P2C2

Design of a Mechanism for Testing Dynamics of a Rotating Shaft with a Moving Support

Sponsored by Dr. Hagay Bamberger and the University of Michigan
Engineering Research Center for Reconfigurable Manufacturing Systems

Matthew Carpenter, Design Engineer, P2C2
Alexander Cicerone, Design Engineer, P2C2
Lisa Perez, Design Engineer, P2C2
Jessica Port, Design Engineer, P2C2

December 15, 2009

Date: December 15, 2009

To: Dr. Hagay Bamberger, Post-Doctorate, University of Michigan

Cc: Dr. Yoram Koren, Professor, University of Michigan

From: Matthew Carpenter, Design Engineer, P2C2
Alexander Cicerone, Design Engineer, P2C2
Lisa Perez, Design Engineer, P2C2
Jessica Port, Design Engineer, P2C2

Subject: Design of a Mechanism for Testing Dynamics of a Rotating Shaft with a Moving Support

---

Abstract

The goal of this project was to measure the dynamic response – namely, the displacement of a shaft from its static configuration – at the free end of an unbalanced shaft that was clamped and rotated by an existing machine. To accomplish this, the team designed and fabricated a system that was implemented in the vertical mill. This system allowed motion of a simple support along the shaft's length while a sensor monitored the displacement from static configuration. Once the team completed this, they ran the system and generated an estimation of the displacements.

Executive Summary

Dr. Hagay Bamberger of the Engineering Research Center for Reconfigurable Manufacturing Systems contracted P2C2 to design, fabricate, and operate a mechanism to measure the dynamics, specifically, the amplitude of vibration, at the free end of an unbalanced rotating shaft. This mechanism was to have a simple support that traverses along the length of the shaft.

The customer requested that the design have the ability to do the following: (1) be structurally integrated with the mill or lathe of the team's selection, (2) continuously collect data while the shaft support traverses along the shaft, and (3) accurately and precisely measure the shaft dynamics. Additionally, it was to be easily operable, maintainable, manufacturable, low-cost, and completed no later than December 10, 2009.

Upon analysis of the engineering specifications that the team established with the customer, P2C2 decomposed the design into four main functions: (1) rotate shaft, (2) support shaft, (3) traverse support, and (4) measure displacement. For the first function, the team narrowed down a shaft design and selected a particular mill based on the customer's requests. For the next three functions, the team generated concepts and analyzed them, creating an alpha design that consisted of a direct-driven platform and laser triangulation sensors.

Taking material and safety into consideration, the team narrowed down specific components that they would use in their final design. They determined how the mechanical aspects of the design would be integrated through both CAD and full-scale paper layouts in the Fadal mill. Similarly, they considered how the electrical and coding aspects of their design (based on the specifications of the sensors and the motor) would be integrated.

P2C2 created detailed manufacturing, testing, and safety procedures on which they were to base any further work on the design. They procured their parts and performed their machining, assembly, and integration. Upon completion of these tasks, they began their validation of the components of the design and their operation of the design.

Although the team was able to complete component validation, they only obtained a few sets of data from their design operation before the shaft hit resonance 1,500 RPM before its expected resonance point, creating large deflections in the shaft, and ultimately rendering the design unrecoverable. In reaction to this result, P2C2 broke down the failure step-by-step and determined the multiple improvement points necessary to further the project.

The following report discusses the information presented above in greater detail.

Contents

Problem Description

There exist many situations in which knowing the dynamics of a rotating shaft at its free end is useful. For example, automobile geartrains must take into account how much the driver gear will be fluctuating from its initial position to determine placement of the adjacent gears for optimal meshing. Likewise, the system bearings must have a rating that can accommodate the shaft dynamics. Another example is the method used to measure the eccentricity of internal combustion engine cylinders, in which the rotating shaft (positioned inside the cylinder) has a sensor at the free end that measures proximity to the cylinder.

These situations considered, Dr. Bamberger, in cooperation with the Engineering Research Center for Reconfigurable Manufacturing Systems, proposed a mechanism with a traversing support that tests the dynamics at the free end of an unbalanced rotating shaft. P2C2 was responsible for the design, fabrication, and operation of such a mechanism scaled to integrate with a mill of the team's selection. Their design was to include a shaft such that the natural frequency of said shaft was just less than the frequency of the chosen mill and a method of creating an unbalance in said shaft.

Background Information

P2C2 completed background research in three relevant areas: the dynamics of rotating shafts, the forces on a bearing given a force on the shaft, and prior art. This research is presented below.

Dynamics of Rotating Shafts

Although it may appear so, no manufactured shaft is perfectly straight or contains a uniform density throughout. These irregularities often leave a shaft unbalanced, and an outward inertia force during rotation of the shaft results. This force can also be exaggerated by placing a mass on one side of the rotating shaft at its free end in order to obtain easily measurable amplitudes of displacement from the axis of rotation.

The amplitude of the consequent displacement from the axis of rotation, $u(z)$, can be theoretically determined using the elastic stability formulas for cantilevered beams. The rotating shaft is treated as a cantilever beam with the only force acting on it being the outward inertia force created by the mass rotating off center. The underlying equations are outlined below. Although these are the underlying equations they do not pertain exactly to the team's set up since they do not include the force of the mass added on to the end of the shaft. With this mass included, the analysis becomes beyond the teams capabilities. In the Forces on Bearing section, this is discussed further with the assistance of Dr. Bamberger.

$$\frac{dV}{dz} = m\Omega^2 u(z) \quad (1),$$

$$\frac{dM}{dz} = V \quad (2),$$

$$M = -EI\frac{d^2u}{dz^2} \quad (3),$$

where m = the mass of the shaft per unit length, Ω = the rotational speed of the shaft, V = shear force, M = moment at a length z along the shaft, E = elastic modulus of the material, and I = second moment of area. These three equations can be used to construct a fourth order ordinary differential equation, which can then be used with the appropriate boundary conditions to determine the natural frequency that the rotating shaft may have [1].

Prior Art

The team found a patent that proposed a device similar to the mechanism that they were to design. This patent presented an instrument that measures the shaft displacement. However, it was distinct from the mechanism P2C2 sought to design in that it does not integrate the moving shaft support, which was the focus of the design that Dr. Bamberger requested. Additionally, it measures values not required of the team: inclination angle, rotational speed, and twist angle vibration [2].

The team saw no violation of this patent in the pursuit of their goals.

Customer Requirements

Dr. Bamberger requested that the team design and prototype a device for measuring the vibrational amplitude of an unbalanced cylindrical shaft rotating about its length. His requirements for the mechanism included:

- Ability to add masses of varying sizes to end of shaft
- Ability to be structurally integrated with existing machine
- Ability to operate within machine's range of speed
- Ability to simply support shaft
- Accurate and precise measurement
- Added mass does not change the inertia of the shaft
- Automated moving support with respect to rotating shaft
- Continuous data collection
- Ease of operation
- Low cost
- Maintainability
- Manufacturability
- Robustness

- Safety of operation and manufacturing
- Shaft made of aluminum, steel, or ultra high molecular weight polyethylene (UHMWPE)
- Shaft with natural frequency same as machine when the support is at least 40 percent down the length of the shaft
- Simulation can be run with no mass
- Small traversing resolution for moving support
- Completion by December 10, 2009

In order to determine the most important requirements, the team consulted with the customer. Based on the consultation, they were able to rank the importance of each requirement on a scale from one through five.

A complete list of the requirements, along with their rankings, Customer Weights, can be found in the quality function deployment (QFD) in Appendix A.

Engineering Specifications

After the team defined customer requirements and their importance, they developed a set of engineering specifications, or quantifiable aspects of the requirements that they used as a pass-or-fail standard when the design and prototype were finished.

The engineering specifications were as follows:

- Resolution of vibration amplitude measurement is within 10 percent of the amplitude
- Resolution of moving support location is less than 5mm
- Fixture reduces vibrations at supported point by 75 percent
- Moving support traverses from 10 percent to 90 percent of L_{shaft}
- Sampling rate is more than twice the frequency of shaft rotation
- Sensor is within 2 percent of L_{shaft} from free end

P2C2 then determined how well the specifications correlate with the customer requirements, and integrated the specifications into the QFD. From the resulting calculations, the team was able to determine that its focus area for the design is the ability of the support to traverse the majority of L_{shaft}.

Functional Decomposition

Before beginning the design, P2C2 broke the project into four distinct subsystems or functions, as shown in the Figure 1.

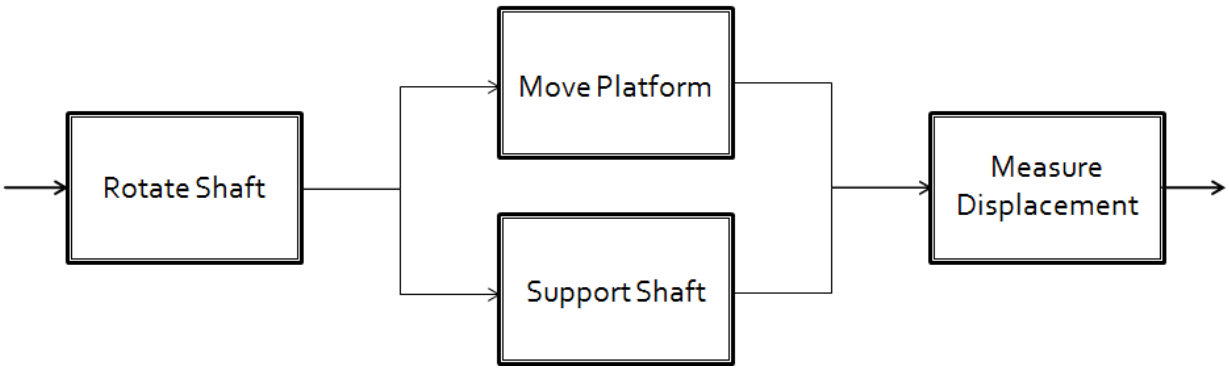


Figure 1: Simplified functional decomposition showing four main subsystems

These subsystems were: (1) rotate shaft, (2) move platform, (3) support shaft, and (4) measure displacement. “Rotate shaft” included the shaft design and the selection of the machine used to rotate the shaft. “Move platform” included the design and movement of the platform or the support mount. “Support shaft” was the design of the simple support, or the contact point between the shaft and the support. Finally, “Measure displacement” included the selection of the sensor used to measure the amplitude of vibration at the end of the shaft. The team brainstormed different ideas to accomplish each of these tasks.

A more detailed functional decomposition that breaks the project down to each input and output can be found in Appendix B.

Shaft Selection

The first box in the functional decomposition was the rotation of the shaft. The team considered three main components in its design: machine, material, and dimensions. This section of the report outlines P2C2’s shaft analysis.

Machine

The customer requested that the machine selected to rotate the shaft have as large an operating speed as possible. This led the team to select the Fadal vertical mill. Another benefit of this machine was that the vertical alignment allowed the team to neglect the effects of gravity.

Material

The customer requested that the shaft be made of aluminum, or UHMWPE. The team eliminated UHMWPE based on the fact that steel and aluminum more accurately represent the engine system for which this device is being created. P2C2 then performed preliminary frequency calculations, which showed that the natural frequency of an aluminum shaft is

similar to that of a steel shaft of the same dimensions. Ultimately, the team chose aluminum for its flexibility, as this property maximized the displacement, which was determined in MATLAB analysis (discussed in the section titled Added Mass on pg. 38) to be fractions of a millimeter, at the end of the shaft.

Dimensions

The customer requested that the shaft have a natural frequency less than 200 Hz (the operating speed of the machine) while the support was at approximately at 40 percent of L_{shaft}. The team determined that the height of the workspace in the Fadal, and thus the maximum length of the shaft, was 30 inches. As a result, they analyzed shafts 10, 15, and 20 inches in length to allow for clearances. These shafts measured 3/8 and 1/2 inch in diameter, and the final diameter and length were determined through FEA.

The team completed calculations to determine the natural frequency of a shaft with a fixed support at one end, modeling the design with the support in the 0 percent position. They also completed calculations of a shaft with a fixed support at one end and a simple support at the other, modeling the design with the simple support in the 100 percent position. They compared the results with the FEA model created in Autodesk Inventor, which utilized a support with a width of 1/64 inch. The equations and position diagrams described are shown below.

Cantilever Beam (support at 0 percent) [3]:

$$f_n = \frac{1.875^2}{2\pi}\sqrt{\frac{EI}{mL^3}} \quad (4),$$

Beam with simple support at end (support at 100 percent) [3]:

$$f_n = \frac{15.4}{2\pi}\sqrt{\frac{EI}{mL^3}} \quad (5),$$

For each equation:

$$m = \frac{\rho D^2 L}{4} \quad (6),$$

$$I = \frac{\pi}{64}D^4 \quad (7),$$

where f_n = the natural frequency of the beam, m = the mass of beam, D = the diameter of beam, L = the length of beam, I = the second moment of area, ρ = the density, and E = the elastic modulus of the material.

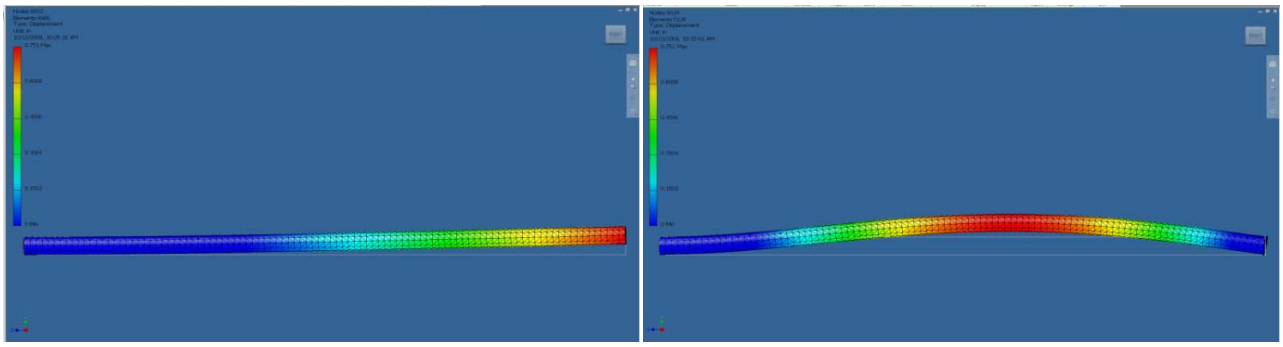


Figure 2: FEA model showing support in 0 percent and 100 percent positions, respectively

Table 1 shows a side-by-side comparison of calculated frequency and the frequency modeled in our FEA model. Comparison shows that the P2C2 FEA model accurately determined the frequency of the shaft at the given locations.

Diameter (in)	Length (in)	Support Location (%)	Freq. from eqns. (Hz)	Freq. from FEA (Hz)
		0	46.28	46.43
0.375	15	100	202.73	209.48
		0	61.71	61.92
0.500	15	100	270.31	274.58

Table 1: Comparison of calculated frequency and frequency determined with model

After confirming that the model was accurate, P2C2 used it to find the frequency of shafts of lengths 10 and 15 inches and diameters of 3/8 and 1/2 inch with the support at various locations to monitor the frequency over a range. Figure 3 shows this data. The line at a frequency of 200 Hz illustrates the maximum safe operating speed of the Fadal mill.

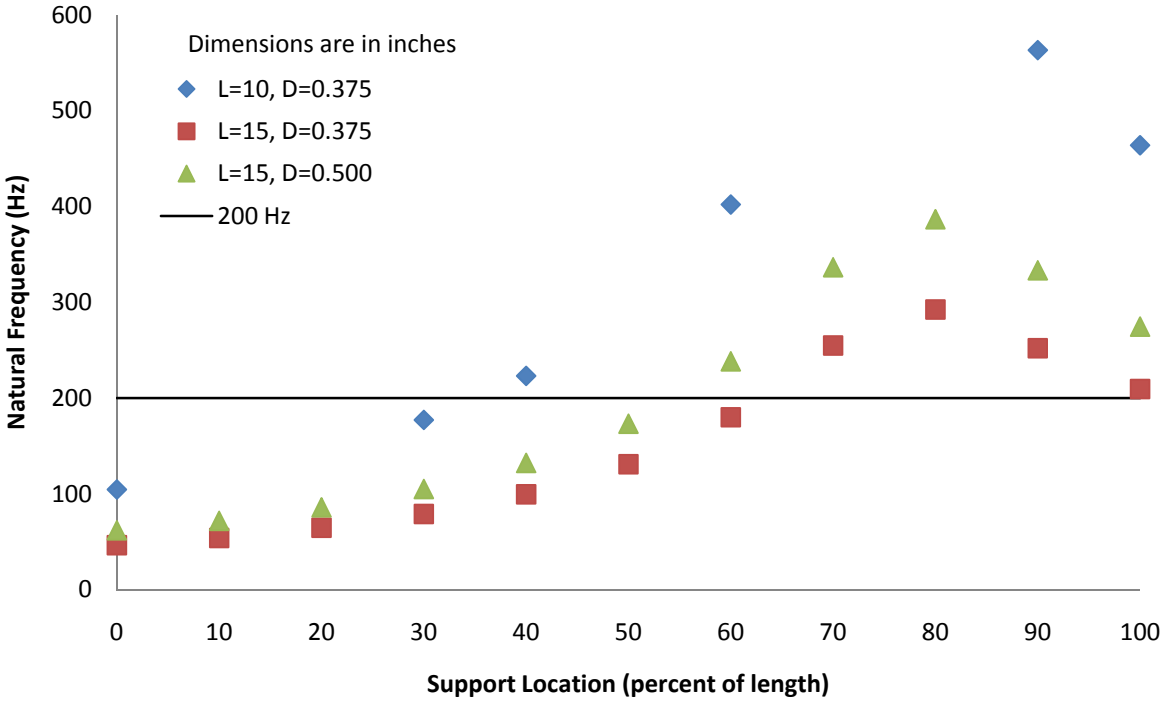


Figure 3: Relationship between location of support and natural frequency of shaft

Interpretation of the above data and discussion with the customer determined that the frequency range of the 10 inch shaft was too large. The team thus shifted its focus to the 15 inch shaft. The support used in the above data was 1/64 inch wide. Bearing research determined that the support will be closer to 1/2 inch wide, so FEA was repeated with a 1/2 inch wide support. The discrepancy is shown in Figures 4 and 5.

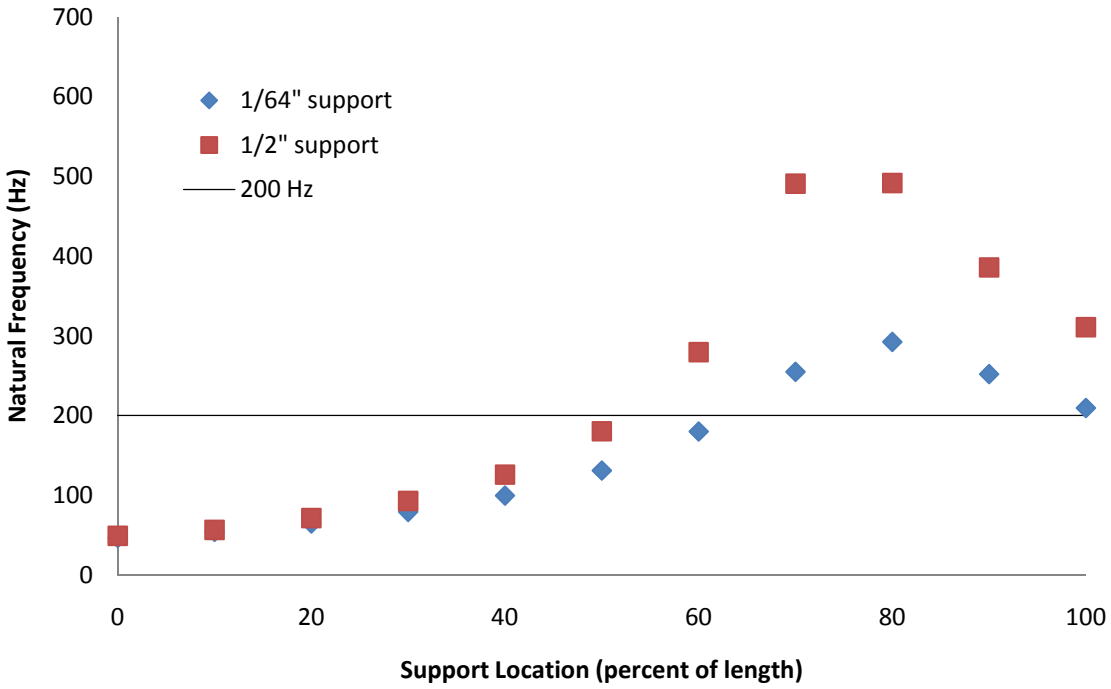


Figure 4: Discrepancy in frequency between 1/64 and 1/2 inch support for 15 inch shaft with 3/8 inch diameter

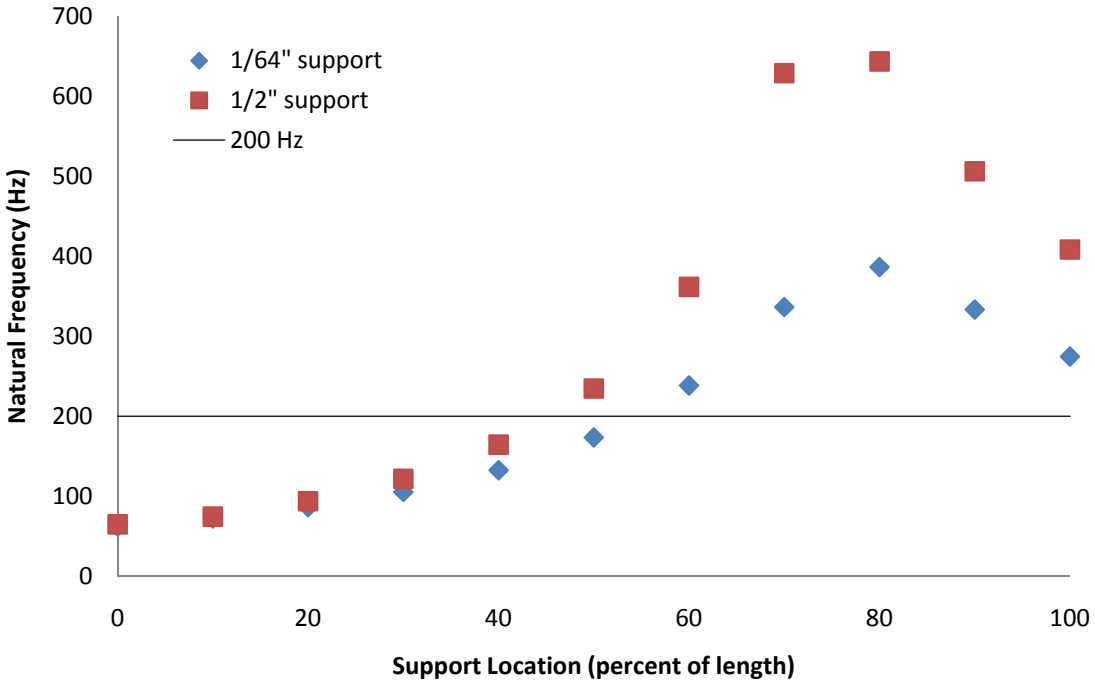


Figure 5: Discrepancy in frequency between 1/64 and 1/2 inch support for 15 inch shaft with 1/2 inch diameter

The mechanics behind the discrepancy are shown in Figure 6. One can see that the displacement at the end of the shaft with the 1/64 inch support, the support that models a simple support, created a larger displacement at the end of the shaft. Thus, the team sought to create a thin support. It should be noted that, despite this fact, all subsequent data was calculated with the 1/2 inch wide support since it more accurately modeled the design. This estimate was considered to be conservative since the selected sleeve bearing was 1/2 inch wide and the self-aligning ball bearing allowed axial misalignment. This combination created a support that acted like something in between a 1/64 inch support and a 1/2 inch support.

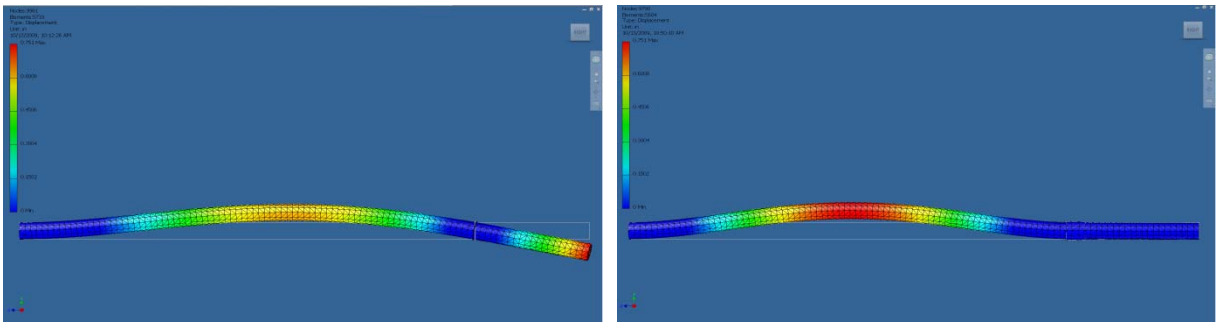


Figure 6: FEA model showing frequency difference between 1/64 inch wide and 1/2 inch wide support

Further discussion with the customer concluded that the frequency for the 15 inch shaft with the wider support exceeded his desired frequency range. The team therefore completed analysis on a shaft with length 20 inches and both diameters at 3/8 inch and 1/2 inch. The data can be found below.

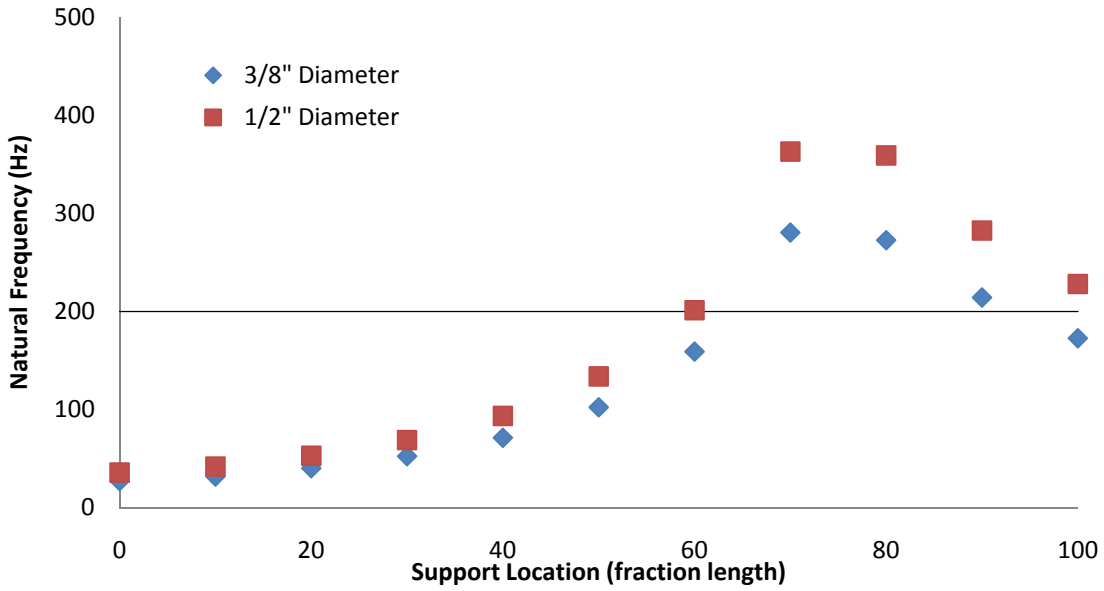


Figure 7: Frequency of a 20 inch shaft with a 1/2 inch wide support

From the data above, P2C2 determined that the 20 inch shaft with a 1/2 inch diameter reached 200 Hz closer to the requested 40 percent than the 3/8 inch diameter shaft. The team thus chose a 20 inch shaft with a 1/2 inch diameter for their design.

Concept Generation and Selection

The following presents P2C2's concept generation for the support shaft, move platform, and measure displacement. The team's more comprehensive basic concept generation can be found in Appendix C.

Support Shaft

The team created several shaft support options that meet the customer requirements and engineering specifications with varying degrees. This section of the report outlines the generated concepts for the shaft support and the pros and cons of each concept.

Self-aligning ball bearing or spherical roller bearing: The first support option involved using either a self-aligning ball bearing or a spherical roller bearing. These types of bearings were considered because their geometries allow for axial misalignment between the inner shaft and the outer housing. This would allow the support to act more like a simple support (only restricting radial displacement, not axial misalignment). Aside from allowing misalignment, this option had the benefits of being readily available for purchase, operable at high speeds, able to sustain significant radial forces, and easily installed. The downside of this option was that these types of bearing were not designed to allow for slip between the inner race and the inside shaft. This would require some clearance between the inner race and the shaft. The clearance would allow radial vibration of the shaft and reduce the supporting behavior.

Figure 8: Self aligning ball bearing and spherical roller bearing [4, 5]

Ball bearing with no inner race: The second support option involved using a single-row ball bearing with no inner race. In this configuration, the balls would contact directly with the shaft. If this geometry worked reliably, it would allow for thrust slip, axial misalignment, and high speed operation. This geometry would be the ideal simple support as there would be only a single point of contact restraining the shaft. The downsides of this option were that this type of bearing was not readily available, there would be added complexity and added concerns about safety, robustness, and reliability.

Figure 9: Ball bearing, no inner race [6]

Externally mounted ball bearings: The third support option involved using three ball bearings mounted externally to the shaft. One of these bearings would have to be moveable to allow adjustment of clamp tightness and insertion of shaft. This option had the benefits of allowing slip between the shaft and the bearings (as the bearings are not mounted directly to the shaft), being able to sustain large forces and function as high speeds, and having components that were readily available for purchase. The downsides of this option were that the geometry of the bearings would constrain axial misalignment, the need for three bearings would add complexity to the design, and the asymmetry of the support would alter the dynamics of the shaft.

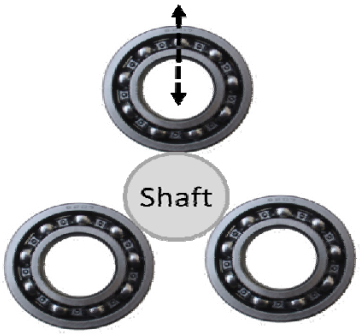


Figure 10: Triple ball bearings [6]

Polymer sleeve bearing: The fourth support option involved using a Vespel (polymer) sleeve to constrain the shaft. This type of bearing would allow for rotation of the shaft and slip between the shaft and the support. Also, Vespel sleeve bearings that can function at high speeds were readily available and easy to install. The group explored the option of using other types of sleeve bearings, including metallic, but these materials were not suitable for the high speed application of our design. The downsides of this option were that the geometry of the bearing would constrain axial misalignment, thus not acting like a simple support. Also, use of a polymer sleeve bearing would only allow for 160N when operating at full speed.

Figure 11: Vespel sleeve [7]

Self-aligning bearing and sleeve bearing: This concept involved the use of a sleeve bearing within a self-aligning ball bearing. This combination would produce the axial misalignment benefits of a self-aligning ball bearing and the axial slip between the sleeve bearing and the shaft. This design would also be able to operate at high speeds and have components that are readily available for purchase. The downsides of the option were that it would add complexity to the design and there was uncertainty as to where the rotation would occur. If the rotation occurred between the sleeve bearing and the shaft, the sleeve bearing would only be able to sustain small loads, but if the rotation occurred within the ball bearing, the support would be able to operate at high speed and maintain large loads.

Figure 12: Combination bearing [4, 7]

Forces on bearing: Other than when the shaft is at resonant frequency, the forces on the bearings would be well within the allowable force range of all the bearing concepts. Force analysis was needed before the team could analyze the designs because some of the concepts, such as the sleeve bearing, were not able to sustain as large of forces as the others. A meeting was held with Dr. Bamberger where he presented MATLAB code that both predicted the displacement of the beam off its axis of rotation and the forces that the support would be subjected to at each point along the shaft. This force was fairly constant for most of the shaft length but would spike when the support reached a location that made the system have a natural frequency equal to the rotational frequency of the shaft. These spikes can be seen in the figures on the following page. The first figure shows the forces on the support, and the second figure shows the predicted shape for the shaft while the support is at a certain location. One predicted shape of deformation approached infinity at the end of the shaft. This was the predicted shape with the support at the natural frequency location. Discussions with Dr. Bamberger led to the conclusion that this location needed to be avoided during testing by stopping the rotation of the shaft and moving the support past this location.

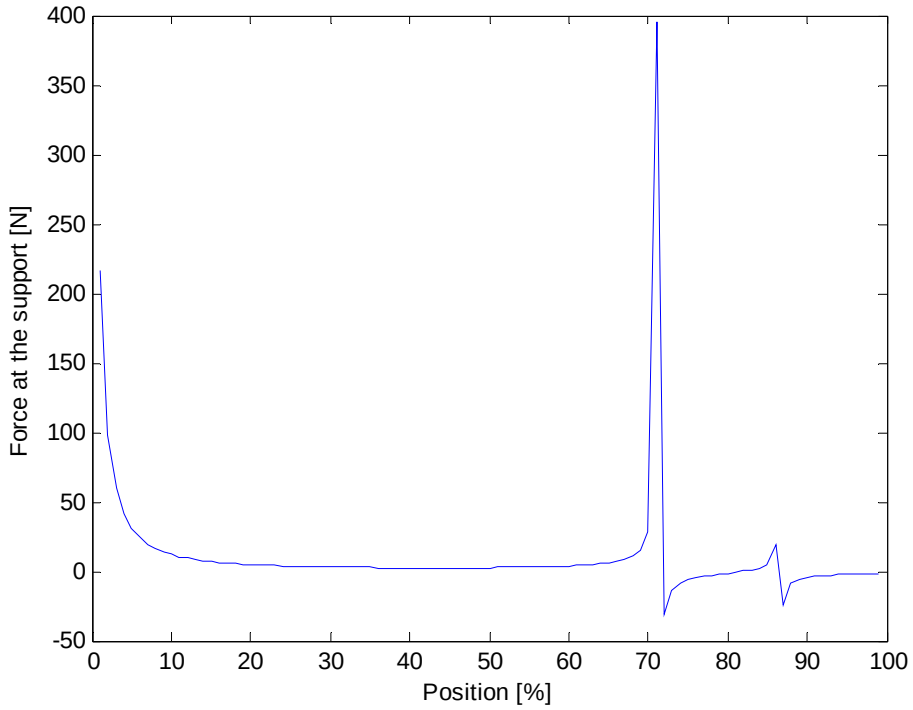


Figure 13: Forces on support vs. support position along shaft

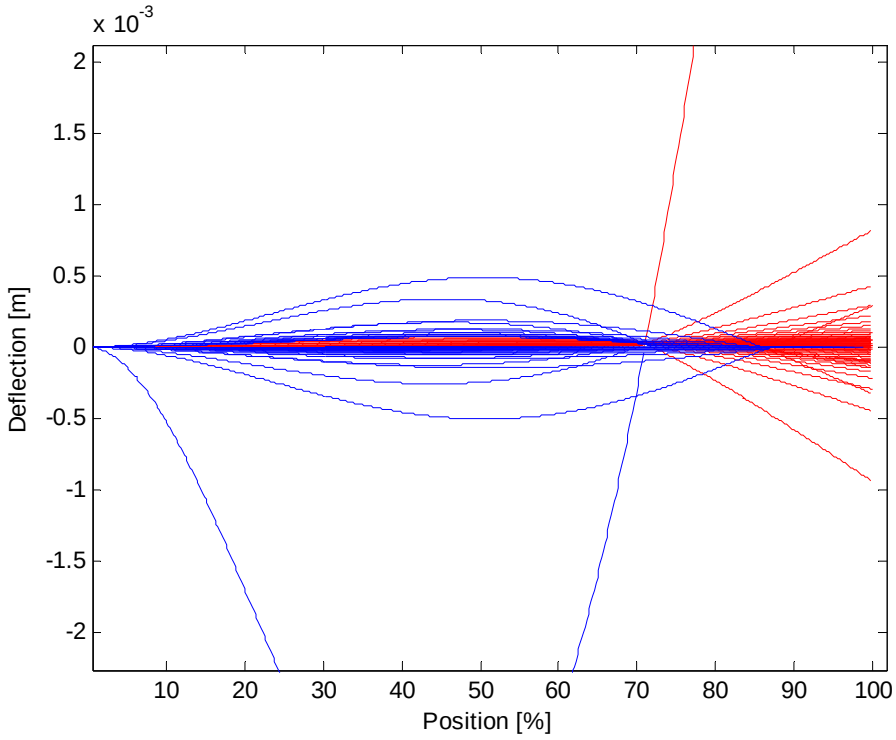


Figure 14: Predicted shape of deformed shaft with support located at blue-to-red transition

Concept selection: The five shaft support concepts above were evaluated against each other based on the customer requirements and the expected forces on the bearings. For each of the customer requirements, the five concepts were ranked 1 to 5 (1 being the worst concept at meeting that requirement and 5 being the best). Each requirement was weighted according to its weighting in the team's current QFD for the project. After the concepts were evaluated on each customer requirement, the total weighted score of each concept was calculated. Using this objective selection process, the team determined that the fifth concept (self-aligning ball bearing with a sleeve bearing) would best meet the customer requirements. It was determined that the benefits of this concept in allowing axial misalignment, high speeds, and high loads would outweigh the downsides of increased complexity to the design. The scoring matrix for evaluating the support options is shown below.

	Ability to operate within machine's range of speed	Ability to restrain shaft	Automated moving support with respect to rotating shaft	Ease of operation	Low cost	Maintainability	Manufacturability	Robustness	Completion by December 10, 2009	**Sum**
CUSTOMER WEIGHTS	5	5	5	2	2	2	1	2	5	
Self-aligning bearing	5	1	1	4	5	3	5	5	5	**99**
Ball bearings, no inner race	5	5	3	1	1	1	2	1	2	**85**
Triple ball bearings	5	3	2	2	4	5	2	2	1	**83**
Sleeve bearing	1	1	5	5	3	3	5	3	5	**93**
Sleeve bearing + self-aligning bearing	2	4	5	3	2	4	3	4	5	**109**

Table 2: Evaluation matrix for support shaft function

Move Platform

The third subsystem of the design consisted of the moving platform that housed the support and traversed its length. Four designs are detailed below.

Dual screw drive with stepper motor(s): This design included a four post tower structure with a vertical bar at each corner. Two of these bars were to be threaded and attached to a single motor with a gear system or to two separate stepper motors. The other two bars were to be smooth. The platform was to translate due to rotation of the threaded shafts that were to be inserted in threaded holes in the platform. This design was advantageous in that it would have had a small movement resolution, the structure would have been sturdy, and stepper motors would have been relatively simple to program. However, some disadvantages were that it would have had binding issues, it would have required a large amount of manufacturing, and the structure would be large.

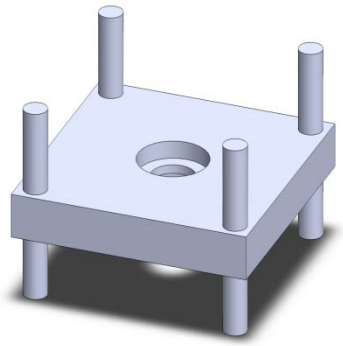


Figure 15: Dual screw with stepper

Scissor lift with pneumatic actuators: This design included a scissor lift with pneumatic actuators to lift the platform. Advantages of this design included the fact that high forces created by the pneumatics would have overcome any friction between the support and the testing shaft, and fast motions are possible. Disadvantages, on the other hand, included poor resolution in movement, a high complexity, and mechanical interference with sensor positioning.

Figure 16: Scissor lift

Platform off linear motor: This design utilized a linear motor that was already available to the team. A support platform was designed and manufactured to integrate with the motor. This platform attached directly to the motor and the structure already in place would attach to the T-bed to support the motor. One advantage of this design was its simplicity. The accuracy of the linear motor was known (and exceeded the requirement), and the motor had a pre-existing support system. The main disadvantage of this design was the difficulty of programming the motor. Additionally, the cantilever nature of the design could have limited the structural rigidity, compared to other concepts.

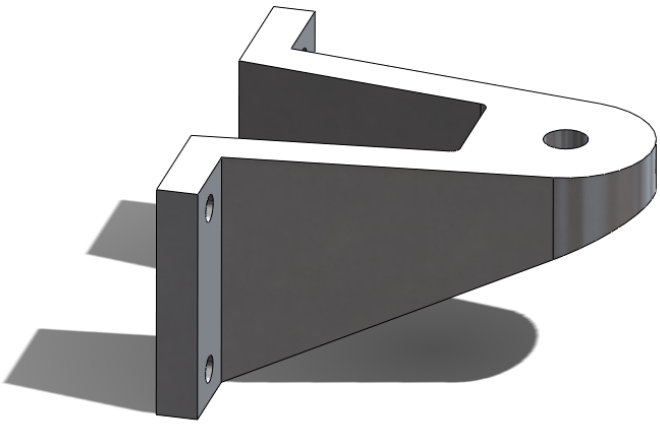


Figure 17: Directly-driven platform

Four post structure with linear motor: This design was similar to the previous in that it also used the linear motor. However, to add more rigidity to the structure, this design included the four post tower structure previously described. The only difference in the tower structure was that for this design, the posts were all smooth. Advantages to this design were that the accuracy of the linear motor was known (and exceeded the requirement) and that it was more structurally sound than the platform described above. Disadvantages included its high amounts of manufacturing, binding issues, and difficulties with motor programming.

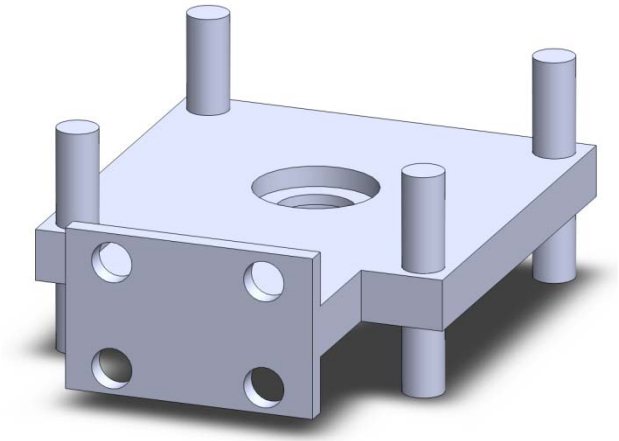


Figure 18: Four post platform

Concept selection: The customer wanted the motion of the platform to be automated and continuous and for it to have a resolution less than 3 percent of L_{shaft}. He required that the support be sturdy enough to properly serve as a simple support on the shaft, be capable of being integrated with an existing machine, and be robust. The team evaluated the above designs on each of these requirements and formed the scoring matrix below.

	Ability to be structurally integrated with existing machine	Ability to restrain shaft	Automated moving support with respect to rotating shaft	Low cost	Maintainability	Manufacturability	Robustness	Small traversing resolution for moving support	Completion by December 10, 2009	Sum
CUSTOMER WEIGHTS	5	5	5	2	2	1	2	4	5	
Scissor lift	1	1	1	1	1	2	1	1	2	**37**
Dual screw	2	3	2	2	3	2	2	4	2	**77**
Four-post structure w/linear motor	3	4	4	3	3	3	3	3	3	**103**
Platform off linear motor	4	3	4	4	4	4	4	3	4	**115**

Table 3: Evaluation matrix for move platform function

Added Mass Options

The team designed several different concepts for the added mass at the end of the shaft. This section of the report outlines the options generated and the pros and cons of each design.

Threaded hole in shaft: The first added mass option involved threading a screw into the end of the shaft. Washers would be placed on the screw to allow the add masses of varying size. This was the most simple of the designs considered, but would alter the mass properties of the shaft; when the screw would be removed, the resulting hole in the shaft would create an off balance mass. Thus, the team would be unable to measure the dynamics of the shaft without an added mass.

Figure 19: Threaded hole in shaft

Threaded collar tightened to shaft: The second added mass option involved adding a collar around the end of the shaft. This collar would be held in place by a set screw and numerous washers could be added to vary the off balance mass. Unlike the previous design, this option was removable which would allow P2C2 to measure the dynamics of the shaft without the added mass. However, the mass of the collar and screw would be significant relative to the mass of the entire shaft and would alter the inertial properties and natural frequency of the shaft. This conclusion was made based on FEA frequency analysis of the shaft with the added mass.

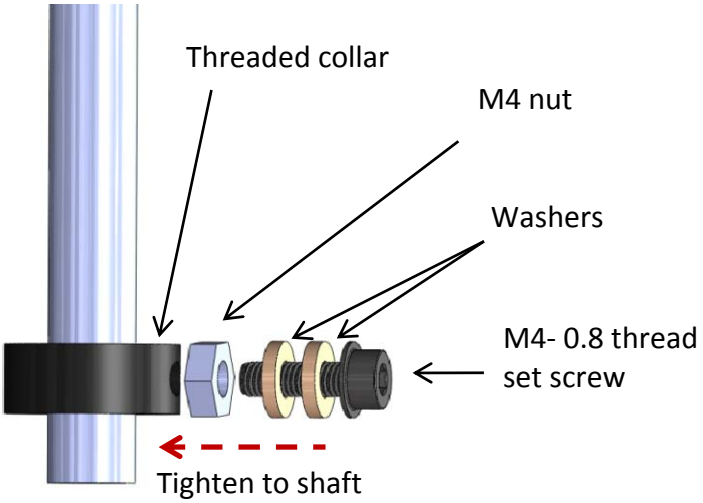


Figure 20: Threaded collar tightened to shaft

Nuts on end of shaft: The third and final of the added mass options consisted of threaded steel pieces of varying sizes. These masses were threaded to the end of the shaft and held in place by nuts, the last of which being a lock nut. This arrangement is shown in Figure 21(a). An alternative arrangement, one with more nuts as shown in Figure 21(b), ensured that the mass of the shaft does not change when the steel piece was removed; only the distribution of the mass changed. With these two configurations, the team better determined the effects of

adding an off balance mass. One disadvantage of this design was that it required the team to add external threads to the end of the shaft.

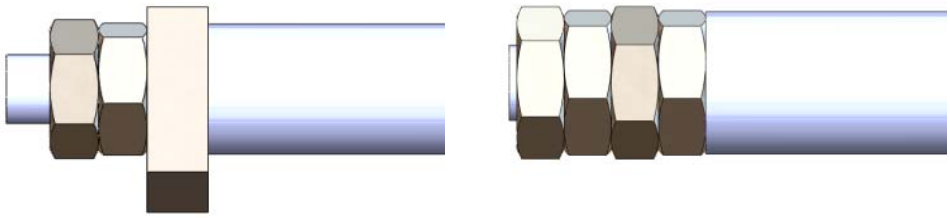


Figure 21: (a) Nuts with off balance mass, (b) Nuts without off balance mass

Concept selection: The three added mass concepts above were evaluated against each other based on the customer requirements. For each of the customer requirement, the three concepts were ranked 1 to 3 (1 being the worst concept at meeting that requirement and 3 being the best). Each requirement was weighted according to its weighting in the team's current QFD for the project. After the concepts were evaluated on each customer requirement, the total weighted score of each concept was calculated. Using this objective selection process, the team determined that the third concept (the nuts on the end of the shaft) best met the customer requirements. The scoring matrix for evaluating the support options is shown below.

	Mass does not change inertia of shaft	Simulation can be run with no mass	Safety - Mass must stay on shaft when rotated	Can add masses of varying sizes	Completion by December 10, 2009	Sum
CUSTOMER WEIGHTS	3	4	5	4	5	
Threaded hole in shaft	1	1	1	2	3	**35**
Threaded collar tightened to shaft	1	3	2	1	3	**44**
Nuts on end of shaft	3	2	3	3	3	**59**

Table 4: Evaluation matrix for the added mass

Measure Displacement

The last function considered was the measurement of the displacement of the shaft at its free end. Upon research and consultation with sensor technical representatives, P2C2 considered several types of displacement sensors for the design, including laser triangulation, confocal, linear variable differential transformer (LVDT), and thru-beam. Below is a brief description of each type, along with their pros and cons.

Laser triangulation: In laser triangulation sensors, a laser diode emits a point of light onto a target surface, which creates scattered light that goes through an optical lens onto an electrical array. The electrical array detects any change in position with respect to the sensor. These types of sensors are advantageous for situations in which the target surface is not flat because of their small beam-spot. Additionally, lasers enable the actual sensor to be positioned far away from the object. However, these sensors are dependent on surface finish [8].

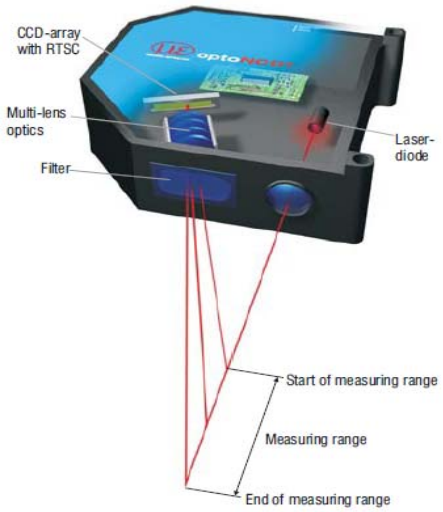


Figure 22: Laser triangulation sensor [8]

Confocal: Confocal sensing entails the focusing of white light onto a target surface via multiple lenses. Each of the lenses corresponds with a certain wavelength, such that when the light reflects off the target surface, only the wavelength that is exactly focused on it passes back through an aperture to a receiver. The receiver, in turn, processes spectral changes and returns displacements. An advantage that confocal sensors have over other sensors is the fact that they are small. However, the method of measurement requires that the sensor is close to its target surface, creating safety issues if the amplitude of vibration is too large [8].

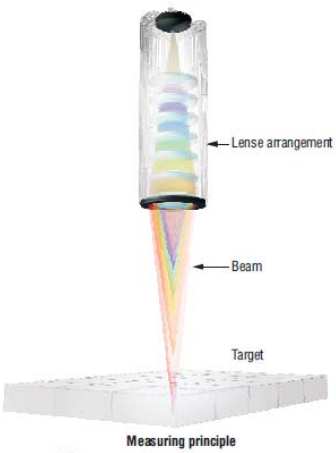


Figure 23: Confocal sensor [8]

Thru-beam: A light source emits a parallel, continuous light curtain which is aimed at a receiver. If there is an object in between the light source and the receiver, interruptions of the light curtain occur, which determine displacements. An advantage to thru-beam sensors is that they are not material-dependent. Another one is that they are able to take 2D data, as opposed to the 1D data of the other sensors presented. However, they tend to have larger footprints, which would make integration with the vertical mill more difficult [9].

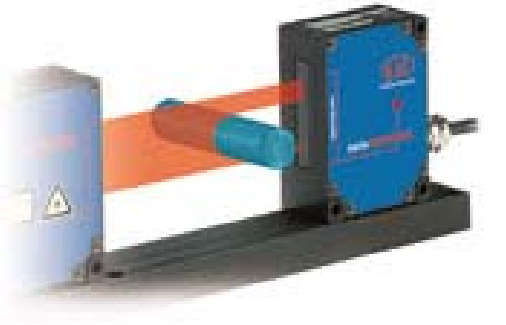


Figure 24: Thru-beam sensor [9]

LVDT: In LVDT sensing, a moveable armature is placed coaxial to a cylindrical coil assembly. When the armature is pushed in, it creates an electrical signal that is proportional to the displacement. LVDT sensors are good because of the fact they are able to measure the position of any solid. They are extremely cheap due to their common use in industry and are small, making for easy integration into systems. However, LVDT sensors require direct contact with the target surface, which can pose a safety issue if the target surface is quickly moving [10].

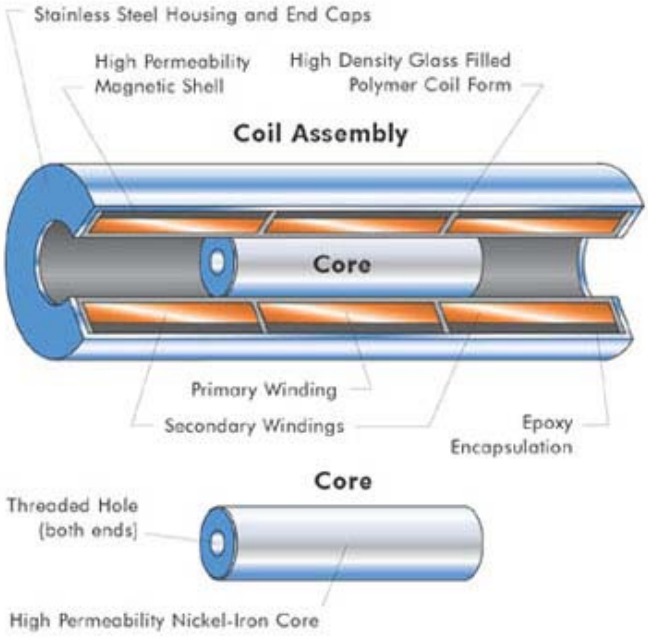


Figure 25: LVDT sensor [11]

Concept selection: After P2C2 considered the differences between the types of sensors, they determined that the customer requirements applicable to measuring displacement were the following: ability to be structurally integrated with existing machine, ability to operate within machine's range of speed, accurate and precise measurement, continuous data collection, ease of operation, low cost, maintainability, and robustness.

In addition to the requirements, the team took two things into consideration. The first was the fact that the sponsor had two laser triangulation sensors that met their needs. This was accounted for in the column for low cost. A four was assigned to the no cost item, and ones were assigned to the other sensors, which all had a much higher cost, ranging from $5,000 to $10,000.

The second consideration was the fact that the customer's lab has worked with a technical contact from Optimet, the company that makes the laser triangulation sensors owned by the lab. P2C2 has established contact with this representative, and he will serve as a reference for any operation and maintenance problems that may arise during our testing. For this reason, the team has ranked the laser triangulation sensor the highest in both ease of operation and maintainability.

Based on the customer requirements and the additional considerations, the team was able to decide which type of sensor would be best for their purposes: the laser triangulation sensor. The scoring matrix for evaluating the sensor options is shown below.

	Ability to be structurally integrated with existing machine	Ability to operate within machine's range of speed	Accurate measurement	Continuous data collection	Ease of operation	Low cost	Maintainability	Precise measurement	Robustness	Completion by December 10, 2009	Sum
CUSTOMER WEIGHTS	5	5	4	4	2	2	2	4	2	5	
Laser triangulation	2	4	4	4	4	4	4	4	4	4	**130**
Confocal	4	4	4	4	3	1	3	4	2	2	**116**
Thru-beam	1	4	4	4	3	1	3	4	4	2	**105**
LVDT	4	4	4	4	3	1	3	4	1	3	**119**

Table 5: Evaluation matrix for measure displacement function

Alpha Design

The team's preliminary design came from the combination of the selected subsystems discussed above. This design included the following components:

- Aluminum shaft (6061 alloy, 20 inches long, 1/2 inch diameter)
- Added mass
- Fadal vertical milling machine
- Provided ATS 10040 linear motor
- Provided motor mount
- Support housing
- Miscellaneous connector set screws
- Self-aligning ball bearing (not depicted)
- Vespel polymer sleeve bearing (not depicted)
- Sensor (not depicted)
- Sensor mount (not depicted)

The general design layout without the sensor is depicted below.

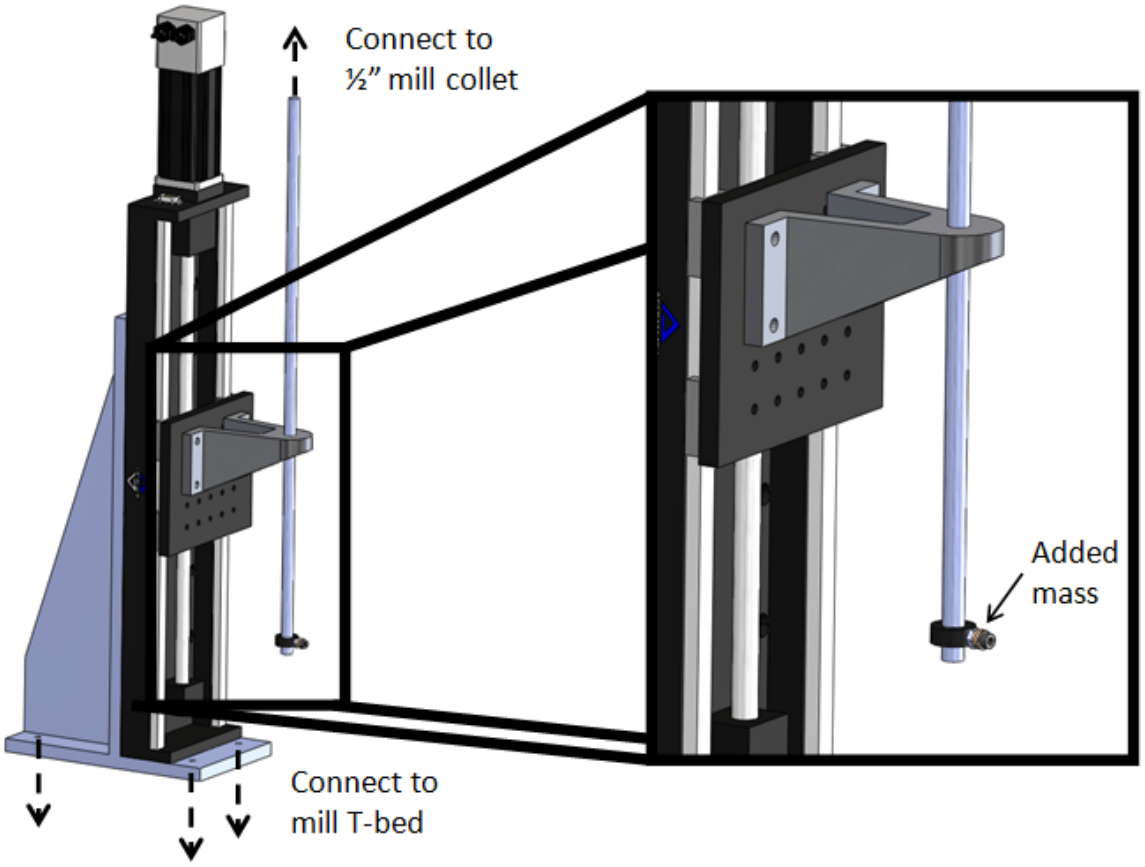


Figure 26: General layout of preliminary design

Support Housing

The main component requiring machining was the support housing. The team made a preliminary material selection of aluminum for this component based on the low loading requirements on the part, the need to minimize weight, and the desire to increase machinability. The support housing was designed to universally hold any of the bearings being considered at the time, with changes only to the internal detail. The final geometry and material were determined before Design Review Three. This later analysis is described in "Bearing Housing" section on pg. 40. See the figure below for the preliminary support housing design.

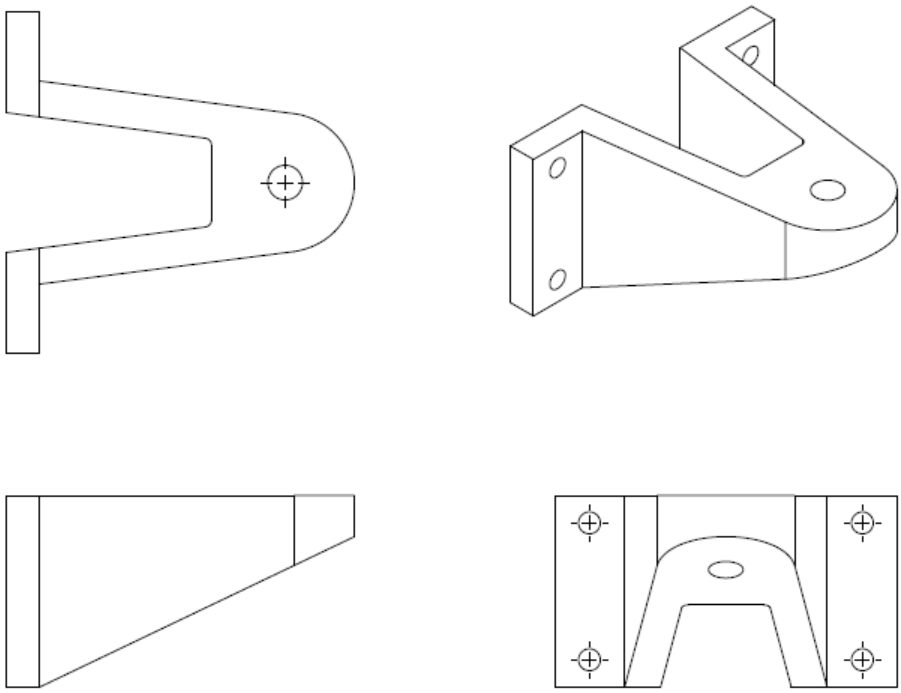


Figure 27: Preliminary support housing design

Connection of Subsystems of Preliminary Design

Figures 28-30 show how the different subsystems are connected.

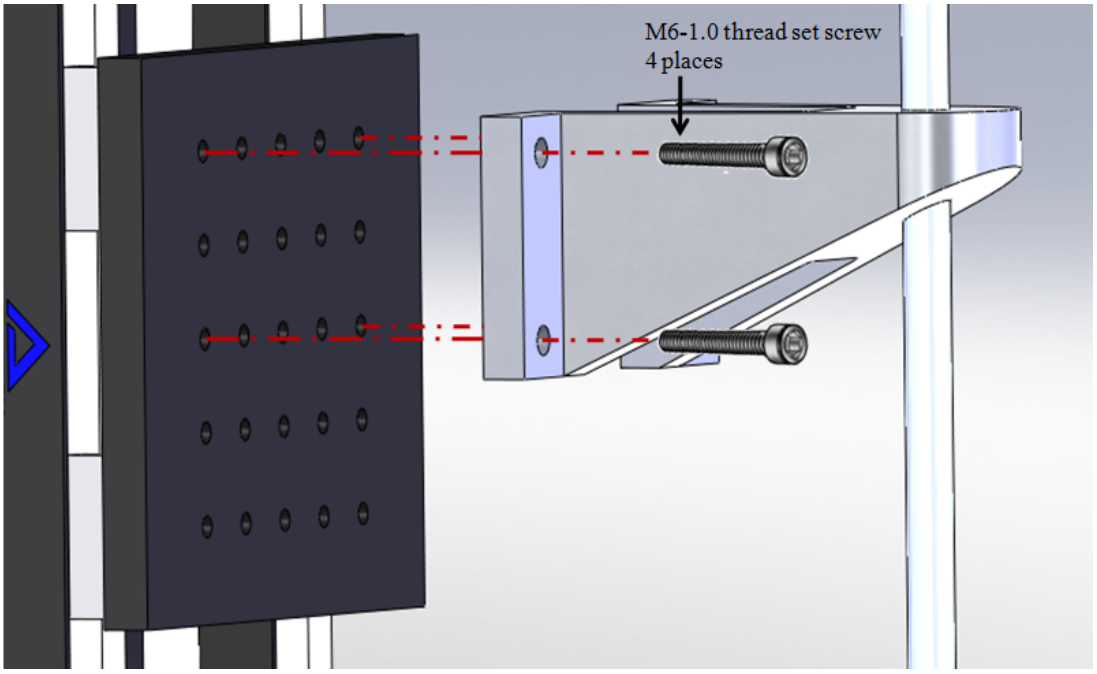


Figure 28: Connection of bearing mount to linear motor

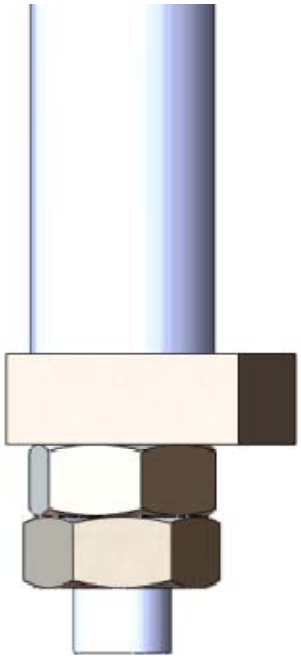


Figure 29: Connection of mass to shaft

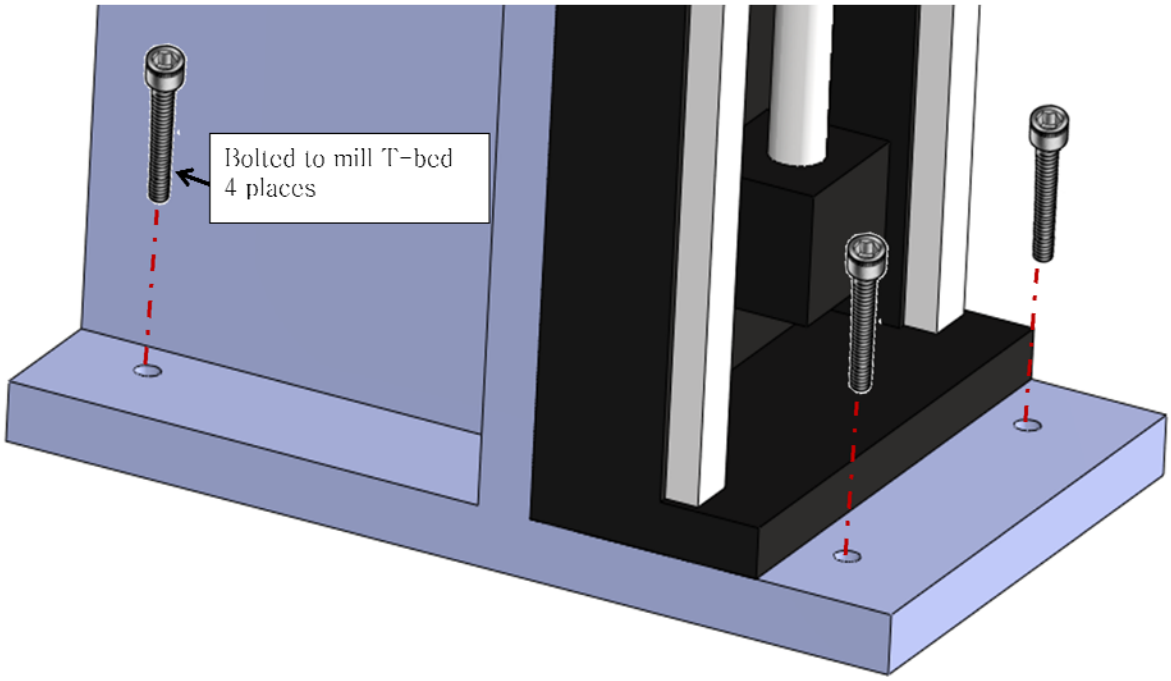


Figure 30: Connection of motor mount to mill T-bed

Engineering Analysis

This alpha design involved several engineering fundamentals: vibrations, material elasticity, and controls/mechatronics. It also entailed that the team use skills such as using engineering software, prototyping, and testing.

Vibrations

The first engineering fundamental was vibrations. The shaft was vibrating at a frequency less than or equal to 200 Hz. An understanding of vibrations was essential to design the shaft dimensions properly and make sure that the support could withstand any forces that these vibrations caused. It was important to know where the support would be along the shaft when the system's resonant frequency would be equal to the rotational speed.

Elastic Deformations

The second engineering fundamental was the study of elastic materials. The shaft vibration needed to be modeled as an elastic body.

Controls and Mechatronics

The third engineering fundamental was controls/mechatronics. Computer programming was required to control the motor, which moved the support along the shaft, and the sensors, which measured the amplitude of the vibrations at the end of the shaft.

Force Analysis Using Finite Element Analysis

P2C2 took advantage of engineering software. In order to properly determine the shaft's dimensions, they used FEA to obtain its predicted natural frequencies and used FEA to do force analysis and design optimization. Additionally, they used a MATLAB simulation created by their sponsor to predict the forces on the support and the displacement of the end of the shaft under certain conditions.

Prototyping

P2C2 applied their prototyping skills in the Reconfigurable Manufacturing Laboratory. The team set up experiments to safely determine how the support behaved with the rotating shaft and how it behaved while traversing the length of the shaft. Finally, they ran tests to evaluate the displacement measuring device.

Material Selection and Environmental Impact

The team also used the Cambridge Engineering Selector 5.1.0 to select the optimal materials for two components and analyzed the environmental impact of the project using SimaPro 7.1. The results of these analyses are included in Appendix G.

Design for Safety

This section outlines the safety procedures considered within the final prototype design. This design had several safety issues arising from one root, high speed rotations. Also, safety was taken into account when dealing with laser measuring devices that were used with the prototype.

The major hazard was the fact that this design included an off-balance shaft that was rotated at 200 Hz. There was an off-center mass attached to the end of the shaft via threaded nuts along the center axis of the shaft. If one of these nuts had vibrated loose and become airborne, it could have caused a major injury. In order to eliminate this hazard, the mechanism was contained within a mill work area that was surrounded by a safety shield made of high impact resistant polycarbonate.

The dynamics of the rotating shaft changed drastically when the system was in or near a resonant state. The theoretical displacements seen approached infinity, making the shaft deform plastically and creating yet another hazard. This additional hazard associated with the rotation of shaft could have been eliminated by keeping the system out of its resonant state. Simulations had been run to determine when the shaft would be in this resonant state so that

that given support location could be avoided accordingly. Although this analysis was performed, the lurking variables such as shaft straightness and uniformity caused the shaft to hit resonance during testing and to fracture.

Another hazard was presented with the use of lasers within the design. FDA Class II laser sensors were used to measure the displacement of the end of the rotating shaft from its axis of rotation. This hazard was difficult to eliminate since the lasers had to be turned on and pointing at the shaft at all times. User awareness was the best countermeasure to avoid this hazard. Since the users were aware that the lasers were on, they were sure not to stare directly into them (the only way a Class II laser could cause harm).

A third hazard was present in the operation of the mechanism. After a test was finished, the bearing in which the shaft rotates could have been hot and could have reached temperatures high enough to burn. The team tried to eliminate this hazard with the use of bearing grease. However, a constant flow of fresh lubricant could not be applied since it would reduce the ability of the lasers to measure the shaft's displacement, so the best way to avoid this hazard was with user awareness, as well. The users were aware that the bearing could be hot after prolonged usage and took the proper precautions to avoid burning.

Final Design

The following section describes, in detail, the numerous components, both mechanical and electrical, that made up the final design. The analysis for each part is also included.

Mechanical Design and Analysis

The team's final design came from the combination of the selected subsystems discussed above. This design included the following systems and components:

- Unbalanced shaft
  - Aluminum shaft
  - Added mass (threaded steel piece and nuts)
- Bearing assembly
  - Bearing adapters
  - Bearing housing
  - Delrin polymer sleeve bearing
  - Self-aligning ball bearing
  - Restraint plate
- Linear motor
  - Aerotech ATS 10040 linear stage (provided)
  - Aerotech BMS60 DC servomotor (provided)
  - Motor cables and controller (provided)
  - Motor mount (provided)
  - T-bed adapters for linear stage

- Sensors
    - Optimet Conoprobe 1.1B point sensors (one with a 50mm lens and the other with a 100mm lens) (provided)
    - Sensor cables (provided)
    - Sensor mounts (provided)
    - T-bed adapter plate for sensor mounts
- Fadal VMC 4020 vertical milling machine (provided)
- Miscellaneous fasteners

The final design layout is depicted on the following page. The engineering drawings for all components manufactured are included in Appendix F.

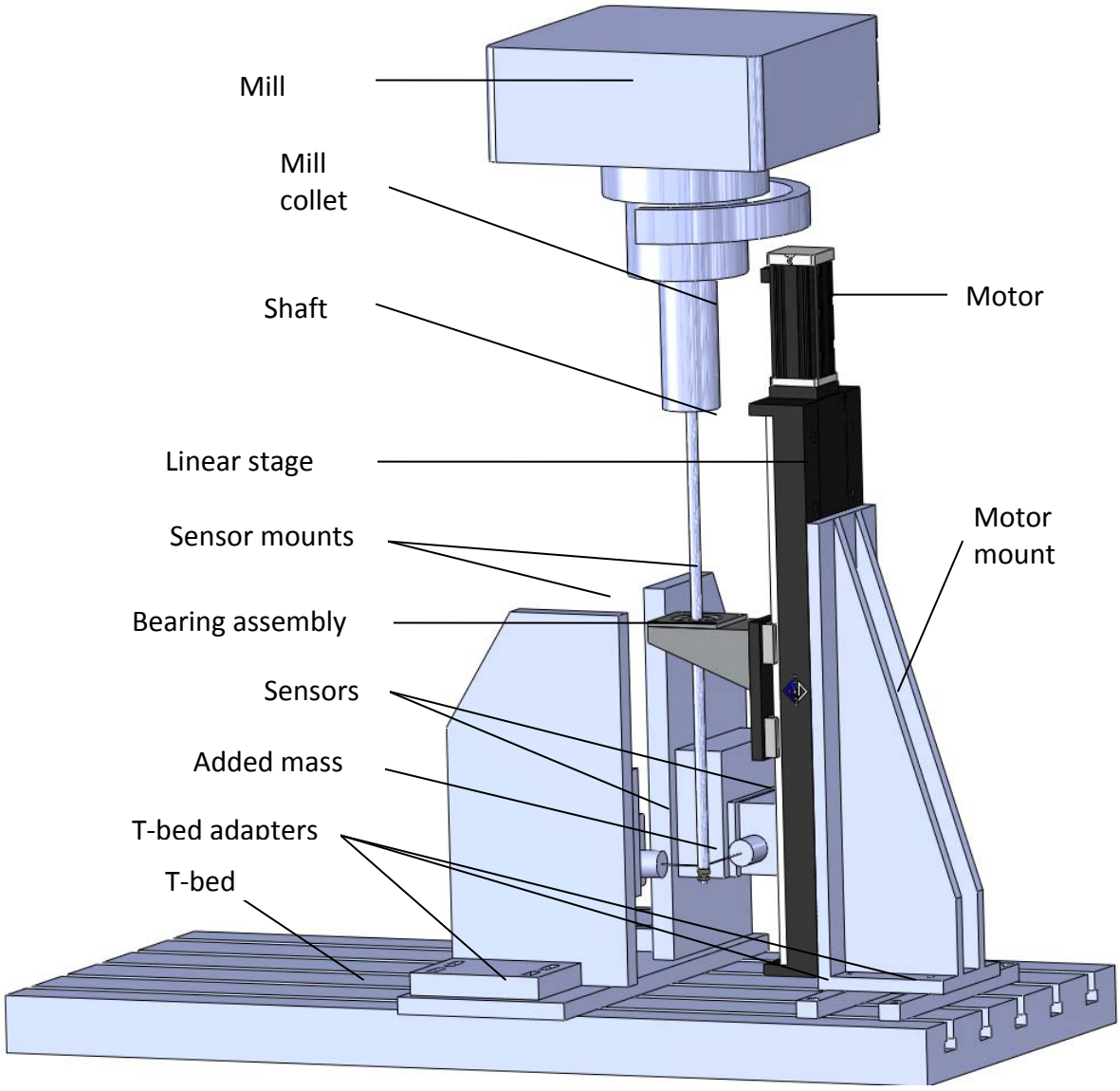


Figure 31: General layout of final design

Aluminum Shaft: The aluminum shaft in the final design was largely the same as it was in the alpha design from Design Review Two. It was still made of 6061 aluminum and had a free length of 508 mm (20 in). The only changes to the shaft were that it had a 12mm diameter, instead of 12.7 mm (1/2 in), and it included the threaded extension to add the off-balance mass. After considering several options for the off-balance mass, the team selected this threaded option, shown in greater detail below, as the addition of the mass would have minimal effect on the dynamics of the shaft. The diameter of the shaft was changed to 12 mm so the team could reduce the size of the sleeve bearing and have more room for the bearing adapters. A single layer of white spray paint was applied to the last inch of the shaft (shown in Figure 32) to increase the effectiveness of the laser sensor, per recommendation of the sensor manufacturer, Optimet. The thickness of this paint is 1µm and did not affect the dynamics of the shaft [12].

Added Mass: The added mass in the final design consisted of threaded steel pieces of various sizes (13x12x6mm, 16x12x6mm, and 19x12x6mm). The hole was located as shown in Figure F7 in Appendix F. These masses were threaded to the end of the shaft and held in place by nuts, the last of which was a lock nut. Having multiple nuts and a lock nut provided redundancy to ensure that nothing vibrated loose during experimentation. Also, the shaft had a right-hand thread to ensure that all components were tightened upward on the shaft when the mill accelerated up to speed as the Fadal rotated clockwise (when looking down from the spindle above). Since deceleration only occurred at the end of the experiment, the team checked the tightness of the mass and nuts after each test and tightened them if needed. When running experiments with no desired added mass, the team added only nuts to the shaft to eliminate the inertial effects of the added mass. An image of the added mass design is shown below.

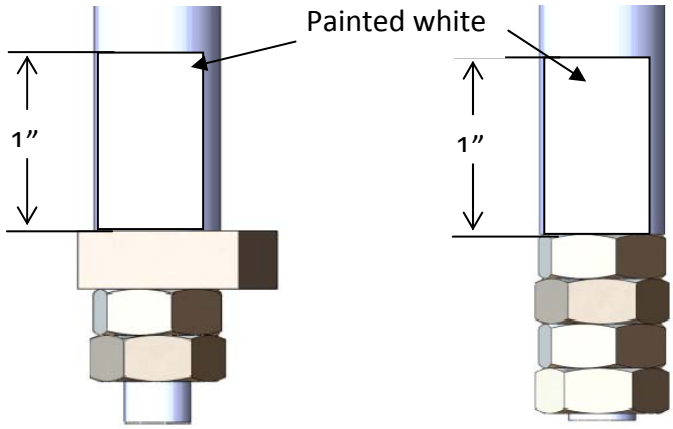


Figure 32: Added mass configurations for final design

The team updated the dynamics models of the system to take into account the changed mass properties of the new design. Both the MATLAB simulation provided by Dr. Hagay Bamberger and the finite element analysis were changed. The results of the MATLAB simulation are shown on the following page for one of the mass configurations of the final design.

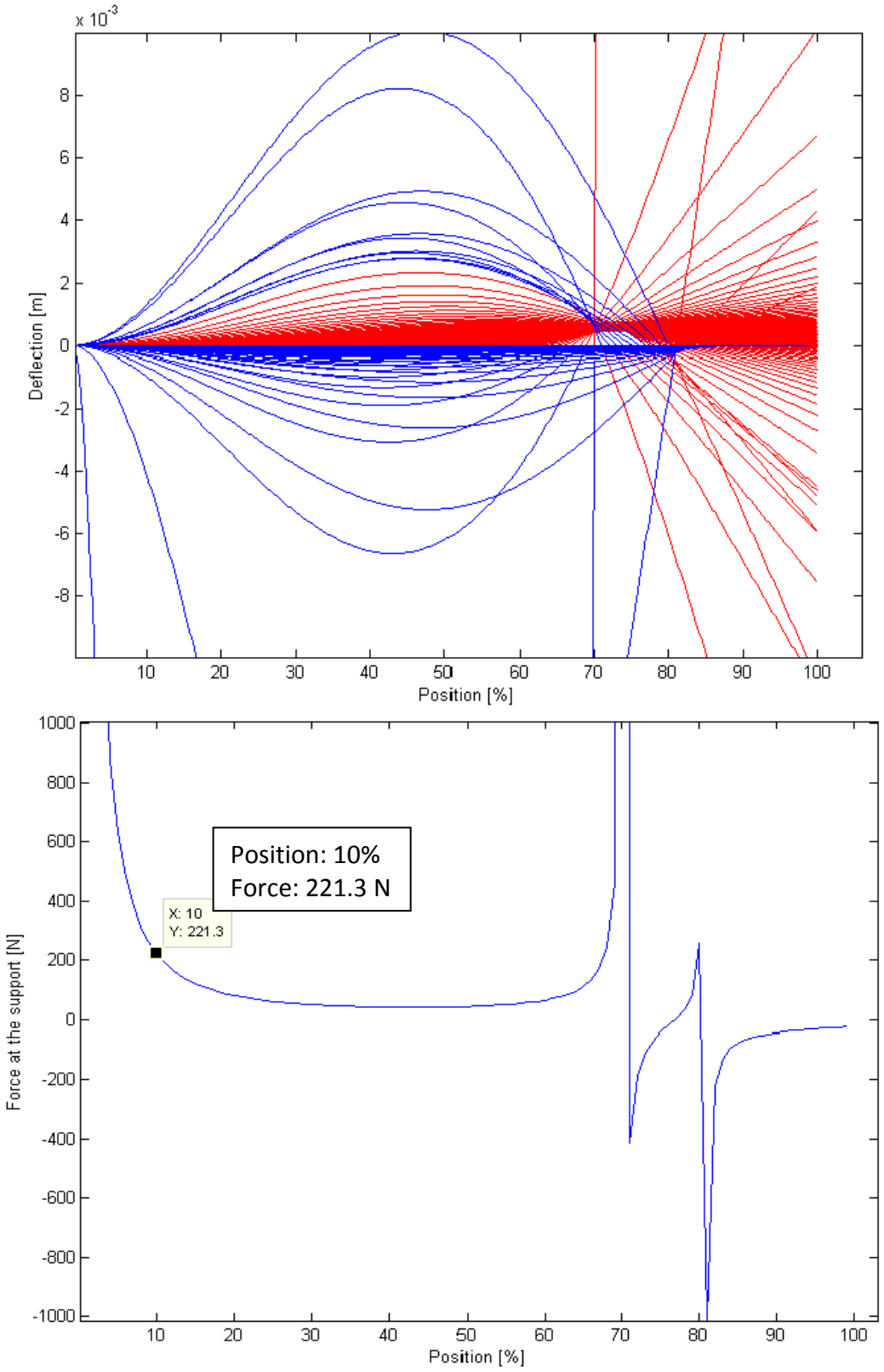


Figure 33: Predicted deflections and forces at support from MATLAB simulation

Bearing Housing: The bearing housing in the final design had the same overall dimensions as the housing described in Design Review Two, but the geometry was altered to house the selected bearing and to facilitate prototyping. It was still made of aluminum as this reduced weight, provided enough rigidity, and decreased machining time. Refer to Appendix G for details about how aluminum was selected as the ideal material for this component. The specific dimensions of the bearing housing were based on the dimensions of the self-aligning ball bearing, the linear stage, and the mill. The self-aligning ball bearing dictated the needed bore diameter and depth, the linear stage dictated the component width, and the mill dictated the length of the component from base to the center of the bore. The external corners of the bearing housing were squared off to allow area for the bearing restraint plate to be fastened and to provide large, flat surfaces on which to locate the edges of the part with the mill. This allowed the team to more easily align the shaft with the bearings during testing.

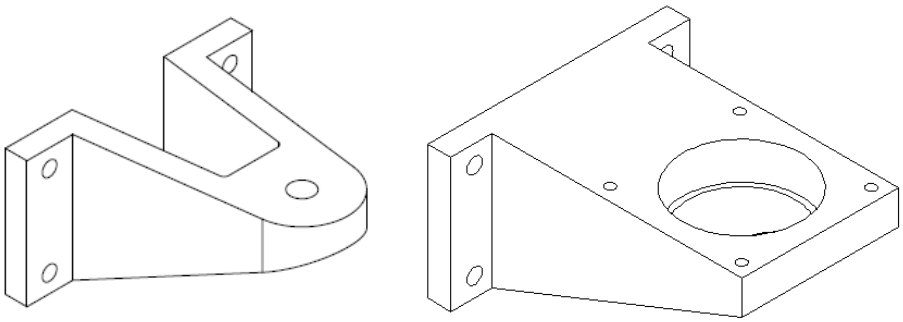


Figure 34: Bearing housing geometries from the alpha design (left) and final design (right)

Finite element analysis was performed on this component to ensure that it would be able to sustain the applied dynamic loads. The team was particularly concerned that the square, concave corners would act as stress risers and cause the component to fail. For this analysis, the team ran the simulations with a combined loading of 1kN thrust force and 1kN radial force applied by the bearing. This loading condition was conservative because the radial force predicted by our MATLAB model was 221N. Observed thrust forces were negligible, since the sleeve bearing allowed the shaft to slide in the axial direction. A screenshot of the FEA mesh is shown below. The area highlighted in red corresponds to portions with a safety factor under 8. The lowest safety factor of this component was determined to be 5.

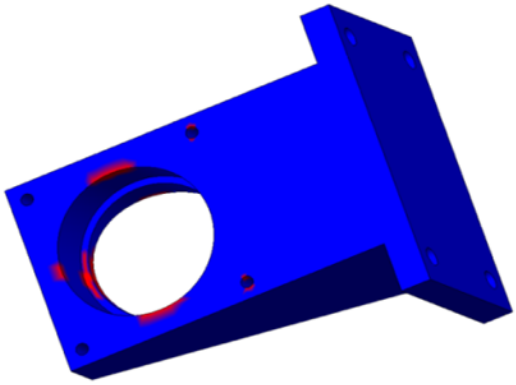


Figure 35: Finite element analysis showed a minimum safety factor of 5 for bearing housing

Bearing Adapters: The bearing adapters were new to the final design. They were needed to clamp the polymer sleeve bearing inside the self-aligning ball bearing. The team chose to manufacture these components out of aluminum, as this would increase machinability and reduce the inertia of the parts. The inner and outer diameters of the adapters were determined by the diameters of the sleeve bearing and self-aligning ball bearing. The lengths of the component were determined by the lengths of the bearings. These lengths were very important, as the team had to ensure that the bearing adapters clamped onto the sleeve bearing before the self-aligning ball bearing. Since the sleeve bearing was made of polymer, the adapters compressed the sleeve bearing slightly until they clamped onto the ball bearing. The upper and lower bearing adapters were held together by four sets of bolts and nuts, as depicted below.

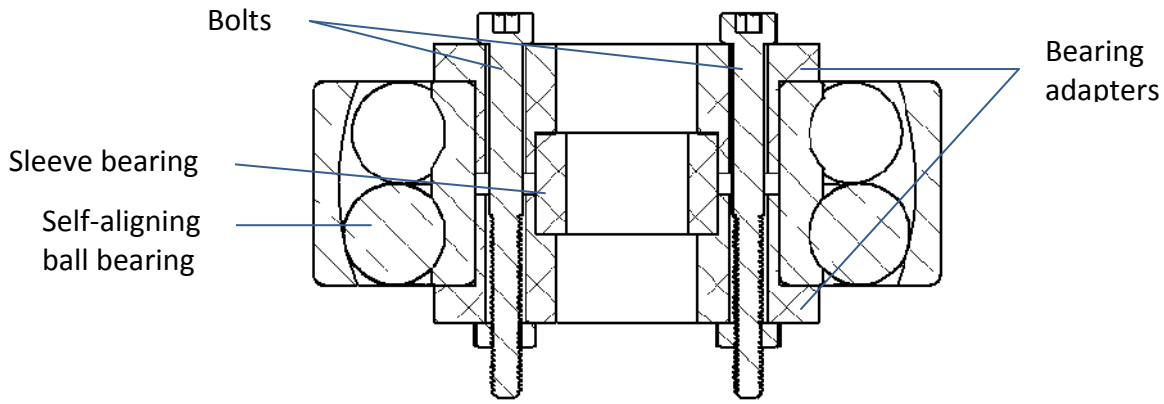


Figure 36: Bearing adapters clamp sleeve bearing inside self-aligning ball bearing

Delrin Polymer Sleeve Bearing: The material of the polymer sleeve bearing was selected since the alpha design was produced for Design Review Two. The team determined that a polymer was more suitable than brass as compressibility and minimized inertia were desired traits. Refer to Appendix G for details about how Delrin was selected as the ideal material for this component. The inner diameter of the sleeve bearing was dictated by the diameter of the shaft to be 12 mm. The team selected the largest available outer diameter so the additional thickness would make it easier to mount the sleeve bearing inside the bearing adapters. The team selected a sleeve bearing length of 10 mm to minimize the area of contact with the shaft as this is the shortest available length of our desired diameter. The team calculated that the bearing would be able to withstand a maximum force of 830 N. This is based on the maximum allowable pressure on a Delrin bearing, P_{max} = 6.89 MPa (1000 psi), and the equation below:

$$F_{max} = P_{max} \cdot d \cdot l$$

where d = the inner diameter of the bearing and l = its length. This is the rating system given by McMaster-Carr, and it assumes that the maximum pressure is distributed across the entire cross sectional area of the shaft ($d \cdot l$). So, the selected sleeve bearing was able to withstand the expected forces of 221N with a safety factor of over 3.5.

Self-Aligning Ball Bearing: The self-aligning ball bearing in the final design was selected based on its dimensions and ability to withstand the largest axial misalignment. The inner diameter of the ball bearing needed to be large enough so that the team could clamp the sleeve bearing inside with the bearing adapters. The selected bearing was also determined to be optimal as it allowed for 2.5° of axial misalignment, the largest of any bearing of this size. The speed and loading were noted to not be driving variables as ball bearings can withstand loads and speeds significantly higher than those expected in the proposed tests.

Bearing Restraint Plate: The bearing restraint plate was new to the final design. The diameter of the hole through the bearing restraint plate was based on the maximum overlap allowed on the outer race of the selected ball bearing. The outer dimensions of the plate and the location of the holes were determined by the geometry of the bearing housing. The bearing housing was designed to allow a clearance between the bearing restraint plate and the bearing housing to clamp the self-aligning ball bearing in place. This is illustrated in Fig. 37 below. The thickness and material were determined by the availability of scrap in the ME Undergraduate Machine Shop, as the stock material for this component were to be obtained from the scrap pile. The forces on this component would be low, so the 5 mm thickness was sufficient.

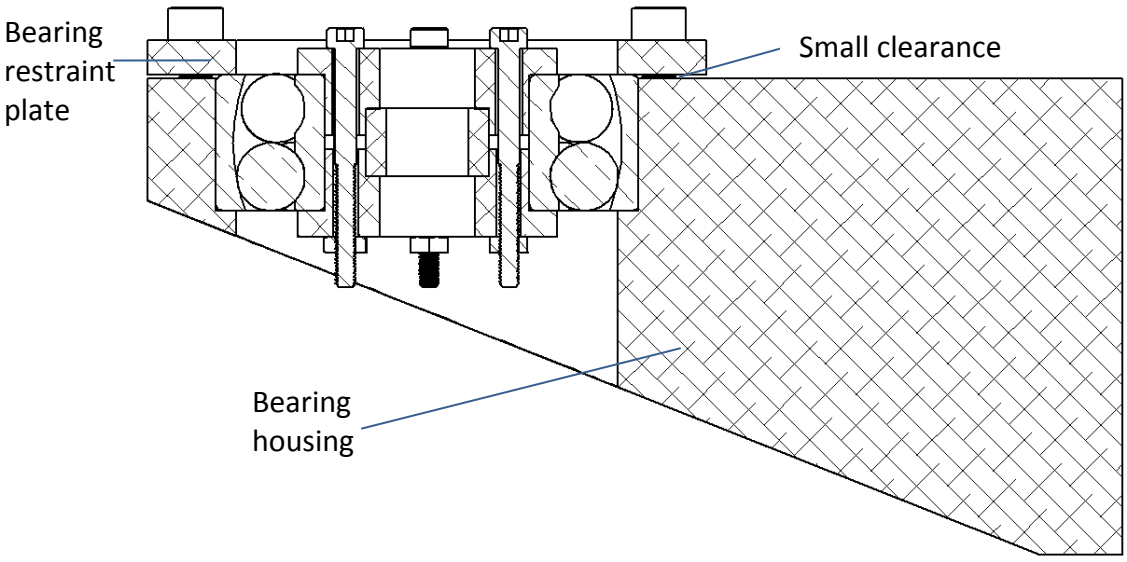


Figure 37: Bearing restraint plate secures bearing assembly into bearing housing

T-Bed Adapters for Linear Stage: The T-bed adapters for the linear stage were designed so that the outer holes would line up with the slots in the T-bed of the mill and the inner slots would line up with the holes in the motor mount. The hole and slot sizes were determined by the fasteners used for the T-bed and the motor mount, respectively. The size of the slot counterbore was determined by the head size of the bolts selected to fasten the adapters to the motor mount. This dimension was designed to be just larger than the head of the bolts so that the slot held the bolt in place and only one tool was needed during assembly to fasten the nut onto the other end. This allowed the team to assemble the motor components without

having to work on the underside of the linear stage. The width and thickness of these adapters were determined by the dimensions of the available aluminum stock at Alro Metals Plus in Ann Arbor.

T-Bed Adapter Plate for Sensor Mounts: The T-bed adapter plate for the sensor mounts was designed to ensure that the sensors could be mounted orthogonally to each other and at the desired standoffs. The geometry of the provided sensors and sensor mounts determined the location of holes in the adapter plate. The driving dimensions in the design of this component were the standoff distances required by the sensors. It is notable that with the adapter plate, each sensor had three translational degrees of freedom while still being orthogonal to the other sensor. The head size of the fasteners determined the depth and diameter of the counterbores on the underside of the adapter plate. The team determined the minimum dimensions of the adapter plate, but the finalized external dimensions were determined by the size of the available scrap material purchased at Alro Metals Plus.

Electrical Design

The electrical and coding aspect of the design consisted of two subsystems that were independent of each other: the sensors and the motor/linear stage.

Sensors: The sensors that P2C2 selected to measure the amplitude of deflection of the rotating shaft were Optimet Conoprobe Mark1.1Bs (one with a 50-mm lens, and one with a 100-mm lens). These laser sensors had the ability to measure displacements at 850 Hz, which made them suitable for the design as the sampling rate was more than twice the maximum frequency at which the shaft was planned to be rotated. Their resolution of 35 microns met the customer requirement on resolution. Additionally, they had a working range of no less than 8 mm and a standoff of no less than 42 mm [13]. Complete specifications for the sensors can be found in Appendix H.

In working with the laser sensors, two safety points were taken into consideration. First, the Conoprobes were Class II lasers. This designation denotes that though regular exposure to the lasers was safe, purposely staring into the beam could cause eye damage. Second, the sensors had dangerous wattage within their casings, with radiation levels reaching 3mW. Because of this, the sensors were not user-serviceable.

To successfully measure the amplitude of displacements over a period of time, P2C2 had begun coded its operation script in C. The pseudocode used to write the script was as follows:

1. Initialize sensors
2. Take x- and y-displacement measurements
3. Send to .csv files with timestamps
4. Repeat 2. and 3. over duration of test.

The finalized code for these steps can be found in Appendix I.

After each test was completed, P2C2 was able to take the data from the two .csv files, import them into Excel, match up the x- and y-displacements from the same times in a spreadsheet, calculate the deflection amplitude of the shaft (see Appendix J), and plot the deflection amplitude vs. time. Because P2C2 modified the sensor programs to enable them to operate both sensors on one computer, they were able to ensure that measurements were synchronized.

Motor/Linear Stage: The motor and linear stage that P2C2 selected to traverse the support along the length of the shaft were the Aerotech BMS60 servomotor and the Aerotech ATS10040 linear stage. This motor/linear stage combination had the ability to traverse 400 mm between 10 percent and 90 percent the length of the shaft. With back of linear stage parallel to ground, it could maintain 50-kg horizontal, 50-kg side, and 25-kg vertical loads – all above what the team saw during operation. It had an encoder which allowed its users to determine its location, accurately and precisely. Additionally, all hardware necessary to run the motor and stage (controller, power cables, encoder cables) were available to them. Complete specifications for the motor and linear stage can be found in Appendix K [14].

In working with the motor/linear stage, various safety points were taken into consideration. First, the users were made aware of pinch points, as the system had many moving parts. The users also considered the temperature of the controller, which, in some applications, could cause burns. The motor/linear stage had to be e-stopped through the program, if needed, (and not by pulling the power cables out) to avoid any mechanical interference with the hard limits. Last, the team had to be careful not to touch the leads of any wires or try to service the controller chassis, as doing so could have caused electrocution [15].

P2C2 programmed the operation script in a language very similar to BASIC. This script allowed the programmer to easily set start and end position, change traversing speed of stage, and create increments of traversing. See Appendix L for motor/linear stage validation code and coding of the design's actual operation.

Integration

To verify whether or not the subsystems discussed would indeed integrate with each other, P2C2 performed a variety of tests. For an initial check, they examined the entire design in CAD and determined where interferences might be an issue during actual installation. They then printed out full-scale templates of the T-bed adapters for the linear stage and the T-bed adapter plate for the sensor mounts and positioned them in the Fadal VMC4020 vertical mill as the actual parts would be positioned. The team put the sensor mounts at their appropriate positions on the template and the linear stage at its appropriate position on the T-bed. Additionally, they moved the mill in all axes to understand what motions might cause interference with the design and conferred with Steve Erskine to establish that they would indeed put a 12-mm shaft in one of the available collets.

Design Meets Customer Requirements

The team feels that they have met all of the customer requirements sans two: ease of operation and the ability to simply support the shaft. The final design was not easily operated due to the fact that two computers were used to operate the mechanism. Although the team modified the code to be able to operate both sensors on one desktop computer, adding motor operation to the desktop would have slowed measurements down, so the team kept the laptop provided by the laboratory to control the linear motor.

Also, the ability to simply support the shaft was not possible. This concept is generally only used in theoretical calculations and is nearly impossible to simulate. This was especially true for the support in the final design. The support needed to allow for axial slip, axial misalignment and rotation. The team did their best to provide a support that mimicked a simple support as much as possible. However, the support in the final design did not contact the shaft at a single point; it contacted it over a surface. Therefore, it was not a perfectly simple support.

Bill of Materials

Purchased Components

The purchased components included two bearings (a self-aligning ball bearing and a polymer sleeve bearing) and various fasteners. See Table 6 for a list of components.

Purchased Stock Material

A large portion of the final design was made up of manufactured parts from raw material stock. Raw material was necessary to make the shaft, the bearing housing, the bearing adapter pieces, the bearing restraint plate, the T-bed adapter plate for the sensor mounts, and the T-bed adapters for the linear stage. See Table 7 for a list of purchased stock material.

Donated Components

Four components of the final design were donated to the team from the Reconfigurable Manufacturing Laboratory. These components were the linear stage and motor, the laser point sensors, the sensor mounts, and the Fadal vertical milling machine. See Table 8 for a list of donated components.

Part Name	Qty	Part/Model Number	Supplier	Size	Function
Self-aligning ball bearing	1	SKF: 2206 ETN9	BDI Inc	OD = 62mm B = 20mm	Allow rotation and axial misalignment
Polymer sleeve bearing	1	2640T16	McMaster-Carr	ID = 12mm B = 10mm	Allow axial translation of support along shaft
T-slot nuts	4	90526A604	McMaster-Carr	Thread = 1/2”-13	Fasten T-bed adapters for linear stage to T-bed
Socket cap screw	4	90128A720	McMaster-Carr	Thread = 1/2”-13 L = 2”	Fasten T-bed adapters for linear stage to T-bed
Cap screw	4	91309A634	McMaster-Carr	Thread = 3/8”-16 L = 2-1/2”	Fasten T-bed adapters to linear stage
Nut	12	92673A125	McMaster-Carr	Thread = 3/8”-16	Fasten adapters to linear stage and sensor mounts
Socket cap screw	4	91292A033	McMaster-Carr	Thread = M3-0.5 mm L = 35 mm	Fasten two bearing adapters halves together
Nut	4	90592A009	McMaster-Carr	Thread = M3-0.5 mm	Fasten two bearing adapters halves together
Socket cap screw	8	90128A632	McMaster-Carr	Thread = 3/8"-16 L= 2"	Fasten sensor mounts to T-bed adapter plate
Socket cap screw	4	030699826483	Home Depot	Thread = M6-1.0 mm L = 16 mm	Fasten bearing housing to linear stage
Socket cap screw	4	030699713585	Home Depot	Thread = #10-24 L = 3/4”	Fasten bearing restraint plate to bearing housing
Nut	4	030699319015	Home Depot	Thread = 1/4”-20	Secure mass to end of shaft
Locking nut	1	030699192618	Home Depot	Thread = 1/4”-20	Secure mass to end of shaft

Table 6: List of purchased materials

Part Name	Qty	Material	Part/Model Number	Supplier	Stock Size	Function
Shaft	1	Aluminum	4634T17	McMaster-Carr	D = 12mm L = 1830mm	Rotate in mill and allow dynamic testing
Bearing housing	1	Aluminum	-	Alro Metals Plus	3.5”x5”X8”	Secure self-aligning ball bearing to linear stage
Bearing adapters	2	Aluminum	8974K181	McMaster-Carr	D = 1-1/2” L = 12”	Clamp polymer sleeve bearing inside self-aligning ball bearing
Bearing restraint plate	1	Aluminum	-	ME Undergraduate Machine Shop	0.25”x5”x22.25”	Secure self-aligning ball bearing inside bearing housing
Sensor mounting plate	1	Steel	-	Alro Metals Plus	5/8”x5”x23”	Allow sensor mounts to be connected to T-bed
T-bed adapters for linear stage	2	Aluminum	-	Alro Metals Plus	1”X4”x15.5”	Allow linear stage to be connected to T-bed

Table 7: List of stock material purchased

Part Name	Qty	Manufacturer	Model Number	Function	Notes
Linear stage	1	Aerotech	ATS10040	Provide linear movement of support along shaft	Mount for linear stage provided
Servomotor	1	Aerotech	BMS60	Move linear stage platform	Controller and cables provided
Laser point sensors	2	Optimet	Conoprobe 1.1B	Measure amplitude of vibrations of shaft	Cables provided
Sensor mounts	2	-	-	Hold sensors horizontally and orthogonally	Each mount allows three degrees of freedom
Vertical milling machine	1	Fadal	VMC 4020	Rotate unbalanced shaft and provide workspace	

Table 8: List of donated components

Initial Fabrication Plan

The following section details the manufacturing plan and assembly process.

Manufacturing

All of the parts were either manufactured in the ME Undergraduate Machine Shop or in the Reconfigurable Manufacturing Laboratory under the supervision of Steve Erskine. All parts were made of steel or aluminum. According to *Machinery's Handbook* [16], the tool speeds for these two materials are 500 and 150 feet per minute, respectively. Since these speeds are in translational motion, they had to be converted to rotational speeds unique to every tool used. The rotational speed of any tool can be found using the equation:

$$RPM = \frac{12V}{\pi D}$$

where V = the translational speed from the *Machinery's Handbook* and D = the tool diameter or part diameter in the case of turning operations. Note that most stock materials were available to the team in English units while design CAD drawings were done in metric units to stay consistent with ordered parts.

Bearing Adapters: The bearing adapter CAD drawing is shown in Appendix F. This part was manufactured from a cylindrical aluminum stock material of outer diameter 1.5 inches. The material was purchased from McMaster-Carr.

Step	Operation	Machine	Tool	Speed	Notes
1	Cut Outer Diameter	Lathe	Turning Tool	350 RPM	Hold in 3 jaw chuck
2	Drill Center Hole	Lathe	1/2" Drill	350 RPM	
3	Bore Out Inner Holes	Lathe	Boring Bar	350 RPM	
4	Cut Outer Diameter	Lathe	Turning Tool	350 RPM	
5	Cut Part From Stock	Lathe	Facing Tool	350 RPM	
6	Cut Chamfer	Lathe	Turning Tool	350 RPM	Part must first be reversed
7	Drill 4 Holes	Mill	3/16" Drill	3000 RPM	

Table 9: Manufacturing plan for bearing adapters (2)

Bearing Restraint Plate: The bearing restraint plate CAD drawing is shown in Appendix F. This part was manufactured from a piece of scrap aluminum sheet stock obtained from Alro Metals in Ann Arbor.

Step	Operation	Machine	Tool	Speed	Notes
1	Cut Inner Hole	Water Jet Cutter	N/A	N/A	
2	Cut Outer Dimensions	Water Jet Cuter	N/A	N/A	
3	Drill 4 Holes	Mill	¼" Drill	3000 RPM	

Table 10: Manufacturing plan for bearing restraint plate

Bearing Housing: The bearing housing CAD drawing is shown in Appendix F. The part was manufactured from scrap aluminum material obtained from Alro Metals in Ann Arbor. The stock material has the dimensions of 143 x 70 x 125 mm.

Step	Operation	Machine	Tool	Speed	Notes
1	Face Left Side	Mill	½" End Mill	3000 RPM	
2	Face Right Side	Mill	½" End Mill	3000 RPM	Part must first be reversed
3	Drill 4 Holes	Mill	No. 25	3000 RPM	Part must first be properly positioned
4	Tap 4 Holes	Hand Tap	#10-24	N/A	
5	Mill Bearing Set Hole	CNC Mill	½" End Mill	3000 RPM	Part must first be properly positioned
6	Mill Slanted Face	CNC Mill	½" Ball Mill	3000 RPM	Part must first be properly positioned

Table 11: Manufacturing plan for bearing housing

T-Bed Adapters: The T-bed adapter CAD drawing is shown in Appendix F. The part was manufactured from scrap aluminum material obtained from Alro Metals in Ann Arbor. The stock material must first be cut in half lengthwise and cut to length (320 mm).

Step	Operation	Machine	Tool	Speed	Notes
1	Cut Lengthwise	Band Saw	N/A		
2	Mill Slot Holes	Mill	3/8" Ball Mill	3000 RPM	
3	Counter Bore Slot Holes	Mill	9/16" End Mill	3000 RPM	
4	Drill End Holes	Mill	7/16" Drill	3000 RPM	

Table 12: Manufacturing plan for T-bed adapters (2)

Sensor Mount Adapter Plate: The sensor mount adapter plate CAD drawing is shown in Appendix F. The part was manufactured from scrap steel material obtained from Alro Metals in Ann Arbor.

Step	Operation	Machine	Tool	Speed	Notes
1	Drill 8 Holes	Mill	½" Drill	500 RPM	
2	Counter Sink 8 Holes	Mill	7/8" End Mill	500 RPM	

Table 13: Manufacturing plan for sensor mount adapter plate

Shaft: The shaft CAD drawing is shown in Appendix F. The stock material was obtained from McMaster-Carr. The shaft already has the desired outer diameter along most of its length, however, the last threaded portion was manufactured.

Step	Operation	Machine	Tool	Speed	Notes
1	Turn Outer Diameter	Lathe	Turning Tool	350 RPM	
2	Thread Outer Diamter	N/A	¼"-20 Thread Die	350 RPM	

Table 14: Manufacturing plan for shaft

Added Mass: The added mass CAD drawing is shown in Appendix F. The stock material was obtained from the student machine shop in the basement of the GG Brown Building.

Step	Operation	Machine	Tool	Speed	Notes
1	Cut Outer Dimensions	Mill	½" End Mill	3000 RPM	
2	Drill Hole	Mill	No. 7 Drill	350 RPM	
3	Thread Hole	N/A	¼"- 20 Tap	N/A	

Table 15: Manufacturing plan for added mass

Pros and Cons of Selected Manufacturing Processes: Using manual and CNC mills and lathes had pros and cons over using rapid prototyping methods such as stereolithography.

Pros

- Tight tolerances could be achieved
- Use of more structurally rigid materials was possible
- There were opportunities for adjustments while manufacturing
- Enjoyment of physically working on a part

Cons

- More time consuming
- Extra work for the team
- Lack of experience using machines caused minor setbacks

These pros and cons considered, the team believed that manufacturing the components out of raw materials using manual and CNC mills and lathes was the proper course of action.

Mechanical Component Assembly

Several components of the final design were assembled and kept assembled at all times while other components needed to be assembled every time the mechanism was placed in the Fadal mill. The sensors and linear stage were not adjusted or moved after they were positioned and aligned as needed. The team booked the mill for final testing so that the mechanism need not be removed. All assembly took place in the Reconfigurable Manufacturing Laboratory.

Several components of the final design remained assembled at all times. These components were the sensors on the mounts, the bearing adapters, and the bearing housing. The sensors attached to mounts to which they had been previously attached. Small screws were used to attach the sensors to small adjustment platforms that were then attached to large steel stands. These stands were then screwed onto the adapter plate that the team made to ensure that the sensors were orthogonal to each other at all times. This process is illustrated in the images below:

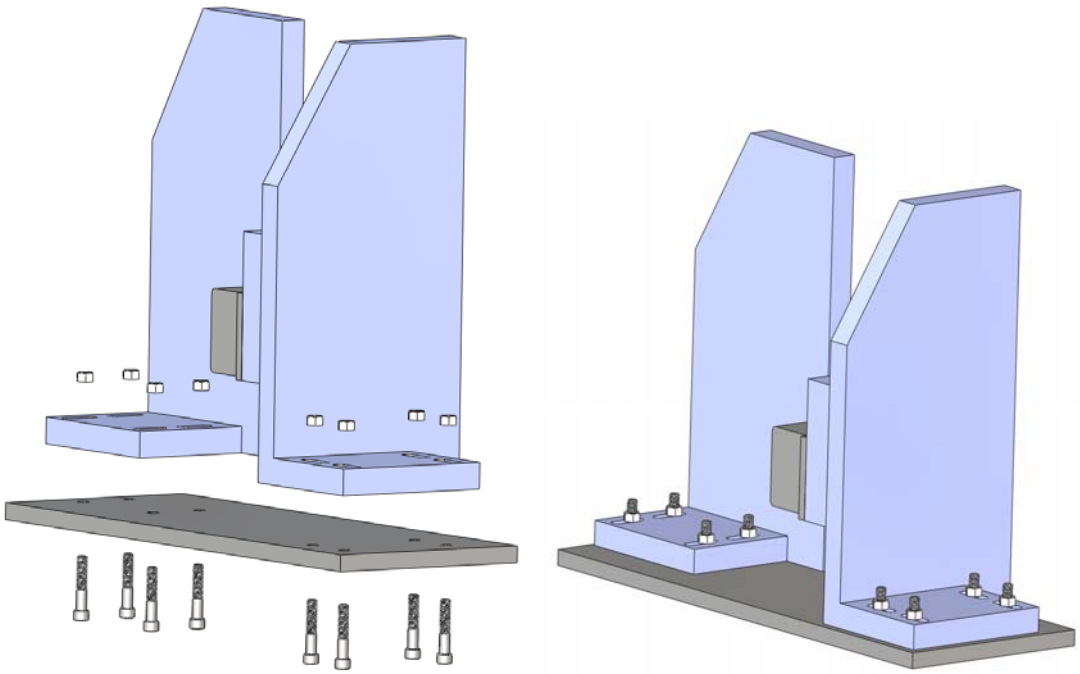


Figure 38: Attachment of sensor mounts to T-bed adapter plate

The bearing housing adapter was used to keep the sleeve bearing inside the self-aligning ball bearing. In order to assemble this set up, the sleeve bearing was first placed inside one of two bearing adapters. This assembly was then placed inside the inner race of the self-aligning ball bearing. Finally, the second of two bearing adapters was inserted to close off the other side of the self-aligning ball bearing. Four screws and four nuts were then used to keep this whole assembly together. The bearing assembly was then placed inside of the bearing housing. A plate was designed that held the bearing in its hole and secured to the bearing housing using four screws. The bearing housing then attached to the platform on the linear motor with four screws. This whole assembly (the bearing assembly in the bearing housing and the housing attached to the linear motor) remained assembled at all times. This process is illustrated in the images on the following page:

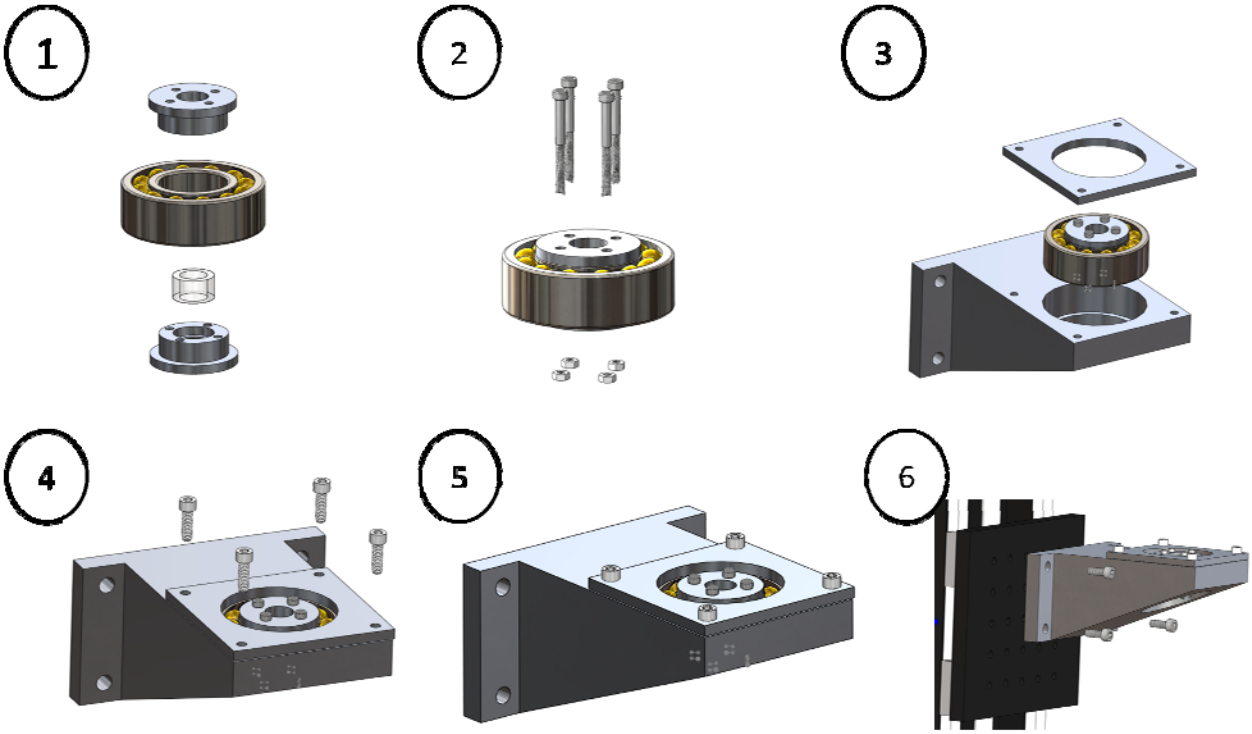


Figure 39: Assembly of bearing support and attachment to linear stage

Aligning the linear stage and sensors with the linear stage was a critical part of ensuring that the support translated the shaft freely and that measurements were taken properly and orthogonally. The following steps depict how the linear stage was mounted into the mill and how the sensors were placed in the proper location.

1. The first step was to install the linear motor. Adapter brackets were screwed to the bottom of the linear motor that allowed the linear motor to attach securely to the T-bed within the Fadal mill. Then the motor was attached to the T-bed using these adapter brackets. It was crucial that the motor be perfectly aligned on the T-bed. That is to say, the sides of the motor should be parallel with the slots in the T-bed. This allowed for accurate alignment of other components later on. The team ensured that the stage was vertical and parallel to the shaft by using the mill spindle and a magnetic measuring gauge. Fine adjustments to make the stage vertical were made by adjusting bolt tightness and increasing stainless steel shims between the linear stage and the adapters. This method ensured that it was vertical to within 0.002 in. This measuring gauge was also used to make sure the face of the linear stage was parallel to the y-axis of the mill to facilitate easy location of the center of the bearings.

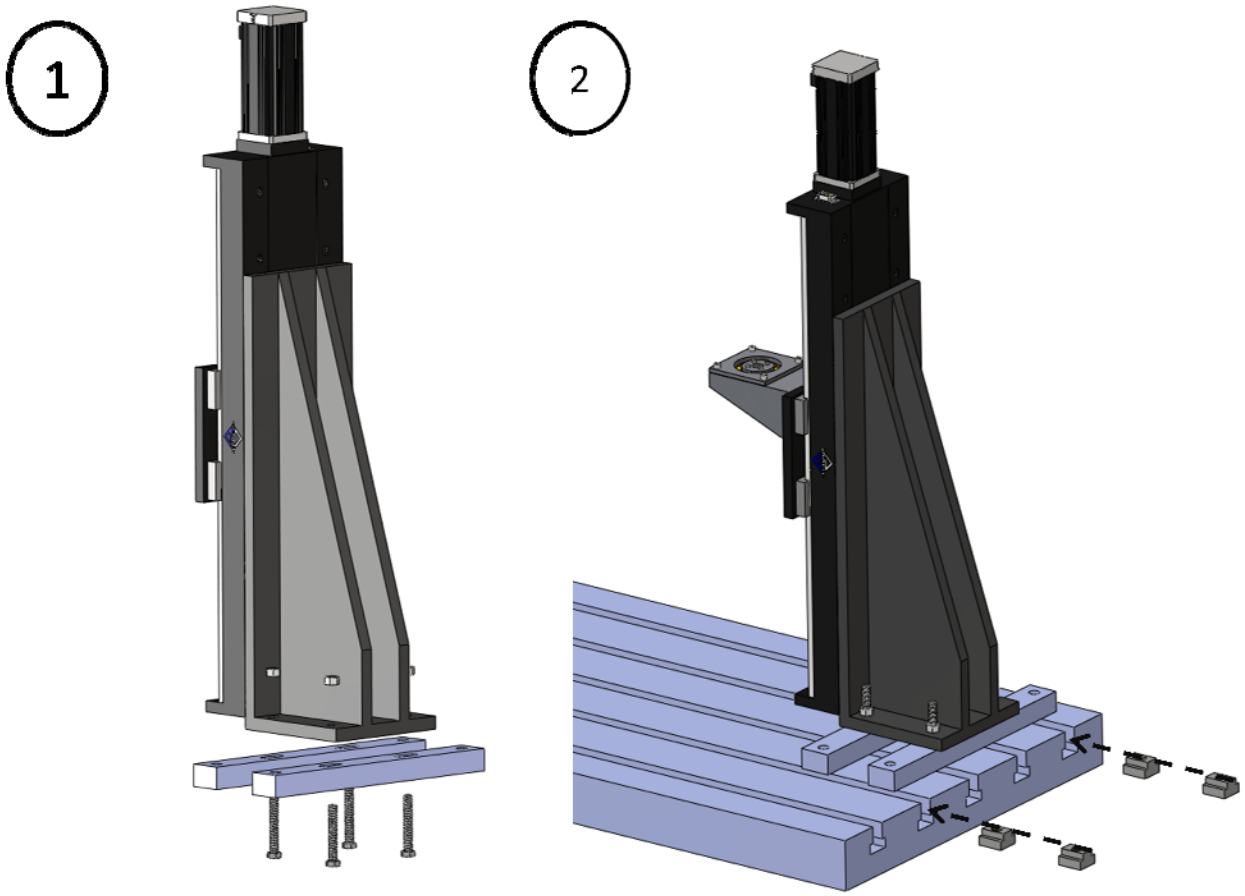


Figure 40: Attachment of linear stage to T-bed adapters and attachment to T-bed

2. The second step was to home in the Fadal mill head using an electronic edge finder. The mill needed to have a home position in an appropriate position so that the location of the center of the bearing can be determined. The mill was homed in on the edges of the bearing housing using an electronic edge finder. The edges were designed square so that this process may be made simpler. The team set the zero coordinates for x and y to correspond to this corner. The team then used the distances used on the CNC to originally produce the bore to hold the bearing to determine the offset from these zero coordinates to the center of the bore. After the center of the hole was located, the team noted this location and used it each morning to align the mill to the proper location.

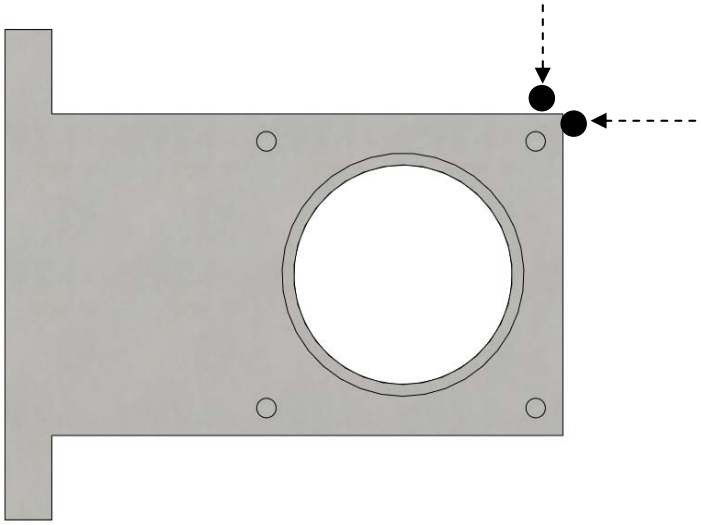


Figure 41: Use edge finder to locate edges of bearing housing and zero mill

3. The mill head was then be moved upward and out of the way so that the rod could be placed properly in the bearing assembly.
4. The mill head was then be moved to a position directly over the bearing hole and brought downward toward the bearing assembly.

5. The rod was then installed into the mill collet so that the exposed length of the shaft was 20 in and the distance from the support at its highest position to the collet of the mill was 2 in (10% of the length of the shaft). These distances were measured manually with a set of calipers. The shaft was left in the collet after it was initially inserted, and the collet was removed from the spindle each day as needed. The needed height of the mill spindle was noted when it was initially determined to be used on subsequent days.

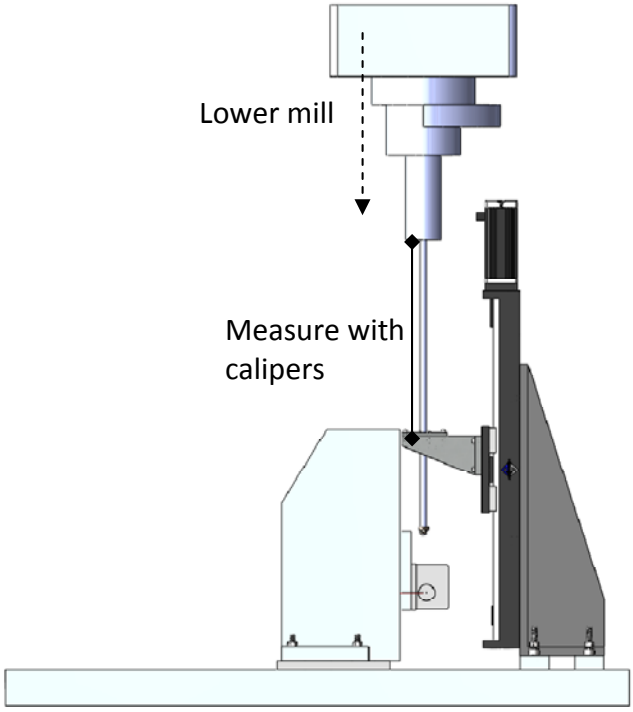


Figure 42: Lower mill head so correct position

6. The sensors and sensor mounting plate were then positioned to properly measure the displacements. Both point lasers hit the center of the rod and were close to the end.

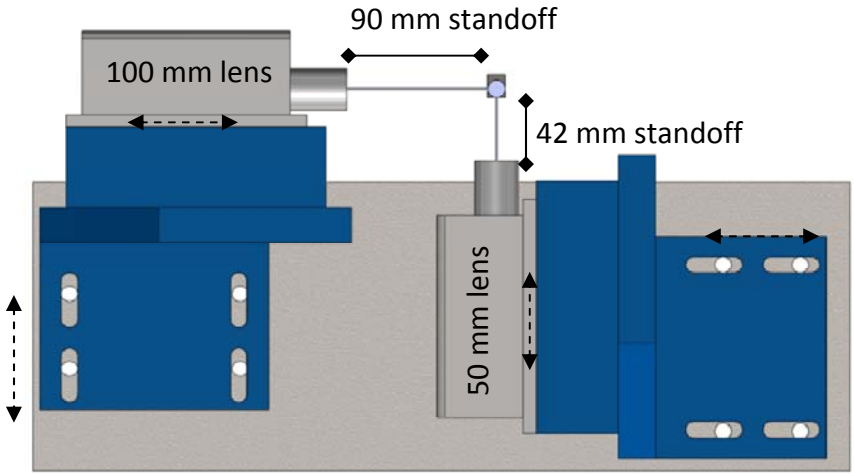


Figure 43: Adjust sensor positioning to obtain desired height and standoffs

7. Finally, the off center mass was screwed onto the end of the shaft along with the two set screws. Tests were also performed without this mass attached.

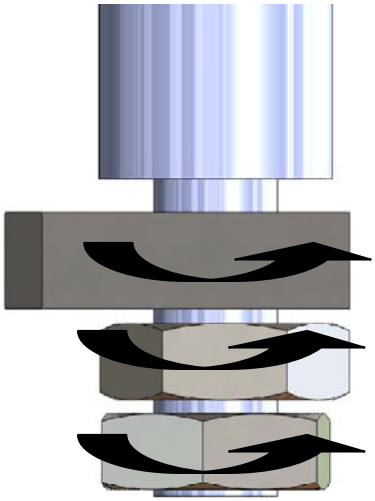


Figure 44: Tighten mass and nuts with torque wrench to specified tightness

Electrical Component Assembly and Operation

This section summarizes the steps used to setup and operate the electrical components included in the design.

Sensor setup: To ensure that the sensors were indeed reading at the correct frequency and positioned exactly at the center of the shaft, the team took the following steps for each sensor and controller combination:

1. Making sure its switch was still in "Off" mode, the team plugged the controller into a power source, the desktop's built-in DB25 connection, and the sensor.

2. Once effective measures had been taken to make sure no one could stare directly into the laser, they activated the sensor by toggling the switch on the controller to "On" mode and launched the Conoprobe-specific Scanner software provided by Optimet.

3. The left side of the screen has a range bar that shows whether or not an object is in range. The sensor was adjusted parallel to ground until the indicator bar was in the green area of the range bar, denoting good proximity.

4. The setting for frequency was adjusted to read 850 Hz.

5. The settings for power and fine power were adjusted until the default graph showed no signs of saturation and the signal quality was maximized.

6. The right side of the screen had a drop-down menu with options for types of graphical output. The Frequency option was chosen, launching a graph of Power vs. Frequency.

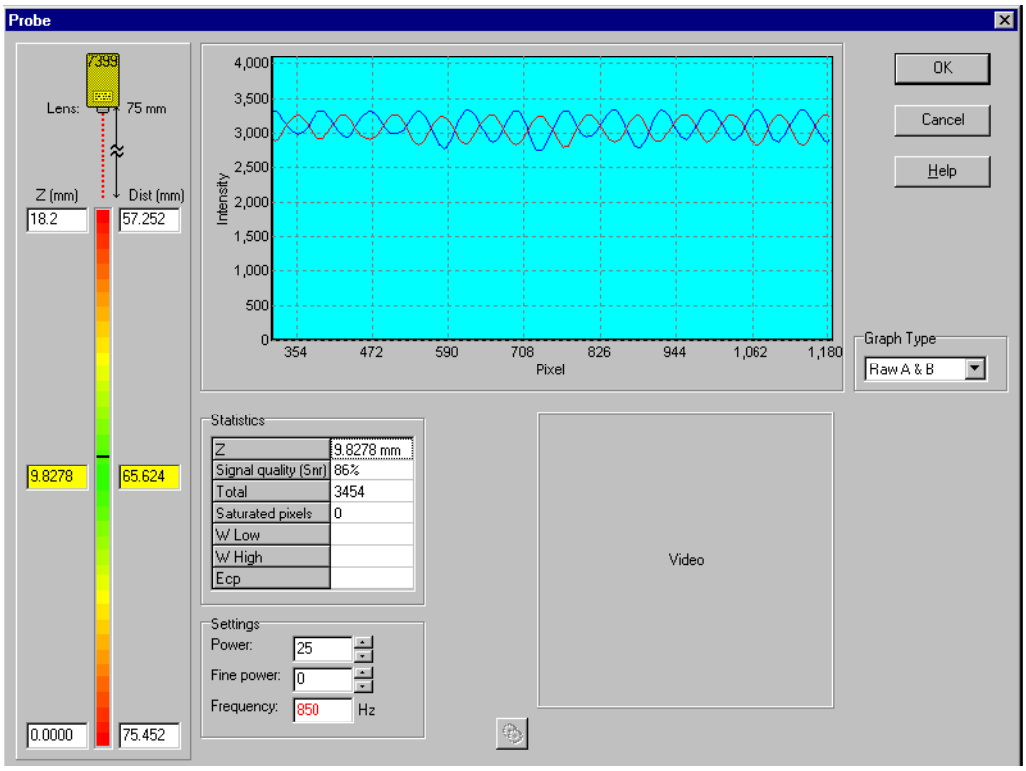


Figure 45: Scanner software

7. A peak around the middle of the Power vs. Frequency graph indicates that the part of the object closest to the sensor is in the middle of the measuring spot. The sensor was adjusted parallel to ground until this was the situation.

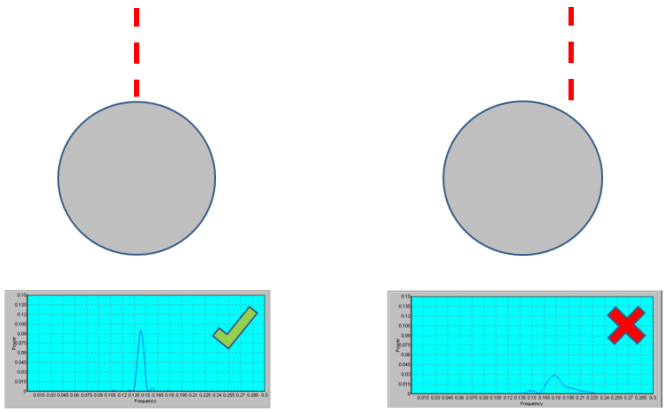


Figure 46: Proper set-up and its corresponding graphical output vs. incorrect set-up and its corresponding graphical output – Circles indicate cross-section of shaft.

Motor Operation: The motor system consisted of a linear stage, a motor, a controller, and a laptop. The following steps were taken:

1. The team plugged the controller into the laptop's USB connection and into a power source.

2. They plugged the motor cables into the serial connections on the controller.

3. Once everyone is clear of the linear stage and motor, they launched the Aerotech IDE 2.52 software.

4. They clicked on dropdown menus along the top of the screen and selected the option allowing them to connect to the network of the controller.

5. The team clicked on dropdown menus along the top of the screen to make sure that the parameters listed for the motor matched up with the motor's actual parameters. If they did not, they created a new parameter set and applied it.

6. They loaded the code for motor driving.

7. They compiled and ran the code (complete code can be found in Appendix L).

8. The team pressed the "Home" button on the screen to move the motor back to its starting location on the shaft.

Sensor Operation: The sensor system consisted of two sensors, two controllers, and a desktop. The following steps were taken:

1. They plugged one controller into the desktop's built-in DB25 connection, and the other controller into the desktop's PCI card DB25 connection. Making sure their switches were still in "Off" mode, the team plugged both controllers into a power source.

2. They plugged each sensor into a controller.

3. Once effective measures had been taken to make sure no one could stare directly into the lasers, they activated the sensors by toggling the switches on the controllers to "On" mode.

4. The team launched Visual C++. They loaded the source code, header code, and libraries for multiple probe operation (Complete source code and header files can be found in Appendix I).

5. They noted the intended location of the .csv files to be created in the source code.

6. They compiled, built, and ran the workspace.

7. They enter the port addresses of the DB25 connections when prompted. Measurement initialized as soon as data started streaming.

8. The team pressed the spacebar of the desktop when measurement time elapsed had been fulfilled.

Pictures of the assembled mechanism, along with the electrical components, can be found in Appendix N.

Prototype Description

Because the team's project was limited to calculating the dynamics of a shaft of specific dimensions and material in a specific mill, their product was a unique design not meant for mass production. Therefore, though the team's work may be used as the starting point off which to create designs for shafts of different dimensions and material, the team and their sponsor deemed a prototype unnecessary.

Validation Plan

The team created a test plan composed of five parts: component tests, impact tests, static support tests, dynamic support tests, and dynamic support tests above the natural frequency. This sequence of tests evaluated each component of the final design to ensure safety, measure accuracy, and confirm that the design meets engineering specifications set described on page 10 (copied below for convenience). The complete test procedure can be found in Appendix M.

Engineering Specifications

The engineering specifications described on page 11 are as follows:

- Resolution of vibration amplitude measurement is within 10 percent of the amplitude
- Resolution of moving support location is less than 5mm
- Fixture reduces vibrations at supported point by 75 percent
- Moving support traverses from 10 percent to 90 percent of L_{shaft}
- Sampling rate is more than twice the frequency of shaft rotation
- Sensor is within 2 percent of L_{shaft} from free end

Component Testing

The component tests performed were written such that they showed that the mechanism designed by the team meets all engineering specifications.

Sensor tests confirmed that the Optimet ConoProbe Mark 1.1B sensors measure displacement with a resolution of less than 9 percent of the magnitude of displacement. The team compared the reading from the sensor to the known displacement of an object, measured with ruler. It was determined from these tests that the ConoProbes measured displacement within 10 percent of the actual displacement. The sampling rate of the lasers was not to be measured because the manufacturer's specifications stated that the laser could operate at up to 850Hz, well over two times the maximum frequency expected of the shaft.

The tests also confirmed that the motor could move the support so that it could traverse 80 percent of the length of the shaft, and that the resolution of the moving support location was fewer than 5mm. Preliminary testing showed that the motor code read in unknown units. Testing showed a linear relationship between these unknown units and millimeters with a conversion factor of 1.603. Further tests confirmed that, using this factor, the motor moved to within 1.2 mm of the desired location.

The team then tested the shaft assembly as one of several safety checks. Two different shaft configurations were tested: one with three nuts and one with two nuts and an offset mass weighing 9g. The team slowly spun the shaft up to speed to confirm that the nuts and masses at the end of the shaft would not loosen.

Finally, the components were tested together. The team tested that the support traversed along the shaft and that the bearings and restrain plates worked as expected. The shaft did not bend while the motor was moving and the motor did not bind. Additionally, the team tested that the shaft spun inside the bearings while the support was stationary. It was noted that, at slower speeds and with smaller displacements, the shaft spun inside the sleeve bearing. At higher speeds and with larger vibration amplitudes, the inner race of the ball bearing rotated with respect to the outer race. It was decided that the location of rotation should not affect the results of the test. The team then ensured that the support was able to traverse the length of the shaft while the shaft was spinning. This test confirmed that the design was able to function for the intended purpose.

There was one requirement that the team felt they were unable to test given the limitations of the equipment, the budget, and the time. This specification stated that the support must reduce the vibrations experienced by the shaft at the point of contact by at least 75 percent. However, given the robustness of the design and the area of contact with the shaft, the team felt that their design would meet this requirement.

The team believes that this series of tests confirmed their design meets all other engineering specifications.

Impact Testing

The impact test determined the natural frequency while the support was at various locations along the length of the shaft. The support was moved to a given location and the stationary shaft was hit with a rubber mallet. The displacements measured with the sensors were used to determine the natural frequency. The team compared this measured frequency to the frequency determined in finite element analysis. The differences are shown in Figure 47.

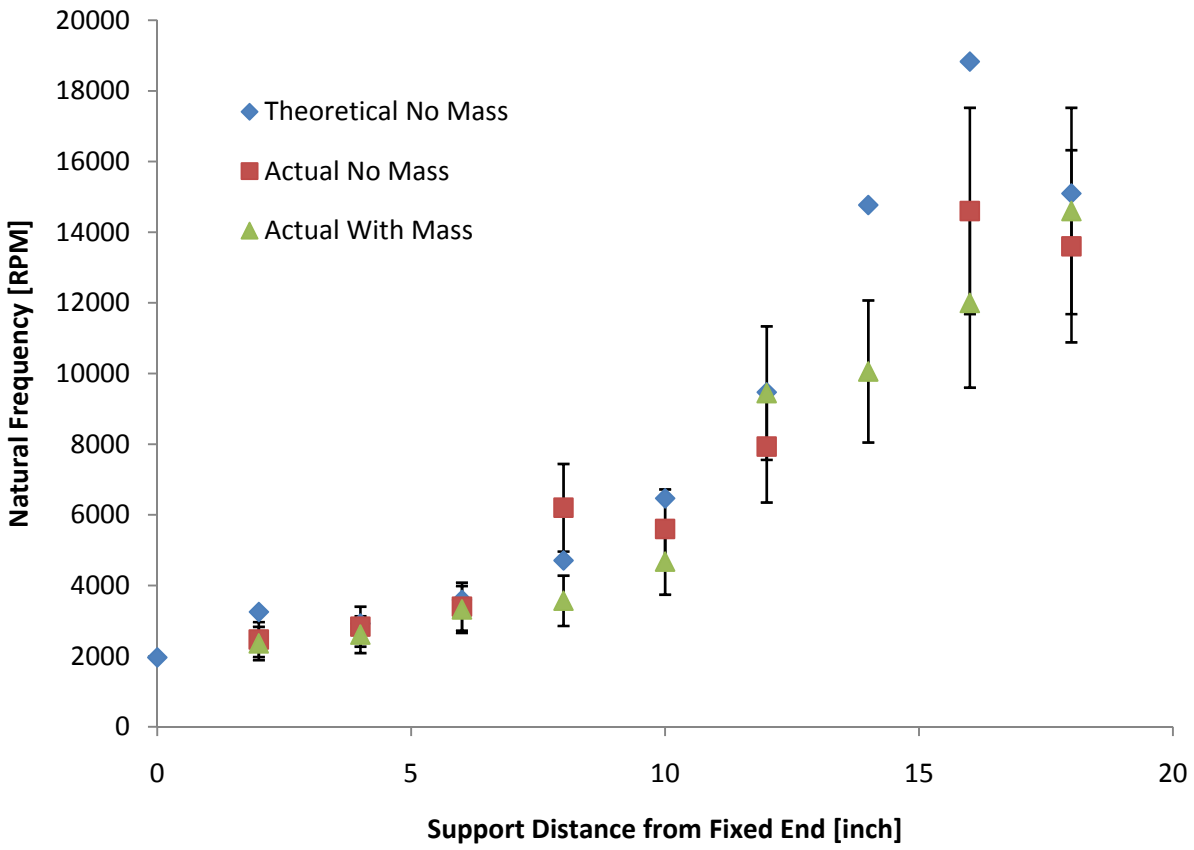


Figure 47: Observed natural frequencies are below expected natural frequencies

Most measured frequencies were equal to the theoretical values within error. The differences were larger as the support moves further down the length of the shaft. These larger differences may be attributed to the difficulty in hitting the shaft when the support was close to the end. Also, the vibrations were smaller and thus, more difficult to measure.

Static Support Testing

The static support test measured the displacement at the end of the shaft while the shaft was rotating and the support was stationary. The test procedure called for two tests: one with three nuts and one with an offset mass and two nuts. Timing and efficiency concerns led the team to change the order of the test procedure outlined in Appendix M during the test. With the approval of the project sponsor, the team tested the shaft with the offset mass before the shaft without an offset mass. Using this setup, testing at 500RPM and at 2000RPM was completed. The results for 500RPM are shown in Figure 48.

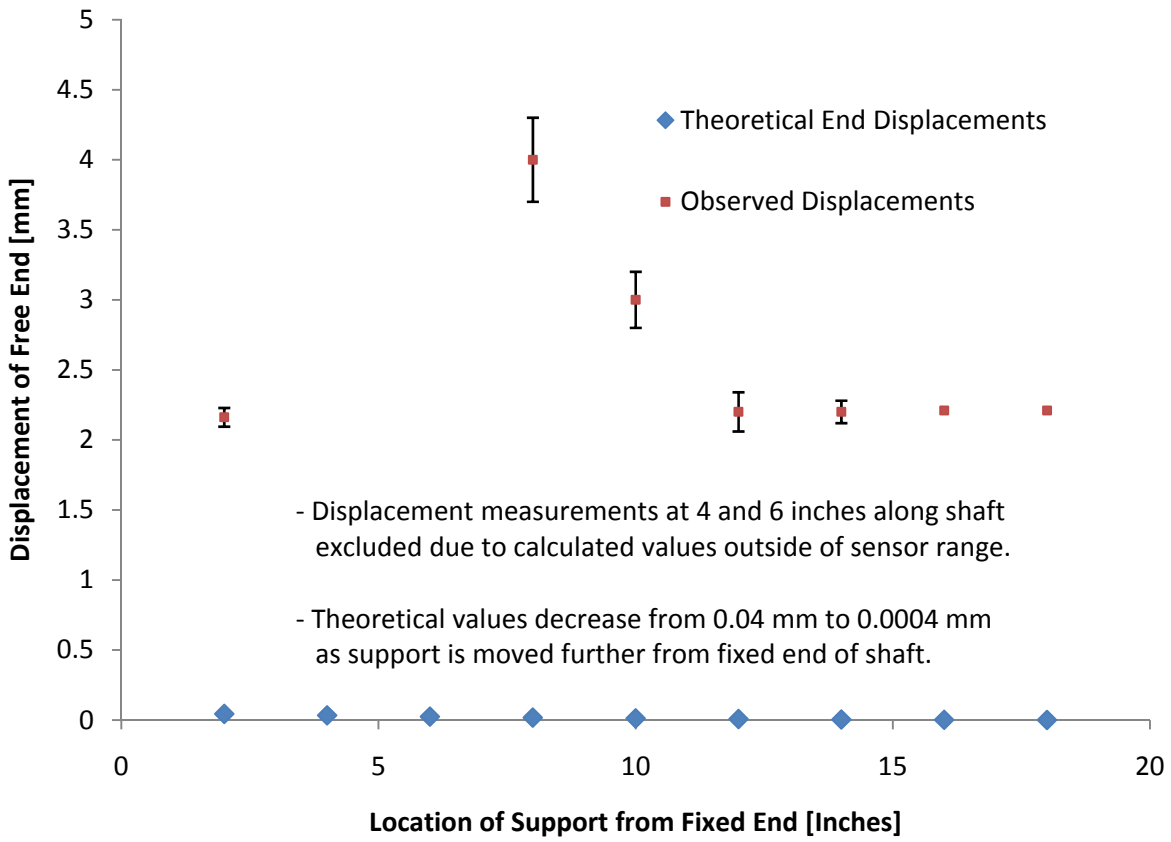


Figure 48: Observed displacements are two to three orders of magnitude greater than the expected displacements

As shown in Figure 48, the displacements measured in testing were far greater than the displacements predicted from MATLAB analysis. The amplitude of vibration while the support was between four and six inches from the fixed end was greater than the range of the sensors, thus the data has been omitted. Similarly, at 2000RPM the amplitude of vibrations for all positions was greater than the range of the sensors. The data is not usable for this analysis.

There are several possible explanations for the differences shown here. The shaft chosen by the team was relatively inexpensive given the budget constraints and thus lacked uniformity and straightness. Additionally, the dynamics of the support and the contribution from the vibrations of the machine are unknown.

During the next test, the speed was gradually increased. At approximately 4000RPM, the team noticed that the amplitude of displacement was rather large. The speed was increased to 5000RPM and the amplitude settled down. Measurements were taken while the support was ten inches down the length of the shaft. As the support was moved to the twelve inch position, the system moved back into the resonant range. Shortly after the support stopped moving, the shaft experienced permanent deformation, preventing further testing. This failure is described in further detail in the section titled Failure Description.

Dynamic Support Testing

Given that the shaft experienced permanent deformation during the static support tests, the team was unable to continue testing. Should this research be continued at a later time, the proposed test procedure is outlined in Appendix M.

Failure Description

During the testing and validation process, failure of the prototype occurred. Pictures of this failure can be found in Appendix O. The following section describes the test activities at the time of and presents a hypothesized cause of the failure. It should be noted that, because this prototype was designed for research purposes, the team does not believe that the failure was a error in design, but rather a demonstration of the factors that must be considered in dealing with rotational dynamics.

Testing Set-Up

When the failure occurred, the team was performing preliminary testing on the prototype. During these tests, the shaft was rotated at a constant angular velocity and data was taken with the support remaining stationary at various lengths down the shaft. The steps in the testing activities were as follows:

1. Place support at starting position
   During this particular test the starting position was 10 inches from the top of the shaft.

2. Start rotating shaft and slowly increase the speed until desired speed was met
   The desired speed was 5000 RPM for this test. In reaching this speed, the team slowly increased the speed through a resonant range.

3. Take data while support remains stationary

4. Move support two inches down to next position
   The support was moved to 12 inches, effectively shortening the length of the shaft and increasing the natural frequency. At this time, the resonance range increased to include the speed of operation and failure occurred.

It should be noted that during all testing the team was under the constant supervision of the Reconfigurable Manufacturing Lab Supervisor, Steve Erskine, and their sponsor, Dr. Hagay Bamberger.

Hypothesized Cause of Failure

The team has hypothesized that failure occurred due to resonance occurring at a lower frequency than expected while the support was at a given location, creating large deflections and plastically deforming the shaft. See Figure 49.

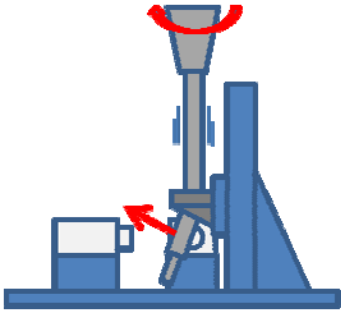


Figure 49: Resonance caused large deflections of shaft

The bearing that was purchased for the prototype only allowed for 2.5° of axial misalignment. Because of large deflections and deformation, this amount of misalignment was exceeded. Consequently, the bearing failed, sending ball bearings out of their retaining rings and removing the indirect points of contact that the inner race had with the outer race. This essentially made the inner race of the bearing another off-balance mass on the shaft, causing further deformation. See Figures 50 and 51 below.

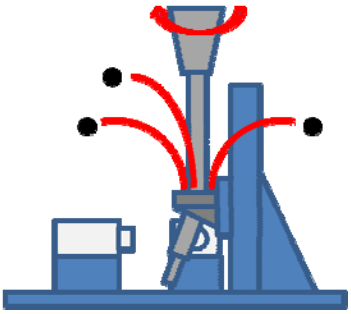


Figure 50: Large deflections of shaft caused bearing to fail

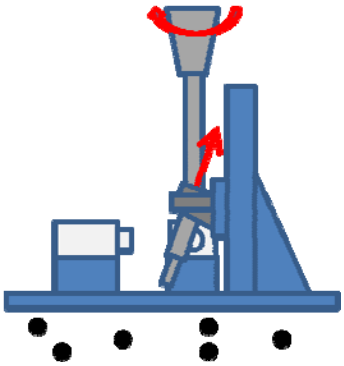


Figure 51: Inner race of bearing was no longer supported by structure of bearing

Given the fact that the deformed shaft was still rotating at 5000 RPM, it interfered with the parts of the mechanism directly surrounding it. The end of the shaft hit both sensors and the steel sensor mounts, cracking the lenses and creating large scratches in both itself and the steel

sensor mounts. Additionally, the middle of the shaft scraped against the support, creating gouges in both itself and the support. See Figure 52 below.

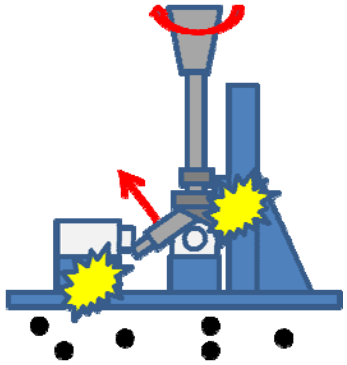


Figure 52: Deformed shaft impacted surrounding components of mechanism

All of the interference with the parts of the mechanism created large stresses in the shaft. Finally, the threaded end of the shaft came off, and a combination of bending and shearing occurred at the collet attachment point. See Figure 53 below.

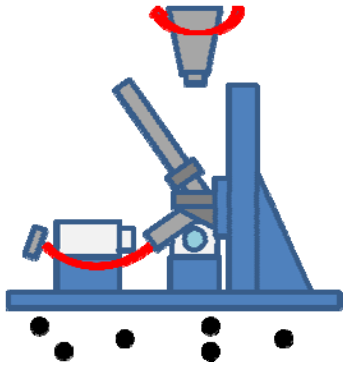


Figure 53: Shaft bends and shears

Failure Analysis – Speed, Force, and Energy Analysis of Components

This section analyses what sorts of speeds, forces, and energies were present in the test setup when the shaft bent and failed and the ball bearing ejected its balls. These values are just estimates as the team does not know exactly when different components failed and had to hypothesize the order of events.

Speed of Bearing Balls upon Ejection

The team has come up with two estimates of possible speeds at which the bearing balls could have been travelling after being ejected from the bearing races. They are 4.7 m/s and 110 m/s. These two values should be thought of as the lowest and greatest possible speeds of the balls, respectively. The team determined that the velocity of the balls must have been lower than the

maximum velocity calculated here since this is roughly the speed of a bullet and would have done notable damage to whatever it struck. Since the team did not notice any such damage to the surroundings upon its inspection after component failure, the balls must not have had this large of a velocity. Both speeds, v_{ball}, were estimated using Eq. 8:

$$v_{ball} = \omega \cdot R_{release} \quad (8),$$

Where ω = the angular velocity of the system (5000 RPM = 523.6 rad/s) and $R_{release}$ = the distance from the center of the ball to the axis of rotation of the mill when the ball was released, shown in the two different configurations in Fig. 54 below.

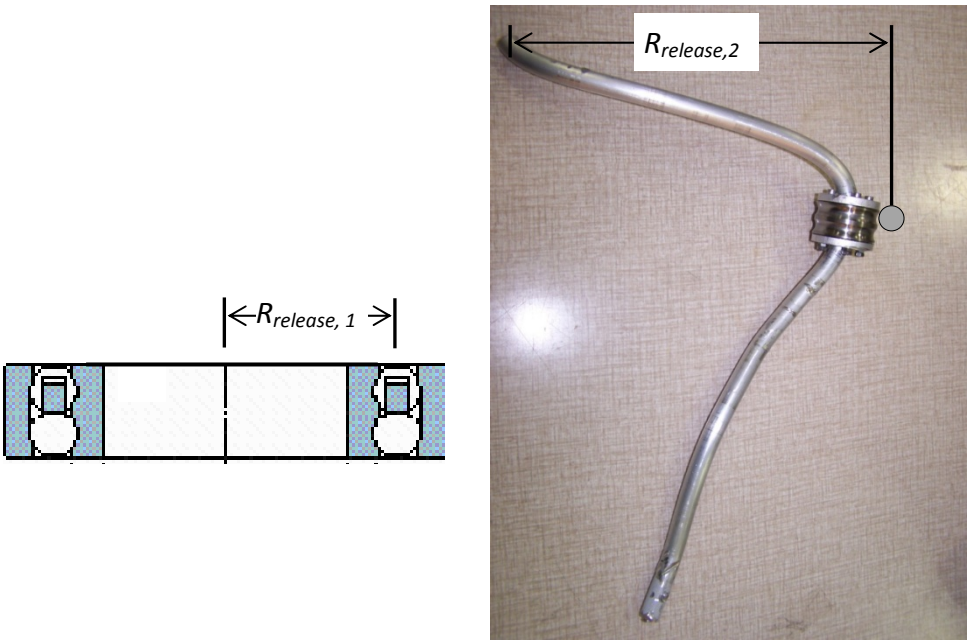


Figure 54: Minimum and maximum radii of ball release, $R_{release,1}$ and $R_{release,2}$, respectively [17]

Force to Fracture Aluminum Shaft

The team has estimated the force needed to fracture the aluminum shaft to be 3300 N (740 lbf). The shaft was determined to have fractured due to bending not shear since the force needed to fail the component in bending with our geometry, $F_{bend,u}$, was found to be smaller than the force needed to fail the component in shear, $F_{shear,u}$. The force needed to shear an aluminum shaft of our diameter was determined to be 23000 N (5200 lbf). It should be noted that a stress riser was observed on the part of the shaft that remained in the collet from the collet pinching it. The actual force needed to fracture the shaft would therefore be lower than the force determined by calculations. These forces were estimated using Eqs. 9-11:

$$\sigma_u = \frac{M \cdot y}{I} = \frac{\left(F_{bend,u} \cdot R_{bend} \cdot \frac{D_{shaft}}{2}\right)}{I} \quad (9),$$

$$I = \frac{\pi}{64} D_{shaft}^4 \quad (10),$$

$$F_{shear,u} = \tau_u \cdot A_\perp \quad (11),$$

Where σ_u = the ultimate tensile strength of aluminum, M = the moment being exerted on the shaft, y = the distance from the axis of the shaft to the point at which stress is being calculated, I = the second moment of area of the shaft, R_{bend} = the moment arm at which the bending force was being applied, D_{shaft} = the diameter of the shaft, τ_u = the shear strength of aluminum, and $\boldsymbol{A}_\perp$ = the cross-sectional area of the shaft. The moment arm at which the bending force was applied, R_{bend}, is shown in Fig. 55 below:

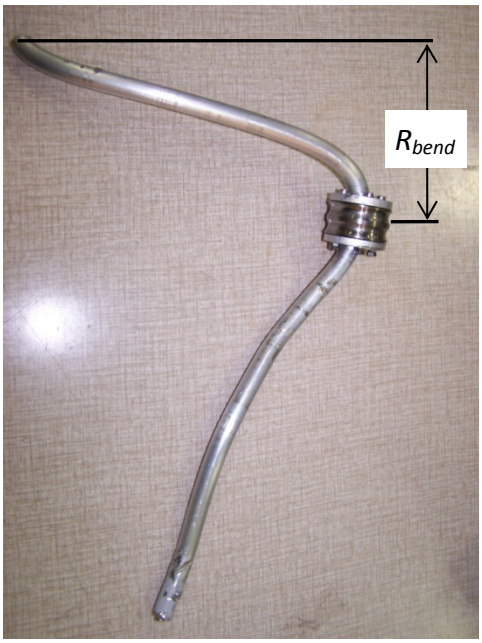


Figure 55: Moment arm of the bending force, R_{bend}

Kinetic Energy of Inner Bearing Components

The team has estimated the kinetic energy, *KE*, of parts of the bearing assembly that remained attached to the shaft (the ball bearing's inner race, sleeve bearing, bearing adapters, and fasteners) to be 480 J. The energy of these components immediately after fracture of the shaft would be lower as some of their energy would be used to fracture the shaft.

$$KE = \frac{1}{2} m \cdot (\omega \cdot R_{KE})^2 \quad (12),$$

Where m = the total mass of the components, ω = the angular velocity of the system (5000 RPM = 523.6 rad/s), and R_{KE} = the distance from the axis of rotation of the mill to the bearing components that remained on the shaft, shown in Fig. 56 below.

Figure 56: Radius of rotation of the bearing components, R_{KE}

Design Change and Critique

The team completed its work on this project and has critiqued its design and test setup. This section includes the description of the deviation of actual manufacturing from the manufacturing plan and a critique of the design. The one change in manufacturing was the addition of a slit in the bearing housing to allow the ball bearing to rotate more freely. The critiques of the design include uncertainty about the simplicity of the support, the quality and material of the shaft, and the suggested addition of a restraining ring. These are detailed below.

Fitting Self-Aligning Ball Bearing into Bearing Housing

The team had to cut a notch into the end of the bearing housing to reduce binding and allow the self-aligning ball bearing to rotate and misalign freely. After machining the bearing housing using the Fadal CNC machine and checking the dimensions with a telescoping gauge and micrometer (error ± 0.001 in), the team press fit the bearing into the bore. After the bearing was fit, it was noticed that it no longer rotated as freely as it had before. The team then cut a notch in the bearing housing to relieve the excess pressure constricting the motion of the bearing. This solved the problem. The team ensured that this modification would not be an issue by referring to the FEA stress analysis and noting that the shaft would fail before exerting a large enough force to damage the bearing housing.

Simplicity of Support

The team is still not satisfied with how simple its "simple" support actually was. Within the time and financial constraints of this project, the team believes it has generated as good a design as possible, but there is no analytical way to check exactly how simple the support was.

Quality and Material of Shaft

The team has come to the conclusion that non-uniformities in the aluminum shaft led to the failure of the test setup. These non-uniformities consist of inconsistent material and the shaft itself being unbalanced. The team would recommend any further work on this project be done with a precision ground, balanced shaft, to fix this problem. Also, one might consider using a steel shaft for preliminary testing to reduce the amplitude of vibrations and ensure safety. The team recommends that anyone working on this project in the future use only English units for any stock materials. During purchasing, the team was unable to find any precision ground aluminum shafts in metric units. However, it was able to find several suppliers that sold ½" diameter precision shafts. This change in conventions would only affect the shaft and sleeve bearing diameters and would be worth the initial effort.

Addition of a Restraining Ring

One final improvement that the team would have added if it had more time is the addition of a restraining ring around the shaft to ensure that the vibrations of the shaft do not bend to an angle larger than the allowable angle of the self-aligning ball bearing. This part could be a simple component that clamps to the T-bed of the mill with the shaft going through its center.

Recommendations for Future Work

The team has completed its work on this project and generated some recommendations for future work to build upon the work that has already been done. These recommendations are based on the team's design concerns and testing results and are as follows:

- Design a more effective and robust simple support
- Use a line sensor to ensure measurements are taken simultaneously and ensure measurements are being taken truly perpendicularly
- Account for damping in analytical modeling
- Account for distributed force of added mass at end of shaft (instead of the point force assumed in our MATLAB model)
- Perform testing with professionally balanced, precision ground shaft
- Perform preliminary testing with steel shaft to reduce vibrational amplitude
- Design safety ring to limit vibrational amplitude from entering plastic deformation range
- Design additional safety barriers to shield from potential projectiles during operation.

Administrative Details

Budget

The team broke the cost of the project into four parts. This breakdown can be found in the table below:

Item	Cost
Bearings	$78.13
Stock Material	$123.82
Electronics and Controls	$36.65
Hardware	$48.07
Total	$286.67

Table 16: Budget breakdown for project

The budget shown in the table above totals to $286.67, well below the $400 provided to the team through the department. As noted in the bill of materials, the motor and sensor were obtained through the Engineering Research Center for Reconfigurable Manufacturing Systems. These components added no cost to the overall budget. Similarly, some of the stock material was provided by Bob Coury from the Undergraduate Machine Shop at the University of Michigan. All other components were purchased through McMaster-Carr, Home Depot, and Grainger.

It should be noted that, given the small tolerances on several of the parts and the tight schedule the team faced, they sent out requisitions for quotes to have the bearing adapters and the shaft professionally manufactured. However, the allotted budget did not provide for these.

Project Timeline

In order to complete the mechanism by the requested date (December 10, 2009), P2C2 constructed a detailed project plan based on the five basic stages of the design process: Definition, Divergence, Transition, Convergence, and Prototyping.

During the first stage of design, Definition, the team met with Dr. Bamberger to discuss the customer needs, completed background research on the dynamic response of shafts and on existing designs that measure natural frequency, and set benchmarks.

During the second and third stages of design, Divergence and Transition, the team discussed the math behind the design with their sponsor, performed vibrational analyses in Autodesk Inventor 2009, and determined the design's functional decomposition. From the functional decomposition, they generated concepts to drive each function block and evaluated the concepts such that they could decide upon an alpha design.

The fourth stage of design, Convergence, consisted of P2C2 proving that the components selected would meet the project needs and determining how they would be integrated into the system. P2C2 performed two checks: one in CAD and one with layouts of components to see that the design would fit into the mill that they are using. They ran a materials analysis with CES and a safety analysis. Additionally, they consulted with Optimet and Aerotech to ensure a thorough understanding of the sensors and motor. Up until this point, the team had followed the Gantt chart that they had established for themselves.

The fifth stage of design, Prototyping, consisted of machining and assembly. In parallel with these processes, the team coded the sensors and the motor, which required more time than originally expected. The team also performed various tests on each design component and the system. Although this process was expected to need more time, P2C2 was only able to perform so many tests on the system, and finished the testing process earlier than expected. In place of the remaining tests, they hypothesized the order of events during failure and generated a list of recommendations for the sponsor.

For a comprehensive layout of the tasks fulfilled to bring the design to completion, please see the Gantt chart in Appendix E.

Acknowledgments

The team would like to acknowledge the following people for making this project possible:

- Dr. Hagay Bamberger and Professor Yoram Koren for their sponsorship of the project and their guidance throughout each step of the engineering design process;
- Steve Erskine for his expert insight on the machining process and operation of machines and his willingness to let the team have non-stop use of the Fadal vertical mill;
- Dr. Reuven Katz for his expert insight on the field of dynamics;
- Sankalp Arrabolu and Andrew Chadderdon for their time and effort spent determining the faulty points in the C code for the sensors and their corresponding fixes;
- Mike Doherty for the technical support he provided regarding the Optimet laser probes and controllers;
- Roger Burg for the technical support he provided regarding the Aerotech linear stage, motor, and controller.

Summary and Conclusions

Dr. Bamberger and the Engineering Research Center for Reconfigurable Manufacturing Systems requested the team to design, fabricate, and operate an add-on sensor system to measure the dynamics at the free end of an unbalanced rotating shaft.

The customer requested that the design be structurally integrated with an existing machine of the team's choice, continuously collect data while the support traverses along the shaft, and

accurately and precisely measure the shaft dynamics. From these design requests, the team communicated with the customer to establish engineering specifications for the overall design and its individual components (the sensor and support) and to determine potential challenges to factor into their project plan.

The team decomposed the design into four main functions: (1) rotate shaft, (2) support shaft, (3) traverse support, and (4) measure displacement. For the first function, the team narrowed down a shaft design based on the customer's requests. For the next three functions, the team generated concepts and analyzed them, narrowing them down to an alpha design that consists of a direct-driven platform that traverses the support and laser triangulation sensors.

After deciding on an alpha design, the team took material and safety items into consideration. These items helped the team to finalize a design consisting of a sleeve bearing held by adapters within a self-aligning bearing, an Aerotech linear stage, and two Optimet laser point sensors of lens designations 50 mm and 100 mm. The team then determined what they needed to procure, and sought cost-effective ways of procuring them.

They set how the mechanical aspects of the design would be integrated through both CAD and full-scale layouts in the Fadal mill. Similarly, they considered how the electrical and coding aspects of their design (based on the specifications of the sensors and the motor) would be integrated. The team created detailed manufacturing, testing, and safety procedures on which they were to base any further work on the design, and performed their machining and assembly.

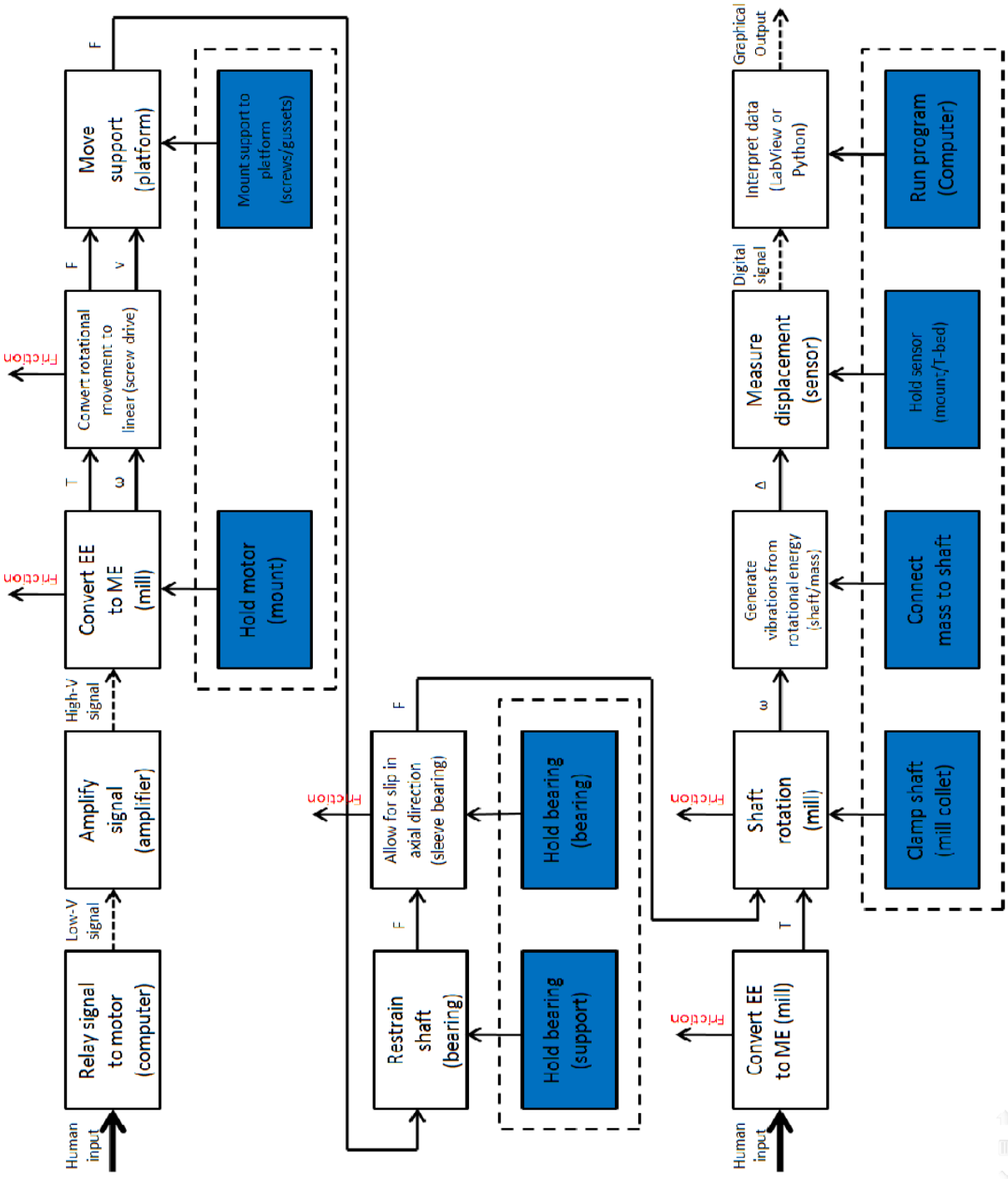
Upon completion of these tasks, they began their validation of the components of the design and their operation of the design. Though the team was able to complete component validation, they only obtained a few sets of data from their design operation before the shaft hit resonance before its expected resonance point, creating large deflections, and ultimately rendering the design unrecoverable. In response to this, the team broke down the failure of the shaft step-by-step, and determined the improvements necessary to further the project.

It should be noted that the team believes that this failure was not caused by design oversight. Rather, the results show the need for a more accurate model and give further insight into an uncharted area of research.

Appendix A – QFD

Accuracy of location of moving support is 3% of length of exposed shaft									
Resolution of measurement is within 10% of amplitude									
Design is integrated with vertical mill (can neglect effect of gravity)			+						
Fixture reduces vibrations at supported point by 80%			-	+					
Moving support traverses from 10% to 90% of length of exposed shaft			-			-			
Sampling rate is more than twice the frequency of shaft rotation				+					
Sensor must be within 2% of length of exposed shaft from free end				+			+		
			Technical Requirements						
Customer Needs	**Customer Weights**	**Keep Type**	Accuracy of location of moving support is 3% of length of exposed s	Resolution of measurement is within 10% of amplitude	Design is integrated with vertical mill (can neglect effect of gravity)	Fixture reduces vibrations at supported point by 80%	Moving support traverses from 10% to 90% of length of exposed sha	Sampling rate is more than twice the frequency of shaft rotation	Sensor must be within 2% of length of exposed shaft from free end
Ability to be structurally integrated with existing machine	5		1		9	3	3		
Ability to add masses of varying sizes to the end of the shaft	4					1			1
Ability to operate within machine's range of speed	4				1	1		9	
Ability to restrain shaft	5		1		3	9	9		
Accurate measurement	5		9	9		1		9	9
Automated moving support with respect to rotating shaft	4		3		1	3	9		3
Continuous data collection	4			9			3		
Ease of operation	2				9				
Low cost	1		3	3	9	1	1	9	
Maintainability	2				3			1	1
Manufacturability	1				9	3	3		3
Mass does not change the inertia of the shaft	3					3			
Precise measurement	5		9	9		3		3	9
Robustness	2			3	9	3	3		1
Safety of operation and manufacturing	5				9	1	3		
Shaft made of aluminum, steel, or UHMWPE	3			9		9			
Shaft natural frequency lower than machine speed while support at least 40% down length of shaft	3		1	3		1		3	
Simulation can be run with no mass at end of shaft	4					1			1
Small traversing resolution for moving support	4		9		3	3	9		
Completion by December 10, 2009	5			3	9			1	1
	Raw score		154	186	230	170	169	121	122
	Relative Weight		13%	16%	20%	15%	15%	11%	11%
	Rank		5	2	1	3	4	7	6
Technical Requirement Units			mm	mm		mm	mm	Hz	mm

Appendix B – Functional Decomposition



Appendix C – Concept Generation

Convert EE to ME

- Motor
- Magnetic field generation (similar to rail gun)
- Antigravity
- Internal combustion engine

Convert rotational to linear (only required for some of above options)

- Pulley system
- Rack and pinion
- Screw drive

Move support

- Manual force
- Spring

Restrain shaft

- Ball bearing
- Sleeve bearing
- Roller bearing
- Magnetic field
- Triangulated bearings on exterior
- Triangulated tension wires
- Combination of ball/sleeve bearing

Allow for thrust slip

- Lubricant
- Sleeve bearing

Rotate shaft

- Fadal (vertical mill)
- Horizontal mill
- Lathe

Generate vibrations (Mass)

- Screw in end of shaft (w/ weights)
- Screw on outside of circumference of shaft (w/ weights)
- Weights with retaining ring (internal from end)
- Remove mass from shaft
- Magnets
- Epoxy weights to shaft (remove w/plyers)

Measure displacement

- Line laser
- Probe (x2)
- Touch sensor
- Ruler
- Photograph
- 4-quarters shadow
- Eddy current sensor
- Capacitive sensor

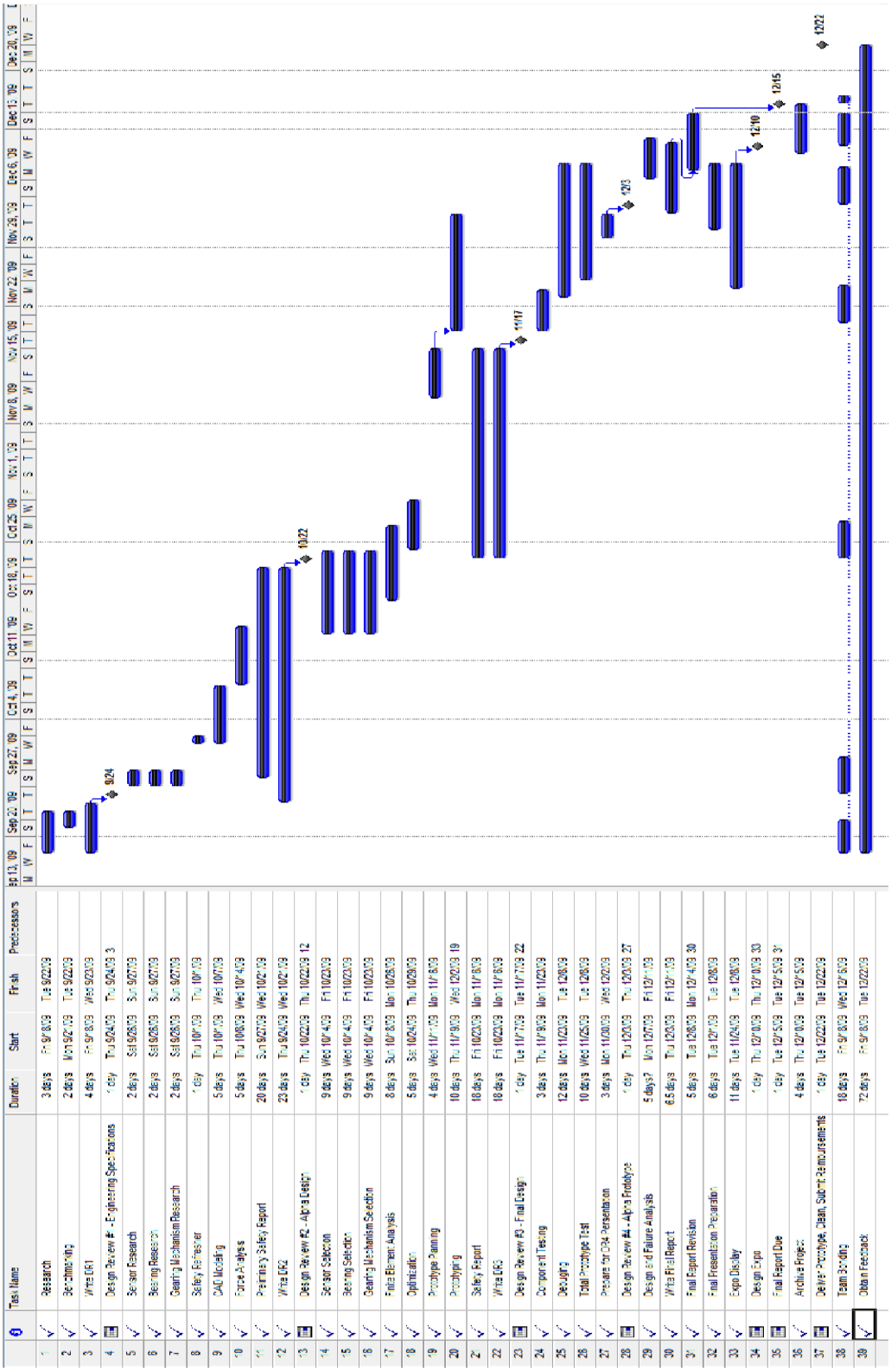
Interpret data

- LabVIEW
- Python
- C

Appendix D – MATLAB Code Provided by Dr. Bamberger, Edited by P2C2

```
clear

% simple support in x=b, mass M at x=l, rotating at Omega

d=4/8*25.4e-3; % diameter [m]
I=pi*d^4/64; % Second moment of inertia [m^4]
E=68.9*10^9; % Elasticity Modulus [N/m^2]
l=20*25.4e-3; % length [m]
ro=2.7e3; % density [kg/m^3]
mu=pi*d^2/4*ro; % % linear mass density [kg/m]
M=1*10^-3; % mass at the end [kg]
e=1*10^-3; % Excentericity of the mass M [m]
Omega=12000/60*2*pi; % angular velocity [rad/sec]
tau=(mu*Omega^2/E/I)^0.25;
figure
for p=1:1:99
b=p/100*l; % position of the simple support [m]

stb=sin(tau*b);ctb=cos(tau*b);
shtb=sinh(tau*b);chtb=cosh(tau*b);
stl=sin(tau*l);ctl=cos(tau*l);
shtl=sinh(tau*l);chtl=cosh(tau*l);
MOm2EI=M*Omega^2/E/I; % for Eq. (8)

coef=[0 1 0 1 0 0 0 0; % Eq. (1)
      1 0 1 0 0 0 0 0; % Eq. (2)
      stb ctb shtb chtb 0 0 0 0; % Eq. (3)
      0 0 0 0 stb ctb shtb chtb; % Eq. (4)
      ctb -stb chtb shtb -ctb stb -chtb -shtb; % Eq. (5)
      -stb -ctb shtb chtb stb ctb -shtb -chtb; % Eq. (6)
      0 0 0 0 -stl -ctl shtl chtl; % Eq. (7)
      0 0 0 0 -tau^3*ctl-MOm2EI*stl tau^3*stl-MOm2EI*ctl tau^3*chtl-
MOm2EI*shtl tau^3*shtl-MOm2EI*chtl]; % Eq. (8)

ab=inv(coef)*[0;0;0;0;0;0;0;MOm2EI*e];
a0=ab(1);a1=ab(2);a2=ab(3);a3=ab(4);b0=ab(5);b1=ab(6);b2=ab(7);b3=ab(8);

Fs(p)=E*I*tau^3*(-a0*ctb+a1*stb+a2*chtb+a3*shtb+b0*ctb-b1*stb-b2*chtb-
b3*shtb); % Force at the support [N] Eq. (9)

x=[0:0.001:b];
plot(x/l*100,a0*sin(tau*x)+a1*cos(tau*x)+a2*sinh(tau*x)+a3*cosh(tau*x),'b')
hold on
x=[b:0.001:l];
plot(x/l*100,b0*sin(tau*x)+b1*cos(tau*x)+b2*sinh(tau*x)+b3*cosh(tau*x),'r')
end

xlabel('Position [%]')
ylabel('Deflection [m]')
figure
plot(Fs)
xlabel('Position [%]')
ylabel('Force at the support [N]'
```

Appendix E – Gantt Chart



	Task Name	Duration	Start	Finish	Predecessors
1	Research	3 days	Fri 9/18/09	Tue 9/22/09	
2	Benchmarking	2 days	Wed 9/23/09	Thu 9/24/09	
3	Write DR1	4 days	Fri 9/18/09	Wed 9/23/09	
4	Design Review #1 - Engineering Specifications	1 day	Thu 9/24/09	Thu 9/24/09	3
5	Sensor Research	2 days	Sat 9/26/09	Sun 9/27/09	
6	Bearing Research	2 days	Sat 9/26/09	Sun 9/27/09	
7	Gearing Mechanism Research	2 days	Sat 9/26/09	Sun 9/27/09	
8	Safety Refresher	1 day	Thu 10/1/09	Thu 10/1/09	
9	CAD Modeling	5 days	Thu 10/1/09	Wed 10/7/09	
10	Force Analyses	5 days	Thu 10/8/09	Wed 10/14/09	
11	Preliminary Safety Report	20 days	Sun 9/27/09	Wed 10/21/09	
12	Write DR2	23 days	Thu 9/24/09	Wed 10/21/09	
13	Design Review #2 - Alpha Design	1 day	Thu 10/22/09	Thu 10/22/09	12
14	Sensor Selection	9 days	Wed 10/14/09	Fri 10/23/09	
15	Bearing Selection	9 days	Wed 10/14/09	Fri 10/23/09	
16	Gearing Mechanism Selection	9 days	Wed 10/14/09	Fri 10/23/09	
17	Finite Element Analysis	8.6 days	Sun 10/18/09	Mon 10/26/09	
18	Optimization	5.6 days	Sat 10/24/09	Thu 10/29/09	
19	Prototype Planning	4 days	Wed 11/4/09	Mon 11/9/09	
20	Prototyping	10 days	Thu 11/5/09	Wed 12/2/09	19
21	Safety Report	18 days	Fri 10/23/09	Mon 11/9/09	
22	Write DR3	18 days	Fri 10/23/09	Mon 11/9/09	
23	Design Review #3 - Final Design	1 day	Tue 11/17/09	Tue 11/17/09	22
24	Component Testing	3 days	Thu 11/5/09	Mon 11/23/09	
25	Debugging	12 days	Mon 11/23/09	Tue 12/8/09	
26	Total Prototype Test	10 days	Wed 11/25/09	Tue 12/8/09	
27	Pressure Vessel Presentation	3 days	Mon 11/30/09	Wed 12/2/09	
28	Design Review #4 - Alpha Prototype	1 day	Thu 12/3/09	Thu 12/3/09	27
29	Design and Failure Analysis	5 days?	Mon 12/7/09	Fri 12/11/09	
30	Write Final Report	6.5 days	Thu 12/3/09	Fri 12/11/09	
31	Final Report Revision	5 days	Tue 12/8/09	Mon 12/14/09	30
32	Final Presentation Presentation	6 days	Tue 12/1/09	Tue 12/8/09	
33	Expo Display	11 days	Tue 11/24/09	Tue 12/8/09	
34	Design Expo	1 day	Thu 12/10/09	Thu 12/10/09	33
35	Final Report Due	1 day	Tue 12/15/09	Tue 12/15/09	31
36	Archive Project	4 days	Thu 12/10/09	Tue 12/15/09	
37	Deliver Prototype, Clean, Submit Assessments	1 day	Tue 12/22/09	Tue 12/22/09	
38	Team Bonding	18 days	Fri 9/18/09	Wed 12/9/09	
39	Obtain Feedback	72 days	Fri 9/18/09	Tue 12/22/09	

Appendix F – CAD Drawings of Manufactured Components

Figure F1: Engineering Drawing of Bearing Adapters (2)

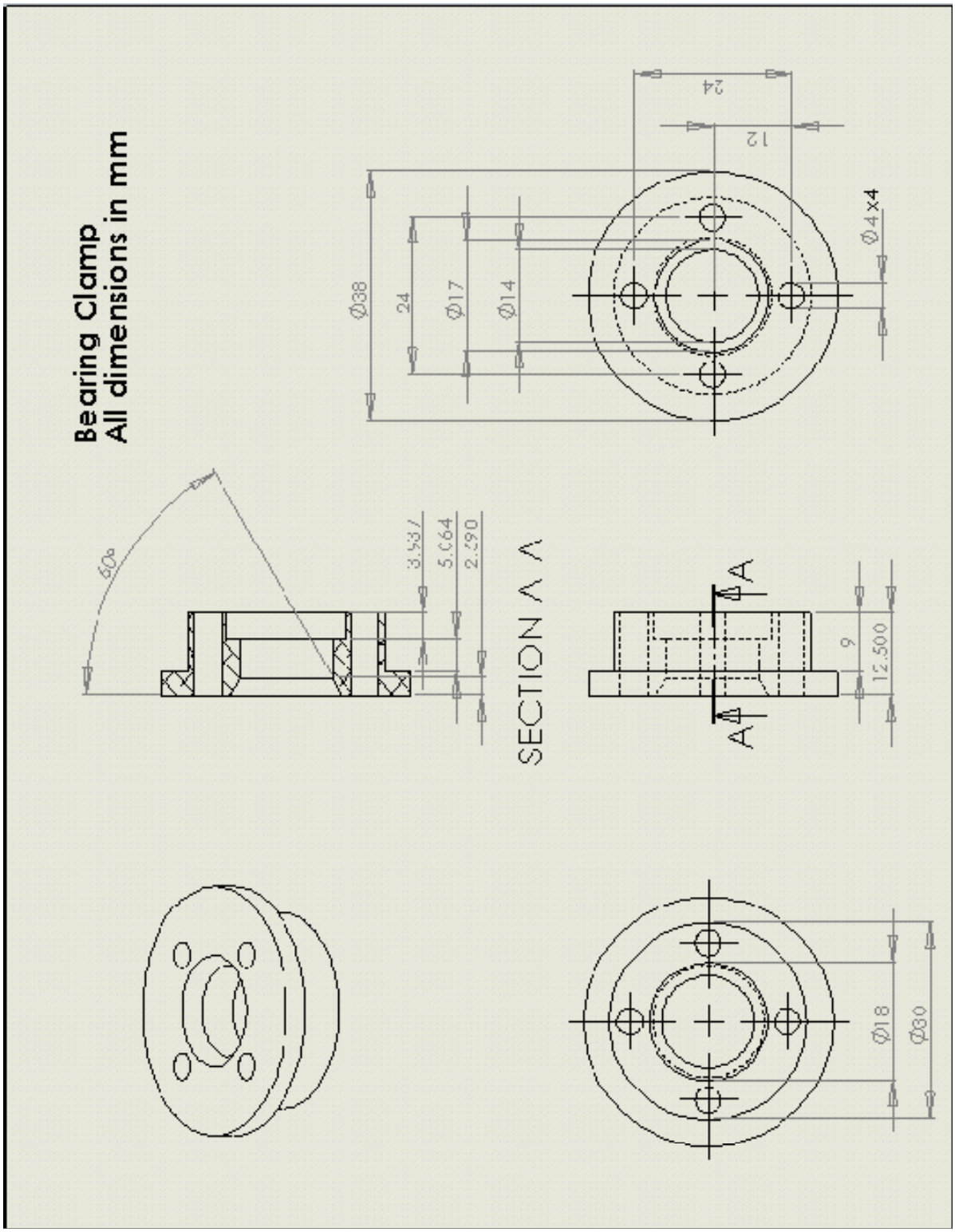


Figure F2: Engineering Drawing of Bearing Restraint Plate

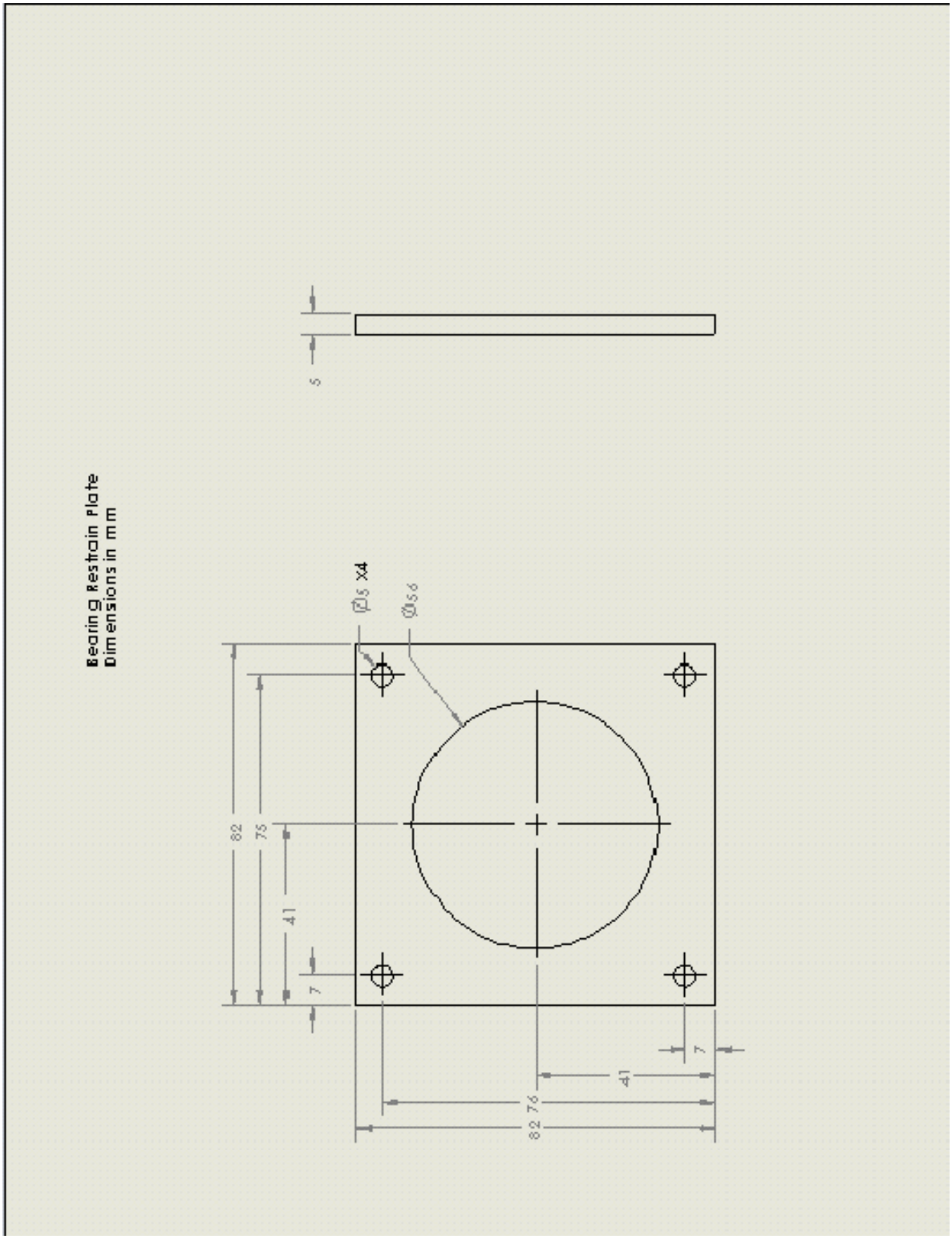


Figure F3: Engineering Drawing of Bearing Housing

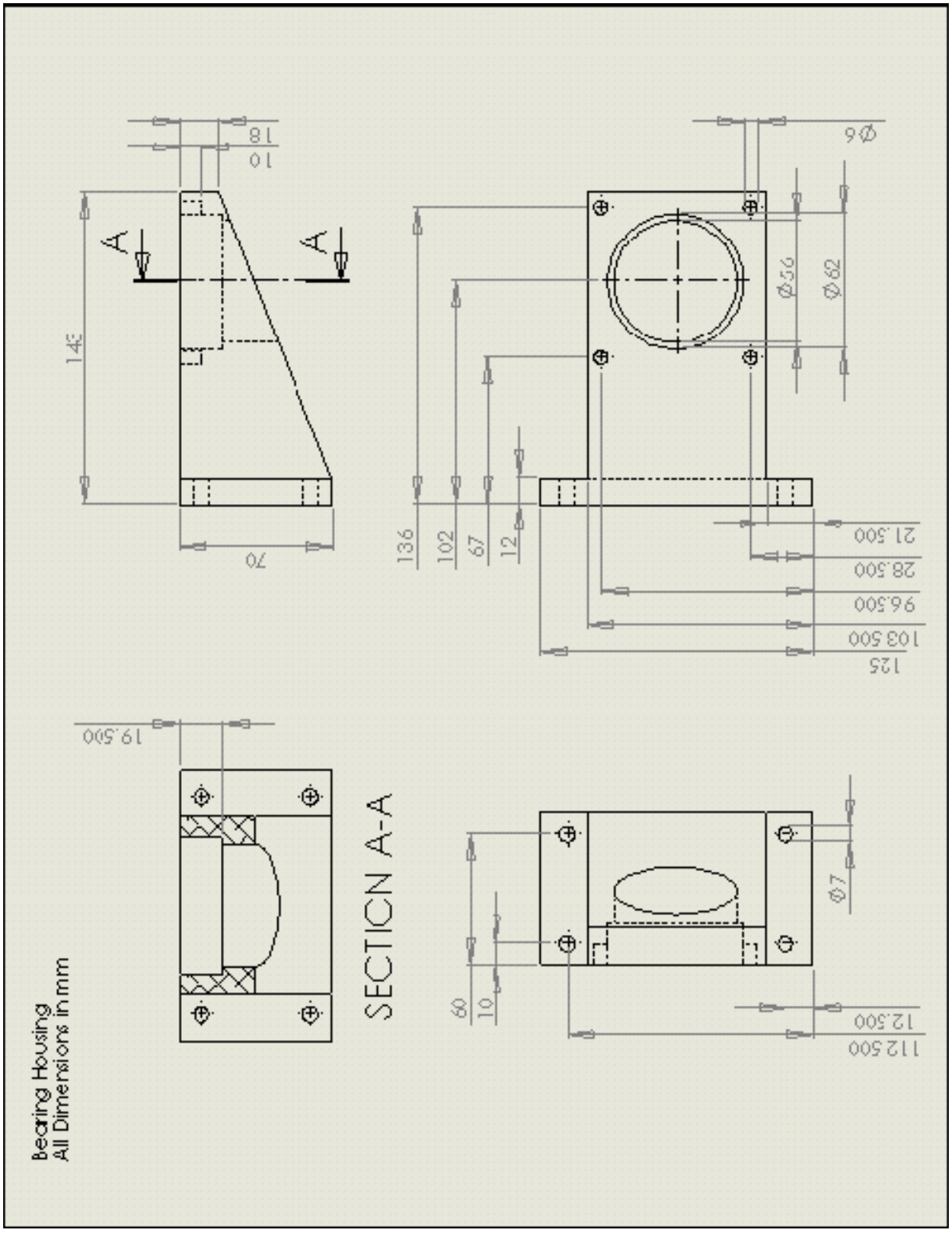


Figure F4: Engineering Drawing of T-Bed Adapters (2)

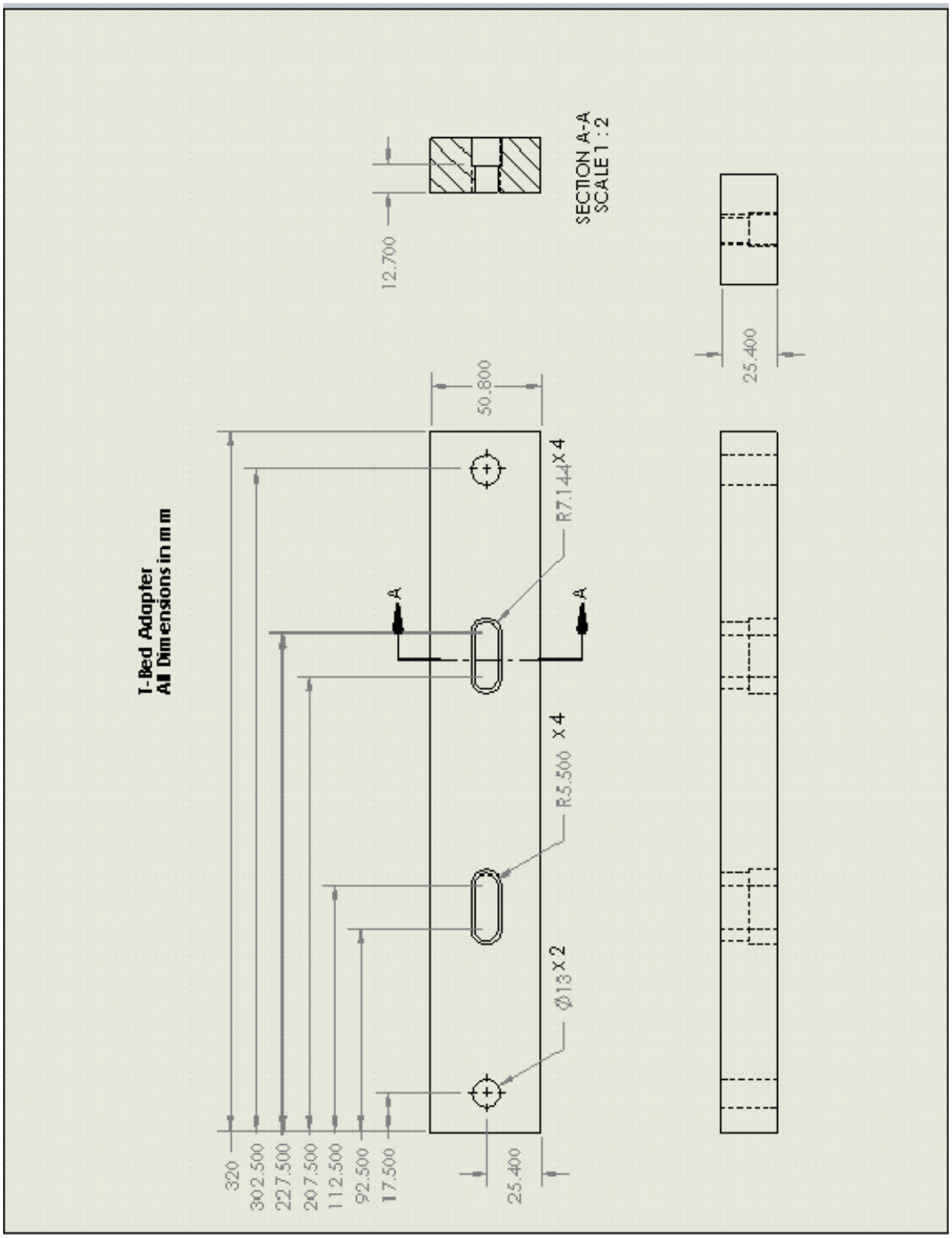


Figure F5: Engineering Drawing of Sensor Mounts Adapter Plate

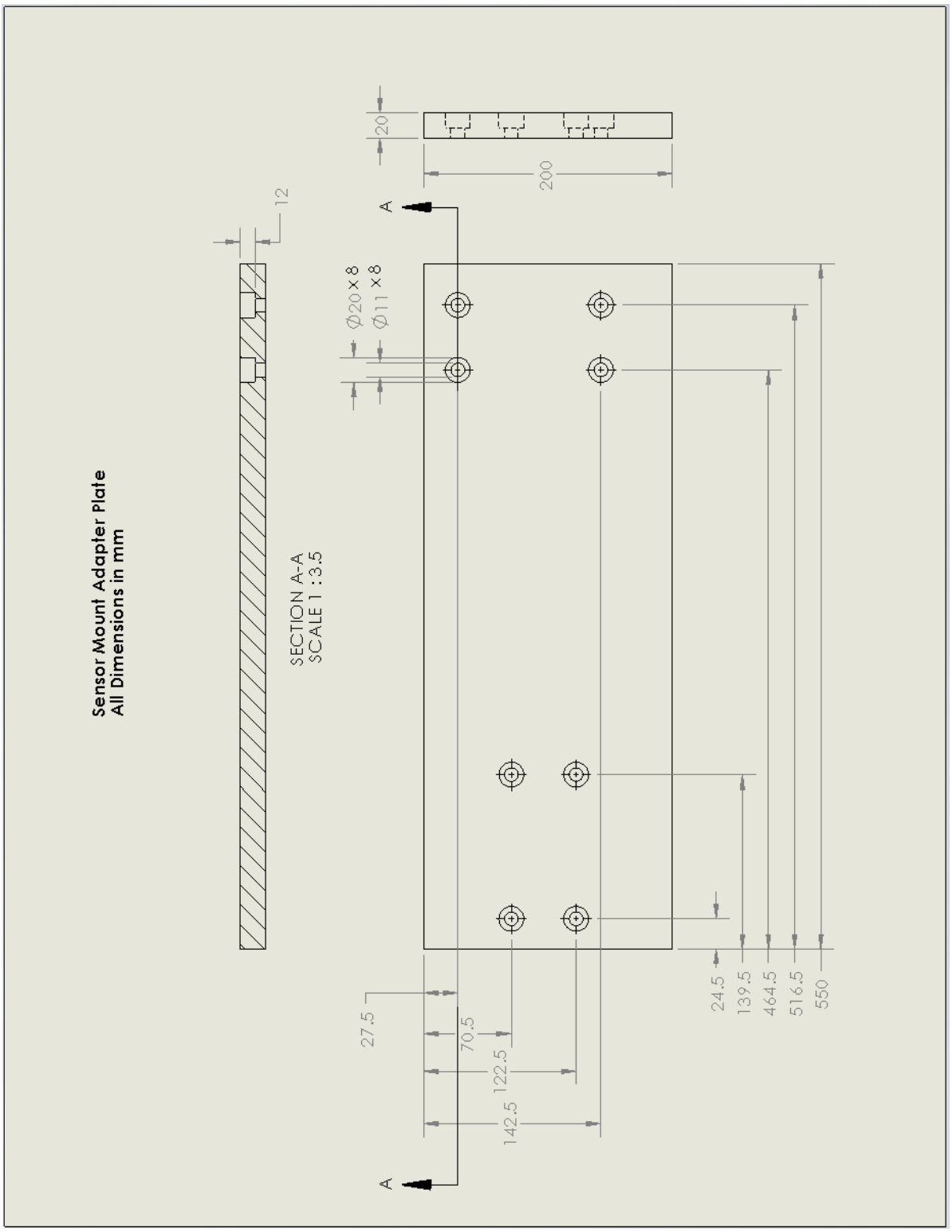


Figure F6: Engineering Drawing of Shaft

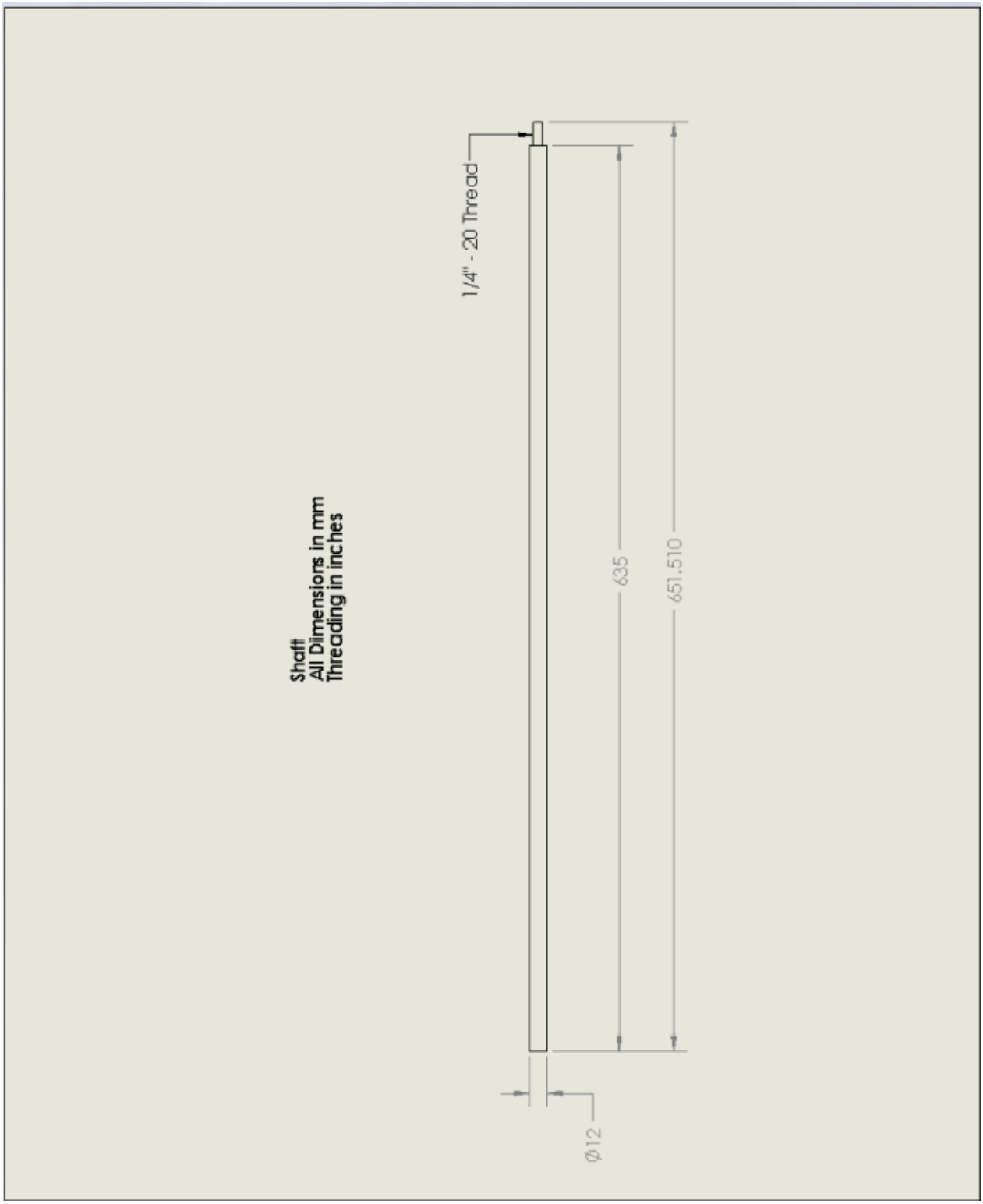
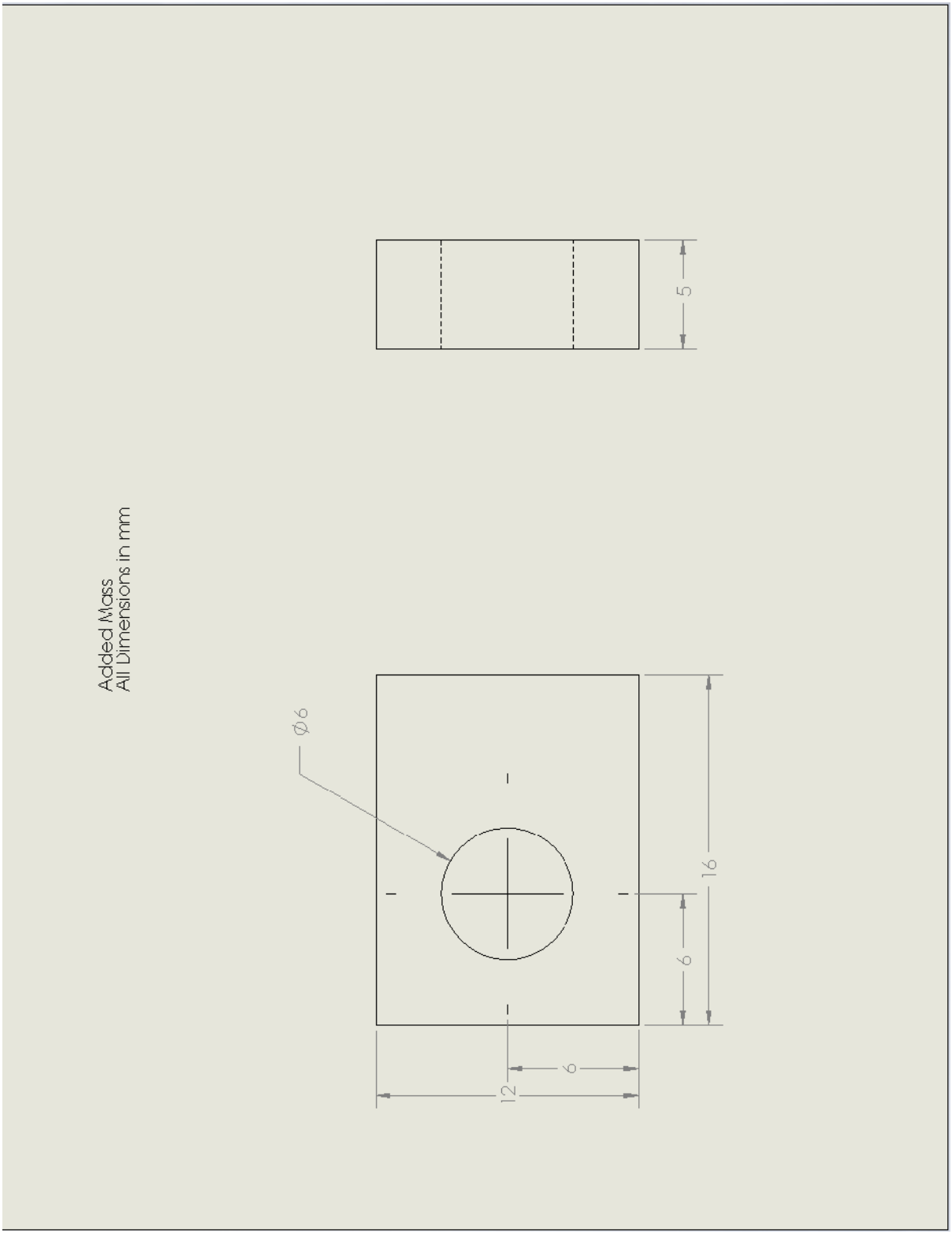


Figure F7: Engineering Drawing of Added Mass



Appendix G – Material Selection and Environmental Impact

Material Selection for Bearing Housing with CES

Function:

Cantilever beam in bending

Objectives:

Minimize deflection

Minimize cost (for a given volume)

Constraints:

Fixed geometry

Easily machinable

Material Index:

$$M = \frac{E}{\rho^2}$$

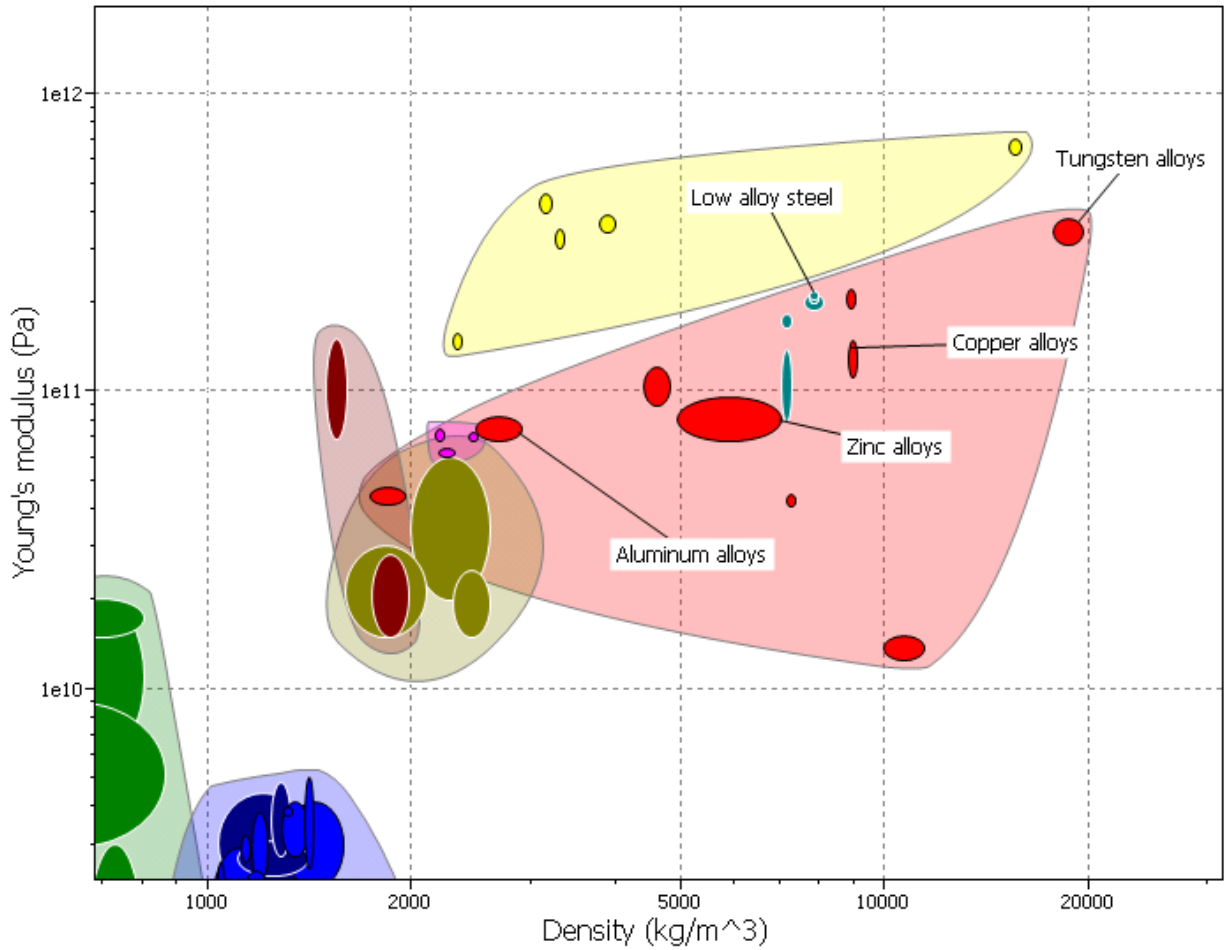


Figure G1: CES results for bearing housing

The above plot was generated with axes based on the components of the material index. From this graph the team determined the top five best material choices that maximized the material index. Tungsten and steel alloys were ruled out due their lack of machinability. Copper was ruled out due to its high price. Finally, zinc alloys were ruled out due to their lack of availability to the team. Therefore, aluminum was chosen as the most suitable material. It is readily available, low cost and easily machinable.

Material Selection for Sleeve Bearing with CES

Function:

Allow axial slip between rotating shaft and self-aligning ball bearing

Objectives:

Maximize stiffness (below that of aluminum)
Minimize cost (for a given volume)

Constraints:

Material has stiffness less than aluminum to facilitate installation

Material Index:

$$M = \frac{E}{C_m \cdot \rho}$$

The plot on the following page was generated with axes based on the components of the material index. From this, the team found which materials are used in commercially available sleeve bearings. Of the available materials, bronze and graphite were ignored as they did not meet the requirement that the Young's modulus be below that of aluminum (to ensure that the sleeve bearing can be tightly clamped into place using a fabricated aluminum adapter). The remaining materials were labeled on the above plot. One material, Vespel (or polyimide) did not appear in CES, it was not an ideal choice because it was the most expensive polymer sleeve bearing.

The team selected an acetal sleeve bearing based on the parity of elastic moduli of the least expensive materials, nylons and acetals. The acetal bearings were the same price as nylon bearings on McMaster-Carr, but they were rated for higher operating loads and speeds. From the above plot, it is evident that nylons and acetals are the best materials for our application. They have Young's moduli over the same range, but acetals appear to cost more.

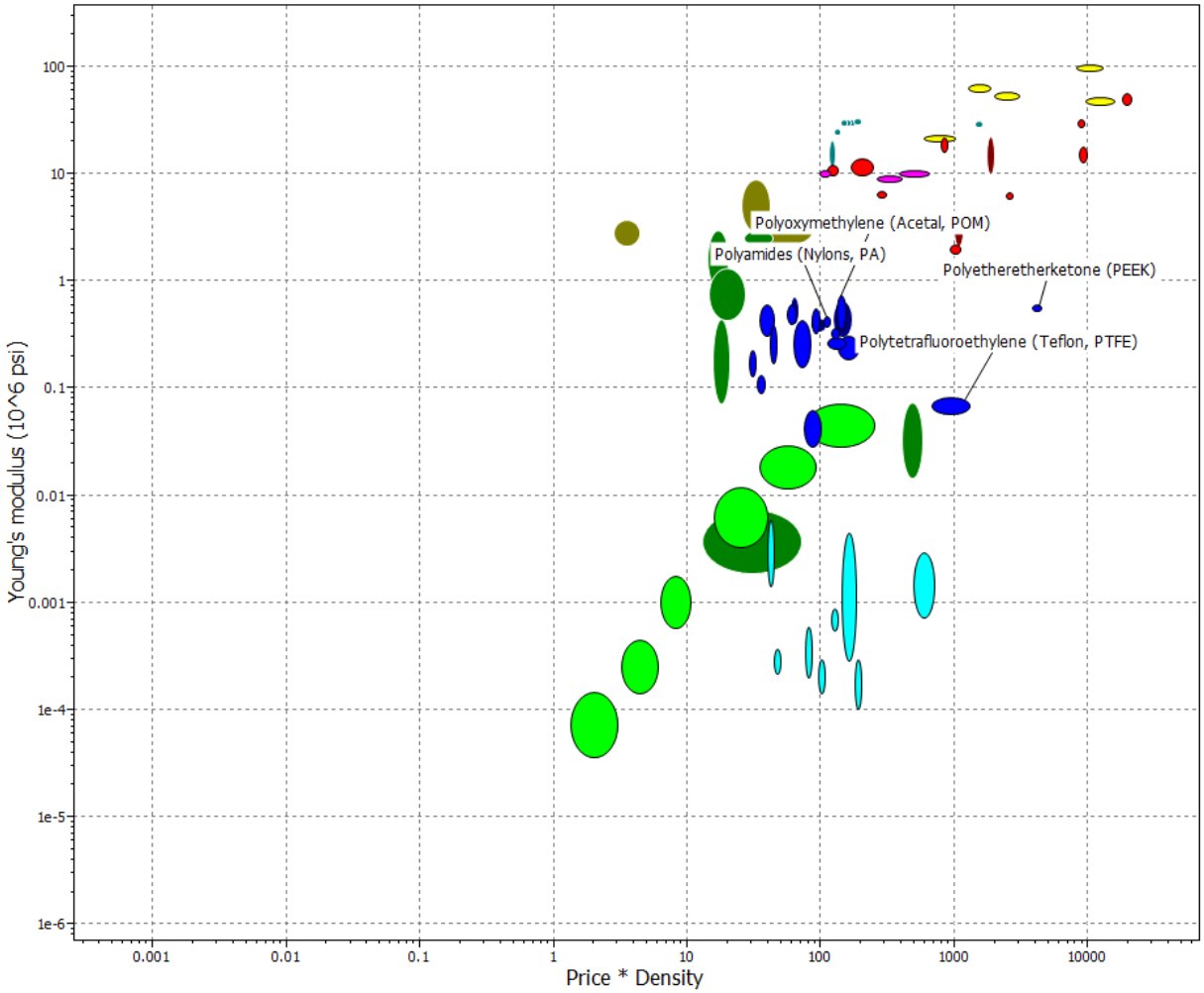


Figure G2: CES results for sleeve bearing

Design for Environmental Sustainability

The team used SimaPro 7.1 software to gauge the environmental effects of two materials considered for the shaft: aluminum and steel. The software was used to calculate the total mass of air emissions, water emissions, raw material usage and waste. The findings are summarized in Figure G3. From this graph, it is apparent that steel has fewer environmental effects than aluminum. During the design process, the team chose aluminum for the rotating shaft because simulations showed that it would allow for greater deflections at the end of the shaft. Looking back, not only would steel have been a more environmentally sound choice, but it would have also been a better choice for the design. It turned out that with aluminum the deflections were too large and were out of the reading range of the sensors. Steel would have solved this problem, as well as had less of a detrimental effect on the environment.

The output graphs from the SimaPro software are also presented below. It can be seen that the meta-category of resources is the most important according to the EcoIndicator 99 points. Also, it can be seen that aluminum has the higher overall EcoIndicator 99 score of about 3 while steel

has a score of about 0.7. A higher EcoIndicator 99 score corresponds to Aluminum being more detrimental to the environment.

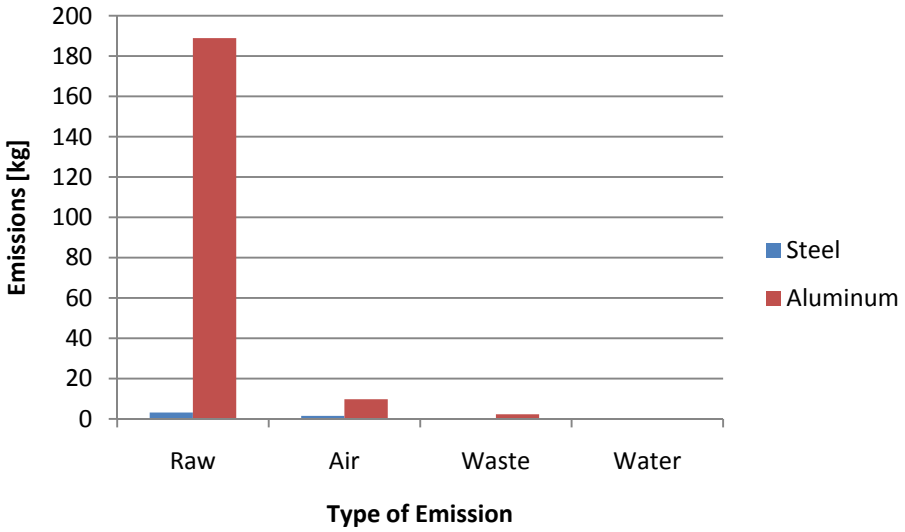


Figure G3: Aluminum uses more raw material and produces more waste and more air and water pollution than steel

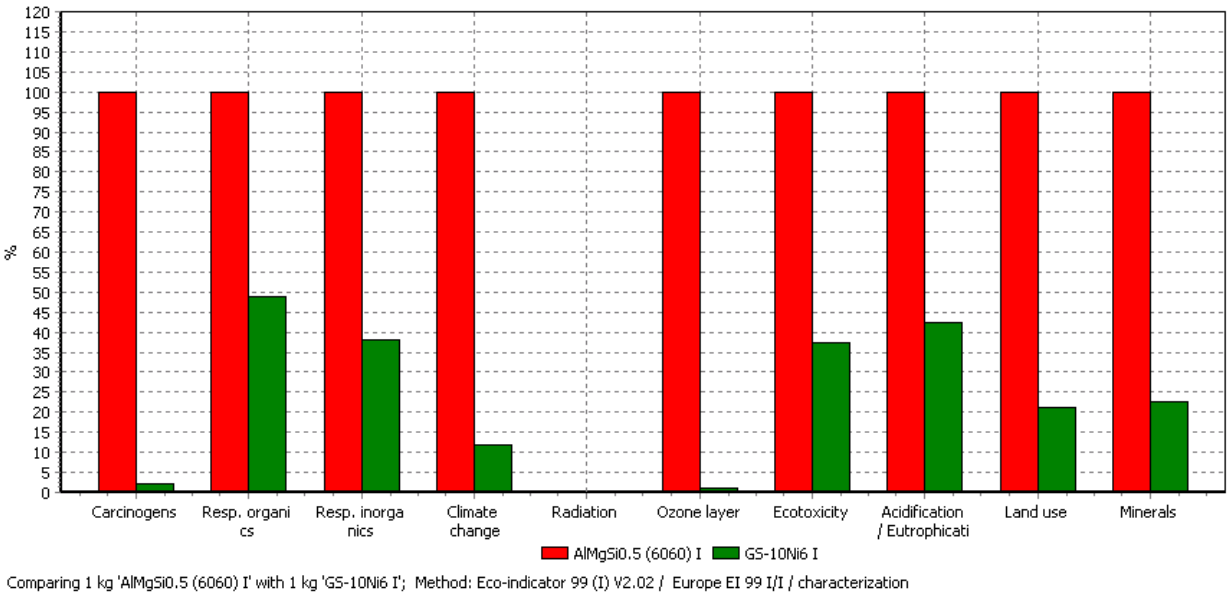


Figure G4: Comparison of materials shows aluminum is more detrimental to the environment

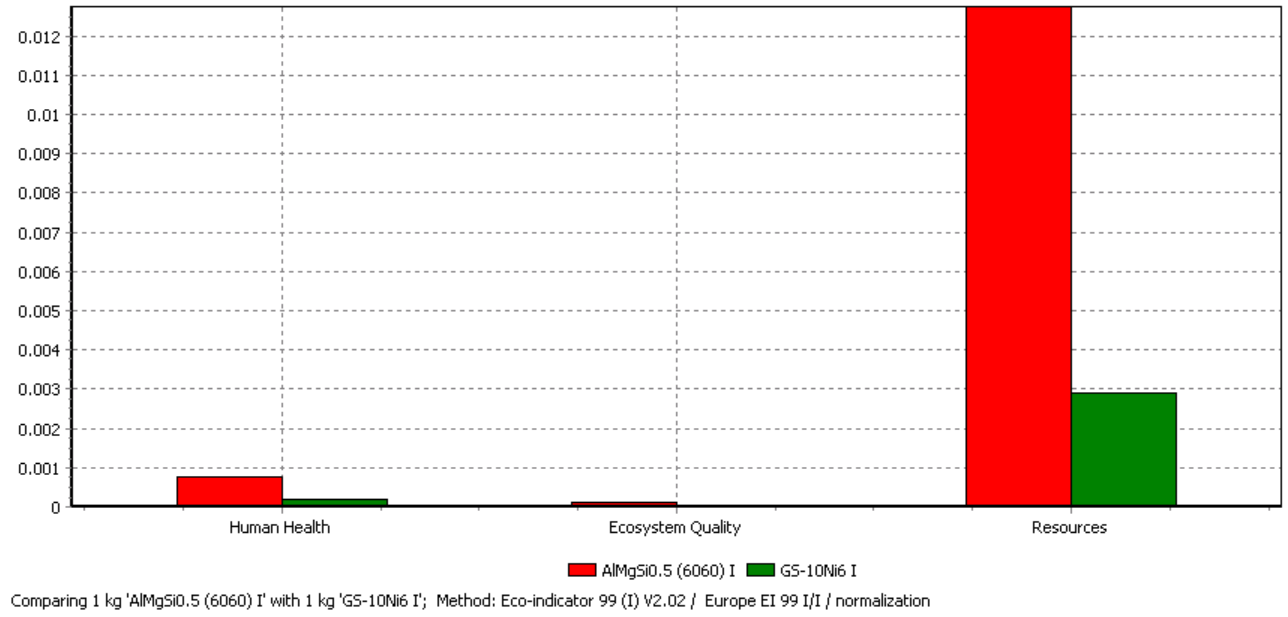


Figure G5: When normalized against the effect of an "average European person" over 1 year, even the less environmentally friendly aluminum does not have a significant impact

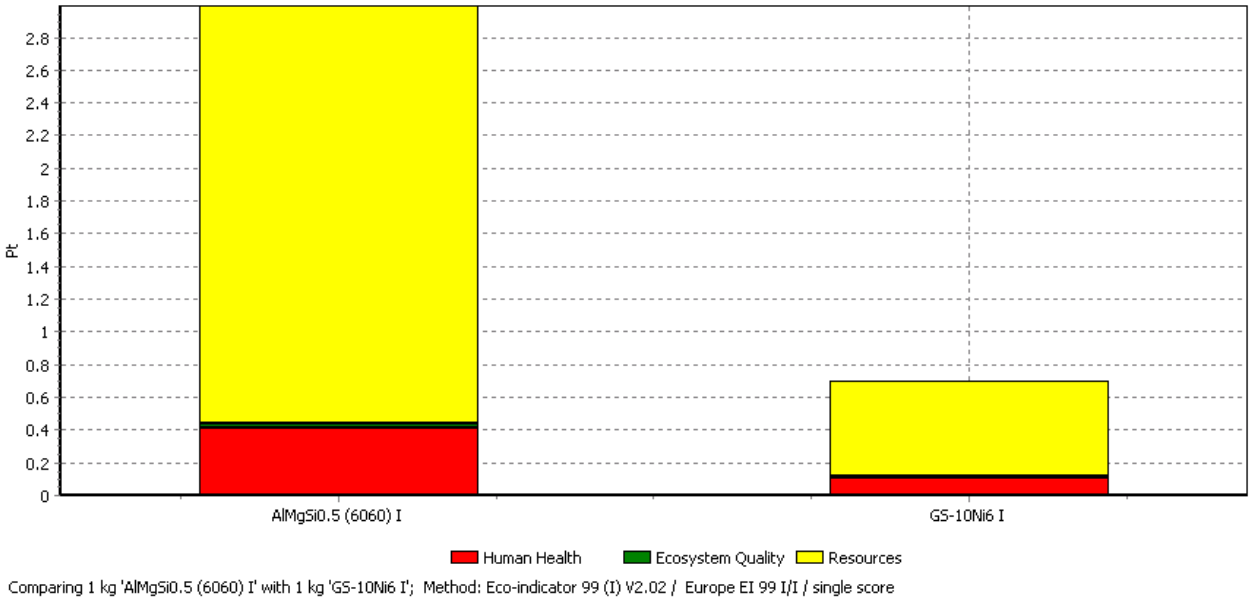


Figure G6: Aluminum has a higher EcoIndicator 99 score, showing that it is more detrimental for the environment

Appendix H – Sensor Specifications from Optimet

	Lens Assembly Type (By Focal Length in mm)*												
	Standard									High Definition			
ConoProbe Specifications	16	25	40	50	75	100	125 ext.*	150	250	16	25	40	50
Z (Vertical) axis													
Precision [1,4] (µm)	<2	<3	<4	<6	<10	<15	<20	<35	<100	<0.5	<1	<2.0	<2.5
Reproducibility 2σ[2] (µm)	<0.15	<0.4	<0.7	<1	<2	<4	<8	<15	<15	<0.1	<0.2	<0.4	<0.5
Working Range (mm)	0.6	1.8	4	8	18	35	45	70	180	0.2	0.6	1.4	2
Standoff [3] (mm)	12	15	40	42	65	90	240	140	240	11	14	37	40
Lateral Axes													
Laser Spot Size (X) [5] (µm)	11	22	30	45	65	75	100	120	220	3.5	6	10	15
Lateral Resolution (X)[6] (µm)	5	12	14	15	25	35	50	50	100	2	4	7	10
Weight													
Lens (g)	460	40	122	25	25	25	400	25	25	460	40	122	25
Probe (g)	700									700			
Control Box (g)	1200												
Data Handling													
Data Rate [7]	850pps												
Macros	Macro commands are provided to automate similar measurements												
Export Data to:	Excel, ASCII text file, BMP, JPEG, UBM, VRML												
Applications													
Precision for radius measurements [7]	Relative to lens accuracy												
Angle measurement [8]	170°												
Working temperature [9]	18 to 35°C												
Continuous shock resistance	245 m/s2-25g-6ms >6000 shocks 6 directions												
Supply Voltage	82-265 VAC 50-60Hz												

www.optimet.com

Appendix I – Sensor Code from Optimet, Edited by P2C2

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h> //for getch
#include "Multiple.h"
#define NUM_PROBES 2

CFifoWrapper *pFifo[NUM_PROBES];

void Cleanup(int iProbeNum, char *sFunctionName) // cleanup function before successful
or non-successful exit
{
if (sFunctionName == NULL) // NULL function name indicates success
printf("Program completed successfully.\n");
else // non-NULL function name indicates error
printf("Error %d in function %s: %s\n", pFifo[iProbeNum]->pfn_FifoGetError(),
sFunctionName, pFifo[iProbeNum]->pfn_FifoGetErrorString());

for (int i = 0; i < NUM_PROBES; i++)
{
	pFifo[i]->pfn_FifoTerminate();
}

delete [] *pFifo;
printf("Hit any key to exit...");
getch();
}

int main(int argc, char *argv[])
{
	TMeasurement M;
	int iPort;
	int i;//, j;
	long lHeadSN;
	int iMCUVersion, iMaxFrequency;
	char temp[100] = "";
	char LPstr[NUM_PROBES][100];
	FILE *LP[NUM_PROBES];
	char fileName[10];

	// Open output files
	for (i = 0; i < NUM_PROBES; i++)
	{
		itoa(i,temp, 10);
		strcpy(fileName, "bbcb");
		strcat(fileName, temp);
		strcat(fileName, ".csv");
		LP[i] = fopen(fileName, "a");
	}



	for (i = 0; i < NUM_PROBES; i++)
	{
		pFifo[i] = new CFifoWrapper(i);
		printf("%x \n",pFifo[i]);
```

```
	if ((pFifo[i] == NULL) || (!pFifo[i]->bComplete))
	{
		printf("Error in Fifo Wrapper construction %d\n", i);
		printf("Hit any key to exit...");
		getch();
		return EXIT_FAILURE;
	}
}

for (i = 0; i < NUM_PROBES; i++)
{
	printf("Enter port for probe %d (in hex).\n", i);
	scanf("%x",&(iPort));

	// use explicit port value
	// DO NOT use the function FifoDetect for multiple probes
	if (!pFifo[i]->pfn_FifoInitEx(iPort))
		{
		Cleanup(i, "FifoInitEx");
		return EXIT_FAILURE;
		}
}

printf("Probes initialized.\n");
//type(sName);

for (i = 0; i < NUM_PROBES; i++)
{
	lHeadSN = pFifo[i]->pfn_FifoGetHeadSN();

	if (pFifo[i]->pfn_FifoGetError() != loeNone)
	{
		Cleanup(i, "FifoGetHeadSN");
		return EXIT_FAILURE;
	}
	printf("Probe %d:\nSerial Number is: %ld\n", i, lHeadSN);
	iMCUVersion = pFifo[i]->pfn_FifoGetMCUVersion();

	if (pFifo[i]->pfn_FifoGetError() != loeNone)
	{
		Cleanup(i, "FifoGetMCUVersion");
		return EXIT_FAILURE;
	}
	printf("MCU version: %d\n", iMCUVersion);
	iMaxFrequency = pFifo[i]->pfn_FifoGetMaxFrequency();

	if (pFifo[i]->pfn_FifoGetError() != loeNone)
	{
		Cleanup(i, "FifoGetMaxFrequency");
		return EXIT_FAILURE;
	}
	printf("Max frequency: %d\n", iMaxFrequency);

	}

for (i = 0; i < NUM_PROBES; i++)

{
	if (!pFifo[i]->pfn_FifoBeginReadMeasurementStream(2000, mTime, eBoth))
	{
		Cleanup(i, "FifoBeginReadMeasurementStream");
		return EXIT_FAILURE;
	}
```

```
	}
	// j=0;
	while (!kbhit())
	{
		// take date stamp here
		// j+=1;
		for (i = 0; i < NUM_PROBES; i++)
		{

			if (pFifo[i]->pfn_FifoGetMeasurementStream(&M))
			{
				//printf("%f\tn=%i", M.Distance,j);
				sprintf(LPstr[i], "%f", M.Distance);
			}
			else
			{
				//printf("-\t\t");
				strcpy(LPstr[i], "-");
			}
			strcat(LPstr[i], ",n = ");
			// this next line wont be needed w/ datestamp
			// itoa(j,temp, 10);
			// temp will be replaced by string of date stamp
			strcat(LPstr[i], temp);
			strcat(LPstr[i], "\n");
		}
		for (i=0;i<NUM_PROBES; i++)
		{
			printf("Sensor: %i, Value: %s\n", i, LPstr[i]);
			fputs(LPstr[i], LP[i]);
		}
	}

	for (i = 0; i < NUM_PROBES; i++)
	{
		if (!pFifo[i]->pfn_FifoAbortMeasurements())
		{
			Cleanup(i, "FifoAbortMeasurements");
			return EXIT_FAILURE;
		}
	}
	printf("Test completed successfilly. \n");
	Cleanup(0, NULL); // NULL indicates success
	for (i = 0; i < NUM_PROBES; i++)
	{
		fclose(LP[i]);
	}

	return EXIT_SUCCESS;
}

// constructor
CFifoWrapper::CFifoWrapper(int iInstanceNumber)
{
	bComplete = false; // bComplete indicates that the constructor did not complete
successfully
	// load the dlls. Fifo.dll should be copied and renamed fifo0.dll, fifo1.dll,
etc.
	char sName[15];
	sprintf (sName, "fifo%d.dll", iInstanceNumber);
	hFifoHandle = LoadLibrary(sName);

	if (hFifoHandle == NULL) // check error status
```

```
	return; // return with bComplete = false, to indicate error status connect the
functions
	pfn_FifoInit = (PFN_FIFOINIT)GetProcAddress(hFifoHandle, "FifoInit");

	if (pfn_FifoInit == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoInitEx = (PFN_FIFOINITEX)GetProcAddress(hFifoHandle, "FifoInitEx");

	if (pfn_FifoInitEx == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoSetActiveLensIndex =
	(PFN_FIFOSETACTIVELENSINDEX)GetProcAddress(hFifoHandle,
"FifoSetActiveLensIndex");

	if (pfn_FifoSetActiveLensIndex == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoSetFrequency = (PFN_FIFOSETFREQUENCY)GetProcAddress(hFifoHandle,
	"FifoSetFrequency");

	if (pfn_FifoSetFrequency == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoSetInBitSet = (PFN_FIFOSETINBITSET)GetProcAddress(hFifoHandle,
	"FifoSetInBitSet");

	if (pfn_FifoSetInBitSet == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoSetLensZTweakUser =
	(PFN_FIFOSETLENSZTWEAKUSER)GetProcAddress(hFifoHandle,
"FifoSetLensZTweakUser");

	if (pfn_FifoSetLensZTweakUser == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoSetMeasurementFilter =
	(PFN_FIFOSETMEASUREMENTFILTER)GetProcAddress(hFifoHandle,
	"FifoSetMeasurementFilter");

	if (pfn_FifoSetMeasurementFilter == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoSetPower = (PFN_FIFOSETPOWER)GetProcAddress(hFifoHandle,
	"FifoSetPower");

	if (pfn_FifoSetPower == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoTerminate = (PFN_FIFOTERMINATE)GetProcAddress(hFifoHandle,
	"FifoTerminate");

	if (pfn_FifoTerminate == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoDetect = (PFN_FIFODETECT)GetProcAddress(hFifoHandle, "FifoDetect");

	if (pfn_FifoDetect == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoGetActiveLensIndex =
	(PFN_FIFOGETACTIVELENSINDEX)GetProcAddress(hFifoHandle,
"FifoGetActiveLensIndex");

	if (pfn_FifoGetActiveLensIndex == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoGetHeadSN = (PFN_FIFOGETHEADSN)GetProcAddress(hFifoHandle,
	"FifoGetHeadSN");

	if (pfn_FifoGetHeadSN == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
```

```
        pfn_FifoGetInBitSetAvailability =
        (PFN_FIFOGETINBITSETAVAILABILITY)GetProcAddress(hFifoHandle,
        "FifoGetInBitSetAvailability");

        if (pfn_FifoGetInBitSetAvailability == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoGetLens = (PFN_FIFOGETLENS)GetProcAddress(hFifoHandle, "FifoGetLens");

        if (pfn_FifoGetLens == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoGetLensCount = (PFN_FIFOGETLENSCOUNT)GetProcAddress(hFifoHandle,
        "FifoGetLensCount");

        if (pfn_FifoGetLensCount == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoGetLensZTweakUser =
        (PFN_FIFOGETLENSZTWEAKUSER)GetProcAddress(hFifoHandle,
"FifoGetLensZTweakUser");

        if (pfn_FifoGetLensZTweakUser == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoGetMaxFrequency = (PFN_FIFOGETMAXFREQUENCY)GetProcAddress(hFifoHandle,
        "FifoGetMaxFrequency");

        if (pfn_FifoGetMaxFrequency == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoGetMCUVersion = (PFN_FIFOGETMCUVERSION)GetProcAddress(hFifoHandle,
        "FifoGetMCUVersion");

        if (pfn_FifoGetMCUVersion == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoGetMCUVersionRequired =
        (PFN_FIFOGETMCUVERSIONREQUIRED)GetProcAddress(hFifoHandle,
        "FifoGetMCUVersionRequired");

        if (pfn_FifoGetMCUVersionRequired == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoGetPower = (PFN_FIFOGETPOWER)GetProcAddress(hFifoHandle,
        "FifoGetPower");

        if (pfn_FifoGetPower == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoAbortMeasurements =
        (PFN_FIFOABORTMEASUREMENTS)GetProcAddress(hFifoHandle,
"FifoAbortMeasurements");

        if (pfn_FifoAbortMeasurements == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoBeginReadMeasurementStream =
        (PFN_FIFOBEGINREADMEASUREMENTSTREAM)GetProcAddress(hFifoHandle,
        "FifoBeginReadMeasurementStream");

        if (pfn_FifoBeginReadMeasurementStream == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoBeginReadMeasurements =
        (PFN_FIFOBEGINREADMEASUREMENTS)GetProcAddress(hFifoHandle,
        "FifoBeginReadMeasurements");

        if (pfn_FifoBeginReadMeasurements == NULL) // check error status
        return; // return with bComplete = false, to indicate error status
        pfn_FifoGetMeasurementStream =
        (PFN_FIFOGETMEASUREMENTSTREAM)GetProcAddress(hFifoHandle,
        "FifoGetMeasurementStream");
```

```
	if (pfn_FifoGetMeasurementStream == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoGetMeasurements = (PFN_FIFOGETMEASUREMENTS)GetProcAddress(hFifoHandle,
	"FifoGetMeasurements");

	if (pfn_FifoGetMeasurements == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoGetMeasurementsCount =
	(PFN_FIFOGETMEASUREMENTSCOUNT)GetProcAddress(hFifoHandle,
	"FifoGetMeasurementsCount");

	if (pfn_FifoGetMeasurementsCount == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoReadMeasurement = (PFN_FIFOREADMEASUREMENT)GetProcAddress(hFifoHandle,
	"FifoReadMeasurement");

	if (pfn_FifoReadMeasurement == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoReadMeasurementStream =
	(PFN_FIFOREADMEASUREMENTSTREAM)GetProcAddress(hFifoHandle,
	"FifoReadMeasurementStream");

	if (pfn_FifoReadMeasurementStream == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoGetError = (PFN_FIFOGETERROR)GetProcAddress(hFifoHandle,
	"FifoGetError");

	if (pfn_FifoGetError == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoGetErrorString = (PFN_FIFOGETERRORSTRING)GetProcAddress(hFifoHandle,
	"FifoGetErrorString");

	if (pfn_FifoGetErrorString == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoGetTimeOut = (PFN_FIFOGETTIMEOUT)GetProcAddress(hFifoHandle,
	"FifoGetTimeOut");

	if (pfn_FifoGetTimeOut == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_FifoSetTimeOut = (PFN_FIFOSETTIMEOUT)GetProcAddress(hFifoHandle,
	"FifoSetTimeOut");

	if (pfn_FifoSetTimeOut == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	pfn_ShowProbeDialog = (PFN_FIFOSHOWPROBEDIALOG)GetProcAddress(hFifoHandle,
	"ShowProbeDialog");

	if (pfn_ShowProbeDialog == NULL) // check error status
	return; // return with bComplete = false, to indicate error status
	bComplete = true; // completed successfully
}

// destructor
CFifoWrapper::~CFifoWrapper()
{
	FreeLibrary(hFifoHandle);
}
```

Sensor Headers Code

```
#ifndef __FIFO_MULTIPLE_INSTANCES
#define __FIFO_MULTIPLE_INSTANCES
#include <windows.h> // support for LoadLibrary & GetProcAddress
#include "Fifo.h"
class CFifoWrapper

{

private:

// handle for the fifo dll
HMODULE hFifoHandle;

// define type definitions for each function type
typedef BYTE (_stdcall *PFN_FIFOINIT)(TLptPort Port) ;
typedef BYTE (_stdcall *PFN_FIFOINITEX)(int Port);
typedef BYTE (_stdcall *PFN_FIFOSETACTIVELENSINDEX)(TLensIndex);
typedef BYTE (_stdcall *PFN_FIFOSETFREQUENCY)(int Frequency);
typedef BYTE (_stdcall *PFN_FIFOSETINBITSET)(int InBitSet);
typedef BYTE (_stdcall *PFN_FIFOSETLENSZTWEAKUSER)(TLensIndex Index, short
ZTweakUser);
typedef BYTE (_stdcall *PFN_FIFOSETMEASUREMENTFILTER)(TMeasurementFilter
MeasurementFilter, WORD wMinValue, WORD wMaxValue);
typedef BYTE (_stdcall *PFN_FIFOSETPOWER)(TPower Power, WORD Value);
typedef void (_stdcall *PFN_FIFOTERMINATE)(void);
typedef TLptPort (_stdcall *PFN_FIFODETECT)(void);
typedef TLensIndex (_stdcall *PFN_FIFOGETACTIVELENSINDEX)(void);
typedef long (_stdcall *PFN_FIFOGETHEADSN)(void);
typedef BYTE (_stdcall *PFN_FIFOGETINBITSETAVAILABILITY)(int InBitSet);
typedef BYTE (_stdcall *PFN_FIFOGETLENS)(TLensIndex Index, TLens *Lens);
typedef TLensIndex (_stdcall *PFN_FIFOGETLENSCOUNT)(void);
typedef short (_stdcall *PFN_FIFOGETLENSZTWEAKUSER)(TLensIndex Index);
typedef int (_stdcall *PFN_FIFOGETMAXFREQUENCY)(void);
typedef int (_stdcall *PFN_FIFOGETMCUVERSION)(void);
typedef int (_stdcall *PFN_FIFOGETMCUVERSIONREQUIRED)(void);
typedef WORD (_stdcall *PFN_FIFOGETPOWER)(TPower Power);
typedef int (_stdcall *PFN_FIFOABORTMEASUREMENTS)(void);
typedef BYTE (_stdcall *PFN_FIFOBEGINREADMEASUREMENTSTREAM)(int
DelayBetweenMeasurements,
TMode DelayBetweenMeasurementsMode,TEdges Edges);
typedef BYTE (_stdcall *PFN_FIFOBEGINREADMEASUREMENTS)(
WORD nMeasurements,
WORD Wait,
TMode WaitMode,
int DelayBetweenMeasurements,
TMode DelayBetweenMeasurementsMode,
TEdges Edges);
typedef BYTE (_stdcall *PFN_FIFOGETMEASUREMENTSTREAM)(TMeasurement
*Measurement);
typedef BYTE (_stdcall *PFN_FIFOGETMEASUREMENTS)(TMeasurement *Measurements,
WORD nMeasurements);
typedef int (_stdcall *PFN_FIFOGETMEASUREMENTSCOUNT)(void);
typedef BYTE (_stdcall *PFN_FIFOREADMEASUREMENT)(TMeasurement *Measurement);
typedef BYTE (_stdcall *PFN_FIFOREADMEASUREMENTSTREAM)(int
DelayBetweenMeasurements,
TMode DelayBetweenMeasurementsMode,
TBeginMeasurementStreamEvent BeginMeasurementStreamEvent,
TMeasurementStreamEvent MeasurementStreamEvent,
TFinishMeasurementStreamEvent FinishMeasurementStreamEvent,
TEdges Edges);
```

```
typedef TLoadError (_stdcall *PFN_FIFOGETERROR)(void);
typedef char *(_stdcall *PFN_FIFOGETERRORSTRING)(void);
typedef int (_stdcall *PFN_FIFOGETTIMEOUT)(void);
typedef void (_stdcall *PFN_FIFOSETTIMEOUT)(int TimeOut);
typedef BYTE (_stdcall *PFN_FIFOSHOWPROBEDIALOG)(BYTE Modal, TParameters
*Parameters, TResults *Results);
public:

// function types for each function
PFN_FIFOINIT pfn_FifoInit;
PFN_FIFOINITEX pfn_FifoInitEx;
PFN_FIFOSETACTIVELENSINDEX pfn_FifoSetActiveLensIndex;
PFN_FIFOSETFREQUENCY pfn_FifoSetFrequency;
PFN_FIFOSETINBITSET pfn_FifoSetInBitSet;
PFN_FIFOSETLENSZTWEAKUSER pfn_FifoSetLensZTweakUser;
PFN_FIFOSETMEASUREMENTFILTER pfn_FifoSetMeasurementFilter;
PFN_FIFOSETPOWER pfn_FifoSetPower;
PFN_FIFOTERMINATE pfn_FifoTerminate;
PFN_FIFODETECT pfn_FifoDetect;
PFN_FIFOGETACTIVELENSINDEX pfn_FifoGetActiveLensIndex;
PFN_FIFOGETHEADSN pfn_FifoGetHeadSN;
PFN_FIFOGETINBITSETAVAILABILITY pfn_FifoGetInBitSetAvailability;
PFN_FIFOGETLENS pfn_FifoGetLens;
PFN_FIFOGETLENSCOUNT pfn_FifoGetLensCount;
PFN_FIFOGETLENSZTWEAKUSER pfn_FifoGetLensZTweakUser;
PFN_FIFOGETMAXFREQUENCY pfn_FifoGetMaxFrequency;
PFN_FIFOGETMCUVERSION pfn_FifoGetMCUVersion;
PFN_FIFOGETMCUVERSIONREQUIRED pfn_FifoGetMCUVersionRequired;
PFN_FIFOGETPOWER pfn_FifoGetPower;
PFN_FIFOABORTMEASUREMENTS pfn_FifoAbortMeasurements;
PFN_FIFOBEGINREADMEASUREMENTSTREAM pfn_FifoBeginReadMeasurementStream;
PFN_FIFOBEGINREADMEASUREMENTS pfn_FifoBeginReadMeasurements;
PFN_FIFOGETMEASUREMENTSTREAM pfn_FifoGetMeasurementStream;
PFN_FIFOGETMEASUREMENTS pfn_FifoGetMeasurements;
PFN_FIFOGETMEASUREMENTSCOUNT pfn_FifoGetMeasurementsCount;
PFN_FIFOREADMEASUREMENT pfn_FifoReadMeasurement;
PFN_FIFOREADMEASUREMENTSTREAM pfn_FifoReadMeasurementStream;
PFN_FIFOGETERROR pfn_FifoGetError;
PFN_FIFOGETERRORSTRING pfn_FifoGetErrorString;
PFN_FIFOGETTIMEOUT pfn_FifoGetTimeOut;
PFN_FIFOSETTIMEOUT pfn_FifoSetTimeOut;
PFN_FIFOSHOWPROBEDIALOG pfn_ShowProbeDialog;

// constructor/destructor
CFifoWrapper(int iInstanceNumber);
virtual ~CFifoWrapper();

// parameter to return error status
bool bComplete;

};

#endif // __FIFO_MULTIPLE_INSTANCES
```

```
/*-------------------------------------------------------------------------*/
/*                                                                         */
/*       Optimet Conoprobe Software - Fifo Import Library                  */
/*       Copyright (c) 1998-2001 Optimet Ltd.                              */
/*                                                                         */
/*       NOTE: to support DOS version (built in Watcom), add:              */
/*             #define WATCOM_DOS                                        */
/*-------------------------------------------------------------------------*/
```

```
#ifndef FifoH
#define FifoH
/*------------------------------------------------------------------------------*/

/*---------------------------------- Types -------------------------------------*/
typedef unsigned char BYTE; /* Byte type definition */
typedef unsigned short WORD; /* Word type definition */
#if defined (WATCOM_DOS) || defined (BORLAND_DOS)
#define _stdcall
#else
#pragma pack(push, packBefore)
#endif
#pragma pack(1)
/* Parallel port via which PC will communicate with EC1000 */
typedef WORD TLptPort;
typedef enum {lpLPT1 = 0x378, lpLPT2 = 0x278} LptPort;

/* Distinction between coarse power level of
   the Conoprobe's laser and its fine tuning */
typedef BYTE TPower;
typedef enum {pFine = 1, pCoarse = 2} Power;

/* Distinguish between set measurement frequency (mTime),
   and measurement driven by external trigger (mPulse),
   and measurement at set frequency but returned at next external trigger
(mTimeTrigger) */
typedef BYTE TMode;
typedef enum {mTime, mPulse, mTimeTrigger} Mode;

/* Index of lens whose calibration factors should be used
   to calculate distance (between 1..31) */
typedef BYTE TLensIndex;

/* Allows programmer to select edge to trigger
   on when measuring in external trigger mode */
typedef WORD TEdges;
typedef enum {eBoth = 0, eFalling = 0x4000, eRising = 0x8000} Edges;

/* System Error codes */
typedef BYTE TLoadError;
typedef enum {loeNone,           /* No error */
              loeCommunication,  /* Communication error occurred */
              loeParam,          /* Parameter error */
              loeTimeOut,        /* EC1000 failed to respond within timeout */
              loeOldMCUVersion,   /* Requested functionality not supported in this
version of the MCU */
              loeUncalibratedRom /* Requested functionality not available in the ROM
*/
} LoadError;

/* Lens structure returned by EC1000 to PC */
typedef struct {
       short FocalDistance; /* Distance from lens (in millimeters) to focal
                            point of laser beam emitted by ConoProbe */
       char Type; /* Type: standard, telescopic, high precision (between a-z/A-Z) */
       char Iteration; /* Number of times calibrated with this lens */
       /* LensID = FocalDistance + Type + Iteration */
       struct {
              BYTE Day;
              BYTE Month;
              WORD Year;
       } DateOfCalibration; /* Date of calibration with this lens */
       float DistanceMin; /* Minimum distance which can be measured with this lens */
```

```
      float DistanceMax; /* Maximum distance which can be measured with this lens */
      /* Range = DistanceMax - DistanceMin */
      short Power; /* Power (coarse) level when calibrating this lens */
      short FinePower; /* Power (fine) level when calibrating this lens */
      short MCUVersion; /* MCU version when calibrating this lens */
       short TemperatureTenthsDegree; // Room Temerature in Tenths of a degree
      BYTE Reserved[232];
} TLens;

/* Measurement structure returned by EC1000 to PC */
typedef struct {
      union {
               float  Frequency;  /*  Signal  described  in  terms  of  power  spectrum
distribution */
               struct {         /* Number of pulses in X & Y */
                   WORD PulseX;
                   WORD PulseY;
               }Pulse;
               float RDistance; /* Rough distance measured in millimeters */
       }Freq;
       float Distance; /* Distance measured in millimeters */
       WORD Snr; /* Signal quality (between 0..1024) */
       short Sat; /* For in-house test purposes only */
       WORD Total; /* For in-house test purposes only */
          BYTE BitSet; /* Measurement error flag, as follows:
                   0 = no error
                   2 = too close to target or bad signal
                   4 = too far from target or signal is too noisy
           FF = unknown */
       BYTE Tag; /* Tag of measurement */
} TMeasurement;

/* Begin measurement stream event function prototype */
typedef void (_stdcall *TBeginMeasurementStreamEvent)(void);
/* Measurement stream event function prototype */
typedef  BYTE  (_stdcall  *TMeasurementStreamEvent)(int  nMeasurement,  TMeasurement*
Measurement);
/* Finish measurement stream event function prototype */
typedef  void  (_stdcall  *TFinishMeasurementStreamEvent)(int  MeasurementsCount,  int
MeasurementsDeliveredCount);

typedef BYTE TMeasurementFilter;
typedef enum {mfNone, mfSnr, mfTotal} MeasurementFilter; // TMeasurementFilter

#if !defined (WATCOM_DOS) && !defined (BORLAND_DOS)
typedef BYTE TLanguage;
typedef enum {lEnglish, lFrench, lGerman} Language; // TLanguage

typedef BYTE TUnits;
typedef enum {uMM = 0, uMils = 1, uMicroM = 2} Units; // TUnits

typedef BYTE TGraphType;
typedef enum {gtNone, gtRawA, gtRawB, gtRawAB, gtRawAMinusB, gtRawAPlusB, gtContrast,
gtFrequency} GraphType; // TGraphType

typedef  int  (_stdcall  *TDlgProc)(void  *hWnd,  int  Msg,  int  wParam,  int  lParam);  //
Callback function prototype

/* TParameters structure used by ShowProbeDialog() function */
typedef struct {
       TLanguage Language;
       TUnits Units;
       TGraphType GraphType;
```

```
        BYTE UseExStatisticsView;
        BYTE UseExGraphView;
        BYTE UseAO;
        short Power;
        short FinePower;
        short Frequency;
        TDlgProc DlgProc;
} TParameters;

/* TResults structure used by ShowProbeDialog() function */
typedef struct {
        short Power;
        short FinePower;
        int Frequency;
} TResults;


#pragma pack(pop, packBefore)
#endif
/*------------------------------- Constants -------------------------------*/
const TLensIndex LensCountMax = 31;
const WORD MeasurementCountMax = 7000;
const WORD PowerMax = 63;
const short ZTweakUserMax = 30000;

// parameters for SetInBitSet & GetInBitSetAvailability
const int iIN_BIT_SET_RDE          = 1;
const int iIN_BIT_SET_XY           = 2;
const int iIN_BIT_SET_NOTCH_FILTER = 4;

#ifdef __cplusplus
extern "C" {
#endif
/*-------------------------------- Functions --------------------------------*/
/*------------------------------ Setup functions ----------------------------*/
/*
      Function Name : FifoInit
      Purpose       : Initialize communication with EC1000
      Parameters    : TLptPort
      Returns       : True if succeeded in communicating with EC1000
      See Also      : FifoDetect, FifoTerminate
*/
BYTE _stdcall FifoInit(TLptPort Port);

/*
      Function Name : FifoInitEx
      Purpose       : Initialize communication with EC1000
      Parameters    : Int value for the LPT port
      Returns       : True if succeeded in communicating with EC1000
      See Also      : FifoDetect, FifoTerminate
*/
BYTE _stdcall FifoInitEx(int Port);

/*
      Function Name : FifoSetActiveLensIndex
      Purpose       : Configure EC1000 to calculate distance based on
                              specified lens calibration data
      Parameters    : TLensIndex
      Returns       : True if succeeded, otherwise returns False
      See Also      : FifoGetActiveLensIndex, FifoGetLens, FifoGetLensCount
*/
BYTE _stdcall FifoSetActiveLensIndex(TLensIndex Index);
```

```
/*
      Function Name : FifoSetFrequency
      Purpose       : Set EC1000's internal measurement frequency for work in
                              single capture mode
      Parameters    : Frequency
      Returns       : True if succeeded, otherwise returns False
      See Also      : FifoGetMaxFrequency, FifoGetMCUVersion, FifoReadMeasurement
*/
BYTE _stdcall FifoSetFrequency(int Frequency);

/*
      Function Name : FifoSetInBitSet
      Purpose       : Set EC1000's measurement structure field data
      Parameters    : InBitSet: Values to return (may be OR'ed together)
                              0 = frequency (default value)
                              iIN_BIT_SET_RDE   =   return   rough   distance   (this
overrides frequency)
                  iIN_BIT_SET_XY = return XY Values (this overrides rough distance)
                  iIN_BIT_SET_NOTCH_FILTER = Activate Notch Filter
      Returns       : True if succeeded, otherwise returns False
    Error Values  : loeOldMCUVersion, loeUncalibratedRom
      See Also        : TMeasurement, FifoBeginReadMeasurements, FifoReadMeasurement,
FifoReadMeasurementStream
*/
BYTE _stdcall FifoSetInBitSet(int InBitSet);

/*
      Function Name : FifoSetLensZTweakUser
      Purpose       : Set to specified lens multiplier distance factor (±30%)
                              Distance(factor)   =   Distance(measured)   *   (1   +
ZTweakUser / 100000)
                              specified lens calibration data
      Parameters    : Index - lens index which ZTweakUser value want to be changed
                              ZTweakUser (±30000) - new multiplier distance factor
to set (0 will disable any factor)
      Returns       : True if succeeded, otherwise returns False
      See Also      : FifoGetLensZTweakUser
*/
BYTE _stdcall FifoSetLensZTweakUser(TLensIndex Index, short ZTweakUser);

/*
      Function Name : FifoSetMeasurementFilter
      Purpose       : Activate filters on the returned TMeasurement structure
      Parameters    : mfFilter - mfNone (disable all filters), mfSnr, mfTotal
                              wMinValue, wMaxValue - the boundaries for the filter
                  (For SNR filter, the limits are between 0 - 1023.
                   For Total filter, the limits are between 0 - 32767.)
      Returns       : True if succeeded, otherwise returns False
      See Also      :
*/
BYTE  _stdcall  FifoSetMeasurementFilter(TMeasurementFilter  MeasurementFilter,  WORD
wMinValue, WORD wMaxValue);

/*
      Function Name : FifoSetPower
      Purpose       : Set Conoprobes laser power level
      Parameters    : Power - pFine for fine tuning, pCoarse for coarse power level
                              Value - power level (between 0..63)
      Returns       : True if succeeded, otherwise returns False
      See Also      : FifoGetPower
*/
BYTE _stdcall FifoSetPower(TPower Power, WORD Value);
```

```
/*
      Function Name : FifoTerminate
      Purpose       : Close down communication between PC and EC1000
      Parameters    : None
      Returns       : None
      See Also      : FifoInit
*/
void _stdcall FifoTerminate(void);

/*--------------------------- Information functions --------------------------*/
/*
      Function Name : FifoDetect
      Purpose       : Auto detect parallel port that EC1000 is connected to
      Parameters    : None
      Returns       : TLptPort
      See Also      : FifoInit
*/
TLptPort _stdcall FifoDetect(void);

/*
      Function Name : FifoGetActiveLensIndex
      Purpose       : Query to EC1000 which lens data is presently being used
                              for distance calculation
      Parameters    : None
      Returns       : Index of lens calibration data presently in use
      See Also      : FifoGetLens, FifoGetLensCount, FifoSetActiveLensIndex
*/
TLensIndex _stdcall FifoGetActiveLensIndex(void);

/*
      Function Name : FifoGetInBitSetAvailability
      Purpose           : Check which features from the SetInBitSet function are
available for this MCU and are calibrated in the Lens information in the ROM
      Parameters    : InBitSet. List of flags for InBitSet OR'ed together
    Returns       : True if succeeded, otherwise returns False
    Error Values  : loeOldMCUVersion, loeUncalibratedRom
      See Also      : FifoSetInBitSet
*/

BYTE _stdcall FifoGetInBitSetAvailability(int InBitSet);

/*
      Function Name : FifoGetLens
      Purpose       : Upload from EC1000 data for a specific lens
      Parameters    : TLensIndex
                    Pointer to TLens structure
      Returns       : True if succeeded, otherwise returns False
      See   Also                  :   FifoGetActiveLensIndex,   FifoGetLensCount,
FifoSetActiveLensIndex
*/
BYTE _stdcall FifoGetLens(TLensIndex Index, TLens *Lens);

/*
      Function Name : FifoGetLensCount
      Purpose       : Query EC1000 for total number of lenses that have been
                              calibrated with it
      Parameters    : None
      Returns       : TLensIndex : Count of lens data stored in ConoProbe
      See Also      : FifoGetActiveLensIndex, FifoGetLens, FifoSetActiveLensIndex
*/
TLensIndex _stdcall FifoGetLensCount(void);

/*
```

```
      Function Name : FifoGetLensZTweakUser
      Purpose       : Upload from EC1000 ZTweakUser value for a specific lens
      Parameters    : TLensIndex
      Returns       : ZTweakUser value for specified lens
      See Also      : FifoSetLensZTweakUser
*/
short _stdcall FifoGetLensZTweakUser(TLensIndex Index);

/*
      Function Name : FifoGetMaxFrequency
      Purpose       : Query EC1000 for its maximum measurement frequency
      Parameters    : None
      Returns       : Maximum working frequency possible on the EC1000
      See Also      : FifoSetFrequency, FifoBeginReadMeasurements
*/
int _stdcall FifoGetMaxFrequency(void);

/*
      Function Name : FifoGetMCUVersion
      Purpose       : Query EC1000 for version of software in its microcontroller
      Parameters    : None
      Returns       : MCU version
      See Also      : FifoGetMCUVersionRequired
*/
int _stdcall FifoGetMCUVersion(void);

/*
      Function Name : FifoGetMCUVersionRequired
      Purpose        : Query Fifo library for MCU version required for using library
correctly
      Parameters    : None
      Returns       : MCU version
      See Also      : FifoGetMCUVersion
*/
int _stdcall FifoGetMCUVersionRequired(void);

/*
      Function Name : FifoGetPower
      Purpose       : Query EC1000 for present setting of its laser power
      Parameters    : Power - pFine for fine tuning, pCoarse for coarse power level
      Returns       : Power level (between 0..63)
      See Also      : FifoSetPower
*/
WORD _stdcall FifoGetPower(TPower Power);

/*--------------------------- Measurement functions --------------------------*/
/*
      Function Name : FifoAbortMeasurements
      Purpose       : To force EC1000 to abandon measurements
      Parameters    : None
      Returns       : Number of measurements performed before aborting
      See   Also                :   FifoBeginReadMeasurements,   FifoGetMeasurements,
FifoGetMeasurementsCount
*/
int _stdcall FifoAbortMeasurements(void);

/*
      Function Name : FifoBeginReadMeasurementStream
      Purpose       : To put EC1000 in stream measurement mode
      Parameters       :  DelayBetweenMeasurements  -  number  of  microseconds/pulses
between measurements
                                DelayBetweenMeasurementsMode - delay in microseconds
or pulses
```

```
                                Edges - trigger on rising, falling, or both edges
      Returns       : True if succeeded, otherwise returns False
      See Also      : FifoAbortMeasurements, FifoGetMeasurementStream
*/
BYTE _stdcall FifoBeginReadMeasurementStream(int DelayBetweenMeasurements,
      TMode DelayBetweenMeasurementsMode,
      TEdges Edges); // DelayBetweenMeasurements in µsec

/*
      Function Name : FifoBeginReadMeasurements
      Purpose       : To put EC1000 in multiple measurement mode
      Parameters      : nMeasurements - number of measurements requestes (between
2..7000)
                                Wait - delay before beginning measurement process in
milliseconds/pulses
                                WaitMode - delay mode before beginning (number of
milliseconds or number of pulses)
                                DelayBetweenMeasurements      -      number      of
microseconds/pulses between measurements
                                DelayBetweenMeasurementsMode - delay in microseconds
or pulses
                                Edges - trigger on rising, falling, or both edges
      Returns       : True if succeeded, otherwise returns False
      See   Also                  :   FifoAbortMeasurements,   FifoGetMeasurements,
FifoGetMeasurementsCount, FifoSetInBitSet
*/
BYTE _stdcall FifoBeginReadMeasurements
       (WORD nMeasurements,
        WORD Wait,
        TMode WaitMode,
        int DelayBetweenMeasurements,
        TMode DelayBetweenMeasurementsMode,
        TEdges Edges);

/*
      Function Name : FifoGetMeasurementStream
      Purpose        : Upload measurement from EC1000 (FIFO queque) when EC1000 is in
stream measurement mode
      Parameters    : Pointer to TMeasurement
      Returns       : True if succeeded, otherwise returns False
      See Also      : FifoAbortMeasurements, FifoBeginReadMeasurementStream
*/
BYTE _stdcall FifoGetMeasurementStream(TMeasurement *Measurement);

/*
      Function Name : FifoGetMeasurements
      Purpose       : Upload measurement buffer from EC1000
      Parameters    : Pointer to TMeasurement, number of measurements to upload
      Returns       : True if succeeded, otherwise returns False
      See   Also                :   FifoAbortMeasurements,   FifoBeginReadMeasurements,
FifoGetMeasurementsCount, FifoSetInBitSet
*/
BYTE _stdcall FifoGetMeasurements(TMeasurement *Measurements, WORD nMeasurements);

/*
      Function Name : FifoGetMeasurementsCount
      Purpose       : Query EC1000 for total number of measurements processed
      Parameters    : None
      Returns       : Number of measurements
      See   Also                :   FifoAbortMeasurements,   FifoBeginReadMeasurements,
FifoGetMeasurements
*/
int _stdcall FifoGetMeasurementsCount(void);
```

```
/*
      Function Name : FifoReadMeasurement
      Purpose       : Instruct EC1000 to perform single capture and upload result
      Parameters    : Pointer to TMeasurement
      Returns       : True if succeeded, otherwise returns False
      See Also      : FifoSetFrequency, FifoSetInBitSet
*/
BYTE _stdcall FifoReadMeasurement(TMeasurement *Measurement);

/*
      Function Name : FifoReadMeasurementStream
      Purpose       : Upload measurements from EC1000 in stream (real-time)
      Instruct EC1000 to perform single capture and upload result
      Parameters       :  DelayBetweenMeasurements  -  number  of  microseconds/pulses
between measurements
                                DelayBetweenMeasurementsMode - delay in microseconds
or pulses
                                BeginMeasurementStreamEvent   -   begin   measurement
stream event function (may be NULL)
                                MeasurementStreamEvent  -  measurement  stream  event
function
                                FinishMeasurementStreamEvent  -  finish  measurement
stream event function (may be NULL)
                                Edges - trigger on rising, falling, or both edges
      Returns       : True if succeeded, otherwise returns False
      See Also      : FifoSetInBitSet, FifoSetTimeOut
*/
BYTE    _stdcall    FifoReadMeasurementStream(int    DelayBetweenMeasurements,    TMode
DelayBetweenMeasurementsMode,
        TBeginMeasurementStreamEvent BeginMeasurementStreamEvent,
        TMeasurementStreamEvent MeasurementStreamEvent,
        TFinishMeasurementStreamEvent FinishMeasurementStreamEvent,
        TEdges Edges);

/*-------------------------- Error handling functions ------------------------*/
/*
      Function Name : FifoGetError
      Purpose       : Get last error status
      Parameters    : None
      Returns       : Last error
      See Also      : FifoGetErrorString
*/
TLoadError _stdcall FifoGetError(void);

/*
      Function Name : FifoGetErrorString
      Purpose       : Get last error information string
      Parameters    : None
      Returns       : Error string
      See Also      : FifoGetError
*/
char *_stdcall FifoGetErrorString(void);

/*------------------------- TimeOut handling functions ----------------------*/
/*
      Function Name : FifoGetTimeOut
      Purpose       : Get communication timeout parameter (in seconds)
      Parameters    : None
      Returns       : Timeout value
      See Also      : FifoSetTimeOut
*/
int _stdcall FifoGetTimeOut(void);
```

```
/*
      Function Name : FifoSetTimeOut
      Purpose       : Set communication timeout parameter (in seconds)
      Parameters    : Timeout value
      Returns       : None
      See Also      : FifoGetTimeOut
*/
void _stdcall FifoSetTimeOut(int TimeOut);

/*----------------------------- Other functions ------------------------------*/
#if !defined (WATCOM_DOS) && !defined (BORLAND_DOS)
/*
      Function Name : ShowProbeDialog
      Purpose       : Shows probe dialog (like in Scanner program)
                    but user can customize it
      Parameters     : Modal - if True shows as dialog (modal), otherwise as window
(non-modal)
                    Parameters - pointer to TParameters structure
                    Results - pointer to TResults structure
      Returns       : True if user clicks OK, otherwise returns False
      See Also      : FifoInit, FifoSetActiveLensIndex
*/
BYTE _stdcall ShowProbeDialog(BYTE Modal, TParameters *Parameters, TResults *Results);
#endif

#ifdef __cplusplus
}
#endif
#endif
```

Appendix J – Amplitude Calculations

Before the shaft starts to spin, P2C2 will initialize the sensors. The sensors will take static x- and y- measurements, x_0 and y_0, and use them and the shaft radius, r, to determine how far the initial center of the shaft is from each sensor, x_{co} and y_{co}.

To calculate the amplitude of displacement at a given point in time, the following equations will be applied to the x- and y-measurements, x and y, from that given point in time.

Calculate x_{leg} and y_{leg}.

$x_{leg} = x_{co} - x$

$y_{leg} = y_{co} - y$

Plug x_{leg} and y_{leg} into equation to determine the coordinates of the center of the circle.
Put coordinates (x_{leg}, 0) and (0, y_{leg}) into circle equation.

$(x - x_{leg})^2 + y^2 = r^2$

$x^2 + (y - y_{leg})^2 = r^2$

Expand.

$x^2 - 2xx_{leg} + x_{leg}^2 + y^2 = r^2$

$x^2 + y^2 - 2yy_{leg} + y_{leg}^2 = r^2$

Subtract one from the other.

$-2xx_{leg} + x_{leg}^2 + 2yy_{leg} - y_{leg}^2 = 0$

Isolate y.

$y = (2xx_{leg} - x_{leg}^2 + y_{leg}^2)/(2y_{leg})$

Plug y back into circle equation and solve for two possible coordinates of the center of the circle [(x_1, y_1) or (x_2, y_2)].

$(x - x_{leg})^2 + [(2xx_{leg} - x_{leg}^2 + y_{leg}^2)/(2y_{leg})]^2 = r^2$

Apply the Pythagorean theorem to the two possible coordinates to obtain the amplitudes (a_1 and a_2).

$a_1^2 = x_1^2 + y_1^2$

$a_2^2 = x_2^2 + y_2^2$

The amplitude less than r is the value to be used for plotting.

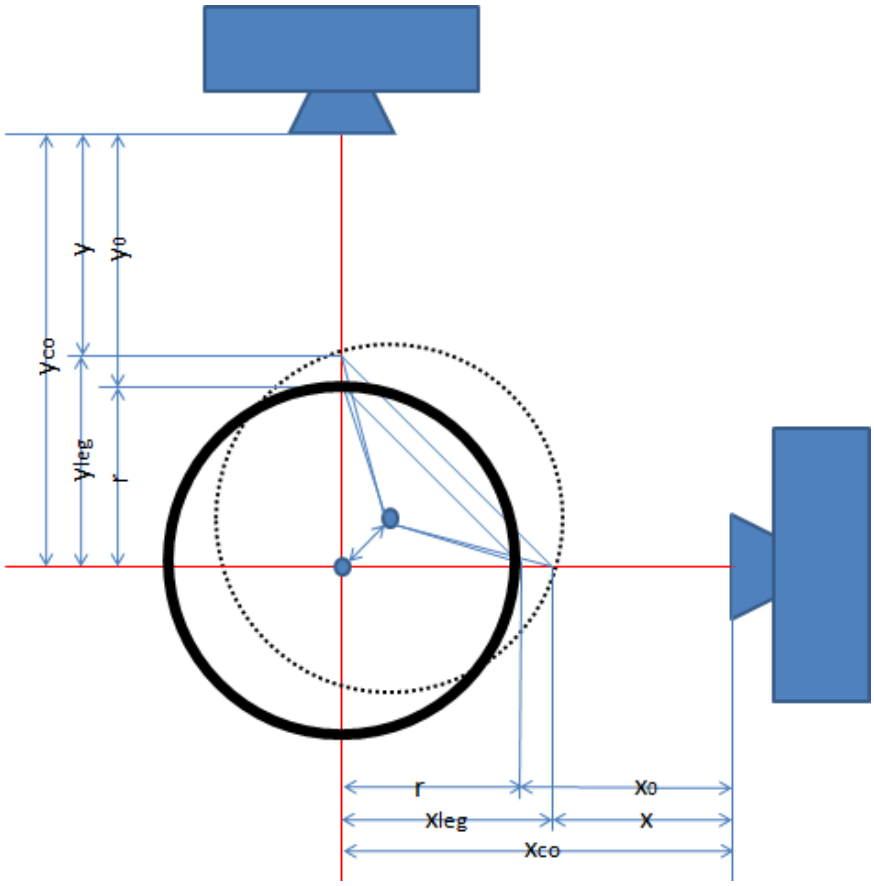


Figure J1: Geometry of amplitude calculation

Appendix K – Motor Specifications from Aerotech

ATS1000 Series		ATS10040	ATS10045	ATS10050	ATS10060
Total Travel		400 mm	450 mm	500 mm	600 mm
Drive System		Precision Ball Screw/Brushless Servomotor (BMS60-A-D25-E2000H)			
Bus Voltage		Up to 160 VDC			
Continuous Current	A_{pk}	Up to 2.3 A			
	A_{rms}	Up to 1.6 A			
Feedback		Noncontact Rotary Encoder (2000 line)			
Resolution	8 mm/rev lead	1.0 µm @ 2000 steps/rev Motor Resolution			
Maximum Travel Speed[1]		400 mm/s			
Maximum Load[2]		50 kg (horizontal); 25 kg (vertical); 50 kg (side)			
Accuracy	Standard	±16 µm	±18 µm	±20 µm	±24 µm
	HALAR[3]	±1.0 µm			
Repeatability (Bidirectional)		±1.0 µm standard; ±0.5 µm with HALAR[3]			
Straightness and Flatness		±9.6 µm	±10.8 µm	±12.0 µm	±14.4 µm
Nominal Stage Weight (with motor)	ATS1000	10.4 kg (22.9 lb)	11.0 kg (24.2 lb)	NA	12.6 kg (27.7 lb)
	ATS1000-B	12.4 kg (27.3 lb)	13.4 kg (29.5 lb)	NA	15.5 kg (34.1 lb)
	ATS1000-H	12.3 kg (27.1 lb)	NA	13.9 kg (30.5 lb)	15.4 kg (33.9 lb)
Construction		Black Anodized Aluminum Body with Hardcoated Tabletop			

Notes:
1. Excessive duty cycle may impact stage accuracy.
2. Payload specifications are for single axis systems and based on ball screw and bearing life of 2500 km (100 million inches) of travel.
3. Available with Aerotech controllers.
4. Specifications are for single-axis systems, measured 50 mm above the tabletop. Performance of multi-axis systems is payload and workpoint dependent. Consult factory for multi-axis or non-standard applications.

www.aerotech.com

Appendix L – Motor Code from Aerotech, Edited by P2C2

```
' ------------------------------------------------
' Validation Code - Traverse 400mm, Top to Bottom -
' ------------------------------------------------
' Author: Lisa Perez
'   Date: 11/18/2009
' Status: Functional
' ------------------------------------------------

PROGRAM

     DIM Distance AS INTEGER

     ' Set distance to travel.
     Distance = 400

     ' Acknowledge any faults.
     FAULTACK X

     ' Enable the axis.
     ENABLE X

     ' Home the axis.
     HOME X

     ' Set the curve factor.
     ' This value ranges anywhere from 0 to 100.
     ' Any value outside this range is a runtime error.
     ' If the scurve is zero, the ramp up and ramp
     ' down is linear. If the value is 100,
     ' the ramp up and ramp down is parabolic.
     ' Any value between these values is the percentage
     ' of the ramp up and ramp down that is parabolic.
     SCURVE  0

     ' Execute the linear move with the given distance
     ' and speed. This type of linear move ramps up to
     ' travel the specified distance and ramps down to zero.
     LINEAR X Distance F 20

     ' Disable axis.
     DISABLE X

END PROGRAM

' ------------------------------------------------
' Validation Code - Traverse 400mm, Bottom to Top -
```

```
' ------------------------------------------------
' Author: Lisa Perez
'   Date: 11/18/2009
' Status: Functional
' ------------------------------------------------

PROGRAM

     DIM NegDistance AS INTEGER

     ' Set distance to travel.
     NegDistance = -400

     ' Acknowledge any faults.
     FAULTACK X

     ' Enable the axis.
     ENABLE X

     ' Set the curve factor.
     ' This value ranges anywhere from 0 to 100.
     ' Any value outside this range is a runtime error.
     ' If the scurve is zero, the ramp up and ramp
     ' down is linear. If the value is 100,
     ' the ramp up and ramp down is parabolic.
     ' Any value between these values is the percentage
     ' of the ramp up and ramp down that is parabolic.
     SCURVE  50

     ' Execute the linear move with the given distance
     ' and speed. This type of linear move ramps up to
     ' travel the specified distance and ramps down to zero.
     LINEAR X NegDistance F 20

     ' Disable axis.
     DISABLE X

END PROGRAM

' ------------------------------------------------
' Validation Code - Traverse Var1, Top to Bottom --
' ------------------------------------------------
' Author: Lisa Perez
'   Date: 11/18/2009
' Status: Functional. Set Var1.
' ------------------------------------------------
```

```
PROGRAM

     DIM Distance AS INTEGER

     ' Set distance to travel.
     Distance = Var1

     ' Acknowledge any faults.
     FAULTACK X

     ' Enable the axis.
     ENABLE X

     ' Home the axis.
     HOME X

     ' Set the curve factor.
     ' This value ranges anywhere from 0 to 100.
     ' Any value outside this range is a runtime error.
     ' If the scurve is zero, the ramp up and ramp
     ' down is linear. If the value is 100,
     ' the ramp up and ramp down is parabolic.
     ' Any value between these values is the percentage
     ' of the ramp up and ramp down that is parabolic.
     SCURVE  0

     ' Execute the linear move with the given distance
     ' and speed. This type of linear move ramps up to
     ' travel the specified distance and ramps down to zero.
     LINEAR X Distance F 20

     ' Disable axis.
     DISABLE X

END PROGRAM

' ------------------------------------------------
' Operation Code ---------------------------------
' ------------------------------------------------
' Author: Lisa Perez
'   Date: 11/18/2009
' Status: Functional. Tweak speed as necessary.
' ------------------------------------------------

PROGRAM

     DIM Distance AS INTEGER
```

```
    ' Set distance to travel.
    Distance = 400

    ' Acknowledge any faults.
    FAULTACK X

    ' Enable the axis.
    ENABLE X

    ' Home the axis.
    HOME X

    ' Set the curve factor.
    ' This value ranges anywhere from 0 to 100.
    ' Any value outside this range is a runtime error.
    ' If the scurve is zero, the ramp up and ramp
    ' down is linear. If the value is 100,
    ' the ramp up and ramp down is parabolic.
    ' Any value between these values is the percentage
    ' of the ramp up and ramp down that is parabolic.
    SCURVE  0

    ' Execute the linear move with the given distance
    ' and speed. This type of linear move ramps up to
    ' travel the specified distance and ramps down to zero.
    LINEAR X Distance F 10

    ' Disable axis.
    DISABLE X

END PROGRAM
```

Appendix M – Test Procedure

Motor Testing

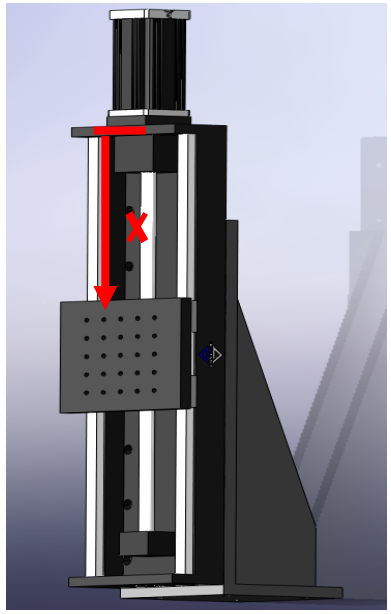
Set motor scale; ensure motor moves desired distance; test with and without support

Without the support connected to the stage:

Test No		Desired (in)	Desired (mm)	Computer Reading	Measured (mm)	Scale
1	Travel from uppermost end of range to lowermost end of range	16	406.4	260	417	1.603846
2	Travel from lowermost end of range to uppermost end of range	16	406.4	260	417	1.603846

After tests 3-5, assembly the support to the stage and then continue with tests 6-8

Test No		Desired (in)	Desired (mm)	Computer Input using scale	Measured (mm)	Error (mm)
3	Motor accurately moves platform to x from uppermost end of range	1	25.4	16	25	0.4
4		15	381	238	381	0
5		6	152.4	95	152	0.4
6	Motor accurately moves support to x from uppermost end of range	3.5	88.9	55	88	0.9
7		8	203.2	127	203	0.2
8		11.5	292.1	182	291	1.1



Motor Testing with Rotating Shaft

Ensure motor traverses while shaft is spinning; test motor after assembly into mill; spin shaft at desired speed and move support to given locations

Test No	Speed (RPM)	Locations (in from home)					Scaled Input (mm)					OK?	Comments
1	100	0	2	6	10	7	0	32	63	63	-48	1	Spinning all took place within sleeve
2	500	0	3	6	8		0	48	48	32		1	Spinning mostly took place within sleeve
3	850	0	3	6	9	12	0	48	48	48	48	1	Spinning all took place within sleeve
4	1000	16	15	14	13	12	253	-16	-16	-16	-16	1	Spinning all took place within sleeve
5	3000	16	15	14	13	12	253	-16	-16	-16	-16	1	Spinning all took place within sleeve
6	4000	16	15	14	13		253	-16	-16	-16		1	Spinning mostly took place within sleeve

Sensor Tests

Set object in known location; move object given distance; meaure displacement with sensors (note: all dimensions are in mm)

	Distance from Lens	Desired Displace-ment	Displace-ment Direction	Object Initial Position	Object Final Position	Initial Sensor Reading	Final Sensor Reading	Actual Displace-ment	Percent Error
	42	2	Toward	378	380	42.11	40.11	2	0
	40	5	Away	380	375	40.11	44.71	-4.6	-8
	44	4	Toward	376	380	43.88	40.02	3.86	-3.5
	44	4	Away	376	372	43.72	47.53	-3.81	-4.75
	48	6	Toward	372	378	47.53	42.03	5.5	-8.33
50mm lens	42	1	Away	378	377	42.03	43	-0.97	-3
	110	25	Toward	309	334	109.91	84.92	24.99	-0.04
	110	2	Away	309	307	109.79	111.78	-1.99	-0.5
	85	5	Toward	334	339	84.9	79.84	5.06	1.2
	85	10	Away	334	324	84.68	95.15	-10.47	4.7
	95	15	Toward	324	339	95.15	79.66	15.49	3.27
100mm lens	95	15	Away	324	309	95	109.65	-14.65	-2.33

Impact Tests

To test for natural frequency, move support to given location, hit end of shaft with rubber mallet, measure displacement

Test No.		Mass	Start position (in)	End position (in)	Motor input	Ok?	File Name
	1		0	0	0	1	aaa
	2		0	2	32	1	aab
	3		2	4	32	1	aac
	4		4	6	32	1	aad
	5		6	8	32	1	aae
	6		8	10	32	1	aaf
	7		10	12	32	1	aag
	8		12	14	32	1	aah
1	9	3 nuts	14	16	32	1	aai
	1		0	0	0	1	aba
	2		0	2	32	1	abb
	3		2	4	32	1	abc
	4		4	6	32	1	abd
	5		6	8	32	1	abe
	6		8	10	32	1	abf
	7		10	12	32	1	abg
	8		12	14	32	1	abh
2	9	big mass	14	16	32	1	abi

Static Support Tests

Spin shaft to desired speed, move motor to given location, take measurements

Test No.			Mass	Speed (RPM)	Start position (in)	End position (in)	Motor input	Ok?	File Name	Notes
		1			0	0	0			
		2			0	2	32			
		3			2	4	32			
		4			4	6	32			
		5			6	8	32			
		6			8	10	32			
		7			10	12	32			
		8			12	14	32			
	1	9	3 nuts	500	14	16	32			
		1			2	2	32			
		2			2	4	32			
		3			4	6	32			
		4			6	8	32			
		5			8	10	32			
		6			10	12	32			
		7			12	14	32			
	2	8	3 nuts	2000	14	16	32			
		1			8	8	126.6954			
		2			8	10	32			
		3			10	12	32			
		4			12	14	32			
	3	5	3 nuts	5000	14	16	32			
		1			12	12	190.0432			
		2			12	14	32			
1	4	3	3 nuts	10000	14	16	32			

Test No.			Mass	Speed (RPM)	Start position (in)	End position (in)	Motor input	Ok?	File Name	Notes
		1			0	0	0	1	bbaa	
		2			0	2	32	1	bbab	
		3			2	4	32	1	bbac	
		4			4	6	32	1	bbad	
		5			6	8	32	1	bbae	
		6			8	10	32	1	bbaf	
		7			10	12	32	1	bbag	
		8		501/501/5	12	14	32	1	bbah	
	1	9	big mass	00	14	16	32	1	bbai	
		1			2	2	32	1	bbba	out of range?
		2			2	4	32	1	bbbb	out of range?
		3			4	6	32	1	bbbc	
		4			6	8	32	1	bbbd	video started
		5			8	10	32	1	bbbe	loud noise
		6			10	12	32	1	bbbf	loud noise
		7			12	14	32	1	bbbg	loud noise
	2	8	big mass	1816/1815	14	16	32	1	bbbh	noise stopped - video stopped
		1			8	8	126.6954	1	bbca	jumped natural frequency
		2			8	10	32		bbcb	FAILURE
		3			10	12	32		bbcc	
		4			12	14	32		bbcd	
	3	5	big mass	4986	14	16	32		bbce	
		1			12	12	190.0432		bbda	
		2			12	14	32		bbdb	
2	4	3	big mass	10000	14	16	32		bbdc	

Dynamic Support Tests

While shaft is spinning at given speed, move support from start position to end position at given speed, measure displacement using sensors

Test No.			Mass	Speed (RPM)	Start position (in)	End position (in)	Motor input (motor units)	Motor speed (in/s)	Motor speed (motor units/s)	Ok?	Notes
		1			0	16	253	1	21		
	1	2	3 nuts	500	0	16	253	3	42		
		1			2	16	222	1	21		
	2	2	3 nuts	2000	2	16	222	3	42		
		1			8	16	127	1	21		
	3	2	3 nuts	5000	8	16	127	3	42		
		1			12	16	63	1	21		
1	4	2	3 nuts	10000	12	16	63	3	42		
		1			0	16	253	1	21		
	1	2	big mass	500	0	16	253	3	42		
		1			2	16	222	1	21		
	2	2	big mass	2000	2	16	222	3	42		
		1			8	16	127	1	21		
	3	2	big mass	5000	8	16	127	3	42		
		1			12	16	63	1	21		
2	4	2	big mass	10000	12	16	63	3	42		

Dynamic Support Tests (above natural frequency)
While shaft is spinning at given speed, move support from start position to end position at given speed, measure displacement using sensors

Test No.			Mass	Speed (RPM)	Start position (in)	End position (in)	Motor input (motor units)	Motor speed (in/s)	Motor speed (motor units/s)	Ok?	Notes
		1			16	0	0	1	21		
	1	2	3 nuts	500	16	0	0	3	42		
		1			16	0	0	1	21		
	2	2	3 nuts	2000	16	0	0	3	42		
		1			16	0	0	1	21		
	3	2	3 nuts	5000	16	0	0	3	42		
		1			16	0	0	1	21		
1	4	2	3 nuts	10000	16	0	0	3	42		
		1			16	0	0	1	21		
	1	2	big mass	500	16	0	0	3	42		
		1			16	0	0	1	21		
	2	2	big mass	2000	16	0	0	3	42		
		1			16	0	0	1	21		
	3	2	big mass	5000	16	0	0	3	42		
		1			16	0	0	1	21		
2	4	2	big mass	10000	16	0	0	3	42		

Appendix N – Pictures of Mechanism

Figure N1: Mechanism set up inside mill

Figure N2: Layout of work area – Testing (Jessica), Sensor Operation (Lisa), Mill Operation (Matt), and Motor Operation (Alex)

Figure N3: Shaft with no offset mass

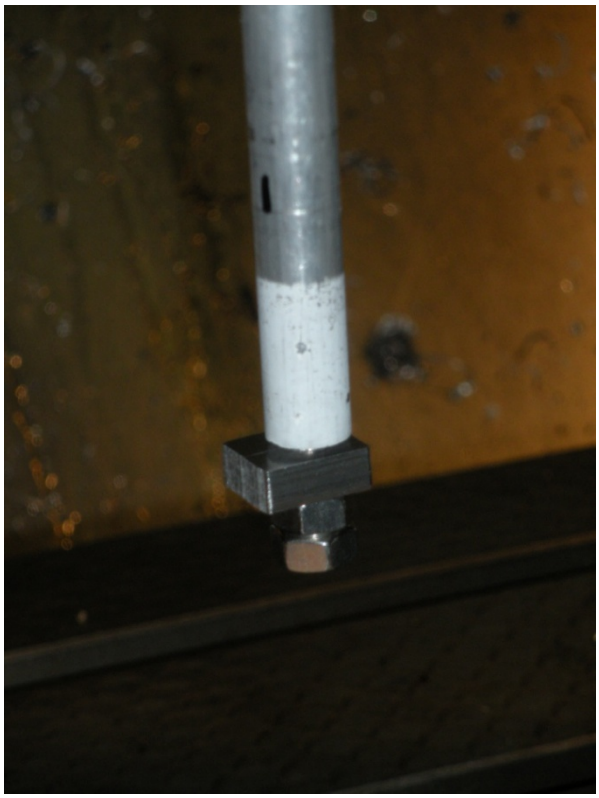


Figure N4: Shaft with middle sized offset mass

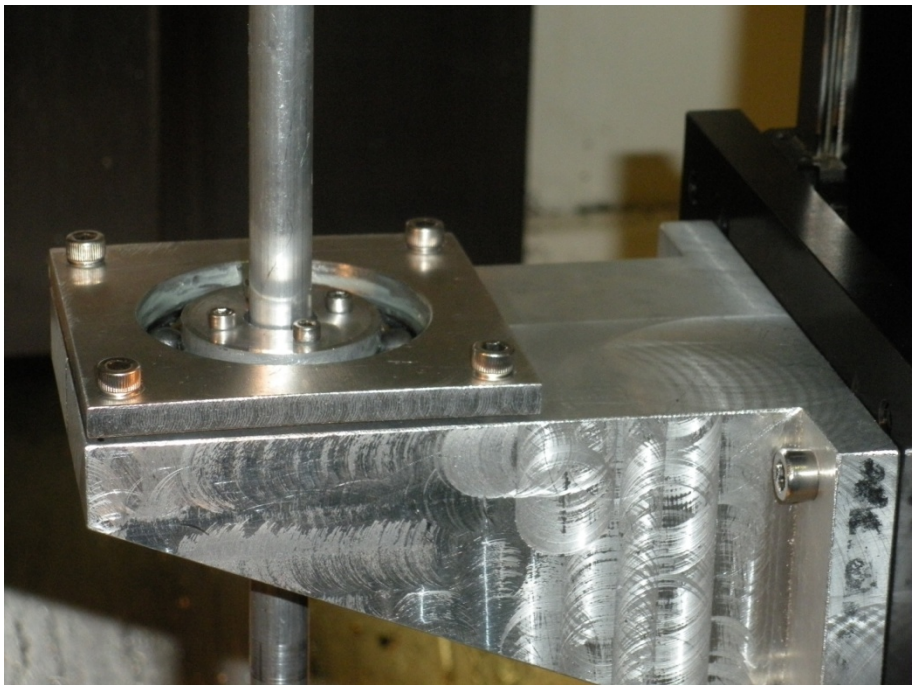


Figure N5: Support assembly with shaft

Appendix O – Pictures of Failure

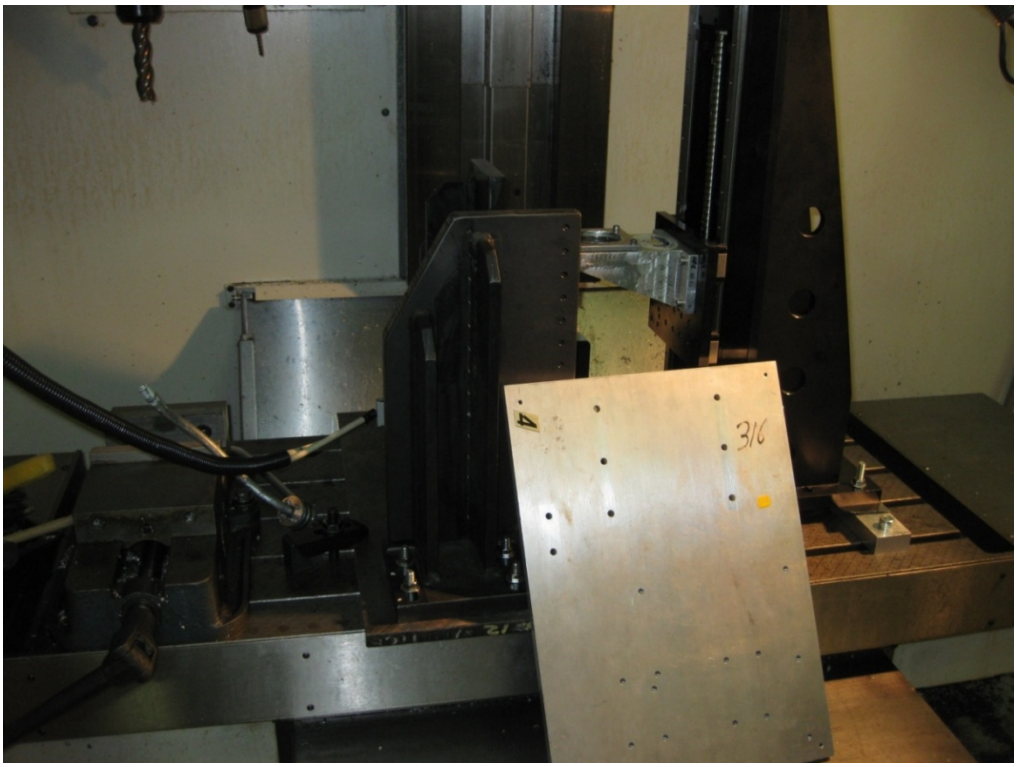


Figure O1: Assembly after failure

Figure O2: Location of shaft after failure

Figure O3: Deformed end of shaft

Figure O4: Threaded end of shaft with offset mass broke off the shaft

Figure O5: Deformed shaft with inner race of ball bearing, two bearing adaptors, and sleeve bearing

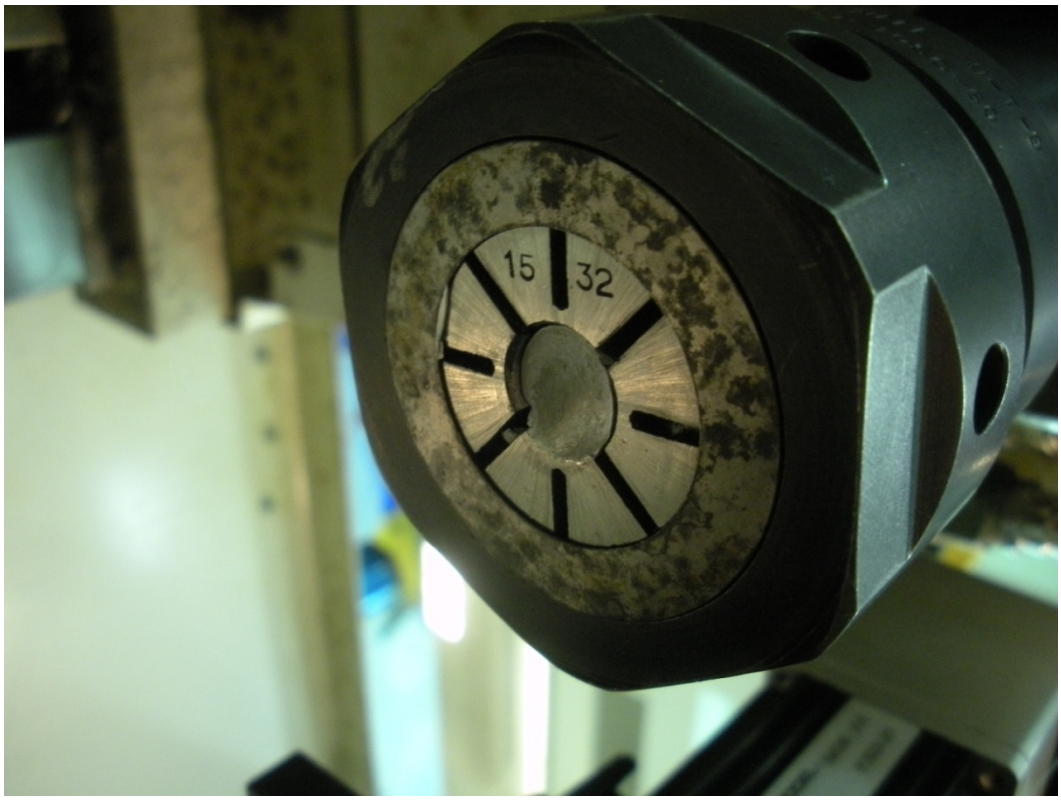


Figure O6: Fixed end of shaft remains in the mill collet.

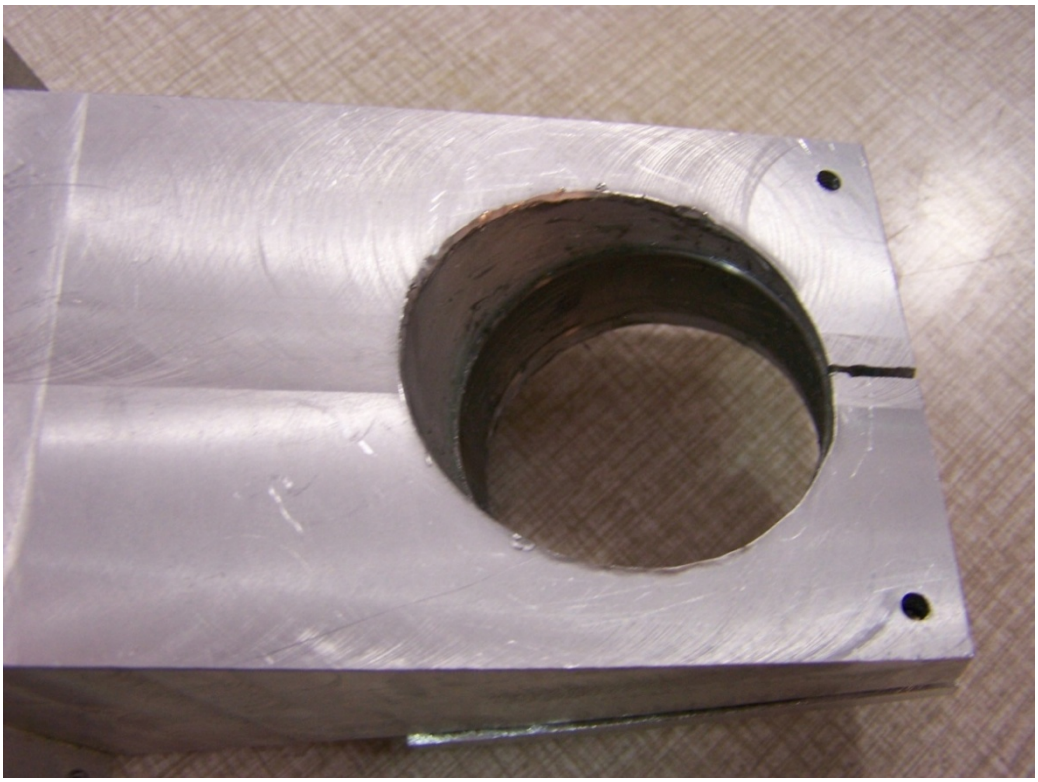


Figure O7: Shaft impacted aluminum support causing deformation of the support. Outer race of ball bearing remains in the support.

Figure O8: All but four of the ball bearings were recovered.

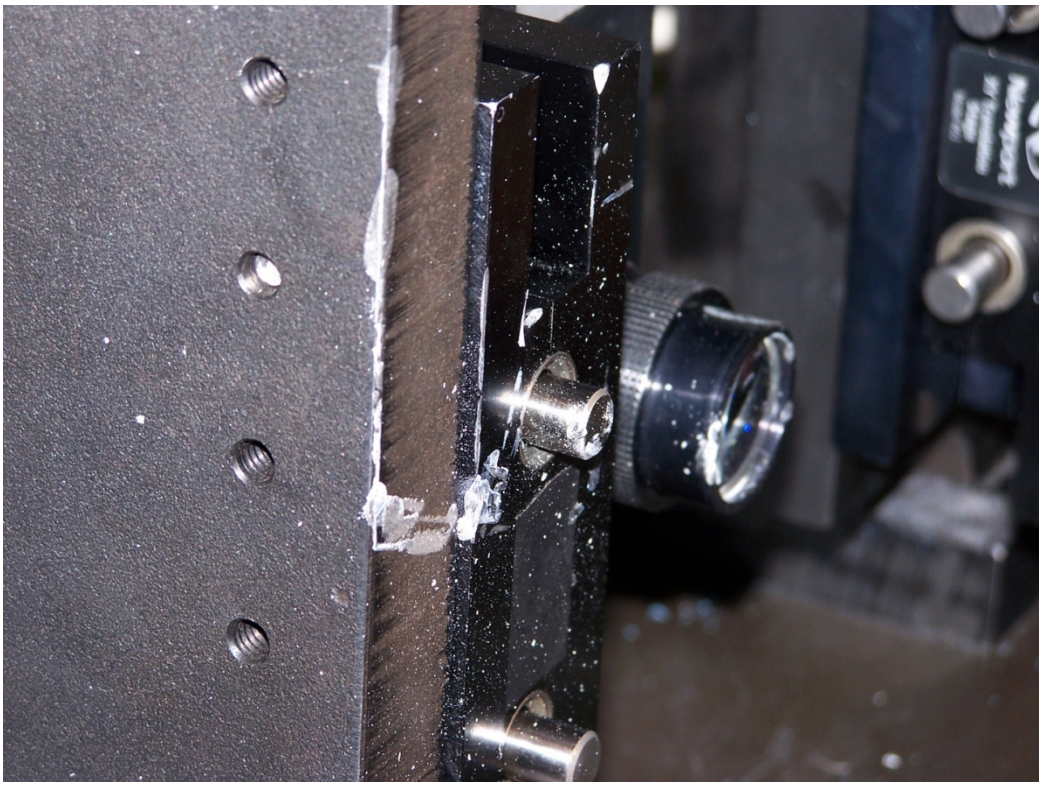


Figure O9: End of shaft deformed steel sensor mounts

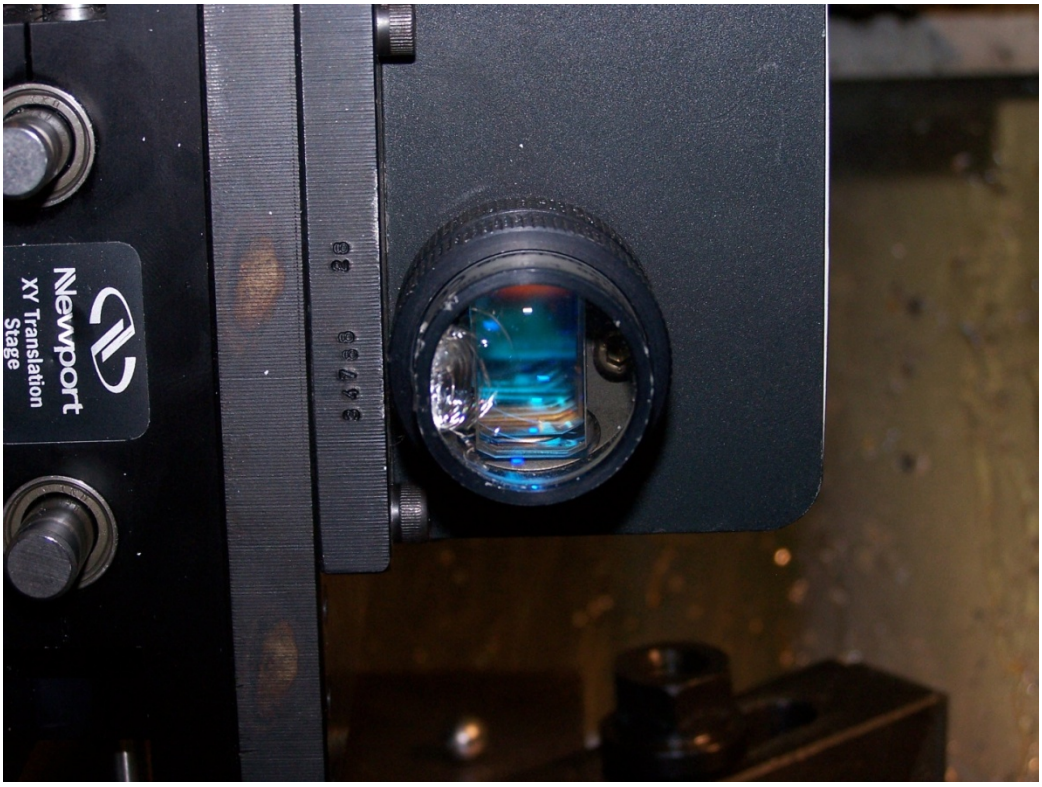


Figure O10: End of shaft cracked sensor lenses

References

[1] Barber, James. (2001). *Intermediate Mechanics of Materials*. McGraw-Hill Publishing Co., p. 514

[2] Watanabe; Yukio (Yokohama, JP), Shimizu; Fukumi (Yokohama, JP), Naruse; Katsuhiko (Yokohama, JP). "Method and apparatus for measuring vibrations of rotating shaft." US Patent 5,293,040. March 8, 1994.

[3] Harris, Cyril M. and Allan G. Piersol. (2002). *Harris' Shock and Vibration Handbook*. 5th Edition. McGraw-Hill Publishing Co. pp. 7.1-7.48.

[4] *NSB Bearings.* Retrieved from http://www.nsbbearings.com/images/prods/SELF%20 ALIGNING%20BALL%BEARINGS.gif.

[5] *Bombay Harbor.* Retrieved from http://www.bombayharbor.com/productimage/ 0054673001243486625/Spherical_Roller_Bearing.jpg.

[6] *Germes Online.* Retrieved from http://www.germes-online.com/direct/dbimage/ 50277064/Deep_Groove_Ball_Bearing.jpg.

[7] *Quality Bearings and Components.* Retrieved from http://www.qbcbearings.com/ BuyRFQ/SleeveBNonMetalPTI.htm.

[8] *Micro Epsilon - TechNote: Precise non-contact displacement sensors*. Retrieved from http://www.micro-epsilon.com/download/products/T001--en--precise-non-contact-sensors.pdf.

[9] *Mirco Epsilon - Measurement Product Guide 2009*. Retrieved from http://www.micro-epsilon.com/download/products/cat--Micro-Epsilon--products--en.pdf. p. 23.

[10] *eFunda – LVDT Introduction.* Retrieved from http://ww.efunda.com/DesignStandards/ sensordslvdt.lvdt_intro.cfm.

[11] *AST Macro Sensors.* Retrieved from http://www.macrosensors.com/images/lvdt diagram.jpg.

[12] Doherty, M. (2009, October and November). Sales representative, USA and Canada – Optimet. Telephone conferencing.

[13] 3D Non-Contact Measurement by Optimet – Conoprobe Specifications. Retrieved from http:// optimet.com/products/conoprobe.htm.

[14] ATS1000 Linear Stages Specifcations. Retrieved from http://www.aerotech.com/products/stages/ ats1000specs .html.

[15] Burg, R. (2009, October and November). Sales representative, Aerotech Michigan and Aerotech Lower Midwest. Motor and linear stage instructional and telephone conferencing.

[16] Jones, F.D., Ryffel, H.H., Oberg, E., McCauley, C.J., & Heald, R.M. (2004). *Machinery's Handbook*, 27th edition. United States: Industrial Press, Inc..

[17] *SKF Group.* Retrieved from http://www.skf.com/skf/productcatalogue /Forwarderzz?action=PPP&lang=en&imperial=false&windowName=null&perfid=114001 &prodid=1140012206.

[18] Grant Design Limited. (2009). CES Selector Version 5.1.0 [Computer Software]. University of Michigan, Ann Arbor, MI

Biographies

Matthew Carpenter

Matt is originally from Brighton, Michigan, just 20 miles north of Ann Arbor. Both of his parents attended General Motors Institute (now Kettering University) and obtained degrees in mechanical and electrical engineering. Their influence has been the main driver for Matt to become an engineer. From the time he was very young, he would always help his father with repairs and projects around the house. Matt learned from first-hand experience with his father what it means to think analytically like an engineer. By the time he was in middle school, Matt had realized that he truly enjoyed engineering and decided to start planning on that career path, as his NFL prospects were not looking as promising. When it became time to select a college, it was an easy decision to attend the University of Michigan (U-M).

Since beginning his studies at U-M, Matt decided to pursue mechanical engineering because he finds it much more enjoyable to work on projects he can visualize and touch. To divert his mind from the endless hours of work and studying, Matt has started playing intramural sports. Over the past 3 years, he's played 9 different IM sports, including soccer, flag football, and broomball. After finishing his undergraduate degree, Matt will pursue a MSE in mechanical engineering at U-M, with a specialization in design and manufacturing. Beyond that, his future career plans are less certain. Matt plans on obtaining a job in industry, but does not yet know where.

Alexander Cicerone

Alex originally came from just north of Detroit, Michigan, but moved to Grand Rapids, Michigan when he was young. With quite a bit of inspiration from his high school physics teacher and an affinity to math and mechanical physics, Alex decided to go into mechanical engineering.

After he obtains his Bachelor's degree from U-M, Alex is going to pursue a Master's degree in industrial and operations engineering. Additionally, Alex thoroughly enjoys subjects that involve dynamics and controls, and looks to concentrate in these fields once he is in the workforce.

In his spare time, Alex likes to play tennis, golf, hockey, and video games such as NHL 2009, Call of Duty 4, and Mass Effect.

Lisa Perez

Lisa Perez is a 5th year student in mechanical engineering and aerospace engineering at U-M. She comes from Sterling Heights, Michigan, just one hour away from the University. Having been inspired by the FIRST Robotics Competition as a high school student, Lisa thoroughly enjoys the field of robotics. She currently teaches a middle school robotics team (the M Go Blue Bots) and a high school robotics team (the Rat Pack) in Ann Arbor, and is taking the mechatronics course at U-M.

After college, Lisa plans to attend graduate school to obtain her Master's degree specializing in the design and control of robots for medical and/or space solutions.

Lisa was a member of the Michigan Marching Band flag line, the Solar Car Team aerodynamics division, and the St. Mary Student Parish Children's Liturgy teaching team. Currently, she is involved as the vice president of the Pi Tau Sigma Mechanical Engineering Honor Society and planning committee chair for the Ann Arbor FIRST District Competition. She volunteers as a referee and an inspector for FIRST events. Lisa also plays tennis and Ultimate Frisbee, runs, and watches large amounts of Wolverine and Red Wing hockey.

Jessica Port

Jessica Port hails from Foster City, California, a small town just 20 miles south of San Francisco. She began her college education at Purdue University in West Lafayette, Indiana, but transferred to U-M shortly thereafter where she is currently in her fifth year studying mechanical engineering and mathematics. She particularly enjoys the fields of dynamics, specifically mechanical vibrations and acoustics, and, after higher education, would like to work in the aerospace industry.

While in school, Jessica has participated in both cooperative education and internship programs. Her four rotations with Hamilton Sundstrand included work in product design and development, manufacturing, and structural dynamic testing. Her summer experience at The Boeing Company involved work in heating and air conditioning and fire protection systems.

After graduating with her Bachelor's degree in May, Jessica will begin working toward her Master's degree and perhaps a PhD.

Jessica currently serves as the Secretary of Affairs for the Pi Tau Sigma Mechanical Engineering Honor Society. In November, she will take over as the Secretary of the Society of Women

Engineers' (SWE) design team, Boeing Team Tech (BTT). Last year, she was selected as a presenter for the annual BTT project, a user-independent wheelchair locking mechanism for mass transit buses, at the National SWE Convention in Baltimore, Maryland. She is also involved in the Tau Beta Pi Engineering Honor Society. In her free time, Jessica tutors elementary aged children in reading and writing through a local service group. She also enjoys reading, swimming, biking, and watching Wolverine football.