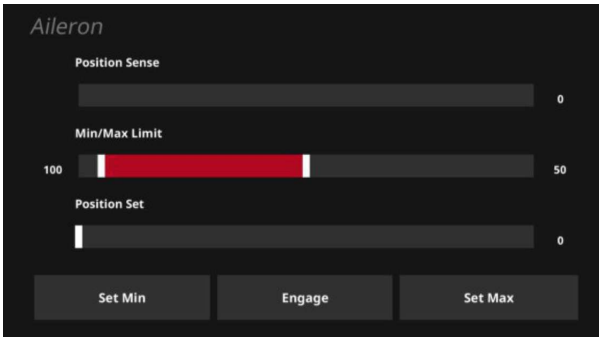


Figure F11: Actuator Debugging Display

spaghetti code that was extracted from Processing's decompiled applet class was moved to an abstract class. The new class allowed for the automatic generation of AWT canvas and panel objects, and each instrument display was given the ability to extend it. Each display panel now consists of a barebones vector image, rounded out by on-screen graphics generated in Processing and modified by global variables. Static components part of the original screens were extracted and integrated in Swing, most notably the top tab bar and the bottom panel.
Apart from the flight displays, a debugging interface was a high priority item since the beginning of the project, since the electrical and mechanical design teams would need a means by which to test their components with realistic commands from software as early as possible. In addition to all the toggle switch emulators, each control surface actuator, and servo motor requires three graphical interface elements to control the position of each; the sensed position of the control, the desired position it should be set to, and the minimum and maximum limits of the control, which are communicated to the hardware motor controllers, and interpreted as hard limits, to prevent any damage to the aircraft. These limits will be recalled from the previous runtime of the software, but should be recalibrated at each start of the aircraft. Buttons have been provided in the debugging interface to set the upper and lower limits, as well as engaging the system itself through the hardware engage pins. To provide the most intuitive interface for the user, the position sense and set sliders are scaled from the bottom limit to the top limit, such that their movement will always have the maximum resolution for the user over the entire range. Toggle switches will illuminate when they are active, and remain dark when not. Numerical positions of each slider are also included to the side, simply for reference by the flight crew, and for the hardware teams to record values without interrogating the program variables.
The debugging interface consists of the following tabs:

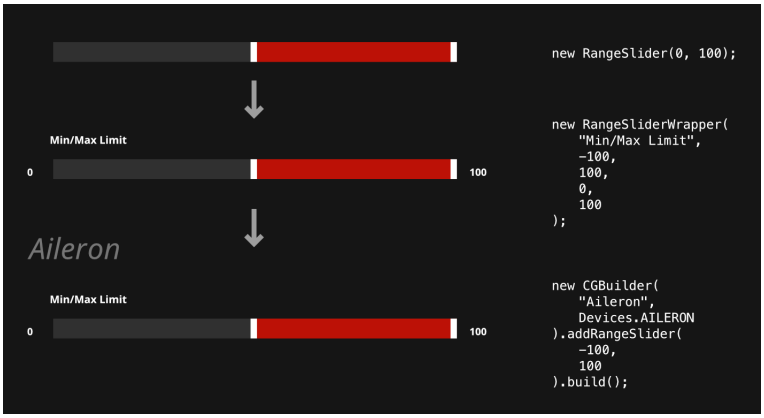


Figure F12: Control Group Builder Flow Diagram

1. **Global Controls:** contains position setters for the global velocity and smoothness across all motors, as well as a toggle switch to enable manually input via a joystick.
2. **Flight Controls:** contains sense sliders, position setters, min/max setters, and engage toggles for the aileron, stabilator, rudder, flap, pitch trim, and rudder trim.
3. **Power:** contains sense sliders, position setters, min/max setters, and engage toggles for the throttle, mixture, carb heat, and brakes, as well toggles for the left and right magnetos and starter.
4. **Program:** contains program-specific functionality, such as changing the airport view of the 3D world.

Within each tab, every device has its own group of controls, presented as a scrollable list (see Figure F11). Each control group is constructed using a Builder pattern, employing a CGBuilder (Control Group Builder) class that accepts a device constant and hooks each subsequent control added to hook into the corresponding variables powering that device. An example of how each component is initialized, wrapped, and packaged into the builder class can be seen in Figure F12.

F7 Flight Management

The higher-level autopilot controls are situated on a separate interface on the right half of the screen where data can be entered by keyboard, using a scratchpad. Many philosophies, including the scratchpad itself, are borrowed from existing commercial level Flight Management Systems (FMS). In this case, the FMS and Mode Control Panel (MCP) appear

Figure F13: Autoflight Mode Control Panel

combined, co-locating the real-time autopilot settings from the MCP, and the performance settings from the FMS. The most prominent feature of this screen are the autopilot master switches, for general Command (CMD) and Command Wheel Steering (CWS) which is a Boeing inspired mode to emulate the fly-by-wire (FBW) features of Airbus aircraft. Since it plays such a crucial role, by selecting the current control model being used, the current phase of flight is centered on the screen. The middle band of the screen is occupied by all the real-time autopilot controls that one would expect on an MCP. The bottom of the screen are several text boxes that would be expected in a traditional FMC, such as origin, destination, and route, but with some additions specific to this prototype. Taxi-out and Taxi-in boxes await either a calculated best route of taxi at an uncontrolled airport, or a prescribed one from a towered airport. Routes must begin from either the nearest taxiway or apron pathway to where the aircraft is currently parked, or from the runway being vacated. Paths are expected to terminate at a runway, or at a parking spot. Should any taxiway route terminate early, the aircraft will navigate to the end of the described path and hold.

Data is entered into the various number and text boxes by typing in the scratchpad. All formats of data entry are permitted in the scratchpad, as it can also be used as a memory aid. New autopilot commands, such as altitudes and headings, are entered in their corresponding boxes, shown in the above interface. If the format of the entry is not recognized, the box will flash red, and the entry will remain in the scratchpad. Long entries, such as routes, can be recalled to the scratchpad for editing by touching the corresponding box with an empty

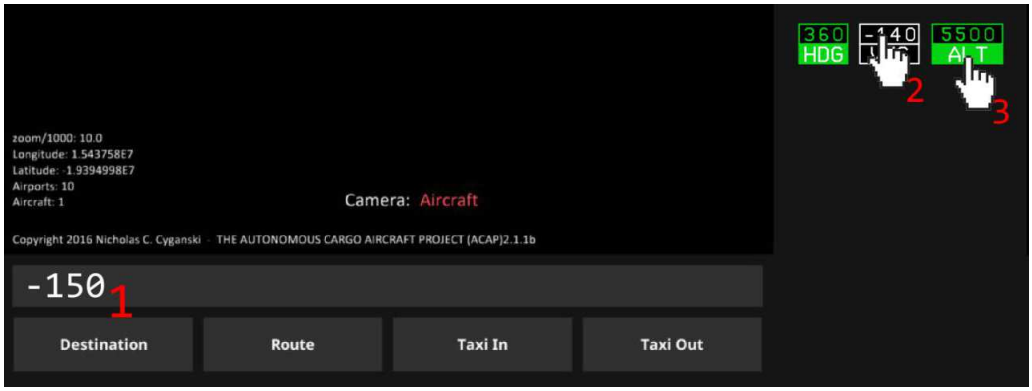


Figure F14: Data Entry with the Scratch Pad

scratchpad. An example data entry procedure, and activation of another autopilot feature is depicted below by the numbered cursors, one through three.

F8 Procedure Stack

The procedure stack, or task queue, provides insight into all tasks the aircraft must complete along its journey at a scope with enough detail to see individual checklist items, but not the finer, real time control adjustments, or settings, required to achieve that procedural item. For testing, and eventual autonomous manipulation of the aircraft, tasks can be added by the user and rearranged in the stack. Items that are currently in progress appear at the top of the stack, and a green progress bar appears in the background. Tasks which are already completed are deleted from the stack, but a log of all items is recorded for debriefing later. If a task fails to complete, such as being unable to move the carburetor heat servo, or an unsatisfactory sensor response to a changed state, the bar turns red, and a dialog appears, alerting the user to the problem. From there, the flight crew has the option to continue task execution of the procedure stack, or halt and fix the problem. Just as in the physical checklists in the cockpit, the procedure stack is organized into groups, such as initialization, or engine start, as seen in this short example queue below.

F9 Design Inspiration

Since this is the first avionics system to combine an experimental system debugging interface and traditional multi-function flight display onto one screen, it allowed the team to pick and choose concepts from existing aircraft and combine them into a new interface. Three aircraft

Figure F15: Procedure Stack Example

companies' products were considered as examples in the design of this new interface, based on their wide proliferation in the commercial aviation industry. Selected examples from these manufacturers are shown below. When compared to the final interface above, it should be evident which elements were taken from.

It is also worth noting that the inspiration for several of the engine instrument displays did not come from a specific avionics company's design methodology, but the industry trend of depicting many values with traditional looking dial-based, so called "steam" gauges. One example of these gauges in an existing popular piece of avionics is the multi-function display of the Garmin G1000 glass panel cockpit system, shown below.

Out of these interfaces, the only one that makes use of a touch screen is the Embraer cockpit, as opposed to the other's use of softkeys. As such, new forms of interaction had to be devised for many of the functions. One type of data entry where this was particularly of concern was the flight management functions, such as entering weight and balance, routes, performance, and configuration data. All the example aircraft mentioned above use a mechanical keypad and softkeys on a Flight Management Computer (FMC), or Master Control Display Unit (MCDU), where data is entered into a "scratchpad" before a softkey is used to commit the data to the interface, as shown below. This implementation used the original scratchpad concept from most FMC equipment, and simply uses a touch interface to enter the data, rather than softkeys.

F10 Audible Interface

Aircraft of all weights and sizes have audible warning systems, beginning with the most basic: a stall warning horn to indicate an approaching stall with a horn sound, and the most complex

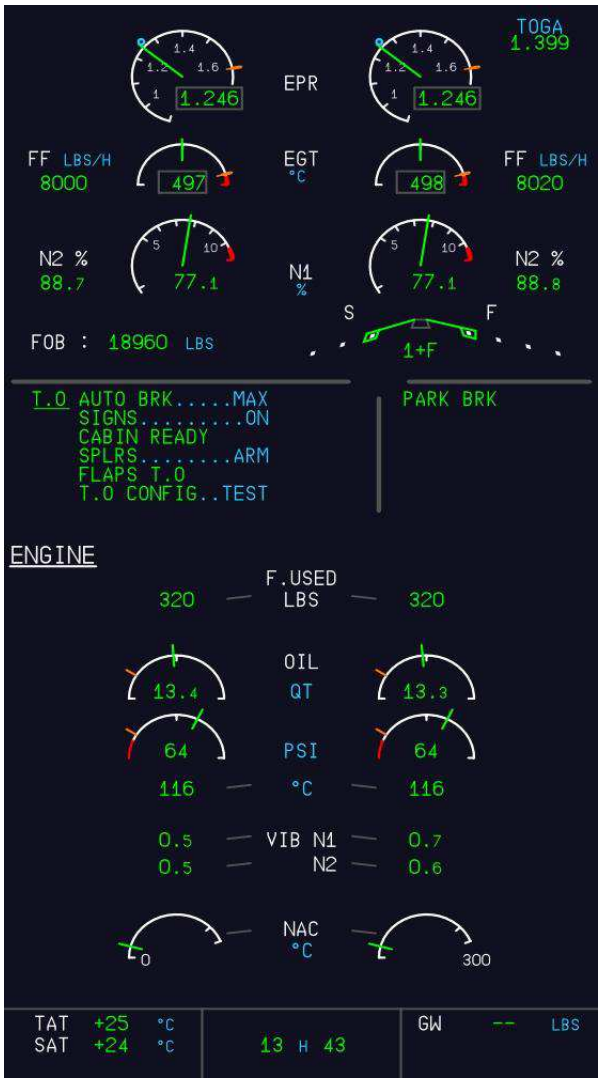


Figure F16: Airbus Engine and Flight Controls Displays

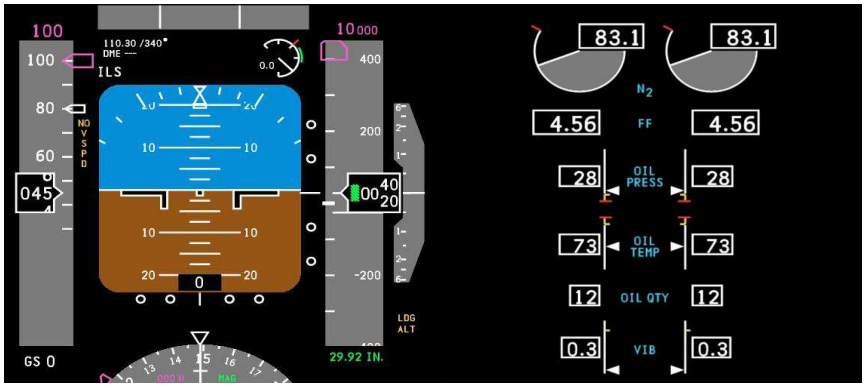


Figure F17: Boeing Primary Flight Display and Engine Instruments

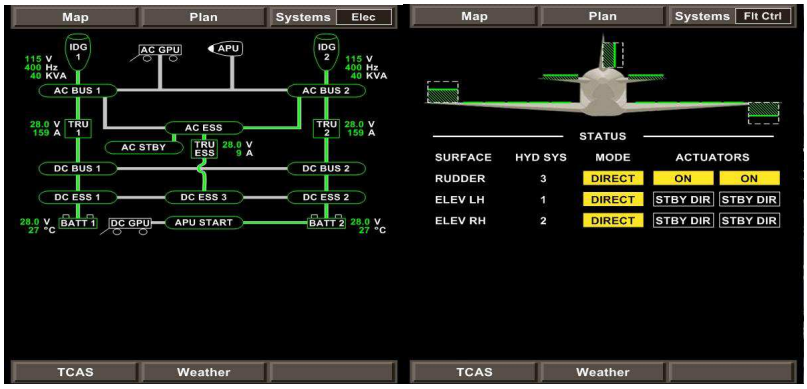


Figure F18: Embraer Electrics and Actuator Displays

Figure F19: Garmin G1000 Secondary Display

Figure F20: Boeing Flight Management Computer

having aural warning systems with recorded voice cues to instruct the flight crews. As N15726 was not equipped with any of these, the first order of business was adding a stall warning horn to the cockpit with an additional lead to the computer, to indicate a stall situation to the autoflight systems, and echo the stall warning horn through the headsets and cockpit speaker for triple redundancy.

When data is being entered in the FMC, a soft clicking sound is heard through the headsets and cockpit speaks to alert the flight crew of any unintentional data entry due to whatever circumstance. Other simple tone-based audible warnings include overspeed warning, 500ft to go until reaching a desired altitude, autopilot disconnect, electrical disconnect, configuration warnings, master caution, and fire warnings. Aural warnings carry different voices for their various sub-systems. In this case, a female voice for the equipment and configurations warnings, a high male voice for changing conditions of flight, and a deep male voice for traffic collision avoidance, and the ground proximity warning system.

Test of these audible warnings can be issued from the FMC debugging page for either method of sounding: headsets or cockpit speaker. They can also be disabled for either set of speakers via a switch on the panel of the aircraft. Volume is controlled via a knob for each set of speakers on the panel as well.

References

[1] Kim Topley, “Preface”, in *JavaFX Dev. Guid.* 2010.

[2] “JOGL - Java Binding for the OpenGL API”.

[3] “SwingNode (JavaFX 8)”, 2015.

Appendix G: Computer Vision for Lane and Horizon Detection

Nicholas S. Bradford

G1. Introduction

This section details the work done using computer vision to assist the ACAP Flight Computer system in localizing with respect to its environment model.

G2. Taxiway Lane Detection

In this section, we discuss our solution to the problem of localizing the aircraft position with respect to the airport taxiway lane markings. The use of GPS applied to the detailed airport maps possessed by the Flight Computer is sufficient for rough navigation, but is too imprecise to be practically viable for taxiway lane following (GPS is typically accurate to within ~16 feet, or 6 feet for high-end receivers [1]).

G2.1 Background

There is an abundance of published literature on the problem of detecting lanes for use on self-driving cars. However, airport taxiway lanes violate the assumptions presumed by these algorithms: taxiway lanes frequently appear and disappear from view, are not constrained to be roughly aligned with the aircraft's motion, and can intersect. At intersection points, the aircraft also may switch which lane it is following.

However, a useful basic template algorithm can still be extracted as common to most approaches [10], [11]. [12], [13], [14], [15]:

1. Filter the image with edge detection, or label lane pixels with a machine learning model
2. Initialize very rough lane estimates (potentially with Hough Line Transform)
3. Fit a line or curve model (often a polynomial or spline) with constrained RANSAC
4. Use hypothesis to update Kalman or particle filter

G2.2 Taxiway Dataset

Our available dataset is a five-minute video at 1920x1080 from a GoPro Hero 3+ mounted atop the rear vertical stabilizer on aircraft's tail (in the future, inputs may be extended to include complex information from the flight computer's environment model).

Figure 3: Sample image from the dataset.

The outputs are the pitch angle and bank angle detected by the algorithm, to be integrated by the Flight Computer's environment model. Each lane is approximated as a single line in form [offset distance in meters, orientation in degrees] from the nose of the plane. This simplifies model calculations, and is more intuitive to reason about when integrating into the Flight Computer environment.

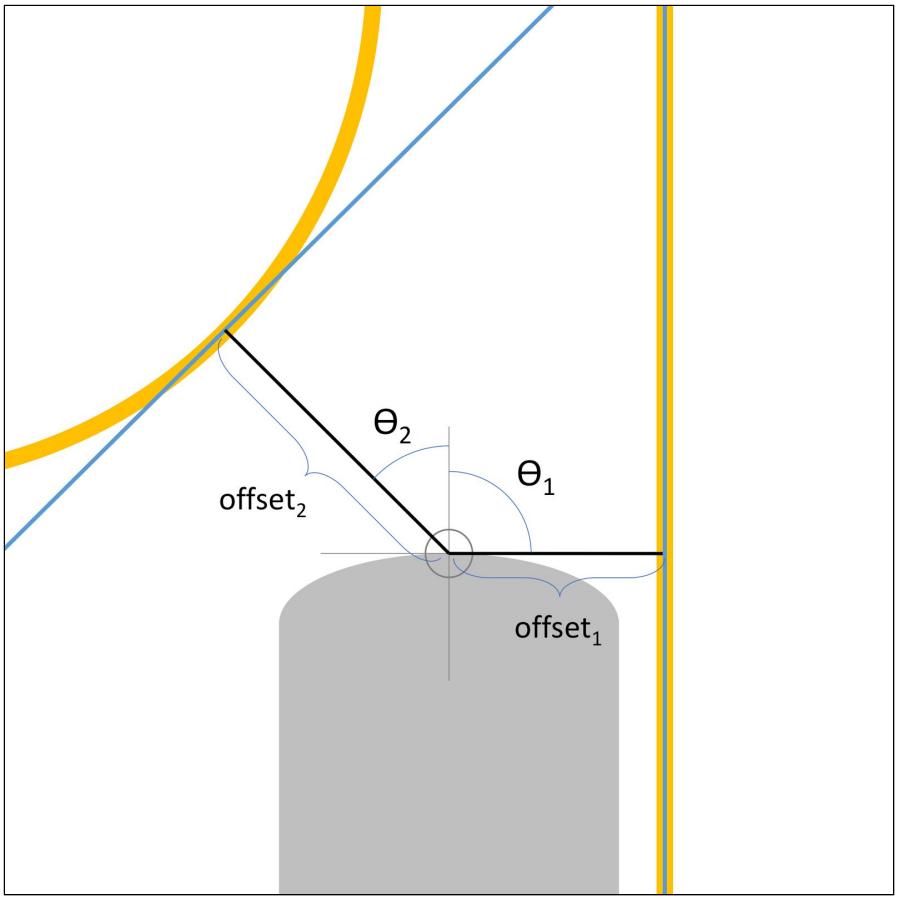


Figure 4: Model diagram. Each lane (yellow) is approximated as a single line (blue) in form [offset distance in meters, orientation in degrees] from the nose of the plane (grey).

Evaluation of our algorithm was done qualitatively by inspecting the output on the dataset (visualized by plotting the approximated lines onto the video). Quantitative testing was not conducted due to time constraints, as well as the small size of the training set and its lack of labels.

G2.3 Lane Detection Algorithm

Our final algorithm for lane detection uses sequential Random Sample Consensus (RANSAC) and particle filtering. Detected lanes are approximated as straight lines in the format of [offset distance in meters, orientation in degrees] from the nose of the aircraft:

1. Apply perspective transform to Region of Interest (ROI)
2. Update N-frame (2-frame) background model
3. Extract background to eliminate propeller motion
4. Extract yellow color
5. Dilate and erode
6. Apply morphological skeleton procedure to extract “centers” of lanes
7. Predict if there are multiple lanes using covariance matrix eigenvalues
8. Use sequential RANSAC to fit up to two lines to data
9. Update particle filtering models with RANSAC hypotheses
10. Return particle filter estimates

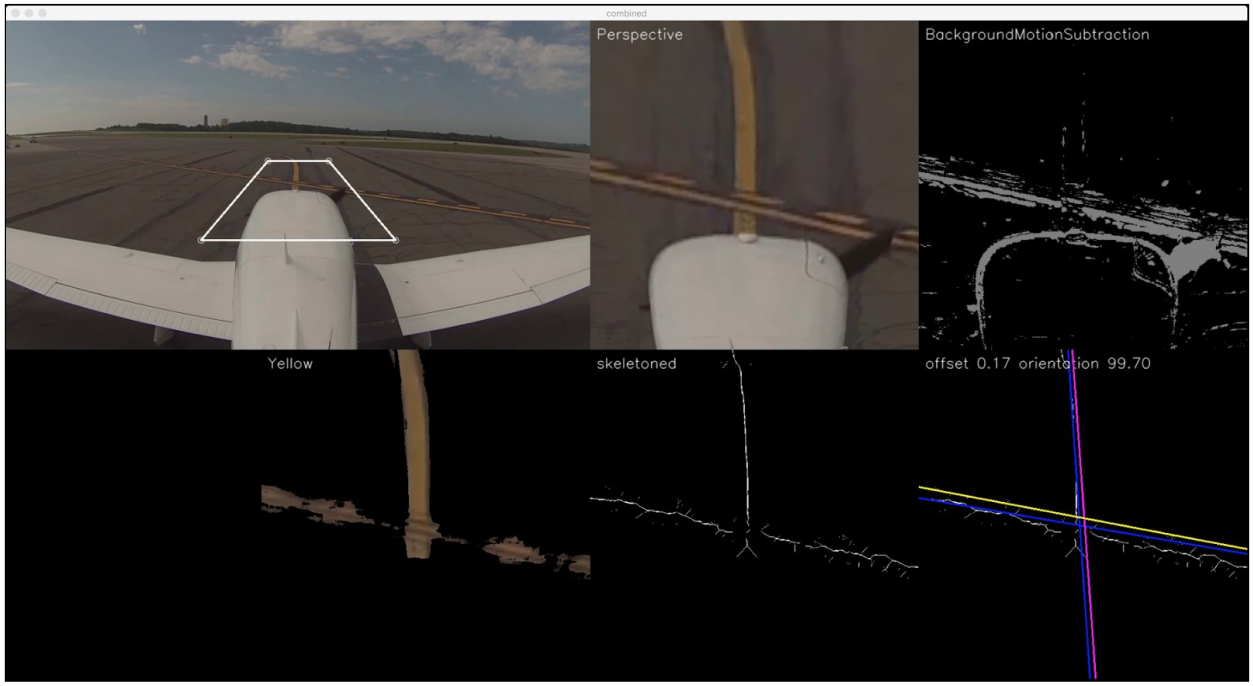


Figure 5: From top to bottom, left to right: The original video frame overlayed with the area to undergo a perspective transform in white; the perspective region; the background subtraction mask; the extracted yellow color; the results of the skeleton technique; fitting RANSAC (blue lines) and applying particle filter (magenta/yellow lines).

G2.3.1 Algorithm Assumptions

Our algorithm makes significant assumptions about the the taxiway environment, the benefits of which are described here:

- The runway is a flat plane of known constant orientation
    - Required for perspective transform
- The taxiway lane markings are clearly defined yellow
    - Required for color extraction
- The plane motion is reasonably slow
    - Required for background subtraction of the propeller
    - Required for particle filtering with no control model for plane motion
- There will never be more than 2 lanes in a single frame
    - Greatly simplifies sequential RANSAC procedure
- Each lane can be approximated as a single line in form [offset distance in meters, orientation in degrees] from the nose of the plane.
    - Simplifies model calculations, and is more intuitive to reason about when integrating into the Flight Computer environment

G2.3.2 Algorithm Parameters

A significant amount of development time involved tuning the following parameters:

- **Image resolution**: reduced to 540x540 for the transformed images
- **Size of perspective ROI**: kept small to keep with assumptions of nearly-straight lanes
- **Background subtraction frame count**: increasing produces a very smooth background model with totally eliminated propeller noise, but loses an unacceptable amount of information about the changing of the lane shapes. We found a two-frame model to strike the appropriate balance.
- **Yellow color in HSV space**: difficult to tune Saturation, as increasing range to include more of the lane marking also introduces a significant amount of noise.

- **Dilation and Erosion quantity:** balance between filling the gaps in yellow lanes, and emphasizing noise.
- **RANSAC parameters:** qualification of inliers, model validity, and several minor params.

G2.3.3 Multi-Lane Detection

Detecting when multiple lanes are present in an image is made difficult by the lack of constraints that can be made about them: they may intersect at a perpendicular angle, a very shallow angle, or not at all.

The naive multi-lane detection method would be to fit a single RANSAC model, check the percent of outliers detected by the fitting process, and declare a multi-lane scenario if that percent is above a certain threshold. However, we found this to be extremely unreliable in practice, especially when a second lane takes up only a very small percentage of the pixels in the image.

A more sophisticated approach would be to fit both single-lane and two-lane RANSAC models, compare the fitting errors, and choose the multi-lane model if its error is significantly lower than that of the single-lane model. However, this approach is dependent on having a running lane-fitting algorithm, has performance tightly coupled with the RANSAC parameters, and has a tunable parameter (the error difference threshold) is not necessarily intuitive to reason about.

Instead, we found that calculating the eigenvalues of the 2x2 covariance matrix of the skeletoned lane pixels and comparing their similarity gives a practically robust measure of whether there are multiple lanes in the image. The intuition is that the covariance matrix eigenvalues are a measure of the variance in the two principal components of the dataset (this is the same method as PCA-Principal Component Analysis), and so if both are similar in magnitude, the dataset spans multiple directions - indicative of multiple lanes.

Our final algorithm first checks the percentage of outliers in a single-lane RANSAC fit, and returns “single-lane” if it is below 20% (to save computation). Otherwise, it compares the covariance matrix eigenvectors, and returns “multi-lane” if they are within the same order of

magnitude. This combined method was qualitatively observed to correctly determine whether multiple lanes are present in an image in the vast majority of cases (quantitative testing was not conducted due to time constraints, as well as the small size of the training set and its lack of labels). Note that this method would fail for lanes intersecting at incredibly shallow angles, or a single lane curving sharply (45 degrees or more) - however, we did not encounter these scenarios while testing.

G2.3.4 Particle Filtering

Probabilistic filtering of state estimations is often achieved using the Kalman filter, which makes assumptions of a linear model with Gaussian posterior and sensor noise. However, because the noise in the created computer vision-based lane-detector is non-Gaussian, an alternative method is required [2]. Particle filtering is one such algorithm widely used in robotics applications, which uses a finite number of particles in a continuous state space to represent an arbitrary probability distribution [3].

The particle filtering algorithm has the following steps [4]:

```
def particleFilter (S, U, Z): # S is particles, U is control, Z is measurement
       S' = empty set
       for i = 0 ... n: # each new particle
              Sample s_j ~ {w} with replacement # weights of current S
              Estimate x' ~ p(x' | U, s_j) # new position given controls
              W' = p(z|x') # new particle weight is likelihood given estimate
              S' = S' union {< x', w'>} # add new particle to set
       for i = 0 ... n: # for each particle,  normalize weights
              W_i /= sum(W)
```

With each new frame, the hypothesis line produced by the RANSAC algorithm are compared to the elements of two particle filter models (one for each line), and matched with the model they are closest to. Our particle filter did not use a control model to estimate the plane's motion (though simple models would include modelling motion as continuously forwards, or as a moving average of the previous several video frames). However, in order to prevent all the

particles from collapsing to a single high-density location (nullifying the advantages of a particle distribution), random Gaussian noise was injected to each particle at the control application step.

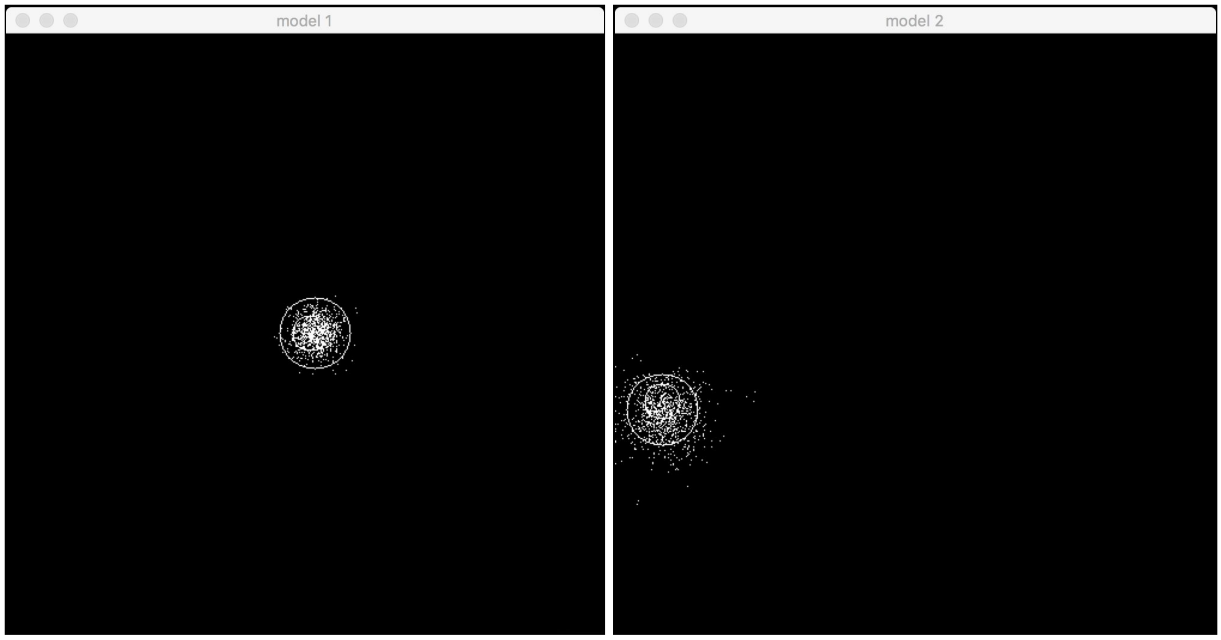


Figure 9: Visualization of the particle filters for model 1 (left) and model 2 (right) in [offset, orientation] space. The large circle represents the mean particle location (the filter estimate), while the small circle represents the most recent measurement. Note that the variance of the particles in model 1 is significantly less than that of model 2, due in this case to model 2's RANSAC estimates being very noisy as the lane comes into view for the first time.

An edge case arises when lanes appear and disappear from view, either in a single frame or permanently. We handle this by applying additional Gaussian noise in place of the evidence when a frame does not include a closely associated lane. Then, after N frames of no lane in view (we used N=5), the lane is declared to have left the field of view, and the particle filter is reset to its initial state of a uniform distribution.

G2.4 Implementation and Performance

The complete system was implemented with OpenCV-Python, with one-way communication of the algorithm's results transmitted to the Flight Computer control program

over Protobuf/ZMQ. Performance tests while running on Ubuntu 16.05 with 2.4GHz i7-4700MQ quad-core processor (the 2GB NVidia GT 755M graphics card is unused in OpenCV-Python) produced an average processing of 14-16 frames per second. Performance tests on the Flight Computer when integrated with the rest of the software have not yet been conducted.

This combined method was qualitatively observed to correctly determine the offset and orientation of one or two lanes in an image in the vast majority of cases, when filtering is active. Qualitative evaluation of our algorithm was done by inspecting the output on the dataset (visualized by plotting the approximated lines onto the video). Quantitative testing was not conducted due to time constraints, as well as the small size of the training set and its lack of labels.

G2.5 Future Work

- Obtain labelled dataset for quantitative testing (potentially using mean squared error on model parameters)
- Two-way ZMQ-Protobuf integration with the Flight Computer control software
- Model the particle filter's particle motion: either as a moving average of the previous changes; or as plane motion forward; or as plane motion towards the center line.
- Increase resolution (to detect lines properly at further distances)
- Perspective Transform: widen field, expand upwards to horizon
- Use ridges/edges instead of color (generates significant noise)
- Optimize with C++ and GPU support.
- Video Stabilization using Visual Odometry

G2.5.1 Alternative Algorithm

We also investigated an alternative algorithm that relies on prior knowledge of the expected lane layout, and then uses template matching to compute a posterior over possible aircraft locations. Although the reliance on an external map and the necessity for two-way communication caused us to decide against this algorithm for our initial work, we believe it may achieve higher performance than our algorithm in many circumstances [9].

G2.5.2 Machine Learning Approaches

Machine learning approaches to computer vision, particularly deep convolutional neural networks, have over the last several years shown “unreasonable effectiveness” at complex tasks ranging even to end-to-end learning of autonomous car steering.

However, two factors made us choose to forego the use of ConvNets in favor of more traditional computer vision techniques. Firstly, we were unable to find any large labelled datasets for airport taxiway lane detection, and procuring such a dataset would have been both time-consuming and monetarily expensive. Secondly, lane detection has a significant number of edge cases that would be difficult to embed into a machine learning model.

We expect that while machine learning may not be ideally suited in the short-term for the horizon detection and lane detection problems, it will undoubtedly play a significant role in future work.

G3. Horizon Detection

In this section, we discuss our solution to the problem of determining the pitch and bank angles of the aircraft by detecting the horizon line. This is useful for the aircraft's control system to better estimate its orientation while in-flight.

G3.1 Background

A simple approach to horizon detection might attempt to find a large line in the image, potentially after preprocessing with edge detection, but this is prone to error if there are other strong edges in the image [5]. Other methods attempt to segment the image in sky and ground portions by labelling each pixel according to a machine learning model or clustering technique, but this requires model training and is difficult to generalize across diverse sky/ground scenes [6], [7].

G3.2 Horizon Dataset

Our available dataset is a 186-second video at 1920x1080 from a GoPro Hero 3+ mounted underneath one of the aircraft's wings (to provide an unobstructed view) while in flight and making several banks in each direction.

Figure 13-1: Sample image from the dataset.

The outputs are the pitch angle and bank angle detected by the algorithm, to be integrated by the Flight Computer's environment model.

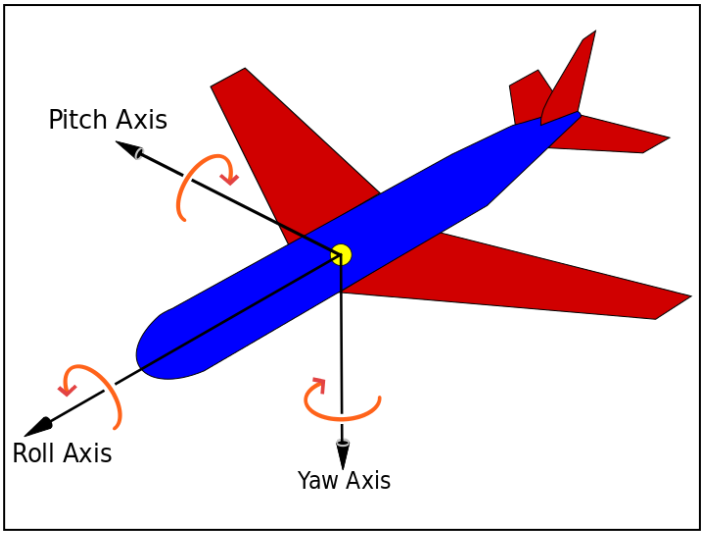


Figure 13-2: Aircraft principal axes of pitch, roll (also called "bank"), and yaw. Source: Wikimedia Commons.

G3.3 Algorithm

Our selected algorithm was introduced in 2003 by Ettinger, Scott M., et al., and was demonstrated to have robust performance over a wide range of different environments, even with exceptionally uneven horizons or distorted images [8]. By defining the horizon as a line that splits an image into two regions with maximum internal similarity, we may define a utility function using the eigenvalues of the covariance matrices of the two regions (across the three RGB color channels), and then search the [pitch angle, bank angle] parameter space to find the position with maximum utility:

$$J = \frac{1}{|\Sigma_s| + |\Sigma_g| + (\lambda_1^s + \lambda_2^s + \lambda_3^s)^2 + (\lambda_1^s + \lambda_2^s + \lambda_3^s)^2},$$

The intuition behind this function is that if a region's pixels are very internally similar, then the eigenvalues (and the determinant) of its 3x3 covariance matrix will be small. The first two terms in the denominator, $|\Sigma_S|$ and $|\Sigma_G|$ are the determinants for Sky and Ground. To make the utility function robust against cases where the covariance matrix is not full rank and the determinant approaches 0 (i.e. when an image has no information in a color channel), the second two terms are added as the squared sums of the eigenvalues for each region.

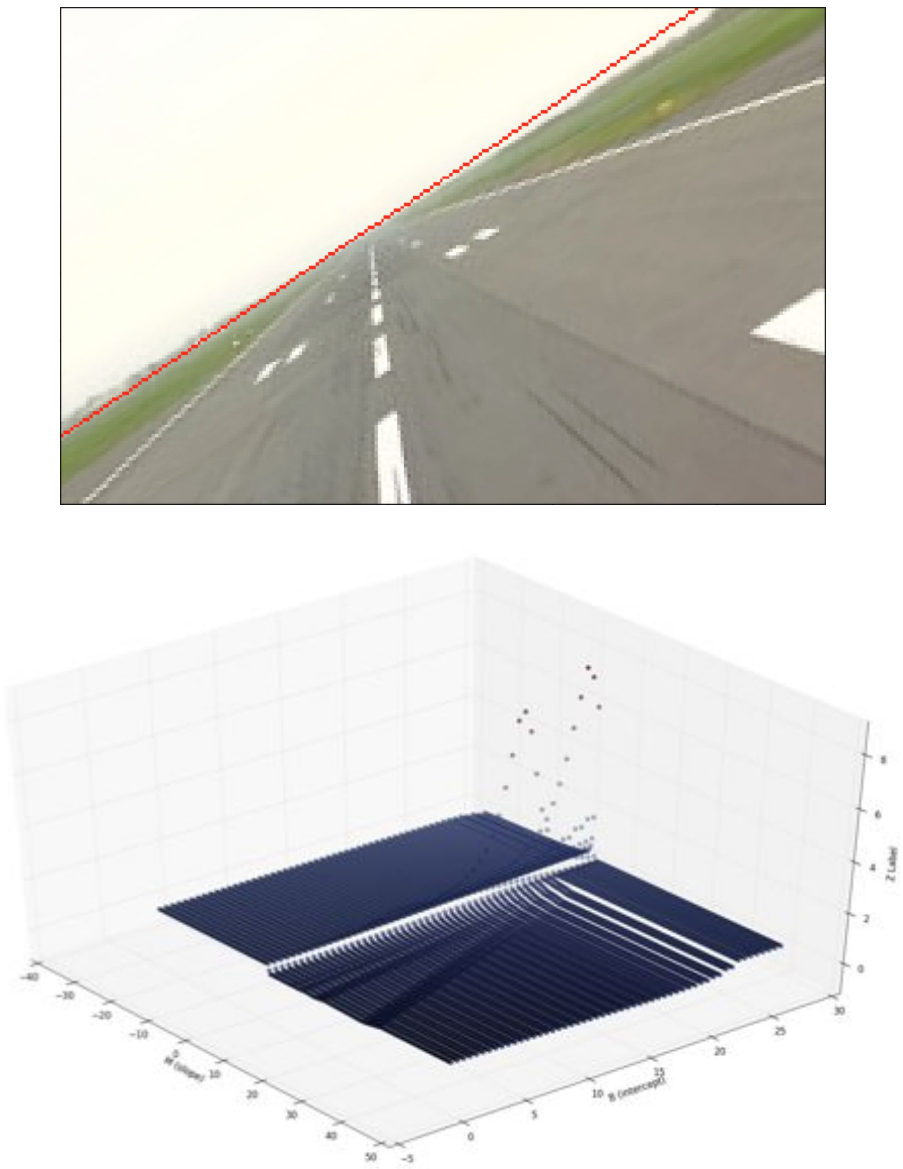


Figure 15: The algorithm running on an artificially rotated stock photo (top) with the detected horizon in red, and the cost function surface (here in transformed Slope-Intercept space instead of pitch-bank space)

Because this function is not convex, we initialize using a coarse grid search over the parameter space on a low-resolution (14x26) image (resized from 1920x1080). From there we run a hill-climbing bisection search on a high-resolution (144x256) image by moving in the direction of greatest ascent while reducing the step size. After several frames have been

processed and a horizon has been identified, the search space (and computation time) can be significantly reduced by only considering parameter sets in close vicinity to the previously detected line. Although [8] recommends against this optimization because of the potential for drift that prevents the algorithm from ever finding the true horizon again, we found that using proper initialization combined with use of the particle filter and periodic global checks should prevent catastrophic errors.

Figure 16: The calculated horizon line superimposed in red onto the input image.

G3.3.1 Failure Cases

- When an image is primarily a single color (such as a sepia-toned sunset or blue sky over the ocean), the algorithm is more likely to break down. This is because all hypothetical horizon lines will span the same color space and so have very similar eigenvalues.
- Though the algorithm can handle exceptionally irregular horizons, large unevenly distributed obstacles such as mountains or buildings can cause the algorithm to include

those obstacles in the Ground segment despite technically being above the true horizon line. This is especially problematic close to the ground.

- Cases of poor visibility, such as with clouds or fog, were not covered.

G3.4 Implementation and Performance

The complete system was implemented with OpenCV-Python and NumPy, with one-way communication of the algorithm's results transmitted to the Flight Computer control program over Protobuf/ZMQ. Performance tests while running on Ubuntu 16.05 with 2.4GHz i7-4700MQ quad-core processor (the 2GB NVidia GT 755M graphics card is unused in OpenCV-Python) produced an average processing of ~10 frames per second.

Our algorithm was qualitatively observed to correctly identify the horizon (within a reasonable margin of error, ~3 degrees) for the entirety of our dataset, despite failing on some individual images (as described in G3.3.1). Qualitative evaluation of our algorithm was done by inspecting the output on the dataset (visualized by plotting the approximated lines onto the video). Quantitative testing was not conducted due to time constraints, as well as the small size of the training set and its lack of labels.

G3.5 Future Work

- Obtain labelled dataset for quantitative testing (potentially using mean squared error on model parameters)
- Optimize calculation of utility function
- Optimize by switching to C++ (and potentially GPU support)
- Extreme attitude detection for when no horizon is visible (covered in [8])
- Add particle filtering

G.4 References

1. GPS.gov. *GPS Accuracy*. Accessed 10 October 2016 from http://www.gps.gov/systems/gps/performance/accuracy/
2. Frew, Eric, et al. "Vision-based navigation for airfield surface operation." *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2008.
3. Thrun, Sebastian. "Particle filters in robotics." *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2002.
4. Thrun, Sebastian. "Intro to Artificial Intelligence: Particle Filter Algorithm." *Udacity*. Accessed from https://classroom.udacity.com/courses/cs271/lessons/48734405/concepts/487251990923#
5. SFWA UAV. *Horizon detection with OpenCV*. Accessed January 2017 from http://au.tono.my/log/20130722-horizon-detection.html
6. Boroujeni, Nasim Sepehri, S. Ali Etemad, and Anthony Whitehead. "Robust horizon detection using segmentation for uav applications." *Computer and Robot Vision (CRV), 2012 Ninth Conference on*. IEEE, 2012.
7. Neto, A. Miranda, et al. "Robust horizon finding algorithm for real-time autonomous navigation based on monocular vision." *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*. IEEE, 2011.
8. Ettinger, Scott M., et al. "Towards flight autonomy: Vision-based horizon detection for micro air vehicles." Florida Conference on Recent Advances in Robotics. Vol. 2002. 2002.
9. Lu, Bowen, Baibing Li, and Wen-Hua Chen. "Map-Enhanced Visual Taxiway Extraction for Autonomous Taxiing of UAVs** This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) Autonomous and Intelligent Systems programme under the grant number EP/J011525/1 with BAE Systems as the leading industrial partner." *IFAC-PapersOnLine* 48.9 (2015): 49-54.
10. Kim, ZuWhan. "Robust lane detection and tracking in challenging scenarios." *IEEE Transactions on Intelligent Transportation Systems* 9.1 (2008): 16-26.

11. Wang, Yue, Eam Khwang Teoh, and Dinggang Shen. "Lane detection and tracking using B-Snake." *Image and Vision computing* 22.4 (2004): 269-280.
12. Aly, Mohamed. "Real time detection of lane markers in urban streets." *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008.
13. López, A., et al. "Robust lane markings detection and road geometry computation." *International Journal of Automotive Technology* 11.3 (2010): 395-407.
14. McCall, Joel C., and Mohan M. Trivedi. "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation." *IEEE transactions on intelligent transportation systems* 7.1 (2006): 20-37.
15. Nieto, Marcos, Jon Arróspide Laborda, and Luis Salgado. "Road environment modeling using robust perspective analysis and recursive Bayesian segmentation." *Machine Vision and Applications* 22.6 (2011): 927-945.

Appendix H

Traffic Collision Avoidance System(TCAS)

Written By: Zhaochen Ding

Table of Contents

H1 Introduction

One challenge that stands before ACAP aircraft is the potential mid-air collision with other airplanes. In order to warn controller(pilot) the presence of other transponder-equipped aircraft which may present a threat of mid-air collision(MAC), our team implemented traffic collision avoidance system(TCAS) that can issue traffic advisory(TA) and resolution advisory(RA) annunciations.

H1.1 Background

TCAS II is an embedded system that provides on-board aircraft conflict detection and resolution which evolved from original operational evaluation prototype in 1982 through plenty of deployment versions. TCAS II is currently the mandatory system in US for aircraft with greater than 30 seats or a takeoff weight over 33,000 pounds and is also installed on a huge portion of turbine-powered general aviation aircraft. The main system is designed to alert controller(pilot) the existence of nearby aircraft that may lead to a MAC and issue a guidance that can assist controller(pilot) to resolve these potential conflicts. When a collision threat is detected, TCAS will estimate the time remaining until the two aircraft reaches the closest point of approach(CPA) and send out two different levels of alerts. As the figure below shows[1], when the distance between the current aircraft position and an intruder aircraft position becomes less than the outermost protected volume, TCAS will issue a Traffic Advisory(TA) which will show the controller(pilot) the position of intruder aircraft. If the distance is still decreasing, a Resolution Advisory that is likely to solve the current situation by guiding the controller(pilot) to maintain or increase the vertical separation with intruder aircraft will be sent by TCAS II to the controller(pilot). The Implementation of TCAS imitates the way Introduction to *TCAS II Version 7.1*[2] issued by Federal Aviation Administration(FAA).

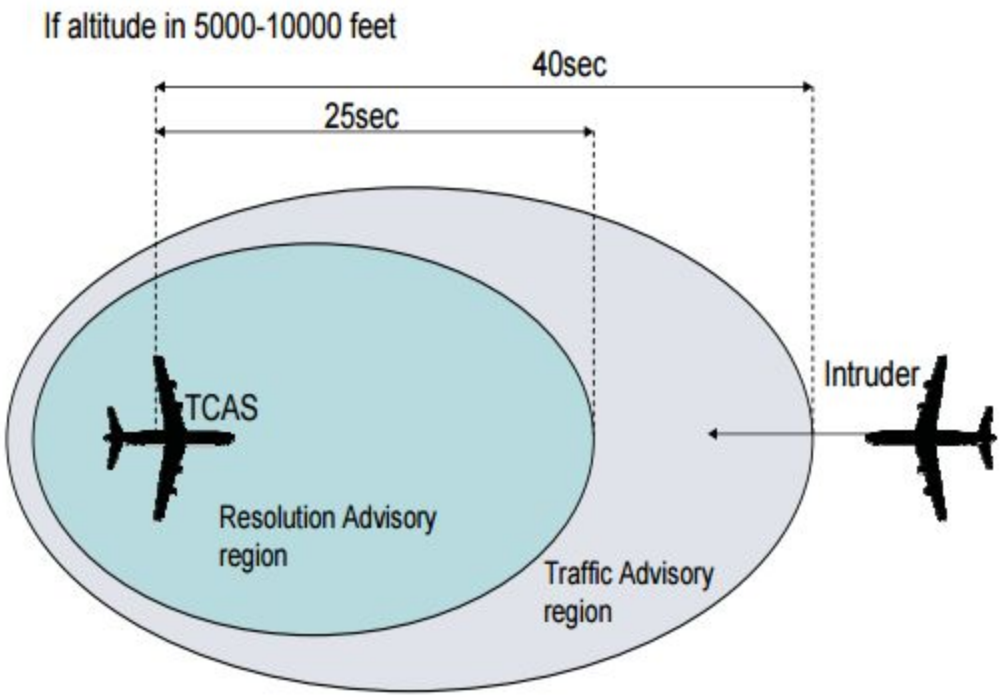


Figure A1: TCAS example when altitude is in 5000 - 10000 feet

H1.2 Hardware and external libraries

Hardware:

- A 1090 MHz ADS-B antenna
- A FlightAware ADS-B 1090 MHz Band-pass SMA Filter
- FlightAware Pro Stick USB ADS-B Receiver
- An Unix-based laptop

Library:

- Json-java.jar
- Apache Commons Maths
- Dump1090
- RTL-SDR

H2 Program Structure

The flow of this project strictly followed the flowchart in FAA distributed booklet, which looked like the figure below[2]:

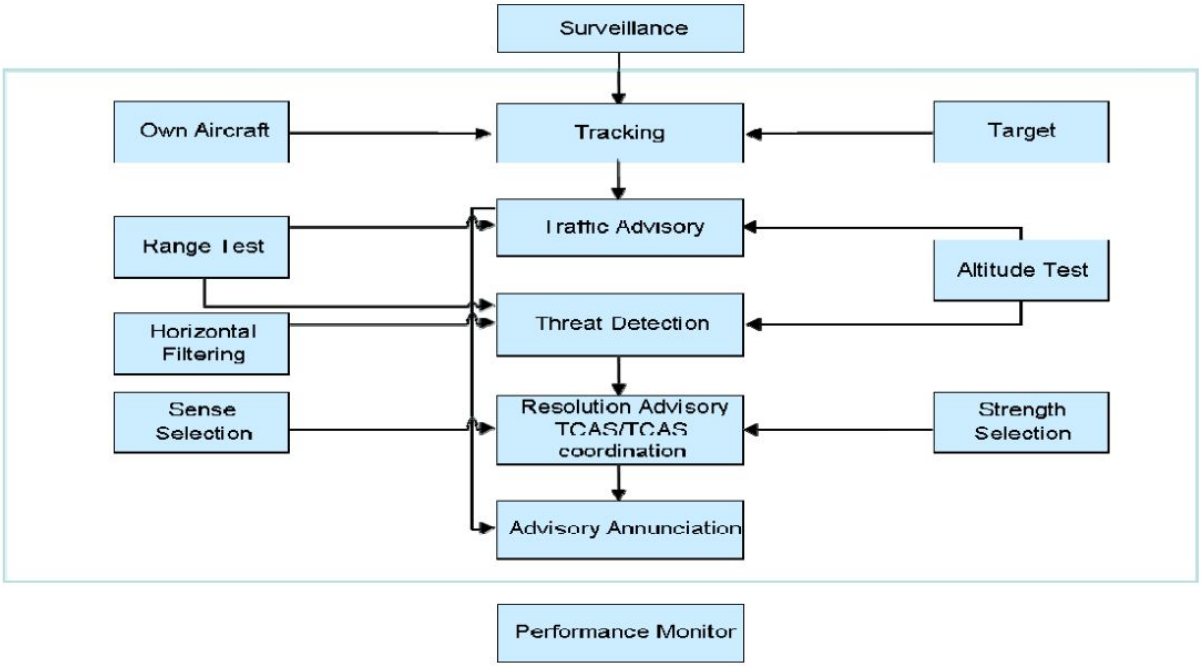


Figure A2: Flow chart of TCAS main system

When the system received the nearby airplane data, it begins to figure out if the own aircraft and the intruder have potential collision possibility inside TA time range, if one intruder passed both the range test and altitude test the system will send a TA to controller(pilot), and the system will check the range and altitude with RA standard, if it also passed the RA detection, the data of this intruder and own aircraft will be utilized to test the sense and strength. Normally in strength selection, TCAS will choose from sending preventive or corrective RAs, but due to safety concerns(corrective RAs may lead to a situation where owner aircraft enter another plane's track), we choose not to let our system generate any corrective RAs. After choosing the correct RA, TCAS will send the final result and solution to the controller(pilot).

H3 Methodology

H3.1 Obtaining the nearby airplane data

Normally, TCAS is based on secondary surveillance radar(SSR) signals, however, since our team didn't have access to SSR, we connected a 1090MHz ADS-B antenna, a FlightAware 1090 MHz Mode S Filter, and a FlightAware Mode S Data Receiver to collect raw data from nearby airplanes. After installing RTL-SDR that supports the antenna and dump1090 that decodes raw data on unix/linux machine, the decoded data of nearby airplanes will be displayed in shell, and a google map that displays nearby aircraft's positions will be available on localhost:8080 if running dump1090 with internet configuration. The command is: {./dump1090 --interactive --net --net-http-port 8080}[3]. Then we created a bash script that curls the JSON file from localhost using command[sudo curl localhost:8080/data.json >> data.json] once per second.

The JSON file acquired will contain all nearby aircraft data with 7 attributes:

Hex: Six digit ICAO aircraft ID hex code

Flight: Flight number of this aircraft(e.g. JBU1012)

Longitude: Current longitude of this aircraft

Latitude:Current latitude of this aircraft

Altitude: Current altitude of this aircraft

Track: Current horizontal bearing angle of this aircraft

Speed: Current ground speed of this aircraft

After curling the json file, we used the external JAR java-json.jar to parse the data into aircraft struct so that the project can make use of these data.

Methods in TCAS are mainly based on the collision avoidance system logic(CAS), the first analysis in CAS logic is to check the sensitivity level, which controls the time threshold for TA and RA issuance. The relationship between sensitivity level and TA or RA alarm thresholds is displayed in the table below[4].

Own Altitude (feet)	SL	Tau (Seconds)		DMOD (nmi)		ZTHR (feet) Altitude Threshold		ALIM (feet)
		TA	RA	TA	RA	TA	RA	RA
< 1000 (AGL)	2	20	N/A	0.30	N/A	850	N/A	N/A
1000 - 2350 (AGL)	3	25	15	0.33	0.20	850	600	300
2350 – 5000	4	30	20	0.48	0.35	850	600	300
5000 – 10000	5	40	25	0.75	0.55	850	600	350
10000 – 20000	6	45	30	1.00	0.80	850	600	400
20000 – 42000	7	48	35	1.30	1.10	850	700	600
> 42000	7	48	35	1.30	1.10	1200	800	700

Table A1. Sensitivity Level Definition and Alarm Thresholds

After knowing the sensitivity level, we need to compute the closest point of approach(CPA) and then calculate a time to reach CPA and check if the time is inside the threshold. Although plenty of published paperwork about TCAS can be found, only one of them created by NASA contained a standard mathematics algorithm for RA based on vector calculations. Moreover, due to the limitation on resource illustrated earlier, we weren't able to receive the closure rate from radar, this approach may return inaccurate or even flawed result if we forcibly use the immediate velocity calculated by time, longitude and latitude. Consequently, we decided to create our own algorithm to judge if any intruder needs to be reported as well as attempting to implement the algorithm in the paper to see the performance. Based on the information we collected before starting, we divided TCAS into two judgements: horizontal judge and vertical judge for both TA and RA. Since the only difference for TA and RA in the judging phase are the threshold value and the output, this report will discuss the algorithms for TA and RA together.

Normally, the possibility for horizontal coincidence is much smaller than vertical coincidence. So when one considers TA or RA signal, the first calculation should always be solving the horizontal Closest Point of Approach(CPA). Our team came up with two different methods to accomplish this task.

H3.2 CPA (Geometrical Approach)

From attributes read from json file, we have the longitude, latitude, horizontal heading angle and ground speed. This approach first takes in two pairs of longitude and latitude (current aircraft and intruder aircraft) and calculate the distance and relative angle using Haversine Formula by James Andrew in 1805:

$$\operatorname{haversin}\left(\frac{d}{r}\right) = \operatorname{haversin}(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)\operatorname{haversin}(\lambda_2 - \lambda_1)$$

In this formula, d is the distance between the two points on a sphere, r is the radius of the sphere, namely 6371 km for earth, ϕ_1, ϕ_2 are the latitude of point1 and point2, and λ_1, λ_2 are the longitude of point1 and point2. The haversin(angle) formula is the haversine function:

$$\operatorname{haversin}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

The bearing angle can be calculated with formula below:

$$\theta = \operatorname{atan2}(\sin(\lambda_2 - \lambda_1) \cdot \cos(\varphi_2), \cos(\varphi_1) \cdot \sin(\varphi_2) - \sin(\varphi_1) \cdot \cos(\varphi_2) \cdot \cos(\lambda_2 - \lambda_1))$$

The the time before possible CPA is just the distance between two points divided by the component of two planes' net velocity in the bearing angle as the figure below[5] shows:

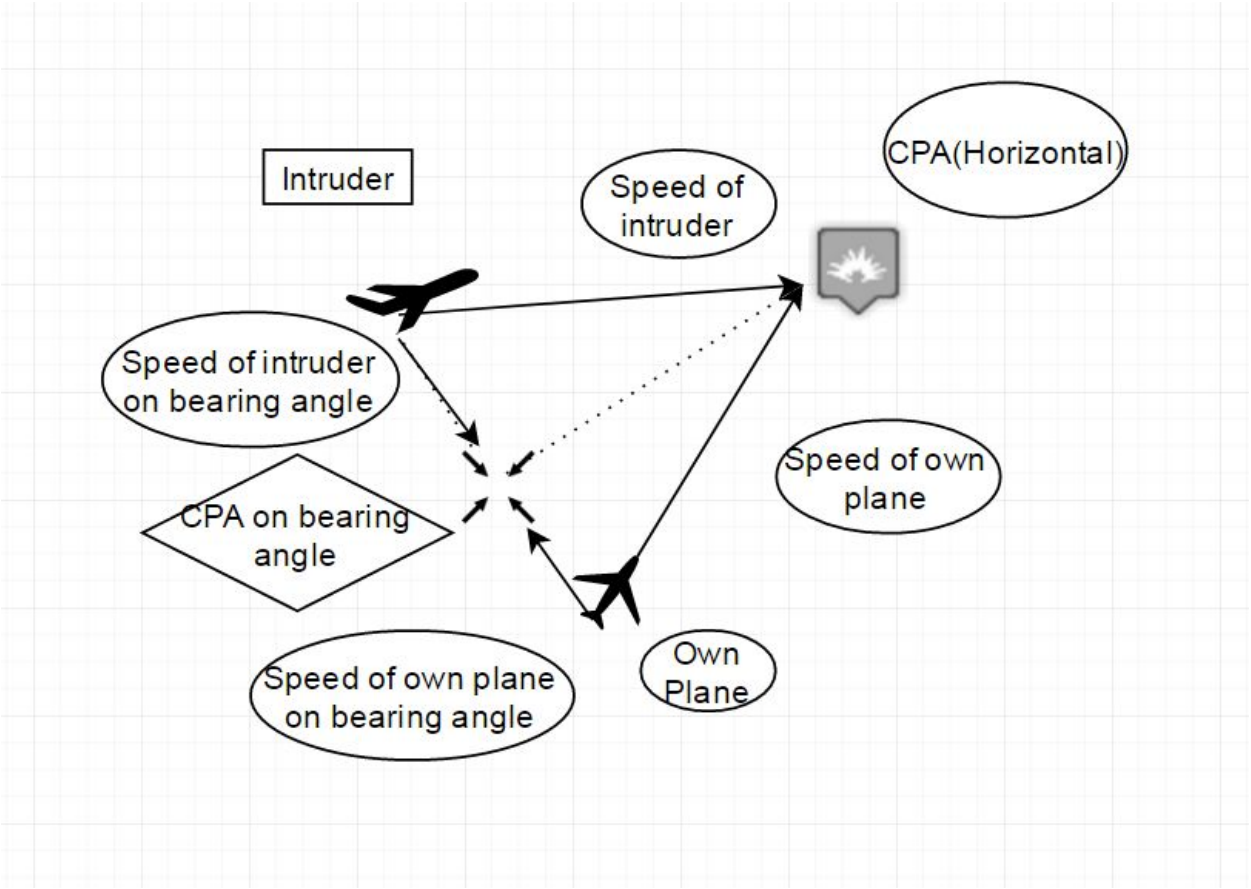


Figure A3: Geometric Evaluation of CPA

H3.3 CPA (Linear Regression Approach)

Another algorithm is based on linear regression and machine learning. Whenever a prior detected plane is loaded, the algorithm will save this information into an array. Based on these data, the algorithm will first select an origin(The longitude and latitude of own aircraft this time) and setup an axis system in which every plane will have a separate point when the plane's data is parsed. The algorithm will then apply a linear fit to the points to get the track of the plane, the final crossing point between track of current plane and intruder plane should be the CPA. Also, if the origin is already too far away, it will set the current own plane as origin. The pseudo code looks like following:

if(origin to location of current plane is more than 5 degree){

Change origin to current location

}

ForEach (planes loaded){

If (plane exist in prior data){

if(this plane's data array is full){

Delete the first data in data array

Add the latitude and longitude pair into this data array

}

Else{ add this pair into data array}

Complete a linear fit to check if this line crosses with the line for owner airplane, if so return the time to CPA

Else{continue to the next plane}

}

H3.4 CPA(Mathematical Model in NASA paper)

Since the horizontal and vertical analysis can be completed separately, three-dimensional model is not necessary in this program. However, a two-dimensional model based on two-dimensional Euclidean geometry was set up to help analyze the horizontal motions of the two aircrafts. With the prerequisite that the aircrafts are moving in a constant speed, the position of both aircraft can be derived in the following formula

$$\mathrm{s(t)} = s_0 + \mathrm{Vt}$$

Now, this model can be simplified by considering the one pair of relative velocity and displacement and other pair equals to zero instead of the two pairs of changing velocity and displacement. In this case, we can easily get a conclusion: if the relative position(s) and velocity(v) are both positive or negative, then the two aircrafts are traveling against each other's direction, but if one of them is positive and another is negative then the two planes are traveling

towards each other. So if displacement is d and velocity is v, if s·v > 0, the horizontal judge can be concluded into the equation below:

$$\text{horiRA} = \text{s} < DMOD_{SL}$$

when s·v < 0, the horizontal judge can be concluded into the equation below:

$$\text{horiRA} = \ Tau_{SL} \geq (DMOD_{SL}^2 - s^2)/s \bullet v$$

Similarly, if s·v > 0, the vertical judge can be concluded into the equation below:

$$\text{vertRA} = \text{s} < ZTHR_{SL}$$

when s·v < 0, the horizontal judge can be concluded into the equation below:

$$\text{vertRA} = \ Tau_{SL} \geq -\frac{\text{s}}{\text{v}}$$

Since an intruder needs to satisfy both horizontal TA/RA and vertical TA/RA before a signal will be sent to controller(pilot), we concluded an algorithm of TA/RA to detect if TA or RA will be issued:

$$\text{RA}(s_{horizontal}, v_{horizontal}, s_{vertical}, v_{vertical}) = horiRA(s_{horizontal}, v_{horizontal}) \ \&\& \ vertRA(s_{vertical}, v_{vertical})$$

$$\text{TA}(s_{horizontal}, v_{horizontal}, s_{vertical}, v_{vertical}) = horiTA(s_{horizontal}, v_{horizontal}) \ \&\& \ vertTA(s_{vertical}, v_{vertical})$$

Finally, we considered the factor of time by changing the s in each equation into s + v*t while t is the time in the future. Also we can use this formula to calculate the time cost before two aircrafts meet each other and compare this time with the time frame by switching input and output[6].

After receiving the time before CPA is reached, the algorithm can now compare the time with tau threshold to see if horizontal TA/RA is triggered.

However, even if horizontal CPA lands inside the tau, TCAS cannot fire alarm at once since currently the time to CPA is just for horizontal judgement. Since dump1090 output we acquired does not show vertical speed, we have to consider an approach to obtain the vertical speed. We decided to save the last loaded aircraft data in an array and compare them with the

newly loaded aircraft data. The vertical closure rate(vertical relative speed) at this time frame should be the altitude change divided by time. As long as we know the distance and speed, we can calculate the the two plane's vertical CPA. If vertical CPA is also inside range, TCAS will introduce the current TA/RA status to the controller(pilot).

H3.5 RA Message(Sense and Strength Selection)

Since we simplified the problem in this report, the illustration may imperceptibly pass a fake message that TA and RA are similar. However, the processes after determining an intruder are totally different. TA will only report the position to the controller(pilot), but RA needs to send out a solution from the solution base judging from the status in CPA.

RA Type	Upward Sense		Downward Sense	
	RA	Required Vertical Rate (fpm)	RA	Required Vertical Rate (fpm)
Positive (Corrective)	Climb	1500 to 2000	Descend	-1500 to -2000
Positive (Corrective)	Crossing Climb	1500 to 2000	Crossing Descend	-1500 to -2000
Positive (Corrective)	Crossing Maintain Climb	1500 to 4400	Crossing Maintain Descend	-1500 to -4400
Positive (Corrective)	Maintain Climb	1500 to 4400	Maintain Descend	-1500 to -4400
Negative (Corrective)	Reduce Descent	0	Reduce Climb	0
*Negative (Corrective)	Reduce Descent	> -500	Reduce Climb	< 500
*Negative (Corrective)	Reduce Descent	> -1000	Reduce Climb	< 1000
*Negative (Corrective)	Reduce Descent	> -2000	Reduce Climb	< 2000
Negative (Preventive)	Do Not Descend	> 0	Do Not Climb	< 0
Negative (Preventive)	Do Not Descend > 500 fpm	> -500	Do Not Climb > 500 fpm	< 500
Negative (Preventive)	Do Not Descend > 1000 fpm	> -1000	Do Not Climb > 1000 fpm	< 1000
Negative (Preventive)	Do Not Descend > 2000 fpm	> -2000	Do Not Climb > 2000 fpm	< 2000

Table A2: Sense Selection for RA

When TCAS figured out a signal is required, it will compare the closure rate with the threshold in this table to check which type of RA it should return, the figure below shows an example[7].

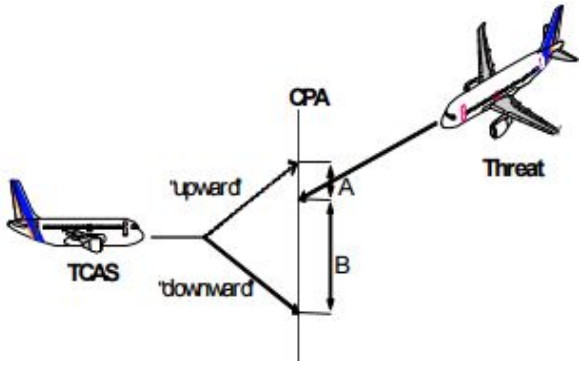


Figure A4: How RA sense selection works

As mentioned before in Section 2, due to safety concerns, we choose to set all our RA signals to be preventive so that the aircraft will not head into another plane's track even the system went disordered.

H4 Performance, Optimization and Future Work

H4.1 Performance

After a general test on these algorithms, we found the linear fit attempt costs the longest time since it loads the biggest quantity of data, and is the most sensitive to the speed change. The algorithm performed especially poor when a plane is taking off or landing due to the fast and frequent velocity change. However, if the speed remains unchanged at most of time, this attempt will become the most accurate one. The geometry attempt takes less space and time to run, and it works pretty fine even the velocity is changing frequently, but when there's an aircraft in RA range, sometimes the distance change cannot be detected by latitude and longitude if the data is rounded into one effective digit (that's what dump1090 does with the net configuration), thus resulting in a situation that the intruder stays stationary for two seconds and suddenly had a huge velocity in one second. The NASA recorded algorithm performed similar to the geometric

approach since they are almost same method besides one is based on vector calculation and the other is based on angular velocity and trigonometry functions.

H4.2 Optimization

TCAS is a part of project that need to always be executed with a high frequency as long as the aircraft is in the air, plus when an intruder inside RA range is detected, the frequency will become once per second, so the performance of program tends to be an extremely crucial factor. The initial design used two arrays to save aircrafts, two hashmaps to save aircraft and time pair, and most methods takes O(n^2) time to execute while several methods requires huge time complexity such as O(n^3). After the initial design, the optimization in time and space complexity took quite amount of time and the final version only requires two arrays and most of the methods are of O(n) time complexity. With these optimized methods, the current TCAS system succeeded in running continuously on an Intel I5-4690 CPU 16GB memory desktop for 14 hour, which means it can keep running throughout a trip from Beijing to New York.

H4.3 Future Work

1. Adding RA solution to edge cases (e.g. when flight reached max height)
2. Figure out a better way to calculate closure rate
3. Adding horizontal RA resolutions.
4. Getting directional antenna so that it can assign bearing to other aircraft directly.

H5 Reference

[1] A. Gotlieb, “TCAS software verification using Constraint Programming”, 2012 https://hal.archives-ouvertes.fr/hal-00807905/document

[2] FAA.gov, ”TCAS II V7.1Intro booklet”, Feb. 28 2011 https://www.faa.gov/documentLibrary/media/Advisory_Circular/TCAS%20II%20V7.1%20Intro%20booklet.pdf

[3] Salvatore Sanfilippo, Dump1090 README https://github.com/antirez/dump1090

[4] FAA.gov, “TCAS II V7.1Intro booklet”, Feb. 28 2011 https://www.faa.gov/documentLibrary/media/Advisory_Circular/TCAS%20II%20V7.1%20Intro%20booklet.pdf

[5] Heinz Erzberger, “Separation Assurance In The Future Air Traffic System”, Mar. 2009 http://www.enri.go.jp/eiwac/2009/ppts/SeparationAssuranceInTheFutureAirTrafficSystem.pdf

[6] NASA, “TCAS-II Resolution Advisory Detection Algorithm” https://shemesh.larc.nasa.gov/people/cam/publications/gnc2013-draft.pdf

[7] TCAS II V7.1Intro booklet, Feb. 28 2011 https://www.faa.gov/documentLibrary/media/Advisory_Circular/TCAS%20II%20V7.1%20Intro%20booklet.pdf