MICRO-AIR VEHICLE CONTROL USING MICROELECTROMECHANICAL SYSTMES SENSORS

By

JOHN DONALD MANGELS III

A Thesis Submitted to The Honors College

In Partial Fulfillment of the Bachelors degree
With Honors in

Aerospace Engineering

THE UNIVERSITY OF ARIZONA

M A Y 2 0 1 7

Approved by:

Dr. Sergey Shkarayev
Department of Aerospace and Mechanical Engineering

Abstract

Micro Air Vehicles (MAV) are small unmanned aircraft that are highly sensitive to environmental disturbances causing dynamic changes in attitude and flight stability compared to more traditional unmanned air vehicles. Controlling the stability of an MAV is difficult and a significant research issue. The goal of this project is to perform a proof of concept study based on literature to demonstrate that Microelectromechanical Systems (MEMS) sensors can control the longitudinal stability of an MAV. MEMS sensors, specifically flow sensors used in this project, predict perturbations and aerodynamic effects which is critical for MAV performance because flight predictions can be used to prevent stall and failure in an MAV. The project focused on developing a control system that implemented MEMS sensors on a wing section and was tested in The University of Arizona's Educational Wind Tunnel.

Senior Design Team Member Contributions

Team Member	Focus & Primary Contribution
Syed Ammar Raza	• Design lead • Solidworks modeling of the Wing section and all wind tunnel hardware interfaces
Kevin Mueller	• Scribe and documentation lead • Administrative tasks
John Mangels	• Computation fluid dynamics • Wind tunnel testing results analysis
Namrah Habib	• Systems engineering: requirement development, and system verification and validation • MATLAB Equations of Motion Modeling in Simulink • Wind tunnel testing
Josh Raymond	• Aerodynamics calculations for wing section • Developed wind tunnel interfaces for both the horizontal and vertical mounting systems. • Servo control system design
Daniel Brauer	• Manufacturing of wing • Review & verification of wing design
Mohammed Azri Adb Rahim	• Manufacturing of wing • Review & verification of wing design
Daniel Sackson	• Control system hardware and software design and manufacturing

COLLEGE OF ENGINEERING
Aerospace & Mechanical Engineering

Innovating Flight through Microelectromechanical Systems Technology

Final Design Report

May 5, 2017

"Flow Characterization and Micro-Air Vehicle Control Using Microelectromechanical System Sensors"

FloSense Personnel

Ammar Raza	*Program Manager, Modeling Lead*
Kevin Mueller	*Program Deputy, Team Scribe/Historian*
Daniel Brauer	*Aircraft Performance/Design Co-Lead*
Namrah Habib	*Electronics Co-Lead*
John Mangels	*Computational Fluid Dynamics Lead*
Azri Rahim	*Aircraft Performance/Design Co-Lead*
Joshua Raymond	*Aerodynamics Lead*
Daniel Sakson	*Manufacturing Lead/Electronics Co-Lead*
Dr. Eniko Enikov	*Team Advisor, Electronics Consultant*
Robert Jacobi	*Team Advisor, Aerodynamics Consultant*
Dr. Jesse Little	*Team Advisor, Educational Wind Tunnel Operator*

Acknowledgements

Dr. Mathieu Joerger	*Aircraft Performance Consultant*
Zoltan Szabo	*Electronics Consultant*
Adrien Bouskela	*Electronics Consultant*

Table of Contents

Introduction

Problem Statement & Background Information

Micro Air Vehicles (MAV) are small unmanned aircraft that are highly sensitive to environmental disturbances causing dynamic changes in attitude and flight stability compared to more traditional unmanned air vehicles. Controlling the stability of an MAV is difficult and a current research issue. In order to address the problem of controlling MAV stability, Microelectromechanical Systems (MEMS) sensors are a current research topic at various institutions because MEMS sensors, specifically flow sensors used in this project, can predict perturbations and aerodynamic effects which is critical for MAV performance and stall prevention. MEMS sensors allow to detect flow speed, characterize the drag and pressure coefficients, in addition to predicting stall behavior unlike traditional Inertial Measurement Units (IMUS), which are used for unmanned vehicle control, that can only detect stall and then adjust flight parameters.

This project aims to perform a proof of concept study based on prior literature to demonstrate that MEMS sensors can indeed control the longitudinal stability of an MAV. The primary objective of this project is to develop a closed loop feedback control system that utilizes MEMS sensors to control the pitch stability on a wing with a single controllable surface. To demonstrate the functionality of the control system, the project also designed a wind tunnel interface for the wing section and tested the controller in The University of Arizona's Education Wind Tunnel located in the Aerospace and Mechanical Engineering Building.

Description of the Customer

The customer for the Micro Air Vehicle Control using Microelectromechanical Systems Sensors product, the MEMS Sensor control system, is The University of Arizona's Aerospace and Mechanical Engineering Department. The customer is interested in research and use of MEMS sensors and would like a control system that can successfully control the longitudinal stability of an MAV. The customer is research orientated and is looking for developments in the use of an old technology with a new application and would like for FloSense to develop and characterize this relationship more so than producing flight hardware.

Scope of the Document

This report describes the detailed design, trade studies, systems engineering, manufacturing, testing, analysis, and results of the Micro Air Vehicle Control Using Microelectromechanical System Sensors Project. The project started initially as a proposal requested by The University of Arizona's Aerospace and Mechanical Engineering Department. This report will explain major design choices and trade-offs between the initial proposal phase, preliminary design review (PDR), and critical design review (CDR), however, the focus of the reports is a detailed description of the final control system, wing design, analyses performed, wind tunnel testing, and results. The report provides complete documentation of the final design, fabrication, and performance of the wing and MEMS control system.

Scope of the Project: Objectives, Limitations, and Goal of Final Product

The final product of this project is to fully develop and test a closed loop feedback control system that utilizes MEMS sensors to control the pitch stability of a wing representative of MAV wings. Since MEMS sensors are not traditionally used, the project shall also determine and characterize the relationship between the MEMS sensors outputs, free stream velocity, and the angle of attack.

The main objectives to produce the final product described above are listed below:

- Design a wing section with a controllable surface that can be used to establish pitch stability
- Develop a controller and controller algorithm that implements closed loop feedback control for the angle of attack using the MEMS sensors on the wing section
- Perform computational fluid dynamics (CFD) analysis to predict the behavior of the wing section and develop relationships for sensor readings, free stream velocity and angle of attack
- Perform wind tunnel testing to calibrate the sensors
- Perform wind tunnel testing to demonstrate the functionality of the controller by testing open loop feedback and closed loop feedback systems
- Perform data analysis on wind tunnel testing results
- Compare wind tunnel testing results and CFD analysis

Limitations of the project include:

- Cost: the project is limited to a total budget of $1000
- Time: the project is constrained to a timeline from September 1st, 2016 to May 1st, 2017
- Testing is limited to The University of Arizona's Education Wind Tunnel and thus the wing structure must be designed to the constraints of the wind tunnel dimensions.
- The wing section design must be accommodated such that two-dimensional flow approximation can be made in wind tunnel testing
- The wing design must be representative of MAV in both size and mass
- MEMS sensor relationships with respect to the free stream velocity and angle of attack are not defined in literature and therefore must be determined as a part of this project.
- FloSense if comprised of Aerospace Engineers only with limited experience and the project focus is heavy on dynamic system controls and electronics, thus, the team will have to take time to learn and become familiar with control systems and electronic design and implementation.

Changes Made Since Critical Design Review

Changes made after CDR were primarily design changes where the wing design was slightly modified to best accommodate the wind tunnel interface as well as the control system. Additionally, some changes were made to accommodate ease of manufacturing. The changes since CDR are listed in Table 1 below alongside a justification for the design change.

System/Component Design Before CDR	System Component Design After CDR	Justification for making the change.
Controller: Arduino Micro	Controller: Arduino Nano	Convenience and cost: The customer opted to use a microcontroller he owned versus purchasing a new controller for the system
Controller Data Collection Interface: LabVIEW	Controller Data Collection Interface: MATLAB	Arduino Nano can interface with MATLAB, where MATLAB can be used to program the software for the controller, perform live data reduction, and post data processing.
Wing Skin Material: Monokote	Wing Skin Material: Shim Stainless Steel	The monokote slightly deformed PLA 3D printed parts and was difficult to remove for controller access, whereas the shim allowed easier removal and concealment and did not require any heat forming

Table 1*: Review of Changes Made Since CDR*

System Requirements

The system level requirements were developed based on the demands of the customer. The requirements are presented in Table 2 and are updated to the most recent negotiation of the project deliverables and scope that occurred after PDR with our customer. These system level requirements were further decomposed into subsystem level requirements based on constraints of equipment available to FloSense and literature research parameters. The subsystem level requirements are summarized in Tables 3 to 5.

System Level Requirement Name	Description	Verification Method		
		Analysis	Inspection	Test
1. MEMS Sensor Control System	The MEMS sensor control system shall be able to detect and characterize the angle of attack to be able to trim the wing in by appropriately adjusting the elevon angle.	x	x	x
2. MEMS Sensor System Wind Tunnel Testing	The MEMS sensor system shall successfully demonstrate functionality on a MAV wing section in a wind tunnel test.	x	x	x

Table 2: *System Level Requirements*

MEMS Controller Subsystem Requirement Name	Description	Verification Method		
		Analysis	Inspection	Test
1.1 MEMS Sensor System Outputs	The sensor system outputs shall be verified by comparing results to CFD results.	x		x
1.2 MEMS Sensor System Interface	The MEMS sensor system shall not affect the airfoil flight parameters or the airfoil boundary layer.	x		x
1.3 MEMS Sensor System Outputs to Stall Adjustment	The MEMS sensor system outputs shall be used to accurately determine and change the elevator angle to maintain the MAV's pitch stability in a closed feedback control system.	x		x
1.4 MEMS Sensor System Size	The MEMS sensor system shall use the least number of sensors to sufficiently characterize stall and flight parameters successfully.	x		

Table 3*: MEMS Subsystem Level Requirements*

Wind Tunnel Testing Subsystem Requirement Name	Description	Verification Method		
		Analysis	Inspection	Test
2.1 Wind Tunnel Testing Results	The wind tunnel test results shall be used to develop the relationship between the sensor output, the angle of attack, and free stream velocity.		x	x
2.2 Wind Tunnel Testing Data Collection Interface	The sensor system shall have a MATLAB Interface to conduct wind tunnel testing.		x	x
2.3 Wind Tunnel Testing Procedures	Wind tunnel testing procedures shall be developed prior to testing.		x	
2.4 Wind Tunnel Testing Hardware Interface	The wing section shall have a wind tunnel testing hardware interface that is not damaging to the Educational Wind Tunnel and is approved by Dr. Little	x	x	

Table 4: *Wind Tunnel Testing Subsystem Level Requirements*

MAV Wing Section Subsystem Requirements	Description	Verification Method		
		Analysis	Inspection	Test
3.1 MAV Sizing Requirement	The MAV shall be able to fit within University of Arizona Education Wind Tunnel which has dimensions of 30.5 cm x 30.5 ft x 61 cm.	x	x	
3.2 MAV Airfoil Selection	The MAV shall use a standard airfoil for the wing.	x	x	
3.3 MAV Wing Geometry and Aspect Ratio	The MAV shall use a simple wing geometry and shall have a minimum aspect ratio of 1.25. This requirement is based on literature research.	x	x	

Table 5: *MAV Wing Section Subsystem Requirements*

Summary of Preliminary Design Review Results

After preliminary design review (PDR), the project experienced a major descope in requirements and what the final product of the project. After PDR, Flosense management negotiated with the customer on project decampment due to limited information and transfer functions on using MEMS sensors to control the pitch stability, Flosense's constrained timeline of the project, in addition to, Flosense team members are part time engineers on this project and their personal time and resources, primarily cost, was limited.

Before PDR the projects aim was to perform flow characterization and use a closed loop feedback controller to suppress MAV attitude perturbations in a free flight experiment on glider launched with a consistent mechanism that would induce stall. The goal of the project was to design a longitudinally static and dynamically stable glider in which stall would purposely be induced and the sensors would adjust the flight path before failure. The project goal was ambitions considering there was no relationship defined in literature that the controller could use to relate the MEMS sensor outputs to the pitch angle, angle of attack, and free stream velocity. Therefore, the wind tunnel experiments would still be required to test and verity the controller system relationships before a free flight experiment where a glider would be controlled to a maximum range flight. The project descope dropped the glider free flight testing which shifted the primary objective of the project from a controlled glider flight experiment to performing wind tunnel testing to develop a relationship between the sensor outputs and flight parameters of the angle of attack and free stream velocity as well as Performa a wind tunnel experiment to show closed loop feedback control of the wing section to establish trim in varying velocity conditions. The PDR results are shown in the table below, alongside the updated CDR results and any descope that may have resulted to do a change in the project system level requirements.

System Component	PDR Results	CDR Results	Descope (if applicable)
Glider/Wing Design	Full glider design with projected design values based on an iterative design process AR range = 2.0 Re range = [7.0e4, 2.0e5] Mass range = [0.17-0.5] kg Wing sweep of 35º Wing span 15 cm Chord length 7.5 cm NACA 4412 airfoil for wing SA7035 airfoil for tail Figure 1 shows the glider design at PDR. Major analysis performed to get these values for the glider design was an iterative spreadsheet calculation that is given in the appendix.	Wing design with a single controllable elevon to perform wind tunnel testing and verify control system. AR range = 1.50 Re range = [7e4, 2.0e5] Mass range = [0.17-0.5] kg Rectangular planform Wing span 29 cm Chord length 19.3 cm NACA 4412 airfoil for the wing and elevon	Descoped from a full glider design to a wing design. The wing dimensions were modified to span the entire length of the wind tunnel to model all the CFD analyses and testing results using the 2D flow approximation.
MEMS Sensor Selection	FS2 Thermal Mass flow sensors	FS2 Thermal Mass flow sensors	N/A
CFD Analysis	CFD analysis was being performed to have a method of verification of the MEMS sensors wind tunnel testing results and allow to predict the relationship between the sensor outputs, free stream velocity, and angle of attack. Additionally, wind tunnel testing was being performed to predict behavior of the glider in the wind tunnel. At PDR, CFD analysis was underway. The focus of	CFD analysis was being performed to have a method of verification of the MEMS sensors wind tunnel testing results and allow to predict the relationship between the sensor outputs, free stream velocity, and angle of attack. Additionally, wind tunnel testing was being performed to predict behavior of the wing in the wind tunnel. CFD analysis had completed flow visualization of the wing at various angles of attack and	N/A This did not change

	CFD was to relate the stagnation point as a function of angle of attack, and use flow detection capability of the sensors to relate the free stream velocity to the angle of attack.	Reynolds numbers. The analysis was in the process of determining the sensor outputs relationship to the angle of attack and free stream velocity.	
Control System Design	The control system design had not been established by PDR.	Control system hardware and software architecture developed. Amplifier circuit Power supply and analog output to microcontroller FS2 thermal mass flow sensor Hitec HS-85MG Servo Arduino Micro microcontroller Plant transfer function relating the angle of attack to the elveon deflection angle Software in development in MATLAB for PID control	The descope of the project allowed to simplify the control system design for testing in that the system did not need to be entirely on the glider. The system was developed to connect to an external power source and interface with MATLAB.
MATLAB Modeling of the EOM	The glider longitudinal EOM had been fully modeled in Simulink to show the behavior of the glider in free flight given our predicted longitudinal stability coefficients of our glider design. The modeled plotted the position as a function of time, the pitch angle as a function of time, and the angle of attack as a function of time.	The model was completely descoped.	The modeled was descoped before CDR since a glider was no longer being developed to test the control system and the focus of the project was constrained to the wind tunnel tests. The model is provided in the appendices.
Wind Tunnel Testing	Wind tunnel testing had only been considered as a constraint on the glider size as this point.	Wind tunnel testing procedures had been fully developed and were in review.	N/A

Table 6*: Table of PDR Results*

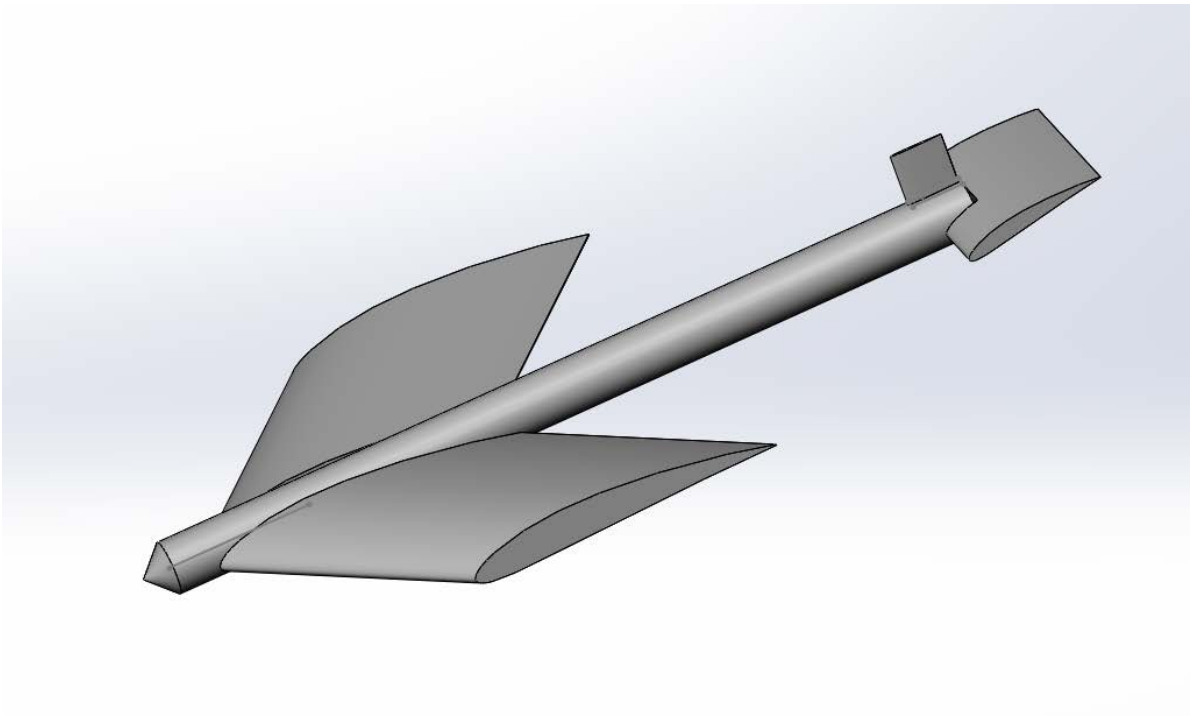


Figure 1: *Glider Design at PDR*

At the point of PDR, the major design was limited to the glider design and development. For the wing design, the NACA 4412 airfoil was selected. Numerous airfoils were explored and their characteristics can be viewed in the MEMS Spreadsheet on the Airfoil Data sheet in the Trade Studies folder. At the time, the main concern was finding an airfoil with relatively shallow stall with the idea of testing for trying to predict stall. Eventually, design parameters shifted more towards having a well-defined airfoil with clear data from simulations. Originally, the SA7035 airfoil was going to be used due to its shallower stall characteristics and having a clean set of data when simulated in XFLR5. However, this airfoil was not being well defined meaning it would be more difficult to manufacture. The NACA 4412 was used instead as it was the second-best choice in terms of data collected in simulations and, due to being a NACA airfoil, was well defined in its shape. Also, a 12% thickness is a fairly common thickness to use on real world wings and this would allow testing for how nicely the microcontroller would fit into the wing. The span of the wing was chosen to be 29cm, which spreads nearly completely across the wind tunnel test section. This allows for 2D assumptions to be valid in analysis and calculations. In order to determine the size of the elevon, a MATLAB code was developed based on moment calculations so that many various options could be explored more efficiently.

The glider was later changed to only be a proof of concept experiment with a wing in a wind tunnel.

Top-Level Design of Final Design Concept

The entire system used in this wing includes: the wing, the microcontroller, the servo and the mounting systems. The wing follows the NACA 4412 airfoil shape due to its well defined aerodynamic properties. The thickness also allows for the electronics to be placed within the wing. The elevon had to be about 22% of the total planform area to deflect approximately 22° in either direction.

Servo Rotary Drive System:

To control the wing's elevon deflection, this wing uses a servo rotary drive system. The reason for choosing this system was to keep all the components within the structure itself as to reduce drag. To achieve this, the servo arm is bent 45° at the end and encased with a snug fitting plastic lid top and bottom. As the rod is rotated, it creates a force on the elevon moving it up or down. The only issue with this system is that the servo arm must not touch any other part of the wing as it will introduce play into the system. This can all be shown in Figure 1:

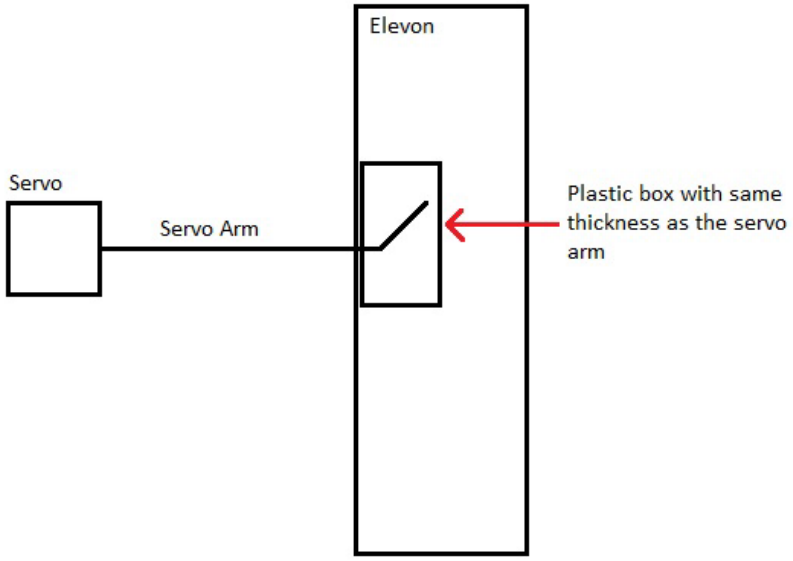


Figure 2: *Servo Rotary Drive System*

The electronics subsystem for the project can be broken down into two main components: the hardware which includes an Arduino Nano microcontroller as well as an analog amplifier and control circuit for the two FS2 thermal mass flow sensors used in the project. The other half of the electronic subsystem includes the firmware run on the Arduino to collect and send the analog voltage data as well as control the servo, and a MATLAB script for signal processing and running the control algorithm. Detailed design will follow in the following two sections.

For aid in visualization of the system architecture, a flow chart that links the hardware to software is shown below:

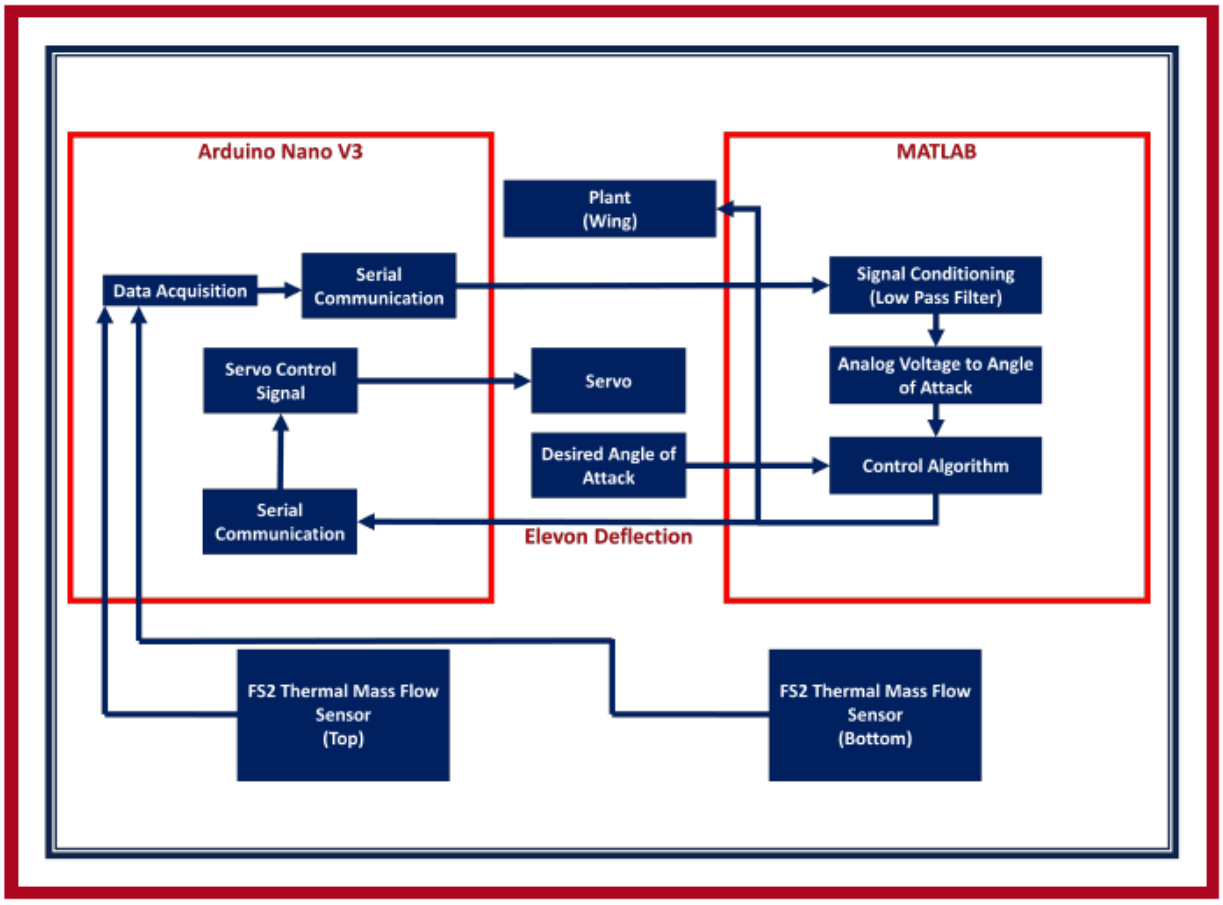


Figure 3: *Control System Architecture*

Subsystem and Interface Design (Hardware)

Wing Design

The design of this project started with the shape of the wing. A NACA 4412 airfoil was chosen for its superior lift capabilities given by the camber. This thickness was chosen to accommodate the electronics within the wing itself. The initial goal of this experiment was to study the flow around low aspect ratio (AR) wings because of the complexities the wings bring. Along with this consideration, the span was chosen so that the wing would similarly model the CFD results that were obtained through 2D flow assumptions. The Educational Wind Tunnel at the University of Arizona had a square cross-section of 30x30-cm constraining the span of the wing to be 29. To obtain a low aspect ratio of 1.5, a chord of 19.3 cm was chosen. Figure XX is the wing in full detail:

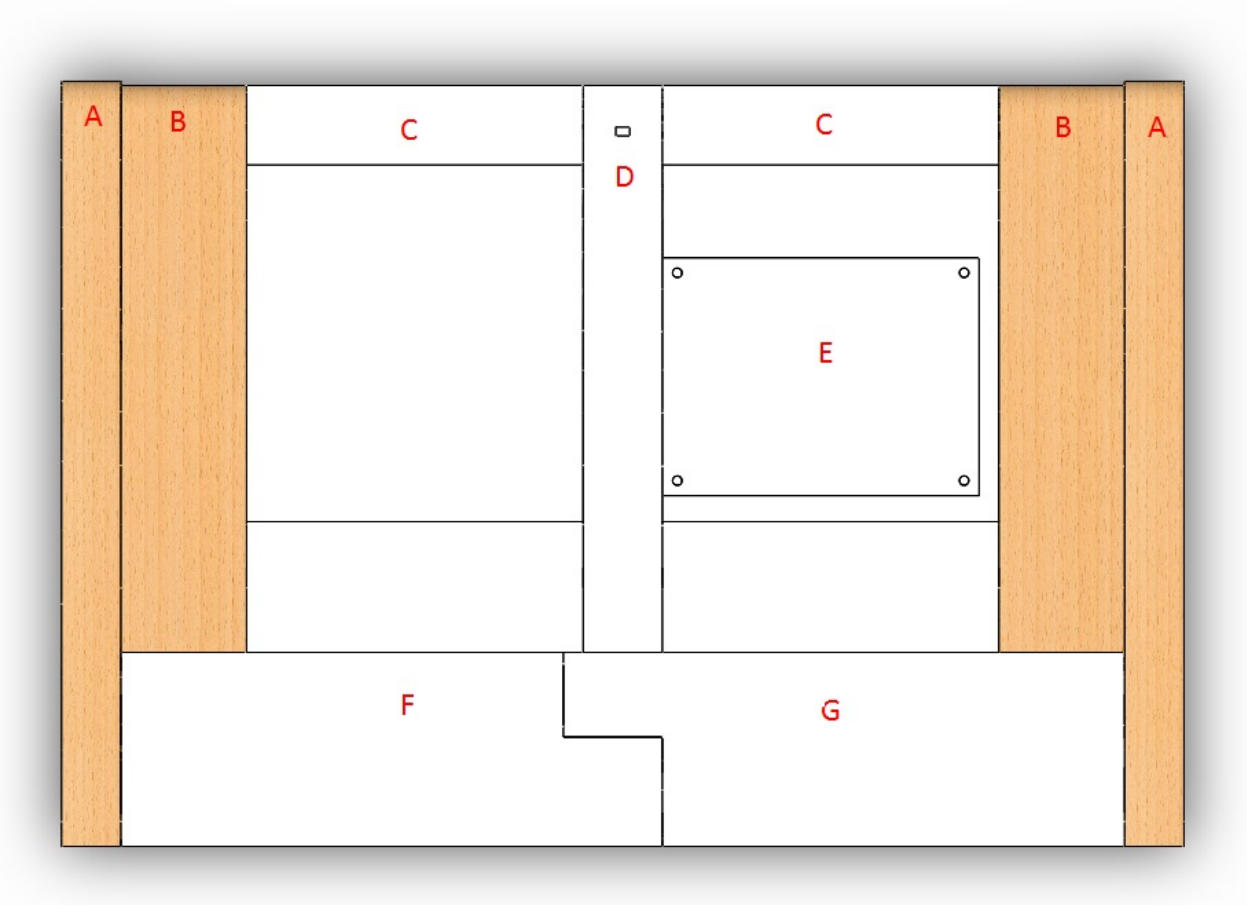


Figure 4: *Top View of Wing Components*

Another restraint in the design of the wing included the size of the elevon (shown as Components F and G). The elevon had to be 22% of the total planform area to deflect approximately 23° vertically in either direction. A decision was made to keep Components A to a thickness of 1.5 cm to maximize the electronic bay area (shown by Components C). This allowed for a 26x4.8 cm area shown by Figure 1. Component B are hollow ribs that follow the profile of the NACA 4412 airfoil but allow for the elevon pieces to fit snugly at the back. The purpose for

the hollow ribs is to maintain the airfoil shape while also reducing the overall mass of the wing. Components C are similar to the hollow ribs; however these are open on the top to allow for ease of access. These components held the electronics which included the microcontroller and the servo. Component E specifically held the microcontroller and fit the dimensions of the circuit board. This piece was removable to allow for access in case a change was needed. Component D is like that of Components B except thinner. Component D also has a small indentation at 4% of its chord length on both the top and bottom surfaces to mark where the MEMS sensors would be placed. The SolidWorks model for Component C does not have the hole required for the servo arm to go through. This mistake was realized after the 3D printing and drilled through the center.

The materials used for the wing were balsa wood, ABS plastic, and a stainless-steel shim. The balsa wood can be shown as a brown color in the picture, while the white components are made of ABS plastic. The outer ribs were made from balsa wood due to its relatively strong but lightweight properties. These pieces were manufactured by using a laser cutter and gluing the layers of wood together. The procedures for using the laser cutter are included in the Appendix section. A 3D printer was used to manufacture the rest of the pieces due to its complex geometries and the precision it required. Since Components C are open on the top, the stainless-steel shim was used to wrap the entire wing. This enabled the wing to have a smooth and symmetric surface. The stainless-steel shim was specifically used for its strength and resistance to bending.

One issue that the group encountered was adhering the balsa ribs to the 3D printed ABS parts. When using the laser cutter to form the balsa ribs, the laser cutter burned off a little bit more of the wood than designed. This led to uneven surfaces between the balsa ribs and required sanding to be smoothed. This sanding then made it difficult to line up the leading edges of the balsa ribs with the 3D printed trays. The solution to this uneven surface was the metal shim that covered the ribs to have a smooth surface.

This entire system, when fully refined, can have real world applications. This product can be put on any air vehicle to predict the flow and keep the plane at a trim level. Typically, most aircraft have Inertial Measurement Units (IMUs) that react to the flow around it. This technology is innovative in that it constantly senses when the plane is about to reach its stall angle of attack and prevents the aircraft from ever reaching that point. This paper discusses how to relate the voltage readings obtained by the sensor to the angle of attack the aircraft is currently experiencing. A linear relationship between the sensors' voltage reading difference and the angle of attack was found as well as between the sensor reading sum and Reynold's Number. The sensor readings' difference only has a linear relationship with the angle of attack at low angles of attack. This finding is important, but most MAVs will only experience low angles of attack due. This will help the plane maintain its optimum flight angle and minimize the disturbance effects.

Wind Tunnel Interfaces

For static tests (such as sensor calibration) the wing was mounted vertically in the wind tunnel used a developed mounting system. A shell was placed at the bottom of the test section where the wing could be inserted into and a seal was made for the hole in the top of the test section so that only the tube from the wing can go out. This seal both prevents leakage from the wind tunnel and adds additional support to the wing as drag pushes it. The hole in the center vertical mount is 0.5mm wider than the NACA 4412 airfoil used on the wing. The two side holes in this mount are sized to have ¼"-20 pan-head machine screws that are 1-1/2" long run through them and have washers placed between the screw head and the plastic to distribute the stress. The purpose of the screws is to attach the mount to the bottom of the test section of the wind tunnel through the holes of the yaw table. These holes in the yaw table are slightly wider than ¼" and therefore will not hold the screws on their own. To resolve this, ¼"-20 hex nuts are screwed onto the bottom of the screws that stick out from the wind tunnel. The 1-1/2" length of the screws allows for sufficient space to put the nuts on. To prevent tripping or separating the air during tests by the holes in the mount, placing tape neatly over the holes will suffice. The yaw table can then be used to easily rotate the shell and therefore rotate the wing to a desired angle of attack. The yaw table is labeled with various angles to allow for calibration tests with the sensors to take place.

The bottom later of the shell is 2mm thick before making the sloped shape towards the center seen in Figure 5. The slope is used to turn the flow on the lower part of the wing towards the wing or nicely around the wing in order to prevent separation from occurring. Separation would cause vortexes to form and could reach the sensors on the wing causing skewed data. The material used to make both the vertical mount and the top seal was PLA due to its lower cost of use while still maintaining sufficient structural integrity. For the vertical mount, a 60% filler was used to save on material costs and 3D printing time.

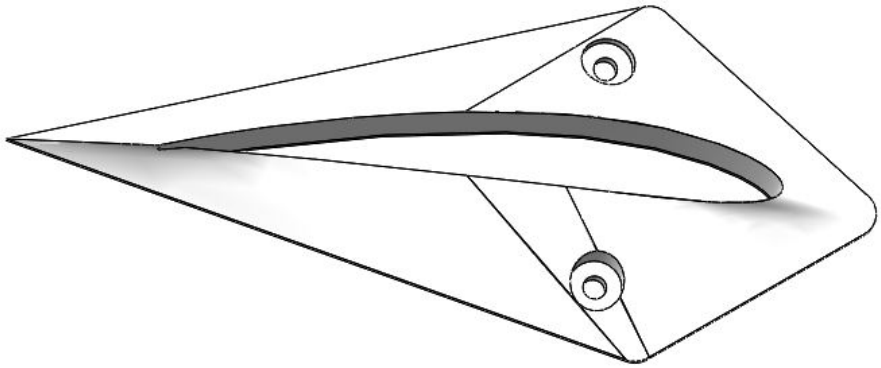


Figure 5*: Vertical Mount Shell*

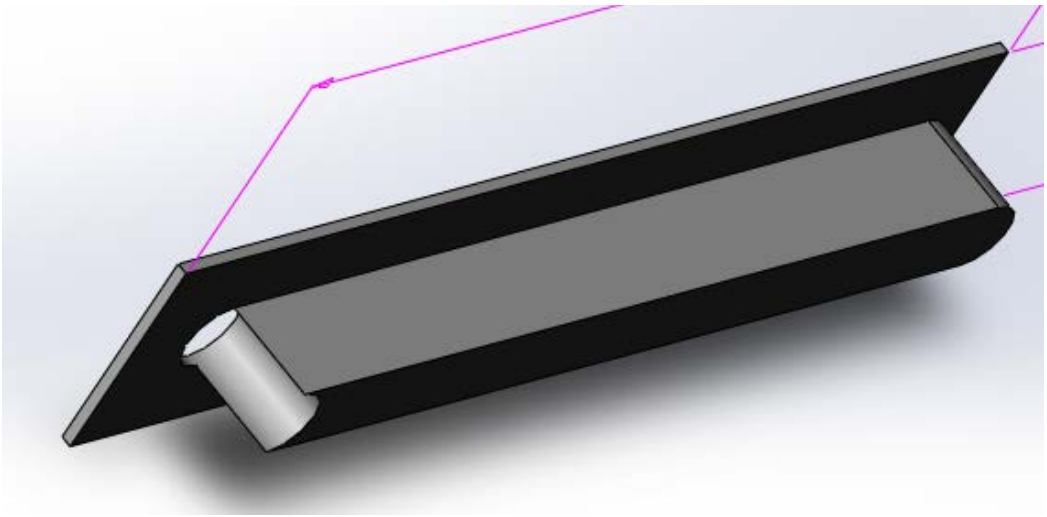


***Figure 6**: Top Seal*

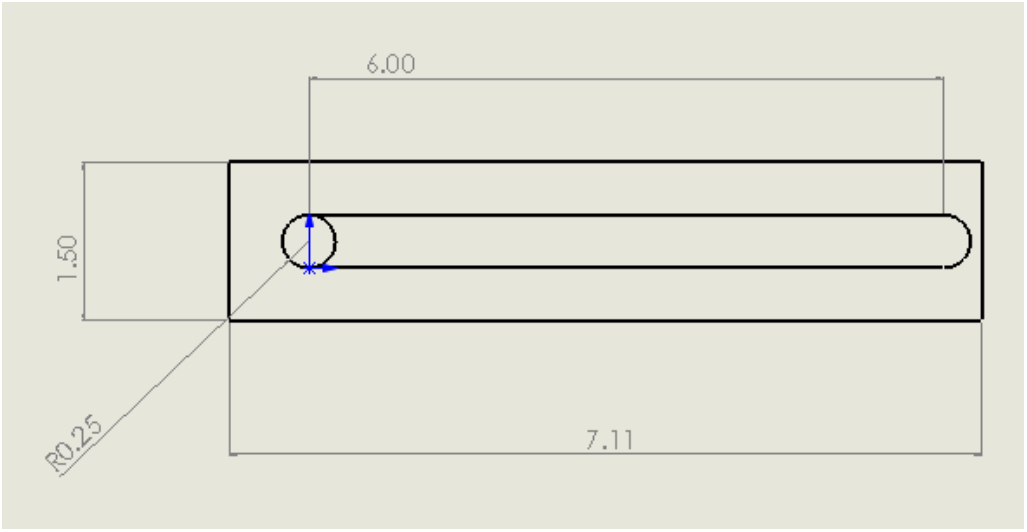


Figure 7: *Top Seal Dimensions*

To allow for free pitching motion in the wind tunnel, wall inserts were manufactured in order to have bearings press fit into them. The purpose of the threading on the wall inserts is to have a nut screwed on from the outside in order to clamp it to the test section wall. A threading of ¾"-16 was used to allow for a more secure attachment and for easier machining. Aluminum was the chosen material for these wall inserts as it allows for easy machining and the wing has a low enough weight were the aluminum would not reach its yield strength during tests. The thickness of the wind tunnel walls is 0.7" meaning that the gap between the head of the wall insert and the threading must be less than this value to ensure the nuts can reach the wind tunnel wall. Currently, the wall inserts that have been made have threading going further than 1" as a precaution that there was an error in the wall thickness measurement. However, during tests it was found that the wall thickness was 0.7" and therefore the 1" long threading in the model will perform fine if the wall inserts are ever remade.

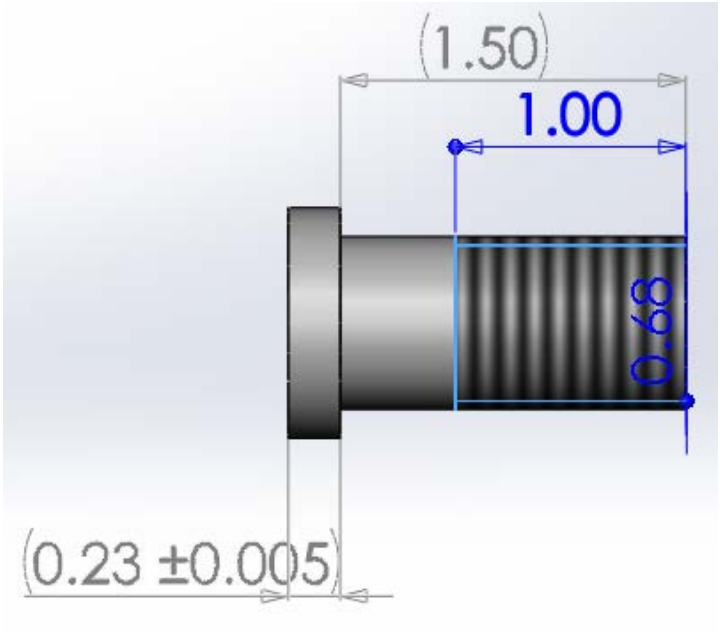


Figure 8: *Wall Insert Side View*

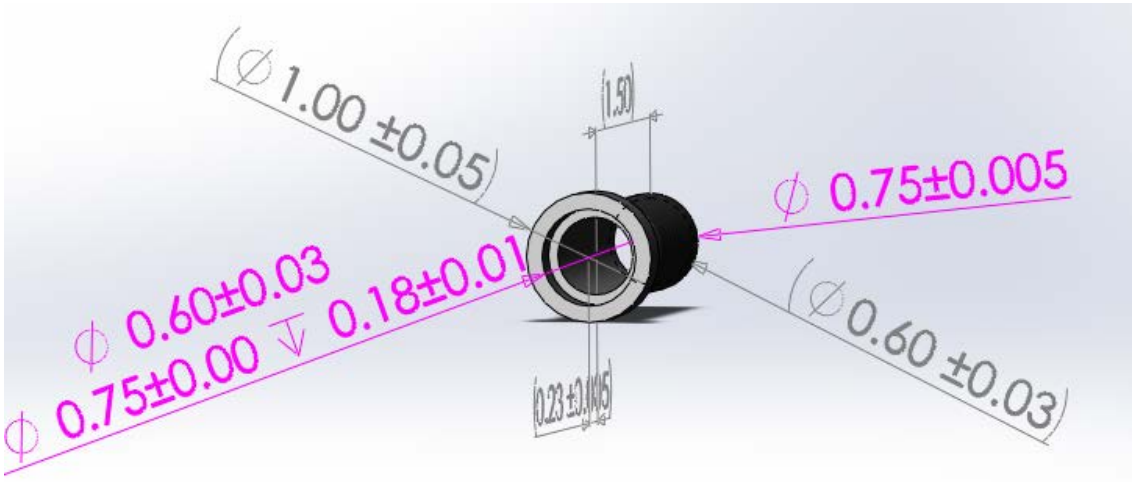


Figure 9: *Wall Insert Front View*

As for the bearings, 0.5” x 0.75” x 0.1562” bearings were utilized as they were one of the thinner options available while also being large enough to have the tube holding the wing slide through relatively easily. Although shown in documentation as 12”, the span and height of the wind tunnel are both closer to 11.5”. From this size constraint, ribs have to be removed from each size of the wing to decrease the span be a few millimeters to avoid the wing rubbing against the mounting system. The 0.5” tube has a wall thickness of 0.035” in order to leave as large of an inner diameter as possible to fit the wiring and the USB cable that runs through it. Due to the skin being on the wing during wind tunnel mounting, the 0.5” USB cable head has to be run through the bearing, however the plastic around a typical USB cable will make it too big to fit and therefore must be removed prior to mounting. As for the wing tubing itself, it is cute to be a total of 13.5” long. This ensures that the tube can be sufficiently pushed through both of the bearings in the wall inserts to avoid it sliding out during testing.

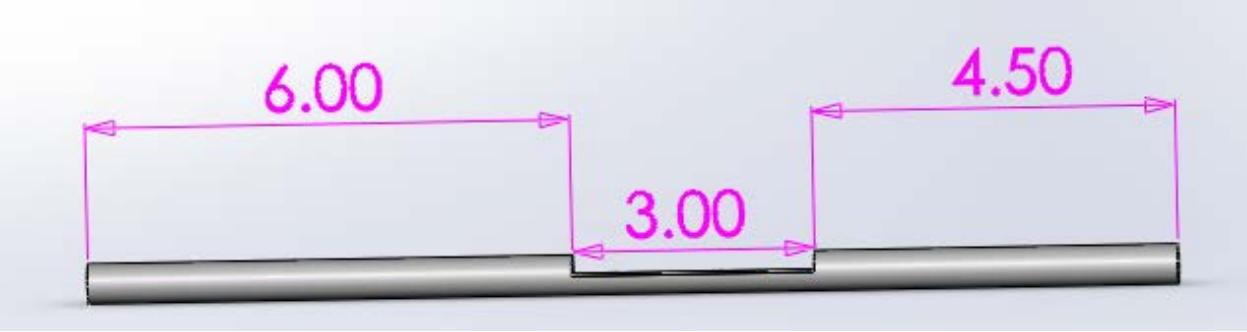


Figure 10*: Wing's Tube*

The sidewall mounting system accomplishes the goal of allowing free pitching motion by having the bearings press fit into the sockets and having the tube slide through. The tube is large enough to be a tight fit so that the wing only rotates and does not translate while small enough to allow relative ease in sliding the tube in and out of the mount. The nut and extended surface of the mounts press towards one another against the wall between them keeping the system secure. Because the tube of the wing is longer than the span of the wind tunnel, the best way to mount it is to slide the tube so it is only sticking out of one side of the wing. Slide this side into one of the wall inserts. Now the tube can be pushed from the outside of the wall insert so that it goes into the opposite wall insert. When sliding the tube, take caution not to move it too much to one side since the wires run through and opening in the tube and too much movement can knock the micro-USB or board loose within the wing.

Electronics and Control System Implementation

The requirements for the hardware design were developed to accommodate for the FS2 Thermal mass flow sensors used in this project (Figure 1). The minimum requirements were to accommodate for two sensors, this is to allow for the angle of attack measurement, however the system was designed such that the system could be expanded to multiple sensors by replicating the amplifier circuit for each desired sensor, this was done to allow for future measurements of other aerodynamic parameters including the prediction of stall.

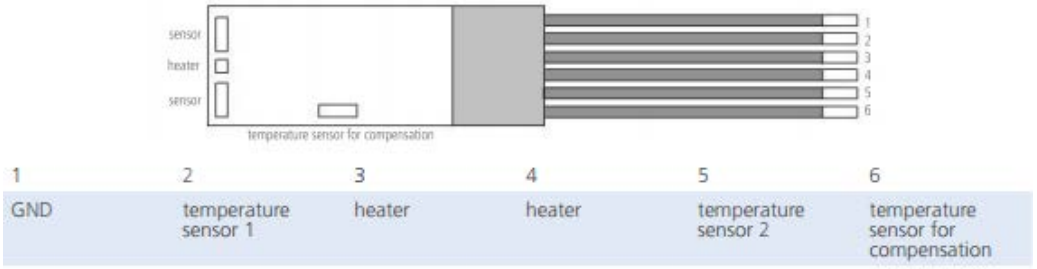


Figure 11: *FS2 Thermal Mass Flow Sensor Layout and Pin Configuration*

The FS2 flow sensor is a mass flow type sensor. It is analogous to a multiple wire hot-wire anemometer. The sensor works on the same principle as the hot wire anemometer which takes a constant temperature (CTA) or constant current (CCA) to operate the sensor. In CTA mode, the controller attempts to keep the wire at a temperature, as flow speed changes the voltage will also change to maintain the set temperature. This change in voltage can be correlated to the flow speed. In CCA mode the anemometer is provided with a constant current. Because of the constant current the temperature of the sensor is not limited, however due to the flow in the tunnel the temperature will equilibrate to a constant value.
This temperature can then be correlated to the flow speed.

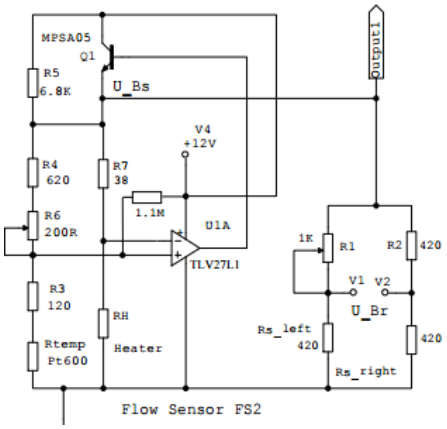


Figure 12: *Analog Amplifier CTA Circuit*

Similarly, the FS2 sensor, has a single heating element which can be operated in a CTA or CCA mode. For our experiment the sensors were operated in CTA mode, this mode was achieved through the analog circuit shown (Figure 2). The components used in this circuit can be found in Appendix E. The way the circuit functions, is by regulating the current to the heather (RH). This is done by using the comparator function of the Op-AMP (Operational Amplifier). While the circuit is open (the transistor is open) The voltage after the the potentiometer (R6) is larger than the voltage after R7 this means the voltage at the output of the opamp is aproximatley 12 Volts (The input of the op amp). Since this output is wired to the base of the transistor this injects current into the heater and increases the temperature on the heater. Since thhe heater is a resistor as it heats up its resistance decreases, when it decreases below a certain threshold the voltage on the negative input of the opamp grows to be larger than the positive input. When this happens the voltage at the output of the opamp goes to zero. This closes the transistor and stops heating the sensor. This system can be tuned by the potentiometers at R6 for coarse tuning, and R1 for fine tuning.

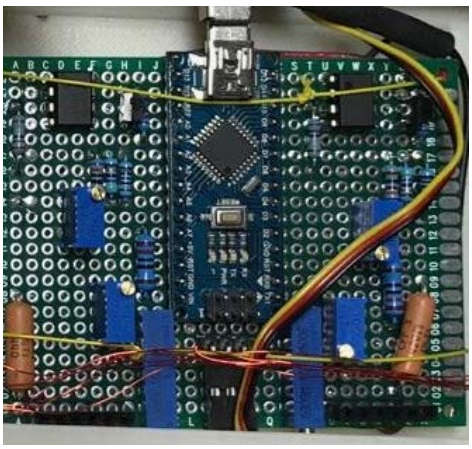


Figure 13: *Top down View of Protoboard with Arduino Nano*

To allow for the two sensors needed as per the requirements the circuit (Figure 2), was duplicated twice on a protyping board (Figure 3). The protoyping board was used because it is simple to add and remove components however due to soldering being required all components are held securely. Prior to final assembly the circuit was tested on a solderless breadboard to ensure function. As can be seen in Figure 3, the two circuits were seperated (spatially) by the Arduino Nano micro controller. For this iteration of the project the left circuit was used for sensor located on the top of the wing and was connected to A0 of the arduino, the sensor located on the bottom of the wing utilized the circuit on the right side of the board and was attached to A1 on the arduino. Between the two analog circuits was a 90-degree header for the servo pins. The servo was routed such that the power connected to the 5V pin on the arduino, ground to ground on the arduino, and control to digital pin 9.

The servo that was used was a Hitec HS-85MG Micro servo. One of the multiple requirements of the elctronic subsytem was its small form factor. This was the primary reason for choosing this servo, as it fit very well into the space that was allowed by the structural design. In addition to its small size it also ouput a large torque, which was required to engage the rotary drive system used for elevator deflection. Due to this motor being of the servo variety it was easily controlled through the servo library of the arduino software.

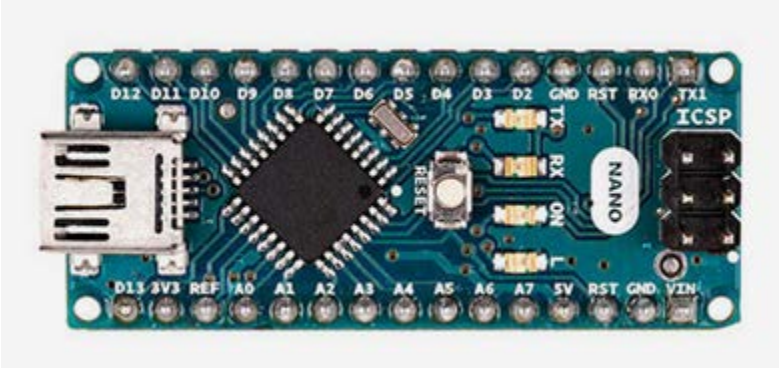


Figure 4: *Closeup of Arduino Nano V3*

The arduino series of micro controllers was an excellent choice for this project as it allowed for rapid prototyping of software. It was also extremly convenient for its USB interface in order to establish serial communications with MATLAB. Additionally the 5V power capabilities allowed for servo power without having to stepdown the voltage from the 12V power supply, and servo control was simple utilizing the servo library of the Arduino language.

Last but not least in the electronic hardware considerations was the power supply. As mentioned before the analog circuit for the sensors requires a 12V input to the Op-Amp, additionally the arduino requires power as well. In the case of the arduino, the usb mini-b cable which served as the data umbilical also doubled as the power source, powering both the microcontroller and the servo attached to it. As for the analog circuit that posed a slightly more difficult challenge. In perliminary designs a fully self contained system was proposed which would have required a 12V battery pack. This would have been a very heavy and difficult solution to implement as space is very limited inside of the wing. For the final iteration presented in this report

the power rails on the protoboard were turned into a ground and +12V rail. Leads were extended from these two rails and run, along side the USB umbilical, outside of the wing. The wind tunnel mounting mechanism was designed such that all 3 wires were able to fit through. The USB was connected to a laptop to power the arduino, while the power and ground leads from the proto board were connected to a bench top voltage/current source which was used in the voltage source mode.

Algorithm Description and Interface Document (Software)

To fulfill the requirements for the software portion of the electronics which included the control system portion of the project, a multi faceted approach was used. The software was broken up into two key components. The first was the arduino firmware, written in arduino, which was run on the Nano microctroller. This firmware collected and serial transmitted sensor data, as well as handled elevon deflection and control from an input signal. The second part of the software was run on a laptop and was written in MATLAB. It collected and applied a Low Pass filter (Figure _) to the incoming analog voltages collected from the sensors. It converted the difference of these voltages to an angle of attack through a polynomial fit, and ran this difference through the control algorithm to generate the elevon deflection which was sent to the microcontroller.

The Arduino firmware is a fairly simple but powerful interface. The code was written to read the analog signal off of pins A0 and A1 (Top and Bottom sensors) these two analog reads are combined and seperated with a common to get the form: TOP,BOTTOM, and streamed over the serial port as a single byte per set of sensor readings. Following the serial write, the code would listen to the serial port for any availiable bytes (elevon deflection signals) and if one was detected, would use the conversion of elevon deflection to servo deflection (Equation 1) and write the proper angle to the servo. This was looped for the duration that the arduino was plugged into the laptop. Although designed to interface with MATLAB the code also allowed for open loop control of the system, by using the arduino's built in serial monitor to send elevon deflection signals to the model. The entierity of the firmware can be seen in Appendix B.

$$\theta_s = \tan(\sin^{-1}\delta_s) \qquad \textbf{(1)}$$

On the other end of the serial communications was the laptop running the MATLAB software. The code began by initializing the arduino on a desired COM port, 3 by default. The code was parametrized with respect to the number of time steps allowed as well as the time steps themselves allowing for quick modifications of those parameters. A low pass filter was also initalized with the properties that can be seen in Table 1. In addition the controller and polynomial fit of the voltage difference were intialized. With the main functions of the software initalized, the software began reading the arduino serial channel as part of a loop. The loop was timed such that reading was synched with the arduino's write rate. For each iteration the sensor readings were read. This reading was low passed through the filter to remedy the noise of the readings. The difference of the filtered data was taken and fit to the angle of attack utilizing the polynomial fit. All of these quantities and their intermediates were plotted live in order to see the current status of the experiment on the screen. Finally the difference of the current angle of attack and the desire angle of attack was taken and fed into the transfer function of the control algoritm. The output of this function gave an elevon deflection angle in radians which was sent back to the arduino. Finally the data was logged to a file to enable easier data processing. The entierty of this code can be seen in Appendix C.

Cutoff Frequency:	10 rad/s or 1.5915 Hz
Transfer Function:	$\frac{1}{(0.1s+1)}$

Table 7*: Low Pass Filter Characteristics*

The purpose of the MATLAB model was to be able to predict the behavior of system and generate plots of angle of attack and position in order to verify the wind tunnel experiments. In order to do this, the longitudinal stability equations of motion for a glider were developed and simplified. This derivation is shown below.

Body Frame Orientation w.r.t Inertial Frame:

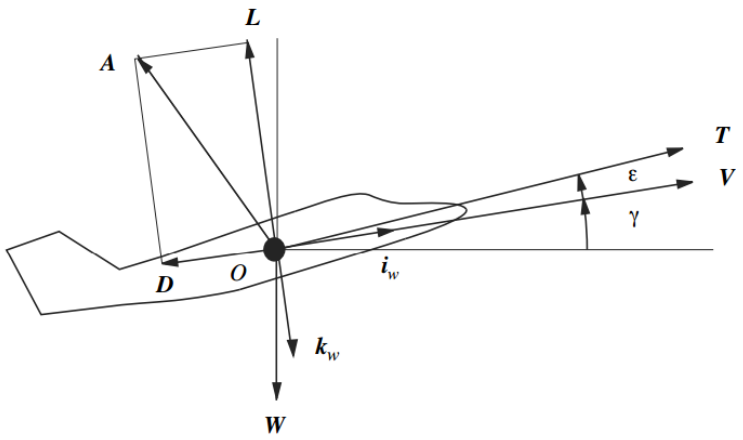


Figure 2.2: Forces Acting on an Airplane in Flight

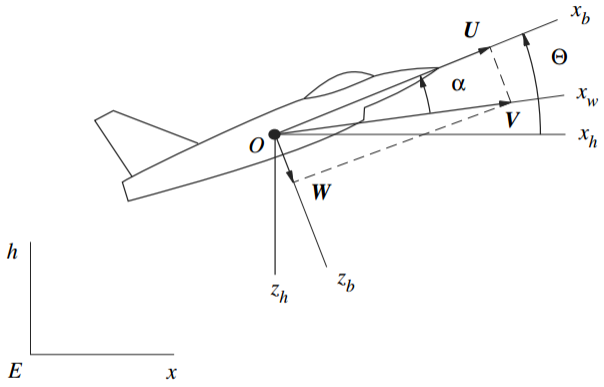


Figure 10.1: Body Axes System

Equations of Motion:

$$V = Ui_b + Wk_b$$

$$V = \sqrt{(U^2 + W^2)}$$

$$\tan\alpha = \frac{W}{U}$$

$$\begin{aligned}
\dot{u} &= (\Delta C_x + 2C_{x_1}\tfrac{u}{U_1})(\bar{q}_1 S/m) - g\theta \\
U_1\dot{\alpha} &= U_1 q + (\Delta C_z + 2C_{z_1}\tfrac{u}{U_1})(\bar{q}_1 S/m) \\
\dot{\theta} &= q \\
\dot{q} &= \Delta C_m \bar{q} S\bar{c}/I_{yy}.
\end{aligned}$$

Substitute in NON- Dimensional Stability Derivatives:

$$\begin{aligned}
\dot{u} &= \left[C_{x_\alpha}\alpha + C_{x_{\delta_E}}\delta_E + (C_{x_u} + 2C_{x_1})\frac{u}{U_1}\right. \\
&+ \left. C_{x_q}\frac{\bar{c}q}{2U_1} + C_{x_{\dot{\alpha}}}\frac{\bar{c}\dot{\alpha}}{2U_1}\right]\frac{\bar{q}_1 S}{m} - g\theta \\
U_1\dot{\alpha} &= U_1 q + \left[C_{z_\alpha}\alpha + C_{z_{\delta_E}}\delta_E + (C_{z_u} + 2C_{z_1})\frac{u}{U_1}\right. \\
&+ \left. C_{z_q}\frac{\bar{c}q}{2U_1} + C_{z_{\dot{\alpha}}}\frac{\bar{c}\dot{\alpha}}{2U_1}\right]\frac{\bar{q}_1 S}{m} \\
\dot{\theta} &= q \\
\dot{q} &= \left[C_{m_\alpha}\alpha + C_{m_{\delta_E}}\delta_E + (C_{m_u} + 2C_{m_1})\frac{u}{U_1}\right. \\
&+ \left. C_{m_q}\frac{\bar{c}q}{2U_1} + C_{m_{\dot{\alpha}}}\frac{\bar{c}\dot{\alpha}}{2U_1}\right]\frac{\bar{q}_1 S\bar{c}}{I_{yy}}
\end{aligned}$$

Rewrite using dimensional Stability Derivatives:

$$\begin{aligned}
\dot{u} &= X_u u + X_\alpha \alpha - g\theta \\
U_1\dot{\alpha} &= U_1\dot{\theta} + Z_u u + Z_\alpha\alpha + Z_{\dot{\alpha}}\dot{\alpha} + Z_q\dot{\theta} + Z_{\delta_E}\delta_E \\
\ddot{\theta} &= M_u u + M_\alpha\alpha + M_{\dot{\alpha}}\dot{\alpha} + M_q\dot{\theta} + M_{\delta_E}\delta_E
\end{aligned}$$

Where the Stability Derivatives are defined as:

$$
\begin{aligned}
X_\alpha &= \frac{\bar{q}_1 S C_{x_\alpha}}{m}\ \frac{ft}{s^2} & X_u &= \frac{\bar{q}_1 S(C_{x_u}+2C_{x_1})}{mU_1}\ \frac{1}{s} \\
X_{\dot{\alpha}} &= \frac{\bar{q}_1 S \bar{c} C_{x_{\dot{\alpha}}}}{2mU_1}\ \frac{ft}{s} & X_q &= \frac{\bar{q}_1 S \bar{c} C_{x_q}}{2mU_1}\ \frac{ft}{s} \\
X_{\delta_E} &= \frac{\bar{q}_1 S C_{x_{\delta_E}}}{m}\ \frac{ft}{s^2} & & \\
Z_\alpha &= \frac{\bar{q}_1 S C_{z_\alpha}}{m}\ \frac{ft}{s^2} & Z_u &= \frac{\bar{q}_1 S(C_{z_u}+2C_{z_1})}{mU_1}\ \frac{1}{s} \\
Z_{\dot{\alpha}} &= \frac{\bar{q}_1 S \bar{c} C_{z_{\dot{\alpha}}}}{2mU_1}\ \frac{ft}{s} & Z_q &= \frac{\bar{q}_1 S \bar{c} C_{z_q}}{2mU_1}\ \frac{ft}{s} \\
Z_{\delta_E} &= \frac{\bar{q}_1 S C_{z_{\delta_E}}}{m}\ \frac{ft}{s^2} & & \\
M_\alpha &= \frac{\bar{q}_1 S \bar{c} C_{m_\alpha}}{I_{yy}}\ \frac{1}{s^2} & M_u &= \frac{\bar{q}_1 S \bar{c}(C_{m_u}+2C_{m_1})}{I_{yy}U_1}\ \frac{1}{ft\ s} \\
M_{\dot{\alpha}} &= \frac{\bar{q}_1 S \bar{c}^2 C_{m_{\dot{\alpha}}}}{2I_{yy}U_1}\ \frac{1}{s} & M_q &= \frac{\bar{q}_1 S \bar{c}^2 C_{m_q}}{2I_{yy}U_1}\ \frac{1}{s} \\
M_{\delta_E} &= \frac{\bar{q}_1 S \bar{c} C_{m_{\delta_E}}}{I_{yy}}\ \frac{1}{s^2}. & &
\end{aligned}
$$

Assumptions:

- Derivation Assumed
  - Angles are small (alpha, theta, epsilon) therefore the derivation assumes small perturbation theory is valid.
  - Subscript (1) denotes initial condition.
  - Neglect the effects of derivative of angle of attack on drag and thrust.
- Simplifying Assumptions
  - Trajectory time of flight is small
  - Density, chord length, center of gravity, and moment of inertia are all constant
  - Linearize alpha equation to get: $\alpha = \frac{w}{U_1}$
  - Wind tunnel constraints: $\alpha = \theta$
  - Ignore z-force equation because the system is then fully defined with the moment and x-force equation.
  - No thrust (T=0)
  - Mu is negligible because we are at low Mach numbers.

Initial Conditions:

- Velocity

$$w_1 = 0$$
$$V_1 = U_1 = constant\ in\ range\ of\ 5 - 20\ m/s$$

- Angle of Attack/Pitch Angle

$$\alpha_1 = \theta_1 = 0$$

Equations of Motion:

X-force Equation:

$$\dot{u} = X_u u + \theta(X_\alpha - g)$$

Moment Equation:

$$\ddot{\theta} = M_u U + M_\alpha \theta + M_{\delta e} \Delta\delta_e$$

Conversion to Inertial Frame:

Using the coordinate system shown above:

$$V_{i1} = V_x = u\ cos\theta + w\ sin\theta$$

$$V_{i3} = V_y = Z = w\ cos\theta - u\ sin\theta$$

Using the simplified EOM given above a transfer function was derived for the case of pure pitching motion. Additionally, a sample calculation for the period of motion was also approximated.

Develop a Transfer Function using the Moment Equation:

- Neglect the effect of Mach Number → Mu = 0

$$\ddot{\theta} = M_q\dot{\theta} + M_\alpha\theta + M_{\delta e}\Delta\delta_e$$

$$H(s) = \frac{\theta(s)}{\Delta\delta_e(s)} = \frac{M_{\delta_e}}{s^2 - sM_q - M_\alpha}$$

Natural Undamped Frequency:

$$\omega_n = \sqrt{-M_\alpha}$$

Dampening Ratio:

$$\xi = -\frac{M_q}{2\,\omega_n}$$

These EOM are valid for any arbitrary glider design. The coefficients for the glider design before PDR and wing design after PDR were also developed using XFOIL and are provided in the appendix. However, due to the descope of the project requirements and main objective, the MATLAB Simulink model of these EOM was not used in the final product verification, however, since the code had been fully developed prior to the requirements descope the code is provided in the appendix as well as on the MEMS MAV Control USB drive. The Simulink code inputs all the longitudinal stability derivatives, and then solves the EOM using the math toolbox in Simulink, and then generates a plot of output results as a function of time.

Analysis

CFD Results

Two-dimensional computational fluid dynamic analyses were performed on a NACA 4412. The analyses were ran at velocities of 8 to 14 m/s at angles of attack of 0 to 15 degrees. It was hypothesized that when plotting the angle of attack versus the velocity difference at the leading edge that there would be a relationship described by *Equation 2*.

$$\alpha(\Delta V, C_{\alpha i}) = C_{\alpha_0} + C_{\alpha_1}\Delta V + C_{\alpha_2}\Delta V^2 + \cdots + C_{\alpha_n}\Delta V^n \qquad (2)$$

where ΔV is the velocity difference at a chosen percent chord, and $C_{\alpha i}$ are parameters that describe the fit and are dependent on the free stream velocity. Velocity data was extracted at 3% chord and the difference of the upper surface and lower surface was determined. Plotting the data and applying the nth order fit revealed a linear relationship between angle of attack and velocity difference. The results are shown in Figure 6.

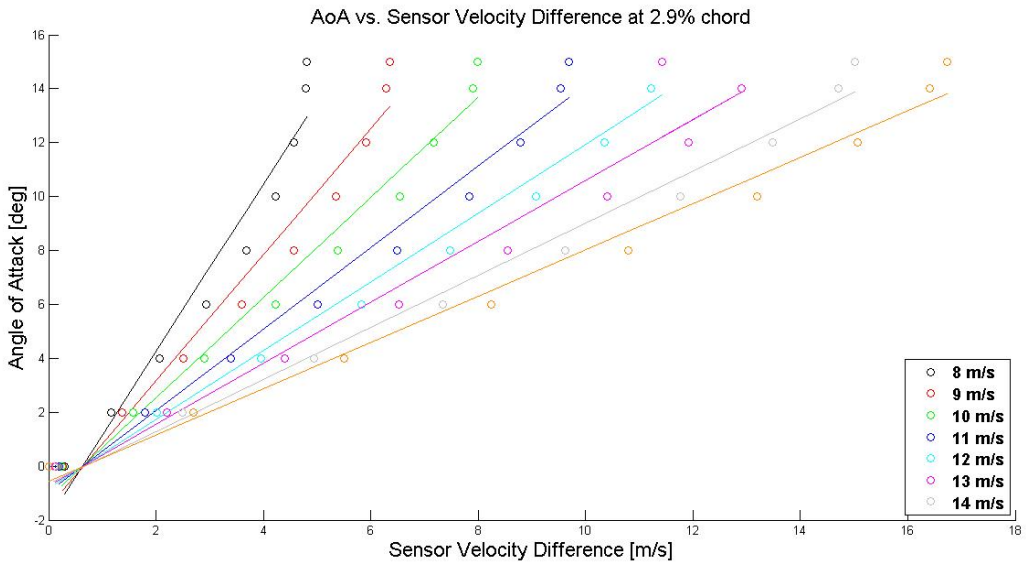


Figure 14: *Angle of Attack vs. Velocity Difference for Varying Velocities Using CFD*

The coefficients for the different linear fits are displayed in Table 3.

Velocity	C_{α_0}	C_{α_1}
8	-1.601	2.890
9	-1.240	2.208
10	-0.938	1.767
11	-0.756	1.442
12	-0.619	1.219
13	-0.500	1.052
14	-0.463	0.926
15	-0.391	0.822

Table 8: *Angle of Attack vs. Velocity Fit Coefficients*

Notable features of the plot in Figure 5 are the breakdown of the linear relationship at high angles of attack and the decrease in slope with the increase in velocity.

Similarly, the sum of the extracted velocity data can be taken and the free stream velocity can be plotted against this sum. Equation 3 was fit to the plots.

$$V_\infty(\sum V, C_{vi}) = C_{v_0} + C_{v_1}\sum V + C_{v_2}\sum V^2 + \cdots + C_{v_n}\sum V^n \quad (3)$$

where $\sum V$ is the sum of the leading edge velocities, and C_{vi} are parameters that describe the fit and are dependent on the angle of attack. Plotting the data and fitting revealed a linear relationship, and the results are shown in Figure 7.

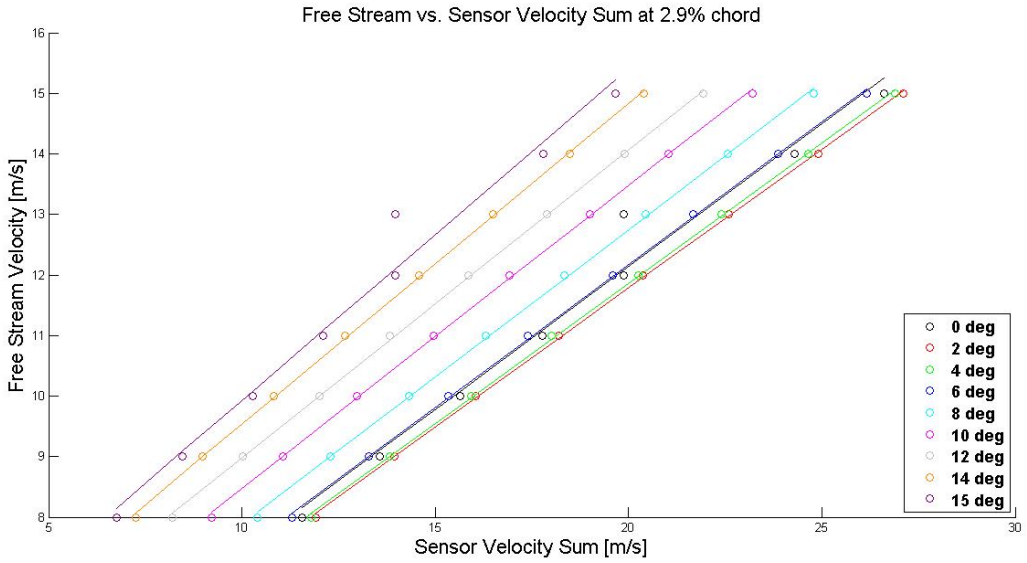


Figure 15*: Free Stream Velocity vs. Velocity Sum for Varying Angles of Attack using CFD*

The coefficients for the different fit equations are displayed in Table 3.

Alpha [deg]	C_{v_0}	C_{v_1}
0	2.336	0.490
2	2.287	0.474
4	2.290	0.478
6	2.402	0.488
8	2.629	0.507
10	3.021	0.526
12	3.411	0.537
14	3.779	0.559

Table 9*: Free Stream vs. Velocity Sum Fit Coefficients*

Wing Design and Coefficients

When first sizing the elevon, it has to be determined how much control the user desires, specifically what angles of attack the user wants the elevon to be able to handle. This is determined through a sum of moment s balance and setting the sum equal to zero.

To develop the transfer function for the control system, the wing will be treated as two separate parts, the elevon (acting as a tail) and the remainder of the wing as the main wing. This method helps alleviate the issue of attempting to calculate the change in lift of the entire wing due to elevon deflection. The elevon can be treated as a tail where its entire surface is the control surface and that the elevon has its own AC with force properties.

Note: whenever a variable is referenced to the wing, it is only the wing, the elevon is not included in that defined variable.

Pseudo Code for Elevon Sizing
Inputs are taken for :

c = chord of wing (m)
xac_w = aerodynamic center of the wing from leading edge (m)
ce = elevon chord (m)
δe = maximum elevon deflection (rad)
S = planform area of the wing (m2)
Se = planform area of the elevon (m2)
AR = aspect ratio of wing
Cm_0 = moment coefficient independent of angle of attack
CL_α = lift coefficient of wing per radian of angle of attack
CL_δe = lift coefficient of elevon per radian of elevon deflection

Calculations worked by code:
xac_e = aerodynamic center of elevon from leading edge of wing (m)
VH = tail volume
CM_α = moment coefficient of wing per radian of angle of attack
CM_δe = moment coefficient of elevon per radian of elevon deflection

The code then proceeds to determine the maximum and minimum angle of attack controllable by the current elevon size based on the negative and positive deflections of the elevon. Both results are displayed in the command window in radians.

Derivation of Transfer Function
Assumptions:
The elevon is considered a tail with a 100% control surface: $\tau = 1$
For the wing tunnel: $\theta = \alpha$

Determining Transfer Function:

$$\ddot{\alpha} - \left(M_{\dot{\alpha}} + M_q\right)\dot{\alpha} - M_\alpha \alpha - M_{\delta e}\delta_e = 0$$

From a Laplace transform:

$$\frac{\alpha(s)}{\delta_e(s)} = \frac{M_{\delta e}}{s^2 - \left(M_{\dot{\alpha}} + M_q\right)s - M_\alpha}$$

$M_{\delta e}$: Moment due to elevon deflection
$M_{\dot{\alpha}}$: Pitching moment with respect to rate of change of angle of attack
M_q: Pitching moment with respect to pitching velocity
M_α: Pitching moment due to current angle of attack

Calculation of various moments:

$$QS_e C_{L_{\alpha_e}}\alpha + QS_w C_{L_{\alpha_w}}\alpha = QSC_{L_\alpha}\alpha$$

$$C_{L_{\alpha_w}} = \frac{SC_{L_\alpha} - S_e C_{L_{\alpha_e}}}{S_w}$$

$$M_{\dot{\alpha}} = C_{m_{\dot{\alpha}}}\left(\frac{\bar{c}}{2u_0}\right)\frac{QS_w\bar{c}}{I_{yy}}$$

$$C_{m_{\dot{\alpha}}} = -2\eta C_{L_{\alpha_e}} V_H \frac{\ell_t}{c}\left(\frac{2C_{L_{\alpha_w}}}{AR}\right)$$

$$M_q = C_{m_q}\left(\frac{\bar{c}}{2u_0}\right)\frac{QS_w\bar{c}}{I_{yy}}$$

$$C_{m_q} = -2\eta C_{L_{\alpha_e}} V_H \frac{\ell_t}{c}$$

$$M_\alpha = C_{m_\alpha}\frac{QS_w\bar{c}}{I_{yy}}$$

$$C_{m_\alpha} = C_{L_{\alpha_w}}\left(\frac{x_{cg}}{\bar{c}} - \frac{x_{ac}}{\bar{c}}\right) - \eta V_H C_{L_{\alpha_e}}\left(1 - \frac{2C_{L_{\alpha_w}}}{AR}\right)$$

$$M_{\delta e} = C_{m_{\delta e}}\frac{QS_w\bar{c}}{I_{yy}}$$

$$C_{m_{\delta e}} = \eta V_H C_{L_{\alpha_e}}\tau$$

$$V_H = \frac{\ell_t S_e}{S_w\bar{c}}$$

Various Wing Parameters

Parameter	Measured Value
c	.144 m
ℓ_t	0.09575 m
Q	85.888 Pa
AR	2.0139
VH	0.1875

Calculated Coefficients

Coefficient	Calculated Value
CL_αw	1.576/rad
CL_δe & CL_αt	2.9794/rad
$C_{m_{\dot{\alpha}}}$	-0.37/rad
C_{m_q}	-0.7429/rad
C_{m_α}	-0.3152/rad
$C_{m_{\delta e}}$	-0.5586/rad

Calculated Moments

Moment	Calculated Value
$M_{\dot{\alpha}}$	-0.500/s
M_q	-1.004/s
M_α	-73.606/s2
$M_{\delta e}$	-130.45/s2

Elevon Deflection Relationship

$$\delta_e = tan^{-1}(\sin(\theta))$$

δ_e: deflection angle of elevon
θ: rotation angle of servo

Development Plan and Implementation

In the critical design review, the plan to construct the wing was to use balsa wood for the outer portion of the wing while using a 3D printer to print the tray and the elevon. The plan for printing was to use the 3D printer in the Aerospace Engineering Machine Shop using ABS material. To cover the wing, a thin sheet of monokote and heatform it to the wing to give it a smoother surface to not introduce any flow disturbances that the 3D printed or balsa wood might have introduced. The general idea from the CDR was kept but there were some changes to the model as well as changes to the materials that were initially planned to be used for the construction of the wing. This process was an iterative process where the design was created then the idea was created. This lead to creating several different iterations of the wing with modifications done to each until the final product was created.

The first iteration was more of a test to get familiar with the materials being used as well as the instrumentation needed to create each part. During this iteration, the laser cutter was used to cut the balsa wood ribs into the desired airfoil shape. The hardest part of this process was to determine the proper settings for the laser cutter without causing the balsa wood to catch on fire while also cutting all the way through. It was found that using the setting of .130 inches to cut the .125 while also keeping the vector cutting at around 0-2% would cut it ideally. There might be some flames that appear when the laser cutter is cutting the balsa wood but that is ok as long as it is not too intense that it would impact the effective shape of the rib. There were two sets of ribs cut using the laser cutter, hollow ribs that go in the inner portion of the wing while the outer ribs were cut with the shape of the airfoil with two holes for the larger rod as well as the elevon rod. Since the balsa wood was cut at ⅛ inch thick sheets, some were combined by gluing them together. This was done by using the outlines of the cutouts to make them as straight as possible. Engineering drawings of these can be found on the following page.

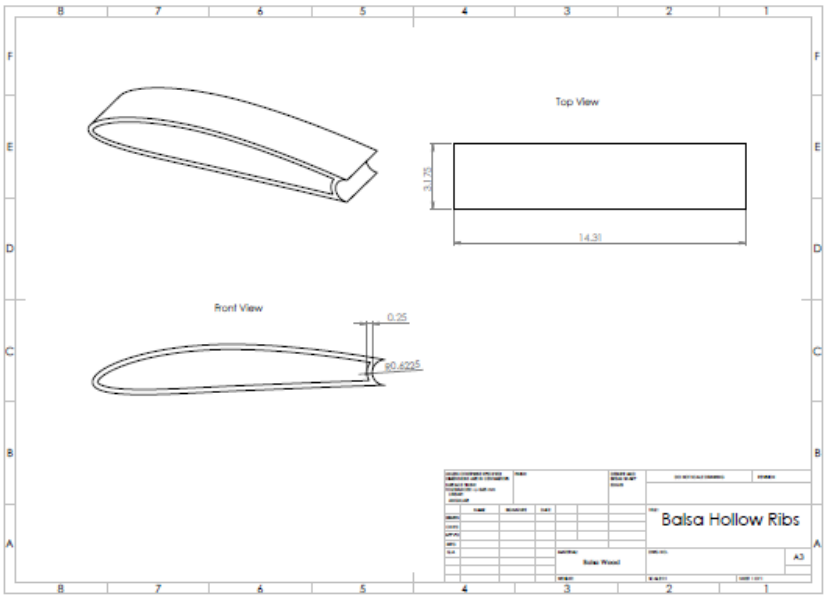


Figure 16: *Balsa Hollow Ribs Engineering Drawing*

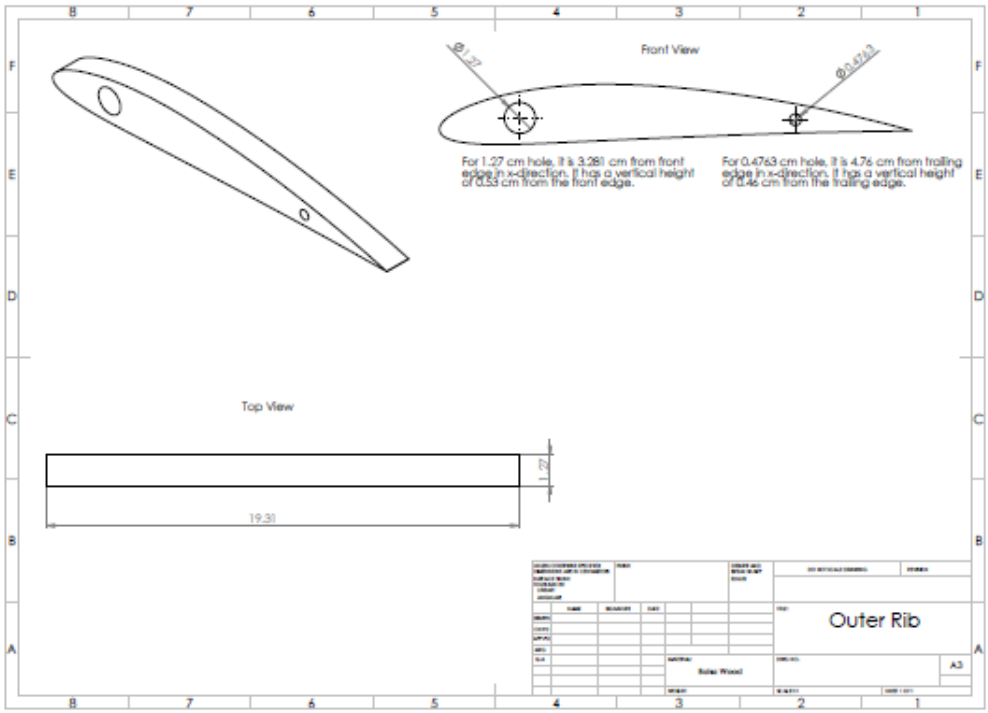


Figure 17: *Outer Rib Engineering Drawing*

The initial iteration of the inner portion of the wing that would hold the electronics, also called the tray, was printed using the AME Machine shop 3D printer. This print was made to have a portion of the leading edge and have a completely solid bottom portion until the trailing edge where it was just cut off. Unfortunately, since this middle portion was too large for the printer to handle, it was split up into two different portions. When combining the first iteration of the wing, instead of placing the tray in the middle and all the ribs towards the sides, some of the hollow ribs were placed between the tray to make it so the monokote would be more easily formed around the wing components without drooping. This unfortunately did not solve this problem so a new approach was taken for the tray construction. At this point, the elevon was not printed.

The second iteration of the tray took the same approach as the first but instead of having a completely open back portion, it had a portion near the trailing edge to help with the process of skinning the wing. The second iteration of the tray was printed using Dr. Enikov's 3D printer in his lab to decrease the cost of printing. This used PLA instead of ABS as the material which at the time was not a problem. The tray was printed with the new design using the new printer and at this point the elevon was printed. The elevon was designed to be fully hollow so the elevon rod would extend from one side to the other. The elevon also had to be printed in two separate pieces because of the length of it. After these were finished printing, the wind tunnel mounting system was printed using the same printer. The two pieces included in this were the top portion that would seal the wind tunnel and to hold the center rod in place and the other was a mounting system, also called the shoe, that was used in the calibration tests. Unfortunately there was sizing problems with the shoe and the seal so they needed to be sanded. The shoe needed to have a hole drilled through it to extend the trailing edge and an end mill was run through it in order to increase the space for the wing to fit in. This problem was determined to be a printer error and not the error of the Solidworks models. When skinning the elevon and the wing sing the monokote, there was a problem. When heat forming the monokote to the elevon, the elevon began to warp. This was not expected because previously the ABS had the monokote heat formed to it. There was also warping done to the tray so both needed to be scraped and needed to be printed again.

The next iteration of the tray and elevon was printed with ABS to prevent any warping. ABS plastic has a higher glass transition temperature and hence is not susceptible to warping when heat is applied to it. To get rid of the possibility of any warp occurring, Monokote was replaced with stainless steel shim stock. Cyanoacrylate glue was used to skin the steel to the tray and elevon. A trap door was added to the new tray so the electronics could be easily accessed. The ABS printed elevon was made smaller so it would fit the wing nicely. Regrettably, a part of the elevon broke due to carelessness. A new elevon was needed and the team used this opportunity to update the elevon to have a hole for the servo arm to go in instead of a slot and to have a larger span to fit the wing snugly. The elevon turned out to be too long span wise and therefore some milling was done at the elevons end. After the elevon was done printing, the ABS tray and balsa wood parts was glued together while the elevon was being acetone welded. All the electronics, including the servo, was then placed inside the wing. It was found later that the hole mechanism of the elevon for the servo arm to go through created more machining complications as the servo arm was made of steel. With that, one last iteration was made to the elevon where an updated version of the slot was implemented into the design.

After the elevon was printed, the wing entered its final assembling stage as the protoboard, servo, servo arm, counter weight and all the wires was placed strategically to fit the needs of the experiment. To skin the wing, a template of the wing was drawn and cut using a shear. Since, the wing is not completely symmetrical, the skin of the wing was divided into 4 parts being two for the outer rib region and two for the inner rib and tray region. Skinning the wing into 4 different parts decreases the possibility of having an air bubble underneath the skin which is highly unfavorable. The skin was then glued to the wing. Since the elevon had a smaller surface area, the shim stock was able to cover the whole elevon entirely. Added that the elevon was symmetrical, skin of the elevon was not divided into multiple parts. Engineering drawings of the final product and all of the parts can be found in the appendix section. A picture of the final product in the wind tunnel can be found below.

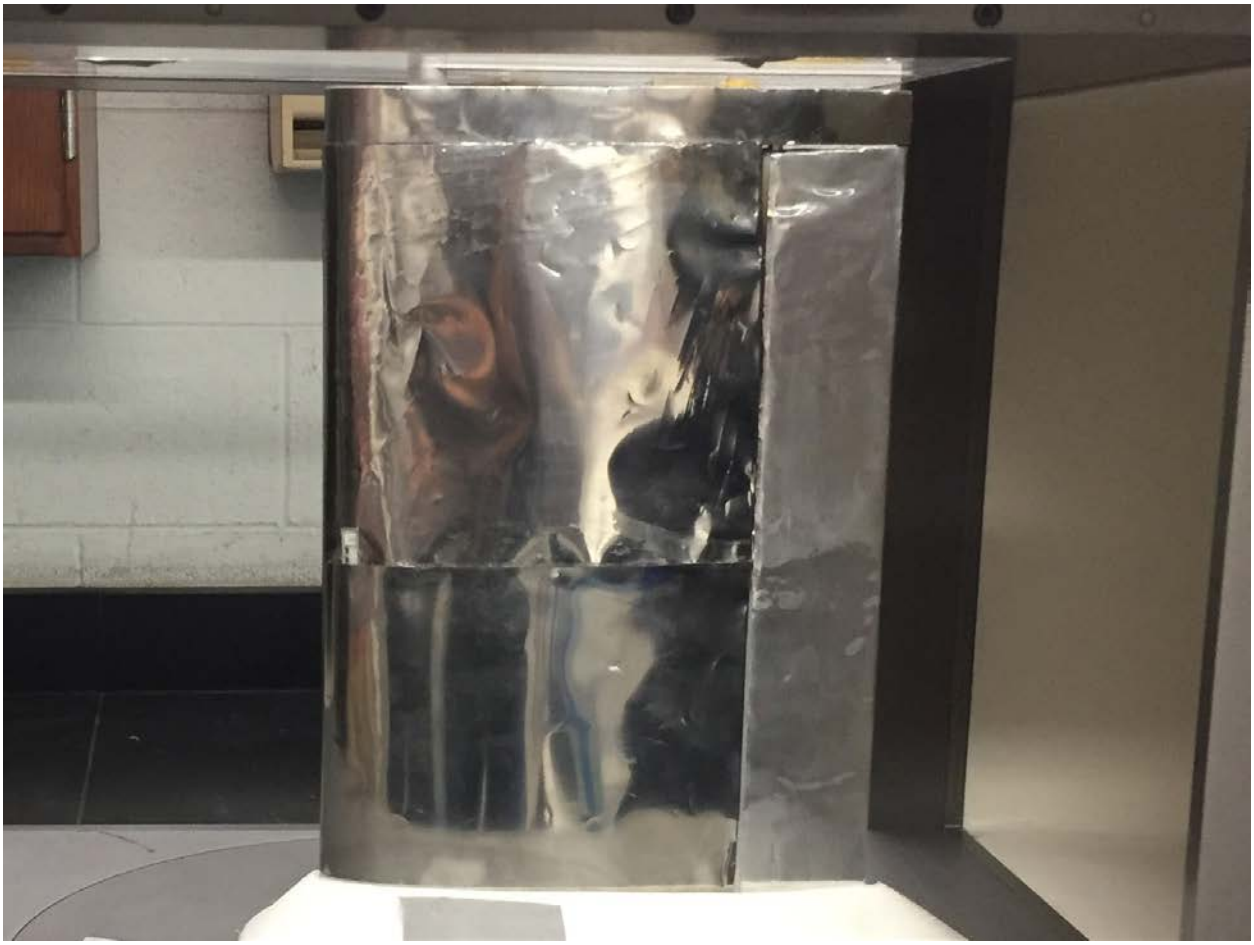


Figure 18: *Final Model of Wing with Electronics Implementation*

Requirements Review and Acceptance Test Performance

Acceptance Tests: Wind Tunnel Testing

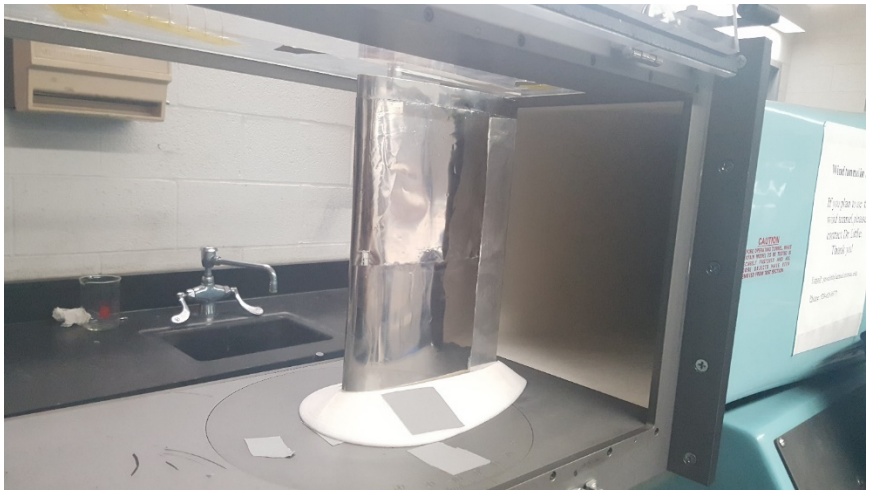
Wind tunnel testing was performed in The University of Arizona's Educational Wind Tunnel. The objective of wind tunnel testing was to develop relationships between the sensor voltage output, the free stream velocity, and the angle of attack. Furthermore, wind tunnel testing was a platform to demonstrate closed loop control of the MEMS sensor control system.

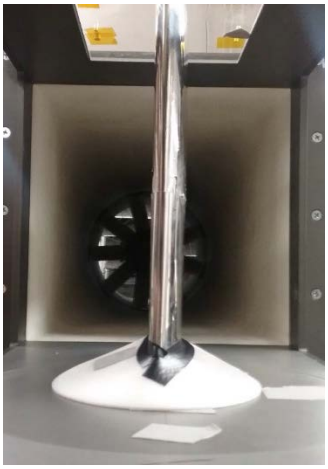
To perform wind tunnel testing, the Aerolab Wind Tunnel procedures were followed. The wind tunnel correction factor and room temperature was used to determine the pressure in millibar for each Reynold's number tested. These values were then used to control the free stream velocity in the wind tunnel. A total of four experiments were performed which are described in Table XXX. The first two experiments performed were calibration experiments with the goal of developing a relationship between the sensor outputs, free stream velocity and the angle of attack. The calibration experiments mounted the wing in the vertical mounting system which was bolted to the base of the wind tunnel. For the flow visualization experiment and the free controlled flight experiment the wing section was mounted horizontally with a bearing fixture that was connected to the sides of the wind tunnel wall.

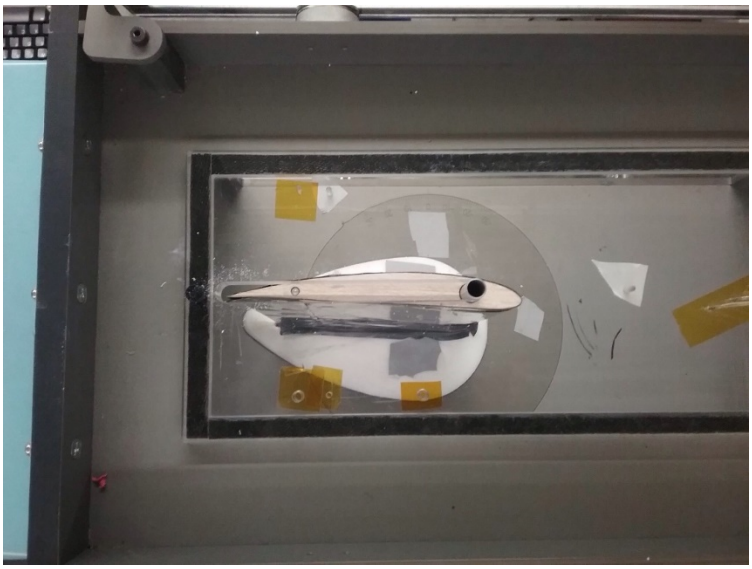
Test Performed	Constant Parameters	Variables	Purpose
Flow Visualization	α & V_∞	N/A	To visualize the air flow around the wing and compare to CFD Analysis
Flow Sensor Speed Calibration	$\alpha = 0°$	V_∞ = [4, 12] m/s	Confirm there is a linear correlation between sensor potential reading and flow velocity
Flow Sensor Angle of Attack Calibration	V_∞ = 10 m/s	α = [-4, 16°]	Confirm correlation between measured sensor potential difference, α, and V_∞. Validate results to CFD correlations
Controlled Free Flight	V_∞	α	Test the controller ability to establish stable pitch trajectory

Table 10: *Wind Tunnel Experiments Performed*

The following images show the wind tunnel with the wing section set up for the calibration experiments in the vertical mounting system.







Test Results

The angle of attack was plotted against the sensor difference data that was collected. A first order equation Equation 4 was then fit to the data in order to confirm that a linear relationship did exist.

$$\alpha(\Delta V,\ C_{\alpha i}) = C_{\alpha 0} + C_{\alpha 1}\Delta V \qquad (4)$$

where ΔV is the sensor reading voltage difference, and $C_{\alpha i}$ are parameters that describe the fit and are dependent on the Reynolds Number. *Figure 1* displays the results.

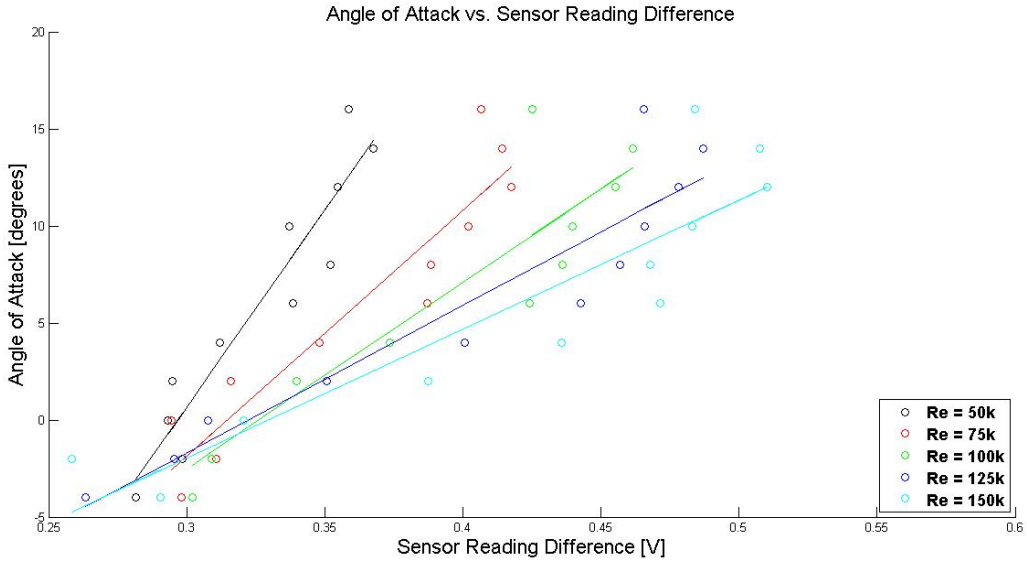


Figure 19*: Angle of Attack vs. Sensor Reading Difference at Various Reynolds Numbers*

As predicted by the CFD analysis, at high angles of attack, the linear relationship breaks down. Performing a linear fit to the range of angles of attack that fall within the linear range produced the following coefficients for the fit equation (*Table 1*).

Re	$C_{\alpha 0}$	$C_{\alpha 1}$
50000	-47.373	164.13
75000	-30.385	100.77
100000	-25.876	81.223
125000	-23.176	71.218
150000	-24.136	69.5

Table 11*: AoA vs. Sensor Difference Linear Fit Equation Coefficients*

Similarly, the Reynolds number was plotted against the sum of the data collected. Equation 5, was then fit to the data and yielded the results shown in Figure 13.

$$\mathrm{Re}(\textstyle\sum V\ ,\ C_{Vi}) = C_{V0} + C_{V1}\sum V \qquad (5)$$

Where $\sum V$ is the sensor reading voltage sum, and C_{Vi} are parameters that describe the fit and are dependent on the angle of attack.

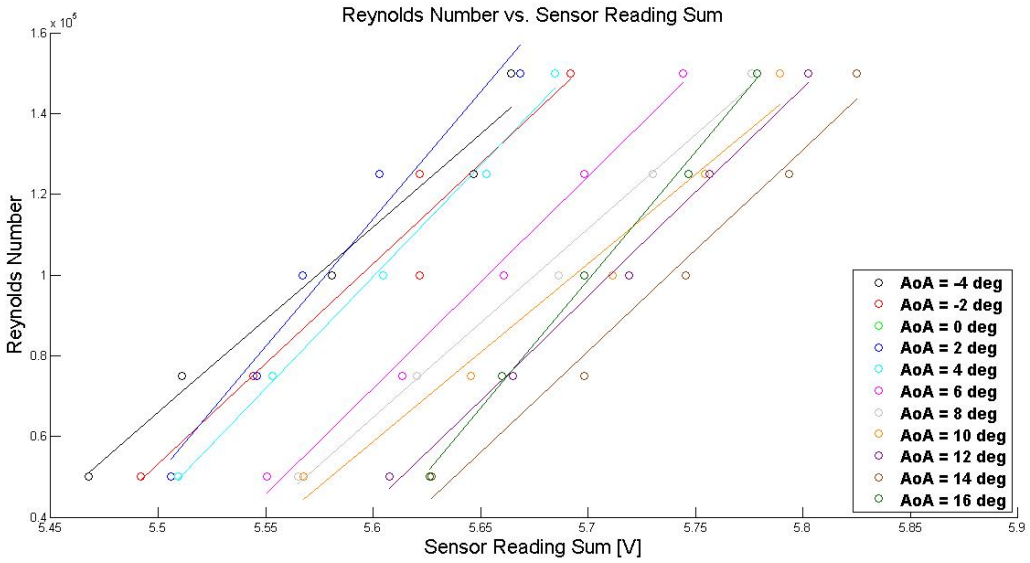


Figure 20: *Reynolds Number vs. Sensor Reading Sum for Varying Angles of Attack*

The coefficients for the fit are shown in Table 12 below.

Alpha [deg]	C_{V0}	C_{V1}
-4	-3.00E+06	517467
-2	-2.00E+06	459793
0	-3.00E+06	497271
2	-3.00E+06	630250
4	-3.00E+06	552284
6	-3.00E+06	525556
8	-3.00E+06	467388
10	-2.00E+06	441033
12	-3.00E+06	516806
14	-3.00E+06	499994
16	-4.00E+06	635914

Table 12: *Reynolds Number vs. Sensor Sum Linear Fit Coefficients*

The above results were noisy which resulted in skewed linear fits. However some trends can be pointed out that are similar to the results of the CFD analysis. First, as predicted by the CFD, the linear relationship breaks down at high angles of attack. This is most likely due to flow separation over the leading edge, causing the sensors to see a flow that does not properly represent the angle that the wing is flying at. Second, as the Reynolds number increases, the slope of the fit curve decreases. This is also observed in the CFD analysis. The Reynolds Number versus sensor reading sum plot confirms the CFD results showing that there is a linear relationship. This data will be useful when determining which equation to use for the angle of attack determination in a free flight scenario with changing free streams. With more collected data, it would be possible to determine a single relationship between the sensor readings and the freestream and angle of attack.

The controlled free flight demonstration was successful in open loop feedback control of the system. MATLAB was used successfully to input a desired elevon deflection angle and the corresponding change in the angle of attack was observed. FloSense did experience technical difficulties due to the controller software. The software would not always properly read the sensor data and was unable to consistently control the elevon deflection angle to produce the desired angle of attack. Due to these challenges in the software, a closed loop control demonstration was not performed due to the high risk of damaging the wing systems as well as the wind tunnel itself. FloSense believes the software interface, specifically the interface between the Arduino IDE code and MATLAB controller code need to be reevaluated to determine the root cause of malfunction. For lessons learned, interfacing multiple software programs is not the best solution. Furthermore, the team was limited in electrical engineering and control software knowledge and thus spent a lot of time to learn the software and hardware to design the controller.

The flow visualization used open loop control of the system at a Reynold number of 150,000 (10m/s) to show the airflow around the wing. A video of open loop control of the wing section as well as the flow visualization video are provided in the MEMS USB drive. In the flow visualization video, at high angles of attack flow separation can be observed which further supports the deviation in linear trend in the calibration results.

The full collected data spreadsheet is provided on the MEMS USB Drive.

The requirements were evaluated based on the final product produced. The verification and justification is given in the table below for the subsystem level requirements.

System Level Requirement Name	Description	Verification Method			Was Verification Criteria Met?	Justification
		Analysis	Inspection	Test		
1. MEMS Sensor Control System	The MEMS sensor control system shall be able to detect and characterize the angle of attack to be able to trim the wing in by appropriately adjusting the elevon angle.	x	x	x	Yes	The wind tunnel testing calibration results did demonstrate the relationship between the sensors and the angle of attack and this relationship matched the CFD analysis. Additionally, through the free flight demonstration experiment the wing was able to successfully trim and establish pitch stability using open loop feedback control.
2. MEMS Sensor System Wind Tunnel Testing	The MEMS sensor system shall successfully demonstrate functionality on a MAV wing	x	x	x	Conditionally	through the free flight demonstration experiment the wing was able to successfully

	section in a wind tunnel test.					trim and establish pitch stability using open loop feedback control. The control system was unable to demonstrate closed loop control due to software malfunctions. Due to this, the software will be reevaluated in the future to build and provide a more robust control system.

Table 13*: System Level Requirements Verification*

MEMS Controller Subsystem Requirement Name	Description	Verification Method			Was Verification Criteria Met?	Justification
		Analysis	Inspection	Test		
1.1 MEMS Sensor System Outputs	The sensor system outputs shall be verified by comparing results to CFD results.	x		x	Yes	The sensor system wind tunnel results were compared to the CFD analysis and showed similarity in the trends. The results of this comparison are provided in the Acceptance Test Section of this report.
1.2 MEMS Sensor System Interface	The MEMS sensor system shall not affect the airfoil flight parameters or the airfoil boundary layer.	x		x	Yes	Analysis was performed to determine the effect of the sensors on the boundary layer flow. Due to the small size of the sensors, the sensors did not affect the boundary layer flow. This also verified in the flow visualization experimentation.
1.3 MEMS Sensor System Outputs to Stall Adjustment	The MEMS sensor system outputs shall be used to accurately determine and change the elevator angle to maintain the MAV's pitch stability in a	x		x	Conditionally	The MEMS sensor system controller interface software in MATLAB could determine the change in the elevon angle its effect on the free stream velocity. The software does have bugs in it and thus closed loop control was not

	closed feedback control system.					demonstrated but open loop control was successful when the software was adequately collecting data.
1.4 MEMS Sensor System Size	The MEMS sensor system shall use the least number of sensors to sufficiently characterize stall and flight parameters successfully.	x			Yes	The sensor system was composed of 2 sensors which is the simplest sensor configuration for control using MEMS sensors. This configuration and its corresponding relationships were developed in the CFD analysis.

Table 14*: MEMS Controller Subsystem Requirements Verification*

Wind Tunnel Testing Subsystem Requirement Name	Description	Verification Method			Was Verification Criteria Met?	Justification
		Analysis	Inspection	Test		
2.1 Wind Tunnel Testing Results	The wind tunnel test results shall be used to develop the relationship between the sensor output, the angle of attack, and free stream velocity.		x	x	Yes	The sensor system wind tunnel calibration results were compared to the CFD analysis and showed similarity in the trends. The results of this comparison are provided in the Acceptance Test Section of this report. The data did show some noise; however, a strong correlation was still determined.
2.2 Wind Tunnel Testing Data Collection Interface	The sensor system shall have a MATLAB Interface to conduct wind tunnel testing.		x	x	Yes	The controller was developed in MATLAB with software code provided in the Appendix.
2.3 Wind Tunnel Testing Procedures	Wind tunnel testing procedures shall be developed		x		Yes	Procedures and the tests to be performed were discussed and developed with Robert Jacobi

	prior to testing.					one of the advisors for the Flosense team. The testing procedures was approved by Robert Jacobi as well.
2.4 Wind Tunnel Testing Hardware Interface	The wing section shall have a wind tunnel testing hardware interface that is not damaging to the Educational Wind Tunnel and is approved by Dr. Little	x	x		Yes	Flosense developed a vertical and horizontal mounting system for the wind tunnel that was approved by Dr. Jesse Little.

Table 15*: Wind Tunnel Testing Subsystem Level Requirements*

MAV Wing Section Subsystem Requirements	Description	Verification Method			Was Verification Criteria Met?	Justification
		Analysis	Inspection	Test		
3.1 MAV Sizing Requirement	The MAV shall be able to fit within University of Arizona Education Wind Tunnel which has dimensions of 30.5 cm x 30.5 ft x 61 cm.	x	x		Yes	The wing was designed to have a span of 29 cm and chord length of 19.3 cm. The wing span dimensions was selected to span the entire wind tunnel length and make the 2D flow approximation for the CFD analysis, and the chord length was selected to ensure the desired AR requirement was met.
3.2 MAV Airfoil Selection	The MAV shall use a standard airfoil for the wing.	x	x		Yes	A NACA 4412 airfoil was used for the elevon and wing design. This selection was made when the project was oriented towards a glider design.
3.3 MAV Wing	The MAV shall use a simple wing	x	x		Yes	The wing used a rectangular planform for

Geometry and Aspect Ratio	geometry and shall have a minimum aspect ratio of 1.25. This requirement is based on literature research.					simplicity in the wind tunnel testing. Additionally, the final AR of the design was 1.5. This value was selected based on literature.

Table 16*: MAV Wing Section Subsystem Requirements*

In conclusion, overall the requirements were mostly satisfied except for performing closed loop control demonstration in the wind tunnel testing. For future improvements of the project, a more robust software program needs to be developed for the control system compared to what was implemented in this project. Other forms of controllers should also be evaluated and considered in the controller implementation. This projected focused on finding a classical control system that satisfied requirements. Future improvements for this project if the team was allocated more time would be to refine the software for the control system.

Closure

The team's greatest accomplishments and successes for this project was obtaining the correct correlations between Reynolds Numbers and angles of attack, verifying the computational fluid dynamics analysis completed in FLUENT and XFOIL. The team was also able to implement the system into the wind tunnel with little to no issues,

In future iterations of the project, the team would recommend a couple of modifications and slight changes to elements of the electronics. As well as a couple of suggestions for easier operation and better results. The first recommendation is for the sensor/model interface. The sensors should have recesses in the structure with small holes for wire routing this would allow for easier positioning as well as a simple method to run the wires into the internal structure. The sensors should also be ordered in the longer wire variety (300mm) this will prevent needing to extend the wires, these are availiable as product number FS2T.0.1E.300, from innovative sensor technologies and other distributors. Other changes should be focused around the electrical hardware and model interface such as the servo mount which is currently nonexistent should be designed such that the servo can be attached with screws, similarly for the proto board or any type of circuit board that might be used in future experiments. The final suggestion is regarding the software implementation. More of the software should be implemented on the Arduino, this will allow for a more even distribution of the computing load between the computer and Arduino, which should increase calculation times, and decrease timing mismatch between the Arduino and MATLAB.

If planning to use the 3D printer located in Dr. Enikov's lab, note that the tolerance on its printing will make holes smaller than designed in SolidWorks. Smaller deviations can be sanded out while larger deviations from design can be drilled or milled. Holes can also be made larger in the model before putting them in the 3D printing program. However, for pieces that must be more exact (such as the elevon), it is recommended to use a more exact printer such as the one located in the AME building's machine shop. For items such as the vertical mount or the wind tunnel seal where being very exact is less important, it can save costs by printing with PLA using Dr. Enikov's printer.

The threading on the wall inserts (if new ones are made in the future) can be reduced to ¾"-10 in order to reduce the possibility of the threading becoming mismatched with the nuts since aluminum when cut thin can be easy to bend. If the current wall inserts made are still used, clean out the threading on the inserts before each use (i.e. by using pressurized air). It is also helpful to add WD-40 to the threading to make the nut slide on more smoothly.

References

Bruining, A. "Angles of Attack Methods in Relation to Aerodynamic Characteristics on Rotating Wind Turbine Blade." (2010): n. pag. Institute of Wind Energy, Delft University of Technology. Web. 12 Nov. 2016.

He Shen, Yunjun Xu, and Charles Remeikas. "Hardware Design and Validation of Pitching Control for Micro Air Vehicles Using Only Pressure Information." *2013 American Control Conference* (2013): 5568-573. Web. 05 Oct. 2016.

LazerToyz. "Lazer V Micro Glider." *LazerToyz*. N.p., n.d. Web. 13 Nov. 2016.

Longitudinal Equations of Motion. Digital image. *Wikipedia.* Wikimedia, 15 May 2008. Web. 05 Dec. 2016.

Mueller, Thomas J., and Gabriel E. Torres. *Aerodynamics of Low Aspect Ratio Wings at Low Reynolds Numbers With Applications to Micro Air Vehicle Design and Optimization*. Rep. no. UNDAS-FR-2025. Notre Dame: U of Notre Dame, November 2001. Online.

Nelson, Robert C. "Stick Fixed Longitudinal Motion." *Flight Stability and Automatic Control*. New York: McGraw-Hill, 1989. 152-55. Print.

Que, Ruiyi, and Rong Zhu. "Aircraft Aerodynamic Parameter Detection Using Micro Hot-Film Flow Sensor Array and BP Neural Network Identification." Sensors 12.12 (2012): 10920-0929. Web.

Sadraey, Mohammad H. "Tail Design." *Aircraft Design: A Systems Engineering Approach*. Chichester, West Sussex: Wiley, 2013. 274-352. Print.

Appendix

Appendix A – Engineering Drawings

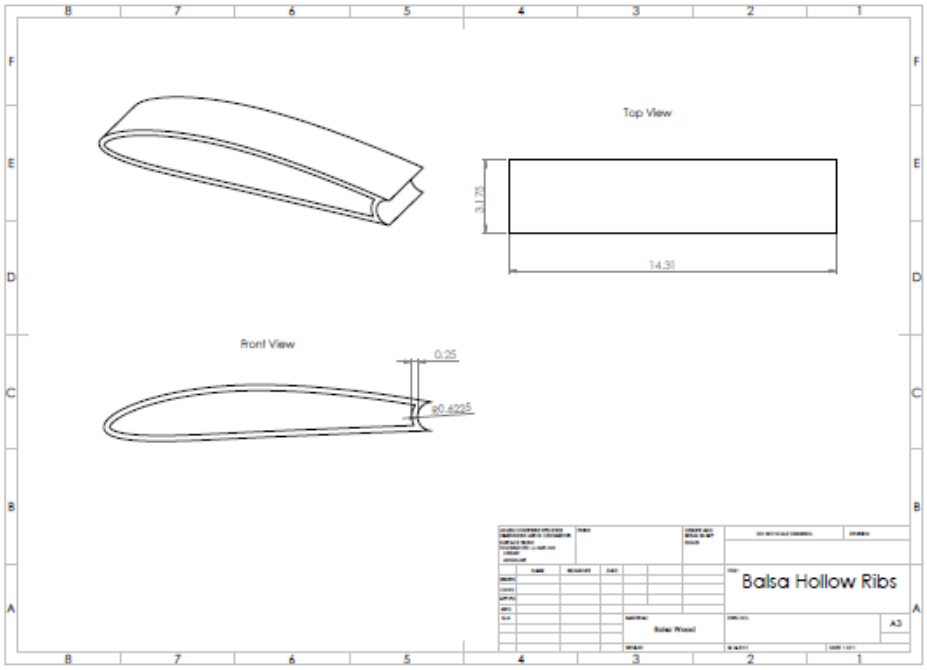


Figure 21: *Balsa Hollow Rib Engineering Drawing*

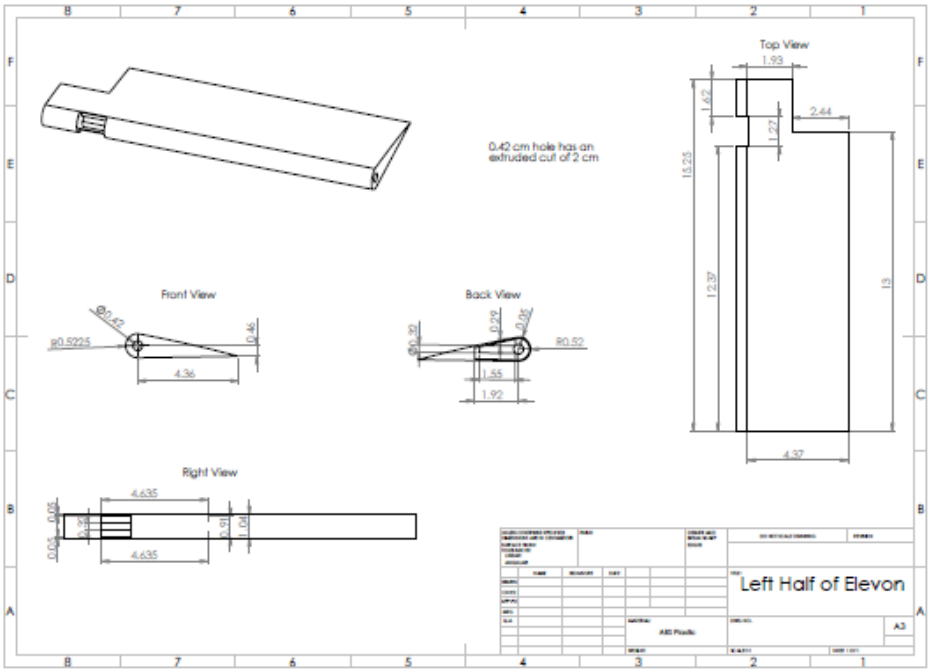


Figure 22: *Left Elevon Engineering Drawing*

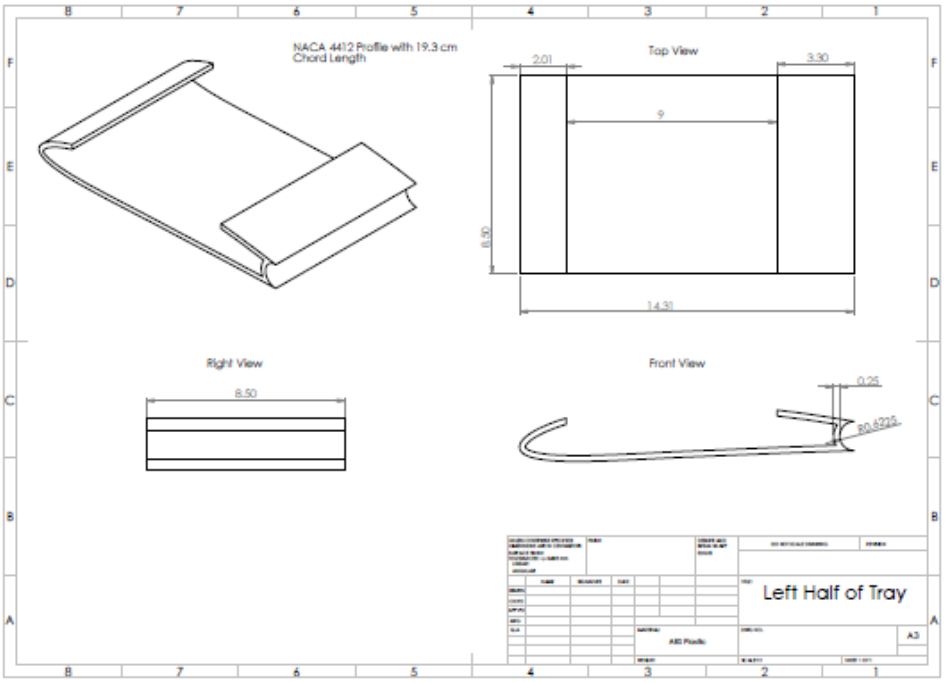


Figure 23*: Left Tray Engineering Drawing*

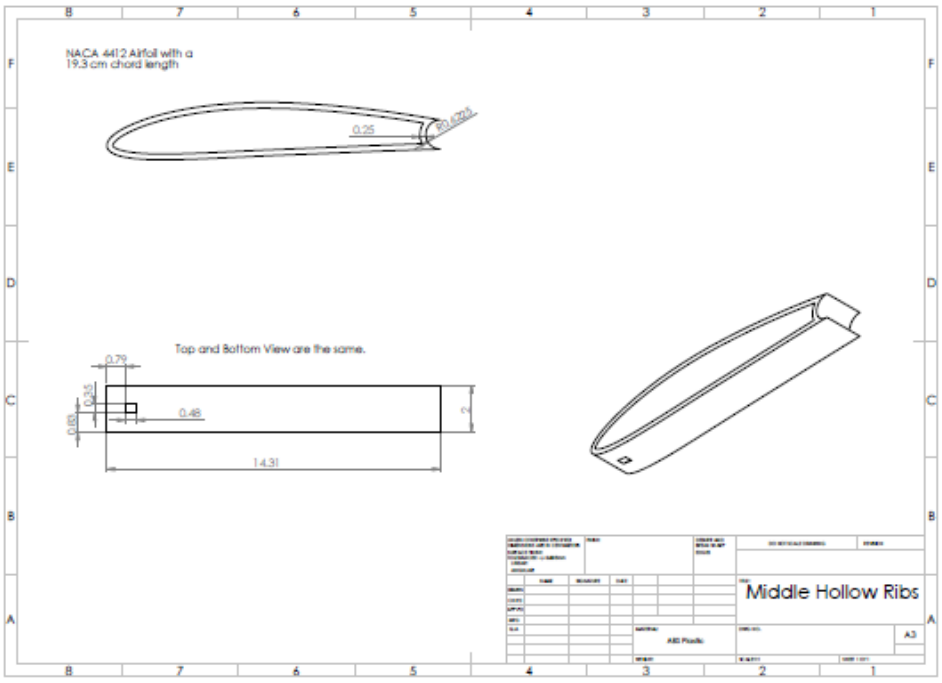


Figure 24*: Middle Hollow Rib Engineering Drawing*

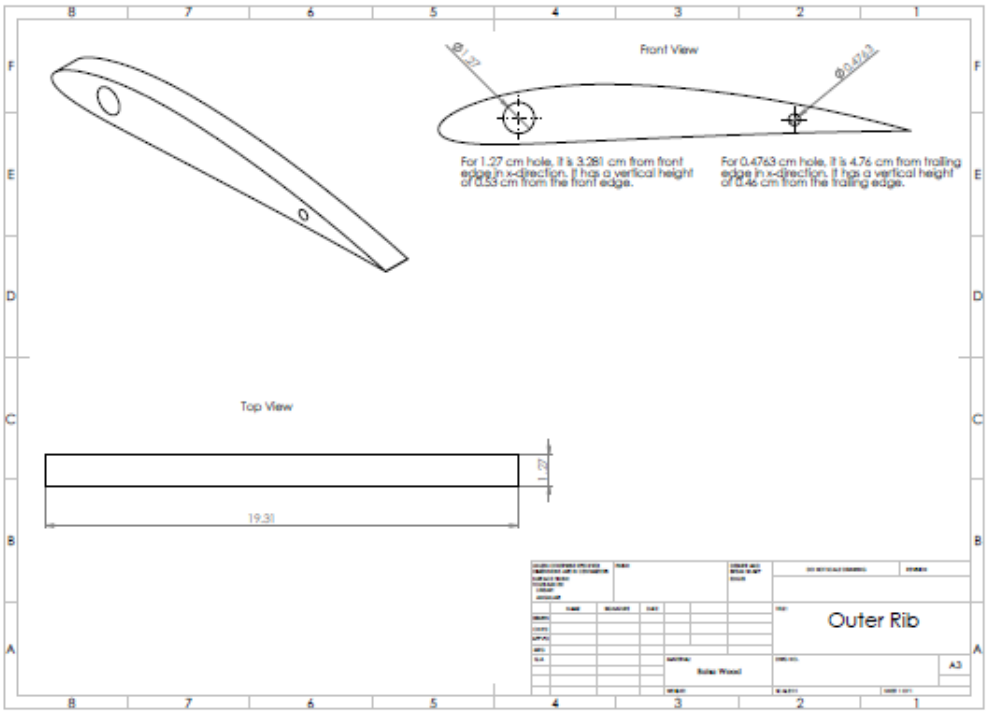


Figure 25: *Outer Rib Engineering Drawing*

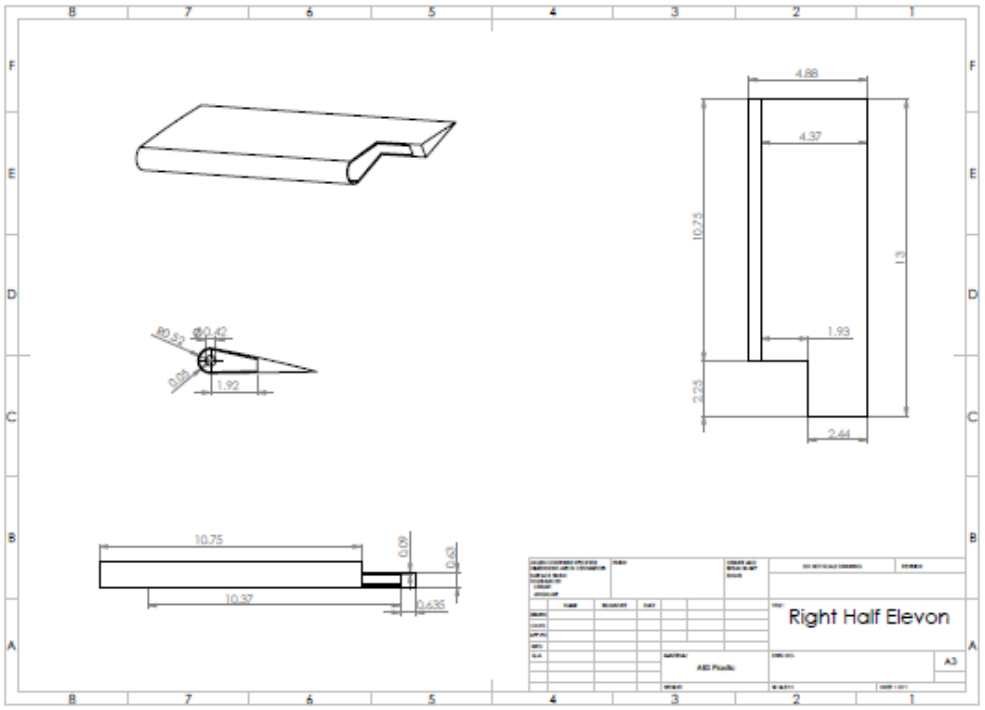


Figure 26: *Right Elevon Engineering Drawing*

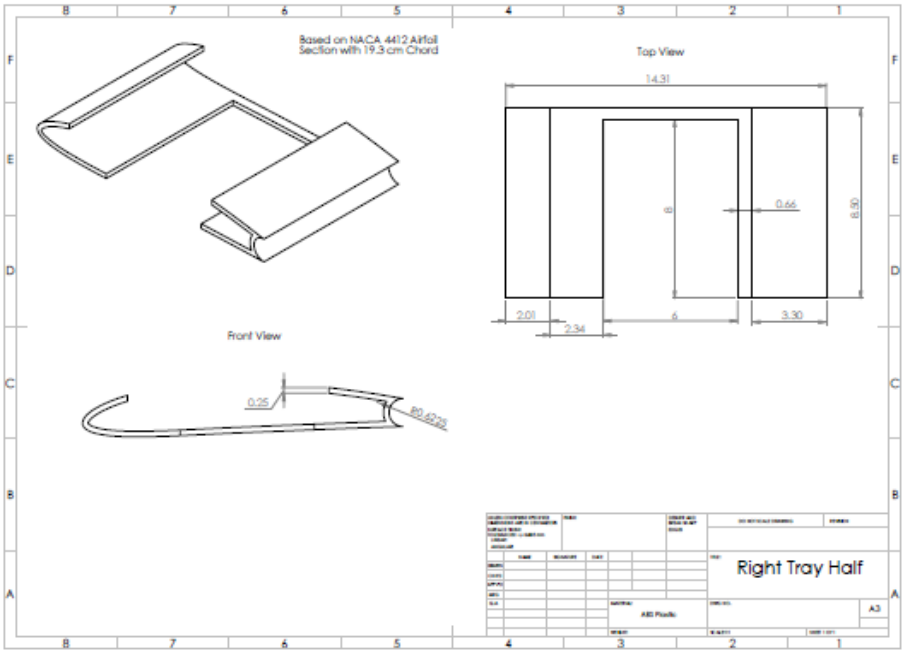


Figure 27*: Right Tray Engineering Drawing*

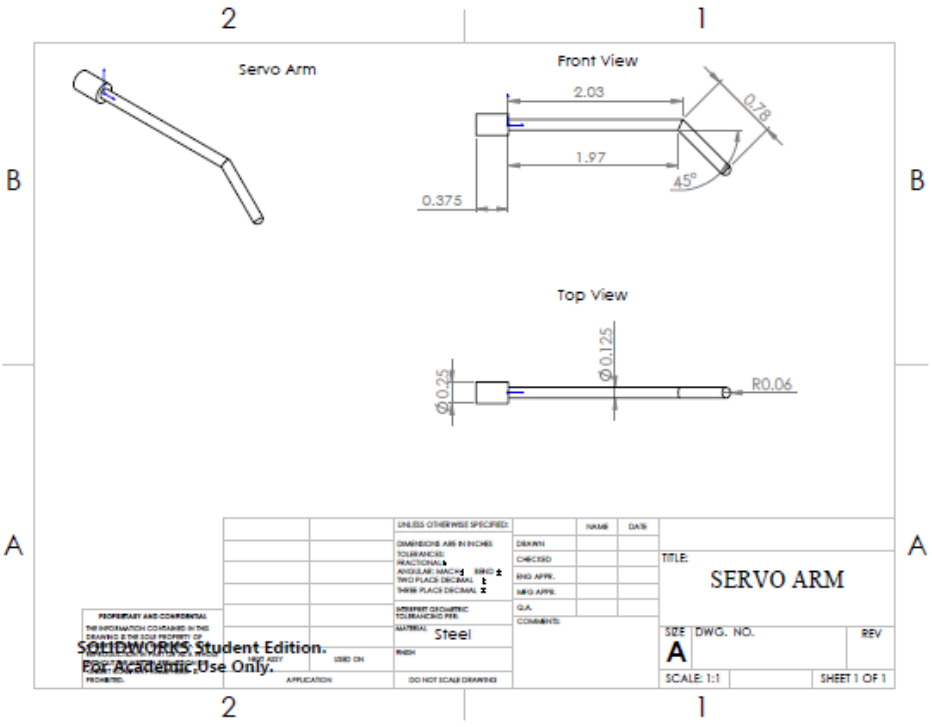


Figure 28*: Servo Arm Engineering Drawing*

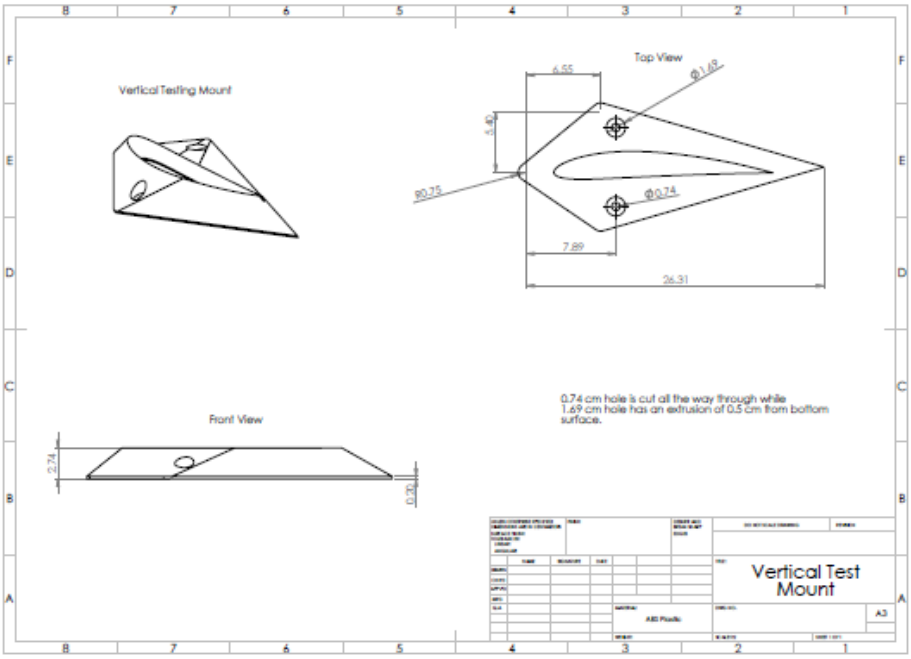


Figure 29*: Vertical Test Mount Engineering Drawing*

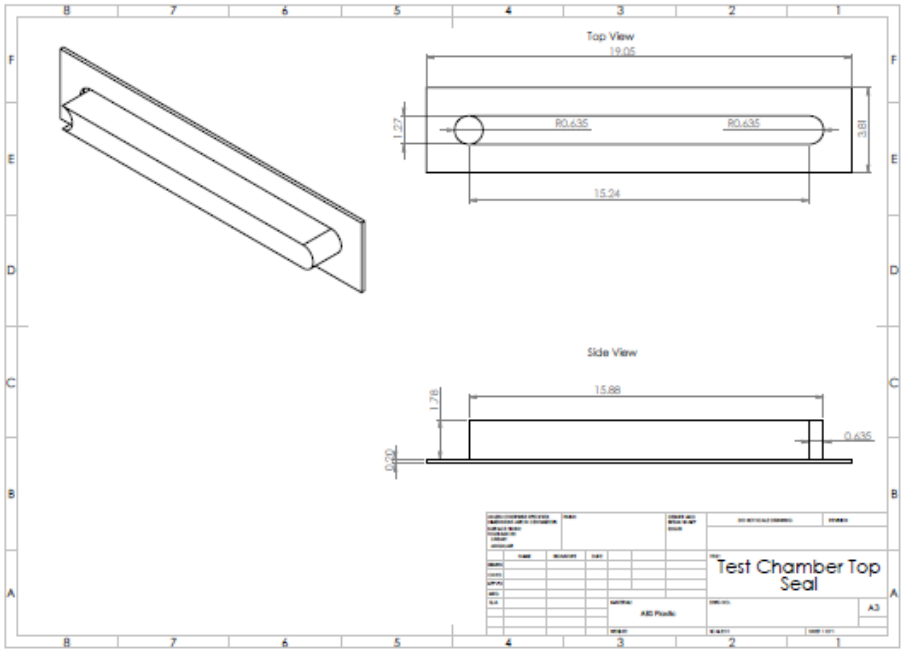


Figure 30*: Test Chamber Top Seal Engineering Drawing*

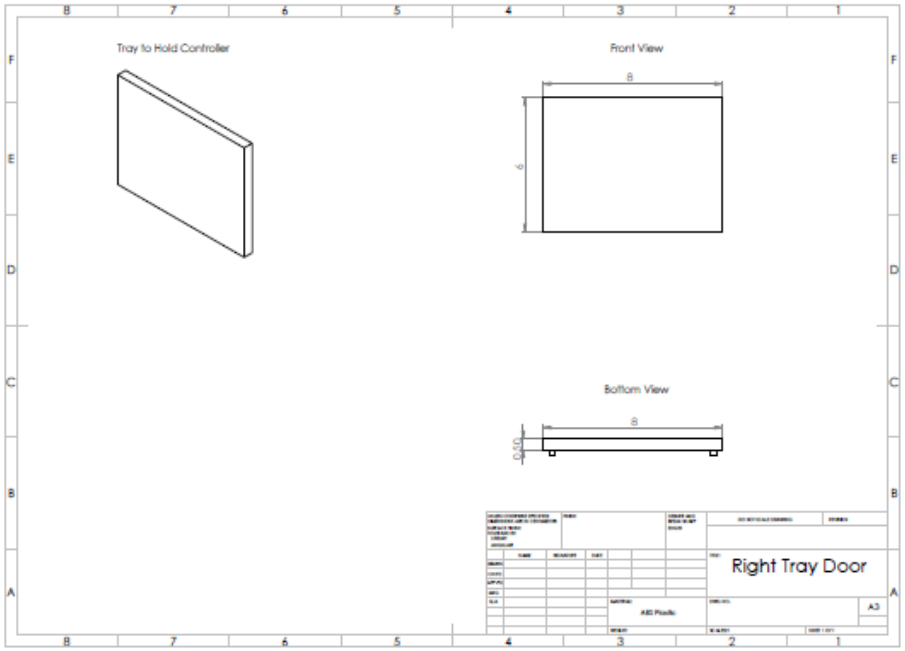


Figure 31*: Right Tray Trapdoor Engineering Drawing*

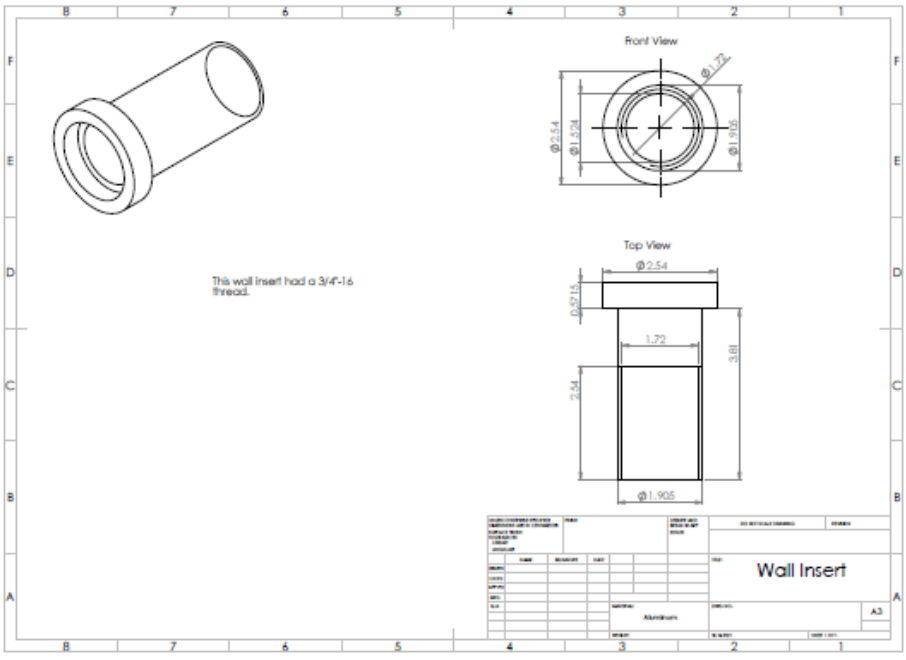


Figure 32*: Wall Insert Engineering Drawing*

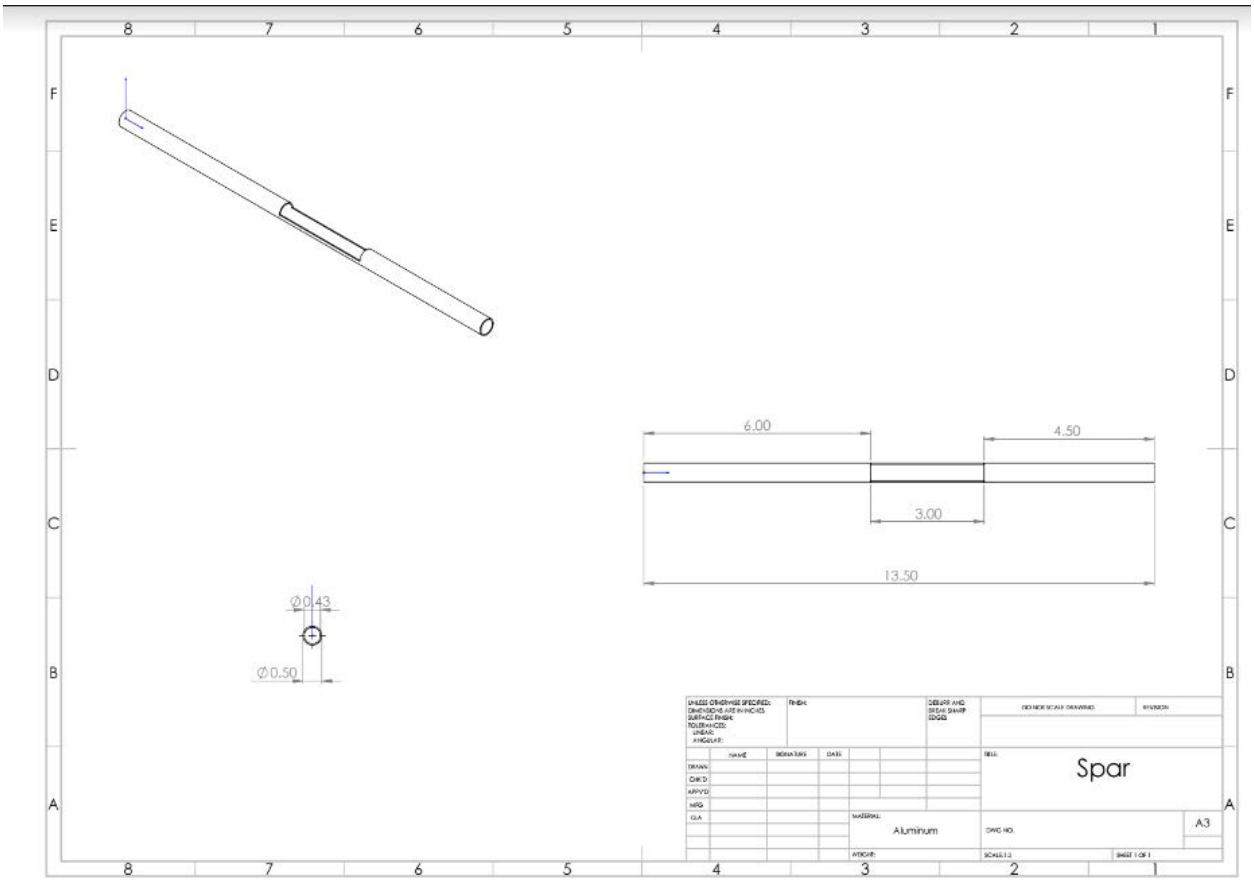


Figure 33*: Spar Engineering Drawing*

Appendix B – Arduino Firmware

```
#include <Servo.h>

const float pi = 3.14159; //Define Pi

//Read/Write Values Definition
int tDesired = 121; //Calibrated zero value of Servo
int pos = 0;
int FS2Top = 0;
int FS2Bot = 0;
int i = 0;
String sigInStr;
String sensorOut;
String comma = ",";
float sigInFlt;

//Pin Definitions
int topPin = A0;
int botPin = A1;
int servoPin = 9;

//Declare Elevator Servo
Servo elevator;

void setup() {

  Serial.begin(9600);        //Begin serial 9600 Baud Rate
```

```
  elevator.attach(servoPin);  //Atach servo to pin servoPin
  elevator.write(tDesired);   //Initalize servo to neutral position
  delay(15);
}

void loop()
{
  //Read Top and Bottom Servo, write to serial port in format Top,Bottom
  FS2Top = analogRead(topPin);
  FS2Bot = analogRead(botPin);
  sensorOut = FS2Top + comma + FS2Bot;
  Serial.println(sensorOut);*/

  //Check if a signal is availiable on the serial port, if yes deflect servo.
  if(Serial.available()>0)
  {
    sigInStr = Serial.readString();
    sigInFlt = sigInStr.toFloat();
    sigInFlt = (180/(2*pi))*asin(tan(sigInFlt*(2*pi/180)));
    elevator.write(tDesired+sigInFlt);
    delay(15); //Modify to Sync MATLAB and Hardware
  }
  delay(20); //Modify to Sync MATLAB and Hardware
}
```

Appendix C – MATLAB Data Processing and Control Software

```
%% ======================== Arduino data logging ==========================

clear all
close all
clc
format long
format compact

% Init Arduino variables
COM_port = 'COM4';

Bit2Volts = 5.0/1023.0;

% Init Arduino
arduino = serial(COM_port,'BaudRate',9600);
fopen(arduino);
% Init Arduino PINS
    % empty

% Loop config
do_loop_plots = true;
time_step = 0.01;                % seconds
number_of_steps = 1000;

% low pass filter
T = 1e-1;
Ts = 0.01;
tf_f = tf(1,[T 1]);
Z = c2d(tf_f,Ts,'zoh');
[num,dem]=tfdata(Z);
num = cell2mat(num);
dem = cell2mat(dem);
a = num(2);
b = dem(2);
c = dem(1);

% calibration diff
diff = importdata('output\calibration_diffrence.txt');

% PID controler gains:
kp = 0;
ki = 0;
kd = 0;

k1= kp + ki + kd;
k2=-kp - 2*kd;
k3= kd;

% wanted
alpha = deg2rad(30);
```

```
% calibration curve coefs
A = 1;
B = 0;

% init
output(1) = 0;




%% Main read, write, and log

FS1_data = nan(number_of_steps,3);
FS2_data = nan(number_of_steps,3);
times = zeros(number_of_steps,1);

figure(1);
subplot(1,3,1);
title('Flow Sensor 1 voltage');
xlabel('recieve time');
ylabel('Read voltage [V]')
grid()
hold on

subplot(1,3,2);
title('Flow Sensor 2 voltage');
xlabel('recieve time');
ylabel('Read voltage [V]')
grid()
hold on

subplot(1,3,3);
title('voltage diffrence');
xlabel('recieve time');
ylabel('Read voltage [V]')
grid()
hold on

n = 1;
while n<=number_of_steps

    % read serial port
    y = fscanf(arduino,'%f,%f');
    Y1(n) = y(1)*Bit2Volts;
    Y2(n) = y(2)*Bit2Volts;

    % filter data
    if n>1
       FS1_data(n,1) =  a*Y1(n-1) - b*FS1_data(n-1,1);
       FS2_data(n,1) =  a*Y2(n-1) - b*FS2_data(n-1,1);
    else
        FS1_data(n,1) = Y1(n);
        FS2_data(n,1) = Y2(n);
    end
```

```
    % voltage diffrence
    delta_v(n) = FS1_data(n,1) - FS2_data(n,1);

    % calibration curve
    cal = A*delta_v(n) + B;
    error(n) = alpha - cal;

    % controler
%     if n>2
%         output(n) = output(n-1) + k1*error(n) + k2*error(n-1) + k3*error(n-
2);
%     else
%         output(n) = 0;
%     end

    % write to output
%     fwrite(arduino,output(n));

%     FS2_v = readVoltage(Arduino,FS2_PIN);
%
%     FS1_dir = readVoltage(Arduino,FS1_dir_PIN);
%     FS2_dir = readVoltage(Arduino,FS2_dir_PIN);
%
%     FS1_h = readVoltage(Arduino,FS1_H_PIN);
%     FS2_h = readVoltage(Arduino,FS2_H_PIN);

%     FS2_data(n,1) = FS2_v;
%     FS1_data(n,2) = FS1_dir;
%     FS2_data(n,2) = FS2_dir;
%     FS1_data(n,3) = FS1_h;
%     FS2_data(n,3) = FS2_h;
%
    if do_loop_plots
        subplot(1,3,1)
        plot(n,FS1_data(n,1),'.r')
        drawnow

        subplot(1,3,2)
        plot(n,FS2_data(n,1),'.r')
        drawnow

        subplot(1,3,3)
        plot(n,delta_v(n)','.r')

%         subplot(2,3,2)
%         plot(n,FS1_dir)
%         drawnow
%
%         subplot(2,3,3)
%         plot(n,FS1_h)
%         drawnow
%
%         subplot(2,3,4)
%         plot(n,FS2_v)
```

```
%          drawnow
%
%          subplot(2,3,5)
%          plot(n,FS2_dir)
%          drawnow
%
%          subplot(2,3,6)
%          plot(n,FS2_h)
%          drawnow
    end
    times(n)=now;
    pause(time_step);
    n = n+1;
end

fclose(arduino);
%% end of run, plot output
dlmwrite('output\FS1_data.txt',FS1_data,'precision',16);
dlmwrite('output\FS2_data.txt',FS2_data,'precision',16);
dlmwrite('output\times.txt',times,'precision',16);

figure(2)
subplot(1,3,1)
plot((times-times(1))*3600*24,FS1_data(:,1),'.r')
grid on
xlabel('times')
ylabel('voltage')
subplot(1,3,2)
plot((times-times(1))*3600*24,FS2_data(:,1),'.r')
grid on
xlabel('times')
ylabel('voltage')
subplot(1,3,3)
plot((times-times(1))*3600*24,delta_v','.r')
grid on
xlabel('times')
ylabel('voltage')

figure(3)
plot(times,C)
grid on

disp(((max(times)-min(times))/length(times))*3600*24);

%% plot fiter and unflitered

figure(2)
plot(Y1,'.r')
hold on
plot(FS1_data(:,1))
grid on
legend('raw data','filtered data')
xlabel('bit value')
ylabel('sample number')
```

```
fprintf('Mean value Sensor 1:')
disp(mean(FS1_data(:,1)))
fprintf('Mean value sensor 2:')
disp(mean(FS2_data(:,1)))

%% save data to txt file

file_name = 'static_output_1.txt';

dlmwrite(file_name,[Y1',Y2',FS1_data(:,1),FS2_data(:,1),delta_v'],'precision'
,16);
```

Appendix D – Laser Cutter Operations Manual

Laser Operation

The laser has 3 different functions.

- Raster imaging using dots to make images. These images must be in gray scale or black.
- Vector marking follows curves and lines to mark the material's surface. It does not cut. Vector marking must be in blue.
- Vector cutting follows curves and lines to cut through a material. Vector cutting must be in red.

For images:

- Use a photo editor to convert the image to gray scale and adjust the contrast accordingly. Save as .jpg.
- Open CorelDRAW on the laser computer and open a new drawing. Import the saved image and adjust the size.
- File->Print. Make sure the printing location is VLS.30.
-

Free-handing drawing or text

- Use CorelDRAW to create simple drawings or text. In order to vector mark, vector cut, or raster, make entities blue, red, or black accordingly.
- For vector cutting, entities must be "Hairline" or they will be interpreted as raster images. Select entities and use the drop down on the right side of the screen to select "Hairline." Be sure to click apply.
- File->Print. Make sure the printing location is VLS.30.

Solidworks

- Open the part to cut in Solidworks. File->Make Drawing From Part.
- Create a custom landscape drawing that has no text boxes of any kind. Only the part entities should appear on the drawing. Delete any axes, annotations, text boxes, etc.
- Make sure that the part scale is 1:1
- File->Save As->PDF
- Use a flash drive to copy the PDF to the laser computer's desktop.
- Open CorelDRAW and open a new drawing. Import the PDF from the desktop. Use a window to select entire image. Arrange->Ungroup All.
- Select and delete any individual entities you do not wish to be in the drawing or image.
- Use a window to select entire image. In the "Outline" tab on the right side of the screen, change the outline to "Hairline" and set the color accordingly. Black for raster. Blue for vector marking. Red for vector cutting.
- File->Print. Make sure the printing location is VLS.30.
-

Please delete your files from the desktop when you are finished using the laser.

Universal Laser Systems Control Panel and Laser Operation

- Once a drawing or image has been printed, the software should open. If it does not, click the red and white diamond on the bottom toolbar.
- Use the drop down menu on the right side of the screen to select "Relocate View." In this view, you can drag the drawing on the screen to the correct location on your material. The rulers on the screen relate to the physical rulers on the machine bed.
- Click "Settings"
    - Select you material from the categories.
    - Type the material thickness into the bar under the material selection.
        - This is the thickness of the actual material and should not include the thickness of any safety plates or mounting material.
    - Slide the "Vector Performance" selector to the left to select "Quality."
    - Click "Apply" and "OK"
- Set the height
    - Use the directional buttons on the screen to move the laser carriage over your material and lift up the metal shield.
    - Slide the black and white height tool under the carriage.
    - Use the up and down keys on the machine to move the bed so that the bottom of the red plate aligns with the top notch of the height tool.
- Use the protective safety plate under your material if you are vector cutting. The plate's thickness should not be included in the "material thickness" but must be accounted for when setting the height.
- When all of the above steps are completed, press the green play button to start the laser process.
- Once the process is complete, wait a few moments before opening the door to allow all fumes to be exhausted.

Do not cut carbon fiber. Attempting to cut carbon fiber produces an excess amount of toxic fumes and large flames. Cutting carbon fiber risks damaging the machine.

Do not hesitate to ask for assistance from one of the shop workers.

Appendix E – Bill of Materials

Material	Vendor	Purchased Online or in Person	Item Number	Unit Cost (might not include Tax or shipping)
Balsawood Midwest ⅛ x4x36 in	Ace Hardware	In Person	5420047	$4.59
Monokote	Hobby Town	In Person	TOPQ0242	$17.99
0.5” OD x 0.43” ID x 3ft long aluminum tube	McMaster-Carr	Online	9056K92	$16.80
Thin Aluminum Rod 3/16 in diameter	McMaster-Carr	Online	8974K21	$3.23
Gizmo Dorks 2.85 mm PLA Filament 1kg (white color)	Amazon	Online	N/A	$26.95
Aluminum Shim Stock Roll, 0.002 in thick, 6 in x 100 in	McMaster-Carr	Online	9708K12	$30.21
Shim Stock, Roll, Cold 302 SS, 0.0020 In	Grainger	In Person	3L603	$26.90
0.5” x 0.75” x 0.1562” Bearing	Boca Bearings	Online	R1212-ZZC	$17.80
Servo - Hitec HS-85MG (Micro Size)	SparkFun	Online	ROB - 11887	$29.95
FS2 Thermal Mass Flow Sensor	Innovative Sensor Technology	Online		$118.97
¼ in Stainless Steel D-Shafting Length 12 in	ServoCity	Online	634094	$4.69
C1 Servo Spline to ¼ in Shaft Coupler (Set Screw)	ServoCity	Online	525134	$4.99
Metal Film Resistors - Through Hole ¼ Watt 6.8K Ohm 1%	Mouser Electronics	Online	660-MF1/4DCT52R6801F	$0.12

Metal Film Resistors - Through Hole 620ohm 1% 100PPM	Mouser Electronics	Online	660-MF1/4DC6200F	$0.08
Metal Film Resistors - Through Hole 1/2 WATT 120 OHM 1%	Mouser Electronics	Online	660-MF1/2DCT52R1200F	$0.16
Metal Film Resistors - Through Hole 1/4W 39 ohm 1%	Mouser Electronics	Online	660-MF1/4DCT52R39R0F	$0.12
Metal Film Resistors - Through Hole 1/4W 1.1M ohm 1% 1.1M OHM 1%	Mouser Electronics	Online	660-MF1/4DCT52R1104F	$0.12
Trimmer Resistors - Through Hole 3/8" 1Kohms Sealed Vertical Adjust	Mouser Electronics	Online	652-3296Y-1-102LF	$2.41
Trimmer Resistors - Through Hole 3/8IN 200 OHM Sealed Vertical Adjust	Mouser Electronics	Online	652-3296Y-1-201LF	$2.41
Metal Film Resistors - Through Hole 3watts 420ohms 1%	Mouser Electronics	Online	71-CPF3-F-420-E3	$1.19
Operational Amplifiers - Op Amps Low Pwr Programmable	Mouser Electronics	Online	595-TLC271CP	$0.75
¾”-16 Yellow Zinc Finish Grade 8 Finished Hex Nut	Fastenal	Online	31161727	$1.40
¼”-20 Phillips Pan-Head Machine Screw	Home Depot	In Person	Model: 3316 Internet: 204794673	$0.87
¼-20 Stainless-Steel Hex Nut	Home Depot	In Person	Model: 2530 Internet: 204794625	$0.29
3/8” D x 1ft long Steel Round Bar	Industrial Metal Supply	Online	N/A	$1.00
1” D x 1 ft long Aluminum Rod	McMaster-Carr	Online	1615T72	$17.80

Appendix F – Elevon Sizing MATLAB Code

```
%Elevon Sizing

%This code takes the dimensions of the wing along with desired angles of

%deflection as inputs and outputs what angles of attack the elevon design

%can control the wing.

clc;clear all


c = .144; %Chord of wing (m)

x_ac_w=c/4; %Aerodynamic center of the wing from leading edge (m)

x_cg = 0.03281; %Center of gravity of the wing from leading edge (m)

e_c = 0.049; %Chord of elevon (m)

tau = 1; %Control surface efficiency factor

de = 0.4363; %Maximum deflection of elevon (radians)

S = 0.05; %Planform area of wing (m^2)

Se = 0.01232; %Planform area of wing (m^2)

AR = 2.0139; %Aspect Ratio of wing

CM0 = -0.062;

CLa = 1.95264; %Lift coefficient of wing per radian of angle of attack

CLde = 2.9794; %Lift coefficient of elevon per radian of deflection


x_ac_e = c-0.75*e_c; %Aerodynamic center of elevon from wing's leading edge (m)

Vh = -Se/S*(x_cg/c-x_ac_e/c); %Tail volume

CMa = CLa*(x_cg/c-x_ac_w/c)-Vh*CLde*(1-2*CLa/AR); %Moment coefficient per radian of
angle of attack

CMde = -Vh*CLde; %Moment coefficient per radian of elevon deflection


a_low = -(CMde*de+CM0)/CMa %Lowest AOA elevon can control (rad)


de = -de;


a_high = -(CMde*de+CM0)/CMa %Highest AOA elevon can control (rad)
```

Appendix G – Computational Fluid Dynamics Documentation

Overview:

The goal of the analysis described below was to extract the velocity data at varying chord locations near the leading edge of the wing. The following will describe the process starting from the geometry building to the post processing.

ANSYS Geometry Builder:

Modeling the c-grid for the airfoil can be done using two approaches, both being fairly similar. The technique chosen ultimately depends on the chosen airfoil which will affect the mesh technique used. The initial modeling is exactly the same, the differences arise when splitting the geometry into separate quadrants.

1) Analysis type and opening design modeler
    a. Select "Geometry" (block 2) in the Fluid Flow schematic.
    b. In the properties menu, change "Analysis Type" to 2D.
        i. THIS MUST BE DONE BEFORE CREATING GEOMETRY
    c. Right click on "Geometry" block and click "Edit Geometry in Design Modeler"
2) Import airfoil geometry
    a. Airfoiltools.com has a good database of airfoils.
        i. Note: Trailing edges are NOT closed
    b. For 4 and 5 - digit NACA airfoils, they also have good airfoil generators
        i. http://airfoiltools.com/airfoil/naca4digit
        ii. There is an option to close TE when generating data points
    c. Import data into Design Modeler or SOLIDWORKS
        i. Design Modeler data format

            #group #point #x_cord #y_cord #z_cord

            1. '#' are ignored by ANSYS
        ii. SOLIDWORKS data format

X	Y	Z
0	0	0
..	..	..

            1. Make sure to add column for Z coordinate
    d. Create surface from sketch
    e. IF USING SOLIDWORKS: Import external geometry to Design Modeler
3) C-Grid Modeling
    a. Want the 'C' portion of C-Grid to contain airfoil
        i. Either offset airfoil so TE is at origin of working plane or,
        ii. Create new working plane that is offset to TE of airfoil

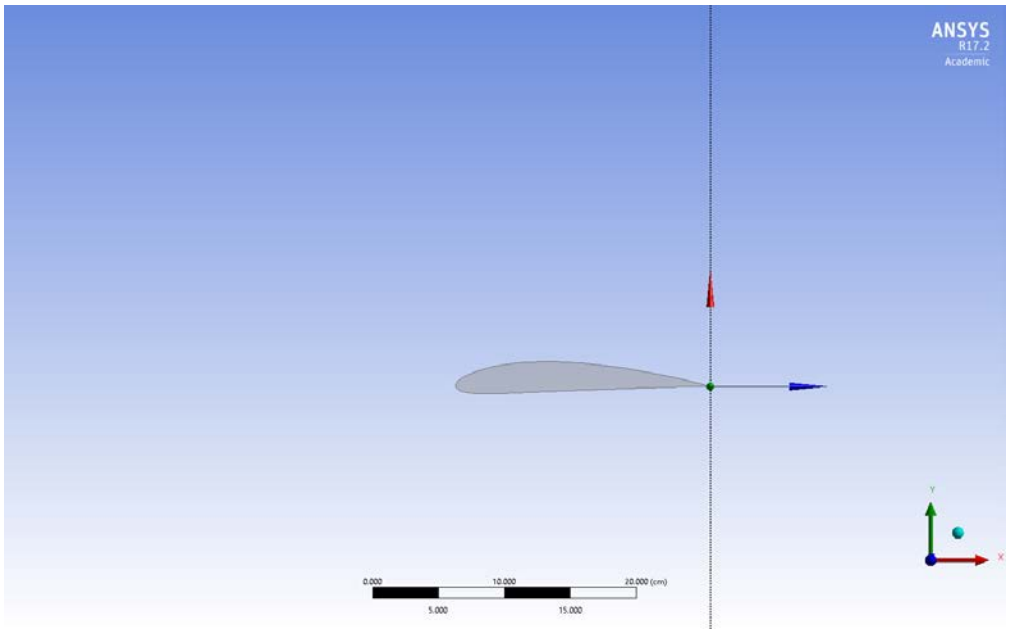


Figure 34*: Airfoil imported to Design Modeler*

b. Select the plane on which the airfoil is located and open the sketching tools

DESIGN MODELER NOTE: To change units, select desired units from "Units" tool bar

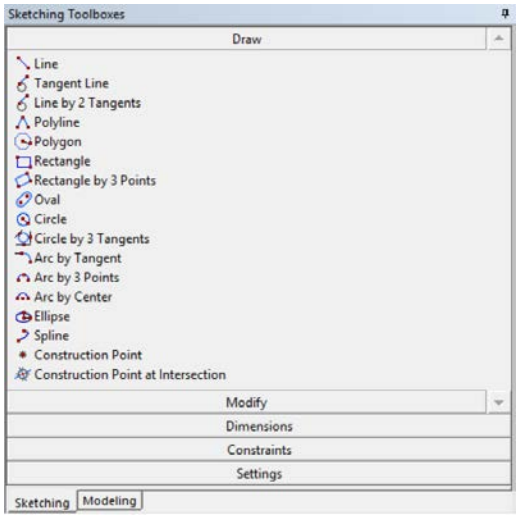


Figure 35*: Sketching Tools*

c. Using "Arc by Center," create an arc centered at the origin. To place two points: first click on upper half of vertical axis, then lower half of vertical axis (The order of point placement is important for orientation when meshing).

Note: When sketching, when a 'P' appears for cursor, this means intersecting with a point, such as the origin. When 'C' appears, this means intersecting with line.

d. Using "Rectangle by 3 Points," create a rectangle with a width of the diameter of the arc (you can set a constraint for "Equal Length").
    i. Again, make the first point in the upper half of plane, second point in the lower half, and third point anywhere in right half of plane.
e. Under "Modify," select "Trim." Click the length of rectangle that runs through origin.

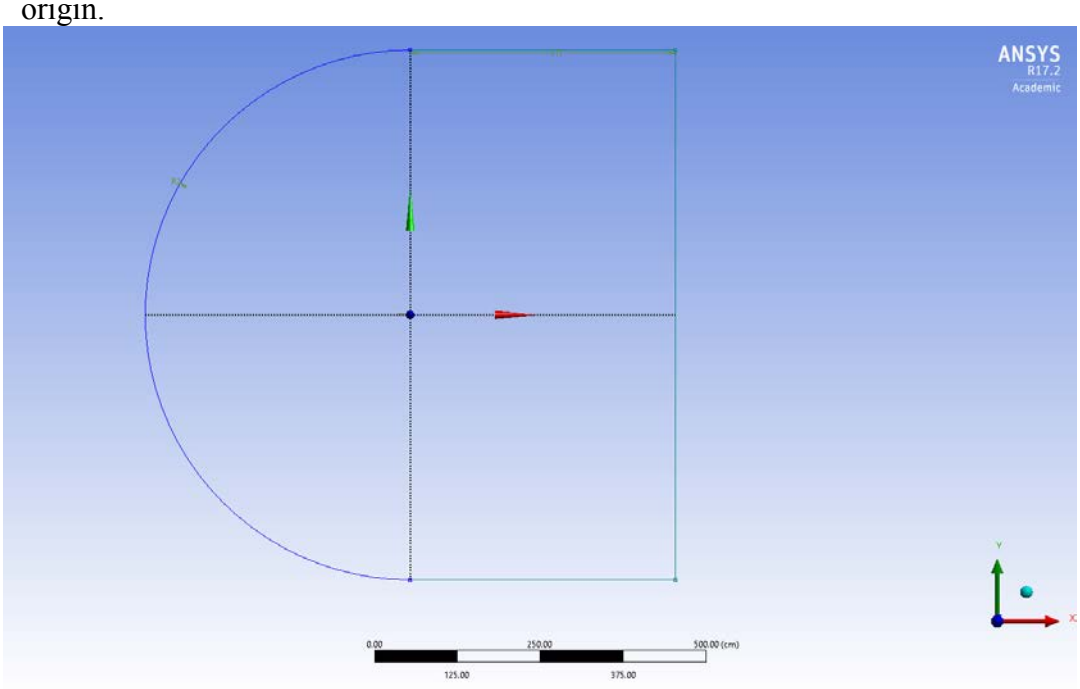


Figure 36*: Final C-Grid Sketch*

f. Define dimensions – move to "Dimensions" tab of sketch tools
    i. Arc radius should be sized to about 10-20 times the chord length
    ii. Rectangle length should be sized about 10-20 times the chord length
g. Create surface from C-Grid sketch.
    i. Creating surface because performing 2D analysis. No need to perform volume mesh.
    ii. In tool bar select "Concept" -> "Surface from Sketch"
    iii. Select C-Grid sketch as geometry
    iv. Operation: "Add Frozen"
    v. Click "Generate"
h. Remove airfoil from flow domain

The surface that was just generated represents the flow domain to be meshed. The next step is to remove the geometry.

    i. In the tool bar select "Create," and select "Boolean"
    ii. Operation: "Subtract"
    iii. Select geometry from which to be removed
    iv. Select tooling geometry (geometry to remove)
    v. Click "Generate"

NOTE: See bottom left corner for layer selection. This will especially be useful when dealing with 3D geometries.

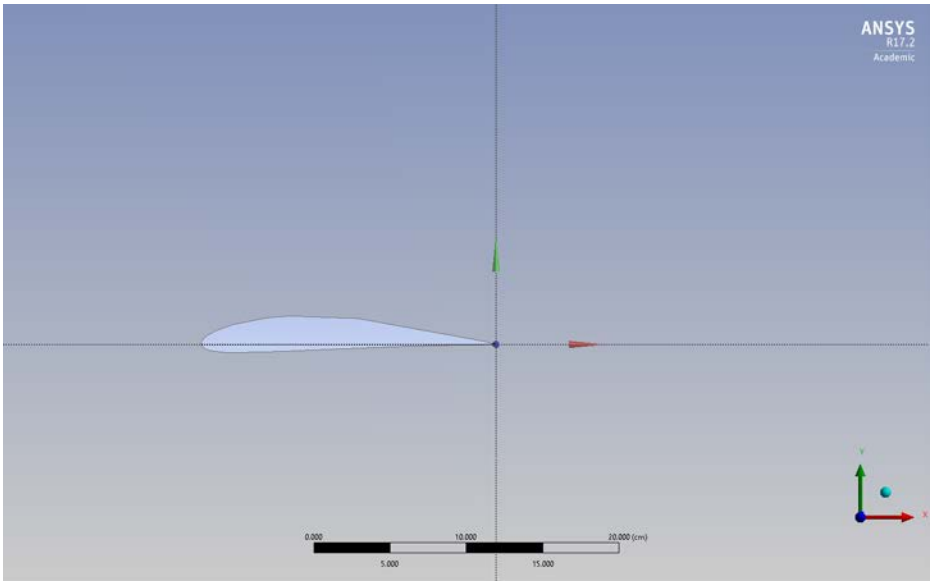


Figure 37*: Boolean Operation on Flow Domain*

DESIGN MODELER NOTE: Must click generate to see changes take place when working on anything outside of sketch tools. Cannot move on to other processes unless geometry is generated.

4) APPROACH 1 (Used for this project) – Geometry for unstructured mesh

It was found that this technique worked best for airfoils with more complex geometries, such as camber.

a. In same plane as C-Grid sketch, select new sketch icon: 
b. Sketch a line from TE of airfoil to edge of flow domain.

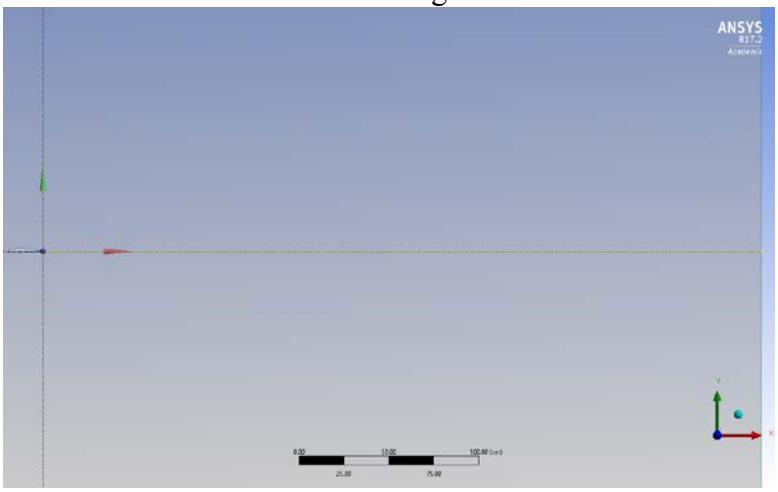


Figure 38*: Wake Separation Line Sketch*

c. "Concept" -> "Line from Sketches," and select sketch of above line
d. Hit "Generate"

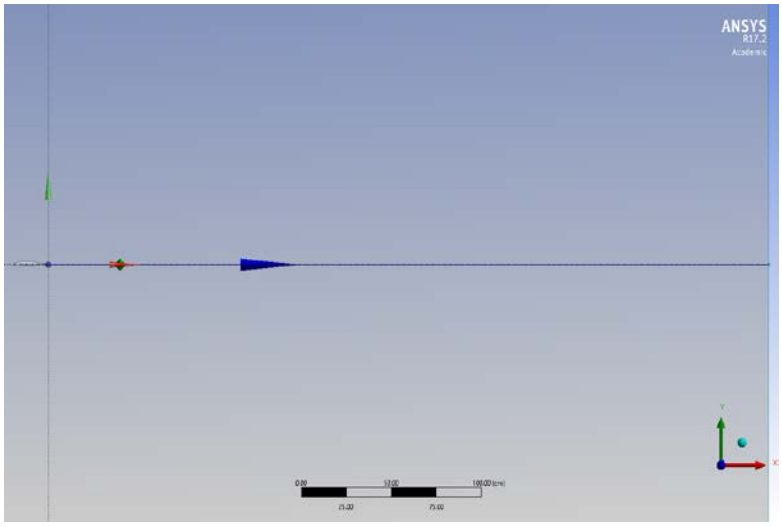


Figure 39: *Wake Separation Line*

e. Project line on to flow domain by selecting "Tools" -> "Projection"
    i. Select recently created line for "Edges" and C-Grid domain for "Target." Hit generate.
    ii. Under "Bodies" in the tree outline, right click on line body and select "Hide"

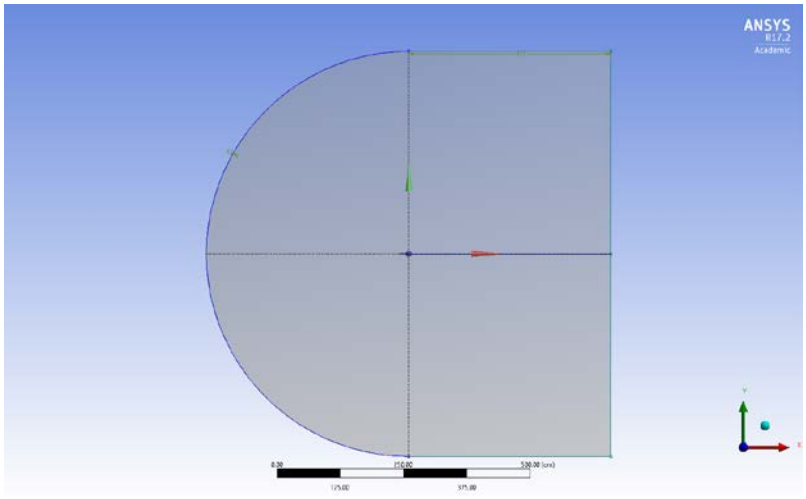


Figure 40: *Final Geometry for Unstructured Mesh*

5) APPROACH 2 - Geometry for structured mesh

To create a structured mesh, the c-grid will be split into 4 segments: The upper and lower half of the 'C' portion, and the upper and lower half of the rectangular portion. This approach is best for symmetric airfoils.

a. Click on the working plane for the C-Grid and click  to create a new sketch on this plane.
b. Sketch a line that runs along the vertical axis of the plane through the entire geometry.
    i. Make sure first point of line is in upper half plane. This will create the correct line orientation.
c. Go to "Concepts" -> "Lines from Sketches" and select the sketch that was just created. Make sure the orientation is in the NEGATIVE vertical direction. Hit Generate
d. Add another new sketch and repeat the process but for a line through the horizontal axis through the geometry (Start line in left half plane for correct orientation).
    i. It may be easier to start at origin and draw line to right half plane, then, under "Modify" in sketch tools, choose "Extend" and select point at origin.
e. Create a line from the new sketch, make sure the orientation is in the POSITIVE horizontal direction.

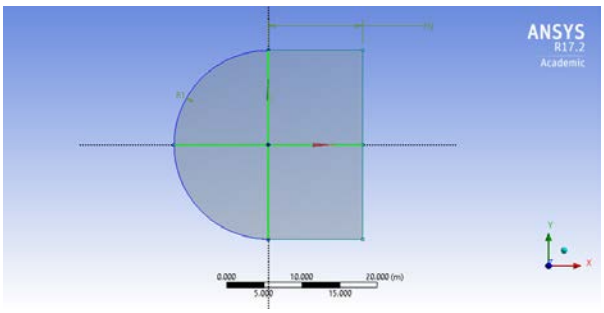


Figure 41*: Lines on C-Grid Domain*

f. Create projections from lines.
    i. "Tools" -> "Projection"
    ii. Edges: Vertical lines that was just created
    iii. Target: C-Grid domain
    iv. Click generate. The domain is now split into half (left and right).
    v. Repeat above steps two more times for horizontal lines in the left and right half sections.
g. The domain will now be split into four regions
h. Under "Bodies" in the tree outline, right click on line body and select "Hide"

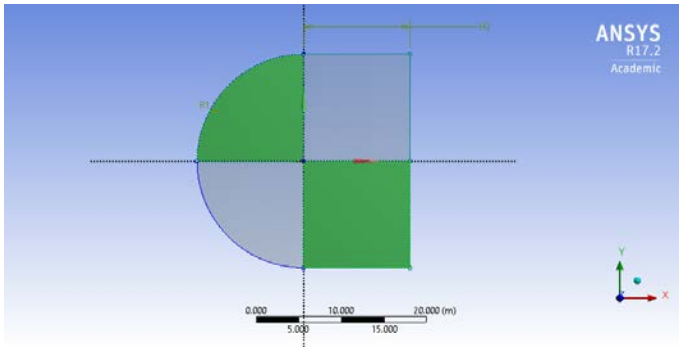


Figure 42*: Final Geometry with Four Quadrants*

6) Named Selections
    a. Select the 'C' portion of geometry and the upper and lower rectangular sections, and right click "named selection"
        i. Name it "Inlet" -> Generate
    b. Select both sections of rear, named "Outlet"
    c. Select the airfoil shape, named "Airfoil"

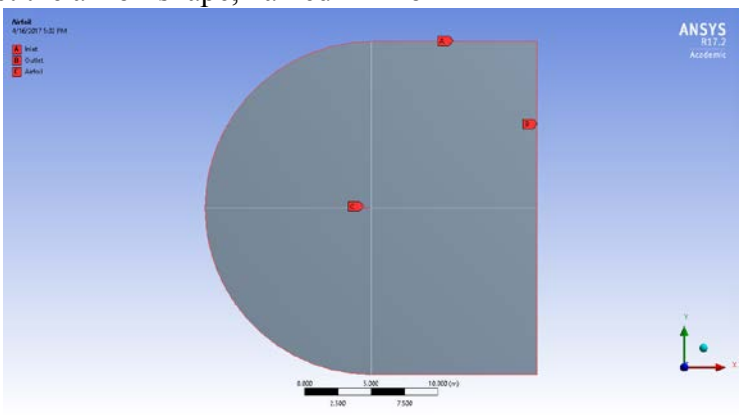


Figure 43*: Named Selections*

ANSYS MESHER

1) Types of mesh techniques

Structured Mesh: A structured mesh refers to the fact that the mesh elements can be identified by their 'ijk' coordinates. Typically, a structured mesh will be a hex mesh (3D) or square mesh (2D). These are numerically advantageous if employed correctly. It can be difficult to generate a high quality structured mesh for complex geometries.

Unstructured Mesh: An unstructured mesh is as its name states, unstructured. It requires a connectivity list to reference the grid elements. Unstructured meshes are advantageous for more complex geometries, however can cause numerical errors due to irregular element shapes.

Mesh Size Controls

i) Point Sizing

Size points using a sphere of influence. Size of sphere can be chosen in sizing options.

ii) Edge Sizing

Size a selected edge of the geometry. Sizing options include number of divisions, size of divisions, or a sphere of influence. Bias can be assigned to create higher mesh density in certain regions, such as the leading edge and trailing edge of an airfoil.

iii) Face Sizing

Size selected faces of the geometry. Sizing options allow for growth rate of mesh.

iv) Body Sizing

Size entire selected body. Allows for finer meshes in areas that need a higher resolution.

v) Body/Sphere of Influence

Use a sphere or a body to control the area of refinement.

Mesh Controls

i) Auto Mesh

ANSYS will automatically perform a tetrahedron mesh.

ii) Face Mesh

Allows for user controlled mesh. Ability to create a mapped mesh.

iii) Inflation

Used in areas where a higher resolution is needed such as along the surface of the airfoil to capture the boundary layer. Can be defined by a smooth transition, total thickness, or by a first layer thickness. Need to define the maximum layers and inflation growth rate. Causes mesh elements to "inflate" from small size to face/body mesh elements size.

2) APPROACH 1 – Unstructured Mesh (Used for this project)
   a. Sizing
      i. Click generate mesh to see the initial auto mesh.

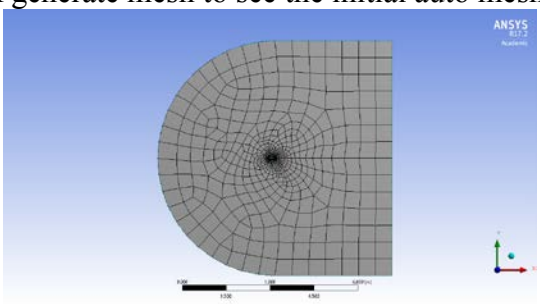


Figure 44: *Auto Mesh Without Sizing*

      ii. Click "Mesh" in the selectin tree and expand "Sizing." Change "Relevance Center" to "fine."

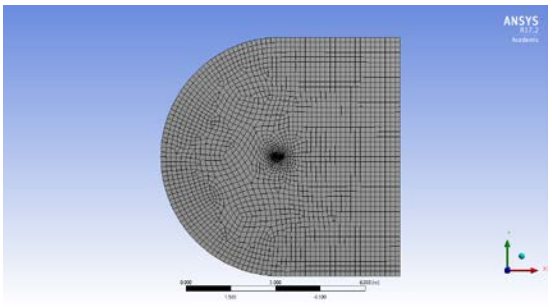
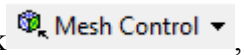


Figure 45: *Auto Mesh With 'Fine' Sizing*

b. Refinement
    i. In order to properly capture the flow around the airfoil the mesh around the airfoil must be refined.
    ii. Click 

, select "Sizing."
    iii. Select the airfoil edges for "Geometry"
        1. This will create an edge sizing
    iv. Enter the following parameters for the sizing:
        1. Type: Number of Divisions
            a. 500
        2. Behavior: Hard
        3. Bias Type: No Bias
    v. Click "Update" to see refinements
    vi. Add another sizing, but this time select the flow domain for the geometry to create a body sizing.
    vii. For "Type" select "Sphere of Influence." Enter the following parameters:
        1. Sphere radius: 50 cm
        2. Element size: 2 cm
            a. Play with sphere size and element sizing. This will change how much of the mesh is finely sized, and also how much computational power is required to mesh.

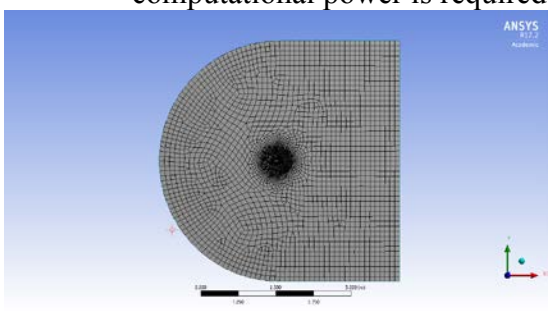


Figure 46: *Mesh with Sphere of Influence and Edge Sizing*

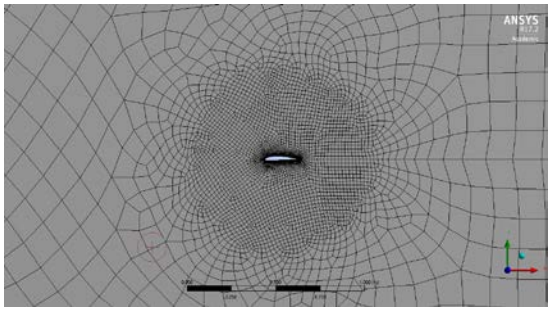


Figure 47*: Zoom View of Sphere of Influence*

viii. Under "Mesh Control" select "Inflation."
ix. Select the flow domain for the Geometry and select the airfoil shape as Boundary
x. Enter the following parameters:
    1. Inflation Option: Smooth Transition
    2. Transition Ratio: Default
    3. Maximum Layers: 10
    4. Growth Rate: 1.2
        a. The above parameters can be changed to refine the mesh in various ways

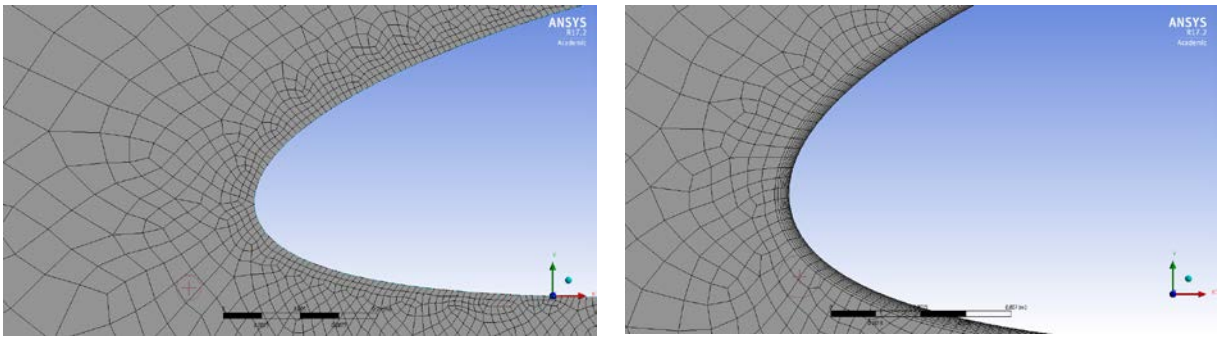


Figure 48*: Comparison of Mesh Without and with Inflation Layer*

c. Trailing Edge Issue

When meshing an inflation layer with an unstructured mesh on an asymmetric airfoil, problems can arise at the trailing edge. These problems are due to the high aspect ratio of the inflated elements and can cause irregularities in the solution. To solve this, it is suggested that the mesh density at the TE be adjusted properly.

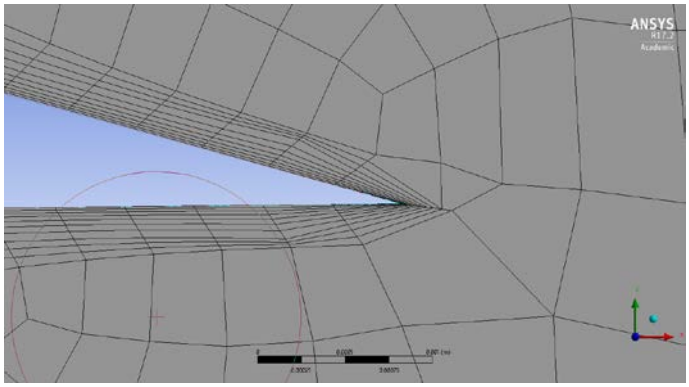


Figure 49*: Trailing Edge Mesh Issue*

3) APPROACH 2 – Structured Mesh
    a. Sizing
        i. Add a sizing to the mesh.
        ii. Select the rear upper, middle upper, forward, and lower rear edges as the geometry to create an edge sizing.
        iii. Enter the following parameters:
            1. Type: Number of Divisions, 50
            2. Size Function: Uniform
            3. Behavior: Hard
            4. Bias Type: ----- --- - -
            5. Bias Factor: 150
                a. The bias for each sized edge must trend towards the center as shown in the figure below. The bias factor can be changed to obtain a higher mesh density near the leading and trailing edge.

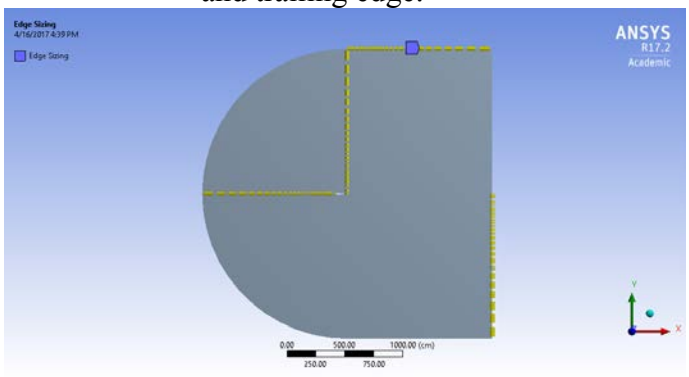


Figure 50*: First Edge Sizing for Structured Mesh*

        iv. Follow a similar process for the remaining edges (not including the 'C' portion)
            1. The "Bias Type" will be - - --- ----- to ensure the sizing trends towards the middle.

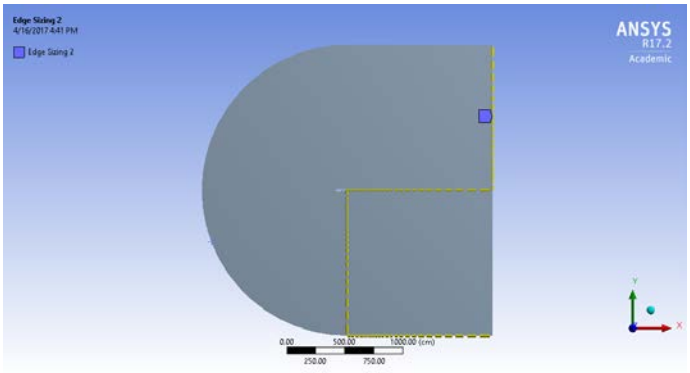


Figure 51: *Second Edge Sizing for Structured Mesh*

v. Add another sizing and select the 'C' edge for the geometry.
    1. Number of Divisions: 100
    2. Behavior: Hard
    3. Bias Type: No Bias

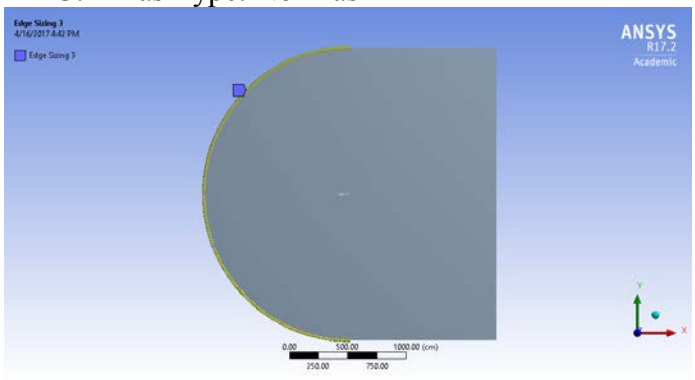


Figure 52*: C Edge Sizing for Structured Mesh*

b. Mesh
    i. Under "Mesh Control" add a face sizing and select the four faces of the flow domain as the geometry. This will allow for a mapped mesh to be created based off of the above sizing.
    ii. Enter the parameters:
        1. Mapped Mesh: Yes
        2. Method: Quadrilaterals
    iii. Click "Generate Mesh"

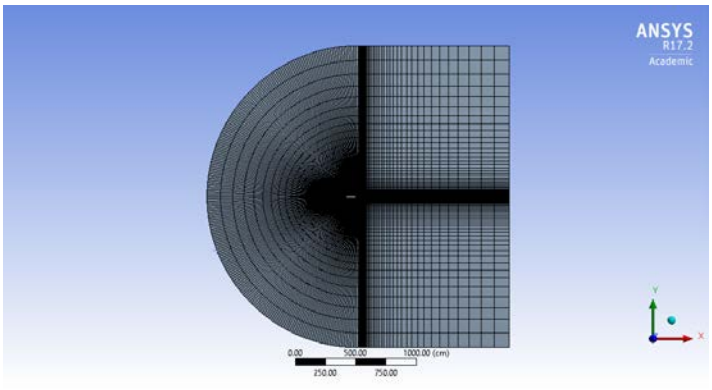


Figure 53*: Structured Mesh of a C-Grid*

4) Mesh statistics

The mesh statistics will provide an idea of whether or not the analysis will produce good results. It is best to have orthogonal elements of low aspect ratios.

a. Number of element

While a high number of elements can result in a good quality mesh, it will result in a long calculation process. It is best to start with a lower element count and increase the density as needed.

b. Orthogonal Quality

The orthogonal quality ranges from 0 to 1, 1 being the best. Try to keep the orthogonal quality greater than 0.1.

c. Aspect Ratio

High aspect ratio cells should generally be avoided as they cause inaccuracies and instabilities in the solution. However, in applications such as boundary layers, they can be acceptable as the gradients of the pressure and velocity are not in the transverse direction.

d. Skewness

The skewness ranges from 0 to 1, 0 being the best. Try to have a skewness less than 0.95

FLUENT Solver

1) Launching solver
   a. When launching fluent the a few options must be selected
      i. Options: Double Precision
      ii. Processing Options: Parallel if machine is capable, if not use serial
      iii. Processes: Dependent on the machine

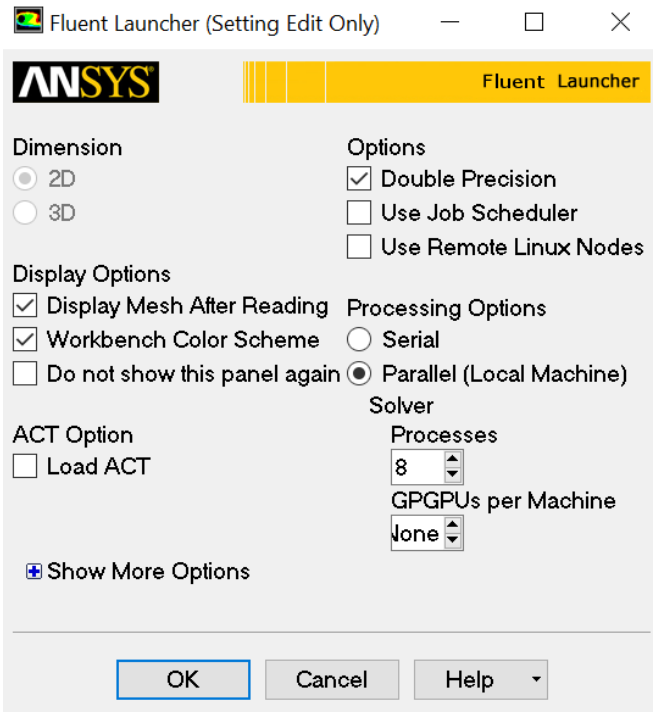


Figure 54: *FLUENT Launcher*

2) Models
    a. Material
        i. There is the option to model the air with constant density, an ideal gas, or an incompressible ideal gas. Use the incompressible ideal gas.

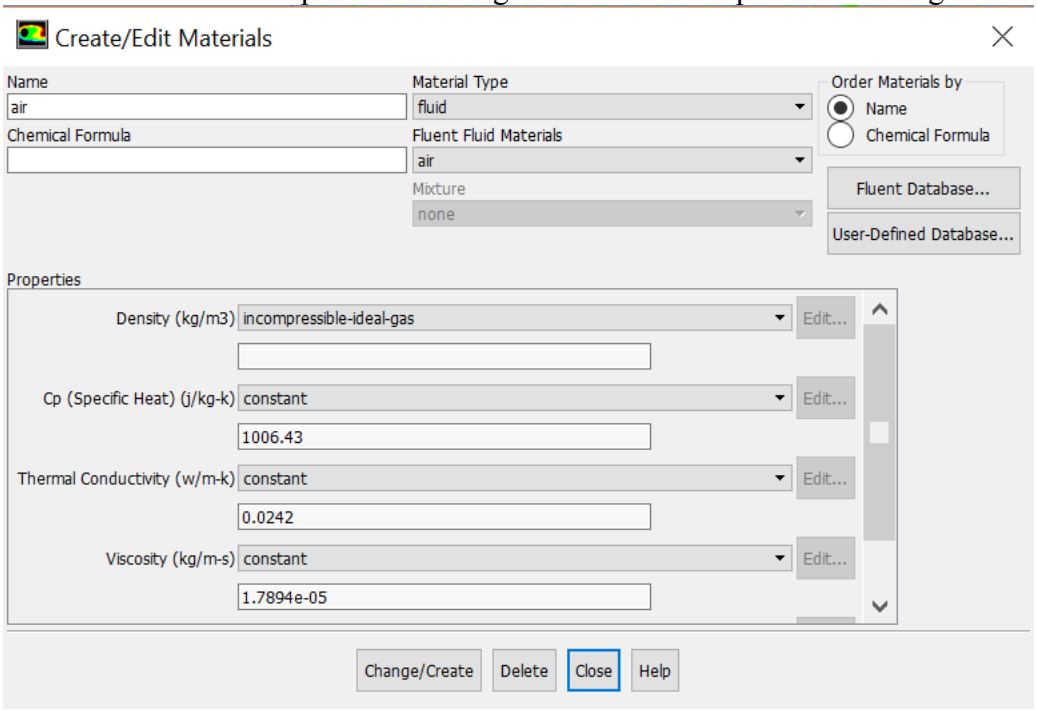


Figure 55: *Material Setup*

b. Turbulence Models
   i. Turbulence model is necessary to model the flow in the wind tunnel. The viscous model allows to add a turbulence model. Common models include Spalart-Allmaras and k-epsilon.
      1. The Spalart-Allmaras turbulence model is a one-equation model and can accurately capture the flow on the wall.

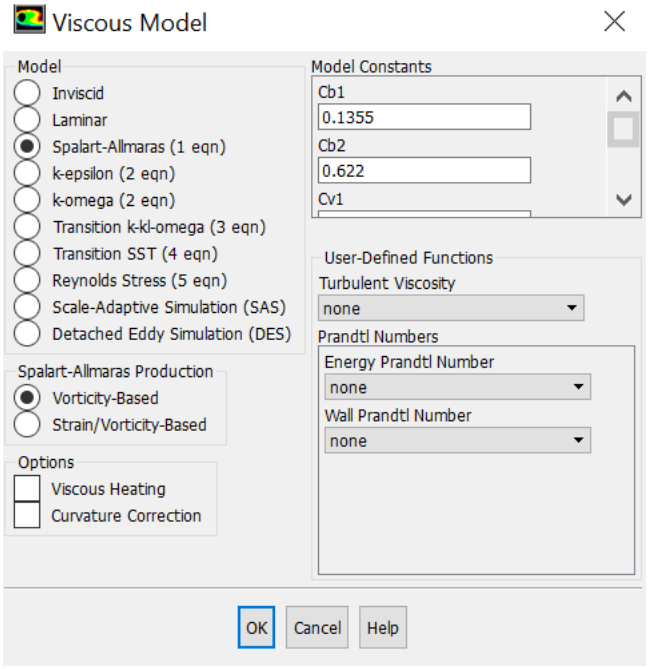


Figure 56: *Viscous Models*

3) Types Boundary Conditions
   a. Pressure Far Field

Models the free stream conditions at infinity. It was determined that the pressure far field boundary conditions has troubles converging at the low Mach numbers that the analysis is to be ran at.

   b. Velocity Inlets

Allows to set the component and magnitude of velocity at a specified location.

   c. Pressure Outlets

Set a desired pressure at a specified outlet location. This is a gauge pressure so make sure to change your operating conditions as necessary.

   d. Walls

Allows for specifications of wall conditions. A no slip will be used for all cases.

4) Setting Boundary Conditions
   a. Set the boundary conditions for the named selections as follows:
      i. Operating Conditions:
         1. Pressure: 101325

ii. airfoil: no slip wall
iii. far_low: Velocity inlet
    1. Specification Method: Magnitude and Direction
    2. Magnitude: Create a parameter to easily control free stream from workbench
    3. Components: Components of angle of attack
iv. far_up: Velocity inlet
    1. Same options as other velocity inlet
v. Inlet: Velocity inlet
    1. Same options as other velocity inlet
vi. Outlet: Pressure outlet
    1. Gauge Pressure: 0 Pa

5) Solution Methods
    a. Use the "SIMPLE" solver.
    b. Use a least squares cell based gradient
    c. For accuracy use second order
        i. First order will allow for easier convergence.
6) Running solutions
    a. Monitors
        i. The "Residuals" will be a default model
            1. Edit to specify convergence criteria. Convergence criteria for this model was 1e-6.
    b. Initialization
        i. Perform a standard initialization
        ii. Compute from inlet and initialize the solution
    c. Run Calculation
        i. Set number of iterations, this analysis use 1500 which was enough for all cases to converge
        ii. Check case. ANSYS will most likely give a high aspect ratio warming due to the boundary layer.
7) Convergence issues and possible solutions
    a. Major convergence issues were from using the Pressure Far Field Boundary Condition
        i. This was solved by switching to the velocity inlet and pressure outlet.
    b. Other convergence issues can occur due to the trailing edge issue described in the meshing section.
        i. Solutions would be to increase mesh density at the trailing edge.

ANSYS Postprocessor

1) Results visualization
    a. Contours
        i. Using the results toolbar,  , select the contour button.
        ii. Select the cases to contour and the domain.
            1. Selecting all cases (angles of attack) will allow for a single contour plot to apply to all results.
        iii. Select the variable to contour.

   b. Streamlines
      i. In the results tool bar, click add streamline
      ii. Similarly, select the cases and domain.
      iii. Select the max number of streams. It helps to define a line in front of the airfoil and make sure there are at least 100 points that describe it.
2) Defining points
   a. CFD Post allows for a point cloud to be defined using a file of points.
   b. Add a file of the airfoil coordinates that are about 0.02mm above the surface.
      i. While the points should be normal to the surface, at such a small distance from the surface, it can be assumed normal.
3) Exports
   a. File -> Export
   b. Choose the variable to export (velocity), the domain (all), and the cases (all).
   c. Export to a file

DATA Interface with MATLAB

The included MATLAB script was written to read in the data points from a file exported by ANSYS. The script then manipulates the data, taking the difference and sum at specified chord locations. The data is plotted and linear fits are determined and the coefficients are printed in the command window.

Outside Resources

CFD-Online: Public forum with docs, references, tutorials, and discussions. Extremely useful source.

https://www.cfd-online.com/

Cornell FLUENT Tutorials: Videos and step by step tutorials of the theory of CFD and simulations, ranging from flow through a pipe to flow through a CD nozzle.

https://confluence.cornell.edu/display/SIMULATION/FLUENT+Learning+Modules

Data Analysis MATLAB Script

```
%Read velocity data from local velocity over airfoil output from

%ANSYS. Plot velocity profile of airfoil, calculate difference of
leading

%edge velocities and plot AoA vs. LE Dif., calculate sum of leading
edge velocities

%and plot freestream vs. LE Sum.


%John Mangels - 25 February 2017


%% READ IN DATA

%ANSYS Output file information

readings = 200;

col = 3;

row = 6;

%Calculation information

num_tests = 8;

num_velocities = 9;

chord = 19.3e-2;

aoa = [0,2,4,6,8,10,12,14];

index = [12,16,18,20,22,24]; %3, 5, 7, 8.5,10,12 percent chord

num_positions = 6;

file_names =
{'airfoil_vel_05.csv','airfoil_vel_08.csv','airfoil_vel_09.csv','airfo
il_vel_10.csv','airfoil_vel_11.csv','airfoil_vel_12.csv','airfoil_vel_
13.csv','airfoil_vel_14.csv','airfoil_vel_15.csv'};

velocities = [5,8,9,10,11,12,13,14,15];

total_sums = zeros(num_positions,num_velocities*num_tests);

j = 1;
```

```
for e = 1:num_velocities

    file = file_names{e};

    %coordinates of velocity readings

    x = csvread(file,6,0,[6,0,6+readings,0]); %TE => LE

    y = csvread(file,6,1,[6,1,6+readings,1]);

    x_perc = abs(x(1:100)/chord)*100; %x coordinates converted to
percent chord (LE => TE)


    %velocity data

    upper_vel = zeros(100,num_tests); %100 readings at each for upper

    lower_vel = zeros(100,num_tests); % 100 readings at each for lower

    %Read data from ANSYS output file

    %******************NOTE: CSV FILES INDEXED FROM
0********************

    for i = 0:num_tests-1

        r1U = row*(i+1)+readings*i;

        r2U = r1U+readings/2-1;

        r1L = r2U + 1;

        r2L = r2U + readings/2;

        upper_vel(:,i+1) =
csvread(file,r1U,col,[r1U,col,r2U,col]); %TE => LE

        lower_vel(:,i+1) =
csvread(file,r1L,col,[r1L,col,r2L,col]); %LE => TE

    end

    upper_vel = flip(upper_vel,1); %LE => TE


    %% PLOT VELOCITY DATA VERSUS CHORD POSTITION
```

```
    figure;

    set(gcf,'name',strcat('Upper and Lower Surface Vel - Freestream:
',num2str(velocities(e)),'m/s'),'numbertitle','off');

    for i = 1:num_tests

        subplot(3,3,i);

        plot(x_perc,upper_vel(:,i),'r',x_perc,lower_vel(:,i),'b');

        title(strcat('Upper and Lower Surface Vel vs. Position - AoA:
',num2str(aoa(i))));

        xlabel('Chord Position (% Chord)');

        ylabel('Velocity [m/s]');

        legend('Upper','Lower');

    end


    %% SUM AND DIFFERENCE CALCULATIONS

    dif = zeros(100,num_tests);

    sum = zeros(100,num_tests);

    for i = 1:num_tests

       dif(:,i) = upper_vel(:,i) - lower_vel(:,i);

       sum(:,i) = upper_vel(:,i) + lower_vel(:,i);

    end


    %% PLOT AoA VS. LE VELOCITY DIFFERENCE FOR DIFFERENT FREE STREAMS

    figure;

    set(gcf,'name',strcat('LE DeltaV - Freestream:
',num2str(velocities(e)),'m/s'),'numbertitle','off');


for i = 1:num_positions

        subplot(2,3,i);
```

```
        p = polyfit(dif(index(i),:),aoa,1); %linear interpolation
between velocity difference and aoa

        y1 = @(x) p(1).*x + p(2);


plot(dif(index(i),:),aoa,'ro',dif(index(i),:),y1(dif(index(i),:)));

        title(strcat('AoA vs. Sensor Velocity Difference at  ',
num2str(x_perc(index(i)),3),'% chord'));

        xlabel('Sensor Velocity Difference [m/s]');

        ylabel('Angle of Attack [deg]');

        if (i == 1) || (i == 2)

            fprintf('Free stream: %i Chord
perc.: %d\n',velocities(e),x_perc(index(i)));

            fprintf('Equation ax+b: %d*x %d\n', p(1), p(2));

        end

end


    %Add summation data to matrix of LE sums for each location

    total_sums(:,j:num_tests*e) = sum(index,:);

    j = num_tests*e+1;

end


%% PLOT FREESTREAM VS. LE VELOCITY SUM FOR DIFFERENCT AoA

for i = 1:2

    t = zeros(num_tests,num_velocities);

    figure;

    set(gcf,'name',strcat('Sensor reading to freestream correlation at
',num2str(x_perc(index(i)),3),'% chord'),'numbertitle','off');

    for j = 1:num_tests

      subplot(3,3,j);
```

```
        t(j,:) = total_sums(i,j:num_tests:j+num_tests*(num_velocities-
1));

        p = polyfit(t(j,:),velocities,1); %linear interpolation between
velocity difference and aoa

        y1 = @(x) p(1).*x + p(2);

        plot(t(j,:),velocities,'ko',t(j,:),y1(t(j,:)));

        fprintf('Position %d aoa: %i\n', x_perc(index(i)),aoa(j));

        fprintf('ax+b: %d*x %d\n', p(1), p(2));

        title(strcat('Freestream vs. LE Velocity Sum - AoA:
',num2str(aoa(j))));

        xlabel('LE Velocity Sum [m/s]');

        ylabel('Free Stream');

    end


end
```