Design of a One-Eyed Face Tracking Robot to Facilitate Human-Robot Interaction

Jonathan Melkun, '23

Submitted to the
Department of Mechanical and Aerospace Engineering
Princeton University
in partial fulfillment of the requirements of
Undergraduate Independent Work.

Final Report

April 26, 2023

Prof. Daniel Nosenchuck,
Alexander Gaillard
Prof. Anirudha Majumdar
MAE 442
77 pages
File Copy

This thesis represents my own work in accordance with University regulations.

Abstract

Eye contact is one of the most important parts of communication between humans. It shows that a person is listening and paying attention to what is being said. Modern robotic assistants like Amazon Echo or Google Home seem to be missing this crucial part, leaving the user with nothing to focus on when they are speaking to the robot. This project seeks to mitigate this issue with the design of Mr. I, a one-eyed face tracking robot that a user would be able to maintain eye contact with. The goal is to make it easier for a person to have a conversation with the robot, facilitating human-robot interaction. The project was successful in that regard as the robot accurately tracks faces and peer feedback has been largely positive.

Acknowledgements

First and foremost, thank you to my advisor Prof. Daniel Nosenchuck for lending me your guidance both throughout this project and for the last four years. From being my freshman year advisor to thesis advisor, we have been through a lot together and you helped me realize what I am truly capable of. Thank you on taking me on when no one else would, even when you were fully booked.

A big thank you to my second advisor Alexander Gaillard. From doing Zoom labs on a laptop in a cart to working on a real-life video game prop to now a face tracking robot, you have helped me grow immeasurably as an engineer at Princeton and I will be forever thankful for having you as a mentor and as a friend.

I'd also like to thank Glenn Northey for helping me around the machine shop with random questions and giving me the resources and the space to succeed.

Thank you to Jonathan Prevost for helping me out with the microelectronics. The project would have gone up in flames without your help.

I would also like to thank SEAS for the generous funding that made this project possible. I would have never been able to afford this kind of project on my own, so I am truly grateful for the opportunity given to me.

Thank you to my friends at Princeton, from the uncountable amount of hours eating meals together at Colo to getting up to late night shenanigans, it's been a wild ride. There were many ups and downs along these four years, but you guys made it all worth it in the end and will always hold a special place in my heart.

And lastly thank you to all the people who provide information on the Internet publicly and for free. This project would not have been possible without the help of countless posts on websites like Arduino forums, Stack Overflow, and YouTube. The internet today is a vast collection of knowledge in almost any subject you can dream of. The resources that are available to me today are incredible compared to just a couple years ago, and I try not to take it for granted. I hope that one day I will be able to return the favor.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Market

The global household robots market was valued at $8.03 billion in 2021 and is expected to grow to $32.9 billion by 2030 [1]. Major players in the market like Amazon Echo and Google Home all allow people to speak to the device just like they would to another person and command it to do simple tasks around the house. The major draw to these robots is automating repetitive tasks like scheduling reminders or household chores. Robots do not get bored of these tasks and also do not require wages, which reduces the overhead expense [1]. This is an example of human-robot interaction, which describes a class of research that studies how robots can better the lives of humans through means of collaboration or companionship [2]. However, natural human-robot interaction is a difficult challenge to tackle. It is unfortunately very difficult for robots to replicate human to human interactions. Recent developments in that area have been software focused, with OpenAI's ChatGPT generating realistic human speech. However, there is still potential in improving the physical aspect of the robots, namely their appearance. Many robots seem to lack a physical presence, opting for a more minimalist look. This is good for efficiency and reducing manufacturing costs, but it comes at the expense of reducing human-robot interaction. A person has nothing to focus on when they are talking to "black box" kinds of robotic assistants like Amazon Alexa or Google Home. Robot companies have sought to fix this issue with simulating a face with a screen. Robots like Aido and Zenbo, seen in Figure 1.1, use an LCD to display various facial expressions. However, this lacks physicality and pales in comparison to a physical mechanism that is more tangible and natural to focus on.

(a) Aido robot. Courtesy of Ingen Dynamics Inc.

(b) Asus Zenbo robot. Courtesy of IEEE Spectrum.

Figure 1.1: Two examples of screen-based robot faces.

Eye contact is one of the most important parts of communication. It demonstrates that the other person in listening and valuing what is being said. If a human interacting with a robot cannot make eye contact with it while talking to it, it may lead them to believe it is not really listening. It also serves as a good indicator for when the robot is actively listening, providing the user with visual feedback. Therefore, this project's hypothesis is that adding a mechanical eye will allow the robot to be more friendly and approachable to humans.

1.2 Design Goal

The objective of this project is to design a mechanically actuated eye mechanism for use in human robot interaction. The goal is to have the eye make and maintain eye contact with the user by recognizing their face and tracking it as the user moves around it. Before getting into the specifics, it is important to note some areas that are considered out of scope for this project. The obvious choice would be to make a full face, with two eyes, nose, and mouth, with realistic movements and animations. However, given the limitations of time and budget, only the eye was specifically focused on in this project. Another limiting factor in this design is budget, using common consumer and hobby level electrical components. As such, the physical size of the project is constrained by the side of these cheap components. This is another reason why only the eye mechanism was the focus, as adding other supplementary mechanisms would certainly enhance the user experience, but without using custom components it is difficult to fit other mechanisms in such a compact space as a robot head.

1.3 Existing Products

There are a multitude of mechanical eye and face tracking camera projects on the internet. Numerous patents have been filed for mechanical representations of the human eye, such as one filed by Disney for use in their animatronic characters [3]. This patent uses fluid suspension and electromagnets to move the eye in a smooth manner. This approach is ultimately out of scope for this project, but still demonstrates that eye mechanisms are possible and have been done before. As for face tracking mechanisms, one product that came up in the preliminary research is the DJI Osmo 6 [4]. It is a essentially a high-tech selfie stick, with a 3-axis stabilizing system that allows the user to insert their phone and focus on the subject. This is a sophisticated product, but the mechanism behind it is quite simple: there is a rotating motor for the 3 axes, and some sensors to detect the movement of the user and actuate the motors according to some control law that stabilizes the camera. This served as a basic outline of how this thesis would be completed, albeit with numerous simplifications. However, one thing that this design lacks is any physical presence. It is essentially a series of rods, opting for a minimal design. This may be good for economics, but for human-robot interaction this is not enough. More human elements would have to be allow people to empathize with the robot. Therefore, instead of just the stabilizing mechanism, research shifted focus to humanoid robots.

1.4 Humanoid Robots

One example of a humanoid robot is named Sophia from Hong Kong based company Hanson Robotics [5]. This robot has a realistic human face, complete with synthetic skin and actuated facial expressions. A similar robot named Ameca by British Company Engineering Arts also has a very realistic face [6]. This is very impressive, but something is still off about them. They almost approach a human appearance, but they fall short just slightly. On top of this, there are diminishing returns as a robot approaches human appearance, as it takes much more effort to perfect it. These "creepy" humanoid robots fall into what is known as the Uncanny Valley.

1.4.1 The Uncanny Valley

One of the guiding design principles of this project was based on the concept of the Uncanny Valley, first proposed by Masahiro Mori in his 1970 essay of the same title [7].

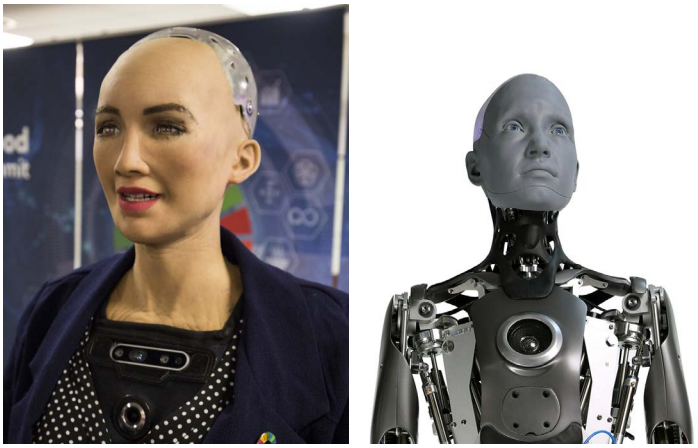


Figure 1.2: Two examples of humanoid robots on the market today.

The classic Uncanny Valley graph is shown in Figure 1.3, which illustrates how making a robot look more like a human can the affect a persons' affinity to it. Industrial robots are designed with functionality in mind, simply having a robotic arm or wheels. As more human-like qualities are added to a robot, like a cartoonish face or legs, peoples' affinity to it goes up. However, there comes a point when adding more human-like qualities where approaching a real human tilts the scale in the opposite direction. If a robot resembles a human very closely but is not perfectly, it falls into what is known as the Uncanny Valley. The person's affinity to the robot violently switches from empathy to aversion. This problem causes many humanoid robot designs, such as those shown in Figure 1.2, to appear creepy or unsettling to humans. This in turn has an effect on peoples' willingness to interact with them. Modern advances in humanoid robots seek to mitigate this issue by going further and further right on the Uncanny Valley graph. If a robot can have perfectly realistic facial expressions, movements, and materials, perhaps they can one day be indistinguishable from a human and escape the Uncanny Valley. In the present day however, technology is not there yet, so a more practical option is instead to target the left portion of the graph. If a robot could hit the peak before hitting the Uncanny Valley, it can maximize its appeal to humans without seeming creepy. Therefore, this project will be following this doctrine, where the robot will have human-like qualities, like a mechanical eye, but will not attempt to replicate human movement directly.

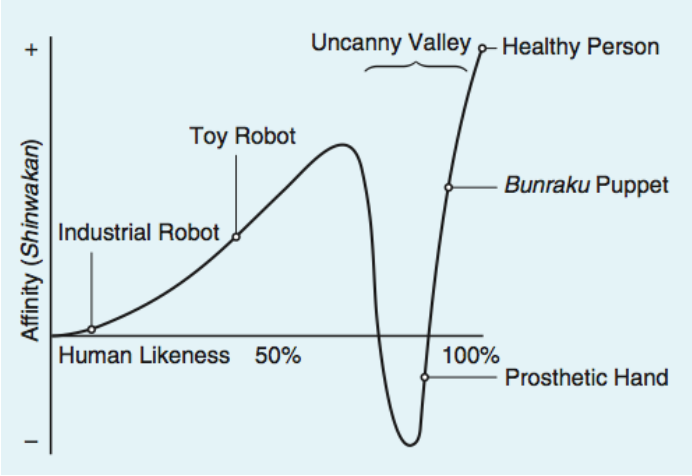


Figure 1.3: Uncanny Valley graph, courtesy of Masahiro Mori.

1.5 One-Eyed Characters

While looking for design inspiration on how to make the robot more friendly, one commonality was identified across numerous examples. Many one-eyed character designs were seen as cute and friendly, with various examples being shown in Figure 1.4. A large inspiration for the aesthetic design of this project was a robot from the video game Portal 2, which were digitally animated but were very expressive for the low amount of joints and components in the design. In particular a YouTube video by the creator DJ Harrigan was an inspiration, where he recreated a character from this video game in real life, complete with actuated mechanisms and aesthetic appearances [8]. This was important in establishing the feasibility of this project, as this project would have to accomplish something similar in a shorter time frame. Another inspiration for the design of the robot was the Pixar Lamp, named "Luxo Jr.", an animated desk lamp that appears in the logo at the beginning of Pixar films. The simplicity of the design is appealing, that with just a head, body and base, the lamp is able to have a physical presence and move around. Another YouTube creator created this recreation of the Pixar lamp, which uses an entire robotic arm and inverse kinematics to follow a face [9]. This is impressive for sure, but out of scope for this project. Still, it gave a good foundation for what the end product should look like and how it should interact with the user.

Research also seemed to indicate that scarier robots had two eyes, such as the Terminator robot pictured in Figure 1.5. Perhaps they look too human, while one eyed robots appear more approachable because they show just enough human qualities without getting too close to the Uncanny Valley. This project proposes this as a novel

approach to facilitating human-robot interaction. The mechanical eye mechanism will be one eyed rather than two eyed, and it is theorized that this would have a better effect on making a robot more approachable and accomplishing the design goal.

Figure 1.4: Various examples of approachable one-eyed characters.

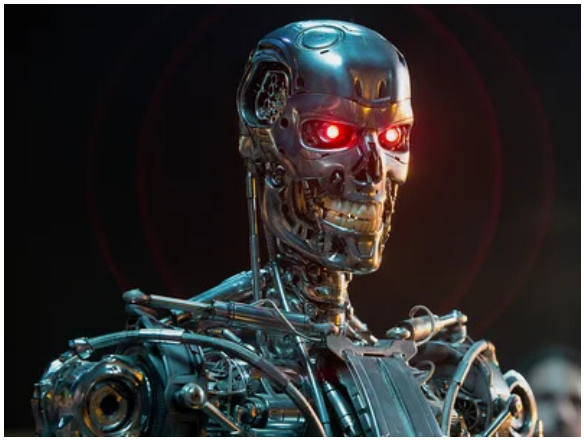


Figure 1.5: An example of a scary robot, courtesy of Paramount Pictures.

Chapter 2

Preliminary Design

2.1 Design Specifications

As a brief overview, the concept for the robot is to have a mechanically actuated eye mechanism with an integrated camera, which is mounted inside a "head", then attached to a "body" which is secured to attached to a rotating "base". These are the terms that will be used moving forward to refer to parts of the robot. The main functionality that the robot is demonstrating is face tracking. The robot, appropriately named Mr. I, will be able to maintain eye contact with the user. To accomplish this behavior, the robot will have three degrees of freedom, two for pan and tilt of the eye, and one for the rotating platform. It could be argued that one of the degrees of freedom is redundant, since simple pan and tilt is enough to track a moving object around a fixed point such as a camera. However, this design seeks to expand on existing face tracking camera products by adding a body to the camera. Not only does this give the robot more of a physical presence, but it also makes the motion of the robot more natural. When humans look at an object, they move their head first, then point their body towards the object. If the robot simply rotated its whole body without moving the eye first, it would look stiff and unnatural. Therefore, this additional degree of freedom is not redundant, in that it serves the purpose of making the robot more inviting and realistic.

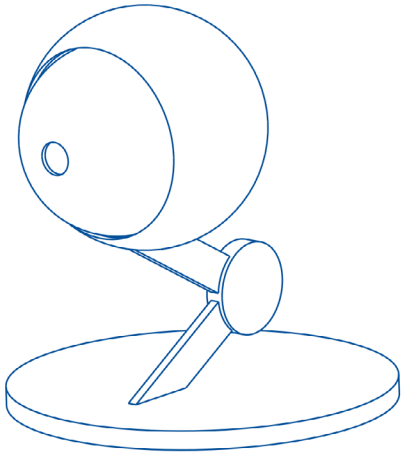


Figure 2.1: Initial Sketch

2.2 Concept Art

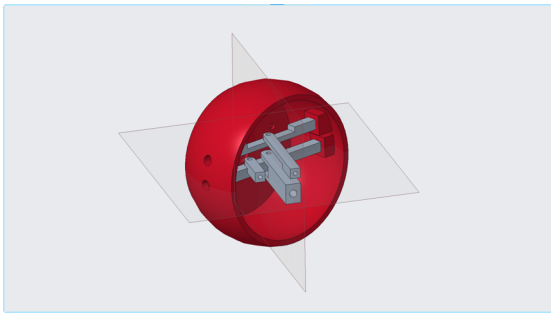
The earliest concept sketch is shown in Figure 2.1. This was an idealized version of the robot with no mechanical components, but it laid the groundwork for the right proportions of the robot and outlined many of physical features that made their way into the final model. Notable features of this sketch are the eye mechanism being offset inside another sphere to act as the "head" of the robot, which would house the mechanical components and hide them for aesthetic purposes. Another thing to note is that the eyeball itself is a sphere with a small hole in the front for a camera see through. This was done to mimic the shape of a human eye, with the "pupil" being the camera hole. With the concept set, design work could then begin on the mechanisms.

2.3 Eye Mechanism

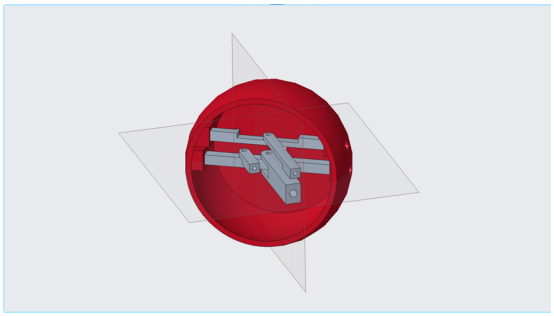
The first and most important part of the robot to be designed was the eye mechanism. The general concept for the eye mechanism was inspired by a project done by hobby content creator Will Cogley [10]. Cogley had created a video and Instructables post about his animatronic eye mechanism. Even more helpful was the fact that he open sourced the entirety of his design, including the model files and Arduino code. While none of the files themselves were used in this project, it was still a great reference and conceptual basis to build my project off of. One thing to note is that the project was a two-eyed mechanism, which would have to be cut down to one eye for this project.

Most importantly, Cogley's mechanism does not incorporate a camera, nor does it accommodate it with the limited space in the small compact design. Cogley's project was on the scale of a normal human eye, but in order to fit a webcam into the eye the entire mechanism had to be redesigned from scratch to scale up the design and make it work for this application.

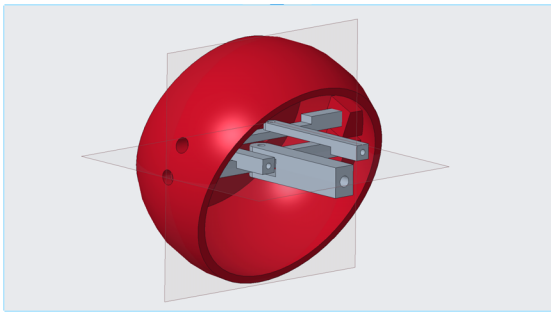
To explain the concept behind the eye mechanism, we can first start with the eye part itself. This is part we are trying to move, since it houses the camera. It is a sphere that rotates around its center point. Rotation about the center of a sphere can be defined in two parameters, which can be referred to as pan and tilt. These move the image camera in the x direction and y direction, respectively. As such, any parts moving the camera in the x-direction from this point forward will be given an "x-" prefix, and the same shall apply for parts that move the camera in the y direction.



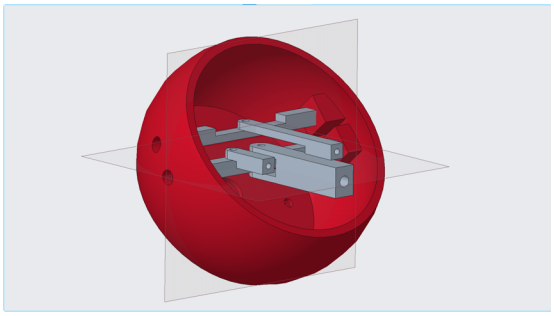
(a) Looking left



(b) Looking right



(c) Looking up



(d) Looking down

Figure 2.2: Basic movement behind the eye mechanism.

The x-motion of the eye is relatively trivial to implement. The eye can simply be attached to a horizontal support, called the x-support, and the support is attached via a pin joint at its center to a main support. Any pushing motion on the x support would exert a torque and pan the camera left and right. This is seen in Figures 2.2a and 2.2b. The y-motion is accomplished by adding a second support above the x-support. By attaching both supports to the eye part with pin connections, this

affords the eye part the extra degree of freedom to tilt up and down as well. This tilting motion can be controlled by exerting a torque on the eye part itself by pushing the y-support forwards and backwards with a linkage connected in the center of the support, shown in Figures 2.2c and 2.2d. The key to making the mechanism work is that the pin joints afford the eye enough degrees of freedom such that the x and y directions can be controlled independently of one another. The moment arms of the x and y linkages intersect at the center of the circle, so motion in the x and y direction can be accomplished by simply pushing their respective supports. This simple motion can be accomplished by two basic servos. A full video animation demonstrating this behavior can be found in Appendix A.

2.3.1 Design Iterations

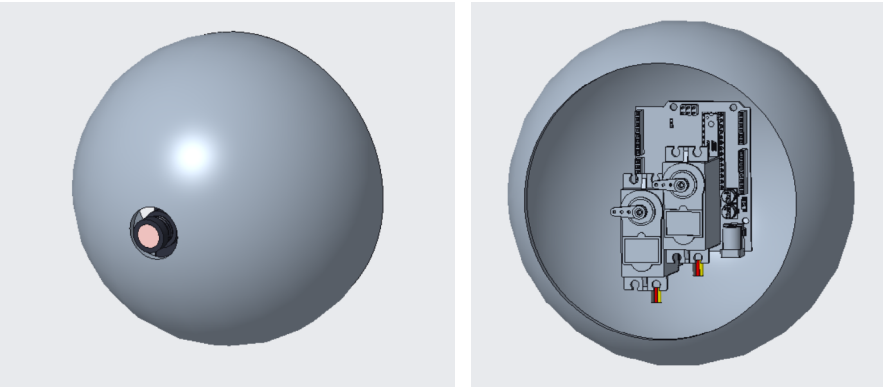


Figure 2.3: First iteration.

The oldest saved draft of the eye mechanism is shown in Figure 2.3. At this early stage, an Arduino Uno was planned to be used as the microcontroller for the project, along with two HS-645MG servos, which are common servo motors used in hobby projects. These servos in particular were chosen for their impressively high torque for their size, a standard 40x40x20mm. At this early stage, higher torque motors were desirable for safety margins, especially since the rest of the mechanism had not been designed yet, but in the end these servos proved to be a wastefully expensive choice. The final servos used were half the price and had the same form factor, so it did not require much refactoring of the design; it was a drop in replacement. Another component that went through some changes was the camera. In this early stage, an ArduCam was planned to be used, as it is able to connect up directly to the Arduino. However, after further research it was concluded that the Arduino chip is not powerful enough to run the computer vision algorithms required to achieve face tracking, so

this part was changed to a USB webcam to easily be able to interface with a laptop. The Logitech C615 USB Webcam was chosen for this purpose because of its low price and compact design, as well as being 1080p. This higher resolution would help in the computer vision developed later on. The webcam incorporated into the eyeball can be seen in Figure 2.4.

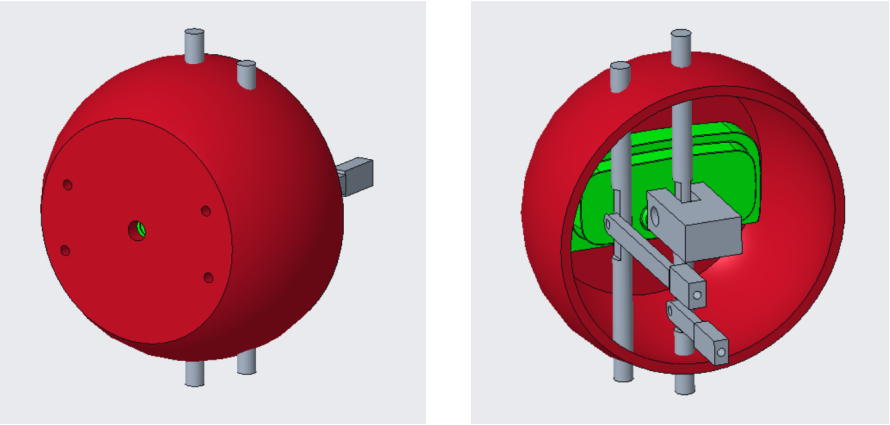
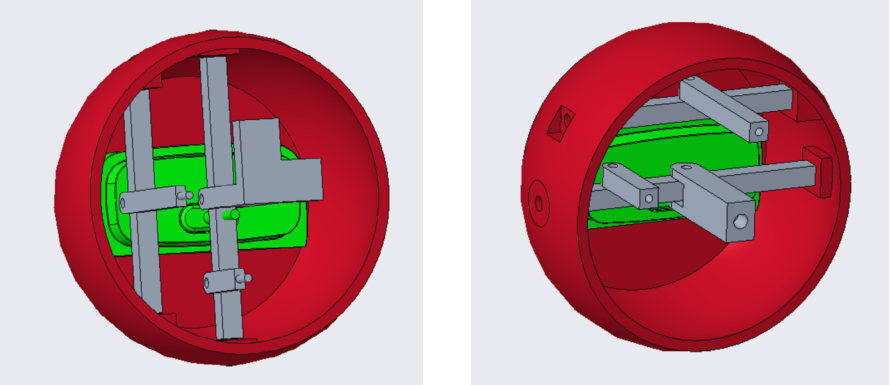


Figure 2.4: Second iteration.

After further design and some early 3D printing testing, one issue with the design was found: the cord for the camera was intersecting with the main vertical support. So, the entire design had to be rotated 90 degrees to put the linkages above the camera, since the cord was coming out of the bottom half of the camera. This is the orientation that is reflected in the final design today, shown in Figure 2.5



(a) Before (b) After

Figure 2.5: Changes in the orientation of the eyeball.

After this stage, servo motors were ready to be added in to supply the pushing motion needed to pan and tilt the eye part. One thing to note is that the servos had to be attached with ball linkages rather than pin joints. This is because the pushing

motion of the eye linkages is not strictly one-dimensional. The y-linkage for instance moves slightly up and down as it rotates around the sphere of the eye part in addition to being driven forwards and backwards by the servo. For this reason, the ball joint was necessary to add some slack in the mechanism to account for this slight up and down movement. This was not necessarily required for the x-servo, as its motion is completely in line with the servo arm, but a ball joint was still used to accommodate any manufacturing defects or misalignments. Since this prototype was done before the rest of the head and robot were designed, a temporary mounting block was modeled and mounted to a piece of scrap metal just to get something up and running quickly. This was named as the first working prototype of the eye mechanism, shown in Figure 2.6.

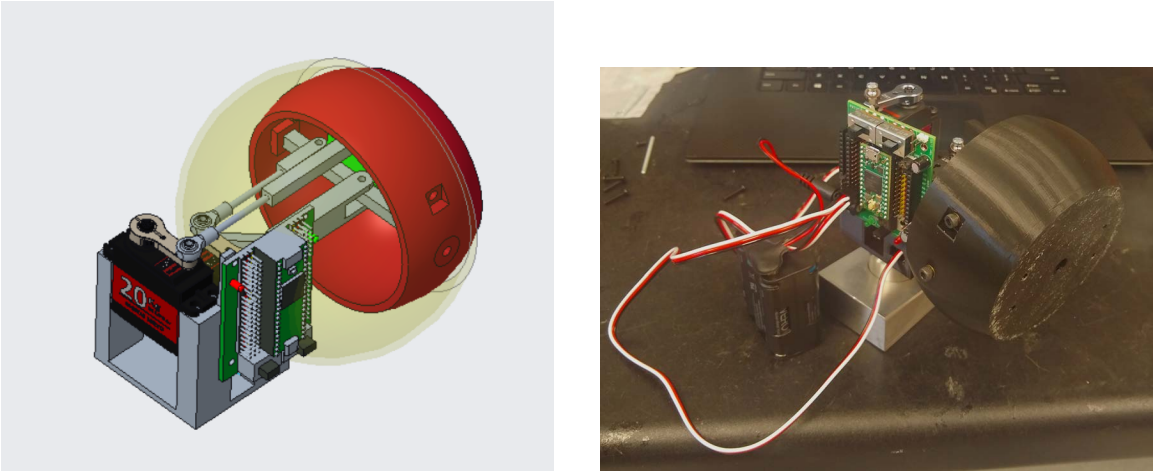


Figure 2.6: First working prototype.

In this prototype, a simple Arduino script was used to cycle the eye through looking through 8 directions, up down left and right and diagonals. The diagonals were to test the independence of the x and y controls, which was successful. A video of the first prototype moving can be seen in Appendix A. This prototype was powered with a Teensy 4.0 microcontroller [11], which is interfaced with a motherboard designed by Patton Robotics [12]. This board allows for the board to be powered by an external power source, in this case a 6 AA battery pack. The board features a voltage regulator to step down the 9V battery source to a useable 5V for the servos and microcontroller. For this reason, this microcontroller and motherboard combination was used all the way into the final design.

One shortcoming identified in this stage was the range of motion of the eye. Previously the eye linkages were placed towards the outside of the eye sphere, as it would increase the moment arm of the servos and allow them to exert a greater torque

on the rotating eye. However, testing showed that this greatly reduced the eye's range of motion, as the back of the eye would intersect with the linkages. The initial test also showed that the torque required was much smaller than planned, since the servos are simply rotating the eye, which requires less force than translating it. As such, the linkages were then moved as close to the center as possible while leaving clearance for all the fasteners and support beams. To accomplish this, the linkages were made more slender and incorporating notches in the bars so that the linkages could rotate within the supports rather than on top of them. The parts of the beams outside the notches were thick enough to provide structural stability to the eye mechanism. These optimizations reduced the material costs of the parts and made the design as compact as possible.

2.4 Head

The head of the robot is the part that surrounds the eye mechanism. This is to protect the mechanism from dust and debris, protect the user from getting caught in the moving parts, and also give the robot a better aesthetic look. The head consists of four plates that form a sphere around the eye mechanism. The motor mounting block has multiple protrusions that provide attaching surfaces for the plates. This is shown in Figure 2.7. The top two plates were designed to be secured with neodymium magnets to be easily removed for inspecting and servicing the inner mechanism. The magnets are not strong enough to hold the bottom plates upside down, so the bottom plates are screwed in to secure them but still allow for disassembly.

2.5 Body

The body of the robot is the part that connects the head to the base. The head is secured to the body via a mounting feature in the motor mount that straddles the body, then secures it in place with two screws. The early design for the body was planned to have multiple parts and have a more acute angle. The idea behind the bend was to model a person bending over, as this would give the body a more human-like shape than a straight line. However, it was found that the initial sketch looked unnatural as it was too bent over. The figure below shows the change in angle that produced a much more natural body shape that made it into the final model.

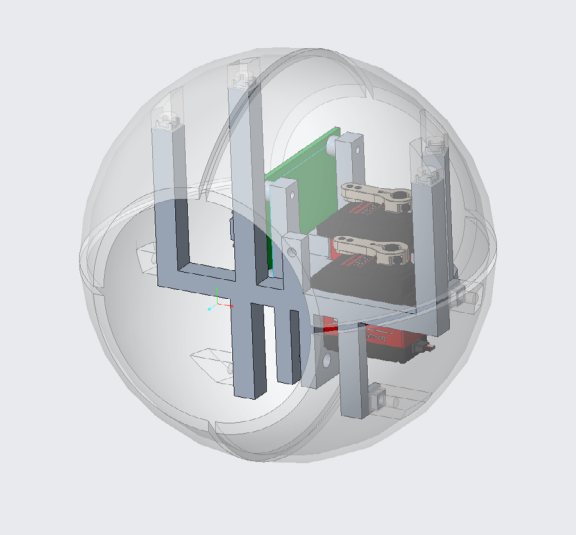
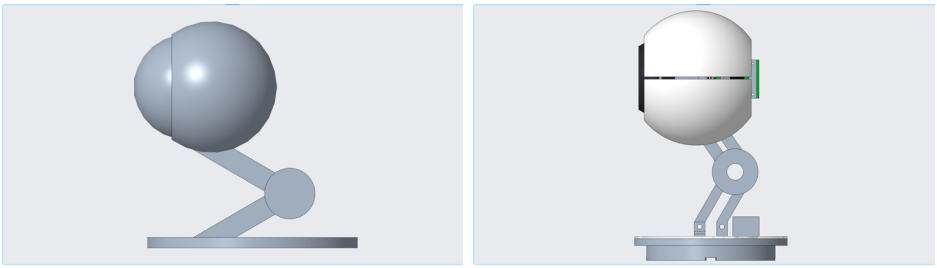


Figure 2.7: An inside look at the head assembly.



(a) Before (b) After

Figure 2.8: Comparison of the shape of the robot body.

2.6 Base

The base of the robot is what allows the robot to stand on a surface by itself. The base had to be large enough such that it would provide enough lateral stability to keep the robot from tipping over. The base also allows the robot to rotate 360°, as outlined previously. The base is divided into two sections, the top half being connected to the rest of the robot and the bottom half that is stationary. A stepper motor was used to rotate the top half of the base, which in turn would rotate the rest of the robot. The stepper motor is attached to the top half of the base. The motor is mated with a gear that interfaces with an internal gear modeled around the bottom half of the base. This internal gear mechanism drives the third and final axis of the robot.

Gears incorporated in the base were obtained from step files from McMaster-Carr [13][14] and scaled and modified to fit this application. The original inner and outer gears had pitch diameter of 0.5 in and 3.00 in respectively, yielding a gear reduction

ratio of 1:6. This ratio was chosen because the motor chosen was relatively small, but capable of very fast speeds. The gear ratio was chosen to be low enough to gear down the speed and increase the torque of the motor, while also keeping the gears large enough to be 3D printed with reasonable accuracy. The STEP files obtained from McMaster-Carr were scaled up by a factor of $\frac{11}{6}$ to fit in the model.

Additionally, it was planned to have a slip ring in the center to allow for continuous rotation of the base, shown in Figure 2.9. This idea was tested but it was found that the slipring was not reliable enough to transmit camera signals over USB. The way a slipring works is it has brushes that make contact with metal inside the slipring as it rotates. This allows it to keep electrical contact but rotate freely. Otherwise, if there were solid wires, they would get wrapped up in each other. However, these brushes are not perfect, and ideally there is always some brushes making contact but for a small instant they may not be making contact. This is fine for a low frequency signal but USB is high bandwidth. Moreover, a camera needs to send a lot of information over USB. This makes it catastrophic if even a single bit of information is lost. It was found that the camera was able to be routed through the slipring but once the slipring rotated even a little bit, the camera feed was distorted and disconnected from the PC. This idea was scrapped from the project, instead constraining the motor to rotate 180° in either direction for a full 360° range of motion. However, this was actually did not have an impact on the functionality of the robot since it can be assumed that the user will not be constantly encircling the robot while having a conversation with it. If the robot is on a table against a wall for instance, there will never be a time where it needs to make a 360° rotation, let alone multiple.

2.6.1 Analysis

A mechanism analysis was performed in Creo to determine the proper placement of the eye in relation to the head and where to place the linkages to get away from the eye. Instead of being constrained as a fixed part, the eye part could be constrained as a ball joint, with the center point being the center of the sphere. Through this analysis the angle of motion of the eye was 40 degrees in a cone. A screenshot from this analysis process is shown in Figure 2.10.

It was also a concern that the robot would not be balanced and potentially fall over. This is due to the top-heavy eyeball being mounted to a slender aluminum body. To test this, a center of gravity analysis was done in Creo to ensure that the design is balanced in both the forward to backward and left to right directions. In order to

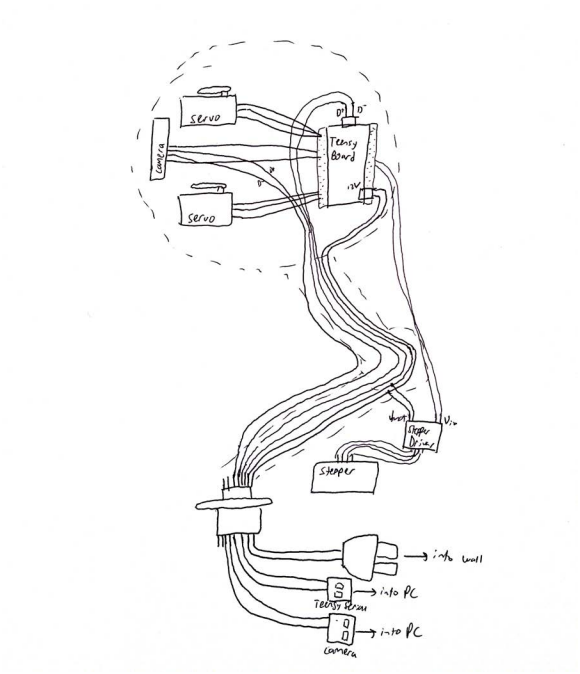


Figure 2.9: Early slipring sketch that was scrapped.

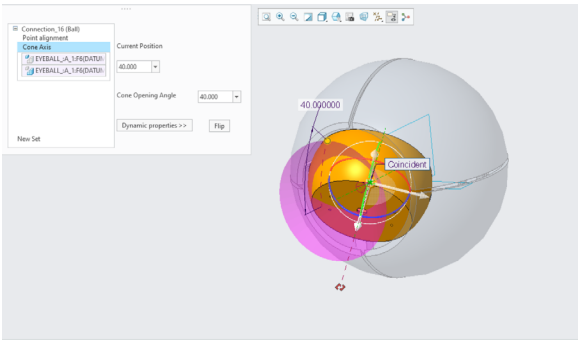


Figure 2.10: Eye range of motion analysis.

perform the calculation in Creo, some approximations had to be made. In Creo, one must assign each part a material with its particular density since different densities affect the mass distribution. The 3D-printed parts were all assigned PLA, while the servo motors and any metal parts were assigned a material of Stainless Steel. This was enough to give a rough approximation, dividing the assembly up into "heavy" and "light" components. Figure 2.11 shows the results of this analysis. The "X" in the figure indicates the center of gravity. As the figure shows, the center of gravity is nearly perfectly balanced in the left to right direction, but slightly offset in the forward to backward direction. Since the robot will be rotating around the center of the platform, it is best if the sphere of the head is centered rather than the center of gravity for symmetry purposes. An offset of half an inch is adequate for keeping

the model balanced, so the project proceeded with keeping the sphere of the head centered on the circular platform for symmetry.

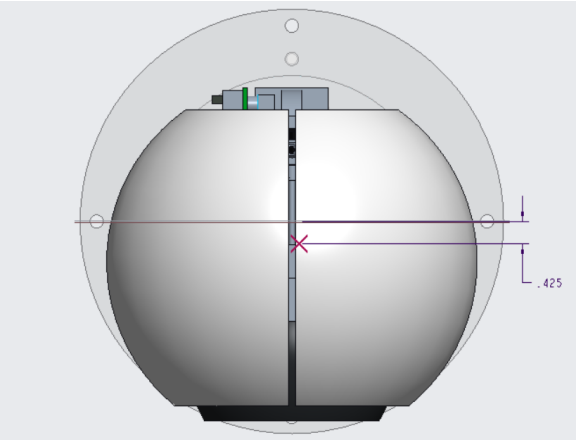


Figure 2.11: Result of center of gravity analysis.

2.7 Finalized Model

The final CAD model of the Mr. I robot can be seen in Figure 2.12. A drawing for the overall dimensions of the model is shown in Figure 2.13. Note that the dimensions provided in the model are only intended to give the reader a sense of scale of the model, and are kept minimal for clarity.

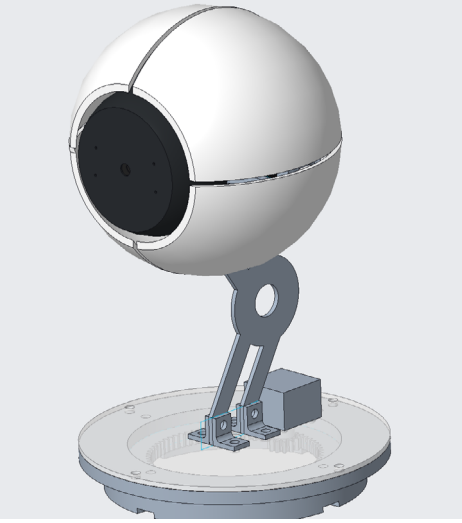


Figure 2.12: Final CAD model of the robot.

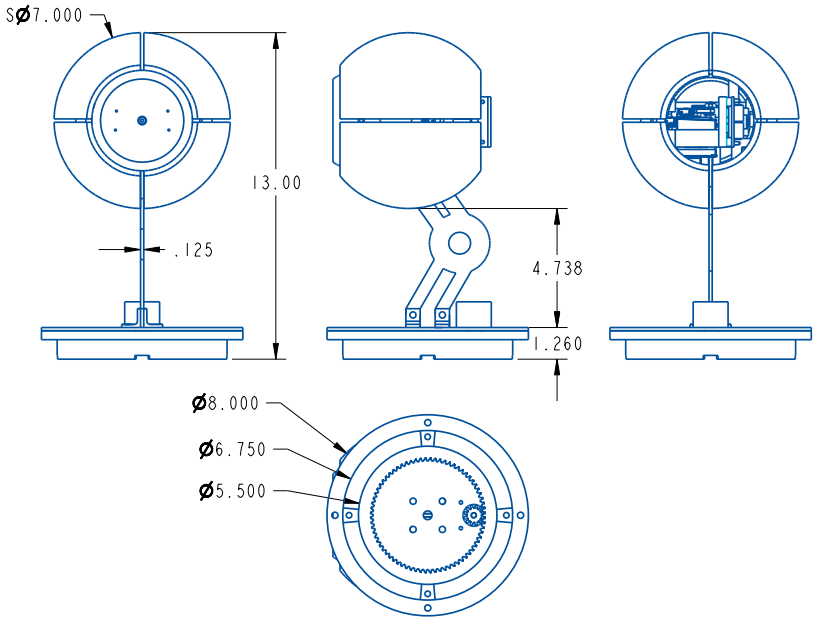


Figure 2.13: Drawing of the full model.

Chapter 3

Physical Model

3.1 Manufacturing

3.1.1 3D Printing

All of the custom modeled parts were 3D printed with PLA on a Creality Ender 3 Pro. This allowed them to be manufactured quickly and accurately. However, 3D printing is not perfect. When the plastic exits the nozzle of the printer, it expands, causing part tolerances to be off by a few hundredths of an inch. This is a difference maker in whether or not two parts will fit properly. This is especially troublesome when the prints take 12+ hours to print and need to be reprinted. A technique used to work around this issue was to slice out a piece of the part where the dimensions mattered and print that first. If it doesn't fit, the part can be reprinted at a fraction of the time and material cost. This allows for much faster tolerance iteration. Once the tolerances have been tuned, the full part is printed and fits just as planned.

3.1.2 Aluminum Body

The part for the body was designed entirely 2-dimensional for CNC milling purposes. The body was CNC milled out of an 1/8" thick aluminum sheet. A screenshot of the CNC setup in Creo is shown in Figure 3.2. The part was made out of aluminum because of its slender shape while also bearing the load of the entire head assembly on top of it. A 3D printed part may have not been rigid enough or too weak to hold the weight, although it would have been easier to manufacture. 3D printing is great for a lot of things, but this was a job for the CNC mill. The aluminum body was

Figure 3.1: Various failed parts that had to be redesigned.

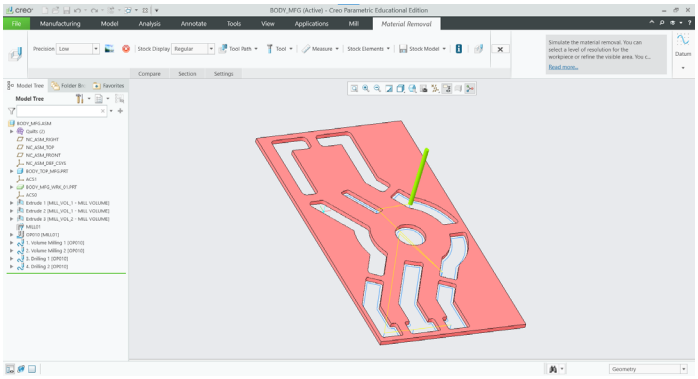


Figure 3.2: Screenshot of making the file in Creo to be sent to the CNC machine for manufacturing the body.

secured to the base with 4 90° brackets. These were cut out of a 0.75”x0.75” piece of L channel stock and drilled on the milling machine to position the holes properly. Figure 3.3 shows the drawing used as a reference to mill and cut the four brackets.

3.1.3 Acrylic Base

The base was laser cut out of a sheet of 1/8” thick clear acrylic. This material was chosen for aesthetic reasons and for its ability to be laser cut, which speeds up manufacturing time significantly.

A lazy susan bearing was used for the base to allow it to rotate freely. This is a common bearing used in turntables, where there are steel balls outlining the perimeter of the circle which reduces the friction while rotating the two concentric circles. The lazy susan bearing was an essential part in making the robot spin around its base

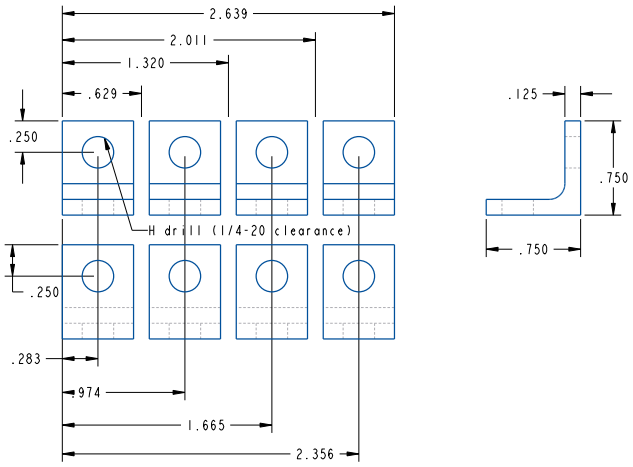
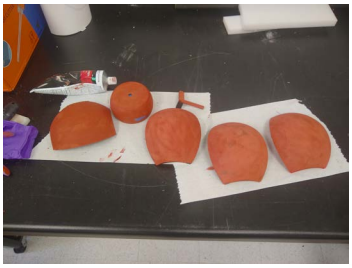


Figure 3.3: Drawing used for manufacturing the four metal brackets connecting the body to the base.

without requiring a large and powerful stepper motor.

3.2 Finishing Parts

Another aesthetic choice for the product was painting the four plates used in the head assembly as well as the eyeball part. 3D printing produces layer lines in a part, which is fine for prototyping, but for a final model these are undesirable, especially in round parts where layers are more apparent. The parts were first treated with a layer of Bondo filler putty, which filled in some layer lines and parts of the print that had poor surface finishes. The filler putty was then sanded with incrementing grits of sandpaper to smooth out the surface of the dried putty. This process was repeated several times until most of the peaks and valleys were smoothed out. After this, a thick layer of automotive filler primer was applied to the parts to further smooth out the parts and prepare the surfaces for painting. The filler primer has a passive smoothing effect because of the surface tension of the liquid, plus it is very easy to sand. The surfaces were brought up to 400 grit before getting ready to paint. The shells were painted white, while the eyeball was painted black. These colors were chosen to mimic the colors of an eyeball. Figure 3.4 shows three intermediary pictures taken throughout this process.



(a) Filler putty application

(b) Filler Primer + sanding

(c) Spray Painting

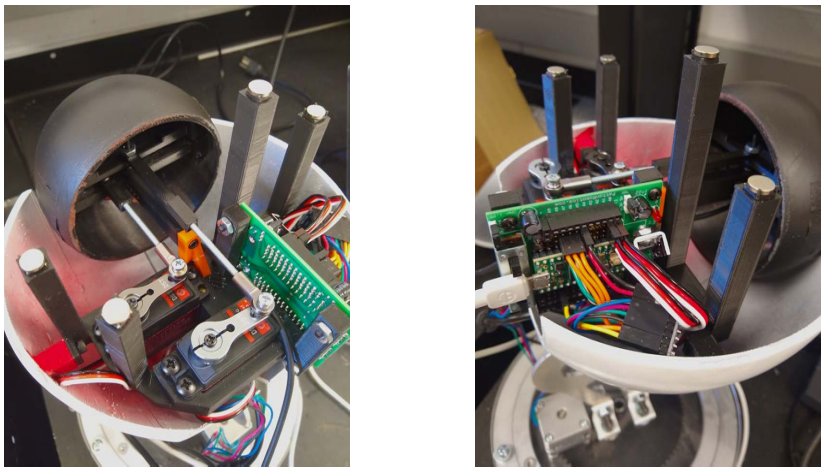
Figure 3.4: Process of finishing parts

3.3 Mounting Components

The head assembly was designed to accommodate the microcontroller, servos, and most of the wiring. This was done to keep a clean look to the robot. An image of the inside of the head can be seen in Figure 3.5a. The servos are mounted onto the mounting block with 8-32 screws. They are held in by heat set threaded inserts, which allows for metal threads to be added to a 3D printed part. This provides a strong metal screw hole, as opposed to tapping the plastic itself. To mount the camera, existing geometry of the product was leveraged. The camera enclosure is essentially a plastic box held together by four screws on the front face. By removing this front face, we can reuse these screw holes as mounting geometry, replacing the face of the camera enclosure with the front face of the eyeball. The eyeball part was designed such that the existing screws could mount the camera. The tolerances on such a small part were a challenge, so the previously mentioned iteration technique was used to get the part to fit properly. The Teensy daughter board was mounted with 5-40 screws, shown in Figure 3.5b. These components are all mounted with screws, which was an an important consideration throughout the design process. This was to allow for the robot to be completely disassembled for troubleshooting and replacing broken parts.

3.4 Final Model

The final model completed and assembled is shown in Figure 3.6.



(a) (b)

Figure 3.5: View of the mounted components inside the head.

Figure 3.6: Photo of the final model.

Chapter 4

Electronics and Software

One of the constraints set early on in this project was keeping the robot low budget. This meant that no proprietary algorithms or software could be used; everything had to be free and open source. Luckily, the internet has a wealth of different options that people have created and make open for others to use.

4.1 Inspiration

The first step of any electronics design is establishing the feasibility of the design goal. Various examples were found on the internet demonstrating similar projects and inspiration was drawn from them to incorporate into this project. For instance, in a video by the YouTube channel Core Electronics showcased face tracking camera using OpenCV and a Raspberry Pi, shown in Figure 4.1. This seemed to be a good proof of concept and was important in establishing the feasibility of this project, especially in such a short time frame. The problem with this model is that it is nothing more than a proof of concept, rather than a full product. It has minimal electronic components and the mechanism is fairly simple. The pan servo moves the tilt servo completely, which accomplishes two degrees of freedom, but is simplistic and most importantly does not resemble a natural eye nor does it resemble the dynamics of my eye mechanism. Another thing to note is the speed and smoothness of the motion of the face tracking mechanism leaves much to be desired, which was an area focused on in this project.

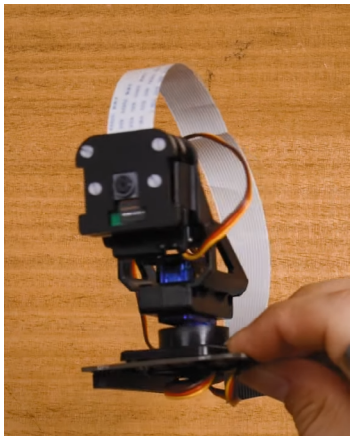


Figure 4.1: Pan and tilt mechanism used in the Core Electronics example.

4.2 Components Selection

4.2.1 Microcontroller

All components used in this project can be found in the Bill of Materials in Appendix F. One important choice to note is the microcontroller, which is essentially the brain of the robot. The Teensy 4.0 was chosen for this for its low price but incredible speed. The Teensy can also be programmed in Arudino, which allows it to tap into the wealth of documentation and example code snippets online. To interface with the components it is possible to simply use a breadboard, but this project opted to leverage a dedicated robotics board to facilitate the process. This board, called PRT_28 by Patton Robotics, provides numerous convenient features like a DC jack, voltage regulator, capacitors for despiking, and separate power rails for 12V and 5V. These are all components that could have been procured on their own, but for $26 the board was the smarter choice given the time frame. A future version of the project would use a custom PCB that would allow for a large simplification of the circuitry. For instance, in hindsight much of the functionality of the Teensy is superfluous and can be accomplished with a much cheaper ATmega328P chip. Still, the Teensy accomplished what it was intended to, which was to serve as a way to control and synchronize the motion of the three motors in the robot.

4.2.2 Actuators

As stated in Section 2.3.1, the motors chosen for the project were common, affordable parts. The servo model used in this design is the DS3218MG, with a control angle of

270°.

Since the stepper motor is being geared up and the robot does not weigh much, the torque requirement on the stepper motor is not very large. As such, a smaller form factor was chosen to fit on the 8" diameter base. The component chosen was a NEMA 14 stepper with 200 steps per rotation. With a control angle of 1.8 per step, these kinds of stepper motors are not known for their precision. However, this can be mitigated by gearing up the motor, which was explained previously, and choosing the right stepper motor driver.

To make the stepper motor spin, the electromagnetic coils in the motor have to be driven in the right order to spin the shaft. This can technically be done with the Teensy, but the smarter choice was to procure a daughter board with dedicated circuitry to efficiently drive the motor. This board, called the stepper driver, powers the stepper motor in accordance to signals from the Teensy. The one used in this project is a RepRap StepStick, running the A4988 chip [15]. This board communicates with the Teensy through two pins, denoted by the STEP and DIR (direction) pins. Every time the Teensy powers the STEP pin, the motor advances one step in the direction indicated by the state of the DIR pin. It is important to note that the motor is running at 12V, while the rest of the components use 5V. The stepper controller also serves the purpose of separating the two power lines and keeps the microcontroller safe from high voltage backflow. Another useful feature of the A4988 is microstepping, which allows the motor to move at fractions of a step. This enables much greater precision than the base 200 steps per revolution of the motor. For the greatest precision and smoothness of the movement, the lowest microstepping configuration was selected at 1/16 step, now giving the motor 3200 steps per revolution. Combined with the 1:6 gear ratio, this allows the stepper motor to rotate the robot with a resolution of $200 \times 16 \times 6 = 19200$ steps per revolution, or 0.01875° per step.

4.2.3 Miscellaneous

A 12V 5A AC adapter of was used to deliver power to the circuit. The 12V of the jack powers the stepper motor directly and passes through a voltage regulator to power the rest of the 5V circuitry.

Additional components include a speaker and microphone which were incorporated into the design for a nominal user to have a natural conversation with the robot listening and responding. The current functionality is limited to playing sound files from a Python script, but the main purpose of including these components was to

future proof the design for features that are planned to be included later down the line.

4.3 Circuit Diagrams

The circuit diagram used to wire up the components can be seen in Figure 4.2. Note that the only components that had to be wired up were the Teensy microcontroller, the two servos, the stepper motor and its associated motor controller board. The other components, such as the camera, microphone, speaker, and the Teensy serial data cable all used a plain USB 2.0 connector, so they simply needed to be plugged into a USB hub and connected to the computer. One thing to note is that the 5V power line of the Teensy USB cable is cut. This isolates the Teensy power from the laptop and allows the laptop to communicate with the microcontroller over the two data lines without the potential of any backflow voltage that could damage the laptop. These USB cables come out of the back of the robot's head, pictured in Figure 4.3.

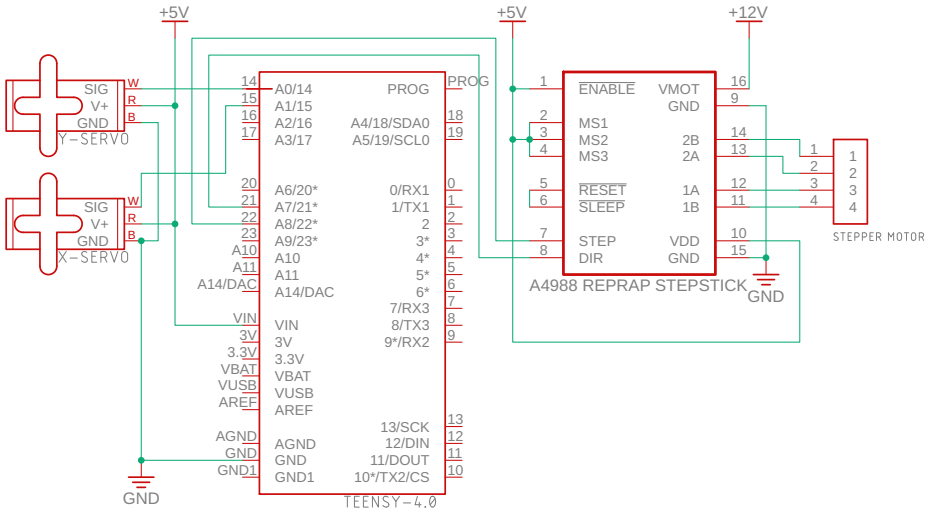


Figure 4.2: Electrical schematic of the circuit. Generated in Autodesk EAGLE.

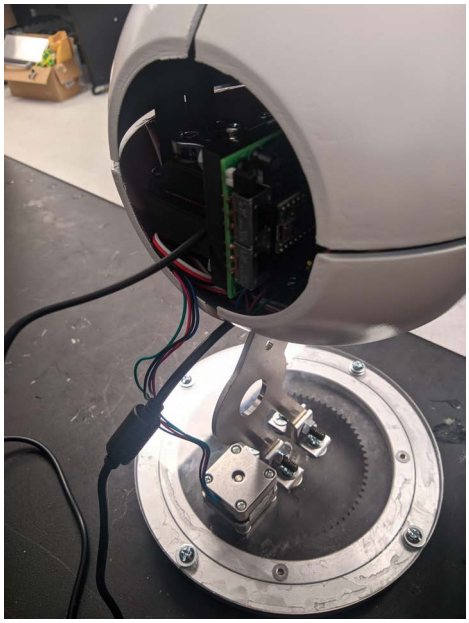


Figure 4.3: A photo showing the wires coming out of the back of the head.

4.4 Software and Programming

The algorithm for following the eyeball needed to be simple but also fast. One measure taken was avoiding the practice of blocking, which is when the processor waits for one action to finish before beginning the next. This would essentially freeze the Arduino when any motor is running and rather than all the motors moving at once, causing them to move one after the other. Instead, the code utilized software libraries that used non-blocking code to drive the actuators, which enabled all the components to move at the same time. This helps with smoothing out the motion of the system and keeping the robot more appealing to interact with.

4.4.1 Software Libraries Used

- Python
  - OpenCV 2 [16]
  - face_recognition [17]
  - pySerialTransfer [18]
- Arduino
  - SerialTransfer [19]

- ServoEasing [20]
- AccelStepper [21]

4.4.2 Control Loop

Figure 4.4 shows the control loop for the program. The control is discrete rather than analog, since the microcontroller runs in discrete time steps. The algorithm is as follows: First, the camera on the computer turns on and receives an image. An example of the image output from the camera is shown in Figure 4.5. Then, the Python facial recognition library is used to find the center of the bounding box drawn around the face in the image and calculates the distance from that point to the center of the image. This distance is then scaled by a constant and sent as a move command to the motors. One effect of this approach is that larger move commands are sent to the motor as the face is farther from the center of the screen. This ensures that the camera will reach the face in fewer moves than a constant stepping distance. As the face gets closer to the center of the image, the motor commands are smaller, giving the system the ability to perform micro adjustments to center the camera properly.

A deadzone is used to decide whether or not to command the motors, since it is practically impossible that the face will be at the very center pixel of the image. The deadzone chosen for this was the middle 6% of the width and height of the image. This seemed to provide a good balance between allowing for imperfect alignments while still producing a convincing face-tracking result to the user.

The servo and stepper motors in turn move the camera, which then captures a new image and the cycle repeats. It is important to note that the move commands are relative to the actuator's current position, so the Python code does not actually know what position the motors are at. All it sees is the camera footage and sends commands to the Teensy. Once the Teensy is ready to receive a command, it picks the most recent one out of the stack and executes it. Of course, there are some hard coded limits in the code to make sure the actuators do not exceed the maximum designed range of motion, which would cause the machine to tear itself apart, which is discussed in more detail in Section 4.5.

To decide when to use the stepper vs the X-servo to correct the x-position of the camera image, the X-servo was set to run 70% of the distance total distance and the stepper was set to run 30%, so that when they are run at the same time they center the camera correctly.

The control loop models a proportional controller, since the control output linearly

scales with the measured input. The full Python and Arduino code can be found in Appendices C and D, respectively. Lots of the skeleton code for this algorithm was based on examples provided by face_recognition, pySerialTransfer, and SerialTransfer.

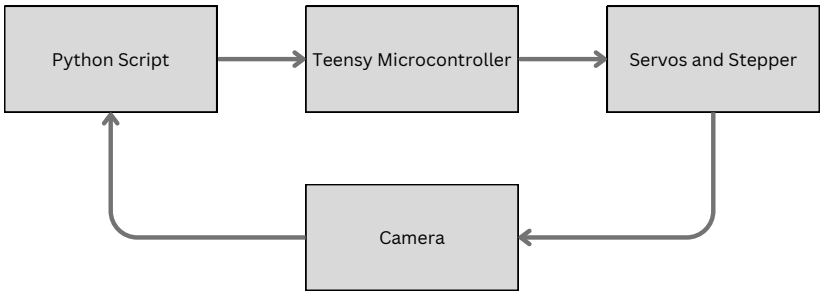


Figure 4.4: Control Loop for the face tracking algorithm.

4.5 Actuator Range of Motion

To command the servos to move to a set position, the microcontroller sends it a PWM pulse in the range of 500-2500 ms, where 1500 ms is the middle position of the motor's range of motion. The eye mechanism does not need the full 270 degrees range of motion of the servos, in fact if the servos rotate that far it will tear the mechanism

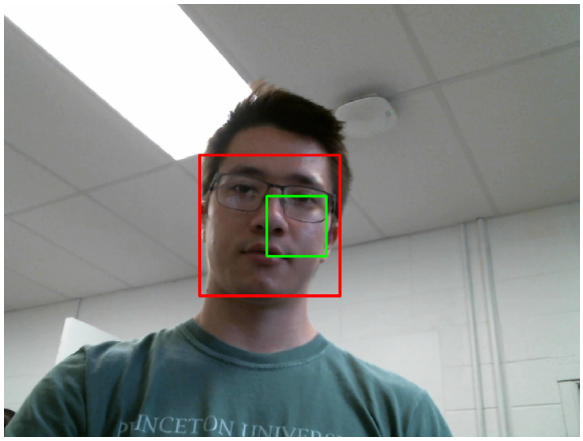


Figure 4.5: Example of what is displayed on the screen for facial recognition.

apart, which happened multiple times by accident. For this reason the millisecond ranges allowed to be sent to the servo were trimmed.

The mechanism was designed so that the motors would hold the eye in a neutral position looking straight ahead when the servos are in their middle 1500 ms position. However, due to assembly imperfections this was not actually the case, so the motors had to be calibrated to find the servo position that would produce the desired neutral position of the eye. As such, after some experimentation these limits were empirically set to 1200-1670 ms for the Y-servo and 1320-1680 ms for the X-servo. Translating this into real world measurements, this means that the arm of the Y-Servo and X-Servo can sweep through an angle of 63.45° and 48.6°, respectively. The stepper motor limits were determined such that the beyond 180° in either direction to protect the cables from getting wrapped up. Since a full rotation of the robot would be 19200 steps, the limits on the motor were set to 9560 steps in either direction.

A confusing detail to note is that in the software the ends of the servo rotation are called 0 and 180, but these are just arbitrary variables. The true angular displacement of the robot is scaled according to the inputted "angle" and the microsecond range previously mentioned. For instance, a servo command of 90 to both motors resets the mechanism and returns it to the neutral looking ahead state. For simplicity, from this point forwards the servo angles will be referred to by their software variables rather than their real world counterparts.

4.6 Integration

Integrating these different components together was a significant challenge. Most notably, the facial recognition library is written in Python and is too computationally intensive to be run on a microcontroller, so the code has to be run on a PC instead. The PC then has to communicate with the Teensy through the Python script to tell the microcontroller how it should move the actuators. Fortunately, the serial data system is designed for this purpose.

Serial data allows for digital data to be sent through electrical signals in a cable to the microcontroller. This data can then be interpreted as a integer, character, or other any data type. This works both ways as well, so the microcontroller can also send values to the PC. This is a relatively trivial task if only one value is being sent to the Teensy, since Arduino has built in serial libraries for this purpose. For instance, I can type in the letter b on my computer and the PC sends it over the USB cable, then the microcontroller can recognize that data and do something with it, like

printing out the value or moving a servo according to an inputted number. [22][23]. However, the challenge is controlling multiple actuators at once. The stock Arduino library uses blocking code, so this would slow down the system tremendously. Also, the microcontroller needs some way to identify which commands to send to which motor. This can be done with the base Arduino library [24], but a more complete solution was to use pySerialTransfer.

The pySerialTransfer library takes advantage of the C data structure *struct*, which allows for packaging multiple variables into a single data structure. This struct is composed of the two servo move commands and the stepper move command. This allows us to separate the move commands into distinct variables that can be read and used by the microcontroller. The struct is first filled with values by the Python script based on the previously mentioned methodology, then the Python script fills a buffer with the struct. Once it finishes putting all three values in the buffer, it sends it off to the Teensy with serial data. The library pySerialTransfer interprets this serial data and fills in the corresponding struct in the Arduino script with the appropriate move commands. Then, the Teensy can use these values to actuate the motors accordingly. The full setup, including the robot and laptop, can be seen in Figure 4.6.

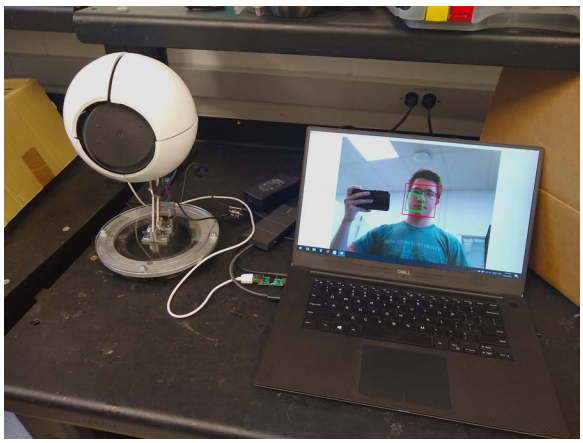


Figure 4.6: Photo of the entire setup.

Chapter 5

Results

5.1 Dynamics

Early versions of the robot used arbitrarily chosen scaling to approximately get the motions right, but a more numerical concert approach based on the dynamics of the system was desired to tune the parameters. The dynamics of the system start at the servo inputting an angle, which then transfers through numerous pin linkages before finally being outputted by the camera seen by the computer. As such, a test was set up to find the relationship between the servo angular displacement and the resulting planar image movement. The test was done by making a custom Python script that only controlled a single axis at a time. The script would record the center of the recognized face, while the user looked at it keeping their face as still as possible, then advance the angle of the servo by 3 degrees, then repeat. This loop ran until the face had moved out of frame of the camera and the software could not see a face anymore. The stepper motor was tested the same way, just was instead told to advance 100 steps every loop. The tests were done over three trials and averaged. The results of the data are plotted in a scatter plot form in Figure 5.1.

A line of best fit was calculated using the data collected in the tests. The slope of the line of best fit was found for each trial and then averaged for a final result, shown in Table 5.1. It was found that for the Y-Servo, for every degree the servo arm moves, the face in the image moves by 7.98 pixels. The results for the X-Servo were similar, with a slope of 7.28. For every step of the stepper motor, it was found that the image moves 0.25 of a pixel. This makes sense since a full revolution of the robot would require 19200 steps, so step commands sent to the motor should be much higher than

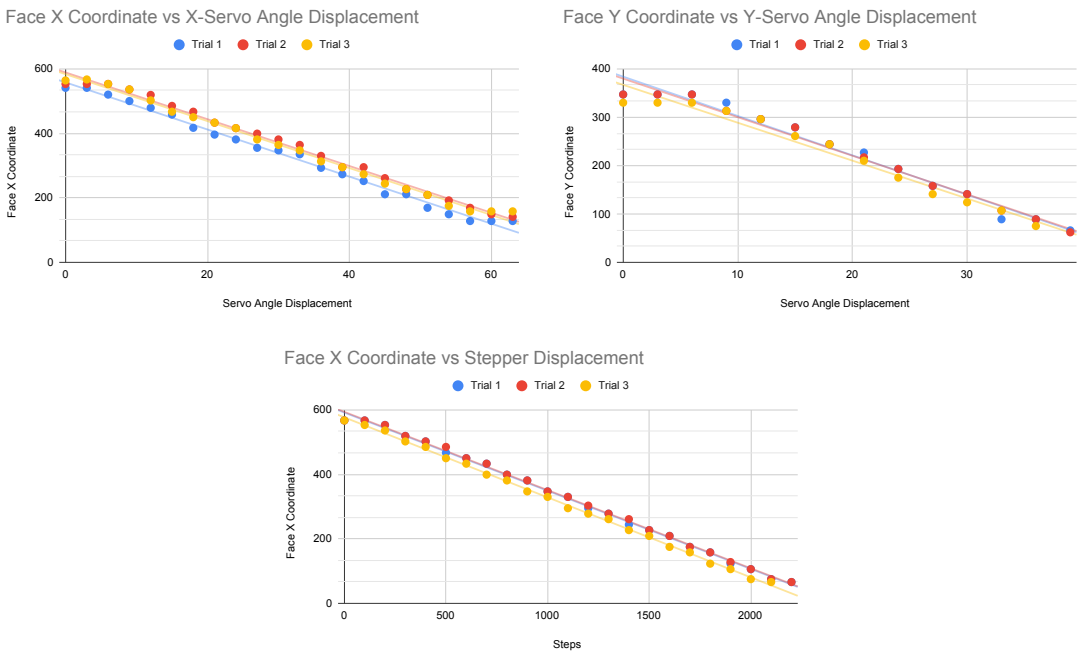


Figure 5.1: Scatter Plots of Resulting Face Location

the servo command to move the same angular displacement. The results of these tests were then used to inform the parameter values of the motor commands in the Python script. The values are used as the scaling factor that converts the distances in the image to the servo movement commands, as seen in Appendix C.

Table 5.1: Average slope of regression line for 3 trials.

	Trial 1	Trial 2	Trial 3	Average
Y-Servo	-8.133	-7.972	-7.835	-7.980
X-Servo	-7.302	-7.253	-7.280	-7.278
Stepper Motor	-0.242	-0.243	-0.248	-0.244

5.2 Speed and Acceleration

From the datasheet of the servo, the maximum rotational speed of the servo is 0.16 sec/60°, which corresponds to 6.54 rad/s. Early versions of the algorithm drove the motors at their maximum speed, but this produced a jarring and jerky movement. Moreover, rapid movements of the camera cause the image to blur and impede the facial recognition accuray. For this reason, the speed of the servos were limited in the software. As such, the ServoEasing and AccelStepper libraries were used in Arduino

to add smooth acceleration and deceleration to the eye movements to make them more natural and inviting. This set a software limit to the speed and acceleration of the robot, The speed was set to 80 degrees/sec in software, which corresponds in the real world to 0.98 rad/s. The X servo was also set to this value, but since the microseconds limits were different this corresponds to 0.75 rad/s. There was no acceleration set for the servos to give them a faster response time. In fact, servo motors have a slight wind up naturally as the motor overcomes its stationary inertia, which another reason why software acceleration was deemed unnecessary.

Similar reasoning applied to the stepper motor. The torque curve of the motor provided by the manufacturer indicates that the maximum speed of the motor is around 4000 half steps per second, which corresponds to a rotational speed of 1005 rad/s. Not only is this much too fast for normal operation, but also the torque of a stepper motor drops off at higher speeds, therefore the max speed was set to 3000 step/s in the software, which corresponds to 15.7 rad/s after the 1:6 gear ratio. The maximum acceleration of the stepper motor set to 1000 steps/s^2 to give it an ease-in ease-out animation. This corresponds to a real-world rotational acceleration of 5.24 rad/s^2.

5.3 Time

With these parameters set, the actual output of the robot had to be measured to validate its performance. For this purpose, the time required for the robot to cycle through the full field of vision was recorded. The results of this test are shown in Table 5.2. This was done with a custom script that swept the two stepper motors through their full range of motion and another script that did the same for the stepper motor. Since the eye servos are synchronized in the software such that they will always reach their end points at the same time, both servos were timed simultaneously producing a single measurement, as shown in Table 5.2. The average time for the eye mechanism to sweep from one side to the other was 4.248 seconds. The average time for the motor to sweep from -180° to $+180^\circ$ was 5.936 seconds. Ignoring acceleration,this corresponds to an average rotational velocity of 0.26 rad/s for the Y-Servo and 0.20 rad/s for the X-Servo, which is well within the bounds set in the software. The stepper motor is moving at an average velocity of 1.06 rad/s, which also falls within the maximum value set in the software. This ensures the safety of the user as the robot will not behave unpredictably and move faster than the software tells it to, even when it is making movements over the complete range of motion.

Trial	Servos	Stepper
1	4.28	5.81
2	4.57	6.06
3	4.15	6.17
4	4.13	5.75
5	4.11	5.89
Average	4.248	5.936

Table 5.2: Measurements on the amount of time required for the motors go through a full move.

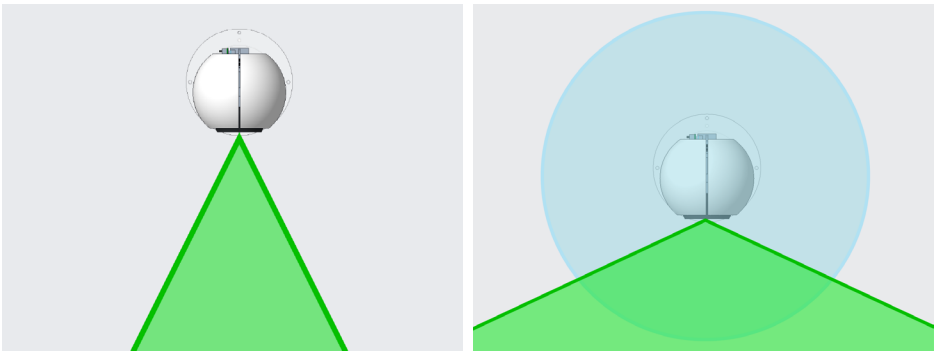
5.4 Field of View

According to B&H Photo Video, the camera has a base field of view of 78 degrees [25]. The camera output is 16:9 ratio, but unfortunately the Python library crops the image to 4:3 for the facial recognition to work, so a test was done to find the resulting field of view after all the software distortions. The facial recognition software was set to run in a loop. I then moved myself to the very edge of the frame, where if I moved even slightly in one direction my face would stop being registered. Then, a marker was placed on the floor directly below me. This was done three times in both directions for a total of 6 measurements. Then, a line was drawn through these points and insersected at a point. The angle between the two lines is the measured field of view of the camera. The field of view was found to be around 60 degrees, shown in Figure 5.2a.

However, with the actuation of the motors, it can be said that this affords the device a much better effective field of view. Since the eye can rotate the camera left and right, this allows the camera to see more of its environment. As such, a similar test was run with the camera panned to its maximum angle left and right, measurements were taken, and the angle between the lines was measured. With the range of motion the effective field of view was measured to be 130°. Figure 5.2b shows this in the green portion. The stepper motor also increases this field of vision. Since it can move through a full 360° range of motion, this increases the effective field of view of the robot to 360°. This can be seen in the diagram as the blue circle.

5.5 Maximum User Movement

There are some limitations on how fast the user can move relative to the robot. The best results are achieved when the user focuses their eyes on the camera and moves



(a) (b)

Figure 5.2: Comparison of the base FOV and the effective FOV with actuation.

slowly around it. The user must not make sudden jerky movements, otherwise the software will not have enough time to catch up to the face. This can be caused by the servos and stepper moving too slowly to track the person. This was tested extensively and it was determined that the speed of the motors was sufficient. The more significant limitation in in the frequency of the software's measurements.

The facial recognition currently runs every 5 frames of video for performance reasons, with 30fps that is 200ms in between measurements. To be continuously tracked by the camera, the user must move slowly enough so that they are not out of frame by the next time the camera takes a measurement. Since the movement on screen is proportional to how far away you are from the camera, results of this calculation will be expressed per meter away from the camera.

For this to happen in normal operation the user would have to be in the center of the camera's field of view in one measurement frame, and out of the field of view in the next measurement frame. Using the field of view of 60 degrees found in Section 5.4, for every meter you move away from the camera, the distance from the center of the image to the right side of the field of view increases by 2.31 meters. Using this and the 30 fps of the camera, the maximum speed a person could move is 11.55 m/s per meter of distance away from the camera. The vertical distance is the same. Of course a user will not be thinking about their exact numerical speed but it gives a good approximation to help the user use the hardware properly.

To mitigate this issue a wide angle lens could be used to cover the blind spots on the sides of the robot. There are lenses that capture a full 360 degree image and then transform it so it looks normal in software. The resulting planar image would then be able to be fed into the facial recognition library.

5.6 Loop Closure Time

A test was done to find the loop closure time of the loop outlined previously in Figure 4.4. The test consisted of standing in front of the camera, moving to a random location, and then standing still and waiting for the robot to start moving. A timer was started every time a movement was made, then stopped the timer once the robot had finished moving. This was repeated 10 times to find an average loop closure time, which is shown in Table 5.3. The average loop closure time was found to be 3.1 seconds, which was deemed an acceptable result. A possible explanation for this value being relatively high is that sometimes the camera overshoots or undershoots the face in the image, so the robot gets to approximately the right spot, but then has to make small adjustments before getting the face into the deadzone of the image and finally stopping all movement. This is this algorithm has a trade-off between response time and accuracy. The camera can respond quickly to a large movement of the user by moving the servos a large amount, but by doing so the camera is making a rough estimate, lowering the immediate accuracy of the face tracking. Fortunately though, the system eventually converges as the face approaches the center of the image. To an average user, this is not a huge issue, a person usually does not stand perfectly still when speaking, so tracking a moving target is prioritized more than achieving a stationary resting position.

Trial	Loop Closure Time (s)
1	2.29
2	2.27
3	1.96
4	3.51
5	2.48
6	2.86
7	3.58
8	3.48
9	4.03
10	3.58
Average	3.1

Table 5.3: Average loop closure time of the robot.

5.7 Power Consumption

To measure the power consumption of the robot, two metrics were collected: the peak current and average current. For the peak current, the custom script from Section 5.3 was reused to drive the motors through their full range of motion repeatedly. A multimeter was connected in series with the 12V power line going into the motherboard. A video was taken and the maximum current value displayed on the multimeter across 30 seconds of operation was recorded. It was found that the peak current of the robot is 0.88A.

The average current was found using a similar method. The current output of the multimeter was recorded every two seconds for a period of 30 seconds. The results can be seen in Figure 5.3. The average current from all these measurements was found to be 0.78A. Using this value, a theoretical battery life for the device can be measured. The battery planned for use would be an 8 pack of rechargeable AA batteries, which would supply a total of 12V and 22.4 Ah. Using the average current measurement it is estimated that the battery life would be 28.79 hours. This is well within a reasonable battery life, so the robot can be used on battery power or plugged into the wall.

Note that battery of laptop running the facial recognition algorithm is neglected, as it is running on its own internal battery and has many background programs running that would make this measurement meaningless.

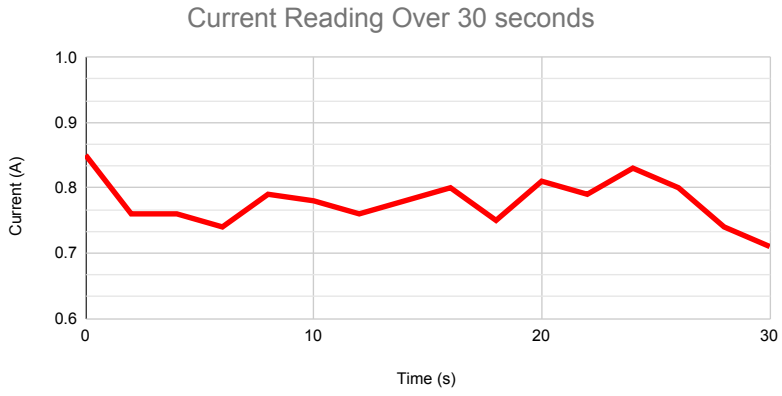


Figure 5.3: Current readings taken every 2 seconds for 30 seconds.

5.8 Facial Recognition Stress Tests

The face tracking algorithm currently only supports tracking a single face. The facial recognition library actually outputs an array of possible faces in the image above a default confidence level of 0.6. Since the array is ordered from most confident to least confident, the first element of the array is used in each iteration to calculate the move commands for the motors. An investigation was conducted on how the robot would behave if multiple people were in frame. Figure 5.4a shows the robot's response to this situation. The algorithm simply picks which face it is most confident in and focuses in on it. This purely depends on the dataset used to train the model used in the library, so this is beyond the scope of this project. Also because the confidence is always changing, sometimes the algorithm decides to rapidly switch between faces unpredictably. A topic of future work could explore different ways of making the robot choose which face to focus on and stay focused on. For instance, the robot could focus on whoever is talking. Two microphones could be installed in the robot to discern the approximate lateral position of the person that is speaking. After a certain threshold of noise is reached the camera could point in that direction. This is a possible solution to this problem, but is out of scope for the current iteration of the robot.

In order for the robot to have consistent operation, the image it receives must be consistent as well. Unfortunately in practice nothing is perfectly consistent, so the algorithm has some issues when running into edge cases. For instance, face tracking performance is diminished in poor lighting conditions. The user has to ensure that their face is fully illuminated and free of obstructions. A test was done to see if wearing sunglasses would be a barrier to the algorithm. An image from this test can be seen in Figure 5.4b. It was found that sunglasses interferes with the facial recognition, so the camera opts to focus on another person in the frame. Skin tone was also an issue in the algorithm, as a darker skinned subject was less likely to be focused on.

5.9 Peer Feedback

Informal focus groups were held to gain peer feedback on the robot. After all, the end goal of the project was to make a friendly robot and make people more willing to interact with it. The responses were generally positive. Groups indicated that they would rather speak to this robot than a box like a Google Home, which demonstrates

(a) Multiple people in frame. (b) Subject wearing sunglasses.

Figure 5.4: Examples of challenges to the facial recognition implementation.

success in that aspect. Negative impressions were given by the people that the robot did not recognize. One participant with darker skin joked that perhaps the robot did not like them, prioritizing another person over themselves. This is unfortunate but ultimately out of this project's control. The main complaint about the robot was the noise. One participant mentioned that if they were in a dark room and heard the growling noise of the stepper motor they would be very frightened. This is a limitation of the components used in this model. Future iterations can add in a more modern stepper driver like the TMC2208 which features silent stepping. People also expected the white part to move with the eye, rather than having the black part be the eyeball and the white part being the head. Perhaps the color scheme of the robot could be another area of future work, where softer colors like brown might be more natural and inviting.

Chapter 6

Conclusion

The project was largely successful, producing a working finished product and receiving generally positive feedback. The results of the project give good evidence to support the hypothesis outlined at the start of the paper; that perhaps there really is something about one-eyed robots that makes them more likeable. Of course, countless challenges and setbacks were met along the way, and numerous things were planned and scrapped or scaled back in order to meet the deadline. The end result is that the robot can track a face, and in that sense this project was a success.

6.1 Project Management

Figure 6.1 shows a summarized timeline of the project, noting the big milestones and dates they were achieved. Some things to note are that the progress was slow for most of the year. By the halfway point in the year, a concrete concept was developed, but this was fell short of the initial goal of having the project physically manufactured by the end of the fall semester to be shown at the PDR presentation. As such, the timeline had to go through a few revisions, especially towards the end of the project. Progress started picking up however around March 1, where after this point the project was worked on in some capacity every day, no matter how little. That little bit was just the starting point needed to get the ball rolling, and the project progressed rapidly after this. It was at this point that a worklog was established to track the hours put into the project not only for documentation purposes but also to provide motivation. Even when hitting setbacks or having to redo things entirely, the hour counter was still going up, more entries were being put into the worklog

and the project was still making progress. Progress is certainly not linear. The full worklog can be found in Appendix E. The total amount of hours spent on this thesis, according to the worklog, is 194 hours, excluding the writing of the report. In the end, all the milestones were met and the robot was functioning as intended.

Date	Milestone
10/24/2022	Initial Design Ideation Complete
11/22/2022	Started Ordering Materials
3/1/2023	First testing with Electrical Components
3/2/2023	Put in big order for materials
3/8/2023	Finalized dimensions of the eye based on camera
3/17/2023	First working prototype of the eyeball subassembly
3/23/2023	Final CAD model assembled with all the parts
3/29/2023	Full Assembly manufactured and assembled, started electronics
4/4/2023	Python face recognition code working
4/7/2023	Figured out how to send/receive serial data with an echo program
4/10/2023	Face following algorithm working
4/11/2023	Physical model finalized
4/19/2023	Finishing aesthetic touches added

Table 6.1: Project timeline.

The project's allocated budget was $1000. This included everything from the raw materials to the tools and components for assembly. One thing to note is that these components are typically sold in bulk, so there are many extra parts left over. Additionally, duplicate parts for the motors and microcontroller were ordered to keep the project on track in case a component got fried. Otherwise, the project would be delayed by an entire week to wait for the replacement component to come in. Speaking of this, another issue was delivery time of the materials; the project simply could not get done in the last minute because of this constraint. As such, many components were ordered before they were even put into the CAD model, leading to some waste when the component was actually unnecessary. Given these challenges however, the project still managed to stay within the budget constraint at a total cost of $981.82. Additionally, costs for a future project would be much less as many of the supplies bought in bulk can be reused, like the fasteners, wires and connectors, tools, and duplicate components. The full bill of materials can be found in Appendix F.

6.2 Future Work

Future work on this project can be divided into hardware and software issues. At this point in the project the hardware issues are the lesser of the two categories, but there are still improvements that can be made. One thing that was an issue was the relatively cumbersome assembly process. Everything was mounted with screws, which meant that a screwdriver had to fit in the area to screw them in. Unfortunately there were some tight spaces in the design and some screws were possible to reach but with great difficulty. For some parts such as the servos the entire assembly would have to be taken apart to remove them since the screws were on the underside of the component. Another thing to change would be to standardize the screw size across all components. Each module was designed separately and pre-bought components all had differently sized screw holes, so a future iteration would only have two or three screw sizes at set lengths to further expedite the assembly process. Time did not allow for another revision to mitigate these issues, so the project had to proceed with these inconveniences. A future iteration would improve the assembly process to the point that anyone could assemble it by themselves. An instruction manual would also be a good step in making this a real product that a consumer could purchase.

Another area that could be explored is different manufacturing methods for the parts. Many parts were 3D printed, but this actually makes them weaker than designed. This is because the layer lines act as stress concentrators, causing the parts to snap along them. This happened multiple times and had to be fixed on the fly with superglue. A future project could explore a fully sheetmetal chassis or more sophisticated plastics manufacturing processes like injection molding.

Moving onto the electronics, one of the main limitations of the current robot is that it requires a laptop to run all the facial recognition code. A future iteration of the robot would be standalone, with something like a Raspberry Pi to run the facial recognition algorithm. This was not implemented on this project due to time constraints, but a final product would certainly be able to run standalone.

On the software side, there is arguably the most room for expansion. Now that the eye mechanism is working, the software of the robot can be expanded on to make it more than a proof of concept. A microphone and speaker were incorporated into the design but were not utilized to their full potential. In the future, more sophisticated software could be written to integrate ChatGPT into the robot to make it possible to hold an actual conversation with the robot and speak to it just as one would to another person. This would be facilitated with the eye contact from the eye

mechanism. This would turn this robot into a more marketable product to create a truly useful personal assistant robot.

Bibliography

[1] P. M. Research. "Household Robots Market Size Global Report, 2022 - 2030," Polaris. (Nov. 2022), [Online]. Available: https://www.polarismarketresearch.com/ (visited on 04/19/2023).

[2] C. B. CIS. "Human-Robot Interaction," Cornell Information Science. (2023), [Online]. Available: https://infosci.cornell.edu/research/human-robot-interaction (visited on 04/26/2023).

[3] L. S. Smoot, K. M. Bassett, and M. Hammond, "Animatronic eye with an electromagnetic drive and fluid suspension and with video capability," U.S. Patent 10179040B2, Jan. 15, 2019. [Online]. Available: https://patents.google.com/patent/US10179040B2/en?q=(Eye+mechanism+for+animatronics+character)&oq=Eye+mechanism+for+animatronics+character (visited on 04/26/2023).

[4] DJI. "Osmo Mobile 6 - Unfold Your Creativity," DJI. (2022), [Online]. Available: https://www.dji.com/photo (visited on 04/20/2023).

[5] H. Robotics. "Sophia," Hanson Robotics. (2023), [Online]. Available: https://www.hansonrobotics.com/sophia/ (visited on 04/20/2023).

[6] E. Arts. "Ameca," Engineered Arts. (2023), [Online]. Available: https://www.engineeredarts.co.uk/robot/ameca/ (visited on 04/26/2023).

[7] Masahiro Mori and Karl F. Macdorman, "The Uncanny Valley: The Original Essay by Masahiro Mori - IEEE Spectrum," *IEEE Robotics & Automation Magzine*, June 2012 Jun. 12, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:209316176.

[8] D. J. Harrigan, director, *Portal 2 - Wheatley in Real Life - Mk II Design*, Jan. 28, 2022. [Online]. Available: https://www.youtube.com/watch?v=oLZX8REK-vs (visited on 04/17/2023).

[9] A. Terranova. "This Adorable Robotic Pixar Lamp Recognizes Your Face," Make: DIY Projects and Ideas for Makers. (Aug. 25, 2015), [Online]. Available: https://makezine.com/article/technology/robotics/pixar-robotic-desk-lamp/ (visited on 04/17/2023).

[10] W. Cogley. "DIY Compact 3D Printed Animatronic Eye Mechanism," Instructables. (Nov. 19, 2019), [Online]. Available: https://www.instructables.com/DIY-Compact-3D-Printed-Animatronic-Eye-Mechanism/ (visited on 04/17/2023).

[11] PJRC. "Teensy® 4.0," PJRC. (2023), [Online]. Available: https://www.pjrc.com/store/teensy40.html (visited on 04/26/2023).

[12] B. Patton. "PRT_28 Patton Robotics Teensy Motherboard Kit," Patton Robotics. (2018), [Online]. Available: https://pattonrobotics.com/products/patton-robotics-teensy-motherboard-kit (visited on 04/26/2023).

[13] McMaster-Carr. "Metal Gear - 20 Degree Pressure Angle," McMaster-Carr. (), [Online]. Available: https://www.mcmaster.com/ (visited on 04/26/2023).

[14] McMaster-Carr. "Metal Internal Gear - 20 Degree Pressure Angle," McMaster-Carr. (), [Online]. Available: https://www.mcmaster.com/ (visited on 04/26/2023).

[15] A. Bowyer. "StepStick - RepRap." (Jun. 2, 2020), [Online]. Available: https://reprap.org/wiki/StepStick (visited on 04/26/2023).

[16] *Opencv/opencv*, OpenCV, Apr. 26, 2023. [Online]. Available: https://github.com/opencv/opencv (visited on 04/26/2023).

[17] A. Geitgey, *Face Recognition*, Apr. 26, 2023. [Online]. Available: https://github.com/ageitgey/face_recognition (visited on 04/26/2023).

[18] PowerBroker2, *pySerialTransfer*, Apr. 8, 2023. [Online]. Available: https://github.com/PowerBroker2/pySerialTransfer (visited on 04/21/2023).

[19] PB2, *SerialTransfer*, Apr. 20, 2023. [Online]. Available: https://github.com/PowerBroker2/SerialTransfer (visited on 04/22/2023).

[20] Armin, *ServoEasing*, Apr. 2, 2023. [Online]. Available: https://github.com/ArminJo/ServoEasing (visited on 04/22/2023).

[21] "AccelStepper: AccelStepper library for Arduino." (), [Online]. Available: https://www.airspayce.com/mikem/arduino/AccelStepper/ (visited on 04/22/2023).

[22] ansh2919. "Serial Communication between Python and Arduino," Hackter. (2021), [Online]. Available: https://www.hackster.io/ansh2919/serial-communication-between-python-and-arduino-e7cce0 (visited on 04/21/2023).

[23] H. Dethe. "Face Tracking OpenCV, Python, & Arduino," Learn Robotics. (Jan. 30, 2019), [Online]. Available: https://www.learnrobotics.org/blog/face-tracking-opencv/ (visited on 04/17/2023).

[24] Robin2. "Serial Input Basics - updated," Arduino Forum. (Apr. 25, 2016), [Online]. Available: https://forum.arduino.cc/t/serial-input-basics-updated/382007 (visited on 04/21/2023).

[25] B. P. Video. "Logitech C615 Webcam," B&H Photo Video. (), [Online]. Available: https://www.bhphotovideo.com/c/product/795734-REG/Logitech_960_000733_C615_HD_Webcam.html (visited on 04/23/2023).

[26] O. Verdier, *Python highlighting in LaTeX*, Apr. 25, 2023. [Online]. Available: https://github.com/olivierverdier/python-latex-highlighting (visited on 04/26/2023).

[27] trihedral, *ArduinoLatexListing*, Oct. 3, 2017. [Online]. Available: https://github.com/trihedral/ArduinoLatexListing (visited on 04/17/2023).

Appendix A

Videos

Eyeball Mechanism: https://youtu.be/FsiWPOwoMAg
Early Prototype Test: https://youtu.be/uXpVkPctlw8
Full Demonstration: https://youtu.be/q94V7J1bo_w

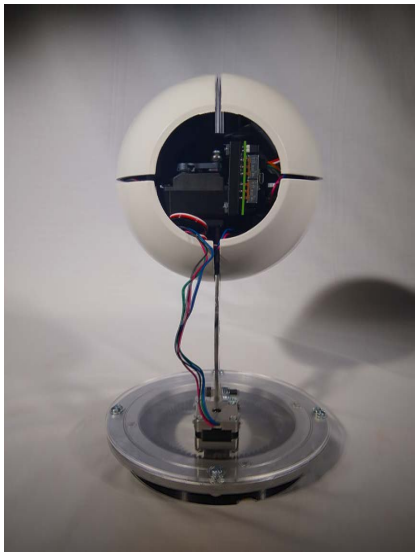
Appendix B

Additional Photos

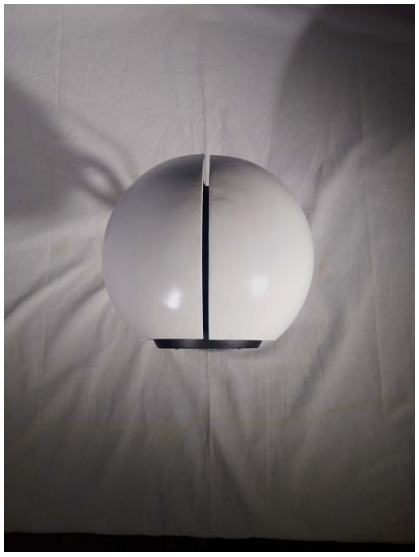
Figure B.1: Photos of model with dark background.

(a) Right View

(b) Left View

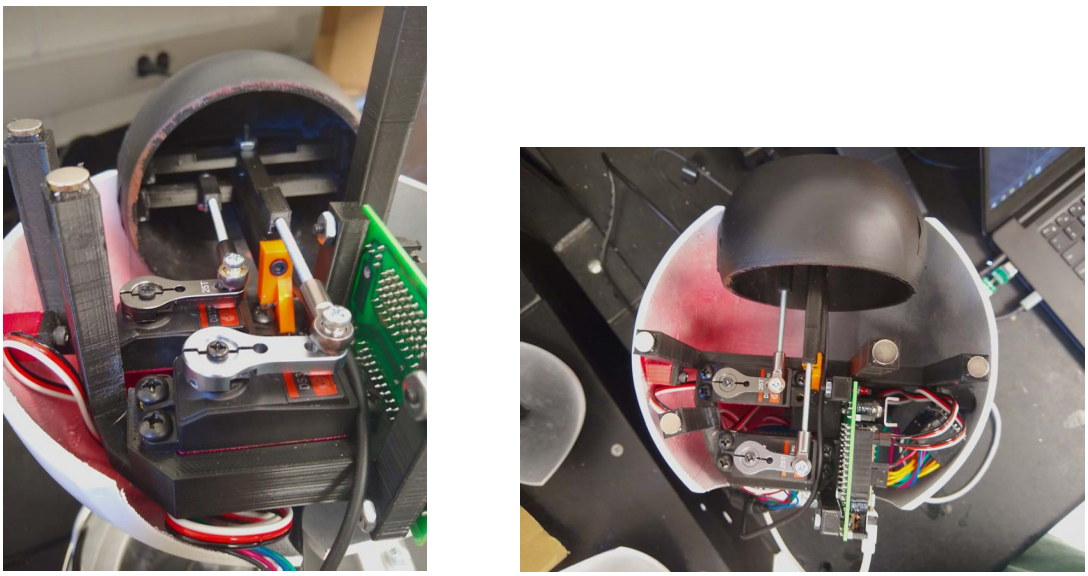


(c) Back View



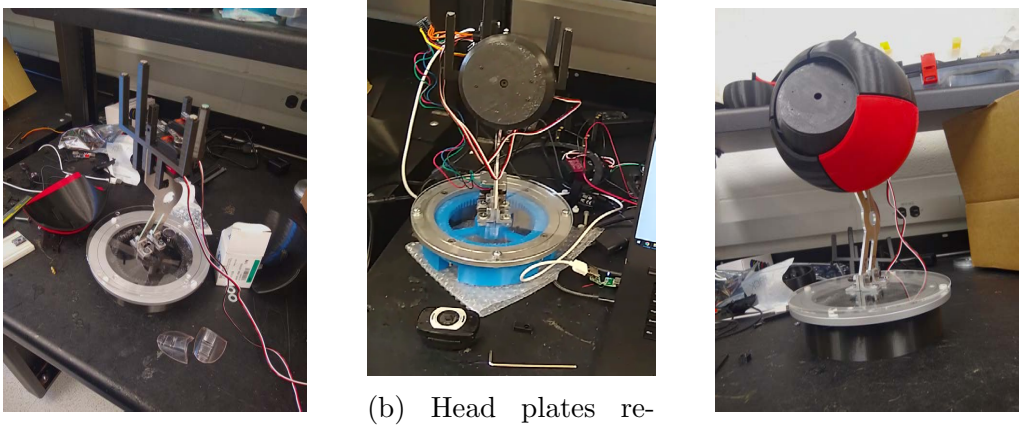
(d) Top View

Figure B.2: Various angles of the robot.



(a) (b)

Figure B.3: Additional photos of the inner workings inside the head.



(a) Core Skeleton. (b) Head plates removed. (c) Before painting.

Figure B.4: Various photos of the intermediary steps in making the robot.

Appendix C

Python Code

Code coloring by python-latex-highlighting [26]

```
# -*- coding: utf-8 -*-
"""
@author: Jonathan Melkun

@description: Sends servo and stepper move commands to the teensy
"""
import os
os.environ["OPENCV_VIDEOIO_MSMF_ENABLE_HW_TRANSFORMS"] = "0"
import time
import cv2
import face_recognition
from pySerialTransfer import pySerialTransfer as txfer

def mapA(x, in_min, in_max, out_min, out_max):
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) +
                                        out_min;

class struct(object):
    servo1_move = 0
    servo2_move = 0
    stepper_move = 0
centering_loop = False
video_capture = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

```
print("Video Capture Success.")
face_locations = []
frameNum = 0
servo_step_x = 0
servo_step_y = 0
width = 640
height = 480
video_capture.set(cv2.CAP_PROP_FRAME_WIDTH, width)
video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT, height)
DEADZONE = 0.06
DEADZONE_X = int(width*DEADZONE)
DEADZONE_Y = int(height*DEADZONE)
centerImgX=width//2
centerImgY=height//2

try:
    packet = struct
    link = txfer.SerialTransfer('COM7')

    link.open()
    time.sleep(2)
    print("Link is open.")
    while True:
        packet.servo1_move = 0
        packet.servo2_move = 0
        packet.stepper_move = 0
        ret, frame = video_capture.read()
        # Only process every tenth frame of video
        if frameNum % 5 == 0:
            # Convert the image from BGR color (which OpenCV uses) to
                                                RGB color (which
                                                face_recognition uses)
            rgb_frame = frame[:, :, ::-1]

            # Find all the faces in the current frame of video
            face = face_recognition.face_locations(rgb_frame)
```

```
        if len(face) != 0:
            centerX = (face[0][3]+face[0][1])//2
            centerY = (face[0][0]+face[0][2])//2
            if centerX < width//2 - DEADZONE_X or centerX > width//2
                                                + DEADZONE_X or
                                               centerY < height//2
                                               - DEADZONE_Y or
                                               centerY > height//2
                                               + DEADZONE_Y:


                packet.servo2_move = int((centerX-centerImgX)/7.98)
                packet.servo1_move = int((centerY-centerImgY)/7.28*0
                                                    .7)
                packet.stepper_move = int((centerX-centerImgX)/0.244
                                                    *0.3)

                #send to Arduino
                sendSize = 0
                sendSize = link.tx_obj(packet.servo1_move, start_pos
                                                    =sendSize)
                sendSize = link.tx_obj(packet.servo2_move, start_pos
                                                    =sendSize)
                sendSize = link.tx_obj(packet.stepper_move,
                                                    start_pos=
                                                    sendSize)
                link.send(sendSize)
                #print("{}, {}, {}, {}".format(packet.servo1_move,
                                                    packet.
                                                    servo2_move,
                                                    packet.
                                                    stepper_move,
                                                    packet.centering
                                                    ))
                #print("Data sent.\n")
            else:
```

```
            print("deadzone")
    frameNum+=1

    # Display the results
    if len(face) != 0:
        # Draw a box around the face
        cv2.rectangle(frame, (face[0][3], face[0][0]),(face[0][1],
                                                face[0][2]), (0, 0, 255)
                                                , 2)
    # Display the resulting image
    cv2.rectangle(frame, (width//2+DEADZONE_X, height//2+DEADZONE_X)
                                            , (width//2-DEADZONE_Y,
                                            height//2-DEADZONE_Y), (0,
                                            255, 0), 2)
    cv2.namedWindow('Video', cv2.WINDOW_GUI_NORMAL)
    cv2.imshow('Video', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        video_capture.release()
        cv2.destroyAllWindows()
    #receive information from Arduino for debugging
    # while(link.available()):
    #     print("Received data!")
    #     recSize = 0

    #     servo1_rec = link.rx_obj(obj_type='i', start_pos=recSize)
    #     recSize += txfer.STRUCT_FORMAT_LENGTHS['i']

    #     servo2_rec = link.rx_obj(obj_type='i', start_pos=recSize)
    #     recSize += txfer.STRUCT_FORMAT_LENGTHS['i']

    #     stepper_rec = link.rx_obj(obj_type='i', start_pos=recSize)
    #     recSize += txfer.STRUCT_FORMAT_LENGTHS['i']

    #     centering_rec = link.rx_obj(obj_type='b', start_pos=
                                            recSize)
    #     recSize += txfer.STRUCT_FORMAT_LENGTHS['b']
```

```
        #     print('{}, {}, {}, {}'.format(servo1_rec, servo2_rec,
                                              stepper_rec, centering_rec))
except KeyboardInterrupt:
    print('KeyboardInterrupt, closing')
    link.close()
    # Release handle to the webcam
    video_capture.release()
    cv2.destroyAllWindows()

except:
    import traceback
    traceback.print_exc()
    # Release handle to the webcam
    video_capture.release()
    cv2.destroyAllWindows()
    link.close()
```

Appendix D

Arduino Code

Code output by ArduinoLatexListing [27]

```
#include "SerialTransfer.h"
#include "ServoEasing.hpp"
#include "SerialTransfer.h"
#include "AccelStepper.h"

#define servo1_pin 14
#define servo2_pin 15

#define stepPin 22
#define dirPin 21
bool not_moved_yet = false;
int maxStepperPosition = 9560/2;
int minStepperPosition = -9560/2;
int servoSpeed = 80;

ServoEasing Servo1;
ServoEasing Servo2;
SerialTransfer myTransfer;
AccelStepper stepper(1, stepPin, dirPin);
```

```
20
21 struct __attribute__((packed)) STRUCT {
22   int servo1_move = 0;
23   int servo2_move = 0;
24   int stepper_move = 0;
25 } packet;
26
27 //used for debugging since I apparently cannot print
     anything other than a struct
28 struct __attribute__((packed)) STRUCT2 {
29   int servo1_move = 333;
30   int servo2_move = 333;
31   int stepper_move = 333;
32 } debugStruct;
33
34 void setup()
35 {
36   Servo1.attach(servo1_pin, 90, 1200, 1670);
37   Servo2.attach(servo2_pin, 90, 1320, 1680);
38   setSpeedForAllServos(servoSpeed);
39   Servo1.setEasingType(EASE_LINEAR);
40   Servo2.setEasingType(EASE_LINEAR);
41
42   stepper.setMaxSpeed(3000);
43   stepper.setAcceleration(1000);
44   stepper.setCurrentPosition(0);
45   Serial.begin(115200);
46   myTransfer.begin(Serial);
47   while (!Serial) {
48     ; // wait for serial port to connect. Needed for native
```

```
        USB
49    }
50  }
51
52  void loop()
53  {
54    if(myTransfer.available())
55    {
56      uint16_t recSize = 0;
57      recSize = myTransfer.rxObj(packet, recSize);
58      not_moved_yet=true;
59      /*
60      //for debugging, sends received packet back to Python
61      uint16_t sendSize = 0;
62      sendSize = myTransfer.txObj(packet, sendSize);
63      myTransfer.sendData(sendSize);
64      */
65    }
66
67    if (not_moved_yet) {
68      // uses non-blocking for theoretical simultaneous
          movement of servos and stepper
69      if (!Servo1.isMoving()){
70        Servo1.setEaseTo(Servo1.getCurrentAngle()+packet.
            servo1_move);
71        Servo2.setEaseTo(Servo2.getCurrentAngle()+packet.
            servo2_move);
72        setEaseToForAllServosSynchronizeAndStartInterrupt(
            servoSpeed);
73      }
```

```
74
75      if (stepper.currentPosition()+packet.stepper_move <
           maxStepperPosition && stepper.currentPosition()+
           packet.stepper_move > minStepperPosition) {
76        stepper.move(packet.stepper_move);
77      }
78      not_moved_yet=false;
79    }
80    stepper.run();
81  }
82  }
```

Appendix E

Full Worklog

Date	Time Start	Time End	Description	Duration	Total Hrs
9/16/2022	8:00:00 AM	9:00:00 AM	first nosenchuck meeting	1:00:00	1:00:00
9/29/2022			thesis proposal doc	0:21:00	1:21:00
10/6/2022	5:38:00 PM	5:43:00 PM	first eye prt file	0:05:00	1:26:00
10/25/2022	9:00:00 PM	12:00:00 AM	looking for inspiration in engineering library	3:00:00	4:26:00
10/25/2022	1:08:00 PM	1:21:00 PM	old draft	0:13:00	4:39:00
10/26/2022			meeting notes doc	0:38:00	5:17:00
10/28/2022	9:30:00 AM	10:00:00 AM	nosenchuck meeting	0:30:00	5:47:00
11/11/2022	8:30:00 AM	9:00:00 AM	nosenchuck meeting	0:30:00	6:17:00
11/18/2022	8:30:00 AM	9:00:00 AM	nosenchuck meeting	0:30:00	6:47:00
12/2/2022	8:30:00 AM	9:00:00 AM	nosenchuck meeting	0:30:00	7:17:00
12/8/2022	11:21:00 AM	12:45:00 PM	PDR presentation	1:24:00	8:41:00
12/8/2022	4:36:00 PM	5:01:00 PM	PDR presentation	0:25:00	9:06:00
12/9/2022	8:30:00 AM	9:30:00 AM	nosenchuck meeting	1:00:00	10:06:00
3/1/2023	8:00:00 PM	12:30:00 AM	fried the teensy 4.1	4:30:00	14:36:00
3/2/2023	10:00 PM	12:00:00 AM	started looking for materials	2:00:00	16:36:00
3/2/2023	8:35:00 AM	9:41:00 AM	BoM sheet	1:06:00	17:42:00
3/2/2023	11:26:00 AM	1:16:00 PM	BoM sheet	1:50:00	19:32:00
3/2/2023	4:55:00 PM	5:49:00 PM	BoM sheet	0:54:00	20:26:00
3/3/2023	4:30:00 PM	4:40:00 PM	BoM sheet	0:10:00	20:36:00
3/5/2023	9:00:00 AM	12:00:00 PM	started modeling the eye, settled on dimensions	3:00:00	23:36:00
3/5/2023	8:00:00 PM	9:00:00 PM		1:00:00	24:36:00
3/5/2023	3:34:00 PM	5:03:00 PM	laid out components	1:29:00	26:05:00
3/6/2023	4:20:00 PM	4:27:00 PM	More BoM and ordering	0:07:00	26:12:00
3/6/2023	9:12:00 PM	11:39:00 PM	put the battery holder in the assembly	2:27:00	28:39:00
3/7/2023	10:45:00 PM	1:24:00 AM	modeling linkages based on instructables	2:39:00	31:18:00
3/8/2023	8:53:00 PM	12:00:00 AM	eyeball_asm_draft1 (first draft with the webcam determining the dimensions)	3:07:00	34:25:00
3/8/2023	8:24:00 PM	11:42:00 PM	refining model up to making stl for printing the next day	3:18:00	37:43:00
3/9/2023	9:51:00 PM	11:30:00 AM	nozzle 0.4mm troubleshooting	13:39:00	51:22:00
3/9/2023	12:45:00 PM	1:30:00 PM	first 3d print started (failed later at night)	0:45:00	52:07:00
3/9/2023	8:48:00 PM	10:34:00 PM	modifying main support	1:46:00	53:53:00

3/11/2023	10:26:00 PM	10:52 PM	Ordering components	0:26:00	54:19:00
3/15/2023	7:39:00 PM	12:25:00 AM	cleaned up first printed parts and figured out 5-40 screws work well as pins	4:46:00	59:05:00
3/16/2023	11:15:00 AM	1:52:00 PM	worked in conference room on support block	2:37:00	61:42:00
3/16/2023	3:15:00 PM	8:45:00 PM	finished up support block and set it up to print	5:30:00	67:12:00
3/17/2023	10:45:00 AM	11:45:00 AM	picked up printed parts and cleaned them	1:00:00	68:12:00
3/17/2023	2:00:00 PM	6:00:00 PM	assembled and printed new parts	4:00:00	72:12:00
3/17/2023	7:50:00 PM	11:31:31 PM	created first full prototype	3:41:31	75:53:31
3/18/2023	10:00:00 PM	12:00:00 AM	sorting out time sheet	2:00:00	77:53:31
3/19/2023	7:57:19 PM	1:00:00 AM	got the camera screw holes correct (yes it took 5 hours)	5:02:41	82:56:11
3/20/2023	2:00:00 PM	3:00:00 PM	cleaned up head v2 parts, reprinted eye because of wrong dimensioning on camera holes	1:00:00	83:56:11
3/20/2023	4:20:00 PM	6:39:30 PM	designed new servo mount	2:19:30	86:15:41
3/20/2023	8:03:59 PM	11:32:43 PM	decided color scheme, modeled outer shell, made support block connections	3:28:43	89:44:24
3/21/2023	12:11:30 PM	1:08:05 PM	worked on body	0:56:35	90:40:59
3/21/2023	9:45:31 PM	12:57:59 AM	finished body, placed baseplate, almost finished the servo mount block, still have to work out how to screw shell pieces	3:12:29	93:53:27
3/22/2023	1:04:04 PM	6:03:50 PM	wiring for slip ring, body connector to base	4:59:47	98:53:14
3/22/2023	7:31:56 PM	8:54:15 PM	started on baseplate with ball bearing	1:22:19	100:15:33
3/23/2023	10:30:00 PM	12:15:09 AM	discovered lazy susan bearing, offset assembly from center of gravity to make space for slip ring	1:45:09	102:00:42
3/24/2023	1:00:00 PM	2:30:00 PM	CNC'd robot body with the help of Al	1:30:00	103:30:42
3/24/2023	3:30:00 PM	6:30:00 PM	did some stepper motor modeling, holes in acrylic plate	3:00:00	106:30:42
3/25/2023	1:30:00 PM	5:30:00 PM	gear modeling	4:00:00	110:30:42
3/25/2023	7:30:00 PM	9:00:00 PM	modeled and printed base with gear	1:30:00	112:00:42
3/26/2023	8:00:00 PM	9:00:00 PM	widened holes for magnets	1:00:00	113:00:42
3/28/2023	8:32:28 AM	9:40:24 AM	magnet tests, reprinting	1:07:56	114:08:38
3/28/2023	10:10:00 AM	11:40:00 AM	laser cut the baseplate	1:30:00	115:38:38
3/29/2023	8:30:00 AM	9:30:00 AM	made drawing for brackets	1:00:00	116:38:38
3/29/2023	12:30:00 PM	2:55:00 PM	milled and drilled brackets	2:25:00	119:03:38

3/29/2023	4:20:00 PM	6:30:00 PM	countersunk and cut screws and glued magnets	2:10:00	121:13:38
3/29/2023	7:30:00 PM	11:15:00 PM	assembled full assembly for the first time	3:45:00	124:58:38
3/31/2023	3:30:00 PM	6:30:00 PM	crimping slipring	3:00:00	127:58:38
4/3/2023	4:30:00 PM	12:47:00 AM	started python coding, gave up on slipring	8:17:00	136:15:38
4/4/2023	7:45:00 AM	9:30:00 AM		1:45:00	138:00:38
4/4/2023	4:45:00 PM	6:00:00 PM		1:15:00	139:15:38
4/4/2023	7:30:00 PM	10:07:37 PM	got face_recognition working in python	2:37:37	141:53:16
4/5/2023	8:45:00 PM	9:50:00 PM	accelstepper investigating	1:05:00	142:58:16
4/5/2023	2:00:00 PM	3:00:00 PM	serial coding	1:00:00	143:58:16
4/5/2023	9:00:00 PM	11:40:00 PM	got stepper moving	2:40:00	146:38:16
4/6/2023	9:00:00 AM	9:50:00 AM	accelstepper working	0:50:00	147:28:16
4/6/2023	8:00:00 PM	11:45:00 PM	pySerialTransfer discovery	3:45:00	151:13:16
4/7/2023	1:00:00 PM	2:30:00 PM	pySerialTransfer echo program	1:30:00	152:43:16
4/8/2023	9:00:00 AM	12:30:00 PM	servo send/receive script	3:30:00	156:13:16
4/8/2023	1:00:00 PM	7:16:43 PM	used struct printing to diagnose why the servos were skipping because I can't print anything else out	6:16:43	162:29:59
4/8/2023	7:45:00 PM	11:53:37 PM	fried my teensy	4:08:37	166:38:36
4/9/2023	10:30:00 AM	12:30:00 PM	actually teensy magically started working again, coding	2:00:00	168:38:36
4/10/2023	2:00:00 PM	6:30:00 PM	got face following code working with servos	4:30:00	173:08:36
4/10/2023	10:00:00 PM	12:15:00 AM	gluing on eye support	2:15:00	175:23:36
4/11/2023	8:00:00 AM	9:00:00 AM	nosenchuck meeting, finished physical model	1:00:00	176:23:36
4/16/2023	11:00:00 AM	1:00:00 PM	added back old step stick, Bondo prepped parts for sanding	2:00:00	178:23:36
4/16/2023	1:30:00 PM	11:45:00 PM	sanding and priming	10:15:00	188:38:36
4/17/2023	4:30:00 PM	6:30:00 PM	final assembly	2:00:00	190:38:36
4/18/2023	7:00:00 PM	10:08:40 PM	assembly and testing	3:08:40	193:47:16

Appendix F

Bill of Materials

Description	Quantity	Unit Price	BoM Price	Vendor	Link
Teensy 4.1	1	$44.88	$44.88	Tinkersphere	https://tinkersphere.com/teens
20KG Digital Servo DS3218MG	5	$15.99	$79.95	Amazon	https://www.amazon.com/ANN
Logitech HD Laptop Webcam C615	1	$29.99	$29.99	Amazon	https://www.amazon.com/Logi
HATCHBOX 1.75mm Black PLA 3D Printer Filament 1 KG Spoo	1	$24.99	$24.99	Amazon	https://www.amazon.com/HAT
KOOKYE 2PCS Mini Servo Motor 360 Degree Continuous Rota	2	$15.99	$31.98	Amazon	https://www.amazon.com/KOC
Teensy 4.0 Development Board	3	$26.80	$80.40	PJRC	https://www.pjrc.com/store/tee
Slip Ring Collector Ring (3 Wires 5A Diameter 22mm)	1	$20.56	$20.56	Amazon	https://www.amazon.com/Taid
22 awg Electrical Wire [Black 100ft Red 100ft]	1	$16.98	$16.98	Amazon	https://www.amazon.com/Silic
Acrylic Sheet 12" x 24" x 1/8"	1	$17.70	$17.70	McMaster	https://www.mcmaster.com/85
6061 Aluminum Sheet 1/8" Thick, 6" x 12"	1	$23.07	$23.07	McMaster	https://www.mcmaster.com/89
Spray Paint, 12 Oz, Flat Black, 6 Pack	1	$41.88	$41.88	Amazon	https://www.amazon.com/Rust
Gloss White Spray Paint 12oz	1	$41.88	$41.88	Amazon	https://www.amazon.com/Rust
Automotive 2-in-1 Filler & Sandable Primer	6	$7.88	$47.28	Amazon	https://www.amazon.com/Rust
Bondo Glazing and Spot Putty	1	$15.56	$15.56	Amazon	https://www.amazon.com/Bond
Heavy Duty Neodymium Magnets, 6 Different Size, 255Pcs	1	$16.99	$16.99	Amazon	https://www.amazon.com/TRY
620pcs 2.54mm/0.1" Connectors	1	$10.39	$10.39	Amazon	https://www.amazon.com/CHE
Tapered Heat-Set Inserts for Plastic 8-32 Thread Size, 0.185" In	1	$19.49	$19.49	McMaster	https://www.mcmaster.com/93
Black Oxide Steel Pan Head Phillips Screw 8-32 Thread Size, 1	1	$8.88	$8.88	McMaster	https://www.mcmaster.com/92
Low-Strength Steel Hex Nut Zinc-Plated, 8-32 Thread Size	1	$1.92	$1.92	McMaster	https://www.mcmaster.com/90
PRT_28 Patton Robotics Teensy Motherboard Kit	1	$25.00	$25.00	Patton Robotics	https://pattonrobotics.com/proc
USB Isolator Board	1	$9.90	$9.90	Amazon	https://www.amazon.com/Isola
2.1x5.5mm Male DC Power Plug to 9V Battery Button Connecto	1	$7.89	$7.89	Amazon	https://www.amazon.com/Hxch
6 x AA Battery Holder with Standard snap Connector 9V Output	1	$5.99	$5.99	Amazon	https://www.amazon.com/Corp
Rechargeable AA Batteries with Charger	1	$28.99	$28.99	Amazon	https://www.amazon.com/Rech
M3 Metal Ball End Head Holder Tie for 1/10 RC D90 Axial SCX1	1	$10.99	$10.99	Amazon	https://www.amazon.com/10Pc
Low-Strength Zinc-Plated Steel Threaded Rod M3 x 0.5 mm Thr	1	$6.32	$6.32	McMaster	https://www.mcmaster.com/94
Loctite Super Glue for Plastic, Wood, Metal, , Quick Dry - 0.14 fl	2	$4.36	$8.72	Amazon	https://www.amazon.com/Locti
Anker 565 11-in-1 USB C Hub	1	$64.99	$64.99	Amazon	https://www.amazon.com/Anke
Black Oxide Steel Pan Head Phillips Screw 5-40 Thread Size, 5	1	$11.59	$11.59	McMaster	https://www.mcmaster.com/92
Low-Strength Steel Hex Nut Zinc-Plated, 5-40 Thread Size	1	$3.98	$3.98	McMaster	https://www.mcmaster.com/90
26 AWG Stranded Wire Kit Spool 25ft	1	$15.99	$15.99	Amazon	https://www.amazon.com/Ferm
Mako Driver Kit + Precision Tweezer Set Bundle	1	$49.98	$49.98	Amazon	https://www.amazon.com/Mak
Songhe Dual-axis XY Joystick Module PS2 Game Joystick	1	$8.99	$8.99	Amazon	https://www.amazon.com/Joys

Steel Pan Head Phillips Screw M3 x 0.5 mm Thread, 12 mm Lor	1	$4.32	$4.32	McMaster	https://www.mcmaster.com/92(
DC 12V 5A Power Supply Adapter	1	$11.99	$11.99	Amazon	https://www.amazon.com/ALIT
Electrical Slip Ring Collector Ring (8 Wires 5A Diameter 22mm)	1	$32.26	$32.26	Amazon	https://www.amazon.com/gp/p
Black-Oxide Alloy Steel Socket Head Screw 1/4"-20 Thread Size	1	$12.36	$12.36	McMaster	https://www.mcmaster.com/91
Low-Strength Steel Hex Nut Grade 2, Zinc-Plated, 1/4"-20 Threa	1	$4.58	$4.58	McMaster	https://www.mcmaster.com/90
Multipurpose 6061 Aluminum 90 Degree Angle with Round Edge	1	$3.72	$3.72	McMaster	https://www.mcmaster.com/89
Turntable Accessories 1 PCS Metal Lazy Susan Hardware Rotat	1	$13.28	$13.28	Amazon	https://www.amazon.com/Gee
STEPPERONLINE Nema 14 Motor Step Motor 1.8deg Bipolar 1	1	$18.53	$18.53	Amazon	https://www.amazon.com/Motc
10Pcs USB 2.0 Breakout Board, MELIFE USB to DIP Adapter 2.	1	$6.99	$6.99	Amazon	https://www.amazon.com/Brea
10Pcs USB to DIP Board, MELIFE USB Type A Male Plug to DIF	1	$6.99	$6.99	Amazon	https://www.amazon.com/dp/B
Mini USB Stereo Speaker	2	$10.95	$21.90	Sparkfun	https://www.sparkfun.com/proc
TMC2208SILENTSTEPSTICK	1	$10.34	$10.34	Digikey	https://www.digikey.com/en/pr
V5618A Heat Sink	1	$0.47	$0.47	Digikey	https://www.digikey.com/en/pr
		Total:	$981.82		