University of Colorado
Department of Aerospace Engineering Sciences
Senior Projects - ASEN 4028

Windmill Hindrance and Maneuverable Propulsion System (WHiMPS)

Project Final Report

Monday 4th May, 2020

Information

Project Customers

Name: Riley Huff, 1LT Email: riley.huff.1@us.af.mil Phone: 937-656-4098

Team Members

Name: Alec Bosshart Email: Alec.Bosshart@Colorado.edu Phone: 303-718-1090	Name: Andrew Robins Email:Andrew.B.Robins@Colorado.edu Phone: 781-580-9170
Name: Isobel Griffin Email: Isobel.Griffin@Colorado.edu Phone: 720-937-3056	Name: Liam Sheffer Email: Liam.Sheffer@Colorado.edu Phone: 719-432-6948
Name: Julia Kincaid Email: Julia.Kincaid@Colorado.edu Phone: 720-951-0019	Name: Jon Weidner Email: jowe3555@colorado.edu Phone: 410-303-7280
Name: Andrew Meikle Email: Andrew.Meikle@Colorado.edu Phone: 719-650-8975	Name: Zoe Witte Email: Zoe.Witte@Colorado.edu Phone: 720-880-8592
Name: Declan Murray Email: Declan.Murray@Colorado.edu Phone: 303-895-9041	Name: Lucas Zardini Email: luza2695@colorado.edu Phone: 331-444-9636
Name: Alexandra Paquin Email:Alexandra.Paquin@Colorado.edu Phone: 720-354-2238	Name: Nicholas Zellmann Email:Nicholas.Zellmann@Colorado.edu Phone: 719-419-4511

Contents

List of Figures

List of Tables

Acronyms

ABS	Acrylonitrile Butadiene Styrene
ADC	Analog to Digital Converter
AFRL	Air Force Research Laboratory
APOP	Aerospace Propulsion Outreach Program
CAD	Computer-Aided Design
CDD	Conceptual Design Document
CONOPS	Concept of Operations
COTS	Commercial Off The Shelf
CU	University of Colorado
DAQ	Data Acquisition
DR	Design Requirement
ECU	Electronic Control Unit
ECM	Electronic Control Module
EGT	Exhaust Gas Temperature
ESD	Electrostatic Discharge
ETCS	Engine Test and Control System
FBD	Functional Block Diagram
FR	Functional Requirement
GND	Ground
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
I2C	Inter-Integrated Circuit
ICUP	Integrated Communication Unit Panel
IDE	Integrated Development Environment
LED	Light-Emitting Diode
LPBB	Linear Potentiometer Breakout Board
MCB	Main Control Board
NI	National Instruments
PAB	Professional Advisory Board
PCB	Printed Circuit Board
PID	Proportional Integral Derivative Controller
POC	Point of Contact
RPM	Rotations Per Minute
SE	Systems Engineer
SPECS	Specialized Propulsion Electronics Control Systems
SSH	Secure Shell
SWIFT	Supersonic Wind Imagining Flow Tunnel
TVM	Thrust Vectoring Mechanism
UAV	Unmanned Autonomous Vehicle
USAF	United States Air Force
USAFA	United States Air Force Academy
USB	Universal Serial Bus
USD	United States Dollar
WBS	Work Breakdown Structure
WPM	Windmill Prevention Mechanism
WHiMPS	Windmill Hindrance and Maneuverable Propulsion System

Nomenclature

α	Paddle Deflection Angle [°]
A_n	Area of Nozzle [mm^2]
A_p	Area of Paddle [mm^2]
d	Actuator Deflectiion [mm]
F	Total Thrust [lb-f]
F_{aero}	Aerodynamic Force [lb-f]
F_{axial}	Axial Thrust Component [lb-f]
F_{drag}	Drag Force [lb-f]
F_{perp}	Perpendicular Thrust Component [lb-f]
F_{pin}	Pin Force [lb-f]
F_{spring}	Spring Force [lb-f]
F_z	Non-Axial Thrust Component [lb-f]
h_x	Convective Heat Transfer Coefficient
k	Thermal Conductivity [$\frac{W}{m*K}$]
L_{aero}	Center of Pressure Location [in]
L_{drag}	Drag Location [in]
L_{pin}	Length of Pin [in]
L_{spring}	Length of Spring [in]
Nu	Nusselt Number
P_p	Primary Fluid Momentum [$\frac{kg}{m-s}$]
P_r	Secondary Fluid Momentum [$\frac{kg}{m-s}$]
P_s	Resultant Fluid Momentum [$\frac{kg}{m-s}$]
Pr	Prandtl Number
Re_x	Reynold's number
θ	Flow Deflection Angle [°]
θ_n	Paddle n Deflection Angle [°]
x	Characteristic Length [m]

1. Project Purpose

Authors: Isobel Griffin

The purpose of WHiMPS is to modify the JetCat P100-RX engine, shown in Figure 1, to allow for at least one axis of thrust vectoring capabilities greater than or equal to ten degrees with no decrement of thrust at zero degrees of vectoring, and to prevent windmilling with the engine off up to Mach 0.8 at 20,000 feet. Jet engine windmilling occurs when an engine is not operating, yet the compressor blades rotate due to oncoming airflow[1]. Although windmilling can be used as an emergency restarting mechanism on a commercial jet engine, it is detrimental to the JetCat P100-RX because the compressor blades are only lubricated by the circulation of fuel. When the engine is turned off, there is no fuel circulating to lubricate the compressor blades, hence leading to permanent engine damage if the blades were to spin. Windmilling can also cause unreliability in starting cold jet engines. Thrust vectoring allows for attitude and angular velocity modifications through directional firing, greatly increasing maneuverability. Improvements of performance through thrust vectoring technology has been demonstrated by the Air Force in the past through a number of aircraft, including the X-31. A key advantage for aircraft with thrust vectoring mechanisms is the ability to control the aircraft at steep angles of attack, when airflow is separated from the control surfaces. The X-31 was able to obtain controlled flight at an angle of attack of 70°[15]. According to NASA, the X-31 was also able to perform a 180° maneuver in post-stall flight, which is well beyond the capability of most aircraft[15]. Modifying and understanding these smaller micro jet engines and their new capabilities is the first step towards improving larger-scale engines. Solutions implemented on the micro jet engine can be applied to more complex systems. In order to accomplish this purpose, WHiMPS will employ a single mechanism software control sequence, a main control board, a graphical user interface (GUI), and mechanical hardware. The Air Force Research Laboratory (AFRL) team from the previous year, SPECS, was able to successfully redesign the ECU of the engine to take full control of its functionality. The SPECS project relied heavily on controlling the engine through their manufactured ECM; however, windmill prevention and thrust vectoring are external components and should not impede the engine's function. An image of the engine with defined body axes can be seen in Fig. 1 below.

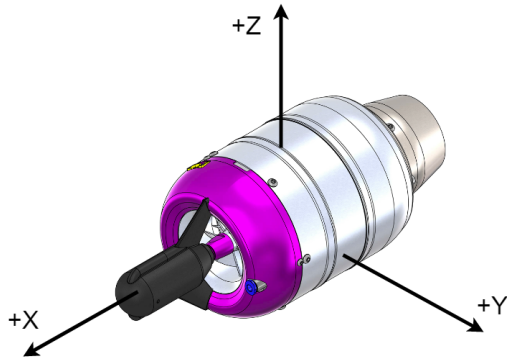


Figure 1: JetCat P100-RX Body Axes

2. Project Objectives and Functional Requirements

Authors: Alec Bosshart, Nick Zellmann

2.1. Levels of Success

The levels of success for the WHiMPS were formed from the requirements set forth by the Air Force Research Laboratory (AFRL) and the WHiMPS' proposed CONOPS. The five levels of success for the Windmill Prevention Mechanism (WPM) and Thrust Vectoring Mechanism (TVM) are outlined in Table 3.

Table 3: WHiMPS Levels of Success

Level	Windmill Prevention	Thrust Vectoring
1	Verify that the control software (using a simulated windmill prevention mechanism) restricts turbine motion to less than 0.5 RPM in response to conditions corresponding to a freestream velocity of Mach 0.8 at 20k feet.	Conduct software analysis to ensure the thrust vectoring mechanism does not diminish straight-line (0°) thrust. Verify that the control software (using a simulated thrust vector control mechanism) is capable of changing the thrust vector angle by ±10° along both the aircraft body Y and Z axes.
2	Same as Level 1.	With the engine off, verify that the integrated control software and thrust vectoring control mechanism is capable of changing the thrust vector angle of the engine along the aircraft body Y axis within a ±10° range, while retaining the original thrust at 0°.
3	Verify that the integrated control software and windmill prevention mechanism restricts turbine motion to less than 0.5 RPM in response to conditions corresponding to a freestream velocity of Mach 0.8 at 20k feet, with the engine off.	With the engine off, verify that the integrated control software and thrust vectoring control mechanism is capable of changing the thrust vector angle by ±10° along both the aircraft body Y and Z axes, while retaining the original thrust at 0°.
4	Verify that the integrated control software, windmill prevention mechanism, and thrust vector control mechanism continues to meet the level three success criteria for both Windmill Prevention and Thrust Vectoring with the engine off.	
5	Meet level four success criteria with the functioning JetCat P100-RX.	

For this project to be successful, the two main requirements set by the customer (the AFRL) must be accomplished. The first requirement is to successfully implement a Windmill Prevention Mechanism (WPM) that prevents the engine's fan blades from rotating more than 0.5 RPM under a Mach 0.8 flow at an altitude of 20k feet. The RPM requirement was not clearly defined by the AFRL, thus the more specific requirement of having a RPM of less than 0.5 was set by the WHiMPS team. The functionality of the WPM must include the engagement and disengagement of the mechanism upon command from signals sent by the user. The second requirement set by the AFRL is to incorporate a Thrust Vectoring Mechanism (TVM) that can vector the thrust of the engine by ±10° on at least one axis (the body y or body z axis). This TVM cannot impede the nominal thrust of the engine when the thrust vector angle is set to zero degrees of deflection.

The WHiMPS team decided to set a level of success for achieving one axis of thrust vectoring and another for incorporating a second vectoring axis, in the body-Y and body-Z axis defined in Figure 1. Thus, our project could be considered successful if we incorporating only a single axis of thrust vectoring, but could reach a higher level of success by incorporating a second axis of vectoring. For the highest level of success, the WPM and TVM must be controlled by the same user interface. Additionally, for the highest level of success, the WPM and TVM mechanisms must be successfully incorporated on an operating engine. Since the engine's operation has been unpredictable in previous years' projects, all of the levels of success for the WHiMPS team except for the highest level revolved around the engine being inoperable. Thus, tests and simulations could be performed to ensure that the WPM and TVM could be operated independent of the engine to reduce the overall risk associated with the engine.

2.2. Proposed Mission CONOPs

The AFRL did not provide specific information as to what the developed WPM and TVM will be used for. Therefore, a proposed mission CONOPs was developed by the WHiMPS team to visualize how the developed mechanisms will be used. The WHiMPS' proposed mission CONOPs is broken into two distinct

sections; one for the WPM and the other for the TVM. In this proposed mission, a micro-jet engine is used as the propulsive system for a small unmanned autonomous vehicle (UAV). This UAV is carried to its mission location with a larger carrier aircraft. A summary of this proposed CONOPs can be seen in Fig. 2.

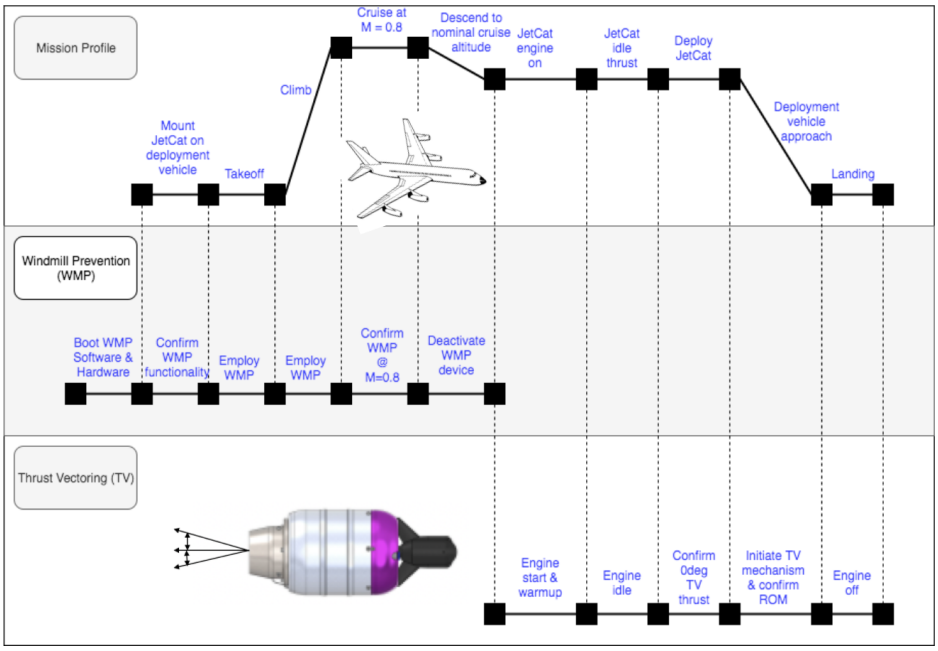


Figure 2: Proposed Mission CONOPs

The only mechanism used in the first half of the proposed mission is the WPM. At the beginning of the mission, the WPM is engaged on the engine of a UAV and is then carried by a larger carrier aircraft to the UAV's point at the carrier aircraft's cruising conditions (20k ft at Mach 0.8). During this half of the mission, the UAV's engine is off and the WPM is preventing windmilling of the engine's fan blades. Once the target location is reached, the WPM will disengage upon command from a user and the second half of the mission will commence.

The second half of the mission solely utilizes the TVM. After the WPM is disengaged from the engine, the JetCat P100-RX engine will start and reach an idle state, the deployment vehicle will detach from the larger carrier aircraft, and the TVM will be sent commands to control the attitude and proposed direction of the deployment vehicle.

2.3. Testing CONOPs

Another CONOPs was developed as a guide for how the WHiMPS team would simulate the mission environment. Because the developed mechanisms are not actually operating in the proposed mission environment, simulating the environment using the resources that CU offers is the best way to ensure that the incorporated mechanisms operate successfully. The testing CONOPs is shown in Fig. 3.

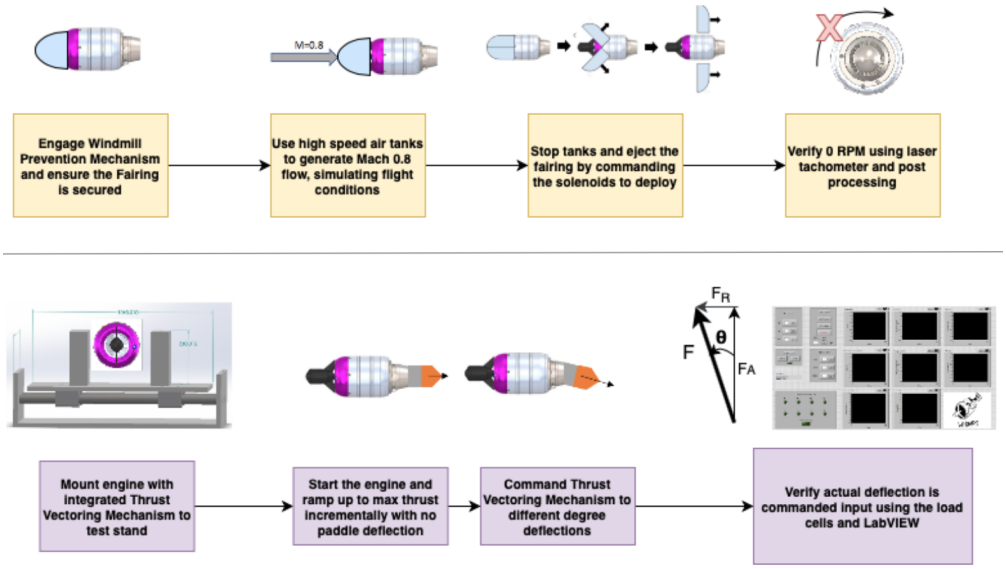


Figure 3: Testing CONOP

Since the two incorporated mechanisms operate under different conditions, two seperate CONOPS were developed. The top half of the testing CONOPS pertains to the WPM, while the bottom half pertains to the TVM. For the WPM testing, the WPM is secured to the front of the engine and a pressure tank system generates Mach 0.8 flow to simulate the mission environment. During the high-speed air test, laser tachometer and post-processing data are used to verify <0.5 RPM of the fan blades. Finally, the WPM is ejected to replicate how it would operate in its proposed operation environment.

For the TVM testing, the JetCat engine is to be mounted onto the WHiMPS developed test stand with the integrated TVM. The engine is then started and ramped-up to obtain maximum thrust with no commanded thrust vector angle. Once the engine is outputting its maximum thrust, commands are sent to the TVM via the user interface and the Main Control Board to vector the thrust to a specific angle of deflection. Load cell data from the test stand is then utilized to ensure that the commanded flow-deflection angle matches the actual flow-deflection angle. This testing procedure verifies that the developed TVM can vector the thrust such that the functional requirements are adequately met.

2.4. Project Deliverables

This project was completed for the AFRL under the supervision and guidance of the University of Colorado, Boulder. Each of these entities had requirements for documentation and deliverables to be submitted throughout the course of the project. The AFRL required monthly update documents containing specifics about the WHiMPS team's progress and upcoming activities, along with participation in the Aerospace Propulsion Outreach Program (APOP) demonstration. Additionally, the AFRL sent a lieutenant to visit the University of Colorado and see our progress. The University required documentation that followed aerospace industry standard for the development of technology. This included a Project Definition Document, Conceptual Design Document, Preliminary Design Review, Critical Design Review, and a Fall Final Report during the first half of this project. During the second half of the project, WHiMPS was required to deliver a Manufacturing Status Review, AIAA conference paper, Testing Readiness Review, Executive Summary, Final Oral Report, and Project Final Report.

2.5. Functional Block Diagram and Software Flow Diagram

The WHiMPS Functional Block Diagram (FBD) is found in Fig. 4 below. This diagram illustrates what and how the electronics used in this project are communicating. As shown in the FBD, the MCB is the brains of the project: relaying data to/from the incorporated mechanisms, regulating voltages for other electronics, and passing information to and from the LabVIEW GUI.

Additionally, the Software Flow Diagram shown in Fig. 5 was developed to illustrate the inputs and outputs from each main component of the project to convey how the system works as a whole.

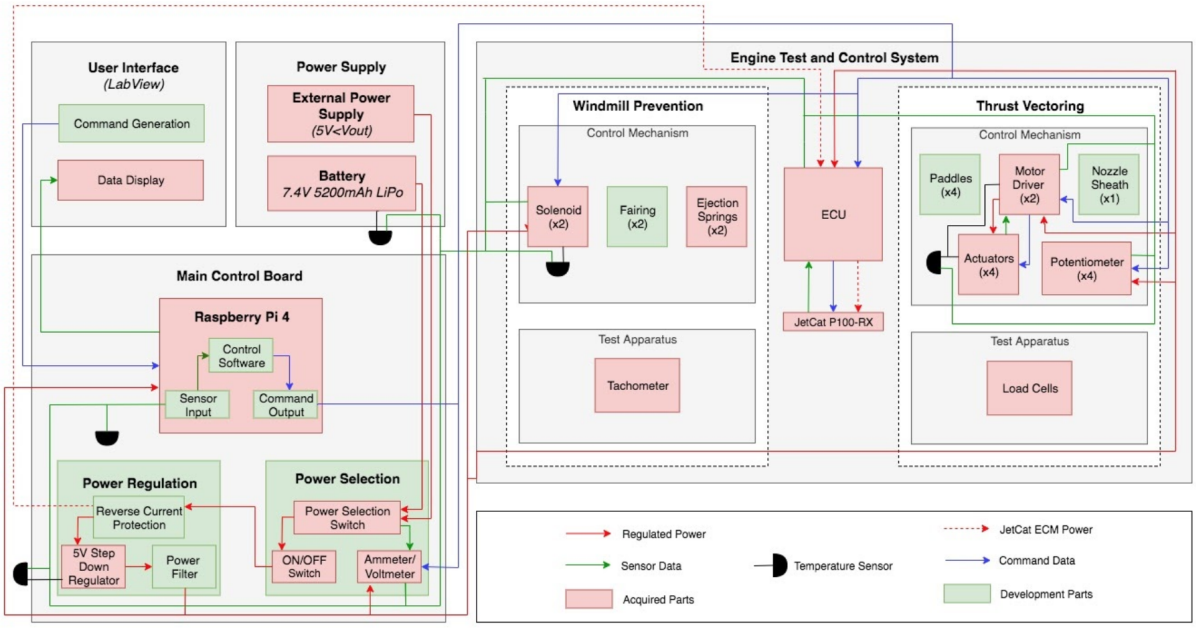


Figure 4: Functional Block Diagram

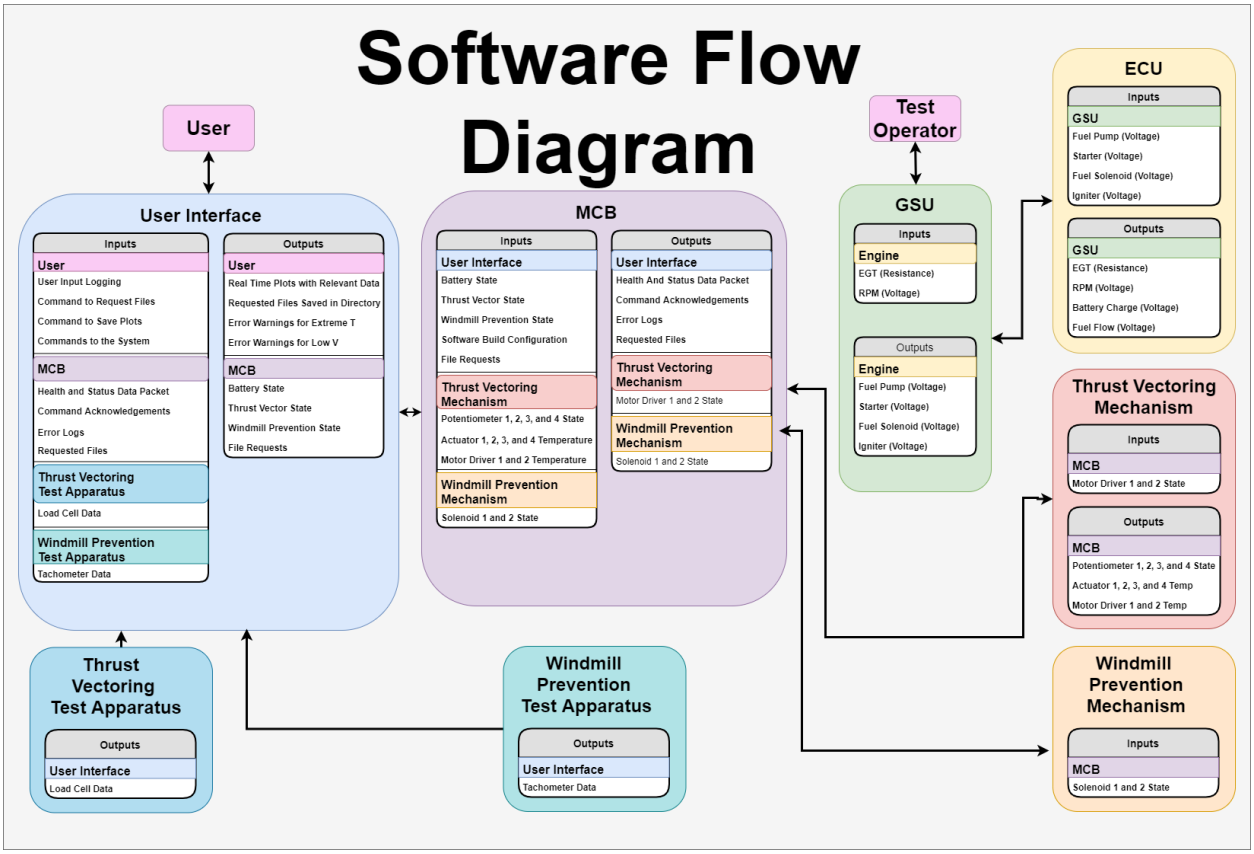


Figure 5: Software Flow Diagram

Beginning with the user interface on the left side of the software flow diagram, the user is given live data

from the MCB, the thrust vectoring test stand (TVM), and the laser tachometer (WPM). This involves data pertaining to the engine via the Jettronics software manufactured by JetCat (EGT, RPM, etc.), states of the motor drivers (TVM), potentiometers (TVM), actuators (TVM), load cell data (TVM), solenoids (WPM), and tachometer data (WPM). From here, the user can input data to command the solenoids to pull the pins pertaining to the WPM and control the vector angle of the TVM.

The right side of the diagram requires the involvement of the test operator. The test operator is in charge of the GSU, which starts and controls the engine during testing. Here, the RPM is inputted to control the engine's performance while data for the fuel flow, RPM, battery charge, and exhaust gas temperature (EGT) are relayed back to the test operator.

This diagram accurately shows where data is being sent from to control the operation of the WPM and TVM, along with how the WHiMPS were able to obtain data necessary for system validation. Additionally, it shows how the WHiMPS controlled the engine during testing in order to operate the engine safely and successfully.

2.6. Functional Requirements

The design of this project was greatly influenced by the five functional requirements and their respective design requirements. These functional requirements can be found listed below.

- **FR 1**: The JetCat P-100RX engine shall have an incorporated thrust vectoring mechanism that permits vectoring $\pm 10^\circ$ in both the body y and body z axes. The change in thrust at zero deflection must be zero.
- **FR 2**: The JetCat engine shall have an incorporated windmill prevention mechanism that restricts the rotational speed of the fan blades to less than 0.5 RPM.
- **FR 3**: The MCB shall control and monitor both of the incorporated mechanisms over the entire operational envelope.
- **FR 4**: The MCB shall run the engine such that it remains within the nominal operating conditions.
- **FR 5**: The MCB shall receive data from and send data to the user interface.

The first functional requirement pertains to the TVM. The AFRL specified that this mechanism must not impede nominal thrust (no change in thrust when vector angle is set to zero). The second functional requirement pertains to the WPM. This requirement ensures that the incorporated mechanism does not allow the engine's fan blades to rotate faster than 0.5 RPM under the specified conditions. Functional requirement three ensures that the MCB controls both the WPM and TVM. This was included in the functional requirements because of our customer's request to control both mechanisms with one single source; for this project, this single source is the MCB. Since the engine poses safety concerns during operation, functional requirement four was implemented to ensure that the engine is run safely. Finally, functional requirement five pertains to the user interface. This project requires the use of a user interface to permit users the ability to control the disengagement of the WPM, vectoring of the thrust to a specific angle, and observation of health and status data from the engine.

3. Design Process and Outcome

Authors: Declan Murray, Jon Weidner, Nick Zellmann, Alexandra Paquin, Lucas Zardini, Alec Bosshart

3.1. Requirements Flowdown

1. **FR 1**: The JetCat P-100RX engine shall have an incorporated thrust vectoring mechanism that permits vectoring $\pm 10^\circ$ in both the body y and body z axes. The change in thrust at zero deflection must be zero.
   Motivation: The task as given by the USAF is to incorporate a thrust vectoring mechanism to the pre-existing JetCat engine. The $\pm 10^\circ$ on both axes is an achievable figure also set in the requirements

by the customer.
Validation: To validate that the engine can vector the thrust by ten degrees on both axes, the WHiMPS team shall perform a test with the engine off to prove that the mechanism performs as expected.

(a) **DR 1.1**: The implemented thrust vectoring mechanism shall not affect the overall operation of the engine.
Motivation: The task for the project as outlined by the USAF is to design and implement a modification to the pre-existing JetCat engine that allows for the use of thrust vectoring in two axes ($\pm 10^{\circ}$). The installed modification shall not fundamentally change the operation of the engine.
Validation: The functionality of the engine shall not change; therefore, the operation of the engine before and after the mechanism is attached shall be the same.

(b) **DR 1.2** The change in thrust at zero deflection must be zero.
Motivation: The USAF states that the incorporated thrust vectoring mechanism must not impede the original thrust of the engine at zero deflection.
Validation: Before the installation of the thrust vectoring mechanism, a number of static tests will be performed at a specified RPM to obtain an average value of the thrust output of the engine at the specified RPM. Once this is performed, the thrust vectoring mechanism shall be attached to the engine. The same number of specified tests will be performed with the mechanism attached to obtain the thrust output with the mechanism attached. An allowable margin of error from day-to-day operation of the engine will be set in place to verify if the mechanism hinders the thrust output.

i. **DR 1.2.1**: The thrust output of the engine shall be measured by the thrust vectoring test apparatus.
Motivation: Thrust measurements must be taken to ensure the requirement set by the USAF is fulfilled.
Validation: The test apparatus shall be built such that the engine can be at full thrust while the apparatus is measuring the thrust output. A torque test will be performed to ensure that the apparatus can withstand the forces expected from the engine.

(c) **DR 1.3**: The thrust vectoring test apparatus shall measure the direction of the thrust vector.
Motivation: To achieve the requirements set by the USAF, the thrust vectoring mechanism must deflect by $\pm 10^{\circ}$ on two axes.
Validation: The mechanism will vector by a given input angle, and return the angle to the user via the user interface. An alternative measurement will be performed to verify that the vector angle outputted by the mechanism is equal to that of another measurement technique.

2. **FR 2**: The JetCat engine shall have an incorporated windmill prevention mechanism that restricts the rotational speed of the fan blades to less than 0.5 RPM.
Motivation: The task as given by the USAF is to prevent the compression fans at the front of the engine from rotating due to a high-speed mach flow (while the engine is off). The 0.5 RPM figure is set to account for vibration effects attributed to the freestream flow impacting the engine and error in the sensors measuring RPM due to this expected vibration.
Validation: Upon installation of the windmill prevention mechanism, a differential pressure representative of the mach 0.8 flow at 20k feet will be applied to the compression fan blades to ensure the mechanism is performing properly.

(a) **DR 2.1**: The windmill prevention mechanism shall be engaged/disengaged via input commands from the user interface to the MCB.
Motivation: The task for the project as outlined by the USAF is to design and implement a modification to the pre-existing JetCat engine to prevent windmilling. This design must be able to

disengage and engage on command such that the engine can still operate when testing the thrust vectoring mechanism.
Validation: Upon installation of the windmill prevention mechanism to the JetCat, the windmill prevention mechanism capabilities shall be tested with zero applied torque to ensure that the device can engage and disengage entirely. This implies that there shall be no windmilling prevention when the mechanism is disengaged, and windmilling shall be prevented when the attachment is engaged.

(b) **DR 2.2**: The windmill prevention test apparatus shall measure the RPM of the turbine.
Motivation: The task as given by the USAF is to prevent the turbine (and, consequently, the compression fans) from rotating under a Mach 0.8 flow. Thus, the RPM measurement of the incorporated sensors shall be zero.
Verification: The incorporated sensors shall be tested when there is zero torque applied to the compression fans. Using this, the sensors shall be calibrated to ensure the error of the sensors are limited. This will provide an accurate reading of the RPM of the compression fans during the testing of the windmill prevention mechanism.

(c) **DR 2.3**: The windmill prevention mechanism shall not affect the overall operation of the engine.
Motivation: The task for the project as outlined by the USAF is to design and implement a modification to the pre-existing JetCat engine that prohibits the compression fan blades from rotating under a Mach 0.8 flow at 20k feet. The installed modification shall not fundamentally change the operation of the engine.
Validation: The functionality of the engine shall not change; therefore, the operation of the engine before and after the mechanism is attached shall be the same.

3. **FR 3**: The MCB shall control and monitor both of the incorporated mechanisms over the entire operational envelope.
Motivation: In order to measure the effectiveness of any modification made to the JetCat engine, the MCB must be able to monitor and control the operation of both mechanisms up to the JetCat's maximum thrust conditions.
Validation: The engine shall be commanded to different thrust settings in increments, up to its maximum thrust setting. This will all be monitored with real time and recorded data for stability at the commanded throttle setting via the MCB.

   (a) **DR 3.1**: The MCB shall control all power distributed to the thrust vectoring mechanism.
   Motivation: The thrust vectoring mechanism must receive the power necessary to power the related components. The MCB shall distribute this power to the necessary components.
   Validation: A simple command will be sent from the user interface to the thrust vectoring components. If the health and status updates sent from the MCB to the user interface verify that the command was accomplished, the power is being successfully distributed to the components.

   (b) **DR 3.2**: The MCB shall control all power distributed to the thrust vectoring test apparatus.
   Motivation: In order to measure exit temperature, generated thrust, and fuel flow rate, a variety of sensors must be utilized. The MCB will distribute power to these sensors.
   Validation: The test apparatus shall be assembled with the engine/thrust vectoring mechanism/thrust vectoring apparatus attached. The sensors will return data from the room-temperature and zero thrust scenario. This verifies that the power is being distributed to the sensors.

   (c) **DR 3.3**: The MCB shall control all power distributed to the windmill prevention mechanism.
   Motivation: The windmill prevention mechanism must receive the power necessary to power the related components. The MCB shall distribute this power to the necessary components.
   Validation: A simple command shall be sent from the user interface to the windmill prevention components. If the health and status updates sent from the MCB to the user interface verify that the command was accomplished, the power is being successfully distributed to the components.

(d) **DR 3.4**: The MCB shall control all power distributed to the windmill prevention test apparatus.
Motivation: In order to measure the engine's RPM, a variety of sensors must be utilized. The MCB will distribute power to these sensors.
Validation: The test apparatus shall be assembled with the engine/thrust vectoring mechanism/thrust vectoring apparatus attached. The sensors will return data from the room-temperature and zero thrust scenario. This verifies that the power is being distributed to the sensors.

(e) **DR 3.5**: The MCB shall send commands to and receive data from the thrust vectoring mechanism.
Motivation: The thrust vectoring mechanism must be controlled to vector in a user-specified direction. Additionally, the thrust vectoring mechanism must return data to the user interface to specify which angle it has reached, when it has finished its task, and when there are errors.
Validation: The thrust vectoring mechanism shall be commanded by the user to vector to a specific angle. As the mechanism is vectoring, data will be transferred from the mechanism to the user, verifying that the MCB and thrust vectoring mechanism are communicating properly.

(f) **DR 3.6**: The MCB shall send commands to and receive data from the thrust vectoring apparatus.
Motivation: The user interface must display exit temperature, fuel flow rate, and generated thrust of the engine while vectoring. The test apparatus will obtain these measurements and send them to the user interface. Additionally, the MCB shall control the engine while it is operating (and thus the exit temperature, fuel flow rate, and generated thrust). Therefore, the MCB and thrust vectoring apparatus must communicate.
Validation: A test run of the engine shall be performed without vectoring the thrust. This test will serve to prove that the MCB and thrust vectoring apparatus are communicating correctly.

(g) **DR 3.7**: The MCB shall send commands to and receive data from the windmill prevention mechanism.
Motivation: The windmill prevention mechanism must engage and disengage via user input.
Validation: A test shall be performed without starting the engine to illustrate that the windmill prevention mechanism is properly communicating with the MCB, and vice versa. Simple commands such as "engage" and "disengage" shall be sent from the MCB to the windmill prevention mechanism. This shall verify that the two are communicating correctly.

(h) **DR 3.8:** The MCB shall send commands to and receive data from the windmill prevention test apparatus.
Motivation: The user interface must display the RPM of the engine's compression fans. The test apparatus shall obtain these measurements and send them to the user interface.
Validation: A test shall be performed where the compression fans are rotated to a specified RPM. The test apparatus shall then return the measured RPM to the user interface for comparison with the known value.

4. **FR 4**: The MCB shall run the engine such that it remains within the nominal operating conditions.
Motivation: The jet fuel, high engine temperatures, and high pressures associated with running a jet engine necessitate safety-conscious operation. Safe operation of the engine will protect personnel from injury and reduce the risk of damage to the engine hardware and test facilities.
Validation: The software/electronics teams will follow faculty-approved testing procedures and adhere to all safety requirements approved by the faculty. The safety/testing lead will run the safety checklist, ensuring completion of checklist prior to engine start.

(a) **DR 4**: The MCB shall maintain operation below 152,000 RPM (upper limit when maximum thrust is 22.5 lbf) unless a new upper safety limit is determined from the engine characterization.
Motivation: The maximum thrust is achieved when the engine is at 152,000 RPM. Therefore, to maintain a safe testing environment, the engine shall not be operated above 152,000 RPM.

Validation: The sensors available in the windmill prevention test apparatus will provide RPM data in real time, which may then be compared to the upper safety limit.

5. **FR 5**: The MCB shall receive data from and send data to the user interface.
Motivation: The user must be able to start, stop, and throttle the engine from a distance in order to have functional control of the JetCat. Additionally, there must be a mode of operation to control the integrated thrust vectoring and windmill prevention mechanisms. Therefore, the MCB must have a user interface that encompasses the controls of all parts of the project.
Validation: The engine's reactions from user inputs may be compared to measurements taken via an external measuring method. This ensures that the expected result is the same as the actual result. The same process shall be performed when testing the performance of the integrated mechanisms.

   (a) **DR 5.1**: The MCB shall send health and status data to the user interface.
   Motivation: The user shall monitor system parameters in real time to monitor and verify system response over time.
   Validation: The user shall observe the user interface and check expected results with actual results when the engine is off. If the pre-discussed readings line up with reality (engine off indicates 0 fuel flow rate, room temperature, etc.), the engine will be turned on and these measurements will be checked with alternative testing methods to ensure the data is still correct.

      i. **DR 5.1.1**: The MCB shall send exhaust gas temperature (EGT) readings to the user interface.
      Motivation: The jet fuel, high engine temperatures, and high pressures associated with running a jet engine necessitate safety-conscious operation. Monitoring the exit temperature of the engine during operational testing improves safety and reduces risk during experimental procedures.
      Validation: The thrust vectoring test apparatus shall send temperature data to the MCB, which will then relay the data to the user interface. The temperature of the ambient air shall be measured by the test apparatus. Another temperature instrument shall be used to ensure that the values on the user interface are reasonable.

      ii. **DR 5.1.2**: The MCB shall send compression fan RPM data to the user interface.
      Motivation: High rotation rates of the engine's compression fans increases the possibility of engine failure and damage to materials. Maintaining the compression fan's RPM below the 152,000 RPM ensures a safe testing environment.
      Validation: A test shall be performed where the compression fans are rotated to a specified RPM. The test apparatus shall then return the measured RPM to the user interface for comparison with the known value.

      iii. **DR 5.1.3**: The MCB shall send data representing the rate of fuel entering the engine to the user interface.
      Motivation: When the emergency stop button is activated, there should be no fuel entering the engine. Therefore, the fuel flow rate must be monitored to ensure a safe testing environment.
      Validation: A specified amount of fuel shall be available for the engine to use while operating. The MCB shall return that fuel flow rate data over time. If the fuel flow rate versus time is plotted, the total fuel used is the area under the curve. This can be calculated to verify if the fuel flow rate outputted by the MCB is correct.

      iv. **DR 5.1.4**: The MCB shall send engine thrust output measurements to the user interface.
      Motivation: To complete the requirements given by the USAF, the change in thrust at zero deflection shall be zero (not accounting for error from day-to-day temperature and pressure changes). Measuring the thrust is necessary to determine if the incorporated mechanism

changes the thrust output.
Validation: The specifications for the JetCat P100-RX engine state that the thrust at 152k RPM generates approximately 22.5 lbf. A best-fit line can be generated to estimate the approximate thrust at a given RPM. This can then be compared to the data obtained by the MCB to verify the data.

(b) **DR 5.2**: The user interface shall have the ability to send commands to the MCB for initiating power start up and shut down sequences.
Motivation: The user must be able to safely start and stop the engine through stock sequences.
Validation: The start/stop sequences will be validated by comparison to set stock sequences for engine start/stop (Appendix A).

(c) **DR 5.3**: The user interface shall have an emergency stop function for the engine.
Motivation: In the event of uncontrolled or improper response, a large manual emergency stop button and software-based emergency stop commands will immediately cut all fuel and power supplied to the engine.
Validation: Pressing the emergency stop button immediately removes all power from the MCB and thus stops the fuel pump and shuts the fuel cutoff valve.

(d) **DR 5.4**: The user interface shall send commands to the MCB to control the engagement/disengagement of the windmill mechanism.
Motivation: The user must be able to send commands to the MCB which will send commands to the actuators that engage/disengage the windmill prevention mechanism.
Validation: The user shall send a simple command to engage the windmill mechanism and observe if the command is met. If so, the user will send another command to disengage the mechanism therefore no longer preventing windmilling from occurring.

(e) **DR 5.5**: The user interface shall send commands to the thrust vectoring mechanism.
Motivation: The requirements as given by the USAF include designing and integrating a thrust vectoring mechanism to the pre-existing JetCat engine. The thrust must be vectored $\pm 10^\circ$ in both the body y and z axes. The user must be able to send command inputs to the mechanism to control how much it vectors.
Validation: The thrust vectoring mechanism shall first be tested with the engine off. The mechanism shall vector $\pm 10^\circ$ on both axes before the incorporation of actual engine thrust.

i. **DR 5.5.1**: The user interface shall send commands to the MCB to vector the thrust along the body y axis to a specified absolute angle.
Motivation: The requirements as given by the USAF include designing and integrating a thrust vectoring mechanism to the pre-existing JetCat engine. The thrust must be vectored $\pm 10^\circ$ in both the body y and z axes. The user must be able to send command inputs to the mechanism to control how much it vectors.
Validation: The thrust vectoring mechanism shall first be tested with the engine off. The mechanism shall vector $\pm 10^\circ$ along the body y axis before the incorporation of actual engine thrust.

ii. **DR 5.5.2**: The user interface shall send commands to the MCB, to vector the thrust along the body z axis to a specified absolute angle.
Motivation: The requirements as given by the USAF include designing and integrating a thrust vectoring mechanism to the pre-existing JetCat engine. The thrust must be vectored $\pm 10^\circ$ in both the body y and z axes. The user must be able to send command inputs to the mechanism to control how much it vectors.
Validation: The thrust vectoring mechanism shall first be tested with the engine off. The mechanism shall vector $\pm 10^\circ$ along the body z axis before the incorporation of actual engine thrust.

(f) **DR 5.6**: The MCB shall send a success command acknowledgement to the user interface upon successful execution of a command.
Motivation: To understand if the MCB is encountering troubles implementing the input commands from the user interface, it is important to know when and where the problems are occurring.
Validation: At the end of a command sequence, the related section of code shall send an output statement to the user interface to notify the user when a command sequence has been completed.

(g) **DR 5.7**: The MCB shall send an error acknowledgement to the user interface upon failure of a command.
Motivation: To understand if the MCB is encountering troubles implementing the input commands from the user interface, it is important to know when and where the problems are occurring.
Validation: At the end of a command sequence, the related section of code shall send an output statement to the user interface to notify the user when a command sequence has failed. Once the MCB has realized that there is an error, the function will override the input command set by the user and stop all components from moving. This reduces the risk associated with moving any parts once an error has been detected.

3.2. Conceptual Designs

Before the detailed design process could begin, the WHiMPS needed to create conceptual designs for each subsystem. Once these designs were determined, a trade study process began to select a single design to develop further. The conceptual designs are briefly discussed in the upcoming sections, but further detail and discussion on them can be found in the Appendix.

3.2.1. Windmill Prevention Mechanism

The WHiMPS had four primary conceptual designs for the WPM subsystem. The first, an engine cover, attempts to prevent windmilling by eliminating the cause of windmilling: air entering the core of the engine. This design's primary advantage is that it does not physically interact with the engine's fan blades, which reduces the risk of the design. The next two preliminary designs, hooks and sealed notches, work off of the same principle. They are designed to prevent windmilling by physically sticking an object between blades that is held in place. These design theoretically will prevent all windmilling and are easy to deactivate, but are more likely to damage fan blades. The WHiMPS last design was a dog clutch, which is often used in the automotive industry. Its principle is to create a surface that fits on top of the engine fan blades, which locks them in place. Once again, this design can damage the engine's fan blades.

3.2.2. Thrust Vectoring Mechanism

WHiMPS had more conceptual designs for the thrust vectoring system, with six in total. These begin with jet vanes, which are four symmetrical airfoils that are plunged into the exhaust of the jet. The airfoils, or vanes, are rotated in place to create an effective angle of attack of the vane, which pulls the exhaust flow using the same principles of lift. Jet vanes were a promising design, but impede the nominal thrust of the jet at zero vectoring because they interact with the exhaust flow. The cruciform worked in a very similar way, with perpendicular vanes that are connected to the same shaft. Next was fluidic thrust vectoring. Fluidic thrust vectoring pulls streams of air from the compressor stage of the engine and allows them to flow along the side of the engine toward the exhaust. By controlling these low pressure streams of air, the exhaust flow can be ”pulled” in the desired direction. While this design does not mechanically interact with the rear of the engine, it does create a substantial amount of risk by physically altering the engine at the compressor stage. The final three designs, a flex-nozzle, external nozzle, and rudders, all use mechanical mechanisms to push the exhaust flow to the desired position. The flex-nozzle and external nozzle both have cylindrical component, like another nozzle, that is actuated to rotate and turn the exhaust flow. The paddles work off of a similar principle, but are four independent surfaces that can be actuated into the flow. These three

systems are more costly from a mass perspective, but allow vectoring without impeding nominal thrust at zero deflection.

3.2.3. MCB Microcontroller

For the microcontroller, WHiMPS examined and created an objective trade study with five different options: the Beaglebone Blue, the Beaglebone Black, the Raspberry Pi 4, the Raspberry Pi Zero WH, and the NVDIA Jetson TX2. The baseline requirements were that the microcontroller must support a Linux OS and that the microcontroller must support either Python or C++ for programming languages. These are required because unit testing and object oriented programming are going to be implemented, thus the languages and the OS must support both of these capabilities to ensure WHiMPS is successful.

3.3. Trade Study Process and Results

3.3.1. Trade Study Methodology

In order to perform an unbiased trade study, a Likert scale was developed to rank each of the options. A Likert scale ranks items from worst to best or 1 to 5, respectively, in various categories. Each category has its own specific weighting factor in order to ensure the more relevant categories pull more weight in the final average. The breakdown structures of each trade matrix were designed first in order to allow for unbiased and justified scoring. Each team member then assigned a score to each category for each trade study conducted and these scores were averaged. In addition, the weights were also assigned to each category and averaged for each trade study. Using the averages, the team then discussed each score to ensure the rationale matched the breakdown structure categories. Using the final scores and weights, the trade matrices were completed and the final solutions were evaluated and selected.

These trade studies were conducted for the three primary components of the system: the thrust-vectoring mechanism, the windmill prevention system, and the MCB/electronics system. An additional trade study was conducted for materials, and is available in the Appendix. The WHiMPS team felt that this trade study was useful in estimating weights and costs, but is not finalized to conduct a formal trade study. Given that the team did not yet have access to a Jetcat P100-Rx engine, exact thermal figures were not available to complete a trade study and select a material. This is particularly relevant to the thrust vectoring mechanism, where the mechanism will need to be able to handle high temperatures and stresses from the exhaust jet.

3.3.2. Windmill Prevention Mechanism and Thrust Vectoring Mechanism

The WHiMPS trade study matrices for the Windmill Prevention and Thrust Vectoring Mechanisms contain the same categories and are therefore grouped here. However, they have different weights for each of the aforementioned subsystems, as discussed in their trade matrix sections. Both systems used the same Liekert scale, outside of the performance section due to different performance goals of each subsystem. Each category's description and a discussion of the thought process that went into it can be found below:

Simplicity All of the design concepts presented have their own unique complications and challenges, as well as a need for certain expertise. This category takes into account two main factors. One consideration in the rating is based upon the amount of experience the team has with each option presented. WHiMPS is aware that a lot of these concepts will require the acquisition of new skills; however, establishing a baseline is important to accomplishing the task in a time efficient manner. In addition to team experience, this category also accounts for the complexity and feasibility of the project. Some of the options presented require more machining and software control than others, so these factors were considered.

Engine/Facility Safety One requirement of the chosen mechanism is allowing the engine to operate safely with the added mechanical parts and software. The trade studies presented offer both internal engine addition options and external engine addition options. Because an internal addition has the potential to hinder the flow or break off and damage the engine and facility, this was one factor considered more dangerous than others. Also, the number of moving parts was also considered as this poses the risk of interference.

Mass/Profile Increasing the mass and profile of the system introduces a potential increase in drag on the system. In addition, adding mass in general can cause an uneven distribution in the engine which would

result in low or no functionality. Using this criteria, the mass/profile scores were evaluated based on the increase in form factor and amount of material required as these both will result in a drag increase and thus loss in efficiency.

Reliability The options presented range from needing consistent upkeep and maintenance due to part failure or wear down to having an extensive lifetime with little to no assistance. The problems presented by the AFRL are real-life concerns and issues that engineers are trying to solve on larger jet engines. Because these solutions have a larger reach, it is important that they can be reliable and are able to survive a long time of extended usage. For reliability, the options were evaluated based upon if they are single use or multi-use, if they are likely to fail, and how often they would require maintenance. Reliability carried more weight in the thrust vectoring category than the windmill prevention category due to the overall increased difficulty in thrust vectoring over windmill prevention. It is predicted that the thrust vectoring mechanism will likely be more complex and need to withstand more harsh operating conditions in the jet-stream. So, the team desired a robust solution that would not require constant fixes to keep working.

Performance The performance of the windmill prevention mechanism is determined by the engine blades RPMs. The goal is a 0.5 RPM value of the fan when the engine is turned off, which allows for noise from the sensors and vibrations. The performance is also measured by the torque capacity that can be applied to the engine/mechanism combination, except in the case of the cover. The performance of WHiMPS TVM subsystem depends upon the ability to design a solution that has thrust vectoring capabilities greater than or equal to ten degrees with no decrement of thrust at zero degrees. The options presented range in the amount of thrust impedance they cause; internal components tend to impede the thrust more, thus this was taken in considering when ranking in the trade matrix. The trade study also took the possibility of two thrust vectoring axes into account, as this was desired (but not required) during the APOP demonstration.

Cost At the time of the trade study development, the WHiMPS discussed cost but ultimately decided to abandon the cost of each conceptual design from the trade study. This was done for two reasons. First, the team did not feel that they could accurately estimate the cost of each subsystem at the time of the trade study. Each design was conceptual, and therefore predicting precise amounts of material needed would be very subjective. Secondly, the team predicted that the relative cost difference between conceptual designs for each subsystem would be small. Cost is important for any project, but relatively small differences in cost should not be used to select one design over another.

WPM Trade Matrix These parameters were quantified on a Liekert scale as discussed in Section 3.3.1. A table with the descriptions of each liekert scale can be found in Fig. 94 in Appendix A. The results of the weighted trade study can be seen below in Figure 6. The engine cover design was selected for its overall simplicity and engine safety compared to other designs.

Weight	Category	Design Choice (Windmill Prevention)			
		Engine Cover	Hooks	Sealed Notches	Dog Clutch
0.21	Simplicity	4.2	3.8	3.1	1.7
0.19	Safety	4.6	2.3	3.4	2
0.13	Mass/Profile	3.4	3.8	3.2	3
0.17	Reliability	3.6	3.5	3.5	3.7
0.3	Performance	3.8	2.9	4	4.4
	Final Weighted Average	3.9	3.2	3.5	3.1

Figure 6: Trade Study for Windmill Prevention Mechanism

TVM Trade Matrix The same process was applied to the TVM Trade study. A table describing the Liekert scale used on the TVM conceptual designs, Fig. 95, can be found in Appendix A. The results of the trade study can be seen below, in Fig. 7. Ultimately, the paddle design (called rudders at this point in time) was selected for its simplicity, safety, and performance.

Weight	Category	Design Choice (Thrust Vectoring)					
		Jet Vane	Flex-Nozzle	Fluidic TV	External Nozzle	Cruciform	Rudders
0.24	Simplicity	4.5	1.6	1	3.8	4.2	4.2
0.14	Engine/Facility Safety	4.3	3	1.3	3.8	3.3	4.7
0.11	Mass/Profile	4.1	2.3	3.8	2.5	3.9	3.4
0.2	Reliability	4.1	2.2	2.5	4	4.3	4.67
0.31	Performance	3.6	4	3.1	4	3.4	4.9
	Final Weighted Average	4.1	2.7	2.3	3.8	3.8	4.5

Figure 7: Trade Study for Thrust Vectoring Mechanism

3.3.3. MCB Microcontroller

Number of Digital Analog I/O Ports For the purposes of the experiment, the MCB microcontroller will require a significant number of I/O ports to support the sensor and interfacing requirements of the WHiMPS project. The microcontroller needs to have a sufficient number of I/O ports to properly support the project.

Familiarity/Documentation This category is important because the more familiarity with the microcontroller, in addition to the availability for open source and community supported documentation increases the chances of success. Given that the team has received advice from a number of PAB members that simply starting the engine will be a significant task, the WHiMPS desire an electronics system that team members have past experience operating.

Number of Cores The WHiMPS wanted multiple cores as it gives us the option to develop a multi-threaded software program. This greatly increases the performance as it allows us to run multiple processes in parallel, such as sending data to the user interface and commanding the thrust vectoring mechanism at the same time.

Processing Speed The processing speed is fairly important because a higher speed allows for data to be acquired and commands to be executed more quickly. This will allow for faster response times for the thrust vectoring system as well as a quicker deployment of the windmill prevention mechanism.

Cost Cost was considered because AFRL wants a more cost effective solution to the proposed problem. It was not factored heavily, because most of the boards considered were similar in cost. Cost was included in the microcontroller trade study as this information is easy to identify.

Board Size Size was factored in because the microcontroller needs to be a part of the hardware that will be attached to a theoretical airframe, so the boards need to be relatively small to maximize their potential for use in a practical manner.

Serial Interface When looking at serial interface, we were hoping to verify that we had options for both SPI and I2C. This is because these are two of the more common serial communication protocols utilized for the types of sensors we will require.

User Serial Interface This was an important category because the WHiMPS desire the ability to receive data in real time as it is being collected, ideally with no cord. This will allow for better understanding of how the system is operating during tests in response to different operating conditions.

Power Power is important to examine because theoretically this system will be deployed on a UAV. To do so, the system will need to operate off of batteries or another power generation system on the UAV. Given that more power will lead to heavier batteries to operate, which has a significant impact on the flight performance of a small scale aircraft.

MCB Microcontroller Trade Matrix Like the previous trade studies, the categories of interest were quantified on a Liekert scale, which is shown in Fig. 96 in Appendix A. The results of the MCB trade study can be seen below, in Fig. 8. The Raspberry Pi 4 was selected, primarily for its serial interface and team familiarity with the board.

Weight	Category	Microcontrollers				
		Beaglebone Blue	Beaglebone Black	Raspberry Pi 4	Raspberry Pi Zero WH	NVIDIA Jetson TX2
0.05	# of Digital/Analog I/O Ports	3	5	4	4	5
0.3	Familiarity/Documentation	1	2	5	5	2
0.1	# of cores	2	2	4	1	4
0.1	Processing Speed	3	3	4	3	4
0.05	Cost	4	4	5	5	3
0.1	Board size	4	4	4	5	2
0.1	Serial interface	3	3	5	1	5
0.1	User Serial Interface	5	3	3	3	3
0.1	Power	1	3	3	3	2
	Final Weighted Average	2.45	2.85	4.25	3.55	3

Figure 8: Trade Study for the Microcontroller

3.4. Design Outcomes

3.4.1. Windmill Prevention Mechanism

Overview The WHiMPS team accomplishes its functional requirements for windmill prevention by using a two-piece fairing which will be mounted on the front of the P100-RX. These fairing pieces will be overlapping and will be held together by a pin system. The sides of the fairing will have tabs that will be attached to a forward retention ring which will also have mounted solenoids. These solenoids will serve to remove the pins in order to deploy the fairing away from the engine. An image of the integrated CAD model can be found below in Figure 9.

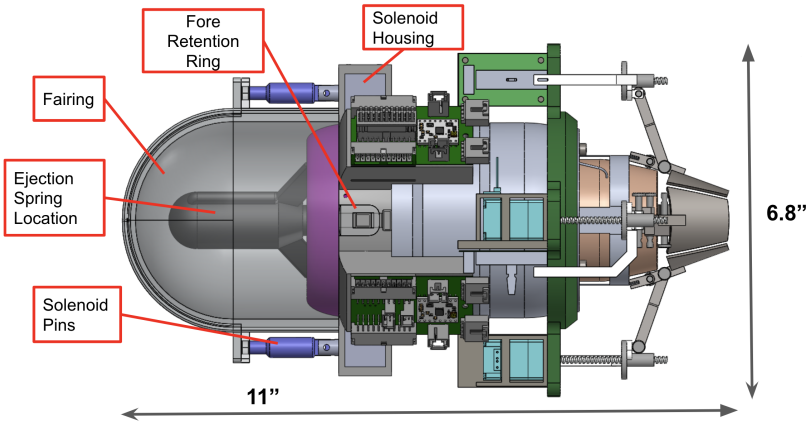


Figure 9: Integrated Windmill Prevention Mechanism

Fairing The WHiMPS fairing is a two piece component that has been manufactured using 3D printed carbon fiber reinforced Onyx from the Aerospace Building. The fairing will be attached to a forward retention ring using side tabs that are optimized to best hold the fairing in place, but will not impede the fairings ability to deploy. Each side of the fairing has a tab meant for the solenoid pin. These tabs are designed to line up when mounted, and allow the solenoid pin to be inserted before deployment. The retention tabs as well as the solenoid pin tabs can be seen clearly in Figure 10.

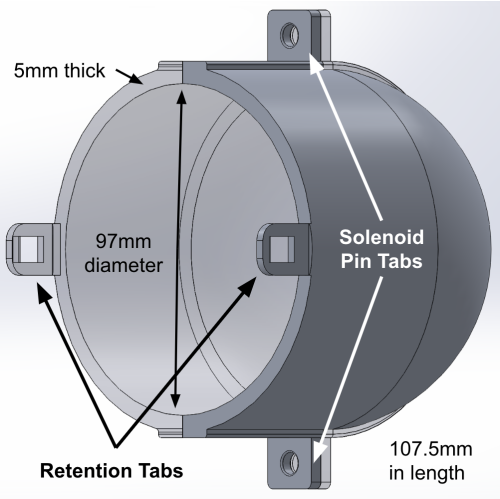


Figure 10: 3D Model of Fairing

In addition to the solenoid pin tabs, the fairing sides are designed so that they will have an overlapping seam. This serves to avoid separation of the fairing due to airflow at the front of the fairing during flight before deployment. An initial validation was completed using physical testing of a 3D printed preliminary fairing model. An anemometer, compressed air at 17 m/s, and a PVC pipe were used to simulate high speed airflow over the fairing. Since this test, the model design has been further developed to include a deeper seam as well as thicker side walls of the fairing. An o-ring can be added to the inside of the seam for redundancy if further testing reveals air leakage.

In order to validate the structural design of the fairing, WHiMPS met with Dr. Maute to discuss structural stability and how to validate the free body diagram of forces acting on the fairing. Due to the complex forces acting, Dr. Maute recommended we perform FEA using the designated pressure differential, and qualify the deformation of the fairing in order to confirm that the design would work. Using SolidWorks FEA, the WHiMPS were able to simulate the testing conditions of the AFRL demonstration in order to investigate the strength of fairing sides and whether or not they would significantly deform due to the pressure differential of 3.5 psi. The results seen showed minor deflection of the fairings, which validated the fairing designs, as the deformation would be in the order of thousandths of millimeters. These results can be seen in Figures 11 and 12.

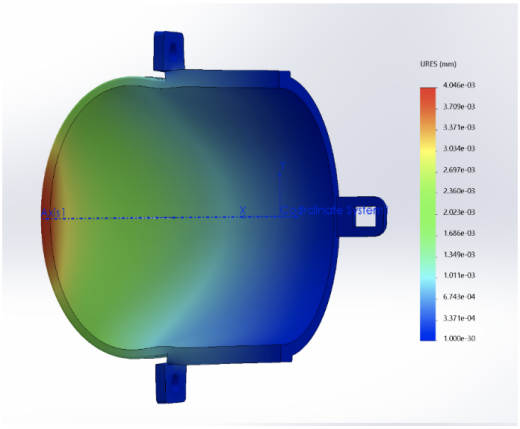


Figure 11: Structural Analysis of Left Half of Fairing

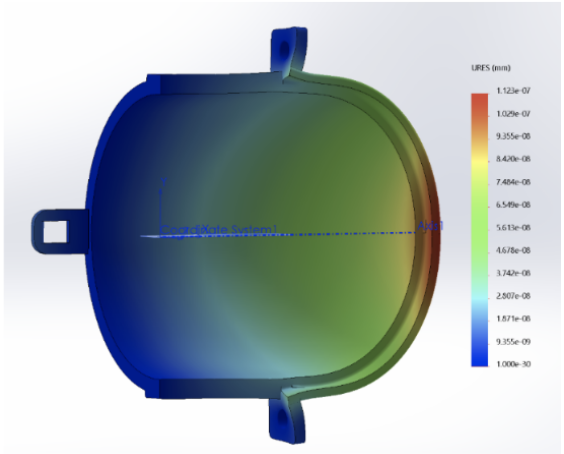


Figure 12: Structural Analysis of Right Half of Fairing

Ejection of Fairing While not required by the customer, WHiMPS incorporated an ejection mechanism for the fairing to improve the feasibility of the CONOPS. The fairing will experience three main forces,

as shown in Figure 13; note that the two sides are modeled as symmetric to simplify the analysis. The aerodynamic force is applied at the center of pressure of the system, the spring force is applied at the tabs at the base, and the drag force is vertically applied at the tip. The drag force experienced under Mach 0.8 flow at 20,000 feet is approximately 20 lb-f. The aerodynamic force will help hold the fairing together until the spring ejects; it will then aid in pulling the fairing pieces apart and into the freestream flow. The full free body diagram is shown in Fig. 13. While the center of pressure is unknown and calculating it is out of the scope of the class, these assumptions were validated by Professor John Mah.

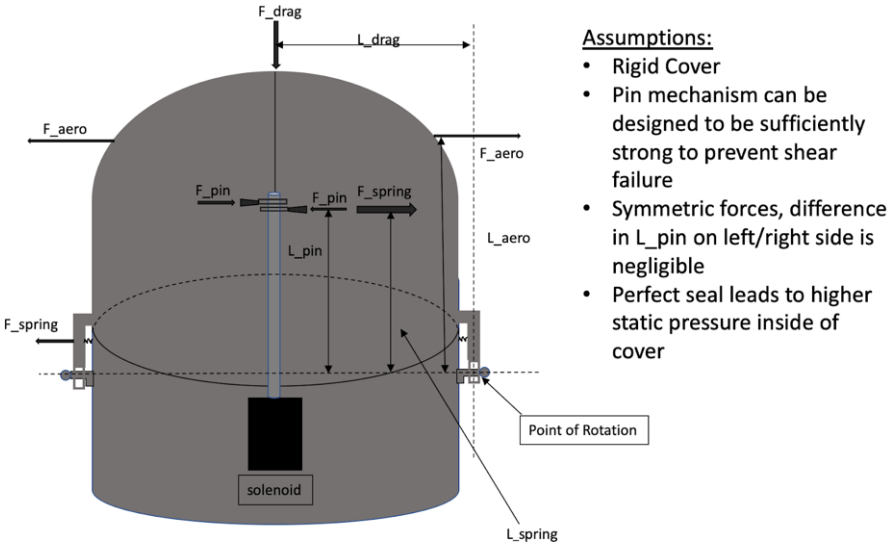


Figure 13: Force and Moments Induced on Fairing During Ejection

For this mechanism, the fairing will be held together with two pins to allow for pre-tensioning with springs. Figure 13 only includes one spring to simplify the diagram. Two solenoids will be used to pull the pins when instructed by the MCB. The springs will then eject their corresponding fairing side, with the help of the aerodynamic forces, and the engine will be ready to start. Figure 13 also provides the moment equation and assumptions used in the analysis. The specific springs and solenoids chosen will be further explained, but this equation proves that a positive moment will be generated onto the fairing, leading to an effective ejection. A model of the fairing ejection can be found below in Figure 14.

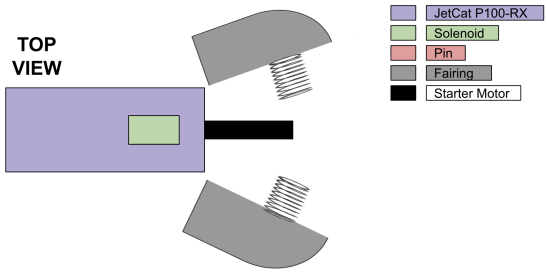


Figure 14: Fairing Ejection Model

Fore Retention Ring The fore retention ring allows for the fairing, solenoids, and the ICUP boards to be mounted to the engine. The ring is designed such that it can be attached to the front of the engine using pre-existing screw locations on the external of the engine. The ring will be latched onto by the fore retention ring,

and the solenoids and ICUP boards will be mounted directly. This ring will be manufactured using carbon fiber reinforced Onyx from the Aerospace Building. This will allow the fore retention ring to withstand the higher temperatures that the front of the engine endures during operation, and provide a thermal barrier between the engine and the mounted electronics. A summary of the derived conduction-convection thermal model of the amount of insulation provided to the electronics can be found below in Figure 15. The resulting surface temperature could be reduced to 59°C with a small amount of forced convection cooling, but can reach up to 125°C under natural convection.

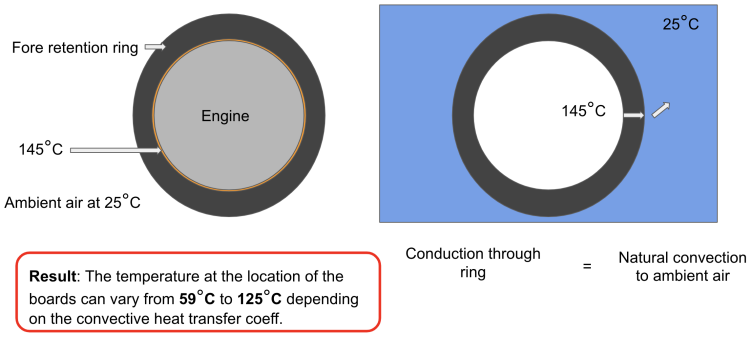


Figure 15: Fore Retention Ring/ICUP Thermal Model

Solenoids and Pins The selected solenoids, Fig. 16, were chosen due to their availability as COTS parts as well as the clevis type pin attachment they provide. This enables the ability to design a pin that can be compatible with both the solenoid and the solenoid pin tabs on the fairing. The selected solenoid runs on either 6VDC at 1 amp, or 5VDC at 0.8 amps, both of which the ICUP board will be able to provide to the solenoid. Use of these solenoids is favorable to the design because of the simplicity they allow for the rest of the windmill prevention system. The functionality of the solenoid-pin system can be found below in Fig. 17.

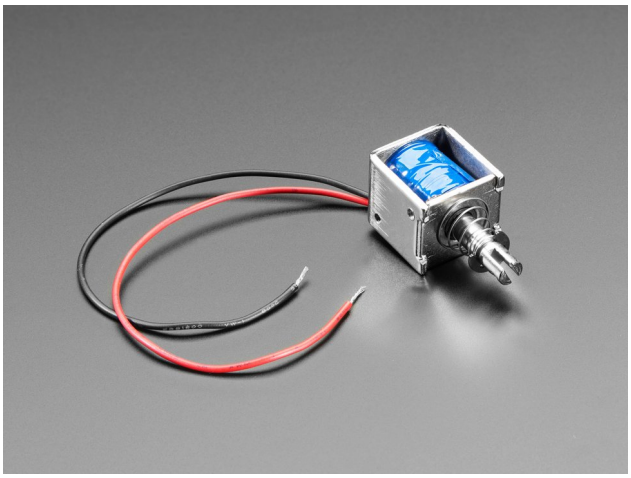


Figure 16: Adafruit Medium Push-Pull Solenoid

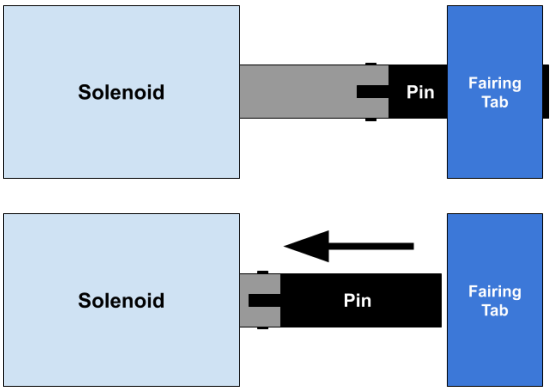


Figure 17: Solenoid-Pin Diagram

Spring The addition of a spring to the inside of the fairing sides was decided on to increase ejection redundancy. In the conditions that the fairing would perform, a speed of Mach 0.8 at 20,000 ft, the WHiMPS are confident that the aerodynamic forces acting on the sides of the fairing will pull the fairing apart and eject into free stream air flow, but the decision to add an additional ejection force that will ensure that the fairing is ejected properly. The current spring choice is a COTS conical compression spring from McMaster-Carr. The selected spring has a spring constant of 1.35 N/mm, and an extension length of 6 centimeters. An image of this compression spring can be found in the Appendix in Figure 97.

3.4.2. Thrust Vectoring Mechanism

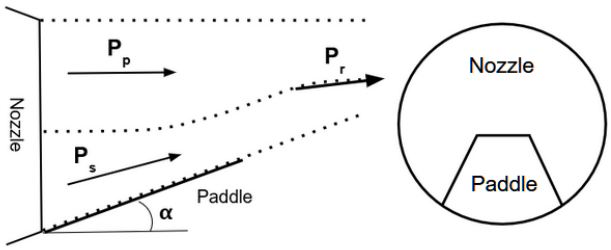
Core Concept Choice The trade studies converged on the paddle technology, with core advantages being simplicity and cost. The thrust vectoring mechanism (TVM) revolves around the paddles; the critical element. Fundamentally, the TVM was to meet design requirements of the paddles in order for the paddles to meet the functional requirements of the system. The design requirements were:

1. The TVM shall anchor the paddle near the exit of the nozzle, preventing translating of the paddle
2. The TVM shall use a controlled device to rotate the paddle about the anchor point
3. The TVM shall have a range of paddle rotation of $20° \leq -5°$, where $0°$ indicates the paddle plane is parallel to the engine central axis, and a positive angle indicates the tip of the paddle is rotated inward, toward the central axis.

These were defined to help inform the design process and establish a solution space. The paddles themselves meet the functional requirements, which is to reach $10°$ of thrust deflection in the two body axes. The rest of the TVM can be thought of as an additional two sub-assemblies: one complies with the first design requirement, the other complies with the second and third design requirements. First the critical element (paddle) will be explored, then the two sub-assemblies.

Paddles The paddles are the critical element of the TVM, as this is the component that is directly used to achieve the functional requirements defined by physically deflecting the exhaust flow of the engine.

Preliminary analysis modeled the paddle as a flat plate with finite area A_p. An analytical solution was derived to relate the angle of rotation (of the paddle) to the angle of deflection (of the exhaust) with a given paddle area and nozzle area. This model, shown in Figure 18, used conservation of momentum to determine a relationship between the paddle deflection and thrust vector. To get a rough estimate of the required surface area for a paddle, the equation can be implicitly solved for the area given a desired thrust vector angle and paddle deflection angle. When two paddles are used, which doubles the effective area in the model, the required paddle area is approximately 1,200 mm^2.



$$\theta = tan^{-1}\left(\frac{sin^2(\alpha)}{A_n/A_p - sin(\alpha) + sin(\alpha)cos(\alpha)}\right)$$

Figure 18: Analytical Model for TVM Paddles based on Conservation of Momentum

The design process then began with constructing a CAD model of the paddle. The paddles can be thought of as an extension of the nozzle, with a moderate bend at the joint so that the effective nozzle exit plane is at an angle to the engine central axis. A nozzle extension was explored in the trade study, but the manufacturing and control mechanism challenges were deemed too great; the paddles act like a nozzle extension but with gaps. To better mimic a nozzle, the paddles were curved about one axis, with the radius of curvature being just greater than the nozzle exit radius. The curvature helps each paddle conform to the cylindrical exhaust plume and "catch" more of the exhaust for deflection. Additionally, when the paddle is rotated into the exhaust, the curvature decreases the amount of exhaust that spills over the edges of the paddle, improving effectiveness. From the analytical model and intuition, an increase in paddle area leads to an increase in exhaust deflection angle for any given paddle rotation angle. Therefore, it is advantageous

to maximize the paddle surface area in contact with exhaust. This can be done by widening the paddle and by making it taller. The paddle cannot be made too tall, otherwise two paddles rotating inward will contact. The design uses four paddles, so each paddle cannot have an arc spanning more than 90°. A wide arc is desirable, as it increases surface area and also tightens gaps between paddles, reducing exhaust bleed. Because of this, the arc angle was designed to be 85°, giving a expansive presence while not introducing risk of paddle contact. The height of each paddle was then increased to meet the area requirements defined by the analytical model mentioned previously. At the required height, the corners of the paddles would contact each other, necessitating the change from a curved rectangle to a curved trapezoid. The trapezoid tapers to a reduced width as the paddle extends outward from the nozzle to prevent corner contacts. The paddle heights were adjusted to accommodate the reduced surface area of this change.

The design met the predicted area requirement of the analytical model, but the team became aware of the shortcomings and assumptions of the model:

1. It assumes a flat plate for aerodynamic simplification, with any bleed flow and edge effects ignored.
2. It only analyzes a paddle's individual performance, while two adjacent paddles should constructively interfere.

Thus, we were in need of a more sophisticated model to predict the aerodynamic performance. The commercial software package *Autodesk CFD* is a computational fluid dynamics tool that was used to make better predictions. The 3D geometry of the paddles and the nozzle were put into the program and boundary conditions were assigned at the nozzle exit to be consistent with information provided by the engine supplier. The results of the CFD model showed that the analytical model demonstrated significantly lower performance for the same paddle area and deflection angle. The CFD model is shown in Figure 19, and a comparison of the analytical and CFD models is shown in Figure 20.

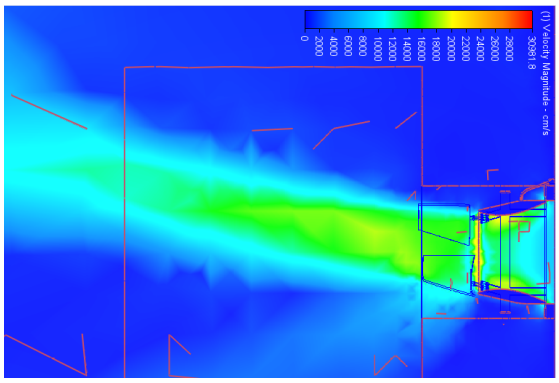


Figure 19: Autodesk CFD Model; paddle area $A = 850mm^2$, paddle angle $\alpha = 20°$, exhaust deflection angle $\theta = 11°$.

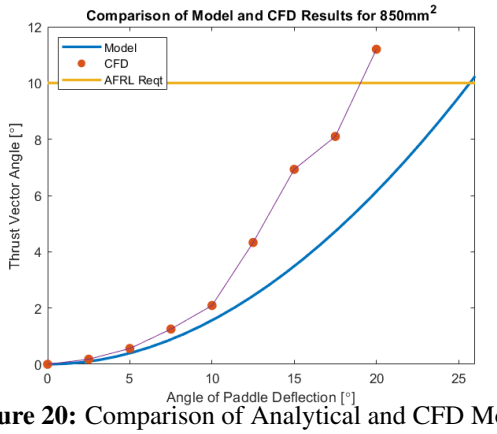


Figure 20: Comparison of Analytical and CFD Models

Iteration of the paddle height settled on an area of around 840 mm^2 to meet our functional requirements. WHiMPS had more confidence in the CFD model, as it made fewer assumptions in contrast with the greatly simplified analytical model. Therefore, the CFD model was used to inform the final design with an appropriate margin. With the finalized surface profile, the team ran a battery of CFD simulations to create a mapping of rotation angles to deflection angles. A continuous spline was fit to this data and used in the control algorithms. The splines and the linear dynamics equations are shown in Figure 21.

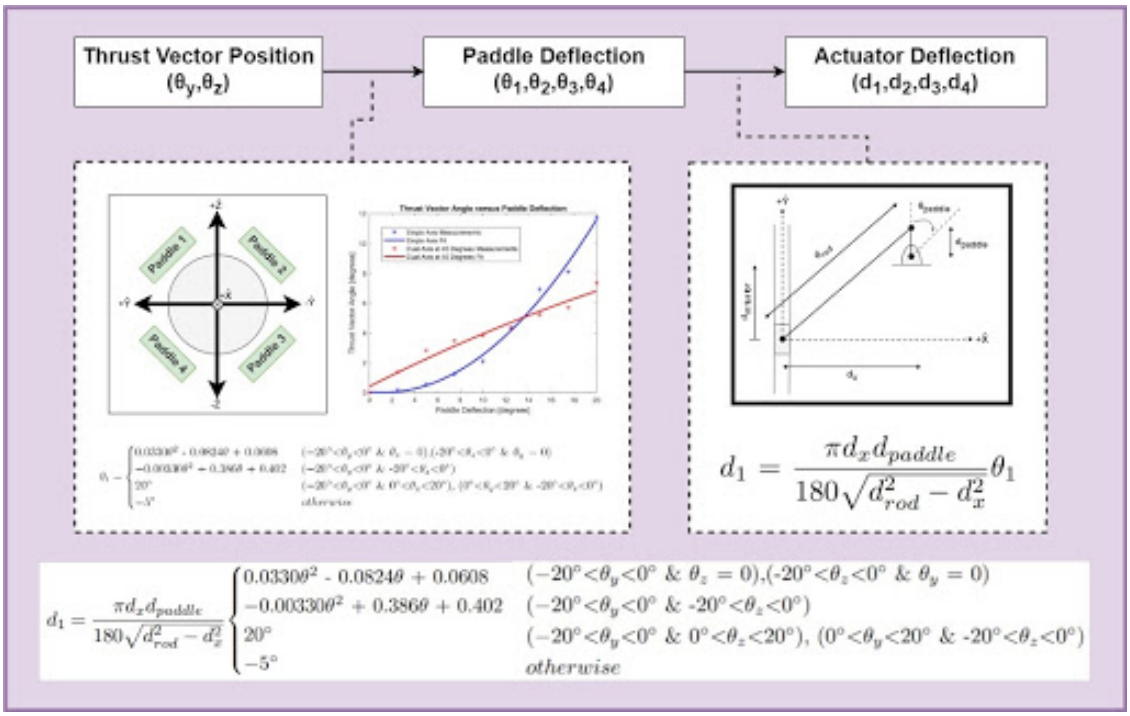


Figure 21: Paddle Deflection Calibration

As aerodynamics informed the surface geometry of the paddle, heat transfer informed the thickness and material of the paddle. A simple 1D thermodynamic model was developed in Matlab to determine thermal equilibrium of the paddle. The overall concept being $\dot{Q}_{in} = \dot{Q}_{out}$. This model made assumptions about the fluid properties and emissivity, as well as the turbulence transition point. A curve fit solution by Churchill and Ozoe (1973) provided a relationship for the convective heat transfer coefficient, appropriate for any fluid forced over an isothermal surface with any Prandtl number. This relationship is shown in Eqn. (1).

$$Nu = \frac{h_x x}{k} = \frac{0.3387 Pr^{1/3} Re_x^{1/2}}{[1 + (0.0468/Pr)^{2/3}]^{1/4}} \tag{1}$$

This formulation is claimed to be accurate to 1%, and its isothermal assumption is not extravagant here. Incorporating the convective effects on both sides of the paddle, the radiative heat loss, and the outward conduction, the model created a system of equations to be solved in order to achieve thermal equilibrium, namely, the surface temperatures on each side of the paddle. This model converges on a solution with minimal residual error solving the system of equations, with preliminary results predicting maximum temperatures around 500°C. This model is summarized in Fig. 22. This immediately narrowed options for the material, and a high strength stainless steel alloy was chosen to handle the high thermal flux without warping. First principle structural analysis established a minimum thickness bound based on the expected load cases, product availability lead the team to use 0.06” (1.5mm) thick sheets as the basis of the paddle structure. Using the 17-4PH stainless steel properties with 0.06” thickness yielded a max paddle temperature of 400°C. Using Solidworks Simulation, shown in Fig. 23, a thermal FEM model was developed and predicted extremely similar results with a maximum temperature of 400°C, though this model included the paddle neck structure. Even in an extreme case where the paddle surface reaches the exhaust temperature around 700°C, the steel chosen has not reached even its hardworking temperature, thus the paddles would not be in danger were they to get stuck at maximum rotation.

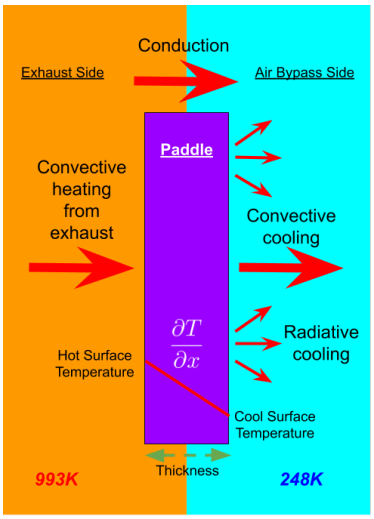


Figure 22: Thermodynamic Model of Paddle

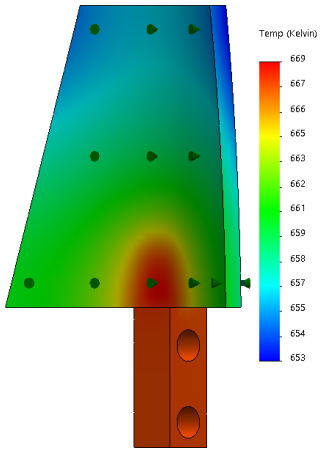


Figure 23: Thermal FEA Results of Paddle

The final design of the paddle consisted of the paddle face, which was just discussed in detail, and the paddle neck. The paddle neck is what transfers the anchoring and rotating functions of the two TVM sub-assemblies mention previously. The paddle neck does not contribute to the exhaust deflection. It is primarily a rectangular prism, with two through-holes to accommodate the anchoring and rotating functions, and a shelf which the paddle face sits upon. This shelf was implemented to assist the welding process that binds the two sub-components together, as it allows the paddle face to sit in position without specialized workholding tools. The dimensions were determined after iterative structural FEA, and the through-holes are sized larger than the pins for tolerancing. The paddle neck receives more heat than the pins and reach higher temperatures, therefore the blind holes thermally expand faster than the pins and introduce even greater tolerances. The bottom hole is pinned to the nozzle sheath, which is rigidly attached to the engine, therefore, the bottom hole is for anchoring the paddle. The top hole is pinned to the linkage rod and the rest of the actuator sub-assembly, therefore, the top hole is for rotating the paddle. The front and back views of the paddle are shown in Figs. 24 and 25, respectively.

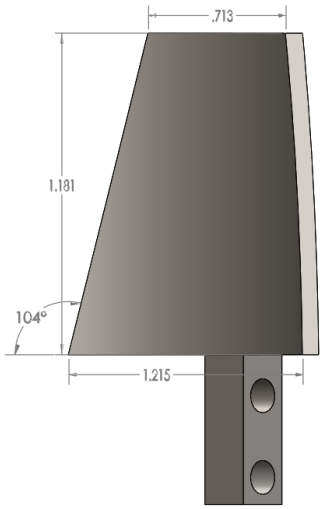


Figure 24: Front Side of Paddle

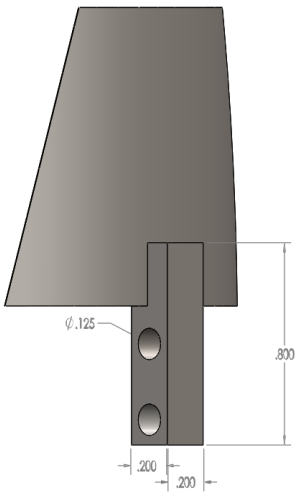


Figure 25: Back Side of Paddle

Nozzle Sheath The nozzle sheath, shown in 26, is the main component of the sub-assembly that complies with the first design requirement of the TVM. The nozzle sheath exists to secure and anchor the paddles around the rim of the nozzle exit. The nozzle sheath acts like a rigid wireframe that provides a mounting

point for the paddle to be pinned to. At the bottom, interfacing with the nozzle base, is an annulus. The base annulus has three holes placed where the engine normally mounts the nozzle. Thus the nozzle sheath is mounted to the engine using OEM bolts. The annulus inner diameter is slightly wider than the nozzle base diameter with tolerance for the vertical members (which are imprecisely bent into place and provide the most uncertainty in the manufacturing process.) The annulus width is about 1”, wider than the nozzle base width, to create an overlap for sandwiching the actuator ring (green).

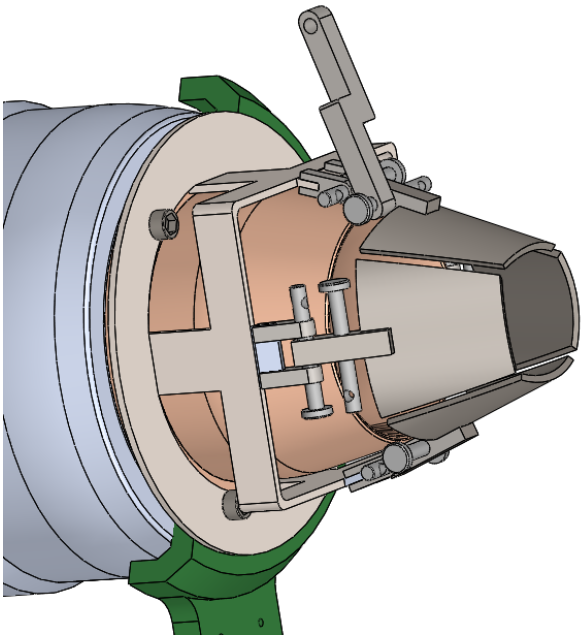


Figure 26: Nozzle Sheath Overview with Actuator Ring Sectioned

On the inner edge of the annulus are four rectangular members that extrude rearward, these connect to the upper square ring, and extrude farther past the ring but now at a 20° angle, these ends are called the sheath tabs. The square ring exists to provide rigidity to the structure, preventing one member from deflecting and spoiling the control system laws. The sheath tabs each have two small journal bearings on each flank. The bearings are separate components that provide the anchoring mechanism for the paddle. The bearings are welded onto the tabs and a steel clevis pin is inserted to join the bearings and the bottom paddle hole. The clevis pin has a cotter pin at the end to prevent sliding out. To simplify acquisitions, the nozzle sheath is also made of 17-4PH stainless steel, 0.06” thick. Both the paddles and the nozzle sheath were intended to be made from the same sheet. The nozzle sheath was subjected to a battery of structural FEA tests simulating every conceivable loading case while using material properties at 400°C, the worst case temperature. The structure passed every test case with significant margins of safety, the minimum being 0.6. These tests however ignored welds, which may degrade local strength.

Actuator Sub-assembly The Actuator sub-assembly consists of seven components that together comply with the second and third design requirements of the TVM.

1. Actuator Ring
2. Linear Actuator
3. Linear Actuator Pod
4. Linear Potentiometer
5. Actuator Nut (and attachment)
6. Linkage Rod
7. Linear Potentiometer Arm

These components exist to enable controlled rotation of the paddles. Together, they form a simple linkage system with closed-loop feedback to electronics.

The actuator ring is at first a 0.2” thick annulus wrapped around the nozzle base. The outer edge is profiled to closer resemble a ’plus’ with widths matching the size of the linear actuators. These four blocks have a central through hole, which the actuator’s threaded rod passes through, and four small through holes for the actuator to screw into. The whole component also has a 0.1” deep channel visible in Figure 27 below. This is to reduce mass and increase surface area for rejecting heat. This component was intended to be composed of aluminum, but due to limited thermal testing and time crunch, a copy made of steel was created instead that would be guaranteed to survive the heat flux in case it was worse than predicted to be transferred from the engine. The linear actuators, which attach to the actuator ring, are NEMA size 08 hybrid linear actuators by Anaheim Automation, and are shown above in Figure 28.

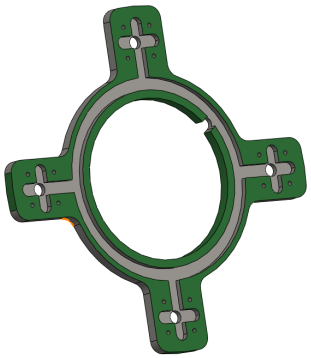


Figure 27: 3D Model of TVM Actuator Ring

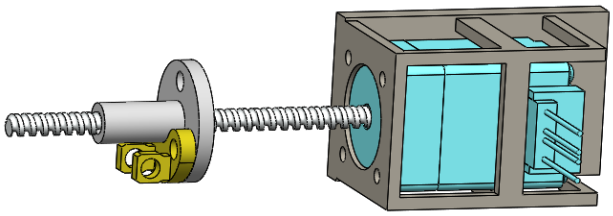


Figure 28: 3D Model of Linear Actuator in Actuator Pod with Actuator Nut at End of Threaded Rod

The linear actuators sit fore of the actuator ring and are attached via four M2 screws to the actuator ring. The threaded rod is spun by the internal motor, but as long as the actuator nut along the threaded rod is fixed in rotation, the threaded rod will not translate, only rotate in place. By fixing the actuator nut in rotation, the nut is forced to translate along the threaded rod. The white component is the OEM actuator nut while the yellow component is a 3D-printed custom addition, it is another journal bearing that is internally referred to as the actuator nut attachment. In this bearing goes a pin that attaches to the linkage rod. Surrounding the linear actuator (cyan) in gray is the actuator pod. This is also 3D-printed from plastic, and provides a light cage for the actuator. Its purpose is twofold: The solid face on the backside of the image provides a mounting location for the linear potentiometer breakout board. The material separating the actuator from the metal actuator ring provides vibrational damping and thermal insulation to protect the electronics. The wires extending out of the page are the electrical connection from the linear actuator to the Integrated Communication Unit Panel (ICUP).

The actuator nut attachment has a secondary purpose that is very beneficial to the assembly. There are substantial tolerance stackups from the several components needing to come together, especially since the manufacturing methods are less than professional. The actuator nut is 3D-printed, and its design intent is to be the remedial action for the assembly failing to properly align. Each of the four paddle-actuator connections would have its own custom actuator nut attachment. Multiple team members have personal 3D printers, making rapid prototyping of this small component very practical.

The linkage rod, shown in Fig. 29, converts the actuator’s linear motion into the paddle’s rotational motion, completing the second design requirement of the TVM. It is a steel bar in a jog configuration consisting of rectangular prisms. On each end is a 0.128” through hole to accommodate 0.125” cotter pins. The rod ends are rounded to prevent corner collisions. The jog is necessary because the threaded rod is in plane with the center of the paddle, thus the rod would have to be a jog or a fork. A fork is more difficult to manufacture, heavier, and busies the area around the paddle, which already has limited space for fastening. The end of the rod near the paddle may experience significant heating from bleed flow and potentially conduction from the pin, which is why this component is composed of stainless steel. The cross-section is

0.2”x0.2”, making one of these linkage rods mass under 10 grams. The final dimensions were informed by manufacturing feasibility and FEM analysis.

To truly satisfy the design and functional requirements, the system must have closed-loop active control. It must be able to determine its own state and command itself to a new state. To accomplish state determination, a linear potentiometer was added to the system. The linear potentiometer is effectively a variable resistance slider, sending electrical signals that indicate the position of the tab along its slide. The linear potentiometer is connected to its breakout board (a PCB) which is mounted onto the actuator pod. The linear potentiometer directly informs the controller of the position of the actuator nut. Via dynamical relations explored in Figure 21, the nut position determines the angular orientation of the paddle. The white member is referred to as the linear potentiometer arm and is rigidly joined to the actuator nut with a slot on the opposite end to fit the potentiometer tab. This component is 3D printed plastic. The full feedback system is shown in Fig. 30.

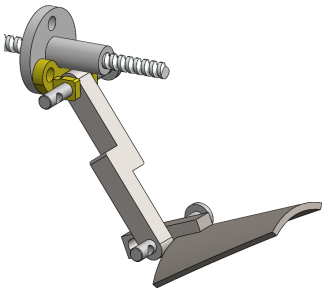


Figure 29: Linkage Rod 3D Model - Pinned Between Actuator Nut and Paddle

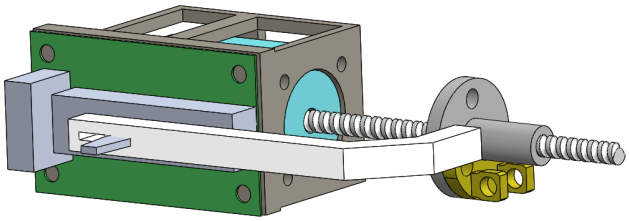


Figure 30: Closed-Loop Feedback Assembly 3D Model

TVM Assembly In concert, all these components meet the design requirements to anchor the paddle, rotate the paddle in a controlled manner, and provide paddle state determination while the paddle itself meet the functional requirements associated with deflecting the exhaust flow. The full assembly is shown in Fig. 31. To briefly summarize the TVM:

1. The nozzle sheath anchors the paddle in place near the nozzle exit
2. The actuator ring provides a sturdy mount for linear actuators.
3. The linear actuators receive commands to rotate the threaded rod.
4. The threaded rod rotates in place while the actuator nut is not allowed to rotate and translates along the threaded rod.
5. The actuator nut is pinned to the linkage rod, converting linear motion into rotational motion.
6. The linkage rod is also pinned to the paddle, making the paddle rotate as the linkage rod rotates according to a deterministic dynamical relationship.
7. The linear potentiometer senses the linear motion and indirectly informs the controller of the paddle state.

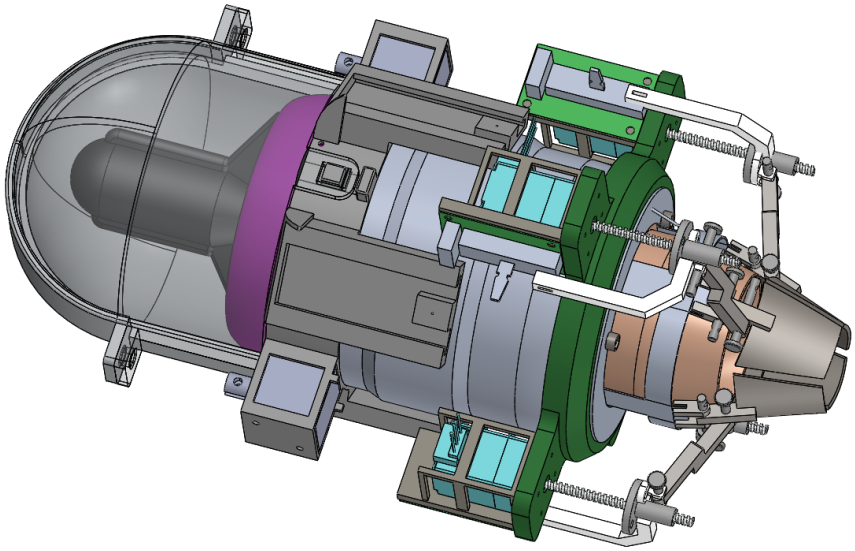


Figure 31: 3D Model Assembly of Entire Thrust Vectoring Mechanism.

3.5. Electronics

The electronics subsystem consists of providing and distributing regulated power and data to the CPU and critical hardware on WHiMPS. Due to the lack of space on the engine, the CPU and power regulation will be placed on a board separate to the hardware. In a mission context, these boards would be placed along with other avionics within the airframe. This is the driving factor to the current electronics design of having three separate PCB's: Main Control Board (MCB), Integrated Unit Communication Panel (ICUP), and Linear Potentiometer Breakout Board (LPBB). The following sections will describe these boards in depth.

3.5.1. Main Control Board

The Main Control Board is the test-bench PCB that is separate from the engine and hardware. This provides the computing power and power regulation that is distributed to the ICUP and LPBB. Specifically, the list below are the circuits that make up the MCB. The final updated design (Revision B) is shown in Figure 32, which is placed in its respective housing, shown in Figure 33. The dimensions of the board is 8.62in x 4.19in. In addition, all connectors are provided by MOLEX, other than the MCB to ICUP harness. This harness is provided by SAMTEC, driven by size requirements for the ICUP connectors.
Components on the MCB board:

1. Power Protection, Regulation, and Selection
2. CPU
3. Load Cell and Amplifier
4. Indicator LEDs
5. Power Check MOSFETS
6. Temperature Sensors
7. MCB <-> ICUP Harness

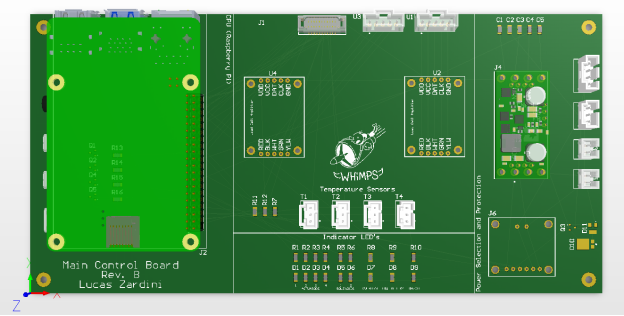


Figure 32: Main Control Board Rev. B Design

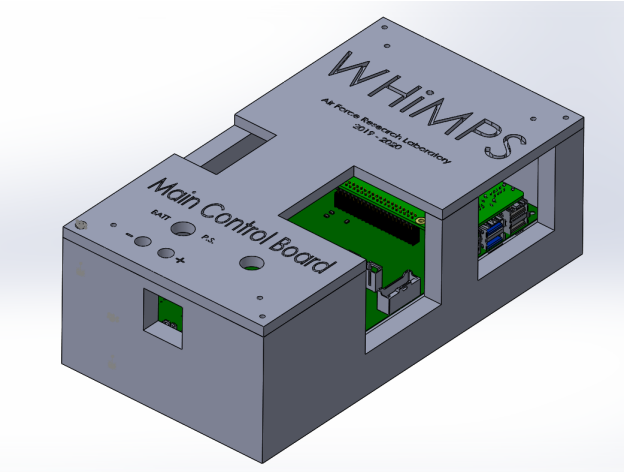


Figure 33: Main Control Board Case

Starting with the power circuitry, the Rev. B schematic for this section is shown in Figure 34. The start of the circuit starts with the power selection. There is a SPDT switch that will choose what power source the system takes in. This is to have the option for either external power or a battery source for testing or the demonstration. A majority of the electronics tests have been using the external power source for convenience, but the battery helps provide simplicity for the demonstration. If the battery is selected through the switch, the current goes through an ammeter before the power gets regulated. This is to monitor the state of the battery. External power does not require any monitoring. In addition, the current then flows through an ON/OFF switch (a SPST). This is for emergencies, in case the power needs to be shut off immediately.

The power is protected through a PMOS FET in the Power Path, as well as a zener diode from voltage to ground. The PMOS FET circuit will prevent the current from being reversed, and the zener diode protects the circuit from overvoltage.

The final part is power regulation. The rest of the PCBs require maximum of 5V, so an efficient step down converter is used to bring the battery/external power voltage (7.6V) to 5V. The current after the converter is now protected, regulated, and properly selected for the rest of the system.

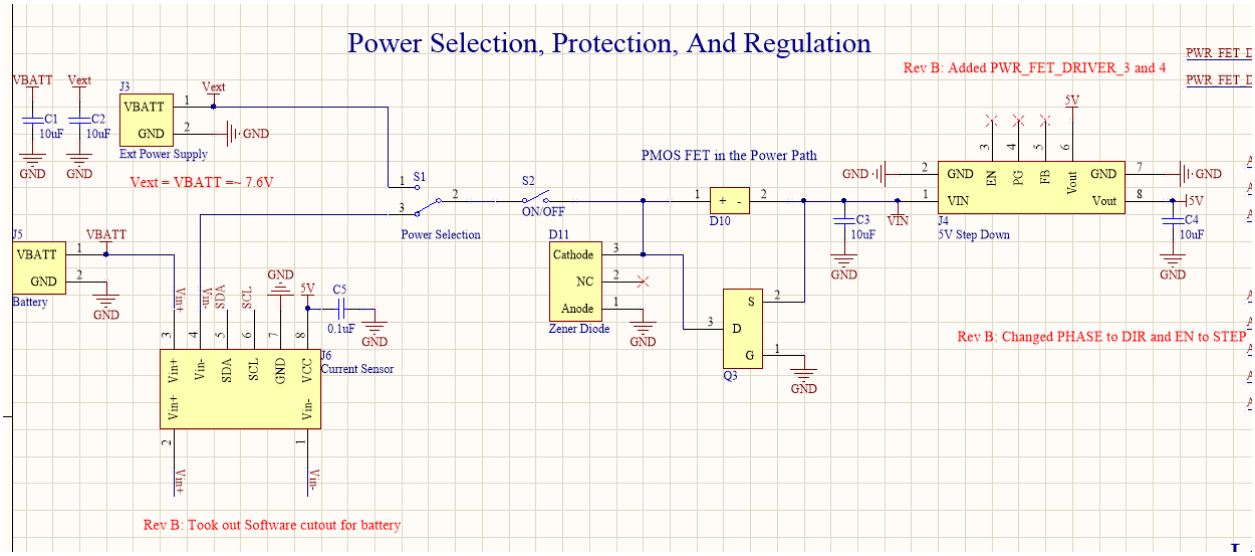


Figure 34: MCB Power Protection, Regulation, and Selection Schematic

The main processing power comes from the CPU, which is the Raspberry Pi 4. The pinouts of all data

lines connected to the CPU can be found in the Appendix, in Figure 99).There are 18 GPIOs, 1 one-wire, and 1 I2C ports used from the Raspberry Pi 4 to control all of the devices on WHiMPS. I2C port has one device (ammeter), one-wire has 12 temperature sensors, and the 18 I/O ports are split between the following: 2 solenoid, 4 power FETS, 4 load cells, 8 actuator, and 2 LEDs (OVERHEAT and Low Battery).

For testing purposes, load cells are needed for determining the thrust vectoring. This was an addition to Rev. B, and there are two placed on the MCB for two axis. To amplify the analog signal directly coming to the MCB from the load sensors, there are proper amplifiers for both cells. This is shown in the Appendix, in Figure (100).

It is also critical to know the state at which the hardware and system is at during testing, therefore indicator LEDs are placed on the MCB. More specifically, there are nine LED circuits displayed and labeled in the Appendix, in Figure 101. Eight of these LEDs are controlled by the CPU, the last one is an ON/OFF LED controlled by the power regulation circuit. The colors of the LEDs distinguish the type of hardware being activated.

For emergency and low power reasons, there will be MOSFET circuits used to control the power going to each of the actuators motor drivers (4 totals). This is highlighted in Figure 35. This allows full software control for the actuators, just in case there is an error during testing/demonstration or to save power during windmilling prevention procedures.

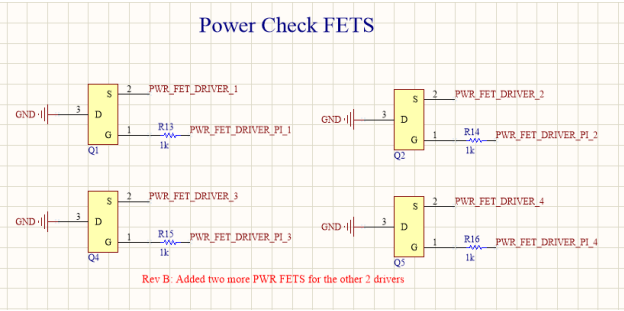


Figure 35: MCB FETS Schematic

The final two parts are the temperature sensors and the MCB < − > ICUP Harness, shown in Figure 36. There are four temperature slots on the MCB. Two are planned to monitor the battery and the step down converter. The two other ports are for backup or to be used for redundancy. The temperature sensors being used are one wire DS18b20 sensors and are consistent throughout the system. The MCB < − > ICUP Harness is a connector between the MCB and ICUP to transmit power and data to the engine components and PCBs. Also, on Figure 36, are the datalines that are being trasmitted to the ICUP.

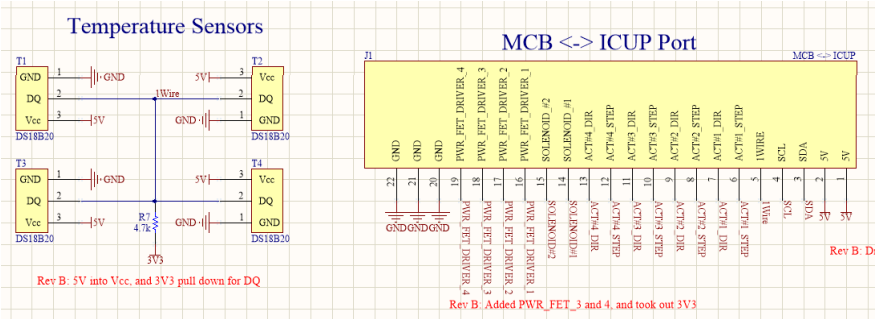


Figure 36: MCB Temperature Sensors and MCB < − > ICUP Harness

3.5.2. Integrated Communication Unit Panel

The Integrated Communication Unit Panel (ICUP) is an extension of the MCB to the engine by providing the necessary hardware power and data. As mentioned in the previous section, the power and data is sent

through the MCB < − > ICUP connector. The Rev. B design is shown below in Figure 37. This design is a four panel ring that surrounds the engine and is connected by SAMTEC connectors. Each panel has a different circuit that provides for the nearest hardware. The only identical part is the panel to panel connector. This provides connection between all the panels, and it helps provide redundancy for the power and data going to each panel. The four panels are connected to the fore retention ring with 3 4-40 screws.

The list of parts/circuits are listed below each panel's schematic. In the following section, the Linear Potentiometer Breakout Board shall be discussed. This is the last PCB design that extends from the ICUP to the actuator potentiometers. In addition, each ICUP panel has the dimension of 2.17in x 1.38in. The schematic of each ICUP board can be found in Appendix A, in Figures 102 - 105.

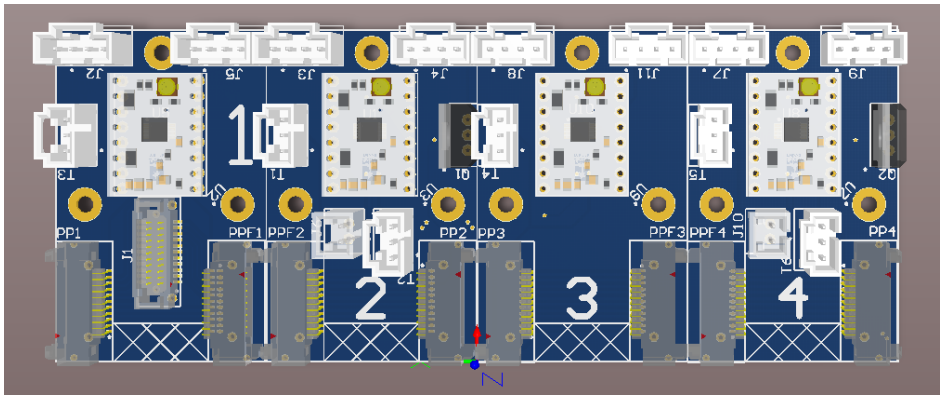


Figure 37: ICUP Revision B Design

The functions and connections of each ICUP panel can be seen in the lists below:

1. Panel 1
    - Panel to Panel Connector (2x)
    - ICUP< − >MCB Connector
    - Motor Driver and Actuator Circuit 1
    - LPBB 1
    - 5V to 2.5V Step Down Converter Circuit 1
    - Temperature Sensors (2x)
        - Motor Driver 1
        - Actuator 1
2. Panel 2
    - Panel to Panel Connector (2x)
    - Motor Driver and Actuator Circuit 2
    - LPBB 2
    - 5V to 2.5V Step Down Converter Circuit 2
    - Solenoid Circuit 1
    - Temperature Sensors (3x)
        - Motor Driver 2
        - Actuator 2
        - Solenoid 1
3. Panel 3
    - Panel to Panel Connector (2x)
    - Motor Driver and Actuator Circuit 3
    - LPBB 3
    - 5V to 2.5V Step Down Converter Circuit 3
    - Temperature Sensors (2x)
        - Motor Driver 3
        - Actuator 3
4. Panel 4
    - Panel to Panel Connector (2x)
    - Motor Driver and Actuator Circuit 4
    - LPBB 4
    - 5V to 2.5V Step Down Converter Circuit 4
    - Solenoid Circuit 2
    - Temperature Sensors (3x)
        - Motor Driver 4
        - Actuator 4
        - Solenoid 2

3.5.3. Linear Potentiometer Breakout Board

The LPBB converts potentiometer analog output to a digital signal. It contains the linear potentiometer and is connected to the Actuator Ring. There are 4 boards which connect to the ICUP and drive the thrust vectoring paddles. Each LPBB is identical to the design shown in Figure 38 (Rev. B). The circuit for each LPBB is shown in Figure 39. The dimensions for the LPBB design is 1.72in x 1.02in.

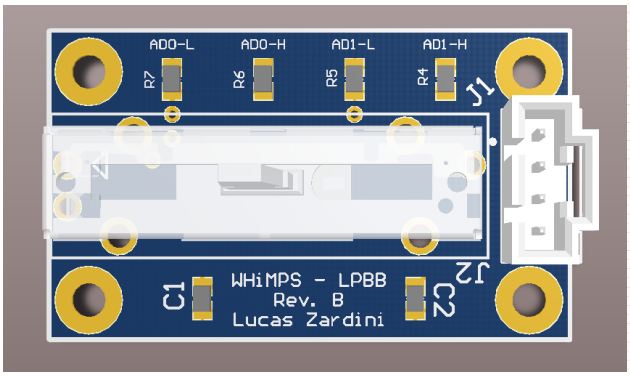


Figure 38: LPBB Rev. B Design

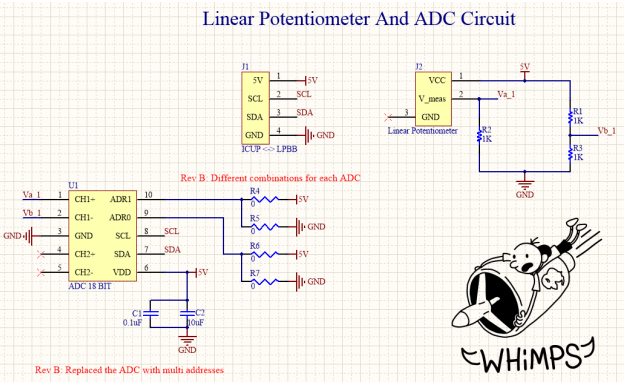


Figure 39: LPBB Circuit

The circuit for each LPBB includes an LPBB $< - >$ ICUP connector, ADC, and linear potentiometer. The linear potentiometer uses a Wheatstone Bridge circuit, used to help derive a differential output to the ADC for more accurate voltage readings. The theory behind this is to include three other identical resistors with the potentiometer and to place the voltage reading points between two identical resistors (fixed voltage) and the voltage potentiometer output. This helps keep a steady, low error voltage reading. For the ADC circuit, it takes in the differential voltages and converts it to a I2C digital signal. There must be four addresses for each LPBB, therefore, to keep the simple/identical design, zero ohm bridge resistors were placed to obtain these four addresses by placing ADDR0/ADDR1 to either high or low. These high or low resistors are tied to 5V or to GND respectively.

3.6. Main Control Board Software

The Main Control Board Software is the main software for the WHiMPS that is responsible for controlling and monitoring the Windmill Prevention Mechanism as well as the Thrust Vectoring Mechanism. It runs on the Main Control Board, or more specifically, the Raspberry Pi 4. The software was developed in the Eclipse Integrated Development Environment (Eclipse IDE) on a Personal Computer (PC) running the Ubuntu19 Operating System (OS). Additionally, the software was programmed in the C++11 programming language and unit tested with the Catch2 Unit Testing Framework.

3.6.1. Software Architecture

Figure 40 shows a high level view of the Main Control Board Software's Architecture. There are many elements to the architecture which are all ultimately utilized by at least one of the three threads running during program execution, the Health Status Thread, the Command Thread, and the Watchdog Thread. It is easiest to consider each of these elements as falling under one of the six top-level categories: Threads, Subsystems, Devices, Serial Drivers, Utilities, and Unit Testing.

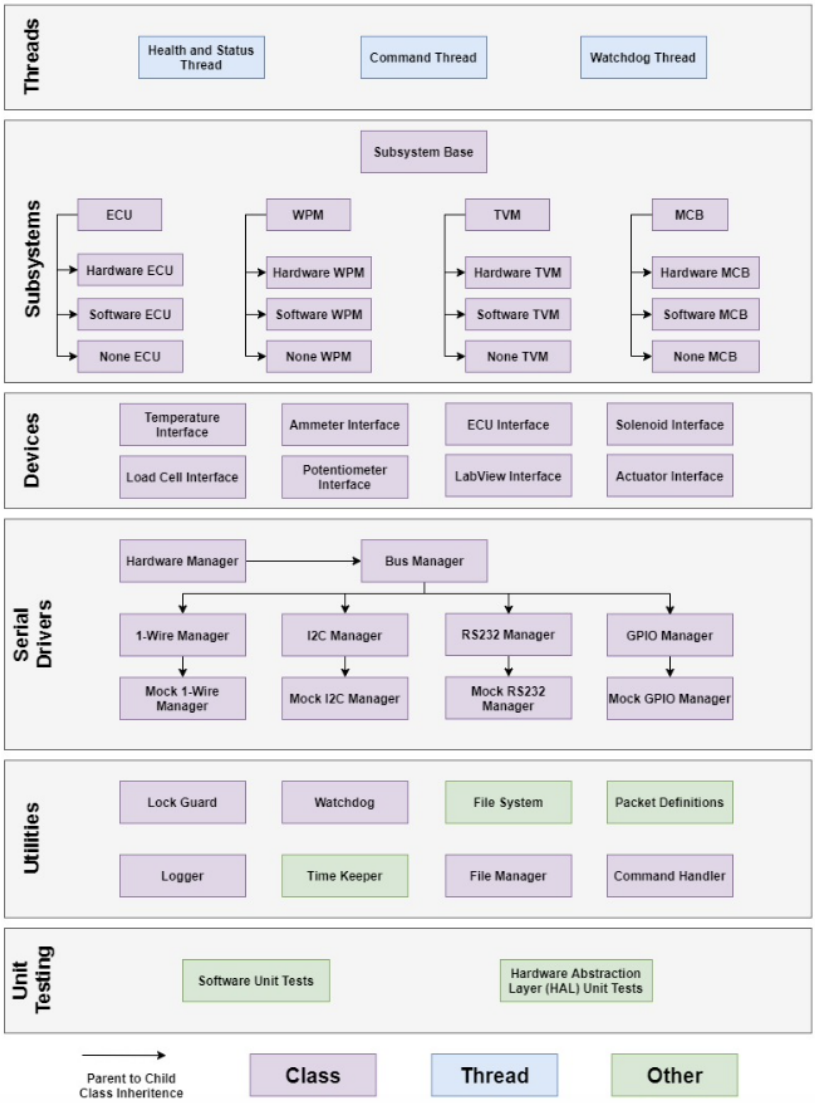


Figure 40: Main Control Board High Level Software Architecture Diagram

Threads

The first of the three threads is the Health and Status Thread. This thread manages the collection of the Health and Status Data Packet which is used by LabView to monitor the operations throughout the entire system. Every one second it collects data from each of the subsystems sensors and devices, packages that data into a predefined format, and send the data over RS232 to the LabView.

The next thread, the Command Thread, polls the RS232 receive buffer every one second to check for incoming Command Packets from LabView. Once a packet is received, it is sent through a validation routine to check the validity of the received command. If the command passes all of the validation checks, the command gets executed and a Command Acknowledgement Packet is sent back to the LabView that identifies the received command and an error code specifying if the command was executed successfully or not. Otherwise, the command does not get executed and a Command Acknowledgment is sent back to the LabView specifying the identified error for that command.

The final thread is the Watchdog Thread. The purpose of this thread is to start running as well as monitor the other two threads to make sure they are continuously running. The Health and Status and Command thread each run their processes in an infinite while loop, where on each loop iteration they ”kick the Watchdog”. By kicking the Watchdog each loop, these threads are telling the watchdog that they are

running nominally and not hung in any spot. The Watchdog Thread also runs in an infinite while loop, where in each loop the Watchdog first sleeps for 10 seconds and then checks that the other two threads have kicked the watchdog within that 10 second period. If a thread has not kicked the Watchdog within that time frame, then this means that the thread is hanging. For this scenario, the Watchdog will restart the thread so it can run again nominally.

Subsystems

The MCB software considers the Windmill Hindrance and Maneuverable Propulsion System as a breakdown of four main subsystems: the Engine Control Unit, the Windmill Prevention Mechanism, the Thrust Vectoring Mechanism, and the Main Control Board. The MCB software both controls and monitors three of these subsystems. Particularly the two temperature sensors and solenoids in the Windmill Prevention Mechanism, the six temperature sensors, two motor drivers, two load cells, and four linear potentiometers within the Thrust Vectoring Mechanism, as well as two temperature sensors, an ammeter, and power control circuitry for the motor drivers on the Main Control Board. The MCB software does not control the ECU, but it does monitor the system by collecting the systems engine turbine RPM, exhaust gas temperature, engine state, fuel flow rate, and battery charge.

As the MCB software was developed in the C++ programming language, many of the object oriented principles were taken advantage of when designing the architecture. One of the most effective uses of these principles is very apparent in the design of the subsystem class objects. The design first contains the highest level class, called a parent class, which is the Subsystem Base. As shown in the Appendix in Listing 1, the Subsystem Base outlines three methods for which all of its inherited classes must have: an initialization routine, a handle command routine, and a get Health and Status Routine. It does not provide definitions of these functions, but all classes that are of type Subsystem Base must provide their own implementations of these functions.

One level below the Subsystem Base is the ECU, WPM, TVM, and MCB classes. These classes are child classes to the Subsystem Base meaning they inherit from the Subsystem Base. By inheriting from the Subsystem Base, these classes can be recognized by two class types. For example, an WPM object can be attributed to both the WPM class and the Subsystem Base class. These second level classes each provide their own implementation for the handle command routines since each system has a unique set of commands that it can execute and all child classes of the second level classes use the same implementation. Additionally, each of these second level classes provide definitions but not implementation methods for each of the commands it can execute. Similarly, any class that inherits from one of these second level classes must provide their own implementation. An example of a second level class definition and implementation is shown in the Appendix in Listings 2 and 3 for the Windmill Prevention Mechanism. The handle command routine is implemented since all WPM child classes handle the commands in the same exact way whereas the eject fairing routine for executing the Eject Fairing Command is only defined since the implementation of executing this command varies with the WPM Class' child classes.

Each second level class is then a parent class to three child classes: a hardware, software, and none (or disconnected) implementation of the second level class. The hardware implementation of the class is instantiated when that subsystems hardware is physically integrated with the MCB, the software implementation is instantiated when the hardware is not physically integrated but it is desired that the MCB monitors and controls the subsystem with simulated hardware serial drivers (or mock serial drivers), and the none implementation is instantiated when the systems hardware is not physically integrated and simulation of that hardware is not necessary. When the objects are being built at the start of the program only one of the lowest level objects for each subsystem will be instantiated. Which lowest level object gets instantiated for each subsystem depends on the build configuration, which will be discussed later in this section. Additionally, each of the lowest level classes can be recognized by three class types. For example, a Software WPM Object is of type Subsystem Base, WPM, and Software WPM.

An example of the None and Software WPM classes are shown in the Appendix, in Listings 4-7. First shown are the header and source files for the None Windmill Prevention Mechanism Class. As shown in

the class constructor, NoneWPM::NoneWPM(), the states of the solenoids and temperature measurements are set to zero. Each time the health and status routine is called, the health and status for that subsystem is filled with zeros. Additionally, if an eject fairing command is received, the command is not executed as the subsystem is in a disconnected state and the associated error code is returned.

The Software WPM class, shown in the Appendix in Listings 6 and 7, provides a greater functionality than the None WPM class. When instantiating the class, a vector of the temperature interface objects and a vector solenoid interface objects are passed in as parameters. These interface objects are built using Mock Hardware Managers that allow the collection of data from these devices to be purely software simulated. Additionally, it will simulate the ejection of the fairing by changing the state of the solenoids. The changing of the state is not explicitly shown in these code blocks as this happens within the eject() function that is implemented else where.

The Hardware WPM code is almost identical to the Software WPM Class, with the exception that the vector of the temperature interface objects and the vector solenoid interface objects are built using non-mock hardware managers. Hence, the class will drive interactions with the physical hardware. In addition, there is more complexity with error handling and threshold checking.

Architecture Design Theory At first this design may seem extremely complicated and quite possibly unnecessary, but there is a good reason for this design as there are three major pay-offs in the long run.

The first pay-off is that as the Windmill Hindrance and Maneuverable Propulsion system is being developed and tested, hardware will become available incrementally rather than all at once. Hence this gives the team the flexibility to run the same program while accounting for which systems are integrated and which systems are not. The addition of the software subsystem classes allows the MCB software to be tested to a much greater degree than the none subsystem classes as it utilizes all of the same methods that the hardware classes use with the exception of using the software simulated hardware drivers.

The second major reason for designing the software in this way is so that each Level of Success can be tested using the same program but with different build configurations. Shown below in Table 4 is each of the build configurations that are utilized to test each level of success. The first level of success for the Windmill Prevention Mechanism corresponds to a purely software simulated system without the other systems integrated. Hence, for the WPM1 build configuration, the WPM is built with the Software WPM Class while the other three subsystems are built with the None classes. Note that for the purposes of the table, the LabView is categorizes as a subsystem, although its implementation in the design is different. The LabView can be built either in a software simulated mode or with actual hardware drivers and is not required to meet the first level of success for the WPM or the TVM, which is why there are a total of four build configurations for the first Level of Success. While there are a total of 9 build configurations listed in the table, in fact many more configurations can be created. If the build target for the program is a Linux PC, any of the None or Software classes can be utilized which gives a total of 32 build configurations for running the main program and an additional 1 build configuration for running the software unit testing build. If the build target for the program is the Raspberry Pi, any of the None, Software, or Hardware classes can be utilized which gives a total of 162 possible build configurations plus an additional two for the software unit testing and hardware unit testing. In total, this design allows the same exact program to be run in 197 different ways without having to change any code.

Table 4: MCB software build configurations corresponding to each level of success

Level of Success	Associated Subsystem	Build Configuration Mnemonic	Run Target	Subsystem Build Mode				
				Windmill Prevention Mechanism	Thrust Vectoring Mechanism	Main Control Board	Engine	LabView
1	Windmill Prevention Mechanism	WM1	Linux PC	Software	None	None	None	Software
	Windmill Prevention Mechanism	WM1_LV	Linux PC	Software	None	None	None	Hardware
	Thrust Vectoring Mechanism	TV1	Linux PC	None	Software	None	None	Software
	Thrust Vectoring Mechanism	TV1_LV	Linux PC	None	Software	None	None	Hardware
2	Thrust Vectoring Mechanism	TV2	Raspberry Pi	Hardware	None	None, Software, or Hardware	None	Hardware
3	Windmill Prevention Mechanism	WP3	Raspberry Pi	Hardware	None	None, Software, or Hardware	None	Hardware
	Thrust Vectoring Mechanism	TV3	Raspberry Pi	None	Hardware	None, Software, or Hardware	None	Hardware
4	Windmill Prevention and Thrust Vectoring Mechanisms	DITL	Raspberry Pi	Hardware	Hardware	None, Software, or Hardware	None	Hardware
5	Windmill Prevention Mechanism, Thrust Vectoring Mechanism, and Engine	DITL	Raspberry Pi	Hardware	Hardware	None, Software, or Hardware	Hardware	Hardware

The final pay-off can be seen in the implementation of the software. As described above, the subsystem classes are organized into a three-level hierarchy where the the top level class is the parent to the four middle level classes and each of the four middle level classes is a parent to three bottom level classes. Only one of the three bottom level classes is instantiated for each subsystem. Since each of the four instantiated lowest level classes are all of type Subsystem Base, this provides the opportunity for these classes to be utilized by the Health and Status Thread as well as the Command Thread with a great deal of abstraction. The concept of abstraction refers to the hiding of information, where the caller only knows the minimum amount of information required when calling the abstraction routine. Recall the three functions within the Subsystem Base: initialize, get health and status, and handle command. These are all abstraction methods because the caller does not need to know what those functions are doing internally. It just needs to know when to call them. For example, the Health and Status Thread knows that it needs to collect health and status data from each of the subsystems every one second. But, it does not need to know whether or not that subsystem is disconnected, software simulated, or hardware integrated. Since all of the instantiated classes provide their own implementations for get health and status, the Health and Status Thread just needs to call that function on each subsystems lowest level object and that low level object determines which implementation of the get health and status method that will be used. An additional layer of abstraction is also taken advantage due to the polymorphic structure of the classes. Polymorphism is when multiple classes have a parallel structure. So for this design, the subsystem classes are polymorphic because each of these classes have a initialize, get health and status, and handle command method with the same inputs, outputs and function name. The impact of this is shown in the code block from the Health and Status Thread below. To collect the health and status data from each of the subsystems all the thread needs to have is a vector of type Subsystem Base containing the the instantiated lowest level objects for each subsystem. It doesn't need to know which

subsystem is which, all it it needs to do is iterate through the vector and call the get health and status function for each object. This is one of the best examples of showing how a program that can run nearly 200 unique build configurations can execute a very complicated process with large variability by using only one for loop containing only one line of code.

```
// Run through the sequence collecting health and status from the subsystems
        for (std::vector<SubsystemBase*>::iterator i = healthSeq.begin();
                i != healthSeq.end(); i++) {
            health.push_back((*i)->getHealthandStatus());
        }
```

Devices

The next subcategory of the MCB software architecture is the Devices. Each device has its interface class which handles data acquisition and processing at one abstraction level above the hardware abstraction layer (or the serial drivers). This means that it does not need to know how the devices serial protocol works at a low level, it just needs to know what functions to call to control or monitor the device. The Temperature Interface class, for example, consists of an constructor, deconstructor, and two functions: a begin sample routine and a get sample routine. When a Temperature Interface object is instantiated for a temperature sensor, a reference to the One Wire Manager is passed in along with a unique identification (ID) for that temperature sensor. Note that the One Wire Manager and unique ID will be discussed in more depth in the next section. When the begin sample function is called by some arbitrary caller, it just needs to call the One Wire Manager's write to file function and identify which device the sample needs to begin for with that devices unique ID, what file it needs to write to (the start file), and what value to write to that file (a 1 specifies that the sensor should begin taking a measurement). Similarly for the get sample routine, the Temperature Interface just needs to call the One Wire Manager's read from file routine once again specifying the unique device ID and which file it would like to read from. Once it gets the measured data for the device, it performs error checking on the read value and then converts the data to the desired data representation if the checks pass. An example of this can be seen in the Appendix in Listing 8.

This same level of abstraction is present in the seven other device interface classes: the Ammeter Interface, Potentiometer Interface, ECU Interface, LabView interface, Solenoid Interface, Load Cell Interface, and Actuator Interface. Recall the above code example of the WPM Class where it collects temperature data. This is another layer of abstraction. When the WPM needs to collect temperature data, all it needs to do is call the begin sample function followed by the read sample function. The WPM is does not need to know any more information regarding the temperature data collection processes as this is handled at much lower abstraction levels.

Serial Drivers

Similarly to the subsystems, the serial drivers are also implemented with a multi-level hierarchy that utilize polymorphism, abstraction, and inheritance. The highest level class is the Hardware Manager which simply defines a type for all of the inheriting classes as well as an initialization routine.

The Bus Manager is a template class that inherits from the Hardware Manager. There are four physical hardware busses, or device types, that contain all of the devices: the 1-Wire Bus, the I2C Bus, the RS232 Bus, and the GPIO bus. Each of these is a child class that inherits from the Bus Manager and Hardware Manager Class: the One Wire Manager, I2C Manager, RS232 Manager, and the GPIO Manager. Furthermore, each of these classes has its own Mock child class at the fourth hierarchical level: The Mock One Wire Manager, Mock I2C Manager, and the Mock GPIO Manager. These child classes will redirect any methods that require hardware interactions to their own implementation of the method. Hence there can be a total of 8 device types.

Only one class of each of the 8 device type managers is allowed to be instantiated, as each object is responsible for managing its entire bus. Before a bus can be initialized, all of the devices it will use must first be attached. The attachment procedure uses a registry system using a devices information, adds that

device to the bus so long as it has not already been registered, and returns the unique device ID that was mentioned above. The attachment process takes place directly before the the devices interface object is created. Once all of the devices have been attached to their bus the main program will go through and initialize all of the devices.

The Bus Manager is a template class because it provides high implementations for all of the routines that every single device on each bus must go through including the attachment routine and the initialization routine. These high level routines then make function calls to methods or reference class attributes that are defined by the corresponding device manager's class. Which implementation gets called is mapped by the type of device (ie. 1-Wire) that an inherited device type class defines itself as.

A small portion for the code of the Bus Manager class and the One Wire Manager class is shown to exemplify the two class' relationship during the initialization process for an entire bus of devices. When the program is ready to initialize all of the devices, it iterates through a vector of type Hardware Manager that contains all of the required device type manager objects. On each object it calls the initializeDevices() function defined in the Bus Manager class. The One Wire Manager Class' definition in the Appendix Listing 9, for example, specifies itself as type One Wire Device which allows the Bus Manager parent class to know what type of device the One Wire Manager class managers. So, when the initializeDevices() method is called on the One Wire Manager Object, the Bus Manager references the vector containing all the One Wire Manager's devices, iterates through this vector, and then calls the initializeDevice() routine for each of the devices attached to the bus. Since the One Wire Manager Class has specified itself as type One Wire Device, the One Wire Manager's implementation of the initializeDevice() routine is explicitly mapped to the Bus Manager call. This is the same for the remaining 7 device types as each of their class definitions specify their unique device type.

Utilities The next subcategory of the the MCB software architecture is the utilities. These contain a variety of important software implementations that are utilized throughout the code.

The first, and arguably the most important, is the Lock Guard. The Lock Guard is a software implemented locking system that ensures the mutual exclusion of shared resources. For example, if the Command Thread is trying to execute a eject fairing command (which changes the GPIO levels of the solenoids) while the Health and Status Thread is requesting the states of the solenoid's GPIO levels, a fatal error will occur from trying to access and change those states at the same time. To prevent this error of having two locations in the code accessing a shared resource at the same time, a locking system was put in place. In this program a resource will be considered at the class level. Hence for all classes that are considered shared resources, they will have their own Lock object attribute (not Lock Guard). When a shared resource in that class is called, for example the GPIO Manager's set method which changes the GPIO level of the specified device, before the GPIO Manager changes the GPIO level it will instantiate a Lock Guard object with its' classes Lock object passed in. This attempts to lock the class which prevents the class' methods from being accessed elsewhere. When creating the instance of the Lock Guard object, it will check if the class is locked. If that class is locked this means that some other location in the code is using one of the GPIO Manager's methods. This instantiation process will wait until the lock has be released. This allows the resource to be accessed as soon as it becomes available. The lock gets released when the function call is completed. This happens when a function call has returned since the program implicitly will call the Lock Guards deconstructor which releases the lock.

The next utility is the Logger. This is a class that is utilized by the rest of the program to send log messages to a terminal console and a file. The logger can be set to one of 6 log levels which are debug, information, warning, error, fatal, and none ranging from the lowest level to the highest level. Setting the log level filters out all messages that are associated with a lower level. When the Debug Level is set, all logger messages will be printed, whereas when the None Level is set, no messages will be printed. Each log message will contain the current Linux timestamp, the log level corresponding to that message, a pseudo representation of the call stack indicating where in the code that message is being printed from, and the message that is requested for printing.

The Watchdog Class, which is utilized by all of the threads, contains all the methods that must be used by or to interact with the Watchdog. This includes methods for kicking the watchdog, starting the threads, and checking the threads.

The Time Keeper utility consists of a header and source file that provides function declarations and implementations for handling and reporting the time. It manages two types of times within the system: the Linux time and the time since the start of program execution.

Likewise, the File System is a small utility that resides in one header file. This header file provides the file locations that are utilized by the logger and each of the serial drivers. It configures the file and directory paths based on the build target (Linux PC or Raspberry Pi), as the file systems on each target vary.

The File Manger is a class that is responsible for all interactions with the target's file system including reads, writes, appends, deletes, and creations. This is a class with static functions and only one globally defined class instance, meaning that all classes can have access to the File Manager without having to pass the File Manager object into their class instantiations nor create a File Manager object themselves. Hence, this improves the elegance of creating all of the other class objects while ensuring the Final Manager has a single locking system.

The utility that manages the packing and unpacking of general Health and Status Packets, subsystem Health and Status Packets, Command Packets, and Command Acknowledgement Packets is the Packet Definitions header file. Within this header file are many structures for each of the packet types. Many of these structures contain packing, unpacking, and validation routines. This ensures that all packets are created and parsed in a very specified and ordered way every single time.

The final utility is the Command Handler Class. This class is responsible for taking in the command packets received from LabView, unpacking the command, performing preliminary validation checks on the packet, and passing the command to the specified subsystem if the preliminary validation checks are successful. Once the command is executed or has failed a preliminary validation check, the Command Handler packages the Command Acknowledgement Packet for the Command Thread to send back to the LabView.

Unit Testing

The final subcategory of the Main Control Board Software is the Unit Testing. There are two types Unit Testing that utilize the Catch2 Unit Testing Framework: Software Unit Testing and Hardware Abstraction Layer (HAL) Unit Testing. The software unit testing consist of several files with test cases for all of the methods written in the software that do not necessitate physical hardware interactions. If a method handles some sort of hardware interaction, the corresponding mock class of that serial driver is utilized. Additionally, the unit tests provide testing for the mock serial drivers themselves. Likewise, the HAL Unit Testing consists of several files with test cases for all of the methods that require physical hardware interactions. All of the software and HAL unit tests include assertions to test nominal operations as well as any edge cases. These tests are not run during normal program execution. Instead they each have their own build configuration that goes through and runs all of the test cases. Note that each test case can have associate tags, so if it is desired that only a select set of test cases are to be run, the executable can be run with one or multiple of the defined tags inputted as parameters.

3.6.2. Software Flow

The Main Control Board Software Flow Chart shown in Figure 41 outlines the procedure for building the program executable as well as the processes that occur when the program executable is ran. Note that this diagram does not describe the execution of the two unit testing builds.

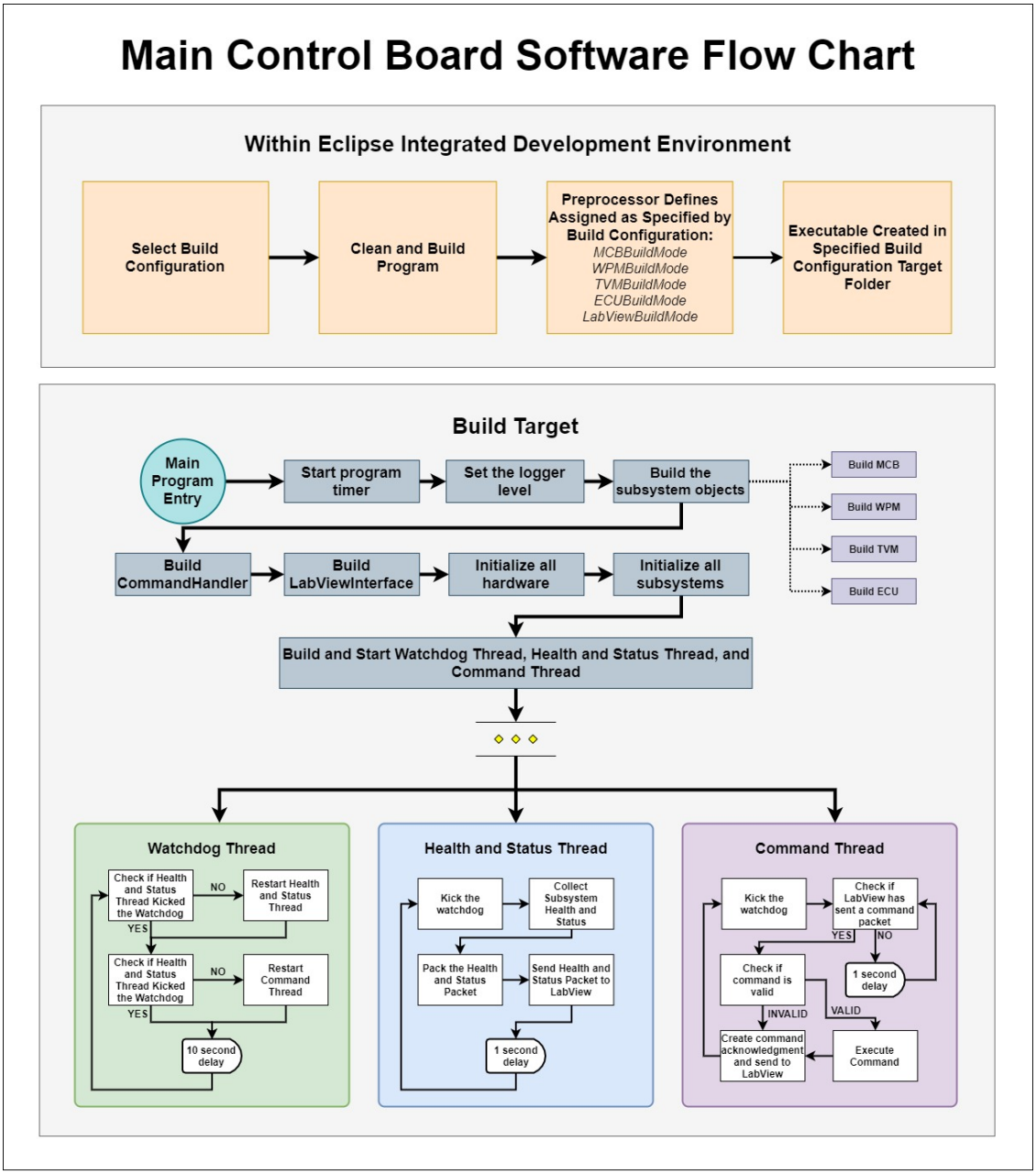


Figure 41: Flow Chart for the Build Process and Program Execution of the Main Control Board Software

Before running the MCB software, the project must first be built within the Eclipse Integrated Development Environment. First one of the build configurations must be selected from the debug icon within the IDE. Each of the build configurations are created within the project settings where the MCBBuildMode, WPMBuildMode, TVMBuildMode, ECUBuildMode, and LabViewBuildMode symbols are assigned corresponding to each build configuration specified in the MCB Software Build Configurations Table (4. Additionally the compiler used for each build configuration is specified depending on the build target. If the build target is a Linux PC it will utilize a 64-bit x86 compiler whereas if the build target is the Raspberry Pi 4 it will utilize a 64-bit ARM cross-compiler. Once a build configuration is selected the program can be cleaned and built. During the pre-compilation build process, the IDE will swap out the subsystem build mode preprocessor defines, or macros, in the program with the symbols those defines are mapped to. Each build configuration will contain its own target folder, so once the compilation process is complete, that software build's executable will be placed in its corresponding target folder.

When the program executable is run on the target device, the code will run the processes indicated in the Build Target box below. It will start the program timer and set the log level. Next it will build the subsystem objects, Command Handler object, and the LabView interface object (using the build mode specified by the LabView's build mode macro). Following this, all of the hardware and subsystems will be initialized. Once all of the objects are created and initialized, each of the threads will be built and then run by the Watchdog. At this point, the program moves into running the three threads in parallel. Each of the three threads run as described in the above Threads section. Note that if the LabView is in software mode, a circular sequence of predefined commands will be continuously passed into the command handler to simulate the receiving of commands from LabView.

In the above flow chart, there are four boxes that stem from the "Build the subsystem objects" box. The processes that occur within these boxes where too involved to include in the flow chart, so they have been diagrammed separately. One of these diagrams can be seen below in Figure 42, which describes the process of building the MCB subsystem object. The diagrams for other subsystem class objects can be found in Appendix B, in Figures 106 - 108.

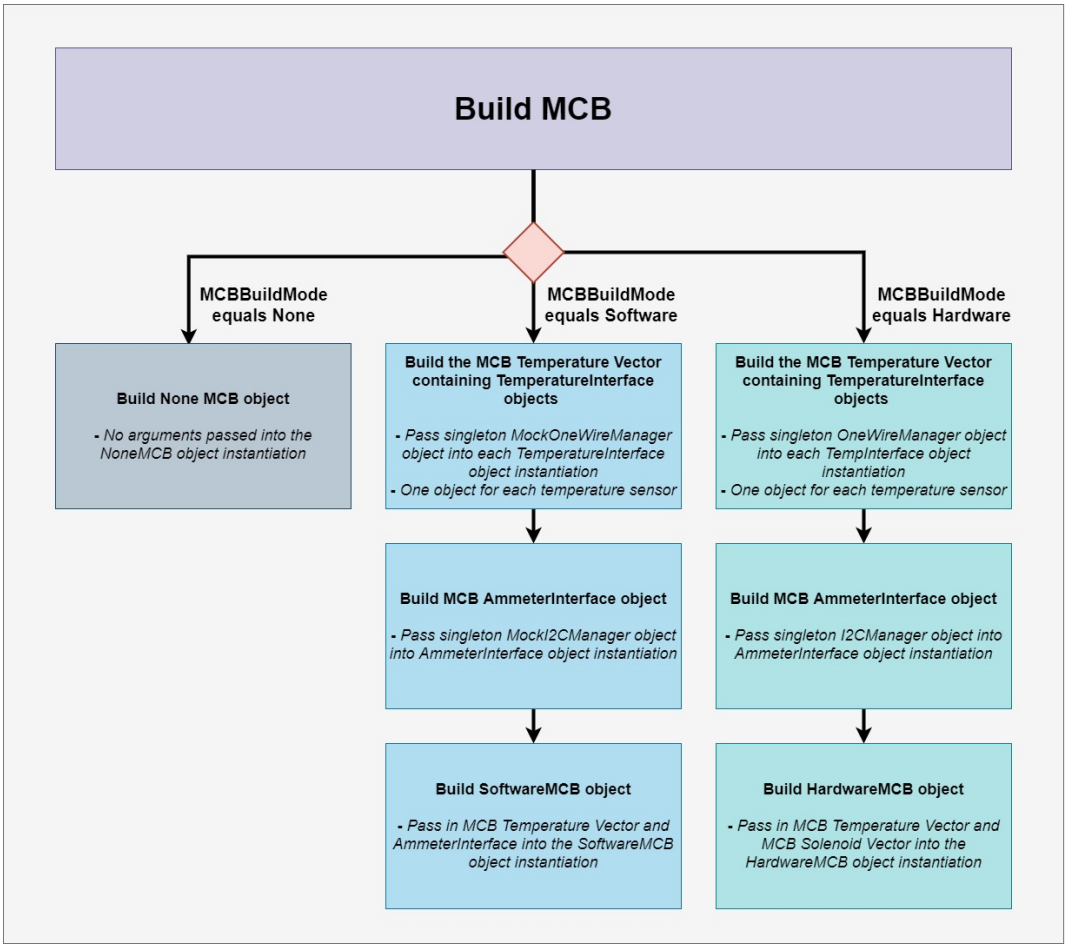


Figure 42: Flow Chart for the Creation of the Main Control Board Subsystem Object

For the Main Control Board subsystem, either a None MCB, Software MCB, or Hardware MCB class object will be instantiated. Which object gets created is dependent on the value of MCBBuildMode symbol. If this symbol equals None, then the None MCB object will be built with no arguments passed into the object instantiation. If the symbol equals Software, the MCB Temperature Interface object vector will be built using the Mock One Wire Manager and the MCB Ammeter Interface object vector will be built with the Mock I2C Manager. The Software MCB object will then be instantiated with the two vectors passed in as the constructors arguments. Finally, if the MCBBuildMode symbol equal Hardware, the Hardware MCB

object will be created with the MCB Temperature Vector and MCB Ammeter vector built with the One Wire Manager and I2C Manager, respectively. The same general process follows for the WPM, the TVM, and the ECU subsystems.

3.6.3. Packet Definitions

As stated previously, the Main Control Board software must handle the unpacking, validation, and packing of multiple types of packets for sending to and receiving from LabView. The first type of packet is the Command Packet as defined in 8 of the Appendix. The Command Packet is either a 7-byte or 23-byte packet containing either four or five fields. First is the command packet opcode, OP_CMDPKT, which is a unique opcode that identifies the packet as a command packet. Next is the subsystem opcode, OP_SUB which specifies which subsystem the command is for. The OP_CMD field then specifies which subsystem command is to be executed. If the command is an adjust thrust vector command, the TV_PARAM field will be included in the command packet to indicate what linear potentiometer position is desired for each of the four potentiometers. The last field in the packet is the checksum which is used to validate that all of the bits in the received packet are correct.

The list of commands that the Main Control Board Software is able to accept are outlined in Table 5.

Table 5: Subsystem Command Definitions

Commands List			
Subsystem	**Command Name**	**Description**	**OPCODE_CMD**
MCB	REBOOT	Reboots the Raspberry Pi	0x00
	REQUEST_HS_FILE	Requests the health and status file	0x01
	REQUEST_LOG_FILE	Requests the log file	0x02
WPM	FAIRING_EJECT	Ejects the fairing	0x00
TVM	ADJUST_TV	Adjusts the thrust vector to position specified by TV_PARAM	0x00

The make up of the Command Acknowledgement Packet is defined in Table 9 of the Appendix. It is a 16-byte packet containing the same fields as the Command Packet but with the addition two timestamps that are collected once the command has been handled: the Linux time as well as the time since the start of program execution. Additionally, there is an error field which indicates to LabView if an error occurred when processing the command request and what that error is if applicable.

The largest packet is the Health and Status Packet. The definition for this packet can be found in the Appendix in Table 10. This packet contains data that the LabView utilizes to monitor every device within the entire system. It is a total of 83 bytes long.

4. Manufacturing

Authors: Jon Weidner, Declan Murray, Alexandra Paquin, Liam Sheffer, Lucas Zardini, Zoe Witte

4.1. Windmill Prevention Mechanism

4.1.1. Scope of Manufacturing

The following table, Table 6, shows the current state of all components needed to build the WPM.

Table 6: WPM Component Manufacturing Scope

Component	Acquisition Method
Fairing	Manufactured
Fore Retention Ring	Manufactured
Solenoids	Acquired
Solenoid Pins	To be Manufactured
Ejection Spring	Acquired

4.1.2. Fairing

Using the Markforged Mark 2 printer located in the University of Colorado Aerospace Building, the windmill prevention fairing has been manufactured using carbon fiber threaded Onyx plastic. This allows for high strength material to be used at a low manufacturing cost. One difficulty that arose was the low quality printing on the inside of the fairing, but since this area is not exposed to airflow, it was decided that it would not be an issue during testing. The printed fairing is shown in Fig. 43.

4.1.3. Fore Retention Ring

The fore retention ring was manufactured in the same method as the fairing, by using the Mark 2 printer. This manufacturing method allows the ring to be manufactured at a low cost, while still fitting the thermodynamic requirements of the ICUP boards. The final printed fore retention ring is shown in Fig. 44.

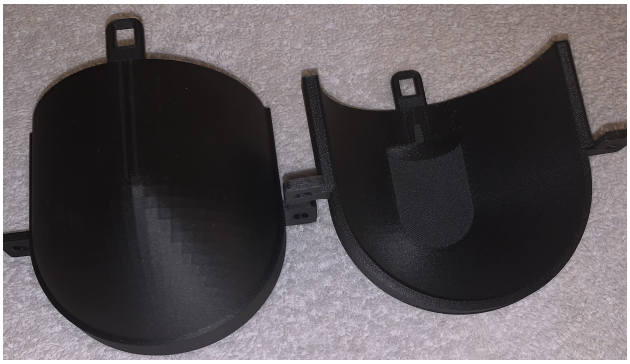


Figure 43: 3-D Printed Fairing

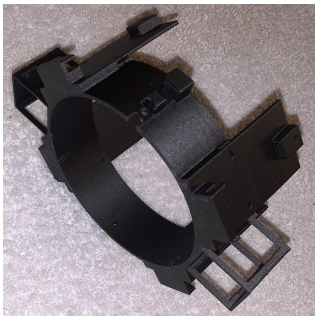


Figure 44: 3-D Printed Fore Retention Ring

4.1.4. Solenoid Pins

The solenoid pins were in the process of being manufactured when the project was brought to a halt. The pins were manufactured using aluminum provided by the machine shop at the aerospace building. One pin was was 50% done when the shop closed down.

4.1.5. Engine Integration

Partial integration tests were able to be accomplished with manufactured components. Figure 45 shows a partially integrated system, missing the COTS solenoids. The fore retention ring would be attached to the engine using the screw holes that can be seen at the edge of the purple compartment of the engine. The ring would be attached such that it does not interfere with the engines ability to be mounted onto the test stand.

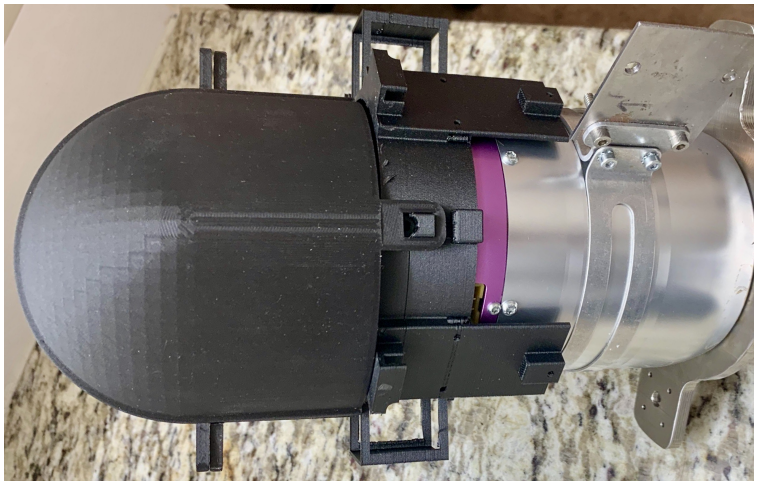


Figure 45: Partially Integrated WPM System

4.2. Thrust Vectoring Mechanism

4.2.1. Scope of Manufacturing

The following table, Table 7, shows the current state of all components needed to build the TVM.

Table 7: TVM Component Manufacturing Scope

Component	Acquisition Method
Nozzle Sheath	Manufactured
Sheath Hinges	Manufactured
Paddles	Manufacturing in Progress
Clevis Pins	Acquired
Linkage Rods	Manufacturing in Progress
Actuator Ring	Manufactured
Linear Actuators	Acquired
Test Stand Modification	Manufactured

4.2.2. Thrust Vectoring Mechanism Manufacturing Process

The bulk of manufacturing was completed using mills. CNC mills created the nozzle sheath and the actuator ring. A CNC mill was required as these elements had curves that could not be made manually. The linkage rods and paddle necks were made using manual mills, as simplicity of cuts and frequent piece flipping made manual mills more time-efficient. The paddle faces were intended to be also CNC milled, but an experimental set were produced using 3D-printing in stainless steel. Additionally, the sheath hinges or journal bearings were also printed in this way.

Nozzle Sheath The nozzle sheath was manufactured in two pieces: the base annulus, and the square ring. The base annulus has slits cut adjacent to the tabs so that when the tabs are bent at a right angle, the center of curvature of the bend would not extend past the inner diameter. The tabs were hammered into the 90° bend using a square block as a jig.

The square ring was machined from the sheet metal flat, the tabs were simultaneously bent to their 20° angles using a brake press, then made into a square by bending at specific locations, again using the brake press. The two pieces were then welded together with a TIG welder. The nozzle sheath was anticipated to be the biggest manufacturing challenge, so the entire component was built with similar but inexpensive sheet metal to test and improve the process. This proved to be invaluable, as the manufacturing lessons learned led

to the second, flight version to be of significantly higher quality. The final version of the integrated nozzle sheath and actuator ring are shown in Fig. 46. The integration of these mentioned components was very successful, but the true test would have been the integration of the two TVM sub-assemblies. Connecting the linear actuators to the linkage rods to the paddles would be the last step before mechanical and hot-fire testing of the whole system.

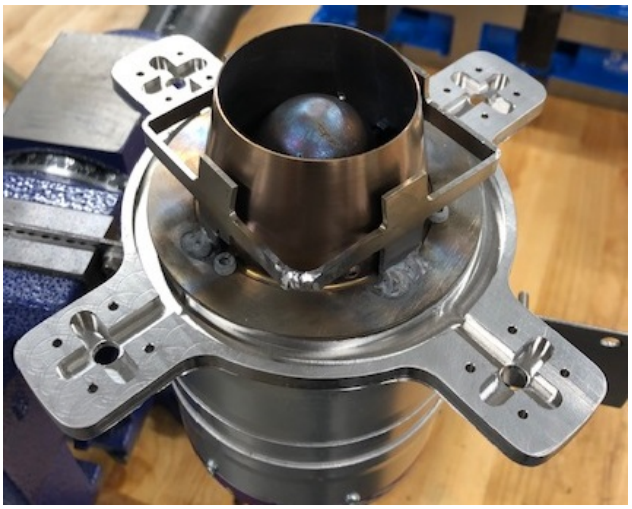


Figure 46: Finished Nozzle Sheath and Actuator Ring Integrated with Engine

4.2.3. Test Stand Modification

The test stand modification was a critical part of validating the effectiveness of the thrust vectoring mechanism. The modification took the existing single-axis test stand owned by C.U. and added a second axis. A more comprehensive description of its function is found in Section 5.5. The manufacturing of the test stand modification was completed on the day before the manufacturing shop was closed, and is shown in Figure 47

The test stand consisted of an aluminum plate bolted to four low-friction, self-lubricating pillow blocks which slid along stainless steel rods connected to the aluminum base. This base was designed to mount onto the preexisting test stand, while the engine mounts connect to the upper aluminum plate on the modification as shown in Figure 48

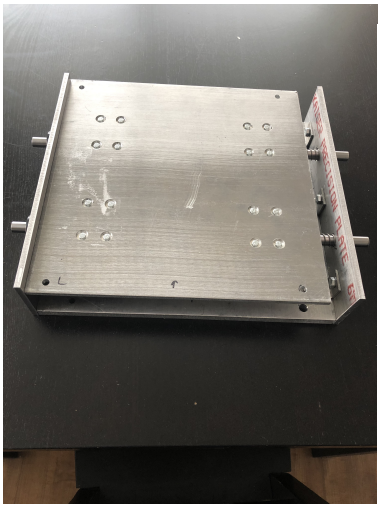


Figure 47: Additional Test Stand Sled

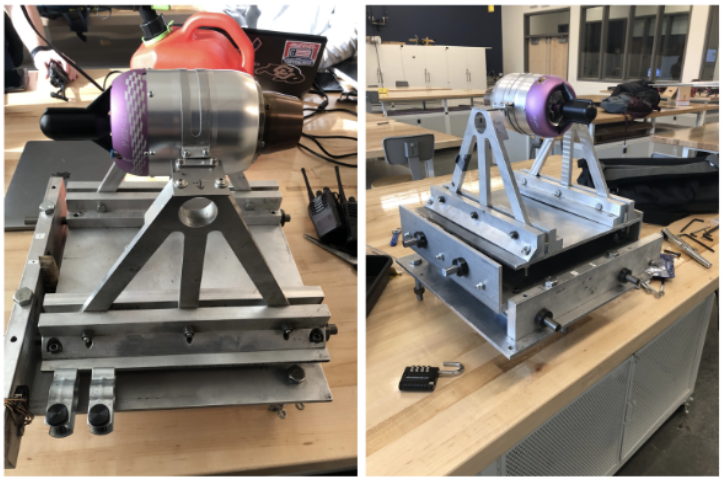


Figure 48: Completed Modified Test Stand

A load cell was to be mounted to the modification (on the left side from the perspective in Figure 47) during testing, and springs were fitted on the steel rods to keep pressure on the load cells and reduce the risk of accidental damage due to any sharp impacts from the sled sliding freely. After manufacturing was completed on the test stand, the upper sled could be easily moved by hand, but without any applied force it would rest against the base as intended. This was the desired behaviour, and gave confidence that the test stand could successfully interact with a load cell without damaging it or causing a great deal of error due to friction.

4.3. Main Control Board Software

The Main Control Board Software was manufactured utilizing an incremental development processes with a large emphasis on unit testing. The overall timeline of the development process was initially structured to meet a series of milestones corresponding to each level of success, starting from the first level and ending with the last level. By meeting a milestone, this meant that all of the Main Control Board Software elements that were required to demonstrate a level of success were completed and tested. Additional milestones were defined to break down the initial workload since all of the Utilities had to be developed before any of the serial drivers, device interfaces, subsystem classes, and threads. The specific milestone due dates were determined based on the expected completion of the other subsystem's elements. Since the Windmill Prevention Mechanism was expected to be completed approximately one month prior to the Thrust Vectoring Mechanism, the completion of the its first and second success criteria within the MCB software was initially the primary focus. Figure 49 shows the sequence of milestones that were set to be achieved, the software additions required to complete that milestone, the current status of each software addition, as well as the estimated time left to meet that milestone. Note that several of the software elements were developed out of order. This is because the first revision of the electronics needed to be tested, so all hardware related elements had to be developed.

Milestone	Hours Left	New Software Additions	Current Status of New Software Additions					
			Implementation			Unit Testing		
			Incomplete	Partially Complete	Complete	Incomplete	Partially Complete	Complete or Not Applicable
Implementation and software unit testing of the utilities	0	Look Guard						
		Logger						
		Watchdog						
		Time Keeper						
		File System						
		File Manager						
		Packet Definitions						
		Command Handler						
Implementation and software unit testing of Windmill Prevention Mechanism Success Criteria 1 dependent classes	0	Hardware Manager						
		Bus Manager						
		Mock One Wire Manager						
		Mock GPIO Manager						
		Mock R8232 Manager						
		Temperature Interface						
		Solenoid Interface						
		LabView Interface						
		Subsystem Base						
		MCB						
		WPM						
		TVM						
		ECU						
		None MCB						
		None TVM						
		None ECU						
		Software MCB						
Windmill Prevention Success Criteria 1	0	Health and Status Thread						N/A
		Command Thread						N/A
		Watchdog Thread						N/A
		Main Program						N/A
		Object Building Routines						N/A
Windmill Prevention Mechanism HAL Unit Testing	0	One Wire Manager						
		GPIO Manager						
Windmill Prevention Mechanism Success Criteria 3	1	Hardware MCB		90%				
		Object Building Routines Update		90%				N/A
Implementation and software unit testing of Thrust Vectoring Mechanism Success Criteria 1 dependent classes	15	Mock I2C Manager						
		Potentiometer Interface		55%			55%	
		Actuator Interface		50%			50%	
		Load Cell Interface						
		Software TVM		70%			70%	
Thrust Vectoring Mechanism Success Criteria 1	0.5	Object Building Routines Update						N/A
LabView and MCB Integration	2	R8232 Manager					90%	
Thrust Vectoring Mechanism HAL Unit Testing	0	I2C Manager						
Thrust Vectoring Mechanism Success Criteria 2 and 3, Success Criteria 4	0.5	Hardware TVM		70%			70%	
		Object Building Routines Update						N/A
Success Criteria 5	7	ECU Interface						
		Software ECU		90%				
		Hardware ECU		90%				
		Object Building Routines Update						N/A
MCB Subsystem Integration	2	Ammeter Interface		55%			55%	
		Software MCB		90%			90%	
		Hardware MCB		90%			90%	
		Object Building Routines Update						N/A

Figure 49: MCB Software Manufacturing Progress

4.4. Electronics

All the PCBs in the project (ICUP, MCB, and LPBB) were designed in Altium. The manufacturing of these boards were done at Advanced Circuits (based in Aurora, Colorado). Two of the ICUP and MCB PCBs were ordered for Revision A. The LPBB was designed and manufactured in a 12 device panel for both Revisions A and B. This means that a flat PCB is placed with 12 LPBBs on it, requiring manual cutting of each LPBB once delivered. This is done at University of Colorado's machine shop. For populating the

PCBs, the standard solder stencil technique is used for all the PCBs. The population for Revision A was done in the Integrated Technology Learning Laboratory's electronics shop.

4.5. Manufacturing Challenges

WHiMPS experienced a number of challenged during the manufacturing phase of the project, beginning with the WPM printing. While the Markforged Mark II printer has amazing precision, with a tolerance of 0.1 mm, there is still error in the dimensions of printing and of the engine measurements taken. The WPM parts were required to have very small tolerances to fit onto the engine, and the resulting print wasn't perfect. WHiMPS was not able to fit the final fore retention ring print onto the engine because it was slightly smaller than the circumference of the engine. However, these issues could have been solved by slightly sanding down the inner surface of the fore retention ring in the CU machine shop.

The major challenges faced when manufacturing the TVM components occurred during the manufacture of the nozzle sheath component due to its complex shape. These challenges were mitigated by creating a prototype out of cheaper scrap material, from which the manufacturing team was able to learn a lot and perfect their methods. One of these challenges was when using the brake press to bend the square ring into its final shape. The brake press had a tendency to create a large radius of curvature at an unpredictable location, which was a problem for the team. In response, the team had to use scrap metal to jig the ring into place, which ultimately improved the placement of the bend. Once the ring and sheate were manufactured, welding them together was a challenging task. Both components were relatively small and difficult to clamp into place, which made welding the pieces together difficult. Even with the help of the CU Aerospace shop supervisors, this task was very difficult.

Additionally, the sheer amount of work required to manufacture the components was a challenge itself. Manufacturing the nozzle sheath and actuator rings took a significant amount of time for the WHiMPS manufacturing team, and left a number of smaller components to still be manufactured. This was a challenge, but it was overcome via additional members of the testing team taking shop courses and helping out.

There were a few challenges when it came to populating the PCBs. Some components were not ordered in time for the first round of population, and pieces from other boards had to be taken in order to complete most of the population at first run through. Many of the components were also extremely small, which required precision in using the solder paste. Overdoing the paste would risk short circuits while under-doing the paste could result in components breaking or falling off. For some of the resistors and capacitors, there was not a sufficient amount of solder paste and they needed to be re-soldered. For others, the paste was too thick and had to be wiped off and done over again. It was also extremely important to pay attention to the orientation of the components so as not to short circuit the board or destroy the component and potentially the board completely. While soldering the LPBBs, there was the challenge of having the linear potentiometer completely flush on the board in order to receive accurate data from it. One of the first LPBBs soldered was not flush, and efforts were made to redo this as well. A high level of detail was also required when crimping the wires to ensure they would connect, and using heat shrink on wires that were soldered to other components. If the heat gun was placed on the component for too long, like a temperature sensor, it could destroy it.

4.6. Integration Plan

Integration of the WHiMPS system onto the engine essentially involves installing bolt-on parts to the engine due to the design decisions to create all engine hardware separate of the engine. This decision was made to mitigate risks to the engine's operation, but also allowed for an easier engine integration. The WHiMPS engine integration plan has two separate phases related to the WPM and TVM respectively.

The WPM integration would begin by installing the fore retention ring into the pre-existing bolts on the front of the engine. These bolts must first be removed, and the fore retention ring is slipped over the starter motor until the engine bolt holes line up with the holes on the fore retention ring. The bolts are symmetrically spaced about the circumference of the engine, so the fore retention ring must be rotated until the fairing connection tabs are perpendicular to the stock starter motor supports. This must be done for the

ejection springs to operate without interference from the starter motor supports. Next, the ICUP panels are installed onto the fore retention ring and connected as described later in this section. Solenoids are placed into the solenoid housings, and the solenoid pins are placed into the attachment points with the solenoids fully retracted. The clevis pins can then be installed to finish the solenoid subassembly. Finally, the fairing halves are installed in the same method that they would be installed prior to every mission in the WHiMPS mission CONOPS. The fairing retention tabs are hooked onto the ICUP connection points, and rotated into place. The springs, attached to each fairing half with an adhesive, is pressed against the starter motor. Once the halves are fully installed with the seam completely overlapping, the solenoids can be extended, pushing the solenoid pins into the solenoid pin tabs. After this is completed, the WPM system is fully integrated and functional.

The team would conduct preliminary integration testing. This would begin with software/electronics interface testing, by testing the operation of the solenoid. A signal would be sent from the MCB to pull the solenoids, and the team would determine whether this signal is properly received by the ICUP and solenoid by seeing if the solenoid retracts. The WPM integration would have been tested by removing the fairing, and applying approximately 90 N of force to the fore retention ring. Given that the drag force is bounded by approximately 88 N of force, this would verify that the structural components are secure. Testing of the fairing mechanical system's integration would be completed during the actual test, as it is a part that will be continually installed and ejected throughout the testing phase.

The TVM integration begins with the attachment of the nozzle sheath and actuator ring. Similar to the fore retention ring, these components are installed using pre-existing bolts. These are the three bolts used to hold the stock nozzle onto the engine. Once these bolts are removed, the actuator ring can be placed over the stock nozzle, flush with the base of the stock nozzle. Once the bolt holes are aligned, the nozzle sheath can be placed on top of the actuator ring and aligned. The nozzle sheath's position must also align the stock exhaust gas temperature sensor with its corresponding cutout in the nozzle sheath. Once everything is aligned, the bolts can be tightened onto the engine. Following the structural component installation, the linkage sub-assembly can be installed. This process begins by attaching the paddle to the nozzle sheath via a clevis pin in the through hole furthest from the paddle surface. Then, the linkage rod can be attached, also via a cotter pin, to the through hole closest to the paddle surface. Following the linkage assembly, the actuator assembly can be integrated. This is done by mounting the actuator pods to the actuator ring, and the actuators inside of that. The actuators are installed with the actuator nut component in place, which is then mounted to the linkage rods via another cotter pin attachment. Once the LPBB's are mounted, the TVM electronics system can be calibrated and operation can begin.

The testing of the TVM integration would occur during the TVM software/electronics calibration phrase. This testing would entail commanding various thrust vectoring angles into the LabView, and measuring the corresponding paddle angle. Using the control laws shown in Figure 21, the closed loop feedback control can be calibrated to ensure that the paddles deflect to the desired angle. Additionally, this would have tested the integration of the TVM mechanical system by ensuring that all kinematics move as expected. Once a paddle is deflected, approximately 10 N of force would be applied to the center of the paddle to test whether the system is properly integrated and rigid.

The electronics can be assembled after the critical hardware has been set. Starting with the MCB, this device will be placed in the MCB case in the test room. This can be done by placing the MCB on 4-40 spacers in the box (power circuits facing the switches side of the case's lid), then fastening the lid on the case with 4 4-40 screws. Then take the ICUP to MCB connector and attach it to the MCB through the case. Next is the ICUP. These panels use the same 4-40 screws to integrate with the fore retention ring. Each panel is oriented where the connector points towards the aft retention ring. Once these panels are screwed on, the ICUP panel to panel connectors can be latched into place. Once the ICUP has been assembled, the actuators, LPBB, and the actuator temperature sensor can be placed. The actuators and their temperature sensors are kept in these pods, and the LPBBs are screwed onto the pod (using $4-40$ screws). These pods are in line to the ICUP panel they connect to. The remaining sets of wires should then attach to their respective ICUP

female connector (four actuators, 4 temperature sensor, and 4 LPBBs). Finally, the on engine components will communicate to the MCB with the ICUP to MCB connector, so once testing is ready to commence, this harness is integrated connected to the ICUP (Panel 1).

5. Verification and Validation

Authors: Alec Bosshart, Andrew Meikle, Julia Kincaid, Alexandra Paquin, Zoe Witte, Liam Sheffer, Andrew Robins, Lucas Zardini

5.1. Electronics Verification

Each electronics board revision had to be tested in order to debug and design new models. The MCB Revision A tested switch functionality, the voltage converter, the power protection circuit, the GPIO/LED circuit, the power MOSFET circuit, and the Raspberry Pi functionality. Everything on Rev. A was successful, but additional devices were needed on the MCB, so a second revision was designed. The ICUP Revision A tested I2C functionality, the solenoid circuit, and the actuator/motor driver circuit. The I2C tested successfully, the solenoid circuit tested successfully after Revision B, and the actuator/motor driver circuit failed. This failure was due to using the wrong motor driver for the selected actuators and was fixed by ordering the correct corresponding driver. The LPBB Revison A only had to test the linear potentiometer circuit and converting it to a digital signal, which passed successfully. The need of revision B was due to the lack of addresses on the ADC chosen for each LPBB. To access each LPBB individually, each one needs a separate address, so Rev. B included a 4 channel ADC with the same circuitry as Rev. A.

Each board has a harness to transfer the data/power to the MCB, so these harnesses needed to be tested for continuity. The original connector harnesses for all boards passed the continuity check and have been used for all tests requiring multiple PCB integration. In addition, there are 12 temperature sensors placed throughout the system using the same temperature device. Each one has been tested on a breadboard correctly (showing a specific address per sensor on the one wire bus).

The MCB, ICUP, and LPBB have been redesigned for revision B. Only the LPBB design has been sent to be manufactured. The MCB and ICUP designs have been checked by Advanced Circuit's DFM test, but were not ordered due to the project coming to a halt. The next steps would be ordering the ICUP and MCB and populating them. Further verification and validation for the electronics would include retesting the Rev. A functionalities (just to test that the devices were populated correctly), then testing the additional devices added to the ICUP and MCB. Once these were accomplished, the integrated system tests with the engine could commence.

5.2. Main Control Board Software Verification

The primary method for with the Main Control Board Software was verified was through the software unit testing and the hardware abstraction layer unit testing. For each method that was developed, a unit test was also developed to check that the method handled nominal cases correctly as well as edge cases. Figure 50 shows the output from running the software unit testing build configuration. As shown, there are 25 test cases with a total of 208 assertions. Note that one test case was typically utilized to test multiple methods, which explains the low test case number. The output shows that all of the tests have passed, which provides verification that each of the implemented routines are working correctly. The output from the HAL unit testing is not included as we no longer have access to the hardware that was required to run these unit tests, hence it could not be acquired.

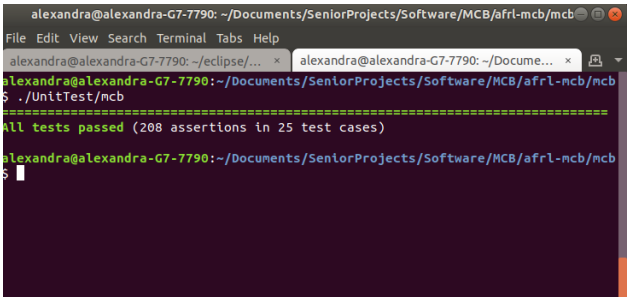


Figure 50: Output from Running MCB Software in Software Unit Testing Build Configuration

The non unit test build configurations are verified by looking at the logger messages printed to the serial terminal and the log file. These are very detailed messages which provide information on everything that is going on with the program. If the logger messages look correct and the system is operating nominally, the build configuration was deemed verified. If any errors occurred when the non unit testing builds were being tested, the build configuration would be switched to one of the unit testing builds. Following this, all of the unit tests would be run. More often than not, the error that was present in the non unit testing build would be exposed in the unit tests making it easy to identify and resolve. The only types of errors that the unit tests are unable to catch in the main program are those related to thread timing, resource sharing, and overall sequencing. Hence, if the unit test does not identify any errors in the program, the error is most likely related to one of those three error types.

5.3. LabView Verification

To verify the models, there were several mechanisms to take data and be able to visualize data. The WHiMPS team developed a test user interface that was able to take and interpret data from the borrowed National Instruments DAQ devices and sensors purchased by the WHiMPS. There was a separate user interface being developed by the WHiMPS that was going to be able to interpret and collect the data that was gathered by the MCB and Raspberry Pi that was being developed at the time of the shut down. For the testing purposes, temperature data was gathered using thermocouples and the NI 9212 DAQ device. The thermocouples were K type thermocouples. Force data was taken using the load cells and the NI 9205 DAQ device. The NI 9205 DAQ contained the Wheatstone bridge circuit and just measured the differential across the load cell, and the load cell was powered externally. The mV/V value, the capacity, and unit conversions were used to convert the voltage to Newtons which is a unit of force. The LabVIEW that was used to conduct testing was able to take in the data from the NI DAQ devices, complete any necessary calculations, display it on plots on the User Interface, and save the data to an excel file for plotting and interpretation after the test was conducted. Figure 51 and Figure 52 show the view of the User Interface component of the LabVIEW, and the back end of the LabVIEW that includes the graphical programming.

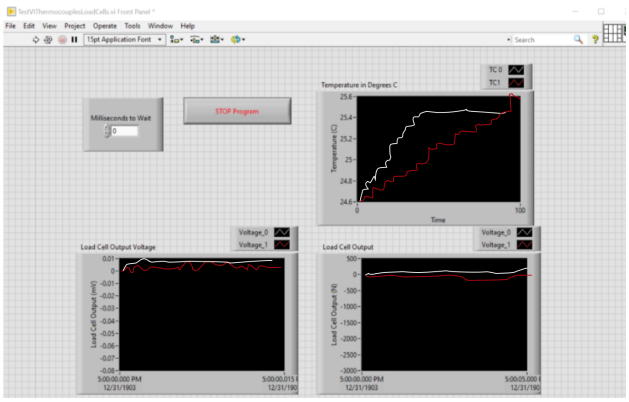


Figure 51: Test LabVIEW Front Panel with Sample Data Plotted

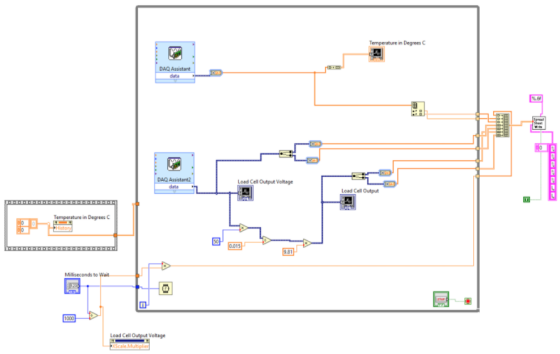


Figure 52: Test LabVIEW Block Diagram

Using this LabVIEW, along with the sensors and NI DAQ devices, the WHiMPS verified expected models. The primary thermal model was maximum temperatures for the engine exterior provided by JetCat, which were well above the temperatures seen at the various locations by the thermocouples. The thrust model was maximum thrust data provided by Jetcat. This value was 100 Newtons. This value was slightly surpassed, but due to the internal sensor error. Therefore, the thrust model was also corroborated. At the time of the shutdown, the LabVIEW to control all of the WHiMPS hardware was in the process of being developed. The final User Interface can be seen in Figure 53.

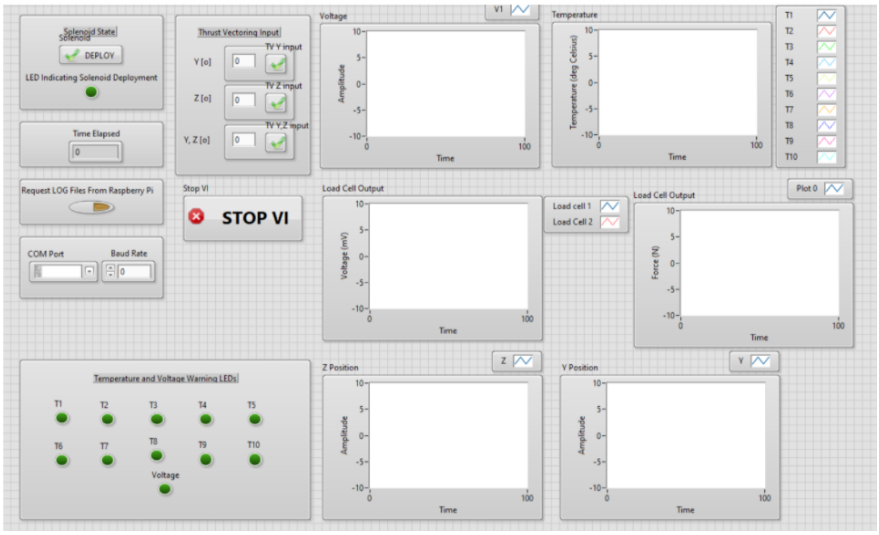


Figure 53: WHiMPS LabVIEW Front Panel

The methodology behind the development and the flow of information can be seen in Figure 54.

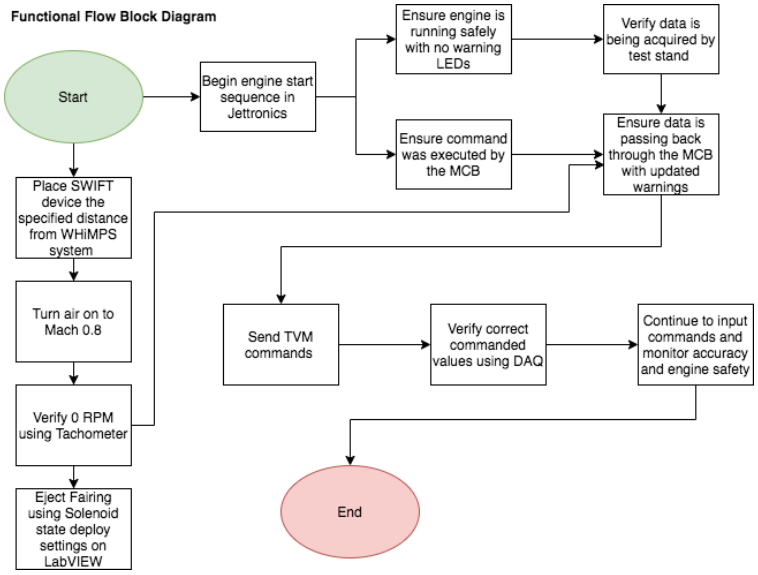


Figure 54: WHiMPS LabVIEW Functional Flow Diagram

The WHiMPS LabVIEW developed for our system would have replaced the NI DAQ devices with the Raspberry Pi 4 and the WHiMPS electronics connected to sensors. The LabVIEW would have displayed this data and saved it to an excel file to be interpreted later.

5.4. WHiMPS Engine Function Verification

Before modifications to the engine were attempted, the WHiMPS needed to verify that the engine worked, was able to be controlled with the available hardware, and would not automatically shut off when the exhaust flow was disturbed. To accomplish this, two separate tests were conducted. The first test was a stock run of the engine, and the second test was an engine run with dummy paddles attached to the rear of the engine. The dummy paddles were representative of the paddles designed to deflect the exhaust from the engine. Both tests were conducted in the CU Aerospace Engine Test Cell. Neither of these tests directly validated any requirements, but the WHiMPS confidence that future tests would be successful.

The stock engine run was conducted in the test cell using the test stand used by previous senior project teams. The test stand used was the unmodified stand discussed in Section 4.2.2. The purpose of this test was to familiarize the WHiMPS team with the engine operation and to collect preliminary temperature data. The temperature data was collected using K-type thermocouples attached to the engine with Kapton tape. The engine run was a success, and the temperature data collected by the thermocouples is shown in Figs. 55 and 56.

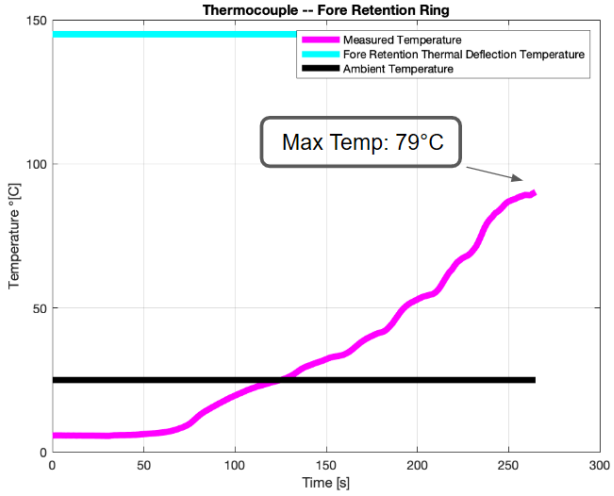


Figure 55: Temperature at Fore Retention Ring Location

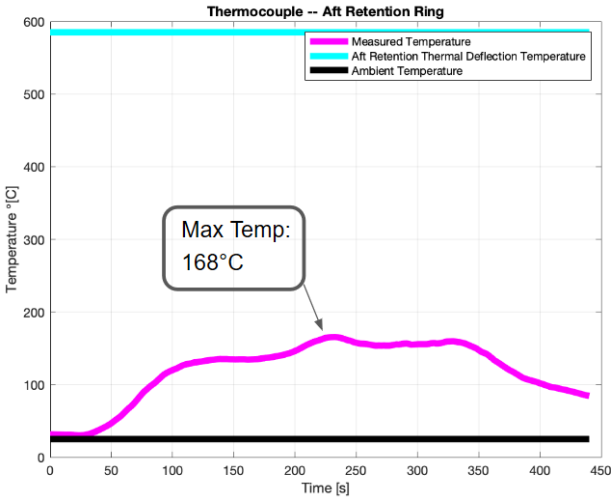


Figure 56: Temperature at Aft Retention Ring Location

These data were collected over two different engine runs. The fore retention ring temperature was measured during a run where the fuel ran low and the test had to be cut short. Due to this issue, the temperature was not able to reach a steady state value. The collected data showed that the temperature of the engine did not match up with the WHiMPS' model. The team expected the temperature of the aluminum engine body to be approximately room temperature (25°) per preliminary recommendations from previous teams. The temperature of the engine body does not approach the thermal limit of the aft retention ring location. It is unknown how close the fore retention ring location temperature would approach the temperature limit of the carbon fiber reinforced Onyx plastic since the test did not reach a steady state. This datum was intended to be measured more accurately in future tests.

There is uncertainty in the need to insulate the electronics on the fore retention ring due to the uncertainty of the maximum temperature at the front of the engine. A simple thermal model was conducted based on the conduction of heat through the fore retention ring, shown previously in Figure 15. As the worst case, the maximum engine temperature is the temperature limit of the ring material. Depending on the convection coefficient around the retention ring, the electronics could experience temperatures between 59°C and 125°C. The temperature limit of the electronics is 95°C, so depending on the actual temperature of the engine and the amount of convection there may be no need for insulation. The model also predicts that a small amount of forced convection cooling from a fan would sufficiently cool the electronics during testing. This would not be an issue on a fully integrated system during a mission as airflow from flying would provide sufficient convective cooling. Further testing would have verified if insulation would be needed, which would have determined if the electronic system could have been successfully mounted on the engine.

The dummy paddle test used the same test setup as the stock engine run. Temperature data was also collected on the back of one of the dummy paddles. The purpose of this test was to ensure the engine would not activate any safety features and shut off when the dummy paddles impeded the exhaust flow. The test was successful and the engine did not shut off or experience any abnormalities during the test. The temperature data collected by the thermocouple on the back of the dummy paddle is shown in Figure 57.

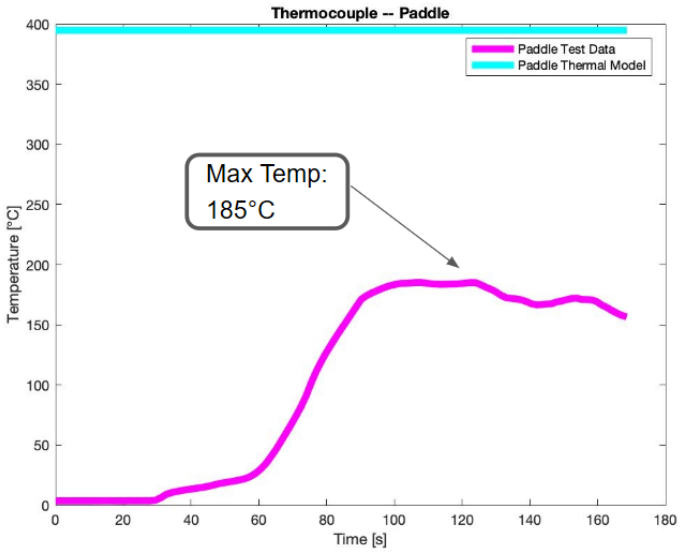


Figure 57: Temperature on Back of Dummy Paddle

The thermal model of the paddle, shown in Figure 18, predicted a maximum temperature of 400°C. The temperature on the back of the paddle never reached this temperature as seen in the data. However, after approximately 100 seconds into the test, the thermocouple started separating from the paddle due to the Kapton tape melting. This introduced a layer of air between the paddle and the thermocouple. The thermal model was a worst case model, so it is not surprising that the temperature of the paddle was lower. Given the thermal limitations of the material selected for the paddles being well above this worst case model, WHiMPS was not concerned with paddle damage due to thermal conditions.

5.5. Windmill Prevention Mechanism Verification

Two types of tests were to be conducted to verify the windmill prevention portion of the project. The first test was a replica of the demonstration that would have been conducted by the AFRL. This test was completed. The second test would have been a test of the fairing under flight conditions, using high speed (Mach 0.8) air to determine the integrity of the fairing and its seal. This test was not completed.

5.5.1. APOP Replica Test

The APOP replica test was designed to mimic the test setup that would have been used by the Air Force to test the WPM. The test setup uses a PVC tube which fits over the front of the engine with the WPM engaged. A compressed air line is then connected to the PVC tube, which is meant to pressurize the tube to 3.5psi. This pressure was defined by the Air Force and is meant to replicate the dynamic pressure experienced by an object moving through the air at Mach 0.8 at 20,000ft. The test rationale is that if the WPM can withstand the pressure differential it would experience at its mission condition, the mechanism would succeed. This test is meant to be an acceptance test, where the WPM either works or fails. If successful, the test would validate FR 2, the requirement to have a functioning windmill prevention system. This test was conducted in the engine test cell and used the aerospace building compressed air. The RPM of the engine fan blades was measured after the test by analyzing video footage from the rear of the engine. A minuscule piece of tape was attached to one of the fan blades to be able to track how many rotations occurred over time. The test setup diagram given to the WHiMPS by the AFRL is shown below in Figure 58, and Figure 59 is the WHiMPS' test setup in the engine test cell.

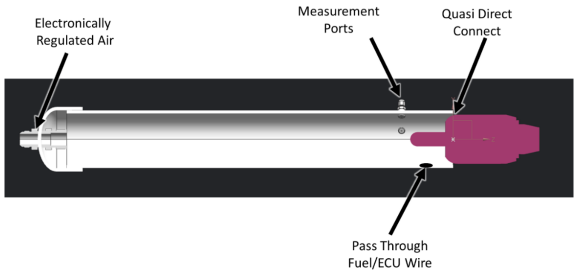


Figure 58: WPM Test Diagram Provided by AFRL

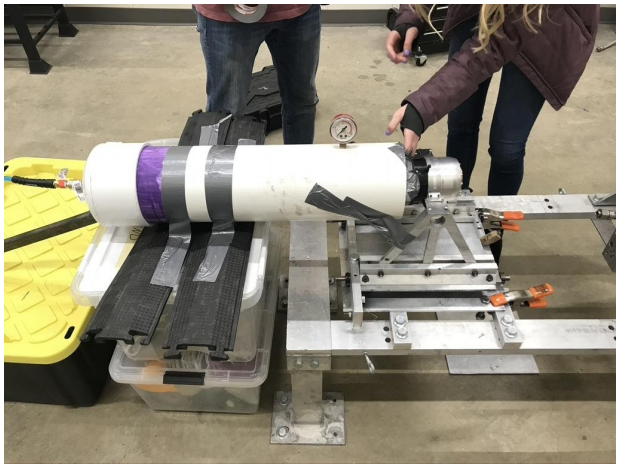


Figure 59: WHiMPS Test Setup for APOP Replica Test

When the WHiMPS conducted this test, the team determined that the building compressed air supply did not have the mass flow rate necessary to produce the needed pressure differential. This was due to a small gap between the fairing and a ring of foam inside of the PVC tube, so all of the air escaped around the fairing and the engine. The team was unable to seal the gap completely because sealing the gap would require the fairing to be artificially held in place. This would not be an accurate test of the fairing design because the natural tendency of the fairing is to shift under pressure due to manufacturing tolerances. However, the video did show that the fan blades did not rotate. Even though this was not at the proper pressure differential, the results still partially validated our CFD model, shown in Figure 60, of the air flowing around the engine and not through the fairing.

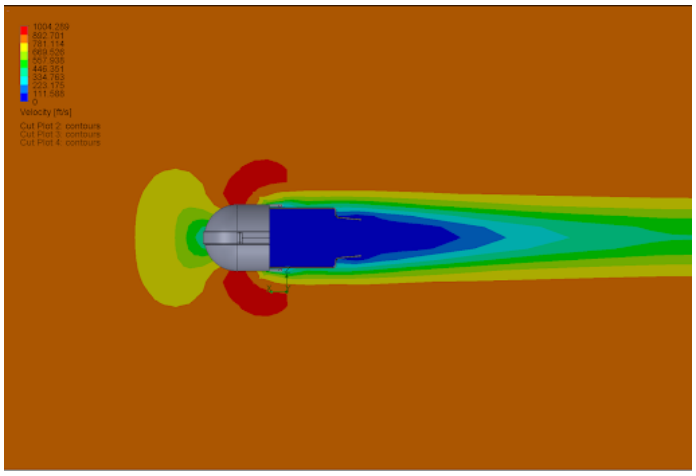


Figure 60: CFD Model of Mach 0.8 Air Impinging on Fairing

The mixed results of the APOP replica test gave confidence that the WPM would be successful when used in Mach 0.8 air (discussed in the next section). This partially validated FR 2, the requirement to have a windmill prevention mechanism. Also, the WHiMPS discussed the issues with the test with the customer and were able to provide insight into the potential issues with the test if it was to be conducted as is by the AFRL.

5.5.2. High Speed Airflow Tests

In addition to replicating the test conditions the AFRL was going to use, the WHiMPS were planning on testing the fairing at the conditions specified in the problem statement. This would have required the team to blow Mach 0.8 air at the engine and fairing. The defined altitude in the problem statement is 20,000ft,

but by conducting the test in Boulder, CO at 5,320ft a small safety margin in the design is demonstrated if successful in testing. Unfortunately, no tests of this type were conducted.

There were two types of high speed airflow tests planned: static and dynamic. Both of these test types would have occurred in the CU engine test cell. The static test required the fairing to be attached to the engine while Mach 0.8 air is blown at the fairing. The test would be successful if the fairing did not break and if there was no rotation of the engine fan blades. This test would validate FR 2 under the design conditions. The dynamic test uses the same setup as the static test, but while the air is still being blown at the engine the fairing is ejected. This test would be successful if the fairing prevents the rotation of the fan blades prior to being ejected and if the fairing does not break during any part of the test. In addition to the hardware, software control of the mechanism would be demonstrated in this test. The dynamic test would have validated FR 2 in the full design conditions and validate the WPM half of FR 3, which requires main control board (MCB) control over the mechanism.

The high speed air was intended to be produced using equipment from the SABRE senior projects team (2016-2017). The equipment was originally used as a cold flow test bed to test nozzle designs meant to achieve supersonic flow. Based on the demonstrated ability for the equipment to achieve supersonic speeds for 20+ seconds, the WHiMPS determined that the equipment would be sufficient for replicating Mach 0.8 airflow for a sufficient amount of time. The test setup from SABRE is shown in Figure 61.

The large tanks shown in Figure 61 are used to hold a large volume of air at 175psi. The air is regulated down to 35psi in the small settling tank. Finally, the air leaves the aluminum nozzle at the end of the settling tank. To achieve the right air velocity, an additional nozzle needed to be manufactured and attached to the end of the aluminum nozzle. The engine would be placed on the test stand and located just behind the 3D printed nozzle. To determine the exit diameter required to produce Mach 0.8 air, the throat diameter used by the SABRE team to achieve supersonic flow was input to the Mach-Area relationship. From this equation, the diameter needed to reach Mach 0.8 based on this throat diameter was determined. The nozzle that would have been used to achieve Mach 0.8 is shown in Figure 62. The nozzle would have been 3D printed using a resin printer, which is what the SABRE team used to demonstrate their system.

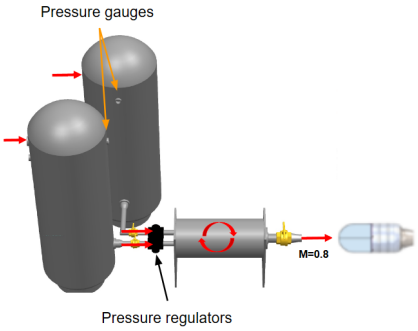


Figure 61: Test Setup from SABRE Senior Projects Team

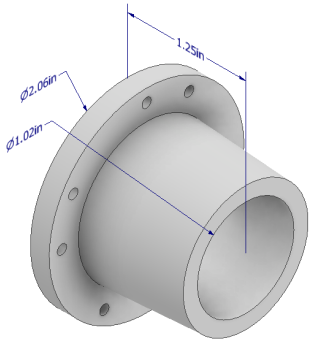


Figure 62: WHiMPS Nozzle for Mach 0.8

Before testing with the actual engine and fairing, the air velocity out of the nozzle would have been measured using a pitot tube to verify it was the correct velocity. Once verified that the combination of pressures and nozzle diameter were sufficient, the pitot tube would not be used anymore. Analysis was going to be completed to determine the optimal distance from the nozzle to measure the air velocity and to place the front of the fairing. A potential issue with the test is that the exit diameter of the 3D printed nozzle is approximately 1in, while the fairing has a diameter of 3.8in. This test would not be perfectly representative of freestream conditions, but very similar conditions would be reasonably expected as long as the engine is placed at the correct distance from the nozzle exit. The RPM of the fan blades would have been collected using the same video camera setup as in the APOP replica test.

5.6. Thrust Vectoring Mechanism Verification

WHiMPS intended to verify the effectiveness of the thrust vectoring mechanism by firing the engine on a two-axis test stand (see Section 4.2.2) and determining the thrust vectoring angle based on the resulting load cell readings. The test would have been conducted in the CU engine test cell. Like the WPM high speed airflow tests, static and dynamic TVM tests were planned. The static test would require the paddles to be deflected before the engine is run. The resultant thrust vector angle would be measured given the paddle deflection. This test would have verified FR 1, the requirement to have a thrust vectoring system. The dynamic test involves the paddles being set at a zero position before the engine is started. Once the engine is running, the paddles will be deflected to a designated angle. With this test, the amount of time required to deflect the paddles under load would be measured. The dynamic test would verify FR 1 and verify the TVM portion of FR 3, requiring MCB control over the TVM.

The test stand consists of two layers of sleds with perpendicular sliders which apply force to load cells. Upon firing the engine, the team would receive readings regarding the thrust being applied in the axial (body-X axis) and perpendicular directions. With these data, the angle at which the engine's thrust was acting would have been calculated via trigonometry:

$$\theta = \tan^{-1}\left(\frac{F_{perp}}{F_{axial}}\right) \tag{2}$$

This is, of course, an idealized calculation that takes the load cell readings as being completely accurate. A number of factors could skew the data and make it inaccurate. The first and least concerning of these factors is the offset angle of the load cells. Ideally, the axial load cell would be perfectly aligned with the engine's thrust direction and the attitude of the perpendicular load cell would be exactly 90 degrees away. The angle between the load cells was verified to be accurate to within a half degree on the test stand modification, and the axial direction was aligned with that of the unmodified test stand during manufacturing. As such, the error from the attitude of the load cells was expected to be minimal. It is, however, a factor that would have been considered when interpreting the data from the modified test stand. A second factor which would have likely been more severe is the friction present in the test stand. Ideally the steel rods on each level of the sled would be perfectly perpendicular. Due to manufacturing tolerances, however, this would not realistically be the case. Any deviation here would apply pressure to the pillow block from the rods and introduce friction into the system which would have compromised the accuracy of the load cell readings. The team intended to perform an analysis on the friction in the test stand to help bound the error in the load cell readings, but was unable to do so before the shutdown. This analysis would have consisted of applying a known force to the load cells through the test stand, and comparing the resulting readings to the known true value. Performing this test with a variety of forces and orientations could have helped WHiMPS establish a percent error caused by friction. In order to perform this test, however, the team would have also needed to bound the error in the load cells themselves. This would have been another source of error that would have contributed to the uncertainty in the thrust vectoring angle. The team experienced some confusion regarding some of the load cell specifications, including their error, due to multiple data sheets being found for the same part number. This anomaly was discovered after the shutdown when separate team members discovered they had been performing their calculations with different variables, so WHiMPS was not able to contact the manufacturer to determine which data sheet was accurate. Additionally, initial load cell testing showed that the error in the readings was higher than expected. As such, further testing was planned to determine the source of this error. Once the team quantified these errors, a total error on the thrust vectoring angle could have been calculated, and the team would have been able to gain insight into the accuracy of the test stand readings. This information would have been used to verify that the thrust vectoring mechanism was capable of deflecting thrust by the required 10 degrees.

Using the two-axis test stand, the WHiMPS can determine the thrust vector angle based on the angle of the paddle deflection. During the dynamic test, the thrust vector angle would be plotted against the paddle deflection angle. An example of the expected data from this test is shown in Fig. 63. The plot shows how

the thrust vector is not expected to be affected significantly at low paddle deflection angles, but will start to increase almost linearly after the paddle deflection angle reaches about 10°.

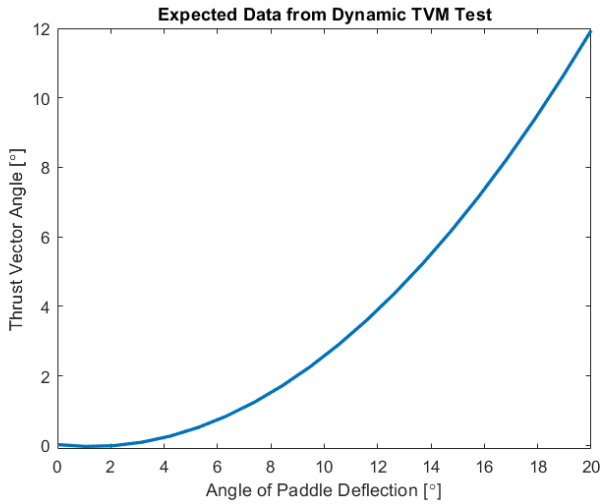


Figure 63: Expected Data from Dynamic TVM Test

5.7. Levels of Success

5.7.1. Windmill Prevention Mechanism First and Second Level of Success

The first and second Success Criteria for the Windmill Prevention Mechanism was tested using the WP1 build configuration of the MCB software which runs on a Linux PC. Since the LabView is set to be in Software Mode, the command thread utilizes a predefined circular sequence of commands that are received from a simulated LabView. Since the MCB, TVM, and ECU are in None (or disconnected) Mode any commands for those subsystems will produce errors. Only valid commands for the WPM will be executed successfully. Table 11 from the Appendix shows the predefined circular sequence of 10 commands. Note that nine of the ten commands sent contain at least one error for the purpose of testing the command handlers validation sequence.

Figures 64 and 65 show the annotated logger messages printed to the terminal. The software uses the WP1 build configuration which puts the Windmill Prevention Mechanism in software mode while the Main Control Board, Thrust vectoring Mechanism, and Engine Control Unit are in None (disconnected) mode. Note that the Health and Status Packets were set to be collected every five seconds rather than every one second for purpose of shortening the terminal output. As shown, the program enters and begins the initialization sequences followed by the creation of the three threads. Then in parallel, the Command Thread receives the circular sequence of ten commands where one command is processed every one second while the Health and status Thread handles the collection of health and status data once every five seconds. An error logger message is printed for each of the invalid commands that are received from the LabView. Note that is error message does not mean that the MCB software is not working nominally, rather it means that it is handling the errors correctly. These logger messages also show that the health and status data is being collected correctly and the packets are created and validated successfully. Only the health and status data collected from the WPM is shown as the data collected from the rest of the subsystems is always zero. When the first and second health and status data is collected from the WPM, the solenoid has not been ejected, hence the GPIO levels are low. Additionally, the temperature measurements are set to a predefined value of 30.1 degrees Celsius.

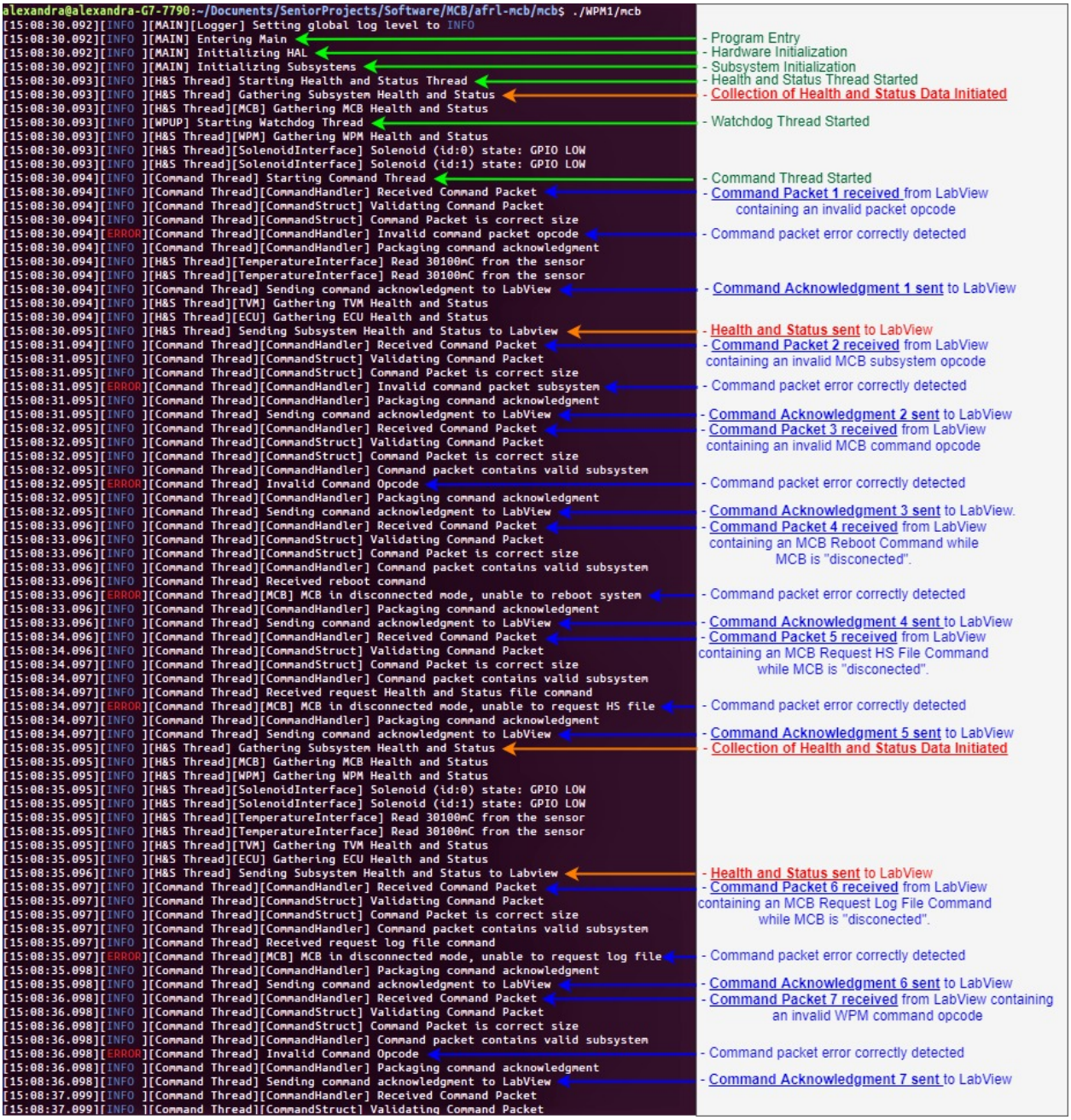


Figure 64: Annotated Screen Capture of Linux PC Terminal Verifying First and Second Level of Success for WPM

The most important verification in this test is to ensure that when a valid eject fairing command is received, the command is processed and executed successfully. As shown in Fig. 65 in teal, the MCB software was able to simulate the actuation (labeled ejection) of the two solenoids, hence simulating the ejection of the fairing. It is also shown during the collection of the next health and status data that the solenoid GPIO levels are now high after the the ejection command is executed. This further demonstrates that MCB software is able so simulate the hardware interactions using the Mock GPIO Manager successfully. Note that nominally, the solenoid's GPIO level would be toggled high to low in a much quicker manner. But it was kept to high only for the purpose of demonstrating the functionality in this document. The program continues past the included logger message infinitely, where it restarts the command sequence and continues to collect health and status data.

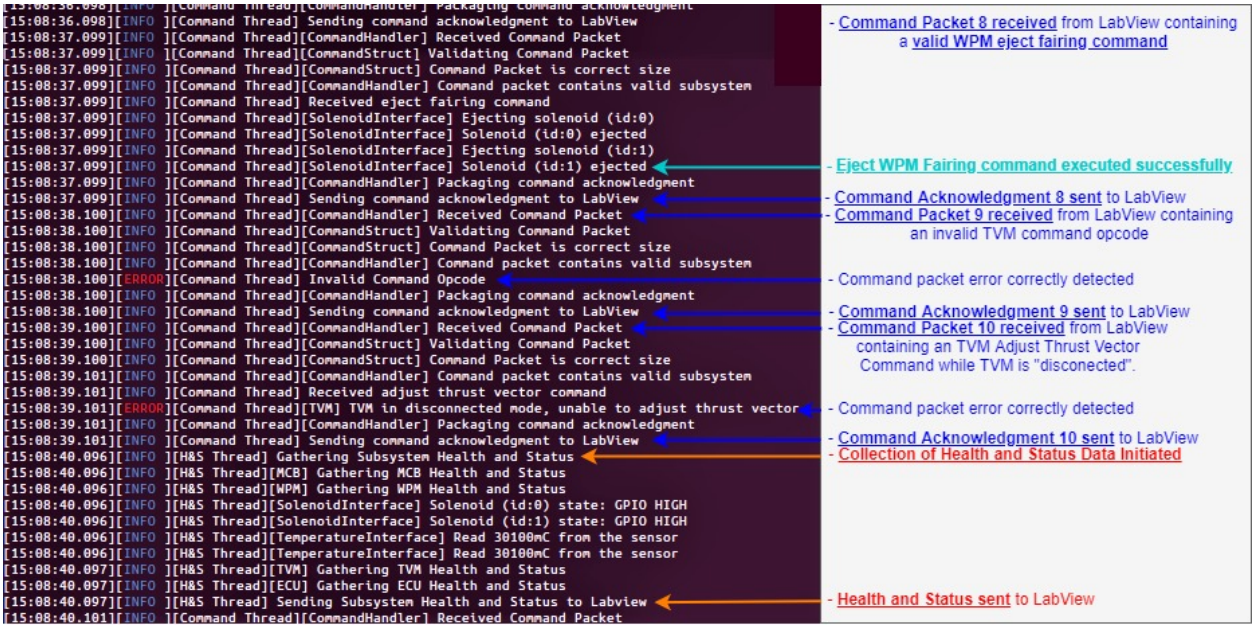


Figure 65: Continuation of Annotated Screen Capture of Linux PC Terminal Verifying the First and Second Level of Success for WPM

5.7.2. Remaining Levels of Success

As shown in the above image, only the first/second level of success for WPM was reached due to hardware limitations preventing the TVM unit testing from being completed for the first level of success in that category. The second level of success for WPM was the same as the first, and the team would have been able to reach TVM level two quickly after the first was reached as it only added a second symmetrical axis. To reach further levels of success, the team would have needed completed and integrated WPM and TVM systems integrated on the engine. At the time of the shutdown, the team was on schedule to finish these tasks within a week. Once this was completed, the aforementioned tests would have been conducted to reach levels 4 and 5 of success. Given that the team managed to operate the Jetcat P100-Rx with the dummy paddle system, there was a possibility of reaching the maximum success defined for this project on schedule.

6. Risk Assessment and Mitigation

Authors: Nick Zellmann

6.1. Windmill Prevention Mechanism

6.1.1. Risks

Some of the major risks that were identified for the Windmill Prevention Mechanism are outlined below. These risks include a variety of possible failures that could result in engine damage (resulting in an inability to reach success level 5) or system failure. Additionally, the risks associated with damaging the engine could result in the inability to complete further engine testing for the Thrust Vectoring Mechanism, thus affecting the success criteria for the TVM. A summary of the risks and mitigations for this section can be found in Fig. 66.

1. *Fairing breaks under stress*. Because of the high-speed air used to test the WPM under its proposed operation environment, the fairing could possibly break due to the stress incurred at the tip of the fairing. Additionally, the holes on the side of the fairing that enable the movement of the solenoid pins could break and render the WPM useless. This could pose an even greater problem where the debris from a broken fairing can enter the inlet of the engine, thus causing internal and possibly

irreversible damage to the engine.

2. *Air passes through the seam of the fairing*. The purpose of the fairing design for the WPM is to redirect all of the incoming airflow away from the inlet of the engine. If the fairing-halves do not perfectly fit together, the air can enter the inlet of the engine through any available gap. Thus, the fairing-halves must completely seal any gaps that would allow enough air into the inlet of the engine and permit fan blade rotation.
3. *Poor ejection trajectory.* With the many electronics on-board the engine, a poor ejection trajectory could damage a multitude of components. Additionally, the spring being mounted to the inside of the fairing and pushing off of the starter motor could pose issues if the spring damages the engine upon ejection.
4. *Other risks.* Solenoid cannot pull pins, solenoid throw not long enough, and the fore retention ring does not fit over screws on the engine's body.

6.1.2. Mitigation

1. *Fairing breaks under stress*. The mitigation for this issue comes down to simulations and material selection. FEA analysis was performed to measure the minimum requirements for the fairing to successfully operate under the Mach 0.8 flow at an altitude of 20k feet. After this, material selection was the best way to ensure that the fairing would not break under stress. Carbon fiber reinforced Onyx was selected as the material due to its strength, ease of manufacturing, and availability through CU resources.
2. *Air passes through the seam of the fairing*. To mitigate the possibility of air passing through the seam between the two halves of the fairing, an o-ring seal was taken into consideration. This would only have been implemented if the fairing did not successfully redirect the air around the engine body; thus, the team did not implement this onto the fairing halves.
3. *Poor ejection trajectory.* The main way to ensure a proper ejection trajectory without damaging on-board electronics is to test under static conditions. Airflow going over the front of the engine would actually assist in proper ejection due to the drag of the separating fairing halves pulling them safely away. Doing this test under static conditions permits observation of how the ejection would occur without damaging components.
4. *Other risks.* Obtaining stronger solenoids and extending the solenoid's throw can mitigate the issues of not being able to successfully pull the pins out of the holes. Additionally, filing down the areas on the fore retention ring that are blocked by screws on the body of the engine could mitigate the issue of the fore retention ring not fitting on the engine body. Another solution is to slightly redesign the location of the screw-holes on the fore retention ring such that there is no restriction between the screw holes on the engine and the fore retention ring.

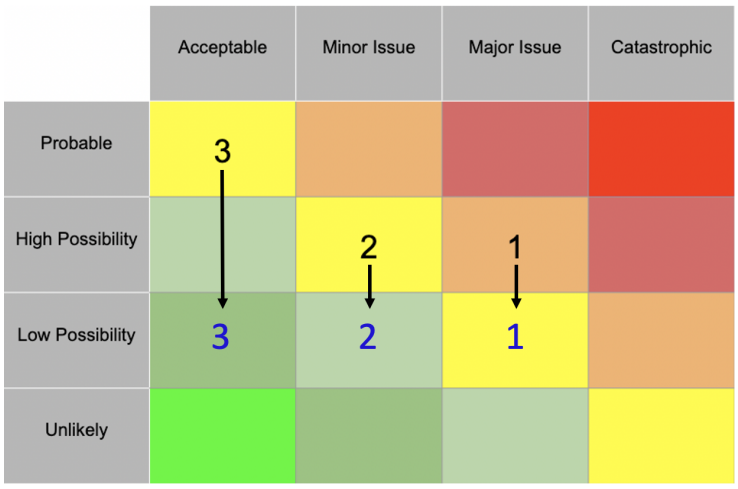


Figure 66: Windmill Prevention Mechanism Risks and Mitigations

6.1.3. Results

The risks in Fig. 66 are shown before mitigation in black, and after mitigation in blue. The risk of the fairing breaking under the stress experienced during the replication of Mach 0.8 flow at 20k feet has a very low chance of breaking. Though our high-speed air test was not conducted due to the COVID-19 shutdown, modelling and simulations showed that the fairing halves supported each other, thus decreasing the likelihood of breaking. Additionally, the APOP replica test that was performed before the shutdown proved that the air did pass around the engine body and did not allow any fan blade rotation. The ejection trajectory was not tested before the shutdown, but the probability of the trajectory impacting the on-board electronics was reduced after revision B of the ICUP was implemented. This revision kept the electronics closer to the engine body and reduced the possibility that there would be a collision between the two during testing. Finally, moving the location of the screw holes on the finalized version of the fore retention ring was in progress during the shut down. This would have ensured that the fore retention ring fit over the engine body without having to sand it down.

6.2. Thrust Vectoring Mechanism

6.2.1. Risks

Some of the major risks that were identified for the Thrust Vectoring Mechanism are outlined below. These risks include a variety of possible failures that could result in engine damage (resulting in an inability to reach success level 5) or system failure. Additionally, the risks associated with damaging the engine could result in the inability to complete further engine testing for the Thrust Vectoring Mechanism, thus affecting the success criteria for the TVM. A summary of the risks and mitigations for this section can be found in Fig. 67.

1. *Engine does not run*. If the engine does not run, verifying that the Thrust Vectoring Mechanism turns the flow by $\pm 10^{\circ}$ will be much more complex than if the two-axis test stand could measure the force on both axes and utilize trig to verify the angle of deflection. Additionally, if the engine does not run, the WHiMPS will not be able to validate whether the accepted design can withstand the extreme temperatures and forces experienced at the rear of the engine.
2. *Paddles/linkage rods break*. High stresses experienced on the paddles and linkage rods at the rear of the engine during operation increases the risk of these components breaking. If these components break during testing, lots of time and resources will be lost. Most likely, if this happens, the TVM will be unsuccessful.
3. *Foreign object intake*. This problem will most likely be caused by any article or substance that flies

into the inlet of the engine during operation. This would cause serious and irreparable damage to the engine.

4. *Other risks.* Actuator ring is not secure to the engine body and engine fire.

6.2.2. Mitigation

1. *Engine does not run*. Create a WPM and TVM that do not rely on the engine to be operable. Thus, all power systems are independent of the engine's. Additionally, the TVM is the only mechanism that needs the engine for testing; therefore, simulation and modelling of the exhaust flow can be used to validate the TVM.
2. *Paddles/linkage rods break*. Stress analyses were performed to better-comprehend the forces and vibrations at the engine's exhaust. From here, the material used for the paddles and linkage rods could maintain a factor of safety (FOS) during the engine's operation and thus reduce the possibility of a failure.
3. *Foreign object intake*. All engine testing will be performed in the CU Aerospace test cell under direct supervision of a PAB. To prevent foreign objects from entering the inlet of the engine during operation, all loose articles will be cleared from the test cell.
4. *Other risks.* An engine fire is possible with any type of jet engine. To mitigate the possible fallout of such a fire, the WHiMPS team kept a safe distance from the engine during operation and had a fire extinguisher in the test cell in case of emergency. During engine testing, only one person would be in the test cell and only for a short period of time. This person's job was to start the engine via the GSU. Once getting the engine started, he/she would quickly return to the safety of the test cell. To mitigate the possible risk of the actuator ring not being secure on the engine body, force analyses were performed to ensure that a factor of safety was implemented. Additionally, CAD modelling ensured that the component would fit into the required area. The forces are distributed between the multiple components in the TVM to reduce vibrational and structural issues.

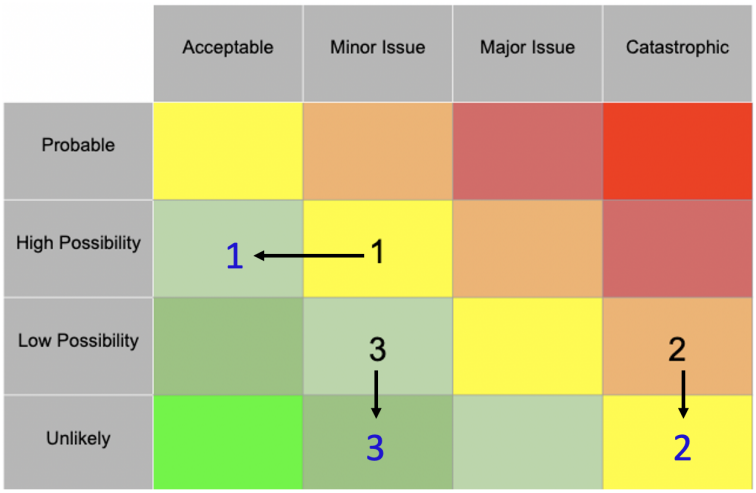


Figure 67: Thrust Vectoring Mechanism Risks and Mitigation

6.2.3. Results

As with the WPM section, the risks are shown before mitigation in black and after mitigation in blue. Running the engine had its complications, but engine runs were performed before the COVID-19 shutdown. After plugging one end of the ethernet cord into the engine body and the other into the computer, it became clear that an error had occurred and the engine would not run. JetCat explained that the wrong ends of the ethernet cord were plugged in, and they simply needed to re-flash the engine to restore its original software. Time was lost during the shipping process, but once the engine returned, another engine test was

successfully completed. The TVM was not completed at the time of the COVID shutdown, and thus could not be tested with the engine. Regardless, the paddles and linkage rods breaking was not a major concern as they were over-designed for thermal and structural purposes. The main concern for the TVM in the final week of working on this project was the amount of time that the components for the TVM took to manufacture. Team members were constantly in the machine shop working around the clock to complete these components. At the time of the shutdown, the WHiMPS were confident that completion of the project on our highest success criteria was attainable.

6.3. Electronics

6.3.1. Risks

Some of the major risks that were identified for the electronics are outlined below. These risks include a variety of possible failures that could result in electronics damage (resulting in an inability to reach success level 5) or system failure. Additionally, the risks associated with damaging the engine could result in the inability to complete further engine testing for both the WPM and TVM, thus affecting the success criteria for the project. A summary of the risks and mitigations for this section can be found in Fig. 68.

1. *On-board electronics overheat/melt*. On-board the engine sits the Integrated Communication Unit Panel (ICUP), solenoids, potentiometers, and linear actuators. The extreme temperatures that are experienced on the engine body during operation can damage (if not destroy) these electrical components and delay the project by weeks.
2. *Improper handling of electronics*. With sensitive electronics that control a jet engine and its incorporated mechanisms, it is relatively easy to touch something that causes irreparable damage. Additionally, there are many ports and different boards that, if plugged in incorrectly, can short and fry electronics if precautions are not taken. Damage to the electronics can delay this project by weeks.
3. *Vibration causing connectors to break*. The possible vibrational effects found during engine operation can cause connectors to break or separate. This would result in losing data, communication, and control of the incorporated mechanisms.
4. *Other risks.* PCB failure, noise and errors from board integration.

6.3.2. Mitigation

1. *On-board electronics overheat/melt*. To reduce the possibility of this occurring, thermal insulation can be used to reduce the thermal impact on the electrical components during testing. Additionally, thermal models were completed that predicted the temperature of PCB's being below their thermal limit.
2. *Improper handling of electronics*. Since there are twelve team members on this project, miscommunications are bound to occur. One team member may not know that something is plugged in, or that they need an ESD bracelet before touching the electronics. Taking the possible human error out of the equation is the best mitigation. Casings for the electronics boards can prevent just anyone from touching them.
3. *Vibration causing connectors to break*. Locking connectors and adhesives on the engine or actuator ring to hold down the wires and cables reduces the risk of the connectors disconnecting or breaking entirely. Obtaining vibrational data from accelerometers to characterize how serious the vibration is can assist in preparing for such effects.
4. *Other risks.* To reduce the risk of PCB failure due to poor manufacturing, an iterative approach to population testing can take place. Additionally, to mitigate the risk of noise and error affecting data and communication due to board integration, wire wrap shielding, Wheatstone bridge circuits, and short trace lengths can be utilized.

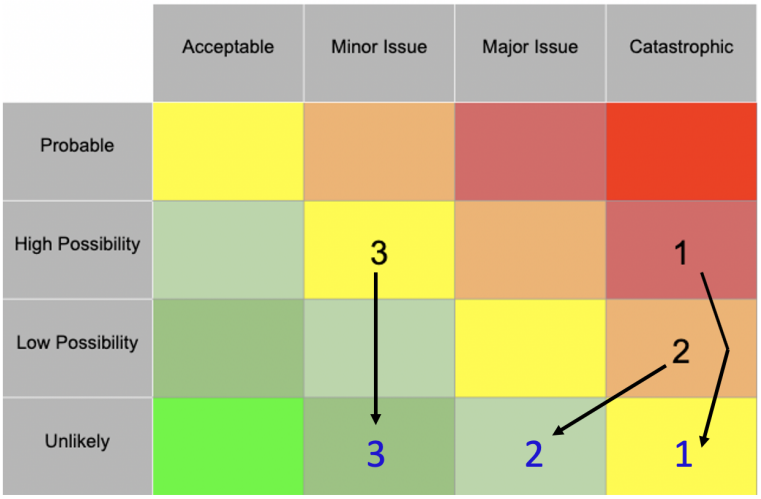


Figure 68: Electronics Risks and Mitigations

6.3.3. Results

Before the shutdown occurred, plans were in place to characterize the vibrational effects found on the engine during operation. Because of the expected high-vibrational state of the engine, finding the correct accelerometer to use was in progress. The main concern for the electronics was the thermal effects from the engine body. Once again, before the shutdown, mitigations were being considered on how the electronics could be shielded from the heat produced by the engine. Nothing was decided on, but this was the main concern of the project at the time of the shutdown.

7. Project Planning

Authors: Alec Bosshart, Andrew Robins, Andrew Meikle

7.1. Team Organization

The WHiMPS team was structured in the form of 12 technical leads forming a management team and 4 major technical teams as shown in Figure 69. The organization of the sub-teams was adjusted in the spring semester to better fit the goals of the team going forward. These changes were to eliminate the analysis and mechanical teams to create the manufacturing teams, which are similar in purpose but contain different members. This change was made to specify what each team would be responsible for.

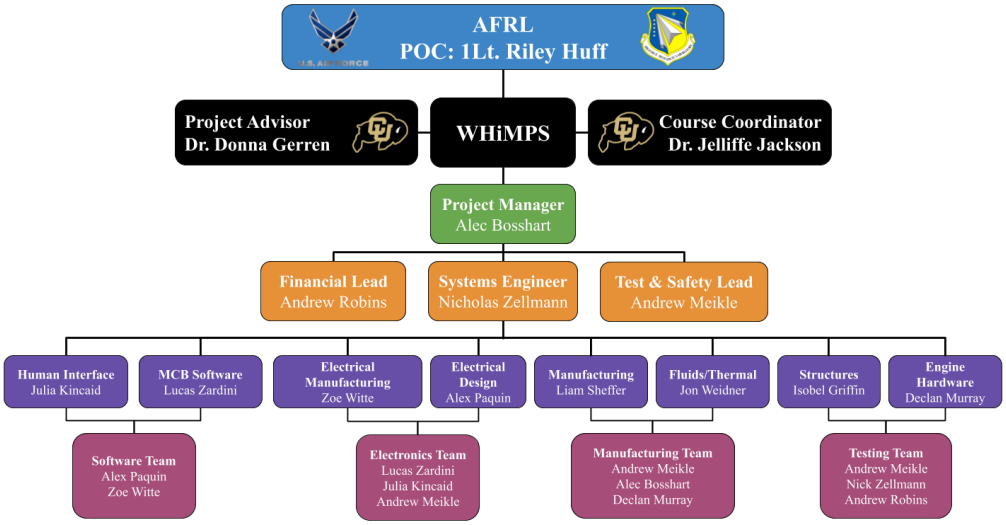


Figure 69: WHiMPS Organizational Structure

7.1.1. Management Team

The management team is composed of the project manager, the systems engineer, the financial lead, and the test & safety lead. This team is responsible for the high level management and organization of the project as a whole. The project manager is responsible for the scheduling of all activities, planning meetings, and adjusting course when unexpected problems occur. The systems engineer maintains responsibility for all technical aspects of the project, which includes the definition of requirements. Budgeting and acquisitions are carried out by the finance lead to ensure that the team is able to purchase necessary materials and components to complete the project. The test & safety lead aids the project manager in scheduling tests, along with planning specifics about how each test will be conducted. The test & safety works closely with the systems engineer to verify the requirements set.

7.1.2. Software Team

The software team is responsible for the design and coding of all software used by the WHiMPS team. The MCB software lead takes responsibility for all software used to monitor and control electrical components on the WHiMPS system, all of which communicate with the Raspberry PI on the MCB. This software includes executing commands sent by the user interface to control the mechanical components of the WHiMPS system. The Human Interface lead is responsible for the development of the software necessary to allow users to send and receive data to and from the WHiMPS system. The human interface lead also leads the data acquisition effort to convert signals from external data acquisition systems into data that can verify requirements and validate models.

7.1.3. Electrical Team

The electrical team is responsible for the design and manufacture of the WHiMPS electronics used to control and monitor the physical components of the WHiMPS system. This includes the selection of COTS electronics and sending signals to these electronics. The electronics design lead designs custom printed circuit boards to accomplish these tasks, and the electronics manufacturing lead populates these boards. The electrical team must maintain strong communication with the software team to ensure that all data and signals are compatible with each other's subsystems.

7.1.4. Manufacturing Team

The manufacturing team is responsible for the manufacture of all custom mechanical systems designed by the WHiMPs team. These components were designed by the analysis team in the spring. The manufacturing lead and fluid/thermal analysis lead worked together to manufacture these components, primarily the

TVM components and modifying the CU test stand to allow for thrust vectoring testing. The fluid/thermal switched over to become one of the manufacturing team leads because he had the most experience in the shop and did not have much additional analysis to complete in the spring semester. This team was essential to the WHiMPS due to the complexity of the TVM mechanical system, which required the manufacture of a number of small parts.

7.1.5. Testing Team

The testing team conducts all tests to verify requirements set by the systems lead. This includes the design and manufacture of all small test rigs used by the team and determining logistics required to conduct tests. The engine hardware lead was responsible for the operation of the engine during stock and TVM testing, where the engine is operating. The test & safety lead oversaw all efforts of this team and ensured that all tests were conducted in a safe manner. Safety was of paramount importance to the WHiMPS team in particular given the hazards associated with the operation of a jet engine.

7.2. Work Breakdown Structure

The work products for the team were broken down into the four primary efforts of the team. These efforts include the electrical, software, mechanical, and testing products required to complete the WHiMPS system. Deliverables and management tasks were also tracked to keep the team on task in relation to course deliverables as well as the project. A diagram of the breakdown structure is shown in Fig. 70. The tasks in green were completed by the team, and the tasks in red were planned to be completed by the end of the semester. Due to the COVID-19 pandemic, the team was not able to complete all of the planned tasks.

Figure 70: WHiMPS Work Breakdown Structure

7.2.1. Electrical

The tasks in the electrical section are broken down by each iterative board revision. Each revision contains a design phase, a population phase, and a testing phase. WHiMPS originally planned for 3 iterations of board development, but determined that two would likely be sufficient after Revision A testing showed minimal issues. Each revision contains all 3 boards that make up the electrical system as described previously in this document. At the time of the COVID-19 shutdown, the electrical team had completed a first revision and determined the source of all issues found during Revision A testing. The design changes were completed to solve these issues, but the acquisitions process was unable to be completed.

7.2.2. Software

The development of the software tasks occurred in incremental steps. The team began by planning how data would flow through the system, and then moved into a preliminary architecture. This included a data acquisition plan and a planned structure of the MCB software functions written in C++. The MCB software required a control algorithm to relate the movement of a linear actuator to the deflection of a thrust vectoring paddle, which was developed next in coordination with the mechanical team. The Labview team developed a front panel that would serve as the primary interface between the team and the WHiMPS electrical system, including external DAQ's.

7.2.3. Mechanical

The work products for the mechanical/manufacturing team follow an incremental pattern as well for the TVM and WPM subsystems. This began with a design phase, completed in the fall, where preliminary designs were created. From there, analysis was constructed to confirm the feasibility of each system. The analysis phase was critical to the project as both developed systems would be exposed to extreme thermodynamic and aerodynamic loads during testing. After each system's design was confirmed, manufacturing began. The TVM system was manufactured using a combination of CNC machines and manual mills inside of the CU Aerospace machine shop. The WPM system was 3D printed in the CU Aerospace Rapid Prototyping lab. Complete integration of each system was not possible due to the COVID-19 shutdown.

7.2.4. Testing

Testing work products began with determining which tests would be conducted. The team settled on an incremental testing progression, beginning with a round of stock engine testing to gain familiarity with the engine and determine whether stock engine electronics would be compatible with the TVM system. From there, The team decided that the primary tests should begin with a static test and then move onto a dynamic test to minimize risk to the mechanical systems and the engine. All of these tests needed to be scheduled, designed and logistics, including safety, needed to be determined following the preliminary plans. Once logistics were determined, the manufacture of testing rigs could begin to prepare for the tests. The static and dynamic WPM and TVM tests were scheduled to occur in April, and were therefore incomplete due to the shutdown.

7.3. Work Plan

Once work products were determined, scheduling could begin. Figures 71 shows the schedule for the manufacturing and testing of the TVM and WPM systems. Figure 72 shows the schedule for the development of the electronics and software work products. A zoomed image of the critical path can be found in Figure 73.

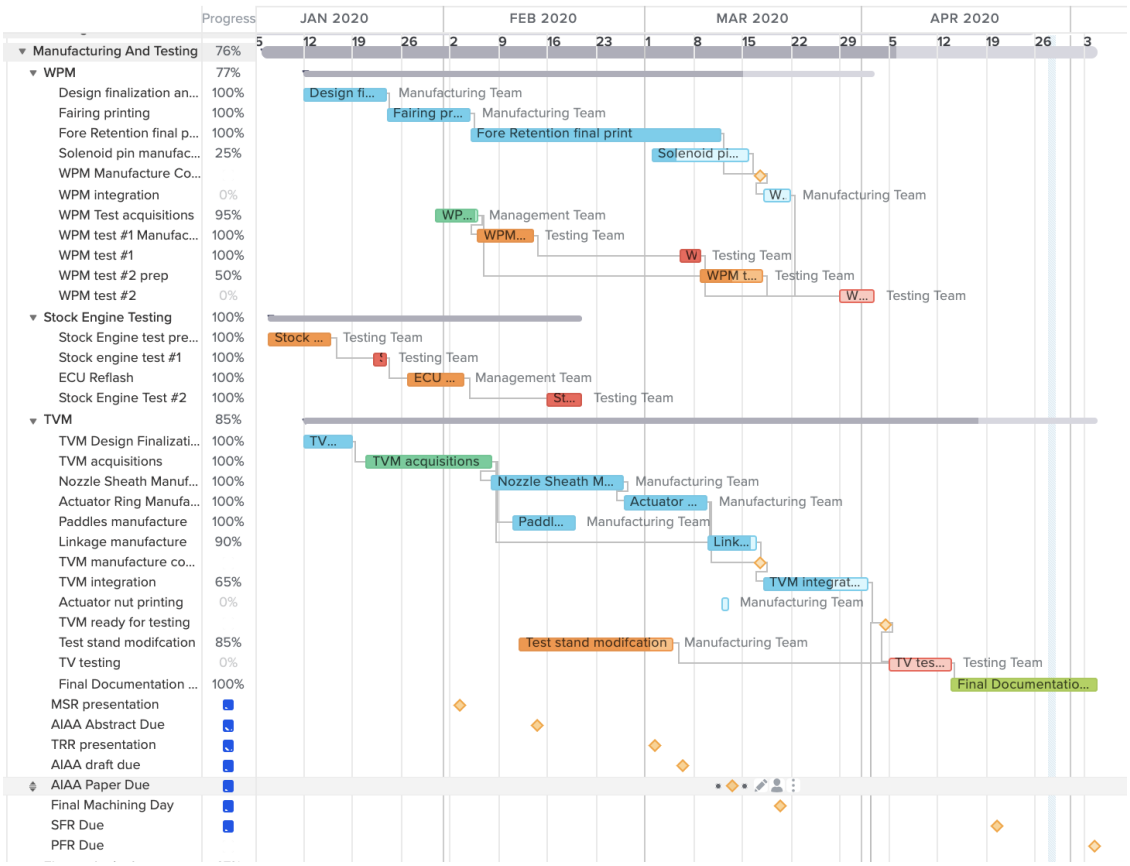


Figure 71: Mechanical Gantt Chart

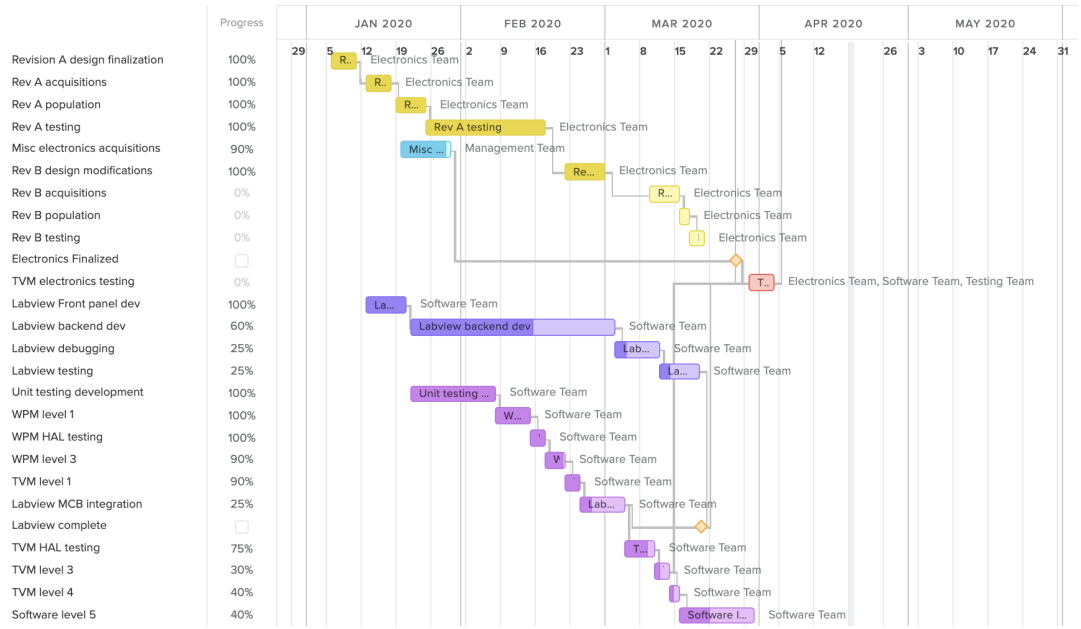


Figure 72: Software and Electronics Gantt Chart

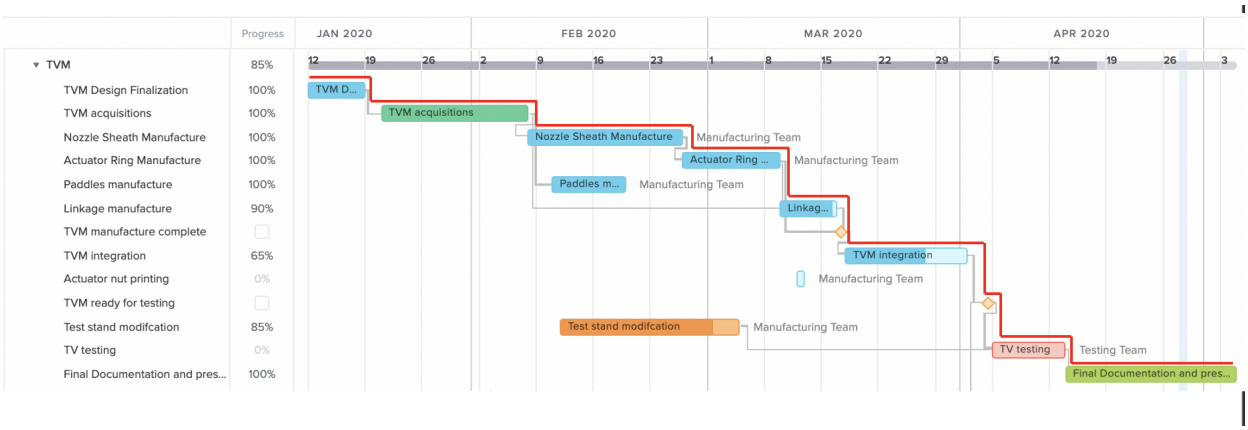


Figure 73: Critical Path

7.3.1. Schedule Margin

Margin was added into the schedule through two primary methods: over estimations and an end of contract margin method. The amount of work needed to complete each task was estimated by the technical sub-team responsible for completing it, and then a 20% addition was added to each task to provide some margin. Given the risk associated with the WHiMPS project, this margin was relatively small in the long run. To compensate for this risk, an end of contract margin method was added as well. All tasks were scheduled to occur back to back with the 20% margin built in, which left as much time as possible at the end of the project to allow flexibility. WHiMPS ended up needing this flexibility after a number of unexpected delays occurred, including a week delay in all TVM and WPM work at the start of the semester due to a hold on the budget. These delays resulted in WHiMPS pushing all major testing back into April.

7.3.2. Critical Path

The critical path of the WHiMPS system was the manufacture and testing of the TVM system. This is because the TVM system was expected to take the most time to develop and was the last testing to occur. Any delay in the TVM manufacturing would have caused a failure of delivering half of the system. As mentioned, WHiMPS experienced delays in receiving their P-card which absorbed a large amount of planned margin in the TVM development. Luckily, the manufacture of the engine attachments, the nozzle sheath and actuator ring, were completed on schedule. Other manufacturing was able to be worked on in parallel to these major items with the help from other sub-teams and nearly completed before the COVID shutdown. Following manufacturing, integration was scheduled to occur. The team managed to integrate the nozzle sheath and actuator ring along with a single paddle-linkage system. Overall, WHiMPS feels that their critical path item was on schedule, and therefore the rest of the project was feasible to complete on schedule.

7.4. Cost Plan

The WHiMPs team has two cost plans over the course of the project. At the beginning of the semester, the team planned on leaving enough money to buy an additional engine in case the engine failed. The cost of an additional engine is approximately $2500, which was left as margin at this time. However, the unexpected occurred and the team had to make numerous un-predicted purchases. These included repairing the engine for $500 and multiple $60 to $100 shipping charges. At this point, the team decided to abandon the thought of buying another engine due to budgetary limitations and time limitations. Given that the first engine took over two months to arrive, the team realized that it was impractical to save 50% of their budget to potentially buy another engine that would not arrive on time to be useful. Additionally, four levels of success could be reached without a functioning engine. Upon this decision, the team decided to utilize the extra budget in order to maximize the project effectiveness. In the following chart, Figure 74, the proposed and actual spending can be seen along with the relative margins. For a more detailed explanation please see the appendix.

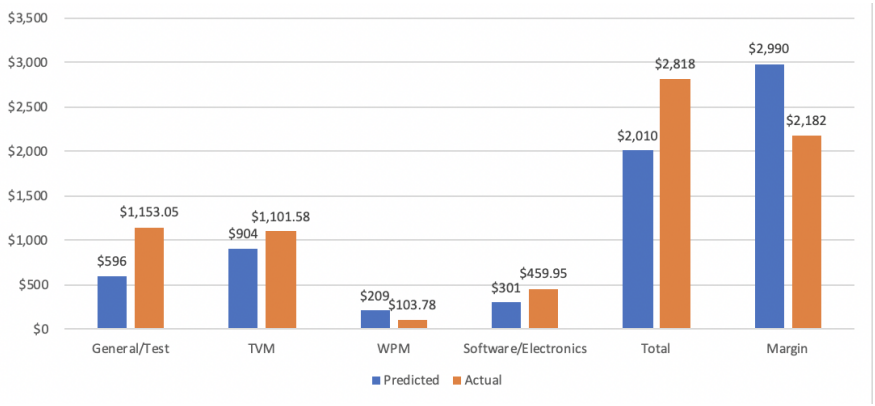


Figure 74: WHiMPS Budget

7.4.1. Budget for Major Items

From the budget chart above, it can be seen that the team spent more than initially proposed. This was for multiple reasons. The first was unanticipated expenses in the beginning of the project, in particular fixing the engine. The team encountered difficulty when we accidentally switched the direction of the fuel line. The engine had to be shipped to Maryland and fixed. The entire ordeal cost the team $660. Another unanticipated expense early on was shipping a broken engine to Wright-Patt AFB. This was at the request of the AFRL as they had sent it to us just for dimensions. That cost the team another $100 in shipping. Additional major items that needed to be purchased were the metals for the TVM. The team needed a particular alloy of stainless steel, which took a lot of trouble to find. In all the team spent approximately $550 on various metals. The linear actuators were also major purchases. Each actuator cost the team $130, coming to a total of $520 before shipping. On the electronics and software side, each revision of the circuit boards would cost the team $90 dollars. All of these major purchases were accounted for in the initial budget calculation, excluding the engine repair and shipping.

7.4.2. Budgetary Margin

While the team went over budget in every category expect Windmill Prevention, it was for good reason. After the decision to use the saved 50% on items for the project the team poured more money into R&D. This included testing materials and processes for the TVM and electronics. Before the shut down the team was sitting on a $2000 dollar margin with all major items ordered. Any remaining purchases would be less than $50 dollars and minor in terms of the budget. From the teams calculations and trajectory, the final budget would be approximately $1800 dollars.

7.5. Test Plans

Figure 75 shows the list of all planned tests with their scheduled date and completion status. The third column also shows some of the specialized equipment used for each of the tests.

Test	Date Scheduled	Procured Equipment	Completed
Stock Engine Run	2/24	Thermocouple & DAQ	✓
Dummy Paddle	2/26	Thermocouple & DAQ	✓
Thrust Baseline	3/12	Load Cell DAQ	✓
WPM APOP	3/3	APOP Test Rig	✓
WPM Static	3/30	SABRE Tanks, Laser Tachometer	✗
WPM Dynamic	4/1	SABRE Tanks, Laser Tachometer	✗
TVM Static	4/6	Load Cell DAQ	✗
TVM Dynamic	4/8	Load Cell DAQ	✗

Figure 75: WHiMPS Test Overview

The scheduling of the tests was based on the availability of the engine and the expected completion date of manufactured components. The Stock Engine Run test was originally scheduled for 2/13, but due to a connector being attached incorrectly the engine ECU was damaged and had to be sent to JetCat America for repair. This delayed all of the test dates by two weeks. The WHiMPS team started with tests that could be done with the stock engine and measurement equipment. While these tests were being completed, the WPM APOP replica test was successfully done. The rest of the WPM tests were scheduled next because the manufacturing of the WPM components and completion of WPM software was expected to be finished before the TVM components. After the WPM tests with the engine, the TVM tests would have been completed.

All of the tests were/would have been conducted in the CU Aerospace Engine Test Cell. This test cell is adjacent to the machine shop, inside of the aerospace engineering building. Coordination with Matt Rhode was required before the test cell was set up for the WHiMPS' tests and to have faculty supervision for the tests. All test requiring the engine used a test stand acquired from a previous team's project. This test stand was clamped to a test stand bench, which was bolted to the floor inside of the test cell. The static and dynamic WPM tests required a source of high speed (Mach 0.8) air to be blown at the engine. The WHiMPS team acquired the cold flow test setup from SABRE, a senior projects team in 2016-2017. Professor Farnsworth had gained custody of the equipment after the senior project team finished their work. The WHiMPS worked with Professor Farnsworth to move all of the necessary equipment (two large holding tanks and one settling tank) into the Lockheed Martin senior projects room. The electronic valves used by the team were found in a large box of miscellaneous equipment used by any team working on an AFRL jet engine project. Additional procured equipment, including DAQ systems, a laser tachometer, and thermocouples were secured to use from Trudy Schwartz.

The WHiMPS used a general safety plan to cover a majority of the potential safety issues. This safety plan includes the danger zone behind the engine which must be cleared before a test, team roles regarding safety, and procedures for accidents. In addition to the safety plan, the WHiMPS created test plans with the full procedures written out. The full safety plan and test plans for the stock engine/dummy paddle test and the APOP replica test can be found in Appendix C.

8. Lessons Learned

Authors: Isobel Griffin, Andrew Meikle

Throughout the design and manufacturing processes, WHiMPS learned many important lessons that would help future teams in their endeavors. The main and most broad lesson learned is the importance of developing, modifying, and sticking to a schedule. In the first semester, WHiMPS failed to follow a strict schedule when preparing deliverables, which reduced their quality. More specifically, set hard deadlines for all assignments which include a draft due date, a review due date, and a finalization date. Working in these increments encourages accountability and ensures there is enough time for editing. If WHiMPS had done this since day one, the quality of deliverables would have increased from the start. Additionally, editing and scheduling multiple reviews with as many PAB members as possible greatly increases the quality of work. The assignments in this course are detailed and tedious, but setting aside a team and a time for editing and reviews will alleviate the stress of the project. WHiMPS had an editing team that would review the first draft of every assignment, followed by a review from our Project Advisor. This drastically improved the quality of our work and proved effective.

There were several pieces of information WHiMPS would have benefited from if they had been provided from the beginning. For next year's team, it is important to know that the JetCat P100-RX is more reliable than micro jet engines used in the past, thus the engine electronics do not need to be redesigned or manufactured. WHiMPS spent a lot of time debating whether to use the SPECS ECU, so this would have saved an immense amount of discussion time. If next years' team has to order an engine from JetCat, WHiMPS would encourage them to determine where to acquire funding for the engine and ordering as soon as possible. WHiMPS spent many weeks in communication with the aerospace department and the AFRL to determine where the funding for an engine would be found. Once this problem was solved, there was a 2 month delay in receiving the engine from the distributor. If WHiMPS had been able to order the engine from the first day, it would have prevented delays in design and manufacturing. Having the engine in hand during the design phase allows the team to more accurately determine dimensions and gain experience with the engine during the first half of the senior design course, which aids in the design process.

In regards to design, the most important lesson learned was the value of models, both CFD and FEA. These models should be created in the design phase and validated in the test phase. WHiMPS developed several models, including stress and temperature of the paddles. If WHiMPS had created a more wide variety of models in the beginning, it would have helped our second semester presentations. After testing was complete, WHiMPS presented the data findings during our oral report, but we soon realized that our lack of models failed to prove that anything had been validated at all. For example, we failed to produce a vibrational model in the design phase, thus proving that the vibrations were expected was not possible. For any future teams, WHiMPS highly recommends that as many models as possible are created to ensure the project success can be truly articulated through testing and verification.

In terms of testing, the biggest lesson learned is to ensure that all tests designed by those external of the team are fully vetted before assuming they will work. WHiMPS took the AFRL's word that the windmill prevention test was easy, feasible, and worked on all designs. Because of this, WHiMPS did not take enough time to critically analyze and determine how and if the test would work. This led to a major failure during the second semester which could have easily been avoided. When developing any test, ensure all measurements, principles, and steps have been carefully reviewed and are feasible. Another key lesson from testing is the importance of a good test plan. When test plans clearly outline personnel, objectives, test, and most importantly safety, time is saved and there are few hiccups. When we presented a clear test plan, we were able to gain the trust of the department, especially Matt Rhode, which made testing quick and efficient for the remainder of the semester. Finally, it is paramount to triple check the connections between engine components before powering anything. The WHiMPS accidentally flipped a connector so it was inserted backwards, which ruined the ECU when it was powered on. The connector between the fuel pump and the ECU (square connector with four square holes) has a direction, but it is incredibly hard to tell which direction is correct. The connector has a small notch on two of the sides, which shows the correct way to

insert the connector. However, do not trust the feel of the connector being inserted because it will easily connect regardless of the direction inserted - the JetCat America service technician confirmed that this is a very common problem. Flipping the connector required the team to send the ECU and the engine back to JetCat America for repair, costing the team $600 and two weeks. If next year's team is not using the fuel pump from the WHiMPS, make sure to determine the correct direction for the connector and mark the fuel pump and the connector so it is very easy to tell if the connector is in the correct orientation.

Finally, WHiMPS learned the importance of being able to respond to problems as they occur. Nothing will ever go exactly as it is planned, and being able to respond to issues and adjust course is critical. As mentioned, WHiMPS experienced extreme unexpected delays in acquiring an engine, repairing the engine, and receiving access to a purchase card. All of these events were detrimental to the project, but the team was able to overcome these challenges. When something occurs, it is important to keep working on what is possible outside of the issue and prepare to recover once the delay is completed. It is easy to become discouraged when tests go wrong, but accomplishes nothing. It is always possible to recover, and advisors can help with this process.

9. Individual Report Contributions

- **Alec Bosshart:** Project Deliverables, Conceptual Design Summaries, Integration Plan, Remaining Levels of Success, Team Organization, Work Breakdown Structure, Work Plan, Editing
- **Isobel Griffin:** Project Purpose, Lessons Learned, Editing
- **Julia Kincaid:** Verification and Validation, Editing
- **Andrew Meikle:** Helped with: Thrust Vectoring Design Outcome, Thrust Vectoring Mechanism Verification, Lessons Learned, Editing; Wrote sections: WHiMPS Engine Function Verification, Windmill Prevention Mechanism Verification, Test Plans
- **Declan Murray:** Design Process and Outcome (WPM Components), Manufacturing (WPM Components)
- **Alexandra Paquin:** Main Control Board Software Design, Main Control Board Software Manufacturing, Main Control Board Software Verification, Levels of Success, and Main Control Board Appendix Content, Editing
- **Andrew Robins:** Cost Plan, Budget Breakdown, Editing
- **Liam Sheffer:** Thrust Vectoring Mechanism Scope of Manufacturing, Test Stand Modification, Thrust Vectoring Mechanism Verification
- **Jon Weidner:** Design Process and Outcome (TVM Components), Manufacturing (TVM Components)
- **Zoe Witte:** Electronics Verification and Validation, Electronics Manufacturing Challenges, Helped with Electronics Design
- **Lucas Zardini:** Electronics Verification, Validation, design, and manufacturing
- **Nicholas Zellmann:** Levels of Success, Proposed Mission CONOPPS, Testing CONOPS, Functional Block Diagrams, Functional Requirements, Design Outcomes, Risk Assessment and Mitigation

References

[1] Von Groll, Gotz. “Windmilling in Aero-Engines”, Imperial College of Science, Technology& Medicine , Retrieved September 8, 2019, from https://www.imperial.ac.uk/.

[2] Quellwerke, “P100-RX,” JetCat Available: https://www.jetcat.de/en/productdetails/produkte/jetcat

[3] V-2 Jet Vane Motor Available: http://afspacemuseum.org/displays/V2jetvaneMotor/.

[4] *Rocket Propulsion Elements, 9th Ed.*, G. Sutton and O. Biblarz, Wiley Publishing, 2017.

[5] Kumar, N., “Co-flow Fluidic Thrust Vectoring Technique,” ResearchGate Available: https://www.researchgate.net/.

[6] “BeagleBone Blue,” DEV-14920 - SparkFun Electronics Available: https://www.sparkfun.com/.

[7] “BeagleBone Black development board with 1GHz AM335x ARM® Cortex-A8 processor,” BeagleBone Black development board with 1GHz AM335x ARM CortexA8 processor Version History Available: https://www.element14.com/.

[8] “Raspberry Pi 4,” Pimoroni Make Tech Treasure For Tinkerers Available: https://shop.pimoroni.com/products/raspberry-pi-4.

[9] “New Pi Zero WH launched,” The MagPi Magazine Available: https://www.raspberrypi.org/.

[10] “NVIDIA Jetson TX2: High Performance AI at the Edge,” NVIDIA Available: https://www.nvidia.com/en-us/.

[11] Pelouch, James, et al. “Thrust Vector Control (TVS) System Study Program.” Nasa.gov/Archive, ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19700027043.pdf.

[12] ”Engine Thrust Vector Nozzles,” Available: http://geekchicpro.com/2018/05/engine-thrust-vector-nozzles/

[13] Binder, Nicolas, and Xavier Carbonneau. “Performance of a Thrust-Vectoring Solution for Unmanned Air Vehicles.” Journal of Propulsion and Power, vol. 28, no. 5, 2012, pp. 1125–1129., doi:10.2514/1.b34421.

[14] Andersson, Mattias and Kordian Goetz, ”FE analysis of a dog clutch for trucks with all-wheel-drive.” Available: https://www.diva-portal.org/smash/get/diva2:634202/FULLTEXT01.pdf

[15] https://www.nasa.gov/centers/armstrong/news/FactSheets/FS-009-DFRC.html

10. Appendices

10.1. Appendix A: Conceptual Designs

10.1.1. Windmill Prevention

Engine Cover To prevent windmilling created by the free stream air passing through the engine, the WHiMPS conceptualized an engine cover that prevents the free stream air from entering the engine while it is turned off. This is demonstrated in Figure 76. A method of removing a cover design is shown in Figure 77.

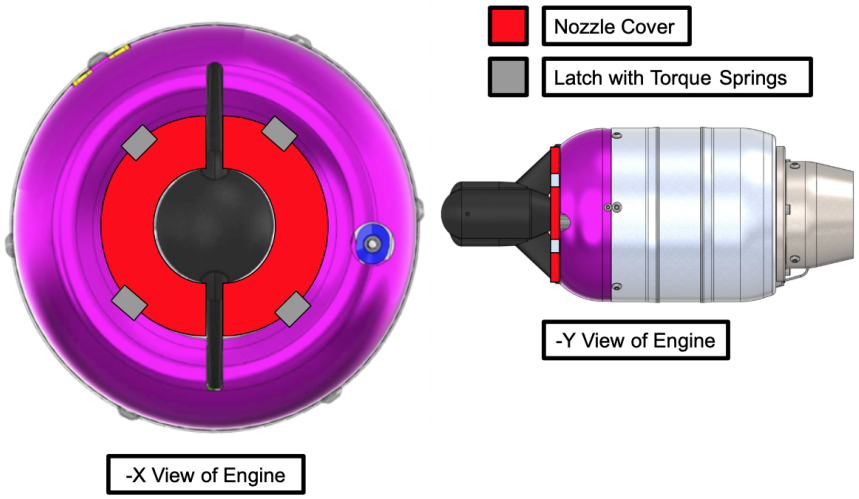


Figure 76: Airflow Engine Cover on JetCat P100-RX[2]

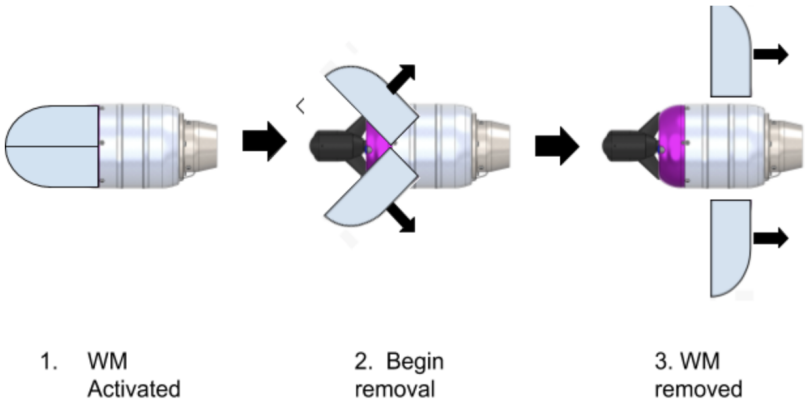


Figure 77: Cover Ejection Sketch

If the airflow can be prevented from entering the engine, there will be no torques induced on the fan blades, thus preventing windmilling. While the cover itself is simple, the ejection or retraction of such is a more complex mechanism to consider. The ejection method is demonstrated in Figure 77, and is similar to a rocket fairing splitting.

Pros	Cons
Mechanically simple construction	Difficult to eject or retract safely
No internal engine components required	Lack of history, lack of research and information
No airflow blockage post-ejection	Ejecting any item is a safety concern

Hooks A hook mechanism, shown in Figure 78, involves a rack and pinion system that allows for one-axis movement of a metal rod with a small hook on the end. The rod with the hook on the end would be pulled back toward the nozzle of the engine. In doing so, the hook would slide between the compressor blades, preventing them from rotating.

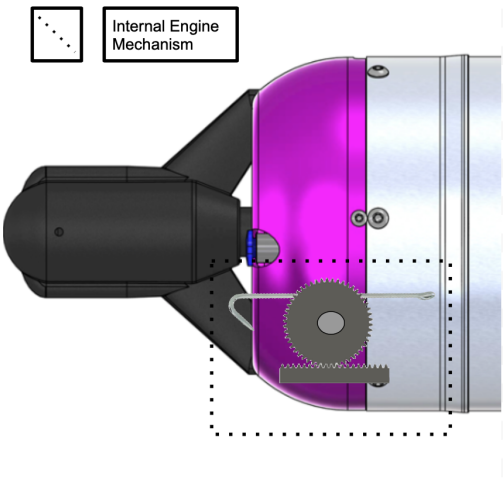


Figure 78: Hook Attached to Rack and Pinion on Engine Nozzle [2]

The hook would be controlled by the user interface and would move via the rack and pinion. This mechanism is simple in the fact that it requires basic mechanical hardware; however, preventing the fan from moving under the strength of Mach 0.8 flow could require a lot of force. Because the system is experiencing such a large force, the potential of damaging the fan blades is greatly increased. Also, the JetCat P100-RX is known to be extremely sensitive to mass distribution and damaging the fan could potentially affect this.

Pros	Cons
Simple mechanical structure and assembly	Potential to damage the engine fan
Rack and pinions are commercially available parts	Potential to offset mass distribution

Sealed Notches Another way of physically impeding the compression fans is the Sealed Notch method. A sketch of this can be viewed in Figure 79 and 80.

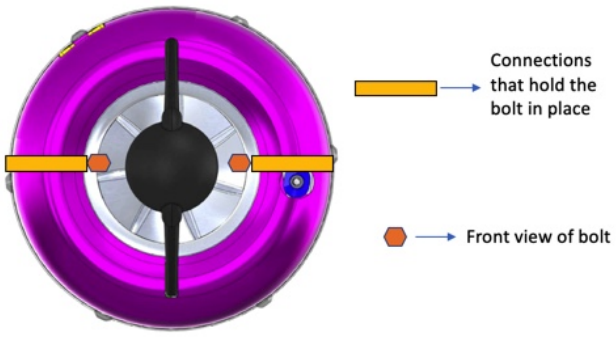


Figure 79: Front View of Notch Attachment [2]

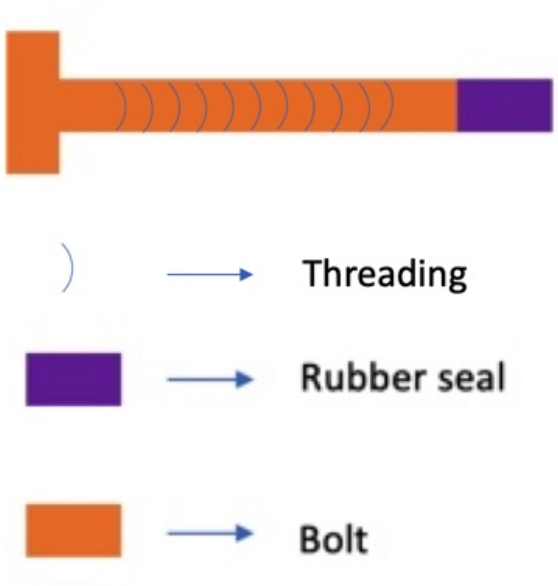


Figure 80: Sealed Notch[2]

The Sealed Notch mechanism illustrates a way to physically prohibit fan blade movement. The mechanism shall consist of two bolts that have a rubber sealing on the ends to provide a buffer between the sensitive fan blades and the bolt. The bolts, when not prohibiting windmilling (engine running), shall be fully unscrewed such that the rubber ends of the bolts do not interact with the compression fan blades. When the mechanism is to perform its windmill prevention, a signal from the MCB will be sent to servos that spin the bolts clockwise, thus pushing the bolt's rubber tips in between the fan blades. This mechanism is simple due to the very few moving parts, but encounters setbacks due to its functionality. Since the bolts would be screwed in between the fan blades, the blades would have to be positioned such that the bolts do not push directly into the blades. Additionally, the JetCat P100-RX engine is known to be extremely sensitive to mass distribution and damaging the fan could be a detrimental risk.

Pros	**Cons**
Mechanically simple (i.e only two moving parts)	Potential damage to fan blades
Cost efficient	Potential to offset mass distribution

Dog Clutch A Dog Clutch, Figure 81, is another design option for preventing windmilling of the engine. This would be implemented in the interior of the engine and consist of two plates that can be connected and disconnected. The first plate would be fixed to a non-rotating part of the engine (the fixed plate), while the second plate would be fixed to the shaft of the turbine (the turbine plate). To prevent windmilling, the fixed plated would be moved by a linear actuator to connect with the turbine plate. This would then lock the turbine plate to the fixed plate and prevent the turbine from moving.

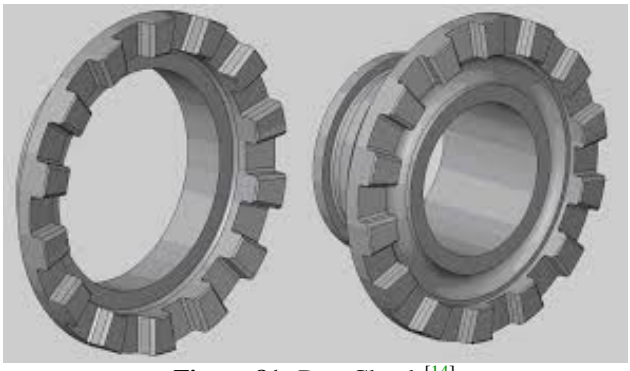


Figure 81: Dog Clutch[14]

Pros	Cons
Proven technology that is commonly used in cars	Must alter the interior of the engine
Will not harm the turbine blades	Most likely will impede airflow within the engine

10.1.2. Thrust Vectoring

Jet Vanes Implementing jet vanes aft of the engine nozzle is a possible method of providing thrust vectoring capabilities. Figures 82 and 83 below provide a visual demonstration of jet vane technology.

Figure 82: Jet Vane Nozzle Mechanism[3]

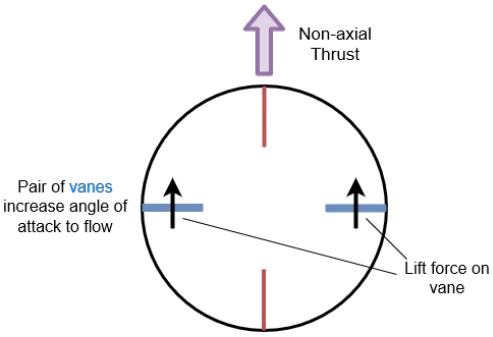


Figure 83: Jet Vane Diagram

Jet vanes are essentially a set of four symmetric airfoils plunged into the jet exhaust flow with the chord aligned with the engine central axis and attached to the nozzle in either a cross or plus formation. This technology has a lot of flight heritage with rockets and missiles but not with jet engines[4]. To accomplish 10° of thrust vectoring, a certain amount of the thrust force must be directed in the desired direction of vectoring. Assuming the engine is meant to induce a pitch-up, the non-axial component of thrust would need to be in the +Z direction. The magnitude of this component is derived with trigonometry in Eq.(3)

$$\frac{F_z}{F} = \sin 10^\circ \tag{3}$$

Thus, the non-axial component must be 17.36% of the total thrust. In the 100 N nominal case of the JetCat P100-RX, this corresponds to 17.36 N redirected normal to the X-axis. This requirement applies to all thrust vectoring technologies. For jet vanes, assuming two vanes can produce lift vectors parallel to the non-axial thrust component vector (illustrated in Figure (83)), the lift force produced by the two vanes is 1/2 the required thrust component, or 8.68 N.

Pros	Cons
Proven technology	Prone to erosion
Simple to analyze	Thrust loss up to 3%

Flex-Nozzle The Flex-Nozzle is comprised of several stacked hoops, each with the same diameter, that can take the shape of a right-angled cylinder or an oblique cylinder. Four rods placed 90 degrees apart are attached to each of the stacked hoops. Each rod is interfaced with a combined linear and two-axis rotary actuator. The Flex-Nozzle will attach to the end of the engine's nozzle. The stacked hoops are restricted to only move along the horizontal plane, so the height of the Flex-Nozzle, h, will remain constant. The goal for this design is to incorporate as many stacked hoops as possible for a given height, with the purpose of smoothing out the interior of the Flex-Nozzle as much as possible. Additionally, any friction between hoops when sliding in horizontal direction will need to be minimized while the hoops must also be held together in the vertical direction such that the Flex-Nozzle is air tight.

Figure 84 shows how the Flex-Nozzle will change shape from a right cylinder (θ=90 degrees) to an oblique cylinder (θ<90 degrees) along one axes. To change the thrust vector, the actuators will simultaneously apply a torque to each rod as well a elongate each rod (to account for the new hypotenuse).

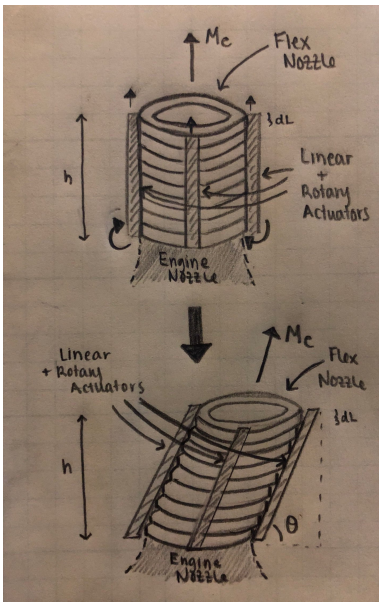


Figure 84: Flex Nozzle

Pros	Cons
Extremely accurate thrust vectoring	Very involved mechanical system (lots of moving parts)
Simple trigonometry required for control	Interior of nozzle is not perfectly smooth - creates stagnation points
Zero thrust impedance at zero degrees of vectoring	Large increase in form factor of the engine

Fluidic Thrust Vectoring Fluidic thrust vectoring is another possible thrust vectoring technology. A sketch showing the principles behind fluidic thrust vectoring can be seen in Figure 85 below.

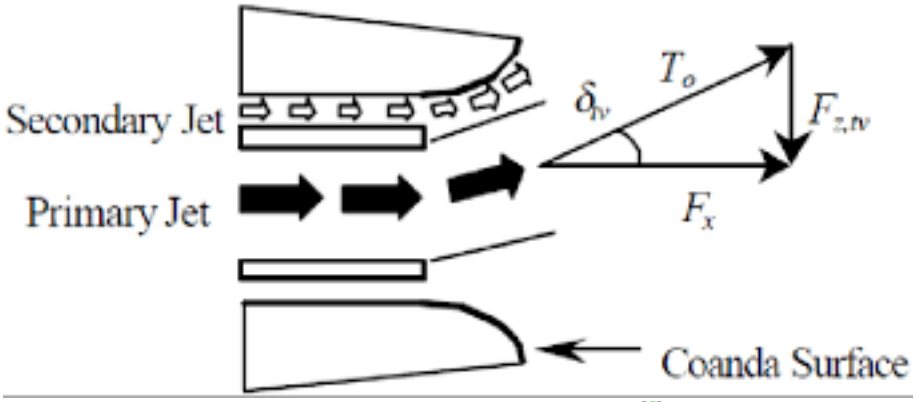


Figure 85: Fluidic Thrust Vectoring[5]

Fluidic thrust vectoring works by injecting a secondary stream of air next to the primary exhaust of the engine. The secondary stream goes over a Coandă surface (a specially curved surface which encourages attachment of flow), which pulls the secondary flow in the direction of the surface. A pressure difference between the sides of the primary exhaust is generated, which pulls the main exhaust stream in the direction of the secondary stream. As seen in Figure 85, this effect is able to vector the thrust of a jet engine.

Pros	**Cons**
No moving parts	Difficult to retrofit on engines
Small form factor	Requires source of pressurized air, method to control flow
Nothing impedes primary jet exhaust	Difficult to analyze

Jetavator A Jetavator is a possible method of thrust vectoring. Essentially, it is an external nozzle placed into the flow and is manipulated within the flow to allow for thrust vectoring. A diagram of this can be seen in Figure 86 below.

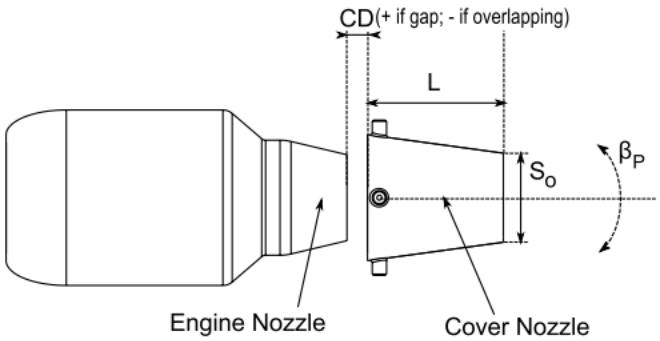


Figure 86: Diagram of a Jetavator on a small UAV engine[13]

A Jetavator is an external ring or nozzle section that is attached outside of the conventional nozzle and is maneuvered within the exhaust flow to allow thrust vectoring capabilities. The Jetavator will be mounted externally on the engine, and attached to four rods which are driven by a linear actuation mechanism that will allow thrust vectoring capability in both axes. This technology is similar to a gimballed rocket nozzle, but is capable of being introduced to the design after production in a ”bolt-on” method. Due to these similarities, there is previous research and applications available for consultation. Preliminary research shows that

Jetavators are effective mechanisms on small scales; NASA determined that they become inefficient above a diameter of 260 inches due to a poor weight scaling with size[13]. However, the WHiMPS thrust vectoring mechanism will be applied on a small scale, making Jetavators a strong candidate for consideration. At zero deflection, the Jetavator will not impede nominal thrust of the Jetcat P100-RX, and even has the possibility of increasing thrust. University of Colorado's previous APOP team, SPECS, has advised that the Jetcat engines have slightly over expanded nozzles, creating the possibility of increasing nominal thrust using a Jetavator. However, WHiMPS has not conducted any studies to verify this advice.

Pros	Cons
Zero thrust impedance at zero degrees of vectoring	Relatively large and mass inefficient compared to other designs
Simple design that relies on 4 motors/actuators	Potential of affecting the exhaust flow, causing engine to shut off due to safety features
Potential for two axis thrust vectoring	

Cruciform Nozzle A cruciform mechanism provides thrust vectoring in a similar manner to jet vanes. This is illustrated below in Figure 87.

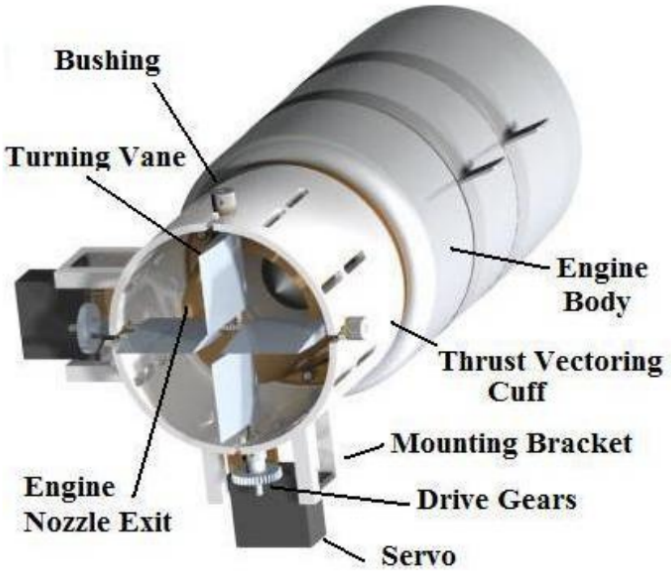


Figure 87: Cruciform Thrust Vectoring Mechanism

The proposed cruciform idea will consist of two airfoil sections with gaps, mounted in or behind the nozzle in a '+' fashion. The paddles could be mounted to fit inside the cross section of the exit nozzle, separated by 90 degrees. The interior paddles shift in four directions together (up,down,left,right), turning the flow. One axis would be held constant while the other rotated, allowing for single axis vectoring. This can then switch, allowing thrust vectoring along two axes. At zero deflection, there will be a decrease in thrust due to the drag caused by the airfoil surfaces as well as flow separation.

Pros	Cons
Simple implementation	Reduced thrust at zero deflection
Dual axis thrust vectoring	One axis at a time

Paddles Figure 88 shows a thrust vectoring system implemented via paddles attached to the end of an engine's nozzle.

Figure 88: 3-Rudder Thrust Vectoring [12]

Paddles are curved extensions of the jet nozzle that can be actuated to fold into the jet stream, deflecting the flow to create the desired vectoring effect. At zero deflection their direction will be parallel to the jet stream, allowing for zero impedance of the stock thrust. The proposed idea will use four rudders (as opposed to the three rudders shown in Figure 88) in an axisymmetric design, where the midpoint of each rudder is offset by 45° from the aircraft y and z axes. The rudders will be actuated in adjacent pairs to provide a net change in the flow direction along these axes.

Pros	**Cons**
Negligible change in zero-deflection thrust	Rudder deflection angle significantly larger than actual TV angle
Documented success on similar mechanisms using 3 rudders	

10.1.3. MCB Microcontroller

Beaglebone Blue The first microcontroller taken into consideration is the Beaglebone Blue Microcontroller. An image of the microprocessor can be seen in Figure 89.

Figure 89: Beaglebone Blue Microcontroller[6]

The Beaglebone Blue is a microcontroller that the WHiMPS examined as a potential option. The Beaglebone Blue has 8 input/output ports. The WHiMPS have no experience with this microcontroller. The board costs 82 USD. There is not a lot of documentation for this microcontroller. This microcontroller has 2 cores and the processing speed is 1 GHz. The board has an area of 7.525 in^2. The power input requirement is 9-18 V which is significantly higher than a lot of the other boards. This board has both SPI and I2C ports and can have data retrieved through a USB or bluetooth connection.

Pros	**Cons**
Ability to get real time data without a physical connection	High power requirement
	Not much documentation

Beaglebone Black The next microcontroller taken into consideration is the Beaglebone Black Microcontroller. An image of this microcontroller is shown in Figure 90 below.

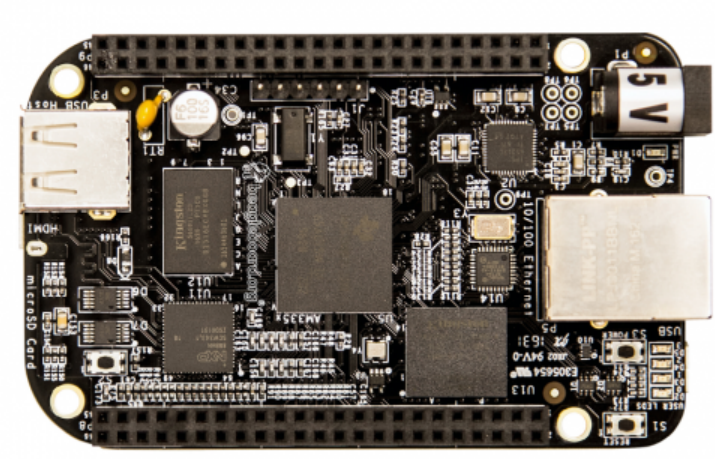


Figure 90: Beaglebone Black Microcontroller[7]

The Beaglebone Black is a microcontroller that the WHiMPS examined to potentially use. There are 65 input/output ports. Nobody on the team has direct experience, but it is open source and community supported which means it is well documented online and easier to navigate. This board is 55 USD. There are 2 cores and this microcontroller has a processing speed of 1 GHz. The area of this board is 7.14 in^2. The

power input requirement is 5V. This board supports both SPI and I2C, and data can be retrieved through an ethernet cable or a USB cable.

Pros	Cons
65 I/O ports	No ability to get real time data without a physical connection

Raspberry Pi 4 Another microcontroller taken into consideration is the Raspberry Pi 4 Microcontroller. An image of this microcontroller can be seen in Figure 91 below.

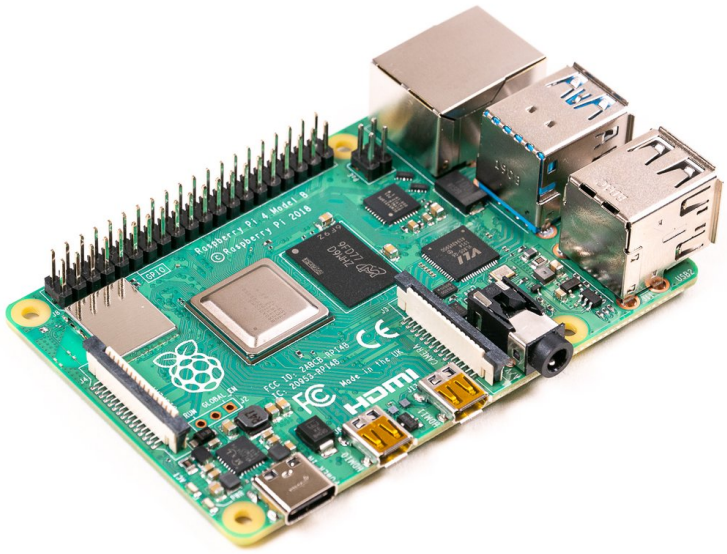


Figure 91: Raspberry Pi 4 Microcontroller[8]

The Raspberry Pi 4 is a microcontroller that the WHiMPS examined as an option. The microcontroller has 14 input/output ports. The WHiMPS have experience using this microcontroller, in addition to it being open source and community supported which is good because more documentation and experience increases the likelihood of success. This board has 4 cores and a processing speed of 1.5 GHz. This board is 35 USD and has an area of 8.05 in^2. There is a 5V power input requirement. There are multiple SPI and I2C ports supported, and real time data can be extracted using a USB-C or an ethernet cable.

Pros	Cons
4 cores and fast processing speed	Need physical connection to retrieve data
Lots of documentation and team experience	

Raspberry Pi Zero WH The fourth microcontroller taken into consideration is the Raspberry Pi Zero WH Microcontroller. An image of this microcontroller can be seen in Figure 92.

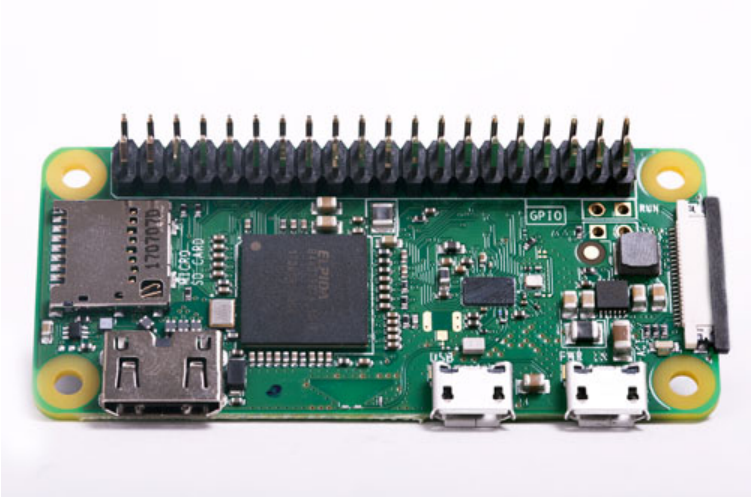


Figure 92: Raspberry Pi Zero WH Microcontroller[9]

The Raspberry Pi Zero WH is a microcontroller that the WHiMPS examined as a potential option. This microcontroller has 8 input/output ports. The WHiMPS have experience with this board in addition to it being open source and community supported which is helpful because increased experience and documentation assist in success. This board only has 1 core and a processing speed of 1 GHz. This board costs 15 USD. The area of this board is 3.02 in^2. This board requires a power input of 5V. This board supports SPI and can only have data retrieved using an SD card which is not ideal.

Pros	**Cons**
Small and low mass	Only 8 I/O ports and 1 core Difficult to retrieve data

NVIDIA Jetson TX2 The final microcontroller taken into consideration for the project is the NVIDIA Microcontroller. An image of this micrcontroller can be seen in Figure 93 below.

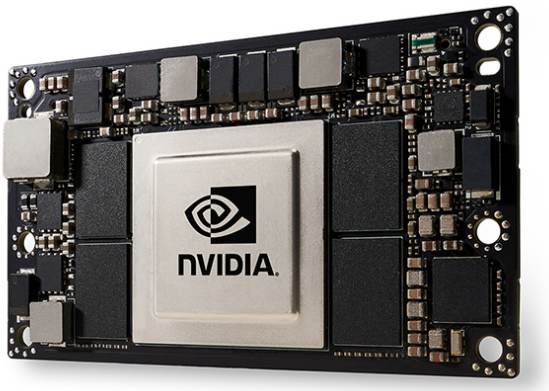


Figure 93: NVIDIA Microcontroller[10]

The NVIDIA Jetson TX2 is a microcontroller that the WHiMPS looked at as a potential microcontroller option. This microcontroller has 70 input/output ports. No team members have experience with this microcontroller, but it is well documented online. This microprocessor has 4 cores and a processing speed of 1.2 GHz. This board costs 400 USD. The area of this board is 41.94 in^2. The input power required is 5.5-19.6 V. This board has multiple SPI and I2C ports supported and can be communicated with by USB or ethernet cables.

Pros	Cons
70+ I/O ports	Very expensive
Lots of serial interface options	Very large and massive

10.1.4. Trade Study Liekert Scales

Windmill Prevention

Windmill Prevention					
	1	2	3	4	5
Simplicity (Weighted 21%)	Very difficult, low percentage of success predicted, little to zero team experience	Difficult, low percentage of success predicted, limited team experience	Moderate difficulty, possible success, some team experience	Expected success, somewhat experienced in	Confident in success, team feels experienced
Safety (Weighted 19%)	Extremely high probability of engine destruction, part failure would result in debris in engine	Very high probability of engine destruction	Moderate probability of engine destruction	Low probability of engine destruction	Part failure would not result in engine failure
Mass/ profile (Weighted 13%)	Greatly increases drag area of the engine and has significant impact on mass of engine, engine no longer effectively useful?	large increase in drag area of the engine and significant impact on mass, but still theoretically able to fly	Moderate impact on drag area of the engine and moderate impact on mass	Some impact on drag area of the engine, but not major. Some impact on mass but still practical to fly	Negligible impact on drag area of the engine and mass increase, does not impact flight
Reliability (Weighted 17%)	Expected consistent failure and large maintenance requirements	Failure predicted and maintenance will be required frequently	Possibility of part failure, some maintenance required	Low possibility of part failure, little maintenance required to keep functional	Part failure not predicted, mimimal to zero maintenance required
Performance (Weighted 30%)	Windmill above 0.5 rpm	Windmill lower than 0.5 rpm, will not prevent vibrational movement	Windmill lower than 0.5 rpm	Windmill completely stopped	Windmill completely stopped, resistant to vibrational motion

Figure 94: Trade Study Breakdown for Windmill Prevention Mechanism

Thrust Vectoring

Thrust Vectoring					
	1	2	3	4	5
Simplicity (Weighted 24%)	Very difficult, low percentage of success predicted, little to zero team experience	Difficult, low percentage of success predicted, limited team experience	Moderate difficulty, possible success, some team experience	Expected success, somewhat experienced in	Confident in success, team feels experienced
Safety (Weighted 14%)	Extremely high probability of engine destruction, part failure would result in debris in engine	Very high probability of engine destruction	Moderate probability of engine destruction	Low probability of engine destruction	Part failure would not result in engine failure
Mass/ profile (Weighted 11%)	Greatly increases drag area of the engine and has significant impact on mass of engine, engine no longer effectively useful?	large increase in drag area of the engine and significant impact on mass, but still theoretically able to fly	Moderate impact on drag area of the engine and moderate impact on mass	Some impact on drag area of the engine, but not major. Some impact on mass but still practical to fly	Negligible impact on drag area of the engine and mass increase, does not impact flight
Reliability (Weighted 20%)	Expected consistent failure and large maintenance requirements	Failure predicted and maintenance will be required frequently	Possibility of part failure, some maintenance required	Low possibility of part failure, little maintenance required to keep functional	Part failure not predicted, mimimal to zero maintenance required
Performance (Weighted 31%)	Thrust impedance, vectoring angles not met	Thrust impedance, vectoring angles not met	Thrust impedance, one axis of thrust angles met	Negligible thrust impedance, thrust angles met	Zero thrust impedance, vectoring angles met

Figure 95: Trade Study Breakdown for Thrust Vectoring Mechanism

MCB Microcontroller

# of Digital/Analog I/O Ports	Not enough I/O ports for our predicted need	Fewer I/O ports than desired for our predicted need	Just enough I/O ports for our predicted need	A few more I/O ports than our prediced need	More I/O ports than we would ever feasibly need
Familiarity/Documentation	Team has no experience with microcontroller environment and it is poorly documented	Team has little experience with microcontroller environment and it is poorly documented	Team is fairly familiar with microcontroller environment and there is some documentation	Team is familliar with the microcontroller environment and there is some documentation	Team is extremely familiar with microcontroller environment and there is lots of documentation
# of cores	1 core	2 cores	3 cores	4 cores	More than 4 cores
Processing Speed	<1MHz	<1GHz	1GHz	>1GHz	>1THz
Cost	>$1000	>$500	>$100	>$50	<$50
Board size	Too large to practically fit on the engine without interfering with form factor >50 in^2	Larger than desired to fit on the engine without interfering with form factor >30 in^2	Fine size to interface with the engine >10 in^2	Good size to interface with the engine <10 in^2	Very good size to interface with the engine <5 in^2
Serial interface	Contains only one source of serial		Contains more than one		Contains more than one
User Serial Interface	Can not obtain real time data		Can only be connected by a cable to obtain real time data		Can connect remotely to obtain real time data
Power	>18V	>9V	>5V	>3.3V	3.3 V

Figure 96: Trade Study Breakdown for the Microcontroller

10.1.5. Design Components

Conical Spring

Figure 97: COTS Conical Compression Spring

MCB Design

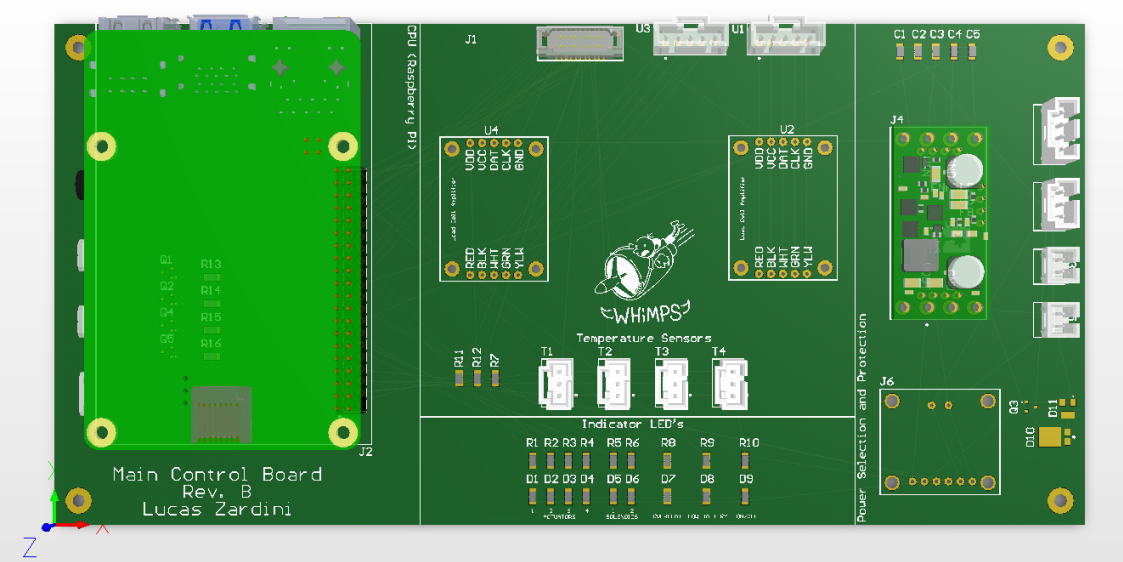


Figure 98: Main Control Board Rev. B Design

Data Line Connections to MCB

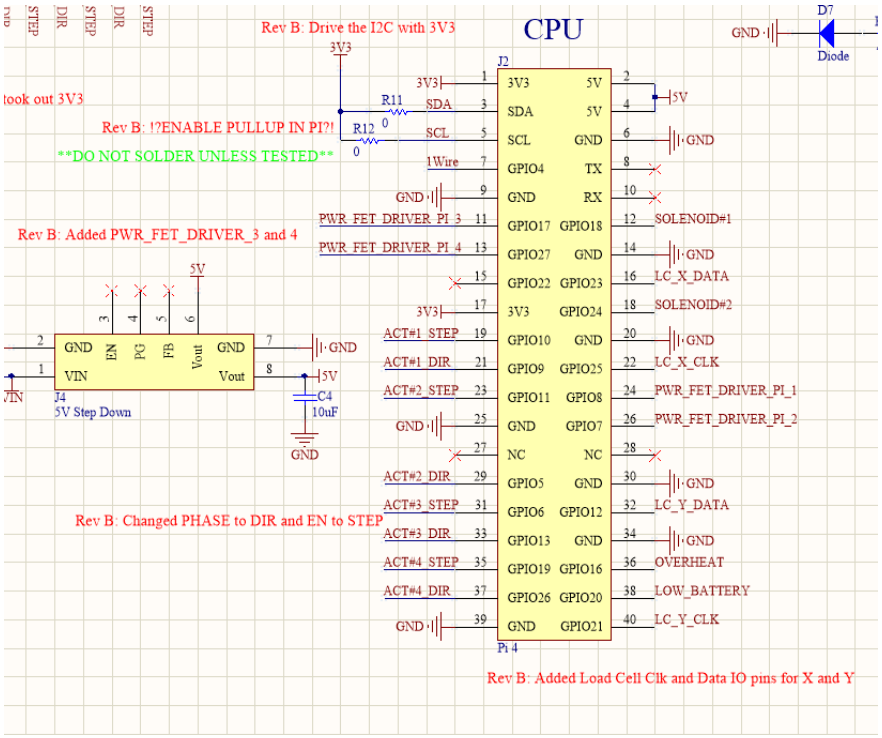


Figure 99: Main Control Board CPU Schematic

MCB Load Cell Schematic

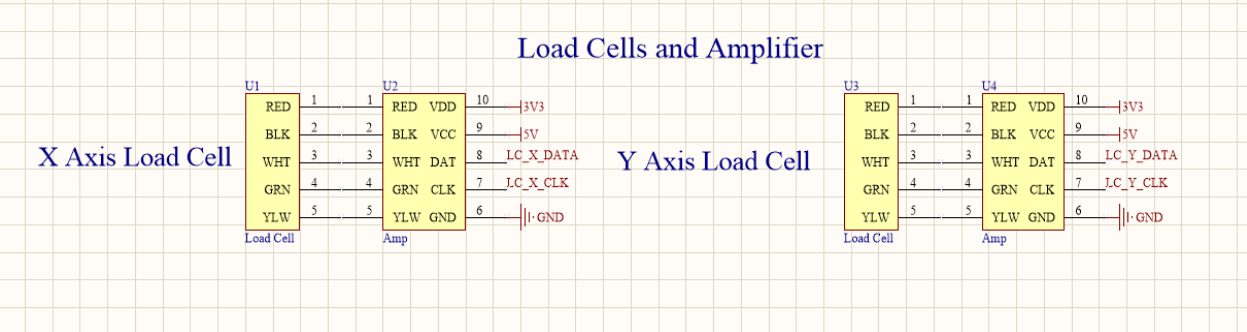


Figure 100: MCB Load Cell Schematic

MCB Indicator LED Circuits

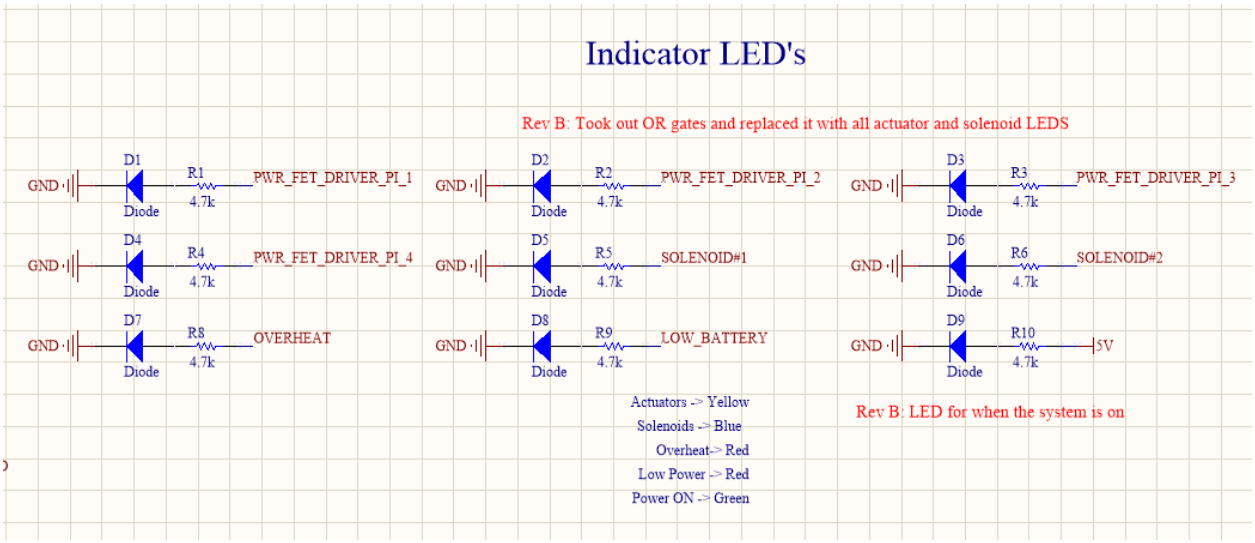


Figure 101: Main Control Board Indicator LED Circuits

ICUP Schematics

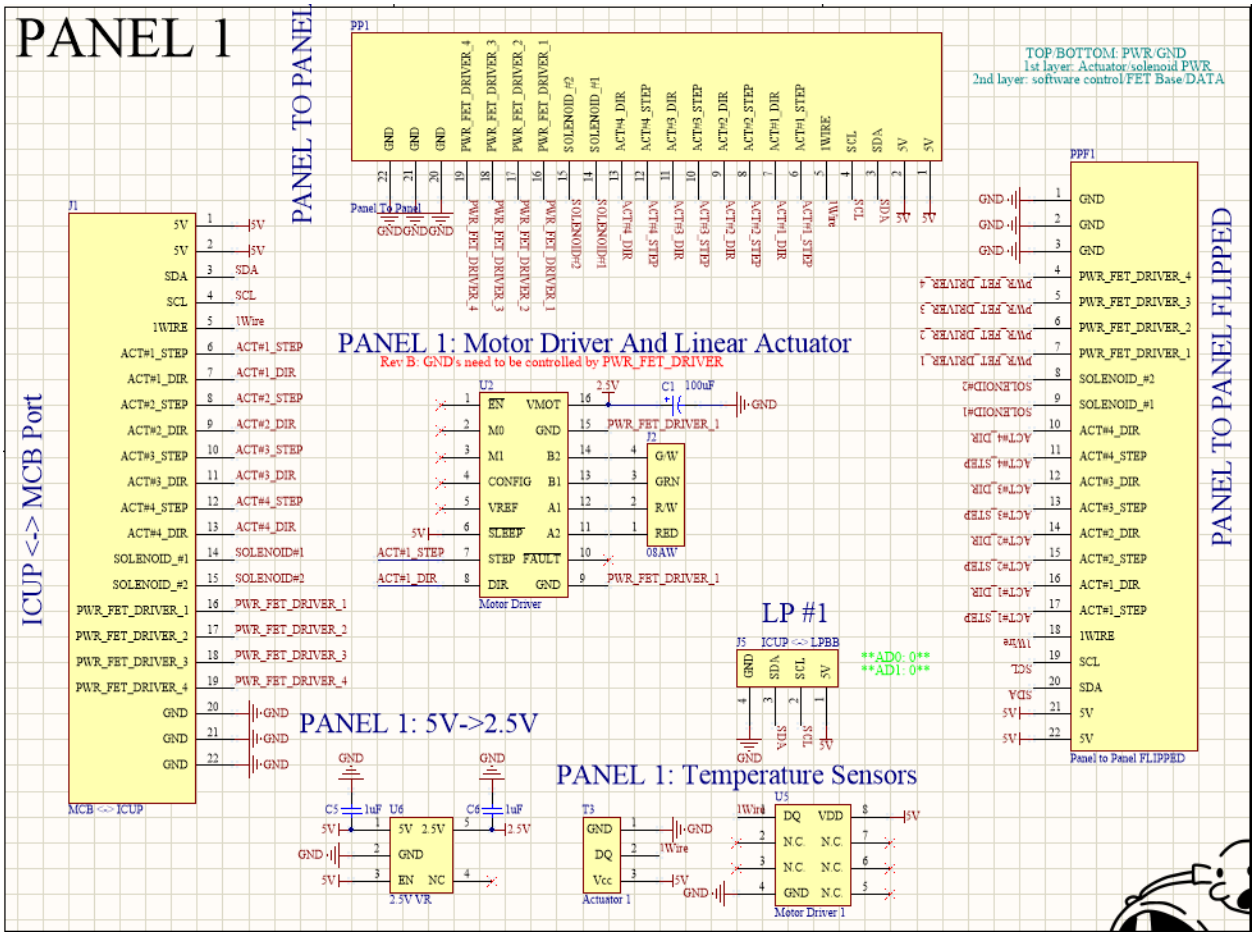


Figure 102: ICUP Panel 1

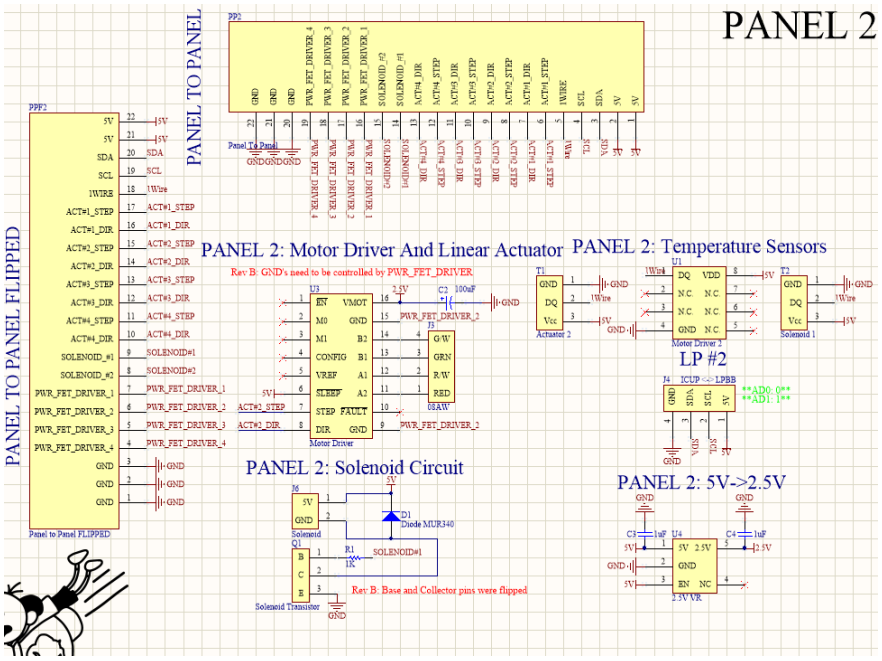


Figure 103: ICUP Panel 2

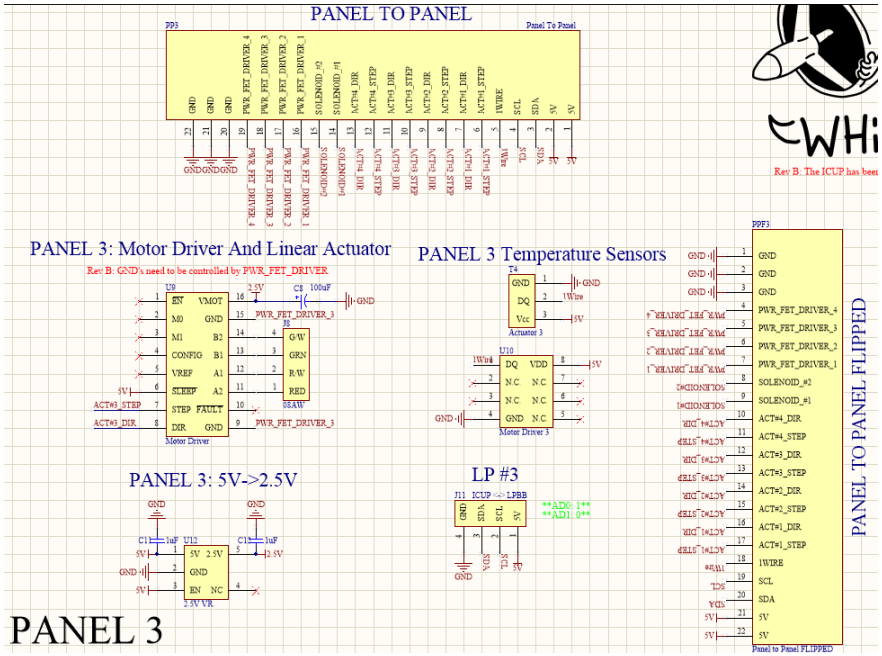


Figure 104: ICUP Panel 3

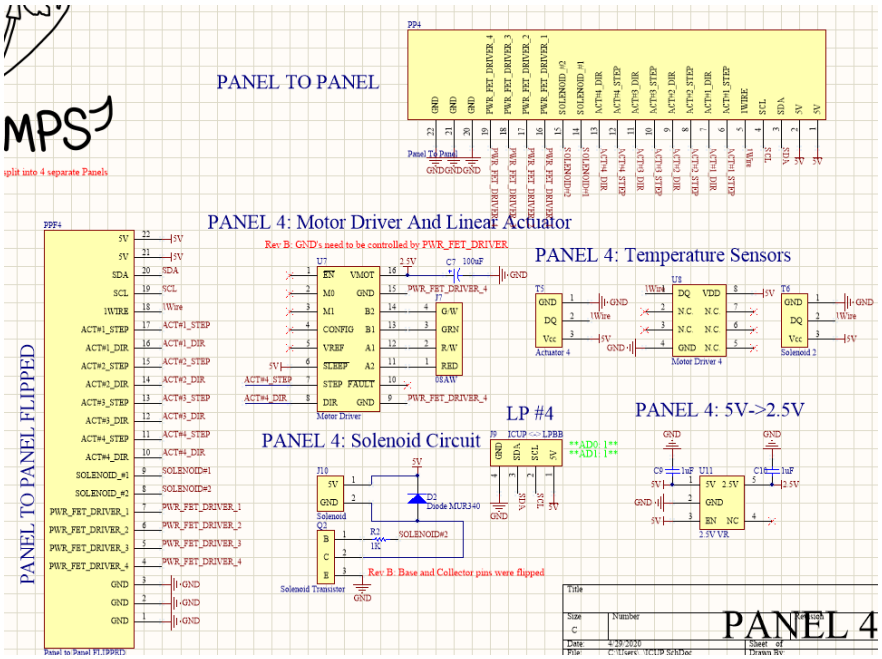


Figure 105: ICUP Panel 4

10.1.6. Budget Breakout

	Parts/Misc	Raw Materials	Boards	Unexpected Expenses
Electrical	295.66	0	71.29	93
Thrust Vectoring	470.86	630.72	0	0
Windmill Prevention	33.78	20	0	50
General, Test Stand	401.05	92	0	660

10.2. Appendix B: Main Control Board Software Reference Material

10.2.1. Code Examples

Listing 1: Subsystem Base Header File

```
/*
 * SubsystemBase.h
 *
 *  Created on: Jan 15, 2020
 *      Author: Alexandra Paquin
 */

#ifndef SUBSYSTEMS_SUBSYSTEMBASE_H_
#define SUBSYSTEMS_SUBSYSTEMBASE_H_

class SubsystemBase{
public:
    SubsystemBase(){};
    virtual ~SubsystemBase(){};

    /* No implimentations for this hierarchical level */
```

```
    virtual void handleCommand(CommandStruct * cmd) = 0;
    virtual void initialize() = 0;
    virtual std::vector<uint8_t> getHealthandStatus() = 0;
};

#endif /* SUBSYSTEMS_SUBSYSTEMBASE_H_ */
```

Listing 2: WPM Class Definition

```
/*
 * WPM.h
 *
 *  Created on: Jan 15, 2020
 *      Author: Alexandra Paquin
 */

#ifndef SUBSYSTEMS_WPM_H_
#define SUBSYSTEMS_WPM_H_

#include "subsystems/SubsystemBase.h"
#include "utility/PacketDefinitions.h"
#include "interfaces/SolenoidInterface.h"
#include "interfaces/TemperatureInterface.h"

enum Solenoid{
    SOLENOID1 = 0,
    SOLENOID2 = 1
};

enum WPMCommandOpcode{
    FAIRING_EJECT = 0x00
};

class WPM: public SubsystemBase{
public:
    WPM();
    ~WPM();

    /*
     * Inputs: Command Packet
     * Outputs: None
     * Functionality: Map opcode to correct command execution
     * Notes: This function will always be called regardless of the software mode
     *
     */
    void handleCommand(CommandStruct * cmd);

private:
    Lock lock;
    LogTags tags;

    uint8_t state_solenoid1;
```

```
    uint8_t state_solenoid2;
    float temp_solenoid1;
    float temp_solenoid2;

    virtual void ejectFairing(CommandStruct * cmd) = 0;

};

#endif /* SUBSYSTEMS_WPM_H_ */
```

Listing 3: WPM Implementation

```
/*
 * WPM.cpp
 *
 *  Created on: Feb 3, 2020
 *      Author: Alexandra Paquin
 */


#include "subsystems/WPM.h"

WPM::WPM(){}
WPM::~WPM(){}

void WPM::handleCommand(CommandStruct * cmd){

    uint8_t opcode = cmd->op_cmd;

    switch(opcode){
    case(FAIRING_EJECT):
        Logger::Stream(LEVEL_INFO,tags) << "Received eject fairing command";
        ejectFairing(cmd);
        break;
    default:
        cmd->error = ERR_OP_CMD;
        Logger::Stream(LEVEL_ERROR,tags) << "Invalid Command Opcode";
        break;
    }
}
```

Listing 4: WPM None Header

```
/*
 * NoneWPM.h
 *
 *  Created on: Jan 15, 2020
 *      Author: Alexandra Paquin
 */

#ifndef SUBSYSTEMS_NONEWPM_H_
#define SUBSYSTEMS_NONEWPM_H_
```

```
#include <subsystems/WPM.h>

class NoneWPM: public WPM{
public:

    /*
     * Constructor
     * Inputs: None
     * Functionality: Set the log tag
     */
    NoneWPM();
    ~NoneWPM();

    /*
     * Inputs: None
     * Outputs: None
     * Functionality: None
     */
    void initialize();

    /*
     * Inputs: None
     * Outputs: Health and Status Data Buffer
     * Functionality: Collect systems health and status data - return buffer full
     * of zeros with size of WPM health and status data packet
     * EdgeCases: Check size of buffer
     */
    std::vector<uint8_t> getHealthandStatus();


private:
    Lock lock;
    LogTags tags;

    uint8_t state_solenoid1;
    uint8_t state_solenoid2;
    float temp_solenoid1;
    float temp_solenoid2;

    /*
     * Inputs: None
     * Outputs: Command Acknowledgment
     * Functionality: Return failure command acknowledgment
     */
    void ejectFairing(CommandStruct * cmd);
};

#endif /* SUBSYSTEMS_NONEWPM_H_ */
```

Listing 5: WPM None Source File

```cpp
/*
 * NoneWPM.cpp
 *
 *  Created on: Feb 3, 2020
 *      Author: Alexandra Paquin
 */

#include "subsystems/NoneWPM.h"

NoneWPM::NoneWPM()
:state_solenoid1(0), state_solenoid2(0), temp_solenoid1(0), temp_solenoid2(0)
{
    tags += LogTag("Name", "WPM");
}

NoneWPM::~NoneWPM(){}

void NoneWPM::initialize(){

}

std::vector<uint8_t> NoneWPM::getHealthandStatus(){
    Logger::Stream(LEVEL_INFO, tags) << "Gathering WPM Health and Status";
    WPMHealthStatus wpm_hs(state_solenoid1,state_solenoid2,temp_solenoid1,temp_solenoid2);
    return wpm_hs.getHS();
}


void NoneWPM::ejectFairing(CommandStruct * cmd){
    Logger::Stream(LEVEL_ERROR,tags) << "WPM in disconnected mode, unable to eject fairing";
    cmd->error = ERR_SUB_MODE_NONE;
}
```

Listing 6: WPM Software Class

```cpp
#include "subsystems/WPM.h"
#include "interfaces/TemperatureInterface.h"
#include "interfaces/SolenoidInterface.h"
#include <cstdint>
using namespace std;


class SoftwareWPM: public WPM{
public:

    /*
     * Constructor
     * Inputs: None
     * Functionality: Set the log tag
     */
    SoftwareWPM(std::vector<TempInterface*> temp_devices,
```

```
    std::vector<SolenoidInterface*> sol_devices);
    ~SoftwareWPM();

    /*
     * Inputs: None
     * Outputs: None
     * Functionality: None
     */
    void initialize();

    /*
     * Inputs: None
     * Outputs: Health and Status Data Buffer
     * Functionality: Collects health and status data
     */
    std::vector<uint8_t> getHealthandStatus();


private:

    Lock lock;
    LogTags tags;

    uint8_t state_solenoid1;     // Stores the current solenoid 1 state
    uint8_t state_solenoid2;     // Stores the current solenoid 2 state
    float temp_solenoid1;        // Stores the current solenoid  1 temperature
    float temp_solenoid2;        // Store the current solenoid 2 temperature

    // Vector of temperature sensor interface objects, one object per temp sensor
    std::vector<TempInterface*> temp_devices;
    // Vector of solenoid interface object, one object per solenoid
    std::vector<SolenoidInterface*> sol_devices;

    /*
     * Inputs: None
     * Outputs: Command Acknowledgment
     * Functionality: Change solenoid states (the attributes), return
     * success command acknowledgment
     */
    void ejectFairing(CommandStruct * cmd);

    /*
     * Inputs: None
     * Outputs: None
     * Functionality: Gets solenoid states using the solenoid interface
     * objects, the states gathered from the object are from the mock
     * GPIO hardware drivers
     */
    void getSolenoidData(void);

    /*
     * Inputs: None
     * Outputs: None
```

```
     * Functionality: Gets solenoid temperatures using the temperature
     * interface objects, the temperature measurements gathered from the
     * object are from the mock OneWire hardware drivers
     */
    void getTempData(void);
};
```

Listing 7: WPM Software Class Implementation

```
/*
 * SoftwareWPM.cpp
 *
 *  Created on: Feb 11, 2020
 *      Author: Alexandra Paquin
 */

#include "subsystems/SoftwareWPM.h"

SoftwareWPM::SoftwareWPM(std::vector<TempInterface*> temp_devices,
std::vector<SolenoidInterface*> sol_devices)
: temp_devices(temp_devices),
    sol_devices(sol_devices),
    state_solenoid1(0),
    state_solenoid2(0),
    temp_solenoid1(30.100),
    temp_solenoid2(30.100)
{
    tags += LogTag("Name", "WPM");
}

// does nothing
SoftwareWPM::~SoftwareWPM(){}

void SoftwareWPM::initialize(){

}

std::vector<uint8_t> SoftwareWPM::getHealthandStatus(){

    Logger::Stream(LEVEL_INFO, tags) << "Gathering WPM Health and Status";
    getSolenoidData();
    getTempData();

    WPMHealthStatus wpm_hs(state_solenoid1,state_solenoid2,
    temp_solenoid1,temp_solenoid2);
    return wpm_hs.getHS();
}

void SoftwareWPM::getSolenoidData(void){

    state_solenoid1 = sol_devices[SOLENOID1]->getState();
    state_solenoid2 = sol_devices[SOLENOID2]->getState();
```

```
}

void SoftwareWPM::getTempData(void){

    temp_devices[SOLENOID1]->beginSample();
    temp_solenoid1 = temp_devices[SOLENOID1]->getSample();

    temp_devices[SOLENOID2]->beginSample();
    temp_solenoid1 = temp_devices[SOLENOID2]->getSample();

}

void SoftwareWPM::ejectFairing(CommandStruct * cmd){

    if (!sol_devices[SOLENOID1]->eject()) cmd->error=ERR_SOL1_EJECT_FAIL;

    if (!sol_devices[SOLENOID2]->eject()) cmd->error=ERR_SOL2_EJECT_FAIL;

}
```

Listing 8: Temperature Interface Example

```
/*
 * TemperatureInterface.cpp
 *
 *  Created on: Feb 10, 2020
 *      Author: Alexandra Paquin
 */


#include "interfaces/TemperatureInterface.h"
#include "utility/Logger.h"
#include <stdio.h>
#include <sstream>
#include <stdint.h>
#include <math.h>

/*!
 * Creates temperature interface with the provided device
 * \param onewireman the OneWireManager to be used
 * \param id the id of the device
 */
TempInterface::TempInterface(OneWireManager& onewireman, int id)
:onewireman(onewireman), id(id)
{
    tags += LogTag("Name", "TemperatureInterface");
}

TempInterface::~TempInterface(){}
```

```
//! Start the temperature sampling process. At least 750ms must pass before calling getSample
void TempInterface::beginSample(){
    onewireman.writeToFile(id, "start", "1");
}

/*!
 * Returns the sampled temperature from the sensor. Returns NAN on error
 * \return the temperature in C
 */
float TempInterface::getSample(){
    std::string data = onewireman.readFromFile(id, "w1_slave");
    std::stringstream ss(data);
    std::string line1, line2;

    if(!std::getline(ss, line1, '\n')){
        Logger::Stream(LEVEL_WARN, tags) << "Invalid data string: \"" << data << "\"";
        return NAN;
    }
    std::getline(ss, line2, '\n');

    if(line1.c_str()[36] != 'Y'){
        Logger::Stream(LEVEL_WARN, tags) << "Data fails crc: \"" << line1 << "\"";
        return NAN;
    }
    Logger::Stream(LEVEL_DEBUG, tags) << "Read \"" << line1 << "\" from the sensor";
    int rawval;
    if(sscanf(line2.c_str(), "%*s %*s %*s %*s %*s %*s %*s %*s %*s t=%d", &rawval) != 1){
        Logger::Stream(LEVEL_WARN, tags) << "Failed to parse \"" << line2 << "\"";
    }

    Logger::Stream(LEVEL_INFO, tags) << "Read " << rawval << "mC from the sensor";
    return (float) rawval / 1000.0;
```

Listing 9: Serial Bus Manager Example

```
template <typename device> class BusManager: public HardwareManager{

    /* Note: most of code has been omitted */

public:
    BusManager():devices(){}
    virtual ~BusManager(){}

protected:
    void initializeDevices(){
            for(typename std::vector<device>::iterator i = devices.begin(); i < devices.end(); i++){
                initializeDevice(*i);
            }
        }

    virtual void initializeDevice(device& dev){};
}
```

```
class OneWireManager: public BusManager<OneWireDevice>{
    /* Note: Most of the internal code has been omitted */
public:
    void initializeDevice(OneWireDevice& dev);
};
```

10.2.2. Subsystem Class Object Build Diagrams

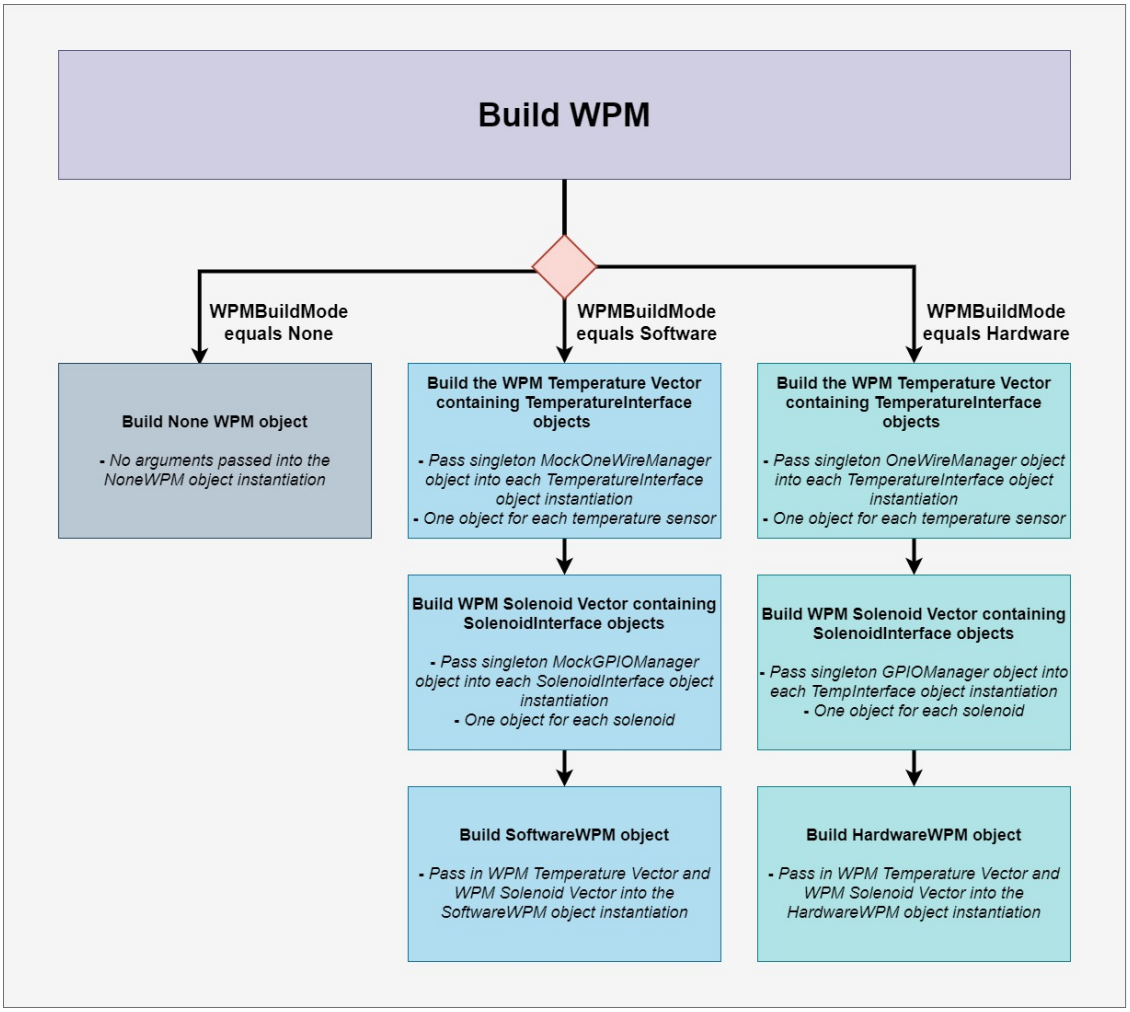


Figure 106: Flow Chart for the creation of the Windmill Prevention Mechanism Subsystem Object

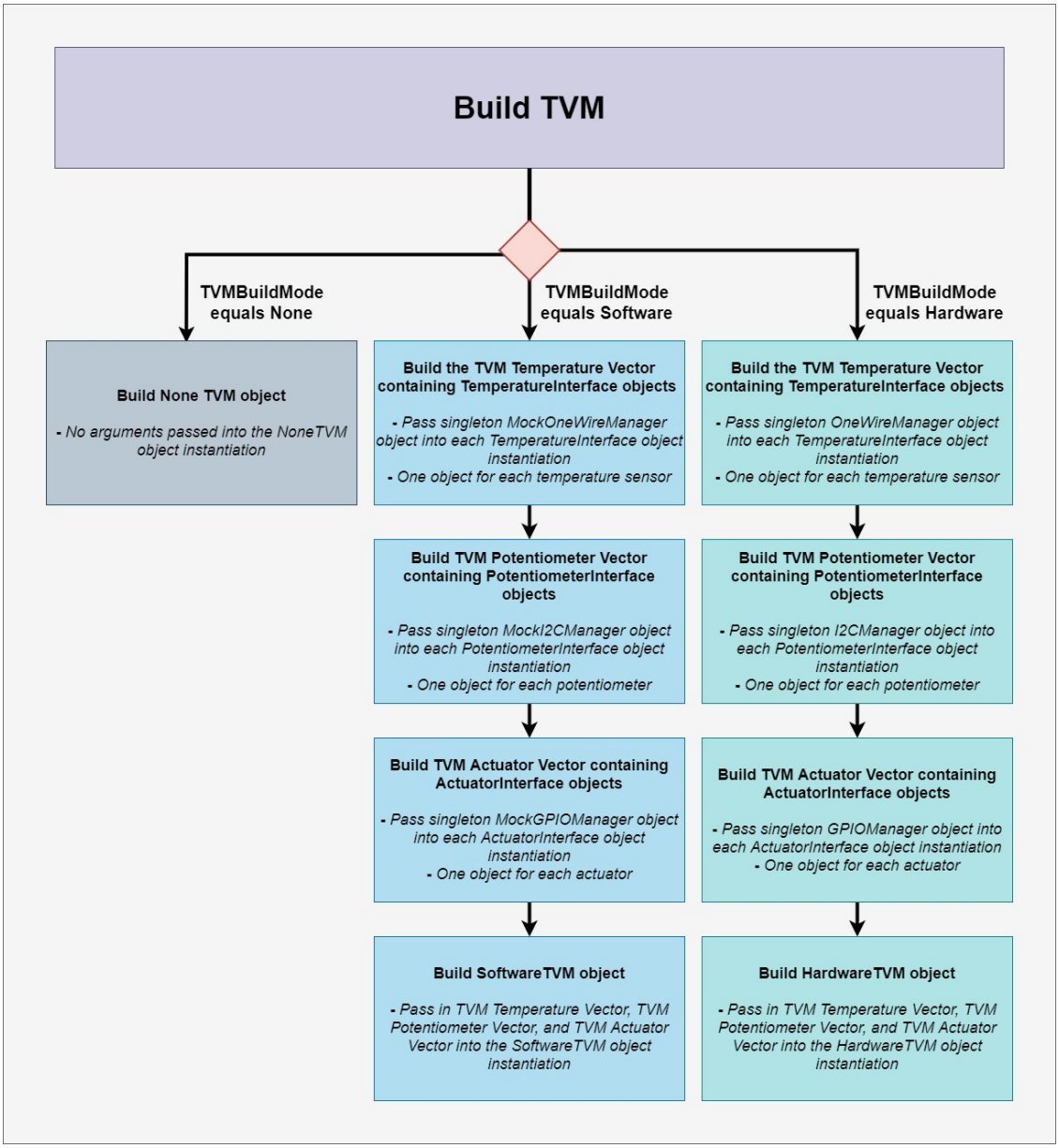


Figure 107: Flow Chart for the creation of the Thrust Vectoring Mechanism Subsystem Object

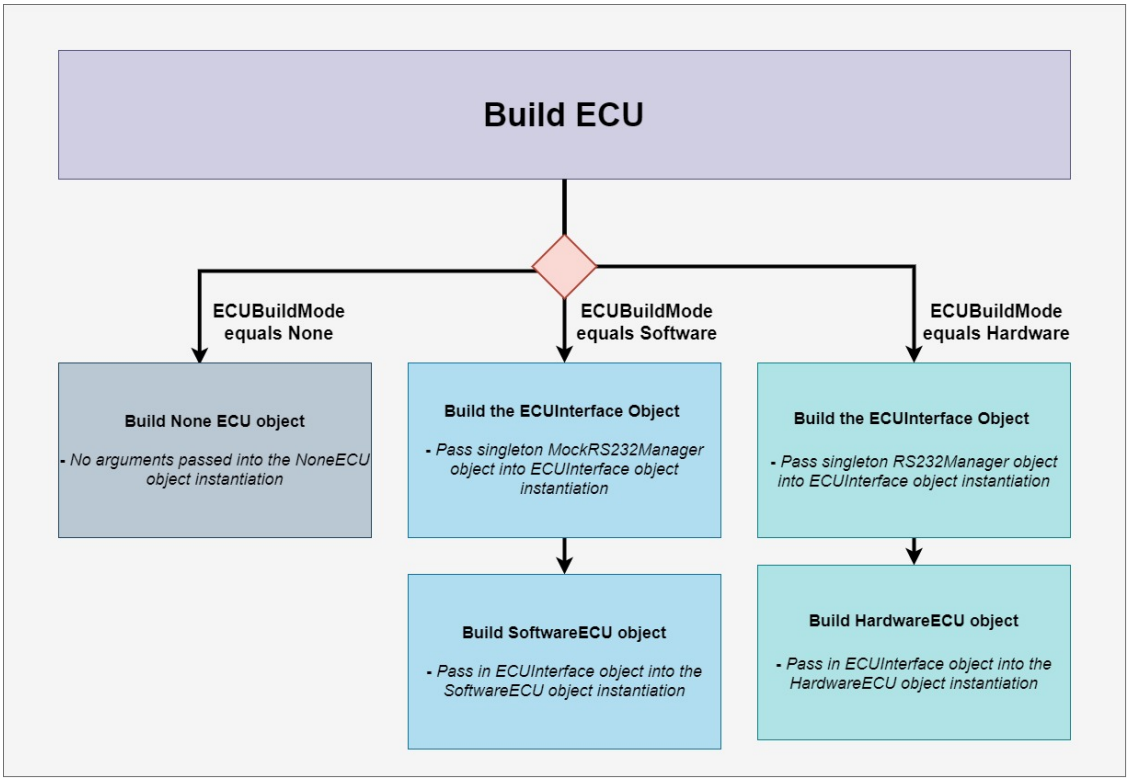


Figure 108: Flow Chart for the creation of the Engine Control Unit Subsystem Object

10.2.3. Packet Definitions

Table 8: Command Packet Definition

Field Name	Description	Size [bytes]	Data Type	Range	Units	Byte Start	Byte Location
OP_CMDPKT	Opcode corresponding to a command packet	1	uint8_t	0x0C	None	0	0
OP_SUB	Subsystem associated with the command	1	uint8_t	0x00: MCB 0x01: WPM 0x02: TVM 0x03: ECU	None	1	1
OP_CMD	Opcode for the subsystem's command	1	uint8_t	Depends on subsystem	None	2	2
TV_PARAM	Parameters for the TV_ADJUST command	16	float (x4)	[-2.72 ,+2.72]	mm	3	[3:18]
CHECKSUM (without TV_PARAM)	32-bit checksum	4	uint32_t	[0,4294967295]	None	3	[3:6]
CHECKSUM (with TV_PARAM)	32-bit checksum	4	uint32_t	[0,4294967295]	None	19	[19:22]

Table 9: Command Acknowledgment Packet Definition

Field Name	Description	Size (bytes)	Type	Range	Units	Byte Start	Byte Location
OP_CMDACK	Opcode corresponding to a command acknowledgement packet	1	uint8_t	0x0C	None	0	0
TIME_LINUX	Current time	4	uint32_t	[0,+4294967295]	ms	1	[1:4]
TIME_ELAPSED	Time since power on	4	uint32_t	[0,+4294967295]	ms	5	[5:8]
OP_SUB	Subsystem associated with command	1	uint8_t	0x00: MCB 0x01: WPM 0x02: TVM 0x03: ECU	None	9	9
OP_CMD	Opcode for the subsystem command	1	uint8_t	Dependent on subsystem	None	10	10
OP_ERROR	Error code for the recieved comand	1	uint8_t	0x00: None 0x01: OP_CMDPKT 0x02: OP_SUB 0x03: OP_CMD 0x04: CHECKSUM 0x05: SIZE 0x06: SUB_MODE_NONE 0x07: SOL1_ERR 0x08: SOL2_ERR 0x09: POT1_ERR 0x0A: POT2_ERR 0x0B: POT3_ERR 0x0C: POT4_ERR 0x0D: REBOOT_ERR 0x0E: RQST_HS_ERR 0x0F: RQST_LOG_ERR 0x10: CRC_ERR	None	11	11
CHECKSUM	Checksum	4	uint32_t	0-4294967295	None	12	[12:15]

Table 10: Health and Status Packet Definition

Field Name	Description	Subsystem	Size [bytes]	Data Type	Range	Units	Byte Start	Byte Location
OPCODE_HS	Opcode corresponding to a health and status data packet	General	1	uint8_t	0x00	None	0	0
TIME_LINUX	Current date and time	General	4	uint32_t	[0,+4294967295]	ms	1	[1:4]
TIME_ELAPSED	Time since power on	General	4	uint32_t	[0,+4294967295]	ms	5	[5:8]
TEMP_REG	Regulator Temperature	MCB	4	float	[-55,+125]	C	9	[9:12]
VOLTAGE_MCB	MCB voltage measured by ammeter	MCB	4	float		mV	13	[13:16]
POWER_MCB	MCB power measured by ammeter	MCB	4	float		mW	17	[17:20]
CURRENT_MCB	MCB current measured by ammeter	MCB	4	float		mA	21	[21:24]
STATE_SOLENOID1	State of solenoid 1	WPM	1	uint8_t	0: Not Ejected 1: Ejected	None	25	25
STATE_SOLENOID2	State of solenoid 2	WPM	1	uint8_t	0: Not Ejected 1: Ejected	None	26	26
TEMP_SOLENOID1	Solenoid 1 Temperature	WPM	4	float	[-55,+125]	C	27	[27:30]
TEMP_SOLENOID2	Solenoid 2 Temperature	WPM	4	float	[-55,+125]	C	31	[31:34]
POS_POT1	Linear Potentiometer 1 Position	TVM	4	float	[-2.72,+2.72]	mm	35	[35:38]
POS_POT2	Linear Potentiometer 2 Position	TVM	4	float	[-2.72,+2.72]	mm	39	[39:42]
POS_POT3	Linear Potentiometer 3 Position	TVM	4	float	[-2.72,+2.72]	mm	43	[43:46]
POS_POT4	Linear Potentiometer 4 Position	TVM	4	float	[-2.72,+2.72]	mm	47	[47:50]
TEMP_POT1	Linear Potentiometer 1 Temperature	TVM	4	float	[-55,+125]	C	51	[51:54]
TEMP_POT2	Linear Potentiometer 2 Temperature	TVM	4	float	[-55,+125]	C	55	[55:58]
TEMP_POT3	Linear Potentiometer 3 Temperature	TVM	4	float	[-55,+125]	C	59	[59:62]
TEMP_POT4	Linear Potentiometer 4 Temperature	TVM	4	float	[-55,+125]	C	63	[63:66]
RPM	Engine turbine RPM	ECU	4	uint32_t	[0,+200000]	Rev/min	67	[67:70]
EGT	Exaust Gas Temperature	ECU	4	float	[-20,+1400]	C	71	[71:74]
STATE_ENGINE	Engine State	ECU	1	uint8_t	0: OFF 1: Wait for RPM (Stby/Start) 2: Ignite 3: Accelerate 4: Stabilise 5: Not used 6: Learn LO 7: Not used 8: AutoOff 9: Not used 10: AutoOff 11: Run (reg.) 12: Acceleration delay 13: SpeedReg (Speed Ctrl) 14: Two-Shaft-Regulate (only for turbines with secondary shaft) 15: PreHeat1 (only for direct Kerosene startup mode) 16: PreHeat2 (only for direct Kerosene startup mode) 17: MainFStrt (only for direct Kersene startup mode) 18: Not used 19: Keros.FullOn (only for direct Kerosene startup mode)	None	75	75
FUEL_FLOW	Engine fuel flow rate	ECU	2	uint16_t	0-10000	mL/min	76	[76:77]
BATT_ENGINE	Engine battery charge	ECU	1	uint8_t	0-15	V	78	78
CHECKSUM	Checksum	ECU	4	uint32_t	0-4294967295	None	79	[79:82]
Total Size								**83**

10.2.4. Command Sequence for Testing with Simulated LabView

The WP1 build configuration was utilized when the commands shown in Table 11 are processed which puts the Windmill Prevention Mechanism in software mode while the Main Control Board, Thrust vectoring Mechanism, and Engine Control Unit are in None (disconnected) mode. Nine of the ten commands sent

contain at least one error for the purpose of testing the command handlers validation sequence.

Table 11: Circular Sequence of Commands Sent from Simulated LabView for Verification of WPM Success Criteria 1 and 2

Command Number	Command Packet Opcode (OP_CMD_PKT)	Subsystem (OP_SUB)	Command Opcode (OP_CMD)	Checksum (CHECKSUM)	Valid Command?
1	0x00: Does not exist	0x00: MCB	0x00: REBOOT	0x00000000	Invalid (first detects invalid command packet opcode)
2	0x0C: OP_CMD_PKT	0x05: Does not exist	0x00: REBOOT	0x00000000	Invalid
3	0x0C: OP_CMD_PKT	0x00: MCB	0x05: Does not exist	0x00000000	Invalid (first detects invalid command opcode)
4	0x0C: OP_CMD_PKT	0x00: MCB	0x00: REBOOT	0x00000000	Invalid (subsystem is disconnected)
5	0x0C: OP_CMD_PKT	0x00: MCB	0x01: REQUEST_HS_FILE	0x00000000	Invalid (subsystem is disconnected)
6	0x0C: OP_CMD_PKT	0x00: MCB	0x02: REQUEST_LOG_FILE	0x00000000	Invalid (subsystem is disconnected)
7	0x0C: OP_CMD_PKT	0x01: WPM	0x05: Does not exist	0x00000000	Invalid (invalid command opcode)
8	0x0C: OP_CMD_PKT	0x01: WPM	0x01: FAIRING_EJECT	0x00000000	Valid
9	0x0C: OP_CMD_PKT	0x02: TVM	0x05: Does not exist	0x00000000	Invalid (first detects invalid command opcode)
10	0x0C: OP_CMD_PKT	0x02: TVM	0x00: ADJUST_TV	0x00000000	Invalid (subsystem is disconnected)

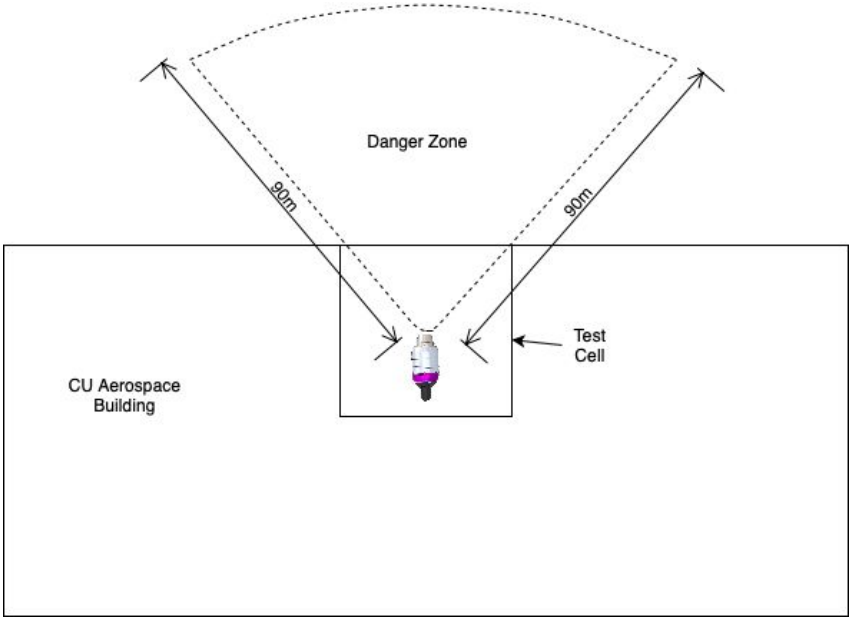
10.3. Appendix C: Test Safety Plan and Test Plans

The following pages contain the WHiMPS engine testing safety plan and two test plans: one for the dummy paddle test and one for the APOP replica test.

WHiMPS Safety Procedures for Engine Testing

The WHiMPS team recognizes the potential danger in testing a live jet engine. To mitigate these hazards, the WHiMPS team will follow the procedures in this document to both prevent incidents from occurring, and to respond to them effectively should an incident occur. It will be mandatory for all members present to review this document before each test to aid the safety officer, as safe testing relies upon the whole team understanding preventative measures and procedures in the case of an accident. To begin, the following preventative measures will be taken during all tests in which the engine is ignited:

1. Before the engine is mounted to the test cell, all flammable materials must be removed from the test cell's area. This includes materials that are not inherently flammable, but have the potential for explosion (ex. inert gas cylinders) or any other item that is a potential hazard. The safety lead and faculty advisor will conduct an inspection of the room before the engine is mounted to identify any missed potential hazard.
2. Before any testing may begin, the CU test cell's garage door must be opened to prevent the buildup of dangerous gasses. To prevent any hazard to anyone in the vicinity of the outside of the garage door, a perimeter will be formed and manned by sentinels. At the discretion of Matt Rhode, caution tape may also be used to form a perimeter along with sentinels. Sentinels must stand outside of the "danger zone" and prevent anyone from entering into the "danger zone". From an analysis of the worst case of all kinetic energy of a turbine blade being sent out the rear of the engine, it was determined that the maximum flight distance of a loose blade is 90m. The following diagram will demonstrate the perimeter formed:



3. Prior to any testing, Matt Rhode will be contacted at least 24 hours in advance. Matt Rhode will communicate this information upstream to the CU emergency services. This

is done because the CU emergency services will be able to respond quicker to any incident that occurs. Notifying the proper entities before the test is critical in mitigating the hazard to human life.

4. To prevent an incident where the engine start-up sequence begins at an unexpected time, connecting the engine battery to the ECU shall be the final step in the test preparation. The party responsible for plugging in the battery will be the only person in the test cell, and will immediately return to the test cell's observation area. Once the door is securely closed behind them, the start-up sequence may begin.
5. The WHiMPS team does not expect to be able to start the engine on the first try. If the engine does not ignite during a test, the shut-down procedure will be started. Once the shut-down procedure concludes, the safety lead may enter the test cell to disconnect the battery. Once the battery is disconnected, the team may enter the test cell. **The team will not be in the test cell with the battery connected to the engine under any circumstances.**
6. All sentinels must wear hearing and eye protection. Hearing and eye protection will be provided for all sentinels viewing the test at the time and location of the test.
7. All parties viewing the engine test must maintain a safe distance from the engine. This will occur in two different scenarios:
    a. The team is conducting a test within the CU test cell. This is the planned scenario for all tests. In this case, all team members must remain inside of the test cell's observation room for the entirety of the engine's operation. The engine shall not begin its start-up sequence until all members are within the observation room with the door securely closed. No member of the team will enter the test cell itself until the engine shut-down procedure has completely concluded under any circumstances.
    b. The team is conducting a test in a location besides the CU test cell. This scenario is not expected to occur. In this case, a blast-shield shall serve the same purpose as the CU test cell's observation room, with the same rules applying. The team will also follow the manufacturer's minimum allowable distances requirements during the operation of the engine. These distances are:

Location	Minimum allowable distance
In front of engine inlet	15 ft
On side of engine	25 ft
Behind engine nozzle	15 ft

8. Until the team gains experience with the Jetcat P100-RX engine, the engine shall not exceed 85% of its maximum thrust. This measure will aid in the prevention of accidental engine damage.

9. During the operation of the engine, a CO2 fire extinguisher must be within arms reach of the safety lead. There must be a clear path between the safety officer, the fire extinguisher, and the door to the test cell. The faculty advisor is singularly responsible for any use of the fire extinguisher. All other parties must stand out of the way and follow any instructions from the faculty advisor.
10. The CU test cell observation room has an access port covered by foam to allow wires to pass from the test cell into the observation room. This port creates potential for flames or projectiles to enter into the observation room in the case of an extreme emergency. To mitigate this, all viewing parties must stand to the side of this port at a minimum distance of 1 ft. There must not be a direct line between the engine, the port, and any viewing party whenever the battery is connected to the engine (e.g. don't stand anywhere that you could be hit by shrapnel if a catastrophic failure occurred).

Procedure in the case of an accident

The team is defining 3 separate cases in the case of a fire outside of the combustor occurring. The safety officer and CU faculty supervising the test will be responsible for categorizing any fire that occurs, and communicating the respective procedure from there.

1. **Minor Fire**: This is a small fire that starts outside of the engine, possibly from the expulsion of fuel from the rear of the engine. A minor fire will be put out via a CO2 fire extinguisher. In the case of a small fire occurring, the engine's shut-down procedure will start immediately. Once this shut-down procedure has concluded, the faculty advisor will enter the test cell and thoroughly extinguish the fire.
2. **Moderate Fire:** A moderate fire is a fire larger than a minor fire, including any fire on the exterior of the engine itself. A moderate fire can be put out with a fire extinguisher if the safety officer is given permission by the CU faculty present during the test. It is the CU faculty member's singular decision whether to extinguish the fire via CO2 fire extinguisher, or calling the emergency services. In the case of a moderate fire, the engine shut-down sequence will be started immediately. While the shut-down sequence is happening, the faculty test advisor will make a decision whether to use a fire extinguisher, or to call the emergency services.
3. **Severe Fire**: A severe fire is a large fire that possesses significant explosion risk or is too large to put out with a fire extinguisher. A fire that has the potential to reach the fuel tank or the potential to cause a lithium fire from the battery would also classify as a severe fire. In the case of a severe fire, the emergency services will be called immediately. The faculty advisor or safety lead can make the decision to evacuate the test cell observation room immediately. A severe fire poses significant risk to human safety.

Although a fire is the most likely incident to occur during these tests, other accidents are possible. In the case of another incident (including damage to the engine or facility), the team must follow directions from the CU faculty advisor present.

Windmill Prevention APOP Demonstration Replica

General Test Information	
Date	
Time	
Location	CU Aerospace Test Cell
Personnel	
Position	Name & Contact Phone Numbers
Test Supervisor	
Safety Lead	
Data recorder	

Emergency Contacts

CUPD Non-Emergency: 303-492-6666
CU Service Desk: (303) 492-5522
Boulder Fire Department: (303) 441-3350
Matt Rhode: (720) 201-1029

Relevant Documentation

1. WHiMPS Safety Plan

Test Roles and Responsibilities

Test Supervisor: Initiates and terminates testing procedure. Reads through each step and verifies that the appropriate actions have been taken before proceeding. Verifies all steps have been completed. Makes decision to terminate or proceed if unanticipated conditions (weather, operational etc) occur. Verify testing objectives are being met.
Safety Officer: Responsible for issuing PPE, verifying that general safety is being observed by all personnel in the vicinity. Responsible for the safe handling of combustibles and safety equipment, such as fire extinguishers or fuel dry.
Data Recorder: Coordinates video and sensor data measurement and collection of all relevant test data throughout the test. Collects all notes and observations from test. Will verify testing materials are reserved and present for test.

Introduction

This test will use the windmill prevention mechanism, the 3D printed fairing. This test will replicate how the Air Force Research Laboratory plans to test the windmill prevention mechanism in Dayton, Ohio in May. The fairing is the current flight model, but modifications may be made based upon the outcomes of this test. Pressure data will be collected manually during the test in case of a failure.

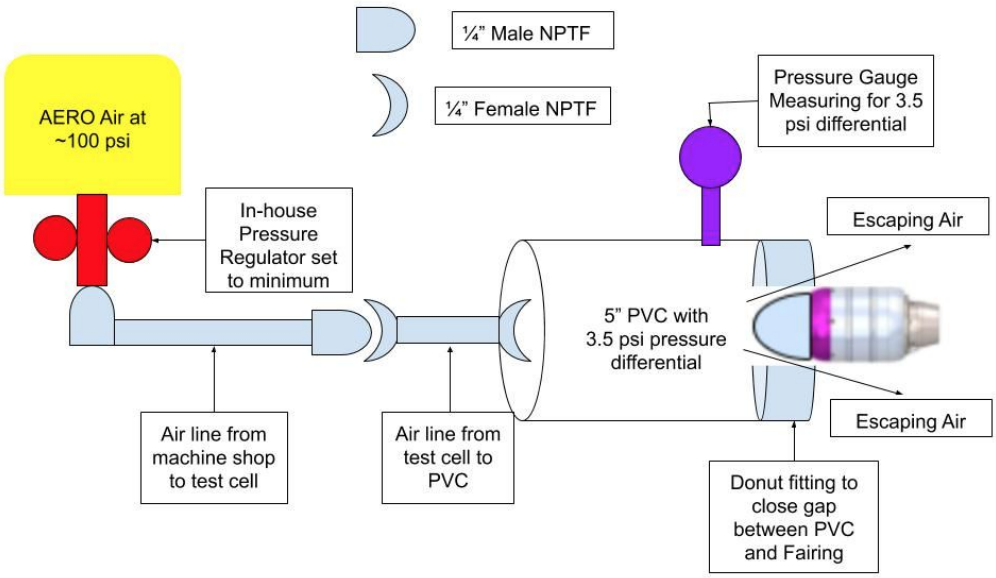
Objectives

1. Confirm the windmill prevention mechanism (fairing) can withstand Mach 0.8 flow
2. Verify that the fairing can withstand the conditions that will be imposed at the APOP demonstration
3. Visually determine the structural integrity of the fairing under a 3.5 psi pressure differential

Test Overview

Scope: The purpose of this test is to confirm that the windmill prevention fairing can withstand Mach 0.8 flow at 20k ft through simulation of a 3.5 psi pressure differential. This test will also confirm that the fairing will be able to survive the APOP demonstration at AFRL in May. We will visually inspect the fairing to determine the structural integrity, and will record a video to determine at what pressure, if any, the fairing fails.

Test Setup Diagram:



Pre-Test Checklists:

	Required Hardware	Check
1	3D printed fairing, two sides	
2	Air compressor	
3	PVC pipe with attached pressure gauge and foam insulation donut fitting	
4	Camera	
5	Air lines and connectors	
6	Pressure regulator	
7	Test mount	

	Preliminary Checks	Check
1	Air line is disconnected from the PVC pipe and is connected to the pressure regulator	
2	Ensure NPT connector and pressure gage are properly fitted to the PVC pipe and air line	
3	Set pressure regulator to minimum pressure	
4	Blast shield is set up behind the engine to block any possible projectiles	
5	Fairing is attached properly to the fore retention ring on the engine	
6	Press-fit the PVC pipe over the engine's fairing	
7	PVC pipe is propped-up in front of the test stand so that it is parallel with the ground. Tape may be wrapped around the exterior of the pipe to ensure it does not roll/slide during the test	
8	Engine ECU connections are made to relay RPM data to the computer in the test cell	

Test Procedure

Note: The air compressor emergency stop button should be easily accessible, and a protective plate shall be placed behind the fairing in case of failure or accidental ejection from the PVC pipe.

Test Checklist:

Step	Test Procedure	Check
1	Plug the air line into the female NPT connector on the PVC pipe	

2	Read pressure gage and relay pressure data over walkie-talkie to a team member standing near the pressure regulator	
3	If the pressure gage is reading less than 3.5 psi, increase the regulated pressure accordingly	
4	Once the pressure gage reads 3.5 psi, record the RPM data from the ECU for 1 minute	
5	After RPM data is recorded, dial down the pressure regulator back to its minimum output	

Disassembly and Cleanup Checklist:

Step	Disassembly and Cleanup Procedure	Check
1	Unplug the NPT connection between the air line and the PVC	
2	Power down and unplug the ECU	
3	Remove tape (if necessary) from the PVC and slide it away from the engine's fairing without tilting the PVC in any direction (reduce possibility of damaging fairing or fore retention ring)	
4	Remove fairing from the engine	
5	Remove fore retention ring from the engine	
6	Return all test materials to the senior projects locker and review data	

Results/Notes

Stock Engine Test Firing with Surrogate Paddles

JetCat P100-RX

General Test Information	
Date	
Time	
Location	CU Aerospace Engine Test Cell
Personnel	
Position	Name & Contact Phone Numbers
Test Supervisor	
Safety Lead	
Data recorder	
Operator	
Sentries	

Emergency Contacts

CUPD Non-Emergency: 303-492-6666

CU Service Desk: (303) 492-5522

Boulder Fire Department: (303) 441-3350

Matt Rhode: (720) 201-1029

Jet Cat Support: (661) 822-4162

Relevant Documentation

1. JetCat RX/RXi Turbines with V10 ECU Manual
2. Turbine Operating Instructions for JetCat RX Turbines
3. WHiMPS Safety Plan

Test Roles and Responsibilities

Test Supervisor: Initiates and terminates testing procedure. Reads through each step and verifies that the appropriate actions have been taken before proceeding. Verifies all steps have

been completed. Makes decision to terminate or proceed if unanticipated conditions (weather, operational etc) occur. Verify testing objectives are being met.

Safety Officer: Responsible for issuing PPE, verifying that general safety is being observed by all personnel in the vicinity. Responsible for the safe handling of combustibles and safety equipment, such as fire extinguishers or fuel dry.

Data Recorder: Coordinates video and sensor data measurement and collection of all relevant test data throughout the test. Collects all notes and observations from test. Will verify testing materials are reserved and present for test.

Operator: Physically operates all devices as needed to adjust engine to desired operation. This could include manual input devices, Labview, or other GUIs as testing progresses.

Sentries: Stand outside of the test cell and prevent people from walking into the danger area behind the engine (described in the WHiMPS Safety Plan).

Introduction

This test will use the stock JetCat P100-RX engine. Once the startup sequence is confirmed, two surrogate paddles will be installed, which are already made to be deflected into the exhaust jet. The paddles are not the actual paddles designed for the system, but will be similar in size and shape to provide a preliminary understanding of how the engine reacts with the paddles in the exhaust flow. Temperature data from the exterior of the engine casing will also be collected.

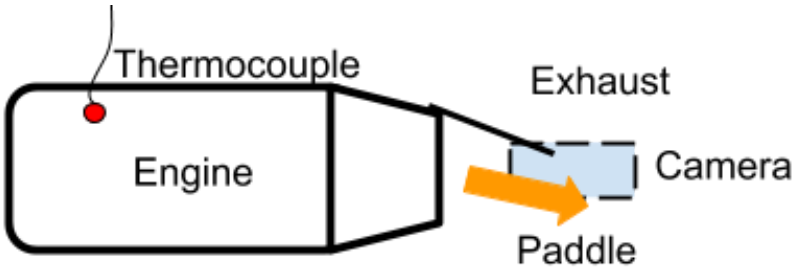
Objectives

1. Confirm the procedure for starting the P100-RX from the GSU.
2. Determine if the stock engine shuts down when the two paddles partially block the exhaust jet.
3. Visually determine if the paddles deflect the exhaust flow by any amount.
4. Determine the temperature profile of the engine casing over time.

Test Overview

Scope: The purpose of this test is to learn how to start the engine from the GSU and to provide qualitative data regarding the performance of the engine when paddles are deflected into the exhaust. There will also be quantitative temperature data collected from the exterior of the engine casing. Qualitative data collected will include video of the exhaust to determine if the exhaust stream deflects in the presence of the paddles.

Test Setup Diagram: The following image shows the test setup.



Pre-Test Checklists:

	Required Hardware	Check
1	Engine, associated tanks and lines	
2	Test stand	
3	Surrogate paddles and attachment mechanism	
4	Camera (could be a phone)	
5	Fuel and oil, in tank	
6	Thermocouples	
7	GSU and battery	

	Preliminary Checks	Check
1	Determine if weather allows testing	
2	Verify no loose objects or FOD on or about test stand	
3	Clear rear of engine of obstructions	
4	Inspect fuel gaskets for signs of damage	
5	Check the fuel lines and filter, make sure they are clean with no restrictions	
6	Fill fuel tank(s) and ensure the tank(s) are full	
7	Ensure there are no fuel leaks	
8	Clean spills (as required) and repair leaks	
9	Ensure camera is placed in correct position to collect data	
10	Verify secure placement of thermocouple	
11	Ensure all personnel are wearing PPE (Ear and Eye Protection, Gloves)	
12	Verify safety equipment (fire extinguisher) is staged and manned	

13	Ensure that all non-essential personnel are safely behind test cell door	

Test Procedure

Note: the emergency stop switch/button should be easily accessible by the safety officer and the operator so they can stop the test if any issues or concerns arise.

Test Checklist

Step	Test Procedure	Check
1	Plug battery into ECU	
2	Turn on ECU using pen (hold button for 5 full seconds)	
3	Within 60 seconds, hold down the Set button and press the Spool button on the GSU to start turbine. Engine should start automatically	
4	Once engine starts it will run through a series of states, ending in the green ‘OK’ LED lighting	
5	If the above procedure works, unplug the battery and proceed to the next steps	
6	Install surrogate paddles (**? or should this be done already and just go with it ?**)	
7	Start engine using procedure in steps 1-4	
8	Increase engine RPM by reasonable increment (currently unknown) by holding down the Spool button and pressing the + button. Let engine stabilize for 30 seconds.	
9	Continue increasing engine RPM by the same procedure in step 5 until max RPM is reached	
10a	If fuel is available, decrease RPM by holding the Spool button and pressing the - button, then letting the engine stabilize for 15 seconds at each decrement.	
10b	If fuel is not available, bring the engine to idle by holding down the Spool button, then pressing Run	
11	Return to idle for a minimum of 30 seconds before planned shutdown	
12	Turn off engine. Allow engine to sit for approximately 5 minutes as the engine completes its cooldown procedure. Make sure to record timing and temperature recordings from GSU during shutdown.	
13	When cooldown is complete, begin tear down of test rig	
14	END	

Disassembly and Cleanup Checklist

Step	Disassembly and Cleanup Procedure	Check
1	Turn off remote control - maybe?	
2	Turn off the receiver switch and unplug receiver battery - maybe?	
3	Unplug ECU battery from ECU	
4	Disconnect ECU Data cable from engine	
5	Drain fuel lines and disconnect	
6	Remove fuel lines from fuel tank, store in fuel container	
7	Unplug ECU Data Cable from ECU	
8	Unplug GSU from mini I/O board and store	
9	Remove engine from test stand and store, will return to locker	
10	Return fuel to fume hood, or into fire safe cabinet	
11	Verify no fuel spilled, or it has been cleaned up	
12	Verify all JetCat Components and load cell removed from Test Rig	
13	Test rig and safety shields properly stowed in test cell	
14	All PPE returned and stowed	
15	No trash, debris or exploded engine parts remain on test site	
16	Leave no man/woman behind	
17	END	

Results/Notes