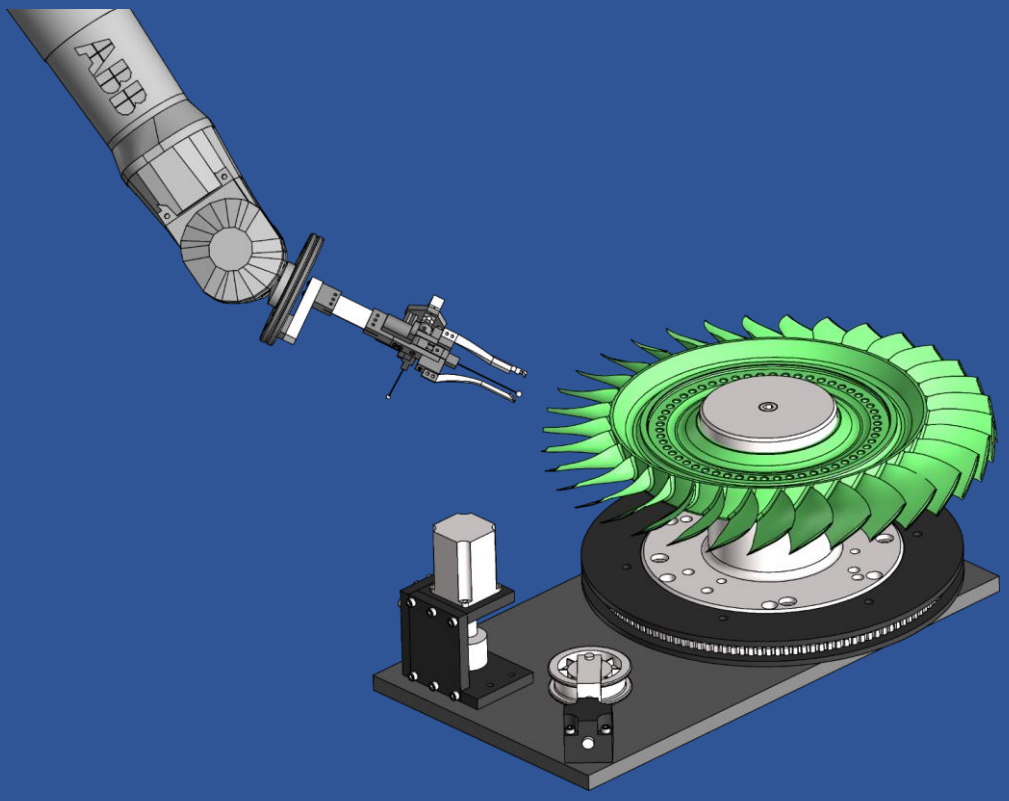
GE Aviation

WPI

Blisk Inspection System



Breanne Happell
Karen Orton
Kevin Ouellette
Charles Sinkler

Advised By:
Michael Gennert
Craig Putnam
Kenneth Stafford

Blisk Inspection System

A Major Qualifying Project Report

Submitted to the Faculty of Worcester Polytechnic Institute

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

In cooperation with

GE Aviation in Hooksett, New Hampshire

Submitted on April 24th, 2017

Submitted By:

Breanne Happell

Karen Orton

Kevin Ouellette

Charles Sinkler

Project Advisors:

Michael Gennert

Craig Putnam

Kenneth Stafford

WPI

GE Aviation

CONTENTS

List of Figures

List of Tables

Robotic Blisk Fillet Inspection

Breanne Happell, Karen Orton, Kevin Ouellette, Charles Sinkler
Advised By: Michael Gennert, Craig Putnam, Kenneth Stafford

***Abstract*—The Robotic Blisk Inspection System utilizes a custom turntable, an ABB robotic arm with custom end of arm tooling (EOAT), and a computer vision algorithm to autonomously determine if the blade root fillets are within tolerance. Currently, GE Aviation in Hooksett, NH uses manual techniques to inspect the bladed disks (blisks) that they produce. Their techniques require consumables and are more costly than a robotic inspection. Positional feedback through force and contact sensing allows the system to navigate the closely spaced blades with the compact EOAT. Compliance in one axis ensures the EOAT is properly seated within the fillet. A computer vision algorithm developed with OpenCV tracks the fillet position to confirm proper radius dimensions.**

Fig. 1: A Single-Stage Blisk

I. Introduction

The GE Aviation plant in Hooksett, New Hampshire manufactures bladed disks (blisks) used to compress intake air in jet engines. Fig. 1 shows a single stage blisk. Blisks vary greatly; the Hooksett Plant manufactures blisks with diameters ranging from 6" to 36". Some blisks have multiple stages, or layers of blades, where the gap between stages varies from 0.7" to 2". Blisks are milled from a single slice of cylindrical titanium.

The LEAP series of GE blisks consists of two single stage blisks, and one double-stage blisk. The part number for these three blisks are 2468M19P01 (P01), 2468M17P02(P02), and 2468M18G01(G01). GE Aviation is increasing its production of the LEAP series, which is projected to compose 60-80% of the Hooksett plant's production volume by 2021.

Blisks, as a critical engine component, are carefully inspected by hand before they leave the factory to ensure they meet quality standards. The blade fillets between the blades and the hub are one of many inspected components. The fillets are inspected to ensure their radii are within a given tolerance. Blade root fillet dimensions are critical to the strength of each blade. The Hooksett Plant wants to automate the inspection of these fillets.

The purpose of this project was to create a robotic system capable of autonomously inspecting the blade root fillets on LEAP series blisks for GE Aviation. GE Aviation aimed to reduce the time and money spent inspecting each blisk during the Quality Assurance (QA) checks. A robotic solution would decrease the need for salaried QA workers, thereby reducing the direct labor costs associated with each blisk. As stated previously, much of the Hooksett plant's business in the coming years will be LEAP components, so the scope of the project was limited to those blisks, although this process could be expanded to more blisks in the future.

II. Background

A. Manual Inspection Methods

GE Aviation's primary method for fillet inspection starts with a fine, white developer powder sprayed onto the blisk. Precision ball gauges are then run along the fillets by an inspector, scraping off the powder where contact is made. A ball gauge that matches the maximum tolerance for fillet radius should make two points of contact with the blisk, producing two tracks in the developer powder along the fillet. A ball gauge that matches the low tolerance will make one point of contact and thus one track in the powder. These are illustrated in Fig. 2. If both conditions are met, the radius is within both tolerance values. This process takes a skilled technician between 15 and 30 minutes for a single stage blisk.

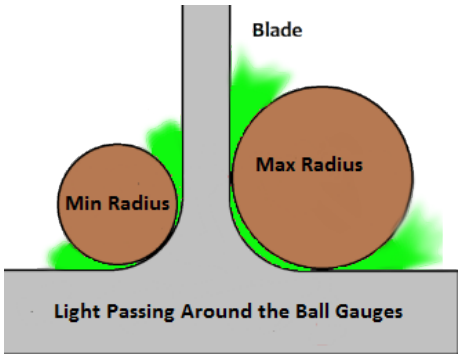


Fig. 2: Contact points of the max and min ball gauges in a fillet.

An accepted secondary inspection method uses a light

behind the same ball gauges to validate the radius of the fillets. The smaller ball produces one region of light on each side of the ball, while the larger ball produces one region in the middle and one on each side. This process is currently used if the technician is unsure of the results from the developer test as the pattern of light passing around the ball is easy to see but generally takes longer than the first method.

The current Takt time for a GE Aviation blisk is roughly three hours. This means each unit must move from one process to the next in three hours to meet production demand. The fillet inspection is one of many tests performed by the Quality Assurance (QA) team. GE Aviaton requires that the inspection process remain under an hour per blisk stage to meet Takt time goals.

B. Previous Project Results

A similar project was attempted in AY2015, but there were many aspects of the previous designs that needed to be improved upon. The system for inspecting blisk fillets used a turntable to hold the blisk and rotate it. An ABB robotic arm then inspected each fillet using a custom End of Arm Tooling (EOAT). Rather than use the standard process of developer powder, the previous team used the light inspection method to determine the radius of the fillet. A camera, ball gauge, and light source were all mounted on the EOAT. The LED illuminated the ball gauge such that the camera could capture the shape of the light passing by the ball gauge. Based on the shape of this light and the size of the ball gauge, the system was designed to determine whether or not the fillet was within tolerance.

The previous team designed and built the EOAT and turntable. They also implemented computer vision to assess each fillet. However, the previous team did not successfully combine each of these components into one functioning system. In addition to this, each component had issues that needed solutions. This section aims to identify these problems.

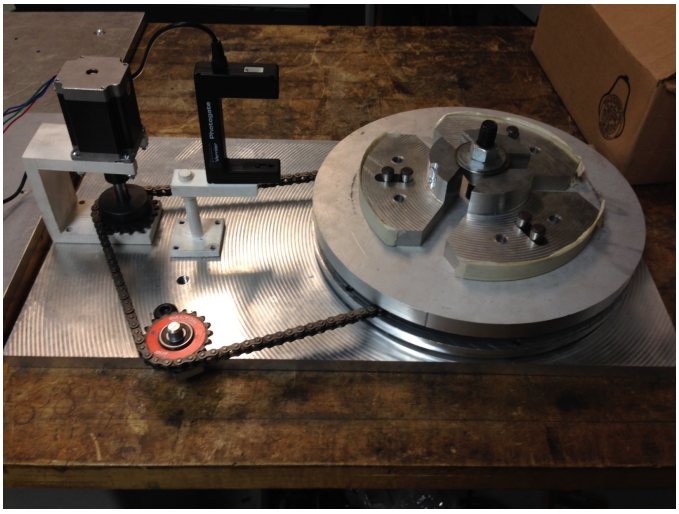


Fig. 3: Previous Turntable

1) Turntable: The previous project team built a turntable to hold and locate the blisk. The goal was to position each new blade of the blisk in the same position as the last, so the robotic arm could make the same motion for each blade. The completed turntable can be seen in Fig. 3.

A three jaw chuck clamped the blisk to the turntable. A stepper motor was used because it can turn small amounts extremely precisely. A photogate sensor confirmed the location of the next blade once the motor had positioned it.

There were a few manufacturing issues with the implementation of the turntable. The first was with the chuck jaws, which held the blisk in place. They were slightly too short and therefore tape was needed to properly clamp the blisk. There was also an issue with the chain rubbing, causing the motor to skip and the precision of the table to be compromised. Finally, the sprocket was eccentric to the assembly, causing the chain to tighten and loosen as the blisk spun. This also caused a significant issue with the precision of the table. Most of the problems with the previous turntable were manufacturing issues.

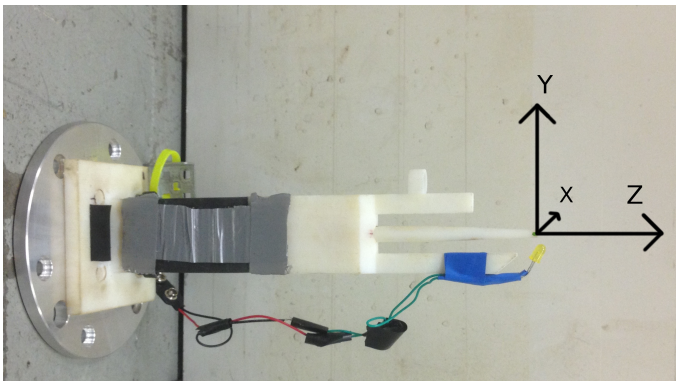


Fig. 4: Previous EOAT

2) End of Arm Tooling (EOAT): The final design of the EOAT, shown in Fig. 4, consisted of three main sections: a hard plastic base, a segment of flexible elastomer, and a hard plastic tip.

The hard plastic base was a platform used to connect the EOAT to the ABB arm. Connected to this was a flexible elastomer measuring 0.75” by 1.5” by 2”. This elastomer could flex a maximum of 2” in the X direction, 0.75” in the Y direction, and 0.25” in the Z direction of the tool’s coordinate frame, as shown in Fig. 4. This flexibility compensated for any error in the positioning of the turntable or the arm by allowing the EOAT to bend into the correct location. This elastomer was successfully created, however in practice it sagged 10° due to the weight of the EOAT, causing issues when pathing with the ABB arm.

The third section of the EOAT, the hard plastic tip, held the ball gauge, camera, and LED. The ball gauge was mounted on a thin rod that extended from the center of the EOAT. Two rods extended above and below the central rod. The longer of these held the LED, which was positioned as close to the ball as possible. The shorter arm held the camera. Because the

camera was too big to fit close to the ball gauge, it had to be mounted parallel to the central rod. A small mirror allowed the camera to inspect the ball gauge at a 90° angle to the hub of the blisk. This section of the arm was successfully created, and images of the back-lit ball gauge were captured. However, because the mirror was mounted approximately an inch away from the camera, only a small portion of each image was used in the image processing, reducing the effective resolution of the camera. Though the LED and mirror required precise positioning, they were held in place by a paper clip. Any slight jarring of the tooling negatively impacted the camera's view of the ball.

3) Computer Vision: The system also used computer vision to determine if the current fillet passed inspection.

VLC Media Player was used to convert the captured video into images for MATLAB processing. The image was first cropped to show only the area of interest. The image was then aligned based on a dot of lime green paint on the ball gauge. Lime green was used because a computer can easily differentiate this color due to its high green amount in the RGB color scale. The image was then converted to grayscale and then to binary, rendering the light and dark areas of the image as a binary matrix.

The captured image was compared, using a 2D correlation program, against images of fillets that passed inspection. If the matching percentage passed a certain threshold, then the image passed inspection. If an image failed inspection, the location was reported to the operator for further inspection.

The cropping and zooming significantly reduced the quality of the images. The MATLAB image processing required a separate laptop with the MATLAB Image Processing Toolbox installed. This increased the complexity and cost of the system. The computer vision algorithm worked by comparing the current image to previously captured passing images. This approach was not a very robust way to verify that light is being detected in each of the segments, as a slight camera repositioning or light adjustment could cause the image to fail.

III. Methodology

A. Procedure

This project was completed over the course of an academic year. The team began by analyzing the similar project that was attempted in 2015. The team examined both the successful and unsuccessful aspects of the project to guide the initial system brainstorming. This analysis is discussed in detail in Section II.

The team then moved into an initial design and analysis phase. The team prototyped critical components to ensure they would function as intended. Based on these prototypes, the team redesigned the system's components. The design process was repeated until a final design was proven sufficient. Once this happened, the final design was manufactured and all the components were tested together. Where necessary, redesigns and other adjustments were made until the system operated optimally.

B. Physical System Overview

The blisk inspection system consists of three physical parts. An ABB IRB1600T six degree of freedom robot arm and corresponding IRC5 controller was used as the basis of the inspection system. To simplify the robot's pathing, a turntable rotates each blade into a known position and holds the blisk steady while it is inspected. An EOAT allows the ABB robot to interact with the blisks. The EOAT holds and illuminates the correct ball gauge in the fillet while capturing pictures to be analyzed.

As with any physical system, positional error had to be considered in the design. This error was expected to include mainly eccentricity in parts assumed to be concentric. As such, the EOAT was designed to have force feedback, reducing the chances of damage from driving into the blisk. The EOAT also has a compliant segment made of spring steel which can flex 1/8" under 1/8 pound of force, allowing the ball gauge to slot properly into the fillet, even if there is some positional error in the turntable.

C. Control System Overview

The control system handles the user interface, as well as the sensors and actuators necessary for performing an inspection. It also communicates with the IRC5 robot controller in order to coordinate robotic arm movements with image collection. When the images are collected, the control system runs computer vision software to determine whether or not a fillet is within its manufacturing tolerance.

Fig. 5: Raspberry Pi 3 Model B

1) Processor: The team chose to use a Raspberry Pi 3B as the main control system. A Raspberry Pi, shown in Fig. 5, is a microcomputer that has pins for controlling digital input and output (IO). The Raspberry Pi was chosen because it can handle both the computer vision for the fillet inspection and the control of other components through digital IO. It also supports Linux, allowing for GUI control of the system. Using a laptop or microcontroller to implement the project software

was also considered. The problem with using a laptop was that it would not be able to handle the digital IO easily. Using a microcontroller was also considered, but it would not be able to run any of the image processing toolboxes necessary for the computer vision inspection. In order to control the IO, image processing, and the robotic arm, a laptop would have to communicate with both a separate microcontroller and the IRC5. In contrast, the Raspberry Pi could be used for all three functions, and was much cheaper and simpler than using both a computer and microcontroller. Though a laptop would likely have more processing power than the Raspberry Pi, the Pi 3B has a 1.2GHz 64-bit quad-core ARMv8 CPU and 1GB of RAM, which was sufficient for this project.

D. Turntable

The turntable is the base of the design, holding the blisk and rotating each blade into place. This component allows the ABB robot to follow the same path when inspecting the same fillet with the same ball gauge on each blisk, simplifying the robot pathing. The turntable is driven by a stepper motor through a timing belt tensioned with a custom spring tensioner to reduce backlash in the system. A stepper motor is used to drive the pulley system, allowing for precision position control. Finally, a custom part provided by GE Aviation is used to interface between the turntable and each blisk. A picture of the final turntable can be seen in Fig. 6.

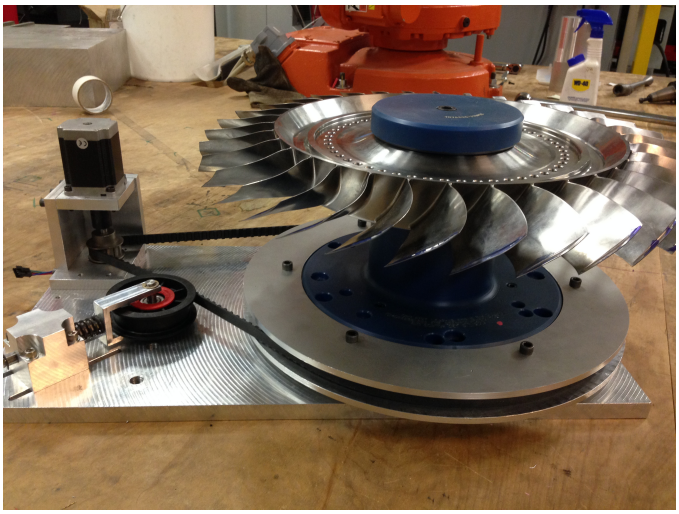


Fig. 6: Turntable

1) Blisk Holding Mechanism: The blisk holding mechanism was designed to hold the blisk rigidly on the turntable. It had to be capable of holding every blisk in the LEAP series. Each blisk has a different inner hub diameter, so the holding mechanism needed to accommodate those variations. The holding mechanism was also designed to hold the blisk at least 6” off the turntable surface to leave room for the EOAT to reach under the blades during the inspection process.

GE Aviation has custom plastic holding devices (adaptors) for each blisk that all have the same mounting holes. As this part was already in use by GE Aviation, it was determined that the system would be easier for GE to recreate if this part was used. The turntable was therefore designed such that the correct adaptor could be bolted on as desired. An example of these adaptors can be seen in Fig. 7.

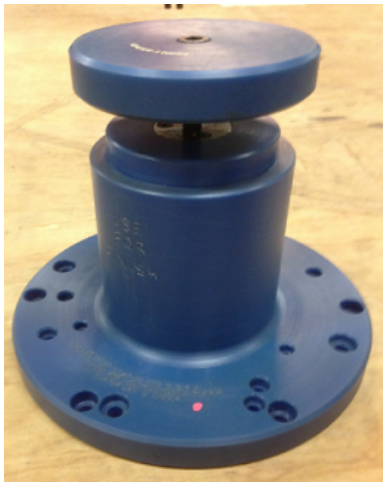


Fig. 7: Blisk Holder

2) Drive System: Three designs were considered for the turntable drive system. The first was a chain drive system, the same design used in the previous project. This system would involve two chain sprockets, one connected to the motor shaft and one connected to the turning disk. This chain system would require adjusting and maintaining the tension on the chain. The second option was a timing belt drive system. This would have a similar design to the chain drive except the chain would be replaced with a timing belt. A direct drive system was also considered, with the motor directly connected to the turning disk.

To compare and decide on the best system, a decision matrix was created with seven criteria that were important to a successful design. From most important to least important those criteria were: accuracy, reliability, speed, cleanliness, ease of manufacturing, and cost of manufacturing. Accuracy, reliability, and speed were most important because they affected the success or failure of our project. Cleanliness was important because it makes the system more desirable to use for GE Aviation. Ease and cost of manufacturing were lowest because, while they make building the system easier, they do not play a key role in the result of the project.

Scores were selected between one and ten for each category with higher scores representing a more desirable score. The direct drive scored lower than the other two as it has no gearing to make it faster. It would be easier and cheaper to manufacture as there are fewer components involved. While the timing belt driven system is more difficult to manufacture, as a custom pulley would need to be built, it would be significantly cleaner than a greasy chain drive. It is also more reliable and accurate as slop and backlash can be removed almost entirely from a timing belt system.

Each factor was weighted based on its importance to the final design. The scores were then multiplied by their weighting factor and added together to generate a total score for each design. Based on the decision matrix, shown in Table I, the timing belt drive was the most well-suited to the needs of the system.

	Weighting Factor	Chain Drive	Timing Belt	Direct Drive
Accuracy Capability	6	9	10	4
Ease of Manufacturing	2	6	4	9
Cost of Manufacturing	1	6	7	10
Cleanliness	3	6	9	10
Speed	4	10	10	6
Reliability	5	8	9	4
Total		170	187	126

TABLE I: Drive System Decision Matrix

Once a timing belt was selected for the drive mechanism, calculations were completed to design the full mechanism. The larger pulley was designed with a diameter of 15" such that it was slightly larger than the plastic blisk holder. The size of the motor pulley was to be determined and affects both the maximum tension on the timing belt and the turning speed.

The worst case moment of inertia was determined using a SolidWorks model. The acrylic pulleys, two aluminum disks above and below the pulleys to keep the belt in place, the blisk holder, and the heaviest blisk were all modeled, resulting in a worst case moment of inertia of 1200 in^2-lb.

A stepper motor was used because it gives the turntable highly accurate turning capabilities without requiring feedback. The motor torque of the stepper motor used is 16 in-lb. It was decided that the repositioning of the blades by the turntable needed to operate at a speed of under five seconds per blade. This will allow enough time for the robot to move between each blade and path along the fillet.

To calculate how quickly the turntable would turn, the moment of inertia of everything being turned, the motor torque, and the gear ratio were needed. The radius of the motor pulley, the motor torque, and the maximum acceleration all determine how much tension the motor puts on the belt. By selecting potential motor pulley radii and checking the resulting belt tension and turning speed, an appropriate motor pulley was selected. The result was a 12 tooth, 1.5" diameter pulley. This produces a maximum belt tension of 21.33 lb, and a worst case of 1.3 second turn time between blades. Given this maximum belt tension, a 0.5" wide L type belt was purchased because it has a working tension of 24.5 lb, which is just above the maximum tension.

As previously stated, a stepper motor was selected to drive the turntable so that the position could be very precisely controlled. The specific stepper motor selected was the Pololu NEMA 23, 3.2V, 2.8A/phase stepper motor. The control system for this stepper motor consists of the Pololu A4988 Stepper Motor Driver, Black Edition. The schematic for the stepper drive system is shown in Fig. 8. The stepper motor driver is configured to microstep the motor with 1/16th of a step, allowing the highest precision possible with this driver. This

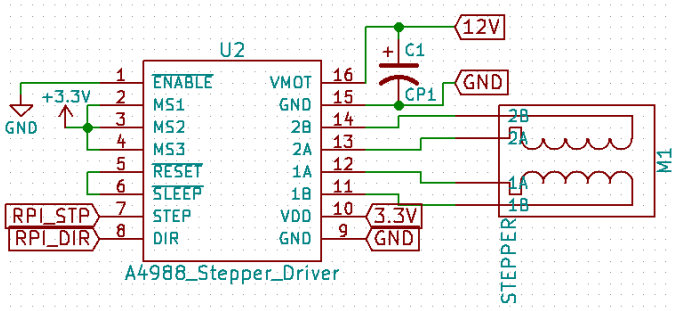


Fig. 8: Stepper Motor Drive Circuitry

gives the overall turntable a theoretical rotational accuracy of ± 0.00060 in at the blisk hub. This is shown in Eq. 1. All calculations are for blisk model 2552M02P02.

$$\begin{aligned} \theta_{step} &= \frac{1.8^\circ}{16} \times \frac{12}{125} = 0.0108^\circ \\ \theta_{blade} &= N \times \frac{360}{Blades} = N \times 10.58^\circ \\ Error_{Angular} &= MAX(MOD(\theta_{blade}, \theta_{step})) \\ Error_{Linear} &= \frac{Error_{Angular}}{360^\circ} \times D_{blisk} = 0.00060'' \end{aligned} \tag{1}$$

The software to control the stepper motor sends signals to the step pin on the motor controller in order to turn the motor. The exact number of steps between the blades on each stage of a blisk was calculated to minimize error in the turning of the blisk. This functionality is handled in a function to increment the blisk by one blade based on the current blade number and which stage is being inspected.

The stepper motor driver is a small surface mount device. It also puts through a large amount of power. A thermal study was completed in SolidWorks of the chip and heatsink to determine if any further cooling mechanisms were needed to keep the chip from entering a thermal shutdown mode. First, a study was conducted with a low convection coefficient, 1.5 W/m^2 K, and 3.37 W going into heat power. This power usage was calculated in Eq. 2. After three hours, the chip reached a maximum of 2575 K, much hotter than the 423 K chip shut off. The results of this study can be seen in Fig. 9. When a fan was added to the study, changing the convection coefficient to 75 W/m^2 K, the chip only reached a maximum of 400 K, safely below the chip cutoff. The full results of this study can be seen in Fig. 10. The full The drive circuit has a small heatsink on it, and is kept cool by a 70 mm computer fan, blowing directly on the chip and exiting the side of the enclosure. The fan is wired to the 12V supply on the circuit board, and is therefore always running when the stepper motor driver is powered.

$$P_{drv} = I_{out}^2 \times R_{drv} = 2.8^2 \times 480m\Omega = 3.37W \tag{2}$$

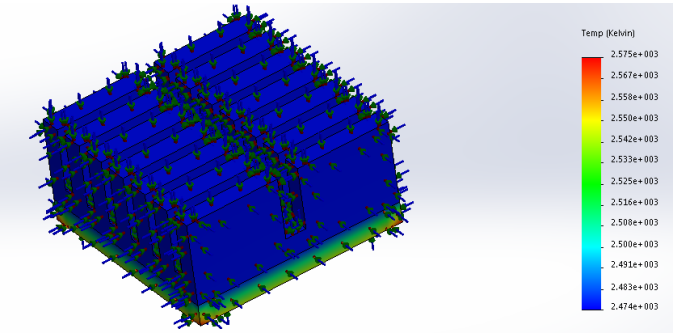


Fig. 9: SolidWorks thermal study without fan

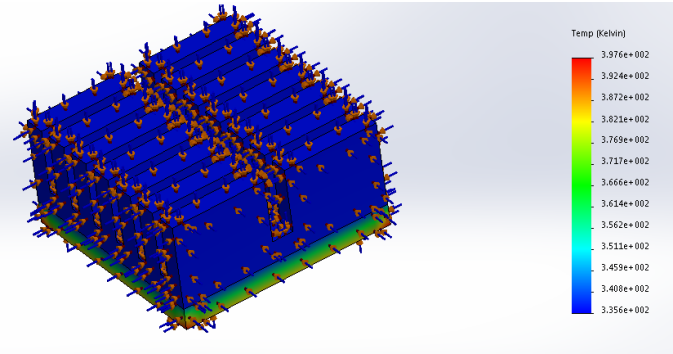


Fig. 10: SolidWorks thermal study with a fan

3) Custom Timing Belt Pulley: A custom timing belt pulley was built for the turntable. The pulley was designed with 126 teeth, giving it an outside diameter of 15.01". A drawing of a timing belt was used to reverse engineer the design of the pulley teeth in SolidWorks. Test pulley wedges were then laser cut, and test fitted with the timing belt. The width of the teeth were then slightly adjusted and tested again until a perfect fit was found. This process produced a timing belt pulley with almost no backlash or slop.

4) Tensioner: To keep the timing belt at a constant operating tension, a spring tensioner was designed. The tensioner can be seen in Fig. 11. A 1/2" thick L timing belt should have 25 lb of working tension. The spring tensioner must provide 25 lb of tension to the loose side of the timing belt, well within the max load of the chosen spring. When the motor accelerates, the tension increases on the tight side of the belt. The maximum tension on the tight side as a result of the motor is 23 lb. This is calculated in Eq. 3. Because this is lower than the tension supplied by the spring tensioner, the belt is always under tension on both sides. This reduces slop and backlash in the turntable. Sliding tracks in the base plate were designed such that the tension could be adjusted if necessary.

$$\begin{aligned} \tau = \tau_{motor} \times \frac{teeth_{load}}{teeth_{motor}} &= 16.44lb \cdot in \times \frac{125}{12} = 173lb \cdot in \\ \alpha = \frac{\tau}{I} &= \frac{173lb \cdot in}{1200lb \cdot in \cdot s^2} = 0.144\frac{rad}{s^2} \\ T = \frac{I \times \alpha}{r} &= \frac{1200lb \cdot in \cdot s^2 \times 0.144\frac{rad}{s^2}}{7.55in} = 23.07lb \end{aligned} \tag{3}$$

The tensioner was made out of aluminum for durability. The tracks in the base plate were counter sunk on the bottom of the plate, allowing the plate to sit flat. The nuts were then ground down on two sides to fit within the countersunk track, so the tensioner can be adjusted without accessing the bottom of the plate. This also increased the surface area of the contact points between the nuts and the base plate, reducing the chance the outside of the nuts would be stripped. A Hexagonal shaft was used to keep the pulley horizontal, and e-clips were used to hold everything together.

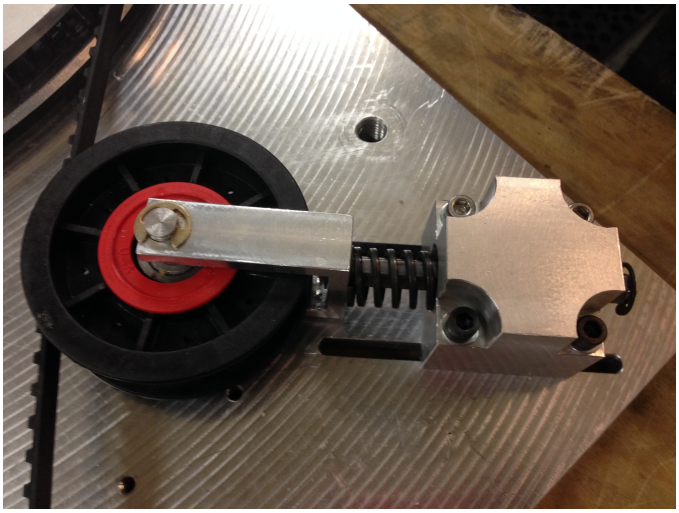


Fig. 11: Turntable Tensioner

5) Motor Mount: The motor mount and drive shaft support were carefully designed to support the load on the drive pulley. There is a bearing in the motor assembly, and a secondary bearing at the bottom of the shaft assembly. The free-body diagram in Fig. 12 illustrates the horizontal forces acting on the motor mount assembly. The calculation shown in Eq. 4 was used to ensure there would not be excessive overhung load on the motor bearing. It was assumed the maximum force applied by the belt would be twice the tension of the belt if each side of the belt around the pulley was pulling in the same direction. Knowing this, the overhung load on the motor bearing was calculated to be 14.7 lb. This is lower than the recommended maximum overhung load on the motor bearing of 20 lb.

$$\begin{aligned} \frac{1.25}{4.25} &= \frac{R2}{2 \times Fb} \\ R2 &= 14.7lb \end{aligned} \tag{4}$$

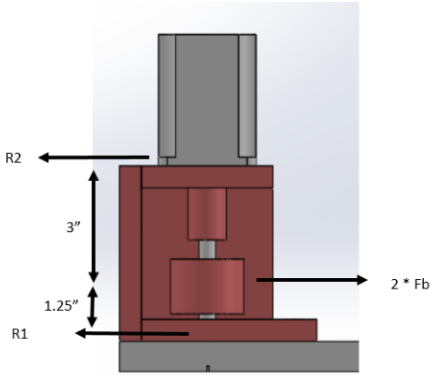


Fig. 12: Motor Mount Calculation

Once the design was confirmed to only need one added bearing, the rest of the support structure was designed. A box shape was used with two sides missing to allow room for the belt. The box was manufactured from four separate aluminum pieces for ease of manufacturing, and 1/2” thick plates were used to allow room for 1/4” bolts to secure the sides together. The 12 tooth pulley purchased had an inner diameter of 3/8”, and so a coupler was used to connect the 1/4” motor shaft to the pulley shaft. A clamp on rigid shaft coupling was used to properly tighten down on the two D shafts. This coupler also allows for easy maintenance, as the timing belt can easily be removed and replaced. A final picture of the motor mount assembly can be seen in Fig. 13.

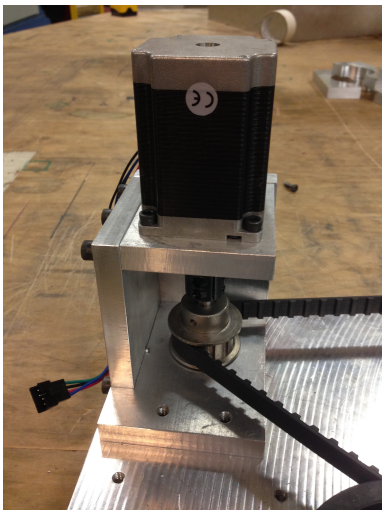


Fig. 13: Turntable Motor Mount

E. End of Arm Tooling (EOAT)

The purpose of the EOAT is to position a ball gauge in the fillet along each blade root such that a camera can take a picture of its fit while a light source backlights the ball. As positioning the ball gauge properly on the fillet is crucial, force feedback and single axis compliance were implemented to compensate for positioning errors in the system. A gauge selector was designed to provide automatic tool switching between the maximum and minimum ball gauges. The final design can be seen in Fig. 14.

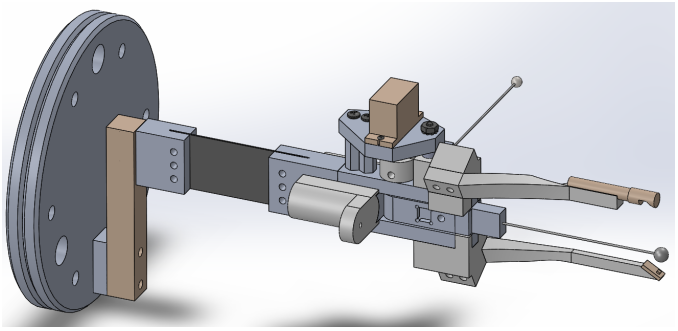


Fig. 14: The final EOAT design

1) Force Feedback: In order to ensure the accurate and secure placement of the ball gauge onto a given fillet, the system needs to be able to detect force on the ball gauge. In the first iteration of the EOAT design, the force sensitive component was an aluminum plate, 1/16” thick, with four strain gauges. The four strain gauges were arranged in a wheatstone bridge circuit to create a differential voltage signal from the compressive force on the end of the aluminum plate. This signal was then passed to an instrumentation amplifier to amplify the signal from its original μVolt range to a usable 0-3.3 Volt range. A standard 10K potentiometer fed through a unity-gain buffer was used to supply a reference voltage to the instrumentation amplifier. The unity-gain buffer is required due to the relatively low input impedance of the instrumentation amplifier reference pin. This circuit, shown in Fig. 15, put out an analog voltage signal dependent on the force on the plate, and needed an Analog-to-Digital Converter (ADC) in order to interface with the system controller.

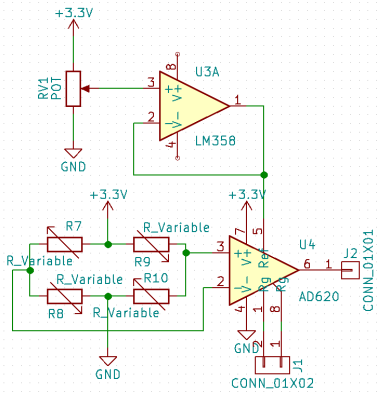


Fig. 15: Analog strain gauge conditioning circuitry. This circuit amplifies the signal from a wheatstone bridge and centers it around a reference voltage.

When tested, the circuitry did not act as expected. While the

strain gauge layout used was designed to eliminate any signal change due to bending forces, any bending forces applied to the sensitive aluminum plate showed up as a signal output. This was a problem because signal changes resulting from bending forces and compressive forces could not be decoupled. This created signal ambiguity that the system would not be able to interpret correctly. Therefore, a change in design was required.

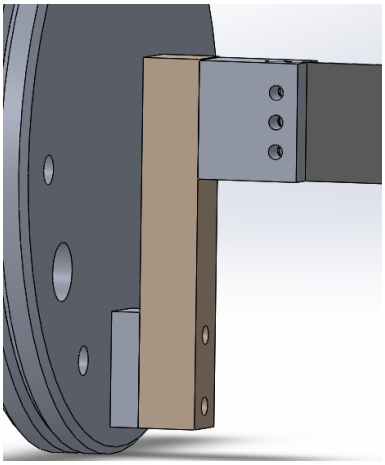


Fig. 16: The load cell, shown colored tan, mounted on the EOAT.

The new design, shown in Fig. 16, uses a single axis load cell mounted to measure compressive loads on the EOAT. Rather than manufacture a force transducer from aluminum stock, the team purchased a load cell, designed to operate with loads up to 5 kg. Because this load cell was designed for relatively small loads, the gain necessary to observe a usable output signal was much lower than previous designs. The load cell did not need to be connected to an instrumentation amplifier with high (>500 V/V) gain. Instead, a small board was purchased that had adjustable gain up to 128 V/V. This board has a 24-bit ADC, and uses bit-banging serial as a way to connect to the Raspberry Pi without requiring any analog pins. The purchased load cell isolates compressive forces from bending forces, and allows the system to observe the forces on the EOAT much more accurately.

Force sensing is handled on the Raspberry Pi as its own class. A separate class, contained within the force sensor class, handles communication with the ADC. Once a force sensor object is initialized, a function is called to zero the readings. This is done to set the idle point to read as a zero force on the Raspberry Pi. The zeroing function is called when the EOAT is positioned between the blades in the same orientation that it will be moving into the fillet. Once the force sensor has been initialized and zeroed, a software timing interrupt is set up to trigger to handle every new reading value. The function triggered by the interrupt returns the force sensing values to the Raspberry Pi. Based on these values, the Raspberry Pi can determine when contact has been made with the blisk. A flag is sent to the ABB controller once the Raspberry Pi senses contact. This force sensing functionality allows the ABB robot to update its path dynamically by providing a source of active feedback. The use of this force sensing to update the robot's path is further explained in Section III-I2, Generating the Paths.

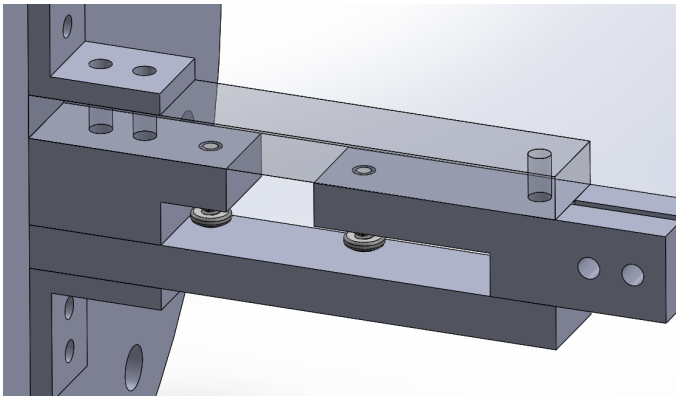


Fig. 17: The original spring-driven single axis compliance segment. With this design, a spring would have been attached to the two bolt heads shown.

2) Single Axis Compliance: As mentioned in Section II-B2, the previous EOAT had compliance about three axes, resulting in sag in the tooling. For this implementation, the EOAT was designed to have compliance about only one axis (the Y-axis of the tool's coordinate frame). This prevented the EOAT from bending as it pathed the fillets while still allowing for it to correct for positioning errors.

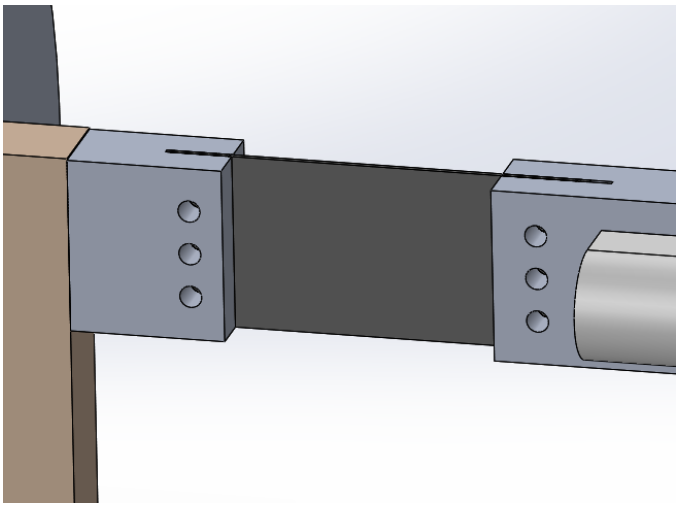


Fig. 18: The final design for the single-axis compliance segment. This design utilized a piece of spring steel, shown in dark gray.

The first compliance segment design can be seen in Fig. 17. With this design, the EOAT would be mounted on a cylindrical pivot. This would allow a horizontal force on the ball gauge to rotate the EOAT about only the Y-axis . A spring mounted

as shown would hold the EOAT centered when no force on the ball gauge was present.

This spring compensation design satisfied the need for single axis compliance. However, the complete segment was nearly three inches long. This length was excessive, as the team was working to keep the EOAT as small as possible. Therefore, the team designed a shorter section shown in Fig. 18 which used a thin segment of spring steel to generate the compliance.

There were a number of requirements which needed to be accounted for in this design. First, the ball gauge needed to be able to deflect 1/8" under 1/8 pounds of force. Second, the steel needed to be thick enough to prevent the EOAT from bending under its own weight. Third, the steel needed to flex more than the ball gauge shaft so as to prevent the gauge from being pushed out of the camera's view.

It was calculated that, with a segment of spring steel 1" x 1.5" x 0.032", all of these requirements would be met. 1/8 lb of force on the ball gauge would generate 0.05" of deflection in the ball gauge and 0.079" of deflection in the spring steel. This totalled to just over 1/8", as desired. The case which would have the largest deflection is when the EOAT is rotated so that the spring steel is flat and parallel with the ground. It was also calculated that, in this worst case scenario, sag in the EOAT due to its weight would cause a 0.012" deflection in the ball gauge. As this was well within the 1/8" of error tolerance the spring steel provided, the spring steel's size was deemed appropriate. Calculations used to determine these numbers can be seen in Equation 5, and the corresponding diagram can be seen in 19.

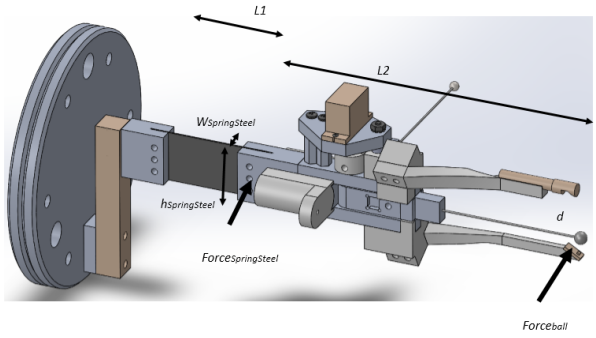


Fig. 19: Variables and dimensions used for deflection calculations

$$
\begin{aligned}
I_{shaft} &= \frac{\pi \times d^4}{64} \\
Deflection_{shaft} &= \frac{l_{shaft}^3 \times Force_{ball}}{3 \times I_{shaft} \times E_{Steel}} \\
I_{springsteel} &= \frac{h_{SpringSteel} \times w_{SpringSteel}^3}{12} \\
Force_{SpringSteel} &= \frac{Force_{ball} \times L_1 + L_2}{L_1} \\
Deflection_{SpringSteel} &= \frac{l_{SpringSteel}^3 \times Force_{SpringSteel}}{3 \times I_{SpringSteel} \times E_{SpringSteel}}
\end{aligned} \tag{5}
$$

F. Z-axis Compliance

Despite the initial belief that single axis compliance was desired, while testing it became apparent that Z-axis compliance was needed. This is because small deviations in the path significantly affected the quality of the image passed to the vision processing. A compliant section was therefore designed which made use of springs and shoulder bolts in through-holes. The design can be seen in Fig. 20. The purpose of this design is to allow the ball gauge to track the fillet even when the robot is not perfectly positioned to press the ball gauge into the fillet.

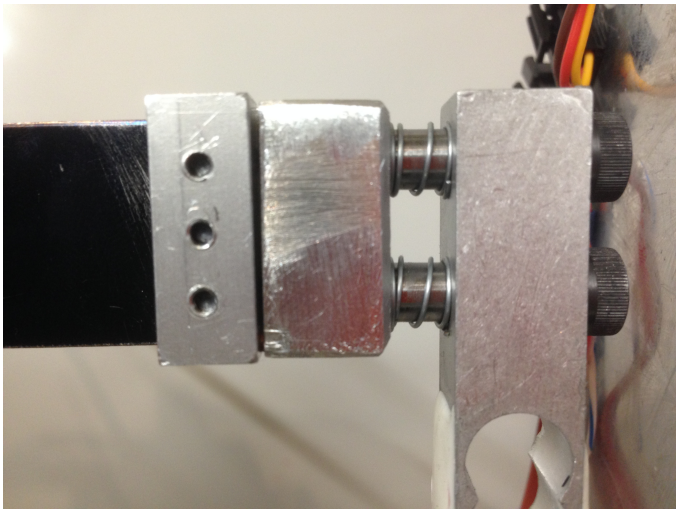


Fig. 20: Z-axis compliance section

1) Gauge Selector: A gauge selection component was added to the EOAT design to make the process more autonomous. The design, shown in Fig. 21, had both ball gauges mounted in an L-shaped pivot so that the shafts of each gauge were 90 degrees apart. The pivot would then be rotated into position by a servo motor. As the forces required to turn this mechanism were minimal, a micro servo was used. A concern with this design was that the ball gauges would need to be held in position by continuously driving the servo to the desired

position. It was decided that this was not ideal, and so multiple designs were made and tested.

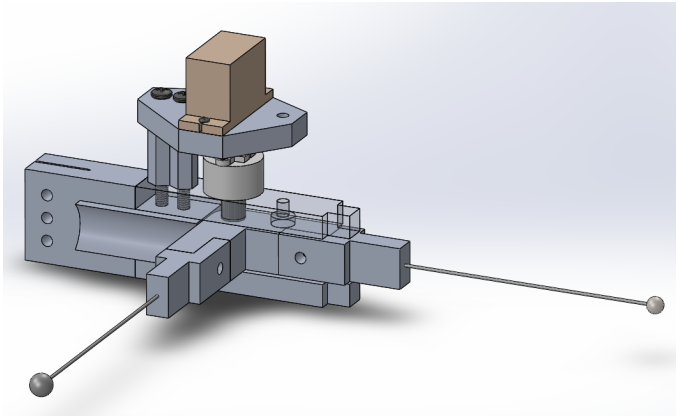


Fig. 21: The gauge selector design used a small servo to rotate an L-shaped piece holding the ball gauges.

The first locking mechanism, which can be seen in Fig. 22, consisted of two electromagnets mounted on each side of the EOAT. Using these electromagnets and the servo, two methods for switching the gauges were devised. In the first method, the servo was rotated to the left and then the left electromagnet would be powered, holding onto the metal screw in the pivot. To switch ball gauges, the magnet would be released, the servo would be driven to the right, and then the right electromagnet would be powered. This motion can be seen in Fig. 23. Using this method, the electromagnets would have to be powered for a majority of the inspection process.

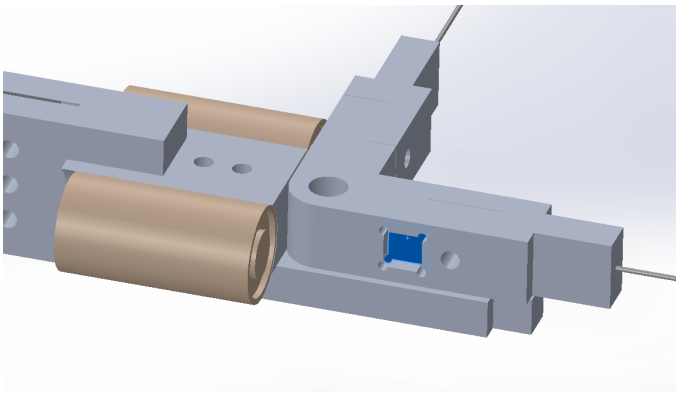


Fig. 22: The original locking mechanism design which used two electromagnets (colored tan) to hold the L-shaped pivot in the proper location. The ferrous metal or permanent magnet used in methods 1 and 2 respectively were mounted in the blue-shaded hole.

In the second method for switching the gauges, the screw in the pivot was replaced by a neodymium permanent magnet. A magnet with a 2.2lb holding force was used so that the permanent magnet could passively hold onto the metal of the

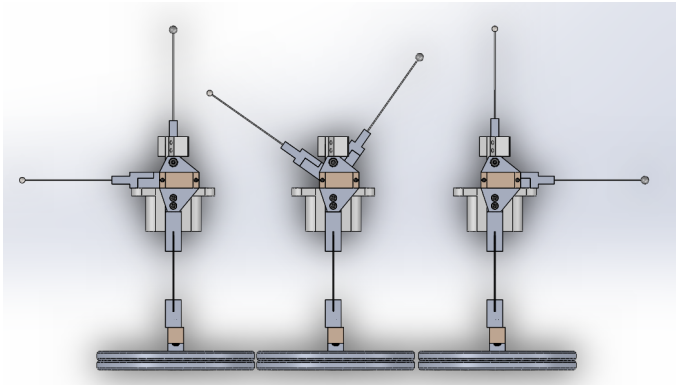


Fig. 23: The motion of the ball gauges as the selector rotates.

electromagnet. When a switch was required, the electromagnet would be powered such that both magnets had matching poles, causing a pushing force to be generated. This would initiate the rotation, which would then be completed by the servo. Once turned far enough, the other permanent magnet would generate a pulling force between it and the electromagnet, finishing the rotation and holding the gauge in place. Using this method, the electromagnets would only be powered for a fraction of the total run time of the inspection process.

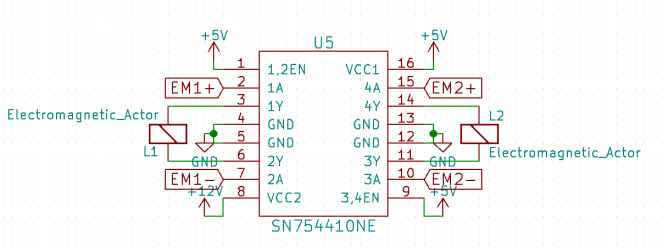


Fig. 24: The circuitry designed to run the electromagnets. In particular, this method was designed for method 2, in which the polarity of the electromagnets has to switch.

The circuitry required to drive the electromagnets in both directions can be seen in Fig. 24. It consists of a quadruple half-H bridge chip, the TI SN754410NE, powered by an external source and controlled by the Raspberry Pi. It is capable of powering both of the electromagnet loads with the positive supply voltage, as well as powering the electromagnet loads with the polarity reversed on the supply. This allows the EOAT gauge selector to run the electromagnets to attract the permanent magnets during the switching phase to pull the arm towards it. The electromagnet can then shut off, and allow the permanent magnet to hold the arm. When the tool must be switched again, the electromagnet is switched on in the opposite direction, repelling the permanent magnet.

One major benefit of this holding method is that the electromagnets do not run for a long period of time. This reduces heat buildup. This design can also provide strong holding forces with a reliable release mechanism.

Using the diagram shown in Fig. 25, it was determined that the electromagnet would need a holding strength of 1 lb. Because the shipping time for a 1lb electromagnet was nearly a month, a 24V, 0.5lb electromagnet was ordered which, while not strong enough for the final design, could be used for testing to verify the magnets worked as desired.

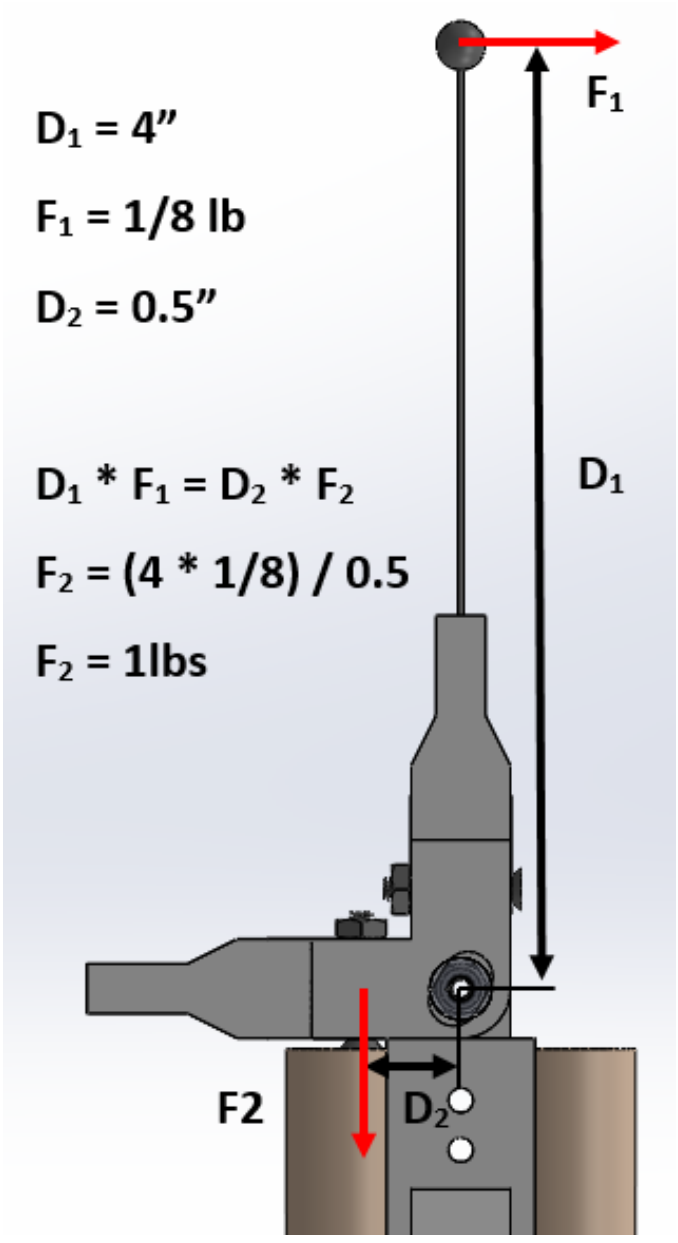


Fig. 25: A free-body diagram of the gauge selector when acted upon by the pushing force of the blisk and the holding force of the electomagnet.

The team conducted tests that confirmed that the magnet would hold metal objects only when the object was touching (or nearly touching) the magnet. However, it was also noted that when the magnet was run at half its rated max power or more, the casing became very hot. For this reason, the team designed the system to use electromagnets at least twice as powerful as needed to minimize the heat they would generate. A 12V, 11 lb electromagnet was chosen as it would both generate less heat, and was small enough to fit easily into the gauge selection design.

Method 2, using permanent magnets which would passively hold the gauge in place, was tested first as only running the electromagnets to release the gauge during a switch would be better than constantly running the electromagnets. The test was conducted without the servo initially so the team could test the interaction between the electromagnet and permanent magnet alone. The team was surprised to discover that powering the electromagnet appeared to have no effect. Though the poles of the two magnets should have repelled each other, the electromagnet and permanent magnets still held together. After further testing, the team concluded that this occurred because the permanent magnet was overpowering the electromagnet. This method could therefore not be used.

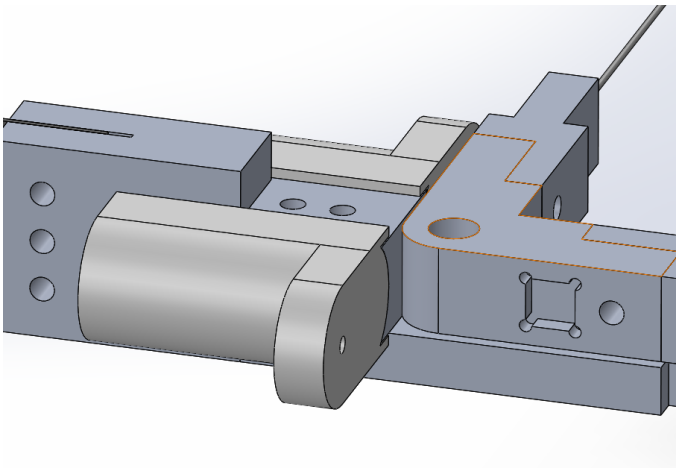


Fig. 26: The hard stops, shown colored white, designed to keep the gauge selector from rotating too far.

After method 2 was determined unusable, method 1 was tested. This method worked as expected, in that the electromagnet held the pivot in place when a ferrous metal was attached to the pivot. However, there were two major issue. First, running the electromagnets for an extended period of time generated a significant amount of heat. Second, the holding force did not meet the 1 lb requirement. The team tested many different ferrous materials, including nuts, bolts, and small pieces of steel. In all cases, the holding force was not strong enough. The team concluded that the ferrous components used were too small. However, bigger components could not be used due to EOAT size restrictions. Therefore, method 1 could not be used either.

As neither method devised for the electromagnets would work, the team needed to devise an alternate method. The team decided to implement the simple solution of driving the servo to a hard stop. Robot pathing would then be adjusted to

always apply force to the ball gauge pushing it into the hard stop. This would remove the need for the servo to be running at all times to hold the ball gauge in place. The hard stops, shown in Fig. 26, were designed to fit into the cutouts used previously by the electromagnets. A screw was added such that the exact positioning of the gauges could be adjusted. Though this method worked, it was quickly determined that the servo used was not robust enough as the plastic gearing was stripped after a few hours of heavy use. The team therefore purchased a similarly sized servo with metal gears as a replacement which was expected to have improved durability and longevity.

Inspecting the three LEAP series blisks requires the use of eight different ball gauges total. As alignment of the ball gauges relative to the LED and camera was critical, a modular ball gauge mount was designed to accommodate each ball gauge. This part would compensate for the varying shaft diameters, shaft lengths, and ball gauge diameters needed to inspect all the LEAP series blisks. As can be seen in Fig. 27, the original mount design meshed with the L-pivot using one larger cylinder and four small pins. These would keep the mount positioned properly on the L-pivot. The two spring steel clips on either side of the mount would then hold the mount to the L-pivot, allowing this part to be easily removed and replaced with a similar piece holding a different ball gauge.

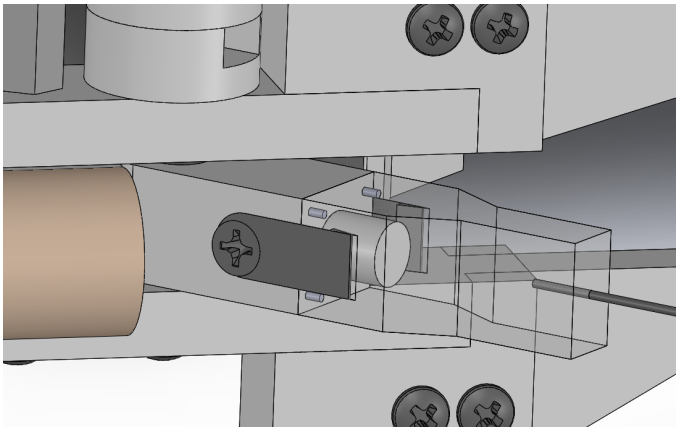


Fig. 27: The original gauge mount design, shown transparent to provide vision of the mounting pins.

The other half of the mount was a thinner, 0.25” wide segment with a hole drilled into the end to press fit a ball gauge shaft into it. This hole diameter and depth could be changed for each ball gauge, ensuring the tip of the ball gauge was always positioned in same place in the camera’s view.

Though this design would have been easy to operate, the team determined that the manufacturing process would be too difficult given that it was to be made out of aluminum and would require tiny, precisely placed pins in order for the mount to line up correctly. Therefore, the mount was redesigned, the result of which can be seen in Fig. 28. As can be seen, the design used a single screw to attach two interlocking L shaped pieces. This design was much easier to manufacture, and would still be fairly simple to operate assuming the operator had the proper screwdriver.

The servo to control switching between the ball gauge sizes is connected to the Raspberry Pi by a signal and power pin. When switching between the two sizes, power is first sent to the power pin. A software PWM signal is then sent to the servo signal pin to change the position of the servo. Originally this was a hardware PWM signal, but the hardware PWM library had a conflict with the library needed to read the load cell ADC. When a hardware PWM signal was sent to the servo, the load cell ADC would not return any values. Switching to software PWM resolved this issue.

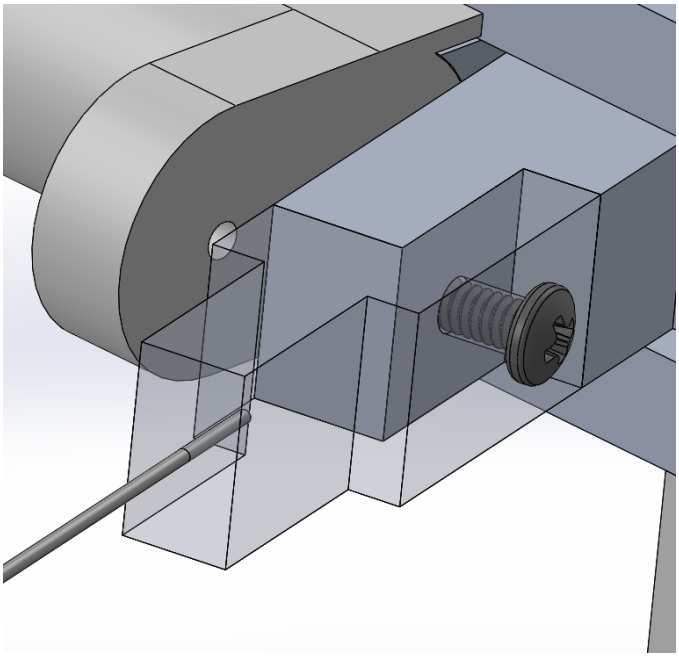


Fig. 28: The final gauge mount design, also shown transparent.

2) Ball Gauges: Originally, the team looked into obtaining ball gauges identical to the those used by GE. As GE could not provide them, the team contacted the manufacturer. The quote the team received indicated that the 8 gauges would total upwards of $2600 and would take 3-4 weeks to arrive. Though the team’s budget was big enough for this expense, time constraints made this an unfeasible option. The team therefore requested quotes from other manufacturers, looking to reduce both the price and shipping time of the gauges. One manufacturer quoted the gauges at approximately $360 with a 1-2 week arrival time. Though the gauges would not have the same level of tolerance, they were accurate enough for this project, so the team ordered them.

The ball gauges did not arrive for almost five weeks. The team therefore 3D printed plastic ball gauges to use for testing. The printed gauges, one of which is shown in Fig. 29, combined both a ball gauge and the modular gauge mount into one component to simplify their use. Though originally white, the printed gauges were colored black with a Sharpie to reduce the amount of light penetrating through the plastic

from the LED for vision testing.

Once the metal ball gauges arrived, it was apparent that they were ordered using the radii of the balls, not the diameters. This meant only two of the gauges were usable, as the rest were too small. As another five week delay would put the project too far behind, the team looked into other ball gauge options. A series of ball bearings without shafts were purchased as they had fast shipping and were inexpensive. Though these balls did not perfectly match the desired sizes, they were within 0.006" of the desired sizes, and so could be used as proof of concept. These balls were mounted on the end of 3D printed shafts. The result can be seen in Fig. 29.

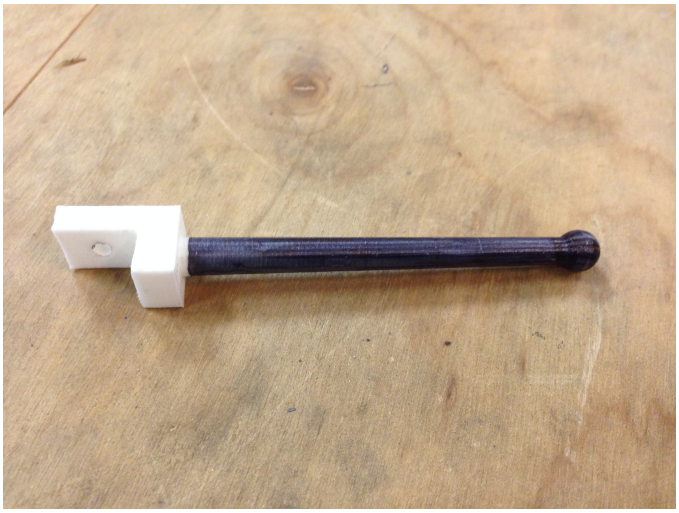


Fig. 29: One of the ball gauges 3D printed for this project. Black Sharpie was used to reduce the light diffusing through the plastic.

3) Light Source: The team decided the light source needed to be fully hidden behind the ball gauge to generate consistent images. A small light source would also keep the EOAT more compact, leaving more clearance within the blisk blades. The team therefore purchased a standard surface mount LED and some blank PCB board. It was later determined that this LED was not bright enough, as the vision system was not able to detect enough light passing the ball gauge. A super bright surface mount LED was purchased to replace the standard surface mount LED.

The surface mount LED was mounted on a 1/8" x 1/4" board based on the circuit in Fig. 30. This board size ensured that it would be small enough to be hidden behind the smallest ball gauge (0.1" diameter) in any picture. The final design for the arm positions the LED such that it pointed at the contact point between the ball gauge and the fillet. The arm was designed to have a curve along its length to fit properly. This can be seen in Fig. 31 and is discussed further in Section III-F5.

4) Camera: As was discussed in Section II-B2, the camera used in the previous project had a number of drawbacks, so a new camera was needed. The team looked into a number of options, judging them based on criteria such as size, cost, and

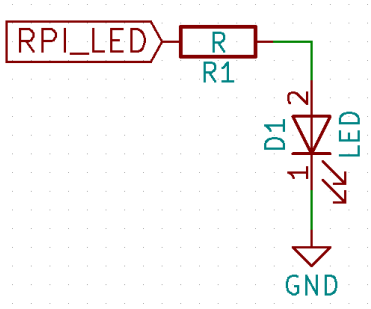


Fig. 30: The circuit diagram for the surface mount LED.

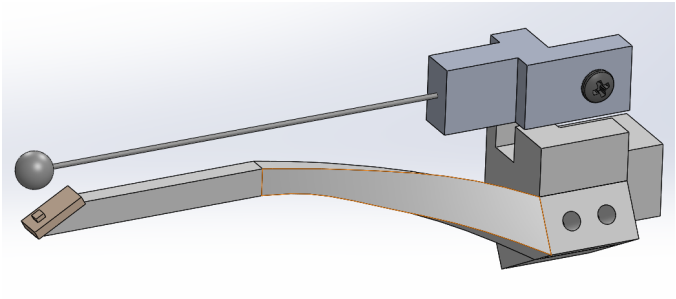


Fig. 31: The final design for the LED arm. The shape of this arm is discussed more in Section III-F5

ease of implementation. The sections that follow discuss the team's findings on each camera option.

The team looked into borescopes, which use fiber optics to allow a camera to view objects in spaces too tight to fit the camera itself. Flexible borescopes were determined to be the ideal solution, as they were both small enough and flexible enough to be positioned closely to the ball gauge. However, these devices are extremely expensive; one flexible borescope would have used most, if not all, of the project's available budget. Therefore, it was decided that a flexible borescope would be used only if no other solution could be found.

A similar decision was made for rigid borescopes. Rigid borescopes, in general, are much cheaper than flexible borescopes. It was determined that the ideal viewing angle for a rigid borescope would be approximately 70°. This is because a larger viewing angle forces the scope to be mounted much closer to the blisk, which introduces a risk of collision between the scope and the blisk, while a smaller viewing angle forces the scope to be mounted at an angle to compensate, introducing clearance issues between the scope and the blades. Because of this viewing angle requirement, the price of a rigid borescope was not much lower than the price of a flexible borescope. Therefore, it was once again decided that a rigid borescope would only be used if no other solution could be found.

The team next looked into using a smaller endoscopic camera. Two possible cameras were discovered, both produced by GiraffeCam. The first was the GiraffeCam 1.0. This camera had the same resolution as the camera used in the previous project, but it was approximately half the diameter and came

with a threaded mirror attachment. This camera was appealing because the smaller diameter meant it could easily fit between the blades of the blisk. In addition to this, the mirror had a solid connection to the camera, meaning the system would be reliable. Finally, this camera was cheap, costing only $20. This meant the camera could be bought and tested before a final decision was made.

Fig. 32: The GiraffeCam 1.0, a flexible endoscopic camera.

The other camera the team looked into was the GiraffeCam 3.0. This camera takes 2 megapixel images, which meant it would generate clearer images when mounted a similar distance from the ball gauge. The camera's diameter was larger than the GiraffeCam 1.0. This camera did not come with a mirror attachment, meaning the team would need to design a mirror mount into the EOAT. Additionally, this camera was more expensive, costing $60.

The team decided to purchase the GiraffeCam 1.0, shown in Fig. 32, for three main reasons:

1) It came with a mirror attachment.
2) It was small enough to easily fit between the blades of a blisk.
3) It was cheap enough to buy just for testing purposes.

The team tested the camera to verify that it would work for the system. The camera was rated for an ideal focal length of two inches, however this distance was too far for a compact EOAT design. To test whether the camera could generate sufficient images with objects closer than two inches, a simple test was set up. A quarter was positioned 1.5", 1.0", 0.75", and 0.5" away from the camera, and a picture was taken at each distance. The resulting images can be seen in Fig. 33. As can be seen, the camera has fairly clear images up to 0.75" away. At 0.5", the image starts to blur significantly. The constraint determining how close the camera needed to be to the ball was the distance between stages of the multi-stage blisk. This distance was measured at 2", meaning this camera could be used as there was enough space to avoid mounting the camera 0.5" from the ball.

Fig. 33: The GiraffeCam at Different Distances (1.5 top left, 0.5 top right, 1 bottom left, 0.75 bottom right)

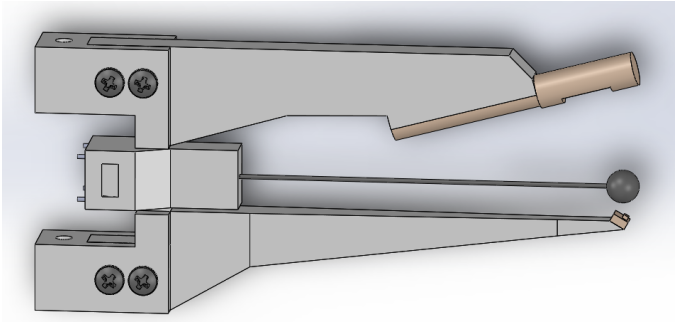


Fig. 34: The original designs for the camera and LED arms. Clearance issues meant these straight arms would not work.

5) Form Factor Restrictions: In their original designs, the arms holding the camera and LED, shown in Fig. 34, were designed to be straight. It was believed that by designing the arms to be 1/4" wide, there would be enough clearance to maneuver the arms along the fillets of each blisk. Though this held true for the small-bladed blisk, blisk P01, a project conducted by two of the team members for a separate class showed that the arms did not have enough clearance to path the largest-blade blisk, blisk P02.

Knowing this, the arms were redesigned. The results, shown in Fig. 35, consist of two straight segments which hold the camera and LED, and two curved segments which provide the necessary clearance. The curve was designed by examining blisk P02 in SolidWorks. It was determined that at the center point of the fillet, a line running tangent to the fillet had an angle of approximately 30 degrees from vertical. The same point on the end of the blade had an angle of 60 degrees from vertical. This meant there was approximately a 30 degree change over the length of the blade. This held approximately

true for all points along the fillet. The curved segment of the camera and LED arms were therefore designed similarly, curving 30 degrees over their length. When tested, this design successfully fit between the blades of all four stages of blades the team needed to inspect.

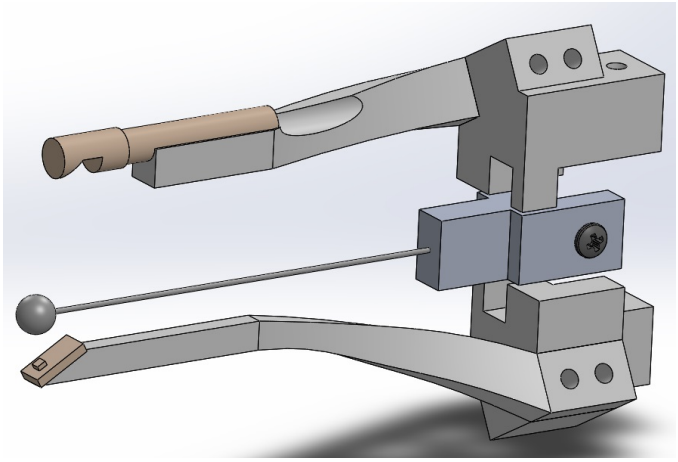


Fig. 35: The final design for the camera and LED arms. The curve matched that of the most complex blades, allowing the EOAT to utilize more of the available space between each blade.

6) Locating Blade One: The turntable design contains no sensor for absolute position for two reasons: the stepper motor has very precise relative position capabilities, and the blisk is placed arbitrarily on the turntable, so locating the position of the turntable does not solve any problems. The system had to therefore be able to accurately locate the first blade on the blisk relative to the robotic arm and turntable position. Doing so would allow the turntable to use differential position measurements to place the other blades in the proper inspection location. By using a differential measurement based on steps of the stepper motor, the system can eliminate any error based on placement of the blisk by the user.

Locating blade 1 is a multi-step process. As described in Section III-J, the operator must first place the blisk on the blisk stand without fully clamping it down. The robot then moves the ball gauge of the EOAT just outside the blisk's radius, and the operator has a chance to adjust(rotate) the blisk to ensure that the EOAT will be inserted between two blades. Then, the operator indicates that the blisk is in position and locks the blisk onto the turntable. The robot moves the EOAT between the blades, and the turntable rotates until contact is made between the side of the LED arm and the blisk.

The system senses if a circuit has been completed to determine if contact has been made with the blisk. A single wire is extended from the ball gauge tip and continues down the robot arm to be connected to a digital input on the Raspberry Pi configured with a pull-up resistor. A grounded copper clip is attached to the blisk to ground it. This circuit is shown in Fig. 36, with the blisk-wire interface being represented by the switch. The titanium blisk is conductive, so when the ball

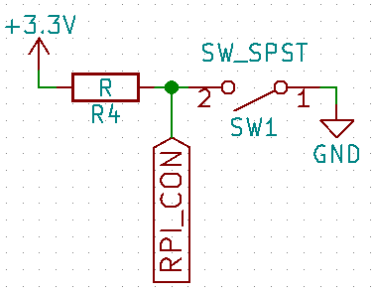


Fig. 36: Contact Sensor Circuitry

gauge touches a blade, the input signal for the Raspberry Pi switches from logic high to logic low. This signals to the Raspberry Pi to use the current position of the turntable stepper motor as the zero position. From this fixed point, the location of the blisk blades is synced between the Raspberry Pi and the IRC5 controller.

G. Computer Vision

Computer Vision was used to determine whether each blisk passed inspection by examining individual images along the blade fillets. The team decided that an image passed if the correct number of light regions could be identified. When using the small ball gauge in the blade fillet only two light regions should be visible. If three light regions are visible then the fillet is undersized. When the large ball gauge is being used, three light regions must be visible for the image to pass inspection. These areas of light are shown in Fig. 37. In order to identify the light regions easier, a bright green LED was used. The bright green color is easier to identify in image processing as it stands out from the rest of the image.

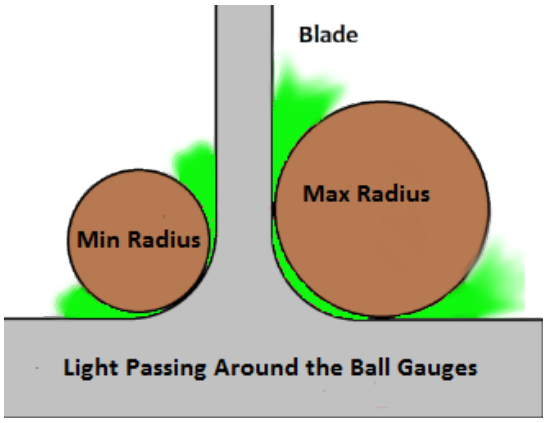


Fig. 37: Light passes around each ball gauge to form lighter areas on the sides and in the middle of the large tolerance gauge, and only forms light areas to the sides of the small tolerance gauge

An important metric for the system is its spatial resolution. This is the number of image frames evaluated per unit distance.

This is important for the inspection system because if the space between each image becomes too large, it is possible that a flaw in a fillet would be missed by the system. Through discussion with GE, it was determined that a minimum spatial resolution of 25 captures per inch (CPI) was acceptable.

In order to determine the spatial resolution, the team first found the distance that the ball would have to travel pathing fillets during the inspection. These distances for each blisk can be found in Table IV in Appendix K. The team then determined how long it would take for the camera to path the fillets.While the team aimed for less than an hour of inspection time per blisk, when calculating the potential spatial resolution the actual time required for an inspection was unknown. The team picked a few different scenarios for the inspection time: If the system were as slow as acceptable, it would take an hour. If the system were very fast, and moving quickly enough to risk breaking parts of the EOAT, the inspection would take 15 minutes. A few other time frames within these bounds were also explored. These can also be found in Table IV in Appendix K.

The processing frame rate of the computer vision is essential to the calculation of the spatial resolution. The frame rate of the camera itself is 30 frames per second (FPS), but the FPS of computer vision processing is what limits the system. The processing FPS was determined once the entire system was completed. When calculating the expected spatial resolution, however, an expected frame rate of 10 FPS was used. The lowest expected spatial resolution was 36 CPI, using the highest pathing speed and longest blade length. All expected numbers for spatial resolution can be found in Table IV in Appendix K.

1) Choosing an Image Processing Toolbox: The two main computer vision platforms that the team chose between were OpenCV and the MATLAB Image Processing Toolbox. Compatibility with a Raspberry Pi was seen as the most important factor. Familiarity with the programming language was seen as the second most important factor, since being more familiar with the language would ease the learning process. Lastly, cost and availability of online resources were other factors the team considered, as both were important.

There are a few benefits to using the open-source image processing toolbox OpenCV, including the availability of resources and the ability to use OpenCV with C++ or Python. There are multiple tutorials online on how to install OpenCV on a Raspberry Pi, whereas there was no proof that the MATLAB Image Processing Toolbox could be installed on the Raspberry Pi. The cost of the MATLAB Toolbox was also a downside, especially compared to OpenCV which is free and open source. The decision matrix summarizing these factors can be seen in Table II. The decision matrix was developed by ranking the importance of the categories for an image processing toolbox, and ranking them on importance with a high ranking indicating a more essential factor. Based on this matrix, the team chose to use OpenCV.

	Raspberry Pi Compatible	Cost	Familiarity with Language	Online Resources Available	Total
Weighting Factor	4	2	3	1	
OpenCV	10	10	8	8	124
Matlab Image Processing Library	2	2	3	4	30

TABLE II: Computer Vision Decision Matrix

2) GRIP for Testing: The Graphically Represented Image Processing engine (GRIP) was used to test OpenCV with pictures of the ball gauge resting in the fillet. GRIP is a User Interface to easily test OpenCV functions. This allowed the team to identify the regions of light passing around the ball gauge. Fig. 39 shows the initial results of using GRIP to identify the contours of green light in the image.

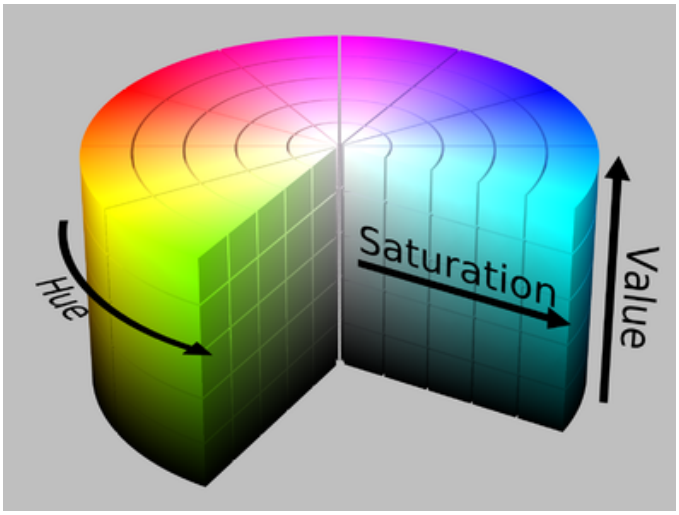


Fig. 38: Hue Saturation Value (HSV) Color Representation

In Fig. 39, the picture on the left shows the initial image of the ball gauge with the LED positioned behind it. As can be seen, the ball gauge was larger than the blade fillet, allowing light to shine past the ball gauge in three different regions. The goal of the inspection process was to detect how many regions of light were present. The GRIP screenshot shows the initial image being processed by a Hue Saturation Value (HSV) Threshold. This allows the user to specify a range for the hue, saturation and value to identify only the parts of the image that meet those standards. The hue specifies what range of the color spectrum that should be searched for. By using a range of 26-71 any green in the image was identified. The value and saturation are both specified as the top part of the range since a bright green is being searched for. Fig. 38 illustrates the HSV visual color representations. After the green regions of light were identified in the image, the contours of these shapes were found using the OpenCV findContours method. These contours are shown in the right picture in the GRIP

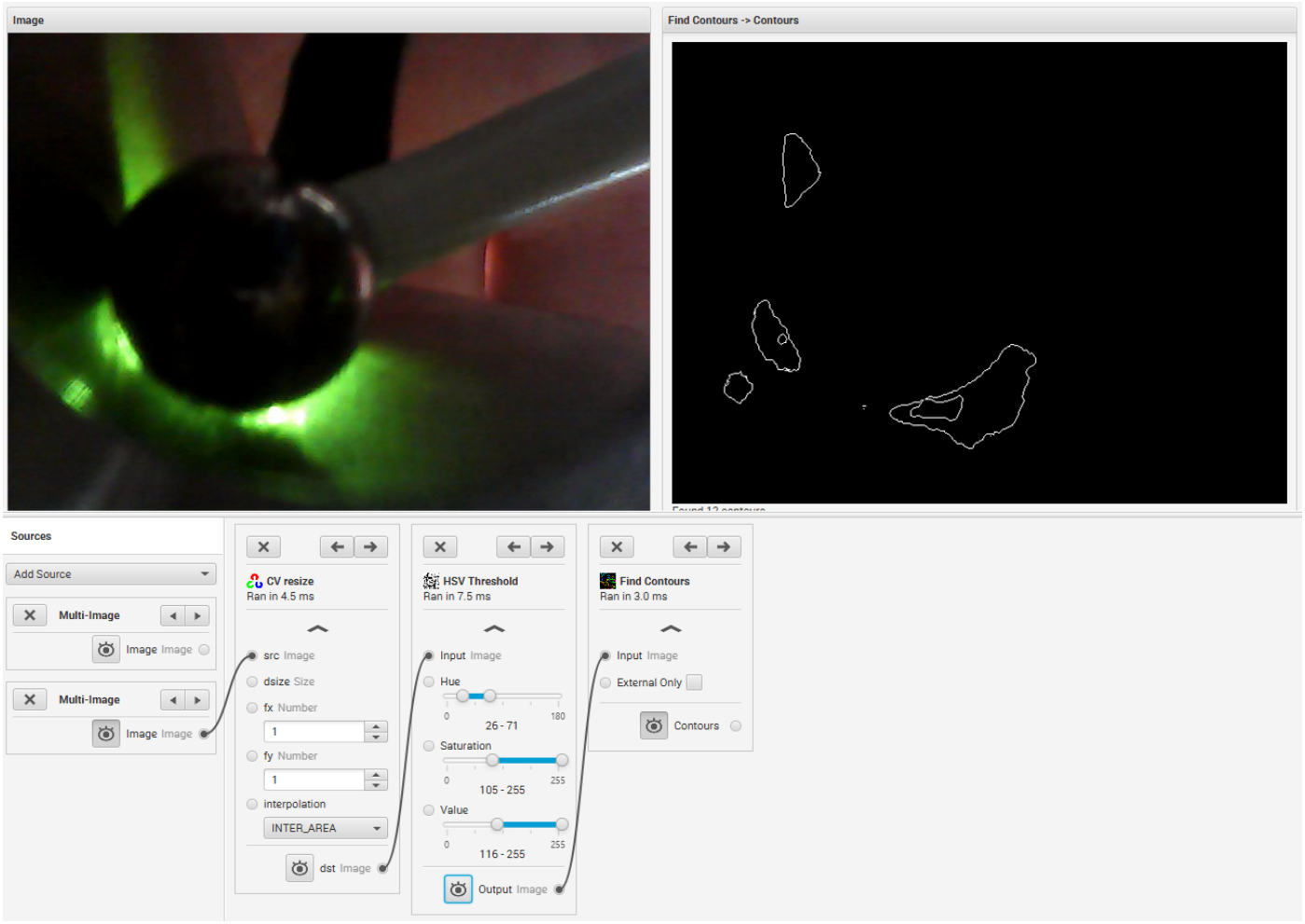


Fig. 39: GRIP for Testing Identifying the Regions of Light

screenshot. By using GRIP to identify the green regions of light and identify the contours, the optimal HSV ranges could be incorporated into the final computer vision algorithm.

3) Computer Vision Algorithm: Computer vision was used to determine whether an image of a ball gauge in a fillet passes. The function to handle the inspection first captured the current image from the camera. The function was passed a fillet position class indicating which ball gauge size was being used and the current position on the blisk. The frame is first converted from RGB color scheme to HSV so that the bright green light can be identified. Once the image's color scheme has been converted, the OpenCV inRange function is used to select only the bright green shapes in the image. After identifying the bright green regions, the contours of these regions are identified with the OpenCV findContours function. These contours are shown in red in Fig. 40. An array of contours is used to store the outlines of each of the bright green regions. The algorithm then iterates through each of the contours to identify the centroid of each region, shown as black circles in Fig. 40.

The algorithm next identified the location of the ball gauge in the image. This was so that the inspection algorithm could dynamically adjust during a fillet inspection. The ball gauge was detected by first filtering the image to accentuate the edges of the ball gauge. The Hough Circles OpenCV function was then used to identify any circles with diameters similar to the current ball gauge diameter. If a circle of the correct diameter and relative position is detected then the stored position of the ball gauge is updated to move in the appropriate direction. This position is updated using a weighted average to slowly adjust the ball gauge position over time while decreasing the effect of incorrect circles detected. The detected ball gauge can be seen outlined in blue in Fig. 41.

Three different regions are positioned in the image based on their proximity to the ball gauge where light was expected to pass through. The regions locations are adjusted based on the current ball gauge location. The algorithm iterates over every contour and increments a counter if one of the centroids was located within the region. If the small gauge was being used, the algorithm then checks that there was a centroid within the

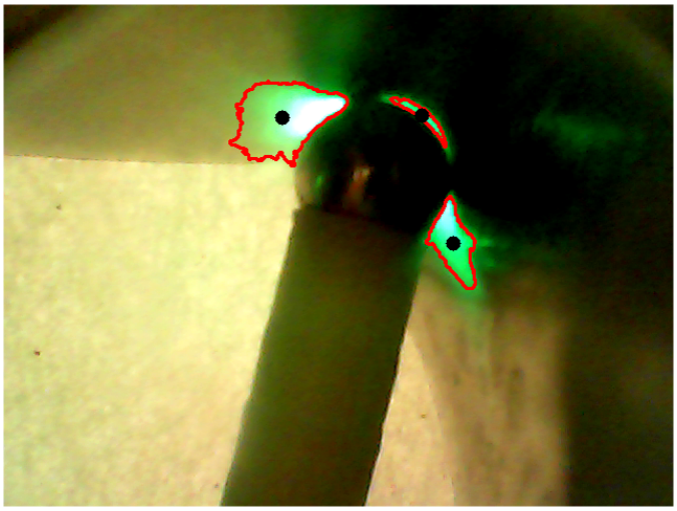


Fig. 40: The large ball gauge seated in a fillet with identification of the green regions of LED light. The contours of the regions of light are shown in red, and the centroids are shown as black circles.

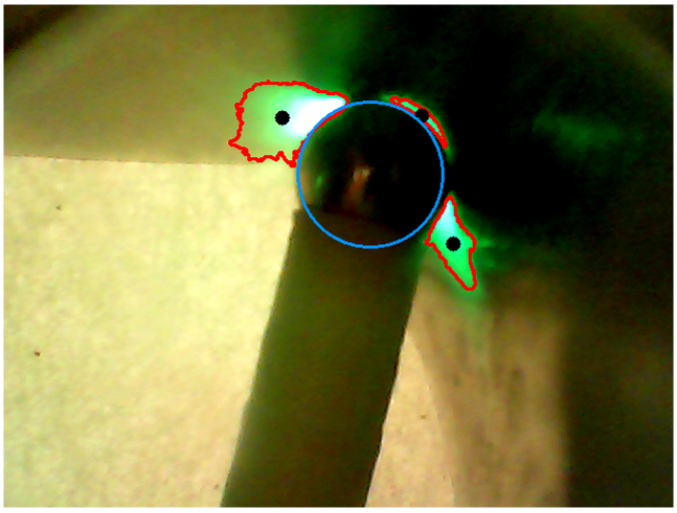


Fig. 41: Identifying the ball gauge location, shown as the bright green circle.

outer regions, but not within the center region. Alternatively if the large ball gauge was being used, the algorithm checks that there were centroids of light in all three of the regions. If these conditions were met for the current ball gauge, then the image passes. A final passing image can be seen in Fig. 42. Otherwise, the image failed. The inspection result was stored in a CSV file for the end user to view. The file contains the position along the fillet and whether or not the position in the fillet passed inspection. A sample results CSV file can be found in Appendix L.

H. Software System

The following section elaborates on the software controlling the project. It will first describe the class structure implemented for the project, and then discuss the communication and control between the Raspberry Pi and IRC5 Controller. Finally the section will wrap up with the overall code flow and execution of the software.

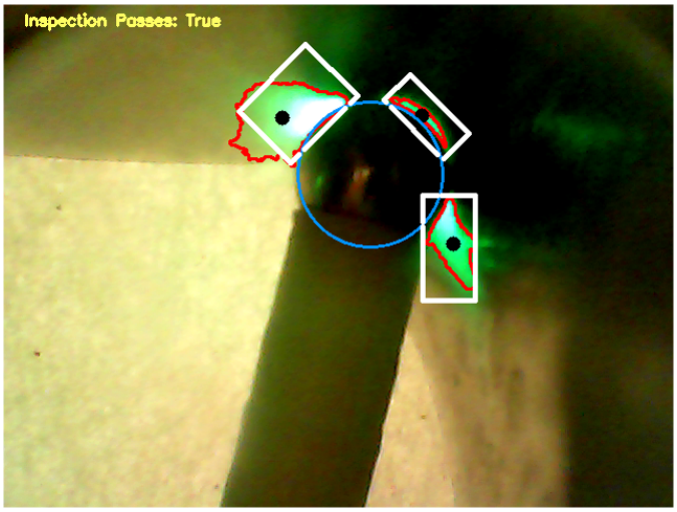


Fig. 42: The final passing image with the regions of expected light shown as the three white boxes.

The team choose to use Object Oriented Design (OOD) for the software structure as the project had functionality that could be neatly encapsulated by classes. The control of each sensor and actuator was encapsulated in its own class. This allowed for the code corresponding to the different elements of the system to be easily separated and abstracted from the top level application layer.

The team decided to use the Entity Boundary Controller (EBC) design pattern for the software as it provides an organized way to handle input from a User Interface to control the hardware classes. The EBC Design Pattern uses three tiers of classes to organize the software design. The entity classes are the classes that represent the components in the system such as the turntable, the blisk, or the EOAT gauge selector. The boundary classes are the classes that represent the different components of the User Interface (UI) of the application. The controller classes are what handle the communication between the boundary and entity classes. The following sections will discuss the classes chosen as entity, boundary, and controller classes, and the reasoning behind the design decisions.

1) Entity Classes: The entity classes are the classes which represent the physical components and processes of the system. The different entity classes used in the software design are shown in Fig. 49 in Appendix 49.

The first of these entity classes is the turntable. This class is responsible for controlling the stepper motor to turn the turntable. The second entity class is the ABB Robot class, which controls the movement of the robotic arm. This class

contains methods which interface with the IRC5, directing the arm's movements around and along each fillet. The ABB Robot class also handles the TCP communication with the IRC5 controller by containing a Socket Server Thread object.

Another entity class used in the software design is the LED class. This class handles turning this light source on and off and dimming it as appropriate. The Force Sensor class is used to receive input from the ADC connected to the load cell. The Gauge Selector class controls the switching between the two different ball gauge sizes on the EOAT by driving a servo. The software design also includes the Circuit Completer class that is used to determine if the EOAT has made contact with the blisk.

One of the most essential classes to the system is the Image Processor class. This class is responsible for the computer vision and determining if the fillet passes or fails inspection. Two of the final classes are the Blisk and Stage classes. Each blisk can be comprised of either one or two stages. These classes are used to store all of the specific information about the three different blisks that the system can inspect.

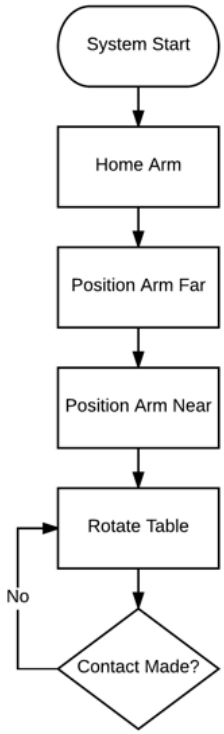


Fig. 43: Inspection Set Up Flowchart

Lastly, and most essentially, is the Blisk Inspection System class. This is the class that is responsible for using all of the other entity classes in order for the system to function correctly. The Blisk Inspection System class has a copy of every other class to interface with the different components of the system. The Blisk Inspection class also stores the three different blisks that it can inspect.

2) Boundary Classes: The boundary classes of the EBC Design Pattern are the classes which define each of the components of the User Interface (UI) for the blisk inspection application. Images of the UI design can be found in Appendix I. The UI was written in Python on the Raspberry Pi using the TkInter library. Fig 47 in Appendix D illustrates the class diagram of the boundary classes used.

3) Controller Classes: The controller classes handle the communication between the user interface and the entity classes. A different controller is used to handle each action that a user can make with the application. Fig. 48 in Appendix E illustrates the class diagram of the controller classes used.

4) Communication with the IRC5 Controller: The Raspberry Pi had to communicate with the IRC5 Controller to control the pathing of the ABB Robotic Arm. This communication was accomplished using a TCP connection. Both the Raspberry Pi and IRC5 Controller were connected to the same router through Ethernet. Though there were a few communication options, including digital IO lines and a UDP socket, the team chose to use a TCP connection because it guaranteed no packet loss.Though UDP is faster, packet loss would be an issue. The main downside of using the digital IO lines was that the number of commands that could be sent through signals was much more limited than the number of commands which could be sent in packet format.

The RAPID program running on the IRC5 was set up as a TCP Client. The main structure of the program consisted of waiting to receive a command from the Raspberry Pi, then handling that command appropriately. At the top of the main polling loop the program had a blocking receive call to get any message transferred over TCP. The message was then parsed to execute the correct command. Based on the command, the IRC5 Controller would direct the robot to execute a path. Once the path was complete, the IRC5 would send a response message to the Raspberry Pi, indicating the command completed.

In order for the Raspberry Pi to have other functionality while the robotic arm was moving, the Python program was multithreaded. This functionality was necessary when the arm was pathing the fillet. The Raspberry Pi had to handle capturing and inspecting the images with computer vision while also waiting for the message from the IRC5 controller stating that the pathing was complete. Initially, the team attempted to achieve this functionality by using non-blocking TCP socket receive calls, but non-blocking receive calls cannot be used for a server, only a client. The socket handler class was given its own thread and used two different queues to communicate with the main program thread. Queues were used as they are a safe data structure that can be used for messages passing between multiple threads of a program. One queue was for sending commands to the socket handler and another queue was used by the socket handler to pass messages to the main program thread. The commands that could be sent to the socket handler included sending or receiving a message, connecting to a socket, or closing the connection.

5) Software Flow of Execution: There are two main phases to the software's execution. The first phase consists of the

interactions with the user in order to prepare the blisk and system for the inspection. During this phase, the system must instruct the user on how to place the blisk on the turntable, wait for the user to do so, and then handle the positioning of the arm. Fig. 43 details the flowchart of the first phase of the blisk inspection system.

The second phase handles the actual inspection of the blisk. This occurs after the user has already set up the system for inspection and chosen to start the inspection process. During this phase the blisk inspection system must control inspecting every fillet on the blisk with both of the two ball gauge sizes, and then determine if each fillet passes or fails inspection. The software flowchart for the inspection phase of the software design can be seen in Fig. 44.

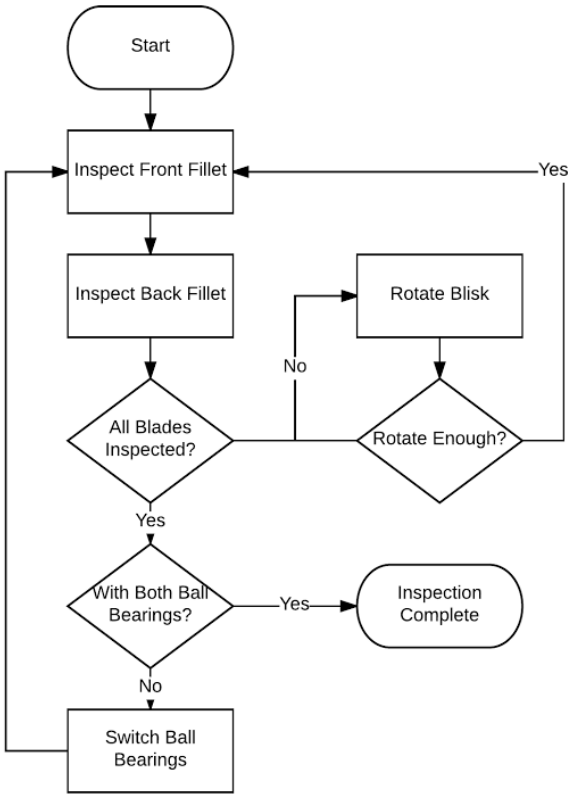


Fig. 44: Phase 2 Flow Chart

I. ABB Robot Pathing

1) Matching Physical and Simulated Environments: In order to begin pathing, the station was set up in RobotStudio. This involved adding the CAD models of the turntable and EOAT to RobotStudio. A work object was then created for the turntable. By making all targets relative to the work object, the team would later have the option of changing the turntable's location without needing to recreate robot targets. Once the environment was setup in RobotStudio, a series of points on the corners of the base plate were created to build an alignment path. By having the robot follow this path, the physical turntable was properly positioned. This was a crucial step because if the physical turntable was not in the same position as the simulated turntable the path would be incorrect.

Using the alignment path, the team discovered a number of inconsistencies between the CAD models and the physical components. The true turntable was about 0.4" taller than the CAD model, and the true EOAT was about 0.4" longer than the CAD model. The team believed these inconsistencies resulted from the designs being altered during manufacturing without updating the CAD models.

After adjusting the CAD models, the alignment path was run again. Doing this revealed that the EOAT had small manufacturing errors in the X- and Y-axes of the tool frame (see Appendix C). These errors caused the true tool center point (TCP) to be about 0.08" off from the expected TCP. This meant that when rotated about the expected TCP, the ball gauge spun in a 0.16" diameter circle. To adjust this, the team made modifications to the CAD model used in RobotStudio to reflect the real EOAT. By doing this, the error was brought down to approximately 0.02".

Another error affecting the pathing was that the true center of the turntable was 0.06" off in the Y direction and 0.2" off from the expected center. The team could not identify which components of the turntable caused this error, and so could not fix the issue. However, it was calculated that, at worst, this and the EOAT error would result in a total offset of about .12" from the expected position. As this was less than the 1/8" error tolerance built into the EOAT, it was determined that the setup was sufficient.

2) Generating the Paths: The movement of the ABB robot is composed of shorter paths corresponding to commands sent by the Raspberry Pi. A home position was first established that provided a constant starting point for the system. This was the only position that was the same for all four blisk stages, as all other paths were adjusted based on the exact size and shape of the stage.

An approach path was created which moved the ball gauge of the EOAT just outside the blades of the blisk. This position allows an inspector to easily line up the blisk such that the EOAT is centered between two blades.

To locate blade one, a path was created that moved the EOAT between the blades of the blisk. While following this path, the EOAT was also oriented so when the turntable was rotated, the LED arm would be the first component to contact the blisk. This is because the contact sensing wire runs down the LED arm to detect contact between the blisk and EOAT. With this positioning, the turntable can be rotated until contact with the LED arm is made, ensuring the real world and simulated systems are the same.

The next step was to path the front and back of the current blade. First, the tooling pathed in between the blades, rotating to maintain clearance as necessary. Once about 0.2" away from the expected location of the fillet, the Raspberry Pi began reading force sensing values. The tooling was moved slowly towards the fillet until the correct force was applied.

At this point, an offset was created using the RAPID PDist command. The PDistOn command calculates the distance between the robot's current location and an expected location. It saves this distance as an offset, and then applies this offset to all targets until the PDistOff command is called. This process was used to compensate for positioning and manufacturing errors present in the system. By doing this, the path could be dynamically updated to compensate for errors without having to constantly poll the force sensing signal from the Raspberry Pi.

The path along the fillet itself was split into two parts. One path had the EOAT start in the middle of the fillet and move up, while the other had the tooling start in the middle and move down. This method of pathing was chosen because it was safer than pathing the entire fillet in one pass. To path the fillet in one pass, the ball gauge would have to be initially placed on either the top or bottom edge of the fillet. Any error in the height accuracy of the system would then create a risk of the ball gauge slipping out of the fillet and past the hub of the blisk. If this occurred, the EOAT could be damaged by the robot trying to move the ball through the blisk.

The orientation of the EOAT while pathing the fillet varied based on which ball gauge was being used. When the maximum ball gauge was being used, the EOAT was oriented with the camera pointing down towards the table. When the minimum ball gauge was being used, the EOAT was oriented with the camera pointing towards the ceiling. This was done to protect the servo controlling the gauge selector, as it allowed the holding force for the gauge selector to be generated by the hard stops.

Once the front side of the fillet was followed, the offset was turned off, and the robot retracted the EOAT from between the blades. The same process was then repeated for the back side of the blade and the remaining blades of the blisk being inspected. For the tandem blisk in the LEAP series, the process was repeated for both stages. Because the turntable rotated a new blade into the same position each time, the same paths could be used for each blade.

J. High-Level Inspection Process

The process an end user must take in order to use the blisk inspection system was one of the first considerations made in designing the system. The first operational step is to power on the Blisk Inspection System. If at any point a user is unsure of how to use the system the instructions will always be linked at the bottom of the screen. The specific instructions for the Blisk Inspection System can be found in Appendix J. After powering the system on, the robot first moves the EOAT to a home position, allowing for a constant starting point. The user then chooses a blisk for inspection from the list of three possible blisks. The user can then place the blisk on the turntable, lock it down, and line up the first blade with a mark on the turntable. This is not expected to be precise, but to instead ensure that the blisk is roughly in the correct position.

During this process the robot will be clear from the workspace to ensure that neither the EOAT nor the blisk is damaged. Once the user indicates that the blisk is in place, the arm will move closer to the blisk, positioning the ball gauge just outside of its known radius. This step is for the user to ensure that the blisk is properly lined up, and make small adjustments to the blisk's position if necessary. Before proceeding, the application will require the user to place the contact lead on the blisk for contact sensing with the EOAT.

Upon choosing to proceed to the next step, the Blisk Inspection System will move the EOAT within the gap between blades on the blisk. The system will then slowly turn the turntable until contact with the blisk and the EOAT is made. The system will then halt turning the turntable, as the blisk has been successfully positioned. The application will then notify the user to remove the contact sensor lead from the blisk. The user can then choose to start the inspection of the blisk. The system will then autonomously inspect the fillets of the blisk.

Once the inspection is complete, a message saying whether the blisk passed or failed inspection will display. The application will list the file where the inspection results are located so the user can view the detailed inspection results.

IV. Results

A. Turntable

The Turntable was fully manufactured and operational. The turntable rotates each blade into position in three seconds. The arc length between blades was measured to be accurate to within .001". The turntable was measured to be .05" eccentric, well within the 1/8 " tolerance of the EOAT. The turntable was tested to work reliably for up to three hours. The blisk is easily placed and removed from the turntable and each action can be completed in 1 minute. The turntable is capable of holding each of the LEAP series of blisks correctly. The belt drive system is clean and reliable, and the belt can be changed by unscrewing a few set screws and lifting the motor.

B. End of Arm Tooling

The EOAT was fully manufactured and can be seen in Fig. 45. It was successfully able to position the ball gauge, LED and camera within the fillets of all three blisks as needed. However, slight errors in manufacturing resulted in the true TCP of the tooling being approximately 0.08" off of the expected TCP obtained from the CAD model. This resulted in difficulties pathing because the ball gauge would change in position by as much as 0.16" when it should have been rotating in place. The team worked to try to fix this, but could never determine the exact cause of this discrepancy.

The load cell was successful in detecting the compressive load on the EOAT, and resulted in consistent, repeatable paths. The spring steel segment was also successful, flexing approximately 1/8" under 1/8 lb of force as desired. This allowed the ball gauge to slide into the fillet consistently without damaging the blisk.

The spring section providing Z-axis compliance was less successful. Because of the weight of the EOAT itself, a large amount of friction was generated between the shoulder bolts and the load cell through holes. This meant stronger springs were needed to prevent the shoulder bolts from sticking. These

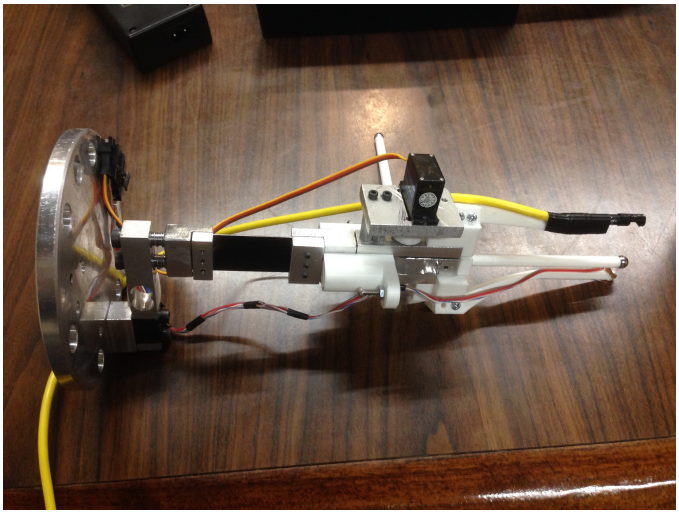


Fig. 45: The fully built EOAT.

stronger springs resulted in less overall compliance than was desired since the force being applied by the EOAT to the blisk remained the same.

Another issue with the Z-axis compliant section of this EOAT was that slight gaps in the clearance holes for the shoulder bolts resulted in significant sag in the EOAT of almost 0.5". This had a huge impact on the paths, causing issues for every phase of the inspection. The team was able to compensate for some of this by adjusting the paths based on the expected sag, however this was not ideal.

The gauge selector works, but was unreliable. This was primarily due to misalignment in the pieces which held the shaft of the gauge selector. Because the pieces were mounted to the EOAT with bolts and clearance holes, these pieces could shift and become out of alignment. This caused large amounts of friction to be generated on the shaft, preventing the servo from being able to rotate the gauge selector. Despite this, when the shafts were in alignment, the gauge selector worked as expected, rotating the desired ball gauge into position quickly and effectively.

The modular gauge mounts worked as desired, taking approximately 1 minute to change each pair. This assumes the proper hex key was easily accessible.

C. Computer Vision

Overall the computer vision was a success. The algorithm met the goal that when accurately configured the algorithm could correctly pass or fail an image based on the amount of light passing past the ball gauge. The computer vision was able to inspect an image in real time while the EOAT was pathing the blisk which was a stretch goal the team initially defined. Another stretch goal that was achieved was storing the inspection results in a CSV file that indicates the inspection result for an image and the corresponding position in the fillet.

Though the algorithm worked correctly, ultimately it was only configured for a few of the fillets. This was due to not all of the paths being finalized in the RAPID program. Ball gauge tracking was also added into the algorithm so that it could dynamically adjust for any changes in position of the ball gauge in the image. The ball gauge tracking was not incorporated into the final version of the algorithm as it cut the processing frames per second roughly in half. The ball gauge position also didn't change enough throughout a fillet inspection to require dynamic position updates. The final FPS the team achieved was 7.2 FPS, yielding a spatial resolution of 130 Captures per Inch (CPI) for the P02 Blisk.

Due to incomplete pathings of the fillet the team was unable to extensively test the reliability of the computer vision algorithm. When running the algorithm through a fillet there were multiple instances of incorrect failures of the images. This was not due to the computer vision algorithm but instead due to the ball gauge not being properly seated in the fillet throughout the path. This caused the LED light to show up as only on one side of the ball gauge, creating a failing image. If paths were able to consistently seat the ball gauge in the fillet then the computer vision algorithm would successfully identify whether an image should pass.

D. Overall System

The blisk inspection system was mostly successful. The GUI application walked the user through each step of the inspection process. By interacting with the GUI, the user could activate each phase of the inspection process.

Communication between the Raspberry Pi and the IRC5 was successfully implemented. This allowed communication in both directions as necessary for each phase of the inspection.

Zeroing blade one using the contact sensing feature of the EOAT worked reliably. This ensured the blisk was properly aligned for each inspection. The system was also able to locate the fillet reliably using the force sensing and the eccentricity of the turntable was always accounted for.

Pathing the tooling along a fillet correctly was the largest unsuccessful part of the project, but because it was essential for a working system it caused the whole system to be unable to inspect a blisk. While multiple successful paths were created in RobotStudio, fine-tuning these paths to be within the tolerances needed to capture good images was a slow process. In addition to this, the manufacturing errors in the EOAT discussed in Section IV-B made generating precise paths even more difficult. As a result, only the paths needed for P01 and P02 using the maximum ball gauges were fine-tuned. The paths for P01 and P02 using the minimum ball gauges were not fine-tuned, and neither were the paths for G02.

By timing the inspection process for a few blades from the completed paths, an estimated total inspection time of one hour and forty five minutes was calculated. This is much longer than the desired one hour inspection time. However, the team believes this could be improved by speeding up the robot once the system is more reliable.

V. Conclusion

Overall, all individual components of the system created by the team work in their current form, though many of them

could be improved. In addition to this, the team is confident that the system could reach a reliable state with more work. The team believes that a Blisk Inspection System is feasible for General Electric Aviation for their LEAP series of blisks.

A. Social Implications of a Robotic Inspection System

It is a common concern with industrial systems that the introduction of autonomous systems results in the loss of human jobs. The Blisk Inspection System would complete the majority of a fillet inspection task autonomously. However, the team believes that this is not synonymous with the loss of human jobs.

The Blisk Inspection System requires an operator. If there are any faults in the system, it is also necessary for the operator to do a manual inspection. Furthermore, the Quality Assurance inspectors at GE Aviation involve many checks beyond the fillet inspection. The Blisk Inspection System was designed as a tool for an operator to use so they have more time for value-added tasks. This would increase the value of a blisk without significantly increasing the labor costs associated with it.

B. Future Work

There were a number of areas in which this project could be improved or expanded by future work. This section describes these areas broken down by topic.

1) Turntable: The primary area in which the turntable could be improved is in its lack of positional feedback. Currently, the stepper motor spins the blisk without feedback. Though this was not an issue with this implementation of the system, a finalized implementation should have positional feedback to account for errors such as the stepper motor or stepper circuitry malfunctioning. Depending on the implementation, this may also remove the need for the grounding clip, removing a setup step from the inspection process.

2) End of Arm Tooling: There were a number of areas in which the EOAT could be improved by future work. The Z-axis compliance section needs significant improvement. It did not generate as much compliance as needed, and it also resulted in significant sag from gravity that caused pathing issues. As such, work would need to be done to solve both of these issues. One potential solution discussed by the team would be to use Delrin such that the contact points would be Delrin against Delrin or Delrin against Steel. This would result in a decrease in the coefficient of friction. Another design could relocate the Z-axis compliance further out on the EOAT. This would reduce the weight applying pressure to the compliant section, reducing the friction generated. It would also reduce the sag at the ball gauge because there would be less distance for the error to translate over.

Another area for improvement would be implementing a locking mechanism for the gauge selector. Though the team's implementation involving hard stops worked, it is not as reliable as a locking mechanism would be. This mechanism would also reduce the number of paths needed to be built in RobotStudio. In addition to this, adding feedback which could verify that the swap was successful would increase the reliability of the system.

The last major area of future work would be to redesign and re-implement the manufacturing and assembly process for the EOAT. In particular, this would include using tighter tolerances, and using pins and precision holes to align components instead of screws and clearance holes. This would help to improve the error in the TCP discussed in Appendix C. This would also help remove the potential for shifting parts to generate heavy friction in the gauge selector as discussed in section III-F1.

3) Computer Vision: There are multiple ways the computer vision could be improved for future work. One of the first ways this could be accomplished is by running the computer vision algorithm on a more powerful processor than the Raspberry Pi. The processing power greatly limited the speed the ABB Robot could path along the blisk fillets. Optimizing the algorithm would also allow for a higher processing FPS. One other possibility is to use machine learning to identify if an image passes inspection.

Additionally the vision from the camera could be used to determine if the ball gauge is seated in the fillet. If the camera could detect when the ball gauge was not in the fillet then this information could be used as feedback to move the arm forward more.

Another way to potentially improve the computer vision is to not have it run in real time when the EOAT is pathing the fillet. Instead the images could be saved and the inspection could happen on a separate thread while the EOAT is completing the pathing of the fillet and positioning to path the next fillet.

4) Final Thoughts: A robotic blisk fillet inspection solution is feasible for GE Aviation. The team hopes that this project will continue, as it is a difficult challenge, and there are many improvements to be made to the system. If a team were to start with the knowledge base that has been created in this document, it is likely that they would be much more successful in completing and implementing a Blisk Inspection System.

APPENDIX A
BLADE NUMBERING SCHEME

The blades of a blisk are referred to by number, starting with 1 and continuing to the max number of blades. On every blisk, there is a mark on one side of the hub which indicates which blade is blade 1. This mark is not standard across all blisks.

When the mark is face up, the direction of counting is such that the leading edge of each blade is met before its trailing edge. On the blisks the team currently has, this means counterclockwise rotation. When the mark is face down, the direction of counting is opposite, such that the trailing edge of each blade is met before the leading edge.

APPENDIX B
RASPBERRY PI PINOUT

Function	GPIO Pin
Stepper Motor Step	20
Stepper Motor Direction	16
Servo PWM Signal	4
Servo Power Control	17
Force Sensor Clock	23
Force Sensor Data	24
Circuit Completer Input	5
LED Signal	22

TABLE III: Raspberry Pi Pinout

APPENDIX C
COORDINATE SYSTEMS

There are a number of different coordinate frames used in this system. What follows is a description of each, including a few ways each coordinate frame is used.

A. The World Frame

The world frame is the default coordinate frame for all projects. It provides a base reference around which all other system elements and coordinate frames can be positioned. Moving the world frame moves every other coordinate system, target, and workobject by an equal amount. In this system, the world frame is located directly beneath the robot in the center of it's base. The Z-axis points vertically, the X-axis points out the front of the robot, and the Y-axis completes the right-handed coordinate system. The world frame is a default defined by RobotStudio.

B. The Base Frame

The base frame defines the robot's location in the world frame, and provides a way to define targets and paths relative to the robot. In this way, moving the base frame (which is equivalent to moving the robot) moves the items defined relative to that frame. This frame is a default defined by RobotStudio.

The base frame is located directly beneath the robot in the center of its base. The Z-axis points vertically through the robot, the X-axis points out the front of the robot, and the Y-axis completes the right-handed coordinate system. This is the same position as the world frame, and so in this project the base and world coordinate frames were interchangeable.

C. The Turntable Workobject

A coordinate frame was created for the turntable such that robot targets and and paths could be defined relative to it. By doing this, the turntable could be moved around without losing all of the work done creating paths along the fillets.

D. The Tool Center Point (TCP)

The tool center point is a coordinate frame which defines the working point of an end of arm tooling. Tool0, a default TCP defined by RobotStudio, defines the mounting point for user-built end of arm toolings. Both the tooling built by the previous team and the tooling built in this implementation had similar TCP's. In both cases, the TCP defined the tip of the ball gauges, with the Z-axis extending out of the end of the ball, the Y-axis extending through the camera, and the X-axis completing the the right handed coordinate frame. These TCP's can be seen in Fig. 46 and Fig. 4.

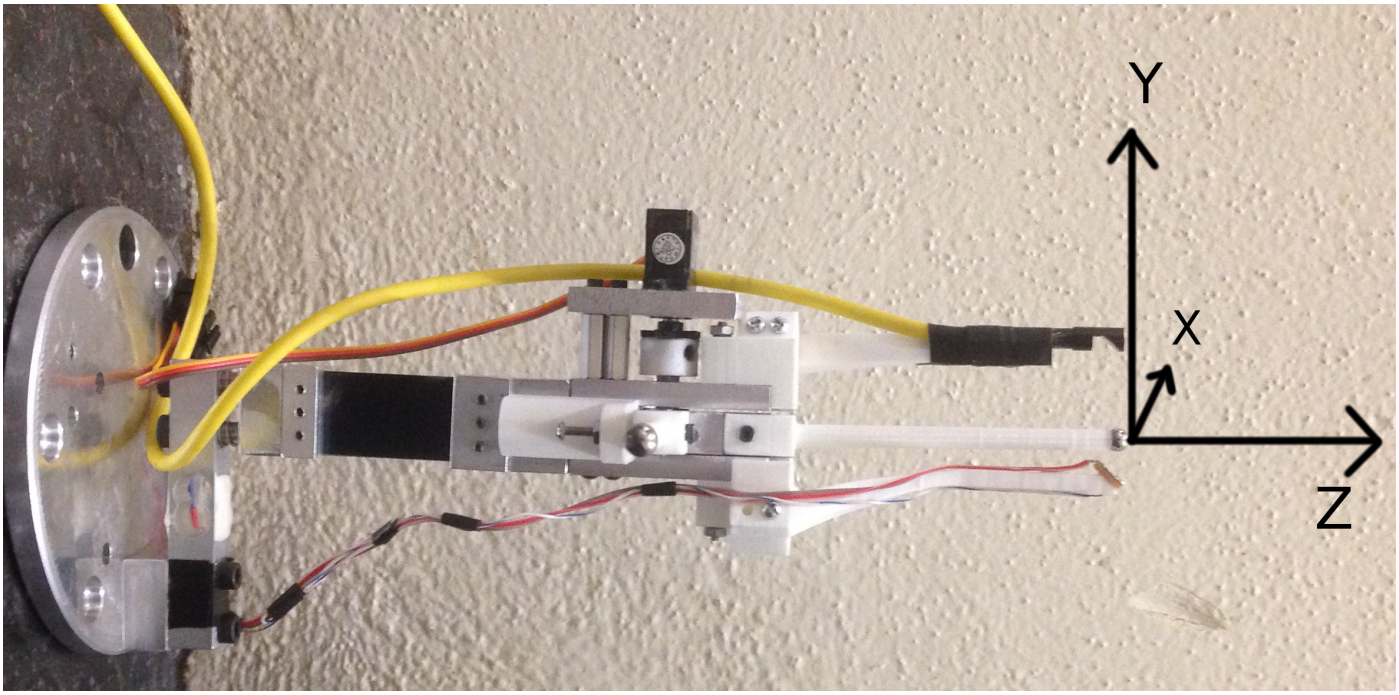


Fig. 46: The EOAT with TCP shown.

APPENDIX D
BOUNDARY CLASS DIAGRAMS

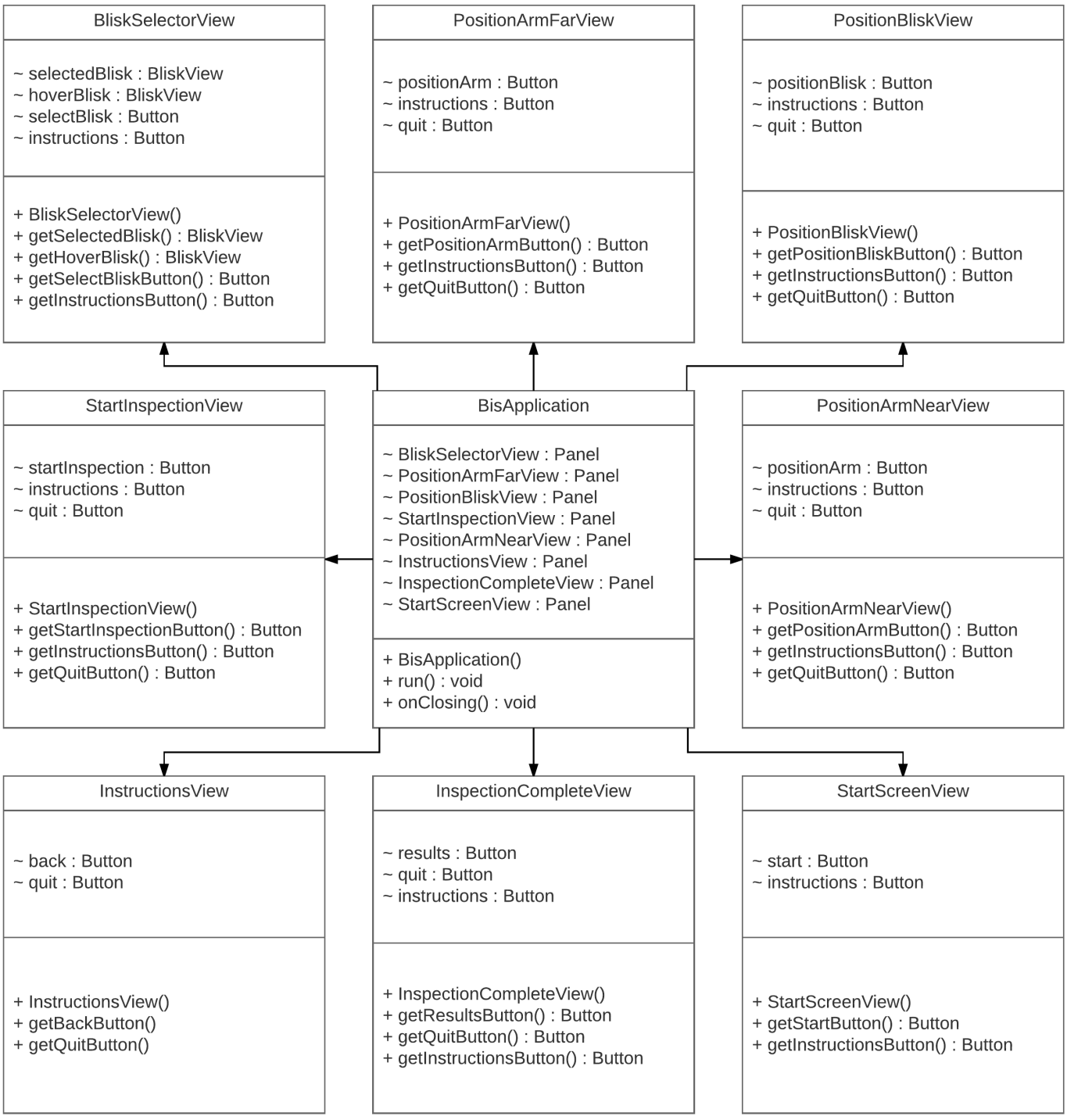


Fig. 47: Boundary Class Diagram

APPENDIX E
CONTROLLER CLASS DIAGRAMS

Fig. 48: Controller Class Diagram

APPENDIX F
ENTITY CLASS DIAGRAMS

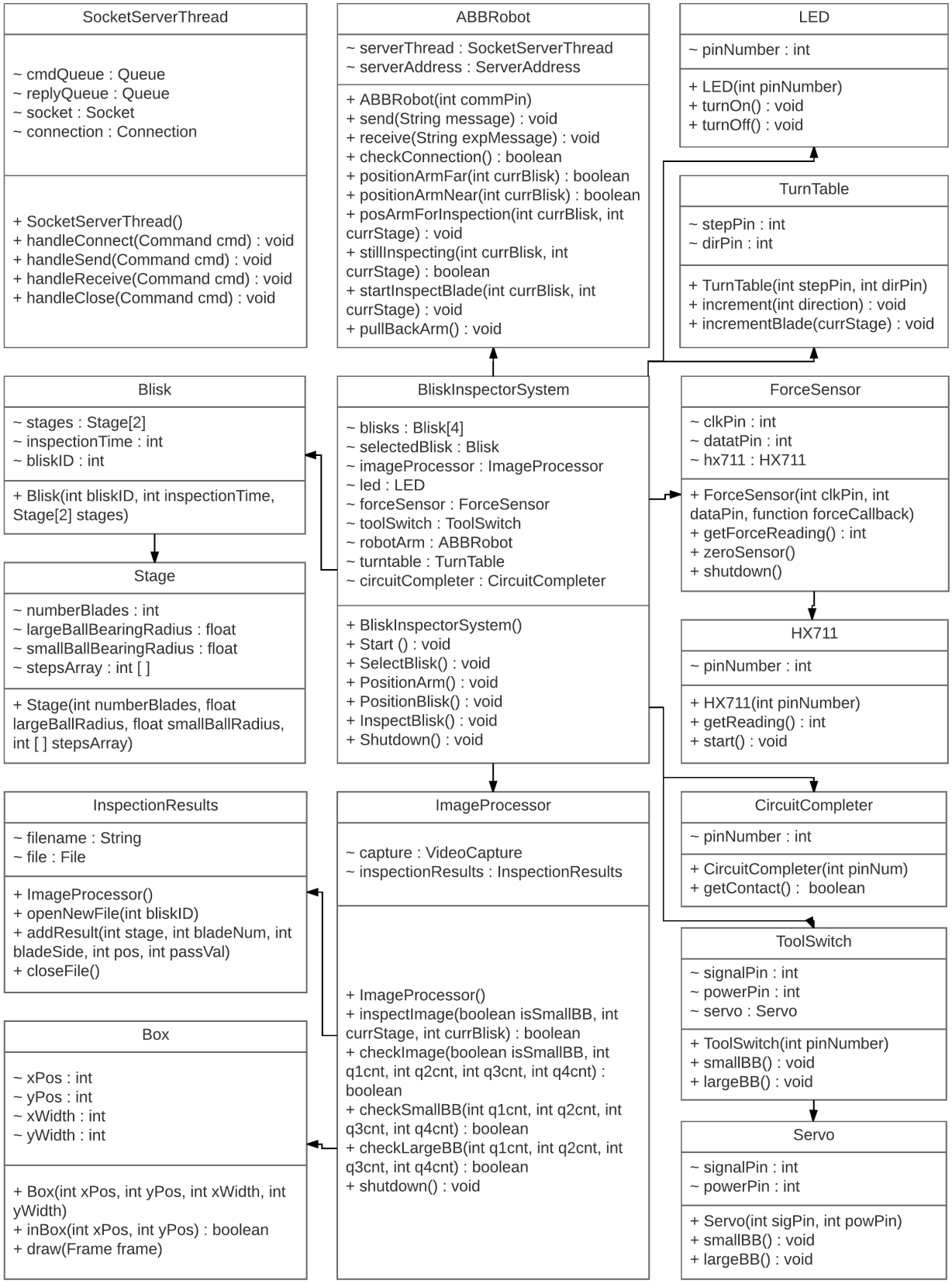


Fig. 49: Entity Class Diagram

APPENDIX G
INITIAL SOFTWARE DESIGN USER STORIES

Nomenclature:

Inspector: The person running the blisk inspection.

Blisk: The Bladed Disk that can be multi-staged.

Blisk Inspection System: The system that is responsible for inspecting the blisk.

Blisk Inspection: The process of inspecting just the radius of the root fillets of the blisk.

Blisk Inspection Application: The software application that controls the blisk inspection.

Blisk Inspection User Stories

I want to go to the blisk inspection opening screen so that I can choose a blisk to inspect.

Use Case: Display Blisk Inspection Options

I want to be able to read instructions on how to operate the blisk inspection system.

Use Case: Display Inspection Instructions

I want to be able to able to select a blisk from a menu to inspect.

Use Case: Select Blisk

I want to be able to position to ABB Robot arm into place for the inspection.

Use Case: Position Arm

I want to be able to have the blisk on the turntable to be correctly positioned for inspection.

Use Case: Position Blisk

I want to be able to start the inspection of the blisk.

Use Case: Start Blisk Inspection

I want to be able to know how much time is left in the inspection process of the blisk.

Use Case: Display Remaining Inspection Time

I want to be able to see a progress bar of the progress of the blisk inspection.

Use Case: Display Inspection Progress

I want to be able to see when the blisk inspection has completed.

Use Case: Display Inspection Completion

I want to be able to see if the blisk inspection passed or failed.

Use Case: Display Inspection Result

I want to be able to see a detailed report of the blisk inspection results.

Use Case: Display Inspection Results Report

I wish to be able to quit the inspection and return to the opening screen

Use Case: Quit Inspection to Return to Opening Screen

APPENDIX H
INITIAL SOFTWARE DESIGN USE CASES

Nomenclature:

Inspector: The person running the blisk inspection.

Blisk: The Bladed Disk that can be multi-staged.

Blisk Inspection System: The system that is responsible for inspecting the blisk.

Blisk Inspection: The process of inspecting just the radius of the root fillets of the blisk.

Blisk Inspection Application: The software application that controls the blisk inspection.

Blisk Inspection Use Cases

Use Case:	Display Blisk Inspection Options
Participating Actor:	Initiated by Inspector
Entry Condition:	The Inspector opens the Blisk Inspection Application
Exit Condition:	The Blisk Inspection Application shows the different blisk options for the Inspector to choose from
Flow of Events:	1. Inspector opens the Blisk Inspection Application. 2. The Blisk Inspection Application shows the different blisk options.

Use Case:	Display Inspection Instructions
Participating Actor:	Initiated by Inspector
Entry Condition:	The Blisk Inspection Application is open.
Exit Condition:	The instructions on how to use the blisk inspection system are now being displayed.
Flow of Events:	1. Inspector selects to view the instructions. 2. The Blisk Inspection Application shows the instructions on how to use the system.

Use Case:	Select Blisk
Participating Actor:	Initiated by inspector
Entry Condition:	The Blisk Inspection Application is at the opening screen with the blisk options menu
Exit Condition:	The Blisk Inspection Application now has the blisk chosen by the Inspector selected for inspection.
Flow of Events:	1. The Inspector selects a blisk for inspection 2. The Blisk Inspection System selects the blisk for inspection

Use Case:	Position Arm
Participating Actor:	Initiated by inspector
Entry Condition:	The Blisk Inspection System already has a blisk selected for inspection and is at the position arm screen
Exit Condition:	The Blisk Inspection System has positioned the arm correctly for the blisk being inspected.
Flow of Events:	1. The Inspector chooses to position the arm for the selected blisk 2. The Blisk Inspection system positions the arm in the correct location for inspection

Use Case:	Position Blisk
Participating Actor:	Initiated by Inspector
Entry Condition:	The arm has been positioned, a blisk has been selected, and the Blisk Inspection Application is on the Position Blisk Screen
Exit Condition:	The Blisk Inspection System has correctly positioned the blisk on the turntable
Flow of Events:	1. The Inspector chooses to position the blisk on the turntable 2. The Blisk Inspection system positions the blisk on the turntable in the correct location for inspection

Use Case:	Start Blisk Inspection
Participating Actor:	Initiated by Inspector
Entry Condition:	A blisk has been selected, the arm has been positioned correctly for the blisk, and the blisk has been positioned on the turntable
Exit Condition:	The Blisk Inspection System has started the blisk inspection
Flow of Events:	1. The Inspector has selected to start the blisk inspection 2. The Blisk Inspection System has started inspecting the blisk

Use Case:	Display Remaining Inspection Time
Participating Actor:	Initiated by Inspector
Entry Condition:	The Blisk Inspection System is in the process of inspecting the blisk
Exit Condition:	The Blisk Inspection Application shows the remaining time for the blisk inspection
Flow of Events:	1. The Inspector has started the blisk inspection 2. The Blisk Inspection Application shows the remaining inspection time for the blisk

Use Case:	Display Inspection Progress
Participating Actor:	Initiated by Inspector
Entry Condition:	The Blisk Inspection System is in the process of inspecting the blisk
Exit Condition:	The Blisk Inspection Application shows the progress of the blisk inspection
Flow of Events:	1. The Inspector has started the blisk inspection 2. The Blisk Inspection Application shows the blisk inspection progress

Use Case:	Display Inspection Completion
Participating Actor:	Initiated by Inspector
Entry Condition:	The Blisk Inspection System completes the inspection of the blisk
Exit Condition:	The Blisk Inspection Application shows that the blisk inspection has completed
Flow of Events:	1. The Inspector has started the blisk inspection 2. The Blisk Inspection Application shows that the blisk inspection has completed

Use Case:	Display Inspection Result
Participating Actor:	Initiated by Inspector
Entry Condition:	The Blisk Inspection system has completed inspecting the blisk
Exit Condition:	The Blisk Inspection system shows if the blisk inspection passed or failed
Flow of Events:	1. The Blisk Inspection system completes inspecting the blisk 2. The Blisk Inspection Application shows if the blisk inspection passed or failed

Use Case:	Display Inspection Results Report
Participating Actor:	Initiated by Inspector
Entry Condition:	The Blisk Inspection system has completed inspecting the blisk
Exit Condition:	The Blisk Inspection Application displays the blisk inspection results report
Flow of Events:	1. The Inspector selects to display the blisk inspection results report 2. The Blisk Inspection Application displays the results report of the blisk inspection

Use Case:	Quit Inspection to Return to Opening Screen
Participating Actor:	Initiated by Inspector
Entry Condition:	The Blisk Inspection System is not on the opening screen
Exit Condition:	The Blisk Inspection Application quits the inspection and returns to the Opening Screen
Flow of Events:	1. The Inspector selects to quit the current inspection to return to the opening screen 2. The Blisk Inspection Application displays quits the current inspection and returns to the Opening Screen

Use Case:	Leave Instructions Page to Return Back to Last Page
Participating Actor:	Initiated by Inspector
Entry Condition:	The Blisk Inspection System is on the instructions page
Exit Condition:	The Blisk Inspection Application returns to the previous page
Flow of Events:	1. The Inspector selects to return back to the last page from the instructions page 2. The Blisk Inspection Application displays the last page that was open before the instructions

APPENDIX I
BLISK APPLICATION SCREENSHOTS

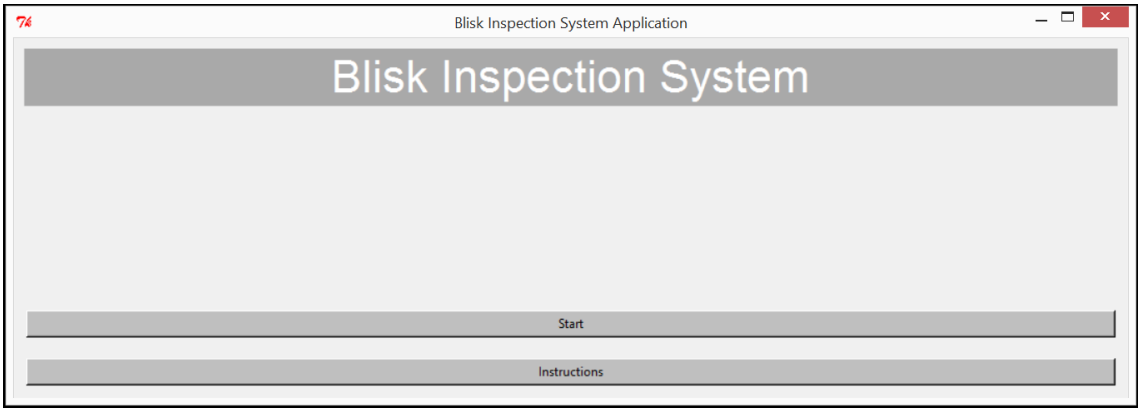


Fig. 50: Blisk Application Start Screen

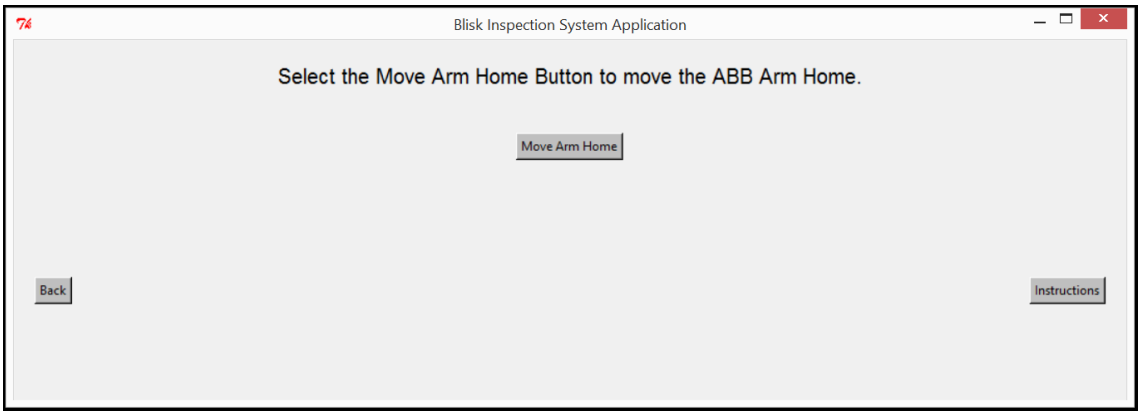


Fig. 51: Blisk Application Home Arm Screen

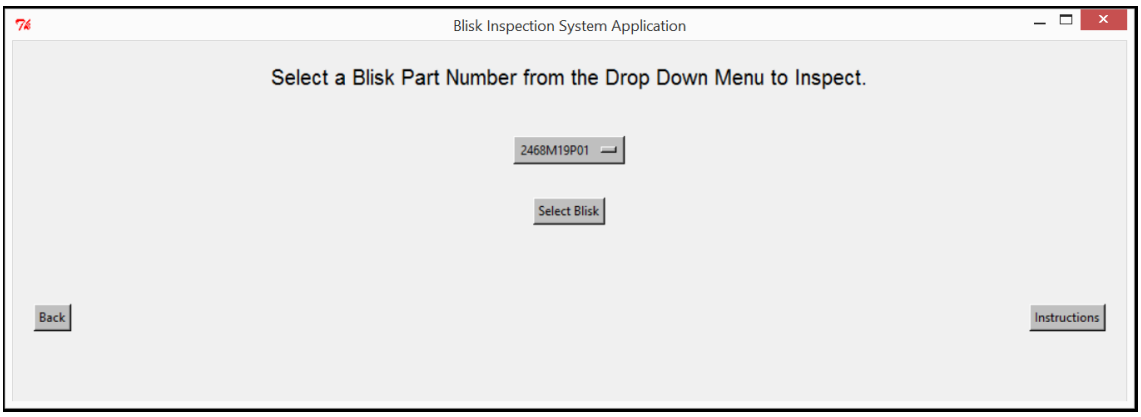


Fig. 52: Blisk Application Select Blisk Screen

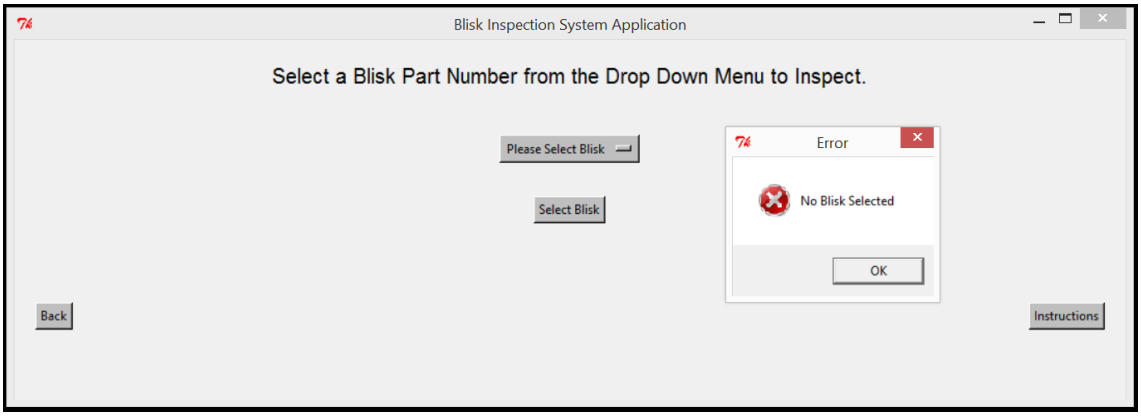


Fig. 53: Blisk Application No Blisk Selected Screen

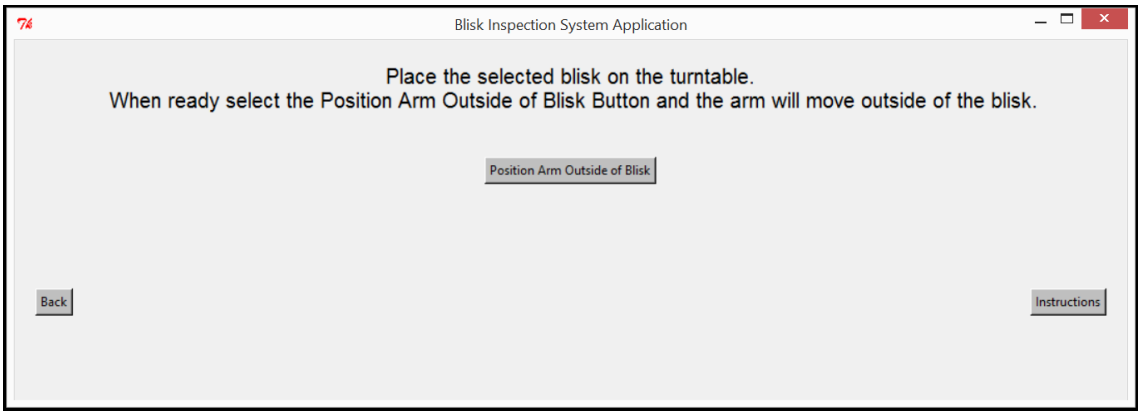


Fig. 54: Blisk Application Position Arm Outside of Blisk Screen

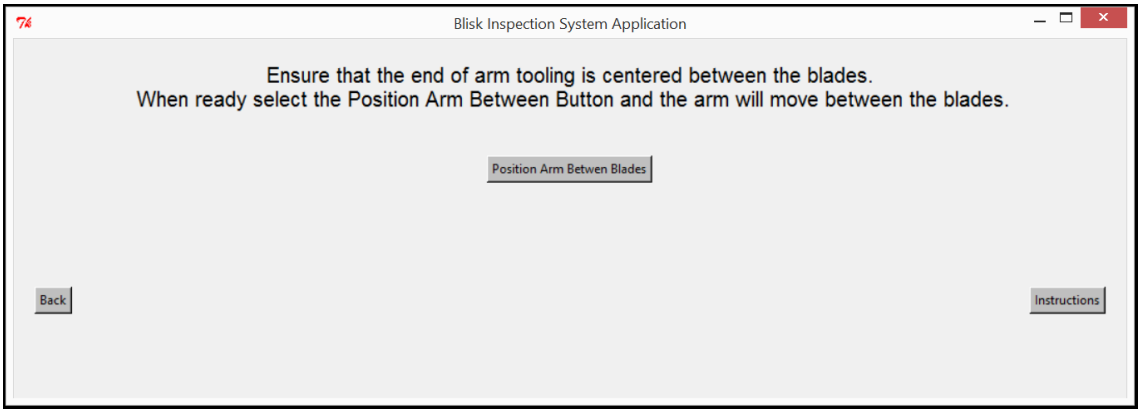


Fig. 55: Blisk Application Position Arm Between Blades Screen

Fig. 56: Blisk Application Turn Blisk Screen

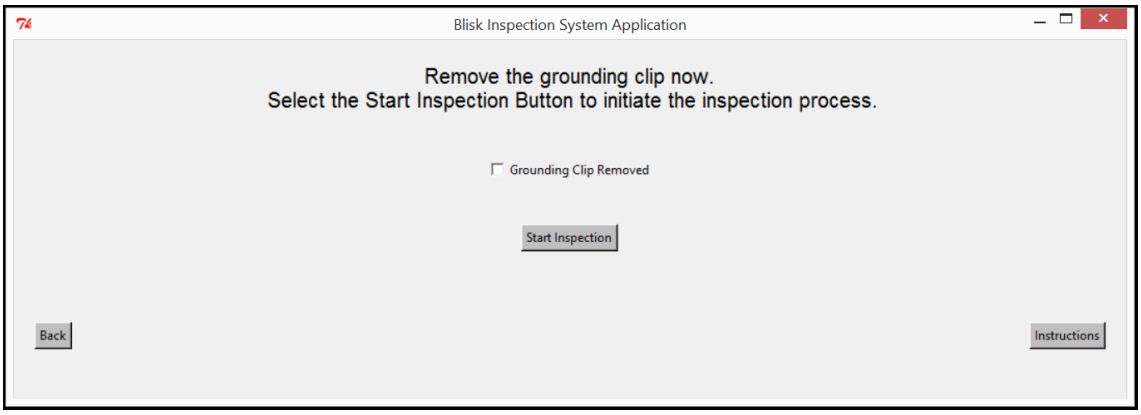


Fig. 57: Blisk Application Start Inspection Screen

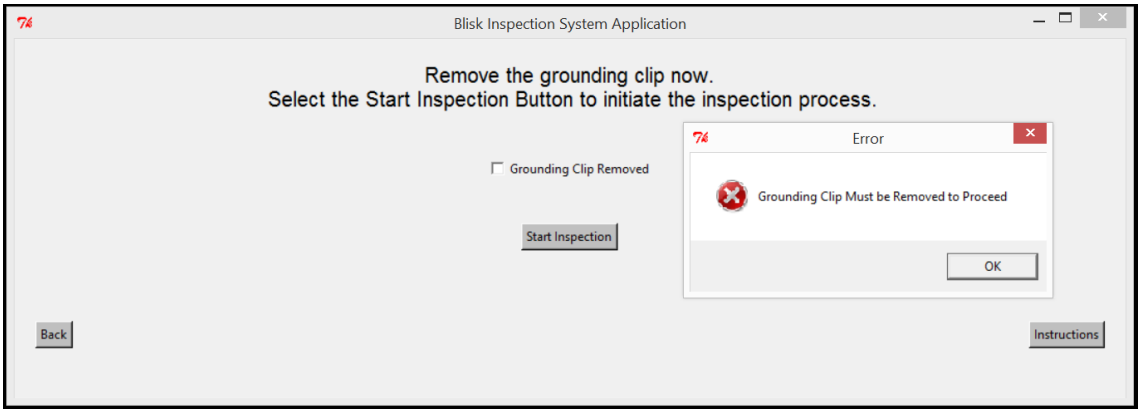


Fig. 58: Blisk Application Grounding Clip not Removed Error Screen

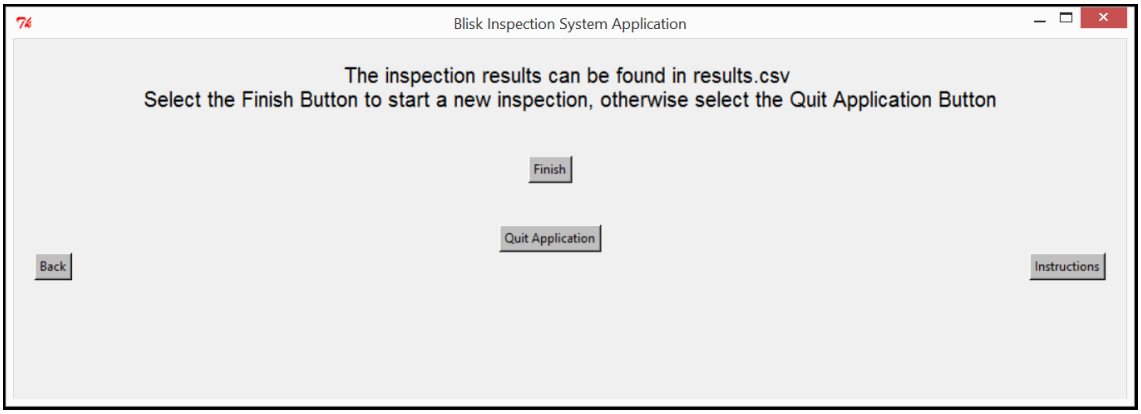


Fig. 59: Blisk Application Inspection Complete Screen

APPENDIX J
BLISK INSPECTION APPLICATION INSTRUCTIONS

This application can be used to inspect the following GE Bladed Disks (Blisks): 2468M19P01, 2468M17P02, and 2468M18G02. Ensure that the IRC5 Controller for the ABB Robotic Arm IRB1600T is powered on. The Raspberry Pi must be on and the wiring must be set up according to the wiring diagram. The End of Arm Tooling (EOAT) must be mounted to the ABB Robot with the correct ball gauge attached. When viewed from the camera side (the camera is on top, looking down onto the EOAT) the small ball gauge is the left arm. Once everything is connected, start up the blisk inspection application. Click the button to home the arm and select a blisk to inspect. Place the blisk on the turntable, but do not tighten it down. Select the button in the application to position the tooling just outside of the blisk. Ensure that the arm is lined up between the blades, with the blade to its right being the first to be inspected. Ensure that the grounding clip is on the blisk before turning the turntable with the application, and remove it afterwards. After this the inspection process will run, and when it is complete the application will identify where the results CSV file can be found.

APPENDIX K
SPATIAL RESOLUTION TABLE

Blisk Model Number	Pathing Distance (in)	Spatial Res. for 15 min. pathing (CPI)	Spatial Res. for 20 min. pathing	Spatial Resolution for 30 min. pathing	Spatial Resolution for 40 min. pathing
P02	148.75	36.31	47.93	72.62	96.15
P01	133	40.60	53.61	81.23	107.29
G01 (Doubled time for tandem blisk)	268.75	40.19	53.04	80.37	106.10

TABLE IV: Spatial Resolution for Various Blisks and Pathing Times

APPENDIX L
SAMPLE RESULTS FILE

Stage	Blade	Blade Side	Ball Bearing	Position from Bottom of Fillet (mm)	Result
0	1	CONCAVE	LARGE	22	TRUE
0	1	CONCAVE	LARGE	23	TRUE
0	1	CONCAVE	LARGE	23	FALSE
0	1	CONCAVE	LARGE	24	FALSE
0	1	CONCAVE	LARGE	25	TRUE
0	1	CONCAVE	LARGE	26	FALSE
0	1	CONCAVE	LARGE	26	TRUE
0	1	CONCAVE	LARGE	27	TRUE
0	1	CONCAVE	LARGE	28	TRUE
0	1	CONCAVE	LARGE	29	TRUE
0	1	CONCAVE	LARGE	30	TRUE
0	1	CONCAVE	LARGE	31	FALSE
0	1	CONCAVE	LARGE	32	TRUE
0	1	CONCAVE	LARGE	33	TRUE
0	1	CONCAVE	LARGE	34	TRUE
0	1	CONCAVE	LARGE	35	TRUE
0	1	CONCAVE	LARGE	36	TRUE
0	1	CONCAVE	LARGE	37	TRUE
0	1	CONCAVE	LARGE	38	TRUE
0	1	CONCAVE	LARGE	39	FALSE
0	1	CONCAVE	LARGE	39	FALSE
0	1	CONCAVE	LARGE	40	TRUE

Fig. 60: Sample Results CSV File from a test run. The file is named BliskP02_April_13_2017_1253PM which indicates that blisk P02 was inspected on April 13, 2007 at 12:53pm.

APPENDIX M
RASPBERRY PI AND IRC5 COMMUNICATION DIAGRAM

Raspberry Pi	Direction	IRC5 Controller
Sends Message	“HOME” →	Moves Home
Asks User to Place Blisk	“HOME” ←	Is Home
Sends Message	“FAR_STAGE” →	Moves Outside Blisk Radius
Ask User to Line up Blisk	“FAR_STAGE” ←	Is Far
Sends Message	“NEAR_STAGE” →	Moves between Blades
Turns Turntable until Contact	“NEAR_STAGE” ←	Is between Blades
Sends Message	"RETRACT_NEAR_P02" →	Moves out from Blades
Asks User to start Inspection	"RETRACT_NEAR_P02" ←	Is out from Blades
Sends Message to Start Inspection	“PREP_PATH” →	Moves Between Blades and Near Center of Fillet
Moves to Force Sensing	“PREP_PATH” ←	Is Ready to Begin Force Sensing
Sends Message to Keep Moving EOAT Forward	“MOVE_EOAT” →	Moves EOAT Forward .1 mm
Checks Force Sensing Values	“MOVE_EOAT” ←	Completed the Move
Sends Message to Inspect	"INSP_PATH" →	Sets Offset and Begins Path
Finish Inspection Process	"INSP_PATH" ←	Full Path Inspection Done
Start the Computer Vision	"START_PATH_UP" ←	Starting the Actual Inspection in the Fillet Moving Up from Center
Start the Computer Vision	"START_PATH_DOWS" ←	Starting the Actual Inspection in the Fillet Moving Down from Center
Stop the Computer Vision	“PAUSE_PATH” ←	Pause the Video as the Ball is out of the Fillet

APPENDIX N
WIRING GUIDE

In order to properly wire the system together, there are a few important steps to follow:

1) First, set up the Raspberry Pi using a mouse, keyboard, and HDMI monitor. Connect the ribbon cable from the Raspberry Pi to the ribbon cable connector on the auxiliary electrical board. Close the Raspberry Pi enclosure to protect the circuitry.
2) Next, connect the 12V supply to the auxiliary electrical board using the barrel connector, but do not connect it to mains power.
3) Connect the cooling fan to the fan connection pins on the auxiliary electrical board.
4) Close the enclosure for the auxiliary electrical board, ensuring that all connectors are outside the box and accessible.
5) Connect the stepper motor to the 4-pin connector with red, black, yellow, and white wires.
6) Connect the load cell to the 4-pin connector with red, black, green, and white wires.
7) Connect the servo to the 3-pin connector with red, black, and yellow wires.
8) Connect the LED and contact sensor (one connector) to the auxiliary electrical board connector with red, black, and white wires.
9) Connect the USB camera from the EOAT to the Raspberry Pi.
10) Once all connections are made, supply electrical power to the external 12V power supply.

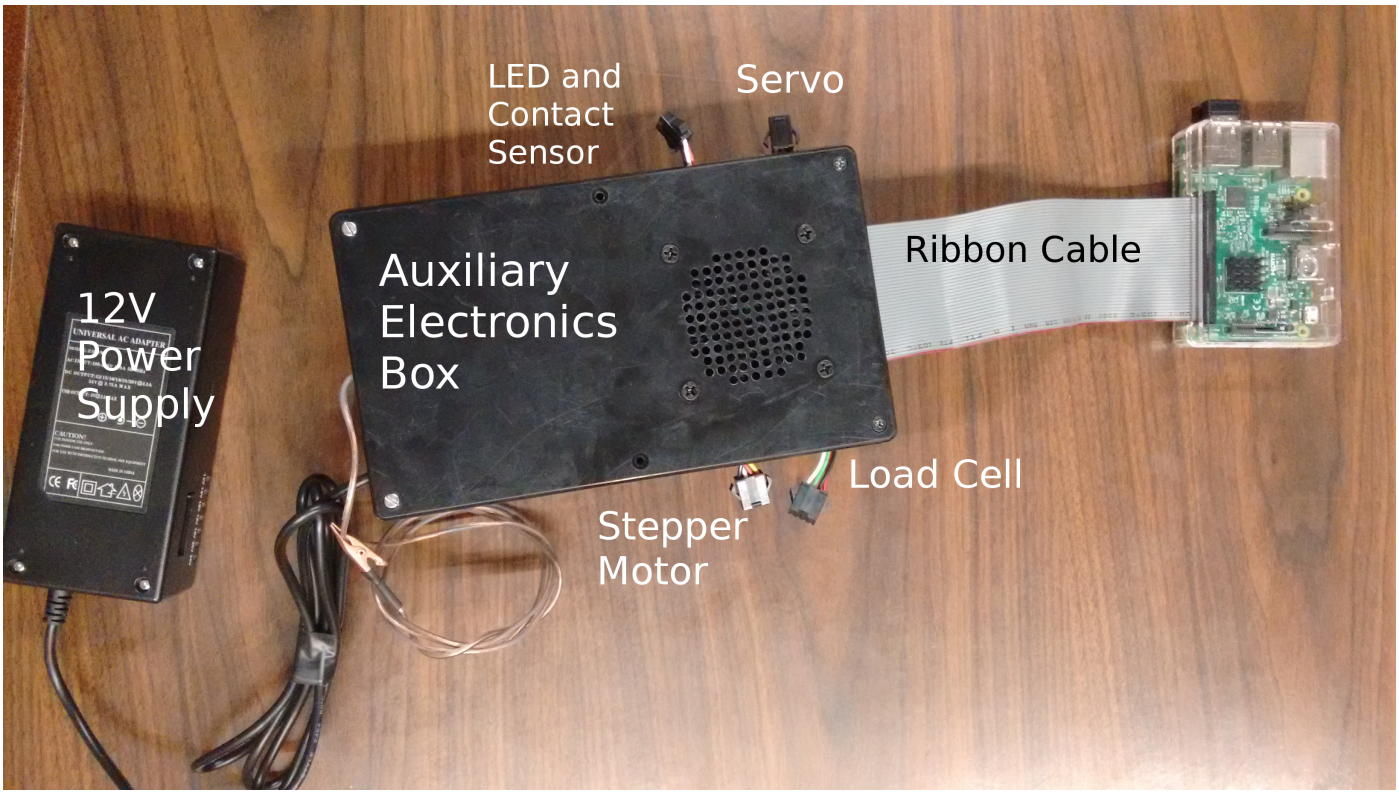


Fig. 61: Wiring Labeled

ACKNOWLEDGMENTS

The team would like to thank Professor Craig Putnam for his continuous devotion to the project. The team would also like to thank Professor Ken Stafford and Professor Michael Gennert for their support over the past year. The authors would also like to thank our primary contact at GE John Graham, as well as all of the other helpful voices at GE Aviation Hooksett.

Additionally the team would like to thank the lab monitors at both Washburn Labs and Higgins Labs for their help in manufacturing various parts for the system. The team would also like to thank the robotics lab monitors including Joe St. Germain and Kevin Harrington. Finally the team would like to thank Meghan Broughton, Jackie Campbell, and Gina Rios for their continuous emotional support.

REFERENCES

[1] HSL and HSV Wikipedia. Wikimedia Foundation, 16 Mar. 2017. Web. 28 Mar. 2017.
[2] RAPID Reference Manual. ABB Flexible Automation. Web. 25 Mar, 2017
[3] A4988 Stepper Motor Driver Datasheet. Allegro Microsystems. Web. 18 Apr, 2017
[4] AD623 Instrumentation Amplifier Datasheet. Analog Devices. Web. 25 Oct, 2016
[5] Synchronous Belt Data Summary. Gates a Tomkins Company. Web. 20 Jan, 2017