**Robotic Spacecraft Simulator: Thrust Measurement System**

Nolan Cain

Srajit Mehrotra

Katrina Mullane

Anthony Newton

Bachelor of Mechanical Engineering

Dalhousie University

Supervised by Dr. Mae Seto, Dr. Robert Bauer

Date: April 6th, 2022

# Abstract

The purpose of the Robotic Spacecraft Simulator (RSS) thruster testing apparatus is to measure the output thrust from thruster nozzles, and the pressure of both the system storage pressure and thruster supply pressure. To measure thrust, a bending load cell transducer was selected along with a pressure transducer to read and record the inlet supply pressure. This report will expand on design iterations throughout project development, the final apparatus, results obtained and recommendations to improve the RSS and testing apparatus. Three experiments were performed with the physical apparatus to characterize the thrust force from the current nozzles implemented on the RSS. First the effects of outlet diameter on thrust force were examined by repeated the experiment using outlet diameters ranging from 0.2mm to 1.0mm. The 0.6mm outlet diameter nozzle was found to produce the maximum thrust force. The second experiment performed analyzed the effects of actuating one nozzle versus four. A slight decrease in output thrust was observed when four nozzles were actuated compared to one due to a decrease in supply pressure per nozzle. Lastly, the effects of varying the inlet supply pressure on thrust force were examined. The thrust force was observed to increase linearly as the supply pressure was increased, reaching a maximum at the highest inlet supply pressure. Based on the results, it is recommended for the ISL to implement the 0.6mm outlet diameter nozzles with a maximum inlet supply pressure if they want to maximize the thrust output. However, if they wish to maximize the system runtime of the RSS, the 0.2mm outlet diameter nozzles with a minimum inlet supply pressure should be used. Other recommendations to improve the physical apparatus and CFD model include running the CFD model on a fully licensed version of ANSYS to improve mesh sizes, purchase an amplifier with a higher sampling rate to better capture transient thrust forces, purchase a more precise second stage regulator to improve repeatability of experiment, obtain a smaller scale load cell to increase resolution, minimize the volume between the solenoid valves and nozzles to decrease the transient period, implement M6 flat O-rings between the thruster nozzle and coupling instead of using Teflon tape, and lastly it is recommended to explore converging-diverging nozzles for implementation on the RSS.

# Contents

[Abstract i](#_Toc100168875)

[Contents ii](#_Toc100168876)

[List of Figures iv](#_Toc100168877)

[List of Tables v](#_Toc100168878)

[List of Equations v](#_Toc100168879)

[Acknowledgements vii](#_Toc100168880)

[1 Introduction 1](#_Toc100168881)

[2 Requirements and Goals 3](#_Toc100168882)

[2.1 Project Requirements 3](#_Toc100168883)

[2.2 Project Goals 3](#_Toc100168884)

[3 Stakeholders 5](#_Toc100168885)

[4 Functional Decomposition 6](#_Toc100168886)

[5 Design Concepts 8](#_Toc100168887)

[5.1 Pressure Determination 8](#_Toc100168888)

[5.2 Torsional Pendulum Thrust Stand 9](#_Toc100168889)

[5.2.1 Advantages 10](#_Toc100168890)

[5.2.2 Disadvantages 10](#_Toc100168891)

[5.3 Piezoelectric Sensor 10](#_Toc100168892)

[5.3.1 Advantages 11](#_Toc100168893)

[5.3.2 Disadvantages 12](#_Toc100168894)

[5.4 Bending Load Cell 12](#_Toc100168895)

[5.4.1 Advantages 13](#_Toc100168896)

[5.4.2 Disadvantages 13](#_Toc100168897)

[5.5 Compressive Load Cell 13](#_Toc100168898)

[5.5.1 Advantages 15](#_Toc100168899)

[5.5.2 Disadvantages 15](#_Toc100168900)

[6 Concept Evaluation 16](#_Toc100168901)

[6.1 Concept Screening 16](#_Toc100168902)

[6.2 Concept Scoring 17](#_Toc100168903)

[7 Design Iterations 19](#_Toc100168904)

[7.1 CFD Model 19](#_Toc100168905)

[7.1.1 SolidWorks Model 19](#_Toc100168906)

[7.1.2 ANSYS Model 21](#_Toc100168907)

[7.2 CAD Model 22](#_Toc100168908)

[7.2.1 Iteration One 23](#_Toc100168909)

[7.2.2 Iteration Two 24](#_Toc100168910)

[7.2.3 Iteration Three 26](#_Toc100168911)

[8 Electrical Design 32](#_Toc100168912)

[8.1.1 Transducer Selection 32](#_Toc100168913)

[8.1.2 First Software Iteration 35](#_Toc100168914)

[8.1.3 Second Software Iteration 37](#_Toc100168915)

[8.1.4 Load Cell Calibration and Unit Conversion 38](#_Toc100168916)

[9 Final Apparatus 39](#_Toc100168917)

[10 Results 41](#_Toc100168918)

[10.1 Thrust Force versus Outlet Diameter 41](#_Toc100168919)

[10.2 Actuation of 1 Nozzle versus 4 42](#_Toc100168920)

[10.3 Thrust Force versus Supply Pressure 43](#_Toc100168921)

[10.4 Results Summary 44](#_Toc100168922)

[11 Project Budget 46](#_Toc100168923)

[11.1 Cost Benefit Analysis 48](#_Toc100168924)

[12 Conclusion 49](#_Toc100168925)

[13 Recommendations 53](#_Toc100168926)

[14 Project Handover 56](#_Toc100168927)

[References 57](#_Toc100168928)

[Appendix A Project Requirements 60](#_Toc100168929)

[Appendix B Code 62](#_Toc100168930)

[Appendix C Engineering Drawings 92](#_Toc100168931)

[Appendix D User Operation Manual 107](#_Toc100168932)

[Appendix E Failure Mode and Effects Analysis 115](#_Toc100168933)

# List of Figures

Figure 1‑1: Artistic rendering of a space debris deorbiting vehicle [5] 2

Figure 3‑1: Stakeholder Map 5

Figure 4‑1: Functional Decomposition 7

Figure 5‑1: Classification Tree 8

Figure 5‑2. Torsional Pendulum Thrust Stand 9

Figure 5‑3: Piezoelectric Sensor Concept Design 11

Figure 5‑4: Bending Load Cell Concept Design 12

Figure 5‑5 : Compressive Load Cell Concept Design 14

Figure 7‑1:CFD Simulation of Thruster Nozzle 20

Figure 7‑2 Velocity Streamlines around Thruster Nozzle 20

Figure 7‑3 Velocity Profile of Thruster Nozzle 22

Figure 7‑4:First CAD model prototype 23

Figure 7‑5:Second CAD model prototype 25

Figure 7‑6: Final CAD model prototype 27

Figure 7‑7 Load Cell Nozzle Coupling 29

Figure 7‑8: Three nozzle coupling 29

Figure 7‑9: Load Cell Assembly 30

Figure 7‑10 Load Cell Assembly in Calibration Configuration 31

Figure 8‑1: Flowchart of experiment procedure 37

Figure 9‑1: Physical apparatus isometric view 39

Figure 9‑2: Physical apparatus top-down view 40

Figure 9‑3: Load cell thruster and three nozzle holders 40

Figure 10‑1: Thrust force versus outlet diameter 42

Figure 10‑2: Thrust force versus number of nozzles actuated 43

Figure 10‑3: Thrust force versus supply pressure 44

# List of Tables

Table 5‑1. Torsional Pendulum Thrust Stand Components List 9

Table 5‑2: Piezoelectric Sensor Parts List 11

Table 5‑3: Bending Load Cell Parts List 12

Table 5‑4: Compressive Load Cell Parts List 14

Table 6‑1. Design Feasibility Matrix 17

Table 6‑2. Scoring methodology for weighted matrix 17

Table 6‑3. Scoring matrix used to determine a final concept. 18

Table 7‑1: First design iteration parts list 23

Table 7‑2: Second iteration design parts list 25

Table 10‑1: Final budget 46

Table 10‑2: Purchase orders 47

List of Equations

[Equation 1 9](#_Toc100157093)

[Equation 2 21](#_Toc100157094)

[Equation 3 21](#_Toc100157095)

[Equation 4 32](#_Toc100157096)

[Equation 5 34](#_Toc100157097)

[Equation 6 38](#_Toc100157098)

[Equation 7 38](#_Toc100157099)

[Equation 8 44](#_Toc100157100)

[Equation 9 45](#_Toc100157101)

# Acknowledgements

This research was conducted at Dalhousie University, located in Halifax, Nova Scotia. It was performed under the direction and supervision of Dr. Seto, Dr. Bauer, Elijah Vautour, and Edward Gregson. The authors Nolan Cain, Srajit Mehrotra, Katrina Mullane, and Anthony Newton, mechanical engineering students from Dalhousie University, conducted the research and design which forms the basis of this report.

# Introduction

Since the first artificial satellite was launched in in 1957, thousands of rockets and satellites have been launched and placed into orbit [1]. As the satellites reach their end of life, they remain in orbit, generating space debris. Space debris is defined as “any piece of machinery or debris left by humans in space” [1]. Over the last 64 years, 11139 satellites have been launched, 7389 are in orbit, while 4219 are currently active according to the United Nations Office for Outer Space Affairs [2]. This leaves 3170 inactive satellites orbiting the Earth along with millions of pieces of debris. This debris consists of pieces of space craft, tiny flecks of paint, and parts of rockets. Most space debris is orbiting in low Earth Orbit (LEO) and is traveling at speeds up to 30 000 km/h [3]. The volume and high velocities of space debris in LEO pose a safety risk to people and property in space and on Earth.

The goal of the Robotic Spacecraft Simulator (RSS) is to simulate the deorbiting of space debris. By using two RSS’s, one acting as space debris and the other a chaser that captures the debris, the simulation of deorbiting space debris can be modelled in two dimensions (2D). From this simulation, the fault tolerance, attitude determination control, and safety hazards associated with deorbiting space debris will be determined.

The current RSS is a small robot that floats on air bearings. The RSS’s 2D movement is controlled via the actuation of air thruster nozzles using electronically controlled solenoid valves. [4]. Figure 1.1 displays an artistic rendering of a space debris deorbiting vehicle. The actual model of the RSS includes a custom 3D printed structure, onboard air supply, air bearings at the base of the structure, eight thrusters located around the base of the structure, an onboard power supply, a custom circuit design and onboard central processing unit. The pneumatic and control systems consist of all equipment necessary to direct the RSS for 2D maneuverability in a microgravity environment [4].

A picture containing text, transport, satellite

Description automatically generated

Figure 1‑1: Artistic rendering of a space debris deorbiting vehicle [5]

Limited research has been conducted on the output force of the eight thrusters and how they affect the control system of the RSS. Furthermore, the thrusters will be mainly used to assist with keeping the RSS stationary requiring small amounts of thrust. The RSS Capstone project is to design and construct a thruster testing apparatus which supports the testing of four thrusters and provides thrust force measurements for one nozzle to allow for characterization of the thrust response. The thrust force only needs to be measured for one nozzle while testing, as it can be assumed that thrusters being actuated simultaneously will produce equal thrust responses. The capability to actuate one to four nozzles individually and simultaneously is required as actuating additional nozzles is assumed to induce a supply pressure drop, thereby reducing the thrust force from the nozzles.

This paper will discuss the formal requirements for the testing apparatus, a functional decomposition of the system, conceptual design ideas, and the design selection process. Following these chapters, iterations on the selected design will be expanded on and the results of the experiment will be discussed. Validation of the experiment and recommendations to improve the RSS are also included to assist the ISL with developing the next version of the RSS.

# Requirements and Goals

The requirements and how they were met for the RSS thruster testing apparatus are presented in tabular form in appendix A.

Project Requirements

The functional components of the thruster testing apparatus include being able to support the testing of at least two thruster nozzles. While recording test data from one thruster nozzle is the requirement, the system must be capable of actuating up to four at one time to determine if there are losses in pressure when more than one thruster nozzle is actuated. The apparatus shall provide thrust force measurements to enable the characterization of transient and steady-state responses from the thruster nozzles and be capable of measuring thrust force responses up to 2 N. Due to the low thrust forces, the apparatus should measure thrust on a resolution of 0.01 N.

The RSS has a limited air supply so the testing apparatus must be capable of measuring both the on-board system storage pressure (0 – 3000 psi) as well as the thruster supply pressure (0 – 110 psi). This will determine the rate of pressure loss during thruster actuation, the maximum runtime of the RSS, and how inlet supply pressure effects thrust output. Due to the testing apparatus being subjected to air up to 110 psi, the apparatus must be designed and constructed to withstand these conditions.

In addition to the thruster testing apparatus, a CFD model must be generated to allow for validation between experimental and simulated test results. The CFD model will permit for testing of various thruster nozzle outlet diameters (0.2 – 1.0 mm) with the results being compared to the experimental results. Lastly, the project must not exceed the budget of $2000 CAD.

Project Goals

In addition to the project requirements set by the stakeholder, there are several goals the RSS capstone team aims to achieve. After developing a successful testing apparatus and characterizing the current thruster nozzles on the RSS, it is desired to test nozzles of various sizes and geometries. By making the testing apparatus configurable to test different nozzles, tube lengths, and valves, the recorded data can be compiled to determine the optimal equipment for the RSS.

# Stakeholders

The stakeholder map displayed in Figure 3.1 outlines the core, direct, and indirect stakeholders involved with the RSS capstone project.

Diagram

Description automatically generated

Figure 3‑1: Stakeholder Map

# Functional Decomposition

Upon reviewing the user requirements document received from the project advisors [6], a functional decomposition of the RSS capstone project was developed. The functional decomposition allowed for identification of the inputs, outputs, and internal requirements of the project. By breaking down large functions into basic sub-functions, existing concept designs were matched to the basic functions. This allowed for complete comprehension of the customer requirements [7]. Developing the functional decomposition also enabled the identification of system boundaries, the disconnection of function from form, and increased the number of design concept combinations.

The goal of the RSS capstone project is to provide a safe and inexpensive method of measuring thrust output from a thruster nozzle up to 2 N, generated from a supply air tank that is regulated to a maximum of 110 psi. The functional decomposition of the thruster testing apparatus is displayed in Figure 4.1. The inputs include the interchangeable materials (thruster nozzles, tubing, and valves), energy in the form of pressurized air and electrical power, manual adjustments of the air supply, and signal inputs to control the air flow to the thruster nozzles. From the inputs, the following outputs are generated through the functional system: pressure data, thrust data, heat, and air. Using the inputs and outputs, the project was broken down into subfunctions that the system must perform.

Diagram

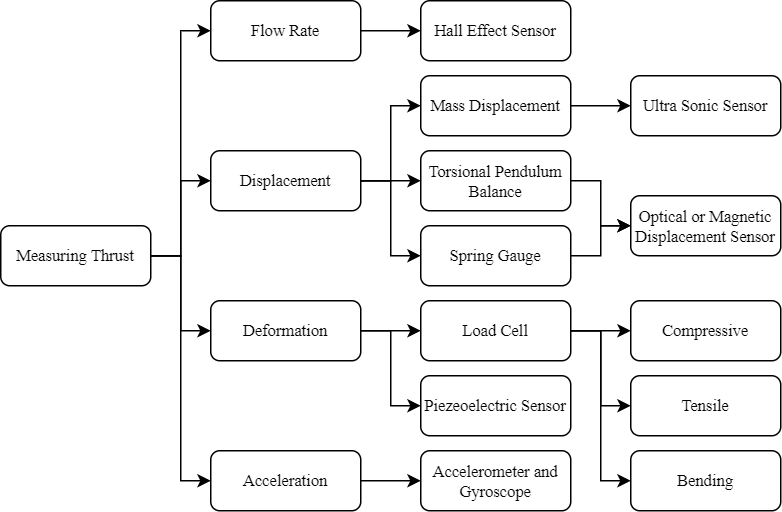
Description automatically generated

Figure 4‑1: Functional Decomposition

# Design Concepts

Many concepts to measure thrust were explored including measuring deformation, displacement, acceleration, and volumetric flow rates. These concepts are expanded on via a classification tree displayed in Figure 5.1.

Figure 5‑1: Classification Tree



From the classification tree, collaboration with the group members, and feedback from the project advisors/clients, it was decided to move forward with exploring the torsional pendulum balance, piezoelectric sensor, bending load cell, and compressive load cell concepts.

Pressure Determination

The inlet air pressure of the thruster nozzles (operating pressure) shall be measured with a Commercial Off-The-Shelf (COTS) digital pressure sensor. Concepts were explored involving pitot tubes, analog barometers, analog pressure gauges, and digital pressure sensors, however, because the transient air pressure in response to thruster actuation must be measured and recorded digitally, a digital pressure sensor is desired. Digital pressure sensors are available for purchase from a variety of manufacturers within the operating pressure range of 0 to 100 PSI for $25 to $160 CAD.

Torsional Pendulum Thrust Stand

A torsional beam utilizes a rotating arm attached to two bearings where there exists a torsional spring, which provides a restoring force normal to gravity. A thruster attached to one end of the rotating arm produces force once activated, rotating the arm. An optical or magnetic sensor measures the displacement of the rotating arm relative to its original position. A counterweight balances the mass of the arm, thus preventing any moment on the bearings. Operating on the premise of Hooke’s Law, thruster force can be calculated:

Equation 1

Where corresponds to thruster force, is the spring constant of the torsional spring, and is the displacement of the rotating arm.

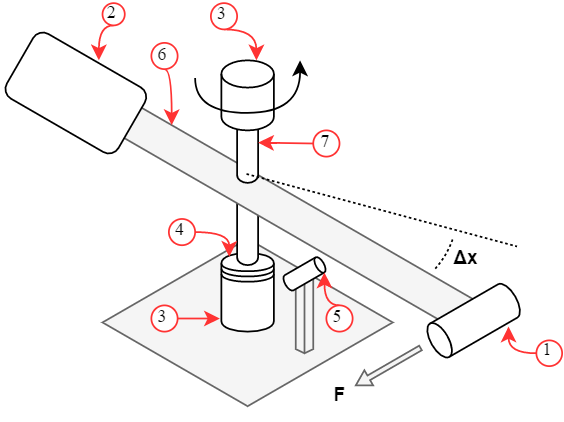


Figure 5‑2. Torsional Pendulum Thrust Stand

Table 5‑1. Torsional Pendulum Thrust Stand Components List

|  |  |
| --- | --- |
| **Part Number** | **Item** |
| 1 | Thruster Nozzle |
| 2 | Counterweight |
| 3 | Bearing |
| 4 | Torsional Spring |
| 5 | Displacement Sensor (Magnetic or Optical) |
| 6 | Rotating Arm |
| 7 | Rotating Shaft |

Advantages

Torsional pendulum designs offer the greatest thrust measurement accuracies (up to ± 1 μN) and resolutions (up to 1 μN) when compared to conventional load cell designs [8]. They are therefore optimal for thrusters with low thrust-to-mass ratios. While these specifications are excessive when compared to the project requirements, such a design would facilitate the testing of nozzles with lower thrusts designed at Dalhousie University.

Disadvantages

The method in which thrust is measured in the torsional pendulum design requires a number of expensive components and sensors, including a displacement sensor, flexural bearing pivots, dampers, and a sophisticated calibration system. This design therefore incurs greater project cost and schedule risk in acquiring materials and executing design and assembly.

Furthermore, this design requires a greater analysis of uncertainties in the measuring apparatus when compared to load cell designs. Uncertainties such as friction due to rotation, mechanical connections from propellant lines which may not provide stable contributions to the spring force, transmission of vibrations from the environment to the pendulum, etc., all must be analyzed carefully to characterize their effects on the thrust force measurements.

Piezoelectric Sensor

This design comprises of measurement of thrust force by measuring deformation in piezoelectric sensor.

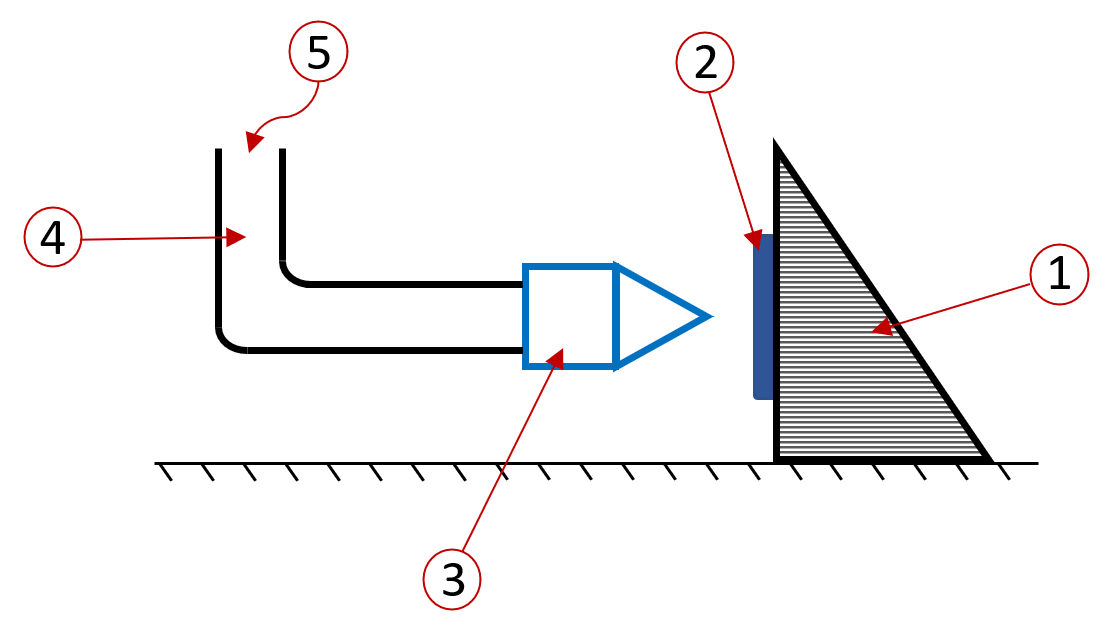


Figure 5‑3: Piezoelectric Sensor Concept Design

Table 5‑2: Piezoelectric Sensor Parts List

|  |  |
| --- | --- |
| **Part Number** | **Item** |
| 1 | Fixed Stand |
| 2 | Piezoelectric Sensor |
| 3 | Thruster Nozzle |
| 4 | Tubing |
| 5 | Thrust Input |

Piezoelectric sensors are placed on a stand at a constant distance from thrusters. The sensors are placed as close to the nozzles as possible without touching the nozzle. On actuation thrust force applies a stress on and deforms the piezoelectric sensors. The sensors measure this deformation and stress, which can then be used to calculate the applied thrust force.

Advantages

Piezoelectric sensors pose a high accuracy and rate of output which is essential to measure the transient state of thrust force. This design is inter-compatible such that nozzles and tubing can be easily swapped without the need of re-calibration of the device. Piezoelectric sensors are inexpensive and thus reduce the overall cost of the test apparatus.

Disadvantages

This design works by applying thrust force directly on the piezoelectric sensors. This setup does not mimic the conditions of the RSS, in which the major thrust force is obtained from allowing the pressurized air to expand to standard air pressure rather than direct thrust force.

Bending Load Cell

This concept utilizes a load cell as a transducer to measure thrust force via deformation as shown in Figure 5.4.

Diagram

Description automatically generated

Figure 5‑4: Bending Load Cell Concept Design

Table 5‑3: Bending Load Cell Parts List

|  |  |
| --- | --- |
| **Part Number** | **Item** |
| 1 | Load Cell |
| 2 | Thruster Nozzle |
| 3 | Thruster Nozzle Stand |
| 4 | Manifold and Solenoid Valves |
| 5 | Microcontroller |
| 6 | Air supply tank |
| 7 | Thruster Nozzle Stand |

A load cell is placed on a base and is connected to a microcontroller. A nozzle is fixed to the top of the load cell, while the other 3 nozzles are fixed on a separate stand. The nozzles are all connected to a manifold of solenoid valves which are wired to the microcontroller and attached to an air supply tank. This design allows for the actuation of up to four thrusters simultaneously and records the data output from one nozzle. When the solenoid valve is commanded to actuate the thrusters, the output force from the nozzle mounted on the load cell will cause a deformation. Strain gauges internal to the load cell will measure the mechanical deformation and convert it to an electrical voltage output signal. The output signal will be recorded by the microcontroller and used to calculate the thrust force produced by the nozzle.

Advantages

Load cells are available in a range of load capacities and resolutions. Due to the low force load of maximum 2 N and resolution requirement of 0.01 N, a micro load cell is desired as they can reach resolutions of 0.0001 N. This design allows for easy access to all thruster nozzles which increases the configurability of the apparatus. The thruster nozzles and tube lengths can easily be switched to test different hardware and determine which thruster nozzle is preferred for the application. Load cells have been used successfully in previous experiments to measure thrust and is well documented online.

This design poses another advantage with its simplicity. There are no moving parts while the thruster nozzle is being actuated, which reduces risk for error. This device can also be calibrated simply via applying a known mass.

Disadvantages

As the load cell is refined for smaller resolutions, the price increases. Therefore, the quality of the load cell may be restricted due to the project budget.

Compressive Load Cell

In this design, a load cell is used as a transducer to measure thrust force.

Diagram

Description automatically generated

Figure 5‑5 : Compressive Load Cell Concept Design

Table 5‑4: Compressive Load Cell Parts List

|  |  |
| --- | --- |
| **Part Number** | **Item** |
| 1 | Load Cell |
| 2 | Reference Nozzle Stand |
| 3 | Thruster Nozzle |
| 4 | Microcontroller |
| 5 | Air Supply Tank |
| 6 | Manifold and Solenoid Valves |
| 7 | Thruster Nozzle Stand |

A load cell is placed on a base and is wired to a microcontroller. One nozzle is fixed to the top of the load cell. Three other nozzles are mounted at the same height as the reference nozzle on a separate stand. All nozzles are connected individually to a manifold of four solenoid valves, connected to a pressure tank. The solenoid valves are wired to the microcontroller. This design allows one to four nozzles to be actuated simultaneously, while only measuring the thrust force of one nozzle. When the solenoid valve is commanded by the microcontroller to actuate the reference nozzle, the downwards thrust force compresses the load cell. The induced strain within the load cell is measured as an analog input to the microcontroller and used to calculate thrust force produced by the nozzle.

Advantages

Load cells are available in a range of load capacities and resolutions. As thrust forces are expected to reach a maximum 2 N and the required measurement resolution is 0.01 N, a micro load cell is desired as they can deliver resolutions of 0.0001 N. This would allow for increased resolution of the measured thrust force, improving the accuracy of the characterized thrust force. This design allows for easy access to all thruster nozzles which increases the configurability of the apparatus. The thruster nozzles and tube lengths can easily be switched to test different hardware and determine which thruster nozzle is preferred for the application. Load cells have been used successfully in previous experiments to measure thrust and is well documented online.

The mechanical simplicity of this design is also a desirable attribute. After the apparatus has been configured with the desired tube lengths and thruster nozzles, there are no moving parts when undergoing testing. This reduces the sources of error in the measurement and makes this design simpler to construct and assemble.

This design requires simple calibration. The initial reading of the load cell can be calibrated based on the known mass applied (mass of thruster nozzle setup compressing load cell).

Disadvantages

As the load cell is refined for smaller resolutions, the price increases. Therefore, the quality of the load cell may be restricted due to the project budget. As the thruster nozzle is propelling air upwards, the energy required to raise the gravitational potential energy of the air may impact the thrust force generated by the nozzle. Additionally, this may violate the condition of choked flow assuming the nozzle exits into ambient pressure.

# Concept Evaluation

Methods used to evaluate the design concepts include performing concept screening and concept scoring. Concept screening evaluated each concept again mandatory criteria and eliminated any that did not meet the requirements. Concept scoring was performed on the remaining design concepts to determine the design that best meets the desired requirements.

## Concept Screening

Each concept was screened based on its feasibility. The feasibility depends on the practical extent to which a design can be completed successfully [9]. The cost, performance, complexity, accessibility, and literature availability were compared for each concept to determine what designs met the requirements for the RSS capstone. Cost must remain under $1250 ($2000 budget - $750 for pneumatic set up), the resolution should meet the requirement of 0.01 N as requested by the ISL lab, and the accuracies should be reasonable to have a small uncertainty in the measurements. Both operational and technical complexity were considered. The operational complexity includes calibration, using the associated software, obtaining results, and the configurability of the setup, while the technical complexity covers the complexity of integrating mechanical, software, and control systems. Manufacturability and material accessibility were also considered to analyze the difficulty of machining and assembling the designs, and to estimate lead times for ordering parts which is critical for the project timeline. Lastly, the availability of literature on past uses of the design concepts were compared to understand what has been done successfully in previous thruster testing projects. Table 6.1 displays the design feasibility matrix for the four concepts.

Table 6‑1. Design Feasibility Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Torsional Pendulum Balance | Piezoelectric | Bending Load Cell | Compressive Load Cell |
| Cost of Design Specific Components ($CAD) | > $2000 | $11.50 - 400 | $100-800 | $80-720 |
| Resolution | 0.000001 N | 0.0028 - 0.0001 N | 0.001-0.0001 N | 0.001-0.0001 N |
| Uncertainty | 0.00001 N | 1% Full Scale | 0.0037 – 0.0007 N | 0.0037 – 0.0007 N |
| Operational Complexity | High | Low | Low | Low |
| Manufacturability | High | Low | Low | Low |
| Material Accessibility | Specialized Components | Commonly Available Components | Commonly Available Components | Commonly Available Components |
| Technical Complexity | Medium-High | Low | Low-Medium | Low-Medium |
| Literature Availability | High | Low | High | Low |

Based on the feasibility matrix, consideration of the torsional pendulum balance will not proceed as a potential design concept for measuring the thrust output force due to exceeding the project budget, it’s use of highly specialized components, and it’s increased complexity. The piezoelectric sensor, bending and compressive load cell will continue to be analyzed as potential solutions for the RSS capstone.

## Concept Scoring

Each concept was scored based on absolute scoring with each criterion being weighted out of 100%. Table 6.2 outlines the absolute scoring scale while Table 6.3 scores the concepts.

Table 6‑2. Scoring methodology for weighted matrix

|  |  |
| --- | --- |
| 1 | Useless, very poor |
| 2 | Poor, inferior |
| 3 | Acceptable |
| 4 | Good, superior |
| 5 | Excellent, much superior |

Table 6‑3. Scoring matrix used to determine a final concept.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Weighting | Piezoelectric | Bending Load Cell | Compressive Load Cell |
| Cost | 0.2 | 4 | 3 | 3 |
| Operational Complexity | 0.1 | 3 | 3 | 4 |
| Manufacturability | 0.15 | 2 | 5 | 5 |
| Previous use in similar applications | 0.25 | 1 | 5 | 2 |
| Mimicking observed flow conditions | 0.3 | 2 | 5 | 2 |
| Total |  | 2.25 | **4.4** | 2.85 |

Cost and previous use cases in similar applications were the most important factors in the consideration of the final design. This is because the project has a small budget relative to the cost of the highly specialized components of each design, leaving little room to prototype freely with materials. Furthermore, the table places strong emphasis on previous use cases to ensure adequate literature resources are available to assist in the design of the thrust measuring apparatus. The math used to derive force from the measurement outputs of each design is complex, and there are multiple sources of measurement uncertainty, therefore having literature on hand to assist is essential if the project is to meet its cost and schedule requirements. The flow criterion represents how well the apparatus mimics the flow conditions the thrusters on the RSS experience under standard operational environment and thus is given the highest weight. Operational complexity and manufacturability were given the smallest weight as the capstone team will ultimately be in charge of assembling the hardware and will have access to technicians and resources at Dalhousie University. After weighting all criterion, and scoring each design, the bending load cell concept was selected as the final concept to proceed with.

# Design Iterations

After selecting a final concept, two prototypes were developed: a bending load cell CAD model and a CFD model. The following sections will expand on the developed prototypes and how they have been iterated through project development.

## CFD Model

Design development was initiated by building a CFD model. CFD simulations were conducted for various outlet diameters of the Creality® brass nozzle. The inlet diameter, constrained by the hose was kept constant at 1.59 mm. The nozzle was exposed to boundary conditions of 60 psi total inlet pressure and 14.6959 psi static (0 psi gauge) environmental air pressure (standard air pressure) at outlet. These simulations included the inlet hose, nozzle and a large area surrounding the nozzle to simulate environment. The CFD model originally developed in SolidWorks® and was later developed in ANSYS® to improve the accuracy of the results.

### SolidWorks Model

Originally CFD simulations were performed using SolidWorks® Flow Simulation. High density (fine) meshes were generated including local mesh sizing around the nozzle converging area and outlet to obtain accurate results. This software utilized the Navier-Stokes equation model to solve steady state flow and the solution converged after 119 iterations.

Results obtained for the 0.4 mm OD nozzle by this simulation were intuitive and aligned with expectations of compressible and supersonic flow.

* Maximum Exit Velocity of 387.3 m/s

As expected, velocity reaches local MACH 1 while entering the neck of the nozzle and decelerated back to being subsonic after being released out of the nozzle.

* Steady Exit Mass Flowrate of 9.753 x 10-5 kg/s

A steady mass flow rate was obtained at the exit of the nozzle. Since the diameter of the nozzle exit is 1.5 mm, such a result in the magnitude of 10-5 kg/s is expected.

* Total thrust force of 41.5 mN generated by the nozzle.

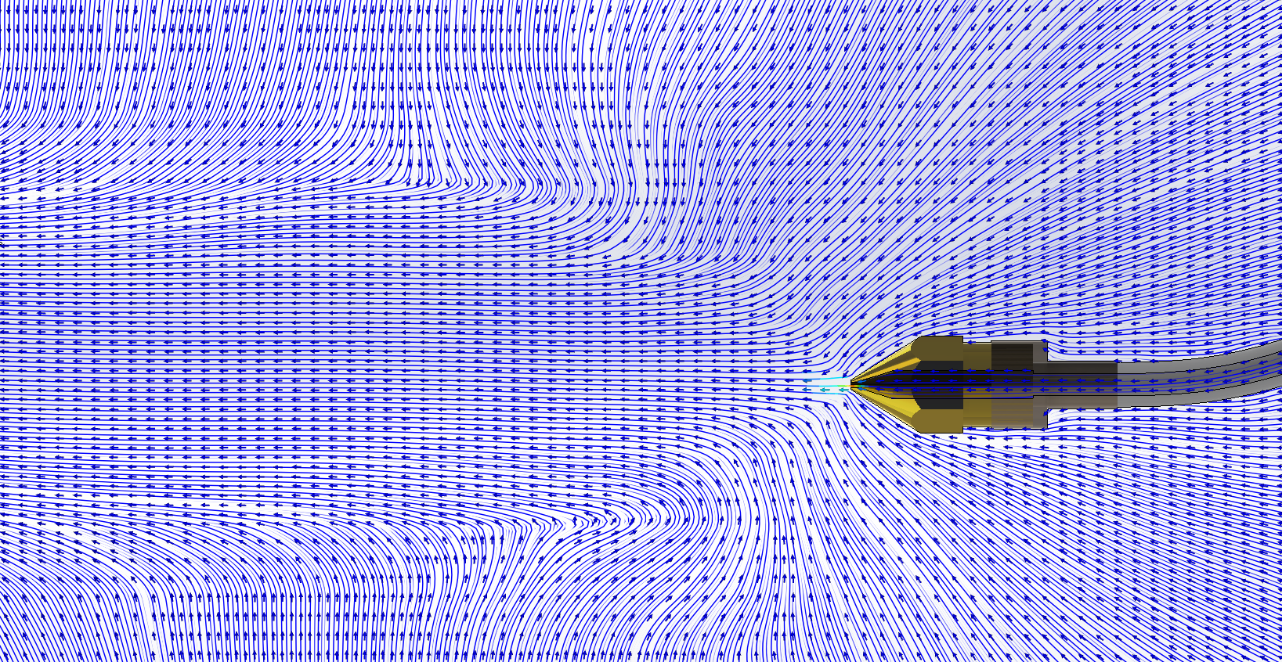


Figure 7‑1:CFD Simulation of Thruster Nozzle

Chart, surface chart

Description automatically generated

Figure 7‑2 Velocity Streamlines around Thruster Nozzle

Due to lack of literature pertaining to internal flow simulation accuracy of SolidWorks® Flow Simulation and on client’s insistence SolidWorks® CFD model was deprecated for ANSYS®.

### ANSYS Model

ANSYS Fluent model on student license was utilized to generate CFD simulations for multiple outlet diameters of the Creality® brass nozzle. ANSYS Fluent used Reynold’s Averaged Navier-Stokes (RANS) mass and momentum continuity equations to simulate the flow dynamics. Multiple flow models were tested within ANSYS Fluent solver to obtain the best suited method to perform CFD.

#### k-epsilon equation solver

Initially k-epsilon solver was used for CFD simulations for the 0.4 mm OD nozzle. The solver was unable to sufficiently account for boundary layer formations on the walls of the nozzle and the outlet. Considering the nozzle used is a purely converging nozzle, the outlet is also the throat of this nozzle and boundary layers play a significant role in the fluid dynamics in this area. As a result, the solution diverged instead of converging.

#### k-omega equation solver

k-omega solver with compressibility corrections was used to perform steady state CFD simulations for the 0.2 mm, 0.3 mm, 0.4 mm, 0.5 mm, 0.6 mm, 0.8 mm, and 1.0 mm outer diameter nozzles. Second order upwind differential scheme and a Hybrid least-squares based method was used to perform these simulations. The values of velocity (Ve), Density (ρe), and gauge pressure (Pe) at nozzle exit were probed from the results and used to calculate thrust force generated as follows:

Equation 2

Equation 3

Where me is mass flow rate at exit and Ae is the exit area.

Results for thrust force generated at steady state are:

* 0.2 mm = 14.56 mN
* 0.3 mm = 40.19 mN
* 0.4 mm = 53.78 mN
* 0.6 mm = 79.51 mN
* 0.8 mm = 125.6 mN
* 1.0 mm = 55.83 mN

A picture containing background pattern

Description automatically generated

Figure 7‑3 Velocity Profile of Thruster Nozzle

The ANSYS student license applied limitations of a maximum of four solver processes, and a maximum of 512,000 cells within the volume mesh which resulted in coarse mesh and inaccurate solutions. To minimize this effect and increase accuracy near the nozzle convergence and outlet, one half of the symmetry of fluid profile was uses to perform simulations and the volume of the environment had to be significantly reduced compared to the SolidWorks model. This produced backflow and the solver requiring 300 to 400 iterations to converge. To limit the time consumed, each nozzle was simulated until backflow occurred and the results were sampled across multiple iterations.

## CAD Model

The CAD model went through several design iterations before arriving at the final model. Improving the layout was paramount to ensure all parts could fit on the plate while minimizing the plates dimensions. Having all parts mounted on the base plate was desirable for ease of transportation. Additionally, mounting components went through several iterations to improve functionality, machinability, and minimize required materials.

Iteration One

A bending load cell is a transducer that converts force to an electrical signal and typically consists of four strain gauges assembled in a Wheatstone bridge configuration [10]. The bending configuration was selected over the compressive configuration as it better models the choked flow conditions the thruster will experience and will produce a greater strain under the same applied load. Additionally, the bending load cell was recommended over the compressive load cell by clients and other Dalhousie faculty. Due to the voltage output of a load cell being too small for the selected Arduino Uno to measure, a load cell amplifier will be used to increase the readability of the output voltages. Figure 7.1 displays the first bending load cell prototype developed, while Table 7.1 lists the associated parts.

A picture containing indoor

Description automatically generated

Figure 7‑4:First CAD model prototype

Table 7‑1: First design iteration parts list

|  |  |
| --- | --- |
| Part | Item |
| 1 | Load Cell |
| 2 | Measured Thruster Nozzle |
| 3 | Thruster Nozzles |
| 4 | Manifold and Solenoid Valve |
| 5 | Air Supply Tank |
| 6 | Arduino Uno |
| 7 | Load Cell Amplifier |
| 8 | Base |

As shown in Figure 7‑4:First CAD model prototype, a bending load cell is mounted to the aluminum base plate with one thruster nozzle mounted on its top surface. This is the only thruster nozzle where the thrust force is measured during experiment. There are also three nozzles mounted to the base plate such that they can be simultaneously actuated during experiment. This satisfies the requirement that up to four nozzles can be actuated to determine the effects of simultaneous thruster actuation on the output thrust force.

This design iteration includes the basic pneumatic components required to function such as: air cannister, manifold, and solenoid valves. The solenoid valves are actuated using the microcontroller. The analog load cell signal is fed into a load cell voltage amplifier and ADC board, which then sen [11]ds a digital signal to the microcontroller for recording.

Iteration Two

The second design iteration improved the initial design concept in these main areas: CAD model accuracy, manufacturability, pneumatic components, and mounting solutions. The second design iteration is displayed in Figure 7-2, and the parts are listed in Table 7-2.

Diagram, engineering drawing

Description automatically generated

Figure 7‑5:Second CAD model prototype

Table 7‑2: Second iteration design parts list

|  |  |
| --- | --- |
| Part | Item |
| 1 | Air Supply Tank |
| 2 | Second Stage Regulator |
| 3 | Solenoid Valves |
| 4 | Manifold |
| 5 | Thruster Nozzle Holder |
| 6 | Load Cell (testing orientation) |
| 7 | Load Cell Calibration Stand |

As shown in Figure 7‑5:Second CAD model prototype custom manufactured components were updated to include mounting holes such that they could be bolted into threaded holes into the ½” aluminum base. Through research, it was found that typical thickness of steel, M3 nuts varied around 0.1” . The base plate was to be manufactured of aluminum to decrease weight, cost, and machining time. Because threads of aluminum are known to be weaker than threads of steel, it was determined that an increase base thickness of ½” was desired. To accommodate all current components in CAD and allow space for electronics, the base plate dimensions were extended to 260 x 450 mm.

In this design iteration, the air supply tank was modelled to reflect the geometry of the air cannister purchased for this project and the RSS. The second-stage regulator was added to the model as it is necessary to regulate the air cannister supply pressure to an acceptable pressure for the solenoid valves. To mimic the RSS design, the same second-stage regulator was purchased for this project.

The electronic components (not depicted in the CAD model) such as the microcontroller, load cell amplifier, and DAQ system, had not yet been selected for purchase.

### Iteration Three

The third design iteration resulted in the final thrust measurement apparatus design. This design includes all pneumatics, mounting solutions, hardware, and sensors necessary to meet the project requirements.

Diagram, engineering drawing

Description automatically generated

Figure 7‑6: Final CAD model prototype

Table 7‑3: Final design parts list

|  |  |
| --- | --- |
| Part | Item |
| 1 | Air Supply Tank |
| 2 | Second Stage Regulator |
| 3 | Solenoid Valves |
| 4 | Manifold |
| 5 | Thruster Nozzle Holder |
| 6 | Load Cell (testing orientation) |
| 7 | Load Cell Calibration Stand |
| 7a | Load Cell (calibration orientation) |
| 8 | Three Thruster Nozzle Holder |
| 9 | Circuit Board |
| 10 | Base |
| 11 | Pressure Transducer |

To avoid the difficulties of accessing nuts underneath the base, all mounting hardware screws into threaded holes in the ½” thick aluminum base. To support the mounting of all components, the base was extended in size to be 12” x 18”. This size supported space for all required components and was readily available for purchase.

The air tank model was revised to reflect the size and shape of the Ninja air tank which had been purchased for this project. A mounting system was added to the design to secure the air cannister to the base. Two 3D printed holders rest below the air tank, which is secured to the base using two brackets, made from bent aluminum flat bar. Rubber strips were place between the surfaces, to distribute the load on the air cannister when mounting, allow for compression when tightening the mounting bolts, and increase friction on the air cannister.

Air from the supply tank is fed through the second stage regulator to reduce the pressure from the air cannister’s output of 300 psi to the desired supply pressure which is manually regulated between 0-100 psi. An electronic pressure transducer is connected to the second stage regulator’s output so that pressure readings can be logged using the microcontroller. The output from the pressure regulator is fed into to manifold supply port, which supplies regulated air to the four connected solenoids. These solenoids are actuated using the microcontroller.

The solenoid valves and manifold were mounted to the base, instead of a raised stand. This design change was made as having the solenoid outputs at the same height as the load cell input was deemed unnecessary, and it was desirable to pre-bend the tubes between the solenoid valves and nozzle couplings. Moving the manifold to the base resulted in all tubes being bent in an S-curve, this resulted in lower reaction forces when the load cell deformed under thruster actuation.

The previous nozzle assemblies depicted using a nozzle and COTS coupling were altered, due to issues on the RSS using the couplings. It was therefore desirable to design and machine custom couplings to suit project needs. This resulted in two custom couplings being created: the load cell mount, and the three-nozzle coupling, shown in Figure 7‑7 and Figure 7‑8, respectively.

A picture containing rectangle

Description automatically generated

Figure 7‑7 Load Cell Nozzle Coupling

A picture containing metalware, hinge

Description automatically generated

Figure 7‑8: Three nozzle coupling

Both couplings were designed to have M6 threaded ends to accept the Creality nozzles in one end, and the M6 barb fittings in the other end. The load cell coupling was designed to mount to the top of the selected load cell using two bolts on both sides. The extended leg length on one side exists to accommodate known masses when the load cell is in its calibration position. The three-nozzle coupling was designed to be mounted on top of the three-nozzle stand using four M3 bolts. These bolts pass through the three-nozzle coupling and thread into embedded nuts in the 3D printed three-nozzle stand. The three-nozzle coupling is used to connect the nozzles to the remaining three solenoid outputs. This allows the user to determine the effects of simultaneous thruster actuation on the output thrust force.

The selected load cell from Laumas was mounted to the base using a bracket instead of a plate to decrease material usage, machining time, and cost. The bracket was ribbed on its sides to increase its stiffness and decrease vibrations during testing. The bracket and load cell nozzle coupling were fixed to the load cell by threading M3 bolts into the threaded holes in the load cell. When the bolts are tightened a residual stress is induced in the load cell. Figure 7‑9 illustrates the load cell assembly, including the load cell, bracket, coupling, nozzle, barb fitting, and required hardware.

A picture containing engineering drawing

Description automatically generated

Figure 7‑9: Load Cell Assembly

This design iteration includes a calibration stand to mount the load cell horizontally to allow the user to apply known forces and calculate the calibration values based on the output voltages. To avoid altering the residual stresses in the load cell, the load cell assembly is moved from the base to the calibration configuration as a solid unit. The calibration configuration is demonstrated in Figure 7‑10.

Diagram, engineering drawing

Description automatically generated

Figure 7‑10 Load Cell Assembly in Calibration Configuration

# Electrical Design

Transducer Selection

The requirements in Appendix A outline that the force is to be measured with a maximum resolution of 0.01 N (1 gram of force), with a maximum expected force of 2 N. Based on the feasibility matrix in Section 6, a bending load cell was selected.

Two Newtons is equivalent to approximately 200 grams. At the time of selecting a load cell, it was estimated that the hardware mounting the thruster to the load cell would weigh a maximum of 50 grams. Load cells are available in capacity ranges of 100, 250, 500, and >1000 grams. While the 50 grams will not impart a bending force on the load cell in a vertical orientation (instead imparting a compressive force which is not measured by the load cell), in the horizontal calibration position it will impart a bending force proportional to its mass. 200 grams in addition to the 50-gram mass of the thruster mount risks exceeding any load cell with a capacity less than or equal to 250 grams, which would result in permanent damage to the load cell. Therefore, a load cell with a capacity of 500 grams was selected, leading to a mass safety factor of 2.

A search was carried out to find the optimal 500 g load cell, in terms of cost, measurement accuracy, and ease of mounting hardware to the load cell. Two load cells were selected for additional investigation, from Laumas and Tacuna systems. The Laumas load cell was selected based on its lower cost, superior accuracy, and ease of mounting. A selection matrix was not required due to the limited number of options available for the project.

The theoretical measurement accuracy of a load cell can be calculated using Equation 4, with data derived from the manufacturer’s specification sheets [12]. The accuracy is expressed in terms of the load cell’s full scale (i.e. the maximum load in mass units which can be applied to the load cell)

Equation 4

Where:

Where combined error is proportional to hysteresis and non-linearity error.

The accuracy of the Laumas 500 g load cell was calculated to be 0.23 g, superior to similar load cells from Tacuna systems.

It is important to note that the load cell output is an analog electrical voltage signal, approximately linearly proportional to the applied load. Therefore, it essentially has an “infinite” measurement resolution. However, the analog signal must be converted to a digital signal using an analog to digital converter (ADC). The ADC slices the load cells analog output into a number of discrete digital values equivalent to the ADC’s resolution. If the ADC operates at 5V, then it can pick up increments of 0.005 V (5V/1024). Unfortunately, the maximum voltage output of many load cells is on the millivolt scale. For example, the Laumas 500g load cell outputs a maximum of 0.005 V. For an ADC with a resolution of 1024, it will effectively only pick up 0 or 500 g.

Therefore, the load cell output either needs to be amplified, such that the voltage signal equivalent to the load resolution required can be picked up by the ADC, or an ADC with a higher resolution must be selected.

Three ADC and amplifier solutions were selected for additional investigation, the SparkFun HX711 combined amplifier and ADC, the RobotShop Load Cell Amplifier Shield, and the Tacuna systems EMBSGB200 load cell amplifier. The SparkFun HX711 was selected due its example use in literature, availability, cost, available Arduino software libraries to easily process load cell data, and sufficient resolution and amplification gain to meet the resolution under worst case conditions.

Worst case conditions include the effects of noise on an ADC’s maximum resolution. Noise reduces the maximum resolution of an ADC and is caused by the sampling rate of the ADC, gain applied to the signal, and the power source. Experimental measurements of the HX711 resolution using its largest gain option, and a wall power source revealed an noise free resolution of 14.3 bits [13], a large reduction from its maximum resolution of 24 bits.

Using the new effective resolution and the following equation to calculate load cell resolution [14]:

Equation 5

Where:

For the Laumas load cell paired with a SparkFun HX711, a theoretical worst-case resolution of 0.48 grams was achieved, successfully meeting the resolution requirements stated by the stakeholders. Since the HX711 is to be powered using a battery, the noise free resolution is likely greater as there is less electrical interference in a battery power source (providing there are no motors or actuators connected to the battery). Therefore, the resolution of the load cell is likely smaller than 0.48 grams.

The accuracy of the ADC/amplifier can be calculated as follows [12]:

Equation 6

By combining the ADC/Amplifier accuracy with the load cell accuracy, the following theoretical accuracy of the total load cell measurement system is calculated:

Equation 7

Resulting in a maximum theoretical measurement accuracy of 0.25 g.

Another important aspect of selecting an ADC is its sampling rate. Supporting literature had shown that similar testing apparatuses used sampling rates ranging from 80 Hz to 200 Hz to adequately capture the transient response of cold gas thrusters. The maximum sampling rate of the HX711 is 80 Hz.

After selecting a load cell and ADC/amplifier, a pressure transducer was selected. The Amphenol SSI P51 pressure transducer was selected based on lead time, compatibility with an Arduino ADC (i.e., it provides a linear voltage output from 0.5 to 4.5 V), ability to measure pressures up to 200 PSI (allowing the stakeholders replace increase their maximum pressure above the 110 PSI requirement), as well as maximum resolution of 0.25 PSI. This transducer’s resolution offers a 10-fold decrease over the analog transducer used on the RSS.

First Software Iteration

A load cell and pressure transducer were selected based on their ability to meet the project requirements. Software was developed to interface with the transducers, and to record and process their output signals.

The following set of software objectives were developed to ensure the proper experiment conditions could be set and required data specified in the requirements could be read in:

* Calibrate load cell using known masses
  + Allows for the proper correlation of load cell output voltage to applied force during experiments
* Read pressure transducer data in units of psi
  + Allows for confirmation that the transducer is reading correct pressure
  + Allows the user to precisely set the experiment pressure
  + Allows for the analysis of pressure changes during thruster actuation
* Read load cell data in units of grams
  + Allows for confirmation of proper calibration
  + Allows for determination of accuracy, resolution of load cell
  + Allows for the measurement of thrust force during thruster actuation
* Turn solenoid valves on/off
  + Allows user to actuate thrusters on/off for set amounts of time depending on experiment
* Record load cell, pressure transducer, and clock time data in a CSV file
  + Allows for post processing in MATLAB or Python if necessary

To ensure the transducers could be tested as soon as they arrived from the suppliers, rather than waiting until the entire apparatus was assembled, the software objectives were broken down into separate Arduino programs which could each be individually tested and debugged:

* Arduino load cell calibration script
* Arduino load cell reading script
* Arduino pressure transducer reading script
* Arduino solenoid control script

After each program was successfully built and tested, plans were made to combine the functionality of each script into one Arduino program. However, due to the limited processing power and memory space of the Arduino, it was determined that the Arduino would be used only to capture data. Any post processing calculations on the data within the Arduino script would delay the capture of data from the transducers, or risk overflowing its memory. Therefore, a second program was developed that would be loaded onto the user’s laptop, to perform any data processing (such as converting the raw voltage values of the pressure transducer into the proper pressure units).

Second Software Iteration

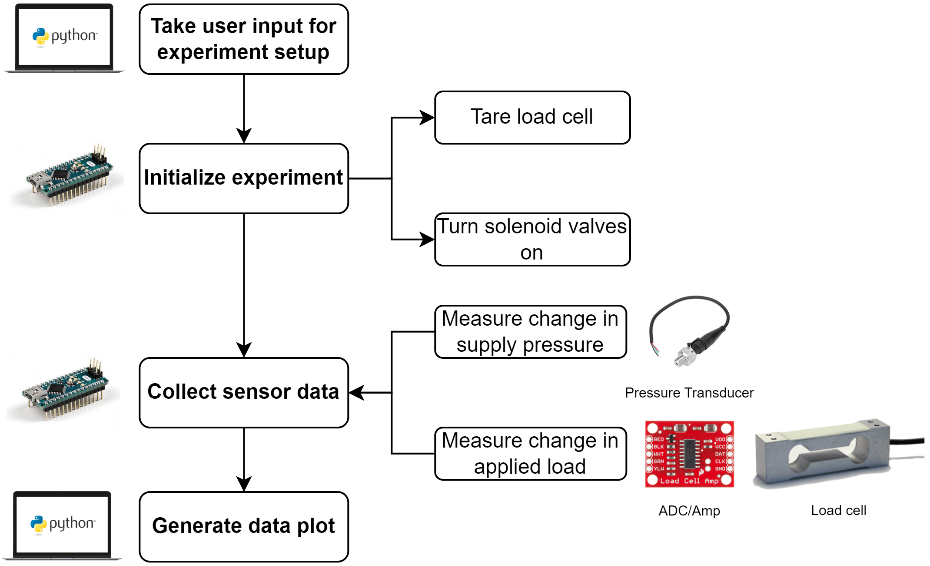


Figure 8‑1: Flowchart of experiment procedure

The final software package involved two Arduino scripts (each to be loaded onto the testing apparatuses Arduino Nano when appropriate), and a Python program loaded onto the user’s computer, as shown in Appendix B.

The first Arduino scripts allows the user to test the functionality of all transducers and perform calibration procedures. This allows the user to ensure all electrical systems are working as intended before carrying out an experiment.

The second Arduino script takes a user input (how long to actuate the thrusters for) from the Python program over serial, and then performs operations such as opening the solenoid valves and reading transducer data over the duration of the experiment. It sends the raw transducer in CSV comma delimited format using serial communication to a Python program, which reads the data live and performs the necessary data conversion calculations. The Python program then saves the processed data into a CSV file and plots the thrust force and pressure curves. A simplified flow chart of this process can be seen in Figure 8‑1.

Subsequent iterations of the program involved prompting the user to enter in the following experiment setup parameters rather than hardcoding the values into the Arduino script. This removed the need for the user to edit the code to change the experiment conditions, such as how long air was to be supplied to the thrusters.

Load Cell Calibration and Unit Conversion

The following load cell calibration method was utilized, based on calibration standards outlined by companies specializing in load cell manufacturing [15]. First, the user is to acquire a precision mass set, or a set of masses whose masses are known (by preferably using a milligram scale). These masses should cover the full scale of the load cell (i.e. 0 to 500 grams). The load cell is then mounted horizontally in the calibration configuration outlined in Section 7.2.3. After loading the calibration script to the Arduino, the user is prompted over Arduino serial monitor to one-by-one place the masses on the load cell mass area and enter the known mass. The Arduino takes a raw reading from the load cell each time a mass is placed on the scale. The Arduino then performs a least squares regression operation, determining the line that best fits the linear relationship between the known mass value and the load cell measured value. From load cell manufacturer specifications, it is known that the relationship between the load cell reading and the applied mass is linear to a high enough degree to be approximated by a first order polynomial equation [15]. This operation results in an output of two variables, aCal and bCal, which can be used to convert the load cell reading to a known mass shown below:

Equation 8

To ensure that the calibration constants aCal and bCal capture the correlation between the load cell output and applied mass accurately, the calibration procedure should be repeated three times, and the constants averaged. A full outline of the calibration process is available in the stakeholder handoff package as described in Section 14.

The calibration constants are then hardcoded into the Python program, allowing the Python program to convert the load cell data into the required format of applied mass of the thruster, in units of force, noting that:

Equation 9

# Final Apparatus

The final design of the apparatus included all components being mounted on an aluminum base plate. After considering multiple layouts, the final arrangement allowed for the circuit boards to also be mounted on the base plate to ease transportation efforts. Figure 9‑1 and Figure 9‑2 display the completed physical apparatus used to measure and record the output thrust force from the nozzles. The electronic components that were neglected from the final CAD model are clearly displayed in these figures. Figure 9‑3 displays the four nozzles the apparatus supports for actuation. All engineering drawings for custom machined parts are included in Appendix C.

An operation manual was developed to assist the ISL with using the apparatus upon completion of the capstone. The user operation manual details the necessary steps to complete pre-testing, testing, and post testing as well as safety precautions to adhere to while operating the apparatus. The operation manual is included in Appendix D.

Due to the use of high-pressure air, a failure modes and effects analysis (FMEA) was conducted to assess safety hazard with the apparatus [16]. The FMEA is included in Appendix E.



Figure 9‑1: Physical apparatus isometric view

A picture containing device, engine, miller

Description automatically generated

Figure 9‑2: Physical apparatus top-down view

A machine on the counter

Description automatically generated with low confidence

Figure 9‑3: Load cell thruster and three nozzle holders

# Results

Three experiments were conducted to gather information on thrust output. All experiments were completed following the pre-testing and post testing procedure in the operation manual, while slight changes were made to the testing section to gather the desired information.

## Thrust Force versus Outlet Diameter

The first experiment examined the effects of nozzle outlet diameter on thrust force. The Creality Ender 3 nozzles currently on the RSS are available with outlet diameters ranging from 0.2 – 1.0 mm. Each size was tested with an inlet supply pressure of 60 psi to mimic conditions on the RSS. Upon completion of testing with the physical apparatus, the results were compared the values generated from the CFD model and are displayed in Figure 1‑1. Thrust force was observed to reach a maximum of 6.81 grams while using the 0.6 mm outlet diameter nozzle. The thrust force began to decrease with outlet diameters larger than 0.6 mm due to the air flow not obtaining a high enough velocity to reach choked conditions. Comparing the experimental data with the values obtained from the CFD model, the results are in good agreement. The CFD model consistently over predicted the output thrust force by 0.3 to 0.7 grams. This error is hypothesized to be due to the simulation not considering frictional effects from wall roughness or boundary layer effects.

Chart, scatter chart

Description automatically generated

Figure 10‑1: Thrust force versus outlet diameter

## Actuation of 1 Nozzle versus 4

The second experiment conducted was to examine the effects on thrust force when actuating one nozzle versus four. This experiment was critical to perform as the RSS will be firing multiple nozzles at a time when station keeping or re-orientating. This experiment was performed concurrently with the first experiment; therefore, an inlet supply pressure of 60 psi was used. The results are displayed in Figure 10‑2. A slight decrease in thrust force was observed when four nozzles were actuated compared to one. This is due to a decrease in supply pressure to each nozzle when more than one valve is opened. Where the decrease in output thrust force is small, the ISL may determine the effects of actuating more than one nozzle negligible while developing propulsion control. Similar to the results from experiment one, the output thrust force follows a parabolic shape while increasing thrust force.

Chart, scatter chart

Description automatically generated

Figure 10‑2: Thrust force versus number of nozzles actuated

## Thrust Force versus Supply Pressure

The last experiement performed analysed the effects of varying the inlet supply pressure on outlet thrust force. For this experiment, the 0.4 mm outlet diameter nozzle was used as it is the current nozzle size implemented on the RSS. The inlet supply pressure was varied from 40 psi to 100 psi in increments of 10 psi. Pressures above 100 psi were not examined due to componenets having maximum allowable pressures of 110 psi. As shown in Figure 10‑3, the thrust force increases linearly as the supply pressure is increased and reaches a maximum of 7.99 grams at 100 psi.

Chart, scatter chart

Description automatically generated

Figure 10‑3: Thrust force versus supply pressure

## Results Summary

Performing the three experiments outlined above and observing the results has led to the following conclusions for the Creality nozzles. To maximize the thrust output for the RSS, the 0.6 mm nozzle with a maximum inlet supply pressure (100 psi) should be used. This combination will allow the RSS to obtain a maximum output thrust force with the currently available parts. However, the goal for the RSS is to maximize the system runtime. System runtime can be determined using Equation 10.

Equation 10

Due to the volume of the air supply cannister being fixed, the only variable that can be modified to maximize system runtime is the outlet volumetric flow rate. Outlet volumetric flow rate (Q) is defined by Equation 11.

Equation 11

Where v is outlet velocity and A is outlet area. To maximize runtime, the outlet volumetric flow rate would need to be minimized as they are inversely proportional. Therefore, a nozzle with a minimum outlet area should be used to decrease the flow rate. The smallest outlet diameter available for the Creality nozzles is 0.2mm. To minimize outlet velocity, a reduced inlet pressure should be used. Therefore, to maximize RSS runtime, the 0.2mm outlet diameter nozzle with a minimum inlet supply pressure should be implemented on the RSS. The RSS current uses the 0.4mm outlet diameter nozzle with 60 psi supply pressure which produces 4.71 grams of thrust force. The 0.2mm outlet diameter was found to produce a thrust force of 1.46 grams when subjected to 60 psi supply pressure. Although using the 0.2mm outlet diameter nozzles would maximize runtime, this thrust force may not be high enough to maneuver the RSS. It is recommended that the ISL perform testing with the 0.2mm outlet diameter nozzle to ensure they are powerful enough to propel the RSS.

# Project Budget

The RSS capstone has an allowable budget of $2000. To track required equipment, purchases, and inventory, a detailed budget has been developed. Approximately $1000 of the budget was allocated towards purchasing the required hardware for the pneumatic setup while nearly $600 was spent on the electronics. Table 8-1 displays the final project budget while tables 8-2 displays the final costs of each purchase order and the remaining budget.

Table 11‑1: Final budget

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Budget No.** | **Qty** | **Item** | **Part #** | **Source** | **Description** | **Lead Time** |
| M2 | 1 | Ninja Cannister and 1st Stage Regulator | NINJATANKLW-90-GRY-PV2 | ANS Gear | Air tank and attached 1st stage regulator | 9-12 days |
| M3 | 1 | Ninja Universal Fill Adapter (UFA) | NINJATANK-FILLADAPTER | ANS Gear | Release air from tank | 9-12 days |
| M5 | 1 | 1/4" OD Nylon Tubing | - | ISL Lab | Tube A on assembly | - |
| M6 | 1 | 1/8" ID Polyurethane Rubber Tubing | - | ISL Lab | Tube B on assembly | - |
| M7 | 1 | 1/16" ID Polyurethane Rubber Tubing | - | ISL Lab | Tube C on assembly | - |
| M9 | 1 | Speed Aire Regulator 1/4IN | 4ZM06 | Grainger Canada | Pneumatic regulator | 2 days |
| M14 | 2 | Plug Square head 1/8" | FAR109A | Grainger Canada | Square head plug, 1/8 in. m pipe threads | 1-2 days |
| M17 | 1 | Pressure gauge 0 - 160 psi | FARPG160CD15 | Grainger Canada | Dry pressure gauge | 2 days |
| M20 | 3 | Thruster Nozzles | - | Amazon | Extruder Nozzles 3D Printer Nozzles 0.2mm-1.0mm | 2 days |
| M21 | 6 | Aluminum Low-Pressure 1/16" Barb ID x M6 x 1mm Male | 5058K956 | Mcmaster Carr | Custom Nozzle Assembly Barb Fitting | 3-5 days |
| M22a | 1 | 3 Nozzle Holder | - | Metals r us | 0.5"x1.5"x1.25" Aluminum | 1 day |
| M22b | 1 | 3 Nozzle Stand | - | 3D Print |  | - |
| M23 | 1 | Load Cell Bracket | - | Metals r us | 0.75"x1"x1" Aluminum | 1 day |
| M24 | 1 | Nozzle Mount | - | Metals r us | 1.5"x3/4"x1.25" Aluminum | 1 day |
| M25 | 4 | M3 x 0.5 mm, 16 mm long, Socket Head | - | Fastenal | Mounting load cell bracket to base | 3-5 days |
| M28 | 2 | Air Cannister Holder | - | 3D Print |  | - |
| M29 | 1 | Air Cannister Bracket | - | Metals r us | Aluminum flat bar 1/8" thick 3/4" wide 30" long | 1 day |
| M31 | 1 | Base | - | Metals r us | Base plate for entire assembly 0.5"x10"x18" | 1 day |
| M32 | 4 | Second Stage Regulator Stand | - | 3D Print |  | - |
| M33 | 1 | Air Cannister rubber sheet | WWG10U461 | Grainger Canada | 1/8"x12"x12" recycled rubber | 1-2 days |
| M34 | 1 | Glue | - | Fastenal | glue for rubber strips | 3-5 days |
| M35 | 4 | M6 x 1mm, 22mm socket head | - | Fastenal | Air cannister bolts | 3-5 days |
| M36 | 2 | 10-32, 7/8” socket head | - | Fastenal | Manifold bolt | 3-5 days |
| M37 | 4 | M4 x 0.5mm, 14mm, 18-8 socket head | - | Fastenal | Three nozzle holder bolts | 3-5 days |
| M38 | 4 | M4 18-8 stainless steel washer | - | Fastenal | Washers for nozzle holder bolts | 3-5 days |
| M39 | 1 | Teflon Tape | - | Fastenal | To prevent leaks from fittings | 3-5 days |
| M40 | 1 | Calibration Stand | - | Metals r us | 4"x2"x2" aluminum | 1 day |
| M41 | 6 | M3 x 0.5 mm, 22 mm socket head | 91292a801 | Mcmaster Carr | Three nozzle holder mounting and load cell to bracket | 3-5 days |
| M42 | 6 | M3 nuts | 91828A211 | Mcmaster Carr | Three nozzle holder mounting | 3-5 days |
| M43 | 6 | M3 x 0.5mm, 18mm long | 91292A029 | Mcmaster Carr | Mounting Calibration stand | 3-5 days |
| M44 | 6 | M3 x 0.5mm x 10 mm long | 91292a113 | Mcmaster Carr | Load cell nozzle mount bolts | 3-5 days |
| M45 | 6 | M3.5 x 0.6mm x 25mm | 92005A727 | Mcmaster Carr | pressure regulator stand mount | 3-5 days |
| M46 | 2 | Push to connect fitting 1/4" T x 1/4" MTP | - | Hose and Fittings Canada | Fitting for tube A to air tank | 1 day |
| M47 | 2 | Push to connect fitting 1/4" T x 1/8" MTP | - | Hose and Fittings Canada | Fitting for tube a to regulator | 1 day |
| E5 | 1 | Laumas Load Cell | - | Laumus | Load cell | 1-2 weeks |
| E6 | 1 | HX711 | SEN-13879 | Digikey | ADC and load cell amp | 1 week |
| E9 | 1 | Pressure Transducer | 32AH8896 | Newark | Pressure transducer for second stage regulator | 2-3 days |

Table 11‑2: Purchase orders

|  |  |  |
| --- | --- | --- |
| **PO No.** | **Source** | **Total Cost** |
| 1 | Clippard | 343.08 |
| 1 | ANS Gear | 348.80 |
| 1 | Grainger Canada | 50.23 |
| 1 | Hose and Fittings Canada | 4.10 |
| 2 | Fastenal | 38.04 |
| 2 | Laumas | 217.66 |
| 2 | Metals-R-Us | 99.80 |
| 3 | Digikey | 24.77 |
| 3 | McMaster Carr | 154.23 |
| 3 | Metals-R-Us | 24.59 |
| 3 | Newark | 171.19 |
| 4 | Grainger Canada | 12.36 |
| 4 | Hose and Fittings Canada | 18.51 |
| 5 | Amazon | 74.34 |
|  | **Total** | **1581.70** |
|  | **Remaining Budget** | **418.30** |

## Cost Benefit Analysis

Reviewing the cost of the project and comparing it to the information the ISL received from the experiments, the value of the results has exceeded the expectation from the ISL. Thrust force characterization of all available sizes for the Crealtiy Ender 3 nozzles has been completed and provided to the ISL. In addition to the results gathered from the experiment, the ISL now has access to a complete second pneumatic system should any of their parts fail, along with airtight fittings for all components. A current issue on the RSS is leaking from the tube connecting the air supply tank to the second stage regulator due to low quality compression fittings. The capstone team explored options to prevent this leaking and were successful with implementing locally sourced push to connect fittings. A second set of fittings was purchased and provided to the ISL for implementation on the RSS. Another issue on the RSS is the use of an analog pressure gage mounted on the second stage regulator to determine the inlet supply pressure. The analog gage is unable to provide accurate pressure readings to within +/- 1 psi. From the physical apparatus, the ISL will now have access to a pressure transducer and associated control code to live readout inlet supply pressure data to +/- 1 psi. Overall, the ISL team was pleased with the results provided by the capstone team and will be implementing the recommendations provided to the next version of the RSS.

# Conclusion

A variety of designs were considered to measure thrust force as shown in the classification tree in Figure 5‑1. Through partaking in ideation internally and with stakeholders, performing concept screening and scoring, the bending load cell design was determined to be the most practical solution. After selecting a design, two prototypes were established.

Project development began with designing a CFD model of the current thruster nozzles. The original model was developed in SolidWorks and used to obtain the expected output velocity and mass flow rates. From these values, thrust force can be calculated. To use better equation solvers and obtain a higher accuracy, CFD models for nozzle diameters ranging from 0.2 mm to 1.0 mm were developed in Ansys Fluent, which utilizes Reynold’s Averaged Navier-Stokes mass and momentum continuity equations to solve the flow in comparison to the generalized Navier-Stokes equation used by SolidWorks. Despite better solver than SolidWorks, Ansys Fluent lacked in accuracy of the results due to student license limitations. Owing to the small outflow area of the Creality Ender 3 nozzles, boundary layers around the neck area required finer meshes due to which result convergence was not possible and instead a finite number of iterations were sampled over a certain flow time to obtain results.

The results generated by these simulations were within an error margin of 20% with respect to the experimental results, with the CFD models consistently over-predicting the steady state thrust force. Hence, the experimental results were successfully validated by the CFD model. The discrepancies in the results can be conferred to be a result of higher viscosities within converging turbulent boundary layers formations and lack of solver models for transition flow conditions.

In addition to a CFD model, a CAD model was also developed. The first iteration of a CAD model prototype of the bending load cell was developed and is displayed in figure 7.1. This model includes all the necessary hardware to operate the system with all parts being listed in table 7.1. The second iteration was updated to increase CAD model accuracy, include additional pneumatic components such as the second stage regulator, and include mounting holes to bolt custom components to the aluminum base. The final design is modelled in CAD and includes all pneumatics, mounting solutions, hardware, and sensors necessary to meet the project requirements. Air is supplied from the air cannister to the second stage regulator. The second stage regulator outputs the regulated pressurized air to an electronic pressure transducer, which sends an analog signal to the microcontroller to record the pressure data, and supplies air to the manifold. The manifold supplies pressurized air to the solenoid valves which are actuated by the microcontroller. The solenoid valves are connected to the thruster nozzles using various custom machined couplings. One thruster nozzle is placed on top of a bending load cell to measure the output thrust during experiment. There are three other nozzles which can also be actuated to determine the effects of simultaneous thruster actuation on the output thrust force. The load cell output signal is fed into a voltage amplifier and ADC board, then the digital signal is sent to the microcontroller for recording. The final design includes a calibration stand to mount the load cell horizontally so that known forces can applied and the calibration values can be determined based on the output voltages.

Having a finalized CAD prototype and selection of electronic components, software development began while waiting for parts to arrive. The final iteration of the software design yielded an Arduino program responsible for the capture of transducer data, and formatting of the data into CSV format. A Python program was built to save the data, process the raw signals into their appropriate measurement units, and save the data as a plot and processed CSV file. This system allows the stakeholders to immediately analyze data and make modifications to the experiment without having to perform any post processing work.

The physical apparatus was built concurrently with finalizing the electrical code to control the apparatus. With assistance from the Dalhousie technicians, all aluminum parts were machined to be mounted on the aluminum base plate. Having all parts, including the electronic circuit boards, reduced the efforts of transportation of the apparatus and the risk of damaging the transducers.Figure 9‑1, Figure 9‑2, and Figure 9‑3 display the final apparatus used to measure and record thrust output from the RSS nozzles with engineering drawings for custom machined parts included in appendix C. Upon completion of assembling the final physical apparatus and electrical control system, a user operation manual was developed to provide instructions for the ISL after the capstone team has completed project handover. This manual is included in appendix D. Lastly, prior to operation of the apparatus, a failure mode and effects analysis were performed to identify potential risks associated with the apparatus. This assessment is included in appendix E.

The first experiment performed with the apparatus was examining the effects of nozzle outlet diameter on thrust force. Crealtity Ender 3 nozzles with outlet diameters ranging from 0.2 – 1.0mm were used throughout this experiment and were subjected to an inlet supply pressure of 60 psi. Completing the experiment and processing the data, the thrust force was observed to reach a maximum using the 0.6 mm outlet diameter while following a parabolic shape with increasing outlet diameter.

The second experiment performed assessed the effects of actuating more than one nozzle on output thrust force. From this experiment, the thrust force was observed to slightly decrease when actuating four nozzles compared to one. This is due to a decrease in supply pressure per nozzle when more than one solenoid valve is actuated.

The last experiment conducted analyzed the effects of inlet supply pressure on output thrust force. Thrust force is directly proportional to inlet pressure, therefore it was expected that the thrust force would increase as supply pressure increased. This hypothesis was confirmed upon completing the experiment where the thrust force increased each increment of 10 psi reaching a maximum at 100 psi.

Completing the experiments allowed the capstone team to come to several conclusions for the Creality nozzles. While the 0.6mm outlet diameter nozzle with a maximum inlet supply pressure of 100 psi will produce the maximum thrust force, the 0.2 mm outlet diameter nozzle should be used to maximize RSS system runtime. However, there are concerns that the 0.2 mm outlet diameter nozzles will not be powerful enough to maneuver the RSS to match the position and speed of space debris.

Reviewing the final costs associated with the project, the project came in at $418 under budget. The final apparatus will benefit the ISL by providing them with a second pneumatic system should any of their parts fail. In addition to the pneumatic system, the ISL will now have access to a higher quality pressure transducer and a 500-gram load cell to perform measurements with. The apparatus was designed to be configurable to test different nozzles, therefore the ISL can use the set up to analyse different nozzle geometries if they wish. The RSS capstone will conclude with project handover on April 13th, 2022.



Recommendations

Upon completion of the RSS capstone several recommendations are proposed to the ISL to improve the RSS and associated models. It is recommended that the CFD simulations are run using an industrial license in ISL. The CFD simulations were limited due to the use of the student version of ANSYS limiting the mesh size. Using an industrial license would allow the simulations to be ran with smaller mesh sizes, increasing the accuracy of the results.

From the results displayed in Figure 10‑1, Figure 10‑2, and Figure 10‑3, to obtain maximum thrust force, the RSS should have the 0.6mm outlet diameter Creality nozzles implemented with a maximum inlet supply pressure. However, if the ISL desires to maximize the RSS runtime, the 0.2mm outlet diameter nozzles should be implemented with a minimum inlet supply pressure. It is recommended that the ISL replace the current nozzles on the RSS with the 0.2 mm nozzles and determine if the nozzles generate adequate thrust force to maneuver the RSS.

The current results demonstrate that the sampling rate of 80 Hz yields satisfactory measurements of the thrust force’s transient response, however, if nozzles of different geometry are to be tested using this apparatus it may be determined that a higher sampling rate is required. In this case, it is recommended that a voltage amplifier/ADC board with a higher sampling rate (such as the aforementioned Tacuna systems amplifier) is purchased and the SparkFun HX711 amplifier/ADC is replaced.

Throughout testing it was determined that the second stage regulator is inconsistent between tests. Without adjusting the pressure regulator, it was shown to vary in output pressure by ±4 psi. As the output pressure is directly related to the thrust force generated by the nozzles, it is recommended that ISL purchases a higher precision second stage regulator to ensure the thrust forces generated on the RSS are consistent.

For the apparatus, it is suggested that a smaller capacity load cell be purchased to replace the current 500g capacity load cell. This load cell was purchased as the project requirements stated that the apparatus must be able to support thrust forces up to 2 N. During testing, the maximum thrust force generated by the Creality nozzles was shown to be 6.81 g. Because the maximum generated thrust force represents only 3.3% of the load cell’s rated capacity, a smaller capacity load cell could be employed without being overloaded. Laumas also offers a 250 g capacity load cell of the same dimensions, therefore, this load cell could be purchased to replace the current load cell on the apparatus without the need for additional mounting components. This replacement would be beneficial as the resolution of the load cell is directly proportional to its rated capacity, therefore, implementing the lower capacity load cell from Laumas would be a simple hardware replacement and decrease the resolution of the load cell by 50%, yielding finer thrust measurement results. It should be noted that the maximum capacity of the load cell should be utilized as the maximum measurable thrust. The mass of the thruster-nozzle-load-cell coupling, barb fitting, and induces a pre-strain in the load cell. As a result, the maximum strain, and subsequently thrust force, that may be exerted on the load cell is lower than the rated capacity.

On both the apparatus and the RSS, it is recommended to minimize the volume between the solenoid valves and the nozzle. This includes minimizing the tube length, and the nozzle coupling chamber volume. When the thrusters are actuated, it takes a finite amount of time for the cold gas to fill the chamber and for the nozzle inlet pressure to match that of the supplied pressure. This results in a longer transient response. Minimizing this volume would decrease the amount of time for the thrust force to reach steady state, which is beneficial for RSS control. This would also minimize the mass of air used during testing as shorter test durations could be used to measure the steady state response.

On the apparatus, it is recommended to purchase M6 flat O-rings to seal the nozzles threaded into the couplings. All tests conducted within the scope of this project were completed using Teflon tape between the nozzles and couplings to provide an airtight seal. As nozzles were repeatedly removed and inserted to conduct thrust tests using different diameter nozzles, pieces of Teflon tape were repeatedly left inside the couplings. Because the couplings are of small diameters and thread sizes, these Teflon debris are difficult to remove. If a large buildup of debris were to occur within the coupling, this could reduce the output thrust force of the nozzle and skew experimental results. Flat O-rings are recommended as a replacement sealant to ensure this buildup cannot occur and experimental results will remain consistent throughout use of the apparatus. If O-rings are not desired, it is alternatively suggested a cleaning regimen is implemented where all fittings are removed from the couplings and the internal chamber is thoroughly cleared of Teflon debris.

Finally, it is recommended to explore the use of different nozzle geometries, specifically converging-diverging nozzle designs. The RSS currently uses converging nozzles designed Creality 3D printers. These nozzles are not designed to generate maximum amounts of thrust force and are therefore of sub-optimal design. The standard nozzle design for cold-gas thrusters is converging-diverging nozzles as they achieve larger thrust forces per mass of cold-gas expelled from the vehicle [17]. It is recommended that ISL explores converging-diverging nozzle designs and tests these nozzles using the thrust measurement apparatus. This would allow the ISL team to implement the nozzle design that produces maximum thrust force per mass of air expelled on the RSS, increasing its runtime.

# Project Handover

Project handover from the 2021/2022 RSS capstone team to the ISL is scheduled for April 13th. On this day, a demonstration will be performed for members of the ISL to ensure the user manual is adequately detailed to perform the experiment without the help from the capstone team. In addition to the demonstration and handover of the physical apparatus, all related control software, CFD models, engineering drawings, and relevant documentation will be uploaded on a USB for delivery to the ISL team.

# References

|  |  |
| --- | --- |
| [1] | J. O'Callaghan, "What is space junk and why is it a problem?," [Online]. Available: https://www.nhm.ac.uk/discover/what-is-space-junk-and-why-is-it-a-problem.html. [Accessed 1 12 2021]. |
| [2] | N. Mohanta, "How many satellites are orbiting the Earth in 2021," 28 5 2021. [Online]. Available: https://www.geospatialworld.net/blogs/how-many-satellites-are-orbiting-the-earth-in-2021/. [Accessed 1 12 2021]. |
| [3] | National Aeronautics and Space Administraion, "Space Debris," [Online]. Available: https://www.nasa.gov/centers/hq/library/find/bibliographies/space\_debris. [Accessed 1 12 2021]. |
| [4] | E. Vautour, "RSS Design Review Document," Halifax, 2021. |
| [5] | G. Gugliotta, "Earth, Clean Up Your Trash!," Smithsonian Magazine, 2017. |
| [6] | E. Vautour, "RSS Propulsion Capstone - Graduate Student User Requiremenrs," Halifax, 2021. |
| [7] | C. Johnston, "Concept Generation," Halifax, 2021. |
| [8] | J. E. Polk, "Recommended Practices in Thrust Measurements," International Electric Propulsion Conference, 2013. |
| [9] | D. Thakur, "What is feasibility study? Types of feasibility. Feasibility study processs.," [Online]. Available: https://ecomputernotes.com/software-engineering/feasibilitystudy. [Accessed 07 11 2021]. |
| [10] | N. Abdullah, F. Zaraini, A. Ammar and T. Wahida Ku Chik, "Load Cell Application in Rocket Thrust Measurement System," *Materials Science and Engineering,* vol. 225, pp. 437-441, 2012. |
| [11] | "McMaster-Carr," McMaster-Carr, [Online]. Available: McMaster-Carr. [Accessed 06 04 2022]. |
| [12] | Renesas, "RA2A1 Weight Measurement Example Using a Load Cell," Renesas, 2020. [Online]. Available: https://www.renesas.com/us/en/document/apn/ra2a1-weight-measurement-example-using-load-cell?language=en. |
| [13] | D. Evans, "HX711 Noise Free Resolution Tests," Arduino, 2015. [Online]. Available: https://forum.arduino.cc/t/hx711-noise-free-resolution-tests/351959. |
| [14] | D. Evans, "Calculating Noise Free Resolution of ADC," Arduino, [Online]. Available: https://forum.arduino.cc/t/calculating-noise-free-resolution-of-adc/962746/4. |
| [15] | Morehouse, "Understanding the Calibration Coefficients for Load Cells and Other Force-Measuring Devices," Morehouse, 2020. [Online]. Available: https://mhforce.com/understanding-the-calibration-coefficients-for-load-cells-and-other-force-measuring-devices/. |
| [16] | T. Brusse-Gendre, "Failure Modes & Effects Analysis," 2001. |
| [17] | A. D. Akshay Reddy Tummala, "An Overview of Cube-Satellite Propulsion Technologies and Trends," MDPI, 2017. |
| [18] | C. Johnston, "Concept Evaluation," Halifax, 2021. |
| [19] | Y. Pan and L. Swan, "Lab #2: Bending Beam Stress," Dalhousie University. |
| [20] | S. Al-Mutlaq, "Getting Started with Load Cells," SparkFun, [Online]. Available: https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all. |
| [21] | eCourses, "Strain Element Mechanics Theory," eCourses, [Online]. Available: https://www.ecourses.ou.edu/cgi-bin/eBook.cgi?doc=session\_blank&topic=me&chap\_sec=08.1&page=theory. |
| [22] | Omega, "Practical Strain Gauge Measurements," [Online]. Available: https://www.omega.co.uk/techref/pdf/StrainGage\_Measurement.pdf. |
| [23] | Omega, "Strain Gauge Installation," [Online]. Available: https://www.omega.co.uk/techref/pdf/Strain-gauge-application-info/how-to-position-strain-gauges.pdf. |
| [24] | A. Kalamkarov, "Chapter 1 - Loading Stresses Strains," [Online]. Available: https://dal.brightspace.com/d2l/le/content/186178/viewContent/2607916/View. |

Project Requirements

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Requirement** | **Rationale** | **Status** | **Modification** | **Verification** |
| 1 | The apparatus shall support the testing of at least two thrusters. | Test multiple thrusters to determine if there is a loss in pressure when actuating more than one thruster. | Complete | Recording data from one thruster, testing up to 4 at one time. | The testing apparatus successfully supports four thrusters while recording output data from one. |
| 2 | The apparatus shall provide thrust force measurements ranging from 0 to a maximum of 2 Newtons from at least one thruster recording over a period up to 10 seconds. | Analyze thrust response when multiple thrusters are being actuated and enable characterization of transient and steady-state response of thruster. | Complete |  | A HX711 was implemented to amplify the output voltage from the load cell to allow the Arduino to read the signal. |
| 3 | The apparatus shall be capable of measuring transient and steady-state thrust force responses up to 2 N. | Design and construct thrust measuring apparatus that is proportional to the expected thrust output. | Complete |  | The selected load cell has a maximum capacity of 5N. |
| 4 | The apparatus should measure thrust on a resolution of 0.01 N. | Want to be able to measure small thrust forces. | Complete |  | The selected load cell has a resolution of 0.005 N. |
| 5 | The apparatus must be capable of measuring both the system storage pressure (0 to 3000 psi) and thruster supply pressure (0 - 110 psi). | To determine the rate of pressure loss during thruster actuation. | Complete |  | Analog gages on air tank reads supply pressure of 0 - 3000 psi, pressure transducer on second stage regulator reads and records pressure of 0 - 200 psi. |
| 6 | The apparatus must allow the operator to vary the supply pressure between tests from 0 - 110 psi. | To determine the effects of supply pressure vs thrust force. | Complete |  | An adjustable second stage regulator was implemented. |
| 7 | The apparatus must be configurable to test different nozzle outlet diameters from 0.2 mm to 1.0 mm. | To determine the effects of different nozzle shapes and sizes on required storage pressure and output thrust force. | Complete |  | The nozzle holder is compatible with M6 fittings to test Creality Ender 3 nozzles with OD's ranging from 0.2-1.0mm |
| 8 | The project budget shall not exceed $2000 CAD. | To ensure the project costs due not exceed the predefined budget. | Complete |  | Total project cost was $1582 |
| 9 | Generate a CFD model that will validate the thruster testing apparatus with a percent error less than 10%. | To allow for validation between experimental and simulated test results. | Complete |  | A CFD model was generated using ANSYS that provided data in agreement with the experimental results  (within 0.3-0.7 grams) |
| 10 | The CFD must simulate the testing of various nozzle outlet diameters from 0.2mm to 1.0mm. | To test nozzle designs out of the project budget. | Complete |  | CFD simulations were performed for 5 out of the 7 nozzle sizes available |
| 11 | The CFD must simulate the testing of eight thrusters with force (0 - 2N), flow (0 - 400m/s) and pressure (0-110 psi) outputs. | To determine simulated results of all thrusters actuating at the same time. | Complete | Due to length of simulation time and only having access to ANSYS student, only one nozzle will be modelled | Only one nozzle was modelled due to restrictions of student license capabilities |
| 12 | The apparatus shall not fail under applied thrust loads of 2 N per nozzle. | Each nozzle will experience up to 5 N of thrust. | Complete |  | Due to the thrust force being small, this was not a concern, |
| 13 | The apparatus shall not fail when subjected to supplied air at 110 psi. | The solenoid valves have a maximum pressure rating of 110 psi, therefore the apparatus will not experience pressures above this value. | Complete |  | All parts sourced were confirmed to be able to operate with pressures applied to them. 300 psi for fittings and tubing prior to second stage regulator. 110 psi for all parts after second stage regulator. |

Code

**RSS\_RunExperiment**

# SUMMARY:

# This Python script reads transducer data from an Arduino's serial output

# and reads it into a CSV file. It then converts the raw transducer data

# into the proper units and saves it to a new CSV file, with an accompanying

# plot. Parameters to open the proper serial port, CSV file names, transducer

# conversion variables are set in 'if \_\_name\_\_ == "\_\_main\_\_"' section.

# SOURCE:

# Written by: Anthony Newton, an768460@dal.ca

# Project: RSS Thruster Testing, 2021-2022

from serial.tools.list\_ports import comports

import serial

import time

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.signal import savgol\_filter

from datetime import datetime

def usrOpenSerial(selectBaud, selectTimeout):

    # View open serial ports

    print('------------------------------------------------------------------')

    print('The following serial ports are open: ')

    portList = [p.device for p in comports()]

    for onePort in portList:

        print(str(onePort))

    # User selects arduino port

    resume = 0

    while (resume == 0):

        selectPort = input("Select Arduino serial port (ex.'/dev/ttyS0'): ")

        for i in range(0, len(portList)):

            if (portList[i] == str(selectPort)):

                print('Port ', portList[i], ' selected.')

                selectPort = portList[i]

                resume = 1

        if (resume == 0):

            print('No matching serial port detected, try again')

    print('------------------------------------------------------------------')

    ser.baudrate = selectBaud

    ser.timeout = selectTimeout

    ser.port = selectPort

    ser.open()

    ser.reset\_input\_buffer

    time.sleep(5)

def readSerialMonitor():

    time.sleep(2)

    if ser.in\_waiting:

        inByte = ser.readline().strip()

        print(inByte.decode('utf-8'))

    else:

        print('Error, nothing returned.')

        print('Press the reset button on Arduino and try again.')

        print('If further trouble is encountered, check HX711 wiring')

        quit()

def talkToArduino(usrInput, command):

    resume = 'false'

    while (resume == 'false'):

        if(usrInput == 'n'):

            exit()

        if command == 'SOLENOIDS':

            print('Inputing amount of solenoids to actuate...')

            ser.write(command.encode('utf-8'))

            time.sleep(4)

            ser.write(usrInput.encode('utf-8'))

            readSerialMonitor()

        if command == 'AIRON':

            print('Inputting time to actuate...')

            ser.write(command.encode('utf-8'))

            time.sleep(4)

            ser.write(usrInput.encode('utf-8'))

            readSerialMonitor()

        if command == 'BEFOREAIR':

            print('Inputing time to read before actuation...')

            ser.write(command.encode('utf-8'))

            time.sleep(4)

            ser.write(usrInput.encode('utf-8'))

            readSerialMonitor()

        if command == 'AFTERAIR':

            print('Inputing time to read after actuation...')

            ser.write(command.encode('utf-8'))

            time.sleep(4)

            ser.write(usrInput.encode('utf-8'))

            readSerialMonitor()

        if command == 'TAREGO':

            print('Command sent...')

            ser.write(command.encode('utf-8'))

            time.sleep(4)

            readSerialMonitor()

            return

        ser.reset\_input\_buffer

        resume = 'true'

def usrCommands(numThruster):

    # Takes user inputs to send to Arduino

    command = 'SOLENOIDS'

    talkToArduino(numThruster, command)

    # Take user input

    usrInput = input('Enter time (ms) to actuate thrusters for (ex. "1000"): ')

    # Indicate what variable user input corresponds to on Arduino

    command = 'AIRON'

    # Send user input and what variable it corresponds to to Arduino

    talkToArduino(usrInput, command)

    usrInput = input(

        'Enter time (ms) to read transducer data before actuating (ex. "1000"): ')

    command = 'BEFOREAIR'

    talkToArduino(usrInput, command)

    usrInput = input(

        'Enter time (ms) to read transducer data after actuating (ex. "1000"): ')

    command = 'AFTERAIR'

    talkToArduino(usrInput, command)

    usrInput = input('Start experiment (y/n): ')

    command = 'TAREGO'

    talkToArduino(usrInput, command)

def serialToCSV(rawDataFileName):

    # Reads data from serial monitor to CSV file

    file = open(rawDataFileName, "w+")

    resume = 0  # true/false condition to record data

    while (resume == 0):

        if ser.in\_waiting:

            incByte = ser.readline().strip()  # Read data from serial as byte format

            # Converts byte from serial monitor to string format

            incByte = incByte.decode('utf-8')

            incByte = incByte.strip()  # Strips string of \r\n characters that Arduino prints

            # print(incByte) # Uncomment this line to see if data is being read

            if (incByte == 'CSVSTART'):

                resume = 1

    while True:

        if ser.in\_waiting:

            incByte = ser.readline().strip()  # Read data from serial as byte format

            # Converts byte from serial monitor to string format

            incByte = incByte.decode('utf-8')

            incByte = incByte.strip()  # Strips string of \r\n characters that Arduino prints

            # print(incByte) # Uncomment this line to see if data is being read

            if (incByte == 'CSVEND'):

                print('Data capture complete.')

                break

            file.write(incByte + "\n")

    file.close()  # Close CSV file

def processData(aCal, bCal, bitCount, maxP, minP, minV, maxV, Vs, rawDataFileName,

                processedDataFileName, processedDataPlotName, thrusterDia, numThruster):

    # PARSE RAW DATA INTO DATAFRAME:

    # Open CSV file containing raw data and generate dataframe.

    # Match the header names defined in Arduino RSS\_LoadCell\_Read script.

    df = pd.read\_csv(rawDataFileName, usecols=['RawLoadReading',

                                               'RawPressureReading', 'Time', 'AirOn', 'TimeStarted',

                                               'TimeEnded'], low\_memory=False)

    rawLoad = df['RawLoadReading'].to\_numpy()

    rawTime = df['Time'].to\_numpy()

    rawPressure = df['RawPressureReading'].to\_numpy()

    AirOn = df['AirOn'].to\_numpy()

    TimeStarted = df['TimeStarted'].to\_numpy()

    TimeEnded = df['TimeEnded'].to\_numpy()

    # Lists which contain processed data

    load = np.zeros(len(rawLoad)-1)

    pressure = np.zeros(len(rawLoad)-1)

    time = np.zeros(len(rawLoad)-1)

    # CONVERT RAW DATA TO PROPER UNITS:

    for i in range(len(rawTime)):

        # SAVE EXPERIMENT SETUP DATA:

        # These parameters are saved by Arduino and are entered into

        # serial monitor at the very end, and are captured here.

        if(i == (len(rawTime)-1)):

            AirOn = AirOn[i]

            TimeStarted = TimeStarted[i] - rawTime[0]

            TimeEnded = TimeEnded[i] - rawTime[0]

            break

        # CONVERT LOAD CELL DATA:

        # Convert raw load cell reading (in counts)

        # to a reading in grams using linear callibration curve fitting

        load[i] = aCal \* rawLoad[i] + bCal

        # CONVERT PRESSURE DATA:

        # Convert raw pressure value reading (in counts) to a voltage value (V)

        pressure[i] = Vs \* rawPressure[i] / (bitCount - 1)

        # Convert pressure voltage reading (in V) to pressure reading (PSI),

        # using linear interpolation

        pressure[i] = (pressure[i] - minV)\*(maxP - minP)/(maxV - minV) + minP

        # CONVERT TIME DATA:

        # Convert arduino clocktime in millis

        # to time in millis which starts at 0

        if(i > 0):

            time[i] = rawTime[i] - rawTime[i-1] + time[i-1]

    maxLoad = np.nanmax(load)

    # OPTIONAL: APPLY SMOOTHING TO DATA

    #pressure\_smoothed = savgol\_filter(pressure, 7, 2)

    # SAVE PROCESSED DATA INTO NEW CSV FILE:

    # Saves processed data as dictionary

    processedData = {'Load\_g': load,

                     'Pressure\_PSI': pressure, 'time\_millis': time}

    # Saves dictionary into pandas dataframe, to save as new CSV

    dfProcessed = pd.DataFrame(processedData)

    # Saves dataframe as CSV

    dfProcessed.to\_csv(processedDataFileName)

    # MASK POINTS LOAD CELL DATA WHERE ELEMENTS ARE EQUAL TO NAN

    loadMask = np.isfinite(load)

    # PLOT PROCESSED DATA:

    fig, axs = plt.subplots(2, figsize=(15, 15))

    # fig.suptitle(f'Thruster Test on {strTime}: {AirOn} ms of Thrust \n \

    #    {numThruster} Thrusters, {thrusterDia} mm Outlet Diameter', fontsize=16 )

    fig.suptitle(f'{numThruster} Thrusters, {thrusterDia} mm Outlet Diameter\n \

        Maximum Thrust: {maxLoad:.4f} g', fontsize=16)

    axs[0].plot(time[loadMask], load[loadMask])

    # Plot time thruster started

    axs[0].axvline(x=TimeStarted, color='k', linestyle='--')

    # Plot time thruster ended

    axs[0].axvline(x=TimeEnded, color='k', linestyle='--')

    axs[0].legend(['Thrust', 'Solenoid On/Off'])

    axs[0].set\_title('Thrust (g) versus Time (ms)')

    axs[0].set(xlabel='Time (ms)', ylabel='Grams of Force (g)')

    axs[0].grid()

    axs[1].plot(time, pressure, color='b', linestyle=':')

    # Comment this out if no smoothing applied

    #axs[1].plot(time, pressure\_smoothed, color='r',)

    axs[1].axvline(x=TimeStarted, color='k', linestyle='--')

    axs[1].axvline(x=TimeEnded, color='k', linestyle='--')

    # axs[1].legend(['Pressure', 'Smoothed Pressure',

    #              'Solenoid On', 'Solenoid Off'])

    axs[1].legend(['Pressure', 'Solenoid On/Off'])

    axs[1].set\_title('Gauge Pressure (PSI) versus Time (ms)')

    axs[1].set(xlabel='Time (ms)', ylabel='Gauge Pressure (PSI)')

    axs[1].grid()

    fig.savefig(processedDataPlotName)

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    # SET SERIAL MONITOR SPECIFICATIONS AND INITIALIZE:

    # Ensure baud rate matches baud rate set in Arduino code

    selectBaud = 115200

    selectTimeout = 60

    ser = serial.Serial()  # Initialize serial ports

    usrOpenSerial(selectBaud, selectTimeout)

    # SET LOAD CELL PARAMETERS:

    # Linear fit calibration values, determined

    # from RSS\_LoadCell\_Callibration\_Order2

    calDate = '2022-03-23'  # Date calallibration values were last entered

    aCal = 0.000262  # slope of linear calibration fit

    bCal = 0.008974  # b point of linear calibration fit

    print('The following linear calibration values will be used:')

    print(f'a = {aCal}, b = {bCal}')

    print(f'They were entered on {calDate}.')

    # SET THRUSTER PARAMETERS

    print('------------------------------------------------------------------')

    numThruster = input(

        "Enter number of solenoids/thrusters to actuate (ex. '1-4'): ")

    thrusterDia = input("Enter diameter of thruster outlet (mm) (ex. '0.4'): ")

    # SET PRESSURE TRANSDUCER PARAMETERS:

    bitCount = 1024  # Maximum resolution of Arduino ADC

    maxPressure = 200  # Maximum pressure of pressure transducer (PSI)

    minPressure = 0  # Minimum pressure of pressure transducer (PSI)

    minPressureVoltage = 0.5  # Minimum output of pressure transducer (V)

    maxPressureVoltage = 4.5  # Maximum output of pressure transducer (V)

    pressureSupplyVoltage = 5.0  # pressure transducer supply voltage (V)

    # SPECIFICY CSV FILE NAME WHICH CONTAINTS RAW AND PROCESSED DATA:

    # The script will append the time of file save to the processed data file

    rawDataFileName = f'RawThrusterData'  # dont add .csv

    processedDataFileName = f'ProcessedThrusterData'  # dont add .csv'

    processedDataPlotName = f'ProcessedThrusterPlot'  # dont add .png

    # Get time at which CSV is generated to add to title of CSV

    strTime = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")

    # Get location of files to save to

    cwd = os.path.dirname(os.path.abspath(\_\_file\_\_))

    # Update file names

    rawDataFileName = f'{cwd}\Plots\{rawDataFileName}\_{numThruster}sol\_D{thrusterDia}\_{strTime}.csv'

    processedDataFileName = f'{cwd}\Plots\{processedDataFileName}\_{numThruster}sol\_D{thrusterDia}\_{strTime}.csv'

    processedDataPlotName = f'{cwd}\Plots\{processedDataPlotName}\_{numThruster}sol\_D{thrusterDia}\_{strTime}.png'

    # START PROGRAM AND SERIAL MONITOR:

    usrCommands(numThruster)

    serialToCSV(rawDataFileName)

    processData(aCal, bCal, bitCount, maxPressure, minPressure, minPressureVoltage,

                maxPressureVoltage, pressureSupplyVoltage, rawDataFileName,

                processedDataFileName, processedDataPlotName, thrusterDia, numThruster)

    # CLOSE SERIAL MONITOR:

    ser.reset\_output\_buffer()

    ser.close()  # Close serial port so it may be used by other programs

**RSS\_CalibrateAndReadLoadCell**

/\*

Callibrate Load cell and HX711 using Arduino and known masses.

Based off of Olkal HX711\_ADC library and examples.

Written by: Anthony Newton, Nolan Cain

Project: RSS Capstone 2021-2022

Updated: 2022-03-24

\*/

// Load Cell

#include <HX711\_ADC.h>

#include <curveFitting.h> //Used to generate linear interpolation curve for callibration

const int HX711\_dat = A4; // HX711 dat pin, refered to as HX711\_dout in HX711\_ADC library

const int HX711\_clk = A5; // HX711 clk pin, refered to as HX711\_sck in HX711\_ADC library

HX711\_ADC LoadCell(HX711\_dat, HX711\_clk);

unsigned long t = 0;

float a = 0.000262; // Calibration value a (slope)

float b = 0.008974; // Calibration value b (y-axis intercept)

// Pressure Transducer

const int pressureInputPin = A6;

const float maxPressure = 200; // Max pressure of transducer (PSI)

const float minV = 0.5; // Min output voltage of transducer (PSI)

const float maxV = 4.5; // Max output voltage of transducer (PSI)

const float supplyV = 5.0; // Supply voltage of Arduino

const float bitSize = 1023.0; // Int resolution of Arduino

const int avgCount = 64; // Amount of samples to average

// Solenoid

void setup()

{

Serial.begin(57600);

Serial.println();

DDRD = B11111110;

LoadCell.begin();

unsigned int stabilizingtime = 2000; //Stabilizing time to improve precision, minimum 2 seconds

bool \_tare = true; //set this to false if you don't want tare to be performed in the next step

LoadCell.start(stabilizingtime, \_tare);

// Check whether HX711 is connected properly

if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {

Serial.println("Timeout, check HX711 wiring and pin designations");

while (1);

}

else {

// Not using default calibration method, keep to 1.0

LoadCell.setCalFactor(1.0);

Serial.println("Loading complete.");

}

while (!LoadCell.update());

}

void loop()

{

Serial.println("To calibrate load cell, enter 'c' in serial monitor.");

Serial.println("To read load/pressure or actuate solenoids, enter 'r' in serial monitor.");

Serial.println(" - Uses hardcoded calibration values for load cell reading.");

while (1)

{

if (Serial.available() > 0)

{

char inByte = Serial.read(); // read user entry in serial

if (inByte == 'c')

{

calibrate();

break;

}

if (inByte == 'r')

{

readTransducer();

break;

}

}

}

}

void readTransducer()

{

// Load Cell

LoadCell.setSamplesInUse(1);

static boolean newDataReady = 0;

const int serialPrintInterval = 0; // increase value to slow down serial print activity

// Pressure Transducer

int rawPressVal = 0; // Value to read raw sample into

float psi = 0;

float V = 0;

const int avgNum = 80; // amount of times to average transducer data

tareLoad();

Serial.println("To return to options, enter 'e' in serial monitor.");

Serial.println("Optionally enter number of solenoids/thrusters to actuate for 2 sec (ex. '1-4'): ");

Serial.println("Mass (g), Pressure (PSI), Time (millis)");

while (1)

{

int counter = 1;

float avgLoad = 0;

while (counter <= avgNum)

{

if (LoadCell.update()) newDataReady = true; // check for new data/start next conversion

if (newDataReady)

{

if (millis() > t + serialPrintInterval)

{

float i = LoadCell.getData();

t = millis();

newDataReady = false;

avgLoad = avgLoad + i;

rawPressVal += analogRead(pressureInputPin); // Read in samples

counter = counter + 1;

}

}

}

rawPressVal = rawPressVal / counter;

V = (supplyV \* rawPressVal) / bitSize;

psi = mapFloat(V, minV, maxV, 0, maxPressure);

// calculate average load cell value

avgLoad = avgLoad / ((float)counter);

avgLoad = a \* avgLoad + b;

Serial.print(avgLoad);

Serial.print(" , ");

Serial.print(psi);

Serial.print(" , ");

Serial.println(t);

// receive command from serial terminal

if (Serial.available() > 0)

{

char inByte = Serial.read();

if (inByte == 't')

{

tareLoad();

}

if (inByte == '1' || inByte == '2' || inByte == '3'|| inByte == '4')

{

solenoids(inByte);

}

else if (inByte == 'e')

{

return;

}

}

}

}

void calibrate()

{

Serial.println("Remove any load applied to the load cell."); // Taring zeros the scale

tareLoad();

boolean \_resume = false;

tareLoad();

Serial.println("Enter number of callibration trials to carry out, a minimum of 2 is required (ex. '2'): ");

int n\_mass; // Determine number of masses/trials used to calibrate the load cell

\_resume = false;

while (\_resume == false)

{

if (Serial.available() > 0)

{

n\_mass = Serial.parseInt(); // read user entry in serial

\_resume = true;

}

}

Serial.println(n\_mass);

// Read in the known callibration masses from user, and read in actual value read by load cell

// Save values to known\_mass and read\_mass respectively

double known\_mass[n\_mass]; // Create array to store known masses for callibration

double read\_mass[n\_mass]; // Create array to store load cell readings of known masses

double avgRead\_mass = 0.0; // Mass values are averaged for amount avgTrial

int avgTrial = 80; // Amount of times to read in a load cell value for each mass which is then averaged

for (int i = 0; i < n\_mass; i++)

{

\_resume = false;

Serial.println("\*\*\*");

Serial.print("Place known mass ");

Serial.print(i + 1);

Serial.println(" on the scale.");

Serial.println("Then send the known mass value in grams (ex.'100.00') from serial monitor:");

while (\_resume == false)

{

LoadCell.update();

if (Serial.available() > 0)

{

known\_mass[i] = Serial.parseFloat();

if (known\_mass[i] != 0)

{

Serial.print("Known mass is ");

Serial.print(known\_mass[i]);

Serial.println(" (g).");

LoadCell.refreshDataSet(); // refresh the dataset to be sure that the known mass is measured correct

read\_mass[i] = LoadCell.getData();

Serial.print("The load cell reads: ");

Serial.print(read\_mass[i]);

Serial.println(".");

\_resume = true;

}

}

}

}

Serial.println("Reading known masses is complete, determining calibration fit...");

// Calculate calibration value using a linear fit. Order is defined at start of script.

// for a first order system: y = mx + b

int order = 1; // Determines the order of the polynomial curve fit

char buf[100];

double coeffs[order + 1];

int ret = fitCurve(order, sizeof(read\_mass) / sizeof(float), read\_mass, known\_mass, sizeof(coeffs) / sizeof(double), coeffs);

// Print calculated coefficients of calibration curve

if (ret == 0)

{ // Returned value is 0 if no error

uint8\_t c = 'a';

Serial.println("Coefficients are");

for (int i = 0; i < sizeof(coeffs) / sizeof(double); i++)

{

snprintf(buf, 100, "%c=", c++);

Serial.print(buf);

Serial.print(coeffs[i], 6);

Serial.print('\t');

}

}

else

{

Serial.println("An error has been encountered with the calibration curve fitting, please try again");

return;

}

Serial.println();

// This code has to be modified if the order of the calibration fit is increased

a = coeffs[0];

b = coeffs[1];

Serial.println("Please record the calibration values. Enter 'e' in serial monitor to return to options.");

}

void tareLoad()

{

LoadCell.update();

LoadCell.tare();

long \_offset = LoadCell.getTareOffset();

LoadCell.setTareOffset(\_offset);

Serial.println("Tare Completed.");

}

float mapFloat(float x, float in\_min, float in\_max, float out\_min, float out\_max)

{

return (x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min;

}

void solenoids(char numSolnd)

{

// Actuates solenoids 1,2,3,4 (or any combination thereof)

// once the time to read before actuation has ended

if (numSolnd == '1')

{

PORTD = B00100000; // 1 (Load cell solenoid)

}

else if (numSolnd == '2')

{

PORTD = B00110000; // 1,2

}

else if (numSolnd == '3')

{

PORTD = B00111000; // 1,2,3

}

else if (numSolnd == '4')

{

PORTD = B00111100; // 1,2,3,4 (all)

}

else

{

Serial.println("Invalid number of solenoids selected");

Serial.println("Please restart");

return;

}

delay(2000);

PORTD = B00000000;

}

**RSS\_** **RSS\_ControlTransducers:**

/\*

\* Summary:

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* This Arduino program reads load cell and pressure transducer data using Arduino.

\* The program, in conjuction with a Python serial monitor script, takes from

\* the user:

\* - How many solenoids to actuate

\* - How long to actuate the solenoids

\* - How long to read data from the load cells

\* It then actuates the solenoids, reads data from the load cell and pressure

\* transducer, and prints it to serial monitor. The python script saves the data

\* to a CSV file. A second python script converts the CSV data to the desired units.

\* NOTE:

\* If using a laptop battery as source of power for Arduino and load cell,

\* please disconect the laptop from it's charger. This will reduce noise in

\* the load cell signal.

\* The HX711 is set to 80 Hz reading by removing a connection on the board itself.

\* Its default is 10 Hz.

\* Conversion of the load cell signal (counts) and pressure transducer signal (V)

\* to grams and PSI is performed in the python script.

\* The user should determine the load cell calibrations beforehand using the

\* Arduino script, and import the values into the Python unit conversion script.

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* Settings for RSS apparatus:

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* Board -> "Arduino Nano"

\* Processor -> "ATmega168"

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*

\* Arduino -> solenoid output pin mapping:

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* terminal 1 terminal 2 terminal 3

\* [[+][2][10][9]] [[+][3][8][7]] [[+][4][6][5]]

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*

\* Open pins left for DAQ: 1,11,12,13,14

\*

\* Written by: Anthony Newton, Nolan Cain

\* Project: RSS Capstone 2021-2022

\*

\* Source for HX711 library: https://github.com/olkal/HX711\_ADC

\*/

#include <HX711\_ADC.h>

// solenoid control pins (here for circuit wiring, don't need to be executed)

//const int pinSolnd1 = 5; // load cell solenoid

//const int pinSolnd2 = 4; // solenoid 2

//const int pinSolnd3 = 3; // solenoid 3

//const int pinSolnd4 = 2; // solenoid 4

//HX711 pins:

const int HX711\_dat = A4; //HX711 dat pin, refered to as HX711\_dout in HX711\_ADC library

const int HX711\_clk = A5; //HX711 clk pin, refered to as HX711\_sck in HX711\_ADC library

const int pinPressure = A6; // pressure read pin

//constructors:

HX711\_ADC LoadCell(HX711\_dat, HX711\_clk);

//constants and global var

boolean \_resume = false;

unsigned long t = 0;

unsigned long air\_on = 1000; //How long solenoid valve is to be actuated for (s)

unsigned long before\_air = 1000; //Reading data before solenoid is actuated.

unsigned long after\_air = 1000; //Reading data after the solenoid is actuated

unsigned int numSolnd; //How many solenoids to actuate

void setup()

{

// variable used for port manipulation

// initialize pins 2,3,4 and 5 as output for actuating solenoids

DDRD = B1111110;

Serial.begin(115200); delay(10);

// initialize pressure transducer read pin

pinMode(pinPressure,INPUT);

// Load cell initialization

LoadCell.begin();

unsigned long stabilizingtime = 5000;

boolean \_tare = true; //set this to false if you don't want tare to be performed in the next step

LoadCell.start(stabilizingtime, \_tare);

if (LoadCell.getTareTimeoutFlag()) {

Serial.println("Timeout, check MCU>HX711 wiring and pin designations");

while (1);

}

else {

LoadCell.setCalFactor(1.0); // We are not using the libraries default calibration method. Therefore set to 1.

// Succesfully initialized

}

}

void loop()

{

Serial.flush();

\_resume = false;

if(Serial.available() > 0)

{

String inBytes = Serial.readString();

if(inBytes == "AIRON")

{

// Reads from user how long to actuate solenoids

while(\_resume == false) {

if(Serial.available() > 0)

{

air\_on = Serial.parseInt();

Serial.println(air\_on);

Serial.flush();

\_resume = true;

}

}

}

else if(inBytes == "BEFOREAIR")

{

// Reads from user how long to read data before solenoids are actuated

while(\_resume == false)

{

if(Serial.available() > 0)

{

before\_air = Serial.parseInt();

Serial.println(before\_air);

Serial.flush();

\_resume = true;

}

}

}

else if(inBytes == "AFTERAIR")

{

// Reads from user how long to read data after solenoids are actuated

while(\_resume == false)

{

if(Serial.available() > 0)

{

after\_air = Serial.parseInt();

Serial.println(after\_air);

Serial.flush();

\_resume = true;

}

}

}

else if(inBytes == "SOLENOIDS")

{

// Reads from user how many solenoids to actuate

while(\_resume == false)

{

if(Serial.available() > 0)

{

numSolnd = Serial.parseInt();

Serial.println(numSolnd);

Serial.flush();

\_resume = true;

}

}

}

else if(inBytes == "TAREGO")

{

// User commands script to start actuation and transducer reading

Serial.println("Preparing experiment...");

delay(5000);

actuate();

}

}

}

// actuate() actuates the solenoid valves and reads transducer data to serial monitor

void actuate()

{

unsigned long time\_on = 0; //How long the solenoid valve has been on for

unsigned long time\_on\_prev = 0;

unsigned long tStarted = 0; // Records when actuation actually started

unsigned long tEnded = 0; // Records when actuation actually ended

unsigned int sample = 0; // Records transudcer sample number

// Variable used to determine whether data from load cell is ready to read in

static boolean newDataReady = 0;

// Variable to control how much data from load cell is recieved (keep at 0)

const int serialPrintInterval = 0;

// Reset pressure

PORTD = B00111100; // 1,2,3,4 (all)

delay(1000);

PORTD = B00000000;

delay(2000);

Serial.println("CSVSTART"); //Indicates start of data to Python script

Serial.println("Sample,RawLoadReading,RawPressureReading,Time,AirOn,TimeStarted,TimeEnded"); // CSV Column headers

boolean turnedOn = false;

\_resume = false;

time\_on\_prev = millis(); // Reads Arduino clock time to determine when the reading loop has started

while(time\_on < (air\_on + before\_air + after\_air))

{

// Turn solenoids on

if (\_resume == false && time\_on > before\_air)

{

// Actuates solenoids 1,2,3,4 (or any combination thereof)

// once the time to read before actuation has ended

if (numSolnd == 1)

{

tStarted = millis();

PORTD = B00100000; // 1 (Load cell solenoid)

}

else if (numSolnd == 2)

{

tStarted = millis();

PORTD = B00110000; // 1,2

}

else if (numSolnd == 3)

{

tStarted = millis();

PORTD = B00111000; // 1,2,3

}

else if (numSolnd == 4)

{

tStarted = millis();

PORTD = B00111100; // 1,2,3,4 (all)

}

else

{

Serial.println("Invalid number of solenoids selected");

Serial.println("Please restart");

return;

}

turnedOn = true;

\_resume = true;

}

if (time\_on > (before\_air + air\_on) && turnedOn == true)

{

PORTD = B00000000; //Turn off solenoids

tEnded = millis(); //Record actual time the solenoids were turned off

turnedOn = false;

}

if (LoadCell.update())

{

newDataReady = true; // Check if data is ready to be read from load cell

}

if (newDataReady)

{

if (millis() > t + serialPrintInterval)

{

float i = LoadCell.getData(); // Read data from load cell. By default, the moving average data set of 1 sample is used.

float p = analogRead(pinPressure);

sample = sample + 1;

Serial.print(sample);

Serial.print(",");

Serial.print(i);

Serial.print(",");

Serial.print(p);

Serial.print(",");

t = millis();

Serial.println(t);

newDataReady = 0;

}

}

else

{

// comment this block out to match frequency of pressure reading

// to frequency of load cell reading

float p = analogRead(pinPressure);

sample = sample + 1;

Serial.print(sample);

Serial.print(",,");

Serial.print(p);

Serial.print(",");

Serial.println(millis());

}

time\_on = millis() - time\_on\_prev; // update time counter

}

// close all solenoids

PORTD = B00000000; //Ensure solenoids are off

Serial.print(",,,,");

Serial.print(air\_on);

Serial.print(",");

Serial.print(tStarted);

Serial.print(",");

Serial.println(tEnded);

Serial.println("CSVEND"); //Tells the python script to stop reading

}

Engineering Drawings

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram, box and whisker chart

Description automatically generated with medium confidence

Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Diagram, schematic

Description automatically generated

Chart, box and whisker chart

Description automatically generated with medium confidence

Diagram, schematic

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

User Operation Manual

**RSS: Thruster Testing Apparatus Operation Manual**

This report will detail the necessary steps to operate the thruster testing apparatus designed and built by the 2021/2022 capstone team. Software required to use the testing apparatus include Arduino IDE and Python 3 (preferably with an editor such as VS Code). Due to the use of high-pressure air, it is recommended that the operators where safety glasses while the air is being actuated.

**It is important to note that the laptop connected to the Arduino should not be plugged into an external power source while the user is reading data from the load cell. This reduces noise in the load cell readings.**

**Do not unscrew the nozzle from the load cell while it is mounted to the base plate. This can damage the transducer. Please remove the load cell from the base plate and hold the nozzle mount while switching the nozzle.**

**Apparatus:**

* RSS thruster testing apparatus (see last page for parts list)
* Precision masses
  + 2 grams to 400 grams (if available, see below)
* LiPo battery
* Micro-USB cable

**Procedure:**

**1) Pretesting**

Record initial supply pressure of the air cannister.

|  |  |
| --- | --- |
| **Initial Supply pressure (psi)** |  |

Before testing the pneumatics with the air supply, each transducer should be tested and verified as well as the solenoid valves, according to the procedure below.

**Calibration of the load cell**

The load cell must be calibrated to ensure the transducer is operating accurately and as error-free as possible. To calibrate the load cell, a minimum of two masses must be used. Using more masses will increase the accuracy of the calibration, as linear regression is used to fit the load cell signal to the mass values on a linear calibration curve. Optimal calibration would use masses ranging from 0% to 100% of the full scale, increasing in small increments, and then 100% to 0% of the full scale decreasing in the same increments. Since the thruster assembly has associated mass, and to ensure that the maximum capacity of the load cell is not exceeded, the following calibration masses are recommended for a 500 g load cell (adjust as necessary for a smaller or larger load cell).

|  |  |
| --- | --- |
| Calibration Trial | Mass (g) |
| 1 | 2 |
| 2 | 5 |
| 3 | 20 |
| 4 | 50 |
| 5 | 100 |
| 6 | 200 |
| 7 | 400 |
| 8 | 200 |
| 9 | 100 |
| 10 | 50 |
| 11 | 20 |
| 12 | 5 |
| 13 | 2 |

The number of masses used can be adjusted as necessary to decrease the time to calibrate or account for what masses the user has available. What is important is that the full range of the load cell is calibrated.

The load cell would ideally be calibrated prior to each use however it is only necessary if the load cell has been removed from its mount, or if it has been over a year since its last use (at a minimum). The following steps outline the procedure for calibrating the load cell.

1. Mount the load cell to the calibration stand on the base plate
2. Ensure the thruster tube is not touching the base plate and thus imparting a force on the load cell
3. Connect the Arduino to the user’s laptop via micro-USB cable
4. Run RSS\_CalibrateAndReadLoadCell.ino
   1. Enter “c” in Arduino serial monitor to begin calibration
   2. Input number of calibration trials. The example in Table 1 above has 13 trials. At a minimum, 2 trials are required with each trial testing a different mass.
   3. Place masses on load cell
   4. Input known mass value
   5. Record data points (a and b) values generated from RSS\_CalibrateAndReadLoadCell.ino
      1. These will be hard coded into RSS\_RunExperiment.py later
5. Unmount the load cell from the calibration stand
6. Mount the load cell to the load cell bracket in testing orientation

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| A |  |
| B |  |

**Verification of solenoid valves**

In addition to testing the transducers, actuation of each solenoid valve should be performed to ensure functionality prior to adding air to the system. The following steps outline the procedure for testing and verifying the solenoid valves.

1. Plug LiPo battery into solenoid valve circuit board
2. Run RSS\_CalibrateAndReadLoadCell.ino
   1. Enter “r” in Arduino serial monitor
   2. Enter a number 1 – 4 to actuate desire solenoid vales
      1. Actuate all 4
3. Ensure all valves are operational by listening for a clicking sound

|  |  |
| --- | --- |
| **Solenoid Valve** | **Operational (Y/N)** |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |

**Verification of the air supply pressure**

The following steps outline the procedure for testing and verifying the pressure of the air.

1. Run RSS\_CalibrateAndReadLoadCell.ino
   1. Enter “r” in Arduino serial monitor
   2. Record the pressure readout with no supply pressure
2. Turn on air supply
   1. While air supply is on, the pressure transducer script will read out the pressure data continuously
   2. Confirm desired pressure is being supplied to system
   3. Adjust second stage regulator as necessary to achieve desired inlet supply pressure
3. Turn off Air supply
4. Adjust pressure reading according to readout when no air was supplied

|  |  |
| --- | --- |
| **Measuring Instrument** | **Readout (psi)** |
| Pressure Transducer (no supply pressure) |  |
| Pressure Transducer (desired value) |  |

**2) Testing**

The following steps are to be performed upon completion of the pretesting requirements.

1. Open RSS\_RunExperiment.py on computer
   1. Hardcode calibration values A and B recorded from load cell calibration into RSS\_RunExperiment.py (initiated as aCal and bCal in Python script)
   2. Update calDate to reflect when aCal and bCal values were entered
2. Load RSS\_ControlTransducers.ino onto Arduino from computer
3. Run RSS\_RunExperiment.py on computer
   1. Input number of solenoids to actuate
   2. Input duration for data collection prior to actuation of solenoids
   3. Input duration for actuation of solenoids
4. Wait for test to complete
   1. While testing is occurring, the raw data from the load cell is being stored in a csv file in RSS\_RunExperiment.py
   2. A second csv file will read the raw data, process it, and generate a plot of force (grams) versus time (seconds)
5. Record observations
6. Repeat testing as necessary
   1. Press reset button on Arduino
   2. Restart RSS\_RunExperiment.py code on computer
7. Close tank valve for air supply

Observations

|  |
| --- |
|  |

**3) Post Testing**

Record final supply pressure of the air cannister.

|  |  |
| --- | --- |
| **Final Supply pressure (psi)** |  |

The following steps are to be completed after all testing is done.

1. Open all solenoid valves to ensure there is no pressure left in the system
   1. Run RSS\_CalibrateAndReadLoadCell.ino and actuate each valve
   2. Unplug LiPo battery after each valve has been actuated
2. Verify there is no pressure by taking a reading from the pressure transducer
   1. Run RSS\_CalibrateAndReadLoadCell.ino and confirm a zero read out
3. Unplug arduino

**Parts List**

All parts will be mounted on the base plate (except for the LiPo battery).

Diagram, engineering drawing

Description automatically generated

|  |  |
| --- | --- |
| **Components​** | |
| 1​ | Air Supply Tank​ |
| 2​ | Second Stage Regulator​ |
| 3​ | Solenoid Valves​ |
| 4​ | Manifold​ |
| 5​ | Measured Thruster Nozzle Coupling​ |
| 6​ | Load Cell (testing orientation)​ |
| 7​ | Load Cell Calibration Stand​ |
| 7a​ | Load Cell (calibration orientation)​ |
| 8​ | Three Thruster Nozzle Coupling​ |
| 9​ | Circuit Board​ |
| 10​ | Base​ |
| 11​ | Pressure Transducer​ |

Failure Mode and Effects Analysis

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Part** | **Function** | **Failure Mode** | **Failure Effect** | **Severity** | **Potential Causes** | **Prob. of Occurrence** | **Control Method** | **Control Effectiveness** | **RPN** |
| Air tank | Supply compressed air | Decompression/air leak  Fatigue crack | Decompression of supplied air  Projectiles | 10 | Repeated pressurization of the tank (end of life cycle)  Mishandling the tank (dropping/hitting)  Debris build up around outlets | 2 | Handle tank with care  Operate with instructions from manufacturer | 2 | 40 |
| Low pressure regulator | Regulate output to 300 psi | Damaged internal components  Fatigue cracks  Creep  Damaged ports | Subjecting parts to air at 3000 psi that are not rated for that pressure | 10 | Mishandling of the regulator attachment (dropping/hitting)  Debris build up | 2 | Handle regulator with care  Clean ports regularly | 2 | 40 |
| Barb fitting 1 | Connect low pressure regulator to tube A | Damaged threads | Air leak | 4 | Over tightening fitting | 4 | Use a torque wrench appropriate for the fastener | 3 | 48 |
| Tube A | Transfer air from air tank to second stage regulator | Excessive crimping pressure  Repeated flexing | Air leak | 4 | Damage to the core of tube when attaching fittings | 7 | Examine hose/fitting interface before start-up  Replace hosing if fitting must be removed | 3 | 84 |
| Barb fitting 2 | Connect tube A to second stage regulator | Damaged threads | Air leak | 4 | Over tightening fitting | 4 | Use a torque wrench appropriate for the fastener | 3 | 48 |
| Second stage regulator | Reduce air pressure to 60 psi | Damaged internal components  Fatigue cracks  Creep  Damaged ports | Subjecting parts to air at 300 psi that are not rated for that pressure | 10 | Mishandling of the regulator attachment (dropping/hitting)  Debris build up | 2 | Handle regulator with care  Clean ports regularly | 2 | 40 |
| Barb fitting 3 | Connect second stage regulator to tube B | Damaged threads | Air leak | 4 | Over tightening fitting | 4 | Use a torque wrench appropriate for the fastener | 3 | 48 |
| Tube B | Transfer air from second stage regulator to manifold | Excessive crimping pressure  Repeated flexing | Air leak | 4 | Damage to the core of tube when attaching fittings | 7 | Examine hose/fitting interface before start-up  Replace hosing if fitting must be removed | 3 | 84 |
| Barb fitting 4 | Connect tube B to the manifold | Damaged threads | Air leak | 4 | Over tightening fitting | 4 | Use a torque wrench appropriate for the fastener | 3 | 48 |
| Manifold | Passes air through the solenoid valves | Fatigue crack  Flow choking  Uneven flow | Air leak  Uneven air flow output | 8 | Mishandling of the manifold (dropping/hitting)  Debris build up | 2 | Handle manifold with care  Clean ports regularly | 2 | 32 |
| Solenoid valves | Open/close air ways to the nozzles | Misread open/close signal  Coil burn out  Damaged seal  Dirt particles  Power supply | Failure to open/close valve  Shocking/burn hazard | 9 | Cycling rate too fast causing heat buildup and coil burnout  Debris buildup  Incorrect power supply voltage/frequency | 5 | Determine maximum cycle rate to prevent burnout  Clean valves regularly  Determine required input voltage/current from spec sheet | 2 | 90 |
| Barb fitting 5 | Connect solenoid valve air ways to tube C | Damaged threads | Air leak | 4 | Over tightening fitting | 4 | Use a torque wrench appropriate for the fastener | 3 | 48 |
| Tube C | Transfer air from solenoids to nozzles | Excessive crimping pressure  Repeated flexing | Air leak | 4 | Damage to the core of tube when attaching fittings | 7 | Examine hose/fitting interface before start-up  Replace hosing if fitting must be removed | 3 | 84 |
| Barb fitting 6 | Connect tube C to the nozzles | Damaged threads | Air leak | 4 | Over tightening fitting | 4 | Use a torque wrench appropriate for the fastener | 3 | 48 |
| Coupling | Connects barb to nozzle | Damaged threads | Air leak | 4 | Over tightening fitting | 4 | Use a torque wrench appropriate for the fastener | 3 | 48 |
| Nozzle | Outputs air generating thrust | Damaged threads  Fatigue cracking | Uneven thrust output | 7 | Over tightening fitting | 4 | Use a torque wrench appropriate for the fastener | 3 | 84 |

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated